AutoGFM: Automated Graph Foundation Model with Adaptive Architecture Customization

Haibo Chen [1] **Xin Wang** [1] **Zeyang Zhang** [1] **Haoyang Li** [1] **Ling Feng** [1] **Wenwu Zhu** [1]

Abstract

Graph foundation models (GFMs) aim to share graph knowledge across diverse domains and tasks to boost graph machine learning. However, existing GFMs rely on hand-designed and fixed graph neural network (GNN) architectures, failing to utilize optimal architectures *w.r.t.* specific domains and tasks, inevitably leading to suboptimal performance in diverse graph domains and tasks. In this paper, we explore graph neural architecture search (GNAS) for GFMs for the first time, which suffers from the problem of *architecture inconsistency*, i.e., the optimal architectures for different tasks and domains vary. We tackle this problem by discovering an invariant graph-architecture relationship across domains and tasks, which imposes three challenges: i) how to capture invariant and variant patterns; ii) how to customize architectures to adapt to diverse domains and tasks; iii) how to mitigate the data domination phenomenon during the architecture search process. To address these challenges, we propose Automated Graph Foundation Model with Adaptive Architecture Customization (**AutoGFM**), providing a theoretical analysis to demonstrate the limitations of existing GNAS. Specifically, we first propose a disentangled contrastive graph encoder to learn invariant and variant patterns. Then, we design an invariant-guided architecture customization strategy to customize architectures for data from diverse domains and tasks. Finally, we propose a curriculum architecture customization mechanism to mitigate the phenomenon of particular data dominating the search process. Extensive experiments demonstrate that **AutoGFM** outperforms baselines, achieving state-of-the-art performance.

[1]Department of Computer Science and Technology, BNRIST, Tsinghua University, Beijing, China. Correspondence to: Xin Wang <xin wang@tsinghua.edu.cn>, Wenwu Zhu <wwzhu@tsinghua.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

1. Introduction

Graph foundation models (GFMs) (Liu et al., 2023b; Xu et al., 2024; Kong et al., 2024) aim to share graph knowledge across diverse graph domains and tasks. GNN-based GFMs (Liu et al., 2023a; Wang et al., 2024b) represent a promising direction, as they enable the transfer of shared knowledge across various domains and tasks, allowing a single graph neural network (GNN) to handle node-level, edge-level, and graph-level tasks across various domains. Specifically, GNN-based GFMs leverage large language models (LLMs) as enhancers, transform the textual features of graphs into unified representations, and unify graph-related tasks through subgraph classification for GNNs.

However, data from different tasks and domains may require different graph neural architectures. For instance, the vanilla GCN (Kipf & Welling, 2017) outperforms GraphSAGE (Hamilton et al., 2017) in the citation network OGBN-arxiv, while failing to demonstrate satisfactory performance in OGBN-proteins (Hu et al., 2020). Since existing GNN-based GFMs rely on hand-designed and fixed GNN architectures, they inevitably fail to adapt to the specific architecture requirements for diverse domains and tasks.

In this paper, we explore the problem of graph neural architecture search (GNAS) for GNN-based graph foundation models, which suffers from the problem of *architecture inconsistency*, i.e., the optimal architecture for different tasks and domains varies. We further leverage a representative group of differentiable graph neural architecture search methods (Liu et al., 2018) as an example and provide theoretical analysis demonstrating their inability to effectively search for graph neural architectures for GFMs under *architecture inconsistency*, resulting in suboptimal architectures. We tackle this problem by discovering an invariant graph-architecture relationship across domains and tasks, which imposes three challenges: i) how to capture invariant and variant patterns, which are entangled in graph data; ii) how to customize graph neural architectures based on the discovered patterns to adapt to data with diverse domains and tasks; iii) how to mitigate the phenomenon of data domination during the architecture search process.

To address these challenges, we propose a novel Automated

Graph Foundation Model with Adaptive Architecture Customization (**AutoGFM**), which customizes graph neural architectures for graph data across diverse tasks and domains. The core idea is to train an architecture mapping function π, which maps $\mathcal{G}$ (graph data) $\rightarrow \mathcal{A}$ (architecture), enabling the customization of architectures for each dataset to address *architecture inconsistency*, while simultaneously facilitating mutual knowledge sharing across diverse domains and tasks within a weight-sharing super-network. Specifically, we first propose a disentangled contrastive graph encoder to learn invariant and variant patterns from graph data. To achieve this, we design a subgraph-level discriminative contrastive learning that captures the invariant and variant patterns from diverse graph data. Second, we propose an invariant-guided architecture customization to tailor graph neural architectures for diverse data. We encourage invariant patterns to retain their ability to customize architectures despite the interference from variant patterns, aiming to eliminate the spurious effects brought by variant patterns. Finally, we propose a curriculum architecture customization mechanism to mitigate the phenomenon of some particular data dominating the search process. We design a curriculum constraint to promote the diversity of customized architectures across different datasets. Extensive experiments demonstrate that our **AutoGFM** model outperforms existing baselines, achieving state-of-the-art performance. The contributions of this paper are summarized as follows:

- We propose to explore the problem of graph neural architecture search for GNN-based graph foundation model, to the best of our knowledge, for the first time.
- We propose Automated Graph Foundation Model with Adaptive Architecture Customization (**AutoGFM**), analyzing the problem of *architecture inconsistency* for GFM and providing a theoretical analysis to demonstrate the limitations of existing mainstream differentiable GNAS methods under such conditions.
- We propose three novel modules to tackle the problem of *architecture inconsistency*, i) disentangled contrastive graph encoder, ii) invariant-guided architecture customization, and iii) curriculum architecture customization mechanism.
- We conduct extensive experiments on eight datasets to demonstrate the superiority of our method over state-of-the-art baselines.

2. Problem Formulation

In this section, we introduce the fundamental concepts and notations used in this paper, including graph data definition, node of interest (NOI) graph, graph neural architecture search, and GNAS for GNN-based GFMs.

2.1. Graph Data Definition

Text-attributed Graphs (TAGs) A text-attributed graph (TAG) is a graph where each node and edge is associated with a text sentence (Liu et al., 2023a). We denote a TAG is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V} = v_1, \ldots, v_{|\mathcal{V}|}$ represents the set of nodes, $\mathcal{E} = e_1, \ldots, e_{|\mathcal{E}|}$ represents the set of edges, and $\mathcal{R} = r_1, \ldots, r_{|\mathcal{R}|}$ represents the set of relations.

Node of Interest (NOI) Subgraph Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$. Following the previous work (Liu et al., 2023a; Wang et al., 2024b), we define the subgraph to unify graph tasks as node of interest subgraph (NOI-graph). An NOI-graph G_h is defined as the subgraph around the NOI. Denote $S_h(v) = \{\mathcal{V}_v^h, \mathcal{E}_v^h, \mathcal{R}_v^h\}$ as the h-hop ego-subgraph around v, consisting of h-hop neighbor nodes of v and all interconnecting edges. For node-level tasks on a node v, the NOI is the node itself, such that $\mathcal{T} = \{v\}$ and $G_h(\mathcal{T}) = S_h(v)$. For link-level tasks involving a node pair (v_i, v_j), we define $\mathcal{T} = \{v_i, v_j\}$, and the NOI-graph is $G_h(\{v_i, v_j\}) = S_h(v_i) \cup S_h(v_j)$. For graph-level tasks, the NOI includes all nodes in the graph, making the NOI-graph $G_h(\mathcal{V}) = (\mathcal{V}, \mathcal{E}, \mathcal{R})$. We define an NOI-graph $G_h(\mathcal{T})$ as:

$$\begin{aligned} G_h(\mathcal{T}) &= \cup_{v \in \mathcal{T}} S_h(v) \\ &= \big(\cup_{v \in \mathcal{T}} \mathcal{V}_v^h, \cup_{v \in \mathcal{T}} \mathcal{E}_v^h, \cup_{v \in \mathcal{T}} \mathcal{R}_v^h \big). \end{aligned} \tag{1}$$

2.2. Graph Neural Architecture Search

Given a data $\mathcal{D} = (\mathcal{G}, \mathcal{Y})$ for a graph neural architecture search (GNAS), we aim to search for a function $F_{\alpha,w} : \mathcal{G} \rightarrow \mathcal{Y}$, with architecture parameters $\alpha \in \mathcal{A}$ and learnable weights $w \in \mathcal{W}$, where $\mathcal{A}$ is the architecture space and $\mathcal{W}$ is the weight space:

$$\alpha^* = \arg\min_{\alpha \in \mathcal{A}} \mathcal{L}(F_{\alpha, w^*(\alpha)}(\mathcal{G}), \mathcal{Y}), \tag{2}$$

$$\text{s.t.} \quad w^*(\alpha) = \arg\min_{w \in \mathcal{W}(\alpha)} \mathcal{L}(F_{\alpha,w}(\mathcal{G}), \mathcal{Y}), \tag{3}$$

where $\mathcal{L}$ represents the loss of predictions made by the architecture $F_{\alpha,w}(\cdot)$ on the graph, and α^* and w^* denote the optimal architecture and weights for the given data $\mathcal{D} = (\mathcal{G}, \mathcal{Y})$. Specifically, α typically represents the selection of GNN operations (e.g., GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2017), GIN (Xu et al., 2018), etc.), which are referred to as operation choices for brevity. GNAS addresses this as a bi-level optimization problem (Elsken et al., 2019).

2.3. GNAS for GNN-based GFMs

We define diverse data as $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$, where $\mathcal{D}_i = \{\mathcal{G}_i, \mathcal{Y}_i\}$ represents the i-th dataset with graph $\mathcal{G}_i$ and label $\mathcal{Y}_i$. Following previous work for GNN-based GFMs (Liu et al., 2023a; Wang et al., 2024b), which leverage

LLMs to unify the node feature space across different graphs and leverage subgraphs to unify graph tasks, enabling a single GNN to be applied to diverse data $\mathcal{D}$ across domains and tasks. Graph neural architecture search for GFM aims to search a graph neural architecture $F_{\alpha,w}$ that achieves performance in diverse data $\mathcal{D}$.

3. Preliminaries

In this section, we first introduce the problem of *architecture inconsistency* in GFM. Then we provide an invariant view of architecture customization and formulate the overall objective for our proposed method.

3.1. Architecture Inconsistency in GFM

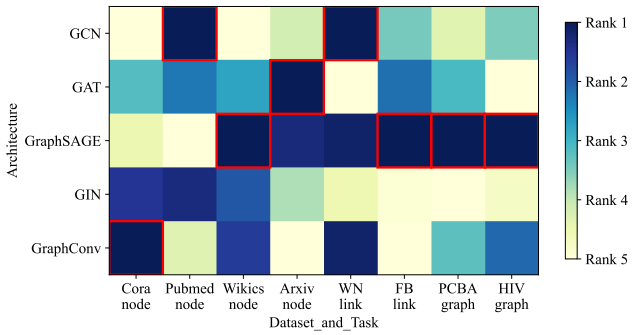

Figure 1. A heatmap visualization illustrating that the optimal architecture can vary across datasets with different domains and tasks. The darker the color of a block, the better the performance on the corresponding dataset. Additionally, red indicates the best-performing architectures for each dataset.

According to our observation, the optimal architecture for graph data across different tasks and domains may exhibit *architecture inconsistency*, meaning that the optimal architectures vary for data with diverse tasks and domains. To validate this, we test various GNN architectures built upon a GNN-based GFM, GFT (Wang et al., 2024b), on datasets with different domains and tasks. We present the performance of each architecture on each dataset using a heatmap in Figure 1. As shown in Figure 1, datasets from different domains and tasks require distinct optimal architectures, highlighting the presence of *architecture inconsistency*. Based on this observation, we introduce the following assumption:

Assumption 3.1. There exist two datasets $\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D}$, the optimal operation required $\mathcal{D}_i$ is different from $\mathcal{D}_j$.

Assumption 3.1 assumes that the optimal architectures required by two different datasets may differ. Then we further provide theoretical analyses showing that *architecture inconsistency* leads to operations optimization conflicts in existing differentiated GNAS methods. We have the following proposition with proof in Appendix A.1.

Proposition 3.2. *If there exist two datasets $\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D}$, the optimal operation for $\mathcal{D}_i$ is different from $\mathcal{D}_j$, the operations will render optimization conflicts.*

Assumption 3.1 serves as a prerequisite condition for Proposition 3.2. Proposition 3.2 demonstrates that when two datasets require different optimal architectures, current mainstream GNAS methods encounter optimization conflicts for GFM. For instance, as illustrated in Figure 1, the optimal architectures for PubMed and Wikics differ. When existing GNAS methods search simultaneously for an architecture optimal for both datasets, they fail to identify a single architecture that performs best for both and are forced to compromise.

To tackle *architecture inconsistency*, our key idea is to train a mapping function π: $\mathcal{G} \rightarrow \mathcal{A}$, which customizes architectures for each data to prevent *architecture inconsistency*, while simultaneously facilitating knowledge sharing across domains and tasks via the weight-sharing supernetwork.

3.2. Invariant View of Architecture Customization

We customize graph neural architectures for graph data across diverse tasks and domains from an invariant perspective. Unlike conventional invariant inference approaches that aim to discover invariant relationships between data and labels (Wu et al., 2022b; Li et al., 2022a; Wu et al., 2022a), our goal is to identify invariant relationships between the graph data and the corresponding architecture, addressing the issue of *architecture inconsistency* by tailoring graph neural architectures for each data individually.

We formalize the four key variables: input graph data G, architecture A, invariant pattern Z_I, and variant pattern Z_V. We divide the architecture mapping function π into two components: encoder $\theta : \mathrm{G} \rightarrow \mathrm{Z}_I$ and predictor $\psi : \mathrm{Z}_I \rightarrow \mathrm{A}$. We make the following assumptions:

Assumption 3.3. (1) $\mathrm{Z}_I = \mathrm{G} \setminus \mathrm{Z}_V$. There are two disjoint parts in the graph data G: invariant part Z_I and variant part Z_V. (2) $\mathrm{Z}_V \not\perp \mathrm{A}$. The variant part Z_V is correlated with the architecture A. (3) $\mathrm{A} \perp \mathrm{Z}_V \,|\, \mathrm{Z}_I$ and $\mathrm{A} = \psi(\mathrm{Z}_I)$. Z_I shields A from the influence of Z_V.

Assumption 3.3 defines what constitutes an invariant pattern for architecture prediction: i) **Condition 1** indicates that the data contains two types of patterns: an invariant pattern Z_I, which reliably predicts the architecture, and a variant pattern Z_V, which cannot stably predict the architecture; ii) **Condition 2** highlights that the variant pattern Z_V is not independent of the architecture A; iii) **Condition 3** states that, given the invariant pattern Z_I, the architecture A is independent of the variant pattern Z_V, and Z_I is sufficient for predicting A.

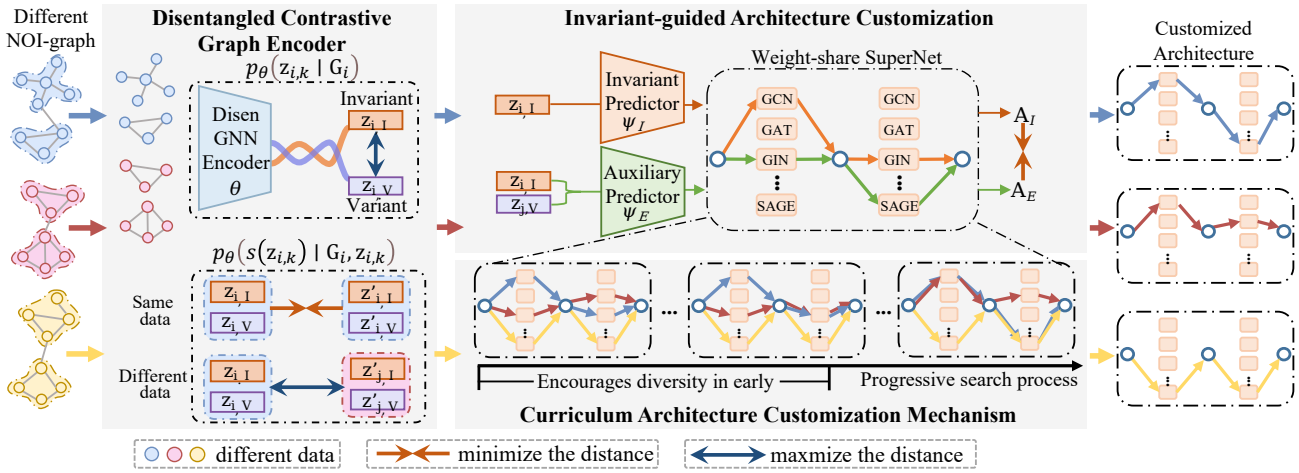

Figure 2. The framework of Automated Graph Foundation Model with Adaptive Architecture Customization(**AutoGFM**). The model consists of three modules: i) Disentangled contrastive graph encoder that discovers invariant and variant patterns from graph data , ii) invariant-guided architecture customization enabling customization of graph neural architectures based on the discovered invariant and variant patterns to adapt to data with diverse domains and tasks, and iii) Curriculum architecture customization mechanism to mitigate the influence of any single data dominating the search process.

3.3. Overall Objective

To satisfy the constraints outlined in Assumption 3.3, we formulate a learning objective that adheres to the specified conditions. Specifically, we minimize the mutual information between Z_I and Z_V to ensure that the two parts remain disjoint. Simultaneously, we maximize the mutual information between Z_I and A , ensuring that Z_I is sufficient for predicting the architecture A. Furthermore, we minimize the mutual information between A and Z_V, conditioned on Z_I, to guarantee that Z_I shields A from the influence of Z_V. The resulting overall learning objective is defined as follows:

$$\max_{\theta,\psi} \; I(\mathrm{Z}_I, \mathrm{A}) - \lambda I(\mathrm{Z}_I, \mathrm{Z}_V) - \beta I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I), \qquad (4)$$

where I denotes the mutual information function, θ represents the encoder that extracts Z_I and Z_V from G, ψ represents the predictor that maps Z_I to A, and λ and β are hyperparameters controlling the trade-off.

4. The Proposed Method: AutoGFM

In this section, we introduce an Automated Graph Foundation Model with Adaptive Architecture Customization (**AutoGFM**) to search for graph neural architectures for each graph data with diverse tasks and domains individually. We first introduce two modules: a disentangled contrastive graph encoder and invariant-guided architecture customization. Besides, we introduce our optimization objective with a curriculum architecture customization mechanism. The overall framework of **AutoGFM** is illustrated in Figure 2.

4.1. Disentangled Contrastive Graph Encoder

In this section, we focus on learning disentangled representations to capture two distinct aspects of graph data. Specifically, we aim to learn two architecture-aware disentangled representations, Z_I and Z_V, we temporarily treat them jointly and denote the two-channel representations as Z_k $(k = 1, 2)$ in this section. The main insight of our proposed method is intuitively based on the following observations: (1) Data from the same sources (i.e., the same domain and task) require similar architectures, so they share a similar Z_I that reflects the architectural requirements. Conversely, graphs from different data sources will have distinct Z_I. (2) To satisfy the Assumption 3.3, the mutual information between Z_I and Z_V should be minimized.

Disentangled NOI-graph Encoder Initially, we adopt GNNs with individual parameters to learn two-channel graph representations of NOI-graphs.

$$\mathrm{H}_k^{(l)} = \mathrm{GNN}_k\big(\mathrm{H}_k^{(l-1)}, \mathbf{A}\big), \quad k = 1, 2, \qquad (5)$$

where $\mathrm{H}_k^{(l)}$ is the k-th channel of the node representation at the l-th layer, $\mathbf{A}$ is the adjacency matrix of the graph. We employ two distinct Readout functions (i.e., pooling functions) and MLPs to derive a NOI-graph-level representation for each channel:

$$\mathrm{z}_k = \mathrm{MLP}_k\big(\mathrm{h}_k\big), \qquad (6)$$

$$\mathrm{h}_k = \mathrm{Readout}_k\big(\mathrm{H}_k^{(L)}\big), \quad k = 1, 2. \qquad (7)$$

NOI-graph Disentangled Contrastive Learning Inspired by self-supervised contrastive learning that captures

discriminative features by pulling similar samples together and pushing dissimilar samples apart in latent space (Jaiswal et al., 2020; Le-Khac et al., 2020; You et al., 2020), we propose an NOI-graph-level contrastive learning method to encourage disentangled representations to reflect the architectural requirements of the graph data. Initially, we encourage the representations Z_I and Z_V to be disentangled:

$$p_\theta\left(\mathrm{z}_{i,k} \mid \mathrm{G}_i\right)=\frac{\exp \phi\left(\mathrm{z}_{i,k}, \mathrm{p}_k\right)}{\sum_{j=1}^{2} \exp \phi\left(\mathrm{z}_{i,k}, \mathrm{p}_j\right)}, \tag{8}$$

where ϕ is a similarity function, G_i is a NOI-graph from i-th graph, $\mathrm{z}_{i,k}$ is the k-th chunk of the representation of a NOI-graph from i-th graph, and p_k is the propotype of the k-th chunk of the representation. Then, we propose a NOI-graph-level instance discriminative task to encourage the representations Z_I to capture different architectural requirements of different data. The task is defined as:

$$p_\theta\left(s(\mathrm{z}_{i,k}) \mid \mathrm{G}_i, \mathrm{z}_{i,k}\right)=\frac{\exp \phi\left(\mathrm{z}_{i,k}, \mathrm{z}'_{i,k}\right)}{\sum_{j=1}^{N} \exp \phi\left(\mathrm{z}_{i,k}, \mathrm{z}'_{j,k}\right)}, \tag{9}$$

where $s(\mathrm{z}_{i,k})$ represents a unique surrogate label assigned to $\mathrm{z}_{i,k}$, and $\mathrm{z}'_{i,k}$ is sampled from the same graph data G_i as $\mathrm{z}_{i,k}$. Then we learn the model parameters θ by calculating the loss function as:

$$\mathcal{L}_{dis}=\sum_i -\log \mathbb{E}_{p_\theta(\mathrm{z}_{i,k}|\mathrm{G}_i)} p_\theta\left(s(\mathrm{z}_{i,k}) \mid \mathrm{G}_i, \mathrm{z}_{i,k}\right). \tag{10}$$

In this way, we encourage the representations Z_I to capture the architectural requirements of the graph data, while ensuring that the representations Z_I and Z_V are disentangled.

4.2. Invariant-guided Architecture Customization

To customize graph neural architectures based on the discovered patterns and enable adaptation to data from diverse domains and tasks, we propose an invariant-guided architecture customization approach. Specifically, we first establish a weight-sharing super-network with a set of prototypes, then utilize an invariant predictor, ψ_I, and an auxiliary predictor, ψ_E, to guide the customization process, ensuring the minimization of $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ in Assumption 3.3.

Weight-sharing Super-network To facilitate differentiable optimization, we employ continuous parameterization and a weight-sharing mechanism (Liu et al., 2018) to implement the mixed operations. The super-network layer with $|\mathcal{O}|$ mixed operations is defined as:

$$\mathrm{H}^{(l)} \leftarrow \sum_{i=1}^{|\mathcal{O}|} \alpha_{l,i} \mathrm{GNN}_i^{(l-1)}(\mathrm{H}^{(l-1)}, \mathbf{A}), \tag{11}$$

where $\mathbf{A}$ is the adjacency matrix of the graph, $\mathrm{H}^{(l)}$ represents the node representations at the l-th layer, $\mathrm{GNN}_i^{(l-1)}$ denotes the mixed GNN operations, $|\mathcal{O}|$ is the number of GNN operation choices, and $\alpha_{l,i}$ indicates the probability of selecting the i-th operation for the l-th layer. Different from previous super-networks (Liu et al., 2018) that use learnable parameters α, we employ a set of prototypes to guide the routing between data and the operations.

Architecture Predictor Given a graph representation $\mathrm{z} \in \mathrm{Z}$, we design an invariant mapping predictor, ψ_I: $\mathrm{z} \rightarrow \{\alpha_{l,i}\}$. The probability $\alpha_{l,i}$ of selecting the i-th operation for the l-th layer is calculated as follows:

$$\alpha_{l,i}=\frac{\exp(\hat{\alpha}_{l,i})}{\sum_{j=1}^{|\mathcal{O}|} \exp(\hat{\alpha}_{l,j})}, \quad \hat{\alpha}_{l,i}=\mathrm{z} \cdot \frac{\mathrm{p}_{l,i}}{\|\mathrm{p}_{l,i}\|_2}, \tag{12}$$

where $\mathrm{p}_{l,i}$ is a learnable prototype of the i-th operation for the l-th layer, and z is the graph representation. Following (Qin et al., 2022a), we adopt the l_2-normalization on p to ensure numerical stability and fair competition among different operations. We utilize the learnable prototypes p as the parameters of the predictor ψ_I to map architecture, *i.e.*, if the graph representation z is similar to the prototype $\mathrm{p}_{l,i}$, the operation i will be selected for the l-th layer.

Invariant-guided Customization The objective of minimizing the conditional mutual information $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ in Equation (4) is not tractable, as the mutual information of high-dimensional vectors is difficult to estimate. Therefore, we utilize an equivalent transformation in Proposition 4.1 to achieve it, with a detailed proof provided in Appendix A.2.

Proposition 4.1. *if* $P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I)$*, the conditional mutual information* $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ *achieves its minimum value of* 0*.*

Proposition 4.1 indicates that we can minimize the conditional mutual information $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ by enforcing $P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I)$. To this end, we utilize an auxiliary predictor ψ_E with an invariant predictor ψ_I to guide the customization process. Specifically, we first predict the architecture A_I based on the invariant patterns Z_I using the invariant predictor ψ_I. Then, we predict the architecture A_E based on the patterns fused from Z_I and Z_V using the auxiliary predictor ψ_E:

$$\mathrm{A}_I=\psi_I(\mathrm{Z}_V), \quad \mathrm{A}_E=\psi_E(\mathrm{Z}_I, \mathrm{Z}_V). \tag{13}$$

We guide $P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I)$ by minimizing the difference between the two predictions $\mathrm{A}_I, \mathrm{A}_E$. Besides, to boost the architecture predictor to fit more data with different variant patterns, we fuse Z_I with Z_V from other

data to predict A_E. We define the loss as:

$$\mathcal{L}_{\text{inv}} = \sum_{i}^{\|\mathcal{D}\|}\sum_{j}^{\|\mathcal{D}\|}\|\mathrm{A}_{I,i} - \mathrm{A}_{E,(i,j)}\|, \tag{14}$$

$$s.t. \quad \mathrm{A}_{I,i} = \psi_I(\mathrm{z}_{I,j}), \mathrm{A}_{E,i,j} = \psi_E(\mathrm{z}_{I,i}, \mathrm{z}_{V,j}), \tag{15}$$

where $\mathrm{A}_{I,i}$ is the predicted architecture for the i-th graph based on the invariant patterns z_I, $\mathrm{A}_{E,i,j}$ are the predicted architectures based on the patterns fused from $\mathrm{z}_{I,i}$ and $\mathrm{z}_{V,j}$.

4.3. Optimization with Curriculum Customization Mechanism

We calculate the task loss of GFM using only the architecture predicted by the invariant predictor ψ_I:

$$\mathcal{L}_{task} = \ell(F_{\psi(\mathrm{Z}_I)}(\mathrm{G}), \mathrm{y}). \tag{16}$$

$\mathcal{L}_{task}$ aim to maximize $I(\mathrm{Z}_I, \mathrm{A})$ in Equation (4). Notably, the computation method of $\mathcal{L}_{task}$ depends on the GFM for which we aim to search for architectures. *e.g.*, GFT (Wang et al., 2024b) utilizes Computation Tree Reconstruction to calculate loss during the pretraining stage.

GFMs need to be simultaneously optimized using multiple datasets with diverse domains and tasks. However, different data have different influences on architectures (Zhou et al., 2022d), mainly on the learnable weights of operations in our study, *e.g.*, some datasets are easier to fit with certain operations but more challenging with others. As a result, operations that fit well in the early stages of training are more likely to be selected, causing other operations to be overlooked.

To mitigate the dominance of data in the search process, we design a curriculum architecture customization constraint that encourages diversity in the customized architectures during the early stages of training. We first calculate the average α for each operation in the l-th layer as:

$$\boldsymbol{\alpha}_l = \frac{\left[\sum_{i=1}^{|\mathcal{D}|}\alpha_{l,1}(\mathrm{z}_i), \sum_{i=1}^{|\mathcal{D}|}\alpha_{l,2}(\mathrm{z}_i), \ldots, \sum_{i=1}^{|\mathcal{D}|}\alpha_{l,J}(\mathrm{z}_i)\right]}{|\mathcal{D}|}, \tag{17}$$

where $\boldsymbol{\alpha}_l$ is the average α for each operation in the l-th layer, $\alpha_{l,j}(\mathrm{z}_i)$ is the probability of selecting the j-th operation for the l-th layer predicted by the invariant predictor ψ_I for the i-th graph, and J is the number of operations. We define the curriculum architecture customization loss as follows:

$$\mathcal{L}_{cur} = \gamma \sum_{l=1}^{L} \mathrm{CV}(\boldsymbol{\alpha}_l), \tag{18}$$

where $\mathrm{CV}(\boldsymbol{\alpha}_l)$ is the coefficient of variation of the average α for each operation in the l-th layer across different data and

Algorithm 1 Training pipeline for **AutoGFM**

Input: data $\mathcal{D} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$, hyperparameters λ, β.
for $t = 1, ..., T$ **do**
  Sample NOI-graphs G_i from $\mathcal{G}_i$.
  Extract Z_I and Z_V from NOI-graphs G_i.
  Calculate $\mathcal{L}_{dis}$ using Equation (10).
  Obtain architectures A_I and A_E predicted by ψ_I and ψ_E in Equation (15).
  Calculate $\mathcal{L}_{inv}$ using Equation (14).
  Calculate $\mathcal{L}_{cur}$ and $\mathcal{L}_{task}$ using Equation (18) and Equation (16), respectively.
  Update θ, ψ_I, ψ_E by minimizing Equation (19).
end for

γ is controlled by a pacing function: $\gamma = 1 - \frac{t}{t_e}$, where t is the current training step and t_e is the step to stop the curriculum customization mechanism. This constraint encourages diversity in early architecture customization, thereby mitigating the influence of any single data dominating the search process. The final training objective is:

$$\min_{\theta,\psi_I,\psi_E} \underbrace{\mathcal{L}_{task}}_{\max I(\mathrm{Z}_I,\mathrm{A})} + \lambda \underbrace{\mathcal{L}_{dis}}_{\min I(\mathrm{Z}_I,\mathrm{Z}_V)} + \beta \underbrace{\mathcal{L}_{inv}}_{\min I(\mathrm{A},\mathrm{Z}_V \mid \mathrm{Z}_I)} + \mathcal{L}_{cur}, \tag{19}$$

where $\mathcal{L}_{task}$ aims to exploit invariant patterns to customize architectures, $\mathcal{L}_{dis}$ encourages the disentanglement of the invariant and variant patterns, $\mathcal{L}_{inv}$ discovers the invariant patterns and variant patterns, and $\mathcal{L}_{cur}$ mitigates the influence of any single data dominating the search process. The overall algorithm is summarized in Algorithm 1.

During the inference stage, given an input graph, we first utilize the disentangled contrastive graph encoder to obtain its invariant pattern representation, denoted as Z_I. Then, Z_I is fed into the invariant predictor ψ_I to generate a customized architecture. This architecture is then used as the GNN component within the GFM to perform prediction.

5. Experiments

In this section, we conduct experiments on real-world datasets with diverse domains and tasks to show the effectiveness of the proposed **AutoGFM** for GFM.

5.1. Experimental Setup

Datasets We employ datasets with diverse domains and tasks. For node-level tasks, we utilize citation networks (Cora, Pubmed, and Arxiv) and the web link network (WikiCS). For edge-level tasks, we utilize Knowledge Graphs (WN18RR, FB15K237). For graph-level tasks, we utilize molecular datasets (HIV, PCBA, and ChEMBL). Following

Table 1. Accuracy (%) with std of different methods in pre-training and fine-tuning setting. The highest result is **bold**. The subscript v represents Vanilla GNNs. We do not explicitly report the performance of GFT separately, as GFT (Wang et al., 2024b) employs GraphSAGE as its GNN architecture, which overlaps with our baselines.

	Node Classification				Link Classification		Graph Classification		
Method	Cora	PubMed	Wiki-CS	Arxiv	WN18RR	FB15K237	HIV	PCBA	***Avg.***
Linear	58.03±2.33	68.66±2.24	70.36±0.58	66.50±0.14	78.50±0.59	87.39±0.07	66.37±1.11	72.30±0.34	71.01
GCN_v	75.65±1.37	75.61±2.10	75.28±1.34	71.40±0.08	73.79±0.39	82.22±0.28	64.84±4.78	71.32±0.49	73.76
GAT_v	76.24±1.62	74.86±1.87	76.78±0.78	70.87±0.24	80.16±0.27	88.93±0.15	65.54±6.93	70.12±0.89	75.44
GIN_v	73.59±2.10	69.51±6.87	49.77±4.72	65.05±0.50	74.02±0.55	83.21±0.53	66.86±3.48	72.69±0.22	69.34
DGI	72.10±0.34	73.13±0.64	75.32±0.95	69.15±0.20	75.75±0.59	81.34±0.15	59.62±1.21	63.31±0.89	71.22
BGRL	71.20±0.30	75.29±1.33	76.53±0.69	71.19±0.18	75.44±0.30	80.66±0.29	63.95±1.06	67.09±1.00	72.67
GraphMAE	73.10±0.40	74.32±0.33	77.61±0.39	70.90±0.31	78.99±0.48	85.30±0.16	61.04±0.55	63.30±0.78	73.07
GIANT	75.13±0.49	72.31±0.53	76.56±0.88	70.10±0.32	84.36±0.30	87.45±0.54	65.44±1.39	61.49±0.99	74.11
GCN	77.97±1.46	77.68±1.43	76.35±0.50	67.22±0.80	92.20±0.40	77.35±3.62	70.89±4.52	74.94±1.69	76.83
GAT	78.96±0.91	77.24±2.04	78.00±0.67	72.82±0.33	75.91±1.29	86.15±2.17	69.07±1.22	76.23±0.61	76.80
GraphSAGE	78.24±1.46	76.28±2.19	79.29±0.53	72.28±0.24	91.57±0.42	89.92±0.27	72.85±2.47	78.32±0.24	79.84
GIN	79.85±1.30	77.57±2.04	78.61±0.55	67.80±0.52	77.87±1.12	71.37±3.06	69.32±2.84	74.27±1.70	74.58
GraphConv	80.12±1.27	76.52±1.39	78.85±0.66	65.75±0.60	91.52±0.33	80.72±2.05	71.81±2.59	76.06±0.46	77.67
Darts	76.97±1.34	77.77±1.59	73.60±2.20	72.10±1.64	78.02±1.52	81.64±3.00	68.85±3.25	75.16±2.21	75.51
Graphnas	76.34±2.25	77.49±2.02	70.98±2.22	68.64±2.04	82.63±0.44	80.72±3.65	67.70±4.50	73.49±2.82	74.75
GASSO	78.24±1.50	77.82±1.68	71.90±2.00	70.85±1.90	81.96±0.58	80.66±2.21	68.37±6.28	76.76±1.64	75.82
Graces	78.30±1.92	76.98±2.57	70.05±3.06	70.23±1.22	84.05±0.49	83.91±3.99	70.93±2.24	75.89±1.54	76.29
Ours	**80.32±1.12**	**78.28±1.40**	**79.45±0.69**	**73.39±1.56**	**93.17±0.88**	**90.27±1.64**	**73.17±2.21**	**78.83±1.54**	**80.86**

(Liu et al., 2023a), we use the textual encoder to unify the node features from different domains.

Baselines We compare our proposed **AutoGFM** with the five categories of baselines: (1) **Vanilla GNNs**: GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2017), GIN (Xu et al., 2018); (2) **Self-supervised methods**: BGRL (Thakoor et al., 2021), GraphMAE (Hou et al., 2022), GIANT (Chien et al., 2022). (3) **GFMs**: OFA (Liu et al., 2023a) and GFT (Wang et al., 2024b). (4) **Manually designed GNNs**: GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2017), GIN (Xu et al., 2018), GraphSAGE (Hamilton et al., 2017), and GraphConv (Morris et al., 2019). (5) **GNAS methods**: DARTS (Liu et al., 2018), GraphNAS (Gao et al., 2021), GASSO (Qin et al., 2021b), Graces (Qin et al., 2022a).

For manually designed GNNs, GNAS baselines, and **AutoGFM** , we utilize GFT (Wang et al., 2024b) as the base model to ensure a fair comparison. Additionally, we adopt the same search space (operations in manually designed GNNs baselines and super-network layers is 2) for both GNAS baselines and **AutoGFM**. We replicate each experiment ten times and report the average results. Further details about experimental setups are provided in Appendix D.

5.2. Main Results

Pre-training and Fine-tuning From the results in Table 1, we observe the following: (1) None of manually designed GNN performs well across all datasets, indicating that fixed architectures struggle to generalize across diverse domains and tasks. (2) Existing GNAS methods fail to discover better architectures for each dataset, highlighting their limitations in adapting to diverse domains and tasks. (3) **AutoGFM** outperforms all baselines across datasets, demonstrating its effectiveness in customizing architectures for different domains and tasks.

Few-shot Learning Few-shot learning is a challenging task that requires models to generalize well with limited labeled samples. We randomly sample a few labeled samples per way from the training set for fine-tuning. From the results in Table 2, despite the limited labeled samples, **AutoGFM** achieves the best performance across all datasets, demonstrating the fast adaptability of its architectures. We provide more experimental results in Appendix B.3.

5.3. Ablation Study

To verify the effectiveness of the key modules in our method, we compare different ablated versions on five datasets: i) w/o D removes and replaces the disentangled contrastive graph encoder with standard GNNs; ii) w/o I removes the invariant-guided architecture customization module by removing the $\mathcal{L}_{inv}$; iii) w/o C removes the curriculum architecture customization mechanism. The results are shown in Figure 3. We observe that the full model achieves the best performance across all datasets, demonstrating the ef-

Table 2. Accuracy (%) with std of different methods in Few-shot learning. The highest result is **bold**.

	Cora-7 way			WN18RR-10 way			CHEMHIV-2 way			
Method	1-shot	3-shot	5-shot	1-shot	3-shot	5-shot	1-shot	3-shot	5-shot	10-shot
OFA	30.38±2.39	36.03±2.11	32.10±1.79	25.82±1.07	30.56±1.02	32.64±1.56	57.17±1.82	59.30±3.04	57.56±3.66	54.36±4.90
GFT	41.40±8.04	43.31±8.11	43.55±7.43	35.33±4.20	35.50±5.02	35.50±4.59	**59.94±7.09**	58.44±7.28	58.78±6.92	58.67±7.54
GCN	43.07±7.37	42.38±7.42	42.57±7.50	29.85±4.14	29.78±3.64	30.40±3.02	59.58±6.15	59.53±8.21	59.28±6.79	59.64±7.82
GAT	46.12±7.10	47.31±7.78	47.71±8.02	34.50±2.98	34.37±3.43	34.70±3.15	56.39±9.80	59.17±9.43	59.33±7.57	59.22±7.20
GraphSAGE	40.50±6.11	42.07±6.12	42.40±6.12	38.03±2.03	38.17±2.34	38.30±2.16	58.33±4.28	58.64±6.22	59.28±6.96	58.17±9.16
GIN	45.29±6.26	47.02±7.32	47.24±7.33	36.62±4.17	36.92±4.03	37.47±3.10	58.72±4.45	57.97±3.73	59.06±4.17	57.22±4.89
GraphConv	38.67±8.50	40.93±8.80	41.60±9.20	38.93±3.77	39.28±2.27	39.62±3.44	53.00±7.75	55.50±8.45	54.67±8.17	53.06±7.47
DARTS	43.29±7.65	42.10±7.45	42.81±7.92	37.22±2.68	38.57±3.22	38.65±3.55	58.31±6.73	58.68±6.72	58.07±6.17	59.05±8.83
GraphNAS	38.64±8.31	40.60±9.23	41.62±9.20	36.79±3.17	37.03±3.32	36.98±5.23	57.62±5.70	58.52±7.05	58.91±6.14	59.32±7.75
GASSO	40.31±6.10	42.07±5.85	42.95±5.75	37.13±3.52	37.42±1.91	37.37±3.31	59.46±5.35	59.38±7.10	59.26±6.45	59.72±5.42
GRACES	45.43±6.73	46.31±7.42	47.57±7.22	38.76±2.63	38.24±3.60	39.13±2.29	59.39±4.01	58.62±9.30	59.45±5.79	59.71±7.18
Ours	**46.29±7.24**	**47.33±7.80**	**47.76±8.06**	**39.34±3.03**	**39.55±2.46**	**40.02±2.26**	59.73±4.46	**59.68±7.74**	**60.08±4.22**	**59.92±4.08**

fectiveness of each module. We further observe the following: i) The disentangled contrastive graph encoder module is designed to extract discriminative invariant and variant patterns from the data by pulling similar samples closer and pushing dissimilar samples apart in the latent space. Removing this module impairs the extraction of invariant patterns and reduces the distinguishability between patterns extracted from different datasets, ultimately harming the effectiveness of architecture prediction; ii) The invariant-guided architecture customization module serves to shield architecture A from the influence of variant patterns Z_V given the invariant pattern Z_I. The substantial performance decrease observed upon removing this module highlights the importance of effectively isolating architecture predictions from Z_V influences, reinforcing the critical role of this module in ensuring the invariance conditions of captured patterns; iii) This curriculum architecture customization mechanism aims to reduce data dominance in the architecture search process. Removing this module causes certain operations, which perform well on specific datasets during early training stages, to dominate the search process. Consequently, other datasets may neglect potentially beneficial operations.

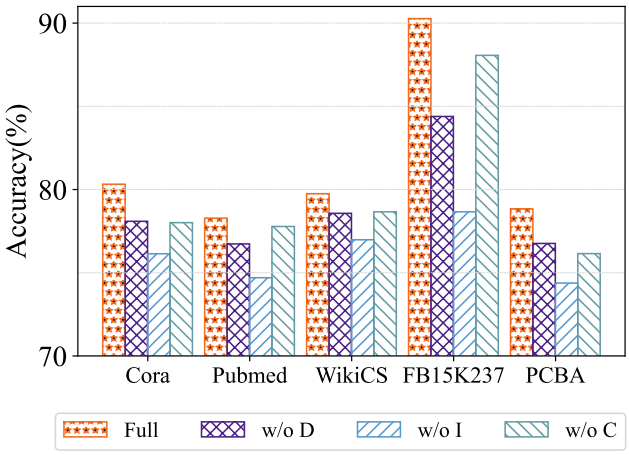

Figure 3. Comparisons of different ablated versions of **AutoGFM** on real-world datasets. "Full" denotes the full version of the method.

5.4. Time Complexity Analysis

Let $|V|$ and $|E|$ denote the number of nodes and edges, and d as the dimensionality. We use d_e and d_a to denote the dimensionality of the disentangled graph encoder and the customized super-network. The time complexity of the GNN layers in both the graph encoder and the super-network is $O(|E|d + |V|d^2)$. Therefore, the time complexity of our disentangled graph encoder is $O(|E|d_e + |V|d_e^2)$. The time complexity of the architecture customization with prototypes is $O(|\mathcal{O}|^2 d_e)$. The time complexity of the customized super-network is $O(|\mathcal{O}|(|E|d_a + |V|d_a^2))$. Thus, the overall computational complexity of our method is given by: $O(|E|d_e + |V|d_e^2 + |\mathcal{O}|^2 d_e + |\mathcal{O}|(|E|d_a + |V|d_a^2))$.

6. Related Work

In this section, we review the related work on GNN-based Graph Foundation Models, graph neural architecture search, graph invariant presentation learning, and graph self-supervised learning. We provide more related work in Appendix C.

6.1. GNN-based Graph Foundation Models

GNN-based GFMs, which leverage LLMs as enhancers, are a promising direction for GFM (He & Hooi, 2024; Mao et al., 2024a; Zhao et al., 2024b; Xia & Huang, 2024; Huang

et al., 2024c). Specifically, it involves using LLMs to transform the textual features of graphs into unified representations and unify graph-related tasks through subgraph classification for GNNs (Sun et al., 2023; Zhao et al., 2024c; Fan et al., 2024; Ren et al., 2024; Pan et al., 2024b; Mao et al., 2024b). This enables the transfer of shared knowledge across various domains. Two key challenges in developing GNN-based Graph Foundation Models (GFMs) are the unification of diverse tasks and domain spaces (Yu et al., 2024; Li et al., 2024b; Zhao et al., 2024a; Xia et al., 2024; Zhu et al., 2024b; Guo et al., 2023). For instance, OFA (Liu et al., 2023a) employs an LLM to unify input features from diverse datasets, converting multiple graph classification tasks into a unified binary classification format. GFT (Wang et al., 2024b) extends OFA by incorporating computation trees to discover transferable patterns across graphs. Nevertheless, these models face challenges in their reliance on manually designed architectures, which can constrain their performance on diverse domains and tasks. More related work about GFMs is included in Appendix C.

6.2. Graph Neural Architecture Search

Neural architecture search (NAS) has gained growing attention for its capability to automate the design of neural architectures tailored to specific tasks (Pham et al., 2018; Qin et al., 2021a; Wang et al., 2025). In particular, graph neural architecture search (GNAS) methods address the distinct challenge of modeling the intricate relationships between architectures and complex graph structures (Gao et al., 2021; Qin et al., 2022b; Guan et al., 2022; Zhang et al., 2023e; Xie et al., 2023). These methods can be broadly classified into three categories: reinforcement-learning-based approaches (Zhou et al., 2022b; Gao et al., 2022; 2023); evolutionary-based strategies (Nunes & Pappa, 2020; Li & King, 2020; Shi et al., 2022; Zhang et al., 2022a;b); and differentiable methods (Ding et al., 2021; Zheng et al., 2023; Huan et al., 2021; Zhang et al., 2023b;d; Qin et al., 2023; Yao et al., 2024; Ge et al., 2025), which enable continuous optimization of architectures within a differentiable search space. However, existing GNAS methods are limited in their ability to search for architectures for GNN-based GFMs.

6.3. Graph Invariant Representation Learning

Graph invariant presentation learning has emerged as a powerful approach for graph representation learning, focusing on capturing the stable relationships between graph data and tasks. Recent works have explored various applications of graph invariant learning in out-of-distribution generalization (Ma et al., 2019; Wu et al., 2022b; Li et al., 2022b;c; Zhang et al., 2022c; 2023c; 2024b; Li et al., 2024a). For instance, DIR (Wu et al., 2022b) discovers causal rationales that remain invariant across different distributions while filtering out spurious patterns that are unstable. DIDA (Zhang et al., 2022c) leverages invariant structures and features with stable predictive performance across distribution shifts. However, these methods focus on capturing stable relationships for accurate label prediction. We apply this concept to architecture search, aiming to define invariant patterns that support stable architecture prediction, and design our method based on this concept.

6.4. Graph Self-supervised Learning

Graph self-supervised learning (SSL) has attracted significant attention in recent years, with numerous methods proposed to learn effective representations from graph data without relying on labeled information. These methods can be broadly classified into two categories: contrastive learning and generative learning. Contrastive learning approaches (You et al., 2020; Hassani & Khasahmadi, 2020; Li et al., 2021; Zhang et al., 2024a; Li et al., 2022d) aim to maximize the agreement between positive pairs of graph samples while minimizing it between negative pairs. In contrast, generative learning approaches (Tan et al., 2023; Xia et al., 2023; Hou et al., 2023) learn representations by reconstructing graph structures or attributes from partially observed data. These SSL techniques have demonstrated strong performance across a variety of graph-related tasks, including node classification, link prediction, and graph classification. In our work, we adopt a contrastive learning strategy to extract distinct patterns from diverse datasets, enabling the model to capture richer information and better identify architecture-specific requirements.

7. Conclusion

Existing graph neural architecture search methods fail to search for architectures for GNN-based GFMs. In this paper, we analyze the problem of *architecture inconsistency*, demonstrate that existing GNAS methods cannot effectively search for architectures for GNN-based GFMs, and tackle it by discovering an invariant graph-architecture relationship. We propose a novel Automated Graph Foundation Model with Adaptive Architecture Customization (**AutoGFM**) to search for graph neural architectures for each graph data with diverse tasks and domains individually. We introduce a disentangled contrastive graph encoder to discover invariant and variant patterns from graph data and an invariant-guided architecture customization module to customize graph neural architectures based on the discovered patterns. We also propose a curriculum architecture customization mechanism to mitigate the phenomenon of some particular data dominating the search process. Experimental results demonstrate that **AutoGFM** outperforms existing methods. One limitation of our work is that we mainly focus on graph neural architecture search on the GNN-based GFMs, and we leave the exploration of other types of GFMs for future work.

Acknowledgements

This work is supported by National Natural Science Foundation of China No.62222209, Beijing National Research Center for Information Science and Technology under Grant No.BNR2023TD03006.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Chen, H., Wang, X., Lan, X., Chen, H., Duan, X., Jia, J., and Zhu, W. Curriculum-listener: Consistency-and complementarity-aware audio-enhanced temporal sentence grounding. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 3117–3128, 2023.

Chen, R., Zhao, T., Jaiswal, A., Shah, N., and Wang, Z. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024.

Chen, Y., Wang, X., Fan, M., Huang, J., Yang, S., and Zhu, W. Curriculum meta-learning for next poi recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2692–2702, 2021.

Chien, E., Chang, W.-C., Hsieh, C.-J., Yu, H.-F., Zhang, J., Milenkovic, O., and Dhillon, I. S. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *International Conference on Learning Representations*, 2022.

Ding, Y., Yao, Q., Zhao, H., and Zhang, T. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 279–288, 2021.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Fan, W., Wang, S., Huang, J., Chen, Z., Song, Y., Tang, W., Mao, H., Liu, H., Liu, X., Yin, D., et al. Graph machine learning in the era of large language models (llms). *arXiv preprint arXiv:2404.14928*, 2024.

Fang, Y., Fan, D., Zha, D., and Tan, Q. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 747–758, 2024.

Fatemi, B., Halcrow, J., and Perozzi, B. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.

Gao, Y., Yang, H., Zhang, P., Zhou, C., and Hu, Y. Graph neural architecture search. In *International joint conference on artificial intelligence*. International Joint Conference on Artificial Intelligence, 2021.

Gao, Y., Zhang, P., Yang, H., Zhou, C., Hu, Y., Tian, Z., Li, Z., and Zhou, J. Graphnas++: Distributed architecture search for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6973–6987, 2022.

Gao, Y., Zhang, P., Zhou, C., Yang, H., Li, Z., Hu, Y., and Philip, S. Y. Hgnas++: efficient architecture search for heterogeneous graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9448–9461, 2023.

Ge, C., Wang, X., Zhang, Z., Qin, Y., Chen, H., Wu, H., Zhang, Y., Yang, Y., and Zhu, W. Behavior importance-aware graph neural architecture search for cross-domain recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 11708–11716, 2025.

Gong, C., Yang, J., and Tao, D. Multi-modal curriculum learning over graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(4):1–25, 2019.

Gong, T., Zhao, Q., Meng, D., and Xu, Z. Why curriculum learning & self-paced learning work in big/noisy data: A theoretical perspective. *Big Data & Information Analytics*, 1(1):111–127, 2015.

Guan, C., Wang, X., Chen, H., Zhang, Z., and Zhu, W. Large-scale graph neural architecture search. In *International Conference on Machine Learning*, pp. 7968–7981. PMLR, 2022.

Guo, Y., Yang, C., Chen, Y., Liu, J., Shi, C., and Du, J. A data-centric framework to endow graph neural networks with out-of-distribution detection ability. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 638–648, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pp. 4116–4126. PMLR, 2020.

He, Y. and Hooi, B. Unigraph: Learning a cross-domain graph foundation model from natural language. *arXiv preprint arXiv:2402.13630*, 2024.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

Hou, Z., He, Y., Cen, Y., Liu, X., Dong, Y., Kharlamov, E., and Tang, J. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*, pp. 737–746, 2023.

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. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Huan, Z., Quanming, Y., and Weiwei, T. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 552–563. IEEE, 2021.

Huang, B., He, F., Wang, Q., Chen, H., Li, G., Feng, Z., Wang, X., and Zhu, W. Neighbor does matter: Curriculum global positive-negative sampling for vision-language pre-training. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 8005–8014, 2024a.

Huang, C., Ren, X., Tang, J., Yin, D., and Chawla, N. Large language models for graphs: Progresses and directions. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 1284–1287, 2024b.

Huang, J., Zhang, X., Mei, Q., and Ma, J. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595*, 2023.

Huang, Q., Ren, H., Chen, P., Kržmanc, G., Zeng, D., Liang, P. S., and Leskovec, J. Prodigy: Enabling in-context learning over graphs. *Advances in Neural Information Processing Systems*, 36, 2024c.

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

Jin, B., Zhang, W., Zhang, Y., Meng, Y., Zhang, X., Zhu, Q., and Han, J. Patton: Language model pretraining on text-rich networks. *arXiv preprint arXiv:2305.12268*, 2023.

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

Kong, L., Feng, J., Liu, H., Huang, C., Huang, J., Chen, Y., and Zhang, M. Gofa: A generative one-for-all model for joint graph language modeling. *arXiv preprint arXiv:2407.09709*, 2024.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934, 2020.

Li, B., Shen, Y., Wang, Y., Zhu, W., Li, D., Keutzer, K., and Zhao, H. Invariant information bottleneck for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7399–7407, 2022a.

Li, H., Wang, X., Zhang, Z., Yuan, Z., Li, H., and Zhu, W. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34:21872–21884, 2021.

Li, H., Wang, X., Zhang, Z., and Zhu, W. Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 35 (7):7328–7340, 2022b.

Li, H., Zhang, Z., Wang, X., and Zhu, W. Learning invariant graph representations for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 35:11828–11841, 2022c.

Li, H., Wang, X., and Zhu, W. Curriculum graph machine learning: A survey. *arXiv preprint arXiv:2302.02926*, 2023.

Li, H., Wang, X., Zhang, Z., Chen, H., Zhang, Z., and Zhu, W. Disentangled graph self-supervised learning for out-of-distribution generalization. In *Forty-first International Conference on Machine Learning*, 2024a.

Li, S., Wang, X., Zhang, A., Wu, Y., He, X., and Chua, T.-S. Let invariant rationale discovery inspire graph contrastive learning. In *International conference on machine learning*, pp. 13052–13065. PMLR, 2022d.

Li, Y. and 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. Springer, 2020.

Li, Y., Wang, P., Li, Z., Yu, J. X., and Li, J. Zerog: Investigating cross-dataset zero-shot transferability in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1725–1735, 2024b.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2023a.

Liu, J., Yang, C., Lu, Z., Chen, J., Li, Y., Zhang, M., Bai, T., Fang, Y., Sun, L., Yu, P. S., et al. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829*, 2023b.

Liu, Z., Yu, X., Fang, Y., and Zhang, X. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pp. 417–428, 2023c.

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.

Mao, H., Chen, Z., Tang, W., Zhao, J., Ma, Y., Zhao, T., Shah, N., Galkin, M., and Tang, J. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024a.

Mao, Q., Liu, Z., Liu, C., Li, Z., and Sun, J. Advancing graph representation learning with large language models: A comprehensive survey of techniques. *arXiv preprint arXiv:2402.05952*, 2024b.

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*, volume 33, pp. 4602–4609, 2019.

Nunes, M. and 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. Springer, 2020.

Pan, B., Zhang, Z., Zhang, Y., Hu, Y., and Zhao, L. Distilling large language models for text-attributed graph learning. *arXiv preprint arXiv:2402.12022*, 2024a.

Pan, S., Zheng, Y., and Liu, Y. Integrating graphs with large language models: Methods and prospects. *IEEE Intelligent Systems*, 39(1):64–68, 2024b.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pp. 4095–4104. PMLR, 2018.

Qin, Y., Wang, X., Cui, P., and Zhu, W. Gqnas: Graph q network for neural architecture search. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 1288–1293. IEEE, 2021a.

Qin, Y., Wang, X., Zhang, Z., and Zhu, W. Graph differentiable architecture search with structure learning. *Advances in neural information processing systems*, 34: 16860–16872, 2021b.

Qin, Y., Wang, X., Zhang, Z., Xie, P., and Zhu, W. Graph neural architecture search under distribution shifts. In *International Conference on Machine Learning*, pp. 18083–18095. PMLR, 2022a.

Qin, Y., Zhang, Z., Wang, X., Zhang, Z., and Zhu, W. Nas-bench-graph: Benchmarking graph neural architecture search. *Advances in neural information processing systems*, 35:54–69, 2022b.

Qin, Y., Wang, X., Zhang, Z., Chen, H., and Zhu, W. Multi-task graph neural architecture search with task-aware collaboration and curriculum. *Advances in neural information processing systems*, 36:24879–24891, 2023.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Ren, X., Tang, J., Yin, D., Chawla, N., and Huang, C. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6616–6626, 2024.

Shi, M., Tang, Y., Zhu, X., Huang, Y., Wilson, D., Zhuang, Y., and Liu, J. Genetic-gnn: Evolutionary architecture search for graph neural networks. *Knowledge-based systems*, 247:108752, 2022.

Sun, X., Cheng, H., Li, J., Liu, B., and Guan, J. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2120–2131, 2023.

Tan, Q., Liu, N., Huang, X., Choi, S.-H., Li, L., Chen, R., and Hu, X. S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pp. 787–795, 2023.

Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., and Huang, C. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024a.

Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., and Huang, C. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024b.

Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

Tian, Y., Song, H., Wang, Z., Wang, H., Hu, Z., Wang, F., Chawla, N. V., and Xu, P. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19080–19088, 2024.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al. Graph attention networks. *stat*, 1050 (20):10–48550, 2017.

Wang, H., Zhou, K., Zhao, X., Wang, J., and Wen, J.-R. Curriculum pre-training heterogeneous subgraph transformer for top-n recommendation. *ACM Transactions on Information Systems*, 41(1):1–28, 2023.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36, 2024a.

Wang, X., Li, H., Zhang, Z., Chen, H., and Zhu, W. Modular machine learning: An indispensable path towards new-generation large language models. *arXiv preprint arXiv:2504.20020*, 2025.

Wang, Z., Zhang, Z., Chawla, N. V., Zhang, C., and Ye, Y. Gft: Graph foundation model with transferable tree vocabulary. *arXiv preprint arXiv:2411.06070*, 2024b.

Wei, X., Gong, X., Zhan, Y., Du, B., Luo, Y., and Hu, W. Clnode: Curriculum learning for node classification. In *WSDM*, pp. 670–678, 2023.

Wu, Q., Zhang, H., Yan, J., and Wipf, D. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022a.

Wu, Y., Yao, J., Xia, X., Yu, J., Wang, R., Han, B., and Liu, T. Mitigating label noise on graph via topological sample selection. *arXiv preprint arXiv:2403.01942*, 2024.

Wu, Y.-X., Wang, X., Zhang, A., He, X., and Chua, T.-S. Discovering invariant rationales for graph neural networks. *arXiv preprint arXiv:2201.12872*, 2022b.

Xia, L. and Huang, C. Anygraph: Graph foundation model in the wild. 2024.

Xia, L., Huang, C., Huang, C., Lin, K., Yu, T., and Kao, B. Automated self-supervised learning for recommendation. In *Proceedings of the ACM web conference 2023*, pp. 992–1002, 2023.

Xia, L., Kao, B., and Huang, C. Opengraph: Towards open graph foundation models. *arXiv preprint arXiv:2403.01121*, 2024.

Xie, B., Chang, H., Zhang, Z., Wang, X., Wang, D., Zhang, Z., Ying, R., and Zhu, W. Adversarially robust neural architecture search for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8143–8152, 2023.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Xu, Y., Liu, X., Duan, K., Fang, Y., Chuang, Y.-N., Zha, D., and Tan, Q. Graphfm: A comprehensive benchmark for graph foundation model. *arXiv preprint arXiv:2406.08310*, 2024.

Yao, Y., Wang, X., Qin, Y., Zhang, Z., Zhu, W., and Mei, H. Data-augmented curriculum graph neural architecture search under distribution shifts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16433–16441, 2024.

Ye, R., Zhang, C., Wang, R., Xu, S., Zhang, Y., et al. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 4(5):7, 2023.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

Yu, X., Qu, M., Feng, X., and Qin, B. Graphagent: Exploiting large language models for interpretable learning on text-attributed graphs.

Yu, X., Zhou, C., Fang, Y., and Zhang, X. Multigprompt for multi-task pre-training and prompting on graphs. In *Proceedings of the ACM on Web Conference 2024*, pp. 515–526, 2024.

Zhang, S., Hu, Z., Subramonian, A., and Sun, Y. Motif-driven contrastive learning of graph representations. *IEEE Transactions on Knowledge and Data Engineering*, 36 (8):4063–4075, 2024a.

Zhang, W., Lin, Z., Shen, Y., Li, Y., Yang, Z., and Cui, B. Deep and flexible graph neural architecture search. In *International Conference on Machine Learning*, pp. 26362–26374. PMLR, 2022a.

Zhang, W., Shen, Y., Lin, Z., Li, Y., Li, X., Ouyang, W., Tao, Y., Yang, Z., and Cui, B. Pasca: A graph neural architecture search system under the scalable paradigm. In *Proceedings of the ACM Web Conference 2022*, pp. 1817–1828, 2022b.

Zhang, Z., Wang, X., Zhang, Z., Li, H., Qin, Z., and Zhu, W. Dynamic graph neural networks under spatio-temporal distribution shift. *Advances in neural information processing systems*, 35:6074–6089, 2022c.

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*, volume 36, pp. 9136–9144, 2022d.

Zhang, Z., Wang, J., and Zhao, L. Relational curriculum learning for graph neural networks, 2023a. URL https://openreview.net/forum?id=1bLT3dGNS0.

Zhang, Z., Wang, X., Guan, C., Zhang, Z., Li, H., and Zhu, W. Autogt: Automated graph transformer architecture search. In *The Eleventh International Conference on Learning Representations*, 2023b.

Zhang, Z., Wang, X., Zhang, Z., Qin, Z., Wen, W., Xue, H., Li, H., and Zhu, W. Spectral invariant learning for dynamic graphs under distribution shifts. *Advances in Neural Information Processing Systems*, 36:6619–6633, 2023c.

Zhang, Z., Wang, X., Zhang, Z., Shen, G., Shen, S., and Zhu, W. Unsupervised graph neural architecture search with disentangled self-supervision. *Advances in Neural Information Processing Systems*, 36:73175–73190, 2023d.

Zhang, Z., Zhang, Z., Wang, X., Qin, Y., Qin, Z., and Zhu, W. Dynamic heterogeneous graph attention neural architecture search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11307–11315, 2023e.

Zhang, Z., Wang, X., Chen, H., Li, H., and Zhu, W. Disentangled dynamic graph attention network for out-of-distribution sequential recommendation. *ACM Transactions on Information Systems*, 43(1):1–42, 2024b.

Zhao, H., Chen, A., Sun, X., Cheng, H., and Li, J. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4443–4454, 2024a.

Zhao, J., Zhuo, L., Shen, Y., Qu, M., Liu, K., Bronstein, M., Zhu, Z., and Tang, J. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

Zhao, J., Mostafa, H., Galkin, M., Bronstein, M., Zhu, Z., and Tang, J. Graphany: A foundation model for node classification on any graph. *arXiv preprint arXiv:2405.20445*, 2024b.

Zhao, Z., Li, Y., Zou, Y., Li, R., and Zhang, R. A survey on self-supervised pre-training of graph foundation models: A knowledge-based perspective. *arXiv preprint arXiv:2403.16137*, 2024c.

Zheng, X., Zhang, M., Chen, C., Zhang, Q., Zhou, C., and Pan, S. Auto-heg: Automated graph neural network on heterophilic graphs. In *Proceedings of the ACM Web Conference 2023*, pp. 611–620, 2023.

Zhou, D., Zheng, L., Fu, D., Han, J., and He, J. Mentorgnn: Deriving curriculum for pre-training gnns. In *CIKM*, pp. 2721–2731, 2022a.

Zhou, K., Huang, X., Song, Q., Chen, R., and Hu, X. Auto-gnn: Neural architecture search of graph neural networks. *Frontiers in big Data*, 5:1029307, 2022b.

Zhou, Y., Chen, H., Pan, Z., Yan, C., Lin, F., Wang, X., and Zhu, W. Curml: A curriculum machine learning library. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 7359–7363, 2022c.

Zhou, Y., Wang, X., Chen, H., Duan, X., Guan, C., and Zhu, W. Curriculum-nas: Curriculum weight-sharing neural architecture search. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 6792–6801, 2022d.

Zhou, Y., Wang, X., Chen, H., Duan, X., and Zhu, W. Intra- and inter-modal curriculum for multimodal learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 3724–3735, 2023.

Zhou, Y., Pan, Z., Wang, X., Chen, H., Li, H., Huang, Y., Xiong, Z., Xiong, F., Xu, P., Zhu, W., et al. Curbench: curriculum learning benchmark. In *Forty-first International Conference on Machine Learning*, 2024.

Zhu, Y., Wang, Y., Shi, H., and Tang, S. Efficient tuning and inference for large language models on textual graphs. *arXiv preprint arXiv:2401.15569*, 2024a.

Zhu, Y., Wang, Y., Shi, H., Zhang, Z., Jiao, D., and Tang, S. Graphcontrol: Adding conditional control to universal graph pre-trained models for graph domain transfer learning. In *Proceedings of the ACM on Web Conference 2024*, pp. 539–550, 2024b.

A. Proof

A.1. Proof of Proposition 3.2

Proposition 3.2: If there exist two datasets $\mathcal{D}_i, \mathcal{D}_j \in \mathcal{D}$, the optimal operation for $\mathcal{D}_i$ is different from $\mathcal{D}_j$, the operations will render optimization conflicts.

Proof. Assume the optimal operation for $\mathcal{D}_i$ is o_1 and the optimal operation for $\mathcal{D}_j$ is o_2. The overall architecture function is defined as: $F(G) = o_1 f_1(G) + o_2 f_2(G)$. The MSE loss function $\mathcal{L}_{all}(F)$ is given by:

$$\mathcal{L}_{all}(F) = \mathcal{L}_i(F) + \mathcal{L}_j(F) + \mathcal{L}_k(F), \quad k \neq i, j, \tag{20}$$

where $\mathcal{L}_i(F) = \sum_k \left(F(G_{i,k}) - y_{i,k}\right)^2$, $\mathcal{L}_j(F) = \sum_k \left(F(G_{j,k}) - y_{j,k}\right)^2$.

We examine the effects of changes in o_1 and o_2 on the two terms $\mathcal{L}_i(F)$ and $\mathcal{L}_j(F)$ by calculating the partial derivatives of the two terms with respect to o_1 and o_2: $\frac{\partial \mathcal{L}_i}{\partial o_1}$, $\frac{\partial \mathcal{L}_i}{\partial o_2}$, $\frac{\partial \mathcal{L}_j}{\partial o_1}$, and $\frac{\partial \mathcal{L}_j}{\partial o_2}$.

We first simplify $\mathcal{L}_i(F)$ as follows:

$$\begin{aligned}
\mathcal{L}_i(F) &= \sum_k \left(F(G_{i,k}) - y_{i,k}\right)^2 && (21)\\
&= \sum_k \left(o_1 f_1(G_{i,k}) + o_2 f_2(G_{i,k}) - y_{i,k}\right)^2 && (22)\\
&= \sum_k \left(o_1 f_1(G_{i,k}) + o_2 f_2(G_{i,k}) - f_1(G_{i,k})\right)^2 && (23)\\
&= \sum_k \left((1-o_2) f_1(G_{i,k}) + o_2 f_2(G_{i,k}) - f_1(G_{i,k})\right)^2 && (24)\\
&= o_2^2 \sum_k \left(f_2(G_{i,k}) - f_1(G_{i,k})\right)^2 && (25)\\
&= (1-o_1)^2 \sum_k \left(f_2(G_{i,k}) - f_1(G_{i,k})\right)^2. && (26)
\end{aligned}$$

Equation (22) expands $F(G_{i,k})$ as $F(G_{i,k}) = o_1 f_1(G_{i,k}) + o_2 f_2(G_{i,k})$. In Equation (23), $y_{i,k}$ is replaced with $f_1(G_{i,k})$ because the optimal operation for $\mathcal{D}_i$ is o_1, i.e., $y_{i,k} = f_1(G_{i,k})$. In Equation (24), o_1 is replaced with $1 - o_2$ due to the constraint $o_1 + o_2 = 1$. Equation (25) simplifies the equation.

Similarly, for $\mathcal{L}_j(F)$:

$$\begin{aligned}
\mathcal{L}_j(F) &= o_1^2 \sum_k \left(f_2(G_{j,k}) - f_1(G_{j,k})\right)^2 \\
&= (1-o_2)^2 \sum_k \left(f_2(G_{j,k}) - f_1(G_{j,k})\right)^2. && (27)
\end{aligned}$$

Then we examine the effects of changes in o_1 on the two terms $\mathcal{L}_i(F)$ and $\mathcal{L}_j(F)$ by calculating the partial derivatives of the two terms with respect to o_1: $\frac{\partial \mathcal{L}_i}{\partial o_1}$, $\frac{\partial \mathcal{L}_j}{\partial o_1}$.

$$\mathcal{L}_i = (1-o_1)^2 \sum_k \left(f_2(G_{i,k}) - f_1(G_{i,k})\right)^2. \tag{28}$$

$$\mathcal{L}_j = o_1^2 \sum_k \left(f_2(G_{j,k}) - f_1(G_{j,k})\right)^2. \tag{29}$$

We calculate $\frac{\partial \mathcal{L}_i}{\partial o_1}$, $\frac{\partial \mathcal{L}_j}{\partial o_1}$ as follows:

$$\frac{\partial \mathcal{L}_i}{\partial o_1} = (2o_1 - 2) \sum_k \left(f_2(G_{i,k}) - f_1(G_{i,k})\right)^2 < 0 \tag{30}$$

$$\frac{\partial \mathcal{L}_j}{\partial o_1} = 2o_1 \sum_k \left(f_2(G_{j,k}) - f_1(G_{j,k})\right)^2 > 0 \tag{31}$$

$o_1, o_2 \in (0, 1)$, and $f_1 \neq f_2$ so that $\sum_k \left(f_2(G_{i,k}) - f_1(G_{i,k})\right)^2$ and $\sum_k \left(f_2(G_{j,k}) - f_1(G_{j,k})\right)^2$ are both positive. Therefore, $\frac{\partial \mathcal{L}_i}{\partial o_1} < 0$ and $\frac{\partial \mathcal{L}_j}{\partial o_1} > 0$. To minimize the loss function $\mathcal{L}_i$, o_1 must be increased, whereas minimizing the loss function $\mathcal{L}_j$ requires decreasing o_1. Similar results about o_2 can be obtained for $\frac{\partial \mathcal{L}_i}{\partial o_2}$, $\frac{\partial \mathcal{L}_j}{\partial o_2}$. Therefore, the operation optimization objective for $\mathcal{D}_i$ is different from $\mathcal{D}_j$,*i.e.*, the operations will render optimization conflicts.

A.2. Proof of Proposition 4.1

Proposition 4.1: if $P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I)$, the conditional mutual information $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ achieves its minimum value of 0: $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I) = 0$

Proof. The conditional mutual information $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ is defined as:

$$I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I) = \mathbb{E}_{\mathrm{Z}_I}\left[\mathbb{E}_{\mathrm{A},\mathrm{Z}_V|\mathrm{Z}_I}\left[\log \frac{P(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)}{P(\mathrm{A} \mid \mathrm{Z}_I)P(\mathrm{Z}_V \mid \mathrm{Z}_I)}\right]\right]. \tag{32}$$

Based on $P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I)$, we can obtain $P(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I) = P(\mathrm{A} \mid \mathrm{Z}_I)P(\mathrm{Z}_V \mid \mathrm{Z}_I)$ as follows:

$$P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I), \tag{33}$$

$$P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V)P(\mathrm{Z}_V \mid \mathrm{Z}_I) = P(\mathrm{A} \mid \mathrm{Z}_I)P(\mathrm{Z}_V \mid \mathrm{Z}_I), \tag{34}$$

$$P(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I) = P(\mathrm{A} \mid \mathrm{Z}_I)P(\mathrm{Z}_V \mid \mathrm{Z}_I). \tag{35}$$

Substituting the conditional independence into the definition of conditional mutual information, we obtain:

$$\begin{aligned} I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I) &= \mathbb{E}_{\mathrm{Z}_I}\left[\mathbb{E}_{\mathrm{A},\mathrm{Z}_V|\mathrm{Z}_I}\left[\log \frac{P(\mathrm{A} \mid \mathrm{Z}_I)P(\mathrm{Z}_V \mid \mathrm{Z}_I)}{P(\mathrm{A} \mid \mathrm{Z}_I)P(\mathrm{Z}_V \mid \mathrm{Z}_I)}\right]\right] \\ &= \mathbb{E}_{\mathrm{Z}_I}\left[\mathbb{E}_{\mathrm{A},\mathrm{Z}_V|\mathrm{Z}_I}\left[\log 1\right]\right] \\ &= 0. \end{aligned} \tag{36}$$

Therefore, when $P(\mathrm{A} \mid \mathrm{Z}_I, \mathrm{Z}_V) = P(\mathrm{A} \mid \mathrm{Z}_I)$, the conditional mutual information $I(\mathrm{A}, \mathrm{Z}_V \mid \mathrm{Z}_I)$ achieves its minimum value of 0. This implies that, given Z_I, there is no additional dependence between A and Z_V.

B. More Experiments

B.1. Architecture Visualizations

To clearly visualize the customized architectures tailored to different datasets, we presented a heatmap in Figure 4, illustrating the choice weights of each operation at each layer. Firstly, we observe that different graph datasets prefer distinct architectures; for example, Cora mainly prefers GraphConv and GraphSAGE, whereas these two operations are rarely selected for PubMed. This observation further supports our earlier assumption that different datasets require different architectures, and some datasets exhibit inconsistent architectural preferences. Moreover, we find that many datasets prefer varying operations across different layers. For instance, the Arxiv dataset prefers GCN in the first layer and GAT in the second layer. Such fine-grained architectural preferences are challenging to meet through manual design, highlighting the advantage of automated, customized architectures.

B.2. Hyperparameters Analysis

We analyze the sensitivity of the important hyperparameters λ and β in our method on the WikiCS dataset. We adjust $\lambda, \beta \in \{1e-1, 1e-2, 1e-3, 1e-4\}$, while maintaining the default value of the other hyperparameters unchanged.

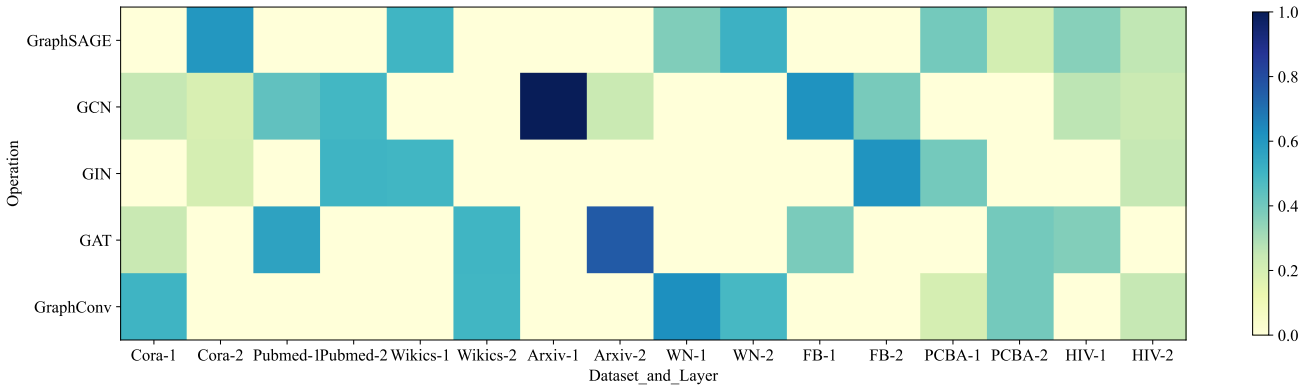

Figure 4. A showcase heatmap displays customized architectures for different datasets, where the number following the dataset name represents the architecture's layer.

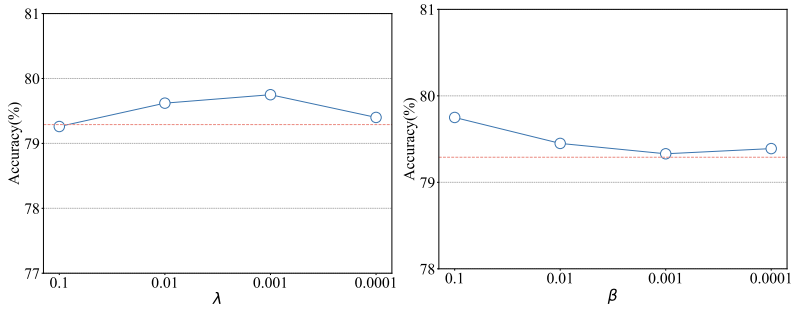

Figure 5. Hyperparameter sensitivity analysis on WikiCS dataset. The blue lines denote the results of our method and the red dashed lines are the results of the best baseline.

The results are shown in Figure 5. The hyperparameter λ in Equation (19) controls the trade-off between L_{task} and L_{dis}. Specifically, L_{task} aims to maximize the mutual information between the invariant pattern Z_I and the architecture A, ensuring that Z_I is sufficient to predict A. In contrast, L_{dis} aims to minimize the mutual information between the invariant pattern Z_I and the variant pattern Z_V, thereby enabling the extraction of two disjoint patterns from the data. We adjust its value within the set $\{1e-1, 1e-2, 1e-3, 1e-4\}$. As shown in Figure 5, when λ is set too low, the model's performance deteriorates, confirming that proper disentanglement of Z_I and Z_V is essential for effective architecture prediction. Conversely, when λ is set too high, performance also declines, indicating that while ensuring the separation between the two patterns, it is equally important that the Z_I retains sufficient information to predict the architecture. Overall, λ is an important hyperparameter for balancing the sufficiency and disentanglement. The hyperparameter β in Equation (19) controls the trade-off between L_{task} and L_{inv}. Specifically, L_{inv} aims to shield architecture A from the influence of Z_V given the invariant pattern Z_I. As demonstrated in Figure 5, setting β too low results in degraded model performance, underscoring the importance of effectively shielding A from the influence of Z_V given Z_I. Thus, β is also a critical hyperparameter for balancing the sufficiency and invariance conditions of the patterns captured by the model.

B.3. More Few-shot Learning results

We conduct more N-way K-shot experiments on the few-shot learning task. The results are shown in Table 3 and Table 4. Our method outperforms the baselines mostly across various N-way K-shot settings, further verifying the effectiveness of the customized architectures.

Table 3. Accuracy (%) with std of different methods on Cora under N-way K-shot settings. The highest result is **bold**.

	7-way			5-way			2-way		
Method	5-shot	3-shot	1-shot	5-shot	3-shot	1-shot	5-shot	3-shot	1-shot
OFA	32.10±1.79	36.03±2.11	30.38±2.39	42.28±2.35	31.28±2.63	23.68±1.67	72.20±3.82	62.22±1.17	51.85±4.35
GFT	43.55±7.43	43.31±8.11	41.40±8.04	52.30±6.57	51.47±6.33	49.80±6.79	75.00±4.08	76.33±3.56	72.92±4.64
GCN	42.57±7.50	42.38±7.42	43.07±7.37	45.00±7.40	44.53±7.93	44.80±8.94	71.58±5.20	71.42±4.03	70.58±4.01
GAT	47.71±8.02	47.31±7.78	46.12±7.10	52.30±6.05	51.73±7.32	50.17±7.41	75.92±3.89	75.17±5.36	72.83±5.48
GraphSAGE	42.40±6.12	42.07±6.12	40.50±6.11	51.17±5.13	50.80±5.34	49.50±5.55	74.20±2.95	74.33±2.47	72.08±5.89
GIN	47.24±7.33	47.02±7.32	45.29±6.26	49.83±7.79	49.17±8.10	48.97±6.73	75.25±8.60	**76.83±8.36**	71.50±7.44
GraphConv	41.60±9.20	40.93±8.80	38.67±8.50	46.13±9.73	44.90±10.41	42.57±8.63	67.75±9.84	67.00±11.48	63.42±8.37
Darts	42.81±7.92	42.10±7.45	43.29±7.65	49.23±7.69	49.50±7.18	46.97±6.00	71.67±4.35	71.42±5.54	69.58±6.01
GraphNAS	41.62±9.20	40.60±9.23	38.64±8.31	50.07±6.66	50.13±7.29	47.30±5.92	71.83±4.21	73.17±5.33	68.00±5.69
GASSO	42.95±5.75	42.07±5.85	40.31±6.10	49.53±6.60	50.87±6.10	47.53±7.40	71.08±6.42	70.92±5.26	68.00±4.41
GRACES	47.57±7.22	46.31±7.42	45.43±6.73	50.17±7.74	49.30±6.12	49.40±6.20	74.81±5.82	74.42±5.47	72.58±4.90
Ours	**47.76±8.06**	**47.33±7.80**	**46.29±7.24**	**53.93±6.95**	**52.50±6.84**	**50.87±5.55**	**76.43±5.45**	76.55±4.48	**73.92±6.64**

Table 4. Accuracy (%) with std of different methods on WN18RR under N-way K-shot settings. The highest result is **bold**.

	10-way			5-way			3-way		
Method	5-shot	3-shot	1-shot	5-shot	3-shot	1-shot	5-shot	3-shot	1-shot
OFA	32.64±1.56	30.56±1.02	25.82±1.07	48.32±3.19	45.04±2.39	34.40±1.47	60.72±3.82	61.29±2.56	51.77±2.65
GFT	35.50±4.59	35.50±5.02	35.33±4.20	48.80±3.61	48.53±3.68	48.13±4.37	62.56±2.71	60.67±3.93	58.44±3.84
GCN	30.40±3.02	29.78±3.64	29.85±4.14	44.70±2.99	44.97±3.95	44.77±3.45	54.06±5.36	53.33±5.64	53.28±4.77
GAT	34.70±3.15	34.37±3.43	34.50±2.98	46.23±4.44	46.33±4.50	46.30±4.43	59.56±3.85	59.39±3.45	58.06±4.34
GraphSAGE	38.30±2.16	38.17±2.34	38.03±2.03	48.10±3.78	47.83±3.88	47.90±3.66	62.39±3.48	61.44±3.48	59.39±3.50
GIN	37.47±3.10	36.92±4.03	36.62±4.17	47.57±5.56	47.80±5.29	47.60±3.81	61.33±5.98	61.83±6.35	58.22±4.93
GraphConv	39.62±3.44	39.28±2.27	38.93±3.77	48.40±3.38	47.63±2.51	46.07±3.23	61.39±4.11	60.06±3.60	59.33±3.43
Darts	38.65±3.55	38.57±3.22	37.22±2.68	47.40±4.36	46.03±3.61	46.43±3.20	60.17±2.32	58.78±4.63	57.89±3.62
GraphNAS	36.98±5.23	37.03±3.32	36.79±3.17	46.07±3.64	47.07±4.21	45.87±3.87	58.56±5.40	60.50±3.89	57.11±1.36
GASSO	37.37±3.31	37.42±1.91	37.13±3.52	47.90±4.14	47.77±3.63	46.20±3.24	59.61±4.94	59.83±5.20	57.83±4.06
GRACES	39.13±2.29	38.24±3.60	38.76±2.63	48.37±3.76	47.67±4.04	47.00±3.15	61.50±3.31	60.00±4.01	59.17±6.35
Ours	**40.02±2.26**	**39.55±2.46**	**39.34±3.03**	**49.93±3.63**	**49.10±3.31**	**48.47±4.38**	**63.11±5.80**	**61.94±2.61**	**59.72±4.26**

C. More Related Works

C.1. LLM-based Graph Foundation Models.

LLM-based Graph Foundation Models employ LLMs as predictors within a unified generative framework for graph tasks (Liu et al., 2023c; Pan et al., 2024a; Fang et al., 2024; Jin et al., 2023; Zhu et al., 2024a; Yu et al.; Huang et al., 2023; 2024b). For instance, InstructGLM (Ye et al., 2023) employs a generative framework in which LLMs predict node labels by generating them based on the nodes' textual attributes. GraphGPT (Tang et al., 2024b) adapts LLMs for downstream graph tasks through instruction tuning, integrating natural language with a graph-text aligner to capture and convey structural graph information. These approaches present a promising direction for the development of GFMs, as LLMs can seamlessly unify the output of various graph tasks. Unlike GNNs, which require task-specific adjustments for model training, LLMs can accept a wide range of queries and generate appropriate responses. However, a key challenge lies in effectively translating graph structures into a format that LLMs can interpret. Current research tackles this problem with two primary approaches. The first involves describing the graph structure using natural language (graph to text) (Zhao et al., 2023; Fatemi et al., 2023; Zhao et al., 2023; Wang et al., 2024a). The second approach draws inspiration from Visual Language Models (VLMs), where the graph is first processed into embeddings using GNNs or projectors (graph to token), and an LLM then decodes the graph embeddings (Chen et al., 2024; Tian et al., 2024; Tang et al., 2024a). These methods demonstrate competence in fundamental reasoning tasks such as connectivity checks and cycle detection, but struggle with more complex graph tasks

that require capturing intricate graph patterns, such as graph classification.

C.2. Graph Curriculum Learning

Curriculum learning is a training strategy that involves presenting training data in a meaningful order, typically starting with simpler examples and gradually progressing to more complex ones. This approach has been shown to improve the performance of various machine learning models (Bengio et al., 2009; Gong et al., 2015; Li et al., 2023; Chen et al., 2021;?; Zhang et al., 2022d; Zhou et al., 2022c;d; Chen et al., 2023; Zhou et al., 2023; Huang et al., 2024a; Zhou et al., 2024). Graph curriculum learning (GCL) is different from traditional curriculum learning due to the inherent dependencies of graph data(Gong et al., 2019; Zhou et al., 2022a; Wang et al., 2023; Wu et al., 2024). Researchers leverage graph structures to measure difficulty through predefined or automated strategies. For instance, CLNode(Wei et al., 2023) is a Curriculum Graph Learning method that measures local difficulty by considering the class diversity among a node's neighbors and uses global features to identify mislabeled nodes. RCL(Zhang et al., 2023a) gradually integrates node relationships into the training process, based on the complexity of those relationships. We utilize the concept of curriculum learning to enhance the architecture search process, mitigating the dominance of specific data on the search process.

D. Experimental Setup

D.1. Dataset

Table 5. Dataset statistics (Liu et al., 2023a).

Dataset	Domain	Task	# Graphs	Avg. #Nodes	Avg. #Edges	# Classes
Cora	Citation	Node	1	2,708	10,556	7
PubMed	Citation	Node	1	19,717	44,338	3
Arxiv	Citation	Node	1	169,343	1,166,243	40
WikiCS	Web link	Node	1	11,701	216,123	10
FB15K237	Knowledge	Link	1	14,541	310,116	237
WN18RR	Knowledge	Link	1	40,943	93,003	11
PCBA	Molecule	Graph	437,929	26.0	28.1	128
HIV	Molecule	Graph	41,127	25.5	27.5	2
ChEMBL	Molecule	Graph	365,065	25.9	55.9	1,048

Dataset Statistics. We follow the preprocessing method described in (Liu et al., 2023a; Wang et al., 2024b), employing the Sentence Transformer (Reimers & Gurevych, 2019) to convert raw textual descriptions of nodes and edges into 768-dimensional features. For knowledge graphs (KGs), we do not transform edge textual information into edge features, as the existing textual information already provides sufficient knowledge for KG completion (Wang et al., 2024b). The statistics of the datasets are detailed in Table 5.

Dataset Splitting. We adopt the same splitting strategy as (Liu et al., 2023a; Wang et al., 2024b). For **Cora** and **PubMed** select 20 labeled nodes per class for training. We utilize a predefined set of 10 splits with different random seeds to compute the average performance. For **WikiCS**, we report the average accuracy over 20 distinct training splits, each generated with 20 different random seeds. In each split, 5% of the nodes from each class are used for training. For **Arxiv, HIV**, and **PCBA**, we employ the official dataset splits and conduct experiments 10 times using different random seeds to determine the average accuracy. The **FB15K237** dataset consists of 272,115 edges in the training set, 17,535 edges in the validation set, and 20,466 edges in the test set. Meanwhile, for **WN18RR**, the corresponding numbers are 86,835, 3,034, and 3,134, respectively. Each experiment is repeated 10 times with different random seeds, and the final results are reported as the average accuracy.

D.2. Baseline

We compare our proposed **AutoGFM** with the following baselines and provide a brief description of each method:

- **Vanilla GNNs**
  - **GCN** (Kipf & Welling, 2017): Graph Convolutional Networks (GCN) utilizes graph convolutional layers to learn node representations by aggregating information from neighboring nodes.

  - **GAT** (Velickovic et al., 2017): Graph Attention Networks (GAT) uses attention mechanisms to weigh the importance of neighboring nodes when updating node representations.
  - **GIN** (Xu et al., 2018): Graph Isomorphism Network (GIN) employs a sum aggregation function and a learnable MLP to update node representations, aiming to achieve maximum discriminative power among graph structures.

- **Self-supervised GNNs**
  - **BGRL** (Thakoor et al., 2021): BGRL leverages a bootstrap-style contrastive learning approach without negative samples, maximizing agreement between online and target networks over augmented graph views.
  - **GraphMAE** (Hou et al., 2022): GraphMAE employs masked autoencoders to learn node representations by reconstructing masked node features from the input graph.
  - **GIANT** (Chien et al., 2022): GIANT is a self-supervised GNN framework that performs multi-granularity contrastive learning across multiple graphs to learn generalizable node representations.

- **GFMs**
  - **OFA** (Liu et al., 2023a): OFA is a GNN-based foundation model that integrates LLMs to process textual features across different domains and unifies graph-related tasks through subgraph classification.
  - **GFT** (Wang et al., 2024b): GFT is a GNN-based foundation model that incorporates computation trees to identify transferable patterns across graph structures, enabling the learning of generalizable representations for various graph domains and tasks.

- **Manually designed GNNs**
  - **GraphSAGE** (Hamilton et al., 2017): GraphSAGE learns node representations in an inductive manner by sampling and aggregating features from a node's local neighborhood using various aggregation functions such as mean or LSTM.
  - **GraphConv** (Morris et al., 2019): GraphConv is a generalized graph convolutional operator that combines node features with their neighbors' features using a trainable transformation, capturing local structure in the graph.

- **GNAS methods**
  - **DARTS** (Liu et al., 2018): DARTS is a differentiable architecture search framework that relaxes the search space into a continuous domain, enabling efficient gradient-based optimization of neural architectures.
  - **GraphNAS** (Gao et al., 2021): GraphNAS applies reinforcement learning to search for optimal GNN architectures by modeling the architecture design process as a sequential decision-making problem.
  - **GASSO** (Qin et al., 2021b): GASSO enables differentiable architecture search via gradient descent and discovers more effective graph neural architectures by incorporating graph structure learning as a denoising process during the search procedure.
  - **Graces** (Qin et al., 2022a): Graces achieves generalization under distribution shifts by designing instance-specific GNN architectures tailored to the unknown distribution of each graph.

For **Vanilla GNNs**, **self-supervised methods**, and **GFMs**, we reproduce the results based on their original papers and publicly available code. To ensure a fair comparison between **manually designed GNNs** and **GNAS** baselines, we employ GFT (Wang et al., 2024b) as the base model. Specifically, for **manually designed GNNs**, we replace the GNN in GFT with various manually designed GNNs and follow identical pretraining and finetuning procedures as GFT. For **GNAS** methods, we substitute the GNN component in GFT with different GNAS methods. Architecture search is performed during the pretraining stage, whereas in the finetuning stage, we further optimize only the parameters of the searched architectures without additional architecture searches.

D.3. Hyperparameters

We evaluate different GNN architectures and GNAS methods based on GFT (Wang et al., 2024b), following the default hyperparameters of GFT to maintain consistency. To ensure a fair comparison, we set the dimensionality of all methods to 768, use the same search space and operations (GCN, GIN, GAT, GraphSAGE, GraphConv), and fix the number of layers to 2. For our method, we explore hyperparameter $\lambda, \beta \in \{1e-1, 1e-2, 1e-3, 1e-4\}$ and empirically select λ and β. The

learning rate of the disentangled contrastive graph encoder is set to $5e-3$, and the learning rate of the architecture predictor is set to $3e-2$. The dimensionality of both the graph encoder and the supernet is 768. Each experiment is conducted 10 times, and we report the average performance along with standard deviations.

D.4. Configurations

We conduct all experiments on the following configurations:

- **Operating System**: Ubuntu 20.04.5 LTS
- **CPU**: Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz
- **GPU**: NVIDIA A100-SXM4-40GB and NVIDIA A100-SXM4-80GB
- **Software**: Python 3.9, CUDA 12.2, PyTorch (Paszke et al., 2019) 1.13.1.