Boosting the Speed and Performance in Training Large-scale Heterogeneous Graph at NeurIPS OGB 22

Zhe Dong*
Team PGL
Baidu Inc./Tianjin University
China
dongzhe05@baidu.com

Hongwei Chen*
Team PGL
Baidu Inc.
China
chenhongwei04@baidu.com

Siming Dai*
Team PGL
Baidu Inc.
China
daisiming@baidu.com

Yunsheng Shi
Team PGL
Baidu Inc.
China
shiyunsheng01@baidu.com

Weibin Li
Team PGL
Baidu Inc.
China
liweibin02@baidu.com

Zhengjie Huang
Team PGL
Baidu Inc.
China
huangzhengjie@baidu.com

Mingguo He
Team PGL
Baidu Inc./Renmin University of China
China
hemingguo@baidu.com

Zeyang Fang
Team PGL
Baidu Inc.
China
fangzeyang@baidu.com

Zeyu Chen
Team PGL
Baidu Inc.
China
chenzeyu01@baidu.com

Shikun Feng
Team PGL
Baidu Inc.
China
fengshikun01@baidu.com

Abstract

The influence of machine learning (ML) on large-scale graph data is substantial, and the 1st OGB Large-Scale Challenge (OGB-LSC) was organized on KDD Cup 2021 to invite participants to develop innovative methods for the large-scale graph network datasets. Team PGL won the championship with the R-UniMP model's final test set score of 75.49% on the MAG240M-LSC competition track last year. MAG240M-LSC is a heterogeneous academic graph extracted from the Microsoft Academic Graph (MAG) with multiple relations between papers, authors, and institutions. Participants are required to predict the topics corresponding to the publication. However, the problem of machine learning over large graphs is not yet solved. The MAG240M-LSC competition track on NeurIPS 2022 is now being held once more. In this competition, R-UniMP model has been substantially enhanced and optimized by an innovative positional encoding for GNN, combining all node hidden features with generalized pagerank weights, and faster neighbor sampling techniques. Besides, we provide a detailed recall of our key strategies and valuable findings during the entire competition. Our best single models can reach 74.04% in the official validation split. For final submission, we train our models with 5-Fold settings. And we make a bagging search for

[*]These authors contributed equally to this work

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

ensemble selections over our local 5-Fold splits. The final submission is bagged over 30 models' predictions, which achieves 75.70% in the final test set. The source code is available at `https://github.com/PaddlePaddle/PGL/tree/main/examples/NeurIPS2022-OGB-Challenge/MAG240M`.

1 Introduction

In recent years, there has been a significant amount of interest in graph-centered machine learning. Hu et al. [1] hold OGB Large-Scale Challenge (OGB-LSC) at NeurIPS 2022 again, which contains three large-scale real-world datasets corresponding to three common graph challenges: node prediction, link prediction, and graph prediction. MAG240M-LSC is one of the tasks asking participants to predict labels for nodes. MAG240M-LSC is extracted from Microsoft Academic Graph (MAG) [2], which contains 244,160,499 nodes and 1,728,364,232 edges, the largest dataset among OGB-LSC. Nodes in MAG240M-LSC represent papers, authors, and institutes. And we have three types of edges: paper-cite-paper, author-write-paper, author-affiliated-institute. Among the 121M paper nodes, there are about 1.4M nodes are from ARXIV annotated with 153 ARXIV subject areas. Features for paper nodes are extracted by powerful pre-trained language model RoBERTA [3] with concatenated title and abstract of the titles as inputs. The task is to predict the primary subject areas of the given ARXIV papers as an ordinary multi-class classification problem. The metric is classification accuracy.

The previous year, Team PGL won the competition for node prediction in MAG240M-LSC based on the R-GAT [1] model to learn node representation from aggregated multi-relation neighborhood information [4]. Additionally, predictions are significantly improved when label propagation and post-smoothing are applied to the observed labels in UniMP [5]. In this competition, Team PGL continues to incorporate various and novel graph neural network techniques and neighbor sampling optimizations to boost performance and speed respectively. The faster sampling speed of neighbors offers us the opportunity to test out more novel methods in R-UniMP model. Utilizing an innovative Positional Encoding for GNN [6] to enhance edge attention between node pairs and a weighted summation of all layers' node representations based on Generalized PageRank coefficients [7], we obtain better single-model scores than last year.

2 Methods

2.1 Positional Encoding for GNN

We adopt sinusoidal encoding [8] based on year of publication [4, 9] and learned network embedding [10] as absolute position encoding (PE) denoted by $\mathbf{Z}$. In general, the PE are simply sum together to origin node feature, which however, may not perform to their full potential. Inspired by PEG [6] which uses separate channels to update the original node features and positional features, we using the euclidean distance of PE as the edge weights between nodes. The distance-based edge weights can be integrated to R-UniMP [4] module for computing attention scores as following:

$$\alpha_{ij} = \mathrm{softmax}\Big(\vec{\mathrm{a}}(\mathbf{W}\mathrm{h_j}||\mathbf{W}\mathrm{h_j}) + \Phi\big(\|\mathrm{Z_i} - \mathrm{Z_j}\|\big)\Big) \tag{1}$$

where a_{ij} is the attention score between node i and j, $\mathbf{W} \in \mathbb{R}^{d \times d}$ are learnable parameters matrix, $\mathbf{Z}_i$ is the PE of node i, Φ is a MLP projection from $\mathbb{R} \to \mathbb{R}^d$. Unlike PEG [6], we still add the position encoding to origin node feature.

2.2 Generalized PageRank

To jointly improve the extraction of node features and topological information, we design to first learn node hidden features via multi-edge type in R-GAT [1] architecture, and then to propagate them through Generalized PageRank techniques (GPR) [7]. The GPR component associates each step of feature propagation with a weight, mitigating the feature-over-smoothing issue. Each node hidden state feature with GPR weight can be mathematically described as:

$$\gamma_k = \beta(1-\beta)^k, Z = \sum_{k=0}^{K} \gamma_k H^{(k)} \tag{2}$$

where γ_k and $H^{(k)}$ represents the k-th step GPR weight and node hidden state feature respectively. $\beta \in (0, 1)$ is a hyperparameter coefficient for PageRank, and Z denotes the final node embedding combined by all node hidden state features in K steps.

3 Experiments

3.1 Implementation Details

All our implementation can be found at `https://github.com/PaddlePaddle/PGL/tree/main/examples/NeurIPS2022-OGB-Challenge/MAG240M`. The code is implemented with Paddle Graph Learning (PGL) which is a graph neural network framework based on message passing paradigms and having highly optimized training speed. We train each of our models with four 8x Tesla A100 (80G), Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz and 1.5TB memories, using only 4 hours. For hyperparameters settings of best single model, we train our model by using AdamW[11] algorithm and batch-size of {512,1024} within 100 epochs, where initial learning rate is 0.0003. The ratio of model dropout, feature dropout and attention dropout is set to 0.4/0.1/0.1. The number of heads of R-UniMP is chosen in {1,2} and number of layers is fixed to 2. The β of GPR is set to 0.1 and the hidden channels of MLP in PEG is 128.

3.2 Best Single Model

Table 1 shows the result of single model in official validation set. During inference stage, we set the number of neighbors sampled to 200 in all layers. Row 1 is the base model[4] with papers' year position encode and metapath2vec (m2v)[10]. Attention dropout and Generalized PageRank can be increased by 0.13% and 0.2% from base model respectively. After using strategy of PEG[6], our best model in row 4 achieves 0.26% improvement.

Table 1: The Birth of Best Single Model

No.	Model	Official Validation
1	R-Unimp + year_pos + m2v	73.78%
2	1 + attn_drop	73.91%
3	2 + GPR	73.98%
4	3 + PEG (year_pos + m2v)	**74.04%**

3.3 Ensemble Models

Benefit from the optimization of training speed, we train 160 models with different hyperparameters within one week. In table 2, we show 5-fold performance before and after post-smoothing (P-S)[12, 4] of the seven strong models with different settings such as metapath2vec (**m2v**, metapath: autor-institution-author, autor-paper-author, and autor-institution-paper-institution-author), expanded metapath2vec (**p2p**, newly added metapath: paper-paper-author-paper), training node vector of p2p's metapath under fixed nodes' roberta feature (**roberta_p2p**), and the non-interactive GAT model without destination node in neighbors' attention score calculation[13] (**not_attn_dst**). Here, we use perform concatenation by **JK-Net**[14] on the output of all layer. The ensemble all models achieved MRR@10 scores of 77.87% and 75.70% in Validation and Test-Challenge datasets respectively

3.4 Speed Performance

In addition to optimizing the strategy of the GNN model itself, we also spend time optimizing the model training speed. Last year we participated in the MAG240M-LSC track of the 1st OGB Large-Scale Challenge, and achieved quite good results. The fly in the ointment was that it took too long to train a model at that time, at around 24 hours on 4 Tesla V100 cards, which seriously reduced the efficiency of model investigation. Therefore, this year we focus on speed optimization, and finally can train one model using only 4 hours, with 6x speedup.

Table 2: 5-Fold Validation Performance and Model Ensemble

model	Valid Result	P-S Result
GPR + PEG (m2v + year_pos)	77.21%	77.36%
PEG (m2v + year_pos)	77.18%	77.33%
GPR + PEG (p2p + year_pos)	77.16%	77.32%
GPR + PEG (m2v + year_pos) + JK(cat)	77.07%	77.23%
GPR + PEG (m2v + year_pos) + JK(cat)	77.04%	77.22%
GPR + PEG (roberta_p2p + year_pos)	77.10%	77.25%
GPR + PEG (m2v + year_pos) + no_attn_dst	77.15%	77.34%

A complete workflow for sampling based mini-batch GNN training usually contains 3 steps: mini-batch graph sampling, feature gathering and model training. In our previous implementation, since we store graph structures and features on CPU, we need to do graph sampling and feature gathering on CPU. After that, the sampled graph and gathered features will be transferred from CPU to GPU, leading the network communication to be a bottleneck for GNN training.

3.4.1 Graph Sampling

One way to speed up graph sampling is to use GPU sampling instead of CPU sampling. However, the entire graph structure is usually too large to store on GPU memory. Therefore, inspired by Quiver [15], we implement the UVA-Based(Unified Virtual Addressing Based) tensor technology, supporting graph structure tensor storing on CPU memory while sampling graph with GPU. In this way, we can achieve much faster graph sampling performance than CPU sampling. We did graph sampling experiment on Reddit dataset, and it shows that the GPU sampling speed can reach 32 to 140 times of the CPU sampling speed.

3.4.2 Feature gathering

After graph sampling, the next problem to be concerned is feature gathering. The node feature of MAG240M dataset is about 350G. Usually, we place the feature on CPU memory, and then pull the gathered feature to GPU for model training, which is quite time-consuming. A very intuitive idea is to put the features directly on GPU. To achieve this, we split graph feature from embedding dimension, shown in Figure 1.

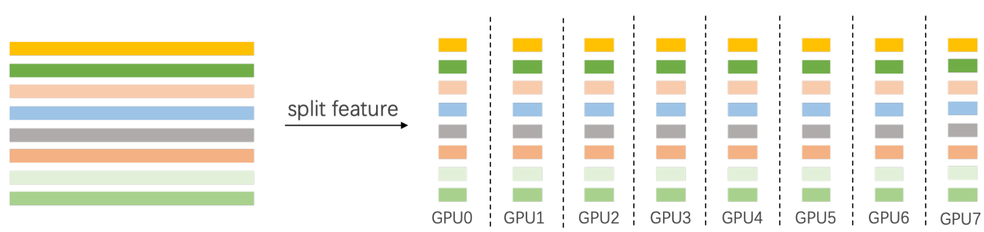

Figure 1: Split feature on embedding dimension. Suppose the shape of feature is [N, 768], then each GPU will store part of feature in the shape of [N, 96].

Suppose we have eight A100-80G GPU cards, each GPU can store 43.75 GB feature. Then we use NCCL library to help gather feature, utilizing the powerful bandwidth of NVLink to transfer data. As for A100-40G, we can use UVA-Based tensor technology to help store graph feature on CPU, but still gather feature using NCCL library. The experiments of training R-GAT model on MAG240M dataset shows that GPU feature mode can achieve fastest training speed compared with other modes. Besides, we also run the DGL example for comparison. It can be seen that our method can greatly improve the training speed [1].

[1]Recently we notice that NVIDIA team release a fast graph neural network training framework named WholeGraph [16], which can achieve a speed of 11.2 seconds per training epoch. WholeGraph's optimization

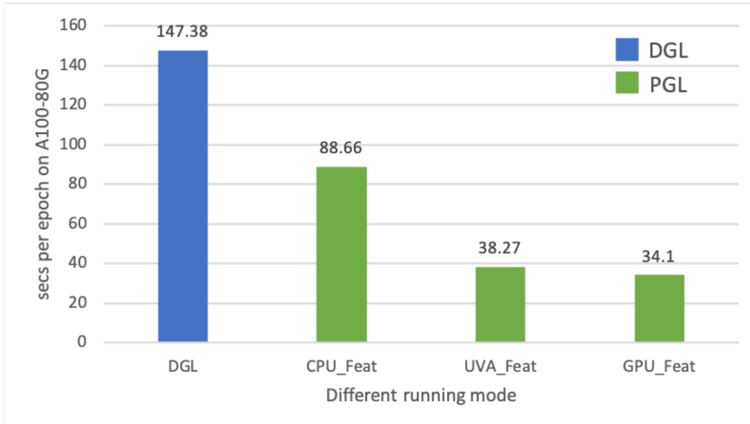

Figure 2: Training R-GAT model on MAG240M dataset. We run different modes of feature gathering, including CPU mode, UVA mode and GPU mode. In addition, we run DGL's example for comparison. Experiments are done on eight 80GB A100 cards and 1.5TB CPU memories.

References

[1] 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.

[2] 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.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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

[5] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[6] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022.

[7] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *International Conference on Learning Representations*, 2021.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the ACM Web Conference*, 2020.

[10] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144, 2017.

idea is similar to ours, except that they further adopt the acceleration of GPU direct access when gathering features.

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[12] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. Combining label propagation and simple models out-performs graph neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[13] 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.

[14] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5449–5458. PMLR, 2018.

[15] Quiver Team. Quiver: Pytorch library for fast and easy distributed graph learning. 2021.

[16] Dongxu Yang, Junhong Liu, Jiaxing Qi, and Junjie Lai. Wholegraph: A fast graph neural network training framework with multi-gpu distributed shared memory architecture. In *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 767–780. IEEE Computer Society, 2022.