实验二：猫狗分类

一、实验目的

1. 进一步理解和掌握卷积神经网络中卷积层、卷积步长、卷积核、池化层、池化核、微调(Fine-tune)等概念。
2. 进一步掌握使用深度学习框架进行图像分类任务的具体流程：如读取数据、构造网络、训练和测试模型等等。
3. 了解 PyTorch（或其他框架）在 GPU 上的使用方法

二、实验环境

本次编程实验主要基于miniconda和pytorch框架进行，使用的各个库的版本号如下：

- python 3.7.16
- pytorch 1.10.0
- torchvision 0.11.0
- numpy 1.21.6
- tensorboard 2.11.2
- cuda 11.8

三、数据预处理

数据集重构

本实验使用实验数据基于 kaggle Dogs vs. Cats 竞赛提供的官方数据集，原数据集划分为训练集（training dataset）和验证集（validation dataset），均包含 dogs 和 cats 两个目录，且每个目录下包含与目录名类别相同的 RGB 图。数据集共 25000 张照片，其中训练集猫狗照片各10000 张，验证集猫狗照片各2500张，由于数据集照片数量过多，考虑到电脑单机单卡性能，现对数据集进行重构。

- 训练数据集：从原始训练数据集中选择8000张打乱的猫狗图片中选择70%作为训练数据集，共计5600张
- 验证数据集：从原始训练数据集中的8000张打乱的猫狗图片中选择30%作为训练数据集，共计2400张
- 测试数据集：从原始训练数据集中选择2000张作为测试集

预处理

- 训练数据集和验证数据集进行数据增强

```
self.transform=transforms.Compose([
                        transforms.Resize(size=(256, 256)),
                        transforms.RandomResizedCrop(size=(224, 224)),
                        transforms.RandomHorizontalFlip(),
                        transforms.ToTensor(),
                        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=
[0.229, 0.224, 0.225])
                    ])
```

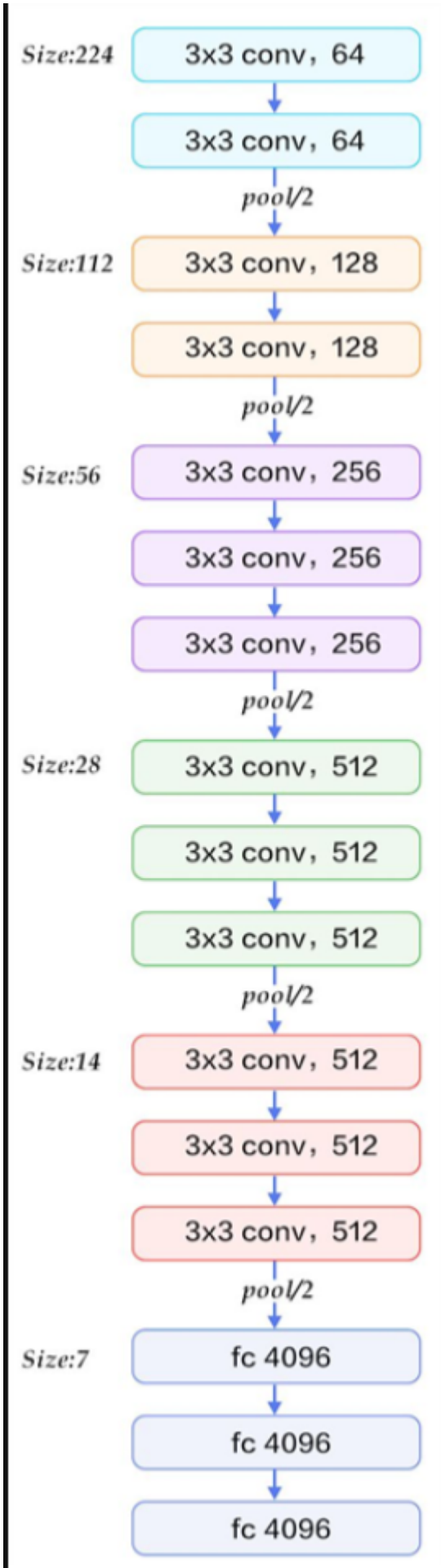
- 测试集不进行数据增强

```
self.transform=transforms.Compose([
                            transforms.Resize(size=(224,224)),
                            transforms.CenterCrop(size=(224,224)),
                            transforms.ToTensor(),
                            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=
[0.229, 0.224, 0.225])
                        ])
```

四、网络架构

本次实验构建了多个网络模型来进行训练，首先自己复现了VGG16这种经典的CNN网络，然后也利用pytorch中自带的经过预训练后的VGG16网络来进行实验，最终的的实验结果中测试集的准确率均达到90%以上的水平，

满足指导书中准确率大于75%的要求。 **VGG16模型简介：** VGG16由小卷积核、小池化核、ReLU组合而成。其



简化图如下（以VGG16为例）

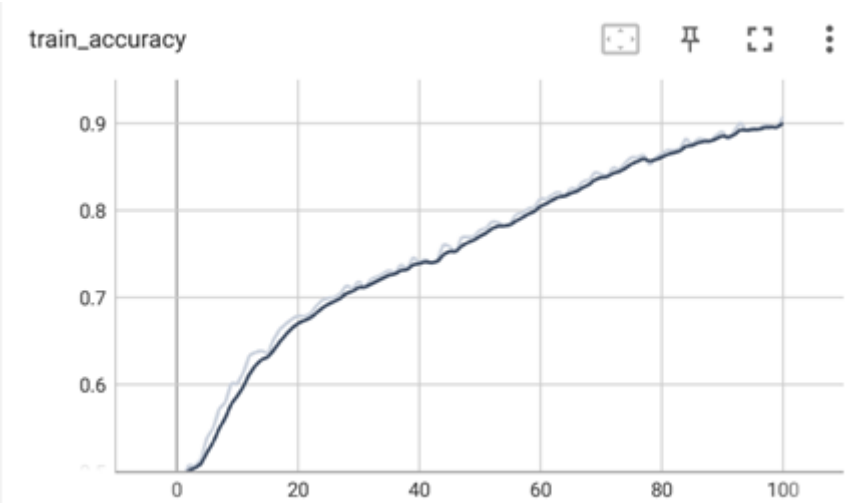
五、实验过程及结果

VGG16复现模型 VGG16是典型的块结构，结构规整，具有很强的拓展性。原来VGG16分类是1000类，而现在实验进行猫狗分类任务也即二分类任务，所以修改VGG16的三个全连接层，代码如下：

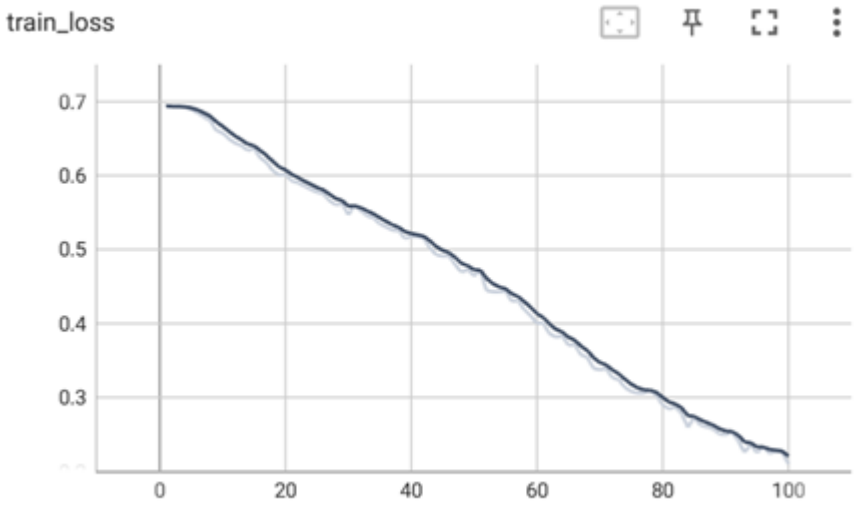
```
self.FC1 = nn.Sequential(
            nn.Flatten(),
            nn.Linear(512 * 7 * 7, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(0.2)
        )
        self.FC2 = nn.Sequential(
            nn.Linear(4096, 512),
            nn.ReLU(inplace=True),
            nn.Dropout(0.2)
        )
        self.FC3 = nn.Sequential(
            nn.Linear(512, 2)
        )
```

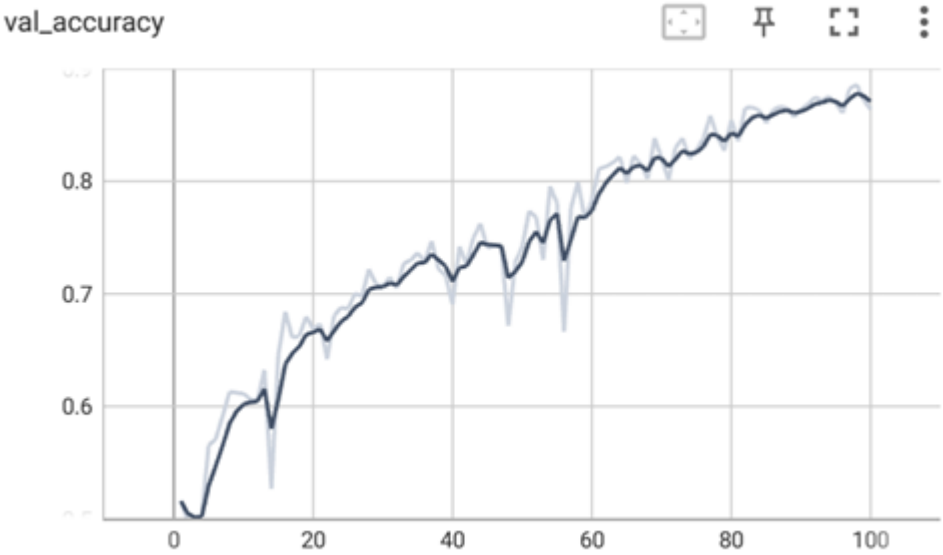
在本次代码中，采用的参数设置为

- 优化器：采用SGD随机梯度下降优化器,设置动量momentum为0.01
- 损失函数：采用 CrossEntropyLoss 交叉熵损失函数
- 超参设置： batch_size=10，num_epochs=100，lr=0.01
- 补充说明：lr利用schedur中的ReduceLROnPlateau进行控制，在5轮损失不下降的平台区缩小为0.01；受限于单机单卡的限制，batch_size的值不能设置太大


train_accuracy
0.9
0.8
0.7
0.6
0
20
40
60
80
100

实验结果如下图所示


train_loss
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

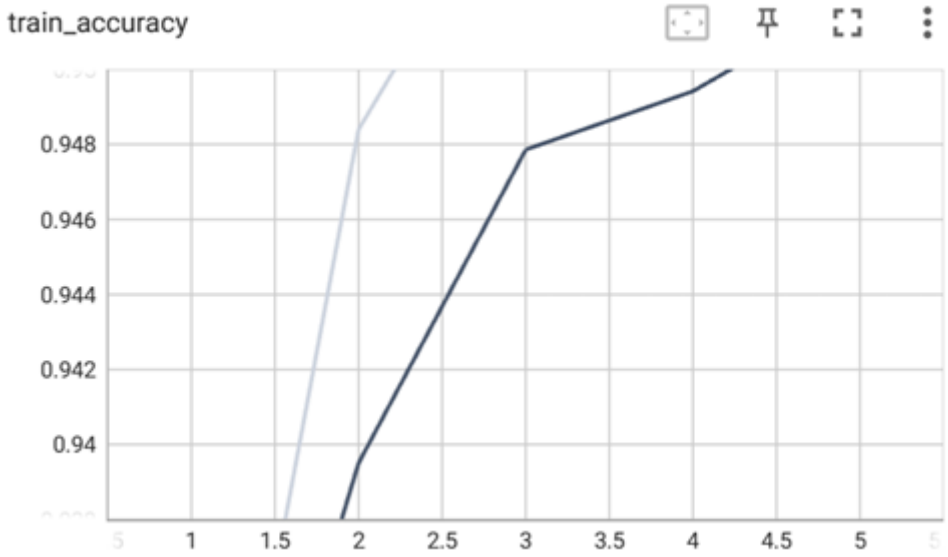



```
epoch: 98, 训练次数: 250, Train Loss: 0.2399, Train Accuracy: 0.8916
epoch: 98, 训练次数: 300, Train Loss: 0.2311, Train Accuracy: 0.8950
epoch: 98, 训练次数: 350, Train Loss: 0.2333, Train Accuracy: 0.8934
epoch: 98, 训练次数: 400, Train Loss: 0.2331, Train Accuracy: 0.8940
epoch: 98, 训练次数: 450, Train Loss: 0.2382, Train Accuracy: 0.8911
epoch: 98, 训练次数: 500, Train Loss: 0.2365, Train Accuracy: 0.8928
epoch: 98, 训练次数: 550, Train Loss: 0.2397, Train Accuracy: 0.8917
在epoch=98下，整体训练集上的Loss: 0.2397
在epoch=98下，整体训练集上的准确率: 0.8920
整体验证集上的准确率: 0.8679
--------第 100 轮训练开始--------
epoch: 99, 训练次数: 0, Train Loss: 0.3692, Train Accuracy: 0.9000
epoch: 99, 训练次数: 50, Train Loss: 0.2294, Train Accuracy: 0.8922
epoch: 99, 训练次数: 100, Train Loss: 0.2354, Train Accuracy: 0.8901
epoch: 99, 训练次数: 150, Train Loss: 0.2427, Train Accuracy: 0.8868
epoch: 99, 训练次数: 200, Train Loss: 0.2425, Train Accuracy: 0.8841
epoch: 99, 训练次数: 250, Train Loss: 0.2420, Train Accuracy: 0.8837
epoch: 99, 训练次数: 300, Train Loss: 0.2387, Train Accuracy: 0.8850
epoch: 99, 训练次数: 350, Train Loss: 0.2372, Train Accuracy: 0.8866
epoch: 99, 训练次数: 400, Train Loss: 0.2374, Train Accuracy: 0.8878
epoch: 99, 训练次数: 450, Train Loss: 0.2381, Train Accuracy: 0.8863
epoch: 99, 训练次数: 500, Train Loss: 0.2333, Train Accuracy: 0.8896
epoch: 99, 训练次数: 550, Train Loss: 0.2295, Train Accuracy: 0.8917
在epoch=99下，整体训练集上的Loss: 0.2295
在epoch=99下，整体训练集上的准确率: 0.8923
整体验证集上的准确率: 0.8842
保存的模型路径为: models_save/VGG16_0416_07_07.pth
模型在epoch为99时保存了一次，此时验证集准确率为0.2295，测试集准确率为0.8842，时间为0416_07_07.pth
length of test_data is2000
整体测试集上的准确率: 0.9180
```

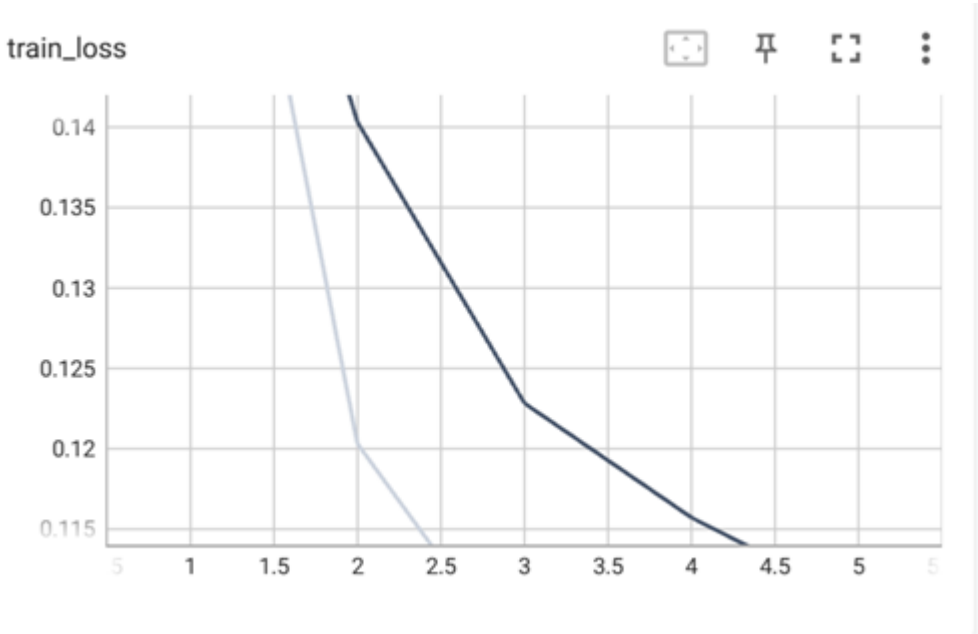
VGG16预训练模型 使用Pytorch下models模块中中所带的 VGG16 的预训练模型，仅需将分类层也即全连接层中的神经元格式更改一下，并在此基础上进行微调即可，具体代码如下：

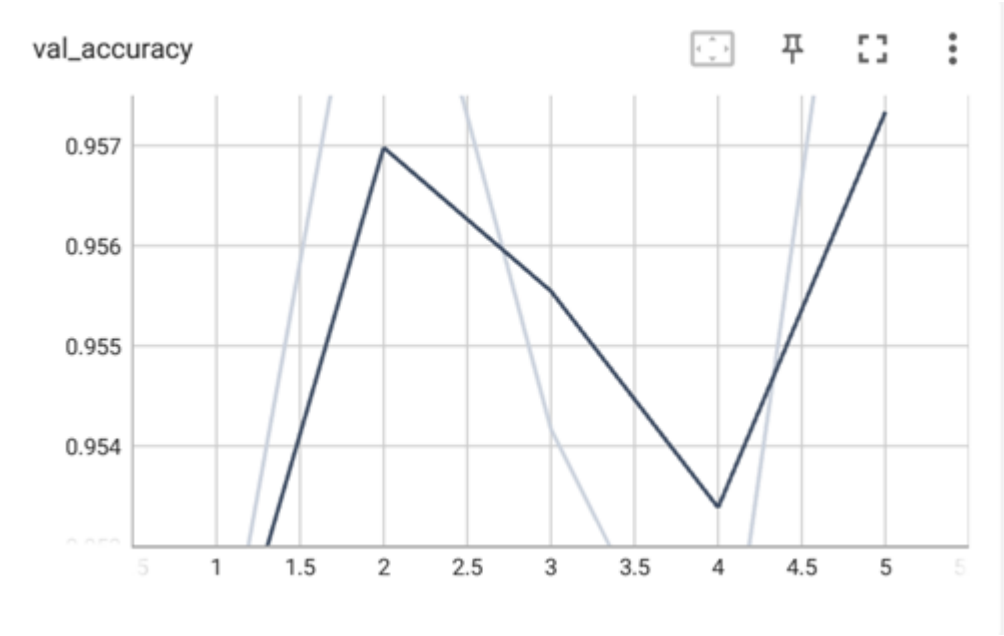
```
model.classifier = nn.Sequential(
        nn.Linear(512 * 7 * 7, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(),
        nn.Linear(4096, 512),
        nn.ReLU(inplace=True),
        nn.Dropout(),
        nn.Linear(512, 2),
    )
```

除epoch设置为5以外，其余超参数设置保持同上一复现模型不变。利用该模型进行训练，可以看到效果非常好，在迭代结束之后，最终得到模型的训练准确率95%，验证集准确率达96%，测试集上的准确率可达98%，


train_accuracy
0.948
0.946
0.944
0.942
0.94
1
1.5
2
2.5
3
3.5
4
4.5
5

得到结果如下图所示：


train_loss
0.14
0.135
0.13
0.125
0.12
0.115
1
1.5
2
2.5
3
3.5
4
4.5
5




```
epoch: 2, 训练次数: 400, Train Loss: 0.1005, Train Accuracy: 0.9576
epoch: 2, 训练次数: 450, Train Loss: 0.1030, Train Accuracy: 0.9563
epoch: 2, 训练次数: 500, Train Loss: 0.1028, Train Accuracy: 0.9571
epoch: 2, 训练次数: 550, Train Loss: 0.1060, Train Accuracy: 0.9559
整体训练集上的Loss: 0.1060
整体训练集上的准确率: 0.9559
整体验证集上的准确率: 0.9542
模型已保存，该模型是在第2个epoch取得95%以上的验证准确率，准确率为：0.9542
-------第 4 轮训练开始-------
epoch: 3, 训练次数: 0, Train Loss: 0.3019, Train Accuracy: 0.8000
epoch: 4, 训练次数: 200, Train Loss: 0.0989, Train Accuracy: 0.9577
epoch: 4, 训练次数: 250, Train Loss: 0.1019, Train Accuracy: 0.9562
epoch: 4, 训练次数: 300, Train Loss: 0.1025, Train Accuracy: 0.9568
epoch: 4, 训练次数: 350, Train Loss: 0.1018, Train Accuracy: 0.9584
epoch: 4, 训练次数: 400, Train Loss: 0.1006, Train Accuracy: 0.9584
epoch: 4, 训练次数: 450, Train Loss: 0.1020, Train Accuracy: 0.9574
epoch: 4, 训练次数: 500, Train Loss: 0.1051, Train Accuracy: 0.9553
epoch: 4, 训练次数: 550, Train Loss: 0.1036, Train Accuracy: 0.9557
整体训练集上的Loss: 0.1036
整体训练集上的准确率: 0.9554
整体验证集上的准确率: 0.9625
模型已保存，该模型是在第4个epoch取得95%以上的验证准确率，准确率为：0.9625
length of test_data is2000
val_accuracy is 1968,total number is 2000
整体测试集上的准确率: 0.9840
```

小结 通过这次实验，自己学会了利用 Pytorch 深度学习框架实现猫狗分类，认识了VGG网络模型，增加了对于卷积神经网络的理解。对于深度学习的整个流程，包括数据预 处理、网络模型构建、模型训练、scheduler、pytorch下model模块的使用等方面都有了更深层次的理解，代码的组织编写水平也有了很大的提高，对于后续实验起到了很好 的启发作用，受益匪浅!