Temporal Dynamics-Aware Adversarial Attacks on Discrete-Time Dynamic Graph Models

Kartik Sharma
Georgia Institute of Technology
Atlanta, GA, United States
ksartik@gatech.edu

Rakshit Trivedi
Massachusetts Institute of Technology
Boston, MA, United States
rstrivedi@csail.mit.edu

Rohit Sridhar
Georgia Institute of Technology
Atlanta, GA, United States
rohitsridhar@gatech.edu

Srijan Kumar
Georgia Institute of Technology
Atlanta, GA, United States
srijan@gatech.edu

ABSTRACT

Real-world graphs such as social networks, communication networks, and rating networks are constantly evolving over time. Many deep learning architectures have been developed to learn effective node representations using both graph structure and dynamics. While being crucial for practical applications, the robustness of these representation learners for dynamic graphs in the presence of adversarial attacks is highly understudied. In this work, we design a novel adversarial attack on discrete-time dynamic graph models where we desire to perturb the input graph sequence in a manner that preserves the temporal dynamics of the graph while dropping the performance of representation learners. To this end, we motivate a novel Temporal Dynamics-Aware Perturbation (TDAP) constraint, which ensures that perturbations introduced at each time step are restricted to only a small fraction of the number of changes in the graph since the previous time step. We present a theoretically-motivated Projected Gradient Descent approach for dynamic graphs to find effective perturbations under the TDAP constraint. Experiments on two tasks — dynamic link prediction and node classification, show that our approach is up to 4x more effective than the baseline methods for attacking these models. We extend our approach to a more practical online setting where graphs become available in real-time and show up to 5x superior performance over baselines We also show that our approach successfully evades state-of-the-art neural approaches for anomaly detection, thereby promoting the need to study robustness as a part of representation-learning approaches for dynamic graphs.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Supervised learning*; • **Information systems** → *Data mining*.

KEYWORDS

Graph Neural Networks, Dynamic Graphs, Adversarial Attacks

ACM Reference Format:
Kartik Sharma, Rakshit Trivedi, Rohit Sridhar, and Srijan Kumar. 2023. Temporal Dynamics-Aware Adversarial Attacks on Discrete-Time Dynamic Graph Models. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599517

1 INTRODUCTION

Graph Neural Networks (GNNs) have been shown to be vulnerable to adversarial perturbations [2, 12, 22, 30, 50, 60]. This has raised major concerns against their use in important industrial applications such as friend/product recommendation [43, 46, 54] and fraud detection [19, 56]. Recent advancements in designing attack and defense mechanisms to address these concerns have predominantly focused on GNN models for static graphs. In reality, the graph structure evolves with time as new interactions happen and new connections are formed [25, 28]. GNN models that incorporate the temporal information are shown to outperform their static counterparts in modeling dynamic networks on tasks such as predicting link existence in the future [8, 18, 23, 39, 44]. These models have been used for security-critical applications such as recommendation engines for e-commerce [27] and social networks [41], urban traffic monitoring [21], and modeling financial networks [16].

However, the vulnerability of these models to adversarial perturbations is less studied. The design of adversarial attacks for dynamic graphs to support such a study is challenging for two reasons — (1) Attacks must simultaneously optimize both the edge(s) to perturb and the time to perturb them, and (2) Attacks must preserve the original graph evolution after perturbation in order to evade detection. Attacks that disturb the graph evolution are not desired since they can be detected by various dynamic graph anomaly detection methods [1, 5, 6]. Thus, it is crucial to formulate effective adversarial attacks over time such that they do not significantly alter the original evolution of the graph structure.

In this work, we introduce a novel **T**emporal **D**ynamics-**A**ware **P**erturbation (TDAP) constraint to formulate evolution-preserving attacks on discrete-time dynamic graphs. This constraint asserts that the number of modifications added at the current timestep should only be a small fraction of the actual number of changes

Table 1: Comparison of our attack with existing works on graph adversarial attacks. Note that an attack is TDAP if the perturbations are aware of the local temporal dynamics.

Method	Dynamic	White-box	Evasion	Targeted	TDAP	Online
PGD [53]		✓	✓	✓		
IG-JSMA [53]		✓	✓	✓		
Fan et al. [14]	✓		✓			
Dyn-Backdoor [9]	✓	✓				
TGA [10]	✓	✓	✓	✓		
TD-PGD (proposed)	✓	✓	✓	✓	✓	✓

with respect to the preceding timestep. We show theoretically that perturbations made under TDAP constraint preserves the rate of change both in the structural and the embedding spaces. To find effective attacks under this proposed constraint, we consider a temporally-local, targeted, white-box, and evasion setting. As noted in Table 1, no prior works exist that can find attacks under our novel setting. Thus, we present a theoretically-grounded Temporal Dynamics-aware Projected Gradient Descent (TD-PGD) approach. The locality of the constraint in time allows us to easily extend this approach to find attacks in a more practical online setting [35] that has not been studied before for dynamic graphs. Here, perturbations are found in real-time without any knowledge of the future snapshots. Our contributions can be summarized as follows:

(1) We introduce a novel Temporal Dynamics-Aware Perturbation (TDAP) constraint to perturb discrete-time dynamic graphs while theoretically preserving the evolution of the graphs.
(2) We present a theoretically-grounded PGD-based white-box attack to find effective attacks on dynamic graphs under the novel TDAP constraint in both offline and online settings.
(3) We show that TD-PGD outperforms the baselines across 4 different datasets, 3 victim models, and 2 tasks in both offline and online settings. We release our code for future reference [1].
(4) We also demonstrate that TDAP-constrained perturbations are not detected by embedding-based anomaly detection methods.

2 RELATED WORK

Representation Learning for Dynamic Graphs. GNNs have been combined with sequential modeling architectures [23] to model dynamic graphs. For instance, discrete-time graphs have been modeled by using GNNs and RNNs together in a pipeline [34, 37] or an embedded manner [8, 39]. Attention-based models have also been proposed to jointly encode the graph structure and its dynamics [44]. For continuous-time graphs, both RNN [27, 31, 47, 48] and attention-based models [41, 52] have been proposed to update embeddings in real-time, upon the occurrence of a new event.

Adversarial Attacks on Graphs. Static GNNs are known to be vulnerable to adversarial attacks in different settings [22]. White-box attacks are studied assuming complete knowledge of the underlying model [50, 53]. Limiting the model knowledge, vulnerability of static GNNs against gray-box [60] and black-box attacks [12] have also been extensively studied. While most attacks are studied under a budget constraint, some works explore other strategies such as edge-rewiring [32, 33], low-degree attack [30], and preservation of homophily [11] and degree statistics [60].

[1]https://github.com/claws-lab/TDAP

In comparison, the literature on adversarial attacks for dynamic graphs is scarce. Time-aware Gradient Attack (TGA) [10] is a white-box evasion attack that greedily selects the perturbations across time under a budget constraint. In addition, attacks to poison training data [9] and black-box attacks using RL approaches [14] have also been proposed.

Anomaly Detection on Dynamic Graphs. Both supervised [6, 49, 57, 58] and unsupervised [18, 45, 55] anomaly detection approaches have been studied in the literature for dynamic graphs [1, 33, 40]. We focus on the more practical unsupervised methods in this work. In the case of dynamic graphs, perturbations must preserve the temporal flow to be imperceptible. Traditional anomaly detection algorithms flag an instance to be anomalous if the distance between consecutive snapshots crosses a threshold [1]. In particular, Graph Edit Distance and Hamming distance between adjacency matrices have been used to monitor communication networks [5, 45]. More recently, neural approaches have looked at the consecutive change in the embedding space to detect anomalies without feature extraction [6, 18]. NetWalk finds anomalies by clustering the embeddings together [55].

Table 1 compares our contributions against prior work.

3 METHODOLOGY

Problem. Let $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_T$ be the original graph snapshots and $\mathcal{G}'_1, \mathcal{G}'_2 \cdots, \mathcal{G}'_T$ be the corresponding perturbed snapshots. Note that $\mathcal{G}_i = (\mathbf{X}_i, \mathbf{A}_i)$ where $\mathbf{X}_i, \mathbf{A}_i$ are the node features and the adjacency matrix for snapshot i, respectively. Also, let $\mathcal{M}$ be a victim dynamic graph model that we want to attack and let $f_{\mathcal{M}}$ be a function that generates the corresponding node embeddings of $\mathcal{G}_t$ given $\mathcal{G}_{1:t-1}$. Let y_{task} be the actual labels for a given task (for dynamic link prediction, these correspond to binary labels representing link existence in the future snapshot).

Then, the objective of the attacker is to introduce structural perturbations $\mathbf{S}_t = \mathbf{A}'_t - \mathbf{A}_t$ at each timestep $t < T$ such that the model inference at timestep T for the target entities E_{tg} deteriorates. More formally, the attacker solves the following optimization problem:

$$\max_{\mathbf{A}'_1, \mathbf{A}'_2, \cdots, \mathbf{A}'_{T-1}} \mathcal{L}_{\text{task}}\left(\widehat{y}_{\text{task}}(f_{\mathcal{M}}(\mathbf{A}'_{1:T-1})), y_{\text{task}}, E_{tg}\right) \quad \text{such that} \quad C(\mathbf{A}'_{1:T-1}) \text{ holds} \tag{1}$$

for some constraint function C on the perturbed adjacency matrices $\mathbf{A}'_t$ for each time t. Here, $\widehat{y}_{\text{task}}$ denotes the predicted labels for the given task and $\mathcal{L}_{\text{task}}$ is a task-specific loss, for example, a binary cross entropy (CE) loss for link prediction.

The constraint function C is designed to ensure imperceptibility of the adversarial perturbations. In the literature, a budget constraint has been widely used to enforce imperceptibility in graphs [12] and computer vision [17]. However, this constraint only bounds the total amount of perturbations that can be introduced by an attacker. When the input is dynamic, as in the case of dynamic graphs, the perturbations should be constrained in the context of how the input evolves. However, since the budget constraint completely ignores the graph dynamics, it could lead to a drastic change in the evolution trend of the graph and thus, making the attacks easily detectable. For instance, with the budget constraint, all the perturbations can be made at a single time step, leading to an anomalous spike, which would be easily detected as a possible attack by graph

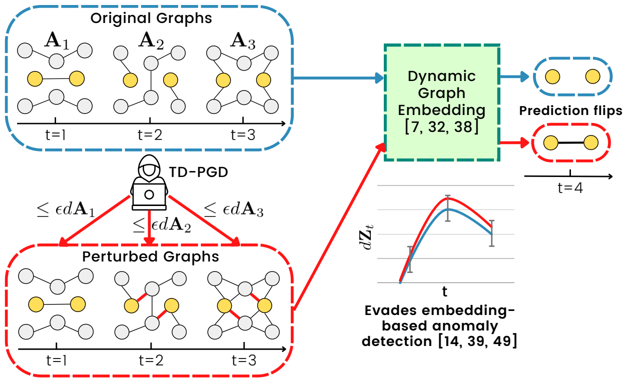

Original Graphs
$\mathbf{A}_1$
$\mathbf{A}_2$
$\mathbf{A}_3$
t=1
t=2
t=3
Dynamic Graph Embedding [7, 32, 38]
Prediction flips
t=4
TD-PGD
$\le \epsilon d\mathbf{A}_1$
$\le \epsilon d\mathbf{A}_2$
$\le \epsilon d\mathbf{A}_3$
Perturbed Graphs
$d\mathbf{Z}_t$
t
Evades embedding-based anomaly detection [14, 39, 49]

Figure 1: Overview of the Temporal Dynamics-aware Perturbation attack. The attacker is able to flip the prediction on the target while (theoretically) evading detection.

anomaly detection methods for dynamic graphs [1, 5, 45]. Thus, a constraint is desired that can ensure that the introduced perturbations do not disrupt the evolving trend of the dynamic graphs.

3.1 Temporal Dynamics-Aware Perturbation (TDAP) Constraint

The simplest measure to study evolution is to consider the change in the input between consecutive time steps. Thus, for a discrete-time input $\{\mathbf{x}_t\}$, this corresponds to the discrete-time differential norm at time t, given by $d\mathbf{x}_t = \|\mathbf{x}_t - \mathbf{x}_{t-1}\|$. Then, we define

Definition 1 (TDAP). *The number of perturbations introduced to input* $\mathbf{x}$ *at time step* t *must not be more than a fraction* ϵ *times the differential at* t, *i.e.* ***TDAP***$(\epsilon) := \|\mathbf{x}'_t - \mathbf{x}_t\| \le \epsilon d\mathbf{x}_t\ \forall t$.

For the case of dynamic graphs when the graph structure evolves (for example, in social networks and transaction networks [39, 44]), this constraint becomes $\|\mathbf{A}'_t - \mathbf{A}_t\|_1 \le \epsilon d\mathbf{A}_t$. Alternatively, a dynamic graph may also involve a temporally-evolving signal at each node [29, 38, 42], in which case, this constraint is applied to the node features as $\|\mathbf{X}'_t - \mathbf{X}_t\|_1 \le \epsilon d\mathbf{X}_t$.

In this work, we focus on dynamic graph structures such that the constraint C (in Equation 1) for the perturbations is a TDAP constraint. The optimization problem for the attacker, thus, becomes

$$\max_{\mathbf{A}'_1, \mathbf{A}'_2, \cdots, \mathbf{A}'_{T-1}} \mathcal{L}_{\text{task}}\left(\widehat{y}_{\text{task}}(f_{\mathcal{M}}(\mathbf{A}'_{1:T-1})), y_{\text{task}}, E_{tg}\right) \tag{2}$$

$$\text{such that } \forall t \in (1,T): \frac{\|\mathbf{A}'_t - \mathbf{A}_t\|}{\|\mathbf{A}_t - \mathbf{A}_{t-1}\|} \le \epsilon$$
$$\|\mathbf{A}'_1 - \mathbf{A}_1\| \le \epsilon_1,$$

where ϵ, ϵ_1 are given parameters for this optimization. We use $\|\cdot\|$ to denote the 1-norm of the matrix flattened into a vector $\mathbf{a}$, unless otherwise mentioned. This means $\|\mathbf{A}'_t - \mathbf{A}_t\| = \|\mathbf{a}'_t - \mathbf{a}_t\|_1$, where $\mathbf{a}, \mathbf{a}'$ are flattened vectors of $\mathbf{A}, \mathbf{A}'$ respectively. Also, let us define the perturbation matrix $\mathbf{S}_t := \mathbf{A}'_t - \mathbf{A}_t$ and $\varepsilon_t := \epsilon\|\mathbf{A}'_t - \mathbf{A}_t\|$. Then, TDAP constraint can be written equivalently as $\|\mathbf{S}_t\| := \|\mathbf{A}'_t - \mathbf{A}_t\| = \|\mathbf{a}'_t - \mathbf{a}_t\|_1 \le \varepsilon_t := \epsilon d\mathbf{A}_t = \epsilon\|\mathbf{a}_t - \mathbf{a}_{t-1}\|_1 = \epsilon\|\mathbf{A}_t - \mathbf{A}_{t-1}\|$.

Implications. We show that TDAP constraint has the following implications on the perturbations.

(1) **Perturbations under TDAP constraint preserves the average rate of structural change.**

Theorem 1. *Let* $\overline{d\mathbf{A}_t} = \frac{1}{t}\sum_{\tau \le t} d\mathbf{A}_\tau, \overline{d\mathbf{A}'_t} = \frac{1}{t}\sum_{\tau \le t} d\mathbf{A}'_\tau$. *Then,*

$$|1 - 2\epsilon|\overline{d\mathbf{A}_t} \le \overline{d\mathbf{A}'_t} \le 2\epsilon\overline{d\mathbf{A}_t} + \beta_t, \tag{3}$$

for some constant $\beta_t \in \mathbb{R}_{\ge 0}$.

Proof. (1) $|\mathbf{1} - \mathbf{2}\epsilon|\overline{\mathbf{dA_t}} \le \overline{\mathbf{dA'_t}}$
Note that $\|\cdot\|_1 \ge \|\cdot\|_2$, thus TDAP also gives us $\|\mathbf{a}'_\tau - \mathbf{a}_\tau\|_2 \le \varepsilon_\tau$. This implies $\mathbf{a}_\tau - \varepsilon_\tau\mathbf{e} \le \mathbf{a}'_\tau \le \mathbf{a}_\tau + \varepsilon_\tau\mathbf{e}$ for all unit vectors $\mathbf{e}$. Substituting the above inequalities in $\|\mathbf{A}'_\tau - \mathbf{A}'_{\tau-1}\| = \|\mathbf{a}'_\tau - \mathbf{a}'_{\tau-1}\|$, we get $\|\mathbf{A}'_\tau - \mathbf{A}'_{\tau-1}\| \ge \|(\mathbf{a}_\tau - \varepsilon_\tau\mathbf{e}_1) - (\mathbf{a}_{\tau-1} + \varepsilon_{\tau-1}\mathbf{e}_2)\|_1$, for some unit vectors $\mathbf{e}_1, \mathbf{e}_2$. Using reverse triangle inequality, $\|\mathbf{A}'_\tau - \mathbf{A}'_{\tau-1}\| \ge |\|\mathbf{a}_\tau - \mathbf{a}_{\tau-1}\|_1 - \|\varepsilon_\tau\mathbf{e}_1 + \varepsilon_{\tau-1}\mathbf{e}_2\|_1| \ge |\|\mathbf{a}_\tau - \mathbf{a}_{\tau-1}\|_1 - (\varepsilon_\tau\|\mathbf{e}_1\|_1 + \varepsilon_{\tau-1}\|\mathbf{e}_2\|_1)|$, due to triangle inequality. Summing both sides over all time steps until t, we get $\sum_{\tau \le t}\|\mathbf{A}'_\tau - \mathbf{A}'_{\tau-1}\| \ge \sum_\tau |\|\mathbf{a}_\tau - \mathbf{a}_{\tau-1}\|_1 - \varepsilon_\tau - \varepsilon_{\tau-1}| \ge |\sum_\tau \|\mathbf{A}_\tau - \mathbf{A}_{\tau-1}\|_1 - \epsilon(\|\mathbf{A}_\tau - \mathbf{A}_{\tau-1}\|_1 + \|\mathbf{A}_{\tau-1} - \mathbf{A}_{\tau-2}\|_1)|$. Replacing $\sum_\tau\|\mathbf{A}_\tau - \mathbf{A}_{\tau-1}\|$ as $\overline{d\mathbf{A}_t}$, we get the desired result.

(2) $\overline{\mathbf{dA'_t}} \le 2\epsilon\overline{\mathbf{dA_t}} + \beta$
By definition of $\mathbf{S}_\tau$, $d\mathbf{A}'_\tau = \|\mathbf{A}'_\tau - \mathbf{A}'_{\tau-1}\| = \|(\mathbf{A}_\tau + \mathbf{S}_\tau) - (\mathbf{A}_{\tau-1} + \mathbf{S}_{\tau-1})\|$. Then, using triangle inequality, we get $d\mathbf{A}'_\tau \le \|\mathbf{A}_\tau + \mathbf{S}_\tau\| + \|\mathbf{A}_{\tau-1} + \mathbf{S}_{\tau-1}\| \le \|\mathbf{A}_\tau\| + \|\mathbf{S}_\tau\| + \|\mathbf{A}_{\tau-1}\| + \|\mathbf{S}_{\tau-1}\|$. Now, since $\|\mathbf{S}_\tau\| \le \epsilon d\mathbf{A}_\tau$, $d\mathbf{A}'_\tau \le \epsilon d\mathbf{A}_\tau + \epsilon d\mathbf{A}_{\tau-1} + \|\mathbf{A}_\tau\| + \|\mathbf{A}_{\tau-1}\|$. Then, we get $\overline{d\mathbf{A}'_t} = \frac{1}{t}\sum_\tau d\mathbf{A}'_\tau \le \frac{1}{t}\sum_\tau(\epsilon d\mathbf{A}_\tau + \epsilon d\mathbf{A}_{\tau-1} + \|\mathbf{A}_\tau\| + \|\mathbf{A}_{\tau-1}\|) \le 2\epsilon\frac{1}{t}\sum_\tau d\mathbf{A}_\tau + \frac{2}{t}\sum_\tau\|\mathbf{A}_\tau\|$, which gives us the desired result for $\beta_t = \frac{2}{t}\sum_\tau\|\mathbf{A}_\tau\|$. □

This means that the average structural evolution after TDAP perturbations remains within a factor of $\max\{2\epsilon, |1 - 2\epsilon|\}$ of its original value. Since ϵ is a parameter controlled by the attacker, he may tune this value to obtain arbitrarily strong bounds on the average evolution. Also, note that the additive factor is just two times the average number of edges in the past (or equivalently the average degree in the past snapshots, for a targeted case). In comparison, a budget constraint only bounds $\sum_t\|\mathbf{A}'_t - \mathbf{A}_t\|$ which gives no such guarantees on $\overline{d\mathbf{A}'_t}$ with respect to $\overline{d\mathbf{A}_t}$.

(2) **Perturbations under TDAP constraint preserves the rate of embedding change.**

Corollary 1. *Let* $d\mathbf{Z}_t = \|\mathbf{Z}_t - \mathbf{Z}_{t-1}\|_1$. *Then,*

$$|1 - 2\epsilon|\chi_t d\mathbf{Z}_t \le d\mathbf{Z}'_t \le 2\epsilon\gamma_t d\mathbf{Z}_t + \beta_t\gamma_t, \tag{4}$$

for β_t *from Theorem 1 and some constants* $\chi_t, \gamma_t \in \mathbb{R}_{\ge 0}$.

Proof. We prove these results by showing $C_{1,t}\overline{d\mathbf{A}_t} \le d\mathbf{Z}_t \le C_{2,t}\overline{d\mathbf{A}_t}$ and $C'_{1,t}\overline{d\mathbf{A}'_t} \le d\mathbf{Z}'_t \le C'_{2,t}\overline{d\mathbf{A}'_t}$. Then, the result follows by applying Theorem 1 such that $\chi_t = C'_{1,t}/C_{2,t}$, $\gamma_t = C'_{2,t}/C_{1,t}$.
Note that $\mathbf{Z}_t = \mathbf{f}_t(\mathbf{a}_t, \mathbf{a}_{t-1}, \cdots, \mathbf{a}_1)$, where $\mathbf{f}_t : \mathbb{R}^{n^2t} \to \mathbb{R}^{nd}$ such that n is the number of nodes. Instead, we consider equivalently the vector function $\mathbf{Z}_t = \mathbf{f}(\mathbf{a}_t, \cdots, \mathbf{a}_1, \mathbf{0}, \mathbf{0}, \cdots, \mathbf{0})$, where we append $(T - t)$ zeros such that $\mathbf{f} : \mathbb{R}^{n^2T} \to \mathbb{R}^{nd}$. Let us consider the concatenated vector $\mathbf{a}_{\le t} = (\mathbf{a}_t, \mathbf{a}_{t-1}, \cdots, \mathbf{a}_1, \mathbf{0}, \cdots, \mathbf{0})$. By Cauchy's Mean Value Theorem in several variables, we have $\nabla^-\mathbf{f} \cdot (\mathbf{a}_{\le t} - \mathbf{a}_{\le t-1}) \le \mathbf{Z}_t - \mathbf{Z}_{t-1} \le \nabla\mathbf{f} \cdot (\mathbf{a}_{\le t} - \mathbf{a}_{\le t-1})$, where ∇^- is the left-hand derivative. This gives us $\|\mathbf{Z}_t - \mathbf{Z}_{t-1}\| \le \|\nabla\mathbf{f}\|\ \|\mathbf{a}_{\le t} - \mathbf{a}_{\le t-1}\|$ by Cauchy-Schwarz inequality and $\|\mathbf{Z}_t -$

$\mathbf{Z}_{t-1}\| \geq \|\nabla^{-}\mathbf{f}\| \, \|\mathbf{a}_{\leq t} - \mathbf{a}_{\leq t-1}\| \cos(\theta)$, by the definition of dot product when θ is the angle in between.

Next, we note that $\|\mathbf{a}_{\leq t} - \mathbf{a}_{\leq t-1}\|_1 = \|(\mathbf{a}_t - \mathbf{a}_{t-1}, \cdots, \mathbf{a}_2 - \mathbf{a}_1, \mathbf{a}_1)\|_1 = \sum_t \|\mathbf{a}_t - \mathbf{a}_{t-1}\|_1 = T\overline{d\mathrm{A}_t}$. Substituting this in mean theorem results, we get $C_{1,t}\overline{d\mathrm{A}_t} \leq d\mathbf{Z}_t \leq C_{2,t}\overline{d\mathrm{A}_t}$ for some constants $C_{1,t}, C_{2,t}$. Similarly, we get for $d\mathbf{Z}'$. □

This means that the embedding change after perturbation remains within a factor that depends on the gradient of the embedding function and the constants from Theorem 1. This gives control to the attacker through ϵ to attack in a manner that distorts the embedding evolution within a permissible value.

(3) **Perturbations under TDAP constraint change the embeddings by a factor of the average rate of structural change.**

Proposition 1. $\|\mathbf{Z}'_t - \mathbf{Z}_t\| \leq \|\nabla \mathbf{f}\| t\epsilon\overline{d\mathrm{A}_t}$.

Proof. By Cauchy's MVT on $\mathbf{f}$ with inputs $\mathbf{a}'_{\leq t}$ and $\mathbf{a}_{\leq t}$, we get $\mathbf{Z}'_t - \mathbf{Z}_t \leq \nabla \mathbf{f} \cdot (\mathbf{a}'_{\leq t} - \mathbf{a}_{\leq t})$, which gives us $\|\mathbf{Z}'_t - \mathbf{Z}_t\| \leq \|\nabla f\| \, \|\mathbf{a}'_{\leq t} - \mathbf{a}_{\leq t}\|$ by Cauchy-Schwarz inequality. Note that $\|\mathbf{a}'_{\leq t} - \mathbf{a}_{\leq t}\|_1 = \sum_\tau \|\mathbf{a}'_\tau - \mathbf{a}_\tau\|_1$. Using TDAP constraint, we then get $\|\mathbf{a}'_{\leq t} - \mathbf{a}_{\leq t}\|_1 \leq \sum_\tau \epsilon \|\mathbf{a}_\tau - \mathbf{a}_{\tau-1}\| \leq t\epsilon\overline{d\mathrm{A}_t}$. □

This means that due to TDAP, embeddings are perturbed by only a factor of the average rate of structural change originally. The attacker can further control this multiplicative factor via ϵ such that the embeddings after perturbation remain close enough to their original values.

3.2 Attack Methods Under TDAP Constraint

While the TDAP constraint allows us to limit the effect of the perturbations on the graph's evolution, it is not clear how one can efficiently find perturbations that maximize a loss function under this constraint. To this end, we present two algorithms to solve the optimization problem of Equation 2.

3.2.1 *Greedy Time-Aware Gradient (TGA(ϵ)).*

Inspired by the closest work [10], a greedy strategy can be adopted to find effective perturbations under our TDAP constraint. We extend this existing work to our novel constraint for a fair comparison. Here, perturbations are simply selected in a greedy manner based on their gradient with respect to the downstream loss, while satisfying the constraint. Following TGA, we further reduce the time complexity of this greedy search by dividing the search space into two steps. First, we find the top-gradient perturbation at each time step and then, select the one that reduces the prediction probability the most. In particular, we greedily select the perturbations with the lowest probability such that TDAP(ϵ) is not violated for any time step.

Complexity. We note here that TGA(ϵ) makes $O(\epsilon \sum_t d\mathrm{A}_t)$ backward calls to the victim model for gradient calculation. Let each backward call take T_{bw} time for a model $\mathcal{M}$. Then, the total time complexity for this algorithm is given by $O(\epsilon \sum_t d\mathrm{A}_t T_{bw})$. An efficient implementation of greedy can greedily pick the top-gradient perturbations without storing gradients. This stores just the selected perturbations and gives a space complexity of $O(\epsilon \sum_t d\mathrm{A}_t)$.

3.2.2 *Temporal Dynamics-aware Projected Gradient Descent.*

Since the constrained optimization in Equation 2 has a general continuous objective, a greedy approach is only sub-optimal (even for a simpler convex objective) with no theoretical guarantees. A more standard approach to do optimization under a convex constraint is to use projected gradient descent (PGD) [3, 4]. Since our problem is in discrete-space, we first relax it into continuous space, find the solution using PGD and then, randomly round it to obtain a valid solution for the discrete problem. In particular, we relax the perturbation matrix S_t into a continuous vector $\mathbf{s}_t$ and show that a closed-form projection operator exists for the TDAP(ϵ) constraint. Algorithm 1 demonstrates the steps involved in this approach (TD-PGD), following the result of Proposition 2.

Algorithm 1 Temporal Dynamics-aware Projected Gradient Descent (TD-PGD)

```
Require: TDAP variables $\varepsilon_t$ (from Thm. 2), Initial vector $\mathbf{s}^{(0)}$, Loss
    function $\mathcal{L}$, Actual labels $y$, Target entities $E_{tg}$, Time steps $T$,
    Learning rate $\eta_i$, Iterations $N$, Rounding iterations $N_r$
Ensure: Perturbation vector $\mathbf{s}_t^{(i)}$ preserves TDAP($\epsilon$) for all $i, t$
1: for $i = 1$ to $N$ do
2:     Perturb: $\mathcal{G}'_t \leftarrow \mathcal{G}_t \oplus \mathbf{s}_t^{(i-1)}$ for all $t$.
3:     Gradient Descent: $\mathbf{s}^{(i)} \leftarrow \mathbf{s}^{(i-1)} + \eta_i \nabla_{\mathbf{s}} \mathcal{L}(\mathcal{G}', y, E_{tg})$.
4:     Project: For all $t$: $\mathbf{s}_t^{(i)} \leftarrow \Pi_C(\mathbf{s}_t^{(i)})$ from Equation 5.
5: $\mathrm{S}_t \leftarrow$ ROUND $(\mathbf{s}_t^{(N)}, N_r, \{\varepsilon_t\})$
```

Proposition 2. *Suppose C denotes the feasible perturbation space for the constraints $\|\mathrm{A}'_t - \mathrm{A}_t\| / \|\mathrm{A}_t - \mathrm{A}_{t-1}\| \leq \epsilon$ for all $1 < t < T$ and $\|\mathrm{A}'_1 - \mathrm{A}_1\| \leq \varepsilon_1$. Then, one can project the perturbation vector $\mathbf{s}_t$ onto C as $\mathbf{s}'_t = \Pi_C(\mathbf{s}_t) = \arg\min_{\mathbf{s}'_t \in C} \frac{1}{2}\|\mathbf{s}'_t - \mathbf{s}_t\|_2^2$:*

$$\Pi_C(\mathbf{s}_t) = \begin{cases} P_{[0,1]}(\mathbf{s}_t - \mu_t \mathbf{1}) & \text{if } \exists \mu_t > 0 : \mathbf{1}^T P_{[0,1]}(\mathbf{s}_t - \mu_t \mathbf{1}) = \varepsilon_t \\ P_{[0,1]}(\mathbf{s}_t) & \text{if } \mathbf{1}^T P_{[0,1]}(\mathbf{s}_t) \leq \varepsilon_t \end{cases} \tag{5}$$

where $\varepsilon_t = \epsilon d\mathrm{A}_t = \epsilon\|\mathrm{A}_t - \mathrm{A}_{t-1}\|$ for $t > 1$, and $P_{[0,1]}(x) = x$ if $x \in [0,1]$, 0 if $x < 0$ and 1 if $x > 1$.

Proof. The constraints are $\mathbf{1}^T \mathbf{s}'_t \leq \varepsilon_t, \mathbf{s}'_t \in [\mathbf{0}, \mathbf{1}]$. Lagrangian of the projection optimization problem can then be written as $L(\mathbf{s}'_t, \mathbf{s}_t; \mu_t, \boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1) = \frac{1}{2}\|\mathbf{s}'_t - \mathbf{s}_t\|_2^2 + \mu_t(\mathbf{1}^T \mathbf{s}'_t - \varepsilon_t) + \boldsymbol{\lambda}_0 \cdot (-\mathbf{s}'_t) + \boldsymbol{\lambda}_1 \cdot (\mathbf{s}'_t - 1)$. KKT conditions imply that $\mathbf{s}'_t - \mathbf{s}_t + \mu_t - \boldsymbol{\lambda}_0 + \boldsymbol{\lambda}_1 = \mathbf{0}$, such that $\mu_t, \boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1 \geq \mathbf{0}$ and $\mu_t(\mathbf{1}^T \mathbf{s}'_t - \varepsilon_t) = 0, \boldsymbol{\lambda}_0 \odot (-\mathbf{s}'_t) = \mathbf{0}, \boldsymbol{\lambda}_1 \odot (\mathbf{s}'_t - 1) = \mathbf{0}$, while satisfying $\mathbf{1}^T \mathbf{s}'_t \leq \varepsilon_t, \mathbf{s}_t \in [\mathbf{0}, \mathbf{1}]$. Thus, $\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1$ can be replaced by an elementwise clamping operation within $[0,1]$, *i.e.*, $P_{[0,1]}(\cdot)$ since they are zero when they are within the range and otherwise equal to a value such that $s_t = 0$ or 1. Thus, $\mathbf{s}'_t = P_{[0,1]}(\mathbf{s}_t - \mu_t \mathbf{1})$ such that $\mu_t = 0$ if $\mathbf{1}^T P_{[0,1]}(\mathbf{s}_t) \leq \varepsilon_t$ otherwise we find $\mu_t \geq 0$ such that $\mathbf{1}^T P_{[0,1]}(\mathbf{s}_t - \mu_t \mathbf{1}) - \varepsilon_t = 0$. □

Following [53], we use the bisection method [3] to solve the equation $\mathbf{1}^T P_{[0,1]}(\mathbf{s}_t - \mu_t \mathbf{1}) = \varepsilon_t$ in μ_t for $\mu_t \in [\min(\mathbf{s}_t - 1), \max(\mathbf{s}_t)]$. This converges in the logarithmic rate, *i.e.* it takes $O(\log_2[(\max(\mathbf{s}_t) - \min(\mathbf{s}_t - 1))/\xi])$ time for ξ-error tolerance.

Randomized Rounding for TD-PGD (ROUND). Inspired by existing works on using PGD for graphs [15, 53], we use randomized rounding in order to efficiently obtain a valid discrete perturbation solution S_t for our constraint from the continuous vector $\mathbf{s}_t^{(N)}$. In particular, for a fixed number of iterations N_r, we randomly sample from the Bernoulli distribution formed by $\mathbf{s}_t^{(N)} \in [0,1]^{n^2}$. Then, we pick the sample that maximizes the loss while satisfying the constraint. Furthermore, to ensure that we obtain at least one solution, we adopt the top-k heuristic sampling strategy in the first iteration [15] for our constraint, such that we select only the top ϵ_t values in $\mathbf{s}_t^{(N)}$ for all t. Thus, $\mathbf{1}^T \mathrm{S}_t = \epsilon_t$ and the obtained discrete solution will satisfy the TDAP constraint at each t.

Complexity. It makes $O(N)$ backward calls to the victim model for gradient calculation, where N is the number of iterations for the TD-PGD loop. In addition, the projection step takes $O(\sum_t \log_2[(\max(\mathbf{s}_t) - \min(\mathbf{s}_t - 1))/\xi])$ time per iteration. Since $\mathbf{s}_t \in [0,1]$ for each element and there are $|\mathcal{V}|$ elements in $\mathbf{s}_t$ ($|\mathcal{V}|$ is the total number of nodes), the projection takes $O(T \log |\mathcal{V}|)$ per iteration. The total time complexity of the TD-PGD loop then becomes $O(NT_{bw} + NT \log |\mathcal{V}|)$. Since TD-PGD stores the whole perturbation vector through its loop, the space complexity becomes $O(|\mathcal{V}|)$.

3.3 Online Adversarial Attacks

We also consider the online version of the problem in Equation 2. In this setting, the perturbations are added in real-time, *i.e.* they are both **immediate** and **irrevocable**. More formally, Equation 2 must now be solved considering online updates of the optimization variables, *i.e.*, (1) $\mathbf{A}'_t$ is updated at time step t without any knowledge of $\mathbf{A}_{t+1:T}$ and (2) $\mathbf{A}'_t$ remains unchanged for future time steps. Note that TDAP constraint must still hold for $\mathbf{A}'_t$ at all time steps t.

Online TD-PGD. Inspired from its theoretical guarantees in online convex optimization [59], we use Online Projected Gradient Descent for our problem. In this framework, we are given a function f_t for each step t and the goal is to choose x_t in an online manner such that the regret on the offline optimum x_t^*, $\mathcal{R}(f,x) := \sum_t (f_t(x_t) - f_t(x_t^*))$ is minimized. In our problem, as defined in Equation 2, we need to minimize a loss $h(\mathbf{A}_{1:T-1})$ at the final time step T. To use online gradient descent, we thus need to write h as $\sum_t f_t(\mathbf{A}_t)$ for some f_t. Let us assume that h is a cross-entropy loss and that the embeddings at each time t are encoded in a sequential manner. Then,

$$h(\{\mathbf{A}_t\}_{t=1}^{T-1}) = -\sum_{d\in\mathcal{D}} y(d) \log p(d, \{\mathbf{A}_t\}) \tag{6}$$
$$= \sum_t -\sum_{d\in\mathcal{D}} y(d) \log p(d, \mathbf{A}_t | \mathbf{A}_1, \cdots, \mathbf{A}_{t-1}).$$

Thus, we define $f_t(\mathbf{A}_t) = -\sum_{d\in\mathcal{D}} y(d) \log p(d, \mathbf{A}_t|\mathbf{A}_1, \cdots, \mathbf{A}_{t-1})$, which is the prediction loss for the data points $\mathcal{D}$ at time step t. Algorithm 1 can then be updated to find attacks in real-time, following Online Gradient Descent. In particular, for time t, we only perturb graphs for $\tau \in [1,t]$ and do the gradient descent on the loss (in line 3) $\mathcal{L}(\{\mathcal{G}_\tau \oplus \mathbf{s}_\tau^{(i-1)}\}_{\tau=1}^t, y(t), \cdot)$. Since the projection operator for TDAP (Equation 5) depends only on the current time step t, we can independently project for the current time, *i.e.* line 4 remains $\mathbf{s}_\tau^{(i)} = \Pi_C(\mathbf{s}_\tau^{(i)})$ but only for $\tau \in [1,t]$.

4 EXPERIMENTAL SETUP

Datasets. We use these 3 datasets for dynamic link prediction – Radoslaw[2], UCI [2], and Reddit[3]. Radoslaw and UCI are email communication networks, where two nodes (users) are connected if they have an email communication at time t. Reddit is a hyperlink network representing directed connections between subreddits if there is a hyperlink from one to the other at a given timestamp [26]. For node classification task, we use one publicly-available dataset, DBLP-5 [51]. This is a co-author network with node attributes as word2vec representations of the author's papers. There are 5 node labels representing the different fields that the authors belong to.

[2]http://konect.cc/networks/
[3]https://snap.stanford.edu/data/soc-RedditHyperlinks.html

Table 2: Description of the datasets and performance of different models on them. For DBLP-5, we show accuracy of node classification, while for the rest, we show ROC-AUC for the dynamic link prediction. T denotes the number of timesteps and Split is the time-interval (weekly (w)) for each snapshot. We use the same split as [51] for DBLP-5.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	T	Split	# Labels	DySAT	EvolveGCN	GC-LSTM
Radoslaw	167	22K	13	3w	-	0.74	0.74	0.81
UCI	1.9K	24K	13	2w	-	0.95	0.87	0.97
Reddit	35K	715K	20	4w	-	0.95	0.94	0.94
DBLP-5	6.6K	43K	10	[51]	5	0.69	0.68	0.69

Table 2 shows the statistics of these datasets. We split each dataset into a finite number of snapshots. Radoslaw is split using a 3-week period in 13 snapshots while the 13 snapshots in UCI denote a 2-week period. The Reddit dataset spans over 3 years; thus, we use a 2-month split to obtain the 20 snapshots. For DBLP-5, we use the publicly available pre-processed data [51].

For datasets with no node features, *i.e.*, Radoslaw, UCI, and Reddit, we use uniformly random features with dimension 10. The pre-processed DBLP-5 has 100 features for each node.

Attack Methods. We consider 4 different attack methods to find perturbations under TDAP constraint. **(1) TD-PGD** is a projected gradient descent with a valid projection operator for the TDAP constraint, as specified in Algorithm 1. **(2) TGA**(ϵ) greedily selects the perturbation with the highest gradient value of the loss (we adapt TGA [10] to our setting, as specified in Section 3.2). **(3) Degree** flips the edges (adds or deletes if already there) attached with the highest degree nodes in the graph at each time step, while making at most $\epsilon d\mathbf{A}_t$ perturbations. **(4) Random** randomly flips (add or delete) at most $\epsilon d\mathbf{A}_t$ edges at each time step t.

Victim Models. We test the performance of the above attack methods on 3 different discrete-time dynamic graph models. **(1) GC-LSTM** [8] embeds GCN into an LSTM to encode the sequence of graphs. **(2) EvolveGCN** [39] uses a recurrent model (RNN-LSTM) to evolve the weights of a GCN. We use the EvolveGCN-O version for our experiments. **(3) DySAT** [44] utilizes joint structural and temporal self-attention to embed. For dynamic link prediction, we trained using a Binary Cross Entropy loss on the edges in all but the last snapshot, which was used as the test set. For node classification, we use a 20% held-out set of nodes in the final snapshot as the test set and minimize the Weighted Cross Entropy loss on the node labels for the rest. Table 2 shows the performance of these models on different datasets.

Metrics. We use the relative drop, as defined below, to evaluate the efficacy of the attack methods.

$$\text{Rel. Drop (\%)} = \frac{\text{Perturbed perf.} - \text{Original perf.}}{\text{Original perf.}} \times 100, \tag{7}$$

where performance (perf.) is evaluated using ROC-AUC for dynamic link prediction and using Accuracy for node classification.

Setup. We consider a targeted setting, which means that perturbation space is limited to adding/deleting an edge directly to the target nodes (both end nodes in case of a target link). Each target is attacked one by one and the total performance is measured using the relevant performance metric over these targets.

Implementation. For TD-PGD optimization, we used ADAM optimizer [24] with the initial learning rate of 10 and ran for 50 iterations. The initial perturbation vector $\mathbf{s}^{(0)}$ was fixed as all ones, giving each perturbation an equal chance at the start. Also, we stop the greedy search in the TGA(ϵ) baseline if the time taken exceeds 300 s per target, which is at least 3 times that of TD-PGD.

We use TorchGeometric-Temporal [4] to train the victim models EvolveGCN and GC-LSTM, while we use the pytorch implementation [5] for DySAT. We adopt the official code of TGA[6] to implement the TGA(ϵ) baseline. All the experiments were conducted on Python 3.8.12 on a Ubuntu 18.04 PC with an Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz processor, 512 GB RAM, and Tesla V100-SXM2 32 GB GPUs.

5 RESULTS

We compare different attack methods against the 3 victim models on dynamic link prediction and node classification tasks in offline and online settings. We also test the detectability of TDAP-constrained perturbations against anomaly detectors. For all the experiments, we vary ϵ from 0 to 1 and fix $\varepsilon_1 = \min_{t>1} \varepsilon_t := \epsilon d\mathbf{A}_t$.

5.1 Relative Performance Drop

5.1.1 ***Dynamic Link Prediction.***
In this section, we show the attack performance on the task of dynamic link prediction [39, 44]. The task here is to predict whether a link (u, v) will appear or not at the future timestep. The objective of the attacker, thus, is to introduce perturbations in the past time steps to make the model mispredict link's existence in future. We test the victim models on the final snapshot for a set of target links. We consider 3 different sets of 100 positive and 100 negative random targets and show the mean relative ROC-AUC drop with error bars.

Figure 2 shows the performance of different attack methods on this task across different datasets and models. TD-PGD outperforms the other baselines in all cases, except in GC-LSTM model trained on `Reddit`. Moreover, TD-PGD is able to drop the AUC by up to 4 times the baselines and lead to $\sim$ 100% drop in the AUC, completely flipping the prediction. We also note that TD-PGD often has a continuously decreasing slope and its performance saturates much later than the other baselines. The second-best baseline is often TGA(ϵ) but in many cases, it is only as good as random. One can also note that EvolveGCN shows a larger drop than the other 2 models across all datasets. This may pertain to the lower model complexity of EvolveGCN compared to others, which makes it highly sensitive to input perturbations. We provide a comparative analysis of the robustness of these models against TD-PGD attack for different datasets in Appendix A.1.

5.1.2 ***Node Classification.***
In this section, we compare the attack performance on the task of transductive node classification [39]. Here, the objective is to predict the node labels of a set of nodes while knowing the labels of the other nodes at that time step.

Figure 3 shows the effect of structural perturbations on this task by different attack strategies for the 3 models. Misclassifying the labels for influential top-degree targets can significantly impact a model's usability in practice. Therefore, we consider the performance on 50 top-degree nodes for each class. Results show that TD-PGD outperforms the baselines in all models except DySAT, in which all attacks perform almost equally. In particular, TD-PGD is able to cause a 30% drop in EvolveGCN while the baselines only lead to a drop of 5%. We also conducted an attack on the node features by extending the TDAP constraint to features, as mentioned in Section 3.1. We find that such feature perturbations are more effective than structure perturbations for sparsely-connected random targets and can bring the performance down by 30% across models. We defer these results to Appendix A.3.

5.2 Detectability of TDAP perturbations

In this section, we show empirical evidence to complement the theoretical implications (see Section 3.1) on the less detectable nature of TDAP constrained perturbations.

Traditional Methods. Traditional methods flag an instance to be anomalous if the distance (Graph Edit, Hamming, or spectral) with the previous snapshot exceeds a certain threshold [5, 45]. Theorem 1 shows that the average rate of structural change remains preserved within certain factors that can be tuned to fool these detectors. Suppose one had chosen a threshold B such that $d\mathbf{A}_t \geq B$ is defined as an anomaly. If initially the graph sequence was not anomalous on average, *i.e.*, $\overline{d\mathbf{A}} \leq B$, then after perturbation, $\overline{d\mathbf{A}'} \geq B$ would happen if $\overline{d\mathbf{A}'} \geq |1-2\epsilon|\overline{d\mathbf{A}} \geq B$, *i.e.*, $\overline{d\mathbf{A}} \geq B/(|1-2\epsilon|)$. Thus, one can choose a value of ϵ such that this does not hold for a given threshold B and original snapshot dynamics $\overline{d\mathbf{A}}$.

DynGem Anomaly Detection Method. DynGem [18] proposes to use the embedding change in consecutive snapshots to detect anomalies. In particular, these methods consider the consecutive change after perturbation, $d\mathbf{Z}'_t = \|\mathbf{Z}'_t - \mathbf{Z}'_{t-1}\|$ and flag an edge as anomalous if $d\mathbf{Z}'_t$ exceeds a threshold.

Here, we test whether such methods would be effective to detect TDAP perturbations as anomalous by conducting a 2-sample t-test between $d\mathbf{Z}$ and $d\mathbf{Z}'$ (from TD-PGD) for each model-dataset pair at different ϵ values. If they are statistically different, then, the perturbations may be detected as anomalies for a certain threshold. Table 3 notes the raw p-values of the 2-sample t-test between $d\mathbf{Z}'$ and $d\mathbf{Z}$. We find that the null hypothesis of the distributions being the same was rejected (*i.e.*, p-value $\leq$ 0.05) in only 40 of the total 117 cases. In particular, we note that perturbations in EvolveGCN after $\epsilon > 0.2$ may be detected across all datasets while for GC-LSTM, the minimum such ϵ varies from 0.2 (on `UCI`) to 0.6 (on `Reddit`). On the other hand, $d\mathbf{Z}'_t$ in DySAT cannot be statistically distinguished from $d\mathbf{Z}_t$ for any ϵ. We also compare the *range* of consecutive differences between embeddings before and after the perturbation using the "Embedding Variability (EV)" metric (defined

[4]https://pytorch-geometric.readthedocs.io/en/latest/index.html#
[5]https://github.com/FeiGSSS/DySAT_pytorch
[6]https://github.com/jianz94/tga

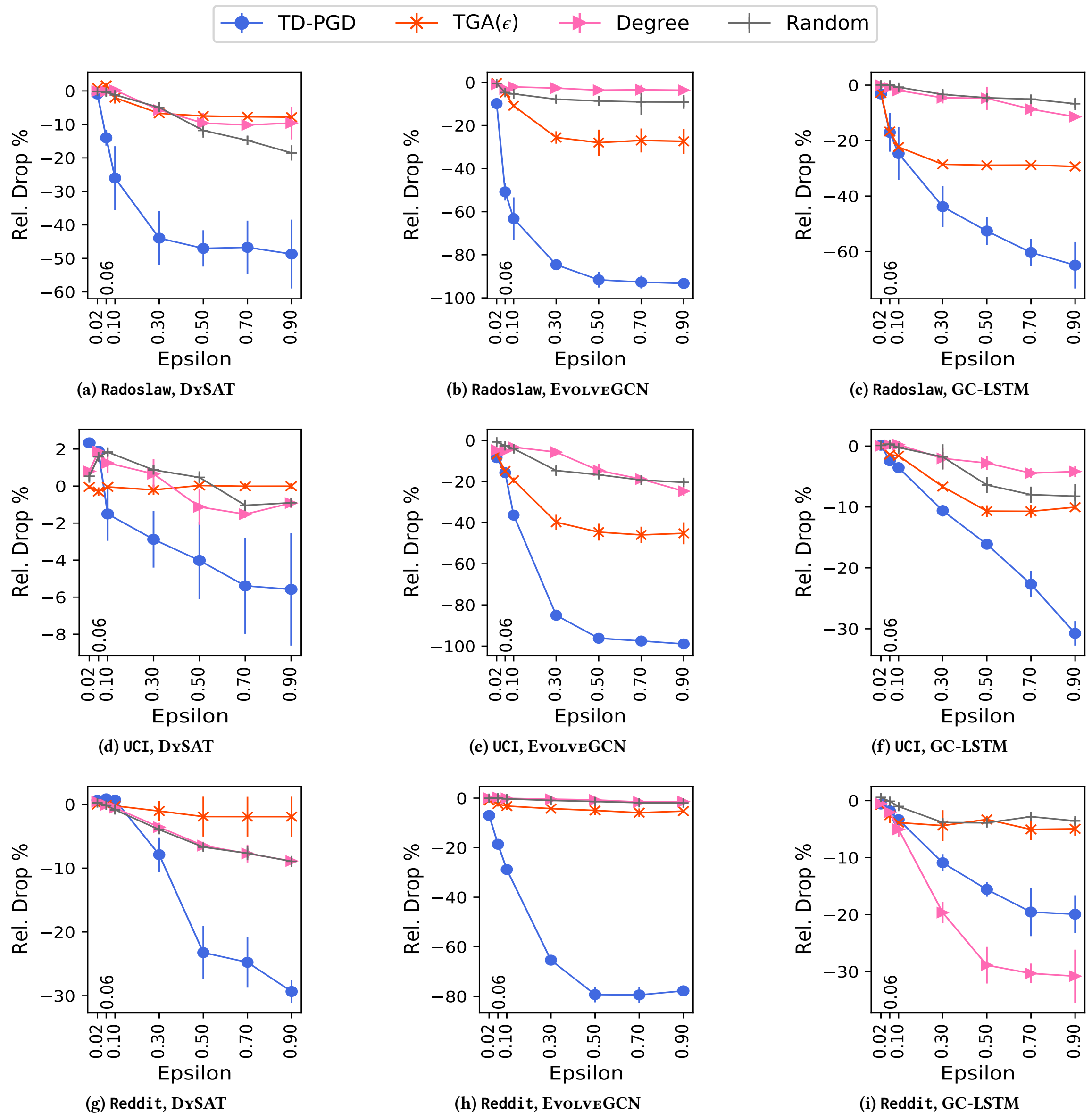


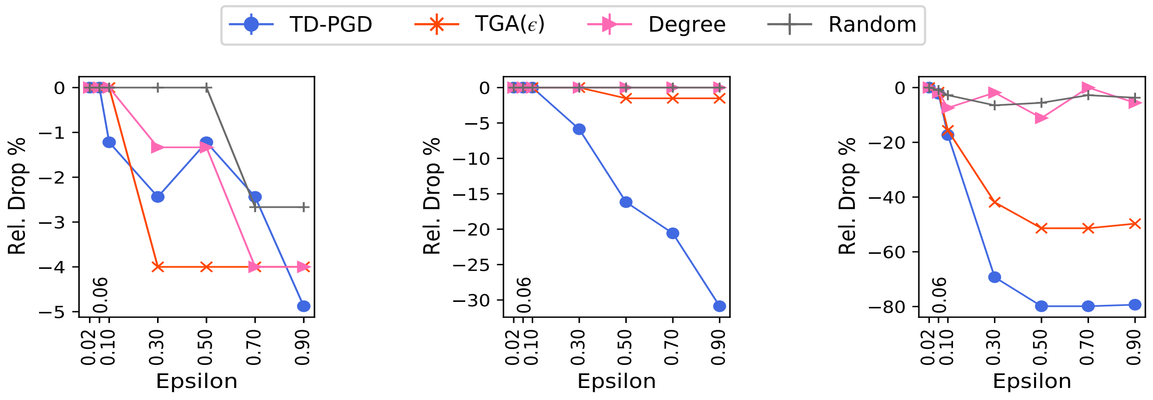
Figure 2: Attack performance on dynamic link prediction task across datasets and models.

and reported in Appendix A.4) to further support the claims of effectiveness against DynGem-based anomaly detection.

NetWalk Anomaly Detection Method. NetWalk [55] clusters the embeddings of a sample set of edges at each timestep and flags an edge to be anomalous if its distance in the embedding space from any cluster exceeds a certain threshold. One can note that TDAP-constrained perturbations would be effective against such a detector as well, due to Proposition 1. The embedding change due to perturbation is only a small factor of the original average rate of change in the adjacency matrices. Thus, the distance of edge embeddings from the cluster centers should also remain bounded.

In order to test the evasion of TDAP perturbations against NetWalk, we used the 3 victim representation models to obtain the embeddings and used the K-Means algorithm with $k = 5$ for clustering at each time step. Perturbations are selected from the TD-PGD algorithm and the edge embeddings are clustered in the original and perturbed embedding space for a fixed set of held-out training edges. The anomalous score is then calculated as the average distance of the perturbed edges to the nearest cluster's centroid in the corresponding embedding space.

We find the anomalous scores for the target edges before and after the perturbation and instead of defining an arbitrary threshold, we conducted a 2-sample t-test between the two distributions at each ϵ for the three models in Radoslaw and UCI datasets. Table 4 notes the raw p-values of the t-test between the Netwalk anomalous scores for the target edges in $\mathbf{Z}'$ and $\mathbf{Z}$. The hypothesis that the



(a) DBLP-5, DySAT (b) DBLP-5, EvolveGCN (c) DBLP-5, GC-LSTM

Figure 3: Attack performance on node classification task for top degree nodes across models.

Table 3: Significance values from 2-sample t-test between dZ' and dZ. Bold rows indicate that the difference is not significant and thus, DynGem-based anomalous scoring would be ineffective to detect TDAP perturbations as made by our method TD-PGD.

Dataset	Model	Epsilon	p-value (dZ', dZ)
Radoslaw	**DySAT**	**[0.02, 0.9]**	**≥ 0.8387**
	EvolveGCN	**[0.02, 0.2)**	**≥ 0.2054**
	EvolveGCN	[0.2, 0.9]	≤ 0.0002*
	GC-LSTM	**[0.02,0.3)**	**≥ 0.1704**
	GC-LSTM	[0.3, 0.9]	≤ 0.0432*
UCI	**DySAT**	**[0.02, 0.9]**	**≥ 0.9252**
	EvolveGCN	**[0.02, 0.3)**	**≥ 0.0878**
	EvolveGCN	[0.3, 0.9]	≤ 0.0197*
	GC-LSTM	**[0.02, 0.2)**	**≥ 0.2674**
	GC-LSTM	[0.2, 0.9]	≤ 0.0378*
Reddit	**DySAT**	**[0.02, 0.9]**	**≥ 0.9636**
	EvolveGCN	**[0.02, 0.3)**	**≥ 0.0735**
	EvolveGCN	[0.3, 0.9]	≤ 0.0271*
	GC-LSTM	**[0.02, 0.6)**	**≥ 0.0554**
	GC-LSTM	[0.6, 0.9]	≤ 0.0270*

two distributions are similar was accepted (*i.e.*, p-value > 0.05) in all but five cases. These cases were EvolveGCN for Radoslaw at $\epsilon = 0.3, 0.5, 0.7, 0.9$ and DySAT for Radoslaw at $\epsilon = 0.9$. In all other cases, we found that the distances from the cluster centroids of these edges are not statistically different before and after TDAP perturbations. Thus, NetWalk detection algorithm fails to detect such perturbations.

5.3 Online Adversarial Attacks

In this section, we consider the online setting as described in Section 3.3 and compare the online version of TD-PGD with the Random and Degree baselines on the dynamic link prediction task. Since the loss at the final step is not available at time step t, one cannot select

Table 4: Significance values from 2-sample t-test between the NetWalk anomalous scores for Z' and Z. Bold rows indicate that the difference is not significant.

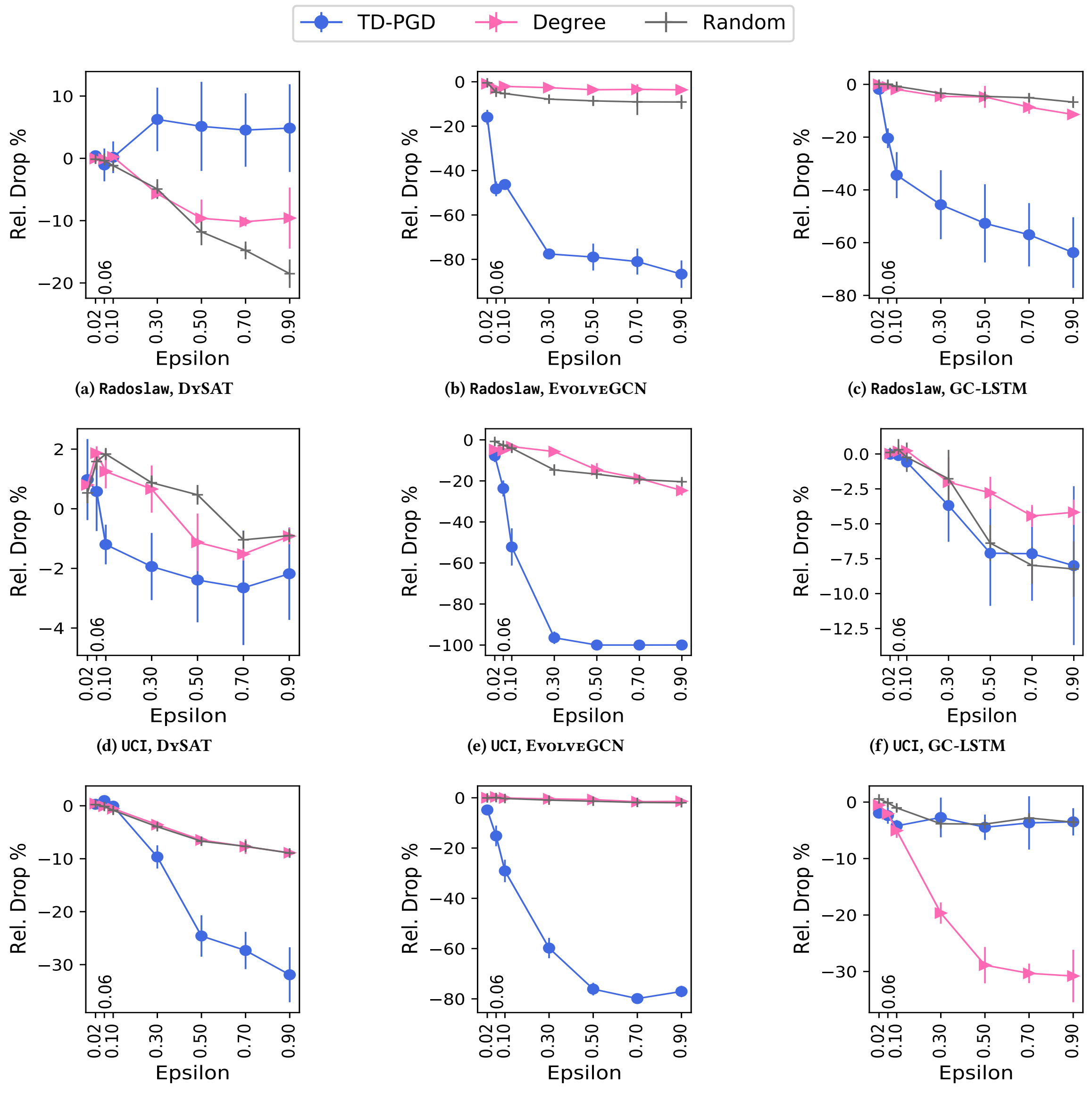
Dataset	Model	Epsilon	p-value (dZ', dZ)
Radoslaw	**DySAT**	**[0.02, 0.7]**	**≥ 0.05301**
	DySAT	0.9	0.0200*
	EvolveGCN	**[0.02, 0.1]**	**≥ 0.2966**
	EvolveGCN	[0.3, 0.9]	≤ 0.00008*
	GC-LSTM	**[0.02, 0.9]**	**≥ 0.1835**
UCI	**DySAT**	**[0.02, 0.9]**	**≥ 0.1005**
	EvolveGCN	**[0.02, 0.1]**	**≥ 0.0581**
	GC-LSTM	**[0.02, 0.9]**	**≥ 0.1806**

the perturbations in a greedy manner of the gradients. Therefore, we do not have a TGA(ϵ) baseline for this setting.

Figure 4 shows the average performance of the three methods for dynamic link prediction task on 3 datasets over 3 random seeds. TD-PGD outperforms the other online baselines in most cases and is able to achieve competent performance to the offline version. In particular, it shows up to 5 times improvement over the existing baselines (for EvolveGCN on UCI), which is close to the offline TD-PGD as shown in Figure 2. However, TD-PGD does not perform well in Figures 4a and 4i. Online TD-PGD perturbs the graph at time t according to the loss at that time step rather than the final step. While it is guaranteed to give strong bounds for a convex objective, some models may learn a complex non-convex function in its input. We conjecture that the degradation may be due to such functions being learned in these cases.

6 CONCLUSION

Our work has shown that state-of-the-art dynamic graph models can be effectively attacked while preserving temporal dynamics. We hope that our work serves as a first step toward opening exciting research avenues for studying attacks and defense mechanisms for both discrete and continuous-time dynamic graphs. Some limitations of our current exposition can be noted. First, the proposed



(a) Radoslaw, DySAT (b) Radoslaw, EvolveGCN (c) Radoslaw, GC-LSTM

(d) UCI, DySAT (e) UCI, EvolveGCN (f) UCI, GC-LSTM

(g) Reddit, DySAT (h) Reddit, EvolveGCN (i) Reddit, GC-LSTM

Figure 4: Online Attack performance on dynamic link prediction task across datasets and models.

method TD-PGD is not memory-efficient and may not scale to larger graphs (due to its $O(|\mathcal{E}|)$ memory complexity for untargeted settings). Second, randomly rounding the solution to discrete space may lead to suboptimal perturbations. Future works can study more effective and efficient methods to attack dynamic graphs under TDAP constraint, possibly in the more restrictive black-box setting. Finally, since TDAP constraint is designed to preserve the local graph (and embedding) evolution, it may be ineffective against frequency-based anomaly detection algorithms [7, 13]. We hope that our work on evolution-preserving attacks inspires others to move away from the impractical budget constraint and explore specific practical constraints for specific domains, such as evolution-based and frequency-based anomaly detection for dynamic graphs.

ACKNOWLEDGEMENTS

This research/material is based upon work supported in part by NSF grants CNS-2154118, IIS-2027689, ITE-2137724, ITE-2230692, CNS-2239879, Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112290102 (subcontract No. PO70745), and funding from Microsoft, Google, and Adobe Inc. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the position or policy of DARPA, DoD, SRI International, NSF, and no official endorsement should be inferred. We thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data mining and Knowledge Discovery* 29, 3 (2015), 626–688.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*. PMLR, 695–704.

[3] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[4] Sébastien Bubeck et al. 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning* 8, 3-4 (2015), 231–357.

[5] Horst Bunke, Peter J Dickinson, Miro Kraetzl, and Walter D Wallis. 2007. *A graph-theoretic approach to enterprise network dynamics*. Vol. 24. Springer Science & Business Media.

[6] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. 2021. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3747–3756.

[7] Yen-Yu Chang, Pan Li, Rok Sosic, MH Afifi, Marco Schweighauser, and Jure Leskovec. 2021. F-fade: Frequency factorization for anomaly detection in edge streams. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 589–597.

[8] Jinyin Chen, Xueke Wang, and Xuanheng Xu. 2018. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv:1812.04206* (2018).

[9] Jinyin Chen, Haiyang Xiong, Haibin Zheng, Jian Zhang, Guodong Jiang, and Yi Liu. 2021. Dyn-Backdoor: Backdoor Attack on Dynamic Link Prediction. *arxiv:2110.03875* (2021).

[10] Jinyin Chen, Jian Zhang, Zhi Chen, Min Du, and Qi Xuan. 2021. Time-aware gradient attack on dynamic network link prediction. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[11] Yongqiang Chen, Han Yang, Yonggang Zhang, MA KAILI, Tongliang Liu, Bo Han, and James Cheng. 2022. Understanding and Improving Graph Injection Attack by Promoting Unnoticeability. In *International Conference on Learning Representations*. https://openreview.net/forum?id=wkMG8cdvh7-

[12] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[13] Dhivya Eswaran and Christos Faloutsos. 2018. Sedanspot: Detecting anomalies in edge streams. In *2018 IEEE International conference on data mining (ICDM)*. IEEE, 953–958.

[14] Houxiang Fan, Binghui Wang, Pan Zhou, Ang Li, Meng Pang, Zichuan Xu, Cai Fu, Hai Li, and Yiran Chen. 2020. Reinforcement learning-based black-box evasion attacks to link prediction in dynamic graphs. *arxiv:2009.00163* (2020).

[15] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of Graph Neural Networks at Scale. *Advances in Neural Information Processing Systems* 34 (2021).

[16] Antonia Gogoglou, Brian Nguyen, Alan Salimov, Jonathan B Rider, and C Bayan Bruss. 2020. Navigating the dynamics of financial embeddings over time. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–8.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

[18] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. Dyngem: Deep embedding method for dynamic graphs. *arXiv:1805.11273* (2018).

[19] Bryan Hooi, Kijung Shin, Hyun Ah Song, Alex Beutel, Neil Shah, and Christos Faloutsos. 2017. Graph-based fraud detection in the face of camouflage. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 4 (2017), 1–26.

[20] Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1520–1529.

[21] Renhe Jiang, Du Yin, Zhaonan Wang, Yizhuo Wang, Jiewen Deng, Hangchen Liu, Zekun Cai, Jinliang Deng, Xuan Song, and Ryosuke Shibasaki. 2021. Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 4515–4525.

[22] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 19–34.

[23] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research* 21, 70 (2020), 1–73.

[24] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[25] Gueorgi Kossinets and Duncan J Watts. 2006. Empirical analysis of an evolving social network. *science* 311, 5757 (2006), 88–90.

[26] Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 933–943.

[27] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1269–1278.

[28] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[29] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*.

[30] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Towards more practical adversarial attacks on graph neural networks. *Advances in neural information processing systems* 33 (2020), 4756–4766.

[31] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 719–728.

[32] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2021. Graph adversarial attack via rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1161–1169.

[33] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2021. Graph Adversarial Attack via Rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1161–1169.

[34] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.

[35] Andjela Mladenovic, Avishek Joey Bose, Hugo Berard, William L Hamilton, Simon Lacoste-Julien, Pascal Vincent, and Gauthier Gidel. 2021. Online Adversarial Attacks. *arxiv:2103.02014* (2021).

[36] Norman Mu and David Wagner. 2021. Defending against adversarial patches with robust self-attention. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*.

[37] Apurva Narayan and Peter HO'N Roe. 2018. Learning graph dynamics using deep neural networks. *IFAC-PapersOnLine* 51, 2 (2018), 433–438.

[38] George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. 2021. Transfer Graph Neural Networks for Pandemic Forecasting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

[39] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5363–5370.

[40] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. 2015. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics* 7, 3 (2015), 223–247.

[41] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

[42] Benedek Rozemberczki, Paul Scherer, Oliver Kiss, Rik Sarkar, and Tamas Ferenci. 2021. Chickenpox cases in hungary: a benchmark dataset for spatiotemporal signal processing with graph neural networks. *arXiv:2102.08100* (2021).

[43] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference 2021*. 2535–2546.

[44] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 519–527.

[45] Peter Shoubridge, Miro Kraetzl, WAL Wallis, and Horst Bunke. 2002. Detection of abnormal change in a time series of graphs. *Journal of Interconnection Networks* 3, 01n02 (2002), 85–101.

[46] Xianfeng Tang, Yozen Liu, Neil Shah, Xiaolin Shi, Prasenjit Mitra, and Suhang Wang. 2020. Knowing your fate: Friendship, action and temporal explanations for user engagement prediction on social apps. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2269–2279.

[47] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*. PMLR, 3462–3471.

[48] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*.

[49] Bin Wang, Teruaki Hayashi, and Yukio Ohsawa. 2020. Hierarchical graph convolutional network for data evaluation of dynamic graphs. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 4475–4481.

[50] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial

Intelligence Organization, 4816–4823. https://doi.org/10.24963/ijcai.2019/669

[51] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Yameng Gu, Xiao Liu, Jingchao Ni, Bo Zong, Haifeng Chen, and Xiang Zhang. 2019. Adaptive neural network for node classification in dynamic networks. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1402–1407.

[52] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*.

[53] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 3961–3967. https://doi.org/10.24963/ijcai.2019/550

[54] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[55] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. 2018. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2672–2681.

[56] Tong Zhao, Bo Ni, Wenhao Yu, Zhichun Guo, Neil Shah, and Meng Jiang. 2021. Action Sequence Augmentation for Early Graph-based Anomaly Detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2668–2678.

[57] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. 2019. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN.. In *IJCAI*. 4419–4425.

[58] Dali Zhu, Yuchen Ma, and Yinlong Liu. 2020. A flexible attentive temporal graph networks for anomaly detection in dynamic networks. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 870–875.

[59] Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*. 928–936.

[60] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.

APPENDIX

A ADDITIONAL EXPERIMENTS

A.1 PGD Performance On Different Models

Figure 5 compares the TD-PGD attack performance over random targets of different victim models for dynamic link prediction. One can note that EvolveGCN is the least robust among these models as TD-PGD causes the most drop in this model across datasets. On the other hand, GC-LSTM and DySAT show similar drops with DySAT being slightly more robust among them. This can be explained due to the implicit robustness of attention-based architecture [20, 36] and lower model complexity of EvolveGCN. GC-LSTM, on the other hand, has a high model complexity as it embeds a GCN in each component of the LSTM module, owing to its robustness.

A.2 Attack Efficiency

Figure 6 compares the running time per target for different attack methods on the largest dataset, `Reddit`. We find that TGA(ϵ) is the most expensive method and scales almost linearly with ϵ (capped at 300 s). TD-PGD takes around half the time than TGA(ϵ) and remains constant with an increase in ϵ. This trend can be attributed to the difference in the complexity of the two methods, as shown in Section 3.2, as the time taken by TD-PGD does not depend on ϵ.

A.3 Node Classification Over Random Targets

Figure 7 compares the drop on the node classification task for different models over random targets. One can note that all attack models perform as well as the other on EvolveGCN and DySAT, while TD-PGD outperforms others on GC-LSTM by a factor of 2. No attack method is found to achieve a significant drop, *i.e.*, below 5%, in the accuracy of DySAT and EvolveGCN on `DBLP-5`.

Feature perturbations. Thus, we explore other ways to attack this task. We find that node attributes are more important for node classification than the temporal structure. Thus, we formulate perturbations over node attributes/features. In particular, we have continuous perturbations $\mathbf{S}_t^X = \mathbf{X}_t' - \mathbf{X}_t$ such that $\|\mathbf{X}_t' - \mathbf{X}_t\| \leq \epsilon d\mathbf{X}_t$ for all t. We then adopt the Algorithm 1 to this problem to find effective feature perturbations. In particular, we replace $\mathbf{s}_t$ to denote the feature perturbation vector at time t, *i.e.*, the vector corresponding to $\mathbf{X}_t' - \mathbf{X}_t$. Finally, since the perturbations are supposed to be in continuous space, we remove the rounding step (line 4) and return the matrix form of $\{\mathbf{s}_t\}$ as $\mathbf{S}^X$. Thus, TD-PGD can be used to find effective attacks in the feature perturbation setting as well.

However, since TGA(ϵ) and Degree takes decisions based on the structure, we omit these baselines for this setting. We use Random to introduce uniformly random perturbations in the feature matrices of the random nodes. Figure 8 compares the attack performance of the two feature perturbation methods on `DBLP-5` over random targets. One can note that TD-PGD is able to achieve around 30% drop for all the models while it could not drop the performance below 5% for these random targets using structural perturbations. This can be explained by the low degree of these targets (average $\sim$ 10 over 10 time steps) which allows for a small no. of perturbations per time step according to the TDAP constraint. Furthermore, the node features here correspond to the word2vec attributes of the author papers while the labels represent the field of the author. Thus, there is a strong connection between the attributes and the downstream labels, which makes feature perturbation more effective than structural co-author perturbations to flip predicted labels for the classification task.

A.4 Detectability of Attack Methods

Embedding Variability. Another way to detect attacks can be extended from [18] by proposing a novel metric **Embedding Variability (EV)**. This compares the consecutive embedding difference for the perturbed graph and that for the original graph. Consecutive embedding difference has been used to identify anomalies in the data [18]. Here, we measure how the range of this difference changes due to the perturbation. In particular, we consider

$$EV(\mathbf{Z}, \mathbf{Z}') := \left| 1 - \frac{\max_\tau d\mathbf{Z}_\tau' - \min_\tau d\mathbf{Z}_\tau'}{\max_\tau d\mathbf{Z}_\tau - \min_\tau d\mathbf{Z}_\tau} \right| \tag{8}$$

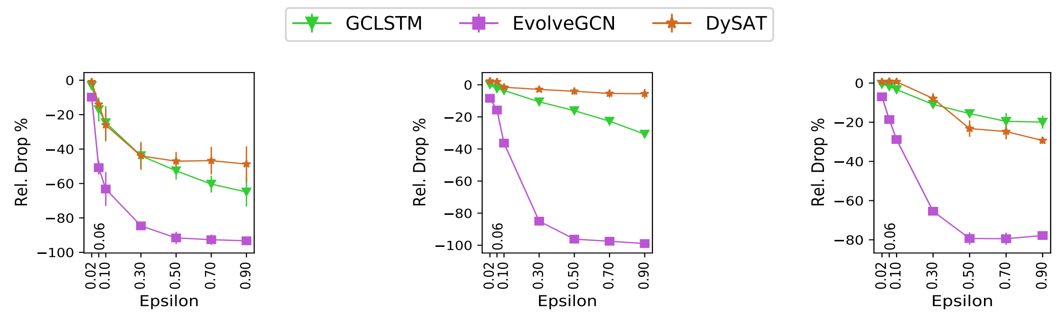
This measures the relative variability of the consecutive change in the embedding space. For the attacks to be less detectable, this metric should be close to 0.

Table 5 compares the attack performance of the best method TD-PGD at $\epsilon = 0.5$ and the corresponding variability in the embeddings, as given by Equation 8. We note the median and 10% and 90% quantile values for the EV metric. One can note that at least 50% of the attacks changed the evolution by a factor of only 1 ± 0.25, as the median EV ≤ 0.25 across datasets. Furthermore, we note that TD-PGD is able to cause up to 90% drop in performance while changing the evolution of 90% of these targets by only 0.62. This shows that TDAP allows for undetectable yet effective attacks. These results further show that anomaly detection methods such as DynGem [18] that bound $d\mathbf{Z}' = \|\mathbf{Z}_t' - \mathbf{Z}_{t-1}'\|$ may fail to detect such attacks as the attacker can choose a specific ϵ at which $d\mathbf{Z}'$ is within a specific factor of $d\mathbf{Z}$ (that is guaranteed by Corollary 1).

We also compare other attack methods for the embedding variability and found Degree to be the most detectable with an EV of almost twice that of TD-PGD. However, regardless of the attack method, TDAP-constrained perturbations are found to have an EV of less than 1 for at least 50% of the targets.

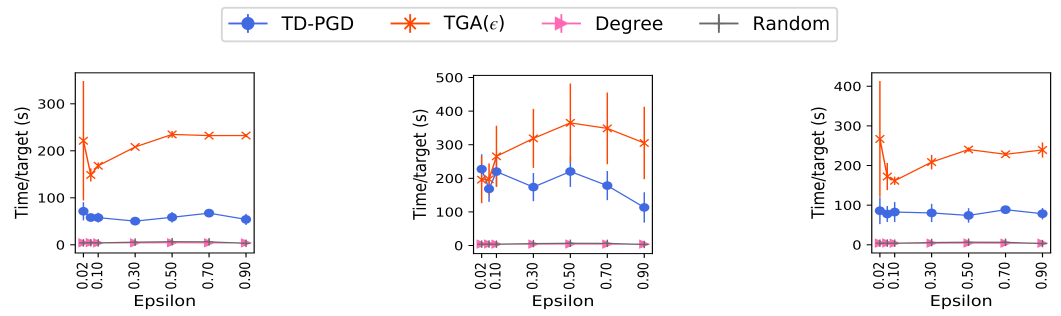
Table 5: Comparison of attack performance and detectability with respect to Embedding Variability (EV) for TD-PGD at $\epsilon = 0.5$. Mean relative drop is noted with the standard deviation in parentheses. While for EV, median values are noted with 10% and 90% quantile values in the parentheses.

Dataset	Model	Rel. Drop %	EV
`Radoslaw`	DySAT	-47.03 (5.42)	0.00 (0.00, 0.04)
	EvolveGCN	-91.61 (3.58)	0.25 (0.03, 0.79)
	GC-LSTM	-52.63 (5.09)	0.07 (0.01, 0.19)
`UCI`	DySAT	-4.02 (2.08)	0.06 (0.01, 0.39)
	EvolveGCN	-96.21 (0.17)	0.11 (0.01, 0.62)
	GC-LSTM	-16.12 (0.75)	0.20 (0.03, 0.85)
`Reddit`	DySAT	-23.24 (4.18)	0.02 (0.00, 0.34)
	EvolveGCN	-79.31 (3.13)	0.13 (0.02, 1.23)
	GC-LSTM	-15.59 (1.28)	0.13 (0.02, 0.84)



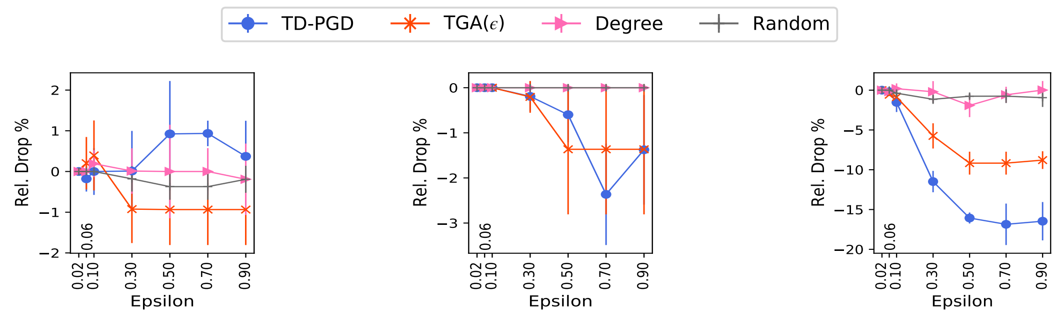
(a) `Radoslaw` (b) `UCI` (c) `Reddit`

Figure 5: TD-PGD performance on dynamic link prediction task across datasets and models.



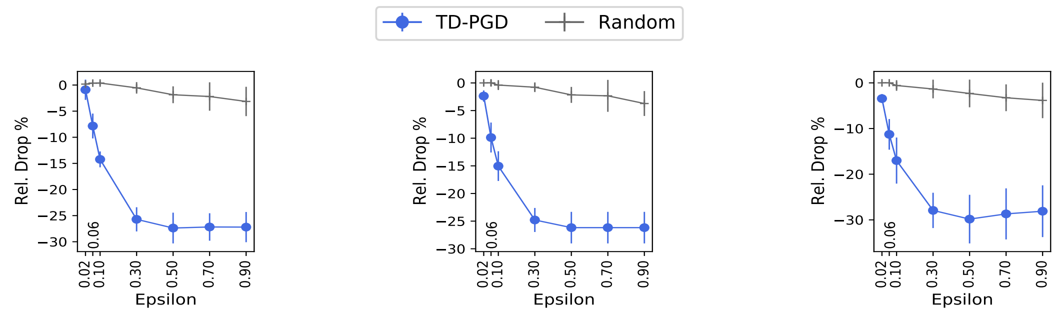
(a) `Reddit`, DySAT (b) `Reddit`, EvolveGCN (c) `Reddit`, GC-LSTM

Figure 6: Running time of different attack methods on the largest dataset (`Reddit`)



(a) `DBLP`, DySAT (b) `DBLP`, EvolveGCN (c) `DBLP`, GC-LSTM

Figure 7: Structural perturbation performance on node classification task over random targets.



(a) `DBLP`, DySAT (b) `DBLP`, EvolveGCN (c) `DBLP`, GC-LSTM

Figure 8: Feature perturbation performance on node classification task over random targets.