|  |  |
| --- | --- |
| **计算机视觉与模式识别**  **实验报告** | |
|  | |
|  |  |
| **名称** | Homework5\_img\_classification |
| **姓名** | 卢佳源 |
| **班级** | 人工智能91 |
| **学号** | 2191121196 |
| Email | bjlujiayuan@126.com |
| **日期** | 2021-12-9 |

# 实验目的

基于Caltech 256数据集的深度网络分类性能比较：

1. 以Caltech 256为实验数据集，按照原始数据集的要求进行训练/验证/测试集的分类；
2. 查找参考代码（如Github等），用至少一种深度网络结构，全部类别的平均精度不低于85%；
3. 复现参考论文或代码；
4. 对算法从处理速度、正确性等方面给出性能评估；
5. 阐述算法的简要描述、实验设定、实验结果等。

# 实验环境

OS：Linux操作系统的ubuntu20.04版本

IDE：VScode,

anaconda3(‘base’:conda), anaconda3(‘sBERT’:conda)

Terminal：gnome-terminal -t $TITLE -x

Language：python(3.8.8 64-bit), python(3.6.2 64-bit)

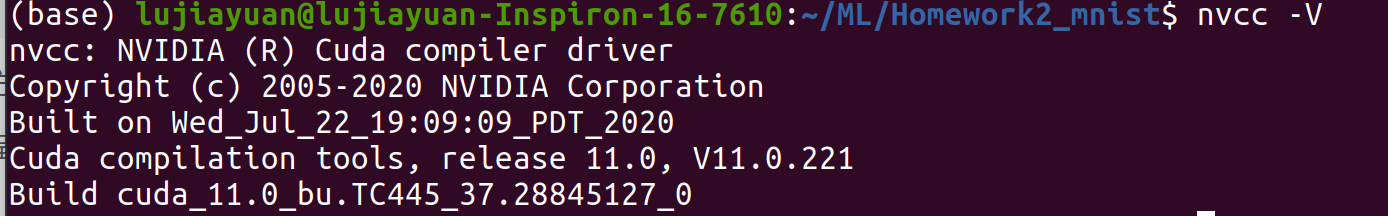
PATH：

code：/home/lujiayuan/cvpr/Homework5/preprocess.py & dataloader\_50.py & dataloader\_152.py

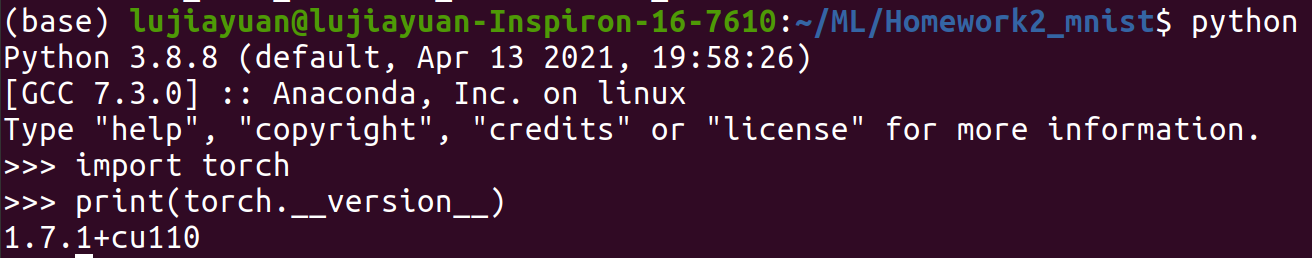
ResNet：/home/lujiayuan/anaconda3/lib/python3.8/site-packages/torchvision/models/resnet.py（做了调整）

Data:/home/lujiayuan/cvpr/Homework5/Data/256\_ObjectCategories

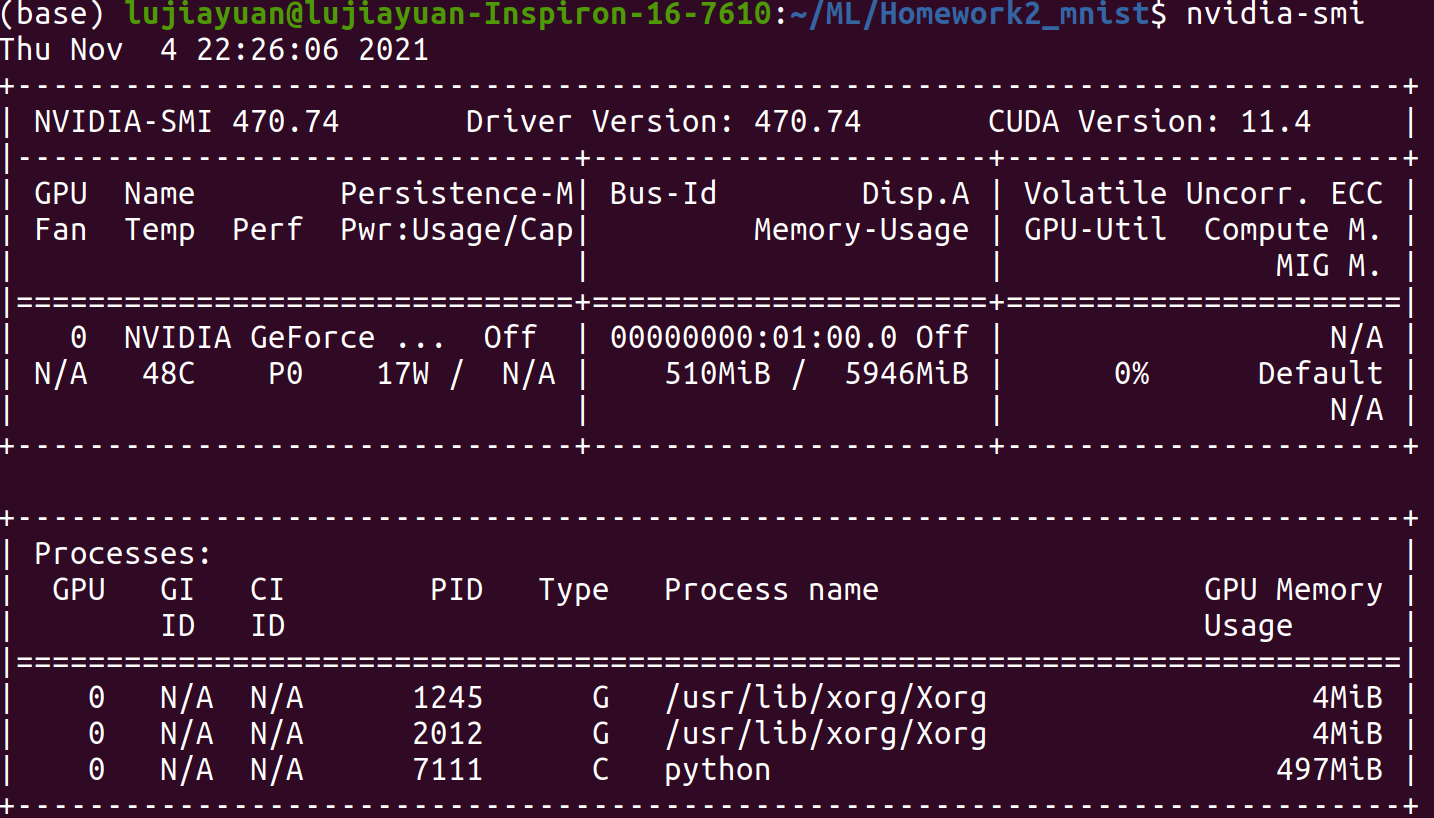
CUDA版本：



pytorch版本：



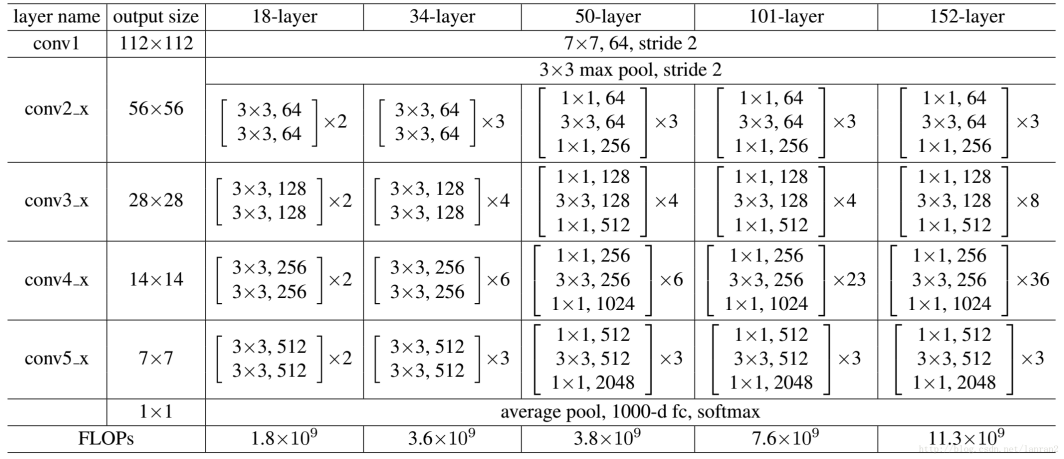
显卡信息：Name: GA106M [GeForce RTX 3060 Mobile / Max-Q]



# 实验方法

1. 数据集划分：代码见preprocess.py
2. 验证集：取每一个类别文件夹的前十张图像，将图像路径保存在dataset-valid.txt中；
3. 测试集：取每一个类别文件夹的第10～40张图像，将图像路径保存在dataset-test.txt中；
4. 训练集：取每一个类别文件夹的第40之后的图像，将图像路径保存在dataset-train.txt中；
5. 神经网络选择与实现：
6. ResNet50和ResNet152的比较：

（1）本次实验中，我选择了resnet50和resnet152这两个残差神经网络进行分类性能比较。选择这两个深度神经网络的原因，是它们属于同一种框架的神经网络，仅在网络层数上有差异，因此在分析性能时可以做到一定程度的控制变量，使得分析结果更加有逻辑和说服力；



图中，矩阵内部表示BasicBlock（18 & 34）或Bottleneck（50以后，降低参数量），矩阵内一行代表做一次卷积运算，ResNet名字由来（以152为例）：

conv1：卷积1\*1=1次；

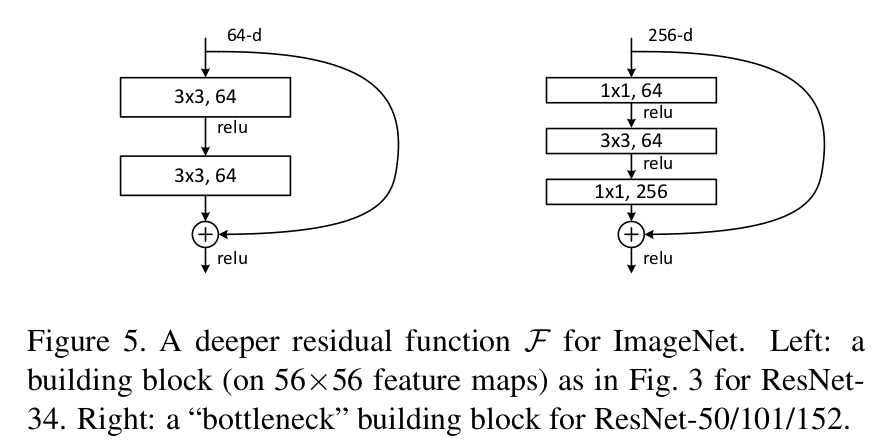
conv2\_x：卷积3\*3=9次；

conv3\_x：卷积3\*8=24次；

conv4\_x：卷积3\*36=108次；

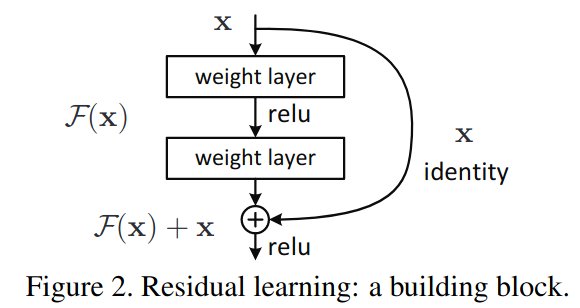
conv5\_x：卷积3\*3=9次；

则1+9+24+108+9=152，即做了152次卷积运算。



1. 残差网络模型简述：

ResNet是将靠前的某一层输出数据直接开辟一条分支作为这一层后面的若干层的某一层的输入，即将较前面网络层可以为较后面特征层提供一定的线性贡献，解决了退化问题，即由于网络深度加深而导致的学习效率变低以及准确率无法有效提升的问题。该模型结构如下：



以ResNet50结构为例：

1. Conv Block：

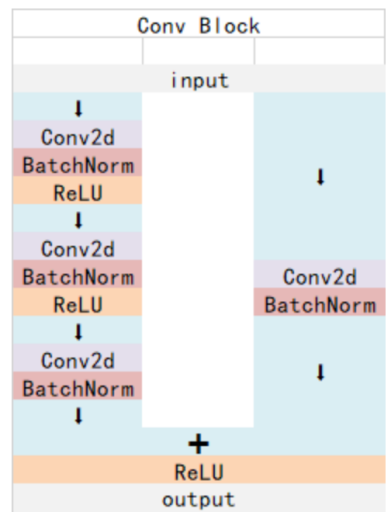
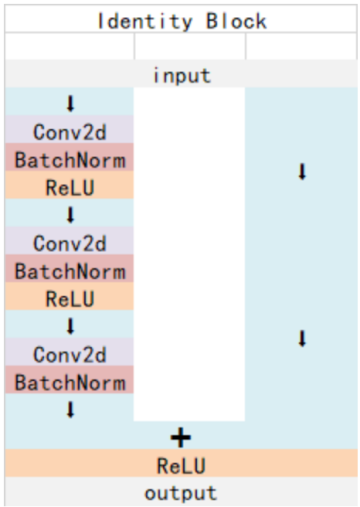
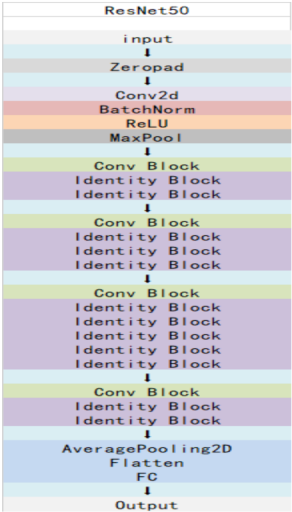
输入和输出维度不同，不能连续串联；

作用：改变网络维度；

1. Identity Block：

输入和输出维度相同，可以串联；

作用：加深网络；

1. 实现过程：
2. 方法1：直接调用torchvision的model模块中的预训练好的resnet函数：

resnet50 = models.resnet50(pretrained=True)

产生问题：该模型需要从pytorch官网上下载，速度十分慢，此方法不可取；

1. 方法2：利用FlyAI平台提前下载好resnet预训练好的模型（需要另外配置FlyAI的平台requirement，如python版本只兼容3.5～3.7，并且需要下载flyai-gpu等），将模型的.pth文件放到代码路径上，之间用torch.load函数加载模型进行训练（更改了/home/lujiayuan/anaconda3/lib/python3.8/site-packages/torchvision/models/resnet.py的原文件，代码修改如下），并保存每一轮训练后的模型，用于对测试集的预测；

def resnet50(pretrained=False, progress=True, \*\*kwargs):

path='/home/lujiayuan/cvpr/Homework5/resnet50-19c8e357.pth'

model=ResNet(Bottleneck, [3, 4, 6, 3],\*\*kwargs)

if pretrained:

model.load\_state\_dict(torch.load(path))

return model

def resnet152(pretrained=False, progress=True, \*\*kwargs):

path='/home/lujiayuan/cvpr/Homework5/resnet152-b121ed2d.pth'

model=ResNet(Bottleneck, [3, 8, 36, 3],\*\*kwargs)

if pretrained:

model.load\_state\_dict(torch.load(path))

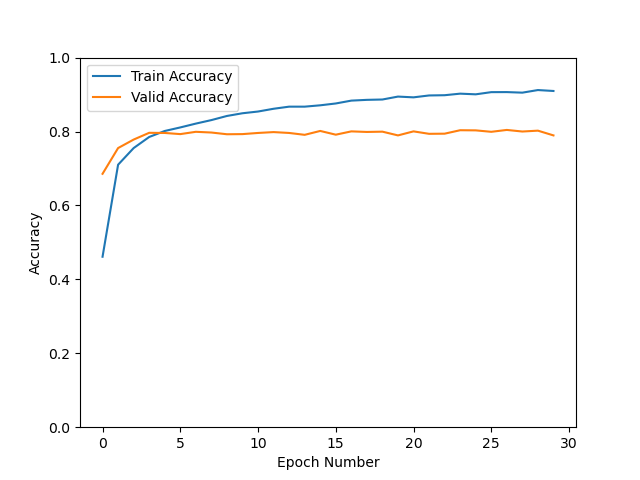
return model

1. 实验思想 & 代码（dataloader.py）流程：
2. 建立训练集、验证集、测试集；
3. 加载模型，将残差网络的最后一层替换为全连接层，定义损失函数和优化器函数；
4. 对训练集进行训练，并用验证集进行模型超参数调整，计算训练和验证集损失率和准确率，并画图；
5. 将准确率最高的一次训练保存的模型加载上来用于测试集的预测，并计算训练模型在测试集上的平均准确率和平均损失率；
6. 详细代码见附件。
7. **实验结果**

1、实验结果：

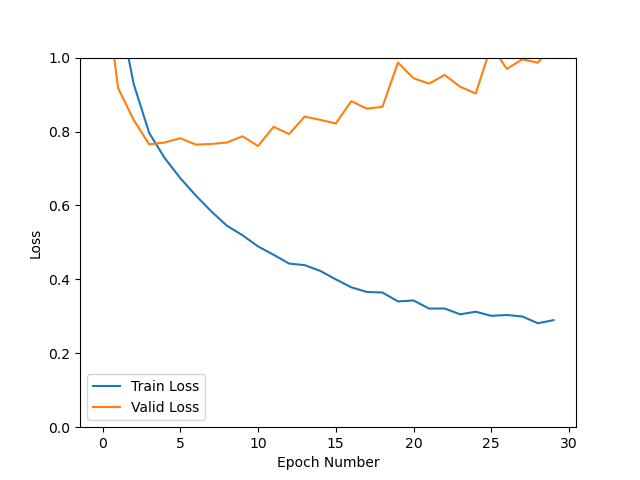
用ResNet50得到的训练集和验证集的准确率函数：

最终训练集准确率为91.2190%，验证集准确率为80.2335%



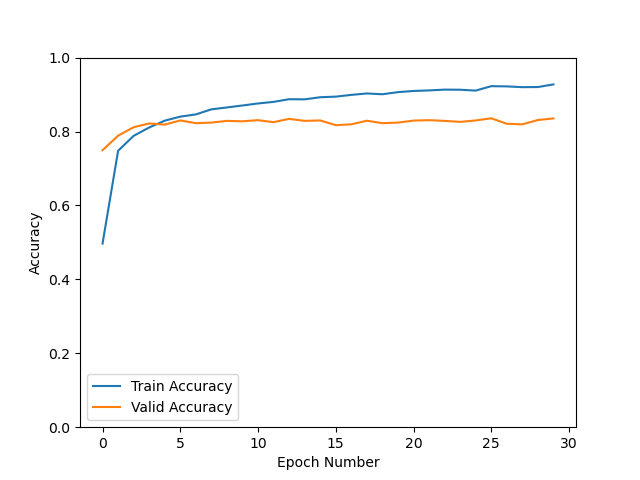
用ResNet50得到的训练集和验证集的损失率函数：

最终训练集损失率为0.2812，验证集损失率为0.9861



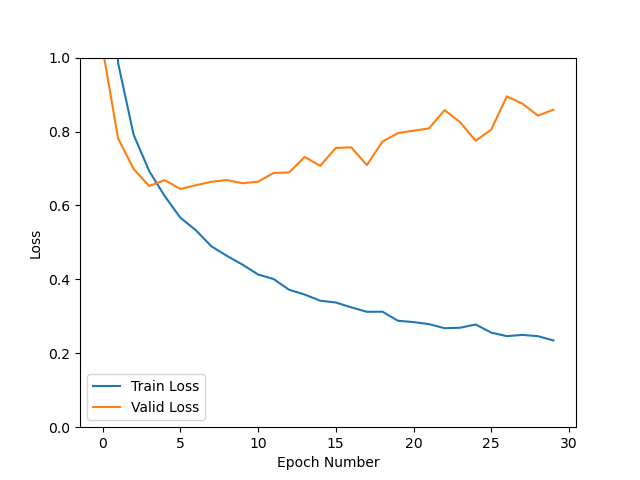
用ResNet152得到的训练集和验证集的准确率函数：

最终训练集准确率为92.7440%，验证集准确率为83.5409%



用ResNet152得到的训练集和验证集的损失率函数：

最终训练集损失率为0.2347，验证集损失率为0.8591



ResNet50训练后模型用于测试集的准确率为0.7941，损失率为1.085

ResNet152训练后模型用于测试集的准确率为0.8158，损失率为1.029

1. 实验结果分析：
2. 由上文的图可知，ResNet50的训练和验证集准确率相差较大，ResNet152的训练和验证集准确率相差较小；另外ResNet152整体训练过程中的准确率要比ResNet50高，损失率比ResNet低；

原因分析：ResNet152网络复杂度更高，对数据的拟合能力更强，平均来看训练后模型的性能更好一点；另外，由于本次实验的数据量过大，ResNet152并没有在该训练数据集上表现出过拟合现象（即训练误差和测试误差增大），反而使训练拟合效果更好；

1. ResNet152模型训练后应用于测试集的效果更好，准确率更高，损失率更小；

原因分析：由于ResNet152训练后模型的准确率更高，损失率更低，因此该模型运用到测试集上的预测效果更好；

1. 在训练过程中，两种网络模型的验证集的损失率均是先下降再上升的趋势，而ResNet152的变化范围更大；

原因分析：验证集数据量太少了，而模型复杂度较高，导致验证集上模型的过拟合，又因为ResNet152比ResNet50的模型复杂度更高，过拟合程度则更大；

1. 实验详细结果见附件：
2. 用ResNet50得到的训练集和验证集的准确率和损失率随训练次数的变化结果：train\_val\_50.txt；
3. 用ResNet152得到的训练集和验证集的准确率和损失率随训练次数的变化结果：train\_val\_152.txt；
4. **遇到问题及解决思路**

问题1：如何得到ResNet预训练模型？

解决方法：由于直接调用torchvision.model.resnet是从pytorch官方网站上下载，速度过慢，因此需要提前离线下载好ResNet模型，直接调用torch.load函数加载模型进行训练。

问题2：如何判断目前使用的网络模型（模型1）对于数据集是过拟合还是欠拟合？

解决方法：选用不同深度的网络再次进行实验，假设选用了一个深度更深的网络模型（模型2），若模型2的训练集准确率比模型1高，且模型2的训练集与测试集（验证集）的误差相差比模型1小，则认为模型1是欠拟合的；若模型的训练集准确率比模型1高，但模型2的训练集与测试集（验证集）的误差相差比模型1大，则认为模型2是过拟合的，可以再选用复杂度比1大，比2小的模型再次验证模型1是否是对当前数据集的最佳模型。

1. **参考文献**

[1] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.