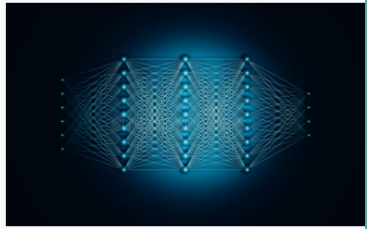
CIKM 2020 19-23 OCTOBER Online



4Paradigm 第四范式

Simplifying Architecture Search for Graph Neural Network

Huan Zhao, Lanning Wei, Quanming Yao

Machine Learning Research Group, 4Paradigm

{zhaohuan,weilanning,yaoquanming}@4paradigm.com

Oct. 20th 2020

Graph Neural Network

- GNN is a very hot topic in recent years
  - Representation learning in graphs
  - Define "convolution" on non-grid (graph) data
  - SOTA performance on various 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"

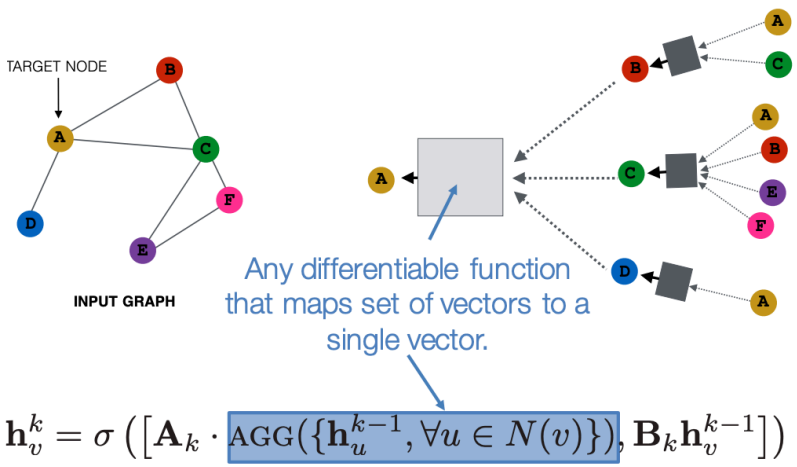
$$\mathbf{h}_v^l = \sigma\left(\mathbf{W}^{(l)} \cdot \mathrm{AGG}_{\mathrm{node}}\left(\{\mathbf{h}_u^{(l-1)}, \forall u \in \widetilde{N}(v)\}\right)\right)$$

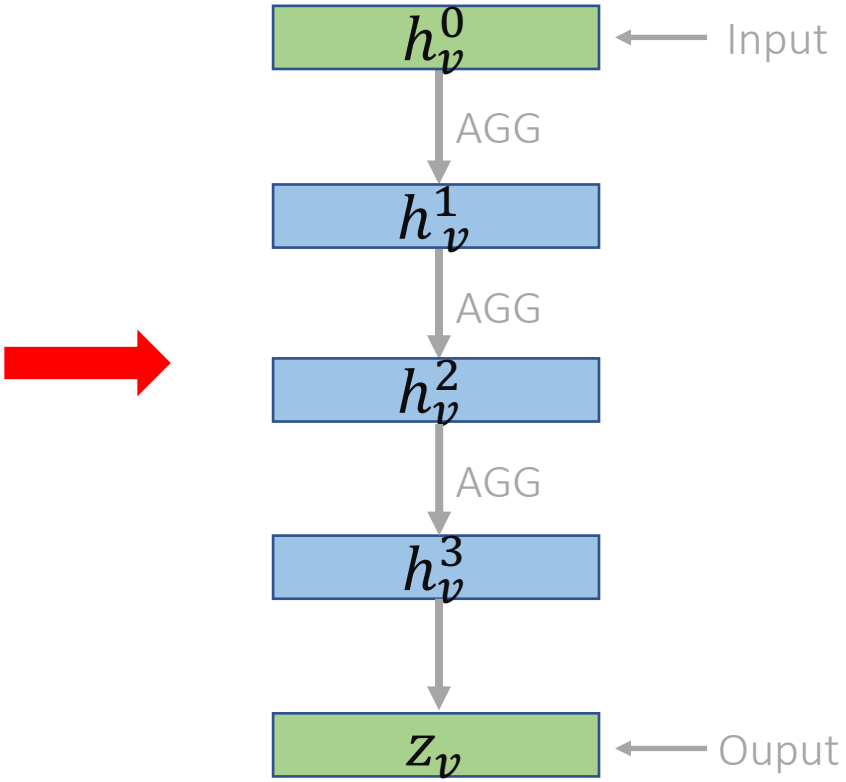
Self contained Neighborhood

- 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.





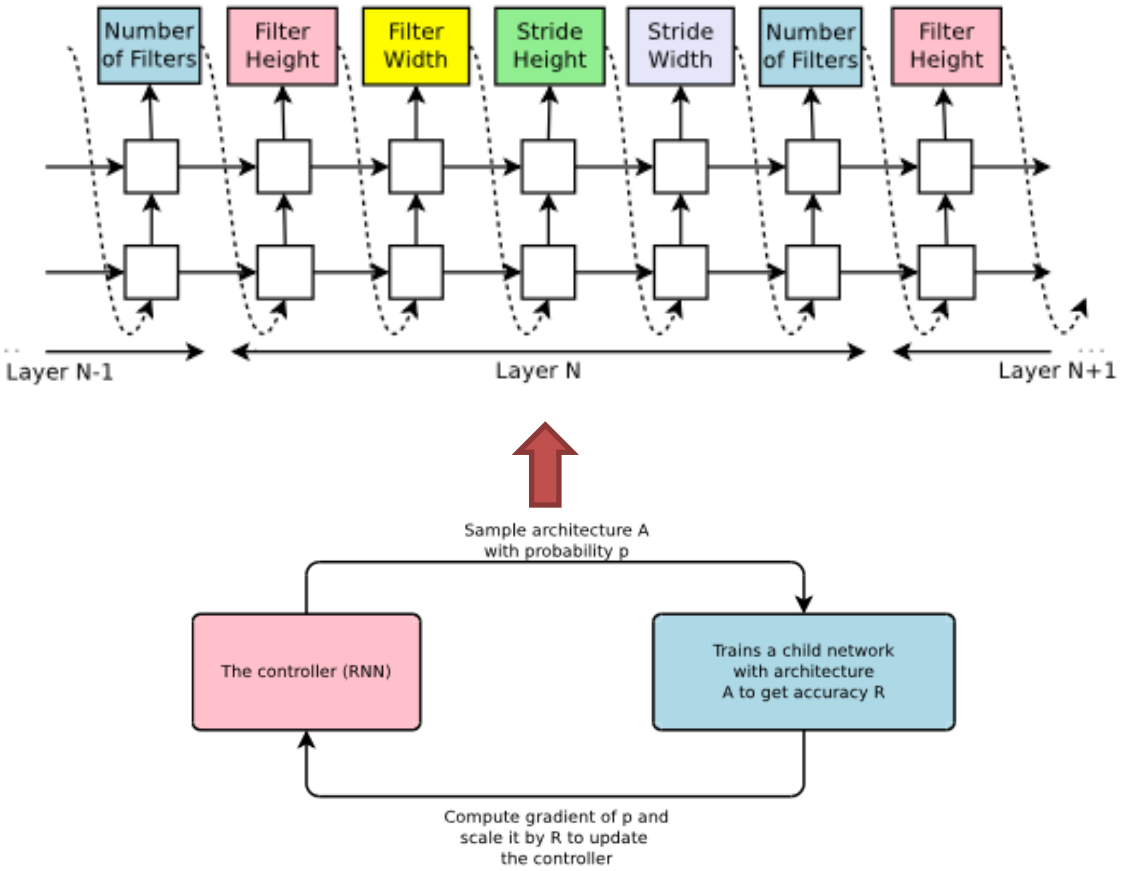
Leftside picture from http://snap.stanford.edu/proj/embeddings-www/

Motivation

- A direct question
  - *Can we obtain data-specific GNN architectures*?
- Neural Architecture Search (NAS)
  - Automatically design SOTA architectures for CNN.
- NAS for GNN
  - Obtain data-specific GNN architectures.
  - Automatically search for unexplored architectures beyond human-designed ones.

NAS

- Trial and Error
  - Iteratively train and evaluate the candidate architectures, and return the best one in the end.



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

Existing NAS methods for GNN

- 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]

- Problems
  - Search space are less expressive and too complex

Gao et al. Graphnas: Graph neural architecture search with reinforcement learning. IJCAI 2020
Zhou et el. Auto-GNN: Neural Architecture Search of Graph Neural Networks. Arxiv 2019

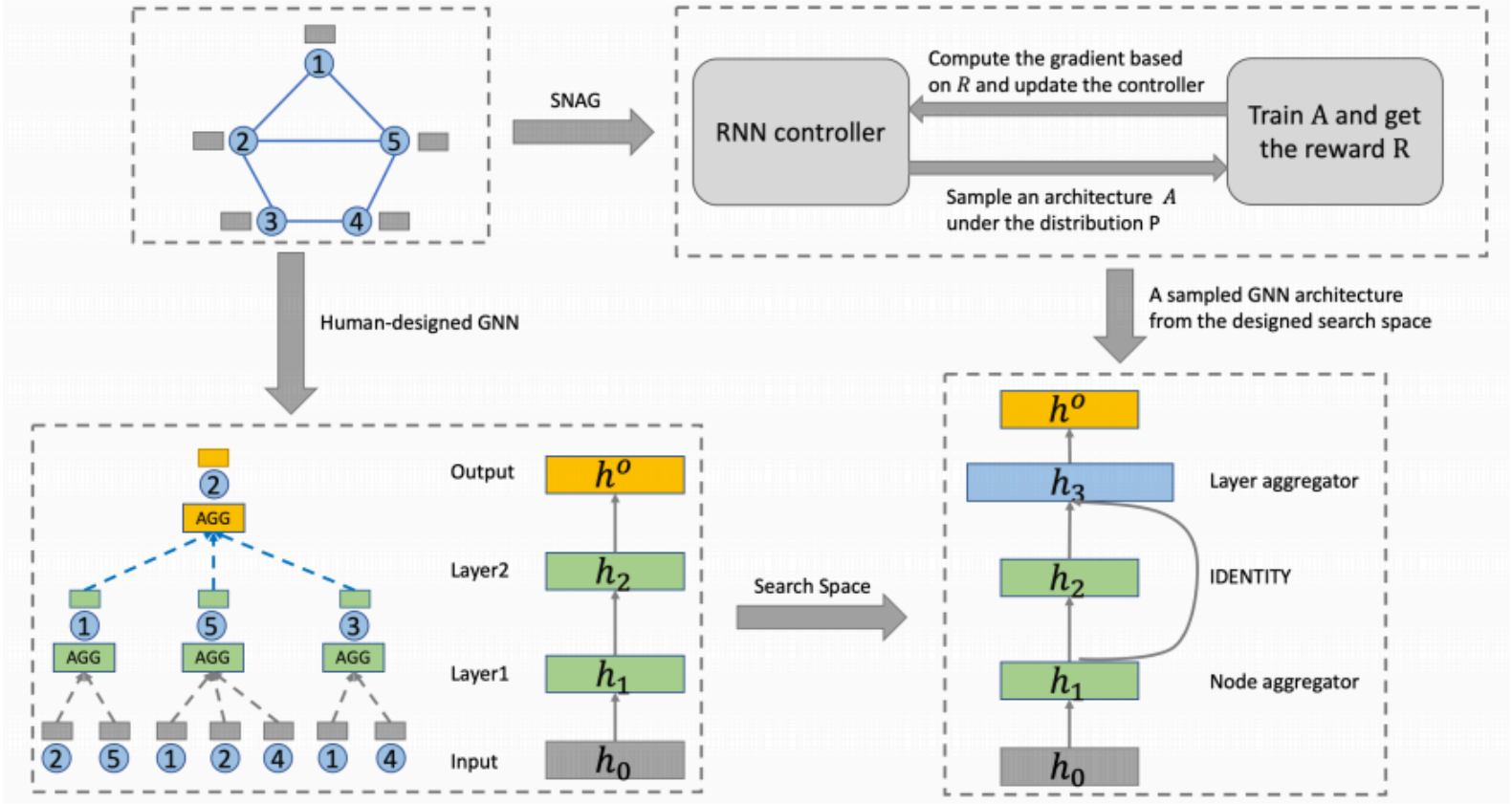
SNAG

- Simplifying Architecture Search for Graph Neural Network (SNAG)
  - Simplified yet expressive search space
  - Two RL variants (w/o weight sharing)
  - Efficacy and efficiency gain on benchmark datasets.

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Framework

- Overview framework



Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Search Space

- Search space
  - Node aggregators + Layer aggregators

	Node aggregators	Layer aggregators	Others
GraphNAS/ Auto-GNN	GCN [17],SAGE-SUM/-MEAN/-MAX [15], MLP [33], GAT [29] , GAT-SYM/-COS/ -LINEAR/-GEN-LINEAR [29] ,	-	Hidden Embedding Size, Attention Head, Activation Function
Ours	All above plus SAGE-LSTM [15] and GeniePath [21]	CONCAT,MAX,LSTM [34]	IDENTITY, ZERO

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Search method

- Search method: ENAS/GraphNAS
  - Reward

$$J(\theta_c) \equiv \mathbb{E}_{P(\alpha;\theta_c)}[\mathcal{R}]$$

  - Policy gradient

$$\nabla_{\theta_c} J(\theta_c) = \sum_{s=1}^{A} \mathbb{E}_{P(\alpha;\theta_c)}[\nabla_{\theta_c} \log P(\alpha_s | \alpha_{s-1:1}; \theta_c)\mathcal{R}].$$

  - MC approximation

$$\nabla_{\theta_c} J(\theta_c) = \frac{1}{m} \sum_{k=1}^{m} \sum_{s=1}^{A} \nabla_{\theta_c} \log P(\alpha_s | \alpha_{s-1:1}; \theta_c)(\mathcal{R}_k - b)$$

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Search method

- Search method: SNAG/SNAG-WS

Algorithm 1 SNAG-Simplified Neural Architecture search for Graph neural network.

Require: The search space $\mathcal{A}$, the number of sampled architectures n, the steps for search T, the epochs for training each child model T_α.

Ensure: The searched architecture.

1: **while** $t = 1, \cdots, T$ **do**
2: Sampled an architecture α based on $P(\alpha, \theta_c)$;
3: **Load OP parameters from saved ones** (*for SNAG-WS*);
4: Train α in T_α epochs, and get the validation accuracy $\mathcal{R}_t$;
5: Update θ_c according to Eq. (4);
6: **Save parameters of all OPs in the sampled achitecture** α (*for SNAG-WS*);
7: **end while**
8: Sampled n architectures from the trained $P(\alpha, \theta_c)$;
9: Train the n architectures from scratch and return the best architecture;

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Experiments

- Settings
  - Four benchmark datasets

	Transductive			Inductive
	Cora	CiteSeer	PubMed	PPI
#nodes	2,708	3,327	19,717	56,944
#edges	5,278	4,552	44,324	818,716
#features	1,433	3,703	500	121
#classes	7	6	3	50

  - Baselines
    - GNNs
    - NAS methods: Random, Bayesian, GraphNAS/GraphNAS-WS

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Experiments

- Results
  - transductive

	Methods	Transductive		
		Cora	CiteSeer	PubMed
Human-designed GNN	GCN	0.8761 (0.0101)	0.7666 (0.0202)	0.8708 (0.0030)
	GCN-JK	0.8770 (0.0118)	0.7713 (0.0136)	0.8777 (0.0037)
	GraphSAGE	0.8741 (0.0159)	0.7599 (0.0094)	0.8784 (0.0044)
	GraphSAGE-JK	0.8841 (0.0015)	0.7654 (0.0054)	0.8822 (0.0066)
	GAT	0.8719 (0.0163)	0.7518 (0.0145)	0.8573 (0.0066)
	GAT-JK	0.8726 (0.0086)	0.7527 (0.0128)	0.8674 (0.0055)
	GIN	0.8600 (0.0083)	0.7340 (0.0139)	0.8799 (0.0046)
	GIN-JK	0.8699 (0.0103)	0.7651 (0.0133)	0.8828 (0.0054)
	GeniePath	0.8670 (0.0123)	0.7594 (0.0137)	0.8796 (0.0039)
	GeniePath-JK	0.8776 (0.0117)	0.7591 (0.0116)	0.8818 (0.0037)
	LGCN	0.8687 (0.0075)	0.7543 (0.0221)	0.8753 (0.0012)
NAS methods	Random	0.8694 (0.0032)	**0.7820 (0.0020)**	0.8888(0.0009)
	Bayesian	0.8580 (0.0027)	0.7650 (0.0021)	0.8842(0.0005)
	GraphNAS	0.8840 (0.0071)	0.7762 (0.0061)	0.8896 (0.0024)
	GraphNAS-WS	0.8808 (0.0101)	0.7613 (0.0156)	0.8842 (0.0103)
ours	SNAG	0.8826 (0.0023)	0.7707 (0.0064)	0.8877 (0.0012)
	SNAG-WS	**0.8895 (0.0051)**	0.7695 (0.0069)	**0.8942 (0.0010)**

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Experiments

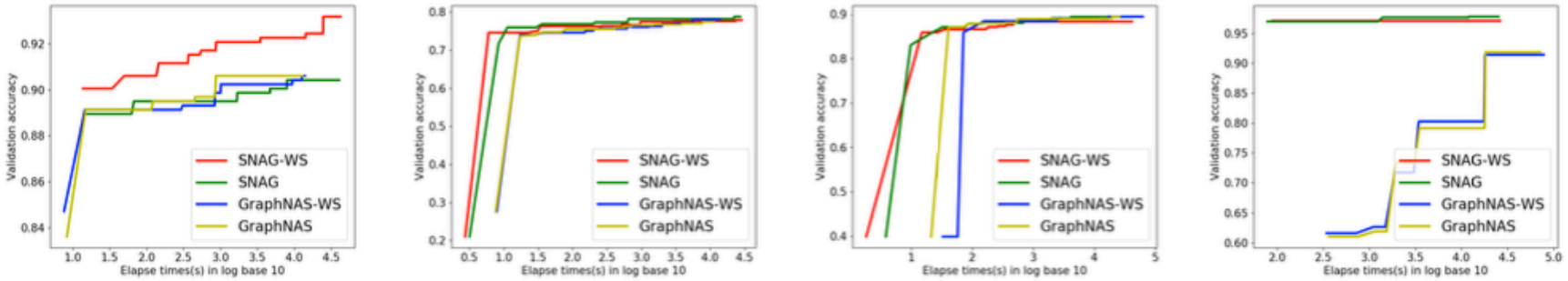
- Results
  - inductive

	Methods	PPI
Human-designed GNN	GCN	0.9333 (0.0019)
	GCN-JK	0.9344 (0.0007)
	GraphSAGE	0.9721 (0.0010)
	GraphSAGE-JK	0.9718 (0.0014)
	GAT	0.9782 (0.0005)
	GAT-JK	0.9781 (0.0003)
	GIN	0.9593 (0.0052)
	GIN-JK	0.9641 (0.0029)
	GeniePath	0.9528 (0.0000)
	GeniePath-JK	0.9644 (0.0000)
NAS methods	Random	0.9882 (0.0011)
	Bayesian	**0.9897 (0.0008)**
	GraphNAS	0.9698 (0.0128)
	GraphNAS-WS	0.9584 (0.0415)
ours	SNAG	**0.9887 (0.0010)**
	SNAG-WS	0.9875 (0.0006)

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Experiments

- Results
  - Speedup in searching



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

Figure 2: The validation accuracy w.r.t. search time (in seconds) in log base 10.

  - Influences of layer aggregators

	SNAG		SNAG-WS	
	layer aggregators (w)	layer aggregators (w/o)	layer aggregators (w)	layer aggregators (w/o)
Cora	0.8826 (0.0023)	0.8822 (0.0071)	0.8895 (0.0051)	0.8892 (0.0062)
CiteSeer	0.7707 (0.0064)	0.7335 (0.0025)	0.7695 (0.0069)	0.7530 (00034)
PubMed	0.8877 (0.0012)	0.8756 (0.0016)	0.8942 (0.0010)	0.8800 (0.0013)
PPI	0.9887 (0.0010)	0.9849 (0.0040)	0.9875 (0.0006)	0.9861 (0.0009)

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Summary

- Simplifying Architecture Search for Graph Neural Network (SNAG)
  - Simplified yet expressive search space
    - Node + Layer Aggregators.
  - Two RL variants based on weight sharing
  - Performance and Efficiency gain on experiments

- Future work
  - More advanced NAS methods
    - Differentiable search methods.

Zhao et al. Simplifying Architecture Search for Graph Neural Network. CIKM-CSSA 2020

Q&A

Thank You! ☺

More questions, you can also contact: zhaohuan@4paradigm.com