-

**PCGCNN实现**

**中国科学院微电子研究所**

**2025.3.25**

# 数据集

G. M. Fu, M. K., & van Diggelen, F. (2020). Android raw GNSS measurement datasets for precise positioning. In Proceedings of the 33rd international technical meeting of the satellite division of the institute of navigation (p. 1925-1937).

描述：<https://www.kaggle.com/competitions/google-smartphone-decimeter-challenge/data>

2020年的数据，（GSDC2021）对每种接收机，包含

* ground\_truth.csv；真值：经纬高、速度真值、DOP等
* [train/test]/[drive\_id]/[phone\_name]/[phone\_name]\_derived.csv：基本信息：卫星位置、速度、伪距、各种偏差
* [train/test]/[drive\_id]/[phone\_name]/[phone\_name]\_GnssLog.txt：扩展信息：CN0、高度角、多普勒频移等

是deep\_gnss采用的数据集，但deep\_gnss没有采用GnssLog.txt文件

GSDC2023-2024

<https://www.kaggle.com/competitions/smartphone-decimeter-2023>

数据集格式变化了，主要有基本信息表格额外提供了wls结果

# Deep\_gnss代码可用部分

数据集部分可直接使用，需要补充读取GnssLog.txt额外信息的部分

没有计算wls作为位置粗估计值，而是真值+随机噪声，但是附带库文件中包含了计算wls结果的函数

# GCNN

graphSAGE：

<https://github.com/twjiang/graphSAGE-pytorch/blob/master/README.md>

<https://github.com/williamleif/GraphSAGE?tab=readme-ov-file>

（论文<https://arxiv.org/pdf/1706.02216>）

GCN

<https://github.com/tkipf/gcn/>

※※※开源库PyTorch Geometric※※※

目前的问题：

* ※将上一时刻的输出作为这一时刻的输入会带来很多问题，主要是训练的问题，需要采用（1）直接展开：RNN的循环展开训练模式，Loss计算时对之前诸多时刻的输入求导。Pytorch自动求导（2）逐步优化（Teacher Forcing），训练时使用真值作为输入，但推理时可能不稳定，需要训练技巧（3）强化学习，适用于控制问题、路径规划等
* 训练输入的问题，若采取（1）的思路，输入图不可能预先生成，必须即时生成，需要改变现有的GCNN输入部分的代码（均是将输入图预先存储在数据文件中）
* 现有模型的任务集中于节点分类任务，即对输入图的节点进行分类，每个类一个标签，但GNSS定位任务是最终输出一个定位结果开源库+图回归模型

# 实现步骤

1. 数据集处理

数据csv读取

计算WLS结果，伪距和伪距残差

※计算速度和预测伪距残差（~~速度可以跳过，用上一时刻的输出的位置直接作为预测值）~~

图构建

1. GCNN实现

~~库调用~~graphSAGE代码应用

※GCNN+循环神经网络

训练和测试框架



# 附录

## PCGCNN实现

import torch

import torch.nn.functional as F

from torch\_geometric.nn import GCNConv

class RecurrentGCN(torch.nn.Module):

def \_\_init\_\_(self, in\_channels, hidden\_dim, output\_dim, T=5):

super(RecurrentGCN, self).\_\_init\_\_()

self.T = T # 递归时间步

self.gcn1 = GCNConv(in\_channels, hidden\_dim)

self.gcn2 = GCNConv(hidden\_dim, hidden\_dim)

self.mlp = torch.nn.Linear(hidden\_dim, output\_dim)

def forward(self, x, edge\_index):

h = x # 初始输入

for t in range(self.T):

h = F.relu(self.gcn1(h, edge\_index))

h = F.relu(self.gcn2(h, edge\_index))

out = self.mlp(h) # 计算输出

h = self.feedback\_transform(out, h) # 反馈到下一时刻

return out # 返回最终时刻的输出

def feedback\_transform(self, out, h):

"""将上一时刻输出转换为下一时刻输入"""

return torch.cat([h, out], dim=-1) # 这里可以换成其他变换