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
AGG(·): how to aggregate information from neighbors
- Requirement: permutation-invariant
- Common choices: mean, max, sum, etc.

\( a_{ij} \): the importance of neighbors

COMBINE(·): 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)

**Graph NAS Search Space**

\[
m_{i}^{(l)} = \text{AGG}^{(l)} \left( \left\{ a_{ij}^{(l)} W^{(l)} h_{j}^{(l)}, \forall j \in \mathcal{N}(i) \right\} \right)
\]

\[
h_{i}^{(l+1)} = \sigma \left( \text{COMBINE}^{(l)} \left[ m_{i}^{(l)}, h_{i}^{(l)} \right] \right),
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONST</td>
<td>( a_{ij}^{\text{const}} = 1 )</td>
</tr>
<tr>
<td>GCN</td>
<td>( a_{ij}^{\text{GCN}} = \frac{1}{\sqrt{</td>
</tr>
<tr>
<td>GAT</td>
<td>( a_{ij}^{\text{GAT}} = \text{LeakyReLU} \left( W_{a} [h_{i}, h_{j}] \right) )</td>
</tr>
<tr>
<td>SYM-GAT</td>
<td>( a_{ij}^{\text{SYM}} = a_{ij}^{\text{GAT}} + a_{ji}^{\text{GAT}} )</td>
</tr>
<tr>
<td>COS</td>
<td>( a_{ij}^{\text{COS}} = \cos \left( W_{a} h_{i}, W_{a} h_{j} \right) )</td>
</tr>
<tr>
<td>LINEAR</td>
<td>( a_{ij}^{\text{LINEAR}} = \tanh \left( \text{sum} \left( W_{a} h_{i}, W_{a} h_{j} \right) \right) )</td>
</tr>
<tr>
<td>GENE-LINEAR</td>
<td>( a_{ij}^{\text{GENE-LINEAR}} = \tanh \left( \text{sum} \left( W_{a} h_{i}, W_{a} h_{j} \right) \right) W_{a}' )</td>
</tr>
</tbody>
</table>

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

\[
y = o^{(x,y)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(z_o^{(x,y)})}{\sum_{o' \in \mathcal{O}} \exp(z_{o'}^{(x,y)})} o(x)
\]

\[
\alpha = \alpha - \nabla_\alpha \mathcal{L}_{val}(W(\alpha), \alpha)
\]

\[
W = W - \nabla_W \mathcal{L}_{train}(W, \alpha)
\]
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Search space</th>
<th>Tasks</th>
<th>Search Strategy</th>
<th>Performance Estimation</th>
<th>Other Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphNAS</td>
<td></td>
<td></td>
<td>RNN controller + RL</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>AGNN[43]</td>
<td>✓</td>
<td>✓</td>
<td>Self-designed controller + RL</td>
<td>-</td>
<td>Simplify the micro search space</td>
</tr>
<tr>
<td>SNAG[44]</td>
<td>✓</td>
<td>✓</td>
<td>RNN controller + RL</td>
<td>-</td>
<td>Support heterogeneous graphs</td>
</tr>
<tr>
<td>PDNAS[45]</td>
<td>✓</td>
<td>✓</td>
<td>Differentiable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>POSE[46]</td>
<td>✓</td>
<td>✓</td>
<td>Evolutionary algorithm</td>
<td>-</td>
<td>Sample small graphs for efficiency</td>
</tr>
<tr>
<td>NAS-GNN[47]</td>
<td>✓</td>
<td>✓</td>
<td>Evolutionary algorithm</td>
<td>-</td>
<td>Handle edge features</td>
</tr>
<tr>
<td>AutoGraph[48]</td>
<td>✓</td>
<td>✓</td>
<td>Evolutionary algorithm</td>
<td>-</td>
<td>Search for quantisation options</td>
</tr>
<tr>
<td>GeneticGNN[49]</td>
<td>✓</td>
<td>✓</td>
<td>Differentiable</td>
<td>-</td>
<td>Transfer across datasets and tasks</td>
</tr>
<tr>
<td>EGAN[50]</td>
<td>✓</td>
<td>✓</td>
<td>Evolutionary algorithm</td>
<td>-</td>
<td>Search spatial-temporal modules</td>
</tr>
<tr>
<td>NAS-GCN[51]</td>
<td>✓</td>
<td>✓</td>
<td>Differentiable</td>
<td>-</td>
<td>Search spatial-temporal modules</td>
</tr>
<tr>
<td>LPGNAS[52]</td>
<td>✓</td>
<td>✓</td>
<td>Random search</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>You et al.[53]</td>
<td>✓</td>
<td>✓</td>
<td>Self-designed algorithm</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SAGS[54]</td>
<td>✓</td>
<td>✓</td>
<td>CEM-RL[56]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Peng et al.[55]</td>
<td>✓</td>
<td>✓</td>
<td>Differentiable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GNAS[57]</td>
<td>✓</td>
<td>✓</td>
<td>Differentiable</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AutoSTG[58]</td>
<td>✓</td>
<td>✓</td>
<td>One-shot</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DSS[59]</td>
<td>✓</td>
<td>✓</td>
<td>One-shot</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SANE[60]</td>
<td>✓</td>
<td>✓</td>
<td>One-shot</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AutoAttend[61]</td>
<td>✓</td>
<td>✓</td>
<td>One-shot</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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

Paper collection: https://github.com/THUMNLab/awesome-auto-graph-learning
Challenges for the Existing Methods

- GraphNAS has many unique and unsolved challenges

Graph Structure

Scalability

Robustness

billions of nodes/edges
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

$$P = \Phi\left(\frac{\sqrt{2n|D|}}{(\delta+1)\sqrt{(n+1)(n+2)}}\right).$$

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

- Synthetic datasets:

More structural information

Less structural information

GASSO: Graph Architecture Search with Structure Optimization

Learn graph structure and GNN architecture through a joint optimization scheme

GASSO: Model

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

\[
\begin{align*}
\min_{\mathcal{A}} & \mathcal{L}_{\text{val}}(W^*, \mathcal{A}) \\
\text{s.t.} & \ W^* = \arg\min_W \mathbb{E}_{A \in \Gamma(A)} \mathcal{L}_{\text{train}}(W, A).
\end{align*}
\]

\[
\begin{align*}
\min_{\mathcal{A}} & \mathcal{L}_{\text{val}}(W^*, \mathcal{A}, G^*) \\
\text{s.t.} & \ G^* = \arg\min_G \mathcal{L}_s(W^*, \mathcal{A}, G) \\
& \ W^* = \arg\min_W \mathbb{E}_{A \in \Gamma(A)} \mathcal{L}_{\text{train}}(W, \mathcal{A}, G).
\end{align*}
\]

- Feature Smoothness Constraint

\[
\mathcal{L}_s = \lambda \sum_{i,j}^N G_{ij} \| h_i - h_j \|_2^2 + \sum_{i,j}^N (G_{ij} - G_{o,ij})^2,
\]

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

- Experiments on graph benchmarks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN†</td>
<td>87.40</td>
<td>79.20</td>
<td>88.40</td>
</tr>
<tr>
<td>GAT†</td>
<td>87.26 ± 0.08</td>
<td>77.82 ± 0.11</td>
<td>86.83 ± 0.11</td>
</tr>
<tr>
<td>ARMA†</td>
<td>86.06 ± 0.05</td>
<td>76.50 ± 0.00</td>
<td>88.70 ± 0.24</td>
</tr>
<tr>
<td>DropEdge†</td>
<td>87.60 ± 0.05</td>
<td>78.57 ± 0.00</td>
<td>87.34 ± 0.24</td>
</tr>
<tr>
<td>DARTS</td>
<td>86.18 ± 0.36</td>
<td>74.96 ± 0.10</td>
<td>88.38 ± 0.18</td>
</tr>
<tr>
<td>GDAS</td>
<td>85.48 ± 0.30</td>
<td>74.20 ± 0.11</td>
<td>89.50 ± 0.14</td>
</tr>
<tr>
<td>ASAP</td>
<td>85.21 ± 0.13</td>
<td>75.14 ± 0.09</td>
<td>88.65 ± 0.10</td>
</tr>
<tr>
<td>XNAS</td>
<td>86.80 ± 0.14</td>
<td>76.33 ± 0.09</td>
<td>88.61 ± 0.25</td>
</tr>
<tr>
<td>GraphNAS‡</td>
<td>86.83 ± 0.56</td>
<td>79.05 ± 0.28</td>
<td>89.99 ± 0.43</td>
</tr>
<tr>
<td><strong>GASSO</strong></td>
<td><strong>87.63 ± 0.29</strong></td>
<td><strong>79.61 ± 0.32</strong></td>
<td><strong>90.52 ± 0.24</strong></td>
</tr>
</tbody>
</table>

- Experiments on larger graph datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Physics</th>
<th>CoraFull</th>
<th>ogbn-arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>95.94</td>
<td>68.08</td>
<td>70.39</td>
</tr>
<tr>
<td>GAT</td>
<td>95.86</td>
<td>65.78</td>
<td>68.53</td>
</tr>
<tr>
<td>DARTS</td>
<td>95.74</td>
<td>68.51</td>
<td>69.52</td>
</tr>
<tr>
<td><strong>GASSO</strong></td>
<td><strong>96.38</strong></td>
<td><strong>68.89</strong></td>
<td><strong>70.52</strong></td>
</tr>
</tbody>
</table>
Dynamic Heterogenous Graphs

- **Dynamic**: structures and features evolve through time
- **Heterogeneous**: various node and edge types
- More complicated structural patterns

Citation network

Finance Graph

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.
DHGAS: Dynamic Heterogeneous Graph Attention

- **Goal:** capture dynamic heterogeneous 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., $\mathcal{N}^t_r(u) = \{v : (u,v) \in \mathcal{E}^t, \phi_e(u,v) = r\}$. 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., $\mathcal{N}(u) = \bigcup_{r,t} \mathcal{N}^t_r(u)$.

- **Time/type-aware node mapping functions**

- **Time/type-aware relation mapping functions**

- **Update with time/type-aware attention**

$$
\begin{align*}
q^t_v &= \mathcal{F}_q^{N}(v,\phi_n(v),t)(h^t_v), \\
k^t_v &= \mathcal{F}_k^{N}(v,\phi_n(v),t)(h^t_v), \\
v^t_v &= \mathcal{F}_v^{N}(v,\phi_n(v),t)(h^t_v), \\
\alpha_{u,v} &= \mathcal{F}_\phi^{R}(u,v,\phi_e(u,v),\Delta t)(q^t_u, k^t_v), \\
\mathcal{F}_\phi^{R}(u,v,\phi_e(u,v),\Delta t)(q, k) &= \frac{qW^{\phi_e(u,v),\Delta t}k^T}{\sqrt{d}}, \\
\end{align*}
$$

$$
\begin{align*}
\hat{h}_u^t &= \text{Update}(h^t_u, \sum_{v \in \mathcal{N}(u)} \hat{\alpha}_{u,v} v^t_v)), \\
\hat{\alpha}_{u,v} &= \frac{\exp(\alpha_{u,v})}{\sum_{v' \in \mathcal{N}(u)} \exp(\alpha_{u,v'})}.
\end{align*}
$$
DHGAS: Attention Parameterization and Localization Space

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

- **Parameterization Space**: how to parameterize attention
  \[ A^{Pa} = A^N \times A^R \]
  \[ A^N = \{1, ..., K_N\}^{T \times |C_n|} \]
  \[ A^R = \{1, ..., K_R\}^{2T \times |C_e|} \]

- **Localization Space**: Locate where to apply attention
  \[ A^{Lo} = \{0, 1\}^{T \times T \times |C_e|} \]

- 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

\[
\tilde{F}(x) = \sum_{i=1}^{|A|} \frac{\exp(\beta_i)}{\sum_{j=1}^{|A|} \exp(\beta_j)} F_i(x)
\]

- Update super-networks weights and architecture differentiably

\[
\begin{align*}
    w &\leftarrow w - \eta_w \frac{\partial L_{\text{train}}}{\partial w}, \\
    \beta &\leftarrow \beta - \eta_\beta \frac{\partial L_{\text{val}}}{\partial \beta}
\end{align*}
\]

- Multi-stage training to stabilize the searching process
DHGAS: Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Link Prediction (AUC%)</th>
<th>Node Classification (F1%)</th>
<th>Node Regression (MAE)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aminer</td>
<td>Ecomm</td>
<td>Yelp</td>
</tr>
<tr>
<td>GCN</td>
<td>73.84 ± 0.06</td>
<td>77.94 ± 0.22</td>
<td>37.02 ± 0.00</td>
</tr>
<tr>
<td>GAT</td>
<td>80.84 ± 0.96</td>
<td>78.49 ± 0.31</td>
<td>35.54 ± 0.00</td>
</tr>
<tr>
<td>RGCN</td>
<td>82.75 ± 0.12</td>
<td>82.27 ± 0.51</td>
<td>37.75 ± 0.00</td>
</tr>
<tr>
<td>HGT</td>
<td>78.43 ± 1.81</td>
<td>81.09 ± 0.52</td>
<td>34.62 ± 0.00</td>
</tr>
<tr>
<td>DyHATR</td>
<td>74.24 ± 2.09</td>
<td>71.69 ± 0.90</td>
<td>34.49 ± 0.16</td>
</tr>
<tr>
<td>HGT+</td>
<td>85.60 ± 0.12</td>
<td>76.68 ± 0.85</td>
<td>38.33 ± 0.00</td>
</tr>
<tr>
<td>HTGNN</td>
<td>78.08 ± 0.80</td>
<td>76.78 ± 6.37</td>
<td>36.33 ± 0.07</td>
</tr>
<tr>
<td>GraphNAS</td>
<td>81.61 ± 0.98</td>
<td>79.37 ± 0.21</td>
<td>37.73 ± 0.00</td>
</tr>
<tr>
<td>DiffMG</td>
<td>85.04 ± 0.30</td>
<td>81.69 ± 0.06</td>
<td>38.65 ± 0.00</td>
</tr>
<tr>
<td>DHGAS</td>
<td>88.13 ± 0.18</td>
<td>86.56 ± 0.58</td>
<td>41.99 ± 0.18</td>
</tr>
</tbody>
</table>

Significantly outperforms baselines for various downstream tasks.
DHGAS: Experiments

- Jointly modeling dynamic and heterogeneous information
- Tailor optimal attention mechanisms for different datasets

Ablation studies

Performance and costs tradeoff

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**
Jointly sample subgraphs and architectures to find the most suitable architecture.
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

**Reward function**: performance + regularizer

$\theta = \arg\max_\theta (R(\theta) + \beta H(\Gamma(\theta)))$
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

GAUSS: Architecture Peer Learning on Graph
# GAUSS: Experiments

## Table 2

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Nodes</th>
<th>#Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>18,333</td>
<td>81,894</td>
</tr>
<tr>
<td>PHYSICS</td>
<td>34,493</td>
<td>247,962</td>
</tr>
<tr>
<td>ARXIV</td>
<td>169,343</td>
<td>1,166,243</td>
</tr>
<tr>
<td>PRODUCTS</td>
<td>2,449,029</td>
<td>61,859,140</td>
</tr>
<tr>
<td>PAPERS100M</td>
<td>111,059,956</td>
<td>1,615,685,872</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CS valid</th>
<th>CS test</th>
<th>Physics valid</th>
<th>Physics test</th>
<th>Arxiv valid</th>
<th>Arxiv test</th>
<th>Products valid</th>
<th>Products test</th>
<th>PAPERS100M valid</th>
<th>PAPERS100M test</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>94.10±0.21</td>
<td>93.98±0.21</td>
<td>96.29±0.05</td>
<td>96.38±0.07</td>
<td>72.76±0.15</td>
<td>71.70±0.18</td>
<td>91.75±0.04</td>
<td>80.19±0.46</td>
<td>70.32±0.11</td>
<td>67.06±0.17</td>
</tr>
<tr>
<td>GAT</td>
<td>93.74±0.27</td>
<td>93.48±0.36</td>
<td>96.25±0.23</td>
<td>96.37±0.23</td>
<td>73.19±0.12</td>
<td>71.85±0.21</td>
<td>90.75±0.16</td>
<td>80.59±0.40</td>
<td>70.26±0.16</td>
<td>67.26±0.06</td>
</tr>
<tr>
<td>SAGE</td>
<td>95.65±0.07</td>
<td>95.33±0.11</td>
<td>96.76±0.10</td>
<td>96.72±0.07</td>
<td>73.11±0.08</td>
<td>71.78±0.15</td>
<td>91.75±0.04</td>
<td>80.19±0.46</td>
<td>70.32±0.11</td>
<td>67.06±0.17</td>
</tr>
<tr>
<td>GIN</td>
<td>92.00±0.43</td>
<td>92.14±0.34</td>
<td>96.03±0.11</td>
<td>96.04±0.15</td>
<td>71.16±0.10</td>
<td>70.01±0.33</td>
<td>91.58±0.30</td>
<td>79.07±0.52</td>
<td>68.98±0.16</td>
<td>65.78±0.09</td>
</tr>
<tr>
<td>GraphNAS</td>
<td>94.90±0.14</td>
<td>94.67±0.23</td>
<td>96.76±0.10</td>
<td>96.72±0.07</td>
<td>72.76±0.15</td>
<td>71.70±0.18</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
<td>OOT</td>
</tr>
<tr>
<td>SGAS</td>
<td>95.62±0.06</td>
<td>95.44±0.06</td>
<td>96.44±0.10</td>
<td>96.50±0.11</td>
<td>72.38±0.11</td>
<td>71.34±0.25</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>DARTS</td>
<td>95.62±0.06</td>
<td>95.44±0.06</td>
<td>96.21±0.16</td>
<td>96.40±0.21</td>
<td>73.43±0.07</td>
<td>72.10±0.25</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>EGAN</td>
<td>95.60±0.10</td>
<td>95.43±0.05</td>
<td>96.39±0.18</td>
<td>96.45±0.19</td>
<td>72.91±0.25</td>
<td>71.75±0.35</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>Basic</td>
<td>95.13±0.07</td>
<td>95.45±0.05</td>
<td>96.25±0.06</td>
<td>96.53±0.06</td>
<td>73.28±0.06</td>
<td>72.00±0.33</td>
<td>91.79±0.11</td>
<td>80.56±0.39</td>
<td>69.49±0.37</td>
<td>66.24±0.46</td>
</tr>
<tr>
<td>GAUSS</td>
<td>96.08±0.11</td>
<td>96.49±0.11</td>
<td>96.79±0.06</td>
<td>96.76±0.08</td>
<td>73.63±0.10</td>
<td>72.35±0.21</td>
<td>91.60±0.12</td>
<td>81.26±0.36</td>
<td>70.57±0.07</td>
<td>67.32±0.18</td>
</tr>
</tbody>
</table>

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

- Clean Graph
- Perturbed Graph

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
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:
- Regularizer to avoid mode collapse:
**Goal**: learn parameters for the customized architectures

**Method**: customized super-network

**Loss functions**:

\[ L = \gamma L_{\text{main}} + (1 - \gamma) L_{\text{reg}} \]

\[ L_{\text{reg}} = L_{\text{sup}} + \beta_1 L_{\text{ssl}} + \beta_2 L_{\cos} \]
### GRACES: Experiments

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

<table>
<thead>
<tr>
<th>dataset</th>
<th>hiv</th>
<th>sider</th>
<th>bace</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>75.99±1.19</td>
<td>59.84±1.54</td>
<td>68.93±6.95</td>
</tr>
<tr>
<td>GAT</td>
<td>76.80±0.80</td>
<td>57.40±2.01</td>
<td>75.34±2.36</td>
</tr>
<tr>
<td>GIN</td>
<td>77.07±1.49</td>
<td>57.57±1.56</td>
<td>73.46±5.24</td>
</tr>
<tr>
<td>SAGE</td>
<td>75.58±1.40</td>
<td>56.36±1.32</td>
<td>74.85±2.74</td>
</tr>
<tr>
<td>GraphConv</td>
<td>74.46±0.86</td>
<td>56.09±1.06</td>
<td>78.87±1.74</td>
</tr>
<tr>
<td>MLP</td>
<td>70.88±0.83</td>
<td>58.16±1.41</td>
<td>71.60±2.30</td>
</tr>
<tr>
<td>ASAP</td>
<td>73.81±1.17</td>
<td>55.77±1.18</td>
<td>71.55±2.74</td>
</tr>
<tr>
<td>DIR</td>
<td>77.05±0.57</td>
<td>57.34±0.36</td>
<td>76.03±2.20</td>
</tr>
<tr>
<td>DARTS</td>
<td>74.04±1.75</td>
<td>60.64±1.37</td>
<td>76.71±1.83</td>
</tr>
<tr>
<td>PAS</td>
<td>71.19±2.28</td>
<td>59.31±1.48</td>
<td>76.59±1.87</td>
</tr>
<tr>
<td><strong>GRACES</strong></td>
<td><strong>77.31±1.00</strong></td>
<td><strong>61.85±2.56</strong></td>
<td><strong>79.46±3.04</strong></td>
</tr>
</tbody>
</table>

**Customization of architectures**
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

\[ h_i^{(l)} = \sigma \left( W^{(l)} \text{Comb} \left( h_i^{(l-1)}, \text{Aggr} \left[ m_{ij}^{(l)}, e_j^{(l)} h_j^{(l-1)}, j \in \tilde{N}(i) \right] \right) \right) \]

<table>
<thead>
<tr>
<th>Operator</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identity</strong></td>
<td>( M^{(l)} = A^{(l-1)} )</td>
</tr>
<tr>
<td><strong>LRA</strong></td>
<td>( A^{(l-1)} = U^{(l-1)} S^{(l-1)} (V^{(l-1)})^T ), ( M^{(l)} = U^{(l-1)} S^{(l-1)} (V^{(l-1)})^T )</td>
</tr>
<tr>
<td><strong>NFS</strong></td>
<td>( m_{ij}^{(l)} = \begin{cases} 0, &amp; \text{if } a_{ij}^{(l-1)} &gt; 0 \text{ and } J_{ij} &lt; \tau \ a_{ij}^{(l-1)}, &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td><strong>NIE</strong></td>
<td>( m_{ij}^{(l)} = \beta m_{ij}^{(l-1)} + (1 - \beta) \hat{a}_{ij}^{(l)} )</td>
</tr>
<tr>
<td><strong>VPO</strong></td>
<td>( M^{(l)} = \sum_{v=1}^{V} \theta_v (A^{(l-1)})_v )</td>
</tr>
</tbody>
</table>
**G-RNA: Robustness Metric**

- **Goal:** consider robustness during search process
- **Method:** robustness metric
  
  \[
  R(A, f) = -E_{A'} \left[ \frac{1}{N} \sum_{i=1}^{N} D_{KL}(f(A)_i || f(A')_i) \right], A' = T_{\Delta}(A)
  \]

- **Approximation:** surrogate model
  \[
  R(A, f) \approx -\frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} (D_{KL}(f(A)_i || f(A'_{i}))).
  \]

- **Verification of the robustness metric**

- **Evolutionary search algorithm**
  - **Fitness function:** \( ACC_{val}(\alpha) + \lambda R(\alpha) \)
# G-RNA: Experimental Results

- **Non-targeted attack**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Proportion of changed edges (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Vanilla GNN</td>
<td>GCN</td>
<td>86.35±0.15</td>
</tr>
<tr>
<td></td>
<td>GCN-JK</td>
<td>87.07±0.12</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>85.28±0.20</td>
</tr>
<tr>
<td></td>
<td>GAT-JK</td>
<td>85.72±0.14</td>
</tr>
<tr>
<td>PubMed</td>
<td>RGCN</td>
<td>86.64±0.08</td>
</tr>
<tr>
<td>Robust GNN</td>
<td>GCN-Jaccard</td>
<td>87.11±0.04</td>
</tr>
<tr>
<td></td>
<td>Pro-GNN</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PTDNet</td>
<td>83.87±0.24</td>
</tr>
<tr>
<td></td>
<td>DropEdge</td>
<td>83.93±0.10</td>
</tr>
<tr>
<td>Graph NAS</td>
<td>GraphNAS</td>
<td>87.26±0.04</td>
</tr>
<tr>
<td></td>
<td>GASSO</td>
<td>86.27±0.12</td>
</tr>
<tr>
<td></td>
<td>G-RNA w/o rob</td>
<td>87.18±0.07</td>
</tr>
<tr>
<td></td>
<td><strong>G-RNA</strong></td>
<td>87.48±0.12</td>
</tr>
</tbody>
</table>
However, there is no automated graph machine learning library yet!
Introduction – AutoGL

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

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
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

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

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

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

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

- **v0.5 (scheduled)**
  - Lightweight version
  - Downstream tasks
  - Biology
  - ...  

- **2020.12**

- **2021.7**

- **2021.12**

- **2022.12**

- **2023.8/9**

**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
Meet AutoGL: The First Ever AutoML Framework for Graph Datasets

31/10/2020
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)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Type</th>
<th>Search Space</th>
<th>Data</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS-Bench-101</td>
<td>Tabular</td>
<td>423k</td>
<td>CV</td>
<td>1</td>
</tr>
<tr>
<td>NAS-Bench-201</td>
<td>Tabular</td>
<td>6k</td>
<td>CV</td>
<td>3</td>
</tr>
<tr>
<td>NAS-Bench-1shot1</td>
<td>Tabular</td>
<td>364k</td>
<td>CV</td>
<td>1</td>
</tr>
<tr>
<td>NAS-Bench-ASR</td>
<td>Tabular</td>
<td>8k</td>
<td>Acoustics</td>
<td>1</td>
</tr>
<tr>
<td>NAS-Bench-NLP</td>
<td>Tabular</td>
<td>14k</td>
<td>NLP</td>
<td>2</td>
</tr>
<tr>
<td>HW-NAS-Bench</td>
<td>Tabular</td>
<td>6k</td>
<td>CV</td>
<td>3</td>
</tr>
<tr>
<td>NATS-Bench</td>
<td>Tabular</td>
<td>32k</td>
<td>CV</td>
<td>3</td>
</tr>
<tr>
<td>NAs-HPO-Bench-II</td>
<td>Surrogate</td>
<td>192k</td>
<td>CV</td>
<td>1</td>
</tr>
<tr>
<td>NAS-Bench-MR</td>
<td>Surrogate</td>
<td>$10^{23}$</td>
<td>CV</td>
<td>4</td>
</tr>
<tr>
<td>TransNAS-Bench</td>
<td>Tabular</td>
<td>7k</td>
<td>CV</td>
<td>14</td>
</tr>
<tr>
<td>NAS-Bench-111</td>
<td>Surrogate</td>
<td>423k</td>
<td>CV</td>
<td>1</td>
</tr>
<tr>
<td>NAS-Bench-311</td>
<td>Surrogate</td>
<td>$10^{18}$</td>
<td>CV</td>
<td>1</td>
</tr>
<tr>
<td>NAS-Bench-Zero</td>
<td>Tabular</td>
<td>34k</td>
<td>CV</td>
<td>3</td>
</tr>
<tr>
<td>Surr-NAS-Bench-FBNet</td>
<td>Surrogate</td>
<td>$10^{21}$</td>
<td>CV</td>
<td>2</td>
</tr>
<tr>
<td>NAS-Bench-Graph</td>
<td>Tabular</td>
<td>26k</td>
<td>Graph</td>
<td>9</td>
</tr>
</tbody>
</table>
NAS-Bench-Graph: Designs

- Search space:
  - Macro space:

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

- Datasets:

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

26,206 architectures cover representative GNNs

9 datasets different sizes/domains
NAS-Bench-Graph: Usage

- Integrated with two representative libraries: AutoGL and NNI

<table>
<thead>
<tr>
<th>Library</th>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
<th>CS</th>
<th>Physics</th>
<th>Photo</th>
<th>Computers</th>
<th>arXiv</th>
<th>proteins</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoGL</td>
<td>GNAS</td>
<td>82.04</td>
<td>70.89</td>
<td>77.79</td>
<td>90.97</td>
<td>92.43</td>
<td>92.43</td>
<td>84.74</td>
<td>72.00</td>
<td>78.71</td>
</tr>
<tr>
<td></td>
<td>Auto-GNN</td>
<td>81.80</td>
<td>70.76</td>
<td>77.69</td>
<td>91.04</td>
<td>92.42</td>
<td>92.38</td>
<td>84.53</td>
<td>72.13</td>
<td>78.54</td>
</tr>
<tr>
<td>NNI</td>
<td>Random</td>
<td>82.09</td>
<td>70.49</td>
<td>77.91</td>
<td>90.93</td>
<td>92.35</td>
<td>92.44</td>
<td>84.78</td>
<td>72.04</td>
<td>78.32</td>
</tr>
<tr>
<td></td>
<td>EA</td>
<td>81.85</td>
<td>70.48</td>
<td>77.96</td>
<td>90.60</td>
<td>92.22</td>
<td>92.43</td>
<td>84.29</td>
<td>71.91</td>
<td>77.93</td>
</tr>
<tr>
<td></td>
<td>RL</td>
<td>82.27</td>
<td>70.66</td>
<td>77.96</td>
<td>90.98</td>
<td>92.48</td>
<td>92.42</td>
<td>84.90</td>
<td>72.13</td>
<td>78.52</td>
</tr>
<tr>
<td>The top 5%</td>
<td></td>
<td>80.63</td>
<td>69.07</td>
<td>76.60</td>
<td>90.01</td>
<td>91.67</td>
<td>91.57</td>
<td>82.77</td>
<td>71.69</td>
<td>78.37</td>
</tr>
</tbody>
</table>

- Example: ~10 lines of codes

```python
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/THU姆NLab/NAS-Bench-Graph
NAS-Bench-Graph: Analysis

Performance distribution

Architecture space smoothness

Architecture distribution & Correlation

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
  - Distribution Shifts
    - ICML'22
  - Adversarial Robustness
    - CVPR'23

- **Robustness**

**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.

Beini Xie
Tsinghua Univ.

Zeyang Zhang
Tsinghua Univ.
THANK YOU!

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