Automated Learning from Graph-Structured Data

Quanming Yao[1,2], Huan Zhao[2], Yongqi Zhang[2]

[1]*Department of Electronic Engineering, Tsinghua University*

[2]*4Paradigm Inc.*

https://quanmingyao.github.io/AutoML.github.io/aaai22-tutorial.html



Schedule at a Glance

Time	Event
00-45 minutes	**Part 1**: Automated Graph Neural Network
	Speaker: Huan Zhao
45-90 minutes	**Part 2**: Hyper-parameter Tuning for Graph-Structured Data
	Speaker: Quanming Yao
90-135 minutes	**Part 3**: Automated Knowledge Graph Reasoning: from Triplets to Subgraphs
	Speaker: Yongqi Zhang
135-150 minutes	**Part 4**: Discussion

Code Repo.

Automated Learning from Graph-Structured Data

Part 1: Automated Graph Neural Network

Dr. Huan Zhao

Senior researcher, 4Paradigm

zhaohuan@4paradigm.com

Feb. 20th 2022

Outline

- Background
  - Graph neural network (GNN)
  - Neural architecture search (NAS)
- Graph neural architecture search
  - Node-level
  - Graph-level
  - GNN topology design
- Conclusion

Graph-based applications

Search Engine

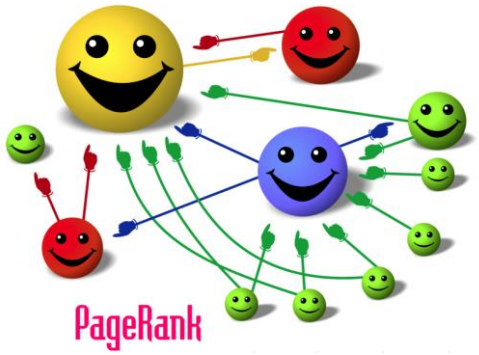


Image Credit: Wikipedia

Recommendation

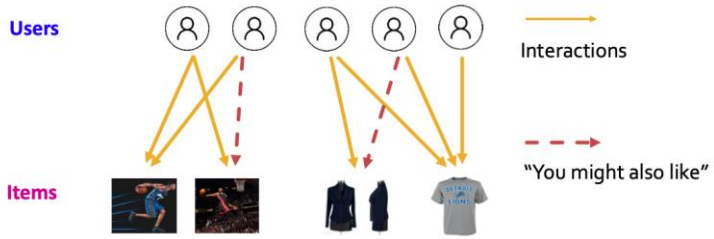


Image Credit: Jure Stanford CS224

Social Network

Image Credit: Shreyansh@Medium

Biomedicine

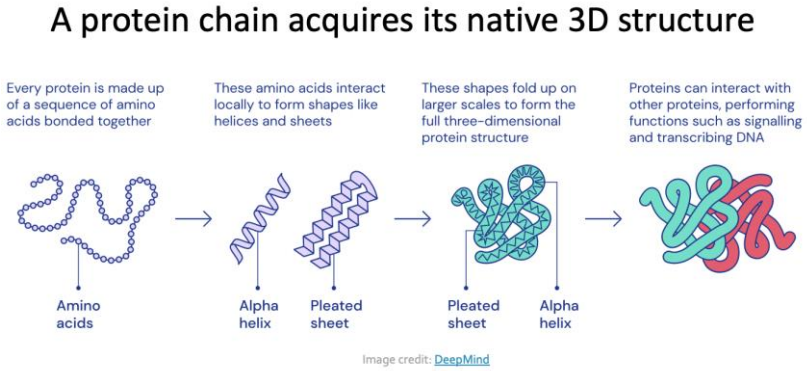


Image Credit: Jure Stanford CS224

Traffic prediction

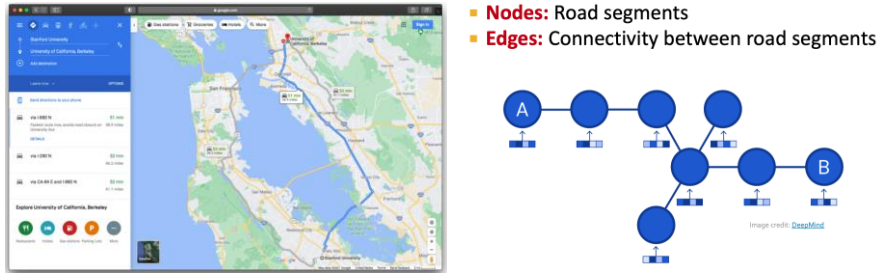


Image Credit: Jure Stanford CS224

Chip Design

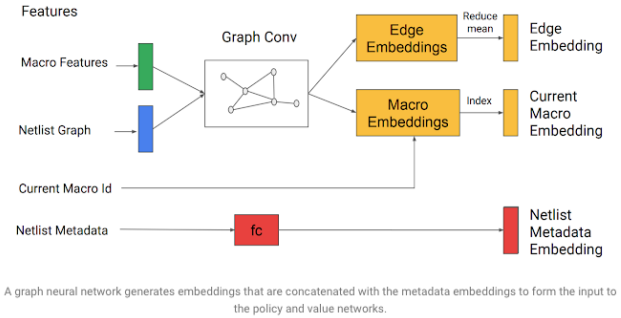
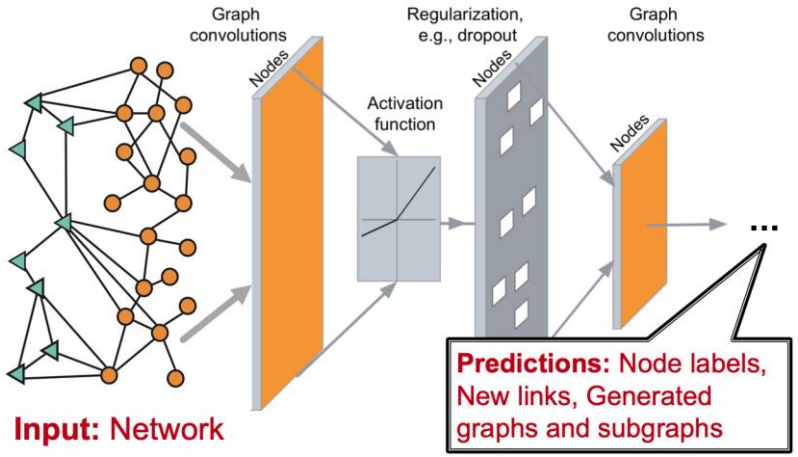


Image Credit: Google Blog

Graph Neural Networks (GNN)

- GNNs have been new state-of-the-art (SOTA) for graph-based tasks.
  - Node-level
  - Edge-level
  - Subgraph-level
  - Graph-level



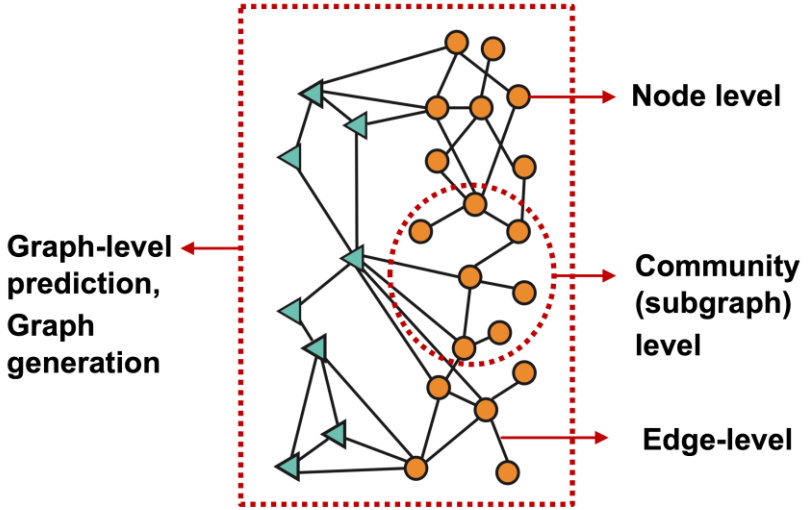


Image Credit: Jure Stanford CS224

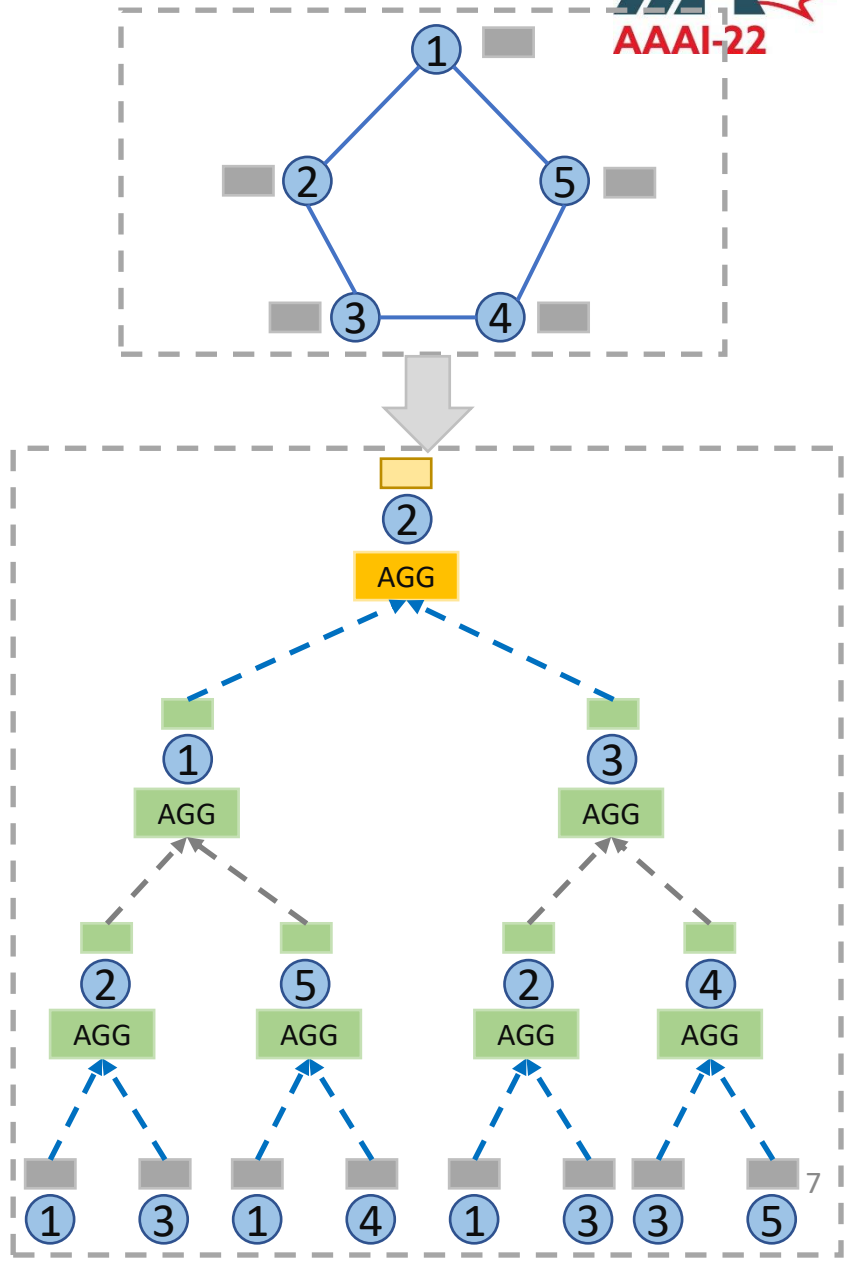
GNN

- Message passing framework
  - Node embedding updated by neighbors
  - K-layer GNN access K-hop neighbors
  - "Neighborhood aggregation"

Self contained Neighborhood

$$\mathbf{h}_v^l = \sigma\left(\mathbf{W}^{(l)} \cdot \mathrm{AGG}_{\mathrm{node}}\left(\{\mathbf{h}_u^{(l-1)}, \forall u \in \widetilde{N}(v)\}\right)\right)$$

- Variants of GNN
  - GCN: normalized sum aggregator
  - GraphSAGE: MEAN, MAX, SUM, LSTM
  - GAT: Attention aggregator
  - GIN: Multi-Layer Perceptrons (MLP)



Design Space for GNN [You et al. 2020]

- The performance of GNN models vary
  - Combinations of 12 key design dimensions.
  - 315,000 instances

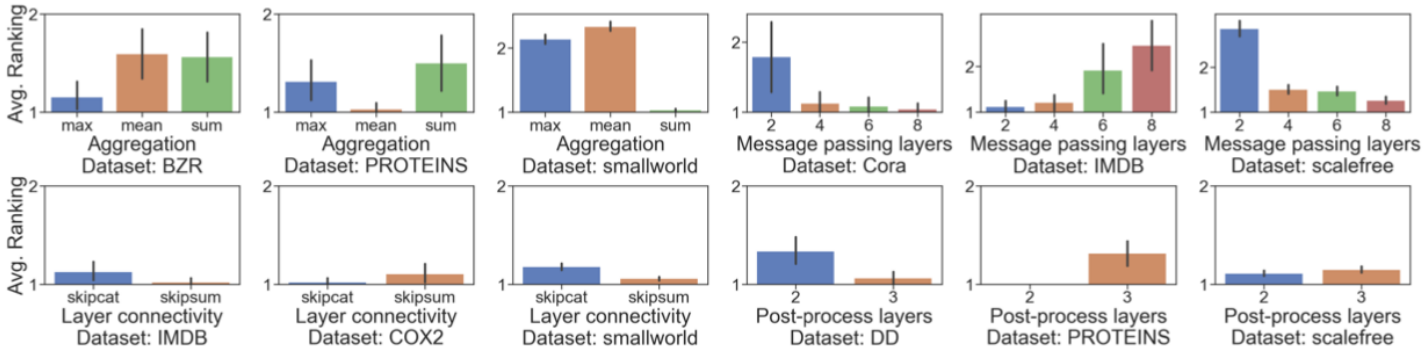
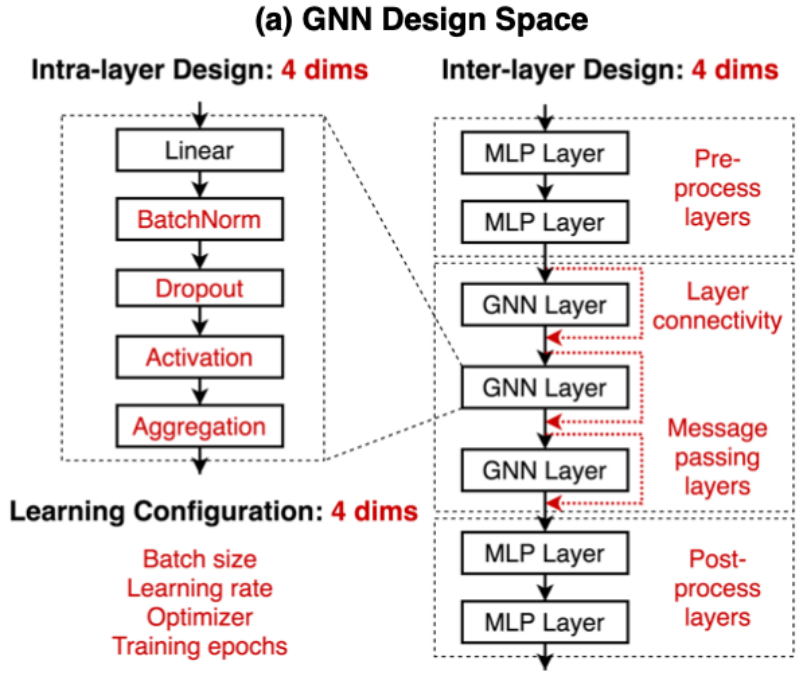


Figure 4: **Ranking analysis for GNN design choices over different GNN tasks**. Lower is better. Preferable design choices greatly vary across GNN tasks.



Batch Normalization	Dropout	Activation	Aggregation
True, False	False, 0.3, 0.6	ReLU, PReLU, Swish	Mean, Max, Sum

Layer connectivity	Pre-process layers	Message passing layers	Post-precess layers
Stack, Skip-Sum, Skip-Cat	1, 2, 3	2, 4, 6, 8	1, 2, 3

Batch size	Learning rate	Optimizer	Training epochs
16, 32, 64	0.1, 0.01, 0.001	SGD, Adam	100, 200, 400

You et al. Design Space for Graph Neural Networks. NeurIPS 2020

Challenges of GNN architecture design

- Different types of graphs
  - Homogeneous graph [1]
  - Heterogeneous graph [2]
  - Hypergraph [3]
  - Dynamic graph [4]

- Diverse applications
  - Recommender system [5]
  - Fraud detection [6]
  - Traffic prediction [7]
  - Text classification [8]
  - Drug discovery [9]

SOTA architectures vary

[1] How powerful are graph neural networks?. ICLR 2019

[2] DiffMG: Differentiable Meta Graph Search for Heterogeneous Graph Neural Networks. KDD 2021

[3] Hypergraph neural networks. AAAI 2019

[4] Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. AAAI 2019

[5] Graph convolutional neural networks for web-scale recommender systems. KDD 2018

[6] Alleviating the inconsistency problem of applying graph neural network to fraud detection. SIGIR 2020

[7] ETA Prediction with Graph Neural Networks in Google Maps. CIKM 2021

[8] Graph convolutional networks for text classification. AAAI 2019

[9] Learning graph-level representation for drug discovery. Arxiv 2018

Challenges of GNN architecture design

- Graph of different properties
  - Homophily: Connected nodes tend to have the same class labels and similar features
  - Heterophily: Connected nodes tend to have different class labels and dissimilar features

$$\textit{homophily ratio } h = \frac{|\{(u,v):(u,v)\in\mathcal{E}\wedge y_u=y_v\}|}{|\mathcal{E}|}$$

Heterophily ⟵ ⟶ Homophily

Table 5: Real data: mean accuracy ± stdev over different data splits. Best model per benchmark highlighted in gray. The "*" results are obtained from [26] and "N/A" denotes non-reported results.

Hom. ratio h	Texas 0.11	Wisconsin 0.21	Actor 0.22	Squirrel 0.22	Chameleon 0.23	Cornell 0.3	Cora Full 0.57	Citeseer 0.74	Pubmed 0.8	Cora 0.81
#Nodes $\lvert\mathcal{V}\rvert$	183	251	7,600	5,201	2,277	183	19,793	3,327	19,717	2,708
#Edges $\lvert\mathcal{E}\rvert$	295	466	26,752	198,493	31,421	280	63,421	4,676	44,327	5,278
#Classes $\lvert\mathcal{Y}\rvert$	5	5	5	5	5	5	70	7	3	6
H_2GCN-1	84.86±6.77	86.67±4.69	35.86±1.03	36.42±1.89	57.11±1.58	82.16±4.80	68.13±0.49	77.07±1.64	89.40±0.34	86.92±1.37
H_2GCN-2	82.16±5.28	85.88±4.22	35.62±1.30	37.90±2.02	59.39±1.98	82.16±6.00	69.05±0.37	76.88±1.77	89.59±0.33	87.81±1.35
GraphSAGE	82.43±6.14	81.18±5.56	34.23±0.99	41.61±0.74	58.73±1.68	75.95±5.01	65.14±0.75	76.04±1.30	88.45±0.50	86.90±1.04
GCN-Cheby	77.30±4.07	79.41±4.46	34.11±1.09	43.86±1.64	55.24±2.76	74.32±7.46	67.41±0.69	75.82±1.53	88.72±0.55	86.76±0.95
MixHop	77.84±7.73	75.88±4.90	32.22±2.34	43.80±1.48	60.50±2.53	73.51±6.34	65.59±0.34	76.26±1.33	85.31±0.61	87.61±0.85
GraphSAGE+JK	83.78±2.21	81.96±4.96	34.28±1.01	40.85±1.29	58.11±1.97	75.68±4.03	65.31±0.58	76.05±1.37	88.34±0.62	85.96±0.83
Cheby+JK	78.38±6.37	82.55±4.57	35.14±1.37	45.03±1.73	63.79±2.27	74.59±7.87	66.87±0.29	74.98±1.18	89.07±0.30	85.49±1.27
GCN+JK	66.49±6.64	74.31±6.43	34.18±0.85	40.45±1.61	63.42±2.00	64.59±8.68	66.72±0.61	74.51±1.75	88.41±0.45	85.79±0.92
GCN	59.46±5.25	59.80±6.99	30.26±0.79	36.89±1.34	59.82±2.58	57.03±4.67	68.39±0.32	76.68±1.64	87.38±0.66	87.28±1.26
GAT	58.38±4.45	55.29±8.71	26.28±1.73	30.62±2.11	54.69±1.95	58.92±3.32	59.81±0.92	75.46±1.72	84.68±0.44	82.68±1.80
GEOM-GCN*	67.57	64.12	31.63	38.14	60.90	60.81	N/A	77.99	90.05	85.27
MLP	81.89±4.78	85.29±3.61	35.76±0.98	29.68±1.81	46.36±2.52	81.08±6.37	58.76±0.50	72.41±2.18	86.65±0.35	74.75±2.22

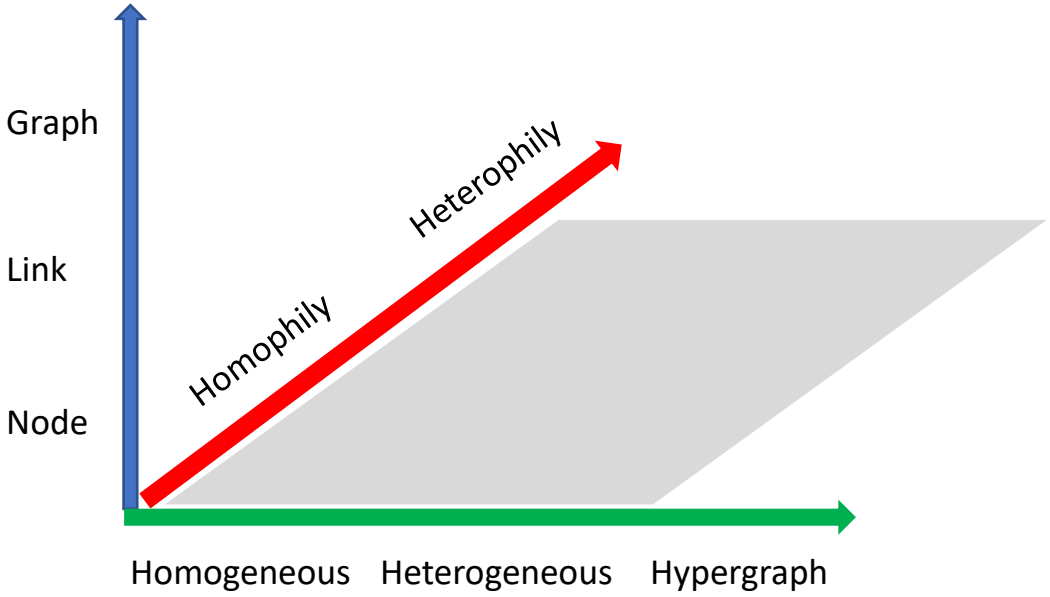
Simple MLP is even better than GNN

Zhu et al. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. NeurIPS 2020

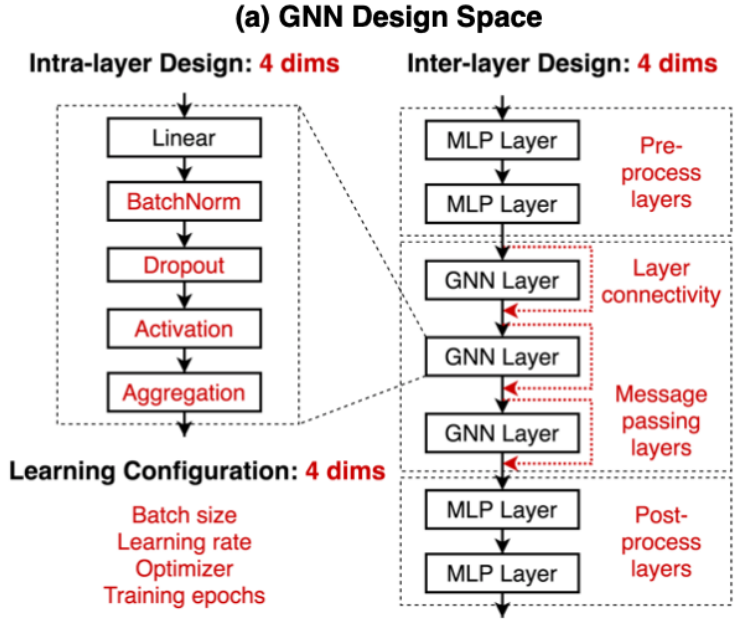
Challenges of GNN architecture design

- Due to the diversity and complexity of graph-based tasks, it makes a challenging problem to design effective GNN given a graph.

- It motivates the exploration of neural architecture search for GNN.



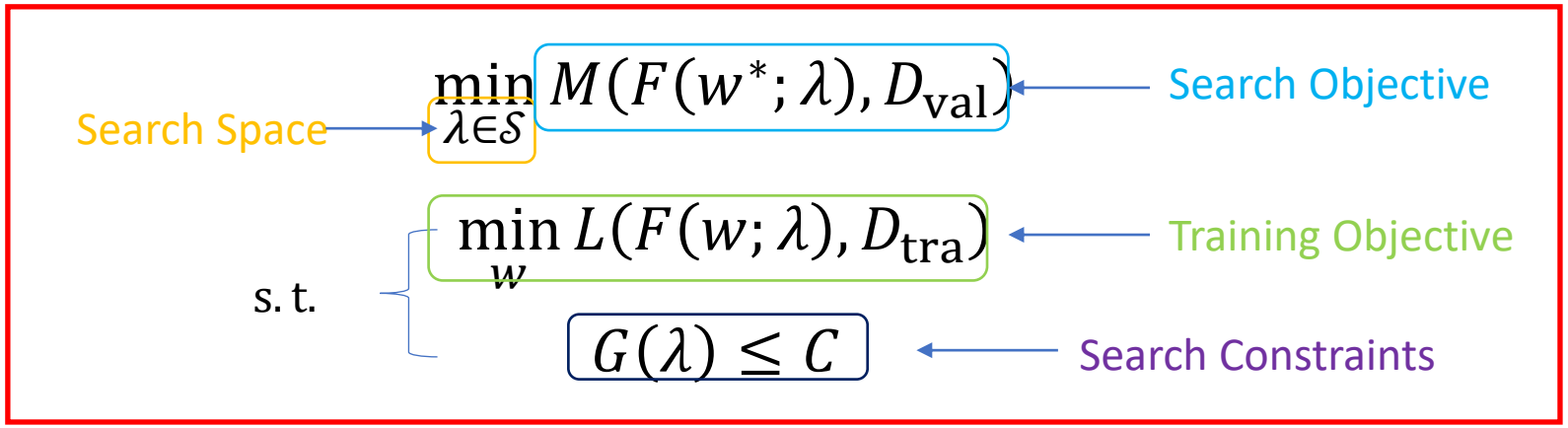
Challenging



Neural architecture search for graph neural network

Neural Architecture Search

- Explore the possibility of automatically searching for unexplored architectures beyond human-designed ones.
- A bi-level optimization problem
  - Derive a search space based on domain knowledge
  - Search objective is usually validation performance
  - Training objective usually comes from classical learning models
  - Search constraint is usually resource budgets

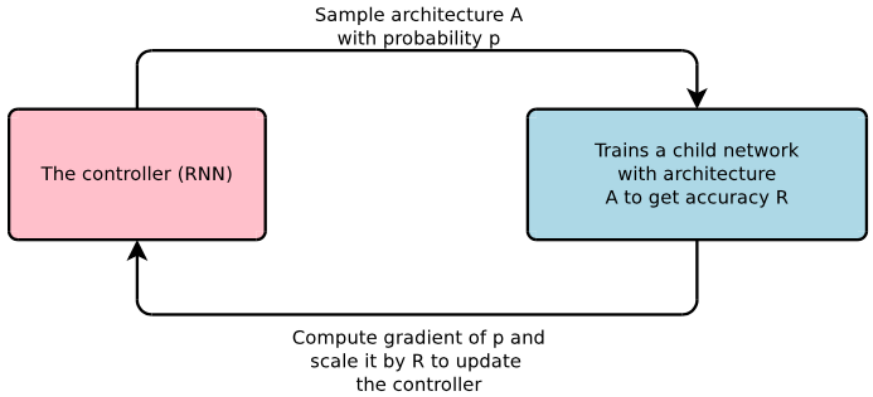


Zoph et al. Neural Architecture Search with Reinforcement Learning. ICLR 2017

Two representative approaches

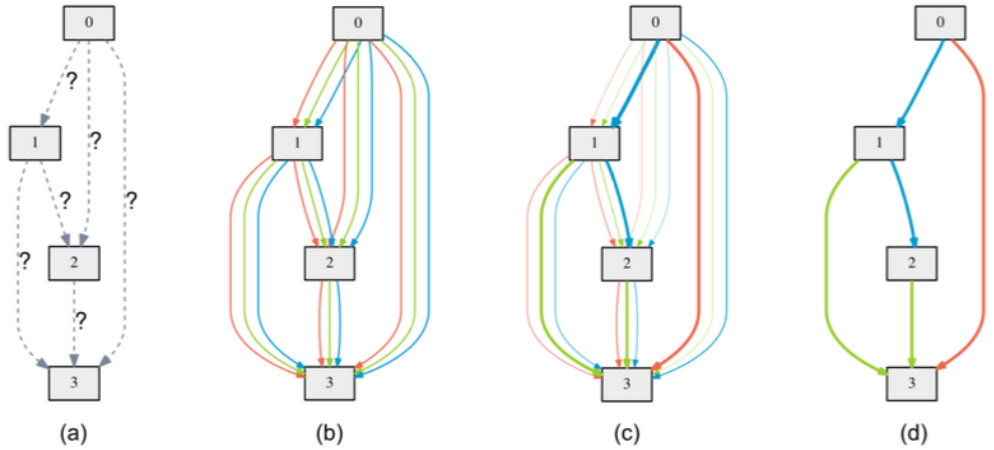
- Stand alone (Reinforcement learning)
  - An RNN controller to sample a candidate architecture
  - Train till convergency
  - Update the RNN controller by the validation accuracy

NASNet [Zoph et al. 2017]



- One-shot (Differentiable architecture search)
  - Train a supernet including all candidate architectures
  - Derive a chidnet from the supernet as the search architecture

DARTS [Liu et al. 2017]

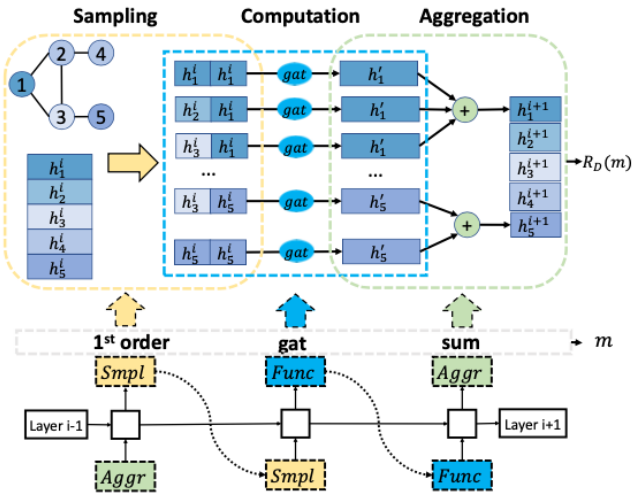


Zoph et al. Neural Architecture Search with Reinforcement Learning. ICLR 2017
Liu et al. Darts: Differentiable architecture search. ICLR 2019

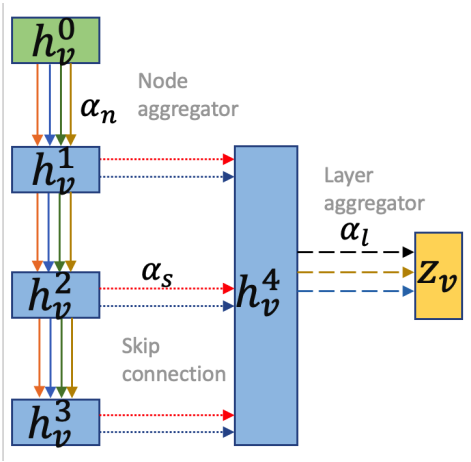
NAS for GNN

- Two representative approaches
  - Search for task-specific GNN architectures.
- GraphNAS [Gao et al. 2020]
  - Reinforcement learning based method
- SANE [Zhao et al. 2021]
  - Differentiable GNN architecture search

GraphNAS [Gao et al. 2020]



SANE [Zhao et al. 2021]



Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020
Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Representative works

- In this tutorial, we will introduce representative works
  - Node-level
    - GraphNAS [Gao et al. 2020]
      - Reinforcement learning based method
    - SANE [Zhao et al. 2021]
      - Differentiable GNN architecture search
  - Graph-level
    - PAS [Wei et al. 2021]
      - Differentiable pooling architecture search
  - GNN topology design
    - F2GNN [Wei et al. 2022]

Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020
Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021
Wei et al. Pooling Architecture Search for Graph Classification. CIKM 2021
Wei et al. Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective. The WebConf 2022
Ding et al. DiffMG: Differentiable Meta Graph Search for Heterogeneous Graph Neural Networks. KDD 2021

GraphNAS [Gao et al. 2020]

- A first graph neural architecture search.
  - Search for combination of GNN layers
- A customized search space and RL based search algorithm.
- SOTA performance in various settings.
  - Semi-supervised
  - Supervised

Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

GraphNAS [Gao et al. 2020]

- Operations in the search space
  - Neighbor sampling $Smpl$
  - Message computation $Func$
  - Message aggregation $Aggr$
  - Multi-head and readout $Read$
  - Activation function σ
  - Number of multi-head k
  - Output dimension d

- An exemplar GNN layer

$$[first\_order, gat, sum, concat, 8, 16, elu]$$

Operators	Values
$Smpl$	$first\_order$
$Func$	$e_{uv}h_u$
$Aggr$	sum, $mean$, max, mlp
$Read$	avg, for the last layer $concat$, otherwise
activate function σ	$sigmoid$, $tanh$, $relu$, $identity$, $softplus$, $leaky\_relu$, $relu6$, elu
multi-head k	1, 2, 4, 6, 8, 16
output dimension d	8, 16, 32, 64, 128, 256, 512

Table 1: Operators of search space $\mathcal{M}$

e_{uv}	Values
const	$e_{uv}^{con} = 1$
gcn	$e_{uv}^{gcn} = 1/\sqrt{d_u d_v}$
gat	$e_{uv}^{gat} = leaky\_relu((W_l * h_u + W_r * h_v))$
sym-gat	$e_{uv}^{sym} = e_{vu}^{gat} + e_{uv}^{gat}$
cos	$e_{uv}^{cos} =< W_l * h_u, W_r * h_v >$
linear	$e_{uv}^{lin} = tanh(sum(W_l * h_u))$
gene-linear	$e_{uv}^{gan} = W_a * tanh(W_l * h_u + W_r * h_v)$

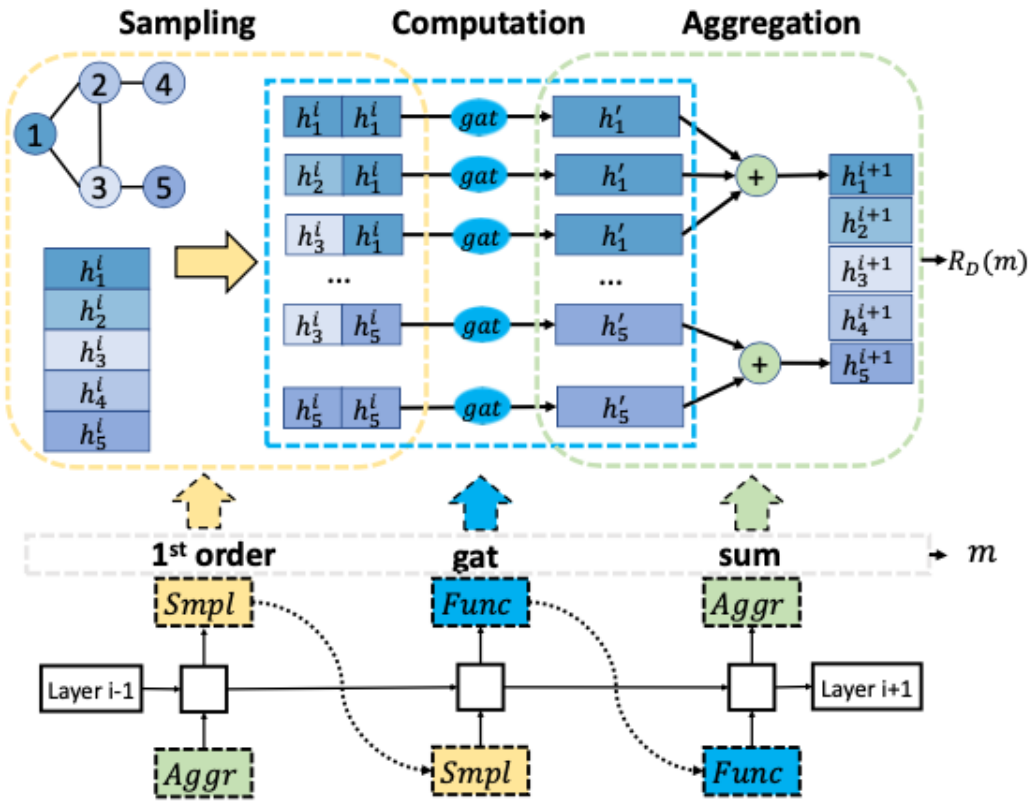
Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

GraphNAS [Gao et al. 2020]

- K-layer GNN
  - List of strings

$$[\ first\_order, gcn, sum, concat, 1, 16, relu,$$
$$first\_order, gat, sum, avg, 8, 16, elu\ ].$$

- Size of the search space
  - $9{,}408^L$
  - 8×10^7 When $L = 2$



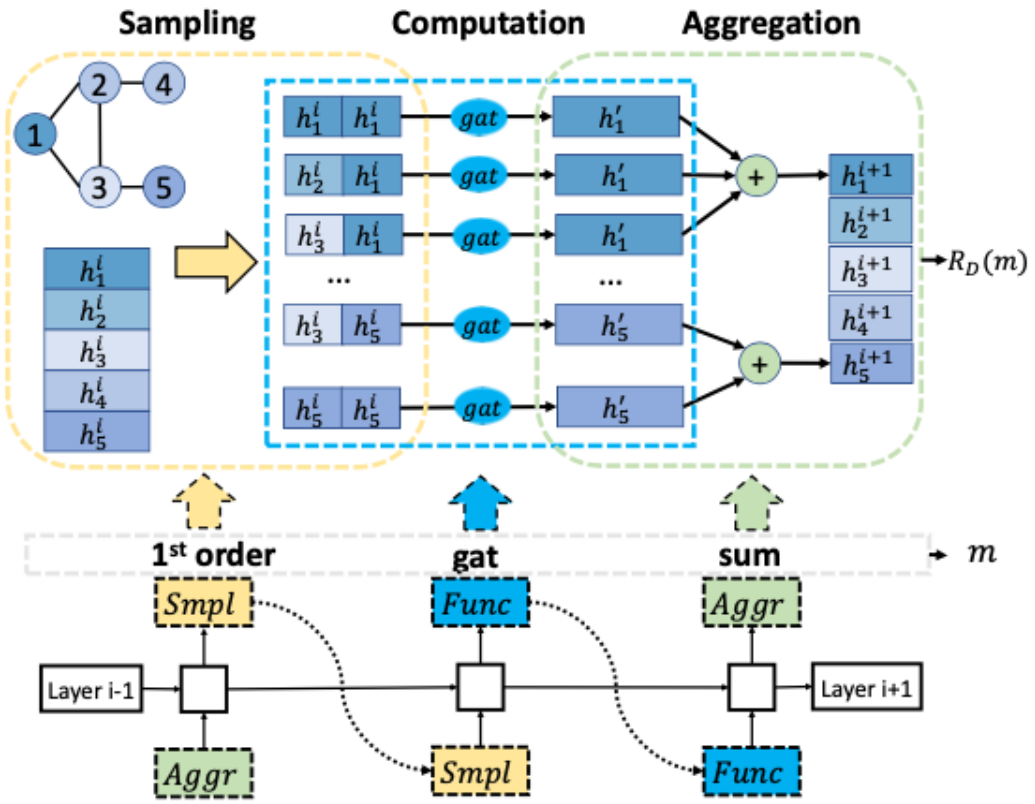
Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

GraphNAS [Gao et al. 2020]

- Search algorithm
  - A RNN controller samples an architecture
  - Training till convergency
  - Update RNN controller based on the validation accuracy

- Search Objective

$$m^* = \arg\max_{m \in \mathcal{M}} \mathbb{E}[R_D(m)].$$



Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

GraphNAS [Gao et al. 2020]

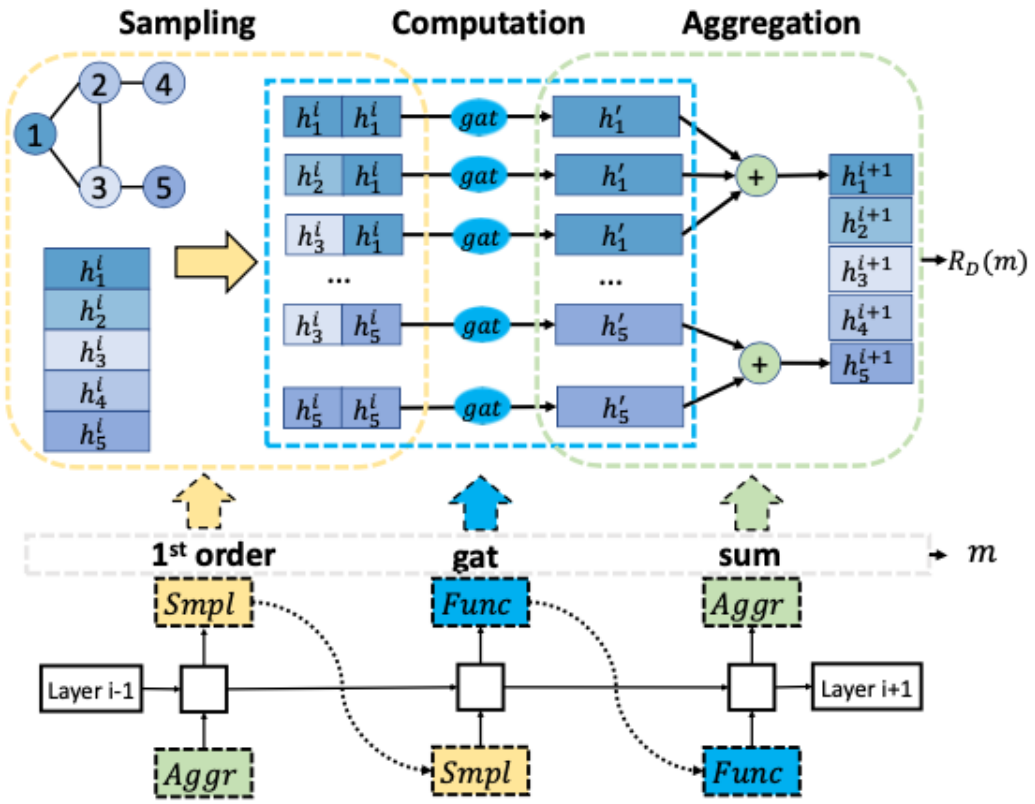
- Search Objective

$$m^* = \arg\max_{m \in \mathcal{M}} \mathbb{E}[R_D(m)].$$

- Policy gradient

$$\nabla_\theta \mathbb{E}_{P(m_{1:T};\theta)}[R]$$
$$= \sum_{t=1}^{T} \mathbb{E}_{P(m_{1:T};\theta)}[\nabla_\theta log P(m_t | m_{t-1:1}; \theta)(R - b)]$$

  - $m_{1:T}$: a list of operators with length T
  - θ : the parameters of the RNN controller



Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

Experiments

- Datasets
  - Three benchmark ones.
- Task
  - Node classification
- Settings
  - Semi-supervised
    - 20 training, 500 validation, 100 test
  - Supervised
    - 500 validation, 500 test, rest training
  - Supervised with randomly split
  - Transfer learning

N, E, F and C denote the number of "Nodes", "Edges", "Features" and "Classes", respectively.

Dataset	N	E	F	C
Cora	2,708	5,278	1,433	7
CiteSeer	3,327	4,552	3,703	6
PubMed	19,717	44,324	500	3

Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

Experiments

- Results
  - SOTA on all datasets.
    - GraphNAS-R: randomly search algorithm
    - GraphNAS-S: simple variant with only one computational node in each layer.

	Cora			Citeseer			Pubmed		
	semi	sup	rand	semi	sup	rand	semi	sup	rand
GCN	81.4±0.5	90.2±0.0	88.3±1.3	70.9±0.5	80.0±0.3	77.2±1.7	79.0±0.4	87.8±0.2	88.1±1.4
GAT	83.0±0.7	89.5±0.3	87.2±1.1	72.5±0.7	78.6±0.3	77.1±1.3	79.0±0.3	86.5±0.6	87.8±1.4
ARMA	82.8±0.6	89.8±0.1	88.2±1.0	72.3±1.1	79.9±0.6	76.7±1.5	78.8±0.3	88.1±0.2	88.7±1.0
APPNP	83.3±0.1	90.4±0.2	87.5±1.4	71.8±0.4	79.2±0.4	77.3±1.6	80.2±0.2	87.4±0.3	88.2±1.1
HGCN	79.8±1.2	89.7±0.4	87.7±1.1	70.0±1.3	79.2±0.5	76.9±1.3	78.4±0.6	88.0±0.5	88.0±1.6
GraphNAS-R	83.3±0.4	90.0±0.3	88.5±1.0	73.4±0.4	81.1±0.3	76.5±1.3	79.0±0.4	90.7±0.6	90.3±0.8
GraphNAS-S	81.4±0.6	90.1±0.3	88.5±1.0	71.7±0.6	79.6±0.5	77.5±2.3	79.5±0.5	88.5±0.2	88.5±1.1
GraphNAS	**83.7±0.4**	**90.6±0.3**	**88.9±1.2**	**73.5±0.3**	**81.2±0.5**	**77.6±1.5**	**80.5±0.3**	**91.2±0.3**	**91.1±1.0**

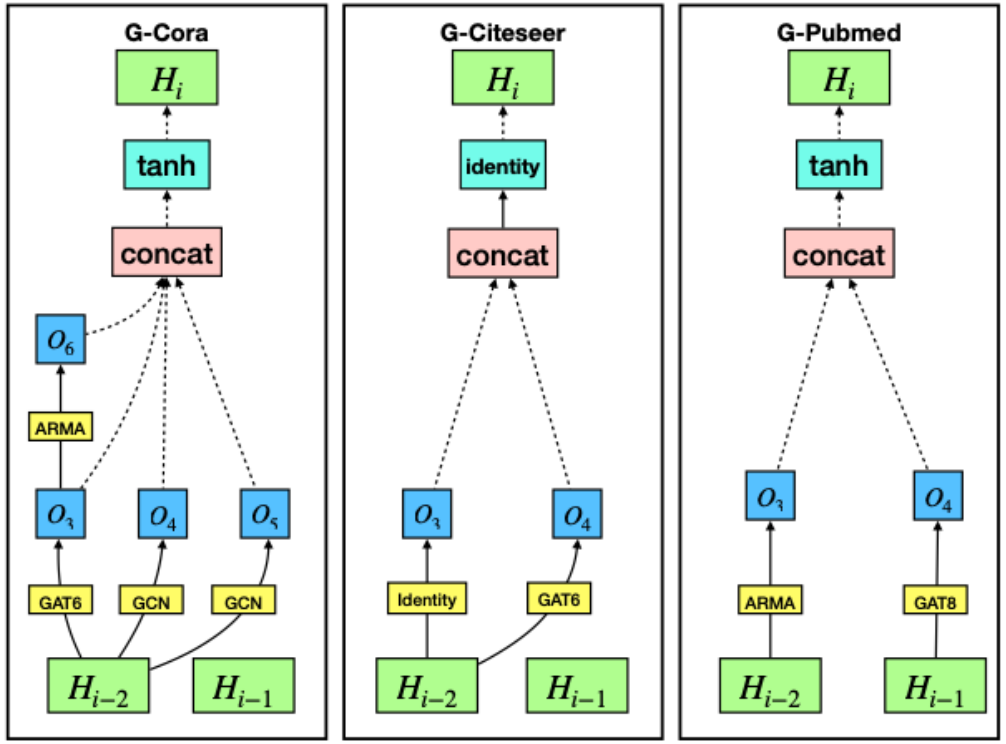
Table 3: Node classification results *w.r.t.* accuracy, where "semi" stands for semi-supervised learning experiments, "sup" for supervised learning experiments and "rand" for supervised learning experiments with randomly split data.

Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

Experiments

- Searched architectures

Figure 5: An example of the graph neural architectures designed by GraphNAS on the supervised learning task. The architecture *G-Cora* designed by GraphNAS on Cora is $[1, gat\_6, 1, gcn, 1, gcn, 3, arma, tanh, concat]$, the architecture *G-Citeseer* designed by GraphNAS on Citeseer is $[1, identity, 1, gat\_6, identity, concat]$, and the architecture *G-Pubmed* designed by GraphNAS on Pubmed is $[2, gat\_8, 1, arma, tanh, concat]$.



Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

Quick Summary

- A first graph neural network architecture search by RL
- Effective but expensive due to "trial-and-error" manner
  - Stand alone
- More efficient search algorithm is needed
  - One-shot

Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

SANE [Zhao et al. 2021]

- Search to Aggregate Neighborhood (SANE) for graph neural networks
  - Differentiable architecture search
  - A research work from our group.

- A more compact search space only search for node and layer aggregation functions
  - The expressive ability of GNNs mainly rely on the aggregation function[Hu et al. 2019]
  - The intermediate layers can further improve the expressive ability [Xu et al. 2018]

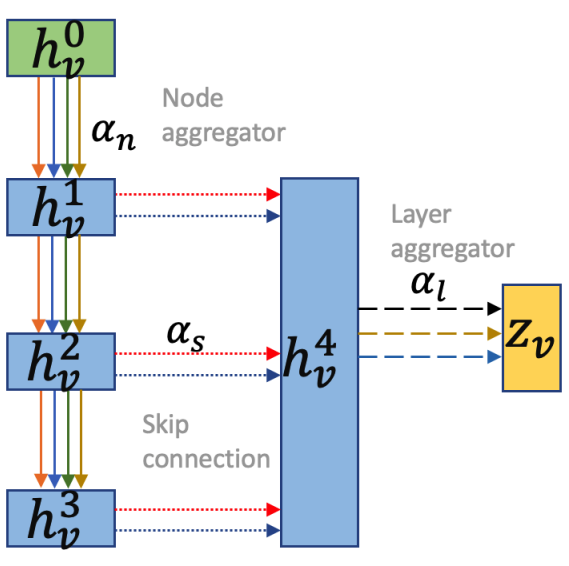
- SOTA performance

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021
Hu et al. How Powerful are Graph Neural Networks? ICLR 2019
Xu et al. Representation Learning on Graphs with Jumping Knowledge Networks. ICML 2018

Search Space

- Message passing framework
  - Node aggregator
  - Layer aggregator

	Operations
O_n	SAGE-SUM, SAGE-MEAN, SAGE-MAX, SAGE-LSTM, GCN, GAT,GAT-SYM, GAT-COS, GAT-LINEAR, GAT-GEN-LINEAR, CNN, MLP, GeniePath
O_l	CONCAT, MAX, SUM, MIN, LSTM
O_s	IDENTITY, ZERO



- Comparing to existing GNNs

	model	node aggregator	layer aggregator	scale to large graph	emulate by SANE
human-designed	GCN [17]	GCN	×	×	✓
	SAGE [15]	SAGE-SUM/-MEAN/-MAX/-LSTM	×	✓	✓
	GAT [30]	GAT, GAT-SYM/-COS/-LINEAR/-GEN-LINEAR	×	×	✓
	GIN [33]	MLP	×	×	✓
	LGCN [11]	CNN	×	×	✓
	GeniePath [21]	GeniePath	×	✓	✓
	JK-Network [34]	depends on the base GNN.	✓	✓	✓
NAS	SANE	learned combination of aggregators	✓	✓	

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Search Space

- More explanations on node aggregator

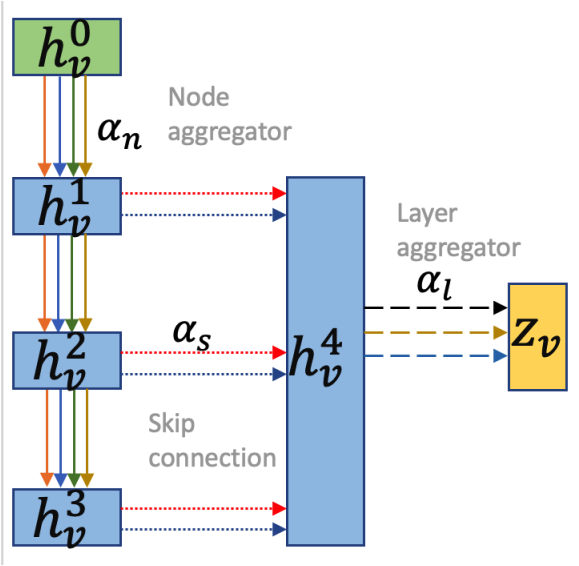
GNN models	Symbol in the paper	Key explanations
GCN [15]	GCN	$F_N^l(v) = \sum_{u \in \widetilde{N}(v)} \left(\text{degree}(v) \cdot \text{degree}(u)\right)^{-1/2} \cdot \mathbf{h}_u^{l-1}$.
GraphSAGE [15]	SAGE-MEAN, SAGE-MAX, SAGE-SUM, SAGE-LSTM	Apply mean, max, sum, or LSTM operation to $\{\mathbf{h}_u \| u \in \widetilde{N}(v)\}$.
GAT [30]	GAT	Compute attention score: $\mathbf{e}_{uv}^{gat} = \text{Leaky\_ReLU}\left(\mathbf{a}\left[\mathbf{W}_u\mathbf{h}_u \| \mathbf{W}_v\mathbf{h}_v\right]\right)$.
	GAT-SYM	$\mathbf{e}_{uv}^{sys} = \mathbf{e}_{uv}^{gat} + \mathbf{e}_{vu}^{gat}$.
	GAT-COS	$\mathbf{e}_{uv}^{cos} = \langle \mathbf{W}_u\mathbf{h}_u, \mathbf{W}_v\mathbf{h}_v \rangle$.
	GAT-LINEAR	$\mathbf{e}_{uv}^{lin} = \tanh\left(\mathbf{W}_u\mathbf{h}_u + \mathbf{W}_v\mathbf{h}_v\right)$.
	GAT-GEN-LINEAR	$\mathbf{e}_{uv}^{gen-lin} = \mathbf{W}_G \tanh\left(\mathbf{W}_u\mathbf{h}_u + \mathbf{W}_v\mathbf{h}_v\right)$.
GIN [33]	MLP	$F_N^l(v) = \text{MLP}\left((1+\epsilon^{l-1}) \cdot \mathbf{h}_v^{l-1} + \sum_{u \in N(v)} \mathbf{h}_u^{l-1}\right)$.
LGCN [11]	CNN	Use 1-D CNN as the aggregator.
GeniePath [21]	GeniePath	Composition of two aggregators, one is GAT, and the other is LSTM-based one.
JK-Network [34]		Depending on the base above GNN.

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Differentiable search algorithm

- Supernet (DAG)
  - Continuous relaxation
  - Mixed OPs

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O} \frac{\exp(\boldsymbol{\alpha}_o^{(i,j)})}{\sum_{o' \in O} \exp(\boldsymbol{\alpha}_{o'}^{(i,j)})} o(x)$$



- Computation process

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}_n^{(l)} \cdot \bar{o}_n(\{\mathbf{h}_u^{(l-1)}, \forall u \in \widetilde{N}(v)\})\right)$$

$$\mathbf{H}_v^{K+1} = \left[\bar{o}_s(\mathbf{h}_v^1), \cdots, \bar{o}_s(\mathbf{h}_v^K)\right]$$

$$\mathbf{z}_v = \bar{o}_l\left(\mathbf{H}_v^{K+1}\right)$$

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

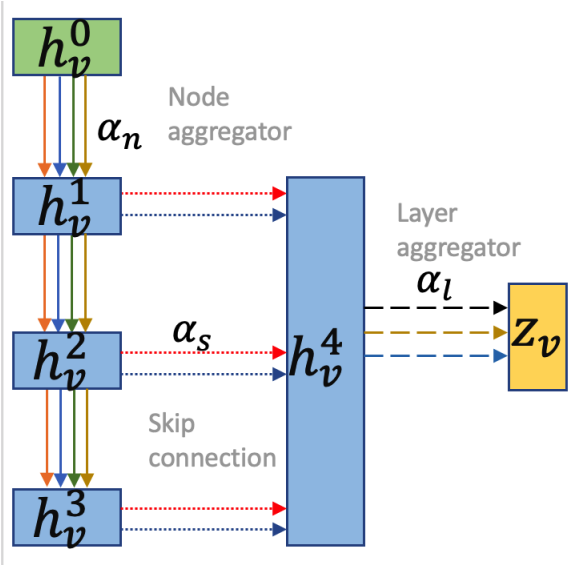
Differentiable search algorithm

- Search $\{\alpha_n, \alpha_s, \alpha_l\}$

DEFINITION 1 (SANE PROBLEM). *Formally, the general paradigm of SANE is to solve a bi-level optimization problem:*

$$\min_{\boldsymbol{\alpha} \in \mathcal{A}} \mathcal{L}_{val}(\boldsymbol{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \ \text{s.t.} \ \boldsymbol{w}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}), \quad (6)$$

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ are the training and validation loss, respectively. $\boldsymbol{\alpha}$ represent a network architecture, where $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_l\}$ and $\boldsymbol{w}^*(\boldsymbol{\alpha})$ the corresponding weights after training.



- Derive architecture
  - Choose the OP with the largest weight.

$$o^{(i,j)} = \arg\max_{o \in O} \boldsymbol{\alpha}_o^{(i,j)}$$

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Differentiable search algorithm

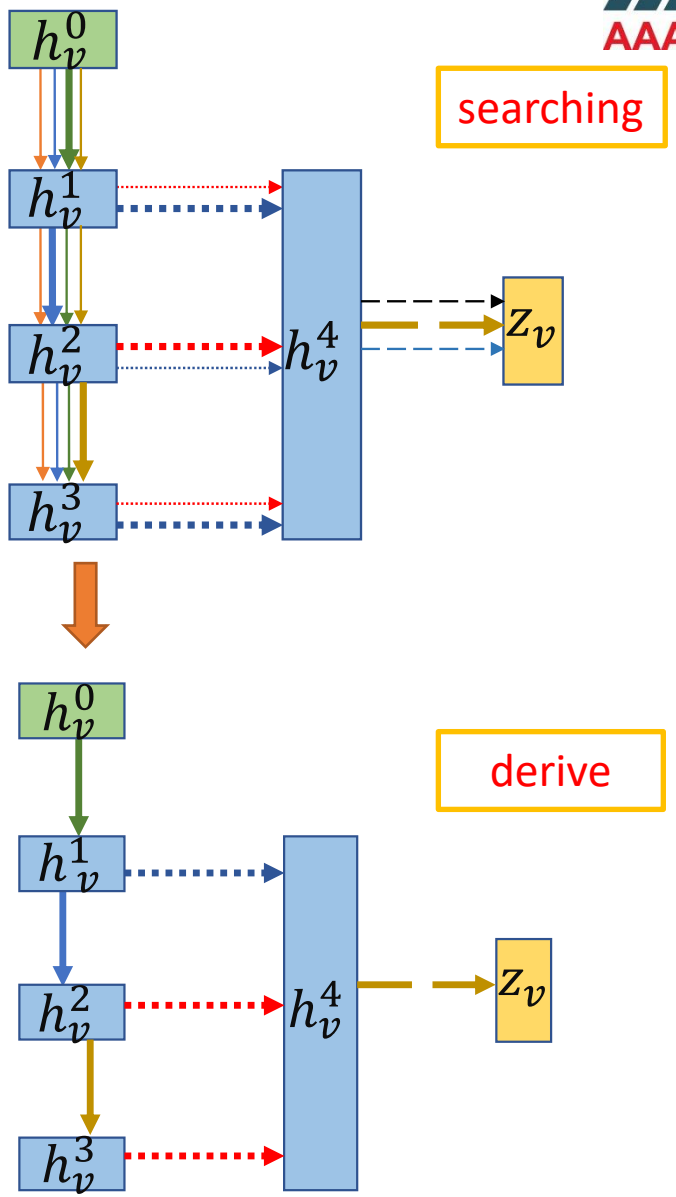
- Gradient-based optimization.

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

Algorithm 1 SANE - Search to Aggregate NEighborhood.

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

1: **while** $t = 1, \cdots, T$ **do**
2: Compute the validation loss $\mathcal{L}_{val}$;
3: Update $\boldsymbol{\alpha}_n$, $\boldsymbol{\alpha}_s$ and $\boldsymbol{\alpha}_l$ by gradient descend rule Eq. (7) with Eq. (3), (4) and (5) respectively;
4: Compute the training loss $\mathcal{L}_{train}$;
5: Update weights $\mathbf{w}$ by descending $\nabla_{\mathbf{w}} \mathcal{L}_{train}(\mathbf{w}, \boldsymbol{\alpha})$ with the architecture $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_l]$;
6: **end while**
7: Derive the final architecture based on the trained $\{\boldsymbol{\alpha}_n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_l\}$;



Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Experiments

- Datasets

Task	Dataset	N	E	F	C
Transductive	Cora	2,708	5,278	1,433	7
	CiteSeer	3,327	4,552	3,703	6
	PubMed	19,717	44,324	500	3
Inductive	PPI	56,944	818,716	121	50

- Task
  - Node classification
- Settings
  - Supervised
  - Transductive and Inductive

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Experiments

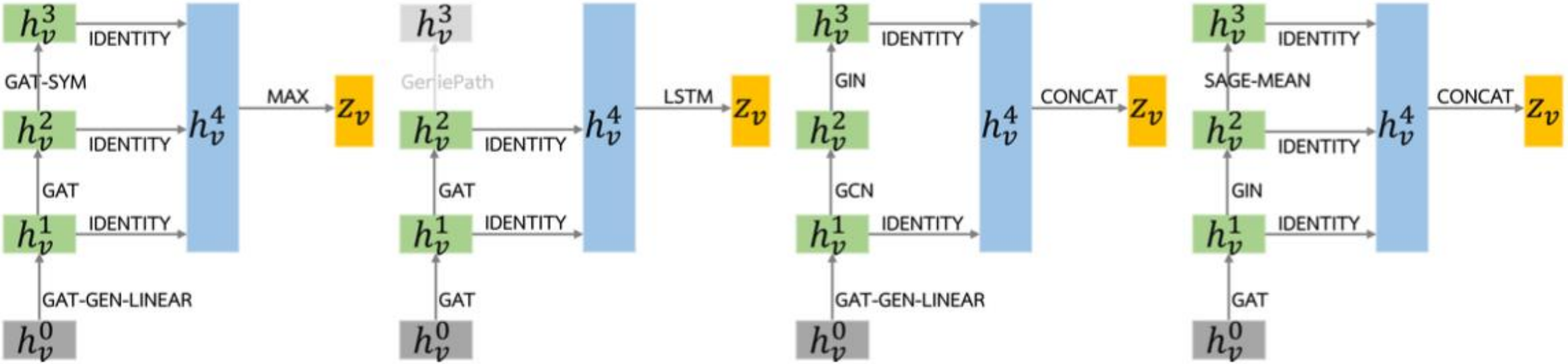
- Performance comparison
  - Human-designed architectures
  - NAS approaches

		Transductive			Inductive
	Methods	Cora	CiteSeer	PubMed	PPI
Human-designed architectures	GCN	0.8811 (0.0101)	0.7666 (0.0202)	0.8858 (0.0030)	0.6500 (0.0000)
	GCN-JK	0.8820 (0.0118)	0.7763 (0.0136)	0.8927 (0.0037)	0.8078(0.0000)
	GraphSAGE	0.8741 (0.0159)	0.7599 (0.0094)	0.8834 (0.0044)	0.6504 (0.0000)
	GraphSAGE-JK	0.8841 (0.0015)	0.7654 (0.0054)	0.8942 (0.0066)	0.8019 (0.0000)
	GAT	0.8719 (0.0163)	0.7518 (0.0145)	0.8573 (0.0066)	0.9414 (0.0000)
	GAT-JK	0.8726 (0.0086)	0.7527 (0.0128)	0.8674 (0.0055)	0.9749 (0.0000)
	GIN	0.8600 (0.0083)	0.7340 (0.0139)	0.8799 (0.0046)	0.8724 (0.0002)
	GIN-JK	0.8699 (0.0103)	0.7651 (0.0133)	0.8878 (0.0054)	0.9467 (0.0000)
	GeniePath	0.8670 (0.0123)	0.7594 (0.0137)	0.8846 (0.0039)	0.7138 (0.0000)
	GeniePath-JK	0.8776 (0.0117)	0.7591 (0.0116)	0.8868 (0.0037)	0.9694 (0.0000)
	LGCN	0.8687 (0.0075)	0.7543 (0.0221)	0.8753 (0.0012)	0.7720 (0.0020)
NAS approaches	Random	0.8594 (0.0072)	0.7062 (0.0042)	0.8866(0.0010)	0.9517 (0.0032)
	Bayesian	0.8835 (0.0072)	0.7335 (0.0006)	0.8801(0.0033)	0.9583 (0.0082)
	GraphNAS	0.8840 (0.0071)	0.7762 (0.0061)	0.8896 (0.0024)	0.9692 (0.0128)
	GraphNAS-WS	0.8808 (0.0101)	0.7613 (0.0156)	0.8842 (0.0103)	0.9584 (0.0415)
one-shot NAS	SANE	**0.8926 (0.0123)**	**0.7859 (0.0108)**	**0.9047 (0.0091)**	**0.9856 (0.0120)**

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Experiments

- Searched architectures

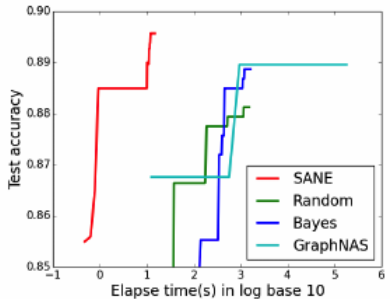


(a) Cora. (b) CiteSeer. (c) PubMed. (d) PPI.

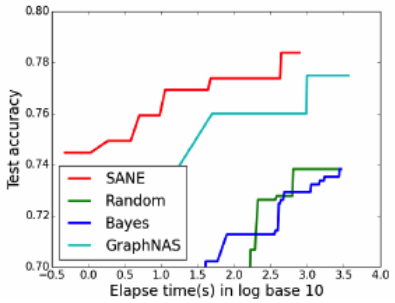
Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Experiments

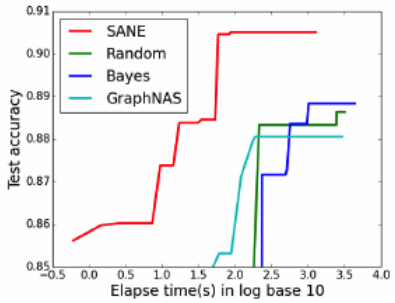
- The test accuracy w.r.t. search time(s)



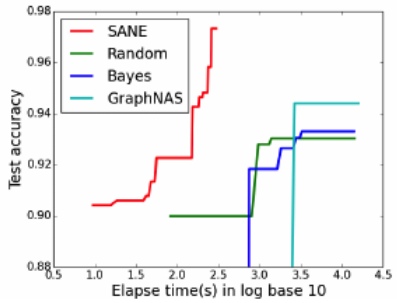
(a) Cora.



(b) CiteSeer.



(c) PubMed.



(d) PPI.

- The total time (s) of running once of each model

	transductive task			inductive task
	Cora	CiteSeer	PubMed	PPI
Random	1,500	2,694	3,174	13,934
Bayesian	1,631	2,895	4,384	14,543
GraphNAS	3,240	3,665	5,917	15,940
SANE	14	35	54	298

Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021

Quick Summary

- SANE shows potential to obtain SOTA GNN architectures more efficiently.

- Comparisons
  - GraphNAS
    - More accurate evaluation on the child model
    - Computationally expensive
  - SANE
    - More efficient
    - Not that robust

PAS [Wei et al. 2021]

- Pooling Architecture Search (PAS) for Graph Classification
  - The first method to learn data-specific pooling architectures.

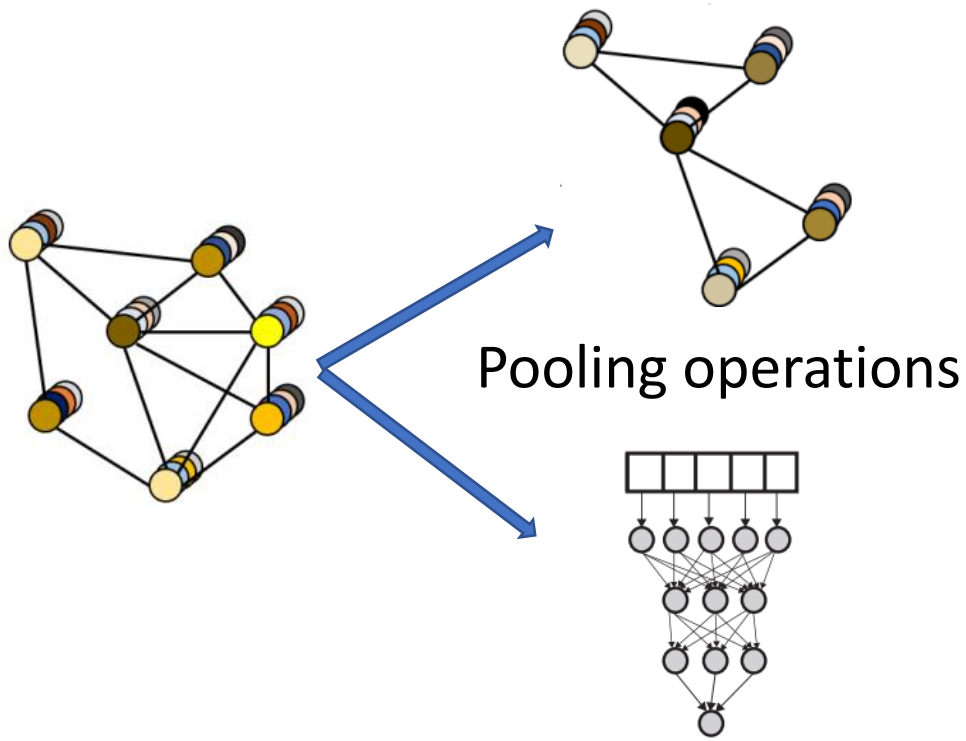
- Differentiable pooling architecture search
  - The hierarchical pooling method is not always superior to the global one.[Diego et al. 2020, Federico et al. 2020]
  - A unified framework which can design the global and hierarchical pooling methods adaptively.
  - A coarsening strategy is designed to relax the search space and enable the usage of the differentiable search algorithm.

- SOTA performance

Wei et al. Pooling Architecture Search for Graph Classification. CIKM 2021
Diego et al. Rethinking pooling in graph neural networks. NeurIPS 2020
Federico et al. A fair comparison of graph neural networks for graph classification. ICLR 2020

Pooling methods

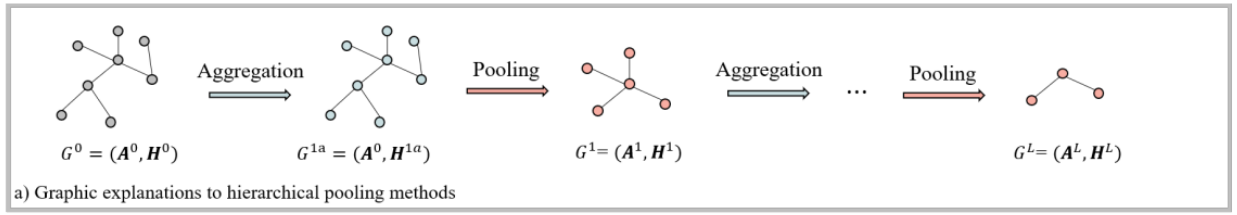
- Global pooling
  - Pool nodes at the end of GNN.
  - Representative works: GIN[1] / DGCNN[2]
  - Learn flat graph-level representation.
- Hierarchical pooling
  - Learn coarser and coarser graphs in the hierarchical pooling methods.
  - Representative works: DiffPool [3] / SAGPool [4] / ASAP [5]



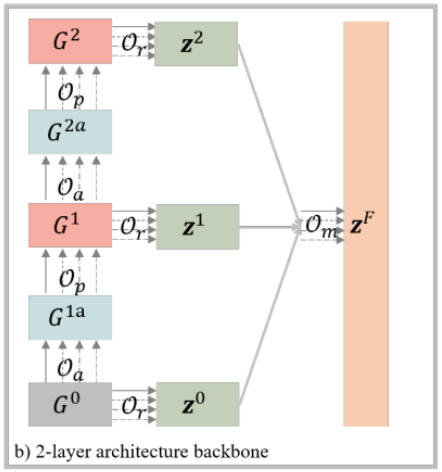
[1] Keyulu Xu et al. How powerful are graph neural networks? ICLR 2019
[2] Muhan Zhang et al. An end-to-end deep learning architecture for graph classification. AAAI 2018
[3] Zhitao Ying et al. Hierarchical graph representation learning with differentiable pooling. NeurIPS 2018
[4] Junhyun Lee et al. Self-Attention Graph Pooling. ICML 2019
[5] Ekagra Ranjan et al. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. AAAI 2020

The unified framework

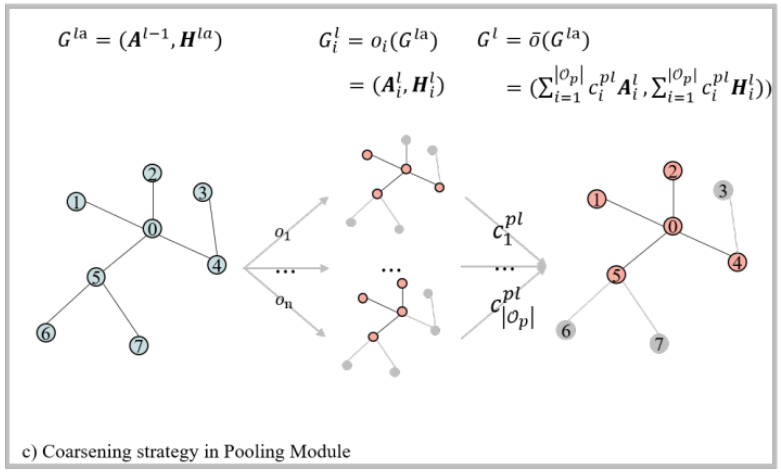
- Four key modules derived from existing pooling methods.
- One novel search space which can cover both the global and hierarchical pooling architectures.
- One coarsening strategy to relax the search space.



a) Graphic explanations to hierarchical pooling methods



b) 2-layer architecture backbone



c) Coarsening strategy in Pooling Module

	Methods	Task	Search Space				Search
			A	P	R	M	Algorithm
GNNs	GraphSAGE [13]	Node	√	-	-	×	-
	JK-Net [42]	Node	√	-	-	√	-
	GIN [41]	Graph	√	×	√	×	-
	DiffPool [44]	Graph	√	√	√	×	-
	SAGPool [20]	Graph	√	√	√	√	-
NAS	GraphNAS [10]	Node	√	-	-	×	RL
	SNAG [50]	Node	√	-	-	√	RL
	RE-MPNN [16]	Graph	√	×	√	√	EA
	AutoGM [45]	Node	√	-	-	×	Bayesian
	SANE [51]	Node	√	-	-	√	Gradient Descent
	DSS [23]	Node	√	-	-	×	Gradient Descent
	GNAS [3]	Node	√	-	-	×	Gradient Descent
	PAS (proposed)	Graph	√	√	√	√	Gradient Descent

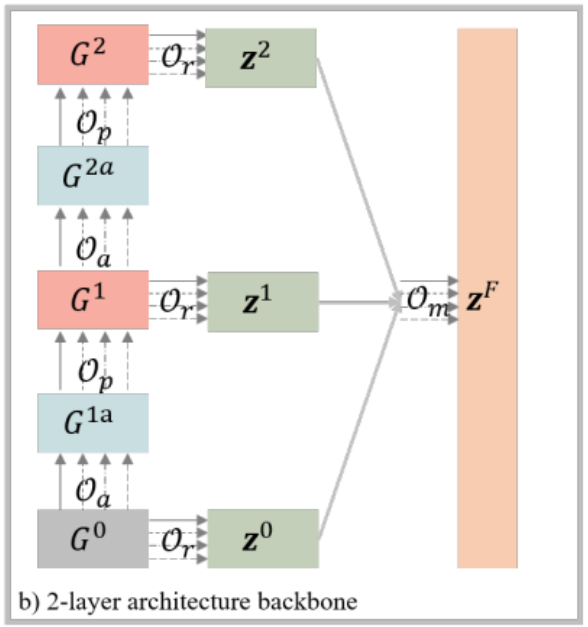
Search Space

Pooling Module:

- Calculate the node scores and generate the coarsen graphs by selecting Top-K nodes.

$$\mathbf{S} = f_s(\mathbf{A}, \mathbf{H}), idx = \mathtt{TOP}_k(\mathbf{S}),$$
$$\mathbf{A}' = \mathbf{A}(idx, idx), \mathbf{H}' = \mathbf{H}(idx, :).$$

- NONE operation: no pooling operation in this layer.



b) 2-layer architecture backbone

Module name	Operations
Aggregation $\mathcal{O}_a$	GCN, GAT, SAGE, GIN, GRAPHCONV, MLP
Pooling $\mathcal{O}_p$	TOPKPOOL, SAGPOOL, ASAP, HOPPOOL_1, HOPPOOL_2, HOPPOOL_3, MLPPOOL, GCPOOL, GAPPOOL, NONE
Readout $\mathcal{O}_r$	GLOBAL_SORT, GLOBAL_ATT, SET2SET, GLOBAL_MEAN, GLOBAL_MAX, GLOBAL_SUM, ZERO
Merge $\mathcal{O}_m$	M_LSTM, M_CONCAT, M_MAX, M_MEAN, M_SUM

Differentiable search algorithm

- It is unachievable to relax the pooling module since different coarse graphs contain diverse nodes and edges.

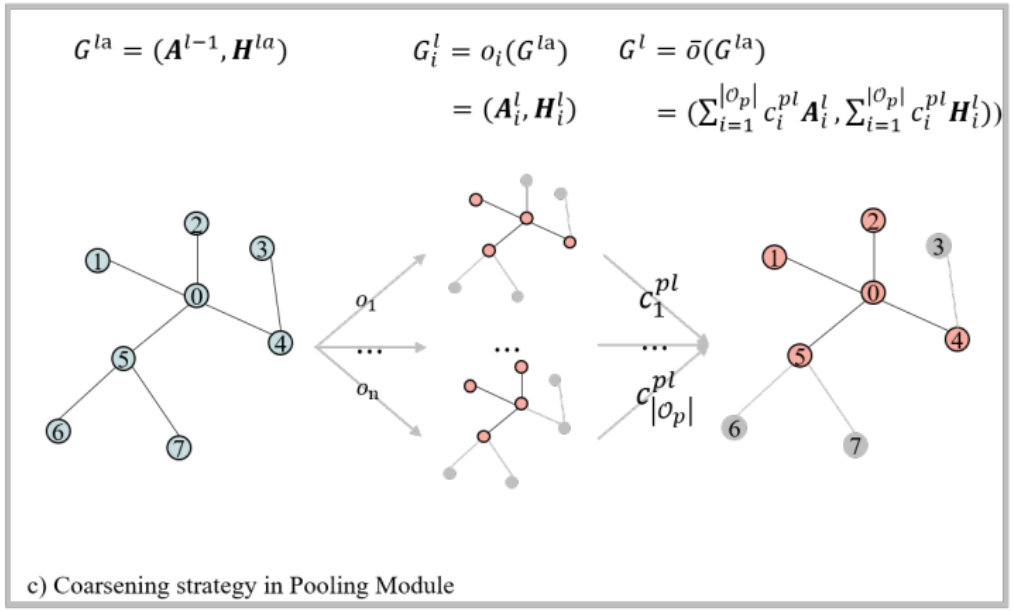
$V_1 = \{0,1,4,5\},\ A_1 \epsilon \mathbb{R}^{4\times4}, H_1 \epsilon \mathbb{R}^{4\times d}$

$V_{\mathrm{n}} = \{0,1,2,5\},\ A_n \epsilon \mathbb{R}^{4\times4}, H_n \epsilon \mathbb{R}^{4\times d}$

Mismatched nodes

- Coarsening strategy
  - In pooling operations, keep the same "shape" when formulating the coarsen graphs.
  - For unselected nodes and edges (in grey), we set the features and weights to 0.

$$G^l = (\boldsymbol{A}^l, \boldsymbol{H}^l) = (\sum_{i=1}^{|O_p|} c_i^{pl} \boldsymbol{A}_i^l, \sum_{i=1}^{|O_p|} c_i^{pl} \boldsymbol{H}_i^l)$$



Experiments

- Datasets
  - 6 real-world datasets from 3 domains
- Baselines
  - Global pooling methods
  - Hierarchical pooling methods
  - NAS methods:
    - RL based: GraphNAS[1] / SNAG[2]
    - EA / Random / Bayesian

Dataset	# of Graphs	# of Feature	# of Classes	Avg.# of Nodes	Avg.# of Edges	Domain
D&D	1,178	89	2	384.3	715.7	Bioinfo
PRO	1,113	3	2	39.1	72.8	Bioinfo
IMDB-B	1,000	0	2	19.8	96.5	Social
IMDB-M	1,500	0	3	13	65.9	Social
COX2	467	3	2	41.2	43.5	Chemistry
NCI109	4127	0	2	29.69	32.13	Chemistry

[1] Gao et al. Graphnas: Graph neural architecture search with reinforcement learning. IJCAI 2020
[2] Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

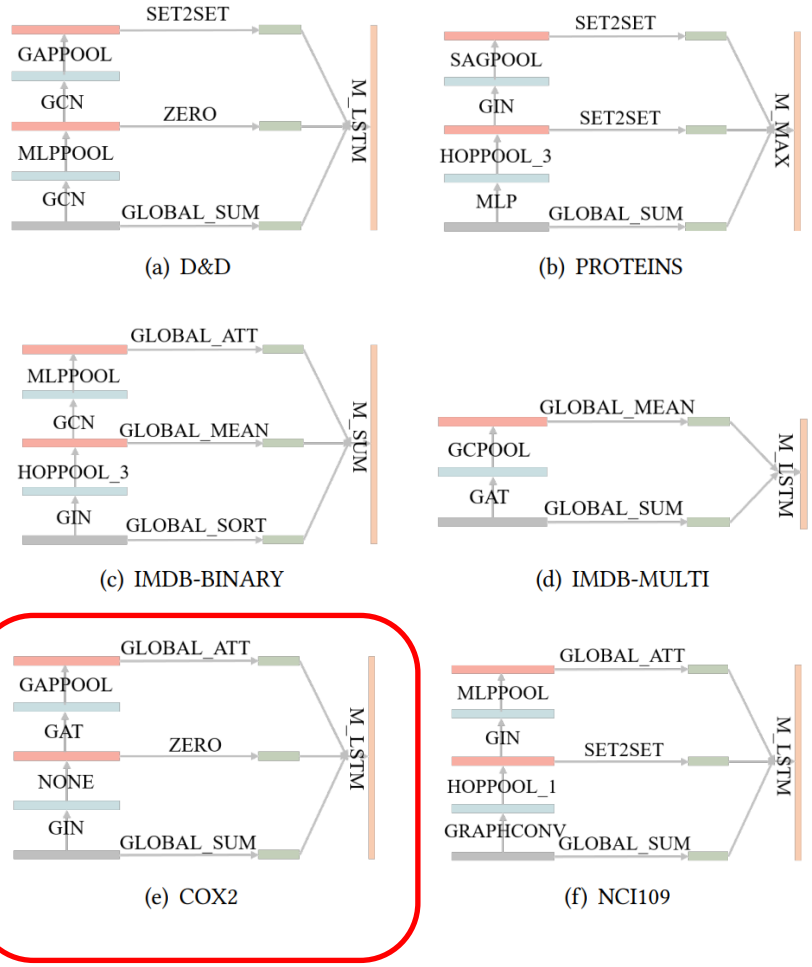
Experiments

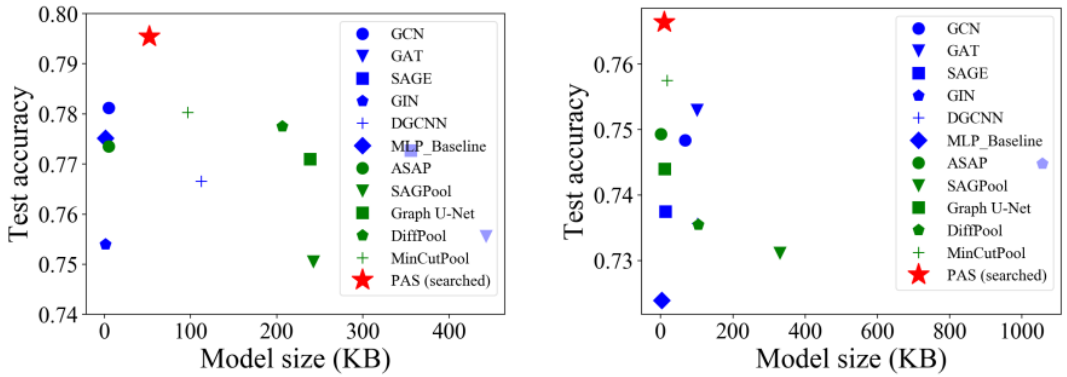
- No absolute winner from human-designed models and PAS consistently outperforms all baselines.
- Compared with NAS baselines, the proposed differentiable search algorithm provide performance gain.

Table 4: Performance comparisons of PAS and all baselines. We report the mean test accuracy and the standard deviation by 10-fold cross-validation. The best results in different groups of baselines are underlined, and the best result on each dataset is in boldface.

	Method	D&D	PROTEINS	IMDB-BINARY	IMDB-MULTI	COX2	NCI109
Global pooling	GCN	0.7812±0.0433	0.7484±0.0282	0.7267±0.0642	0.5040±0.0302	0.7923±0.0219	0.7344 ± 0.0192
	GAT	0.7556±0.0372	0.7530±0.0372	0.7407±0.0453	0.4967±0.0430	0.8156±0.0417	0.7410 ± 0.0245
	SAGE	0.7727±0.0406	0.7375±0.0297	0.7217±0.0529	0.4853±0.0543	0.8031±0.0594	0.7553 ± 0.0164
	GIN	0.7540±0.0368	0.7448±0.0278	0.7167±0.0277	0.4980±0.0250	0.8309±0.0417	0.7456 ± 0.0210
	GCN-JK	0.7769±0.0235	0.7539±0.0517	0.7347±0.0445	0.4886±0.0496	0.7966±0.0226	0.7383 ± 0.0188
	GAT-JK	0.7809±0.0618	0.7457±0.0405	0.7327±0.0468	0.4953±0.0384	0.8072±0.0287	0.7172 ± 0.0292
	SAGE-JK	0.7735±0.0420	0.7494±0.0355	0.7287±0.0618	0.4973±0.0340	0.7990±0.0415	0.7325 ± 0.0356
	GIN-JK	0.7513±0.0395	0.7312±0.0451	0.7207±0.0486	0.5080±0.0302	0.8117±0.0467	0.7441 ±0.0220
	DGCNN	0.7666±0.0403	0.7357±0.0469	0.7367±0.0570	0.4900±0.0356	0.7985±0.0264	0.7506 ± 0.0165
	MLP-Baseline	0.7752±0.0390	0.7239±0.0353	0.7287±0.0520	0.4980±0.0428	0.7838±0.0104	0.6445 ± 0.0160
Hierarchical pooling	ASAP	0.7735±0.0415	0.7493±0.0357	0.7427±0.0397	0.5013±0.0344	0.8095±0.0320	0.7376 ± 0.0224
	SAGPool	0.7506±0.0506	0.7312±0.0447	0.7487±0.0409	0.4933±0.0490	0.7945±0.0298	0.6489 ± 0.0315
	Graph U-Net	0.7710±0.0517	0.7440±0.0349	0.7317±0.0484	0.4880±0.0319	0.8030±0.0421	0.7279 ± 0.0229
	DiffPool	0.7775±0.0400	0.7355±0.0322	0.7186±0.0563	0.4953±0.0398	0.7966±0.0264	0.7315 ± 0.0214
	MinCutPool	0.7803±0.0363	0.7575±0.0380	0.7077±0.0489	0.4900±0.0283	0.8007±0.0385	0.7405±0.0248
NAS	GraphNAS	0.7198±0.0454	0.7251±0.0336	0.7110±0.0230	0.4693±0.0364	0.7773±0.0140	0.7228 ± 0.0228
	GraphNAS-WS	0.7674±0.0455	0.7520±0.0251	0.7360±0.0463	0.4827±0.0350	0.7816±0.0058	0.7049 ± 0.0192
	SNAG	0.7223±0.0386	0.7053±0.0311	0.7250±0.0461	0.4933±0.0289	0.7903±0.0212	0.7090±0.0224
	SNAG-WS	0.7351±0.0303	0.7233±0.0244	0.7360±0.0516	0.5000±0.0248	0.8054±0.0381	0.7063±0.0160
	EA	0.7514±0.0301	0.7341±0.0298	0.7400±0.0412	0.4860±0.0405	0.7945±0.0159	0.7324±0.0126
	Random	0.7792±0.0482	0.7394±0.0423	0.7210±0.0554	0.4980±0.0398	0.8073±0.0231	0.7306±0.0241
	Bayesian	0.7555±0.0321	0.7314±0.0239	0.7270±0.0335	0.4980±0.0397	0.8029±0.0172	0.7204±0.0114
	PAS	**0.7896±0.0368**	**0.7664±0.0329**	**0.7510±0.0532**	**0.5220±0.0373**	**0.8344±0.0633**	**0.7684±0.0272**

Experiments



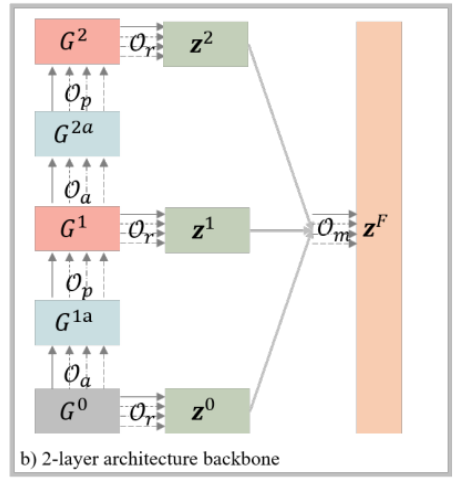


- Data-specific pooling architectures are obtained.
- Global pooling architecture on COX2 is obtained by PAS.
- These searched architectures can achieve SOTA performance with moderate size in terms of the model parameters.

Ablation study

- Ablation studies on 4 modules:
  - Aggregation Module: fixed aggregators
  - Pooling Module: fixed NONE operation
  - Readout Module: fixed GLOBAL_MEAN operation
  - Merge Module: remove Merge Module

Fixed		D&D	IMDB-MULTI
Aggregation	PAS-GCN	0.7835±0.0407	0.5027±0.0409
	PAS-GAT	0.7878±0.0376	0.5087±0.0417
Pooling	PAS-Global	0.7708±0.0330	0.5173±0.0447
Readout	PAS-FR	0.7436±0.0472	0.5033±0.0436
Merge	PAS-RM	0.7682±0.0336	0.5047±0.0380
	PAS	**0.7896±0.0368**	**0.5220±0.0373**



b) 2-layer architecture backbone

Module name	Operations
Aggregation $\mathcal{O}_a$	GCN, GAT, SAGE, GIN, GRAPHCONV, MLP
Pooling $\mathcal{O}_p$	TOPKPOOL, SAGPOOL, ASAP, HOPPOOL_1, HOPPOOL_2, HOPPOOL_3, MLPPOOL, GCPOOL, GAPPOOL, NONE
Readout $\mathcal{O}_r$	GLOBAL_SORT, GLOBAL_ATT, SET2SET, GLOBAL_MEAN, GLOBAL_MAX, GLOBAL_SUM, ZERO
Merge $\mathcal{O}_m$	M_LSTM, M_CONCAT, M_MAX, M_MEAN, M_SUM

Quick Summary

- PAS is the first method to learn pooling architectures for graph classification.

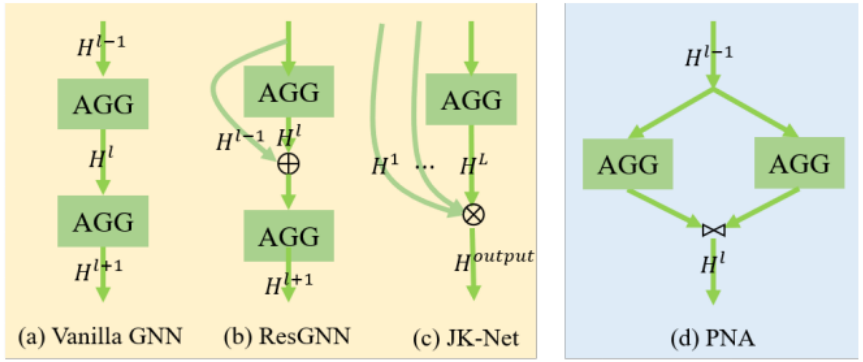
- Comparisons
  - Search space
    - Design the global pooling functions.[Jiang et al. 2020, Wang et al. 2021]
    - Design pooling architectures on the global and hierarchical methods adaptively.
  - Search algorithm
    - Design a coarsening strategy to enable the usage of gradient descent.

Jiang et al. Graph neural network architecture search for molecular property prediction. Big Data 2020
Wang et al. AutoGEL: An Automated Graph Neural Network with Explicit Link Information. NeurIPS 2021

F^{2}GNN [Wei et al. 2022]

- Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective
  - A systematic approach for the GNN topology design.
  - An orthogonal and complementary approach to existing Graph NAS methods.

- GNN topology design
  - Stack aggregation operations.
  - Use identity skip-connections.
  - Use multiple aggregation operations in each layer.
- SOTA performance



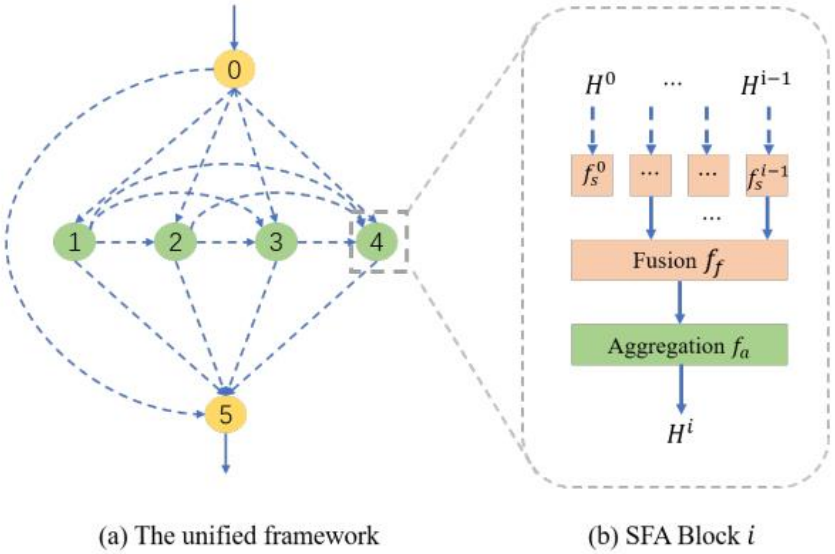
Wei et al. Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective. WWW 2022
Diego et al. Rethinking pooling in graph neural networks. NeurIPS 2020
Federico et al. A fair comparison of graph neural networks for graph classification. ICLR 2020

GNN topology design

- Two mainstream GNN topology design manners
  - Stacking AGG: extract higher-level feature / Over-smoothing.
  - Using multiple AGG: adequate and independent feature extraction / costly to obtain the higher-level information.
- Design Target: Can we enjoy the benefits while alleviate the corresponding deficiencies on top of these two topology design manners?
- GNN topology units
  - The topology of a neural network is represented by its “computational graph”.
  - Design the links is equivalent to selecting the features of different levels.
  - Topology design target → feature selection and fusion strategies in GNNs.

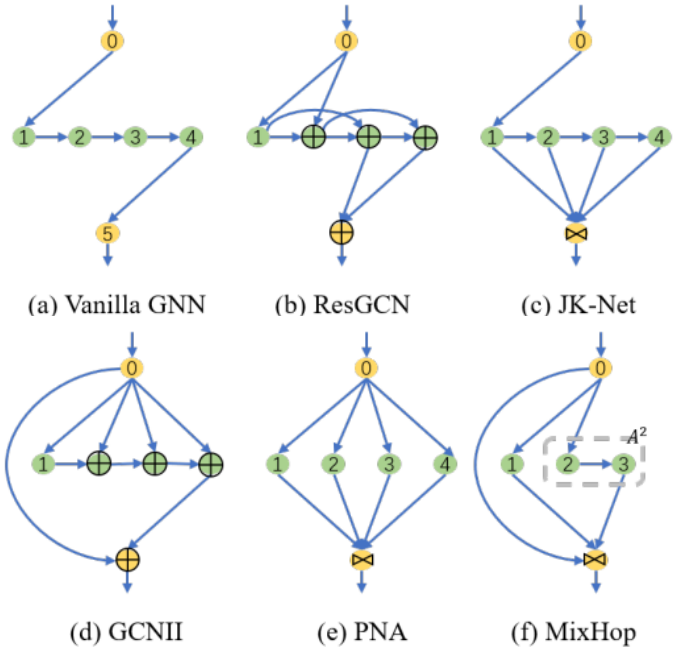
The Unified Framework

- The feature selection and fusion strategies lead to the key difference of topology designs in GNNs.



(a) The unified framework (b) SFA Block i

- Feature fusion framework
  - Pre-processing: 2-layer MLPs
  - SFA Block: Selection + Fusion + Aggregation
  - Post-processing: Selection + Fusion + 2-layer MLPs



(a) Vanilla GNN (b) ResGCN (c) JK-Net

(d) GCNII (e) PNA (f) MixHop

Search Space

- Predefined aggregation operation: Design the Selection and Fusion Operations with GraphSAGE (F^2SAGE) and GAT (F^2GAT).
- Learnable aggregation operation (F^2GNN): Design the Selection, Fusion and Aggregation Operations.

Table 1: The operations used in our search space.

	Operations
Selection $\mathcal{O}_s$	ZERO, IDENTITY
Fusion $\mathcal{O}_f$	SUM, MEAN, MAX, CONCAT, LSTM, ATT
Aggregation $\mathcal{O}_a$	GCN, GAT, SAGE, GIN

Search Algorithm

- The optimization gap in feature fusion
  - Mixed results

$$\bar{o}^{ij}(\mathbf{x}_i) = \sum_{k=1}^{|O_s|} c_k^{ij} o_k^{ij}(\mathbf{x}_i) = c_1^{ij}\mathbf{0} + c_2^{ij}\mathbf{x}_i = c_2^{ij}\mathbf{x}_i .$$

  - ChildNet results:
    - ZERO: $c_1^{ij}\mathbf{0}$
    - IDENTITY: $c_2^{ij}\mathbf{x}_i$

The IDENTITY operation has a large influence when ZERO is selected, and the influence will accumulate along with the feature selection operation in the framework.

- Improved search with the usage of temperature

  Add a small temperature λ in the softmax.

$$c_k = \frac{\exp(\bar{\alpha}_k/\lambda)}{\sum_{i=1}^{|O|}\exp(\alpha_i/\lambda)}$$

Experiments

- Datasets
  - Five homophily + three Heterophily
- Baselines
  - Stacking baselines
  - Various identity skip-connections
  - Multiple aggregations
  - Graph NAS baselines

Datasets	#Nodes	#Edges	#Features	#Classes	h
Cora [37]	2,708	5,278	1,433	7	0.81
Computers [32]	13,381	245,778	767	10	0.78
DBLP [2]	17,716	105,734	1,639	4	0.83
PubMed [37]	19,717	44,324	500	3	0.80
Physics [38]	34,493	495,924	8,415	5	0.93
Wisconsin [34]	251	466	1,703	5	0.21
Actor [34]	7,600	30,019	932	5	0.22
Flickr [52]	89,250	899,756	500	7	0.32

$$h = \frac{|\{(u,v):(u,v)\in\mathcal{E}\wedge y_u=y_v\}|}{|\mathcal{E}|}$$

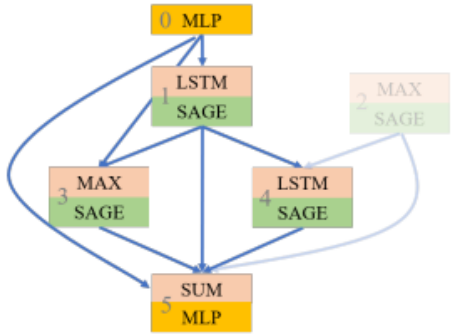
Gao et al. Graphnas: Graph neural architecture search with reinforcement learning. IJCAI 2020
Huan Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Experiments

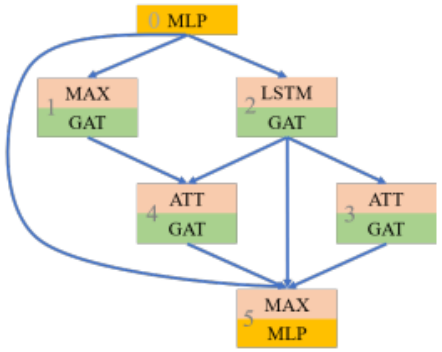
- Adaptive GNN topologies (ours+Random) have better performance than the human-designed topologies.
- Our methods rank 1st compared with all the human-designed topologies and the Random Baselines.

Aggregation	Topology	Cora	DBLP	PubMed	Computers	Physics	Actor	Wisconsin	Flickr	Avg. Rank (Group)	Avg. Rank (All)
SAGE	Stacking (L2)	86.09(0.50)	83.58(0.33)	88.96(0.29)	91.14(0.30)	96.42(0.11)	34.78(1.10)	79.61(5.56)	51.21(0.71)	6.63	15.00
	Stacking (L4)	85.68(0.61)	83.83(0.32)	88.23(0.28)	90.52(0.42)	95.97(0.14)	34.61(1.08)	60.39(10.77)	53.07(0.50)	8.25	17.00
	RES (L4)	85.66(0.52)	83.39(0.30)	88.99(0.25)	91.51(0.18)	96.31(0.17)	35.16(0.94)	76.47(5.26)	53.72(0.27)	5.25	13.13
	DENSE (L4)	86.68(0.59)	83.30(0.73)	89.42(0.27)	90.74(0.51)	96.48(0.14)	34.78(0.60)	77.06(6.01)	53.17(0.19)	4.50	12.75
	JK (L4)	86.47(0.60)	83.94(0.62)	89.21(0.29)	91.21(0.30)	96.56(0.05)	36.53(0.92)	81.96(4.71)	52.41(0.33)	4.75	10.38
	GNNII (L4)	85.83(0.42)	84.46(0.45)	89.21(0.24)	91.38(0.27)	96.45(0.15)	35.70(1.11)	81.57(4.13)	52.24(0.29)	4.50	11.50
	PNA (L2)	84.29(0.67)	82.76(0.42)	89.25(0.26)	90.67(0.42)	96.32(0.10)	33.89(2.68)	75.29(6.46)	52.09(0.73)	8.88	17.75
	MixHop (L2)	84.81(0.95)	82.65(0.65)	89.25(0.28)	88.56(1.61)	96.11(0.17)	35.19(0.62)	81.57(2.51)	51.75(0.59)	6.75	17.75
	Random	86.75(0.29)	83.60(0.29)	89.21(0.04)	91.30(0.19)	96.46(0.03)	36.30(0.58)	85.10(5.63)	54.10(0.15)	3.50	8.75
	F^2SAGE	87.72(0.26)	84.81(0.06)	89.73(0.26)	91.81(0.26)	96.72(0.01)	36.61(1.00)	85.88(1.92)	53.66(0.16)	2.00	4.38
GAT	Stacking (L2)	85.92(0.72)	84.34(0.26)	87.56(0.23)	91.49(0.21)	95.76(0.16)	29.28(1.02)	53.73(7.24)	53.83(0.28)	5.25	14.25
	Stacking (L4)	86.16(0.55)	84.29(0.41)	85.73(0.34)	89.08(0.43)	93.47(3.93)	26.45(1.00)	45.29(5.65)	50.34(2.68)	8.25	19.88
	RES (L4)	84.66(0.92)	84.11(0.34)	87.56(0.44)	90.84(0.49)	95.67(0.28)	28.98(0.36)	48.82(3.77)	53.63(0.24)	7.50	18.50
	DENSE (L4)	85.31(0.86)	83.43(0.37)	88.67(0.19)	91.30(0.37)	96.16(0.06)	31.78(1.03)	53.33(7.73)	53.61(0.26)	6.25	16.38
	JK (L4)	86.55(0.46)	83.73(0.35)	89.71(0.16)	91.80(0.23)	96.80(0.09)	35.43(0.88)	84.51(5.58)	53.02(0.29)	3.88	8.75
	GNNII (L4)	85.40(1.06)	83.83(0.33)	88.44(0.25)	91.91(0.11)	96.14(0.15)	30.29(0.78)	55.29(6.25)	53.03(0.29)	5.38	15.00
	PNA (L2)	85.06(0.72)	83.46(0.47)	87.18(0.30)	90.84(0.24)	95.85(0.18)	28.56(0.82)	49.22(5.91)	54.02(0.33)	7.38	18.25
	MixHop (L2)	85.38(1.04)	82.50(0.34)	88.91(0.19)	91.27(0.37)	96.46(0.21)	35.70(0.90)	81.57(4.40)	53.67(0.30)	5.13	13.25
	Random	85.73(0.06)	83.60(0.19)	88.86(0.18)	91.76(0.14)	96.84(0.09)	36.07(0.83)	86.08(4.15)	52.43(0.29)	4.38	10.38
	F^2GAT	88.31(0.12)	84.76(0.04)	90.38(0.14)	92.04(0.17)	97.10(0.03)	36.65(1.13)	87.06(4.13)	53.45(0.19)	1.63	3.13
Learnable	SNAG (L4)	84.99(1.04)	84.29(0.15)	87.93(0.16)	85.98(0.72)	96.18(0.11)	28.13(0.74)	43.92(4.65)	53.50(0.31)	4.00	18.63
	SANE (L4)	86.40(0.38)	84.58(0.13)	89.34(0.31)	91.02(0.21)	96.80(0.06)	36.77(1.15)	86.47(3.09)	53.92(0.14)	2.63	6.38
	Random	86.99(0.60)	84.62(0.15)	89.37(0.26)	91.03(0.20)	96.72(0.04)	36.29(1.52)	85.49(4.31)	54.33(0.11)	2.25	6.13
	F^2GNN	87.42(0.42)	84.95(0.15)	89.79(0.20)	91.42(0.26)	96.92(0.06)	37.08(1.00)	88.24(3.72)	53.96(0.20)	1.13	2.75

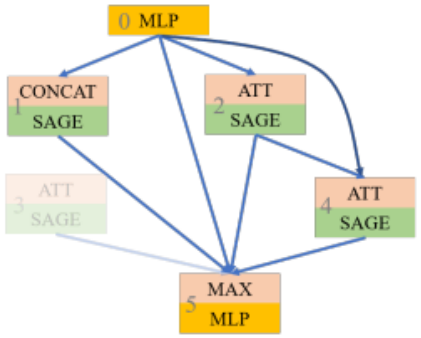
Experiments



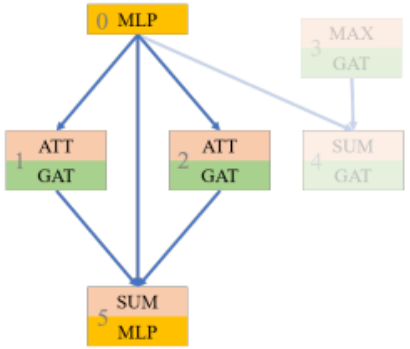
(a) F²SAGE on Cora



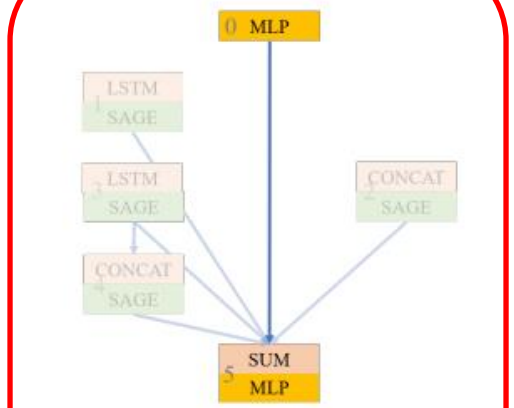
(b) F²GAT on Cora



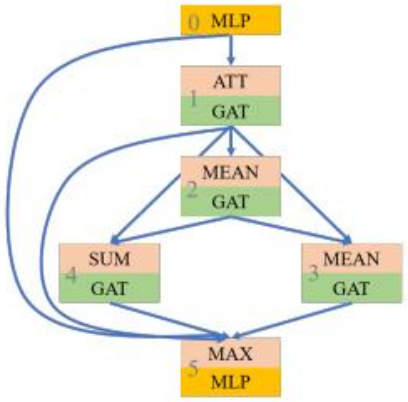
(c) F²SAGE on Physics



(d) F²GAT on Physics



(e) F²SAGE on Actor



(f) F²GAT on Actor

- Data-specific GNN topologies are obtained.
- The initial feature is selected in most GNNs.
- We can benefit from the multiple aggregation design manner.
- On the heterophily dataset Actor, we obtained an MLP network, which shows that the graph structure is not always useful for the final performance.

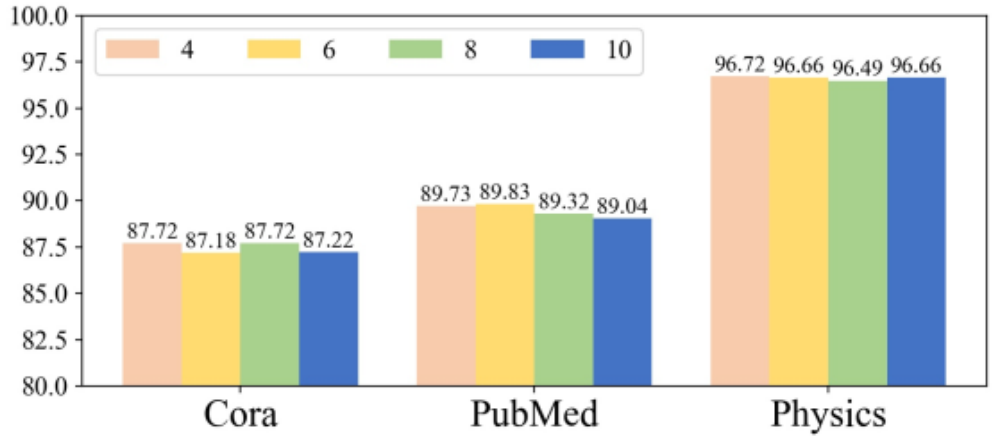
NAS provides a unifying solution of GNN design on both homophily and heterophily graphs.

Ablation study

- Designing the fusion strategy is significant.
- The optimization gap has a large influence on the feature selection, and it can be addressed with λ .
- The increasing number of SFA blocks do not bring about the performance drop due to the adaptive utilization of different levels of features .

Method	Cora	PubMed	Physics
F^2SAGE-SUM	84.73(0.63)	89.39(0.21)	96.44(0.01)
F^2SAGE-MEAN	84.30(0.61)	89.58(0.22)	96.42(0.03)
F^2SAGE-CONCAT	86.07(0.45)	89.31(0.19)	96.69(0.01)
F^2SAGE	87.72(0.26)	89.73(0.26)	96.72(0.01)

Temperature λ	F^2SAGE		F^2AGG	
	Supernet	Childnet	Supernet	Childnet
1	80.33	6.68	86.83	85.71
0.1	73.65	10.96	84.23	83.86
0.01	70.13	70.13	84.60	84.60
0.001	80.15	80.15	86.83	86.83



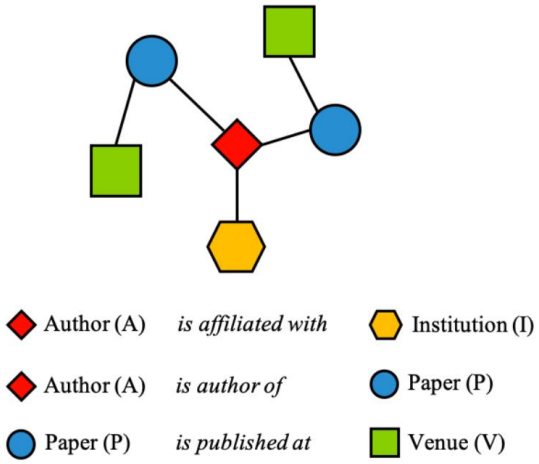
Quick Summary

- F^{2}GNN focuses on designing the topology of GNNs in the feature fusion perspective.
- Comparisons
  - Existing Graph NAS methods focus on designing the aggregation operations or the skip connections on top of the stacking manner. [Gao et al. 2020, Zhao et al. 2021]
  - RAN designed the topologies by randomly wiring the edges. [Valsesia et al. 2021, Xie et al. 2019]
  - F^{2}GNN provides a systematic approach to design the GNN topologies.
- F^{2}GNN provides a unifying solution of GNN design on both homophily and heterophily graphs.

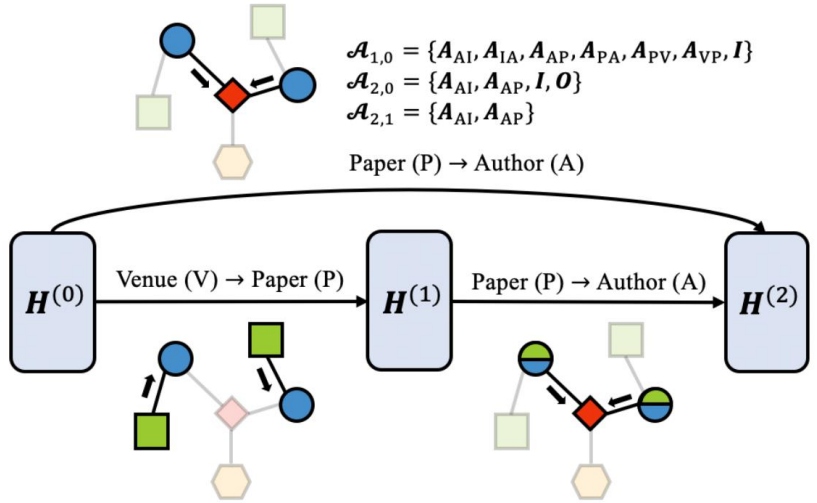
Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020
Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021
Xie et al. Exploring randomly wired neural networks for image recognition. ICCV 2019
Valsesia et al. DON'T STACK LAYERS IN GRAPH NEURAL NETWORKS, WIRE THEM RANDOMLY. ICLR 2021 workshop on 'Geometrical and Topological Representation Learning

One more work -- DiffMG

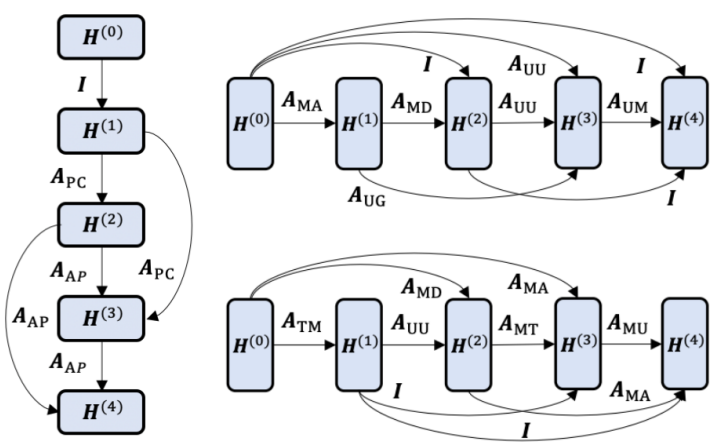
- Heterogeneous Graph
  - Data-specific HeteGNN
  - Utilized different relations



(a) An example academic network with network schema.



(b) Search space with $K = 2$.



(a) DBLP. (b) Douban.

Figure 3: Architectures discovered by DiffMG.

Table 3: Macro F1 scores (%) on the node classification task.

	metapath2vec	GCN	GAT	HAN	MAGNN	GTN	HGT	DiffMG
DBLP	89.93±0.45	90.46±0.41	93.92±0.28	92.13±0.26	92.81±0.30	93.98±0.32	93.67±0.22	**94.45±0.15**
ACM	67.13±0.50	92.56±0.20	92.50±0.23	91.20±0.25	91.15±0.19	92.62±0.17	91.83±0.23	**92.65±0.15**
IMDB	40.82±1.48	55.19±0.99	53.37±1.27	55.09±0.67	56.44±0.63	59.68±0.72	59.35±0.79	**61.04±0.56**

Table 4: AUC (%) on the recommendation task.

	metapath2vec	GCN	GAT	HAN	MAGNN	GEMS	GTN	HGT	DiffMG
Amazon	58.17±0.14	66.64±1.00	55.70±1.13	67.35±0.11	68.26±0.09	70.66±0.14	71.82±0.18	74.75±0.08	**75.28±0.08**
Yelp	51.98±0.14	58.98±0.52	56.55±0.05	64.28±0.20	64.73±0.24	65.12±0.27	66.27±0.31	68.07±0.35	**68.77±0.13**
Douban	51.60±0.07	77.95±0.05	77.58±0.33	82.65±0.08	82.44±0.17	83.00±0.05	83.26±0.10	83.38±0.06	**83.78±0.09**

Ding et al. DiffMG: Differentiable Meta Graph Search for Heterogeneous Graph Neural Networks. KDD 2021

Summary

- It is an interesting and challenging problem to design data-specific GNN due to the diversity and complexity of graph-based tasks.
- NAS show great potential in discovering better GNN models in practice.
- More following works can be checked in [Zhang et al. 2021] .

- Future directions
  - Large-scale Graph Architecture search
    - Open Graph Benchmark (OGB)
  - More challenging settings
    - Heterophily/adversary/noisy

Zhang et al. Automated Machine Learning on Graphs: A Survey. IJCAI 2021

References

- Semi-supervised classification with graph convolutional networks. ICLR 2017
- Inductive representation learning on large graphs. NeurIPS 2017
- Graph attention networks. ICLR 2018
- Representation Learning on Graphs with Jumping Knowledge Networks. ICML 2018
- Adaptive Stochastic Natural Gradient Method for One-Shot Neural Architecture Search. ICML 2019
- GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020
- DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. ICLR 2019
- How powerful are graph neural networks?. ICLR 2019
- Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020
- Search to aggregate neighborhood for graph neural networks. ICDE 2021
- Design Space for Graph Neural Networks. NeurIPS 2020
- Policy-GNN: Aggregation Optimization for Graph Neural Networks. KDD 2020
- Taking Human out of Learning Applications: A Survey on Automated Machine Learning. Arxiv 2018
- Automated Machine Learning on Graphs: A Survey. IJCAI 2021
- Pooling Architecture Search for Graph Classification. CIKM 2021
- Designing the Topology of Graph Neural Networks: A Novel Feature Fusion Perspective. WWW 2022
- Graph Differentiable Architecture Search with Structure Learning. NeurIPS 2021
- Rethinking Graph Neural Architecture Search from Message-passing. CVPR 2021
- One-shot Graph Neural Architecture Search with Dynamic Search Space. AAAI 2021

References

- DiffMG: Differentiable Meta Graph Search for Heterogeneous Graph Neural Networks. KDD 2021
- Graph neural network architecture search for molecular property prediction. Big Data 2020
- AutoGEL: An Automated Graph Neural Network with Explicit Link Information. NeurIPS 2021
- GraphPAS: Parallel Architecture Search for Graph Neural Networks . SIGIR 2021
- Exploring randomly wired neural networks for image recognition. ICCV 2019
- DON'T STACK LAYERS IN GRAPH NEURAL NETWORKS, WIRE THEM RANDOMLY. Arxiv 2021
- Pooling related
- An end-to-end deep learning architecture for graph classification. AAAI 2018
- Hierarchical graph representation learning with differentiable pooling. NeurIPS 2018
- Self-Attention Graph Pooling. ICML 2019
- ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. AAAI 2020

Automated Learning from Graph-Structured Data

Part 2: Automated Graph Neural Network

Q&A

Code and slides can be accessed in Github (left)

https://hzhaoaf.github.io/

Dr. Huan Zhao

Senior researcher, 4Paradigm

Homepage