实 验 报 告

|  |  |
| --- | --- |
| 学年学期： | 2023 -- 2024 学年第一学期 |
| 课程名称： | 深度学习实验 |
| 指导老师： | 刘 立 |
| 学院（部）： | 计算机学院 |
| 班 级： |  |
| 学 号： |  |
| 姓 名： |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 目标 | 实验一 | 实验二 | 实验三 | 实验四 |
| 成绩 |  |  |  |  |
| 教师签名 |  | | | |



实 验 报 告

第 十二 周

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 | 基于百度API大模型的图像收集与动物识别 | | |
| 实验性质  （必修、选修） | 必修 | 实验类型  （验证、设计、创新、综合） | 设计 |
| 实验课时 | 4 | 实验日期、时间 | 5.25 |
| 实验仪器设备  (实验软硬件要求) | 1、计算机  2、Windows  3、Jupyter | | |
| 实验目的 | 1、Jupyter安装  2、百度API的调用  3、python爬取识别图片  4、百度动物识别模型的调用与使用 | | |
| 实验内容 | 1. 使用python爬取识别的图片并将其保存在文件夹中、 2. 调用百度动物识别模型，提取出其关键字与动物特征。 | | |
| 成绩评定 |  | | |

|  |
| --- |
| 1. **实验基本原理**   动物识别技术的基本原理是通过对动物图像或视频数据进行分析和处理，提取出有效的特征表示，并将其与预先建立的动物类别进行比较和匹配，从而实现对动物种类的识别和分类。  基于人工特征的野生动物识别方法主要通过人工对野生动物图像中具有辨识度的特征信息进行提取，并通过特征比对的方式就可以对野生动物所属的类别进行识别判断。   1. **实验内容** 2. 使用python爬取识别的图片并将其保存在文件夹中 3. 调用百度动物识别模型，提取出其关键字与动物特征。 4. **实验环境**   Jupyer软件   1. **实验步骤** 2. 爬取图片   （1）代码示例:  # 图片数据集代码  import re #筛选url  import requests #请求  import os #创建文件夹  word = input("请输入要下载的图片:")  #发送请求需要一个网址，一个地址。这个地址是由用户自己寻找的  url = 'https://image.baidu.com/search/acjson?tn=resultjson\_com&logid=5853806806594529489&ipn\  =rj&ct=201326592&is=&fp=result&fr=ala&word={}&queryWord={}&cl=2&lm=-1&ie=utf-8&oe=utf-8&adpicid\  =&st=&z=&ic=&hd=&latest=&copyright=&s=&se=&tab=&width=&height=&face=&istype=&qc=&nc=&expermode=\  &nojc=&isAsync=&pn=30&rn=30&gsm=1e&1658411978178='.format(word, word)  # 伪装浏览器  headers = {"User-Agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)\  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36 Edg/99.0.1150.39'}  files = './images/{}'.format(word) #创建文件夹路径  if not os.path.exists(files): #如果没有文件，就执行下面的语句  os.makedirs(files) #创建文件，如果有的话，将会被覆盖    # 发送请求  ans = requests.get(url=url,headers=headers) #关键字赋值  if ans.status\_code != 200:  print("网址出错，找不到：",ans.status\_code)  req = ans.text #将请求回来的数据变成文本  res = '"thumbURL":"(.\*?)"' #正则式  data = re.findall(res,req) #根据正则式进行筛选  i = 1  for item in data: #遍历data的数据  i += 1  response = requests.get(url=item,headers=headers).content #将图片转为二进制文件  file = files+'\\'+str(i) + 'li.jpg'  with open(file,'wb') as f:  f.write(response)   1. 输入要提取的图片类型   在这里我提取出了一种动物类型（狐狸）与一种非动物类型（乌冬面）的图片:  图片3  **图 1执行乌冬面类型图片下载**  图片4  **图 2执行狐狸类型图片下载**  **图片5**  **图 3乌冬面类型的图片展示**  **图片6**  **图 4狐狸类型的图片展示**   1. 判断种类   （1）from aip import AipImageClassify  """ 你的APPID AK SK 三个密匙"""  APP\_ID = '16892891'  API\_KEY = 'F1E5bADOdfa9e7Wqck98xpVh'  SECRET\_KEY = 'XS6Xz4Q1Vf7sutxtODWVXQISgD7C7vbn'  client = AipImageClassify(APP\_ID,API\_KEY,SECRET\_KEY)  def get\_file\_content(filePath):  with open(filePath,'rb') as fp:  return fp.read()  image = get\_file\_content('./images/乌龟/11li.jpg')#识别demo.png 图片  ''' 调用动物识别 '''  result = (client.dishDetect(image))  i = 0  while(i < 5):  print("种类:",result["result"][i]["name"])  #print("概率:",float(result["result"][i]["score"])\*100)  i += 1  （2）.执行种类判断  对乌龟进行菜类判断发现乌龟不是菜类  图片7  **图 5 狐狸种类（菜）判断**   1. 执行动物种类判断与特征提取   （1）.代码示例:  from aip import AipImageClassify  """ 你的APPID AK SK 三个密匙"""  APP\_ID = '16892891'  API\_KEY = 'F1E5bADOdfa9e7Wqck98xpVh'  SECRET\_KEY = 'XS6Xz4Q1Vf7sutxtODWVXQISgD7C7vbn'  client = AipImageClassify(APP\_ID,API\_KEY,SECRET\_KEY)  def get\_file\_content(filePath):  with open(filePath,'rb') as fp:  return fp.read()  image = get\_file\_content('./images/乌龟/11li.jpg')#识别demo.png 图片  ''' 调用动物识别 '''  # 使用options 获取 百度百科信息  options = {}  options["baike\_num"] = 5 #找到百科的前5 个信息  #带参数 调用动物识别  #print(client.animalDetect(image,options))  respon = client.animalDetect(image,options)  respon\_list = respon["result"]  for item in respon\_list:  print("名字:",item["name"])  description = item.get("baike\_info", {}).get("description", "No description available")  # item.get("baike\_info", {}) 首先尝试从当前字典中获取 "baike\_info" 键对应的值。如果该键不存在，它将返回一个空字典 {} 作为默认值。  # 然后，.get("description", "No description available") 尝试从 "baike\_info" 字典中获取 "description" 键对应的值。  # 如果该键不存在，它将返回字符串 "No description available" 作为默认值。  print("介绍:",description)  (2).执行动物识别与特征提取  对地球执行时可以发现不是动物种类  **图片8**  **图 6 地球的动物判断**  对乌龟执行时可以发现是动物种类并且提取出了特征。  **图片9**  **图 7 乌龟的动物判断**   1. **实验心得**   在这个实验中，我学到了如何整合不同的技术工具和平台来解决具体问题，例如Python编程、网络爬虫和百度API。通过编写爬虫脚本从百度图片搜索中获取动物图片，我掌握了网络请求发送、响应解析和文件操作等技能，这显著提升了我的实际操作经验。在调用百度的动物识别模型时，我深切体会到深度学习模型在图像识别中的高效性和准确性。通过API接口调用，我无需深入了解模型的内部结构，就能快速获得动物识别的结果，这个过程让我更加深刻地理解了AI技术在实际应用中的价值。另外，通过设置参数从百度百科获取动物的详细信息，我学会了如何处理和分析复杂的JSON数据结构，以及如何优雅地应对数据缺失问题。使用.get()方法安全地访问字典中的键值，有效避免了由于键不存在而导致的程序中断，这是处理不确定数据源时的一项实用技巧。  整个实验过程中，我不断强化了自己的Python编程基础，特别是在处理网络请求、正则表达式和文件操作方面。我学会了根据需求构建简单的爬虫，包括请求的定制、响应的解析、数据的提取和存储，这为未来的数据采集项目奠定了坚实的基础。我还熟练掌握了第三方API的调用方法，包括认证、参数设置和结果处理，这对于未来开发集成多种服务的应用至关重要。通过解析JSON数据，我学会了从大量数据中提取有价值的信息，并学会了如何优雅地处理数据缺失的情况。 |

实 验 报 告

第 13 周

|  |  |  |  |
| --- | --- | --- | --- |
| 实 验 名 称 | 实验二：基于 LeNet 手写数字体识别 | | |
| 实验性质 | 必修 | 实验类型 | 设计型 |
| 实验课时 | 2 | 实验日期、时间 | 2024年6月1日 |
| 实验消耗器材 | 无 | | |
| 实验仪器设备 | 硬件环境 JetsonNX  操作系统 Linux  实验设备 机械臂，Jetson NX  实验配件 教具 | | |
| 实验目的 | （1）了解 LeNet 网络结构特点及优化方法；  （2）掌握手写数字识别方法；  （3）掌握深度学习模型预测与可视化方法。 | | |
| 实验内容（实验原理、运用的理论知识和数据、算法、程序、步骤和方法） | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **实验基本原理**   **1.激活函数**  多层感知机和线性回归相似的根本问题在于二者都只是简单地仿射变换，即使再多  的隐藏层也只是简单的叠加仿射，因此需要引入非线性函数来使得隐藏层输出成为非线  性变换，这种非线性函数就是激活函数，有了这样的非线性激活函数以后，神经网络的  表达能力更加强大了。每一层的输出通过激活函数之后，就会变得比以前复杂很多，从  而提升神经网络的表达能力。常见的激活函数有：  **Sigmoid 函数:**    其中*e* 是纳皮尔常数，其值是 2.7182…。  **ReLU 函数**：  又称为修正线性单元，是一个分段函数: *F(x)*  max*(* 0*,x)* ，大于 0 的数直接输出，小  于 0 的数则输出为 0，在 0 这个地方不连续，优点为在随机梯度下降的训练中收敛很快，  在输入为正数的时候，不存在梯度饱和问题。  **2.Mnist 训练数据集**  用于分类的训练数据有很多，由于本实验不对数据集进行分析，因此可以直接读取Mnist 数据集。希望卷积神经网络通过训练集能够训练出对数据类别准确分类的模型，Mnist 数据集总共包含 10 个类别的数据，图片的形状为 28\*28 像素。因此，网络的输入是 28\*28 像素手写数字，输出是数字 0~9，在本实验中，考虑到硬件的限制，因此设置的两层隐藏卷积神经网络的单元为 4,8。  **3.交叉熵损失函数**  损失函数在模型训练中用来衡量真实值和预测值之间的差距，数值越小代表真实值和预测值越接近，交叉熵常用来表示衡量两个不同概率分布的差异程度，在深度学习中就是真实概率分布和预测概率分布之间的差异。真实标签可以用类别分布表示，对于真实样本 *i*，可以构造标签个数的向量空间*Y* 。使得 *y i* 个元素为 1，其余的为 0，这样，在训练过程中只需要预测值 *y i*    真实概率尽可能地接近 1 就行。比如在图像分类中，*y i*  3，只需要预测值 *y i* 比其余的 *y i-*1 和 *y i*1 都大就可以了，尽管 *y i* 只有 0.5，那他也是预测正确  的，但平方误差就要求严格很多，那么交叉熵是对两个概率分布的衡量差异最合适的函数，它的数学公式如下：  *H(x)*   *p(xx )*log*(p(xx ))*  **4.随机梯度下降**  通过随机挑选一组模型的初始值，在优化的过程中对选择的参数进行多次迭代，每  次迭代的过程中都可能降低损失函数的值。在迭代的过程中，求小批量数据样本组成的  训练样本，并求取它们的平均损失来作为模型参数的梯度，最后和一个预先设定的正数  的乘积来作为依次迭代的减小量。   1. **实验内容**   手写数字识别是 OCR 字符识别中经典数据集。本实验通过构建简版的 LeNet 网络模型，运用卷积、池化、激活等常用的构建神经网络结构，利用数据集 Mnist 进行模型训练，并在最后对从未进行训练的测试图片进行识别，建立完整的神经网络运行理念。   1. **实验环境**   操作系统 Linux   1. **实验步骤**   **1.**运行 **jupyter lab**  （1）打开桌面的“实验 ”文件夹，在空白处单击鼠标右键，再点击“在终端打开 ”， 在终端界面输入“jupyter lab ”；  （2）在jupyter lab 编程界面选择 Notebook 下 Python3 ，进入程序编辑器；  （3）鼠标右键单击“未命名.ipynp ”的新建程序块，选择“Rename ”将程序名命 为实验名称。  **2.**导入库文件及函数  首先引入 Torch、Torchvision、Numpy 等相关库，利用 Torch、Torchvision 相关库中的对象与函数建立本实验相关模型以及卷积池化操作。利用 MatPlotLib 库来显示本实验训练所需的图像。  ***\*****参考代码****: \****   |  | | --- | | import torch  import time  import torchvision  import torch.nn as nn import numpy as np  import torchvision.transforms as transforms import matplotlib.pyplot as plt  from tqdm import tqdm |   **3.建立数据集**  （1）下载数据集  通过 Torchvison 下载 Mnist 数据集，其中包括训练集和测试集两个数据集，该数据 集包含了 10 个类别，分别为:0 、1 、2 、3、4 、5 、6 、7 、8 、9 ，并将其设为相应的标签。 且数据集为灰度图像，因此通道数为 1。  ***\*参考代码: \****   |  | | --- | | batch\_size = 4  num\_workers = 2  #下载训练集到指定目录  mnist\_train = torchvision.datasets.MNIST(root=**' /home/retoo/Desktop/实验/数据集/3.深度学习** **/9\_1DataSets/MNIST'**, train=True, download=True, transform=transforms.ToTensor())  # 下载测试集到指定目录  mnist\_test = torchvision.datasets.MNIST(root=**' /home/retoo/Desktop/实验/数据集/3.深度学习** **/9\_1DataSets/MNIST'**, train=False, download=True, transform=transforms.ToTensor())  # 将训练集组成一个批次  train\_iter = torch.utils.data.DataLoader(mnist\_train, batch\_size=batch\_size, shuffle=True, num\_workers=num\_workers)  # 将测试集组成一个批次  test\_iter = torch.utils.data.DataLoader(mnist\_test, batch\_size=batch\_size, shuffle=True, num\_workers=num\_workers) |  1. 选数据集中第一个数据，查看其数据维度   ***\*参考代码: \****   |  | | --- | | feature, label = mnist\_train[10]  plt.imsave(**'/home/retoo/Desktop/实验/数据集/3.深度学习** **/9\_1DataSets/test\_img/5.5.jpg'**,feature[0,:].numpy())  print(feature.shape, label) # Channel x Height x Width |   输出结果如下:  6758fb7f5073e6644020381834de523  **4.定义模型**  本实验模型采用 LeNet 网络，它与前面所学习的多层感知机不同的是采用了多个卷 积函数对图片进行二维的卷积的操作（这里 Mnist数据图像只有一层通道），通过设置 不同的卷积核尺寸以及填充来控制输出特征的尺寸大小，在两层卷积操作后连接两层全 连接，在最后全连接输出层输出类别数量维度，可以实现对 Mnist数据的模型训练。  ***\*参考代码: \****  ***截图20240601085306***  **5.定义损失函数**  交叉熵损失已经在 Torch 库中定义，可以直接调用，这里直接在 Torch 库中直接调 用。  ***\*参考代码: \****   |  | | --- | | loss = torch.nn.CrossEntropyLoss() |   **6.定义优化函数**  优化器已经在 Torch 库中定义，可以直接调用，这里直接使用Torch 库中直接调用。  optimizer **=** torch**.**optim**.**Adam**(**net**.**parameters**(),** lr**=**lr**)**  **7.定义训练流程**  **（1）精度评价函数**  定义在训练的过程中需要同时统计测试集的预测情况来监控模型是否出现过拟合 或者欠拟合的情况，这里定义一个精度评价函数完成对测试集的监控。  ***\*参考代码: \****   |  | | --- | | **def** evaluate\_accuracy(data\_iter, net, device=**None**):  **if** device **is None and** isinstance(net, torch.nn.Module):  # 如果没指定 device 就使用 net 的 device  device = list(net.parameters())[0].device acc\_sum, n = 0.0, 0  **with** torch.no\_grad():  **for** X, y **in** data\_iter:  **if** isinstance(net, torch.nn.Module):  net.eval() # 评估模式, 这会关闭 dropout  acc\_sum += (net(X.to(device)).argmax(dim= 1) == y.to(device)).float().sum().cpu().item()  net.train() # 改回训练模式 **else**: # 自定义的模型,  **if**(**'is\_training' in** net. code .co\_varnames): # 如果有 is\_training 这个参数 # 将 is\_training 设置成 False  acc\_sum += (net(X, is\_training=**False**).argmax(dim= 1) == y).float().sum().item() **else**:  acc\_sum += (net(X).argmax(dim= 1) == y).float().sum().item()  n += y.shape[0] **return** acc\_sum / n |   **（2）定义训练函数**  使用小批量随机梯度下降来优化模型的损失函数。在训练模型时，迭代周期数 num\_epochs 和学习率 lr 都是可以调整的超参数。改变它们的值可能会得到分类更准确 的模型。  ***\*参考代码: \****   |  | | --- | | **def** train\_ch5(net, train\_iter, test\_iter, batch\_size, optimizer, device, num\_epochs) : net = net.to(device)  print(**"training** **on** **"**, device) |  |  | | --- | | **for** epoch **in** range(num\_epochs):  train l sum, train\_acc\_sum, n, batch\_count, start = 0.0, 0.0, 0, 0, time.time() **for** X, y **in** train\_iter :  X = X.to(device) y = y.to(device) y\_hat = net(X)  l = loss(y\_hat, y)  optimizer.zero\_grad() l.backward()  optimizer.step ()  train l sum += l.cpu().item()  train\_acc\_sum += (y\_hat.argmax(dim=1) == y).sum().cpu().item() n += y.shape[0]  batch\_count += 1  test\_acc = evaluate\_accuracy (test\_iter, net)  print(**'epoch** **%d,** **loss** **%.4f,** **train** **acc** **%.3f,** **test** **acc** **%.3f,** **time** **%.1f** **sec'**  % (epoch + 1, train l sum / batch\_count, train\_acc\_sum / n, test\_acc, time.time()  - start))  state = {**'net'** :net.state\_dict(), **'optimizer'** :optimizer.state\_dict(), **'epoch'** :epoch}  torch.save(state,**"/home/retoo/Desktop/实验/数据集/3.深度学习** **/9\_2model/LeNet\_epoch\_{}.pth"**.format(epoch)) |   **（3）训练模型**  调用上面定义的训练函数对模型进行训练。（注：模型训练，需要二十分钟左右。）  ***\*参考代码: \****   |  | | --- | | lr, num\_epochs = 0.001, 5  optimizer = torch.optim.Adam(net.parameters(), lr=lr)  train\_ch5(net, train\_iter, test\_iter, batch\_size, optimizer, device, num\_epochs) |   通过 5 个 epoch 后的模型训练结果展示如下:    由于数据读取的过程中具有随机性，因此实验结果具有一定的波动，建议可以多试验几次.  **8.模型预测**  完成模型的训练后，需要查看模型的泛化性能，在目录home/retoo/Desktop/实验/数据集 /3.深度学习/9\_ 1DataSets 下含有数字 5 ，7 ，9 等图片，任意选择一张对其进行预测，看模型 结果是否正确。（注：模型预测，需等待二十分钟左右）  （1）读取测试图片，并显示  ***\*参考代码: \****   |  | | --- | | **import** cv2  **import** matplotlib.pyplot **as** plt  image = plt.imread(**"/home/retoo/Desktop/实验/数据集/3.深度学习/9\_1DataSets/test\_img/5.4.jpg"**) plt.imshow(image) |  1. **实验结果**      1. **实验心得**   在实验中，我首先对LeNet网络有了全面的理论认识。LeNet是Yann LeCun在1998年提出的，是最早用于手写数字识别的卷积神经网络之一。通过学习，我了解到LeNet通过一系列卷积层、池化层和全连接层逐步提取图像特征，最终实现对手写数字的有效分类。实验让我有机会将这些抽象概念转化为实际代码，亲手搭建和训练模型，感受深度学习模型的魅力。  实验的实践部分显著提升了我的技能。从零开始搭建LeNet模型的过程，使我熟悉了PyTorch等深度学习框架的使用，特别是在模型定义、权重初始化、损失函数选择和优化器配置等方面。我还学会了如何调整网络参数，如卷积核大小、步长、填充等，以优化模型性能。通过处理和加载MNIST数据集，我掌握了数据标准化和批量化等重要步骤，理解了它们对模型训练效率和准确性的意义。我还学会了如何有效地利用数据增强来提高模型的泛化能力。  此外，实验中我通过实施交叉验证、计算准确率和分析混淆矩阵，学会了如何客观评价模型性能。同时，使用TensorBoard等工具对训练过程进行可视化，使我直观地看到了损失和准确率随时间变化的趋势，对模型学习动态有了更深的理解。这次实验不仅是对LeNet模型的一次复现，更是对深度学习流程的一次全面实践。面对模型训练初期的低准确率，我通过不断调整超参数、优化模型结构，最终使模型在测试集上达到满意效果，这一过程充满了挑战与成就感。 |

实 验 报 告

第 13 周

|  |  |  |  |
| --- | --- | --- | --- |
| 实 验 名 称 | 实验三：基于 RNN 歌曲自动编曲设计 | | |
| 实验性质 | 必修 | 实验类型 | 设计型 |
| 实验课时 | 2 | 实验日期、时间 | 2024年6月1日 |
| 实验消耗器材 | 无 | | |
| 实验仪器设备 | 硬件环境 JetsonNX  操作系统 Linux  实验设备 机械臂，Jetson NX  实验配件 教具 | | |
| 实验目的 | （1）掌握深度学习中自然语言处理数据编码方法；  （2）了解自然语音模型；  （3）掌握 RNN 模型基本原理与应用；  （4）掌握基于 Pytorch 的 RNN 模型训练与测试方法。 | | |
| 实验内容（实验原理、运用的理论知识和数据、算法、程序、步骤和方法） | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **实验基本原理**   自然语言处理的重要技术是语言模型， 自然语言处理中最常见的数据是文本数据， 通常把自然语言文本看作一段离散的时间序列。假设一段长度为 m 的文本中的词依次 为w1 , *w*2 , ..., *wm* ，那么在时间序列里面就是*wt*（1 < *t* < *m*），可看作时间 t 的输出或者标签。 语言模型对该词序列计算概率为:  *p*(*w*1*,w*2*,...,wm* )  语言模型可以提升语音识别和机器翻译的性能，基于语言模型的词序列的概率只考 虑临近词的概率，若需要考虑到更早时间的词的概率，那么模型的参数将会指数级增长。 但是循环神经网络可以通过隐藏层来存储时间上的信息，用非刚性的方法记忆不同长度 的序列。  **1.语言模型的计算**  语言模型对词序列的概率计算如下，假设序列中每个词都是单独生成的，那么有:  *p*(*w*1*,w*2*,...,wm* ) =  例如一个含有四个文本的概率为:  *p*(*w*1*,w*2*,w*3*,w*4 ) = *p*(*w*1 )*p*(*w*2 *| w*1 )*p*(*w*3 *| w*1*,w*2 )*p*(*w*4 *| w*1*,w*2*,w*3 )  为了计算语言模型，需要计算词的概率，以及一个词在给定前几个词的情况下的条 件概率，即语言模型参数。设训练数据集为一个大型文本语言词库，如维基百科的所有  条目。词的概率可以通过该词在训练数据集中的相对词频来计算。例如，*p*(*w*1 ) 可以计 算为*w*1 在训练数据集中的词频（词出现的次数）与训练数据集的总词数之比。因此， 根据条件概率定义，一个词在给定前几个词的情况下的条件概率也可以通过训练数据集  中的相对词频计算。例如，*p(w*2 *| w*1 *)* 可以计算为*w*1*,w*2 两词相邻的频率与*w*1 词频的比值， 因为该比值即*p(w*1*,w*2 *)* 与*p(w*1 *)* 之比；而*p(w*3 *| w*1*,w*2 *)* 同理可以计算为*w*1*,w*2 和*w*3 三词相 邻的频率与*w*1 和*w*2 两词相邻的频率的比值。以此类推。  **2.循环神经网络**  因为文本数据存在时间相关性，因此文本数据集*Xt* 是在一个时间步 t 的小批量输 入，设*Ht* 是该时间步的隐藏变量，在循环神经网络中，与卷积神经网络不同的是，这  里需要保存上一时间步的隐藏变量*Ht* -1 。并且需要一个新的隐藏权重*Whh* ，该权重的主 要功能是对上一时间步的隐藏变量的筛选，当前时间步隐藏变量由上一时间步和当前时 间步的输入有关。  *Ht* = *φ (Xt Wxh* + *Ht* -1*Whh* + *bh )*  这里的*Ht -*1*Whh* 可以知道已经存储上一时间步的历史信息，该隐藏变量又称为隐藏状 态，由于隐藏状态在当前时间步的定义包含了上一时间步的隐藏状态，上式的计算是循 环的。本章中的循环神经网络均基于上式中隐藏状态的循环计算。在时间步 t ，输出层 的输出和多层感知机中的计算类似：  *Ot* = *HtWhq* + *bq*  **3.one-hot 向量**  为了将词表示成向量输入进神经网络，一个最简单的办法就是 one-hot 向量。假设 数据集中文本的数量是 N，其中每个字符在文本中都与一个 0 到 N-1 的连续整数值索引 一一对应，那么任意字符在该集中都有唯一的一个索引对应。  **4.数据集**  循环神经网络的数据集不同于卷积神经网络的图片形式，它是一个文本数据，里面 包含各种字符，这里使用公开数据集-周杰伦的专辑歌词，从第一张专辑《jay》到第十 章专辑《跨时代》的所有歌词。  **5.数据采样**  针对文本数据的采样有随机采样和相邻采样，由于文本具有一定的连续性，因此这 里仅介绍相邻采样，有兴趣的同学可自学了解随机采样。用一个小批量最终时间步的隐 藏状态来初始化下一个小批量的隐藏状态，从而使下一个小批量的输出也取决于当前小 批量的输入，并如此循环下去。这对实现循环神经网络造成了两方面影响:一方面， 在 训练模型时，只需在每一个迭代周期开始时初始化隐藏状态；另一方面，当多个相邻小 批量通过传递隐藏状态串联起来时，模型参数的梯度计算将依赖所有串联起来的小批量 序列。同一迭代周期中，随着迭代次数的增加，梯度的计算开销会越来越大。 为了使 模型参数的梯度计算只依赖一次迭代读取的小批量序列，可以在每次读取小批量前将隐 藏状态从计算图中分离出来。  **6.梯度裁剪**  循环神经网络很容易出现梯度爆炸，为了应对梯度爆炸，可以裁剪梯度。假设把所 有模型参数梯度的元素拼接成一个向量*g* ，并设裁剪的阈值是*θ*, 则裁剪后的梯度为：  min*(*1*)g*  **7.交叉熵损失函数**  损失函数在模型训练中用来衡量真实值和预测值之间的差距，数值越小代表真实值 和预测值越接近，交叉熵常用来表示衡量两个不同概率分布的差异程度，在深度学习中 就是真实概率分布和预测概率分布之间的差异，在循环神经网络中，同样的可以使用交 叉熵损失来评估真实值和预测值之间的差距：     1. **实验内容**   本循环神经网络主要用来处理时序性数据，由于其数据结构特殊性，因此网络结构的 设计与卷积网络有所不同，循环网络的数据具有时序性特征，上一时刻对下时刻的行为 有所影响，因此循环网络需要有中间隐藏单元进行时序性存储。  本实验要求在掌握 RNN 网络基本原理和自然语言模型的基础上，利用 RNN 网络 实现自然语言的处理，把“分开 ”与“不分开 ”作为歌词的前缀创造一段长度为 10 的 歌词。   1. **实验步骤**   **1.**运行 **jupyter lab**  （1）打开桌面的“实验 ”文件夹，在空白处单击鼠标右键，再点击“在终端打开 ”， 在终端界面输入“jupyter lab ”；  （2）在jupyter lab 编程界面选择 Notebook 下 Python3 ，进入程序编辑器；  （3）鼠标右键单击“未命名.ipynp ”的新建程序块，选择“Rename ”将程序名命 为实验名称。  **2**首先引入 Torch ，TorchVision，zipfile 等相关库，Torch ，TorchVision 主要用来构建 网络结构，zipfile 则是用来读取压缩文件并解压，此时并不改变文件格式，只是在程序 中进行了解压缩。  ***\*参考代码: \****   |  | | --- | | **import** torch  **import** numpy **as** np *#***import** torchvision  **import** zipfile  **import** torch.nn **as** nn **import** time  **import** math |   **3.加载数据**  通过模块 zipfile 解压歌曲文件 **jaychou\_lyrics.txt.zip** 后建立字符索引，将每个字符 映射成一个从 0 开始的整数，将数据集中的所有不同字符提取出来，然后逐一将其映射 到索引来构建词典。此时数据格式已经初步满足循环神经网络需求。  ***\*参考代码: \****   |  | | --- | | device = torch.device( **'cpu'**)  **def** load\_data\_jay\_lyrics():  """Load the Jay Chou lyric data set (available in the Chinese book)."""  **with** zipfile.ZipFile(**'/home/retoo/Desktop/实验/数据集/3 .深度学习** **/10jay\_song/jaychou\_lyrics.txt.zip'**) **as** zin:  **with** zin.open(**'jaychou\_lyrics.txt'**) **as** f:  corpus\_chars = f.read().decode(**'utf-8'**)  corpus\_chars = corpus\_chars.replace(**'\n'**, **' '**).replace(**'\r'**, **' '**)  corpus\_chars = corpus\_chars[0:10000] idx to char = list(set(corpus\_chars))  char to idx = dict([(char, i) **for** i, char **in** enumerate(idx to char)]) vocab\_size = len(char to idx)  corpus\_indices = [char to idx[char] **for** char **in** corpus\_chars]  **return** corpus\_indices, char\_to\_idx, idx\_to\_char, vocab\_size  (corpus\_indices, char to idx, idx to char, vocab\_size) = load\_data\_jay\_lyrics() |   **4.one-hot 向量**  将数据索引进一步转换为向量，再输入循环神经网络，简单的操作就是将索引进行 one-hot 向量，每个字符都成为一个从 0 到 N-1 的连续整数值与索引一一对应。  ***\*参考代码: \****   |  | | --- | | **def** one\_hot(x, n\_class, dtype=torch.float32):  *# X shape: (batch), output shape: (batch, n\_class)*  x = x.long()  res = torch.zeros(x.shape[0], n\_class, dtype=dtype, device=x.device) res.scatter\_(1, x.view(-1, 1), 1)  **return** res  **def** to\_onehot(X, n\_class):  *# X shape: (batch, seq\_len), output: seq\_len elements of (batch, n\_class)*  **return** [one\_hot(X[:, i], n\_class) **for** i **in** range(X.shape[1])] |   **5.定义相邻采样**  相邻采样就是将原始数据集相邻的两个小批次在原始序列上的位置相毗邻，可以用 一个时间步的隐藏状态来初始化下一个小批量的隐藏状态，从而使得下一个批次的输出 也取决于上一个批次的输入。  ***\*参考代码: \****   |  | | --- | | **def** data\_iter\_consecutive(corpus\_indices, batch\_size, num\_steps, device=**None**): **if** device **is None**:  device = torch.device(**'cuda' if** torch.cuda.is\_available() **else 'cpu'**)  corpus\_indices = torch.tensor(corpus\_indices, dtype=torch.float32, device=device) data\_len = len(corpus\_indices)  batch\_len = data\_len // batch\_size  indices = corpus\_indices[0: batch\_size\*batch\_len].view(batch\_size, batch\_len) epoch\_size = (batch\_len - 1) // num\_steps  **for** i **in** range(epoch\_size): i = i \* num\_steps  X = indices[:, i: i + num\_steps]  Y = indices[:, i + 1: i + num\_steps + 1] **yield** X, Y |   **6.定义优化器**  这里仍然使用自定义的优化器，与 Torch 框架里的 nn 模块的实现功能一致。  ***\*参考代码: \****   |  | | --- | | **def** sgd(params, lr, batch\_size): |  |  | | --- | | """Mini-batch stochastic gradient descent."""  **for** param **in** params:  param[:] = param - lr \* param.grad / batch\_size |   **7.定义模型**  根据先前的图像分类思路，这里建立一个完整的循环神经网络模型，定义了在一个 时间步里如何计算隐藏状态和输出，同时使用全连接输出层得到输出。输出的大小等于 词典的大小。  ***\*参考代码: \****   |  | | --- | | num\_hiddens = 32  rnn\_layer = nn.RNN(input\_size=vocab\_size, hidden\_size=num\_hiddens)  **class** RNNModel(nn.Module):  **def** init (self, rnn\_layer, vocab\_size):  super(RNNModel, self). init () self.rnn = rnn\_layer  self.hidden\_size = rnn\_layer.hidden\_size \* (2 **if** rnn\_layer.bidirectional **else** 1) self.vocab\_size = vocab\_size  self.dense = nn.Linear(self.hidden\_size, vocab\_size) self.state = **None**  **def** forward(self, inputs, state): # inputs: (batch, seq\_len) # 获取 one-hot 向量表示  X = to\_onehot(inputs, self.vocab\_size) # X 是个 list Y, self.state = self.rnn(torch.stack(X), state)  # 全连接层会首先将 Y 的形状变成(num\_steps \* batch\_size, num\_hiddens) ，它的输出 # 形状为(num\_steps \* batch\_size, vocab\_size)  output = self.dense(Y.view(-1, Y.shape[-1])) **return** output, self.state |   **8.定义预测函数**  ***\*参考代码: \****    QQ截图20240601144717  **9.裁剪梯度**  由于循环神经网络的特殊性，时间步的大小不同导致电脑运行内存不同，且参数量 巨大，极易发生梯度爆炸，因此为了应对模型出现梯度爆炸的问题，采用梯度裁剪策略， 超出一定阈值的参数进行裁断。使梯度爆炸问题得到缓解。  ***\*参考代码: \****   |  | | --- | | **def** grad\_clipping(params, theta, device):  norm = torch.tensor([0.0], device=device) **for** param **in** params:  norm += (param.grad.data \*\* 2).sum() norm = norm.sqrt().item()  **if** norm > theta:  **for** param **in** params:  param.grad.data \*= (theta / norm) |   **10.定义模型训练函数**  根据数据的采样方式不同，训练模式也不相同，隐藏状态的初始化也不同，这里采 用相邻采样的模式。  ***\*参考代码: \****    QQ截图2  **11.采用相邻采样训练模型**  首先，设置模型超参数。将根据前缀“分开”和“不分开”分别创作长度为 10 个字符 （不考虑前缀长度）的一段歌词。每过 35 个迭代周期便根据当前训练的模型创作一段 歌词。  ***\*参考代码: \****  截图2   1. **实验结果**   **截图20240601145236**   1. **实验心得**   在实验过程中，我掌握了如何使用独热编码（One-Hot Encoding）将音乐序列转换为向量形式，为RNN模型提供必要的数据准备。设计RNN模型时，我理解了隐藏状态的保留机制，并学会了根据任务需求调整模型结构和参数，例如隐藏层的大小和层数。在训练过程中，我学会了监控损失函数和准确率的变化，以判断模型是否过拟合或欠拟合，并据此调整学习率和批量大小等超参数。通过实践“相邻采样”策略，我体会到不同采样方式对模型训练效果的影响，并学会根据训练进展调整策略。  每经过35个迭代周期，我会根据当前训练的模型创作一段新歌词，这让我直接感受到模型的学习成果，并能通过创作的音乐片段直观评估模型的生成质量。这不仅是技术上的挑战，也是对创意和审美的一次考验。整个实验富有创造性，不仅让我见证了深度学习技术在艺术创作中的魅力，也让我认识到理论知识与实际应用之间的差距。通过不断调试和优化，直到模型能够生成具有连贯性和创意的音乐片段，我深刻体会到深度学习项目的复杂性和乐趣。 |