Inverse APPNP: Solution for OGB-LSC 2022 MAG240M

Hang Li[1,*] **Haoyu Han**[1*], **Hongzhi Wen**[1*],
Wei Jin[1], **Feng Shi**[2], **Victor Lee**[2], **Jiliang Tang**[1]
[1]Michigan State University [2]TigerGraph Company
{lihang4,hanhaoy1,wenhongz,jinwei2,tangjili}@msu.edu,
{bill.shi,victor}@tigergraph.com

Abstract

Graph neural networks (GNNs) have demonstrated empirical success on the node classification task. Despite the success, they have majorly been validated on comparatively small datasets, which does not fulfill the growing need for handling extremely large-scale graph data in real-world scenarios. To tackle these issues, we develop models that are both efficient and effective for massive graph datasets. In particular, we participated in the Open Graph Benchmark Large-Scale Challenge (OGB-LSC) MAG240M tack in NeurIPS 2022 and our solution achieved the top 3 performance on the provided huge heterogeneous academic graph. To facilitate the development of GNNs on massive datasets, we release the source code at: https://github.com/hzlihang99/NeurIPS2022-MAG240M.

Keywords

MAG240M, Large-Scale Node Classification, Graph Neural Networks.

1 Introduction

Graph representation learning has become a trending topic in the machine learning community due to the prosperity of graph data. In recent years, various graph ML models including graph neural networks (GNNs) [1, 2, 3, 4] have shown their power in extracting information from graphs in various domains such as social networks [5], recommender systems [6, 7], knowledge graphs [8], molecule simulations [9] and single-cell analysis [7, 10]. However, in real-world scenarios, there is a growing need for handling extremely large-scale graphs, while the efforts on developing graph ML models for massive datasets have been quite limited.

To facilitate the development of graph ML models for large-scale graphs, Hu et al. [11] organized the second OGB Large-Scale Challenge (OGB-LSC) in NeurIPS 2022 where participants are asked to perform the node classification task on a large-scale real-world graph dataset MAG240M. Specifically, MAG240M is extracted from Microsoft Academic Graph (MAG)[12], which consists of 244,160,499 nodes and 1,728,364,232 edges. There are three types of nodes, i.e., author, institution and paper, and three types of edges, e.g., paper-cite-paper, author-write-paper and author-affiliated-institution. Among the 121 million paper nodes, 1.4 million papers are extracted from arXiv and annotated with 153 arXiv subject category labels. The competition holder splits these papers into training, validation, and test sets based on their publication years. The goal of the competition is to predict the category of last year's published papers based on the previous years' paper categories and their connection graphs. The OGB-LSC has been successfully held and received a significant number of comprehensive solutions.

[*]These authors contribute equally to this work.

Preprint. Under review.

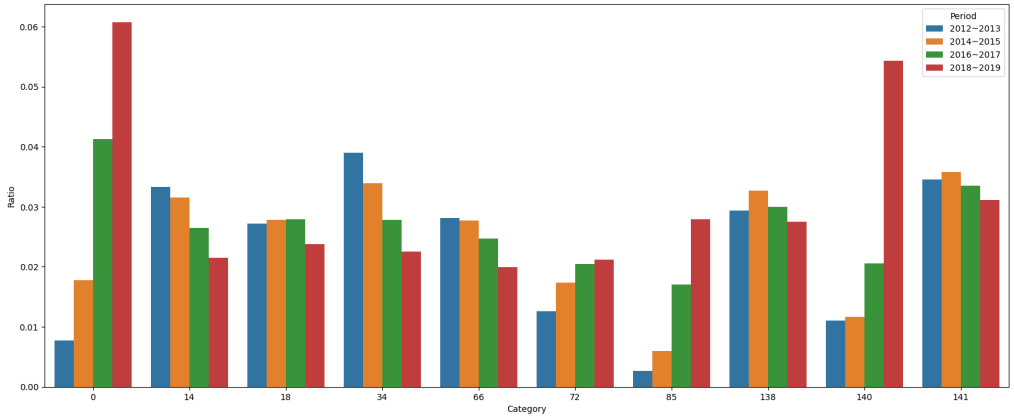

Figure 1: Label distribution shift of top 10 most frequent categories from 2012 to 2019

As one of the winning team (top-3 ranking) in this competition, we successfully tackled the problems with more advanced algorithms. On the basis of previous successful models like R-unimp [13], we combined their predicted logits with an inverse APPNP [14] model, which achieved a solid performance boost on the validation set. In the rest of this paper, we introduce the details about the algorithms and experiments that we explored during the competition. In Section 2.1, we first introduce findings from the preliminary data analysis. Based on these findings, in Section 2.3, we explored potential approaches to improve the last year's best method by incorporating some newly proposed algorithms. In Section 3, we introduce the details of our algorithms and show the improvements brought by the method.

2 Preliminaries

2.1 Data Analysis

MAG240M-LSC is a heterogeneous academic graph extracted from the Microsoft Academic Graph (MAG) which consists of three types of nodes, i.e., Paper, Author and Institution. There are 121M papers collected from 1901 to 2020, of which only 1.4M arxiv papers were annotated with 153 arXiv subject areas. Besides, only paper nodes have features, whereas the features of author and institution nodes are generated by paper nodes via 1-hop feature aggregation.

2.1.1 Label Distribution Shift

We analyze the label distribution of the arxiv papers from different years. Specifically, we choose the 10 most popular topics in 2018-2019, which is the latest available year from the train and validation split of MAG240M, and investigate their changing trend over years. From Fig.1, we find that each category's percentage is changing with the years. Besides, there are some obvious trends in categories such as 0, 14, 34, and 66; and we name this phenomenon as label distribution shift.

2.1.2 2-hop Neighbors

The competition task is to predict the labels of arxiv papers. Among last year's solutions, the two-layer RGAT achieves the best performance, while adding more layers has little effect on performance. As a result, we filter out 2-hop neighbors of arxiv papers to save time and memory costs. The statistics of these two graphs are shown in Table 1.

Table 1: Statistic of two graphs.

Graph	Nodes	Edges
Original	244 million	3.4 billion
2-hop Subgraph	78 million	2.2 billion

2.2 Baseline Models

In this subsection, we compare the performance of different baseline models. As the MAG240M-LSC is a heterogeneous network, Relational Graph Attention Neural Network (RGAT) [11] was widely utilized in the solutions of the previous year and has shown better performance than other methods. Since the features for author and institution nodes are directly generated from the paper nodes, the features of these three types of nodes are homogeneous. Then we wonder if considering the types of nodes is necessary. To answer this question, we train both RGAT which considers node types and GAT which overlooks the node types on the whole graph. The performance is shown in Table 2. We can observe that the GAT performs slightly better than the RGAT, which demonstrates that it is not necessary to consider the types of nodes. That means that we can convert the original heterogeneous network to a homogeneous graph. This data transformation gives us abundant flexibility to explore various GNN models designed for homogeneous graphs.

One issue with GAT and RGAT is the high time complexity. Due to the limited computation resources, it is hard to tune these two models. Some decoupled GNNs, such as SGC [15] and SIGN [16], achieve great efficiency since they only need to propagate once. However, they often perform worse than GCN. APPNP [14] is also a decoupled GNN which achieves good performance, but it needs to propagate in every epoch. Therefore, we introduce a new model called Inverse APPNP, which first uses the personalized PageRank propagation to propagate original features and then trains an MLP based on the new features. In particular, the new feature generation process can be expressed as follows:

$$\begin{aligned} \mathbf{Z}^{(0)} &= \mathbf{X} \\ \mathbf{Z}^{(k+1)} &= (1-\alpha)\tilde{\mathbf{A}}\mathbf{Z}^{(k)} + \alpha\mathbf{X}, \end{aligned} \tag{1}$$

where $\mathbf{X}$ is the original feature matrix, $\tilde{\mathbf{A}}$ is the normalized adjacency matrix, k is the propagation steps, and α is the teleportation parameter. Note that this propagation is only needed once and it can be seen as a data preprocessing step. Then we can train MLPs based on the new features. From the experiments, we find that $k = 5$ and $alpha = 0.05$ perform the best. As shown in Table 2, the performance of Inverse APPNP is slightly worse than GAT and RGAT. However, the training time of Inverse APPNP is much less than GAT and RGAT. Consequently, we can conduct numerous investigations because training an MLP requires less computational cost.

Table 2: Validation Accuracy of baselines.

Method	MLP	GAT	RGAT	Inverse APPNP
Valid ACC	52.67	70.21	70.02	69.77

2.3 Explorations

To boost the performance of existing baseline models, we experimented with various investigations like year embedding, input feature, model structure and loss function during the competition. Unfortunately, most of these investigations did not bring steady performance gains. Besides, due to our computing resource limitations, we finally chose to not apply some of them in our submitted result. Though some explorations did not directly contribute to our final model's performance, we still want to share interesting observations we made. We hope that these findings can shed some light on future research on MAG240M dataset.

2.3.1 Solving Label Distribution Shift Issue

In Section 2.1.1, we found that the label distributions of different years are quite different. To mitigate this issue, we choose three different strategies.

Loss Reweighting. From the analysis, the nearby years have similar label distributions. Given that the validation papers are from 2019 and the test papers are from 2020, we give higher weight to the loss for papers from the most recent years. Specifically, the loss can be written as follows:

$$L = \sum_i w_{year[i]} CE(\hat{y}_i, y_i), \tag{2}$$

where $w_{year[i]}$ is the weight for the year of paper i, $\hat{y}$ is the prediction and CE is the Cross Entropy loss. We tried different strategies for selecting the weight, and $w_{2019} = 5, w_{2018} = 4, w_{2017} = 3, w_{2016} = 2, w_{others} = 1$ achieves the best.

One-hot Embedding. For this investigation, we first convert years into one-hot vectors and then concatenate the one-hot vectors with node features. Since there is no test year during training, we set the year of test papers to the same year as validation papers.

Position Embedding. The position Embedding in transformer [17] has shown great success. We also adopt the position embedding method to get year embedding.

From Table 3, we can find that each strategy improves performance for the Inverse APPNP model. The performance can be further improved by combining them.

Table 3: Validation Accuracy of Different Strategies.

Strategies	No	Loss Reweighting	One-hot Embedding	Position Embedding	Combine
Valid ACC	69.77	70.60	69.92	70.78	71.16

2.3.2 Input Feature Tuning

Prior research has shown that the quality of node features is one of the crucial factors for the success of node classification. However, in this competition, MAG240M only provides valid features on paper nodes. Most recent research adopted the strategy provided by Hu et al[11], which takes the mean feature aggregation result of the first order neighbors as the initial features for author and institution nodes. This aggregation strategy is naive yet effective and it can also be seen as a layer of GNN model. Based on this fact, during our exploration, we hope to develop a new method, which introduces different sources of information to author and institution nodes and increases the expressiveness of the representations generated by the GNN models.

The most straightforward way to initialize nodes with valid features other than neighbor aggregation is to use each node's unique id, which is a common trick used in most of the recommendation systems[18]. However, achieving such goal is non-trial, since the number of author nodes is extremely large in MAG240M, and it is impossible for us to train a learnable embedding matrix for each author node with our available computing resources. To simplify the problem, we choose to only train a learnable embedding matrix for institution nodes, and use a similar neighbor aggregation method to generate node features for author nodes. Under this setting, we train the model with modified input features and compare it with the original baseline model. Unfortunately, this modification did not bring the model with steady improvements and we argue that this may be caused by neglecting the difference between authors with similar institution affiliations.

2.3.3 Module and Loss Replacement

The winning team of last year utilized the R-GAT model as its backbone model and applied a bag of tricks to boost its performance. In our explorations, we tried to replace the components of the winning teams' model with some most recent proposed tricks and modules to further improve its performance. First, since R-GAT builds individual GAT models for different types of relation, it suffers from the static attention defects of the original GAT model [2]. To increase the expressiveness of the current baseline, we replace all GAT layers with the GATv2 [19] module. After applying such change, our new model receives some unsteady performance gain, but the GPU memory consumption increased by $4\times$ times. Due to the limitation of the computing resource, we chose to not apply it in our final submission. Apart from the module update, we also tried the symmetric adjacency matrix normalization trick and robust loss function [20]. However, the modified model's performance did not outperform the baseline model and we argue that those tricks may be fit for the original paper-to-paper graph and further efforts are needed to adapt them to the MAG240M dataset.

3 Our Methods

In this section, we introduce two methods for the competition. The first one is our submitted solution. And the second one can further improve the performance, but due to time and computational resource constraints, we didn't submit the results by the deadline.

3.1 Inverse APPNP + RGAT

The winner solution of last year [21] provided some tricks to train RGAT. However, the code provided is written by PaddlePaddle. To better integrate with other models, we implement the same tricks for RGAT using Pytorch. Besides, most of the tricks have been designed for RGAT and they are not applicable to other models. To further boost the Inverse APPNP, we add the embeddings of RGAT into the input features. We also concatenate the pretrained metapath2vec embedding into the input features. The whole architecture is shown in Figure 2, where the numbers are the number of latent dimensions.

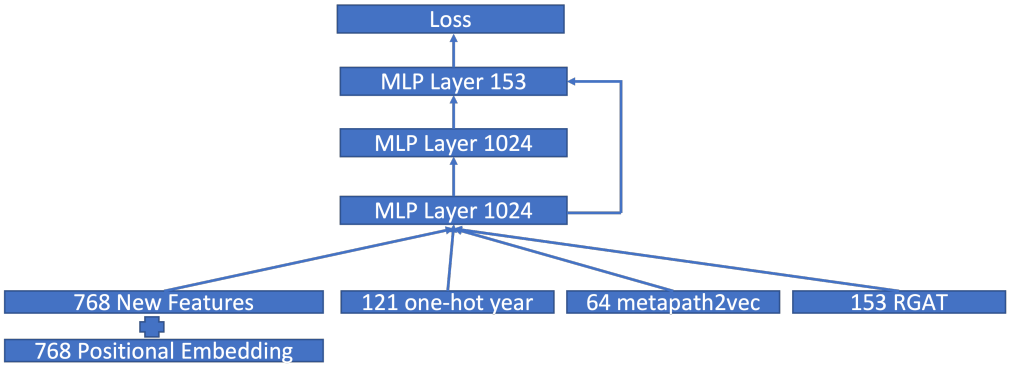

Figure 2: The architecture of Inverse APPNP + RGAT.

Using the validation set to train the model is a common trick for improving performance. To make full use of the validation set, we randomly split the validation set into 5 folds. For each training, we choose 1 fold as the validation set and add the other 4 folds into the training set. The performance is shown in Table 4. From the results, our method has an average improvement of 0.24% compared to RGAT.

For the submission results, we ensemble all the results of our method and RGAT with post-smoothing [21].

Table 4: Validation Accuracy of InverseAPPNP + RGAT.

Folds	RGAT	Inverse APPNP + RGAT	Improvement
Fold 0	76.91	77.16	0.25
Fold 1	76.78	77.05	0.27
Fold 2	76.78	76.94	0.16
Fold 3	77.00	77.06	0.06
Fold 4	76.18	76.66	0.48

3.2 Multiview4GNN

Our Multiview4GNN [22] adopts an alternating training procedure. We replace the backbone model from MLP to RGAT. The whole framework is shown in Figure 3. First, we train an RGAT model using the true labels. Then, the Feature Enhanced Label Propagation is employed to generate an equal number of Pseudo Labels with corresponding weights. Afterwards, we finetune the RGAT model with the true and pseudo labels. This procedure can be repeated multiple times, and finally we use the generated pseudo labels for prediction. Initial experiment results indicate that Multiview4GNN (77.21) is superior to RGAT (76.91). Due to time and resources constraints, we were unable to submit this result. But we hope that this method can inspire some new designs in the future.

4 Conclusion

In this report, we present our explorations and solutions for the OGB-LSC 2022 competition. From the data analysis, we identify the label distribution shift issue and then investigate three potential

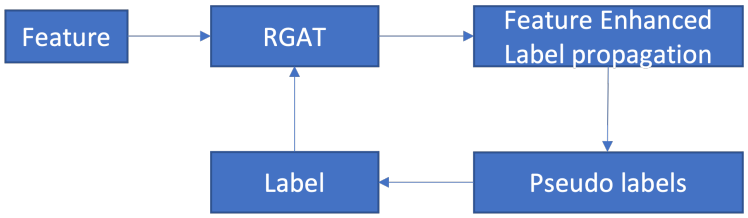

Figure 3: The architecture of Multiview4GNN.

solutions. By comparing the baselines, we find that the heterogeneity of MAG240M-LSC graph may not be significant, giving us the flexibility to investigate various GNN models. Inspired by SGC and APPNP, we propose a novel, efficient method - Inverse APPNP. Furthermore, combining RGAT with Inverse APPNP yields good results. We further provide a potential solution Multiview4GNN which demonstrates promising initial results. We hope that our explorations can help with new model designs in the future.

References

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

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

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.

[4] Yao Ma and Jiliang Tang. Deep learning on graphs. Cambridge University Press, 2021.

[5] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. arXiv preprint arXiv:1111.4503, 2011.

[6] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pages 974–983, 2018.

[7] Hongzhi Wen, Jiayuan Ding, Wei Jin, Yiqi Wang, Yuying Xie, and Jiliang Tang. Graph neural networks for multimodal single-cell data integration. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 4153–4163, 2022.

[8] Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. Estimating node importance in knowledge graphs using graph neural networks. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pages 596–606, 2019.

[9] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. Machine Learning: Science and Technology, 2(2):025023, 2021.

[10] Dylan Molho, Jiayuan Ding, Zhaoheng Li, Hongzhi Wen, Wenzhuo Tang, Yixin Wang, Julian Venegas, Wei Jin, Renming Liu, Runze Su, et al. Deep learning in single-cell analysis. arXiv preprint arXiv:2210.12385, 2022.

[11] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. arXiv preprint arXiv:2103.09430, 2021.

[12] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. Quantitative Science Studies, 1(1):396–413, 2020.

[13] Yunsheng Shi, PGL Team, Zhengjie Huang, Weibin Li, Weiyue Su, and Shikun Feng. Runimp: Solution for kddcup 2021 mag240m-lsc.

[14] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997, 2018.

[15] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In International conference on machine learning, pages 6861–6871. PMLR, 2019.

[16] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. arXiv preprint arXiv:2004.11198, 2020.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[18] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. Multi-interest network with dynamic routing for recommendation at tmall. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pages 2615–2623, 2019.

[19] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? arXiv preprint arXiv:2105.14491, 2021.

[20] Yangkun Wang, Jiarui Jin, Weinan Zhang, Yong Yu, Zheng Zhang, and David Wipf. Bag of tricks for node classification with graph neural networks. arXiv preprint arXiv:2103.13355, 2021.

[21] Yunsheng Shi, PGL Team, Zhengjie Huang, Weibin Li, Weiyue Su, Shikun Feng, et al. R-unimp: Solution for kdd cup 2021 mag240m-lsc. Open Graph Benchmark-Large-Scale Challenge@ KDD Cup 2021, 2021.

[22] Haoyu Han, Xiaorui Liu, Haitao Mao, Torkamani Ali, Feng Shi, Victor Lee, and Jiliang Tang. Graph neural networks as multi-view learning. arXiv preprint arXiv:2206.03638, 2022.