哈爾濱工業大學

人工智能软件开发与实践

实验报告

题　　目	卷积神经网络 CNN
学　　院	计算机科学与技术
专　　业	人工智能
学　　号	2021112845
学　　生	张智雄
任 课 教 师	武小荷

哈尔滨工业大学计算机科学与技术学院

2023.9

实验三：卷积神经网络 CNN

1、 实验内容

搭建 Python 和 Pytorch 环境，并在 MNIST 或 CIFAR-10 数据集上使用卷积神经网络 CNN 实现手写体数字识别或图像分类，主要包括三个部分：

a) 下载、预处理数据

b) 使用 Pytorch 实现 CNN 分类器，网络结构自行选择（AlexNet，VGG，ResNet 等），batch size 和 epoch 按需设置

c) 测试集分类精确度不低于 90%

2、 算法简介及其实现细节

2.1 卷积神经网络(CNN)的基本原理

卷积神经网络(Convolutional Neural Network，CNN)是一种深度学习模型，特别设计用于处理和分析具有网格结构的数据，如图像和视频。它能够自动学习图像中的特征并进行高效的图像分类、对象检测、图像生成和分割等任务，其模型结构主要包含以下部分：

a) 卷积层: 卷积层负责从图像中提取特征，如边缘和纹理。它们通过应用过滤器来捕捉这些特征，逐渐形成更复杂的视觉模式。

b) 池化层: 池化层在保留基本信息的同时减小了特征图的大小。最常见的方法是最大池化，它有助于缩小图像，同时保持关键特征并增强鲁棒性。

c) 全连接层: 全连接层结合从前一层提取的特征进行分类和决策。他们将这些特征映射到不同的类别，识别图像中的内容。

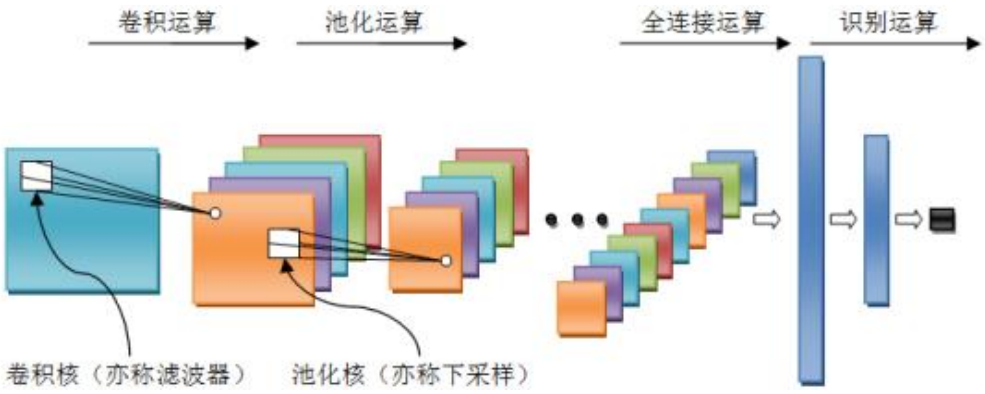

图 1 卷积神经网络模型结构

2.2 AlexNet 的基本结构

AlexNet 网络结构相对简单，使用了 8 层卷积神经网络，前 5 层是卷积层，剩下的 3 层是全连接层，具体如下图 2 所示。

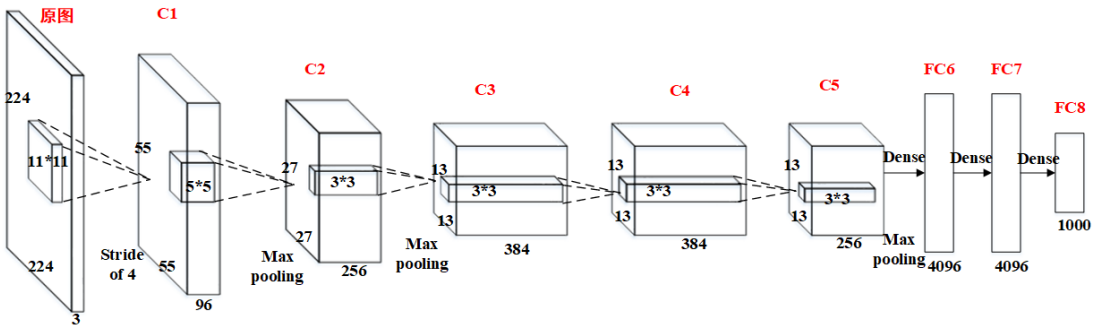

图 2 AlexNet 判别器结构

与原始的 LeNet 相比，AlexNet 网络结构更深，同时还包括以下特点：

a) ReLU 激活函数的引入：采用修正线性单元(ReLU)的深度卷积神经网络能够大幅提高训练速度，同时能够有效防止过拟合现象的出现。

b) 层叠池化操作：AlexNet 中池化层采用了层叠池化操作，即池化大小>步长，这种类卷积操作可以使相邻像素间产生信息交互和保留必要的联系。

c) Dropout 操作：Dropout 操作会将概率小于 0.5 的每个隐层神经元的输出设为 0，即去掉一些神经节点，能够有效防止过拟合现象的出现。

2.3 ResNet 的基本结构

ResNet 的主要创新是引入了残差连接（或称为跳跃连接），以解决深度神经网络训练中的梯度消失和梯度爆炸问题。其核心思想是通过跳跃连接将输入信号直接传递到网络中的后续层，使得网络可以学习到残差信息，从而更轻松地训练非常深的神经网络。这些跳跃连接可以是简单的恒等映射，也可以包含适当的线性变换以匹配维度。这种结构使得神经网络的训练更加稳定，从而能够构建非常深的网络。

下图 3 为 34 层 ResNet 模型的架构图，仿照 AlexNet 的 8 层网络结构可以将 ResNet 划分成 8 个构建层。

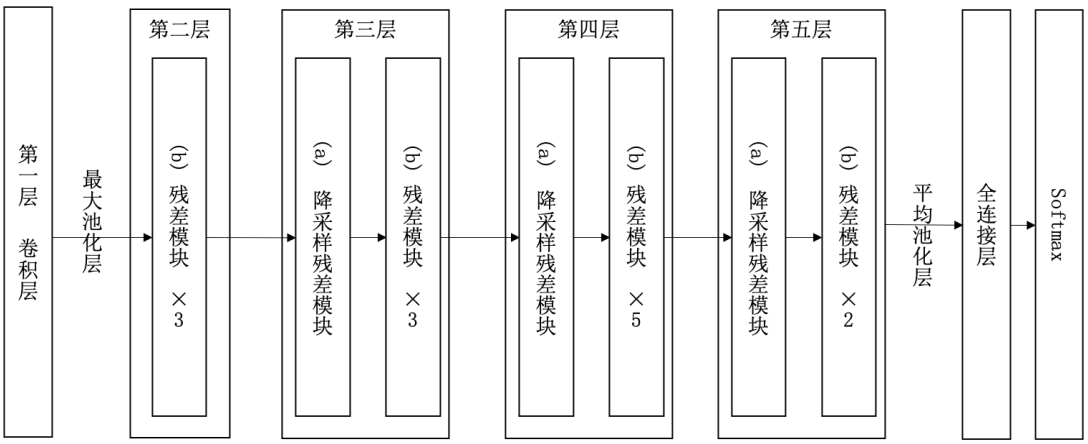

图 3 ResNet 判别器结构

3、 实验设置及结果分析（包括实验数据集）

3.1 实验数据集及数据预处理

MNIST 数据集(Mixed National Institute of Standards and Technology database)是美国国家标准与技术研究院收集整理的大型手写数字数据集，包含 60,000 个样本的训练集以及 10,000 个样本的测试集。其中包括 0 到 9 的数字。

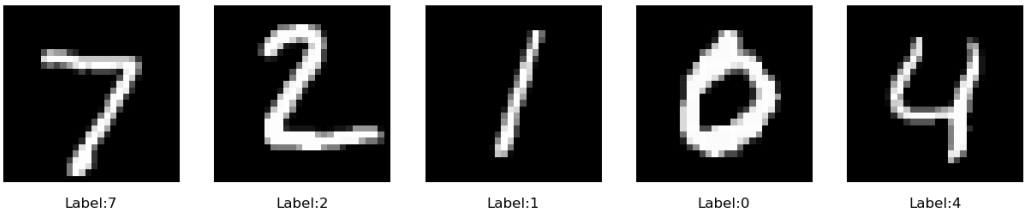

图 4 MNIST 数据集

CIFAR-10 数据集(Canadian Institute for Advanced Research-10)是一个由加拿大高级研究所创建和维护的经典计算机视觉数据集，包含 60,000 张图像。其中分为飞机、汽车、猫、狗等 10 个不同类别。

图 5 CIFAR-10 数据集

在本实验中，使用 torchvision 自带的数据集加载 MNIST 和 CIFAR-10 数据集，并使用 transforms.ToTensor 方法加载为 Tensor 张量，最后通过 DataLoader 加载进 GPU 进行运算。

```python
def data_processing(str):
    if str == 'mnist':
        transform = transforms.ToTensor()  # 转换为张量
        trainset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=transform)
        trainloader = torch.utils.data.DataLoader(trainset, batch_size=128, shuffle=True)

        testset = torchvision.datasets.MNIST(root='./data', train=False, download=True, transform=transform)
        testloader = torch.utils.data.DataLoader(testset, batch_size=128, shuffle=False)
        return trainloader, testloader
    elif str == 'cifar':
        transform = transforms.ToTensor()  # 转换为张量
        trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
        trainloader = torch.utils.data.DataLoader(trainset, batch_size=128, shuffle=True)

        testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
        testloader = torch.utils.data.DataLoader(testset, batch_size=128, shuffle=False)
        return trainloader, testloader
    return None, None
```

图 6 数据预处理代码

3.2 模型设计

在本次实验中，仿照 AlexNet，实现了包含五个卷积层和三个全连接层构建

一个深度卷积神经网络，网络的定义是重写 nn.Module 实现的，卷积层和全连接层之间将数据通过 view 拉平，同时加入 Dropout 层防止数据过拟合。

```python
class AlexNet(nn.Module):
    def __init__(self, width_mult=1):
        super(AlexNet, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),  # 32*28*28
            nn.MaxPool2d(kernel_size=2, stride=2),  # 32*14*14
            nn.ReLU(inplace=True),
        )
        self.layer2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3, padding=1),  # 64*14*14
            nn.MaxPool2d(kernel_size=2, stride=2),  # 64*7*7
            nn.ReLU(inplace=True),
        )
        self.layer3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)# 128*7*7
        self.layer4 = nn.Sequential(
            nn.Conv2d(128, 256, kernel_size=3, padding=1),  # 256*7*7
        )
        self.layer5 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),  # 256*7*7
            nn.MaxPool2d(kernel_size=3, stride=2),  # 256*3*3
            nn.ReLU(inplace=True),
        )
        self.dropout = nn.Dropout(0.5)
        self.fc1 = nn.Linear(256 * 3 * 3, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 10)
```

图 7 AlexNet 模型结构代码

3.3 实验结果

在本次实验中，使用交叉熵损失函数和 SGD 优化器，将模型输入通道根据数据集设为1或者3，并设置训练超参数$epoch$为10，$batch\ size$为128，学习率$learning\ rate$为0.01。训练过程中损失函数$loss$的值和在测试集上的准确率变化如下图 8 所示。

实验发现，随训练过程的进行，损失函数不断降低，在测试集上准确率逐渐升高，最终测试正确率最高能够达到约98.94%。损失函数和测试准确率在训练最后阶段呈现波动态，可能原因是在局部最优点附近振荡。

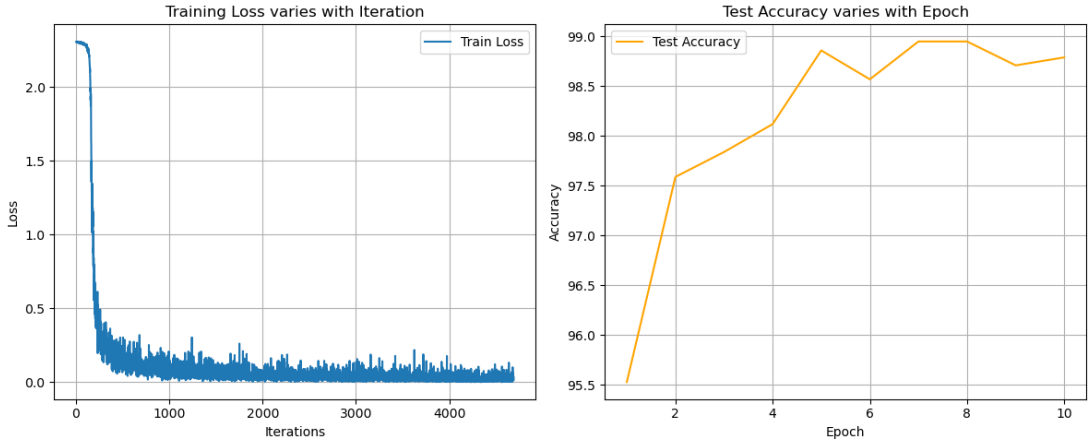

图 8 MNIST 实验结果（左为损失变化，右为测试集上准确率）

而后通过 torch.load 方法加载模型对测试集进行直观展示，模型能够对手写数字作出较为准确的分类，具有一定的泛化能力。

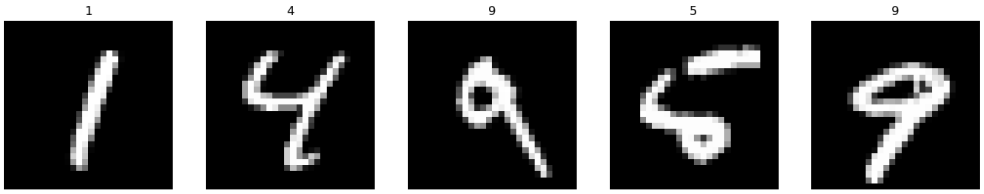

图 9 测试集上分类效果

将此模型应用在 CIFAR-10 数据集上进行训练，分类效果大幅降低，经过 30 个$epoch$后准确率仅达到75.11%。

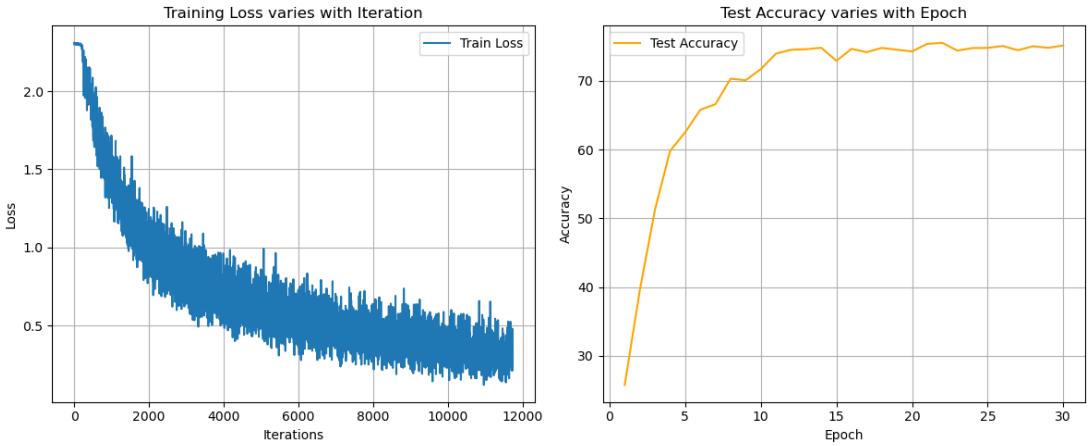

图 10 CIFAR-10 实验结果（左为损失变化，右为测试集上准确率）

另外使用 torch.model 中封装好的 ResNet18 和 ResNeXt50 模型在 CIFAR 数据集上进行对比训练，经过足够的$epoch$训练后，发现测试准确率略有提升，但最高只能达到 85%左右，分析可能有以下原因：

a) 模型的表达能力较差，难以在 3 通道较为复杂的彩色图像中实现较为准确的识别分类。

b) 数据集图像分辨率较低，且种类间具有较为相似的特征，包含信息较少。

c) 数据集缺乏预处理和增强，模型可能很难捕捉到不同类别间的关键特征。

d) 模型训练过程中可能存在过拟合现象，需要对它们进行适当调整或加入正则化约束其参数的更新。

4、 实验结论

卷积神经网络使用卷积操作，相较于全连接，其网络层与层之间的连接是稀疏的。其次同一层的卷积的参数是共享的，且每一层的卷积运算可以并行处理，具有较快的学习和推理速度，同时也具有较强的表示和学习能力，在图像分类领域具有较为广泛的应用。此外，CNN 还可用作其他任务的基础模型，如生成对抗网络（GAN），作为其 backbone 模型来辅助生成高质量的图像。

5、 实验收获

在本次实验中，通过实际使用调试进一步了解了现有深度学习框架 Pytorch，在自行搭建 CNN 模型框架的过程中，熟悉了 torch.nn 中 Conv2d、MaxPool2d、Dropout、Linear 等常用网络层的接口使用，锻炼提高了我们的代码能力。

同时，也较为深入的学习了如 AlexNet、ResNet、ResNeXt 等卷积网络模型，通过实际在训练集上的测试作了对比

在代码实际编写过程中，会因为不注意维度的变化和处理、输入图像的通道数导致程序出错，同时卷积核的大小、步长以及填充也会影响训练的结果。此外，数据存放的位置，是否加载进 GPU 进行运算，设置合理的学习率也是我们编写代码中需要考虑的，在逐步调试代码的过程中也是对我们能力的锻炼。

6、 参考文献

[1] 李航. 统计学习方法[M]. 清华大学出版社, 2012.

[2] 周志华. 机器学习[M]. 清华大学出版社, 2016.

[3] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolute onal neural networks[J]. Advances in neural information processing systems, 2012, 25.

[4] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.