|  |  |  |
| --- | --- | --- |
| 第二章 神经网络设计实验 | | |
| 姓名：吴文韬 | 学号：20373914 | 班级： 202111 |
| **注意：课程实验代码和报告需独立完成，严禁抄袭!如发现有抄袭情况（同年级或者抄袭高年级），相应的实验或报告部分直接0分。** 实验2.1基于三层神经网络实现手写数字分类(60’’) **1.介绍一下“手写数字分类任务”和本实验中使用的数据集。**  **手写数字分类任务：该实验主要步骤如下**   1. **数据预处理：将手写数字图片转换为向量形式，并将数据集分为训练集和测试集** 2. **网络结构设计：设计三层神经网络结构，全连接层、ReLU激活函数层和Softmax损失层** 3. **前向传播：将输入数据通过神经网络进行前向传播，计算输出结果**   def forward(self, input): *# 前向传播计算*  start\_time = time.time()  *self*.input = input  *# 全连接层的前向传播，计算输出结果*  *self*.output = *self*.input.dot(*self*.weight) + *self*.bias  *return* *self*.output   1. **反向传播：根据输出结果和标签数据计算误差，并通过反向传播算法更新网络参数**   def backward(self, top\_diff): *# 反向传播的计算*  *# 全连接层的反向传播，计算参数梯度和本层损失*  *self*.d\_weight = np.matmul(*self*.input.T, top\_diff)  *self*.d\_bias = np.matmul(np.ones([1, top\_diff.shape[0]]), top\_diff)  bottom\_diff = np.matmul(top\_diff, *self*.weight.T)  *return* bottom\_diff   1. **最后从文件读取训练得到的模型参数，调用网络推断模块测试网络的精度**   **数据集：**  **MNIST手写数字库介绍：**  **该库是一个常用的机器学习数据集，包含了大量手写数字图片以及标签，由美国国家标准与技术研究所（NIST）收集并发布，包含了60000张训练图片和10000张测试图片，每张图片都是28x28像素的灰度图像，每个像素的值在0到255之间。每张图片都对应一个0到9之间的数字标签，表示这张图片上的手写数字是几**  **2.本实验的神经网络都包含哪些层？各个层的功能是什么？**  **包含了三个全连接、两个ReLU激活函数层和一个Softmax损失层**  **全连接层：将上一层的所有神经元都连接到本层的每个神经元上，从而实现特征的非线性变换和降维。在反向传播过程中，全连接层计算参数梯度和本层损失，用于更新网络参数**  **ReLU激活函数层：将全连接层的输出进行非线性变换，增强网络的表达能力，解决梯度消失问题，在反向传播中，ReLU层将上一层的误差传递给下一层，用于更新网络参数**  **Softmax损失层：计算网络的预测值与真实标签之间的差距，并将其转化为损失值，在反向传播过程中，Softmax损失层计算输出层的误差，并将误差传递到前一层，用于更新网络参数**  **3.网络的训练(training)过程和网络的推理(inference)过程有什么区别？**  **输入数据：训练已知、推理未知**  **参数更新：训练需要根据损失函数进行反向传播更新、推理固定**  **输出结果：训练需要与标签数据进行比较，计算损失函数并反向传播，推理则直接输出最终预测结果**  **迭代次数：训练多次、推理只需一次前向传播**  **4.参数和超参数的区别是什么？本实验中都使用了哪些超参数？**  **参数是神经网络中需要学习的变量，其值是通过训练数据和反向传播算法来更新的；而超参数是神经网络中需要手动设置的变量，以便更好的优化模型**  **本实验中使用了如下超参数：**  **std标准差、batch\_size样本数、input\_size输入层神经元个数、hdden1第一个隐藏层的神经元个数、hidden2第二个隐藏层的神经元个数、out\_classes输出层的神经元个数、lr学习率、max\_epoch最大迭代次数**  **5.什么是正则化？有哪些正则化技术？**  正则化是一种用于防止模型过拟合的技术  常见的正则化包括：L1正则化、L2正则化、Dropout正则化、数据增强 实验2.2 基于 DLP 平台实现手写数字分类(30’’) **1.什么是量化？量化会带来什么好处/坏处？**  **量化是指将模型中的浮点数参数和激活值转化为定点数或整数，从而减少模型计算量的过程。**  **好处：加快模型的计算速度和减少模型占用存储空间**  **坏处：损失模型精度、需要调参（增大模型使用难度）**  **2.pycnml、CNML库以及DLP之间的关系是什么？**  **pycnml和CNML都是用于深度学习加速的库，其中 CNML 是 Cambricon 公司开发的深度学习加速库，而 pycnml 是 CNML 的 Python 接口。CNML 库提供了一系列的深度学习算子实现，包括卷积、池化、全连接、激活函数等，可以在 Cambricon 公司的 AI 芯片上进行加速。pycnml 库则提供了 CNML 库的 Python 接口，使得用户可以在 Python 中使用 CNML 库提供的深度学习算子**  **DLP则是Cambricon 公司开发的深度学习框架，它基于 pycnml 和 CNML 库实现了一系列的深度学习算法和模型，包括卷积神经网络、循环神经网络、目标检测、语音识别等。**  **DLP 框架可以在 Cambricon 公司的 AI 芯片上进行加速，从而提高深度学习模型的运行速度和能效。**  **3.使用DLP相对于CPU有哪些优势和劣势?**  优势：因为可以通过Cambrion的AI芯片进行加速（并且其公司也对DLP进行了一系列特定优化），所以可以实现更快的运行速度、更高的能效、更小的模型尺寸  劣势：同样也是因为AI芯片的限制问题，DLP需要依赖特定的硬件，使用DLP需要购买特定AI芯片，增加学习成本；学习路线问题，DLP是新的深度学习框架，社区不如TensorFlow或PyTorch，遇到问题可能无法及时解决。 | | |