CogDL：助力图机器学习的研究与应用

岑宇阔
KEG实验室，清华大学计算机系
https://github.com/THUDM/cogdl

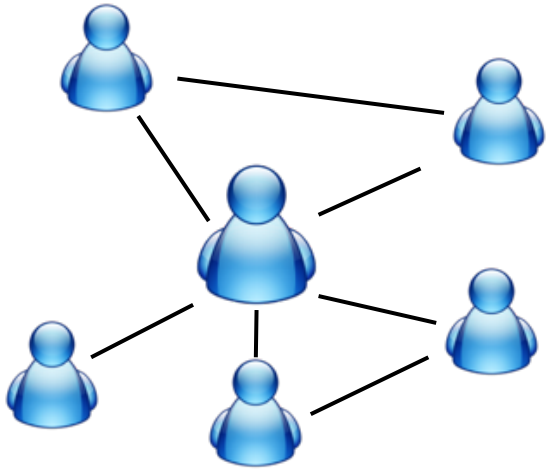
CogDL交流群
群号：792564770

报告提纲

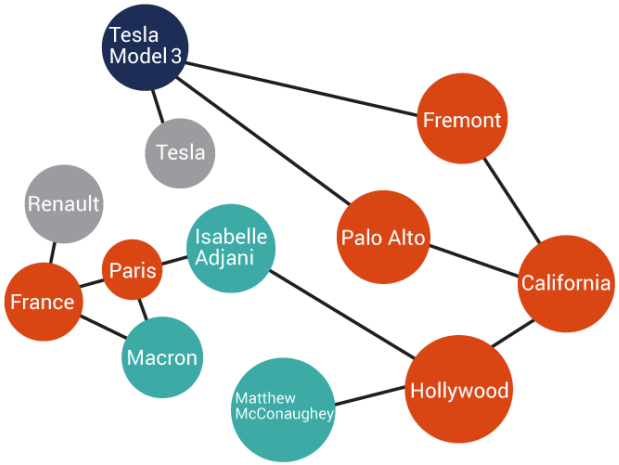
- 图机器学习简介
  - 背景、发展、挑战
- 图神经网络模型基础
  - GCN / GraphSAGE / GAT模型
- 图机器学习框架CogDL
  - 简介、基础用法、复现结果、自定义用法
- 图深度学习的前沿研究
  - 大规模图、深层网络、自动图机器学习、图基准

为什么要研究图(Graph)?

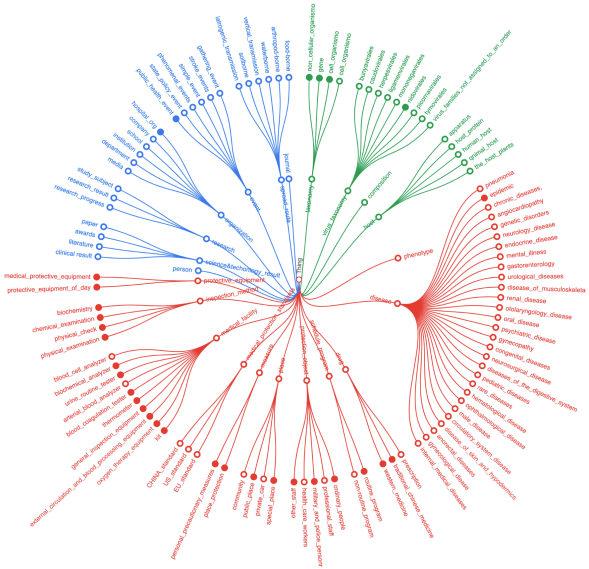
- 图是一种通用的语言，来描述和分析实体和实体之间的关系/交互。图结构数据无处不在。



社交网络



知识图谱

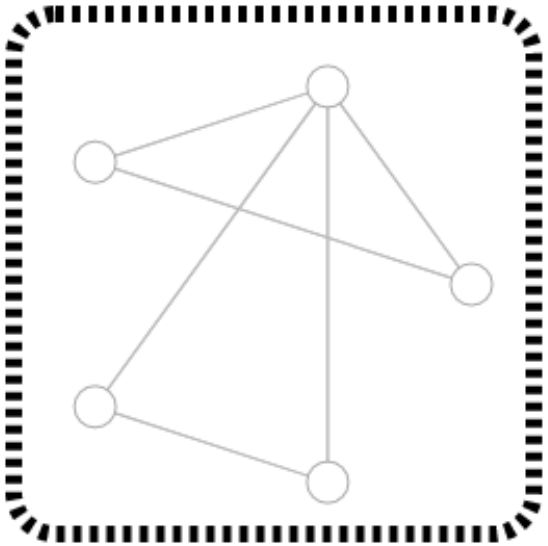


COVID图谱

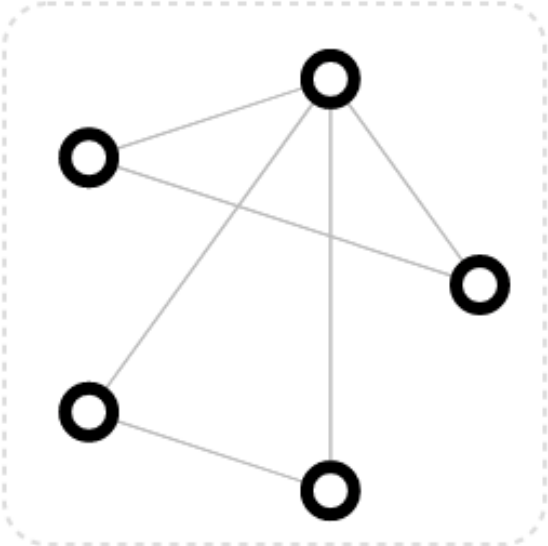
from Prof. Jie Tang's slides

图的形式化定义 (1)

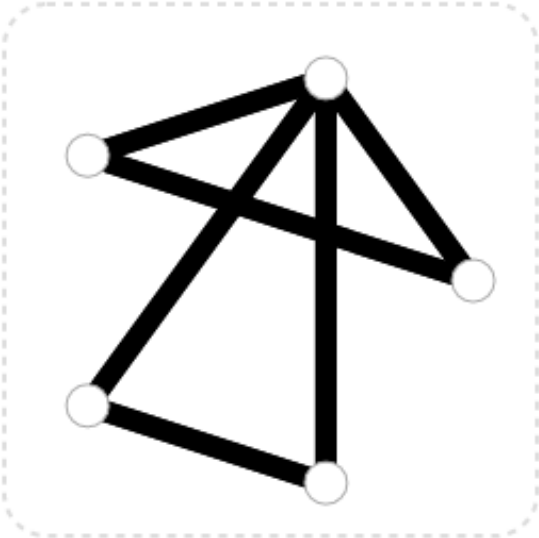
- 图$G = (V, E)$，$|V| = \mathrm{N}$个节点和$|E| = M$条边



G: 图(graph)



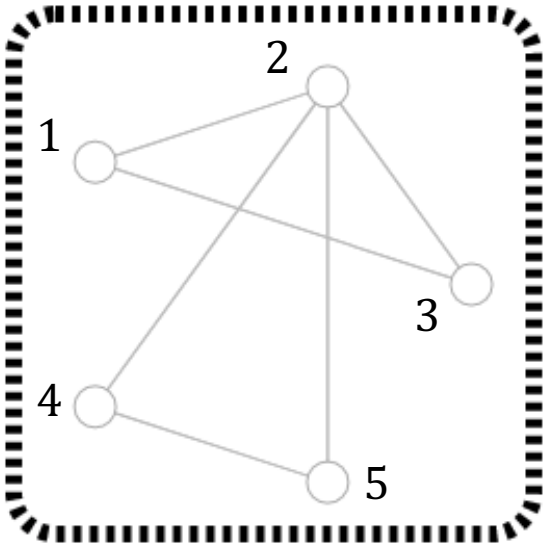
V: 节点(vertex)



E: 边(edge)

图的形式化定义 (2)

- 通常用邻接矩阵$A \in \mathbb{R}^{N \times N}$来表示图结构信息。
- $A_{i,j} > 0$表明存在一条节点i到j的边。
- 若G为无向图，$A_{i,j} = A_{j,i}$

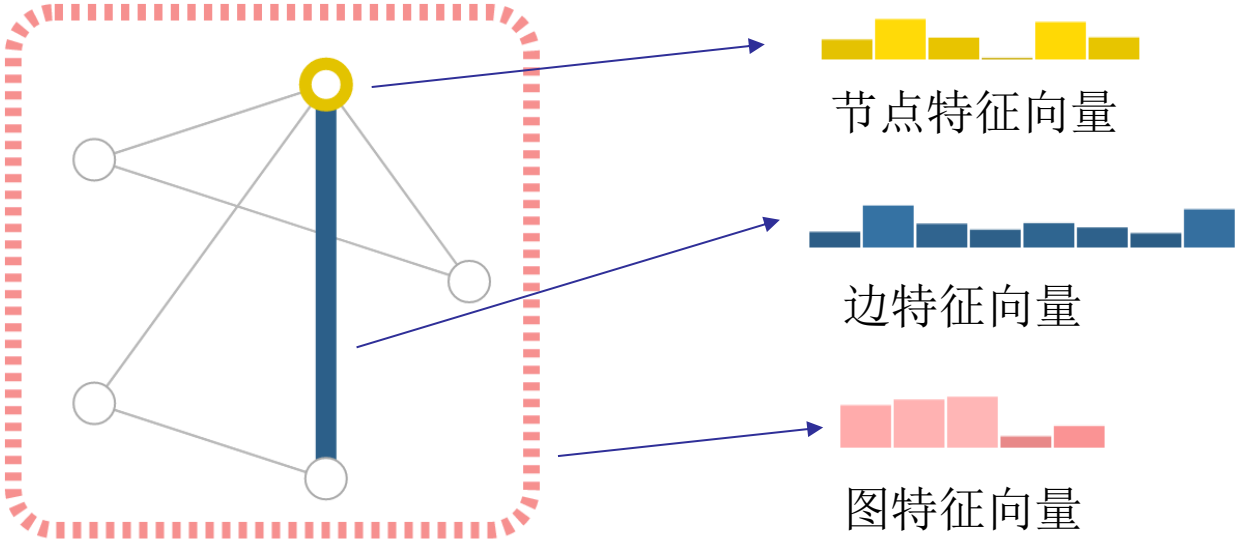


$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

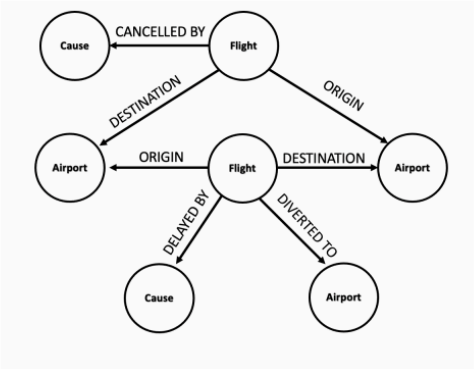
邻接矩阵A

图的形式化定义 (2)

- 图中的每个节点通常包含一定的信息，以特征向量表示。
- 用特征矩阵$\boldsymbol{X} \in \mathbb{R}^{N \times F}$来表示所有节点的特征向量。
- 在某些图中，边或整个图也可能有相应的特征向量。



图结构数据 (1)



Event Graphs

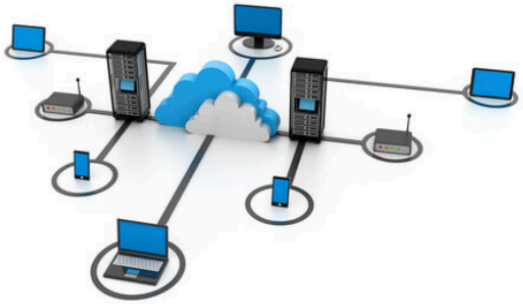
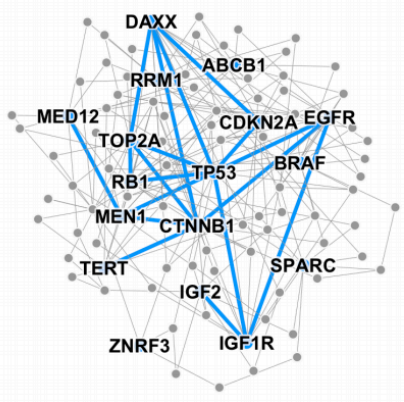


Image credit: SalientNetworks

Computer Networks



Disease Pathways

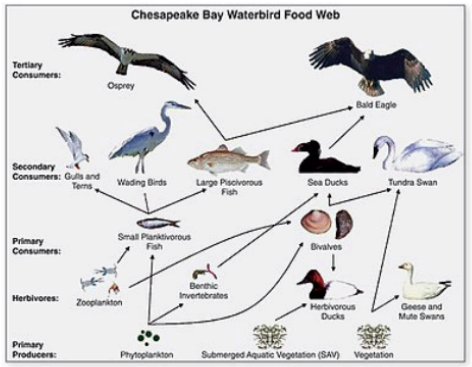


Image credit: Wikipedia

Food Webs

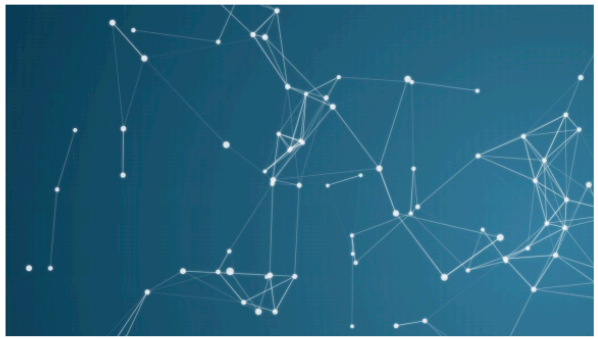


Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

Underground Networks

from Prof. Jure Leskovec's slides (cs224w)

图结构数据 (2)

Image credit: Medium

Social Networks

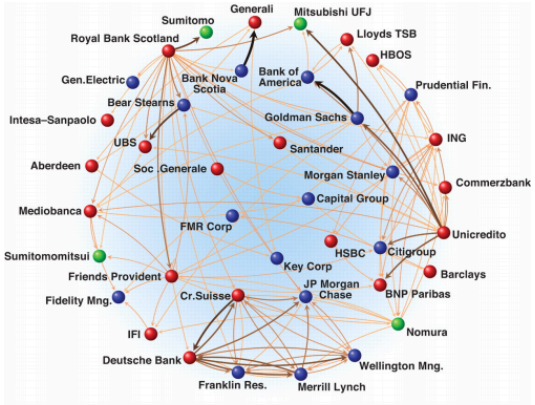
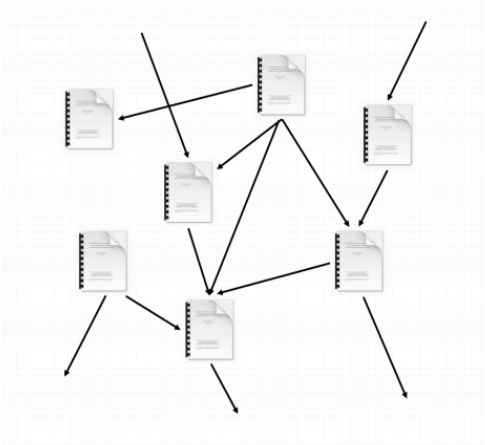


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks



Citation Networks

Image credit: Missoula Current News

Internet

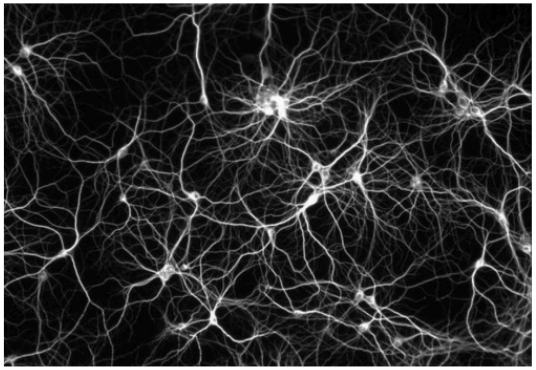


Image credit: The Conversation

Networks of Neurons

from Prof. Jure Leskovec's slides (cs224w)

图结构数据 (3)

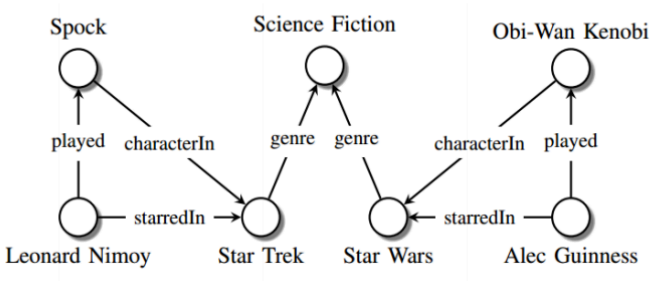


Image credit: Maximilian Nickel et al

Knowledge Graphs

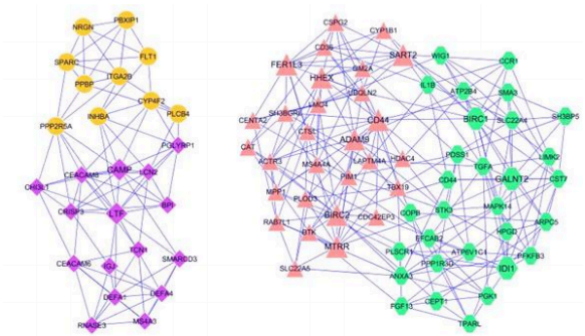


Image credit: ese.wustl.edu

Regulatory Networks

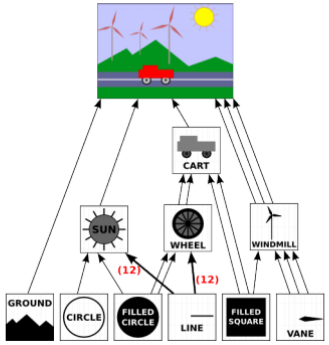


Image credit: math.hws.edu

Scene Graphs

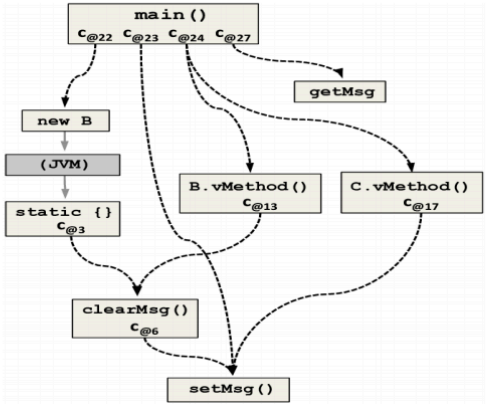


Image credit: ResearchGate

Code Graphs

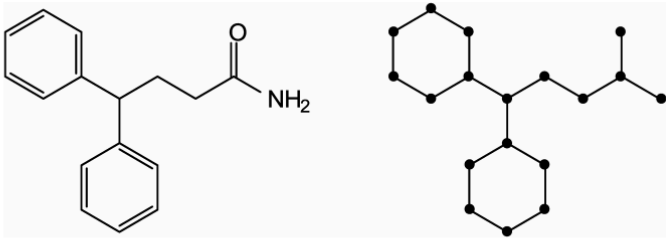


Image credit: MDPI

Molecules

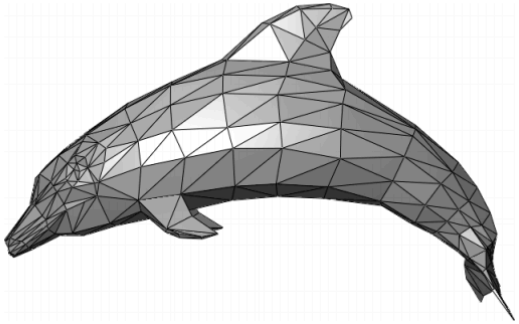


Image credit: Wikipedia

3D Shapes

from Prof. Jure Leskovec's slides (cs224w)

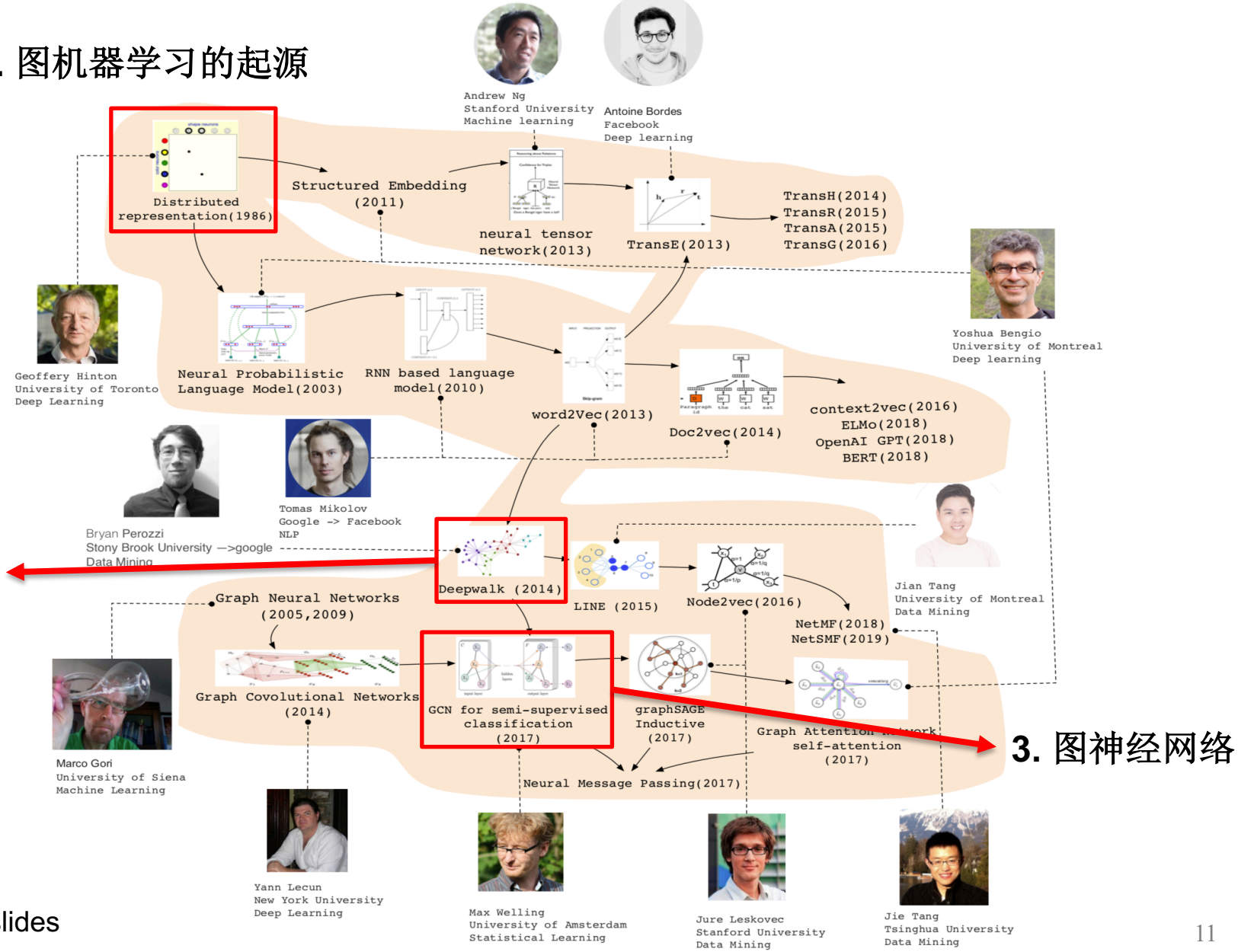
图机器学习(Graph Machine Learning)

使用机器学习算法解决图数据上的问题：

- **节点分类 (Node classification)：** 预测节点的属性
  - 例子：预测社交网络中用户的类别
- **链接预测 (Link prediction)：** 预测两个节点之间是否缺失边
  - 例子：知识图谱补全
- **图分类 (Graph classification)：** 预测整个图的性质
  - 例子：分子的性质预测
- **社区检测 (Community detection)：** 检测节点是否形成了社区
  - 例子：社交网络中的社区检测

图机器学习的发展

1. 图机器学习的起源



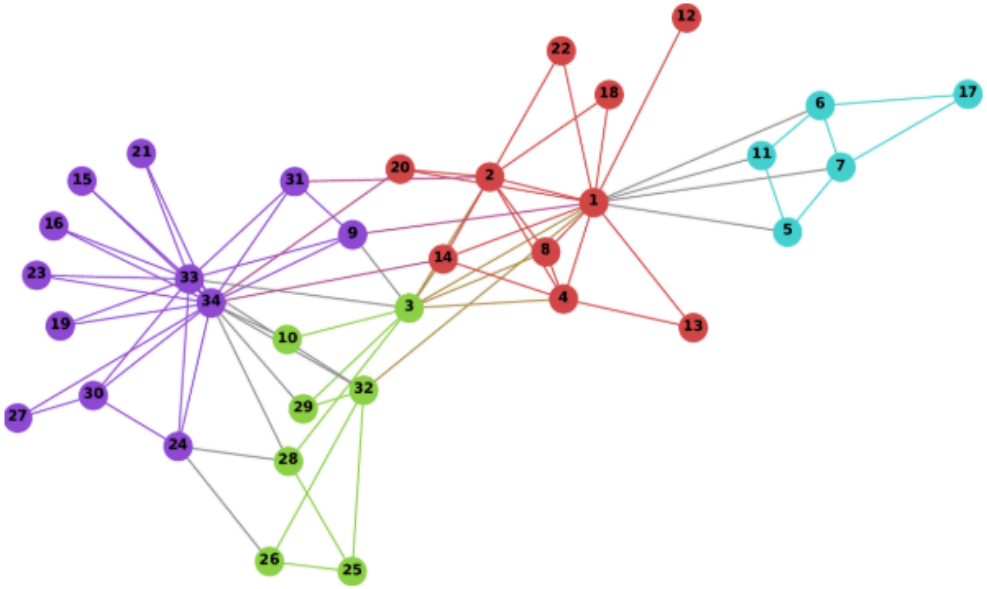
2. 图表示学习

3. 图神经网络

from Prof. Jie Tang's slides

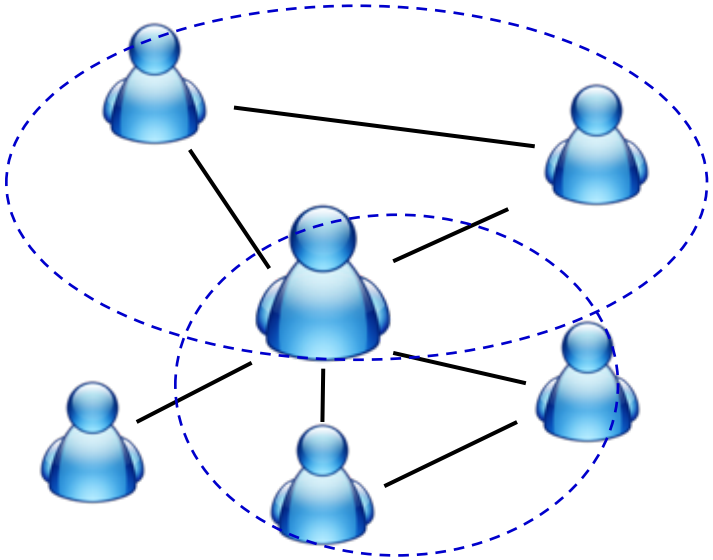
图机器学习的挑战

- 图结构的复杂性：
  - 图的大小不定
  - 拓扑结构不定
  - 节点顺序不定
  - 静态图 vs. 动态图
  - 同构图 vs. 异构图
  - 图鲁棒性
  - …



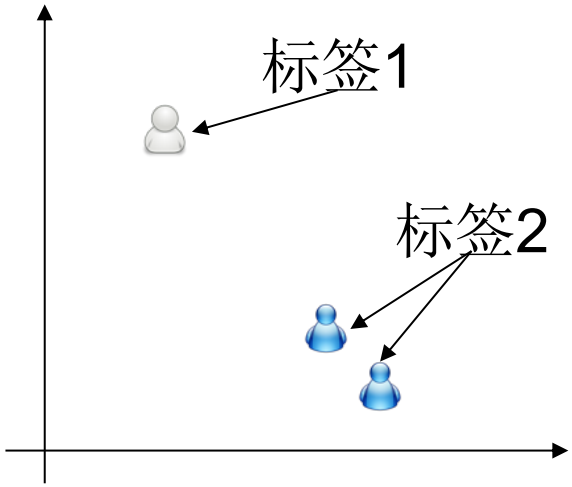
图表示学习(Graph Representation Learning)

目标： 给定图$G = (V, E)$，学习一个映射$f: u \rightarrow \mathbb{R}^d$，得到节点$u$的表示向量$f(u) = z_u$



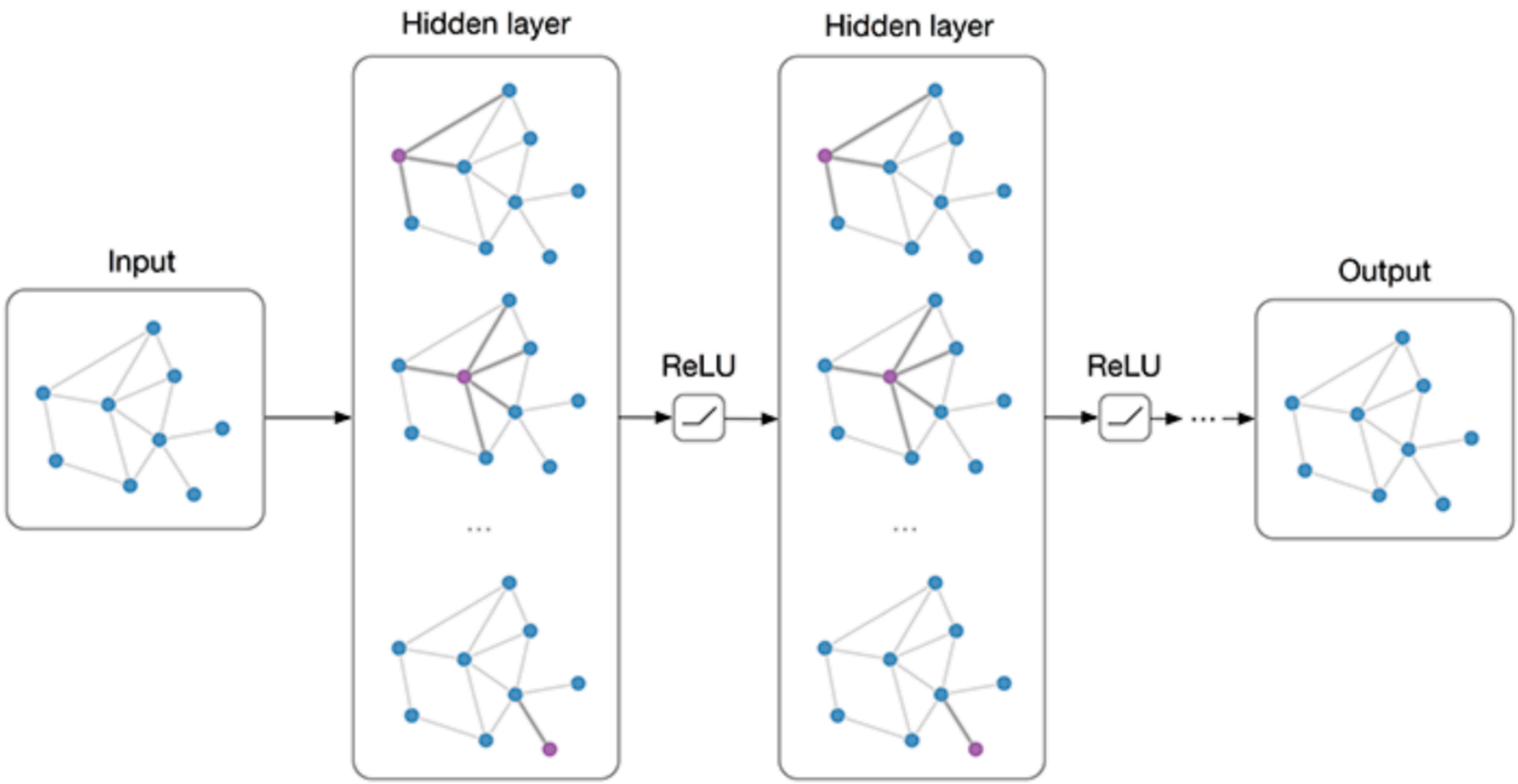
d维表示向量z_u，$d << |V|$

以节点分类为例：相同标签的节点在向量空间中**距离更接近。**



图深度学习

- 构建适合图数据的神经网络！



Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In ICLR ’17.

回顾卷积神经网络（CNN）

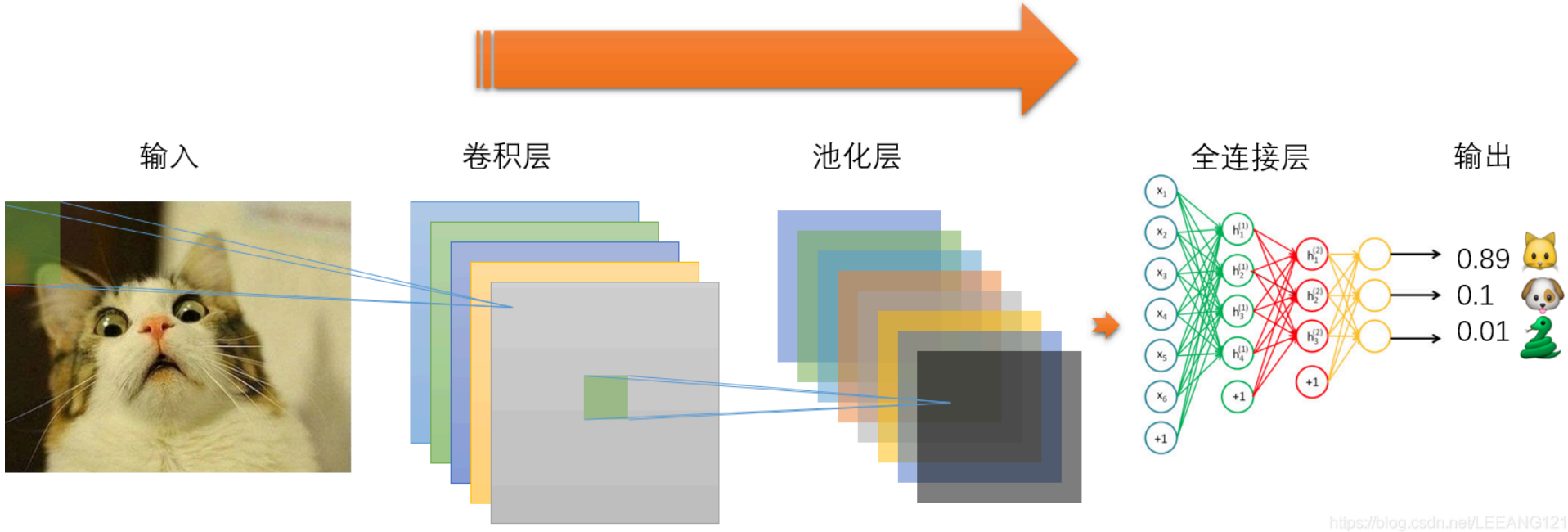
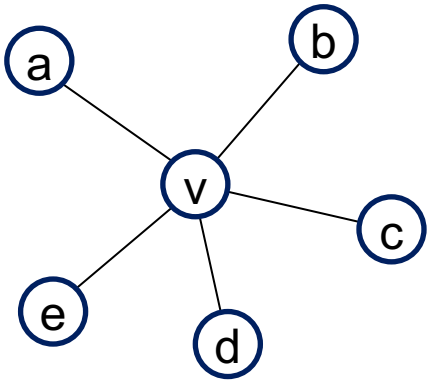


image credit: https://www.guyuehome.com/23142

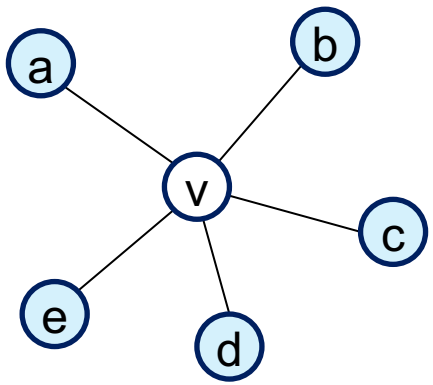
图神经网络初步



$$\boldsymbol{h}_v = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

- **邻居聚合:**
  - 聚合邻居节点的信息并传给神经网络
  - 类似于CNN中的卷积操作---图卷积（graph convolution）！

图卷积网络 GCN



第k层的训练参数

非线性的激活函数 (例如, ReLU)

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in N(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

节点v第k层的表示

节点v的邻居节点集合

1. Kipf et al. Semi-supervised Classification with Graph Convolutional Networks. ICLR 2017

图卷积网络 GCN

- $\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u\in \mathrm{N}(v)\cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$
- **<=>**$\boldsymbol{H}^{(k)} = \sigma\left(\widehat{\boldsymbol{D}}^{-\frac{1}{2}}\widehat{\boldsymbol{A}}\widehat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}_k\right)$
- 其中$\widehat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$，$\widehat{\boldsymbol{D}}_{ii} = \sum_j \widehat{\boldsymbol{A}}_{ij}$
- 不妨令$\widetilde{\boldsymbol{A}} = \widehat{\boldsymbol{D}}^{-\frac{1}{2}}\widehat{\boldsymbol{A}}\widehat{\boldsymbol{D}}^{-\frac{1}{2}}$
- **=>** $\boldsymbol{H}^{(k)} = \sigma\left(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}_k\right)$

图卷积网络 GCN 表现

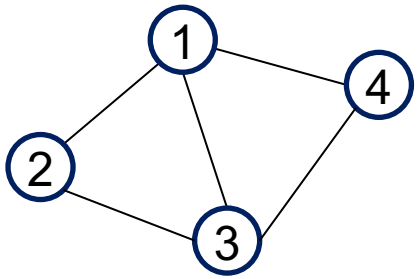
- 2层GCN：$\boldsymbol{Z} = \mathrm{softmax}\left(\tilde{\boldsymbol{A}}\,\sigma\left(\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_0\right)\boldsymbol{W}_1\right)$

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)

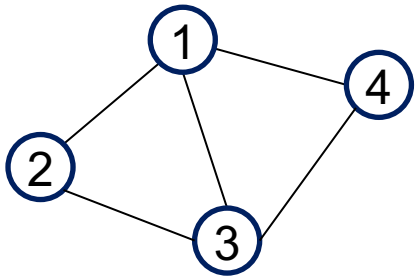
GCN 例子

- $\widehat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$



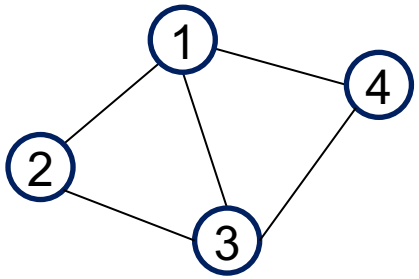
- $\widehat{\boldsymbol{D}} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \widehat{\boldsymbol{D}}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{3}} \end{bmatrix}$

GCN 例子



- $\widetilde{\boldsymbol{A}} = \widehat{\boldsymbol{D}}^{-\frac{1}{2}}\widehat{\boldsymbol{A}}\widehat{\boldsymbol{D}}^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{4} & \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} & \frac{1}{3} & \frac{\sqrt{3}}{6} & 0 \\ \frac{1}{4} & \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} & 0 & \frac{\sqrt{3}}{6} & \frac{1}{3} \end{bmatrix}$, 其中 $\frac{\sqrt{3}}{6} \approx 0.2887$

GCN 例子

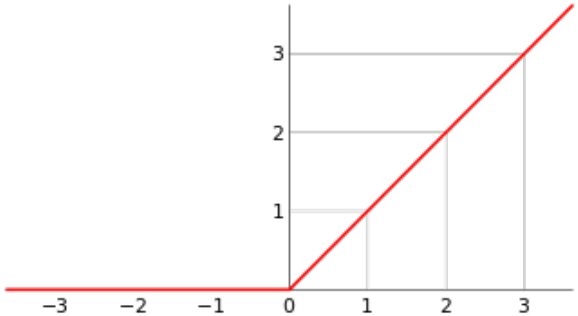


- $\boldsymbol{H}^{(k)} = \sigma\left(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}_k\right)$
- $\boldsymbol{H}^{(0)} = \boldsymbol{X} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \boldsymbol{W}_1 = \begin{bmatrix} 1 & -0.5 \\ 0.5 & 1 \end{bmatrix}$
- $\boldsymbol{H}^{(0)}\boldsymbol{W}_1 = \begin{bmatrix} 1 & -0.5 \\ 0.5 & 1 \\ 1 & -0.5 \\ 1.5 & 0.5 \end{bmatrix}$

GCN 例子

- $$\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(0)}\boldsymbol{W}_1 = \begin{bmatrix} \frac{1}{4} & \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} & \frac{1}{3} & \frac{\sqrt{3}}{6} & 0 \\ \frac{1}{4} & \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} & 0 & \frac{\sqrt{3}}{6} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 1 & -0.5 \\ 0.5 & 1 \\ 1 & -0.5 \\ 1.5 & 0.5 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{3}+\frac{1}{2} & \frac{\sqrt{3}}{4}-\frac{1}{4} \\ \frac{\sqrt{3}}{3}+\frac{1}{6} & -\frac{\sqrt{3}}{6}+\frac{1}{3} \\ \frac{\sqrt{3}}{3}+\frac{1}{2} & \frac{\sqrt{3}}{4}-\frac{1}{4} \\ \frac{\sqrt{3}}{3}+\frac{1}{2} & -\frac{\sqrt{3}}{6}+\frac{1}{6} \end{bmatrix}$$

- $$\boldsymbol{H}^{(1)} = ReLU\left(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(0)}\boldsymbol{W}_1\right) = \begin{bmatrix} \frac{\sqrt{3}}{3}+\frac{1}{2} & \frac{\sqrt{3}}{4}-\frac{1}{4} \\ \frac{\sqrt{3}}{3}+\frac{1}{6} & -\frac{\sqrt{3}}{6}+\frac{1}{3} \\ \frac{\sqrt{3}}{3}+\frac{1}{2} & \frac{\sqrt{3}}{4}-\frac{1}{4} \\ \frac{\sqrt{3}}{3}+\frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 1.0774 & 0.1830 \\ 0.7440 & 0.0447 \\ 1.0774 & 0.1830 \\ 1.0774 & 0 \end{bmatrix}$$



GCN 训练

- $Z = \mathrm{softmax}\left(\boldsymbol{H}^{(2)}\right) = \mathrm{softmax}\left(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(1)}\boldsymbol{W}_2\right)$
- $\boldsymbol{W}_2 = \begin{bmatrix} 0.5 & -0.5 \\ 1 & 0.5 \end{bmatrix}$
- $Z = \mathrm{softmax}\left(\boldsymbol{H}^{(2)}\right) = \mathrm{softmax}\left(\begin{bmatrix} 0.6366 & -0.4800 \\ 0.5556 & -0.3747 \\ 0.6366 & -0.4800 \\ 0.5962 & -0.4377 \end{bmatrix}\right)$

$$= \begin{bmatrix} 0.7534 & 0.2466 \\ 0.7171 & 0.2829 \\ 0.7534 & 0.2466 \\ 0.7377 & 0.2623 \end{bmatrix}$$

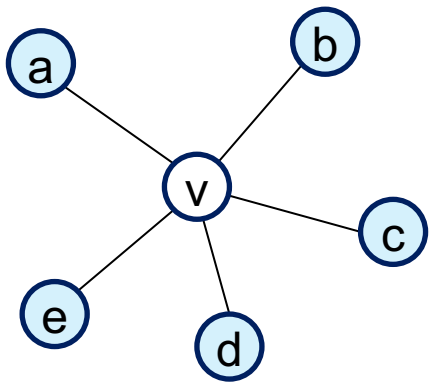
GCN 训练

- $Y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$
- 损失函数使用cross entropy

$$L = -\frac{1}{4}\sum_{i=1}^{4}\sum_{j=1}^{2} Y_{ij} \ln Z_{ij} = 0.5334$$

- 通过反向传播来更新模型参数$\boldsymbol{W}_1, \boldsymbol{W}_2$

GraphSAGE



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

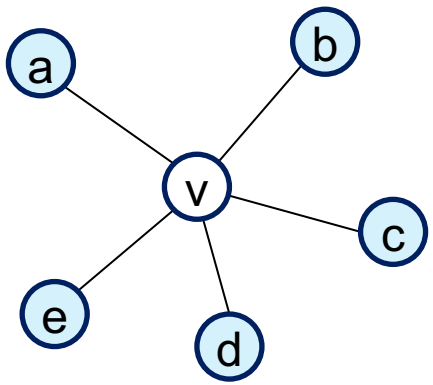
GraphSAGE

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}_k \cdot \mathrm{AGG}\left(\{h_u^{k-1}, \forall u \in N(v)\}\right), \boldsymbol{B}_k h_v^{k-1}])$$

通用的聚合函数: 任何将一个向量集合映射到单个向量的可微函数

1. Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
2. Slide snipping from “Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning”

GraphSAGE



GCN

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

GraphSAGE

不使用简单的求和，而是将邻居节点的表示聚合起来后跟自己的表示拼接起来

$$\boldsymbol{h}_v^k = \sigma\left([\boldsymbol{A}_k \cdot \mathrm{AGG}\left(\{h_u^{k-1}, \forall u \in N(v)\}\right), \boldsymbol{B}_k h_v^{k-1}]\right)$$

通用的聚合函数: 任何将一个向量集合映射到单个向量的可微函数

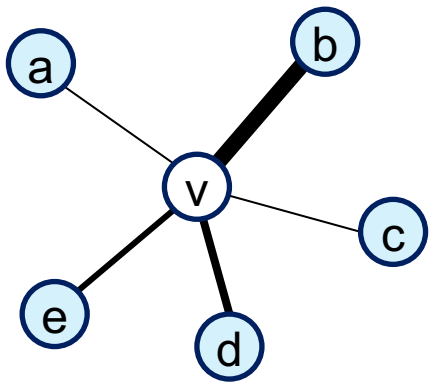
1. Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
2. Slide snipping from “Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning”

GraphSAGE 表现

- 有监督(supervised)，无监督(unsupervised)
- 不同聚合函数GCN, mean, LSTM, pool

	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

图注意力网络 GAT



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

GAT

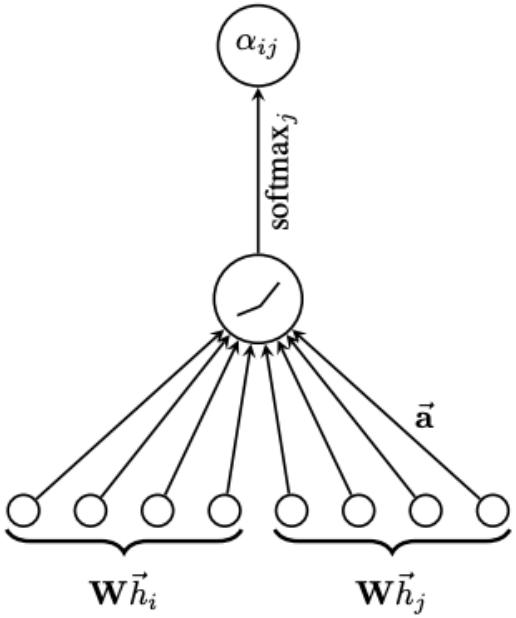
$$\boldsymbol{h}_v^k = \sigma(\sum_{u \in \mathrm{N}(v) \cup v} \alpha_{v,u} \boldsymbol{W}^k \boldsymbol{h}_u^{k-1})$$

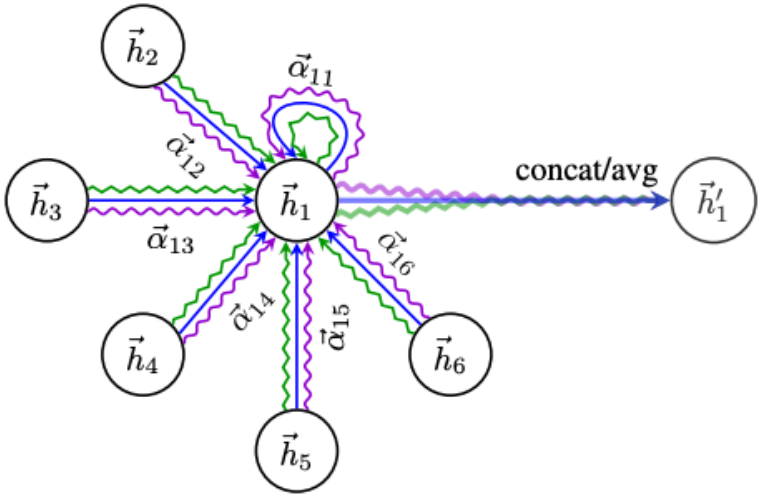
可学习的注意力权重

1. Velickovic et al. Graph Attention Networks. ICLR 2018

图注意力网络 GAT

- 如何来计算注意力权重？
- 系数$e_{vu} = \boldsymbol{r}^{\boldsymbol{T}}[\boldsymbol{W}\boldsymbol{h}_v || \boldsymbol{W}\boldsymbol{h}_u]$
- 注意力 $\alpha_{vu} = softmax(e_{vu}) = \dfrac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$





图注意力网络 GAT 表现

Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	**0.973** ± 0.002

CogDL简介

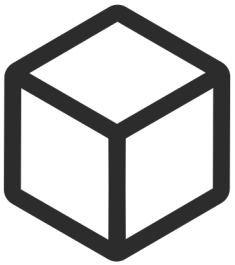
愿景

在图机器学习相关的下游任务和应用中，为广大的研发人员提供易用的接口、可扩展的模块和高效的性能。

理念

易用接口

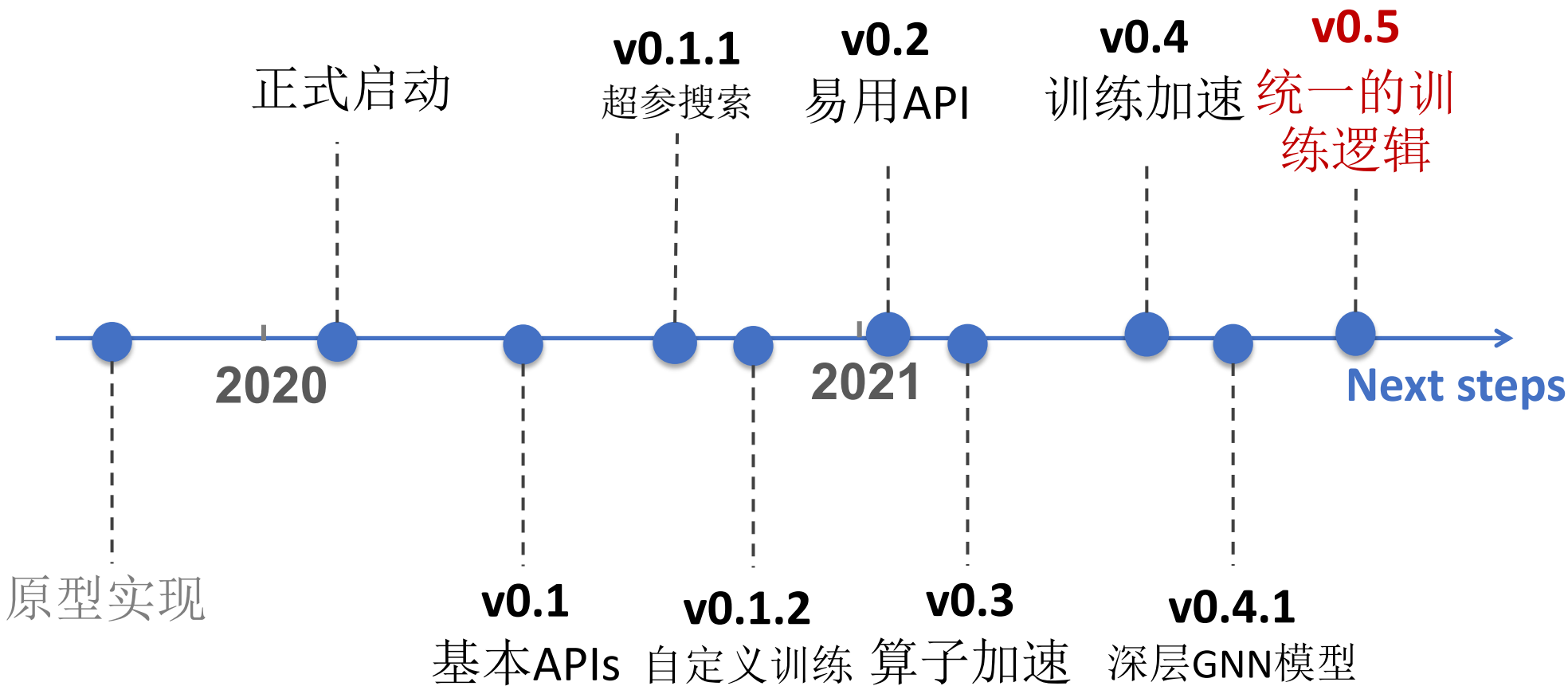
可扩展性



高效计算

Yukuo Cen, Zhenyu Hou, Yan Wang, Qibin Chen, Yizhen Luo, Xingcheng Yao, Aohan Zeng, Shiguang Guo, Peng Zhang, Guohao Dai, Yu Wang, Chang Zhou, Hongxia Yang, and Jie Tang. CogDL: An Extensive Toolkit for Deep Learning on Graphs. arXiv preprint 2021.

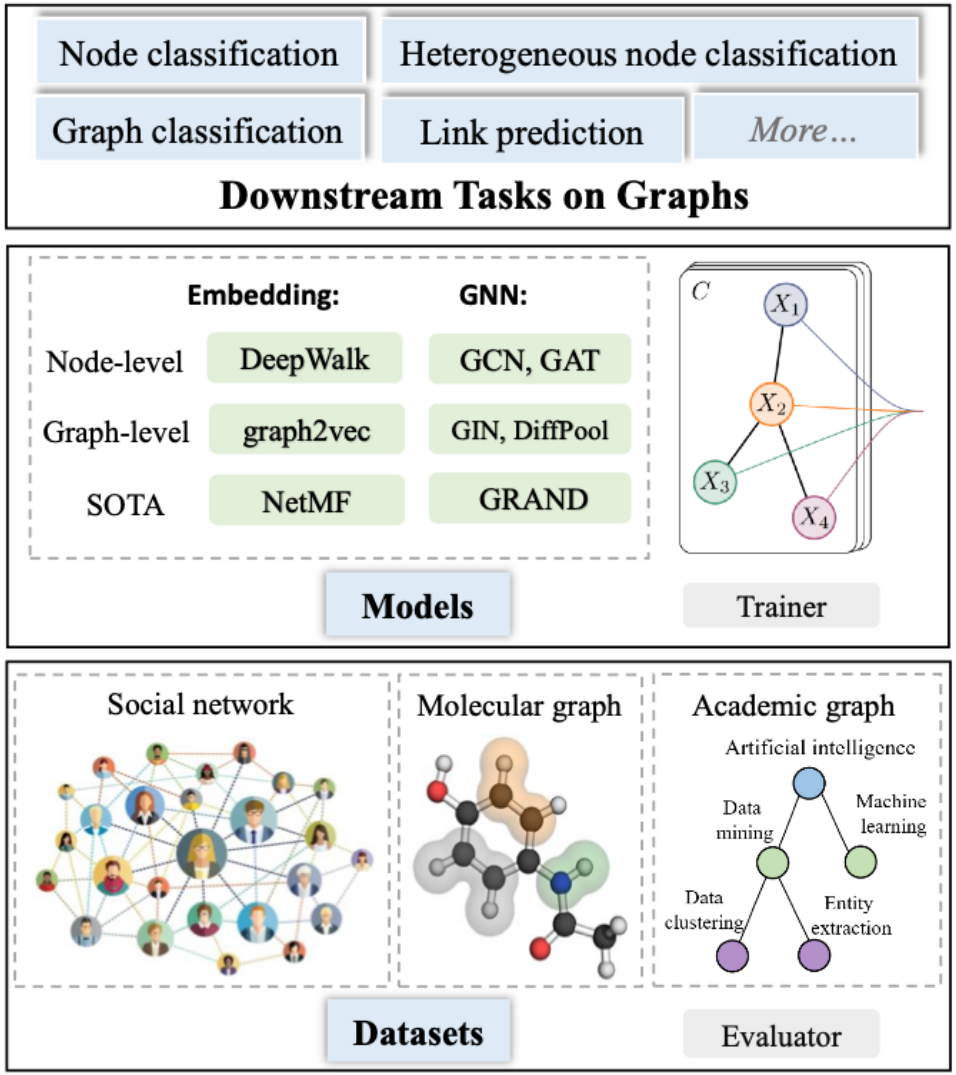
CogDL开发进展



基于PyTorch框架
CogDL安装方式: **pip install cogdl**
or git clone https://github.com/THUDM/cogdl

CogDL中集成的任务、模型、数据集

- 集成10+类下游任务:
  - 节点分类
  - 图分类
- 集成60+个不同的数据集:
  - 社交网络
  - 学术网络
  - 分子图
- 集成70+种不同的模型:
  - 表示学习模型
  - 图神经网络模型



CogDL基础用法 - Experiment API

- 传入数据集、模型、(超参)、(超参搜索范围)

```python
from cogdl import experiment

# basic usage
experiment(dataset="cora", model="gcn")

# set other hyper-parameters
experiment(dataset="cora", model="gcn", hidden_size=32, epochs=200)

# run over multiple models on different seeds
experiment(dataset="cora", model=["gcn", "gat"], seed=[0, 1])


def search_space(trial):
    return {
        "lr": trial.suggest_categorical("lr", [1e-3, 5e-3, 1e-2]),
        "hidden_size": trial.suggest_categorical("hidden_size", [32, 64, 128]),
        "dropout": trial.suggest_uniform("dropout", 0.5, 0.8),
    }


experiment(dataset="cora", model="gcn", seed=[1, 2], search_space=search_space, n_trials=3)
```

Experiment API结果

```python
from cogdl import experiment

# basic usage
experiment(dataset="cora", model="gcn")
```

✓ 11.9s Python

```
Namespace(activation='relu', actnn=False, checkpoint_path='./checkpoints/model.pt', cpu=False, cpu_inference=False, dataset=['cora'], devices=[0], distributed=False, dropout=0.5, dw='node_classification_dw', epochs=500, eval_step=1, hidden_size=64, load_emb_path=None, local_rank=0, log_path='.', logger=None, lr=0.01, master_addr='localhost', master_port=13425, max_epoch=None, model=['gcn'], mw='node_classification_mw', n_trials=3, n_warmup_steps=0, no_test=False, norm=None, nstage=1, num_classes=None, num_features=None, num_layers=2, patience=100, progress_bar='epoch', project='cogdl-exp', residual=False, resume_training=False, rp_ratio=1, save_emb_path=None, seed=[1], split=[0], unsup=False, use_best_config=False, weight_decay=0)

|-----------------------------------------------------------------------------------------|
    *** Running (`cora`, `gcn`, `node_classification_dw`, `node_classification_mw`)
|-----------------------------------------------------------------------------------------|
Downloading https://cloud.tsinghua.edu.cn/d/6808093f7f8042bfa1f0/files/?p=%2Fcora.zip&dl=1
unpacking cora.zip
Processing...
Done!
Model Parameters: 92231

Epoch: 146, train_loss:  0.0109, val_acc:  0.7860:  29%|▉▉▉       | 147/500 [00:01<00:04, 74.92it/s]

Using time  0.0039
Saving 47-th model to ./checkpoints/model.pt ...
Loading model from ./checkpoints/model.pt ...
{'test_acc': 0.817, 'val_acc': 0.798}
| Variant         | test_acc      | val_acc       |
|-----------------|---------------|---------------|
| ('cora', 'gcn') | 0.8170±0.0000 | 0.7980±0.0000 |
```

节点分类任务（图神经网络）

- 学术引用数据集：
  - Cora, Citeseer, Pubmed
- 两类图神经网络模型：
  - 监督模型：GCN, GAT, GRAND, …
  - 无监督模型：MVGRL, DGI

Rank	Method	Cora	Citeseer	Pubmed	*Reproducible*
1	GRAND [12]	84.8	75.1	82.4	*Yes*
2	GCNII [7]	85.1	71.3	80.2	*Yes*
3	MVGRL [20]	83.6 ↓	73.0	80.1	*Partial*
4	APPNP [26]	**84.3** ↑	72.0	80.0	*Yes*
5	Graph-Unet [15]	83.3 ↓	71.2 ↓	79.0	*Partial*
6	GDC [27]	82.5	72.1	79.8	*Yes*
7	GAT [53]	82.9	71.0	78.9	*Yes*
8	DropEdge [38]	82.1	72.1	79.7	*Yes*
9	GCN [25]	**82.3** ↑	**71.4** ↑	79.5	*Yes*
10	DGI [52]	82.0	71.2	76.5	*Yes*
11	JK-net [58]	81.8	69.5	77.7	*Yes*
12	Chebyshev [8]	79.0	69.8	68.6	*Yes*

图分类任务

- 两类不同类型的图数据
  - 生物信息数据集：MUTAG, PTC, NCI1, PROTEINS
  - 社交网络数据集：IMDB-B/M, COLLAB, REDDIT-B
- 两类图分类模型
  - 无监督模型：InfoGraph, graph2vec, DGK
  - 有监督模型：GIN, DiffPool, SortPool, …

Algorithm	MUTAG	PTC	NCI1	PROTEINS	IMDB-B	IMDB-M	COLLAB	REDDIT-B	*Reproducible*
GIN [57]	92.06	67.82	81.66	75.19	76.10	51.80	79.52	83.10 ↓	*Yes*
InfoGraph [42]	88.95	60.74	76.64	73.93	74.50	51.33	79.40	76.55	*Yes*
DiffPool [62]	85.18	58.00	69.09	75.30	72.50	50.50	79.27	81.20	*Yes*
SortPool [67]	87.25	62.04	**73.99** ↑	74.48	75.40	50.47	**80.07** ↑	78.15	*Yes*
graph2vec [31]	83.68	54.76 ↓	71.85	73.30	73.90	52.27	**85.58** ↑	91.77	*Yes*
PATCHY_SAN [32]	86.12	61.60	69.82	75.38	**76.00** ↑	46.40	74.34	60.61	*Yes*
DGCNN [56]	83.33	56.72	65.96	66.75	71.60	49.20	77.45	86.20	*Yes*
SAGPool [28]	71.73 ↓	59.92	72.87	74.03	74.80	51.33	/	89.21	*Yes*
DGK [59]	85.58	57.28	/	72.59	55.00 ↓	40.40 ↓	/	/	*Partial*

自定义GNN模型

```python
class GCN(BaseModel):
    def __init__(self, in_feats, hidden_size, out_feats, dropout):
        super(GCN, self).__init__()
        self.conv1 = GCNLayer(in_feats, hidden_size)
        self.conv2 = GCNLayer(hidden_size, out_feats)
        self.dropout = nn.Dropout(dropout)

    def forward(self, graph):
        graph.sym_norm()
        h = graph.x
        h = F.relu(self.conv1(graph, self.dropout(h)))
        h = self.conv2(graph, self.dropout(h))
        return h

model = GCN(in_feats=1433, hidden_size=64, out_feats=7, dropout=0.1)
experiment(dataset="cora", model=model, dw="node_classification_dw", mw="node_classification_mw")
```

```
|-----------------------------------------------------------------------------------------|
    *** Running (`cora`, `GCN`, `node_classification_dw`, `node_classification_mw`)
|-----------------------------------------------------------------------------------------|
Epoch: 228, train_loss:  0.1349, val_acc:  0.7860:  46%|[redacted]  | 229/500 [00:01<00:01, 157.35it/s]
Saving 129-th model to ./checkpoints/model.pt ...
Loading model from ./checkpoints/model.pt ...
{'test_acc': 0.813, 'val_acc': 0.8}
| Variant        | test_acc       | val_acc        |
|----------------|----------------|----------------|
| ('cora', GCN)  | 0.8130±0.0000  | 0.8000±0.0000  |
```

自定义数据集

```python
from cogdl.experiments import experiment
from cogdl.datasets import NodeDataset, generate_random_graph

data = generate_random_graph(num_nodes=100, num_edges=1000, num_feats=64)
dataset = NodeDataset(data=data)
experiment(dw="node_classification_dw", mw="node_classification_mw", dataset=dataset, model="gcn")
```

```
|-----------------------------------------------------------------------------------------------|
    *** Running (`data.pt`, `gcn`, `node_classification_dw`, `node_classification_mw`)
|-----------------------------------------------------------------------------------------------|
Epoch: 498, train_loss:  80.6607, val_acc:  0.7000: 100%|██████████████████████████████████| 499/500 [00:03<00:00, 162.90it/s]
Saving 399-th model to ./checkpoints/model.pt ...
Loading model from ./checkpoints/model.pt ...
{'test_acc': 0.5666666666666667, 'val_acc': 0.75}
| Variant          | test_acc      | val_acc       |
|------------------|---------------|---------------|
| (data.pt, 'gcn') | 0.5667±0.0000 | 0.7500±0.0000 |
```

Pipeline API

- 传入应用名、模型、（参数）

```
import numpy as np
from cogdl import pipeline

# build a pipeline for generating embeddings
# pass model name with its hyper-parameters to this API
generator = pipeline("generate-emb", model="prone")

# generate embedding by an unweighted graph
edge_index = np.array([[0, 1], [0, 2], [0, 3], [1, 2], [2, 3]])
outputs = generator(edge_index)
print(outputs)

# build a pipeline for generating embeddings using unsupervised GNNs
# pass model name and num_features with its hyper-parameters to this API
generator = pipeline("generate-emb", model="dgi", num_features=8, hidden_size=4)
outputs = generator(edge_index, x=np.random.randn(4, 8))
print(outputs)
```

- prone
- netmf
- netsmf
- deepwalk
- line
- node2vec
- hope
- sdne
- grarep
- dngr
- spectral

- unsup_graphsage
- dgi
- mvgrl
- grace

图深度学习面临的挑战

- 超大规模的图数据
  - 模型训练开销大
  - 显卡内存受限

→ 如何高效地在大规模图数据上训练GNN

- 深层图神经网络难以训练
  - 过拟合/平滑现象
  - 显卡内存受限

→ 如何训练深层GNN模型

- 如何选择合适的GNN

→ 图上的自动机器学习

- 如何公平地比较GNN

→ 不同类型的图基准

图深度学习的前沿研究

- 高效的大规模图训练
  - 图稀疏存储及高效的稀疏算子
  - 图上的mini-batch训练及多卡加速
- 深层图神经网络模型
  - 相关工作介绍（从浅层到千层模型）
  - 基于压缩激活输出的训练方式
- 图上的自动机器学习
- 不同类型的图基准

回顾邻接矩阵的稀疏存储

- 坐标列表(COO)格式：
  - (row, col) 或 (row, col, value)，大小为|E|*2/3
  - [[0,0,1], [0,2,2], [1,2,3], [2,0,4], [2,1,5], [2,2,6]]
- 按行压缩(CSR)格式：
  - row_ptr：大小为 |V|+1
  - col_indices：大小为 |E|
  - value: 大小为 |E|
  - [0, 2, 3, 6], [0, 2, 2, 0, 1, 2], [1, 2, 3, 4, 5, 6]

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 0 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

CogDL中的Graph存储

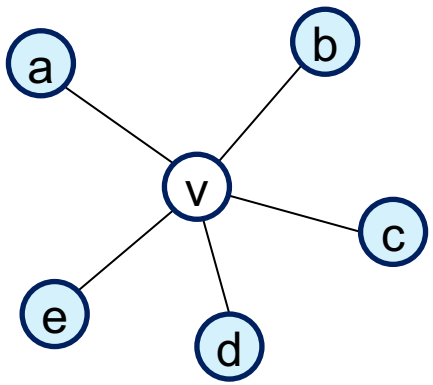
class Graph: (defined in cogdl.data)

- x：节点特征矩阵
- y：节点标签
- edge_index：COO形式的边表
- edge_weight：边权（如果有的话）
- edge_attr：边上的特征（如果有的话）
- row_ptr：CSR形式的一部分
- col_indices：CSR形式的一部分

CogDL中的Graph用法

- Graph初始化
  - g = Graph(edge_index=edge_index)
  - g.edge_weight = torch.rand(n)
- 常用的操作：
  - add_self_loops() 添加自环
  - sym_norm() GCN式的对称归一化
  - degrees() 获取节点度数
  - subgraph() 获取节点的导出子图
  - …

回顾图卷积网络 GCN



第k层的训练参数

非线性的激活函数 (例如, ReLU)

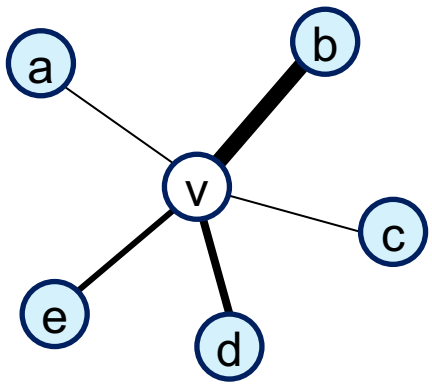
$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in N(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

节点v第k层的表示

节点v的邻居节点集合

1. Kipf et al. Semi-supervised Classification with Graph Convolutional Networks. ICLR 2017

回顾图注意力网络 GAT



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

GAT

$$\boldsymbol{h}_v^k = \sigma(\sum_{u \in \mathrm{N}(v) \cup v} \alpha_{v,u} \boldsymbol{W}^k \boldsymbol{h}_u^{k-1})$$

可学习的注意力权重

1. Velickovic et al. Graph Attention Networks. ICLR 2018

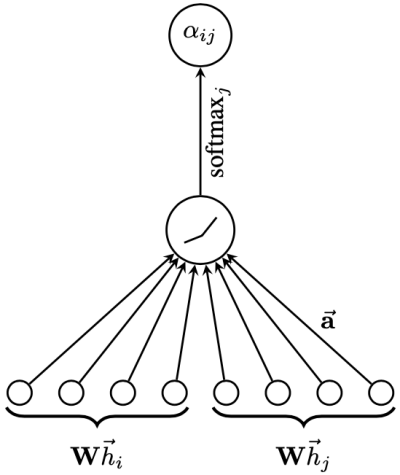
图神经网络中的底层操作

- GCN（稀疏矩阵乘法 SpMM）

$$H^{(i+1)} = AH^{(i)}W$$

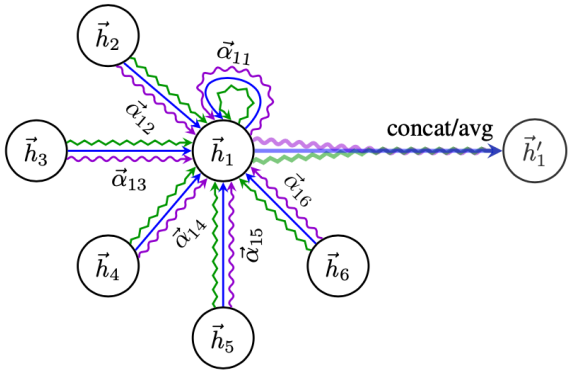
- GAT（ 注意力计算 Edge-wise-softmax）

$$a_{ij} = softmax(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}$$



- GAT（多头稀疏矩阵乘法 Multi-Head SpMM）

$$h_i = CONCAT\left(\sigma\left(\sum_{j \in N_i} a_{ij}^k \boldsymbol{W}^k h_j\right)\right)$$



CogDL中GCN/GAT层

$$H^{(i+1)} = AH^{(i)}\boldsymbol{W}^{(\mathrm{i})}$$

$$a_{ij} = softmax(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in N_i}\exp(e_{ik})}$$

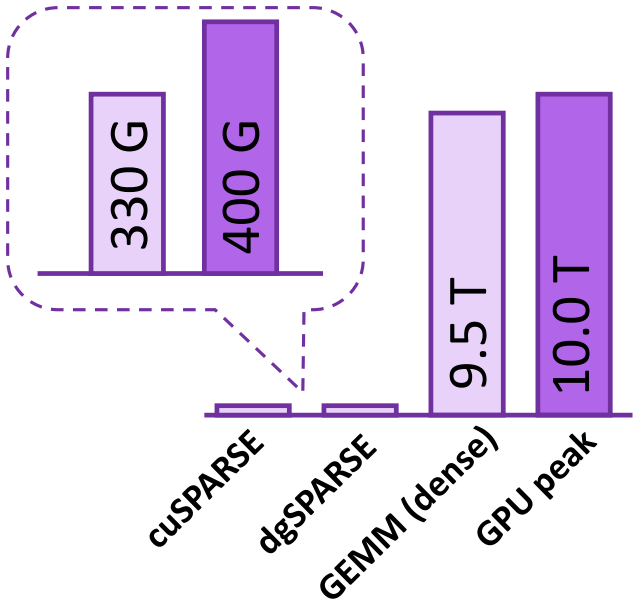
$$h_i = CONCAT\left(\sigma\left(\sum_{j\in N_i} a_{ij}^k \boldsymbol{W}^k h_j\right)\right)$$

```python
class GCNLayer(nn.Module):
    """
    Simple GCN layer, similar to https://arxiv.o
    """

    def __init__(self, in_features, out_features

    def reset_parameters(self):…

    def forward(self, graph, x):
        support = torch.mm(x, self.weight)
        out = spmm(graph, support)
```

```python
class GATLayer(nn.Module):
    """
    Sparse version GAT layer, similar to https://arxiv.org/ab
    """

    def __init__(self, in_features, out_features, nhead=1, al

    def reset_parameters(self):…

    def forward(self, graph, x):
        h = torch.matmul(x, self.W)
        edge_score = self.compute_edge_score(graph, x, h)
        # edge_attention: E * H
        edge_attention = mul_edge_softmax(graph, edge_score)
        edge_attention = self.dropout(edge_attention)

        out = mh_spmm(graph, edge_attention, h)
```

高效的稀疏算子

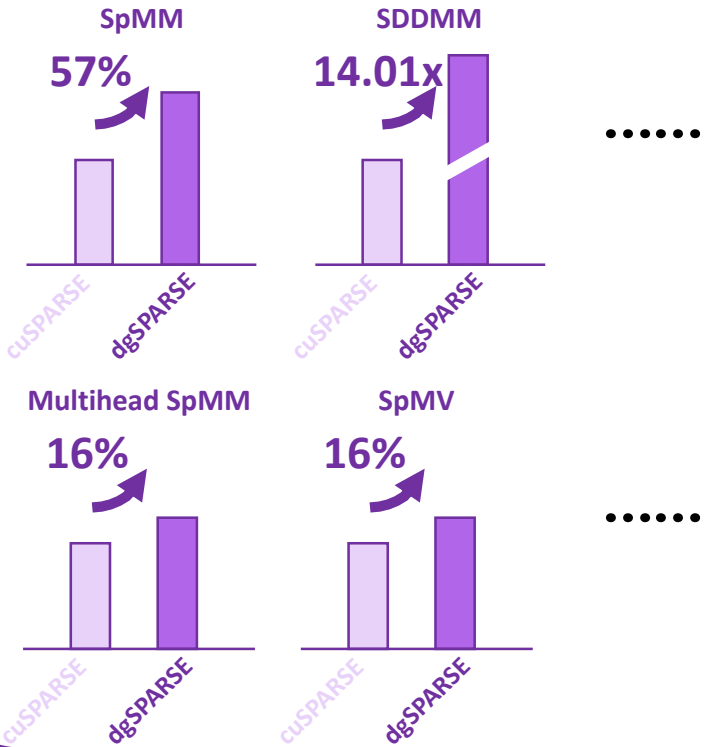
缺少高效稀疏算子



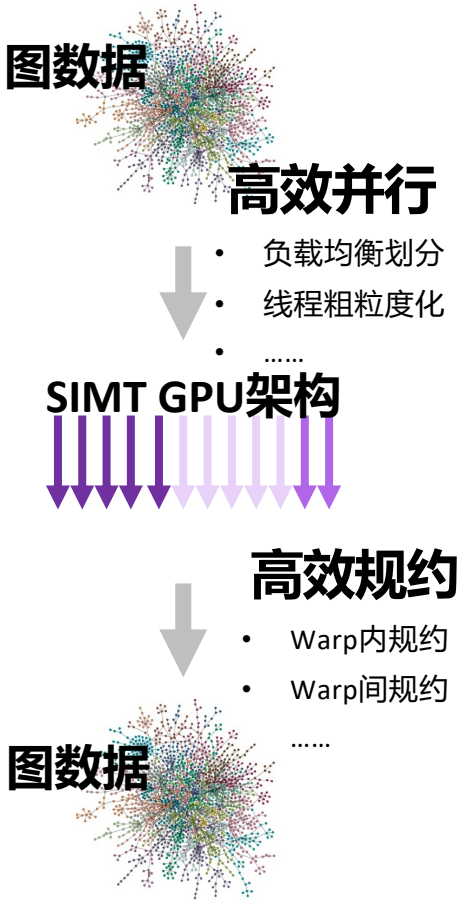
稀疏算力与GPU峰值算力存在巨大鸿沟
如cuSPARSE商用库性能存在优化空间

dgSPARSE, Deep Graph SPARSE

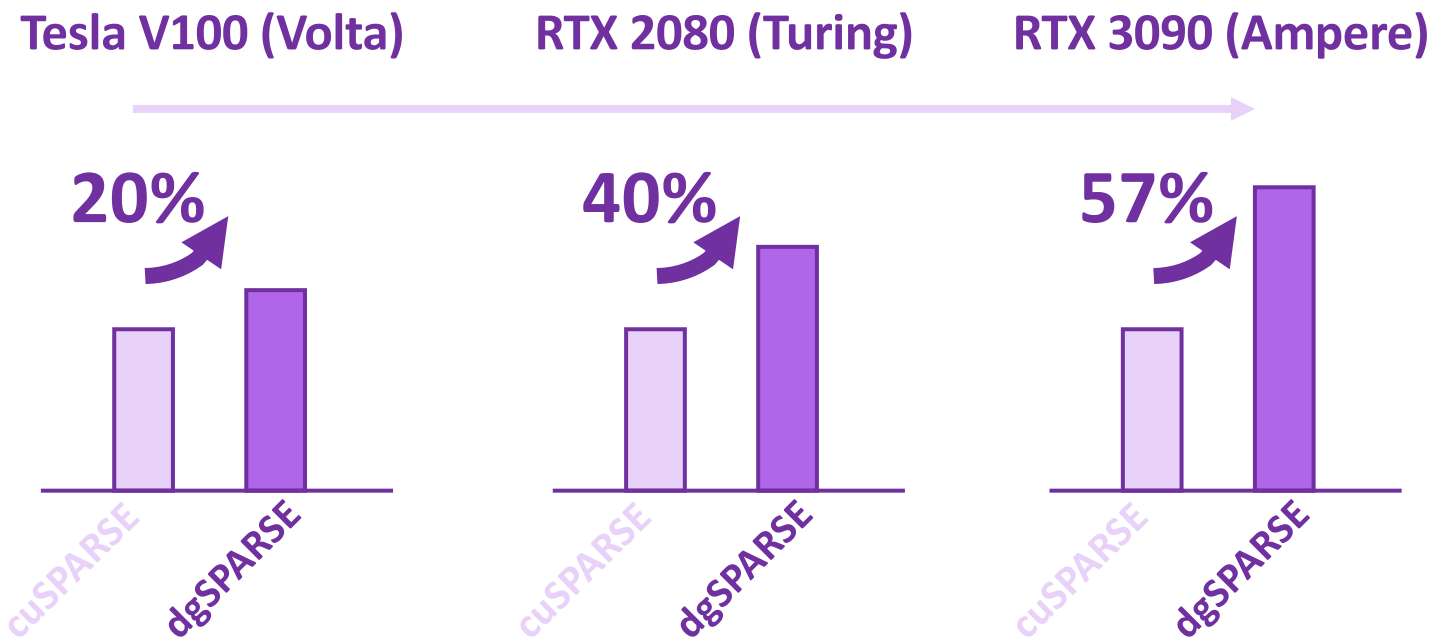
高效稀疏算子GPU实现



Deep Graph Sparse (dgSPARSE)库



简单但有效的设计思想，针对GPU的SIMT架构优化



GCN/GAT模型的性能对比

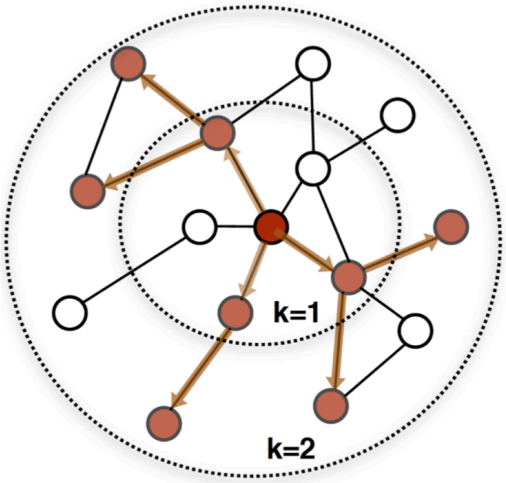
- 在不同GPU上对比GCN模型的性能
- 实验设置：2层GCN/GAT，hidden size=128
- 由dgSPAESE提供算子支持

Model	GPU	Dataset	Tranining per epoch (s)				Inference per epoch (s)			
			torch	PyG	DGL	CogDL	torch	PyG	DGL	CogDL
GCN	2080Ti (11G)	Flickr	0.025	0.0084	0.012	0.085	0.012	0.0034	0.007	0.0035
		Reddit	0.445	0.122	0.102	0.081	0.218	0.045	0.049	0.039
		Yelp	0.412	0.151	0.151	0.110	0.191	0.053	0.063	0.040
	3090 (24G)	Flickr	0.017	0.006	0.008	0.007	0.008	0.002	0.004	0.002
		Reddit	0.263	0.062	0.060	0.050	0.127	0.022	0.0314	0.022
		Yelp	0.230	0.081	0.080	0.062	0.106	0.029	0.036	0.023
GAT	2080Ti (11G)	PubMed	0.017	0.016	0.011	0.012	0.004	0.006	0.004	0.003
		Flickr	0.082	0.090	0.047	0.056	0.023	0.030	0.019	0.014
		⋆Reddit	__‡	__‡	0.406	0.537	__‡	__‡	0.163	0.086
	3090 (24G)	PubMed	0.043	0.011	0.011	0.016	0.004	0.004	0.003	0.002
		Flickr	0.097	0.059	0.033	0.044	0.021	0.024	0.013	0.009
		Reddit	0.671	__‡	0.301	0.373	0.112	__‡	0.113	0.088
		Yelp	0.614	__‡	0.294	0.404	0.118	__‡	0.105	0.113

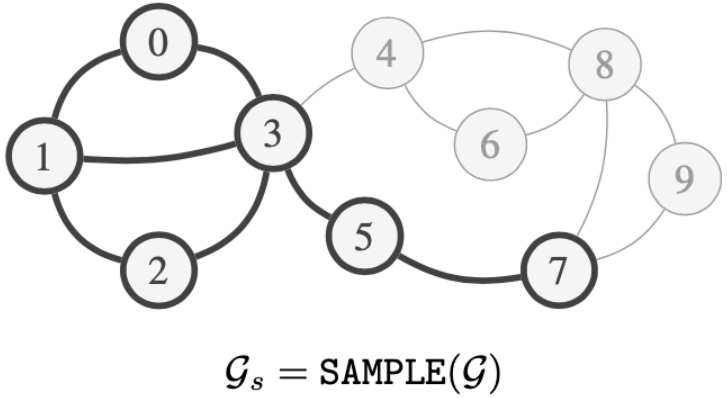
* supported by dgSPARSE: https://github.com/dgSPARSE

在超大规模图上训练GNNs

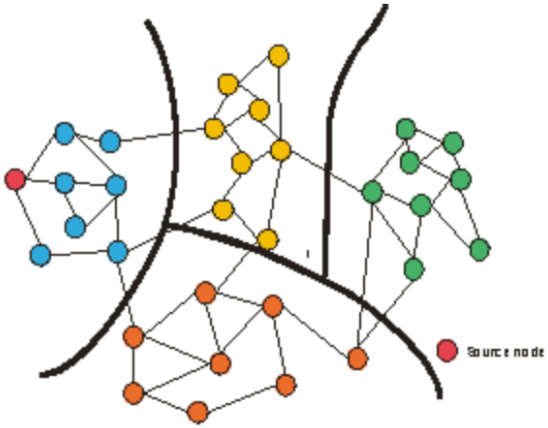
- 超大规模的社交网络、推荐系统 （亿级节点）
- 超大规模训练的主要挑战在于图神经网络的空间复杂度高
- 可以通过mini-batch的方式来训练图神经网络



Neighbor Sampling
(NeurIPS ’17)



GraphSAINT
(ICLR ’20)

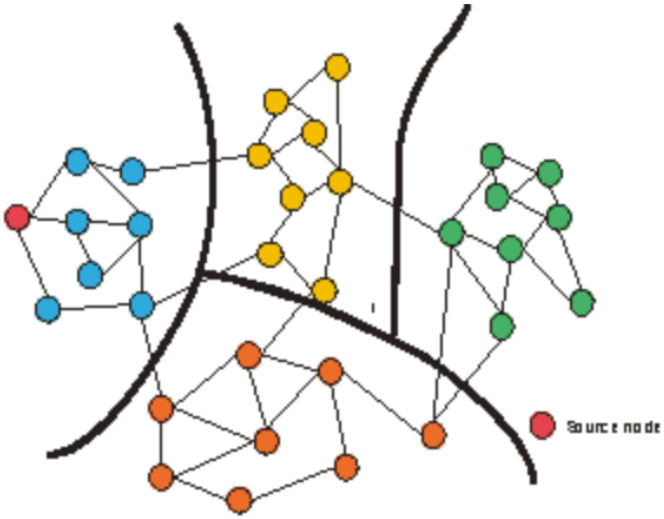


ClusterGCN
(KDD ’19)

1. Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In NeurIPS ’17.
2. Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In KDD ’19.
3. Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In ICLR ’20.

ClusterGCN

- 图划分METIS
- 通过mini-batch的方式来训练GNNs



- 显存占用: O(NLF) => O(bLF)

	GCN [9]	Vanilla SGD	GraphSAGE [5]	FastGCN [1]	VR-GCN [2]	Cluster-GCN
Time complexity	$O(L\|\|A\|\|_0F + LNF^2)$	$O(d^LNF^2)$	$O(r^LNF^2)$	$O(rLNF^2)$	$O(L\|\|A\|\|_0F + LNF^2 + r^LNF^2)$	$O(L\|\|A\|\|_0F + LNF^2)$
Memory complexity	$O(LNF + LF^2)$	$O(bd^LF + LF^2)$	$O(br^LF + LF^2)$	$O(brLF + LF^2)$	$O(LNF + LF^2)$	$O(bLF + LF^2)$

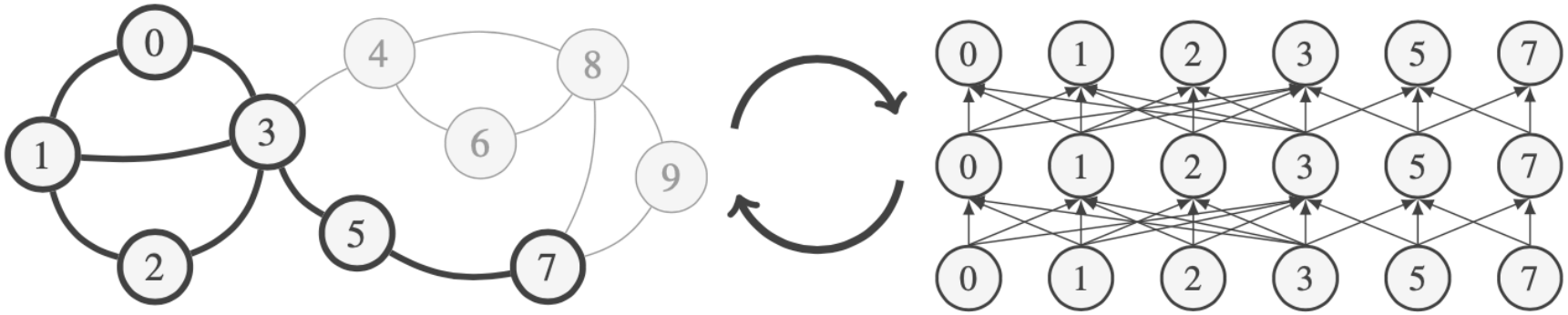
	Time		Memory		Test F1 score	
	VRGCN	Cluster-GCN	VRGCN	Cluster-GCN	VRGCN	Cluster-GCN
Amazon2M (2-layer)	337s	1223s	7476 MB	2228 MB	89.03	89.00
Amazon2M (3-layer)	1961s	1523s	11218 MB	2235 MB	90.21	90.21
Amazon2M (4-layer)	N/A	2289s	OOM	2241 MB	N/A	90.41

GraphSAINT

- 节点v的表示：

$$\zeta_v^{(\ell+1)} = \sum_{u\in\mathcal{V}} \frac{\widetilde{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} \left(\boldsymbol{W}^{(\ell)}\right)^{\mathsf{T}} \boldsymbol{x}_u^{(\ell)} \mathbb{1}_{u|v} = \sum_{u\in\mathcal{V}} \frac{\widetilde{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} \tilde{\boldsymbol{x}}_u^{(\ell)} \mathbb{1}_{u|v}$$

- 无偏的采样器
  - 基于节点/边/随机游走的采样器



$\mathcal{G}_s = \texttt{SAMPLE}(\mathcal{G})$ Full GCN on $\mathcal{G}_s$

GraphSAINT 表现

Dataset	Nodes	Edges	Degree	Feature	Classes	Train / Val / Test
PPI	14,755	225,270	15	50	121 (m)	0.66 / 0.12 / 0.22
Flickr	89,250	899,756	10	500	7 (s)	0.50 / 0.25 / 0.25
Reddit	232,965	11,606,919	50	602	41 (s)	0.66 / 0.10 / 0.24
Yelp	716,847	6,977,410	10	300	100 (m)	0.75 / 0.10 / 0.15
Amazon	1,598,960	132,169,734	83	200	107 (m)	0.85 / 0.05 / 0.10
PPI (large version)	56,944	818,716	14	50	121 (m)	0.79 / 0.11 / 0.10

Method	PPI	Flickr	Reddit	Yelp	Amazon
GCN	0.515±0.006	0.492±0.003	0.933±0.000	0.378±0.001	0.281±0.005
GraphSAGE	0.637±0.006	0.501±0.013	0.953±0.001	0.634±0.006	0.758±0.002
FastGCN	0.513±0.032	0.504±0.001	0.924±0.001	0.265±0.053	0.174±0.021
S-GCN	0.963±0.010	0.482±0.003	0.964±0.001	0.640±0.002	—‡
AS-GCN	0.687±0.012	0.504±0.002	0.958±0.001	—‡	—‡
ClusterGCN	0.875±0.004	0.481±0.005	0.954±0.001	0.609±0.005	0.759±0.008
GraphSAINT-Node	0.960±0.001	0.507±0.001	0.962±0.001	0.641±0.000	0.782±0.004
GraphSAINT-Edge	**0.981**±0.007	0.510±0.002	**0.966**±0.001	**0.653**±0.003	0.807±0.001
GraphSAINT-RW	**0.981**±0.004	**0.511**±0.001	**0.966**±0.001	**0.653**±0.003	**0.815**±0.001
GraphSAINT-MRW	0.980±0.006	0.510±0.001	0.964±0.000	0.652±0.001	0.809±0.001

GNN的多卡并行训练

GNN的多卡并行训练:

采样策略 + PyTorch DDP

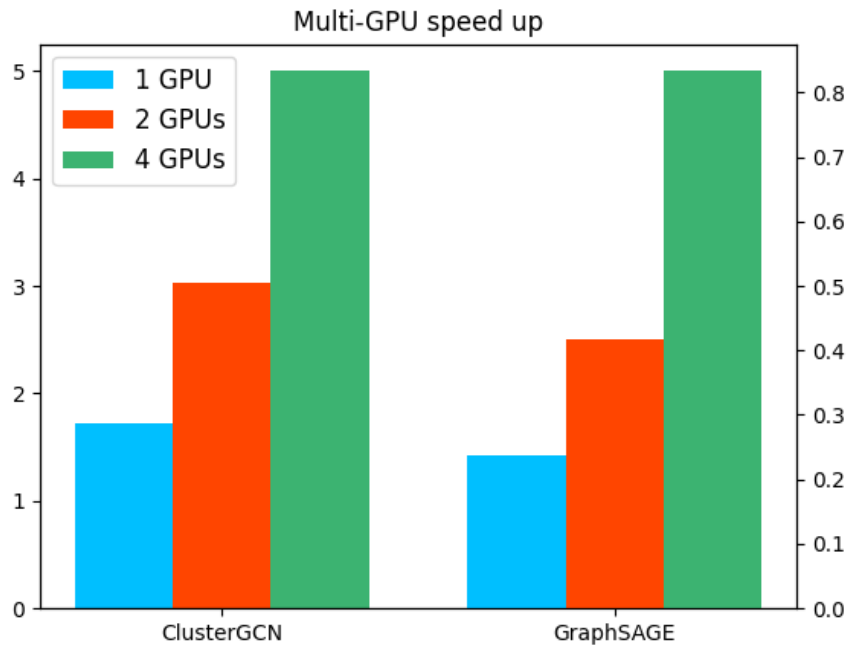
✓ CogDL中的采样策略

[+] NeighborSampling

[+] ClusterGCN

[+] GraphSAINT

✓ 4 GPUs ~ 3x↑ 加速



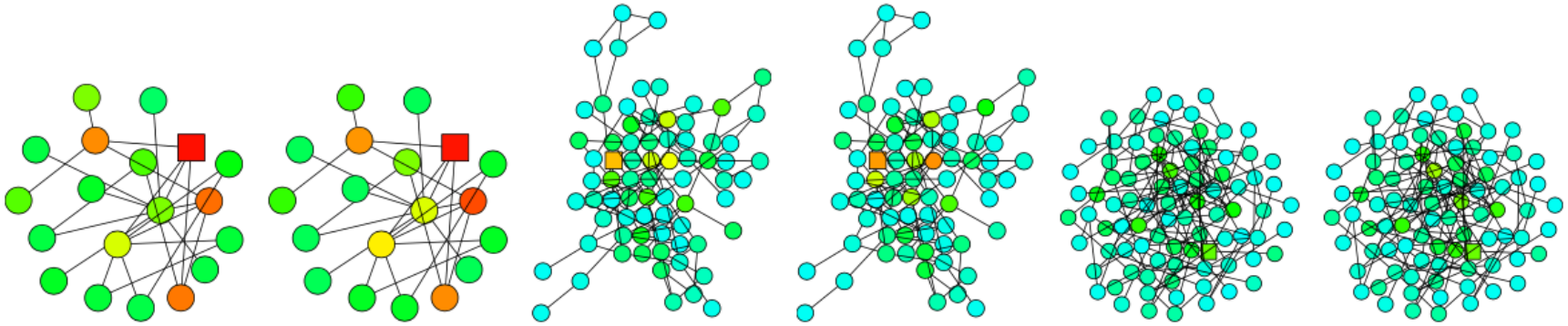
Usage: python scripts/train.py --model gcn --dataset reddit --dw cluster_dw **--distributed --devices 0 1 2 3**

深层GNNs的相关工作

- JKNet (ICML’18)
- APPNP (ICLR’19)
- GCNII (ICML’20)
- DeeperGCN (Arxiv 2020)
- RevGNN (ICML’21)

JKNet (ICML’18)

- GCN影响力分布和随机游走分布的关联:



(a) 2 layer GCN (b) 2 step r.w. (c) 4 layer GCN (d) 4 step r.w. (e) 6 layer GCN (f) 6 step r.w.

Figure 2. Influence distributions of GCNs and random walk distributions starting at the square node

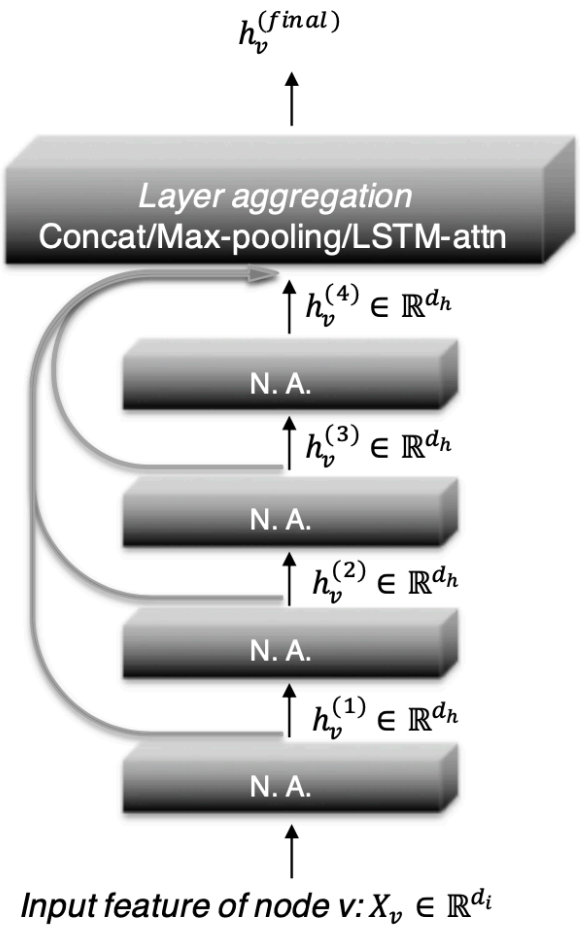
- 很难确定传播的步数!
- 层聚合（Layer aggregation）！

Xu, Keyulu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In ICML’18.

JKNet (ICML'18)

- 将所有层的输出聚合：
  - 拼接（Concatenation）
  - 最大池化（Max-pooling）
  - 长短时记忆网络（LSTM）

Model	Citeseer	Model	Cora
GCN (2)	77.3 (1.3)	GCN (2)	88.2 (0.7)
GAT (2)	76.2 (0.8)	GAT (3)	87.7 (0.3)
JK-MaxPool (1)	77.7 (0.5)	JK-Maxpool (6)	**89.6** (0.5)
JK-Concat (1)	**78.3** (0.8)	JK-Concat (6)	89.1 (1.1)
JK-LSTM (2)	74.7 (0.9)	JK-LSTM (1)	85.8 (1.0)



APPNP (ICLR'19)

- Personalized PageRank (PPR):

$$\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) + \alpha\boldsymbol{i}_x$$

- 通过求解上述方程，可以得到:

$$\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = \alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1}\boldsymbol{i}_x$$

- Personalized propagation of neural predictions (PPNP):
  - 通过各自的特征生成预测结果，再将结果通过PPR的方式进行传播

$$\boldsymbol{Z}_{\text{PPNP}} = \text{softmax}\left(\alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1}\boldsymbol{H}\right), \qquad \boldsymbol{H}_{i,:} = f_\theta(\boldsymbol{X}_{i,:}),$$

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR'19.

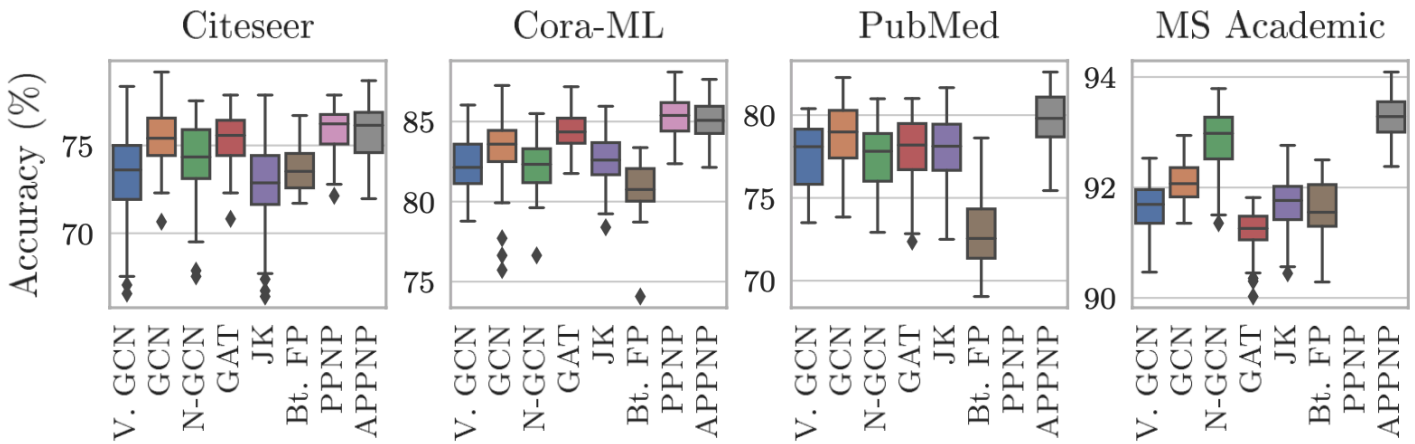
APPNP (ICLR'19)

- PPNP 需要 $O(n^2)$ 时间去计算 PPR 矩阵:

$$\mathbf{\Pi}_{\text{ppr}} = \alpha(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}})^{-1}$$

- Approximate PPNP (APPNP):
  - 通过多次迭代传播来近似左乘PPR矩阵的效果：

$$\boldsymbol{Z}^{(0)} = \boldsymbol{H} = f_\theta(\boldsymbol{X}),$$
$$\boldsymbol{Z}^{(k+1)} = (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(k)} + \alpha\boldsymbol{H},$$
$$\boldsymbol{Z}^{(K)} = \text{softmax}\left((1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(K-1)} + \alpha\boldsymbol{H}\right),$$



GCNII (ICML’20)

- 初始残差连接（Initial residual connection）：
  - 与APPNP类似 (但APPNP是一个浅层的模型)
  - 将平滑后的表示和初始特征进行组合

$$(1-\alpha)\tilde{\boldsymbol{P}}\boldsymbol{H}^{(\ell)} + \alpha\boldsymbol{H}^{(0)}$$

- 恒等映射（Identity mapping）：
  - 借鉴ResNet的想法
  - 将参数矩阵与单位阵进行组合

$$(1-\beta_\ell)\mathbf{I}_n + \beta_\ell\mathbf{W}^{(\ell)}$$

Chen, Ming, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. "Simple and deep graph convolutional networks." In ICML’20.

GCNII结果

- GCNII (结合上述两种技术)

$$\mathbf{H}^{(\ell+1)}=\sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}+\alpha_\ell\mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}\right)\right)$$

- GCNII*: 给PH和$H^{(0)}$分配不同的参数矩阵W_1, W_2

$$\mathbf{H}^{(\ell+1)}=\sigma\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\left((1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}_1^{(\ell)}\right)+\right.$$
$$\left.+\alpha_\ell\mathbf{H}^{(0)}\left((1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}_2^{(\ell)}\right)\right).$$

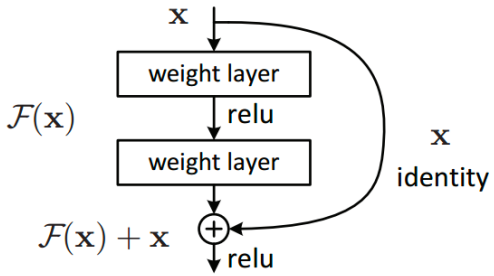
Method	Cora	Cite.	Pumb.	Cham.	Corn.	Texa.	Wisc.
GCN	85.77	73.68	88.13	28.18	52.70	52.16	45.88
GAT	86.37	74.32	87.62	42.93	54.32	58.38	49.41
Geom-GCN-I	85.19	**77.99**	90.05	60.31	56.76	57.58	58.24
Geom-GCN-P	84.93	75.14	88.09	60.90	60.81	67.57	64.12
Geom-GCN-S	85.27	74.71	84.75	59.96	55.68	59.73	56.67
APPNP	87.87	76.53	89.40	54.3	73.51	65.41	69.02
JKNet	85.25 (16)	75.85 (8)	88.94 (64)	60.07 (32)	57.30 (4)	56.49 (32)	48.82 (8)
JKNet(Drop)	87.46 (16)	75.96 (8)	89.45 (64)	62.08 (32)	61.08 (4)	57.30 (32)	50.59 (8)
Incep(Drop)	86.86 (8)	76.83 (8)	89.18 (4)	61.71 (8)	61.62 (16)	57.84 (8)	50.20 (8)
GCNII	**88.49** (64)	77.08 (64)	89.57 (64)	60.61 (8)	74.86 (16)	69.46 (32)	74.12 (16)
GCNII*	88.01 (64)	77.13 (64)	**90.30** (64)	**62.48** (8)	**76.49** (16)	**77.84** (32)	**81.57** (16)

DeeperGCN

- 广义的聚合函数（Mean-Max聚合）
  - 找到一种介于Mean和Max之间的更好的聚合操作
- SoftMax_Agg: $\sum_{u\in\mathcal{N}(v)} \frac{exp(\beta \mathbf{m}_{vu})}{\sum_{i\in\mathcal{N}(v)} exp(\beta \mathbf{m}_{vi})}$
  - $\lim_{\beta\to 0} \text{SoftMax\_Agg}_\beta = \text{Mean}$
  - $\lim_{\beta\to \infty} \text{SoftMax\_Agg}_\beta = \text{Max}$
- PowerMean: $\left(\frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} \mathbf{m}_{vu}^p\right)^{1/p}$
  - $\text{PowerMean\_Agg}_{p=1} = \text{Mean}$
  - $\lim_{p\to\infty} \text{PowerMean\_Agg}_p = \text{Max}$

DeeperGCN

- 更好的残差连接方式:
  - BN/LN → ReLU → GraphConv → Addition



	PlainGCN			ResGCN			ResGCN+		
#layers	Sum	Mean	Max	Sum	Mean	Max	Sum	Mean	Max
3	0.824	0.793	**0.834**	0.824	0.786	0.824	0.830	0.792	0.829
7	0.811	0.796	0.823	0.831	0.803	0.843	0.841	0.813	0.845
14	0.821	0.802	0.824	0.843	0.808	0.850	0.840	0.813	0.848
28	0.819	0.794	0.825	0.837	0.807	0.847	0.845	0.819	0.855
56	0.824	0.808	0.825	0.841	0.813	**0.851**	0.843	0.810	0.853
112	0.823	0.810	0.824	0.840	0.805	**0.851**	0.853	0.820	**0.858**
avg.	0.820	0.801	0.826	0.836	0.804	0.844	0.842	0.811	**0.848**

	SoftMax_Agg			PowerMean_Agg		
#layers	Fixed	β	$\beta\&s$	Fixed	p	$p\&s$
3	0.821	0.832	0.837	0.802	0.818	0.838
7	0.835	0.846	0.848	0.797	0.841	0.851
14	0.833	0.849	0.851	0.814	0.840	0.849
28	0.845	0.852	0.853	0.816	0.847	**0.854**
56	0.849	**0.860**	0.854	0.818	0.846	-
112	0.844	0.858	0.858	0.824	-	-
avg.	0.838	**0.850**	**0.850**	0.812	0.838	**0.848**

从112层到1000层GNN

- DeeperGCN: All You Need to Train Deeper GCNs
  - Li et al., June 2020
  - GNN block设计(Normalization → ReLU → GraphConv → Addition)
  - 广义消息聚合函数
  - 最高达到112层 (受到GPU显存的限制)

- Training Graph Neural Networks with 1000 Layers
  - Li et al., ICML 2021
  - **可逆连接！**
  - 最高达到1000层

Li, Guohao, et al. "Deepergcn: All you need to train deeper gcns." *arXiv preprint arXiv:2006.07739* (2020).
Li, Guohao, et al. "Training Graph Neural Networks with 1000 Layers." arXiv preprint arXiv:2106.07476 (2021).

GNN中的反向传播

- 考虑图卷积操作：

$$\boldsymbol{H}^{(l+1)} = \boldsymbol{A}\boldsymbol{H}^{(l)}\boldsymbol{W}_l$$

- 那么反向传播的梯度为：

$$\nabla_{\boldsymbol{H}^{(l)}} = \boldsymbol{A}^T \nabla_{\boldsymbol{H}^{(l+1)}} \boldsymbol{W}_l^T$$

$$\nabla_{\boldsymbol{W}_l} = \boldsymbol{H}^{(l)^T} \boldsymbol{A}^T \nabla_{\boldsymbol{H}^{(l+1)}}$$

$$\nabla_{\boldsymbol{A}} = \nabla_{\boldsymbol{H}^{(l+1)}} \boldsymbol{W}_l^T \boldsymbol{H}^{(l)^T}$$

- 为了能够进行反向传播，我们需要保存每一层的输出$\boldsymbol{H}^{(l)}$，每一层占用$O(ND)$的显存。

1000层GNN (ICML'21)

- 挑战: 训练需要占用$O(LND)$显存, 与层数成正比！
- **可逆连接!**
- (起源于NeurIPS 2017: The reversible residual network: Backpropagation without storing activations)
- 分块可逆GNN block:

$$\langle X_1, X_2, ..., X_C \rangle \mapsto \langle X'_1, X'_2, ..., X'_C \rangle$$

$$X'_0 = \sum_{i=2}^{C} X_i$$

$$X'_i = f_{w_i}(X'_{i-1}, A, U) + X_i,\ i \in \{1, \cdots, C\}$$

$$X_i = X'_i - f_{w_i}(X'_{i-1}, A, U),\ i \in \{2, \cdots, C\}$$

$$X'_0 = \sum_{i=2}^{C} X_i$$

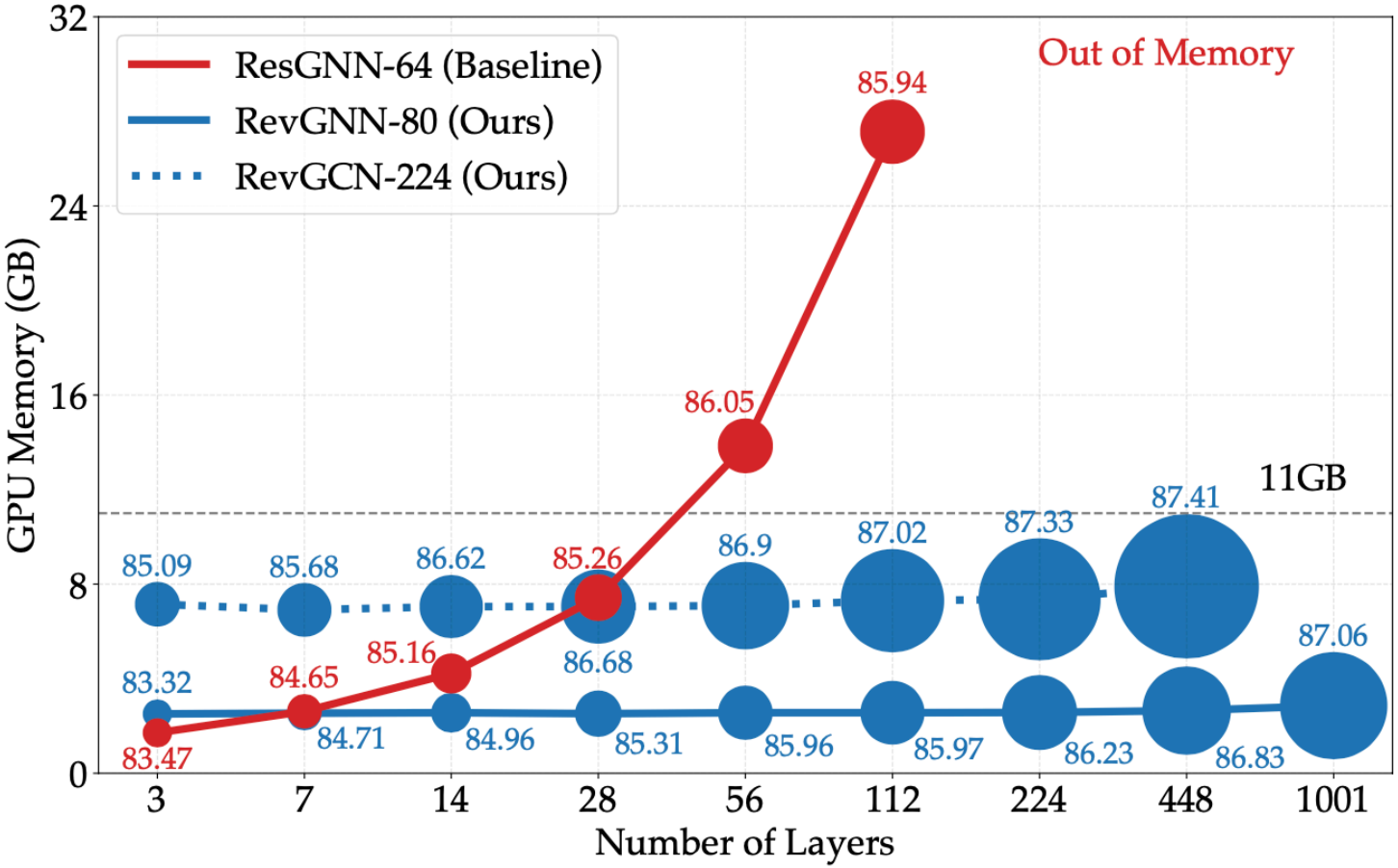
$$X_1 = X'_1 - f_{w_1}(X'_0, A, U).$$

RevGNN的表现

- ogbn-proteins数据集：
  - 节点代表蛋白质
  - 边代表蛋白质之间的关联（同源、共同表达等）

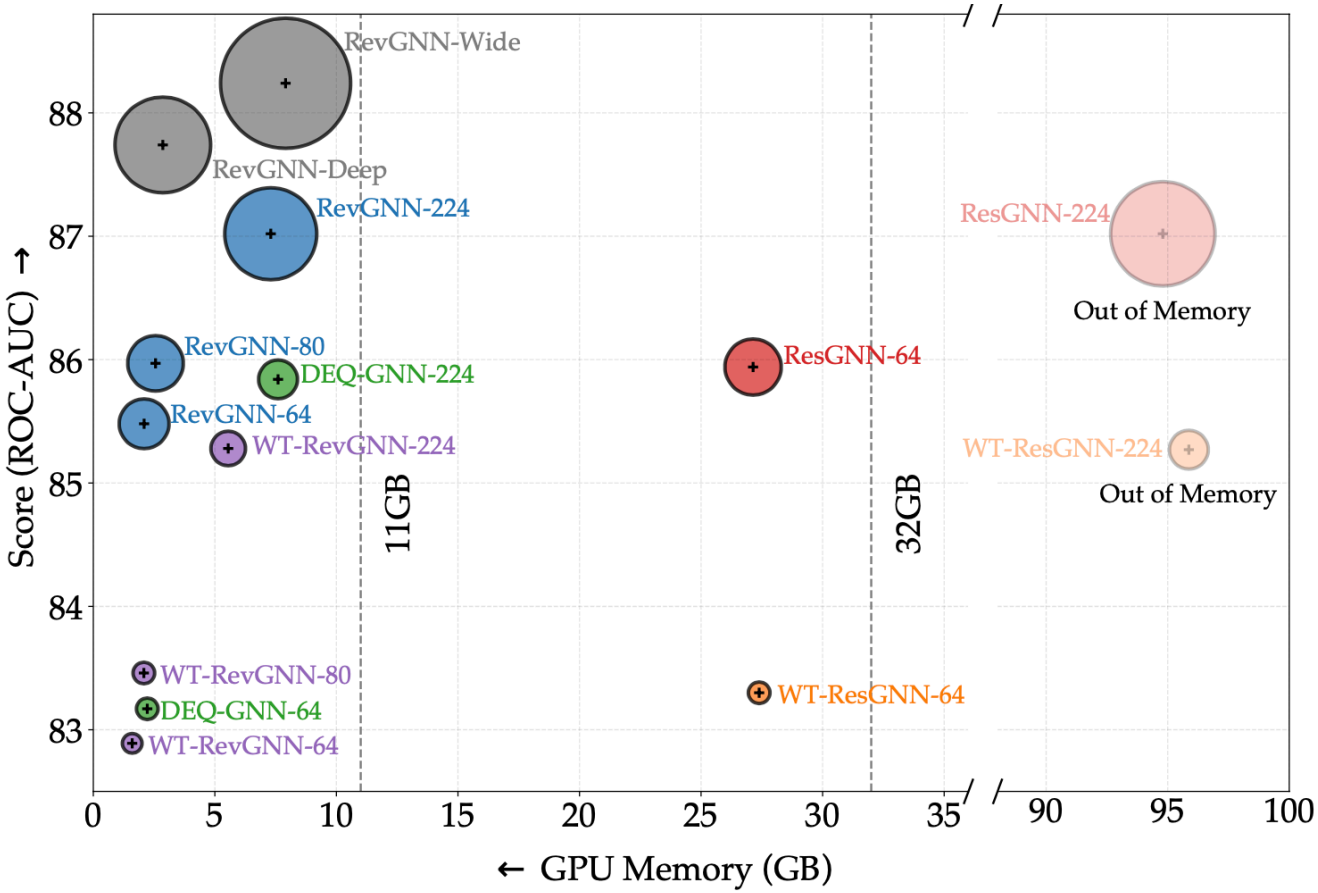
Model	ROC-AUC ↑	Mem ↓	Params
GCN (Kipf & Welling)	72.51 ± 0.35	4.68	96.9k
GraphSAGE (Hamilton et al.)	77.68 ± 0.20	3.12	193k
DeeperGCN (Li et al.)	86.16 ± 0.16	27.1	2.37M
UniMP (Shi et al.)	86.42 ± 0.08	27.2	1.91M
GAT (Veličković et al.)	86.82 ± 0.21	6.74	2.48M
UniMP+CEF (Shi et al.)	86.91 ± 0.18	27.2	1.96M
Ours (RevGNN-Deep)	87.74 ± 0.13	**2.86**	20.03M
Ours (RevGNN-Wide)	**88.24** ± 0.15	7.91	68.47M

RevGNN的表现


ResGNN-64 (Baseline)
RevGNN-80 (Ours)
RevGCN-224 (Ours)
Out of Memory
11GB
GPU Memory (GB)
Number of Layers
32
24
16
8
0
3
7
14
28
56
112
224
448
1001
85.94
86.05
85.26
85.16
84.65
83.47
85.09
85.68
86.62
86.9
87.02
87.33
87.41
87.06
83.32
84.71
84.96
86.68
85.31
85.96
85.97
86.23
86.83

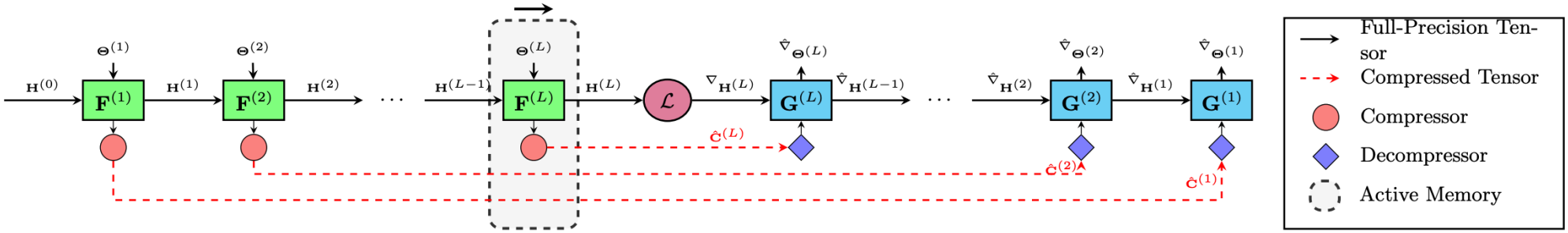
RevGNN v.s. 所有变种

- **RevGNN-Wide**
  - 448层+224隐层大小
- **RevGNN-Deep**
  - 1001层+80隐层大小
- 与这些模型对比RevGNN/ResGNN/WT/DEQ-x (x:隐层大小)
- 数据点的大小正比于模型参数的根号



ActNN：基于压缩激活输出的训练

- ActNN : Reducing Training Memory Footprint via **2-Bit Activation** Compressed Training (By Jianfei Chen, Tsinghua)
- “ActNN能够降低激活输出占用的显存12×.”
- https://github.com/ucbrise/actnn



Chen, Jianfei, et al. "ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training." In ICML’21.

ActNN实现

```
class RegularLayer:
  def forward(context, input):
    context.save_for_backward(input)
    return compute_output(input)

  def backward(context, grad_output):
    input = context.saved_tensors
    return compute_gradient(grad_output, input)

class ActivationCompressedLayer:
  def forward(context, input):
    context.save_for_backward(compress(input))
    return compute_output(input)

  def backward(context, grad_output):
    input = decompress(context.saved_tensors))
    return compute_gradient(grad_output, input)
```

ActNN表现

- 在ImageNet上的表现

Bits	32	4	3	2	1.5	1.25
FP	**77.1**	N/A	N/A	N/A	N/A	N/A
BLPA	N/A	**76.6**	Div.	Div.	N/A	N/A
ActNN (L2)	N/A	-	**77.4**	0.1	N/A	N/A
ActNN (L2.5)	N/A	-	-	**77.1**	75.9	75.1
ActNN (L3)	N/A	-	-	**76.9**	76.4	75.9

Network	Batch	Total Mem. (GB)			Act. Mem. (GB)		
		FP	ActNN (L3)	R	FP	ActNN (L3)	R
ResNet-152	32	6.01	1.18	5×	5.28	0.44	12×
	64	11.32	1.64	7×	10.57	0.88	12×
	96	OOM	2.11	/	OOM	1.32	/
	512	OOM	8.27	/	OOM	7.01	/
FCN-HR-48	2	5.76	1.39	4×	4.76	0.39	12×
	4	10.52	1.79	6×	9.52	0.79	12×
	6	OOM	2.17	/	OOM	1.18	/
	20	OOM	4.91	/	OOM	3.91	/

CogDL中的SpMM + ActNN

```python
class SPMMFunction(torch.autograd.Function):
    @staticmethod
    def forward(ctx, rowptr, colind, feat, edge_weight_csr=None, sym=False):
        if edge_weight_csr is None:
            out = spmm.csr_spmm_no_edge_value(rowptr, colind, feat)
        else:
            out = spmm.csr_spmm(rowptr, colind, edge_weight_csr, feat)
        ctx.backward_csc = (rowptr, colind, feat, edge_weight_csr, sym)
        return out

    @staticmethod
    def backward(ctx, grad_out):
        rowptr, colind, feat, edge_weight_csr, sym = ctx.backward_csc
        if edge_weight_csr is not None:
            grad_out = grad_out.contiguous()
            if sym:
                colptr, rowind, edge_weight_csc = rowptr, colind, edge_weigh
            else:
                colptr, rowind, edge_weight_csc = spmm.csr2csc(rowptr, colin
            grad_feat = spmm.csr_spmm(colptr, rowind, edge_weight_csc, grad_
            grad_edge_weight = sddmm.csr_sddmm(rowptr, colind, grad_out, fea
        else:
            if sym is False:
                colptr, rowind, edge_weight_csc = spmm.csr2csc(rowptr, colin
                grad_feat = spmm.csr_spmm_no_edge_value(colptr, rowind, grad
            else:
                grad_feat = spmm.csr_spmm_no_edge_value(rowptr, colind, grad
            grad_edge_weight = None
        return None, None, grad_feat, grad_edge_weight, None
```

```python
class ActSPMMFunction(torch.autograd.Function):
    @staticmethod
    def forward(ctx, rowptr, colind, feat, edge_weight_csr=None, sym=False):
        if edge_weight_csr is None:
            out = spmm.csr_spmm_no_edge_value(rowptr, colind, feat)
        else:
            out = spmm.csr_spmm(rowptr, colind, edge_weight_csr, feat)
        quantized = quantize_activation(feat, None)
        ctx.backward_csc = (rowptr, colind, quantized, edge_weight_csr, sym)
        ctx.other_args = feat.shape

        return out

    @staticmethod
    def backward(ctx, grad_out):
        rowptr, colind, quantized, edge_weight_csr, sym = ctx.backward_csc
        q_input_shape = ctx.other_args
        feat = dequantize_activation(quantized, q_input_shape)
        del quantized, ctx.backward_csc

        if edge_weight_csr is not None:
            grad_out = grad_out.contiguous()
            if sym:
                colptr, rowind, edge_weight_csc = rowptr, colind, edge_weight_cs
            else:
                colptr, rowind, edge_weight_csc = spmm.csr2csc(rowptr, colind, e
            grad_feat = spmm.csr_spmm(colptr, rowind, edge_weight_csc, grad_out)
            grad_edge_weight = sddmm.csr_sddmm(rowptr, colind, grad_out, feat)
```

GCN + ActNN 实验结果

- CogDL的默认参数

数据集	原始GCN	GCN + actnn
Cora	81.30 ± 0.22	81.27 ± 0.19
Citeseer	71.73 ± 0.54	71.70 ± 0.28
Pubmed	79.17 ± 0.12	79.10 ± 0.08
Flickr	50.74 ± 0.10	50.89 ± 0.04
Reddit	95.01 ± 0.02	94.89 ± 0.01

GCN + ActNN 显存占用

- 实验设置：$H = Dropout\left(ReLU\left(BN(AHW)\right)\right)$

#dataset, #layers, #hidden	Origin GCN	GCN + actnn	ratio	Idea ratio
PPI, 5, 2048	3704	420	8.8x	raw: 32*2 / (2.125*2+1) =12.2x +bn: 32*3 / (2.125*3+1) = 13.0x +bn+dropout: 32*4 / (2.125*3+2)=15.3x
PPI, 5, 2048 (+bn)	5484	539	10.2x	
PPI, 5, 2048 (+bn, +dropout)	7711	594	13.0x	
Flickr, 5, 512	1420	154	9.2x	
Flickr, 5, 512 (+bn)	2117	201	10.5x	
Flickr, 5, 512 (+bn, +dropout)	2991	223	13.4x	
Flickr, 10, 512	3178	311	10.2x	
Flickr, 10, 512 (+bn)	4747	415	11.4x	
Flickr, 10, 512 (+bn, +dropout)	6712	465	14.4x	

图上的自动机器学习

- 图上的自动机器学习
  - 针对哪种图学习任务
  - 如何定义搜索空间
  - 使用什么样的搜索策略

- CogDL中自动图机器学习用法:
  - experiment(dataset=“cora”)
  - experiment(dataset=“cora”, model=“autognn”)

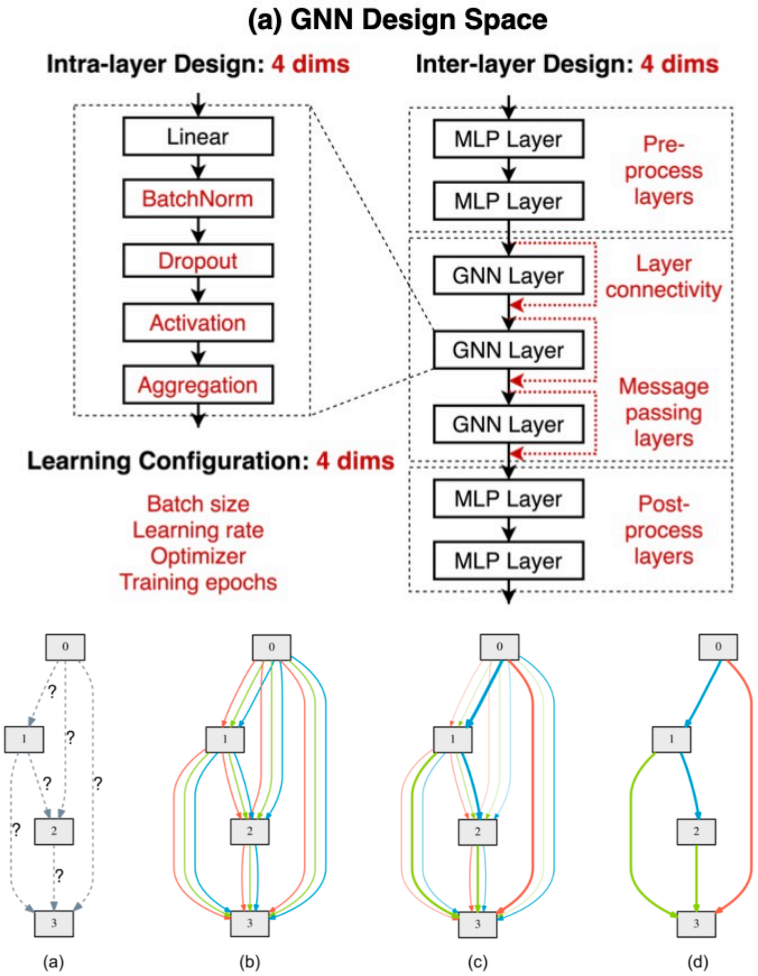
图上的自动机器学习（Survey）

Method	Search space					Tasks		Search Strategy
	Micro	Macro	Pooling	HP	Layers	Node	Graph	
GraphNAS [2020]	✓	✓	✗	✗	Fixed	✓	✗	RNN controller + RL
AGNN [2019]	✓	✗	✗	✗	Fixed	✓	✗	Self-designed controller + RL
SNAG [2020a]	✓	✓	✗	✗	Fixed	✓	✗	RNN controller + RL
PDNAS [2020c]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable
POSE [2020]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable
NAS-GNN [2020]	✓	✗	✗	✓	Fixed	✓	✗	Evolutionary algorithm
AutoGraph [2020]	✓	✓	✗	✗	Various	✓	✗	Evolutionary algorithm
GeneticGNN [2020b]	✓	✗	✗	✓	Fixed	✓	✗	Evolutionary algorithm
EGAN [2021a]	✓	✓	✗	✗	Fixed	✓	✓	Differentiable
NAS-GCN [2020]	✓	✓	✓	✗	Fixed	✗	✓	Evolutionary algorithm
LPGNAS [2020b]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable
You *et al.* [2020b]	✓	✓	✗	✓	Various	✓	✓	Random search
SAGS [2020]	✓	✗	✗	✗	Fixed	✓	✓	Self-designed algorithm
Peng *et al.* [2020]	✓	✗	✗	✗	Fixed	✗	✓	CEM-RL [2019]
GNAS[2021]	✓	✓	✗	✗	Various	✓	✓	Differentiable
AutoSTG[2021]	✗	✓	✗	✗	Fixed	✓	✗	Differentiable
DSS[2021]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable
SANE[2021b]	✓	✓	✗	✗	Fixed	✓	✗	Differentiable
AutoAttend[2021b]	✓	✓	✗	✗	Fixed	✓	✓	Evolutionary algorithm

Zhang, Ziwei, Xin Wang, and Wenwu Zhu. "Automated Machine Learning on Graphs: A Survey." *arXiv preprint arXiv:2103.00742* (2021).

图上的自动机器学习

- 用法: experiment(dataset, model) => experiment(dataset)
- 超参搜索优化 (HPO)
- 神经网络架构搜索 (NAS)
  - 定义搜索空间
    - 层内设计
      - ***BN, Dropout, Activation, Aggregation***
    - 层间设计
      - ***Layer connectivity(stack, skip),***
  - 搜索策略
    - *基于**RL**的方法*
    - *可微方法*
    - *进化算法*



1. You, Jiaxuan, Zhitao Ying, and Jure Leskovec. "Design space for graph neural networks." Advances in Neural Information Processing Systems 33 (2020).
2. Li, Guohao, et al. "Sgas: Sequential greedy architecture search." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

Open Graph Benchmark

- 由斯坦福等单位发布的图基准
- 多种任务、多种领域、多种规模

Category	Name	#Graphs	Average #Nodes	Average #Edges	Average Node Deg.	Average Clust. Coeff.	MaxSCC Ratio	Graph Diameter
Node ogbn-	products	1	2,449,029	61,859,140	50.5	0.411	0.974	27
	proteins	1	132,534	39,561,252	597.0	0.280	1.000	9
	arxiv	1	169,343	1,166,243	13.7	0.226	1.000	23
	papers100M	1	111,059,956	1,615,685,872	29.1	0.085	1.000	25
	mag	1	1,939,743	21,111,007	21.7	0.098	1.000	6
Link ogbl-	ppa	1	576,289	30,326,273	73.7	0.223	0.999	14
	collab	1	235,868	1,285,465	8.2	0.729	0.987	22
	ddi	1	4,267	1,334,889	500.5	0.514	1.000	5
	citation	1	2,927,963	30,561,187	20.7	0.178	0.996	21
	wikikg	1	2,500,604	17,137,181	12.2	0.168	1.000	26
	biokg	1	93,773	5,088,434	47.5	0.409	0.999	8
Graph ogbg-	molhiv	41,127	25.5	27.5	2.2	0.002	0.993	12.0
	molpcba	437,929	26.0	28.1	2.2	0.002	0.999	13.6
	ppa	158,100	243.4	2,266.1	18.3	0.513	1.000	4.8
	code	452,741	125.2	124.2	2.0	0.0	1.000	13.5

OGB排行榜

Leaderboard for ogbn-products

The classification accuracy on the test and validation sets. The higher, the better.

Package: >=1.1.1

Rank	Method	Ext. data	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	GIANT-XRT+SAGN+MCR+C&S	Yes	0.8673 ± 0.0008	0.9387 ± 0.0002	Yufei He (CogDL Team)	Paper, Code	1,154,654	GeForce RTX™ 3090 24GB (GPU)	Dec 8, 2021
2	GIANT-XRT+SAGN+MCR	Yes	0.8651 ± 0.0009	0.9389 ± 0.0002	Yufei He (CogDL Team)	Paper, Code	1,154,654	GeForce RTX™ 3090 24GB (GPU)	Dec 8, 2021
3	GIANT-XRT+SAGN+SLE+C&S (use raw text)	Yes	0.8643 ± 0.0020	0.9352 ± 0.0005	Eli Chien (UIUC)	Paper, Code	1,548,382	Tesla T4 (16GB GPU)	Nov 8, 2021
4	GIANT-XRT+SAGN+SLE (use raw text)	Yes	0.8622 ± 0.0022	0.9363 ± 0.0005	Eli Chien (UIUC)	Paper, Code	1,548,382	Tesla T4 (16GB GPU)	Nov 8, 2021
5	GIANT-XRT+GAMLP+MCR	Yes	0.8591 ± 0.0008	0.9402 ± 0.0004	Yufei He (CogDL Team)	Paper, Code	2,144,151	GeForce RTX™ 3090 24GB (GPU)	Dec 8, 2021
6	GAMLP+RLU+SCR+C&S	No	0.8520 ± 0.0008	0.9304 ± 0.0005	Yufei He (CogDL Team)	Paper, Code	3,335,831	GeForce RTX™ 3090 24GB (GPU)	Dec 8, 2021
7	GAMLP+RLU+SCR	No	0.8505 ± 0.0009	0.9292 ± 0.0005	Yufei He (CogDL Team)	Paper, Code	3,335,831	GeForce RTX™ 3090 24GB (GPU)	Dec 8, 2021
8	SAGN+SLE (4 stages)+C&S	No	0.8485 ± 0.0010	0.9302 ± 0.0003	Chuxiong Sun (CTRI)	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Sep 21, 2021

https://github.com/THUDM/CRGNN

异构图基准 (HGB)

- 一个统一的异构图基准（包含数据集和评估方式）
- **论文:** Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. *(KDD'21)*
- **代码 & 数据:** https://github.com/THUDM/HGB
- **排行榜:** https://www.biendata.xyz/hgb/
- 在HGB中，我们提供了一个简单的baseline：Simple-HGB。我们发现一个非常简单的异构模型就能够达到SOTA的结果。
- GAT + relation type attention + residual connection + L2 norm
  - cogdl implementation

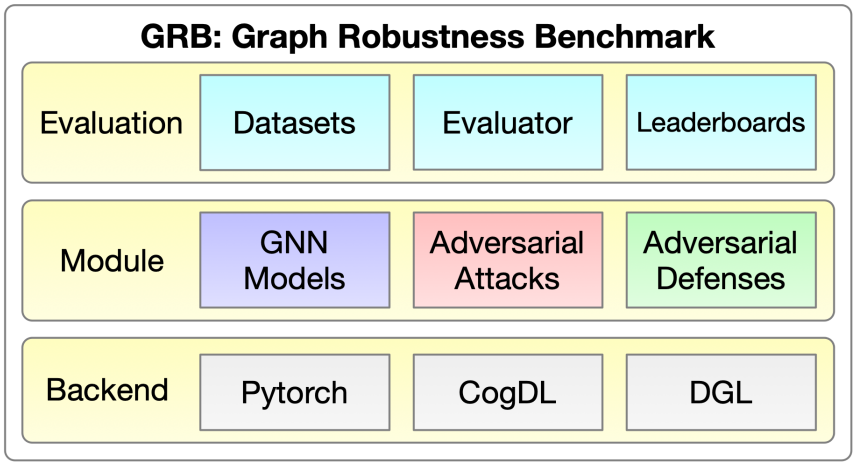
图鲁棒性基准 (GRB)

背景:

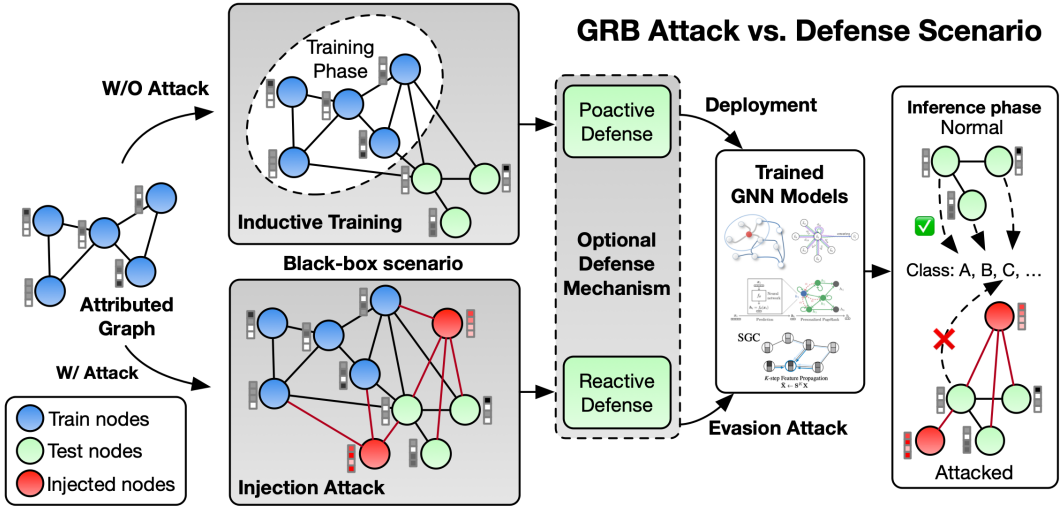
一些工作已经证明，图上的对抗攻击在很多任务上都能够影响图机器学习模型的鲁棒性。

存在的问题:

1. 之前的工作中没有定义清楚攻击模型。
2. 缺少统一和完整的评估方式。



GRB框架



图鲁棒性评估的示例

图鲁棒性基准 (GRB)

Homepage: https://cogdl.ai/grb/home

Github: https://github.com/THUDM/grb

Leaderboard: https://cogdl.ai/grb/leaderboard/

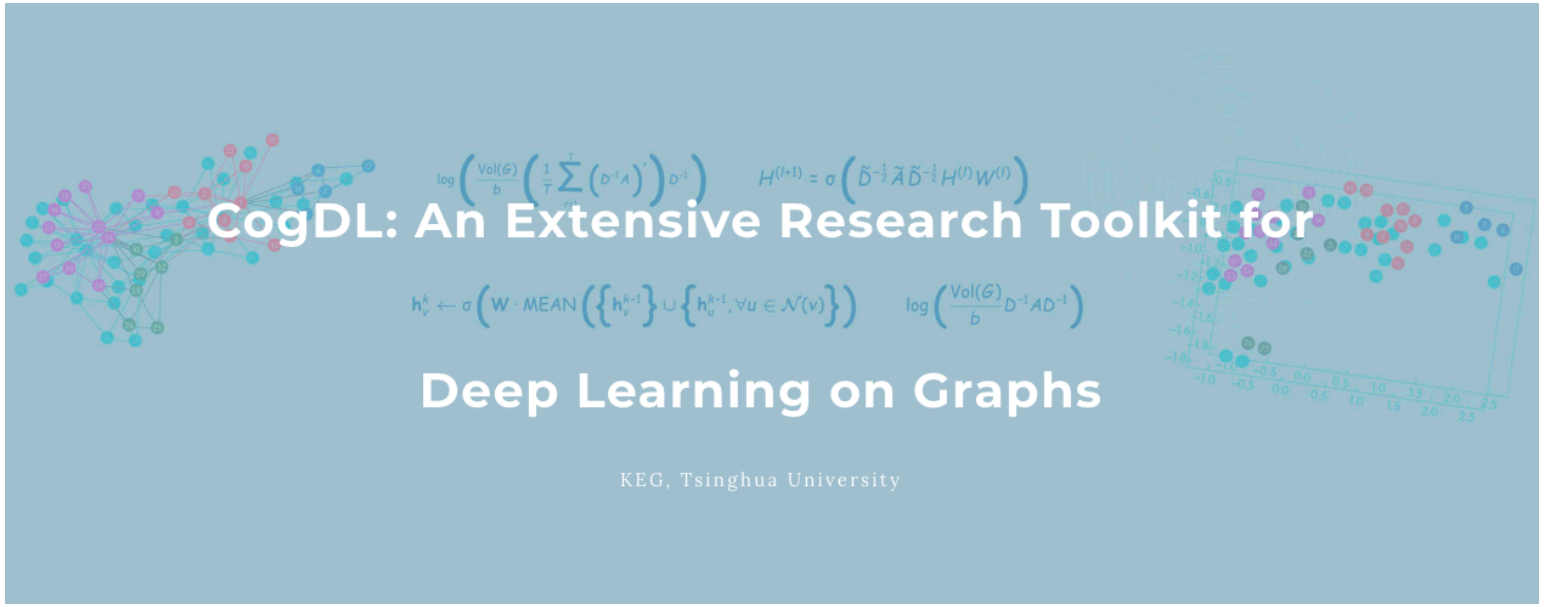
Docs: https://grb.readthedocs.io/

Graph Robustness Benchmark: Rethinking and Benchmarking Adversarial Robustness of Graph Neural Networks

Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Jie Tang

总结

- 图机器学习简介（背景、发展、挑战）
- 图神经网络模型基础（GCN、GraphSAGE、GAT）
- 图机器学习框架CogDL（简介、用法、结果）
- 图深度学习的前沿研究（大图、深层、自动、图基准）



欢迎大家关注图学习
与CogDL！

谢谢!

感谢所有的合作者！

- 侯振宇、郑勤锴、唐杰 et al.（清华KEG实验室）
- 于仲明、戴国浩、汪玉 et al.（清华NICS实验室）
- 周畅、杨红霞（阿里巴巴）
- 杨洋（浙江大学）
- 张鹏（智谱.AI）

https://github.com/THUDM/cogdl