

# 使用GCN做节点分类预测

## 实验结果

```
In [10]: device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = GCN_NET(dataset.num_node_features, 16, dataset.num_classes)
model = model.to(device)
data = dataset[0].to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)
loss_fn = torch.nn.CrossEntropyLoss()
model.train()
for epoch in range(200):
    optimizer.zero_grad()
    out = model(data)
    loss = loss_fn(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
    print('epoch: {} , Loss: {}'.format(epoch, loss))
```

```
epoch:20 , Loss:0.05575030774301529
epoch:27 , Loss:0.04867511987686157
epoch:28 , Loss:0.04276970028877258
epoch:29 , Loss:0.03786579892039299
epoch:30 , Loss:0.0338091105222702
epoch:31 , Loss:0.0304552111774683
epoch:32 , Loss:0.0276775099337101
epoch:33 , Loss:0.025362346321344376
epoch:34 , Loss:0.023414991796016693
epoch:35 , Loss:0.021763788536190987
epoch:36 , Loss:0.020356738939881325
epoch:37 , Loss:0.019156713038682938
epoch:38 , Loss:0.01813316158950329
epoch:39 , Loss:0.017263229936361313
epoch:40 , Loss:0.016530413180589676
epoch:41 , Loss:0.015922121703624725
epoch:42 , Loss:0.015422011725604534
epoch:43 , Loss:0.015014124102890491
epoch:44 , Loss:0.014683390967547894
epoch:45 , Loss:0.014418873004615307
```

```
In [11]: model.eval()
pred = model(data)
_, labels = pred.max(dim=1)
correct = int(labels[data.test_mask].eq(data.y[data.test_mask]).sum()).item()
acc = correct/int(data.test_mask.sum())
print('GCN Accuracy: {}'.format(acc))
```

GCN Accuracy:0.817

## 总结

- 配置环境花的时间时间大概是最多的，配置好了CUDA、cudnn、Pytorch等，遇到问题比较多（但都是小问题），不过都解决了。
- 看了一些关于GCN的讲解，论文中的公式推导大概有点感觉了，不过还是比较抽象。
- 李沐讲综述的时候提到的一个网址，感觉很有趣<https://distill.pub/2021/gnn-intro/>

# 图对抗（顺便跑了）

```
jupyter 攻击目标节点使其分类错误 最新检查点: 9 小时前 (已自动保存) Logout
File Edit View Insert Cell Kernel Help 可信 Python 3 (ipykernel)
运行 代码
perturbed_data.edge_index = torch.tensor(perturbed_edge_index)
perturbed_data = perturbed_data.to(device)

model.eval()
perturbed_pred = model(perturbed_data)
_, perturbed_labels = perturbed_pred.max(dim=1)

if labels[target] == data.y[target]:
    n_t += 1
    if perturbed_labels[target] != data.y[target]:
        n_a += 1

print('target {} '.format(target))
print('true label: {} '.format(data.y[target]))
print('before attack: {} '.format(labels[target].data.cpu().numpy()))
print('after attack: {} '.format(perturbed_labels[target].data.cpu().numpy()))
print('Attack Success Rate is {} '.format(n_a/n_t))
# print(n_a, n_t)
print('Final Attack Success Rate is {} '.format(n_a/n_t))

-0.34591135]
after attack: 1 [ 1.4553372  1.8075616 -1.2178975 -2.9053364 -2.8944843  0.8334562
-1.1642725]
Attack Success Rate is 0.9365942028985508
target 2706
true label: 3
before attack: 3 [-2.6059568 -0.77854866 -1.4395901  4.62946 -0.51745176 -1.7714872
-3.3358107 ]
after attack: 2 [-2.8112838 -0.4158819  3.7232087  1.3551742 -1.8035733 -2.9073741
-4.0050406]
Attack Success Rate is 0.93662290629244
target 2707
true label: 3
before attack: 3 [-2.4856572 -0.59238446 -1.2492504  4.316483 -0.79296 -1.5004936
-3.1821826 ]
after attack: 1 [-1.8157189  3.2507746 -1.1494437  1.2595955 -2.1542678 -2.3311355
-3.5602884]
Attack Success Rate is 0.9366515837104072
Final Attack Success Rate is 0.9366515837104072

In [14]: import winsound
duration = 1000 # 持续时间/ms
frequency = 500 # 频率/Hz
winsound.Beep(frequency, duration)
```

# SL-DSGCN

论文: Investigating and Mitigating Degree-Related Biases in Graph Convolutional Networks

来源: 2020CIKM

## 概要

- GCN在图的半监督学习上能取得良好表现, 但训练难度大
- 当标签数据有限, GCN在**low-degree nodes**的更新信息不足, 预测表现不佳
- 提出**SL-DSGCN**(Self-Supervised-Learning Degree-Specific GCN), 从模型和数据上减轻关于度的偏差
- 在数据集中验证了SL-DSGCN提高了自我训练GCN能力和低度节点的准确性。

## 分析问题

- 考虑如何捕捉不同层之间节点的复杂关系 / 如何为低度节点提供信息。
- 使用了 Cora, Citeseer, Pubmed, Reddit 四个数据集, 通过可视化分析可以发现预测准确率随着节点度数的变化分布。
- 度数在数据集中并非独立同分布, 而度数越大的点往往容易与带标签邻居相邻, 可以频繁参与优化过程, 这也是GCN在度数大的点性能好的原因之一。

## 实现过程

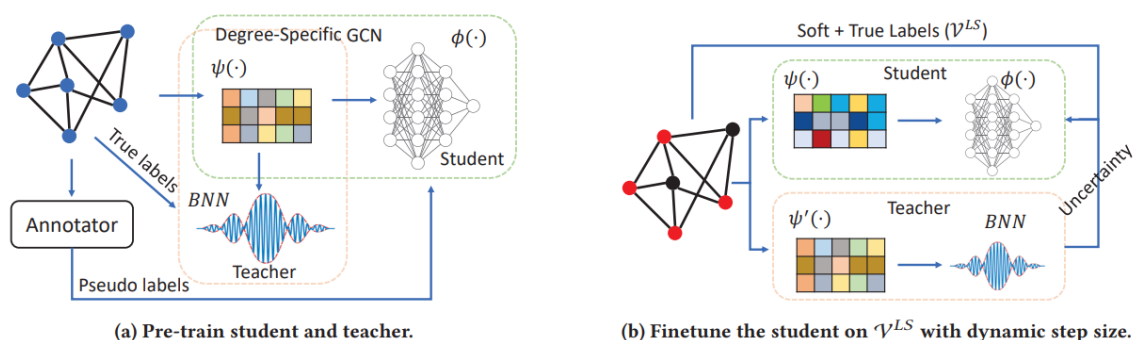


Figure 6: Overall framework of SL-DSGCN.

### Degree-Specific GCN Layer

- SL-DSGCN的核心部分，用来捕捉不同度数节点的差异和相似性
- 传统GCN使用与相邻节点的加权平均来计算特征，该结构将节点的度考虑在内，使度不同的节点在传递消息和更新特征过程有所不同。（也许是特征拼接之后再行图卷积操作）
- 该层参数由RNN生成，大概是因为度不同的节点通过同一个RNN可以获得不同的度的特征信息，从而在性质相似的点中，特征由度来决定。

### Self-Supervised-Training with Bayesian Teacher Network

- Pre-training with the Annotator

teacher-student Network是用一个更宽更复杂，但已经训练好的神经网络(teacher)去训练另一个窄而深的网络(studen)。这篇论文就是使用这种方法生成**软标签和不确定分数**。

论文里提出带Bayesian teacher network的子监督训练，大概步骤：（1）使用**图注释器**为未标记的节点生成伪标签。

- （2）使用生成的伪标签作为额外的监督，在有标签和无标签的数据上训练学生网络。
- （3）使用训练好的学生网络，对图中的所有节点进行预测。
- （4）在学生网络产生的预测上训练贝叶斯教师网络。
- （5）使用训练好的**贝叶斯教师网络**来完善由图注释器生成的伪标签。
- （6）重复步骤2-5，直到收敛。

- Fine-tuning Student with Uncertainty Scores

训练好的Bayesian teacher network可以计算每个节点的不确定分数，这东西可以给损失函数加权优化收敛，这样可以更好地利用未标记数据来训练。

### Training Algorithm

---

**Algorithm 1: Self-learning for SL-DSGCN**

---

**Input:**  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$

**Output:** Parameters  $\theta$  of student network  $\phi(\psi(\cdot))$

// Pre-training

1 Acquire pseudo-labels for  $\mathcal{V}^U$  using a graph annotator;

2 Pre-train  $\phi(\psi(\cdot))$  on pseudo labels;

3 Fix  $\psi(\cdot)$  and pre-train BNN part of the teacher network;

4 Acquire prediction  $y_i^s$  and uncertainty score  $c_i$  for every node in  $\mathcal{V}^U$  from the teacher;

// Fine-tuning

5 Compile a soft-labeled node set  $\mathcal{V}^S \subset \mathcal{V}^U$  where the teacher network agrees with the annotator;

6 Build  $\mathcal{V}^{LS} = \mathcal{V}^L \cup \mathcal{V}^S$  to fine-tune the student network;

7 **while** *not converge* **do**

8     Compute dynamic step size  $\eta_i$  for  $v_i \in \mathcal{V}^{LS}$  as

$$\eta_i = \eta \cdot \eta_i^c \cdot \eta_i^d;$$

9     Update parameters of the student network as

$$\theta' = \theta - \sum_{v_i \in \mathcal{V}^{LS}} \eta_i \mathcal{L}(v_i; \theta);$$

10 **end**

---

## 一些思考

- 所以 graph annotator 要怎么设计得高效快速？按照Paper的说法是一种能够给未标记节点生成伪标签的启发式算法，本质上是也是需要经过训练来提高准确率的，这样的话它应该决定了学生网络的训练效果的阈值了。既然给训练过程分层了，那收敛应该也会变慢，感觉这是一个权衡，毕竟也有可能只对学生网络多迭代几次也能达到同样效果。
- 这篇论文是2020年的，但RNN应该早就可以用更优秀的结构来代替了？比方说LSTM和Transformer啥的。
- 在稠密图的表现如何？感觉准度应该差不了多少但是计算会比较慢吧，所以感觉应该要先判断图的性质再考虑是否进入SL-DSGCN结构。

## Meta-tail2vec

---

论文：Towards Locality-Aware Meta-Learning of Tail Node Embeddings on Networks

来源：2020CIKM

[GitHub - smufang/meta-tail2vec: Implementation of meta-tail2vec published in CIKM 2020 paper "Towards Locality-Aware Meta-Learning of Tail Node Embeddings on Networks".](#)

## 摘要

- 提出meta-tail2vec 方法来优化尾部节点嵌入问题
  - 对于**尾部节点的嵌入**问题，采用重建的回归模型来计算。
  - 对于网络的局部感知位置问题，提出**删边和元学习**的方法局部调整
- 在公共数据集中进行大量实验，验证了meta-tail2vec的性能提升

## 背景

- network embedding
- GNN

## 分析问题

- 对于没有 丰富结构链接/低度数 的节点，嵌入性能较差
- 一节点的度在网络中变化很大，不是独立同分布
- 某些网络的节点度的特点为长尾分布，即相当一部分节点处于尾部且**度数非常低**
- 如何在有限的结构信息中学习有效的**尾部节点的嵌入向量**？

### 问题的难点

- 尾部节点提供的信息非常少
- 嵌入时需要考虑每个节点的结构位置，在于无法适应每个节点（链接很少的尾部节点），否则会过拟合。

## 方法

- 对于尾部节点信息稀缺的问题，考虑利用头部节点的嵌入(Oracle embeddings)来训练**回归模型**并且通过**自监督学习重建**(视为一个few-shot问题)。利用该模型，可以预测尾部节点的嵌入使其逼近Oracle embeddings。同时，通过删边操作模拟尾部节点的有限信息。
  - 在邻居随机选k个(**支持集合**)断掉，那样尾部节点嵌入学习就可以变成一个**few-shot**
  - 使支持集合来更新先验知识，再用查询节点更新回归模型
  - 回归模型可以用来预测尾部节点的嵌入向量，由嵌入器和回归其组成
  - 嵌入器将每个节点降维，回归器则将嵌入向量映射到目标值。
- 针对每个节点结构位置问题，采用**元学习**范式，采用模型无关的MAML(Model-Agnostic Meta-Learning)元学习框架。
  - 通过在多个任务上进行梯度下降来学习通用初始化参数，使这些参数快速适应新任务
  - 更新后的参数在查询集上预测从而得到嵌入向量。
- 实验是在3个公共数据集上做节点分类和链路预测任务

## 一些思考

- 这篇论文陌生用的方法好像有点多，有点看不过来，像文中提到的SDNE、ARGA、DDGCN等一系列嵌入方法都没来得及细究，不过大致就是元尾部嵌入要比这些都好就是了
- 关于few-shot的部分有点像直接用了掩码的思想，总之就是把邻居mask掉去训练。虽然文章的方法是把**询问节点**的k个邻居mask掉，但我觉得直接在整个图上按度数贡献到概率上再随机mask也是可以的。
- 感觉这个方法会很慢，举个例子我有一个菊花图，中心是A。首先一个问题是其他全是尾部节点，A永远在支持集合中，给的权重太高了；其次是如果不自适应跳数的话，实验中2跳的性能阈值就会很慢很慢，因为这个情况每个点的信息都是整张图所有节点给的。

# TNGNN

论文：Tail-GNN: Tail-Node Graph Neural Networks

来源：2021KDD

## 摘要

- 针对长尾效应提出**Tail-GNN**模型
- 提出**transferable neighborhood translation**的概念，捕获节点与邻居的传递关系
- 在五个公共数据集上进行实验，Tail-GNN取得了SOTA成绩

## 背景

- 图数据集的普遍特征：长尾分布
- 最近的GNN模型虽然很好地学习了节点表示，但是不能针对处理尾部节点
- 尾部节点的结构信息缺失影响节点分类和链路预测等任务

## 分析问题

- 如何发现尾节点缺失的邻域信息？
  - Tail-GNN uses a novel concept of transferable neighborhood translation to model the variable ties between a target node and its neighbors. This allows it to predict missing neighborhood information for tail nodes and complement their neighborhood aggregation.
- 如何在保持节点间通用性的同时，对每个尾节点的缺失信息进行局部定位？
  - Tail-GNN uses a transferable form of neighborhood translation that can be further localized to suit the local context of each node. This allows Tail-GNN to maintain generality across nodes while still being able to adapt to local contexts and provide accurate predictions for missing neighborhood information.

## 方法

### Neighborhood translation

- contrastive strategy
- localizing strategy

### neighborhood aggregation

- for head nodes: directly aggregate over the observed neighborhood
- for tail nodes:  $h_v^{l+1} = \mathcal{M}(h_v^l, \{m_v^l\} \cup h_i^l : i \in \mathcal{N}_v; \theta^{l+1})$

### Overall loss

- task loss
- loss for missing information constraint
- loss for adversarial constraint

$$\text{Overall loss} : \text{Min}_{\Theta}(\text{Max}_{\theta_d}(\mathcal{L}_t + \mu\mathcal{L}_m - \eta\mathcal{L}_d))$$

## 一些思考

- 消融实验好像没有讨论到超参数对Tail-GNN的影响，只有某一部分方法缺失的影响，内部的参数到底是怎么定义的？
- 随机缺失信息应该会导致每一次的准确率都不太一样吧，这是怎么做的比较实验？
- 这个模型是包含了对比学习？或者说是不是可以用对比学习来优化调参？

# Cold Brew

论文：Cold Brew: Distilling Graph Node Representations with Incomplete or Missing Neighborhoods

来源：2022 ICLR

## 摘要

- 提出Cold Brew方法解决SCS(孤立点Strict Cold Start)节点的噪声邻居问题
- 提出FCR(特征贡献比 Feature Contribution Ratio)度量GNN中节点特征对预测结果的贡献程度
- 使用Teacher-Student Distilling提高SCS的泛化能力
- 在GNN层添加独立的结构嵌入来缓解节点过平滑问题
- 再公开数据集专有电子商务数据集上表现优异

## 问题分析/方法

- 跟前几篇文章类似，度数的幂律分布、传统GNN只能学习已知节点的嵌入表示等通病
  - 经过之前度过的文章，有很多通解，半监督/自监督/元学习之类的，而本文的Cold Brew用了Teacher-Student模型，所以大概是用类似半监督学习的思想来解决的。
- 如何有效地提炼Teacher模型的信息，以便进行SCS泛化
  - SE(Structural Embedding, 结构嵌入)：在Teacher GNN训练时额外增加一个结构来学习结构信息，可以通过梯度反向传播。因每个阶段嵌入不同而且都参与了特征转换，缓解过度平滑的问题(Oversmoothing)。
- 如何让Student模型利用这些信息
  - 学习从节点特征到该结构的映射，而不需要知道教师模型所使用的图结构（解决了MLP无法吸收结构信息的弊端），对于SCS来说可以找到一组虚拟邻居。
- 传统方法SCS只能用MLP映射到标签上忽略图的结构信息
  - 教师模型可以学习到图结构信息，并嵌入到低维空间，而学生模型是MLP模型，但可以从教师模型中蒸馏出图结构信息。
- 使用FCR作为度量指标来衡量节点学习的难度来优化模型架构设计
  - 因为我们的标签信息来自于节点本身特征和邻接结构，然后文章说前者用MLP映射，后者用LP(Label Propagation)来映射(嵌入了邻居信息)。FCR是按照他们有关两者信息的比值的公式来定义的(5a/5b)。

## 一些思考

- Cold Brew只对SCS Work吗？应该可以考虑为一种对low degree节点的通解
- FCR为什么这样算读不懂，但是感觉计算方式是可以改进的
- SCS的定义不是孤立点吗？我的理解是在图里是完全没有结构信息的，还是不懂它如何映射。所以是指它是标签缺失的，所以只能用MLP学习自己然后走Student模型吗？

## 尝试复现

### 准备环境

在Github上fork了一份到自己仓库里，它给了requirements，装环境很方便

```
conda create -n gtg
conda activate gtg
pip install -r requirements.txt
```

- 报错1:缺失工具链

```
error: subprocess-exited-with-error

× Preparing metadata (pyproject.toml) did not run successfully.
| exit code: 1
└> [6 lines of output]

    Cargo, the Rust package manager, is not installed or is not on PATH.
    This package requires Rust and Cargo to compile extensions. Install it
through
    the system's package manager or via https://rustup.rs/

    Checking for Rust toolchain....
    [end of output]

note: This error originates from a subprocess, and is likely not a problem with
pip.
error: metadata-generation-failed

× Encountered error while generating package metadata.
└> see above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

喂给chatgpt，看来是缺少Rust和Cargo工具链，去官网下完之后重开终端就可以了。

- 报错2: python版本不正确

```
ERROR: Ignored the following versions that require a different python version:
0.7 Requires-Python >=3.6, <3.7; 0.8 Requires-Python >=3.6, <3.7
ERROR: Could not find a version that satisfies the requirement dataclasses==0.8
(from versions: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6)
ERROR: No matching distribution found for dataclasses==0.8
```



仓库作者没有说Python的版本，但是dataclasses0.7要Python>=3.6且<3.7，torch要python>=3.6.2，我就首先用了3.6。

- 报错3：找不到与要求的 dgl 版本 0.7.0 相匹配的分发版本

```
ERROR: Could not find a version that satisfies the requirement dgl==0.7.0 (from
versions: 0.0.1, 0.1.0, 0.1.2, 0.1.3, 0.2, 0.3, 0.3.1, 0.4, 0.4.1, 0.4.2, 0.4.3,
0.4.3.post1, 0.4.3.post2, 0.5.0, 0.5.1, 0.5.2, 0.5.3, 0.6.0, 0.6.0.post1, 0.6.1,
0.9.0, 0.9.1, 1.0.0, 1.0.1)
ERROR: No matching distribution found for dgl==0.7.0
```

一开始我通过安装更高版本来解决问题，但是发现很多包都会遇到同样的版本不兼容的问题，省级了pip不能解决问题。我索性直接改了requirements.txt，把

*dgl, fastparquet, joblib, Mako, mkl - fft, mkl - random, mkl - service, ml - collections*

的指定版本给去掉了。但是最终还是解决不了问题，因为所有版本都不兼容了。最后用了3.8版本的python。意识到作者给出的requirements只是pip list并没有拓扑顺序，所以我就分块去装环境。如果pip没有就用conda装，然后除了mkl版本没装指定的以外，其余的都没问题了。

```
C:\Windows\System32\cmd.exe - "D:\Env\anaconda3\condabin\conda.bat" activate gtg - "D:\Env\anaconda3\co
```

```
(gtg) C:\Users\lenovo\Documents\GitHub\gnn-tail-generalization>pip list
Package            Version
-----
abs1-py            0.13.0
alembic            1.6.5
astor              0.8.1
attrs             21.2.0
cachetools        4.2.2
certifi            2021.5.30
chardet            4.0.0
charset-normalizer 3.1.0
click              8.0.1
cliff              3.8.0
cloudpickle        1.6.0
cmaes              0.8.2
cmd2               2.1.2
colorama           0.4.4
colorlog           5.0.1
contextlib2        21.6.0
cramjam            2.3.2
cyclor             0.10.0
Cython             0.29.14
dataclasses        0.6
decorator          4.4.2
dgl                1.1.0
dill               0.3.4
dpcpp-cpp-rt       2023.1.0
fastparquet        0.6.2
filelock           3.0.12
fsspec             2021.6.1
gast               0.5.0
gdown              4.0.1
gensim             3.8.3
google-auth        1.35.0
google-auth-oauthlib 0.4.6
googledrivedownloader 0.4
GPUutil            1.4.0
greenlet           1.1.0
grpcio             1.38.1
gym                0.26.2
gym-notices        0.0.8
h5py               3.8.0
huggingface-hub    0.0.12
idna               2.10
imageio            2.9.0
importlib-metadata 6.6.0
intel-cmplr-lib-rt 2023.1.0
```

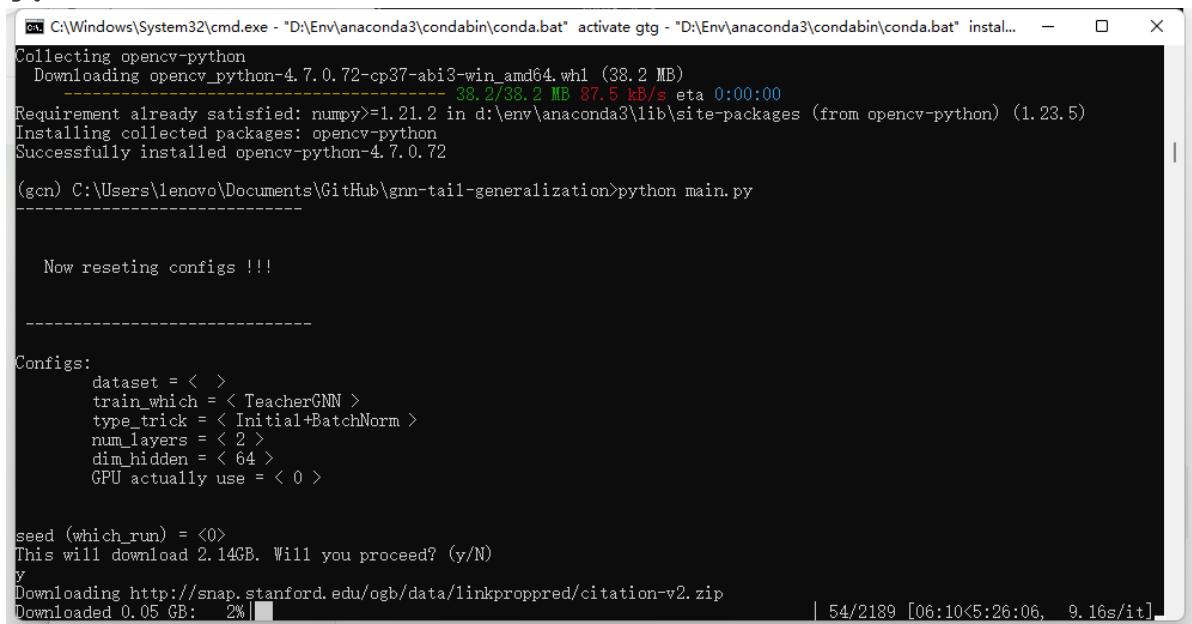
## 尝试运行

通过终端尝试运行以下语句

```
python main.py --dataset=Cora --train_which=TeacherGNN --whetherHasSE=000 --
want_headtail=1 --num_layers=2 --use_special_split=1
```

报错说我的torch-geometric有问题，但是list已经有了，看来跟torch版本不是很搭，然后我放弃了按照readme里的说法，去网上找torch1.9.0+cu111对应的geometric，发现x64没包。

因为越搞越乱了，重写建了个环境，这次直接装核心依赖，忽略requirements.txt。然后所有的包都没有指定版本(最后是torch2.0.1+cpu)，然后直接跑main.py，遇到没有装的包再pip install，结果work了。



```
C:\Windows\System32\cmd.exe - "D:\Env\anaconda3\condabin\conda.bat" activate gtn - "D:\Env\anaconda3\condabin\conda.bat" instal...
Collecting opencv-python
  Downloading opencv_python-4.7.0.72-cp37-abi3-win_amd64.whl (38.2 MB)
    38.2/38.2 MB 87.5 kB/s eta 0:00:00
Requirement already satisfied: numpy>=1.21.2 in d:\env\anaconda3\lib\site-packages (from opencv-python) (1.23.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.7.0.72

(gcn) C:\Users\lenovo\Documents\GitHub\gtn-tail-generalization>python main.py

-----

Now reseting configs !!!

-----

Configs:
  dataset = < >
  train_which = < TeacherGNN >
  type_trick = < Initial+BatchNorm >
  num_layers = < 2 >
  dim_hidden = < 64 >
  GPU actually use = < 0 >

seed (which_run) = <0>
This will download 2.14GB. Will you proceed? (y/N)
y
Downloading http://snap.stanford.edu/ogb/data/linkproppred/citation-v2.zip
Downloaded 0.05 GB: 2% | 54/2189 [06:10<5:26:06, 9.16s/it]
```

(目前正在下载数据集)

感觉这四篇论文就看到这里，看看最近能不能用时间（课设比较多）复现一个，然后用另外一篇markdown记录一下。