清華大學 Tsinghua University

media and network lab

DataFun.

Automated Machine Learning on Graphs

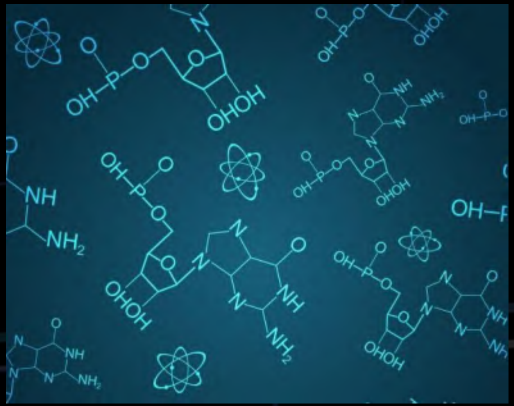
Ziwei Zhang
Tsinghua University

DataFunCon # 2023

Graphs are Ubiquitous

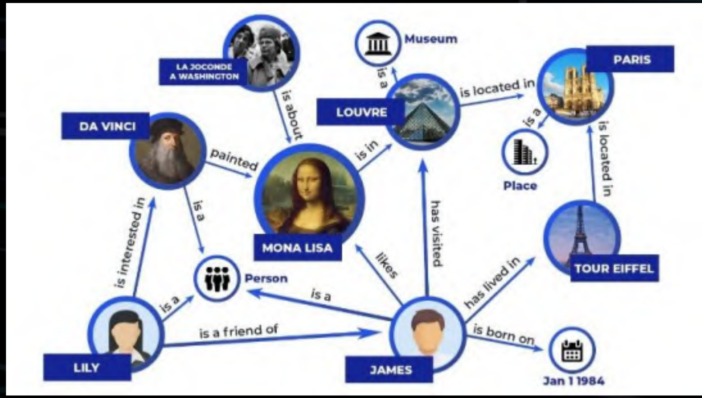
Social Network

Logistics Network



Biology Network

Traffic Network

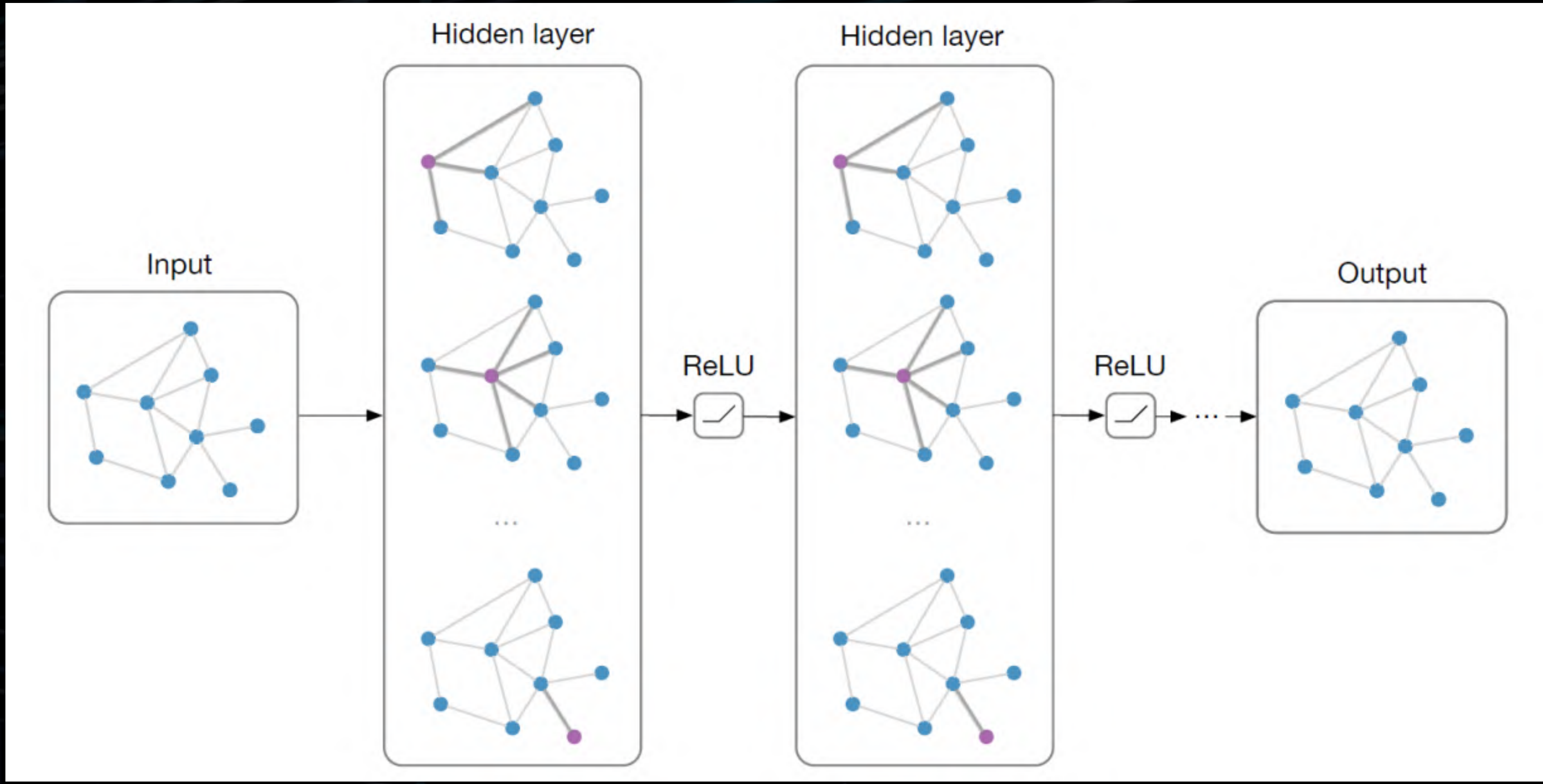


Knowledge Graphs

Information Network

Images are credit to web search engines

Graph Neural Network

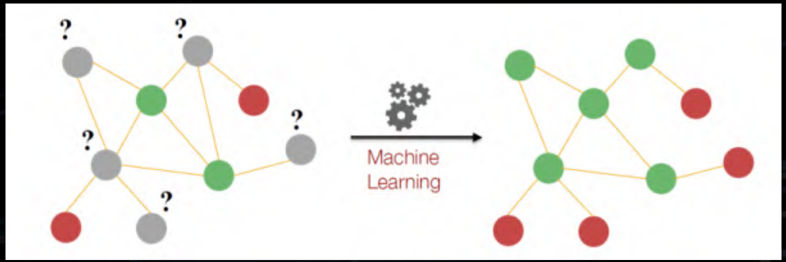


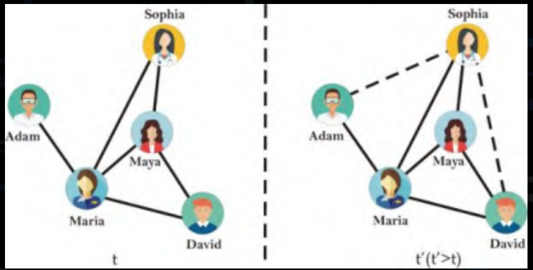
- Design neural networks directly applicable for graphs for end-to-end learning
- Message-passing framework: nodes exchange messages along structures

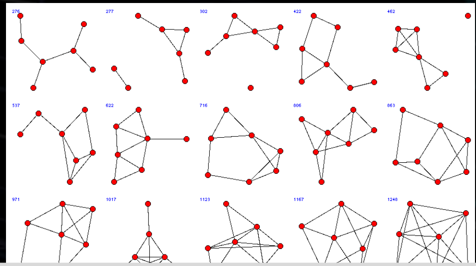
Image cited from Kipf and Welling, ICLR 2017

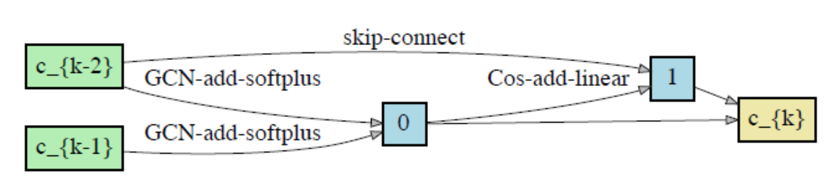
Problems in Traditional Graph Learning Methods

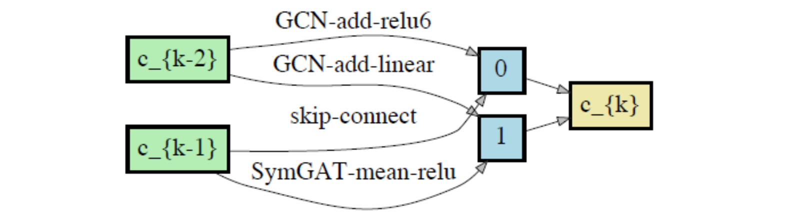
- Manually design architectures and hyper-parameters through trial-and-error
- Each dataset/task is handled separately

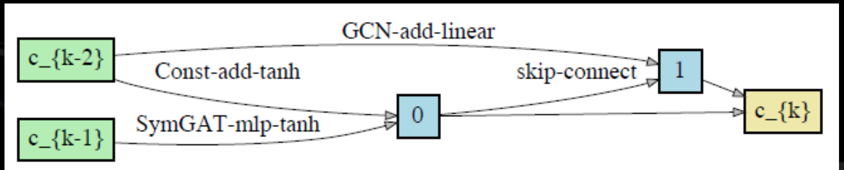






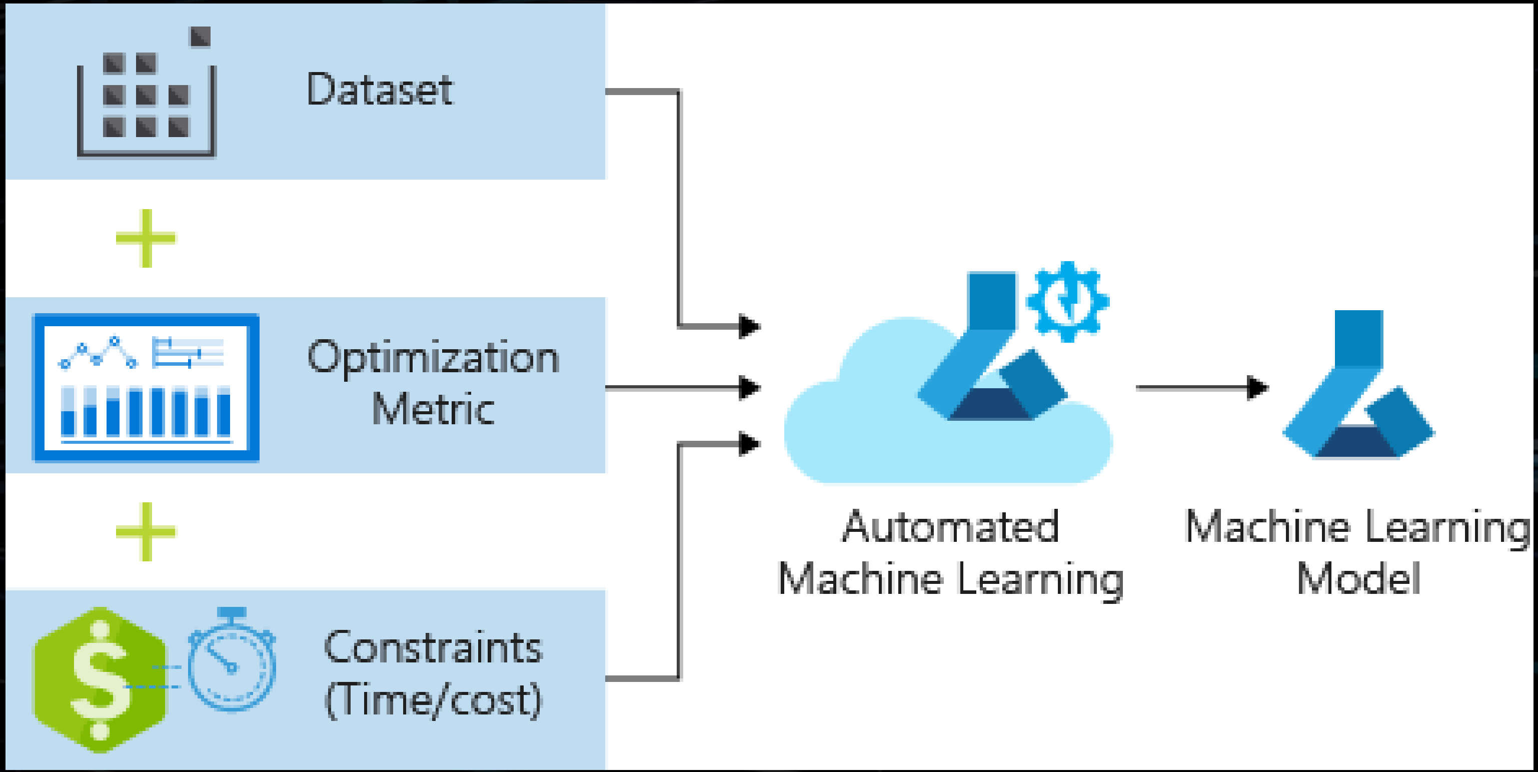






The adaptivity of graph machine learning is limited!

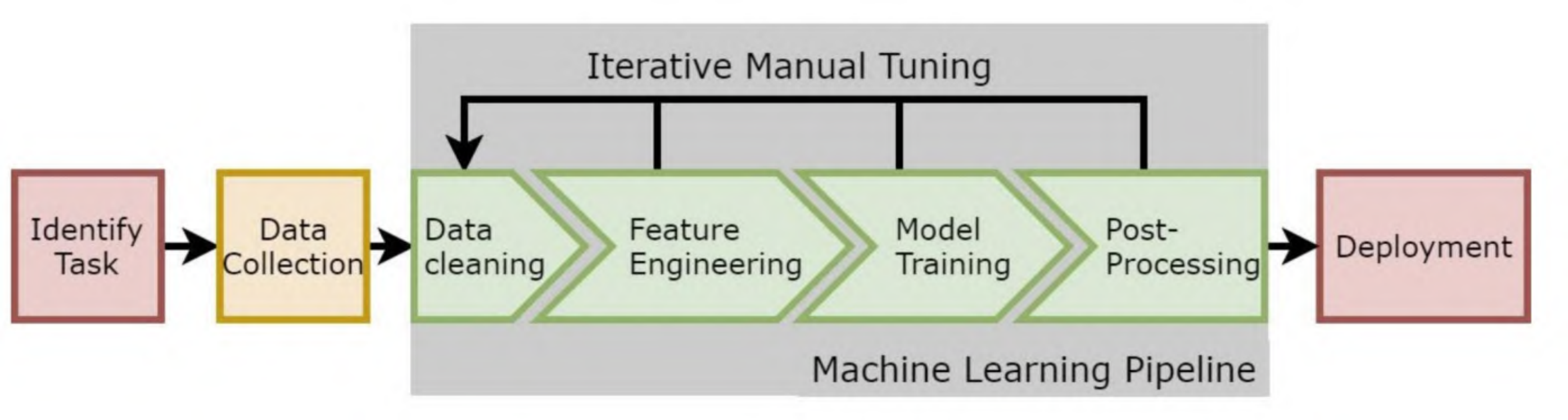
A Glance of AutoML



Design ML methods → Design AutoML methods

Picture credit to Microsoft Azure Machine Learning AutoML

ML vs. AutoML

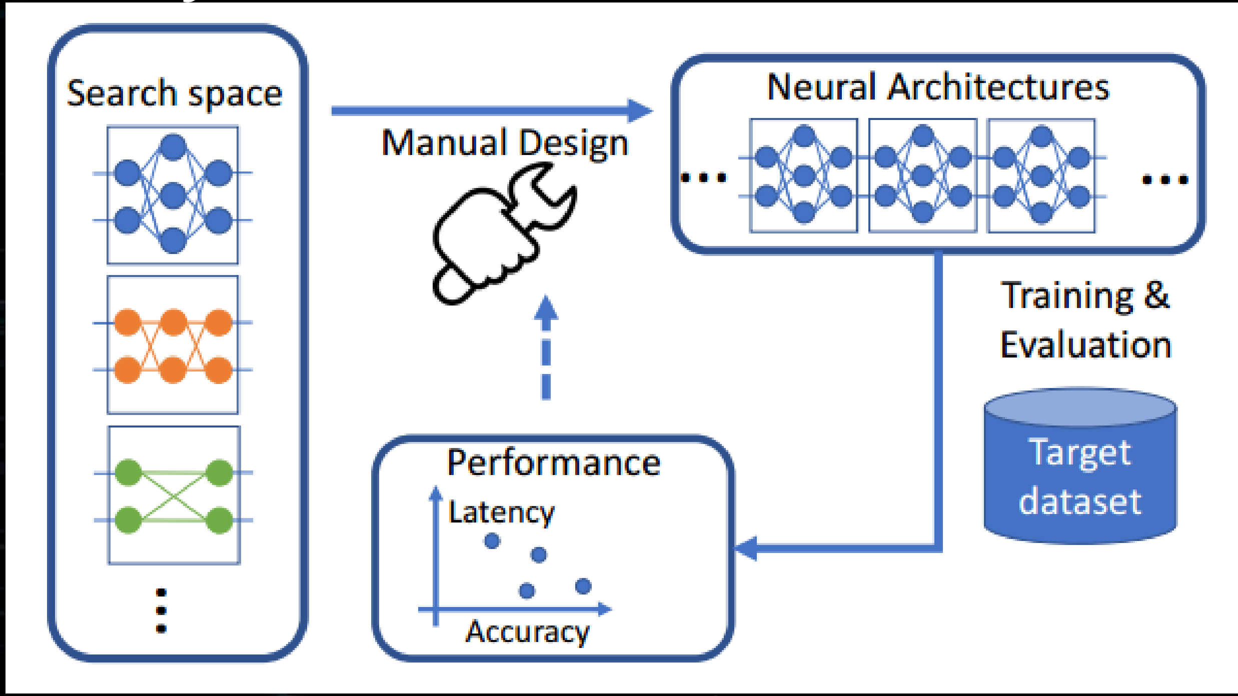


- Rely on **expert knowledge**
- **Tedious** trail-and-error
- **Limited** by human design

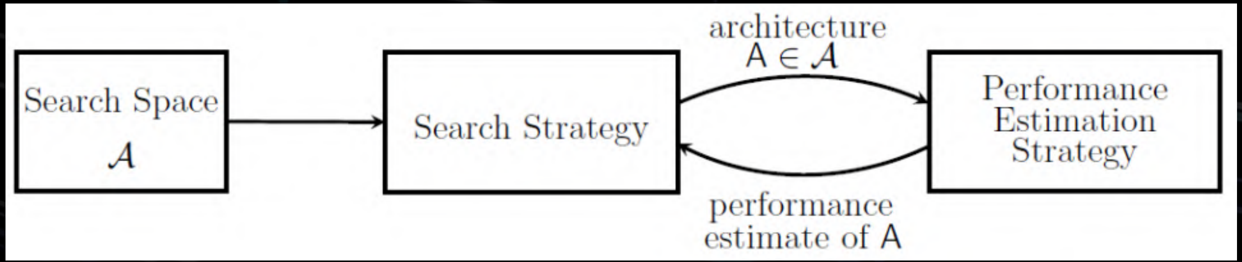
- **Free human** out of the loop
- **High** optimization **effectiveness**
- **Discover & extract** patterns and combinations **automatically**

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

DataFun.

Graph NAS Search Space

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

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

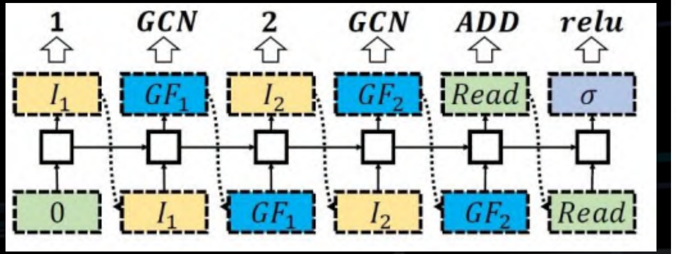
- $\mathrm{AGG}(\cdot)$: how to aggregate information from neighbors
  - Requirement: permutation-invariant
  - Common choices: mean, max, sum, etc.
- a_{ij}: the importance of neighbors
- $\mathrm{COMBINE}(\cdot)$: how to update representation
  - Common choices: CONCAT, SUM, MLP, etc.
- $\sigma(\cdot)$: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

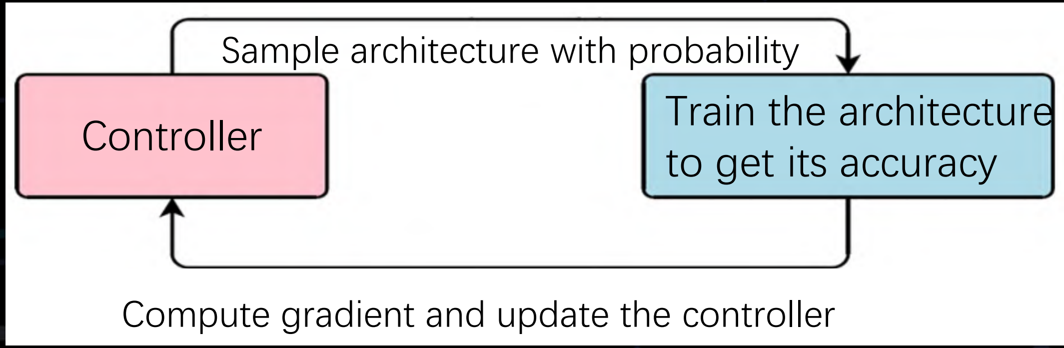
Type	Formulation
CONST	$a_{ij}^{\text{const}} = 1$
GCN	$a_{ij}^{\text{gcn}} = \frac{1}{\sqrt{\lvert\mathcal{N}(i)\rvert\lvert\mathcal{N}(j)\rvert}}$
GAT	$a_{ij}^{\text{gat}} = \mathrm{LeakyReLU}\left(\mathrm{ATT}\left(\mathbf{W}_a\left[\mathbf{h}_i, \mathbf{h}_j\right]\right)\right)$
SYM-GAT	$a_{ij}^{\text{sym}}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

Graph Neural Architecture Search, *IJCAI 2020.*

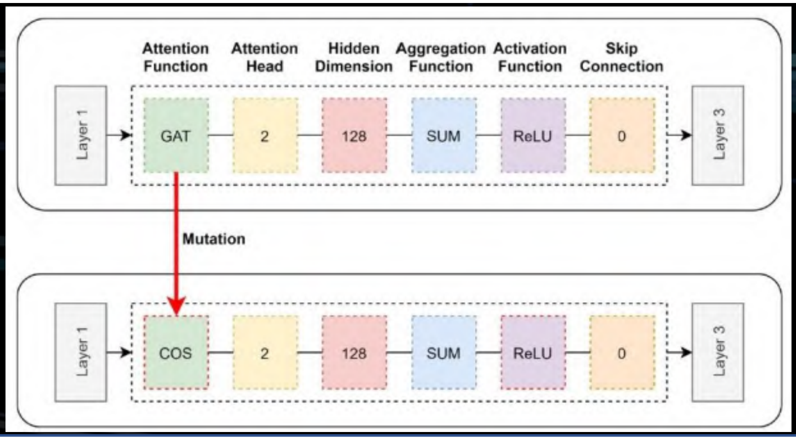
Graph NAS Search Strategy

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





  - Evolutionary
    - Define how to evolve and how to select

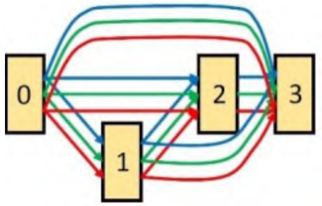


  - Differentiable
    - Super-net: mix all possible operations

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

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

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



DataFun.

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

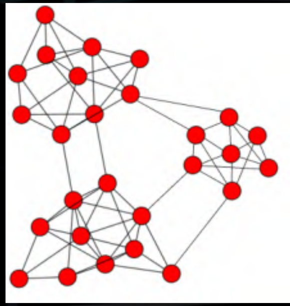
Table 1: A summary of different NAS methods for graph machine learnings.

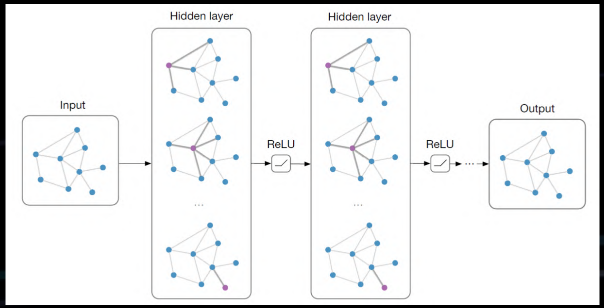
Paper collection: https://github.com/THUMNLab/awesome-auto-graph-learning
KDD 2021 Tutorial: https://zw-zhang.github.io/files/2021_KDD_AutoMLonGraph.pdf

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

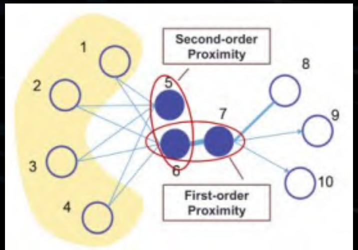
Challenges for the Existing Methods

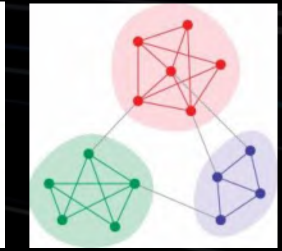
- GraphNAS has many unique and unsolved challenges

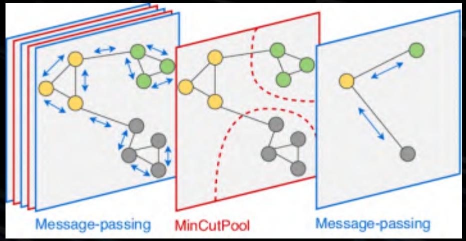




Graph Structure



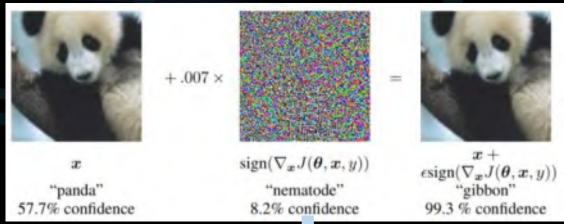


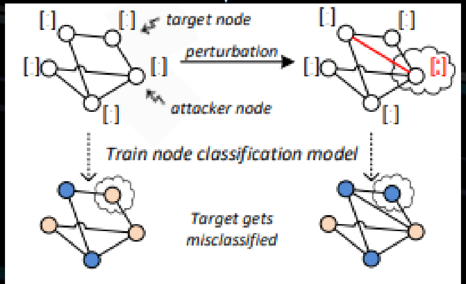


Scalability

billions of nodes/edges

Robustness



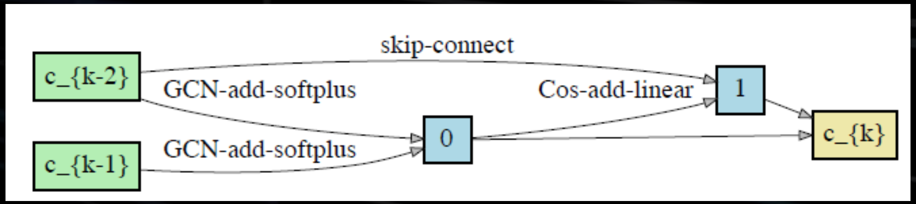


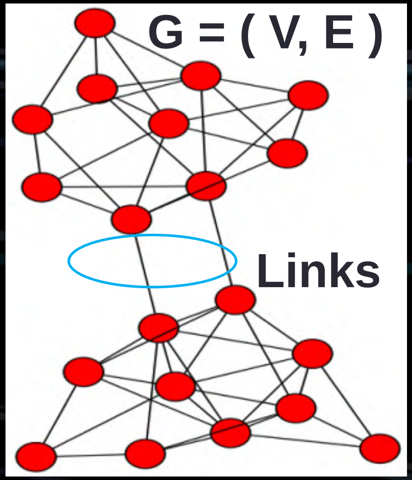
Challenge 1: Graph Structure

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

→ Q1: Is the input graph structure optimal?

→ Q2: How to select architectures and graph structures that suit each other?





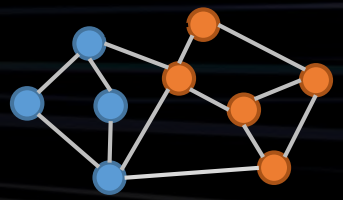
Challenge: how to model different graph structure in GraphNAS

Analysis

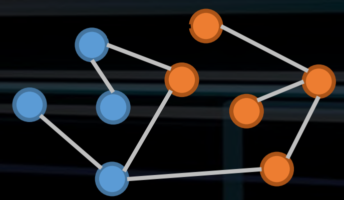
- Different operations fit graphs with different amount of information

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

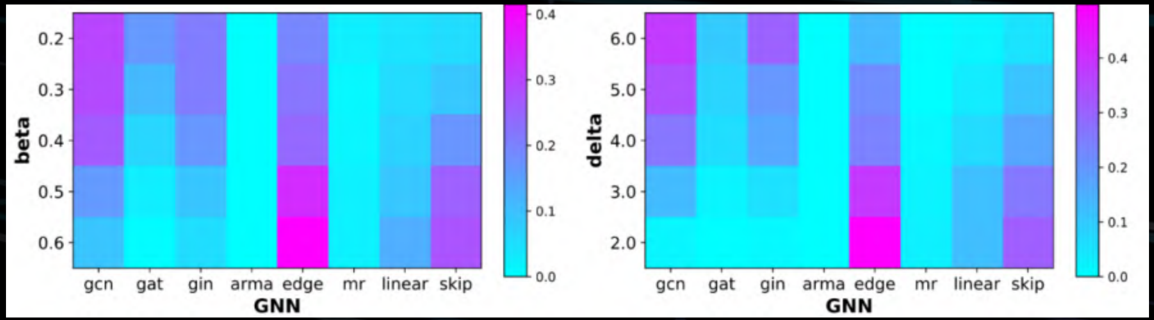


More structural information



Less structural information

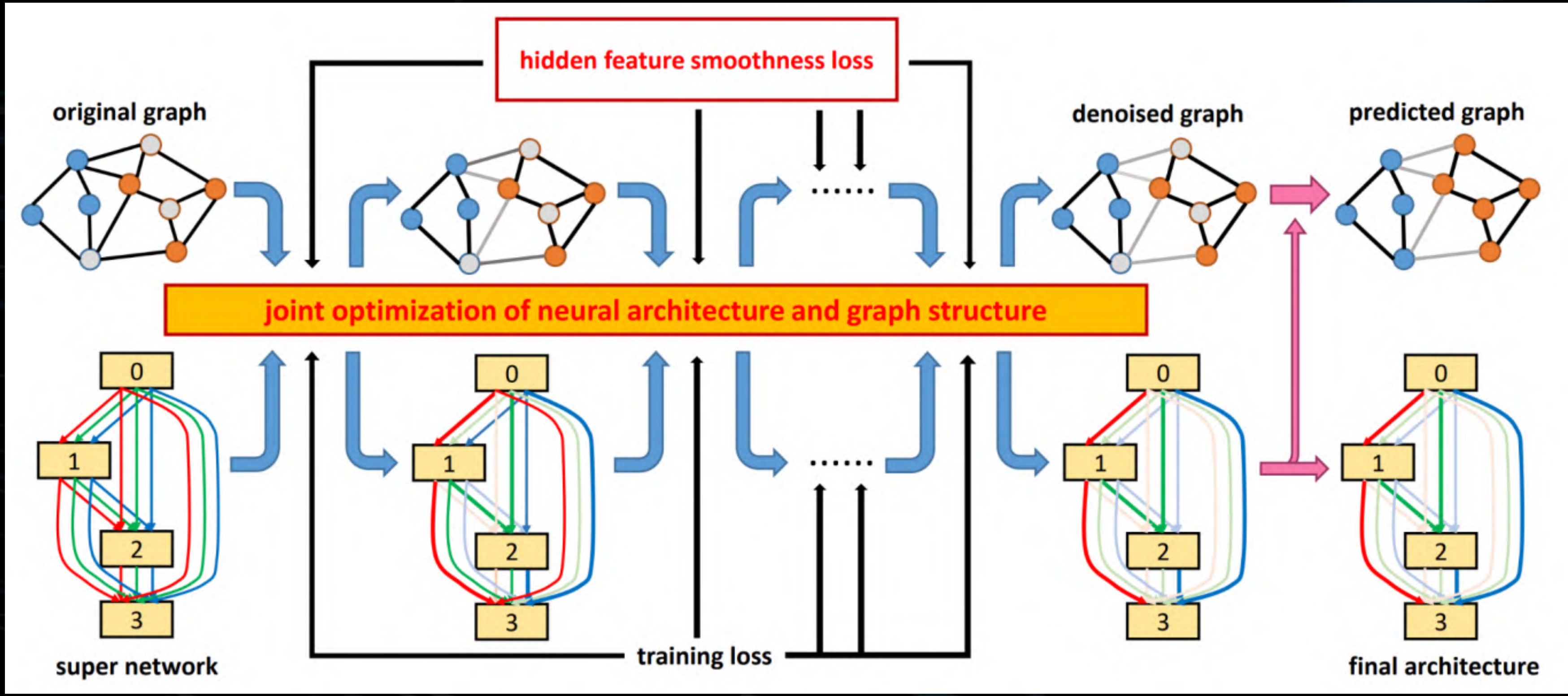
- Synthetic datasets:



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

GASSO: Graph Architecture Search with Structure Optimization

Learn graph structure and GNN architecture through a joint optimization scheme



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

DataFun.

GASSO: Model

- Formulation: bi-level optimization to tri-level optimization

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

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

  - Feature Smoothness Constraint

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

    - Homophily assumption/first-order neighborhood
    - Mask original edges: $G = G_o \odot M$
- Possible extensions: removing edge → adding edges
  - Challenge: time complexity, there are $O(n^2)$ possible edges

GASSO: Experiments

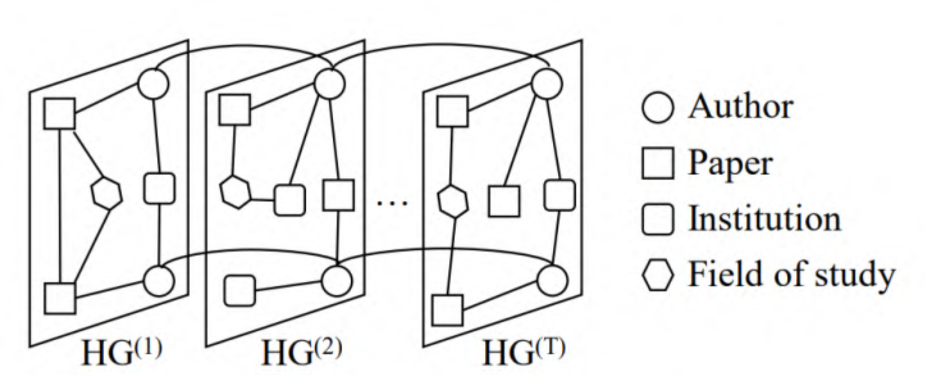
- Experiments on graph benchmarks

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	$\mathbf{87.63 \pm 0.29}$	$\mathbf{79.61 \pm 0.32}$	$\mathbf{90.52 \pm 0.24}$

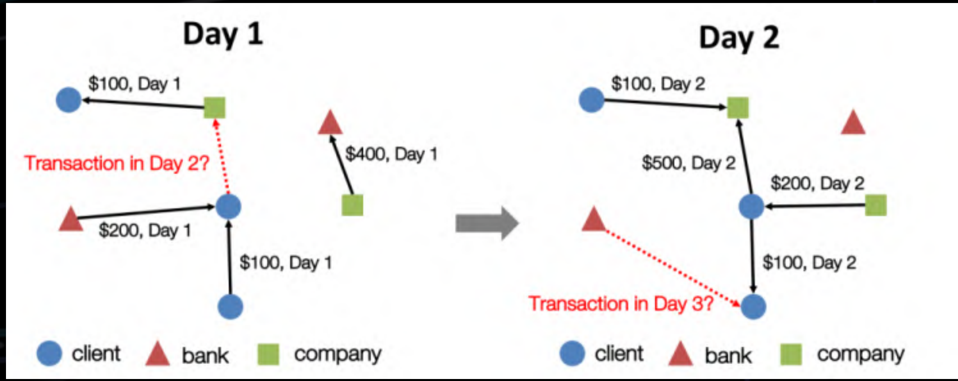
- Experiments on larger graph datasets

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

Dynamic Heterogenous Graphs



Citation network



Finance Graph

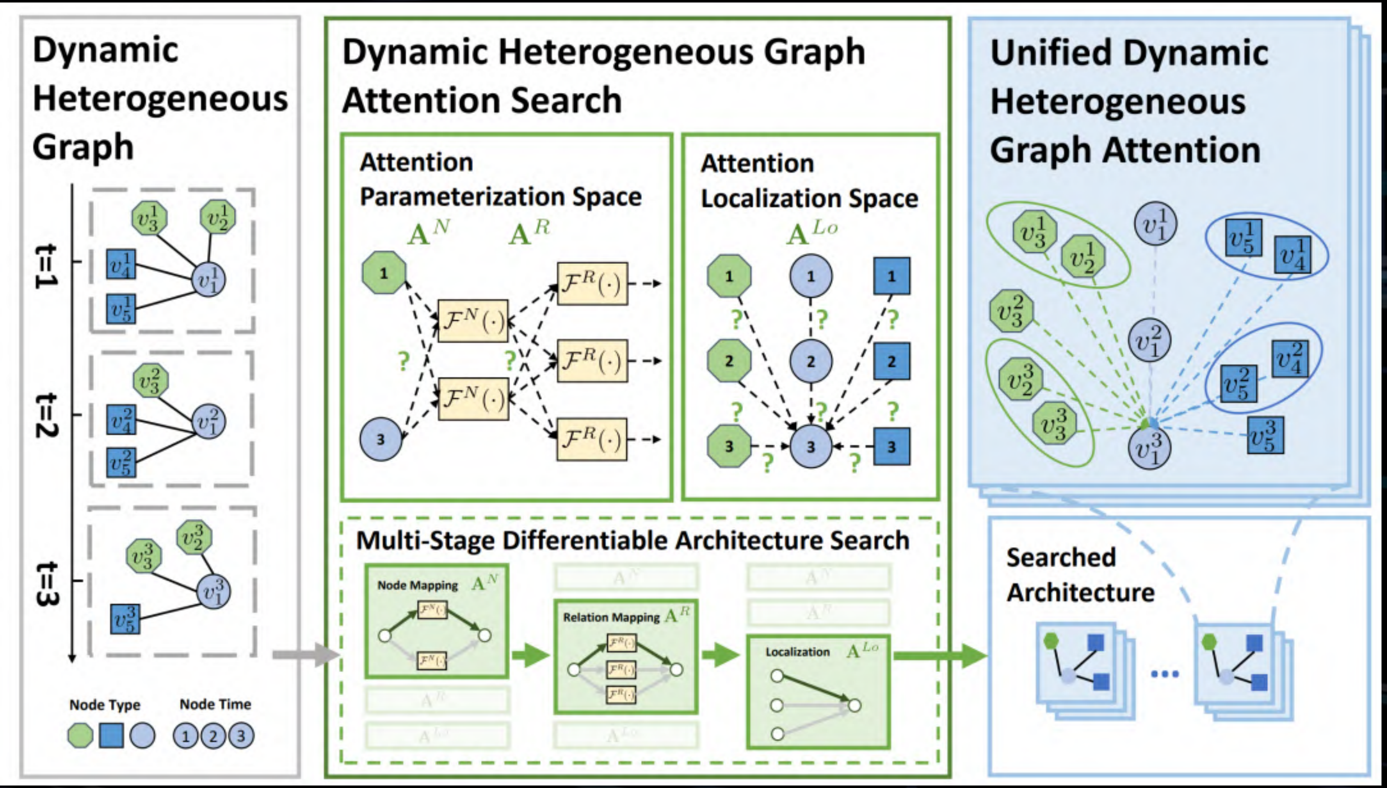
- **Dynamic**: structures and features evolve through time
- **Heterogeneous**: various node and edge types

- More complicated structural patterns

How to model the dynamic and heterogenous structure in GraphNAS?

Dynamic Heterogeneous Graph Architecture Search

Automatically tailor an optimal attention-based architecture for dynamic heterogeneous graphs



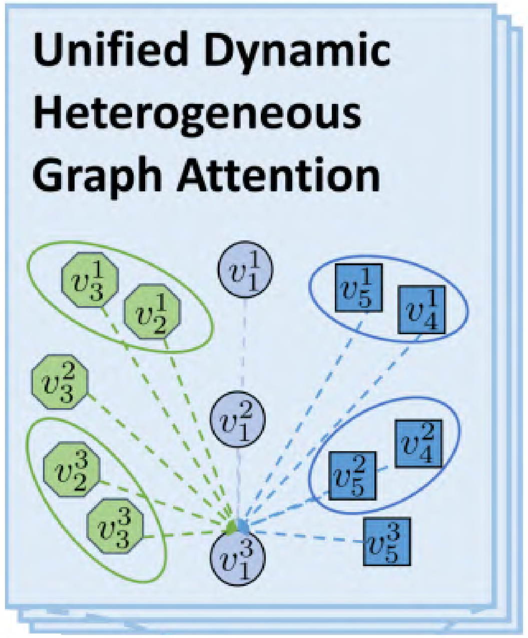
Dynamic Heterogeneous Graph Attention Neural Architecture Search. ***AAAI, 2023.***

DataFun.

DHGAS: Dynamic Heterogeneous Graph Attention

- **Goal**: capture dynamic heterogenous information through attention

Definition 2 ***Dynamic Heterogeneous Neighborhood****: for the neighborhood of each node u, we use subscripts to denote the relation type and superscripts to denote the time stamp, i.e., $\mathcal{N}_r^t(u) = \{v : (u,v) \in \mathcal{E}^t, \phi_e(u,v) = r\}$. With a slight abuse of notations, we use $\mathcal{N}(u)$ to denote all types of neighbors at all time stamps in dynamic heterogeneous graphs, i.e., $\mathcal{N}(u) = \bigcup_{r,t} \mathcal{N}_r^t(u)$.*



- Time/type-aware node mapping functions

$$\begin{cases} \mathbf{q}_u^t = \mathcal{F}^N_{q,\phi_n(u),t}(\mathbf{h}_u^t), \\ \mathbf{k}_v^{t'} = \mathcal{F}^N_{k,\phi_n(v),t'}(\mathbf{h}_v^{t'}), \\ \mathbf{v}_v^{t'} = \mathcal{F}^N_{v,\phi_n(v),t'}(\mathbf{h}_v^{t'}), \end{cases}$$

- Time/type-aware relation mapping functions

$$\begin{cases} \alpha_{u,v} = \mathcal{F}^R_{\phi_e(u,v),\Delta t}(\mathbf{q}_u^t, \mathbf{k}_v^{t'}), \\ \mathcal{F}^R_{\phi_e(u,v),\Delta t}(\mathbf{q}, \mathbf{k}) = \dfrac{\mathbf{q}\mathbf{W}_{\phi_e(u,v),\Delta t}\mathbf{k}^\top}{\sqrt{d}}, \end{cases}$$

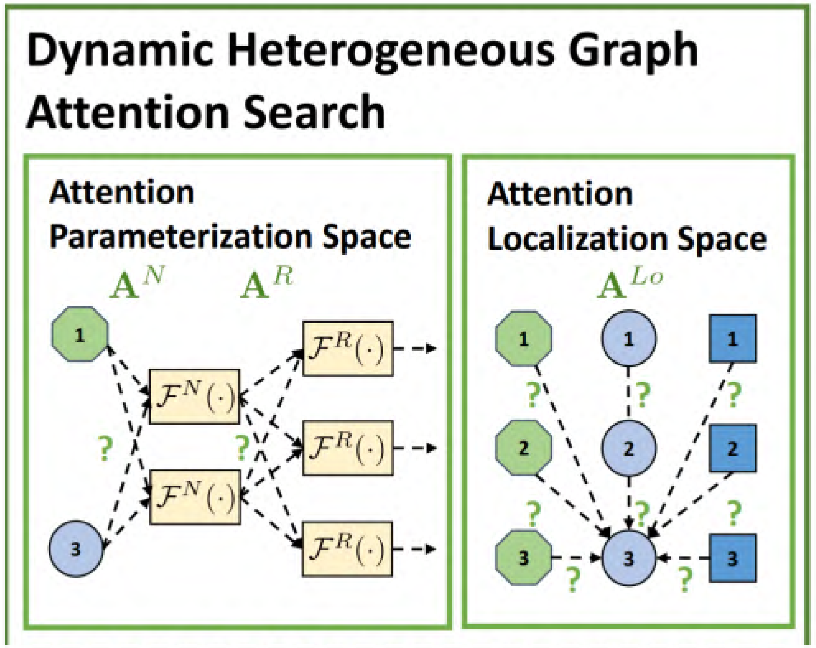
- Update with time/type-aware attention

$$\mathbf{h}_u^t \leftarrow \text{Update}(\mathbf{h}_u^t, \sum\nolimits_{v \in \mathcal{N}(u)} \hat{\alpha}_{u,v} \mathbf{v}_v^{t'})),$$

$$\hat{\alpha}_{u,v} = \frac{\exp(\alpha_{u,v})}{\sum_{v' \in \mathcal{N}(u)} \exp(\alpha_{u,v'})}.$$

DHGAS: Attention Parameterization and Localization Space

- **Goal**: a concise yet expressive search space based on attention



- Parameterization Space: how to parameterize attention

$$\mathcal{A}^{Pa} = \mathcal{A}^N \times \mathcal{A}^R \qquad \mathcal{A}^N = \{1, ..., K_N\}^{T \times |\mathcal{C}_n|} \qquad \mathcal{A}^R = \{1, ..., K_R\}^{2T \times |\mathcal{C}_e|}$$

- Localization Space: Locate where to apply attention $\mathcal{A}^{Lo} = \{0, 1\}^{T \times T \times |\mathcal{C}_e|}$
- Cover many classic GNNs as special cases: GCN, GAT, type-aware MLP, HGT(WWW20), DyHATR(ECML20), HTGNN(SDM22), etc.

DHGAS: Multi-Stage Differentiable Architecture Search

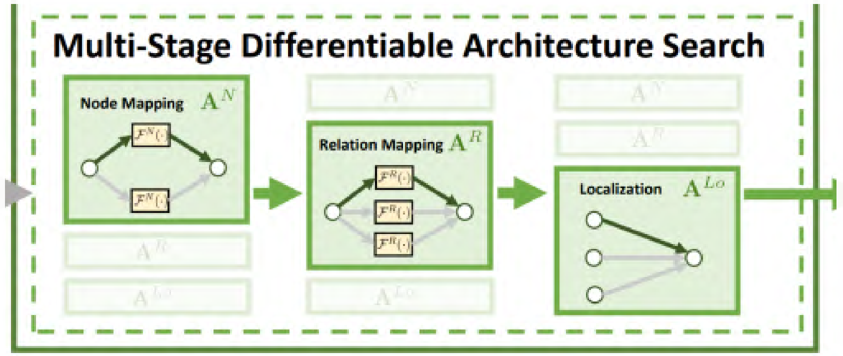
- **Goal**: efficient and differentiable search strategy
  - Relax discrete operation choices to continuous ones

$$\bar{\mathcal{F}}(\mathbf{x}) = \sum_{i=1}^{|A|} \frac{\exp(\beta_i)}{\sum_{j=1}^{|A|} \exp(\beta_j)} \mathcal{F}_i(\mathbf{x})$$

  - Update super-networks weights and architecture differentiably

$$\mathbf{w} \leftarrow \mathbf{w} - \eta_{\mathbf{w}} \frac{\partial \mathcal{L}_{\text{train}}}{\partial \mathbf{w}}, \beta \leftarrow \beta - \eta_{\beta} \frac{\partial \mathcal{L}_{\text{val}}}{\partial \beta}$$

  - Multi-stage training to stabilize the searching process



DataFun.

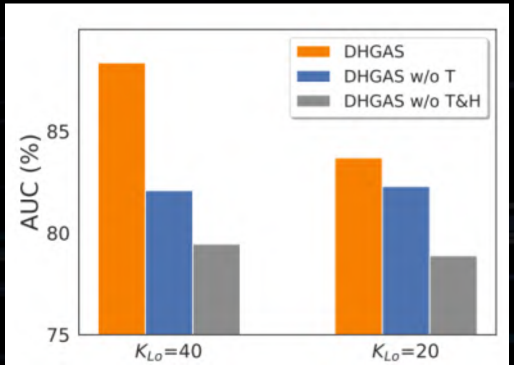
DHGAS: Experiments

Task Metric	Link Prediction (AUC%) ↑		Node Classification (F1%) ↑		Node Regression (MAE) ↓
Dataset	**Aminer**	**Ecomm**	**Yelp**	**Drugs**	**COVID-19**
GCN	73.84 ± 0.06	77.94 ± 0.22	37.02 ± 0.00	56.43 ± 0.21	846 ± 101
GAT	80.84 ± 0.96	78.49 ± 0.31	35.54 ± 0.00	57.06 ± 0.00	821 ± 91
RGCN	82.75 ± 0.12	$\underline{82.27 \pm 0.51}$	37.75 ± 0.00	57.97 ± 0.14	833 ± 95
HGT	78.43 ± 1.81	81.09 ± 0.52	34.62 ± 0.00	57.65 ± 0.01	805 ± 88
DyHATR	74.24 ± 2.09	71.69 ± 0.90	34.49 ± 0.16	55.51 ± 0.09	643 ± 36
HGT+	$\underline{85.60 \pm 0.12}$	76.68 ± 0.85	38.33 ± 0.00	$\underline{59.09 \pm 0.00}$	-
HTGNN	78.08 ± 0.80	76.78 ± 6.37	36.33 ± 0.07	56.24 ± 0.34	$\underline{555 \pm 34}$
GraphNAS	81.61 ± 0.98	79.37 ± 0.21	37.73 ± 0.00	57.13 ± 0.52	820 ± 43
DiffMG	85.04 ± 0.30	81.69 ± 0.06	$\underline{38.65 \pm 0.00}$	58.45 ± 0.15	629 ± 63
DHGAS	$\mathbf{88.13 \pm 0.18}$	$\mathbf{86.56 \pm 0.58}$	$\mathbf{41.99 \pm 0.18}$	$\mathbf{62.35 \pm 0.03}$	$\mathbf{536 \pm 43}$

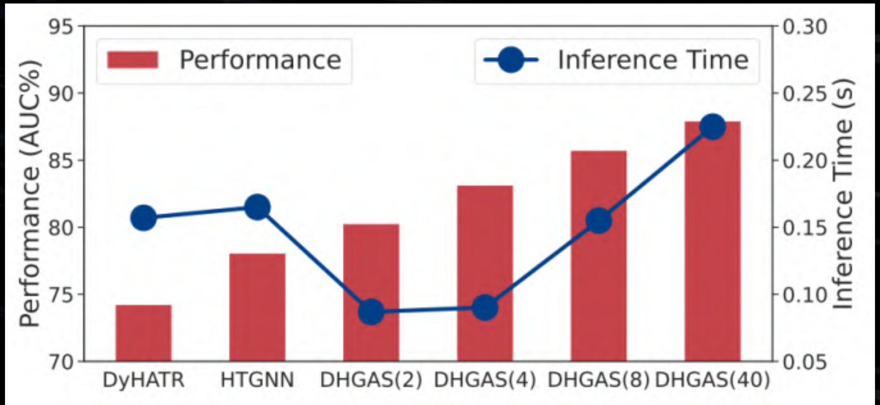
Significantly outperforms baselines for various downstream tasks

DHGAS: Experiments

- Jointly modeling dynamic and heterogeneous information

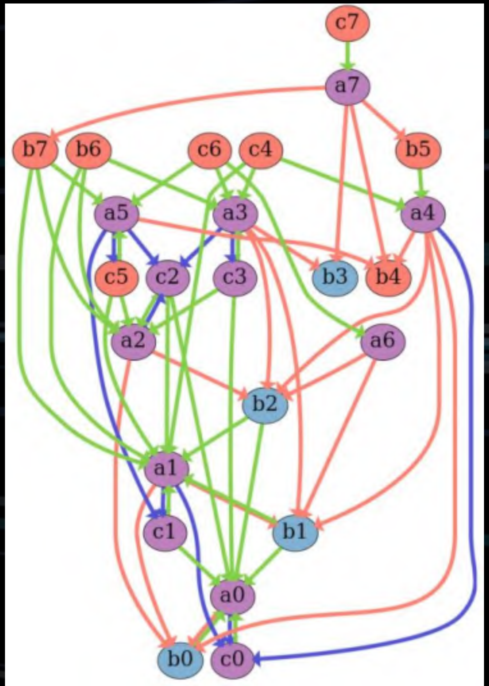


Ablation studies

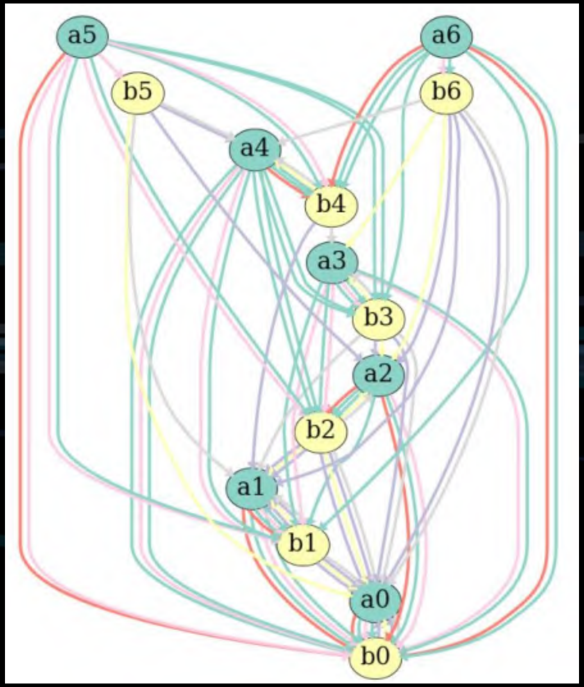


Performance and costs tradeoff

- Tailor optimal attention mechanisms for different datasets



Aminer



Ecomm

Architecture showcase

Challenge 2: 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)

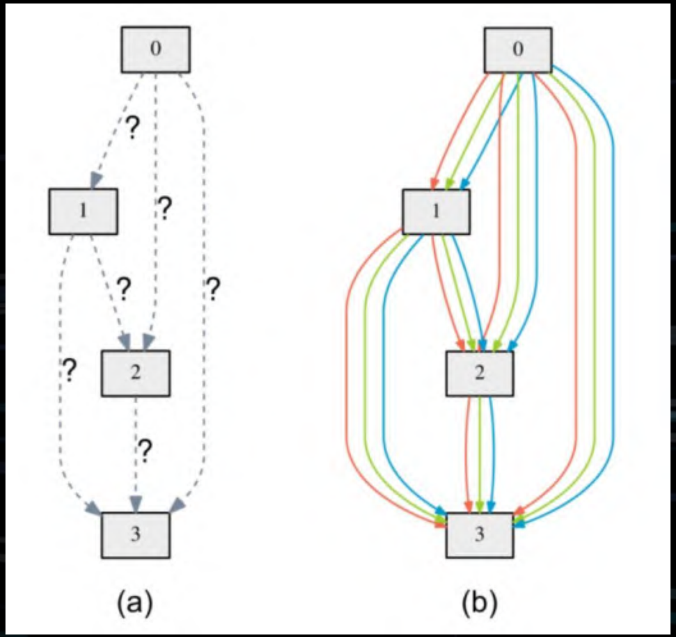
Citation Networks

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

Challenge: how to efficiently scale to billion-scale graphs

SuperNet Training

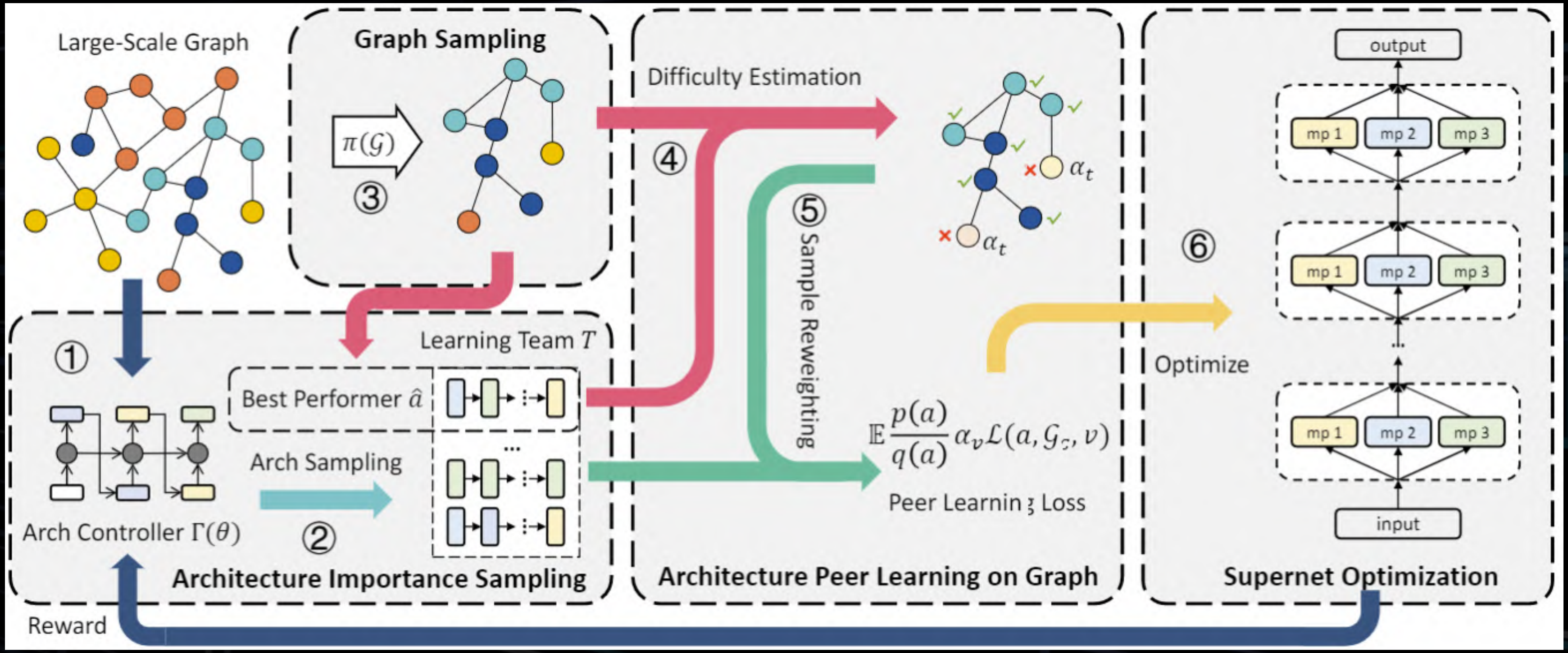
- Supernet: combine all possible operations of the search space



  - 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 NAS

Jointly sample subgraphs and architectures to find the most suitable architecture



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

GAUSS: Architecture Importance Sampling

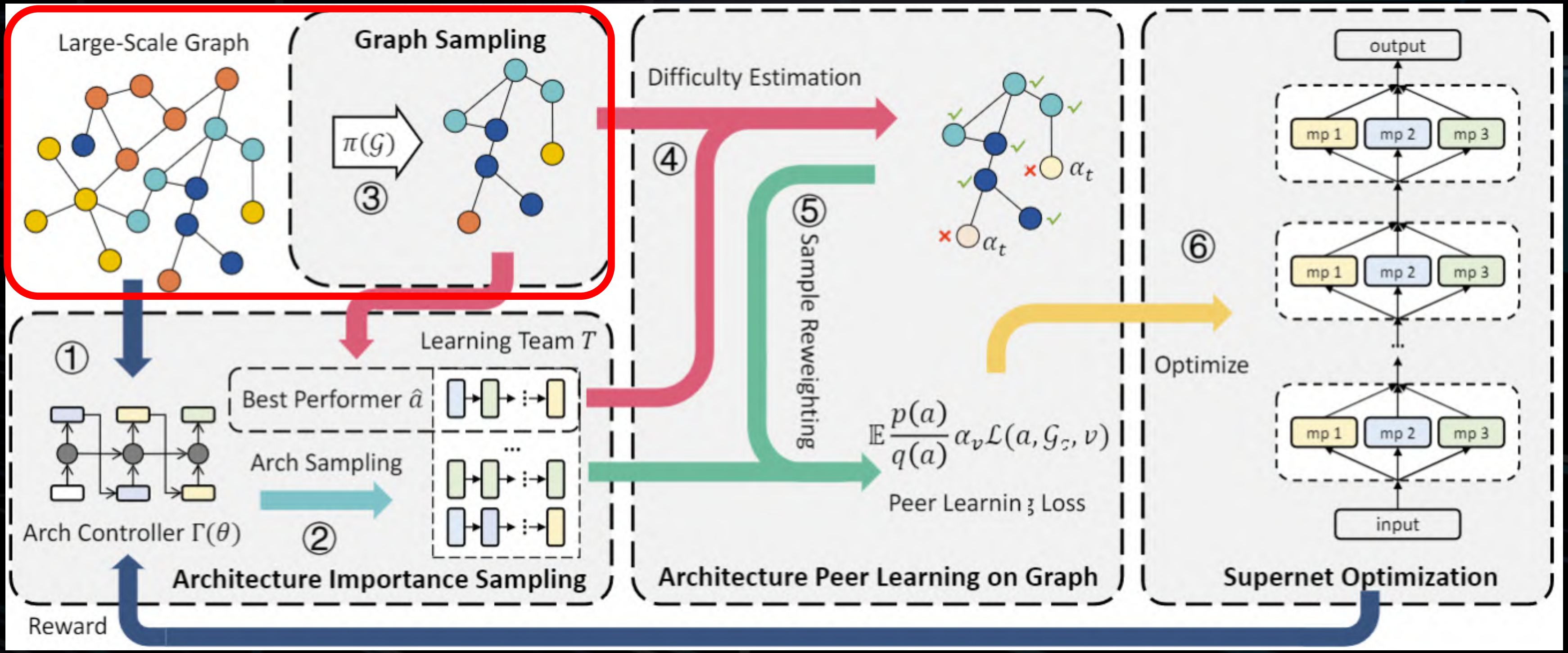
- **Goal**: stabilize the training of the supernet
- **Method**: important sampling of architectures
  - $\Gamma(\mathcal{A})$: proposal distribution
  - Learning proposal distribution: reinforcement learning with GRU controller

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

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

Reward function: performance + regularizer

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



GAUSS: Architecture Peer Learning on Graph

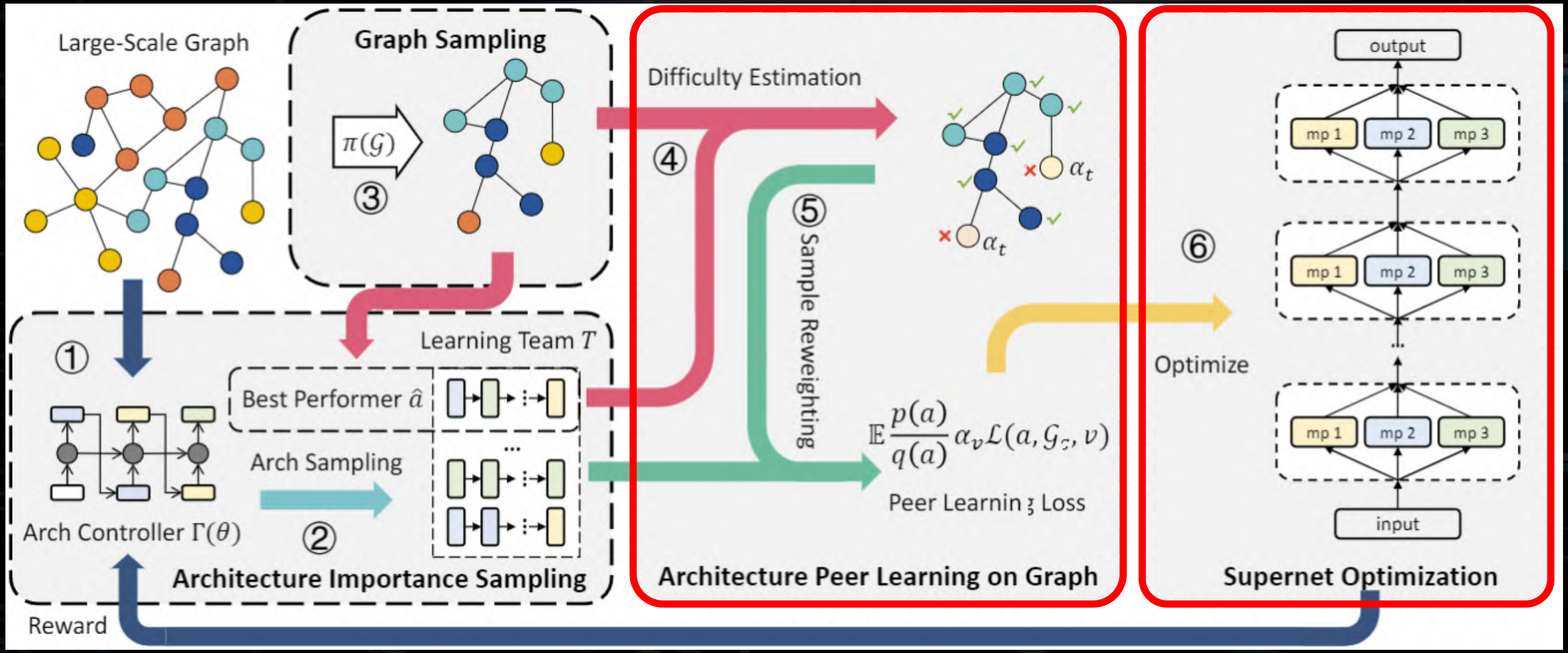
- **Goal**: smooth the optimization objective
- **Assumption**: "senior students" can teach "junior students"
- **Method**: assign weights to different samples, gradually progress from easier parts to difficult parts

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

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

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

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



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

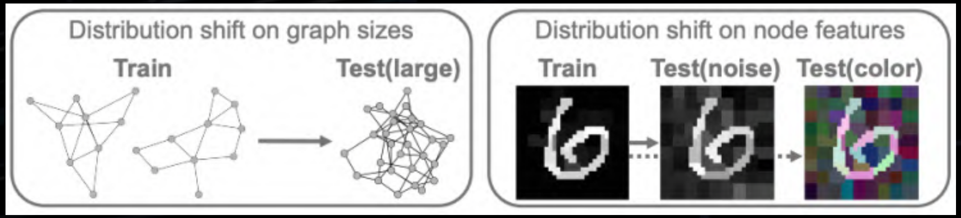
Table 2. The results of our proposed method and baseline methods. We report both the validation and test accuracy [%] over 10 runs with different seeds. OOT means out-of-time (cannot converge within 1 single GPU day), while OOM means out-of-memory (cannot run on a Tesla V100 GPU with 32GB memory). The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

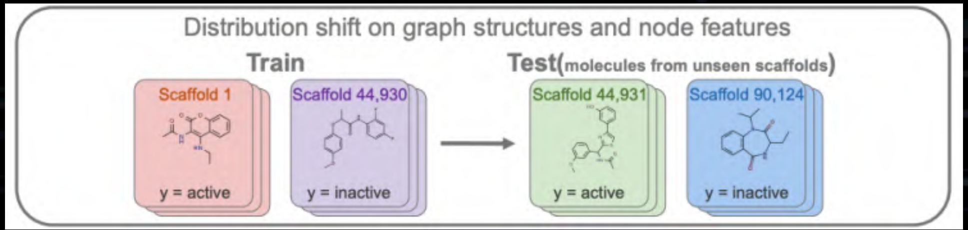
Methods	CS		Physics		Arxiv		Products		Papers100M	
	valid	test	valid	test	valid	test	valid	test	valid	test
GCN	$94.10_{0.21}$	$93.98_{0.21}$	$96.29_{0.05}$	$96.38_{0.07}$	$72.76_{0.15}$	$71.70_{0.18}$	$91.75_{0.04}$	$80.19_{0.46}$	$70.32_{0.11}$	$67.06_{0.17}$
GAT	$93.74_{0.27}$	$93.48_{0.36}$	$96.25_{0.23}$	$96.37_{0.23}$	$73.19_{0.12}$	$71.85_{0.21}$	$90.75_{0.16}$	$80.59_{0.40}$	$70.26_{0.16}$	$67.26_{0.06}$
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	$\mathbf{96.08}_{0.11}$	$\mathbf{96.49}_{0.11}$	$\mathbf{96.79}_{0.06}$	$\mathbf{96.76}_{0.08}$	$\mathbf{73.63}_{0.10}$	$\mathbf{72.35}_{0.21}$	$91.60_{0.12}$	$\mathbf{81.26}_{0.36}$	$\mathbf{70.57}_{0.07}$	$\mathbf{67.32}_{0.18}$

Can process billion-scale graphs using single GPU

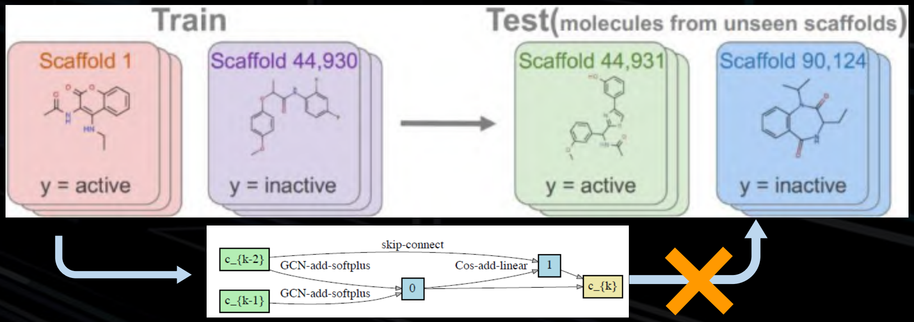
Challenge 3: Robustness

- Distribution shifts

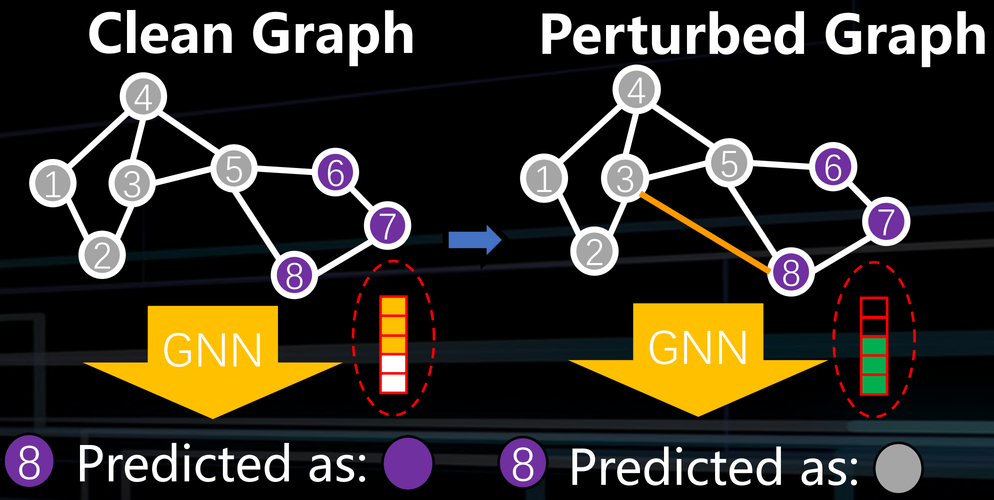




- Searching a fixed architecture may fail to generalize



- Adversarial attacks



- Greatly affects risk-sensitive applications

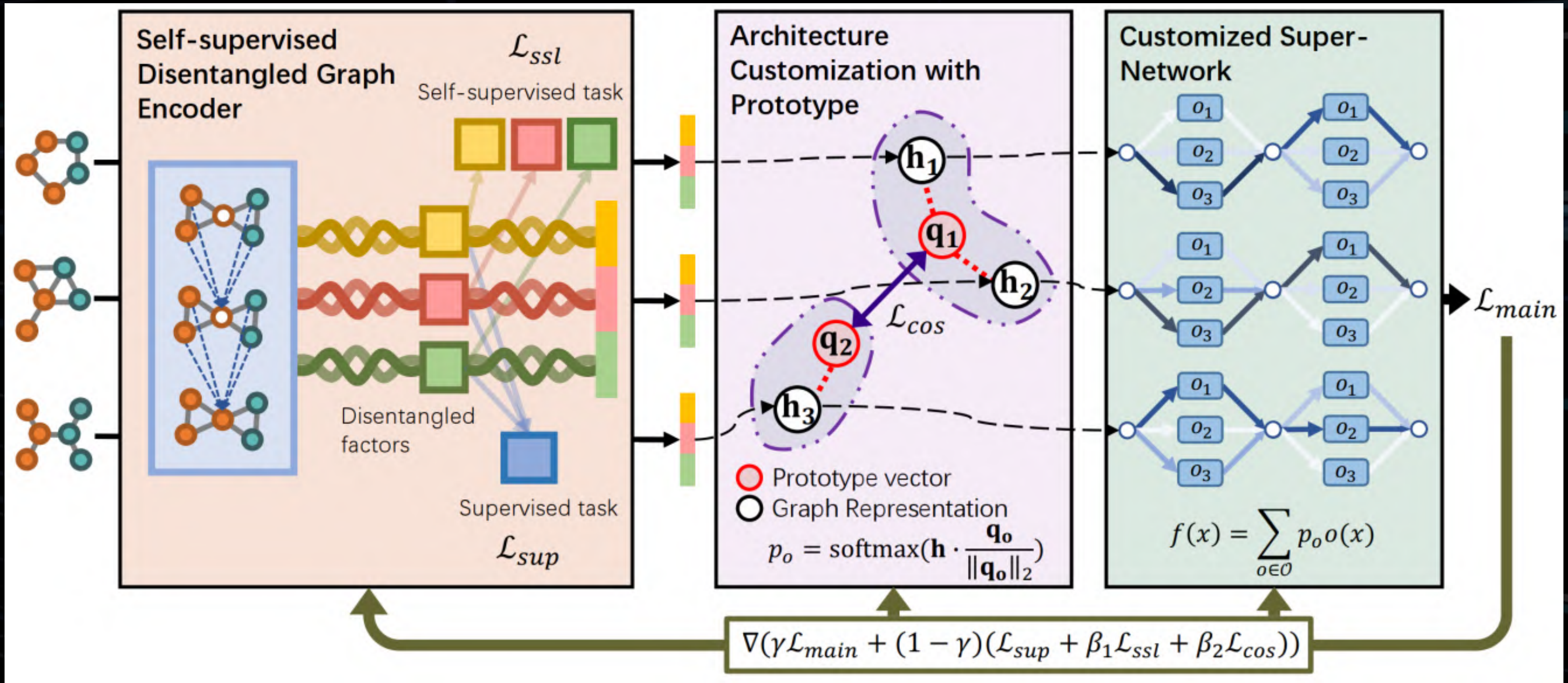
Fraud Detection

Cyber security

…

GRACES: Graph NAS under Distribution Shifts

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



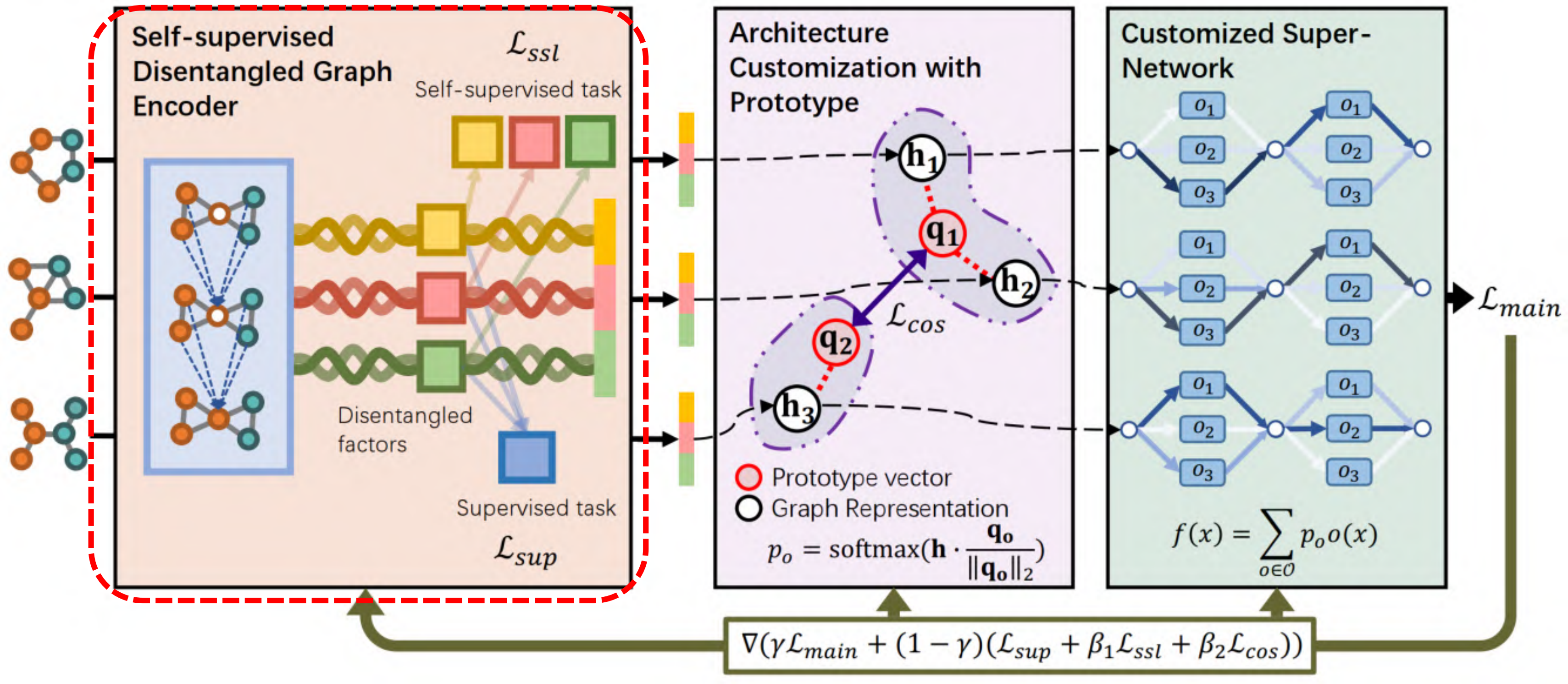
Graph Neural Architecture Search under Distribution Shifts. ***ICML, 2022.***

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
  - Supervised loss: the downstream task
  - Self-supervised loss: node degree as regularization

$$\mathbf{H}^{(l)} = \Big\Vert_{k=1}^{K} \mathrm{GNN}(\mathbf{H}_k^{(l-1)}, \mathbf{A})$$

$$\mathcal{L}_{sup} = \sum_{i=1}^{N_{tr}} \ell\left(\mathcal{C}\left(\mathbf{h}_i\right), y_i\right) \quad \mathcal{L}_{ssl} = \sum_{i=1}^{N_{tr}} \sum_{k=1}^{K-1} \ell_{ssl}\left(\hat{y}_{i,k}^{ssl}, y_{i,k}^{ssl}\right)$$



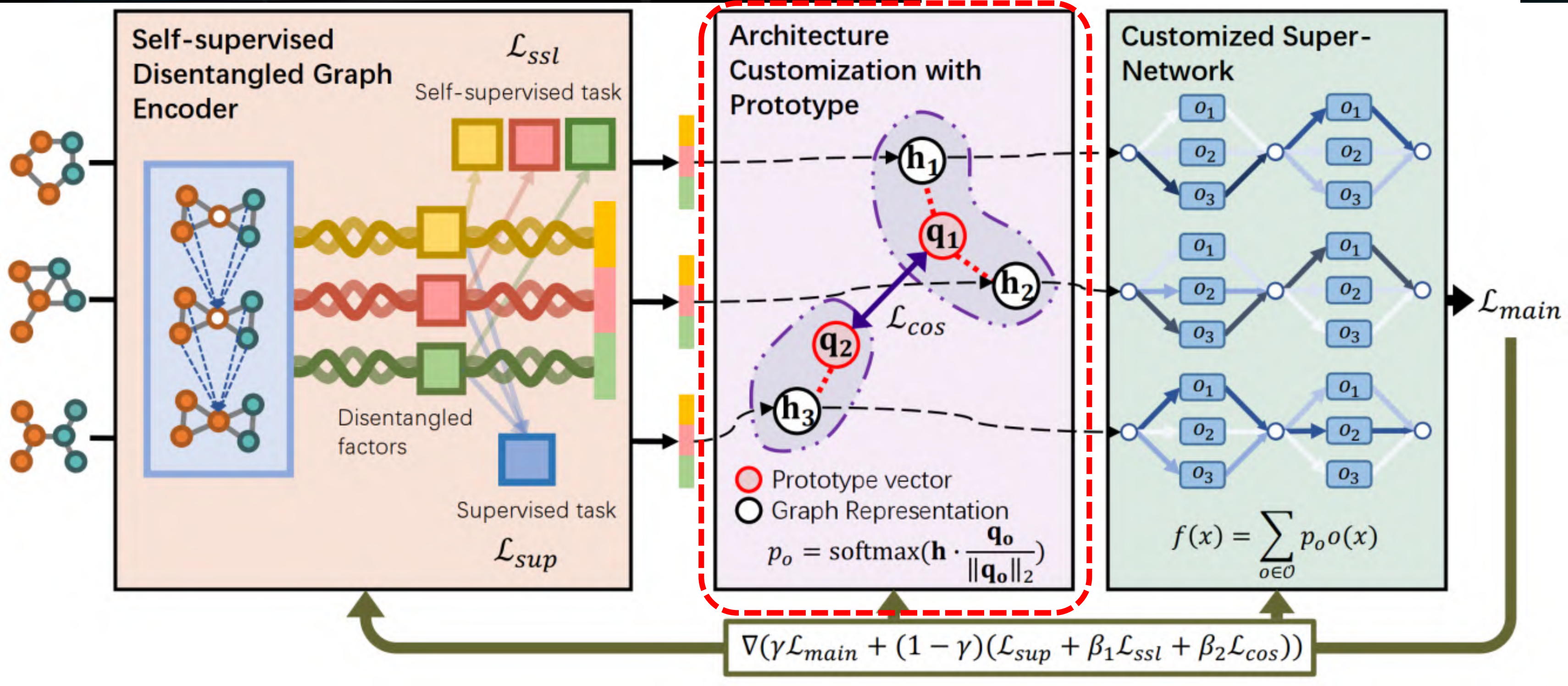
GRACES: Architecture Customization

- **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:

$$\hat{p}_o^i = \mathbf{h}\cdot\frac{\mathbf{q}_o^i}{\|\mathbf{q}_o^i\|_2}, p_o^i = \frac{\exp\left(\hat{p}_o^i\right)}{\sum_{o'\in\mathcal{O}}\exp\left(\hat{p}_{o'}^i\right)},$$

  - Regularizer to avoid mode collapse:

$$\mathcal{L}_{cos} = \sum_i \sum_{o,o'\in\mathcal{O},o\neq o'} \frac{\mathbf{q}_o^i\cdot\mathbf{q}_{o'}^i}{\|\mathbf{q}_o^i\|_2 \|\cdot\mathbf{q}_{o'}^i\|_2}$$



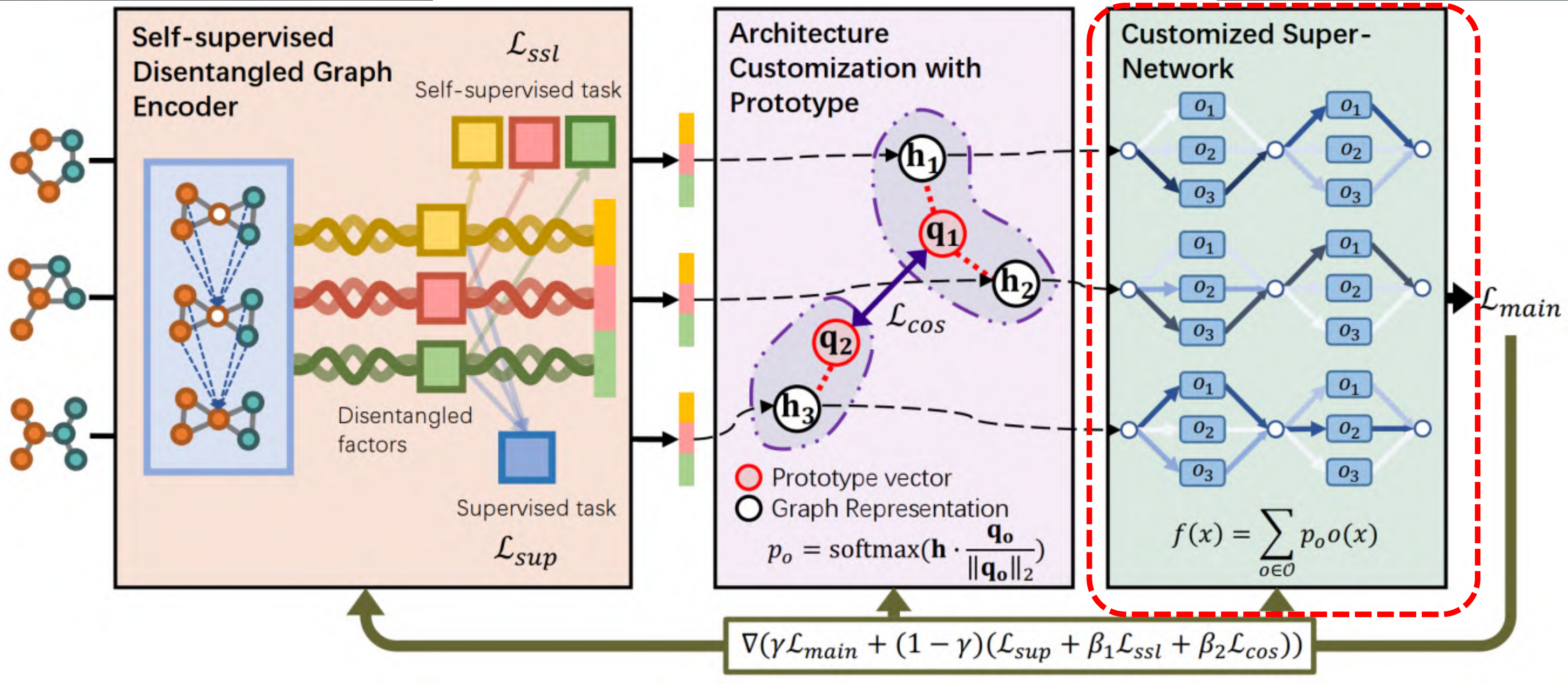
DataFun.

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:**

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

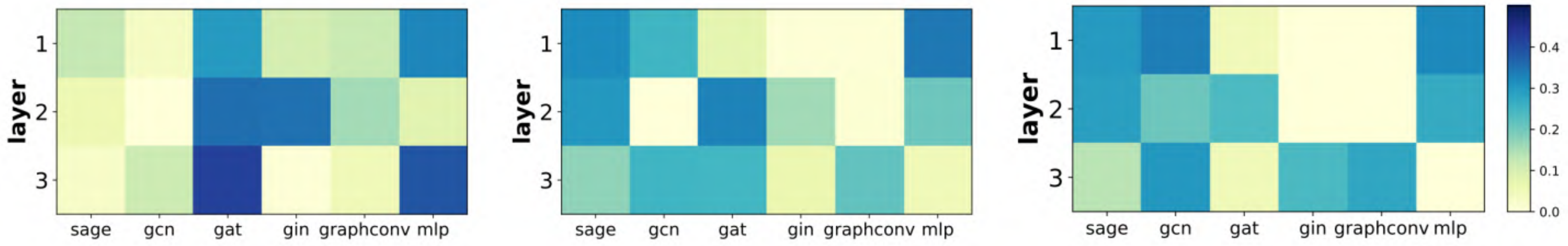
$$\mathcal{L}_{reg} = \mathcal{L}_{sup} + \beta_1 \mathcal{L}_{ssl} + \beta_2 \mathcal{L}_{cos}$$



GRACES: Experiments

bias	$b = 0.7$	$b = 0.8$	$b = 0.9$
GCN	$48.39_{\pm1.69}$	$41.55_{\pm3.88}$	$39.13_{\pm1.76}$
GAT	$50.75_{\pm4.89}$	$42.48_{\pm2.46}$	$40.10_{\pm5.19}$
GIN	$36.83_{\pm5.49}$	$34.83_{\pm3.10}$	$37.45_{\pm3.59}$
SAGE	$46.66_{\pm2.51}$	$44.50_{\pm5.79}$	$44.79_{\pm4.83}$
GraphConv	$47.29_{\pm1.95}$	$44.67_{\pm5.88}$	$44.82_{\pm4.84}$
MLP	$48.27_{\pm1.27}$	$46.73_{\pm3.48}$	$46.41_{\pm2.34}$
ASAP	$54.07_{\pm13.85}$	$48.32_{\pm12.72}$	$43.52_{\pm8.41}$
DIR	$50.08_{\pm3.46}$	$48.22_{\pm6.27}$	$43.11_{\pm5.43}$
random	$45.92_{\pm4.29}$	$51.72_{\pm5.38}$	$45.89_{\pm5.09}$
DARTS	$50.63_{\pm8.90}$	$45.41_{\pm7.71}$	$44.44_{\pm4.42}$
GNAS	$55.18_{\pm18.62}$	$51.64_{\pm19.22}$	$37.56_{\pm5.43}$
PAS	$52.15_{\pm4.35}$	$43.12_{\pm5.95}$	$39.84_{\pm1.67}$
GRACES	$\mathbf{65.72}_{\pm17.47}$	$\mathbf{59.57}_{\pm17.37}$	$\mathbf{50.94}_{\pm8.14}$

dataset	hiv	sider	bace
GCN	$75.99_{\pm1.19}$	$59.84_{\pm1.54}$	$68.93_{\pm6.95}$
GAT	$76.80_{\pm0.58}$	$57.40_{\pm2.01}$	$75.34_{\pm2.36}$
GIN	$77.07_{\pm1.49}$	$57.57_{\pm1.56}$	$73.46_{\pm5.24}$
SAGE	$75.58_{\pm1.40}$	$56.36_{\pm1.32}$	$74.85_{\pm2.74}$
GraphConv	$74.46_{\pm0.86}$	$56.09_{\pm1.06}$	$78.87_{\pm1.74}$
MLP	$70.88_{\pm0.83}$	$58.16_{\pm1.41}$	$71.60_{\pm2.30}$
ASAP	$73.81_{\pm1.17}$	$55.77_{\pm1.18}$	$71.55_{\pm2.74}$
DIR	$77.05_{\pm0.57}$	$57.34_{\pm0.36}$	$76.03_{\pm2.20}$
DARTS	$74.04_{\pm1.75}$	$60.64_{\pm1.37}$	$76.71_{\pm1.83}$
PAS	$71.19_{\pm2.28}$	$59.31_{\pm1.48}$	$76.59_{\pm1.87}$
GRACES	$\mathbf{77.31}_{\pm1.00}$	$\mathbf{61.85}_{\pm2.56}$	$\mathbf{79.46}_{\pm3.04}$



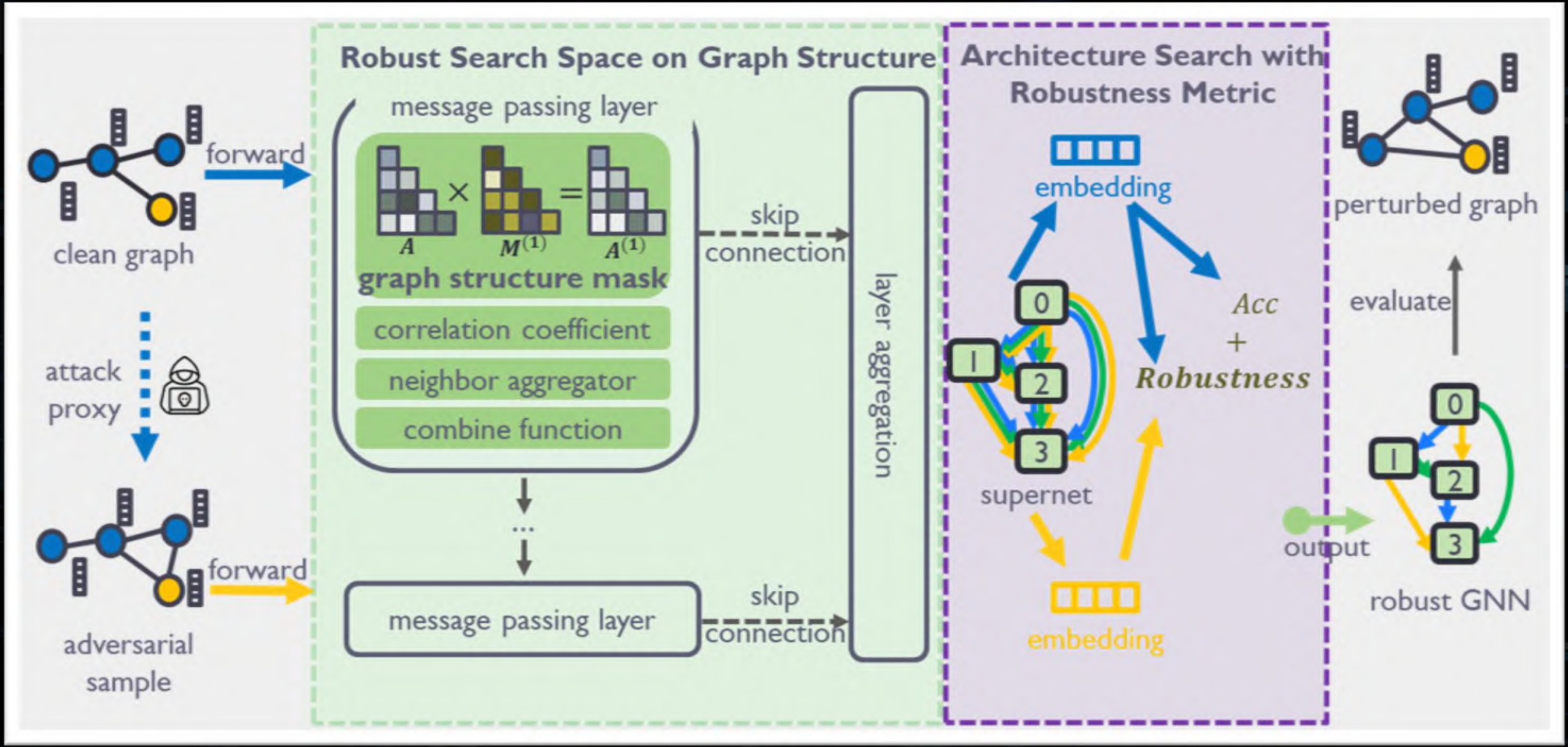
(a) *Tree*-based graphs (b) *Ladder*-based graphs (c) *Wheel*-based graphs

Customization of architectures

DataFun.

Robust Graph Neural Architecture Search

Robust search space and **robustness-aware search strategy** of GraphNAS

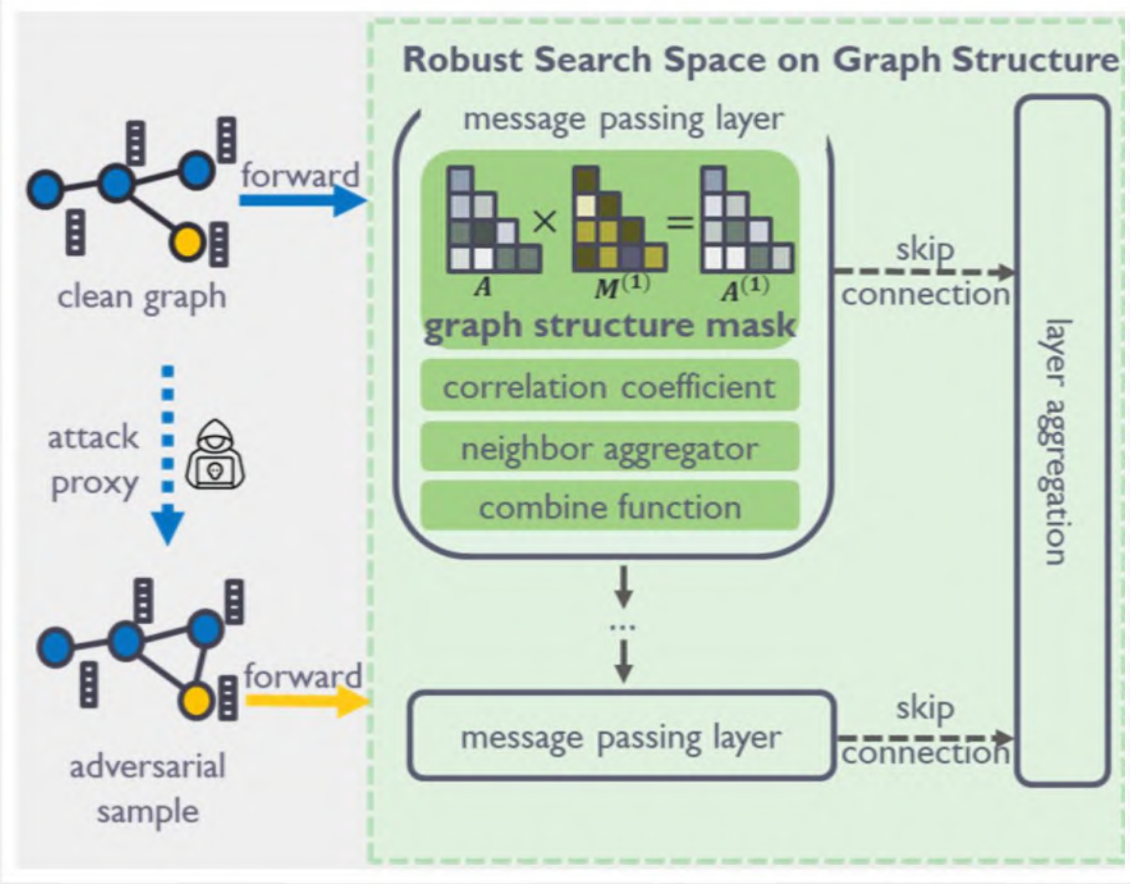


Adversarially Robust Neural Architecture Search for Graph Neural Networks. ***CVPR, 2023.***

G-RNA: Robust Search Space

- Goal: remove noises in the structure
- Method: graph structure mask

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)}\,\mathrm{Comb}\left(\mathbf{h}_i^{(l-1)}, \mathrm{Aggr}\left(\boxed{m_{ij}^{(l)}}\, e_{ij}^{(l)}\,\mathbf{h}_j^{(l-1)}, j \in \tilde{\mathcal{N}}(i)\right)\right)\right)$$



candidate operations

$$\boxed{\mathbf{M}^{(l)}} = \mathcal{D}(\mathbf{A}^{(l-1)}), \mathbf{A}^{(l)} = \mathbf{A} \odot \mathbf{M}^{(l)}$$

$\mathcal{O}_{\mathcal{D}}$	Formula
Identity	$\mathbf{M}^{(l)} = \mathbf{A}^{(l-1)}$
LRA	$\mathbf{A}^{(l-1)} = \mathbf{U}^{(l-1)}\mathbf{S}^{(l-1)}(\mathbf{V}^{(l-1)})^T$, $\mathbf{M}^{(l)} = \mathbf{U}_r^{(l-1)}\mathbf{S}_r^{(l-1)}(\mathbf{V}_r^{(l-1)})^T$
NFS	$m_{ij}^{(l)} = \begin{cases} 0, & if\ a_{ij}^{(l-1)} > 0\ and\ J_{ij} < \tau \\ a_{ij}^{(l-1)}, & otherwise \end{cases}$
NIE	$m_{ij}^{(l)} = \beta m_{ij}^{(l-1)} + (1-\beta)\hat{\alpha}_{ij}^{(l)}$
VPO	$\mathbf{M}^{(l)} = \sum_{v=1}^{V} \theta_v (\mathbf{A}^{(l-1)})^v$

G-RNA: Robustness Metric

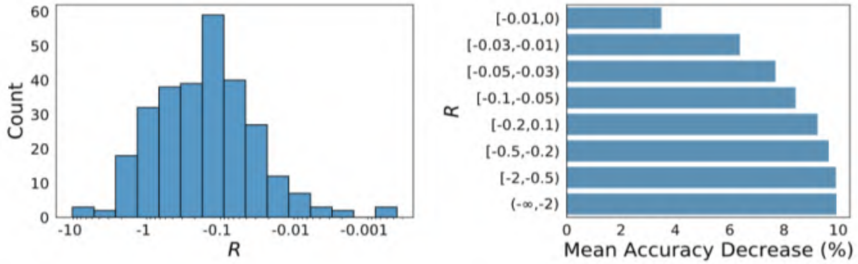
- Goal: consider robustness during search process
- Method: robustness metric

$$\mathcal{R}(\mathbf{A}, f) = -\mathbb{E}_{\mathbf{A}'}\left[\frac{1}{N}\sum_{i=1}^{N} D_{KL}\left(f(\mathbf{A})_i \| f(\mathbf{A}')_i\right)\right], \mathbf{A}' = \mathcal{T}_{\Delta}(\mathbf{A})$$

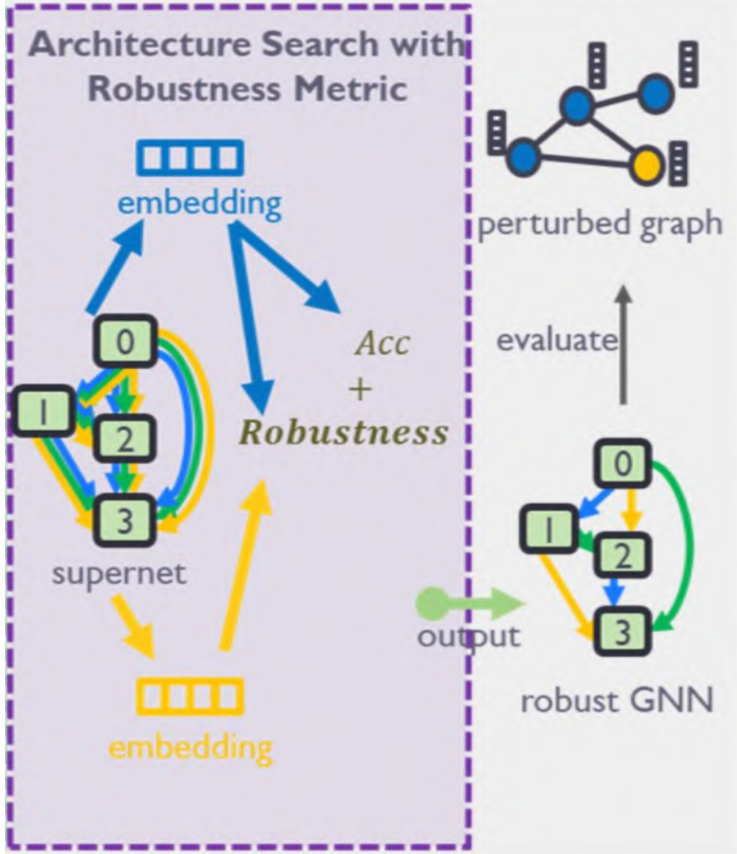
- Approximation: surrogate model

$$\mathcal{R}(\mathbf{A}, f) \approx -\frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(D_{KL}(f(\mathbf{A})_i \| f(\mathbf{A}'_t)_i)\right).$$

- Verification of the robustness metric



- Evolutionary search algorithm
  - Fitness function: $\mathrm{ACC}_{val}(\boldsymbol{\alpha}) + \lambda\mathcal{R}(\boldsymbol{\alpha})$



G-RNA: Experimental Results

- Non-targeted attack

Dataset	Model		Proportion of changed edges (%)					
			0	5	10	15	20	25
PubMed	Vanilla GNN	GCN	86.35±0.15	82.70±0.13	80.56±0.16	77.85±0.17	75.85±0.18	73.68±0.22
		GCN-JK	87.07±0.12	82.76±0.15	81.56±0.18	80.22±0.38	79.14±0.44	77.31±0.24
		GAT	85.28±0.20	81.02±0.31	79.58±0.16	76.39±0.43	74.41±0.20	72.22±0.24
		GAT-JK	85.72±0.14	82.37±0.10	80.60±0.23	78.50±0.15	76.39±0.14	74.02±0.25
	Robust GNN	RGCN	86.64±0.08	82.90±0.18	80.73±0.19	77.86±0.17	75.89±0.15	73.74±0.22
		GCN-Jaccard	87.11±0.04	83.95±0.06	82.30±0.08	80.16±0.07	78.83±0.13	76.86±0.17
		Pro-GNN	-	-	-	-	-	-
		PTDNet	83.87±0.24	74.32±0.44	68.80±0.34	67.32±0.18	66.50±0.12	65.21±0.34
		DropEdge	83.93±0.10	83.24±0.12	82.33±0.15	81.06±0.18	79.21±0.14	76.88±0.28
	Graph NAS	GraphNAS	87.26±0.04	83.56±0.08	80.00±3.98	77.86±2.59	72.97±3.88	68.05±2.26
		GASSO	86.27±0.12	84.15±0.15	83.18±0.21	82.56±0.25	81.73±0.36	83.25±1.26
		G-RNA w/o rob	87.18±0.07	82.59±0.14	80.29±0.17	78.11±0.24	75.98±0.33	73.60±0.25
		G-RNA	**87.48±0.12**	**87.01±0.11**	**86.5±0.14**	**86.04±0.21**	**85.94±0.18**	**85.82±0.12**

DataFun.

AutoML library on Graph

- Graph related
  - PyTorch geometric
  - DGL DEEP GRAPH LIBRARY
  - Paddle Graph Learning
  - PyTorch BigGraph
  - graph-learn

Gap!

- AutoML related
  - NNI Neural Network Intelligence
  - AutoKeras
  - HYPEROPT
  - H_2O AutoML
  - TPOT

However, there is no automated graph machine learning library yet!

DataFun.

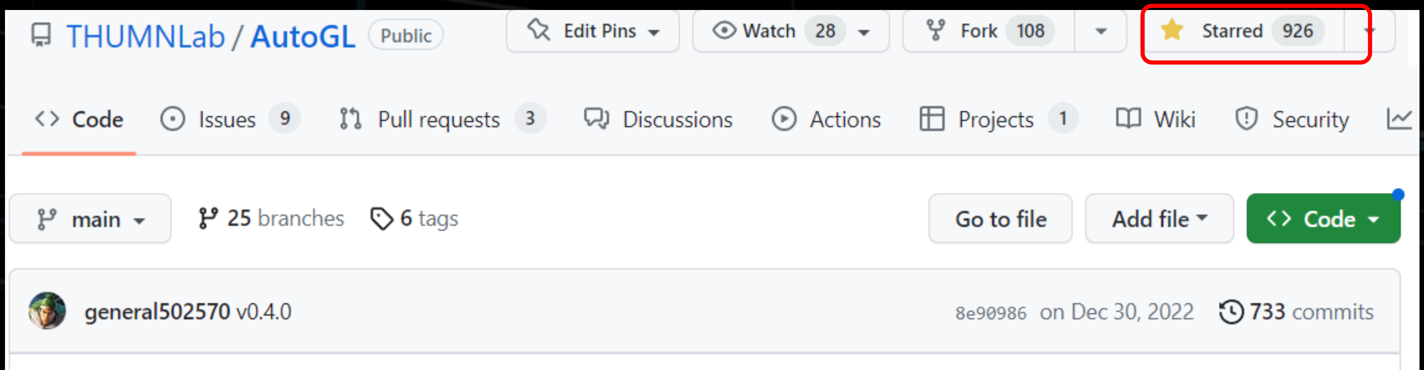
Introduction – AutoGL

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

AutoGL

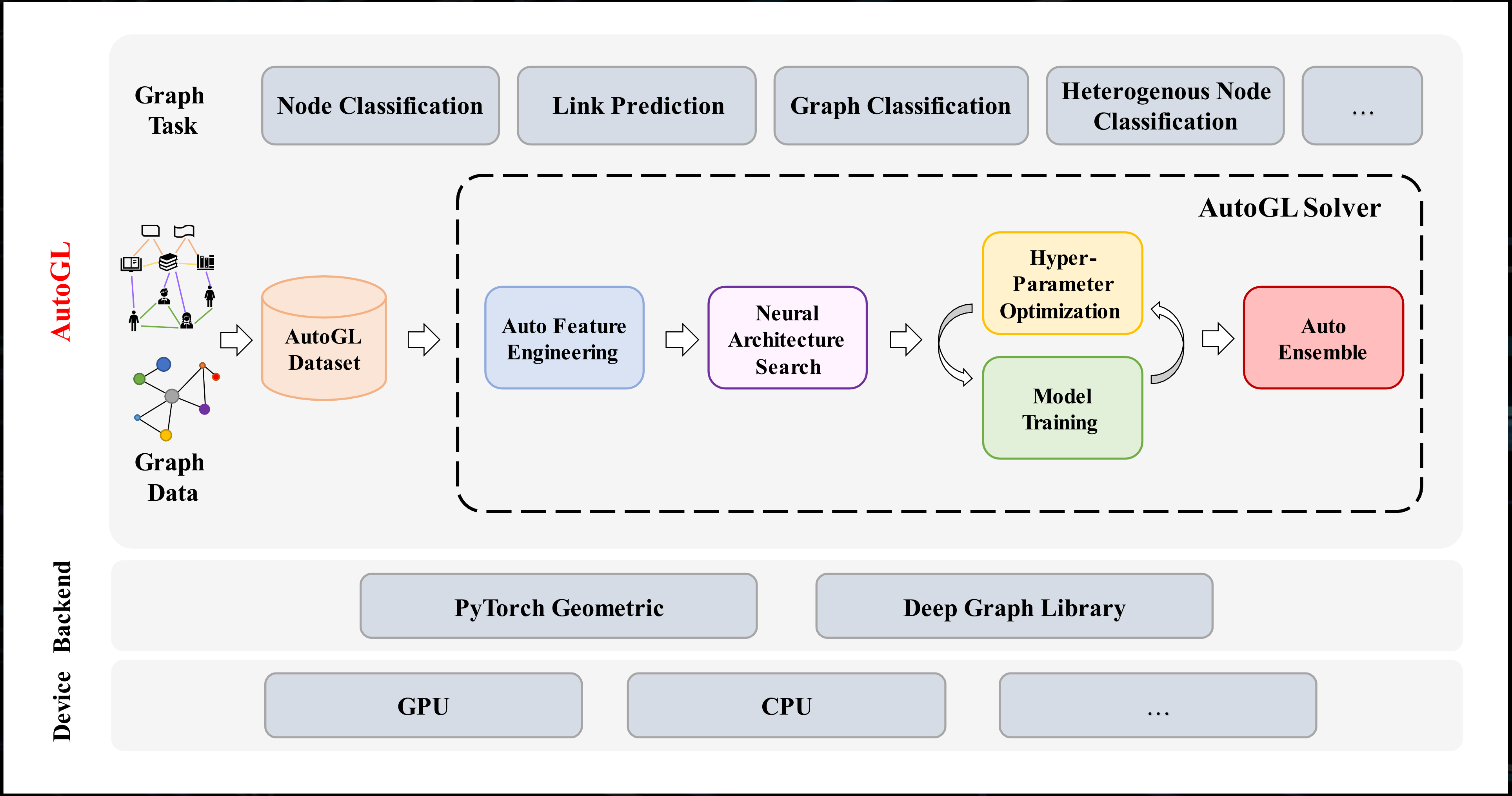
Open source | **Easy to use** | **Flexible to be extended**

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

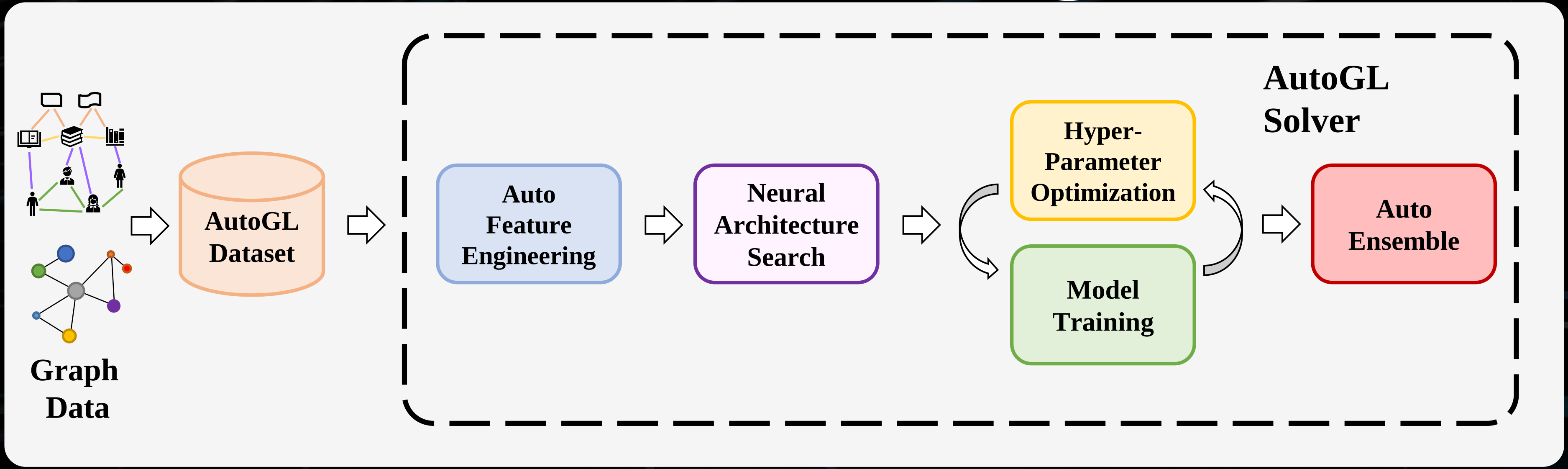


DataFun.

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

DataFun.

AutoGL Roadmap

2020.12 — v0.1
- Initial release
- Overall pipeline
- Dataset, feature engineering, model training, auto ensemble
- Hyper-parameter optimization

2021.7 — v0.2
- Neural Architecture Search
- Graph sampling
- Model enhancement
- Unit tests

2021.12 — v0.3
- DGL backend, PyG 2.0
- Heterogenous graphs
- Decoupled modeling
- NAS enhancements

2022.12 — v0.4
- NAS-Bench-Graph
- Model robustness
- Self-supervised learning
- Refined tutorial/documentation

2023.8/9 — v0.5 (scheduled)
- Lightweight version
- Downstream tasks
  - Biology

  …

- Team member (~10)
  - 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, Yipeng Zhang

Media Coverage

AIM

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

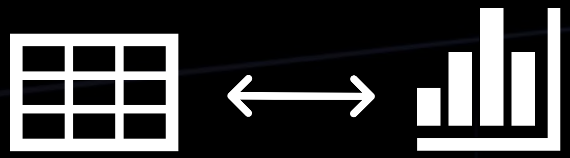
2020-12-23 17:47 来源：澎湃新闻

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。

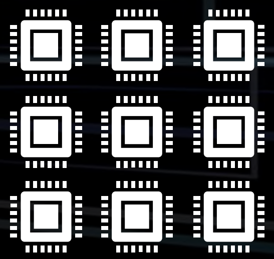
澎湃新闻（www.thepaper.cn）记者从清华大学计算机系朱文武教授领导的网络与媒体实验室获悉，该实验室于2020年12月21日发布了世界首个自动图学习框架与开源工具包AutoGL。AutoGL框架及开源工具包能够为开发人员进行图学习算法设计和调优提供便利，简化图学习算法开发与应用的流程，提升图学习相关的科研和应用效率。

The Evaluation of Graph NAS Methods

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



  - Computationally expensive



  - Diverse evaluation protocols

DataFun.

NAS-Bench-Graph

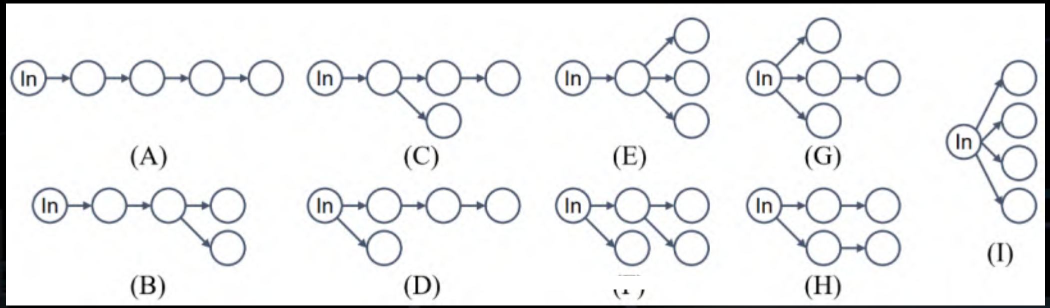
- The first tabular NAS benchmark for GraphNAS
  - Unified, Reproducible, Efficient
  - Provide detailed metrics of all architectures (exhaust 8,000 GPU hours)

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

DataFun.

NAS-Bench-Graph: Usage

- Integrated with two representative libraries: AutoGL and NNI

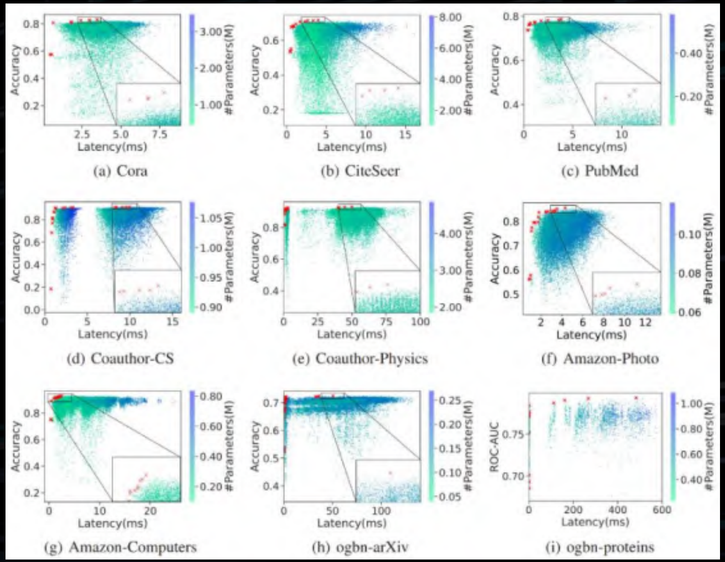
Library	Method	Cora	CiteSeer	PubMed	CS	Physics	Photo	Computers	arXiv	proteins
AutoGL	GNAS	$82.04_{0.17}$	$\mathbf{70.89}_{0.16}$	$77.79_{0.02}$	$90.97_{0.06}$	$92.43_{0.04}$	$92.43_{0.03}$	$84.74_{0.20}$	$72.00_{0.02}$	$\mathbf{78.71}_{0.11}$
	Auto-GNN	$81.80_{0.00}$	$70.76_{0.12}$	$77.69_{0.16}$	$\mathbf{91.04}_{0.04}$	$92.42_{0.16}$	$92.38_{0.01}$	$84.53_{0.14}$	$\mathbf{72.13}_{0.03}$	$78.54_{0.30}$
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

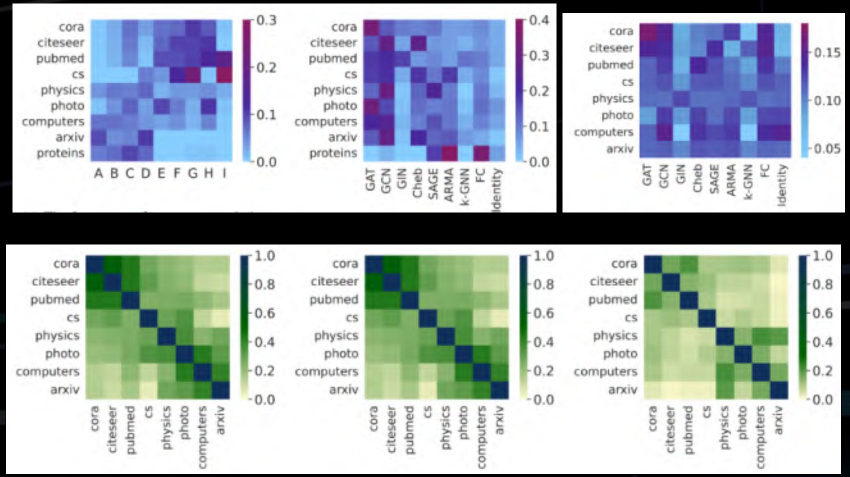
```
from readbench import read
bench = read('cora0.bench')  # dataset and seed
info = bench[arch.valid_hash()]
epoch = 50
info['dur'][epoch][0]   # training performance
info['dur'][epoch][1]   # validation performance
info['dur'][epoch][2]   # testing performance
info['dur'][epoch][3]   # training loss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

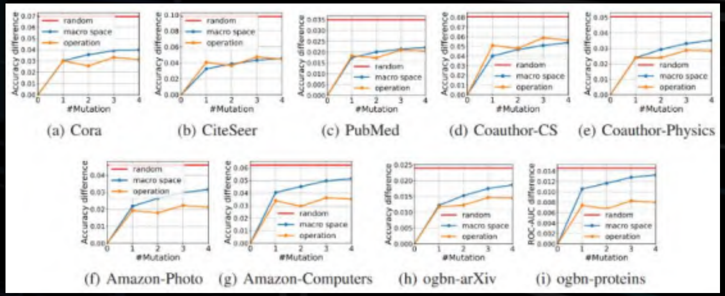
NAS-Bench-Graph: Analysis



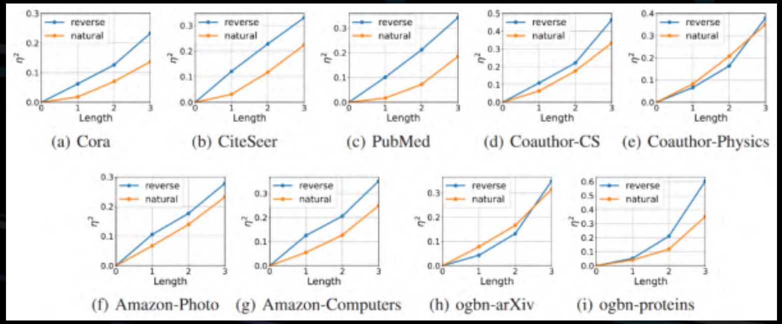
Performance distribution



Architecture distribution & Correlation

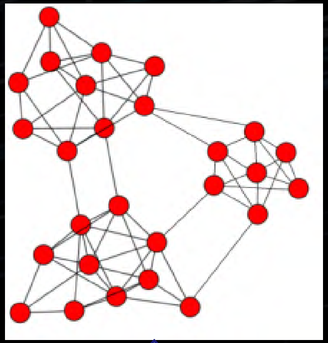


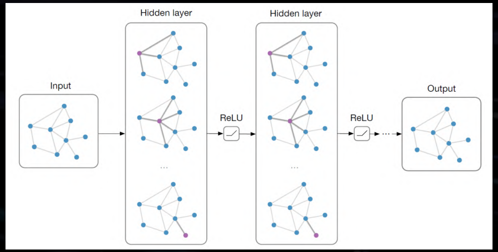
Architecture space smoothness



Influence of operations at different depth

Recap: Our Recent Works on GraphNAS





Graph Structure	Structure Learning	NeurIPS'21
	Dynamic Heterogenous Graph	AAAI'23
Scalability	Billion-scale Graphs	ICML'22
Robustness	Distribution Shifts	ICML'22
	Adversarial Robustness	CVPR'23

NAS-Bench-Graph NeurIPS'22

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.

Beini Xie
Tsinghua Univ.

Zeyang Zhang
Tsinghua Univ.

THANK YOU!

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