Pooling Architecture Search for Graph Property Prediction in Open Graph Benchmark

Xu Wang
AutoGraph team (4Paradigm)
wangxu01@4paradigm.com

Huan Zhao
AutoGraph team (4Paradigm)
zhaohuan@4paradigm.com

Lanning Wei
AutoGraph team (4Paradigm)
weilanning18z@ict.ac.cn

Quanming Yao
AutoGraph team (EE Tsinghua)
qyaoaa@tsinghua.edu.cn

Abstract

In this techinical report, we present our solution for three OGB graph classification tasks ogbg-molhiv, ogbg-molpcba and ogbg-ppa.

1 Introduction

Graph is widely used in the abstraction of complex systems with interactive objects, such as social networks, knowledge graphs, molecular graphs, biological networks and so on. Machine learning on graph, especially deep learning, is an emerging field. Recently, great progress has been made in the theoretical and methodological research of graph learning, and good results have been produced in various applications.

As a real, large-scale and diverse benchmark data set for graph learning, OGB [1] provides a set of standardized evaluation criteria for graph machine learning model. Among them, ogbg molhiv (small) and ogbg molpcba (medium) are two molecular attribute prediction data sets with different sizes, which are adopted from the MoleculeNet [2], and all molecules are pre-processed with RDKit [3]. Each graph represents a molecule, where nodes represent atoms and edges represent chemical bonds. The task of both data sets is to predict the target molecular characteristics as accurately as possible, in which the molecular characteristics are labeled as binary labels, e.g, whether a molecule can inhibits HIV replication.

2 Method

In this section, we elaborate on the proposed novel framework PAS (Pooling Architecture Search) [4], which is based on NAS (Neural Architecture Search) to search for adaptive architectures for graph classification, consisting of the novel search space and the efficient search algorithm.

2.1 The Design of the Search Space

We use a 2-layer architecture backbone as an example to elaborate on the designed modules in the search space as shown in Figure 1, the detailed oprations are given in the following.

Aggregation Module. We add six widely used GNNs: GCN [5], GAT [6], GIN [7], MF [8], Transformer [9] and DeeperGCN [10], which denoted as *GCNConv*, *GATConv*, *GINConv*, *MFConv*, *TransformerConv* and *GenConv*. Besides, we incorporate the operation *MLP*, which applies a two-layer MLP (Multilayer Perceptrons) to update node embeddings without using the graph structure.

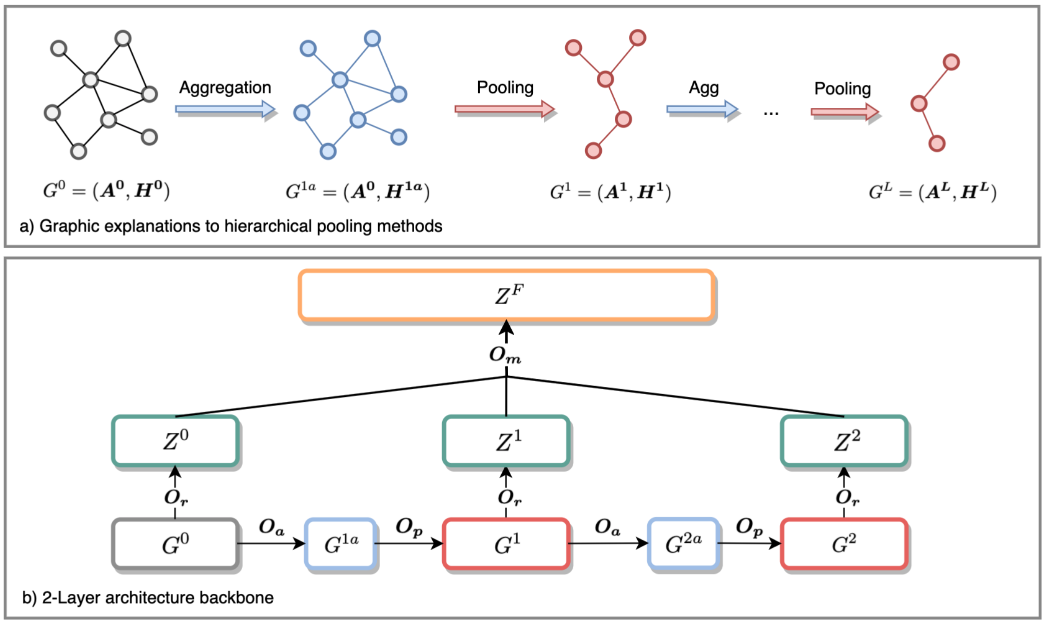
Aggregation
Pooling
Agg
...
Pooling
$G^0 = (\boldsymbol{A^0}, \boldsymbol{H^0})$
$G^{1a} = (\boldsymbol{A^0}, \boldsymbol{H^{1a}})$
$G^1 = (\boldsymbol{A^1}, \boldsymbol{H^1})$
$G^L = (\boldsymbol{A^L}, \boldsymbol{H^L})$
a) Graphic explanations to hierarchical pooling methods
Z^F
$\boldsymbol{O_m}$
Z^0
Z^1
Z^2
$\boldsymbol{O_r}$
$\boldsymbol{O_r}$
$\boldsymbol{O_r}$
G^0
$\boldsymbol{O_a}$
G^{1a}
$\boldsymbol{O_p}$
G^1
$\boldsymbol{O_a}$
G^{2a}
$\boldsymbol{O_p}$
G^2
b) 2-Layer architecture backbone

Figure 1: (a) Hierarchical methods use one aggregation and one pooling operation in each layer, which is responsible for update node embeddings and generate the coarse graph. (b) A 2-layer supernet as an example.

Pooling Module. In the original PAS, we select top-k nodes to form a course graph by calculating the score function. We incorporate 3 existing pooling operations *TOPKPOOL* [11], *SAGPOOL* [12] and *ASAP* [13] in our search space. However, for the two graph property prediction data sets of OGB, the average diameter is about 14. The pooling layer is slightly redundant for smaller graphs, and the result of adding pooling layer is not good after testing. Therefore, we delete the search of pooling layer here.

Readout Module. We provide 3 global pooling functions to obtain the graph representation vector: *GLOBAL_MEAN*, *GLOBAL_MAX* and *GLOBAL_SUM*, which generate the graph embedding for the final representation.

Merge Module. Motivated by JK-Networks [14] that intermediate layers help to fomulate expressive embeddings, we add 5 merge functions to incorporate the graph representations in each layer: LSTM, concatenation, max, mean and sum, which denoted as *M_LSTM*, *M_CONCAT*, *M_MAX*, *M_MEAN* and *M_SUM* in our search space.

2.2 Differentiable Search Algorithm

Continuous relaxion. As shown in Figure 1, we construct a supernet representing the search space for a 2-layer architecture backbone consisting of the four modules, where each node denotes the representations and each edge represents one operation from the corresponding sets $\boldsymbol{O} \in \{\boldsymbol{O_a}, \boldsymbol{O_r}, \boldsymbol{O_m}\}$. To make the search space continuous, the discrete selection of operation in each module is relaxed by a weighted summation of all possible operations as

$$\bar{o}(x) = \sum_{i=1}^{|\boldsymbol{O}|} \boldsymbol{c}_i \boldsymbol{o}_i(x), \tag{1}$$

where x represents the input representation of each module, and $\boldsymbol{c}_i$ denotes the weight of the i-th operation $\boldsymbol{o}_i$ in the set $\boldsymbol{O}$. It can be generated by a reparameterization trick as shown in

$$c_i = \frac{\exp((\log\boldsymbol{\alpha}_i + \boldsymbol{G}_i)/\lambda)}{\sum_{j=1}^{|\boldsymbol{O}|}\exp((\log\boldsymbol{\alpha}_i + \boldsymbol{G}_i)/\lambda)}, \tag{2}$$

where $\boldsymbol{\alpha}_i$ is the corresponding supernet parameter for c_i, $\boldsymbol{G}_i = -\log(-\log(\boldsymbol{U}_i))$ is the Gumble random variable, and $\boldsymbol{U}_i$ is a uniform random variable, λ is the temperature of softmax. It is also called Gumble-Softmax [14], which is designed to approximate discrete distribution in a differentiable manner and shown useful for supernet training in NAS.

For Aggregation, Readout and Merge Module in the supernet, we can generate the mixed results by constraining the node embedding dimension as shown in

$$\mathrm{H}^{la} = \sum_{i=1}^{|\boldsymbol{O}_{\boldsymbol{a}}|} \boldsymbol{c}_i^{al}\boldsymbol{o}_i(\mathrm{A}^{l-1}, \mathrm{H}^{l-1}), \tag{3}$$

$$\mathrm{z}^{F} = \sum_{i=1}^{|\boldsymbol{O}_{\boldsymbol{m}}|} \boldsymbol{c}_i^{m}\boldsymbol{o}_i(\mathrm{z}^0, \mathrm{z}^1, ..., \mathrm{z}^L), \tag{4}$$

where $\boldsymbol{c}_i^{al}$ denotes the weights of i-th aggregation operation, $\boldsymbol{c}_i^{m}$ denotes the i-th merge operation.

Optimization by gradient descent. Based on these mixed results of each module in the supernet, the prediction p can be calculated by a classifier. Finally, we calculate the cross-entropy loss on training data, denoted as $\mathcal{L}_{train}$, and optimize the expected performance of the architectures sampled with $p(\boldsymbol{C})$ as shown in

$$\min_{\boldsymbol{\alpha}, \mathrm{W}} \mathbb{E}_{\mathrm{C}\sim p_{\alpha}(\mathrm{C})}[\mathcal{L}_{train}(\mathrm{C}, \mathrm{W})]. \tag{5}$$

Since $\mathcal{L}_{train}$ is differentiable, and the Gumbel-Softmax relaxes the discrete architecture distribution into continuous and differentiable, we can optimize the parameters α and W with gradient descent.

2.3 KL Divergence Constraint

We add KL Divergence constraint loss to restrain the distributions of graph feature augmented from same graphs to be similar [15]. For two distribution p and q, the more common way to see KL divergence written is as follows: $D(p,q) = \sum_{i=1}^{N} p(x_i) \cdot \log\frac{p(x_i)}{q(x_i)}$.

Table 1: Hyper-parameter search space

Hyper-parameter	ogbg-molhiv	ogbg-molpcba	ogbg-ppa
learning rate	[5e-3, 1e-2, 3e-2, 5e-2, 1e-1]	[5e-4, 1e-3, 3e-3, 5e-3, 1e-2]	[5e-3, 1e-2, 3e-2, 5e-2, 1e-1]
batch size	[128, 256, 512]	[256, 512, 1024]	[128, 256, 512]
hidden size	[256, 512]	[512, 1024]	[256, 512]
dropout	[0.1, 0.2, 0.3]	[0.1, 0.2, 0.3]	[0.1, 0.2, 0.3]
gamma	[500, 700, 1000]		
virtual node	[True, False]	[True, False]	[True, False]

3 Experiments and Results

3.1 Tuned hyper-parameters

The hyperparametric search space of our method on three data sets is shown in the Table 1.

3.2 ogbg-molhiv

For ogbg-molhiv, as the dataset is relatively small, we follow the Neural FingerPrints [16] to get MorganFingerprint and MACCSFingerprint. Then we join train a PAS model and the FingerPrints model. We use the softmax aggregator with a temperature beta for two model above. Considering the small scale of the graph structure in the dataset, we cancel the search of pooling and merge operations. We also use heterogeneous interpolation process as a means of data enhancement, the result is showed in Table 2.

Table 2: Verification ROC-AUC on ogbg-molhiv dataset.

Dataset	Method	Test AUROC	Validation AUROC	Hardware	Parameters
ogbg-molhiv	PAS	0.8221 ± 0.0017	0.8178 ± 0.0031	RTX3090	26,706,952
ogbg-molhiv	PAS+FPs	0.8420 ± 0.0015	0.8238 ± 0.0028	RTX3090	26,706,953

3.3 ogbg-molpcba

For ogbg-molpcba, the graph structure in the dataset is still small, and we still cancel the search of pooling and merge operations, the result is showed in Table 3.

Table 3: Verification AP on ogbg-molpcba dataset.

Dataset	Method	Test AP	Validation AP	Hardware
ogbg-molpcba	PAS	0.3012 ± 0.0039	0.3151 ± 0.0047	RTX3090

3.4 ogbg-ppa

For ogbg-ppa, the scale of graph structure in the dataset has increased significantly. We use a complete PAS search framework, the result is showed in Table 4.

Table 4: Verification ACC on ogbg-molpcba dataset.

Dataset	Method	Test ACC	Validation ACC	Hardware
ogbg-ppa	PAS	0.7828 ± 0.0024	0.7523 ± 0.0028	RTX3090

4 Conclusion

In this report, we mainly applied the proposed PAS and combined with the method on the leaderboards to verify the effect on three datasets of OGB. In the future, we will further explore the effect of network structure on graph property prediction, including the application of our recent work F2GNN [17].

References

[1] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[2] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[3] Greg Landrum. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

[4] Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. Pooling architecture search for graph classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2091–2100, 2021.

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

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

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

[8] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*, 2015.

[9] 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.

[10] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[11] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[12] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743. PMLR, 2019.

[13] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

[14] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[15] Zhuoning Yuan, Yan Yan, Milan Sonka, and Tianbao Yang. Robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification. *arXiv preprint arXiv:2012.03173*, 2020.

[16] Shanzhuo Zhang. Molecule representation learning by leveraging chemical information. `https://github.com/PaddlePaddle/PaddleHelix/blob/dev/competition/ogbg_molhiv/Molecule_Representation_Learning_by_Leveraging_Chemical_Information.pdf`.

[17] Lanning Wei, Huan Zhao, and Zhiqiang He. Designing the topology of graph neural networks: A novel feature fusion perspective. *arXiv preprint arXiv:2112.14531*, 2021.