IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 25, 2021, accepted September 2, 2021, date of publication September 10, 2021,
date of current version September 21, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3111908*

# Learning Augmentation for GNNs With Consistency Regularization

**HYEONJIN PARK[1], SEUNGHUN LEE[1], DASOL HWANG[1], JISU JEONG[2,3], KYUNG-MIN KIM[2,3], JUNG-WOO HA[2], AND HYUNWOO J. KIM[1]**

[1]Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea
[2]Naver AI Laboratory, Seongnam 13561, Republic of Korea
[3]Naver CLOVA, Seongnam 13561, Republic of Korea

Corresponding author: Hyunwoo J. Kim (hyunwoojkim@korea.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP), Naver Corporation, grant funded by Korean Government, MSIT, (Research on CPU vulnerability detection and validation) under Grant 2019-0-00533, and in part by the ICT Creative Consilience Program, supervised by the IITP, under Grant IITP-2021-2020-0-01819.

**ABSTRACT** Graph neural networks (GNNs) have demonstrated superior performance in various tasks on graphs. However, existing GNNs often suffer from weak-generalization due to sparsely labeled datasets. Here we propose a novel framework that learns to augment the input features using topological information and automatically controls the strength of augmentation. Our framework learns the augmentor to minimize GNNs' loss on unseen labeled data while maximizing the consistency of GNNs' predictions on unlabeled data. This can be formulated as a meta-learning problem and our framework alternately optimizes the augmentor and GNNs for a target task. Our extensive experiments demonstrate that the proposed framework is applicable to any GNNs and significantly improves the performance of graph neural networks on node classification. In particular, our method provides 5.78% improvement with Graph convolutional network (GCN) on average across five benchmark datasets.

**INDEX TERMS** Graph neural networks, augmentation, semi-supervised learning, meta-learning.

## I. INTRODUCTION

Graph neural networks (GNNs) [1] have been widely used for representation learning on graph-structured data due to their superior performance in various applications such as node classification [2]–[4], link prediction [5]–[8] and graph classification [9]–[11]. They have been proven effective by achieving state-of-the-art performance on diverse datasets such as social networks [12], [13], citation networks [4], physics [14], molecules [15], and knowledge graphs [16], [17]. However, GNNs often suffer from *weak-generalization* due to small and sparsely labeled graph benchmark datasets [18].

One solution to the overfitting problem is data augmentation. It increases the amount and diversity of data and improves the generalization of neural networks that are trained on randomly augmented samples. Conventional data augmentation includes, for instance in image recognition, simple transformations such as random cropping, cutout, additive Gaussian noise, or blurring. Intuitively, data augmentation teaches the invariances to neural networks for a target task. The key to effective data augmentation is to find an optimal combination of the operations and augmentation strength. Since every task needs different invariances, data augmentation has been manually optimized with domain knowledge for each dataset. To reduce the manual efforts and scale up the data augmentation to new tasks and domains, *learning augmentation* has been recently proposed; this includes AutoAugment [19] that optimizes hyperparameters of a non-differentiable augmentor via reinforcement learning; PDB [20] generates augmentation policy schedules using population-based training; Fast AutoAugment [21] searches the optimal policy via Bayesian optimization; Faster AutoAugment [22] approximates the gradient information and optimizes the augmentation policies using straight-through estimator [23].

The learning augmentation paradigm, however, has less been studied in the graph domain. One technical challenge is that it is difficult to define simple operations to augment graph data without changing the meaning. The recent advances in

The associate editor coordinating the review of this manuscript and approving it for publication was Felix Albu.

learning augmentation mainly focused on finding an optimal policy to combine existing operations and their strength rather than learning the operations. But only a few simple operations for graph data augmentation have been proposed in the literature such as DropEdge [18], DropNode [24], attribute masking [25]. In the last year, with these simple operations, [26], [27] studied frameworks to learn graph data augmentation but they are limited to perturbations of adjacency matrices. In this work, we propose a framework to learn graph data augmentation that transforms input features instead of adjacency matrices. Our framework learns topology-aware input feature transformations and adaptively controls the strength of augmentation based on a GNN's performance. It is formulated as a meta-learning problem and our framework explicitly maximizes the generalization power of graph neural networks that are trained on augmented data. We observe that scarcely labeled data may cause *meta-overfitting*, i.e., overfitting in meta-learning, and result in excessive and ineffective data augmentation. Inspired by the success of consistency regularization in representation learning [28]–[32], we regularize our augmentation model by maintaining the consistency of GNN's predictions leveraging unlabeled data. To our knowledge, this is the first work to study consistency regularization in the context of learning augmentation. In a conventional setting, consistency regularization was used for representation learning with a manually designed augmentation method. Our experiments show that consistency regularization is effective to reduce meta-overfitting as well.

Our **contributions** are summarized as follows:

- Unlike existing graph data augmentations based on simple removing/adding edges, dropping nodes or masking attributes, we propose a novel framework that learns operation to augment input features and adaptively controls the strength of augmentation.
- We study consistency regularization in the context of learning augmentation. To our knowledge, this is the first work to show the effectiveness of the consistency regularizer in learning augmentation and reducing meta-overfitting.
- We demonstrate the effectiveness of our framework to improve the generalization power of GNNs. Overall, our method provides 5.78%, 2.63% and 3.14% improvements on average across various datasets compared to the vanilla baselines GCN [4], GraphSAGE [2], and SGC [33], respectively.

## II. RELATED WORK

### A. GRAPH NEURAL NETWORKS

Graph neural networks (GNNs) have been widely adopted in representation learning on graphs for various tasks such as node classification, link prediction, and graph classification. They can be categorized as spectral approaches [4], [34], [35] and non-spectral (spatial) approaches [2], [3], [36]. [4] presented GCN which developed spectral approaches into spatial approaches by localizing first-order approximation of graph convolutions. GraphSAGE [2] is one of non-spectral approaches and learns to generate embeddings by sampling and aggregating features from the neighborhood of nodes. The existing works in graph representation learning only leverage a small subset of nodes. Therefore, to fully utilize a large amount of unlabeled data, recent works for semi-supervised learning [24], [37] have emerged.

### B. GRAPH AUGMENTATION

Data augmentation is an effective technique to improve generalization by increasing the diversity of training data. It is becoming the *de facto* necessity for modern machine learning model training to employ simple data augmentation (e.g., image rotation, cropping, flipping, translation, and so on). Despite its effectiveness, few approaches have been explored to graph domain and most of which is based on the graph topology. DropEdge [18] is a simple graph data augmentation strategy to randomly remove a certain number of edges from the graph. Similarly, a method to propagate the perturbed node features by randomly dropping on a node-based was proposed in [24]. [27] introduces AdaEdge which is to adaptively adjust the graph topology by removing the inter-class edges and adding the intra-class edges based on the prediction. The proposed method is to augment node features based on the topological structure of the graph by interpolating between original input features and augmented features.

### C. META-LEARNING FOR GRAPHS

Meta-learning has shown success in diverse tasks [38] and there are some works applying meta-learning to data augmentation on images [39]–[41]. Recently, it is also applied for non-Euclidean domains. For instance, Meta-GNN [42] and Meta-Graph [43] use a gradient-based meta-learning method in a few-shot setting for node classification and link prediction, respectively. MetaR [44] is a meta-learning framework leveraging a knowledge graph for few-shot link prediction. [45] proposed a meta-learner that explicitly relates tasks describing the relations of predicted classes. [46] proposed G-META which can learn transferable knowledge faster via meta gradients by leveraging local subgraphs. Most works for graphs are specialized in a few-shot setting and are used to learn a learner. Recently, SELAR [47] can learn how to assist the primary task by optimizing a weight function not a classifier via meta-learning. In this paper, we propose a graph augmentation strategy in a meta-learning manner which learns how to augment input features to improve performance on unseen data.

## III. METHOD

We present a novel framework (**AugGCR**) that learns data **Aug**mentation for **G**NNs with the **C**onsistency **R**egularization to enhance the generalization power of GNNs. Our framework includes 1) an augmentor that generates augmented input features taking into account graph topologies and 2) a novel learning strategy to alternately train a GNN and

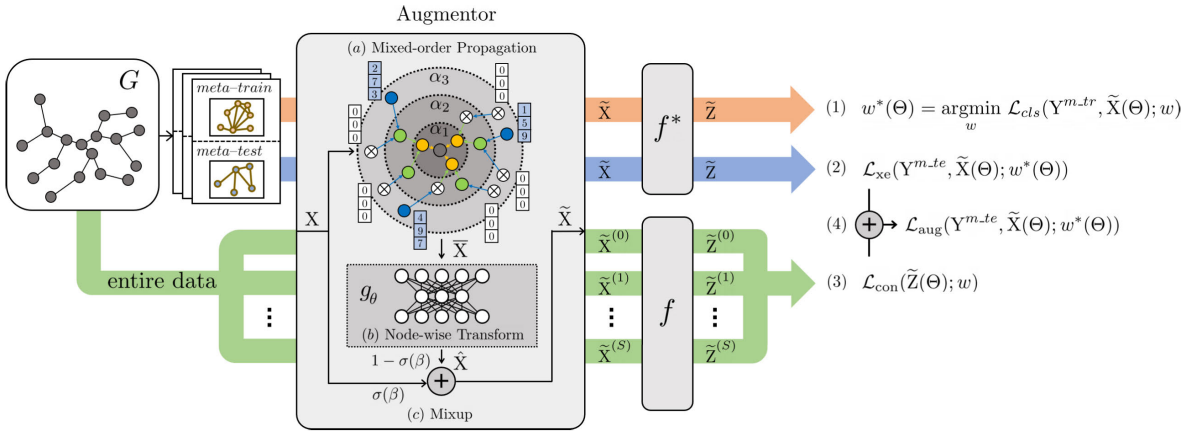



**FIGURE 1.** Overview of AugGCR. AugGCR learns to augment input features using an augmentor (mid) which is parameterized with $\Theta = (\{\alpha_k\}_{k=0}^{K}, \theta, \beta)$ and is alternately optimized with a target model $f(\cdot; w)$. Our augmentation consists of three steps: (a) *Learnable mixed-order propagation* randomly drops node features and propagates the node features to $k$-hop neighbors with weight $\alpha_k$ (b) *Node-wise transformation* applies a two-layer MLP denoted by $g_\theta$ (c) *Mixup* interpolates the augmented samples with original features. Our optimization involves three different data splits: meta-train data ($m\_tr$), meta-test data ($m\_te$), entire data including unlabeled nodes. The training procedure of Augmentor can be divided into three steps. First, (1) update the dummy target model $f^*$ with meta-train data and get a cross-entropy loss, $\mathcal{L}_{\mathrm{xe}}$, with (2) on meta-test data. Second, get a consistency regularization loss, $\mathcal{L}_{\mathrm{con}}$, in (3) with $S$ different augmentation $\widetilde{\mathrm{X}}$ from entire data. Lastly, optimize $\Theta$ with $\mathcal{L}_{\mathrm{aug}}$ in (4). (Best viewed in color).

the augmentor avoiding both overfitting and meta-overfitting to scarce label information.

### A. TOPOLOGY-AWARE AUGMENTATION

Given an input graph $G = (V, E)$ with $|V|$ nodes, their input features $\mathrm{X} \in R^{|V|\times d}$, and adjacency matrix $\mathrm{A} \in R^{|V|\times|V|}$, our augmentor generates the augmented features $\widetilde{\mathrm{X}} \in R^{|V|\times d}$ taking account into the graph topology as follows:

$$\widetilde{\mathrm{X}} \sim \mathrm{Aug}(\mathrm{X}; \mathrm{A}, \Theta), \tag{1}$$

where $\mathrm{Aug}(\cdot)$ is the augmentor and $\Theta$ is its model parameters. Our augmentor consists of three components: a learnable mixed-order propagation, node-wise nonlinear transformation, and mix-up with original features.

**Algorithm 1** Augmentor

**Input:** training feature X, $\hat{\mathrm{A}}$; drop ratio $\delta$, # hop $K$;
**Output:** augmented feature $\widetilde{\mathrm{X}}$

1: **function** Aug
2: $\quad \mathrm{X}' \leftarrow \mathrm{DropNode}(\mathrm{X}, \delta)$
3: $\quad \overline{\mathrm{X}}' \leftarrow \sum_{k=0}^{K} \alpha_k \hat{\mathrm{A}}^k \mathrm{X}'$
4: $\quad \hat{\mathrm{X}}' \leftarrow g(\overline{\mathrm{X}}'; \theta)$
5: $\quad \widetilde{\mathrm{X}} \leftarrow \sigma(\beta) \cdot \mathrm{X} + (1 - \sigma(\beta))\hat{\mathrm{X}}'$
6: **end function**

#### 1) LEARNABLE MIXED-ORDER PROPAGATION

To perturb features based on graph topologies, our augmentor first randomly drops node features as DropNode [24]. The features from DropNode with drop ratio $\delta$ can be formally written as $\mathrm{X}' = \mathrm{diag}(\epsilon_1, \ldots, \epsilon_N)\mathrm{X}$, where $\epsilon_i \sim \mathrm{Bernoulli}(1-\delta)$. Then the randomly dropped node features are propagated by the power series of normalized adjacency matrix as in SGC [33]. For augmentation, [24] adopted the mixed-order propagation, *i.e.*, $\overline{\mathrm{X}} = \sum_{k=0}^{K} \frac{1}{K+1} \hat{\mathrm{A}}^k \widetilde{\mathrm{X}}$, where $\hat{\mathrm{A}} = \widetilde{\mathrm{D}}^{-\frac{1}{2}}(\mathrm{A}+\mathrm{I})\widetilde{\mathrm{D}}^{-\frac{1}{2}}$ is a normalized adjacency matrix and $\widetilde{\mathrm{D}}$ denotes the degree matrix of $\mathrm{A} + \mathrm{I}$. Note that, $\hat{\mathrm{A}}^0$ is the identity matrix. We extend this feature perturbation with learnable weights $\{\alpha_k\}_{k=0}^{K}$ for the power series of $\hat{\mathrm{A}}$, *i.e.*, $\overline{\mathrm{X}} = \sum_{k=0}^{K} \alpha_k \hat{\mathrm{A}}^k \mathrm{X}$. Our preliminary experiment shows that the coefficients allow more flexible feature augmentation.

#### 2) NODE-WISE TRANSFORMATION

In the second step, the augmented node features obtained by the learnable mixed-order propagation are further transformed by a simple two-layer perceptron, *i.e.*, $\hat{\mathrm{X}} = g(\overline{\mathrm{X}}; \theta)$. As mentioned in [24], the multi-layer perceptron (MLP) can be replaced with any neural networks.

#### 3) MIXUP WITH ORIGINAL FEATURES

Lastly, our augmentor generates the final augmented node features by adapatively averaging the perturbed feature matrix $\hat{\mathrm{X}}$ and the original node feature matrix X as:

$$\widetilde{\mathrm{X}} = \sigma(\beta)\mathrm{X} + (1 - \sigma(\beta))\hat{\mathrm{X}} \tag{2}$$

where $\beta$ is a learnable parameter and $\sigma$ is a sigmoid function. The mixup coefficient $\beta$ can control the strength of augmentation by adjusting the influence of original feature matrix $X$ on augmented features matrix $\widetilde{\mathrm{X}}$. Target models' sensitivity to $\beta$ will be discussed in Section IV-D. Our augmentation process is summarized in Algorithm 1.

### B. STRATEGIES FOR LEARNING AUGMENTATION

We present our meta-learning formulation to learn the augmentor with consistency regularization to reduce meta-overfitting of the augmentor.

#### 1) LEARNING TO AUGMENT VIA META-LEARNING

The objective of our augmentation model is to generate data to improve learner's generalization power. In other words, the objective is to maximize the accuracy of a learner GNN on unseen meta-test data $D^{m\_te}$. Given meta-train data $D^{m\_tr} = \{(\mathrm{X}_i^{m\_tr}, \mathrm{Y}_i^{m\_tr})\}_{i=1}^{T}$ and meta-test data $D^{m\_te} = \{(\mathrm{X}_i^{m\_te}, \mathrm{Y}_i^{m\_te})\}_{i=1}^{U}$, where both of which belong to a training set, learning augmentation for a GNN parameterized by $w$ is naturally formulated into a meta-learning problem as:

$$\begin{aligned} &\min_{\Theta, w^*} \mathcal{L}_{\mathrm{aug}}(\mathrm{Y}^{m\_te}, \widetilde{\mathrm{X}}(\Theta); w^*(\Theta)) \\ &\text{s.t. } w^*(\Theta) = \operatorname*{argmin}_{w} \mathcal{L}_{\mathrm{cls}}(\mathrm{Y}^{m\_tr}, \widetilde{\mathrm{X}}(\Theta); w), \end{aligned} \tag{3}$$

where $\widetilde{\mathrm{X}}(\Theta)$ is the augmented samples from our augmentor parametrized by $\Theta$, and $\mathcal{L}_{\mathrm{cls}}(\cdot)$ and $\mathcal{L}_{\mathrm{aug}}(\cdot)$ are the cross-entropy $\mathcal{L}_{\mathrm{xe}}(\cdot)$ as a surrogate loss function for the accuracy on meta-train data and meta-test data respectively. Note that $\widetilde{\mathrm{X}}(\Theta)$ is used for both losses since in node classification the input features of neighbor nodes are utilized by GNNs. More concretely, $\mathcal{L}_{\mathrm{cls}}(\cdot)$ in (3) is defined as

$$\mathcal{L}_{\mathrm{cls}}(\mathrm{Y}^{m\_tr}, \widetilde{\mathrm{X}}; w) = \mathcal{L}_{\mathrm{xe}}(\mathrm{Y}^{m\_tr}, \widetilde{\mathrm{X}}; w) = -\sum_{i \in \mathcal{I}^{tr}} \mathrm{Y}_i \log(\widetilde{\mathrm{Z}}_i), \tag{4}$$

where $\mathcal{I}^{tr}$ is the indices of the meta-train data and $\widetilde{\mathrm{Z}} = f(\widetilde{\mathrm{X}}; w)$ is the predictions of the learner GNN. $\mathcal{L}_{\mathrm{aug}}(\cdot)$ is similarly defined with $\mathcal{I}^{te}$, which is the indices of the meta-test data. Although the formulation in (3) is seemingly plausible, it is problematic when the meta data is small. It suffers from *meta-overfitting*. The augmentor maximizes the accuracy of the learner GNN on the small meta-test data rather than improving its generalization power. This often leads to the GNN's overfitting.

#### 2) CONSISTENCY REGULARIZATION FOR AUGMENTATION

We impose consistency regularization to address meta-overfitting and leverage underutilized node features for learning augmentation. Now, the loss for the augmentor is defined as

$$\begin{aligned} &\mathcal{L}_{\mathrm{aug}}(\mathrm{Y}^{m\_te}, \widetilde{\mathrm{X}}(\Theta); w^*(\Theta)) \\ &\quad = \mathcal{L}_{\mathrm{xe}}(\mathrm{Y}^{m\_te}, \widetilde{\mathrm{X}}(\Theta); w^*(\Theta)) + \mathcal{L}_{\mathrm{con}}(\widetilde{\mathrm{Z}}(\Theta); w), \end{aligned} \tag{5}$$

where $\mathcal{L}_{\mathrm{xe}}(\cdot)$ and $\mathcal{L}_{\mathrm{con}}(\cdot)$ are the cross-entropy and consistency loss. Specifically, consistency loss is defined as

$$\mathcal{L}_{\mathrm{con}}(\widetilde{\mathrm{Z}}) = \sum_{s=1}^{S} \sum_{i=1}^{|V|} ||\hat{\mathrm{Z}}_i - \widetilde{\mathrm{Z}}_i^{(s)}||_2^2, \tag{6}$$

where $\widetilde{\mathrm{Z}}$ denotes the predictions of the learner GNN on augmented samples, *i.e.*, $\widetilde{\mathrm{Z}} = f(\widetilde{\mathrm{X}}(\Theta); w)$ and $\hat{\mathrm{Z}}_i$ means the expected probability of $i$th node. Given the average of outputs $\overline{\mathrm{Z}} = \frac{1}{S}\sum_{s=1}^{S} \widetilde{\mathrm{Z}}^{(s)}$ on $S$ augmented samples, we apply the sharpening trick [48] to get the expected probability $\hat{\mathrm{Z}}$ by reducing the entropy of the predictions. It is written as

$$\hat{\mathrm{Z}}_{ic} = \overline{\mathrm{Z}}_{ic}^{\frac{1}{t}} / \sum_{j=1}^{\mathrm{C}} \overline{\mathrm{Z}}_{ij}^{\prime\frac{1}{t}}, \tag{7}$$

for all $i = 1 \ldots |V|$ and $c = 1 \ldots C$, where $|V|$ is the number of nodes, $C$ is the number of classes, and $0 < t \leq 1$ is a temperature to control the sharpness. In equation 7, $\hat{\mathrm{Z}}_{ic}$ denotes the expected probability of $c$th class for $i$th node, *i.e.*, $\hat{\mathrm{Z}}_i = [\hat{\mathrm{Z}}_{i1}, \ldots, \hat{\mathrm{Z}}_{iC}]$.

Since the consistency loss does not require labels, we apply the loss to all nodes including unlabeled data. The consistency loss, $\mathcal{L}_{\mathrm{con}}$, is commonly used to constrain model predictions to be invariant to input noise. In this work, we apply it to regularize the augmentor not the learner model. This regularizes meta-overfitting and excessive augmentation that drastically changes the predictions of GNNs on augmented data leveraging a large amount of unlabeled data. More discussion on the effectiveness of consistency regularization is in Section IV-D.

**Algorithm 2** AugGCR

**Input:** training data D, node feature matrix X, # epoch $N$, # folds for cross validation $C$, # augmentations $S$, learning rate $\eta_g$, $\eta_f$, DropNode ratio $\delta$
**Output:** network parameter $w$

**for** $n = 1$ to $N$ **do**
  **for** $c = 1$ to $C$ **do**
    $D^{m\_tr}, D^{m\_te} \leftarrow$ CVSplit(D, $c$)
    $\widetilde{\mathrm{X}}(\Theta^n) \leftarrow \mathrm{Aug}(\mathrm{X}; \Theta^n)$
    $\hat{w}^n(\Theta^n) \leftarrow w^n - \eta_f \nabla_w \mathcal{L}_{cls}(\mathrm{Y}^{m\_tr}, \widetilde{\mathrm{X}}(\Theta^n); w^n)$
    $g_c \leftarrow \nabla_\Theta \mathcal{L}_{\mathrm{xe}}(\mathrm{Y}^{m\_te}, \widetilde{\mathrm{X}}(\Theta^n); \hat{w}^n(\Theta^n))$
  **end for**
  **for** $s = 1$ to $S$ **do**
    $\widetilde{\mathrm{X}}^{(s)}(\Theta^n) \leftarrow \mathrm{Aug}(\mathrm{X}; \Theta^n)$
    $\widetilde{\mathrm{Z}}^{(s)}(\Theta^n) \leftarrow f(\widetilde{\mathrm{X}}^{(s)}(\Theta^n); w^n)$
  **end for**
  $g_r \leftarrow \nabla_\Theta \mathcal{L}_{con}(\widetilde{\mathrm{Z}}^{(s)}(\Theta^n))$
  $\Theta^{n+1} \leftarrow \Theta^n - \eta_g(\lambda \sum_{c=1}^{C} g_c + g_r)$
  $\widetilde{\mathrm{X}}(\Theta^{n+1}) \leftarrow \mathrm{Aug}(\mathrm{X}; \Theta^{n+1})$
  $g_w \leftarrow \nabla_w \mathcal{L}_{cls}(\mathrm{Y}^{m\_tr+m\_te}, \widetilde{\mathrm{X}}(\Theta^{n+1}); w^n)$
  $w^{n+1} \leftarrow w^n - \eta_f g_w$
**end for**

#### 3) OPTIMIZATION

Our estimation algorithm is outlined in Algorithm 2. To solve bi-level optimization in (3), we alternately optimize our augmentor and GNN. The lower-level optimization task is to train the GNN parameterized by $w$ minimizing the supervised loss $\mathcal{L}_{cls}$ on augmented train samples $\widetilde{\mathrm{X}}(\Theta^n)$. We approximate the solution to the lower-level optimization with the updated

**TABLE 1. Statistics and details of datasets for node classification. The number of Train, Validation, and Test denote the number of nodes in train set, validation set, and test set, respectively.**

| Dataset | # Nodes | # Edges | # Features | # Classes | # Train | # Validation | # Test |
|---|---|---|---|---|---|---|---|
| IMDB | 12772 | 37288 | 1256 | 3 | 300 | 300 | 2339 |
| DBLP | 18405 | 67946 | 334 | 4 | 800 | 400 | 2857 |
| AIRPORT | 1190 | 13599 | 238 | 4 | 119 | 238 | 833 |
| SQUIRREL | 5201 | 217073 | 2089 | 5 | 2496 | 1664 | 1041 |
| CHAMELEON | 2277 | 36101 | 2325 | 5 | 1092 | 729 | 456 |

parameters $\hat{w}$ using a gradient descent step as

$$w^*(\Theta) \approx \hat{w}^n(\Theta^n) = w^n - \eta_f \nabla_w \mathcal{L}_{cls}(\mathrm{Y}^{m\_tr}, \widetilde{\mathrm{X}}(\Theta^n); w^n), \quad (8)$$

where $\eta_f$ is a learning rate for $w$. $w^*(\Theta)$ is a computational graph for updating $\Theta$ and it is not explicitly evaluated. We do *not* update the GNN parameter $w$ at this point. Using this first-order approximation, the upper-level optimization minimizes $\mathcal{L}_{\text{aug}}(\cdot)$ on entire data. The equation for updating the augmentation model parameters $\Theta$ is given as

$$\Theta^{n+1} = \Theta^n - \eta_g \nabla_\Theta \mathcal{L}_{aug}(\mathrm{Y}^{m\_te}, \widetilde{\mathrm{X}}(\Theta^n); \hat{w}^n(\Theta^n)). \quad (9)$$

We adopt the cross-validation to alleviate meta-overfitting and average $C$ gradients from $C$ different splits as Line 2-7 in Algorithm 2. The augmentor parameter $\Theta$ is optimized with supervised loss $\mathcal{L}_{cls}$ on meta-test data and consistency loss $\mathcal{L}_{con}$ on total data including unlabeled data.

After updating the augmentation model parameters, we now update the GNN parameters $w$ using $\widetilde{\mathrm{X}}(\Theta^{n+1})$:

$$w^{n+1} = w^n - \eta_f \nabla_w \mathcal{L}_{cls}(\mathrm{Y}^{m\_tr+m\_te}, \widetilde{\mathrm{X}}(\Theta^{n+1}); w^n). \quad (10)$$

The classifier parameter $w$ is optimized with supervised loss $\mathcal{L}_{cls}$ on meta-train and meta-test data that belong to a training set in a standard supervised learning. Note that we study our augmentor in the context of Test-time Augmentation (TTA) [49]–[52] so the augmentor is be optimized by minimizing the loss on augmented data $\widetilde{X}$ instead of the original data $X$.

## IV. EXPERIMENT

We evaluate the effectiveness of our framework on five popular benchmark datasets. The results include the following. (1) Our framework improves the performance of a various learner GNNs, e.g., GCN, SGC, GraphSAGE compared to other augmentation methods: AdaEdge, DropEdge, DropNode and AddNoise. (2) It also shows competitive performance compared to other state-of-the-art semi-supervised methods: GRAND and GraphMix. (3) Ablation study shows that topology-aware augmentation and keeping consistency of GNNs predictions on unlabeled data improve the performance of GNNs.

**Datasets.** We evaluate our method on five benchmark datasets for node classification in various domains: movie dataset IMDB, citation network DBLP preprocessed by [53], air-traffic dataset AIRPORT [54], Wikipedia page-page network SQUIRREL and CHAMELEON [55] preprocessed by [56]. The statistics of datasets are summarized in Table 1.

**TABLE 2. Hyperparameters of AugGCR.**

| Hyperparameters | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|
| # Layers | 2 | 2 | 3 | 3 | 3 |
| Hidden layer size | 64 | 64 | 64 | 64 | 64 |
| Learning rate (classifier) $\eta_f$ | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 |
| L2 weight decay coefficient | 0 | 0 | 0 | 0 | 0 |
| Dropout rate | 0 | 0 | 0 | 0 | 0 |
| # Epochs | 200 | 200 | 500 | 100 | 100 |
| Learning rate (augmentor) $\eta_g$ | 0.001 | 0.001 | 0.002 | 0.0001 | 0.001 |
| # Data augmentations $S$ | 3 | 3 | 3 | 3 | 3 |
| Cross-entropy loss coefficient $\lambda$ | 0.01 | 0.0001 | 0.1 | 0.0005 | 1 |
| Drop ratio $\delta$ for consistency | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| Drop ratio $\delta$ for cross-entropy | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Sharpening temperature $t$ | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| # Hops $K$ | 8 | 8 | 8 | 8 | 8 |

**Implementation Details.** All experiments are implemented in Pytorch [57] with the geometric deep learning library TorchGeometric [58], and we use marco-F1 score as an evaluation metric. We set the dimensionality of hidden representations to 64 and the number of layers to 2 for IMDB, DBLP (and 3 for AIRPORT, SQUIRREL, and CHAMELEON) for all the methods. Models are optimized by the Adam [59] optimizer for 200 epochs except for AIRPORT in which models are trained for 500 epochs. We did not use the batch normalization, dropout, and weight decay. More Details of the hyperparameters setting are in Table 2. Hyperparameter tuning for baseline methods is performed by a grid search. The drop ratio $\delta$ for baselines DropEdge and DropNode are tuned by a grid search in $\{0.1, 0.2, \cdots, 0.9\}$. For the subroutine DropNode in our AugGCR, we did not perform a hyperparameter-tuning for the drop ratio. $\delta = 0.5$ is used for AugGCR in all the experiments. The test accuracy of models with the best performance on the validation sets is reported. All experiments are repeated 10 times and average accuracy and standard deviation are reported. The standard deviation of Gaussian noise in AddNoise is searched in $\{0.001, 0.01, 0.1, 1\}$. AdaEdge has two hyperparameters the number of adding/removing edges and they are searched in $\{0, 0.2|E|, 0.4|E|, 0.8|E|\}$.

**Baselines.** We evaluate the effectiveness of the proposed method by integrating with three widely-used graph neural networks: GCN [4], SGC [33], and GraphSAGE [2]. We compare our method with four augmentation

**TABLE 3.** Node classification performance (*F*1-score) of GNNs trained with supervised loss by various augmentation strategies.

| BaseGNNs | Augmentation | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|---|
| GCN | Vanilla | 58.18±0.54 | 85.32±0.41 | 55.68±1.67 | 25.94±1.22 | 36.16±2.43 |
| | +AdaEdge | 60.66±1.01 | 85.35±0.52 | 56.90±0.74 | 25.56±1.12 ↓ | 39.25±1.49 |
| | +DropEdge | 58.37±0.93 | 84.71±0.83 ↓ | 59.47±0.91 | 26.44±0.71 | 42.58±2.58 |
| | +DropNode | 58.92±1.06 | 85.44±0.62 | 59.78±1.04 | 26.23±0.86 | 34.71±1.39 ↓ |
| | +AddNoise | 59.19±1.10 | 85.50±0.28 | 58.06±0.67 | 25.58±1.06 ↓ | 39.89±1.59 |
| | +AugGCR | **63.72±1.21**† | **92.97±0.55**† | **61.21±1.07**† | **27.68±1.63**† | **44.62±1.86**† |
| GraphSAGE | Vanilla | 56.40±1.06 | 93.08±0.41 | 51.89±1.59 | 36.62±0.46 | 48.13±1.39 |
| | +AdaEdge | 60.03±0.95 | **93.89±0.16** | 56.33±0.83 | 29.08±0.90 ↓ | 46.90±1.27 ↓ |
| | +DropEdge | 56.50±1.60 | 92.95±0.43 ↓ | 51.58±1.19 ↓ | 36.23±0.60 ↓ | 49.66±1.46 |
| | +DropNode | 56.85±0.98 | 92.50±0.31 ↓ | 52.08±1.01 | 35.46±0.87 ↓ | 48.04±1.51 ↓ |
| | +AddNoise | 57.04±1.00 | 92.05±1.01 ↓ | **56.87±0.80** | 36.18±0.76 ↓ | 48.77±0.98 |
| | +AugGCR | **62.41±1.10**† | 93.12±0.47 | 55.32±2.69 | **37.71±0.61**† | **49.68±0.81** |
| SGC | Vanilla | 59.43±1.96 | 93.51±0.31 | 57.11±1.29 | 26.58±1.21 | 36.02±2.89 |
| | +AdaEdge | 60.20±1.04 | 91.05±0.30 ↓ | 58.26±0.33 | 26.35±1.38 ↓ | 43.57±2.17 |
| | +DropEdge | 59.20±1.33 ↓ | 93.10±0.56 ↓ | 57.94±1.25 | 27.10±1.17 | 39.93±1.77 |
| | +DropNode | 58.55±2.19 ↓ | **93.94±0.33** | 58.48±0.50 | 27.09±1.11 | 37.83±3.02 |
| | +AddNoise | 60.01±0.99 | 92.12±0.69 ↓ | 58.40±1.04 | 25.84±0.93 ↓ | 36.74±3.09 |
| | +AugGCR | **62.57±0.63**† | 93.26±0.63 ↓ | **59.55±1.28**† | **28.47±1.58**† | **44.49±1.61** |

↓ indicates the performance degradation by augmentation compared to Vanilla. † denotes that the improvement compared to the second-best algorithm (underlined) is statistically significant ( $p < 0.05$ ) by $t$-test.

**TABLE 4.** Node classification performance (*F*1-score) of GNNs trained with semi-supervised loss by various learning strategies.

| Learning Method | BaseGNNs | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|---|
| GCN+CR | GCN | 59.46±1.77 | 85.24±0.86 | 54.47±1.6 | 25.81±0.99 | 37.40±1.78 |
| GraphMix | GCN | 57.17±1.02 | 83.40±0.82 | 56.28±1.11 | 25.96±0.68 | 35.67±1.42 |
| GRAND | GRAND | 61.27±1.60 | 91.76±0.26 | 56.42±3.53 | 24.16±1.69 | 44.22±1.80 |
| AugGCR | GCN | **63.72±1.21** | **92.97±0.55** | **61.21±1.07** | **27.68±1.63** | **44.62±1.86** |

strategies: AdaEdge [27] that adaptively adjusts the graph topology by removing/adding edges based on its predictions; DropEdge [18] that randomly removes a fixed number of edges from the graph; DropNode [24] that randomly drops the entire feature vector of a node with probability $\delta$ and scales up features by a factor of $\frac{1}{(1-\delta)}$, *i.e.*, $\widetilde{\mathrm{X}}'_i = \frac{\epsilon_i}{1-\delta}\mathrm{X}_i$, where $\epsilon_i \sim \text{Bernoulli}(1-\delta)$; AddNoise that adds random Gaussian noise $\epsilon \sim N(0, \sigma)$ to node features for augmentation.

### A. RESULTS ON NODE CLASSIFICATION

Table 3 shows that our method significantly improves node classification performance of most base models (denoted as Vanilla). AugGCR consistently enhances the F1-score for almost all the GNNs compared to other augmentation strategies. More specifically, we observe that AugGCR improves the performance by 6.0% compared to the vanilla GraphSAGE model on IMDB. In addition, AugGCR provides an 8.5% absolute performance improvement on both GCN and SGC compared to Vanilla on CHAMELEON. Due to the saturated performance of GraphSAGE and SGC on DBLP, room for improvement is limited compared to Vanilla. In the GCN case, our method yields a high performance gain of 7.7% compared to the Vanilla. In *heterophilic* graph datasets such as SQUIRREL and CHAMELEON, the performance of GNNs considering the *homophily* assumption where neighbor nodes have similar labels, is generally inferior to MLP that does not utilize the graph structure. Nevertheless, our method successfully learns an augmentor that adjusts the input features leveraging the graph topology improves the performance of all Vanilla models in heterophilic graph datasets. In summary, AugGCR shows 5.8%, 2.6%, and 3.1% improvement on average on GCN, GraphSAGE, and SGC over all datasets. It is worth noting that no significant gain was observed in the baseline augmentations even if we employed TTA setting on those methods for fair comparison.

**TABLE 5.** Training time (s) per epoch of Vanilla and AugGCR with GCN.

| Method | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|
| Vanilla (GCN) | 0.0089 | 0.0087 | 0.0097 | 0.0120 | 0.0105 |
| AugGCR | 0.3377 | 0.2159 | 0.1302 | 0.4231 | 0.2096 |

We also evaluate the effectiveness of AugGCR on widely used citation network datasets; Citeseer, Cora, and Pubmed. We observe that AugGCR consistently enhances the performance (Accuracy) compared to the Vanilla model. More specifically, we observe that AugGCR improves the F1-score by 2.1% on Citeseer dataset, 1.3% on Cora dataset, and 1.4% on Pubmed dataset compared to the Vanilla model.

In addition, we examine the training time of AugGCR. Since the meta-learning optimization usually takes quite a long time, AugGCR needs more training time compared to Vanilla (GCN). See Table 5 for details.

### B. COMPARISON WITH SEMI-SUPERVISED METHODS

We verify the efficacy of our AugGCR on semi-supervised node classification by comparing it with a strong baseline and two state-of-the-art methods. GCN+CR minimizes the cross-entropy on labeled data as well as the consistency loss on unlabeled data augmented by DropNode; GraphMix [37] is a regularization method based on semi-supervised learning by linear interpolation between two data on graphs; GRAND [24] minimizes both classification and consistency losses with pre-defined augmentation for representation learning. Our AugGCR and GraphMix use GCNs as a base model. GRAND proposes both the architecture and semi-supervised learning method. As mentioned in [24], GCN equipped with GRAND underperforms their own architecture, we did not include it in our experiments. Table 4 presents the semi-supervised node classification performance. We observe that our method outperforms all baselines on all the 5 datasets. In particular, on AIRPORT dataset the proposed method improves the F1-score by 5.5% on average compared to all baselines. Overall, our method provides 6.2%, 6.3%, and 2.5% improvements on average compared to the baselines: GCN+CR, GraphMix, and GRAND.

### C. ABLATION STUDY

In this paper, we present AugGCR that includes a new architecture for an augmentor and meta-learning formulation to optimize the augmentor. We provide an ablation study of each component in AugGCR in Table 6. We also examine the effectiveness of our architecture and formulation comparing with a standard graph neural network and its supervised loss, *e.g.*, GCN and the cross-entropy, which are described in Table 7. Lastly, in Table 8, we observe the efficacy of the learnable coefficients, $\alpha_k$ and $\beta$, in Algorithm 1.

#### 1) ABLATION STUDY OF EACH COMPONENT

Table 6 summarizes the results of an ablation study to examine the contribution of each component in our framework. Experiments were conducted with GCN for all datasets with respect to three components: **A**ugmentor, graph **T**opology, and **C**onsistency. 'Vanilla' denotes a standard training of GCN without augmentation. '$+\mathcal{A}$' trains the GNN with our augmentor but neither consistency loss, nor graph topology is used. To be specific, the loss for augmentation has no consistency regularization term, *i.e.*, $\mathcal{L}_{\text{aug}} := \mathcal{L}_{\text{xe}}$ in (3) and the learnable mixed-order propagation ($\sum_{k=0}^{K} \alpha_k \hat{\text{A}}^k \text{X}$) is removed from Algorithm 1. '$+\mathcal{A}\&\mathcal{T}$' trains the GNN with our augmentor and topology information leveraged with the learnable mixed-order propagation. Similarly, '$+\mathcal{A}\&\mathcal{C}$' trains a GNN learning the augmentor with consistency regularization $\mathcal{L}_{\text{aug}} := \mathcal{L}_{\text{xe}} + \mathcal{L}_{\text{con}}$ as in (5) but graph topology is not used. Our experiments show that on average using all the components achieves 2.1% and 4.0% improvement compared to $+\mathcal{A}\&\mathcal{C}$ and $+\mathcal{A}\&\mathcal{T}$, respectively. Besides, AugGCR improves the performance on average by 4.9% compared to $+\mathcal{A}$. In IMDB and CHAMELEON, the gain from topology information in AugGCR is not significant. The gap between AugGCR and $+\mathcal{A}\&\mathcal{C}$ is statistically insignificant since the difference is much smaller than the variance of accuracy.

#### 2) ABLATION STUDY OF THE ARCHITECTURE AND THE FORMULATION

Table 7 compares our AugGCR with a vanilla GCN without augmentation and two variants of AugGCR. The results of the vanilla GCN without augmentation is shown in the first row. The second row of Table 7 shows the performance of AugGCR without our formulation. In other words, the stack of AugGCR's augmetor and the learner GNN (GCN) is trained by the standard supervised training with the cross-entropy loss. This construction improves the performance in IMDB, DBLP and CHAM datasets whereas in SQUI. degradation is observed. The results in the third row of Table 7 are obtained from the AugGCR formulation but the architecture of the augmentor is not ours. A standard GCN is used to learn augmentation. In this setting two different GCNs are used; one for a learner GNN and the other for the augmentor. This proves the efficacy of our formulation (or learning method) that our formulation can learn effective augmentation by a graph neural network. The augmentation with our architecture and formulation shown in the last row consistently provides the largest improvement in all five datasets.

#### 3) ABLATION STUDY OF LEARNABLE COEFFICIENTS; $\alpha_k$ AND $\beta$

We conduct an ablation study of learnable coefficients $\alpha_k$ and $\beta$ in Algorithm 1. In Table 8, "$-\alpha_k$" denotes adopting the constant $1/(K+1)$ instead of the learnable parameter $\alpha_k$. "$-\beta$" means generating the augmented node features only by $\hat{X}'$ in Algorithm 1. "$-\alpha_k, \beta$" means adopting both

**TABLE 6.** Ablation study of AugGCR (+$\mathcal{A}$&$\mathcal{T}$&$\mathcal{C}$). $\mathcal{A}$: augmentor, $\mathcal{T}$: graph topology, $\mathcal{C}$: consistency regularization.

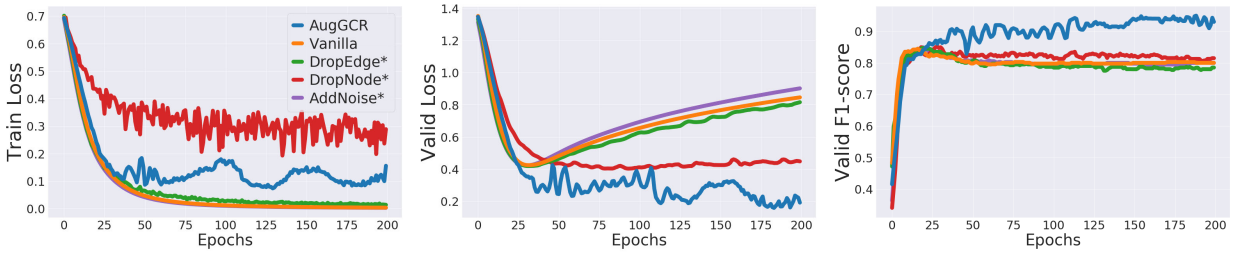
| | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|
| Vanilla (GCN) | 58.18±0.54 | 85.32±0.41 | 55.68±1.67 | 25.94±1.22 | 36.16±2.43 |
| +$\mathcal{A}$ | 59.25±1.71 | 82.98±1.01 | 55.14±1.81 | 27.36±0.90 | 41.05±1.61 |
| +$\mathcal{A}$&$\mathcal{T}$ | 59.69±1.75 | 83.84±0.88 | 58.18±2.05 | 27.02±1.77 | 41.25±2.01 |
| +$\mathcal{A}$&$\mathcal{C}$ | **63.76±1.27** | 86.59±0.60 | 57.53±2.42 | 27.05±1.63 | **44.75±3.14** |
| AugGCR | 63.72±1.21 | **92.97±0.55** | **61.21±1.07** | **27.68±1.63** | 44.62±1.86 |

**TABLE 7.** Ablation study of AugGCR's architecture and formulation.

| Learning method | Augmentor architecture | BaseGNNs | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|---|---|
| Standard | - | GCN | 58.18±0.54 | 85.32±0.41 | 55.68±1.67 | 25.94±1.22 | 36.16±2.43 |
| Standard | AugGCR | GCN | 60.92±1.76 | 92.43±0.39 | 55.99±0.64 | 23.61±1.37 | 37.47±2.09 |
| AugGCR | GCN | GCN | 61.27±1.96 | 91.25±0.43 | 60.33±0.55 | 25.82±1.47 | 39.77 ±1.70 |
| AugGCR | AugGCR | GCN | **63.72±1.21** | **92.97±0.55** | **61.21±1.07** | **27.68±1.63** | **44.62±1.86** |

**TABLE 8.** Ablation study of AugGCR on learnable coefficient $\alpha_k$ and $\beta$ in Algorithm 1.

| | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|
| w/o $\alpha_k, \beta$ | 33.85±9.96 | 82.12±8.83 | 59.79±1.07 | 26.76±2.02 | 34.76±3.31 |
| w/o $\beta$ | 34.83±11.1 | 80.28±7.33 | 60.18±2.11 | 26.81±2.33 | 33.94±3.23 |
| w/o $\alpha_k$ | 62.87±1.43 | 90.88±1.14 | 61.04±1.19 | 26.86±1.72 | 44.00±1.10 |
| AugGCR | **63.72±1.21** | **92.97±0.55** | **61.21±1.07** | **27.68±1.63** | **44.62±1.86** |




(a) Training loss curve (b) Validation loss curve (c) F1-score on validation set

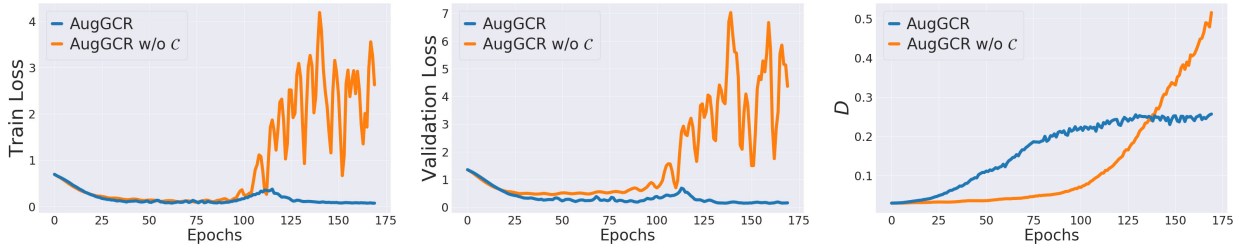
**FIGURE 2.** Learning curve of GCN trained with various augmentation methods including AugGCR (blue) on DBLP dataset. It shows the cross entropy losses of train set (left), validation set (mid) and the F1-score on validation set (right). Most of the other augmentation methods suffer from over-fitting while AugGCR shows considerable generalization ability.

"$-\alpha_k$" and "$-\beta$". It shows that AugGCR has 10.83% improvement on average compared to AugGCR without $\beta$, which implies that controlling the strength of augmentation by adjusting the influence of original node features on augmented node features is essential for proper augmentation. Especially, it provides 28.89% performance gain on IMDB dataset. In addition, $\alpha_k$ consistently provides improvements in all five datasets. Thus, we can observe that learning coefficients $\alpha_k$ and $\beta$ are important to AugGCR, especially $\beta$.

### D. ANALYSIS

#### 1) PREVENTING OVERFITTING

In Figure 2 from the left, the graphs are placed in order of training loss, validation loss, and F1-score for validation set. The learning curves of various augmentation methods in the configuration with the best-performing (*) with GCN on DBLP (*e.g.,* DropEdge*, drop ratio $\delta$ is set at 0.6.). It shows that the training loss curve decreases rapidly along the training epochs, except for AugGCR and DropNode. As we can expect from the training loss, the validation loss curves for Vanilla, DropEdge, and AddNoise increase, which implies that the model overfits to training data. It is interesting that the F1-score gaps between DropNode and other baselines suffering from overfitting are negligible despite their large validation loss differences. In contrast, the training loss curve of AugGCR decreases less compared to other steeper decreasing curves and its validation loss curve constantly drops by feeding in adjusted input features through the augmentor. This demonstrates the effect of AugGCR on preventing overfitting.




(a) Training loss (b) Validation loss (c) $\sum_i |\mathrm{X}_i - \widetilde{\mathrm{X}}_i|$

**FIGURE 3.** Learning curves of AugGCR on DBLP dataset. It shows a cross-entropy loss of (a) train data, (b) validation data, and (c) distance between the original features $\mathbf{X_i}$ and the augmented features $\widetilde{\mathbf{X}}_\mathbf{i}$, *i.e.*, $\sum_i |\mathbf{X_i} - \widetilde{\mathbf{X}}_\mathbf{i}|$. AugGCR (blue) shows stable learning curves whereas AugGCR without consistency loss (orange) shows slower convergence and even diverges. Also in (c) the large distance between original samples and augmented samples indicate that AugGCR without consistency loss (orange) generates excessive augmentations caused by meta-overfitting. This shows the effectiveness of consistency regularization for learning augmentation.

**TABLE 9.** Node classification performance ($F1$-score) of AugGCR using supervised/semi-supervised setting for training the Classifier and Augmentor.

| Classifier | Augmentor | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|---|
| supervised | semi-supervised | 63.72±1.21 | 92.97±0.55 | 61.21±1.07 | 27.68±1.63 | 44.62±1.86 |
| semi-supervised | semi-supervised | 64.99±1.13 | 93.26±0.38 | 61.42±0.87 | 28.22±1.66 | 45.48±0.98 |

**TABLE 10.** Node classification performance ($F1$-score) of AugGCR and other regularization techniques; dropout and weight decay.

| Method | IMDB | DBLP | AIR. | SQUI. | CHAM. |
|---|---|---|---|---|---|
| Vanilla | 57.91±0.86 | 84.48±1.63 | 54.44±1.07 | 25.13±1.37 | 36.52±2.05 |
| Decay | 57.91±0.86 | 84.98±0.34 | 55.10±0.90 | 25.82±1.10 | 33.01±2.29 |
| Dropout | 59.63±0.58 | 85.17±0.46 | 57.27±1.46 | 26.23±0.93 | 36.50±1.77 |
| AugGCR | **63.72±1.21** | **92.97±0.55** | **61.21±1.07** | **27.68±1.63** | **44.62±1.86** |

As shown in the rightmost in Figure 2, we show that the F1-score of AugGCR outperforms significantly other augmentation baselines.

### 2) CONSISTENCY REGULARIZATION

We analyze the efficacy of consistency regularization, which is a component of our proposed framework. In FIGURE 3, we illustrate the training and validation loss curves of AugGCR and AugGCR without consistency regularization. It is conducted on DBLP dataset with GCN as a learner model. The rightmost is the curve for the distance between original and augmented node features of AugGCR and AugGCR w/o $\mathcal{C}$. The distance is measured by MAE (Mean Absolute Error), *i.e.*, $\sum_i |\mathrm{X}_i - \widetilde{\mathrm{X}}_i|$. As shown in FIGURE 3(c), consistency loss prevents the excessive feature augmentation. It is noteworthy that AugGCR utilizes the consistency regularization only for training the augmentor and not the classifier. Our preliminary experiment showed that there was no significant performance gain in adopting consistency regularization for the classifier. See Table 9 for more details.

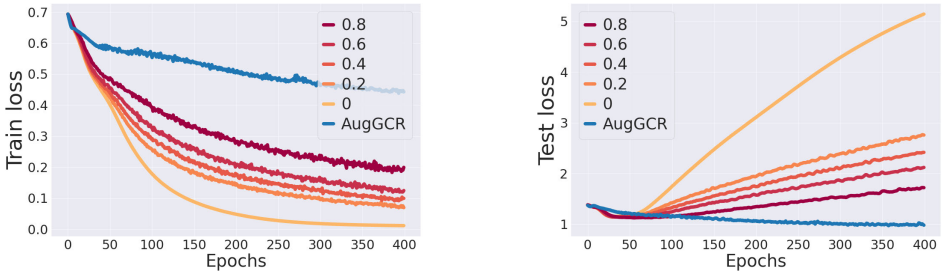
### 3) COMPARISON WITH OTHER REGULARIZATION TECHNIQUES

Dropout and weight decay are also standard regularization techniques to deal with weak-generalization problem of deep neural networks. To study the difference between AugGCR and other regularization techniques, we plot the learning curves of GCN with AugGCR, dropout and weight decay. FIGURE 4 shows the train and test loss of GCN on AIRPORT dataset with various configurations. We can observe that AugGCR has considerable generalization ability while the other regularization techniques still suffer from overfitting. We also observe the node classification performance ($F1$-score) of dropout and weight decay. It is performed by grid search, dropout ratio in $\{0, 0.2, 0.4, 0.6, 0.8\}$ and weight decay in $\{0, 0.0001, 0.0005, 0.001\}$, but the performance gain is small compared to the gain of AugGCR. See Table 10 for details.

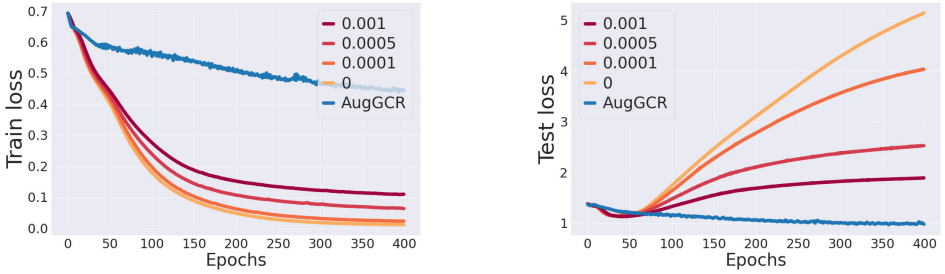
### 4) MIXUP COEFFICIENT $\beta$

Our proposed framework can adaptively augment the node feature matrix by the mixup coefficient $\beta$, which is a learnable parameter to average the perturbed feature matrix $\hat{\mathrm{X}}$ and the original node feature matrix X, *i.e.*, $\widetilde{\mathrm{X}} = \sigma(\beta)\mathrm{X} + (1 - \sigma(\beta))\hat{\mathrm{X}}$. It means that $\beta$ adjusts the influence of the original node features in augmented features. To analyze the change of $\beta$, we plot the cross-entropy loss of AugGCR and Vanilla on training and validation data with GCN as a base model in DBLP.

We observe that $\beta$ dynamically adjusts as the model is optimized. In FIGURE 5, after several updates, we show that




(a) Learning curves of GCN with AugGCR (blue) vs. different dropout ratio




(b) Learning curves of AugGCR (blue) vs. different weight decay.

**FIGURE 4. Comparison of learning curves between AugGCR (blue) and other regularization techniques, dropout (first row) and weight decay (second row). It shows the cross-entropy losses of train set (first column) and the test set (second column).**

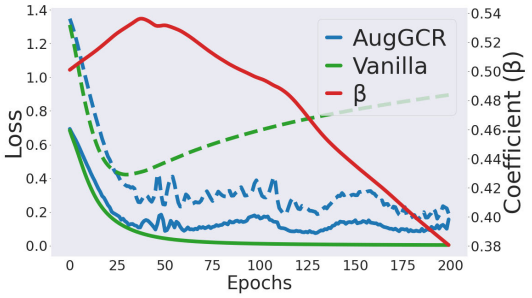



**FIGURE 5. Learning curve of GCN, AugGCR with GCN as a learner model, and $\beta$. Training and validation loss are illustrated in solid and dash lines respectively. AugGCR (blue) adaptively augment node feature matrix by learnable mixup coefficient $\beta$ (red). At the beginning of increasing the validation loss of Vanilla (green), AugGCR downgrades $\beta$ to alleviate the overfitting by focusing on the perturbed feature $\hat{\mathbf{X}}$.**

$\beta$ decreases to attend to the perturbed feature more than the original feature. It can be interpreted that leveraging node feature matrix perturbed by learnable mixed-order propagation helps to keep model from overfitting to the input feature.

## V. CONCLUSION

We propose AugGCR, a novel framework to learn how to augment input features of graph-structured data to enhance the generalization ability of the learner model. Overall, the procedure alternately optimizes the augmentor and the learner model, which can be formulated as meta-learning. Also, we adopt a consistency regularizer to make the augmentor generate data that the learner can keep consistency on. AugGCR is applicable to any GNNs and yields up to 8.46% improvement in F1-score over the vanilla model. We analyze the behavior of AugGCR on generalization ability. In future work, we are planning to adopt self-supervised learning to enhance the generalization ability of our augmentor.

## ACKNOWLEDGMENT

*(Hyeonjin Park and Seunghun Lee equally contributed to this work.)*

## REFERENCES

[1] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bulletin.*, vol. 40, no. 3, pp. 52–74, Sep. 2017.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1025–1035.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018, pp. 1–12.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017.

[5] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. ESWC*, 2018, pp. 593–607.

[6] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. NIPS*, vol. 31, 2018, pp. 5165–5175.

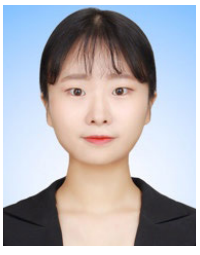
[7] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop*, 2016.

[8] Z. Li, Z. Liu, J. Huang, G. Tang, Y. Duan, Z. Zhang, and Y. Yang, "MV-GCN: Multi-view graph convolutional networks for link prediction," *IEEE Access*, vol. 7, pp. 176317–176328, 2019.

[9] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. NIPS*, vol. 31, 2018, pp. 4805–4815.

[10] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. NIPS*, vol. 28, 2015, pp. 2224–2232.

[11] Y. Liang, Y. Zhang, D. Gao, and Q. Xu, "An end-to-end multiplex graph neural network for graph representation learning," *IEEE Access*, vol. 9, pp. 58861–58869, 2021.
[12] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1225–1234.
[13] T. Zhong, T. Wang, J. Wang, J. Wu, and F. Zhou, "Multiple-aspect attentional graph neural networks for online social network user localization," *IEEE Access*, vol. 8, pp. 95223–95234, 2020.
[14] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *Proc. NIPS*, vol. 29, 2016, pp. 4502–4510.
[15] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "MoleculeNet: A benchmark for molecular machine learning," *Chem. Sci.*, vol. 9, no. 2, pp. 513–530, 2018.
[16] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 950–958.
[17] T. Wang, D. Shi, Z. Wang, S. Xu, and H. Xu, "MRP2Rec: Exploring multiple-step relation path semantics for knowledge graph-based recommendations," *IEEE Access*, vol. 8, pp. 134817–134825, 2020.
[18] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *Proc. ICLR*, 2020.
[19] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.
[20] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, "Population based augmentation: Efficient learning of augmentation policy schedules," in *Proc. ICML*, 2019, pp. 2731–2741.
[21] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," in *Proc. NIPS*, vol. 32, 2019, pp. 6665–6675.
[22] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Faster autoaugment: Learning augmentation strategies using backpropagation," in *Proc. ECCV*, vol. 12370. Springer, 2020, pp. 1–16.
[23] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*. [Online]. Available: https://arxiv.org/abs/1308.3432
[24] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Proc. NIPS*, vol. 33, 2020, pp. 22092–22103.
[25] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. NIPS*, vol. 33, 2020, pp. 5812–5823.
[26] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, "Robust graph representation learning via neural sparsification," in *Proc. ICML*, 2020, pp. 11458–11468.
[27] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3438–3445.
[28] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," in *Proc. NIPS*, vol. 33, 2020, pp. 6256–6268.
[29] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1195–1204.
[30] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1163–1171.
[31] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *Proc. ICLR*, 2017.
[32] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," in *Proc. NIPS*, vol. 27, 2014, pp. 3365–3373.
[33] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. ICML*, 2019, pp. 6861–6871.
[34] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. ICLR*, 2013.
[35] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, vol. 29, 2016, pp. 3837–3845.
[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, 2019.
[37] V. Verma, M. Qu, A. Lamb, Y. Bengio, J. Kannala, and J. Tang, "Graphmix: Regularized training of graph neural networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 10024–10032.
[38] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020, *arXiv:2004.05439*. [Online]. Available: https://arxiv.org/abs/2004.05439
[39] H. B. Lee, T. Nam, E. Yang, and S. J. Hwang, "Meta dropout: Learning to perturb latent features for generalization," in *Proc. ICLR*, 2019.
[40] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang, "Cross-domain few-shot classification via learned feature-wise transformation," 2020, *arXiv:2001.08735*. [Online]. Available: https://arxiv.org/abs/2001.08735
[41] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, "DADA: Differentiable automatic data augmentation," 2020, *arXiv:2003.03780*. [Online]. Available: https://arxiv.org/abs/2003.03780
[42] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-GNN: On few-shot node classification in graph meta-learning," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2357–2360.
[43] A. J. Bose, A. Jain, P. Molino, and W. L. Hamilton, "Meta-graph: Few shot link prediction via meta learning," 2019, *arXiv:1912.09867*. [Online]. Available: https://arxiv.org/abs/1912.09867
[44] M. Chen, W. Zhang, W. Zhang, Q. Chen, and H. Chen, "Meta relational learning for few-shot link prediction in knowledge graphs," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 4216–4225.
[45] L. Liu, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Learning to propagate for graph meta-learning," in *Proc. NIPS*, vol. 32, 2019, pp. 1037–1048.
[46] K. Huang and M. Zitnik, "Graph meta learning via local subgraphs," in *Proc. NIPS*, vol. 33, 2020, pp. 5862–5874.
[47] D. Hwang, J. Park, S. Kwon, K.-M. Kim, J.-W. Ha, and H. J. Kim, "Self-supervised auxiliary learning with meta-paths for heterogeneous graphs," in *Proc. NIPS*, vol. 33, 2020, pp. 10294–10305.
[48] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Proc. NIPS*, vol. 32, 2019.
[49] I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," in *Proc. NIPS*, vol. 33, 2020, pp. 4163–4174.
[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.
[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.
[53] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, May 2019, pp. 2022–2032.
[54] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 385–394.
[55] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *J. Complex Netw.*, vol. 9, no. 2, pp. 1–20, May 2021.
[56] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," in *Proc. ICLR*, 2020.
[57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in Pytorch," in *Proc. NIPS Workshop*, 2017.
[58] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *Proc. ICLR*, 2019.
[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.



**HYEONJIN PARK** was born in Seoul, South Korea, in 1996. She received the B.S. degree in applied statistics and computer engineering from the University of Konkuk, Seoul, in 2020. She is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, Korea University, Seoul.

She is currently supervised by Prof. Hyunwoo J. Kim. Her research interests include geometric deep learning on graphs and their applications.

**SEUNGHUN LEE** received the B.S. degree from the Department of Computer Science and Engineering, Korea University, in 2020, where he is currently pursuing the M.S. degree.

His research interests include neural networks, especially in graph neural networks and semi-supervised learning.

**DASOL HWANG** received the B.S. degree in applied mathematics from Kyung Hee University, South Korea. She is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, Korea University.

She is currently supervised by Prof. Hyunwoo J. Kim. Her research interests include graph neural networks and machine learning and their applications.

**JISU JEONG** received the B.S., M.S., and Ph.D. degrees in mathematical sciences from KAIST. The main topic of his thesis was graph theory, including width parameters and fixed parameter tractable algorithms.

He was a Senior Engineer at Samsung SDS and a Research and Development Data Scientist at WATCHA. Since 2020, he has been working at Naver. His research interests include diversity and cold-start problem in recommender systems and graph neural networks.

**KYUNG-MIN KIM** received the Ph.D. degree in machine learning from Seoul National University, in 2018.

He was a Co-Founder at Surromind Robotics, South Korea. He is currently working with Naver CLOVA. His main job is realizing AI in various business domains. Specifically, he is developing recommender systems, targeted advertising systems, and demand forecasting engines. He believes that it is possible to bridge the gap between academic results and real-world problems, so building a next-generation machine learning model for business problem is his mission. His research interests include massive-scale deep learning, multimodal learning, and recommender systems.

**JUNG-WOO HA** received the B.S. and Ph.D. degrees from the Department of Computer Science, Seoul National University.

He is currently the Head of Naver AI Laboratory. He also works as the Co-Director of two AI research centers cooperating with academia, such as SNU-NAVER Hyperscale AI Center and KAIST-NAVER Hypercreative AI Center. His contributions in the field of many machine learning, computer vision, and natural language processing research communities as the program committees or reviewers, including ICLR, NeurIPS, and ICML. His main research interests include deep learning-based computer vision, natural language processing, recommendation, and industrial applications.

**HYUNWOO J. KIM** received the Ph.D. degree major in computer science and minor in statistics from the University of Wisconsin–Madison, in 2017.

He worked at Amazon Lab126, Sunnyvale, CA, USA. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Korea University. His research interests include geometric deep learning for graphs and manifolds, manifold statistics, machine learning, computer vision, and medical imaging.