Automated Machine Learning on Graphs: A Survey

Ziwei Zhang, Xin Wang, Wenwu Zhu
Tsinghua University
Graphs are Ubiquitous

Biology Network

Social Network

Traffic Network
Graph Tasks

Link Prediction

Graph Classification

Node Classification

Images are from search engines
Graph Applications

Natural Language Processing

Computer Vision

Reasoning

Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling, *EMNLP 2017*

Neural Motifs: Scene Graph Parsing with Global Context, *CVPR 2018*

Learning by Abstraction: The Neural State Machine, *NeurIPS 2019*
Graph Applications

- Structural Engineering
- Physical Simulation
- Drug Repurposing for Covid-19

Learning to Simulate and Design for Structural Engineering, ICML 2020
JAX, M.D. A Framework for Differentiable Physics, NeurIPS 2020
Network Medicine Framework for Identifying Drug Repurposing Opportunities for COVID-19, arXiv 2020
Graph in Industry

- Application scenario: recommendation, prediction, classification, anomaly detection, generation, etc.
- Many tech giants have developed their graph systems
  - Alibaba: Graph-Learn(AliGraph), Euler
  - Amazon: Deep Graph Library (DGL)
  - Baidu: Paddle Graph Learning (PGL)
  - DeepMind: Graph Nets
  - Facebook: PyTorch-BigGraph (PBG)
  - Tencent: Plato

Machine learning on graphs has important and diverse applications!
Machine Learning on Graphs

Network Embedding

Graph Neural Networks


Network Embedding

- Learn vectorized representation of nodes
- Then apply classical vector-based machine learning algorithms
Graph Neural Network

- Design neural networks directly applicable for graphs for end-to-end learning
- Message-passing framework: nodes exchange messages along structures
Problems in Existing Graph Learning Methods

- Manually design architectures and hyper-parameters through trial-and-error
- Each task needs to be handled separately

Automated graph machine learning is critically needed!
A Glance of AutoML

Design ML methods → Design AutoML methods
ML vs. AutoML

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

- **Free human** out of the loop
- **High** optimization efficiency
- **Discover & extract** patterns and combinations **automatically**
Automated Graph Learning

- Automated Machine Learning on Graph
  - Graph Hyper-Parameter Optimization (HPO)
  - Graph Neural Architecture Search (NAS)

- The key: *Graph Structure!*

Various diverse graph structures may place complex impacts on graph HPO and graph NAS.
Challenge: Uniqueness of graph ML

Data

G = (V, E)

NN architecture

Search Space

- zeroize
- skip-connect
- 1x1 conv
- 3x3 conv
- 3x3 avg pool

predefined operation set

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

Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017
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
Challenge: Complexity and diversity of graph tasks

- Link Prediction
- Community Detection
- Node Classification
- Network Distance
- Node Importance
- Graph Classification
- Graph Matching

Various graph properties

Various applications

Various domains

- No single method can perfectly handle all scenarios
Social Networks
- WeChat: 1.2 billion monthly active users (Sep 2020)
- 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
- 133 million authors, 277 million publications, 1.1 billion citations (AMiner, Feb 2021)

Challenge: how to handle billion-scale graphs?
Hyper-Parameter Optimization

- Goal: automatically find the optimal hyper-parameters

- Formulation: bi-level optimization

\[
\min_{\alpha \in \mathcal{A}} \mathcal{L}_{\text{val}} (W^*(\alpha), \alpha) \\
\text{s.t. } W^*(\alpha) = \arg\min_W (\mathcal{L}_{\text{train}} (W, \alpha))
\]

- Challenge: each trial of the inner loop on graph is computationally expensive, especially for large-scale graphs
AutoNE: Framework

Transfer the knowledge about optimal hyper-parameters from sampled subgraphs to the original massive graph

**Goal**: sample representative subgraphs that share similar properties with the original large-scale graph

**Challenge**: preserve diversity of the origin graph

**Method**: multi-start random walk strategy

- Supervised: nodes with different labels
- Unsupervised: from different discovered communities, e.g., a greedy algorithm that maximizes modularity
**AutoNE: Signature Extraction Module**

- **Goal**: learn a vector representation for each subgraph so that knowledge can be transferred across different subgraphs
- **Challenge**: learn comprehensive graph signatures
- **Method**: NetLSD [Tsitsulin et al. KDD18]
  - Based on spectral graph theory, heat diffusion process on a graph
    \[ h_t(G) = tr(H_t) = tr(e^{-tL}) = \sum_j e^{-t\lambda_j} \]
**Goal**: transfer knowledge about hyper-parameters in sampled subgraphs to the original large-scale graph

**Assumption**: two similar graphs have similar optimal hyper-parameter

**Method**: Gaussian Process based meta-learner

$$
\ln p(f \mid X) = -\frac{1}{2} f^\top K(X, X)^{-1} f - \frac{1}{2} \ln \det(K(X, X)) + \text{constant}.
$$
Neural Architecture Search (NAS)

- Goal: automatically learn the best neural architecture

- Categorization

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


## NAS for Graph Machine Learning

- **Summary of NAS for graph ML**

<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 [2020]</td>
<td>✓</td>
<td>✓</td>
<td>Fixed</td>
<td>RNN controller + RL</td>
<td>-</td>
</tr>
<tr>
<td>AGNN [2019]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Self-designed controller + RL</td>
<td>-</td>
</tr>
<tr>
<td>SNAG [2020a]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>RNN controller + RL</td>
<td>Inherit weights</td>
</tr>
<tr>
<td>PDNAS [2020c]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>Single-path one-shot</td>
</tr>
<tr>
<td>POSE [2020]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>Single-path one-shot</td>
</tr>
<tr>
<td>NAS-GNN [2020]</td>
<td>x</td>
<td>x</td>
<td>Fixed</td>
<td>Evolutionary algorithm</td>
<td>-</td>
</tr>
<tr>
<td>AutoGraph [2020]</td>
<td>✓</td>
<td>x</td>
<td>Various</td>
<td>Evolutionary algorithm</td>
<td>-</td>
</tr>
<tr>
<td>GeneticGNN [2020b]</td>
<td>✓</td>
<td>✓</td>
<td>Fixed</td>
<td>Evolutionary algorithm</td>
<td>-</td>
</tr>
<tr>
<td>EGAN [2021a]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>One-shot</td>
</tr>
<tr>
<td>NAS-GCN [2020]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Evolutionary algorithm</td>
<td>-</td>
</tr>
<tr>
<td>LPGNAS [2020b]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>Single-path one-shot</td>
</tr>
<tr>
<td>You et al. [2020b]</td>
<td>✓</td>
<td>x</td>
<td>Various</td>
<td>Random search</td>
<td>-</td>
</tr>
<tr>
<td>SAGS [2020]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Self-designed algorithm</td>
<td>-</td>
</tr>
<tr>
<td>Peng et al. [2020]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>CEM-RL [2019]</td>
<td>-</td>
</tr>
<tr>
<td>GNAS [2021]</td>
<td>✓</td>
<td>x</td>
<td>Various</td>
<td>Differentiable</td>
<td>One-shot</td>
</tr>
<tr>
<td>AutoSTG [2021]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>One-shot+meta learning</td>
</tr>
<tr>
<td>DSS [2021]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>One-shot</td>
</tr>
<tr>
<td>SANE [2021b]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Differentiable</td>
<td>One-shot</td>
</tr>
<tr>
<td>AutoAttend [2021b]</td>
<td>✓</td>
<td>x</td>
<td>Fixed</td>
<td>Evolutionary algorithm</td>
<td>One-shot</td>
</tr>
</tbody>
</table>

**Table 1:** A summary of different NAS methods for graph machine learnings.
Graph NAS Search Space

- Message-passing framework of GNNs
  \[ m_i^{(l)} = \text{AGG}^{(l)} \left( \{ a_{ij}^{(l)} W^{(l)} h_i^{(l)}, \forall j \in \mathcal{N}(i) \} \right) \]
  \[ h_i^{(l+1)} = \sigma \left( \text{COMBINE}^{(l)} \left[ m_i^{(l)}, h_i^{(l)} \right] \right), \]
  - $h_i^{(l)}$: the representation of node $v_i$ in the $l^{th}$ layer
  - $m_i^{(l)}$: the received message of node $v_i$ in the $l^{th}$ layer

- Micro search space:
  - Aggregation function $\text{AGG}(\cdot)$: mean, max, sum, etc.
  - Combining function $\text{COMBINE}(\cdot)$: CONCAT, SUM, MLP, etc.
  - Aggregation weights $a_{ij}$ and attention heads
  - Non-linearity $\sigma(\cdot)$: Sigmoid, ReLU, tanh, etc.
  - Dimensionality

---

Graph Neural Architecture Search, *IJCAI* 2020.
Graph NAS Search Space

- Macro search space: how to arrange different layers
  - Residual connection, dense connection, etc.

Formulation:

$$H^{(l)} = \sum_{j < l} F_{jl} \left( H^{(j)} \right)$$

- $F_{jl}$: connectivity pattern from $j^{th}$ to the $l^{th}$ layer
  - ZERO (not connecting), IDENTITY (residual connection), MLP, etc.

DeepGCNs: Can GCNs Go as Deep as CNNs? ICCV 2019
Graph Neural Architecture Search, IJCAI 2020.
Graph NAS Search Space

- **Other search spaces**
  - **Pooling methods:**
    - Aggregate node-level representation into graph-level representation
  - **Hyper-parameters:** similar to HPO for graphs
    - Number of layers, number of epochs, optimizer, dropout rate, etc.
  - **Spaces for specific tasks:**
    - E.g., spatial-temporal graph operators

![Graph NAS Search Space Diagram](image)
Graph NAS Search Strategy

- Most previous general NAS search strategies can be directly applied
  - Controller (e.g., RNN) + Reinforcement learning (RL)
  - Evolutionary
  - Differentiable

- Controller samples architecture (e.g., as a sequence)
- RL feedback rewards (e.g., validation performance) to update the controller
Graph NAS Search Strategy

- Most previous general NAS search strategies can be directly applied
  - Controller (e.g., RNN) + Reinforcement learning (RL)
  - Evolutionary
  - Differentiable

- Need to define how to sample parents, generate offspring, and update populations
- E.g., remove the worst individual (Real, et al., 2017), remove the oldest individual (Real, et al., 2018), or no remove (Liu, et al., 2018)
Graph NAS Search Strategy

- Most previous general NAS search strategies can be directly applied
  - Controller (e.g., RNN) + Reinforcement learning (RL)
  - Evolutionary
  - Differentiable

- Generate a super-network to combine operations of the search space
- Continuous relaxation to make the model differentiable

DARTS: Differentiable Architecture Search, ICLR 2019
Graph NAS Performance Estimation

- Low-fidelity training
  - Reduce number of epochs
  - Reduce training data: sample subgraphs as in HPO

- Inheriting weights
  - Challenge: parameters in graph ML (e.g., GNNs) are unlike other NNs
  - E.g., constraints by AGNN (Zhou et al., 2019)
    - Same weight shapes
    - Same attention and activation functions

- Weight sharing in differentiable NAS with one-shot model
AutoML library on Graph

- Graph related
  - PyTorch geometric
  - DGL Deep Graph Library
  - Paddle Graph Learning
  - PyTorch BigGraph

- AutoML related
  - AutoKeras
  - H2O AutoML
  - TPOT

Gap!
Introduction – AutoGL

- We design the world’s first 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
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
Feature Engineering

Node-level Graph Generators
- Graphlet, EigenGNN, Pagerank, Onehot, ...

Selector
- Pyg.transform

Netlsd

NetworkX

Auto Feature Engineering

Neural Architecture Search

Hyper-Parameter Optimization

Model Training

Auto GL Solver

Auto Ensemble
Neural Architecture Search

Data → AutoGL Dataset → Auto Feature Engineering → Neural Architecture Search → Hyper-Parameter Optimization → Model Training → Auto Ensemble

Neural Architecture Search

Algorithms: Random, One-Shot, RL

Search Space: GraphNAS, Single Path

Methods: ENAS, Darts, Vanilla RL, GraphNAS, Macro, Micro
Hyper-Parameter Optimization

- AutoGL Dataset
- Auto Feature Engineering
- Neural Architecture Search
- Hyper-Parameter Optimization
- Model Training
- AutoGL Solver
- Auto Ensemble

Hyper-Parameter Optimization

- General-Purpose: Random, Bayes, Grid, CAMES, Anneal, TPE
- Graph Aware: AutoNE
Model Training

**Trainer**
- Learning rate
- Epochs
- Optimizer
- Loss
- Early Stopping

**Model**
- Forward
- Ops & Architectures
- Dropout & Hidden

Currently supported models
- Node classification
  - GCN
  - GAT
  - GraphSAGE
- Link Prediction
- Graph classification
  - TopKPool
  - GIN
Ensemble

Data -> AutoGL Dataset -> Auto Feature Engineering -> Neural Architecture Search -> Hyper-Parameter Optimization -> Model Training -> Auto Ensemble

- Voting
- Stacking

Meta-learner
Example Results

**Table 1: The results of node classification**

<table>
<thead>
<tr>
<th>Model</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>80.9 ± 0.7</td>
<td>70.9 ± 0.7</td>
<td>78.7 ± 0.6</td>
</tr>
<tr>
<td>GAT</td>
<td>82.3 ± 0.7</td>
<td>71.9 ± 0.6</td>
<td>77.9 ± 0.4</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>74.5 ± 1.8</td>
<td>67.2 ± 0.9</td>
<td>76.8 ± 0.6</td>
</tr>
<tr>
<td>AutoGL</td>
<td><strong>83.2 ± 0.6</strong></td>
<td><strong>72.4 ± 0.6</strong></td>
<td><strong>79.3 ± 0.4</strong></td>
</tr>
</tbody>
</table>

**Table 2: The results of graph classification**

<table>
<thead>
<tr>
<th>Model</th>
<th>MUTAG</th>
<th>PROTEINS</th>
<th>IMDB-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-K Pooling</td>
<td>80.8 ± 7.1</td>
<td>69.5 ± 4.4</td>
<td>71.0 ± 5.5</td>
</tr>
<tr>
<td>GIN</td>
<td>82.7 ± 6.9</td>
<td>66.5 ± 3.9</td>
<td>69.1 ± 3.7</td>
</tr>
<tr>
<td>AutoGL</td>
<td><strong>87.6 ± 6.0</strong></td>
<td><strong>73.3 ± 4.4</strong></td>
<td><strong>72.1 ± 5.0</strong></td>
</tr>
</tbody>
</table>

**Table 3: The results of different HPO methods for node classification**

<table>
<thead>
<tr>
<th>Method</th>
<th>Trials</th>
<th>GCN Cora</th>
<th>GAT Cora</th>
<th>GCN CiteSeer</th>
<th>GAT CiteSeer</th>
<th>GCN PubMed</th>
<th>GAT PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>80.9 ± 0.7</td>
<td>82.3 ± 0.7</td>
<td>70.9 ± 0.7</td>
<td>71.9 ± 0.6</td>
<td>78.7 ± 0.6</td>
<td>77.9 ± 0.4</td>
</tr>
<tr>
<td>random</td>
<td>1</td>
<td>81.0 ± 0.6</td>
<td>81.4 ± 1.1</td>
<td>70.4 ± 0.7</td>
<td>70.1 ± 1.1</td>
<td>78.3 ± 0.8</td>
<td>76.9 ± 0.8</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>82.0 ± 0.6</td>
<td>82.5 ± 0.7</td>
<td>71.5 ± 0.6</td>
<td><strong>72.2 ± 0.7</strong></td>
<td>79.1 ± 0.3</td>
<td>78.2 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>81.8 ± 1.1</td>
<td><strong>83.2 ± 0.7</strong></td>
<td>71.1 ± 1.0</td>
<td>72.1 ± 1.0</td>
<td><strong>79.2 ± 0.4</strong></td>
<td>78.2 ± 0.4</td>
</tr>
<tr>
<td>TPE</td>
<td>1</td>
<td>81.8 ± 0.6</td>
<td>81.9 ± 1.0</td>
<td>70.1 ± 1.2</td>
<td>71.0 ± 1.2</td>
<td>78.7 ± 0.6</td>
<td>77.7 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>82.0 ± 0.7</td>
<td>82.3 ± 1.2</td>
<td>71.2 ± 0.6</td>
<td>72.1 ± 0.7</td>
<td>79.0 ± 0.4</td>
<td>78.3 ± 0.4</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td><strong>82.1 ± 1.0</strong></td>
<td>83.2 ± 0.8</td>
<td><strong>72.4 ± 0.6</strong></td>
<td>71.6 ± 0.8</td>
<td>79.1 ± 0.6</td>
<td>78.1 ± 0.4</td>
</tr>
</tbody>
</table>
AutoGL Plans

Incoming new features:

- DGL backend
- More large-scale graph support
  - E.g., sampling, distributed, etc.
- More graph tasks
  - E.g., heterogenous graphs, spatial-temporal graphs, etc.

Warmly welcome all feedbacks and suggestions!

Contact: autogl@tsinghua.edu.cn
Overview of Our Representative Works

Our roadmap for automated machine learning on graphs

AutoGL Tool and Library

**AutoGL HPO**
- **AutoNE**
  - Scalability
- **e-AutoGR**
  - Scalability + Explainability

**AutoGL NAS**
- **AutoAttend**
  - Attention
- **GASSO**
  - Graph Structure

**AutoNE, AutoAttend, GASSO**
- Signature Extraction
- Meta-learner
- Hand-crafted attention representation
- Automated attention representation
- Explainable Graph Features
- Run e-AutoGR with hyper-parameters to obtain the attention graph structure
Summary and Future Directions

- Machine Learning on Graphs
- Automate Graph Machine Learning
  - Graph HPO
  - Graph NAS
- AutoGL Platform

Open Problems:
- Graph models for AutoML
  - E.g., regard each NN as a Directed Acyclic Graph (DAG)
  - E.g., using GNNs as surrogate models in model performance prediction
- Robustness and explainability
- Hardware-aware models
- Comprehensive evaluation protocols
Thanks!

Ziwei Zhang,  Tsinghua University
zwzhang@tsinghua.edu.cn
https://zw-zhang.github.io/