Hypergraph Neural Architecture Search

Wei Lin[1,2], Xu Peng[1,2], Zhengtao Yu[3,4], Taisong Jin[1,2]*

[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, 361005, China.
[2]School of Informatics, Xiamen University, 361005, China.
[3]Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, 650500, China
[4]Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming, 650500, China
{lvvviolette, penglingxiao}@stu.xmu.edu.cn, ztyu@hotmail.com, jintaisong@xmu.edu.cn

Abstract

In recent years, Hypergraph Neural Networks (HGNNs) have achieved considerable success by manually designing architectures, which are capable of extracting effective patterns with high-order interactions from non-Euclidean data. However, such mechanism is extremely inefficient, demanding tremendous human efforts to tune diverse model parameters. In this paper, we propose a novel Hypergraph Neural Architecture Search (HyperNAS) to automatically design the optimal HGNNs. The proposed model constructs a search space suitable for hypergraphs, and derives hypergraph architectures through differentiable search strategies. A hypergraph structure-aware distance criterion is introduced as a guideline for obtaining an optimal hypergraph architecture via the leave-one-out method. Experimental results for node classification on benchmark Cora, Citeseer, Pubmed citation networks and hypergraph datasets show that HyperNAS outperforms existing HGNNs models and graph NAS methods.

Introduction

For real-world applications, there exist many irregular data structures, which can be effectively modeled via the graph structure. To extract the useful information from the graph structured data, graph neural networks (GNNs) have been proposed to address various learning tasks. The existing GNNs have achieved promising performance in real applications such as node classification (Xiao et al. 2022; Tian et al. 2023), community detection (Qiu et al. 2022), traffic forecasting (Jiang and Luo 2022) and biological problems (Bongini et al. 2023).

Graph neural networks (GNNs) model pairwise connections between two data samples via a normal graph. However, the data structure in real-world tasks may exceed pairwise relations that cannot be effectively modeled by normal graphs. Instead of edges in normal graphs, the hyperedge in a hypergraph can connect the arbitrary number of vertices. Thus, hypergraph-based learning methods are the more flexible and natural to extract the useful information from the complex graph data, which are attracting more and more attention from the researchers.

*Corresponding author.

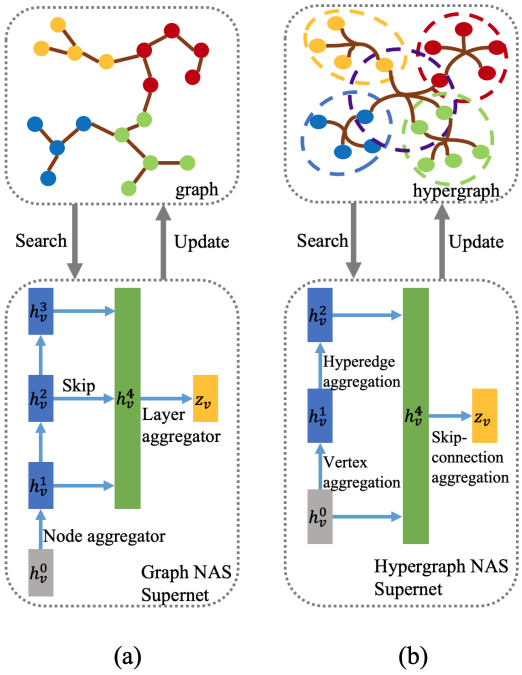

Figure 1: Intuitive comparison between our work and the existing graph NAS work. (a) Graph NAS is dedicated to the study of normal graphs. (b) Our work is based on hypergraphs.

In recent years, hypergraph neural networks (HGNNs) have become a popular learning tool to extract the complex patterns in non-Euclidean data, which is first proposed in (Feng et al. 2019). It designs a hyperedge convolution operation to leverage the high-order correlations across data. Since then, a variety of hypergraph neural networks have been proposed for different learning tasks, including but not limited to image retrieval (Zeng et al. 2023), quadratic assignment problem (Wang, Yan, and Yang 2021), biomedical science (Klimm, Deane, and Reinert 2021; Saifuddin et al. 2022), keypoint matching (Kim et al. 2022) and node classification (Bai, Zhang, and Torr 2021; Gao et al. 2022).

Although a lot of progress has been made in the literature,

it is intractable to design an effective hypergraph networks practically. Similar to CNNs that highly rely on the design of architectures, the results of hypergraph neural networks are primarily depended on the architecture design, including vertex feature aggregation (Arya et al. 2020) and hyperedge feature aggregation (Jiang et al. 2019; Gao et al. 2022). Obtaining the optimal architecture usually involves thousands of reiterative cross validation steps. It requires numerous domain knowledge and expert efforts, which is a labor-intensive task.

Until very recently, neural architectures search (NAS) (Zoph and Le 2017) has become an effective tool to address the aforementioned issue. Through learning in expressive search spaces and efficient search strategies, automatic graph learning via NAS has achieved promising progress on various graph data analysis tasks (Gao et al. 2021; Huan, Quanming, and Weiwei 2021; Zheng et al. 2023; Gao et al. 2023). Thus, applying NAS to derive the optimal neural architectures for GNNs motivates us to use NAS to search the best architecture for HGNNs. To the best of our knowledge, there are still a research gap in developing hypergraph NAS for automatic hypergraph learning. Perhaps the most direct approach to implementing hypergraph NAS is to employ NAS methods commonly used in normal graphs. But this straightforward solution would incur some issues: First, the inherent structural differences between normal graphs and hypergraphs make graph-based search spaces unsuitable for hypergraphs, as shown in Fig. 1. The key to hypergraph architecture design lies in the effective way to aggregate vertex and hyperedge features. Unlike graphs that only allow two nodes to connect, hypergraphs allow an arbitrary number of nodes to form hyperedges, thus graph-based feature aggregation methods cannot be used directly. Second, simply utilizing existing general architecture selection strategies limits search performance, resulting in suboptimal HGNN architectures.

To address the above challenges, we propose a novel hypergraph neural architecture search method, namely *HyperNAS*, for automatic hypergraph learning. The proposed model defines a search space suitable for hypergraphs, which can well emulate the artificially designed HGNN architectures. Then, HyperNAS designs a differentiable search algorithm and adopts the advanced one-shot NAS paradigm to train a supernet containing all candidate architectures. Furthermore, a hypergraph structure-aware distance criterion is introduced as a guideline for obtaining an optimal hypergraph architecture. Extensive experiments on benchmark graph and hypergraph datasets demonstrate the effectiveness of the proposed framework. Experimental results of the transfer learning task further demonstrate the power of HyperNAS. Our contributions are summarized as follows:

1. We propose a hypergraph neural architecture search method, termed HyperNAS, to enable automatic hypergraph learning. To the best of our knowledge, this is the first attempt to apply NAS to hypergraphs.
2. By designing a search space suitable for hypergraphs, HyperNAS emulates existing human-designed HGNN architectures. To select the optimal HGNN architecture guided by the hypergraph structure, we derive a hypergraph structure-aware distance criterion to obtain a formidable HGNN architecture in the leave-one-out manner.
3. The extensive experimental results on benchmark graph and hypergraph datasets demonstrate that the proposed method is capable to design the optimal neural architectures that outperform the manually-designed graph and hypergraph architectures as well as graph NAS methods.

Related Work

Hypergraph Neural Networks

Existing hypergraph neural architectures mainly contain three types of operators: hypergraph construction, vertex feature aggregation, and hyperedge feature aggregation. Inspired by convolutional neural networks, hypergraph neural network (HGNN) (Feng et al. 2019) is the first deep learning model for hypergraph, which leverages hypergraph Laplacian to represent hypergraphs from a spectral graph perspective. (Zhang, Zou, and Ma 2020; Bai, Zhang, and Torr 2021) generalize the convolutional operation or attention mechanism to hypergraph. HGNN+ (Gao et al. 2022) enables the learning of optimal representations in a single hypergraph framework by bridging multi-modal/multi-type data and hyperedge groups.

On the other hand, some studies are devoted to the dynamic modification of hypergraph structure for feature embedding. DHGNN (Jiang et al. 2019) utilizes k-means clustering strategy to update the hypergraph structure based on the local and global features, respectively. (Yin et al. 2022) extracts features of historical context content to provide guidance for dynamic hypergraph construction. (Yao et al. 2022) introduces the attention mechanism to achieve alternate update of vertices and hyperedges. Despite the desirable success of HGNNs, domain knowledge is highly required to manually design the architecture. In this paper, we utilize NAS to search feature aggregation operators for automatic hypergraph learning instead of manual design.

Neural Architecture Search

NAS-RL (Zoph and Le 2017) and MetaQNN (Baker et al. 2016) are considered as the pioneers in the field of NAS, which leverage reinforcement learning (RL) to search the suitable network architectures. However, the aforementioned methods spend hundreds of GPU days or even more computing resources on searching architectures. Darts (Liu, Simonyan, and Yang 2018) exploits differentiable NAS by relaxing discrete architectures into a continuous space, and jointly learns supernet weights and architecture weights, which greatly reduces the amount of calculation and speeds up the search. Furthermore, some of the works adopt the evolutionary algorithm (EA) (Real et al. 2019), bayesian optimization (BO) (White, Neiswanger, and Savani 2021), random search (Xie et al. 2018) and hybrid-based (Yang et al. 2020) strategies. With the improvement of search efficiency, NAS has been applied to object detection (Guo et al. 2020), image classification (He et al. 2023), text-to-image synthesis (Li et al. 2022) and the other fields.

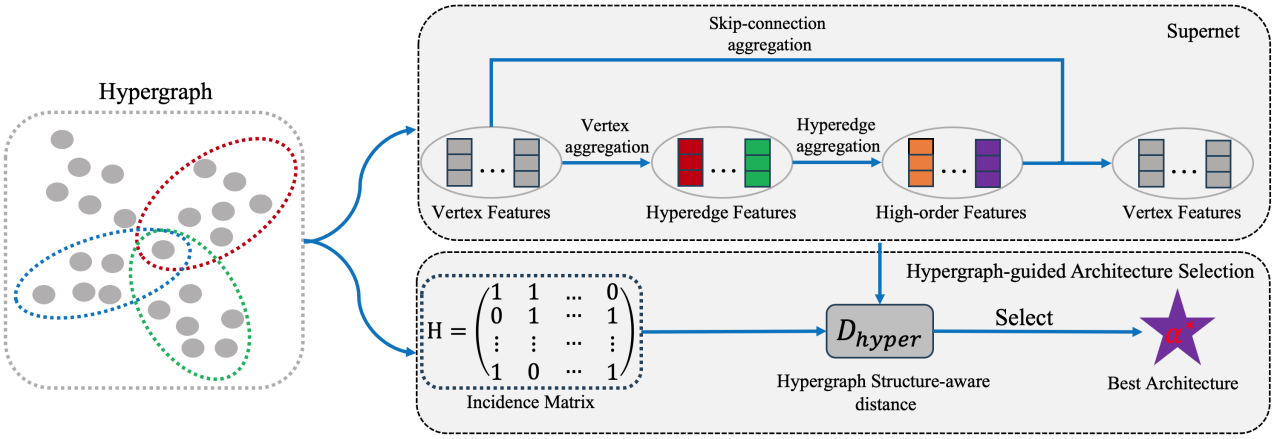

Figure 2: An illustration of the proposed HyperNAS framework. (Best viewed in color) The supernet is generated based on the constructed search space. Furthermore, the hypergraph structure is used to guide the architecture selection process, resulting in a superior architecture.

With the guidance of NAS, there are many studies focusing on extending neural architecture search to GNNs. GraphNAS (Gao et al. 2021) designs a search space to include operators from state-of-the-art GNNs and leverages reinforcement learning to solve the challenging problem of applying NAS to graphs. Concurrently, researchers have used more search strategies in the field of graph NAS, such as bayesian optimization (Yoon et al. 2020), evolution learning (Li and King 2020), random search (You, Ying, and Leskovec 2020). SANE (Huan, Quanming, and Weiwei 2021) uses a gradient-based search strategy, which greatly improves the search efficiency. (Zheng et al. 2023; Gao et al. 2023) further use NAS on the heterophilic graph.

In a summary, considering that NAS-based search has achieved the satisfactory results for CNNs, RNNs, as well as GNNs, we make the attempt to apply NAS to designing hypergraph neural architectures in this article.

Methodology

Definitions and Notations

A hypergraph is an extension of a graph, where normal graph is a special case of hypergraph. An edge in a graph connects only two vertices. Different from a normal graph, each hyperedge in a hypergraph can connect an arbitrary number of vertices. A hypergraph G is a pair $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set of elements, termed vertices, and $E = \{e_i = (v_1, \ldots, v_k)\}$ is a set of non-empty subsets of V called hyperedges. k is used to denote the number of nodes in the hyperedge. d represents the dimension of the vertex feature. A hypergraph G can be described by an $|V| \times |E|$ incidence matrix H. D_v and D_e denote the diagonal matrices of vertex degrees and edge degrees, respectively. Each hyperedge is assigned with a weight by w_e. The feature embedding is represented as $X = \{x_1, \ldots, x_n\}$, where $x_i (i = 1, \ldots, n)$ represents the feature of the i-th sample. X_v represents the vertex feature and X_e is the hyperedge feature. For k vertices, a $k \times k$ transformation matrix learned from the vertex features by multi-layer perception (MLP) is M_t, which takes into account the information of both the vertices and the channels.

Search Space

Fig. 2 shows the framework of the proposed model. Although a graph is a special case of a hypergraph, the search space in the previously proposed graph NAS cannot be directly applied to the hypergraph NAS. Thus, it is crucial to propose a search space suitable for hypergraph neural architecture search. As shown in Table 1, in order to design an expressive search space suitable for hypergraph, we focus on three key important parts: vertex aggregation, hyperedge aggregation and skip-connection aggregation, which are introduced as follows:

- Vertex Aggregation: The features of the hyperedge need to be obtained by aggregating the features of the vertices in the hyperedge. Specifically, hyperedge feature can be calculated by

$$X_e = conv(Merg(X_v^{(1)}, \cdots, X_v^{(k)})), \tag{1}$$

  where $X_v^{(i)}$ is the features of the i-th vertex in a hyperedge. The $Merg(\cdot)$ mechanism merges the message of all the vertices and the $conv(\cdot)$ operator indicates that 1-dimension convolution is used to compact the derived result, as is shown in Fig. 3. We denote the vertex aggregators set by $\mathcal{O}_v$.
- Hyperedge Aggregation: We regard each vertex as a center point c, and then aggregate the hyperedge features associated with it to obtain the high-order feature of c, denoted as X_h. The attention mechanism (Kim et al. 2020) is employed to generate the weights for each hyperedge in different ways. The high-order feature is calculated as

$$X_h = \sum_{i=0}^{m} w_e^{(i)} X_e^{(i)}, \tag{2}$$

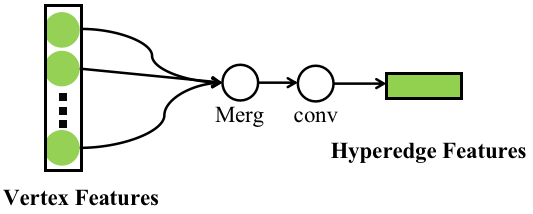

Figure 3: Vertex message aggregation. For k vertices, the corresponding features are aggregated via the aggregation function. Then 1-dimension convolution is performed on the aggregated features to obtain the hyperedge feature.

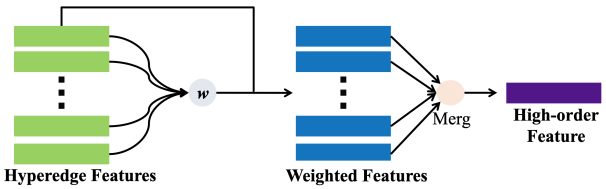

Figure 4: Hyperedge message aggregation. Weights for each hyperedge are generated, and then the high-order feature is calculated through the obtained weights.

Action	Values
O_v	$hgcn, hgat, sage_{sum}, sage_{mean}, trans$
O_h	$att_{sum}, att_{mean}, att_{max}, att_{mlp}$
O_s	$fusion, sum, mean, max$

Table 1: Search space $\mathcal{A}$.

where m represents the number of hyperedges associated with the centroid vertex, and w represents the calculated weights of hyperedges. The main process is shown in Fig. 4, where the *Merg* operation merges the weighted Features, such as sum. The process of calculating weights is similar to the pooling operation in CNN. We denote the hyperedge aggregators set by $\mathcal{O}_h$.

- Skip-connection Aggregation: Skip connection has been shown to be effective in previous work (Ji et al. 2020). In this work, we use four different methods to aggregate high-order features and original centroid vertex features to obtain new centroid vertex features. We denote the skip-connection aggregators set by $\mathcal{O}_s$.

Differentiable Search

Inspired by (Liu, Simonyan, and Yang 2018), we relax the categorical choice of a particular operation to a softmax over all possible operations to make the search space continuous:

$$\bar{o}(x) = \sum_{o \in \mathcal{O}} \frac{exp(\alpha_o)}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'})} o(x), \tag{3}$$

where the operation mixing weights for each node c are parameterized by a vector α of dimension $|\mathcal{O}|$, and $\mathcal{O}$ is drawn from three sets of operations: $\mathcal{O}_v, \mathcal{O}_h, \mathcal{O}_s$, as described in the previous section. Then the corresponding $\alpha_v, \alpha_h, \alpha_s$ can be calculated. x represents input features of a certain layer.

Let $\bar{o}_v$,$\bar{o}_h$, and $\bar{o}_s$ be the mixed operations of O_v, O_h, and O_s based on Eq. 3 respectively. The vertex aggregation and the hyperedge aggregation processes of HyperNAS are

$$X_e = \bar{o}_v(X_v^{(k)}, \forall X_v^{(k)} \in e), \tag{4}$$

$$X_h = \bar{o}_h(X_e, \forall e \ni c). \tag{5}$$

Ultimately, the final embedding of the center point c is computed as:

$$Z_v = \bar{o}_s(X_v^{(c)}, X_h), \tag{6}$$

As can be seen from the aforementioned equations, the computation process is the summation of all operations within the corresponding sets. In general, HyperNAS is a bi-level optimization task:

$$\min_{\alpha \in \mathcal{A}} \mathcal{L}_{val}(w^*(\alpha), \alpha), \tag{7}$$

$$s.t. w^*(\alpha) = arg\min_{w} \mathcal{L}_{tra}(w, \alpha), \tag{8}$$

where $\mathcal{L}_{tra}$ and $\mathcal{L}_{val}$ denote the training and validation loss respectively. $\alpha = \alpha_v, \alpha_h, \alpha_s$ as the upper-level variable represents the network architecture, and $w^*(\alpha)$ as the lower-level variable is the corresponding weight after training.

Hypergraph-guided Architecture Selection

In general, the architecture parameter α is optimized on the validation data (i.e. Eq. 7), and the network weights w are optimized on the training data (i.e. Eq. 8). With the above sequential relaxation, we can apply the recently popular one-shot NAS method (Liu, Simonyan, and Yang 2018; Yao et al. 2020). Specifically, following Darts (Liu, Simonyan, and Yang 2018), we give gradient-based approximations for optimization:

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha) \approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{tra}(w, \alpha), \alpha), \tag{9}$$

where w is the current weight and ξ is the learning rate for optimizing w.

However, the above approach implies that only operations corresponding to the largest edge weights in the architecture supernet can be kept to construct the final HGNN. However, (Wang et al. 2021) has demonstrated that architectural scale is not sufficient to indicate operational strength in the final architectural choice for graph NAS. Therefore, we only use the gradient descent strategy as training.

In the node classification setting, (Zhu et al. 2021) proposes a compatibility matrix to model the linked possibility of nodes in any two classes. (Zheng et al. 2023) further restricts the connection possibility of nodes in any two classes predicted by the supernet to approximate the ground truth. In order to select optimal HGNN architectures guided exclusively by the hypergraph structure, we derive the hypergraph structure-aware distance as a criterion to guide the selection of HGNN architectures. Specifically, the proposed hypergraph structure-aware distance D_{hyper} can be defined by the Euclidean distance as $D_{hyper} = ||\hat{S} - S||$, where $\hat{S}$ and S are the hypergraph structure-aware matrices denoted as:

$$S = (Y^T \Theta Y) \oslash (Y^T \Theta E), \hat{S} = (\hat{Y}^T \Theta \hat{Y}) \oslash (\hat{Y}^T \Theta E), \tag{10}$$

Algorithm 1: Hypergraph neural architecture search

Input: Search space $\mathcal{A}$, number of gradient descent iterations T, edge set $\mathcal{E}$ of supernet $\mathcal{S}$.
Output: A HGNN model with the set of selected operations $\alpha^* = \{\alpha_v^*, \alpha_h^*, \alpha_s^*\}$.

for t = 1, $\cdots$, T **do**
  Compute the validation loss $\mathcal{L}_{val}$;
  Update α_v, α_h and α_s by gradient descend rule (9) with (4), (5) and (6) respectively;
  Compute the training loss $\mathcal{L}_{tra}$;
  Update weights w by descending $\nabla_w \mathcal{L}_{tra}(w, \alpha)$ with the architecture $\alpha = \{\alpha_v, \alpha_h, \alpha_s\}$;
end for
while $|\mathcal{E}| > 0$ **do**
  Randomly remove an edge $e \in \mathcal{E}$;
  for all operations $o \in \mathcal{O}$ on edge e **do**
    Calculate $D_{hyper}(\backslash o)$ when o is removed;
  end for
  Select the best operation by $\text{argmin}_o\, D_{hyper}(\backslash o)$;
end while
return The best operation $\alpha^* = \{\alpha_v^*, \alpha_h^*, \alpha_s^*\}$.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,278	1,433	7
Citeseer	3,327	4,552	3,703	6
Pubmed	19,717	44,324	500	3
Coauthor-CS	18,333	81,894	6,805	15
Photos	7,650	119,081	745	8
Computers	13752	245,861	767	10

Table 2: Details of datasets with graph structure.

where Y and $\hat{Y}$ are the ground-truth and predicted label matrix for all nodes, E is the all-ones matrix, and $\oslash$ denotes the Hadamard division. Θ represents the adjacency matrix of the hypergraph, which is defined as:

$$\Theta = HWD_e^{-1}H^T, \tag{11}$$

where W represents the hyperedge weight matrix. The proposed hypergraph structure-aware distance D_{hyper} constrains the discrepancy between the predicted label and the ground-truth label by virtue of the hypergraph structure information. The smaller D_{hyper} is, the better the ability of the candidate operation to discriminate the node class. Guided by D_{hyper}, we select the optimal HGNN architecture from the pre-trained supernet S_c. Furthermore, inspired by (Wang et al. 2021; Zheng et al. 2023), we adopt the leave-one-out method to directly evaluate the contribution of each candidate operation to the supernet. In general, HyperNAS first constructs a comprehensive hypergraph search space, and then uses gradient descent to train. After that, HyperNAS selects the final HGNN architecture guided by the hypergraph structure-aware distance. The main process of HyperNAS is listed in Algorithm 1.

Dataset	DBLP	Cora-CA
Nodes	43413	2708
Hyperedges	22535	1072
Avg. hyperedge size	4.7±6.1	4.2±4.1
Feature	1425	1433
Classes	6	7

Table 3: Details of co-authorship hypergraph datasets.

Experiments

Datasets

Since graph is a special case of hypergraph, the proposed model is employed on a typical graph-structured dataset: Citation Network dataset (Sen et al. 2008). The label rate (LR) represents the number of labeled vertices used for training.

To understand whether the model generated by HyperNAS can be generalized to different tasks, we apply HyperNAS to the node classification on different datasets including the Coauthor-CS, Amazon-Photos and Amazon-Computers (Shchur et al. 2018). The Details of the graph structure datasets are shown in Table 2

Furthermore, we apply HyperNAS to real-world hypergraph datasets: DBLP and Cora Co-authorship(Cora-CA), as shown in Table 3. Each hyperedge represents all documents co-authored by an author in co-authorship dataset.

Baselines

We compare the proposed HyperNAS with two categories of the baseline methods including the human-designed GNN/HGNN models and the graph NAS methods including GCN (Welling and Kipf 2017), GAT (Velickovic et al. 2017), HGNN (Feng et al. 2019), DHGNN (Jiang et al. 2019), HCHA (Bai, Zhang, and Torr 2021), HGNN$^+$ (Gao et al. 2022), GraphNAS (Gao et al. 2021), SANE (Huan, Quanming, and Weiwei 2021), HGNAS++ (Gao et al. 2023). For the transfer learning, we select six state-of-the-art models as baseline: GCN, GAT, GMI (Peng et al. 2020), MVGRL (Hassani and Khasahmadi 2020), GCA (Zhu et al. 2021), DHGNN and SANE. We use the node classification accuracy as the metric to evaluate the performance of different methods.

Experimental Settings

Train/Validation/Test Split For the Cora citation dataset, we follow (Jiang et al. 2019) as the experimental setup of the proposed method. We choose the standard split (Yang, Cohen, and Salakhudinov 2016) and randomly select different proportions (2%, 5.2%, 10%, 20%, 30%, and 44%) of data as training sets to evaluate the performance of the compared methods. For the Citeseer and Pubmed datasets, we follow the experimental setup in (Welling and Kipf 2017), where 3.6% in the Citeseer dataset is used for training and 0.3% in the Pubmed dataset. For the Coauthor and Amazon datasets, we randomly select 30 nodes from each class to build the training and validation sets, and then use the remaining nodes as the test set. For two co-authorship hy-

Type	Method	Accuracy(%)		
		Cora	Citeseer	Pubmed
Human-designed models	GCN (Welling and Kipf 2017)	81.4	70.9	79.0
	GAT (Velickovic et al. 2017)	83.0	72.5	79.0
	HGNN (Feng et al. 2019)	81.6	71.9	80.1
	DHGNN (Jiang et al. 2019)	82.5	70.0	79.9
	HCHA (Bai, Zhang, and Torr 2021)	82.7	71.2	78.4
	HGNN$^+$ (Gao et al. 2022)	83.3	73.0	80.7
Graph NAS	GraphNAS (Gao et al. 2021)	83.2	73.5	80.3
	SANE (Huan, Quanming, and Weiwei 2021)	83.6	73.9	81.0
	HGNAS++ (Gao et al. 2023)	83.5	73.8	81.1
Hypergraph NAS	Random (ours)	82.8	73.6	80.7
	HyperNAS-RL (ours)	83.3	73.7	80.9
	HyperNAS (ours)	**83.9**	**74.1**	**81.3**

Table 4: Comparison of accuracies on citation networks. "Random" and "HyperNAS-RL" represent two variants of HyperNAS.

LR(%)	#Train	Accuracy(%)				
		GCN	GAT	HGNN	DHGNN	HyperNAS
std	140	81.4	83.0	81.6	82.5	**83.9**
2	54	69.6	74.8	75.4	76.9	**79.1**
5.2	140	77.8	79.4	79.7	80.2	**82.4**
10	270	79.9	81.5	80.0	81.6	**84.5**
20	540	81.4	83.5	80.1	83.6	**84.9**
30	812	81.9	84.5	82.0	85.0	**85.6**
44	1200	82.0	85.2	81.9	85.6	**86.1**

Table 5: Node classification accuracies for different splits on Cora. "LR" represents label rate, "#Train" represents the number of training samples and "std" represents Cora standard split. The standard split experiment is a fixed training set, whereas the training sets of the other experiments are randomly selected. Regardless of how the train set is chosen, the experiments share the same validation and test sets.

Method	Accuracy(%)	
	DBLP	Cora-CA
HGNN (Feng et al. 2019)	74.9	66.3
HyperGCN (Yadati et al. 2019)	76.3	69.1
HyperSAGE (Arya et al. 2020)	77.2	72.1
HGNN$^+$ (Gao et al. 2022)	77.3	72.2
HyperNAS(ours)	**77.6**	**72.6**

Table 6: Performance of HyperNAS and other hypergraph learning methods on hypergraph datasets.

pergraph datasets, We use the same train-test split provided by (Yadati et al. 2019) in their public implementation.

Hypergraph Construction The hypergraph construction technique introduced in (Jiang et al. 2019) is employed for generating hypergraphs from graph datasets. When constructing the hyperedges, 400 cluster centers are used in the k-means and each hyperedge consists of 64 vertices.

Training Details Following the setup in (Huan, Quanming, and Weiwei 2021), we search with different random seeds and fine-tune the hyperparameters on the validation data. Dropout layers with the dropout rate of 0.5 are applied to avoid the over-fitting. We use Adam optimizer to optimize our cross-entropy loss function with a learning rate of 0.01. All experiments are performed on a single NVIDIA 3090.

Results and Discussions

For the node classification problem, we build the HyperNAS model as shown in Fig. 2. For the proposed HyperNAS, we conduct the node classification experiments with 30 standard splits on Cora, Citeseer, and Pubmed, and report the average results. The experimental and comparison results of semi-supervised node classification are listed in Table 4.

As shown in Table 4, the proposed HyperNAS method consistantly outperforms the baseline methods. The HyperNAS approach exhibits remarkable performance gains in comparison to prevailing human-designed methods and Graph NAS methods when evaluated on standard splits.

We further compare the architectures designed by HyperNAS with the graph/hypergraph-based neural network baselines on the different dataset splits. The experimental results are listed in Table 5.

As shown in Table 5, when 2%, 5.2%, 10%, 20%, 30%, and 44% of randomly sampled data are used as training sets for the Cora dataset, the proposed HyperNAS outperforms the DHGNN by a margin of 2.2%, 2.2%, 2.9%, 1.3%, 0.6%, 0.5%, respectively. Compared to graph neural networks, the performance enhancement of hypergraph neural networks is more evident when the size of training set is small. These experimental results further demonstrate that the model de-

Method	Accuracy(%)		
	Coauthor-CS	Photos	Computers
GCN (Welling and Kipf 2017)	85.8	92.4	75.3
GAT (Velickovic et al. 2017)	85.9	91.0	76.1
GMI (Peng et al. 2020)	83.9	91.7	78.2
MVGRL (Hassani and Khasahmadi 2020)	86.1	89.7	79.6
GCA (Zhu et al. 2021)	89.4	91.8	81.5
DHGNN (Jiang et al. 2019)	89.1	92.1	81.9
SANE (Huan, Quanming, and Weiwei 2021)	89.4	92.5	82.1
HyperNAS (ours)	**89.7**	**92.7**	**82.5**

Table 7: Performance comparisons of transferring HyperNAS architectures designed on Citation Network to the other datasets.

Selection strategy	Accuracy(%)		
	Cora	Citeseer	Pubmed
Weight size-based	83.5	73.8	80.9
Validation loss-based	83.3	73.6	80.8
Hyper-guided (ours)	**83.9**	**74.1**	**81.3**

Table 8: Comparison of the proposed hypergraph-guided architecture selection with other selection methods.

signed by hypergraph neural architecture search has the superior performance than the existing hypergraph neural network. Specifically, as the size of training set increases, the performance gain of the proposed method is still evident.

Furthermore, we evaluate the proposed method on two co-authorship hypergraph datasets to investigate their ability to solve hypergraph-related problems. The results, as presented in Table 6, show that HyperNAS exhibits superior performance in hypergraph-based node classification tasks, surpassing the performance of existing methods.

For transfer learning, we apply the architectures obtained by HyperNAS in the citation networks to employ the node classification task on the different datasets including the Coauthor network Coauthor-CS and the product networks of Amazon-Photos and Amazon-Computers. The experimental results of transfer learning are listed in Table 7.

From Table 7, we observe that the model designed by HyperNAS is competitive in predictive accuracy when the derived architecture is transferred to new datasets for node classification task. These experimental results demonstrate that the proposed HyperNAS can train more expressive HGNNs, which can easily be transferred to the other graph datasets to employ the node classification task.

Variants of HyperNAS

We construct two variants: (1) Random search (denoted as "Random") (Bergstra and Bengio 2012): architectures are randomly selected; (2) Reinforcement learning (denoted as "HyperNAS-RL") (Zoph and Le 2017): hypergraph NAS is implemented using reinforcement learning.

Table 4 lists the results of the architectures designed by two variants on citation networks. We further conduct experiments for different splits on Cora dataset and show the performance in Figure 5. Experimental results show that HyperNAS has the best test accuracy, which means our proposed

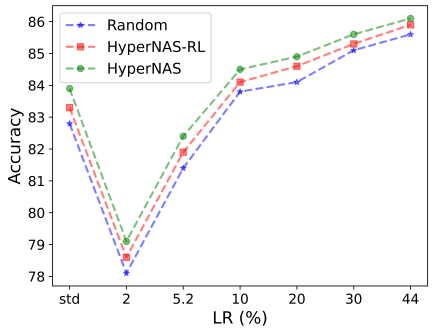

Figure 5: Classification accuracy of Random, HyperNAS-RL and HyperNAS for different splits on Cora dataset.

method outperforms other variants.

Ablation Experiments

The effectiveness of hypergraph-guided architecture selection. We compare the proposed hypergraph-guided architecture selection with two alternative methods, namely weight size-based (Liu, Simonyan, and Yang 2018) and validation loss-based (Wang et al. 2021) approaches. The comparison results are listed in Table 8. In general, our proposed hypergraph-guided scheme with hypergraph structure-aware distance criteria achieves the best classification performance consistently on all datasets, illustrating its effectiveness in architecture selection.

Conclusion

In this paper, we have proposed a novel hypergraph neural architecture search (HyperNAS), which is the first attempt to apply neural architecture search to hypergraph neural networks. The propose method is designed to automatically design the optimal hypergraph neural architecture via the NAS approach. The extensive experimental results on the benchmark graph and hypergraph datasets demonstrates the efficacy of the proposed method. For the future work, we note that the scalability of the proposed method is an important issue, specifically for the large-scale graph datasets.

Acknowledgements

This work was supported by National Key R&D Program of China (No. 2022ZD0118202), in part by the National Natural Science Foundation of China (Nos. 62072386, U21B2027,61972186, U23A20388 and 62266028), in part by Yunnan Provincial Major Science and Technology (Nos. 202302AD080003, 202202AD080003 and 202303AP140008), in part by the General Projects of Basic Research in Yunnan Province (Nos. 202301AS070047, 202301AT070471). We also thank Yayao Hong for valuable discussions.

References

Arya, D.; Gupta, D. K.; Rudinac, S.; and Worring, M. 2020. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv preprint arXiv:2010.04558*.

Bai, S.; Zhang, F.; and Torr, P. H. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110: 107637(1–8).

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing Neural Network Architectures using Reinforcement Learning. In *Proceedings of International Conference on Learning Representations*.

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2): 281–305.

Bongini, P.; Pancino, N.; Scarselli, F.; and Bianchini, M. 2023. BioGNN: How Graph Neural Networks Can Solve Biological Problems. In *Artificial Intelligence and Machine Learning for Healthcare*, 211–231.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; and Gao, Y. 2019. Hypergraph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Gao, Y.; Feng, Y.; Ji, S.; and Ji, R. 2022. HGNN+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3181–3199.

Gao, Y.; Wang, M.; Tao, D.; Ji, R.; and Dai, Q. 2012. 3-D object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing*, 21(9): 4290–4303.

Gao, Y.; Yang, H.; Zhang, P.; Zhou, C.; and Hu, Y. 2021. Graph neural architecture search. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Gao, Y.; Zhang, P.; Zhou, C.; Yang, H.; Li, Z.; Hu, Y.; and Philip, S. Y. 2023. HGNAS++: efficient architecture search for heterogeneous graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*.

Guo, J.; Han, K.; Wang, Y.; Zhang, C.; Yang, Z.; Wu, H.; Chen, X.; and Xu, C. 2020. Hit-detector: Hierarchical trinity architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *Proceedings of International Conference on Machine Learning*.

He, X.; Yao, J.; Wang, Y.; Tang, Z.; Cheung, K. C.; See, S.; Han, B.; and Chu, X. 2023. NAS-LID: Efficient Neural Architecture Search with Local Intrinsic Dimension. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Huan, Z.; Quanming, Y.; and Weiwei, T. 2021. Search to aggregate neighborhood for graph neural network. In *Proceedings of IEEE International Conference on Data Engineering*.

Huang, Y.; Liu, Q.; and Metaxas, D. 2009. Video object segmentation by hypergraph cut. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Ji, S.; Feng, Y.; Ji, R.; Zhao, X.; Tang, W.; and Gao, Y. 2020. Dual channel hypergraph collaborative filtering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Jiang, J.; Wei, Y.; Feng, Y.; Cao, J.; and Gao, Y. 2019. Dynamic Hypergraph Neural Networks. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Jiang, W.; and Luo, J. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207: 117921(1–28).

Kim, E.-S.; Kang, W. Y.; On, K.-W.; Heo, Y.-J.; and Zhang, B.-T. 2020. Hypergraph attention networks for multimodal learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Kim, J.; Oh, S.; Cho, S.; and Hong, S. 2022. Equivariant Hypergraph Neural Networks. In *Proceedings of European Conference on Computer Vision*.

Klimm, F.; Deane, C. M.; and Reinert, G. 2021. Hypergraphs for predicting essential genes using multiprotein complex data. *Journal of Complex Networks*, 9(2): cnaa028(1–25).

Li, W.; Wen, S.; Shi, K.; Yang, Y.; and Huang, T. 2022. Neural architecture search with a lightweight transformer for text-to-image synthesis. *IEEE Transactions on Network Science and Engineering*, 9(3): 1567–1576.

Li, Y.; and King, I. 2020. Autograph: Automated graph neural network. In *Proceedings of International Conference on Neural Information Processing*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Peng, Z.; Huang, W.; Luo, M.; Zheng, Q.; Rong, Y.; Xu, T.; and Huang, J. 2020. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference*.

Qiu, C.; Huang, Z.; Xu, W.; and Li, H. 2022. VGAER: graph neural network reconstruction based community detection. *arXiv preprint arXiv:2201.04066*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Saifuddin, K. M.; Bumgardnerr, B.; Tanvir, F.; and Akbas, E. 2022. HyGNN: Drug-Drug Interaction Prediction via Hypergraph Neural Network. *arXiv preprint arXiv:2206.12747*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI Magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Tian, Y.; Dong, K.; Zhang, C.; Zhang, C.; and Chawla, N. V. 2023. Heterogeneous graph masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y.; et al. 2017. Graph Attention Networks. *Stat*, 1050(20): 10–48550.

Wang, R.; Cheng, M.; Chen, X.; Tang, X.; and Hsieh, C.-J. 2021. Rethinking Architecture Selection in Differentiable NAS. In *Proceedings of International Conference on Learning Representations*.

Wang, R.; Yan, J.; and Yang, X. 2021. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9): 5261–5279.

Welling, M.; and Kipf, T. N. 2017. Semi-supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations*.

White, C.; Neiswanger, W.; and Savani, Y. 2021. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xiao, S.; Wang, S.; Dai, Y.; and Guo, W. 2022. Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications*, 33(1): 1–19.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Yadati, N.; Nimishakavi, M.; Yadav, P.; Nitin, V.; Louis, A.; and Talukdar, P. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in Neural Information Processing Systems*.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of International Conference on Machine Learning*.

Yang, Z.; Wang, Y.; Chen, X.; Shi, B.; Xu, C.; Xu, C.; Tian, Q.; and Xu, C. 2020. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yao, F.; Sun, X.; Liu, N.; Tian, C.; Xu, L.; Hu, L.; and Ding, C. 2022. Hypergraph-Enhanced Textual-Visual Matching Network for Cross-Modal Remote Sensing Image Retrieval via Dynamic Hypergraph Learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 16: 688–701.

Yao, Q.; Xu, J.; Tu, W.-W.; and Zhu, Z. 2020. Efficient neural architecture search via proximal iterations. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yin, N.; Feng, F.; Luo, Z.; Zhang, X.; Wang, W.; Luo, X.; Chen, C.; and Hua, X.-S. 2022. Dynamic hypergraph convolutional network. In *Proceedings of the IEEE International Conference on Data Engineering*.

Yoon, M.; Gervet, T.; Hooi, B.; and Faloutsos, C. 2020. Autonomous graph mining algorithm search with best speed/accuracy trade-off. In *Proceedings of the IEEE International Conference on Data Mining*.

You, J.; Ying, Z.; and Leskovec, J. 2020. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33: 17009–17021.

Zeng, Y.; Jin, Q.; Bao, T.; and Li, W. 2023. Multi-Modal Knowledge Hypergraph for Diverse Image Retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, R.; Zou, Y.; and Ma, J. 2020. Hyper-SAGNN: a self-attention based graph neural network for hypergraphs. In *Proceedings of International Conference on Learning Representations*.

Zhang, Z.; Lin, H.; Gao, Y.; and BNRist, K. 2018. Dynamic hypergraph structure learning. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Zheng, X.; Zhang, M.; Chen, C.; Zhang, Q.; Zhou, C.; and Pan, S. 2023. Auto-HeG: Automated Graph Neural Network on Heterophilic Graphs. In *Proceedings of the ACM Web Conference*.

Zhou, D.; Huang, J.; and Schölkopf, B. 2006. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in Neural Information Processing Systems*, 19.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference*.

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