2023－2024学年第一学期

****

### 实 验 报 告



课 程 名 深度学习理论与实践

课 程 号

学 生 姓 名

学 生 姓 号

专 业 班 级

所 在 学 院 计算机与计算科学学院

指 导 老 师

实验报告日期： 2024年 9 月 27 日

**课程实验****清单**

**浙大城市学院实验报告**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验项目统一编号 | 31C0120501 | | 实验项目名称 | 102Flowers神经网络分类 |
| 实验时间 | 2小时 | | 实验地点 | 理4-220 |
| 小组合作：□是 ☑否 | | 实验人员： | | |
| 一、实验目的和要求（或设计要求及指标）  **目的**：通过实验巩固神经网络、随机梯度下降法、激活函数以及超参数等深度学习基础知识；熟悉深度学习模型训练和测试的基本代码流程；  **要求**：对比实验验证1）神经网络深度、2）激活函数、3）batchsize、4）学习率、以及5）不同优化器对模型收敛和性能等方面的影响。 | | | | |
| 二、实验内容和原理（或设计方案及原理）  1. 神经网络实现分类器。代码实现数据集的读取、模型定义、迭代优化和测试以及训练信息保存等代码；  2. 对比神经网络不同深度对分类收敛和性能的影响。  3. 对比不同激活函数对分类收敛和性能的影响  4. 对比batchsize对分类收敛和性能的影响。  5. 对比不同学习率对分类收敛和性能的影响。  6. 对比不同优化器对分类收敛和性能的影响。 | | | | |
| 三、主要仪器设备及工具（仪器设备名称、型号规格）或开发设计软件及工具名称  电脑：thinkbook16 | | | | |
| 四、操作方法与实验步骤（或设计方法与实施过程）  1.准备数据集torchvision.datasets.Flower102.  2.通过上课已有的代码模型进行实验对比项目。  import torch  import torchvision  from torchvision import transforms  from torch.utils.data import DataLoader  from flower\_c import FlowerClassifier  if \_\_name\_\_ == '\_\_main\_\_':      transform = torchvision.transforms.Compose([      torchvision.transforms.Resize((224, 224)),      torchvision.transforms.ToTensor(),      torchvision.transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  ])      #准备数据集 train, test, val      train\_set = torchvision.datasets.Flowers102(          root=r'D:\dataset',          split = 'train',          transform = transform,          download = True      )      test\_set = torchvision.datasets.Flowers102(          root=r'D:\dataset',          split = 'test',          transform = transform,          download = True      )      val\_set = torchvision.datasets.Flowers102(          root=r'D:\dataset',          split = 'val',          transform = transform,          download = True      )      print(len(train\_set))      print(len(val\_set))      train\_dataloader = torch.utils.data.DataLoader(dataset = train\_set,                                                  batch\_size = 102,                                                  shuffle =True, #是否打乱样本                                                  num\_workers =2)#子进程个数，加快进程读取      test\_dataloader = torch.utils.data.DataLoader(dataset =test\_set, batch\_size = 102, shuffle =True, num\_workers =2)      val\_dataloader =torch.utils.data.DataLoader(dataset =val\_set, batch\_size = 102, shuffle =True, num\_workers =2)      # for xs, labels in train\_dataloader:      #     print(xs.shape, labels.shape)      model =FlowerClassifier(num\_classes = 102)      device  = 'cpu'      model.to(device)  #初始化损失函数      loss\_ce = torch.nn.CrossEntropyLoss()      optimizer = torch.optim.SGD(model.parameters(),lr = 0.001,momentum=0.9)  #循环优化模型      num\_epochs =100          for epoch in range(num\_epochs):          print('Epoch: #',epoch+1)    Running\_loss =0          for xs, labels in train\_dataloader:              #注意数据tensor和model必须在同一个设备              xs = xs.to(device)              xs=xs.reshape(102,-1)              labels = labels.to(device)              ys\_logit = model(xs)              ce\_loss = loss\_ce(ys\_logit,labels)              running\_loss += ce\_loss.item()              optimizer.zero\_grad()              ce\_loss.backward()              optimizer.step()          print('Start validation...')          correct = 0          total = 0          with torch.no\_grad():              for xs,labels in val\_dataloader:                  xs = xs.to(device)                  xs=xs.reshape(102,-1)                  labels = labels.to(device)                  ys\_logit = model(xs)                  pred\_labels = torch.argmax(ys\_logit,dim=-1)                  total +=labels.size(0)                  correct += (pred\_labels==labels).sum().item()                # torch.save(model.state\_dict(), 'flower\_classifier.pth')          print('Epoch [{}/{}], Loss: {:.4f}'.format(epoch + 1, num\_epochs, ce\_loss.item()),'Accuracy', 100\*correct/total)      print('Training complete')   1. 对比神经网络不同深度对分类收敛和性能的影响，对比神经网络不同深度对分类收敛和性能的影响。（二层）   class FlowerClassifier(nn.Module):      def \_\_init\_\_(self, num\_classes=102):          super(FlowerClassifier, self).\_\_init\_\_()          self.network = nn.Sequential(              nn.Linear(in\_features=224\*224\*3, out\_features=2048),              nn.ReLU(),              nn.Linear(in\_features=2048, out\_features=num\_classes)  # 直接连接到输出层          )        def forward(self, xs):          xs = xs.view(xs.size(0), -1)          ys\_logit = self.network(xs)          return ys\_logit   1. 对比不同激活函数对分类收敛和性能的影响.使用不同激活函数（Sigmoid、Tanh）对模型性能进行对比。（Sigmoid）   class FlowerClassifier(nn.Module):      def \_\_init\_\_(self, num\_classes=102):          super(FlowerClassifier, self).\_\_init\_\_()          self.network = nn.Sequential(              nn.Linear(in\_features=224\*224\*3, out\_features=2048),              nn.Sigmoid(),              nn.Linear(in\_features=2048, out\_features=1024),              nn.Sigmoid(),              nn.Linear(in\_features=1024, out\_features=256),              nn.Sigmoid(),              nn.Linear(in\_features=256, out\_features=102)          )        def forward(self, xs):          xs = xs.view(xs.size(0), -1)          ys\_logit = self.network(xs)          return ys\_logit   1. 对比batchsize对分类收敛和性能的影响。设置不同的 batch size（32, 64, 128）观察对收敛和性能的影响（batchsize=64）       train\_dataloader = torch.utils.data.DataLoader(dataset = train\_set,                                                 batch\_size = 64,                                                 shuffle =True, #是否打乱样本                                                 num\_workers =2)   1. 对比不同学习率对分类收敛和性能的影响,测试不同的学习率（ 0.01, 0.001, 0.0001）对训练收敛速度和结果的影响。（lr=0.0001）   optimizer = torch.optim.SGD(model.parameters(),lr = 0.0001,momentum=0.9)  7.对比不同优化器对分类收敛和性能的影响,使用 SGD 和 Adam 等不同优化器，观察对模型训练效果的影响。（Adam）  optimizer = torch.optim.Adam(model.parameters(), lr=0.001) | | | | |
| 五、实验数据记录和处理（或设计效果）  恒定对比参数 四层神经网络， ReLU激活函数，batchsize 102,学习率0.001.  优化器SGD     1. 两层神经网      1. 激活函数为Sigmoid      1. batchsize=64      1. lr=0.0001 2. 优化器为Adam   屏幕截图 2024-09-28 111548 | | | | |
| 六、实验结果与分析（或设计成效分析）  请写明！需要有图表可视化结果展示，比如不同深度网络训练过程中的损失函数曲线和验证集准确率曲线。表格给出测试集的accuracy、recall、f1-score、AUC等指标对比结果。  原固定参数的数据图和表格       1. 两层神经网     **过拟合减少**：两层神经网络减少了过拟合的风险，使得模型更容易泛化。  **梯度传递更好**：浅层网络避免了深层网络中可能出现的梯度消失问题。  **模型复杂度与数据适配**：两层神经网络足以捕捉数据特征，而不需要过度复杂的模型。  **训练速度提升**：参数减少后，每轮训练的计算开销降低，优化过程加快。   1. 激活函数为Sigmoid      ****梯度消失问题**** **Sigmoid 函数的输出范围**在 [0,1] 之间，这导致它在接近 0 或 1 的区域时，梯度非常小。尤其是当网络比较深时，Sigmoid 的梯度会快速衰减，导致网络的参数更新非常缓慢，甚至无法有效更新。相较之下，**ReLU** 在正数区域的梯度是常数，因此更适合深度网络。  由于梯度消失，导致模型在训练时参数更新非常微小，训练无法有效进行。这也会导致损失函数稳定在一个较高的值，而模型的准确率无法提升。 ****Sigmoid 的饱和区域**** 在输入值非常大或非常小时，Sigmoid 函数的输出接近 0 或 1，**进入饱和区域**，此时导数接近 0。这会导致反向传播时几乎没有梯度传回去，网络无法有效更新参数，损失也可能停滞在某个值   1. batchsize=64       **梯度噪声增加**：较小的 batch size 会导致每个批次包含的样本数量减少，梯度估计的方差增大。这意味着每次参数更新时，梯度方向的波动性更大，优化过程可能变得不稳定，损失的下降速度减慢。  **梯度更新的不稳定性**：在小 batch size 下，模型可能更容易陷入局部最优或鞍点，因为梯度更新受噪声影响较大。   1. lr=0.0001       学习率从 0.001 降低到 0.0001 时，出现了损失下降变慢、准确率部分下降的现象，  **步长变小**：学习率控制模型权重更新的步长。较小的学习率（0.0001）意味着每次优化时更新的幅度变小，因此模型的收敛速度变慢。这就导致损失函数的下降速度变慢。  **更新不足**：学习率过小可能会导致模型在优化过程中每次权重调整幅度太小，无法显著改善模型性能。这会导致损失和准确率的改善不明显，甚至停滞。  **局部最小值或平台效应**：较小的学习率可能会导致模型卡在局部最小值或优化的“平台”区域，无法有效跳出，从而使损失和准确率长期没有显著变化。     1. 优化器为Adam       虽然 Adam 在早期损失下降更快，但在后期，SGD 通过稳定的步长调整，可能逐渐追上 Adam 的效果，因此准确率变化类似。 | | | | |
| 七、讨论、心得  **深度网络**：四层网络在处理复杂任务时能够提取更多的特征，但浅层网络（两层）在一般任务中收敛更快，性能提升更明显。  **激活函数**：ReLU 更适合深层网络和大规模任务，而 Sigmoid 容易导致梯度消失。  **batchsize**：大 batchsize 收敛快，但小 batchsize 有助于泛化能力的提升。  **学习率**：较大的学习率收敛快，但容易波动；较小的学习率收敛慢，但最终性能更好。  **优化器**：Adam 在初期收敛更快，但最终性能与 SGD 类似，尤其在任务简单时，SGD 表现依然稳健。 | | | | |
| 八、指导教师评语  实验报告评分（百分制）： 分  指导教师签名：  日 期： 年 月 日 | | | | |