清華大學
Tsinghua University

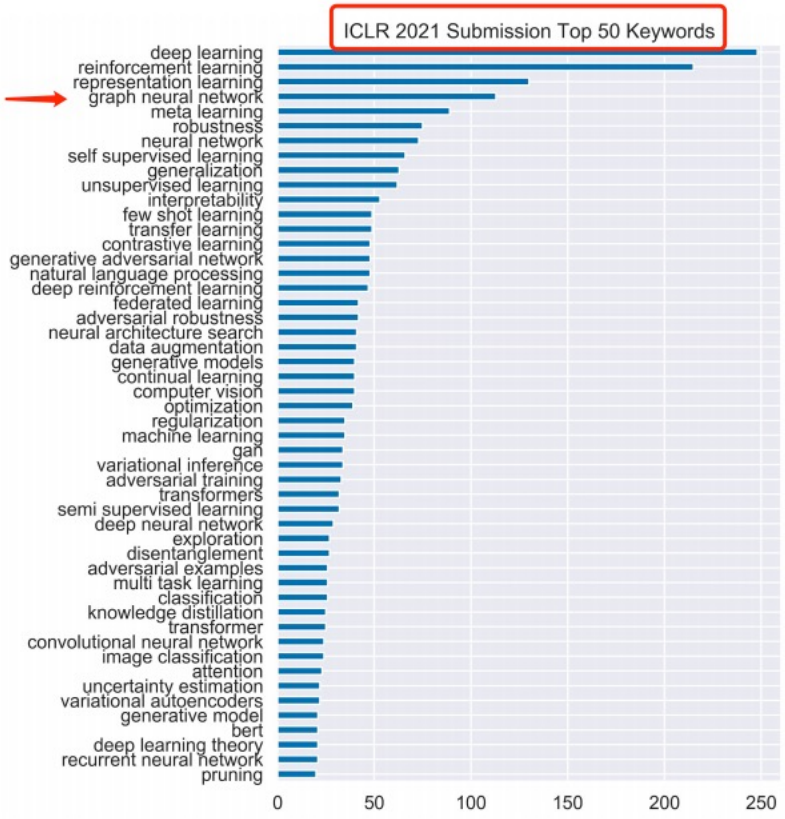
4Paradigm
第四范式

Search to aggregate neighborhood for graph neural network

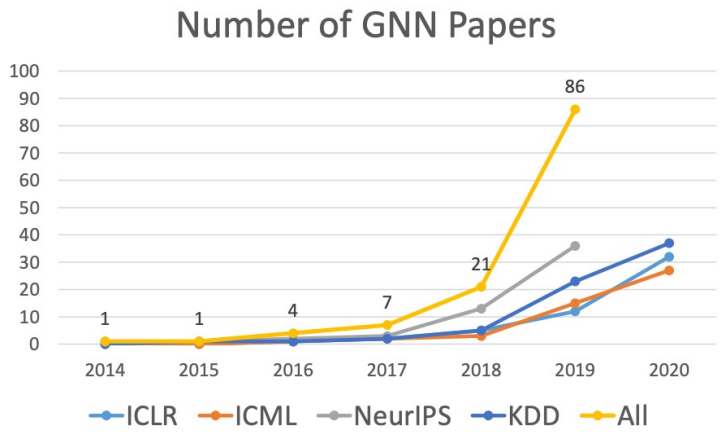
Huan Zhao, Quanming Yao, Weiwei Tu
4Paradigm & Department of Electronic Engineering, Tsinghua University
{zhaohuan,yaoquanming,tuweiwei}@4paradigm.com

Graph Neural Network

- Graph neural networks have been a hot topic in recent years.
  - Research



Slide Credit: Petar Veličković



Slide Credit: Tecent AI Lab

Graph Neural Network

- Graph neural networks have been a hot topic in recent years.
  - Applications

Commodity Recommendation

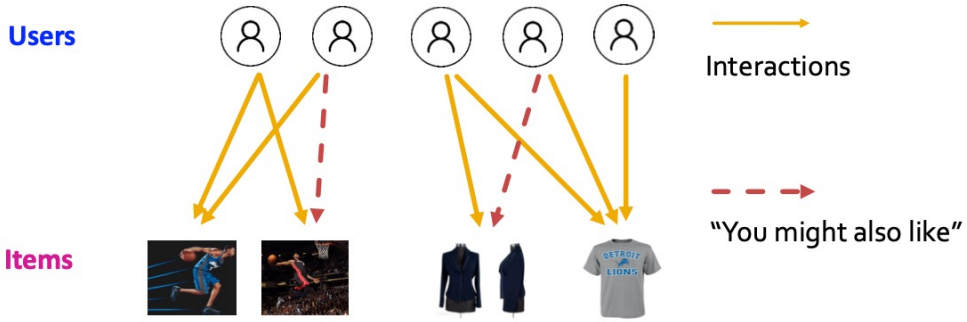


Image Credit: Jure Stanford CS224

Traffic Prediction

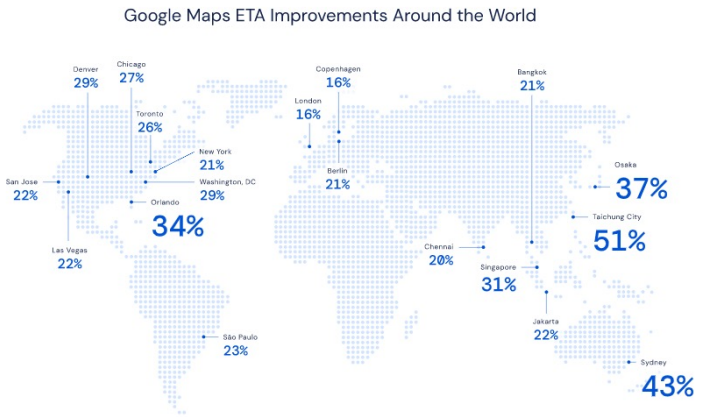


Image Credit: DeepMind

Molecular Classification

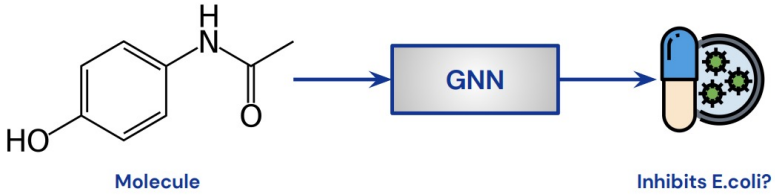


Image Credit: Petar Veličković

Graph Neural Network

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

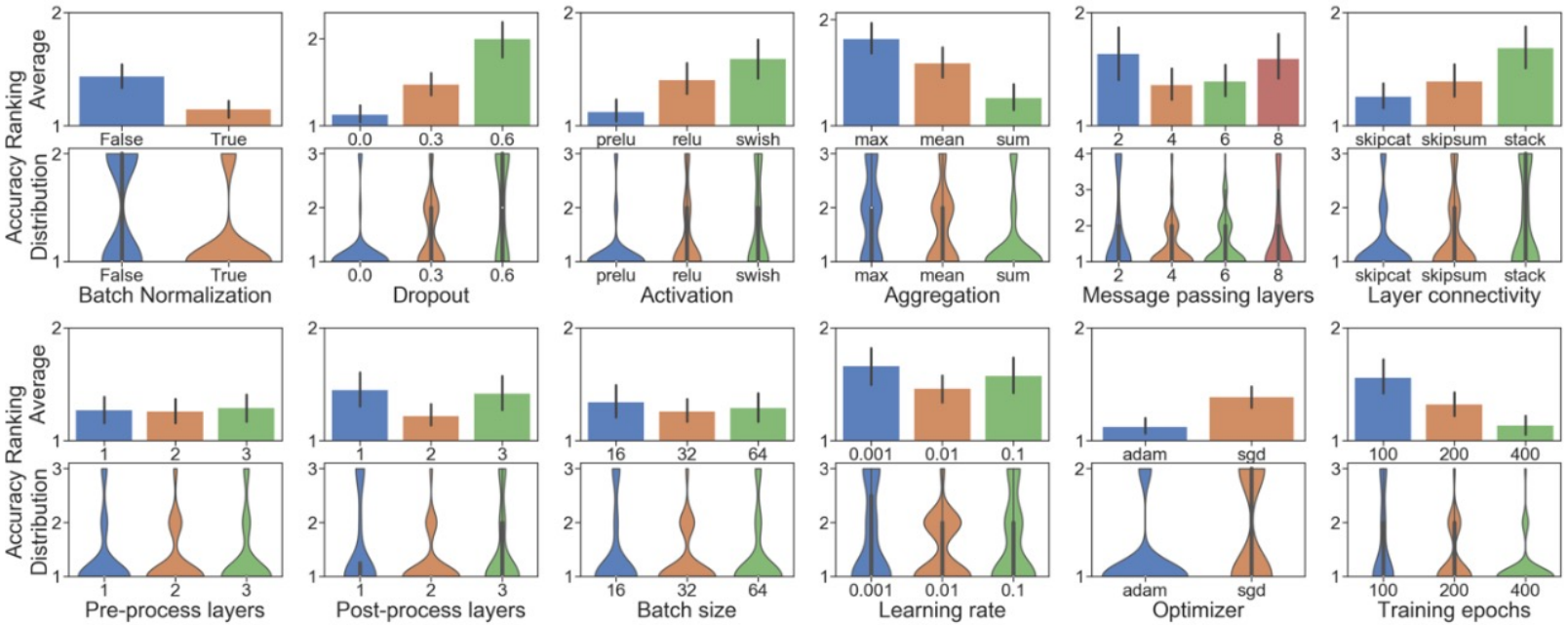
Self contained Neighborhood

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

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

Graph Neural Network

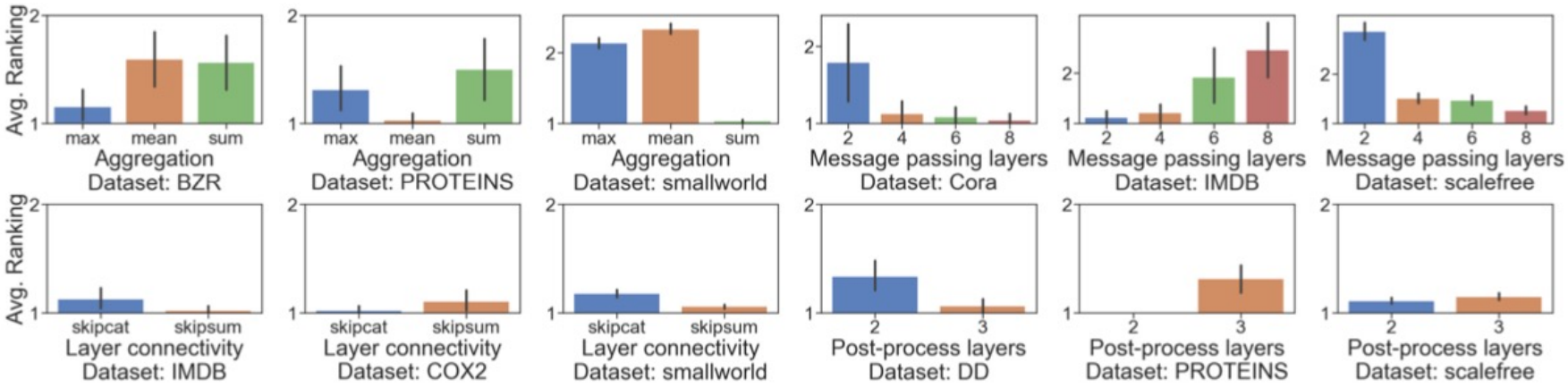
- Architecture challenge
  - Tons of GNN models have been designed.
    - E.g. 315, 000 GNN models by 12 dimensions.



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

Graph Neural Network

- Architecture challenge
  - Performance of GNN models vary in different datasets



It leads to the application of neural architecture search (NAS)

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

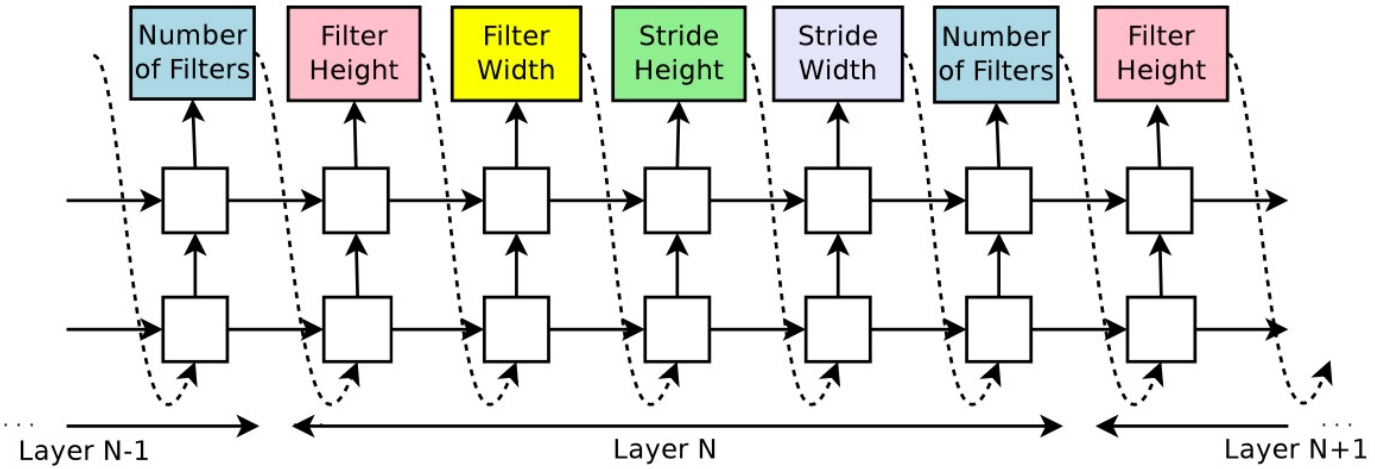
Neural Architecture Search

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

- Trial and Error
  - Iteratively train and evaluate the candidate architectures until obtaining the best one.
    - Search space
    - Search algorithm

Neural Architecture Search

- An example in CNN architecture search.



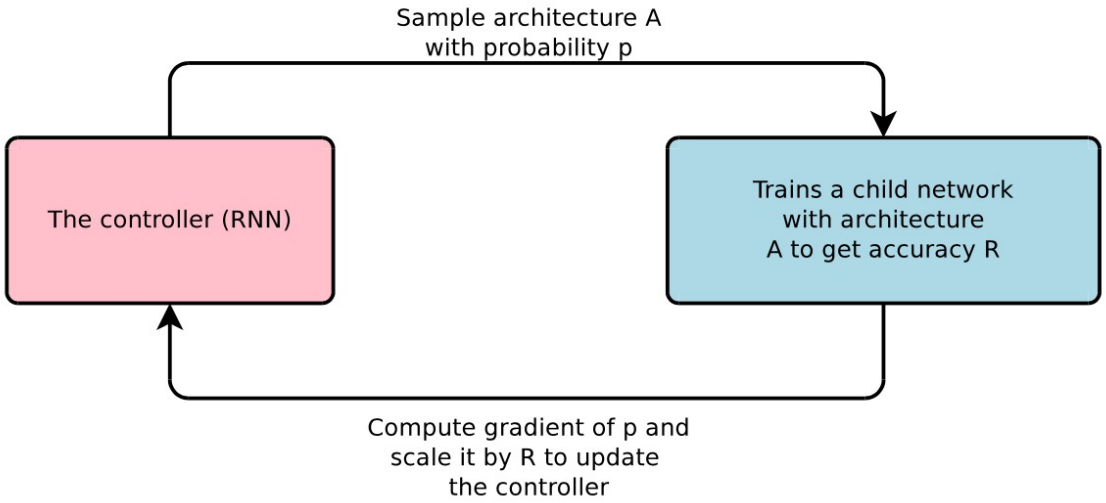
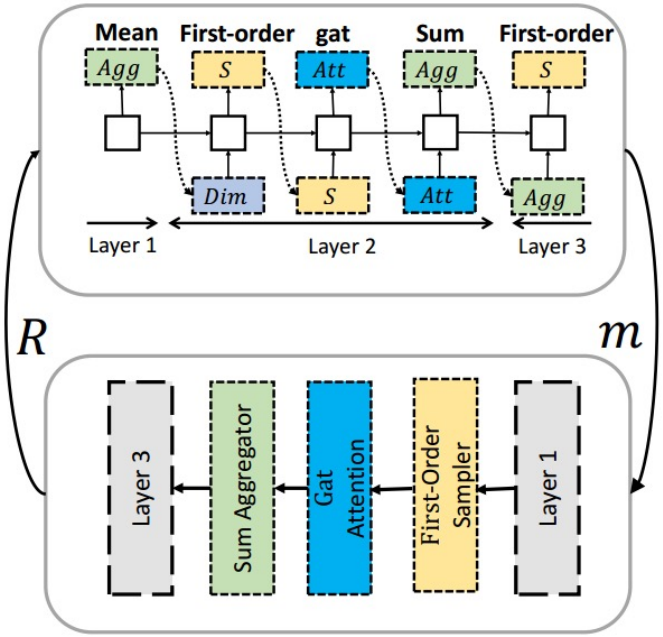


Image Credit: Zoph et al.

NAS for GNN

- GraphNAS [1]
  - The first reinforcement learning method for graph neural architecture search
  - Search space

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



  - Search algorithm
    - Update the RNN controller based on the validation accuracy of the sampled GNN models.

1. Gao et al. Graph Neural architecture search. IJCAI 2020

NAS for GNN

- GraphNAS obtain data-specific GNN models in different datasets.
  - Similar works like Auto-GNN [1]
- However, the RL-based algorithm is inherently expensive, since it has to train from scratch every sampled architecture.
  - Computational challenge
- Thus, we propose a more efficient method for graph neural architecture search in this work.

1. Zhou et al. Auto-GNN: Neural architecture search of graph neural networks. Technical Report 2019

Differentiable graph architecture search

- Search to Aggregate Neighborhood (SANE) for GNN.

- Contributions
  - A novel and expressive search space
  - Differentiable search algorithm
    - Two orders more efficient
  - Extensive experiments demonstrate the effectiveness and efficiency of SANE

Search Space

- Message passing framework
  - Node aggregator
    - 11 popular aggregators
  - Layer aggregator
    - Inspired by JK network [1]
    - Search to skip or not for each layer.
- The detailed search space

	Operations
$\mathcal{O}_n$	SAGE-SUM, SAGE-MEAN, SAGE-MAX, GCN, GAT,GAT-SYM, GAT-COS, GAT-LINEAR, GAT-GEN-LINEAR, GIN, GeniePath
$\mathcal{O}_l$	CONCAT, MAX, LSTM
$\mathcal{O}_s$	IDENTITY, ZERO

Xu et al. Representation Learning on Graphs with Jumping Knowledge Networks . ICML 2018

Search Space

- Compared to human-designed GNNs

	Model	Node aggregators	Layer aggregators	Emulate by SANE
Human-designed architectures	GCN [14]	GCN	×	✓
	SAGE [3]	SAGE-SUM/-MEAN/-MAX	×	✓
	GAT [4]	GAT, GAT-SYM/-COS/ -LINEAR/-GEN-LINEAR	×	✓
	GIN [16]	GIN	×	✓
	LGCN [15]	CNN	×	✓
	GeniePath [18]	GeniePath	×	✓
	JK-Network [17]	depends on the base GNN	✓	✓
NAS	SANE	learned combination of aggregators	✓	

- Compared to existing NAS methods for GNN

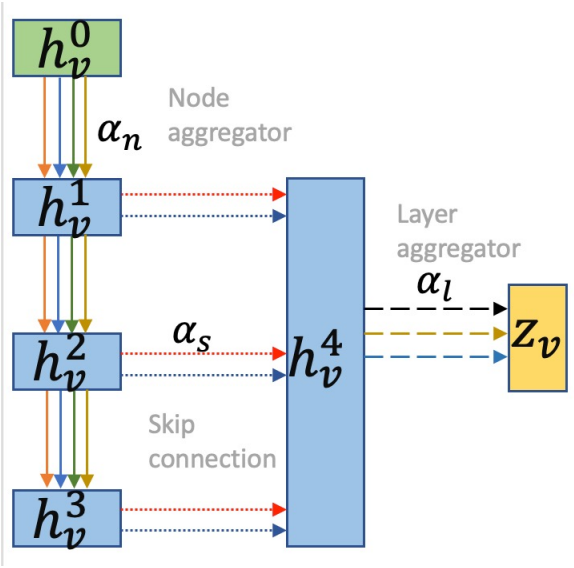
TABLE III: A detailed comparison between SANE and existing NAS methods for GNN.

	Search space		Search algorithm
	Node aggregators	Layer aggregators	
GraphNAS, Auto-GNN	√	×	RL
Policy-GNN	×	×	RL
SANE	√	√	Differentiable

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

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

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

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

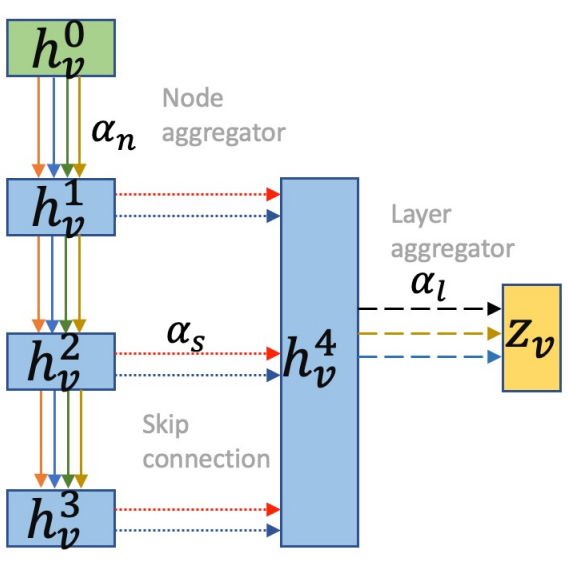
Differentiable search algorithm

- 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.

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

Differentiable search algorithm

- Gradient-based optimization
  - First-order approximation [1]

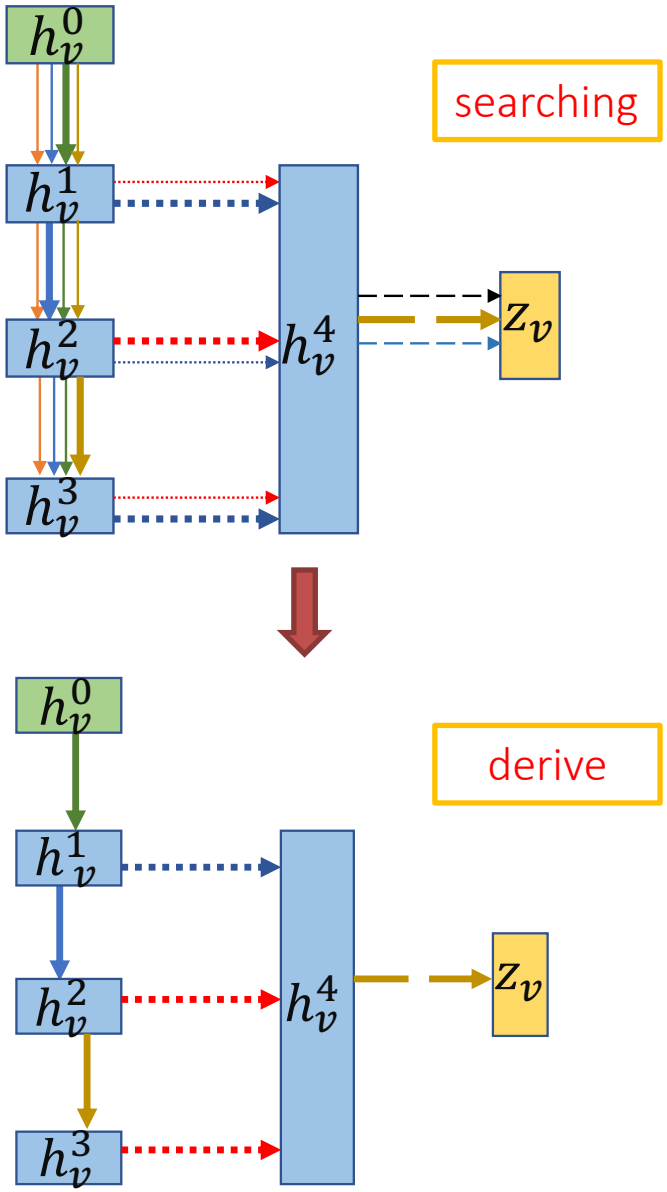
$$\nabla_{\alpha}\mathcal{L}_{val}(w^{*}(\alpha),\alpha) \approx \nabla_{\alpha}\mathcal{L}_{val}(\mathbf{w}-\xi\nabla_{\mathbf{w}}\mathcal{L}_{train}(\mathbf{w},\alpha),\alpha)$$

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 $\{\boldsymbol{\alpha}_n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_l\}$;



Liu et al. DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. ICLR 2019

Experiments

- Datasets and Tasks
  - Node classification
    - Transductive
    - Inductive
  - Entity alignment
    - DB task

TABLE V: The statistics of the dataset DBP15K$_{ZH-EN}$.

	#Entities	#Relations	#Attributes	#Rel.triples	#Attr.triples
Chinese	66,469	2,830	8.113	153,929	379,684
English	98,125	2,317	7,173	237,674	567,755

TABLE IV: Dataset statistics of the datasets in the experiments. N, E, F and C denote the number of "Nodes", "Edges" "Features" and "Classes", respectively.

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

Experiments

- Node classification
  - Baselines
    - Human-designed GNNs
      - GCN, GraphSAGE, GAT, GIN, LGCN, GeniePath
      - GCN-JK, GraphSAGE-JK, GAT-JK, GAT-JK, GIN-JK, GeniePath-JK
    - NAS for GNNs
      - Random, Bayesian, GraphNAS, GraphNAS-WS
  - Settings
    - 6:2:2 in transductive
    - 10:1:1 in inductive

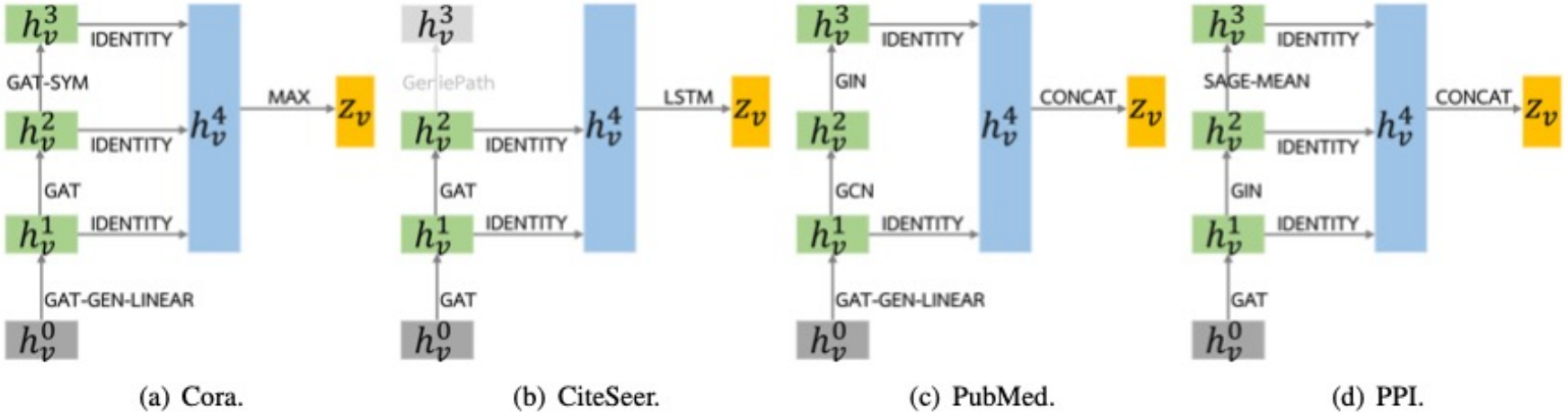
Experiments

- Performance comparison
  - Node classification
  - Best performance compared to all baselines.

		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)**

Experiments

- Searched architectures
  - Attention aggregators are more likely to selected



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

Experiments

- Entity alignment
  - 2-layer GNN as backbone
  - 30% 10% 60% -> train/validation/test
  - "GAT-GeniePath"

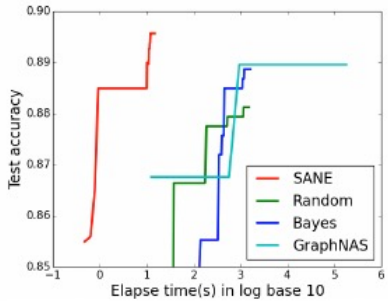
TABLE VIII: The results of DB task. We use Hits@K as the evaluation metric, and the results of JAPE and GCN-Align are from [42]. Note that JAPE is the variant using the same features as GCN-Align.

	ZH→EN			EN→ZH		
	@1	@10	@50	@1	@10	@50
JAPE	33.32	69.28	86.40	33.02	66.92	85.15
GCN-Align	41.25	74.38	86.23	36.49	69.94	82.45
SANE	**42.10**	**74.51**	**88.12**	**38.41**	**70.23**	**85.43**

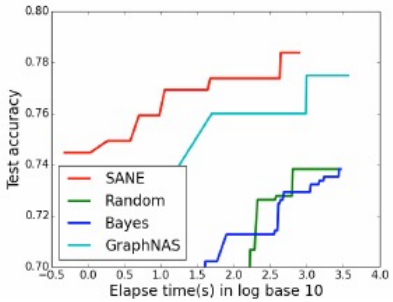
More exploration of graph architecture search in DB tasks in future work.

Experiments

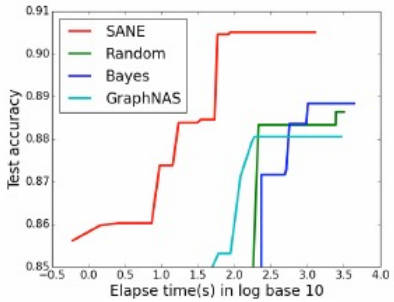
- Search Efficiency
  - The test accuracy w.r.t. search time (s)



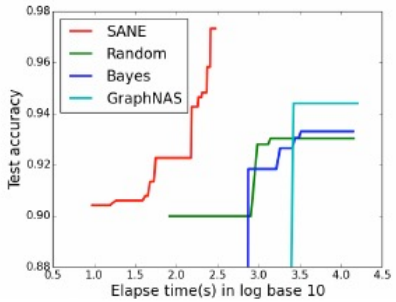
(a) Cora.



(b) CiteSeer.



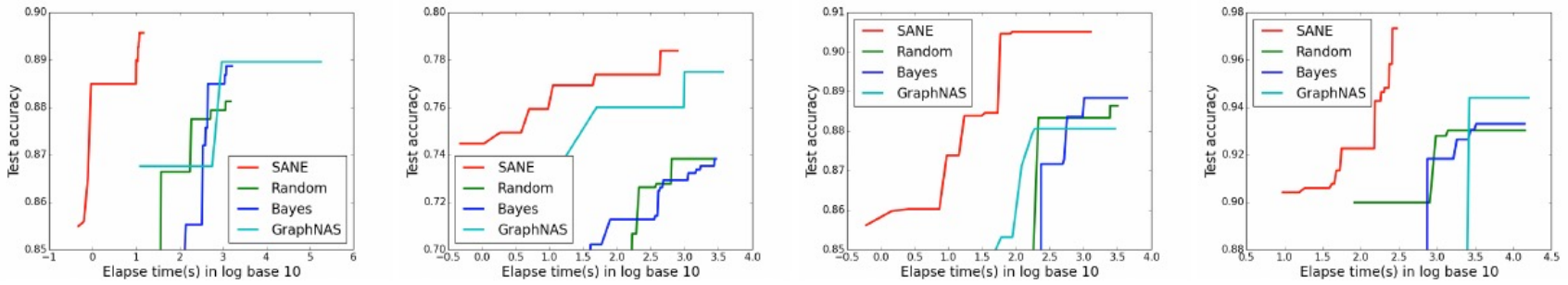
(c) PubMed.



(d) PPI.

Experiments

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



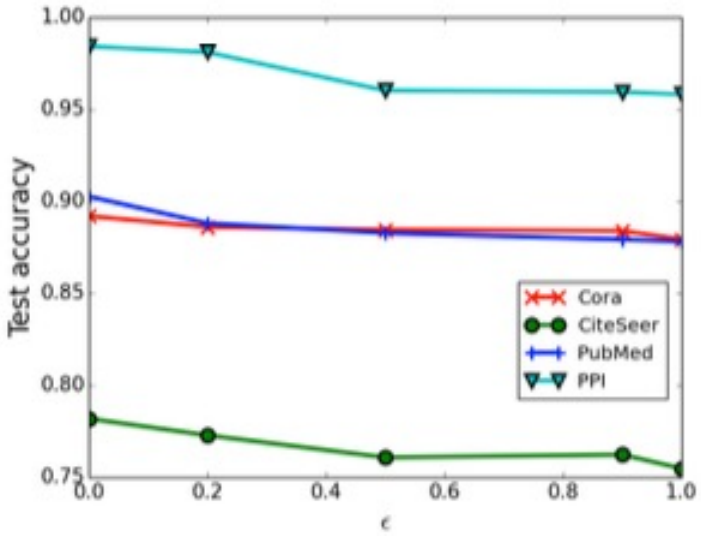
(a) Cora. (b) CiteSeer. (c) PubMed. (d) PPI.

  - The running cost (s) of each method during the search phase.

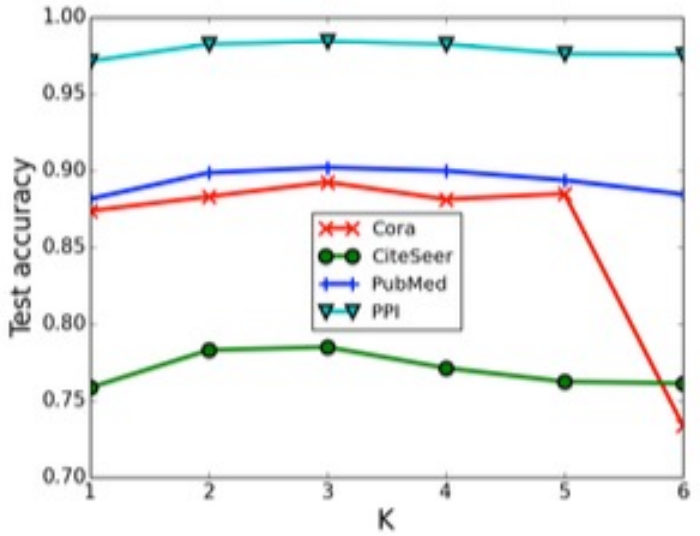
	transductive task			inductive task
	Cora	CiteSeer	PubMed	PPI
Random	1,500	2,694	3,174	13,934
Bayesian	1,631	2,895	4,384	14,543
GraphNAS	3,240	3,665	5,917	15,940
SANE	14	35	54	298

Experiments

- Ablation Study
  - The influence of differentiable search
  - The influence of K



(a) Random explore: ϵ.



(b) Number of GNN layers: K.

Experiments

- Ablation Study
  - The efficacy of the designed search space

Methods	Cora	CiteSeer	PubMed	PPI
GraphNAS	0.8840 (0.0071)	**0.7762 (0.0061)**	0.8896 (0.0024)	0.9698 (0.0128)
GraphNAS-WS	0.8808 (0.0101)	0.7613 (0.0156)	0.8842 (0.0103)	0.9584 (0.0415)
GraphNAS(SANE search space)	0.8826 (0.0023)	0.7707 (0.0064)	0.8877 (0.0012)	**0.9887 (0.0010)**
GraphNAS-WS(SANE search space)	**0.8895 (0.0051)**	0.7695 (0.0069)	**0.8942 (0.0010)**	0.9875 (0.0006)

  - Failure of searching for universal approximator

TABLE X: The performance of searching for MLP. We list the best performance of SANE from Table VI as comparison.

Dataset	Random	Bayesian	SANE
Cora	0.8698 (0.0011)	0.8470 (0.0032)	**0.8926 (0.0123)**
CiteSeer	0.7298 (0.0078)	0.7103 (0.0057)	**0.7859 (0.0108)**
PubMed	0.8662 (0.0030)	0.8699 (0.0065)	**0.9047 (0.0091)**
PPI	0.8166 (0.0089)	0.8685 (0.0017)	**0.9856 (0.0120)**

Conclusion and Future work

- Conclusion
  - We design a novel and effective search space
  - We develop a differentiable search algorithm
  - Extensive experiments demonstrate the effectiveness and efficiency of the proposed framework.
- Future work
  - More advanced one-shot NAS approaches
    - Stochastic Neural Architecture Search (SNAS) [1]
  - More tasks
    - Open Graph Benchmark [2]

1 Xie et al. SNAS: Stochastic neural architecture search. ICLR 2019
2 Hu et al. Open graph benchmark: Datasets for machine learning on graphs neurIPS. NeurIPS 2020

Acknowledgements

- Support from 4Paradigm.
- Lanning Wei helps implement some baselines.

1 Xie et al. SNAS: Stochastic neural architecture search. ICLR 2019

2 Hu et al. Open graph benchmark: Datasets for machine learning on graphs neurIPS. NeurIPS 2020

Q&A

- Paper & Code
  - https://github.com/AutoML-4Paradigm/SANE
  - Any questions, open an issue or drop an email to zhaohuan@4paradigm.com

- More codes on AutoML research can be found in our Github.
  - https://github.com/AutoML-4Paradigm
- More works related to AutoGraph can be found in my homepage.
  - https://hzhaoaf.github.io/