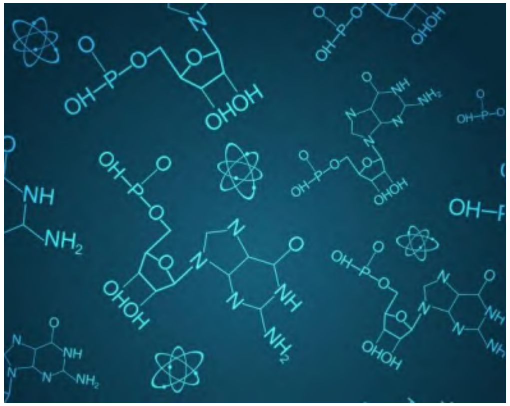
Automated Machine Learning on Graphs

Ziwei Zhang
Tsinghua University

2022.11.19@LOGS

Graphs are Ubiquitous

Social Network



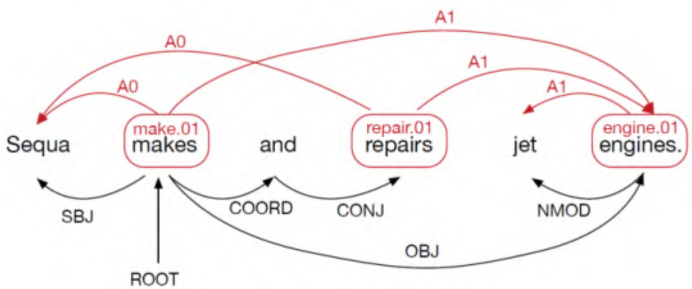
Biology Network

Traffic Network

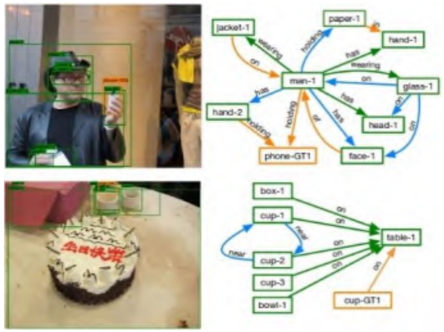
Information Network

Images are credit to web search engines

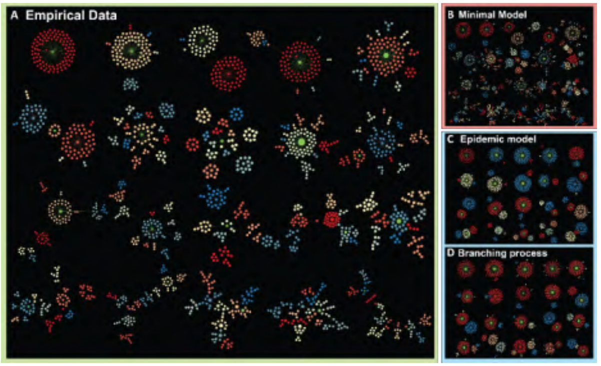
Graph Applications



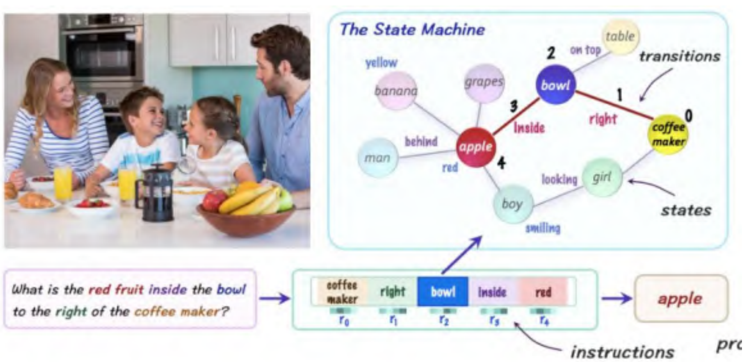
Natural Language Processing



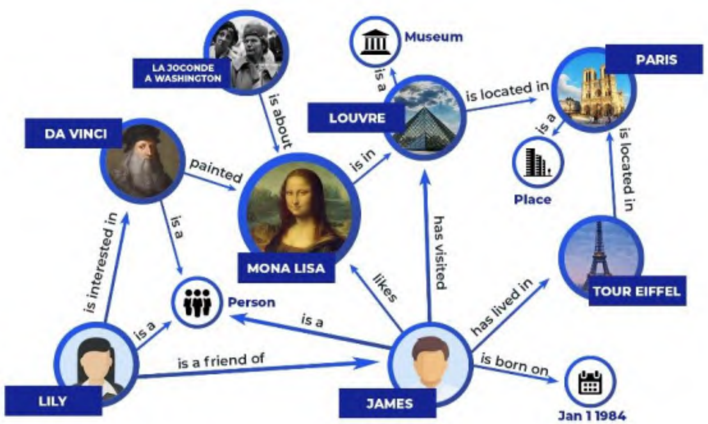
Computer Vision



Data Mining



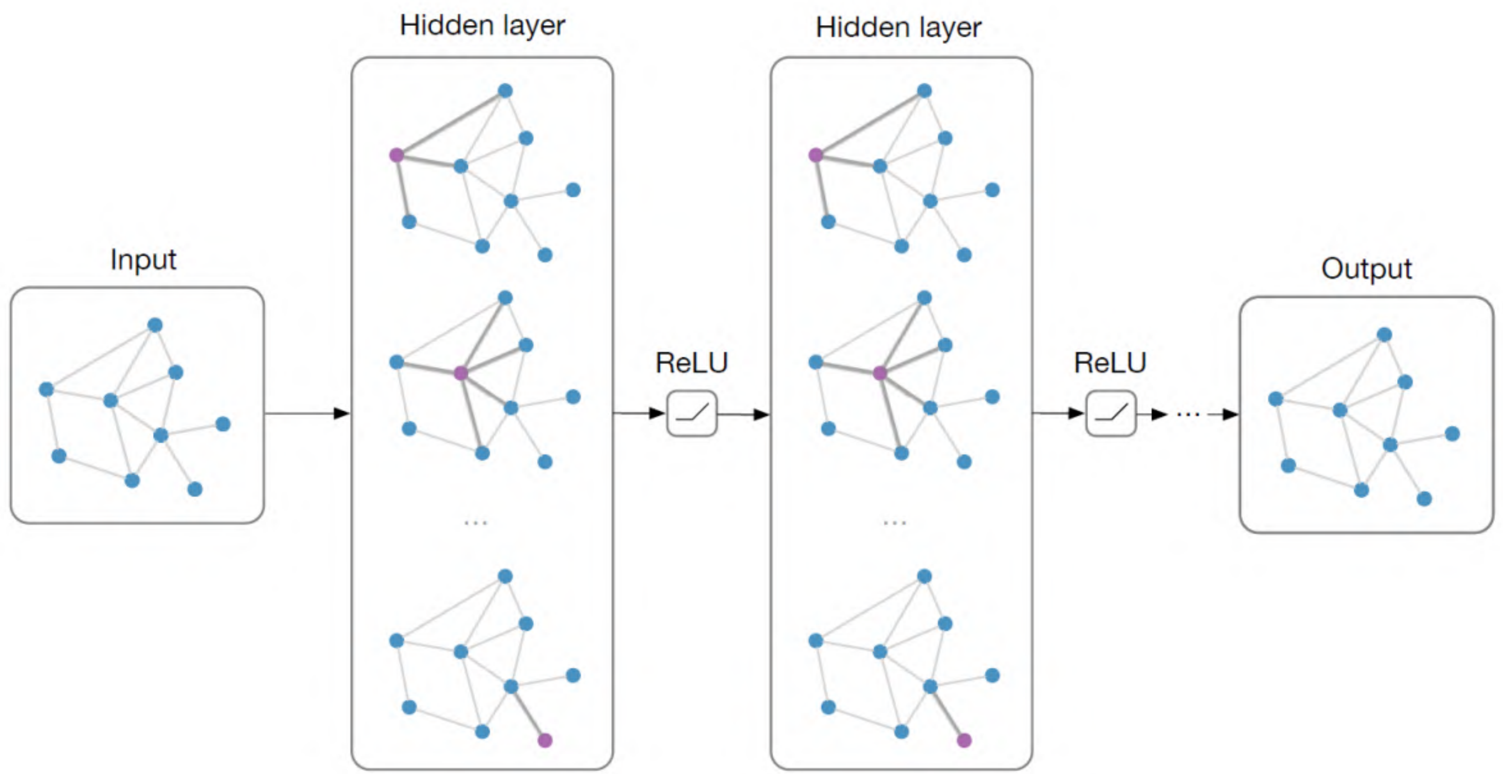
Multimedia



Information Retrieval

Images cited from: Marcheggiani and Titov, EMNLP 2017; Zellers et al., CVPR 2018, Hudson and Manning, NeurIPS 2019.

Graph Neural Network

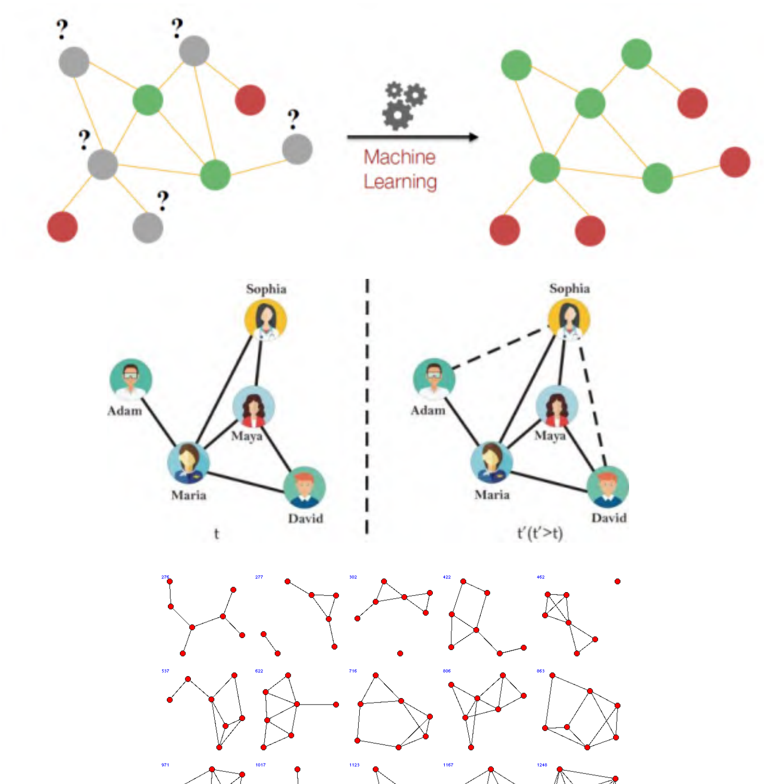


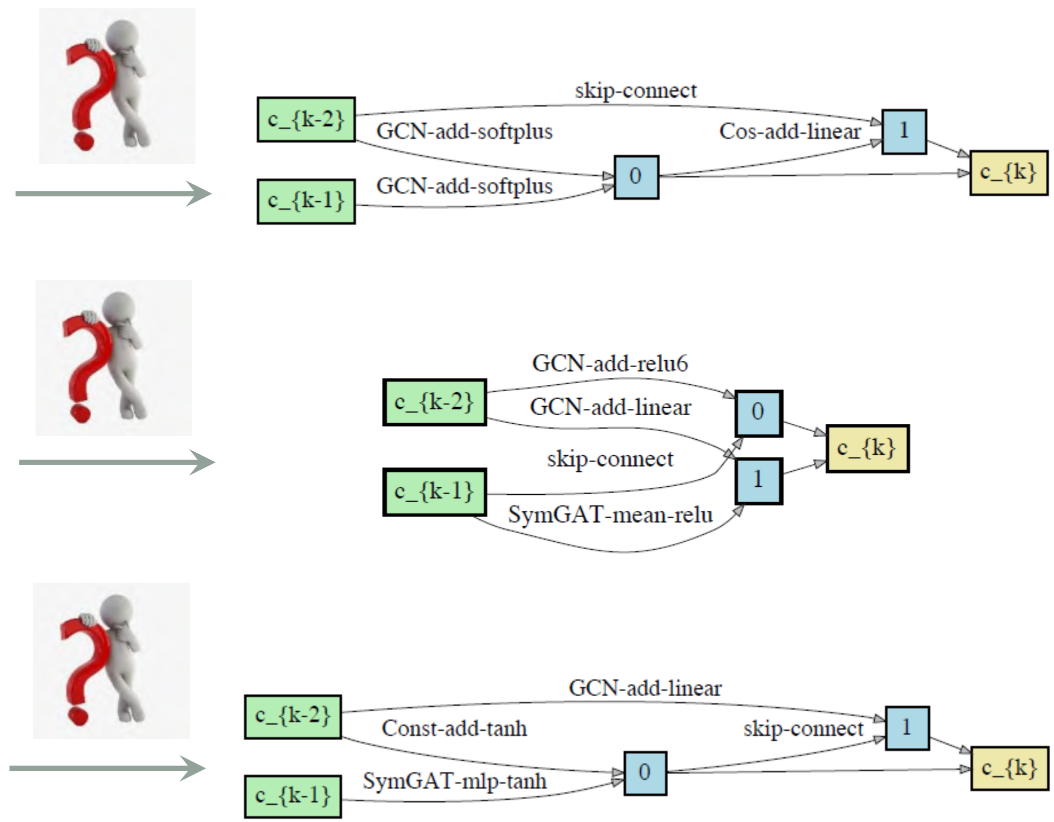
- Design neural networks directly applicable for graphs for end-to-end learning
- Message-passing framework: nodes exchange messages along structures

Image cited from Kipf and Welling, ICLR 2017

The Existing Problems in Traditional Graph Learning Methods

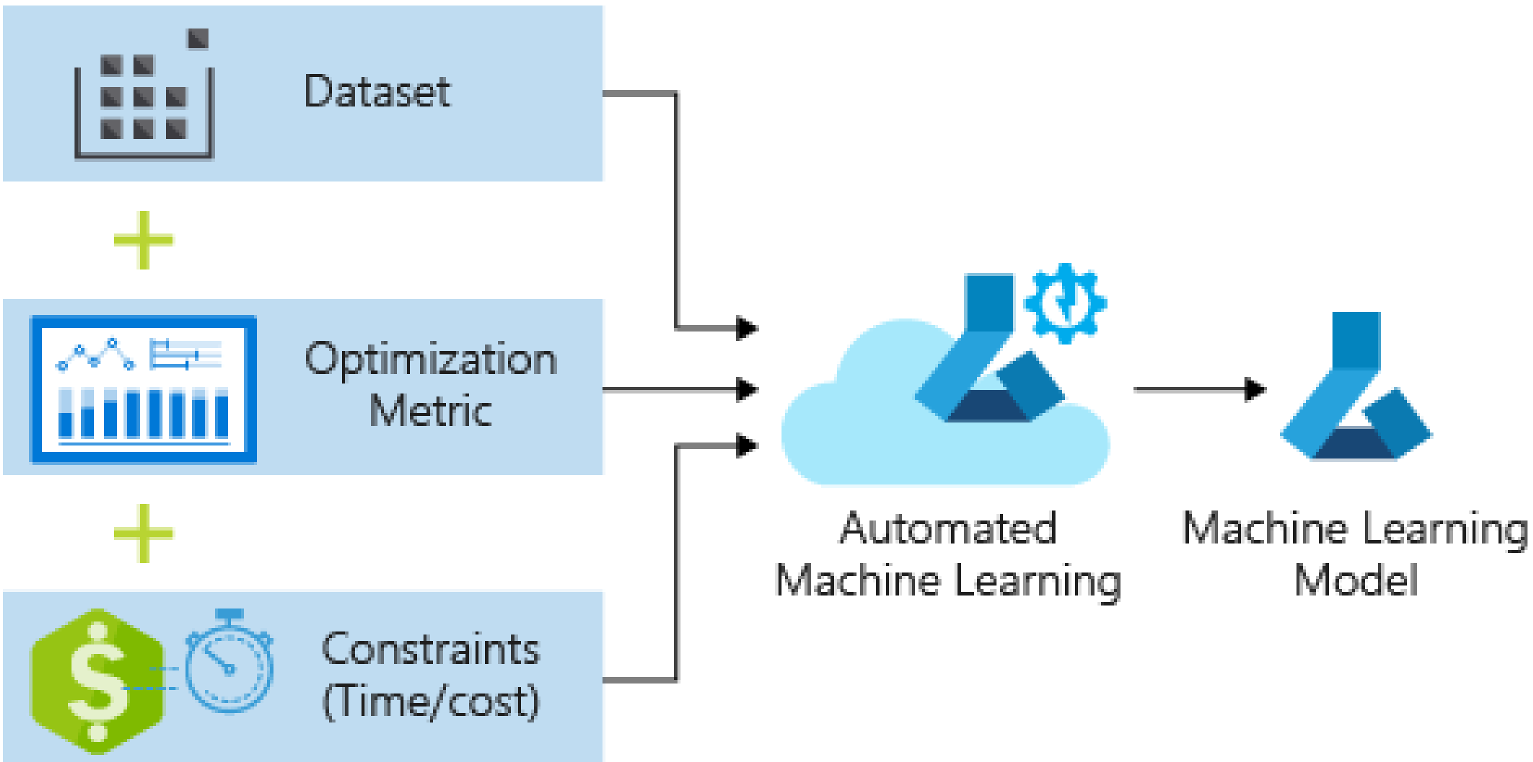
- Manually design architectures and hyper-parameters through trial-and-error
- Each dataset/task is handled separately





The adaptivity of graph machine learning is limited!

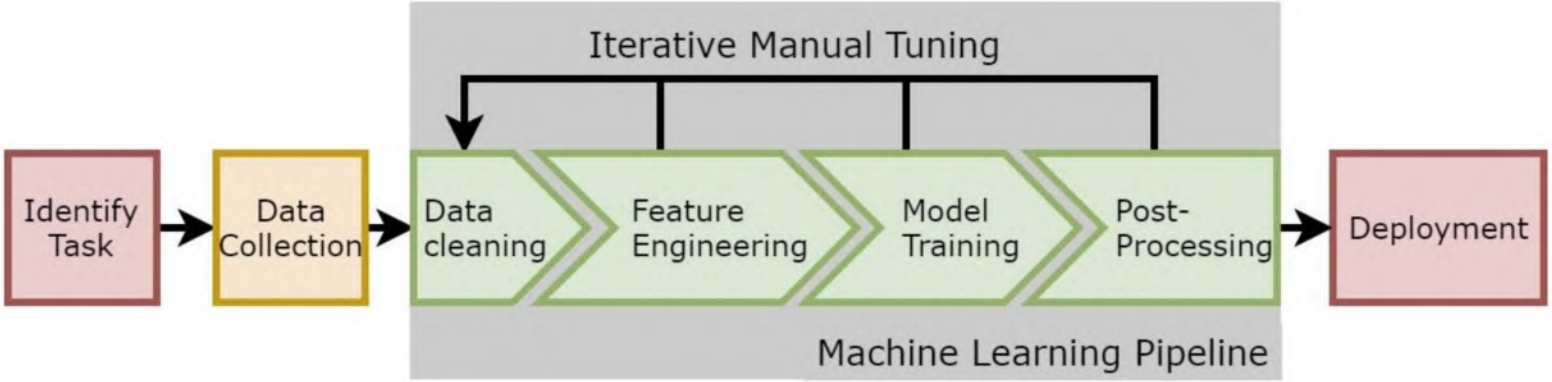
A Glance of AutoML



Design ML methods → Design AutoML methods

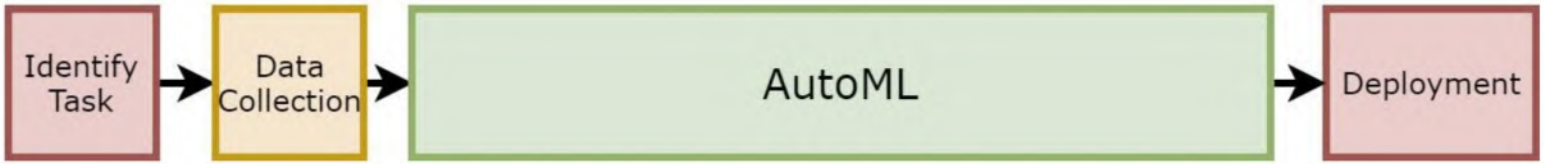
Picture credit to Microsoft Azure Machine Learning AutoML

ML vs. AutoML



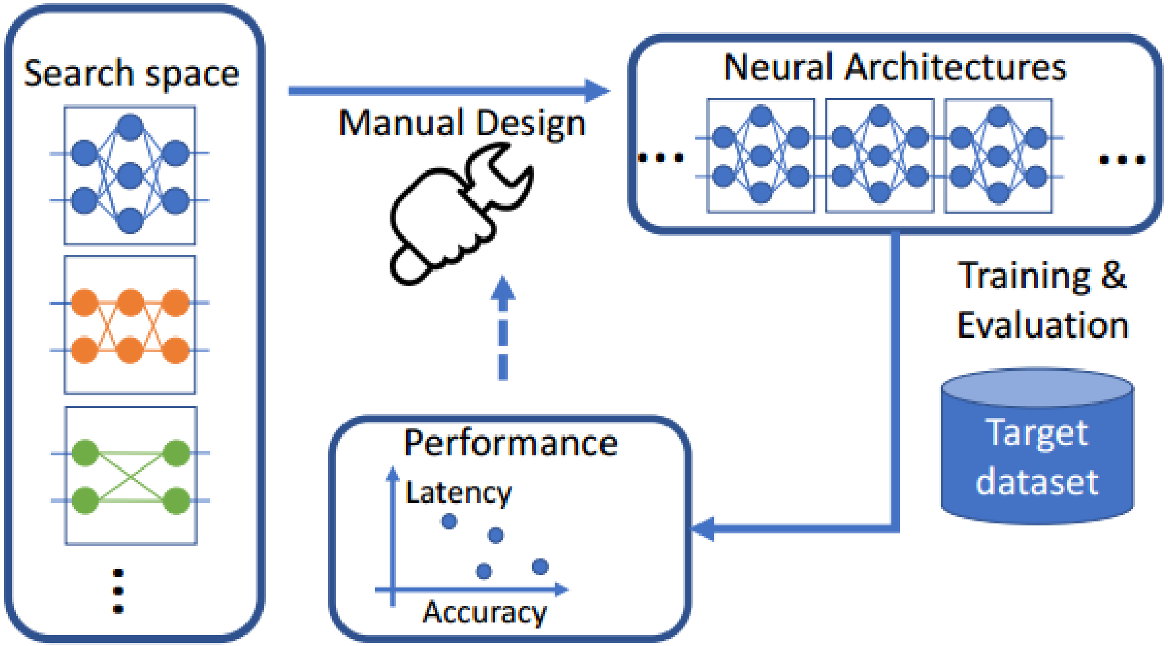
- Rely on **expert knowledge**
- **Tedious** trail-and-error
- **Limited** by human design

- **Free human** out of the loop
- **High** optimization **effectiveness**
- **Discover & extract** patterns and combinations **automatically**

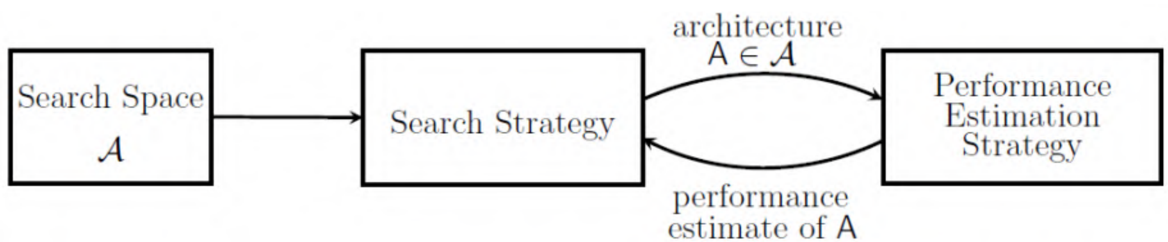


Graph Neural Architecture Search (NAS)

- NAS: automatically learn the best neural architecture



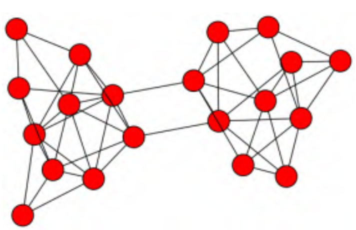
- Key designs



FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search, CVPR 2019
Neural Architecture Search A Survey, JMLR 2019

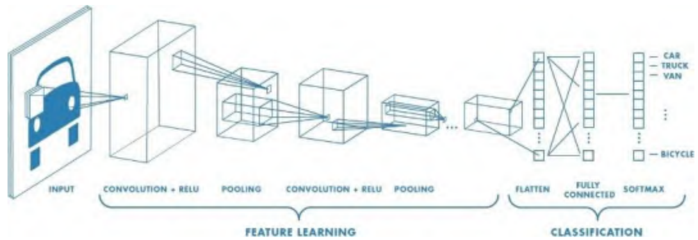
Graph NAS: Search Space

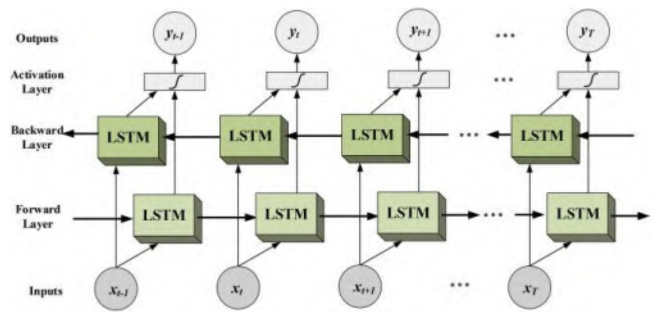
Data

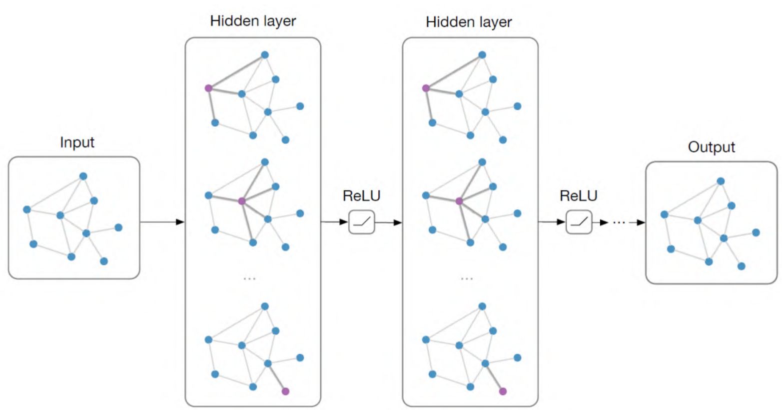


G = (V, E)

NN architecture







Search Space

- zeroize
- skip-connect
- 1×1 conv
- 3x3 conv
- 3x3 avg pool

predefined operation set

Linear: $f(x_1, ..., x_n) = W_1 x_1 + .. + W_n x_n + b$,
Blending (element wise): $f(z, x, y) = z \odot x + (1 - z) \odot y$
Element wise product and sum,
Activations: Tanh, Sigmoid, and LeakyReLU.

?

NAS-Bench-201 Extending the Scope of Reproducible Neural Architecture Search, *ICLR 2020*
NAS-Bench-NLP Neural Architecture Search Benchmark for Natural Language Processing, *arXiv 2020*

Graph NAS Search Space

$$\mathbf{m}_i^{(l)} = \mathrm{AGG}^{(l)}\left(\left\{a_{ij}^{(l)}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)}, \forall j \in \mathcal{N}(i)\right\}\right)$$

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathrm{COMBINE}^{(l)}\left[\mathbf{m}_i^{(l)}, \mathbf{h}_i^{(l)}\right]\right),$$

- $\mathrm{AGG}(\cdot)$: how to aggregate information from neighbors
  - Requirement: permutation-invariant
  - Common choices: mean, max, sum, etc.
- a_{ij}: the importance of neighbors
- $\mathrm{COMBINE}(\cdot)$: how to update representation
  - Common choices: CONCAT, SUM, MLP, etc.
- $\sigma(\cdot)$: Sigmoid, ReLU, tanh, etc.

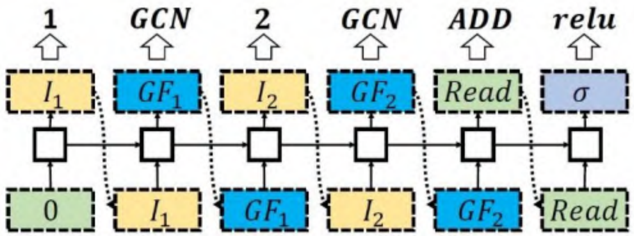
Type	Formulation
CONST	$a_{ij}^{\text{const}} = 1$
GCN	$a_{ij}^{\text{gcn}} = \frac{1}{\sqrt{\lvert\mathcal{N}(i)\rvert\lvert\mathcal{N}(j)\rvert}}$
GAT	$a_{ij}^{\text{gat}} = \text{LeakyReLU}\left(\text{ATT}\left(\mathbf{W}_a\left[\mathbf{h}_i, \mathbf{h}_j\right]\right)\right)$
SYM-GAT	$a_{ij}^{\text{sym}} = a_{ij}^{\text{gat}} + a_{ji}^{\text{gat}}$
COS	$a_{ij}^{\text{cos}} = \cos\left(\mathbf{W}_a\mathbf{h}_i, \mathbf{W}_a\mathbf{h}_j\right)$
LINEAR	$a_{ij}^{\text{lin}} = \tanh\left(\text{sum}\left(\mathbf{W}_a\mathbf{h}_i + \mathbf{W}_a\mathbf{h}_j\right)\right)$
GENE-LINEAR	$a_{ij}^{\text{gene}} = \tanh\left(\text{sum}\left(\mathbf{W}_a\mathbf{h}_i + \mathbf{W}_a\mathbf{h}_j\right)\right)\mathbf{W}'_a$

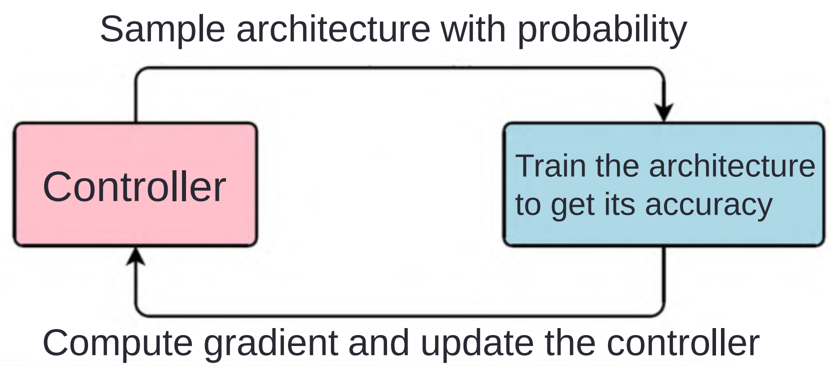
- Dimensionality of $h_i^{(l)}$, the number of attention heads (when using attention)

Graph Neural Architecture Search, *IJCAI 2020.*

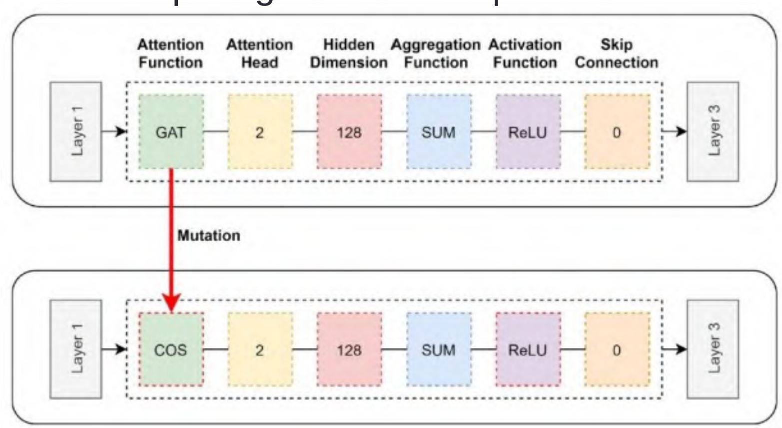
Graph NAS Search Strategy

- Most previous general NAS search strategies can be directly applied
  - Reinforcement learning
    - Controller:

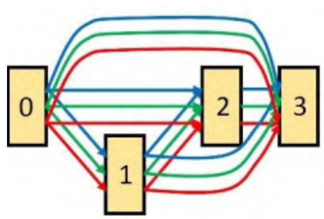




  - Evolutionary
    - Define how to evolve and how to select



  - Differentiable
    - Super-net: mix all possible operations



$$\mathbf{y} = o^{(x,y)}(\mathbf{x}) = \sum\nolimits_{o\in\mathcal{O}} \frac{\exp(\mathbf{z}_o^{(x,y)})}{\sum_{o'\in\mathcal{O}} \exp(\mathbf{z}_{o'}^{(x,y)})} o(\mathbf{x})$$

$$\alpha = \alpha - \nabla_\alpha \mathcal{L}_{val}(\mathbf{W}(\alpha), \alpha)$$

$$\mathbf{W} = \mathbf{W} - \nabla_\mathbf{W} \mathcal{L}_{train}(\mathbf{W}, \alpha)$$

Survey

Automated Machine Learning on Graphs: A Survey

Ziwei Zhang*, **Xin Wang*** and **Wenwu Zhu**†
Tsinghua University, Beijing, China
zw-zhang16@mails.tsinghua.edu.cn, {xin_wang,wwzhu}@tsinghua.edu.cn

Method	Search space: Micro	Macro	Pooling	HP	Layers	Tasks: Node	Graph	Search Strategy	Performance Estimation	Other Characteristics
GraphNAS [34]	✓	✓	✗	✗	Fixed	✓	✗	RNN controller + RL	-	-
AGNN [43]	✓	✗	✗	✗	Fixed	✓	✗	Self-designed controller + RL	Inherit weights	-
SNAG [44]	✓	✓	✗	✗	Fixed	✓	✗	RNN controller + RL	Inherit weights	Simplify the micro search space
PDNAS [45]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	Single-path one-shot	-
POSE [46]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	Single-path one-shot	Support heterogenous graphs
NAS-GNN [47]	✓	✗	✗	✓	Fixed	✓	✗	Evolutionary algorithm	-	-
AutoGraph [48]	✓	✓	✗	✗	Various	✓	✗	Evolutionary algorithm	-	-
GeneticGNN [49]	✓	✗	✗	✓	Fixed	✓	✗	Evolutionary algorithm	-	-
EGAN [50]	✓	✓	✗	✗	Fixed	✓	✓	Differentiable	One-shot	Sample small graphs for efficiency
NAS-GCN [51]	✓	✓	✓	✗	Fixed	✗	✓	Evolutionary algorithm	-	Handle edge features
LPGNAS [52]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	Single-path one-shot	Search for quantisation options
You *et al.* [53]	✓	✓	✗	✓	Various	✓	✓	Random search	-	Transfer across datasets and tasks
SAGS [54]	✓	✗	✗	✗	Fixed	✓	✓	Self-designed algorithm	-	-
Peng *et al.* [55]	✓	✗	✗	✗	Fixed	✗	✓	CEM-RL [56]	-	Search spatial-temporal modules
GNAS[57]	✓	✓	✗	✗	Various	✓	✓	Differentiable	One-shot	-
AutoSTG[58]	✗	✓	✗	✗	Fixed	✓	✗	Differentiable	One-shot+meta learning	Search spatial-temporal modules
DSS[59]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	One-shot	Dynamically update search space
SANE[60]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	One-shot	-
AutoAttend[61]	✓	✓	✗	✗	Fixed	✓	✓	Evolutionary algorithm	One-shot	Cross-layer attention

Table 1: A summary of different NAS methods for graph machine learnings.

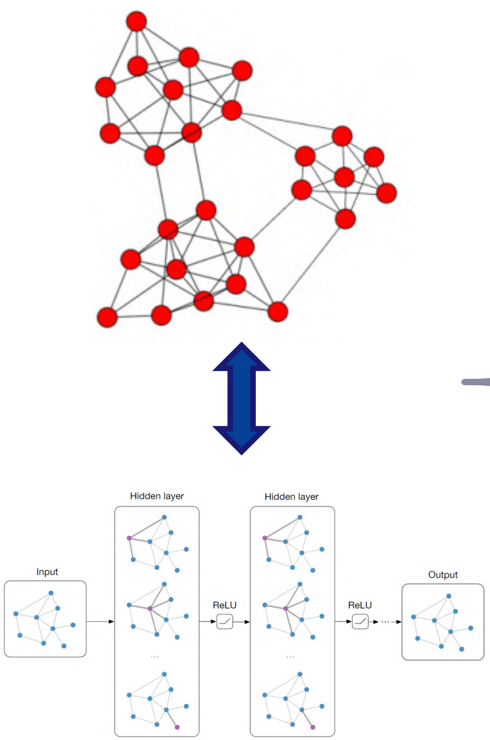
Paper collection: https://github.com/THUMNLab/awesome-auto-graph-learning
Tutorial KDD 2021: https://zw-zhang.github.io/files/2021_KDD_AutoMLonGraph.pdf

Automated Machine Learning on Graphs: A Survey. ***IJCAI, 2021.***

Challenges for the Existing Methods

- The existing methods partially solve the applicability problem

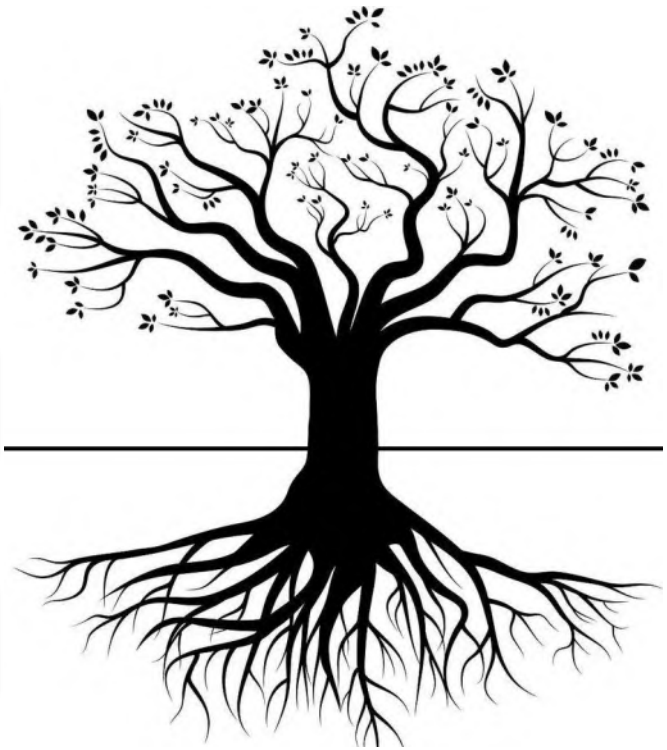
…but GraphNAS has many unique and unsolved challenges



Graph Structure

Scalability

Generalization

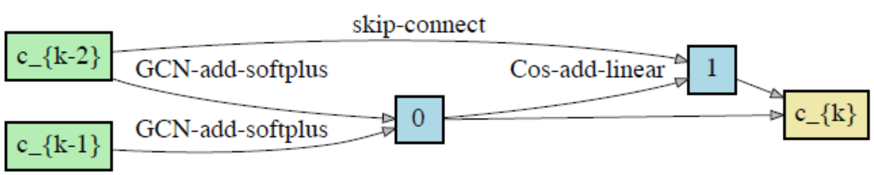


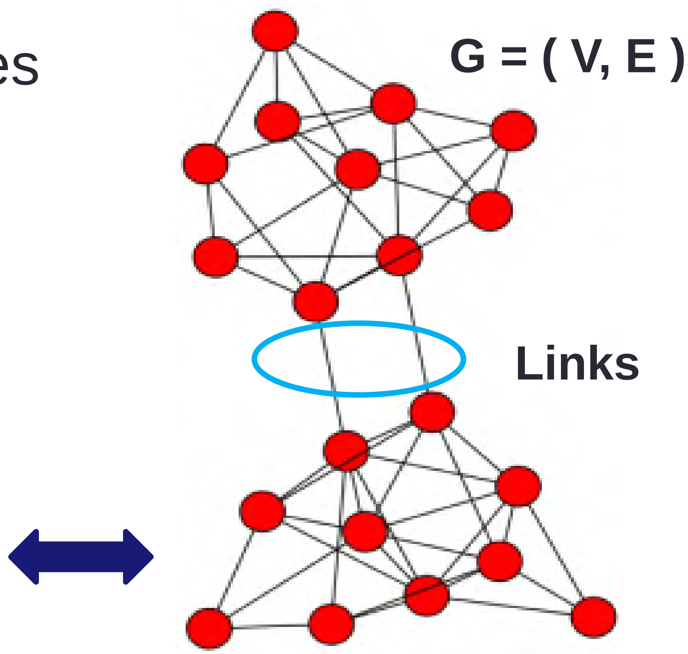
Challenges: Graph Structure

- Graph structure is the key to GraphNAS
- Previous works assume fixed structures

→ Is the input graph structure optimal?

→ How to select architectures and graph structures that suit each other?





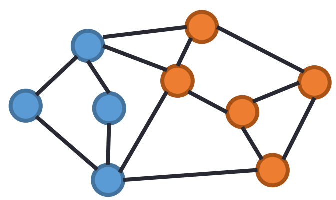
Challenge: how to theoretically model graph structure in GraphNAS

Analysis

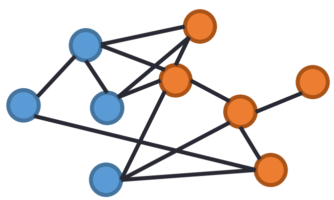
- Different operations fit graphs with different amount of information

Theorem 2 *Under our synthetic graph setting, let n be the number of edges connected the target node, the relative distance between the centers of two classes is $|D|$, which follows $D \sim \mathcal{N}(0, \beta^2)$. Then, the probability of that linear operation gives more accurate prediction than GCN on the target node is* $P = \Phi[\frac{\sqrt{2}n|D|}{(\delta+1)\sqrt{(n+1)(n+2)}}]$.

- Factors to determine the amount of information: signal to noise ratio

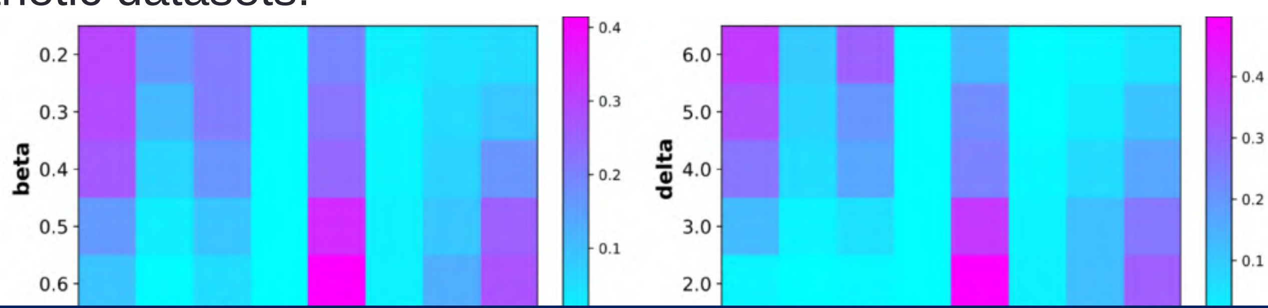


Less noise



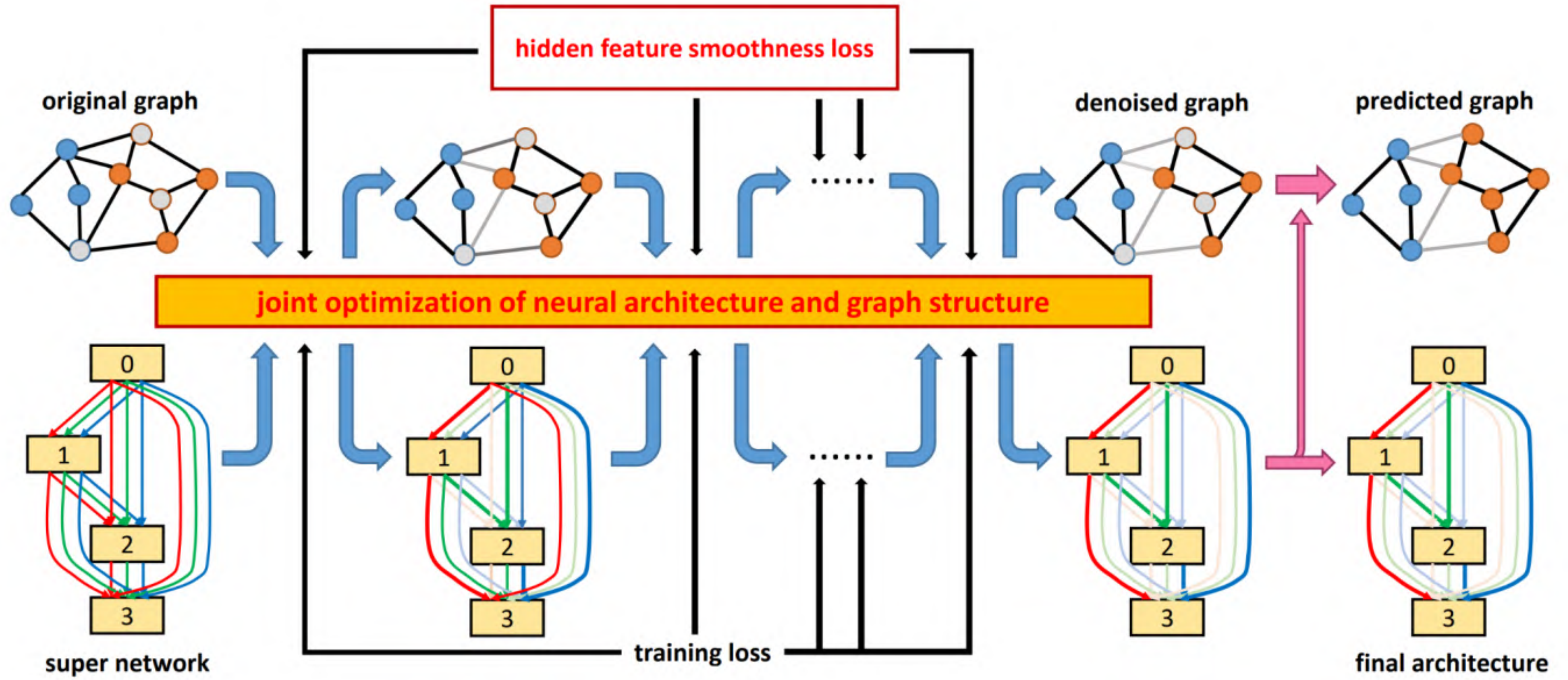
Large noise

- Synthetic datasets:



How to reduce structural noises while searching architectures?

GASSO: Jointly Learn Graph Structure and Neural Architecture



Learn graph structure and neural architecture through a joint optimization scheme

Graph Differentiable Architecture Search with Structure Learning. ***NeurIPS, 2021.***

GASSO: Model

- Formulation: tri-level optimization

$$\min_{\mathcal{A}} \mathcal{L}_{val}(W^*, \mathcal{A}, G^*)$$
$$s.t. \quad G^* = argmin_G \mathcal{L}_s(W^*, \mathcal{A}, G),$$
$$W^* = argmin_W \mathbb{E}_{\mathcal{A} \in \Gamma(\mathcal{A})} \mathcal{L}_{train}(W, \mathcal{A}, G).$$

  - Feature Smoothness Constraint

$$\mathcal{L}_s = \lambda \sum_{i,j}^{N} G_{ij} \parallel \mathbf{h}_i - \mathbf{h}_j \parallel_2 + \sum_{i,j}^{N} (G_{ij} - G_{o,ij})^2,$$

    - Mask original edges: $G = G_o \odot M$

- Possible extensions: adding edges
  - Challenge: time complexity, there are $O(n^2)$ possible edges

GASSO: Experiments

- Experiments on graph benchmarks

Dataset	Cora	Citeseer	Pubmed
GCN[†]	87.40	79.20	88.40
GAT[†]	87.26 ± 0.08	77.82 ± 0.11	86.83 ± 0.11
ARMA[†]	86.06 ± 0.05	76.50 ± 0.00	88.70 ± 0.24
DropEdge[†]	87.60 ± 0.05	78.57 ± 0.00	87.34 ± 0.24
DARTS	86.18 ± 0.36	74.96 ± 0.10	88.38 ± 0.18
GDAS	85.48 ± 0.30	74.20 ± 0.11	89.50 ± 0.14
ASAP	85.21 ± 0.13	75.14 ± 0.09	88.65 ± 0.10
XNAS	86.80 ± 0.14	76.33 ± 0.09	88.61 ± 0.25
GraphNAS[‡]	86.83 ± 0.56	79.05 ± 0.28	89.99 ± 0.43
GASSO	$\mathbf{87.63 \pm 0.29}$	$\mathbf{79.61 \pm 0.32}$	$\mathbf{90.52 \pm 0.24}$

- Experiments on larger graph datasets

Dataset	Physics	CoraFull	ogbn-arxiv
GCN	95.94	68.08	70.39
GAT	95.86	65.78	68.53
DARTS	95.74	68.51	69.52
GASSO	96.38	68.89	70.52

Challenge: Large-scale Graphs

Social Networks

- WeChat: 1.29 billion monthly active users (Aug 2022)
- Facebook: 2.8 billion active users (2020)

E-commerce Networks

- Millions of sellers, about 0.9 billion buyers, 10.6 trillion turnovers in China (2019)

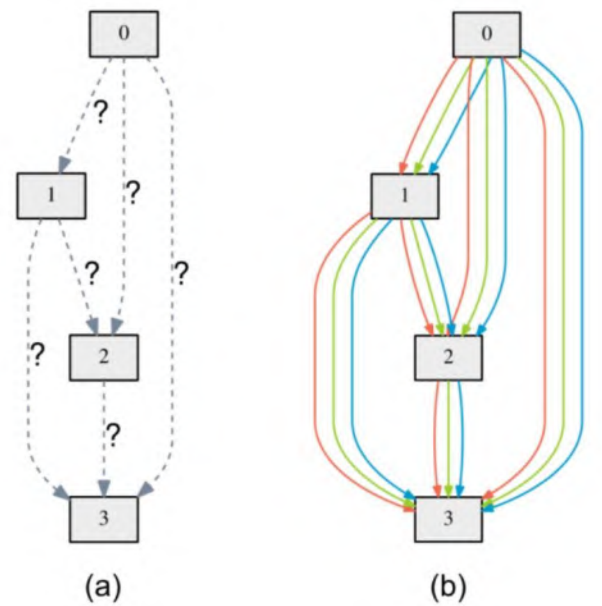
Citation Networks

- 131 million authors, 185 million publications, 754 million citations (Aminer, Aug 2022)

Challenge: how to efficiently scale to billion-scale graphs

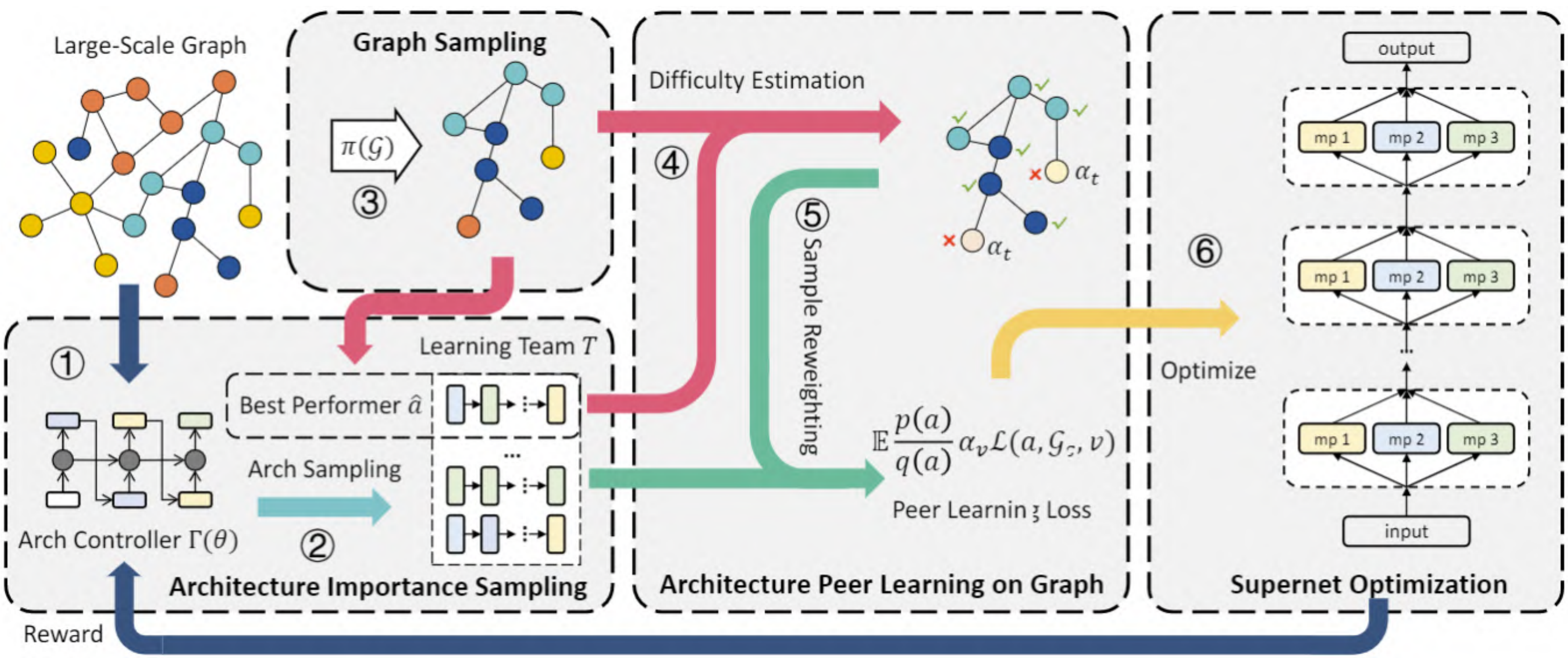
SuperNet Training

- Supernet: combine all possible operations of the search space



  - Trained by sampling architectures and back-propagations
- Supernet training for large-scale graphs:
  - Using the whole graph → computational bottleneck
  - Straight-forwardly sampling subgraphs → consistency issue

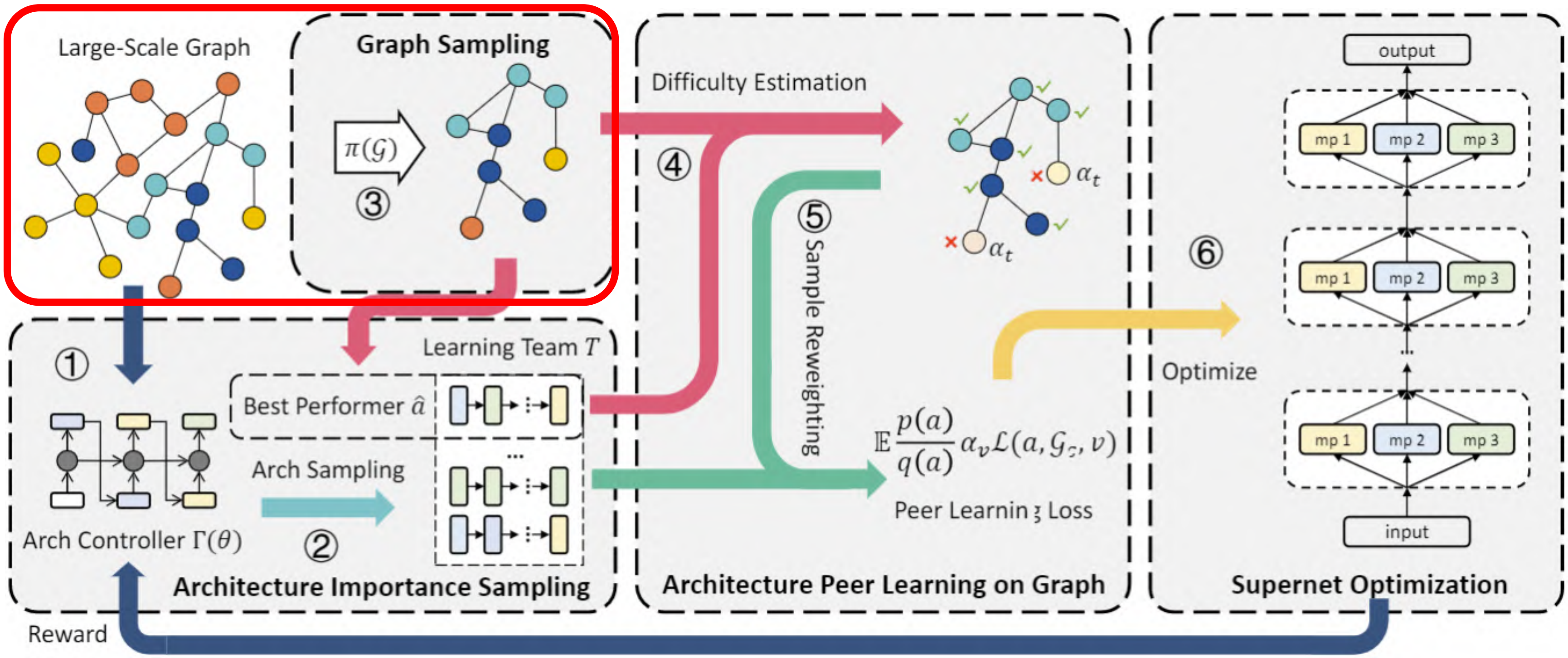
GAUSS: Large-scale Graph Neural Architecture Search



Jointly sample subgraphs and architectures to find the most suitable architecture

Large-scale Graph Neural Architecture Search. ***ICML, 2022.***

GAUSS: Architecture Importance Sampling



- **Goal**: stabilize the training of the supernet
- **Method**: important sampling of architectures
  - $\Gamma(\mathcal{A})$: proposal distribution
  - Learning proposal distribution: reinforcement learning with GRU controller

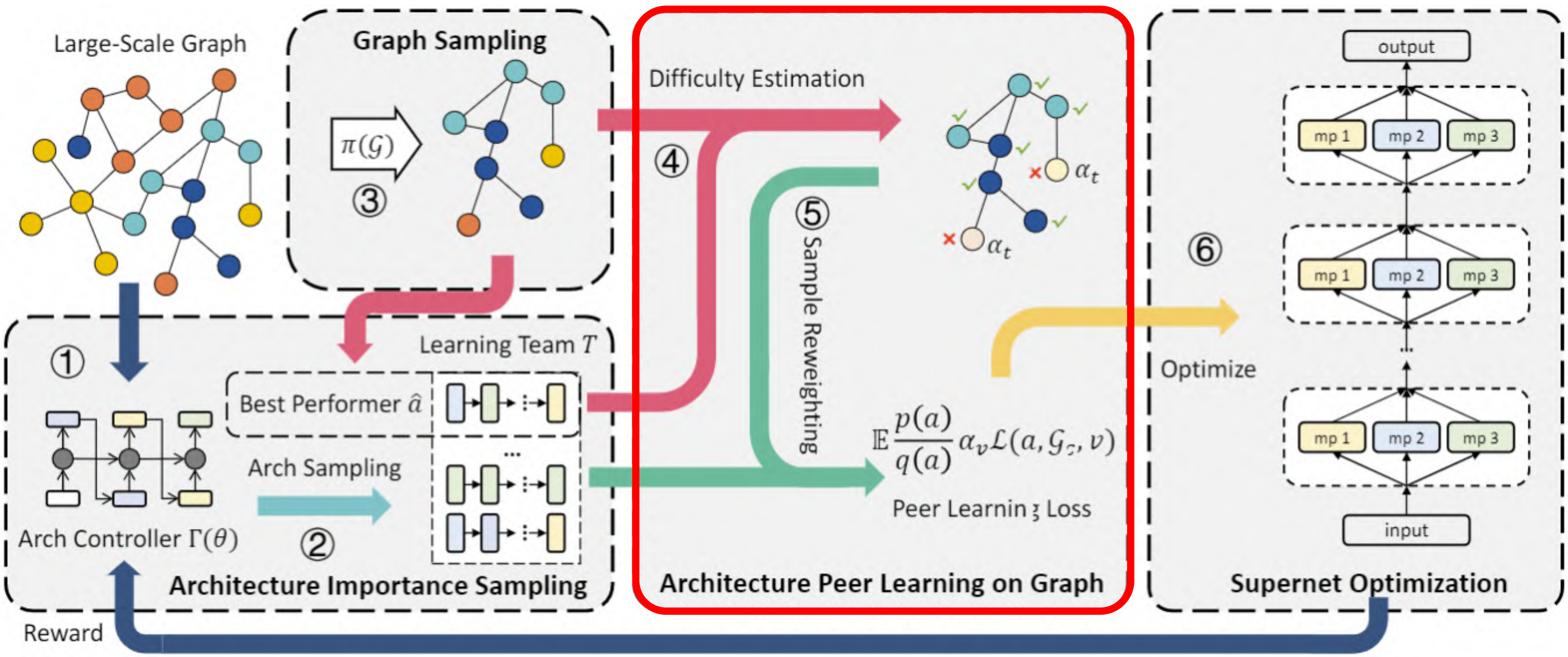
$$\mathrm{Acc}(\mathcal{A}) \triangleq \mathbb{E}_{a\in\mathcal{A}}\mathrm{Acc}_{\text{valid}}(a) = \mathbb{E}_{a\sim\Gamma(\mathcal{A})}\frac{p(a)}{q(a)}\mathrm{Acc}_{\text{valid}}(a),$$

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{0},\ x_0 = \texttt{<bos>} \\ \mathbf{h}_l &= \mathrm{GRU}(\mathrm{Emb}(x_{l-1}), \mathbf{h}_{l-1}) \quad l \in \{1, ..., L\} \\ q(x_l|x_{0:l-1}) &= \mathrm{Softmax}(\mathbf{W}\mathbf{h}_l) \quad l \in \{1, ..., L\} \\ x_l &= \mathrm{Sample}(q(x_l|x_{0:l-1})) \quad l \in \{1, ..., L\} \end{aligned}$$

Reward function: performance + regularizer

$$\theta = \mathrm{argmax}_{\theta}(\mathcal{R}(\theta) + \beta\mathcal{H}(\Gamma(\theta))),$$

GAUSS: Architecture Peer Learning on Graph



- **Goal**: smooth the optimization objective
- **Assumption**: “senior students” can teach “junior students”
- **Method**: assign weights to different samples, gradually progress from easier parts to difficult parts

$$\hat{\mathcal{L}} = \mathbb{E}_{T\in\mathcal{A}^n,\mathcal{G}_s\sim\pi(\mathcal{G})}\mathbb{E}_{a\in T,v\in\mathcal{V}_s}\alpha_v\mathcal{L}(a,\mathcal{G}_s,v)$$

$$\hat{a} = \text{argmax}_{a\in T}\text{Acc}_{\text{train}}(a,\mathcal{G}_s)$$

$$\alpha_v = \begin{cases} \alpha_t & l(\hat{a},v)\neq y_v \text{ and } p(\hat{a},v)>\lambda \\ 1 & \text{Otherwise} \end{cases},$$

$$\alpha_t = \alpha_{min}\times(1-\frac{t}{T_{total}})+\frac{t}{T_{total}},$$

GAUSS: Experiments

Dataset	#Nodes	#Edges
CS	18,333	81,894
Physics	34,493	247,962
Arxiv	169,343	1,166,243
Products	2,449,029	61,859,140
Papers100M	111,059,956	1,615,685,872

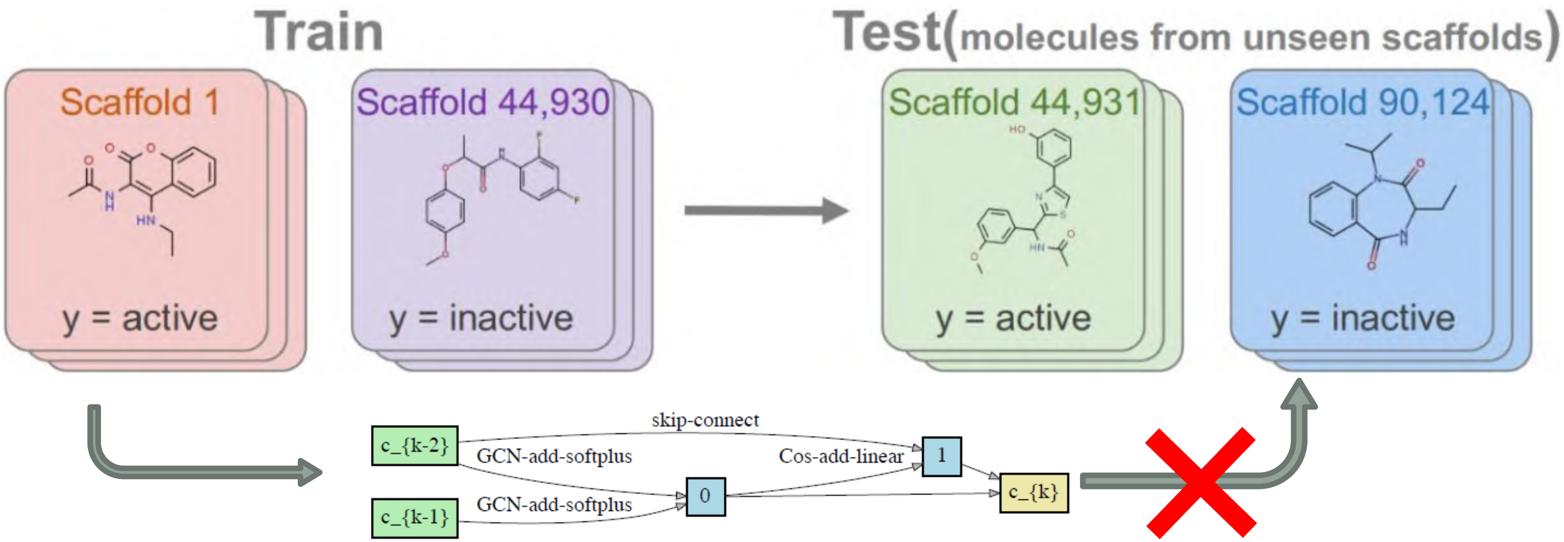
x1000

Table 2. The results of our proposed method and baseline methods. We report both the validation and test accuracy [%] over 10 runs with different seeds. OOT means out-of-time (cannot converge within 1 single GPU day), while OOM means out-of-memory (cannot run on a Tesla V100 GPU with 32GB memory). The results of the best hand-crafted and automated method are in bold, respectively.

Methods	CS		Physics		Arxiv		Products		Papers100M	
	valid	test	valid	test	valid	test	valid	test	valid	test
GCN	$94.10_{0.21}$	$93.98_{0.21}$	$96.29_{0.05}$	$96.38_{0.07}$	$72.76_{0.15}$	$71.70_{0.18}$	$91.75_{0.04}$	$80.19_{0.46}$	$70.32_{0.11}$	$67.06_{0.17}$
GAT	$93.74_{0.27}$	$93.48_{0.36}$	$96.25_{0.23}$	$96.37_{0.23}$	$73.19_{0.12}$	$71.85_{0.21}$	$90.75_{0.16}$	$80.59_{0.40}$	$70.26_{0.16}$	$67.26_{0.06}$
SAGE	$95.65_{0.07}$	$95.33_{0.11}$	$96.76_{0.10}$	$96.72_{0.07}$	$73.11_{0.08}$	$71.78_{0.15}$	$91.75_{0.04}$	$80.19_{0.46}$	$70.32_{0.11}$	$67.06_{0.17}$
GIN	$92.00_{0.43}$	$92.14_{0.34}$	$96.03_{0.11}$	$96.04_{0.15}$	$71.16_{0.10}$	$70.01_{0.33}$	$91.58_{0.10}$	$79.07_{0.52}$	$68.98_{0.16}$	$65.78_{0.09}$
GraphNAS	$94.90_{0.14}$	$94.67_{0.23}$	$96.76_{0.10}$	$96.72_{0.07}$	$72.76_{0.15}$	$71.70_{0.18}$	OOT	OOT	OOT	OOT
SGAS	$95.62_{0.06}$	$95.44_{0.06}$	$96.44_{0.10}$	$96.50_{0.11}$	$72.38_{0.11}$	$71.34_{0.25}$	OOM	OOM	OOM	OOM
DARTS	$95.62_{0.06}$	$95.44_{0.06}$	$96.21_{0.16}$	$96.40_{0.21}$	$73.43_{0.07}$	$72.10_{0.25}$	OOM	OOM	OOM	OOM
EGAN	$95.60_{0.10}$	$95.43_{0.05}$	$96.39_{0.18}$	$96.45_{0.19}$	$72.91_{0.25}$	$71.75_{0.35}$	OOM	OOM	OOM	OOM
Basic	$95.13_{0.07}$	$95.45_{0.05}$	$96.25_{0.06}$	$96.53_{0.09}$	$73.28_{0.08}$	$72.06_{0.33}$	$\mathbf{91.79}_{0.11}$	$80.56_{0.39}$	$69.49_{0.37}$	$66.24_{0.46}$
GAUSS	$\mathbf{96.08}_{0.11}$	$\mathbf{96.49}_{0.11}$	$\mathbf{96.79}_{0.06}$	$\mathbf{96.76}_{0.08}$	$\mathbf{73.63}_{0.10}$	$\mathbf{72.35}_{0.21}$	$91.60_{0.12}$	$\mathbf{81.26}_{0.36}$	$\mathbf{70.57}_{0.07}$	$\mathbf{67.32}_{0.18}$

Challenge: Distribution Shifts

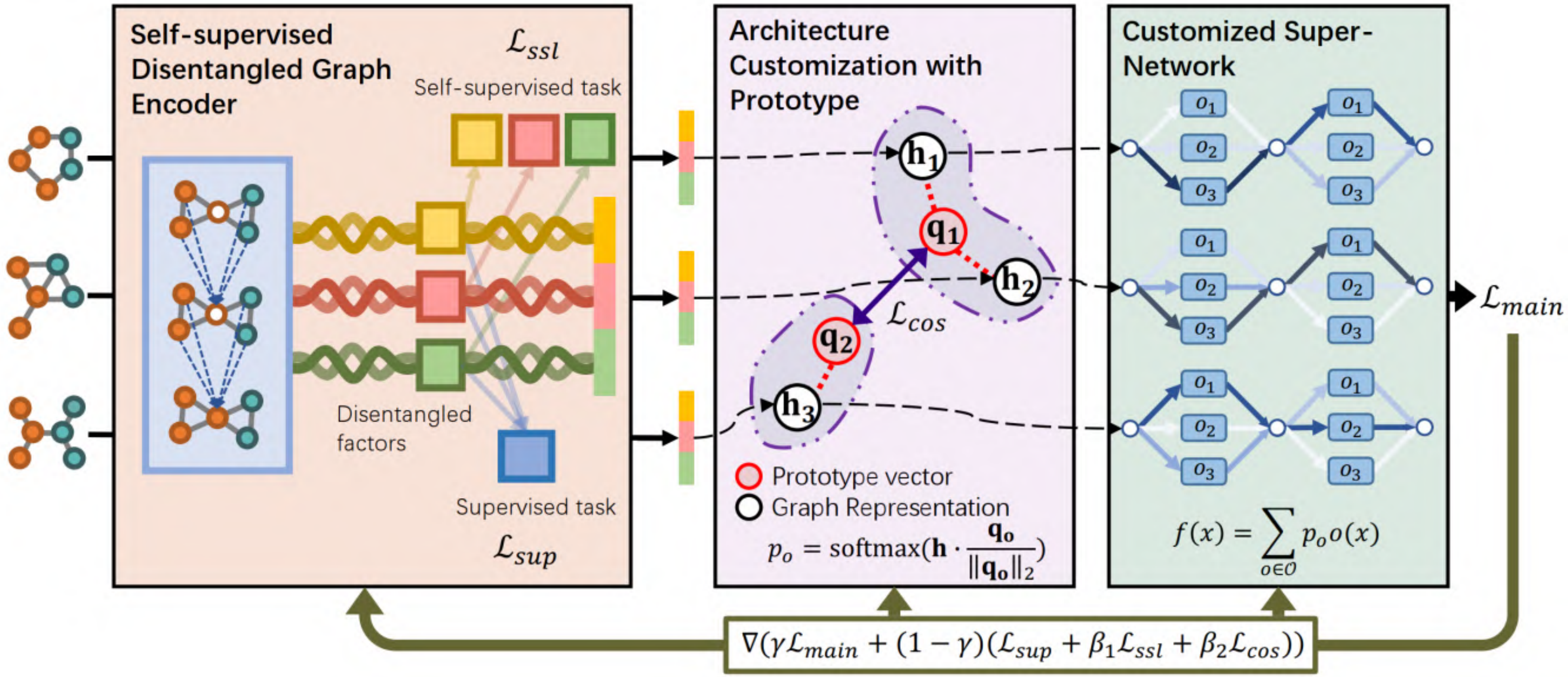
- Distribution shifts naturally exist in graph data



- Searching a fixed architecture on the training data may fail to generalize

Challenge: how to make GraphNAS capable of out-of-distribution generalization

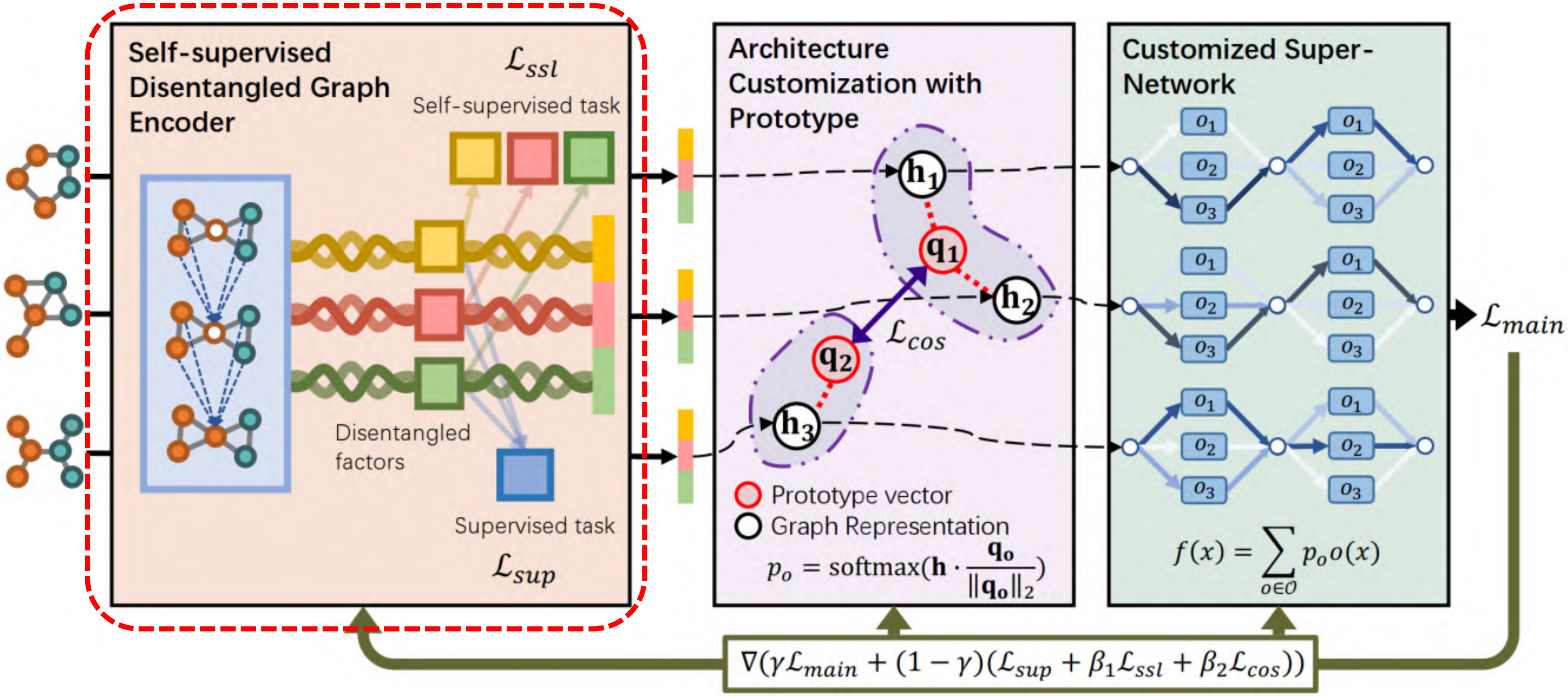
GRACES: Graph Neural Architecture Search under Distribution Shifts



Customize a unique GNN architecture for each graph instance to handle distribution shifts

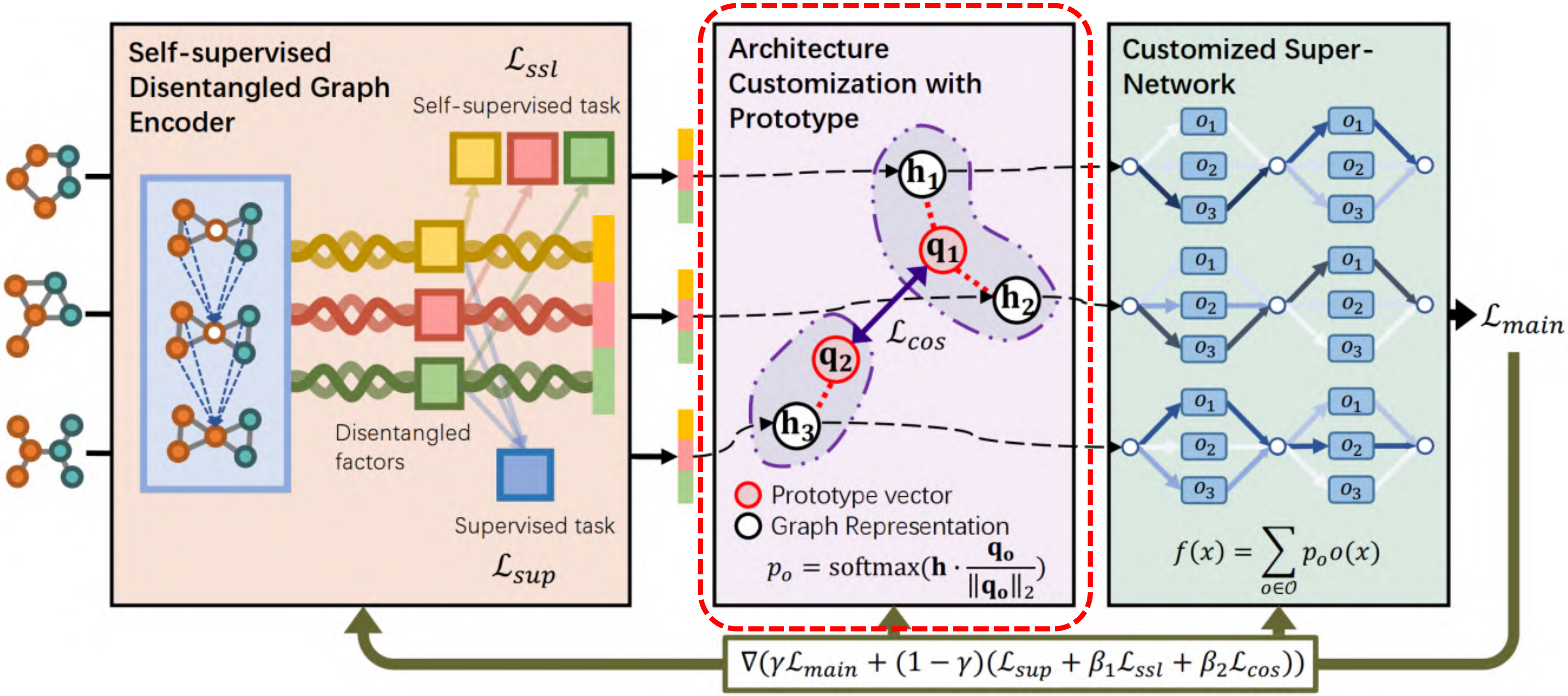
Graph Neural Architecture Search under Distribution Shifts. ***ICML, 2022.***

GRACES: Graph Encoder



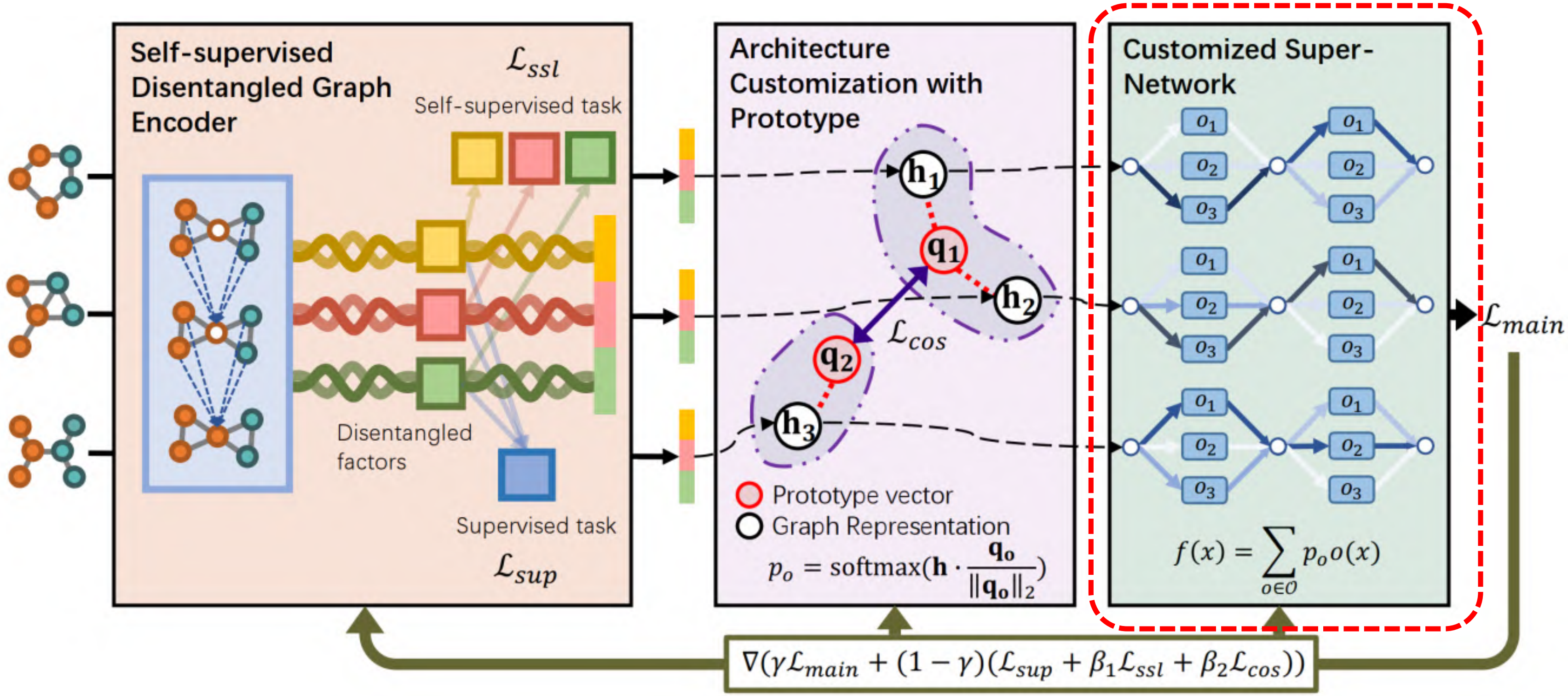
- **Goal**: learn a vector representation for each graph to reflect its characteristics
- **Challenge**: preserve diverse properties of the original graph
- **Method**: self-supervised disentangled graph encoder
  - Encoder: disentangled GNN $\mathbf{H}^{(l)} = \Big\|_{k=1}^{K} \text{GNN}(\mathbf{H}_k^{(l-1)}, \mathbf{A})$ $\quad \mathcal{L}_{sup} = \sum_{i=1}^{N_{tr}} \ell\left(\mathcal{C}\left(\mathbf{h}_i\right), y_i\right)$
  - Supervised loss: the downstream task
  - Self-supervised loss: node degree as regularization $\mathcal{L}_{ssl} = \sum_{i=1}^{N_{tr}} \sum_{k=1}^{K-1} \ell_{ssl}\left(\hat{y}_{i,k}^{ssl}, y_{i,k}^{ssl}\right)$

GRACES: Architecture Customization



- **Goal**: customize an architecture based on the graph representation
- **Assumption:** graphs with similar characteristics need similar architectures
- **Method**: prototype based architecture customization
  - Probabilities of choosing operations: $\hat{p}_o^i = \mathbf{h} \cdot \frac{\mathbf{q}_o^i}{\|\mathbf{q}_o^i\|_2}, p_o^i = \frac{\exp(\hat{p}_o^i)}{\sum_{o' \in \mathcal{O}} \exp(\hat{p}_{o'}^i)},$
  - Regularizer to avoid mode collapse: $\mathcal{L}_{cos} = \sum_i \sum_{o,o' \in \mathcal{O}, o \neq o'} \frac{\mathbf{q}_o^i \cdot \mathbf{q}_{o'}^i}{\|\mathbf{q}_o^i\|_2 \|\cdot \mathbf{q}_{o'}^i\|_2}$

GRACES: Learning Architecture Parameters



- **Goal**: learn parameters for the customized architectures
- **Method**: customized super-network $f^i(\mathbf{x}) = \sum_{o \in \mathcal{O}} p_o^i o(\mathbf{x})$
- **Loss functions:**

$$\mathcal{L} = \gamma \mathcal{L}_{main} + (1-\gamma)\mathcal{L}_{reg}$$

$$\mathcal{L}_{reg} = \mathcal{L}_{sup} + \beta_1 \mathcal{L}_{ssl} + \beta_2 \mathcal{L}_{cos}$$

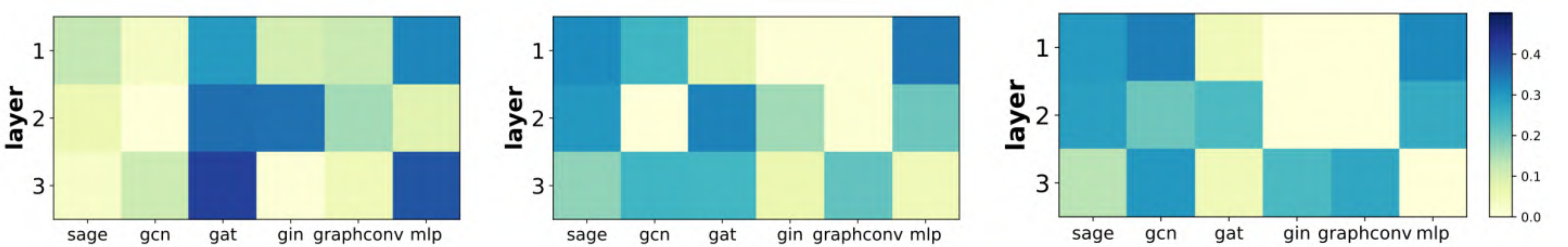
GRACES: Experiments

Synthetic OOD graph datasets

bias	$b = 0.7$	$b = 0.8$	$b = 0.9$
GCN	$48.39_{\pm 1.69}$	$41.55_{\pm 3.88}$	$39.13_{\pm 1.76}$
GAT	$50.75_{\pm 4.89}$	$42.48_{\pm 2.46}$	$40.10_{\pm 5.19}$
GIN	$36.83_{\pm 5.49}$	$34.83_{\pm 3.10}$	$37.45_{\pm 3.59}$
SAGE	$46.66_{\pm 2.51}$	$44.50_{\pm 5.79}$	$44.79_{\pm 4.83}$
GraphConv	$47.29_{\pm 1.95}$	$44.67_{\pm 5.88}$	$44.82_{\pm 4.84}$
MLP	$48.27_{\pm 1.27}$	$46.73_{\pm 3.48}$	$46.41_{\pm 2.34}$
ASAP	$54.07_{\pm 13.85}$	$48.32_{\pm 12.72}$	$43.52_{\pm 8.41}$
DIR	$50.08_{\pm 3.46}$	$48.22_{\pm 6.27}$	$43.11_{\pm 5.43}$
random	$45.92_{\pm 4.29}$	$51.72_{\pm 5.38}$	$45.89_{\pm 5.09}$
DARTS	$50.63_{\pm 8.90}$	$45.41_{\pm 7.71}$	$44.44_{\pm 4.42}$
GNAS	$55.18_{\pm 18.62}$	$51.64_{\pm 19.22}$	$37.56_{\pm 5.43}$
PAS	$52.15_{\pm 4.35}$	$43.12_{\pm 5.95}$	$39.84_{\pm 1.67}$
GRACES	$\mathbf{65.72}_{\pm 17.47}$	$\mathbf{59.57}_{\pm 17.37}$	$\mathbf{50.94}_{\pm 8.14}$

Real-world OOD graph datasets

dataset	hiv	sider	bace
GCN	$75.99_{\pm 1.19}$	$59.84_{\pm 1.54}$	$68.93_{\pm 6.95}$
GAT	$76.80_{\pm 0.58}$	$57.40_{\pm 2.01}$	$75.34_{\pm 2.36}$
GIN	$77.07_{\pm 1.49}$	$57.57_{\pm 1.56}$	$73.46_{\pm 5.24}$
SAGE	$75.58_{\pm 1.40}$	$56.36_{\pm 1.32}$	$74.85_{\pm 2.74}$
GraphConv	$74.46_{\pm 0.86}$	$56.09_{\pm 1.06}$	$78.87_{\pm 1.74}$
MLP	$70.88_{\pm 0.83}$	$58.16_{\pm 1.41}$	$71.60_{\pm 2.30}$
ASAP	$73.81_{\pm 1.17}$	$55.77_{\pm 1.18}$	$71.55_{\pm 2.74}$
DIR	$77.05_{\pm 0.57}$	$57.34_{\pm 0.36}$	$76.03_{\pm 2.20}$
DARTS	$74.04_{\pm 1.75}$	$60.64_{\pm 1.37}$	$76.71_{\pm 1.83}$
PAS	$71.19_{\pm 2.28}$	$59.31_{\pm 1.48}$	$76.59_{\pm 1.87}$
GRACES	$\mathbf{77.31}_{\pm 1.00}$	$\mathbf{61.85}_{\pm 2.56}$	$\mathbf{79.46}_{\pm 3.04}$

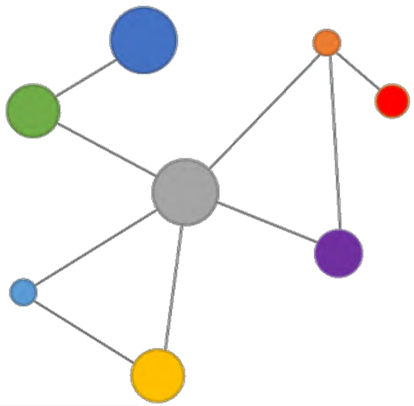


(a) *Tree*-based graphs (b) *Ladder*-based graphs (c) *Wheel*-based graphs

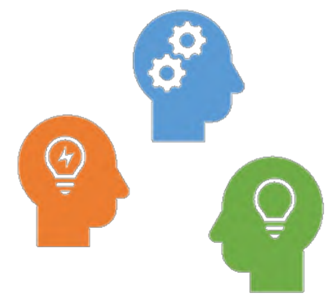
Customization of architectures

Introduction – AutoGL

- We design an autoML framework & toolkit for machine learning on graphs.

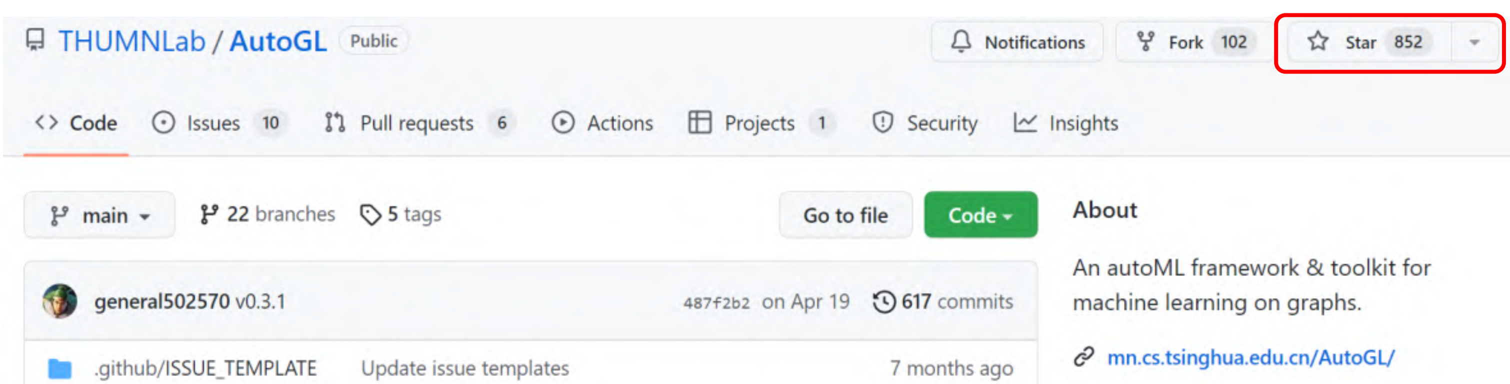


AutoGL

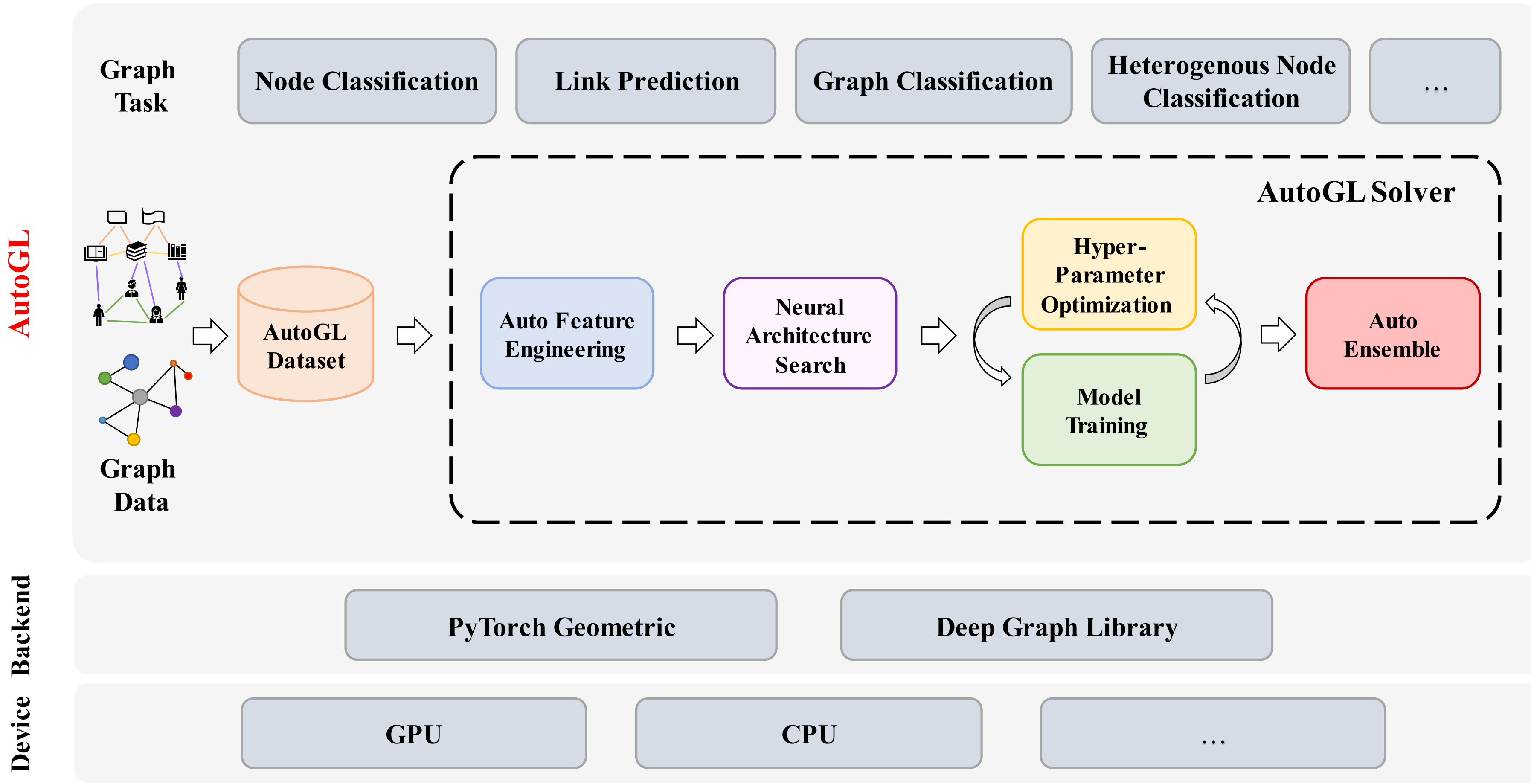


Open source | **Easy to use** | **Flexible to be extended**

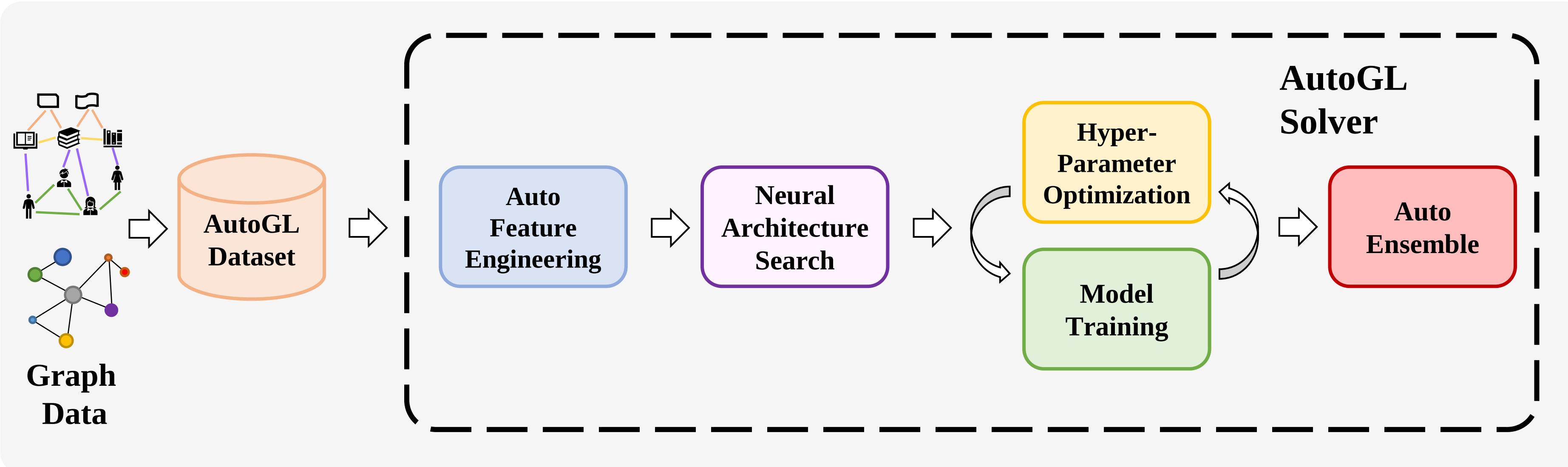
https://mn.cs.tsinghua.edu.cn/AutoGL
https://github.com/THUMNLab/AutoGL



Overall Framework

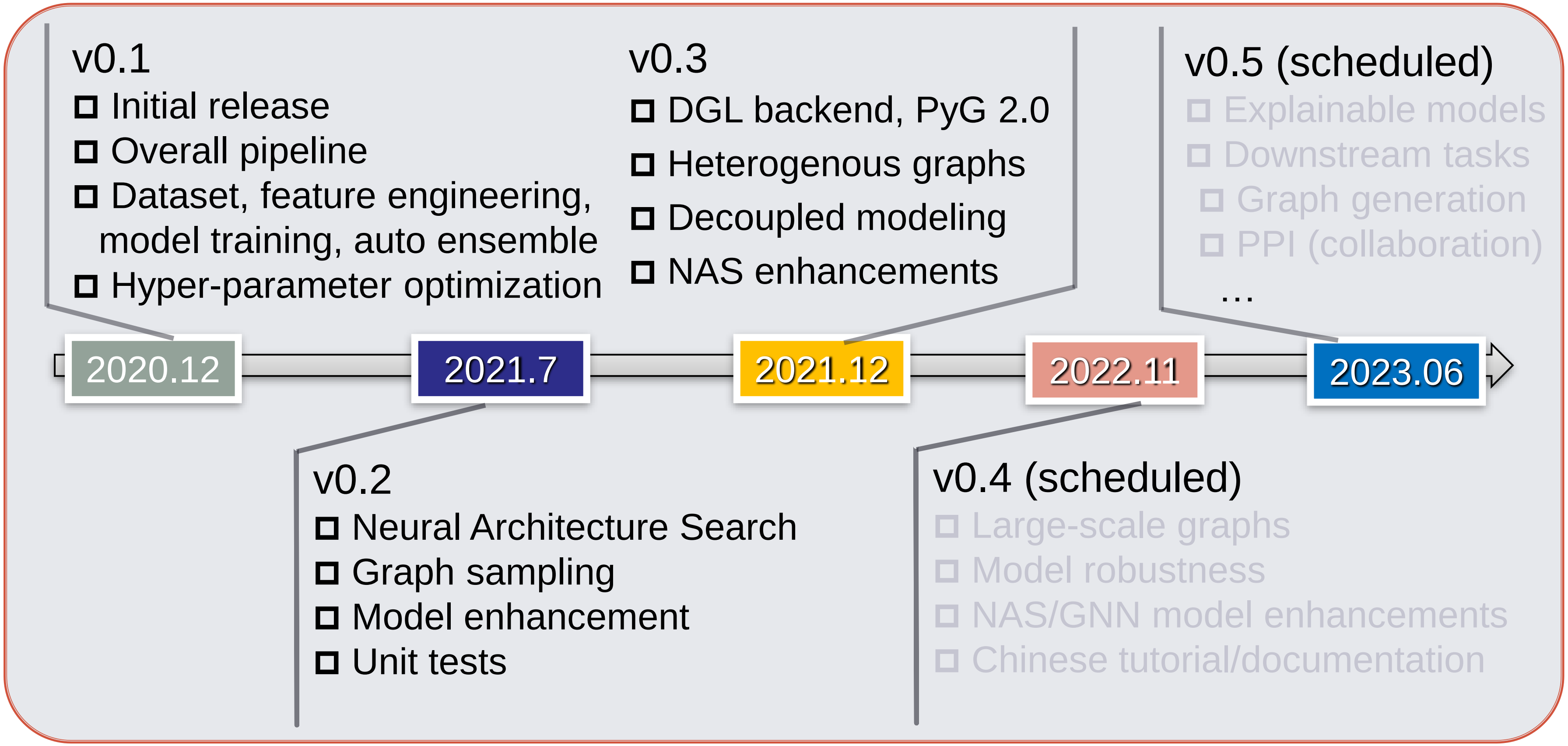


Modular Design



- Key modules:
  - AutoGL Dataset: manage graph datasets
  - AutoGL Solver: a high-level API to control the overall pipeline
  - Five functional modules:
    - Auto Feature Engineering
    - Neural Architecture Search
    - Hyper-parameter Optimization
    - Model Training
    - Auto Ensemble

AutoGL Roadmap



- Team member (~15)
  - Architect: Chaoyu Guan (v0.1-v0.3), Yijian Qin (v0.4-v0.5)
  - Programmer: Haoyang Li, Zeyang Zhang, Heng Chang, Zixin Sun, Beini Xie, Jie Cai, Zizhao Zhang, Jiyan Jiang, Yao Yang, Fang Shen
  - Tester: Yipeng Zhang, Peiwen Li

Media Coverage

WHO WE ARE | ADVERTISE | HIRING SERVICES | CONFERENCES | RESEARCH | VIDEOS | HACKATHONS | ACADEMIC RANKINGS | CAREERS | CONTACT US

Meet AutoGL: The First Ever AutoML Framework for Graph Datasets

31/12/2020

MARKTECHPOST.COM

Home | Interview | AI Shorts | Tutorials | Unicorns | University Research | AI

Home › Technology › AI Shorts › Researchers From Tsinghua University Introduces AutoGL: An AutoML Fran

Technology | AI Shorts | Artificial Intelligence | AutoML | Editors Pick | Guest Post | Machine Learning | Tech News | University Research | Tsinghua University | Uncategorized

Researchers From Tsinghua University Introduces AutoGL: An AutoML Framework For Machine Learning On Graph

By **Kriti Maloo** - January 26, 2021

澎湃 THE PAPER

精选 | 视频 | 时事 | 财经 | 澎湃号 | 思想 | 生活 | 战疫 | 问吧 | 订阅

清华发布首个自动图学习框架，或有助于蛋白质建模和新药发现

澎湃新闻记者 张唯

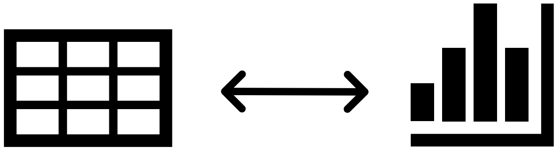
2020-12-23 17:47 来源：澎湃新闻

当前，人工智能领域的自动图机器学习研究悄然兴起，小到蛋白质分子结构，大到城市交通网络，都有自动图机器学习的用武之地。

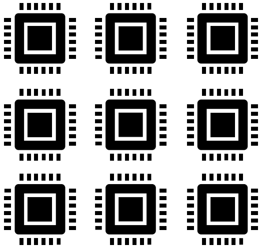
澎湃新闻（www.thepaper.cn）记者从清华大学计算机系朱文武教授领导的网络与媒体实验室获悉，该实验室于2020年12月21日发布了世界首个自动图学习框架与开源工具包AutoGL。AutoGL框架及开源工具包能够为开发人员进行图学习算法设计和调优提供便利，简化图学习算法开发与应用的流程，提升图学习相关的科研和应用效率。

The Evaluation of Graph NAS Methods

- How to properly evaluate different GraphNAS algorithms
  - Incomparable and irreproducible results



  - Computationally expensive



  - Diverse evaluation protocols

NAS-Bench-Graph

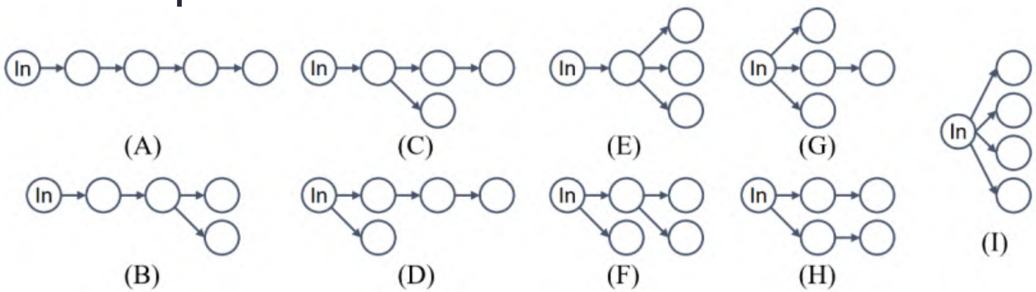
- The first tabular NAS benchmark for GraphNAS
  - Unified, Reproducible, Efficient
  - Provide detailed metrics of all architectures (exhaust 8,000 GPU hours)

Benchmark	Type	Search Space	Data	Datasets
NAS-Bench-101 [17]	Tabular	423k	CV	1
NAS-Bench-201 [4]	Tabular	6k	CV	3
NAS-Bench-1shot1 [19]	Tabular	364k	CV	1
NAS-Bench-ASR [12]	Tabular	8k	Acoustics	1
NAS-Bench-NLP [9]	Tabular	14k	NLP	2
HW-NAS-Bench [10]	Tabular	6k	CV	3
NATS-Bench [3]	Tabular	32k	CV	3
NAs-HPO-Bench-II [7]	Surrogate	192k	CV	1
NAS-Bench-MR [2]	Surrogate	10^{23}	CV	4
TransNAS-Bench [5]	Tabular	7k	CV	14
NAS-Bench-111 [16]	Surrogate	423k	CV	1
NAS-Bench-311 [16]	Surrogate	10^{18}	CV	1
NAS-Bench-Zero [1]	Tabular	34k	CV	3
Surr-NAS-Bench-FBNet [20]	Surrogate	10^{21}	CV	2
NAS-Bench-Graph	Tabular	26k	Graph	9

NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. ***NeurIPS, 2022.***

NAS-Bench-Graph: Designs

- Search space:
  - Macro space:



26,206 architectures
cover representative GNNs

  - Operations: GCN, GAT, GraphSAGE, GIN, ARMA, k-GNN, MLP
- Datasets:

Dataset	#Vertices	#Links	#Features	#Classes	Metric
Cora	2,708	5,429	1,433	7	Accuracy
CiteSeer	3,327	4,732	3,703	6	Accuracy
PubMed	19,717	44,338	500	3	Accuracy
Coauthor-CS	18,333	81,894	6,805	15	Accuracy
Coauthor-Physics	34,493	247,962	8,415	5	Accuracy
Amazon-Photo	7,487	119,043	745	8	Accuracy
Amazon-Computers	13,381	245,778	767	10	Accuracy
ogbn-arxiv	169,343	1,166,243	128	40	Accuracy
ogbn-proteins	132,534	39,561,252	8	112	ROC-AUC

9 datasets
different sizes/domains

NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. ***NeurIPS, 2022.***

NAS-Bench-Graph: Usage

- Integrated with two representative libraries: AutoGL and NNI

Library	Method	Cora	CiteSeer	PubMed	CS	Physics	Photo	Computers	arXiv	proteins
AutoGL	GNAS	$82.04_{0.17}$	$\mathbf{70.89}_{0.16}$	$77.79_{0.02}$	$90.97_{0.06}$	$92.43_{0.04}$	$92.43_{0.03}$	$84.74_{0.20}$	$72.00_{0.02}$	$\mathbf{78.71}_{0.11}$
	Auto-GNN	$81.80_{0.00}$	$70.76_{0.12}$	$77.69_{0.16}$	$\mathbf{91.04}_{0.04}$	$92.42_{0.16}$	$92.38_{0.01}$	$84.53_{0.14}$	$\mathbf{72.13}_{0.03}$	$78.54_{0.30}$
NNI	Random	$82.09_{0.08}$	$70.49_{0.08}$	$77.91_{0.07}$	$90.93_{0.07}$	$92.35_{0.05}$	$\mathbf{92.44}_{0.02}$	$84.78_{0.14}$	$72.04_{0.05}$	$78.32_{0.14}$
	EA	$81.85_{0.20}$	$70.48_{0.12}$	$\mathbf{77.96}_{0.12}$	$90.60_{0.07}$	$92.22_{0.08}$	$92.43_{0.02}$	$84.29_{0.29}$	$71.91_{0.06}$	$77.93_{0.21}$
	RL	$\mathbf{82.27}_{0.21}$	$70.66_{0.12}$	$\mathbf{77.96}_{0.09}$	$90.98_{0.01}$	$\mathbf{92.48}_{0.03}$	$92.42_{0.06}$	$\mathbf{84.90}_{0.19}$	$\mathbf{72.13}_{0.05}$	$78.52_{0.18}$
The top 5%		80.63	69.07	76.60	90.01	91.67	91.57	82.77	71.69	78.37

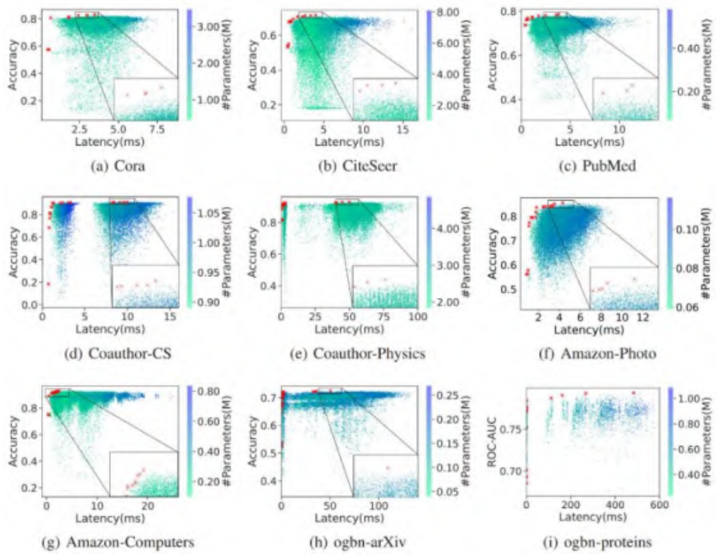
- Example: ~10 lines of codes

```
from readbench import read
bench = read('cora0.bench')  # dataset and seed
info = bench[arch.valid_hash()]
epoch = 50
info['dur'][epoch][0]   # training performance
info['dur'][epoch][1]   # validation performance
info['dur'][epoch][2]   # testing performance
info['dur'][epoch][3]   # training loss
info['dur'][epoch][4]   # validation loss
info['dur'][epoch][5]   # testing loss
info['dur'][epoch][6]   # best performance
```

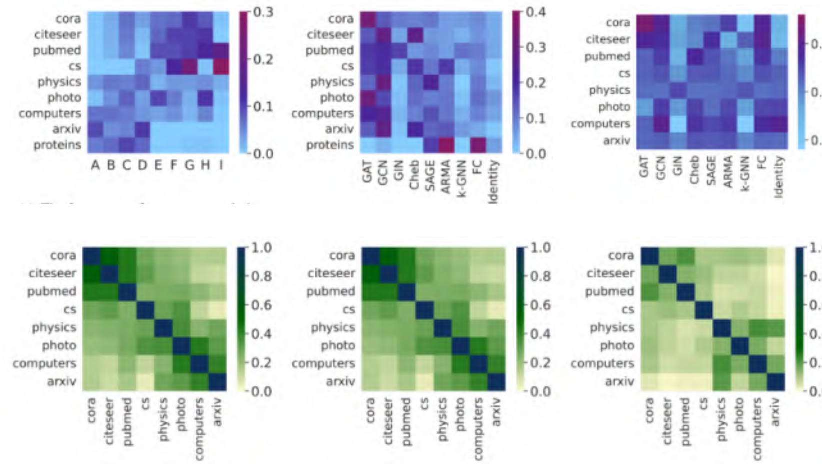
- Open source: https://github.com/THUMNLab/NAS-Bench-Graph

NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. ***NeurIPS, 2022.***

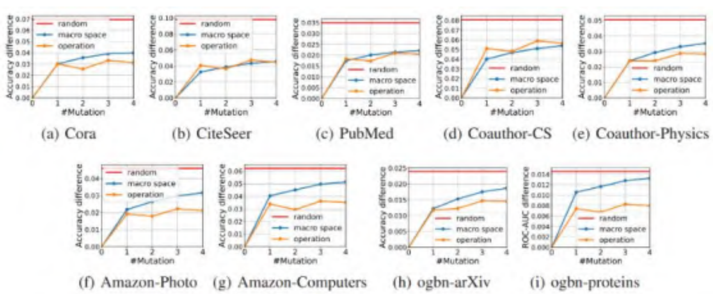
NAS-Bench-Graph: Analysis



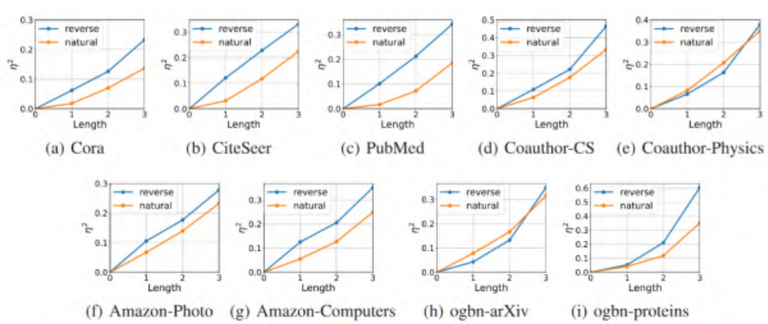
Performance distribution



Architecture distribution & Correlation



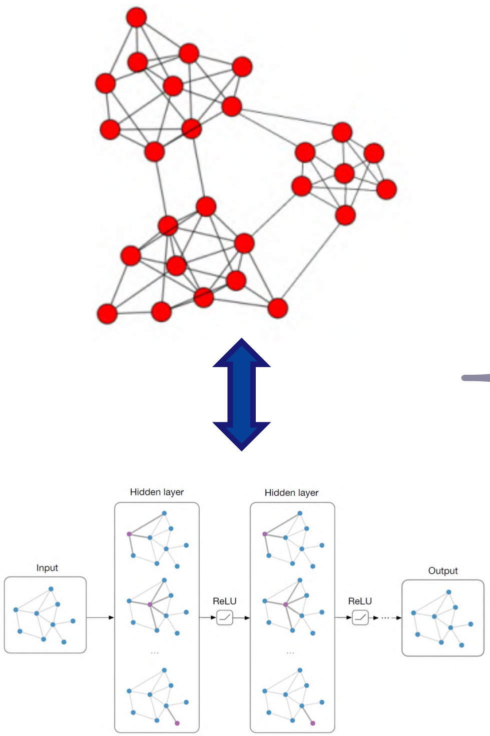
Architecture space smoothness



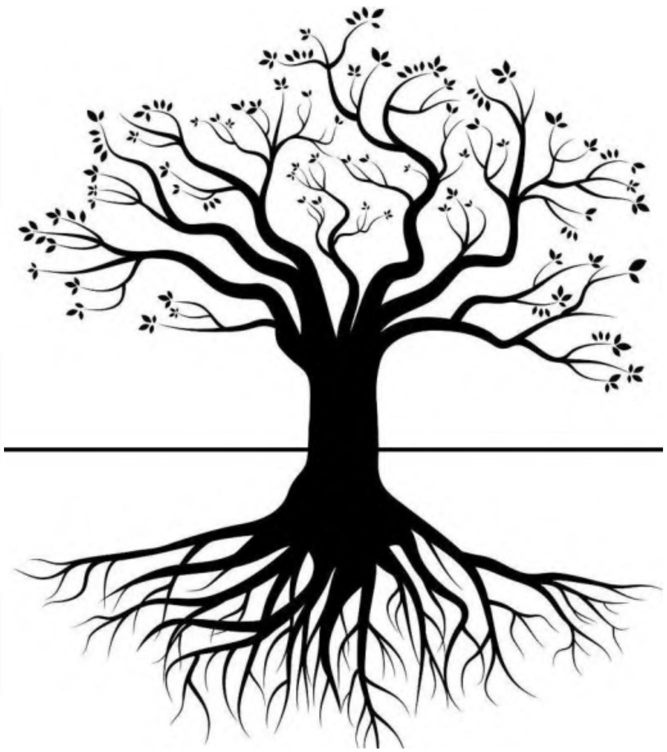
Influence of operations at different depth

NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. ***NeurIPS, 2022.***

Recap: Our Recent Works on GraphNAS



- Graph Structure
- Scalability
- Generalization



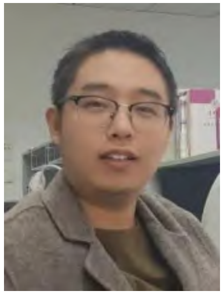
NAS-Bench-Graph

AutoGL: a library for automated graph machine learning

Acknowledgements

Wenwu Zhu
Tsinghua Univ.

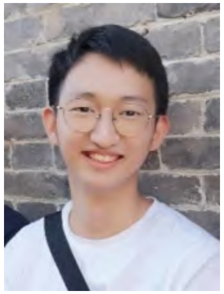
Chaoyu Guan
Tsinghua Univ.



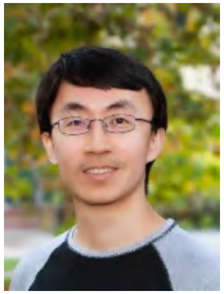
Yijian Qin
Tsinghua Univ.

Xin Wang
Tsinghua Univ.

Zeyang Zhang
Tsinghua Univ.



Pengtao Xie
UCSD



THANK YOU!

https://zw-zhang.github.io
zwzhang@tsinghua.edu.cn