Multi-dim Node-Specific Graph Neural Architecture Search for Distribution Shift

Qiyi Wang*, Jun Chen*, and Jishen Lin*

Tongji University, Shanghai 201800, China

Abstract. Graph neural architecture search (GraphNAS) has shown promise in mitigating performance degradation of graph neural networks (GNNs) under distribution shift. Recent methods employ weight sharing and generate tailored GNN architectures for each graph in an end-to-end manner. However, they often overlook the distributional characteristics across graphs and require extensive training data to function effectively. In settings with sparse or single training graphs, these approaches fail to discover optimal mappings between graphs and architectures, leading to poor generalization under distribution shifts. In this work, we propose NodeNAS, a node-specific graph neural architecture search framework that customizes aggregation strategies at the node level. NodeNAS disentangles node topology from graph distribution, enabling flexible architecture adaptation with limited data. Building on this, we introduce Multi-dim NodeNAS (MNNAS), an adaptive aggregation attention-based extension that learns a node-specific architecture customizer with enhanced generalizability. By expanding the vertical depth of the search space, MNNAS supports simultaneous customization of node-specific architectures across multiple dimensions. Moreover, we model the power-law distribution of node degrees under different assortativity conditions, capturing structure-invariant properties to guide architecture selection across dimensions. Extensive experiments on both supervised and unsupervised tasks demonstrate that MNNAS significantly outperforms state-of-the-art methods and achieves robust generalization under distribution shift.

Keywords: NodeNAS · Architecture Customization · Distribution Shift.

1 Introduction

Graph Neural Networks (GNNs) have achieved remarkable success across a wide array of graph-structured learning tasks, including graph classification, partitioning, and community detection [40,24,36]. At the heart of GNNs lies the message passing mechanism, wherein each node aggregates information from its local neighborhood to iteratively refine its representation. While this design enables GNNs to capture complex relational dependencies, it also creates a fundamental

[1] * Both authors contributed equally to this paper.

vulnerability: the learned representations and predictive performance of GNNs are inherently sensitive to the underlying graph structure. As a result, GNNs often suffer significant performance degradation when deployed in settings where the test distribution deviates from the training distribution, a phenomenon commonly referred to as distribution shift.

To tackle the limitations posed by distribution shifts, Graph Neural Architecture Search (GraphNAS) has emerged as a promising paradigm. By automating the design of GNN architectures through data-driven search, GraphNAS allows the model to explore diverse aggregation and propagation strategies beyond manually crafted heuristics. Early GraphNAS frameworks [6,49] treat architecture design as a black-box optimization problem within a discrete search space. While these methods provide a degree of architectural flexibility, they implicitly assume that training and test data are drawn from an independent and identically distributed (IID) setting—an assumption often violated in real-world graphs.

Recent advancements in GraphNAS have introduced more expressive search paradigms by employing differentiable search techniques and weight sharing across architectures [29,44]. These methods learn representations of entire graphs and dynamically generate graph-specific architectures in an end-to-end manner. By relaxing the selection process over candidate operations into a learnable soft weighting scheme, these approaches have demonstrated improved robustness under distribution shift. However, despite these advances, several critical limitations remain.

First, current GraphNAS methods typically require large amounts of training data to effectively model the diverse architectural preferences across graph instances. When faced with sparse data or even single-graph training scenarios, these models struggle to infer reliable mappings from graphs to architectures, thereby failing to generalize under distribution shift.

Second, most existing methods assume a graph-level granularity for architecture customization, where all nodes within a graph share the same aggregation strategy. This overlooks a fundamental property of many real-world graphs: the node degree distribution follows a long-tail pattern, consisting of a small number of high-degree (head) nodes and a large number of low-degree (tail) nodes. A single, shared aggregation function cannot effectively adapt to such heterogeneous topological roles. To date, no existing GraphNAS framework provides node-level architectural differentiation sensitive to these structural disparities.

To overcome these challenges, we propose Node-specific Neural Architecture Search (NodeNAS), a novel framework designed to learn node-specific aggregation strategies within a unified end-to-end architecture search paradigm. As illustrated in Figure 1, NodeNAS aims to associate each node with a customized architecture by learning a node-level probability distribution over candidate operations. This design enables the model to assign distinct aggregation mechanisms to nodes with different degrees or topological roles. Moreover, NodeNAS explicitly disentangles the node degree distribution from global graph properties such as assortativity, allowing the model to isolate and suppress spurious mo-

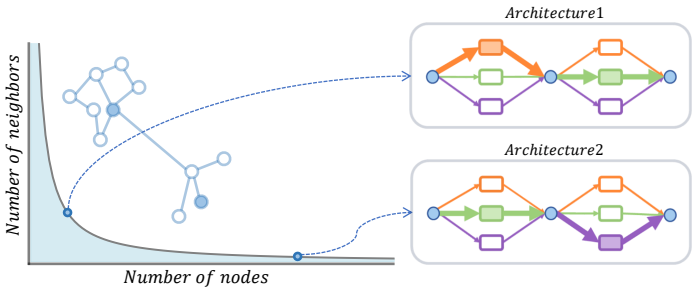

Fig. 1: An overview of NodeNAS

tifs during the learning of node-architecture mappings. In doing so, NodeNAS improves generalization to unseen graphs under distribution shift.

Building on this foundation, we further introduce Multi-dimension NodeNAS (MNNAS), a principled extension that expands the search space along multiple semantic dimensions to capture deeper and more transferable structural patterns. Specifically, MNNAS introduces a multi-dimensional architecture search network in which each node can be simultaneously associated with multiple candidate architectures across different Search Dimensions (S-Dims). These dimensions correspond to independent latent factors governing architecture variation, thereby enriching the representational diversity of the learned GNN.

To maintain parameter efficiency and improve generalizability, we design the operation-to-embedding mapping to be shared across all S-Dims. A dedicated mapping encoder projects candidate operations into a continuous embedding space, allowing for differentiable mixture weights during architecture selection. In addition, MNNAS integrates an adaptive aggregation attention mechanism guided by a link pattern encoder, which captures distribution-invariant patterns across graphs with varying assortativities. This mechanism facilitates the parallel customization of multiple node-specific architectures while identifying and neutralizing spurious structural correlations.

The final node representation is computed by integrating the outputs from multiple customized architectures. Through this design, MNNAS not only achieves end-to-end training efficiency via weight sharing but also enhances robustness under distribution shift. From an information-theoretic perspective, we analyze the interpretability and generalization behavior of MNNAS through the lens of the Information Bottleneck (IB) principle.

We conduct extensive experiments on both supervised and unsupervised benchmarks to evaluate the proposed framework. Results consistently show that MNNAS significantly outperforms existing GraphNAS and GNN models in scenarios characterized by distribution shifts, validating the effectiveness of multi-dimension, node-specific architecture customization.

Our key contributions are summarized as follows:

- We propose a novel node-specific graph neural architecture search framework, NodeNAS, which learns individualized embedding update strategies for each node. Unlike conventional GraphNAS approaches that assign a unified architecture per graph, NodeNAS enables fine-grained customization at the node level. This design allows the model to effectively adapt to graphs with diverse and unknown distributional properties, mitigating performance degradation arising from structural heterogeneity.
- We introduce Multi-dim NodeNAS (MNNAS), a multi-dimensional extension of NodeNAS, which supports simultaneous architecture customization across multiple latent semantic dimensions. To this end, we design an adaptive aggregation attention mechanism that explicitly disentangles node degree distributions (following power-law patterns) from graph-level assortativity. The resulting multi-dimensional search network enables interpretable and high-performing architecture customization for individual nodes, significantly enhancing generalization performance under distribution shift.
- To the best of our knowledge, MNNAS is the first NAS framework capable of achieving strong generalization under distribution shift even in the extreme case of single-graph training. Moreover, it is the first to extend neural architecture search to unsupervised graph tasks such as community detection, while maintaining robust generalization performance across a wide range of out-of-distribution graph scenarios.

2 Preliminaries

2.1 Out-Of-Distribution Generalization

Given the graph space $\mathcal{G}$ and label space $\mathcal{Y}$, we define a training graph dataset $\mathcal{G}_{\text{tr}} = \{g_i\}_{i=1}^{N_{\text{tr}}}, g_i \in \mathcal{G}$, along with a corresponding label set $\mathcal{Y}_{\text{tr}} = \{y_i\}_{i=1}^{N_{\text{tr}}}, y_i \in \mathcal{Y}$. Similarly, the test graph dataset is denoted as $\mathcal{G}_{\text{te}} = \{g_i\}_{i=1}^{N_{\text{te}}}$, and the label set as $\mathcal{Y}_{\text{te}} = \{y_i\}_{i=1}^{N_{\text{te}}}$. The objective of out-of-distribution generalization is to achieve a model $F : \mathcal{G} \to \mathcal{Y}$ using $\mathcal{G}_{\text{tr}}$ and $\mathcal{Y}_{\text{tr}}$, which performs effectively on $\mathcal{G}_{\text{te}}$ and $\mathcal{Y}_{\text{te}}$, under the assumption that the distributions $P(\mathcal{G}_{\text{tr}}, \mathcal{Y}_{\text{tr}}) \neq P(\mathcal{G}_{\text{te}}, \mathcal{Y}_{\text{te}})$, where $P(\mathcal{G}, \mathcal{Y})$ represents the distribution of the graphs and their labels. The objective of OOD generalization can be expressed as:

$$\arg\min_{F} \mathbf{E}_{\mathcal{G},\mathcal{Y}\sim P(\mathcal{G}_{\text{te}},\mathcal{Y}_{\text{te}})} \left[l\left(F(\mathcal{G}), \mathcal{Y}\right) \mid \mathcal{G}_{\text{tr}}, \mathcal{Y}_{\text{tr}} \right], \tag{1}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function. In this paper, we explore a setting where neither the test graphs $\mathcal{G}_{\text{te}}$ nor their corresponding labels $\mathcal{Y}_{\text{te}}$ are available during the training phase.

2.2 Differentiable GraphNAS

Unlike traditional GraphNAS approaches, which treat selecting the best architecture as a black-box optimization problem within a discrete domain, differentiable GraphNAS[29,44] relaxes the discrete search space into a continuous one

and allows for efficient optimization via gradient descent. We define the set of candidate operations as $\mathcal{O} = \{o_1, o_2, \ldots, o_K\}$, where each $o_k \in \mathcal{O}$ represents an operation from the search space, and K is the total number of operations in $\mathcal{O}$. Furthermore, differentiable GraphNAS relaxes the rigid selection of operations in $\mathcal{O}$ into a soft selection where each candidate is assigned a probability. Eq. (2) illustrates an example of differentiable search along the architecture space. The output of l^{th} layer can be represented as:

$$\mathbf{h}_i^{(l+1)} = \sum_{o \in \mathcal{O}} p^o o(\mathbf{h}_i^{(l)}), \tag{2}$$

where $\mathbf{h}_i^{(l)}$ represents the embedding of node i at the l^{th} layer, and p^o is the probability associated with the corresponding candidate operation o. The probability distribution across this dimension is normalized such that $\sum_{o \in \mathcal{O}} p^o = 1$. In each graph, all nodes share the same set of probabilities, and final architectures can be obtained by retaining the candidate operations with the highest probabilities during the testing process.

2.3 Power Law Distributions and Assortativity

Power Law Distribution In many real-world graphs, such as social networks and molecular networks, the degree distribution of nodes often follows a power law. This distribution indicates that the probability $P(k)$ of a node having k connections is proportional to $k^{-\alpha}$, where α is a positive constant:

$$P(k) \propto k^{-\alpha} \tag{3}$$

The power-law distribution reflects the heterogeneity of node connectivity, where a small number of nodes (i.e., hubs) account for the majority of edges, while most nodes exhibit a long-tail degree distribution, as illustrated in Fig. 1. To maximize the performance of GNNs, different message aggregation mechanisms can be employed for nodes at various positions within the power-law distribution.

However, differences in global topological characteristics necessitate a differentiated treatment of the power-law distribution across different types of graphs. Incorporating a global perspective enables the model to discern spurious motifs between the current graph structure and degree distribution.

Assortativity is a measure of the tendency of nodes in a graph to connect with other nodes that are similar in some specified attribute. This metric often reveals important structural patterns, such as the tendency of individuals in social networks to associate with others who are similar to themselves. Formally, the assortativity coefficient can be defined as:

$$q_j = \frac{j p_j}{\sum_k k p_k}, \tag{4}$$

$$\gamma = \frac{\sum_{jk} jk \left(e_{jk} - q_j q_k\right)}{\sigma_q^2}, \tag{5}$$

where p_j is the proportion of nodes of degree j, e_{jk} is the proportion of edges in the graph that connect nodes of degree j to nodes of degree k, and σ_q is the standard deviation of q. γ typically ranges from -1 to 1, where a positive γ indicates that high-degree nodes tend to connect with other high-degree nodes, while a negative γ suggests they connect with low-degree nodes. We leverage γ to enable NodeNAS to learn invariant representations of graphs with distribution shifts, thus enabling the architectures searched with generalizability.

3 Node-Specific Graph Neural Architecture Search

NodeNAS introduces a paradigm shift from traditional GNN architectures, where a singular embedding update method is uniformly applied all nodes in the graph within each layer. In contrast, NodeNAS proposes a flexible and adaptive framework that dynamically tailors the appropriate node information aggregation method based on the specific needs of each node within the given graph. Specifically, given a graph $\mathcal{G} = \{V, E\}$ with V denoting the set of nodes and E denoting the set of edges, NodeNAS aims to learn an architecture mapping function $\Phi : \mathcal{G} \rightarrow \mathcal{A} \times W_{\mathcal{A}}$, where $\mathcal{A}$ represents the architecture for each node i.e., $\mathcal{A} = \{A_1, A_2, ..., A_N\}$, and $W_{\mathcal{A}}$ the associated weights. $N = |V|$ denotes the number of nodes in $\mathcal{G}$.

To ensure the differentiability of NodeNAS, we also introduce weight sharing and assign each node a probability vector $\mathbf{p}$. $\mathbf{p}$ represents the probabilities of different operations, such as $GATConv$, being applied to the node. Define the set of candidate operations as $\mathcal{O} = \{o_1, o_2, \ldots, o_K\}$, where each o_k in $\mathcal{O}$ represents an operation in the search space, and K is the total number of operations. A_i can be further express as $A_i = \{(o, p_i^o)\}$, where o is the candidate operations satisfying $o \in \mathcal{O}$ and p_i^o is corresponding probability. In l^{th} layer, the embedding update for the node i can be expressed as:

$$\mathbf{h}_i^{(l+1)} = \sum_{o \in \mathcal{O}} p_i^o o(\mathbf{h}_i^{(l)}), \tag{6}$$

where $\mathbf{h}_i^{(l)}$ represents the embedding of node i at the l^{th} layer. The weights of operation o are shared across different graphs, ensuring that node-specific architectures can be tailored end-to-end for each graph, while enabling efficient optimization through gradient descent.

4 Adaptive Aggregation Attention Based Multi-Dim NodeNAS

4.1 Framework

In our proposed method, we tailor an unique node-specific architecture for each graph by maximizing the learning of intrinsic relationships between node and graph distributions from limited datasets. In particular, our method supports

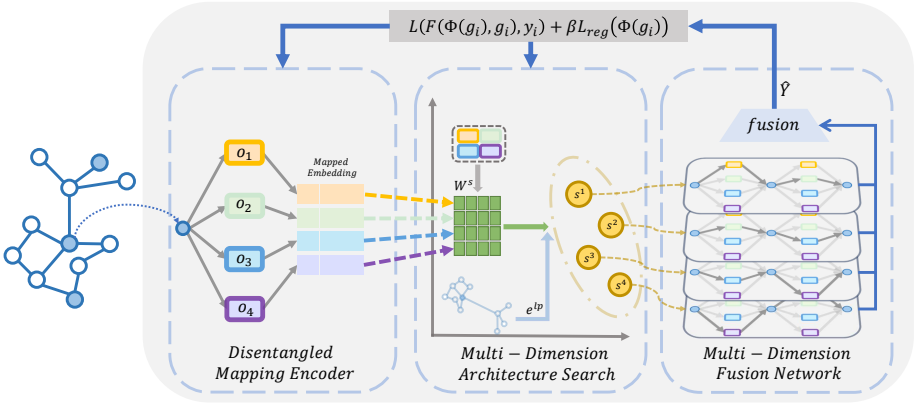

Fig. 2: An overview of our proposed MNNAS model.

searching the architectures across multiple dimensions, allowing for more flexibility in expressing learned decoupled information in architecture customization.

Specifically, we aim to learn an architecture mapping function $\Phi : G \rightarrow A \times \mathcal{W}_A$. Unlike Section 3, A_i of node i contains a set of architectures composed of multiple search dimensions, i.e., $A_i = \{A_i^1, A_i^2, \ldots, A_i^Z\}$, where Z is the number of S-Dims and $A_i^z = \{(o_k, p^{z,o_k})\}$ represents the architecture searched in the z^{th} S-Dim. $|A_i^z| = |\mathbf{p}| = K$ always holds for any z, indicating that the search space within each S-Dim includes the full set of candidate operations, thereby enabling the model to learn architectural preferences differentiably for different nodes across graphs. Φ can further be decomposed into a set of mappings for each S-Dim, i.e., $\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_Z\}$, with each Φ_z mapping the graph to a suitable architecture A_z and corresponding weights $\mathcal{W}_{A_z}$ in z^{th} S-Dim. Therefore, Eq.1 can be transformed into the following form:

$$\min_{\Phi \subset S_p} \sum_{(g_i\ y_i) \in \mathcal{G}_{\text{tr}}} [\mathcal{L}(F(\sum_{z=1}^{Z} \Phi_z(g_i), g_i), y_i) + \beta \mathcal{L}_{\text{reg}}(\Phi(g_i)], \tag{7}$$

where β is a hyperparameter and $\mathcal{L}_{\text{reg}}(\Phi(g_i))$ is the regularizer for the architectures. S_p represents the search space, a two-dimensional space composed of the operation plane and the probability plane, and $F(\cdot)$ is used to obtain the output for downstream tasks under the tailored architectures and weights.

For each $g_j \in G_{\text{te}}$, we utilize the trained Φ to generate unique architectures that are tailored to g_j and perform well under distribution shifts. Especially, for certain unsupervised tasks, such as community detection and graph partitioning, $\mathcal{G}_{\text{tr}}$ contains only a single graph and y_i is not required.

4.2 Disentangled Mapping Encoder

We aims for the encoder to rapidly aggregate overall semantic features. However, in GNNs, information is primarily propagated through edges, necessitating a

deep convolutional network to capture global information. Moreover, the multi-layer fixed GNN architecture exacerbates the nonlinear dependency between the input features and the adjacency matrix, which arises due to spurious motifs in the distribution. Therefore, our encoder only takes features as input and manually aggregates the global semantic information to accelerate information propagation between nodes. Specifically, our encoder is designed as follows:

$$\mathbf{h}_i^{(l)} = \text{ENCODER}(\mathbf{h}_i^{(l-1)} + \eta \ \text{LINEAR}(\frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i^{(l-1)})), \tag{8}$$

where η is a initial hyperparameter and $\mathbf{h}_i^{(l)}$ is the embedding of node i at l^{th} layer. $\mathbf{h}_i^{(0)}$ is initialized with the features $\mathbf{x}_i$ of node i.

After obtaining the node embeddings, we map candidate operations to learnable embeddings for each node. Specifically, for each operation o in $\mathcal{O}$, we learn a corresponding embedding $\mathbf{e}_i^o$. During the architecture search, we replace $\mathcal{O}$ with the set E_i which is composed of mapped embeddings. The mapping from operations to embeddings denoted by $\Phi_{\mathcal{O}\to E}$ can be expressed as:

$$E_i = [\mathbf{e}_i^1, ..., \mathbf{e}_i^K] = [o_1(\mathbf{h}_i^{(l)}, ..., o_K(\mathbf{h}_i^{(l)})], \tag{9}$$

where $\mathbf{e}_i^k$ represents the mapped embedding of o_k for node i in l^{th} layer and k is the number of candidate operations. l is omitted here to simplify the expression. $\Phi_{\mathcal{O}\to E}$ allows the model to perform multi-dimension search under single-dimension computation complexity. This is facilitated by that the mapping is shared among different S-Dims, i.e. $\mathbf{e}_{i(dim_a)}^k = \mathbf{e}_{i(dim_b)}^k = \mathbf{e}_i^k$ always holds in both the a^{th} S-Dim and b^{th} S-Dim. Each operation can simultaneously participate in the architecture search of multiple S-Dims but is computed only once.

Furthermore, to prevent mode collapse, where the mapped embeddings of different operations trend to be indistinguishable during training, we incorporate a regularization term that leverages cosine distance to maintain diversity among mapped embeddings:

$$L_{\cos} = \sum_i \sum_{\substack{o,o'\in\mathcal{O} \\ o\neq o'}} \frac{\mathbf{e}_i^o \cdot \mathbf{e}_i^{o'}}{\|\mathbf{e}_i^o\|_2 \|\mathbf{e}_i^{o'}\|_2}, \tag{10}$$

where $\mathbf{e}_i^o$ and $\mathbf{e}_i^{o'}$ denote the embeddings of node i for operations o and o', respectively. $L_{\cos}$ ensures orthogonality of $\Phi_{\mathcal{O}\to E}$, facilitating targeted encoding of disentangled information within the graph.

4.3 Multi-Dimension Architecture Search

We propose a multi-dimension architecture customization method, mapping each graph to node-specific GNN architectures across multiple S-Dims, i.e., tailoring multiple sets of A_i^k. It includes two components: the link pattern encoder, which

learns structure-invariant information across different graph distributions that follow power-law distribution, and adaptive aggregation attention, which guides the model to search architecture across multiple S-Dims simultaneously.

Link Pattern Encoder With limited datasets, node-specific architecture search could better express the disentangled information, while the multi-dim architecture amplifies the generalization brought by NodeNAS. However, capturing such disentangled information hidden in the distribution is challenging. When addressing spurious correlations in the training set, it is crucial to distinguish the preferences of nodes with varying degrees within similar graphs and those of nodes with similar degrees across different graphs.

To address these issues, we incorporate the degree distribution of different nodes into the link pattern encoder and use assortativity to quantify the overall graph structure, promoting the disentanglement of nodes and graphs in terms of topology. Specially, the link pattern encoder can be express as:

$$\mathbf{e}_i^{lp} = \text{ENCODER}(\gamma_g,\ \frac{d_i^2}{\bar{d^2}},\ \frac{d_i}{\bar{d}},\ \frac{1}{|E|}\sum_{(a,b)\in E} d_a d_b),\ i \in g, \tag{11}$$

where d_a and d_b are the degrees of nodes a and b, respectively, connected by an edge. $\bar{d}$ and $\bar{d^2}$ are the mean degree and mean square degree, respectively, of all nodes in graph g where node i resides. Especially, we approximate assortativity coefficient as a function of degree statistics, i.e., rewriting Eq. 5 as follow:

$$\gamma_g \approx \frac{\frac{1}{|E|}\sum_{(a,b)\in E} d_a d_b - \left[\frac{1}{|E|}\sum_{(a,b)\in E}\frac{1}{2}(d_a + d_b)\right]^2}{\frac{1}{|E|}\sum_{(a,b)\in E}\frac{1}{2}(d_a^2 + d_b^2) - \left[\frac{1}{|E|}\sum_{(a,b)\in E}\frac{1}{2}(d_a + d_b)\right]^2}, \tag{12}$$

where the numerator denotes the actual versus expected differences in degrees of connected node and the denominator denotes the statistical properties of the degree distribution.

Adaptive Aggregation Attention Adaptive aggregation attention combines link pattern vectors $\mathbf{e}^{lp}$, searching probability values for each candidate operation in every S-Dim. Specifically, for node i in the z^{th} S-Dim, the probability of different operations in $\mathcal{O}$ can be calculated as follow:

$$\mathbf{s}_i^z = \mathbf{e}_i^{lp} \odot \frac{\mathbf{e}_i^z W^s (E_i)^\top}{\sqrt{d}},\ p_i^{z,o} = \frac{\exp(s_i^{z,o})}{\sum_{o'\in\mathcal{O}}\exp(s_i^{z,o'})}, \tag{13}$$

where $\odot$ denotes Hadamard Product and $\mathbf{s}_i^z$ denotes the operation preference vector representing the architecture subspace, which is the projection of candidate operations onto the z^{th} S-Dim. $p_i^{z,o}$ represents the probability of operation o in z^{th} S-Dim for node i. $d = |\mathbf{e}_i^z|$ is the dimension of the mapped embedding in Eq. 9, and W^s is the weight matrix satisfying $W^s \in \mathbb{R}^{d\times d}$. Eq. 13 imposes a constraint that the number of dimensions we search must always equal the number of candidate operations in $\mathcal{O}$, denoted as $|S_i| = Z = |\mathcal{O}| = K$. The attention indirectly provides a normalization constraint, preventing optimization

dilemmas that could arise from indiscriminately expanding search dimensions in the context of limited datasets.

Adaptive aggreation attention leverages the link pattern encoder to decorrelate node features from specific graph distributions and further quantify the importance of each mapped embedding within E_i. Essentially, this mechanism is based on $att \leftarrow q \times k$, centered on different mapped embeddings to maximize the probability of similar representations learned through various operations.

4.4 Multi-Dimension Fusion Network

We integrate architectures searched from different S-Dims into a continuous space, jointly considering architectures in all S-Dims to learn the final node representations. The node representations learned after the fusion of multi-dimension architectures can be calculated as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma(\frac{1}{Z}\sum_{z=1}^{Z}\sum_{o\in\mathcal{O}} p_i^{z,o} o(\mathbf{h}_i^{(l)})) + \mathbf{h}_i^{(l)}, \tag{14}$$

where $\mathbf{h}_i^{(l)}$ denotes the initial node embedding at the l^{th} layer, and $\mathbf{h}_i^{(l+1)}$ represents the output node representation. σ here denotes activation function. Due to the mapping function $\Phi_{\mathcal{O}\to E}$, we assign probabilities to mapped embeddings in S_p within each S-Dim rather than to the operations, avoiding redundant computations and unnecessary learnable parameters. Shortcut connections is utilized in Eq. 14 to prevent overfitting and mitigate the influence of irrelevant features.

Our approach eliminates the retraining and architecture discretization steps common in NAS methods by maintaining continuous architectures with weight-sharing across graphs. This enables end-to-end execution through shared network parameters, effectively creating an ensemble model that bypasses separate architecture-specific training.

4.5 Theoretical Insights

In this section, we present a theoretical analysis of how adaptive aggregation attention enhances the model's generalization capability through the lens of information bottleneck theory.

Let us consider a node i with an operation mapping set E_i and the corresponding output representation as Z_i. Thus, the mapping function can be expressed as $f(Z_i|E_i)$. We assume a distribution $E_i \sim \text{Gaussian}(E_i^{'}, \epsilon)$, where E_i represents the noisy input variable, $E_i^{'}$ is the invariant target variable, and ϵ denotes the variance of the Gaussian noise. The information bottleneck can be formulated as follows:

$$f_{IB}(Z_i|E_i) = \arg\min_{f(Z_i|E_i)} I(E_i, Z_i) - I(Z_i, E_i^{'}), \tag{15}$$

where $I(\cdot,\cdot)$ denotes the mutual information. In the Eq.13, we decompose $W^s = W^q(W^o)^\top$ and transform the probability of operation o to $(\mathbf{e}_i^z W^q)(\mathbf{e}_i^o(e_{i,o}^{lp} W^o))^\top$,

and we get $S_i = Q(E_i)K^\top(E_i, \mathbf{e}_i^{lp})$. Based on the derivation in [48,8], we derive the iterative process for optimizing Eq.15 and Eq.13.

$$Z_i = \sum_{o \in O} \frac{\exp(\mathbf{e}_i^z W^q)(\mathbf{e}_i^o (e_{i,o}^{lp} W^o))^\top)}{\sum_{o' \in O} \exp(\mathbf{e}_i^z W^q)(\mathbf{e}_i^{o'} (e_{i,o'}^{lp} W^o))^\top)} \mathbf{e}_i^o = \sum_{o \in O} p_i^{z,o} \mathbf{e}_i^o, \tag{16}$$

where Eq. 16 elucidates the relationship between attention mechanisms across operations and the information bottleneck. Furthermore, previous studies [43] has demonstrated the effectiveness of the information bottleneck for generalization, particularly in aiding GNNs to discard spurious features. Therefore, we assert that MNNAS facilitates generalization, a claim that is subsequently substantiated by extensive empirical evidence.

4.6 Complexity Analysis

Let $|V|$ and $|E|$ denote the number of nodes and edges in the graph, respectively. We define d_i as the dimensionality of the input features, d_e as the dimensionality of the initial node embeddings, d_m as the dimensionality of the mapped embeddings for the different operations in $\mathcal{O}$, and d_o as the dimensionality of the output.

Number of Learnable Parameters: In our framework, the node encoder module comprises $O(2d_i d_e)$ parameters, the module for learning mapped embeddings includes $O(|\mathcal{O}|d_e d_m)$ learnable parameters, the link pattern encoder contains $4|\mathcal{O}|$ parameters, the adaptive aggregation attention has $O(d_m^2)$ parameters, and the multi-dimension fusion network has no learnable parameters. The final output layer consists of $O(d_m d_o)$ parameters. For an η-layer network, the total number of learnable parameters is given by: $O(2d_i d_e + d_m d_o + \eta(d_m^2 + |\mathcal{O}|(4 + d_e d_m)))$.

Time Complexity: The time complexity of the node encoder is $O(|V|d_i d_e)$. For the mapped embeddings module, it is $O(|V||\mathcal{O}|d_e d_m)$. The multi-dimension architecture search module has a time complexity of $O(|V||\mathcal{O}|(d_m^2 + |\mathcal{O}|^2))$, and the multi-dimension fusion module's time complexity is $O(|V||\mathcal{O}|^2 d_m)$. The output layer has a time complexity of $O(|V|d_m d_o)$. Additionally, the time complexity for L_{cos} is $O(|V||\mathcal{O}|^2 d_e)$.

Given that different S-Dims share operation weights through Φ, and considering that $|\mathcal{O}|$ is a small constant, the time complexity for multi-dimension search remain equivalent to those for single-dimension search, thus not incurring additional training costs. Consequently, the total time complexity can be simplified to: $O(|V|(d_i d_e + |\mathcal{O}|d_e d_m + |\mathcal{O}|d_m^2 + d_m d_o))$,

5 Experiments

In this section, we report experimental results to verify the effectiveness of our model.

Table 1: Dataset Statistics.

Dataset	Motif	hiv	bace	sider	Cora	Cite	Pub	BA	ER	RR	NW
Setting	Sup.	Sup.	Sup.	Sup.	Unsup.	Unsup.	Unsup.	Unsup.	Unsup.	Unsup.	Unsup.
Task	Graph	Graph	Graph	Graph	Node	Node	Node	Node	Node	Node	Node
Graphs	18,000	41,127	1,513	1,427	1	1	1	1	1	1	1
Avg. Nodes	26.1	25.5	34.1	33.6	2,708	3,327	19,717	10,000	10,000	10,000	10,000
Avg. Edges	36.3	27.5	36.9	35.4	10,556	9,104	88,648	99,950	99,950	100,000	22,092
Classes	3	2	2	2	7	6	3	-	-	-	-

Dataset: We established both synthetic and real-world datasets to evaluate the performance of MNNAS in supervised graph classification as well as unsupervised community detection and inverse graph partitioning tasks, specifically under conditions involving distribution shifts.

For graph classification, we use the Spurious-Motif (Motif) synthetic dataset, which integrates base and motif shapes, and the OGBG-Mol datasets (hiv, bace, sider) for molecular property predictions. Community detection utilizes the Cora, CiteSeer, and PubMed datasets. For inverse graph partitioning tasks, synthetic datasets including Erdős-Rényi (ER), random regular (RR), Barabási-Albert (BA), and Newman-Watts-Strogatz (NW) graphs are employed.

Baselines: We compare our model with the following baselines.

1. **Outstanding GNNs:** Commonly used manually designed GNNs include GCN[13], GAT[33], GIN[40], GraphSAGE[9], GraphConv[23] , GAP [24] and ClusterNet[36]. Recently proposed methods like ASAP[30], DIR[37], PNA[3], and GSAT[21] have demonstrated strong performance in graph-level tasks with OOD settings.
2. **Outstanding NAS:** Our study evaluates six advanced NAS baselines, including DARTS[20] and five GraphNAS based algorithms: GraphNAS[6], PAS[35], GASSO[28], GRACES[29], and DCGAS[44], which is currently the state-of-the-art in GNAS.

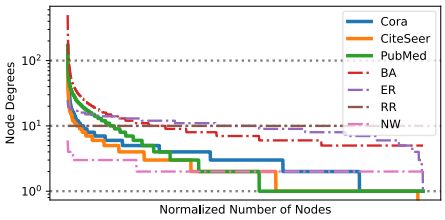

Fig. 3: Degree distribution of datasets.

5.1 Graph Classification on Synthetic and Real Datasets

Experimental Setting For the Spurious-Motif dataset, we followed the experimental setup outlined in[29]. Additionally, we performed experiments on real dataset OGBG-Mol. Each experiment was replicated ten times with different random seeds, and results are presented as averages with standard deviations.

Qualitative Results: Table 2 shows that our model outperforms baseline methods on both synthetic and real datasets. Traditional GNNs underperform on synthetic data due to spurious correlations and distribution shifts. In real datasets,

Table 2: Test Accuracy on Spurious-Motif and Test AUC-ROC on OGBG-Mol.

Method	**Spurious-Motif (Accuracy)**			**OGBG-Mol (AUC-ROC)**		
	b=0.7	b=0.8	b=0.9	hiv	sider	bace
GCN	$48.39_{\pm 1.69}$	$41.55_{\pm 3.88}$	$39.13_{\pm 1.76}$	$75.99_{\pm 1.19}$	$59.84_{\pm 1.54}$	$68.93_{\pm 6.95}$
GAT	$50.75_{\pm 4.89}$	$42.48_{\pm 2.46}$	$40.10_{\pm 5.19}$	$76.80_{\pm 0.58}$	$57.40_{\pm 2.01}$	$75.34_{\pm 2.36}$
GIN	$36.83_{\pm 5.49}$	$34.83_{\pm 3.10}$	$37.45_{\pm 3.59}$	$77.07_{\pm 1.49}$	$57.57_{\pm 1.56}$	$73.46_{\pm 5.24}$
SAGE	$46.66_{\pm 2.51}$	$44.50_{\pm 5.79}$	$44.79_{\pm 4.83}$	$75.58_{\pm 1.40}$	$56.36_{\pm 1.32}$	$74.85_{\pm 2.74}$
GraphConv	$47.29_{\pm 1.95}$	$44.67_{\pm 5.88}$	$44.82_{\pm 4.84}$	$74.46_{\pm 0.86}$	$56.09_{\pm 1.06}$	$78.87_{\pm 1.74}$
ASAP	$54.07_{\pm 13.85}$	$48.32_{\pm 12.72}$	$43.52_{\pm 8.41}$	$73.81_{\pm 1.17}$	$55.77_{\pm 1.18}$	$71.55_{\pm 2.74}$
DIR	$50.08_{\pm 3.46}$	$48.22_{\pm 6.27}$	$43.11_{\pm 5.43}$	$77.05_{\pm 0.57}$	$57.34_{\pm 0.36}$	$76.03_{\pm 2.20}$
DARTS	$50.63_{\pm 8.90}$	$45.41_{\pm 7.71}$	$44.44_{\pm 4.42}$	$74.04_{\pm 1.75}$	$60.64_{\pm 1.37}$	$76.71_{\pm 1.83}$
GraphNAS	$55.18_{\pm 18.62}$	$51.64_{\pm 19.22}$	$37.56_{\pm 5.43}$	-	-	-
PAS	$52.15_{\pm 4.35}$	$43.12_{\pm 5.95}$	$39.84_{\pm 1.67}$	$71.19_{\pm 2.28}$	$59.31_{\pm 1.48}$	$76.59_{\pm 1.87}$
GRACES	$65.72_{\pm 17.47}$	$59.57_{\pm 17.37}$	$50.94_{\pm 8.14}$	$77.31_{\pm 1.00}$	$61.85_{\pm 2.56}$	$79.46_{\pm 3.04}$
DCGAS	$87.68_{\pm 6.12}$	$75.45_{\pm 17.40}$	$61.42_{\pm 16.26}$	$\mathbf{78.04_{\pm 0.71}}$	$63.46_{\pm 1.42}$	$81.31_{\pm 1.94}$
MNNAS	$\mathbf{97.53_{\pm 3.65}}$	$\mathbf{98.42_{\pm 1.65}}$	$\mathbf{93.19_{\pm 6.17}}$	$76.55_{\pm 3.04}$	$\mathbf{65.46_{\pm 1.18}}$	$\mathbf{84.69_{\pm 3.67}}$

GNN effectiveness varies with graph characteristics. While NAS methods slightly improve upon manual GNN designs, they struggle with distribution changes. Conversely, MNNAS effectively reduces spurious correlations in graph distributions, notably improving performance, especially on synthetic datasets.

5.2 Community Detection

Experimental Setting We evaluate community detection performance on real datasets. Each dataset is partitioned into 10 communities as per the methodology described in the[36]. To standardize feature dimensions across datasets, we employ Non-negative Matrix Factorization instead of original node features.

Qualitative Results Table 3 illustrates a notable decline in generalization performance among manually designed GNN models during tests. These models often perform well on training datasets but fail to maintain efficacy on testing datasets. A similar trend is observed with differentiable NAS methods, indicating that applying a uniform GNN approach across an entire graph diminishes generalization due to varying node preferences. Conversely, our model consistently excels in both training and testing phases. The superior performance is attributed primarily to its capacity for node-specific architecture searches, which allows for the effective separation of invariant features specific to nodes.

5.3 Inverse Graph Partition

Experimental Setting To assess model generalization beyond typical power-law distributed datasets, we use four synthetic graphs with diverse distributions:

Table 3: Test Modularity on the real-world datasets.

Method	$\mathbf{Cora_{tr}}$	$\mathbf{Cite_{te}}$	$\mathbf{Pub_{te}}$	$\mathbf{Cite_{tr}}$	$\mathbf{Cora_{te}}$	$\mathbf{Pub_{te}}$	$\mathbf{Pub_{tr}}$	$\mathbf{Cite_{te}}$	$\mathbf{Cora_{te}}$
GCN	$0.65_{\pm 0.02}$	$0.56_{\pm 0.01}$	$0.50_{\pm 0.02}$	$0.65_{\pm 0.01}$	$0.58_{\pm 0.02}$	$0.50_{\pm 0.02}$	$0.64_{\pm 0.01}$	$0.55_{\pm 0.03}$	$0.54_{\pm 0.02}$
GAT	$0.59_{\pm 0.03}$	$0.52_{\pm 0.05}$	$0.42_{\pm 0.04}$	$0.61_{\pm 0.04}$	$0.50_{\pm 0.05}$	$0.48_{\pm 0.02}$	$0.60_{\pm 0.02}$	$0.52_{\pm 0.01}$	$0.49_{\pm 0.04}$
GIN	$0.58_{\pm 0.08}$	$0.52_{\pm 0.06}$	$0.45_{\pm 0.03}$	$0.66_{\pm 0.03}$	$0.52_{\pm 0.03}$	$0.41_{\pm 0.05}$	$0.56_{\pm 0.08}$	$0.53_{\pm 0.07}$	$0.49_{\pm 0.08}$
SAGE	$0.60_{\pm 0.03}$	$0.47_{\pm 0.02}$	$0.44_{\pm 0.03}$	$0.64_{\pm 0.03}$	$0.51_{\pm 0.05}$	$0.50_{\pm 0.03}$	$0.60_{\pm 0.03}$	$0.48_{\pm 0.04}$	$0.48_{\pm 0.03}$
GraphConv	$0.57_{\pm 0.03}$	$0.42_{\pm 0.03}$	$0.41_{\pm 0.04}$	$0.45_{\pm 0.22}$	$0.30_{\pm 0.16}$	$0.29_{\pm 0.15}$	$0.53_{\pm 0.02}$	$0.41_{\pm 0.03}$	$0.39_{\pm 0.03}$
MLP	$0.64_{\pm 0.02}$	$0.48_{\pm 0.03}$	$0.28_{\pm 0.03}$	$0.66_{\pm 0.04}$	$0.47_{\pm 0.05}$	$0.30_{\pm 0.06}$	$0.58_{\pm 0.04}$	$0.44_{\pm 0.05}$	$0.34_{\pm 0.03}$
ClusterNet	$0.59_{\pm 0.02}$	$0.56_{\pm 0.06}$	$0.42_{\pm 0.04}$	$0.70_{\pm 0.03}$	$0.46_{\pm 0.01}$	$0.31_{\pm 0.06}$	$0.61_{\pm 0.02}$	$0.58_{\pm 0.06}$	$0.47_{\pm 0.03}$
DARTS	$0.66_{\pm 0.02}$	$0.53_{\pm 0.03}$	$0.42_{\pm 0.06}$	$0.67_{\pm 0.02}$	$0.53_{\pm 0.03}$	$0.47_{\pm 0.02}$	$0.62_{\pm 0.04}$	$0.54_{\pm 0.02}$	$0.51_{\pm 0.03}$
GASSO	$0.63_{\pm 0.03}$	$0.58_{\pm 0.04}$	$0.52_{\pm 0.03}$	$0.68_{\pm 0.03}$	$0.57_{\pm 0.04}$	$\mathbf{0.53_{\pm 0.04}}$	$0.66_{\pm 0.04}$	$0.61_{\pm 0.03}$	$0.59_{\pm 0.03}$
MNNAS	$\mathbf{0.69_{\pm 0.02}}$	$\mathbf{0.63_{\pm 0.02}}$	$\mathbf{0.53_{\pm 0.01}}$	$\mathbf{0.72_{\pm 0.02}}$	$\mathbf{0.60_{\pm 0.01}}$	$0.52_{\pm 0.03}$	$\mathbf{0.68_{\pm 0.01}}$	$\mathbf{0.61_{\pm 0.01}}$	$\mathbf{0.61_{\pm 0.02}}$

Table 4: Test Ratio of inter-subgraph to total edges on the synthetic datasets.

Method	$\mathbf{BA1000_{tr}}$	$\mathbf{BA10000_{te}}$	$\mathbf{RR10000_{te}}$	$\mathbf{ER10000_{te}}$	$\mathbf{NW10000_{te}}$
GCN	$0.95_{\pm 0.01}$	$0.86_{\pm 0.10}$	$0.87_{\pm 0.07}$	$0.87_{\pm 0.06}$	$0.81_{\pm 0.11}$
GAT	$0.95_{\pm 0.01}$	$0.87_{\pm 0.05}$	$0.75_{\pm 0.07}$	$0.75_{\pm 0.07}$	$0.72_{\pm 0.09}$
SAGE	$\mathbf{0.99_{\pm 0.00}}$	$0.95_{\pm 0.01}$	$0.92_{\pm 0.04}$	$0.90_{\pm 0.05}$	$0.80_{\pm 0.09}$
GIN	$\mathbf{0.99_{\pm 0.00}}$	$0.92_{\pm 0.00}$	$0.80_{\pm 0.05}$	$0.81_{\pm 0.04}$	$0.66_{\pm 0.09}$
GraphConv	$\mathbf{0.99_{\pm 0.00}}$	$0.94_{\pm 0.01}$	$0.87_{\pm 0.05}$	$0.87_{\pm 0.04}$	$0.81_{\pm 0.04}$
MLP	$0.98_{\pm 0.00}$	$0.92_{\pm 0.01}$	$0.91_{\pm 0.01}$	$0.90_{\pm 0.01}$	$0.76_{\pm 0.06}$
GAP	$0.96_{\pm 0.01}$	$0.91_{\pm 0.02}$	$0.90_{\pm 0.02}$	$0.90_{\pm 0.02}$	$0.81_{\pm 0.10}$
MNNAS	$\mathbf{0.99_{\pm 0.00}}$	$\mathbf{0.96_{\pm 0.00}}$	$\mathbf{0.98_{\pm 0.00}}$	$\mathbf{0.97_{\pm 0.00}}$	$\mathbf{0.84_{\pm 0.05}}$

BA, ER, RR, and NW. Training is conducted on a BA graph with 1,000 nodes, while testing uses larger graphs of 10,000 nodes . Each graph is split into 10 subgraphs to maximize inter-subgraph edges and minimize intra-subgraph edges.
Qualitative Results The results in Table 4 demonstrate that employing a single GNN for generalization across datasets with diverse node distributions significantly degrades performance. This is due to the requirement for models to adapt to varying graph distributions globally and to distinct structural node features locally. Our proposed MNNAS model addresses this challenge by incorporating a link pattern encoder that decouples invariant structural factors, allowing for tailored architecture customization at the node level.

5.4 Interpretability

In Fig. 4, we illustrates the architectural preferences of nodes across four datasets, categorized into five groups based on their degree, from highest to lowest. For Cora, high-degree nodes favor Linear architectures, while low-degree nodes prefer GINConv. In Citeseer, as the degree distribution shifts, architectural preferences transition from GCNConv to GraphConv; similar patterns are observed in

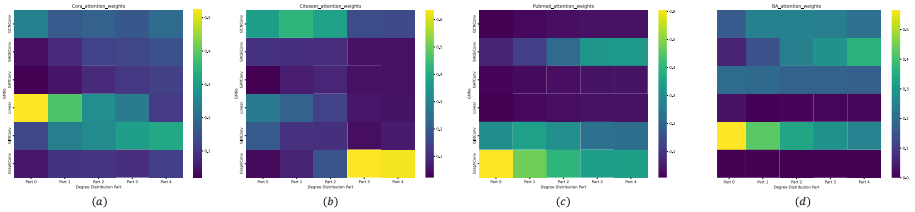

Fig. 4: (a)-(d) illustrate the statistically derived architecture customization patterns of MNNAS for different degree distributions across datasets

Pubmed and BA. These results indicate that nodes within each graph exhibit distinct architectural preferences influenced by their degree, underscoring the significance of power-law distributions. Moreover, nodes across different graphs demonstrate varying preferences, highlighting the effectiveness of incorporating graph-level topological features, such as assortativity, in differentiating power-law distributions across diverse graphs.

6 Related Work

6.1 Graph Neural Architecture Search

In recent years, Neural Architecture Search has demonstrated potential in automatically identifying optimal solutions for machine learning tasks, alleviating the burden of manual hyperparameter tuning while ensuring optimal solutions can be obtained without extensive expert knowledge. Various search strategies have been proposed to rapidly pinpoint the best design, including reinforcement learning based NAS [50,27,2,1], evolutionary algorithms based NAS [22,31,19,38], and Bayesian optimization based NAS [12]. ENAS [27] introduced the concept of parameter sharing, highlighting that all designs share weights to improve search efficiency. Subsequently, some works have transformed discrete search of NAS into a differentiable search to accelerate the process end-to-end, such as DARTS [20] and SNAS [39], which aggregate excellent GNN architecture operations and weight all operations in the search space.

Inspired by NAS success stories, Graph Neural Architecture Search has garnered widespread attention, giving rise to many outstanding GraphNAS works. GraphNAS [6] first used reinforcement learning to aggregate excellent GNN architectures as the search space. AGNN [49] employed a Recurrent Neural Network controller to eliminate noise in architecture search caused by modifying aggregation functions. MoHINRec [46] proposed a GraphNAS method that simplifies the search space and applies it to recommendation systems to enhance accuracy. Additionally, some works optimized the search process, such as evolutionary architecture search [32] using evolutionary algorithms to generate more diverse sub-designs, random search [45] sampling various designs with equal probability, and many excellent works [4,25,18]. Notably, PDNAS [47] first applied differentiable search to GraphNAS, expanding the discrete search space

into a continuous one through Gumbel-Sigmoid. This has heuristic significance for our work. Furthermore, GraphNAS focusing on graph classification tasks [35,11,26] has also been widely studied and applied in applications like chemical property classification of protein molecules. GRACES [29] is the first GraphNAS to consider distribution shifts, adapting architecture customization for non-I.I.D. graphs through a disentangled encoder.

6.2 GNN Generalization

Graph Neural Networks have representational dependencies and struggle to generalize to unknown network structures, often underperforming with non-I.I.D. graphs [10,14]. Recently, some cutting-edge works have attempted to design GNN architectures with higher generalization abilities to mitigate performance drops caused by distribution shifts, including GNNs combining stochastic attention mechanism [21], random Fourier features [42], Hilbert-Schmidt Independence Criterion [17] and so on. Some researchers have focused on the graph's local structure to analyze GNN generalization out of distribution, including designing generalization bounds [7], edge masking generators [5], and learning local adaptive structure perception [41].

Several works have tried differentiating treatments for various graph data, such as Policy-GNN [15] using reinforcement learning to determine the aggregation layers for each node, OODGNN [16] learning de-correlated graph representations through sample reweighting, and Customized-GNN [34] generating different training parameters for different graphs. GRACES [29] utilizes differentiable search-based GraphNAS to customize GNN architectures for different graphs, being the first algorithm to consider using architecture search to enhance GNN generalization capabilities. DCGAS [44] builds on GRACES by incorporating a diffusion model-based data augmentation module, effectively enhancing the classification accuracy on non-I.I.D. graphs.

7 Conclusion

In this paper, we present the Multi-dimension Node-specific graph Neural Architecture Search (MNNAS) framework, designed to enhance the generalization capabilities of GNAS in the face of distribution shifts. By customizing node-specific architectures that reflect the inherent variability of graph structures, MNNAS overcomes the limitations of existing GNAS methods, which typically depend on large training datasets and struggle with distribution patterns across different graphs. Our extensive experimental evaluations demonstrate that MNNAS, incorporating an adaptive aggregation attention mechanism and modeling power-law distributions, achieves superior performance across various supervised and unsupervised tasks under out-of-distribution conditions.

References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 (2016)
2. Cai, H., Chen, T., Zhang, W., Yu, Y., Wang, J.: Efficient architecture search by network transformation. In: Proc. of AAAI (2018)
3. Corso, G., Cavalleri, L., Beaini, D., Liò, P., Veličković, P.: Principal neighbourhood aggregation for graph nets. Proc. of NeurIPS pp. 13260–13271 (2020)
4. Ding, Y., Yao, Q., Zhang, T.: Propagation model search for graph neural networks. arXiv preprint arXiv:2010.03250 (2020)
5. Fan, S., Wang, X., Mo, Y., Shi, C., Tang, J.: Debiasing graph neural networks via learning disentangled causal substructure. Proc. of NeurIPS pp. 24934–24946 (2022)
6. Gao, Y., Yang, H., Zhang, P., Zhou, C., Hu, Y.: Graph neural architecture search. In: Proc. of IJCAI (2021)
7. Garg, V., Jegelka, S., Jaakkola, T.: Generalization and representational limits of graph neural networks. In: Proc. of ICML. pp. 3419–3430 (2020)
8. Guo, K., Wen, H., Jin, W., Guo, Y., Tang, J., Chang, Y.: Investigating out-of-distribution generalization of gnns: An architecture perspective. In: Proc. of KDD. pp. 932–943 (2024)
9. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Proc. of NeurIPS (2017)
10. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. Proc. of NeurIPS pp. 22118–22133 (2020)
11. Jiang, S., Balaprakash, P.: Graph neural network architecture search for molecular property prediction. In: 2020 IEEE International conference on big data (big data). pp. 1346–1353 (2020)
12. Jin, H., Song, Q., Hu, X.: Auto-keras: An efficient neural architecture search system. In: Proc. of KDD. pp. 1946–1956 (2019)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Koh, P.W., Sagawa, S., Marklund, H., Xie, S.M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R.L., Gao, I., et al.: Wilds: A benchmark of in-the-wild distribution shifts. In: Proc. of ICML. pp. 5637–5664 (2021)
15. Lai, K.H., Zha, D., Zhou, K., Hu, X.: Policy-gnn: Aggregation optimization for graph neural networks. In: Proc. of KDD. pp. 461–471 (2020)
16. Li, H., Wang, X., Zhang, Z., Zhu, W.: Ood-gnn: Out-of-distribution generalized graph neural network. IEEE Transactions on Knowledge and Data Engineering (2022)
17. Li, H., Zhang, Z., Wang, X., Zhu, W.: Disentangled graph contrastive learning with independence promotion. IEEE Transactions on Knowledge and Data Engineering (2022)
18. Li, Y., King, I.: Autograph: Automated graph neural network. In: Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 23–27, 2020, Proceedings, Part II 27. pp. 189–201 (2020)
19. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436 (2017)

20. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
21. Miao, S., Liu, M., Li, P.: Interpretable and generalizable graph learning via stochastic attention mechanism. In: Proc. of ICML. pp. 15524–15543 (2022)
22. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al.: Evolving deep neural networks. In: Artificial intelligence in the age of neural networks and brain computing, pp. 269–287 (2024)
23. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proc. of AAAI. pp. 4602–4609 (2019)
24. Nazi, A., Hang, W., Goldie, A., Ravi, S., Mirhoseini, A.: Gap: Generalizable approximate graph partitioning framework. arxiv 2019. arXiv preprint arXiv:1903.00614
25. Nunes, M., Pappa, G.L.: Neural architecture search in graph neural networks. In: Intelligent Systems: 9th Brazilian Conference, BRACIS 2020, Rio Grande, Brazil, October 20–23, 2020, Proceedings, Part I 9. pp. 302–317 (2020)
26. Peng, W., Hong, X., Chen, H., Zhao, G.: Learning graph convolutional network for skeleton-based human action recognition by neural searching. In: Proc. of AAAI. pp. 2669–2676 (2020)
27. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: Proc. of ICML. pp. 4095–4104 (2018)
28. Qin, Y., Wang, X., Zhang, Z., Zhu, W.: Graph differentiable architecture search with structure learning. Proc. of NeurIPS pp. 16860–16872 (2021)
29. Qin, Y., Wang, X., Zhang, Z., Xie, P., Zhu, W.: Graph neural architecture search under distribution shifts. In: Proc. of ICML. pp. 18083–18095 (2022)
30. Ranjan, E., Sanyal, S., Talukdar, P.: Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In: Proc. of AAAI. pp. 5470–5477 (2020)
31. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proc. of AAAI. pp. 4780–4789 (2019)
32. Shi, M., Tang, Y., Zhu, X., Huang, Y., Wilson, D., Zhuang, Y., Liu, J.: Genetic-gnn: Evolutionary architecture search for graph neural networks. Knowledge-Based Systems p. 108752 (2022)
33. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
34. Wang, Y., Ma, Y., Jin, W., Li, C., Aggarwal, C., Tang, J.: Customized graph neural networks. arXiv preprint arXiv:2005.12386 (2020)
35. Wei, L., Zhao, H., Yao, Q., He, Z.: Pooling architecture search for graph classification. In: Proc. of CIKM. pp. 2091–2100 (2021)
36. Wilder, B., Ewing, E., Dilkina, B., Tambe, M.: End to end learning and optimization on graphs. Proc. of NeurIPS (2019)
37. Wu, Y.X., Wang, X., Zhang, A., He, X., seng Chua, T.: Discovering invariant rationales for graph neural networks. In: ICLR (2022)
38. Xie, L., Yuille, A.: Genetic cnn. In: Proc. of ICCV. pp. 1379–1388 (2017)
39. Xie, S., Zheng, H., Liu, C., Lin, L.: Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926 (2018)
40. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)

41. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Proc. of ICML. pp. 5453–5462 (2018)
42. Xu, K., Zhang, M., Li, J., Du, S.S., Kawarabayashi, K.i., Jegelka, S.: How neural networks extrapolate: From feedforward to graph neural networks. arXiv preprint arXiv:2009.11848 (2020)
43. Yang, L., Zheng, J., Wang, H., Liu, Z., Huang, Z., Hong, S., Zhang, W., Cui, B.: Individual and structural graph information bottlenecks for out-of-distribution generalization. IEEE Transactions on Knowledge and Data Engineering (2023)
44. Yao, Y., Wang, X., Qin, Y., Zhang, Z., Zhu, W., Mei, H.: Data-augmented curriculum graph neural architecture search under distribution shifts (2024)
45. You, J., Ying, Z., Leskovec, J.: Design space for graph neural networks. Proc. of NeurIPS pp. 17009–17021 (2020)
46. Zhao, H., Zhou, Y., Song, Y., Lee, D.L.: Motif enhanced recommendation over heterogeneous information network. In: Proc. of CIKM. pp. 2189–2192 (2019)
47. Zhao, Y., Wang, D., Gao, X., Mullins, R., Lio, P., Jamnik, M.: Probabilistic dual network architecture search on graphs. arXiv preprint arXiv:2003.09676 (2020)
48. Zhou, D., Yu, Z., Xie, E., Xiao, C., Anandkumar, A., Feng, J., Alvarez, J.M.: Understanding the robustness in vision transformers. In: Proc. of ICML. pp. 27378–27394 (2022)
49. Zhou, K., Huang, X., Song, Q., Chen, R., Hu, X.: Auto-gnn: Neural architecture search of graph neural networks. Frontiers in big Data p. 1029307 (2022)
50. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proc. of CVPR. pp. 8697–8710 (2018)