Knowledge Distillation Improves Graph Structure Augmentation for Graph Neural Networks

Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z. Li

Abstract

Graph (structure) augmentation aims to perturb the graph structure through heuristic or probabilistic rules, enabling the nodes to capture richer contextual information and thus improving generalization performance. While there have been a few graph structure augmentation methods proposed recently, none of them are aware of a potential negative augmentation problem, which may be caused by overly severe distribution shifts between the original and augmented graphs. In this paper, we take an important graph property, namely graph homophily, to analyze the distribution shifts between the two graphs and thus measure the severity of an augmentation algorithm suffering from negative augmentation. To tackle this problem, we propose a novel Knowledge Distillation for Graph Augmentation (KDGA) framework, which helps to reduce the potential negative effects of distribution shifts, i.e., negative augmentation problem. Specifically, KDGA extracts the knowledge of any GNN teacher model trained on the augmented graphs and injects it into a partially parameter-shared student model that is tested on the original graph. As a simple but efficient framework, KDGA is applicable to a variety of existing graph augmentation methods and can significantly improve the performance of various GNN architectures. For three popular graph augmentation methods, namely GAUG, MH-Aug, and GraphAug, the experimental results show that the learned student models outperform their vanilla implementations by an average accuracy of 4.6% (GAUG), 4.2% (MH-Aug), and 4.6% (GraphAug) on eight graph datasets. Codes are available at: https://github.com/LirongWu/KDGA.

1 Introduction

In many real-world applications, including social networks, chemical molecules, and citation networks, data can be naturally modeled as graphs. Recently, the emerging Graph Neural Networks (GNNs) [4, 13, 48, 47, 22, 24, 28, 29] have demonstrated their powerful capability due to their superior performance in various graph-related tasks, including link prediction [59], node classification [21, 50], and graph classification [7]. Despite their great success, GNNs usually suffer from weak generalization due to its heavy reliance on the quantity of annotated labels and the quality of the graph structure. To boost generalization capabilities, a natural solution is to increase the amount of training data by creating plausible variations of existing data, which have been widely adopted in fields such as computer vision [33, 30, 9, 39, 25, 31, 26, 27] and natural language processing [46, 1, 36, 5, 34].

The data augmentation on graphs can be mainly divided into two branches: node feature augmentation and graph structure augmentation. While the former has been well studied by directly extending existing approaches for image and text data to graph data [36, 19, 15], comparatively little work has been done to study graph structure augmentation [35, 2, 62, 32]. Following the nomenclature of existing works [62, 32], we directly abbreviate graph (structure) augmentation to graph augmentation.
for the sake of brevity in this paper. The purpose of graph augmentation is to reasonably perturb the graph structure through heuristic or probabilistic rules, enabling the nodes to capture richer contextual information and thus improving generalization performance. For example, DropEdge [35] randomly removes a fraction of edges before each training epoch, in an approach reminiscent of dropout [40]. Besides, AdaEdge [2] iteratively adds (removes) edges between nodes predicted to have the same (different) labels with high confidence. In contrast to these heuristic methods, GAUG [62] proposes to optimize the graph augmentation and GNN parameters in an end-to-end manner. Similarly, MH-Aug [32] proposes a novel framework that draws a sequence of augmented graphs from an explicit target distribution, which enables flexible control of the strength and diversity of augmentation.

In this paper, we identify a potential negative augmentation problem for existing graph augmentation methods, i.e., the augmentation may cause overly severe distribution shift between the augmented graphs used for training and the original graph used for testing, which leads to suboptimal generalization. Moreover, we conduct extensive experiments to demonstrate the existence and hazard of distribution shifts and find that the direction of distribution shifts may be opposed on homophily and heterophily graphs. We propose a solution to the identified problem by adopting a Knowledge Distillation for Graph Augmentation (KDGA) framework, which helps to reduce the potential negative effects of distribution shifts. Specifically, it extracts the knowledge of any GNN teacher model trained on the augmented graphs and injects it into a partially parameter-shared student model that is tested on the original graph. As a general framework, KDGA can significantly improve the vanilla implementations of various popular graph augmentation methods and GNN architectures.

Our contributions are summarized as follows: (1) We are the first to identify a potential negative augmentation problem for graph augmentation, and more importantly, we have described in detail what it represents, how it arises, what impact it has, and how to deal with it. (2) We propose a novel Knowledge Distillation for Graph Augmentation (KDGA) framework for the identified problem by directly distilling contextual information from augmented graphs. (3) We provide comprehensive experimental results showing that KDGA is applicable to a variety of graph augmentation methods and GNN models; it substantially outperforms the vanilla implementations across various datasets.

2 Background and Related Work

Structure Augmentation for Graphs. Data augmentation is an effective technique to improve generalization. Despite the great progress on node feature augmentation [56, 19, 15], comparatively little work study graph (structure) augmentation [35, 2, 62, 32, 6] due to the non-Euclidean property of structures. For graph data, the mainstream algorithms for structure augmentation are divided into two categories: heuristic and learning-based. As a typical heuristic algorithm, DropEdge [35] randomly remove edges according to the hand-crafted probability. In a similar way, AdaEdge [2] iteratively adds (removes) edges between nodes predicted to have the same (different) labels. Different from the above heuristic methods, GAUG [62] propose to optimize the graph augmentation and learnable GNN parameters in an end-to-end manner. Instead, MH-Aug [32] proposes a sampling-based augmentation, where a sequence of augmented graphs are directly drawn from an explicit target distribution.

Graph Structure Learning and Graph Contrastive Learning. Two closely related topics to graph augmentation are Graph Structure Learning [18, 23, 58, 20, 3] and Graph Contrastive Learning [19, 15, 57, 51, 63, 52], but they are quite different in terms of learning objectives and evaluation protocols. The learning goal of structure learning is to estimate a new structure with high quality [10, 8]. Instead, graph augmentation aims to reasonably perturb the graph structure during training to produce a set of augmented graphs, enabling nodes to receive richer contextual information; such augmentations allow the model to generalize better across those variations. As for the evaluation protocol, the augmented graphs are only used during training and are not available during testing. In contrast, for graph structure learning, the learned structure is used during both training and testing.

There are also some recent works [41, 65, 55] exploring how to perform data augmentation for graph contrastive learning, but they focus on automatically selecting the most appropriate transformations from a given pool to improve contrastive learning, rather than learning customized augmentation strategies for GNNs. More importantly, graph contrastive learning aims to learn transferable knowledge from abundant unlabeled data in an unsupervised setting and then generalize the learned knowledge to downstream tasks. Instead, graph augmentation usually works in a semi-supervised setting and then generalizes to the test set.
setting, i.e., the label information is available during training. The graph structure learning and contrastive learning are not newly born topics, and we refer readers to the recent surveys [49, 64].

3 Preliminaries

Notions. Given a graph \( G = (V, E) \), where \( V \) is the set of \( N = |V| \) nodes with features \( X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{N \times d} \) and \( E \) denotes the edge set. Each node \( v_i \in V \) is associated with a \( d \)-dimensional features vector \( x_i \), and each edge \( e_{i,j} \in E \) denotes a connection between node \( v_i \) and \( v_j \). The graph structure can also be denoted by an adjacency matrix \( A \in [0, 1]^{N \times N} \) with \( A_{i,j} = 1 \) if \( e_{i,j} \in E \) and \( A_{i,j} = 0 \) if \( e_{i,j} \notin E \). Consider a semi-supervised node classification task where only a subset of node \( V_L \) with corresponding labels \( Y_L \) are known, we denote the labeled set as \( D_L = (V_L, Y_L) \) and unlabeled set as \( D_U = (V_U, Y_U) \), where \( Y_U = V \setminus V_L \). The node classification task aims to learn a mapping \( \Phi : V \rightarrow Y \) on labeled data \( D_L \), so that it can be used to infer labels \( Y_U \).

Background on Graph Homophily Ratio. The homophily ratio is an important graph property that reflects the extent to which the graph structure adheres to the "label smoothness" criterion. The graph homophily ratio \( r \) can be defined as the fraction of intra-class edges in the graph, as follows

\[
r = \frac{\{(i, j) : (i, j) \in E \land y_i = y_j\}}{|E|}
\] (1)

where \( y_i \) and \( y_j \) are the ground-truth labels of node \( v_i \) and \( v_j \). In practice, the distribution space size of a discrete graph structure \( A \in [0, 1]^{N \times N} \) is \( 2^{N^2} \), making it tractable to directly estimate the distribution differences between two discrete graph structures. In this paper, we take the graph homophily as a desirable option to analyze the distribution shifts between the original and augmented graphs, thus measuring the severity of an algorithm suffering from the negative augmentation problem.

4 Methodology

In this section, we first make problem statements for graph augmentation in Sec. 4.1, highlight our motivations by analyzing the distribution shift between the original and augmented graphs in Sec. 4.2, then present a novel teacher-student Knowledge Distillation for Graph Augmentation (KDGA) framework in Sec. 4.3, and finally provide one of its specific instantiations in Sec. 4.4.

4.1 Problem Statement

Graph Representation Learning. From the perspective of statistical learning, the key of node classification is to learn a mapping \( p(Y | X, A) \) based on node features \( X \) and graph structure \( A \). The learned mapping can be used to infer labels \( Y_U \) on the graph structure \( A \) as shown in Fig. 1(a).

Graph Structure Learning. The goal of graph structure learning is to estimate a more accurate structure \( \hat{A} \) by another mapping \( p(\hat{A} | X, A) \) and then feed it into the mapping \( p(Y | X, \hat{A}) \) along with node features \( X \). Finally, the learned mapping \( p(Y | X, \hat{A}) \) can be used to infer labels \( Y_U \) on the estimated (high-quality) structure \( \hat{A} \) instead of the original structure \( A \) as shown in Fig. 1(b).

Graph Augmentation. Instead of directly working with the original graph, we would like to leverage graph augmentation to reasonably perturb the graph structure and learn more generalizable representations. In other words, we are interested in the following variant, as follows

\[
p(Y | X, A) = \sum_{\hat{A} \in [0,1]^{N \times N}} p(Y | X, \hat{A})p(\hat{A} | X, A)
\] (2)

where \( \hat{A} \in [0, 1]^{N \times N} \) is the augmented graph (structure). In practice, the distribution space size of \( \hat{A} \) is \( 2^{N^2} \), and it is intractable to enumerate all possible \( \hat{A} \) as well as estimate the exact values of the mappings \( p(Y | X, \hat{A}) \) and \( p(\hat{A} | X, A) \). Therefore, we approximate them by tractable functions as

\[
p(Y | X, A) = \sum_{\hat{A} \in [0,1]^{N \times N}} q_\theta(Y | X, \hat{A})q_\phi(\hat{A} | X, A)
\] (3)

where \( q_\theta(\cdot) \) and \( q_\phi(\cdot) \) are approximation functions for \( p(Y | X, \hat{A}) \) and \( p(\hat{A} | X, A) \) parameterized by \( \theta \) and \( \phi \), respectively. In practice, the function \( q_\theta(Y | X, \hat{A}) \) can be generally implemented by
GNNs, and the function $q_{\phi}(\hat{A} \mid X, A)$ can be implemented by graph augmentation methods to model the distributions of augmented graph structures. Once the model training is finished, the mapping $q_{\theta}(Y \mid X, A)$ can be used to infer labels $Y_{\hat{A}}$ on the original structure $A$ as shown in Fig. 1(c).

In summary, unlike graph representation and graph structure learning that leverage the same structure (A or $\hat{A}$) for both training and testing, the graph structures for training and testing in graph augmentation are completely different, which may lead to a potential negative augmentation problem.

### 4.2 Motivation: Potential Negative Augmentation Problem

One may create a model by specifying specific implementations for functions $q_{\phi}(Y \mid X, \hat{A})$ and $q_{\phi}(\hat{A} \mid X, A)$ and then optimize it by maximizing the posterior $p(Y \mid X, A)$ defined in Eq. (3). As we will explain here, however, this model may suffer from a potential negative augmentation problem caused by overly severe distribution shifts between the original and augmented graphs.

The distribution shift itself is not necessarily harmful; it is actually a neutral phenomenon. A proper distribution shift helps the model “see” more different graphs, enabling the nodes to receive more contextual information, thus improving generalization; however, an overly severe distribution shift can lead to a potential negative augmentation problem. To illustrate it, we consider a node $v_1$ (id 129) of class $C_1$ from the real-world Wisconsin dataset in Fig. 2(a), it is initially connected to a node with the same class $C_1$ and three nodes from another class $C_2$ in the original structure $A$. During the training process, the original structure $A$ and node features $X$ are fed together into $q_{\phi}(\hat{A} \mid X, A)$ to generate an augmented structure $\hat{A}$. Under the downstream supervision, it disconnects from three nodes from class $C_2$ and reconnects with three nodes from the same class $C_1$, resulting in an...
augmented structure \( \hat{A} \) with a much higher homophily ratio, that is, an overly severe distribution shift between the original and augmented graphs from the perspective of graph homophily property.

As a result, a model trained on the augmented structure \( \hat{A} \) can successfully predict node \( i \) as class \( C_1 \), but make a wrong prediction \( C_2 \) for node \( i \) when tested on the original structure \( A \), which is termed as "negative augmentation". Furthermore, we plot the homophily ratios of the original structure \( A \) and augmented structure \( \hat{A} \) on eight datasets in Fig. 2(b), from which we can observe significant distribution shifts between the two graphs. Moreover, while the above analysis is developed on a heterophily (Wisconsin) graph, we find that the identified distributional shift also exists in homophily graphs, only in a different direction. Please see Sec. 5.3 for detailed experimental settings and results.

### 4.3 Knowledge Distillation for Graph Augmentation (KDGA)

The distribution shift is essentially a trade-off between better generalizability and higher risks of negative augmentation. However, the optimal distribution shift may vary from dataset to dataset, or even from node to node, making it challenging to directly control the levels of distribution shifts. In this paper, we have not attempted to control or prevent distribution shifts. Instead, we allow for the existence of any level of distribution shifts, but we reduce their negative impact, i.e., the potential negative augmentation problem, by the proposed KDGA framework, which gradually distills the contextual information from the augmented graphs into a student model tested on the original graph.

The idea of KDGA is straightforward, yet as we will see, extremely effective. In our case, we first generate soft distributions \( z_i^T \) and \( z_i^S \) for node \( v_i \) with the teacher and student models, respectively. The knowledge distillation is first introduced in [14], where knowledge was transferred from a Teacher Model.

The distribution shift is essentially a trade-off between better generalizability and higher risks of negative augmentation. However, the optimal distribution shift may vary from dataset to dataset, or even from node to node, making it challenging to directly control the levels of distribution shifts. In this paper, not to get a simpler student model, we adopt the knowledge distillation framework to address the identified negative augmentation problem caused by overly severe distribution shifts between the two graphs. In short, we extract the knowledge of any teacher model trained on the augmented graphs and inject it into a student model tested on the original graph as in Fig. 1(d).

**Teacher Model.** The teacher model can be implemented by any GNN, which takes node features \( X \) and augmented structure \( \hat{A} \) as input and learn latent node representations via neighborhood feature aggregation. Considering a \( L \)-layer GNN \( f_\theta(X, \hat{A}) \), the formulation of the \( l \)-th layer is as follows

\[
h_i^{(l+1)} = \text{UPDATE}^{(l)} \left( h_i^{(l)}, \text{AGGREGATE}^{(l)} \left( \{ h_j^{(l)} : v_j \in N_i^{\hat{A}} \} \right) \right)
\]

where \( 0 \leq l \leq L - 1 \), \( h_i^{(0)} = x_i \) is the input feature, and \( N_i^{\hat{A}} \) is the neighborhood of node \( v_i \) in the augmented structure \( \hat{A} \). After \( L \) message-passing layers, the final node embedding \( h_i^{(L)} \) is passed to a linear prediction head \( g^T(\cdot) \) to obtain logits \( z_i^T = g^T(h_i^{(L)}) \), and the model is trained by a cross-entropy loss \( H(\cdot) \) with ground-truth labels \( y_i \), given by

\[
\mathcal{L}_{STP}^T = \sum_{i \in V_L} H(y_i; \text{softmax}(z_i^T)).
\]

**Student Model.** The student model \( f_\theta(X, A) \) shares the parameters \( \theta \) with the teacher model, but differs in that it takes the original structure \( A \) as input, as shown in Fig. 3(c). Besides, an additional linear prediction head \( g^S(\cdot) \) is used to map the node embedding \( h_i^{(L)} \) to logits \( z_i^S = g^S(h_i^{(L)}) \). As already explained earlier, the augmented graphs enable the teacher model to receive richer contextual information, which helps to improve model generalization. To allow the student model tested on the original structure to also benefit from it, we consider the contextual neighborhood information from both original and augmented structures and distill them into the student model, defined as

\[
\mathcal{L}_{GKD} = \frac{\tau_1^2}{|V|} \sum_{i \in V} \sum_{j \in (N_i^{\hat{A}} \cap N_i^{A}) \cup i} \mathcal{D}_{KL} \left( \text{softmax}(z_i^T/\tau_1) \cdot \text{softmax}(z_i^S/\tau_1) \right)
\]

where \( \tau_1 \) is the distillation temperature, and \( \tau_1^2 \) is used to keep the gradient stability of this loss [14].

**Discussions.** While a large number of methods on graph knowledge distillation [61, 53, 60] have been proposed, most of them adopt the standard teacher-student knowledge distillation framework.
as shown in Fig. 3(a), where the inputs to both teacher and student models are the same (structure). Despite many progresses, their contributions have mostly focused on the special design of the teacher or student models. For example, CPF [54] proposes to distill knowledge from a teacher GNN to a student MLP, but it specifically incorporates label propagation [16] into the student model to improve performance. In contrast, GDK [11] utilizes label propagation in the teacher model to fully exploit both feature and topological information. In our proposed KDGA framework, the graph structures fed to the teacher and student models are completely different. Moreover, unlike the scheme in Fig. 3(b) where two parameter-independent teacher and student models are used, we adopt the architecture shown in Fig. 3(c) where the GNN parameters are shared but with two independent prediction heads to increase discriminability. The behind motivation is that a parameter-independent student model has the risk of quickly fitting with the original structure under the optimization of downstream supervision, while failing to take full advantage of rich contextual information from the augmented graphs. A detailed comparison of parameter-independent and parameter-shared schemes is reported in Table. 2.

$$A$$

$$B$$

$$C$$

Figure 3: (a) standard teacher-student distillation; (b) distillation with parameter-independent teacher and student models; (c) distillation with parameter-shared teacher and student models.

### 4.4 Instantiating KDGA with GraphAug

In practice, any existing graph augmentation method can be used to instantiate the proposed KDGA framework and achieve consistent improvements over the vanilla implementations, as shown in Table. 1. In this subsection, we adopt a probabilistic generative-based graph augmentation method to model the function $q_0(\hat{A} | X, A)$, termed GraphAug, and use it to instantiate our KDGA framework. Specifically, we introduce a set of discrete variables $\Lambda = \{\lambda_{i,j}\}_{i,j=1}^N$ to model the distribution of the augmented graph, where $\lambda_{i,j} \in \{0, 1\}$ denotes the augmentation probability between node $v_i$ and $v_j$. Moreover, we avoid estimating the probability $p(\lambda_{i,j} | \mu_{i,j})$ using independent local parameter $\mu_{i,j}$ and instead fits a shared neural network to estimate it. Specifically, we first transform the input to a low-dimensional hidden space, done by multiplying the node features with a parameter matrix $\mathbf{W} \in \mathbb{R}^{F \times d}$, that is, $e_i = \mathbf{W}x_i$. Then, we directly parameterize the probability $\lambda_{i,j}$ as

$$p(\lambda_{i,j} | X, A) = \sigma(e_i e_j^T)$$

(7)

where $\sigma(\cdot)$ is an element-wise sigmoid function. Next, to sample discrete augmented graphs from the learned augmentation distribution and make the sampling process differentiable, we adopt Gumbel-Softmax sampling [17]. Specifically, the sampling process can be formulated as

$$\hat{A}_{i,j} = \left[ \frac{1}{1 + \exp(-\log M_{i,j} + G)} + \frac{1}{2} \right], \text{where } M_{i,j} = \alpha p(\lambda_{i,j} | X, A) + (1 - \alpha)A_{i,j}$$

(8)

where $\alpha \in [0, 1]$ is the fusion factor to control the intensity of the graph augmentation, $\tau_2$ is the augmentation temperature, and $G \sim \text{Gumbel}(0, 1)$ is a gumbel random variate.

To warm-up the proposed GraphAug module, we first pre-train it with loss $L_{Aug} = \frac{1}{|V|} \mathcal{H}(A_{i,j}, \hat{A}_{i,j})$, where $\mathcal{H}(\cdot)$ denotes the cross-entropy loss. Besides, we use classification loss $L_{Cla} = \frac{1}{|V|} \sum_{i \in V} \left( \mathcal{H}(y_i, \text{softmax}(z_i^T)) + \mathcal{H}(y_i, \text{softmax}(z_i^T)) \right)$ to pre-train the teacher and student models until it converges. Finally, the total loss to train the whole framework is defined as follows

$$L_{total} = L_{Cla} + \lambda L_{Aug} + \kappa L_{GKD}$$

(9)

where $\lambda$ and $\kappa$ are the weights to balance the influence of the two losses $L_{Aug}$ and $L_{GKD}$.
Table 1: Accuracy ± std (%) on eight datasets (as well as their homophily ratios), with three GNN architectures and five graph augmentation methods considered. The best metrics are marked by bold.

<table>
<thead>
<tr>
<th>BaseGNN</th>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Cornell</th>
<th>Chameleon</th>
<th>Squirrel</th>
<th>Actor</th>
<th>Wisconsin</th>
<th>Texas</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.81</td>
<td>0.74</td>
<td>0.30</td>
<td>0.23</td>
<td>0.22</td>
<td>0.22</td>
<td>0.21</td>
<td>0.11</td>
</tr>
<tr>
<td>Vanilla</td>
<td></td>
<td>81.5±0.8</td>
<td>71.0±0.7</td>
<td>57.0±0.7</td>
<td>59.8±1.6</td>
<td>36.9±1.7</td>
<td>30.3±0.6</td>
<td>59.8±0.9</td>
<td>59.5±3.3</td>
</tr>
<tr>
<td>DropEdge [35]</td>
<td></td>
<td>82.2±0.7</td>
<td>71.9±0.6</td>
<td>59.3±2.9</td>
<td>61.2±1.3</td>
<td>38.1±1.5</td>
<td>30.9±1.0</td>
<td>61.8±4.4</td>
<td>62.3±6.4</td>
</tr>
<tr>
<td>AdaEdge [2]</td>
<td></td>
<td>82.3±0.7</td>
<td>69.7±0.8</td>
<td>57.8±4.3</td>
<td>59.5±2.3</td>
<td>37.6±1.4</td>
<td>31.4±1.2</td>
<td>60.4±4.7</td>
<td>58.8±5.0</td>
</tr>
<tr>
<td>SSL [63]</td>
<td></td>
<td>83.8±0.7</td>
<td>72.9±0.6</td>
<td>58.8±3.2</td>
<td>60.4±2.1</td>
<td>39.5±1.9</td>
<td>30.5±1.2</td>
<td>62.8±4.5</td>
<td>63.1±6.6</td>
</tr>
<tr>
<td>GraphMix [44]</td>
<td></td>
<td>83.9±0.7</td>
<td>74.7±2.6</td>
<td>60.5±3.7</td>
<td>61.2±2.3</td>
<td>40.1±1.5</td>
<td>31.4±0.8</td>
<td>62.9±5.0</td>
<td>62.3±6.4</td>
</tr>
</tbody>
</table>

**GCN**

| GAUG [62] | 83.6±0.7 | 73.3±1.1 | 55.8±4.0 | 59.3±1.4 | 36.3±0.8 | 27.9±0.9 | 57.5±1.1 | 58.0±4.2 |
| MH-Aug [32] | 85.3±8.5 | 73.6±0.8 | 63.2±3.6 | 63.0±1.2 | 46.2±2.0 | 33.3±0.8 | 65.0±2.5 | 67.4±3.8 |
| MH-Aug (w/ KDGA) | 83.6±0.7 | 73.3±1.1 | 55.8±4.0 | 59.3±1.4 | 36.3±0.8 | 27.9±0.9 | 57.5±1.1 | 58.0±4.2 |

**SAGE**

| GAUG [62] | 82.0±0.7 | 72.7±0.7 | 74.9±4.2 | 58.2±1.3 | 40.5±0.8 | 34.4±1.1 | 80.7±6.1 | 82.0±4.3 |
| GAUG (w/ KDGA) | 84.5±0.7 | 73.4±0.7 | 80.6±3.5 | 61.8±1.6 | 46.4±1.4 | 36.6±0.7 | 85.5±3.2 | 84.5±3.6 |

**GAT**

| GAUG [62] | 82.3±0.8 | 72.8±0.7 | 75.9±5.0 | 58.8±1.4 | 40.2±1.2 | 33.2±0.7 | 79.9±2.1 | 81.9±4.6 |
| GAUG (w/ KDGA) | 84.8±0.8 | 73.5±0.5 | 81.4±2.8 | 61.0±1.8 | 45.6±0.9 | 36.9±1.4 | 84.5±3.3 | 84.8±3.8 |

5 Experiments

**Datasets.** The effectiveness of the proposed KDGA framework is evaluated on eight datasets. We use two commonly used homophily graph datasets, namely Cora [38] and Citeseer [12] as well as six heterophily graph datasets: Cornell, Texas, Wisconsin, Actor [42], Chameleon and Squirrel [37]. A statistical overview of these datasets is available in Appendix B. We defer the implementation details and the best hyperparameter settings for each dataset to Appendix B and supplementary material.

**Baselines.** As a general framework, KDGA can be combined with any GNN architecture and existing graph augmentation methods. In this paper, we consider three GNN architectures, GCN [21], GraphSAGE [13], and GAT [43]. Besides, to demonstrate the applicability of KDGA to various graph augmentation methods in addition to the proposed GraphAug, we also consider two state-of-the-art learning-based baselines, GAUG [62] and MH-Aug [32]. In particular, two heuristics methods, DropEdge and AdaEdge, are also included in the comparison as baselines. Moreover, we also compare KDGA with two self-supervised methods: (1) GraphMix [44], a regularization method that performs linear interpolation between two data on graphs, and (2) SSL [63], that proposes two self-supervised tasks to fully exploit available information embedded in the graph structure. Each set of experiments is run five times with different random seeds, and the average performance is reported.
5.1 Comparative Results

To evaluate the powerful capabilities of the proposed KDGA framework, we instantiate it with three learning-based graph augmentation methods, GAUG, MH-Aug, and GraphAug. The experiments are conducted on eight datasets with three different GNN architectures. From the experimental results shown in Table 1, we can make the following observations: (1) Two heuristic graph augmentation methods, DropEdge and AdaEdge, can improve the performance of the vanilla GNNs overall. However, such improvements are usually very limited and do not work for all datasets and GNN architectures. For example, on the Citeseer dataset, the performance of AdaEdge drops over the vanilla GNNs by 1.9% (GCN), 1.7% (GraphSAGE), and 2.3% (GAT), respectively. (2) There are huge gaps in the effectiveness of three learning-based augmentation methods on homophily and heterophily graphs. While these methods can significantly improve performance on homophily graphs, their performance gains on heterophily graphs are greatly reduced and even detrimental. For example, with GCN as the GNN architecture, the performance of GAUG improves by 2.1% on Cora, but drops by 1.5% and 1.2% on Texas and Cornell. Such negative augmentation is mainly caused by the overly severe distribution shift between the original and augmented graphs as analyzed in Sec. 4.2. (3) The proposed KDGA framework can consistently improve the performance of vanilla graph augmentation methods across three GNN architectures on all eight datasets, especially for those heterophily graphs. For example, with GCN as the GNN architecture, the performance of GraphAug can be improved by 9.7% and 8.6% on the Squirrel and Texas datasets. (4) Two semi-supervised approaches, SSL and GraphMix, can achieve comparable or even better performance than learning-based graph augmentation, especially on heterophily graphs. However, by combining with KDGA, GAUG, MH-Aug, and GraphAug outperform both SSL and GraphMix by a large margin overall.

5.2 Analysis on the Distribution Shift and Negative Augmentation

Next, we qualitatively and quantitatively analyze the distribution shift between the original and augmented graphs and explain how it can cause a potential negative augmentation. Without loss of generality, we consider GCN as the GNN architecture and GraphAug as the augmentation method.

Visualizations of Neighborhood Variations. First, we pick a node with the most drastic neighborhood variations and visualize its neighborhood on the original graph structure (top) and augmented graph structure (bottom), where each node is colored according to its ground-truth label. The visualizations show that there would be a huge gap between the neighborhoods of the original and augmented graphs, which causes a model that is well trained on the augmented graph to predict poorly on the original graph during testing. Taking the Wisconsin dataset as an example, the selected node is connected to four nodes from the same class in the augmented graph, so it can be well trained to make correct predictions. However, its neighborhood context is completely changed in the original graph, where the node is connected to three nodes from different classes, so it will be predicted with high confidence to an incorrect class.

Training Curves of Homophily Ratios. We plot in Fig. 5 the training curves (w/o GKD Loss) of the homophily ratios of the original and augmented graphs during training. It can be seen that their gaps are enlarged as training proceeds, which indicates that the distribution of the augmented graphs is gradually shifting from the original graph. This shift may even reach 0.5 for some datasets (e.g., Texas), in which case the graph homophily property is completely reversed. More importantly, we find that the direction of distribution shifts may be completely opposite for homophily and heterophily graphs, which makes it more challenging to solve the negative augmentation problem.
which is caused by overly severe distribution shifts between the original and augmented graphs. To address this problem, we propose a novel Knowledge Distillation for Graph Augmentation (KDGA) framework by directly distilling contextual information from a teacher model trained on the augmented graphs into a partially parameter-shared student model. Extensive experiments show that KDGA outperforms the vanilla implementations of existing augmentation methods and GNN architectures. Limitations still exist, such as KDGA requires an initial raw graph structure for augmentation and cannot be applied to those structure-unknown scenarios, which will be left for future work.

7 Acknowledgement

This work is supported in part by Ministry of Science and Technology of the People’s Republic of China (No. 2021YFA1301603) and National Natural Science Foundation of China (No. U21A20427).
References


[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph 


26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 338–348, 
2020.

[25] Xiaofeng Liu, Yang Zou, Lingsheng Kong, Zhihui Diao, Junliang Yan, Jun Wang, Site Li, Ping Jia, and Jane 
You. Data augmentation via latent space interpolation for image classification. In 2018 24th International 

[26] Zicheng Liu, Siyuan Li, Ge Wang, Cheng Tan, Lifong Wu, and Stan Z Li. Decoupled mixup for data-

[27] Zicheng Liu, Siyuan Li, Di Wu, Zhiyuan Chen, Lifong Wu, Jianzhu Guo, and Stan Z Li. Unveiling the 

[28] Zihan Liu, Yun Luo, Lifong Wu, Siyuan Li, Zicheng Liu, and Stan Z Li. Are gradients on graph structure 

[29] Zihan Liu, Yun Luo, Zelin Zang, and Stan Z Li. Surrogate representation learning with isometric mapping 
for gray-box graph adversarial attacks. In Proceedings of the Fifteenth ACM International Conference on 
Web Search and Data Mining, pages 591–598, 2022.

IEEE, 2018.

[31] Sarah O’Gara and Kevin McGuinness. Comparing data augmentation strategies for deep image classifica-
tion. 2019.

[32] Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, 
and Hyunwoo J Kim. Metropolis-hastings data augmentation for graph neural networks. Advances in 


[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: 
a simple way to prevent neural networks from overfitting. The journal of machine learning research, 

[35] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph 


