

人工智能实验报告

实验名称： 卷积神经网络-手写体数字识别

|  |  |  |  |
| --- | --- | --- | --- |
| 学生学号 | 202110120401 | 202110120420 | 202110120408 |
| 学生姓名 | 刘广润 | 赵威 | 喻慧超 |
| 分工 | 网络实现,测试算法性能，撰写报告 | 算法测试，绘制表并进行对比，撰写报告 | 优化算法、撰写报告 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验评价结果** | | | | |
| **占比** | 10% | 40% | 50% | 总分 |
| **评分标准** | 报告的规范性。报告中的术语、格式、图表、数据、公式、标注及参考文献是否符合规范要求。 | 报告的严谨性。结构是否严谨，论述的层次是否清晰，逻辑是否合理，语言是否准确。 | 实验的充分性。实验是否完备，数据是否合理，是否有创新性成果或独立见解。 |
| 评分 |  |  |  |  |

。

报告提交时间： 2023 年 12 月 23 日

# 试验任务

MNIST数据集是一个手写体数字的图像数据集，训练集包括60000张图片，测试集包括10000张图片，每张图片是一个8位的灰度图片，尺寸为28× 28,训练集的前20张图片如图1.1所示。现要求训练一个卷积神经网络，用于识别数字图片。

# 实验平台

Windows操作系统，Python编译环境，pytorch，matplotlib等程序库。

# 实验原理

## 神经网络基础

### 神经元

神经网络的基本构建块是神经元，模仿了人脑中的神经元。在人工神经网络中，一个神经元接收一组输入，对这些输入进行加权求和，然后通过一个激活函数产生输出。每个输入都有一个权重（weight），这些权重决定了相应输入的重要性。

### 激活函数

激活函数决定了一个神经元是否应该被激活，即是否对给定的输入信息做出反应。激活函数对输入信号进行非线性转换，使得神经网络可以学习和执行更复杂的任务。常见的激活函数包括Sigmoid、Tanh和ReLU（Rectified Linear Unit）。

### 前向传播

前向传播是神经网络计算输出的过程。在这个过程中，输入数据从输入层传递到隐藏层（如果有的话），最后到输出层。在每一层，神经元的输出由权重、输入和激活函数共同决定。输出层的结果是神经网络对给定输入的响应或预测。

### 反向传播

反向传播是训练神经网络时使用的主要算法。在这个过程中，网络的预测输出与实际输出进行比较，计算出损失函数（loss function）的值。损失函数衡量了预测值与实际值之间的差异。然后，这个损失被用来计算关于网络权重的梯度，即损失对每个权重的导数。最后，这些梯度被用于更新网络的权重，以减少损失。反向传播通常与梯度下降（gradient descent）优化算法结合使用，以优化网络的权重和偏置。

## 卷积神经网络

### 卷积神经网络的结构

卷积神经网络（CNN）是一种专门用于处理具有类似网格结构的数据的神经网络，例如图像（2D网格的像素）。CNN在图像识别、分类和处理等领域表现出色。它的核心结构包括卷积层、池化层和全连接层，每个层次都有其独特的作用和重要性。

### 卷积层

* 作用：卷积层是CNN的核心，主要负责从输入图像中提取特征。它通过卷积操作，使用一组可学习的过滤器（或称为核）对输入图像进行处理。每个过滤器负责从原始图像中提取某些特定类型的特征（如边缘、角点、纹理等）。
* 重要性：卷积层能够保持输入图像的空间关系，使网络能够学习图像中的空间层级和模式。此外，由于参数共享和局部连接的特性，卷积层具有较少的参数，从而减少了模型的复杂性和过拟合的风险。

### 池化层

* 作用：池化层（通常是最大池化或平均池化）跟在卷积层之后，用于降低特征图的空间维度（宽度和高度）。这通过减小卷积层输出的尺寸来降低后续层的计算负担，并在一定程度上提高模型的不变性（如平移不变性）。
* 重要性：池化操作有助于使特征表示更加紧凑和鲁棒。因为它减少了模型中可学习参数的数量，它还有助于降低过拟合的风险。

### 全连接层

* 作用：在CNN的末端，全连接层用于基于前面层提取的特征进行分类或回归。在这一层中，之前卷积和池化得到的特征图被展平为一维向量，然后通过一个或多个全连接层进行处理，以执行具体的任务（如分类）。
* 重要性：全连接层将卷积层和池化层提取的局部特征整合起来，用于进行最终的判决或预测。这是实现特定任务（如识别图像中的对象）的关键步骤。

## 图像处理在CNN中的应用

### 归一化

* 作用：归一化是调整图像数据的尺度，使其位于一个小的、指定的范围内，通常是0到1或-1到1。这通常通过将每个像素值除以255（像素的最大值）来实现。
* 重要性：归一化有助于模型更快地收敛，因为它确保了所有输入特征都在相同的尺度上。这还有助于避免网络中某些部分的激活值过大，从而防止梯度爆炸。

### 大小调整

* 作用：CNN要求输入图像具有固定的大小。大小调整是改变图像尺寸以满足网络输入要求的过程。
* 重要性：通过确保所有输入图像都具有相同的尺寸，CNN能够有效地学习和提取特征。此外，较小的图像尺寸可以减少模型的计算负担。

### 数据增强

* 作用：数据增强是通过应用一系列随机变换（如旋转、缩放、平移、翻转等）来增加训练数据多样性的技术。
* 重要性：数据增强有助于模型学习到更加鲁棒的特征表示，从而提高其对新、未见过的图像的泛化能力。这对于防止过拟合尤其重要，尤其是在训练数据有限的情况下。

### 颜色空间转换

* 作用：有时，将图像从一种颜色空间转换到另一种颜色空间（如从RGB到灰度或HSV）可以改善模型的性能。
* 重要性：不同的颜色空间可以突出图像的不同特征。例如，灰度图像消除了颜色信息，有助于模型专注于纹理和形状；而HSV空间可能更适合处理颜色和亮度的变化。

### 批处理归一化

* 作用：虽然不是图像预处理的直接步骤，但批处理归一化是在网络的每个层之后对激活值进行归一化的技术。
* 重要性：批处理归一化可以加速训练过程，提高模型的稳定性，有助于缓解内部协变量偏移问题。

# 实验步骤

## 数据集下载和预处理

### 图像预处理

* 将图像转换为张量格式，这对于使用PyTorch神经网络是必需的。
* 将图像像素值归一化到[-1, 1]的范围内，使用的均值和标准差均为0.5。

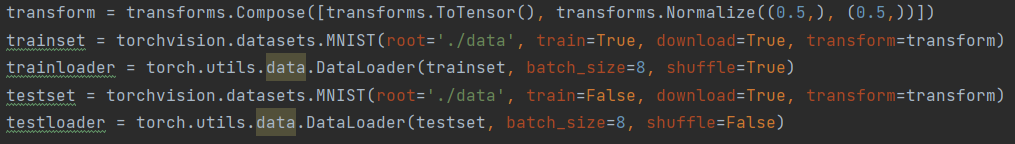
### 下载和加载MNIST数据集

* 下载MNIST手写数字训练数据集到本地目录'./data'，如果该目录下没有数据集则进行下载。
* 同样地，下载MNIST手写数字测试数据集到同一目录。

### 设置数据加载器

* 为训练数据集创建一个数据加载器，每个批次处理4个图像，数据在每个训练轮次开始时被打乱。
* 为测试数据集创建一个数据加载器，每个批次处理4个图像，但数据不会被打乱。

### 代码



## 构建模型

### 两个卷积层

* 第一个卷积层接受单通道输入（如灰度图像）并输出6个通道的特征图，使用的卷积核大小为5x5。
* 第二个卷积层接受第一个卷积层的6个通道输出作为输入，并进一步输出16个通道的特征图，卷积核大小同样为5x5。

### 最大池化层

* 网络中使用了一个2x2的最大池化层，用于在两个卷积层之后降低特征图的空间尺寸，减少后续全连接层的参数数量，从而控制过拟合。

### 三个全连接层

* 第一个全连接层将展平后的特征图（由于两次最大池化，特征图大小减小到4x4，通道数为16）转换为120个节点。
* 第二个全连接层将这120个节点进一步转换为84个节点。
* 第三个全连接层将这84个节点映射到10个输出节点，对应于10个数字类别。

### 激活函数ReLU

* 在卷积层和全连接层之后使用了ReLU（Rectified Linear Unit）激活函数，以引入非线性，使得网络能够学习复杂的模式。

## 训练过程

### 损失函数的选择

* 目的：损失函数（Loss Function）衡量模型的预测输出与实际标签之间的差异，是训练过程中优化的关键目标。
* 常用损失函数：对于分类问题，如MNIST手写数字识别，通常使用交叉熵损失函数（Cross-Entropy Loss）。这种损失函数对于输出概率分布与目标分布之间的差异进行量化，适用于多类分类问题。

### 优化器的选择

* 目的：优化器（Optimizer）决定了网络如何更新其权重来最小化损失函数。
* 使用的优化器：具有动量的SGD：考虑了前一步的更新，可以加快训练速度并减少震荡。

### Epoch

* 太少的epoch可能导致模型欠拟合，即模型没有充分学习数据特征。
* 太多的epoch可能导致过拟合，即模型过度学习训练数据中的噪声。
* 需要通过实验确定合适的epoch数，同时可以使用如早停（Early Stopping）等技术来防止过拟合。

## 超参调整

尝试了batchsize为4和batchsize为8。

# 实验结果分析

## 实验指标

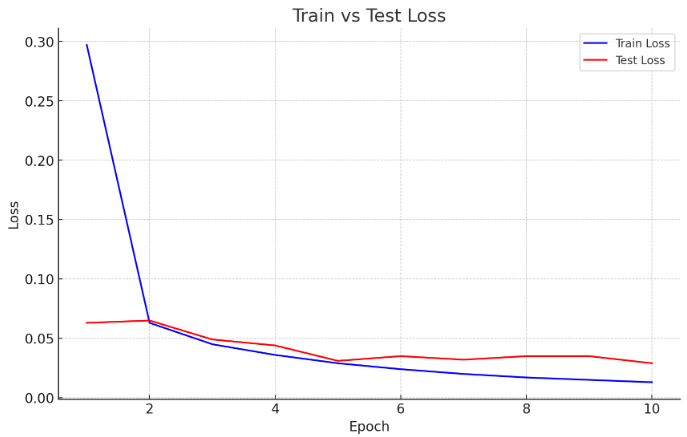


图 1 batchsize=4

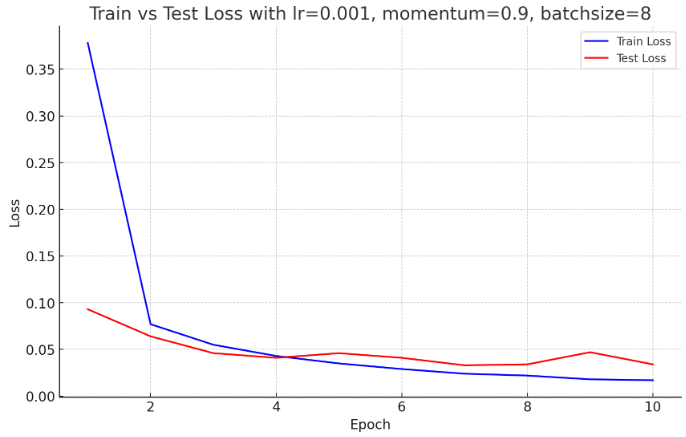


图 2 batchsize=8

## 结果分析

### 大Batch Size

* 计算效率：更大的批处理大小可以更充分地利用高效的矩阵运算，尤其是在使用GPU时。这可以减少训练时间。
* 稳定的梯度估计：由于考虑了更多的样本，大批处理大小可以提供更稳定、更准确的梯度估计。
* 内存利用率：在某些情况下，较大的批处理可以更有效地利用硬件资源，如GPU的内存。
* 内存要求：大批处理大小需要更多的内存，这可能超出某些系统的硬件限制。
* 过拟合风险：较大的批处理大小可能导致模型过度拟合训练数据。
* 可能陷入局部最小值：大批处理大小有时会导致优化过程陷入局部最小值或鞍点。

### 小Batch Size

* 更好的泛化能力：小批处理大小有助于增加训练过程中的噪声，这可以作为一种正则化形式，提高模型的泛化能力。
* 快速收敛：小批处理可以加快权重更新的频率，从而加速学习过程。
* 内存友好：小批处理大小对内存的要求较低，适用于内存限制的硬件。
* 梯度估计的方差较大：较小的样本量可能导致每次迭代的梯度估计更加不稳定。
* 计算效率较低：小批处理大小可能无法充分利用现代硬件（特别是GPU）的高度并行性能。
* 可能需要更多的Epoch：由于每个批次的样本数量减少，可能需要更多的迭代次数（Epoch）来完成训练。

# 代码

https://github.com/SonicAge1/ConvNet