G-Mixup: Graph Data Augmentation for Graph Classification

2022 ICML Outstanding Paper

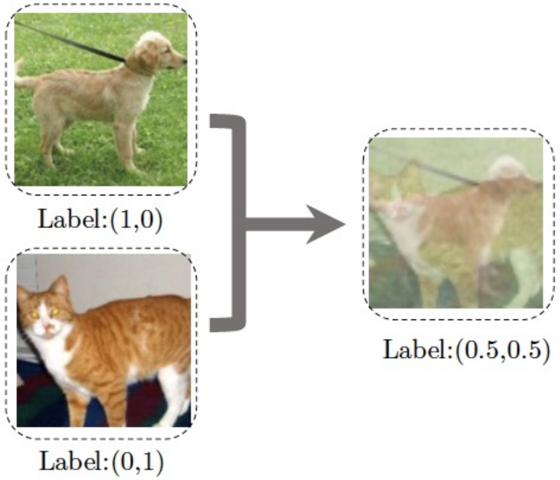
Yuting Hu

Content

Background and Motivation

Mixup

Mixup is a cross-instance data augmentation method, which linearly interpolates random sample pair to generate more synthetic training data.



$$\mathbf{x}_{new} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j,$$
$$\mathbf{y}_{new} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j,$$

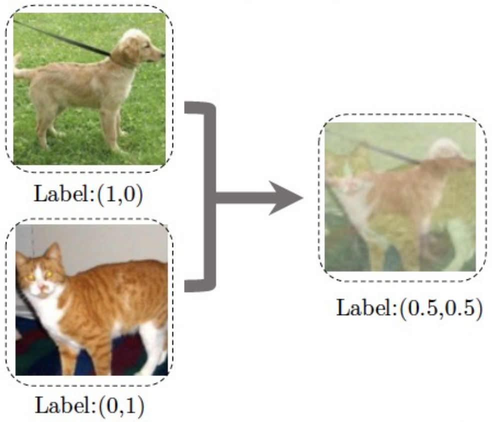
where $(\mathbf{x}_i, \mathbf{y}_i)$, $(\mathbf{x}_j, \mathbf{y}_j)$ are two samples randomly drawn from training data.

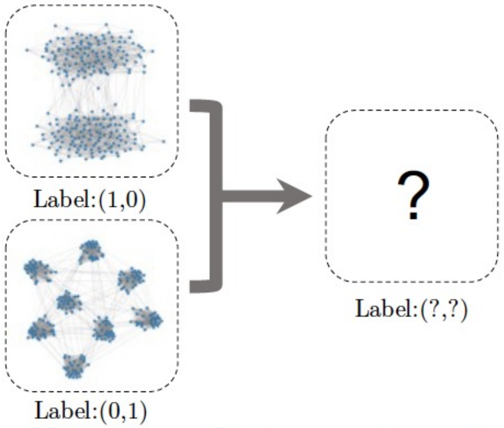
Mixup have been empirically and theoretically shown to improve the generalization and robustness of deep neural networks (H. Zhang et al., 2017; L. Zhang et al., 2021).

Can we mix up input graph pair to improve graph neural networks?

Challenges for Graph Mixup

Graph data is different from image data:





1. Image data is regular (image can be represented as matrix)
2. Image data is well-aligned (pixel to pixel correspondence)
3. Image data is grid-like data

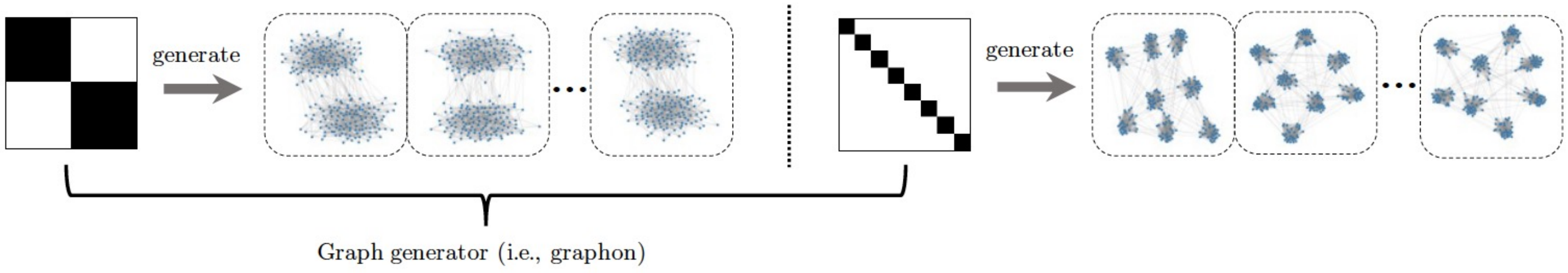
- Image is in Euclidean space

1. Graph data is irregular (the number of nodes)
2. Graph data is not well-aligned (nodes not naturally ordered)
3. Graph has divergent topology information

- Graph is in non-Euclidean space

Graph Generator: Graphon

The real-world graphs can be regarded as generated from generator (i.e., graphon[1]). For example,



The graphons of different graphs are **regular**, **well-aligned**, and **in Euclidean space**.

We propose to mix up graph generator (i.e., graphon) to achieve the input graph mixup.

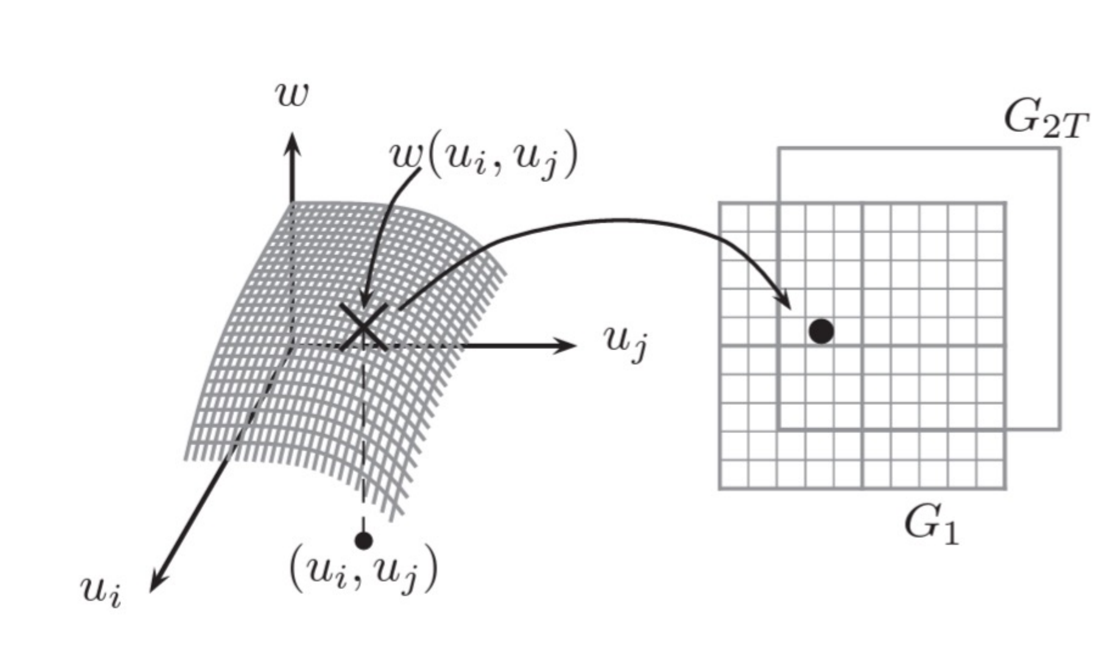
Methodology

Graphon Concept

Concept: theory predicts that every convergent sequence of graphs $\{G_n\}$ has a limit object that preserves many local and global properties of the graphs in the sequence. This limit object, which is called a graphon, can be represented by measurable functions $w:[0,1]^2 \to [0,1]$, in a way that any w' obtained from measure preserving transformations of w describes the same graphon.

Graphons are usually seen as kernel functions for random network models (Lawrence 2005). To construct an n-vertex random graph $\mathcal{G}(n, w)$ for a given w, we first assign a random label $u_i \sim \text{Uniform}[0,1]$ to each vertex $i \in \{1, \ldots, n\}$, and connect any two vertices i and j with probability $w(u_i, u_j)$, *i.e.*,

$$\Pr\left(G[i,j] = 1 \mid u_i, u_j\right) = w(u_i, u_j), \qquad i, j = 1, \ldots, n, \tag{1}$$



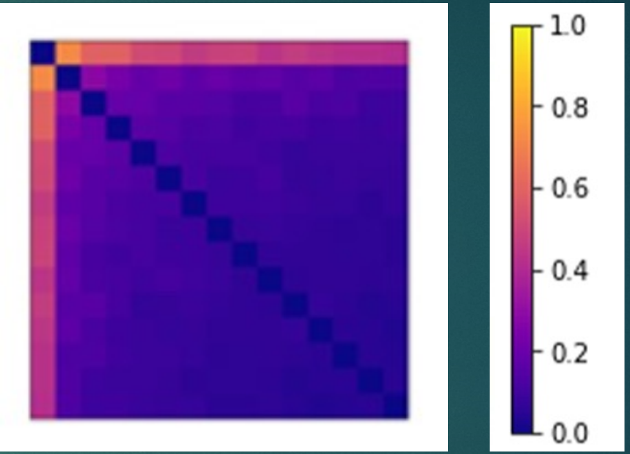
Reference: Stochastic block model approximation of a graphon: Theory and consistent estimation
https://arxiv.org/pdf/1311.1731.pdf

Graphon Estimation

This paper uses **step function** to approximate graphon.

[0, 1] / k slots

a step function $\mathbf{W}^P : [0,1]^2 \mapsto [0,1]$ is defined 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.



Algorithm 1 Graphon Estimation

Input: graph set $\mathcal{G}$, graphon estimator g ▷ each graph G has adjacency matrix $\mathbf{A}$ and node features matrix $\mathbf{X}$
Init: sorted adjacency matrix set $\bar{A} = \{\}$
for each graph G in $\mathcal{G}$ **do**
  Calculate the degree of each nodes in G
  Calculate sorted adjacency matrix $\bar{\mathbf{A}}$ by sorting $\mathbf{A}$ based on the degree
  Calculate sorted node features matrix $\bar{\mathbf{X}}$ by sorting $\mathbf{X}$ based on the degree
  Add the sorted adjacency matrix $\bar{\mathbf{A}}$ to $\bar{A}$
end for
Estimate step function $\mathbf{W}_{\mathcal{G}}$ with $\bar{A}$ using g. ▷ we use LG as g in experiments
Obtain graphon node feature $\bar{\mathbf{X}}_{\mathcal{G}}$ by average pooling $\mathbf{X}$ ▷ we can use other pooling method (e.g., maxpooling)
Return: $\mathbf{W}_{\mathcal{G}}$, $\bar{\mathbf{X}}_{\mathcal{G}}$

Synthetic Graphs Generation

A graphon can provide arbitrarily sized graphs

$$u_1, \ldots, u_K \overset{\text{iid}}{\sim} \text{Unif}_{[0,1]}, \mathbb{G}(K, W)_{ij} \overset{\text{iid}}{\sim} \text{Bern}(W(u_i, u_j)),$$
$$\forall i, j \in [K].$$

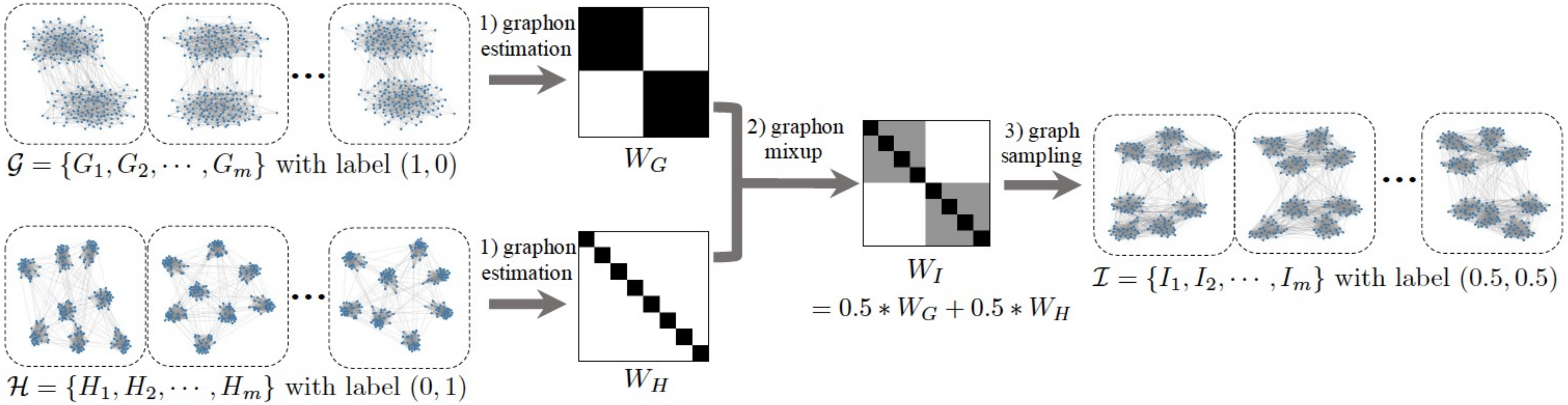
1. Sample k nodes independently from a uniform distribution on [0,1]
2. Generate adjacency matrix with each element is a 0-1 dist with step function.

How about the node features?

This paper adopts average pooling of aligned node features (sort node by degree) to define node features for syntetic graphs.

G-Mixup

We propose to mixup the generator (i.e., graphon) of graphs, mix up the graphons of different classes, and then generate synthetic graphs.



The formal mathematical expression are as follows:

(1) Graphon Estimation: $\mathcal{G} \rightarrow W_{\mathcal{G}}, \mathcal{H} \rightarrow W_{\mathcal{H}}$

(2) Graphon Mixup: $W_{\mathcal{I}} = \lambda W_{\mathcal{G}} + (1-\lambda) W_{\mathcal{H}}$

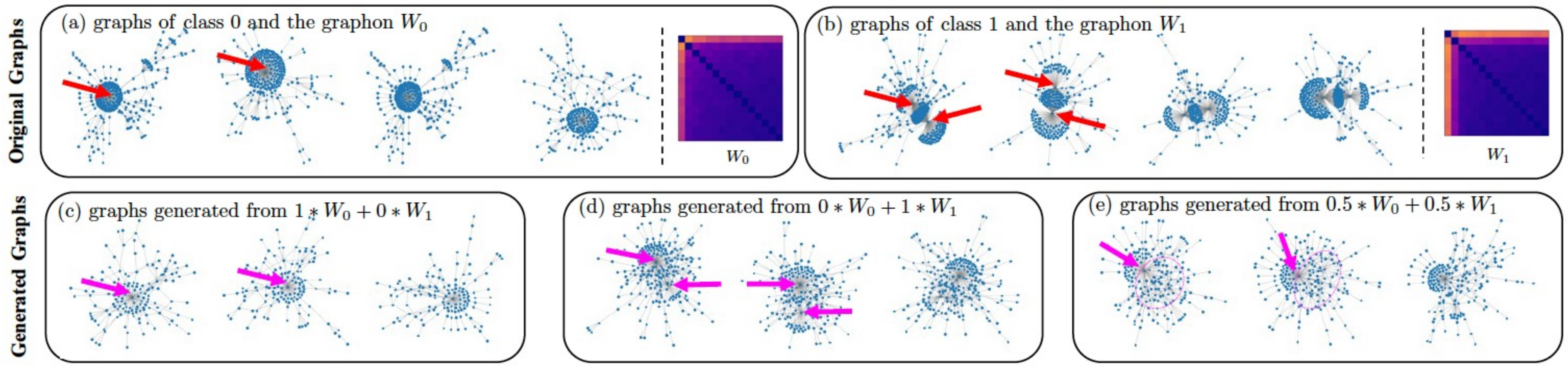
(3) Graph Generation: $\{I_1, I_2, \cdots, I_m\} \overset{\text{i.i.d}}{\sim} \mathbb{G}(K, W_{\mathcal{I}})$

(4) Label Mixup: $\mathbf{y}_{\mathcal{I}} = \lambda \mathbf{y}_{\mathcal{G}} + (1-\lambda) \mathbf{y}_{\mathcal{H}}$

Results

One Case

We visualize the generated synthetic graphs on REDDIT-BINARY dataset.

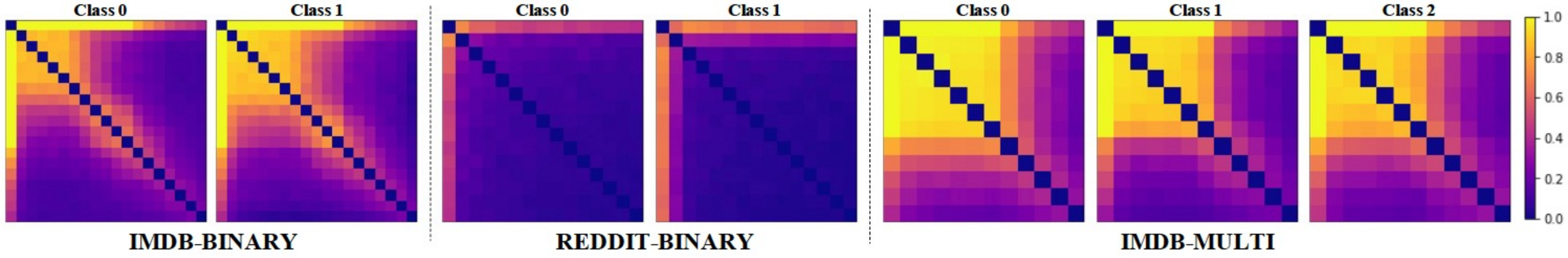


We make the following observations:

1. The class 0 has one high-degree node while class 1 have two (a)(b).
2. The generated graphs based on
   - $(1 * W_0 + 0 * W_1)$ have one high-degree node (c).
   - $(0 * W_0 + 1 * W_1)$ have two high-degree nodes (d).
   - $(0.5 * W_0 + 0.5 * W_1)$ have a high-degree node and a dense subgraph (e).
3. Graphs generated by $\mathcal{G}$-Mixup are the mixture of original graphs.

Validation

We visualize the estimated graphons on IMDB-BINARY, REDDIT-BINARY, and IMDB-MULTI.

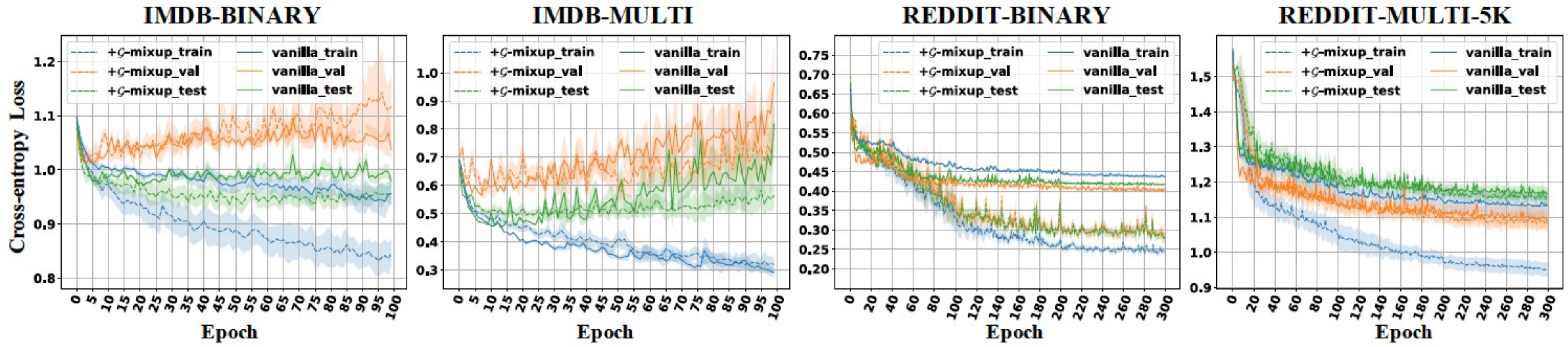


We make the following observations:

1. Real-world graphs of different classes have different graphons.
2. This observation lays a solid foundation for our proposed method.

Performance

We present the training/validation/test curves on IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY and REDDIT-MULTI-5K with GCN.



We make the following observations:

1. The loss curves of $\mathcal{G}$-Mixup are lower than the vanilla model.
2. $\mathcal{G}$-Mixup can improve the generalization of graph neural networks.

Performance

We use different GNNs for graph classification and report the performance comparisons of $\mathcal{G}$-Mixup.

	Dataset	IMDB-B	IMDB-M	REDD-B	REDD-M5	REDD-M12
	#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	48.79	78.82	45.07	46.90
	w/ Dropedge	72.50	49.08	81.25	51.35	47.08
	w/ DropNode	72.00	48.58	79.25	49.35	47.93
	w/ Subgraph	68.50	49.58	74.33	48.70	47.49
	w/ M-Mixup	72.83	49.50	75.75	49.82	46.92
	w/ $\mathcal{G}$-Mixup	**72.87**	**51.30**	**89.81**	**51.51**	**48.06**
GIN	vanilla	71.55	48.83	92.59	55.19	50.23
	w/ Dropedge	**72.20**	48.83	92.00	55.10	49.77
	w/ DropNode	72.16	48.33	90.25	53.26	49.95
	w/ Subgraph	68.50	47.25	90.33	54.60	49.67
	w/ M-Mixup	70.83	49.88	90.75	54.95	49.81
	w/ $\mathcal{G}$-Mixup	71.94	**50.46**	**92.90**	**55.49**	**50.50**

	Method	IMDB-B	IMDB-M	REDD-B	REDD-M5k
TopKPool	vanilla	72.37	50.57	90.30	45.07
	w/ Dropedge	71.75	48.75	88.96	**47.43**
	w/ DropNode	69.16	48.50	81.33	46.15
	w/ Subgraph	67.83	50.83	86.08	45.75
	w/ M-Mixup	71.83	51.22	87.58	45.60
	w/ $\mathcal{G}$-Mixup	**72.80**	**51.30**	**90.40**	46.48
DiffPool	vanilla	71.68	47.75	78.40	31.61
	w/ Dropedge	69.16	49.44	76.00	34.46
	w/ DropNode	70.25	46.83	76.68	33.10
	w/ Subgraph	69.50	46.00	76.06	31.65
	w/ M-Mixup	66.50	45.16	78.37	34.46
	w/ $\mathcal{G}$-Mixup	**73.25**	**50.70**	**78.87**	**38.42**
MincutPool	vanilla	73.25	49.04	84.95	49.32
	w/ Dropedge	69.16	49.66	81.37	47.20
	w/ DropNode	73.50	49.91	85.68	46.82
	w/ Subgraph	70.25	48.18	84.91	49.22
	w/ M-Mixup	70.62	49.96	85.12	47.20
	w/ $\mathcal{G}$-Mixup	**73.93**	**50.29**	**85.87**	**50.12**

We make the following observation:

1. $\mathcal{G}$-Mixup can improve the performance of GNNs on various datasets.