香港中文大學
The Chinese University of Hong Kong

The Introduction of Neural Architecture Search in Graph Neural Networks

Peng XU

Department of Computer Science and Engineering
Chinese University of Hong Kong
19s051059@stu.hit.edu.cn

March 26, 2022

Outline

1 Background

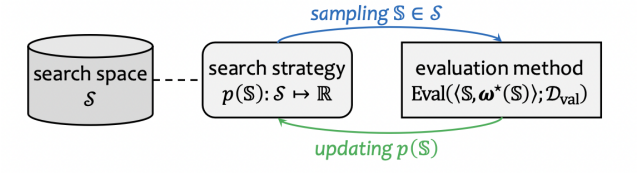
2 Related Works
2.1 Search Space
2.2 Search Strategy
2.3 Performance Estimation Strategy

Background

NAS background

Neural architecture search (NAS) is a technique for automating the design of artificial neural networks. NAS generally includes three major parts [11]:

- Search Space: The search space define the type(s) of ANN that can be designed and optimized.
- Search Strategy: The search strategy defines the approach used to explore the search space.
- Performance Estimation Strategy: The performance estimation strategy evaluates the performance of a possible ANN from its design.

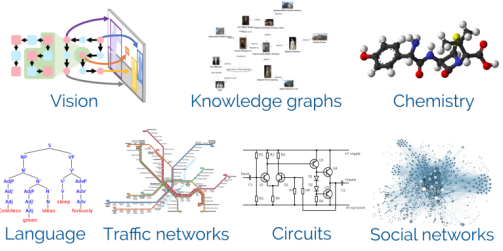


The three components of NAS.

GNN background

Graph Neural Network (GNN) is a very hot topic in recent years [10].

- Representation learning in graphs
- Define "convolution" on graph(non-grid) data
- Powerful expression capability in dealing with graph structure data
- Applications of Graph Neural Network:
  - Recommendation
  - Fraud Detection
  - Spam detection
  - Bioinformatics
  - Netlist Representation
  - ...



Typical Graph Structural data.

A brief history of GNNs

In the early age, GNNs are motivated from the spectral perspective

- Spectral GNN [1] that applies the Laplacian operators directly.
- ChebNet [2] approximated these operators the using summation instead to avoid a high computational cost.

A brief history of GNNs

Message-passing mechanism of modern GNNs:

- GCN [5] further simplifies ChebNet by using its first order, and reaches the balance between efficiency and effectiveness
- GraphSAGE [4] concatenates nodes features with mean/max/LSTM pooled neighbouring information.
- GAT [9] aggregates neighbour information using learnable attention weights.
- GIN [13] converts the aggregation as a learnable function based on the Weisfeiler-Lehman test instead of prefixed ones as other GNNs, aiming to maximise the power of GNNs.

Motivation

- A direct question
  - Can we obtain data-specific GNN architectures?
- Neural Architecture Search(NAS)
  - Automatically design SOTA architectures for CNN. =>
  - Automatically design SOTA architectures for GNN.
- NAS for GNNs
  - Obtain data-specific GNN architectures.
  - Automatically search for unexplored architectures beyond human-designed ones.
  - Provides an automated process to design graph neural network structures for different data.

Related Works

The Outline of Related Works

1. Search Space
   - Micro Search Space
   - Macro Search Space
2. Search Strategy
   - Controller + Reinforce Learning
   - Evolutionary Algorithms
   - Differentiable
3. Performance Estimation Strategy
   - Weight Sharing

Search Space

Micro Search Space

The micro search space defines how nodes exchange messages with others in each layer.

$$\mathbf{m}_i^{(l)} = \text{AGG}^{(l)} \left(\left\{ a_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_i^{(l)}, \forall j \in \mathcal{N}(i) \right\} \right) \tag{1}$$

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\text{COMBINE}^{(l)} \left[\mathbf{m}_i^{(l)}, \mathbf{h}_i^{(l)} \right] \right) \tag{2}$$

where $\mathbf{h}_i$ denotes the node representation of node v_i in the l-th layer, $m^(l)$ is the message for node v_i, $AGG(\cdot)$ is the aggregation function, $a_{ij}^{(l)}$ denotes the weights from node v_j to node v_i, $COMBINE^{(l)}(\cdot)$ is the combining function for combining the node representation $\mathbf{h}_i$ and the message $m^(l)$ for v_i, $\mathbf{W}^{(l)}$ are learnable weights, and $\sigma(\dot{})$ is an activation function.

Micro Search Space

Gao et al.[3] and Zhou et al.[16] adopted micro search spaces compose the following components:

- Aggregation function $AGG(\cdot)$: SUM, MEAN, MAX, MLP
- Aggregation weights a_{ij}: CONST, GCN, GAT, SYMGAT, COS, LINEAR, GENELINEAR
- Number of heads when using attentions: $1, 2, 4, 6, 8, 16, \ldots$
- Combining function $COMBINE(\cdot)$: CONCAT, ADD, MLP;
- Dimensionality of $\mathbf{h}^{(l)}$: $8, 16, 32, 64, 128, 256, 512, \ldots$
- Non-linear activation function $\dot{\sigma}()$: Sigmoid, Tanh, ReLU, Identity, Softplus, LeakyReLU, ReLU6, and ELU;

Attention Mechanisms	Equations
CONSTANT	1
GCN	$\frac{1}{\sqrt{\lvert N(i)\rvert \lvert N(j)\rvert}}$
GAT	$\text{LeakyReLU}(\vec{a}(W^{(k)}x_i^{(k-1)} \lvert\rvert W^{(k)}x_j^{(k-1)}))$
SYM-GAT	$a_{ij}^{(k)} + a_{ji}^{(k)}$ based on GAT
COS	$\vec{a}(W^{(k)}x_i^{(k-1)} \lvert\rvert W^{(k)}x_j^{(k-1)})$
LINEAR	$\tanh(\vec{a}_l W^{(k)}x_i^{(k-1)} + \vec{a}_r W^{(k)}x_j^{(k-1)})$
GERE-LINEAR	$W_G \tanh(W^{(k)}x_i^{(k-1)} + W^{(k)}x_i^{(k-1)})$

Attention functions

Macro Search Space

Similar to residual connections and dense connections in CNNs, node representations in one layer of GNNs do not necessarily solely depend on the immediate previous layer.
These connectivity patterns between layers form the macro search space. Zhao et al.[15] formulated such designs as:

$$\mathbf{H}^{(l)} = \sum_{j<l} \mathcal{F}_{jl}\left(\mathbf{H}^{(I)}\right) \tag{3}$$

where $\mathcal{F}_{jl}$ can be the message-passing layer:

- ZERO (i.e., not connecting)
- IDENTITY (e.g., residual connections)[1]
- MLP

[1]Since the dimensionality of $H^{(j)}$ can vary, IDENTITY can only be adopted if the dimensionality of each layer matches.

Search Strategy

Controller + RL

A widely adopted NAS search strategy uses a controller to generate the neural architecture descriptions and train the controller with reinforcement learning to maximize the model performance as rewards[17].

- Reward

$$\mathcal{J}(\theta_c) = E_{P(a_{1:T};\theta_c)}[R] \tag{4}$$

- Policy gradient

$$\nabla \mathcal{J}(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R] \tag{5}$$

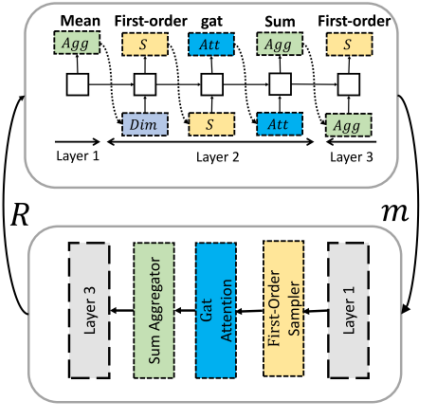
- Monte Carlo method approximation

$$\frac{1}{m} \sum_{k=1}^{M} \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R] \tag{6}$$

GraphNAS

GraphNAS [3] enables automatic search of the best graph neural architecture based on reinforcement learning.

- Use a recurrent network to generate variable-length strings that describe the architectures of graph neural networks
- Train the recurrent network with. reinforcement learning to maximize the expected accuracy of the generated architectures.



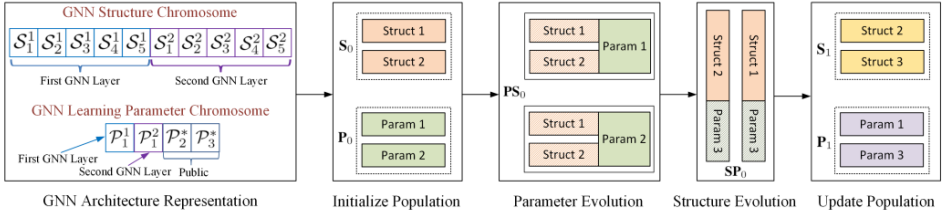
The framework of GraphNAS.

Evolutionary Algorithms

Evolutionary algorithms are a class of optimization algorithms inspired by biological evolution.For NAS,[7]

1. Randomly generated architectures are considered initial individuals in a population.
2. Then, new architectures are generated using mutations and crossover operaions on the population.
3. The architectures are evaluated and selected to form the new population, and the same process is repeated.
4. The best architectures are recorded while updating the population, and the final solutions are obtained after sufficient updating steps.

Genetic-GNN[8] proposes an co-evolution process to alternatively update the GNN architecture and the learning hyper-parameters to find the best fit of each other.



The framework of Genetic-GNN.

Differentiable NAS methods such as DARTS [6] have gained popularity in recent years. Instead of optimizing different operations separately, differentiable methods construct a single supernetwork (known as the one-shot model) containing all possible operations.

$$\bar{o}^{(i,j)} = \sum_{o \in \mathcal{O}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'}^{(i,j)})} o(x) \tag{7}$$

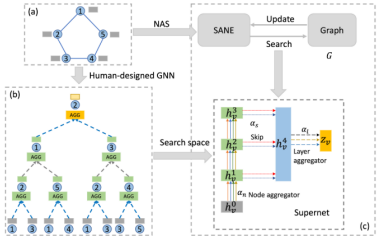
DARTS adopted a bi-level optimization framework to optimize the weight and architecture parameter:

$$\min_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{\omega}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \tag{8}$$

$$s.t. \boldsymbol{\omega}^*(\boldsymbol{\alpha}) = \mathbf{argmin}_{\boldsymbol{\omega}} \mathcal{L}_{train}(\boldsymbol{\omega}^*, \boldsymbol{\alpha}) \tag{9}$$

Search to aggregate neighborhood for graph neural networks (SANE)[15] applied differentiable search algorithm to obtain data-specific GNN architectures and adopted a gradient-based approximation to update the architecture parameters, which is more computationally efficient than previous reinforcement learning based methods.

$$\nabla_\alpha \mathcal{L}_{val}(\boldsymbol{\omega}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \nabla_\alpha \mathcal{L}_{val}(\boldsymbol{\omega} - \xi \nabla_\omega \mathcal{L}_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha}), \boldsymbol{\alpha}) \tag{10}$$



The Search Space of SANE.

Algorithm 1 SANE - Search to Aggregate NEighborhood.

Require: The search space $\mathcal{A}$, the number of top architectures k, the epochs T for search.
Ensure: The k searched architectures $\mathcal{A}_k$.

```
while t = 1, ..., T do
    Compute the validation loss L_val;
    Update α_n, α_s and α_l by gradient descend rule (8) with (3), (4) and (5) respectively;
    Compute the training loss L_tra;
    Update weights w by descending ∇_w L_tra(w, α) with the architecture α = {α_n, α_s, α_l};
end while
Derive the final architecture {α_n*, α_s*, α_l*} based on the trained {α_n, α_s, α_l};
return Searched {α_n*, α_s*, α_l*}.
```

The optimization framework of SANE.

Performance Estimation Strategy

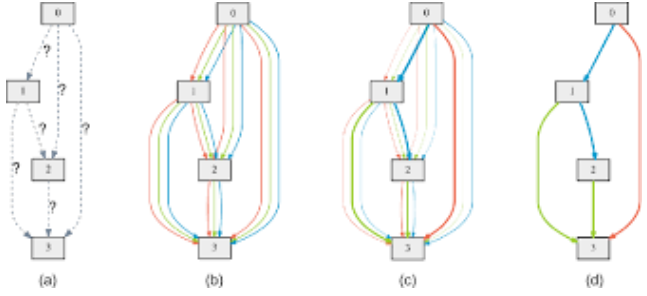
Performance Estimation Strategy

A strategy successfully applied to CNNs is sharing weights among different models, known as weight sharing [12].

- One-shot model
- Inherit weights

One-shot model

For differentiable NAS with a large one-shot model, parameter sharing is naturally achieved since the architectures and weights are jointly trained in one model.



One-shot model in differentiable NAS.

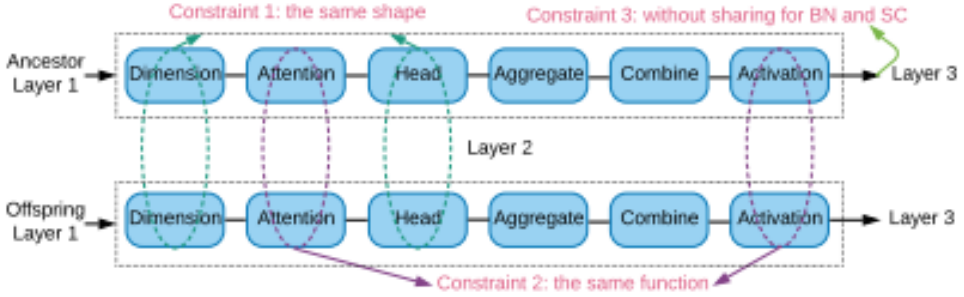
Inherit weights

For NAS without a one-shot model, sharing weights among different architecture is more difficult but not entirely impossible. For example, inheriting weights from previous architectures is feasible and reasonable in CNNs[7]. [2]

[2]However, since there is still a lack of understandings of what weights in GNNs represent, we need to be more cautious about inheriting weights[14].

Inherit weights

AutoGNN[16] proposes three constraints for parameter inheritance: same weight shapes, same attention and activation functions, and no parameter sharing in batch normalization and skip connections.



Inherit weights

References I

[1] Joan Bruna et al. “Spectral Networks and Locally Connected Networks on Graphs”. In: *ICLR*. 2014.

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *NeurIPS*. 2016.

[3] Yang Gao et al. “Graph Neural Architecture Search.”. In: *IJCAI*. 2020.

[4] William L. Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *NeurIPS*. 2017.

[5] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *ICLR*. 2017.

[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. “DARTS: Differentiable Architecture Search”. In: *ICLR*. 2019.

[7] Esteban Real et al. “Large-Scale Evolution of Image Classifiers”. In: *ICML*. 2017.

References II

[8] Min Shi et al. “Evolutionary architecture search for graph neural networks”. In: *arXiv preprint arXiv:2009.10199* (2020).

[9] Petar Velickovic et al. “Graph Attention Networks”. In: *ICLR*. 2018.

[10] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021), pp. 4–24. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.2978386. URL: http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[11] Lingxi Xie and Alan L. Yuille. “Genetic CNN”. In: *ICCV*. 2017.

[12] Lingxi Xie et al. “Weight-Sharing Neural Architecture Search: A Battle to Shrink the Optimization Gap”. In: *ACM Comput. Surv.* 54.9 (2022), 183:1–183:37.

[13] Keyulu Xu et al. “How Powerful are Graph Neural Networks?” In: *ICML*. 2019.

[14] Huan Zhao, Lanning Wei, Zhiqiang He, et al. “Efficient graph neural architecture search”. In: (2020).

References III

[15] Huan Zhao, Quanming Yao, and Weiwei Tu. “Search to aggregate neighborhood for graph neural network”. In: *ICDE*. 2021.

[16] Kaixiong Zhou et al. “Auto-GNN: Neural Architecture Search of Graph Neural Networks”. In: *CoRR* abs/1909.03184 (2019).

[17] Barret Zoph and Quoc V. Le. *Neural Architecture Search with Reinforcement Learning*. 2017. arXiv: 1611.01578 [cs.LG].

THANK YOU!