Graph Neural Architecture Search Under Distribution Shifts

Yijian Qin [1] **Xin Wang** [1 2] **Ziwei Zhang** [1] **Pengtao Xie** [3] **Wenwu Zhu** [1]

Abstract

Graph neural architecture search has shown great potentials for automatically designing graph neural network (GNN) architectures for graph classification tasks. However, when there is a distribution shift between training and test graphs, the existing approaches fail to deal with the problem of adapting to unknown test graph structures since they only search for a fixed architecture for all graphs. To solve this problem, we propose a novel Graph neuRal Architecture Customization with disEntangled Self-supervised learning (GRACES) model which is able to generalize under distribution shifts through tailoring a customized GNN architecture suitable for each graph instance with unknown distribution. Specifically, we design a self-supervised disentangled graph encoder to characterize invariant factors hidden in diverse graph structures. Then, we propose a prototype based architecture self-customization strategy to generate the most suitable GNN architecture weights in a continuous space for each graph instance. We further propose a customized supernetwork to share weights among different architectures for the sake of efficient training. Extensive experiments on both synthetic and real-world datasets demonstrate that our proposed GRACES model can adapt to diverse graph structures and achieve state-of-the-art performance for graph classification tasks under distribution shifts.

1. Introduction

Graph-structured data has attracted lots of attention in recent years for its flexible representation ability in various domains. Graph neural networks (GNNs) models such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019) have been proposed and achieved great successes in many graph tasks. Generally, GNNs learn node representations by a recursive message passing scheme where nodes aggregate information from their neighbors iteratively. Then for the graph classification task, GNNs use pooling methods to derive graph-level representations. Different GNN architectures mainly differ in their message-passing mechanism, i.e., how to exchange information, to adapt to the demands of different graph scenarios.

To save human efforts on designing GNN architectures for different tasks and automatically design more powerful GNNs, graph neural architecture search (GraphNAS) (Li et al., 2020; Gao et al., 2020; Wei et al., 2021) has been utilized to search for an optimal GNN architecture. These automatically designed architectures have achieved competitive or better performances compared with manually designed GNNs on datasets with the same distributions under the independently and identically distributed (I.I.D.) assumption, i.e., the training and test graphs are independently sampled from the identical distribution.

Nevertheless, distribution shifts are ubiquitous and inevitable in real-world graph applications where there exist a large number of unforeseen and uncontrollable hidden factors. Taking drug discovery as an example, there exists only a limited amount of training data that can be obtained for experiments, and the interaction mechanism varies greatly for different molecules due to their complex chemical properties (Ji et al., 2022). That being the case, the GNN models designed for drug discovery frequently have to be tested on data with distribution shifts.

The existing GraphNAS approaches under the I.I.D. assumption only search a single fixed GNN architecture based on the training set before directly applying the selected architecture on the test set, failing to deal with varying distribution shifts under the out-of-distribution setting. Because the single GNN architecture discovered by existing methods may overfit the distributions of the training graph data, it may fail to make accurate predictions on test data with various distributions different from the training data.

To solve this problem, in this paper we are the first to study graph neural architecture search for graph classification under distribution shifts, to the best of our knowledge. We propose Graph neuRal Architecture Customization with

[1]Department of Computer Science and Technology, Tsinghua University [2]THU-Bosch JCML center [3]UC San Diego. Correspondence to: Wenwu Zhu <wwzhu@tsinghua.edu.cn>, Xin Wang <xin_wang@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

disEntangled Self-supervised learning (GRACES), which is able to capture key information on graphs with widely varying distributions under the out-of-distribution settings through tailoring a unique GNN architecture for each graph instance. Specifically, we first design a *self-supervised disentangled graph encoder* which projects graphs into a disentangled latent space, where each disentangled factor in the space is trained by the supervised task and corresponding self-supervised learning task simultaneously. This design is able to capture the key information hidden in graphs in a more controllable manner via the self-supervised disentangled graph representation, thus improving the ability of the representations to generalize under distribution shifts. We then propose *architecture self-customization with prototype* to tailor specialized GNN architectures for graphs based on the similarities of their representations with *prototypes vectors* in the latent space, where each prototype vector corresponds to one different operation. We further design the *customized super-network* with differentiable weights on the mixture of different operations, which has great flexibility to ensemble different combinations of operations and enable the proposed GRACES model to be easily optimized in an end-to-end fashion through gradient based methods. We remark that our designs of disentangled graph representations and learnable prototype-operation mapping together are able to enhance the generalization ability of our proposed model under distribution shifts. Extensive experiments on both synthetic and real-world datasets validate the superiority of our proposed GRACES model over existing baselines. Detailed ablation studies further verify the designs of GRACES. [1] Our contributions are summarized as follows.

- We are the first to study graph neural architecture search for graph classification under distribution shifts by proposing the Graph neuRal Architecture Customization with disEntangled Self-supervised learning (GRACES) model, to the best of our knowledge.
- We design three cascaded modules, i.e., self-supervised disentangled graph encoder, architecture self-customization with prototype strategy, and customized super-network, to tailor a unique GNN architecture for each graph instance, thus enabling the ability of our proposed GRACES model in dealing with generalization under distribution shifts with non-I.I.D. settings.
- Extensive experimental results demonstrate that our proposed GRACES model is able to significantly outperform state-of-the-art baselines in terms of graph classification accuracy on both synthetic and real-world datasets.

The rest of the paper is organized as follows. In Section 2, we introduce the problem formulation and preliminaries. We present our proposed method in Section 3 and report experimental results in Section 4. We review related works in Section 5. In Section 6, we conclude the paper.

[1] Our code will be released at https://github.com/THUMNLab/AutoGL

2. Problem Formulation and Preliminaries

Denote the graph space as $\mathcal{G}$ and the label space as $\mathcal{Y}$. We consider a training graph dataset $G_{tr} = \{g_i\}_{i=1}^{N_{tr}}, g_i \in \mathcal{G}$ and the corresponding label set $Y_{tr} = \{y_i\}_{i=1}^{N_{tr}}, y_i \in \mathcal{Y}$. The test graph dataset is denoted as $G_{te} = \{g_i\}_{i=1}^{N_{te}}$ and $Y_{te} = \{y_i\}_{i=1}^{N_{te}}$. The goal of generalization under distribution shifts is to design a model $F : \mathcal{G} \to \mathcal{Y}$ using G_{tr} and Y_{tr} which works well on G_{te} and Y_{te} under the assumption that $P(G_{tr}, Y_{tr}) \neq P(G_{te}, Y_{te})$, i.e.,

$$\arg\min_{F} \mathbb{E}_{G,Y\sim P(G_{te},Y_{te})} \left[\ell\left(F(G), Y\right) | G_{tr}, Y_{tr}\right], \quad (1)$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function. In this paper, we consider a common yet challenging setting that neither Y_{te} nor unlabeled G_{te} is available in the training phase (Wang et al., 2021a). Besides, we mainly focus on F being GNNs, which are state-of-the-art models for graph machine learning. A typical GNN consists of two parts: an architecture $\alpha \in \mathcal{A}$ and learnable weights $w \in \mathcal{W}$, where $\mathcal{A}$ and $\mathcal{W}$ denotes the architecture space and the weight space, respectively. Therefore, we denote GNNs as the following mapping function $F_{\alpha,w} : \mathcal{G} \to \mathcal{Y}$.

For searching GNN architectures, we mostly focus on different GNN layers, i.e., message-passing functions. Therefore, we consider a search space of standard layer-by-layer architectures without sophisticated connections such as residual or jumping connections, though our proposed method can be easily generalized. We choose five widely used GNN layers as our operation candidate set $\mathcal{O}$, including GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GIN (Xu et al., 2019), SAGE (Hamilton et al., 2017), and GraphConv (Morris et al., 2019). Besides, we also adopt MLP, which does not consider graph structures. We fix the pooling layer at the end of the GNN architecture as the standard global mean pooling.

3. The Proposed Method

In this section, we present our proposed method. First, we introduce our framework in Section 3.1. In Section 3.2, we present the self-supervised disentangled graph encoder to capture diverse graph structures. Then, we propose the architecture self-customization with prototype strategy in Section 3.3, which maps the learned graph representation into a tailored GNN architecture. In Section 3.4, we introduce the customized super-network which enables efficient training by weight sharing. We show the optimization procedure in Section 3.5. We analyze the complexity of our method in Section 3.6. Finally, we give some discussion

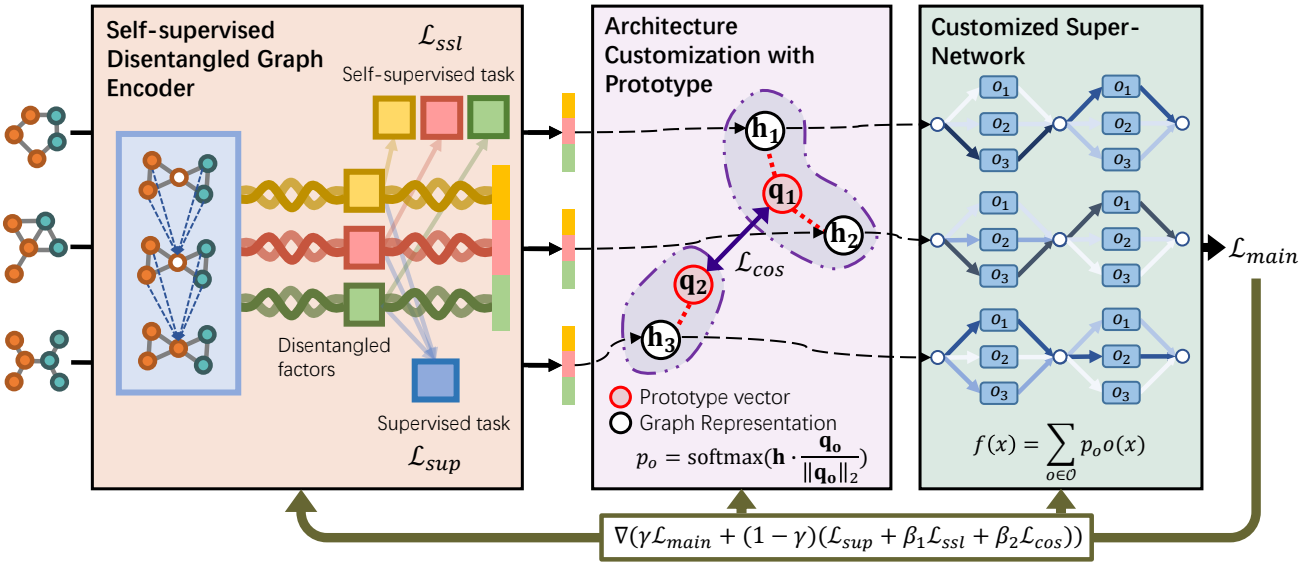


Figure 1. An overview of our proposed GRACES model. The self-supervised disentangled graph encoder captures diverse graph structures by a self-supervised and a supervised loss. Then, the architecture self-customization with prototype module tailors the most suitable GNN architecture based on the learned graph representation. Finally, the customized super-network enables efficient training by weight sharing.

about our model in Section 3.7.

3.1. Framework

In our proposed method, instead of using a fixed GNN architecture for all graphs as in the existing methods, we customize a GNN architecture for each graph. In this way, our proposed method is more flexible and can better handle test graphs under distribution shift since it is known that different GNN architectures suit different graphs (Corso et al., 2020; Xu et al., 2021). To achieve this goal, we aim to learn an architecture mapping function $\Phi_A : \mathcal{G} \to \mathcal{A}$ and a weight mapping function $\Phi_w : \mathcal{G} \times \mathcal{A} \to \mathcal{W}$ so that these functions can automatically generate the optimal GNN for different graphs, including the architecture and its weights. Since the architecture only depends on the graph in our settings, we can further simplify the weight mapping function as $\Phi_w : \mathcal{G} \to \mathcal{W}$. Therefore, we transform Eq. (1) into the following objective function:

$$\min_{\Phi_A, \Phi_w} \gamma \sum_{i=1}^{N_{tr}} \ell\left(F_{\Phi_1(g_i), \Phi_2(g_i)}\left(g_i\right), y_i\right) + (1-\gamma)\mathcal{L}_{reg}, \quad (2)$$

where $\mathcal{L}_{reg}$ is the regularizer and γ is a hyper-parameter. In the following sections, we will introduce in details how to properly design Φ_A, Φ_w, and $\mathcal{L}_{reg}$ so that our proposed method can generalize under distribution shifts.

3.2. Self-supervised disentangled graph Encoder

Graphs from different distributions can have diverse graph structures. To capture such diverse graph structures, we use a self-supervised disentangled graph encoder to learn low-dimensional representations of graphs. Specifically, we adopt K GNNs and learn K-chunk graph representations:

$$\mathbf{H}^{(l)} = \Big\|_{k=1}^{K} \text{GNN}(\mathbf{H}_k^{(l-1)}, \mathbf{A}), \quad (3)$$

where $\mathbf{H}_k^{(l)}$ is the k-th chunk of the node representation at the l-th layer, $\mathbf{A}$ is the adjacent matrix of the graph, and $\|$ represents concatenation. Using these disentangled GNN layers, we can capture different latent factors of the graphs. Then, we adopt a readout layer to aggregate node-level representations into a graph-level representation:

$$\mathbf{h} = \text{Readout}(\mathbf{H}^{(L)}). \quad (4)$$

To learn the parameters of the self-supervised disentangled graph encoder, we use both graph supervised learning and self-supervised learning tasks.

Supervised learning. The downstream target graph task naturally provides supervision signals for learning the self-supervised disentangled graph encoder. Therefore, We place a classification layer after the obtained graph representation to get the prediction for the graph classification task. Denote the graph representation for g_i as $\mathbf{h}_i$. The supervised learning loss is as follows:

$$\mathcal{L}_{sup} = \sum_{i=1}^{N_{tr}} \ell\left(\mathcal{C}\left(\mathbf{h}_i\right), y_i\right) \quad (5)$$

where $\mathcal{C}(\cdot)$ is the classification layer.

Self-supervised learning (SSL). Graph SSL aims to learn informative graph representation through pretext tasks,

which has shown several advantages including reducing label reliance, enhancing robustness, and model generalization ability (Liu et al., 2021; Yehudai et al., 2021). Therefore, we propose to use graph SSL to complement the supervised learning task. Specifically, we set the SSL auxiliary task by generating pseudo labels from graphs structures and using the pseudo labels as extra supervision signals. Besides, we adopt different pseudo labels for different chunks of the disentangled GNN so that our disentangled encoder can capture different factors of the graph structure. In this paper, we focus on the degree distribution of graphs as a representative and explainable structural feature, while it is straightforward to generalize to other graph structures. Specifically, for the k-th GNN chunk, we generate pseudo labels by calculating the ratio of nodes that exactly have degree k. We formulate the SSL objective function as:

$$\mathcal{L}_{ssl} = \sum_{i=1}^{N_{tr}} \sum_{k=1}^{K-1} \ell_{ssl}\left(\hat{y}_{i,k}^{ssl}, y_{i,k}^{ssl}\right), \tag{6}$$

where $y_{i,k}^{ssl}$ is the pseudo-label and $\hat{y}_{i,k}^{ssl}$ is obtained by adopting a regression function, i.e., a linear layer followed by an activation function, on the k-th chunk of the graph representation $\mathbf{h}_i$. Notice that we also leave the last chunk without SSL tasks to allow more flexibility in learning the disentangled graph representations.

3.3. Architecture Self-customization with prototype

After obtaining the graph representation $\mathbf{h}$, we propose the architecture self-customization with prototype strategy to map the representation into a tailored GNN architecture. Specifically, denote the probability of choosing an operation o in the i-th layer of the searched architecture as p_o^i, where $i \in \{1, 2, ..., N\}$, N is the number of layers, and $o \in \mathcal{O}$. We calculate the probability as follows:

$$\hat{p}_o^i = \mathbf{h} \cdot \frac{\mathbf{q}_o^i}{\|\mathbf{q}_o^i\|_2}, p_o^i = \frac{\exp\left(\hat{p}_o^i\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\hat{p}_{o'}^i\right)}, \tag{7}$$

where $\mathbf{q}_o^i$ is a learnable prototype vector representation of the operation o. We adopt the l_2-normalization on $\mathbf{q}$ to ensure numerical stability and fair competition among different operations. Intuitively, in Eq. (7), we learn a prototype vector for each candidate operation and select operations based on the preferences of the graph, i.e., if the graph representation has a large projection on a prototype vector, its corresponding operation is more likely to be selected. Besides, by using the exponential function, the length of $\mathbf{h}$ can decide the shape of p_o^i i.e., the larger $\|\mathbf{h}\|_2$, the more likely that p_o^i are dominated by a few values, indicating that the graph requires specific operations.

Besides, to avoid the mode collapse problem, i.e., vectors of different operations are similar and therefore become indistinguishable, we adopt the following regularizer based on cosine distances between vectors to keep the diversity of operations:

$$\mathcal{L}_{cos} = \sum_i \sum_{o,o' \in \mathcal{O}, o \neq o'} \frac{\mathbf{q}_o^i \cdot \mathbf{q}_{o'}^i}{\|\mathbf{q}_o^i\|_2 \left\|.\mathbf{q}_{o'}^i\right\|_2} \tag{8}$$

Using the architecture self-customization with prototype, we can tailor the most suitable GNN architectures based on the graph representation.

3.4. Weight Generation using customized super-network

Besides GNN architectures, we also need the weights of the architectures. Following the NAS literature (Liu et al., 2019; Pham et al., 2018), we adopt a super-network to obtain the weights of architectures. Specifically, in the super-network, all possible operations are jointly considered by mixing different operations into a continuous space as follows:

$$f^i(\mathbf{x}) = \sum_{o \in \mathcal{O}} p_o^i o(\mathbf{x}) \tag{9}$$

where $\mathbf{x}$ is the input of layer and $f^i(\mathbf{x})$ is the output. Then, we can optimize all the weights using gradient descend methods. Besides, since weights of different architectures are shared, the training will be much more efficient compared to training weights for different architectures separately.

A caveat to notice is that in most NAS literature, the architecture is discretized at the end of the search phase by choosing the operation with the largest p_o^i for all the layers. Then, the weights of the selected architecture are retrained. However, retraining is infeasible in our framework since test graphs can be tailored with different architectures from those for training graphs. Therefore, we directly use the weights of the super-network as the weights in the searched architecture. Besides, we also keep the continuous architecture without the discretization step, enhancing flexibility on architecture customization and simplifying the optimization strategy. Moreover, the intuition is that the customized super-network serves as a strong ensemble model with p_o^i being the ensemble weights, which is also known to benefit out-of-distribution generalization (Shen et al., 2021).

3.5. Optimization Procedure

We have introduced three additional loss functions as the regularizer in Eq. (2), i.e., $\mathcal{L}_{sup}$ and $\mathcal{L}_{ssl}$ for the self-supervised disentangled graph encoder and $\mathcal{L}_{cos}$ for the self-customization module. Therefore, we have:

$$\begin{aligned} \mathcal{L} &= \gamma \mathcal{L}_{main} + (1-\gamma)\mathcal{L}_{reg} \\ \mathcal{L}_{reg} &= \mathcal{L}_{sup} + \beta_1 \mathcal{L}_{ssl} + \beta_2 \mathcal{L}_{cos}, \end{aligned} \tag{10}$$

where $\mathcal{L}_{main}$ is the supervision loss of the tailored architectures in Eq. (2), β_1 and β_2 are hyper-parameters.

Algorithm 1 The Algorithm Framework of Our Proposed Method

Input: Training Dataset G_{tr} and Y_{tr}, Hyper-parameters γ_0, $\Delta\gamma$, β_1, β_2
Initialize all leanable parameters and set $\gamma = \gamma_0$
while Not Converge **do**
  Calculate graph representations $\mathbf{h}$ using Eqs. (3) (4)
  Calculate $\mathcal{L}_{sup}$ and $\mathcal{L}_{ssl}$ using Eq. (5) and Eq. (6)
  Calculate architecture probability p_o^i using Eq. (7)
  Calculate $\mathcal{L}_{cos}$ using Eq. (8)
  Get the parameters from the super-network
  Calculate the overall loss in Eq. (2)
  Update parameters using gradient descends
  Update $\gamma = \gamma - \Delta\gamma$
end while

For the overall optimization, we have two groups of loss functions: the classification loss and the regularizer. At an early stage of the training procedure, the self-supervised disentangled graph encoder may have not been properly trained and the learned graph representation is also not informative, leading to unstable architecture customization. Therefore, we set larger weights for the regularizer, i.e., a smaller initial γ in Eq. (2), to force the self-supervised disentangled graph encoder to learn through its supervised learning and SSL tasks. As the training procedure continues, we can gradually focus more on training the self-customization module and the super-network by increasing γ as:

$$\gamma_t = \gamma_0 + t\Delta\gamma, \tag{11}$$

where γ_t is the hyper-parameter value at the t-th epoch, $\Delta\gamma$ is a small constant. The overall algorithm is shown in Algorithm 1. In the evaluation phase, we directly generate the most suitable GNN architecture with its parameters for the test graphs without retraining.

3.6. Complexity Analysis

Denote $|V|$, $|E|$ as the number of nodes and edges in the graph, respectively, and d as the dimensionality of hidden representations. We use d_e and d_s to denote the dimensionality of the self-supervised disentangled graph encoder and the customized super-network, respectively. Notice that d_e is the overall dimensionality of K-chunks, i.e., the dimensionality of each chunk is d_e/K.

Time complexity. The time complexity of most message-passing GNNs is $O(|E|d+|V|d^2)$. Therefore, the time complexity of our self-supervised disentangled graph encoder is $O(|E|d_e + |V|d_e^2)$. The time complexity of the architecture self-customization with prototype is $O(|\mathcal{O}|^2 d_e)$, since the most time-consuming step is calculating $\mathcal{L}_{cos}$ in Eq (8). The time complexity of the customized super-network is $O(|\mathcal{O}|(|E|d_s+|V|d_s^2))$. The overall time complexity of our method is $O(|E|(d_e+|\mathcal{O}|d_s)+|V|(d_e^2+|\mathcal{O}|d_s^2)+|\mathcal{O}|^2 d_e)$.

Number of learnable parameters. The number of learnable parameters of a typical message-passing GNN is $O(d^2)$. In our framework, the self-supervised disentangled graph encoder has $O(d_e^2)$ parameters, the architecture self-customization module has $O(|\mathcal{O}|d_e)$ parameters, and the customized super-network has $O(|\mathcal{O}|d_s^2)$ parameters. Thus the total number of learnable parameters is $O(d_e^2 + |\mathcal{O}|d_e + |\mathcal{O}|d_s^2))$.

The above analyses show that our proposed method has a linear time complexity with respect to the number of nodes and edges, and the number of learnable parameters is constant, on par with previous GNNs and graph NAS methods. Besides, in practice, $|\mathcal{O}|$ is a small constant (e.g., $|\mathcal{O}| = 6$ in our search space) and we find that usually $d_e \ll d_s$. Therefore, the time complexity and the number of learnable parameters mainly depend on d_s. To ensure a fair comparison with GNN baselines, we set a relatively small d_s for our method so that all methods have a comparable number of parameters, i.e., $|\mathcal{O}|d_s^2 \approx d^2$.

3.7. Discussion

In this part, we provide some intuitive explanation of our model. The continuous architectures can be regarded as a type of ensemble model (Deng et al., 2020), like dropout or multi-head attention. Different submodels in the architecture learn different knowledge from the data. These submodels also affect each other during the training phase, which is de facto a type of regularization. Furthermore, our model can be seen as a variant of the attention mechanism. Traditional attention mechanism can be represented as follows:

$$\mathbf{Attn}_{\mathbf{q}\to\mathbf{k}} = Sim(\mathbf{q}, \mathbf{k}), \tag{12}$$

$$\mathbf{out} = \mathbf{Attn}_{\mathbf{q}\to\mathbf{k}}\mathbf{v}, \tag{13}$$

where $\mathbf{k}$, $\mathbf{q}$, and $\mathbf{v}$ indicate key, query, and value. $\mathbf{Attn}$ is the attention score calculated by a similarity function. In out framework, the **key** is the operation prototype vectors, the **query** is graph representations, and the **value** is candidate GNNs in the search space. Our architecture customization strategy calculate the attention scores of keys (operations) on of a certain query (graph) by Equation (7), then the scores is applied on the values (operations) by Equation (9). Different graphs can attend on different operations to use a customized neural network to learn. We empirically find this mechanism can help out-of-distribution generalization.

4. Experiments

In this section, we report experimental results to verify the effectiveness of our model. We also conduct detailed abla-

Table 1. Dataset Statistics.

Dataset	#Graphs	Split(%)	Avg. $\|V\|$	Avg. $\|E\|$	#Tasks	#Classes	Metric
Spurious-Motif	18,000	50/16.7/33.3	26.1	36.3	1	3	Accuracy
OGBG-MolHIV	41,127	80/10/10	25.5	27.5	1	2	ROC-AUC
OGBG-MolSIDER	1,427	80/10/10	33.6	35.4	27	2	ROC-AUC
OGBG-MolBACE	1,513	80/10/10	34.1	36.9	1	2	ROC-AUC

tion studies to analyze each of our model components.

4.1. Experiment Setting

Datasets. We adopt both synthetic and real-world datasets for graph-level tasks with distribution shifts.

- **Spurious-Motif** (Wu et al., 2022; Ying et al., 2019) is a synthetic dataset where each graph is composed of one base shape (*tree*, *ladder*, and *wheel* denoted by $S = 0, 1, 2$), and one motif shape (*cycle*, *house*, and *crane* denoted by $C = 0, 1, 2$). The base shape is usually larger than the motif shape but the ground-truth label is determined by the motif shape solely. In the distribution shift setting, a manual bias b is added to the distribution between the base and the motif shape in the training set:

$$P(S) = \begin{cases} b & \text{if } S = C, \\ \frac{1-b}{2} & \text{otherwise.} \end{cases} \tag{14}$$

  In the test set, all base and motif shapes are independent with equal probabilities. Thus, we can control the distribution shift by varying b.

- **OGBG-Mol*** (Hu et al., 2020; Wu et al., 2018) is a set of molecular property prediction datasets. Graphs represent molecules and labels are chemical properties of molecules. The datasets are split by the scaffold value, which attempts to separate molecules with different structural frameworks, providing great challenge to graph property prediction.

The statistics of all datasets are shown in Table 1.

Baselines. We compare our model with 10 baselines from the following two different categories.

- **Manually design GNNs**: we include the GNNs in our search space as our baselines, i.e., GCN, GAT, GIN, SAGE, and GraphConv. Global mean pooling is used in these GNNs to generate the graph-level representation. We also include MLP and two recent methods: ASAP (Ranjan et al., 2020) and DIR (Wu et al., 2022).

- **Graph Neural Architecture Search**: we consider two classic NAS baselines, random search and DARTS (Liu et al., 2019). We also consider two GraphNAS baselines, GNAS (Gao et al., 2020), an reinforcement learning based method, and PAS (Wei et al., 2021), a recent GraphNAS method specifically designed for graph classification tasks.

Table 2. The test accuracy of all the methods on the synthetic dataset Spurious-Motif. Numbers after the $\pm$ signs represent standard deviations. The best results are in bold.

bias	$b = 0.7$	$b = 0.8$	$b = 0.9$
GCN	$48.39_{\pm 1.69}$	$41.55_{\pm 3.88}$	$39.13_{\pm 1.76}$
GAT	$50.75_{\pm 4.89}$	$42.48_{\pm 2.46}$	$40.10_{\pm 5.19}$
GIN	$36.83_{\pm 5.49}$	$34.83_{\pm 3.10}$	$37.45_{\pm 3.59}$
SAGE	$46.66_{\pm 2.51}$	$44.50_{\pm 5.79}$	$44.79_{\pm 4.83}$
GraphConv	$47.29_{\pm 1.95}$	$44.67_{\pm 5.88}$	$44.82_{\pm 4.84}$
MLP	$48.27_{\pm 1.27}$	$46.73_{\pm 3.48}$	$46.41_{\pm 2.34}$
ASAP	$54.07_{\pm 13.85}$	$48.32_{\pm 12.72}$	$43.52_{\pm 8.41}$
DIR	$50.08_{\pm 3.46}$	$48.22_{\pm 6.27}$	$43.11_{\pm 5.43}$
random	$45.92_{\pm 4.29}$	$51.72_{\pm 5.38}$	$45.89_{\pm 5.09}$
DARTS	$50.63_{\pm 8.90}$	$45.41_{\pm 7.71}$	$44.44_{\pm 4.42}$
GNAS	$55.18_{\pm 18.62}$	$51.64_{\pm 19.22}$	$37.56_{\pm 5.43}$
PAS	$52.15_{\pm 4.35}$	$43.12_{\pm 5.95}$	$39.84_{\pm 1.67}$
GRACES	$\mathbf{65.72_{\pm 17.47}}$	$\mathbf{59.57_{\pm 17.37}}$	$\mathbf{50.94_{\pm 8.14}}$

More experimental details including the hyper-parameter settings are provided in Appendix B.

4.2. Results on Synthetic Datasets

Experimental Setting We select three different bias values b for Spurious-Motif, i.e., 0.7, 0.8, and 0.9. We run all experiments 10 times with different random seeds and report the average results with standard deviations.

Qualitative Results We summarize the experimental results in Table 2. The table shows that our model outperforms all baselines in all three settings by a large margin. Specifically, we find that all GNNs perform poorly, indicating that they are easily affected by spurious correlations and cannot handle the distribution shift. Moreover, we find that NAS methods achieve slightly better results than manually designed GNNs in most cases, demonstrating the importance of automating architecture. Nevertheless, these methods also suffer from distribution shifts. In contrast, GRACES shows much better results by customizing architectures for different graphs and capturing the ground-truth predictive

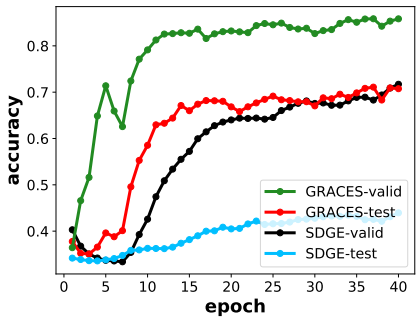


Figure 2. The validation and test accuracy of GRACES and self-supervised disentangled graph encoder (SDGE) on Spurious-Motif.

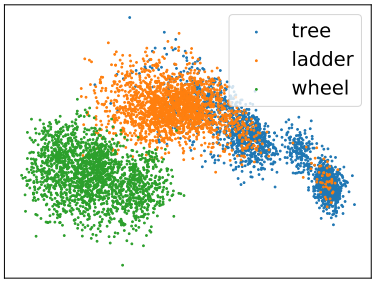


Figure 3. The visualization of graph representations for the test dataset on Spurious-Motif.

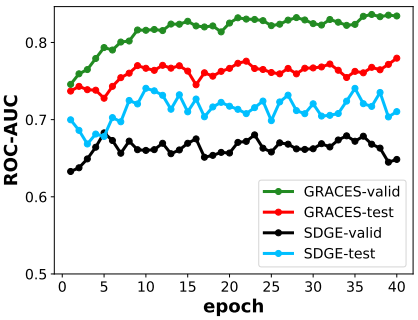
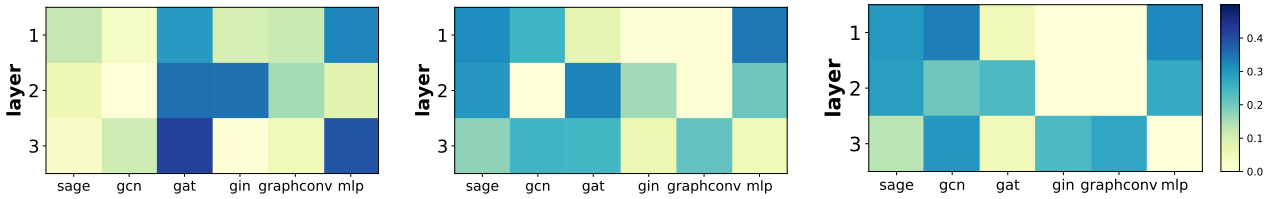


Figure 4. The validation and the test accuracy of GRACES and self-supervised disentangled graph encoder (SDGE) on OGBG-MolHIV.



(a) *Tree*-based graphs (b) *Ladder*-based graphs (c) *Wheel*-based graphs

Figure 5. Visualizations of operation probabilities for graphs with different base shapes on Spurious-Motif.

patterns under distribution shifts.

Analysis To better understand the mechanism of our model, we conduct some analyses. First, since the self-supervised disentangled graph encoder in our model is also trained with the supervised task, one may wonder whether the disentangled encoder per se can achieve satisfactory generalization under distribution shifts. We report the results of Spurious-Motif for $b = 0.8$ in Figure 2. The results show that the self-supervised disentangled graph encoder does not achieve satisfactory generalization performance, indicating that using a disentangled GNN alone cannot well handle distribution shifts. Nevertheless, we can observe that the test accuracy of GRACES and the validation accuracy of the encoder improve simultaneously, i.e., around the 8-th epoch. The results indicate that training the disentangled encoder can help to improve the performance and generalization ability of the full GRACES model. A plausible reason is that the self-supervised disentangled graph encoder captures the shortcut structure features in the training set, i.e., the base shape in Spurious-Motif. As a result, graphs with similar base shapes will have similar representations and thus have similar customized GNN architectures. Therefore, individual GNN architecture in the super-network only needs to deal with graphs with similar base shapes, i.e., without the spurious correlation, which leads to a good generalization ability.

Besides, we visualize the learned graph representation in a 2-D plane using PCA to reduce dimensionality. The results of the test graphs when $b = 0.8$ are shown in Figure 3. We can observe that our learned graph representation can separate ground-truth base shapes well into different parts, showing that the self-supervised disentangled graph encoder can capture structure information.

Moreover, for graphs with different base shapes, we show the operation probabilities p_o^i in expectation in Figure 5. We observe that graphs with different base shapes prefer different architectures, e.g., *tree*-based graphs prefer GAT and MLP in the third layer, while these two operations are seldomly chosen in the other two types of graphs. The operation distributions are similar for *ladder*-based and *wheel*-based graphs in the first layer, but differ in other layers. The results revalidate our analysis above that different base shapes prefer different architectures.

4.3. Results on Real-world Datasets

We further conduct experiments on three molecular graph classification benchmarks in OGBG-Mol*: HIV, SIDER, and BACE. We report the results in Table 3. The results show that our proposed GRACES model again outperforms all the baselines on the three datasets, demonstrating that our model is able to capture the complex distribution shifts in some cases. Two existing graph NAS methods, DARTS and PAS, fail to outperform manually designed GNNs on

Table 3. The test ROC-AUC of all the methods on the real-world datasets OGBG-Mol*. Numbers after the ± signs represent standard deviations. The best results are in bold.

dataset	hiv	sider	bace
GCN	$75.99_{\pm 1.19}$	$59.84_{\pm 1.54}$	$68.93_{\pm 6.95}$
GAT	$76.80_{\pm 0.58}$	$57.40_{\pm 2.01}$	$75.34_{\pm 2.36}$
GIN	$77.07_{\pm 1.49}$	$57.57_{\pm 1.56}$	$73.46_{\pm 5.24}$
SAGE	$75.58_{\pm 1.40}$	$56.36_{\pm 1.32}$	$74.85_{\pm 2.74}$
GraphConv	$74.46_{\pm 0.86}$	$56.09_{\pm 1.06}$	$78.87_{\pm 1.74}$
MLP	$70.88_{\pm 0.83}$	$58.16_{\pm 1.41}$	$71.60_{\pm 2.30}$
ASAP	$73.81_{\pm 1.17}$	$55.77_{\pm 1.18}$	$71.55_{\pm 2.74}$
DIR	$77.05_{\pm 0.57}$	$57.34_{\pm 0.36}$	$76.03_{\pm 2.20}$
DARTS	$74.04_{\pm 1.75}$	$60.64_{\pm 1.37}$	$76.71_{\pm 1.83}$
PAS	$71.19_{\pm 2.28}$	$59.31_{\pm 1.48}$	$76.59_{\pm 1.87}$
GRACES	$\mathbf{77.31_{\pm 1.00}}$	$\mathbf{61.85_{\pm 2.56}}$	$\mathbf{79.46_{\pm 3.04}}$

real-world graphs.

Analysis To better analyze our proposed method on real-world graphs, we also plot the validation and test results of GRACES and the results of our self-supervised disentangled graph encoder on OGBG-MolHIV. The results are shown in Figure 4. We can see that GRACES report better results than the encoder in both the validation and the test set, indicating that both the encoder and customized super-network are indispensable for GRACES.

4.4. Ablation Study

In this section, we evaluate the effectiveness of each module of our framework by conducting ablation studies. We compare the following variants of our model as follows:

- GRACES-MIX: we remove architecture customization so that all operations have an equal weight p_o^i.
- GRACES-GCN: we replace the self-supervised disentangled graph encoder with a vanilla GCN.
- GRACES-FC: we replace the architecture self-customization with prototype with a fully-connected layer.
- GRACES-NO-COS: we remove the cosine distance loss.
- GRACES-DISCRETE: we let the architecture self-customization strategy to output discrete architectures, i.e., constraining $\{p_o^i | o \in \mathcal{O}\}$ to be one-hot.

We also compare with the best manually designed GNNs, denoted as "manual". We examine these variants on the synthetic dataset Spurious-Motif. The results are presented in Table 4. We have the following observations.

Table 4. The test accuracy of different variants of GRACES on the synthetic dataset Spurious-Motif.

bias	$b = 0.7$	$b = 0.8$	$b = 0.9$
manual	$50.75_{\pm 4.89}$	$46.73_{\pm 3.48}$	$46.41_{\pm 2.34}$
MIX	$52.80_{\pm 10.57}$	$49.24_{\pm 5.42}$	$43.11_{\pm 1.86}$
GCN	$58.86_{\pm 13.39}$	$51.62_{\pm 13.23}$	$47.85_{\pm 9.57}$
FC	$63.76_{\pm 14.68}$	$57.11_{\pm 20.38}$	$47.39_{\pm 10.28}$
NO-COS	$55.80_{\pm 18.13}$	$51.65_{\pm 14.12}$	$44.95_{\pm 9.41}$
DISCRETE	$42.84_{\pm 7.84}$	$39.81_{\pm 6.62}$	$40.80_{\pm 5.74}$
GRACES	$\mathbf{65.72_{\pm 17.47}}$	$\mathbf{59.57_{\pm 17.37}}$	$\mathbf{50.94_{\pm 8.14}}$

Table 5. Empirical search time on Spurious-Motif and OGBG-Mol* (NVIDIA GeForce RTX 3090).

DATASET	SPURIOUS-MOTIF	HIV	SIDER	BACE
GRACES	972S	2325S	115S	114S
DARTS	872S	2049S	111S	108S

Overall, our proposed full GRACES model outperforms all the variants under all three settings, demonstrating that each component of our method is indispensable to achieve satisfactory generalization performance under distribution shifts. GRACES-MIX, which simply mixes all candidate operations without architecture customization achieves slightly better performances than the manually designed GNNs in the search space generally. Since we have ensured that the number of learnable parameters in our proposed method equals to manually designed GNNs (see Section 3.6), the results indicate that mixing different message-passing layers can be beneficial. Our proposed GRACES can naturally utilize this advantage, thanks to our customized super-network. In contrast, architecture discretization leads to poor performance, indicating that this technique, which is widely used in the NAS literature, does not suit our problem.

In addition, if we adopt a normal GCN as the encoder, the accuracy will drop severely, indicating the importance of capturing diverse graph structures using our proposed self-supervised disentangled graph encoder. As for the architecture self-customization with prototype, either replacing it with a fully-connected layer or removing the consine distance loss causes performance degradation, demonstrating that our strategy can better customize the architectures.

4.5. Search Time

We also measure the search time of DARTS and our proposed method and show the results in Table 5. Our model takes comparable running time as DARTS, indicating our model design does not bring much burden more than DARTS on architecture searching phase. The results also confirms our theoretical complexity analysis in Section 3.6, demonstrate that our model has not only high effectiveness

but also high efficiency.

5. Related Works

5.1. Graph neural network

Message-passing GNNs (Kipf & Welling, 2017; Veličković et al., 2018; Xu et al., 2019; Hamilton et al., 2017) have been proposed as an effective framework for graph machine learning following the neighborhood aggregation scheme. At each layer, nodes learn representations by aggregating their neighbors' representations. After K such layers, the vector representation of each node captures both the structural and the semantic information within the k-hop neighborhood region of the node. Then, the representation of the whole graph is learned by pooling all node representations (Kipf & Welling, 2017; Xu et al., 2019). Disentanglement technique (Wang et al., 2021b; 2022a; Chen et al., 2021) is also applied on GNNs (Ma et al., 2019; Li et al., 2021), which helps to generate more representative features with different factors for graphs.

Distribution shifts is inevitable in real-world data (Zhu et al., 2015). However, most existing GNNs ignore the problem of out-of-distribution generalization and suffer from severe performance deterioration when there exist distribution shifts between training and test graphs (Hu et al., 2020; Koh et al., 2021; Wu et al., 2018; Zhang et al., 2022). Some pioneer works start to study the distribution shift problems of GNNs such as size generalization (Knyazev et al., 2019; Yehudai et al., 2021; Bevilacqua et al., 2021), node-level tasks (Fan et al., 2022; Zhu et al., 2021), or invariant learning (Wu et al., 2022). However, they do not consider the architecture perspective. Xu et al. (2021) theoretically show that GNN architectures can affect the generalization ability, but they do not consider how to obtain the optimal architecture given a graph dataset. Another line of GNN works focus on treating different graph data instances differently. Policy-GNN (Lai et al., 2020) determines the number of aggregation layers for each node using reinforcement learning. Customized-GNN (Wang et al., 2021d) generates different GNN parameters for different graph instances. Nevertheless, these works do not explore searching GNN architecture under distribution shifts.

5.2. Neural architecture search

Recent years have witnessed a surge of research interests on NAS methods, which aim at designing neural architectures automatically for given tasks. Since the architecture search space is discrete, reinforcement learning (Zoph & Le, 2017; Pham et al., 2018; Qin et al., 2021a) and evolution algorithm (Xie & Yuille, 2017; Liu et al., 2018) are often used in NAS methods. Besides, another strategy is transferring the discrete architecture search space into a differentiable space, e.g., DARTS (Liu et al., 2019) and SNAS (Xie et al., 2019) construct a super-network where all candidate operations are mixed and make it possible to update the architecture as well as the weights simultaneously through the classical gradient descent method.

Recently, instance-aware NAS searches for a mapping function from an image to an architecture so that different instances can have different architectures in the test phase. InstaNAS (Cheng et al., 2020) uses a controller to capture the representations of images and assign them to different discrete CNN architectures. However, this framework is not differentiable and hard to optimize. DDW (Yuan et al., 2021) uses global spatial information of images to generate wiring patterns among different layers but does not consider operation selection. Besides, NAS-OoD (Bai et al., 2021) aims to search for an architecture that can generalize to out-of-distribution data. Nevertheless, all the above works focus on computer vision and cannot be easily applied to GNNs.

Automated graph machine learning is gaining an increasing number of attentions from the research community. (Zhang et al., 2021; Wang et al., 2022b), including hyper-parameter optimization on graphs (Tu et al., 2019; Wang et al., 2021c) and graph neural architecture search (GraphNAS) (Li et al., 2020; Gao et al., 2020; Zhou et al., 2019; Guan et al., 2021; Qin et al., 2021b; Cai et al., 2022). Existing GraphNAS on graph classification tasks (Jiang & Balaprakash, 2020; Peng et al., 2020; Wei et al., 2021; Zhili Wang, 2021; Cai et al., 2021) are all based on the I.I.D. assumption and ignore the distribution shift problem.

6. Conclusion

In this paper, we propose a novel GRACES method to improve the generalization ability of GNAS under distribution shifts. Our core idea is tailoring a customized GNN architecture suitable for each graph instance with unknown distribution by designing a self-supervised disentangled graph encoder, the architecture self-customization with prototype strategy, and the customized super-network. Extensive experiments on both synthetic and real-world datasets demonstrate that our proposed model can adapt to diverse graph structures and achieve state-of-the-art performance for the graph classification task under distribution shifts. Detailed ablation studies further verify the designs of our proposed.

Acknowledgement

This work is supported by the National Key Research and Development Program of China No. 2020AAA0106300National Natural Science Foundation of China No. 62102222 and partially funded by THU-Bosch JCML center.

References

Bai, H., Zhou, F., Hong, L., Ye, N., Chan, S.-H. G., and Li, Z. Nas-ood: Neural architecture search for out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8320–8329, 2021.

Bevilacqua, B., Zhou, Y., and Ribeiro, B. Size-invariant graph representations for graph classification extrapolations. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 837–851, 2021.

Cai, J., Wang, X., Guan, C., Tang, Y., Xu, J., Zhong, B., and Zhu, W. Multimodal continual graph learning with neural architecture search. In *Proceedings of the ACM Web Conference 2022*, pp. 1292–1300, 2022.

Cai, S., Li, L., Deng, J., Zhang, B., Zha, Z., Su, L., and Huang, Q. Rethinking graph neural architecture search from message-passing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6657–6666, 2021.

Chen, H., Chen, Y., Wang, X., Xie, R., Wang, R., Xia, F., and Zhu, W. Curriculum disentangled recommendation with noisy multi-feedback. *Advances in Neural Information Processing Systems*, 34, 2021.

Cheng, A., Lin, C. H., Juan, D., Wei, W., and Sun, M. Instanas: Instance-aware neural architecture search. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33, 2020.

Deng, Z., Dong, Y., Zhang, S., and Zhu, J. Understanding and exploring the network with stochastic architectures. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*, 2020.

Fan, S., Wang, X., Shi, C., Kuang, K., Liu, N., and Wang, B. Debiased graph neural networks with agnostic label selection bias. *IEEE transactions on neural networks and learning systems*, 2022.

Gao, Y., Yang, H., Zhang, P., Zhou, C., and Hu, Y. Graph neural architecture search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

Guan, C., Wang, X., and Zhu, W. Autoattend: Automated attention representation search. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 2017.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in neural information processing systems*, 2020.

Ji, Y., Zhang, L., Wu, J., Wu, B., Huang, L.-K., Xu, T., Rong, Y., Li, L., Ren, J., Xue, D., Lai, H., Xu, S., Feng, J., Liu, W., Luo, P., Zhou, S., Huang, J., Zhao, P., and Bian, Y. Drugood: Out-of-distribution (ood) dataset curator and benchmark for ai-aided drug discovery – a focus on affinity prediction problems with noise annotations. In *arXiv:2201.09637*, 2022.

Jiang, S. and Balaprakash, P. Graph neural network architecture search for molecular property prediction. In *2020 IEEE International Conference on Big Data*, pp. 1346–1353, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Knyazev, B., Taylor, G. W., and Amer, M. Understanding attention and generalization in graph neural networks. *Advances in Neural Information Processing Systems*, 32: 4202–4212, 2019.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, 2021.

Lai, K., Zha, D., Zhou, K., and Hu, X. Policy-gnn: Aggregation optimization for graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 461–471. ACM, 2020.

Li, G., Qian, G., Delgadillo, I. C., Muller, M., Thabet, A., and Ghanem, B. Sgas: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Li, H., Wang, X., Zhang, Z., Yuan, Z., Li, H., and Zhu, W. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search. In *Proceedings of the 6th*

International Conference on Learning Representations, 2018.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Liu, Y., Pan, S., Jin, M., Zhou, C., Xia, F., and Yu, P. S. Graph self-supervised learning: A survey. *arXiv preprint arXiv:2103.00111*, 2021.

Ma, J., Cui, P., Kuang, K., Wang, X., and Zhu, W. Disentangled graph convolutional networks. In *International conference on machine learning*, pp. 4212–4221. PMLR, 2019.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Peng, W., Hong, X., Chen, H., and Zhao, G. Learning graph convolutional network for skeleton-based human action recognition by neural searching. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 2669–2676, 2020.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Qin, Y., Wang, X., Cui, P., and Zhu, W. Gqnas: Graph q network for neural architecture search. In *21st IEEE International Conference on Data Mining*, 2021a.

Qin, Y., Wang, X., Zhang, Z., and Zhu, W. Graph differentiable architecture search with structure learning. In *Advances in neural information processing systems*, 2021b.

Ranjan, E., Sanyal, S., and Talukdar, P. P. ASAP: adaptive structure aware pooling for learning hierarchical graph representations. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., and Cui, P. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Tu, K., Ma, J., Cui, P., Pei, J., and Zhu, W. Autone: Hyperparameter optimization for massive network embedding. In *KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Wang, J., Lan, C., Liu, C., Ouyang, Y., and Qin, T. Generalizing to unseen domains: A survey on domain generalization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4627–4635, 8 2021a. Survey Track.

Wang, X., Chen, H., and Zhu, W. Multimodal disentangled representation for recommendation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2021b.

Wang, X., Fan, S., Kuang, K., and Zhu, W. Explainable automated graph representation learning with hyperparameter importance. In *Proceedings of the 38th International Conference on Machine Learning*, 2021c.

Wang, X., Chen, H., Zhou, Y., Ma, J., and Zhu, W. Disentangled representation learning for recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022a.

Wang, X., Zhang, Z., and Zhu, W. Automated graph machine learning: Approaches, libraries and directions. *arXiv preprint arXiv:2201.01288*, 2022b.

Wang, Y., Ma, Y., Jin, W., Li, C., Aggarwal, C., and Tang, J. Customized graph neural networks, 2021d.

Wei, L., Zhao, H., Yao, Q., and He, Z. Pooling architecture search for graph classification. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management*, 2021.

Wu, Y., Wang, X., Zhang, A., He, X., and Chua, T.-S. Discovering invariant rationales for graph neural networks. In *International Conference on Learning Representations*, 2022.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. S. Moleculenet: A benchmark for molecular machine learning. *Chemical sciense*, 2018.

Xie, L. and Yuille, A. Genetic cnn. In *Proceedings of the IEEE international conference on computer vision*, 2017.

Xie, S., Zheng, H., Liu, C., and Lin, L. Snas: stochastic neural architecture search. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations*, 2019.

Xu, K., Zhang, M., Li, J., Du, S. S., Kawarabayashi, K.-I., and Jegelka, S. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations*, 2021.

Yehudai, G., Fetaya, E., Meirom, E., Chechik, G., and Maron, H. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pp. 11975–11986. PMLR, 2021.

Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, 2019.

Yuan, K., Li, Q., Guo, S., Chen, D., Zhou, A., Yu, F., and Liu, Z. Differentiable dynamic wirings for neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Zhang, Z., Wang, X., and Zhu, W. Automated machine learning on graphs: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021. Survey track.

Zhang, Z., Zhang, Z., Wang, X., and Zhu, W. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *Proceedings of the aaai conference on artificial intelligence*, 2022.

Zhili Wang, Shimin Di, L. C. Autogel: An automated graph neural network with explicit link information. In *Advances in neural information processing systems*, 2021.

Zhou, K., Song, Q., Huang, X., and Hu, X. Auto-gnn: Neural architecture search of graph neural networks. *CoRR*, 2019.

Zhu, Q., Ponomareva, N., Han, J., and Perozzi, B. Shift-robust gnns: Overcoming the limitations of localized graph training data. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhu, W., Cui, P., Wang, Z., and Hua, G. Multimedia big data computing. *IEEE multimedia*, 22(3):96–c3, 2015.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

A. Notations

Table 6. Meanings of notations.

Notation	Meaning
$\mathcal{G}$	The graph space
$\mathcal{Y}$	The label space
$\mathcal{W}$	The weight space
$\mathcal{A}$	The architecture space
$\mathcal{O}$	The operation search space
G, Y	A graph dataset and its corresponding labels
g, y	A graph and its corresponding label
F	A model mapping $\mathcal{G} \to \mathcal{Y}$
ℓ	A loss function
K	The number of chunks
$\mathbf{H}^{(l)}$	The node representation at the i-th layer
$\mathbf{h}$	The graph representation
$\mathbf{q}_o^i$	The prototype vector of operation o at the i-th layer
p_o^i	The weight of operation o at the i-th layer
$\mathcal{L}_{sup}$	The supervision loss of the encoder
$\mathcal{L}_{ssl}$	The self-supervision loss of the encoder
$\mathcal{L}_{cos}$	The cosine distance loss of the prototype vectors
$\mathcal{L}_{main}$	The supervision loss of the final prediction given by the super-network
$\mathcal{L}_{reg}$	The sum of all regularizer losses
γ	The hyper-parameter to control the contribution of regularizer
β_1	The hyper-parameter to control the contribution of $\mathcal{L}_{ssl}$
β_2	The hyper-parameter to control the contribution of $\mathcal{L}_{cos}$
d_e	The dimension of the encoder
d_s	The dimension of the super-network

B. Hyper-parameter Settings

We use different learning rate for the three parts of our model. Typically, the learning rate of the self-supervised disentangled encoder is 1.5e-4. The learning rate of the architecture self-customization strategy is 1e-4. Besides, the training procedure of these two parts are consine annealing scheduled. The learning rate of the customized super-network is 2e-3. We initial γ as 0.07 and increase it to 0.5 linearly. In addition, we set $\beta_1 = 0.05$ and $\beta_2 = 0.002$. For our method and all baselines, we set the number of layers as 2 in OGBG-MolHIV and 3 in the other datasets, For all methods, we use edge features in OGBG datasets and virtual node mechanism in OGBG-MolHIV and OGBG-MolSIDER. For all methods, we fix the first layer as GIN in Spurious-Motif since some of them cannot deal with constant node features. The hidden dimension in Spurious-Motif is 64 for all baselines and 26 for our method. In OGBG-Mol* datasets, the hidden dimension is 300 for baselines and 128 for our method.