G-Mixup Revisited: A Practical Re-implementation and Analysis of Graph Data Augmentation for Graph Classification

Tanya Djavaherpour
McMaster University
Hamilton, ON, Canada
djavahet@mcmaster.ca

ABSTRACT

This report presents a comprehensive re-implementation of the G-Mixup method, an innovative approach for graph data augmentation to enhance graph classification models' performance. Initially proposed in [3], G-Mixup has demonstrated its effectiveness in improving the generalization and robustness of Graph Neural Networks (GNNs). Our re-implementation seeks to validate these findings by closely following the methodology outlined in the original paper, employing IMDB-BINARY, IMDB-MULTI, and REDDIT-BINARY for testing and comparison.

Key aspects of our study include a detailed analysis of the G-Mixup algorithm, the challenges encountered during the re-implementation process, and an evaluation of its impact on model performance.

Through this re-implementation, we aim to provide a deeper understanding of G-Mixup's inner workings and its potential applications in graph-based machine learning tasks. The report concludes with a discussion of possible future enhancements and the broader implications of our findings for the field of graph data processing.

PVLDB Reference Format:
Tanya Djavaherpour. G-Mixup Revisited: A Practical Re-implementation and Analysis of Graph Data Augmentation for Graph Classification.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/tanya-jp/G-Mixup.

1 INTRODUCTION

In recent years, deep learning has shown a revolution in different machine learning tasks, such as image classification and speech recognition. Typically, these successes pivot around data represented in Euclidean spaces. A growing number of applications use data created from non-Euclidean domains and represented as graphs with complicated interactions and interdependence among items [6]. GNNs utilize graph convolutions to extract local characteristics embedded in network data including graphs [5]. Nevertheless, the majority of currently used graph data augmentation techniques work to enhance graphs only within a single graph by changing its edges or nodes, which prevents information from being shared across instances. There is still much to learn about between-graph augmentation techniques, or data augmentation between graphs. Mixing up graph data presents a significant challenge due to its unique characteristics, making it difficult to apply the Mixup strategy commonly used for regular, well-aligned Euclidean data like images and tables [3].

The limiting objects for sequences of big, finite graphs with respect to the so-called cut metric are called Graphons, which is shorthand for graph functions; [1] and [2] introduced and developed them. To address the aforementioned issues and challanges in graph processing, in [3] they proposed G-Mixup, a class-level graph data augmentation approach that mixes up graph data using graphons.

2 CORE STRUCTURAL ELEMENTS

In the original paper, [3], the implemented technique was tested on five different datasets, including MDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI-5K, and REDDIT-MULTI-12K. These datasets are available via `torch_geometric` package in `Python`. As shown in Table 1, REDDIT-MULTI-5K and REDDIT-MULTI-12K are large datasets, and due to lack of resources we did not run experiments for these two.

In this section, we will discuss our implementation and the results.

2.1 Figures

As illustrated in Figure 1, we extended our experiment to more epochs than what is reported in the paper [3]. This was done to ensure that the reported epoch is precisely the one before overfitting occurs. Based on our study, the reported epoch was indeed correct. Figure 2 depicts the achieved accuracy just before the onset of overfitting.

2.2 Tables

The results we achieved are presented in Table 2. The reported results in [3] are based on 10 different runs. In our study, due to limited resources, each experiment was conducted only once. As indicated in Table 2, our results are consistent with those reported in [3].

2.3 Listings and Styles

For the implementation we chose `Google Colab` for several reasons. First, the ability of `Google Colab` in handling compatible packages was a key factor. Another reason was related to the `torch_geometric` package that, as mentioned before, is used to receive datasets presented in Table 1. This package requires `CUDA` to be installed on the system; however, it can be accessible with some commands on `Google Colab`. The last but not least reason is the provided `GPU` on `Google Colab` that can be helpful in speeding up the training process. The `Python` version used in our implementation is `3.10.12`.

Isomorphism, in the context of graph theory, is a concept used to determine if two graphs are essentially the same, differing only in

Table 1: Details of Datasets Utilized in Our Study, Referenced from [3].

Dataset	REDD-B	IMDB-B	IMDB-M	REDD-M5	REDD-M12
#graphs	2000	1000	1500	4999	11929
#classes	2	2	3	5	11
#avg.nodes	429.63	19.77	13.00	508.52	391.41
#avg.edges	497.75	96.53	65.94	594.87	456.89

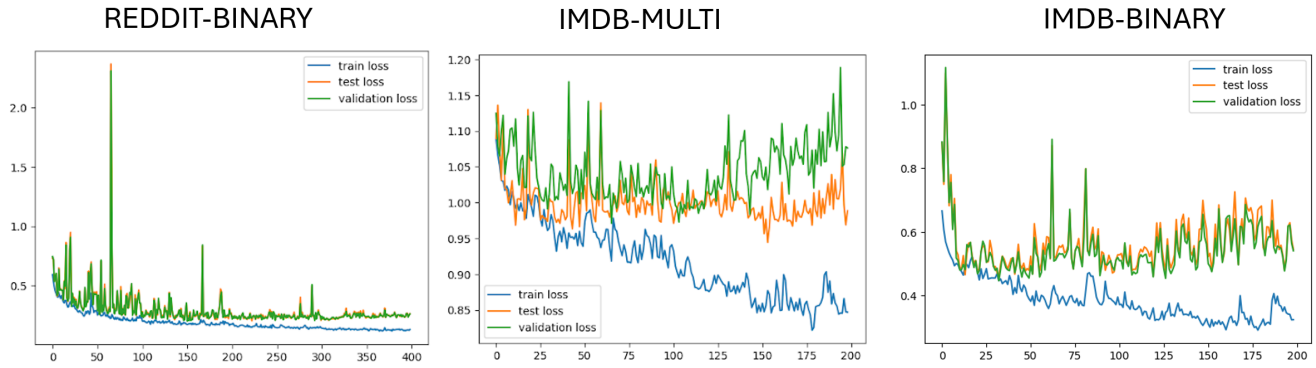

Figure 1: The training/validation/test loss curves on IMDB-BINARY, IMDB-MULTI and REDDIT-BINARY with GIN as backbone.

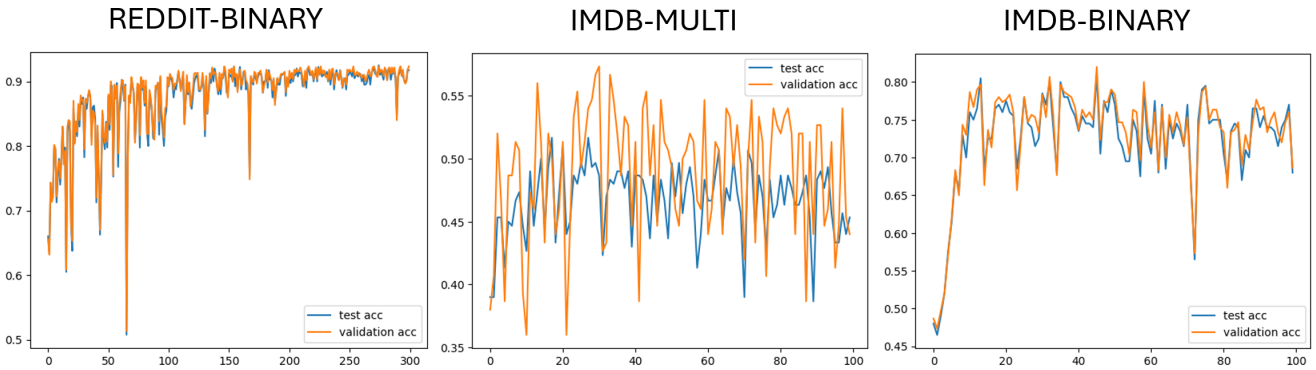

Figure 2: The validation/test accuracy curves on IMDB-BINARY, IMDB-MULTI and REDDIT-BINARY with GIN as backbone before overfitting.

Table 2: Comparative Analysis of Our Findings with the Results from [3].

Dataset	REDD-B	IMDB-B	IMDB-M
G-Mixup [3]	92.90 ±0.87	71.94 ±3.00	50.46 ±1.49
Our G-Mixup	92.33	71.67	49.33

the labeling of their vertices and the representation of their edges. In [8] they designed a basic neural architecture, Graph Isomorphism Network (GIN), and demonstrated its discriminative and representational power. In this study, like the original paper [3], we tested our method on GINs using `PyTorch`. Generally, in this implementation, `PyTorch` has been used for data processing and the training process.

All other details in the implementation, such as the division percentage of the training/testing/validation data, number of batches, and the number of hidden layers, are based on the original paper [3].

Based on the reference paper [3], the learning rate starts at 0:01 and decreases by 50% after every 100 epochs. The batch size is set at 128. We divided the dataset into train, val, and test groups in a 7:1:2 ratio.

2.4 Math and Equations

The mathematical part behind our implementation is exactly based on what is used in the [3]. These formula are as:

$$\text{Graphon Estimation:} \quad \mathcal{G} \to W_{\mathcal{G}}, \mathcal{H} \to W_{\mathcal{H}} \tag{1}$$

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

$$\text{Graph Generation:} \quad \{I_1, I_2, \ldots, I_m\} \sim \mathcal{G}(K, W_{\mathcal{I}}) \tag{3}$$

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

where $W_{\mathcal{G}}$ and $W_{\mathcal{H}}$ are the graphon representations for the sets of graphs $\mathcal{G}$ and $\mathcal{H}$, respectively. The composite graphon is symbolized by $W_{\mathcal{I}}$, and the scalar λ within the closed interval [0,1] serves as a balancing parameter to modulate the proportional contributions from each originating graph set. The assemblage of synthetic graphs derived from $W_{\mathcal{I}}$ is denoted as $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$. The vectors $y_{\mathcal{G}} \in \mathbb{R}^C$ and $y_{\mathcal{H}} \in \mathbb{R}^C$ comprise the ground-truth labels for graphs in $\mathcal{G}$ and $\mathcal{H}$, respectively, where C represents the count of classes. The labeling vector for synthetic graphs within the set $\mathcal{I}$ is represented as $y_{\mathcal{I}} \in \mathbb{R}^C$ [3].

3 CITATIONS

As mentioned in this article, the entire part of this study is based on [3]. In this report, we have endeavored to reference seminal studies in the domain of graph estimation and graph processing. There are additional pivotal studies employed in the original paper, such as [4], [7], [9], and [10].

ACKNOWLEDGMENTS

I would like to extend my gratitude to Professor Lingyang Chu, whose insightful guidance and unwavering support have been invaluable throughout the course of the project for the Machine Learning on Graphs course at McMaster University.

REFERENCES

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

[2] Christian Borgs, Jennifer T Chayes, László Lovász, Vera T Sós, and Katalin Vesztergombi. 2012. Convergent sequences of dense graphs II. Multiway cuts and statistical physics. *Annals of Mathematics* (2012), 151–219.

[3] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. 2022. G-Mixup: Graph Data Augmentation for Graph Classification. *arXiv preprint arXiv:2202.07179* (2022).

[4] László Lovász. 2012. *Large networks and graph limits*. Vol. 60. American Mathematical Soc.

[5] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. 2020. Graphon Neural Networks and the Transferability of Graph Neural Networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1702–1712. https://proceedings.neurips.cc/paper_files/paper/2020/file/12bcd658ef0a540cabc36cdf2b1046fd-Paper.pdf

[6] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[7] Hongteng Xu, Dixin Luo, Lawrence Carin, and Hongyuan Zha. 2021. Learning graphons via structured gromov-wasserstein barycenters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10505–10513.

[8] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[9] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[10] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. 2020. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819* (2020).