RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[¬Re] G-Mixup: Graph Data Augmentation for Graph Classification

Dylan Cordaro[1], Shelby Cox[1], Yiman Ren[1], and Teresa Yu[1]

[1]University of Michigan, Ann Arbor, Michigan, USA

Edited by
Koustuv Sinha,
Maurits Bleeker,
Samarth Bhargav

Received
04 February 2023

Published
20 July 2023

DOI
10.5281/zenodo.8173737

Reproducibility Summary

Scope of Reproducibility — We study the graph data mixup method $\mathcal{G}$-Mixup developed by Han, Jiang, Liu, and Hu [1]. We investigate their claims that $\mathcal{G}$-Mixup theoretically produces synthetic graphs that contain a mixture of key graph topologies of source graphs, experimentally improves the performance of graph neural networks (GNNs), and experimentally performs better than another graph data augmentation methods.

Methodology — For the theoretic claims, we give more detailed proofs of the authors' theorems by using results from [2]. For the experimental claims, we utilize the authors' publicly available code and our own implementation of the GCN neural network. We run experiments on the datasets IMDB-B, REDDIT-B, and PROTEINS. For the IMDB-B and PROTEINS datasets, we use Google Colab, which provides a NVIDIA P100 and Tesla T4 GPU and up to 32GB of RAM. Each experiments took approximately 10 GPU minutes. For the REDDIT-B dataset, we used the campus computing cluster, which gives a Tesla V100 GPU with up to 90 GB RAM. It took about 20 minutes each of these experiments.

Results — We verify their theoretical claims that $\mathcal{G}$-Mixup indeed leads to a mixture of "key graph topologies." We reproduce their experimental results on classification accuracy for REDDIT-B to within about 6% of the reported value, and for IMDB-B to within about 2% of the reported value. However, our experimental results do not provide statistically significant evidence to support the paper's experimental claims that $\mathcal{G}$-Mixup improves the performance of GNNs or performs better than other graph data augmentation methods.

What was easy — The authors provided sufficient mathematical background and useful citations to fill in the gaps in their proofs. The authors' code was fairly easy to run with the scripts provided, once we installed the correct packages. It was easy to run the code on other datasets from PyTorch Geometric.

What was difficult — Setting up the code environment was difficult. We also initially faced memory limits when generating the graphons for the REDDIT-B dataset.

Communication with original authors — We have communicated with the corresponding author over email about our memory usage and neural network architecture.

Correspondence should be addressed to Shelby Cox (spcox@umich.edu)
The authors have declared that no competing interests exist.
Code is available at https://github.com/shelbycox/EECS-553-GMixup-Reproduction. – SWH swh:1:dir:9c61eef8a6a063061f952ef48145dd9389d45281.
Open peer review is available at https://openreview.net/forum?id=T54wy0ahGLG&noteId=pvukxfvJGX.

1 Introduction

This paper [1] develops $\mathcal{G}$-Mixup, a data augmentation technique that involves mixing up graph data using the theory of graphons. Past work on mixup and data augmentation has mostly focused on Euclidean data [3] or within-graph data augmentation [4, 5]. This new method proposed by the authors allows for the augmentation of graph data across different classes of graphs. The authors of [1] claim to provide theoretical and experimental evidence that the augmented data generated by $\mathcal{G}$-Mixup can improve the generalization and robustness of graph neural networks (GNNs), compared to both the classical setting (no data augmentation) and the setting in which other data augmentation techniques are used.

2 Scope of reproducibility

In [1], the authors propose a new method for augmenting graph data, $\mathcal{G}$-Mixup, which they claim theoretically generates synthetic graphs that are mixtures of original graphs and improves the generalization and robustness of Graph Neural Networks (GNNs). In this report, we investigate the following two claims from [1]:

- Claim 1 (*Theoretical mixing*): Theoretically, $\mathcal{G}$-Mixup produces synthetic graphs that have a mixture of the key graph topologies of source graphs coming from distinct classes.
- Claim 2 (*Improved experimental performance of GNNs*): Augmenting graph data using $\mathcal{G}$-Mixup increases the test accuracy and decreases the test cross-entropy loss of a GNNs when compared to (a) a GNN without augmentation, and (b) a GNN with a different graph augmentation algorithm.

In the original paper, **Claim 1** is supported by [1, Theorem 4.2], [1, Theorem 4.3], and proofs in [1, Appendix A]. In Sections 4.1.1 and A.2, we provide a more detailed proof of one of the authors' key lemmas, as well as a corrected statement of the second theorem.

Claim 2 is supported by the experiments in [1, Section 5.3], whose results are described in [1, Table 2], and [1, Figure 4]. We reproduce some of these experiments for two of the authors original datasets, IMDB-B and REDDIT-B, and a new dataset which is not used by the authors, PROTEINS, in Section 4.1.2. We also compare $\mathcal{G}$-Mixup with another graph augmentation algorithm (graphon-based edge perturbation) for IMDB-B, REDDIT-B, PROTEINS.

3 Methodology

The authors provide their code publicly on Github.[1] We reused the author's code from Github in a Google Colab notebook with a GPU. We implemented the GCN architecture, added a hyperparameter to modify graphon resolution, and added code to log the neural network statistics by epoch. Otherwise, we left the source code unchanged.

3.1 Model descriptions

The authors propose a new data augmentation method, $\mathcal{G}$-Mixup, for irregular, not-well-aligned graph data with divergent topology between graph classes, to which existing mixup methods are not directly applicable. The $\mathcal{G}$-Mixup theory is based on the **graphon**, which is a continuous, bounded, and symmetric function from $[0, 1]^2$ to $[0, 1]$ and can

[1] The code from their Github is linked here: https://github.com/ahxt/g-mixup.

be thought of as the weight matrix of a graph with infinite number of vertices.

Broadly, the $\mathcal{G}$-Mixup model proposed in the paper has the following key steps: i) estimate a graphon, $\mathcal{W}_{\mathcal{G}}$, for each class of graphs $\mathcal{G}$; ii) mix up the graphons of different graph classes $\mathcal{G}$ and $\mathcal{H}$, as in (1); and iii) generate synthetic graphs based on the mixed graphons and mix up the labels. In (1), λ is called the **mixup ratio/parameter**. When $\lambda = 0$, $\mathcal{G}$-Mixup becomes the graph data augmentation method **graphon-based edge perturbation** [6].

$$W_{\mathcal{I}} = \lambda W_{\mathcal{G}} + (1 - \lambda) W_{\mathcal{H}} \tag{1}$$

To implement the graphon estimation step, the paper uses step functions to approximate graphon vertex features. The step function estimation methods are well-studied; they first align the vertices in a set of graphs based on degree and then estimate the step function from all the aligned adjacency matrices. The paper uses universal singular value thresholding (USVT) [7] as the estimation method in their code, and we also use this estimator in our experiments.

The authors then mix up the estimated graphons as follows. For n_{aug} cycles (this number is called the **augmentation number**), two graphons from the set of estimated graphons are randomly chosen. Then, synthetic graphs are generated from the mixed-up graphon. The number of synthetic graphs is controlled by the **augmentation ratio** α, and the number of nodes for the synthetic graphs is controlled partially by the **graphon resolution**. More details on these parameters are described in Section A.1.

After obtaining the synthetic graph data, the paper uses the Graph Convolutional Network (GCN) [8] and Graph Isomorphism Network (GIN) [9] neural networks for their experiments. See [1, AF.2] for the specific architectures of the GNNs used. For all of our experiments, we use the GCN model, as the paper's results using the GIN model do not provide statistically significant evidence to support their claims.

3.2 Datasets

We tested the authors' experimental claims on the datasets IMDB-B, REDDIT-B, and PROTEINS, which are part of the TUDataset collection[2] [10]. The IMDB-B and REDDIT-B datasets are used in the paper while the PROTEINS dataset is not used in the paper. The relevant statistics for the datasets are shown in Table 1. The edges of the graphs are unlabeled, and labels on vertices are ignored in our code.

Dataset	PROTEINS	REDD-B	IMDB-B
# graphs	1113	2000	1000
# classes	2	2	2
class priors	0.596/ 0.404	0.5/0.5	0.5/0.5
# avg. vertices	39.058	429.627	19.77
# avg. edges	72.816	497.794	96.53
avg. density	0.0477	0.0027	0.2469

Table 1. Statistics for the PROTEINS, REDDIT-B (REDD-B), and IMDB-B datasets.

We used a 7:1:2 train-validation-test split, which is the same split used by the authors. The original authors' code does all the preprocessing necessary for the datasets through the PyTorch Geometric library.

[2]The datasets can be downloaded from the TUDataset repository, linked here: https://chrsmrrs.github.io/datasets/docs/datasets/.

3.3 Hyperparameters

In this subsection, we state the values used for hyperparameters related to $\mathcal{G}$-Mixup. We describe these hyperparameters further in Section A.1. We set the augmentation ratio to be $\alpha = 0.2$, mixup ratio/interval to be $\lambda \in [\Lambda_1, \Lambda_2] = [0.1, 0.2]$, augmentation number to be $n_{\text{aug}} = 10$, and the graphon resolution to be the median number of nodes in the training set. As an additional experiment, we also perform hyperparameter searches for the mixup ratio and graphon resolutions. The results are described in Section 4.2.2.

We now also state the hyperparameters for the GCN model architecture used by the original paper. We use the same values. There are 64 hidden features, the activation function is ReLU, the batch size is 128, the initial learning rate is 0.01 and drops by half every 100 epochs, and there are 4 layers.

3.4 Experimental setup and code

We used the authors' code on Github, and used Google Colab to perform our experiments. The authors only had the GIN neural network implemented in their code, so we implemented the GCN neural network per their specifications. For IMDB-B we train for 100 epochs, for PROTEINS we train for 300 epochs, and for REDDIT-B we train for 500 epochs.[3] The best test epoch is selected on a validation set with the loss function **mixup cross-entropy loss** defined below, and we report the test accuracy on 10 runs.

We use the above experimental setup to compare the baseline GCN model with the GCN model using $\mathcal{G}$-Mixup for the IMDB-B, REDDIT-B, and PROTEINS datasets (Sections 4.1.2 and 4.2.1). In addition, we use this setup to do hyperparameter searches for the mixup ratio and graphon resolution (Section 4.2.2); the search for the mixup ratio also gives a comparison of $\mathcal{G}$-Mixup with another graph data augmentation method.

As the label of a graphon is a probability distribution, we define the **mixup cross-entropy loss function** to be as in Equation 2, where p is our generated probability distribution and q is our target probability distribution.

$$-\sum_i q_i \log(p_i) \tag{2}$$

Note that this definition subsumes that of cross-entropy loss.[4] We also note that the original paper does not explicitly state how they define cross-entropy loss, but they use this definition in the code.

3.5 Computational requirements

For the PROTEINS and IMDB-B datasets, we use the free tier of Google Colab, which gives a Tesla T4 GPU with an average of 12 GB RAM. It takes around 11 minutes to run 10 seeds of 500 epochs of $\mathcal{G}$-Mixup on the GCN model.

For the REDDIT-B dataset, we used the campus computing cluster, which gives a Tesla V100 GPU with up to 90 GB RAM. It takes around 20 minutes to run 10 seeds with 500 epochs with or without $\mathcal{G}$-Mixup.

In terms of software, we use PyTorch Geometric 2.1.0 and PyTorch 1.12.1 on Python 3.7. As of date, these are the defaults of Google Colab.

[3] We determined the number of epochs by looking at output graphs for 1000 epochs, and cutting off training when the validation loss started to increase consistently.

[4] This definition was first defined here: https://github.com/moskomule/mixup.pytorch.

4 Results

Our main results are summarized as follows.

- We provide a more detailed proof of a key lemma of the paper and correct a constant in one of their theorems; these results study if the mixed-up graphon and the synthetic graphs sampled from the graphon preserve the "key graph topologies" of the original graphs.
- We find that $\mathcal{G}$-Mixup gives higher classification accuracy and lower test loss than the baseline GCN model for the REDDIT-B and PROTEINS datasets. For the IMDB-B dataset, we find that $\mathcal{G}$-Mixup gives lower test loss but not higher classification accuracy than the baseline GCN model. However, our results are not statistically significant. Our values for classification accuracy for the REDDIT-B and IMDB-B datasets are within about 6% and 2% respectively of the original paper's values.
- We find that varying the mixup ratio and graphon resolution hyperparameters does not greatly affect classification accuracy of $\mathcal{G}$-Mixup for the three datasets. Finally, we find that $\mathcal{G}$-Mixup does not outperform another graph data augmentation method (graphon-based edge perturbation) as measured by classification accuracy for the datasets.

4.1 Results reproducing original paper

Result 1 — We discuss the paper's theorems supporting Claim 1 of Section 2, which is that the synthetic graph generated by $\mathcal{G}$-Mixup is a mixture of original graphs. In particular, we give a more detailed proof of [1, Lemma A.2] and correct their statement of [1, Theorem 4.3]. The original paper uses the notion of **discriminative motifs** to capture key graph topologies. The relevant definitions are provided in Section A.2.

In Section A.2, we give a detailed proof of the following result, which is the key lemma used to prove Theorem 4.2. We note that the paper's original proof of this result is less than ten lines long.

Lemma 4.1 ([1] Lemma A.2)**.** Let F be a simple graph and let W, W' be two graphons. Let $e(F)$ denote the number of edges of F. Then,

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

In the paper, Lemma 4.1 is applied to prove the following main result. This result says that the difference in homomorphism densities of a given discriminative motif for graph class with respect to the graphon estimating that class and with respect to the mixed up graphon is upper-bounded by the cut norm between the graphons estimating the two distinct graph classes. This suggests that the mixed-up graphon indeed contains graph topologies of both classes of graphs that it is mixing up.

Theorem 4.2 ([1], Theorem 4.2)**.** Let $\mathcal{G}, \mathcal{H}$ be two sets of graphs with corresponding graphons $W_{\mathcal{G}}, W_{\mathcal{H}}$ and corresponding discriminative motif sets $\mathcal{F}_{\mathcal{G}}, \mathcal{F}_{\mathcal{H}}$. Let $\lambda \in (0, 1)$, and let $W_{\mathcal{I}} = \lambda W_{\mathcal{G}} + (1 - \lambda)W_{\mathcal{H}}$ be the mixed graphon. Then,

$$|t(F_{\mathcal{G}}, W_{\mathcal{I}}) - t(F_{\mathcal{G}}, W_{\mathcal{G}})| \leq (1 - \lambda)e(F_{\mathcal{G}})\|W_{\mathcal{H}} - W_{\mathcal{G}}\|_{\square},$$
$$|t(F_{\mathcal{H}}, W_{\mathcal{I}}) - t(F_{\mathcal{H}}, W_{\mathcal{H}})| \leq \lambda e(F_{\mathcal{H}})\|W_{\mathcal{H}} - W_{\mathcal{G}}\|_{\square}.$$

The authors then state [1, Theorem 4.3]. The purpose of this result is to show that the probability that the graph topology of a random graph sampled from a mixed-up graphon is very different from the graph topology of the mixed-up graphon can be made

arbitrarily small, as long as the number of vertices of the random graph is sufficiently large. However, the proof of this result given is incorrect, as it incorrectly cites the following result of [2].

Lemma 4.3 ([2], Theorem 2.5)**.** Let W be a graphon, let $n \geq 1$, and let $0 < \varepsilon < 1$. Let F be a simple graph. Then, the W-random graph $\mathbb{G} = \mathbb{G}(n, W)$ satisfies

$$\Pr(|t(F, \mathbb{G}) - t(F, W)| > \varepsilon) \leq 2 \exp\left(-\frac{\varepsilon^2 n}{18v(F)^2}\right).$$

The authors of [1] have an 8 instead of an 18 in the denominator of the fraction on the right hand side of the inequality. Therefore, the correct statement of [1, Theorem 4.3] is the following.

Theorem 4.4 ([1], corrected Theorem 4.3)**.** Let $W_{\mathcal{I}}$ be the mixed graphon, let $n \geq 1$, and let $0 < \varepsilon < 1$. Let $F_{\mathcal{I}}$ be the mixed discrminative motif. Then the $W_{\mathcal{I}}$-random graph $\mathbb{G} = \mathbb{G}(n, W_{\mathcal{I}})$ satisfies

$$\Pr(|t(F_{\mathcal{I}}, \mathbb{G}) - t(F_{\mathcal{I}}, W_{\mathcal{I}})| > \varepsilon) \leq 2 \exp\left(-\frac{\varepsilon^2 n}{18v(F_{\mathcal{I}})^2}\right).$$

Proof. We apply Lemma 4.3, with $F = F_{\mathcal{I}}$, $W = W_{\mathcal{I}}$. The corrected theorem statement then follows. □

Result 2 – We now provide our experimental results comparing $\mathcal{G}$-Mixup's performance on the GCN architecture with a vanilla GCN network on the IMDB-B and REDDIT-B datasets. This experiment is related to Claim 2(a) of Section 2, which is that augmenting graph data using $\mathcal{G}$-Mixup increases classification accuracy and decreases test loss of GNNs.

Table 2 shows the classification accuracy results for our experiments. The authors' original results for this experiment utilizing the entire REDDIT-B dataset is shown in Table 3. Our results for the REDDIT-B dataset are within about 6% of the authors' reported values, and our results for the IMDB-B dataset are within about 2% of the authors' reported values.

Method	PROTEINS	REDDIT-B	IMDB-B
vanilla	58.52 ± 3.11	81.4 ± 5.49	$\mathbf{73.15 \pm 2.5}$
w/ $\mathcal{G}$-Mixup	$\mathbf{64.66 \pm 5.06}$	$\mathbf{84.8 \pm 4.75}$	71.3 ± 3.3

Table 2. Our performance comparisons of $\mathcal{G}$-Mixup using the GCN architecture. The metric is classification accuracy.

Method	REDDIT-B	IMDB-B
vanilla	78.82 ± 1.33	72.18 ± 1.55
w/ $\mathcal{G}$-mixup	$\mathbf{89.81 \pm 1.70}$	$\mathbf{72.87 \pm 3.85}$

Table 3. Original performance comparisons of $\mathcal{G}$-Mixup using the GCN architecture. The metric is classification accuracy. This table is part of [1, Table 2].

Figure 2 in Section A.3 compares the loss for our experiment on the REDDIT-B dataset, and Figure 3 compares the loss curves for the IMDB-B dataset. The line depicts the mean loss over our ten runs, and the shading shows standard deviation. It appears that $\mathcal{G}$-Mixup performs better in terms of test loss for both datasets. However, the standard deviations overlap. The authors' original results for the REDDIT-B and IMDB-B dataset are shown in Figure 1. We note that the authors do not explicitly state what the lines or shadings of their figures represent.

4.2 Results beyond original paper

Additional Result 1 — Using the hyperparameters specified in Section 3.3 on the PROTEINS dataset, we tested the performance of the GCN model with and without $\mathcal{G}$-Mixup. This experiment is related to Claim 2(a) of Section 2. We chose this dataset because we wanted to analyze the performance of $\mathcal{G}$-Mixup on an additional full dataset, and this dataset was small enough in terms of memory for us to implement our experiment.

Table 2 shows the classification accuracy results for our experiment in the first column. As in our experiment on the REDDIT-B dataset, the performance of the GCN model with the $\mathcal{G}$-Mixup procedure leads to higher classification accuracy than the baseline GCN model.

Figure 4 in Section A.3 compares the loss for our experiments on the training, validation, and test splits of the PROTEINS dataset. It appears that $\mathcal{G}$-Mixup with the GCN architecture perform better in terms of loss. In fact, the vanilla GCN network does not seem to converge at all, as the classification accuracy for the training set does not appear to improve.

Additional Result 2 — We did hyperparameter searches for the mixup ratio and graphon resolution on the three datasets. We chose to study these hyperparameters because according to Theorem 4.2 and Theorem 4.4, the mixup ratio λ and graphon resolution (which upper bounds the number of nodes for a synthetic graph) affect the probability that a synthetic graph generated from the mixed-up graphon contains a mixed-up version of original graph data's discriminative motifs.

This claim is also related to claim 2(b): by setting the mixup ratio $\lambda = 0$, $\mathcal{G}$-Mixup degenerates into another graph data augmentation method, **graphon-based edge perturbation** [6]. Thus, the hyperparameter search for the mixup ratio also allows us to compare the performance of $\mathcal{G}$-Mixup with a "simpler" graph data augmentation method.

Our results for the mixup ratio and graphon resolution are shown in Tables 4 and 5 respectively. Overall, it does not appear that varying either hyperparameter leads to significant changes in the classification accuracy of $\mathcal{G}$-Mixup.

λ	0	0.001	0.01	0.1	0.5	No mixup
PROTEINS	62.69 ± 4.1	62.69 ± 4.8	62.91 ± 4.99	61.84 ± 3.53	63.36 ± 4.98	58.52 ± 3.11
IMDB-B	72.0 ± 3.61	73.55 ± 2.24	72.64 ± 2.75	73.8 ± 3.14	72.39 ± 2.61	73.15 ± 2.5
REDDIT-B	89.78 ± 1.16	89.20 ± 1.48	88.75 ± 3.29	87.82 ± 3.87	74.78 ± 2.72	81.4 ± 5.49

Table 4. Average accuracy $\pm$ standard deviation over 10 seeds for varying mix-up ratios λ. The last column is using the baseline GCN model with no $\mathcal{G}$-Mixup.

r	-15	-10	-5	0	5	10	15	No mixup
PROTEINS	61.84 ± 3.55	63.86 ± 4.12	63.41 ± 3.21	62.47 ± 5.09	62.69 ± 3.72	62.96 ± 2.98	63.81 ± 3.32	58.52 ± 3.11
IMDB-B	73.34 ± 2.26	74.05 ± 1.98	73.14 ± 2.49	73.5 ± 2.99	73.5 ± 3.59	73.8 ± 2.47	73.75 ± 2.74	73.15 ± 2.5
REDDIT-B	83.15 ± 5.32	84.73 ± 4.78	82.78 ± 6.51	84.8 ± 4.75	84.9 ± 5.14	81.58 ± 5.31	85.03 ± 5.41	81.4 ± 5.49

Table 5. Average accuracy $\pm$ standard deviation over 10 seeds for varying graphon resolutions given by (median number of nodes in training graphs) $+ r$. The last column is using the baseline GCN model with no $\mathcal{G}$-Mixup.

5 Discussion

Verification of Claims — We summarize whether or not our results support the original paper's claims.

- Result 1 supports the authors' Claim 1, which is their theoretic claim that $\mathcal{G}$-Mixup produces synthetic graphs that contain a mixture of key graph topologies of source graphs.
- In Result 2, we are able to closely replicate the original paper's values for classification accuracy for the REDDIT-B and IMDB-B datasets. In addition, our results for the REDDIT-B and PROTEINS datasets from Result 2 and Additional Result 1 support Claim 2(a), which is the authors' claim that $\mathcal{G}$-Mixup leads to better performance over a vanilla GCN model. However, our results are not statistically significant, and we are unsure how the original paper's standard deviations are much lower than ours with the same experimental setup of 10 random seeds. In addition, our classification results for the IMDB-B dataset do not support this claim, as we found that the vanilla GCN model gives better classification accuracy results than does the GCN model with $\mathcal{G}$-Mixup.
- In Additional Result 2, we found that varying the mixup ratio or the graphon resolution of the $\mathcal{G}$-Mixup method does not affect the classification accuracy. Furthermore, our results for the mixup ratio experiment do not support Claim 2(b), which is the authors' claim that $\mathcal{G}$-Mixup leads to better performance over other graph data augmentation methods.

Overall Conclusion — Although some of our results seem to indicate that $\mathcal{G}$-Mixup is a useful graph data augmentation method, we cannot provide statistically significant evidence of the original paper's experimental claims. Our verification of the authors' theoretical results support their theoretical claim. However, we feel that some of the assumptions used for the theoretical claim are too strong to apply $\mathcal{G}$-Mixup in practice. For example, it is assumed that every class in a graph dataset can be effectively estimated by a graphon, and that discriminative motifs exist and are an effective way to measure key graph topologies. In addition, our results from the hyperparameter searches suggest that it may not be the mixup of graph class topologies that actually leads to any better performance results, but rather that $\mathcal{G}$-Mixup injects noise to the data, which has also been shown to lead to better performance [11].

Strengths: Our proofs for the paper's theoretical results are much more extensive. We were able to essentially reproduce the results of the authors experiments.
Weaknesses: The main weaknesses of our experimental results relate to relatively high standard deviations for all of our results, which affected our analysis. Given more time, it would be ideal to run our experiments using more seeds.

5.1 What was easy

Although the authors' proofs of theoretical claims were sparse, they provided many helpful citations that helped us to fill in gaps easily. We also found these citations helpful overall for understanding the theory of graphons. The authors' code makes it possible to easily run the experiments on other datasets in the PyTorch-Geometric library.

5.2 What was difficult

We initially had issues running the $\mathcal{G}$-Mixup model for the full REDDIT-B dataset on Google Colab because we ran out of memory. We believe that the authors' code is not

well-optimized in this regard. Since most of the computation RAM is in generating the graphons, and the graphon construction algorithm is deterministic, it would have been easier to verify the authors' results if they provided pre-made graphons on some (pre-determined) training data. It was also difficult to install the correct Python packages in our campus computing cluster.

5.3 Communication with original authors

We have exchanged emails with the corresponding author, who offered suggestions on how to address our memory issues with the REDDIT-B dataset. However, their suggestions were not able to be implemented or they did not resolve our issues by the time of the writing of this report.

References

1. X. Han, Z. Jiang, N. Liu, and X. Hu. "G-Mixup: Graph Data Augmentation for Graph Classification." In: **Proceedings of the 39th International Conference on Machine Learning** 162 (July 2022), pp. 8230–8248. URL: https://proceedings.mlr.press/v162/han22c.html.
2. L. Lovasz and B. Szegedy. "Limits of dense graph sequences." In: **J. Combin. Theory Ser. B** 96.6 (2006), pp. 933–957. DOI: 10.1016/j.jctb.2006.05.002. URL: https://doi-org.proxy.lib.umich.edu/10.1016/j.jctb.2006.05.002.
3. H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization." In: **International Conference on Learning Representations** (2018).
4. Y. Rong, W. Huang, T. Xu, and J. Huang. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification." In: **International Conference on Learning Representations** (2020).
5. Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. "Graph Contrastive Learning with Augmentations." In: **Advances in Neural Information Processing Systems** 33 (2020), pp. 5812–5823.
6. Z. Hu, Y. Fang, and L. Lin. "Training graph neural networks by graphon estimation." In: **ArXiv** abs/2109.01918 (2021).
7. S. Chatterjee. "Matrix estimation by Universal Singular Value Thresholding." In: **The Annals of Statistics** 43.1 (2015), pp. 177–214. DOI: 10.1214/14-AOS1272.
8. T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: 2017.
9. K. Xu, W. Hu, J. Leskovec, and S. Jegelka. "How Powerful are Graph Neural Networks?" In: **ArXiv** abs/1810.00826 (2019).
10. C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. "TUDataset: A collection of benchmark datasets for learning with graphs." In: **ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)** (2020).
11. Y. Grandvalet, S. Canu, and S. Boucheron. "Noise Injection: Theoretical Prospects." In: **Neural Computation** 9.5 (July 1997), pp. 1093–1108. DOI: 10.1162/neco.1997.9.5.1093.

A Supplementary Materials

A.1 Description of $\mathcal{G}$-Mixup hyperparameters

We describe the hyperparameters relevant to the $\mathcal{G}$-Mixup method. We also state the values for these hyperparameters used in the in the original paper, as stated in the paper; we note that some of these values in the original author's code were different, and so we therefore changed it in our code.

- Augmentation ratio α: Multiplying the number of training graphs by this ratio gives the number of synthetic graphs generated by applying $\mathcal{G}$-Mixup to the graphons estimated from the training data. The authors use $\alpha = 0.2$ in their experiments.
- Mixup ratio λ and mixup interval $[\Lambda_1, \Lambda_2]$: Given graphons W_1, W_2 and mixup ratio $\lambda \in [0, 1]$, the authors' algorithm produces a new mixed-up graphon

$$\lambda W_1 + (1 - \lambda) W_2. \tag{3}$$

  The mixup ratio λ is randomly sampled from the interval $[\Lambda_1, \Lambda_2]$. The authors use the mixup interval $[0.1, 0.2]$ in their experiments.
- Augmentation number n_{aug}: In the $\mathcal{G}$-Mixup algorithm, we generate n_{aug} mixed-up graphons from any two distinct classes among all of the classes; the mixup ratios of these mixed-up graphons are given by λ_i for $i = 1, 2, \ldots, n_{\text{aug}}$. We generate $\lfloor \alpha n / n_{\text{aug}} \rfloor$ synthetic graphs from each mixed-up graphon, where n is the size of the original training set. For binary classification with $\Lambda_1 = \Lambda_2$, this parameter is made irrelevant. The authors use $n_{\text{aug}} = 10$ in their experiments.
- Graphon resolution: If n is the graphon resolution, then in the graphon estimation step, an $n \times n$ matrix is used to represent the graphon. To sample from the graphon, an $n \times n$ adjacency matrix is then generated. Isolated nodes are then removed to provide a synthetic graph. Thus, the resolution influences the size of synthetic graphs, but does not determine it. In particular, it is an upper bound on the number of nodes in the synthetically generated graphs. The authors use the median number of nodes in the training set as the resolution.

A.2 Theoretical result details

We first provide some of the relevant definitions and background for Section 4.1.1 (Result 1).

Definition A.1. A **discriminative motif** F_G of a graph G is the subgraph with the minimal number of nodes and edges that can decide the class of the graph G. Let $\mathcal{F}_{\mathcal{G}}$ denote the set of discrminative motifs for a set $\mathcal{G}$ of graphs.

The authors use this notion of discriminative motifs as their measure for "key topologies" among graphs of a certain class or label.

Given an arbitrary graph or graphon, we want to measure "how often" such a motif appears in the graph or graphon. The following definition will be used as this measure.

Definition A.2. Let F be an arbitrary graph.

- Let G be a graph. Then the **homomorphism density** of F with respect to the graph G is

$$t(F, G) = \frac{\hom(F, G)}{|V(G)|^{|V(F)|}},$$

  where $\hom(F, G)$ denotes the total number of graph homomorphisms from F to G, and $|V(F)|, |V(G)|$ denote the number of nodes of the graphs F, G respectively.

- Let W be a graphon. Then the **homomorphism density** of F with respect to the graphon W is

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

  where $E(F), V(F)$ denote the edge and vertex sets of F respectively.

Finally, the following norm on graphons provides a way to measure the "similarity" of graphons.

Definition A.3. Let $W : [0,1]^2 \to \mathbb{R}$ be a measurable function. Then the **cut norm** of W is defined as

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

where the supremum is taken over all measurable subsets $S, T \subset [0,1]$.

We note that the use of the box ($\square$) in the cut norm notation is the standard notation, and is used to distinguish cut norm from other norms.

Remark A.4. The cut norm is indeed a norm. In particular, it is homogeneous, so for $\alpha \in \mathbb{R}$ is a scalar and $W : [0,1]^2 \to \mathbb{R}$ a measurable function, then αW is also a measurable function on $[0,1]^2$, and

$$\|\alpha W\|_\square = |\alpha| \|W\|_\square.$$

In particular, note that

$$\|W\|_\square = \|-W\|_\square.$$

We now provide our detailed proof of Lemma 4.1.

Proof of Lemma 4.1. We expand upon the proof of this result in [2, Lemma 4.1]. First, notice that an equivalent definition for the cut norm of a graphon U is

$$\|U\|_\square = \sup_{f,g} \left| \int_{[0,1]^2} U(x,y) f(x) g(y)\, dx\, dy \right|, \tag{4}$$

where the supremum is taken over all measurable functions $f, g : [0,1] \to [0,1]$. Let $V(F) = \{1, \dots, n\}$ and $E(F) = \{e_1, \dots, e_m\}$, where $e_t = (i_t, j_t)$. Then

$$t(F, W) - t(F, W') = \int_{[0,1]^n} \left(\prod_{e_t\in E(F)} W(x_{i_t}, x_{j_t}) - \prod_{e_t\in E(F)} W'(x_{i_t}, x_{j_t}) \right) \prod_{i\in V(F)} dx_i.$$

For $t = 1, \dots, m$, define

$$\begin{aligned} X_t(x_1, \dots, x_n) &= \left(\prod_{k=1}^{t-1} W(x_{i_k}, x_{j_k}) \right) (W(x_{i_t}, x_{j_t}) - W'(x_{i_t}, x_{j_t})) \left(\prod_{k=t+1}^{m} W'(x_{i_k}, x_{j_k}) \right) \\ &= W(x_{i_1}, x_{j_1}) \cdots W(x_{i_t}, x_{j_t}) W'(x_{i_{t+1}}, x_{j_{t+1}}) \cdots W'(x_{i_m}, x_{j_m}) \\ &\quad - W(x_{i_1}, x_{j_1}) \cdots W(x_{i_{t-1}}, x_{j_{t-1}}) W'(x_{i_t}, x_{j_t}) \cdots W'(x_{i_m}, x_{j_m}). \end{aligned}$$

Notice that for $t = 1, \dots, m-1$, the second term of $X_t(x_1, \dots, x_n)$ cancels out with the first term of $X_{t+1}(x_1, \dots, x_n)$. Thus,

$$\prod_{e_t\in E(F)} W(x_{i_t}, x_{j_t}) - \prod_{e_t\in E(F)} W'(x_{i_t}, x_{j_t}) = \sum_{t=1}^{m} X_t(x_1, \dots, x_n).$$

Fix all variables $x_j \in [0,1]$, where $k \neq i_t, j_t$; we then have that

$$f(x_{i_t}) = \prod_{k=1}^{t-1} W(x_{i_k}, x_{j_k}), \quad g(x_{j_t}) = \prod_{k=t+1}^{m} W'(x_{i_k}, x_{j_k})$$

are both measurable functions $[0,1] \to [0,1]$ in x_{i_t} and x_{j_t} respectively. Then by Equation 4,

$$\left| \int_{[0,1]^2} X_t(x_1, \ldots, x_n)\, dx_{i_t}\, dx_{j_t} \right| \leq \|W - W'\|_{\square},$$

and so

$$\begin{aligned} \left| \int_{[0,1]^n} X_t(x_1, \ldots, x_n) \prod_{i=1}^{n} dx_i \right| &\leq \int_{[0,1]^{n-2}} \left| \int_{[0,1]^2} X_t(x_1, \ldots, x_n)\, dx_{i_t}\, dx_{j_t} \right| \prod_{i \neq i_t, j_t} dx_i \\ &\leq \|W - W'\|_{\square}. \end{aligned}$$

Thus, we see that

$$\begin{aligned} |t(F,W) - t(F,W')| &= \left| \int_{[0,1]^n} \sum_{t=1}^{m} X_t(x_1, \ldots, x_n) \prod_{i=1}^{n} dx_i \right| \\ &\leq \sum_{t=1}^{m} \left| \int_{[0,1]^n} X_t(x_1, \ldots, x_n) \prod_{i=1}^{n} dx_i \right| \\ &\leq m \|W - W'\|_{\square}. \end{aligned}$$

Recall that $m = e(F)$, and so we have the desired inequality. □

A.3 Figures for experimental results

The following figures show the loss curves for experiments from Sections 4.1.2 and 4.2.1, as well as the original paper's loss curves.

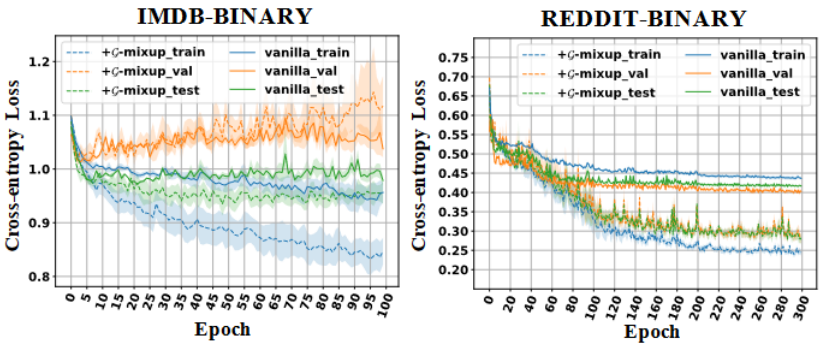

Figure 1. The training/validation/test loss curves on IMDB-B and REDDIT-B with GCN as backbone. The curves are depicted on ten runs. This is part of Figure 4 in the original paper [1].

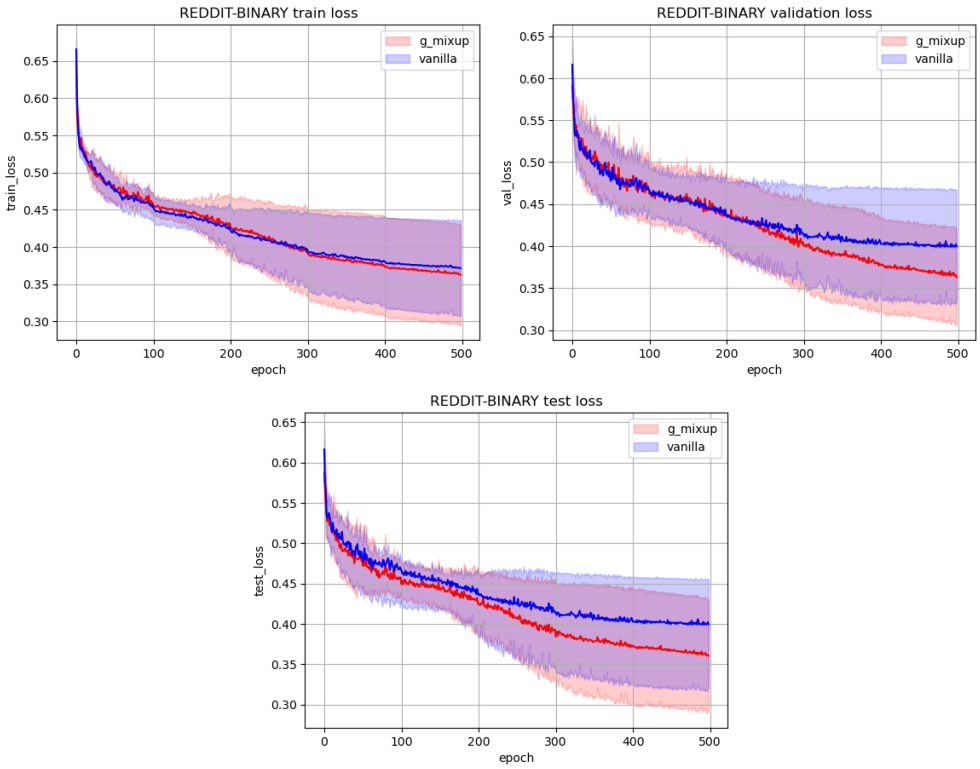

Figure 2. The training/validation/test curves on REDDIT-B with GCN as a backbone. The curves are depicted on ten runs. The line is the mean of the corresponding loss, while the shaded area is ± the standard deviation of the losses.

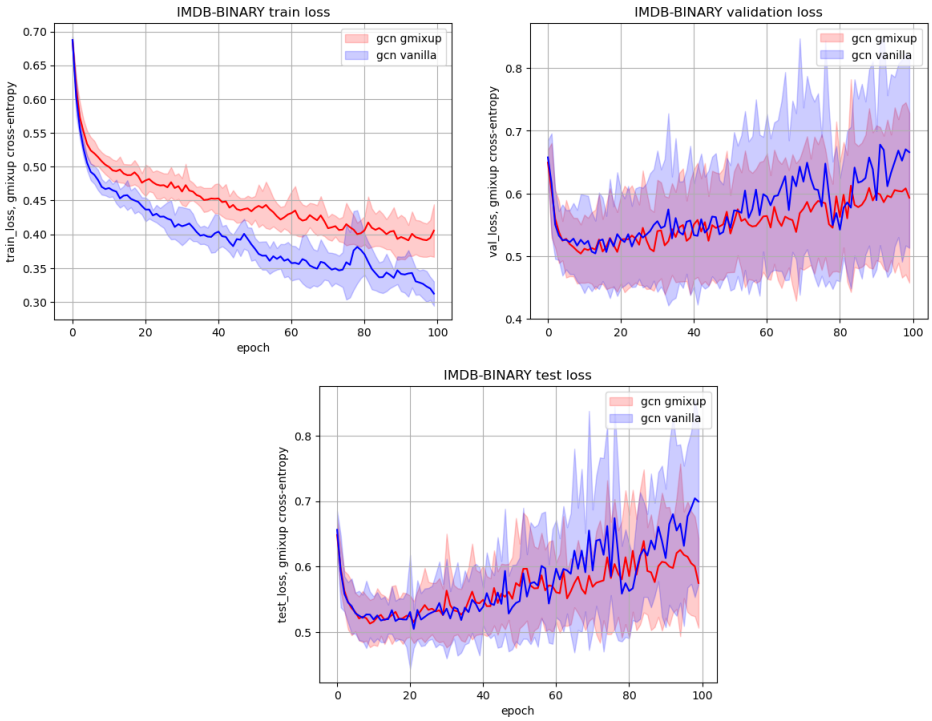

Figure 3. The training/validation/test curves on IMDB-BINARY with GCN as a backbone. The curves are depicted on ten runs. The line is the mean of the corresponding loss, while the shaded area is ± the standard deviation of the losses.

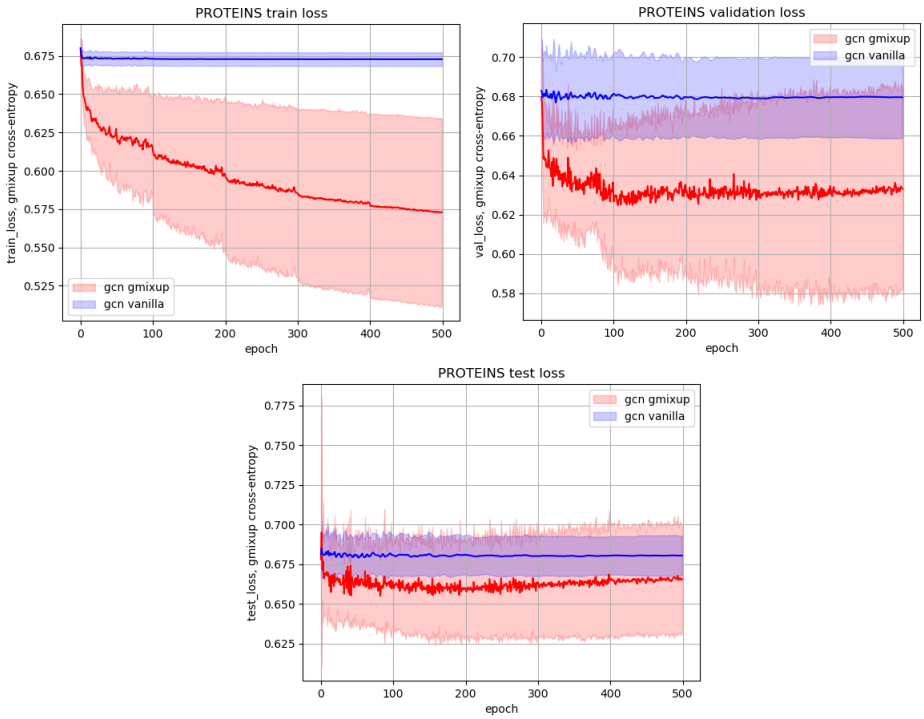

Figure 4. The training/validation/test curves on PROTEINS with GCN as a backbone. The curves are depicted on ten runs. The line is the mean of the corresponding loss, while the shaded area is ± the standard deviation of the losses.