华中科技大学

计算机视觉实验报告

姓　　名：王逸

学　　院：计算机科学与技术

专　　业：计算机科学与技术

班　　级：CS2011

学　　号：U202014774

指导教师：刘康

分数	
教师签名	

2023 年　4 月 4 日

目　录

实验二 基于卷积神经网络的 CIFAR-10 数据集分类.....1

1.1 实验要求.....1
1.2 实验内容.....1
1.3 实验结果分析.....4
1.4 实验小结.....7

实验二 基于卷积神经网络的 CIFAR-10 数据集分类

1.1 实验要求

设计一个卷积神经网络，在 CIFAR-10 数据集上实现分类任务。

注意事项：

1. 不能直接导入现有的 CNN 网络，比如 VGG、ResNet 等，可以以现有网络为基础进行改进。
2. 可以尝试不同的卷积神经网络设计、使用不同的激活函数等，观察网络性能。
3. 深度学习框架任选。
4. 实验报告包含网络设计、在 10 类测试集上的平均准确率截图、每一类的准确率截图以及必要的分析等。

1.2 实验内容

A. 准备数据集与数据增广 get_cifar10_data()

本实验直接通过 torchvision.datasets 导入 CIFAR-10 的数据集，随后对原始数据进行数据预处理，使训练得到的模型具有更强的泛化能力。

以下代码分别对应操作：

训练数据：随机裁剪、随机翻转、转变为 Pytorch 张量、标准化处理。

测试数据：转变为 Pytorch 张量、标准化处理。

其中标准化处理输入的参数为三个通道的标准差。

```
# data augmentation
transform_train = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

transform_test = transforms.Compose([
transforms.ToTensor(),
transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])
```

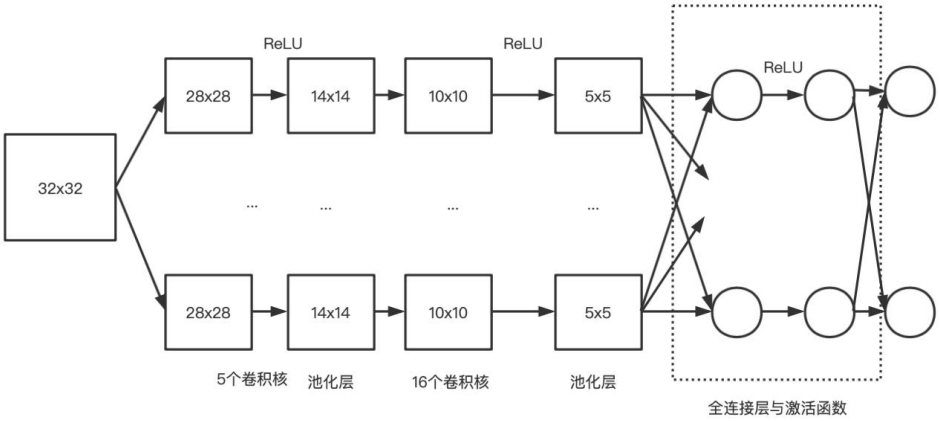
之后以调用 DataLoader 打乱封装得到 train_loader 和 test_loader。

```
train_dataset = CIFAR10(root='./dataset', train=True, download=True, transform=transform_train)
test_dataset = CIFAR10(root='./dataset', train=False, download=True, transform=transform_test)
train_loader = DataLoader(train_dataset, shuffle=True, batch_size=batch_size)
test_loader = DataLoader(test_dataset, shuffle=True, batch_size=batch_size)
```

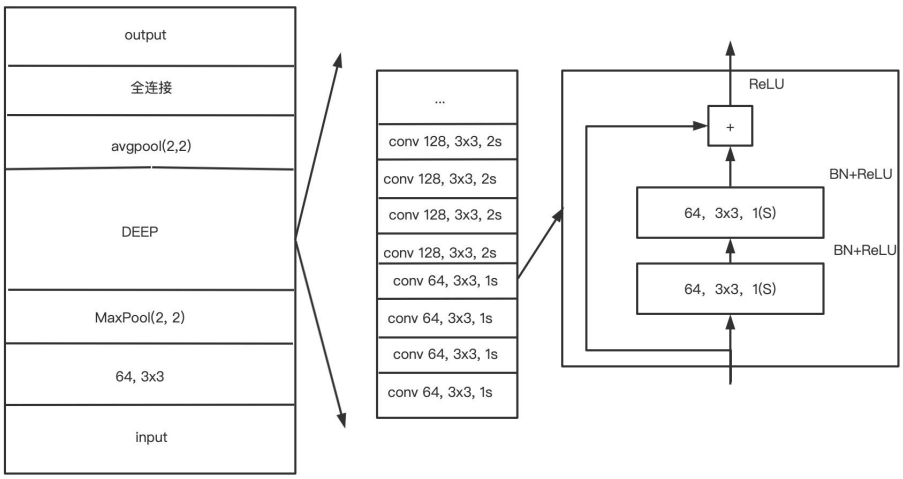
B. 卷积神经网络模型搭建

本任务需要搭建卷积神经网络结构，输入 channel 为 3，最终分类数为 10。除此之外还需要定义中间卷积层、池化层以及全连接层。在后面进行性能比较时，我采用了三种网络：LeNet、ResNet 和 VGG. 另外在卷积层与激活函数之间增加归一化操作。三种网络结构图分别如下：

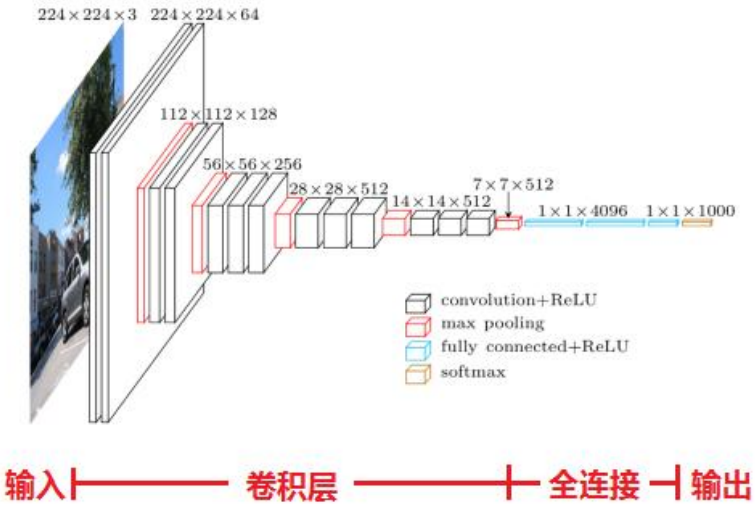
- LeNet：两层卷积层、两层最大池化层以及三层全连接层



- ResNet：一层卷积层、八个残差块，一个平均池化以及一层全连接层。与传统的 ResNet 不同，将第一层卷积核大小改为 3x3 以匹配 32*32*3 的图片大小。



- VGG：卷积层统一使用 3x3 的小卷积核与 2x2 的池化核。



代码部分如下见代码文件夹/lab2/Net

C. 训练

定义初始化模型 model、损失函数（该实验中采用 Cross_EntropyLoss 损失函数）与 SGD 优化器。

Adam 优化器虽然一开始收敛速度较快，但到后期可能会出现模型难以收敛的状况。但由于 Adam 可以自适应调整学习率，因此我们需要为增加一个 lr_scheduler 动态调整学习率。在此使用 StepLR 等间隔调整学习率，每次运行时执行的 epoch 为 40，每隔 5 次将学习率调整为原来的 0.4。

```
# model
model = get_network(args.network).to(DEVICE)
if save_model:
    model.load_state_dict(torch.load('./model/resnet_40_0.01_model.pth'))
    print('load pretrained model')

cirterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=args.lr, momentum=0.9, weight_decay=5e-4)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.1)
```

为了方便控制变量，在命令行调用时加入参数 network—本次训练使用哪种神经网络、epoch—迭代次数、lr—学习率。默认值为('lenet', '40', '0.01')

调用事例为：python main.py --network resnet --lr 0.0001 --epoch 15

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Network')
    parser.add_argument("--network", type=str, default='lenet',
                        help="type of cnn. such as lenet, resnet, vgg...")
    parser.add_argument("--epoch", type=int,
                        help="number of epoc
    parser.add_argument("--lr", type=float, default=0.01,
                        help="learning rate. such as 1, 2, 4...")
    args = parser.parse_args()
```

(parameter) default: Any

D. 画图分析

本次实验需要画出两种图形：

一是模型测试准确率随着 epoch 的变化曲线；二则是模型在十个分类上的各自准确率。这里采用曲线图和柱状图。

该部分与实验一中类似，代码可见文件 draw_fig.py。

1.3 实验结果分析

a. 不同的神经网络对网络性能的影响

该部分通过选择三个不同的神经网络，对于给定的学习率进行训练，得到相应的准确率以及十分类各自的平均准确率，并在同一个曲线图中画出。

编号	神经网络种类	学习率	迭代次数
1	LeNet	0.01	40
2	VGG13	0.01	40
3	ResNet18	0.01	40

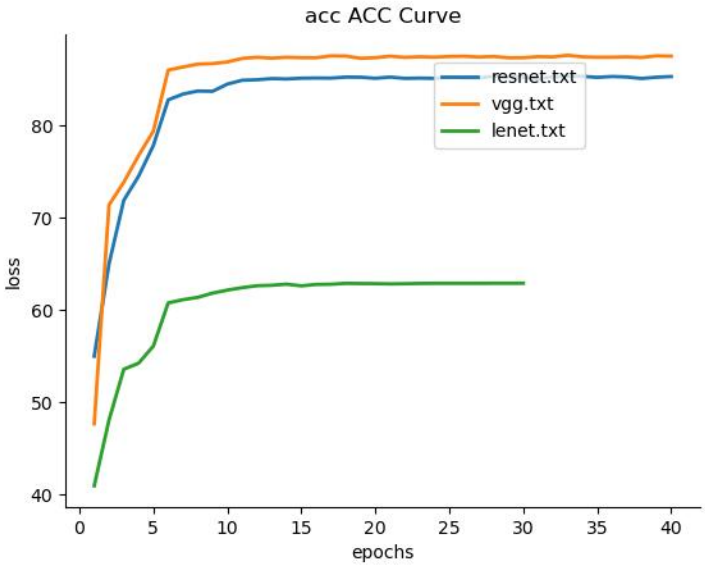


图 1 lr=0.01, epoch=40

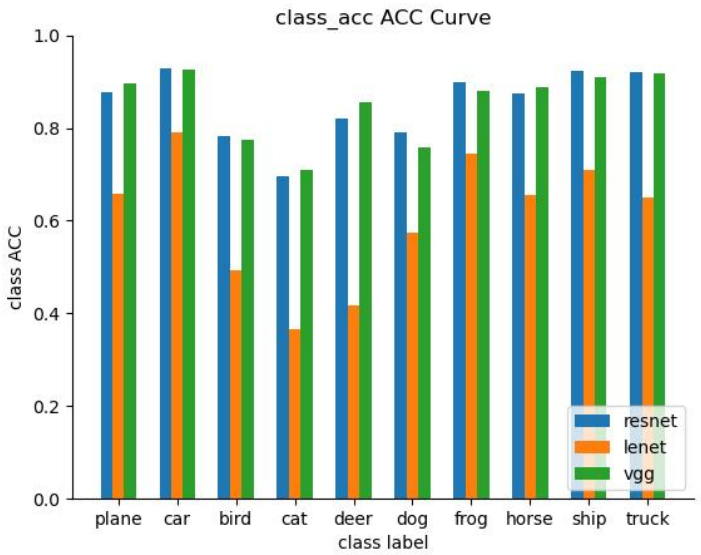


图 2 十分类准确率

通过以上三组实验，可以看出：

- 最简单的卷积神经网络（LeNet）只能达到 64%左右的正确率，而复杂一点的例如 ResNet 或者 vgg 均可达到 87 或 88%左右
- 所有的网络都大概在 epoch 7-10 左右达到稳定，之后正确率缓慢增加直到平稳。并且最开始的 LeNet 是在本机的 cpu 上运行，而到了后面的 resnet 和 vgg 在 cpu 上运行需要耗费大量的时间, 故转而到服务器的 gpu 上运行。
- 而关于十分类上预测的准确率，我们可以看出有关动物的例如猫狗、鹿的准确率较低，猜测是由于面部比较复杂与相似，简单的神经网络并不能很好的将其区分开来。

b. 不同的学习率对网络性能的影响

改变学习率。

编号	神经网络种类	学习率	迭代次数
1	VGG13	0.1/0.01	40
2	ResNet18	0.01/0.01	40

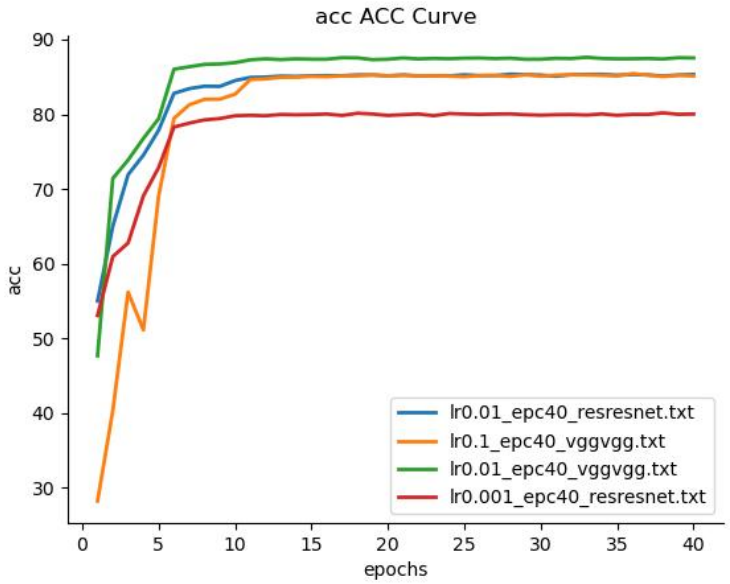
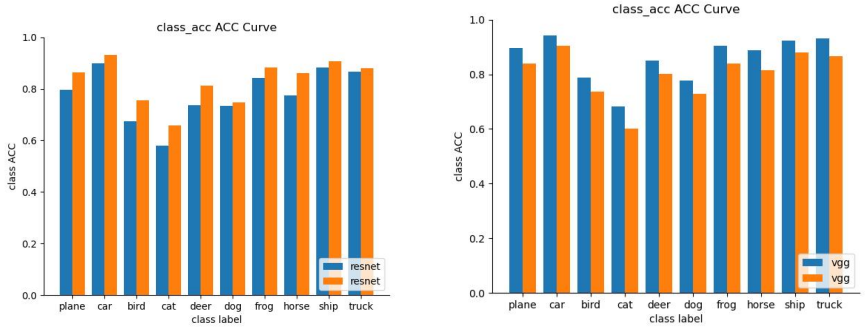


图 3 Tanh，layer=1



通过以上图像可以看出：

- 0.01 是在没有预训练模型的强况下最优的数值，过高或过低都会降低预测准确性。

c. 重载模型手动降低学习率

每一次重新训练模型到后面会发现预测准确率在 20epoch 左右开始不会发生太大变化。为了进一步训练得到更高的准确率，我们在每次训练结束后保存模型参数，在下一次训练开始前载入参数，并且手动更改更小的学习率继续训练。

在本次试验中，主要对 ResNet 进行重载训练，然而在训练至 lr=0.0005，epoch=150 左右发现之后的再次训练准确率也基本维持在 89.6%左右不再发生改变，遂停止。

曲线图中的跳跃点为每次重载模型改变学习率造成。

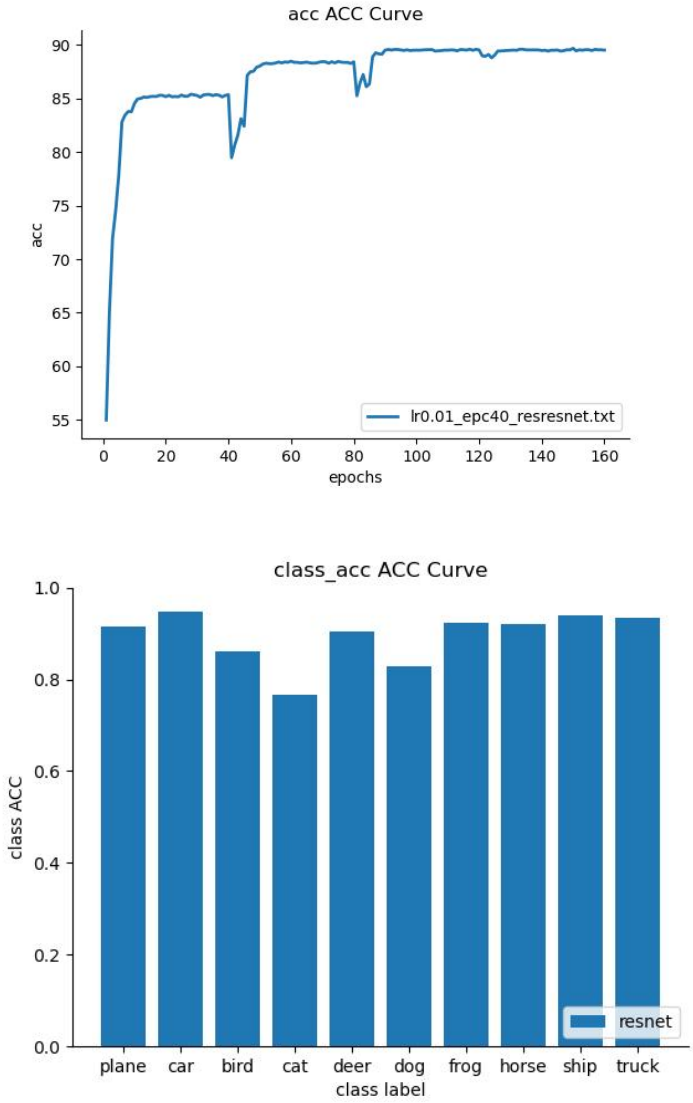


图 4 resnet, lr=0.0005, epoch=160

1.4 实验小结

本次实验过程中除了以上三种考察因素之外, 还手动调节了以下 VGG 网络的各种参数, 例如将通道数从 64 改为 96，dropout 改为 0.4，准确率均略微提高，但最终也只达到了 89% 左右的正确率。从上面的分析中我们可以看见本实验中所使用的网络基于硬件设备的原因并不复杂，若在之后能增加网络层数或许能得到更高的结果。