Do Not Train It: A Linear Neural Architecture Search of Graph Neural Networks

Peng Xu[1*], Lin Zhang[2*], Xuanzhou Liu[3], Jiaqi Sun[3], Yue Zhao[4], Haiqin Yang[2], Bei Yu[1]

The Chinese University of Hong Kong, International Digital Economy Academy, Tsinghua University, Carnegie Mellon University

Highlights

- We convert the traditional NAS problem into a sparse coding task with linear complexity of backward computing called **NAC**.
- We have theoretically justified the power of untrained GNN in the NAS-GNN problem, where updating the neural weights is not necessary.
- NAC achieves higher accuracy (up to 18.8%) and much faster convergence (up to 200×) without any updating on neural weights.

Background

NAS in Graph Neural Networks(NAS-GNN)

- Sample-based
  1. Reinforcement Learning-based: GraphNAS [gao2020graphnas]
- Weight-Sharing-based
  1. Differential-based: SANE [huan2021sane]

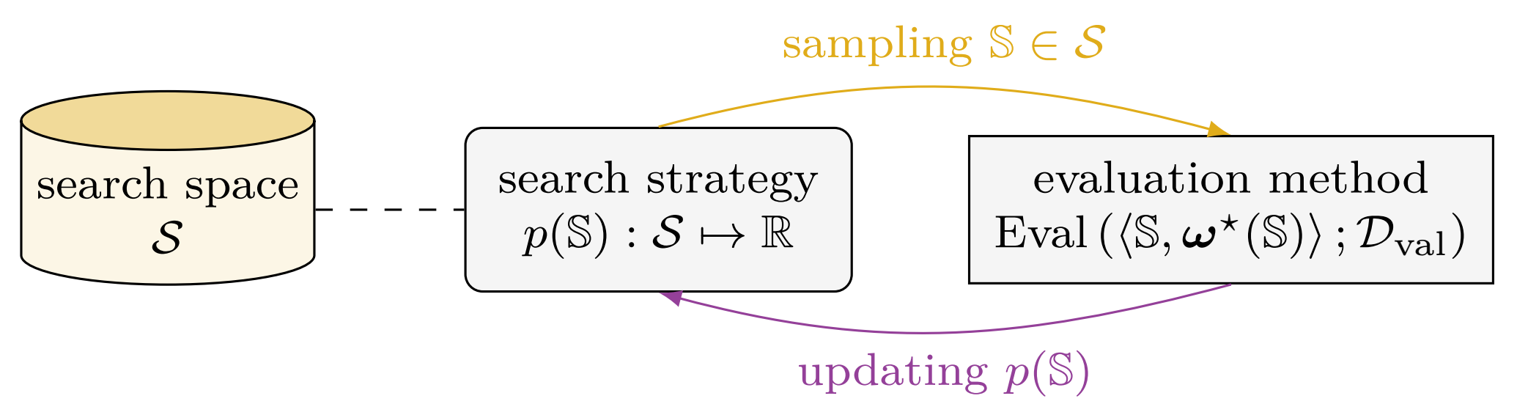
sampling $\mathbb{S} \in \mathcal{S}$
search space $\mathcal{S}$
search strategy $p(\mathbb{S}) : \mathcal{S} \mapsto \mathbb{R}$
evaluation method Eval $(\langle \mathbb{S}, \boldsymbol{\omega}^\star(\mathbb{S}) \rangle ; \mathcal{D}_{\text{val}})$
updating $p(\mathbb{S})$

Figure 1. The three components of NAS.

Differential-based NAS-GNN

The differential-based method addresses *two* subproblems of weights and architectures alternatively using a bi-level optimization framework to reduce the search effort [huan2021sane]:

$$\begin{cases} \alpha^* = \underset{\alpha}{\operatorname{argmax}} \mathcal{L}_{val}\left(w^*(\alpha), \alpha\right), \\ w^* = \underset{w}{\operatorname{argmin}} \mathcal{L}_{train}(\alpha, w) \end{cases} \tag{1}$$

where $\mathcal{L}$ denotes a loss function (e.g., cross-entropy loss) on training and validating dataset w.r.t w, and α is the architecture parameter.

Motivation

Existing Problems Two optimization challenges cause the instability of the differential-based methods:

- Optimization challenge due to bi-level optimization
- Inaccurate estimation caused by softmax

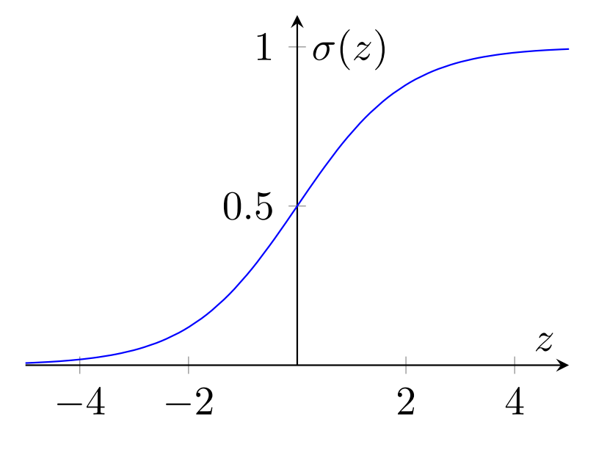

Figure 2. Softmax

In this work, we attempt to address the optimization difficulty in DARTS from a new perspective.

- **No-update Scheme**: We utilize the power of untrained GNN [kipf2016gcn] in the NAS-GNN problem and provide a theoretical analysis.
- **Sparse Coding**: The ultimate goal of NAS is to assign the coefficients of unimportant operators as zeros.

Methodology

The optimality of untrained GNNs and the connection between NAS in GNNs and sparse coding allow us to waive the effort of updating weights and make us focus on architecture updates.
As shown in the Fig. 3, NAC directly learns architecture α with a fixed dictionary.

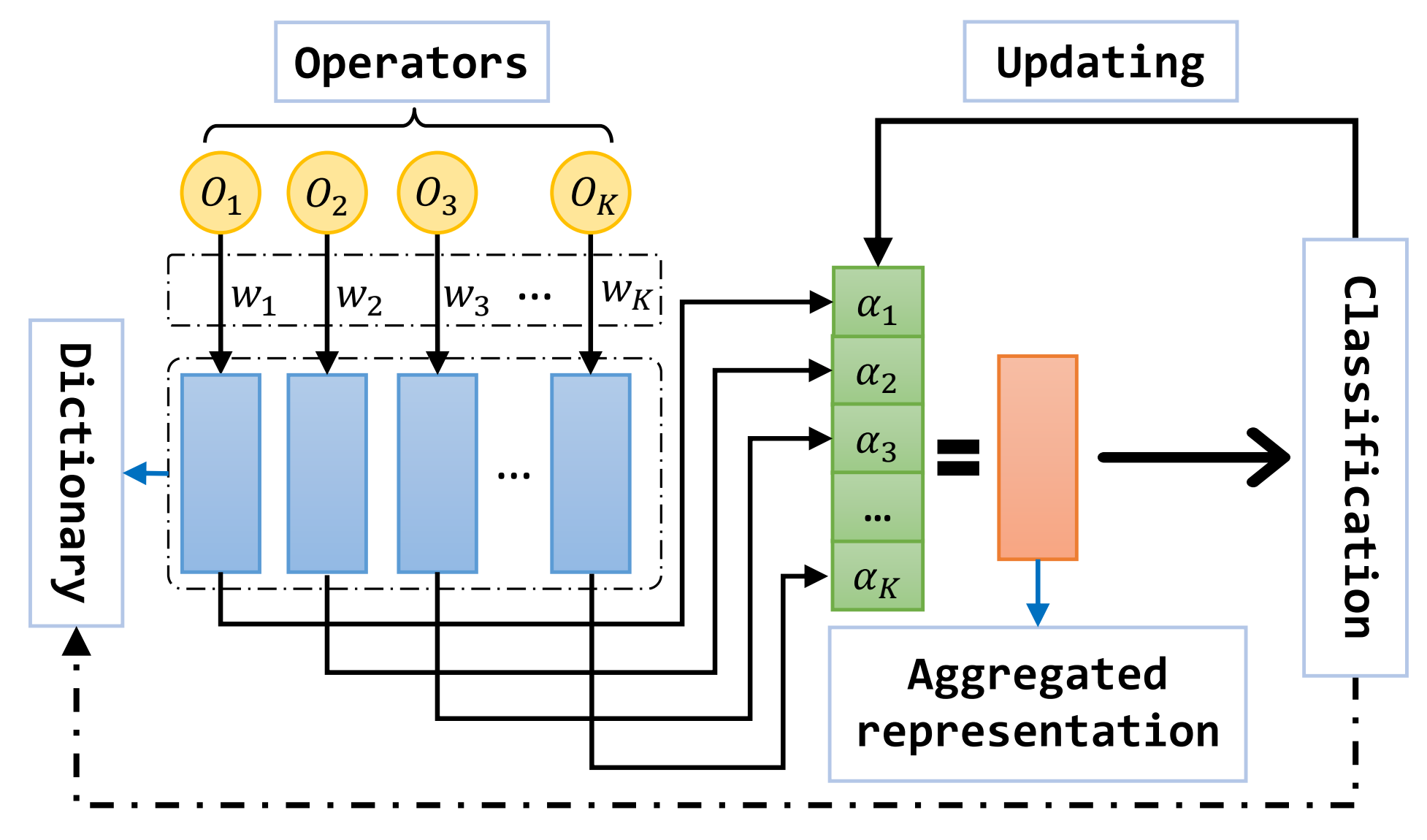

Figure 3. The framework of the proposed NAC in one layer.

$$\min_{\boldsymbol{\alpha}} \quad \mathcal{L}\left(\boldsymbol{y}, f(\mathbf{h}^L)\right) + \rho\|\boldsymbol{\alpha}\|_1, \text{where} \begin{cases} \mathbf{h}_v^l = \phi\left[\boldsymbol{W}^l \cdot \bar{o}^l\left(\left\{\mathbf{h}_u^{l-1}, \forall u \in N(v)\right\}\right)\right] \\ \bar{o}^l(x) = \boldsymbol{o}^l\hat{\boldsymbol{\alpha}}_l = \sum_{k=1}^K \frac{\alpha_{lk}}{||\boldsymbol{\alpha_l}||_2} o^{lk}(x) \\ \boldsymbol{o}^l = [o^{l1}(x), o^{l2}(x), \dots, o^{lK}(x)] \end{cases} \tag{2}$$

where ρ is the sparsity hyperparameter, $f(\cdot)$ is a linear function as the final output layer, e.g., MLP, $\mathbf{h}$ represents all nodes' embeddings.

Theoretical Findings

Analysis of No-update Scheme in GNNs

We investigate why an untrained GNN model can attain the same performance as the optimal one.

Analysis for No-update Scheme in GNNs

Assume $\boldsymbol{W}_l(0)$ is randomly initialized for all $l \in [1, L]$, if $\prod_{l=1}^L \boldsymbol{W}_l(0)$ is full rank, there must exist a weight matrix for the output layer, i.e., $\tilde{\boldsymbol{W}}_o$, that makes the final output the same as the one from a well-trained network:

$$\boldsymbol{A}^L\boldsymbol{X}\prod_{l=1}^{L}\boldsymbol{W}_l^*\boldsymbol{W}_o^* = \boldsymbol{A}^L\boldsymbol{X}\prod_{l=1}^{L}\boldsymbol{W}_l(0)\tilde{\boldsymbol{W}}_o. \tag{3}$$

Architecture Searching via Sparse Coding

We put forward the following corollary to show the natural connection between NAS in GNNs and Sparse Coding.

The Unified Format of Search Space in NAC

The search space of GNNs can be unified as $\sum_{k=0}^K \mathrm{P}^k(\boldsymbol{L})\boldsymbol{X}\boldsymbol{W}^k = \boldsymbol{D}\boldsymbol{W}$ when removing the activation function, where $\boldsymbol{D} = \|\{\mathrm{P}^k(\boldsymbol{L})\boldsymbol{X}\}$ is the fixed base, and $\boldsymbol{W} = \|\{\boldsymbol{W}^k\}^T$ is the trainable parameters. Here, $\|$ stands for concatenating a set of matrices horizontally.

Evaluation Results

Table 1. Experimental results on the compared methods: our NAC attains superior performance in both accuracy (%) and efficiency (in minutes).

	CiteSeer		Cora		PubMed		Computers	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
RS	$70.12_{\pm 2.36}$	14.4	$71.26_{\pm 4.68}$	30.6	$86.75_{\pm 0.82}$	187.8	$77.84_{\pm 1.35}$	8.75
BO	$70.95_{\pm 1.62}$	18	$68.59_{\pm 6.66}$	31.2	$87.42_{\pm 0.68}$	189.6	$77.46_{\pm 2.02}$	17.65
GraphNAS	$68.69_{\pm 1.30}$	253.8	$71.26_{\pm 4.90}$	245.4	$86.07_{\pm 0.51}$	1363.8	$73.97_{\pm 1.79}$	86.37
GraphNAS-WS	$65.35_{\pm 5.13}$	80.4	$72.14_{\pm 2.59}$	161.4	$85.71_{\pm 1.05}$	965.4	$72.99_{\pm 3.44}$	42.47
SANE	$71.84_{\pm 1.33}$	4.2	$84.58_{\pm 0.53}$	10.2	$87.55_{\pm 0.78}$	107.4	$90.70_{\pm 0.89}$	0.72
NAC	**74.62**$_{\pm 0.38}$	**1.2**	**87.41**$_{\pm 0.92}$	**1.2**	$88.04_{\pm 1.06}$	**9.0**	**91.64**$_{\pm 0.14}$	**0.23**
NAC-updating	$74.17_{\pm 1.18}$	4.2	$86.62_{\pm 1.14}$	3.6	**88.10**$_{\pm 0.86}$	25.8	$90.89_{\pm 1.10}$	0.70

Observations

- **Accuracy**: NAC beats all baselines and attains up to 18.8% improvement over the Bayesian method.
- **Efficiency**: NAC achieves superior performance up to 200× time faster than GraphNAS.

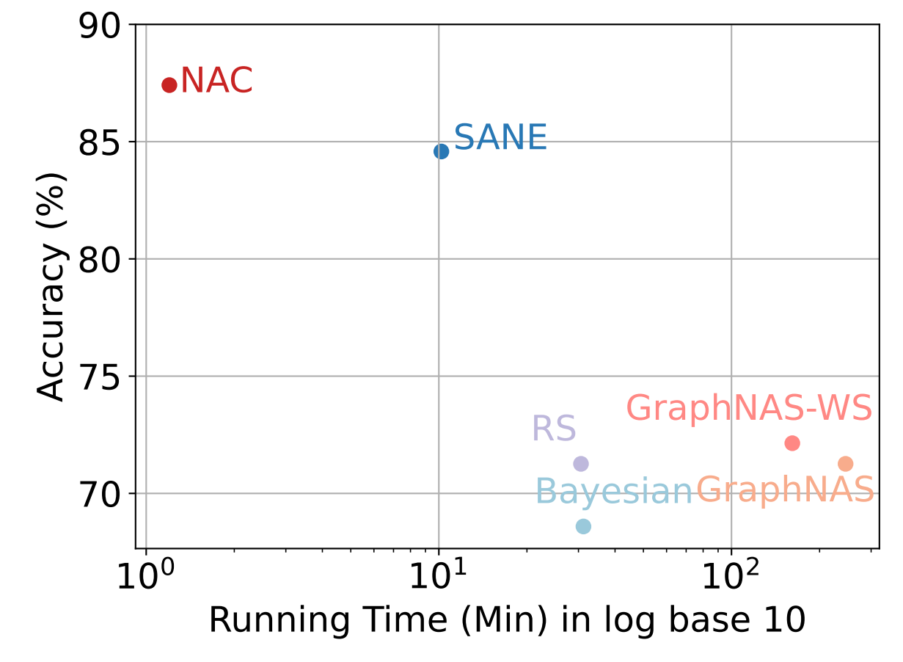

Figure 4. Performance on Cora.

No-update Scheme at Work. To further validate our no-update scheme, we evaluate its effect on other weight-sharing methods, which implies that we can improve the performance of NAS-GNN methods by simply fixing the weights.

Table 2. Comparison between SANE and SANE* (w/o. weight updates).

	CiteSeer	Cora	Pubmed	Computers
	Acc(%)	Acc(%)	Acc(%)	Acc(%)
SANE	$71.84_{\pm 1.33}$	$84.58_{\pm 0.53}$	$87.55_{\pm 0.78}$	$90.70_{\pm 0.89}$
SANE*	**71.95**$_{\pm 1.32}$	**85.46**$_{\pm 0.76}$	**88.12**$_{\pm 0.35}$	**90.86**$_{\pm 0.80}$

Analysis of Convergence. A notable benefit of the NAC framework is its guaranteed convergence from the sparse coding perspective. **NAC converges much faster than SNAE in only around 20 epochs.**

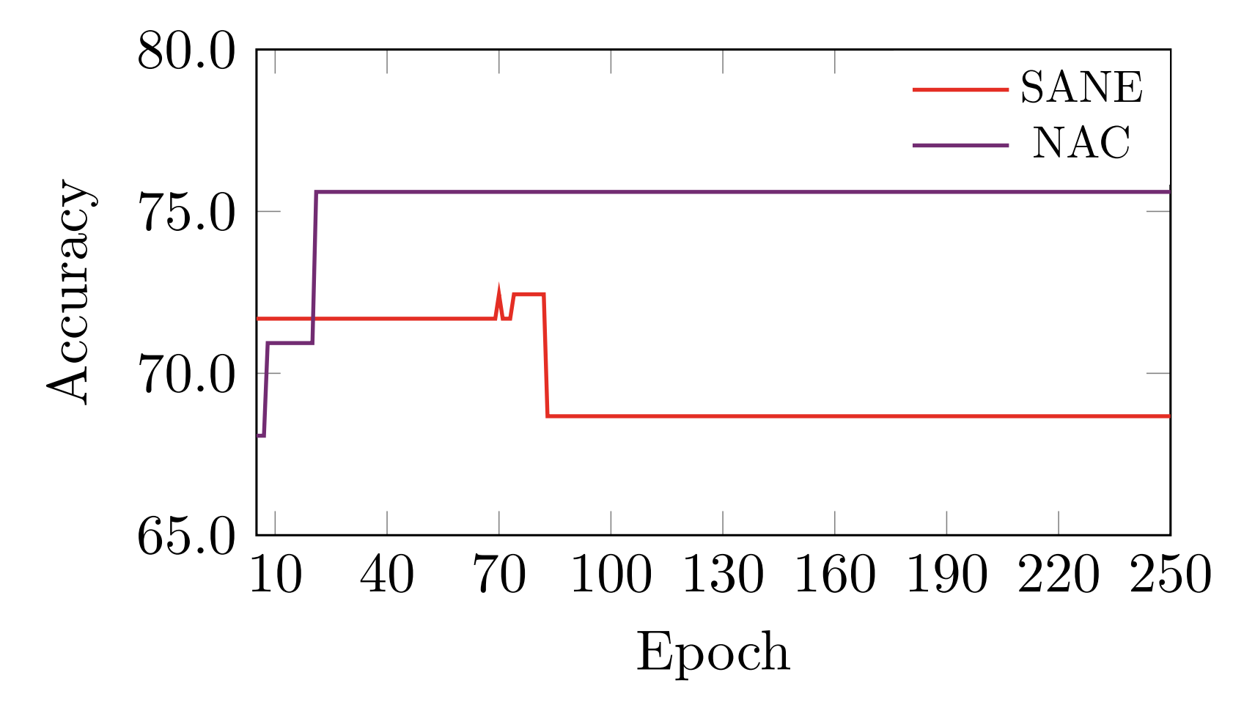

Figure 5. Convergence for SANE and NAC in terms of accuracy on Pubmed dataset.

References

[1] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *Proc. IJCAI*, 2020.

[2] ZHAO Huan, YAO Quanming, and TU Weiwei. Search to aggregate neighborhood for graph neural network. In *Proc. ICDE*, pages 552–563, 2021.

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2017.