2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Distribution-Aware Unsupervised Attacks on Graph Contrastive Learning

Jinhua Huang, Jing Zhu, Zhao Ma(✉), Hongli Ding, Yizhuo Wang, Xucen Luo

School of Computer Science, China University of Geosciences, Wuhan 430078, China
728312569@qq.com

Abstract. Graph contrastive learning has significantly advanced unsupervised graph representation learning, achieving performance comparable to supervised models. However, the robustness of graph contrastive learning models remains a major challenge. Most existing adversarial attacks are designed for supervised settings, making them inapplicable when label information is unavailable in unsupervised scenarios. To address this limitation, we propose D2AGCL, a distribution-aware unsupervised attack specifically designed for graph contrastive learning models. Our approach poisons graph data to degrade the overall quality of graph contrast learning embeddings by dynamically adjusting attack zones and gradient aggregation strategies, thus compromising the performance of downstream tasks. Extensive experiments on multiple benchmark datasets demonstrate that D2AGCL outperforms existing unsupervised attack methods and even achieves comparable or superior performance against supervised adversarial baselines.

Keywords: Graph Contrastive Learning, Unsupervised Attack, Distribution Shift.

1 Introduction

Graphs are a flexible and powerful data structure that naturally represent complex relationships and interactions. In recent years, with the exponential growth of data and the widespread application of complex systems, graph-structured data has been extensively used in various domains. With the rapid advancement of deep learning, researchers have proposed a range of graph-based machine learning models to extract structural information from graphs. Notable models include Graph Neural Networks (GNNs), DeepWalk [5], and Node2Vec [11], which embed graph data into a low-dimensional space to facilitate downstream tasks such as classification, regression, clustering, and recommendation[15]. The quality of these embeddings directly impacts the performance of downstream tasks.

However, recent studies have shown that these models exhibit significant vulnerability to adversarial attacks [16, 17, 18], such as node feature perturbations, topological manipulations, and fake node injections, which can substantially degrade model performance and pose security risks in real-world applications. Therefore, developing robust graph learning models against adversarial attacks is a critical research direction. Existing robustness studies, however, have predominantly focused on supervised and semi-supervised models [12, 14], despite the fact that many real-world graphs lack labeled data [14], and manual annotation is often impractical. This limitation has driven the development of unsupervised models, such as DeepWalk [5] and Node2Vec [11]. Nevertheless, a significant performance gap remains between these unsupervised methods and their supervised counterparts. The emergence of Graph Contrastive Learning (GCL) has bridged this gap, enabling unsupervised models to achieve performance comparable to supervised methods while exhibiting greater resilience against adversarial attacks[4, 19, 21, 22]. This robustness advantage arises because most existing adversarial attack methods rely on label information, which contradicts the fundamental setting of unsupervised contrastive learning. As a result, validating the robustness of graph contrastive learning models requires unsupervised attack strategies. Despite the recent advances in graph adversarial attacks, existing methods either rely on label information or fail to effectively exploit distribution shift, limiting their applicability to unsupervised GCL models.

To address this challenge, Bojchevski et al. [7] proposed an unsupervised attack specifically targeting DeepWalk. However, experimental results indicate that this method is ineffective against graph contrastive learning models. Later, Zhang et al. [1] introduced CLGA, an unsupervised attack on graph contrastive learning, which leverages gradient backpropagation to identify and perturb edges with the highest gradients in the adjacency matrix. While CLGA achieves promising attack performance, its gradient-based edge selection strategy, which perturbs the highest-gradient edges across the entire graph, results in a notable performance gap compared to supervised attacks, such as Metattack [8]. Li et al. [2] further improved CLGA and proposed PAGCL. PAGCL first performs node injection attacks before flipping edges based on gradient information. While this approach indeed enhances attack effectiveness, the introduction of fake nodes significantly increases the attack budget, potentially reducing the stealthiness of the attack.

To narrow the gap between unsupervised and supervised attacks under the same budget constraints, we propose D2AGCL, which integrates distribution shift theory with gradient-based perturbation strategies, ensuring a more effective and efficient attack without additional computational overhead. By dynamically adjusting the attack budget allocation, our approach prioritizes perturbations in regions that contribute the most to distribution shift. Notably, D2AGCL achieves performance comparable to supervised attacks without leveraging label information, and in some cases, even surpasses them.

Our Main Contributions Are as Follows:

- We propose D2AGCL, an unsupervised attack method designed for graph contrastive learning. This method does not rely on label information and perturbs the graph structure by dynamically allocating the attack budget based on gradient information, thereby degrading the performance of various downstream tasks.

- We introduce a novel attack budget allocation strategy based on distribution shift theory, which enables our attack to strategically perturb critical regions of the graph, significantly improving attack efficiency without increasing computational overhead.
- We conduct extensive experiments on three benchmark datasets, evaluating both node classification and link prediction tasks to demonstrate the effectiveness of D2AGCL. The results reveal that D2AGCL achieves performance comparable to supervised attack methods in both structural inference scenarios. Besides, We visualize the learned embeddings to show how D2AGCL influences the quality of them.

2 Preliminary

2.1 Graph Contrastive Learning

In graph-based machine learning, we consider a graph structure $G = (V, A)$, where V is the set of nodes and A is the set of edges. The topological structure of the graph can be mathematically represented by the adjacency matrix $A \in \mathbb{R}^{N \times N}$, where $N = |V|$ denotes the total number of nodes. can also be used to describe the edge relationships. Here, N denotes the total number of nodes. Each node carries feature information, which is described by the feature matrix $X \in \mathbb{R}^{N \times d}$, where d represents the feature dimension. The objective of graph contrastive learning (GCL) is to train an encoder $f(A, X)$ to generate node embeddings that can be effectively utilized in downstream tasks such as link prediction and node classification.

Currently the most widely used methods, such as the GCA framework proposed by Zhu et al. [4], primarily follow a three-stage node-level contrastive learning paradigm:

1. Multi-View Augmentation. To introduce structured variations, two augmented graph views v_1 and v_2 are by applying independent random augmentation operators t_1 and t_2 to the original graph.
2. Shared Encoder Representation. The augmented views v_1 and v_2 are processed by a shared-parameter encoder $f(A, X)$, producing node embeddings for each node in different views.
3. Contrastive Loss Optimization. A symmetric contrastive loss function is employed to pull embeddings of the same node (positive pairs) closer while pushing apart embeddings of different nodes (negative pairs). The contrastive loss for node i in view v_1 is formulated as follows:

$$L(\alpha_i^1, \alpha_i^2) = -\log \frac{e^{\psi(\alpha_i^1, \alpha_i^2)/\tau}}{e^{\psi(\alpha_i^1, \alpha_i^2)/\tau} + \sum_{j \neq i} (e^{\psi(\alpha_i^1, \alpha_j^1)/\tau} + e^{\psi(\alpha_i^1, \alpha_i^2)/\tau})} \tag{1}$$

Where ψ denotes the similarity function (e.g., cosine similarity, bilinear similarity, or multi-head attention similarity), and τ denotes the temperature parameter controlling distribution sharpness. Since the loss function is directionally asymmetric, a bi-directional symmetric loss is adopted to balance both views:

$$\mathcal{L} = \sum_{i=1}^{N} L(\alpha_i^1, \alpha_i^2) + L(\alpha_i^2, \alpha_i^1) \tag{2}$$

by forcing the model to distinguish between embeddings of the same node across different augmented views and negative samples from other nodes, this contrastive learning paradigm significantly enhances representation robustness against random perturbations, outperforming traditional graph representation learning methods in terms of generalization capability.

2.2 Distribution Shift in Graph Adversarial Attacks

In gradient-based graph adversarial attacks, Metattack has been widely adopted as a baseline model due to its strong performance. Metattack formulates the adversarial attack problem as a bilevel optimization problem and solves it using meta-learning techniques. Studies on Metattack have revealed that gradient-based perturbations are not uniformly distributed across the graph. From the perspective of distribution shift [13], it has been mathematically demonstrated that perturbations applied to the smaller partition of the dataset can be more effective in amplifying distribution shift, thereby improving the attack's success rate.

The concept of distribution shift is prevalent in machine learning and refers to the performance degradation of a model when the training and test data distributions differ[3, 23, 24]. The core issue lies in the fact that distribution shift arises when the distribution learned during training fails to generalize to the actual test distribution, leading to misaligned decision boundaries. There are two highly significant distributions in the distribution offset, namely $p_{train}(x, y)$ and $p_{test}(x, y)$, which represent the joint probability distribution of train data and test data. x denotes the input variable, and y denotes the output variable. For instance, in graph classification, x consists of structural information, including node attributes and graph topology, while y corresponds to node labels. In unsupervised graph adversarial attacks, the attacker has the ability to modify the entire graph structure, including both training and test nodes. Thus, graph adversarial attacks can selectively manipulate local structures, exacerbating the distribution shift between training and test nodes, thereby systematically degrading the model's inference capability. For a clean graph, we assume that $p_{train}(x, y)$ and $p_{test}(x, y)$ match the true distribution $p(x, y)$. However, in an attacked graph, adversarial perturbations directly alter the corresponding probability distributions. After applying perturbations, $p_{train}(x, y)$ and $p_{test}(x, y)$ may deviate from $p(x, y)$, leading to distribution shift. By factorizing the joint distribution $p(\tilde{x}, y)$, we obtain the following result:

$$p(\tilde{x}, y) = p(y)p(\tilde{x} \mid y) \tag{3}$$

$\tilde{x}$ means structural perturbation of the graph. Eq. 3 implies that when node labels remain unchanged, and only the graph structure is perturbed, the primary impact on the true distribution stems from modifications to $p_{train}(\tilde{x} \mid y)$ and $p_{test}(\tilde{x} \mid y)$. Therefore, the distribution shift in graph adversarial attacks can be quantified as follows:

$$\frac{1}{|C|}\sum_{y \in C} D_{KL}(p_{train}(\tilde{x} \mid y = c_i), p_{test}(\tilde{x} \mid y = c_i)) \tag{4}$$

C represents the total number of classes, c_i denotes the class of i, and D_{KL} is the Kullback-Leibler (KL) divergence, which measures the difference between two probability distributions. $p_{train}(\tilde{x} \mid y = c_i)$ and $p_{test}(\tilde{x} \mid y = c_i)$ refer to the distribution of node embeddings within class c_i for training and test nodes, respectively. This formula computes the average KL divergence across all classes, quantifying the effectiveness of adversarial attacks by measuring how much the distribution of training and test embeddings diverges after perturbation. Based on Eq. 4, Li et al. demonstrated that in graph adversarial attacks, when the training set is small, uniformly inserting heterophilic edges within the training set results in a larger distribution shift.

3 The Proposed Method

3.1 Method Overview

We propose a distribution-aware unsupervised graph contrastive learning attack framework (D2AGCL), designed to poison graph data by dynamically adjusting attack regions and gradient aggregation strategies, thereby degrading the overall quality of graph contrastive learning embeddings and ultimately impairing downstream task performance. The framework structure is illustrated in Fig. 1.

D2AGCL integrates adaptive data partitioning and gradient-based information. First, it categorizes edges into three regions (Train-Train, Train-Test, Test-Test) based on the partitioning of the training and test sets. Attack budgets are allocated from the perspective of distribution shift, ensuring a targeted perturbation strategy. Within each region, nodes with degrees lower than the average are selected to form a candidate edge set, enhancing the effectiveness of the attack. Subsequently, adaptive data augmentation is applied to the original graph to generate two contrastive views, which are then processed by a shared graph neural network to compute the contrastive loss. The gradient of the adjacency matrix is obtained via backpropagation.

Unlike uniform gradient aggregation, D2AGCL dynamically weights gradients based on regional importance—for instance, when the training set size is small, higher weight is assigned to Train-Test region gradients—to maximize the distribution shift between training and test embeddings. Finally, the edges with the largest absolute gradient values are iteratively perturbed until the predefined perturbation budget is reached. Experimental results demonstrate that D2AGCL significantly outperforms baseline methods in node classification and link prediction tasks. Additionally, the effectiveness of the distribution-aware attack mechanism has been verified.

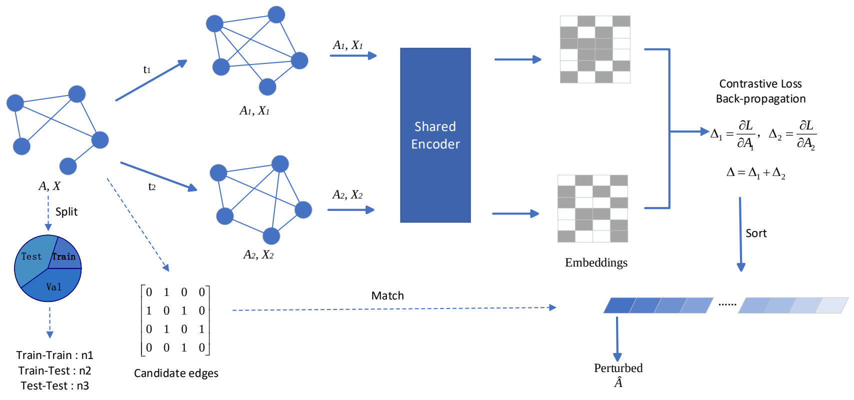


Fig. 1. The overall framework of D2AGCL. It is a single iteration of the attack. The clean graph is taken as input, and the output is a perturbed adjacency matrix Â where one edge has been modified. In the next iteration, Â is used as input for further attacks.

3.2 Distribution-Aware Attack Budget Allocation Strategy

Existing studies indicate that gradient-driven malicious modifications on graphs are not uniformly distributed. From the perspective of distribution shift, perturbing nodes in the smaller partition of a dataset can significantly amplify the distribution shift, thereby enhancing attack efficiency. Given a limited attack budget, dynamically adjusting the attack ratio across different partitions based on dataset distribution can effectively improve attack performance.

we partition the graph edges into three categories based on the division of the training and test sets: Train-Train (TT), Train-Test (Tt), and Test-Test (tt). The Train-Train (TT) region consists of edges connecting two training nodes, the Train-Test (Tt) region consists of edges connecting training nodes and test nodes, and the Test-Test (tt) region consists of edges connecting two test nodes. Based on this, we define a sensitivity weight for each region: W_{TT} , W_{Tt} and W_{tt} . These weights are computed according to the proportion of intra-region edges relative to the total number of edges. Specifically, for each region, we denote this proportion as α_i .These weights reflect the contribution of different regions to distribution shift and are calculated as follows:

$$\alpha_i = \frac{Ei}{E}, W_i = \frac{1}{e^{\alpha_i - 0.5} + \varepsilon}, i \in \{TT, Tt, tt\} \tag{5}$$

E_i represents the number of edges that can be modified within the corresponding region, and E is the total number of edges in the dataset. ε is a smoothing factor. Specifically, when the training set is small (datasets typically follow a 10%/10%/80% split for training, validation, and testing), W_{TT} has the highest weight. In this case, the attacker should prioritize perturbing edges within the training set to maximize the embedding discrepancy between the training and test sets. As the training set proportion increases to 50%, the weights across different regions become more balanced, leading to a more evenly distributed attack. When the test set proportion is small, W_{tt} reaches its highest value, causing the attack to be concentrated around the test set.

Finally, the attack budget is allocated based on the normalized sensitivity weights. Therefore, when the dataset is unevenly partitioned, the attack budget is primarily focused on the smaller partitions, where perturbations have a greater impact on the distribution shift [3].

3.3 Gradient-Guided Edge Perturbation

Efficiently selecting perturbation target edges is crucial in untargeted poisoning attacks against graph contrastive learning. In the previous section, we allocated the attack budget unevenly based on the dataset distribution. In this section, we focus on selecting specific edges to perturb within the allocated budget. We aim to reduce the overall performance of node embeddings learned by the graph contrastive learning model by poisoning the graph structure. In contrastive learning, the model employs a contrastive loss function to measure embedding quality. The core idea of contrastive loss is to learn embeddings by optimizing the distance between positive samples (similar samples) and negative samples (dissimilar samples). Specifically, contrastive learning enhances embedding quality by minimizing contrastive loss, which involves reducing the similarity between negative samples while increasing the similarity between positive samples. This process improves both the discrimination and overall quality of the learned embeddings. Consequently, to poison the graph data, we aim to maximize the contrastive loss function. This problem can be formulated by Eq. 6:

$$\begin{aligned} \max_{\hat{A}} \quad & \mathcal{L}(f_{\theta'}(A_1, X_1), f_{\theta'}(A_2, X_2)) \\ s.t. \quad & \theta' = \arg\min_{\theta} \mathcal{L}(f_{\theta}(A_1, X_1), f_{\theta}(A_2, X_2)), \\ & (A_1, X_1) = t_1(\hat{A}, X), (A_2, X_2) = t_2(\hat{A}, X), \left\| A - \hat{A} \right\| = \sigma. \end{aligned} \tag{6}$$

where $\mathcal{L}$ represents the contrastive loss function in graph contrastive learning, and f denotes the encoder. θ and θ' denotes the learnable parameters of the shared encoder before and after perturbation. A and X are the adjacency matrix and feature matrix of the clean graph, while $\hat{A}$ is the poisoned adjacency matrix. The functions t_1 and t_2 represent two random graph augmentation operations, yielding the adjacency matrices A_1 and A_2 corresponding feature matrices X_1 and X_2. The last constraint in the equation limits the number of perturbed edges within a given threshold σ.

Since Eq. 6 represents a bilevel optimization problem with a discrete adjacency matrix, direct gradient computation is infeasible. Following the meta-gradient theory [8], we approximate the gradient by back-propagating the contrastive loss and update the adjacency matrix accordingly to maximize the loss. For differentiable encoders, such as Graph Convolutional Networks (GCNs), the gradient can be computed by Eq. 7.

$$\Delta_1 = \frac{\partial \mathcal{L}}{\partial A_1} = \frac{\partial \mathcal{L}}{\partial f(A_1, X_1)} \cdot \frac{\partial f(A_1, X_1)}{\partial A_1}, \ \Delta_2 = \frac{\partial \mathcal{L}}{\partial A_2} = \frac{\partial \mathcal{L}}{\partial f(A_2, X_2)} \cdot \frac{\partial f(A_2, X_2)}{\partial A_2} \tag{7}$$

Edges that contain more informative structural signals tend to have a greater impact on graph learning models, which is reflected in their larger gradient values. To conduct a more effective attack, we prioritize perturbing these high-information edges, as disrupting them can significantly impair the model's ability to extract meaningful graph representations. Our primary goal is to identify and target these crucial edges.

However, not all random augmentation operations are differentiable. Operations such as node addition, node removal, or subgraph sampling introduce randomness, making the gradient of the original graph Δ difficult to compute directly. However, the gradients of the two augmented views v_1 and v_2, obtained through transformations t_1 and t_2, can be computed efficiently. We can't approximate the gradient of the original graph using augmented views, because we cannot determine whether the largest gradients in the augmented views result from meaningful structural information or artifacts of the augmentation process.

To address this issue, let us revisit traditional graph contrastive learning. The goal of contrastive learning is to minimize the difference between augmented views by updating model parameters. Suppose that two contrastive views were identical, in this case, their adjacency matrix gradients would be zero. Now, if we introduce augmentations to generate two distinct views v_1 and v_2, and an edge exists in v_1 but is missing in v_2, we expect the gradient in v_1 to be negative and the gradient in v_2 to be positive. This is consistent with the contrastive learning objective, which seeks to reduce the differences between views. Thus, to mitigate augmentation-induced noise, we propose summing the gradients of corresponding edges across both views, effectively balancing positive and negative gradients and minimizing augmentation-related bias. The final gradient computation is as follows:

$$\Delta' = \sum_{i=1}^{K} \Delta_1^i + \Delta_2^i \tag{8}$$

where for each i, Δ_1^i and Δ_2^i are the gradients computed from the two augmented views. We use Δ' as the final selection criterion for edge perturbation. In the first iteration, we train the contrastive model on the input graph to obtain the initial gradient. Using this gradient, we select and flip a single edge to maximize the contrastive loss. We then update the adjacency matrix and repeat the process in the next iteration. Algorithm 1 provides a step-by-step overview of the entire attack procedure. A more intuitive illustration can be found in Fig. 1.

3.4 Complexity

Our attack method only sorts the gradients based on GCA and filters some nodes, so its time complexity is low. The main time is the training process of the model, because we only choose to perturb one edge in one iteration. For different models, you can choose to select more than one edge at a time. In addition, the spatial complexity of D2AGCL is $O(N^2)$, where N represents the number of nodes, which is terrible in large graphs. However, we can calculate the gradient only for a subset of the graph rather than for the whole graph, which is widely used in unsupervised learning.

Algorithm 1: D2AGCL

```
Input: Adjacency matrix of clean graph A, feature matric X, shared differentiable
       encoder $f$, random augmentation set T, iterations K, total attack budget δ,
       training set split ratio $α_1$, test set split ratio $α_2$.
Output: Poisoned adjacency matrix Â.
4. Initialization: $p_i = 0$, $i \in \{TT, Tt, tt\}$; $\hat{A} = A$;
5. $w_{TT} = 1/(e^{\alpha_1 - 0.5} + \varepsilon)$, $w_{Tt} = 1/(e^{1-\alpha_1-\alpha_2-0.5} + \varepsilon)$, $w_{tt} = 1/(e^{\alpha_2 - 0.5} + \varepsilon)$;
6. Normalize weights $w_{TT}, w_{Tt}, w_{tt}$; Allocate attack budgets $\delta_i = w_i \cdot \delta$;
7. while $p_i \le \delta_i$ do:
8.     Train $f$ with $\hat{A}$ and $X$;
9.     $\Delta' = 0$;
10.    for k = 1 to K do
11.        Sample two stochastic augmentations $t_1^k, t_2^k \in T$;
12.        Obtain two views $(A_1^k, X_1^k) = t_1^k(\hat{A}, X)$, $(A_2^k, X_2^k) = t_2^k(\hat{A}, X)$;
13.        Forward propagate $(A_1^k, X_1^k), (A_2^k, X_2^k)$ through $f$ and compute contrastive
           loss $\mathcal{L}$;
14.        Obtain the gradients of $A_1^k$ and $A_2^k$: $\Delta_1^k = \partial\mathcal{L}/\partial A_1^k$, $\Delta_2^k = \partial\mathcal{L}/\partial A_2^k$;
15.        $\Delta' = \Delta_1^k + \Delta_2^k$;
16.    end for
17.    Flip one edge with both the largest absolute gradient in $\Delta'$ and the correct direc-
       tion, i.e., if the index of the edge is $[m, n]$, then it should satisfy either $\hat{A}[m,n] = 1$,
       $\Delta'[m,n] < 0$ or $\hat{A}[m,n] = 0$, $\Delta'[m,n] > 0$;
18.    $\hat{A}[m,n] = 1 - \hat{A}[m,n]$;
19.    $p_i = p_i + 1$;
20. end while
```

4 Experiments

4.1 Setup

Datasets. To evaluate the effectiveness of our attack method D2AGCL, we conduct experiments on three widely used benchmark datasets: Cora, CiteSeer, and PolBlogs.

These datasets are commonly used in graph-based learning tasks. Specifically, Cora and CiteSeer are citation networks, while PolBlogs is a political blog graph, which differs from the others and its nodes do not contain feature information. Table 1 provides a detailed summary of these datasets.

Table 1. The detailed information about the datasets.

Dataset	Node	Edges	Classes	Features
Cora	2708	5278	7	1433
CiteSeer	3327	4552	6	3703
PolBlogs	1490	16715	2	None

Baseline. We compare D2AGCL against five untargeted poisoning attack baselines, including four supervised methods: PGD [9], DICE [10], MinMax [9], and Metattack [8], as well as three unsupervised methods: Node Embedding Attack (NEA) [7] , CLGA [1] and PAGCL[2]. Since supervised attack methods leverage label information as additional knowledge, they are generally expected to perform better than unsupervised attacks. However, we demonstrate that D2AGCL achieves comparable performance and even outperforms certain supervised baselines in specific scenarios.

Parameter Settings. To ensure consistency across all attack models, we use a 2-layer GCN as a surrogate model to generate the poisoned graph. The poisoned graph is then used as the input for GCA [4]. We set three different attack budget thresholds: 0.01, 0.05, and 0.10, which constrain the maximum number of perturbed edges. These thresholds allow us to compare the attack performance of different models under varying attack budgets. For dataset splitting, we follow the standard settings used in [12] for Cora and CiteSeer, while for PolBlogs, we split the dataset into 10% for training, 10% for validation, and 80% for testing. In [12], the splitting criterion for Cora and CiteSeer is as follows: the number of training nodes is determined as 20 times the number of classes (e.g., Cora has 7 classes, so the training set contains 140 nodes). The validation set consists of 500 nodes, while the test set contains 1,000 nodes. Similarly, for PolBlogs, the training set comprises 10% of the total nodes, ensuring that it remains the smallest proportion of the dataset. Additionally, we allow dynamic random splits by adjusting the dataset ratio. We conduct experiments with different hyperparameter settings, such as the learning rate, to ensure optimal model performance. To maintain fairness in our experiments, we follow the recommendations of Zhu et al. [4] and fix the hyperparameters of GCA, including the temperature τ and random augmentation rates. Specifically, we set $\tau = 0.4$, edge dropout rates to 0.3 and 0.4, and feature dropout rates for the two views to 0.1 and 0.0, respectively.

4.2 Experimental Results and Analysis

We validate and evaluate our attack method in node classification and link prediction, two common downstream tasks in graph neural networks. In node classification, we use the learned node embeddings to train a simple logistic regression model and report the classification accuracy. Furthermore, we employ a two-layer MLP (Multi-Layer Perceptron) as the projection head to map the embeddings into a new space. We train the MLP using negative sampling and a margin-based loss, enabling the model to perform link prediction effectively. Each experiment is repeated 20 times, and we record the average accuracy across all runs.

Table 2. Accuracy of node classification of GCA trained logistic regression model under three different attack budgets. The best attack methods are bold, while the second best is underlined.

Type	Attack Methods	Cora			CiteSeer			PolBlogs		
		1%	5%	10%	1%	5%	10%	1%	5%	10%
Supervised	Metattack	0.7586	0.6928	0.6168	0.5920	**0.3986**	**0.2952**	0.8208	0.8039	0.8011
	PGD	0.7680	0.7592	0.7402	0.6098	0.6198	0.6056	0.8100	0.8010	0.7987
	MinMax	0.7624	0.7218	0.6174	0.6302	0.5254	0.5618	0.8016	0.7913	0.7986
	Dice	0.7712	0.7642	0.724	0.6256	0.5774	0.5246	0.8107	0.7847	0.7394
Unsupervised	NEA	0.7490	0.7710	0.7670	0.6442	0.6448	0.6608	0.8187	0.8042	0.7892
	CLGA	0.7316	0.7188	0.6814	0.6368	0.5906	0.5368	0.8088	0.7944	0.7726
	PAGCL	**0.6752**	**0.6429**	**0.5938**	**0.5410**	0.5220	0.5010	0.7845	0.7724	0.7689
	D2AGCL	0.7131	0.6589	0.6464	0.5863	0.5194	0.4382	**0.7139**	**0.6860**	**0.6605**

Table 3. Accuracy of link prediction of the MLP trained after GCA under three different attack budgets. The best attack methods are bold, while the second best is underlined.

Type	Attack Methods	Cora			CiteSeer			PolBlogs		
		1%	5%	10%	1%	5%	10%	1%	5%	10%
Supervised	Metattack	0.9010	0.8733	0.8500	0.9109	0.8853	0.8544	0.8617	0.8585	0.8635
	PGD	0.9143	0.9073	0.9073	0.9169	0.9248	0.9057	0.8605	0.8584	0.8625
	MinMax	0.9116	0.9004	0.8944	0.9145	0.8890	0.8981	0.9145	0.8890	0.8981
	Dice	0.9046	0.8828	0.8593	0.9137	0.8918	0.8679	0.8551	0.8450	0.8352
Unsupervised	NEA	0.9164	0.9099	0.9101	0.9239	0.9168	0.9196	0.8593	0.8543	0.8587
	CLGA	0.9012	0.8741	0.8420	0.9114	0.8911	0.8610	0.8584	0.8598	0.8563
	PAGCL	**0.8078**	**0.7951**	**0.7769**	**0.7891**	**0.7782**	**0.7617**	0.8469	0.8449	0.8379
	D2AGCL	0.8561	0.8228	0.7927	0.8805	0.8781	0.8396	**0.7728**	**0.7730**	**0.7420**

Table 2 presents the classification accuracy of the logistic regression model on poisoned graphs generated by and baseline attack methods under different attack budgets. The results are reported using the default dataset splits. From the experimental results, D2AGCL outperforms all baselines except Metattack and PAGCL across all three datasets. D2AGCL achieves the best performance on PolBlogs, and in Cora and CiteSeer, it almost always achieved suboptimal performance. PAGCL builds upon CLGA by introducing a node injection attack. Before applying edge perturbations as in the original CLGA method, PAGCL first injects malicious nodes into the clean graph. This strategy achieves performance comparable to supervised attacks. However, it violates the assumption of equal attack budget. In their comparison, the authors only consider the perturbation ratio of edges, allowing PAGCL to introduce significantly more perturbation under the same nominal budget, thus compromising the fairness of the evaluation. Therefore, if PAGCL is excluded from consideration, these results demonstrate that D2AGCL effectively degrades the performance of graph contrastive learning in node classification tasks, achieving a level comparable to supervised attack models. Table 3 presents the accuracy on another downstream task: link prediction. Excluding PAGCL, D2AGCL consistently achieves the best performance, especially on PolBlogs. Therefore our unsupervised attack D2AGCL is shown to be able to reduce both the downstream node classification performance and link prediction performance of graph contrastive learning.

Fig. 2 compares the classification accuracy of D2AGCL and the unsupervised attack method CLGA under different dataset partition settings. D2AGCL consistently outperforms CLGA across all configurations. Moreover, as the dataset split ratio changes, D2AGCL exhibits a performance trend of an initial decrease followed by an increase, whereas CLGA's attack effectiveness continuously declines. This confirms that D2AGCL dynamically adjusts its attack budget based on the dataset distribution, thereby achieving a more effective poisoning attack.

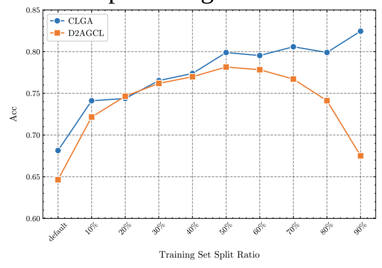
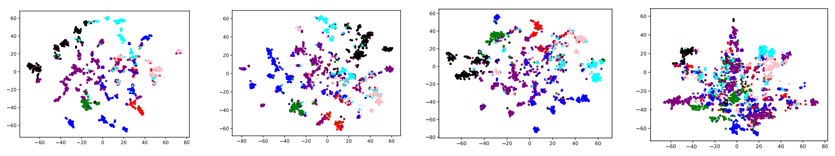


Fig. 2. Node Classification Accuracy of CLGA and D2AGCL on the Cora Dataset Under Different Data Splits. The x-axis represents the training set split ratio, while the y-axis denotes the accuracy of node classification under a 10% attack budget.



(a)Clean graph (b)Metattack (c)CLGA (d) D2AGCL

Fig. 3. Visualization of node embeddings learned by GCA under different attacks on Cora. Different Colors represent different classes of nodes.

To evaluate the adversarial effectiveness, we conduct visual analysis of node embeddings under different attack scenarios using t-SNE dimensionality reduction. As shown in Fig. 3, we compare the embedding distributions of the original graph with those perturbed by Metattack, CLGA, and our proposed D2AGCL method. The visualization reveals distinct topological distortion patterns: While Metattack generates moderately compact clusters with overlapping boundaries, CLGA exhibits dispersed yet structurally ambiguous formations. Notably, D2AGCL demonstrates superior adversarial characteristics by creating tightly entangled embeddings in inter-cluster regions. This phenomenon is particularly evident in the boundary areas where node representations from different classes become inseparable, manifesting as a fused density core near cluster intersections.

5 Conclusion

In this paper, we propose D2AGCL, an unsupervised adversarial attack specifically designed for graph contrastive learning (GCL) models. Unlike existing adversarial attacks that rely on label information, D2AGCL effectively perturbs the graph structure by integrating distribution shift theory with gradient-based attack strategies. Our approach dynamically allocates the attack budget to maximize distribution shift, ensuring that perturbations are strategically applied to the most critical regions of the graph. Extensive experiments on three benchmark datasets demonstrate that D2AGCL achieves attack performance comparable to supervised methods, despite operating in an unsupervised setting. Our findings highlight the vulnerability of GCL models to adversarial perturbations, emphasizing the need for more robust unsupervised defense mechanisms. Future work will explore adaptive defenses against such attacks, as well as extending our approach to heterophilic graphs and dynamic graph settings.

References

1. Zhang, S., Chen, H., Sun, X., Li, Y., Xu, G.: Unsupervised Graph Poisoning Attack via Contrastive Loss Back-Propagation. In: Proceedings of the ACM Web Conference 2022, pp. 1322–1330(2022)
2. Li, Q., Wang, Z., Li, Z.: PAGCL: An Unsupervised Graph Poisoned Attack for Graph Contrastive Learning Model. In: Future Generation Computer Systems 2023, vol. 149, pp. 240–249(2023)
3. Li, K., Liu, Y., Ao, X., & He, Q.: Revisiting graph adversarial attack and defense from a data distribution perspective. In: The Eleventh International Conference on Learning Representations (2023)
4. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph Contrastive Learning with Adaptive Augmentation. In: Proceedings of the Web Conference 2021, pp. 2069–2080(2021)
5. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: Online Learning of Social Representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701–710(2014)
6. Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., Lin, X.: Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, pp. 3961–3967(2019)
7. Bojchevski, A., Günnemann, S.: Adversarial Attacks on Node Embeddings via Graph Poisoning. In: Proceedings of the 36th International Conference on Machine Learning, pp. 695-704(2019)
8. Zügner, D., Günnemann, S.: Adversarial Attacks on Graph Neural Networks via Meta Learning. In: International Conference on Learning Representation (2019)

9. Xu K., Chen H., Liu S., Chen P., Weng T., Hong M., and Lin X.: Topology attack and defense for graph neural networks: an optimization perspective. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 3961–3967 (2019).
10. Waniek, M., Michalak, T., Wooldridge, M. *et al*.: Hiding individuals and communities in a social network, pp. 139–147 (2018).
11. Grover A., Leskovec J.: Node2vec: Scalable Feature Learning for Networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864(2016).
12. Yang, Z., Cohen, W.Cohen W., Salakhutdinov, R.: Revisiting Semi-Supervised Learning with Graph Embeddings. In: Proceedings of the 33rd International Conference on Machine Learning, pp. 40–48 (2016)
13. Ding, R., Yang, J., Ji, F., Zhong, X., Xie, L.: FR-GNN: Mitigating the Impact of Distribution Shift on Graph Neural Networks via Test-Time Feature Reconstruction. In: IEEE Internet of Things Journal, pp. 23521–23531 (2024).
14. Li, G., Yu, Z., Yang, K., Lin, M., Chen, C. L. P.: Exploring Feature Selection With Limited Labels: A Comprehensive Survey of Semi-Supervised and Unsupervised Approaches. In: IEEE Transactions on Knowledge and Data Engineering, pp. 6124–6144 (2024).
15. Liu, X., Zhang, F., Hou, Z., Mian, et al.: Self-Supervised Learning: Generative or Contrastive. In: IEEE Transactions on Knowledge and Data Engineering, pp. 857–876 (2023).
16. Wang, X., Chang, H., Xie, B., et al.: Revisiting Adversarial Attacks on Graph Neural Networks for Graph Classification. In: IEEE Transactions on Knowledge and Data Engineering, pp. 2166–2178 (2024).
17. Wang, H., Xu, C., Shi, C., Zheng, P., Zhang, S., Cheng, M., Chen, H. Unsupervised Heterogeneous Graph Rewriting Attack via Node Clustering. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining pp. 3057-3068.
18. Yu, H., Liang, K., Hu, D., et al.: GZOO: Black-Box Node Injection Attack on Graph Neural Networks via Zeroth-Order Optimization. In: IEEE Transactions on Knowledge and Data Engineering, pp. 319–333 (2025).
19. Huang, Y., Zhao, J., He, D., Jin, D., Huang, Y.: Does GCL Need a Large Number of Negative Samples? Enhancing Graph Contrastive Learning with Effective and Efficient Negative Sampling. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025).
20. Zhang, X., Bao, P., Pan, S.: Maximizing Malicious Influence in Node Injection Attack. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, pp. 958–966 (2024).
21. Xu, J., Yang, Y., Chen, J., et al.: Unsupervised Adversarially Robust Representation Learning on Graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 4, pp. 4290–4298 (2022).
22. Deng, B., Chen, J., Hu, Y., Xu, Z., Chen, C., Zhang, T.: PROSPECT: Learn MLPs on Graphs Robust against Adversarial Structure Attacks. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pp. 425–435 (2024).
23. Wiles, O., Gowal, S., Stimberg, F., Rebuffi, S.-A., Ktena, I., Dvijotham, K., Cemgil, A.T.: A Fine-Grained Analysis on Distribution Shift. In: International Conference on Learning Representations (ICLR) (2022).
24. Kim, S., Im, H., Lee, W., Lee, S., Kang, P.: RobustMixGen: Data augmentation for enhancing robustness of visual–language models in the presence of distribution shift. In: Neurocomputing, vol. 619, p. 129167 (2025).