ML Lab3 Report

高健翔521021910197

1. 系统设计
   1. 模型设计

采用CNN-LSTM模型：

CNN部分：

对于输入的时空序列数据首先通过卷积神经网络进行特征提取，CNN捕捉数据的空间局部特征，处理视频不同帧的空间结构数据，在我的CNN模型设计中，用到卷积层提取特征，池化层进行维度降低，激活函数加入非线性因素；使用的预训练模型是reenet101。.

LSTM部分：

对于输入的序列，CNN已经将特征提取出来，LSTM负责处理输入数据的时序特征，对于图像序列，LSTM将序列中的帧是为时间步，并学习帧之间的时序关系，LSTM具有门控结构，可以决定记忆或者遗忘的内容，以传递先前时间步的信息。

最终，将LSTM的输出作为全连接层的输入，输出维度为类别数量，哪一维度的值越大，表明在该类型的概率越高，取最大的下标作为最终的分类。

* 1. 训练方法

对于分类任务，选择交叉熵函数作为Loss\_Function

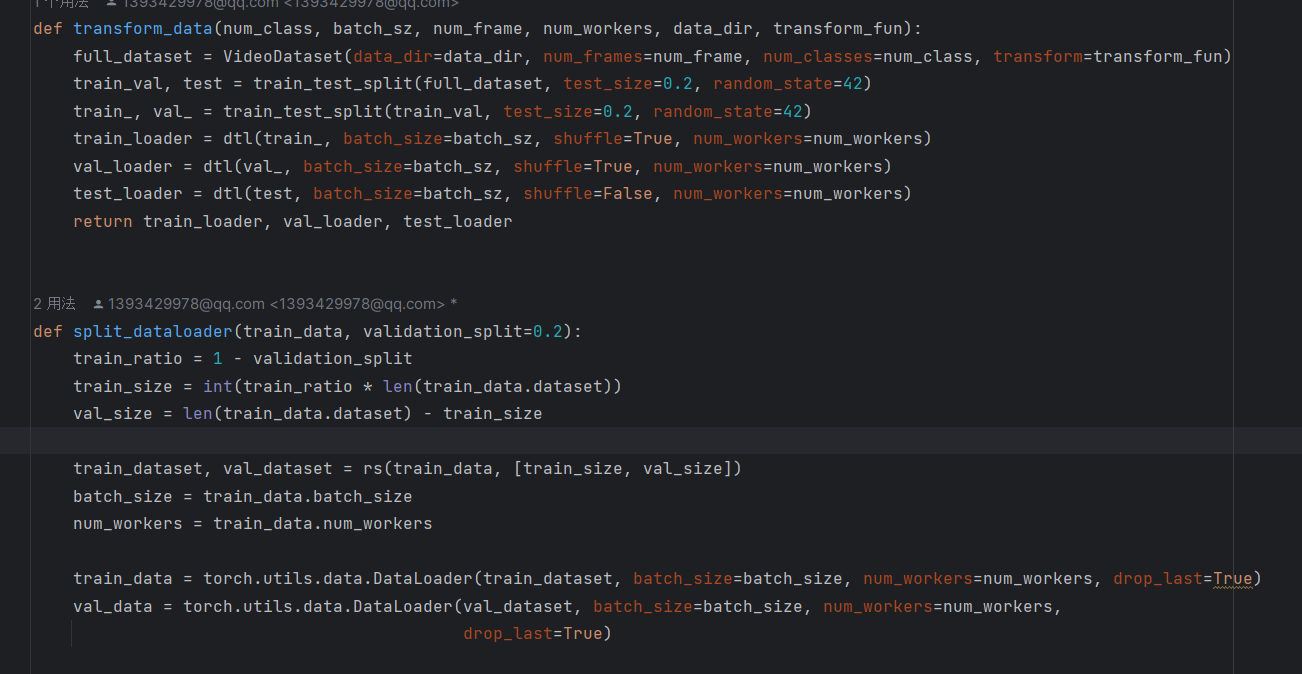
优化器选择adam优化器

自行选择LR调整策略，本次实验使用了ReduceLROnPlateau和CosineAnnealingWarmRestarts

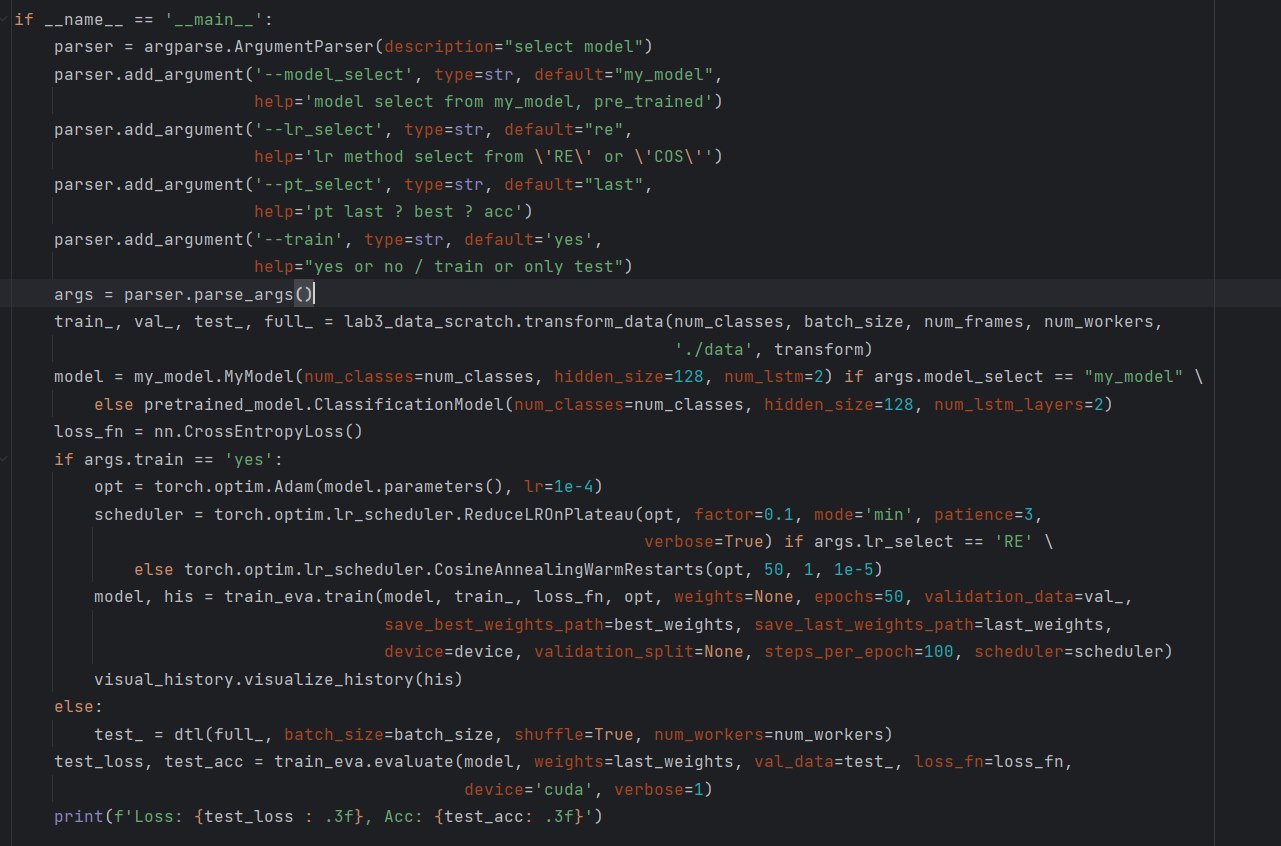
重写数据集划分函数，在准备过程中便将数据集分割为train，val，test三个模块

自行构造CNN-LSTM模型，并参考他人代码构造基于预训练网络的模型

下面是对代码的重构：



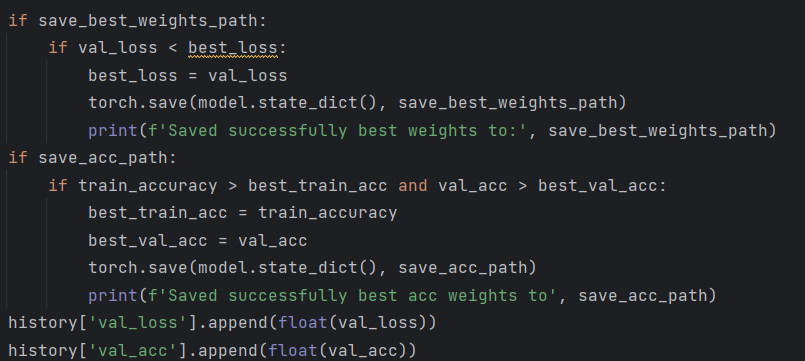
数据分割部分：参考老师的代码，重新写出了将dataset分为train，val，test三个dataset并构造dataloader的函数；参考kaggle网站代码，使用torch random\_split函数重新构造分割dataloader的代码，当train函数的验证集参数为空时，调用该函数从训练集分割出一部分



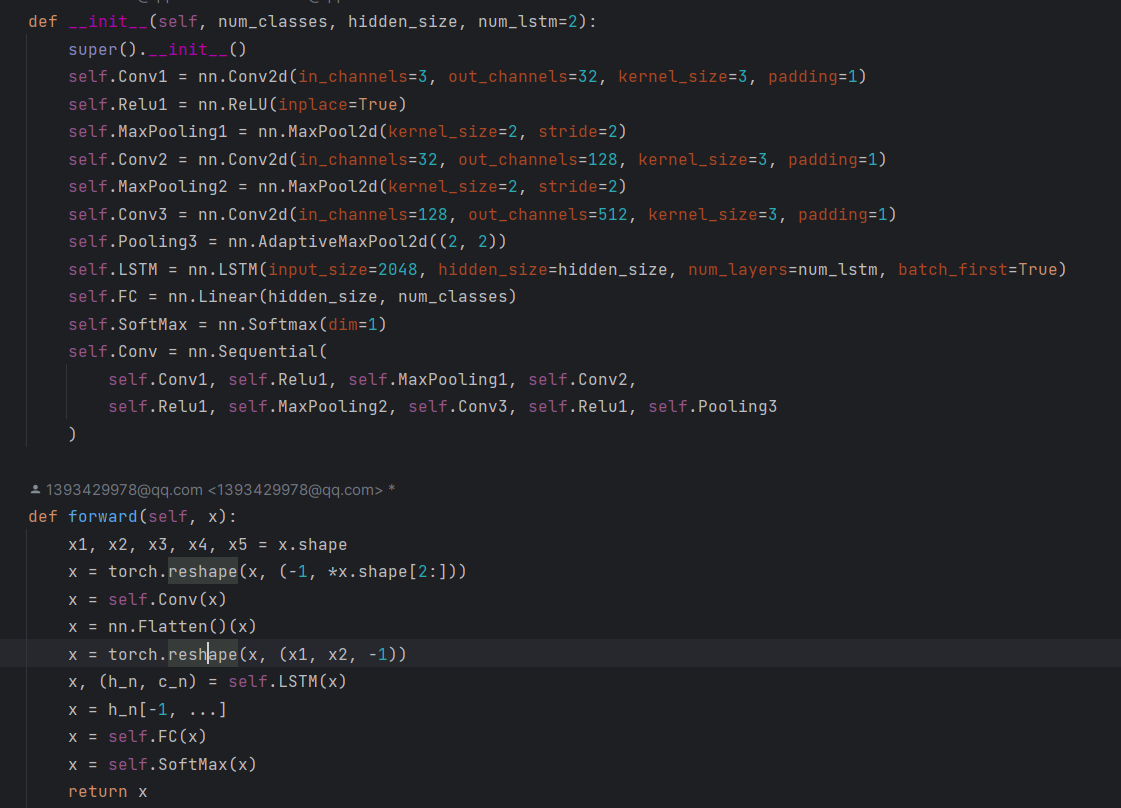
增加main.py文件负责参数解析，配置相应的优化器，lr调整策略和模型选择，然后调用训练，测试函数，最终输出结果



读取单一视频文件，使用模型进行分类



增加了一个根据训练集，验证集准确率来保存模型参数的部分



自行设计模型

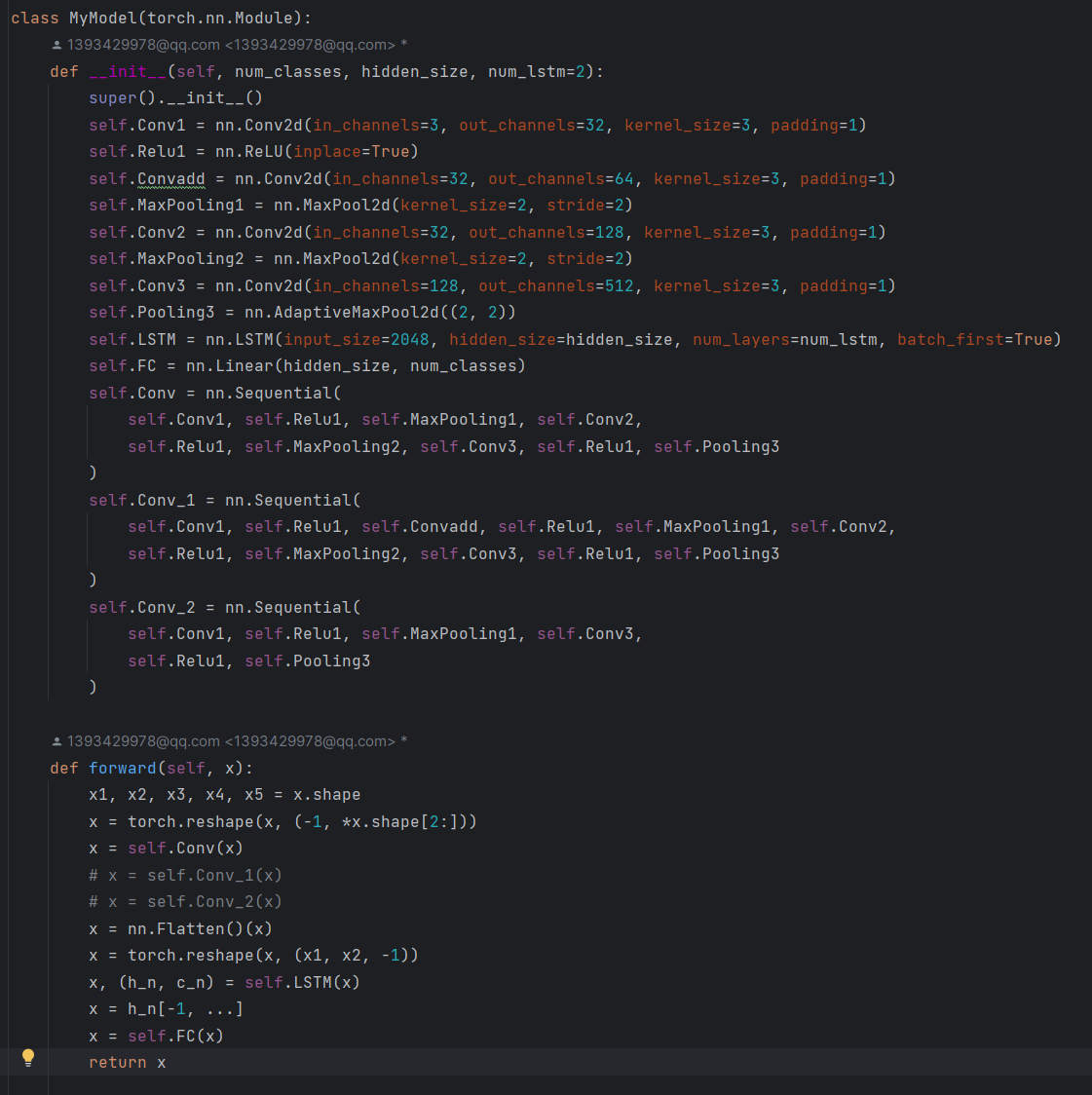


参考代码给出的模型，观察到backbone函数返回的张量有多种，之后会尝试不同的张量形状

1. 实验结果
   1. 自定义CNN-LSTM

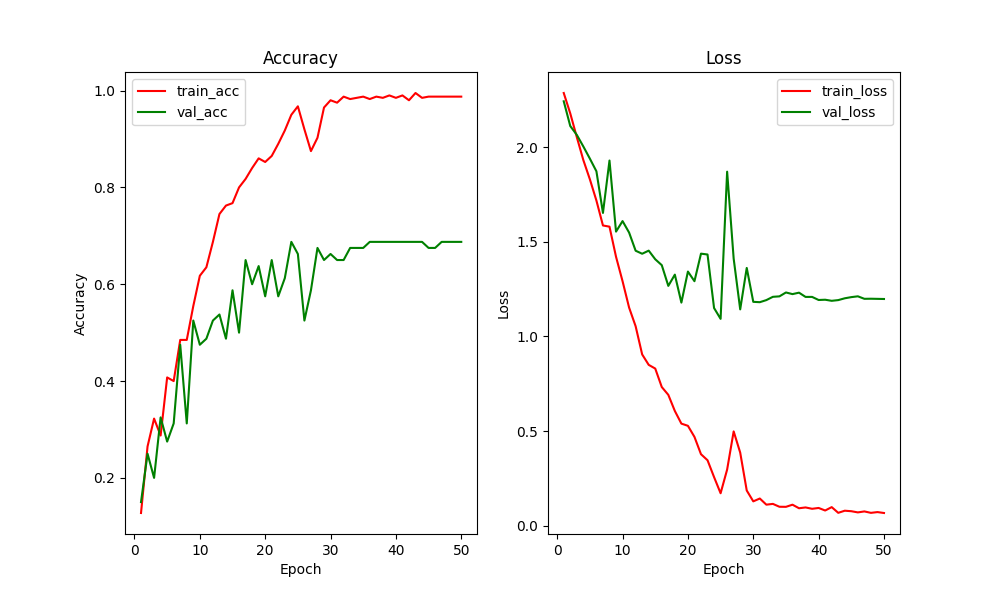
Size: 128 \* 128 frame: 40 epoch: 50 batch\_size: 4 lr\_scheduler: ReduceLROnPlateau lr: 1e-4

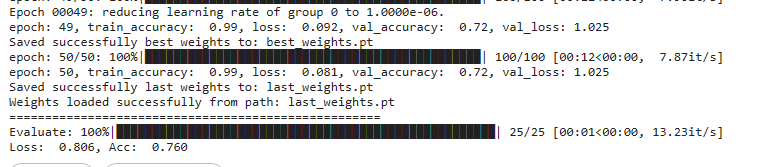
模型为：



* Self.Conv self.Conv\_1 self.Conv2对应不同的CNN网络 本次采取self.Conv

训练过程及结果：

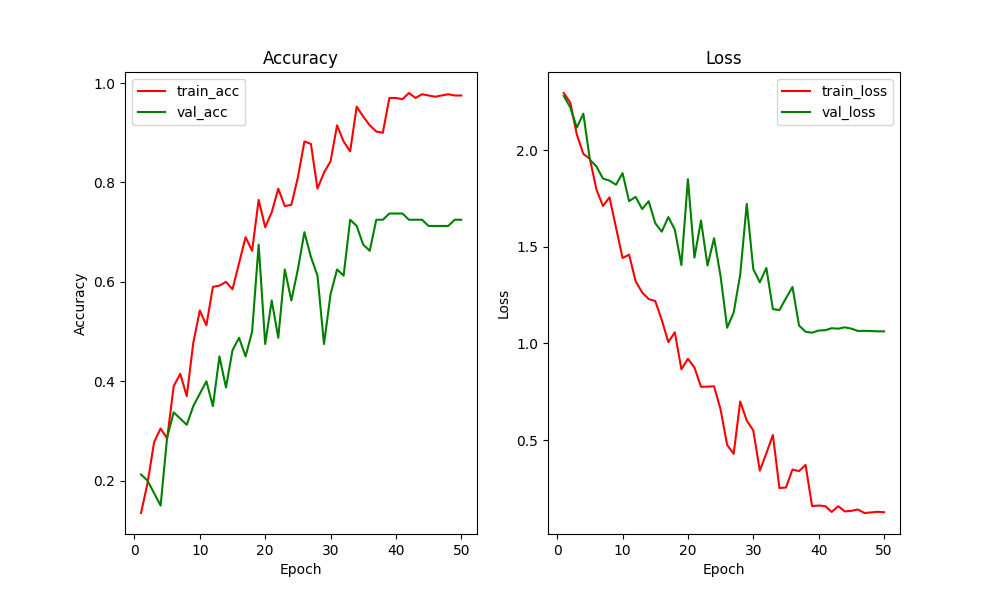


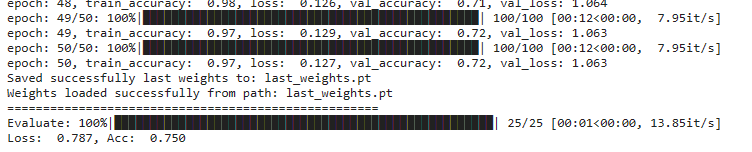


测试集中Accuracy为0.76 Loss为0.806满足对自定义CNN的准确率要求

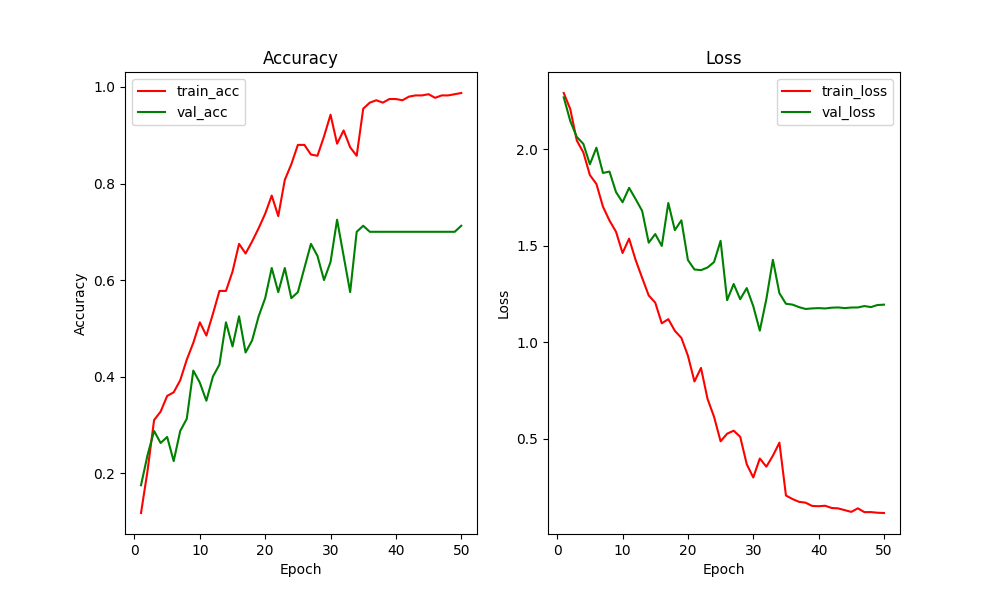
不同参数/CNN形状下的结果：

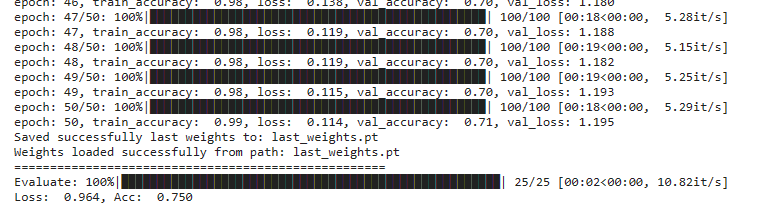
* + - 1. 调整网络结构(需要在自定义的模型下修改)
         1. 增加一层LSTM：



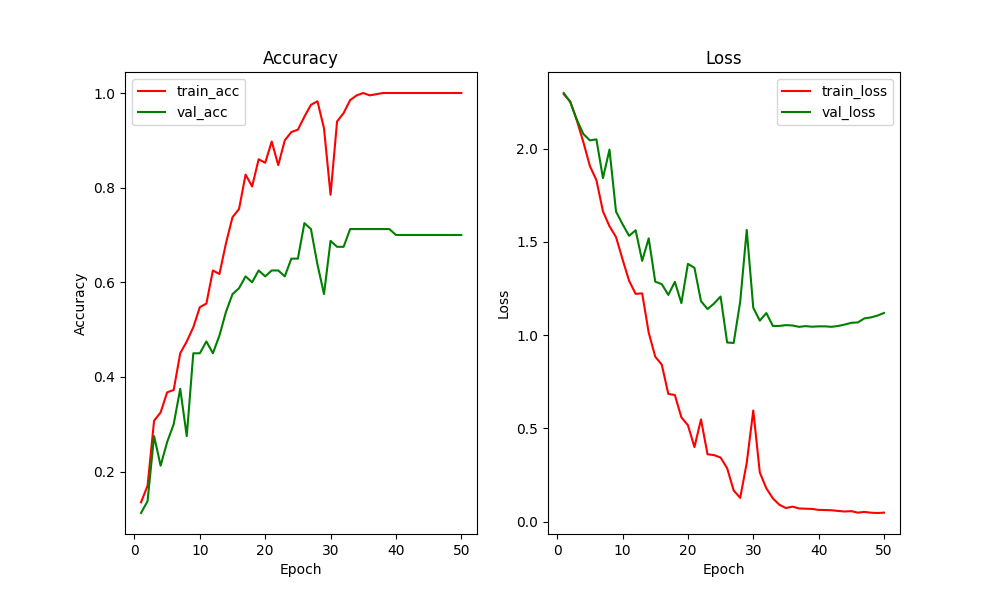


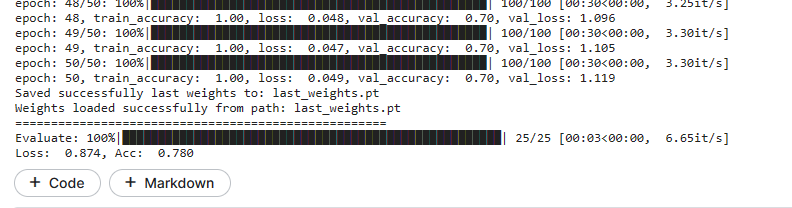
* + - * 1. 增加一层Conv2d(self.Conv\_1)



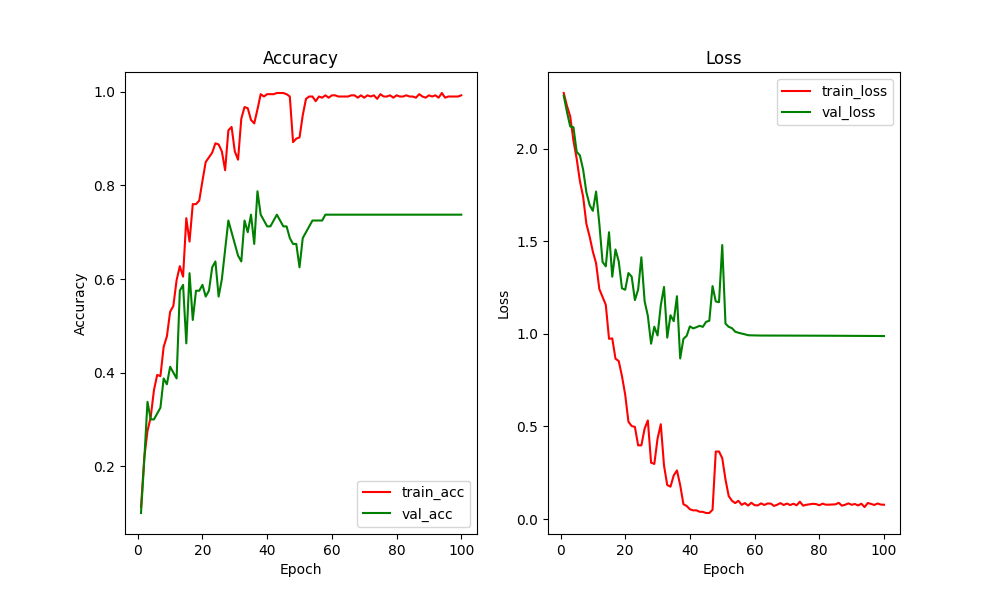


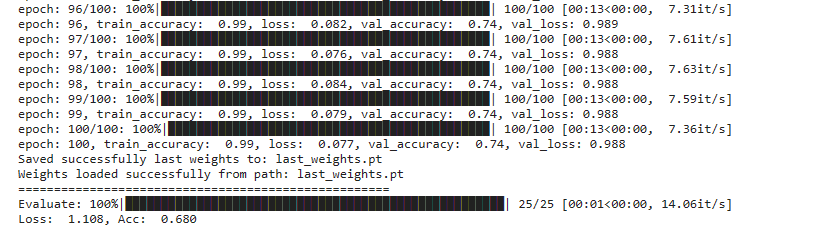
* + - * 1. 减少一层卷积



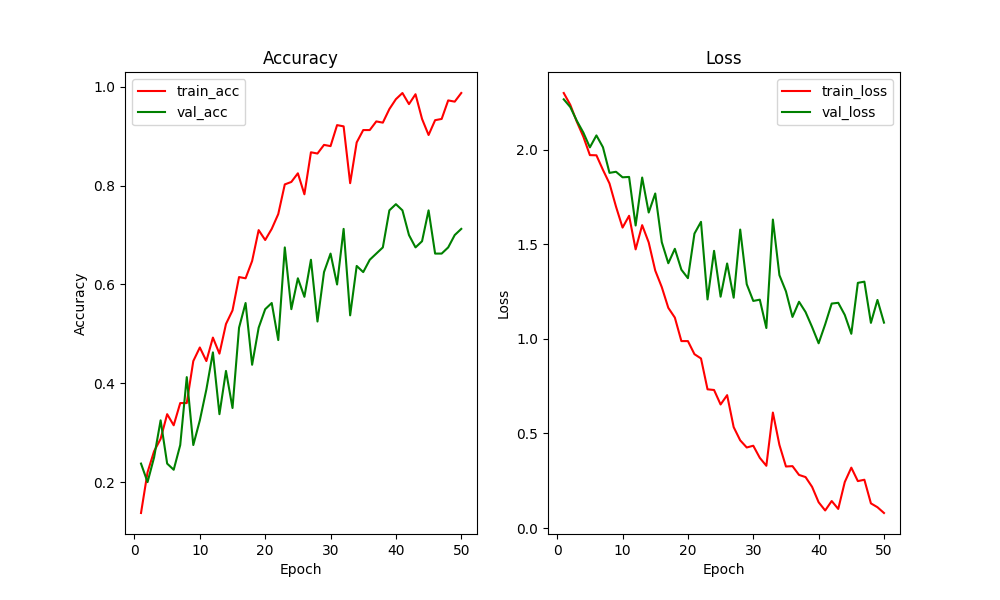


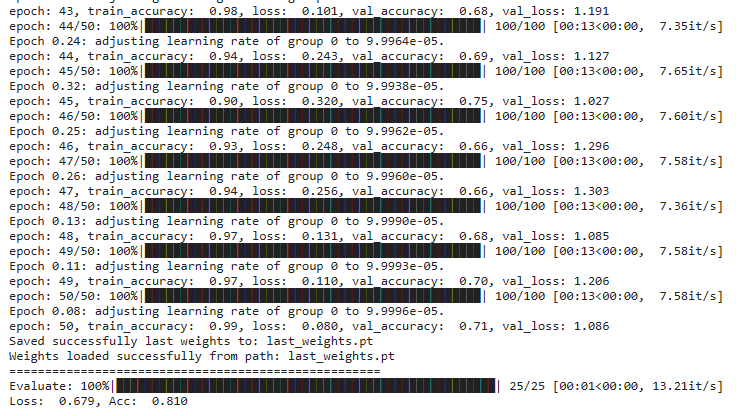
* + - 1. 调整训练次数(本实验将训练次数调整至100次)
         1. 实验结果：



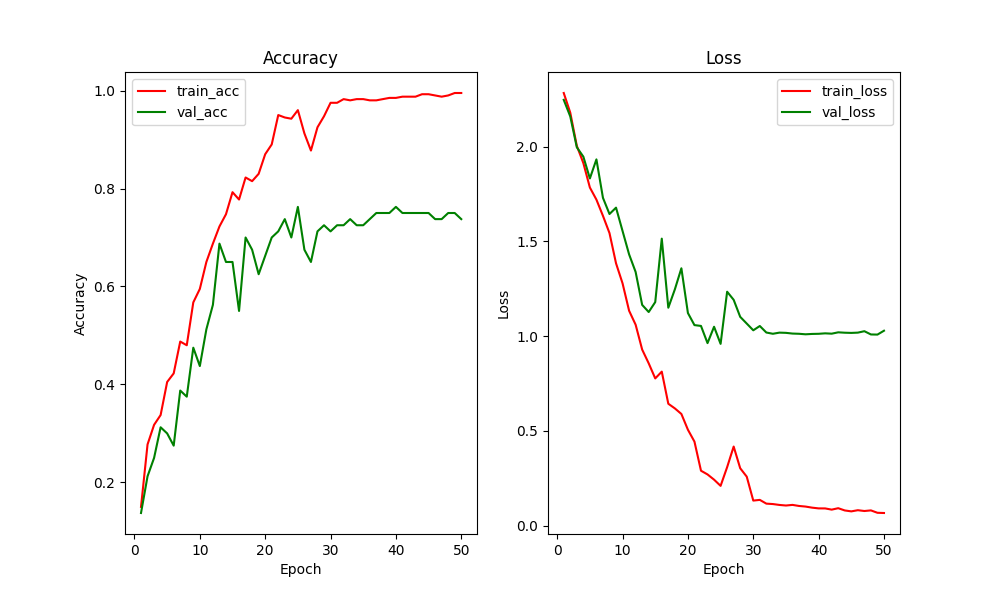


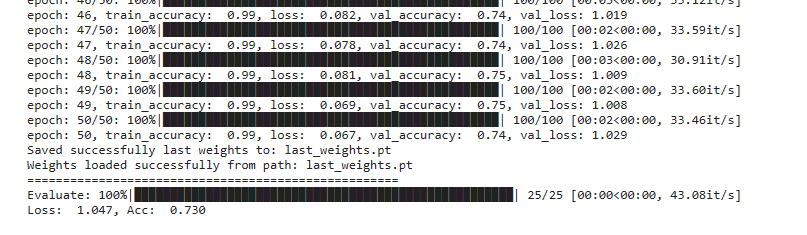
* + - 1. 调整lr优化方法(采取余弦退火的调整策略，但是参数好像没配置太好)
         1. 实验结果：



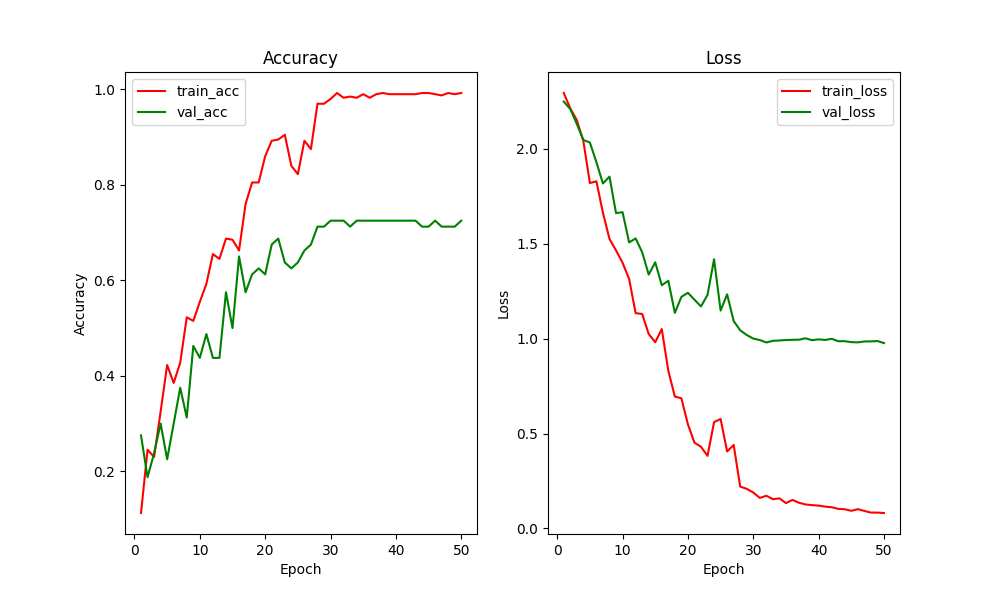


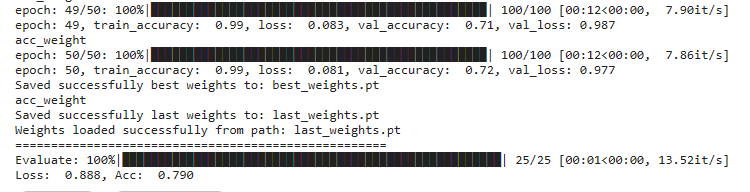
* + - * 1. 观察到图像似乎并没有完全收敛，再增大epoch补做一组
      1. 调整视频frame ，size
         1. 调整至frame 30，size(64, 64)



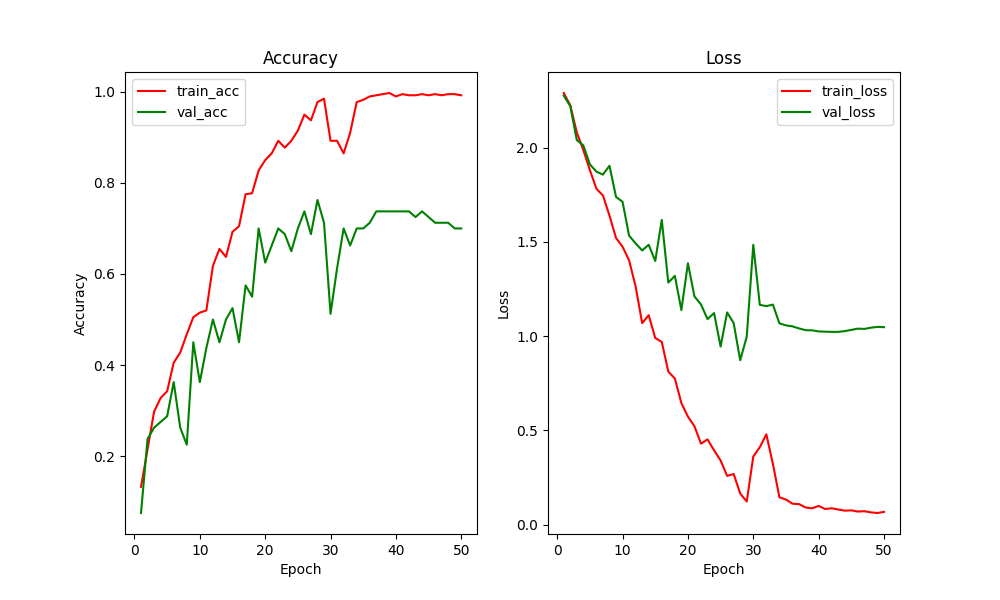


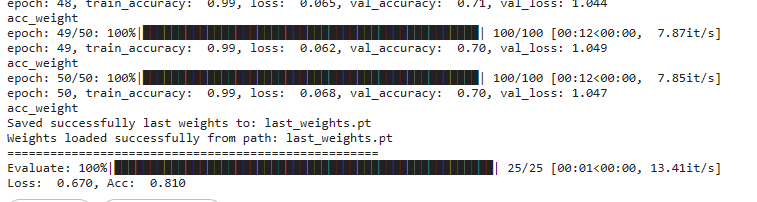
* + - 1. 调整学习率：
         1. 减小学习率至5e-5





* + - * 1. 增大学习率至2e-4

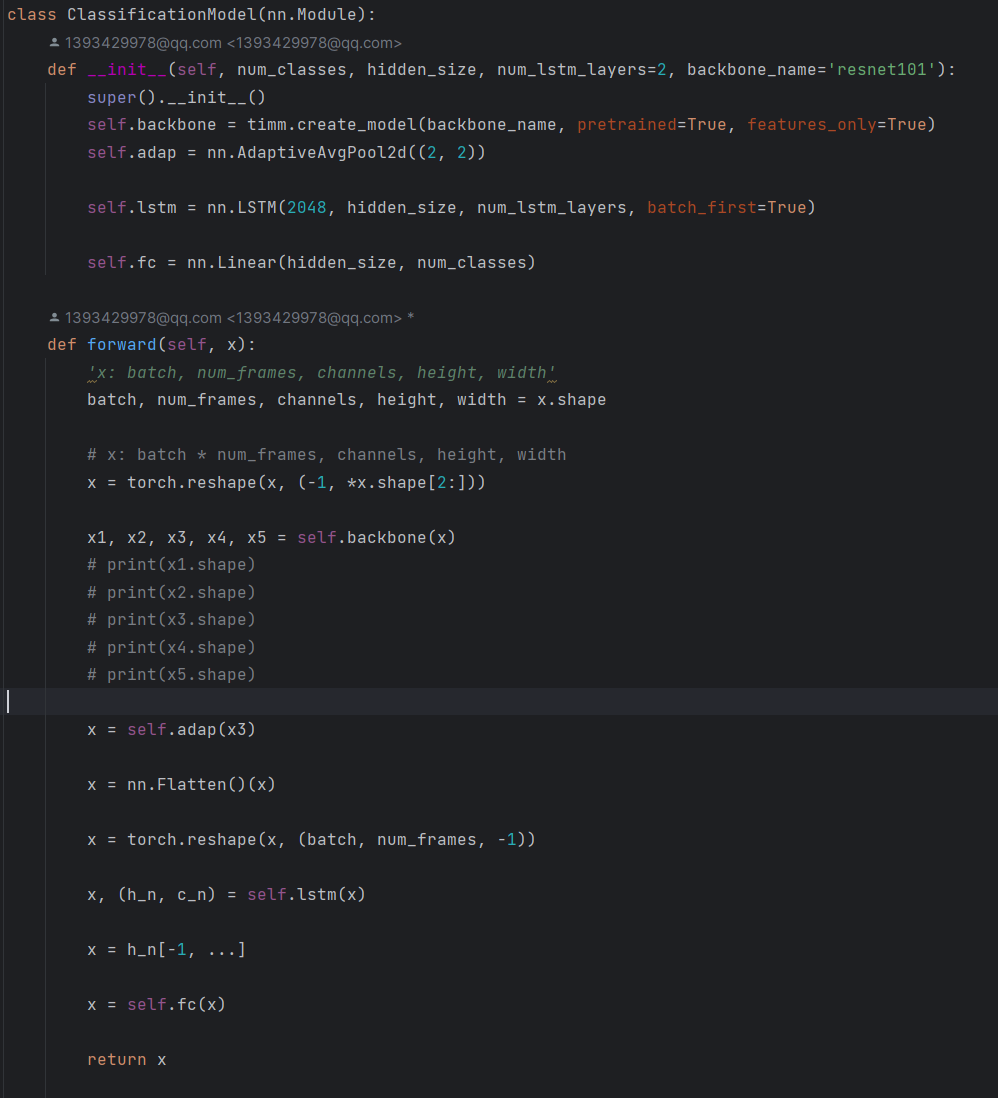




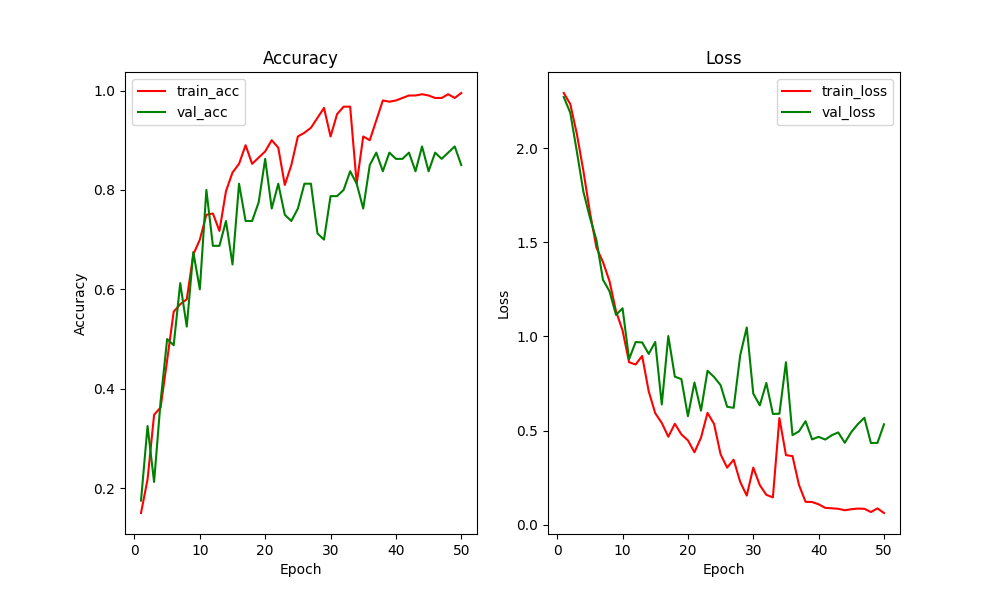
* 1. 在预训练模型上增量训练

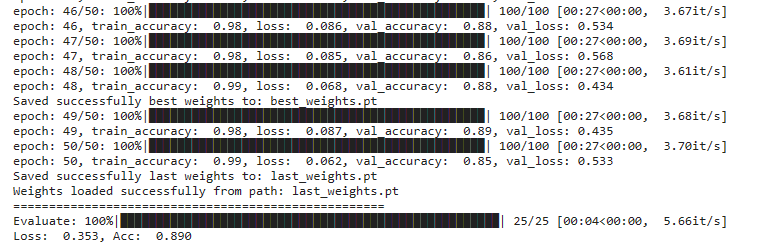
参数：resnet101，输出通道数为512，2层lstm，size(128, 128), frame: 64, epoch: 50

模型：首先采用实例代码给出的模型



训练过程及结果

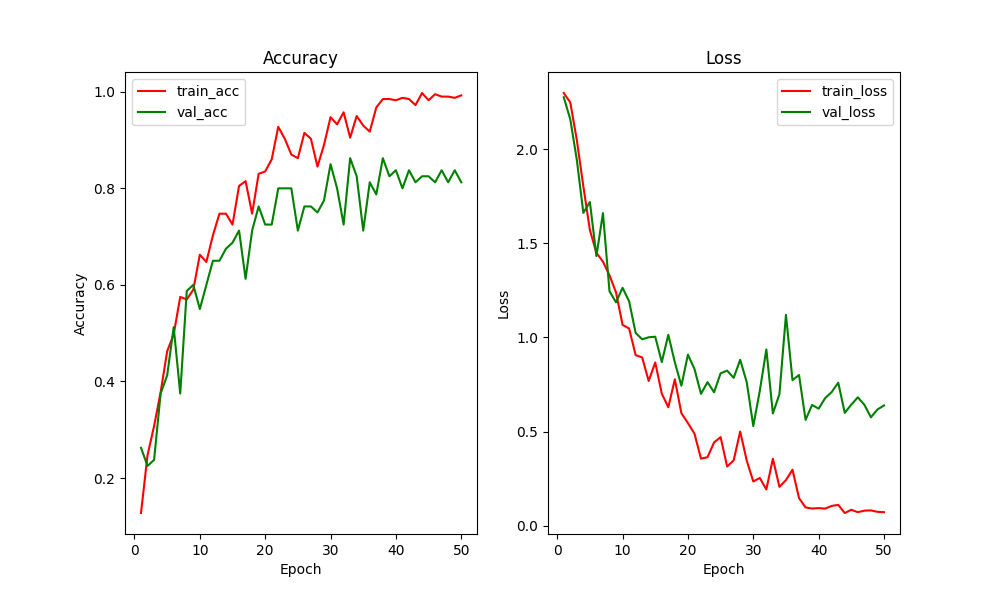


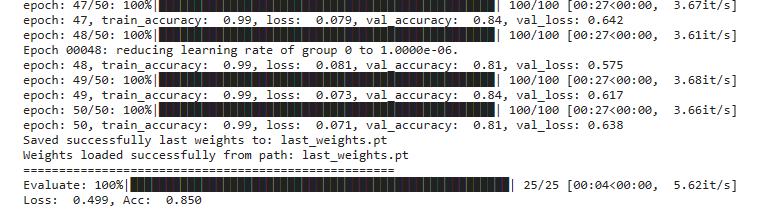


测试集loss 为0.353 accuracy为0.89 符合实验对准确率的要求

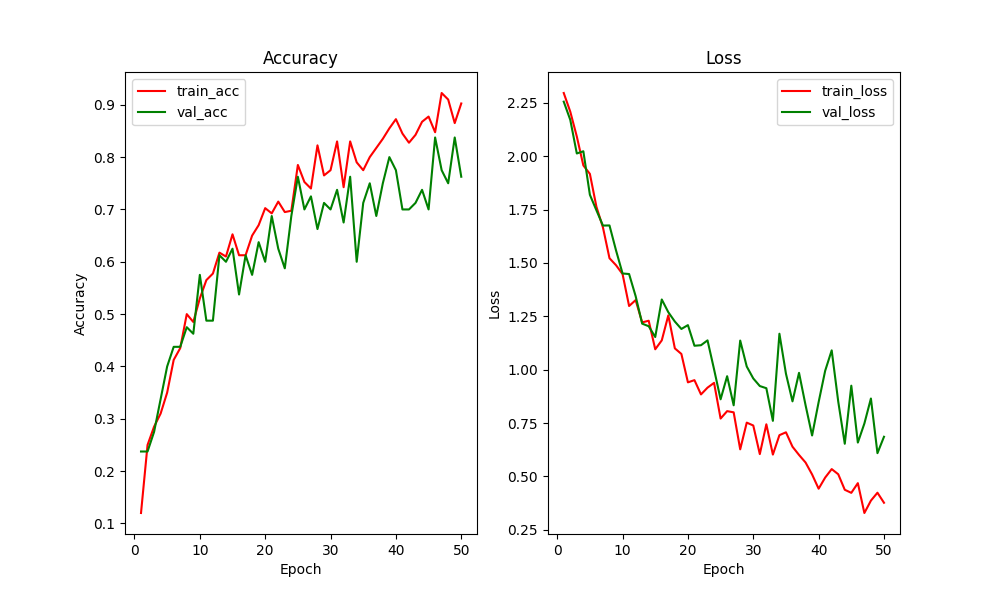
不同参数下的结果：

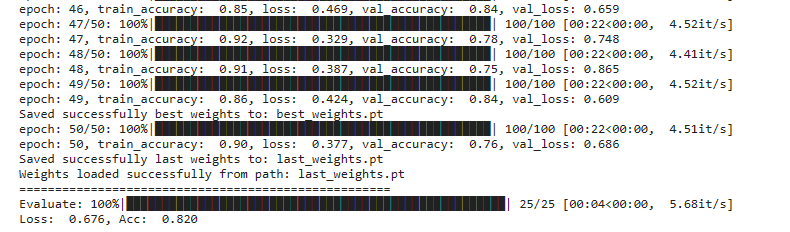
* + - 1. 调整网络结构
         1. 增加一层lstm：



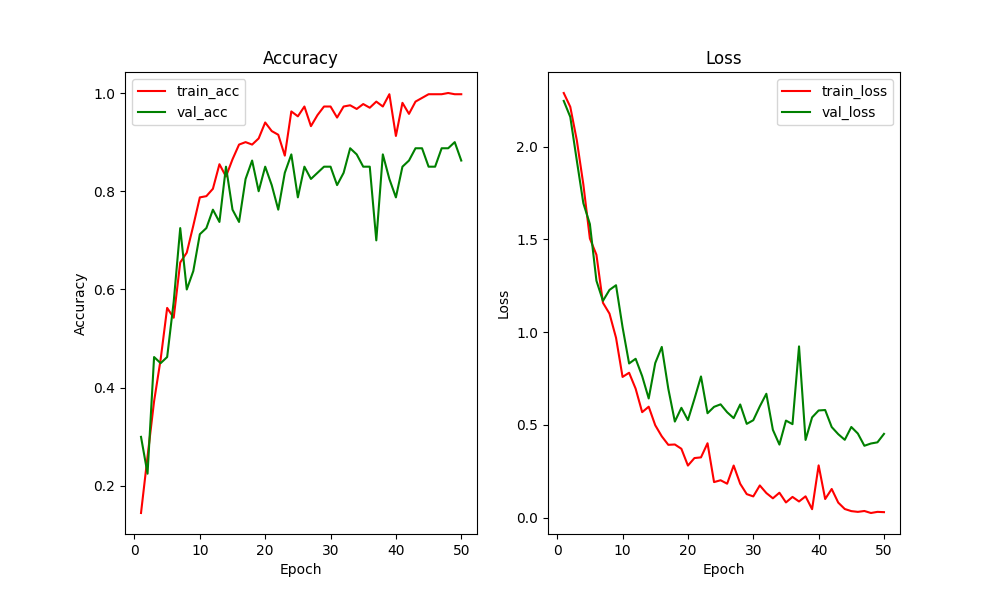


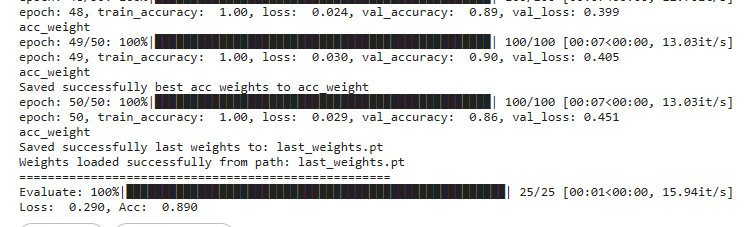
* + - * 1. 使用输出为256通道的预训练网络：



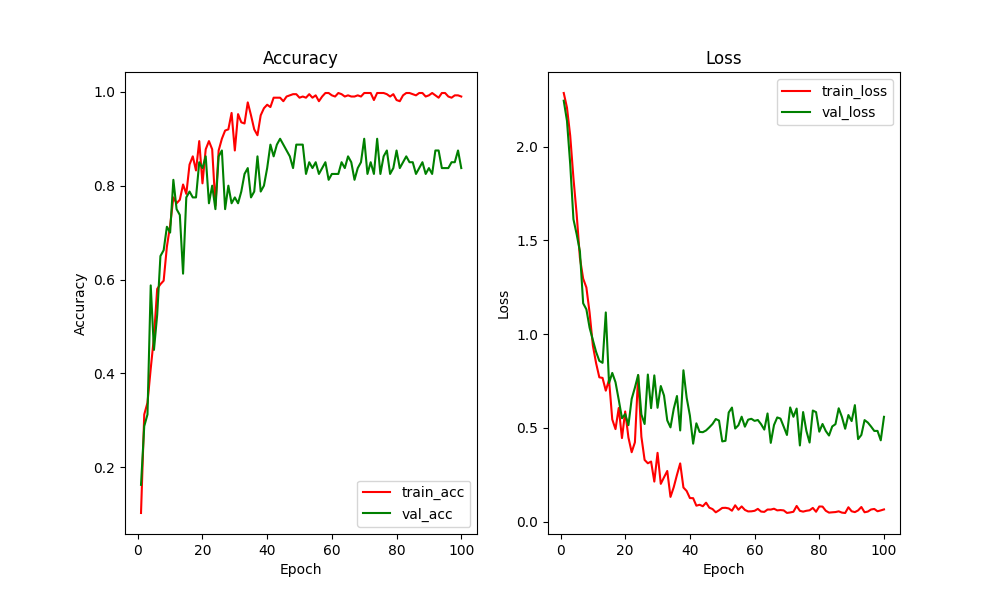


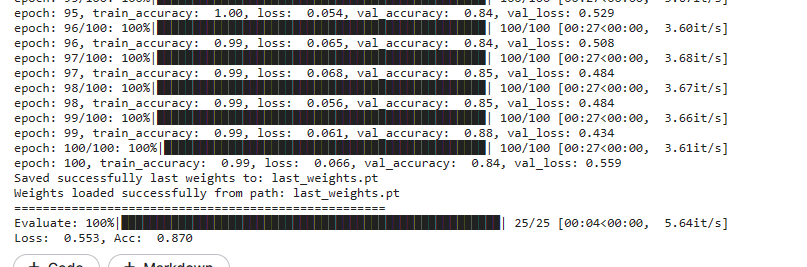
* + - 1. 调整视频frame ，size
         1. 调整为frame: 30, size: (64, 64)





* + - 1. 调整训练次数
         1. 调整至100次



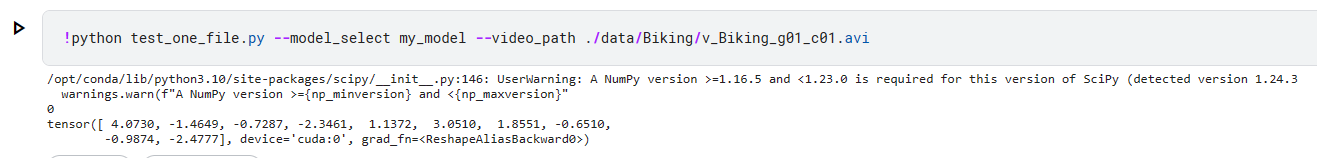


* 1. 单独取出一些视频进行分类：(选择的是第一类 0，第二类的一个视频 1)

****

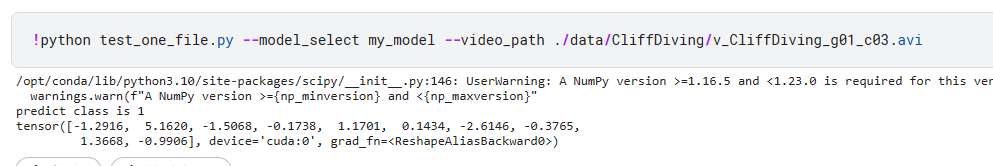
自行编写的程序，用于测试单一视频文件(上面已经列出)

* + 1. 原始自定义CNN

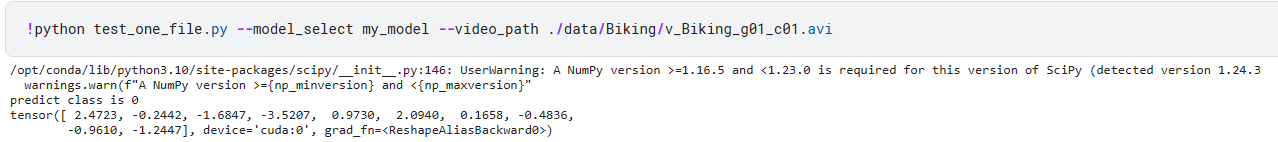


分类结果

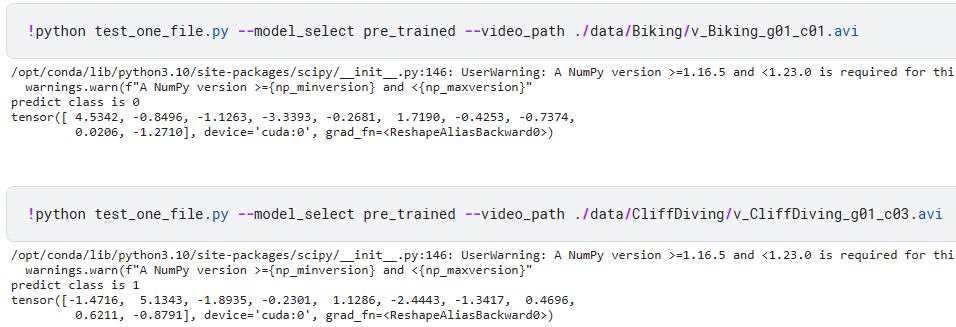
* + 1. 调整一些参数的自定义CNN(不是全部)
       1. 减少一层CNN



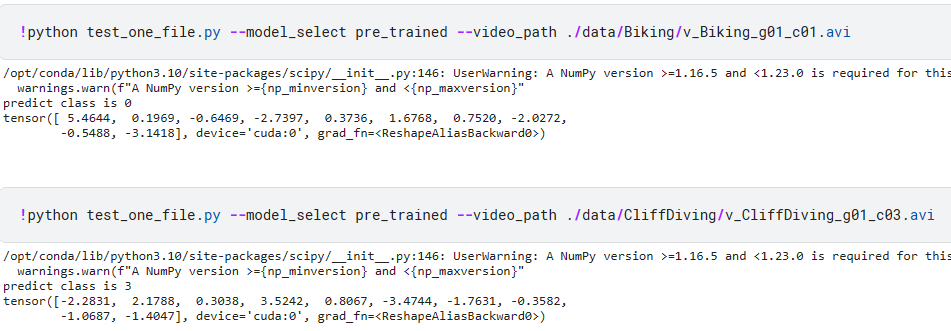
* + - 1. 增加一层CNN



* + 1. 基于预训练模型的
       1. 示例模型：

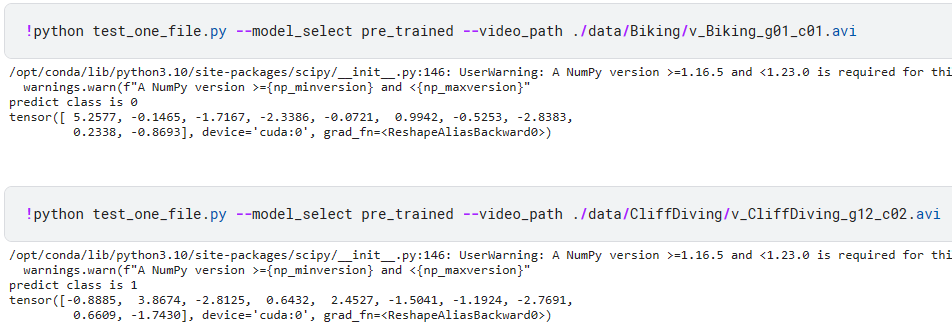


* + - 1. 增加一层lstm



出现错误预测

* + - 1. 增加至100训练次数



1. 一些简单的总结
   1. 自定义网络在验证集上的loss和准确率一直与训练集相差较大

在我自己的网络中，可能出现了过拟合的情况，导致验证集上虽然收敛，但是收敛点并不符合预期，可能是我在自定义的模型中学习率直接套用了在预训练模型上的学习率，导致陷入局部最优的情况，无法找到最优解(在最后对学习率略作调整的部分，发现效果比原始的好一些)

* 1. 使用余弦退火调整策略在自定义网络的效果好一些

在对实验过程输出的观察中，我发现原始的ReduceLROnPlateau并没有及时调整学习率，因为很多时候验证集的loss都在下降，但是下降的幅度很低，这无法触发学习率调整，因此采用余弦退火的方式，按照余弦周期进行学习率调整，学习率在每个epoch都会稍微发生变化

* 1. 增加网络层数，或者使用更大的frame size的效果并不一定更准确

本实验的数据集数量可能并不是很多，增大网络复杂性，或者视频的帧数尺寸，容易导致过拟合的情况，时空信息的表示也不是越准确越好，有时忽略一些不关键的视频信息对于提高模型的泛化性也有所帮助

总之，本次实验涉及到的参数过多，我没有办法在某个参数上做很多的测试，并且每次训练的结果都有一点不确定性，上面的数据也是多次测试后取出的比较中肯的结果，虽然本次实验不能得到一般性的结论，但还是完成了要求的任务，并且尝试了对不同参数的调整和对不同模型的构造与选择，也增进了的对机器学习实验中”微调”的理解。