

计算机视觉与应用

实践

练习2——LeNet-5在MNIST

数据集上的训练和测试

|  |  |  |  |
| --- | --- | --- | --- |
| **姓 名:** | 林雅惠 | **学 号:** | 123106222874 |
| **学 院:** | 计算机科学与工程学院 | | |

**2024年4月30日**

目录

[一、 实验原理及步骤 3](#_Toc8203)

[二、 自定义函数实现 4](#_Toc4144)

[2.1实验方法 4](#_Toc7814)

[2.1.1 LeNet模型类 4](#_Toc15719)

[2.1.2 模型训练 6](#_Toc4231)

[2.1.3 模型测试 7](#_Toc8081)

[2.1实验结果与分析 7](#_Toc21029)

[三、 Pytorch实现 7](#_Toc28798)

[3.1实验方法 7](#_Toc13221)

[3.2实验结果及分析 8](#_Toc3662)

# 实验原理及步骤

LeNet-5是由Yann LeCun等人于1998年提出的深度卷积神经网络，被广泛应用于手写数字识别任务。其原理如下：

1. 卷积层

LeNet-5包含两个卷积层，分别用于提取图像的特征。每个卷积层由多个卷积核组成，每个卷积核与输入图像进行卷积操作以生成特征映射。每个卷积核心学习一组特征，通过对输入图像进行滑动，检测出图像中的边缘、纹理等特征。

1. 池化层

在每个卷积层之后，LeNet-5使用最大池化操作来减少特征映射的大小，并提取最显著的特征。最大池化操作通过在每个池化区域中选择最大值来实现。池化操作有助于提高模型的局部不变性和鲁棒性，同时减少了模型的参数数量。

1. 全连接层

在经过卷积和池化操作后，LeNet-5使用两个全连接层来进行分类。全连接层将所有的特征连接到输出层，用于对输入图像进行分类。最后一个全连接层的输出通过softmax函数进行处理，以生成每个类别的概率分布。

1. 激活函数

LeNet-5使用非线性激活函数来引入非线性特性，这有助于模型学习复杂的特征和决策边界。

1. 损失函数和优化器

在训练过程中，LeNet-5使用交叉熵损失函数来衡量模型输出与真实标签之间的差异。优化器用于调整模型参数，以最小化损失函数。

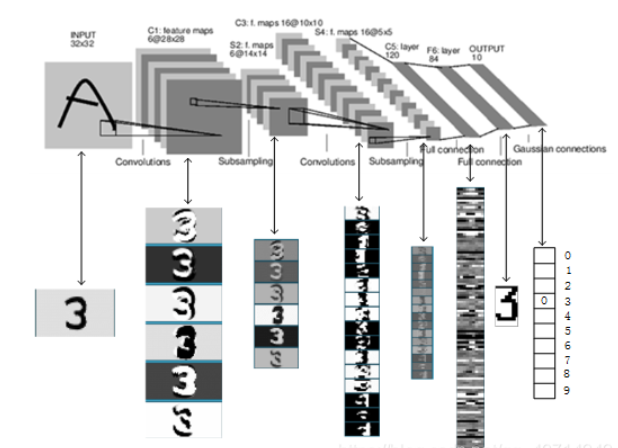


图1 原理图

# 自定义函数实现

## 2.1实验方法

本实验中对经典的LeNet-5模型进行了改进，经典的LeNet-5模型包含卷积层、池化层、卷积层、池化层、全连接层、全连接层、输出层，本实验中改进的LeNet模型结构包含卷积层、池化层、ReLU激活层、卷积层、池化层、ReLU激活层、全连接层、ReLU激活层、全连接层、ReLU激活层、输出层。相对于经典的模型结构，添加了额外的ReLU激活层，提高了模型的非线性特性，梯度传播稳定性和泛化能力，同时还能在一定程度上减少过拟合的风险，从而提高模型的性能和稳定性。

本实验中在保持基本的卷积层和池化层的结构时，还使用了ReLU激活函数代替了传统的Sigmoid激活函数，该改进具有以下优势：

1. 激活函数优化：解决了Sigmoid激活函数中的梯度消失问题。
2. 训练速度提升：由于ReLU的计算简单，使用ReLU可以加速神经网络的训练过程，相较于Sigmoid函数需要进行较为复杂的指数运算。
3. 减少过拟合风险：提供了非线性能力的同时还帮助减轻了过拟合的问题。
4. 维持网络稀疏性：ReLU在负数部分的输出为0，这一特性使得网络中的激活节点数量减少，提高了计算效率。
5. 改善网络泛化能力：由于ReLU的特性和结合了多个ReLU层，这种结构提供了很好的泛化能力。

### 2.1.1 LeNet模型类

在模型类中定义一个可训练对象的基类、卷积层类、池化层类、激活函数类、全连接层类、损失函数类、以及模型主类。

可训练对象的基类Trainable中首先定义训练过程中的属性，包括学习率、学习率衰减、动量、最大训练步数、每个训练批次的数据量、权重衰减等，接着\_init\_方法中对属性进行初始化，在sgd\_momentum方法中利用动量更新权重和偏置的变化。这个类在神经网络训练过程中，用于设置训练参数，并通过sgd\_momentum方法实现带动量的SGD（随机梯度下降），有助于提高训练的稳定性和效率。

卷积层类继承自Trainable类，在\_init\_方法中对层的名称、卷积核的大小、输入通道数、输出通道数、权重、偏置进行初始化。在forward方法中执行卷积层的前向传播操作，计算输出特征图的尺寸、对输入特征图进行卷积计算，通过循环在每个位置上应用卷积核，并使用矩阵乘法计算输出，最后返回前向传播的结果。在backward方法中执行卷积层的反向传播操作，计算梯度并更新参数，计算权重和偏置的梯度，计算反向传播的梯度，用于传递到前一层，使用动量更新权重和偏置的梯度，使用学习率和权重衰减更新权重和偏置，最后返回反向传播的梯度。

池化层类定义两个方法，分别是forward方法和backward方法，在forward方法中执行池化层的前向传播操作，接收输入，其中包括样本的数量、输入特征图的高度、输入特征图的宽度、输入特征图的通道数，计算输入特征图的形状，并将其重新进行调整，以便进行池化操作，使用np.max函数在池化窗口内找到最大值，并将结果存储在out中，保存池化时的掩码，以便在反向传播时使用，返回池化后的结果。在backward方法中执行池化层的反向传播操作，接收上游传递下来的梯度，其形状与前向传播中的out相同，将梯度进行调整，以便与前向传播时使用的掩码相乘，通过乘以掩码，将梯度传播回原始的输入尺寸，将结果重新调整后返回。

激活函数类定义了两个方法，分别是forward方法和backward方法。在forward中执行ReLU激活函数的前向传播操作，接收输入x，将输入x和0比较，生成一个布尔值的矩阵，其中大于0的元素为true，小于等于0的元素为false，将布尔值矩阵与输入x相乘，返回经过激活函数ReLU激活函数处理后的结果。在backward方法中执行ReLU激活函数的反向传播，接收上游传递下来的梯度，将激活函数前向传播时保存的输入与0比较，生成一个布尔值矩阵，其中大于0的元素为false，将布尔值矩阵与梯度相乘，返回经过ReLU激活函数反向传播出后的梯度。

全连接层类中定义了三个方法，分别是\_init\_方法、forward方法和backward方法。在\_init\_方法中先初始化全连接层对象，接收层的名称、输入的通道数和输出的通道数参数，使用高斯分布生成权重矩阵，并乘以一个缩放因子以保持梯度稳定性，初始化偏置向量，所有的元素初始值为0。在forward方法中执行全连接层的前向传播操作，接收输入x，如果输入的维度为4，则将对其进行调整，将输入x与权重矩阵相乘，然后加上偏置向量，返回全连接层前向传播后的结果。在backward方法中，执行全连接层的反向传播操作，接收上游传递下面的梯度，计算权重的梯度和偏置的梯度，计算反向传播的梯度，并将结果重新调整为原始输入的形状，使用随机梯度下降与动量更新权重和偏置，最后返回全连接层反向传播后的梯度。

损失函数类中定义了三个方法，分别是forward方法、backward方法和set\_label方法。在forward方法中，计算输入x的softmax函数，如果已经设置了标签，则计算交叉熵损失和准确率，根据是否设置了标签，返回预测类别或者交叉熵损失和准确率。在backward方法中，计算softmax损失的反向传播梯度。在set\_label方法中，设置真实标签，以便在计算损失和准确率时使用。

模型主类中实现了一个LeNet神经网络模型，并提供了训练，预测，保存和加载模型的功能。在\_init\_方法中创建一系列层，包括两个卷积层、两个池化层、两个全连接层、四个ReLU激活函数层，按照LeNet模型的结构一次排列。在train方法中，接收输入图像和对应的标签，在每个训练步骤中，从输入中取出一个大小为batch\_size的批次进行训练，通过前向传播计算模型输出，并设置softmax损失层的标签为真实标签，然后进行反向传播计算梯度并更新模型参数。在predict方法中对输入图像进行预测，通过前向传播计算模型输出。在save方法中将模型的参数保存在指定路径下的文件中。在load方法中从模型训练得到的文件中加载模型。

### 2.1.2 模型训练

模型训练时，先调用get\_training\_data函数加载训练数据集，该函数中先遍历所有的图像，读取灰度图像，并将其调整为32\*32像素，将图像转换为浮点型，并进行归一化处理，然后将其存储在images列表中，接着根据文件名提取标签，将标签转换为一个长度为10的独热编码，并存储在labels列表中，返回处理后的图像和标签数组。创建一个lenet实例，调用lenet.train方法对加载的训练数据集进行训练，最后调用lenet.save方法将训练好的模型保存在文件中。

### 2.1.3 模型测试

模型测试时，首先加载并预处理测试数据集，遍历每个文件，读取灰度图像，并将其调整大小为32\*32像素，将图像转换为浮点型，并进行归一化处理，然后存储在images列表中，从文件名中提取标签，并将其存储在labels列表中。将处理后的图像和标签转化为NumPy数组。创建一个lenet实例，使用lenet.load方法加载之前保存的训练好的模型文件，再调用lenet.predict方法对测试数据集进行预测，得到预测结果，最后计算预测的准确率。

## 2.1实验结果与分析

实验训练，训练的数据集为MNIST数据集的60000张图像，经过500轮的训练，得到的模型的准确率为98%左右，如图2所示。

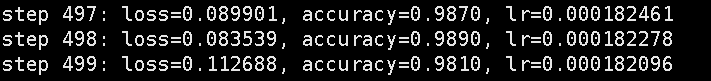


图2 训练结果

训练完毕后，采用模型测试程序对训练好的模型进行测试，测试的数据集为MNIST的10000张图像，测试得到的准确率为98.14%，如图3所示。



图3 测试结果

# Pytorch实现

## 3.1实验方法

训练模型的过程中，定义一个LeNet-5模型，包括初始化函数和前向传播函数。在初始化函数中定义模型的各个层，包括两个卷积层、两个池化层以及三个全连接层，每个层的参数包括输入通道数、输出通道数、卷积核大小、池化核大小和步长以及输出节点数。在前向传播函数中，定义了LeNet-5模型的前向计算过程，通过对输入x进行一系列卷积、池化和全连接操作。设置模型训练的超参数，包括批量大小、学习率和训练轮次。

准备训练和测试数据集，使用datasets.MNIST方法从torchvision库中加载MNIST数据集，指定数据集的存储路径，并将数据转换为PyTorch的张量形式，使用DataLoader将加载的数据集转换为可迭代的数据加载器，设置每个批次的样本数量。

实例化模型和优化器，使用LeNet5实例化一个LeNet-5模型，使用optim.SDG实例化一随机梯度下降优化器，用于优化模型的参数。定义模型训练后保存的路径以及保存的文件名。模型训练的过程中，将模型设置为训练模型，并遍历训练集中的批次数据，对每个批次的数据将数据输入模型进行前向传播，计算损失，根据损失计算梯度并更新模型，每一轮次的训练结束后，在测试数据集上评估模型性能。训练完毕后，将模型参数保存到对应的文件路径底下。

使用保存的模型进行预测，遍历测试集中的一个批次的数据，并随机选择其中一个样本，将选择的样本输入模型进行预测，得到预测结果。

## 3.2实验结果及分析

设置的训练轮次为500轮，当训练到100轮时，模型的准确率已经趋于一个稳定值，所以在训练的过程中可以设置训练轮次为100轮次。训练的模型准确率为98.5%左右，训练结果如图4所示。

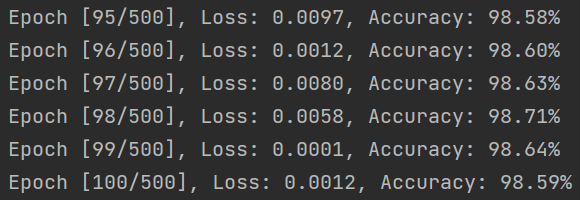


图4 训练过程

加载训练后保存的模型，对测试数据集进行预测，预测10个数据，其中存在两个预测错误，预测结果如图5所示。

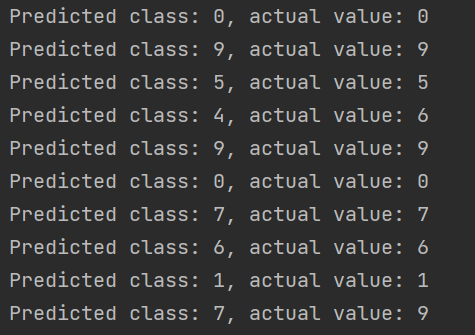


图5 预测结果