Social Network Analysis and Mining (2026) 16:24
https://doi.org/10.1007/s13278-025-01566-0

ORIGINAL ARTICLE

A multi-rounded adversarial scenario for graph-based promo fraud detection

Hafizh Adi Prasetya[1,2] · **Xin Liu**[2] · **Tsuyoshi Murata**[1,2] · **Akiyoshi Matono**[2]

Received: 16 June 2025 / Accepted: 3 December 2025

Abstract
Developing reliable models for detecting fraud is increasingly vital amidst the digitalization of our financial institutions and activities. While graph-based fraud detection models showed promising performances in recent years, the underpinning fraud definition and evaluation process they utilized rely heavily on a limited set of labeled datasets. This raises concerns about their vulnerability to distributional shifts and adversarial strategies often exhibited by real-life fraudsters. In response, we propose a novel scenario for graph fraud detection called Multi-round Adversarial Fraud Detection. Here, the fraud detection model is trained and evaluated iteratively on an adversarially evolving graph. This scenario more closely mimics fraud detection activities in real life while being less reliant on the underlying datasets since the graph evolution is induced by generator functions rooted in common fraud behavior. We show that existing models struggle to achieve good multi-round performance under the proposed scenario with F1 scores that consistently hover below 56 percent on subsequent rounds. To improve the aforementioned performance we propose Temporally Pre-trained Node Embedder (TPNE), a module that leverages self-supervised pre-training approach, explicitly separating and enhancing temporal information across multiple rounds. TPNE is both model and label-agnostic, improving the best-performing baseline by up to 4.6 percent in on-round F1 score and up to 32.9 percent in final recall.

Keywords Graph learning · Fraud detection · Continual learning · Self-supervised learning

1 Introduction

The prevalence of fraud is a vital problem for our digitalized financial institutions. Cumulative global merchant losses from online payment fraud are projected to exceed $362 billion for 2023–2028 (Malone 2023). Furthermore, annual loss from promo fraud-the exploitation of promotional incentives outside their intended use, reaches $89 billion (Ekata 2023) for US retailers alone. Acknowledging the far-reaching nature of fraud, it is then necessary to examine the issue of fraud detection through different approaches and lenses. Most recently, the usage of Graph Neural Networks (GNNs) is becoming increasingly popular (Wang 2010). Indeed, most financial activities are naturally graph-structured, where nodes represent the actors and edges represent their transactions. Results of GNN-based detectors designed for both specific (Hyun et al. 2023) and general fraud-like behavior (Dou et al. 2020; Liu et al. 2021b; Shi et al. 2022; Li et al. 2022c; Xu et al. 2024a; Zhuo et al. 2024) show that the general effectiveness of GNNs extends to fraud detection.

Existing GNN-based fraud detectors are designed around certain fraud-like characteristics like camouflaging (Dou et al. 2020), label imbalance (Liu et al. 2021b; Zhuo et al. 2024; Hyun et al. 2024), or heterophily (Shi et al. 2022; Wu et al. 2023a). However, the definition of fraud assumed in their developments is often unclear. Despite assuming

✉ Hafizh Adi Prasetya
hafizh@net.comp.isct.ac.jp

Xin Liu
xin.liu@aist.go.jp

Tsuyoshi Murata
murata@comp.isct.ac.jp

Akiyoshi Matono
a.matono@aist.go.jp

[1] Department of Computer Science, Institute of Science Tokyo, Ookayama, Meguro, Tokyo 152-8550, Japan

[2] Intelligent Platforms Research Institute, National Institute of Advanced Industrial Science and Technology, Aomi, Koto, Tokyo 135-0064, Japan

Published online: 28 December 2025

certain characteristics, evaluations rely entirely on the available ground truth labels. This causes an overreliance on existing datasets which limits the model's adaptability and generalization power. For example, consider two of the most used datasets for general-purpose fraud detection work (McAuley and Leskovec 2013; Rayana and Akoglu 2015). Both are review networks extracted from online platforms whose labels are generated by an in-house filtering system. Exclusively relying on them for evaluation can inadvertently lead to simple imitation of the original filtering systems. Strong results on these datasets are valuable, but concerns of vulnerability to commonly observed long-term distributional shifts and adversarials have long been raised (Phua et al. 2010). Evaluation on more datasets helps, but sourcing a fraud dataset is tricky; fraud are inherently rare and labels are often costly. Furthermore, publicly available fraud datasets are often opaque (Tang et al. 2022).

To address the issue, a few directions have been proposed. One is extending the model design to capture temporal information and/or using dynamic graphs (Ren et al. 2023; Tian and Liu 2024; Yang et al. 2025b). This allows the model to learn from the distributional shifts present in dynamic graphs, resulting in better generalization. Another is using adversarials to train fraud detectors (Ren et al. 2020; Wu et al. 2023b, 2024), acknowledging that fraudsters are very quick to adapt and change strategies (Kurshan and Shen 2020). This approach reduces the reliance on existing datasets by using a strongly-defined fraud behavior to learn from. These two directions are respectively inspired by existing bodies of works on Dynamic Graph Learning (DGL) (Kazemi et al. 2020) and graph adversarial learning (Jin et al. 2021).

Table 1 Conceptual comparison between recent research/surveys and ours

	Past work/survey	Aspect			
		Fraud	Temporality	Adversarial	Multi-rounds
Dynamic graph learning	(Kazemi et al. 2020; Zhang et al. 2023)		✓		
Graph adversarial learning	(Fang et al. 2024; Tao et al. 2024)			✓	
Continual graph learning	(Yuan et al. 2023)				✓
Graph Fraud Detection	(Hyun et al. 2023; Wang et al. 2023; Wu et al. 2023a; Hyun et al. 2024; Roy et al. 2024; Zhuo et al. 2024)	✓			
	(Ren et al. 2023; Tian and Liu 2024; Yang et al. 2025b)	✓	✓		
	(Wu et al. 2023b, 2024)	✓		✓	
	(Li et al. 2024)	✓			✓
	Our work	✓	✓	✓	✓

The two directions correctly capture two different aspects of fraud detection: temporality and adversarial adaptation(Kurshan and Shen 2020). Detecting fraud in real life is a tug-of-war between fraudsters and detectors, each taking turns to adapt to the other's strategy. As such, fraud detection works ideally need to address both aspects. Existing works on the graph domain addressed each of the two aspects individually, but not together. Additionally, most of them still consider only a single iteration of model training and evaluation, failing to capture the back-and-forth nature of fraud detection. The study of iterative model training is mostly studied within the field of Continual Graph Learning (CGL)(Yuan et al. 2023), but typical CGL works do not concern themselves with adversarials. Our study excels through capturing these different aspects altogether, reframing the existing works in a unified direction that better reflects reality. Table 1 succinctly summarizes the difference between our work and existing adjacent ones.

In this work, we advance graph-based fraud detection by introducing a challenging scenario called **Multi-round Adversarial Fraud Detection** which simultaneously considers temporality and adversarial adaptation. This is done by conducting multiple model training/evaluation rounds and defining generator functions that govern the growth of the underlying graph between rounds. It offers freedom in defining the generator functions so that we are able to study the long-term dynamics of different kinds of fraud. Additionally, it reduces reliance on the labels in existing fraud datasets. Defining the generator function here can be equated to defining the fraud behavior itself, which provides clarity with respect to the objective and capabilities of the developed fraud detector model.

We here propose a simple greedy fraud generator function that specifically emulates promo fraud; it considers the fraud instances that have not been detected yet and duplicates them as new fraud instances. As a simple illustration, say a company schedules the training and inference of its transaction fraud detector weekly. At the beginning of the week, it correctly detects 100 fraud but unknowingly misses 10. In this situation, our generator function will take the 10 missed fraud and produces a large number of new fraud with identical information, knowing that the model this week will not classify them as fraud. This is a common behavior found in many real-world promo fraud. One example is coupon abuse, where a promotional coupon is used exceedingly beyond its intended amount, either by exploiting a hole in the provider's system or using bots. Another is

referral fraud, where a single actor creates a large number of new accounts to use a referral promotion usually provided to first-time users.

The term "fraud" specifically used within the context of this work does not refer to all forms of malicious deception. To be precise, we here define fraud as an act of crime of achieving direct financial gain through deceit. So this means it includes common financial fraud such as promo fraud, credit card fraud, and fake transactions, but excludes stuff like misinformation and fake news. This work recognizes two distinct fraud types: common fraud as implicitly defined by the ground truth of the dataset, and promo fraud as explicitly defined by our generator function. Our scenario poses several important challenges. Most important among them is that the fraud detector is required to adapt to harsh distributional shifts in subsequent rounds caused by the switch from naturally occurring general fraud to generated ones. For promo fraud, this adaptation requires the model to intentionally forget only mistaken knowledge that can contradict further learning. This kind of selective forgetting is beneficial in combating overfitting and improving the model's robustness (Wang et al. 2024b). In short, our proposed scenario improves on past works by adding layers of complexity that better mimic real-life fraud while reducing the reliance on dataset labels.

To complement the scenario, we also introduce **Temporally Pre-trained Node Embedder (TPNE)**. TPNE is a label and model-agnostic node embedder that improves adaptability to detect more fraud in the long run. To accommodate temporality, TPNE adopts a multi-step, multi-round information aggregation scheme. It first generates round-wise snapshots of the graph through masking and a sliding window. Then, it generates intermediary embeddings per neighborhood hop in the style of Jumping Knowledge Network (Xu et al. 2018b) before aggregating them into the final embedding with distinctly separate static and temporal parts. TPNE is trained in a self-supervised manner with an objective that promotes the independence and saliency of the encoded temporal information. Figure 1 shows an overview of our work.

The graph sequences show how the underlying graph evolves over multiple rounds. In each round, the fraud detector trains on existing nodes with known labels to predict the labels of newly emerging ones. The adversarial generator function then responds by looking at the current prediction status to generate new nodes and introduce them into the graph. As can be seen on the right side, in each round the proposed embedder takes a number of graph snapshots in a sliding window and uses them in a two-step learning procedure: a self-supervised training step for the embedder and a standard supervised step for the final classifier. The key technical contributions of our work can be summarized as follows:

- We designed a graph-based fraud detection scenario that can accommodate both the temporal and adversarial aspects of real-life fraud detection through multiple rounds of detection.
- Under the scenario, we additionally introduced a generator function inspired by simple real-life promo fraud behavior to study the efficiency of existing approaches in dealing with temporal and adversarial fraud.
- We proposed TPNE, a node embedder module that can maximally capture both static and temporal information, leveraging them to detect fraud behavior over multiple rounds.

We also support our technical contributions through extensive sets of experiments on datasets of varying sizes and domains. Results of our experiments show that past anomaly detection baselines and rudimentary strategies fail to accurately detect fraud produced by our generator. Additionally, we showed that the proposed embedder mitigates this failure by improving the adaptability of the state-of-the-art over multiple rounds.

The remainder of the paper is organized as follows: Sect. 2 summarizes existing works from relevant fields. Section 3, defines a general framework that we use to detail our scenario in Sect. 4. In Sect. 5, we describe the proposed TPNE. Section 6 contains the setup and results of our experiments. We provide in-depth discussions and implications of our work in Sect. 7 and conclude it in Sect. 8.

2 Related works

In this section, we introduce readers to several bodies of research relevant to our work. At the end of each subsection, we also provide a brief explanation on how our work contrasts or connects to the aforementioned works.

2.1 Graph-based fraud and anomaly detection works

Early works (Wang et al. 2019) were motivated by GNN's ability to consider local structure through neighborhood aggregation while succeeding ones focused on designing models based on certain characteristics associated with fraud. For example, to counter camouflaging fraudsters, CARE-GNN (Dou et al. 2020) utilizes reinforcement learning while ACD (Wang et al. 2024b) utilizes Graph Adversarial Networks (GAN). Other characteristics like heterophily and label imbalance have also been addressed.

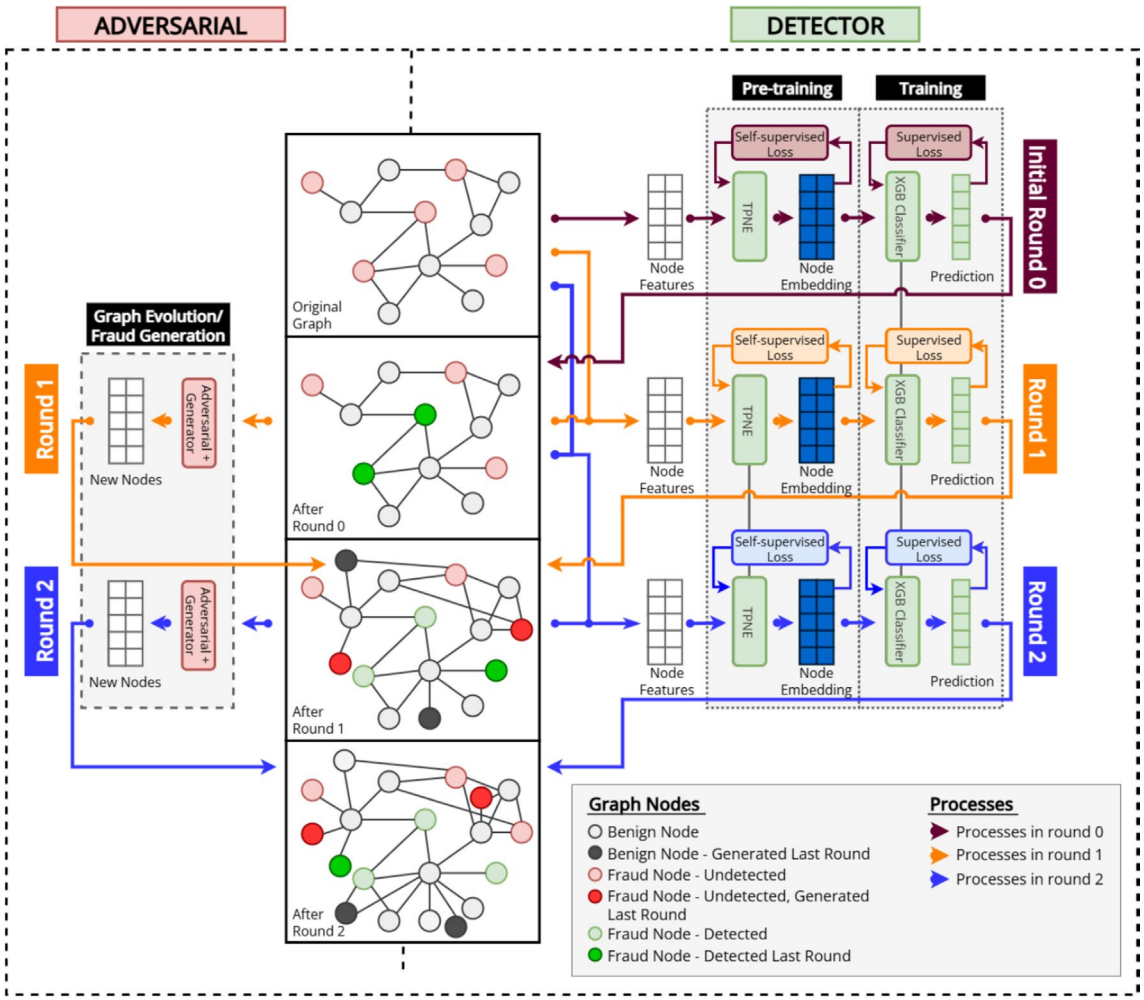

Fig. 1 High-level overview of the proposed Multi-round Adversarial Fraud Detection scenario and the accompanying TPNE. See how the underlying graph structure (middle) grows through the generator functions (left). Additionally, the predicted label of the nodes (middle) changes each round based on the model's prediction (right). Each round TPNE aggregates information from the latest W rounds and uses it for a two-step training process. Here, $W = 3$

GAGA (Wang et al. 2023) utilized a multi-relational group aggregation while SplitGNN (Wu et al. 2023a) opted for a spectral-based approach. PC-GNN (Liu et al. 2021b) adds a label-balancing sampler to deal with class imbalance and FRAUDRE (Zhang et al. 2021a) utilizes an unbalanced loss function. PMP (Zhuo et al. 2024) and LEX-GNN (Hyun et al. 2024) proposed a message-passing mechanism that discriminates fraud and benign neighbors.

Graph Anomaly Detection (GAD) is inseparable from graph-based fraud detection due to the similar data characteristics. GAD models are also often designed to deal with weak homophily (Gao et al. 2023; Qiao and Pang 2024) or class imbalance (Liu et al. 2022a; Zhang et al. 2022). The latest GAD benchmark (Tang et al. 2023) even includes several graph-based fraud detectors. In the benchmark, evaluation is done in a dataset-reliant single-round scenario and an XGBoost-based model with a parameterless feature aggregation was found to be most effective. We consider this finding as one of the limitations of existing fraud detection works and the importance of introducing temporality and adversarial strategies to the detection scenario.

We highlight several works with a degree of similarity with our own: CCL (Li et al. 2024) studied the case where fraud detection models need to be trained multiple times using data from different regional sources; GLSGNN (Wu et al. 2024) considered simulated adversarials when learning the graph structure; STA-GT (Tian and Liu 2024) leveraged temporal information in its encoding module; and both MetaGAD and GAD-NR (Xu et al. 2024b; Roy et al. 2024) utilize a self-supervised graph encoder backbone. Lastly, we note that most recent works share our concern on label reliance for graph anomaly/fraud detection and adopt the unsupervised setting. Examples include HUGE (Pan et al. 2025), which guides their unsupervised learning process

using label heterophily patterns, DiffGAD (Li et al. 2025), which utilizes diffusion sampling to inject the learned latent space with discriminative content, and SALAD (Ma et al. 2025), which is a structure-biased GAD framework with a custom node representation augmentation.

As previously stated, a number of research gaps remain underexplored by existing works on graph-based fraud detection. Among them is an overreliance on a small pool of labeled dataset for model evaluation. Additionally, there is a lack of consideration towards the adaptable and temporal nature of fraud. The main strength and contribution of our work lies in connecting these overlooked aspects of fraud/anomaly detection work (label scarcity, multi-round training, adversarials) and showing how they naturally fit modern fraud detection scenarios.

2.2 Multi-rounded learning scenario and forgetting

CGL involves the study of progressive knowledge accumulation (learning) over a continuous data stream for graph-structured data (Yuan et al. 2023). Specifically, it tries to avoid the occurrence of a negative backward transfer that deteriorates performance on past models, i.e., catastrophic forgetting (McCloskey and Cohen 1989). Currently, there are three main approaches to mitigate forgetting: using regularization to penalize the loss of past information such as TWP (Liu et al. 2021a); saving small slices of training data from past rounds such as ContinualGNN (Wang et al. 2020); and designing specific GNN modules to accommodate CGL such as TGN (Wang et al. 2022). Recent survey papers (Yuan et al. 2023; Zhang et al. 2024) offer comprehensive reviews on this topic.

Beyond being a focal point in CGL research, forgetting is a widely observed phenomenon in various tasks such as test-time adaptation, and meta-learning (Wang et al. 2024b). One important concept that is closely related to our work is beneficial forgetting: the notion that in some cases, we want to forget previously learned knowledge. The reason can be to mitigate overfitting (Shibata et al. 2021), to better learn new knowledge (Baik et al. 2020), or to respect data privacy (Cao and Yang 2015).

Multi-round Adversarial Fraud Detection scenario is similar to the common CGL framework in the way that they both comprise multiple training rounds. Within the context of fraud detection, our scenario is advantageous in two ways. First, round datasets in CGL are predetermined splits of the original dataset, whereas in our work, they are dynamically generated to accommodate the adaptive nature of frauds. Second, the primary focus of CGL research is mitigating catastrophic forgetting instead of promoting a beneficial one. Our work focuses on beneficial forgetting required to deal with promo fraud, which delineates it from the canon works of CGL.

2.3 Adversarial attacks on graph

Fraudsters are conceptually similar to Graph Adversarial Attacks (Chen et al. 2020a) due to the way they both consider the current capability of the detector model to generate future fraud patterns. Several criteria are commonly used to classify adversarial attacks. One criteria is the timing; the more popular poisoning attacks like Binarizedattack (Zhu et al. 2022) modify graph structure pre-learning in order to disrupt the training process, while evasion attacks like Projective Ranking (Zhang et al. 2021b) modify graph structure post-learning to mislead inference. Another criteria is the form of structure modification; an attack can either be edge modifications (Xu et al. 2019), or node injections (Fang et al. 2024).

Graph adversarial attack research can be grouped into those that developed novel attack algorithms and those who seek to improve model robustness under adversaries. A relevant recent trend from the latter group is the utilization of self-supervised methods to achieve better robustness like DefenseVGAE (Zhang and Ma 2024). Not only a reconstruction-based approach, a contrastive learning objective has also been shown to be useful to learn a refined graph structure representation with reduced adversarial elements (Li et al. 2022b; Tao et al. 2024). We refer readers to the excellent surveys (Jin et al. (2021); Sun et al. (2022), text) for a comprehensive review of this topic.

We note that the promo fraud studied in this work is a form of evasion node injection attack since we add new nodes after model training. Additionally, it is untargeted in the sense that it does not specifically aim to cause misclassification of certain nodes and is gray-box since we only provide the fraudster with the fraud nodes' labels, current prediction status, and local structure. Our work distinguishes itself from works on adversarial attack through its overall focus on fraud detection, where adversarials are positioned as a smaller part of a bigger whole.

2.4 Other related works and approaches

This work specifically address fraud detection within the domain of graph-structured data, which is a subset of a large body of work addressing the prevalence of malicious deception in our socioeconomic system. Outside of the graph domain, ongoing fraud detection research address similar challenges such as label imbalance and lack of data (Mienye and Jere 2024). Unsurprisingly, the usage of less label-intensive approaches like unsupervised learning (Hilal et al. 2022) is also a promising approach here.

Recent breakthroughs in Large Language Model (LLM) spawned works that try to utilize them in fraud detection (Chakraborty et al. 2024; Korkanti 2024). However, their number remained very few due to the non-textual nature of the dataset involved. FLAG (Yang et al. 2025a) tries to bridge this by using LLMs to enrich node features, but the inherently large computational power required for LLM makes comparison with non-LLM methods unfair.

Adjacent to fraud detection are works focusing on the detection of similar socioeconomic maladies such as fake news and misinformation. These works are reminiscent to fraud detection in the way that a subset of them operates within the graph domain (Wang et al. 2024a; Zhu et al. 2024). A number of surveys (Alnabhan and Branco 2024; Alghamdi et al. 2024) are available for readers interested in navigating these extensive body of works.

3 General framework

Here we introduce a general framework for a multi-round node classification task which then will be expanded to our specific scenario in Sect. 4. Let the standard notation $G = \{V, E, X, Y\}$ signify a labeled, undirected graph where V represents the set of n nodes in the graph and E the set of m edges. $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^n$ respectively represents the feature matrix and label vector.

To define a multi-round node classification task, let $\mathcal{R} = \{R^0, ..., R^T\}$ be a set of T rounds each consisting of three labeled graphs: $R^t = \{G^t, G^t_Q, G^t_S\}$. G^t represents the **test graph** used for the label prediction at test-time. $G^t_Q \subset G^t$ represents the **query graph** containing only nodes used in model evaluation. Finally, $G^t_S \subset G^t$ represents the **support graph** containing only the training nodes in R^t. Each of these comprises its own set of nodes, edges, features, and labels, i.e., $G^t_Q = \{V^t_Q, E^t_Q, X^t_Q, Y^t_Q\}$.

Let $F = \{f^0_\theta, ..., f^T_\theta\}$ be a set of predictors corresponding to each respective round, parametrized by θ. Each f_θ takes in a graph G, a set of nodes V and their feature matrix X to predict a set of labels for each node. That is, $f_\theta : G \times V \times X \to \hat{Y}$ where $\hat{Y}$ is the predicted node labels.

For each round R^t, f^t_θ is evaluated through its prediction result for the set of nodes in the query subgraph V^t_Q using G^t as the underlying graph structure. If we let $\mathbb{L}$ denote a function that measures the error of f^t_θ's prediction given G^t and G^t_Q and $\mathcal{L} : \hat{Y} \times Y \to \mathbb{R}$ denote a loss function that measures the correctness of $\hat{Y}$ w.r.t. to the true label Y, then the objective for each R^t is to find parameters θ that best minimize $\mathbb{L}$ through $\mathcal{L}$. Formally:

$$\min_\theta \mathbb{L}(f^t_\theta, G^t, G^t_Q) := \mathcal{L}(f^t_\theta(G^t, V^t_Q), Y^t_Q) = \mathcal{L}(\hat{Y}^t_Q, Y^t_Q) \quad (1)$$

On training time, the actual optimization is approximated using G^t_S as follows:

$$\min_\theta \mathbb{L}(f^t_\theta, G^t_S) := \mathcal{L}(f^t_\theta(G^t_S, V^t_S), Y^t_S) = \mathcal{L}(\hat{Y}^t_S, Y^t_S) \quad (2)$$

Table 2 summarizes the notations used in this paper. Our formalization differs from ones commonly used in CGL (Yuan et al. 2023) by distinctly separating model training and evaluation through the specification of G^t, G^t_Q, and G^t_S. This allows greater flexibility in describing the training and evaluation process each round.

4 Multi-round adversarial fraud detection

In this section, we exactly define our Multi-round Adversarial Fraud Detection scenario based on the previously established framework. Specifically, we describe how each of the three graphs is generated in a high-level overview before explaining the generator functions in detail.

4.1 Subgraph definitions

First, denote $G_{main} = \{V_{main}, E_{main}, X_{main}, Y_{main}\}$ as the main graph containing all original nodes and edges from the dataset with binary labels representing benign (0) and fraud (1) nodes; $\forall y \in Y : y \in \{0, 1\}$. For a scenario consisting of T rounds, we define the test graph, query graph, and support graph as follows:

- **In the initial round R^0** $G^0 = G_{main}$, i.e., the initial test graph is the original graph.
- $G^0_Q = \text{SAMPLE}(G_{main}, r)$, i.e., the initial query graph is a sampled subgraph of the original; SAMPLE extract a subgraph containing $r * |V_{main}|$ unique nodes from G_{main}.
- $G^0_S = G^0 - G^0_Q$, i.e., the initial support graph is a subgraph of the test graph consisting of nodes not present in the query graph. **In subsequent rounds, R^t, $(0 < t < T)$** [1]First, let $P^t = f^t_\theta(G^t, V^t) \odot Y^t = \hat{Y}^t \odot Y^t$, i.e., P^t is a vector that contains the correctness of f^t_θ's prediction each round.
- Then, let $P^t_{hist} = P^0 \vee ... \vee P^t$, i.e., P^t_{hist} is a vector that contains the historical correctness of f_θ's prediction up until R^t.

[1] $\cup$ denotes the graph union while $\vee$ denotes element-wise logical OR.

Table 2 A list of specific notations used throughout this paper and their descriptions

Notation	Description
Scenario-related	
T, t	The maximum round number and an variable for arbitrary round number
G_{main}	The base graph containing all original nodes and edges from the dataset
$\mathcal{R}$	The set of containing all T learning rounds
R^t	A single learning round; is a set comprised of G^t, G^t_Q. G^t_S, and f^t_θ
G^t	The **test graph** in R^t, used to evaluate the model at the end of the round
G^t_Q	The **query graph** in R^t, contains new nodes to be predicted on the round
G^t_S	The **support graph** in R^t, contains nodes available for training
f^t_θ	The fraud detection model/function in parametrized by θ in R^t
$\hat{Y}^t$	The vector of labels predicted by f^t_θ in R^t
P^t	Correctness of f^t_θ's prediction; $P^t \in \mathbb{R}^{\lvert V_t \rvert}$
P^t_{hist}	Cumulative correctness of f_θ's prediction up until R^t; $P^t_{hist} \in \mathbb{R}^{\lvert V_t \rvert}$
$g()$, $h()$	Generator function for the query/support graph G^t_Q/G^t_S every round
nPos, *nNeg*	The number of new positive/negative samples to be generated each round
TPNE-related	
L	The number of layers present in the embedder
$\mathcal{W}$, w	The size of the sliding window and a variable for arbitrary window position
W	Learnable weight in the initial feature transform and subsequent GNN layers
$H^{t.w}_l$, $\hat{H}^{t.w}_l$	Intermediate node embedding produced by an embedding layer in R^t for the graph snapshot of round $t-w$ after l hop(s) of aggregation
H^t_{static}, H^t_{temp}	Final static/temporal embedding in R^t
H^t_{final}	Final node embedding in R^t, a concatenation of H^t_{static} and H^t_{temp}
$\mathcal{L}_{recon}$	Reconstruction loss; diff. of mean neighbor distance between $H^{t.w}_l$ and $\hat{H}^{t.w}_l$
$\mathcal{L}_{disent}$	Temporal disentanglement loss; cosine distance between H_{static} and H_{temp}
$\mathcal{L}_{temp}$	Temporal maximization loss; inverse of average L2norm of H_{temp}
$\mathcal{L}_{final}$	The final loss
α, β	Hyperparameter controlling the weight of $\mathcal{L}_{disent}$ and $\mathcal{L}_{temp}$ in $\mathcal{L}_{final}$

- $G^t = G^{t-1} \cup g(G^{t-1}, P^{t-1}_{hist})$, i.e., the test graph in R^t is a union of the previous round's and the output of query generator function g.
- $G^t_Q = g(G^{t-1}, P^{t-1}_{hist}) \setminus G^{t-1}$, i.e., the query graph in R^t is a subgraph containing only new nodes generated by g.
- $G^t_S = G^{t-1}_S \cup h(G^{t-1}, \hat{Y}^{t-1}_Q)$, i.e., the support graph in R^t is a union of the previous support and a subgraph generated by the support generator function h.

Figure 2 depicts the three graphs. The test graph contains the entire graph structure used for evaluation, including the newly generated ones. The query graph is a small subgraph containing only the newly generated nodes that will be used for evaluation. Finally, the support graph is the subgraph used for model training. It is important to also note that the support graph is not fully labeled, the support generator function separately keeps track on what nodes were added with and without their labels.

Our scenario has two defining characteristics. First, the original dataset is used entirely for an initial inductive node classification round R^0 whereas common CGL setting splits it into smaller subgraphs for each round/task. Second, new nodes used for training/evaluation in subsequent rounds are generated by functions that takes past structures and predictions as input. This means learning in each round will depend on the result of previous ones, making it significantly harder to solve and analyze.

4.2 Generator functions

We introduce a simple g that mimics promo fraud and an h that mimics a typical fraud decisioning process in real life.

4.2.1 Query generator

We propose a simple procedure to generate new fraud each round. First, get all fraudulent nodes that are yet to be correctly detected. Then, randomly sample *nPos* nodes from the list. Finally, duplicate them (features and edges) and add the duplicates to the graph as new nodes. This emulates a simple spamming strategy very commonly found in promo fraud cases (Ekata 2023; Riskified 2023): when an existing fraud is not yet detected, fraudsters do it as much as possible.

As explained in Sect. 4.2, the information available for the query generator function is limited to the graph structure and past prediction results. The procedure above only utilizes the structural information from the 1-hop neighborhood and the prediction result of existing fraudulent nodes. We use the same procedure for generating benign nodes, but we sample from all existing ones to emulate just new nodes coming in. The pseudocode can be seen in Algorithm 1. The function ADDDUPLICATE(G, v, V, E, X, Y) adds a duplicate v to G and connects it to all of v's neighbors.

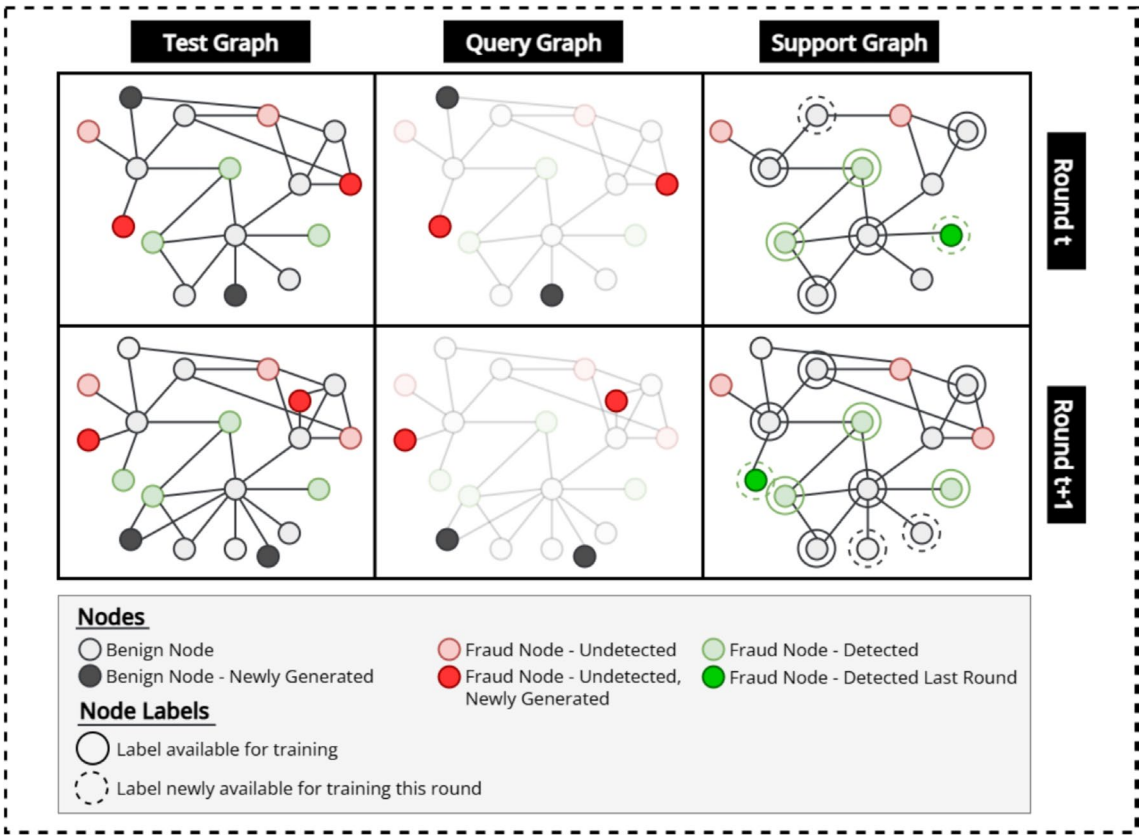

Fig. 2 An example of the state of test graph G^t, query graph G^t_Q, and support graph G^t_S for two consecutive rounds

Input: G^{t-1}, P^{t-1}_{hist}, $nPos$, $nNeg$
Output: G^t_Q

function GREEDYQUERY($G^{t-1}, P^{t-1}_{hist}, nPos, nNeg$)
  $\{V, E, X, Y\} \leftarrow G^{t-1}$
  $poolPos$, $poolNeg \leftarrow \emptyset$
  $G^t_Q \leftarrow \emptyset$

  ▷ Construct eligible node pools for fraud and benigns
  for $v \in V$ **do**
    if $(Y[v] == 1)$ and $(P^{t-1}_{hist}[v] == 0)$ **then** ▷ Misclassified fraud
      $poolPos \leftarrow poolPos \cup v$
    if $(Y[v] == 0)$ **then** ▷ All benigns
      $poolNeg \leftarrow poolNeg \cup v$

  ▷ Generate and add fraud
  for $i \leftarrow 0$ **to** $nPos$ **do**
    $v \leftarrow$ SAMPLE($poolPos$)
    $G^t_Q \leftarrow$ ADDDUPLICATE(G^t_Q, v, V, E, X, Y)

  ▷ Generate and add benigns
  for $i \leftarrow 0$ **to** $nNeg$ **do**
    $v \leftarrow$ SAMPLE($poolNeg$)
    $G^t_Q \leftarrow$ ADDDUPLICATE(G^t_Q, v, V, E, X, Y)
  return G^t_Q

Algorithm 1 Query Graph Generator g

4.2.2 Support generator

In formulating the support generator function, we consider how actors (e.g., companies, banks) typically validate the predictions of their fraud detection models for unseen data. In practice, simply enacting policies based on the prediction won't yield information about the true label. For example, if a company decides to preemptively block accounts they suspect then it becomes impossible to know if those accounts are actually fraudulent. This is because no new information about the account will arrive exactly due to that blocking. Similarly, in the case where the model predicts accounts as benign, the account may be just fraudulent in ways the model fails to detect.

Acquiring the true label usually involves manual qualitative checking which is costly when the data amount is huge. Therefore, having the true labels of all new nodes that appeared in previous rounds seems unrealistic. As a compromise, the support graph generator works as follows: it adds every node from the previous test graph to the support graph but provides the labels only when the model predicts them as fraud. This follows the assumption that on the simple policy of freezing or blocking the associated real-life entity predicted as fraud, mistakes will always lead to complaints that reveals the true label. On the other hand, there is no policy enacted for nodes predicted as benign. As such, there is no feedback from the real-life entity that proxies the true label. A real-life limitation to this assumption can come from inactive nodes (e.g., inactive user in an online marketplace) which will not produce complaints even though policies are enacted against them. Then again, depending on the business this limitation can be relatively harmless due to the low perceived values of inactive nodes.

The pseudocode for the support generator can be seen in Algorithm 2.

Input: G^{t-1}, $\hat{Y}^{t-1}$
Output: Support graph for next round G_S^t
  function CHECKPOSITIVES($G^{t-1}, \hat{Y}^{t-1}$)
    $\{V, E, X, Y\} \leftarrow G^{t-1}$
    $G_S^t \leftarrow \emptyset$
    $pool \leftarrow \emptyset$
    for $v \in V$ **do**
      if $(\hat{Y}^{t-1}[v] == 1)$ **then** ▷ Labeled if predicted as fraud last round
        $addNode(G_S^t, v, X[v], Y[v])$
      else ▷ Not labeled otherwise
        $addNode(G_S^t, v, X[v], \varnothing)$
    return G_S^t

Algorithm 2 Support Graph Generator h

5 Temporally pre-trained node embedder

In this section, we describe our proposed model. We see that our scenario and generator functions present two distinct challenges to tackle. First, the query generator g greedily chooses misclassified fraud nodes (type II errors) as the base for new ones each round. This means that to detect new fraud nodes, the fraud detection model needs to learn to correct the misclassifications it previously made. Second, no new information is given in the next training phase in direct relation to this correction. New labels are only provided for correctly classified fraud nodes and misclassified benign nodes (type I errors). Therefore, the model must be able to infer the type II errors indirectly from an alternative source of information.

To deal with these challenges, we introduce a label and model-agnostic self-supervised node embedder called **Temporally Pre-trained Node Embedder (TPNE)**. We pair TPNE with the state-of-the-art XGBoost-based approach based on the GAD benchmark; henceforth referred to as **TPNE-XGB**. TPNE consists of two main components: the Temporal Aggregator and Self-supervised Pre-training Objective.

5.1 Temporal aggregator

We incorporate the temporal aspect of fraud detection in our proposed scenario by switching from a static graph to a dynamic one. Therefore, in addition to static information from node features, we also need to extract temporal information from the evolving graph structure. This distinction between static and temporal information underpins the design of many DGL models as the two may have different uses (Kazemi et al. 2020). Note that the graph evolution in

our scenario is governed by generator functions and happens in large increments over a set amount of rounds. In other words, the granularity for the graph in our scenario is extremely coarse unlike common DGL works where most datasets are fine-grained (i.e. minutes/seconds intervals). With this in mind, TPNE employs a sliding window mechanism to accommodate temporality.

As seen in Fig. 4, given a window size of $\mathcal{W}$ in round t, we first generate $\mathcal{W}$ different temporary round-wise embeddings $\{H_0^{t.0}, ..., H_0^{t.\mathcal{W}}\}$ through masking and multiplication with an initial weight matrix W_0. Then, each of resulting matrices is passed through L GNN layers. The result of this process is $\mathcal{W} \times (L+1)$ different intermediate embeddings each representing the node features of the graph for the latest $\mathcal{W}$ rounds up until L hops of aggregations. We use the Graph Convolutional Layer (GCN) as the GNN layer of choice. For round t and temporary round embedding number w:

$$h_{v.0}^{t.w} = x_v^{t.w} W_0 \tag{3}$$

$$h_{v.(l)}^{t.w} = \sigma(b_l + \sum_{u \in \mathcal{N}(v)} \frac{1}{c_{uv}} h_{u.(l-1)}^{t.w} W_l) \tag{4}$$

$\mathcal{N}(u)$ is the neighbors of v, $c_{uv} = \sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(v)|}$, and σ is the activation function.

Next, two different embeddings are constructed from these feature matrices: a **static node embedding** H_{static}^t and a **temporal node embedding** H_{temp}^t. The former is produced by concatenating all intermediate feature matrices produced from the current unmasked node feature. This scheme is inspired by Jumping Knowledge Network (Xu et al. 2018b) and has been shown to improve performance through residual information from different hops. The latter is produced by doing the same concatenation for all w window-wise sets of matrices, concatenating the resulting matrices, and multiplying it with a weight matrix. The final embedding H_{final}^t is a concatenation between the two:

$$h_{v.static}^t = \Big\|_{i=0}^{L} h_{v.(i)}^{t.0} \tag{5}$$

$$h_{v.temp}^t = (\Big\|_{j=0}^{\mathcal{W}} \Big\|_{i=0}^{L} h_{v.(i)}^{t.j}) W_{temp} \tag{6}$$

$$h_{v.final}^t = h_{v.static}^t \,||\, h_{v.temp}^t \tag{7}$$

where $||$ denotes concatenation. Figure 4 illustrates the whole embedding process.

The two distinct embedding parts are designed to respectively capture static and temporal information. The static part is expected to be used to first try and classify fraud purely from the aggregated node features, most importantly in the initial round $t = 0$. The temporal part will then be utilized to correct predictions of misclassified fraud nodes in subsequent rounds $t > 0$ by capturing the graph evolution pattern.

Compared to past works on graph fraud/anomaly detection which consistently used transformer-based modules to capture temporality (Ren et al. 2023; Tian and Liu 2024; Yang et al. 2025b), the main strength of our proposed aggregation here is its simplicity, leading to less computational demand. Additionally, our aggregation schema is also agnostic to assumptions on the graph homophily, making it more general in theory. While we acknowledge the trade-off for this design in terms of expressiveness, we will see later that the decoupled nature of our node embedder compensates for this by utilizing a more powerful classifier head to achieve better performance.

5.2 Self-supervised pre-training

Capturing temporal information is necessary to detect misclassifications over multiple rounds, but discouraging the model from memorizing that misclassification is also important. It is possible that despite having access to temporal information, the model opts to fit to misleading static information. In essence, we want our model to selectively forget the learned pattern that leads to misclassifications. To achieve this, we introduce a self-supervised pre-training step for the embedder. Self-supervised learning (SSL) has been empirically shown to increase robustness against overfitting and various data noises for general models (Hendrycks et al. 2019) and specifically GNNs (Liu et al. 2022b). As such, it is very much in line with our goal of detecting fraud in a realistic setting.

We propose three key objectives to guide the self-supervised pre-training process: **Neighborhood Reconstruction**, **Temporal Disentanglement**, and **Temporal Maximization**. Each of these objectives is represented as a loss function and the final objective of the pre-training for every round t is to minimize the sum of these losses weighted by the hyperparameters α and β:

$$\mathcal{L}_{final}^t = \mathcal{L}_{recon}^t + \alpha \mathcal{L}_{disent}^t + \beta \mathcal{L}_{temp}^t \tag{8}$$

The key advantage offered by our pre-training objective here compared to past works such as GAD-NR (Roy et al. (2024), text) and MetaGAD (Xu et al. 2024b) is the capture of temporal information, which is a given considering that they were not designed for multi-round anomaly detection. Nonetheless, as detailed below our pre-training scheme and objective is more space and time-efficient by being relatively straightforward and devoid of computationally expensive processes such as sampling (GAD-NR) or Meta-learning (MetaGAD).

5.2.1 Neighborhood reconstruction

The most important objective for the self-supervised pre-training is to reconstruct the original graph. There are many ways to formulate graph reconstruction such as using perturbation, masking, etc. (Liu et al. 2022b), but here we focus on the local node neighborhood which is common for fraud detection works. We aim to reconstruct the average neighborhood distance for each nodes from a non-parametrized aggregation as follows:

$$\mathcal{L}^t_{recon} = \sum_{l=0}^{L} \sum_{v \in V} \frac{1}{|V|} MSLELoss \left(\sum_{u \in \mathcal{N}(v)} \frac{\mathcal{D}(\hat{h}^{t.0}_{l.v}, \hat{h}^{t.0}_{l.u})}{|\mathcal{N}|}, \sum_{u \in \mathcal{N}(v)} \frac{\mathcal{D}(h^{t.0}_{l.v}, h^{t.0}_{l.u})}{|\mathcal{N}|} \right) \tag{9}$$

As seen above, we do this for every node and every intermediate embedding. To get the non-parametrized aggregation from the original node feature, we replace the GCN layer on TPNE with a parameterless GIN as can be seen in the upper part of Fig. 4.

5.2.2 Temporal disentanglement

Considering that the fraud for round $t = 0$ originated from the dataset while fraud for subsequent rounds $t > 0$ are generated, we can say that they are products of different underlying distributions. As such, the required information to detect them should also be different. Equation (10) shows the term we introduce to promote differences between the resulting temporal part of the embedding and its static counterpart. The term maximizes the cosine distance between H_{static} and H_{temp}. By penalizing their correlation we expect the model to correctly utilize them for the two distinct fraud types in our scenario. We call this objective **Temporal Disentanglement**, taking inspiration from existing works that utilize disentanglement in DGL (Zhang et al. 2023) and Graph Out-of-distribution Learning (Li et al. 2022a).

$$\mathcal{L}^t_{disent} = 1 - \left(\frac{1}{|V|} \sum_{v \in V} \frac{\hat{h}^t_{temp.v} \cdot \hat{h}^t_{static.v}}{||\hat{h}^t_{temp.v}|| \; ||\hat{h}^t_{static.v}||} \right) \tag{10}$$

5.2.3 Temporal maximization

The final part of the pre-training objective is a term that penalizes weak signals for the temporal part of the embedding. This is done as follows:

$$\mathcal{L}^t_{static} = \frac{1}{log\left(\frac{\sum_{v \in V} ||h^t_{temp.v}||_2}{|V|} + \epsilon \right)} \tag{11}$$

where ϵ is set to $1e-10$ and is added to avoid log operation on 0. Along with the disentanglement term, this is expected to promote the learning of salient and meaningful temporal signals that can be useful in detecting misclassified fraud over multiple rounds.

5.3 Training and inference process

The learning process of TPNE does not require any labels. As such, it can be decoupled from the learning process of the actual classifier module. Figure 3 shows how TPNE is used in the overall learning process. Instead of jointly training TPNE and the classifier, TPNE is first trained in a self-supervised manner. Afterwards, a node embedding is produced from the original feature vector and graph structure through one forward pass on the trained TPNE. The classifier is then trained using that embedding, fully independent of TPNE.

A pseudocode detailing the training and inference process for a multi-round node classification task consisting of T rounds is provided in Algorithm 3.

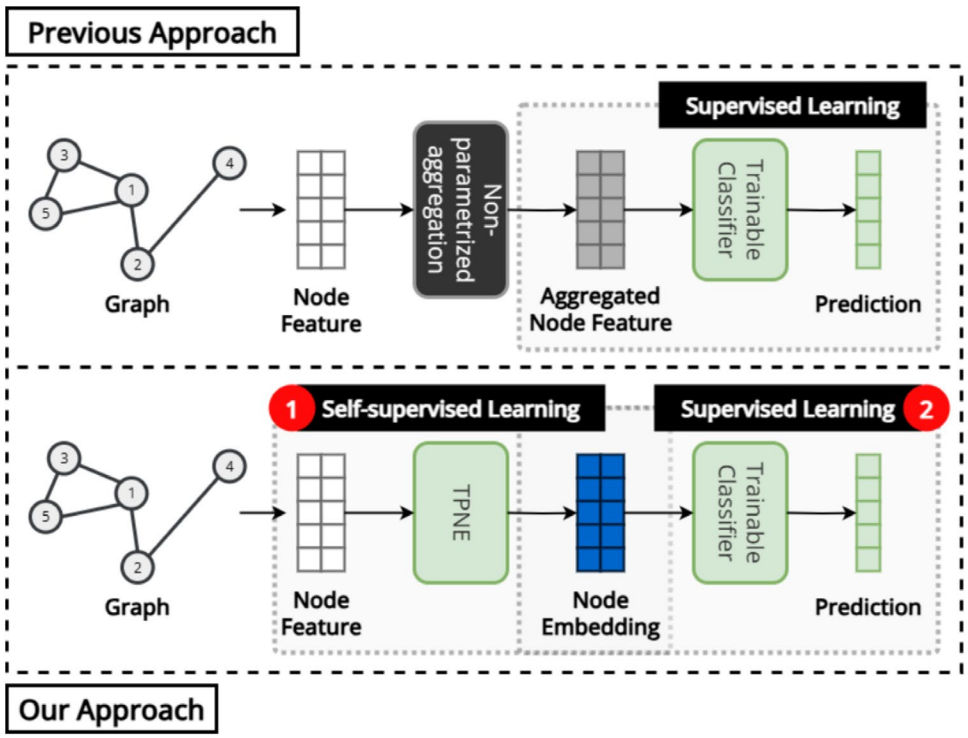

Fig. 3 Difference between the current state-of-the-art in Graph Anomaly Detection which uses a parameterless neighborhood aggregation (top) before training only the classifier in a supervised manner and our approach of doing a self-supervised learning step using TPNE before training the classifier (bottom)

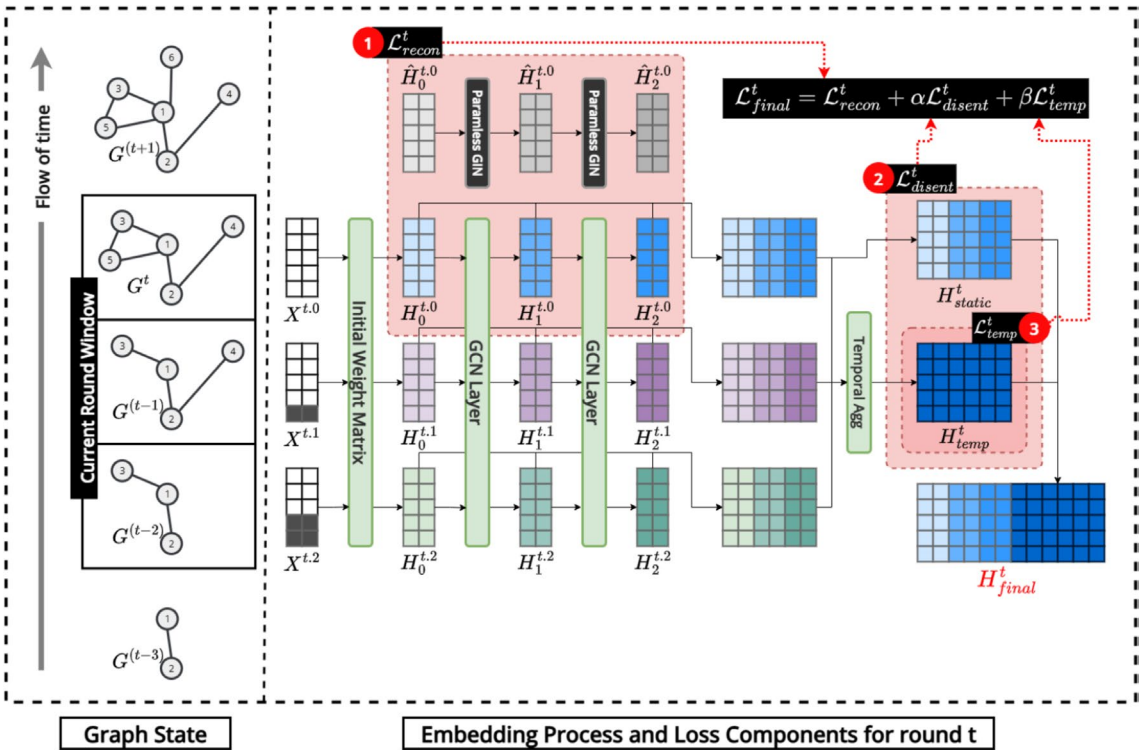

Fig. 4 Example of aggregation process using TPNE for window size $\mathcal{W} = 3$ and number of layer $L = 2$. Left side shows the state of the graph in the last 3 rounds and side shows the corresponding aggregation scheme. The three smaller red rectangles summarize the calculation $\mathcal{L}_{recon}$ (1), $\mathcal{L}_{disent}$ (2), and $\mathcal{L}_{temp}$ (3) which only happens on model training and not inference

Input: $G_{main}, T, L, W, maxEpoch$

1: **for** $t \leftarrow 0$ **to** T **do**
2: ▷ Prepare graphs for current round (Section 4.2)
3: **if** t == 0 **then** ▷ Initial round
4: $G^t \leftarrow G_{main}$
5: Randomize node age in V_t (to solve cold start problem)
6: $G^t_Q \leftarrow \text{SAMPLE}(G_{main}, r)$
7: $G^t_S \leftarrow G^t - G^t_Q$
8: **else** ▷ Subsequent round
9: $G^t \leftarrow G^{t-1} \cup g(G^{t-1}, P^{t-1}_{hist})$
10: $G^t_Q \leftarrow \text{GREEDYQUERY}(G^{t-1}, P^{t-1}_{hist}) \setminus G^{t-1}$
11: $G^t_S \leftarrow G^{t-1}_S \cup \text{CHECKPOSITIVES}(G^{t-1}, \hat{Y}^{t-1}_Q)$
12:
13: ▷ TPNE's self-supervised training
14: **for** $e \leftarrow 0$ **to** $maxEpoch$ **do**
15: ▷ Forward pass
16: **for** $w \leftarrow 0$ **to** W **do** ▷ Pass for each window
17: Generate $X^{t.w}$ from X^t_S using node age for masking
18: Calculate $H^{t.w}_0$ from $X^{t.w}$ (equation 3)
19: **for** $l \leftarrow 1$ **to** L **do** ▷ Pass for each layer in window
20: Calculate $H^{t.w}_l$ from $H^{t.w}_0$ on G^t_S (equation 4)
21: Calculate H^t_{static} (equation 5)
22: Calculate H^t_{temp} (equation 6)
23: Calculate H^t_{final} (equation 7)
24:
25: ▷ Backward pass
26: Calculate $\mathcal{L}^t_{recon}$ (equation 9)
27: Calculate $\mathcal{L}^t_{disent}$ (equation 10)
28: Calculate $\mathcal{L}^t_{temp}$ (equation 11)
29: $\mathcal{L}^t_{final} \leftarrow \mathcal{L}^t_{recon} + \alpha\mathcal{L}^t_{disent} + \beta\mathcal{L}^t_{temp}$
30: Calculate gradient from $\mathcal{L}^t_{final}$ and update weights
31: ▷ Classifier's supervised training
32: Get $H^t_{S.final}$ from forward pass on TPNE on G^t_S (line 15 to 23)
33: Train classifier on $G^t_S, H^t_{S.final}$, and Y^t_S
34:
35: ▷ Final inference and prediction
36: Get $H^t final$ from forward pass on TPNE on G^t (line 15 to 23)
37: Get final prediction $\hat{Y}^t$ from classifier using $H^t final$ on G^t
38: Update P^t and P^t_{hist} (Section 4.2)

Algorithm 3 TPNE Training/Inference in the Multi-round Adversarial Fraud Detection scenario

6 Experiments

In this section, we present the setup and the results of our experiments for evaluating the effectiveness of existing models and TPNE under our scenario. Unfortunately, to the best of our knowledge no existing work can serve as a true baseline due to the newness of the learning setting. Here, we opt to compare our method to existing state-of-the-arts for single-round supervised GAD with a simple continual training scheme. We also compare recent work from unsupervised and semi-supervised GAD to provide additional insights. In addition, while the availability of the entire graph in our scenario makes it fundamentally different from CGL, we also conduct a smaller set of comparisons to well-known CGL methods reported outside of this section in Appendix TODO.

First, we evaluate under the standard scheme of continuing training each round to see how existing modes perform under the proposed scenario. Later on, we see how that standard scheme compares to simple threshold-based spam-detecting strategies. Finally we also conduct ablation and sensitivity studies on TPNE to validate its design.

Table 3 Detailed statistics and description of the fraud datasets used in this work

Dataset	Statistics					
	$\|V\|$	$\|E\|$	Fraud%	#Feat	Node	Edge
Tolokers	11,758	530,758	21.8%	10	User	Work collaboration
Amazon	11,944	8,847,096	6.8%	25	User	Review correlation
YelpChi	45,954	7,739,912	14.5%	32	Review	Reviewer interaction
Elliptic++	203,769	234,355	2.2%	183	Transaction	Blockchain transaction record
T-Finance	39,357	42,484,443	4.5%	10	User	Transaction record
TC	200,236	23,690,536	13.5%	96	Transaction	IP/MAC matches

6.1 Experimental setup

6.1.1 Datasets

We utilize multiple popular fraud-related datasets with varying domains and sizes. Smaller datasets include the popular Amazon (McAuley and Leskovec 2013) and YelpChi (Rayana and Akoglu 2015) review datasets. Additionally we also use Tolokers (Platonov et al. 2023), a user network mined from an online crowdsourcing platform. Larger datasets include Elliptic++ (Elmougy and Liu 2023) and T-Finance (Tang et al. 2022), two common datasets for transaction fraud, and the TC dataset (Tian and Liu 2024), a graph constructed from an online transaction history dataset. Both are at least one order magnitude larger than Amazon/YelpChi.

We use dataset files sourced from GADBench (Tang et al. 2023) except for the TC dataset, which we independently acquire from the primary source and then preprocess. We flatten all graphs to be undirected and single-relational. The detailed statistics can be seen in Table 3. Full dataset and source code is available on inquiry.

6.1.2 Model baselines

Standard GNN Models. These are various popular GNNs with simple architectures: GCN (Kipf and Welling 2016), GraphSAGE (Hamilton et al. 2017), GIN (Xu et al. 2018a), GAT (Veli?kovi? et al. 2018), and GCNII (Chen et al. 2020b). We include these models due to the newness of our scenario.

GNN-based Supervised GAD Models. These are designed to detect single-round fraud/anomalies: BWGNN (Tang et al. 2022) and GHRN (Gao et al. 2023) are spectral-based, while PMP (Zhuo et al. 2024) uses a custom message passing scheme to achieve state-of-the-art on supervised GAD.

Other GNN-based GAD Models. GAD-NR (Roy et al. (2024), text) and MetaGAD (Xu et al. 2024b) utilize an unsupervised embedder similar to TPNE to outperform state-of-the-art in unsupervised and weakly-supervised GAD, respectively. We also attach an XGB Classifier to GAD-NR since the original paper focused on a fully unsupervised scenario.

XGBoost-based Models. These use an XGBoost classifier which have been shown to perform extremely well at detecting anomalies (Tang et al. 2023) despite not being specifically designed to handle fraud. Here we choose a variant with the raw node feature (XGB) and neighborhood-aggregated feature (XGB-GIN) as baselines.

TPNE Variants. For the ablation study, we introduce 2 extra variants of TPNE-XGB: one that is pre-trained without the temporal aggregation (PNE-XGB) and one that only utilizes the temporal aggregation scheme without pre-training (TNE-XGB). PNE-XGB only uses $\mathcal{W} = 1$ and only $\mathcal{L}_{recon}$ as a pre-training objective. TNE-XGB replaces the GCN layers with parameterless GIN layers and swaps the weight matrix used for temporal aggregation in Equation (6) with one of the following:

$$\hat{h}^t_{v.temp} = AGG(\forall j \in 1...\mathcal{W} : \mathop{||}_{i=0}^{L} \hat{h}^{t.0}_{v.(i)} - \mathop{||}_{i=0}^{L} \hat{h}^{t.j}_{v.(i)}) \tag{12}$$

$$\hat{h}^t_{v.temp} = AGG(\forall j \in 1...\mathcal{W} : \mathop{||}_{i=0}^{L} \hat{h}^{t.j-1}_{v.(i)} - \mathop{||}_{i=0}^{L} \hat{h}^{t.j}_{v.(i)}) \tag{13}$$

Where AGG is a choice between an average or sum aggregation. The choice between Eqs. (12) and (13) and the choice of AGG is treated as hyperparameters.

Precluded Baselines. This time around, we preclude baseline models from the field of CGL and DGL from this section. Among other reasons, this is primarily meant to focus discussions on baselines that are contextually designed for fraud and anomaly detection. For DGL models, most of them rely on either a self-attention mechanism or recurrent neural networks, which incurs significant computational load. Considering also the non-triviality in adopting some of them due to them either primarily being designed for temporal link prediction or operating in more granular snapshots, we leave unified comparisons across these different fields for future work. In the case of CGL, existing models focus on minimizing forgetting, which is actually counter-relevant in our promo fraud setting. However, considering how CGL's multi-round nature is identical to ours, we still provide results from a set of experiments performed

in small graphs using common CGL baselines in Appendix A.

6.1.3 High-level baselines

Alternative Retraining Scheme. Along with the continual retraining chosen for the main experiment set, we compare two additional naive retraining schemes to observe how they affect fraud detection performance in the multi-round scenario.

- **FIRST:** Model is trained only in the initial round R^0. As such, it only learns from nodes in G_{main}, and not newly generated ones.
- **CONTINUE:** Model is trained on every round in a continual manner, i.e., the training in R^t is done on the model from R^{t-1}. This is the default retraining scheme used in the baseline comparison.
- **RESET:** A new model is trained from scratch every round. In CGL, this often represents the performance ceiling w.r.t. forgetting.**Threshold-based Strategy.** We also propose three simple model-agnostic strategies to serve as a reference for the effectiveness of GNN-based learning approaches and our proposed model. Considering that the query node generator we chose for this work produces promo fraud, the strategies below are based on thresholding which is common in combatting them. Pseudocodes for these can be found in Appendix C.

- **THRES-DEG:** Utilizes structural information. This strategy considers new neighbors of unusually high-degree nodes as spams.
- **THRES-FEAT:** Utilizes feature information. This strategy considers new nodes that lie in a densely-populated feature space as spams.
- **THRES-AGGFEAT:** Similar to THRES-FEAT, but uses the final embedding produced by the GNN backbone instead of the original feature.**Relaxations.** Finally, we also provide results under relaxed constraints on the training data to serve as an approximation of the performance ceiling.

- **TRANSDUCTIVE:** Operates under a transductive setting where the model is trained in a semi-supervised manner on a graph that includes the new nodes generated each round.
- **ORACLE:** Uses the entire graph along with the true label as the training set, including the new nodes generated each round. Provided to check if a model can fit to the generated fraud's pattern.

6.1.4 Evaluation metrics

To account for the multi-round nature of our scenario, we provide three metrics that measure different aspects of the model performance:

- **Initial F1:** how much the model learns and retains from the initial round without forgetting:

$$R^T\text{-}\ f1^0 = f1(f_\theta^T(G^T, V_Q^0), Y_Q^0) \tag{14}$$

- **On-round F1:** how good the model is in detecting newly seen generated fraud:

$$R^T\text{-}\ avgf1^{t>0} = \frac{1}{T-1}\sum_{t=1}^{T} f1(f_\theta^T(G^t, V_Q^t), Y_Q^t) \tag{15}$$

- **Final Recall:** the proportion of fraud that have been correctly detected by the model after the final round:

$$R^T\text{-}\ recall^0 = recall(P_{hist}^T, Y^0) \tag{16}$$

Note that improving one of these metrics may degrade the others. As such, model evaluation involves considering which metric(s) may be more important and how much the other can be compromised. All results are displayed as percentages.

6.1.5 Additional details

Data split and round increment. All models are trained and evaluated inductively; nodes and edges used for evaluations are not available in the training graph. Note that in R^0 the training is fully supervised, as the training graph is fully labeled. In subsequent rounds, training is semi-supervised since labels for the generated nodes are provided conditionally. For R^0, the train:validation:test ratio is 3:2:5. The training and validation splits comprise the initial support graph G_S^0 and the testing split comprises the initial query graph G_Q^0. Splits are done randomly for all datasets except the TC dataset, where we use existing timestamps for node creation.

On each subsequent round $R^{t>0}$, the generator functions produce new nodes amounting to 5% of the original graph G_{main} with identical fraud ratio. Unless specifically mentioned, every experiment consists of ten rounds. This means at the end of the experiment, the node count of the graph grows to 145% the original. All experiments are repeated ten times and uses randomized data splits except for the TC dataset.

Table 4 Results for standard training scheme of continuing training on the same model each round on the smaller datasets: Tolokers, Amazon, and YelpChi

	Tolokers				Amazon			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank
GCN	75.4 ± 7.77	65.6 ± 5.02	49.8 ± 1.31	6.00	69.2 ± 25.9	57.9 ± 19.4	49.2 ± 3.05	9.00
GraphSAGE	69.4 ± 6.94	75.3 ± 6.82	49.4 ± 0.98	4.67	$\mathbf{93.8 \pm 6.01}$	88.2 ± 11.0	55.4 ± 6.58	3.67
GIN	53.7 ± 3.27	65.6 ± 0.68	42.4 ± 0.84	11.67	79.7 ± 19.2	83.6 ± 24.6	54.4 ± 5.32	5.33
GAT	67.7 ± 3.52	68.0 ± 3.57	46.8 ± 0.83	8.67	56.5 ± 11.5	90.7 ± 1.10	53.4 ± 2.75	6.00
GCNII	59.8 ± 5.42	71.5 ± 2.01	46.9 ± 0.60	8.67	49.8 ± 7.54	91.5 ± 1.16	50.1 ± 1.71	8.00
BWGNN	61.6 ± 2.81	77.2 ± 2.29	47.9 ± 0.81	6.00	63.1 ± 8.41	88.8 ± 7.72	53.6 ± 2.02	6.33
GHRN	62.2 ± 4.86	72.4 ± 3.20	47.8 ± 0.98	7.00	72.5 ± 8.40	88.6 ± 8.32	54.2 ± 1.23	5.33
PMP	73.1 ± 9.26	72.4 ± 4.89	50.5 ± 2.17	4.33	49.2 ± 9.64	88.9 ± 7.51	49.1 ± 1.79	10.00
MetaGAD	62.5 ± 8.82	62.6 ± 1.67	42.2 ± 2.75	11.00	31.2 ± 5.36	63.7 ± 5.63	48.8 ± 0.79	12.33
GAD-NR	74.5 ± 4.78	71.2 ± 3.16	51.8 ± 1.39	4.33	52.5 ± 11.3	55.6 ± 8.13	51.6 ± 1.12	10.00
XGB	36.4 ± 1.40	73.1 ± 0.76	44.1 ± 0.56	9.33	50.2 ± 2.25	95.3 ± 0.51	51.8 ± 1.20	6.67
XGB-GIN	58.3 ± 1.65	$\mathbf{86.9 \pm 0.77}$	50.7 ± 0.62	5.00	55.5 ± 3.97	$\mathbf{95.4 \pm 0.58}$	$\mathbf{55.4 \pm 1.19}$	**3.33**
TPNE-XGB	$\mathbf{85.9 \pm 3.36}$	69.1 ± 1.23	$\mathbf{55.2 \pm 1.67}$	**3.67**	65.3 ± 7.76	89.3 ± 1.95	54.7 ± 2.82	4.33
	YelpChi				Overall Rank			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank				
GCN	67.9 ± 21.8	50.3 ± 7.40	45.6 ± 1.01	10.33	8.44			
GraphSAGE	83.1 ± 10.8	70.4 ± 15.5	51.6 ± 1.71	5.00	4.44			
GIN	78.5 ± 16.9	56.5 ± 17.0	45.4 ± 1.93	9.33	8.78			
GAT	80.4 ± 12.0	62.9 ± 8.18	47.2 ± 1.19	7.33	7.33			
GCNII	58.9 ± 9.24	75.2 ± 4.66	49.1 ± 1.33	7.67	8.11			
BWGNN	$\mathbf{88.8 \pm 3.86}$	61.6 ± 18.9	50.6 ± 2.00	5.67	6.00			
GHRN	86.2 ± 7.28	66.8 ± 12.4	52.8 ± 0.94	4.33	5.56			
PMP	71.4 ± 3.98	79.7 ± 9.59	54.8 ± 0.75	4.33	6.22			
MetaGAD	7.27 ± 4.45	48.7 ± 2.08	46.1 ± 0.31	12.33	11.89			
GAD-NR	65.7 ± 30.6	49.4 ± 3.35	48.2 ± 1.62	10.00	8.22			
XGB	55.7 ± 2.96	93.2 ± 1.43	52.5 ± 0.82	6.33	7.44			
XGB-GIN	62.2 ± 1.52	$\mathbf{95.6 \pm 0.31}$	55.1 ± 0.60	4.33	4.22			
TPNE-XGB	77.2 ± 2.07	70.6 ± 1.15	$\mathbf{56.1 \pm 0.58}$	**4.00**	**4.00**			

*R^T - recall^0 (Final Recall): Cumulative recall of all fraud after the final round

*R^T - fl^0 (Initial F1): F1 Score of test set from round 0 after the final round

* R^T -$avgfl^{t>0}$ (On-round F1): F1 Score for test set of round t > 0 immediately after that round ends

Parameter setting and optimization. The following parameter settings are used unless mentioned otherwise in Subsection 6.2: we use the Adam optimizer with a learning rate of 0.01. Where applicable, we use LeakyReLU as the activation function, layer normalization, and a dropout rate of 0.1. The embedding dimension is set to 64 and the number of layers is set to 2. We set the maximum number of training epochs to be 500 for R^0 and 300 for $R^{t>0}$. Early stopping is performed when the validation score stagnated for 75 epochs. To deal with the cold start issue for the proposed model which requires temporal information of nodes, every round before training we randomize the age of nodes from R^0 to a value between 1 and $\mathcal{W}$.

To ensure each model has a chance to achieve optimum performance, we perform hyperparameter optimization using grid search. Details can be found in Appendix B.

6.2 Main experiment results

6.2.1 Multi-round detection with continual training

Tables 4 and 5 contain results of the first set of experiments where the models are trained continually each round. The Rank column contains the average rank for each of the three metrics in each dataset, assuming equal importance among them. Bold indicates the best result in the column.

Our result aligns with the latest benchmark (Tang et al. 2023). XGB-GIN wins in terms of initial F1 (R^T-$f1^0$), followed by XGB. However, the proposed TPNE-XGB is better when we consider all metrics with an overall rank of 4.00 on the small datasets and 3.67 on the larger ones. More precisely, it provides high final recall and on-round F1 with the highest R^T-$avgf1^{t>0}$ seen in Tolokers and YelpChi

Table 5 Experiment results for standard training scheme of continuing training on the same model each round for the larger datasets: TC, T-Finance, and Elliptic

	TC				T-Finance			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank
GCN	47.7 ± 5.05	87.2 ± 0.46	50.5 ± 0.38	6.33	72.2 ± 15.7	64.2 ± 24.1	46.8 ± 3.16	7.00
GraphSAGE	61.4 ± 7.05	86.6 ± 0.82	52.3 ± 1.05	4.33	19.6 ± 20.3	49.0 ± 0.56	47.5 ± 4.67	10.00
GIN	41.3 ± 0.52	87.9 ± 0.62	50.1 ± 0.31	7.00	$\mathbf{90.2 \pm 9.40}$	68.1 ± 33.5	50.8 ± 3.47	5.00
GAT	57.0 ± 6.53	80.0 ± 11.7	49.9 ± 0.96	7.33	57.6 ± 9.49	64.2 ± 10.0	50.0 ± 2.02	7.67
GCNII	46.4 ± 5.31	88.3 ± 0.74	49.0 ± 0.78	7.33	64.9 ± 7.83	84.2 ± 11.7	54.8 ± 1.30	3.33
BWGNN	47.7 ± 3.10	89.4 ± 0.42	51.2 ± 0.25	4.33	67.1 ± 6.96	77.1 ± 20.7	51.5 ± 2.08	4.33
GHRN	48.3 ± 4.28	87.3 ± 1.71	49.8 ± 0.98	6.67	62.6 ± 6.46	87.5 ± 11.9	52.9 ± 1.76	4.00
PMP	71.0 ± 9.26	85.9 ± 0.05	54.4 ± 1.52	4.00	19.5 ± 25.3	48.8 ± 0.01	48.3 ± 1.08	10.33
XGB	38.4 ± 0.16	85.9 ± 16.1	46.3 ± 0.09	10.00	42.6 ± 1.70	91.7 ± 0.95	51.2 ± 0.58	5.67
XGB-GIN	50.9 ± 2.08	$\mathbf{91.2 \pm 0.49}$	50.9 ± 0.66	**3.67**	49.9 ± 1.25	$\mathbf{95.0 \pm 0.41}$	51.3 ± 0.52	4.67
TPNE-XGB	$\mathbf{83.8 \pm 1.54}$	78.8 ± 0.74	$\mathbf{55.5 \pm 0.98}$	4.33	62.4 ± 2.61	88.8 ± 0.83	$\mathbf{54.8 \pm 1.19}$	**3.33**
	Elliptic++				Overall Rank			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank				
GCN	75.1 ± 10.6	85.3 ± 9.06	57.8 ± 1.50	5.33	6.22			
GraphSAGE	79.1 ± 13.1	82.6 ± 14.2	57.9 ± 4.22	5.00	6.44			
GIN	85.5 ± 12.6	61.6 ± 27.0	53.5 ± 2.99	6.00	6.00			
GAT	69.1 ± 10.5	70.1 ± 11.2	52.2 ± 1.84	7.67	7.56			
GCNII	28.1 ± 25.2	52.0 ± 8.91	42.0 ± 7.39	10.00	6.89			
BWGNN	$\mathbf{86.5} \pm 6.17$	89.5 ± 13.2	$\mathbf{62.0} \pm 3.27$	**2.00**	**3.56**			
GHRN	85.2 ± 4.97	88.2 ± 20.3	59.4 ± 3.01	3.33	4.67			
PMP	OOM	OOM	OOM	OOM	7.17			
XGB	50.0 ± 0.65	$\mathbf{95.5} \pm 0.06$	53.1 ± 0.45	6.00	7.22			
XGB-GIN	51.5 ± 0.91	95.4 ± 0.04	54.6 ± 0.47	5.33	4.56			
TPNE-XGB	67.8 ± 5.29	89.7 ± 1.30	58.9 ± 1.86	4.33	4.00			

*R^T - recall^0 (Final Recall): Cumulative recall of all fraud after the final round

*R^T - $f1^0$ (Initial F1): F1 Score of test set from round 0 after the final round

* R^T -$avgf1^{t>0}$ (On-round F1): F1 Score for test set of round t > 0 immediately after that round ends

along with above average R^T-$recall^0$ and R^T-$f1^0$. This is a favored behavior for long-term fraud detection.

On the other hand, simpler GNNs (GCN, GraphSAGE, GIN) tend to behave erratically; they excel in final recall at times (for example, GraphSAGE on Amazon/YelpChi), but are also inconsistent as indicated by the high deviation. GAT and GCNII are more consistent but showed lower peak performance. The other 5 GAD models show middling performance, lagging behind XGB-based models when it comes to initial F1 and lagging behind TPNE-XGB when it comes to on-round F1 and final recall.

Table 5 shows that similar results can be observed in large datasets. In TC and T-Finance, XGB-GIN excels on initial F1 but lacks the high on-round F1 and final recall of TPNE-XGB. The erratic behavior for simple GNN models persists as indicated by the standard deviation, and lighter GAD baselines (BWGNN, GHRN, PMP) sit in the middle. The Elliptic++ dataset produced a slightly different result, with spectral-based models (BWGNN, GHRN) leading the pack. Here, XGB-based models seem to overfit to the initial training set and failed to adapt to newer generated fraud, as indicated by the high initial F1 but low final recall. This overfitting is most likely due to the dataset's extreme sparsity. The average node degree in Elliptic++ is 1.15, so the initial learning process is strongly dominated not by structural information but by the raw features of a node and its singular neighbor. That being said, our model managed to avoid this pitfall and still exhibited a high on-round F1 just behind BWGNN and GHRN. When we combine the ranks for small and large datasets, TPNE-XGB achieves the highest overall rank at 4.00, with XGB-GIN just behind at 4.39.

Results for MetaGAD and GAD-NR are missing due to execution time exceeding the maximal 24-hour/run.

We plot the roundwise F1 score for the Tolokers and Amazon experiments in Fig. 5. See that in the initial round, XGB-GIN (orange line in c) outperforms everything else with BWGNN, GHRN, and PMP (blue, orange, green line in b) trailing behind it. However, that performance declines when we start to consider subsequent rounds; on-round F1 hovers around 0.5 for all baselines for all datasets except for

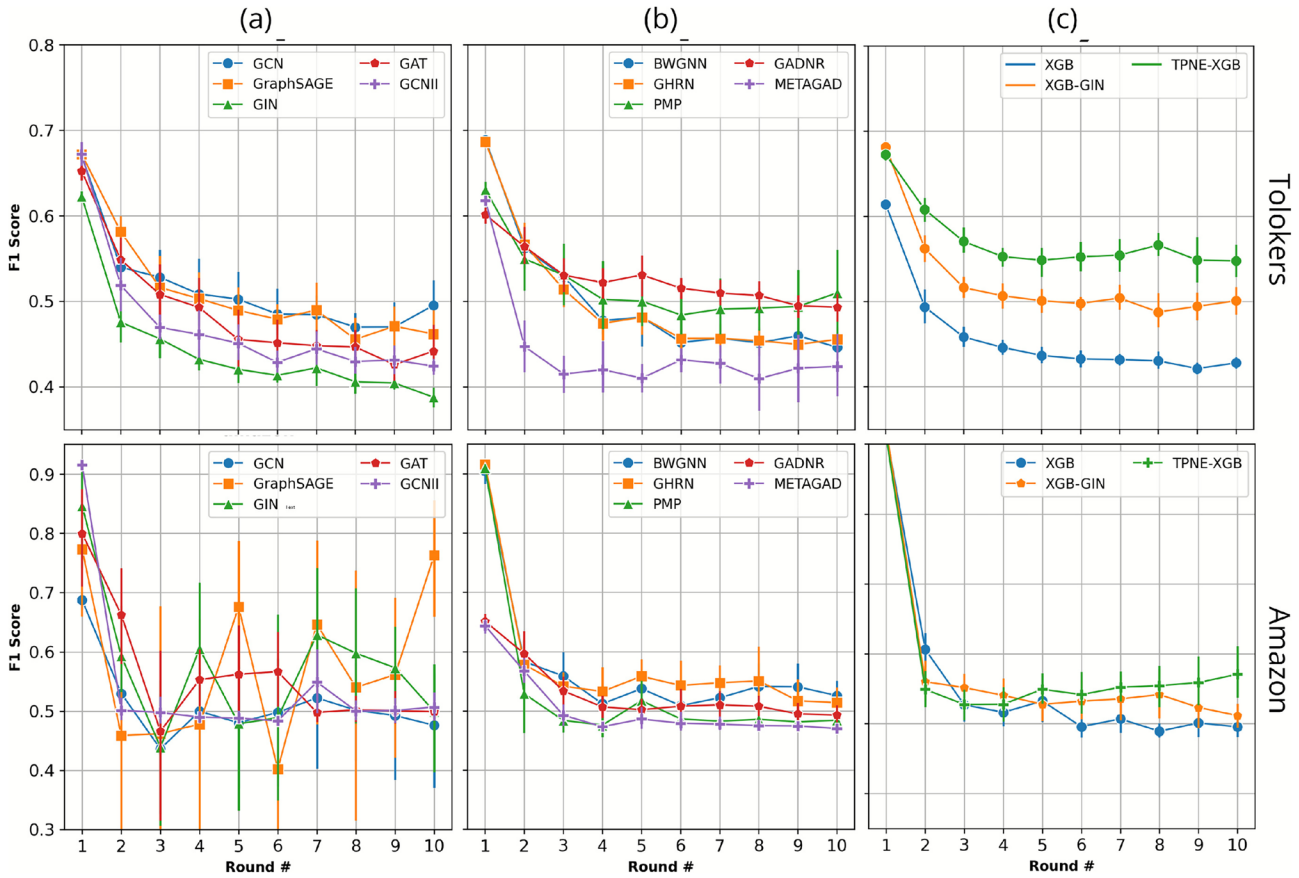

Fig. 5 Evolution of F1 score during the 10 detection rounds of simple Graph Neural Networks (**a**), Graph Anomaly Detection baselines (**b**), and XGBoost-based models (**c**) on the Tolokers (top) and Amazon (bottom) dataset. Marker values are average of 10 repetitions and bands denote standard deviation

TPNE-XGB (green line in c) which maintains the highest F1 score on subsequent rounds.

6.2.2 Training alternatives and threshold-based strategies

Results for the second experiment set which compares different training schemes, high-level strategies, and training data constraints for XGB-GIN and TPNE-XGB is shown in Table 6. Here, the CONTINUE row in bold text represents the performance of the standard retraining scheme and can be considered as the primary point of comparison.

First, we examine the effect of different learning schemes. As expected, not retraining the model on subsequent rounds (FIRST) degrades all metrics for both models across all datasets, except for initial F1 of TPNE-XGB on Amazon. Training from scratch every round (RESET) produces a more interesting result; it improves the final recall and on-round F1 of XGB-GIN on Tolokers and YelpChi by 2-5 percent at the cost of a marginal decline in initial F1. However, RESET degrades the final recall and on-round F1 of TPNE-XGB across all datasets. This relates to the initial performance for the standard scheme (CONTINUE) and is discussed in Sect. 7.2.

Next, we examine the effect of augmenting the model prediction with a threshold-based fraud detection strategy. We see that THRES-DEGREE is a terrible strategy as it almost always causes significant degradation of on-round F1. On the other hand, both feature-based strategies achieved mixed results: THRES-FEAT improves performance on Amazon and Yelpchi for both XGB-GIN and TPNE-XGB, although with a marginal decline in final recall for the latter. THRES-AGGFEAT causes a marginal decline on Tolokers and only slight changes on the other two datasets. We can conclude that employing threshold-based strategies does almost nothing for TPNE-XGB and is only marginally useful for XGB-GIN. It is important to note that these threshold-based strategies can not close the final recall gap between XGB-GIN and TPNE-XGB.

Table 6 Experiment results for different training schemes, threshold-based strategies, and training data constraints on the smaller datasets: Tolokers, Amazon, and YelpChi

Strategy/Scheme	XGB-GIN			TPNE-XGB		
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$
Tolokers						
FIRST	53.2 ± 3.06	68.2 ± 0.92	43.9 ± 0.63	46.4 ± 3.23	90.9 ± 0.92	49.1 ± 1.33
CONTINUE	$\mathbf{58.3 \pm 1.65}$	$\mathbf{86.9 \pm 0.77}$	$\mathbf{50.7 \pm 0.62}$	$\mathbf{58.3 \pm 1.65}$	$\mathbf{86.9 \pm 0.77}$	$\mathbf{50.7 \pm 0.62}$
RESET	62.7 ± 2.18	84.4 ± 0.61	52.4 ± 0.85	72.7 ± 1.91	81.2 ± 0.36	53.4 ± 0.49
THRES-FEAT	55.3 ± 2.93	84.6 ± 1.01	49.8 ± 1.05	85.8 ± 1.74	67.4 ± 1.69	54.5 ± 1.33
THRES-AGGREAT	53.7 ± 2.48	84.2 ± 0.73	48.7 ± 1.16	85.4 ± 2.87	67.2 ± 0.66	54.5 ± 1.84
THRES-DEGREE	55.0 ± 3.26	85.0 ± 0.83	37.4 ± 0.80	83.7 ± 2.56	68.2 ± 1.49	39.7 ± 0.72
TRANSDUCTIVE	57.1 ± 4.42	95.6 ± 0.80	53.6 ± 1.46	82.5 ± 2.51	79.1 ± 1.07	54.3 ± 0.52
ORACLE	99.5 ± 0.21	92.9 ± 0.34	94.1 ± 0.30	98.4 ± 0.20	88.0 ± 0.43	86.4 ± 0.46
Amazon						
FIRST	52.7 ± 5.26	91.2 ± 1.04	50.9 ± 1.07	46.4 ± 3.23	90.9 ± 0.92	49.1 ± 1.33
CONTINUE	$\mathbf{55.5 \pm 3.97}$	$\mathbf{95.4 \pm 0.58}$	$\mathbf{55.4 \pm 1.19}$	$\mathbf{65.3 \pm 7.76}$	$\mathbf{89.3 \pm 1.95}$	$\mathbf{54.7 \pm 2.82}$
RESET	55.3 ± 3.10	95.1 ± 0.89	55.3 ± 0.85	56.5 ± 3.37	94.2 ± 0.59	53.2 ± 1.66
THRES-FEAT	55.9 ± 4.04	89.0 ± 0.62	65.7 ± 1.64	62.4 ± 6.70	90.2 ± 1.37	58.7 ± 3.05
THRES-AGGREAT	56.3 ± 2.63	95.2 ± 0.65	58.8 ± 1.88	62.0 ± 5.94	92.1 ± 1.54	54.5 ± 2.27
THRES-DEGREE	56.1 ± 3.17	95.4 ± 0.62	8.33 ± 0.36	57.0 ± 6.04	92.0 ± 2.10	9.67 ± 0.31
TRANSDUCTIVE	57.1 ± 4.42	95.6 ± 0.80	53.6 ± 1.46	58.8 ± 5.22	93.8 ± 0.92	54.9 ± 1.68
ORACLE	99.9 ± 0.08	98.4 ± 0.43	98.7 ± 0.61	99.4 ± 0.21	97.7 ± 0.28	96.5 ± 0.89
YelpChi						
FIRST	50.2 ± 1.41	84.1 ± 0.22	49.0 ± 0.54	46.4 ± 3.23	90.9 ± 0.92	49.1 ± 1.33
CONTINUE	$\mathbf{62.2 \pm 1.52}$	$\mathbf{95.6 \pm 0.31}$	$\mathbf{55.1 \pm 0.60}$	$\mathbf{77.2 \pm 2.07}$	$\mathbf{70.6 \pm 1.15}$	$\mathbf{56.1 \pm 0.58}$
RESET	67.0 ± 1.17	94.5 ± 0.30	57.1 ± 0.64	64.0 ± 2.44	85.8 ± 0.42	54.8 ± 0.80
THRES-FEAT	62.5 ± 1.31	96.2 ± 0.24	72.8 ± 0.87	76.5 ± 2.42	71.9 ± 1.05	63.9 ± 0.74
THRES-AGGREAT	62.7 ± 2.02	95.4 ± 0.30	55.4 ± 0.53	77.2 ± 1.59	71.0 ± 1.41	55.4 ± 0.46
THRES-DEGREE	62.1 ± 1.62	95.6 ± 0.30	56.2 ± 0.57	76.6 ± 1.88	71.2 ± 1.85	55.1 ± 0.69
TRANSDUCTIVE	61.4 ± 1.58	95.6 ± 0.25	54.7 ± 0.74	76.5 ± 1.81	86.2 ± 0.46	56.1 ± 0.40
ORACLE	99.9 ± 0.01	99.4 ± 0.06	99.1 ± 0.05	98.7 ± 0.16	93.0 ± 0.23	87.9 ± 1.68

*R^T - recall^0 (Final Recall): Cumulative recall of all fraud after the final round

*R^T - $f1^0$ (Initial F1): F1 Score of test set from round 0 after the final round

* R^T -$avgf1^{t>0}$ (On-round F1): F1 Score for test set of round t > 0 immediately after that round ends

6.3 Ablation and sensitivity studies

6.3.1 Model ablation

Table 7 contains results for our ablation study. We compare the full TPNE-XGB to the two variants described in Sect. 6.1.2 and three additional ones in which we set α, β, and both parameters to zero to see how the absence of the two temporal objectives affect performance. The three variants are respectively named **TPNE-XGB \ A**, **TPNE-XGB \ B**, and **TPNE-XGB \ AB**. For the ablation, we calculate the overall rank across all five datasets.

Both pre-training and temporal aggregation are necessary to achieve the best result. The PNE-XGB variant ranks last on every dataset with a significant decrease in initial F1. While the TNE-XGB variant shows relatively stronger performance, its overall rank is lower than the TPNE-XGB variants, especially in terms of final recall. For variants with partial pre-training objectives, we can see that the full TPNE-XGB utilizing all three objectives still outperforms everything else. From these, we see that omitting Temporal Disentanglement (TPNE-XGB\A) marginally improves final recall at the cost of initial F1. On the other hand, omitting Temporal Maximization (TPNE-XGB\B) simply degrades overall performance without a clear pattern. Omitting both marginally improves initial and on-round F1 at the cost of final recall.

6.3.2 Effects of α and β

Figure 6 shows the result of the sensitivity analysis experiments. The effect of Temporal Disentanglement, controlled by α, can be seen through the different plot hues. The two plots on the left show that increasing α resulted in higher recall and the two middle ones show that it also resulted

Table 7 Ablation results for all datasets: Tolokers, Amazon, YelpChi, TC and T-Finance

	Tolokers				Amazon			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank
TNE-XGB	77.7 ± 2.01	73.5 ± 1.03	52.7 ± 0.90	4.33	58.8 ± 3.36	92.1 ± 0.74	54.6 ± 2.26	2.67
PNE-XGB	80.5 ± 12.8	58.6 ± 5.17	52.1 ± 1.91	5.67	35.8 ± 16.0	52.2 ± 5.77	49.9 ± 1.47	6.00
TPNE-XGB\AB	86.1 ± 1.23	68.7 ± 1.09	54.9 ± 1.12	3.33	67.2 ± 2.30	87.4 ± 1.44	53.9 ± 2.01	3.33
TPNE-XGB\A	81.9 ± 2.45	75.6 ± 1.92	56.7 ± 0.50	2.00	54.9 ± 4.98	92.4 ± 1.95	52.7 ± 1.98	3.67
TPNE-XGB\B	86.1 ± 2.54	69.2 ± 1.47	55.2 ± 0.46	2.00	66.5 ± 5.12	87.8 ± 1.43	54.0 ± 1.75	3.00
TPNE-XGB	85.9 ± 3.36	69.1 ± 1.23	55.2 ± 1.67	3.00	65.3 ± 7.76	89.3 ± 1.95	54.7 ± 2.82	2.33
	YelpChi				TC			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank
TNE-XGB	67.8 ± 1.36	88.6 ± 0.80	56.4 ± 0.91	2.33	59.3 ± 2.36	88.9 ± 0.60	55.2 ± 1.10	3.67
PNE-XGB	79.6 ± 23.4	48.0 ± 1.71	48.0 ± 0.96	4.33	86.1 ± 1.82	67.9 ± 1.09	52.5 ± 0.44	4.33
TPNE-XGB\AB	77.1 ± 1.54	67.9 ± 1.71	54.3 ± 0.66	4.33	82.1 ± 1.08	79.7 ± 0.98	56.4 ± 0.55	2.67
TPNE-XGB\A	64.5 ± 2.80	80.2 ± 2.56	54.9 ± 0.69	4.00	80.0 ± 2.03	80.2 ± 0.80	55.8 ± 1.07	3.00
TPNE-XGB\B	75.2 ± 1.70	71.6 ± 0.99	55.7 ± 0.36	3.33	82.8 ± 1.40	77.5 ± 0.96	55.2 ± 0.77	4.00
TPNE-XGB	77.2 ± 2.07	70.6 ± 1.15	56.1 ± 0.58	2.67	83.8 ± 1.54	78.8 ± 0.74	55.5 ± 0.98	3.00
	T-Finance				Overall Rank			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank				
PNE-XGB	54.8 ± 2.06	89.1 ± 0.72	54.5 ± 1.26	3.33	3.27			
TNE-XGB	62.0 ± 9.90	68.0 ± 13.9	53.5 ± 1.81	5.33	5.13			
TPNE-XGB\AB	63.2 ± 1.83	85.9 ± 1.76	54.2 ± 0.57	3.33	3.30			
TPNE-XGB\A	58.0 ± 1.81	90.6 ± 0.78	54.0 ± 1.10	3.67	3.27			
TPNE-XGB\B	62.3 ± 2.58	88.4 ± 0.92	54.3 ± 1.35	3.00	3.07			
TPNE-XGB	62.1 ± 2.64	88.8 ± 0.83	54.8 ± 1.19	2.33	2.67			

*R^T - recall^0 (Final Recall): Cumulative recall of all fraud after the final round

*R^T - $f1^0$ (Initial F1): F1 Score of test set from round 0 after the final round

* R^T -$avgf1^{t>0}$ (On-round F1): F1 Score for test set of round t > 0 immediately after that round ends

in lower initial F1 (i.e. more forgetting). The two plots on the right show different results for on-round F1: higher α leads to lower on-round F1 in Tolokers while the opposite is true in Amazon. The effect of Temporal Maximization, controlled by β, can be seen in Fig. 6 on the X-axis. The top two plots show that final recall tends to slightly decline with β and the two middle plots show that initial F1 conversely tends to increase. This directly mirrors the effect of Temporal Disentanglement (α).

7 Discussion

This section contains in-depth discussions on our scenario and proposed model in relation to the results presented above. Additionally, we also provide additional details regarding the implications, limitations, and complexity of our work/model given the underlying assumptions it adopts.

7.1 Fraud detection behavior under the proposed scenario

Results shown in Sect. 6.2.1 indicate that existing models struggle to perform in subsequent rounds, which is expected since they were developed in a single round setting. However, the fact that they fail despite having access to both the entire graph structure and labels from past rounds for retraining is evidence for the following: adapting models to a multi-round fraud detection setting is nontrivial. As such, research on fraud detection beyond a single detection round is invaluable.

Certain qualities are important to perform in this new, multi-round scenario. The proposed generator function is a relatively straightforward simulator of a promo fraud (Ekata 2023) which tracks undetected fraud and spams them. As previously mentioned, we see that despite its simplicity this generator function challenges the fraud detectors to recognize and correct past mistakes. This capability is critical to real-life fraud detection where fraudsters adapt to existing detectors and continually change their strategies. We can

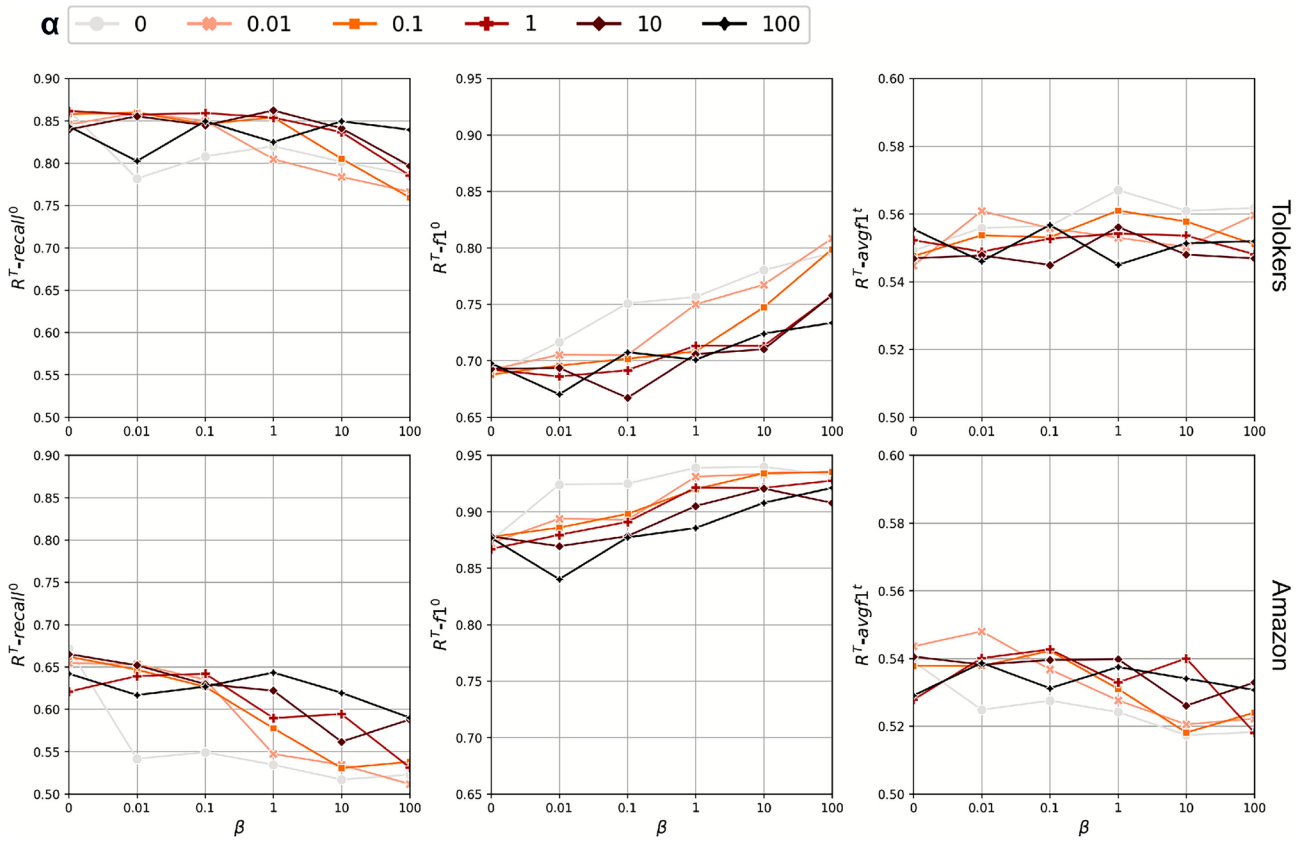

Fig. 6 Sensitivity plot for the three performance metrics on different α (line hue) and β (X axis) on the Tolokers (top) and Amazon dataset (bottom)

further break it down into two more specific capabilities: the capability to memorize relevant knowledge and the capability to forget irrelevant ones.

The baselines lack the second capability under a naive retraining scheme. The main indicators for this are low final recall over multiple rounds (R^T-$recall^0$) and low on-round F1 (R^T-$avgf1^{t>0}$) as seen in Tables 4 and 5. Prime example of this is the vanilla XGB model, which consistently places in the bottom two for final recall in the large datasets. Here, the model simply learns about the original fraud in the initial round and memorizes any learned knowledge to the end no matter if it's right or wrong. We also see that the availability of the entire dataset every round can be detrimental, especially for nodes with strong initial F1; re-learning the same sets of data reinforces and solidifies mistakes. The best-performing baseline (XGB-GIN) is the prime example of this with its strong initial F1 ($R^T - f1^0$) and low final recall.

Meanwhile, we see that simple models tend to behave erratically (for example, see GraphSAGE's performance for the Amazon dataset in Fig. 5). Their F1 scores oscillate with a large deviation, which corresponds to an unusually high R^T-$recall^0$ when we cross-check with Table 4. Examining the execution logs shows that these simple models often collapsed into a state where almost all nodes are predicted as fraudulent for one round, resulting in a very high type 1 error, low F1 score, and high final recall. This collapse additionally provided the labels for all the nodes and bumped the performance of the next round. This is an undesired behavior for real fraud detection systems. With regard to error types, we found that other than this erratic behavior due to a bias towards type 1 error, there is no interesting pattern to note. No natural tendency towards a certain type of error was observed, especially for the higher-performing XGB-based models.

In summary, the proposed Multi-round Adversarial Fraud Detection scenario poses a new set of challenges that are natural and unique to fraud detection but have yet to be solved by existing models. As explained in Sect. 2, the phenomenon of forgetting over multiple training is also a central focus of continual learning research. However, our scenario

differs in the sense that due to the adversarial nature of the generative function, we require a specific type of forgetting.

7.2 On self-supervised pre-training and temporal aggregation

The proposed TPNE-XGB exhibited better performance over multiple rounds. It successfully forgets past misclassifications to better respond to newly generated fraud, as shown by the high on-round F1 and final recall, accompanied by slightly lower initial F1 seen in Tables 4 and 5. The relatively lower deviation value suggests that TPNE-XGB achieves a higher degree of consistency compared to its contemporaries. Judging from the ablation result in Sect. 6.3, we believe both self-supervised pre-training and temporal aggregation of TPNE-XGB are important to achieve this.

The objective of the pre-training step in TPNE-XGB is not to detect fraud but to learn a more informative representation to improve downstream task performance. This means that in every round, TPNE-XGB learns to do two different things using two distinct sources of information. Most baselines learn solely to detect fraud from label supervision, whereas TPNE-XGB also learns from the overall graph structure. This is the advantage given by SSL; while supervision from labels stays identical throughout subsequent rounds, the local graph structure doesn't. TPNE-XGB suffers less from harmful reinforcement of misclassifications due to changes in the embeddings generated by the embedder module, resulting in better final recall and on-round F1. Additionally, Table 6 shows that resetting the model from scratch did not improve results for TPNE-XGB. This is because TPNE can already forget irrelevant information.

Self-supervised pre-training alone is not enough to achieve a good balance of on-round F1, final recall, and initial F1. As indicated in Table 7, pre-training using a simpler reconstruction loss (PNE-XGB) causes the model to learn less in the initial round (low R^T-$f1^0$). The temporal aggregation in TPNE-XGB alleviates this by aggregating information from different neighborhood hops and rounds, providing the model with richer features to learn. Finally, the hyperparameters α and β control the informativeness of the temporal embedding part, which is primarily utilized to selectively forget mistaken classifications. The sensitivity analysis roughly supports our intuition w.r.t. the temporal embedding part. As it becomes more disentangled (higher α), TPNE-XGB becomes more reliant on the temporal embedding part and tends to forget the initially learned pattern from the static part.

In summary, splitting the learning process for fraud detection in two, each with a different objective and supervision (i.e., self-supervised then supervised), allows TPNE-XGB to rely less on label supervision and in turn suffer less from memorizing misclassifications. This improves on-round F1 and final recall at the cost of only a slight degradation in initial F1. The temporal aggregation, along with the three-fold self-supervised learning objective, controls the balance between learning and forgetting and is critical in achieving optimal performance.

While TPNE-XGB managed to exhibit favorable behavior in our scenario, it does come with several weaknesses/limitations. The first one can be seen in the results from Elliptic++ shown in Table 5. Using XGB-based model increases the tendency for the model to overfit, especially when structural information is scarce. The second weakness it inherits from the GCN backbone is the inability to capture information from distant neighbors and unconnected nodes. As a result, spectral-based approaches like GHRN and BWGNN can outperform TPNE-XGB in some cases. Finally, the pre-training phase incurs a non-neglectable computational cost. Therefore, its scalability to extremely large datasets is closer to common GNNs rather than XGBoost. This can also be seen on the empirical runtime analysis provided in Sect. 7.4.

7.3 Theoretical and practical implications

Our work confirms that extending fraud detection research to a multi-round, adversarial scenario requires a non-trivial adaptation of existing models. Specifically, our experiment results indicate that the key to good performance here is the ability to selectively forget and retain information throughout multiple rounds. The behavior of our proposed TPNE-XGB model reveals that self-supervised representation learning and the utilization of temporal information produce that ability to a degree. Additionally, our experiment results validate the previous benchmarking effort on supervised GAD in which a simple XGB-based model outperforms state-of-the-art GAD models.

This carries several implications. In the theoretical sense, it reveals the shortcomings of past graph fraud detection works when dealing with a more complicated fraud detection scenario. We see that fraud generated by a simple adversarial function are capable of deceiving recent fraud detection models. Future works would benefit greatly from taking an adversarial lens in the vein of (Ren et al. 2020; Wu et al. 2023b, 2024). Additionally, this work highlights the importance of evaluating multiple aspects of performance (i.e. final recall, initial F1, on-round F1) since improving one usually means a trade-off on the other(s). Finally, results from TPNE-XGB show the strength of a two-step learning paradigm that which allows combining a flexible self-supervised embedder with a lighter classifier like XGBoost. As also recently shown (Fini et al. 2022), self-supervised learners exhibit more robustness through

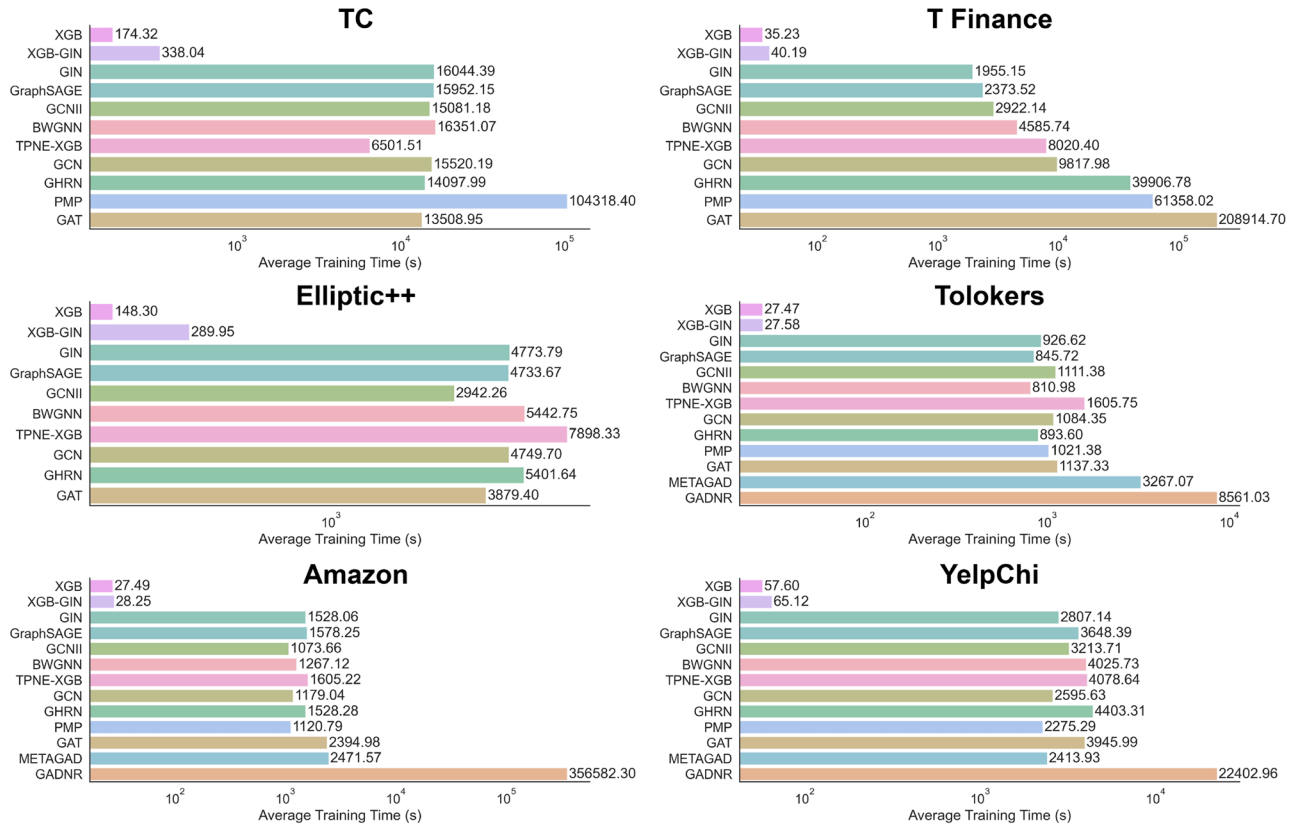

Fig. 7 Training time per round (in seconds) for all models in the main experiment set. Shown numbers are a result of averaging across all 10 detection rounds and all 10 repetitions

multiple learning processes. Future fraud detection modeling will benefit greatly by focusing on a self-supervised embedder backbone.

In the practical sense, results from the experiments with simple strategies in Sect. 6.2.2 suggest that companies should move beyond simple threshold-based strategies for promo fraud due to their tendency to produce false positives. Modern fraud patterns are complex and adaptable. As such, their detection often demands detailed and sophisticated models. With regard to data collection policies, we note that obtaining reliable fraud labels remains the most rewarding endeavor. However, we advise companies to collect labels multiple times over a long period as opposed to a single instance of collection even if done on a larger scale. As shown through our work, developing a fraud detection model through a single training on a large amount of data leaves the model vulnerable even to simple adversarial adaptation.

To conclude this subsection, we touch on how TPNE adds a layer of practical interpretability in fraud prediction. One of the implications of the explicit separation between static and temporal information in the embedding produced by TPNE is that by employing established feature-based interpretation techniques, we can weigh the importance of the two different information type for a given prediction. This is also only possible due to TPNE decoupling the embedder with the classifier, which allows us to potentially use a simpler and more interpretable model to understand how the two different information are used in predicting fraud.

7.4 Runtime and complexity analysis

TPNE uses GCN layers as its main building block, and its three learning objectives are relatively straightforward to calculate. As such, training and inference using TPNE do not incur a significantly higher overhead compared to simple GNN-based fraud detectors. To empirically show this, we record the average training time per round for all experiments shown in Tables 4 and 5. The result is shown in Fig. 7.

As can be seen, XGBoost-based baselines with non-parametric aggregations (XGB, XGB-GIN) with $O(|V|)$ time complexity (Chen and Guestrin 2016) offer the fastest training time, up to three orders of magnitude lower than the other models. Trailing after simple GNN and spectral-based models (GCN, GIN, GraphSAGE, GCNII, BWGNN, GHRN, TPNE-XGB), which have relatively similar training durations. The slight variation among them is due to

the usage of early stopping in our experiments. However, averaging across all rounds and all repetitions should give a relatively robust estimate on the actual runtime of these models in the wild. Among the slower models are those that incorporate edge-wise calculations like GAT's attention mechanism and PMP. This slowdown is most apparent in T-finance, which has the most graph edges. In particular, PMP reported $O(L|V|F + (L|E| + |V|)F^2)$ time complexity. The multiplication term between $|E|$ and F^2 renders it less feasible on datasets with a lot of edges. Finally, MetaGAD with its bi-level optimization and GAD-NR with its sampling-based neighborhood reconstruction require the most time and space among the baselines.

To support the empirical observation above, we can analyze the time complexity of training TPNE by breaking it down into steps as described in Algorithm 3. Let F denote both the number of features and hidden dimension for TPNE, then:

- Generating $H_0^{t.w}$ for all W windows (line 18) is $O(|V|F^2)$
- Generating $H_l^{t.w}$ for all L GCN layers in all W windows (line 18) is $O(WL|V|F^2 + WL|E|F)$
- Generating H_{temp}^t is also $O(WL|V|F^2)$
- Combined, the forward pass have $O(W|V|F^2 + WL|V|F^2 + WL|E|F + WL|V|F^2) = WL|V|F^2 + WL|E|F$

In practice, W, L, and F are sufficiently small constants. Therefore, the time complexity of TPNE can be expressed as $O(|V| + |E|)$, identical to a GCN. This means that in terms of scalability to extremely large graph, TPNE-XGB will still struggle as it is bound by the number of nodes and edges. For GNNs, sampling-based approach such as GraphSAGE can be a possible solution. For TPNE-XGB, an additional sampling and batched temporal aggregation mechanism is required in the future. We note here that our method still scales better than more sophisticated GAD solutions such as MetaGAD and GAD-NR.

7.5 Assumptions and limitations

Here we reiterate the main assumptions we use in defining the scope of this work. First, this work specifically focuses on node injection-based evasion attacks for node classification. This is because common promo fraud types like coupon and referral abuse are mostly done through mass creation of new user accounts i.e., node-injection. While many other attack types such as poisoning attacks, anomalous edge detection, feature perturbations, and node/edge deletion are equally valuable to examine, we expect that focusing on the most representative attack type for promo fraud will provide readers with better clarity in understanding the topic. Related to this point, considering how TPNE succeeded in providing salient temporal information, we surmise that its performance on different attacks and tasks should at least surpass baselines designed for static graphs.

Second, we assume clean labels on the initial training graph. While procuring a training dataset with clean label can be costly, it is definitely something that can be and is often produced in practice. For a domain where reliability and trust is vital like finance, companies do routinely employ services to produce data with clean labels. We follow this assumption following the majority of prior works to promote consistency in the research canon. One important thing to note in relation to this is the fact that in our scenario, subsequent training rounds actually involve unlabeled nodes. This makes our scenario more realistic than most contemporaries mentioned in Sect. 2.1. We acknowledge that real life datasets are often noisy and as such, performance validation under label noise remains a limitation we have yet to address here.

While we acknowledge the many kinds of fraudulent behavior in the wild and the possibility of less-than-ideal data condition, each of these issues exhibits fundamentally different dynamics and demands different things from the detector model. As such, limiting the work to a single type of fraud under a set of reasonable assumptions helped us in producing a reasonable and focused experiment sets and analysis. We see this as something necessary, especially considering that this work primarily aims to introduce the Multi-round Adversarial Fraud Detection.

7.6 Extensions and future works

We recognize that several design choices for both our Multi-round Adversarial Fraud Detection scenario and TPNE are rather simplistic in nature due to their newness. With regard to this, we would like to point out that extension of either the scenario or model can serve as interesting avenues for future works.

Extension of the scenario can be done through a more sophisticated generator function involving learners (resulting in a GAN-like arrangement on a training round granularity instead of training epoch) or probabilistic graphical models. Alternatively, the function can be guided with common fraud characteristics like the label heterophily of common fraud or feature-based camouflaging of money laundering. This way, a multitude of other fraud types can be studied under our scenario. Furthermore, since we decouple the generators from the detection models, exploring these new generator variants should require no adjustments to the other scenario components, like the detector model. In fact, it is possible to benchmark any existing

Table 8 Results for several well-known CGL methods on the smaller datasets: Tolokers, Amazon, and YelpChi

	Tolokers				Amazon			
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank
EWC	58.6 ± 5.69	68.8 ± 0.55	46.1 ± 1.00	4.67	45.0 ± 5.40	69.7 ± 3.99	50.1 ± 3.05	3.00
LWF	74.3 ± 2.65	67.8 ± 2.94	49.5 ± 0.97	3.00	71.7 ± 23.9	54.6 ± 14.6	47.8 ± 6.58	4.00
TWP	70.6 ± 6.22	63.4 ± 1.01	47.1 ± 1.37	5.33	41.6 ± 3.83	62.8 ± 1.61	48.5 ± 5.32	4.33
ER-GNN	73.8 ± 5.61	64.8 ± 7.54	49.7 ± 1.66	3.67	62.9 ± 18.4	60.9 ± 9.08	50.0 ± 2.75	3.67
CGNN	81.5 ± 10.5	66.4 ± 6.46	48.1 ± 1.94	3.33	$\mathbf{83.7 \pm 17.7}$	49.6 ± 28.3	45.3 ± 1.71	4.33
TPNE-XGB	$\mathbf{85.9 \pm 3.36}$	$\mathbf{69.1 \pm 1.23}$	$\mathbf{55.2 \pm 1.67}$	**1.00**	65.3 ± 7.76	$\mathbf{89.3 \pm 1.95}$	$\mathbf{54.7 \pm 2.82}$	**1.67**
	YelpChi							
	R^T-$recall^0$	R^T-$f1^0$	R^T-$avgf1^{t>0}$	Rank				
EWC	35.9 ± 16.2	56.5 ± 2.64	45.5 ± 0.86	3.67				
LWF	46.9 ± 13.8	51.8 ± 4.25	45.8 ± 1.11	3.67				
TWP	34.6 ± 3.72	55.4 ± 0.70	46.4 ± 0.45	3.67				
ER-GNN	68.2 ± 16.0	50.4 ± 9.93	43.0 ± 2.23	5.00				
CGNN	$\mathbf{79.0 \pm 18.8}$	51.3 ± 12.8	44.6 ± 1.27	3.67				
TPNE-XGB	77.2 ± 2.07	$\mathbf{70.6 \pm 1.15}$	$\mathbf{56.1 \pm 0.58}$	**1.33**				

*R^T - recall^0 (Final Recall): Cumulative recall of all fraud after the final round

*R^T - fl^0 (Initial F1): F1 Score of test set from round 0 after the final round

* R^T -$avgfl^{t>0}$ (On-round F1): F1 Score for test set of round t > 0 immediately after that round ends

models implemented under our scenario against any future generator functions.

With regard to TPNE, it is possible to extend it by replacing the current GCN building blocks with a more powerful module, e.g., transformer-based GNNs. Another modification could involve allocating different modules for different windows. This way, each time period will have its own model/attention weights, and theoretically a richer temporal information could be captured at the cost of added complexity. Alternatively, different temporal aggregation methods can be explored.

Direct continuation of this work can take many directions. The most straightforward one is just to try and achieve better performance using the exact generator function for promo fraud. This may be possible if we focus on model design with greater control over what is being forgotten and what is being memorized. We believe that a self-supervised backbone is necessary for this. Alternatively, other generator functions that produce different fraud behaviors could be studied as explained above. Outside the exploration of the generator function, refining the scenario with additional constraints to make it more closely reflect real life can be a valuable research direction. As previously stated, this could be in the form of limitations to the training data, such as making nodes expire after a certain number of rounds. Finally, as noted in Sect. 6.1.2, unified comparisons across the field of CGL, DGL, and this new scenario can also be a valuable direction for future works.

8 Conclusion

In this paper, we describe a new scenario for fraud detection called the Multi-round Adversarial Fraud Detection scenario. This scenario incorporates the temporal and adversarial nature of fraud detection, opening up a new perspective for future fraud detection model development. Under the scenario, we implement a simple spam-like adversarial strategy and show that it poses a significant challenge to existing baselines which produces an average round-wise F1 score of 44.1−55.4 on subsequent training rounds. We also provide an exemplar solution for that challenge in the form of a label and model-agnostic node embedder called TPNE. The key idea of TPNE is to leverage temporal information through a sliding window aggregator, and then maximize its usefulness through a disentanglement and maximization term in a self-supervised pre-training step. Extensive experiments on a diverse selection of datasets prove that TPNE can produce a more favorable fraud detection behavior compared to the baselines, exceeding the best baseline's on-round F1 by up to 4.6 percent and its overall final recall by up to 32.9 percent. A deeper analysis of the experiment results reveals the central problems that exist in the Multi-round Adversarial Fraud Detection scenario: the capture of temporal patterns and the forgetting/retaining of learned detection patterns over several rounds.

Table 9 Search space for hyperparameter optimization; variation between datasets is due to time constraints

Model	Hyperparameter	Search Space
Shared		
GNN parameters	Layers	[2]
(for all GNN models	Hidden/embedding dim	[64]
unless listed below)	Activation	[ReLU]
Model Specific		
GraphSAGE	aggregation	[mean]
GIN	aggregation	[mean]
GAT	attention head	[1, 3, 5, 7] *(Amazon, Tolokers, Yelp)*
		[2, 4, 6] *(TC)*
		[2] *(T-Finance)*
GHRN	drop rate	[0.01, 0.025, 0.05, 0.1, 0.25] *(Amazon, Tolokers, Yelp)*
		[0.01, 0.05, 0.25] *(T-Finance, TC)*
GAD-NR	hidden dim	[8, 16]
XGB & XGB-GIN	booster	[gbtree]
	eta	[0.3]
	gamma	[0]
	max tree depth	[6]
TPNE-XGB	alpha	[0, 0.01, 0.1, 1, 10, 100, 1000] *(Amazon, Tolokers, Yelp)*
		[0, 1, 10, 100] *(T-Finance, TC)*
	beta	[0, 0.01, 0.1, 1, 10, 100] *(Amazon, Tolokers, Yelp)*
		[0, 1, 10, 100] *(T-Finance, TC)*

Appendix A: Additional experimental results

Here, we provide results produced by running a number of well-known Continual Learning (CL) and Continual Graph Learning (CGL) approaches under our scenario. This includes regularization-based approaches such as EWC (Kirkpatrick et al. 2017), LWF (Li and Hoiem 2017), and TWP (Liu et al. 2021a), replay-based approaches such as ER-GNN (Zhou and Cao 2021), and hybrid ones such as CGNN (Wang et al. 2020). For these approaches, the training sets for subsequent rounds are limited to newly generated nodes to mimic the common CGL setting. Considering how our scenario requires forgetting, the expectation is that this limitation will actually prevent CGL model from overfitting to achieve higher on-round F1. Full results can be found in Table 8 below.

It is immediately apparent from the low numbers for all metrics that CGL models fit poorly to our scenario. Indeed, all of them are designed to memorize past learnings instead of correcting them. Note that their memorization ability fails on a long horizon (10 rounds) we adopted in our experiments as shown by the low initial F1, but this also doesn't allow them to learn any meaningful temporal pattern to discriminate newly generated fraud as shown by the low on-round F1. Overall, they mostly ended up detecting less frauds overall in addition to being less stable as shown by the high deviation in the recall number.

Appendix B: Additional details

B.1 Parameter search space

The default values and corresponding search space for our hyperparameter optimization can be found in Table 9. Due to the larger computational load for the bigger datase and the multi-round nature of our work, the parameter search space presented was first narrowed down qualitatively.

B.2 Hardware specifications

The experiments outlined in Section 5 were done in two distinct machines:

- A linux server equipped with an Intel Core i9-10900X, 96GB or RAM, and an NVIDIA RTX A6000 GPU with a 48GB memory.
- A Fujitsu Primergy GX2570 M6 compute node in ABCI 2.0 (https://docs.abci.ai/en/system-overview/); Intel Xeon®Processor Gold 5318Y, 16GB of RAM, and an 80GB NVIDIA HGX A100 GPU.The former is used to produce all results involving MetaGAD, GAD-NR, and PMP in Table 4; all but the last row of Table 5; and the first two rows from the result for the TC dataset and T-Finance in Table 7. The same version of Pytorch, DGL, and CUDA Toolkit is used even between different machines. For some experiment sets that required more GPU memory than what is available, we switch to training and inference on CPU.

Appendix C: Pseudocodes

Below are pseudocodes for the threshold-based strategy used for the experiments discussed in Sect. 6.2.2. ONEHOPSUBGRAPH returns a subgraph containing the 1-hop neighborhood of the input nodes, top_vs. ACTIVENODES(G, w, t) returns a list of nodes created within the last w rounds as of round t. ITERATIVEDBSCAN (V_{act}, H_{act}, p) returns the top p densest nodes using the DBSCAN algorithm (Ester et al. 1996).

Input: Graph G, current round number t, window size w, top percentile threshold p
Output: Set of spam suspect nodes V_{spam}
  function THRES-FEAT(G, t, w, p)
    $\{V, E, X, Y\} \leftarrow G$
    if r == 0 **then** ▷ Do nothing on initial round
      return None
    else
      $act\_vs \leftarrow$ ACTIVENODES(G, w, t)
      $G_{act} \leftarrow$ ONEHOPSUBGRAPH(G, act_vs)
      $\{V_{act}, E_{act}, X_{act}, Y_{act}\} \leftarrow G_{act}$
      $V_{spam} \leftarrow$ ITERATIVEDBSCAN(V_{act}, X_{act}, p)
    return V_{spam}

Algorithm 4 Feature-based Spam Filter Strategy

Input: Graph G, feature aggregator F, current round number t, window size w, top percentile threshold p
Output: Set of spam suspect nodes V_{spam}
  function THRES-AGGFEAT(G, F, t, w, p)
    $\{V, E, X, Y\} \leftarrow G$
    if r == 0 **then** ▷ Do nothing on initial round
      return None
    else
      $act\_vs \leftarrow$ ACTIVENODES(G, w, t)
      $G_{act} \leftarrow$ ONEHOPSUBGRAPH(G, act_vs)
      $\{V_{act}, E_{act}, X_{act}, Y_{act}\} \leftarrow G_{act}$
      $H_{act} \leftarrow F(G_{act})$ ▷ Get aggregated feature
      $V_{spam} \leftarrow$ ITERATIVEDBSCAN(V_{act}, H_{act}, p)
    return V_{spam}

Algorithm 5 Aggregated Feature Threshold-based Spam Filter Strategy

Input: Graph G, current round number t, window size w, top percentile threshold p
Output: Set of spam suspect nodes V_{spam}
  function THRES-DEGREE(G, t, w, p)
    $\{V, E, X, Y\} \leftarrow G$
    if r == 0 **then** ▷ Do nothing on initial round
      return None
    else
      $act\_vs \leftarrow$ ACTIVENODES(G, r, w)
      $G_{act} \leftarrow$ ONEHOPSUBGRAPH(G, act_vs)
      $\{V_{act}, E_{act}, X_{act}, Y_{act}\} \leftarrow G_{act}$

      $degrees \leftarrow$ DEG(G, V_{act}) ▷ Get node degrees on original graph
      $thr \leftarrow$ PERC$_{1-p}$($degrees$) ▷ Get threshold

      $top\_vs \leftarrow \emptyset$
      for $v \in V_{act}$ **do** ▷ Get nodes with degree above threshold
        if $deg(v) \geq thr$ **then**
          $top\_vs \leftarrow top\_vs \cup v$

      $G_{sus} \leftarrow$ ONEHOPSUBGRAPH(G_{act}, top_vs)
      $\{V_{sus}, E_{sus}, X_{sus}, Y_{sus}\} \leftarrow G_{sus}$

      $V_{spam} \leftarrow act\_vs \cap V_{sus}$ ▷ Spams are active nodes in G_{sus}
      return V_{spam}

Algorithm 6 Degree Threshold Spam Filter Strategy

Author Contributions HAP conceptualized the work, implemented/performed the experiments, and wrote the first manuscript draft. LX, TM, and AM provided supervision and resources throughout the work. All authors discussed the results, reviews, and contributed to the final manuscript.

Funding This paper is based on results obtained from the project,JPNP25006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO), JSPS Grant-in-Aid for Scientific Research (grant number 23H03451, 25K03231), and Indonesia Endowment Fund for Education (LPDP).

Data Availability Datasets, code for preprocessing, model training and evaluation are available on https://github.com/hafizhadi/multiround-promo-fraud.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Consent for publication Not applicable.

Materials availability Not applicable.

Ethics approval and consent to participate Not applicable.

References

Alghamdi J, Luo S, Lin Y (2024) A comprehensive survey on machine learning approaches for fake news detection. Multimedia Tools and Applications 83(17):51009–51067

Alnabhan MQ, Branco P (2024) Fake news detection using deep learning: A systematic literature review. IEEE Access

Baik S, Hong S, , et al (2020) Learning to forget for meta-learning. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), p 2376–2384

Cao Y, Yang J (2015) Towards making systems forget with machine unlearning. In: IEEE Symposium on Security and Privacy, p 463–480

Chakraborty J, Xia W, Majumder A, et al (2024) Detoxbench: Benchmarking large language models for multitask fraud & abuse detection. arXiv preprint arXiv:2409.06072

Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 785–794

Chen L, Li J, Peng J, et al (2020a) A survey of adversarial learning on graphs. arXiv preprint arXiv:2003.05730

Chen M, Wei Z, Huang Z, et al (2020b) Simple and deep graph convolutional networks. In: Proceedings of the 37th International Conference on Machine Learning, p 1725–1735

Dou Y, Liu Z, Sun L, et al (2020) Enhancing graph neural network-based fraud detectors against camouflaged ffaudsters. In: Proceedings of the 29th ACM International Conference on Information & Knowledge management, p 315–324

Ekata (2023) Reining in promo fraud: A data-driven approach for distinguishing between valid and fraudulent activity. White paper

Elmougy Y, Liu L (2023) Demystifying fraudulent transactions and illicit nodes in the bitcoin network for financial forensics. In: Proceedings of the 29th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 3979–3990

Ester M, Kriegel HP, Sander J, et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. Knowledge Discovery and Data Mining p 226–231

Fang J, Wen H, Wu J et al (2024) Gani: Global attacks on graph neural networks via imperceptible node injections. IEEE Transactions on Computational Social Systems 11(4):5374–5387

Fini E, da Costa VGT, Alameda-Pineda X, et al (2022) Self-supervised models are continual learners. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, p 9621–9630

Gao Y, Wang X, He X et al (2023) Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. Proceedings of the ACM Web Conference 2023:1528–1538

Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, p 1025–1035

Hendrycks D, Mazeika M, Kadavath S, et al (2019) Using self-supervised learning can improve model robustness and uncertainty. Advances in neural information processing systems 32

Hilal W, Gadsden SA, Yawney J (2022) Financial fraud: a review of anomaly detection techniques and recent advances. Expert Syst Appl 193:116429

Hyun W, Lee J, Suh B (2023) Anti-money laundering in cryptocurrency via multi-relational graph neural network. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, p 118–130

Hyun W, Lee I, Suh B (2024) Lex-gnn: Label-exploring graph neural network for accurate fraud detection. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, p 3802–3806

Jin W, Li Y, Xu H et al (2021) Adversarial attacks and defenses on graphs. ACM SIGKDD Explorations Newsl 22(2):19–34

Kazemi SM, Goel R, Jain K et al (2020) Representation learning for dynamic graphs: A survey. J Mach Learn Res 21(70):1–73

Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907

Kirkpatrick J, Pascanu R, Rabinowitz N et al (2017) Overcoming catastrophic forgetting in neural networks. Proc Natl Acad Sci 114(13):3521–3526

Korkanti S (2024) Enhancing financial fraud detection using llms and advanced data analytics. In: 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems, p 1328–1334

Kurshan E, Shen H (2020) Graph computing for financial crime and fraud detection: Trends and challenges and outlook. International Journal of Semantic Computing 14(04):565–589

Li Z, Hoiem D (2017) Learning without forgetting. IEEE Trans Pattern Anal Mach Intell 40(12):2935–2947

Li Y, Yang X, Gao Q et al (2024) Cross-regional fraud detection via continual llearning with tnowledge transfer. IEEE Trans Knowl Data Eng 36(12):7865–7877

Li Z, Chen D, Liu Q, et al (2022c) The devil is in the conflict: Disentangled information graph neural networks for fraud detection. In: 2022 IEEE International Conference on Data Mining (ICDM), p 1059–1064

Li J, Gao Y, Lu J, et al (2025) Diffgad: A diffusion-based unsupervised graph anomaly detector. In: 13th International Conference on Learning Representations

Li K, Liu Y, Ao X, et al (2022b) Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In: Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 925–935

Liu Y, Ao X, Qin Z et al (2021) Pick and choose: a gnn-based imbalanced learning approach for fraud detection. Proceedings of the Web Conference 2021:3168–3177

Liu Y, Jin M, Pan S et al (2022) Graph self-supervised learning: A survey. IEEE Trans Knowl Data Eng 35(6):5879–5900

Liu F, Ma X, Wu J, et al (2022a) Dagad: Data augmentation for graph anomaly detection. In: 2022 IEEE International Conference on Data Mining (ICDM), p 259–268

Liu H, Yang Y, Wang X (2021a) Overcoming catastrophic forgetting in graph neural networks. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence, p 8653–8661

Li H, Wang X, Zhang Z, et al (2022a) Out-of-distribution generalization on graphs: A survey. arXiv preprint arXiv:2202.07987

Ma X, Liu F, Wu J et al (2025) Rethinking unsupervised graph anomaly detection with deep learning: Residuals and objectives. IEEE Trans Knowl Data Eng 37:881–895

Malone C (2023) Combatting online payment fraud. White paper, Juniper Research Ltd

McAuley JJ, Leskovec J (2013) From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In: Proceedings of the 22nd International Conference on World Wide Web, p 897–908

McCloskey M, Cohen NJ (1989) Catastrophic interference in connectionist networks: The sequential learning problem. Psychol Learn Motiv 24:109–165

Mienye ID, Jere N (2024) Deep learning for credit card fraud detection: A review of algorithms, challenges, and solutions. IEEE Access 12:96893–96910

Pan J, Liu Y, Zheng X, et al (2025) A label-free heterophily-guided approach for unsupervised graph fraud detection. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence, p 12443–12451

Phua C, Lee V, Smith K, et al (2010) A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119

Platonov O, Kuznedelev D, Diskin M, et al (2023) A critical look at the evaluation of gnns under heterophily: Are we really making progress? arXiv preprint arXiv:2302.11640

Qiao H, Pang G (2024) Truncated affinity maximization: One-class homophily modeling for graph anomaly detection. In: Proceedings of the 37th International Conference on Neural Information Processing Systems, p 49490–49512

Rayana S, Akoglu L (2015) Collective opinion spam detection: Bridging review networks and metadata. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 985–994

Ren L, Hu R, Li D et al (2023) Dynamic graph neural network-based fraud detectors against collaborative fraudsters. Knowl-Based Syst 278:110888

Ren Y, Wang B, Zhang J, et al (2020) Adversarial active learning based heterogeneous graph neural network for fake news detection. In: 2020 IEEE International Conference on Data Mining (ICDM), p 452–461

Riskified (2023) Policy abuse and its impacts on merchants - global benchmarks 2023. Tech. rep

Roy A, Shu J, Li J, et al (2024) Gad-nr: Graph anomaly detection via neighborhood reconstruction. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, p 576–585

Shi F, Cao Y, Shang Y et al (2022) H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. Proceedings of the ACM Web Conference 2022:1486–1494

Shibata T, Irie G, Ikami D, et al (2021) Learning with selective forgetting. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence, p 989–996

Sun L, Dou Y, Yang C et al (2022) Adversarial attack and defense on graph data: A survey. IEEE Trans Knowl Data Eng 35(8):7693–7711

Tang J, Hua F, Gao Z, et al (2023) Gadbench: Revisiting and benchmarking supervised graph anomaly detection. In: Proceedings of the 37th International Conference on Neural Information Processing Systems, p 29628–29653

Tang J, Li J, Gao Z, et al (2022) Rethinking graph neural networks for anomaly detection. In: Proceedings of the 39th International Conference on Machine Learning, p 21076–21089

Tao Q, Liao J, Zhang E et al (2024) A dual robust graph neural network against graph adversarial attacks. Neural Netw 175:106276

Tian Y, Liu G (2024) Spatial-temporal-aware graph transformer for transaction fraud detection. IEEE Trans Industr Inf 20(11):12659–12668

Veli?kovi? P, Cucurull G, Casanova A, et al (2018) Graph attention networks. In: 6th International Conference on Learning Representations

Wang S (2010) A comprehensive survey of data mining-based accounting-fraud detection research. In: 2010 International Conference on Intelligent Computation Technology and Automation, vol 1, p 50–53

Wang Y, Zhang J, Huang Z et al (2023) Label information enhanced fraud detection against low homophily in graphs. Proceedings of the ACM Web Conference 2023:406–416

Wang D, Lin J, Cui P, et al (2019) A semi-supervised graph attentive network for financial fraud detection. In: 2019 IEEE international conference on data mining (ICDM), p 598–607

Wang C, Qiu Y, Gao D, et al (2022) Lifelong graph learning. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), p 13719–13728

Wang J, Song G, Wu Y, et al (2020) Streaming graph neural networks via continual learning. In: Proceedings of the 29th ACM international conference on information & knowledge management, p 1515–1524

Wang L, Wang Z, Wu L, et al (2024a) Bots shield fake news: adversarial attack on user engagement based fake news detection. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, p 2369–2378

Wang Z, Yang E, Shen L, et al (2024b) A comprehensive survey of forgetting in deep learning beyond continual learning. IEEE Transactions on Pattern Analysis and Machine Intelligence

Wu J, Xu W, Liu Q et al (2023) Adversarial contrastive learning for evidence-aware fake news detection with graph neural networks. IEEE Trans Knowl Data Eng 36(11):5591–5604

Wu J, Liu X, Cheng D, et al (2024) Safeguarding fraud detection from attacks: A robust graph learning approach. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, p 7500–7508

Wu B, Yao X, Zhang B, et al (2023a) Splitgnn: Spectral graph neural network for fraud detection against heterophily. In: Proceedings

of the 32nd ACM International Conference on Information and Knowledge Management, p 2737–2746
Xu K, Chen H, Liu S, et al (2019) Topology attack and defense for graph neural networks: An optimization perspective. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, p 3961–3967
Xu X, Ding K, Chen C, et al (2024b) Metagad: Meta representation adaptation for few-shot graph anomaly detection. In: In 2024 IEEE 11th International Conference on Data Science and Advanced Analytics (DSAA), p 1–10
Xu K, Hu W, Leskovec J, et al (2018a) How powerful are graph neural networks? arXiv preprint arXiv:1810.00826
Xu K, Li C, Tian Y, et al (2018b) Representation learning on graphs with jumping knowledge networks. In: Proceedings of the 35th International Conference on Machine Learning, pp 5453–5462
Xu F, Wang N, Wu H, et al (2024a) Revisiting graph-based fraud detection in sight of heterophily and spectrum. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence, p 9214–9222
Yang C, Liu H, Wang D, et al (2025a) Flag: Fraud detection with llm-enhanced graph neural network. In: Proceedings of the 31st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining vol 2, p 5150–5160
Yang X, Zhao X, Shen Z (2025b) A generalizable anomaly detection method in dynamic graphs. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence, p 22001–22009
Yuan Q, Guan SU, Ni P, et al (2023) Continual graph learning: A survey. arXiv preprint arXiv:2301.12230
Zhang K, Cao Q, Fang G, et al (2023) Dyted: Disentangled representation learning for discrete-time dynamic graph. In: Proceedings of the 29th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 3309–3320
Zhang A, Ma J (2024) Defensevgae: Defending against adversarial attacks on graph data via a variational graph autoencoder. In: Advanced Intelligent Computing Technology and Applications, p 313–324
Zhang X, Song D, Tao D (2024) Continual learning on graphs: Challenges, solutions, and opportunities. arXiv preprint arXiv:2402.11565
Zhang G, Wu J, Yang J, et al (2021a) Fraudre: Fraud detection dual-resistant to graph inconsistency and imbalance. In: 2021 IEEE international conference on data mining (ICDM), p 867–876
Zhang H, Wu B, Yang X, et al (2021b) Projective ranking: A transferable evasion attack method on graph neural networks. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, p 3617–3621
Zhang G, Yang Z, Wu J, et al (2022) Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. In: Proceedings of the 36th International Conference on Neural Information Processing System, p 24144–24157
Zhou F, Cao C (2021) Overcoming catastrophic forgetting in graph neural networks with experience replay. In: Proceedings of the 35th AAAI conference on artificial intelligence, p 4714–4722
Zhu J, Gao C, Yin Z, et al (2024) Propagation structure-aware graph transformer for robust and interpretable fake news detection. In: Proceedings of the 30th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p 4652–4663
Zhu Y, Lai Y, Zhao K, et al (2022) Binarizedattack: Structural poisoning attacks to graph-based anomaly detection. 2022 IEEE 38th International Conference on Data Engineering (ICDE), p 14–26
Zhuo W, Liu Z, Hooi B, et al (2024) Partitioning message passing for graph fraud detection. In: 12th International Conference on Learning Representations