实验 2-卷积神经网络实现

符兴 7203610316

1 实验环境

Ubuntu 20.04，Python 3.9， PyTorch1.13，Cuda11.7，TensorBoard；

```
{ home } python
Python 3.9.16 (main, Mar  8 2023, 14:00:05)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.__version__
'1.13.1'
>>> torch.version.cuda
'11.7'
>>>
```

图 1 Python 环境

2 实验过程

2.1 读取数据

```
# 读取数据
print("{}\n{}\n{}".format("*"*30, "正在读取数据", "*"*30))
for i in os.listdir("./101_ObjectCategories"):
    labelToNum.update({i:[]})
    tmp = os.listdir("./101_ObjectCategories/"+i)
    for j in range(0, len(tmp)):                                                    划分数据集
        labelToNum[i].append(["./101_ObjectCategories/"+i+"/"+tmp[j], len(labelToNum)])
    labelToNum[i] = np.random.permutation(np.array(labelToNum[i]))

for id, i in enumerate(os.listdir("./101_ObjectCategories")):
    imgPathTrain += labelToNum[i][0:int(round(0.8*len(labelToNum[i]))),0].tolist()
    imgLabelTrain += labelToNum[i][0:int(round(0.8*len(labelToNum[i]))),1].tolist()
    if len(labelToNum[i])<60:
        for _ in range(2):
            imgPathTrain += labelToNum[i][0:int(round(0.8*len(labelToNum[i]))),0].tolist()
            imgLabelTrain += labelToNum[i][0:int(round(0.8*len(labelToNum[i]))),1].tolist()  少类别数据增强
    imgPathValid += labelToNum[i][int(round(0.8*len(labelToNum[i]))):int(round(0.9*len(labelToNum[i]))),0].tolist()
    imgLabelValid += labelToNum[i][int(round(0.8*len(labelToNum[i]))):int(round(0.9*len(labelToNum[i]))),1].tolist()
    imgPathTest += labelToNum[i][int(round(0.9*len(labelToNum[i]))):,0].tolist()
    imgLabelTest += labelToNum[i][int(round(0.9*len(labelToNum[i]))):,1].tolist()

# 写入
f = open("./imgPathTrain.pkl","wb")
pickle.dump(imgPathTrain,f)                    划分好的数据集信息写入文件中
f.close()

f = open("./imgLabelTrain.pkl","wb")
pickle.dump(imgLabelTrain,f)
f.close()

f = open("./imgPathValid.pkl","wb")
pickle.dump(imgPathValid,f)
f.close()
```

图 2 划分数据集

本次实验所使用的数据集是 Caltech101，其中包含 101 种类别的物体，每种

类别大约 40 到 800 个图像；由于每个类别的数量不相等，且有较大的差异；在本次实验中对于那些类别较少的样本进行数据增强操作；在程序划分数据集时，对数量较少类别的图片进行多次图像预处理，如使用 RandomAffine()函数对图像进行仿射变换，对局部信息进行截取；以及使用 RandomHorizontalFlip()和 RandomVerticalFlip()对图片进行镜像操作；然后将多张预处理的图片放入训练集中；

```
def __getitem__(self, item):
    img = Image.open(self.imgPath[item]).convert("RGB")
    label = atoi(self.imgClass[item])
    if self.id == 1:                                                    这里使用的两种预处理：仿射变换和翻转
        img = transforms.RandomAffine(degrees=0,translate=(0.1,0.1),shear=(-15,15))(img)
        img = transforms.RandomHorizontalFlip(p=0.8)(img)
    if self.transform is not None:
        img = self.transform(img)
    tmp = [0 if i != label else 1 for i in range(0,self.classCnt)]
    labelTensor = torch.tensor(tmp, dtype=torch.float)
    return img, labelTensor, label

def __len__(self):
    return len(self.imgPath)
```

图 3 在线数据增强

除此之外，在划分数据集时还会对每个类别进行随机采样；在划分好数据集后，会将划分的数据集信息保存下来，之后在每次训练的时候直接读取该信息即可，保证每次训练的时候数据集都是一致的；

2.2 模型结构

```
class AlexNet(nn.Module):
    def __init__(self, labelCnt, dropout = 0.3):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 96, kernel_size=11, stride=4, padding=0), # 96*54*54
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=0), # 96*26*26

            nn.Conv2d(96, 256, kernel_size=5, stride=1, padding=2), # 256*26*26
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=0), # 256*12*12

            nn.Conv2d(256, 384, kernel_size=3, stride=1, padding=1), # 384*12*12
            nn.ReLU(),

            nn.Conv2d(384, 384, kernel_size=3, stride=1, padding=1), # 384*12*12
            nn.ReLU(),

            nn.Conv2d(384, 256, kernel_size=3, stride=1, padding=1), # 256*12*12
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=0), # 256*5*5
        )
        self.classifier = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(256 * 5 * 5, 2048),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(2048, 1024),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(1024, labelCnt),
        )
```

图 4 模型结构

本次实验是仿照 AlexNet 网络进行设计，大致分为两个部分，一个是基于卷积神经网络进行特征提取，另一个是根据特征进行分类的网络；

在特征提取层中，大致可以分为五个部分：

(1) 卷积层+池化层：卷积层的输出维度 96，kernal_size：11，步长为 4，padding 为 0；池化层的 kernal_size 为 3，步长为 2，padding 为 0；最后的输出为 96*26*26；

(2) 卷积层+池化层：卷积层的输出维度 256，kernal_size：5，步长为 1，padding 为 2；池化层的 kernal_size 为 3，步长为 2，padding 为 0；最后的输出为 256*12*12；

(3) 卷积层：卷积层的输出维度 384，kernal_size：3，步长为 1，padding 为 1；最后的输出为 384*12*12；

(4) 卷积层：卷积层的输出维度 384，kernal_size：3，步长为 1，padding 为 1；最后的输出为 384*12*12；

(5) 卷积层+池化层：卷积层的输出维度 256，kernal_size：3，步长为 1，padding 为 1；池化层的 kernal_size 为 3，步长为 2，padding 为 0；最后的输出为 256*5*5；

在分类层中，大致可以分为几个部分：

(1) Dropout+Linear+ReLU：输出为 4096*1

(2) Dropout+Linear+ReLU：输出为 2048*1

(3) Dropout+Linear：输出为 101*1

同时，在本次实验中使用交叉熵作为损失函数，由于 PyTorch 所提供的计算交叉熵的 CrossEntropyLoss()函数中已经集成了 softmax，则在 forward 函数中不需要显式对结果做归一化处理；优化器选择 Adam 优化器，并且使用 MultiStepLR 对学习率根据 epoch 进行动态调整；

3 实验结果

(1) 没有数据增强的训练，在测试集上的测试结果为 0.65

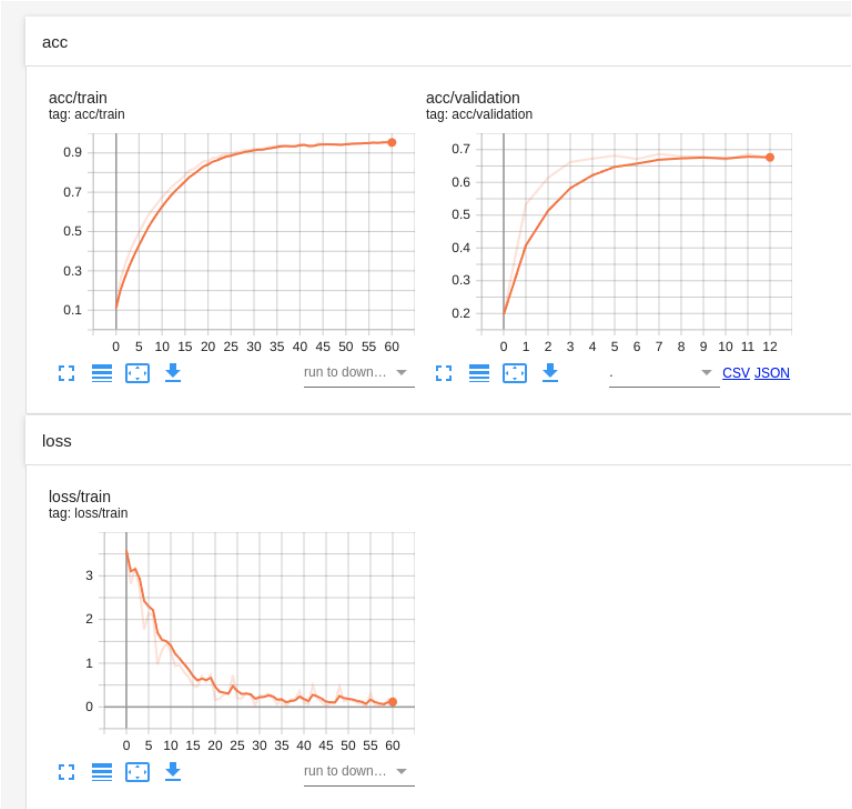

图 5 没有数据增强的训练结果

(2) 使用仿射变换数据增强的训练，在测试集上的测试结果为 0.702

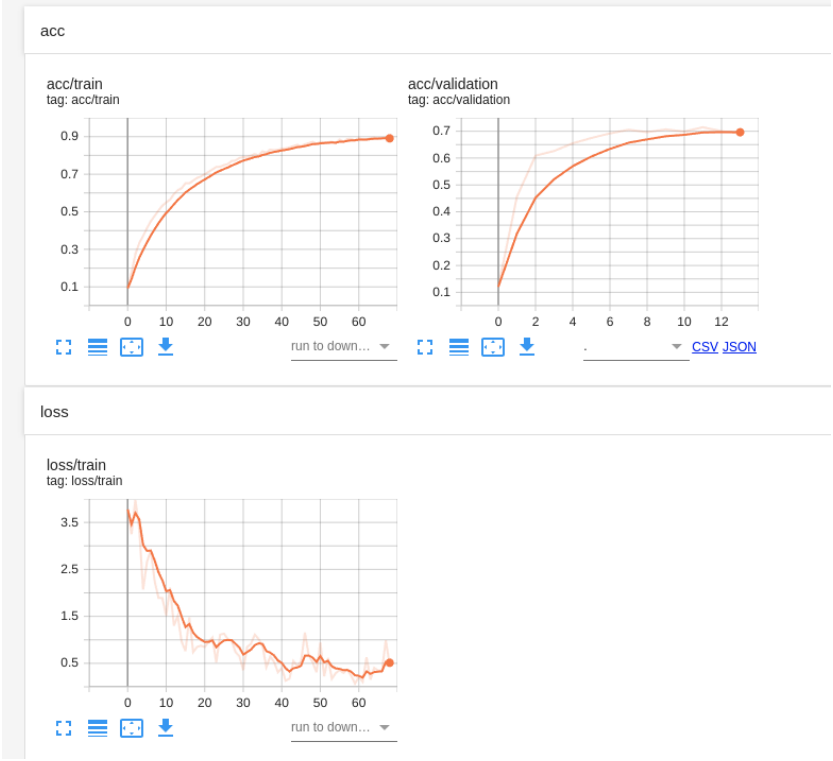

图 6 在线数据增强的训练结果

(3) 使用多种数据增强+权重衰减的训练，在测试集上的测试结果为 0.728

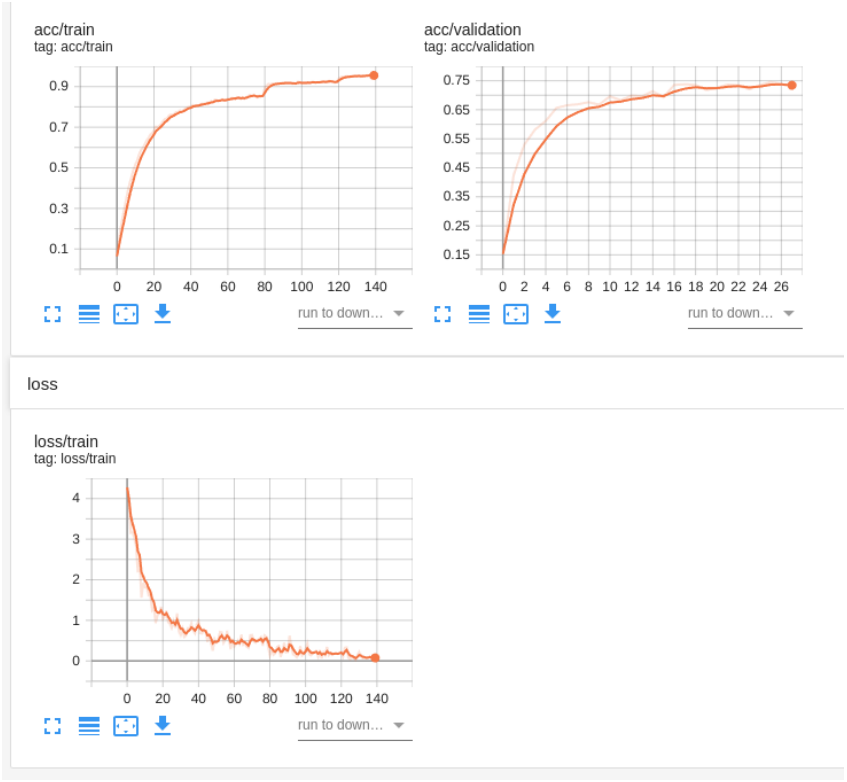

图 7 多种数据增强+权重衰减的训练结果

(4) 只有权重衰减的训练，在测试集上的测试结果为 0.71

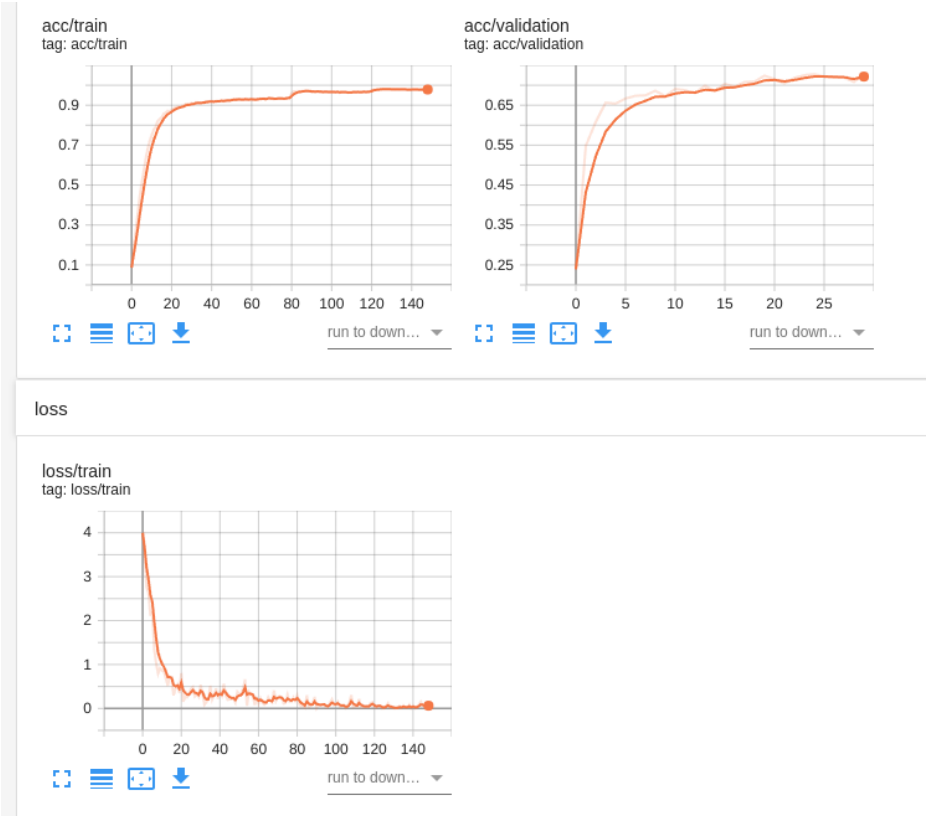

图 8 只有权重衰减的训练结果

通过上述四组实验数据的对比，可以发现，纯粹使用原有数据集进行训练的模型，在训练的后期出现了过拟合的情况，即训练集正确率仍在上升，但是验证集的正确率已趋于不变；当使用一定程度的数据增强时，模型的泛化能力有所增强，如图 6 所示，验证集的最高正确可以超过 0.7，而在图 5 所示的数据中，验证集的最高正确率不超过 0.7；

除此之外，本次实验还设置了权重衰减的实验组，如图 7 所示，在多种数据增强和权重衰减的训练中，验证集的最高正确率可以超过 0.75，在测试集上的正确率也能达到 0.728，相比图 6 的数据而言，模型的精度提高了 7%左右；同时，单独设置了一个只有权重衰减的训练过程，其验证集最高精度不超过 0.75，在测试集上的测试结果为 0.71，低于图 6 的模型精度。

4 心得与体会

在本次实验中，发现在训练的过程中可以使用多种手段去提高模型的精度，如数据增强、权重衰减等；前者是通过增加数据的样本量达到提高模型泛化能力的目的；而后者是通过在反向传播的过程中，依据模型权重去限制损失函数的计算，进而避免模型过拟合；通过多次实验发现，数据增强的手段非常有效，通过对图像的翻转、仿射变换、截取以及增加噪声等等方式，模型可以学习到更多物体的特征，进而有更高的精度；