

GL.zip 的使用说明 模型结构说明

一

环境配置

配置好 python 环境:

```
conda create --name py312 python=3.12
```

```
conda activate py312
```

安装必要的包:

Pytorch:

(网址: <https://pytorch.org/get-started/locally/>)

```
conda install pytorch torchvision torchaudio pytorch-cuda=12.4 -c pytorch -c nvidia
```

PyG:

(网址: <https://pytorch-geometric.readthedocs.io/en/latest/install/installation.html>)

```
pip install torch_geometric
```

```
pip install pyg_lib torch_scatter torch_sparse torch_cluster torch_spline_conv -f  
https://data.pyg.org/whl/torch-2.4.0+cu124.html
```

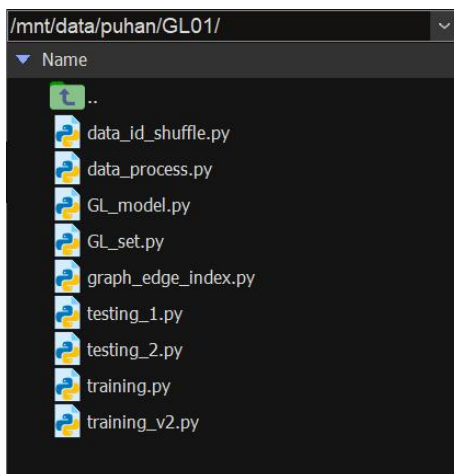
安装 pandas, numpy

二

解压

解压 GL01.zip 到任意路径

在 windows 或者 linux 上都可以



三

下载数据

下载地址:

<https://disk.pku.edu.cn/link/AA9BCF06CEBB9A4546B0FA8897632A9D3C>

文件夹名: data_batteryack

有效期限: 永久有效

四

程序运行前的设置

GL_set.py 是整个 python project 的设置文件

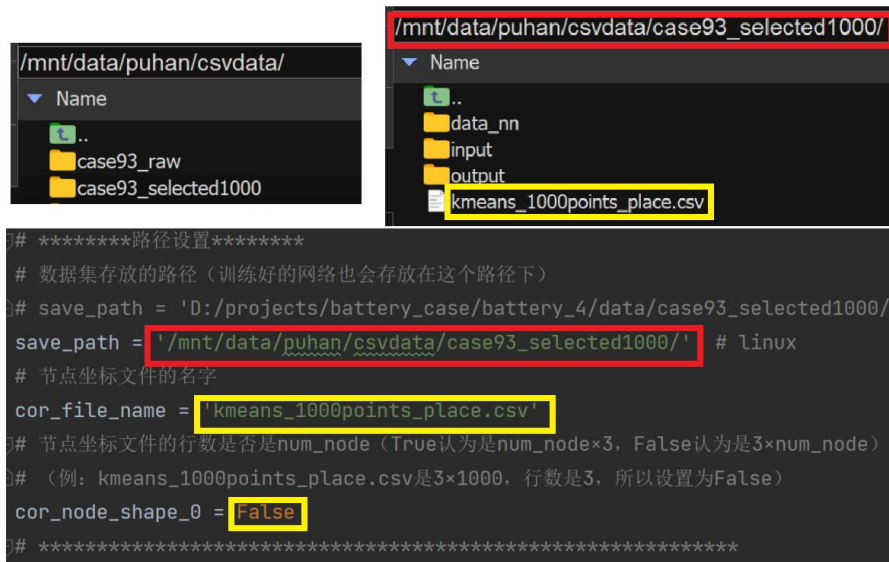
必要操作:

在数据集目录下新建 data_nn 文件夹

在 GL_set.py 中, 修改 save_path 为数据集的路径 (红框处)

非必要操作:

根据 GL_set.py 的注释, 修改程序中其它你认为需要修改的部分



The image shows two screenshots. The top left screenshot shows a file explorer view of the directory `/mnt/data/puhan/csvdata/` with subdirectories `case93_raw` and `case93_selected1000`. The top right screenshot shows a file explorer view of the directory `/mnt/data/puhan/csvdata/case93_selected1000/` with subdirectories `data_nn`, `input`, and `output`, and a file `kmeans_1000points_place.csv`. The bottom screenshot shows the `GL_set.py` file with the following code:

```
# *****路径设置*****  
# 数据集存放的路径 (训练好的网络也会存放在这个路径下)  
# save_path = 'D:/projects/battery_case/battery_4/data/case93_selected1000/'  
save_path = '/mnt/data/puhan/csvdata/case93_selected1000/' # linux  
# 节点坐标文件的名字  
cor_file_name = 'kmeans_1000points_place.csv'  
# 节点坐标文件的行数是否是num_node (True认为是num_node*3, False认为是3*num_node)  
# (例: kmeans_1000points_place.csv是3*1000, 行数是3, 所以设置为False)  
cor_node_shape_0 = False  
# *****
```

五

根据坐标点建立图结构 数据集预处理

在 GL01 目录下, 依次运行

`python graph_edge_index.py`

`python data_id_shuffle.py`

`nohup python data_process.py &`

运行完毕后, `save_path` 目录下会生成 `edge_index.csv` 和 `shuffle_id.csv`, 并会在上一步新建的 `data_nn` 文件夹里生成预处理后的数据集

更多内容详见代码注释

```
puhan@cmhpc1001:~$ export PATH=~/.anaconda3/bin:$PATH  
puhan@cmhpc1001:~$ source activate  
(base) puhan@cmhpc1001:~$ conda activate py312  
(py312) puhan@cmhpc1001:~$ cd GL01/  
(py312) puhan@cmhpc1001:~/GL01$ python graph_edge_index.py  
点云坐标矩阵: torch.Size([1000, 3])  
num_node: 1000  
边连接矩阵: torch.Size([2, 6000])  
edge index path: /mnt/data/puhan/csvdata/case93_selected1000/edge_index.csv  
end  
(py312) puhan@cmhpc1001:~/GL01$ python data_id_shuffle.py  
shuffle id path: /mnt/data/puhan/csvdata/case93_selected1000/shuffle_id.csv  
end  
(py312) puhan@cmhpc1001:~/GL01$ nohup python data_process.py &
```



六

训练模型

利用 data_nn 文件夹里的预处理后的数据集，训练神经网络

打开 training.py，修改 name_pth 为训练好后的模型文件的名称

运行 training.py 后，会在 save_path 目录下生成名为 name_pth 的 pth 文件

在 GL01 目录下，运行

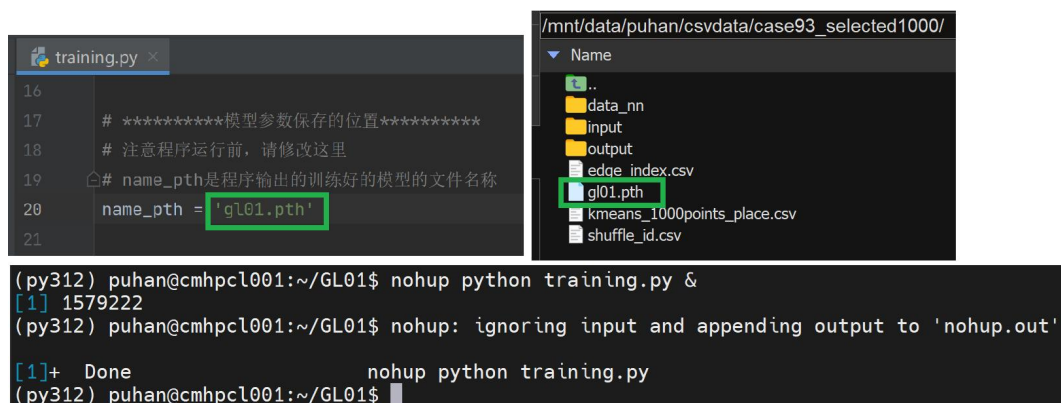
`python training.py`

也可以将其改为后台指令

`nohup python training.py &`

如果 data_nn 文件夹的文件过多或过大，内存不足，可以尝试改为运行 training_v2.py

关于 training.py 和 training_v2.py 的区别，详见代码注释



七

测试模型

测试 save_path 目录下的 pth 文件的效果

在 GL01 目录下，运行

`python testing_1.py`

和

`python testing_2.py`

关于 testing_1.py 和 testing_2.py 的区别，详见代码注释

实际上，该模型的效果很不好

但是整体的代码框架和思路可以作为一个参考和启发

八

模型设计思路

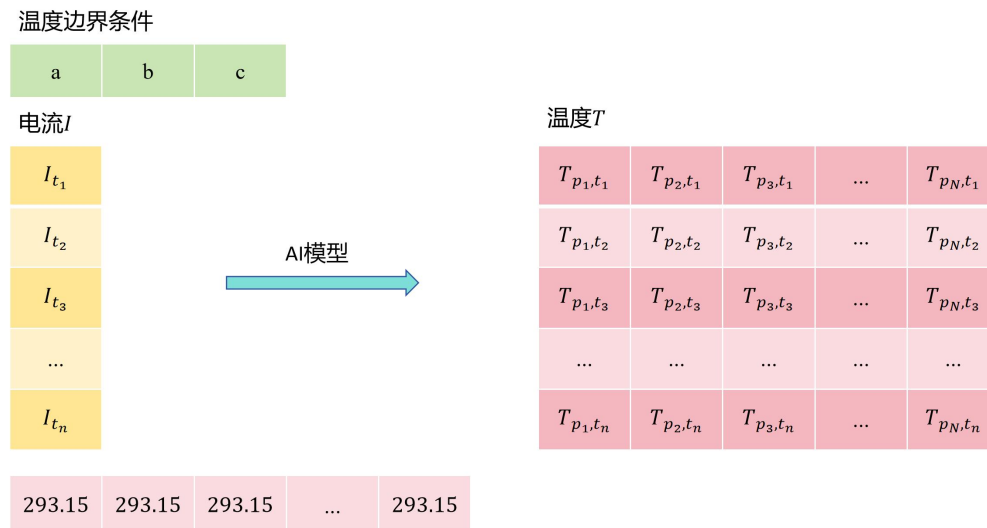
整体目标：

训练一个神经网络

输入：电流（ n 个时序值）、温度边界条件（3 个值）

输出： N 个点在 n 个时刻的温度（ $n \times N$ 的矩阵）

其中， $n=20$ ， $N=742826$ ，即 742826 个点温度随时间的变化（ t_1 时刻到 t_{20} 时刻）， t_0 时刻默认所有点的温度都是 293.15K



神经网络 GL 的设计思路：

单时间步预测

使用方法：

在最终使用时，把 GL 连续重复使用 20 次

上一时刻的输出作为下一时刻的输入

即：输入 t_0 时刻的温度，通过 GL 输出 t_1 时刻的温度；将输出再次作为输入，通过 GL 输出 t_2 时刻的温度；将输出再次作为输入，通过 GL 输出 t_3 时刻的温度；如是循环，直到最后输出 t_{20} 时刻的温度

训练过程：

因为是单时间步预测，所以是单时间步训练。一共有 93 个算例，每个算例 20 个时间步，因此可以拆成 $93 \times 20 = 1860$ 个数据集，并将其一部分作为训练集，一部分作为测试集。

致命问题：

单时间步预测，会造成误差累积

例如： t_9 时刻的预测结果已经不准确，但是此预测结果又会作为 GL 的输入去输出 t_{10} 时刻，导致 t_{10} 时刻的预测结果也不准确； t_{10} 时刻又会作为 GL 的输入去输出 t_{11} 时刻，进而影响之后的 t_{11} 、 t_{12} 等。误差会随时间步逐渐增大，进而一发不可收拾。

网络结构：

网络将擅长空间预测的图网络 GNN，和擅长时序预测的长短期记忆网络 LSTM 结合了起来
图网络 GNN：输入当前时刻的温度，输出下一时刻的温度增量

长短期记忆网络 LSTM：输入当前及之前所有时刻的电流，输出下一时刻的温度增量

普通网络：输入当前时刻的边界温度，输出下一时刻的温度增量

三个温度增量加和，即是下一时刻的温度增量

物理意义：

神经网络并不需要追求物理意义，但此网络结构是有一定物理意义的（虽然效果很不好）

图网络 GNN：整合空间信息，好比温度在空间上的热传导

长短期记忆网络 LSTM：整合时间信息，好比时序电流的生热

普通网络：好比电池边界处，电池与空气或水的热对流

