$\mathcal{G}$-MIXUP: GRAPH DATA AUGMENTATION FOR GRAPH CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This work develops *mixup to graph data*. Mixup has shown superiority in improving the generalization and robustness of neural networks by interpolating features and labels of random two samples. Traditionally, Mixup can operate on regular, grid-like, and Euclidean data such as image or tabular data. However, it is challenging to directly adopt Mixup to augment graph data because two graphs typically: 1) have different numbers of nodes; 2) are not readily aligned; and 3) have unique topologies in non-Euclidean space. To this end, we propose $\mathcal{G}$-Mixup to augment graphs for graph classification by interpolating the generator (i.e., graphon) of different classes of graphs. Specifically, we first use graphs within the same class to estimate a graphon. Then, instead of directly manipulating graphs, we interpolate graphons of different classes in the Euclidean space to get mixed graphons, where the synthetic graphs are generated through sampling based on the mixed graphons. Extensive experiments show that $\mathcal{G}$-Mixup substantially improves the generalization and robustness of GNNs.

1 INTRODUCTION

Recently deep learning has been widely adopted to graph analysis. Graph Neural Networks (GNNs) (Wu et al., 2020; Zhou et al., 2020b; Zhang et al., 2020; Xu et al., 2018) have made many significant breakthroughs on graph classification. Meanwhile, data augmentation (e.g., DropEdge (Rong et al., 2020), Subgraph (You et al., 2020)) has also been adopted to graph analysis by generating synthetic graphs to create more training data for improving the generalization of graph classification models. However, existing graph data augmentation strategies typically aim to augment graphs at a *within-graph* level by either modifying edges or nodes in individual graph, which limits them to only generating new graphs based on one individual graph. The *between-graph* augmentation methods (i.e., augmenting graphs between graphs) are still under-explored.

In parallel with the development of graph neural networks, Mixup (Zhang et al., 2017) and its variants (e.g., Manifold Mixup (Verma et al., 2019)), as data augmentation methods, have been theoretically and empirically shown to improve the generalization and robustness of deep neural networks in image recognition (Zhang et al., 2017; Verma et al., 2019; Zhang et al., 2021) and natural language processing (Guo et al., 2019; Guo, 2020). The basic idea of Mixup is to linearly interpolate continuous values of random sample pairs to generate more synthetic training data. The formal mathematical expression of Mixup is $\mathbf{x}_{new} = \lambda\mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \mathbf{y}_{new} = \lambda\mathbf{y}_i + (1-\lambda)\mathbf{y}_j$, where $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ are two samples drawn at random from training data and the target $\mathbf{y}$ are one-hot labels. With *graph neural network* and *mixup* in mind, the following question naturally arises,

Can we mix up graph data to improve the generalization and robustness of GNNs?

It remains an open and challenging problem to mix up the graph data due to the characteristics of graphs and the requirements of applying Mixup. Typically, Mixup requires that original data instances are regular and well-aligned in Euclidean space, such as image data and table data. However, graph data is distinctly different from image data due to the following characteristics: *(i) graph data is irregular*. The number of nodes in different graphs are typically different to others; *(ii) graph data is not well-aligned*. The nodes in different graphs can not be aligned well directly; *(iii) graph topology between classes are divergent*. The topologies of a pair of graphs from different classes are usually different while the topologies of those from the same class are usually similar. thus make it challenging to directly adopt the Mixup strategy to graph data.

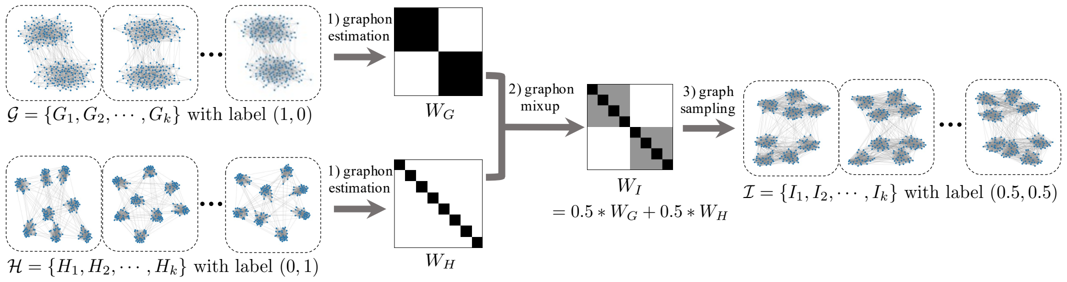
1) graphon estimation
W_G
1) graphon estimation
W_H
2) graphon mixup
W_I
$= 0.5 * W_G + 0.5 * W_H$
3) graph sampling
$\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ with label $(1, 0)$
$\mathcal{H} = \{H_1, H_2, \cdots, H_k\}$ with label $(0, 1)$
$\mathcal{I} = \{I_1, I_2, \cdots, I_k\}$ with label $(0.5, 0.5)$

Figure 1: An overview of $\mathcal{G}$-Mixup. The task is binary graph classification. We have two classes of graphs $\mathcal{G}$ and $\mathcal{H}$ with different topologies ($\mathcal{G}$ has two communities while $\mathcal{H}$ has eight communities). $\mathcal{G}$ and $\mathcal{H}$ have different graphons. We mix up the graphons $W_{\mathcal{G}}$ and $W_{\mathcal{H}}$ to obtain a mixed graphon $W_{\mathcal{I}}$, and then sample new graphs from the mixed graphon. Intuitively, the synthetic graphs have two major communities and each of which has four sub-communities, demonstrating that the generated graphs preserve the structure of original graphs from both classes.

To tackle the aforementioned problems, we propose $\mathcal{G}$-Mixup, a class-level graph data augmentation method, to mix up graph data based on graphons. The graphs within one class have the same generator (i.e., graphon). We mix up the graphons of different classes and then generate synthetic graphs. Informally, a graphon can be thought of as a probability matrix (e.g., the matrix W_G and W_H in Figure 1), where $W(i, j)$ represents the probability of edge between node i and j. The real-world graphs can be regraded as generated from graphons. In this way, we can mix two classes of graphs by mixing their generators. Since the graphons of different graphs is regular, well-aligned, and is defined in Euclidean space, it is easy and natural to mix up the graphons and then generate the synthetic graphs therefrom. We also provide theoretical analysis of graphons mixup, which guarantees that the generated graphs will preserve the key characteristics of both original classes. Our proposed method is illustrated in Figure 1 with an example. Given two graph training sets $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ and $\mathcal{H} = \{H_1, H_2, \cdots, H_k\}$ with different labels and distinct topologies (i.e., $\mathcal{G}$ has two communities while $\mathcal{H}$ has eight communities), we estimate graphons $W_{\mathcal{G}}$ and $W_{\mathcal{H}}$ respectively from $\mathcal{G}$ and $\mathcal{H}$. We then mix up the two graphons and obtain a mixed graphon $W_{\mathcal{I}}$. After that, we sample synthetic graphs from $W_{\mathcal{I}}$ as additional training graphs. The generated synthetic graphs have two major communities and each of them have four sub-communities, which is a mixture of the two graph sets. It thus shows that $\mathcal{G}$-Mixup is capable of mixing up graphs.

In summary, our **main contributions** are three-fold. *Firstly*, we propose $\mathcal{G}$-Mixup to augment the training graphs for graph classification. Since directly mixing up graphs is intractable, $\mathcal{G}$-Mixup mixes the graphons of different classes of graphs to generate synthetic graphs. *Secondly*, we theoretically prove that the synthetic graph will be the mixture of the original graphs, where the key topology (i.e., discriminative motif) of source graphs will be preserved. *Thirdly*, we demonstrate the efficacy of the proposed $\mathcal{G}$-Mixup on various graph neural network backbones and datasets. Extensive experimental results show the proposed $\mathcal{G}$-Mixup substantially improves the performance of graph neural networks in terms of enhancing their generalization and robustness.

2 PRELIMINARIES

In this section, we first go over the notations used in this paper, and then introduce graph related concepts including graph homomorphism and graphons, which will be used for theoretical analysis in this work. Finally, we briefly review the graph neural networks for graph classification.

2.1 NOTATIONS

Given a graph G, we use $V(G)$ and $E(G)$ to denote its nodes and edges, respectively. The number of nodes is $\mathrm{v}(G) = |V(G)|$, and the number of edges is $\mathrm{e}(G) = |E(G)|$. We use G, H, I to denote graphs and $\mathcal{G}, \mathcal{H}, \mathcal{I}$ to denote graph set. $\mathbf{y}_{\mathcal{G}} \in \mathbb{R}^C$ denotes the label of graph set $\mathcal{G}$, where C is number of classes of graphs. A graph could contain some interesting and frequent patterns and subgraphs which are called *motifs*. The motifs in graph G is denoted as F_G. The set of motifs in graph set $\mathcal{G}$ is denoted as $\mathcal{F}_{\mathcal{G}}$. $W_{\mathcal{G}}$ denotes the graphon of graph set $\mathcal{G}$. $\mathbf{W}$ denotes the step function. $\mathrm{Unif}_{[0,1]}$ denotes the uniform distribution between 0 and 1. $\mathrm{Bern}(\cdot)$ denotes the Bernoulli distribution. $\mathbb{G}(n, W)$ denotes the random graph with n nodes based on graphon W.

2.2 Graph Homomorphism and Graphons

Graph Homomorphism. A graph homomorphism is an adjacency-preserving mapping between two graphs, i.e., mapping adjacent vertices in one graph to adjacent vertices in the other. Formally, a *graph homomorphism* ϕ: $F \to G$ is a map from $V(F)$ to $V(G)$, where if $\{u, v\} \in E(F)$, then $\{\phi(u), \phi(v)\} \in E(G)$. For two graphs H and G, there could be multiple graph homomorphisms between them. Let $\mathrm{hom}(H, G)$ denotes the total number of graph homomorphisms from graph H to graph G. For example, $\mathrm{hom}(\bullet, G) = |V(G)|$ if graph H is $\bullet$, $\mathrm{hom}(\bullet\!\!-\!\!\bullet, G) = 2|E(G)|$ if graph H is $\bullet\!\!-\!\!\bullet$, and $\mathrm{hom}(\triangle, G)$ is six times the number of triangles in G. There are in total $|V(G)|^{|V(H)|}$ mappings from H to G, but only some of them are homomorphisms. Thus, we define *homomorphism density* to measure the relative frequency that the graph H appears in graph G:

$$t(H, G) = \frac{\mathrm{hom}(H, G)}{|V(G)|^{|V(H)|}}.$$

For example, $t(\bullet, G) = |V(G)|/n^1 = 1$, $t(\bullet\!\!-\!\!\bullet, G) = 2|E(G)|/n^2$.

Graphon. A graphon (Airoldi et al., 2013) is a continuous, bounded and symmetric function $W : [0, 1]^2 \to [0, 1]$ which may be thought of as the weight matrix of a graph with infinite number of nodes. Then, given two points $u_i, u_j \in [0, 1]$, $W(i, j)$ represents the probability that nodes i and j be related with an edge. Various quantities of a graph can be calculated as a function of the graphon. For example, the degree of nodes in graphs can be easily extended to a degree distribution function in graphons, which is characterized by its graphon marginal $d_W(x) = \int_0^1 W(x, y)dy$. Similarly, the concept of homomorphism density can be naturally extended from graphs to graphons. Given an arbitrary graph motif F, its homomorphism density with respect to graphon W is defined by

$$t(F, W) = \int_{[0,1]^{V(F)}} \prod_{i,j \in E(F)} W(x_i, x_j) \prod_{i \in V(F)} dx_i.$$

For example, the edge density of graphon W is $t(\bullet\!\!-\!\!\bullet, W) = \int_{[0,1]^2} W(x, y)\; dxdy$, and the triangle density of graphon W is $t(\triangle, W) = \int_{[0,1]^3} W(x, y)W(x, z)W(y, z)\; dxdydz$.

2.3 Graph Classification with Graph neural networks

Given a set of graphs, graph classification aims to assign a class label for each graph G. Recently, graph neural networks have become the state-of-the-art approach for graph classification. Without loss of generalization, we present the formal expression of a graph convolution network (GCN) (Kipf & Welling, 2016). The forward propagation at k-th layer is described as the following:

$$\mathbf{a}_i^{(k)} = \mathrm{AGG}^{(k)}\left(\left\{\mathbf{h}_j^{(k-1)} : j \in \mathcal{N}(i)\right\}\right), \quad \mathbf{h}_i^{(k)} = \mathrm{COMBINE}^{(k)}\left(\mathbf{h}_i^{(k-1)}, \mathbf{a}_i^{(k)}\right), \tag{1}$$

where $\mathbf{h}_i^{(k)} \in \mathbb{R}^{n \times d_k}$ is the intermediate representation of node i at the k-th layer, $\mathcal{N}(i)$ denotes the neighbors of node i. $\mathrm{AGG}(\cdot)$ is an aggregation function to collect embedding representations from neighbors, and $\mathrm{COMBINE}(\cdot)$ combines neighbor representation and its representation at $(k-1)$-th layer followed by nonlinear transformation. For graph classification, a graph-level representation is obtained by summarizing all node-level representations in the graph by a readout function:

$$\mathbf{h}_G = \mathrm{READOUT}\left(\left\{\mathbf{h}_i^{(k)} : i \in E(G)\right\}\right), \quad \hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{h}_G), \tag{2}$$

where $\mathrm{READOUT}(\cdot)$ is the readout function, which can be a simple function such as average or sophisticated pooling function (Gao & Ji, 2019; Ying et al., 2018), $\mathbf{h}_G$ is the representation of graph G, and $\hat{\mathbf{y}} \in \mathbb{R}^C$ is the output estimating the probability that G belongs to each of the C classes.

3 Methodology

In this section, we first formally introduce the proposed $\mathcal{G}$-Mixup, and then present theoretical analysis of graph generation via graphon interpolation from the graph homomorphism density perspective.

3.1 $\mathcal{G}$ -MIXUP

Different from interpolation of image data in Euclidean space, adopting Mixup to graph data is nontrivial since graphs are irregular, unaligned and non-Euclidean data. In this work, we show that this challenge could be tackled via graphon theories. By intuition, graphon can be thought of as a generator to generate graphs. The real-world graphs can be regraded as generated form a graphon, which has same homomorphism density of arbitrary motif to that of graphon. With this in mind, we propose $\mathcal{G}$-Mixup, a class-level data augmentation via graphon interpolation. $\mathcal{G}$-Mixup interpolates different graph generator to obtain a new generator. Then, synthetic graphs are sampled based on the mixed graphon for graph data augmentation. The graphs sampled from this generator partially possess properties of all the original graphs. Formally, $\mathcal{G}$-Mixup can be formulated as the following:

$$\text{Graphon Estimation:} \quad \mathcal{G} \rightarrow W_{\mathcal{G}}, \mathcal{H} \rightarrow W_{\mathcal{H}}, \tag{3}$$

$$\text{Graphon Mixup:} \quad W_{\mathcal{I}} = \lambda W_{\mathcal{G}} + (1-\lambda) W_{\mathcal{H}}, \tag{4}$$

$$\text{Graph Generation:} \quad \{I_1, I_2, \cdots, I_k\} \overset{\text{i.i.d}}{\sim} \mathbb{G}(k, W_{\mathcal{I}}), \tag{5}$$

$$\text{Label Mixup:} \quad \mathbf{y}_{\mathcal{I}} = \lambda \mathbf{y}_{\mathcal{G}} + (1-\lambda)\mathbf{y}_{\mathcal{H}}, \tag{6}$$

where $W_{\mathcal{G}}, W_{\mathcal{H}}$ are graphons of the graph set $\mathcal{G}$ and $\mathcal{H}$. The mixed graphon is denoted by $W_{\mathcal{I}}$, and $\lambda \in [0,1]$ is the trade-off hyperparameter to control the contributions from different original graphs for interpolation. The set of synthetic graphs generated from $W_{\mathcal{I}}$ is $\mathcal{I} = \{I_1, I_2, \cdots, I_k\}$. The $\mathbf{y}_{\mathcal{G}} \in \mathbb{R}^C$ and $\mathbf{y}_{\mathcal{H}} \in \mathbb{R}^C$ are vectors containing ground-truth labels for graph G and H respectively. The label vector of a generated graph in $\mathcal{I}$ is set as $\mathbf{y}_{\mathcal{I}} \in \mathbb{R}^C$, where C is the total classes of graphs.

As illustrated in Figure 1 and the above equations, the proposed $\mathcal{G}$-Mixup includes three key steps: **i) estimate the graphon** for each class of graphs with the same label, **ii) mix up the graphons** of different classes of graphs, and **iii) generate the synthetic graphs** based on the mixed graphon. Specifically, we have $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ with label $\mathbf{y}_G$, and $\mathcal{H} = \{H_1, H_2, \cdots, H_k\}$ with label $\mathbf{y}_{\mathcal{H}}$. Graphons $W_{\mathcal{G}}$ and $W_{\mathcal{H}}$ are estimated from graph set $\mathcal{G}$ and $\mathcal{H}$, then we mix them up though linearly interpolating two graphons and their training labels and obtain $W_{\mathcal{I}}$ and $\mathbf{y}_{\mathcal{I}}$. The synthetic graph set $\mathcal{I}$ is sampled based on $W_{\mathcal{I}}$, which will be used as additional training graphs.

3.2 IMPLEMENTATION

In this section, we introduce the details of the implementation of estimating graphon from the observed graphs and generating synthetic graphs in the real-world scenario.

Graphon Estimation and Mixup. Estimating graphons from observed graphs is a prerequisite for $\mathcal{G}$-Mixup, however, it is intractable because graphon is an unknown function without closed-form solution in real-world graphs. Therefore, we use step function (Lovász, 2012; Xu et al., 2021) as an approximation of graphon. [1] The step function estimation methods are well-studied, which first aligns the nodes of a series of graphs based on simple node measurements (e.g., degree) and then estimates the step function from all the aligned adjacency matrices. The step function estimation methods used includes sorting-and-smoothing (SAS) method (Chan & Airoldi, 2014), stochastic block approximation (SBA) (Airoldi et al., 2013), "largest gap" (LG) (Channarond et al., 2012), matrix completion (MC) (Keshavan et al., 2010), universal singular value thresholding (USVT) (Chatterjee et al., 2015). Formally, a step function $\mathbf{W}^P : [0,1]^2 \mapsto [0,1]$ is define as $\mathbf{W}^P(x,y) = \sum_{k,k'=1}^{K} w_{kk'} \mathbb{1}_{\mathcal{P}_k \times \mathcal{P}_{k'}}(x,y)$, where $\mathcal{P} = (\mathcal{P}_1, .., \mathcal{P}_K)$ denotes the partition of $[0,1]$ into K adjacent intervals of length $1/K$, $w_{kk'} \in [0,1]$, and indicator function $\mathbb{1}_{\mathcal{P}_k \times \mathcal{P}_{k'}}(x,y)$ equals to 1 if $(x,y) \in \mathcal{P}_k \times \mathcal{P}_{k'}$ and otherwise it is 0. *Essentially, the step function can be seen as a matrix* $\mathbf{W} = [w_{kk'}] \in [0,1]^{K \times K}$, *where* $\mathbf{W}_{ij}$ *is the probability of edge between node* i *and* j. *In practice, we use the matrix-form step function as graphon to mix up and generation synthetic graphs.*

For binary classification, we have $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ and $\mathcal{H} = \{H_1, H_2, \cdots, H_k\}$ with different labels, we estimate their step functions $\mathbf{W}_{\mathcal{G}} \in \mathbb{R}^{K \times K}$ and $\mathbf{W}_{\mathcal{H}} \in \mathbb{R}^{K \times K}$ We let K be the average number of nodes in all graphs. *For multi-class classification*, we first estimate the step function for each class of graphs and then randomly select two to perform mix-up. The resultant step function is $\mathbf{W}_{\mathcal{I}} = \lambda \mathbf{W}_{\mathcal{G}} + (1-\lambda)\mathbf{W}_{\mathcal{H}} \in \mathbb{R}^{K \times K}$, which serves as the generator of synthetic graphs.

[1]Because weak regularity lemma of graphon (Frieze & Kannan, 1999) indicates that an arbitrary graphon can be approximated well by step function. Detailed discussion is in Appendix A.4.

Synthetic Graphs Generation. A graphon W provides a distribution to generating arbitrarily sized graphs. Specifically, a k-node random graph $\mathbb{G}(k, W_{\mathcal{I}})$ can be generated following the process:

$$u_1, \ldots, u_k \overset{\text{iid}}{\sim} \text{Unif}_{[0,1]}, \quad (\mathbb{G}(k, W)_{ij} | u_1, \ldots, u_k) \overset{\text{iid}}{\sim} \text{Bern}(W(u_i, u_j)), \forall i, j \in [k]. \tag{7}$$

Since we only estimate the step function $\mathbf{W}$ to approximate the graph W, we set $W(u_i, u_j) = \mathbf{W}_I[\lfloor 1/u_i \rfloor, \lfloor 1/u_j \rfloor]$, and $\lfloor \cdot \rfloor$ is the floor function. The first step samples K nodes independently from a uniform distribution on $[0, 1]$. The second step generates an adjacency matrix $\boldsymbol{A} = [a_{ij}] \in \{0, 1\}^{K \times K}$, whose element values follow the Bernoulli distributions determined by the step function. A graph is thus obtained as G where $V(G) = \{1, ..., K\}$ and $E(G) = \{(i, j) \mid a_{ij} = 1\}$. A set of synthetic graphs can be generated by conducting the above process independently multiple times. For node features, we generate them of synthetic graphs based the original two sets of graphs. Specifically, we generate the node feature of each graphons. In the graphon estimation, we align the node features with the process of the adjacency matrix. For each graphon we have a set of node features, we can pooling the node features to obtain the *graphon features*. The the node features would inherit form the graphon features.

Computational Complexity Analysis. We now discuss computational complexity of the proposed $\mathcal{G}$-Mixup. The major additional computation costs come from graphon estimation and graph generation. *For graphon estimation*, suppose we have M graphs and each of them has N nodes and E edges, and estimate step function with K partitions to approximate a graphon, the complexity of used graphon estimation methods (Xu et al., 2021) is in Table 1. *For graph generation*, suppose we need to generate K graphs with N nodes, the computational complexity is $\mathcal{O}(KN)$ for node generation and $\mathcal{O}(KN^2)$ for edge generation, so the overall complexity of graph generation is $\mathcal{O}(KN^2)$.

Table 1: Computational complexity of graphon estimation (Xu et al., 2021)

Method	Complexity
MC	$\mathcal{O}(N^3)$
USVT	$\mathcal{O}(N^3)$
LG	$\mathcal{O}(MN^2)$
SBA	$\mathcal{O}(MKN \log N)$
SAS	$\mathcal{O}(MN \log N + K^2 \log K^2)$

4 Theoretical Justification

In the following, we will theoretically prove that, *the synthetic graphs generated by G-Mixup will be a mixture of original graphs*. We first define the discriminative motif, and then we justify the graphon mixup operation (Equation 4) and graph generation operation (Equation 5) by analysing the homomorphism density of discriminative motifs of the original graphs and the synthetic graphs.

Definition 1 (Discriminative Motif) *A discriminative motif F_G of graph G is the subgraph, with the minimal number of nodes and edges, that can decide the class the graph G. Discriminative motifs $\mathcal{F}_{\mathcal{G}}$ is the set of the discriminative motif of every graph in the graph set $\mathcal{G}$.*

Intuitively, the discriminative motif is the key topology of a graph, by which the graph can be distinguished. We assume that *(i) every graph G has a discriminative motif F_G* , and *(ii) a set of graphs $\mathcal{G}$ has a finite set of discriminative motifs $\mathcal{F}_{\mathcal{G}}$*. The goal of graph classification is to filter out structural noise in graphs (Fox & Rajamanickam, 2019) and recognize the key typologies (discriminative motifs) to predict the class label. For example, benzene (a compound in chemistry) is distinguished by the discriminative motif (benzene ring).

4.1 Will discriminative motifs $F_{\mathcal{G}}$ and $F_{\mathcal{H}}$ exist in $\lambda W_{\mathcal{G}} + (1 - \lambda) W_{\mathcal{H}}$?

We answer this question by exploring the difference in homomorphism density of discriminative motifs between the original graphon and mixed graphon. We propose the following theorem,

Theorem 1 *Given two sets of graphs $\mathcal{G}$ and $\mathcal{H}$, the corresponding graphons are $W_{\mathcal{G}}$ and $W_{\mathcal{H}}$, and the corresponding discriminative motif set $\mathcal{F}_{\mathcal{G}}$ and $\mathcal{F}_{\mathcal{H}}$. For every discriminative motif $F_{\mathcal{G}} \in \mathcal{F}_{\mathcal{G}}$ and $F_{\mathcal{H}} \in \mathcal{F}_{\mathcal{H}}$, the difference between the homomorphism density of $F_{\mathcal{G}}/F_{\mathcal{H}}$ in the mixed graphon $W_{\mathcal{I}} = \lambda W_{\mathcal{G}} + (1 - \lambda) W_{\mathcal{H}}$ and that of the graphon $W_{\mathcal{H}}/W_{\mathcal{G}}$ is upper bounded by*

$$\begin{aligned} |t(F_{\mathcal{G}}, W_{\mathcal{I}}) - t(F_{\mathcal{G}}, W_{\mathcal{G}})| &\leq (1 - \lambda) \mathrm{e}(F_{\mathcal{G}}) ||W_{\mathcal{H}} - W_{\mathcal{G}}||_{\square} \,, \\ |t(F_{\mathcal{H}}, W_{\mathcal{I}}) - t(F_{\mathcal{H}}, W_{\mathcal{H}})| &\leq \lambda \mathrm{e}(F_{\mathcal{H}}) ||W_{\mathcal{H}} - W_{\mathcal{G}}||_{\square} \end{aligned} \tag{8}$$

where $\mathrm{e}(F)$ is the number of nodes in graph F and $||W_{\mathcal{H}} - W_{\mathcal{G}}||_{\square}$ is the cut norm [2].

[2] Details about cut norm are in Appendic A.1

Proof Sketch. The proof follows the derivation of Counting Lemma for Graphons (Lemma 10.23 in Lovász (2012)), which relates the homomorphism density and the cut distance $||W_{\mathcal{H}} - W_{\mathcal{G}}||$ of graphons. Specifically, we take the two graphons in this Lemma to deduce the bound of the difference of homomorphism densities of $W_{\mathcal{I}}$ and $W_{\mathcal{G}}/W_{\mathcal{H}}$. Detailed proof are in Appendix A.2. ■

Theorem 1 suggests that the difference in the homomorphism densities of the mixed graphon and original graphons is upper bounded. Note that difference depends on the hyperparameter λ, the edge number $\mathrm{e}(F_{\mathcal{G}})/\mathrm{e}(F_{\mathcal{H}})$ and the cut norm $||W_{\mathcal{H}} - W_{\mathcal{G}}||_{\square}$. Since the $\mathrm{e}(F_{\mathcal{G}})/\mathrm{e}(F_{\mathcal{H}})$ and the cut norm $||W_{\mathcal{H}} - W_{\mathcal{G}}||_{\square}$ are decided by the dataset (can be regraded as a constant), the difference in homomorphism densities will be decided by λ. On this basis, the corresponding label of the graphon is set to $\lambda \mathbf{y}_{\mathcal{G}} + (1-\lambda)\mathbf{y}_{\mathcal{H}}$. **Therefore, $\mathcal{G}$-Mixup can preserve the different discriminative motifs of the two different graphons into one mixed graphon.**

4.2 Will the generated graphs from graphon $W_{\mathcal{I}}$ preserve the mixture of discriminative motifs?

Ideally, the generated graphs should inherit the homomorphism density of discriminative motifs from the graphon. To verify this, we propose the following theorem.

Theorem 2 *Let W_I be the mixed graphon , $n \geq 1$, $0 < \varepsilon < 1$, and let F_I be the mixed discriminative motif, then the $W_{\mathcal{I}}$-random graph $\mathbb{G} = \mathbb{G}(n, W_{\mathcal{I}})$ satisfies*

$$\mathrm{P}\left(|t(F_{\mathcal{I}}, \mathbb{G}) - t(F_{\mathcal{I}}, W_{\mathcal{I}})| > \varepsilon\right) \leq 2\exp\left(-\frac{\varepsilon^2 n}{8\mathrm{v}(F_I)^2}\right). \tag{9}$$

Theorem 2 states that for any nonzero specified margin ε, no matter how small, with a sufficiently large samples of graph from mixed graphon, the homomorphism density of discriminative motif in synthetic graphs will approximately equal to that in graphon $t(F_{\mathcal{I}}, \mathbb{G}) \approx t(F_{\mathcal{I}}, W_{\mathcal{I}})$ with high probability. In other words, it reads the synthetic graphs will preserve the discriminative motif of the mixed graphon with a very high probability if the sample number n is large enough. The detailed proof is presented in Appendix A.3. **Therefore, $\mathcal{G}$-Mixup can preserve the different discriminative motifs of the two different graphs into one mixed graph.**

4.3 Discussion

This section discusses the differences and relations between $\mathcal{G}$-Mixup and other graph augmentation strategies, such as DropEdge (Rong et al., 2020), and Manifold Mixup (Wang et al., 2021).

Relation to Edge Perturbation Method . Edge perturbation methods is to randomly perturb the edges to improve the GNNs, inlcuding DropEdge (Rong et al., 2020), Graphon-based edge perturbation (Hu et al., 2021). DropEdge drops graph edges independently with a specified probability, aiming to prevent over-smoothing and over-fitting issues in GNNs. Graphon-based edge perturbation (Rong et al., 2020) improves the Dropedge by dropping edge based on an estimated probability. One of the limitations of such methods is that the edge permutation is based on the one individual graph, the graphs will not mix up. Edge perturbation methods will not mix the two discriminative motifs together and only randomly add remove some edges while the graphon will add some edges based on the graphon. DropEdge and Graphon-based edge perturbation (Hu et al., 2021) are special cases of $\mathcal{G}$-Mixup while setting different hyperparameter λ. *i) $\mathcal{G}$-Mixup will degenerate into Graphon-based edge perturbation*, while $\lambda = 0$ in Equation 4. The mathematical expression is $W_{\mathcal{I}} = W_{\mathcal{H}}, \{I_1, I_2, \cdots, I_k\} \overset{\text{i.i.d}}{\sim} \mathbb{G}(k, W_{\mathcal{I}}), \mathbf{y}_{\mathcal{I}} = \mathbf{y}_{\mathcal{H}}$. *ii) $\mathcal{G}$-Mixup will degenerate into DropEdge*, while setting $\lambda = 0$ and masking graphons with adjacency matrix $\mathbf{A}$ in Equation 4. The expression is $W_{\mathcal{I}} = \mathbf{A} \odot W_{\mathcal{H}}, \{I_1, I_2, \cdots, I_k\} \overset{\text{i.i.d}}{\sim} \mathbb{G}(k, W_{\mathcal{I}}), \mathbf{y}_{\mathcal{I}} = \mathbf{y}_{\mathcal{H}}$, where $\odot$ is element-wise multiplication.

Relation to Manifold Mixup for Graph. The methods proposed by Wang et al. (2021) is to adopt Manifold Mixup to graph data, which mix up graphs representation in the embedding space. The Manifold Mixup is to stabilize the model training by interpolating hidden representation. Interpolating hidden representation limits its applications by that 1) learning algorithms must have hidden representation of graphs and 2) models must be modified to adapt Manifold Mixup. In contrast, $\mathcal{G}$-Mixup is capable of generating synthetic graphs without modifying models. As a data augmentation method, our proposal has broader applications, e.g., creating graphs for graph contrastive learning.

5 Experiments

We evaluate the performance of the proposed $\mathcal{G}$-Mixup in this section. First, we visualize graphons and graph generation results to investigate what $\mathcal{G}$-Mixup actually do on real-world datasets in Section 5.1 and Section 5.2. Then, we evaluate the effectiveness of $\mathcal{G}$-Mixup in graph classification with various datasets and GNN backbones in Section 5.3, as well as how it improve the robustness of GNNs against label corruption and adversarial examples in Section 5.4.

5.1 Does different classes of real-world graphs have different graphons?

We visualize the estimated graphon in Figure 2 to examine whether there are different graphons for different classes of graphs. **Observation 1: different classes of graphs have different graphons in real-world dataset.** As shown in Figure 2, the graphons of different class of graphs in one dataset are distinctly different. The graphons of IMDB-BINAERY in Figure 2 shows that the graphon of class 1 has larger dense area, which indicates that the graphs in this class have a more large communities than the graphs of class 0. The graphons of REDDIT-BINARY in Figure 2 shows that graphs of class 0 have one high-degree nodes while the graphs of class 1 have two. This observation validates that real-world graphs of different classes have distinctly different graphons, which lays a solid foundation for generating the mixture of graphs by mixing up graphons.

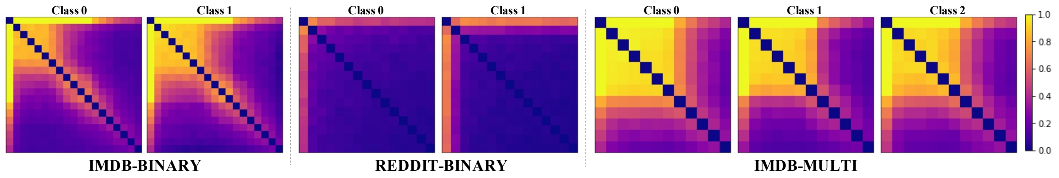

Figure 2: Estimated graphons on IMDB-BINARY, REDDIT-BINARY, and IMDB-MULTI. Obviously, graphons of different classes of graphs are quiet different. This observation validates the divergence of graphons between different classes of graphs, which is the basis of the $\mathcal{G}$-Mixup. The graphons are estimated by LG. More estimated graphons via various methods are in Appendix B.3.

5.2 What is $\mathcal{G}$-Mixup doing? A case study

To investigate the outcome of $\mathcal{G}$-Mixup in real-world scenarios, we visualize the generated synthetic graphs in REDDIT-BINARY dataset in Figure 3. **Observation 2: synthetic graphs are indeed the mixture of the original graphs.** Original graphs and the generated synthetic graphs are visualized in Figure 3(a)(b) and Figure 3(c)(d)(e), respectively. Figure 3 demonstrates that mixed graphon $0.5 * W_{\mathcal{G}} + 0.5 * W_{\mathcal{H}}$ is able to generate graphs with a high-degree node and a dense subgraph, which can be regarded as the mixture of graphs with one high-degree node and two high-degree nodes. It validates that $\mathcal{G}$-Mixup prefer to preserve the discriminative motifs from the original graphs.

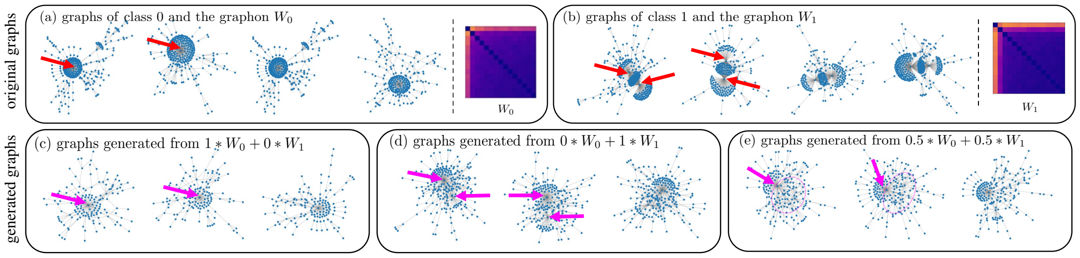
original graphs
(a) graphs of class 0 and the graphon W_0
W_0
(b) graphs of class 1 and the graphon W_1
W_1
generated graphs
(c) graphs generated from $1 * W_0 + 0 * W_1$
(d) graphs generated from $0 * W_0 + 1 * W_1$
(e) graphs generated from $0.5 * W_0 + 0.5 * W_1$

Figure 3: The visualization of generated synthetic graphs on dataset REDDIT-BINARY. The first row is the original graphs in the dataset while the second row is the generated graphs through the proposed $\mathcal{G}$-Mixup. The graphs in (a) and (b) are the original graphs of class 0 and class 1. The distinct difference of these two classes of graphs is that graphs of class 0 have one high-degree node while graphs of class 1 have two (marked with → in (a) and (b)). (c)/(d) shows graphs generated with the mixed graphon $(1 * W_0 + 0 * W_1)$ / $(0 * W_0 + 1 * W_1)$, which have one/two high-degree node/nodes (marked with → in (c) and (d)) because the mixed graphon only contains W_0/W_1. The synthetic graphs generated from $(0.5 * W_0 + 0.5 * W_1)$ is the mixture of graphs of class 0 and class 1, which appears as a high-degree node and a dense subgraph (marked with → and ○ in (e), respectively). The visualization shows that synthetic graphs are the mixture of the original graphs.

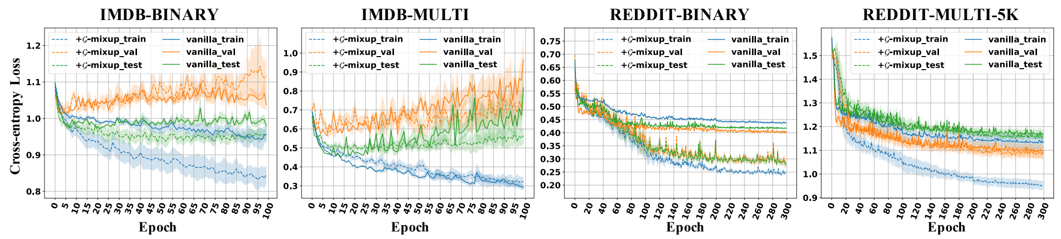

Figure 4: The training/validation/test curves on IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY and REDDIT-MULTI-5K with GCN as backbone. The curves are depicted on ten runs.

Table 2: Performance comparisons of $\mathcal{G}$-Mixup with different graph neural networks on different dataset. The metric is the classification accuracy and its standard deviation. The best performance is in **boldface**. Experimental settings are in Appendix B.1. Experiments with more backbones (DiffPool, MincutPool, GMT) and molecular property prediction are in Appendix C.1 and Appendix C.2.

	Dataset	IMDB-B	IMDB-M	REDDIT-B	REDD-M5k	REDD-M12k
	#graphs	1000	1500	2000	4999	11929
	#classes	2	3	2	5	11
	#avg.nodes	19.77	13.00	429.63	508.52	391.41
	#avg.edges	96.53	65.94	497.75	594.87	456.89
GCN	vanilla	72.18±1.55	48.79±2.72	78.82±1.33	45.07±1.70	46.90±0.73
	w/ Dropedge	72.50±0.31	49.08±1.89	81.25±8.15	51.35±1.54	47.08±0.55
	w/ NodeDropping	72.00±4.09	48.58±2.85	79.25±0.35	49.35±1.80	47.93±0.64
	w/ Subgraph	68.50±4.76	49.58±2.61	74.33±2.88	48.70±1.63	47.49±0.93
	w/ ManfoldMixup	72.83±1.75	49.50±1.97	75.75±4.53	49.82±0.85	46.92±1.05
	w/ $\mathcal{G}$-Mixup	**72.87**±3.85	**51.30**±2.14	**89.81**±0.74	**51.51**±1.70	**48.06**±0.53
GIN	vanilla	71.55±3.53	48.83±2.75	92.59±0.86	55.19±1.02	50.23±0.83
	w/ Dropedge	**72.20**±1.82	48.83±3.02	92.00±1.13	55.10±0.44	49.77±0.76
	w/ NodeDropping	72.16±0.28	48.33±0.98	90.25±0.98	53.26±4.99	49.95±1.70
	w/ Subgraph	68.50±0.86	47.25±3.78	90.33±0.87	54.60±3.15	49.67±0.90
	w/ ManfoldMixup	70.83±1.04	49.88±1.34	90.75±1.78	54.95±0.86	49.81±0.80
	w/ $\mathcal{G}$-Mixup	71.94±3.00	**50.46**±1.49	**92.90**±0.87	**55.49**±0.53	**50.50**±0.41
TopKPool	vanilla	72.37±5.01	50.57±1.62	90.30±1.47	45.07±1.70	**45.06**±1.70
	w/ Dropedge	71.75±2.18	48.75±2.94	88.96±1.90	**47.43**±1.82	44.56±1.41
	w/ NodeDropping	69.16±1.04	48.50±2.50	81.33±4.48	46.15±2.28	44.49±1.15
	w/ Subgraph	67.83±4.01	50.83±2.38	86.08±2.12	45.75±2.47	46.18±1.53
	w/ ManfoldMixup	71.83±3.03	51.22±1.17	87.58±3.16	45.60±2.35	43.81±0.95
	w/ $\mathcal{G}$-Mixup	**72.80**±3.33	**51.30**±2.14	**90.40**±0.89	46.48±1.70	43.72±1.65

5.3 CAN $\mathcal{G}$-MIXUP IMPROVE THE PERFORMANCE AND GENERALIZATION OF GNNS?

To validate the effectiveness of our proposal, we experiment to compare the performance with various backbones of GNNs on various datasets, and summarize results in Table 2 as well as the training curves in Figure 4. Our main observations are: **Observation 3: $\mathcal{G}$-Mixup can significantly improves the performance of various graph neural networks on various datasets.** From Table 2, our proposal gain 12 best performances among 15 reported accuracies, which substantially improve the performance of GNNs. Overall, our proposal performs 2.84% better than vanilla model. Note that $\mathcal{G}$-Mixup and baseline models adopt the same architecture of GNNs (e.g., layers, activation functions) and the same training hyperparameters (e.g., optimizer, learning rate). Considering both model performance and experimental setting, the improvement adequately validates the effectiveness of our proposal. **Observation 4: $\mathcal{G}$-Mixup can significantly improve the generalization of various backbones of graph neural networks.** From the loss curve on test data (green line) in Figure 4, the loss of test data of $\mathcal{G}$-Mixup (dashed green lines) are consistently lower than the vanilla model (solid green lines). Considering both the better performance and the better test loss curves, our proposal substantially is able to improve the generalization of GNNs. **Observation 5: $\mathcal{G}$-Mixup largely stabilizes the model training.** As shown in Table 2, $\mathcal{G}$-Mixup achieves 11 lower standard deviation among total 15 reported numbers than vanilla model. Additionally, the train/validation/test curves of $\mathcal{G}$-Mixup (dashed line) in Figure 4 are more stable than vanilla model (solid line). These all indicate that our proposal $\mathcal{G}$-Mixup is capable of stabilizing the training of graph neural networks.

5.4 Can $\mathcal{G}$-Mixup virtually improve the robustness of GNNs?

We investigate the two kinds of robustness of the proposed $\mathcal{G}$-Mixup, including *Label Corruption Robustness* and *Topology Corruption Robustness*, and report the results in Table 3 and 4, respectively. More experimental settings are presented in Appendix B.2. **Observation 6: $\mathcal{G}$-Mixup improves the robustness of graph neural networks.** Table 3 shows our proposal gains the better performance, indicating it is more robust to noisy label than vanilla baseline. Table 4 shows that $\mathcal{G}$-Mixup is more robust when graph topology is corrupted since the accuracy is consistently better than baselines. This can be an advantage of $\mathcal{G}$-Mixup when graph labels or topologies are noisy.

Table 3: Robustness to label corruption with different corruption ratio.

Models	Methods	10%	20%	30%	40%
IMDB-B	vanilla	72.30±3.67	69.43±4.80	63.65±8.87	**55.21**±8.75
	w/ Dropedge	72.00±2.44	69.52±3.25	64.12±3.44	48.50±0.00
	w/ ManfoldMixup	71.87±3.56	69.03±4.85	65.62±9.89	48.50±0.00
	w/ $\mathcal{G}$-Mixup	**72.56**±3.08	**69.87**±5.41	**65.50**±8.90	52.56±6.97
REDDIT-B	vanilla	**73.90**±1.43	75.68±2.75	68.12±0.81	46.50±0.00
	w/ Dropedge	73.75±1.28	72.06±1.42	46.50±0.00	46.50±0.00
	w/ ManfoldMixup	71.96±1.97	76.00±2.24	54.43±1.09	46.50±0.00
	w/ $\mathcal{G}$-Mixup	71.94±3.00	**76.34**±1.49	**74.21**±1.85	**53.50**±0.00

Table 4: Robustness to topology corruption with different corruption ratio.

Models	Methods	10%	20%	30%	40%
Removing edges	vanilla	77.96±3.71	67.59±5.73	64.96±8.87	65.71±8.31
	w/ Dropedge	74.40±2.26	65.12±3.51	65.93±2.32	57.87±4.14
	w/ ManfoldMixup	75.62±1.59	65.81±3.84	59.81±9.45	57.31±3.15
	w/ $\mathcal{G}$-Mixup	**81.46**±3.08	**71.12**±7.47	**67.46**±8.90	**66.25**±7.78
Adding edges	vanilla	76.12±5.73	74.37±6.48	72.31±2.69	72.00±2.92
	w/ Dropedge	70.53±1.47	70.18±1.29	71.18±1.53	70.90±1.53
	w/ ManfoldMixup	73.41±2.40	71.87±1.28	71.50±2.03	71.21±2.00
	w/ $\mathcal{G}$-Mixup	**84.31**±3.21	**82.21**±4.31	**77.00**±2.25	**75.56**±3.05

6 Related Works

Graph Data Augmentation Graph neural networks (GNNs) also achieve the state-of-the-art performance on graph classification task (Kipf & Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017; Xu et al., 2018; Zhang et al., 2018). In parallel, graph data augmentation methods are also proposes to improve the performance of GNNs. There are three categories of graph data augmentation, including node perturbation (You et al., 2020), edge perturbation (Rong et al., 2020; You et al., 2020), and subgraph sampling (You et al., 2020). We have discussed the detailed differences to the mainstream graph augmentation method in Section 4.3. However, the common limitation of the existing graph data augmentation methods is that these methods are based on one single graph while $\mathcal{G}$-Mixup is able to augment new graphs using multiple input graphs. Besides, there are model-independent graph data augmentation methods (Zhou et al., 2020a; Zhao et al., 2021) and graph data augmentation method for node classification, which is not applicable while our methods is genral plug-in model-agnostic graph data augmentation methods for graph classification.

Graphon Estimation. Graphons and convergent graph sequences have been broadly studied in mathematics (Lovász, 2012; Lovász & Szegedy, 2006; Borgs et al., 2008) and have been applied to to network science (Avella-Medina et al., 2018; Vizuete et al., 2021) and graph neural networks (Ruiz et al., 2020a;b). There are tow lines of works to estimate step functions, one is based on stochastic block models, such as sorting-and-smoothing (SAS) method (Chan & Airoldi, 2014), stochastic block approximation (SBA) (Airoldi et al., 2013), "largest gap" (LG) (Channarond et al., 2012); another one is based on matrix decomposition, such as matrix completion (MC) (Keshavan et al., 2010), universal singular value thresholding (USVT) (Chatterjee et al., 2015). We estimate the step function using the above five methods (Xu et al., 2021) and the estimation are in Appendix B.3.

7 Conclusion

This work develops $\mathcal{G}$-Mixup to augment graph data. Unlike image data, graph data is irregular, unaligned and in non-Euclidean space, making it hard to be mixed up. However, the graphs within one class have the same generator (i.e., graphon), which is regular, well-aligned and in Euclidean space. Thus we turn to mix up the graphons of different classes then generate synthetic graphs. $\mathcal{G}$-Mixup is a new graph data augmentation algorithm which mix up the input graph to interpolate the topologies of different classes of graphs. A variety of experiments have shown that graph neural networks trained with $\mathcal{G}$-Mixup achieve better performance and generalization in terms of model accuracy and model loss, and improve the robustness to noisy labels and corrupted topologies.

REFERENCES

Edoardo M Airoldi, Thiago B Costa, and Stanley H Chan. Stochastic blockmodel approximation of a graphon: Theory and consistent estimation. *arXiv preprint arXiv:1311.1731*, 2013.

Marco Avella-Medina, Francesca Parise, Michael T Schaub, and Santiago Segarra. Centrality measures for graphons: Accounting for uncertainty in networks. *IEEE Transactions on Network Science and Engineering*, 7(1):520–537, 2018.

Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations*, 2020.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pp. 874–883. PMLR, 2020.

Christian Borgs, Jennifer T Chayes, László Lovász, Vera T Sós, and Katalin Vesztergombi. Convergent sequences of dense graphs i: Subgraph frequencies, metric properties and testing. *Advances in Mathematics*, 219(6):1801–1851, 2008.

Stanley Chan and Edoardo Airoldi. A consistent histogram estimator for exchangeable graph models. In *International Conference on Machine Learning*, pp. 208–216, 2014.

Antoine Channarond, Jean-Jacques Daudin, Stéphane Robin, et al. Classification and estimation in the stochastic blockmodel based on the empirical degrees. *Electronic Journal of Statistics*, 6: 2574–2601, 2012.

Sourav Chatterjee et al. Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43(1):177–214, 2015.

James Fox and Sivasankaran Rajamanickam. How robust are graph neural networks to structural noise? *arXiv preprint arXiv:1912.10206*, 2019.

Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

Hongyang Gao and Shuiwang Ji. Graph u-nets. *arXiv preprint arXiv:1905.05178*, 2019.

Hongyu Guo. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Hongyu Guo, Yongyi Mao, and Richong Zhang. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020.

Ziqing Hu, Yihao Fang, and Lizhen Lin. Training graph neural networks by graphon estimation, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE transactions on information theory*, 56(6):2980–2998, 2010.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012.

László Lovász and Balázs Szegedy. Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B*, 96(6):933–957, 2006.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020a.

Luana Ruiz, Zhiyang Wang, and Alejandro Ribeiro. Graph and graphon neural network stability. *arXiv preprint arXiv:2010.12529*, 2020b.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pp. 6438–6447. PMLR, 2019.

Renato Vizuete, Federica Garin, and Paolo Frasca. The laplacian spectrum of large graphs sampled from graphons. *IEEE Transactions on Network Science and Engineering*, 2021.

Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, pp. 3663–3674, 2021.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Hongteng Xu, Dixin Luo, Lawrence Carin, and Hongyuan Zha. Learning graphons via structured gromov-wasserstein barycenters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 10505–10513, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 4805–4815, 2018.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2017.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11015–11023, 2021.

Yufei Zhao. Graph theory and additive combinatorics, 2019. URL https://yufeizhao.com/gtac/.

Jiajun Zhou, Jie Shen, and Qi Xuan. Data augmentation for graph classification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2341–2344, 2020a.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020b.

A Proof of Theorem

In the appendix, we first present the preliminaries in Appendix A.1. And then we present complete proof for Theorem 1 and Theorem 2 in Section A.3 and A.2, respectively.

A.1 Preliminaries

Cut norm (Lovász, 2012; Zhao, 2019) is used to measure structural similarity of two graphons. The definition of cut norm is as follow:

Definition 2 *The cut norm of grapon W is defined as*

$$\|W\|_\square = \sup_{S,T\subset[0,1]}\left|\int_{S\times T} W(x,y)dxdy\right|, \tag{10}$$

where the supremum is taken over all measurable subsets S and T .

The following lemma follows the derivation of counting lemma for graphons, are known in the paper (Lovász, 2012). It will be used to prove the Theorem 1.

Lemma 1 *Let F be a simple graph and let $W, W' \in \mathcal{W}$. Then*

$$|t(F,W) - t(F,W')| \leq \mathrm{e}(F)||W - W'||_\square \tag{11}$$

Proof of Lemma 1. The proof follows Zhao (2019). For an arbitrary simple graph F, by the triangle inequality we have

$$\begin{aligned}
&|t(F,W) - t(F,W')| \\
&= \left|\int\left(\prod_{u_iv_i\in E} W\left(u_i,v_i\right) - \prod_{u_iv_i\in E} W'\left(u_i,v_i\right)\right)\prod_{v\in V} dv\right| \\
&\leq \sum_{i=1}^{|E|}\left|\int\left(\prod_{j=1}^{i-1} W'\left(u_j,v_j\right)\left(W\left(u_i,v_i\right) - W'\left(u_i,v_i\right)\right)\prod_{k=i+1}^{|E|} W\left(u_k,v_k\right)\right)\prod_{v\in V} dv\right|
\end{aligned} \tag{12}$$

Here, each absolute value term in the sum is bounded by the cut norm $\|W - W'\|_\square$ if we fix all other irrelavant variables (everything except u_i and v_i for the i-th term), altogether implying that

$$|\ t(F,W) - t(F,W')| \leq \mathrm{e}(F)||W - W'||_\square \tag{13}$$

■

Lemma 2 (Corollary 10.4 in Lovász & Szegedy (2006)) *Let W be a graphon, $n \geq 1$, $0 < \varepsilon < 1$, and let F be a simple graph, then the W-random graph $\mathbb{G} = \mathbb{G}(n, W)$ satisfies*

$$\mathrm{P}\left(|t(F,\mathbb{G}) - t(F,W)| > \varepsilon\right) \leq 2\exp\left(-\frac{\varepsilon^2 n}{8\mathrm{v}(F)^2}\right) \tag{14}$$

A.2 Proof of Theorem 1

We have the mixed graphon $W_\mathcal{I} = \lambda W_\mathcal{G} + (1-\lambda)W_\mathcal{H}$. Let $W = W_\mathcal{I}$, $W' = W_\mathcal{G}$, and $F = F_\mathcal{G}$ in Lemma 1, we have,

$$\begin{aligned}
|t(F_\mathcal{G},W_\mathcal{I}) - t(F_\mathcal{G},W_\mathcal{G})| &\leq \mathrm{e}(F_\mathcal{G})||W_\mathcal{I} - W_\mathcal{G}||_\square \\
|t(F_\mathcal{G},\lambda W_\mathcal{G} + (1-\lambda)W_\mathcal{H}) - t(F,W_\mathcal{G})| &\leq \mathrm{e}(F_\mathcal{G})||\lambda W_\mathcal{G} + (1-\lambda)W_\mathcal{H} - W_\mathcal{G}||_\square \\
&\leq \mathrm{e}(F_\mathcal{G})||(1-\lambda)(W_\mathcal{H} - W_\mathcal{G})||_\square
\end{aligned} \tag{15}$$

Recall that the cut norm $\|W\|_\square = \sup_{S,T\subseteq[0,1]}\left|\int_{S\times T} W\right|$.

obviously, suppose $\alpha \in \mathbb{R}$, we have

$$\|\alpha W\|_\square = \sup_{S,T\subseteq[0,1]} \left| \int_{S\times T} \alpha W \right| = \sup_{S,T\subseteq[0,1]} \left| \alpha \int_{S\times T} W \right| = \alpha \|W\|_\square \tag{16}$$

Based on Equation 15 and Equation 16, we have

$$\begin{aligned} |t(F_\mathcal{G}, \lambda W_\mathcal{G} + (1-\lambda)W_\mathcal{H}) - t(F_\mathcal{G}, W_\mathcal{G})| &\leq \mathrm{e}(F_\mathcal{G})||(1-\lambda)(W_\mathcal{H} - W_\mathcal{G})||_\square \\ &\leq (1-\lambda)\mathrm{e}(F_\mathcal{G})||W_\mathcal{H} - W_\mathcal{G}||_\square \end{aligned} \tag{17}$$

Similarly, let $W = W_\mathcal{I}$, $W' = W_\mathcal{H}$ and $F = F_\mathcal{H}$ in Lemma 1, We can also easily obtain

$$|t(F_\mathcal{H}, \lambda W_\mathcal{G} + (1-\lambda)W_\mathcal{H}) - t(F_\mathcal{H}, W_\mathcal{H})| \leq \lambda\mathrm{e}(F_\mathcal{H})||W_\mathcal{H} - W_\mathcal{G}||_\square \tag{18}$$

Equation 17 and 18 produce the upper bound in Theorem 8. ■

A.3 Proof of Theorem 2

Let F and W be the discriminative motif F_G and the mixed graphon $W_\mathcal{I}$ in Lemma 2, we will have

$$\mathrm{P}\left(|t(F_\mathcal{I}, \mathbb{G}) - t(F_\mathcal{I}, W_\mathcal{I})| > \varepsilon\right) \leq 2\exp\left(-\frac{\varepsilon^2 n}{8\mathrm{v}(F_\mathcal{I})^2}\right) \tag{19}$$

which produces the result in 9. ■

A.4 Graphons Estimation by step functions

The proof follows Xu et al. (2021). A graphon can always be approximated by a step function in the cut norm (Frieze & Kannan, 1999).

Let $\mathcal{P} = (\mathcal{P}_1, .., \mathcal{P}_K)$ be a partition of Ω into K measurable sets. We define a step function $W_\mathcal{P} : \Omega^2 \mapsto [0,1]$ as

$$W_\mathcal{P}(x,y) = \sum\nolimits_{k,k'=1}^{K} w_{kk'} 1_{\mathcal{P}_k \times \mathcal{P}_{k'}}(x,y), \tag{20}$$

where each $w_{kk'} \in [0,1]$ and the indicator function $1_{\mathcal{P}_k \times \mathcal{P}_{k'}}(x,y)$ is 1 if $(x,y) \in \mathcal{P}_k \times \mathcal{P}_{k'}$, otherwise it is 0. The weak regularity lemma Lovász (2012) shown below guarantees that every graphon can be approximated well in the cut norm by step functions.

Theorem 3 (Weak Regularity Lemma (Lemma 9.9 in (Lovász, 2012))) *For every graphon W and $K \geq 1$, there always exists a step function $\mathbf{W}$ with $|\mathcal{P}| = K$ steps such that*

$$\|W - \mathbf{W}\|_\square \leq \frac{2}{\sqrt{\log K}} \|W\|_{L_2}. \tag{21}$$

B Experiments Setting

B.1 Experimental Setting

To ensure a fair comparison, we use the same hyperparater for modeling training and the same architecture for vanilla model and other baselines. For model training, we use the Adam optimizer(Kingma & Ba, 2015). The initial learning rate is 0.01 and will drop the learning rate by half every 100 epochs. The batch size is set to 128. We split the dataset into train/val/test data by $7:1:2$. The best epoch are determined by the best validation accuracy. Note that best test epoch is selected on a validation set. We also report the test accuracy on ten runs.

For architecture of graph neural networks, the details are listed as follows,

- **GCN** (Kipf & Welling, 2016). Four GNN layers and global mean pooling are applied. All the hidden units is set to 64. The activation is ReLU (Nair & Hinton, 2010).

- **TopKPool** (Gao & Ji, 2019). Three GNN layers and three TopK pooling are applied. A there-layer percetron are adopted to predict the labels. All the hidden units is set to 64. The activation is ReLU (Nair & Hinton, 2010).
- **GIN** (Xu et al., 2018). We apply five GNN layers and all MLPs have two layers. Batch normalization (Ioffe & Szegedy, 2015) is applied on every hidden layer. All hidden units are set to 64. The activation is ReLU (Nair & Hinton, 2010).

For hyperparemeter in $\mathcal{G}$-Mixup, we generate 20% more graph for training graph. The graphons are estimated based on the training graphs. We use different $\lambda \in [0.1, 0.2]$ to mix up the graphon and generate synthetic with different strength of mixing up.

B.2 Experimental Setting of Robustness

The graph neural network adopted in this experiment is GCN, the architecture of which is as above. For label corruption, we randomly corrupt the graph labels with different corruption ratio $10\%, 20\%, 30\%, 40\%$. For topology corruption, we we randomly remove/add edges with different corruption ratio $10\%, 20\%, 30\%, 40\%$. The dataset for topology corruption is REDDIT-BINARY.

B.3 visualization of graphons on more real-world dataset

$\mathcal{G}$-Mixup explores five graphon estimation methods, including sorting-and-smoothing (SAS) method (Chan & Airoldi, 2014), stochastic block approximation (SBA) (Airoldi et al., 2013), "largest gap" (LG) (Channarond et al., 2012), matrix completion (MC) (Keshavan et al., 2010) and the universal singular value thresholding (USVT) (Chatterjee et al., 2015). We present the estimated graphon by LG in Figure 2. Here we present more visualization of graphons on IMDB-BINARY, REDDIT-BINARY and IMDB-MULTI dataset. An obvious observation is that graphons of different classes of graphs are different. This observation further validates the divergence of graphon between different classes of graphs.

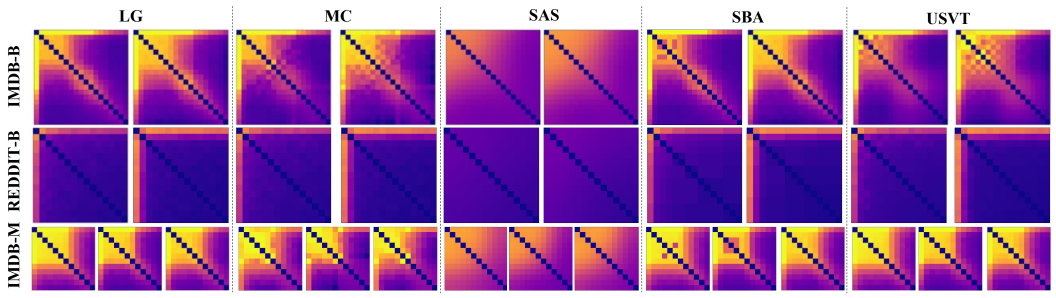

Figure 5: The estimated graphon on various dataset with different graphon estimation methods.

C Additional Experiments For Rebuttal

In this appendix, we conduct additional experiments to further investigate the proposed method. The additional experiments include 1) more graph neural networks ((DiffPool, MincutPool, GMT)) in Appendix C.1, 2) molecular property prediction task with OGB datasets in Appendix C.2, 3) experiments on the impact of the nodes number of generated graphs in Appendix C.3 and 4) experiments on the performance of GCN with different layers in Appendix C.4.

C.1 Experiment on More Graph Neural Networks (DiffPool, MincutPool, GMT)

To further validate the effectiveness of $\mathcal{G}$-Mixup on more graph neural networks, we experiment with DiffPool (Ying et al., 2018), MincutPool (Bianchi et al., 2020) and GMT (Baek et al., 2020). For GMT, we use their released code and the recommended hyperparameters for their used datasets (D&D, MUTAG, PROTEINS, IMDB-B, IMDB-M) in their paper. To reproduce its results, we use their official code and the above datasets. The results are presented in Tables 5 and 6.

The details of backbones are listed as follows:

- **DiffPool (Ying et al., 2018)** is a differentiable graph pooling methods that can be adapted to various GNN architectures, which maps nodes to clusters based on their learned embeddings.
- **MincutPool (Bianchi et al., 2020)** is a differentiable pooling baselines. It learns a clustering function that can be quickly evaluated on out-of-sample graphs.
- **GMT (Baek et al., 2020)** is a multi-head attention based global pooling layer to generate graph representation, which captures the interaction between nodes according to their structure.

Our main observations are: **Observation 7: $\mathcal{G}$-Mixup improves the performance of DiffPool and MincutPool on various datasets.** From Table 5, our proposal gains 7 best performances among 8 reported accuracies, which substantially improve the performance of DiffPool and MincutPool. **Observation 8: $\mathcal{G}$-Mixup can significantly improve the performance of GMT.** Table 6 shows that $\mathcal{G}$-Mixup outperform all the baselines on all datasets. Overall, $\mathcal{G}$-Mixup outperform vanilla, Dropedge, ManifoldMixup by 1.44%, 1.28%, 2.01%, respectively. This indicates the superiority of $\mathcal{G}$-Mixup for graph classification task.

Table 5: Performance comparisons of $\mathcal{G}$-Mixup with DiffPool and MincutPool on different datasets. The metric is classification accuracy and its standard deviation. The best performance is in **boldface**.

Backbone	Method	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M5k
DiffPool	vanilla	71.68$_{\pm 3.40}$	47.75$_{\pm 2.34}$	78.40$_{\pm 4.38}$	31.61$_{\pm 5.95}$
	w/ Dropedge	69.16$_{\pm 2.51}$	49.44$_{\pm 2.50}$	76.00$_{\pm 5.50}$	34.46$_{\pm 6.80}$
	w/ NodeDropping	70.25$_{\pm 3.01}$	46.83$_{\pm 1.34}$	76.68$_{\pm 2.57}$	33.10$_{\pm 5.53}$
	w/ Subgraph	69.50$_{\pm 2.16}$	46.00$_{\pm 4.43}$	76.06$_{\pm 2.81}$	31.65$_{\pm 4.43}$
	w/ ManfoldMixup	66.50$_{\pm 4.04}$	45.16$_{\pm 4.63}$	78.37$_{\pm 2.29}$	34.46$_{\pm 6.80}$
	w/ $\mathcal{G}$-Mixup	**73.25**$_{\pm 3.89}$	**50.70**$_{\pm 2.79}$	**78.87**$_{\pm 2.27}$	**38.42**$_{\pm 6.51}$
MincutPool	vanilla	73.25$_{\pm 3.27}$	49.04$_{\pm 3.57}$	84.95$_{\pm 3.25}$	49.32$_{\pm 2.67}$
	w/ Dropedge	69.16$_{\pm 2.51}$	49.66$_{\pm 1.73}$	81.37$_{\pm 1.59}$	47.20$_{\pm 1.10}$
	w/ NodeDropping	73.50$_{\pm 3.89}$	49.91$_{\pm 2.83}$	85.68$_{\pm 2.04}$	46.82$_{\pm 4.60}$
	w/ Subgraph	70.25$_{\pm 1.84}$	48.18$_{\pm 1.10}$	84.91$_{\pm 2.50}$	49.22$_{\pm 2.49}$
	w/ ManfoldMixup	70.62$_{\pm 2.09}$	49.96$_{\pm 1.86}$	85.12$_{\pm 2.29}$	47.20$_{\pm 1.10}$
	w/ $\mathcal{G}$-Mixup	**73.93**$_{\pm 2.84}$	**50.29**$_{\pm 2.30}$	**85.87**$_{\pm 1.37}$	**50.12**$_{\pm 2.47}$

Table 6: Performance comparisons of $\mathcal{G}$-Mixup with GMT on different dataset. The metric is the classification accuracy and its standard deviation. The best performance is in **boldface**.

Backbone	Method	D&D	MUTAG	PROTEINS	IMDB-B	IMDB-M
GMT	vanilla	78.29$_{\pm 5.77}$	82.77$_{\pm 6.30}$	74.59$_{\pm 5.29}$	73.60$_{\pm 3.87}$	50.73$_{\pm 3.03}$
	w/ Dropedge	78.37$_{\pm 4.17}$	82.22$_{\pm 8.88}$	74.32$_{\pm 5.42}$	73.40$_{\pm 3.85}$	50.73$_{\pm 3.09}$
	w/ ManfoldMixup	77.69$_{\pm 3.81}$	82.22$_{\pm 10.48}$	74.41$_{\pm 3.97}$	73.70$_{\pm 3.79}$	49.93$_{\pm 3.49}$
	w/ $\mathcal{G}$-Mixup	**79.57**$_{\pm 3.69}$	**84.44**$_{\pm 8.88}$	**75.13**$_{\pm 5.06}$	**74.70**$_{\pm 3.76}$	**51.33**$_{\pm 3.52}$

C.2 Experiment on molecular property prediction

We experiment on molecular property prediction task (Hu et al., 2020), including ogbg-molhiv, ogbg-molbace, ogbg-molbbbp. In these dataset, each graph represents a molecule, where nodes are atoms, and edges are chemical bonds. We adopte official reference graph neural network backbones (gcn, gcn-vitual, gin, gin-vitual) as our backbones, and we generate the edge attributes randomly for synthetic graphs. The results are presented in Table 7. **Observation 9: $\mathcal{G}$-Mixup can improve the performance of GNNs on molecular property prediction task with the experimental setting for a fair comparison.** Table 7 shows that $\mathcal{G}$-Mixup gains 9 best performances among 12 reported AUCs.

Table 7: Performance comparisons of $\mathcal{G}$-Mixup on molecular property prediction task. The metric is AUROC [3] and its standard deviation. The best performance is in **boldface**.

Backbones	Mehtods	ogbg-molhiv	ogbg-molbbbp	ogbg-molbace
GCN	vanilla	$76.24_{\pm 0.98}$	$68.05_{\pm 1.52}$	$80.36_{\pm 1.56}$
	w/ Dropedge	$75.93_{\pm 0.76}$	$68.02_{\pm 0.95}$	$80.22_{\pm 1.59}$
	w/ ManifoldMixup	$76.24_{\pm 1.40}$	$68.36_{\pm 2.05}$	$80.46_{\pm 2.05}$
	w/ G-Mixup	$\mathbf{76.29}_{\pm 0.80}$	$\mathbf{68.45}_{\pm 0.84}$	$\mathbf{80.73}_{\pm 2.06}$
GCN-virtual	vanilla	$75.62_{\pm 1.65}$	$65.13_{\pm 1.11}$	$\mathbf{74.49}_{\pm 3.04}$
	w/ Dropedge	$74.64_{\pm 1.32}$	$66.46_{\pm 1.61}$	$69.75_{\pm 3.47}$
	w/ ManifoldMixup	$74.04_{\pm 2.06}$	$65.51_{\pm 1.74}$	$73.10_{\pm 4.97}$
	w/ G-Mixup	$\mathbf{76.56}_{\pm 0.80}$	$\mathbf{67.20}_{\pm 1.30}$	$73.55_{\pm 4.79}$
GIN	vanilla	$77.08_{\pm 1.96}$	$68.42_{\pm 2.31}$	$75.91_{\pm 1.01}$
	w/ Dropedge	$75.77_{\pm 1.75}$	$66.16_{\pm 2.96}$	$70.50_{\pm 6.24}$
	w/ ManifoldMixup	$75.73_{\pm 1.25}$	$68.15_{\pm 2.04}$	$77.44_{\pm 4.13}$
	w/ G-Mixup	$\mathbf{77.14}_{\pm 0.45}$	$\mathbf{69.28}_{\pm 1.24}$	$\mathbf{77.79}_{\pm 3.34}$
GIN-virtual	vanilla	$\mathbf{77.52}_{\pm 1.56}$	$67.10_{\pm 2.10}$	$\mathbf{74.19}_{\pm 4.99}$
	w/ Dropedge	$76.83_{\pm 1.11}$	$68.87_{\pm 1.17}$	$72.20_{\pm 3.37}$
	w/ ManifoldMixup	$76.51_{\pm 2.22}$	$68.04_{\pm 2.87}$	$74.17_{\pm 1.38}$
	w/ G-Mixup	$77.09_{\pm 1.07}$	$\mathbf{70.02}_{\pm 1.68}$	$73.53_{\pm 3.98}$

C.3 What is the impact of number of nodes of generated graphs?

We investigate the impact of number of nodes of generated synthetic graphs by $\mathcal{G}$ -Mixup and report the results in Figure 6. Specifically, $\mathcal{G}$ -Mixup generates synthetic graphs with different the numbers (hyperparameters K) of nodes and use them to train graph neural networks. From Figure 6, we can see that average number of nodes in the original graphs is a better choice for hyperparameters K for $\mathcal{G}$ -Mixup, which is accords with intuition.

C.4 How $\mathcal{G}$ -Mixup Perform when graph neural network goes deeper?

We investigate the performance of $\mathcal{G}$-Mixup when the GCN goes deeper and report the results in Figure 7. We experimented with different numbers $(2-9)$ of layers to investigate the performance of $\mathcal{G}$ -Mixup. **Observation 10: $\mathcal{G}$-Mixup improves the performance of graph neural networks with different layers**. The left figure in Figure 7 shows $\mathcal{G}$ -Mixup gains better performance for IMDB-BINARY dataset while the depth of GCNs is $2-6$. The performance with deeper GCN $(7-9)$ are comparable to baselines, however, the accuracy of deeper GCN is much lower than shallow ones. The right figure in Figure 7 shows $\mathcal{G}$ -Mixup gains better performance by a significant margin for REDDIT-BINARY dataset while the depth of GCNs is $2-9$. This validates the effectiveness of $\mathcal{G}$-Mixup when graph neural network goes deeper.

[3] Area Under Receiver Operating Characteristic

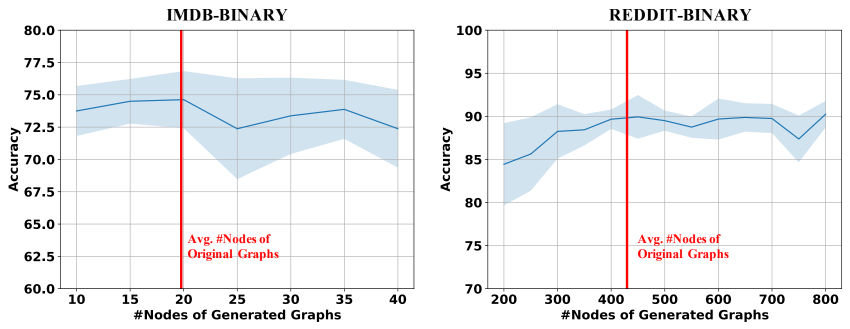

Figure 6: The impact of the node numbers of generated synthetic graphs on IMDB-BINARY and REDDIT-BINARY datasets. The red vertical line indicates the average number of all the original graphs. The blue line represents that classification accuracy with different number of nodes of generated graphs. Obviously, the accuracy reach the maximum values around the red line on both two datasets

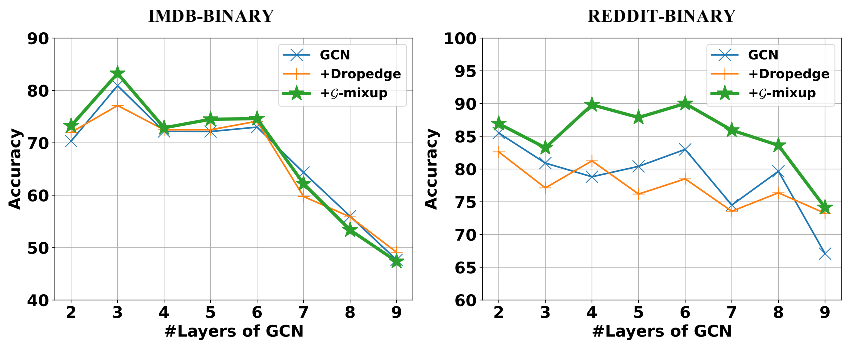

Figure 7: The performance of $\mathcal{G}$-mixup using GCNs with different depth on IMDB-BINARY and REDDIT-BINARY. This figure show that $\mathcal{G}$-Mixup consistently improve GCN when it goes deeper.