哈爾濱工業大學

模式识别与深度学习实验报告

实验 二

题　　目　卷积神经网络实现

学　　院　计算学部

专　　业

学　　号

学　　生

任课教师　左旺孟

哈尔滨工业大学计算机科学与技术学院

2023 年春季

实验二 报告

一、 实验目的

基于 PyTorch 实现 AlexNet 结构,在 Caltech101 数据集上进行验证,并使用 tensorboard 进行训练数据可视化。如有条件，尝试不同参数的影响，尝试其他网络结构。请勿使用 torchvision.models.AlexNet。

二、 实验环境

硬件配置（华为云服务器）：

CPU: 8 核 64GB

GPU: 1*P100(16GB)

软件配置：

OS：ubuntu18.04

IDE：Jupyter Notebook

Packages in Environment：pytorch1.4-cuda10.1-cudnn7

三、 实验内容

3.1 实验基本参数

由于在实验过程中，可能需要要对 AlexNet 进行超参数的反复调节、改变迭代次数、调整图片尺寸等，因此可以通过 args＝parser.parse_args()向 ArgumentParser 对象统一添加参数。

```
parser = argparse.ArgumentParser('Caltech101_AlexNet')
parser.add_argument('--ObjectCategories_dir', type=str, default='101_ObjectCategories')
parser.add_argument('--seed', type=int, default=20) # 数据集划分随机种子
parser.add_argument('--batch_size', type=int, default=10)
parser.add_argument('--img_size', type=int, default=224) # 通过序号选择用哪一个多层感知，用来对比效果
parser.add_argument('--nn', type=str, default='0') # 通过序号选择用哪一个神经网络，用来对比效果
parser.add_argument('--gpu', type=int, default=0) # -1时为CPU
parser.add_argument('--niters', type=int, default=50) # 训练的epoch数
parser.add_argument('--model_dir', type=str, default='./model/AlexNet.pth') # 最佳模型存储路径
parser.add_argument('--tensorboard_dir', type=str, default='')
args = parser.parse_args(args=['--batch_size', '128', '--niters', '50', '--nn', '3', '--tensorboard_dir', ''])

device = torch.device('cpu' if args.gpu == -1 else 'cuda:' + str(args.gpu))
```

3.2 Caltech 101 数据读取与预处理

3.2.1 数据读取

在数据集网站下载训练集，保存到本地后，上传到华为云服务器进行训练。

忽略 BACKGROUND_Google 类别。其他图像数据按照 8:1:1 随机划分训练集、验证集和测试集，每个集中保持每个类别的比例与原始数据集中的类别比例相同。其中，随机种子是固定的，因此每次运行程序的训练集、验证集和测试集保持相同，避免数据集对模型准确率产生的干扰。

为方便和原始 AlexNet 模型比较效果，将所有图像 Resize 到 224 * 224，通过 torchvision.transforms.Resize 函数，用双线性插值方法（InterpolationMode.BILINEAR）计算，即新像素的值是通过对周围四个像素的值进行加权平均得到的。

```
class Caltech101Dataset(Dataset):
    def __init__(self, args, split='train'):
        data_dir = args.ObjectCategories_dir
        categories = os.listdir(data_dir)
        categories.remove('BACKGROUND_Google')

        self.transforms = transforms.Compose([
            transforms.Resize((args.img_size, args.img_size)),
            transforms.ToTensor(),])

        X_train, y_train, X_val, y_val, X_test, y_test = [], [], [], [], [], []

        for i, cat in enumerate(categories):
            cat_dir = os.path.join(data_dir, cat)
            X = []
            y = []
            for file in os.listdir(cat_dir):
                img_path = os.path.join(cat_dir, file)
                img = self.transforms(Image.open(img_path).convert('RGB'))
                X.append(img)
                y.append(i)

            # stratify=y 表示在分割数据时要保持标签 y 的比例，这意味着训练集和测试集中每个类别的样本比例与原始数据集中相同。
            X_train_val, X_test_cat, y_train_val, y_test_cat = train_test_split(X, y, test_size=0.1,
                                                                                stratify=y, random_state=args.seed)
            X_train_cat, X_val_cat, y_train_cat, y_val_cat = train_test_split(X_train_val, y_train_val,
                                                                              test_size=0.1111, stratify=y_train_val,
                                                                              random_state=args.seed)
```

此外，留出接口 split='train', 'val' 或 'test'来选择取训练集、验证集或测试集。

```
if split == 'train':
    self.X = X_train
    self.y = y_train
elif split == 'val':
    self.X = X_val
    self.y = y_val
elif split == 'test':
    self.X = X_test
    self.y = y_test
```

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

def __getitem__(self, idx):
    return self.X[idx], self.y[idx]
```

最后这个数据集的构造继承了 Dataset 类，并重写了 init， getitem 和 len 函数，其中 getitem 返回的是 image 和 label 的字典类型。

3.2.2 数据加载

各个集中的图像数据和原本的标签数据通过 Datalodaer 将数据分批次，因为神经网络需要按批传入数据和反向传播。其中，传入的 args.batch_size 参数可以作为超参数用来调

节训练效果。

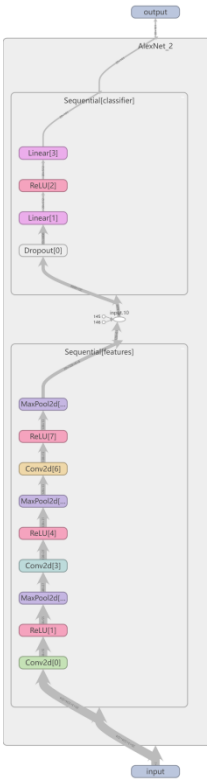
```
# 加载数据，分割成训练、验证、测试集（保持类别比例）
train_data = Caltech101Dataset(args, split='train')
torch.manual_seed(args.seed)
train_loader = DataLoader(train_data, batch_size=args.batch_size, shuffle=True) # 打乱训练集训练，防过拟合
val_data = Caltech101Dataset(args, split='val')
val_loader = DataLoader(val_data, batch_size=args.batch_size, shuffle=False)
test_data = Caltech101Dataset(args, split='test')
test_loader = DataLoader(test_data, batch_size=args.batch_size, shuffle=False)
print ('Data loaded.')
```

3.3 搭建网络

由于实验要求使用 AlexNet 和优化后的其他网络结构进行分类，且原始 AlexNet 出现了比较强的过拟合现象，因此对其用几个更简化网络进行对比。

损失函数使用多分类问题常用的交叉熵 nn.CrossEntropyLoss()。优化器选用 Adam 优化器。学习率通过 ReduceLROnPlateau 动态调整，如果验证损失在一段时间内不再下降，学习率将会减少。

在实验中，从 AlexNet_0（用一张显卡训练的原始 AlexNet）到 AlexNet_4，网络结构逐渐简化，并逐渐加入更多的 dropout 层。因篇幅有限，只以实验最佳网络结构 AlexNet_2 为例，展示参数，其他网络结构详见代码。



AlexNet_2		
层数	定义	输出尺寸
C1	48 个 11*11*3 的卷积核，步长 4，填充 2	[48, 54, 54]
Relu	inplace=True	[48, 27, 27]
M1	3*3 池化，步长 2	[48, 27, 27]
C2	128 个 11*11*3 的卷积核，步长 5，填充 2	[128, 27, 27]
Relu	inplace=True	[128, 27, 27]
M2	3*3 池化，步长 2	[128, 13, 13]
C3	128 个 11*11*3 的卷积核，步长 3，填充 1	[128, 13, 13]
Relu	inplace=True	[128, 13, 13]
M1	3*3 池化，步长 2	[128, 6, 6]
Flatten		128 * 6 * 6
Dropout	p = 0.5	128 * 6 * 6
L1	weight.size=(128 * 6 * 6, 512)	512
Relu	inplace=True	512
L2	weight.size=(512, 101)	101

3.4 训练神经网络并可视化训练结果

在训练的每一个 epoch，都进行对训练损失，验证损失、验证 Top1 准确率和训练 Top5 准确率进行验证，并保存在 tensorboard 中。

```
    # 训练过程参数存进tensorboard
    writer.add_scalar('train_loss', train_loss, epoch)
    writer.add_scalar('val_accuracy', val_accur, epoch)
    writer.add_scalar('val_top5_accuracy', val_accur_top5, epoch)

writer.close()
print ('Save tensornoard.')
```

在迭代过程中保存具有最高验证 Top1 准确率的神经网络，用于在测试集上测试实际准确率。

四、 实验结果与分析

4.1 实验基本设置

训练轮数（Epoch）：50

训练设备：GPU

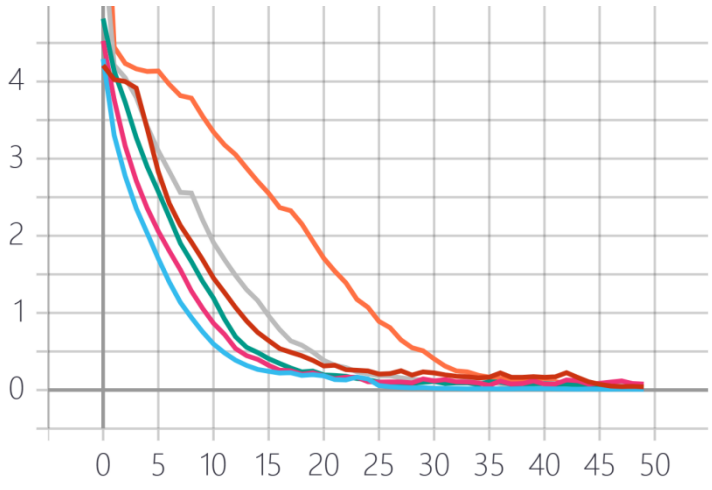
4.2 结果与分析

4.2.1 神经网络相同、batch_size 不同的结果

相同参数：AlexNet_0（即用一个显卡实现的原始 AlexNet）

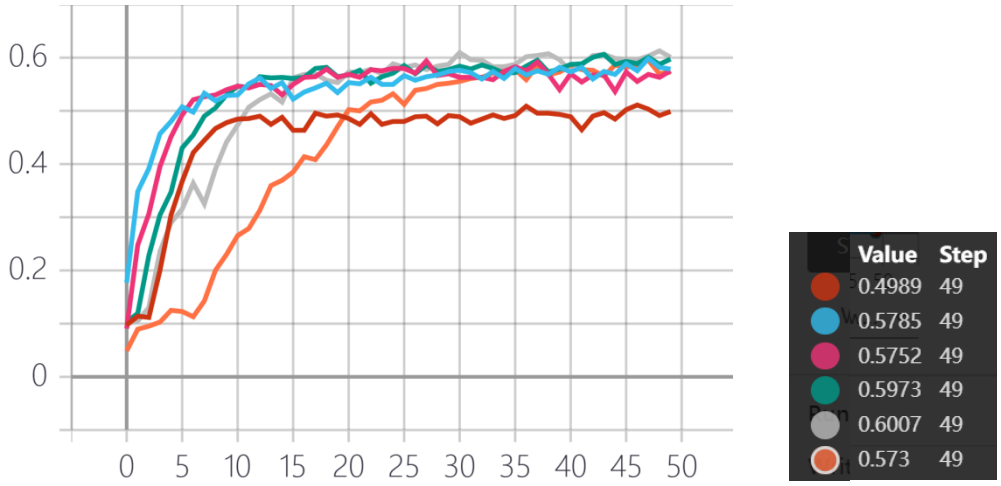
不同参数：batch_size = 32, 64, 128, 256, 512, 1024

不同 batch_size 从小到大依次的颜色表示：  ，即棕红色代表 batch_size=32，蓝色代表 batch_size=64，等等。



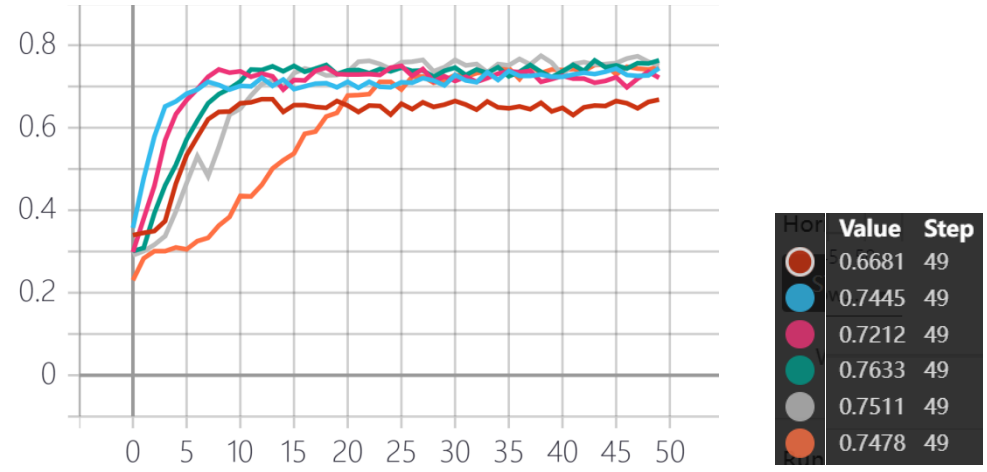
Training Loss

从中可知，batch_size=64（蓝色线）时，训练集损失下降最快，训练集损失最小。而 batch_size=32（红棕色线）过大和 batch_size=1024（桔色线）过小时损失下降都较慢。



Val_Top1_Accuracy

从中可以看出，batch_size=256（绿色线）, 512（灰色线）时，测试集 Top1 准确率最高，分别达到 0.5973 和 0.6007. 而 batch_size=32（棕红色线）的准确率明显较低。



Val_Top5_Accuracy

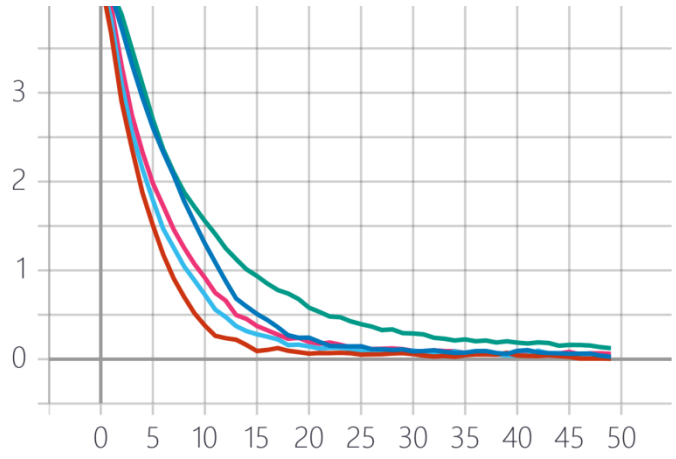
从中可以看出，batch_size=256（绿色线）, 512（灰色线）时，测试集 Top5 准确率依然最高，分别达到 0.7633 和 0.7511. 而 batch_size=32（棕红色线）的准确率同样明显低于其他参数的情况。总的来说，batch_size 对准确率的影响不是很大，只要不是太小即可。

4.2.2 batch_size 相同、神经网络不同结果

相同参数：batch_size = 256(之前实验中表现最好的 batch_size)

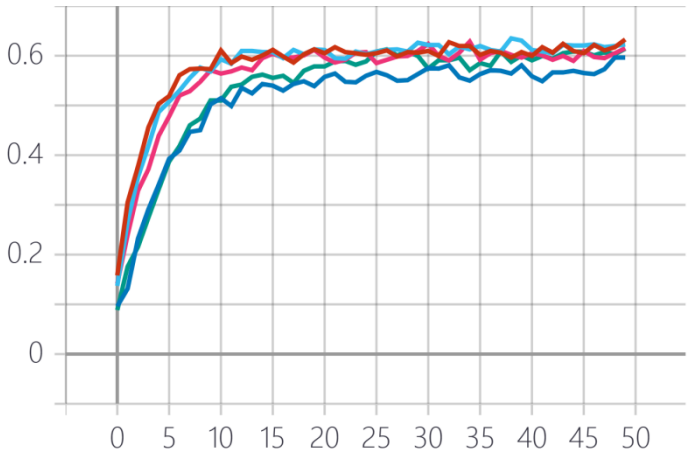
不同参数：AlexNet_0, AlexNet_1, AlexNet_2, AlexNet_3, AlexNet_4

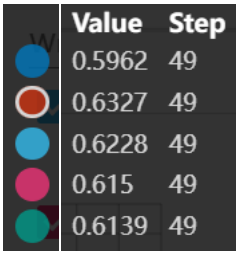
不同神经网络从左到右依次的颜色表示：  ，即深蓝色代表 AlexNet_0，红色代表 AlexNet_1，浅蓝色代表 AlexNet_2，等等。



Training Loss

其中，网络结构最简单的 AlexNet_4（绿色线）训练损失下降最慢，结合观察验证集损失也下降较慢。可能是由于模型能力不足，发生欠拟合。

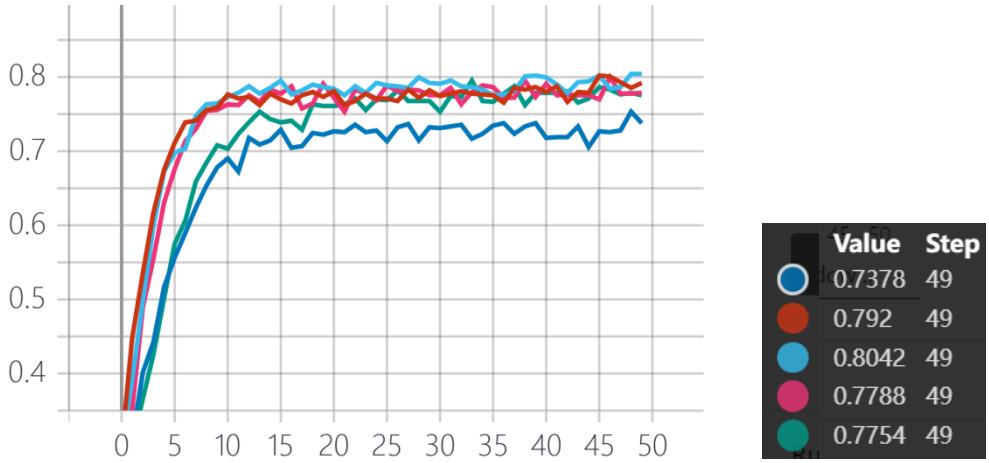




	Value	Step
	0.5962	49
	0.6327	49
	0.6228	49
	0.615	49
	0.6139	49

Val_Top1_Accuracy

随着网络结构从复杂到简单，验证集 Top 1 准确率经历了从上升到下降的变化，其中 AlexNet_1（红色线）和 AlexNet_2（浅蓝色线）准确率最高。



Val_Top5_Accuracy

随着网络结构从复杂到简单，验证集 Top 5 准确率也经历了从上升到下降的变化，其中同样是 AlexNet_1（红色线）和 AlexNet_2（浅蓝色线）准确率最高。

	Cross Entopy Loss	Top1 Accuracy	Top5 Accuracy
AlexNet_0	3.607513	57.63%	75.30%
AlexNet_1	3.253128	63.67%	81.01%
AlexNet_2	3.274276	64.11%	81.78%
AlexNet_3	3.191111	60.15%	79.58%
AlexNet_4	3.070188	60.81%	78.92%

Test Result

根据测试集准确率可以看出，AlexNet_2 是最佳网络。这可能是因为更复杂的网络容易过拟合，而更简单的网络容易欠拟合。

五、 结语

通过该实验，我体会到了对网络结构更改对于分类结构的影响，体会了从过拟合到欠拟合的调参过程，见证准确率从上升到下降的过程。同时，也体验了使用服务器进行深度学习训练的过程，收获颇丰。