DShield: Defending against Backdoor Attacks on Graph Neural Networks via Discrepancy Learning

Hao Yu*, Chuan Ma†✉, Xinhang Wan*, Jun Wang*, Tao Xiang†, Meng Shen‡, Xinwang Liu*✉
*National University of Defense Technology, †Chongqing University, ‡Beijing Institute of Technology
*{yu_haocs, wanxinhang, wang_jun, xinwangliu}@nudt.edu.cn, †{chuan.ma, txiang}@cqu.edu.cn, ‡shenmeng@bit.edu.cn

Abstract*—Graph Neural Networks (GNNs) are vulnerable to backdoor attacks, where triggers inserted into original graphs cause adversary-determined predictions. Backdoor attacks on GNNs, typically focusing on node classification tasks, are categorized by dirty- and clean-label attacks and pose challenges due to the interconnected nature of normal and poisoned nodes. Current defenses are indeed circumvented by sophisticated triggers and often rely on strong assumptions borrowed from other domains (e.g., rapid loss drops on poisoned images). They lead to high attack risks, failing to effectively protect against both dirty- and clean-label attacks simultaneously. To tackle these challenges, we propose DShield, a comprehensive defense framework with a discrepancy learning mechanism to defend against various graph backdoor attacks. Specifically, we reveal two vital facts during the attacking process: *semantic drift* where dirty-label attacks modify the semantic information of poisoned nodes, and *attribute over-emphasis* where clean-label attacks exaggerate specific attributes to enforce adversary-determined predictions. Motivated by those, DShield employs a self-supervised learning framework to construct a model without relying on manipulated label information. Subsequently, it utilizes both the self-supervised and backdoored models to analyze discrepancies in semantic information and attribute importance, effectively filtering out poisoned nodes. Finally, DShield trains normal models using the preserved nodes, thereby minimizing the impact of poisoned nodes. Compared with 6 state-of-the-art defenses under 21 backdoor attacks, we conduct evaluations on 7 datasets with 2 victim models to demonstrate that DShield effectively mitigates backdoor threats with minimal degradation in performance on normal nodes. For instance, on the Cora dataset, DShield reduces the attack success rate to 1.33% from 54.47% achieved by the second-best defense Prune while maintaining an 82.15% performance on normal nodes. The source code is available at https://github.com/csyuhao/DShield.

I. Introduction

Graph-structured data, such as social media networks [25] and mobile payment networks [24], are ubiquitous. Within these intricate networks, nodes serve as representations of identities and are interconnected. Graph Neural Networks (GNNs) leverage a message-passing mechanism to update node representations by aggregating information from neighboring nodes. This capability has proven instrumental in various tasks, such as node classification [3] and graph classification [19].

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240798
www.ndss-symposium.org

Despite the significant achievements of GNNs, recent studies have revealed their susceptibility to backdoor attacks [38], [42]. These attacks aim to induce undesirable behaviors in backdoored models by injecting *triggers* into original graphs. Triggers, varying from single nodes to subgraphs, allow the compromised GNN to behave normally with unaltered graphs but misbehave when presented with manipulated ones. Current backdoor attacks on GNNs are divided into graph and node classification attacks. The former misleads the model to misclassify manipulated graphs, similar to image classification attacks, and is defended by image domain defenses. In contrast, the latter misclassifies nodes with triggers, presenting a greater challenge due to the interconnection of normal and poisoned nodes. In node classification attacks, this malicious behavior is achieved by manipulating the training graph with a small subset of poisoned nodes [5], [43]. These nodes are assigned target labels, distinguishing between dirty- and clean-label attacks based on whether their ground-truth labels match the target labels. Specifically, in dirty-label attacks, the ground-truth labels of poisoned nodes differ from the target labels, and vice versa. Consequently, node classification attacks pose a significant threat, and our primary focus is addressing them.

Deficiencies of Existing Defenses. Current defense mechanisms can be broadly categorized into three types: *preprocess-based defenses* [5], *detection-based defenses* [8], [20], and *poison-suppression-based defenses* [29]. 1) Preprocess-based defenses introduce a preprocessing operation to disrupt trigger patterns, thereby preventing backdoor activation. 2) Detection-based defenses, inspired by assumptions and observations in other domains such as rapid loss drop on poisoned image data, rely on scanning model predictions to detect the presence of a backdoor in the graph. 3) Poison-suppression-based defenses aim to diminish the impact of poisoned nodes, thereby preventing the creation of backdoored models. However, these defenses can be circumvented by custom trigger generation constraints, rendering comprehensive trigger detection challenging and potentially compromising performance on normal nodes. Furthermore, node classification attacks encompass both dirty- and clean-label varieties (as discussed in Section IV), and existing defenses cannot simultaneously mitigate both types. Thus, effectively defending against node classification attacks remains an ongoing and unresolved issue.

Our Goals and Contributions. To address these challenges, we first observe *semantic drift* and *attribute over-emphasis* facts of poisoned nodes, commonly exist in backdoor attacks on node classification tasks. Motivated by these facts, we

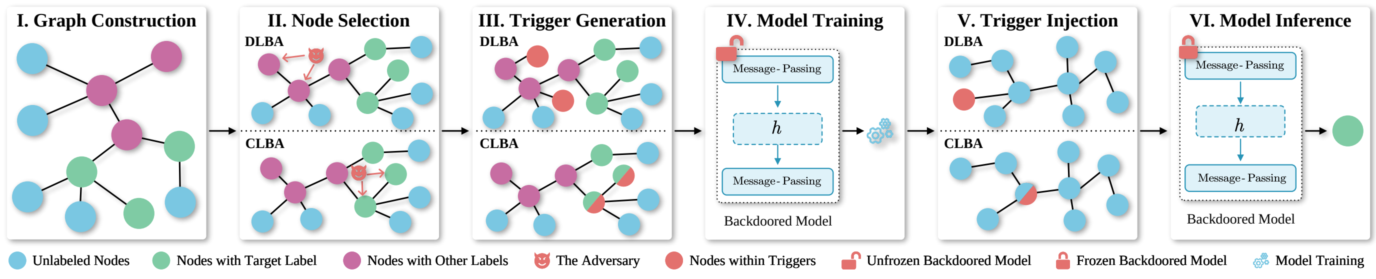


Fig. 1: Illustration of the backdoor attack procedure. For brevity, a single node represents the trigger g, while different attacks utilize various forms of triggers, such as single nodes or subgraphs. Triggers are injected into the original graph via attachment or modification of attributes or structures of existing nodes.

propose a comprehensive defense, DShield, to leverage the discrepancy learning mechanism to analyze the divergences between poisoned and normal nodes. It employs self-supervised and backdoored models to obtain semantic information and attribute importance of nodes, identifies poisoned nodes via a clustering algorithm, and utilizes a backdoor-free training mechanism to eliminate their influence. The defense consists of three key components: 1) an *auxiliary model training* module that creates a self-supervised model without manipulated labels and a backdoored model with manipulated labels to analyze semantic information and attribute importance of each node, 2) a *discrepancy matrix construction* module that builds the discrepancy matrix to assess the relation among nodes with the same label and identify poisoned nodes, and 3) a *backdoor-free model training* module that discovers the target label and obtains the normal model by removing identified poisoned nodes. The contributions are summarized as follows:

- We observe semantic drift and attribute over-emphasis among poisoned nodes in node classification backdoor attacks and thereby devise a learning paradigm to obtain divergences in semantic information and attribute importance between poisoned and normal nodes.
- We propose DShield that employs self-supervised learning and attribute importance analysis to learn two discrepancy matrices for analyzing semantic drift and attribute over-emphasis among nodes and utilizes a backdoor-free training mechanism to develop a normal model.
- We conduct evaluations across 4 datasets to show that DShield outperforms state-of-the-art baselines in defending against various node classification backdoor attacks. Specifically, DShield significantly reduces attack success rates to below 10% on most attack cases.
- We evaluate DShield's performance against 5 adaptive attacks where the adversary employs multi-target attacks, e.g., mixtures of dirty- and clean-label attacks, and is aware of DShield's inner mechanisms. Experimental results indicate that adaptive attacks cannot circumvent DShield.
- We further show the scalability of DShield by adapting it to mitigate graph classification backdoor attacks on 3 datasets. Our findings show semantic drift and attribute over-emphasis also rooted in graph classification attacks, highlighting DShield's ability to counter such attacks.

II. Background and Related Work

A. Graph Neural Networks

We consider an attribute graph denoted as $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$, where the node set is $\mathcal{V} = \{v_1, \cdots, v_N\}$, and the structure is expressed as $\mathcal{E} = \{(v_1, v_i), \cdots, (v_j, v_N)\}$. The matrices $\boldsymbol{A}$ and $\boldsymbol{X}$ represent the adjacency and attribute matrices, respectively. The binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N\times N}$ defines node connections, with $A_{ij} = 1$ indicating a connection between nodes v_i and v_j, and $A_{ij} = 0$ denoting no connection. The attribute matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ contains attributes for each node, where $\boldsymbol{x}_i$ represents the attributes of node v_i. In this paper, we focus on a semi-supervised node classification task in the inductive setting [1], [22], commonly encountered in practical scenarios. For instance, GNNs trained on social networks often need to make predictions for newly registered users. In this scenario, the entire graph $\mathcal{G}$ is partitioned into the training graph $\mathcal{G}^{\text{train}}$ and the testing graph $\mathcal{G}^{\text{test}}$, with sets of nodes within $\mathcal{G}^{\text{train}}$ and $\mathcal{G}^{\text{test}}$ denoted as $\mathcal{V}^{\text{train}}$ and $\mathcal{V}^{\text{test}}$, respectively. A subset of nodes $\mathcal{V}^l \subset \mathcal{V}^{\text{train}}$ is labeled with $\boldsymbol{Y}^l \in \{0,1\}^{N\times C}$, where C represents the total number of labels. The test nodes $\mathcal{V}^{\text{test}}$ are not included in the training graph $\mathcal{G}^{\text{train}}$, i.e., $\mathcal{V}^{\text{train}} \cap \mathcal{V}^{\text{test}} = \emptyset$.

GNNs have emerged as the dominant approach for modeling graph data. Typically, the model f with parameters θ applied to a node classification task takes a graph $\mathcal{G}$ as input and employs the message-passing mechanism [36] to learn representations of each node. Specifically, the message passing towards node v at layer l is defined as follows:

$$\boldsymbol{h}_i^{l+1} = \sigma(\rho(\boldsymbol{h}_i^l, \{\boldsymbol{h}_j^l, \forall\, v_j \in \mathcal{N}(v_i)\})\boldsymbol{W}^l), \tag{1}$$

where $\boldsymbol{h}_i^l$ and $\boldsymbol{W}^l$ represent the intermediate representations of node v_i and the trainable parameter at the l-th layer, respectively. $\sigma(\cdot)$ denotes the activation function, and $\rho(\cdot)$ is the aggregation function to collect information from neighbors $\mathcal{N}(v_i)$. The training loss is calculated as follows:

$$\mathcal{L}(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{|\mathcal{V}^l|} \sum_{v_i \in \mathcal{V}^l} \text{CE}(f(\boldsymbol{A}, \boldsymbol{X})_i, \boldsymbol{y}_i), \tag{2}$$

where $|\cdot|$ represents the cardinality of a set, $\text{CE}(\cdot,\cdot)$ is the cross-entropy loss, and $\boldsymbol{y}_i$ denotes the one-hot label vector corresponding to node v_i within the label matrix.

B. Backdoor Attacks on Node Classification Tasks

Backdoor attacks constitute an emerging area of research raising security concerns regarding training with third-party resources. Existing attacks generally comprise six steps: 1) graph construction, 2) node selection, 3) trigger generation, 4) model training, 5) trigger injection, and 6) model inference, as depicted in Fig. 1. Further details about these six steps are provided in Appendix A-A. Following existing studies [40], [48], they can be divided into two types:

Dirty-Label Backdoor Attacks (DLBAs). The majority of existing attacks belong to this category. During trigger generation, the adversary injects a trigger g into subgraphs centered around a set of poisoned nodes $\mathcal{V}^p \subset \mathcal{V}^{\text{train}}$ and assigns $\mathcal{V}^p$ the target label y^t, crafting a manipulated graph, described as:

$$\widetilde{\mathcal{G}} = (\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{Y}}) = \mathcal{M}(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}; g, y^t, \mathcal{V}^p), \tag{3}$$

where $\mathcal{M}$ represents the method of injecting triggers into the original graph. The poisoning rate r_p is calculated via $r_p = {|\mathcal{V}^p|}/{|\mathcal{V}^t|}$. GNNs trained on the manipulated graph are optimized to predict the poisoned nodes $\mathcal{V}^p$ as the target label y^t, linking the trigger g with the target label. In the trigger injection step, the adversary can inject the trigger g into the subgraph centered around a test node v to classify v as the target label by the backdoored GNN.

Initial efforts have been made for dirty-label graph backdoor attacks. SBA [50] formulates subgraphs as the trigger g, and GTA [38] employs a trigger generator to acquire optimal poisoned-node-specific triggers. EBA [43] applies GraphLIME [12], a GNN explainability approach, to select the optimal trigger-attaching position and modify a subset of node features as triggers. TRAP [44] utilizes a surrogate model to construct poisoned-node-specific and pattern-flexible triggers. GB-FGSM [4] and LGCB [4] designate the trigger as a single node, activating the backdoor when the trigger node is connected to the poisoned node. UGBA [5] designs an adaptive trigger generator to obtain inconspicuous triggers.

Clean-Label Backdoor Attacks (CLBAs). Recent research focuses on clean-label attacks to improve the stealth of backdoor attacks, where ground-truth labels of poisoned nodes and target labels are consistent. The adversary samples a subset of training nodes $\mathcal{V}^p$ with the target label y^t and alters attributes of $\mathcal{V}^p$ as the trigger g without changing their structures and labels, leading to a manipulated graph, expressed as:

$$\widetilde{\mathcal{G}} = (\boldsymbol{A}, \widetilde{\boldsymbol{X}}, \boldsymbol{Y}) = \mathcal{M}(\boldsymbol{X}; g, y^t, \mathcal{V}^p). \tag{4}$$

In the model inference step, the adversary's goal is to cause the backdoored model to misclassify the node v, whose ground-truth label is not the target label y^t, as y^t, by attaching g.

Only two clean-label backdoor attacks exist for node classification tasks. GCBA [49] adopts the poisoned node as the trigger g and directly alters the attributes of poisoned nodes to minimize the difference between embeddings of poisoned nodes and embeddings of nodes with the target label, manipulating the behavior of the downstream classifier built on the pre-trained GNN encoder. Additionally, Yang *et al.* [45] proposed PerCBA, devising a generator to generate different perturbed attributes for the target label.

C. Defense against Node Classification Attacks

Defenses against attacks can be broadly categorized into three groups: *preprocess-based defenses*, *detection-based defenses*, and *poison-suppression-based defenses*.

Preprocess-Based Defenses. These defenses aim to disrupt trigger patterns in manipulated graphs to prevent backdoor activation. Motivated by real-world graphs, such as social networks, exhibiting the homophily property where nodes with similar attributes are connected, one such approach, introduced by Dai *et al.* [5], employs a preprocessing operation called *prune* to remove edges linking nodes with low cosine similarities. However, they can be evaded by enhancing metrics applied in the preprocessing operation. For instance, UGBA [5] introduces an unnoticeable constraint to ensure that nodes within triggers are similar to poisoned nodes.

Detection-Based Defenses. These defenses aim to identify trigger presence in manipulated graphs through analysis of model predictions. Motivated by the tendency of backdoored GNNs to rely on simple sub-graphs for poisoned nodes and more complex structures for normal nodes, Guan *et al.* [8] proposed explanation-guided backdoor detection methods, such as XGBD-PGExplainer and XGBD-GNNExplainer, leveraging topological information to attribute model predictions to crucial subgraphs. If the model's loss value on an explanatory subgraph of a node falls below a threshold, indicating that the explanatory subgraph effectively preserves core information for node classification, it is deemed poisoned. Additionally, Li *et al.* [20] observed that in image domains, the loss on poisoned images decreases notably faster compared to normal images during training with a mixture of both. However, assumptions or phenomena underlying these defenses may not universally apply, as GNNs rely on smaller subgraphs for decision-making on normal nodes as well, and graphs differ from images where individual nodes are interconnected through edges. Excessive removal of normal nodes could degrade model performance on such nodes. Furthermore, the effectiveness of explanation-guided defenses is influenced by explanation methods, and generating explanatory subgraphs for each node individually remains time-intensive.

Poison-Suppression-Based Defenses. These defenses aim to diminish the impact of poisoned nodes during training to prevent the creation of backdoored models. For instance, Zhang *et al.* [49] utilized *random smoothing* to randomly subsample subgraphs for GNN training, thereby corrupting potential triggers within the graph. The subsampling procedure, controlled by a subsampling ratio, involves removing a portion of nodes and masking the features of the remaining nodes. Repeating this process generates a set of subgraphs used for GNN training, resulting in a smoother GNN. Nonetheless, these defenses may lead to reduced performance on normal nodes and cannot completely eliminate the influence of triggers. Additionally, Zhang *et al.* [49] demonstrated that "random smoothing" is ineffective against GCBA.

Unlike previous defenses, our DShield makes no assumptions about trigger specifications like size and is designed based on intrinsic characteristics observed in backdoor attacks, providing a comprehensive defense against both dirty- and clean-label attacks, as well as graph classification attacks.

III. Threat Model and Design Goals

A. Threat Model

We consider a scenario where the defender leverages third-party data to build a backdoor-free model denoted as f [48]. In this context, an evil actor, referred to as the backdoor adversary, can manipulate a subset of nodes by injecting triggers into the graph.

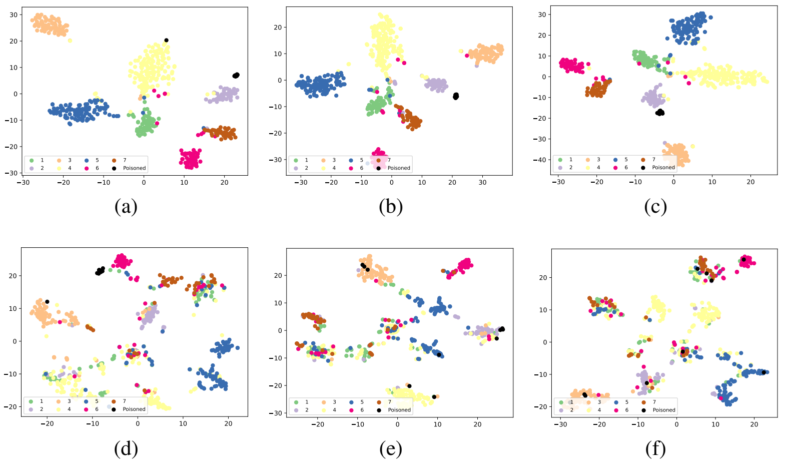


Fig. 2: Semantic Drift: t-SNE visualization of latent representations of nodes within the manipulated graph generated by distinct backdoor attacks on the **Cora** dataset. Fig. (a)-(c): GNNs trained with the manipulated label information using the semi-supervised learning paradigm. Fig. (d)-(f): GNNs trained with the manipulated label information using self-supervised learning paradigm. Fig. (a)-(c): Manipulated graph generated by GTA [38], GB-FGSM [4] and UGBA [5]. This fact indicates that the semantic information of poisoned nodes differs from that of normal nodes with the same label.

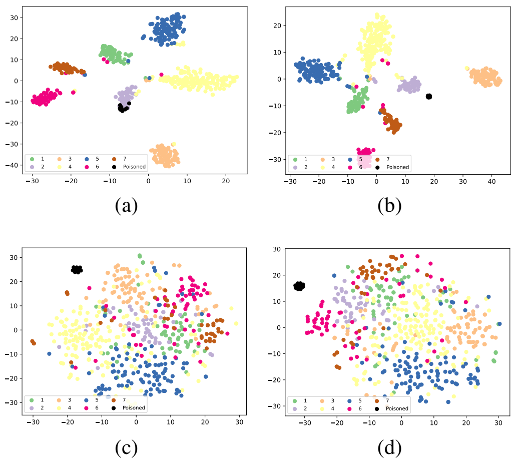


Fig. 3: Attribute Over-emphasis: t-SNE visualization of attribute importance within the manipulated graph on the **Cora** dataset. Fig. (a)-(b): GNNs trained using the semi-supervised learning paradigm. Fig. (c)-(d): Visualization of attribute importance. Fig. (a)-(b): Manipulated graph generated by GCBA [49] and PerCBA [45]. This fact denotes that backdoored models over-emphasize certain attributes of poisoned nodes.

Adversary's Knowledge and Abilities. Based on assumptions from prior studies on backdoor attacks [5], [44], [49], [33], it is assumed that the defender utilizes third-party data to construct a GNN model. Naturally, the adversary: 1) possesses knowledge of the graph, including poisoned nodes, partial topology, and class count, 2) lacks knowledge of the defender's model architecture and parameters, 3) typically does not know defenses, though we also explore a scenario where the adversary is aware of DShield's mechanism and employs adaptive and multi-target attacks, e.g., mixtures of dirty- and clean-label attacks, in the Subsection VI-C.

We presume the adversary is proficient in three aspects: 1) Given the absence of direct access to the backdoored model, they utilize a surrogate model in trigger construction, acknowledging potential disparities in architecture and parameters. 2) Triggers may manifest as nodes or subgraphs, with diverse methods $\mathcal{M}$ employed for injecting triggers into the graph. 3) Despite being unaware of defense mechanisms, the adversary considers various defenses during trigger construction.

Adversary's Goals. Broadly, the adversary's objectives encompass *efficiency* and *stealthiness*. Regarding *efficiency*, the adversary aims to enhance the performance of the backdoored model $\widetilde{f}(\cdot;\phi)$, parameterized by ϕ, on poisoned nodes:

$$\arg\min_{\phi} \frac{1}{|\mathcal{V}^p|} \sum_{v_i \in \mathcal{V}^p} \mathrm{CE}(\widetilde{f}(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}; \phi)_i, \boldsymbol{y}^t), \tag{5}$$

where $\boldsymbol{y}^t$ is the one-hot vector with the y^t-th index set to 1. Simultaneously, for *stealthiness*, the adversary aims for the backdoored model to exhibit normal behavior for non-poisoned nodes, expressed as:

$$\widetilde{f}(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}; \phi)_i = \begin{cases} y^t, & v_i \in \mathcal{V}^p; \\ f(\boldsymbol{A}, \boldsymbol{X}; \theta)_i, & v_i \notin \mathcal{V}^p. \end{cases} \tag{6}$$

Defender's Knowledge and Abilities. Concerning the untrustworthy graph dataset from the third party, the defender remains unaware of trigger presence. Even if triggers are injected, the defender lacks information about target labels, the specific backdoor attack employed, or trigger specifications like size and poisoning rate r_p.

Naturally, the defender has complete access to the manipulated graph and can utilize diverse analysis algorithms to identify injected triggers. For instance, the defender may train their models or detectors over the graph $\widetilde{\mathcal{G}}$. Unlike Jiang *et al.* [16], we posit a practical scenario where defenders lack validation datasets for identifying poisoned nodes, considering the resource-intensive nature of creating such datasets [18].

Our Design Goals of Defense. Subsequently, aligned with adversary objectives, we establish *effectiveness* and *robustness* as core design goals for DShield. For *effectiveness*, regardless of backdoor attacks, the model deployed with DShield should perform comparably to a model trained exclusively on normal nodes. Moreover, despite the attack's efficiency, the defended backdoored model should minimize performance degradation on poisoned nodes. Regarding *robustness*, the defense must effectively mitigate various attacks, including DLBAs and CLBAs, and also protect against backdoor attacks in graph classification tasks without making any assumptions about trigger specifications.

IV. Motivation

This section critically examines the current design of node classification backdoor attacks and observes two facts among poisoned ones. In this section, we set the target label $y^t = 2$.

Semantic Drift of DLBAs. DLBAs involve injecting triggers into the subgraph centered around poisoned nodes, thereby

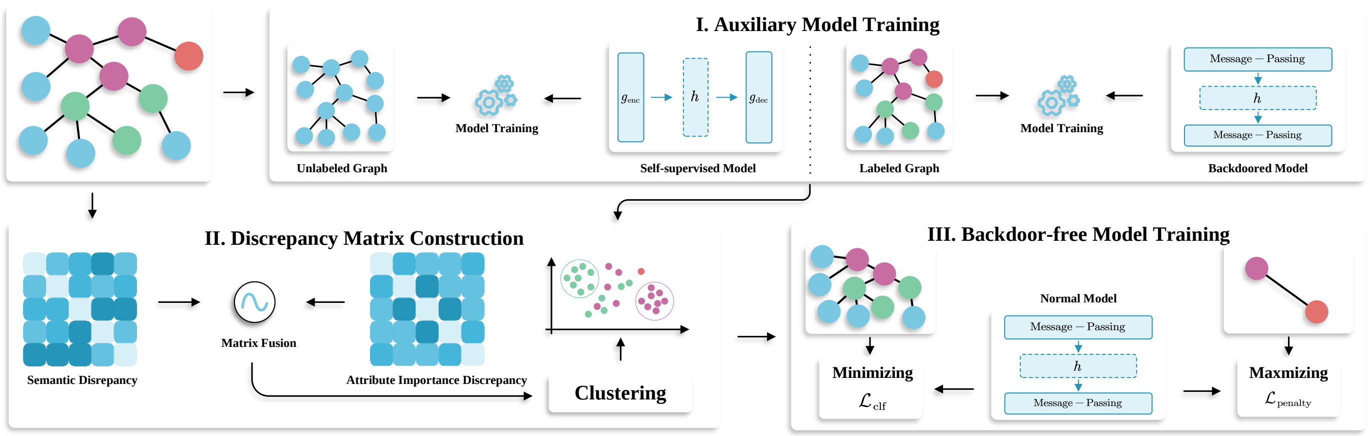


Fig. 4: Illustration of DShield, consisting of the "auxiliary model training", "discrepancy matrix construction", and "backdoor-free model training" modules for the construction of a backdoor-free model on the manipulated graph.

altering the labels of these nodes to a predefined target label. Models trained on such manipulated graphs establish a strong correlation between the trigger and the target label, resulting in predictions associating nodes connected to the trigger with the target label during model inference. Consequently, the semantic information of poisoned nodes remains different from that of normal nodes labeled y^t, and DLBAs forge a strong association between triggers and the target label, effectively embedding backdoors in GNNs.

To verify this, we conduct three DLBAs, i.e., GTA [38], GB-FGSM [4], and UGBA [5], on the Cora [32] dataset. Specifically, we execute semi-supervised learning on the manipulated graph with the manipulated label information and self-supervised learning on the manipulated graph without the label information by leveraging GraphCL [46]. We visualize the poisoned nodes in the latent representation space generated by these two learned GNNs using t-SNE [35].

From the experimental results (Fig. 2), we observe that poisoned nodes tend to cluster together after the standard semi-supervised training process, which is close to normal nodes with the target label y^t. This observation implies why existing DLBAs can succeed, aligning with findings in the image domain [11]. Associated with the end-to-end semi-supervised training paradigm, GNNs can shrink the distance between poisoned nodes in the latent representation space and connect the learned trigger-related features with the target label. In contrast, as shown in Fig. (2d)-(2f), poisoned nodes lie far from the normal nodes with the target label after the self-supervised learning process on the unlabeled poisoned graph. It indicates that the semantic information of poisoned nodes remains different following the trigger injection. This fact is called *semantic drift* between the semantic information of nodes' attributes and structures, and their labels.

Attribute Over-emphasis of CLBAs. Compared with DLBAs, existing CLBAs do not inject new nodes into the original graph, thereby ensuring the stealthiness of backdoor attacks. Instead, they alter the attributes of poisoned nodes and prompt GNNs to establish connections between altered attributes and y^t. Consequently, the altered attributes involuntary play a crucial role in guiding backdoored GNNs to classify poisoned nodes into the target label.

On the Cora dataset, we conduct two cutting-edge CLBAs, namely GCBA [49] and PerCBA [45]. Specifically, we execute semi-supervised learning on the manipulated graph with the manipulated label information and perform attribute importance analysis using the gradient-guided explanation mechanism [31] using backdoored GNNs. We visualize nodes in the latent representation space generated by backdoored GNNs and the attribute importance of nodes using t-SNE. Detailed settings are available in Appendix A-B.

Based on the findings (Fig. 3), we observe that akin to discoveries in DLBAs, latent representations of poisoned nodes exhibit a propensity to aggregate and are situated close to the cluster of normal nodes labeled the target label. This elucidates why CLBAs can effectively implant backdoors into GNNs. Moreover, evaluations on the attribute importance analysis indicate that, since CLBAs alter attributes of poisoned nodes to strengthen connections between altered attributes and the target label, backdoored GNNs inevitably and excessively rely on altered attributes to make predictions. As shown in Fig. (3c) and (3d), poisoned nodes with altered attributes demonstrate high similarities of important attributes, which is called *attribute over-emphasis*.

In summary, the semantic drift of DLBAs reflects the changes in latent representation distances of poisoned nodes in self-supervised and backdoored models, while the attribute over-emphasis of CLBAs reflects an excessive reliance on specific attributes of poisoned nodes within backdoored models. The former highlights disparities in semantic information, while the latter signifies variations in attribute importance. Additionally, these discrepancies, stemming from the effectiveness of DLBAs and CLBAs, remain consistent across various backdoor attacks. Therefore, constructing self-supervised and backdoored models to identify these discrepancies is essential for detecting poisoned nodes.

V. The Proposed DShield

In this section, we propose a unified defensive framework, namely DShield, that leverages these two discrepancies to identify and mitigate the impact of poisoned nodes in both dirty- and clean-label scenarios. Additionally, the proposed scheme can also defeat attacks on graph classification tasks.

A. Overview

The DShield framework (Fig. 4), comprises three primary modules: "auxiliary model training", "discrepancy matrix construction", and "backdoor-free model training". These modules collectively aim to develop a backdoor-free model resilient to backdoor attacks within the manipulated graph.

DShield employs three distinct GNN models: a self-supervised model g, a backdoored model $\widetilde{f}$, and a normal model f. The self-supervised model g uses an encoder-decoder architecture, where $g_{\text{enc}}(\cdot)$ extracts latent representations $\boldsymbol{h}$ of each node via a message-passing mechanism, and $g_{\text{dec}}(\cdot)$ reconstructs attributes during self-supervised learning. Additionally, $\widetilde{f}$ and f are traditional node classification models with no specific constraints.

Auxiliary Model Training Module. Given the high similarity of latent representations between poisoned and normal nodes and the significance of analyzing attribute importance in CLBAs (Section IV), we train the backdoored model $\widetilde{f}$ using the manipulated graph. To address the lack of prior knowledge about backdoor attacks, we utilize the self-supervised learning paradigm to train the self-supervised model g. The backdoored model $\widetilde{f}$ and the self-supervised model g enable the analysis of semantic differences between the two models and attribute importance differences within the backdoored model.

Discrepancy Matrix Construction Module. Inspired by the semantic and attribute importance discrepancies identified in Section IV, two distinct discrepancy matrices are constructed. Clustering algorithms are then applied to identify nodes whose distributions of semantic information and attribute importance deviate from those of the majority of nodes. Based on the attribute drift fact observed in DLBAs, the semantic discrepancy matrix quantifies variations in node latent representations between models trained via self-supervised learning and backdoored models trained through semi-supervised learning. Meanwhile, following the attribute over-emphasis fact in CLBAs, the attribute importance discrepancy matrix assesses the influence of each attribute on prediction results using the backdoored model.

Backdoor-free Model Training Module. The graph is partitioned into segments based on the nodes identified in the previous module. A customized loss function is designed to construct the backdoor-free model, enhancing performance on the sub-graph without triggers while minimizing performance on the sub-graph containing triggers.

B. Auxiliary Model Training

Due to the observed semantic drift in DLBAs, we opt to analyze differences in latent representations of backdoored models $\widetilde{f}$ and self-supervised model g. Thus, we first train the backdoored model $\widetilde{f}$ on the manipulated graph $\widetilde{\mathcal{G}}$ using the standard semi-supervised learning described in Eq. (2). Subsequently, we adopt a self-supervised learning mechanism to train the model g, which consists of the encoder model g_{enc} and the decoder model g_{dec}. The self-supervised learning framework adopted by DShield follows the graph contrastive learning paradigm, where the model aims to maximize the consistency between diverse views [23], [41]. However, existing paradigms face two challenges: 1) the augmentation function might inadvertently overlook significant edges and attributes when generating different graph views, and 2) the inclusion of selected false negative node pairs in contrastive learning could distort learned representations. To alleviate these challenges, DShield employs a *three-fold* mechanism consisting of view augmentation, view encoding, and contrast and reconstruction.

View Augmentation. Graph contrastive learning aims to enhance consistency between different augmented views, thereby producing representations that resist perturbations introduced by augmentation schemes [21], [39], [53], [34]. This stage involves sampling two stochastic augmentation functions, denoted as $t_1 \sim \mathcal{T}$ and $t_2 \sim \mathcal{T}$, with $\mathcal{T}$ representing the set of all possible augmentation functions. The resulting graph views denoted as $\widehat{\mathcal{G}}_1 = t_1(\widetilde{\mathcal{G}})$ and $\widehat{\mathcal{G}}_2 = t_2(\widetilde{\mathcal{G}})$, are obtained by incorporating view augmentation functions, such as dropping structures and masking attributes. The objective is to preserve essential structures and attributes while perturbing potentially less significant ones.

For structure-level augmentation, a direct method involves randomly removing edges in the manipulated graph. The modified subset $\widehat{\mathcal{E}}$ is sampled from the original edge set $\mathcal{E}$ based on the probability distribution:

$$P\{(v_i, v_j) \in \widehat{\mathcal{E}}\} = 1 - p^s_{ij}, \tag{7}$$

where $(v_i, v_j) \in \widetilde{\mathcal{E}}$ represents an existing edge in the manipulated graph, and p^s_{ij} is the probability of removing (v_i, v_j). The importance of each edge (v_i, v_j) is quantified as p^s_{ij}, ensuring that the augmentation function is more likely to corrupt unimportant edges while preserving crucial connective structures. The homophily property of graphs, where nodes tend to connect with similar others, informs the assessment of edge importance, expressed as:

$$p^s_{ij} = \frac{\exp(-s(\widetilde{\boldsymbol{h}}_i, \widetilde{\boldsymbol{h}}_j))}{\sum_{(v_k,v_t)\in\widetilde{\mathcal{E}}} \exp(-s(\widetilde{\boldsymbol{h}}_k, \widetilde{\boldsymbol{h}}_t))}, \tag{8}$$

where $\widetilde{\boldsymbol{h}}_i = \widetilde{f}(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}; \phi_1)_i$ denotes the latent representation of node v_i in the backdoored models, and $s(\cdot,\cdot)$ is the similarity function measuring distances between vectors. Given that the backdoored model is a classification model, we denote ϕ_1 as the parameters of the middle layers used to obtain the latent representations of nodes.

Attribute-level augmentation introduces noise to node attributes by masking a fraction of dimensions with zeros. To avoid masking crucial attributes, an adaptive attribute masking approach is designed based on attribute contributions to prediction results. In detail, a random vector $\widehat{\boldsymbol{m}}_i \in \{0,1\}^F$ is sampled, where each dimension is independently drawn from a Bernoulli distribution, i.e., $\widehat{m}_{ij} \sim \text{Bern}(1 - p^a_{ij})$, and p^a_{ij} represents the probability of discarding the j-th attribute of node v_i. Each attribute's importance is assessed through:

$$\boldsymbol{g}_i = \frac{\partial\, \text{CE}(f(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}; \phi)_i, \boldsymbol{y}_i)}{\partial \boldsymbol{x}_i}, \tag{9}$$

where $\boldsymbol{y}_i$ denotes the label within the graph. Subsequently, attributes with positive contributions are retained and others discarded, as indicated by $w_{ij} = \mathbb{1}_{g_{ij}>0}$, where $\mathbb{1}(\cdot)$ denotes the indicator function. Finally, the attribute importance is used to modify the randomly sampled vector $\widehat{\boldsymbol{m}}_i = \widehat{\boldsymbol{m}}_i \oplus \boldsymbol{w}_i$, where

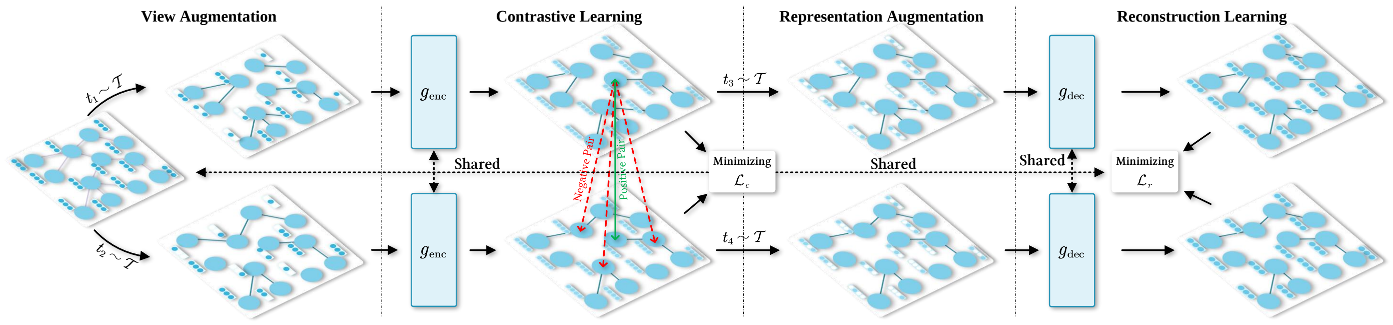


Fig. 5: Framework of self-supervised learning adopted in the "auxiliary model training" module.

$\oplus$ denotes the union operation. The perturbed attributes $\widehat{\boldsymbol{X}}$ are computed as:

$$\widehat{\boldsymbol{X}} = [\boldsymbol{x}_1 \circ \widehat{\boldsymbol{m}}_1; \boldsymbol{x}_2 \circ \widehat{\boldsymbol{m}}_2; \ldots; \boldsymbol{x}_N \circ \widehat{\boldsymbol{m}}_N]^\top . \tag{10}$$

Here, $[\cdot;\cdot]$ is the concatenation operator, and $\circ$ denotes element-wise multiplication.

View Encoding. View encoding aims to extract latent representations of nodes by feeding the data from two augmented graphs, $\widehat{\mathcal{G}}_1$ and $\widehat{\mathcal{G}}_2$, into the encoder g_{enc}. This process ensures that the generated representations preserve both the structure and attribute information of augmented views, expressed as:

$$\widehat{\boldsymbol{H}}_1 = g_{\text{enc}}(\widehat{\boldsymbol{A}}_1, \widehat{\boldsymbol{X}}_1), \quad \widehat{\boldsymbol{H}}_2 = g_{\text{enc}}(\widehat{\boldsymbol{A}}_2, \widehat{\boldsymbol{X}}_2). \tag{11}$$

The encoder g_{enc} shows flexibility in graph contrastive learning, with examples including Graph Convolutional Networks (GCNs) [17] and Graph Attention Networks (GATs) [36].

Contrast and Reconstruction. Contrastive learning aims to obtain discriminative representations by distinguishing between representations of the same node in two different views and those of other nodes. For a given node v_i, its representations in one view, e.g., $\widehat{\mathcal{G}}_1$, serve as the anchor, while its representations in the other view, e.g., $\widehat{\mathcal{G}}_2$, constitute the positive sample. Representations of nodes other than v_i in the other view are considered negative samples. Formally, for a chosen anchor view, the contrastive loss is defined as:

$$\mathcal{L}_{c_1} = \sum_{v_i \in \mathcal{V}^{\text{train}}} -\frac{1}{|\mathcal{P}(v_i)|} \sum_{v_j \in \mathcal{P}(v_i)} \log \frac{\exp(s(\widehat{\boldsymbol{h}}_{1,i}, \widehat{\boldsymbol{h}}_{2,j})/\tau)}{\sum_{v_k \in \mathcal{N}(v_i)} \exp(s(\widehat{\boldsymbol{h}}_{1,i}, \widehat{\boldsymbol{h}}_{2,k})/\tau)} , \tag{12}$$

where $\mathcal{P}(v_i)$ denotes the set of positive nodes for node v_i, and $\mathcal{N}(v_i)$ represents the set of negative nodes for node v_i. The similarity function is denoted as $s(\cdot,\cdot)$, and the τ is the temperature coefficient. Given that each view can serve as the anchor and Eq. (12) treats $\widehat{\mathcal{G}}_1$ as the anchor view, we can naturally deduce the contrastive loss $\mathcal{L}_{c_2}$ by replacing $\widehat{\mathcal{G}}_1$ of Eq. (12) with $\widehat{\mathcal{G}}_2$. The contrastive loss is represented as:

$$\mathcal{L}_c = {}^1\!/\!_2\mathcal{L}_{c_1} + {}^1\!/\!_2\mathcal{L}_{c_2}. \tag{13}$$

In Eq. (12), the number of positive node pairs is considerably smaller than the number of negative pairs, resulting in numerous false negative samples. These false negative pairs, under the objective of contrastive learning, are encouraged to be distanced from the anchors in the representation space, thereby potentially amplifying the divergence between nodes with the same true labels. To address this challenge, we introduce *representation augmentation* and *reconstruction loss* as proposed by Zhao *et al.* [52] to enhance the mutual information between original attributes and corresponding representations, thus alleviating the loss of semantic information for false negative nodes.

Specifically, a random vector $\widehat{\boldsymbol{m}}_i \in \{0,1\}^F$ is sampled, where each dimension is independently drawn from a Bernoulli distribution, i.e., $\widehat{\boldsymbol{m}}_i \sim \text{Bern}(1-p^r_{ij})$, with p^r_{ij} representing the probability of masking the j-th dimension of the representation of node v_i. Subsequently, the perturbed representations are computed as:

$$\widehat{\boldsymbol{H}}'_1 = \left[\widehat{\boldsymbol{h}}_1 \odot \widehat{\boldsymbol{m}}'_1; \widehat{\boldsymbol{h}}_2 \odot \widehat{\boldsymbol{m}}'_2; \cdots, \widehat{\boldsymbol{h}}_N \odot \widehat{\boldsymbol{m}}'_N\right]^\top , \tag{14}$$

which is analogous to $\widehat{\boldsymbol{H}}'_2$. Subsequently, the masked representations are fed into the block g_{dec} to reconstruct node attributes, as expressed in:

$$\widehat{\boldsymbol{X}}'_1 = g_{\text{dec}}(\widehat{\boldsymbol{A}}_1, \widehat{\boldsymbol{H}}'_1), \quad \widehat{\boldsymbol{X}}'_2 = g_{\text{dec}}(\widehat{\boldsymbol{A}}_2, \widehat{\boldsymbol{H}}'_2). \tag{15}$$

The reconstruction loss is subsequently defined as:

$$\mathcal{L}_r = \frac{1}{2}\sum_{i=1}^{N}\left(1 - s(\widetilde{\boldsymbol{x}}_i, \widehat{\boldsymbol{x}}'_{1,i})\right)^2 + \frac{1}{2}\sum_{i=1}^{N}\left(1 - s(\widetilde{\boldsymbol{x}}_i, \widehat{\boldsymbol{x}}'_{2,i})\right)^2 , \tag{16}$$

where $\widehat{\boldsymbol{x}}'_{1,i}$ represents the reconstructed attributes of node v_i in the first augmented view, and $s(\cdot,\cdot)$ denotes the similarity function, set to the cosine function, considering variations in attribute magnitudes [9].

In summary, the total loss is expressed as:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_r. \tag{17}$$

After constructing the self-supervised model g and the backdoored model $\widetilde{f}$, we would analyze labeled nodes from the perspective of two observed facts to identify potentially poisoned nodes.

C. Discrepancy Matrix Construction

To identify poisoned nodes, the defender strives to construct a discrepancy matrix by considering the semantic drift and attribute over-emphasis facts, as elucidated in Section IV. Nevertheless, considering that the training graph may be extraordinarily large, the task of building the discrepancy matrix to detect potentially poisoned nodes from the entire pool of labeled nodes remains formidable. Thus, a discrepancy matrix

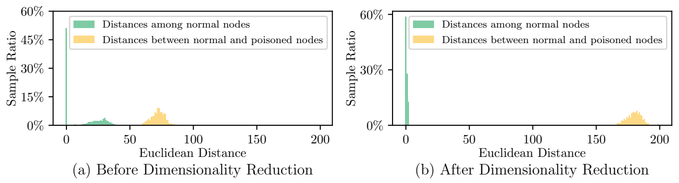


Fig. 6: Illustration of UMAP's impact, showcasing its ability to enlarge distances between normal and poisoned nodes.

is constructed for each label, as the defender is unaware of the target label. Subsequently, noise-tolerant clustering algorithms leverage these matrices to identify potentially poisoned nodes for each label.

Semantic Discrepancy Matrix. In light of observed facts indicating differences in semantic information between poisoned nodes and normal nodes on DLBAs, the initial step involves obtaining latent representations of training nodes with labels:

$$\widetilde{\boldsymbol{H}}_{\text{sl}} = \widetilde{f}(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}; \phi_1), \quad \widetilde{\boldsymbol{H}}_{\text{ssl}} = g_{\text{enc}}(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}), \tag{18}$$

where $\widetilde{f}(\cdot)$ denotes the backdoored model trained through semi-supervised learning, ϕ_1 is the parameters of the first layer of the victim model, and g_{enc} represents the encoder part of the self-supervised model g trained through self-supervised learning. Subsequently, the Euclidean distance matrices [7] for nodes with the label y are computed as:

$$\begin{aligned} \boldsymbol{D}^y_{\text{sl}} &= \mathbf{1}\,\text{diag}(\widetilde{\boldsymbol{H}}^y_{\text{sl}}\widetilde{\boldsymbol{H}}^{y\top}_{\text{sl}})^\top - 2\widetilde{\boldsymbol{H}}^y_{\text{sl}}\widetilde{\boldsymbol{H}}^{y\top}_{\text{sl}} + \text{diag}(\widetilde{\boldsymbol{H}}^y_{\text{sl}}\widetilde{\boldsymbol{H}}^{y\top}_{\text{sl}})\mathbf{1}^\top; \\ \boldsymbol{D}^y_{\text{ssl}} &= \mathbf{1}\,\text{diag}(\widetilde{\boldsymbol{H}}^y_{\text{ssl}}\widetilde{\boldsymbol{H}}^{y\top}_{\text{ssl}})^\top - 2\widetilde{\boldsymbol{H}}^y_{\text{ssl}}\widetilde{\boldsymbol{H}}^{y\top}_{\text{ssl}} + \text{diag}(\widetilde{\boldsymbol{H}}^y_{\text{ssl}}\widetilde{\boldsymbol{H}}^{y\top}_{\text{ssl}})\mathbf{1}^\top, \end{aligned} \tag{19}$$

where $\mathbf{1}$ signifies the column vector of all ones, and $\text{diag}(\boldsymbol{A})$ denotes a column vector of the diagonal entries of $\boldsymbol{A}$. The matrix $\widetilde{\boldsymbol{H}}^y_{\text{sl}} \in \mathbb{R}^{N_y \times d}$ represents latent representations of nodes labeled y, with N_y denoting the number of nodes labeled y.

Given substantial changes in the distance between poisoned and normal nodes labeled y^t in latent representations trained through semi-supervised and self-supervised learning, the semantic discrepancy matrix $\boldsymbol{D}_s$ captures these changes:

$$\boldsymbol{D}^y_s = \max(\boldsymbol{D}^y_{\text{ssl}} - \boldsymbol{D}^y_{\text{sl}}, 0). \tag{20}$$

As explained in Section IV, poisoned nodes exhibit similarly from normal nodes labeled with the target label in latent representations trained through semi-supervised learning while displaying a substantial distance in those trained through self-supervised learning. Thus, only positive distance is preserved.

Attribute Importance Discrepancy Matrix. In alignment with previous facts indicating that poisoned nodes manipulated by CLBAs exhibit exaggerated attention to specific attributes distinct from normal nodes, the attribute importance discrepancy matrix is constructed to assess the dissimilarity between nodes sharing the same labels. Formally, we derive the mask matrix $\boldsymbol{W}$ indicating the positions of attributes contributing positively to prediction results. Consequently, for nodes with the label y, these attributes are obtained through the equation:

$$\widetilde{\boldsymbol{X}}^y = \widetilde{\boldsymbol{X}}^y \circ \boldsymbol{W}^y, \tag{21}$$

where $\boldsymbol{W}^y$ solely comprises the mask of nodes with the label y. However, the calculation of Euclidean distance is not directly applicable to node attributes due to their high dimensionality, posing challenges of the curse of dimensionality [14].

To mitigate this challenge, we introduce a manifold-based dimension reduction technique, specifically Uniform Manifold Approximation and Projection (UMAP) [26], to reduce the dimension of $\widetilde{\boldsymbol{X}}^y$ from $\mathbb{R}^{N_y \times F}$ to $\mathbb{R}^{N_y \times d}$:

$$\overline{\boldsymbol{X}}^y = \text{UMAP}(\widetilde{\boldsymbol{X}}^y, d). \tag{22}$$

UMAP is chosen due to its absence of restrictions on latent representations' dimensionality and its efficient computation speed while preserving the global structure. In Fig. 6, we present a visualization depicting alterations in distances among poisoned nodes, as well as distances between normal and poisoned nodes, both before and after employing dimensionality reduction on the Cora dataset. Our observation indicates that UMAP effectively amplifies the divergences in attribute importance between normal and poisoned nodes. Finally, the distance of important attributes within nodes with the label y is computed as:

$$\boldsymbol{D}^y_a = \mathbf{1}\,\text{diag}(\overline{\boldsymbol{X}}^y\overline{\boldsymbol{X}}^{y\top})^\top - 2\overline{\boldsymbol{X}}^y\overline{\boldsymbol{X}}^{y\top} + \text{diag}(\overline{\boldsymbol{X}}^y\overline{\boldsymbol{X}}^{y\top})\mathbf{1}^\top. \tag{23}$$

Additionally, Fig. 12 (e)-(f) highlights the differences in the elements of the attribute importance discrepancy matrix between normal and malicious nodes in the Cora dataset.

Clustering. Given the unknown specific type of backdoor attack to the defender, a fusion of the two discrepancy matrices is performed based on the divergence observed in Euclidean distances within nodes:

$$\boldsymbol{D}^y = \frac{\text{std}(\boldsymbol{D}^y_s)}{\text{std}(\boldsymbol{D}^y_s) + \text{std}(\boldsymbol{D}^y_a)}\boldsymbol{D}^y_s + \frac{\text{std}(\boldsymbol{D}^y_a)}{\text{std}(\boldsymbol{D}^y_s) + \text{std}(\boldsymbol{D}^y_a)}\boldsymbol{D}^y_a. \tag{24}$$

Here, the standard deviation $\text{std}(\cdot)$ of elements within the discrepancy matrix is employed to quantify the information magnitude contained therein.

For all nodes labeled y, a clustering algorithm is deployed to partition these nodes into two clusters based on the merged discrepancy matrix $\boldsymbol{D}^y$. In this context, HDBSCAN [2] is employed as the clustering algorithm, leveraging its capability to cluster nodes based on the density of the distance distribution while dynamically determining the required number of clusters. Specifically, HDBSCAN designates nodes as outliers if they do not conform to any cluster, allowing DShield to categorize poisoned nodes as outliers. Since the number of poisoned nodes is unknown to the defender, we assume the attack can manipulate a maximum of 50% of nodes with the y label, denoted as $N_y/2+1$, and set this as the minimum cluster size. We then assume the cluster with the majority of nodes to be normal, denoted as $\mathcal{V}^y_1$. All remaining nodes, potentially poisoned, are designated as outliers and represented as $\mathcal{V}^y_2$:

$$\mathcal{V}^y_1, \mathcal{V}^y_2 = \text{HDBSCAN}(\boldsymbol{D}^y). \tag{25}$$

D. Backdoor-free Model Training

Having identified candidate poisoned nodes for each label, the direct elimination of all candidate poisoned nodes, while effective in bolstering defense performance, invariably leads to a degradation in model performance on normal nodes. Thus, an elegant approach involves determining the target label y^t and retaining nodes associated with other labels.

Target Label Discovery. Drawing insights from the case study in Section IV, wherein poisoned nodes exhibit substantial semantic divergence or excessive focus on particular attributes, the identification of y^t assigned to poisoned nodes entails leveraging the differences in structures and attributes between normal and poisoned nodes. A node classification model denoted as f', is trained on all normal nodes $\{\mathcal{V}_1^1, \cdots, \mathcal{V}_1^y, \cdots, \mathcal{V}_1^C\}$, employing distances of latent representations between normal nodes $\mathcal{V}_1^y$ and candidate poisoned nodes $\mathcal{V}_2^y$ to determine the suspicious score of the y label.

To ensure the discernment of latent representation differences stemming from distinct structures and attributes, the training graph structure is bifurcated into two parts:

$$\widetilde{A}_{ij}^{y,\text{pos}} \begin{cases} = \widetilde{A}_{ij}, & \text{if } v_i \notin \mathcal{V}_2^y \text{ and } v_j \notin \mathcal{V}_2^y; \\ = 0, & \text{if } v_i \in \mathcal{V}_2^y \text{ or } v_j \in \mathcal{V}_2^y, \end{cases} \tag{26}$$

and

$$\widetilde{A}_{ij}^{y,\text{neg}} \begin{cases} = \widetilde{A}_{ij}, & \text{if } v_i \in \mathcal{V}_2^y \text{ or } v_j \in \mathcal{V}_2^y; \\ = 0, & \text{if } v_i \notin \mathcal{V}_2^y \text{ and } v_j \notin \mathcal{V}_2^y. \end{cases} \tag{27}$$

Here, $\widetilde{A}_{ij}^{y,\text{pos}}$ ensures no edges are connected to potentially poisoned nodes $\mathcal{V}_2^y$, while $\widetilde{A}_{ij}^{y,\text{neg}}$ contains edges where at least one node is from $\mathcal{V}_2^y$.

Subsequently, a customized loss function is devised to maximize the model's performance on normal nodes while minimizing its performance on candidate poisoned nodes. The first term can be defined as:

$$\mathcal{L}_{\text{pos}} = \frac{1}{|\mathcal{V}_1^y|} \sum_{v_i \in \mathcal{V}_1^y} \text{CE}(f'(\widetilde{\boldsymbol{A}}^{y,\text{pos}}, \widetilde{\boldsymbol{X}})_i, \boldsymbol{y}_i). \tag{28}$$

The second term can be expressed as:

$$\mathcal{L}_{\text{neg}} = \frac{1}{|\mathcal{V}_2^y|} \sum_{v_i \in \mathcal{V}_2^y} \text{CE}(f'(\widetilde{\boldsymbol{A}}^{y,\text{neg}}, \widetilde{\boldsymbol{X}})_i, \boldsymbol{y}_i). \tag{29}$$

The overall loss function for model training is then defined as:

$$\mathcal{L} = \mathcal{L}_{\text{pos}} - \log(\mathcal{L}_{\text{neg}}), \tag{30}$$

where $\log(\cdot)$ is incorporated to prevent a situation where the difference between $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{neg}}$ is minimized while the values of $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{neg}}$ increase.

The center of latent representations of nodes within the set $\mathcal{V}_1^y$ is obtained as:

$$\boldsymbol{h}^y = \frac{1}{|\mathcal{V}_1^y|} \sum_{v_i \in \mathcal{V}_1^y} f'(\widetilde{\boldsymbol{A}}^{y,\text{pos}}, \widetilde{\boldsymbol{X}}; \phi_1')_i. \tag{31}$$

Finally, the suspicious score for the label y is computed as:

$$s^y = \frac{1}{|\mathcal{V}_2^y|} \sum_{v_i \in \mathcal{V}_2^y} \left\| f'(\widetilde{\boldsymbol{A}}^{y,\text{neg}}, \widetilde{\boldsymbol{X}}; \phi_1')_i - \boldsymbol{h}^y \right\|_2^2. \tag{32}$$

Given that the defender cannot be certain whether the third-party graph is manipulated or the number of target labels, we adopt the anomaly detection technique with Median Absolute Deviation (MAD) [30] to identify target labels, represented as:

$$\mathbb{y} = \{ y \mid y \in [1, C] \wedge {}^{|s^y - \text{median}(\boldsymbol{s})|}/_{\text{mad}(\boldsymbol{s})} \leq \beta \}, \tag{33}$$

where $\text{median}(\boldsymbol{s})$ is the median value of all suspicious scores, and $\text{mad}(\boldsymbol{s})$ is calculated as:

$$\text{mad}(\boldsymbol{s}) = \text{median}(|s^1 - \text{median}(\boldsymbol{s})|, \cdots, |s^C - \text{median}(\boldsymbol{s})|) . \tag{34}$$

Consequently, candidate poisoned nodes $\mathcal{V}_2^{\mathbb{y}}$ are identified as poisoned nodes, while candidate poisoned nodes with other labels are considered normal nodes, resulting in:

$$\mathcal{V}^\dagger = \bigcup_{y=1}^{C} \mathcal{V}_1^y \cup \bigcup_{y=1, y \notin \mathbb{y}}^{C} \mathcal{V}_2^y; \qquad \mathcal{V}^\ddagger = \mathcal{V}_2^{\mathbb{y}}, \tag{35}$$

where $\mathcal{V}^\dagger$ and $\mathcal{V}^\ddagger$ represent the sets of normal nodes and identified poisoned nodes, respectively.

Backdoor-free Training. Based on identified normal nodes, a classification loss is formulated to optimize model parameters:

$$\mathcal{L}_{\text{clf}} = \frac{1}{|\mathcal{V}^\dagger|} \sum_{v_i \in \mathcal{V}^\dagger} \text{CE}(f(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}})_i, \boldsymbol{y}_i). \tag{36}$$

Some backdoor attacks, such as GB-FGSM [4], utilize optimization-based methods to generate triggers, enhancing their transferability to deceive models. To address this issue, we introduce a penalty loss to minimize the performance of the backdoor-free model f on these triggers, reducing the transferability of these attacks. However, as poisoned nodes $\mathcal{V}^\ddagger$ are embedded within the structures, negative influences introduced by the message-passing mechanism need to be mitigated. Given the identified poisoned nodes, the subgraph associated with them is constructed:

$$\widetilde{A}_{ij}' = \begin{cases} \widetilde{A}_{ij}, & \text{if } v_i \in \mathcal{V}^\ddagger \text{ or } v_i \in \mathcal{V}^\ddagger; \\ 0, & \text{if } v_i \notin \mathcal{V}^\ddagger \text{ and } v_i \notin \mathcal{V}^\ddagger. \end{cases} \tag{37}$$

Subsequently, a penalty loss is then introduced to eliminate the influences:

$$\mathcal{L}_{\text{penalty}} = \frac{1}{|\mathcal{V}^\ddagger|} \sum_{v_i \in \mathcal{V}^\ddagger} \text{CE}(f(\widetilde{\boldsymbol{A}}', \widetilde{\boldsymbol{X}})_i, \boldsymbol{y}_i). \tag{38}$$

The total loss function is expressed as:

$$\mathcal{L} = \mathcal{L}_{\text{clf}} - \gamma \log \mathcal{L}_{\text{penalty}}. \tag{39}$$

In summary, we introduce a self-supervised mechanism to train the self-supervised model and utilize the self-supervised and backdoored models along with an attribute importance analysis technique to identify poisoned nodes through semantic and attribute importance discrepancies. This approach effectively defends against node classification attacks, including DLBAs and CLBAs, and also mitigates backdoor attacks on graph classification tasks with slight modifications (Details can be found in Appendix A-E).

VI. Experimental Evaluation

This section presents a comprehensive evaluation of the performance of DShield across datasets of various scales. The objective is to address the following research questions:

- **RQ1-Effectiveness**: To what extent does DShield effectively mitigate the efficacy of various attacks while concurrently preserving the performance of the victim model on normal nodes?

TABLE I: Comparison of DShield's effectiveness with state-of-the-art defenses against DLBAs (%) using GCN as the victim.

Datasets	Defenses	SBA [50]		GTA [38]		EBA [43]		GB-FGSM [4]		LGCB [4]		UGBA [5]		TRAP [44]	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
Cora	*no-defense*	53.14$_{\pm 2.10}$	84.07$_{\pm 1.02}$	96.22$_{\pm 4.41}$	82.74$_{\pm 2.38}$	71.81$_{\pm 9.44}$	**83.48**$_{\pm 1.32}$	97.34$_{\pm 1.64}$	82.52$_{\pm 1.53}$	97.79$_{\pm 1.01}$	83.70$_{\pm 0.52}$	97.51$_{\pm 2.07}$	83.03$_{\pm 1.32}$	19.86$_{\pm 2.42}$	**84.15**$_{\pm 1.09}$
	Prune [5]	30.26$_{\pm 2.78}$	83.26$_{\pm 0.96}$	16.42$_{\pm 2.35}$	**84.00**$_{\pm 1.80}$	100.0$_{\pm 0.00}$	80.74$_{\pm 1.57}$	56.98$_{\pm 1.00}$	81.92$_{\pm 3.36}$	56.98$_{\pm 0.93}$	81.92$_{\pm 1.95}$	54.47$_{\pm 2.19}$	83.41$_{\pm 1.52}$	11.36$_{\pm 2.08}$	81.48$_{\pm 1.87}$
	Isolate [5]	04.06$_{\pm 2.42}$	37.19$_{\pm 1.07}$	03.54$_{\pm 1.85}$	77.78$_{\pm 1.48}$	96.43$_{\pm 1.66}$	80.67$_{\pm 1.29}$	56.09$_{\pm 0.78}$	81.85$_{\pm 1.08}$	56.46$_{\pm 0.78}$	81.63$_{\pm 1.97}$	56.02$_{\pm 0.80}$	80.74$_{\pm 3.21}$	**10.15**$_{\pm 1.94}$	67.26$_{\pm 1.63}$
	PXGBD [8]	46.37$_{\pm 2.03}$	84.00$_{\pm 0.85}$	97.05$_{\pm 2.46}$	83.40$_{\pm 1.74}$	71.04$_{\pm 9.65}$	82.74$_{\pm 1.40}$	97.86$_{\pm 1.44}$	82.74$_{\pm 0.93}$	98.38$_{\pm 1.64}$	82.37$_{\pm 1.13}$	97.86$_{\pm 2.17}$	**84.00**$_{\pm 0.55}$	22.14$_{\pm 2.31}$	81.55$_{\pm 1.37}$
	GXGBD [8]	56.28$_{\pm 1.31}$	**84.17**$_{\pm 1.30}$	83.77$_{\pm 1.56}$	83.42$_{\pm 1.10}$	88.19$_{\pm 2.81}$	73.52$_{\pm 2.75}$	98.16$_{\pm 1.16}$	**84.07**$_{\pm 1.09}$	98.43$_{\pm 1.01}$	**84.26**$_{\pm 0.98}$	80.81$_{\pm 2.30}$	83.15$_{\pm 2.31}$	19.70$_{\pm 3.84}$	81.18$_{\pm 0.80}$
	ABL [20]	54.86$_{\pm 2.37}$	82.81$_{\pm 1.40}$	97.79$_{\pm 2.09}$	82.59$_{\pm 2.38}$	81.43$_{\pm 3.98}$	83.26$_{\pm 1.37}$	97.56$_{\pm 1.34}$	83.11$_{\pm 0.81}$	98.08$_{\pm 0.84}$	83.70$_{\pm 0.59}$	97.42$_{\pm 1.04}$	83.89$_{\pm 2.19}$	18.38$_{\pm 2.52}$	83.33$_{\pm 0.64}$
	RS [49]	08.12$_{\pm 1.51}$	80.81$_{\pm 1.03}$	94.83$_{\pm 2.93}$	80.67$_{\pm 0.99}$	58.92$_{\pm 2.72}$	81.11$_{\pm 1.05}$	97.27$_{\pm 0.85}$	80.15$_{\pm 1.24}$	96.97$_{\pm 1.09}$	80.15$_{\pm 1.84}$	90.41$_{\pm 1.56}$	80.81$_{\pm 1.03}$	20.81$_{\pm 1.78}$	79.85$_{\pm 1.13}$
	DShield	**01.33**$_{\pm 2.56}$	81.78$_{\pm 1.93}$	**00.74**$_{\pm 0.74}$	81.92$_{\pm 1.98}$	**02.95**$_{\pm 2.32}$	81.50$_{\pm 1.08}$	**02.51**$_{\pm 2.83}$	82.29$_{\pm 0.71}$	**00.89**$_{\pm 1.15}$	82.57$_{\pm 0.51}$	**01.33**$_{\pm 1.21}$	82.15$_{\pm 0.80}$	13.38$_{\pm 2.12}$	82.07$_{\pm 1.65}$
PubMed	*no-defense*	62.84$_{\pm 5.05}$	85.17$_{\pm 0.20}$	96.54$_{\pm 2.28}$	85.09$_{\pm 0.26}$	84.79$_{\pm 2.10}$	85.23$_{\pm 0.17}$	99.67$_{\pm 0.66}$	85.03$_{\pm 0.15}$	97.14$_{\pm 1.44}$	85.16$_{\pm 0.22}$	93.27$_{\pm 3.23}$	85.20$_{\pm 0.25}$	48.34$_{\pm 0.93}$	85.21$_{\pm 0.38}$
	Prune [5]	53.01$_{\pm 2.59}$	**85.33**$_{\pm 0.37}$	46.15$_{\pm 0.40}$	**85.30**$_{\pm 0.21}$	99.95$_{\pm 0.00}$	**85.44**$_{\pm 0.18}$	70.28$_{\pm 0.40}$	**85.31**$_{\pm 0.25}$	69.36$_{\pm 0.56}$	**85.36**$_{\pm 0.18}$	66.85$_{\pm 1.47}$	**85.38**$_{\pm 0.29}$	**41.77**$_{\pm 0.28}$	**85.33**$_{\pm 0.25}$
	Isolate [5]	40.69$_{\pm 0.47}$	81.25$_{\pm 0.28}$	45.46$_{\pm 0.22}$	84.47$_{\pm 0.21}$	99.36$_{\pm 0.38}$	84.56$_{\pm 0.22}$	70.16$_{\pm 0.31}$	85.14$_{\pm 0.15}$	69.80$_{\pm 0.90}$	85.11$_{\pm 0.13}$	67.62$_{\pm 1.34}$	85.14$_{\pm 0.20}$	43.30$_{\pm 3.41}$	83.34$_{\pm 1.59}$
	PXGBD [8]	55.26$_{\pm 3.06}$	83.55$_{\pm 1.01}$	99.67$_{\pm 0.54}$	83.17$_{\pm 0.92}$	86.26$_{\pm 5.89}$	82.64$_{\pm 1.57}$	100.0$_{\pm 0.00}$	82.32$_{\pm 1.47}$	100.0$_{\pm 0.00}$	82.38$_{\pm 0.99}$	91.80$_{\pm 0.87}$	82.61$_{\pm 1.35}$	49.80$_{\pm 1.70}$	81.98$_{\pm 1.45}$
	GXGBD [8]	56.20$_{\pm 1.18}$	83.93$_{\pm 0.76}$	99.71$_{\pm 0.35}$	82.57$_{\pm 0.64}$	90.21$_{\pm 2.27}$	75.45$_{\pm 2.23}$	100.0$_{\pm 0.00}$	82.31$_{\pm 0.85}$	100.0$_{\pm 0.00}$	82.41$_{\pm 0.82}$	81.95$_{\pm 0.00}$	82.82$_{\pm 0.91}$	49.36$_{\pm 3.16}$	82.84$_{\pm 1.47}$
	ABL [20]	65.16$_{\pm 4.55}$	85.21$_{\pm 0.38}$	99.98$_{\pm 0.03}$	85.13$_{\pm 0.21}$	89.08$_{\pm 2.61}$	84.98$_{\pm 0.16}$	99.98$_{\pm 0.03}$	84.89$_{\pm 0.13}$	100.0$_{\pm 0.00}$	85.04$_{\pm 0.17}$	91.45$_{\pm 5.44}$	85.00$_{\pm 0.30}$	49.36$_{\pm 0.68}$	84.64$_{\pm 1.01}$
	RS [49]	51.10$_{\pm 3.63}$	83.13$_{\pm 0.45}$	97.53$_{\pm 1.33}$	82.99$_{\pm 0.37}$	89.86$_{\pm 1.20}$	83.68$_{\pm 0.32}$	100.0$_{\pm 0.00}$	83.39$_{\pm 0.58}$	100.0$_{\pm 0.00}$	83.40$_{\pm 0.57}$	59.68$_{\pm 4.13}$	83.17$_{\pm 0.31}$	48.48$_{\pm 3.50}$	82.83$_{\pm 0.71}$
	DShield	**35.91**$_{\pm 3.43}$	82.01$_{\pm 0.47}$	**00.73**$_{\pm 0.33}$	85.28$_{\pm 0.12}$	**02.78**$_{\pm 1.58}$	85.02$_{\pm 0.29}$	**00.66**$_{\pm 0.72}$	83.71$_{\pm 0.24}$	**03.54**$_{\pm 1.97}$	84.11$_{\pm 0.56}$	**02.24**$_{\pm 0.97}$	84.70$_{\pm 0.42}$	45.77$_{\pm 0.76}$	82.17$_{\pm 0.36}$
Flickr	*no-defense*	98.91$_{\pm 0.37}$	50.35$_{\pm 0.24}$	95.24$_{\pm 1.75}$	**50.78**$_{\pm 0.16}$	56.89$_{\pm 4.06}$	50.79$_{\pm 0.03}$	100.0$_{\pm 0.01}$	50.61$_{\pm 0.04}$	99.54$_{\pm 0.35}$	50.58$_{\pm 0.16}$	94.77$_{\pm 3.17}$	**50.86**$_{\pm 0.08}$	19.42$_{\pm 4.27}$	**50.69**$_{\pm 0.17}$
	Prune [5]	10.29$_{\pm 4.19}$	50.80$_{\pm 0.14}$	91.89$_{\pm 6.26}$	50.70$_{\pm 0.09}$	100.0$_{\pm 0.00}$	42.24$_{\pm 0.25}$	99.93$_{\pm 0.01}$	50.61$_{\pm 0.00}$	99.53$_{\pm 0.27}$	50.58$_{\pm 0.20}$	95.22$_{\pm 1.91}$	50.84$_{\pm 0.08}$	14.06$_{\pm 1.58}$	49.94$_{\pm 0.24}$
	Isolate [5]	00.30$_{\pm 0.67}$	27.76$_{\pm 0.42}$	93.99$_{\pm 1.66}$	50.75$_{\pm 0.16}$	57.59$_{\pm 6.60}$	**50.86**$_{\pm 0.01}$	99.92$_{\pm 0.00}$	50.71$_{\pm 0.16}$	99.47$_{\pm 0.30}$	50.64$_{\pm 0.11}$	95.30$_{\pm 2.38}$	50.95$_{\pm 0.13}$	20.36$_{\pm 4.50}$	50.25$_{\pm 0.26}$
	PXGBD [8]	99.85$_{\pm 0.16}$	47.79$_{\pm 2.82}$	94.47$_{\pm 6.07}$	48.37$_{\pm 1.19}$	00.03$_{\pm 0.04}$	47.67$_{\pm 0.61}$	99.91$_{\pm 0.10}$	48.82$_{\pm 2.14}$	99.90$_{\pm 0.07}$	47.22$_{\pm 1.36}$	96.23$_{\pm 2.78}$	48.07$_{\pm 1.17}$	24.81$_{\pm 3.24}$	47.65$_{\pm 1.48}$
	GXGBD [8]	98.96$_{\pm 0.83}$	44.29$_{\pm 4.17}$	99.37$_{\pm 0.58}$	40.96$_{\pm 0.70}$	38.64$_{\pm 2.06}$	41.05$_{\pm 0.08}$	100.0$_{\pm 0.01}$	41.57$_{\pm 0.02}$	100.0$_{\pm 0.01}$	41.31$_{\pm 1.41}$	97.93$_{\pm 1.93}$	40.98$_{\pm 0.61}$	00.10$_{\pm 0.21}$	40.34$_{\pm 0.02}$
	ABL [20]	40.39$_{\pm 4.05}$	50.77$_{\pm 0.17}$	95.93$_{\pm 2.41}$	50.39$_{\pm 0.30}$	58.28$_{\pm 3.33}$	50.70$_{\pm 0.03}$	99.98$_{\pm 0.03}$	50.47$_{\pm 0.21}$	99.58$_{\pm 0.36}$	50.51$_{\pm 0.23}$	95.93$_{\pm 3.39}$	50.22$_{\pm 0.62}$	22.34$_{\pm 2.98}$	50.59$_{\pm 0.28}$
	RS [49]	**00.00**$_{\pm 0.00}$	40.45$_{\pm 0.11}$	**00.17**$_{\pm 0.34}$	40.55$_{\pm 0.19}$	**00.00**$_{\pm 0.00}$	40.46$_{\pm 0.08}$	98.77$_{\pm 0.27}$	40.55$_{\pm 0.23}$	97.57$_{\pm 3.29}$	40.53$_{\pm 0.15}$	**00.00**$_{\pm 0.00}$	40.51$_{\pm 0.15}$	**00.00**$_{\pm 0.00}$	40.50$_{\pm 0.13}$
	DShield	**00.00**$_{\pm 0.00}$	**50.84**$_{\pm 0.59}$	00.47$_{\pm 0.50}$	50.26$_{\pm 0.93}$	**00.00**$_{\pm 0.00}$	50.76$_{\pm 0.36}$	**00.00**$_{\pm 0.00}$	**51.21**$_{\pm 0.12}$	**00.00**$_{\pm 0.00}$	**50.74**$_{\pm 0.79}$	03.49$_{\pm 4.56}$	50.64$_{\pm 0.80}$	05.56$_{\pm 4.13}$	50.46$_{\pm 0.20}$
OGBN -arXiv	*no-defense*	84.80$_{\pm 3.52}$	59.97$_{\pm 0.04}$	52.11$_{\pm 1.68}$	59.54$_{\pm 0.09}$	81.16$_{\pm 2.14}$	60.08$_{\pm 0.29}$	82.91$_{\pm 0.76}$	60.14$_{\pm 0.86}$	99.16$_{\pm 0.18}$	59.89$_{\pm 0.39}$	99.35$_{\pm 0.39}$	59.65$_{\pm 0.49}$	06.90$_{\pm 0.88}$	59.49$_{\pm 0.57}$
	Prune [5]	80.95$_{\pm 5.68}$	60.33$_{\pm 0.38}$	12.41$_{\pm 0.78}$	59.55$_{\pm 0.00}$	64.64$_{\pm 2.77}$	60.09$_{\pm 0.19}$	82.64$_{\pm 1.88}$	60.27$_{\pm 0.69}$	99.23$_{\pm 0.13}$	59.96$_{\pm 0.64}$	99.39$_{\pm 0.18}$	59.35$_{\pm 0.99}$	06.56$_{\pm 0.14}$	59.64$_{\pm 0.83}$
	Isolate [5]	03.30$_{\pm 1.03}$	44.09$_{\pm 0.09}$	11.66$_{\pm 1.20}$	59.64$_{\pm 0.01}$	63.56$_{\pm 6.55}$	60.10$_{\pm 0.29}$	83.01$_{\pm 0.95}$	60.19$_{\pm 0.68}$	99.16$_{\pm 0.21}$	59.90$_{\pm 0.37}$	99.41$_{\pm 0.13}$	59.39$_{\pm 0.78}$	06.56$_{\pm 0.37}$	59.91$_{\pm 0.10}$
	PXGBD [8]	84.33$_{\pm 4.26}$	61.04$_{\pm 0.42}$	53.70$_{\pm 5.25}$	60.93$_{\pm 0.07}$	00.03$_{\pm 0.05}$	**60.77**$_{\pm 0.31}$	85.65$_{\pm 0.58}$	**60.90**$_{\pm 0.07}$	99.34$_{\pm 0.18}$	**60.56**$_{\pm 0.10}$	99.85$_{\pm 0.05}$	**60.70**$_{\pm 0.21}$	04.93$_{\pm 0.33}$	**60.22**$_{\pm 0.10}$
	GXGBD [8]	92.21$_{\pm 3.08}$	54.68$_{\pm 1.07}$	48.35$_{\pm 4.19}$	53.73$_{\pm 0.47}$	44.91$_{\pm 6.92}$	54.60$_{\pm 1.27}$	80.26$_{\pm 2.02}$	54.77$_{\pm 0.30}$	98.76$_{\pm 0.09}$	53.47$_{\pm 0.11}$	99.61$_{\pm 0.06}$	52.87$_{\pm 0.37}$	19.56$_{\pm 1.39}$	41.13$_{\pm 0.97}$
	ABL [20]	84.69$_{\pm 3.59}$	59.75$_{\pm 0.02}$	49.56$_{\pm 6.43}$	60.23$_{\pm 0.33}$	74.69$_{\pm 2.04}$	59.71$_{\pm 0.41}$	82.34$_{\pm 1.84}$	59.52$_{\pm 0.03}$	99.30$_{\pm 0.28}$	60.30$_{\pm 0.73}$	98.75$_{\pm 0.20}$	57.74$_{\pm 0.21}$	07.06$_{\pm 1.38}$	59.98$_{\pm 0.45}$
	RS [49]	05.74$_{\pm 4.96}$	53.71$_{\pm 0.34}$	22.09$_{\pm 6.98}$	55.13$_{\pm 1.20}$	70.42$_{\pm 7.76}$	54.98$_{\pm 0.89}$	80.55$_{\pm 0.98}$	55.58$_{\pm 0.37}$	99.73$_{\pm 0.06}$	54.91$_{\pm 0.13}$	99.02$_{\pm 0.48}$	55.20$_{\pm 0.69}$	04.14$_{\pm 0.23}$	55.08$_{\pm 1.13}$
	DShield	**00.00**$_{\pm 0.00}$	**62.55**$_{\pm 1.12}$	**00.92**$_{\pm 1.29}$	**62.72**$_{\pm 0.03}$	**00.00**$_{\pm 0.00}$	59.36$_{\pm 0.30}$	**35.72**$_{\pm 7.90}$	60.10$_{\pm 0.23}$	**00.00**$_{\pm 0.00}$	57.92$_{\pm 2.32}$	**00.00**$_{\pm 0.00}$	58.32$_{\pm 0.07}$	**00.39**$_{\pm 0.12}$	59.91$_{\pm 0.08}$

- **RQ2-Robustness**: Can DShield maintain its efficacy across diverse configurations of attacks, including different poisoning rates (r_p), trigger sizes, and adaptive attack scenarios, e.g., mixtures of dirty- and clean-label attacks?
- **RQ3-Sensitivity**: Does the proposed approach exhibit consistency in its performance across varying settings of hyper-parameters, i.e., β and γ?
- **RQ4-Ablation Studies**: Do proposed mechanisms, such as the reconstruction loss, enhance DShield's performance?
- **RQ5-Graph Classification Attacks**: Can DShield defend against backdoor attacks on graph classification tasks?

A. Experimental Setup

Datasets. The evaluation uses 7 public datasets: Cora [32], Pubmed [32], Flickr [47], OGBN-arXiv [10], ENZYMES [15], PROTEINS [15], and MNIST [27]. The first 4 datasets, commonly employed for inductive semi-supervised node classification, include citation networks (Cora, Pubmed), a large-scale image caption graph (Flickr), and a substantial citation network (OGBN-arXiv). The last 3 datasets are employed for graph classification tasks. Dataset statistics are summarized in Table VII. For node classification tasks, we adopt the inductive task approach, where the adversary cannot access the testing graph during the poisoning phase. Following the dataset split approach by Dai *et al.* [5], we randomly mask out 20% of nodes, with half designated as poisoned nodes for attack evaluation and the other half as clean test nodes to assess backdoored model predictions on normal nodes. The training graph $\mathcal{G}^{\text{train}}$ comprises the remaining 80% of nodes, with labeled and validation node sets each accounting for 10%. For graph classification tasks, we randomly select 80% of the data as the training dataset and 10% as the testing dataset. All experiments are repeated 3 times, with mean values and standard deviations reported.

Baselines. DShield is compared against 6 state-of-the-art defenses: Prune [5], Isolate [5], XGBD-PGExplainer [8] (abbreviated as PXGBD), XGBD-GNNExplainer [8] (abbreviated as GXGBD), ABL [20], and RandomSmoothing [49] (abbreviated

TABLE II: Comparison of DShield's effectiveness with state-of-the-art defenses for 2 CLBAs (%). The victim model is GCN.

Defenses	Cora				PubMed				Flickr				OGBN-arXiv			
	GCBA [49]		PerCBA [45]		GCBA [49]		PerCBA [45]		GCBA [49]		PerCBA [45]		GCBA [49]		PerCBA [45]	
	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
no-defense	81.74$_{\pm 3.93}$	**84.74**$_{\pm 1.27}$	84.87$_{\pm 3.13}$	83.18$_{\pm 1.54}$	78.86$_{\pm 7.76}$	85.30$_{\pm 0.13}$	100.0$_{\pm 0.00}$	85.23$_{\pm 0.13}$	60.48$_{\pm 4.72}$	**50.84**$_{\pm 0.16}$	07.05$_{\pm 5.11}$	**50.86**$_{\pm 0.18}$	26.84$_{\pm 2.79}$	**60.43**$_{\pm 0.30}$	18.06$_{\pm 1.48}$	59.81$_{\pm 0.15}$
Prune [5]	99.70$_{\pm 0.31}$	82.37$_{\pm 0.42}$	69.01$_{\pm 1.31}$	82.66$_{\pm 0.55}$	99.83$_{\pm 0.18}$	**85.55**$_{\pm 0.16}$	95.58$_{\pm 4.50}$	**85.34**$_{\pm 0.30}$	78.50$_{\pm 3.63}$	49.97$_{\pm 0.27}$	30.38$_{\pm 2.70}$	50.12$_{\pm 0.15}$	23.14$_{\pm 0.30}$	60.40$_{\pm 0.35}$	19.03$_{\pm 1.15}$	59.78$_{\pm 0.17}$
Isolate [5]	99.88$_{\pm 0.21}$	71.92$_{\pm 2.05}$	81.73$_{\pm 2.35}$	69.04$_{\pm 2.07}$	99.10$_{\pm 1.09}$	84.12$_{\pm 0.14}$	97.88$_{\pm 1.05}$	84.26$_{\pm 1.01}$	66.44$_{\pm 5.40}$	50.34$_{\pm 0.08}$	30.15$_{\pm 3.17}$	50.52$_{\pm 0.18}$	22.24$_{\pm 0.98}$	**60.43**$_{\pm 0.08}$	19.00$_{\pm 0.64}$	59.75$_{\pm 0.18}$
PXGBD [8]	82.29$_{\pm 3.66}$	82.89$_{\pm 0.88}$	27.68$_{\pm 2.58}$	82.74$_{\pm 1.19}$	96.91$_{\pm 3.08}$	82.82$_{\pm 1.34}$	93.11$_{\pm 0.36}$	83.31$_{\pm 1.54}$	26.50$_{\pm 0.79}$	47.74$_{\pm 0.98}$	00.16$_{\pm 0.28}$	47.29$_{\pm 2.67}$	11.87$_{\pm 0.71}$	60.42$_{\pm 0.17}$	00.78$_{\pm 0.49}$	**60.42**$_{\pm 0.89}$
GXGBD [8]	04.61$_{\pm 1.77}$	82.96$_{\pm 0.52}$	80.69$_{\pm 2.98}$	**84.30**$_{\pm 1.07}$	97.30$_{\pm 0.51}$	82.43$_{\pm 1.11}$	72.90$_{\pm 1.18}$	82.78$_{\pm 0.32}$	49.90$_{\pm 2.39}$	40.57$_{\pm 0.30}$	02.42$_{\pm 4.19}$	40.61$_{\pm 0.44}$	09.21$_{\pm 1.54}$	55.94$_{\pm 0.16}$	00.34$_{\pm 0.14}$	48.51$_{\pm 0.43}$
ABL [20]	79.70$_{\pm 3.89}$	84.30$_{\pm 1.35}$	88.68$_{\pm 4.25}$	83.70$_{\pm 0.91}$	83.40$_{\pm 1.46}$	84.98$_{\pm 0.23}$	82.50$_{\pm 3.53}$	85.13$_{\pm 0.36}$	66.88$_{\pm 0.84}$	50.49$_{\pm 0.21}$	01.35$_{\pm 0.60}$	50.68$_{\pm 0.04}$	02.73$_{\pm 3.84}$	58.40$_{\pm 1.61}$	22.04$_{\pm 2.39}$	60.23$_{\pm 0.79}$
RS [49]	42.44$_{\pm 4.11}$	81.26$_{\pm 1.92}$	85.22$_{\pm 3.15}$	80.74$_{\pm 1.14}$	74.05$_{\pm 1.70}$	83.28$_{\pm 0.58}$	88.46$_{\pm 3.41}$	83.51$_{\pm 0.27}$	**00.02**$_{\pm 0.03}$	40.48$_{\pm 0.09}$	23.72$_{\pm 0.59}$	40.50$_{\pm 0.16}$	27.76$_{\pm 0.08}$	55.36$_{\pm 0.10}$	43.48$_{\pm 5.32}$	55.38$_{\pm 0.19}$
DShield	**01.11**$_{\pm 1.04}$	81.18$_{\pm 1.32}$	**03.41**$_{\pm 2.53}$	81.63$_{\pm 2.04}$	**01.91**$_{\pm 1.66}$	84.24$_{\pm 0.24}$	**08.09**$_{\pm 2.04}$	83.69$_{\pm 2.01}$	01.66$_{\pm 2.34}$	50.42$_{\pm 0.25}$	**00.04**$_{\pm 0.05}$	**50.86**$_{\pm 0.06}$	**00.00**$_{\pm 0.00}$	59.96$_{\pm 0.54}$	**00.00**$_{\pm 0.00}$	59.92$_{\pm 0.44}$

as RS). Details of each defense are listed in Section II.

Backdoor Attacks. The comprehensive assessment of DShield includes 10 DLBAs (SBA [50], GTA [38], EBA [43], GB-FGSM [4], LGCB [4], UGBA [5], TRAP [44], "TLN", DPGBA [51], and MLGB [37]) and 2 CLBAs (GCBA [49] and PerCBA [45]) for node classification attacks. The "TLN" denotes the attack in which the attributes of nodes within the triggers correspond to attributes of the target label. A two-layer GCN model serves as the surrogate model for backdoor attacks. For graph classification attacks, we adapt SBA [50], EBA [43], and GCBA [49] into G-SBA, G-EBA, and G-GCBA, respectively, as detailed in Appendix A-E. Additionally, we utilize SBAG [6] to inject triggers into the original graphs. In all experiments, label 2 is designated as the target label. The number of nodes within triggers for SBA, GTA, UGBA, and TRAP is set to 3, and 1 for other attacks. The number of poisoned nodes $|\mathcal{V}^p|$ is set to 10, 40, 80, and 160 for Cora, PubMed, Flickr, and OGB-arXiv. For graph classification attacks, we randomly sample 10% as poisoned graphs.

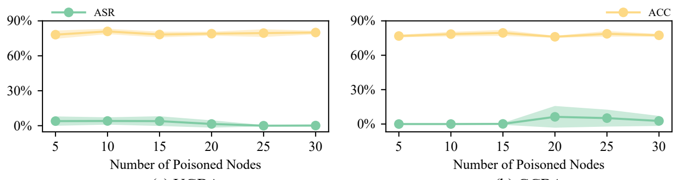
Victim Models. The victim models employed in evaluations include GNNs with varying architectures: GCN [17] and GAT [36]. Unless specified, the GCN model serves as the default GNN model.

Metrics. Evaluation metrics comprise the Attack Success Rate (ASR), measuring the effectiveness of the attack. ASR quantifies the proportion of poisoned nodes with triggers (not the target label) misclassified to the target label y^t. Accuracy (ACC), the rate of correctly classified normal nodes to all normal nodes, is utilized to assess the backdoored GNN's performance on normal nodes.

B. Effectiveness Evaluation

In this subsection, we assess the defensive efficacy of DShield against various attacks. Our investigation includes experiments devised to mitigate 10 DLBAs and 2 CLBAs across 4 node classification datasets. The assessments involving the GCN victim model are detailed in Table I, Table II, and Table X, while the results for the GAT model are provided in Appendix A-C. The findings reveal three observations:

1) DShield exhibits a notable performance against most backdoor attacks, surpassing the efficacy of conventional defenses. For instance, it achieves an average ASR of 1.33%, significantly better than the second-best ASR of 54.47% when



(a) UGBA (b) GCBA

Fig. 7: DShield's performance across various poisoning rates on Cora dataset. The shading represents the standard deviation.

defending against UGBA on the Cora dataset. This superiority stems from DShield's precision in identifying nodes with structural or attribute distributions deviating from normal nodes, consequently mitigating their influence during training.

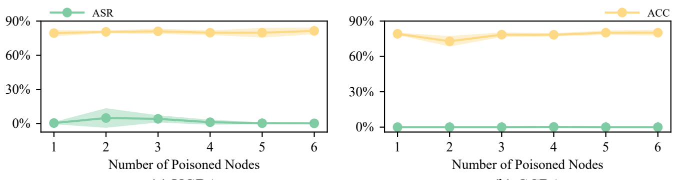
2) Although DShield demonstrates superior defense performance in most scenarios, a slight decline in model performance on normal nodes is observed. For instance, when countering SBA, the performance of the model trained with DShield on normal nodes degrades by approximately 3% compared to the model trained without any defenses. Since Eq. (39) degrades the performance on poisoned nodes during the backdoor-free training process, penalized poisoned nodes inevitably affect model performance on normal nodes through inherent connections between poisoned and normal nodes.

3) Notably, other defense mechanisms, except DShield, do not consistently degrade attack performance. When defending against EBA using the Prune method, the victim model suffers a higher ASR (average 100.00%) than no defense (average 71.81%). This counter-intuitive result occurs because these defense mechanisms remove normal nodes or edges, inadvertently amplifying the influence of poisoned nodes.

C. Robustness Evaluation

In this section, we investigate the performance of DShield under diverse experimental settings.

Poisoning Rate r_p. To assess DShield's performance under varying poisoning rates, we select 6 levels of r_p, i.e., $\{5, 10, 15, 20, 25, 30\}$, while keeping the number of labeled nodes constant. We deploy 6 different backdoor attacks on the 4 node classification datasets. The experimental results on the Cora dataset against UGBA and GCBA are shown in Fig. 7, with results against other attacks provided in Fig. 13. These results show that DShield maintains stability across varying r_p



(a) UGBA (b) GCBA

Fig. 8: The performance of DShield under different trigger sizes on the Cora dataset.

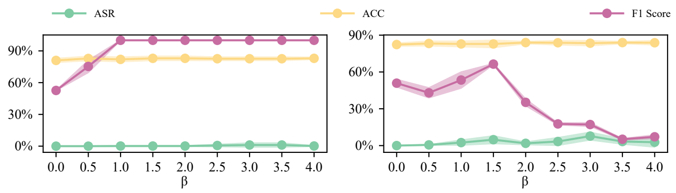
TABLE III: Robustness of DShield against 5 adaptive attacks (%) on the Cora and PubMed datasets.

Datasets	Attacks	*no-defense*		*no-poison*		DShield	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
Cora	UGBA+LGCB	$97.91_{\pm 0.93}$	$84.32_{\pm 1.19}$	$08.00_{\pm 0.21}$	$78.02_{\pm 5.00}$	$00.49_{\pm 4.16}$	$82.22_{\pm 6.72}$
	UGBA+GCBA	$88.07_{\pm 1.89}$	$83.83_{\pm 2.04}$	$08.00_{\pm 0.43}$	$78.15_{\pm 4.98}$	$03.94_{\pm 4.49}$	$81.97_{\pm 5.46}$
	GCBA+PerCBA	$74.78_{\pm 3.77}$	$83.82_{\pm 1.13}$	$08.24_{\pm 0.93}$	$77.53_{\pm 3.36}$	$08.49_{\pm 2.82}$	$80.99_{\pm 4.64}$
	AdaDA	$84.51_{\pm 2.61}$	$84.07_{\pm 0.52}$	$07.75_{\pm 0.64}$	$84.07_{\pm 1.29}$	$04.43_{\pm 3.13}$	$76.86_{\pm 0.26}$
	AdaCA	$89.85_{\pm 6.52}$	$82.59_{\pm 0.52}$	$09.47_{\pm 5.18}$	$84.69_{\pm 0.94}$	$00.74_{\pm 0.98}$	$81.23_{\pm 2.46}$
PubMed	UGBA+LGCB	$88.73_{\pm 5.16}$	$84.98_{\pm 0.26}$	$37.92_{\pm 1.71}$	$82.23_{\pm 1.97}$	$00.15_{\pm 0.27}$	$82.50_{\pm 0.74}$
	UGBA+GCBA	$91.89_{\pm 3.07}$	$85.42_{\pm 0.03}$	$39.61_{\pm 4.17}$	$83.66_{\pm 0.43}$	$02.92_{\pm 1.17}$	$82.51_{\pm 1.19}$
	GCBA+PerCBA	$89.01_{\pm 5.25}$	$84.53_{\pm 0.62}$	$39.52_{\pm 3.38}$	$83.49_{\pm 0.61}$	$13.39_{\pm 2.08}$	$81.28_{\pm 1.17}$
	AdaDA	$88.72_{\pm 3.34}$	$85.22_{\pm 0.04}$	$40.23_{\pm 0.72}$	$83.17_{\pm 3.57}$	$32.61_{\pm 8.89}$	$79.17_{\pm 1.47}$
	AdaCA	$87.88_{\pm 5.12}$	$85.13_{\pm 0.31}$	$03.82_{\pm 1.68}$	$85.39_{\pm 0.13}$	$05.68_{\pm 0.79}$	$82.60_{\pm 1.87}$

by setting the minimum cluster size to at least 50% of nodes with the same label.

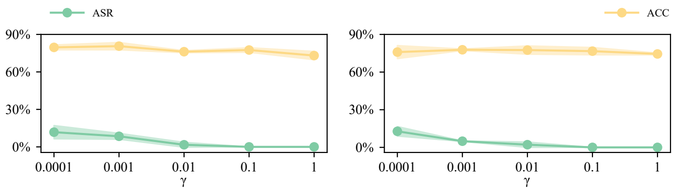
Trigger Size. Additionally, we assess the robustness of DShield across various trigger sizes on the Cora dataset. We analyze 6 different backdoor attacks, where UGBA, SBA, and GTA attach sub-graphs to poisoned nodes as triggers, and LGCB, GCBA, and PerCBA directly modify attributes of the poisoned nodes to use as triggers. Thus, we vary the size of the trigger sub-graphs for the former 3 attacks (i.e., $\{1, 2, 3, 4, 5, 6\}$) and the number of perturbed attributes for the latter 3 attacks (i.e., $\{100, 150, 200, 250, 300, 350\}$). Experimental results against UGBA and GCBA attacks on the Cora dataset, depicted in Fig. 8, demonstrate that DShield maintains robustness across different trigger sizes without assuming specific sizes as PXGBD and GXGBD do. Its effectiveness lies in identifying nodes with disparate distributions of latent representations or altered attributes, regardless of trigger size.

Adaptive Attacks. To assess the DShield's performance against more powerful backdoor attacks, we introduce 5 adaptive attacks: UGBA+LGCB, UGBA+GCBA, GCBA+PerCBA, AdaDA (Adaptive DLBA), and AdaCA (Adaptive CLBA). The first 3 attacks are multi-target attacks generating triggers for half of the poisoned nodes: UGBA+LGCB uses two dirty-label attacks (UGBA and LGCB), UGBA+GCBA combines dirty- and clean-label attacks (UGBA and GCBA), and GCBA+PerCBA employs two clean-label attacks (GCBA and PerCBA). The target label y^t for the first attack (e.g., UGBA in UGBA+LGCB) is set to 2, while for the second attack (e.g., LGCB in UGBA+LGCB), it is set to 3. Additionally, AdaDA and AdaCA incorporate two regularized losses to enhance the attack stealthiness, aiming to circumvent the self-supervised learning framework and UMAP used by DShield. Detailed descriptions of these attacks are provided in Appendix A-D.



(a) UGBA (b) GCBA

Fig. 9: The impact of parameter β on the Cora dataset when defending against UGBA.



(a) UGBA (b) GCBA

Fig. 10: The impact of parameter γ on the Cora dataset when defending against GCBA.

Table III presents results on the Cora and PubMed datasets and experimental results on other datasets are listed in Table XII, where *no-poison* indicates scenarios where no triggers are injected into the training graph. Key insights include:

1) DShield can simultaneously defend against dirty- and clean-label attacks by using the threshold β and the median absolute deviation to identify anomalous labels as target labels, effectively defending against attacks aimed at multiple labels.

2) The efficacy of DShield stems from the execution of the self-supervised learning framework and UMAP on the manipulated graph. Consequently, during trigger generation, the adversary lacks access to parameters utilized by DShield. Furthermore, disparities in parameters between the adversary and DShield hinder the crafted triggers from evading detection.

D. Sensitivity Analysis

In this subsection, we conduct the sensitivity analysis to assess DShield's performance under various hyper-parameter configurations. Eq. (33) uses the threshold β to detect anomalous nodes, while Eq. (39) utilizes γ to adjust the classification loss for normal and poisoned nodes. Accordingly, we assess the impact of different values of β and γ, specifically choosing $\beta \in \{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$ and $\gamma_2 \in \{1, 0.1, 0.01, 10^{-3}, 10^{-4}\}$ on the Cora dataset when defending against UGBA and GCBA attacks. Since β determines the number of identified poisoned nodes, we also report the F1 Score metric for identification. Experimental results are presented in Fig. 9 and 10, yielding the following observations:

1) The parameter β significantly affects identification precision. For example, in the UGBA attack, as β increases, the F1 Score gradually improves. However, if β becomes excessively large, the F1 Score starts to decline. This occurs because an overly large β regards no suspicious score as an outlier value.

2) Increasing the parameter γ results in a decline in the victim model's performance on poisoned nodes. This observation confirms the efficacy of the discrepancy matrix, which

TABLE IV: Ablation Studies of DShield against UGBA and GCBA (%). The victim model is GCN.

Datasets	Attacks	*w/o* $\mathcal{L}_r$		*w/o* Adaptive Dropping		*w/o* UMAP	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
Cora	UGBA	63.84$_{\pm 4.70}$	83.82$_{\pm 0.57}$	63.84$_{\pm 4.70}$	83.82$_{\pm 0.57}$	00.12$_{\pm 0.21}$	82.59$_{\pm 1.85}$
	GCBA	00.25$_{\pm 0.43}$	81.36$_{\pm 2.14}$	02.09$_{\pm 3.62}$	82.47$_{\pm 1.13}$	74.54$_{\pm 3.65}$	83.21$_{\pm 1.54}$
PubMed	UGBA	63.04$_{\pm 2.37}$	84.91$_{\pm 0.08}$	61.24$_{\pm 6.13}$	84.93$_{\pm 0.13}$	00.20$_{\pm 0.19}$	84.41$_{\pm 0.34}$
	GCBA	01.07$_{\pm 0.60}$	84.34$_{\pm 0.47}$	00.95$_{\pm 1.21}$	84.46$_{\pm 0.64}$	15.91$_{\pm 1.56}$	85.29$_{\pm 0.33}$
Flickr	UGBA	07.31$_{\pm 1.23}$	51.39$_{\pm 0.06}$	06.84$_{\pm 5.23}$	51.02$_{\pm 0.50}$	03.00$_{\pm 5.20}$	51.07$_{\pm 0.15}$
	GCBA	00.00$_{\pm 0.00}$	50.68$_{\pm 0.25}$	00.00$_{\pm 0.00}$	50.77$_{\pm 0.29}$	00.26$_{\pm 0.08}$	50.70$_{\pm 0.27}$
OGBN -arXiv	UGBA	74.05$_{\pm 2.75}$	60.19$_{\pm 0.78}$	96.21$_{\pm 4.76}$	59.42$_{\pm 0.34}$	00.00$_{\pm 0.00}$	59.41$_{\pm 0.42}$
	GCBA	00.20$_{\pm 0.24}$	60.06$_{\pm 0.55}$	01.09$_{\pm 1.38}$	60.08$_{\pm 0.60}$	04.87$_{\pm 1.23}$	60.24$_{\pm 0.54}$

adeptly detects distribution disparities in latent representations or attributes between poisoned and normal nodes, thereby enabling precise identification of poisoned nodes.

E. Ablation Studies

In this subsection, we assess the impact of DShield's components, including the reconstruction loss $\mathcal{L}_r$, "Adaptive Dropping", and UMAP. "Adaptive Dropping" involves selectively removing attributes and edges based on attribute importance and node similarities connected by an edge. Our evaluations focus on defending against UGBA and GCBA attacks across four node classification datasets. Table IV presents experimental results, where "*w/o* $\mathcal{L}_r$" denotes the absence of reconstruction loss during self-supervised training, "*w/o* Adaptive Dropping" indicates random removal of attributes and edges, and "*w/o* UMAP" signifies no reduction in dimensionality for important attributes when constructing the attribute importance matrix.

The loss $\mathcal{L}_r$ and "Adaptive Dropping" are crucial for defending against DLBAs, whereas UMAP is effective against CLBAs. For example, on the Cora dataset, in the absence of $\mathcal{L}_r$, UGBA achieves an average ASR of 63.84%, while GCBA achieves only 0.25%. This disparity arises because $\mathcal{L}_r$ and "Adaptive Dropping" operate within the self-supervised learning framework, whereas UMAP contributes to the attribute importance matrix. Moreover, UMAP's impact diminishes when the attribute dimensionality is insufficiently large.

F. Defending against Attacks on Graph Classification Tasks

In this subsection, we assess DShield's performance against backdoor attacks on graph classification tasks. We adapt DShield as G-DShield to suit this purpose with slight modifications. To inject triggers into the original graphs, we utilize SBAG [6] and adapt SBA [50], EBA [43], and GCBA [49] into G-SBA, G-EBA, and G-GCBA, respectively. All evaluations are conducted on 3 graph classification datasets, i.e., ENZYMES, PROTEINS, and MNIST. Detail descriptions of these attacks and G-DShield are provided in Appendix A-E. Experimental results in Table V and Table XI show that semantic drift and attribute over-emphasis also occur in graph classification tasks and the defense efficiency is not limited to node classification backdoor attacks.

TABLE V: DShield against backdoor attacks on graph classification tasks (%). The victim model is GCN.

Datasets	Defenses	G-SBA		G-EBA		G-GCBA	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
ENZYMES	*no-defense*	100.0$_{\pm 0.00}$	32.67$_{\pm 0.94}$	100.0$_{\pm 0.00}$	27.50$_{\pm 3.54}$	100.0$_{\pm 0.00}$	23.33$_{\pm 0.00}$
	G-DShield	01.67$_{\pm 2.35}$	35.00$_{\pm 2.36}$	02.50$_{\pm 3.54}$	31.67$_{\pm 2.35}$	00.00$_{\pm 0.00}$	30.84$_{\pm 1.18}$
PROTEINS	*no-defense*	100.0$_{\pm 0.00}$	66.67$_{\pm 1.27}$	99.56$_{\pm 0.63}$	71.62$_{\pm 3.18}$	99.11$_{\pm 0.00}$	73.43$_{\pm 5.73}$
	G-DShield	00.00$_{\pm 0.00}$	74.63$_{\pm 4.04}$	04.29$_{\pm 6.06}$	74.33$_{\pm 3.19}$	00.98$_{\pm 1.39}$	71.17$_{\pm 1.27}$
MNIST	*no-defense*	100.0$_{\pm 0.00}$	36.33$_{\pm 0.67}$	100.0$_{\pm 0.00}$	36.89$_{\pm 0.02}$	100.0$_{\pm 0.00}$	38.34$_{\pm 0.66}$
	G-DShield	00.00$_{\pm 0.00}$	38.86$_{\pm 0.37}$	01.87$_{\pm 2.64}$	39.77$_{\pm 1.65}$	00.00$_{\pm 0.00}$	40.57$_{\pm 1.89}$

TABLE VI: The execution time of various defenses against UGBA and GCBA on the Cora and PubMed datasets (s).

Defenses	Cora		PubMed	
	UGBA	GCBA	UGBA	GCBA
Prune	0000.92$_{\pm 0.01}$	0000.89$_{\pm 0.10}$	0000.98$_{\pm 0.02}$	0000.91$_{\pm 0.13}$
Isolate	0000.87$_{\pm 0.09}$	0000.95$_{\pm 0.10}$	0000.93$_{\pm 0.08}$	0000.97$_{\pm 0.02}$
PXGBD	0120.14$_{\pm 0.25}$	0118.89$_{\pm 1.85}$	0908.47$_{\pm 2.60}$	0864.93$_{\pm 8.14}$
GXGBD	0209.72$_{\pm 1.37}$	0202.13$_{\pm 6.19}$	1564.67$_{\pm 5.90}$	1584.93$_{\pm 3.15}$
ABL	0001.08$_{\pm 0.03}$	0001.07$_{\pm 0.04}$	0002.86$_{\pm 0.20}$	0002.61$_{\pm 0.09}$
RS	0001.49$_{\pm 0.02}$	0001.49$_{\pm 0.01}$	0001.62$_{\pm 0.02}$	0001.62$_{\pm 0.01}$
DShield	0064.38$_{\pm 0.46}$	0064.51$_{\pm 1.13}$	0178.84$_{\pm 7.64}$	0193.21$_{\pm 4.55}$
DShield-**cuML**	0017.98$_{\pm 3.73}$	0020.41$_{\pm 0.67}$	0046.18$_{\pm 2.19}$	0044.42$_{\pm 0.29}$

VII. Discussion

Effect on Benign Performance. Although DShield effectively alleviates the influence of poisoned nodes, as demonstrated in Section VI, it is essential to acknowledge that this mitigation comes at the cost of a concomitant degradation in the performance of the victim model on normal nodes. This degradation is an intrinsic consequence, primarily stemming from the pre-existing connections between poisoned and normal nodes before the adversary's injection of triggers. Such connections cannot be completely severed without jeopardizing the accurate predictions for the respective normal nodes.

Defense Execution Time. The execution time of various defenses is shown in Table VI and Table XIII. It demonstrates that DShield achieves optimal defense performance while consuming a moderate amount of time. To further enhance detection speed, we utilize the cuML library [28] for implementing the UMAP and HDBSCAN algorithms, referred to as DShield-cuML, which enables faster detection. Additionally, efforts are underway to detect multiple labels in parallel to further expedite the defense process.

VIII. Conclusion

In this paper, we proposed DShield, a comprehensive and resilient framework based on discrepancy learning to safeguard GNNs against backdoor attacks. We devised a self-supervised learning framework and attribute importance analysis technique to construct a discrepancy matrix for semantic and attribute importance differences between normal and poisoned nodes. Subsequently, this matrix facilitates the identification of poisoned nodes, followed by the implementation of a backdoor-free training mechanism to train a GNN model devoid of backdoors on node and graph classification tasks. Extensive experimental results highlighted DShield's effectiveness in significantly mitigating the negative effects of poisoned nodes, while also maintaining the model's performance on normal nodes. Future research endeavors will focus on broadening its mechanisms to counter backdoor attacks on other domains.

Acknowledgment

This work is supported by the National Key R&D Program of China under Grant 2022YFB3103500, the National Natural Science Foundation of China under Grants 62325604, 62276271, 62072062, 62222201, U20A20176, and U23A20304, Beijing Nova Program with No. 20220484174, Beijing Natural Science Foundation with Nos. M23020.

References

[1] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

[2] R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *PAKDD*, volume 7819 of *Lecture Notes in Computer Science*, pages 160–172. Springer, 2013.

[3] D. Chen, Y. Lin, G. Zhao, X. Ren, P. Li, J. Zhou, and X. Sun. Topology-imbalance learning for semi-supervised node classification. In *NeurIPS*, pages 29885–29897, 2021.

[4] L. Chen, Q. Peng, J. Li, Y. Liu, J. Chen, Y. Li, and Z. Zheng. Neighboring backdoor attacks on graph convolutional network. *CoRR*, abs/2201.06202, 2022.

[5] E. Dai, M. Lin, X. Zhang, and S. Wang. Unnoticeable backdoor attacks on graph neural networks. In *WWW*, pages 2263–2273. ACM, 2023.

[6] J. Dai, Z. Xiong, and C. Cao. A semantic backdoor attack against graph convolutional networks. *Neurocomputing*, 600:128133, 2024.

[7] I. Dokmanic, R. Parhizkar, J. Ranieri, and M. Vetterli. Euclidean distance matrices: Essential theory, algorithms, and applications. *IEEE Signal Process. Mag.*, 32(6):12–30, 2015.

[8] Z. Guan, M. Du, and N. Liu. XGBD: explanation-guided graph backdoor detection. In *ECAI*, volume 372, pages 932–939. IOS Press, 2023.

[9] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang. Graphmae: Self-supervised masked graph autoencoders. In *KDD*, pages 594–604. ACM, 2022.

[10] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

[11] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren. Backdoor defense via decoupling the training process. In *ICLR*. OpenReview.net, 2022.

[12] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Trans. Knowl. Data Eng.*, 35(7):6968–6972, 2023.

[13] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 604–613. ACM, 1998.

[15] S. Ivanov, S. Sviridov, and E. Burnaev. Understanding isomorphism bias in graph data sets. *CoRR*, abs/1910.12091, 2019.

[16] B. Jiang and Z. Li. Defending against backdoor attack on graph nerual network by explainability. *CoRR*, abs/2209.02902, 2022.

[17] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017.

[18] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *CVPR*, pages 298–307. Computer Vision Foundation / IEEE, 2020.

[19] J. B. Lee, R. A. Rossi, and X. Kong. Graph classification using structural attention. In *SIGKDD*, pages 1666–1674. ACM, 2018.

[20] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma. Anti-backdoor learning: Training clean models on poisoned data. In *NeurIPS*, pages 14900–14912, 2021.

[21] K. Liang, L. Meng, M. Liu, Y. Liu, W. Tu, S. Wang, S. Zhou, X. Liu, F. Sun, and K. He. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. *IEEE TPAMI*, 2024.

[22] K. Liang, L. Meng, S. Zhou, W. Tu, S. Wang, Y. Liu, M. Liu, L. Zhao, X. Dong, and X. Liu. Mines: Message intercommunication for inductive relation reasoning over neighbor-enhanced subgraphs. In *AAAI*, pages 10645–10653, 2024.

[23] M. Liu, K. Liang, Y. Zhao, W. Tu, S. Zhou, X. Gan, X. Liu, and K. He. Self-supervised temporal graph learning with temporal and structural intensity alignment. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[24] Y. Liu, X. Yang, S. Zhou, X. Liu, Z. Wang, K. Liang, W. Tu, L. Li, J. Duan, and C. Chen. Hard sample aware network for contrastive deep graph clustering. In *Proc. of AAAI*, volume 37, pages 8914–8922, 2023.

[25] Y. Liu, S. Zhu, J. Xia, Y. Ma, J. Ma, W. Zhong, X. Liu, S. Yu, and K. Zhang. End-to-end learnable clustering for intent learning in recommendation. In *NeurIPS*, 2024.

[26] L. McInnes and J. Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018.

[27] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5425–5434. IEEE Computer Society, 2017.

[28] S. Raschka, J. Patterson, and C. Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*, 2020.

[29] E. Rosenfeld, E. Winston, P. Ravikumar, and J. Z. Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *ICML*, volume 119, pages 8230–8241. PMLR, 2020.

[30] P. J. Rousseeuw and M. Hubert. Anomaly detection by robust statistics. *WIREs Data Mining Knowl. Discov.*, 8(2), 2018.

[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359, 2020.

[32] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

[33] M. Shen, C. Li, Q. Li, H. Lu, L. Zhu, and K. Xu. Transferability of white-box perturbations: Query-Efficient adversarial attacks against commercial DNN services. In *USENIX Security*, pages 2991–3008, 2024.

[34] M. Shen, C. Li, H. Yu, Q. Li, L. Zhu, and K. Xu. Decision-based query efficient adversarial attack via adaptive boundary learning. *IEEE Trans. Dependable Secur. Comput.*, 21(4):1740–1753, 2024.

[35] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[36] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*. OpenReview.net, 2018.

[37] K. Wang, H. Deng, Y. Xu, Z. Liu, and Y. Fang. Multi-target label backdoor attacks on graph neural networks. *Pattern Recognit.*, 152:110449, 2024.

[38] Z. Xi, R. Pang, S. Ji, and T. Wang. Graph backdoor. In *USENIX Security*, pages 1523–1540, 2021.

[39] T. Xiao, X. Wang, A. A. Efros, and T. Darrell. What should not be contrastive in contrastive learning. In *ICLR*. OpenReview.net, 2021.

[40] X. Xing, M. Xu, Y. Bai, and D. Yang. A clean-label graph backdoor attack method in node classification task. *CoRR*, abs/2401.00163, 2024.

[41] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang. Infogcl: Information-aware graph contrastive learning. In *NeurIPS*, pages 30414–30425, 2021.

[42] J. Xu and S. Picek. Poster: Multi-target & multi-trigger backdoor attacks on graph neural networks. In *CCS*, pages 3570–3572. ACM, 2023.

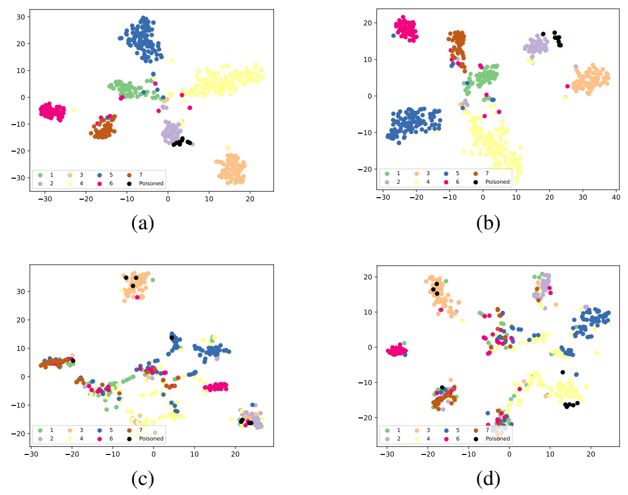


Fig. 11: Semantic Drift: t-SNE visualization of latent representations of nodes within the manipulated graph generated by SBA [50] and EBA [43] on the Cora dataset. Fig. (a) & (b): GNNs trained with the manipulated label information using the semi-supervised learning paradigm. Fig. (c) & (d): GNNs trained with the manipulated label information using self-supervised learning paradigm. Fig. (a) & (b): Manipulated graph generated by SBA [50] and EBA [43].

[43] J. Xu, M. Xue, and S. Picek. Explainability-based backdoor attacks against graph neural networks. In *WiseML@WiSec 2021*, pages 31–36. ACM, 2021.

[44] S. Yang, B. G. Doan, P. Montague, O. Y. de Vel, T. Abraham, S. Camtepe, D. C. Ranasinghe, and S. S. Kanhere. Transferable graph backdoor attack. In *RAID*, pages 321–332. ACM, 2022.

[45] X. Yang, G. Li, C. Zhang, M. Han, and W. Yang. Percba: Persistent clean-label backdoor attacks on semi-supervised graph node classification. In *IJCAI (AISafety-SafeRL)*, volume 3505 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[46] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. In *NeurIPS*, 2020.

[47] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. K. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *ICLR*. OpenReview.net, 2020.

[48] Y. Zeng, M. Pan, H. A. Just, L. Lyu, M. Qiu, and R. Jia. Narcissus: A practical clean-label backdoor attack with limited information. In *CCS*, pages 771–785. ACM, 2023.

[49] H. Zhang, J. Chen, L. Lin, J. Jia, and D. Wu. Graph contrastive backdoor attacks. In *ICML*, volume 202, pages 40888–40910. PMLR, 2023.

[50] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong. Backdoor attacks to graph neural networks. In *SACMAT*, pages 15–26. ACM, 2021.

[51] Z. Zhang, M. Lin, E. Dai, and S. Wang. Rethinking graph backdoor attacks: A distribution-preserving perspective. In *KDD*, pages 4386–4397. ACM, 2024.

[52] Z. Zhao, R. Wang, Z. Wang, F. Nie, and X. Li. Graph joint representation clustering via penalized graph contrastive learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2023.

[53] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang. Graph contrastive learning with adaptive augmentation. In *WWW*, pages 2069–2080. ACM / IW3C2, 2021.

Appendix A
Appendix

A. Additional Details of Node Classification Attacks

In this subsection, we will provide additional details of backdoor attacks on node classification tasks.

TABLE VII: Dataset Statistics.

Datasets	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	F	C	$\lvert\mathcal{V}^l\rvert$ or $\lvert\mathcal{G}^l\rvert$	$\lvert\mathcal{V}^p\rvert$ or $\lvert\mathcal{G}^p\rvert$
Cora	1	2,708	5,429	1,443	7	541	10
PubMed	1	19,717	44,338	500	3	3,943	40
Flickr	1	89,250	899,756	500	7	17,850	80
OGBN-arXiv	1	169,343	1,166,243	128	40	29,091	160
ENZYMES	600	32	124	3	2	480	60
PROTEINS	1,113	39	145	3	2	668	222
MNIST	70,000	75	1,393	1	10	42,000	14,000

Node Selection. Nodes with representative features and structures are likely to lead to successful backdoor attacks when triggers are attached. Therefore, existing studies select diverse and representative nodes as poisoned nodes to enforce the victim model to predict these nodes with triggers as the target label y^t. Dai *et al.* [5] introduce 5 approaches for choosing poisoned nodes: "Random", "Degree", "Clustering Coefficient", "Cluster", and "Cluster&Degree". The "Random" method arbitrarily selects nodes. The "Degree" method prioritizes nodes with lower degrees to minimize disruption to model performance on normal nodes. The "Clustering Coefficient" method chooses nodes with high clustering coefficients. The "Cluster" method identifies representative nodes through clustering analysis. The "Cluster&Degree" method combines clustering analysis and degree prioritization to select representative nodes with relatively lower degrees. Since the "Cluster&Degree" method has shown superior performance, we use it as the default approach for poisoned node selection.

Trigger Generation. Backdoor attacks employ various forms of triggers and methods for their generation. For instance, Dai *et al.* [5] use small subgraphs as triggers and develop a trigger generator to produce them. Xu *et al.* [43] use attributes as triggers, leveraging GNN explainability to identify and modify the least representative attributes for trigger injection.

Model Training. The adversary trains the GNN model using the standard semi-supervised learning approach on the manipulated graph.

Trigger Injection. The adversary follows the same trigger generation process to produce and inject triggers into the testing graph.

B. Detailed Settings for Case Studies

In Section IV, we conduct case studies to reveal ubiquitous discrepancies in semantic information and attribute importance between normal and poisoned nodes. This section elaborates comprehensively on the setup of attacks, methodologies for self-supervised training, analysis of attribute importance, parameters for t-SNE visualization, and the presentation of experimental results for SBA and EBA attacks.

Attack Setups. Seven distinct backdoor attacks are performed on the Cora dataset, including SBA [50], GTA [38], EBA [43], GB-FGSM [4], UGBA [5], GCBA [49], and PerCBA [45]. SBA, GTA, EBA, and UGBA utilize sub-graphs as triggers, each comprising 3 nodes. Conversely, GB-FGSM, GCBA, and PerCBA employ single nodes as triggers and directly modify attributes of poisoned nodes, with trigger sizes set to 200. The

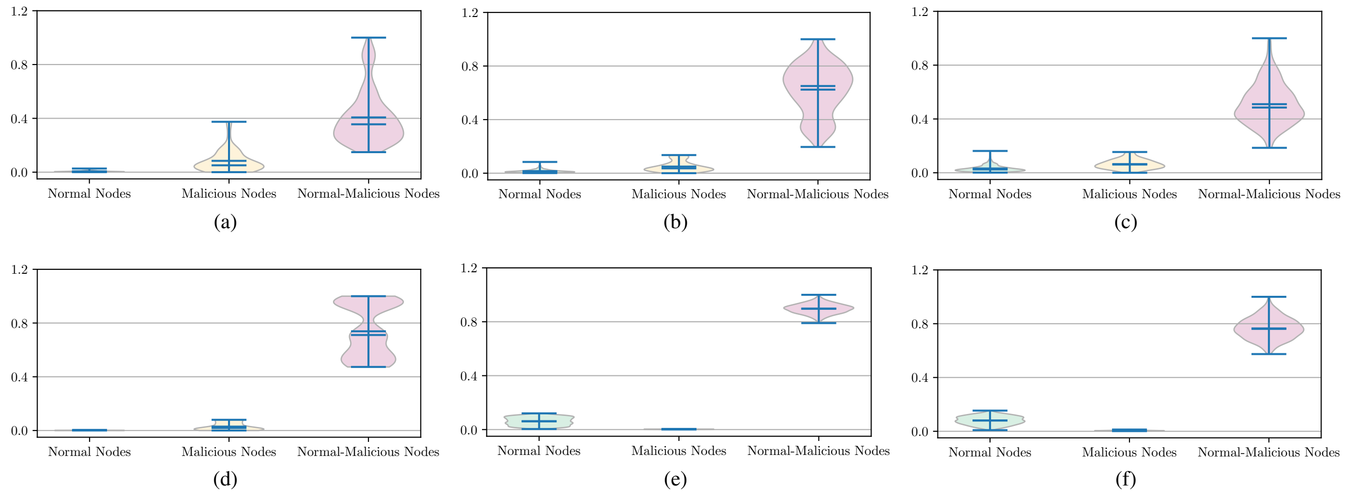


Fig. 12: Figs. (a)–(d) present the quantitative analysis of *semantic drift* on the Cora dataset for backdoored graphs generated by GTA, GB-FGSM, UGBA, and SBA attacks, while Figs. (e) and (f) illustrate the analysis of *attribute over-emphasis* for graphs generated by GCBA and PerCBA attacks. The visualized differences within the elements of the semantic discrepancy matrix $\boldsymbol{D}_s^y$ and the attribute importance discrepancy matrix $\boldsymbol{D}_a^y$ among normal nodes, malicious nodes, and between normal and malicious nodes, demonstrate a significant distance between normal and malicious nodes.

Cora dataset's data splits remain consistent with those detailed in Section VI, with 10 poisoned nodes. Additionally, the victim model employed is a two-layer GCN model.

Self-supervised Training Setups. Following the removal of all label information, self-supervised learning is conducted on the unlabelled manipulated graph using GraphCL [46]. Two distinct data augmentations, i.e., edge perturbation and attribute masking, are employed to generate varied perspectives of the original graph. Both the edge dropout ratio and attribute masking ratio are set to 0.2. Furthermore, the Adam optimizer with a learning rate of 10^{-3} is utilized, with training continuing until convergence.

Attribute Importance Analysis. Attribute importance is assessed utilizing Eq. (9), followed by the derivation of the mask matrix $\boldsymbol{W}$, delineating the positions of attributes that positively contribute to prediction results. Consequently, the truly positive attributes influencing victim model predictions are obtained as $\widetilde{\boldsymbol{X}}' = \widetilde{\boldsymbol{X}} \circ \boldsymbol{W}$, which is subsequently visualized using t-SNE.

t-SNE Visualization Settings. The latent representations of labeled nodes within the manipulated graph are visualized by treating the output of the first layer $\widetilde{f}(\cdot;\phi_1)$ of the victim model as node representations. Additionally, for visualizing attribute importance, $\widetilde{\boldsymbol{X}}'$ is employed as inputs. Matplotlib [13] is utilized to generate two-dimensional embeddings, with all labeled nodes utilized for enhanced visualization clarity.

Visualizations of SBA and EBA. The results of these two attacks are depicted in Fig. 11. These visualizations distinctly illustrate the semantic discrepancies between normal nodes and those manipulated by these attacks.

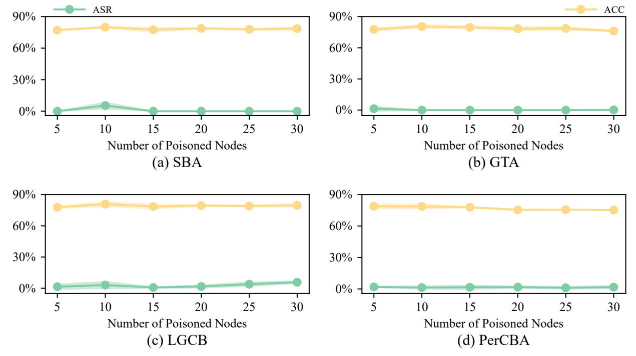


Fig. 13: DShield's performance across various poisoning rates on Cora dataset. The shading represents the standard deviation.

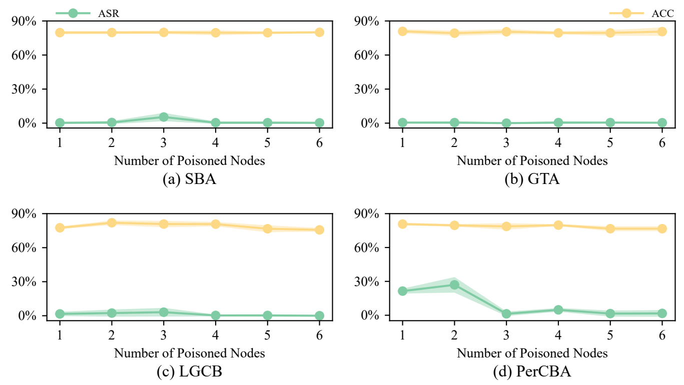


Fig. 14: DShield's performance against 3 DLBAs and 1 CLBA under different trigger sizes on the Cora dataset.

C. Additional Effectiveness Evaluations

Beyond assessing the effectiveness of DShield with the GCN model, we also investigate its performance when employed with GAT on the four node classification datasets. The experimental results are presented in Table VIII and Table IX.

TABLE VIII: Comparison of DShield's effectiveness with state-of-the-art defenses for DLBAs (%). The victim model is GAT.

Datasets	Defenses	SBA [50]		GTA [38]		EBA [43]		GB-FGSM [4]		LGCB [4]		UGBA [5]		TRAP [44]	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
Cora	*no-defense*	57.20±6.78	82.34±2.17	89.49±9.13	82.96±0.98	99.45±0.78	**84.20**±1.40	63.10±0.52	**83.82**±0.77	71.96±4.70	82.10±2.80	99.26±1.05	**84.94**±0.22	18.08±2.56	79.75±1.30
	Prune [5]	10.83±4.40	82.71±1.40	13.90±4.06	**83.95**±1.19	100.0±0.00	78.52±1.96	55.54±0.78	81.97±2.26	52.03±4.63	**82.84**±2.41	35.79±0.52	81.97±1.54	12.06±3.14	81.11±3.03
	Isolate [5]	**03.95**±1.05	38.27±2.99	09.84±2.80	77.29±1.19	97.05±5.12	81.85±2.22	51.78±2.79	83.58±1.13	47.35±0.93	**82.84**±0.85	52.98±5.98	82.34±1.75	13.99±3.18	68.89±1.96
	PXGBD [8]	27.18±5.73	83.58±0.43	97.11±3.57	80.62±0.77	46.94±3.55	80.62±1.30	55.19±0.23	80.74±0.74	71.59±5.73	80.86±2.04	82.66±1.56	81.36±2.10	14.76±0.52	79.88±0.93
	GXGBD [8]	46.63±2.43	82.96±1.93	79.12±1.25	82.10±0.77	78.36±0.86	70.25±4.06	86.53±1.82	82.59±2.06	71.96±4.70	82.22±0.37	99.26±1.05	80.86±3.65	16.85±2.46	81.11±0.37
	ABL [20]	71.98±1.53	**84.82**±2.57	42.51±1.67	82.59±0.98	90.59±4.44	83.83±1.50	43.93±1.07	81.73±1.54	80.45±3.13	82.59±2.60	10.83±4.40	84.57±1.41	15.13±2.24	**82.59**±1.96
	RS [49]	06.64±1.33	71.11±3.76	54.71±1.18	70.62±2.99	78.78±2.87	72.22±2.90	61.38±3.14	71.36±3.52	79.95±7.60	71.11±3.39	83.27±3.58	71.85±0.74	17.34±2.96	69.51±2.04
	DShield	04.82±0.63	81.73±0.77	**00.00**±0.00	81.85±1.28	**00.00**±0.00	82.47±1.40	**07.61**±1.28	82.59±1.28	**02.71**±4.37	81.48±2.43	**00.00**±0.00	83.21±0.77	**10.82**±3.70	82.22±1.70
PubMed	*no-defense*	50.61±5.02	83.88±0.49	39.99±1.72	84.10±0.24	90.54±1.47	83.52±0.50	96.45±1.07	83.54±0.18	100.0±0.00	83.23±0.33	98.73±1.76	84.27±0.60	47.52±0.57	83.64±0.40
	Prune [5]	46.73±7.07	**84.34**±0.15	47.12±0.41	84.01±0.67	100.0±0.00	82.48±0.90	70.74±0.00	83.99±0.33	70.82±0.04	83.97±0.29	66.95±4.95	**84.30**±0.51	43.08±0.47	82.73±0.11
	Isolate [5]	39.01±1.56	81.16±0.46	46.37±1.29	83.07±0.23	99.88±0.18	83.51±0.79	70.08±0.00	**84.20**±0.11	70.72±0.54	83.72±0.22	69.49±1.24	83.80±0.32	**41.79**±0.43	83.54±0.32
	PXGBD [8]	57.05±9.68	83.83±0.37	85.12±2.26	84.17±0.42	36.83±0.10	83.41±0.50	95.08±2.22	83.82±0.71	99.30±1.00	83.64±0.54	100.0±0.00	83.97±0.15	46.60±0.86	83.84±0.25
	GXGBD [8]	51.40±3.70	84.00±0.51	82.50±3.46	83.49±0.25	97.38±2.91	82.24±0.07	95.34±6.60	83.69±0.25	98.18±2.30	83.89±0.18	96.71±4.16	83.71±0.28	47.87±0.14	83.69±0.54
	ABL [20]	65.32±5.66	84.11±0.37	91.23±3.44	84.30±0.18	91.10±2.83	83.82±0.29	97.34±3.76	83.64±0.18	46.66±4.73	83.67±0.22	96.31±3.81	83.71±0.44	47.60±1.11	**83.87**±0.57
	RS [49]	32.31±1.58	75.19±0.51	93.36±0.22	74.18±1.29	95.62±1.18	73.57±1.15	100.0±0.00	77.26±5.72	96.77±2.50	78.31±5.93	98.41±1.11	75.04±1.49	44.88±7.96	78.34±5.31
	DShield	**19.08**±1.64	81.89±0.40	**00.05**±0.00	**84.51**±0.23	**00.00**±0.00	**84.29**±0.10	**00.00**±0.00	83.66±0.40	**00.03**±0.04	**84.12**±0.13	**00.03**±0.03	83.68±0.38	46.14±0.97	82.63±0.98
Flickr	*no-defense*	00.10±0.13	49.12±0.16	89.08±1.63	49.37±0.17	85.62±1.11	49.13±0.02	93.01±1.31	49.32±0.15	97.62±0.06	49.33±0.59	91.90±2.21	49.28±0.27	04.22±0.15	49.34±0.01
	Prune [5]	00.01±0.01	49.40±0.15	94.78±0.26	49.23±0.07	100.0±0.00	42.28±0.44	98.62±1.12	**49.63**±0.22	97.87±1.36	49.50±0.28	94.23±2.77	**49.37**±0.01	00.46±0.25	47.97±0.23
	Isolate [5]	**00.00**±0.00	27.57±0.00	93.12±2.87	49.17±0.10	87.50±2.32	49.47±0.25	95.87±2.60	49.60±0.78	96.78±2.98	49.54±0.16	95.28±4.74	49.20±0.10	03.47±0.12	48.99±0.10
	PXGBD [8]	**00.00**±0.00	47.89±1.13	97.65±1.33	48.48±0.66	00.15±0.21	45.96±2.71	99.23±0.49	49.08±0.77	99.03±0.74	48.91±0.08	89.99±1.24	48.04±0.33	01.76±0.49	48.19±0.01
	GXGBD [8]	**00.00**±0.00	44.12±3.90	94.41±6.23	45.71±1.63	00.48±0.67	45.75±2.09	99.66±0.48	44.65±1.30	69.11±1.26	41.12±1.08	11.60±2.26	47.00±0.59	**00.00**±0.00	40.35±0.00
	ABL [20]	01.84±0.71	**49.57**±0.11	95.92±2.12	**49.50**±0.28	98.03±1.46	**49.48**±0.01	98.33±1.17	49.24±0.21	98.23±0.81	49.54±0.15	93.97±0.74	49.15±0.40	13.36±1.19	49.19±0.38
	RS [49]	**00.00**±0.00	40.35±0.00	**00.00**±0.00	40.35±0.00	**00.00**±0.00	40.35±0.00	**00.00**±0.00	40.35±0.00	**00.00**±0.00	40.35±0.00	**00.00**±0.00	40.35±0.00	**00.00**±0.00	40.35±0.00
	DShield	00.01±0.01	49.03±0.57	00.04±0.05	49.37±0.34	**00.00**±0.00	48.36±0.15	**00.00**±0.00	49.18±0.45	**00.00**±0.00	**49.67**±0.09	**00.00**±0.00	49.19±0.37	00.01±0.01	**49.68**±0.01
OGBN -arXiv	*no-defense*	21.43±1.41	61.16±0.01	80.87±0.71	61.30±0.21	99.46±0.71	61.34±0.27	80.55±0.71	61.30±0.21	01.21±0.71	61.04±0.16	50.68±0.20	59.53±1.00	04.15±0.71	61.17±0.03
	Prune [5]	21.43±0.71	61.21±0.08	09.24±0.71	61.18±0.04	99.36±0.71	60.99±0.23	75.77±1.41	61.21±0.08	01.35±0.93	61.31±0.22	44.87±7.10	61.16±0.35	05.39±0.71	61.18±0.04
	Isolate [5]	07.55±0.64	50.78±0.31	10.91±0.13	62.38±0.88	98.70±0.98	61.55±0.64	80.01±1.40	62.14±1.22	01.33±0.95	61.83±0.25	85.43±3.71	61.53±0.28	06.13±1.23	62.13±1.23
	PXGBD [8]	28.74±0.37	62.42±0.83	**04.20**±1.13	62.33±0.95	99.98±0.03	60.79±1.12	82.78±1.73	62.98±1.44	01.31±0.98	62.35±0.92	96.28±4.50	61.70±0.27	04.29±1.00	62.47±0.75
	GXGBD [8]	13.50±2.13	56.64±2.31	67.93±0.10	57.22±1.73	01.57±2.02	57.65±0.50	77.49±3.52	57.66±0.48	01.45±0.78	56.72±2.43	95.55±6.30	56.67±1.12	02.72±0.40	56.93±0.11
	ABL [20]	02.84±1.18	60.99±0.01	82.69±0.44	58.43±0.81	99.90±0.14	56.96±2.77	82.13±1.59	53.84±0.23	00.57±0.80	56.13±1.59	01.99±1.91	48.48±1.22	04.62±0.88	56.29±1.82
	RS [49]	01.24±0.33	54.14±0.20	04.84±0.23	55.45±2.19	55.00±1.41	53.71±0.41	47.70±0.42	55.20±1.14	04.66±0.49	54.58±0.59	66.23±1.74	52.28±3.22	02.39±0.86	55.47±2.16
	DShield	**00.00**±0.00	**64.62**±0.00	31.89±2.52	**65.49**±0.33	**00.99**±1.40	**65.09**±0.40	**32.01**±2.33	**64.17**±0.16	**00.00**±0.00	**64.68**±0.13	**00.43**±0.61	**64.76**±1.21	**01.01**±0.49	**64.53**±0.49

TABLE IX: Comparison of DShield's effectiveness with state-of-the-art defenses for 2 CLBAs (%). The victim model is GAT.

Defenses	Cora				PubMed				Flickr				OGBN-arXiv			
	GCBA [49]		PerCBA [45]		GCBA [49]		PerCBA [45]		GCBA [49]		PerCBA [45]		GCBA [49]		PerCBA [45]	
	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
no-defense	96.92±5.33	83.70±1.11	85.06±1.31	83.33±1.96	85.73±4.84	**84.30**±0.25	100.0±0.00	83.64±0.61	88.00±0.08	**49.46**±0.06	99.13±0.04	**49.56**±0.21	41.86±1.41	61.51±0.50	92.79±0.71	61.24±0.13
Prune [5]	99.88±0.21	81.97±2.04	95.54±5.70	83.45±0.77	100.0±0.00	83.76±0.00	97.52±2.30	83.26±0.50	90.23±1.85	48.33±0.84	91.97±0.52	48.15±1.13	52.95±0.71	61.47±0.45	87.11±2.12	61.44±0.40
Isolate [5]	17.18±2.84	73.58±2.04	82.04±5.93	72.22±0.64	100.0±0.00	83.38±0.47	99.47±0.75	83.01±0.36	89.51±1.78	49.26±0.34	97.06±0.10	49.16±0.06	34.08±1.31	62.25±1.06	92.77±1.75	62.09±1.29
PXGBD [8]	30.26±1.05	81.85±0.74	68.34±2.50	82.59±0.00	84.46±0.32	84.22±0.72	17.63±3.04	81.91±0.47	34.40±0.59	47.47±1.45	42.52±1.58	47.60±0.48	17.78±0.32	62.49±0.73	**00.00**±0.00	62.45±0.78
GXGBD [8]	37.33±2.02	82.34±1.90	33.77±5.48	81.48±1.70	85.81±0.13	84.10±0.76	03.99±1.04	81.31±1.32	00.13±0.18	43.01±3.75	06.25±1.77	40.35±0.00	02.28±1.03	57.68±0.45	03.01±4.24	57.19±1.15
ABL [20]	43.25±2.56	**84.32**±0.57	68.38±2.45	**83.95**±2.04	80.01±0.01	84.22±0.21	05.43±6.24	81.76±1.26	89.18±0.47	49.44±0.16	99.44±0.14	49.49±0.11	36.03±1.46	61.64±0.52	91.52±2.15	56.31±1.85
RS [49]	51.29±9.91	72.60±5.46	74.73±3.91	72.96±4.86	79.58±0.59	77.70±5.49	81.38±0.20	76.36±3.08	**00.00**±0.00	40.35±0.00	09.66±4.46	40.35±0.00	33.95±1.48	56.05±2.76	89.17±1.17	55.07±1.32
DShield	**02.21**±3.83	81.94±2.57	**00.62**±1.07	80.99±1.40	**00.00**±0.00	84.10±0.51	**00.02**±0.03	**84.44**±0.23	**00.00**±0.00	48.40±0.21	**00.01**±0.01	48.03±1.36	**00.00**±0.00	**64.04**±0.86	**00.00**±0.00	**65.49**±0.37

While aligning with the conclusions from Subsection VI-B, key observations arise:

In certain situations, PXGBD and GXGBD demonstrate effectiveness when defending against two CLBAs. For instance, when defending against GCBA with the GAT victim model, GXGBD achieves the second-lowest ASR at 2.28% on the OGBN-arXiv dataset. This observation likely stems from the fact that CLBAs primarily modify attributes of poisoned nodes, leading the victim model to rely solely on these altered

TABLE X: DShield's performance (%) against TLN, DPGBA, and MLGB attacks on the Cora, PubMed, and Flickr datasets.

Attacks	Defenses	Cora		PubMed		Flickr	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
TLN	*no-defense*	16.90$_{\pm 2.11}$	84.59$_{\pm 1.07}$	49.99$_{\pm 0.47}$	85.36$_{\pm 0.12}$	06.32$_{\pm 3.11}$	50.67$_{\pm 0.24}$
	DShield	14.76$_{\pm 1.08}$	83.70$_{\pm 1.48}$	40.97$_{\pm 1.25}$	80.53$_{\pm 1.58}$	05.38$_{\pm 1.96}$	51.17$_{\pm 0.08}$
DPGBA [51]	*no-defense*	96.18$_{\pm 5.40}$	84.57$_{\pm 0.57}$	87.43$_{\pm 1.68}$	85.19$_{\pm 0.18}$	19.93$_{\pm 3.66}$	50.76$_{\pm 0.15}$
	DShield	02.71$_{\pm 4.69}$	78.27$_{\pm 3.02}$	00.00$_{\pm 0.00}$	81.55$_{\pm 0.95}$	00.34$_{\pm 0.58}$	50.58$_{\pm 0.21}$
MLGB [37]	*no-defense*	14.51$_{\pm 4.26}$	84.69$_{\pm 0.21}$	77.48$_{\pm 2.43}$	85.10$_{\pm 0.11}$	07.54$_{\pm 0.23}$	50.79$_{\pm 0.07}$
	DShield	02.09$_{\pm 0.93}$	78.62$_{\pm 1.52}$	00.03$_{\pm 0.03}$	82.14$_{\pm 0.75}$	00.00$_{\pm 0.00}$	51.02$_{\pm 0.12}$

TABLE XI: DShield's performance (%) against SBAG on graph classification tasks.

Attacks	Defenses	PROTEINS		ENZYMES	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑
SBAG [6]	*no-defense*	97.32$_{\pm 2.36}$	62.46$_{\pm 3.17}$	99.44$_{\pm 0.96}$	25.56$_{\pm 0.18}$
	DShield	00.00$_{\pm 0.00}$	59.46$_{\pm 3.01}$	00.00$_{\pm 0.00}$	23.33$_{\pm 1.67}$

attributes for predictions. Consequently, the explanatory subgraphs generated by PXGBD and GXGBD remain small. Given an appropriately set threshold for the model's loss value, these defenses successfully identify the poisoned nodes.

D. Additional Robustness Evaluations

Adaptive Attacks. In Subsection VI-C, we evaluate DShield's performance against 5 adaptive attacks. UGBA+LGCB, UGBA+GCBA, and GCBA+PerCBA each combine two distinct backdoor attacks with different target labels. The target label y^t for the first attack (e.g., UGBA in UGBA+LGCB) is set to 2, while for the second attack (e.g., LGCB in UGBA+LGCB), it is set to 3. These experiments test DShield's effectiveness against simultaneous multiple attacks.

Additionally, we devise two adaptive attacks, AdaDA (Adaptive DLBA) and AdaCA (Adaptive CLBA), to evaluate DShield's performance when the adversary is aware of its inner mechanisms. AdaDA, derived from UGBA, uses a sub-graph as the trigger and includes a trigger generator to craft node attributes and structures. During trigger generator optimization, a self-supervised model is trained without label information, following the framework outlined in Subsection V-B. A regularized loss is introduced to minimize the distances between the latent representations of normal nodes with the target label and poisoned nodes. AdaCA, derived from GCBA, formulates a regularized loss to minimize the distances of low-dimensional embeddings generated by UMAP between normal nodes with the target label and poisoned nodes. The results in Subsection VI-C show that using two regularized losses fails to overcome DShield. This situation is akin to the chicken-and-egg problem: the adversary alternates between optimizing the trigger generator and the parameters of the self-supervised model or UMAP, while the defender trains the self-supervised model or UMAP using the manipulated graph produced by the finalized generator. This divergence prevents the adversary from accessing the parameters of the self-supervised model and UMAP learned by DShield.

TABLE XII: Robustness of DShield against 5 adaptive attacks (%) on the Flickr and OGBN-arXiv datasets. The victim model is GCN.

Datasets	Attacks	*no-defense*		*no-poison*		DShield	
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	ASR ↓	ACC ↑
Flickr	UGBA+LGCB	93.37$_{\pm 2.15}$	50.48$_{\pm 0.08}$	00.65$_{\pm 0.16}$	46.54$_{\pm 3.23}$	00.10$_{\pm 0.17}$	49.95$_{\pm 0.68}$
	UGBA+GCBA	57.66$_{\pm 6.26}$	50.46$_{\pm 0.49}$	03.17$_{\pm 0.55}$	49.44$_{\pm 1.38}$	00.00$_{\pm 0.00}$	50.37$_{\pm 0.14}$
	GCBA+PerCBA	47.07$_{\pm 5.60}$	50.84$_{\pm 0.25}$	04.67$_{\pm 0.98}$	48.77$_{\pm 0.38}$	00.22$_{\pm 0.29}$	50.26$_{\pm 0.15}$
	AdaDA	94.25$_{\pm 1.64}$	50.93$_{\pm 0.06}$	02.91$_{\pm 1.56}$	50.70$_{\pm 0.11}$	00.00$_{\pm 0.00}$	51.18$_{\pm 0.92}$
	AdaCA	68.29$_{\pm 5.81}$	50.75$_{\pm 0.31}$	03.03$_{\pm 4.55}$	50.51$_{\pm 0.15}$	05.85$_{\pm 0.42}$	51.69$_{\pm 0.21}$
OGBN -arXiv	UGBA+LGCB	66.43$_{\pm 3.85}$	59.64$_{\pm 0.01}$	01.80$_{\pm 0.48}$	56.86$_{\pm 0.58}$	02.18$_{\pm 0.88}$	59.56$_{\pm 0.31}$
	UGBA+GCBA	45.07$_{\pm 2.29}$	59.83$_{\pm 0.20}$	01.02$_{\pm 0.19}$	55.76$_{\pm 0.18}$	00.50$_{\pm 0.55}$	59.39$_{\pm 0.17}$
	GCBA+PerCBA	22.20$_{\pm 1.12}$	60.70$_{\pm 0.05}$	01.13$_{\pm 0.01}$	55.94$_{\pm 0.08}$	00.06$_{\pm 0.08}$	60.34$_{\pm 0.22}$
	AdaDA	86.67$_{\pm 2.70}$	60.16$_{\pm 0.20}$	00.87$_{\pm 0.11}$	60.46$_{\pm 0.11}$	05.89$_{\pm 0.60}$	60.56$_{\pm 0.64}$
	AdaCA	19.57$_{\pm 3.90}$	60.01$_{\pm 0.58}$	04.60$_{\pm 2.82}$	60.33$_{\pm 0.22}$	05.16$_{\pm 2.53}$	59.56$_{\pm 0.55}$

TABLE XIII: Execution time of defenses against UGBA and GCBA attacks on Flickr and OGBN-arXiv datasets (s).

Defenses	Flickr		OGBN-arXiv	
	UGBA	GCBA	UGBA	GCBA
Prune	0001.33$_{\pm 0.02}$	0001.26$_{\pm 0.03}$	0003.53$_{\pm 0.02}$	0003.49$_{\pm 0.01}$
Isolate	0001.34$_{\pm 0.02}$	0001.42$_{\pm 0.10}$	0003.53$_{\pm 0.01}$	0003.52$_{\pm 0.00}$
PXGBD	1405.16$_{\pm 2.07}$	1410.61$_{\pm 8.47}$	2039.15$_{\pm 3.71}$	2020.80$_{\pm 5.76}$
GXGBD	0732.94$_{\pm 6.86}$	0769.42$_{\pm 4.45}$	0782.09$_{\pm 4.37}$	0774.23$_{\pm 1.93}$
ABL	0029.78$_{\pm 0.30}$	0022.02$_{\pm 0.00}$	0104.46$_{\pm 0.57}$	0069.74$_{\pm 1.02}$
RS	0002.27$_{\pm 0.00}$	0002.35$_{\pm 0.07}$	0004.81$_{\pm 0.07}$	0004.75$_{\pm 0.03}$
DShield	1068.12$_{\pm 2.40}$	1051.70$_{\pm 0.90}$	1205.46$_{\pm 4.09}$	1229.70$_{\pm 3.10}$
DShield-**cuML**	0151.71$_{\pm 2.59}$	0150.17$_{\pm 0.83}$	0180.55$_{\pm 1.31}$	0178.83$_{\pm 0.95}$

E. Defending Against Attacks on Graph Classification Tasks

To evaluate DShield's performance against backdoor attacks on graph classification tasks, we first adapt DShield to G-DShield to suit graph classification tasks and then assess its performance against these attacks.

Backdoor Attacks on Graph Classification Tasks. To generate triggers for poisoned graphs, we adapt SBA [50], EBA [43], and GCBA [49] to G-SBA, G-EBA, and G-GCBA to inject triggers into the original graphs. Specifically, G-SBA randomly generates small graphs and attaches these to one node of the original graphs. G-EBA uses the GNN explainability approach to find the optimal trigger-attaching position and alters the attributes of these positions. G-GCBA modifies the attributes of the original graphs to minimize the difference between the graph representations of poisoned and normal graphs. Additionally, we utilize SBAG [6] to inject triggers into the original graphs.

G-DShield. In Section V, we devised DShield for node classification attacks. For graph classification tasks, where each data sample is a graph, we adapt node-level contrastive learning to graph-level contrastive learning [41] in Eq. (12) and utilize graph representations instead of node representations, resulting in G-DShield. Consequently, G-DShield identifies poisoned graphs rather than poisoned nodes. Results in Table V show the performance of the proposed G-DShield.

Appendix B
Artifact Appendix

A. Description & Requirements

This subsection provides all necessary information to recreate the experimental setup for running the proposed artifact.

1) How to access: The artifact can be accessed at https://doi.org/10.5281/zenodo.14209303.

2) Hardware dependencies: None.

3) Software dependencies: As the code has been tested on the Windows 11 platform, we recommend using this operating system. The artifact is implemented in Python, utilizing PyTorch, scikit-learn, umap-learn, and other libraries. All required libraries must be installed before running the evaluations. Detailed specifications for these dependencies are provided in the *requirements.txt* file within the repository.

4) Benchmarks: This work focuses on defending against backdoor attacks in graph neural networks. The datasets used include Cora, PubMed, Flickr, OGBN-arXiv, ENZYMES, PROTEINS, and MNIST, with statistics detailed in Table VII. The GNN models employed are GCN, GraphSage, and GAT. Due to time constraints (limited to one day), the evaluation setups are designed for small datasets, i.e., Cora, PubMed, and ENZYMES, and a single GNN model (GCN). However, if reviewers have additional time for validation, larger datasets and a wider variety of GNN models may also be feasible.

B. Artifact Installation & Configuration

This subsection provides the high-level steps required to install and configure environments for evaluating the artifact.

1) Install Python: Download the Python installer from https://www.python.org/downloads/release/python-31011/.

2) Install Required Python Libraries: The required Python libraries are specified in the *requirements.txt* file, which includes the names and version numbers. To install these libraries, open the terminal and execute the following command:

```
pip install -r requirements.txt
```

Specifically, PyTorch (version 2.0.0) can be configured either with or without GPU support.

C. Experiment Workflow

This work addresses both node and graph classification tasks; accordingly, we provide two implementations of the proposed DShield, located in the *NodeClassificationTasks* and *GraphClassificationTasks* folders. The DShield implementation can be found in the *dshield.py* file within the *defense* sub-folder. Detailed command lines for running the evaluations are provided in the *README.md* file.

D. Major Claims

We summarize the key claims of our paper:

- (C1): Semantic drift and attribute over-emphasis occur in node classification backdoor attacks, as demonstrated by experiment (E1), with results shown in Figures 2, 3, and 11.
- (C2): DShield effectively identifies poisoned nodes, reducing the Attack Success Rate (ASR) to below 10% in most cases, outperforming state-of-the-art baselines, as shown by experiment (E2) in Tables I and II.
- (C3): Adaptive attacks cannot bypass DShield, as demonstrated by experiment (E3), with results presented in Table III.
- (C4): DShield can be adapted to mitigate graph classification backdoor attacks, supported by experiment (E4) in Table V.

E. Evaluation

1) Experiment (E1): [10 human-minutes + 1.5 compute-hours] This experiment conducts case studies to validate the presence of both *semantic drift* and *attribute over-emphasis*.

[Preparation] Open the terminal and navigate to the *NodeClassificationTasks* directory:

```
cd NodeClassificationTasks/
```

[Execution] In the paper, we perform case studies on five dirty-label backdoor attacks and two clean-label backdoor attacks. To reduce the evaluation time, we focus on two dirty-label attacks (GTA and UGBA) and two clean-label attacks (GCBA and PerCBA) for the case studies. For instance, to verify the existence of semantic drift in the GTA backdoor attack on the Cora dataset, execute the following command:

```
python viz_main.py --seed=1027 --model=GCN
↪  --dataset=Cora --benign_epochs=200
↪  --trigger_size=3 --vs_number=10 --use_vs_number
↪  --target_class=1
↪  --selection_method=cluster_degree
↪  --attack_method=GTA --gta_thrd=0.5
↪  --gta_lr=0.01 --gta_trojan_epochs=400
↪  --gta_loss_factor=0.0001 --defense_method=none
```

To check for attribute over-emphasis in the GCBA backdoor attack on the Cora dataset, run the following command:

```
python viz_main.py --seed=1027 --model=GCN
↪  --dataset=Cora --benign_epochs=200
↪  --trigger_size=3 --vs_number=10 --use_vs_number
↪  --target_class=1 --selection_method=clean_label
↪  --attack_method=GCBA --gcba_num_hidden=512
↪  --gcba_feat_budget=100 --gcba_trojan_epochs=300
↪  --gcba_ssl_tau=0.8 --gcba_tau=0.2
↪  --gcba_edge_drop_ratio=0.5
↪  --defense_method=none
```

Other commands are provided in the *README.md* file.

[Results] 1) For the semantic drift analysis, the script generates two files: *Cora_Supervised_GTA.svg* and *Cora_SSL _GTA.svg*. The former provides a t-SNE visualization of latent representations using a semi-supervised learning paradigm, while the latter illustrates latent representations using a self-supervised learning paradigm. In the semi-supervised case, poisoned nodes (black nodes) cluster near normal nodes with the target label 2. However, under the self-supervised paradigm, poisoned nodes are distant from normal nodes with the target label. 2) For the attribute over-emphasis analysis, the script generates two files: *Cora_Supervised_GTA.svg* and

Cora_Feat_GTA.svg. The former shows a t-SNE visualization of latent representations under semi-supervised learning, and the latter illustrates attribute importance. The latent representations of poisoned nodes tend to cluster near normal nodes labeled with the target label. Moreover, as GCBA modifies node attributes to strengthen the correlation between altered attributes and the target label, poisoned nodes form tighter clusters, demonstrating increased similarity among important attributes.

2) Experiment (E2): [40 human-minutes + 3 compute-hours] This experiment evaluates DShield's effectiveness in defending against various backdoor attacks.

[Preparation] Open the terminal and navigate to the *NodeClassificationTasks* directory:

```
cd NodeClassificationTasks/
```

[Execution] In the paper, we evaluate seven dirty-label backdoor attacks and two clean-label backdoor attacks. To reduce the evaluation time, we select two of the strongest dirty-label and two clean-label backdoor attacks to assess the effectiveness of DShield. Additionally, we use the Cora and PubMed datasets to further validate DShield's performance. For example, to execute the UGBA attack on the Cora dataset, the following command is used:

```
python main.py --seed=1027 --model=GCN
↪  --dataset=Cora --benign_epochs=200
↪  --trigger_size=3 --vs_number=10 --use_vs_number
↪  --target_class=1
↪  --selection_method=cluster_degree
↪  --attack_method=UGBA --ugba_thrd=0.5
↪  --ugba_trojan_epochs=200 --ugba_inner_epochs=5
↪  --ugba_target_loss_weight=5
↪  --ugba_homo_loss_weight=50
↪  --ugba_homo_boost_thrd=1.0
↪  --defense_method=DShield
↪  --dshield_pretrain_epochs=400
↪  --dshield_finetune_epochs=400
↪  --dshield_classify_epochs=200
↪  --dshield_neg_epochs=100 --dshield_kappa1=5
↪  --dshield_kappa2=5 --dshield_kappa3=0.1
↪  --dshield_edge_drop_ratio=0.20
↪  --dshield_feature_drop_ratio=0.20
↪  --dshield_tau=0.9 --dshield_balance_factor=0.5
↪  --dshield_classify_rounds=1
↪  --dshield_thresh=2.5
```

Additional command lines for other scenarios are provided in the *README.md* file.

[Results] The scripts generate a series of intermediate results, with **the final two output lines displaying the ASR and Accuracy (ACC)**, respectively. In general, the ASR and ACC values should align closely with those reported in Table I and Table II of the paper, with the ASR expected to be below 10%.

3) Experiment (E3): [20 human-minutes + 1 compute-hour] This experiment evaluates DShield's robustness against adaptive attacks on node classification tasks.

[Preparation] Open the terminal and navigate to the *NodeClassificationTasks* directory:

```
cd NodeClassificationTasks/
```

[Execution] In the paper, we evaluate five adaptive attacks across the Cora, PubMed, Flickr, and OGBN-arXiv datasets. To minimize evaluation time, we focus exclusively on the Cora dataset for our assessments. To execute the UGBA+LGCB attack on the Cora dataset, use the following command:

```
python main.py --seed=1027 --model=GCN
↪  --dataset=Cora --benign_epochs=200
↪  --trigger_size=3 --vs_number=20 --use_vs_number
↪  --target_class=1-2 --selection_method=mixture
↪  --attack_method=UGBA-LGCB --UGBA_thrd=0.5
↪  --ugba_trojan_epochs=200 --ugba_inner_epochs=5
↪  --ugba_target_loss_weight=5
↪  --ugba_homo_loss_weight=50
↪  --ugba_homo_boost_thrd=1.0
↪  --lgcb_num_budgets=200 --defense_method=DShield
↪  --dshield_pretrain_epochs=400
↪  --dshield_finetune_epochs=400
↪  --dshield_classify_epochs=400
↪  --dshield_kappa1=5 --dshield_kappa2=5
↪  --dshield_kappa3=0.01
↪  --dshield_edge_drop_ratio=0.20
↪  --dshield_feature_drop_ratio=0.20
↪  --dshield_tau=0.9 --dshield_balance_factor=0.5
↪  --dshield_classify_rounds=1
↪  --dshield_thresh=2.5
```

[Results] **The last two lines of output present the ASR and ACC, respectively.** Typically, the values for ASR and ACC should closely correspond with those reported in Table III of the paper.

4) Experiment (E4): [30 human-minutes + 2 compute-hour] This experiment evaluates DShield's performance in defending against graph classification attacks.

[Preparation] Open the terminal and navigate to the *GraphClassificationTasks* directory:

```
cd GraphClassificationTasks/
```

[Execution] In the paper, we evaluate three graph classification attacks across the ENZYMES, PROTEINS, and MNIST datasets. To save the evaluation time, we provide the instructions on the ENZYMES dataset for our assessments. Other larger datasets, such as PROTEINS and MNIST, will cost more. For instance, to conduct the G-SBA attack on the ENZYMES dataset, use the following command:

```
python main.py --seed=1027 --model=GCN
↪  --dataset=ENZYMES --benign_epochs=200
↪  --trigger_size=20 --vs_ratio=0.1
↪  --target_class=1 --attack_method=SBA
↪  --sba_attack_method=Rand_Gene
↪  --sba_trigger_prob=0.5 --defense_method=DShield
↪  --dshield_pretrain_epochs=400
↪  --dshield_finetune_epochs=400
↪  --dshield_classify_epochs=400
↪  --dshield_neg_epochs=100 --dshield_kappa1=0.1
↪  --dshield_edge_drop_ratio=0.20
↪  --dshield_feature_drop_ratio=0.20
↪  --dshield_tau=0.9 --dshield_balance_factor=0.5
↪  --dshield_classify_rounds=1
↪  --dshield_thresh=2.5
```

[Results] **The last two lines of output present the ASR and ACC, respectively.** Typically, the values for ASR and ACC should closely correspond with those reported in Table V of the paper.