NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search

Yijian Qin, Ziwei Zhang, Xin Wang[*] **Zeyang Zhang, Wenwu Zhu**[*]
Department of Computer Science and Technology
Tsinghua University
Beijing, China 100084
qinyj19@mails.tsinghua.edu.cn, {zwzhang,xin_wang}@tsinghua.edu.cn
zy-zhang20@mails.tsinghua.edu.cn, wwzhu@tsinghua.edu.cn

Abstract

Graph neural architecture search (GraphNAS) has recently aroused considerable attention in both academia and industry. However, two key challenges seriously hinder the further research of GraphNAS. First, since there is no consensus for the experimental setting, the empirical results in different research papers are often not comparable and even not reproducible, leading to unfair comparisons. Secondly, GraphNAS often needs extensive computations, which makes it highly inefficient and inaccessible to researchers without access to large-scale computation. To solve these challenges, we propose NAS-Bench-Graph, a tailored benchmark that supports unified, reproducible, and efficient evaluations for GraphNAS. Specifically, we construct a unified, expressive yet compact search space, covering 26,206 unique graph neural network (GNN) architectures and propose a principled evaluation protocol. To avoid unnecessary repetitive training, we have trained and evaluated all of these architectures on nine representative graph datasets, recording detailed metrics including train, validation, and test performance in each epoch, the latency, the number of parameters, etc. Based on our proposed benchmark, the performance of GNN architectures can be directly obtained by a look-up table without any further computation, which enables fair, fully reproducible, and efficient comparisons. To demonstrate its usage, we make in-depth analyses of our proposed NAS-Bench-Graph, revealing several interesting findings for GraphNAS. We also showcase how the benchmark can be easily compatible with GraphNAS open libraries such as AutoGL and NNI. To the best of our knowledge, our work is the first benchmark for graph neural architecture search.

1 Introduction

With the prevalence of graph data, graph machine learning [77], such as graph neural networks (GNNs) [83, 67], has been widely adopted in diverse tasks, including recommendation systems [66], bioinformatics [55], urban computing [24], physical simulations [54], combinatorial optimization [2], etc. Graph neural architecture search (GraphNAS), aiming to automatically discover the optimal GNN architecture for a given graph dataset and task, is at the front of graph machine learning research and has drawn increasing attention in the past few years [78].

Despite the progress in GraphNAS research, there exist two key challenges that seriously hinder the further development of GraphNAS:

---

[*]Corresponding Authors.

- The experimental settings, such as dataset splits, hyper-parameter settings, and evaluation protocols differ greatly from paper to paper. As a result, the experimental results cannot be guaranteed comparable and reproducible, making fair comparisons of different methods extremely difficult.

- GraphNAS often requires extensive computations and therefore is highly inefficient, especially for large-scale graphs. Besides, the computational bottleneck makes GraphNAS research inaccessible to those without abundant computing resources.

Similar challenges have arisen in other domains of NAS research [86], which gives birth to the idea of tabular NAS Benchmarks [72, 12, 27, 75]. Tabular NAS benchmarks provide pre-computed evaluations for all possible architectures in the search space by a table lookup. These benchmarks dramatically boost NAS research, for example, by speeding up the experiments since no architecture training is needed during the search procedure and creating fair comparisons of different NAS algorithms [35]. Inspired by the success of tabular NAS benchmarks and to solve the challenges of GraphNAS, we propose NAS-Bench-Graph[2], the first tabular NAS benchmark that supports unified, reproducible, and efficient evaluations for GraphNAS. Specifically, we first construct a unified GraphNAS search space by formulating the macro space of message-passing as a constrained directed acyclic graph and carefully choose operations from seven GNN candidates. Our proposed search space is expressive yet compact, resulting in 26,206 unique architectures and covering many representative GNNs. We further propose a principled protocol for dataset splits, choosing hyper-parameters, and evaluations. We have trained and evaluated all of these architectures on nine representative graph datasets with different sizes and application domains and recorded detailed metrics during the training and testing process. All the code and evaluation results have been open-sourced. Therefore, our proposed benchmark can enable fair, fully reproducible, and efficient comparisons for different GraphNAS methods.

To explore our proposed NAS-Bench-Graph, we make in-depth analyses from four perspectives with several interesting observations. First, the performance distribution shows that though reasonably effective architectures are common, architectures with extremely good results are rare, and these powerful architectures have diverse efficiencies, as measured by the model latency. Therefore, how to find architectures with both high efficiency and effectiveness is challenging. Second, the architecture distribution suggests that different graph datasets differ greatly in the macro space and operation choices. The cross-datasets correlations further suggest that different graph datasets exhibit complicated patterns and simply transferring the best-performing architectures from similar graph datasets cannot lead to the optimal result. Lastly, detailed architecture explorations demonstrate that architecture space exhibits certain degrees of smoothness, which supports the mutation process in evolutionary search strategies, and deeper parts of architectures are more influential than lower parts, which may inspire more advanced reinforcement learning based search strategies.

To demonstrate the usage of NAS-Bench-Graph, we have integrated it with two representative open libraries: AutoGL [18], the first dedicated library for GraphNAS, and NNI[3], a widely adopted library for general NAS. Experiments demonstrate that NAS-Bench-Graph can be easily compatible with different search strategies including random search, reinforcement learning based methods, and evolutionary algorithms.

Our contributions are summarized as follows.

- We propose NAS-Bench-Graph, a tailored GraphNAS benchmark that enables fair, fully reproducible, and efficient empirical comparisons for GraphNAS research. We are the first to study benchmarking GraphNAS research to the best of our knowledge.

- We have trained and evaluated all GNN architectures in our tailored search space on nine common graph datasets with a unified and principled evaluation protocol. Based on our proposed benchmark, the performance of architectures can be directly obtained without repetitive training.

- We make in-depth analyses for our proposed benchmark and showcase how it can be easily compatible with GraphNAS open libraries such as AutoGL and NNI.

---

[2]`https://github.com/THUMNLab/NAS-Bench-Graph`
[3]`https://github.com/microsoft/nni`

2   Related Works

2.1   Graph Neural Architecture Search

GNNs have shown impressive performance for graph machine learning in the past few years [25, 1, 59, 69, 68, 30, 31, 32, 39, 37, 38]. However, the existing GNNs are manually designed, which require expert knowledge, are labor-intensive, and unadaptable to changes in graph datasets and tasks. Motivated by these problems, automated graph learning has drawn increasing attention in the past few years, including hyper-parameter optimization on graphs [57, 61], and GraphNAS [16, 84, 48, 18, 82, 23, 10, 73, 22, 62, 64, 5, 17]. GraphNAS aims to automatically discover the optimal GNN architectures. Similar to general NAS [14], GraphNAS can be categorized based on the search space, the search strategy, and the performance estimation strategy [78]. For the search space, both micro [16, 80, 34] and macro [63, 15] spaces for the message-passing functions in GNNs are studied, as well as other functions such as pooling [23, 65], heterogeneous graphs [10], and spatial-temporal graphs [45]. Search strategies include reinforcement learning based methods [16, 84, 40], evolutionary algorithms [44, 53, 18], and differentiable methods [82, 79, 81]. Performance estimation strategies can be generally divided into training from scratch [16], hand-designed weight-sharing mechanisms [84, 8], using supernets [48, 6, 18, 49], and prediction-based methods [56]. Despite these progresses, how to properly evaluate and compare GraphNAS methods receives less attention. Our proposed benchmark can support fair and efficient comparisons of different GraphNAS methods as well as motivating new GraphNAS research.

2.2   NAS Benchmarks

Since the introduction of NAS-Bench-101 [72], many different benchmarks have been introduced for NAS. However, most previous works focus on computer vision tasks such as NAS-Bench-101 [72], NAS-Bench-201 [12], NATS-Bench [11], NAS-Bench-1shot1 [74], Surr-NAS-Bench [75], HW-NAS-Bench [28], NAS-HPO-Bench-II [20], TransNAS-Bench-101 [13], NAS-Bench-Zero [7], NAS-Bench-x11 [70], and NAS-Bench-360 [58]. Some recent benchmarks also study tabular data (NAS-HPO-Bench [26]), natural language processing (NAS-Bench-NLP [27] and NAS-Bench-x11 [70]), acoustics (NAS-Bench-ASR [41]), and sequence (NAS-Bench-360 [58]). We draw inspiration from these benchmarks and propose the first tailored NAS benchmark for graphs. We provide more comparisons with the existing NAS benchmarks in Appendix D.2.

3   Benchmark Design

In this section, we describe our design for the NAS benchmark construction, including the search space (Section 3.1), the datasets used (Section 3.2), and the experiment settings (Section 3.3). We provide some preliminaries of GNNs and GraphNAS in Appendix D.1.

3.1   Search Space Design

To balance the effectiveness and efficiency, we design a tractable yet expressive search space. Specifically, we consider the macro search space of GNN architectures as a directed acyclic graph (DAG) to formulate the computation[4], i.e., each computing node indicates a representation of vertices and each edge indicates an operation. Concretely, the DAG contains six nodes (including the input and output node) and we constrain that each intermediate node has only one incoming edge. The resulted DAG has 9 choices, as illustrated in Fig 1. Those intermediate nodes without successor nodes are connected to the output node by concatenation. Besides this macro space, we also consider optional fully-connected pre-process and post-process layers as GraphGym [73] and PasCa [76]. Notice that to avoid exploding the search space, we consider the numbers of pre-process and post-process layers as hyper-parameters, which will be discussed in Section 3.3. Finally, we adopt a task-specific fully connected layer to obtain the model prediction.

For the candidate operations, we consider 7 most widely adopted GNN layers: GCN [25], GAT [59], GraphSAGE [19], GIN [68], ChebNet [9], ARMA [4], and k-GNN [43]. Besides, we also consider Identity to support residual connection and fully connected layer that does not use graph structures.

---

[4]To distinguish the computation graph and graph data, we refer to the computation graph using nodes and edges, and the graph data using vertices and links.
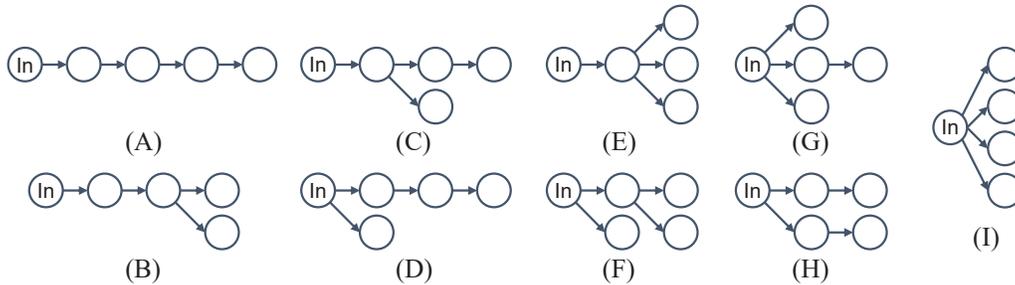
Figure 1: An illustration of the 9 different choices of our macro search space. Each node indicates a representation of vertices and each edge indicates an operation. We omit the output node for clarity.

Table 1: The statistics of the adopted datasets.

Dataset	#Vertices	#Links	#Features	#Classes	Metric
Cora	2,708	5,429	1,433	7	Accuracy
CiteSeer	3,327	4,732	3,703	6	Accuracy
PubMed	19,717	44,338	500	3	Accuracy
Coauthor-CS	18,333	81,894	6,805	15	Accuracy
Coauthor-Physics	34,493	247,962	8,415	5	Accuracy
Amazon-Photo	7,487	119,043	745	8	Accuracy
Amazon-Computers	13,381	245,778	767	10	Accuracy
ogbn-arxiv	169,343	1,166,243	128	40	Accuracy
ogbn-proteins	132,534	39,561,252	8	112	ROC-AUC

In summary, our designed search space contains 26,206 different architectures (after removing isomorphic structures, i.e., architectures that appear differently but have the same functionality, see Appendix A.5), which covers many representative GNN variants, including methods mentioned above as well as more advanced architectures such as JK-Net [69], residual- and dense-like GNNs [29].

3.2 Datasets

We adopt **nine** publicly available datasets commonly used in GraphNAS: Cora, CiteSeer, and PubMed [51], Coauthor-CS, Coauthor-Physics, Amazon-Photo, and Amazon-Computer [52], ogbn-arXiv and ogbn-proteins [21]. The statistics and evaluation metrics of the datasets are summarized in Table 1. These datasets cover different sizes from thousands of vertices and links to millions of links, and various application domains including citation graphs, e-commerce graphs, and protein graphs. More details about the datasets are provided in Appendix A.1.

The dataset splits are as follows. For Cora, CiteSeer, and PubMed, we use the public semi-supervised setting by [71], i.e., 20 nodes per class for training and 500 nodes for validation. For two Amazon and two Coauthor datasets, we follow [52] and randomly split the train/validation/test set in a semi-supervised setting, i.e., 20 nodes per class for training, 30 nodes per class for validation, and the rest for testing. For ogbn-arXiv and ogbn-proteins, we follow the official splits of the dataset.

For ogbn-proteins, we find through preliminary studies that using GIN and k-GNN operations consistently makes the model parameters converge to explosion and therefore results in meaningless results. Besides, GAT and ChebNet will result in out-of-memory errors for our largest GPUs with 32GB of memories. Therefore, to avoid wasting computational resources, we restrict the candidate operations for ogbn-proteins to be GCN, ARMA, GraphSAGE, Identity and fully connected layer. After such changes, there are 2,021 feasible architectures for ogbn-proteins.

3.3 Experimental Setting

Hyper-parameters To ensure fair and reproducible comparisons, we propose a unified evaluation protocol. Specifically, we consider the following hyper-parameters with tailored ranges:

- Number of pre-process layers: 0 or 1.
- Number of post-process layers: 0 or 1.
- Dimensionality of hidden units: 64, 128, or 256.
- Dropout rate: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8.
- Optimizer: SGD or Adam.
- Learning Rate (LR): 0.5, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001.
- Weight Decay: 0 or 0.0005.
- Number of training epochs: 200, 300, 400, 500.

For each dataset, we fix the hyper-parameters for all architectures to ensure a fair comparison. Notice that jointly enumerating architectures and hyper-parameters will result in billions of architecture hyper-parameter pairs and is infeasible in practice. Therefore, we first optimize the hyper-parameters to a proper value which can accommodate different GNN architectures, and then focus on the GNN architectures. Specifically, we adopt 30 GNN architectures from our search space as "anchors" and adopt random search for hyper-parameter optimization [3]. The 30 anchor architectures are composed of 20 randomly selected architectures from our search space and 10 classic GNN architectures including GCN, GAT, GIN, GraphSAGE, and ARMA with 2 and 3 layers. We optimize the hyper-parameters by maximizing the average performance of the anchor architectures. The detailed selected hyper-parameters for each dataset are shown in the Appendix A.2.

Metrics   During training each architecture, we record the following metric covering both model effectiveness and efficiency: train/validation/test loss value and evaluation metric at each epoch, the model latency, and the number of parameters. The hardware and software configurations are provided in Appendix A.3. Besides, all experiments are repeated three times with different random seeds. The total time cost of creating our benchmark is approximately 8,000 GPU hours (see Appendix A.4).

4 Analyses

In this section, we carry out empirical analyses to gain insights for our proposed benchmark. All the following analyses are based on the average performances of three random seeds.

4.1 Performance Distribution

We first visualize the distribution of performances, including the accuracy, the latency, and the numbers of parameters, of all architectures in Figure 2. We make several interesting observations. For the effectiveness aspect, many architectures can obtain a reasonably good result, but architectures with exceptionally strong results are still rare. On the other hand, the latency and the numbers of parameters of architectures differ greatly. Since both model effectiveness and efficiency are critical for GraphNAS, we mark the Pareto-optimal with respect to accuracy and latency in the figure. The results show that, even for top-ranking architectures with similar accuracy, their latency varies greatly. In addition, we observe that the number of parameters and the latency are positively correlated in general (details are shown in the Appendix B.1).

4.2 Architecture Distribution

As introduced in Section 3.1, our proposed search space mainly consists of the macro space (i.e., the DAG) and candidate GNN operations. To gain insights of how different macro space and GNN operation choices contribute to the model effectiveness, we select the top 5% architectures on each dataset and plot the frequency of the macro search space and operation choices. The results are shown in Figure 3. We make the following observations.

First, there exist significant differences in the macro space choices for different datasets, indicating that different GNN architectures suit different graph data. For example, Cora, CiteSeer and PubMed tend to select a 2-layer DAG, i.e., (E), (F), (G), and (H) (please refer to Figure 1 for the detailed DAGs). PudMed and CS also prefer the 1-layer DAG (I), which is hardly selected in other datasets. Physics, Photo, and Computers show more balanced distributions on the macro space. ogbn-arXiv

(a) Cora   (b) CiteSeer   (c) PubMed

(d) Coauthor-CS   (e) Coauthor-Physics   (f) Amazon-Photo

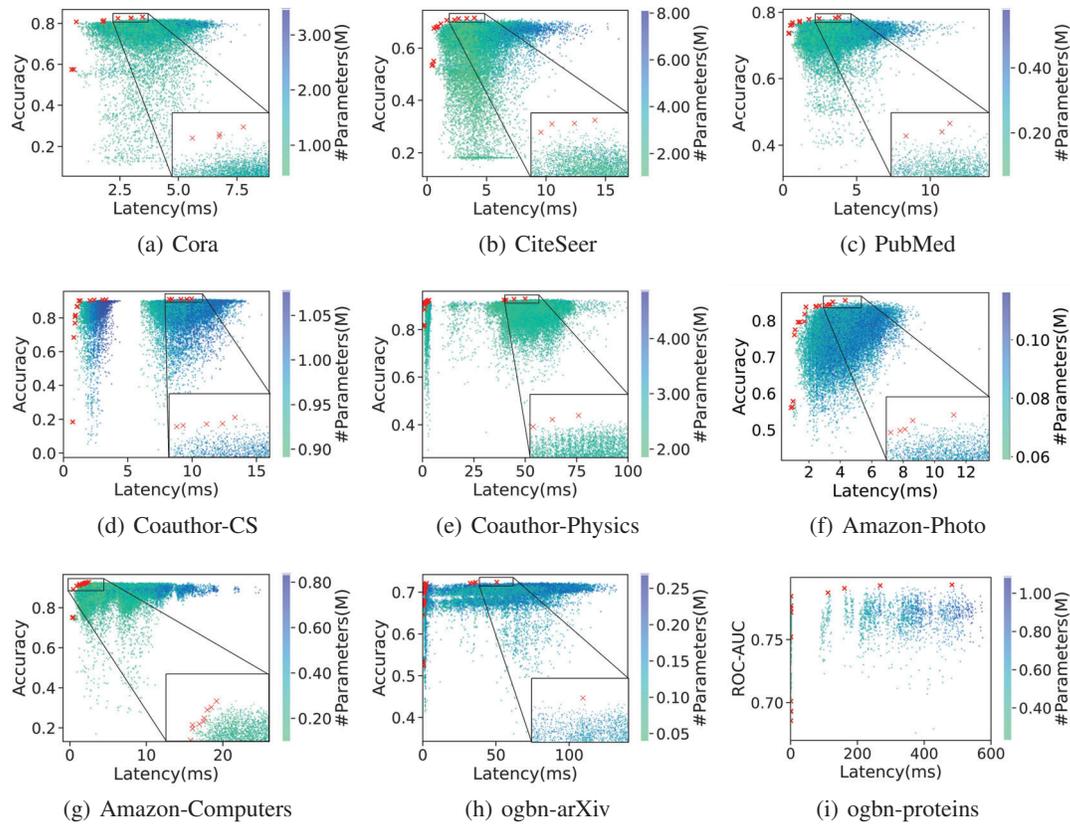(g) Amazon-Computers   (h) ogbn-arXiv   (i) ogbn-proteins

Figure 2: The distribution of accuracy, latency, and the numbers of parameters of all architectures. The Pareto-optimal architectures w.r.t. accuracy and latency are marked with red crosses.

(a) The frequency of macro space choices

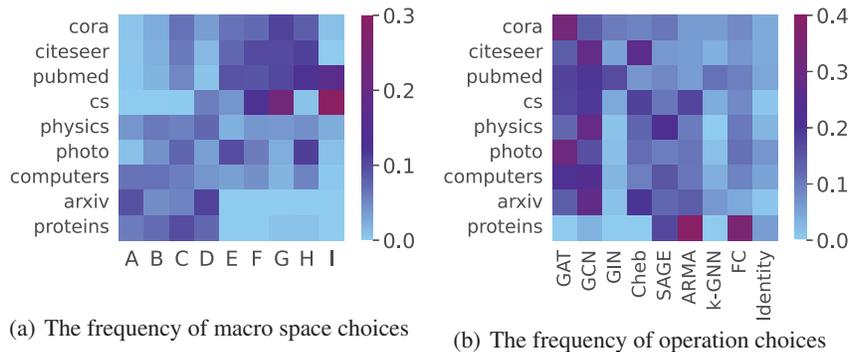(b) The frequency of operation choices

Figure 3: The frequency of the macro space and operation choices in the top 5% architectures of different datasets. Please refer to Figure 1 for the macro space choices.

and ogbn-proteins select deeper architectures more frequently, e.g., the 4-layer DAG (A) and the 3-layer DAG (B), (C), and (D).

As for the operation distribution, different datasets show more similar patterns. GCN and GAT are selected most frequently in almost all datasets. Surprisingly, even though GIN and k-GNN are shown to be theoretically more expressive in terms of the Weifeiler-Lehman test, they are only selected in the relatively small datasets, i.e., Cora, CiteSeer, and PubMed. A plausible reason is that GIN and k-GNN adopt the summation aggregation function in the GNN layers, which is not suitable for node-level tasks in large-scale graphs. Moreover, different from NAS in computer vision where

(a) Pearson correlation coefficient   (b) Kendall's rank correlation   (c) The overlapping ratio of the top 5% architectures
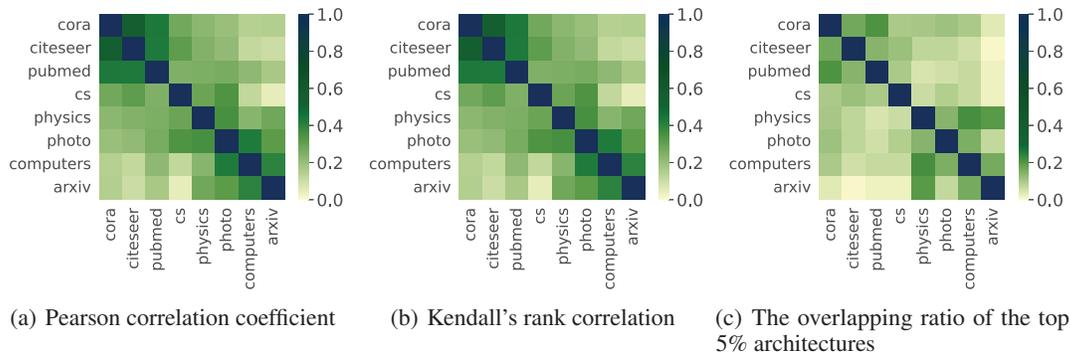
Figure 4: The architecture performance correlation across different datasets using three metrics.

Identity operation plays an important role in improving model performance [60], Identity is hardly chosen in any graph datasets. More analyses about the frequency of operations with different depths can be found in the Appendix B.2.

4.3 Cross-datasets Correlations

To measure how architectures perform across different datasets, we calculate the performance correlation of all architectures on dataset pairs as You et al. [73]. Specifically, we adopt three metrics: Pearson correlation coefficient, Kendall rank correlation, and the overlapping ratio of the top 5% architectures, i.e., if there are N architectures belonging to the top 5% of both two different datasets in terms of accuracy, then the overlapping ratio is $N/(26, 206 \times 5\%)$. We show the results in Figure 4.

We can observe that the correlation matrix has roughly block structures, indicating that there exist groups of datasets in which architectures share more correlations. For example, Cora, CiteSeer and PubMed generally show strong correlations. The correlations are also relatively high between Physics, Photo, Computers, and ogbn-arXiv. Notice that even for these datasets, only the Pearson correlation coefficient and Kendall rank correlation have large values, while the overlapping ratio of the top 5% architectures is considerably lower (e.g., no larger than 0.3). Since we usually aim to discover best architectures, the results indicate that best-performing architectures in different graph datasets exhibit complicated patterns, and directly transferring the best architecture from a similar dataset as You et al. [73] may not lead to the optimal result. More analyses about the transferability of the optimal architectures on different datasets can be found in the Appendix B.3.

4.4 Detailed Explorations in Architectures

Since enumerating all possible architectures to find the best-performing one is infeasible in practice, NAS search strategies inevitably need prior assumptions on the architectures. These explicit or implicit assumptions largely determine the effectiveness of the NAS methods. Next, we explore the architectures in details to verify some common assumptions.

Evolutionary Algorithm is one of the earliest adopted optimization methods for NAS [50]. In the mutation process of evolutionary algorithms, i.e., randomly changing choices in the search space, a common assumption is that similar architectures have relatively similar performance so that smoothly mutating architectures is feasible [84]. To verify this assumption, we calculate the performance difference between architectures with different number of mutations together with the average performance difference between randomly chosen architectures as a reference line. The results are shown in Figure 5. We can observe that the performance difference between mutated architectures is considerably smaller than two random architecture, verifying the smoothness assumption in mutations. Besides, we observe that the performance difference increases as the number of mutations in general and changing operation choices usually leads to smaller differences than changing the macro space choices. These observations may inspire further research of evolution algorithms for GraphNAS.

Reinforcement Learning (RL) is also widely adopted in NAS [87, 47, 85]. In RL-based NAS, architectures are usually generated by a Markov Decision Process, i.e., deciding the architectures
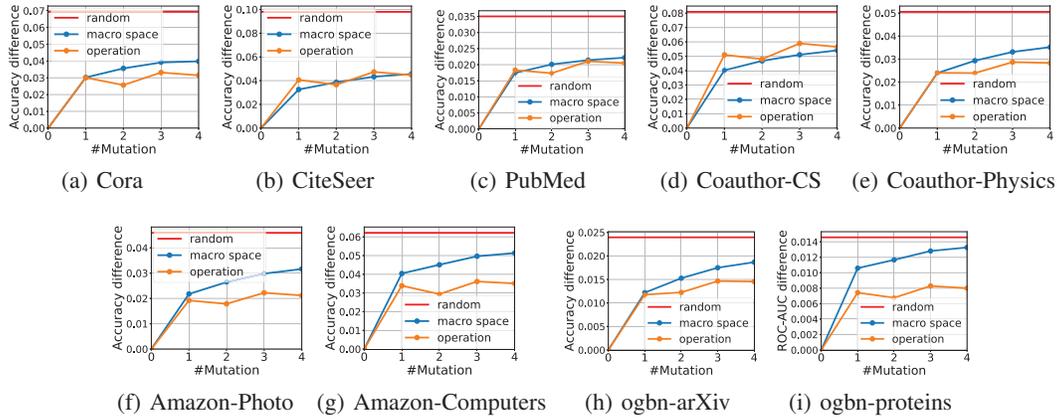
Figure 5: The performance difference between architectures with different number of mutations. The red lines indicate the average performance difference between two random architectures.
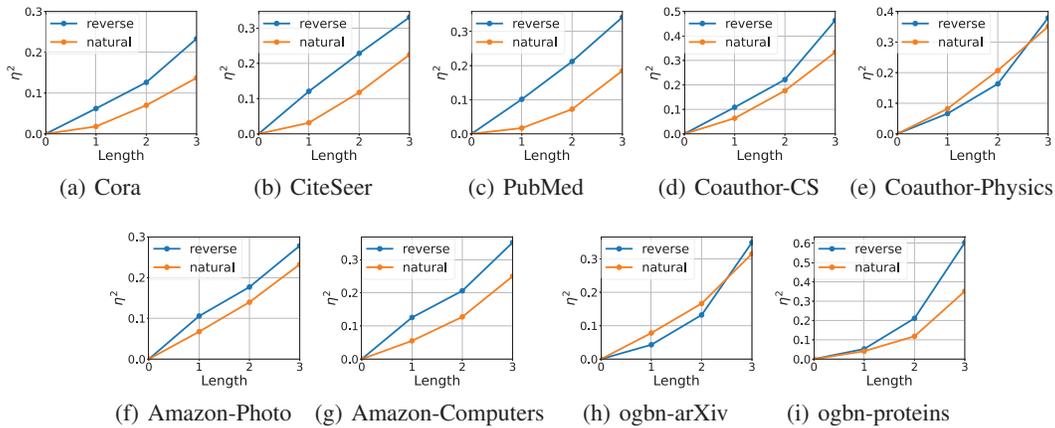

Figure 6: η^2 for the architecture performance with different lengths of prefixes/suffixes as groups. Larger η^2 means that the groups can better explain the variance and the corresponding architectures choices are more important for the performance.

using a sequential order [87]. Most GraphNAS methods simply assume a natural order, i.e., generating architectures from lower parts to deeper parts [16]. To gain insights for such methods, we analyze the importance of different positions of architectures. Specifically, we group architectures using their prefixes (i.e., lower-parts choices) and their suffixes (i.e., deeper-parts choices), corresponding to generating architectures using the natural order and the reserve order. Then, we adopt the following metric from analysis of variance [46]:

$$\eta^2 = \frac{SSB}{SST} = \frac{\sum_{k=1}^{K} n_k(\mu_k - \overline{\mu})^2}{\sum_{i=1}^{N}(x_i - \overline{\mu})^2},$$

where SSB and SST are the between-group variations and the total variation, N is the number of architectures, x_i is the accuracy of an architecture i, $\overline{\mu}$ is the mean accuracy, K is the number of groups, n_k is the number of architectures in the group k, and μ_k is the mean performance of architectures in the group k. In short, a larger η^2 means the groups can better explain the variances in the samples so that the factors for the group (i.e., architecture choices) are more important. The results of varying the length of the prefixes and suffixed are shown in Figure 6. Somewhat surprisingly, in most datasets, η^2 in the reverse order is larger than that in the natural order, indicating that deeper parts of the architecture are more influential for the performance than the lower parts. With the length of prefixes and suffixes growing, η^2 grows significantly. The results indicate that the natural order

Table 2: The performance of NAS methods in AutoGL and NNI using NAS-Bench-Graph. The best performance for each dataset is marked in bold. We also show the performance of the top 5% architecture (i.e., 20-quantiles) as a reference line. The results are averaged over five experiments with different random seeds and the standard errors are shown in the bottom right.

Library	Method	Cora	CiteSeer	PubMed	CS	Physics	Photo	Computers	arXiv	proteins
AutoGL	GNAS	$82.04_{0.17}$	$\mathbf{70.89}_{0.16}$	$77.79_{0.02}$	$90.97_{0.06}$	$92.43_{0.04}$	$92.43_{0.03}$	$84.74_{0.20}$	$72.00_{0.02}$	$\mathbf{78.71}_{0.11}$
	Auto-GNN	$81.80_{0.00}$	$70.76_{0.12}$	$77.69_{0.16}$	$\mathbf{91.04}_{0.04}$	$92.42_{0.16}$	$92.38_{0.01}$	$84.53_{0.14}$	$\mathbf{72.13}_{0.03}$	$78.54_{0.30}$
NNI	Random	$82.09_{0.08}$	$70.49_{0.08}$	$77.91_{0.07}$	$90.93_{0.07}$	$92.35_{0.05}$	$\mathbf{92.44}_{0.02}$	$84.78_{0.14}$	$72.04_{0.05}$	$78.32_{0.14}$
	EA	$81.85_{0.20}$	$70.48_{0.12}$	$\mathbf{77.96}_{0.12}$	$90.60_{0.07}$	$92.22_{0.08}$	$92.43_{0.02}$	$84.29_{0.29}$	$71.91_{0.06}$	$77.93_{0.21}$
	RL	$\mathbf{82.27}_{0.21}$	$70.66_{0.12}$	$\mathbf{77.96}_{0.09}$	$90.98_{0.01}$	$\mathbf{92.48}_{0.03}$	$92.42_{0.06}$	$\mathbf{84.90}_{0.19}$	$\mathbf{72.13}_{0.05}$	$78.52_{0.18}$
The top 5%		80.63	69.07	76.60	90.01	91.67	91.57	82.77	71.69	78.37

to convert architectures into sequences may not be the optimal solution for GraphNAS and more research could be explored in this direction.

Differentiable Method. DARTS [36] in one of the most popular differentiable NAS algorithms. We adopt DARTS on our proposed search space as the search algorithm, and we plot the average weight of different operations Figure 7. Compared to the weights distribution of the top 5% architectures shown in Figure 3(b), we can find that in some datasets, DARTS dispatches the largest weight to the most frequent operations in best-performing architecture, e.g., GAT for Cora and GCN for Amazon-Computers, indicating the effectiveness of DARTS in searching best-performing architectures.
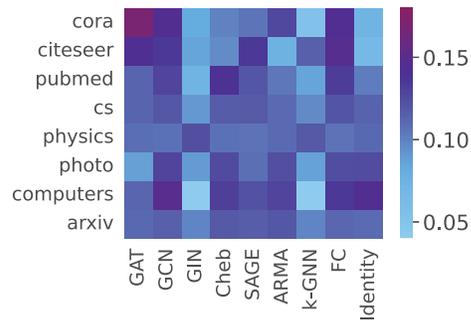

Figure 7: The average weight of different operations obtained using DARTS as the search algorithm

5 Example Usages

In this section, we showcase the usage of NAS-Bench-Graph with existing open libraries including AutoGL [18] and NNI [42] (detailed example codes are provided in the Appendix C). Specifically, we run two NAS algorithms through AutoGL: GNAS [16] and Auto-GNN [84]. For NNI, we adopt Random Search [33], Evolutionary Algorithm (EA), and Policy-based Reinforcement Learning (RL). To ensure fair comparisons, we only let each algorithm access the performances of 2% of the all architectures in the search space. We report the results in Table 2. We also show the performances of the top 5% architecture, i.e, the 20-quantiles of each dataset in the table.

From the results, we can observe that all algorithms outperform the top 5% performance, indicating that they can learn informative patterns in NAS-Bench-Graph. However, no algorithm can consistently win on all datasets. Surprisingly, Random Search is still a strong baseline when compared with other methods and even performs the best on two datasets, partially corroborating the findings in [33] for general NAS. The results indicate that further research on GraphNAS is still urgently needed.

To investigate the learning of different GraphNAS methods, we plot the curves of the optimal performance with respect to the number of architectures. The results are shown in Figure 8. We can find that different algorithms behave differently. For example, EA and AGNN show a few "jumps", i.e., the performance largely increases, while RL shows more smooth increasing patterns. Taking closer looks at the learning curve may provide inspirations for developing new algorithms for GraphNAS.

6 Conclusion and Future Work

In this paper, we propose NAS-Bench-Graph, the first tailored NAS benchmark for graph neural networks. We have trained all 26,206 GNN architectures in our designed search space on nine representative graph datasets with a unified evaluation protocol. NAS-Bench-Graph can support
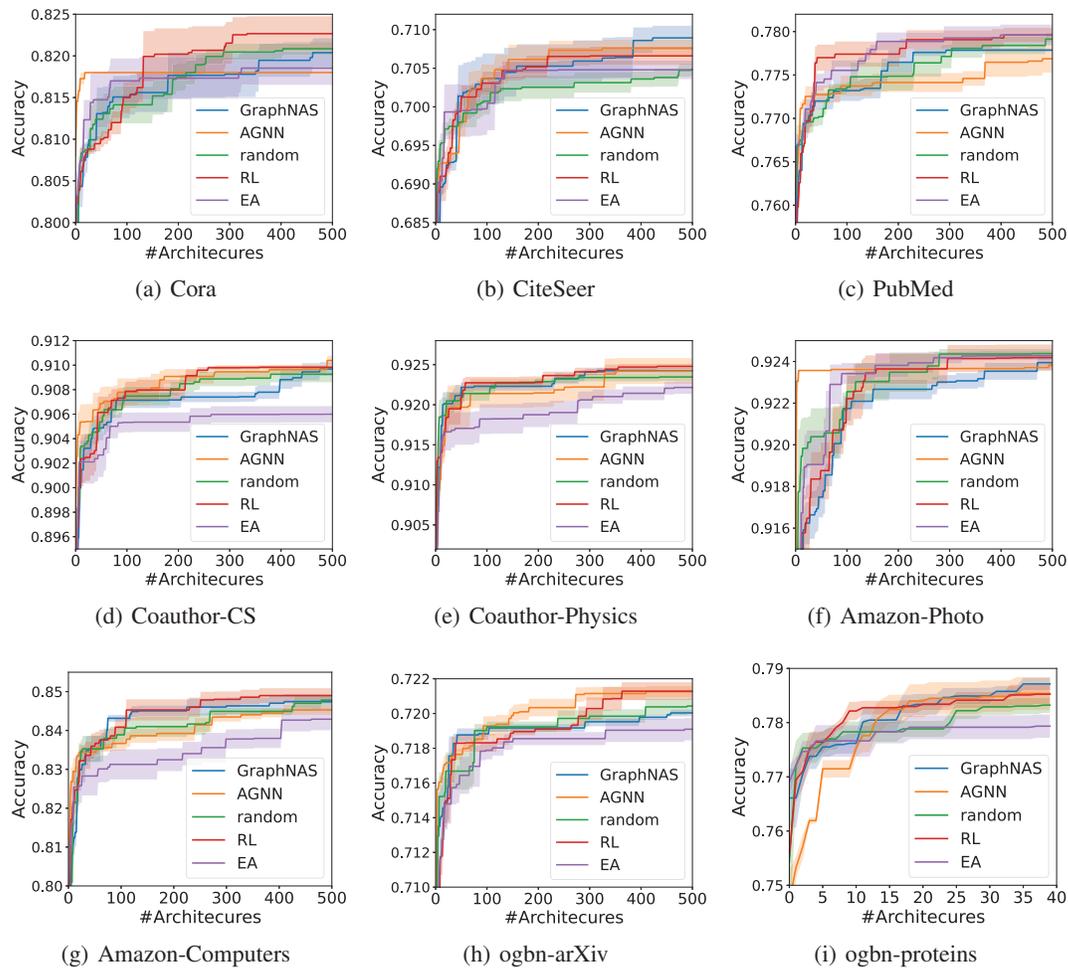
Figure 8: The learning curve of the optimal performance with respect to the number of searched architectures. All results are averaged over five experiments with different random seeds. The standard errors are shown in the background.

unified, reproducible, and efficient evaluations for GraphNAS. We also provide in-depth analyses for NAS-Bench-Graph and show how it can be easily compatible with the existing GraphNAS libraries.

Since constructing tabular NAS benchmarks consumes extensive computational resources, our proposed NAS-Bench-Graph has a relatively limited search space (i.e., 26,206 unique GNN architectures). A possible direction is constructing surrogate benchmarks for GraphNAS to allow larger search spaces. We also plan to extend our proposed benchmark to other graph tasks besides node classification, such as link prediction and graph classification.

Acknowledgments and Disclosure of Funding

References

[1] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[2] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[5] Jie Cai, Xin Wang, Chaoyu Guan, Yateng Tang, Jin Xu, Bin Zhong, and Wenwu Zhu. Multi-modal continual graph learning with neural architecture search. In *Proceedings of the ACM Web Conference 2022*, pages 1292–1300, 2022.

[6] Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6657–6666, 2021.

[7] Hanlin Chen, Ming Lin, Xiuyu Sun, and Hao Li. Nas-bench-zero: A large scale dataset for understanding zero-shot neural architecture search. In *OpenReview*, 2021.

[8] Jiamin Chen, Jianliang Gao, Yibo Chen, Moctard Babatounde Oloulade, Tengfei Lyu, and Zhao Li. Graphpas: Parallel architecture search for graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2182–2186, 2021.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[10] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 279–288, 2021.

[11] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[12] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2019.

[13] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5251–5260, 2021.

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[15] Guosheng Feng, Chunnan Wang, and Hongzhi Wang. Search for deep graph neural networks. *arXiv preprint arXiv:2109.10047*, 2021.

[16] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 20, pages 1403–1409, 2020.

[17] Chaoyu Guan, Xin Wang, Hong Chen, Ziwei Zhang, and Wenwu Zhu. Large-scale graph neural architecture search. In *International Conference on Machine Learning*, pages 7968–7981. PMLR, 2022.

[18] Chaoyu Guan, Xin Wang, and Wenwu Zhu. Autoattend: Automated attention representation search. In *International Conference on Machine Learning*, pages 3864–3874. PMLR, 2021.

[19] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[20] Yoichi Hirose, Nozomu Yoshinari, and Shinichi Shirakawa. Nas-hpo-bench-ii: A benchmark dataset on joint optimization of convolutional neural network architecture and training hyperparameters. In *Asian Conference on Machine Learning*, pages 1349–1364. PMLR, 2021.

[21] 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. *Neural Information Processing Systems*, 2020.

[22] ZHAO Huan, YAO Quanming, and TU Weiwei. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering*, pages 552–563. IEEE, 2021.

[23] Shengli Jiang and Prasanna Balaprakash. Graph neural network architecture search for molecular property prediction. In *2020 IEEE International Conference on Big Data*, pages 1346–1353. IEEE, 2020.

[24] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *arXiv preprint arXiv:2101.11174*, 2021.

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[26] Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*, 2019.

[27] Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. Nas-bench-nlp: neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*, 2020.

[28] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. Hw-nas-bench: Hardware-aware neural architecture search benchmark. In *International Conference on Learning Representations*, 2020.

[29] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276, 2019.

[30] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[31] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34:21872–21884, 2021.

[32] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Disentangled graph contrastive learning with independence promotion. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[33] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377. PMLR, 2020.

[34] Yanxi Li, Zean Wen, Yunhe Wang, and Chang Xu. One-shot graph neural architecture search with dynamic search space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8510–8517, 2021.

[35] Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020.

[36] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[37] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven CH Hoi. Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4267–4275, 2021.

[38] Zemin Liu, Qiheng Mao, Chenghao Liu, Yuan Fang, and Jianling Sun. On size-oriented long-tailed graph classification of graph neural networks. In *Proceedings of the ACM Web Conference 2022*, pages 1506–1516, 2022.

[39] Zemin Liu, Trung-Kien Nguyen, and Yuan Fang. Tail-gnn: Tail-node graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1109–1119, 2021.

[40] Qing Lu, Weiwen Jiang, Meng Jiang, Jingtong Hu, Sakyasingha Dasgupta, and Yiyu Shi. Fgnas: Fpga-aware graph neural architecture search. *OpenReview*, 2020.

[41] Abhinav Mehrotra, Alberto Gil CP Ramos, Sourav Bhattacharya, Łukasz Dudziak, Ravichander Vipperla, Thomas Chau, Mohamed S Abdelfattah, Samin Ishtiaq, and Nicholas Donald Lane. Nas-bench-asr: Reproducible neural architecture search for speech recognition. In *International Conference on Learning Representations*, 2020.

[42] Microsoft. Neural Network Intelligence, 1 2021.

[43] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[44] Matheus Nunes and Gisele L Pappa. Neural architecture search in graph neural networks. In *Brazilian Conference on Intelligent Systems*, pages 302–317. Springer, 2020.

[45] Zheyi Pan, Songyu Ke, Xiaodu Yang, Yuxuan Liang, Yong Yu, Junbo Zhang, and Yu Zheng. Autostg: Neural architecture search for predictions of spatio-temporal graph. In *Proceedings of the Web Conference 2021*, pages 1846–1855, 2021.

[46] Charles A Pierce, Richard A Block, and Herman Aguinis. Cautionary note on reporting eta-squared values from multifactor anova designs. *Educational and psychological measurement*, 64(6):916–924, 2004.

[47] Yijian Qin, Xin Wang, Peng Cui, and Wenwu Zhu. Gqnas: Graph q network for neural architecture search. In *21st IEEE International Conference on Data Mining*, 2021.

[48] Yijian Qin, Xin Wang, Zeyang Zhang, and Wenwu Zhu. Graph differentiable architecture search with structure learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[49] Yijian Qin, Xin Wang, Ziwei Zhang, Pengtao Xie, and Wenwu Zhu. Graph neural architecture search under distribution shifts. In *International Conference on Machine Learning*, pages 18083–18095. PMLR, 2022.

[50] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

[51] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[52] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

[53] Min Shi, David A Wilson, Xingquan Zhu, Yu Huang, Yuan Zhuang, Jianxun Liu, and Yufei Tang. Evolutionary architecture search for graph neural networks. *arXiv preprint arXiv:2009.10199*, 2020.

[54] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

[55] Chang Su, Jie Tong, Yongjun Zhu, Peng Cui, and Fei Wang. Network embedding in biomedical data science. *Briefings in bioinformatics*, 21(1):182–197, 2020.

[56] Junwei Sun, Bai Wang, and Bin Wu. Automated graph representation learning for node classification. In *2021 International Joint Conference on Neural Networks*, pages 1–7. IEEE, 2021.

[57] Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, and Wenwu Zhu. Autone: Hyperparameter optimization for massive network embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 216–225, 2019.

[58] Renbo Tu, Mikhail Khodak, Nicholas Carl Roberts, Nina Balcan, and Ameet Talwalkar. Nas-bench-360: Benchmarking diverse tasks for neural architecture search. In *OpenReview*, 2021.

[59] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[60] Xingchen Wan. On redundancy and diversity in cell-based neural architecture search. In *Presented at the 10th International Conference on Learning Representations*, volume 25, page 29, 2022.

[61] Xin Wang, Shuyi Fan, Kun Kuang, and Wenwu Zhu. Explainable automated graph representation learning with hyperparameter importance. In *International Conference on Machine Learning*, pages 10727–10737. PMLR, 2021.

[62] Zhili Wang, Shimin Di, and Lei Chen. Autogel: An automated graph neural network with explicit link information. *Advances in Neural Information Processing Systems*, 34, 2021.

[63] Lanning Wei, Huan Zhao, and Zhiqiang He. Learn layer-wise connections in graph neural networks. *arXiv preprint arXiv:2112.13585*, 2021.

[64] Lanning Wei, Huan Zhao, and Zhiqiang He. Designing the topology of graph neural networks: A novel feature fusion perspective. In *Proceedings of the ACM Web Conference 2022*, pages 1381–1391, 2022.

[65] 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.

[66] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020.

[67] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[69] 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.

[70] Shen Yan, Colin White, Yash Savani, and Frank Hutter. Nas-bench-x11 and the power of learning curves. *Advances in Neural Information Processing Systems*, 34, 2021.

[71] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[72] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

[73] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[74] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2019.

[75] Arber Zela, Julien Niklas Siems, Lucas Zimmer, Jovita Lukasik, Margret Keuper, and Frank Hutter. Surrogate nas benchmarks: Going beyond the limited search spaces of tabular nas benchmarks. In *International Conference on Learning Representations*, 2021.

[76] Wentao Zhang, Yu Shen, Zheyu Lin, Yang Li, Xiaosen Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. Pasca: a graph neural architecture search system under the scalable paradigm. *Proceedings of the Web Conference 2022*, 2022.

[77] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[78] Ziwei Zhang, Xin Wang, and Wenwu Zhu. Automated machine learning on graphs: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[79] Huan Zhao, Lanning Wei, quanming yao, and Zhiqiang He. Efficient graph neural architecture search, 2021.

[80] Huan Zhao, Lanning Wei, and Quanming Yao. Simplifying architecture search for graph neural network. *arXiv preprint arXiv:2008.11652*, 2020.

[81] Yiren Zhao, Duo Wang, Daniel Bates, Robert Mullins, Mateja Jamnik, and Pietro Lio. Learned low precision graph neural networks. In *The 2nd Workshop on Machine Learning and Systems*, 2021.

[82] Yiren Zhao, Duo Wang, Xitong Gao, Robert Mullins, Pietro Lio, and Mateja Jamnik. Probabilistic dual network architecture search on graphs. *arXiv preprint arXiv:2003.09676*, 2020.

[83] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[84] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184*, 2019.

[85] Yuwei Zhou, Xin Wang, Hong Chen, Xuguang Duan, Chaoyu Guan, and Wenwu Zhu. Curriculum-nas: Curriculum weight-sharing neural architecture search. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 6792–6801, 2022.

[86] Wenwu Zhu and Xin Wang. Automated machine learning and meta-learning for multimedia, 2021.

[87] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [Yes]

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]