《深度学习及其应用》课程实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 使用CNN实现手写数字识别实验 | | | 实验序号 | 2 | 实验日期 | 20231018 |
| 姓 名 | **赵诗文** | **院系** | 计算机学院 | 班 级 | 21104502 | 学 号 | 2110450209 |
| 专 业 | **人工智能** | | | 指导教师 | 林涛 | 成 绩 | 合格 |
| **一、实验目的和要求**  目的：了解简单的CNN网络的工作原理，并实现LeNet5模型，实现手写数字识别功能。使用MNIST数据集  要求：  1、熟悉实验平台的使用方法。  2、基于人工智能实验平台完成实验，输入不同的手写数字图片，利用神经网络进行识别，读懂算法代码内容；  3、在实验平台上修改输入图片，查看识别结果是否正确，如果有问题，请对代码进行修改；  4、完成实验报告内容，提交报告。 | | | | | | | |
| 二、实验步骤  可以参考实验平台上的视频：  用账号登录到实训平台，选择课程《深度学习及其应用》  在教学平台上学习实验视频，地址：10.2.253.234  也可以在实验平台上完成代码，实验平台地址：10.2.253.243:10010。建议大家手动完成代码，最后将代码在实验平台上运行。     1. 导入PYTorch模型   import torch  实验过程截图放到实验报告中。  实验数据在学习平台上的“扩展”按钮下可以下载。  2．导入PYTorch中的实验数据包  PyTorch提供了一个名为torchvision的库，其中包含了许多常用的计算机视觉数据集，包括MNIST。我们可以使用torchvision中的datasets模块来下载MNIST数据集。也可以使用老师提供的本地mnist数据集，实验平台上也有数据集可以使用，在公共数据集中。数据集中图像大小是28\*28  import torchvision  PYTorch还可以使用transforms模块对数据集进行的转换。可以把数据集转换为PyTorch张量  import torchvision.transforms as transforms  3、利用PYTorch构建LeNet5的网络模型  from torch import nn  class my\_LeNet5(nn.Module):  def \_\_init\_\_(self):  super(my\_LeNet5, self).\_\_init\_\_()  self.conv1 = nn.Conv2d(1, 6, 5,2)  #第一个参数是输入的通道数，第二个参数为输出通道数，第三个参数为卷积核大小（h\*w），第四个参数为padding位大小  self.sigmoid = nn.Sigmoid()  self.s2 = nn.AvgPool2d(kernel\_seze=2,stride=2)  …………  self.flatten = nn.flatern()  self.f6 = nn.Linear(120, 84)  self.output = nn.Linear(84, 10)  前向传播代码  def forward(self, x):  x = self.Sigmoid(self.c1(x))  x = self.s2(x)  x = self.Sigmoid(self.c3(x))  x = self.s4(x)  x = self.c5(x)  x = self.flatten(x)  x = self.f6(x)  x = self.output(x)  return x  输入数字8的图片，利用训练好的前馈神经网络进行识别，如果代码有问题，请对代码进行修改。  4、训练网络  5、测试网络  输入一个测试集中的数字图像，显示出其结果  **附录：**  如何安装PYTorch  可以通过在线安装  也可以将安装包下载到本地装，清华镜像下载地址为：  <https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple/torch/>  可以选择自己需要的版本安装  pip install torch-1.8.1-cp38-cp38-win\_amd64.whl  pip install torchvision-0.9.1-cp38-cp38-win\_amd64.whl  注意torch和torchvision的版本，版本不配合可能会出现错误，torch1.8.1和torchvision0.9.1是一种配合版本 | | | | | | | |
| **三、实验结果与分析**  （截图和代码放到这里）  1 工程前期准备  1.1 导入数据集    2 数据预处理      3 模型构建  3.1 卷积神经网络算法模型搭建并训练      3.2 查看日志    添加测试集并查看预测结果      3.3 修改源码      3.4 查看日志    3.5 添加测试集并查看预测结果    用LeNet5模型实现手写数字识别  import tensorflow as tf  from tensorflow.keras import layers, models  from tensorflow.keras.datasets import mnist  from tensorflow.keras.utils import to\_categorical  # 加载MNIST数据集  (train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()  # 数据预处理  train\_images = train\_images.reshape((60000, 28, 28, 1))  train\_images = train\_images.astype('float32') / 255  test\_images = test\_images.reshape((10000, 28, 28, 1))  test\_images = test\_images.astype('float32') / 255  train\_labels = to\_categorical(train\_labels)  test\_labels = to\_categorical(test\_labels)  # 定义LeNet5模型  def LeNet5():  model = models.Sequential()  model.add(layers.Conv2D(6, (5, 5), activation='tanh', input\_shape=(28, 28, 1)))  model.add(layers.MaxPooling2D((2, 2)))  model.add(layers.Conv2D(16, (5, 5), activation='tanh'))  model.add(layers.MaxPooling2D((2, 2)))  model.add(layers.Flatten())  model.add(layers.Dense(120, activation='tanh'))  model.add(layers.Dense(84, activation='tanh'))  model.add(layers.Dense(10, activation='softmax'))  return model  # 创建LeNet5模型实例  model = LeNet5()  # 编译模型  model.compile(optimizer = tf.keras.optimizers.Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])  # 训练模型  model.fit(train\_images, train\_labels, epochs=20, batch\_size=128)  # 评估模型  test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)  print('Test accuracy:', test\_acc)  import numpy as np  from PIL import Image  def preprocess\_image(image\_path):  img = Image.open(image\_path).convert('L') # 转换为灰度图像  img = img.resize((28, 28)) # 调整尺寸为28x28像素  img\_array = np.array(img) / 255.0 # 归一化到0-1之间  img\_array = np.expand\_dims(img\_array, axis=0) # 增加一个维度以匹配模型输入  return img\_array  new\_image\_array = preprocess\_image(r"C:\Users\Huawei\Desktop\predict\_image（8）.png")  # 使用模型进行预测  predictions = model.predict(new\_image\_array)  # 解析预测结果  def parse\_predictions(predictions):  predicted\_class = np.argmax(predictions)  return predicted\_class  predicted\_class = parse\_predictions(predictions)  print("预测的类别标签：", predicted\_class) | | | | | | | |