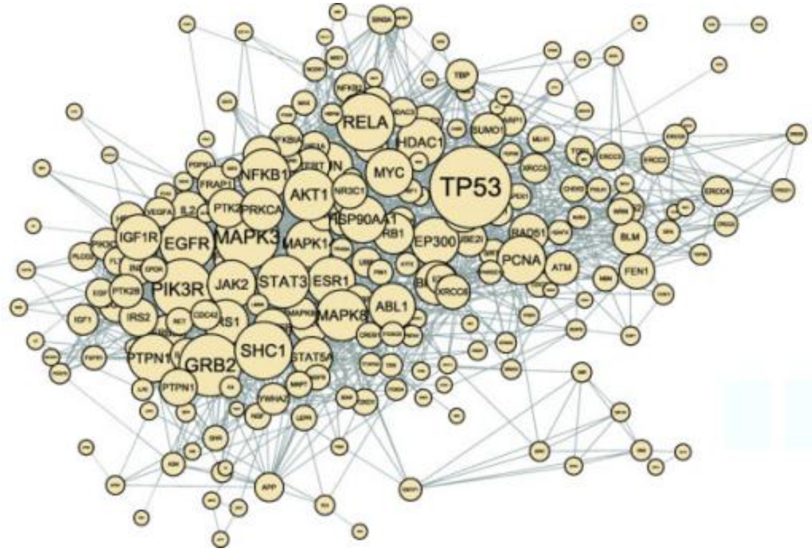
Automated Machine Learning on Graphs: A Survey

Ziwei Zhang, Xin Wang, Wenwu Zhu

Tsinghua University

Graphs are Ubiquitous

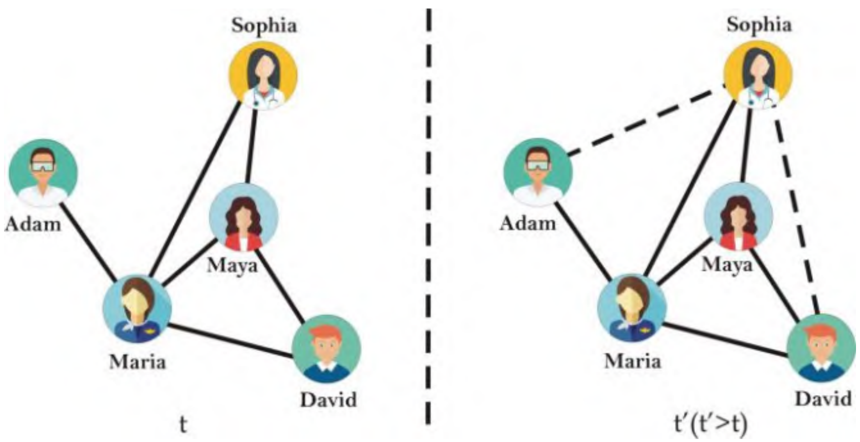


Biology Network

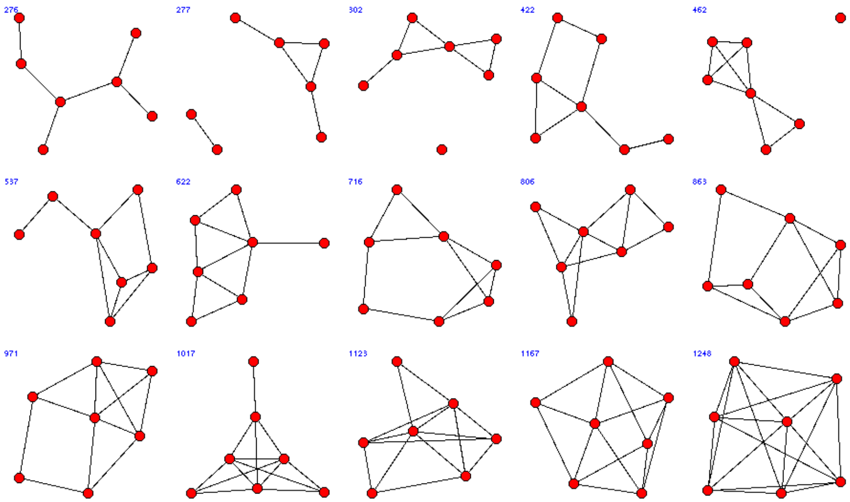
Social Network

Traffic Network

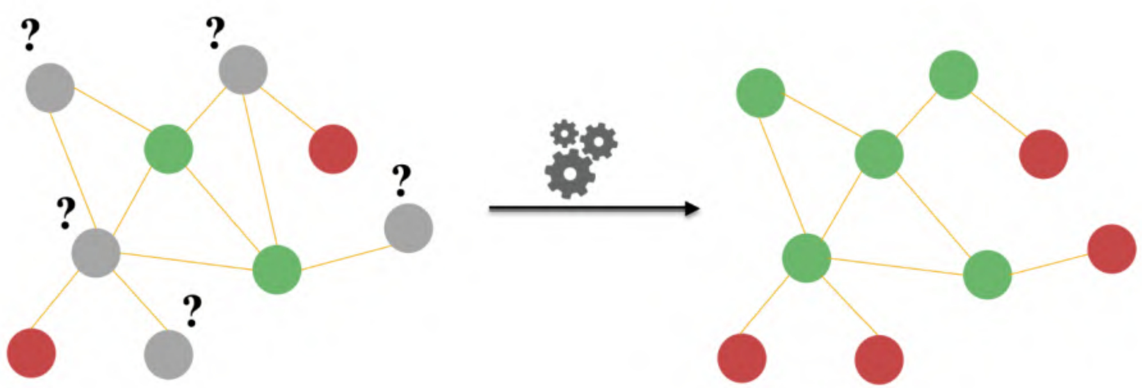
Graph Tasks



Link Prediction



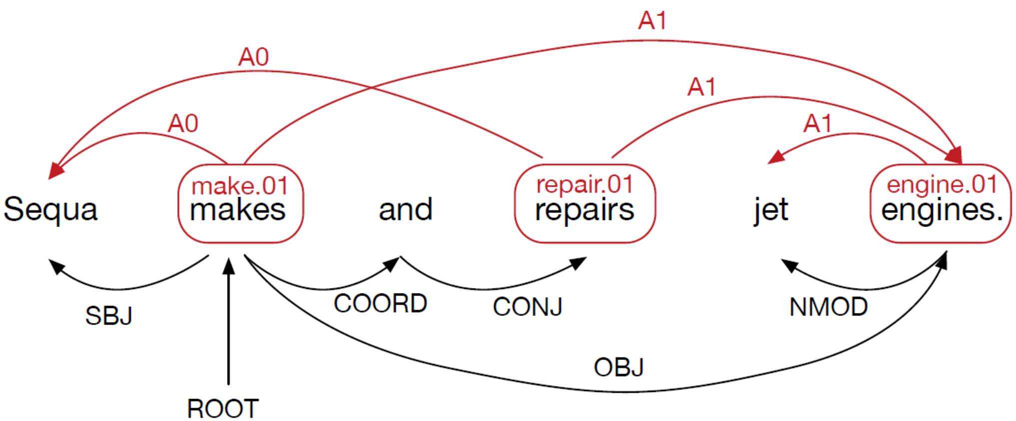
Graph Classification



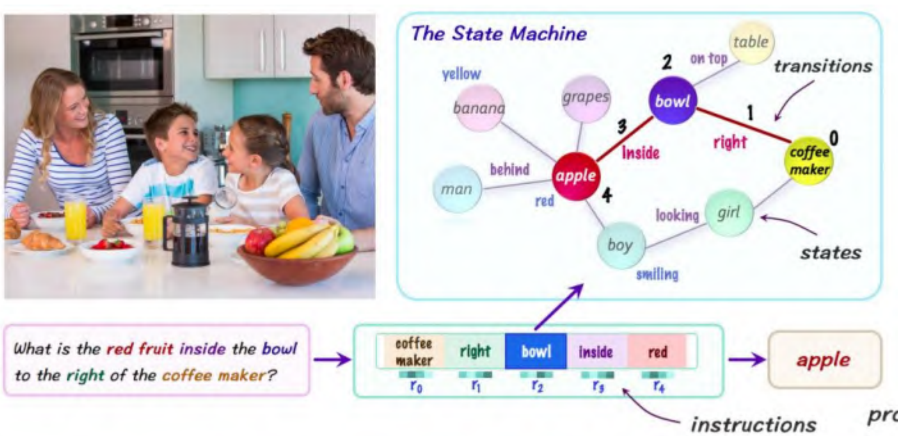
Node Classification

Images are from search engines

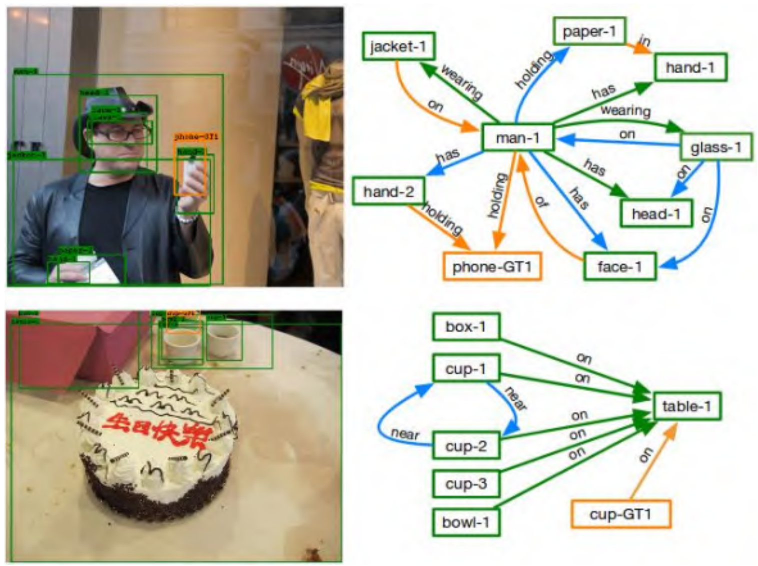
Graph Applications



Natural Language Processing



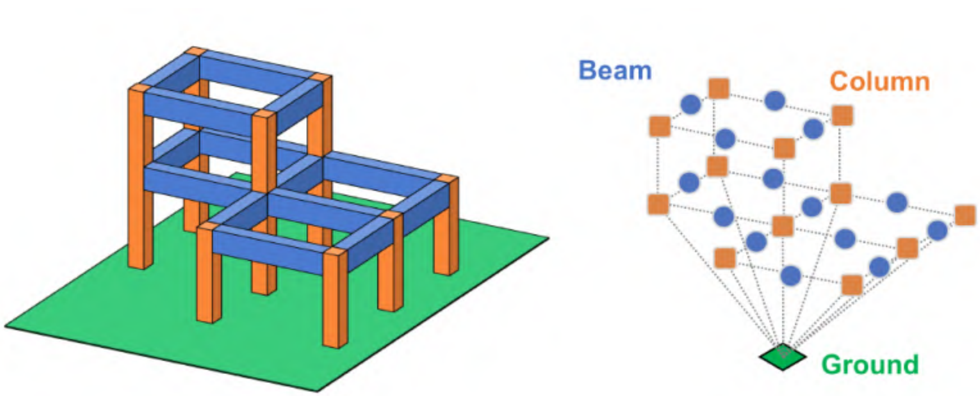
Reasoning



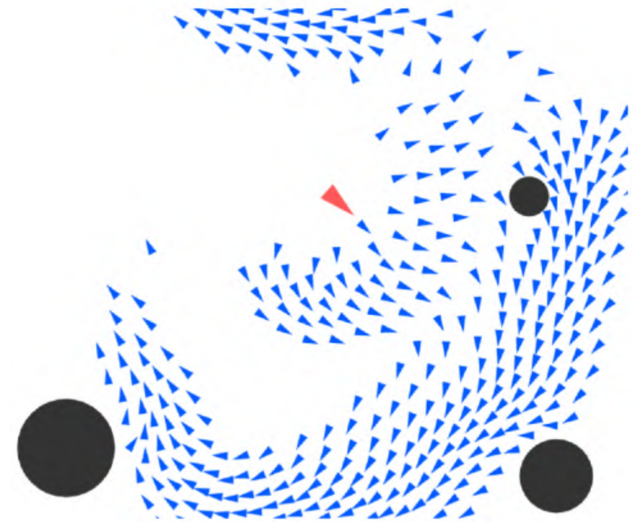
Computer Vision

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*

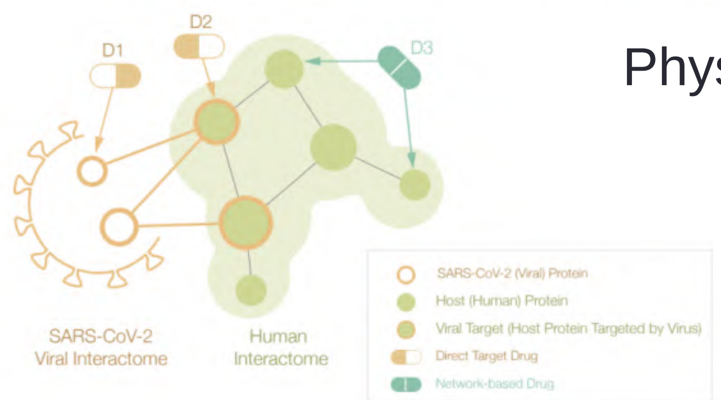
Graph Applications



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*

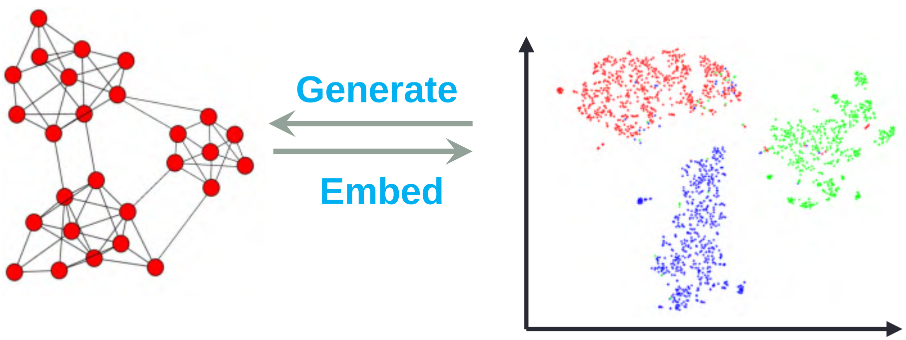
Graph in Industry

- 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

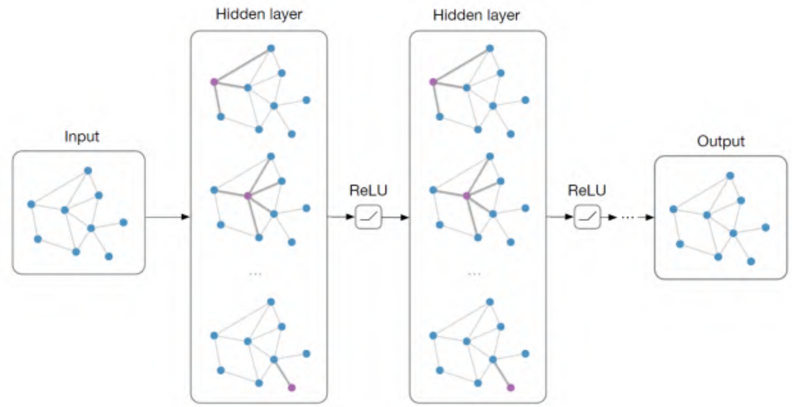
  ……

Machine learning on graphs has important and diverse applications!

Machine Learning on Graphs



Network Embedding



Graph Neural Networks

IEEE TRANSACTIONS ON
KNOWLEDGE AND DATA ENGINEERING

A Survey on Network Embedding

Issue No. 01 - (preprint vol.)
ISSN: 1041-4347
pp: 1
DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/TKDE.2018.2849727

Peng Cui , Computer Science Department, Tsinghua University, Beijing, Beijing China (e-mail: cuip@tsinghua.edu.cn)
Xiao Wang , Computer Science, Tsinghua University, Beijing, Beijing China (e-mail: wangxiao007@mail.tsinghua.edu.cn)
Jian Pei , School of Computing Science, Simon Fraser University, Burnaby, British Columbia Canada (e-mail: jpei@cs.sfu.ca)
Wenwu Zhu , Department of Computer Science, Tsinghua University, Beijing, Beijing China (e-mail: wwzhu@tsinghua.edu.cn)

ABSTRACT
Network embedding assigns nodes in a network to low-dimensional representations and effectively preserves the network structure. Recently, a significant amount of progresses have been made toward this emerging network analysis paradigm. In this survey, we focus on categorizing and then reviewing the current development on network embedding methods, and point out its future research directions. We first summarize the motivation of network embedding. We discuss the classical graph embedding algorithms and their relationship with network embedding. Afterwards and primarily, we provide a comprehensive overview of a large number of network embedding methods in a systematic manner, covering the structure- and property-preserving network embedding methods, the network embedding methods with side information and the advanced information preserving network embedding methods. Moreover, several evaluation approaches for network embedding and some useful online resources, including the network data sets and softwares, are reviewed, too. Finally, we discuss the framework of exploiting these network embedding methods to build an effective system and point out some potential future directions.

Journals & Magazines > IEEE Transactions on Knowledg... > Early Access

Deep Learning on Graphs: A Survey

Publisher: IEEE | Cite This | PDF

Ziwei Zhang ; Peng Cui ; Wenwu Zhu All Authors

3 Paper Citations | 1508 Full Text Views

Abstract
Authors
Citations
Keywords
Metrics
Media

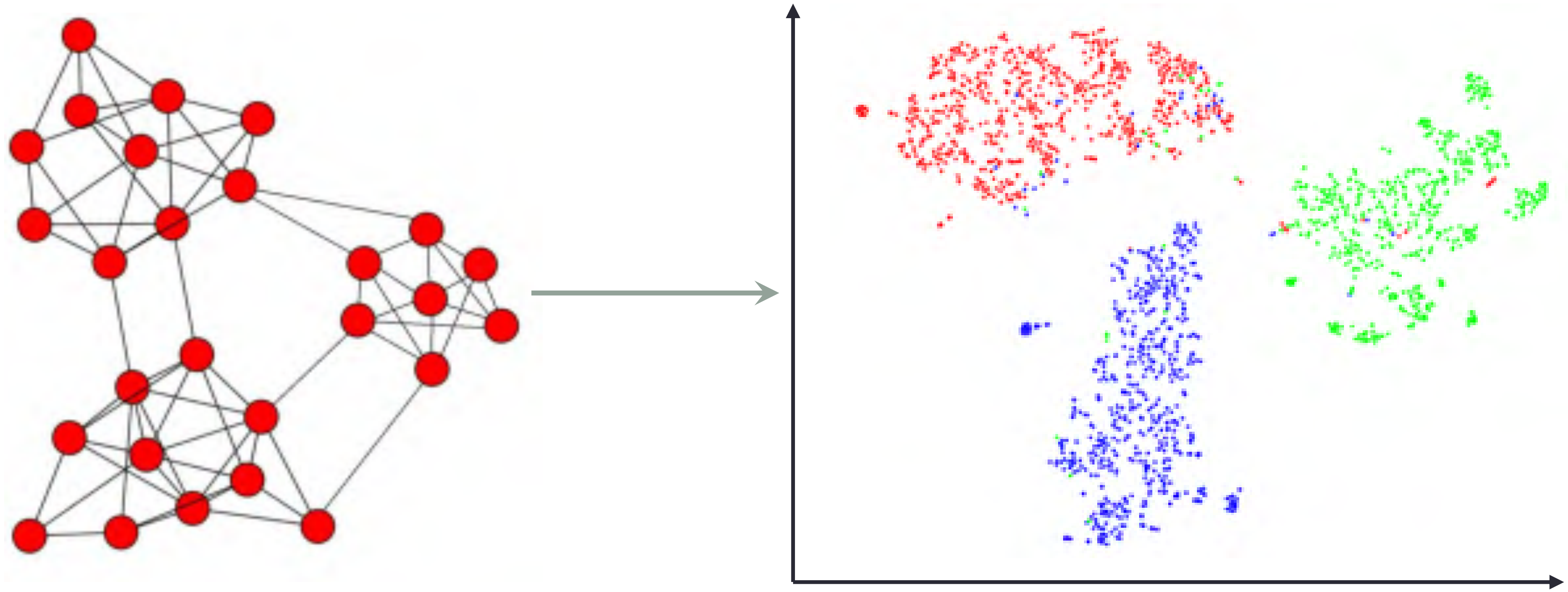
Abstract:
Deep learning has been shown to be successful in a number of domains, ranging from acoustics, images, to natural language processing. However, applying deep learning to the ubiquitous graph data is non-trivial because of the unique characteristics of graphs. Recently, substantial research efforts have been devoted to applying deep learning methods to graphs, resulting in beneficial advances in graph analysis techniques. In this survey, we comprehensively review the different types of deep learning methods on graphs. We divide the existing methods into five categories based on their model architectures and training strategies: graph recurrent neural networks, graph convolutional networks, graph autoencoders, graph reinforcement learning, and graph adversarial methods. We then provide a comprehensive overview of these methods in a systematic manner mainly by following their development history. We also analyze the differences and compositions of different methods. Finally, we briefly outline the applications in which they have been used and discuss potential future research directions.

Published in: IEEE Transactions on Knowledge and Data Engineering (Early Access)

Peng Cui, Xiao Wang, Jian Pei, Wenwu Zhu. A Survey on Network Embedding. *IEEE TKDE, 2018.*

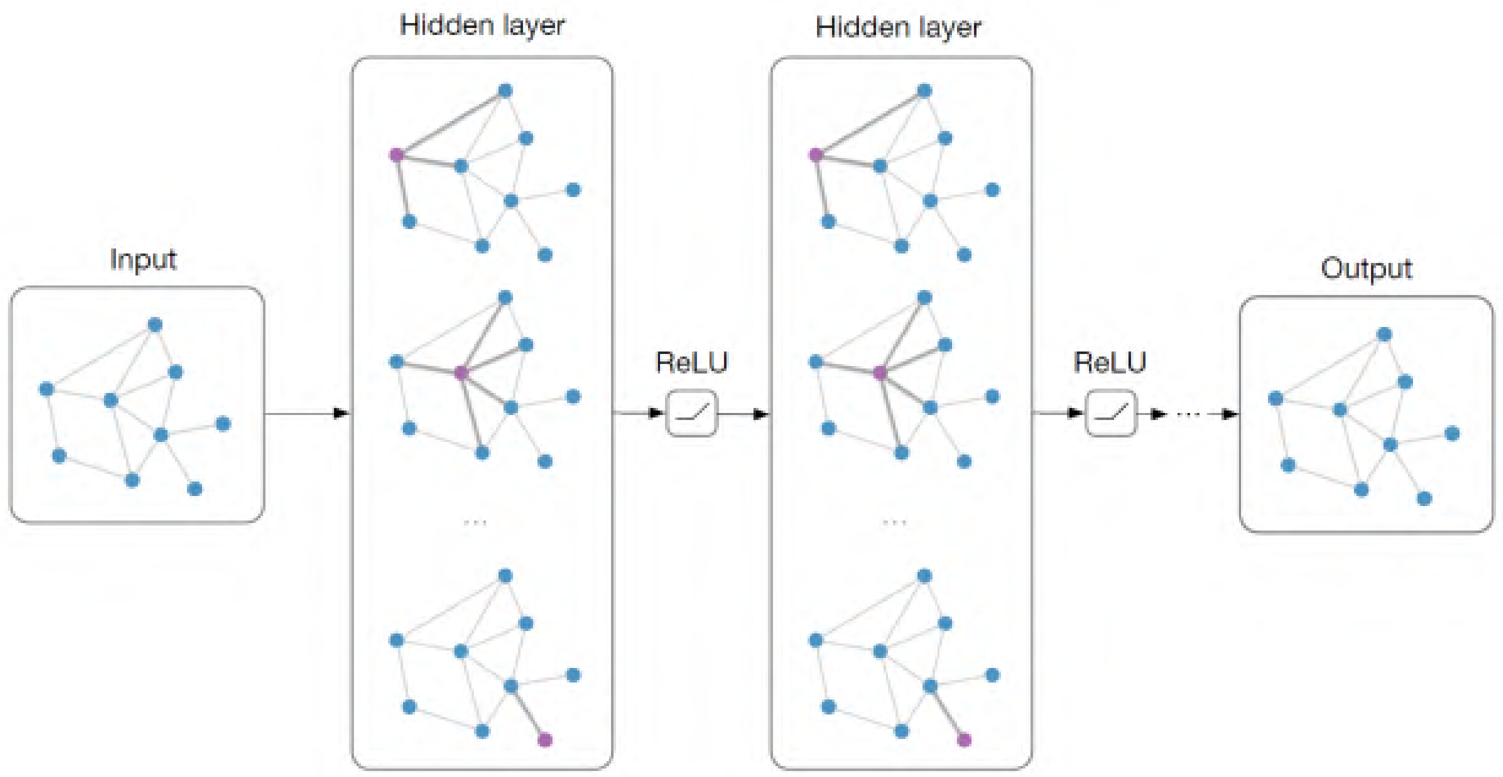
Ziwei Zhang, Peng Cui, Wenwu Zhu. Deep Learning on Graphs: A Survey. *IEEE TKDE, 2020.*

Network Embedding



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

Graph Neural Network

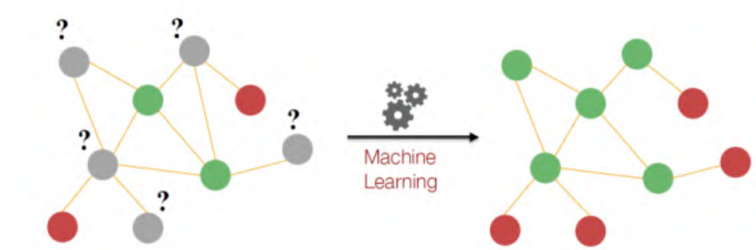


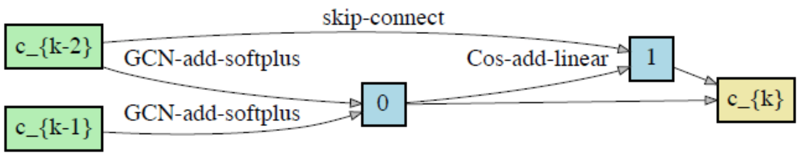
- Design neural networks directly applicable for graphs for end-to-end learning
- Message-passing framework: nodes exchange messages along structures

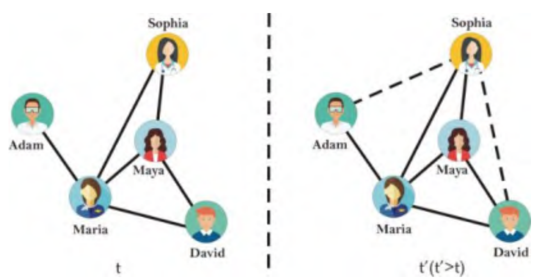
Semi-supervised classification with graph convolutional networks, ICLR 2017

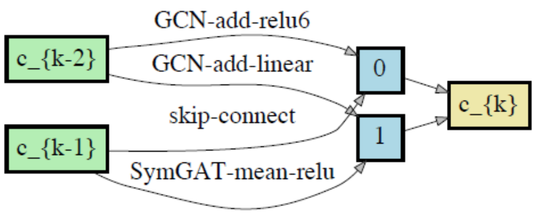
Problems in Existing Graph Learning Methods

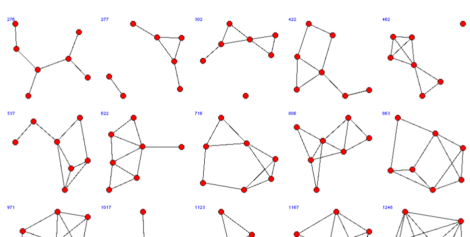
- Manually design architectures and hyper-parameters through trial-and-error
- Each task needs be handled separately

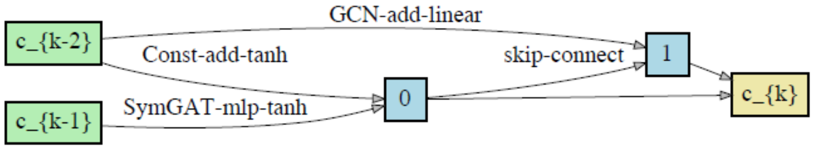






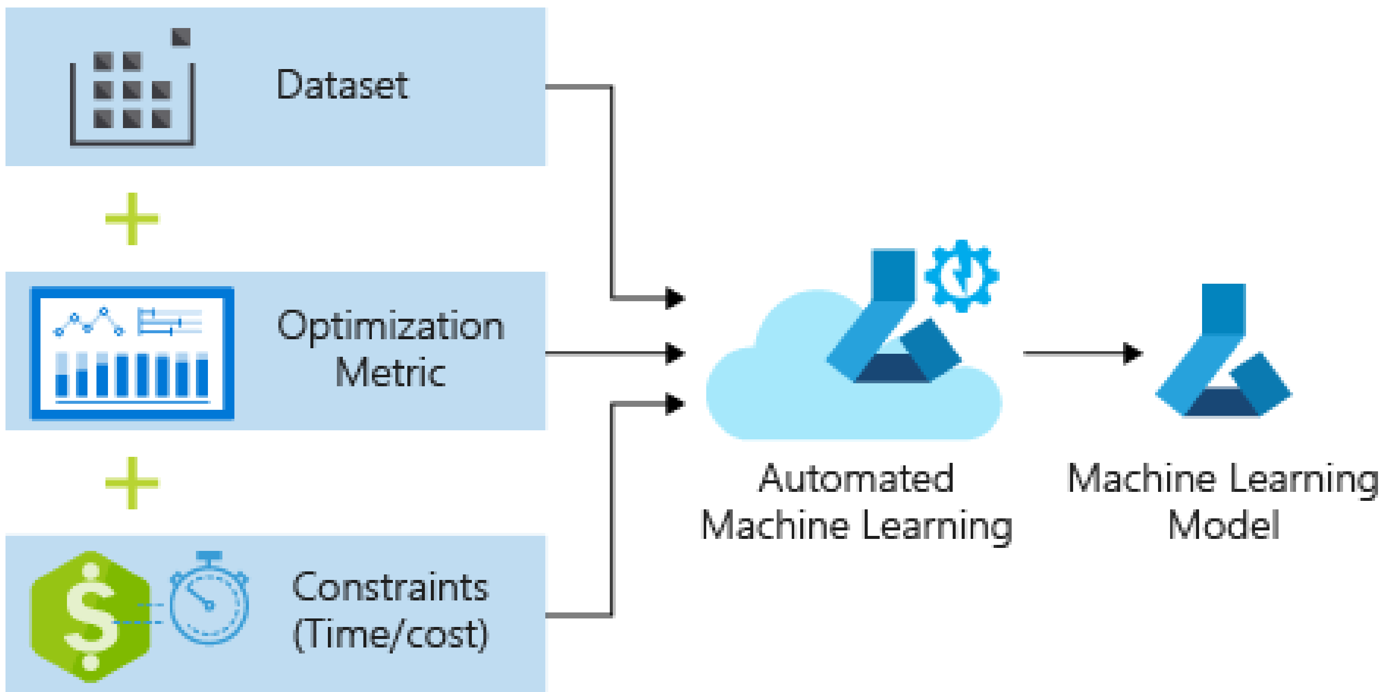






Automated graph machine learning is critically needed!

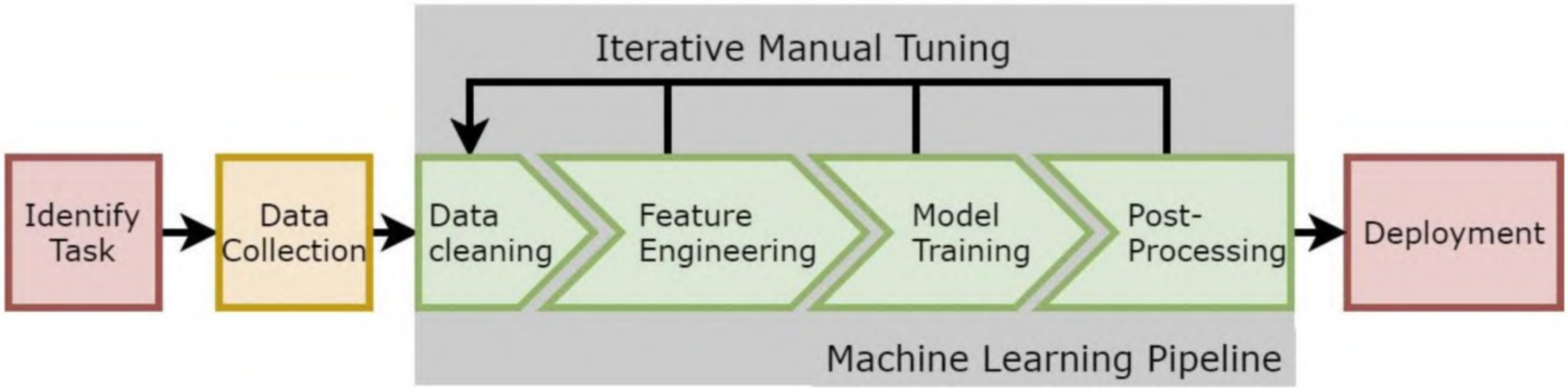
A Glance of AutoML



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
- **Low** tuning **efficiency**
- **Limited** by human design

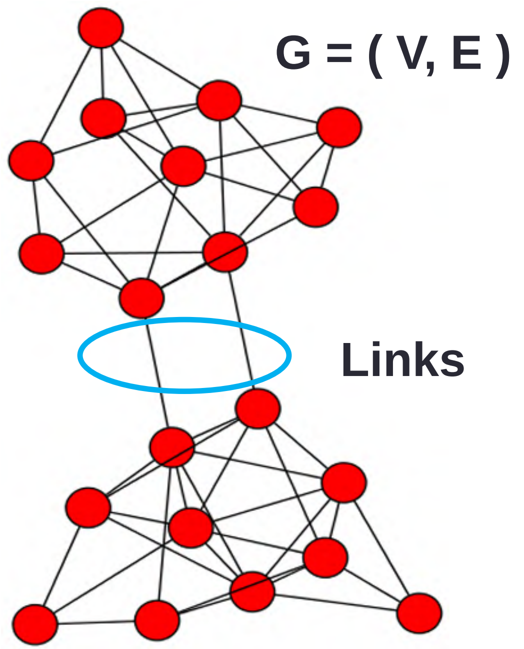
- **Free human** out of the loop
- **High** optimization **efficiency**
- **Discover & extract** patterns and combinations **automatically**

Automated Graph Learning

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

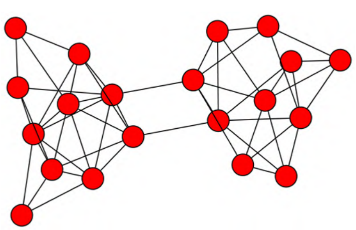
- The key: *Graph Structure!*

Various diverse graph structures may place complex impacts on graph HPO and graph NAS



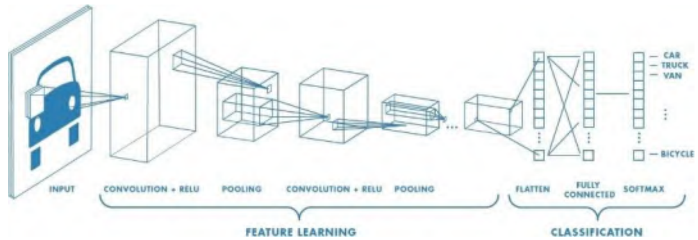
Challenge: Uniqueness of graph ML

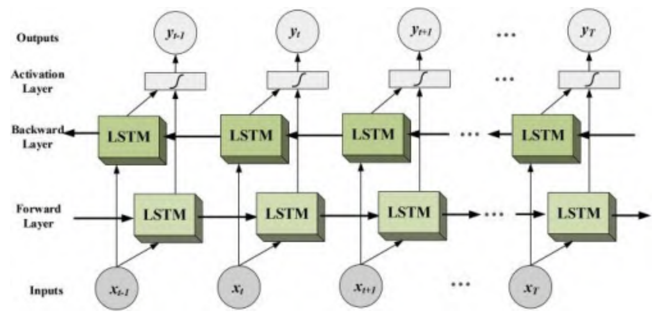
Data

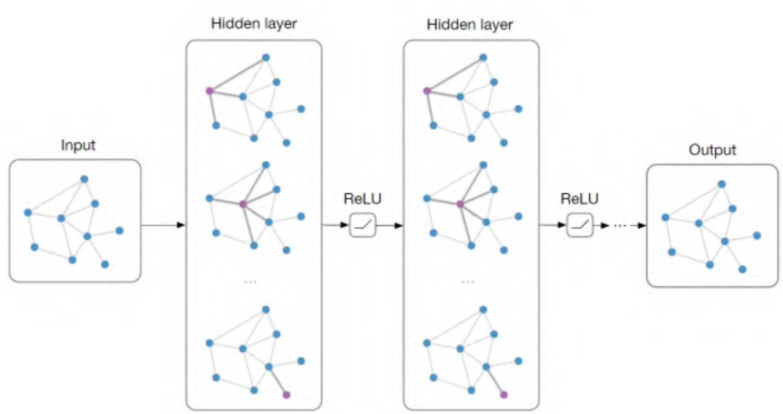


G = (V, E)

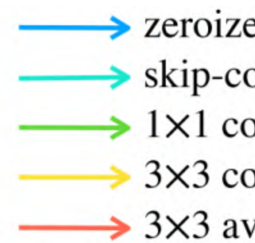
NN architecture







Search Space



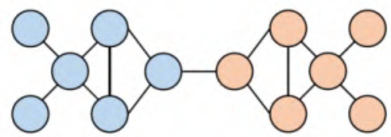
predefined operation set

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

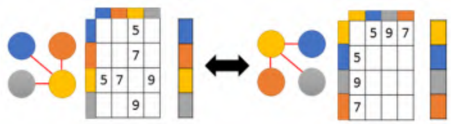
?

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*

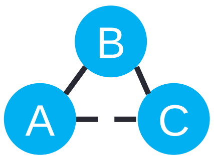
Challenge: Complexity and diversity of graph tasks



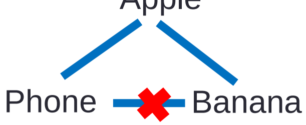
High-order Proximity



Permutation-equivariance



Transitivity



Non-transitivity

Various graph properties

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

Various applications

Various domains

- No single method can perfectly handle all scenarios

Challenge: Scalability

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)

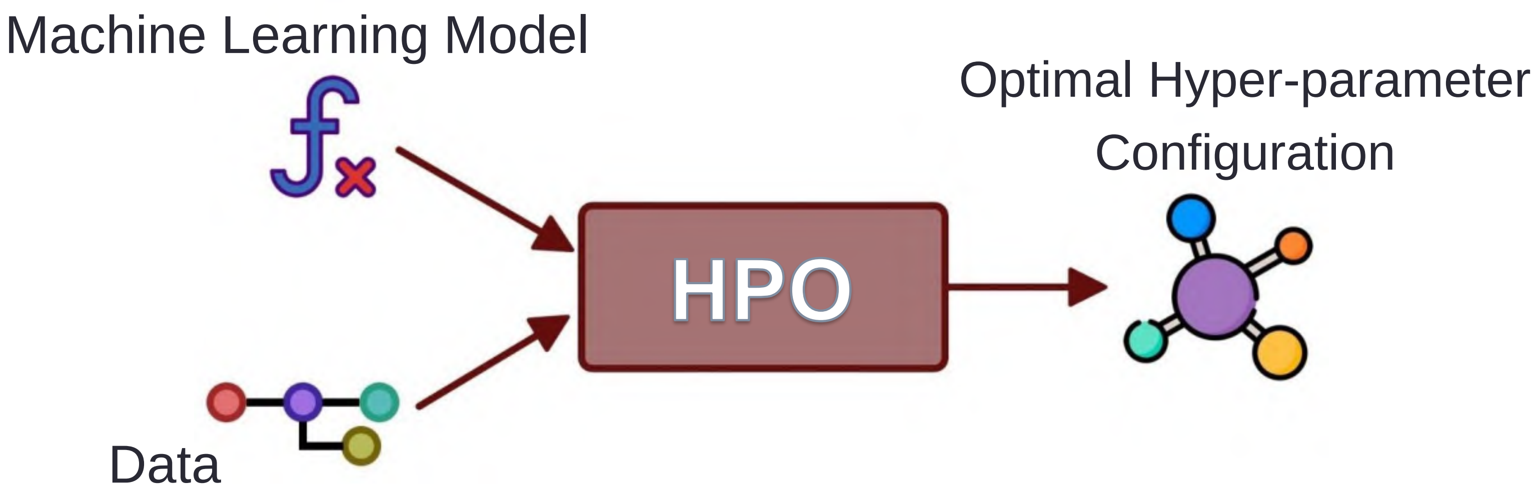
Citation Networks

- 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

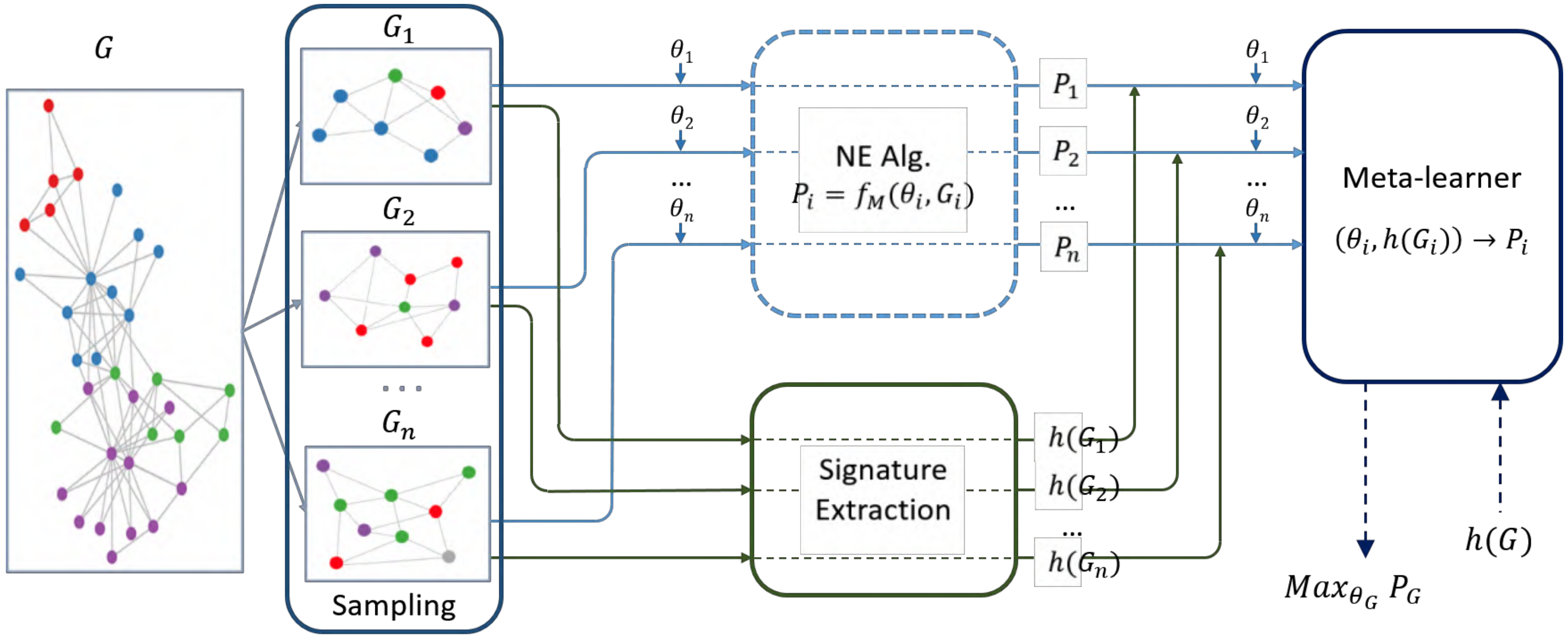


- Formulation: bi-level optimization

$$\min_{\alpha \in \mathcal{A}} \mathcal{L}_{val}\left(\mathbf{W}^*(\alpha), \alpha\right)$$
$$\text{s.t.} \quad \mathbf{W}^*(\alpha) = \arg\min_{\mathbf{W}} \left(\mathcal{L}_{train}\left(\mathbf{W}, \alpha\right)\right)$$

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

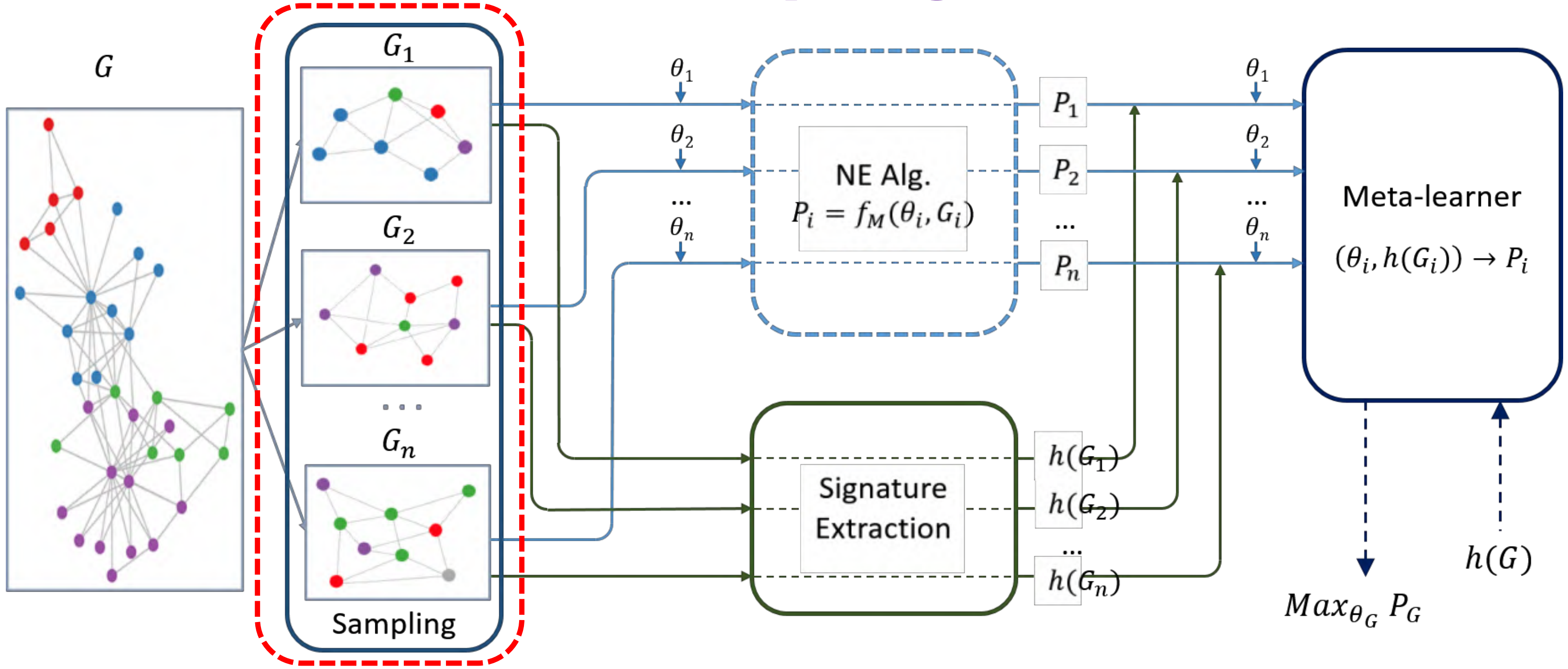
AutoNE: Framework



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

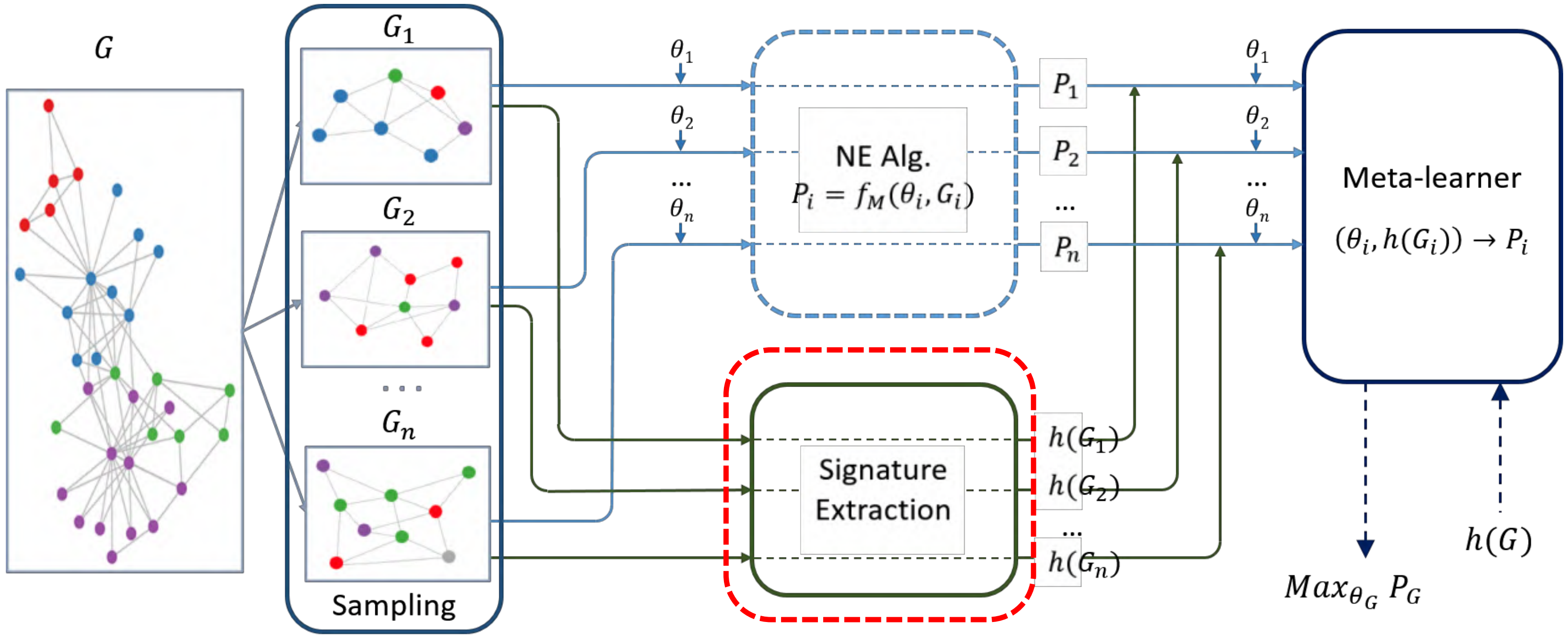
Tu Ke, Jianxin Ma, Peng Cui, Jian Pei, and Wenwu Zhu.
AutoNE: Hyperparameter optimization for massive network embedding. ***KDD 2019***.

AutoNE: Sampling Module



- **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

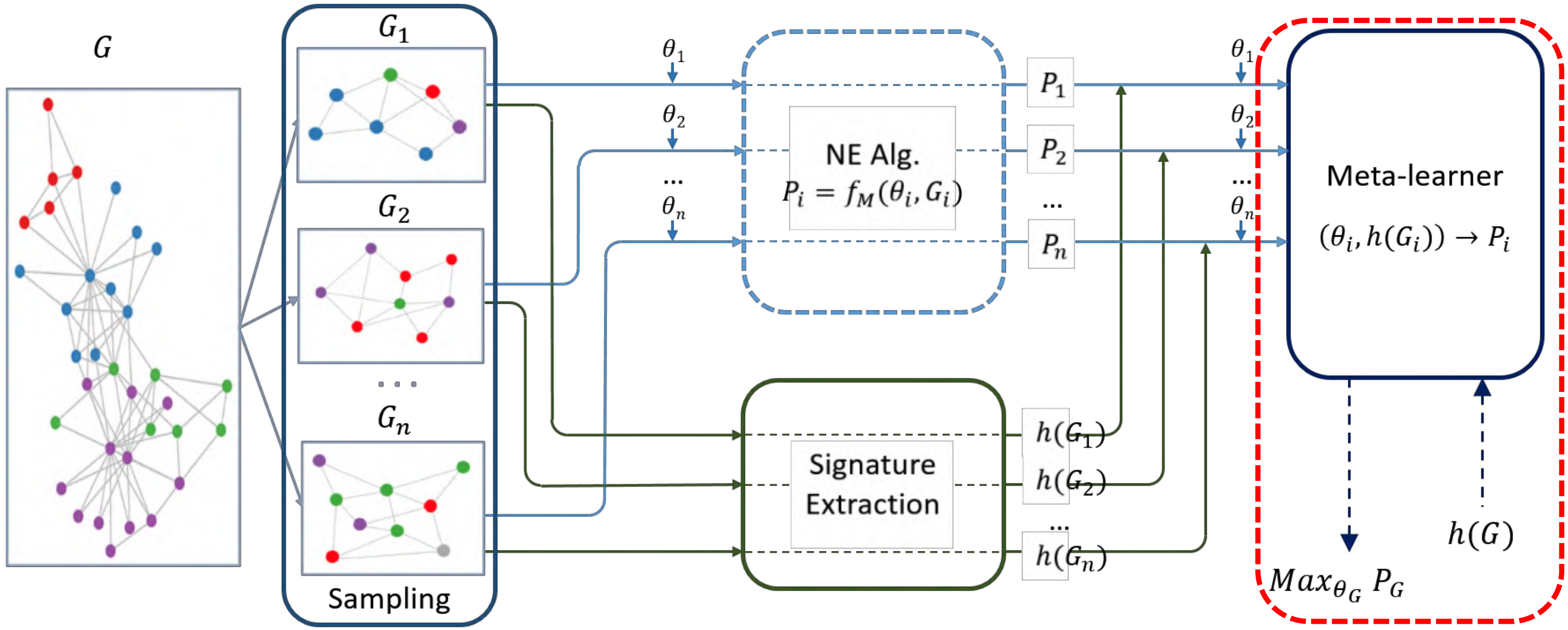
AutoNE: Signature Extraction Module



- **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

$$h_t(G) = tr(H_t) = tr(e^{-tL}) = \sum_j e^{-t\lambda_j}$$

AutoNE: Meta-Learning Module

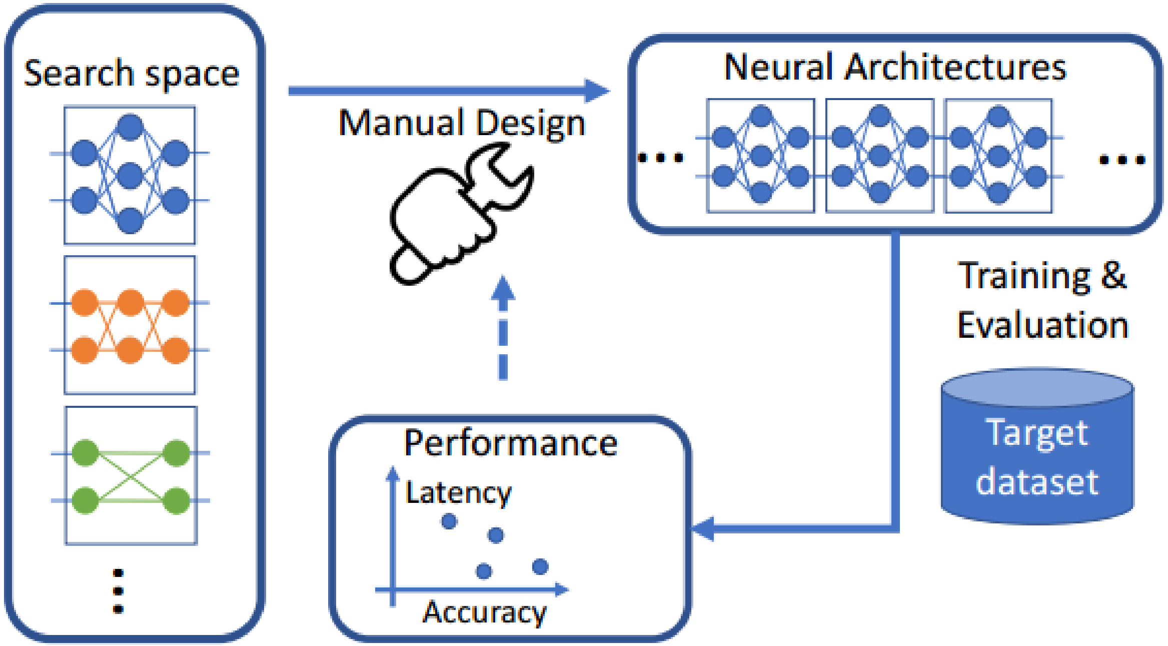


- **Goal**: transfer knowledge about hyper-parameters in sampled subgraphs to the original large-scale graph
- **Assumption**: two similar graphs have similar optimal hyper-parameter
- **Method**: Gaussian Process based meta-learner

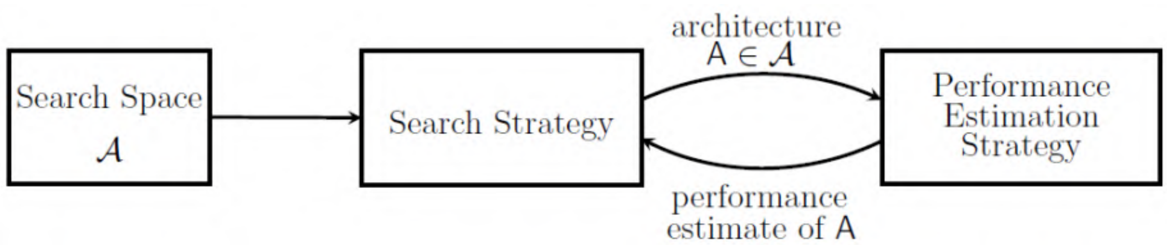
$$\ln p(\mathrm{f} \mid \mathrm{X}) = -\frac{1}{2}\mathrm{f}^\top K(\mathrm{X},\mathrm{X})^{-1}\mathrm{f} - \frac{1}{2}\ln\det(K(\mathrm{X},\mathrm{X})) + constant.$$

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

Method	Search space: Micro	Macro	Pooling	HP	Layers	Tasks: Node	Graph	Search Strategy	Performance Estimation	Other Characteristics
GraphNAS [2020]	✓	✓	✗	✗	Fixed	✓	✗	RNN controller + RL	-	-
AGNN [2019]	✓	✗	✗	✗	Fixed	✓	✗	Self-designed controller + RL	Inherit weights	-
SNAG [2020a]	✓	✓	✗	✗	Fixed	✓	✗	RNN controller + RL	Inherit weights	Simplify the micro search space
PDNAS [2020c]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	Single-path one-shot	-
POSE [2020]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	Single-path one-shot	Support heterogenous graphs
NAS-GNN [2020]	✓	✗	✗	✓	Fixed	✓	✗	Evolutionary algorithm	-	-
AutoGraph [2020]	✓	✓	✗	✗	Various	✓	✗	Evolutionary algorithm	-	-
GeneticGNN [2020b]	✓	✗	✗	✓	Fixed	✓	✗	Evolutionary algorithm	-	-
EGAN [2021a]	✓	✓	✗	✗	Fixed	✓	✓	Differentiable	One-shot	Sample small graphs for efficiency
NAS-GCN [2020]	✓	✓	✓	✗	Fixed	✗	✓	Evolutionary algorithm	-	Handle edge features
LPGNAS [2020b]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	Single-path one-shot	Search for quantisation options
You *et al.* [2020b]	✓	✓	✗	✓	Various	✓	✓	Random search	-	Transfer across datasets and tasks
SAGS [2020]	✓	✗	✗	✗	Fixed	✓	✓	Self-designed algorithm	-	-
Peng *et al.* [2020]	✓	✗	✗	✗	Fixed	✗	✓	CEM-RL [2019]	-	Search spatial-temporal modules
GNAS[2021]	✓	✓	✗	✗	Various	✓	✓	Differentiable	One-shot	-
AutoSTG[2021]	✗	✓	✗	✗	Fixed	✓	✗	Differentiable	One-shot+meta learning	Search spatial-temporal modules
DSS[2021]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	One-shot	Dynamically update search space
SANE[2021b]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable	One-shot	-
AutoAttend[2021b]	✓	✓	✗	✗	Fixed	✓	✓	Evolutionary algorithm	One-shot	Cross-layer attention

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

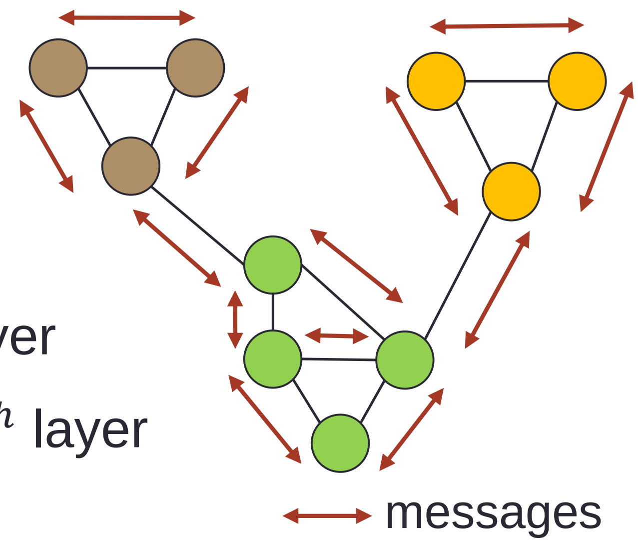
Graph NAS Search Space

- Message-passing framework of GNNs

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

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

  - $\mathrm{h}_i^{(l)}$: the representation of node v_i in the l^{th} layer
  - $\mathrm{m}_i^{(l)}$: the received message of node v_i in the l^{th} layer



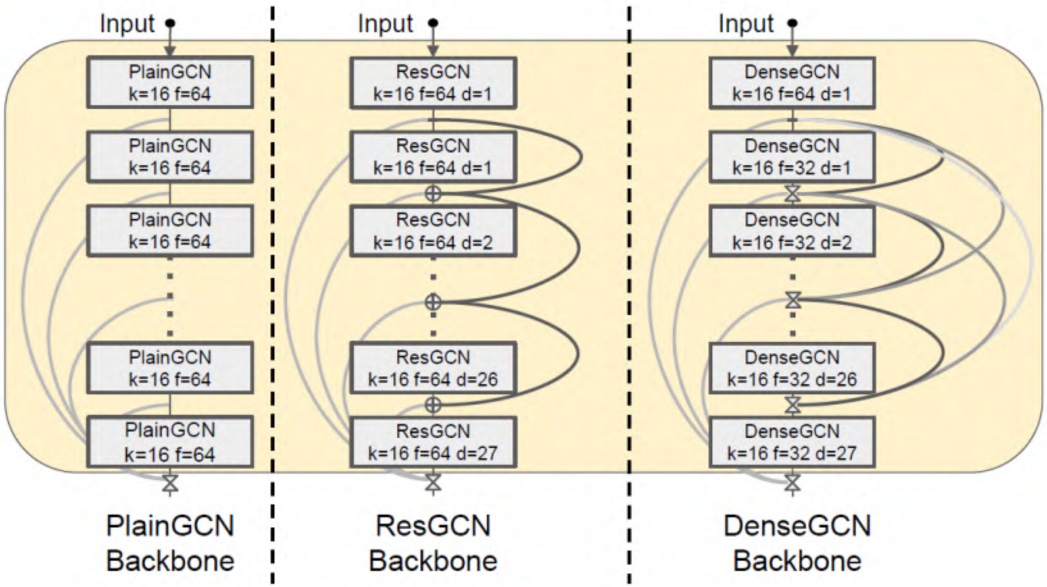
- Micro search space:
  - Aggregation function $\mathrm{AGG}(\cdot)$: mean, max, sum, etc.
  - Combining function $\mathrm{COMBINE}(\cdot)$: CONCAT, SUM, MLP, etc.
  - Aggregation weights a_{ij} and attention heads
  - Non-linearity $\sigma(\cdot)$: Sigmoid, ReLU, tanh, etc.
  - Dimensionality

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	$a_{ij}^{\cos} = \cos\left(\mathbf{W}_a\mathbf{h}_i, \mathbf{W}_a\mathbf{h}_j\right)$
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	$a_{ij}^{\text{gene}} = \tanh\left(\mathrm{sum}\left(\mathbf{W}_a\mathbf{h}_i + \mathbf{W}_a\mathbf{h}_j\right)\right)\mathbf{W}_a'$

Neural message passing for quantum chemistry. *ICML, 2017.*
Graph Neural Architecture Search, *IJCAI 2020.*

Graph NAS Search Space

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



  - Formulation:

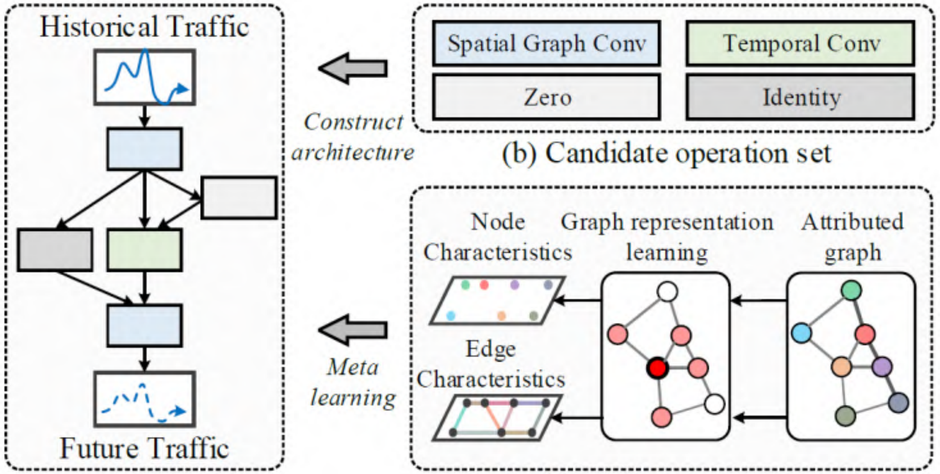
$$\mathbf{H}^{(l)} = \sum\nolimits_{j<l} \mathcal{F}_{jl}\left(\mathbf{H}^{(j)}\right)$$

    - $\mathcal{F}_{jl}$: 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: $$\mathbf{h}_{\mathcal{G}} = \mathrm{POOL}\left(\mathbf{H}\right)$$
    - 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

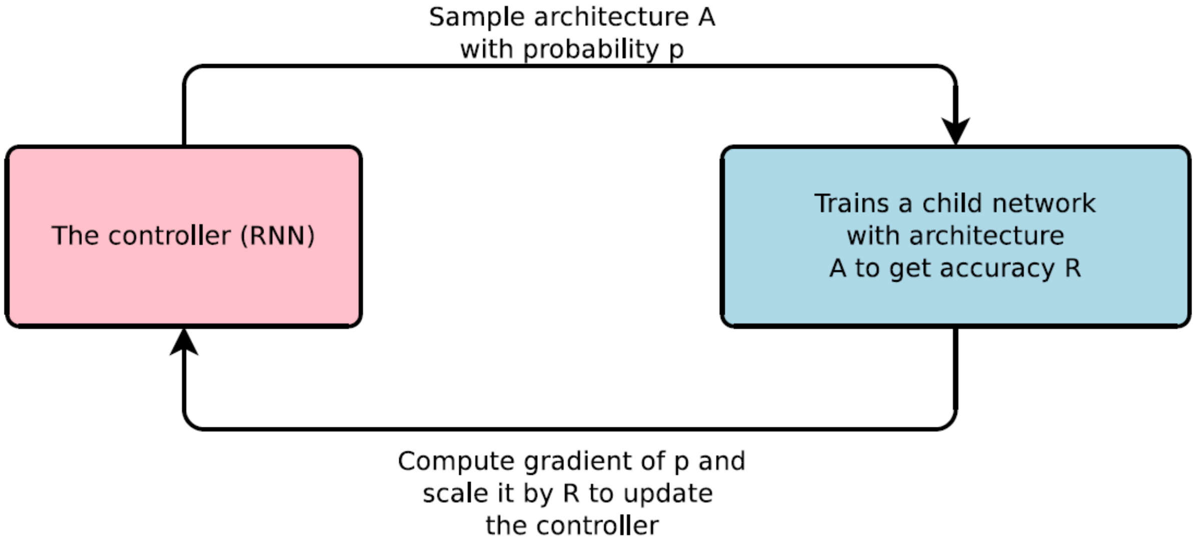


(a) Architectures (b) Candidate operation set (c) Node and edge characteristics

AutoSTG Neural Architecture Search for Predictions of Spatio-Temporal Graphs, *WWW 2021*

Graph NAS Search Strategy

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

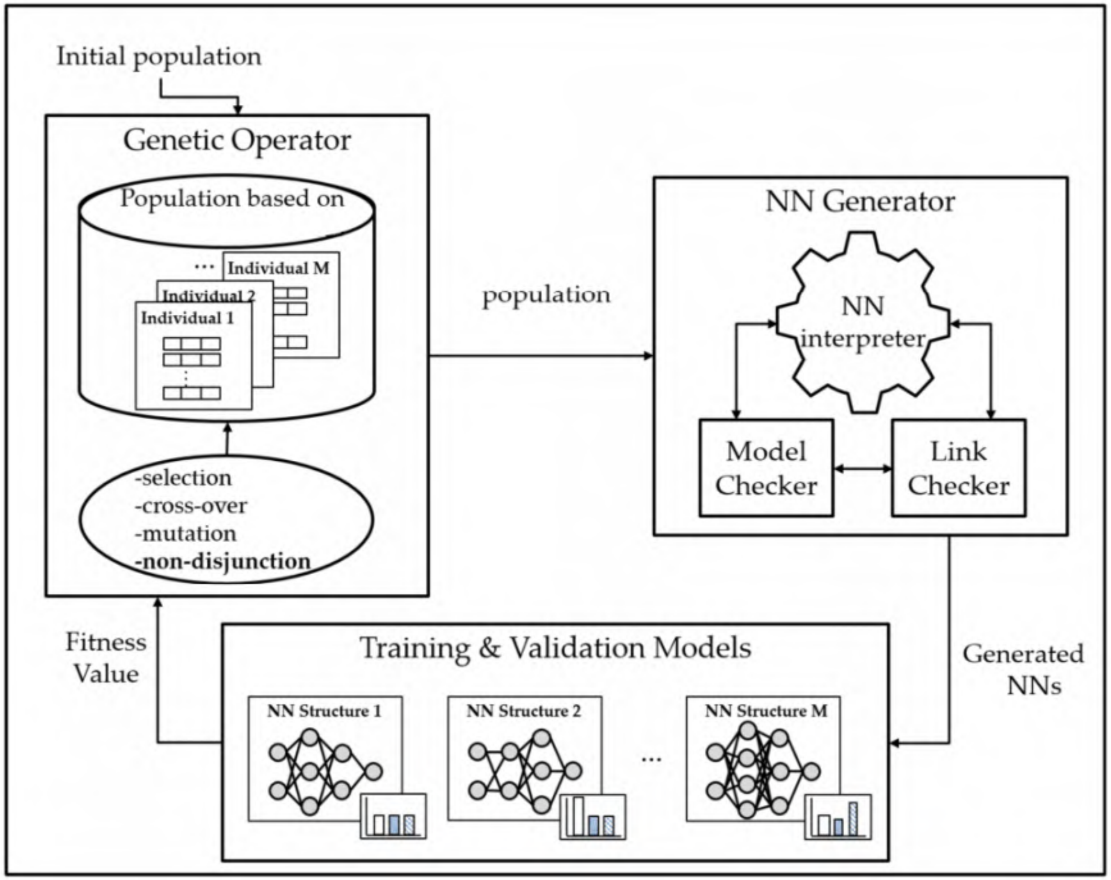


- Controller samples architecture (e.g., as a sequence)
- RL feedback rewards (e.g., validation performance) to update the controller

Neural Architecture Search with Reinforcement Learning, *ICLR 2017*

Graph NAS Search Strategy

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

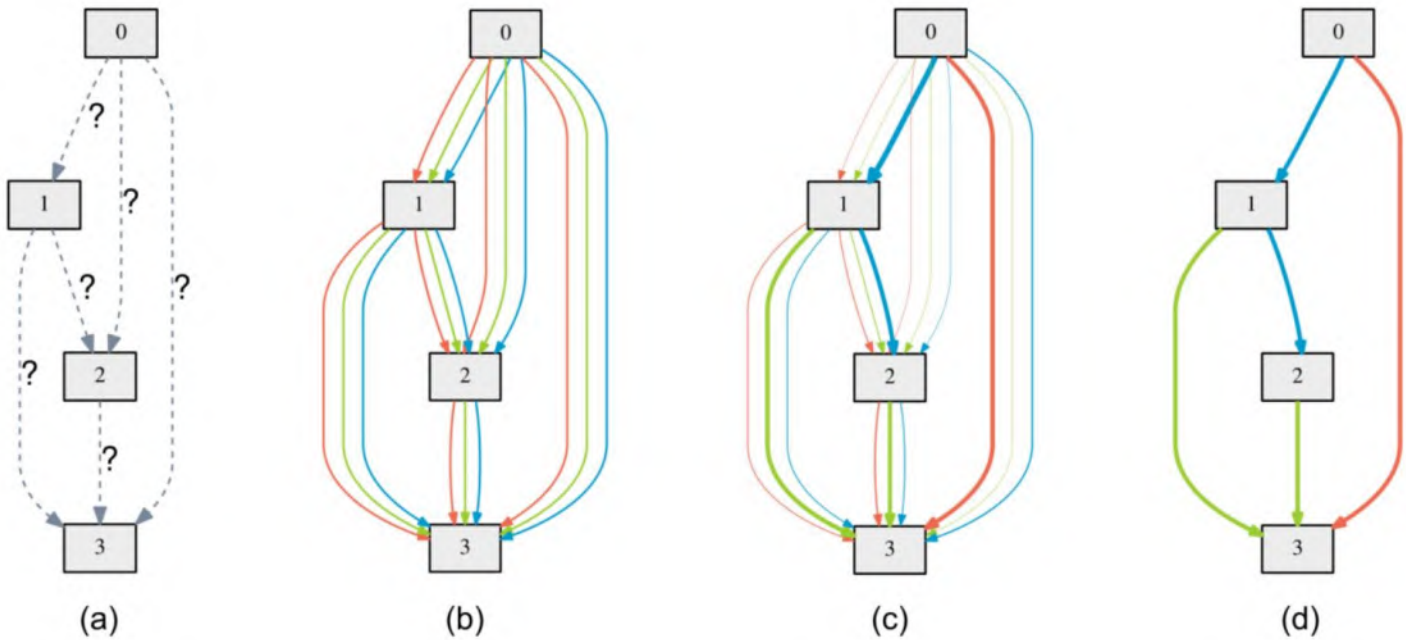


- Need to define how to sample parents, generate offspring, and update populations
- E.g., remove the worst individual (Real, et al., 2017), remove the oldest individual (Real, et al., 2018), or no remove (Liu, et al., 2018)

Evolutionary Neural Architecture Search (NAS) Using Chromosome Non-Disjunction for Korean Grammaticality Tasks, *Applied Science 2020*

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

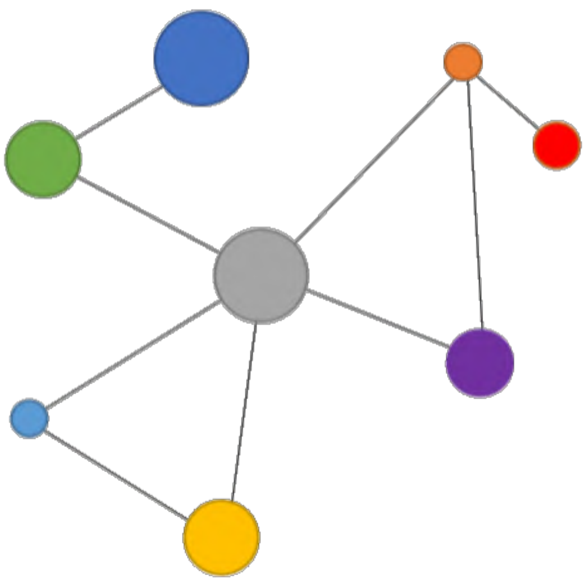
- Graph related
  - PyTorch geometric
  - DGL DEEP GRAPH LIBRARY
  - Paddle Graph Learning
  - PyTorch BigGraph
  - graph-learn

Gap!

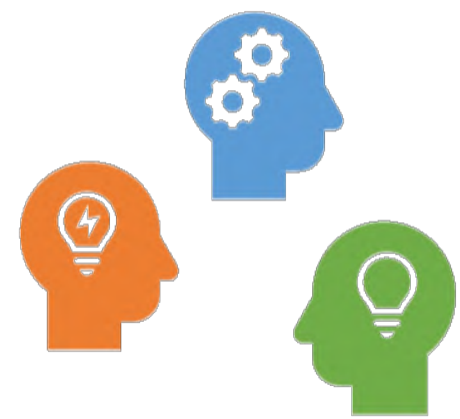
- AutoML related
  - NNI Neural Network Intelligence
  - AutoKeras
  - HYPEROPT
  - H2O AutoML
  - TPOT

Introduction – AutoGL

- We design the world’s first autoML framework & toolkit for machine learning on graphs.



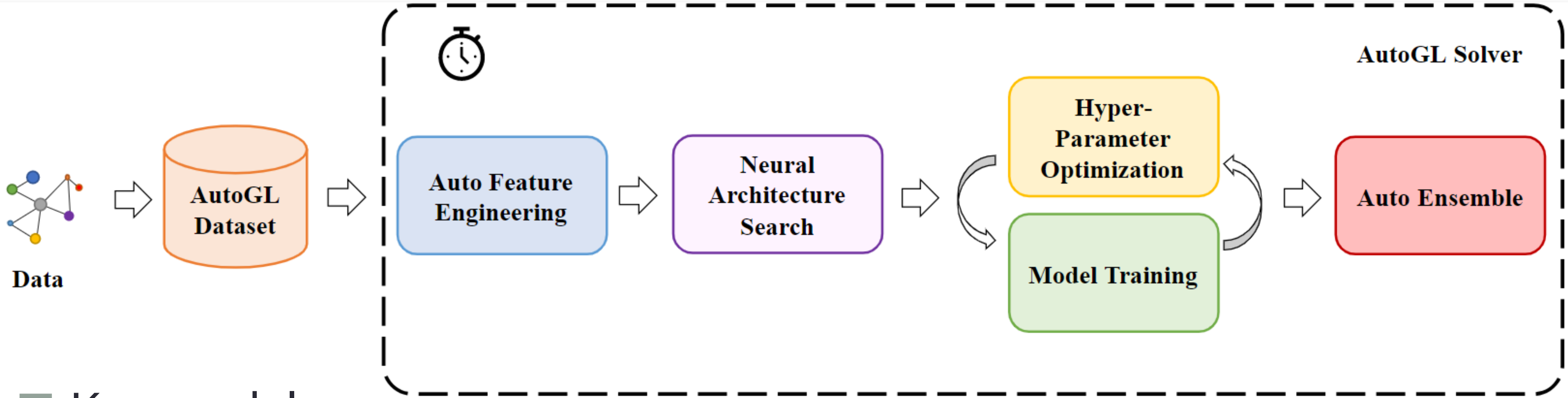
AutoGL



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

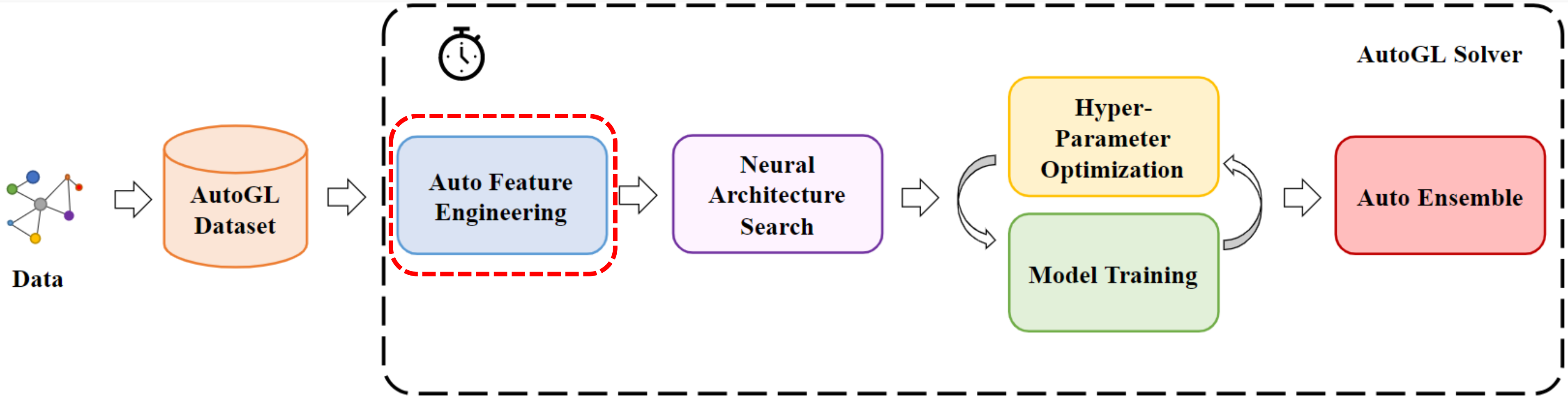
https://mn.cs.Tsinghua.edu.cn/AutoGL
https://github.com/THUMNLab/AutoGL

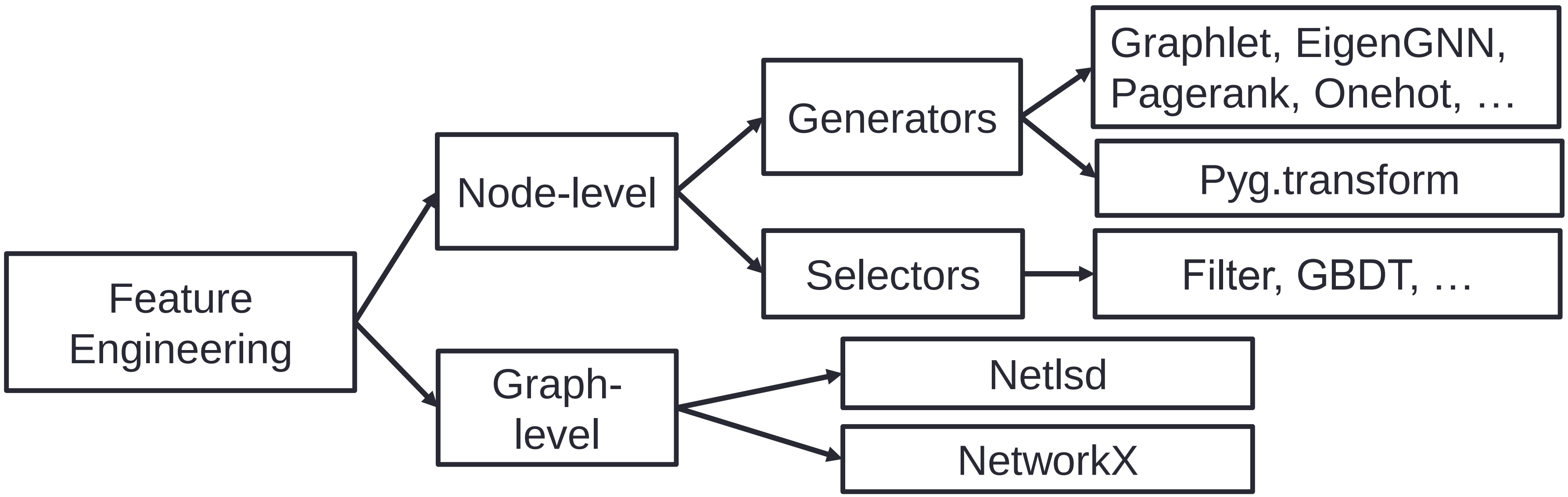
Modular Design



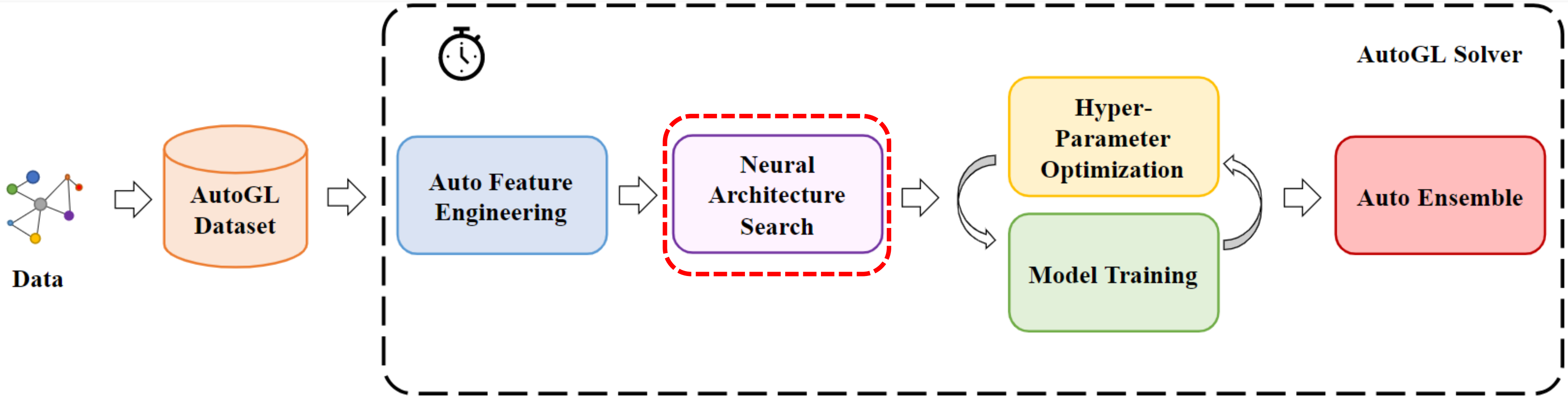
- Key modules:
  - AutoGL Dataset: manage graph datasets
  - AutoGL Solver: a high-level API to control the overall pipeline
  - Five functional modules:
    - Auto Feature Engineering,
    - Neural Architecture Search,
    - Hyper-parameter Optimization
    - Model Training
    - Auto Ensemble

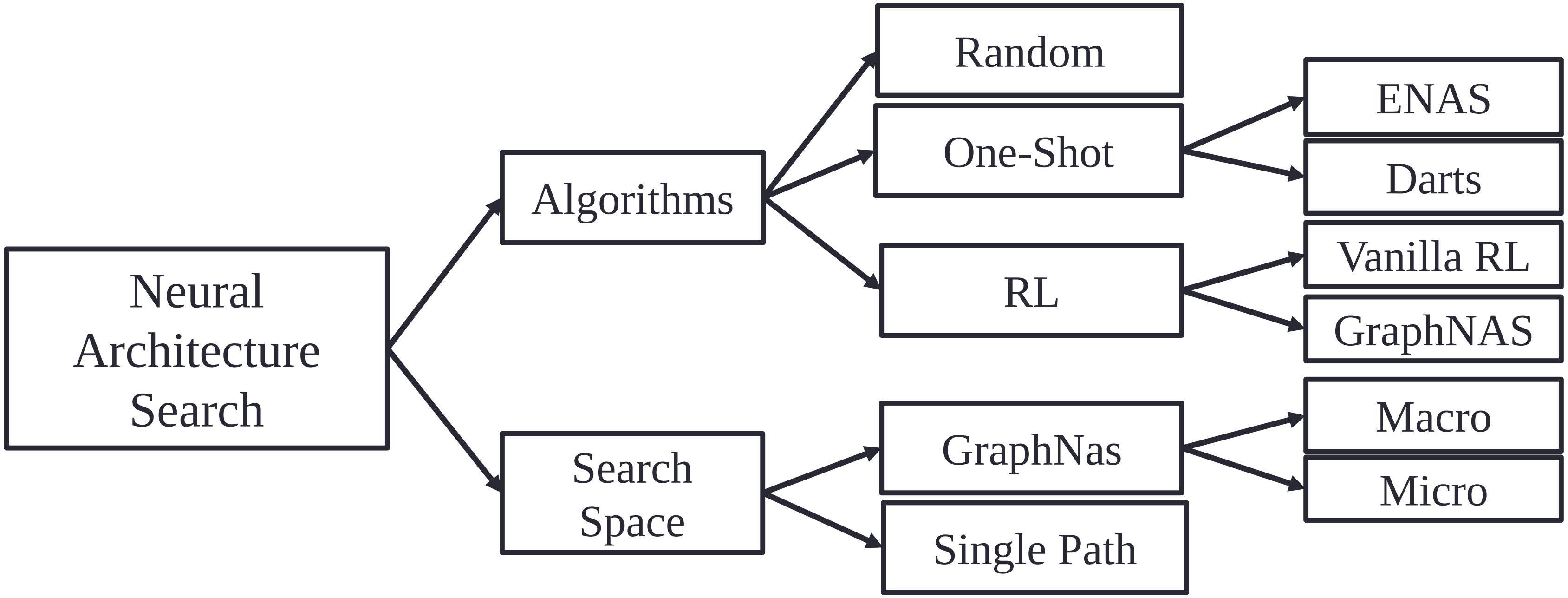
Feature Engineering



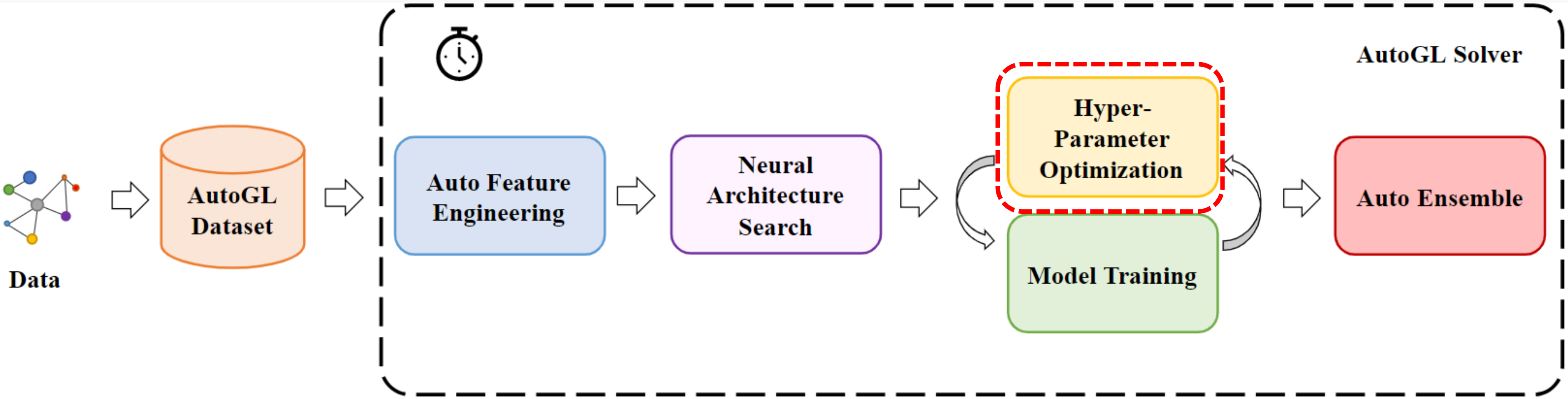


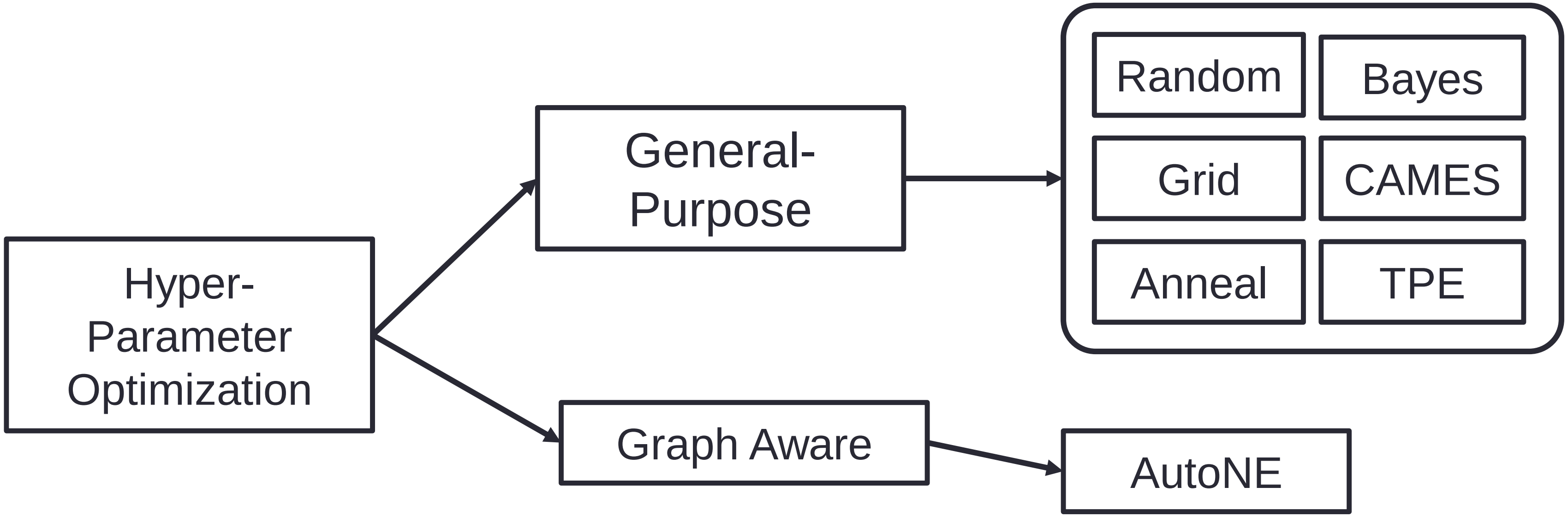
Neural Architecture Search



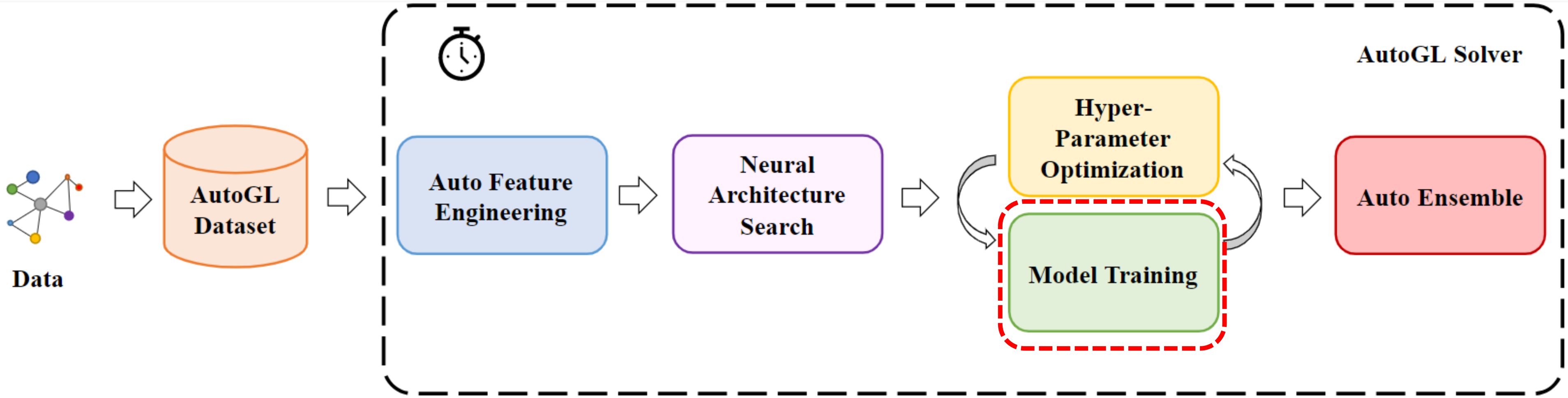


Hyper-Parameter Optimization





Model Training



Trainer

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

…

Model

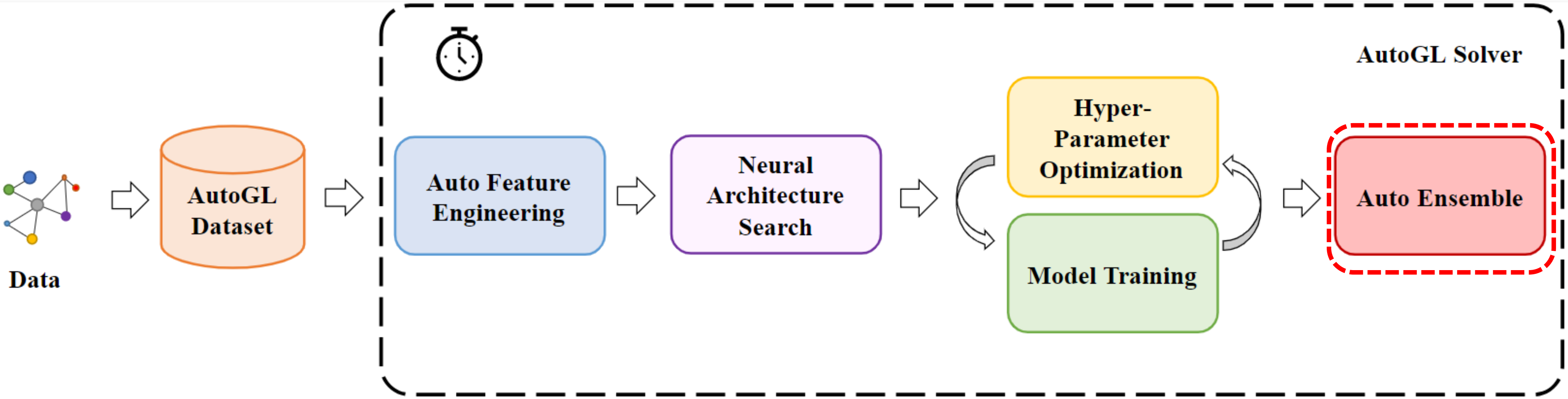
- Forward
- Ops & Architectures
- Dropout & Hidden

…

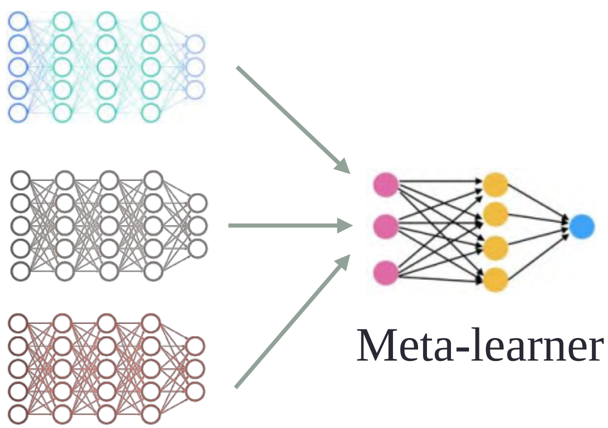
Currently supported models

- Node classification
  - GCN
  - GAT
  - GraphSAGE
- Link Prediction
- Graph classification
  - TopKPool
  - GIN

Ensemble



voting



stacking

Example Results

Table 1: The results of node classification

Model	Cora	CiteSeer	PubMed
GCN	80.9 ± 0.7	70.9 ± 0.7	78.7 ± 0.6
GAT	82.3 ± 0.7	71.9 ± 0.6	77.9 ± 0.4
GraphSAGE	74.5 ± 1.8	67.2 ± 0.9	76.8 ± 0.6
AutoGL	$\mathbf{83.2 \pm 0.6}$	$\mathbf{72.4 \pm 0.6}$	$\mathbf{79.3 \pm 0.4}$

Table 2: The results of graph classification

Model	MUTAG	PROTEINS	IMDB-B
Top-K Pooling	80.8 ± 7.1	69.5 ± 4.4	71.0 ± 5.5
GIN	82.7 ± 6.9	66.5 ± 3.9	69.1 ± 3.7
AutoGL	$\mathbf{87.6 \pm 6.0}$	$\mathbf{73.3 \pm 4.4}$	$\mathbf{72.1 \pm 5.0}$

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

Method	Trials	Cora GCN	Cora GAT	CiteSeer GCN	CiteSeer GAT	PubMed GCN	PubMed GAT
None		80.9 ± 0.7	82.3 ± 0.7	70.9 ± 0.7	71.9 ± 0.6	78.7 ± 0.6	77.9 ± 0.4
random	1	81.0 ± 0.6	81.4 ± 1.1	70.4 ± 0.7	70.1 ± 1.1	78.3 ± 0.8	76.9 ± 0.8
	10	82.0 ± 0.6	82.5 ± 0.7	71.5 ± 0.6	$\mathbf{72.2 \pm 0.7}$	79.1 ± 0.3	78.2 ± 0.3
	50	81.8 ± 1.1	$\mathbf{83.2 \pm 0.7}$	71.1 ± 1.0	72.1 ± 1.0	$\mathbf{79.2 \pm 0.4}$	78.2 ± 0.4
TPE	1	81.8 ± 0.6	81.9 ± 1.0	70.1 ± 1.2	71.0 ± 1.2	78.7 ± 0.6	77.7 ± 0.6
	10	82.0 ± 0.7	82.3 ± 1.2	71.2 ± 0.6	72.1 ± 0.7	79.0 ± 0.4	$\mathbf{78.3 \pm 0.4}$
	50	$\mathbf{82.1 \pm 1.0}$	83.2 ± 0.8	$\mathbf{72.4 \pm 0.6}$	71.6 ± 0.8	79.1 ± 0.6	78.1 ± 0.4

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!

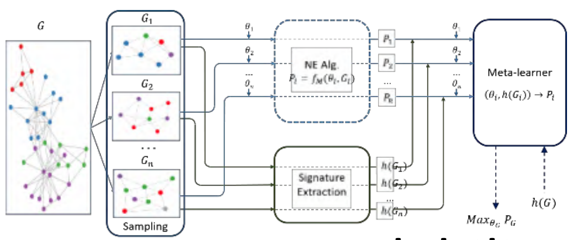
Contact: autogl@tsinghua.edu.cn

Overview of Our Representative Works

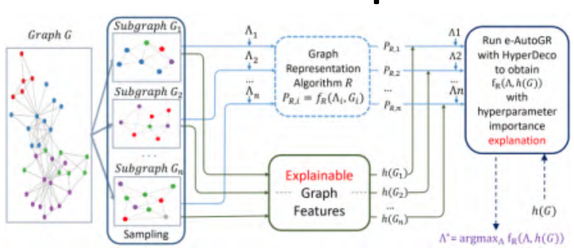
Our roadmap for automated machine learning on graphs

AutoGL HPO

AutoNE Scalability

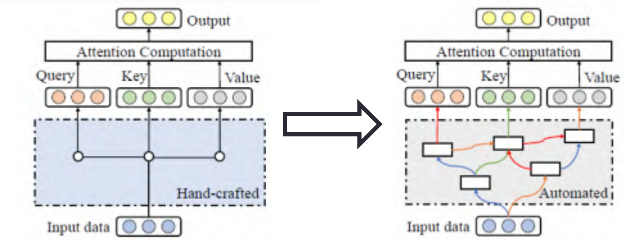


e-AutoGR Scalability + Explainability



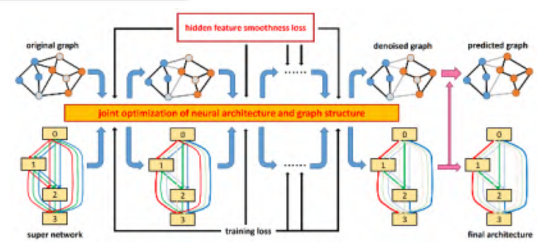
AutoGL NAS

AutoAttend Attention



(a) hand-crafted attention representation (b) automated attention representation

GASSO Graph Structure



AutoGL Tool and Library

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/