文件结构：

--data

--train

--000

--001

--002

--003

--004

--test

--000

--001

--002

--003

--004

--readdata.py

--train.py

--Indrnn\_plainnet.py

--cuda\_IndRNN\_onlyrecurrent.py

--opts.py

--utils.py

--CNN.py

预处理：

运行readdata.py对数据集进行预处理，得到train\_label path.txt，test\_label path.txt，train\_label path\_add.txt三个文件，每一行记录了label和.npy文件的相对路径。其中train\_label path\_add.txt为进行了数据增强处理的文件与原文件的路径存储，进行数据增强得到的.npy文件储存于add\_data文件夹中。

训练与测试：

由于视频数据存在时间序列的相关性，我们首先尝试使用了参考资料中的IndRNN模型进行训练。

模型运行环境需要CUDA，可能会因为CUDA版本不一致导致报错，目前训练使用的CUDA版本为11.1。

Pycharm命令行输入运行train.py：

python -u train.py --model 'plainIndRNN' --bn\_location 'bn\_after' --u\_lastlayer\_ini --constrain\_U --num\_layers 3 --hidden\_size 512 --dropout 0.8 --batch\_size 64 --ifAugmentation --seq\_len 20 --use\_bneval（不进行数据增强时删除ifAugmentation参数：python -u train.py --model 'plainIndRNN' --bn\_location 'bn\_after' --u\_lastlayer\_ini --constrain\_U --num\_layers 3 --hidden\_size 512 --dropout 0.8 --batch\_size 64 --seq\_len 20 --use\_bneval）

cuda\_IndRNN\_onlyrecurrent.py, Indrnn\_plainnet.py, opts.py, utils.py来自[GitHub - Sunnydreamrain/IndRNN\_pytorch: Independently Recurrent Neural Networks (IndRNN) implemented in pytorch.](https://github.com/Sunnydreamrain/IndRNN_pytorch)下的action\_recognition文件夹，里面储存了适用于基于关节骨架的姿态分类任务的IndRNN模型的代码。实际模型运行时进行了一定参数调整。

我们摒弃了此模型输入数据的部分，直接运用pytorch的Dataset，Dataloader输入数据。由于输入数据集样本量较小，我们重新写了模型训练、测试的运行代码（train.py），启用了BN层，取消了原本训练模型时调整学习率的部分，并且建议bathes，batch\_size取30,64以获得较好的训练效果。

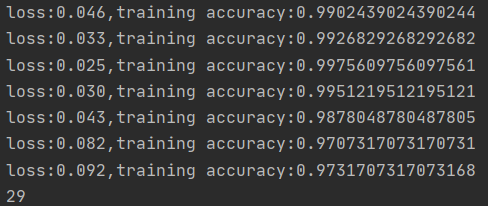
cuda\_IndRNN\_onlyrecurrent.py调用CUDA进行加速计算，train.py运行IndRNN模型，模型框架和细节储存于Indrnn\_plainnet.py、utils.py，opts.py设置运行参数。

由于数据集较小，我们尝试了数据增强，应用效果较好。

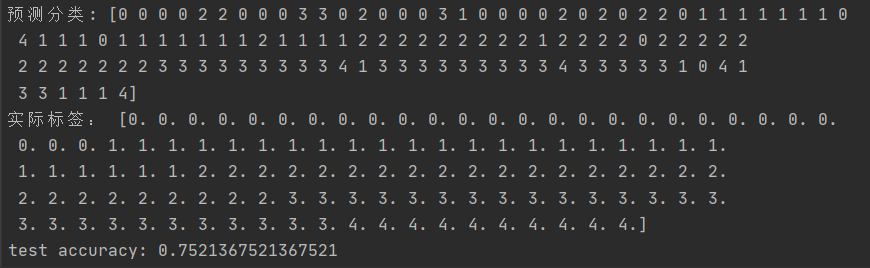
运行结果：

batches=30，batch\_size=64，无数据增强：

最后一轮训练：

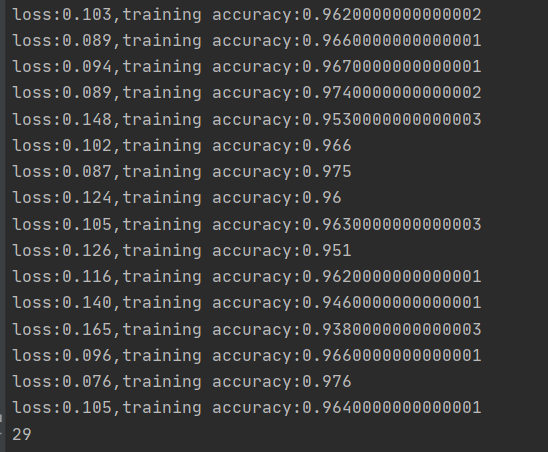


测试结果：

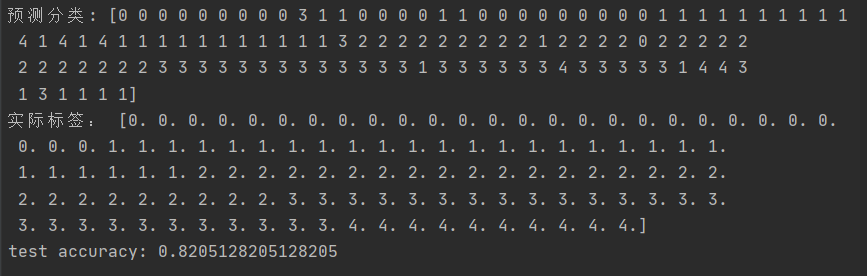


batches=30，batch\_size=64，有数据增强：

最后一轮训练：



测试结果：



模型细节：

训练输入维度17\*3，输出分类5，3个隐藏层，每层512个参数，dataloader的shuffle=False。

输入数据形式为[320,10,51]，320为视频帧维度，10为每次训练的batchsize大小，51为合并后的关节三维坐标数据。

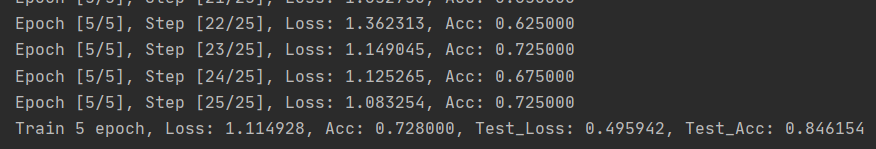
测试时输入batchsize=1，否则每一批的输入数据会输出同一个分类结果。训练时输入dataloader的shuffle=False，否则会出现难以拟合的情况。

训练沿用了原模型的嵌套循环调用训练，训练次数为batches\*(len(train\_dataset)/batch\_size)，即训练次数随训练集大小增加而增加。

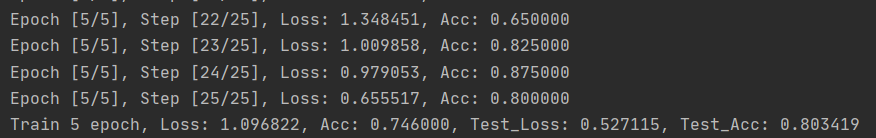
学习率=2e-4，采用交叉熵损失函数和Adam优化器。

CNN：

由于IndRNN模型较大且复杂，训练次数较多，而使用数据集较小。因此我们同时尝试训练了一个CNN模型进行分类任务（CNN.py），摒弃了数据集序列上的关联，并且每个npy数组只取前40帧数据。结果显示CNN在此数据集上同样表现良好，训练拟合速度快，预测准确率较高。



我们同样尝试了进行数据增强，发现对于CNN模型而言数据增强效果并不好。



模型细节：

第一层卷积层：输入channel：40，输出channel：80，卷积核2\*2

第二层卷积层：输入channel：80，输出channel：60，卷积核2\*2

第一层全连接层：输入900，输出200

第二层全连接层：输入200，输出80

激活函数为ReLu函数，学习率设置为1e-4，采用交叉熵函数和Adam优化器。