Cross-Context Backdoor Attacks against Graph Prompt Learning

Xiaoting Lyu School of Computer Science and Technology Beijing Jiaotong University Beijing, China xiaoting.lyu@bjtu.edu.cn

Hangwei Qian CFAR, A*STAR Singapore qian hangwei@cfar.a-star.edu.sg

Yufei Han Inria, Univ. Rennes, IRISA France yufei.han@inria.fr

Ivor Tsang CFAR, A*STAR Singapore ivor tsang@cfar.a-star.edu.sg

Wei Wang* School of Computer Science and Technology Beijing Jiaotong University Beijing, China wangwei1@bjtu.edu.cn

Xiangliang Zhang University of Notre Dame Indiana, USA xzhang33@nd.edu

ABSTRACT

Graph Prompt Learning (GPL) bridges significant disparities between pretraining and downstream applications to alleviate the knowledge transfer bottleneck in real-world graph learning. While GPL offers superior effectiveness in graph knowledge transfer and computational efficiency, the security risks posed by backdoor poisoning effects embedded in pretrained models remain largely unexplored. Our study provides a comprehensive analysis of GPL's vulnerability to backdoor attacks. We introduce CrossBA, the first cross-context backdoor attack against GPL, which manipulates only the pretraining phase without requiring knowledge of downstream applications. Our investigation reveals both theoretically and empirically that tuning trigger graphs, combined with prompt transformations, can seamlessly transfer the backdoor threat from pretrained encoders to downstream applications. Through extensive experiments involving 3 representative GPL methods across 5 distinct cross-context scenarios and 5 benchmark datasets of node and graph classification tasks, we demonstrate that CrossBA consistently achieves high attack success rates while preserving the functionality of downstream applications over clean input. We also explore potential countermeasures against CrossBA and conclude that current defenses are insufficient to mitigate CrossBA. Our study highlights the persistent backdoor threats to GPL systems, raising trustworthiness concerns in the practices of GPL techniques.

CCS CONCEPTS

• Computing methodologies → Machine learning.

KEYWORDS

Backdoor Attacks, Graph Prompt Learning, Cross-context Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a

KDD'24, August 25-29, 2024, Barcelona, Spain ACM ISBN 978-x-xxxx-xxxx-x/YY/MM https://doi.org/10.1145/nnnnnnn.nnnnnnn

fee. Request permissions from permissions@acm.org.

ACM Reference Format:

Xiaoting Lyu, Yufei Han, Wei Wang, Hangwei Qian, Ivor Tsang, and Xiangliang Zhang. 2024. Cross-Context Backdoor Attacks against Graph Prompt Learning. In KDD'24: SIGKDD Conference on Knowledge Discovery and Data Mining, August 25-29, 2024, Barcelona, Spain. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnnnnnnnnn

INTRODUCTION

Real-world graph learning tasks pose challenges in generalization and knowledge transfer when deploying pretrained graph neural networks (GNNs) to downstream applications divergent from the pretraining stage. For instance, a GNN pretrained on social networks may be utilized in recommendation systems, while encoders designed for link prediction might be repurposed for node or graph classification tasks. The substantial differences between pretraining and downstream applications, including variations in problem domains, semantic space, and learning objectives [13, 22, 23], present obstacles for transferring the graph knowledge in pretrained GNN models to diverse downstream applications. In response, Graph Prompt Learning (GPL) [13, 22-24] has emerged as a promising solution for such graph learning tasks requiring the generalization of graph knowledge across various application contexts (abbreviated as cross-context graph learning). Inspired by prompt learning in Large Language Models (LLMs) [12, 17], GPL involves training GNN encoders initially on unannotated pretext graph data and then tailoring prompts for downstream applications to guide these encoders. This approach effectively bridges the gap between pretraining and downstream tasks without altering the GNN's parameters, thereby avoiding resource-intensive data annotation and model retraining while facilitating robust generalization of pretrained GNN encoders across diverse downstream applications.

While GPL facilitates knowledge transfer across diverse graph learning tasks in cross-context scenarios, it also exposes downstream applications to the risk of inheriting backdoors embedded in pretrained models. Attackers can implant backdoors into pretrained GNN encoders, leading to downstream models built on these encoders inheriting the backdoor poisoning effects and misclassifying backdoored inputs to attacker-desired target labels. Such backdoor vulnerabilities [2, 3, 7, 16, 31, 37] have been identified in prompt learning in Natural Language Processing (NLP), which involves using rare words as triggers and associating them with specific target classes or output embeddings. However, adapting these NLP-based

^{*}Corresponding author

attacks to graph learning encounters challenges due to fundamental differences in data structure and learning paradigms.

Prior research [19, 27-29, 36] has revealed vulnerabilities of GNNs to backdoor attacks, manipulating labeled graph data during supervised training to induce misclassification. However, these attacks are not applicable to GPL scenarios. Unlike traditional GNN tasks, GPL constructs GNN encoders using unlabeled data during unsupervised pretraining, limiting attackers' access to labeled training data for injecting backdoor noise. Recent work [33] targeting graph contrastive learning (GCL) associates triggers with target class embeddings to cause misclassification. However, this method requires prior knowledge about downstream applications, rendering it infeasible in cross-context GPL scenarios where attackers only control the pretraining process without information about downstream applications. Additionally, existing research fails to investigate the generalization and transferability of backdoor attacks across different cross-context GPL scenarios. In summary, organizing successful backdoor attacks against cross-context GPL systems remains largely unexplored.

Presented Work. We study the feasibility of transferable back-door attacks against various GPL methods across diverse cross-context scenarios. Building on prior research [4, 8, 13, 22–24, 34], we categorize cross-context learning into 5 scenarios: cross-task, cross-domain, cross-dataset, cross-class, and cross-distribution, as detailed in Section 3, based on the disparities between pretraining and downstream graph data. These scenarios offer a comprehensive assessment of the inherent backdoor threats to GPL.

Realizing such attacks in the aforementioned cross-context GPL scenarios poses unique challenges compared to conventional graph backdoor attacks. First, attackers in cross-context GPL scenarios can only access and control the unlabeled graph data collected during pretraining. They lack awareness of downstream applications where the model may be deployed, rendering traditional backdoor attacks unfeasible. Second, the divergences between pretraining and downstream applications in semantic spaces, structural patterns, and learning objectives challenge the Independent and Identically Distributed (IID) assumption fundamental to machine learning models, hindering the generalization of the backdoor poisoning effects embedded in the pretrained model. Third, successful backdoor attacks should ensure that the backdoored model remains functional on clean input in downstream applications. Additionally, the attack should retain its effectiveness even when countermeasure mechanisms are deployed by downstream users.

To this end, we propose *CrossBA*, the first cross-context backdoor attack method against GPL, which addresses the aforementioned challenges from the following perspectives.

First of all, CrossBA formulates the backdoor attack as a feature collision-oriented optimization problem during the pretraining stage. CrossBA is designed to simultaneously associate backdoored graphs with the embedding of the trigger graph and ensure that these embeddings are distinct from those of clean graphs. In this way, any backdoored graph will be mapped to the target embedding, causing downstream applications to misclassify it as the target class determined by the target embedding. Furthermore, our theoretical analysis in Section 5.4 unveils the intrinsic vulnerability of GPL systems to backdoor poisoning effects embedded in pretrained GNN models. While prompt learning facilitates knowledge transfer to

downstream applications, it inadvertently amplifies the transferability of backdoors. Additionally, both our theoretical analysis and empirical observations demonstrate that optimizing the trigger graph to align the loss landscape of the backdoor and main learning task can further enhance the transferability of backdoors in cross-context GPL scenarios. *Finally*, to enhance the stealthiness of the attack, *CrossBA* optimizes the trigger graph to align the loss landscape of the backdoor task with that of the main task, while also reinforcing the closeness between the embeddings of the backdoor and backdoor-free GNN encoders on the same clean input graphs, thereby minimizing utility loss in downstream applications. Moreover, *CrossBA* constrains the node features of the trigger graph to closely resemble those of clean nodes, aiding in evading potential countermeasures deployed by downstream users.

We evaluate CrossBA against 3 representative GPL methods, ProG [22], ProG-Meta [22] and GraphPrompt [13], using 5 realworld graph datasets for both graph and node classification tasks. Our evaluation covers 5 various cross-context graph learning scenarios, including cross-distribution, cross-class, cross-dataset, crossdomain, and cross-task, as detailed in Section 3. Despite challenges, CrossBA consistently achieves attack success rates exceeding 0.85 across various downstream applications in all 5 cross-context scenarios, while maintaining high utility with less than a 0.06 drop in classification accuracy compared to backdoor-free counterparts. Comparisons with GCBA [33], a relevant backdoor attack method for GCL, reveal CrossBA's superior performance across diverse cross-context scenarios. For instance, against the GAT model trained by ProG on CiteSeer, CrossBA outperforms GCBA by at least 68% in both accuracy over clean data and attack success rate. Additionally, we discuss potential countermeasures against CrossBA and conclude that the current defenses in GPL are insufficient to mitigate CrossBA, yet effective against GCBA. For example, in cross-domain scenarios, CrossBA achieves attack success rates above 0.90 facing PruneG, while those of the baselines drop below 0.50.

Our contributions focus on answering the 3 research questions: **RQ1:** How does an attacker organize successful cross-context backdoor attacks against GPL systems?

RQ2: Do different GPL methods exhibit equal susceptibility to cross-context backdoor attacks? Is there a shared underlying cause for the vulnerability of various GPL methods to such attacks?

RQ3: Does the threat of backdoor attacks persist across different cross-context GPL scenarios?

To address **RQ1**, we begin by providing an overview of relevant literature and foundational concepts of the GPL framework in Sections 2 and 3. We then delve into the threat model of cross-context backdoor attacks and the design of *CrossBA* in Sections 4 and 5. For **RQ2**, we theoretically analyze the feasibility of *CrossBA* within the GPL framework in Section 5.4, revealing the inherent backdoor vulnerability in GPL. To bolster the findings for **RQ1** and **RQ2**, and to address **RQ3**, we conduct a comprehensive empirical evaluation in Section 6 to demonstrate the persistent threat posed by *CrossBA* across diverse cross-context scenarios and against different GPL methods. Section 7 concludes the entire paper.

2 RELATED WORKS

Graph Prompt Learning. Prompt learning, initially successful in NLP [10], has been extended to graph data, tailoring prompts for downstream tasks to guide the pretrained model to perform effectively without altering its parameters. Prompts in graph learning manifest in two forms: prompt as tokens and prompt as graphs [9, 11, 13, 15, 21–23]. Two representative methods of these GPL methods are GraphPrompt [13] and ProG [22]. Both unify pretraining and downstream tasks into a common template but differ in prompts. ProG uses learnable, graph-structured variables as prompts attached to the input graph, while GraphPrompt embeds prompt tokens as learnable vectors into the hidden layers of the GNN model, enhancing the Readout operation.

Backdoor Attacks. Backdoor attacks on GNNs have gained attention recently [5, 19, 27-29, 36]. In these attacks, the backdoored GNN model predicts an attacker-chosen label for any testing input embedded with triggers. Notably, [36] employs the Erdos-Rényi (ER) model to generate subgraphs as triggers, [27] introduces an adaptive trigger generator enhancing attack effectiveness, [28] designs triggers that preserve the labels of backdoored samples, and [29] proposes unnoticeable graph backdoor attacks to bypass defense mechanisms. However, in cross-context GPL scenarios, traditional graph backdoor attacks are inapplicable, as the attacker only controls the pretraining process with unlabeled data. GCBA [33], targeting GCL, manipulates the victim GNN encoder to associate a trigger with the target class's embedding. Yet, GCBA necessitates knowledge of the target class in downstream tasks, impractical in cross-context GPL scenarios. Our study explores the feasibility of backdoor attacks in cross-context GPL scenarios, where attackers can only use unlabeled data to pretrain GNN encoders. Importantly, attackers cannot access or interfere with downstream applications.

3 PRELIMINARIES

We focus on the workflow of cross-context graph prompt learning [13, 22]. Attackers pretrain the GNN encoder using self-supervised learning on unlabeled graph data. Downstream users then learn the prompts with few-shot training samples based on the pretrained encoder. Relevant concepts and definitions are introduced below. **GNN encoder.** GNNs have become a predominant approach for learning graph embeddings. Typically, GNNs utilize a neighborhood aggregation strategy, wherein the encoder iteratively updates a node's embedding by aggregating embeddings from its neighbors through message passing. Formally, at the k-th layer, the embedding of node v_i is given by:

$$h_{v_i}^k = \text{AGGREGATE}\left(h_{v_i}^{k-1}, \{v_j \in \mathcal{N}(v_i) \cup v_i\}, \theta^{(k)}\right) \tag{1}$$

where $\mathcal{N}(v_i)$ is the set of first-order neighbors of node v_i in the graph G, and the AGGREGATE function combines neighborhood node embeddings to update the node embedding. The final node embedding of v_i is denoted as $h_{v_i} = E_{\theta}(G, v_i)$, where θ denotes the parameters of the encoder E. The graph embedding $E_{\theta}(G)$ is then obtained through a Readout function that aggregates node embeddings from the entire graph. The objective functions of pretraining include various self-supervised tasks such as GraphCL [32] and link prediction [13], which enable the model to capture rich structural and feature-based graph patterns. Our study focuses on inductive

learning, where the GNN encoder's input is the inductive graph of a given node, including the node itself and its k-hop neighbors.

Graph prompt learning. This study is involved in two sophisticated GPL methods: ProG [22] and GraphPrompt [13]. Both ProG and GraphPrompt unify pretraining and downstream tasks into graph-level tasks, introducing learnable prompts to guide these tasks. In ProG, the prompt graph is denoted as $G_{pro} = (P, S)$, where $P = \{v_1^p, v_2^p, \dots, v_{|P|}^p\}$ represents the set of |P| nodes, each characterized by a token vector $\mathbf{p}_i \in \mathbb{R}^{1 \times d}$. The set $S = \{(v_i^p, v_i^p) | v_i^p, v_i^p \in P\}$ defines the prompt graph's topology structure. A prompted graph is obtained by inserting the prompt graph G_{pro} into the input graph G. The parameters for both the prompt graph and the answering function of downstream tasks are optimized through few-shot learning. ProG-Meta enhances ProG by incorporating meta-learning techniques. GraphPrompt introduces prompts within the hidden layers of the GNN model to assist the graph pooling operation. Given the node set $V = \{v_1, v_2, \dots, v_N\}$ in G with each node's embedding h_{v_i} , and a learnable prompt vector p_i for the downstream task j, the prompt-assisted readout operation for graph G is defined as a reweighed readout function, Readout($\{p_i \odot h_{v_i} | v_i \in V\}$), where ⊙ represents element-wise multiplication. The prompt vector p_i updates its parameters by gradient descent to minimize the graph similarity loss.

Cross-context few-shot learning. Cross-context few-shot learning is designed to facilitate rapid adaptation of models to new tasks in diverse application contexts using limited labeled examples. The inherent data heterogeneity in this paradigm results in disparities in both semantic space and data distribution between pretraining and downstream contexts. We summarize the cross-context scenarios studied in previous works [4, 8, 13, 22–24] and delineate five cross-context scenarios to cover different levels of such disparities:

- Cross-task: This setting reflects the divergence in the goal of graph learning tasks. The pretraining may be conducted for classification of an entire graph, such as ENZYMES, while the downstream task involves classification of nodes, like CiteSeer.
- Cross-domain: The pretraining and downstream data originate
 from distinct domains but share the same task type. For instance,
 pretraining could be conducted on commercial product networks
 (e.g., Amazon), while downstream tasks involve academic citation
 networks (e.g., Cora), both focusing on node classification tasks.
- Cross-dataset: This setting involves different datasets within the same domain. For instance, the pretraining is on dataset like Cora, while the downstream task involves a different dataset like CiteSeer. Both are academic citation networks but different.
- Cross-class: Both the pretraining and downstream datasets stem from the same data source, such as Cora. However, they focus on different classes, i.e., pretraining and downstream tasks have different class distributions.
- Cross-distribution: In this scenario, both the pretraining and downstream datasets are sourced from the same data origin, sharing identical features and label spaces. However, they present distinctly different data distributions.

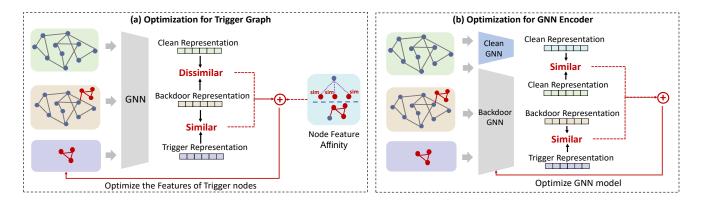


Figure 1: The attack workflow of CrossBA.

4 THREAT MODELS

Attacker's goal. Attackers aim to mislead downstream models built upon the attacker-crafted GNN encoder to classify backdoored inputs as the target class. Simultaneously, the downstream model should behave normally for clean inputs. Specifically, in crosscontext scenarios, attackers build the backdoored GNN encoder at the pretraining phase, which memorizes the association between backdoored inputs and the attacker-desired output embedding. This backdoored GNN encoder is then adapted for downstream applications using the GPL methods. To ensure the transferability of backdoors in cross-context GPL, the attack should be organized with the following properties: 1) Context-agnostic: The backdoor attack should remain consistently effective across various downstream contexts. 2) Prompt-agnostic: The attack should be adaptable to different designs of graph prompt learning methods used by downstream users. 3) Stealthy: The backdoored GNN model should contribute close classification accuracy to backdoor-free models on clean data of downstream applications. Furthermore, the attack remains effective with the defense mechanisms deployed by downstream users.

Attacker's capability. We assume that the attacker possesses complete control over the pretraining process. The attacker can inject backdoor triggers into the unlabeled pretraining data and access the training methods for the backdoored GNN encoder. However, the attacker is incapable of accessing or manipulating the labeled data and the GPL training process utilized by downstream users. Attacker's knowledge. The attacker has full knowledge of the pretraining phase, including the dataset, the architecture of the pretrained GNN encoder, and the training method. On the contrary, the attacker lacks any information regarding the GPL training process conducted by downstream users, including specifics about

5 CROSS-CONTEXT BACKDOOR ATTACK

the downstream datasets utilized and the GPL methods applied.

5.1 Overview of CrossBA

Figure 1 provides an overview of *CrossBA*. In our attack scenario, attackers train the backdoored GNN encoder using unlabeled graph data and self-supervised methods like graph contrastive learning. *CrossBA* aims to inject embedding collisions between backdoored

graphs and the trigger graph into the backdoored GNN encoder, while maintaining a distinct embedding for the backdoored graph compared to its clean counterpart. This backdoor poisoning effect causes the backdoored graph to be associated with a target embedding distant from the clean input, leading to misclassification in downstream models. Additionally, CrossBA also optimizes the standard contrastive learning objective over clean graphs to ensure robust classification performance.

Formally, the trigger graph serves as the backdoor signal is $\Delta_G = (V_\Delta, E_\Delta)$, with $V_\Delta = \{v_\Delta^1, \dots, v_\Delta^C\}$ representing the set of C trigger nodes. Each trigger node has a feature vector $\mathbf{x}_\Delta^i \in \mathbb{R}^{1 \times d}$, matching the input graph's node feature dimension. $E_\Delta = \{(v_\Delta^i, v_\Delta^j) | v_\Delta^i, v_\Delta^j \in V_\Delta\}$ defines the links of the trigger graph. To integrate the trigger graph Δ_G into the input graph G for generating the backdoored graph, the attacker randomly selects a node in G as the anchor node v_{att} , linking it to a specific node v_Δ^i in the trigger graph. We limit the trigger graph to a 3-node and fully connected graph, significantly smaller than graphs in both pretraining and downstream datasets.

5.2 Attack Objective of CrossBA

We define the objective function of the *CrossBA* attack in Eq.2, which jointly optimizes the backdoored GNN encoder parameters θ_b and the node features of the trigger graph Δ_G .

$$\begin{split} \theta_b^*, \Delta_G^* &= \underset{\theta_b, \Delta_G}{\arg\min} \, \mathcal{L}_{\text{bdk}} + \mathcal{L}_{\text{clr}} + \alpha \, \mathcal{L}_{\text{sim}}^c + \beta \, \mathcal{L}_{\text{sim}}^a, \\ \text{where} \quad \mathcal{L}_{\text{bdk}} &= -\frac{1}{|D|} \sum\limits_{G_i \in D} \sin \left(\hat{E}_{\theta_b} \left(G_i \oplus \Delta_G \right), \hat{E}_{\theta_b} \left(\Delta_G \right) \right) \\ &+ \lambda \frac{1}{|D|} \sum\limits_{G_i \in D} \sin \left(\hat{E}_{\theta_b} \left(G_i \oplus \Delta_G \right), \hat{E}_{\theta_b} \left(G_i \right) \right), \\ &\underset{\text{exp} \left(\sin(H_{G_i}, H_{G_i^+}) / \tau \right)}{\exp\left(\sin(H_{G_i}, H_{G_i^+}) / \tau \right)} \\ \mathcal{L}_{\text{clr}} &= -\frac{1}{|D|} \sum\limits_{G_i \in D} \log \frac{\exp\left(\sin(H_{G_i}, H_{G_i^+}) / \tau \right)}{\exp\left(\sin(H_{G_i}, H_{G_i^-}) / \tau \right)}, \\ \mathcal{L}_{\text{sim}}^c &= -\frac{1}{|D|} \sum\limits_{G_i \in D} \sum\limits_{v_\Delta^i \in V_\Delta} \sin \left(\hat{E}_{\theta_b} \left(G_i \right), E_{\theta_c} \left(G_i \right) \right), \\ \mathcal{L}_{\text{sim}}^a &= -\frac{1}{|D|} \sum\limits_{G_i \in D} \sum\limits_{v_\Delta^i \in V_\Delta} \sin \left(\text{feat} (v_\Delta^i), \text{feat} (v_{att}) \right). \end{split}$$

where \hat{E}_{θ_b} represents the backdoored GNN encoder, with θ_b denoting its parameters. G_i is a clean graph in the pretraining dataset D. G_i^+ is an augmented graph of G_i obtained by randomly adding or removing links, while G_k^- is a graph different from G_i in D. The operation \oplus attaches the trigger graph Δ_G into the anchor node v_{att}

of the clean graph G. feat (v_i) denotes the feature vector of a node v_i . E_{θ_c} is the backdoor-free GNN encoder trained with clean graph data. $H_{G_i} = \hat{E}_{\theta_b}(G_i)$ denotes the embedding of the clean graph G_i . sim(u,v) is the similarity between the embeddings u and v. τ is the temperature parameter. The parameters λ , α , and β balance the weight of the loss terms.

The main learning loss \mathcal{L}_{clr} defines a contrastive learning objective of the main task as in [32]. The goal is to train a GNN encoder capable of producing distinctive embeddings for clean graph data. Specifically, we maximize the similarity between the embeddings of the graph G_i and its augmented counterpart G_i^+ . Simultaneously, we maximize the dissimilarity between the embedding of any other graph G_i^- and that of G_i .

The backdoor learning loss \mathcal{L}_{bdk} defines the backdoor task's objective. To address the attacker's lack of knowledge about downstream datasets, CrossBA utilizes the embedding of the trigger graph as the target, and induces the backdoor mapping into the GNN encoder by colliding the embeddings of backdoored graphs with that of the trigger graph. By jointly tuning the backdoored GNN encoder and the trigger graph, we ensure that the embeddings of backdoored graphs resemble the target embedding while being different from clean graphs' embeddings. Consequently, when applied to downstream applications, backdoored graphs will be misclassified to the target class associated with the target embedding.

Compared to manually specifying a static backdoor mapping (with a fixed trigger graph and target embedding), our design offers several advantages. First, by jointly tuning the target embedding and trigger graph, we align the loss landscapes of the main task and the backdoor task within the fixed GNN encoder. This minimizes the backdoor learning loss while preserving the utility of the backdoored GNN encoder on clean graph data. Second, fine-tuning the trigger graph, as supported by our theoretical analysis in Section 5.4, reduces the disparity in backdoor task performance between pretraining and downstream applications, thereby enhancing backdoor transferability. Lastly, we parameterize the tuning of the target embedding by optimizing the trigger graph, ensuring it remains within the embedding space of the graphs in D. Directly optimizing the target embedding as an independent variable may lead to extreme values outside the span of graphs in D, making them prone to detection by downstream anomaly detection methods.

The embedding alignment loss $\mathcal{L}_{\mathrm{sim}}^c$ is designed to ensure that backdoors do not impair the GNN encoder's ability to generate discriminative embeddings for clean graph data. The attacker can build a clean GNN encoder E_{θ_c} using clean pretraining data as a reference model. By aligning the output embeddings of E_{θ_c} with those of \hat{E}_{θ_b} on the same clean inputs, the backdoored GNN encoder \hat{E}_{θ_b} can perform similarly to the clean encoder E_{θ_c} on clean data.

The node feature affinity loss \mathcal{L}^a_{sim} is designed to enhance the stealthiness of CrossBA to evade anomaly detection-based sanitary checks over node features. Since adjacent nodes in a graph typically share similar features, downstream users can check the node feature consistency to identify abnormal nodes with significantly deviated feature values from their neighbors [6, 33]. To circumvent such defenses, we optimize the node features of the trigger graph by

minimizing $\mathcal{L}_{\text{sim}}^a$, enforcing feature consistency between the nodes in the trigger graph and the anchor nodes in the clean input graph.

5.3 Alternating Optimization for CrossBA

Algorithm 1 in Appendix A outlines the procedural flow of the CrossBA attack. Initially, attackers employ self-supervised methods, such as GraphCL [32], to train a clean GNN encoder E_{θ_c} by minimizing \mathcal{L}_{clr} . The trigger injection is then conducted by optimizing the attack objective function in Eq. 2 in alternating order. During each attack round, attackers first freeze the GNN encoder \hat{E}_{θ_b} and optimize the node features of the trigger graph Δ_G . Subsequently, attackers optimize the backdoored GNN encoder \hat{E}_{θ_b} based on the optimized trigger graph Δ_G^* and the clean GNN encoder E_{θ_c} .

Tuning trigger graph. After injecting the trigger graph Δ_G into the input graph G, the attacker optimizes the node features of Δ_G , potentially optimizing the target embedding, by minimizing the backdoor learning loss \mathcal{L}_{bdk} and the node feature affinity loss $\mathcal{L}_{\text{sim}}^a$ with respect to the fixed GNN encoder \hat{E}_{θ_b} . For simplicity, the update of the trigger node features in one step is given by:

$$\Delta_G^t = \Delta_G^{t-1} - \gamma_t \nabla_{\Delta_G^{t-1}} (\mathcal{L}_{\text{bdk}} + \beta \mathcal{L}_{\text{sim}}^a)$$
 (3)

Tuning backdoored GNN encoder. Upon completing trigger optimization, the attacker connects the optimized trigger graph Δ_G to the anchor node in G, recreating backdoored graphs. The optimization aims to maximize the similarity between the embeddings of backdoored graphs and the trigger graph, as well as the similarity of the clean graph's embeddings between \hat{E}_{θ_b} and E_{θ_c} . For simplicity, the GNN encoder parameters are updated accordingly:

$$\begin{aligned} \theta_b^t &= \theta_b^{t-1} - \gamma_{\mathcal{G}} \nabla_{\theta_b^{t-1}} \left(-\frac{1}{|D|} \sum_{G_i \in D} \operatorname{sim} \left(\hat{E}_{\theta_b} \left(G_i \oplus \Delta_G^t \right), \hat{E}_{\theta_b} \left(\Delta_G \right) \right) \\ &- \alpha \frac{1}{|D|} \sum_{G_i \in D} \operatorname{sim} \left(\hat{E}_{\theta_b} \left(G_i \right), E_{\theta_c} \left(G_i \right) \right) \end{aligned} \tag{4}$$

5.4 Attack Feasibility of CrossBA

In this section, we explore the feasibility of the proposed CrossBA attack against cross-context GPL. To simplify the analysis, we adopt the prompt graph setting from [22], where the prompt graph G_{pro} is attached to the input graph G. We define D_s and D_t as the distributions for the pretraining and downstream graph datasets, respectively. Downstream users create the prompted graph by $G_t \otimes G_{pro}$. The encoder outputs the embedding of the prompted graph as $\hat{h}(G_t \otimes G_{pro}) = \hat{E}_{\theta_h}(G_t \otimes G_{pro})$.

Theorem 1. Assuming a $L_{\hat{E}}$ -Lipschitz continuous GNN encoder \hat{E}_{θ_b} with k_G -node input graphs, and the node feature matrix X of a graph G=(V,E) has a bounded Frobenius norm, i.e., $|X|_{fro} \leq \mu$. Upon freezing the GNN encoder \hat{E}_{θ_b} , the backdoor learning loss \mathcal{L}_{bdk} in Eq.2 is upper-bounded by the main learning loss \mathcal{L}_{clr} at the pretraining stage, as given in Eq. 5:

$$\mathcal{L}_{\text{bdk}}^{s} \leq \mathcal{L}_{\text{clr}}^{s} + 2\sqrt{d(\Delta_{G}^{+}, G_{s} \oplus \Delta_{G})} + C$$

$$d(\Delta_{G}^{+}, G_{s} \oplus \Delta_{G}) = 2\sqrt{k_{G}}(k_{G} - 1)L_{\hat{E}}\mu - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(\Delta_{G,i}^{+}), \hat{h}(G_{s,j} \oplus \Delta_{G}))}{nm}$$
(5)

Similarly, the main learning loss in the downstream context can be upper bounded by the main learning loss in the pretraining context:

$$\mathcal{L}_{clr}^{t} \leq \mathcal{L}_{clr}^{s} + 2\sqrt{d(G_{s}, G_{t} \otimes G_{pro})} + C_{0}$$

$$d(G_{s}, G_{t} \otimes G_{pro}) = 2\sqrt{k_{G}}(k_{G} - 1)L_{\hat{E}}\mu - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(G_{s,i}).\hat{h}(G_{t,j} \otimes G_{pro}))}{nm}$$
(6)

Combing Eq.5 and Eq.6, the backdoor learning loss in the downstream context can be upper bounded as in Eq.7:

$$\mathcal{L}_{\text{bdk}}^{t} \leq \mathcal{L}_{\text{clr}}^{s} + \left(8\sqrt{k_{G}}(k_{G} - 1)L_{\hat{E}}\mu - 4\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{s(\hat{h}(\Delta_{G,i}^{t}),\hat{h}(G_{s,j}\oplus\Delta_{G}))}{nm}\right)^{1/2} + \left(8\sqrt{k_{G}}(k_{G} - 1)L_{\hat{E}}\mu - 4\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{s(\hat{h}(G_{s,i}\oplus\Delta_{G}),\hat{h}((G_{t,j}\oplus\Delta_{G})\otimes G_{pro}))}{nm}\right)^{1/2} + C_{1}$$

where Δ_G^+ are augmented trigger graphs with randomly added or removed links from Δ_G . G_s and G_t are the graphs sampled from the pretraining and downstream data distribution D_s and D_t , respectively. The RKHS kernel function s(*,*) measures the similarity between two graph embeddings. C, C_0 , and C_1 are constants for the GNN encoder's complexity and the optimal learning loss of an ideal encoder.

Observation 1. Transferability of graph prompt learning v.s. transferability of backdoor attacks.

Proposition 1. Given a backdoored GNN encoder \hat{E}_{θ_b} , there exists a prompt graph G_{pro} that maximizes the similarity values $\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}\frac{s(\hat{h}(G_{s,i}),\hat{h}(G_{t,j}\otimes G_{pro}))}{nm} \text{ between the graphs from the pretraining dataset and the prompted graphs from the downstream dataset.}$

By combining Proposition 1, Eq.6, and Eq.7, we discover that employing prompts in the cross-context GPL presents both advantages and disadvantages. On one hand, augmenting downstream graph data from D_t with the prompt G_{pro} enhances the adaptability of pretrained GNN encoders to downstream tasks by alleviating distributional disparities between D_t and D_s within the GNN encoder's embedding space. This results in a well-trained GNN encoder capable of generating discriminative embeddings for new tasks. However, on the other hand, as indicated by Eq.7, incorporating the prompt graph into the backdoored graphs from D_t inadvertently facilitates backdoor attack transferability by reducing the upper bound of the backdoor learning loss in downstream tasks.

Observation 2. Tuning the trigger graph enhances the transferability of backdoor poisoning effects while concurrently preserving the utility of backdoored GNN models.

PROPOSITION 2. Given a backdoored GNN encoder \hat{E}_{θ_b} , there exists a trigger graph Δ_G that maximizes the similarity measure $\frac{1}{nm}\sum\limits_{i=1}^n\sum\limits_{j=1}^m s(\hat{h}(\Delta_{G,i}^+),\hat{h}(G_{s,j}\oplus\Delta_G))$ between the augmented variants of the trigger graph and the backdoored graphs in pretraining.

In CrossBA, we propose optimizing the trigger graph Δ_G with a fixed GNN encoder during the pretraining stage to minimize the backdoor learning loss. This process, as indicated by Eq.5 and Proposition 2, improves the alignment between the embeddings of Δ_G^+ and $G_S \oplus \Delta_G$, narrowing the disparity between the main learning loss and the backdoor learning loss on pretraining data. This ensures that given a well-trained GNN encoder, the trigger tuning module reduces the backdoor learning loss without compromising

the performance of the main task. Furthermore, as shown in Eq.7, tuning the trigger graph during pretraining, along with the prompt graph, further facilitates backdoor attack transferability by lowering the upper bound of the backdoor learning loss in downstream tasks, leading to misclassification with backdoored input.

The theoretical investigation elucidates the feasibility of delivering cross-context backdoor attacks following the design of *CrossBA*, providing a response to **RQ1**. Moreover, Observation 1, which addresses **RQ2**, reveals the dual nature of knowledge transfer within GPL's prompt learning. While prompt learning enhances downstream models with the pretrained model's expertise, it also poses the risk of backdoor transfer. Our findings underscore substantial concerns regarding the trustworthiness of GPL methods. Proofs of Theorem 1 and Propositions 1 and 2 are provided in Appendix B.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Settings

Datasets and the Backbone GNN Models. We utilize 5 benchmark datasets for evaluation: CiteSeer [30], Cora [30], Amazon-Computers [18], Amazon-Photo [18], and ENZYMES [1]. CiteSeer, Cora, Amazon-Computers, and Amazon-Photo are utilized for node classification tasks, while ENZYMES is employed for graph classification tasks. Furthermore, following the setting in [22], we define graph classification tasks using the node classification datasets. In GPL systems, we employ two advanced GNN models: Graph Attention Network (GAT) [25] and Graph Transformer (GT) [20]. Details about datasets and GNN models can be found in Appendix C.

GPL Methods. We evaluate the attack performance against mainstream GPL methods applicable to both node and graph classification tasks, categorized into two types [23]: *Prompt as Graphs* and *Prompt as Tokens.* For the former branch, we choose ProG [22] and ProG-Meta [22], formulating prompts as subgraph patterns. For the latter, we target GraphPrompt [13], which considers prompts as tokens added to the Readout module of GNNs.

Baseline Attacks. GCBA [33] emerges as the most relevant backdoor attack for our study, considering the threat model. While not explicitly tailored for GPL, GCBA aims to inject backdoor poisoning noise into a GNN encoder trained via GCL. In GCBA, the attacker gathers graph data of the target class in downstream applications and utilizes the GCBA-crafting method to inject the backdoor into the GNN encoder. However, GCBA does not directly apply to crosscontext GPL scenarios as it necessitates access to downstream applications. We introduce two variants of GCBA adapted to our threat model: GCBA R and GCBA M. In both variants, the attacker initially clusters the embeddings of clean data collected during the pretraining stage, utilizing the backdoor-free GNN encoder. Subsequently, GCBA_R randomly selects the embedding at the center of a cluster as the target embedding, while GCBA_M chooses the cluster center embedding farthest from other clusters as the target embedding. Further details can be found in Appendix D.

Evaluation Metrics. We employ 3 metrics to evaluate attack effectiveness: (1) Attack Success Rate (ASR) [33], representing the accuracy with which a backdoored downstream model classifies backdoored inputs to the target class designated by the target embedding, (2) Accuracy of the Main Task (ACC), measuring the classification accuracy of backdoored downstream models over clean

			Node Classification									Graph Classification			
GPL	Model	Attack	CiteSeer		Cora		Computers		Photo		CiteSeer-Graph		Photo-Graph		
			ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	
ProG	GAT	GCBA_R GCBA_M CrossBA	0.24 (+0.60) 0.24 (+0.60) 0.83(+0.01)	0.51 0.62 0.90	0.33(+0.32) 0.18(+0.47) 0.64 (+0.01)	0.76 0.00 1.00	0.43(+0.29) 0.40(+0.32) 0.70(+0.02)	0.18 0.44 1.00	0.61(+0.18) 0.65(+0.14) 0.79(-0.00)	0.15 0.18 0.94	0.17(+0.54) 0.17(+0.54) 0.76(-0.05)	0.00 0.00 1.00	0.66(+0.23) 0.69(+0.20) 0.89(-0.00)	0.24 0.15 0.91	
	GT	GCBA_R GCBA_M CrossBA	0.35(+0.47) 0.35(+0.47) 0.82(-0.00)	0.96 0.96 1.00	0.35(+0.38) 0.45(+0.28) 0.73(-0.00)	0.69 0.23 0.99	0.47(+0.27) 0.41(+0.33) 0.72(+0.02)	1.00 0.75 1.00	0.57(+0.22) 0.60(+0.19) 0.79(-0.00)	0.85 0.30 1.00	0.37(+0.42) 0.33(+0.46) 0.80(-0.01)	0.32 0.21 1.00	0.58(+0.33) 0.57(+0.34) 0.92(-0.01)	0.89 0.56 0.99	
Graph Prompt	GAT	GCBA_R GCBA_M CrossBA	0.69(+0.07) 0.78(-0.02) 0.80(-0.04)	0.05 0.01 1.00	0.47(+0.11) 0.43(+0.15) 0.67(-0.09)	0.14 0.07 1.00	0.49(+0.11) 0.43(+0.17) 0.62(-0.02)	0.14 0.13 1.00	0.59(+0.07) 0.56(+0.10) 0.68(-0.02)	0.00 0.15 0.99	0.72(-0.00) 0.67(+0.05) 0.72(-0.00)	0.04 0.13 0.99	0.58(+0.20) 0.51(+0.27) 0.74(+0.04)	0.00 0.28 1.00	
	GT	GCBA_R GCBA_M CrossBA	0.69(+0.09) 0.64(+0.14) 0.79(-0.01)	0.16 0.04 1.00	0.56(+0.15) 0.53(+0.18) 0.70(+0.01)	0.19 0.25 0.99	0.50(+0.15) 0.47(+0.18) 0.67(-0.02)	0.15 0.10 1.00	0.62(+0.13) 0.59(+0.16) 0.74(+0.01)	0.13 0.12 1.00	0.66(+0.09) 0.74(+0.01) 0.77(-0.02)	0.10 0.11 1.00	0.55(+0.29) 0.57(+0.27) 0.84(-0.00)	0.25 0.14 1.00	
ProG Meta	GAT	GCBA_R GCBA_M CrossBA	0.85(-0.17) 0.50(+0.18) 0.84(-0.16)	0.56 0.00 1.00	0.50(+0.38) 0.50(+0.38) 0.89(-0.01)	0.00 0.00 1.00	0.57(+0.25) 0.57(+0.25) 0.79(+0.03)	0.00 0.00 1.00	0.75(+0.24) 0.50(+0.49) 0.99(-0.00)	0.88 0.00 1.00	0.50(+0.38) 0.50(+0.38) 0.85(+0.03)	0.00 0.00 1.00	0.86(+0.14) 0.98(+0.02) 1.00(-0.00)	0.94 0.00 1.00	
	GT	GCBA_R GCBA_M CrossBA	0.93(-0.02) 0.50(+0.41) 0.93(-0.02)	0.90 0.00 1.00	0.90(+0.01) 0.88(+0.03) 0.90(+0.01)	0.31 0.05 1.00	0.57(+0.29) 0.57(+0.29) 0.85(+0.01)	0.00 0.00 1.00	0.97(+0.02) 0.99(-0.00) 0.99(-0.00)	0.71 0.10 1.00	0.92(+0.03) 0.87(+0.08) 0.94(+0.01)	0.05 1.00 1.00	0.50(+0.50) 1.00(-0.00) 1.00(-0.00)	0.00 0.98 1.00	

Table 1: ACC, ASR, and AD in cross-distribution scenarios.

graph data, and (3) Accuracy Drop (AD) [33], denoting the difference in ACC between backdoor-free and backdoored downstream models. A lower AD indicates less utility loss of backdoored GNN encoders. A successful backdoor attack should ensure high ACC (low AD) and high ASR values simultaneously.

Implementations. For the details about the implementations, including the cross-context scenarios, the GPL methods, and all the attacks, please refer to Appendix E. We provide codes in the link 1 .

6.2 Experimental Results

(1) Attack Performance in Cross-Context GPL Scenarios. We evaluate the feasibility of backdoor attacks on node and graph classification tasks across 5 cross-context scenarios using 3 mainstream GPL methods, addressing 3 research questions. Tables 1–3 and 5–6 present the ACC and ASR of all attack methods in cross-distribution, cross-class, cross-domain, cross-dataset, and cross-task scenarios, respectively. Due to space limitations and consistent trends across different scenarios, we report results for cross-distribution, cross-class, and cross-domain scenarios here and defer those for cross-dataset and cross-task scenarios to Appendix F. The highest ASR and ACC values, and the lowest AD values achieved among all the attacks, including our *CrossBA*, are highlighted in bold.

Superior attack performance of CrossBA across different cross-context scenarios (RQ1 and RQ3). The results reveal that CrossBA consistently outperforms the baseline methods across 5 cross-context GPL scenarios, achieving the highest ASR values while maintaining comparable ACC levels to those of backdoorfree models. Specifically, CrossBA achieves ASR values exceeding 0.87 in all scenarios, with only a negligible difference in ACC compared to the backdoor-free models, at most 0.06 lower. In contrast, the baseline attacks fail to achieve comparable ASR to CrossBA across various scenarios while maintaining ACC simultaneously.

Summary. CrossBA effectively generalizes the backdoor poisoning effects to diverse cross-context scenarios, outperforming two baseline methods, while maintaining the utility of GNN encoders for downstream tasks. This observation confirms the theoretical principles outlined in Section 5.4, highlighting the efficacy of CrossBA in enhancing backdoor transferability in various cross-context scenarios while preserving model utility in downstream tasks.

Consistent attack performance of CrossBA against different GPL methods (RQ2). CrossBA demonstrates robust attack effectiveness across various GPL methods, achieving high ASR while preserving the usability of backdoored models in downstream tasks. Specifically, against all 3 GPL methods, CrossBA consistently achieves ASR values above 0.90 in cross-class, cross-distribution, cross-dataset, and cross-domain scenarios. In cross-task scenarios, CrossBA achieves ASRs exceeding 0.87. In contrast, baseline attacks fail to deliver consistent attack performance against different GPL methods. For instance, in cross-class scenarios, both baseline methods show ASR values below 0.05 against GAT models trained by all three GPL methods on CiteSeer, indicating a complete failure of the attack. The baseline methods achieve ASR values over 0.99 against ProG on Cora but only reach ASRs of 0.07 for GraphPrompt.

For example, the 2 baseline methods never achieve ASR above 0.32 against the GT model trained by GraphPrompt in all 5 scenarios, and their ASR values against GAT models trained by all 3 GPL methods on CiteSeer are below 0.48 in cross-dataset scenarios. Additionally, when the baselines achieve comparable ASR to *CrossBA*, they suffer a significant drop in ACC. For instance, in cross-class scenarios against ProG, the baseline attacks achieve an ASR of about 0.99 on Cora, similar to *CrossBA*, but with an ACC at least 0.27 lower. Similarly, in cross-distribution scenarios against the GT model trained by ProG, the baseline attacks achieve an ASR of about 0.96, close to *CrossBA*, but with ACC values almost half of *CrossBA*'s.

 $^{^{1}}https://github.com/xtLyu/CrossBA\\$

Table 2: ACC, ASR, and AD in cross-class scenarios.

		Attack	Node Classification								Gra	Graph Classification			
GPL	Model		CiteSeer		Cora		Computers		Photo		CiteSeer-Graph		Photo-Graph		
			ACC(AD)	ASR											
ProG	GAT	GCBA_R GCBA_M CrossBA	0.25(+0.68) 0.25(+0.68) 0.93(-0.00)	0.00 0.00 1.00	0.47(+0.31) 0.49(+0.29) 0.78(-0.00)	1.00 1.00 1.00	0.55(+0.28) 0.56(+0.27) 0.83(-0.00)	1.00 0.95 1.00	0.73(+0.06) 0.68(+0.11) 0.79(-0.00)	0.21 0.36 1.00	0.26(+0.62) 0.47(+0.41) 0.87(+0.01)	0.00 1.00 1.00	0.80(+0.12) 0.79(+0.13) 0.93(-0.01)	0.84 1.00 1.00	
	GT	GCBA_R GCBA_M CrossBA	0.49(+0.44) 0.49(+0.44) 0.93(-0.00)	0.29 0.34 1.00	0.61(+0.27) 0.52(+0.36) 0.88(-0.00)	0.99 1.00 1.00	0.60(+0.22) 0.70(+0.12) 0.82(-0.00)	0.72 1.00 1.00	0.62(+0.13) 0.69(+0.06) 0.76(-0.01)	1.00 0.71 1.00	0.49(+0.39) 0.66(+0.22) 0.88(-0.00)	0.46 0.26 1.00	0.76(+0.10) 0.80(+0.06) 0.88(-0.02)	0.01 0.80 1.00	
Graph Prompt	GAT	GCBA_R GCBA_M CrossBA	0.92(-0.04) 0.89(-0.01) 0.92(-0.04)	0.05 0.02 1.00	0.86(-0.01) 0.81(+0.04) 0.87(-0.02)	0.07 0.03 1.00	0.69(+0.06) 0.70(+0.05) 0.77(-0.02)	0.10 0.06 1.00	0.74(-0.04) 0.60(+0.10) 0.72(-0.02)	0.01 0.00 0.94	0.87(-0.03) 0.84(-0.00) 0.83(+0.01)	0.06 0.06 1.00	0.61(+0.17) 0.67(+0.11) 0.79(-0.01)	0.28 0.27 1.00	
	GT	GCBA_R GCBA_M CrossBA	0.87(+0.06) 0.88(+0.05) 0.93(-0.00)	0.03 0.05 1.00	0.76(+0.11) 0.83(+0.04) 0.88(-0.01)	0.03 0.00 1.00	0.72(+0.08) 0.70(+0.10) 0.80(-0.00)	0.08 0.05 1.00	0.66(+0.12) 0.66(+0.12) 0.79(-0.01)	0.03 0.15 1.00	0.88(-0.01) 0.87(-0.00) 0.89(-0.02)	0.05 0.04 1.00	0.59(+0.21) 0.66(+0.14) 0.87(-0.07)	0.24 0.15 1.00	
ProG Meta	GAT	GCBA_R GCBA_M CrossBA	0.50(+0.30) 0.50(+0.30) 0.75(+0.05)	0.00 0.00 1.00	0.50(+0.14) 0.50(+0.14) 0.80(-0.16)	0.00 0.00 1.00	0.57(+0.24) 0.57(+0.24) 0.80(+0.01)	0.00 0.00 1.00	0.99(-0.00) 0.76(+0.23) 0.99(-0.00)	1.00 1.00 1.00	0.50(+0.42) 0.50(+0.42) 0.94(-0.02)	0.00 0.00 1.00	1.00(-0.00) 0.88(+0.12) 1.00(-0.00)	1.00 1.00 1.00	
	GT	GCBA_R GCBA_M CrossBA	0.50(+0.45) 0.77(+0.18) 0.95(-0.00)	0.00 1.00 1.00	0.52(+0.35) 0.60(+0.27) 0.84(+0.03)	0.00 1.00 1.00	0.57(-0.28) 0.57(-0.28) 0.85(-0.00)	0.00 0.00 1.00	0.99(+0.01) 0.99(+0.01) 1.00(-0.00)	1.00 1.00 1.00	0.50(+0.44) 0.58(+0.36) 0.95(-0.01)	0.00 0.02 1.00	0.87(+0.13) 0.94(+0.06) 1.00(-0.00)	0.38 1.00 1.00	

Table 3: ACC, ASR, and AD in cross-domain scenarios. Photo is used as the pretraining dataset.

		Attack	No	de Cla	ssification		Graph Classification					
GPL	Model		CiteSee	r	Cora		CiteSeer-G	raph	Cora-Graph			
	Model		ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR		
		GCBA_R	0.24(+0.50)	1.00	0.18(+0.40)	1.00	0.31(+0.41)	0.80	0.14(+0.74)	0.00		
	GAT	GCBA_M	0.20(+0.54)	0.00	0.32(+0.26)	1.00	0.20(+0.52)	0.88	0.14(+0.74)	0.00		
ProG		CrossBA	0.76(-0.02)	1.00	0.73(-0.15)	1.00	0.72(-0.00)	1.00	0.85(+0.03)	1.00		
1100		GCBA R	0.40(+0.43)	1.00	0.34(+0.39)	0.00	0.33(+0.47)	1.00	0.14(+0.76)	0.00		
	GT	GCBA_M	0.42(+0.41)	1.00	0.33(+0.40)	1.00	0.44(+0.36)	1.00	0.40(+0.50)	1.00		
		CrossBA	0.83(-0.00)	1.00	0.73(-0.00)	1.00	0.80(-0.00)	1.00	0.90(-0.00)	1.00		
		GCBA_R	0.75(-0.03)	0.85	0.68(-0.00)	0.24	0.59(+0.15)	0.00	0.78(+0.05)	0.60		
	GAT	GCBA_M	0.77(-0.05)	0.71	0.58(+0.10)	0.00	0.63(+0.11)	0.03	0.64(+0.19)	0.23		
Graph		CrossBA	0.76(-0.04)	1.00	0.70(-0.02)	1.00	0.74(-0.00)	1.00	0.81(+0.02)	1.00		
Prompt		GCBA_R	0.64(+0.15)	0.19	0.57(+0.14)	0.02	0.68(+0.08)	0.00	0.74(+0.10)	0.04		
	GT	GCBA_M	0.70(+0.09)	0.00	0.53(+0.18)	0.00	0.58(+0.18)	0.00	0.76(+0.08)	0.00		
		CrossBA	0.79(-0.00)	0.99	0.72(-0.01)	0.99	0.77(-0.01)	1.00	0.82(+0.02)	1.00		
		GCBA_R	0.50(+0.34)	0.89	0.50(+0.39)	0.03	0.50(+0.38)	0.00	0.50(+0.26)	0.00		
	GAT	GCBA M	0.50(+0.34)	0.00	0.50(+0.39)	0.00	0.50(+0.38)	1.00	0.50(+0.26)	0.00		
ProG		CrossBA	0.80(+0.04)	1.00	0.88(+0.01)	1.00	0.90(-0.02)	1.00	0.87(-0.11)	1.00		
Meta		GCBA_R	0.91(+0.02)	1.00	0.83(+0.07)	0.52	0.84(+0.08)	1.00	0.96(+0.01)	1.00		
	GT	GCBA_M	0.50(+0.43)	0.00	0.50(+0.40)	1.00	0.50(+0.42)	0.00	0.50(+0.47)	0.00		
		CrossBA	0.93(-0.00)	1.00	0.90(-0.00)	1.00	0.94(-0.02)	1.00	0.95(+0.02)	1.00		

Summary. The results reveal a common vulnerability of *CrossBA* to diverse GPL techniques. This aligns with the theoretical foundations presented in Section 5.4, highlighting the intrinsic risks within the GPL framework that the prompt learning module of GPL facilitates the transfer of backdoors to downstream applications. (2) *Potential Countermeasures*. We evaluate the resilience of *CrossBA* against potential countermeasures of downstream users. Given the absence of specific defenses tailored for cross-context GPL, we adapt *PruneG* [26], originally designed to mitigate adversarial attacks on GNNs. Other defense methods, such as *RandSample* [36] and *GNNGuard* [35], are not suitable for cross-context GPL

scenarios. This is because both <code>RandSample</code> and <code>GNNGuard</code> entail training GNN encoders, which is impractical in cross-context GPL. Therefore, we exclusively employ <code>PruneG</code> in our evaluation. <code>PruneG</code> is a preprocessing technique that eliminates edges between nodes with dissimilar features, removing components with fewer connected nodes. Figure 2 illustrates the effectiveness of different attack methods against <code>PruneG</code> across 5 cross-context scenarios. <code>CrossBA</code> consistently outperforms the baselines, achieving high ASR values exceeding 0.70 against <code>PruneG</code>. In cross-domain scenarios, <code>CrossBA</code> achieves ASRs above 0.90, while the ASRs of the baselines remain below 0.50. Figure 3 in Appendix G demonstrates

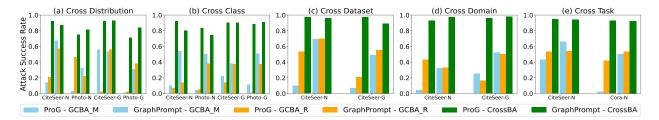


Figure 2: ASR of backdoor attacks against PruneG in 5 cross-context scenarios. "-N" denotes the node classification task, while "-G" represents the graph classification task.

that CrossBA ensures trigger node features closely resemble those of clean nodes, achieving indistinguishable similarity for trigger edges, affirming its stealthiness. More details are provided in Appendix G. (3) Ablation Study. We conduct ablation studies on CrossBA to evaluate the significance of its components. Three variants of CrossBA are considered: (1) CrossBA without trigger optimization, using a fixed trigger graph and target embedding; (2) CrossBA without embedding alignment; (3) CrossBA without node feature affinity. The attack performance of CrossBA and its variants against 2 GPL methods with PruneG across 5 cross-context scenarios is presented in Figure 4 in Appendix H. The results show that removing any component significantly reduces ASRs. The ASR value of CrossBA remains the highest one compared to the variants. These findings affirm the necessity of integrating trigger graph optimization, embedding alignment-based regularization, and node feature affinity constraint together to achieve successful cross-context backdoor attacks against GPL.

(4) Impact of Prompt Tokens. Figure 5 in Appendix I illustrates how the number of prompt tokens affects the attack performance of CrossBA against ProG across 5 cross-context scenarios. The results unveil that the effectiveness of CrossBA remains robust, exhibiting little impact from variations in the number of prompt tokens.

(5) Impact of Trigger Nodes. Figure 6 in Appendix J illustrates the impact of the number of trigger nodes on *CrossBA*'s attack performance against ProG and GraphPrompt across five different scenarios. The results demonstrate that *CrossBA* consistently achieves ASR values exceeding 0.80, regardless of the number of trigger nodes, highlighting the attack's robustness and efficiency.

7 CONCLUSION AND FUTURE WORK

In this study, we conduct theoretical and empirical investigations to assess the feasibility of backdoor attacks in cross-context graph prompt learning. Our findings reveal that optimizing trigger graphs, coupled with prompt transformations, significantly enhances backdoor transferability. We introduce *CrossBA*, the first cross-context backdoor attack tailored for GPL, and evaluate its performance across five cross-context scenarios encompassing node and graph classification tasks. Additionally, we explore potential defenses against such attacks. Our results demonstrate that *CrossBA* seamlessly embeds backdoors into various downstream models across diverse cross-context GPL scenarios without compromising main task performance, even under defense deployment. However, we

acknowledge that our study does not address backdoor attack transferability in textual attribute graphs. As a result, we aim to investigate the transferability of backdoors within textual attribute graphs in the future.

8 ACKNOWLEDGMENT

This research is supported in part by Systematic Major Project of China State Railway Group Corporation Limited (No.P2023W002), the French National Research Agency with the reference ANR-23-IAS4-0001 (CKRISP), and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

REFERENCES

- Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl_1 (2005), i47-i56.
- [2] Xiangrui Cai, haidong xu, Sihan Xu, Ying Zhang, and Xiaojie Yuan. 2022. Bad-Prompt: Backdoor Attacks on Continuous Prompts. In Advances in Neural Information Processing Systems, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=rlN6fO3OrP
- [3] Kangjie Chen, Yuxian Meng, Xiaofei Sun, Shangwei Guo, Tianwei Zhang, Jiwei Li, and Chun Fan. 2022. BadPre: Task-agnostic Backdoor Attacks to Pre-trained NLP Foundation Models. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net.
- [4] Mouxiang Chen, Zemin Liu, Chenghao Liu, Jundong Li, Qiheng Mao, and Jianling Sun. 2023. ULTRA-DP: Unifying Graph Pre-training with Multi-task Graph Dual Prompt. CoRR abs/2310.14845 (2023). arXiv:2310.14845
- [5] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. 2023. Unnoticeable Backdoor Attacks on Graph Neural Networks. In Proceedings of the ACM Web Conference 2023 (Austin, TX, USA) (WWW '23). Association for Computing Machinery, New York, NY, USA, 2263–2273. https://doi.org/10.1145/3543507. 3583392
- [6] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. 2023. Unnoticeable backdoor attacks on graph neural networks. In Proceedings of the ACM Web Conference 2023. 2263–2273.
- [7] Wei Du, Peixuan Li, Boqun Li, Haodong Zhao, and Gongshen Liu. 2023. UOR: Universal Backdoor Attacks on Pre-trained Language Models. CoRR abs/2305.09574 (2023). arXiv:2305.09574
- [8] Taoran Fang, Yunchao Zhang, Yang Yang, and Chunping Wang. 2022. Prompt Tuning for Graph Neural Networks. CoRR abs/2209.15240 (2022). arXiv:2209.15240
- [9] Taoran Fang, Yunchao Mercer Zhang, Yang Yang, Chunping Wang, and Lei CHEN. 2023. Universal Prompt Tuning for Graph Neural Networks. In Thirty-seventh Conference on Neural Information Processing Systems. https://openreview.net/ forum?id=0LmWBhIYLi
- [10] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021. Association for Computational Linguistics, 3816–3830.

- [11] Qingqing Ge, Zeyuan Zhao, Yiding Liu, Anfeng Cheng, Xiang Li, Shuaiqiang Wang, and Dawei Yin. 2023. Enhancing Graph Neural Networks with Structure Based Prompt. CoRR abs/2310.17394 (2023). https://doi.org/10.48550/ARXIV. 2310.17394 arXiv:2310.17394
- [12] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Comput. Surv. 55, 9 (2023), 195:1-195:35. https://doi.org/10.1145/3560815
- [13] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023. ACM, 417-428.
- [14] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR, Vol. 37). JMLR.org, 97-105.
- [15] Yihong Ma, Ning Yan, Jiayu Li, Masood S. Mortazavi, and Nitesh V. Chawla. 2023. HetGPT: Harnessing the Power of Prompt Tuning in Pre-Trained Heterogeneous Graph Neural Networks. CoRR abs/2310.15318 (2023). https://doi.org/10.48550/ ARXIV.2310.15318 arXiv:2310.15318
- [16] Kai Mei, Zheng Li, Zhenting Wang, Yang Zhang, and Shiqing Ma. 2023. NOTABLE: Transferable Backdoor Attacks Against Prompt-based NLP Models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023. Association for Computational Linguistics, 15551-15565.
- [17] Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020. Getting Closer to AI Complete Question Answering: A Set of Prerequisite Real Tasks. Proceedings of the AAAI Conference on Artificial Intelligence 34, 05 (Apr. 2020), 8722-8731. https://doi.org/10.1609/aaai.v34i05.6398
- [18] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018).
- [19] Yu Sheng, Rong Chen, Guanyu Cai, and Li Kuang. 2021. Backdoor Attack of Graph Neural Networks Based on Subgraph Trigger. In Collaborative Computing: Networking, Applications and Worksharing - 17th EAI International Conference, CollaborateCom 2021, Virtual Event, October 16-18, 2021, Proceedings, Part II (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 407). Springer, 276-296.
- [20] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semisupervised classification. arXiv preprint arXiv:2009.03509 (2020).
- [21] Reza Shirkavand and Heng Huang. 2023. Deep Prompt Tuning for Graph Transformers. CoRR abs/2309.10131 (2023). https://doi.org/10.48550/ARXIV.2309.10131 arXiv:2309 10131
- [22] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (KDD'23) (Long Beach, CA, USA). 2120-2131.
- [23] Xiangguo Sun, Jiawen Zhang, Xixi Wu, Hong Cheng, Yun Xiong, and Jia Li. 2023. Graph Prompt Learning: A Comprehensive Survey and Beyond. arXiv:2311.16534 (2023). arXiv:2311.16534
- [24] Zhen Tan, Ruocheng Guo, Kaize Ding, and Huan Liu. 2023. Virtual Node Tuning for Few-shot Node Classification. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023. ACM, 2177-2188.
- [25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. stat 1050, 20 (2017),
- [26] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. arXiv preprint arXiv:1903.01610 (2019).
- [27] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph Backdoor. In 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021. USENIX Association, 1523-1540.
- [28] Jing Xu and Stjepan Picek. 2022. Poster: Clean-label Backdoor Attack on Graph Neural Networks. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11,
- [29] 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 2019, Macao, China, August 10-16, 2019. ijcai.org, 3961–3967.
- [30] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semisupervised learning with graph embeddings. In International conference on machine learning. PMLR, 40-48.
- [31] Hongwei Yao, Jian Lou, and Zhan Qin. 2023. PoisonPrompt: Backdoor Attack on Prompt-based Large Language Models. arXiv:2310.12439 [cs.CL]

- [32] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. Advances in neural information processing systems 33 (2020), 5812-5823.
- Hangfan Zhang, Jinghui Chen, Lu Lin, Jinyuan Jia, and Dinghao Wu. 2023. Graph Contrastive Backdoor Attacks. In International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202). PMLR, 40888-40910.
- Qiannan Zhang, Shichao Pei, Qiang Yang, Chuxu Zhang, Nitesh V. Chawla, and Xiangliang Zhang. 2023. Cross-Domain Few-Shot Graph Classification with a Reinforced Task Coordinator. In Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023. AAAI Press, 4893-4901
- [35] Xiang Zhang and Marinka Zitnik. 2020. GNNGUARD: defending graph neural networks against adversarial attacks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (Vancouver, BC, Canada) (NIPS'20). Curran Associates Inc., Red Hook, NY, USA, Article 777, 13 pages.
- [36] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor Attacks to Graph Neural Networks. In SACMAT '21: The 26th ACM Symposium on Access Control Models and Technologies, Virtual Event, Spain, June 16-18, 2021. ACM, 15-26.
- [37] Shuai Zhao and Jinming Wen. 2023. Prompt as Triggers for Backdoor Attack: Examining the Vulnerability in Language Models. https://synthical.com/article/ 7b1c5bc5-4335-4a54-b52b-1289fae245d3. arXiv:2305.01219 [cs.AI]

A OPTIMIZATION ALGORITHM

The pseudocode for the proposed CrossBA attack method is shown in Algorithm 1. In this study, we introduce an optimization frame-

Algorithm 1 CrossBA

Input: The pretraining dataset $D = \{G_1, G_2, \dots, G_N\}$, the regularization weights α , β , and λ , the learning rate of the trigger graph γ_t , the learning rate of the GNN encoder γ_q , the number of attack rounds T.

Output: The optimized trigger graph Δ_G and the backdoor poisoned GNN encoder model \hat{E}_{θ_b} .

- 1: Initialize Δ_G^0 and θ_c^0 . 2: Train the clean GNN encoder E_{θ_c} based on D by minimizing \mathcal{L}_{clr} .
- 3: $\theta_b^0 \leftarrow \theta_c, t \leftarrow 1$.
- 4: while $t \le T$ do
- For each graph G in D, connect the trigger graph Δ_G^{t-1} to the anchor node to form a backdoored graph G'.
- Obtain Δ_G^t using Eq.3.
- Update the trigger graph embedded in the backdoored graphs with the optimized trigger graph Δ_C^t .
- Obtain θ_h^t using Eq.4.
- 9: end while

work for joint tuning of the trigger graph and a backdoored GNN encoder. Let *n* denote the number of training samples, *d* represent the feature dimensionality, R be the number of training rounds for the clean encoder, and A reflect the time complexity of optimizing the trigger. Initially, the clean GNN encoder is trained for *R* rounds, with each round having a time complexity of $O(nd^2)$, where the d^2 term arises from pairwise feature interactions. Given the *R* rounds, the initial training phase has a complexity of $O(Rnd^2)$. During the attack phase, the trigger is optimized using all samples with complexity O(An). Subsequently, the optimizer is employed to finetune the backdoored GNN encoder. The overall time complexity for the attack process thus becomes $O(Rnd^2 + T(An + nd^2))$, where *T* represents the number of attack rounds.

B PROOFS OF THEOREM AND PROPOSITION

DEFINITION 1.1. Anchored classifier. We define a classifier f anchored at an arbitrarily given graph $G_{anchor} \in D$, where D denotes the distribution of the graph data. E is the graph encoding module of f. Given an input graph G, the classifier f_{anchor} is formulated based on the RBF kernel as follows:

$$f_{anchor}(G) = exp(-\gamma(||E(G) - E(G_{anchor})||^2)$$
(8)

where γ controls the smoothness of the kernel. f_{anchor} produces a soft decision output normalized between 0 and 1. A higher/lower output from f_{anchor} indicates that G has a more/less similar embedding to that of G_{anchor} .

With slight reformulations, the loss functions of the main learning task \mathcal{L}_{clr} and the backdoor learning task \mathcal{L}_{bdk} in Eq.2 can be unified as a cross-entropy loss of the anchored classifier. This can be expressed as:

$$\mathcal{L} = \underset{G \sim D}{\mathbb{E}} \underset{Ganchor}{\mathbb{E}} - I(G \in \{G^{\text{pos}}_{\text{anchor}}\}) f_{G_{\text{anchor}}}(G)$$
$$-I(G \in \{G^{\text{neg}}_{\text{anchor}}\}) (1 - f_{G_{\text{anchor}}}(G))$$
(9)

where $\{G_{\text{anchor}}^{\text{pos}}\}$ and $\{G_{\text{anchor}}^{\text{neg}}\}$ are the sets of graphs forming positive and negative pairs with the given anchor graph G_{anchor} . \mathcal{L} simplifies to \mathcal{L}_{clr} in Eq.2 if 1) $\{G_{\text{anchor}}^{\text{pos}}\}$ denotes the set of

 \mathcal{L} simplifies to \mathcal{L}_{clr} in Eq.2 if 1) $\{G_{anchor}^{pos}\}$ denotes the set of augmented graphs derived from the trigger graph $G_{anchor} = \Delta_G$ by adding or removing links randomly from Δ_G ; and 2) G_{anchor}^{neg} represents any graph sampled from distribution D that is not in the set of augmented graphs.

Similarly, \mathcal{L} simplifies to \mathcal{L}_{bdk} in Eq.2 when 1) G_{anchor}^{pos} is defined as $G \oplus \Delta_G$ with $G \in D$; and 2) G_{anchor}^{neg} follows the same setting as in \mathcal{L}_{clr} , being any graph from D other than Δ_G .

In this sense, the main learning task aims to minimize \mathcal{L} , with the distribution of input graphs as $D_{G^+_{\mathrm{anchor}}} \cup D$, where $D_{G^+_{\mathrm{anchor}}}$ denotes the distribution of the augmented graphs of G_{anchor} . In contrast, the backdoor learning task minimizes \mathcal{L} over a different distribution of input graphs as $D_{G \oplus G_{\mathrm{anchor}}} \cup D$.

Proof to Theorem 1. For any anchored classifier f, we treat $D_{G^+_{\text{anchor}}} \cup D$ and $D_{G \oplus G_{\text{anchor}}} \cup D$ as the data distributions of the source and target problem domain. Additionally, within the anchored classifier, we designate the graph encoder E as the backdoored encoder \hat{E}_{θ_b} , which is trained during the pretraining stage. Following Theorem 1 in [14], the backdoor learning loss during the pretraining phase is upper bounded, as shown in Eq.10:

$$\mathcal{L}_{\text{bdk}}^{s} \leq \mathcal{L}_{\text{clr}}^{s} + 2\sqrt{d(\Delta_{G}^{+}, G_{s} \oplus \Delta_{G})} + C$$

$$-\sum_{i,j=1}^{m} \frac{n^{2}}{n^{2}}$$

$$d(\Delta_{G}^{+}, G_{s} \oplus \Delta_{G}) = \sum_{i,k=1}^{n} \frac{s(\hat{h}(\Delta_{G,i}^{+}), \hat{h}(\Delta_{G,k}^{+}))}{n^{2}} + \sum_{i,j=1}^{m} \frac{s(\hat{h}(G_{s,i} \oplus \Delta_{G}), \hat{h}(G_{s,i} \oplus \Delta_{G}))}{n^{2}} - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(\Delta_{G,i}^{+}), \hat{h}(G_{s,j} \oplus \Delta_{G}))}{nm} - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(G_{s,i} \oplus \Delta_{G}), \hat{h}((G_{t,j} \oplus \Delta_{G}) \otimes G_{pro}))}{nm}$$

$$(10)$$

where C represents a constant that encapsulates the model complexity of \hat{E}_{θ_b} and the minimum achievable loss with the ideal parameters for \hat{E}_{θ_b} . s(,) is an RKHS kernel function measuring the similarity between graph embeddings produced by $\hat{E}_{\theta b}$.

Furthermore, for the simplicity of analysis, we assume the encoder \hat{E}_{θ_b} is a GIN model defined by: $\hat{h}(G) = (A + (1 + \epsilon)I)X\theta_b$.

A denotes the adjacency matrix corresponding to the input graph G. The matrix X is a feature matrix, where each row corresponds to the features of node v_i of G. θ_b is a multi-layer encoder with a Lipschitz constant denoted by $L_{\hat{E}}$. The first two terms are upper bounded independently of the kernel choice.

$$\begin{split} s(\hat{h}(\Delta_{G,i}^{+}), \hat{h}(\Delta_{G,k}^{+})) &\leq \|\hat{h}(\Delta_{G,i}^{+})\| \\ &\leq \sqrt{k_G} \|\hat{h}(\Delta_{G,i}^{+})\|_{fro} \\ &\leq \sqrt{k_G} (k_G - 1) L_{\hat{E}} \|(A + (1 + \epsilon)I)X\| \quad \text{\vdash Lipschitz-continuity condition} \\ &\leq \sqrt{k_G} (k_G - 1) L_{\hat{E}} \|A + (1 + \epsilon)I\|_2 \|X\|_{fro} \\ &\leq \sqrt{k_G} (k_G - 1) L_{\hat{E}} \|X\|_{fro} \\ &\leq \sqrt{k_G} (k_G - 1) L_{\hat{E}} \mu \quad \text{\vdash Bounded frobenius norm} \end{split}$$

where $\|\|$ denotes an abitrary norm of $\hat{h}(\Delta_{G,i}^+)$. The same bound applies to $s(\hat{h}(G_{s,j} \oplus \Delta_G), \hat{h}(G_{s,l} \oplus \Delta_G))$. Integrating Eq.11 into Eq.10, we can derive Eq.5 in Theorem 1.

Similarly, we establish the proof for Eq.6 as follows. Following Theorem 1 in [14], we derive the upper bound for the main learning task in downstream applications, as indicated in Eq.12:

$$\mathcal{L}_{clr}^{t} \leq \mathcal{L}_{clr}^{s} + 2\sqrt{d(G_{s}, G_{t} \otimes G_{pro})} + C_{0}$$

$$d(G_{s}, G_{t} \otimes G_{pro}) = \sum_{i,j=1}^{n} \frac{s(\hat{h}(G_{s,i}), \hat{h}(G_{s,j}))}{n^{2}}$$

$$+ \sum_{i,j=1}^{m} \frac{s(\hat{h}(G_{t,i} \otimes G_{pro}), \hat{h}(G_{t,j} \otimes G_{pro}))}{m^{2}}$$

$$- \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(G_{s,i}), \hat{h}(G_{t,j} \otimes G_{pro}))}{nm}$$

$$(12)$$

Using Eq.11, we can obtain $s(\hat{h}(G_{s,i}), \hat{h}(G_{s,j})) \leq \sqrt{k_G}(k_G-1)L_{\hat{E}}\mu$ and similarly, $s(\hat{h}(G_{t,i}), \hat{h}(G_{t,j})) \leq \sqrt{k_G}(k_G-1)L_{\hat{E}}\mu$. Then, we derive Eq.6 by incorporating these inequalities into Eq.12. We can further derive the upper bound of the backdoor learning task $\mathcal{L}_{\text{bdk}}^t$ in the downstream application context using the loss $\mathcal{L}_{\text{bdk}}^s$ of the backdoor learning task at the pretraining stage in Eq.13:

$$\mathcal{L}_{\text{bdk}}^{t} \leq \mathcal{L}_{\text{bdk}}^{s} + 2\sqrt{d(G_{s} \oplus \Delta_{G}, (G_{t,j} \oplus \Delta_{G}) \otimes G_{pro})} + C_{1}$$

$$d(G_{s} \oplus \Delta_{G}, (G_{t,j} \oplus \Delta_{G}) \otimes G_{pro})$$

$$= \sum_{i,j=1}^{n} \frac{s(\hat{h}(G_{s,i} \oplus \Delta_{G}), \hat{h}(G_{s,j} \oplus \Delta_{G}))}{n^{2}}$$

$$+ \sum_{i,j=1}^{m} \frac{s(\hat{h}((G_{t,i} \oplus \Delta_{G}) \otimes G_{pro}), \hat{h}((G_{t,j} \oplus \Delta_{G}) \otimes G_{pro}))}{m^{2}}$$

$$- \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(G_{s,i} \oplus \Delta_{G}), \hat{h}((G_{t,j} \oplus \Delta_{G}) \otimes G_{pro}))}{nm}$$

$$\leq 2\sqrt{k_{G}}(k_{G} - 1)L_{\hat{E}}\mu - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{s(\hat{h}(G_{s,i} \oplus \Delta_{G}), \hat{h}((G_{t,j} \oplus \Delta_{G}) \otimes G_{pro}))}{nm}$$

$$(13)$$

By combing Eq.6 and Eq.13 together, we can derive Eq.7.

Proof to Proposition 1. For the simplicity of analysis, we consider prompts as prompt graph patterns G_{pro} attached to an input graph G. We can then view the prompted graph $G \otimes G_{pro}$ as the result

of applying an isolated component transformation g_{ict} to the input graph G.

Proposition 3. **Proposition 5 in [8].** For a frozen graph encoder E, we assume each node of an input graph G has a vdim-dimensional feature vector. There hence exists a node-wise feature perturbation $\delta \text{feat}(V) \in R^{vdim}$ applied to the node features of G (noted as feat(V)), which satisfies:

$$E(G \otimes G_{pro}) = E(G')$$

$$G' = (\text{feat}(V) + \delta \text{feat}(V), A)$$
(14)

where A is the graph adjacency matrix of G.

Building on Proposition 3, we can deduce the analytical form of δ feat(V) by assuming the encoder E employs a GIN architecture. We follow the assumption originally established in the proof to Proposition 3, the GIN structure has a linear layer for node feature extraction and a sum readout function. Therefore, the node embedding produced by *E* can be given as:

$$H = (A + (1 + \epsilon)) \text{feat}(V)\theta \tag{15}$$

where θ denotes the parameters of the linear layer in the GINbased graph encoder. The graph embedding of *G* is then derived as SUM(H), by adding up the node embedding of each node $V \in G$. Hence, we can given the analytical expression of δ feat(V) as below, following Appendix A.3 in [8]:

$$\delta \mathrm{feat}(V)^{i} = \frac{\sum_{j=1}^{k_{pro}} (A_{pro} + (1+\epsilon)I) \mathrm{feat}(V_{pro})_{j}^{i}}{\mathrm{Deg} + N(1+\epsilon)} \tag{16}$$

where $\delta \text{feat}(V)^i$ is the *i*-th element of $\delta \text{feat}(V)$. The injected prompt graph has k_{pro} nodes in total. We assume the node feature vectors in G and the prompt graph G_{pro} share the same dimension. $feat(V_{pro})_{i}^{i}$ thus denotes the *i*-th node feature element of the node j of the prompt graph. A_{pro} is the adjacency matrix of the prompt graph G_{pro} . Deg and N are the total degree of all of the nodes and the number of graph nodes of *G*.

In summary, embeddings of a prompted graph $G_{t,i} \otimes G_{pro}$ of the downstream application context can be represented as below. Let $V(G_{t,i})$ are the nodes of $G_{t,i}$. $\delta \text{feat}(V(G_{t,i}))$ is the additional perturbation over the node features of $G_{t,i}$. feat (V_{pro}) denotes the node features of the prompt graph G_{pro} .

$$H^{I}(G_{t,i} \oplus G_{pro}) = (A_{t,i} + (1 + \epsilon))(\text{feat}(V(G_{t,i})) + \delta \text{feat}(V(G_{t,i})))\theta$$

$$\delta \text{feat}(V(G_{t,i})) = \left[\frac{\sum_{j=1}^{k_{pro}} (A_{pro} + (1 + \epsilon)I)\text{feat}(V_{pro})_{j}^{0}}{\text{Deg}_{t,i} + N_{t,i}(1 + \epsilon)}, \frac{\sum_{j=1}^{k_{pro}} (A_{pro} + (1 + \epsilon)I)\text{feat}(V_{pro})_{j}^{1}}{\text{Deg}_{t,i} + N_{t,i}(1 + \epsilon)}, \dots, \frac{\sum_{j=1}^{k_{pro}} (A_{pro} + (1 + \epsilon)I)\text{feat}(V_{pro})_{j}^{vdim}}{\text{Deg}_{t,i} + N_{t,i}(1 + \epsilon)}\right]$$

$$\frac{\sum_{j=1}^{k_{pro}} (A_{pro} + (1 + \epsilon)I)\text{feat}(V_{pro})_{j}^{vdim}}{\text{Deg}_{t,i} + N_{t,i}(1 + \epsilon)}$$

Similarly, for an input graph $G_{s,i}$ at the pretraining stage, let $V(G_{s,i})$ denote the set of nodes in $G_{s,i}$. Embeddings of $G_{s,i}$ produced by the encoder are given as:

$$H^{s}(G_{s,i}) = (A_{s,i} + (1+\epsilon))\operatorname{feat}(V(G_{s,i}))\theta \tag{18}$$

Assuming the prompt graph G_{pro} always has a fully connected structure, tuning the node features of the prompt graph feat(V_{pro}) to minimize $\sum\limits_{i=1}^n\sum\limits_{j=1}^m\frac{s(\hat{h}(G_{s,i}),\hat{h}(G_{t,j}\otimes G_{pro}))}{nm}$ in Eq.6 is equivalent to

solving the following least squared regression problem:

$$\begin{split} & \operatorname{feat}^*(V_{pro}) = \underset{\operatorname{feat}(V_{pro})}{\operatorname{arg \; min}} \sum_{i=1}^n \sum_{j=1}^m \|\operatorname{SUM}((A_{s,i} + (1+\epsilon))\operatorname{feat}(V(G_{s,i}))\theta) - \\ & \operatorname{SUM}((A_{t,j} + (1+\epsilon))(\operatorname{feat}(V(G_{t,j})) + \delta \operatorname{feat}(V(G_{t,j})))\theta)\|_2^2 \end{split}$$

where $\| \|_2$ is the L-2 norm. Therefore, for any set of the graphs $\{G_{s,i}\}\$ and $\{G_{t,i}\}\$ for pretraining and the downstream learning tasks, there exists an optimal feat* (V_{pro}) solving the optimization problem in Eq.19.

Proof of Proposition 2. Given a graph Δ_G as the trigger graph in *CrossBA*, an augmented graph Δ_G^+ is derived by adding / removing links from Δ_G . The backdoored graph $G \oplus \Delta_G$ is generated by linking the trigger graph Δ_G to an anchor node in an input graph

Proposition 4. Proposition 4 in [8]. We perform a link transformation over a graph G, i.e. adding / removing links from a graph G containing k_G nodes, to derive an augmented graph G^+ . For a frozen graph encoder E, we assume each node of an input graph G has a vdim-dimensional feature vector. There hence exists a node-wise feature perturbation $\delta \text{feat}(V) \in R^{vdim}$ applied to the node features of G (noted as feat(V)), which satisfies:

$$H(G^{+}) = (A + (1 + \epsilon)I)(\text{feat}(V(G)) + \delta \text{feat}(V(G^{+})))\theta$$

$$\delta \text{feat}(V)^{k} = \frac{\sum_{i,j=1}^{k_{G}} \delta A_{i,j} \text{feat}(V(G))_{j}^{k}}{Deg + (1 + \epsilon)N}$$
(20)

where A is the graph adjacency matrix of G. $\delta A_{i,j}$ are the difference between the graph adjacency of G and G^+ , i.e. $\delta A_{i,j} = A_{i,j} - A_{i,j}^+$ $feat(V)_{j}^{k}$ ($j = 1, 2, 3..., k_{G}$ and k = 1, 2, 3, ..., vdim) denotes the k dimension of the feature vector of node j.

Using Proposition 5 in [8], the node embedding of the backdoored graph $G_{s,i} \oplus \Delta_G$ can be expressed as:

$$\begin{split} H(G_{s,i} \oplus \Delta_G) &= (A_{s,i} + (1+\epsilon))(\operatorname{feat}(V(G_{s,i})) + \delta \operatorname{feat}(V(G_{s,i})))\theta \\ \delta \operatorname{feat}(V(G_{s,i})) &= \big[\frac{\sum_{j=1}^{k_{\operatorname{trigger}}} (A_{\operatorname{trigger}} + (1+\epsilon)I)\operatorname{feat}(V(\Delta_G))_j^0}{\operatorname{Deg}_{s,i} + N_{s,i}(1+\epsilon)}, \\ \frac{\sum_{j=1}^{k_{\operatorname{trigger}}} (A_{\operatorname{trigger}} + (1+\epsilon)I)\operatorname{feat}(V(\Delta_G))_j^1}{\operatorname{Deg}_{s,i} + N_{s,i}(1+\epsilon)}, \dots \\ \frac{\sum_{j=1}^{k_{\operatorname{trigger}}} (A_{\operatorname{trigger}} + (1+\epsilon)I)\operatorname{feat}(V(\Delta_G))_j^{\operatorname{vdim}}}{\operatorname{Deg}_{s,i} + N_{s,i}(1+\epsilon)} \big] \end{split}$$

where k_{trigger} , A_{trigger} and feat(Δ_G) are the number of the nodes, the adjacency matrix and the node feature matrix of the trigger graph Δ_G . Deg_{s,i} and $N_{s,i}$ are the total degree and the number of nodes in the graph $G_{s,i}$.

In summary, tuning feat $(V(\Delta_G))$, the node features of the trigger graph, to minimize $\frac{1}{nm}\sum\limits_{i=1}^n\sum\limits_{j=1}^m s(\hat{h}(\Delta_{G,i}^+),\hat{h}(G_{s,j}\oplus\Delta_G))$ can be formulated as a least squared regression problem in Eq.22. For any set of $\Delta_{G,i}^+$ and $G_{s,j}$, there exists an optimal feat $V(\Delta_G)$ solving the

Table 4: Statistics of datasets.

Datasets	Nodes	Edges	Features	Labels	Tasks
CiteSeer	3,327	9,104	3,703	6	N
Cora	2,708	5,429	1,433	7	N
Amazon-Computers	13,752	491,722	767	10	N
Amazon-Photo	7,650	491,722	767	8	N
ENZYMES	32.63(Avg.)	62.14(Avg.)	18	6	G

optimization problem in Eq.22.

$$feat^*(V(\Delta_G)) =$$

$$\mathop{\arg\min}_{\mathrm{feat}(V(\Delta_G))} \ \sum_{i=1}^n \sum_{j=1}^m \|\mathrm{SUM}((A+(1+\epsilon)I)(\mathrm{feat}(V(G))+\delta\mathrm{feat}(V(G_i^+)))\theta) - \|\mathrm{SUM}(A+(1+\epsilon)I)(\mathrm{feat}(V(G))+\delta\mathrm{feat}(V(G_i^+)))\theta)\| \leq C_i \|\mathrm{SUM}(A+(1+\epsilon)I)(\mathrm{feat}(V(G))+\delta\mathrm{feat}(V(G_i^+)))\theta\| + C_i \|\mathrm{SUM}(A+(1+\epsilon)I)(\mathrm{feat}(V(G))+\delta\mathrm{feat}(V(G)))\| + C_i \|\mathrm{SUM}(A+(1+\epsilon)I)(\mathrm{feat}(V(G))+\delta\mathrm{feat}(V(G))\| +$$

$$SUM((A_{s,i} + (1+\epsilon))(feat(V(G_{s,i})) + \delta feat(V(G_{s,i})))\theta)\|_2^2$$
(22)

C DATASETS AND GNN MODELS

CiteSeer [30] comprises 3,312 scientific publications categorized into 6 classes, connected by 4,732 citation links. Cora [30] consists of 2,708 scientific publications, each assigned to one of 7 categories within a citation network with 5,429 links. Amazon-Computers and Amazon-Photo [18] are subgraphs of the Amazon co-purchase graph, where nodes represent products, and edges indicate frequent co-purchases. Amazon-Computers has 13,752 nodes across 10 categories and 491,722 edges, while Amazon-Photo comprises 7,650 nodes from 8 categories and 238,162 edges. The ENZYMES dataset [1] contains 600 enzymes from the BRENDA enzyme database, classified into 6 EC enzyme categories. For graph classification datasets derived from node classification datasets, we follow the methodology proposed in [22], involving edge and subgraph sampling from the original data. Detailed statistics are presented in Table 4, where the last column indicates the type of downstream task for each dataset: "N" for node classification and "G" for graph classification.

GAT [25] and GT [20] are two advanced GNN models. GAT employs neighborhood aggregation for node embedding learning and distinguishes itself by assigning varied weights to neighboring nodes, thereby modifying their influence in the aggregation process. GT integrates the processing capabilities for graph-structured data with the self-attention mechanisms of Transformer networks, effectively capturing complex relationships and feature dependencies between nodes in a graph.

D BASELINE ATTACKS

GCBA [33] is the most relevant backdoor attack method to our study, given the threat model definition. Unlike *CrossBA*, which targets the GPL framework, GCBA aims to inject backdoor poisoning noise into a GNN encoder trained using Graph Contrastive Learning (GCL). The attacker in GCBA can collect the graph data of the target class in the downstream applications. The attack is then formulated to maximize the similarity between the embeddings of backdoored graph data and those of clean graph data belonging to the target class in the downstream task, as well as the similarity between embeddings of clean data from both the backdoored and clean GNN encoders. However, GCBA is not directly applicable to cross-context GPL scenarios as it relies on access to downstream

application data. To facilitate a fair comparison, we introduce two variants of GCBA, namely GCBA_R and GCBA_M. In both variants, the attacker first clusters the embeddings of clean graph data collected during pretraining using the backdoor-free GNN encoder. Then, GCBA_R randomly selects an embedding centered around a cluster in the embedding space as the target embedding, while GCBA_M selects the cluster center furthest away from other class centers as the target embedding. Subsequently, both variants follow the same workflow as GCBA to execute the attack.

Compared to *CrossBA*, GCBA differs in two significant aspects in its algorithmic design. Firstly, while *CrossBA* optimizes the target embedding during training of the backdoored GNN encoder (as shown in Eq. 2), GCBA uses a fixed target embedding derived from the downstream dataset's target class. Secondly, *CrossBA* includes a node feature affinity constraint in its attack objective, aiding in circumventing countermeasures that inspect node feature outliers, such as PruneG. However, GCBA permits arbitrary node features in the injected trigger graph. Consequently, *CrossBA* presents a more stealthy attack compared to GCBA, particularly against countermeasures deployed by downstream users.

E IMPLEMENTATION DETAILS

We implement CrossBA using PyTorch and execute it on an NVIDIA 3090 GPU. For both GAT and GT, we adopt a two-layer graph neural network structure with a hidden dimension set to 100. Following the methodology of ProG [22], we utilize Singular Value Decomposition (SVD) to reduce the initial feature dimension of the data to 100. The number of prompt nodes used in ProG and ProG-Meta is set to 15. The self-supervised learning method employed by ProG and ProG-Meta is GraphCL [32]. Consistent with the original paper's settings, we employ the Adam optimizer for ProG and ProG-Meta. On most evaluated datasets, the learning rate for GT is set to 0.001, and for GAT, it is set to 0.0001. Similarly, following the settings of GraphPrompt [13], we use the AdamW optimizer with a unified learning rate of 0.01 for all test instances. The self-supervised learning method used by GraphPrompt is the link prediction method [13]. Additionally, for downstream tasks, we employ 200-shot learning for ProG and ProG-Meta, and 100-shot learning for GraphPrompt.

In this study, we investigate five distinct cross-context scenarios. To address cross-distribution, cross-dataset, and cross-domain scenarios, we adopt the methodology of ProG [22], which leverages clustering methods to divide the entire graph into 200 distinct subgraphs for creating the pretraining dataset. For downstream tasks, we follow ProG's approach to construct induced graph datasets tailored for both node and graph classification tasks. In cross-class scenarios, we initially divide the label space into pretraining classes and downstream classes. Subsequently, we partition the entire graph into two disjoint subgraphs according to these classes, which are then used to form induced graph datasets for pretraining and downstream tasks, respectively. For the cross-task scenario, we utilize either the real graph classification dataset ENZYMES or the generated induced graph dataset for graph classification tasks as the pretraining dataset. The induced graph dataset for node classification tasks is employed as our downstream dataset.

Regarding the backdoor attack setting, for all the attack methods, the trigger graph is designed to consist of only three nodes,

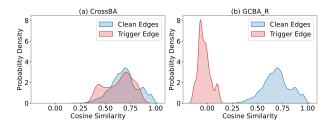


Figure 3: Kernel density estimation of node feature similarity on CiteSeer.

significantly fewer than the number of nodes in the input graph. During pretraining, the attacker randomly selects a node in the input graph as the anchor node. We set $\alpha=0.5$ for both the baseline attack methods and CrossBA. Additionally, for CrossBA, we set λ and β to 0.05 across the majority of datasets. The Adam optimizer is used for optimizing the trigger graph and the backdoored GNN encoder. Specifically, for CrossBA, we set γ_t to 0.01 and γ_g to 0.0001. For GCBA_R and GCBA_M, following the setting in [33], we set γ_t to 0.0015 and γ_g to 0.001.

F ATTACK RESULTS IN CROSS-DATASET AND CROSS-TASK SCENARIOS

Table 5 highlights the attack performance of various backdoor attacks in cross-dataset scenarios, where Cora is used as the pretraining dataset for CiteSeer and CiteSeer-Graph, while Computers serves as the pretraining dataset for Photo and Photo-Graph. The results underscore the superior attack capabilities of *CrossBA* across diverse downstream applications. *CrossBA* consistently achieves ASR values surpassing 0.83 in all tested cases, with a maximum decrease of only 0.06 in ACC compared to the clean model. In contrast, baseline methods encounter challenges in transferring backdoors to downstream models without significantly compromising the main task's performance.

In cross-task scenarios, the GNN encoder model is pretrained on ENZYMES, consisting of actual molecular graphs. The downstream node classification tasks involve CiteSeer and Cora. From the results in Table 6, we find that *CrossBA* effectively injects backdoors into downstream models pretrained on the graph classification task while maintaining high ACC values in node classification tasks. Conversely, the baseline methods often face challenges in transferring backdoors to downstream node classification tasks. For example, when targeting GraphPrompt, the baseline methods struggle to achieve an ASR higher than 0.41. In contrast, *CrossBA* consistently achieves ASR values surpassing 0.97.

G ATTACK AGAINST PRUNEG

We explore potential countermeasures against the *CrossBA* attack. Given the lack of specific defenses for cross-context GPL scenarios, we adapt defenses from other scenarios to mitigate the *CrossBA* attack. Real-world graphs, such as social networks, typically exhibit homophily, where nodes with similar features are connected by edges. However, trigger graph injection can disrupt this homophily. Therefore, a category of defense methods exists that improves GNN robustness by pruning edges connecting nodes with low feature

similarity [26]. We extend the *PruneG* defense to align with our threat model, where the downstream user acts as the defender. Given a graph, the defender calculates the similarity of node features connected by edges and prunes those edges with similarity below the threshold, removing the component with fewer nodes that the edge connects.

Figure 2 shows the PruneG defense's effectiveness against backdoor attacks in both node and graph classification tasks across 5 cross-context scenarios and 2 GPL methods. Cora is employed for pretraining in cross-dataset scenarios, Photo serves as the pretraining dataset in cross-domain scenarios, and the induced graph dataset for the graph classification task is used as the pretraining dataset in cross-task scenarios. All the baseline methods exhibit poor attack performance facing the *PruneG* defense. In contrast, CrossBA successfully evades detection, achieving high ASR values. For example, in cross-domain scenarios, CrossBA attains ASR values above 0.90 on all the tested cases, while the baselines' ASR values remain below 0.50. In CrossBA, the optimization of trigger node features is constrained to closely resemble those of the clean anchor nodes. Figure 3 illustrates the kernel density of node feature similarity values, highlighting that trigger nodes created by the baseline attacks exhibit minimal similarity to anchor nodes' features, making them more easily detected. Conversely, CrossBA achieves indistinguishable similarity for trigger edges compared to clean edges, confirming its stealthiness.

H ABLATION STUDY

In our ablation studies on CrossBA to assess the significance of its components, we consider three variants: (1) CrossBA without trigger optimization, employing a fixed trigger graph and target embedding; (2) CrossBA without embedding alignment ($\alpha=0$); (3) CrossBA without node feature affinity ($\beta=0$). Figure 4 illustrates the attack performance of CrossBA and its variants against two GPL methods with the Prune defense, encompassing five cross-context scenarios. Cora is used for pretraining in cross-dataset scenarios, Photo serves as the pretraining dataset in cross-domain settings, and the induced graph dataset for the graph classification task is employed in cross-task scenarios.

Comparing *CrossBA* to its three variants, we observe that *CrossBA* consistently maintains stable attack performance across various cross-context GPL scenarios, even in the presence of defense mechanisms. Each component of *CrossBA* is crucial for achieving high ASR values. For instance, in cross-task scenarios involving Graphprompt, the removal of any component significantly lowers ASR values to below 0.20. Conversely, *CrossBA* attains an ASR of 0.90, underscoring the significance of each component to the attack's success.

I IMPACT OF PROMPT TOKENS.

Figure 5 illustrates the impact of the number of prompt tokens on the attack performance of *CrossBA* against ProG in both node and graph classification tasks on CiteSeer across five cross-context scenarios. In cross-dataset scenarios, Cora is employed for pretraining, while Photo serves as the pretraining dataset for cross-domain settings. In cross-task scenarios, the generated induced graph dataset for the graph classification task is utilized for pretraining.

Table 5: ACC, ASR, and AD in cross-dataset scenarios. Cora is used as the pretraining dataset for CiteSeer and CiteSeer-Graph, and Computers is used for Photo and Photo-Graph.

		Attack	No	ode Cla	ssification	Gra				
GPL	Model		CiteSee	r	Photo		CiteSeer-G	raph	Photo-Graph	
	11104101		ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR	ACC(AD)	ASR
		GCBA_R	0.24(+0.57)	0.48	0.44(+0.34)	0.59	0.17(+0.61)	0.00	0.59(+0.30)	0.28
	GAT	GCBA_M	0.19(+0.62)	0.00	0.46(+0.32)	0.28	0.17(+0.61)	0.06	0.44(+0.45)	0.68
ProG		CrossBA	0.83(-0.02)	1.00	0.78(-0.00)	1.00	0.79(-0.01)	1.00	0.89(-0.00)	0.97
1100		GCBA_R	0.48(+0.34)	0.80	0.55(+0.26)	0.95	0.44(+0.35)	0.42	0.67(+0.27)	0.99
	GT	GCBA_M	0.48(+0.34)	0.90	0.52(+0.29)	0.16	0.44(+0.35)	0.75	0.58(+0.36)	0.62
		CrossBA	0.82(-0.00)	1.00	0.81(-0.00)	1.00	0.79(-0.00)	1.00	0.93(+0.01)	1.00
		GCBA_R	0.72(-0.00)	0.09	0.60(+0.12)	0.12	0.67(+0.06)	0.10	0.71(+0.10)	0.10
	GAT	GCBA_M	0.59(+0.13)	0.12	0.64(+0.08)	0.03	0.64(+0.09)	0.03	0.65(+0.16)	0.16
Graph		CrossBA	0.67(+0.05)	0.97	0.70(+0.02)	1.00	0.67(+0.06)	0.93	0.82(-0.01)	1.00
Prompt	-	GCBA_R	0.66(+0.13)	0.06	0.59(+0.13)	0.13	0.59(+0.18)	0.04	0.68(+0.16)	0.09
	GT	GCBA_M	0.55(+0.24)	0.01	0.58(+0.14)	0.12	0.66(+0.11)	0.09	0.66(+0.18)	0.32
		CrossBA	0.80(-0.01)	1.00	0.71(+0.01)	0.99	0.77(-0.00)	1.00	0.85(+0.01)	0.93
		GCBA_R	0.50(+0.17)	0.00	0.83(+0.14)	1.00	0.50(+0.29)	0.00	1.00(-0.00)	0.96
	GAT	GCBA_M	0.50(+0.17)	0.00	0.50(+0.47)	0.00	0.50(+0.29)	0.00	0.75(+0.25)	1.00
ProG		CrossBA	0.92(-0.25)	1.00	0.97(-0.00)	1.00	0.89(-0.10)	1.00	1.00(-0.00)	1.00
Meta		GCBA_R	0.94(-0.00)	1.00	0.89(+0.10)	0.91	0.50(+0.44)	0.00	1.00(-0.00)	0.01
	GT	GCBA_M	0.71(+0.23)	0.29	0.55(+0.44)	0.00	0.68(+0.26)	1.00	0.95(+0.05)	0.96
		CrossBA	0.94(-0.00)	1.00	0.98(+0.01)	1.00	0.92(+0.02)	1.00	1.00(-0.00)	1.00

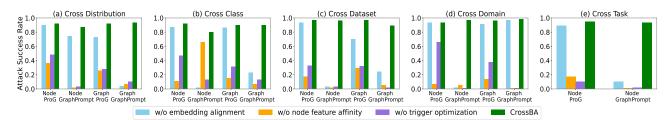


Figure 4: Ablation study of CrossBA against 2 GPL methods with Prune on CiteSeer across 5 cross-context scenarios. "Node" represents the node classification task, and "Graph" denotes the graph classification task.

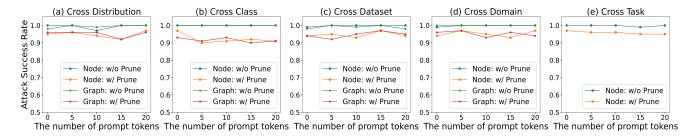


Figure 5: Impact of the number of prompt tokens on attack performance of CrossBA against ProG based on CiteSeer in 5 cross-context scenarios. "Node" represents the node classification task, and "Graph" denotes the graph classification task.

The results reveal that the effectiveness of *CrossBA* remains robust, exhibiting little impact from variations in the number of prompt tokens. Across all five cross-context scenarios, *CrossBA* consistently maintains an ASR above 0.90, even with an increased number of prompt tokens. Importantly, this stability is observed regardless of the presence of defense mechanisms, emphasizing the resilience of *CrossBA* to variations in prompt token numbers.

I IMPACT OF TRIGGER NODES.

Figure 6 illustrates the impact of varying the number of trigger nodes on the attack performance of *CrossBA* against ProG and GraphPrompt in both node and graph classification tasks on Cite-Seer across five cross-context scenarios. In cross-dataset scenarios, the pretraining is conducted using Cora, while Photo serves as the pretraining dataset for cross-domain settings. In cross-task scenarios, the generated induced graph dataset for the graph classification

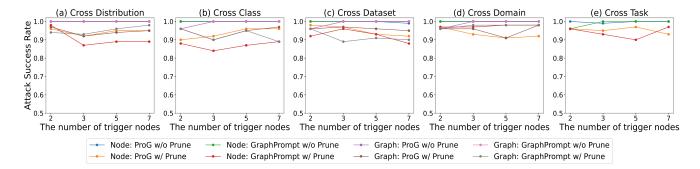


Figure 6: Impact of the number of trigger nodes on attack performance of CrossBA against ProG and GraphPrompt based on CiteSeer in 5 cross-context scenarios. "Node" represents the node classification task, and "Graph" denotes the graph classification task

Table 6: ACC, ASR, and AD in cross-task scenarios. ENZYMES is the pretraining dataset.

GPL	Model	Attack	CiteSee	er	Cora		
012	11104101	1111111111	ACC(AD)	ASR	ACC(AD)	ASR	
		GCBA_R	0.50(+0.26)	0.23	0.24(+0.42)	0.00	
	GAT	GCBA_M	0.61(+0.15)	0.10	0.24(+0.42)	0.00	
ProG		CrossBA	0.75(+0.01)	1.00	0.72(-0.06)	1.00	
1100		GCBA_R	0.55(+0.15)	0.85	0.44(+0.20)	0.80	
	GT	GCBA_M	0.57(+0.13)	0.77	0.44(+0.20)	0.93	
		CrossBA	0.82(-0.12)	0.87	0.64(-0.00)	1.00	
		GCBA R	0.70(+0.04)	0.03	0.26(+0.38)	0.41	
	GAT	GCBA_M	0.71(+0.03)	0.00	0.34(+0.30)	0.13	
Graph		CrossBA	0.75(-0.01)	1.00	0.65(-0.01)	0.97	
Prompt		GCBA_R	0.69(+0.06)	0.05	0.56(-0.03)	0.24	
	GT	GCBA_M	0.77(-0.02)	0.04	0.50(+0.03)	0.26	
		CrossBA	0.75(-0.00)	1.00	0.10 0.24(+0.42) 0.00 0.72(-0.06) 0.72(-0	1.00	
		GCBA R	0.50(+0.14)	0.00	0.50(+0.38)	0.00	
	GAT	GCBA M	0.76(-0.12)	1.00	0.50(+0.38)	0.00	
ProG		CrossBA	0.90(-0.26)	1.00	0.88(-0.00)	0.87	
Meta		GCBA_R	0.91(+0.03)	0.61	0.50(+0.41)	0.00	
	GT	GCBA_M	0.89(+0.05)	0.44	0.51(+0.40)	1.00	
		CrossBA	0.93(+0.01)	1.00	0.91(-0.00)	1.00	

task is utilized for pretraining. The results highlight that *CrossBA* consistently maintains stable attack performance, achieving an ASR above 0.80 across all the test cases, regardless of the specific number of trigger nodes employed.