Tutorial outline

1. What is Automated machine learning (AutoML) - A retrospective view
   – Dr. Quanming Yao (4Paradigm) – 50mins talk + 10min break
2. Recommender System: Basic and Why AutoML is Needed?
   – Prof. Yong Li (Tsinghua) – 35mins + 5 mins break
3. Recent Advances in Automated Recommender System
   – Mr. Chen Gao (Tsinghua) – 35mins + 5 mins break
4. Automated Graph Neural Network for Recommender System
   – Dr. Huan Zhao (4Paradigm) – 35mins + 5 mins break
5. Automated Knowledge Graph Embedding
   – Dr. Yongqi Zhang (4Paradigm) – 35mins + 5 mins break

Automated Graph Neural Network for Recommendation System

Huan Zhao, Lanning Wei, Quanming Yao
Machine Learning Research Group, 4Paradigm
Aug. 24th 2020

Outline

- Introduction
  - Graph Neural Network (GNN)
  - GNN for recommendation
  - Automated machine learning (AutoML) for GNN

- Neural Architecture Search (NAS) for GNN
  - Reinforcement learning
  - Differentiable architecture search

- Automated GNN for recommendation

Graph Neural Network

- GNN is a very hot topic in recent years
  - Representation learning in graphs
  - Define "convolution" on non-grid data

- Applications
  - Recommendation [Ying et al. KDD 2018]
  - Fraud Detection [Liu et al. AAAI 2019]
  - Spam detection [Li et al. CIKM 2019]
  - Bioinformatics [Zitnik et al. Bioinformatics 2017]

Graph Neural Network

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

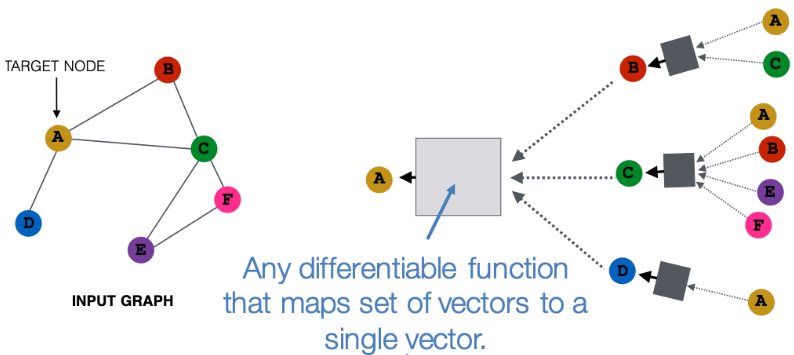
Self contained Neighborhood

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

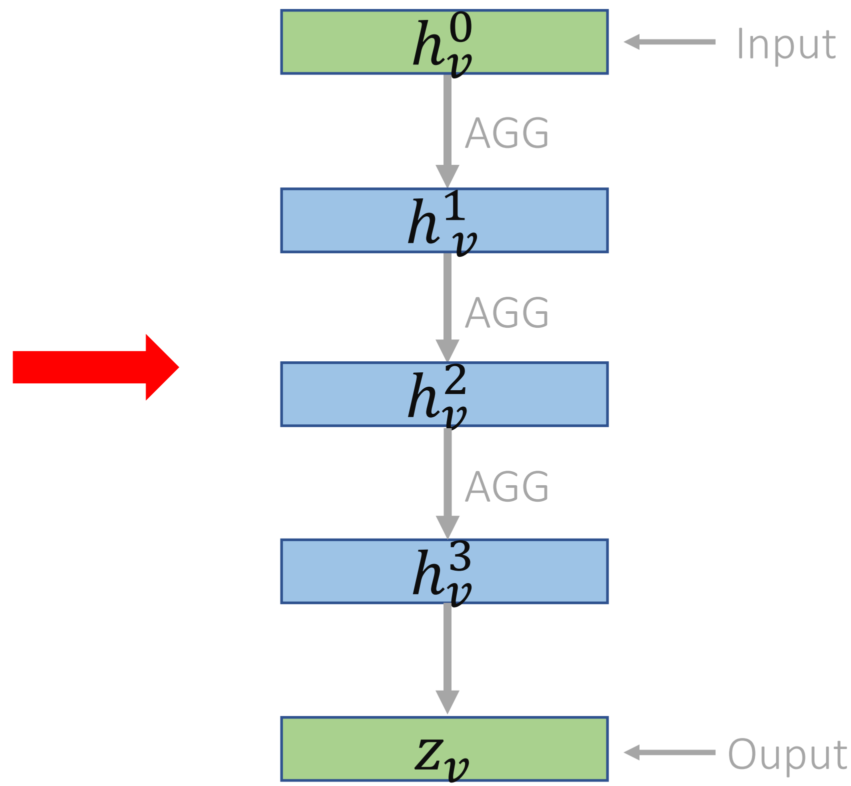
- Variants of GNN
  - GCN: normalized sum aggregator
  - GraphSAGE: MEAN, MAX, SUM, LSTM
  - GAT: Attention aggregator
  - GIN: Multi-Layer Perceptrons (MLP)

Graph Neural Network

- An example GNN architecture.



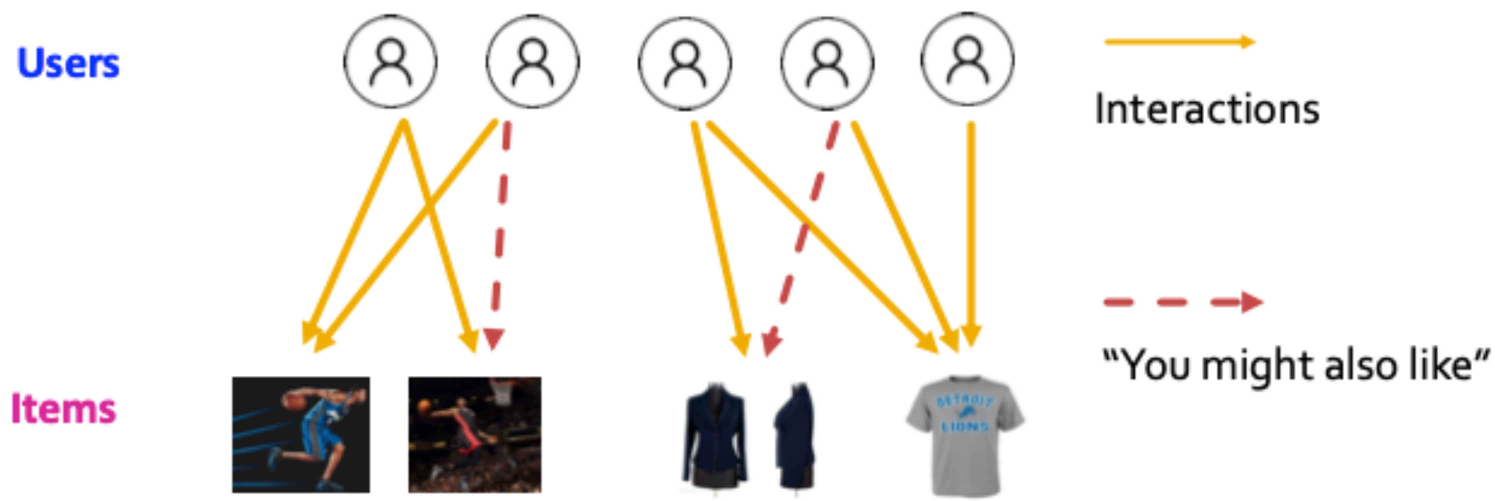
$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{A}_k \cdot \boxed{\mathrm{AGG}(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\})}, \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$



Leftside picture from http://snap.stanford.edu/proj/embeddings-www/

GNN for Recommendation

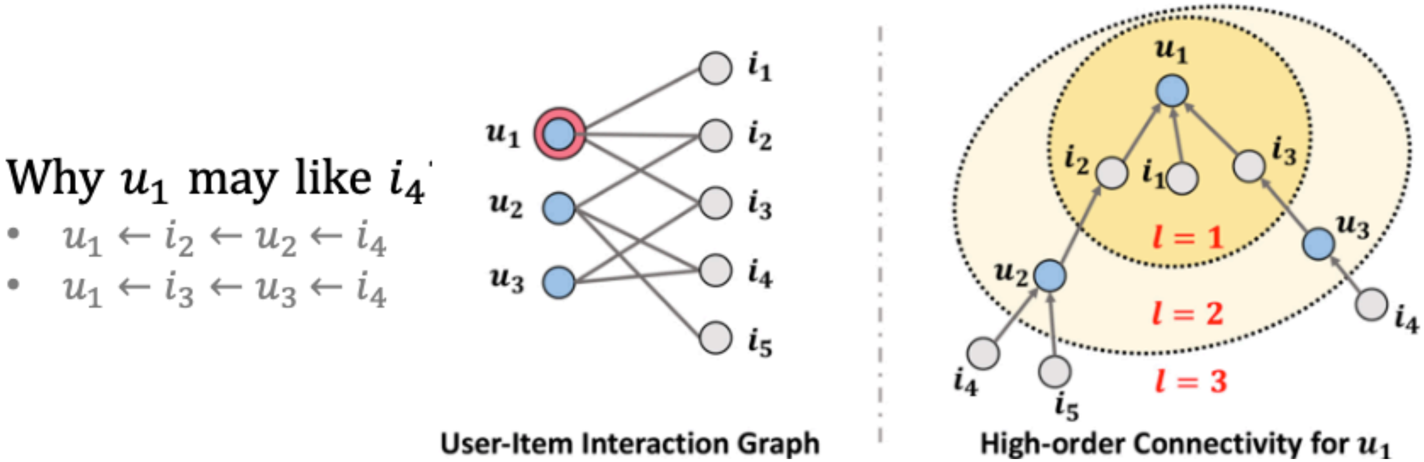
- Recommendation can be naturally modeled by graph.
  - Bipartite graph for user-item interactions.



Picture from http://web.stanford.edu/class/cs224w/slides/19-applications.pdf

GNN for Recommendation

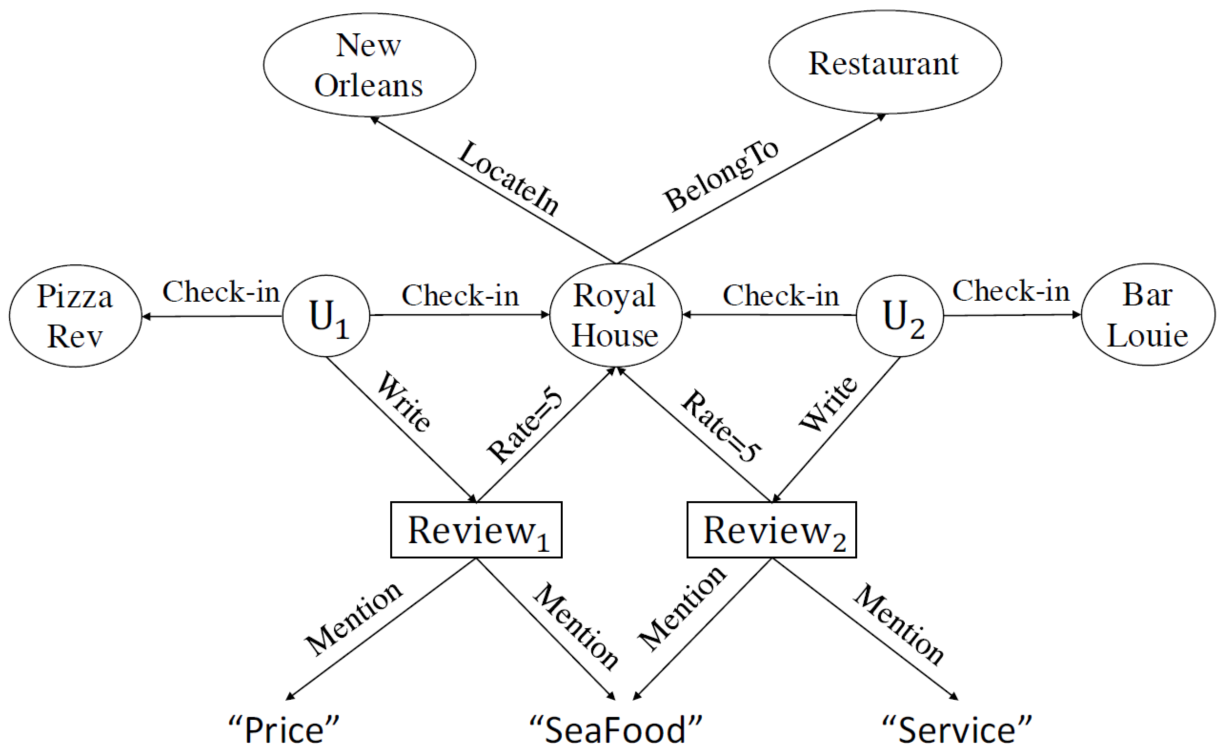
- Recommendation can be naturally modeled by graph.
  - Neural Graph collaborative filtering (NGCF)



Wang et al. Neural Graph Collaborative Filtering. SIGIR 2019

GNN for Recommendation

- Recommendation can be naturally modeled by graph.
  – Heterogeneous graph for rich side information



Zhao et al. Side Information Fusion for Recommender Systems over Heterogeneous Information Network. Arxiv 2020

GNN for Recommendation

- Recommendation can be naturally modeled by graph.
  - Similarity computation between user and item
  - User and item representation by GNN.
    - Scenario-specific recommendation models are useful.
      - Data-
    - AutoML can help.

Neural Architecture Search (NAS) can be exploited.

AutoML for GNN

- What can be searched ?
  - Revisiting message passing framework
    - the l-th layer embedding of node v:

Permutation invariant function

Adjacency matrix

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

Path of length L

Hyper-parameters

Neural Architecture Search (NAS) can be exploited.

NAS for GNN

- Background
  - Graph Neural Network (GNN)
  - Neural Architecture Search (NAS)

- Two methods
  - Reinforcement Learning
  - Differentiable architecture search

- Discussion

NAS for GNN

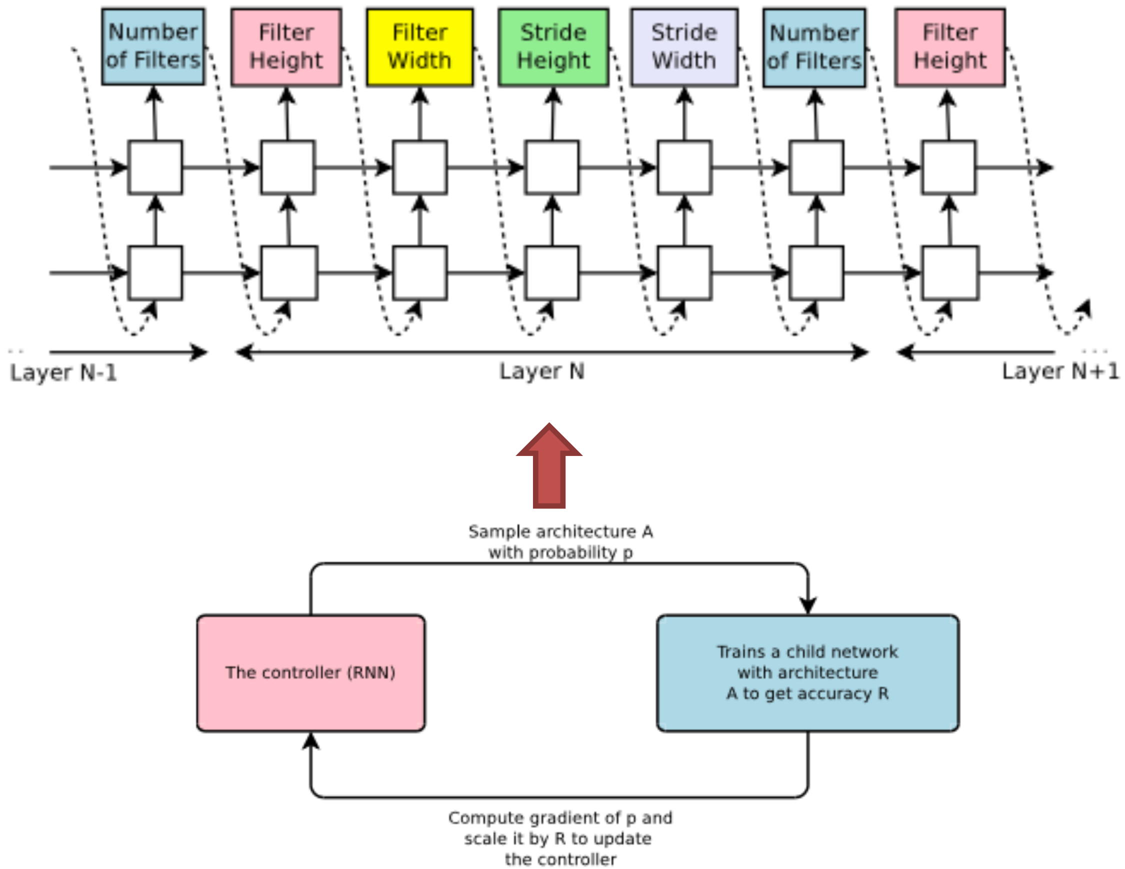
- Background
  - Graph Neural Network (GNN)
  - Neural Architecture Search (NAS)
- Two methods
  - Reinforcement Learning
  - Differentiable architecture search
- Discussion

Neural Architecture Search

- Neural Architecture Search (NAS)
  - Exploring the possibility of automatically searching for unexplored architectures beyond human-designed ones.
  - Can obtain data-specific models.

NAS

- Trial and Error
  - Iteratively train and evaluate the candidate architectures, keeping tracking of the best ones



Zoph et al. NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING. ICLR 2017

NAS for GNN

- Motivation
  - Existing challenges of GNN
    - Architecture challenge
    - Efficiency challenge

- How can NAS help ?
  - Automatically search for unexplored and optimal GNN architectures.
  - Transfer searched architecture from smaller graphs to larger graphs.

NAS for GNN

- Background
  - Graph Neural Network (GNN)
  - Neural Architecture Search (NAS)
- Two methods
  - Reinforcement Learning (RL)
  - Differentiable architecture search
- Discussion

RL-based NAS

- RL framework
  - Sample a child model and train from scratch.
  - Update based on the validation accuracy (reward)

- Existing works
  - GraphNAS [Gao et al. 2020]
  - Auto-GNN[Zhou et al. 2019]
  - Policy-GNN[Zhou et al. 2020]

GraphNAS [Gao et al. 2020]

- Search Space
  - Different components

Actions	Contents
SAM	Sample neighbors
ATT	Const, gcn, gat, sym-gat, cos
AGG	Sum, mean, max, mlp
Heads K	1, 2, 4, 8, 16
DIM	16, 32, 64, 128
ACT	Relu, elu, tanh, linear, sigmoid

  - Combine with Message Passing

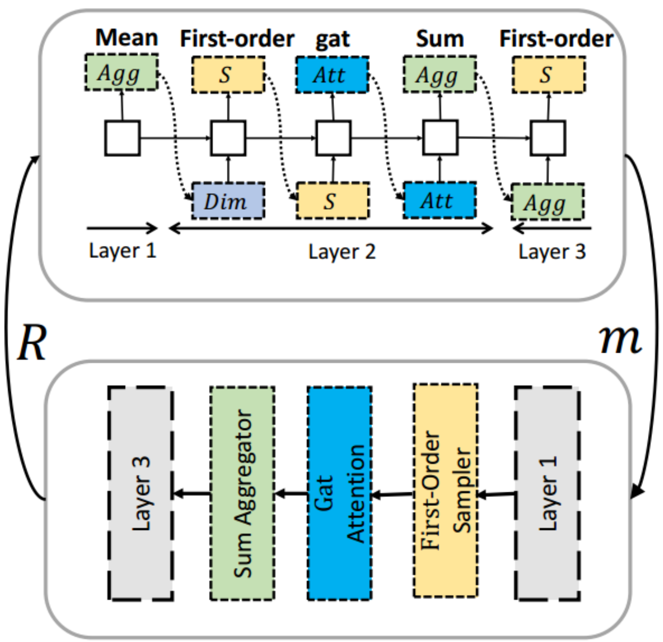
$$\mathrm{m}_{\mathrm{v}}^{t+1} = \boxed{\sum}_{\omega \epsilon N(v)} \boxed{M_t}(h_v^t, h_\omega^t, \boxed{e_{v\omega}}) \qquad h_v^{t+1} = \boxed{U_t}(h_v^t, \mathrm{m}_{\mathrm{v}}^{t+1})$$

[AGG , SAM] DIM [ATT , Heads K] ACT

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

GraphNAS [Gao et al. 2020]

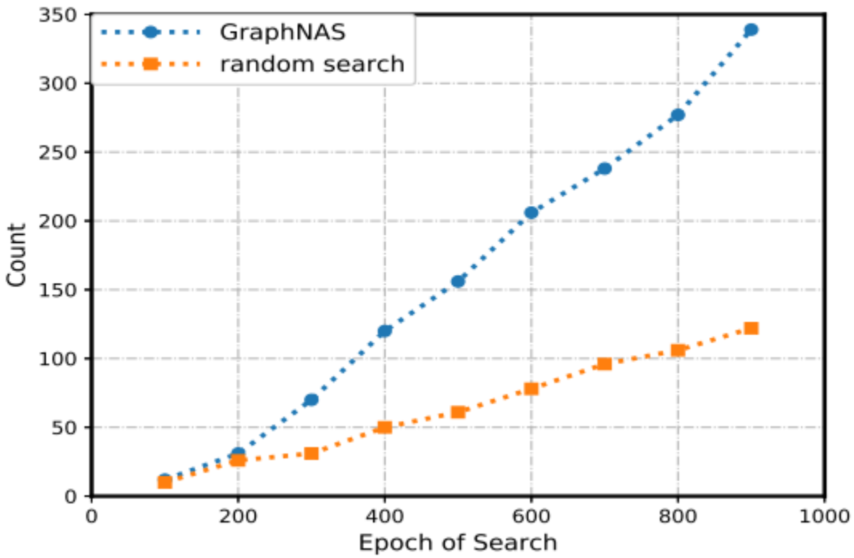
- Search algorithm
  - RNN generates child architectures
  - Train child models and get valid acc
  - Update RNN parameters.



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

GraphNAS [Gao et al. 2020]

- Experiment results



Model	Cora	Citeseer	Pubmed
GCN	81.5+/-0.4	70.9+/-0.5	79.0+/-0.4
SGC	81.0+/-0.0	71.9+/-0.1	78.9+/-0.0
GAT	83.0+/-0.7	72.5+/-0.7	79.0+/-0.3
LGCN	83.3+/-0.5	73.0+/-0.6	79.5+/-0.2
DGCN	82.0+/-0.2	72.2+/-0.3	78.6+/-0.1
ARMA	82.8+/-0.6	72.3+/-1.1	78.8+/-0.3
APPNP	83.3+/-0.6	71.8+/-0.4	80.2+/-0.2
simple-NAS	81.4+/-0.6	71.7+/-0.6	79.5+/-0.5
GraphNAS	**84.3+/-0.4**	**73.7+/-0.2**	**80.6+/-0.2**

Left: Count models whose acc over 0.81 on Cora datasets
Right: Semi-supervised node classification w.r.t. accuracy

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

Auto-GNN [Zhou et al. 2019]

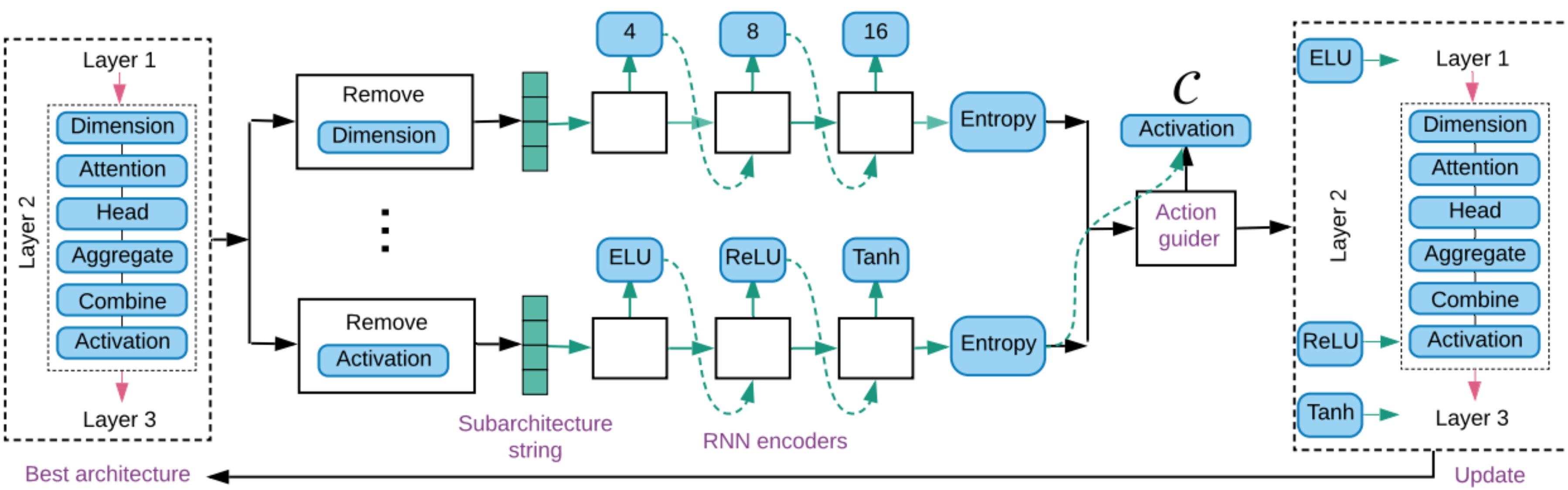
- Search space
  - Different components

Actions	Contents
ATT	Const, gcn, gat, cos, linear
AGG	Sum, mean, max
Heads K	1,2,4,6,8,16
DIM	4,8,16,32,64,128,256
ACT	Relu, elu, tanh, linear, sigmoid

Gao et al. Auto-GNN: Neural Architecture Search of Graph Neural Networks. Arxiv 2019

Auto-GNN [Zhou et al. 2019]

- Search algorithm
  - Modification based on a well-performed architecture.
  - Modify architecture
    - Individual RNN controller for each class
    - Action guider selects a class with decision entropy
    - Use policy gradient to update parameters.

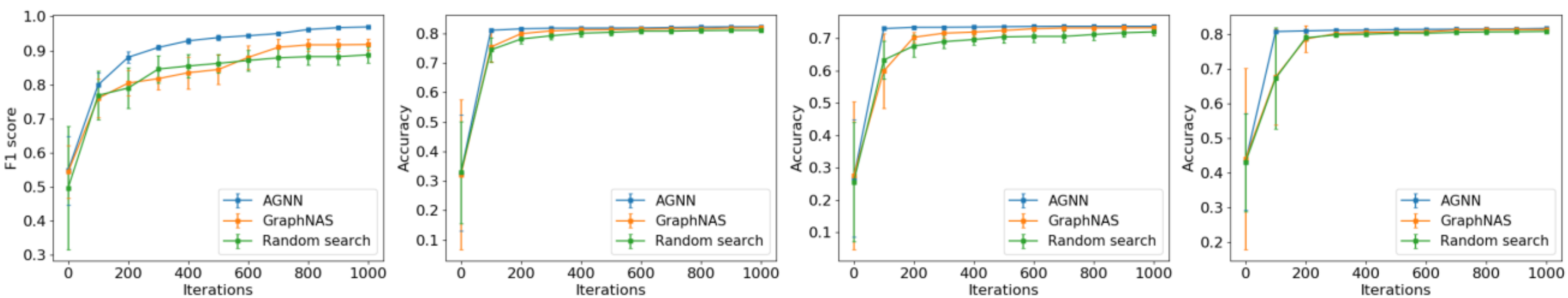


Gao et al. Auto-GNN: Neural Architecture Search of Graph Neural Networks. Arxiv 2019

Auto-GNN [Zhou et al. 2019]

- Experiment results

Baseline Class	Model	#Layers	Cora		Citeseer		Pubmed	
			#Params	Accuracy	#Params	Accuracy	#Params	Accuracy
Handcrafted Architectures	Chebyshev	2	0.09M	81.2%	0.09M	69.8%	0.09M	74.4%
	GCN	2	0.02M	81.5%	0.05M	70.3%	0.02M	79.0.5%
	GAT	2	0.09M	83.0 ± 0.7%	0.23M	72.5 ± 0.7%	0.03M	79.0 ± 0.3%
	LGCN	3 ~ 4	0.06M	83.3 ± 0.5%	0.05M	73.0 ± 0.6%	0.05M	79.5 ± 0.2%
NAS Baselines	GraphNAS-w/o share	2	0.09M	82.7 ± 0.4%	0.23M	73.5 ± 1.0%	0.03M	78.8 ± 0.5%
	GraphNAS-with share	2	0.07M	83.3 ± 0.6%	1.91M	72.4 ± 1.3%	0.07M	78.1 ± 0.8%
	Random-w/o share	2	0.37M	81.4 ± 1.1%	0.95M	72.9 ± 0.2%	0.13M	77.9 ± 0.5%
	Random-with share	2	2.95M	82.3 ± 0.5%	0.95M	69.9 ± 1.7%	0.13M	77.9 ± 0.4%
AGNN	AGNN-w/o share	2	0.05M	**83.6 ± 0.3%**	0.71M	**73.8 ± 0.7%**	0.07M	**79.7 ± 0.4%**
	AGNN-with share	2	0.37M	82.7 ± 0.6%	1.90M	72.7 ± 0.4%	0.03M	79.0 ± 0.5%

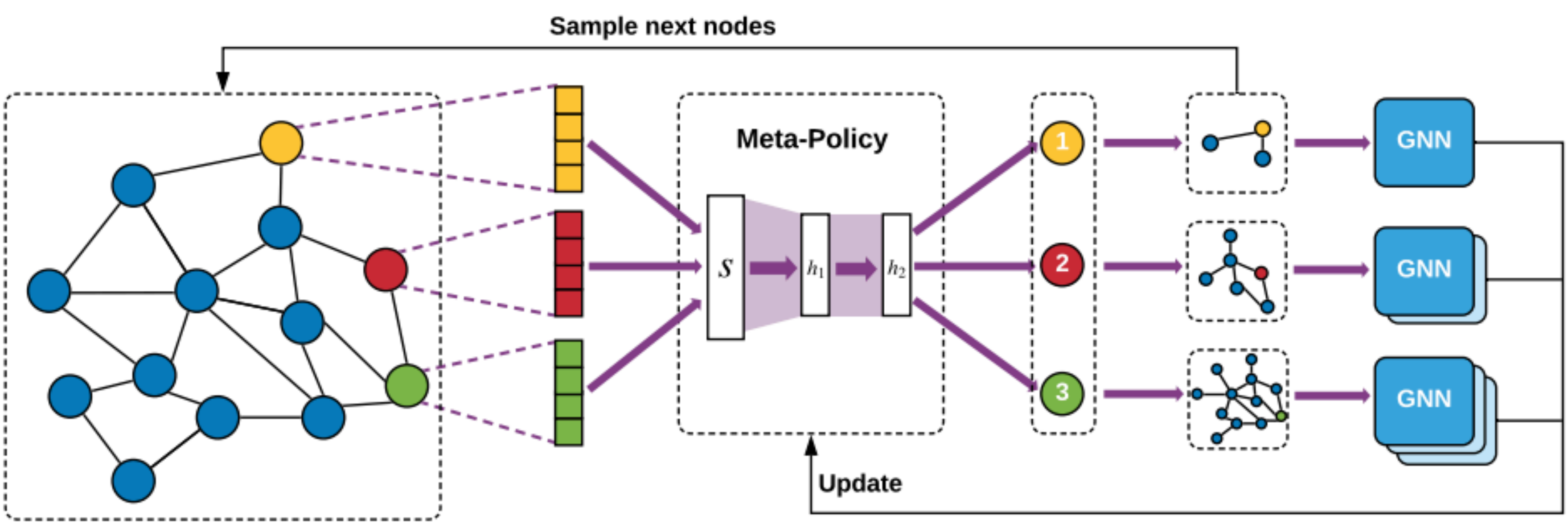


(a) PPI (b) Cora (c) Citeseer (d) Pubmed

Gao et al. Auto-GNN: Neural Architecture Search of Graph Neural Networks. Arxiv 2019

Policy-GNN [Lai et al. 2020]

- GraphNAS and Auto-GNN assumes a pre-defined number of layers for all nodes in GNN.
- Policy-GNN
  - Search for the number of layers per node based on node attributes.



Lai et al. Policy-GNN: Aggregation Optimization for Graph Neural Networks . SIGKDD 2020

Policy-GNN [Lai et al. 2020]

- Search for the number of hops per node based on node attributes.
  - Performance gain is significant.

Baseline Class	Model	#Layers	Cora		Citeseer		Pubmed	
			Accuracy	↑	Accuracy	↑	Accuracy	↑
Network Embedding	DeepWalk [32]	-	0.672	+36.8%	0.432	+107.6%	0.653	+41.0%
	Node2vec [10]	-	0.749	+22.7%	0.547	+64.0%	0.753	+22.3%
Static-GNNs	Chebyshev [5]	2	0.812	+13.2%	0.698	+28.5%	0.744	+23.8%
	GCN [20]	2	0.815	+12.8%	0.703	+27.6%	0.790	+16.6%
	GraphSAGE [14]	2	0.822	+11.8%	0.714	+25.6%	0.871	+5.7%
	FastGCN [4]	2	0.850	+8.1%	0.776	+15.6%	0.880	+4.7%
	GAT [37]	2	0.830± 0.006	+10.7%	0.725± 0.005	23.7%	0.790± 0.002	+16.6%
	LGCN [8]	2	0.833± 0.004	+10.3%	0.730± 0.004	+22.9%	0.795± 0.002	+15.8%
	g-U-Nets [7]	4	0.844± 0.005	+8.9%	0.732±0.003	+22.5%	0.796± 0.002	+15.7%
	Adapt [17]	2	0.874± 0.003	+5.1%	0.796± 0.002	+12.7%	0.906± 0.002	+1.7%
NAS-GNNs	GraphNAS [9]	2	0.833 ± 0.002	+10.3%	0.735± 0.007	+22.0%	0.781± 0.003	+17.9%
	AGNN [45]	2	0.836 ± 0.003	+9.9%	0.738± 0.005	+21.5%	0.797± 0.003	+15.6%
Policy-GNN	Random Policy	2 ~ 5	0.770± 0.021	+19.4%	0.656± 0.024	+36.7%	0.788 ± 0.011	+16.9%
	Policy-GNN	2 ~ 5	**0.919± 0.014**	-	**0.897± 0.021**	-	**0.921± 0.022**	-

Lai et al. Policy-GNN: Aggregation Optimization for Graph Neural Networks . SIGKDD 2020

Policy-GNN [Lai et al. 2020]

- Search for the number of hops per node based on node attributes.
  - Distribution of number of GNN layers

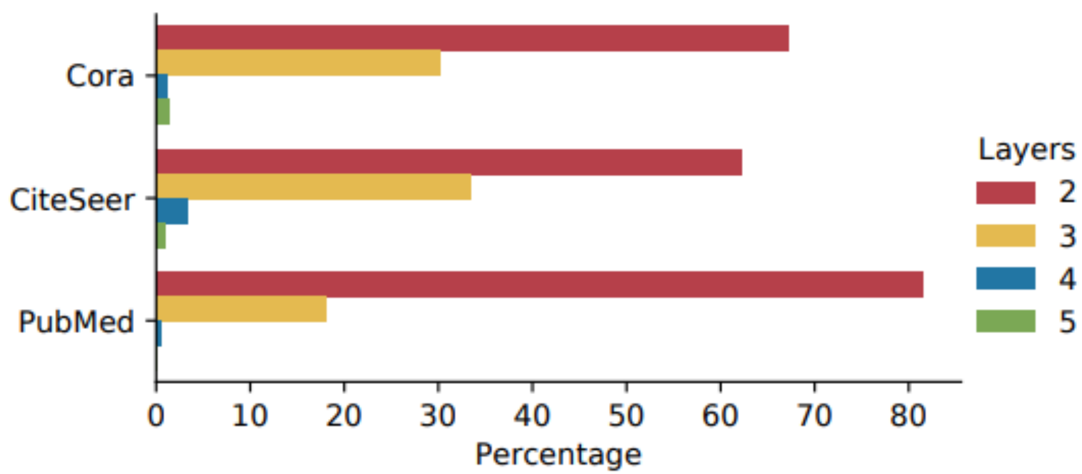


Figure 4: The percentage of the nodes that are assigned to different layers of GCN by *Policy-GNN*.

Lai et al. Policy-GNN: Aggregation Optimization for Graph Neural Networks . SIGKDD 2020

NAS for GNN

- Background
  - Graph Neural Network (GNN)
  - Neural Architecture Search (NAS)
- Two methods
  - Reinforcement Learning (RL)
  - Differentiable architecture search
- Discussion

Differentiable architecture search

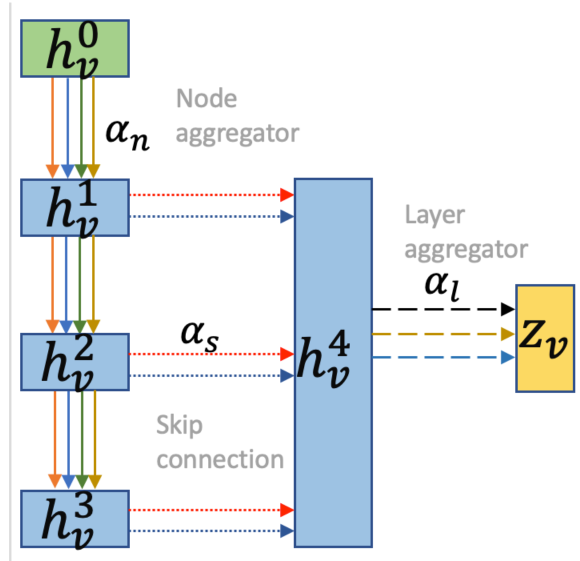
- Search to Aggregate Neighborhood (SANE) for GNN.
  - *Ongoing work*.

- Contributions
  - Novel and effective search space
  - Differentiable search algorithm
  - Transfer learning in large graphs

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	✓	✓	

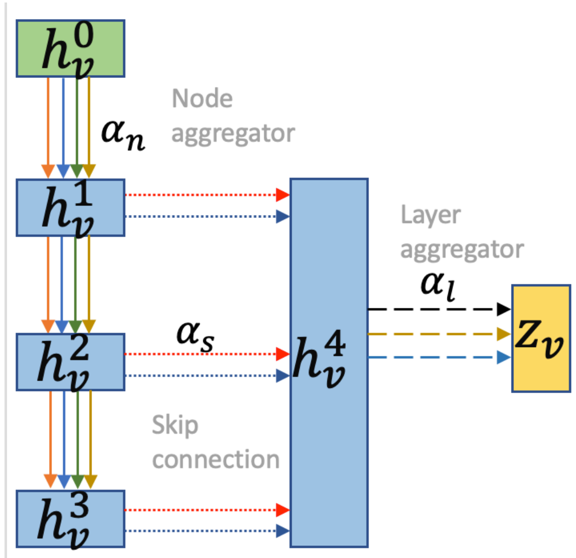
Search Space

- 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 $\{\mathbf{h}_u \mid u \in \widetilde{N}(v)\}$.
GAT [30]	GAT	Compute attention score: $\mathbf{e}_{uv}^{gat} = \text{Leaky\_ReLU}\left(\mathbf{a}\left[\mathbf{W}_u\mathbf{h}_u \Vert \mathbf{W}_v\mathbf{h}_v\right]\right)$.
	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 [33]	MLP	$F_N^l(v) = \text{MLP}\left(\left(1 + \epsilon^{l-1}\right) \cdot \mathbf{h}_v^{l-1} + \sum_{u \in N(v)} \mathbf{h}_u^{l-1}\right)$.
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.

Differentiable search algorithm

- Supernet (DAG)
  – Continuous relaxation
  – Mixed OPs



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

- Computation process

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

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

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

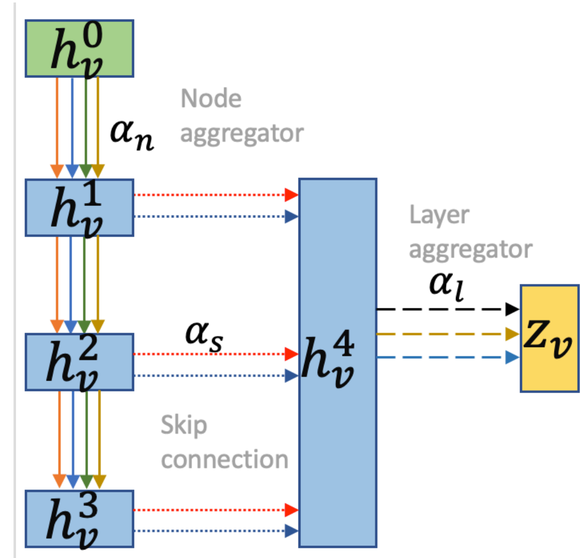
Differentiable search algorithm

- 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:*

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

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 $\mathbf{w}^*(\boldsymbol{\alpha})$ the corresponding weights after training.



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

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

Differentiable search algorithm

- Gradient-based optimization.

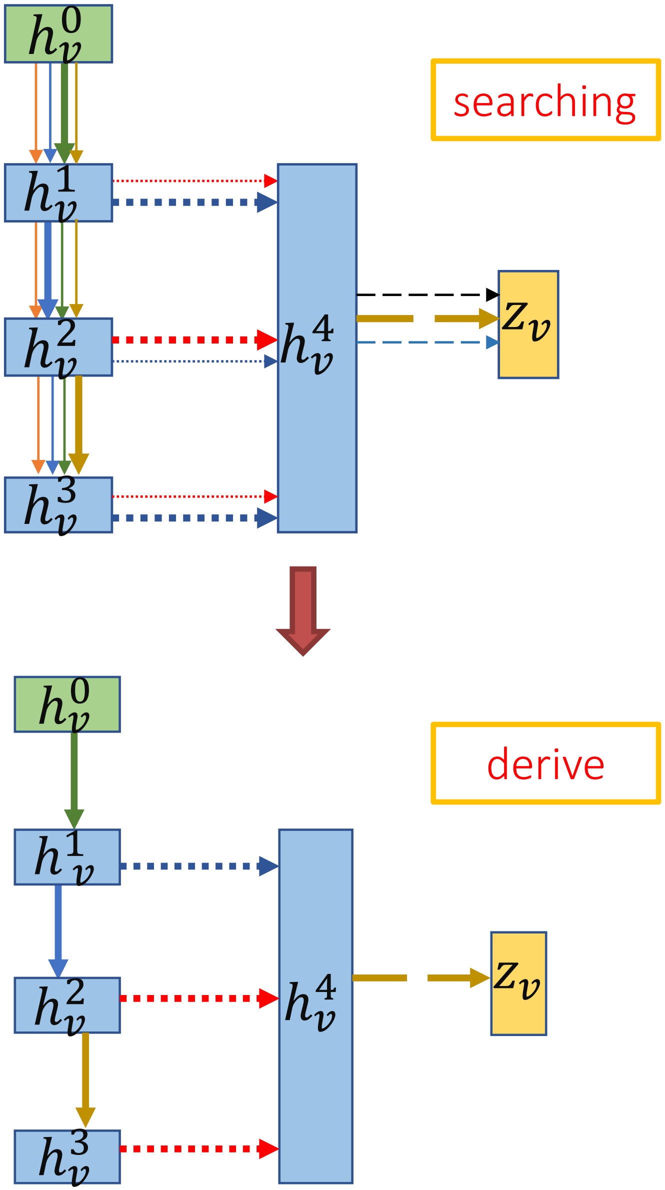
$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\mathbf{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\mathbf{w} - \xi \nabla_{\mathbf{w}} \mathcal{L}_{train}(\mathbf{w}, \boldsymbol{\alpha}), \boldsymbol{\alpha})$$

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 $\mathcal{A}_k$.

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



Transfer learning for large graphs

- Sample a small graph.
- Search in the small graph.
- Transfer to the large graph

Algorithm 2 Transferable SANE.

Require: The search space $\mathcal{F}$, the number of top architectures k, the epochs for search T, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Ensure: The k searched architectures $\mathcal{A}_k$ and corresponding test performance on $\mathcal{G}$.

1: Compute the PageRank values for nodes in $\mathcal{V}$, denoted as $\mathbf{p}$;
2: /*The detail of the sampling method is the RPN in [18]*/
3: Sample N, $N = 15\%|\mathcal{V}|$ nodes $\mathcal{V}_s$, according to the PageRank vector $\mathbf{p}$;
4: Construct a proxy graph $\mathcal{G}_{\text{proxy}} = (\mathcal{V}_s, \mathcal{E}_s)$, with $\mathcal{E}_s = \{e_{uv} | \forall e_{uv} \in \mathcal{E} \text{ and } u \in \mathcal{V}_s, v \in \mathcal{V}_s\}$.;
5: Obtain $\{\boldsymbol{\alpha}_n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_l\}$ by SANE (Algorithm 1) on $\mathcal{G}_{\text{proxy}}$, given $\mathcal{F}, k, T$, obtain the best k architecture.
6: Re-train the architecture $\{\boldsymbol{\alpha}_n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_l\}$ on $\mathcal{G}$, and return the test performance.

Experiments

- Task
  - Node classification
  - Transducive/inductive/transfer

- Datasets

Task	Datasets	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
transfer	Reddit	232,965	57,307,946	602	50

Experiments

- Performance comparison
  - Human-designed architectures
  - NAS approaches

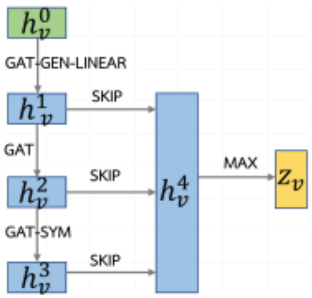
	Methods	Transductive			Inductive
		Cora	CiteSeer	PubMed	PPI
Human-designed architectures	GCN	0.8811 (0.0101)	0.7666 (0.0202)	0.8858 (0.0030)	0.6500 (0.0000)
	GCN-JK	0.8820 (0.0118)	0.7763 (0.0136)	0.8927 (0.0037)	0.8078(0.0000)
	GraphSAGE	0.8741 (0.0159)	0.7599 (0.0094)	0.8834 (0.0044)	0.6504 (0.0000)
	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)**

Experiments

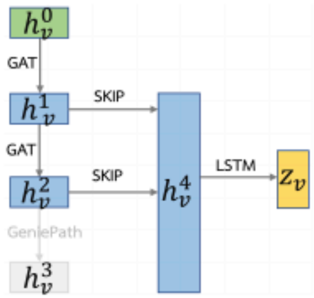
- Transferred learning

	Methods	Reddit
Human-designed architectures	GraphSAGE	0.9379 (0.0003)
	GraphSAGE-JK	0.9421 (0.0005)
NAS approaches	Random	0.9284 (0.0005)
	Bayesian	0.9379 (0.0002)
	GraphNAS	0.9432 (0.0004)
	GraphNAS-WS	0.9418 (0.0010)
one-shot NAS	SANE	**0.9528 (0.0008)**

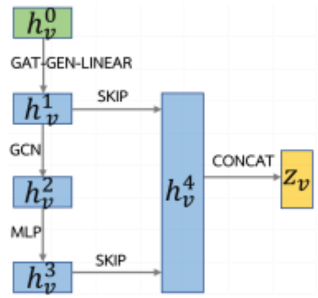
- Searched architectures



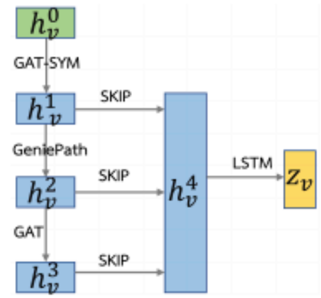
(a) Cora.



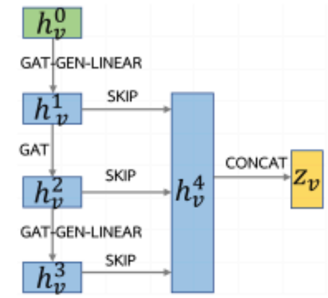
(b) CiteSeer.



(c) PubMed.



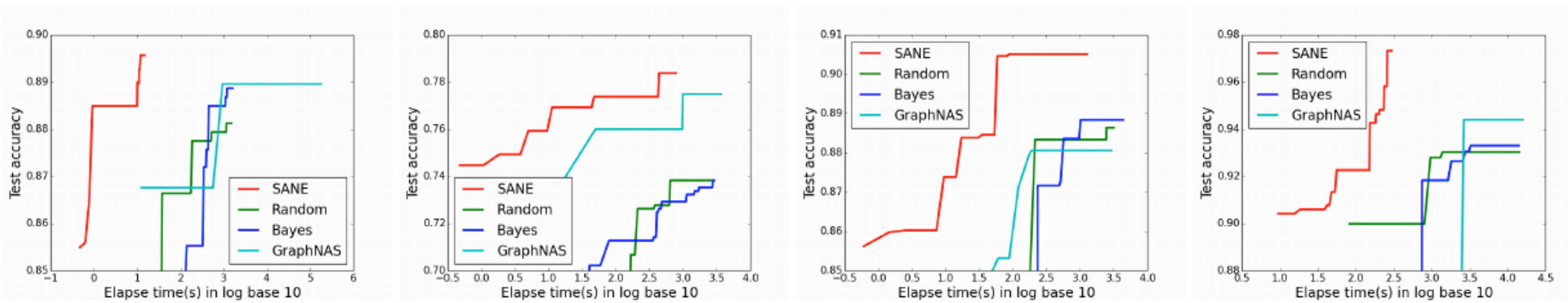
(d) PPI.



(e) Reddit.

Experiments

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



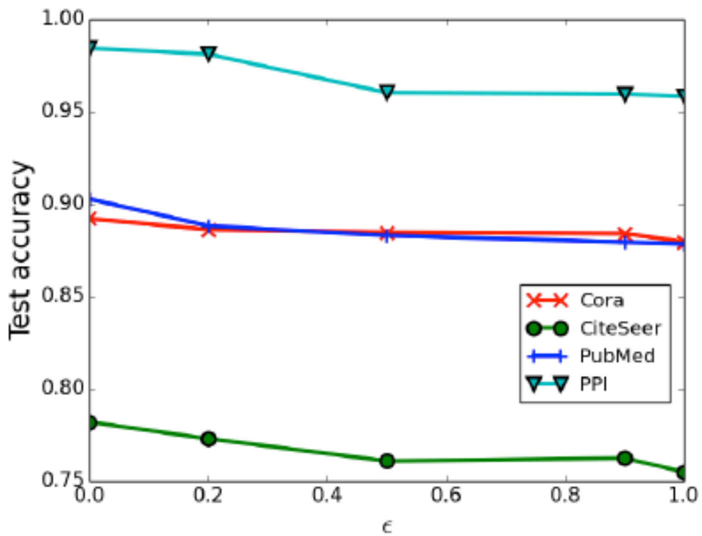
(a) Cora. (b) CiteSeer. (c) PubMed. (d) PPI.

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

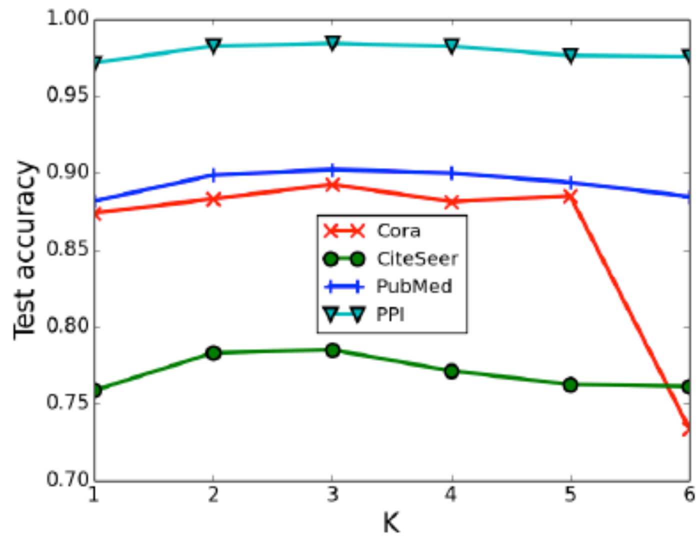
	transductive task			inductive task
	Cora	CiteSeer	PubMed	PPI
Random	1,500	2,694	3,174	13,934
Bayesian	1,631	2,895	4,384	14,543
GraphNAS	3,240	3,665	5,917	15,940
SANE	14	35	54	298

Experiments

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



(a) Random explore: ϵ.



(b) Number of GNN layers: K.

Discussion

- Conclusion
  - Novel and Effective search space
  - Differentiable search algorithm
  - Transfer learning in large graphs

- Future work
  - More advanced one-shot NAS approaches
    - NASP, ASNG-NAS, etc.
  - Other graph-based tasks
    - Graph-based recommendation

Automated GNN for Recommendation

- How to construct the graphs ?
  - Multiple graphs, bipartite graphs, heterogeneous graph?

- How to define neighbors beyond aggregation schemas?
  - Neighbors of different types
    - Rich side information
  - Neighbors in different hops
    - Higher-order relations

- How to address the scalability problem ?
  - Billion-scale graph in reality.

Summary

- GNN is naturally suitable for recommendation
- Automated machine learning can obtain data-specific GNN architectures.
- Automated GNN can further improve recommendation
  - Some key problems remain unexplored.

References

- Semi-supervised classification with graph convolutional networks. ICLR 2016
- Inductive representation learning on large graphs. NeurIPS 2017
- Graph attention networks. ICLR 2018
- Representation Learning on Graphs with Jumping Knowledge Networks. ICML 2018
- Geniepath: Graph neural networks with adaptive receptive paths. AAAI 2019
- Spam Review Detection with Graph Convolutional Networks. CIKM 2019
- Adaptive Stochastic Natural Gradient Method for One-Shot Neural Architecture Search. ICML 2019
- GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. IJCAI 2020
- AutoGNN: Neural Architecture Search of Graph Neural Networks. Arxiv 2019
- DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. ICLR 2019
- How powerful are graph neural networks?. ICLR 2019
- UNDERSTANDING AND ROBUSTIFYING DIFFERENTIABLE ARCHITECTURE SEARCH. ICLR 2020
- Efficient Neural Architecture Search via Proximal Iterations. AAAI 2020
- Policy-GNN: Aggregation Optimization for Graph Neural Networks. SIGKDD 2020

Q&A

Thank You! ☺