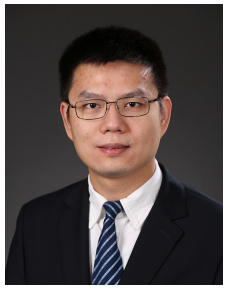
# Automated Recommender System (RecSys)

Quanming Yao[1,2], Yong Li[1], Chen Gao[1], Huan Zhao[2], Yongqi Zhang[2]

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

[2]*4Paradigm Inc.*

https://quanmingyao.github.io/AutoML.github.io/ijcai21-tutorial.html





# Tutorial Outline

1. An introduction to Automated Machine Learning (AutoML)
   - Background on technical tools from machine learning
2. Why AutoML is Needed in RecSys and Recent Advances
   - Exemplar works introducing AutoML into RecSys
3. Automated Graph Representation Learning for RecSys
   - Explore neural architecture search for GNN based RecSys
4. Automated Knowledge Graph (KG) Embedding
   - Explore AutoML for KG Embedding based RecSys

# Schedule at a Glance

| Time | Event |
|---|---|
| 0:00-0:40 minutes | **Part 1**: An introduction to Automated Machine Learning (AutoML) |
| | **Speaker**: Quanming Yao |
| 0:40-1:20 minutes | **Part 2**: Why AutoML is Needed in RecSys and Recent Advances |
| | **Speaker**: Chen Gao |
| 1:20-1:30 minutes | **Break** |
| 1:30-2:10 minutes | **Part 3**: Automated Graph Neural Network for RecSys |
| | **Speaker**: Huan Zhao |
| 2:10-2:50 minutes | **Part 4**: Automated Knowledge Graph Embedding |
| | **Speaker**: Yongqi Zhang |
| 2:50-3:00 minutes | **Part 5**: Discussion |

Automated Recommender System (RecSys) Tutorial

# Part 3: Automated Graph Neural Network for Recommender System

Dr. Huan Zhao

Senior researcher, 4Paradigm

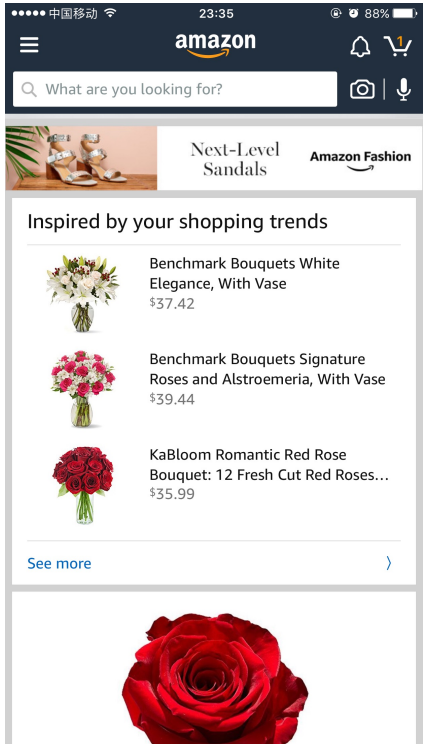
zhaohuan@4paradigm.com

Aug. 22th 2021

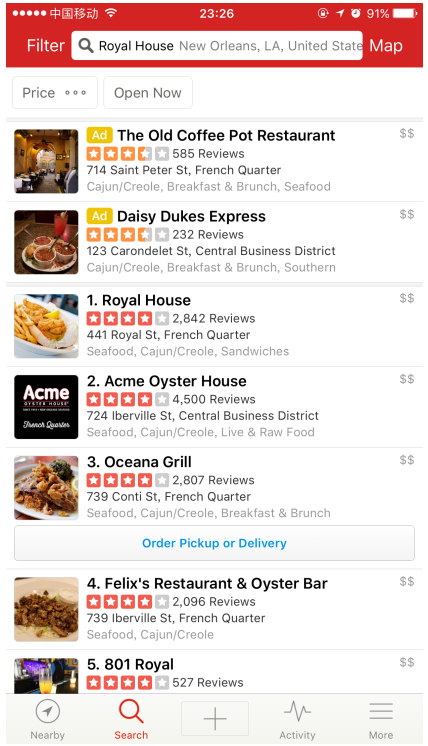
# Outline

- Recommender System (RS)
  - Collaborative Filtering (CF)
  - Graph Neural Network (GNN) based CF
- Neural architecture search (NAS) for GNN
  - Reinforcement learning
  - Differentiable architecture search
- Discussion on neural architecture search for GNN based CF
- Conclusion

# Recommender Systems(RSs) are Everywhere




commodity



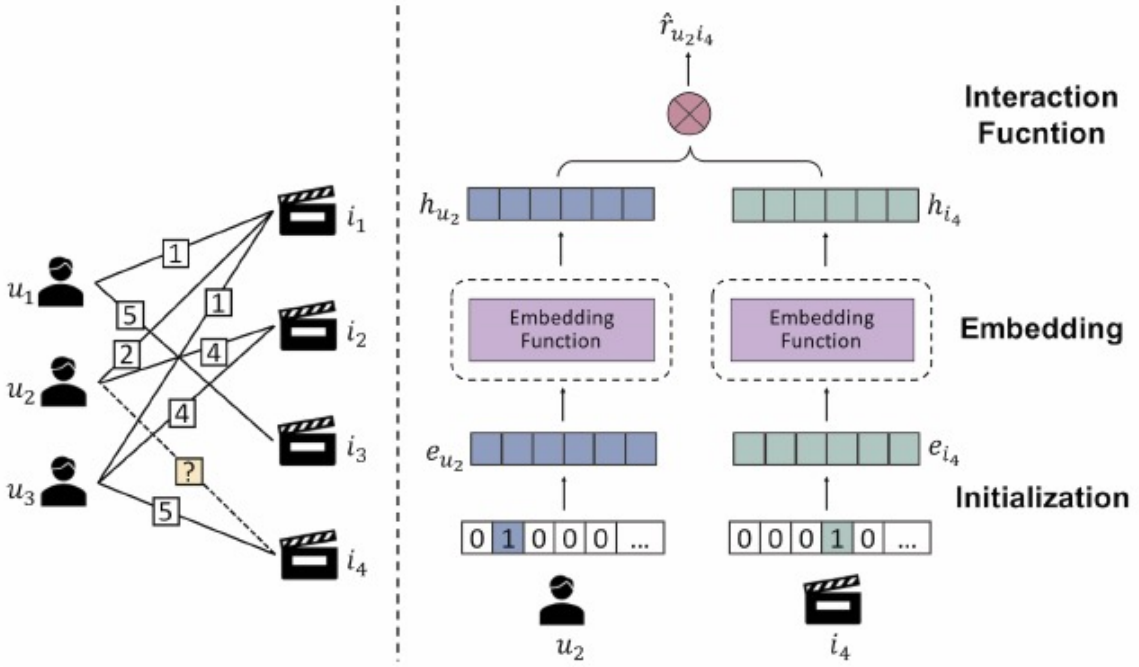

business


commodity


news

# Collaborative Filtering (CF)

- One of the most popular RS methods.
  - Predict the ratings of users giving to items.
  - Similar users share similar behaviors
- Embedding based methods
  - Matrix Factorization
  - Neural Collaborative Filtering
- Graph Neural Network (GNN)
  - User-item interactions → bipartite graph



# PinSAGE [Ying et al. 2018]

- The first graph convolutional neural networks for industrial RecSys.
  - Recommend similar items to users based previously pinned ones.
  - Bipartite graph


Image credit: Jure

Ying et al. Graph convolutional neural networks for web-scale recommender systems. KDD 2018

# PinSAGE [Ying et al. 2018]

- The first graph convolutional neural networks for industrial RecSys.
  - Learn item representations by recursively aggregating neighborhoods.

**Algorithm 1:** CONVOLVE

**Input** : Current embedding $\mathbf{z}_u$ for node $u$; set of neighbor embeddings $\{\mathbf{z}_v | v \in \mathcal{N}(u)\}$, set of neighbor weights $\boldsymbol{\alpha}$; symmetric vector function $\gamma(\cdot)$

**Output**: New embedding $\mathbf{z}_u^{\text{NEW}}$ for node $u$

$\mathbf{n}_u \leftarrow \gamma\left(\{\text{ReLU}\left(\mathbf{Q}\mathbf{h}_v + \mathbf{q}\right) \mid v \in \mathcal{N}(u)\}, \boldsymbol{\alpha}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \text{ReLU}\left(\mathbf{W} \cdot \text{CONCAT}(\mathbf{z}_u, \mathbf{n}_u) + \mathbf{w}\right)$;

$\mathbf{z}_u^{\text{NEW}} \leftarrow \mathbf{z}_u^{\text{NEW}} / \|\mathbf{z}_u^{\text{NEW}}\|_2$

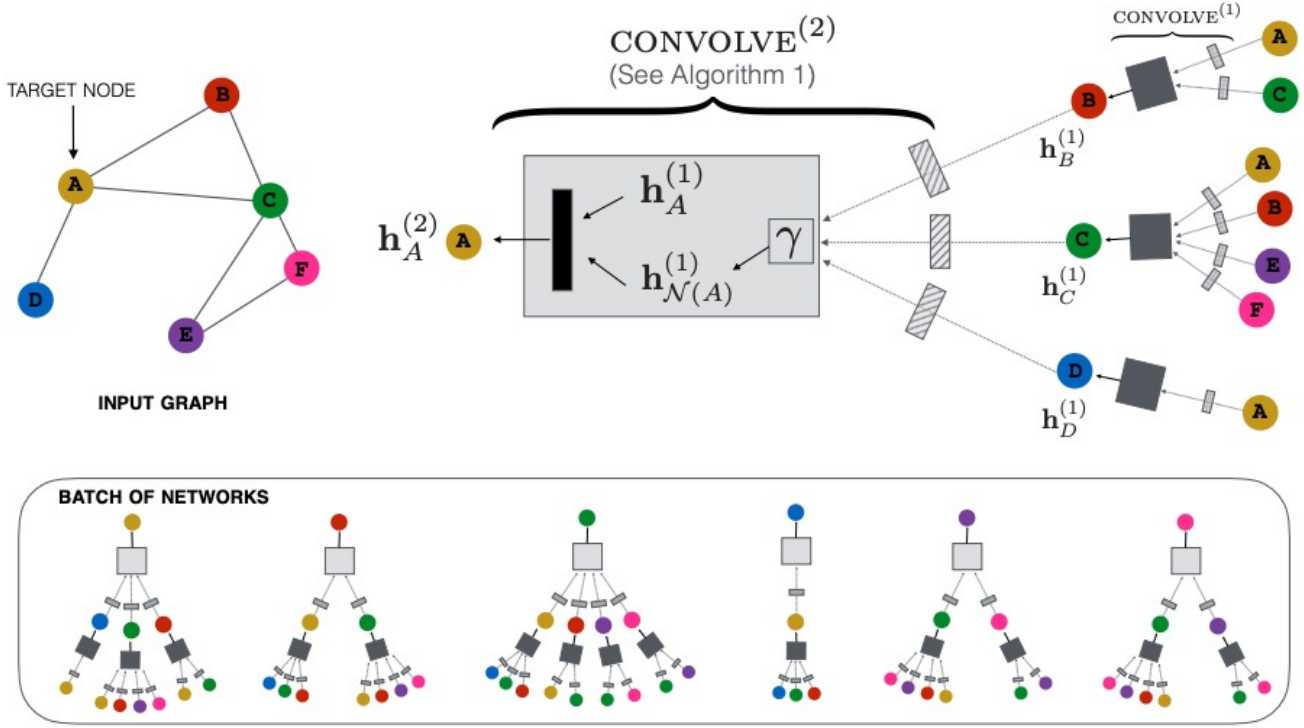



Image credit: Jure

Ying et al. Graph convolutional neural networks for web-scale recommender systems. KDD 2018

# NGCF [Wang et al. 2019]

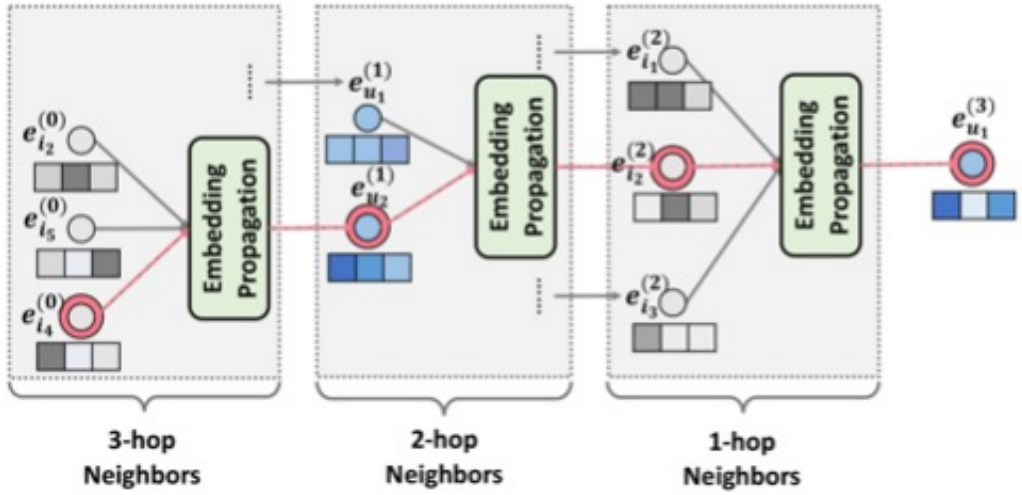
- Neural Graph Collaborative Filtering
  - To capture high-order connectivity in the bipartite graph.
    - Stacking GNN layers.

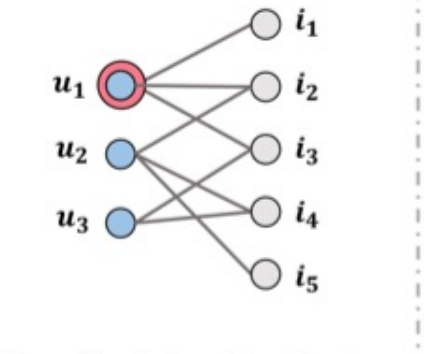
$$\mathbf{e}_u^{(l)} = \text{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}^{(l)}\right)$$

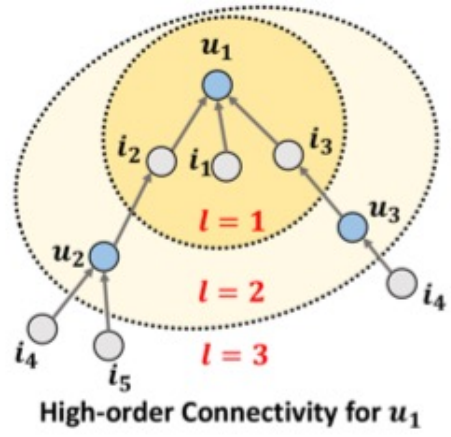
$$\begin{cases} \mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right), \\ \mathbf{m}_{u\leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases}$$

Why $u_1$ may like $i_4$

- $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$
- $u_1 \leftarrow i_3 \leftarrow u_3 \leftarrow i_4$



# GNN based CF

- Various GNN models have been designed for CF [Wu et al. 2020]
  - Pros
    - User-item interactions are naturally modeled by a bipartite graph
      - More accurate representations.
    - High-order information is captured by stacking GNN layers
      - Recursively neighborhood aggregation
  - Cons
    - Recommending scenarios are quite diverse…
      - News, music, e-commerce, business, books, etc.
    - GNN models are difficult to generalize across recommending scenarios.
      - Computationally expensive to design a top-performing model in a new scenario.

**Table 2: Statistics of the datasets.**

| Dataset | # of Users | # of Items | # of Interactions | Rating Scale | Density |
|---|---|---|---|---|---|
| Yelp[1] | 58,069 | 31,721 | 1,160,605 | [1,5] | 0.063% |
| Amazon-CDs [7] | 31,296 | 24,379 | 622,163 | [1,5] | 0.082% |
| Amazon-Movies [7] | 44,439 | 25,047 | 1,070,860 | [1,5] | 0.096% |
| YahooMusic [4, 23] | 1,357 | 1,363 | 5,335 | [1,100] | 0.28% |
| Amazon-Beauty [7] | 7,068 | 3,570 | 79,506 | [1,5] | 0.32% |
| Flixster[11, 23] | 2,341 | 2,956 | 26,173 | [0.5,5] | 0.38% |
| Douban [21, 23] | 2,999 | 3,000 | 136,891 | [1,5] | 1.52% |
| MovieLens-1M[2] | 6,040 | 3,706 | 1,000,209 | [1,5] | 4.47% |
| MovieLens-100K[3] | 943 | 1,682 | 100,000 | [1,5] | 6.31% |

- AutoML can help in designing data-specific GNN architectures!
  - Neural Architecture Search (NAS)

Wu et al. Graph neural networks in recommender systems: a survey. 2020.

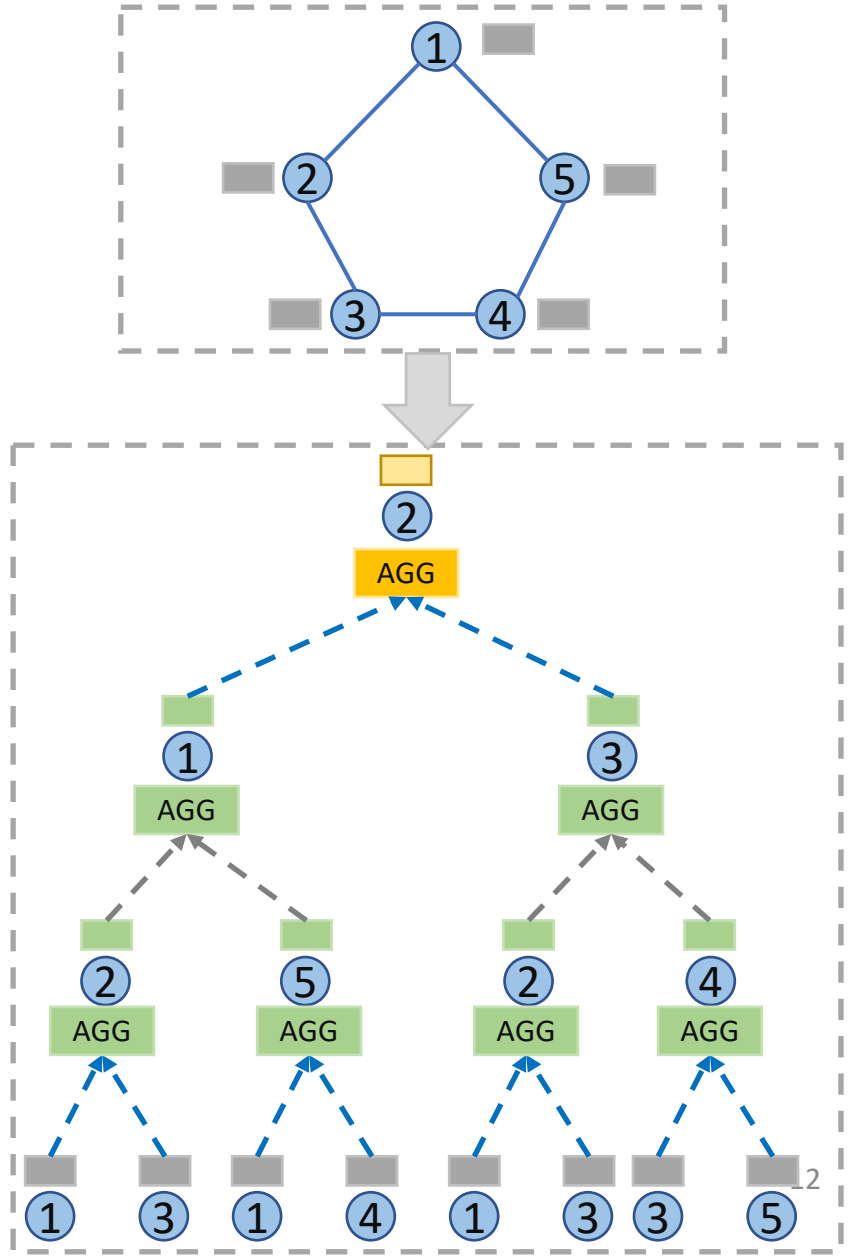
# Graph Neural Network Revisited

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

Self contained Neighborhood

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

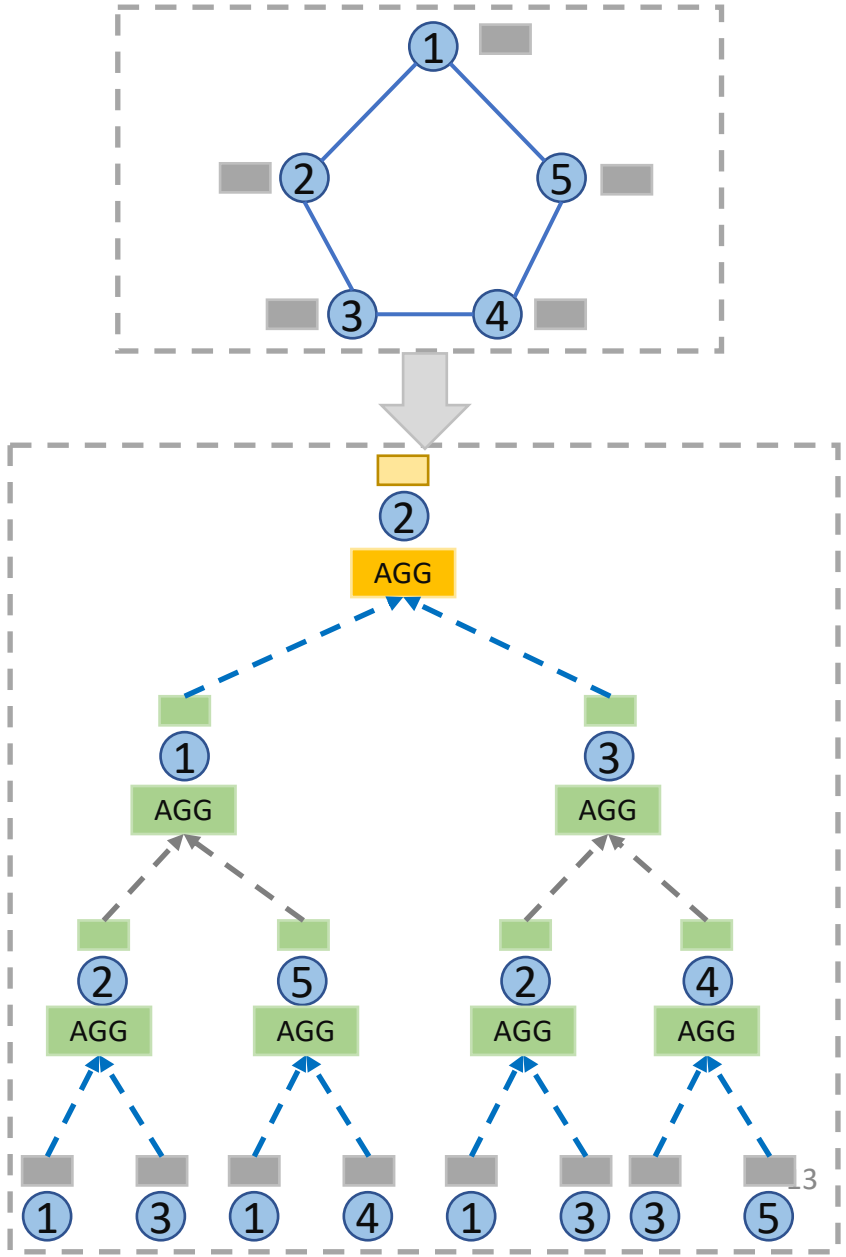
- Variants of GNN
  - GCN: normalized sum aggregator
  - GraphSAGE: MEAN, MAX, SUM, LSTM
  - GAT: Attention aggregator
  - GIN: Multi-Layer Perceptrons (MLP)



# Graph Neural Network Revisited

- Message passing framework
  - Instantiate existing GNN models. [Zhao et al. 2021]

| GNN models | Symbol in the paper | Key explanations |
|---|---|---|
| GCN [14] | 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 [3] | SAGE-MEAN, SAGE-MAX, SAGE-SUM | Apply mean, max, or sum operation to $\{\mathbf{h}_u \vert u \in \widetilde{N}(v)\}$. |
| GAT [4] | GAT | Compute attention score: $\mathbf{e}_{uv}^{gat} = \text{Leaky\_ReLU}\left(\mathbf{a}\left[\mathbf{W}_u\mathbf{h}_u \vert\vert \mathbf{W}_v\mathbf{h}_v\right]\right)$. |
| | 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 [16] | GIN | $F_N^l(v) = \text{MLP}\left((1+\epsilon^{l-1}) \cdot \mathbf{h}_v^{l-1} + \sum_{u \in N(v)} \mathbf{h}_u^{l-1}\right)$. |
| LGCN [15] | CNN | Use 1-D CNN as the aggregator, equivalent to a weighted summation aggregator. |
| GeniePath [18] | GeniePath | Composition of GAT and LSTM-based aggregators |




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

# Design Space for GNN [You et al. 2020]

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

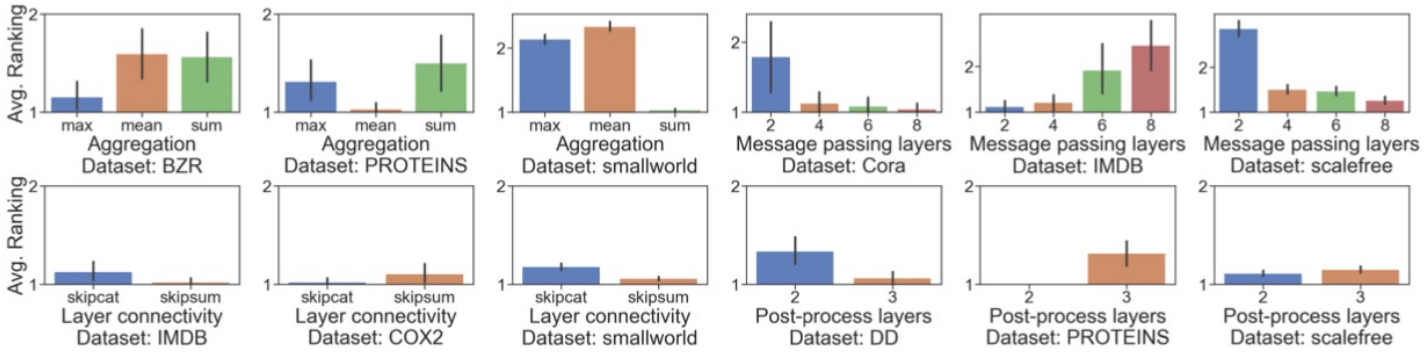
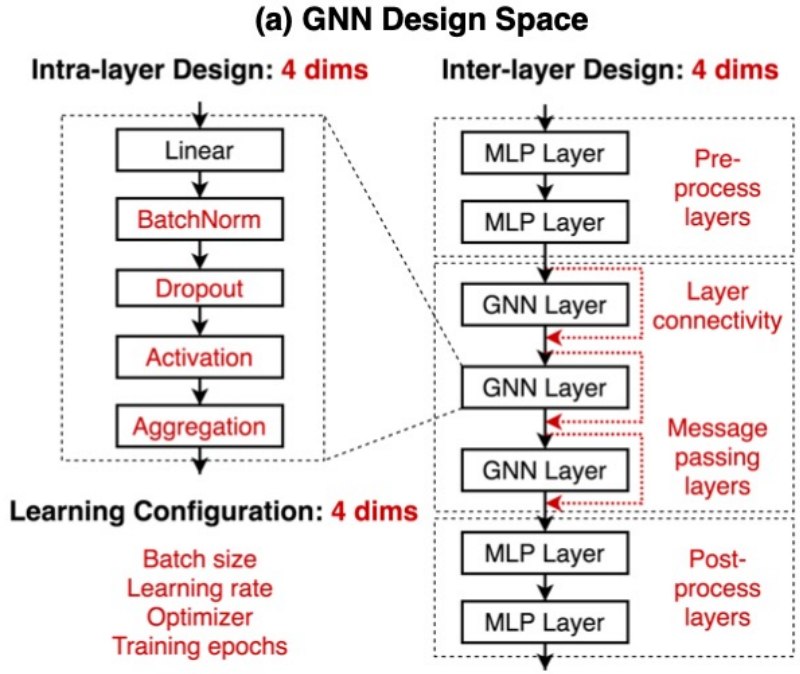



Figure 4: **Ranking analysis for GNN design choices over different GNN tasks**. Lower is better. Preferable design choices greatly vary across GNN tasks.




| 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 |

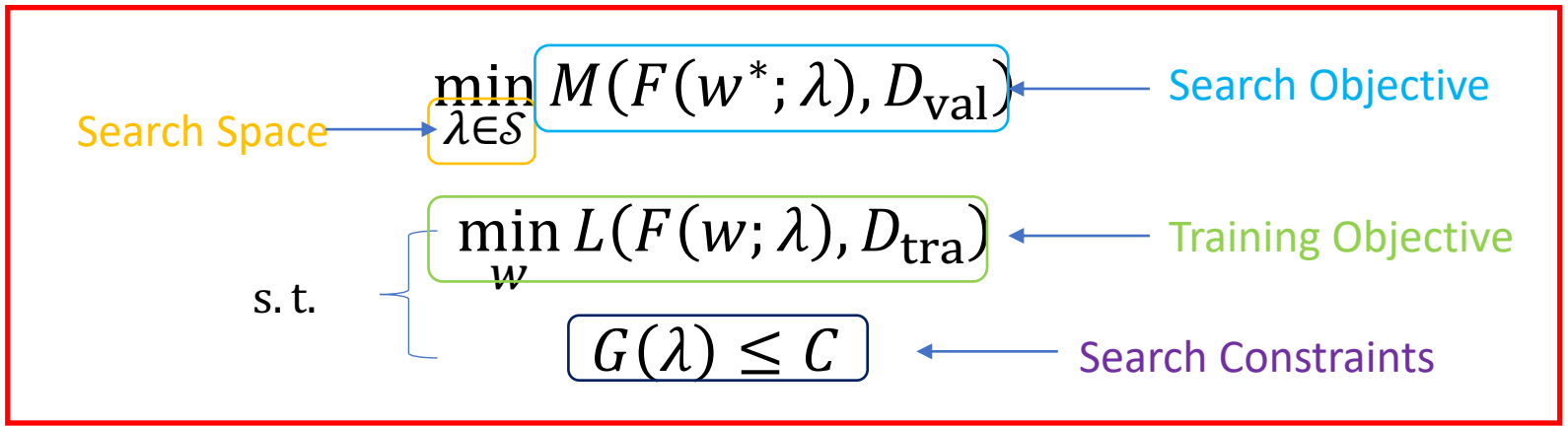


You et al. Design Space for Graph Neural Networks. NeurIPS 2020

# 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



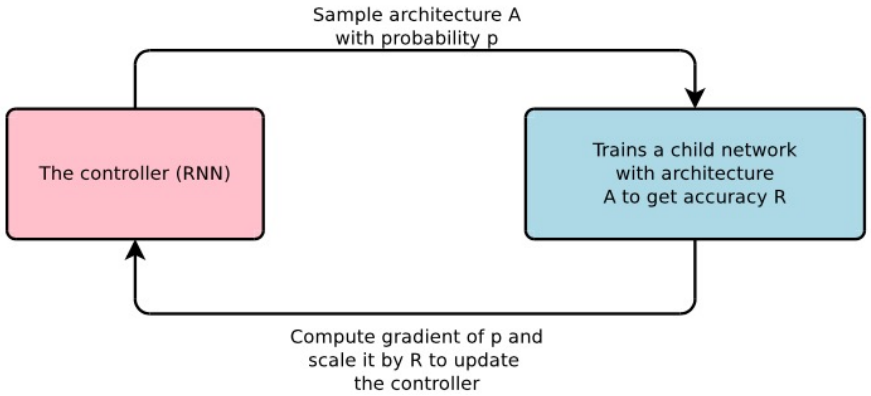



Zoph et al. Neural Architecture Search with Reinforcement Learning. ICLR 2017

# Two representative approaches

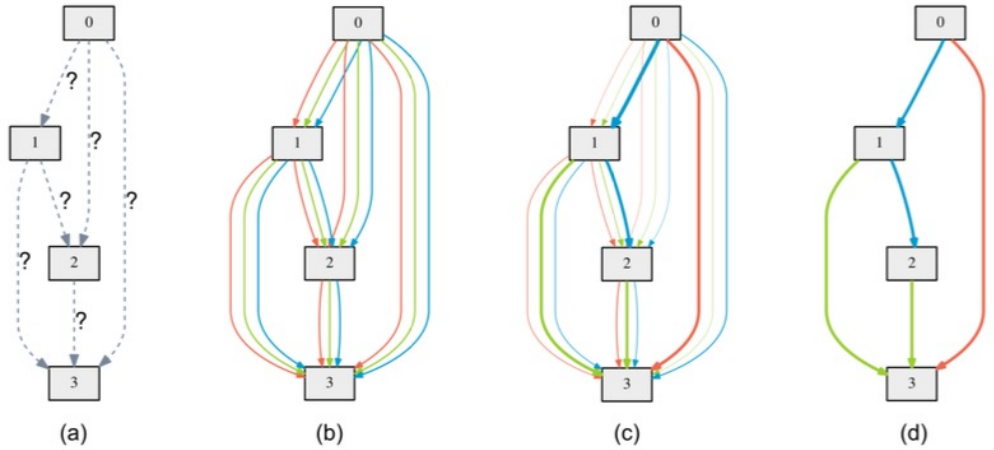
- Stand alone (Reinforcement learning)
  - An RNN controller to sample a candidate architecture
  - Train till convergency
  - Update the RNN controller by the validation accuracy

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]

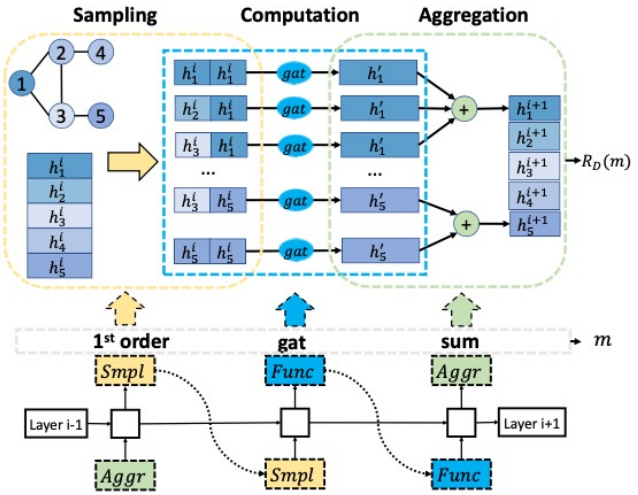



Zoph et al. Neural Architecture Search with Reinforcement Learning. ICLR 2017
Liu et al. Darts: Differentiable architecture search. ICLR 2019

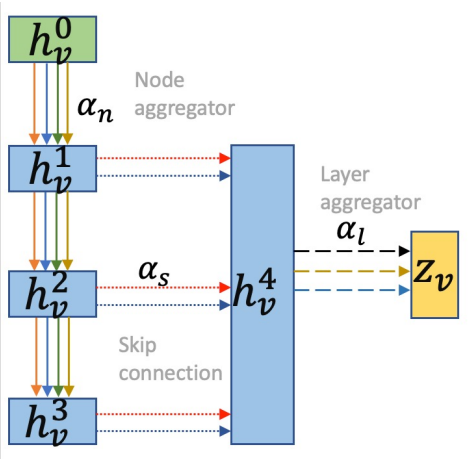
# NAS for GNN

- Two representative approaches
  - Search for combinations of GNN layers.
- GraphNAS [Gao et al. 2020]
  - Reinforcement learning based method
- SANE [Zhao et al. 2021]
  - Differentiable 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

# 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 $\sigma$
  - 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<br>$concat$, otherwise |
| activate function $\sigma$ | $sigmoid$, $tanh$, $relu$, $identity$,<br>$softplus$, $leaky\_relu$, $relu6$, $elu$ |
| 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 | $e_{uv}^{gan} = W_a * tanh(W_l * h_u + W_r * h_v)$ |



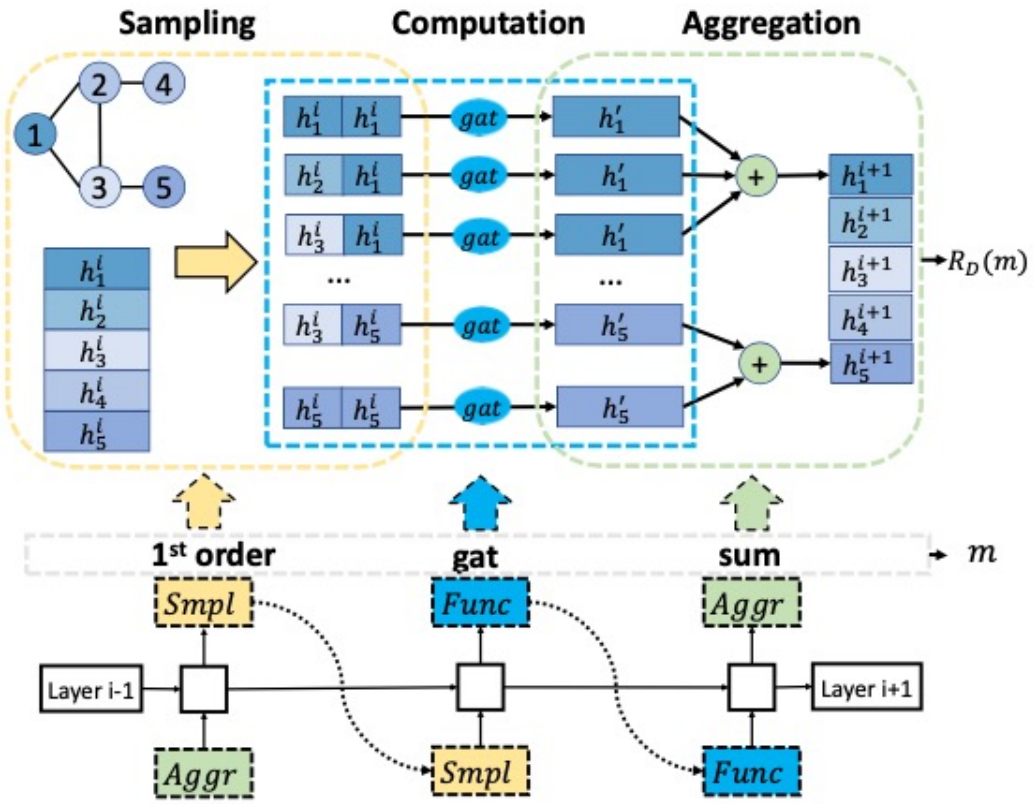
Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

# 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
  - $9{,}408^L$
  - $8 \times 10^7$ When $L = 2$





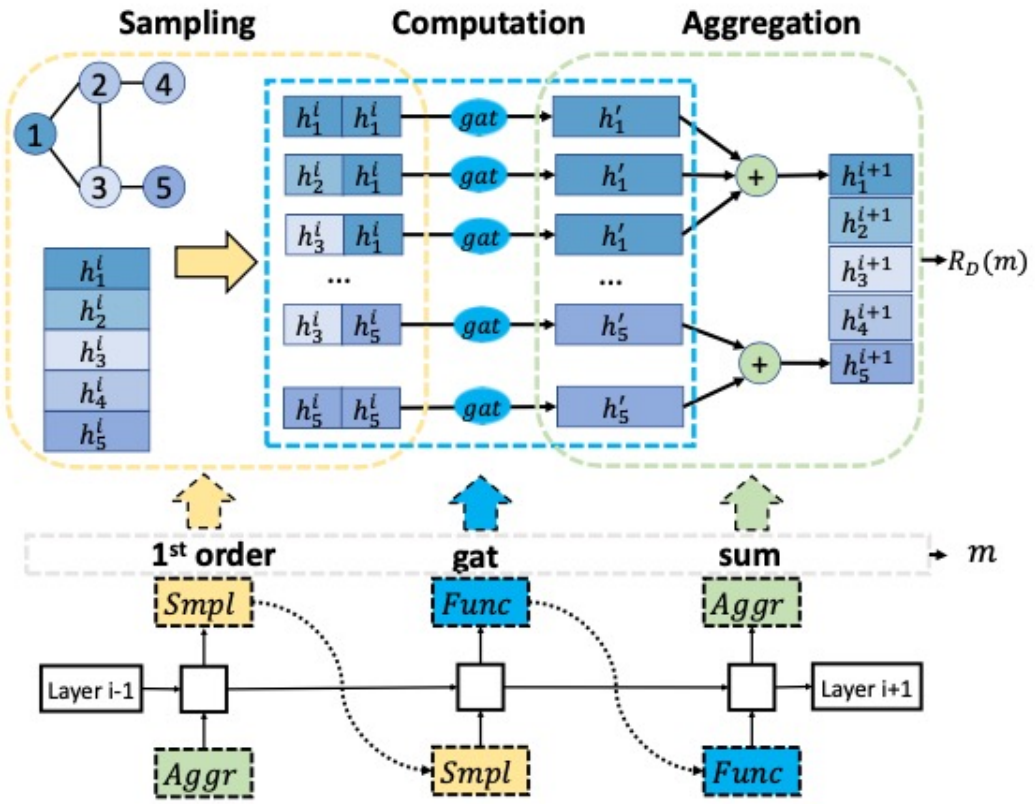

Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

# 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]

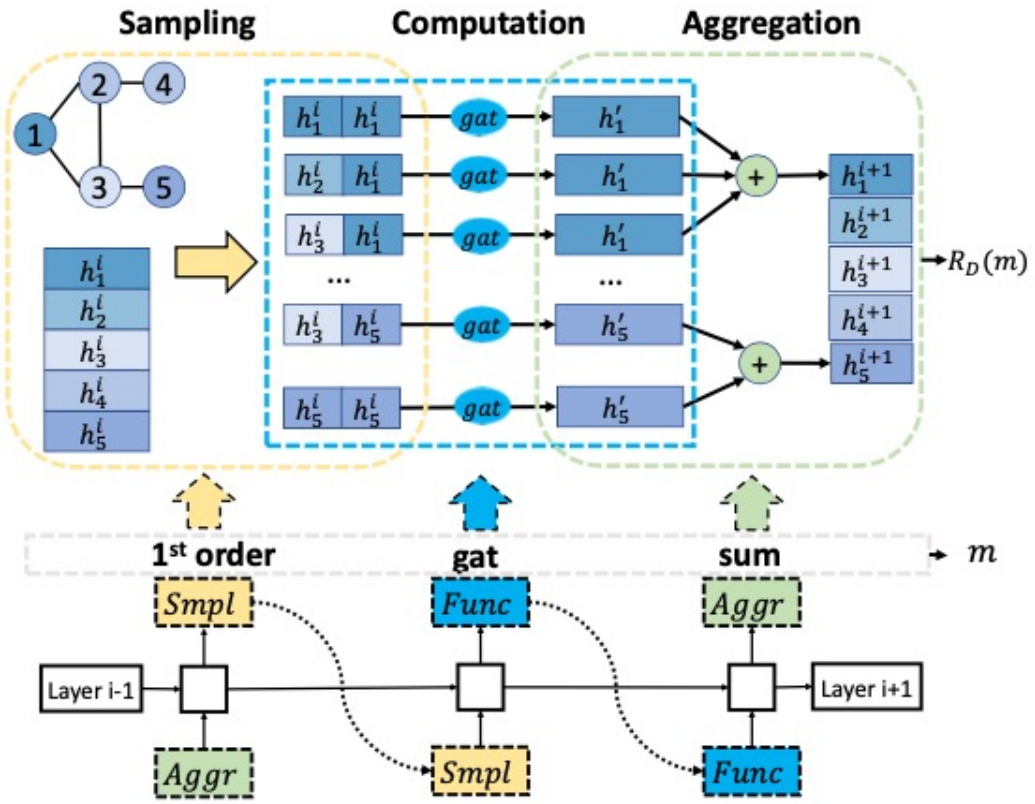
- Search Objective

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

- 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
  - $\theta$ : the parameters of the RNN controller






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

# GraphNAS [Gao et al. 2020]

- How to build deeper GNNs.
  - Recall the size is $9{,}408^L$
  - Extremely expensive when $L$ is large

- Avoid exponential growth of search space
  - Three constraints
    - Independency of each layer
    - Reduce some operators
    - User different message functions for multi-head
- A DAG to represent each GNN layer
  - Five nodes
    - two input(1,2), two intermediate(3,4), one output(5)

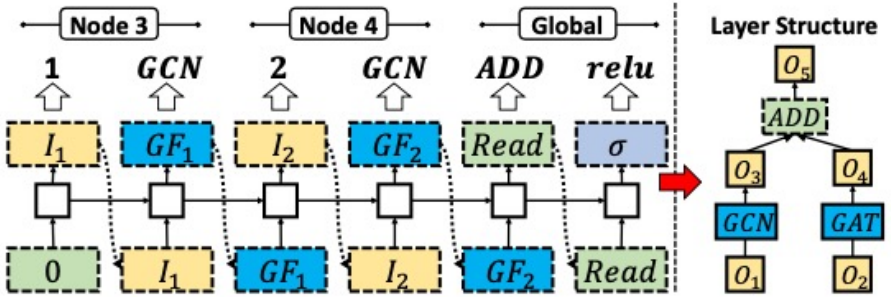



Figure 2: An illustration of GraphNAS constructing a single GNN layer at the right-hand side. The layer has two input states $O_1$ and $O_2$, two intermediate states $O_3$ and $O_4$, and an output state $O_5$. The controller at the left-hand side samples $O_2$ from $\{O_1, O_2, O_3\}$ and take $O_2$ as the input of $O_4$, and then samples $GAT$ for processing $O_2$. The output state $O_5 = relu(O_3 + O_4)$ collects information from $O_3$ and $O_4$, and the controller assigns a readout operator $add$ and an activation operator $relu$ for $O_5$. As a result, this layer can be described as a list of operators: [1,$gcn$, 2, $gat$, $add$, $relu$].

$$O_{out} = \sigma(Read(O_i|3 \leqslant i \leqslant B+2)),$$



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.

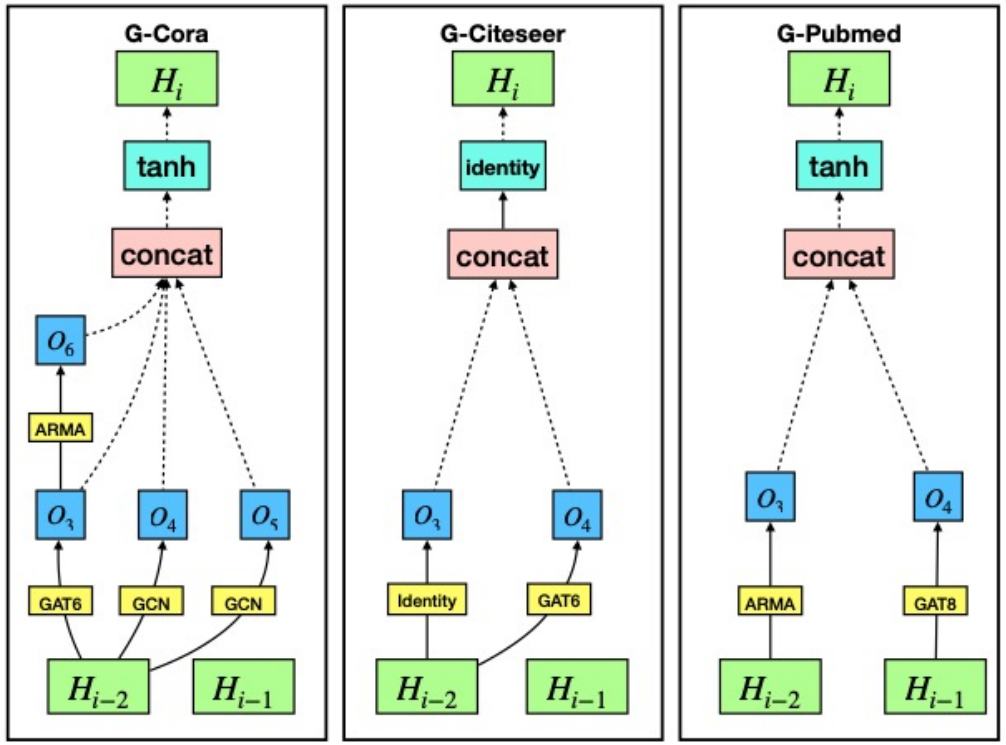


Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020

# 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

# Experiments

- More Results
  - Transfer learning
  - Influence of search epochs.

| Model | CS | Physics | Computers | Photo |
|---|---|---|---|---|
| GCN | 95.5±0.3 | 98.3±0.2 | 88.0±0.6 | 95.4±0.3 |
| GAT | 95.5±0.3 | 98.1±0.2 | 89.1±0.6 | 95.6±0.3 |
| ARMA | 95.4±0.2 | **98.5±0.1** | 86.1±1.0 | 94.8±0.8 |
| APPNP | 95.6±0.2 | **98.5±0.1** | 89.8±0.4 | 95.8±0.3 |
| GraphNAS | **97.1±0.2** | **98.5±0.2** | **92.0±0.4** | **96.5±0.4** |

Table 4: Transferring architectures designed by GraphNAS on the citation networks to the other four datasets

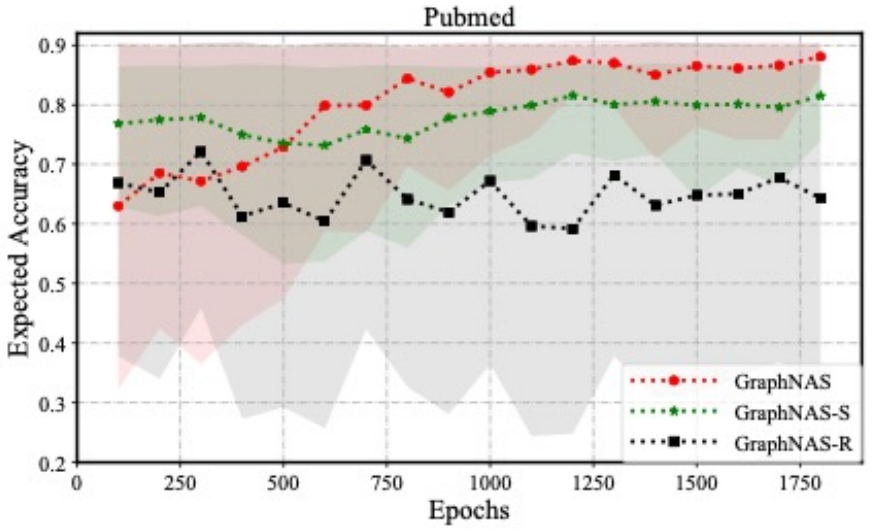



Figure 4: Comparisons *w.r.t.* the number of search epochs on Pubmed. The expected accuracy of the architecture designed by GraphNAS raises with the search epochs. GraphNAS outperforms Simple-NAS after 500 epochs.



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]

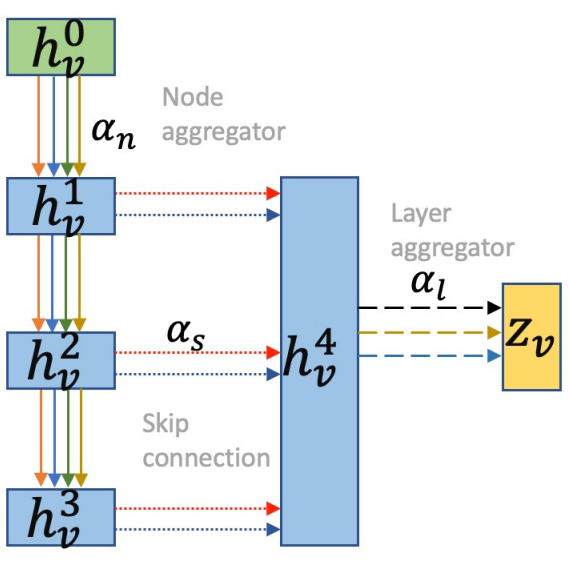
- SOTA performance

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

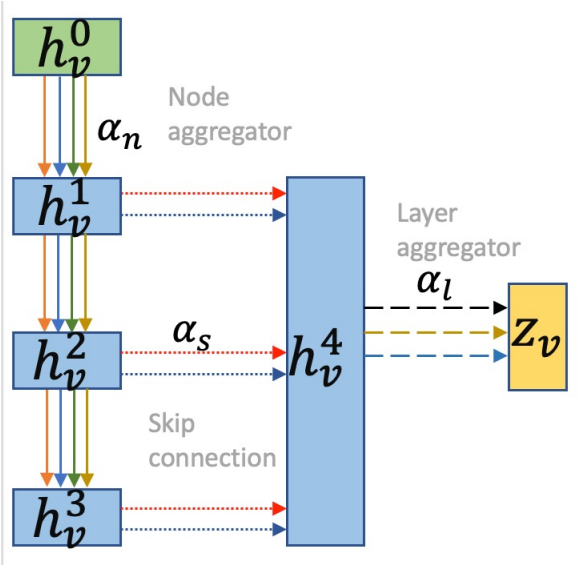
| GNN models | Symbol in the paper | Key explanations |
|---|---|---|
| GCN [15] | GCN | $F_N^l(v) = \sum_{u \in \widetilde{N}(v)} \left(\text{degree}(v) \cdot \text{degree}(u)\right)^{-1/2} \cdot \mathbf{h}_u^{l-1}$. |
| GraphSAGE [15] | SAGE-MEAN, SAGE-MAX, SAGE-SUM, SAGE-LSTM | Apply mean, max, sum, 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

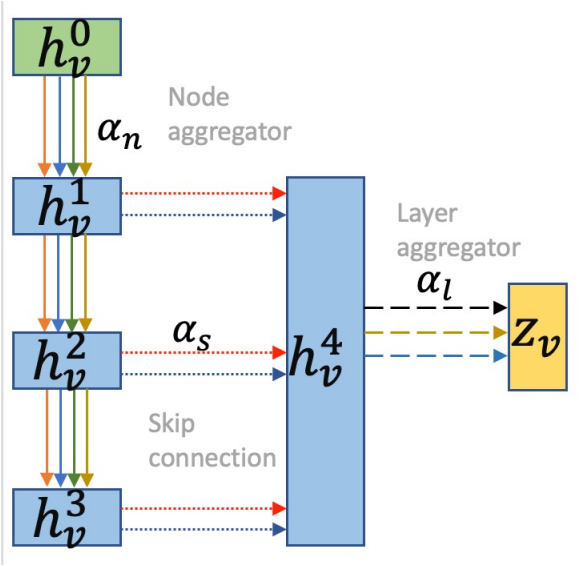
# Differentiable search algorithm

- 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

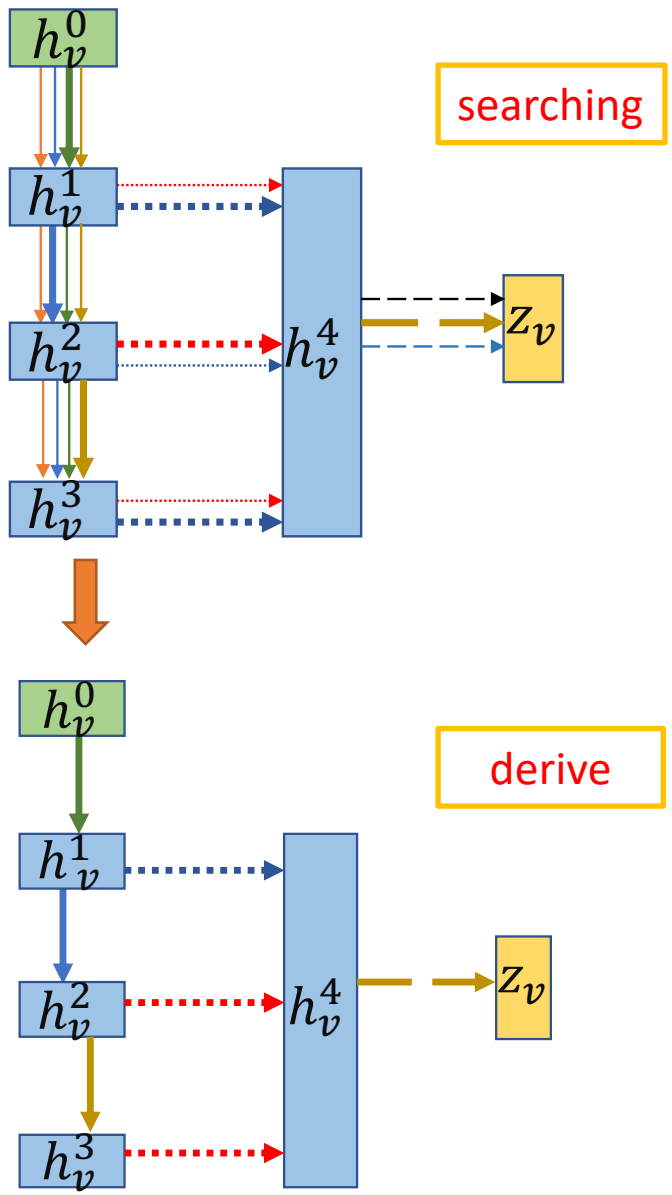
- Gradient-based optimization.

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

**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

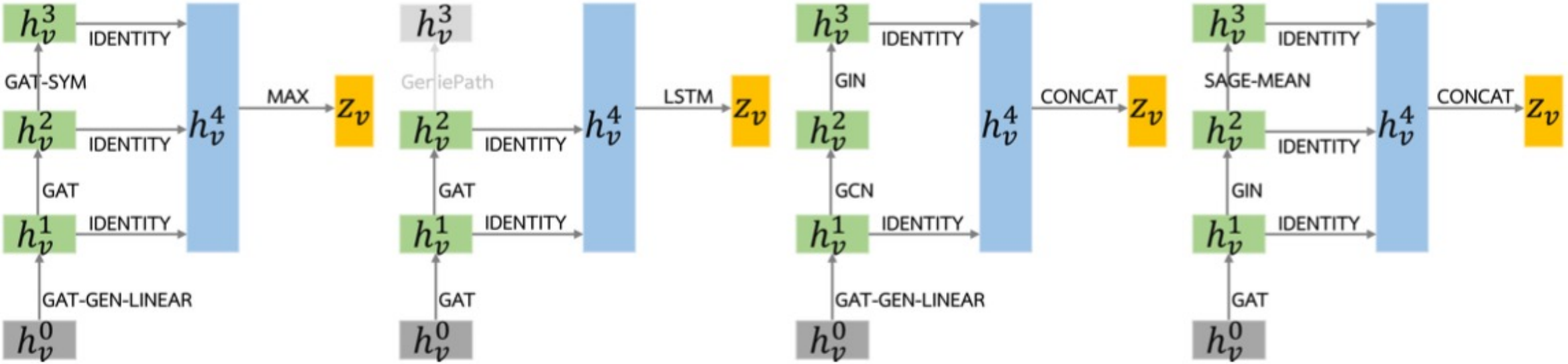
- Performance comparison
  - Human-designed architectures
  - NAS approaches

| | | 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

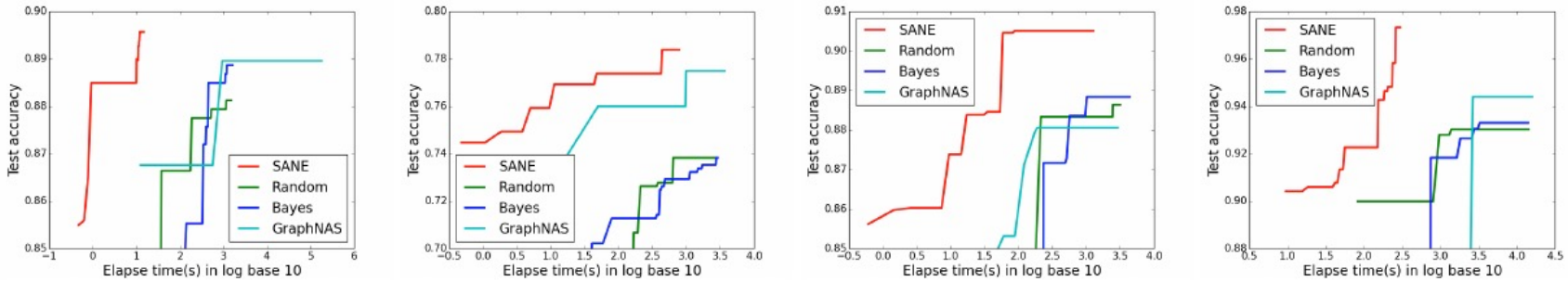



(a) Cora. (b) CiteSeer. (c) PubMed. (d) PPI.

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.

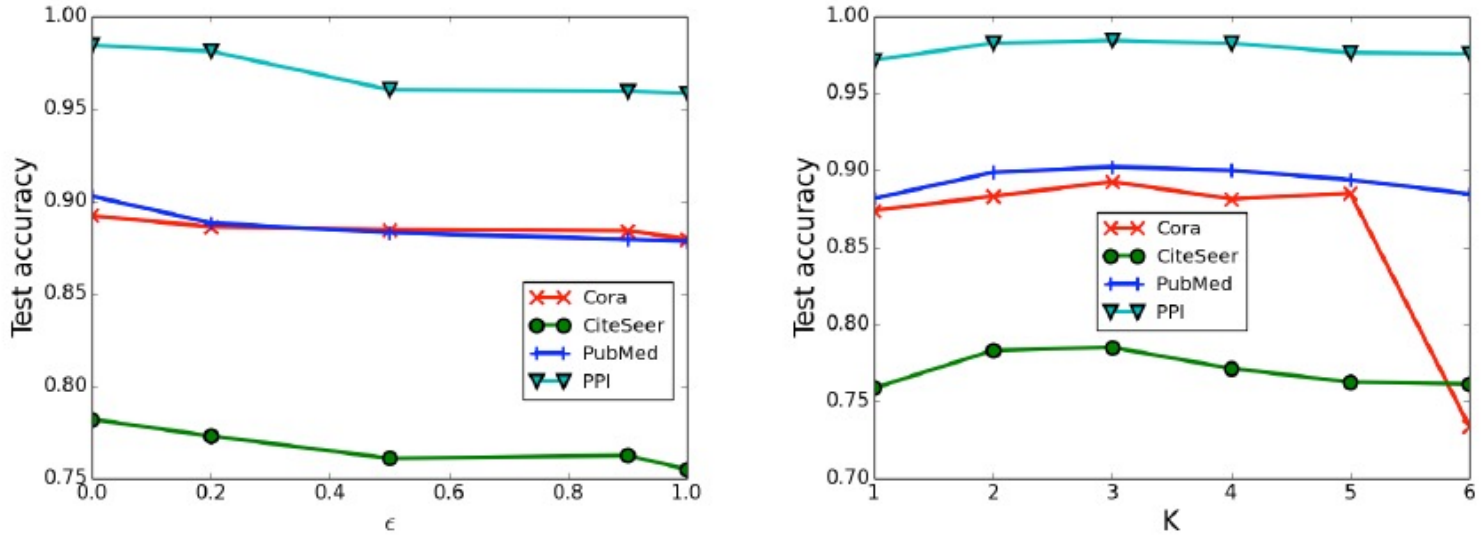
- The total time (s) of running once of each model

| | 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

# Experiments

- Ablation study
  - The influence of differentiable search
  - The influence of K
- The test accuracy w.r.t. different $\epsilon$ and $K$ for transductive and inductive tasks.




(a) Random explore: $\epsilon$. (b) Number of GNN layers: $K$.

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.

- More following works can be checked in [Zhang et al. 2021] .

# Neural architecture search for GNN based CF

# NAS for GNN based CF

- It is straightforward to apply GraphNAS and SANE to GNN based CF.
  - Adapt to edge-level task, i.e., link prediction.

- May need to take into consideration domain-specific challenges
  - Capture higher-order information effectively
    - Alleviate the over-smooth problem when stacking GNN layers.
  - Complex users' behaviors (multiple interests, multiple types of behaviors, etc.)
    - Multiplex graph [Gao et al. 2020]
  - Modeling rich side information beyond user-item interaction matrix
    - Heterogeneous graph based RS [Zhao et al. 2017]

Gao et al. Multi-behavior recommendation with graph convolutional networks. SIGIR 2020.
Zhao et al. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. KDD 2017

# Potential design

- Search space
  - Try to include as many as possible human-designed architectures for CF.
  - Keep the search space in a moderate size.

- Search algorithm
  - Stand alone or one-shot
  - Can try both

- We leave it for upcoming work

# Conclusion

- Recommender System (RS)
  - Collaborative Filtering (CF)
  - Graph Neural Network (GNN) based CF
- Neural architecture search (NAS) for GNN
  - Reinforcement learning (GraphNAS)
  - Differentiable architecture search (SANE)
- Discussion on neural architecture search for GNN based CF
  - Potential domain-specific challenges & design

# References

1. Ying et al. Graph convolutional neural networks for web-scale recommender systems. KDD 2018
2. Wang et al. Neural Graph Collaborative Filtering. SIGIR 2019.
3. Wu et al. Graph neural networks in recommender systems: a survey. 2020.
4. You et al. Design Space for Graph Neural Networks. NeurIPS 2020
5. Zoph et al. Neural Architecture Search with Reinforcement Learning. ICLR 2017
6. Liu et al. Darts: Differentiable architecture search. ICLR 2019
7. Gao et al. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020
8. Zhao et al. Search to aggregate neighborhood for graph neural networks. ICDE 2021
9. Hu et al. How Powerful are Graph Neural Networks? ICLR 2019
10. Xu et al. Representation Learning on Graphs with Jumping Knowledge Networks. ICML 2018
11. Zhang et al. Automated Machine Learning on Graphs: A Survey. IJCAI 2021
12. Zhao et al. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. KDD 2017
13. Gao et al. Multi-behavior recommendation with graph convolutional networks. SIGIR 2020.

Automated Recommender System (RecSys) Tutorial

# Part 3: Automated Graph Neural Network for Recommender System

Official site: https://quanmingyao.github.io/AutoML.github.io/ijcai21-tutorial.html

# Thanks!

# Q&A

Code and slides can be accessed in my homepage
https://hzhaoaf.github.io/

Dr. Huan Zhao
Senior researcher in 4Paradigm