第8章 计算机视觉处理

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 人工智能基础与应用 | | 章名 | | | 计算机视觉处理 | |
| 教学内容 | 计算机视觉处理1 | | | | 课时 | 2 | |
| 项目性质 | □演示性 □验证性 □设计性 √综合性 | | | | | | |
| 授课班级 |  | 授课日期 | |  | | 授课地点 |  |
| 教学目标 | 了解计算机视觉开发  掌握手写数字识别项目的实现 | | | | | | |
| 教学内容 | 1. 计算机视觉开发介绍 2. 手写数字识别 | | | | | | |
| 教学重点 | 使用TensorFlow设计手写数字识别项目 | | | | | | |
| 教学难点 | 使用TensorFlow设计手写数字识别项目 | | | | | | |
| 教学准备 | 装有Python的计算机  教学课件PPT  教材：《人工智能基础与应用（微课版）》 | | | | | | |
| 作业设计 |  | | | | | | |

教学过程

|  |  |
| --- | --- |
| **教学环节** | **教学内容与过程**  **（教学内容、教学方法、组织形式、教学手段）** |
| **课前组织** | 做好上课前的各项准备工作（打开计算机、打开课件、打开软件、打开授课计划、教案等），吸引学生注意力。 |
| **课程说明** | 【课前说明】  从计算机视觉比较突出的应用领域引入本节课学习内容。  【目的】  使学生从了解本节课的学习目标、学习重点、考评方式等方面明确课程学习的要求和目标。 |
| **课程内容描述** | 8.1 计算机视觉开发介绍  计算机视觉是一个跨学科的领域，涉及的部分学科如图所示。  8-1  计算机视觉涉及的部分学科  在20世纪60年代后期，一些涉及了人工智能相关方向的大学开始研究计算机视觉，它旨在模仿人类的视觉系统，开始的时候，开发者希望利用计算机视觉从图像中提取三维结构，以实现对整个场景的理解。20世纪70年代的研究包括从图像中提取边缘、标记线条、进行非多面体和多面体建模、将对象表示为较小结构的互连、光流以及运动估计等，为当今的计算机视觉奠定了基础。  计算机视觉是深度学习最先取得突破性成就的领域。2012年，在ILSVRC大赛上，基于卷积神经网络的AlexNet模型获得了当年图像分类的冠军。历年ILSVRC比赛冠军模型错误率如图8-2所示。从下图中可以看出，在2012年以前，传统的视觉处理方法错误率最低的为2011年的25.80%。在2012年，将深度学习引入计算机视觉后，错误率降到了16.40%。从2013年开始，比赛中的前20名都使用了深度学习算法。2013年之后，ILSVRC比赛就基本上只有深度学习算法参赛了。2012—2016年，通过对算法的研究以及优化，识别错误率在不断地下降，这让图像分类问题得到了很好的解决。在2015年，当年的冠军模型ResNet将错误率下降到了3.60%，要低于人工标注的错误率5.1%，实现了计算机视觉上的突破。  8-2  历年ILSVRC比赛冠军模型错误率  神经网络和深度学习极大地推动了计算机视觉的发展，发展较好的几个方向如下。  1．图像分类  在图像分类问题中，图像上只有单一类别，将很多带有标记的数据集进行训练之后，可以对新的、未知的、具有单一类别的图像进行预测，类似于教小孩子看图识物，这种方法是数据驱动的方法，也是图像分类最常用的方法。例如，对猫的类别进行训练后，再将下图所示的照片输入网络进行预测，在网络训练效果不错的前提下，可以识别出这是一只猫。    猫的照片  2．目标检测  与图像分类不同，进行目标检测的图像中并不一定只有单一类别的物体。在处理这类问题时，需要在数据上针对各个对象画出边界框和标签，训练完成后可以对新的图像进行预测，目标检测如图所示，方框可以圈出猫的位置。    目标检测  3．语义分割  语义分割与目标检测不同，语义分割需要对每个像素进行语义上的理解，由于需要对每个像素属于图像上的哪个部分做出分类，所以每个像素都拥有标签，语义分割如图所示。    语义分割  计算机视觉比较突出的应用领域如下。医学图像检验：从图像数据中提取信息以诊断患者患病类别；工业领域：在该领域，计算机视觉有时被称为机器视觉，如产品质量把控，机器视觉也大量运用于农业上，以去除不良幼苗或除虫；安防、娱乐领域：传统机器学习的方法运用于人脸识别时并不能很好地满足精度要求，并且同一个人在不同光照、姿态下的特征会有差异，在深度学习运用于计算机视觉后，算法能够提升识别准确率；光学字符识别：将计算机无法理解的图像形式转换成计算机可以理解的文本格式；自动驾驶：可以在马路上无人驾驶汽车，还可以进行自动泊车等操作。  8.2 手写数字识别  基于MNIST数据集的手写数字识别是学习深度学习的一个入门级例子，本节将使用TensorFlow设计一个手写数字识别项目。 8.2.1 项目介绍 本项目采用卷积神经网络，为了保证整个项目的完整性，在训练过程中不仅要显示损失或者准确率，而且在训练完成后需要保存得到的模型，然后调用摄像头来实时预测新的图像，新图像可以是数据集中的，也可以是自己手写的。通过实现整个过程，将OpenCV、神经网络以及TensorFlow结合起来学习，项目流程图如图所示。  8-6  项目流程图  在5.4节中，通过TensorFlow框架实现了一个类似于LeNet-5的神经网络，来解决MNIST数据集上的手写数字识别问题，本节训练过程依然使用该网络，并且在最后训练出模型，模型文件以变量的形式存储参数，该变量需要在代码中初始化。在训练过程中，将更新的参数存储到变量中，使用tf.train.Saver()对象将所有的变量添加到Graph中。  保存模型的函数为：  save\_path = saver.save(sess, model\_path)  如果每隔一定的迭代步数就保存一次模型，就把迭代步数作为参数传进去：  save\_path = saver.save(sess, model\_path, global\_step=step,write\_meta\_graph=False)  在模型保存之后，调用该模型可以完成新数据的分类预测，模型在保存后会生成4个文件，TensorFlow模型如图所示。    TensorFlow模型  其中，model.meta是训练过程中保存的元数据；model.data-00000-of-00001和model.index是检查点文件，存储着训练过程中保存的模型；checkpoint是记录文件，保存最新检查点文件的记录。 8.2.2 图像获取以及预处理 本项目中使用两种方式获取图像：直接读取图像和调用摄像头从视频流中获取图像。新建CV目录，在CV目录下新建mnist\_predict项目目录，将5.4.1小节训练完成的模型目录放在mnist\_ predict目录下。  1．从图像文件中读取并处理  【例8-1】 在mnist\_predict目录下新建文件，命名为read\_pic.py，使用OpenCV读取新图像，并进行预处理，在PyCharm中编写如下代码。  import os  import cv2  os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  # 将输入的彩色图像转换为二值化图  def color\_input(endimg):  # 灰度化转换  img\_gray = cv2.cvtColor(endimg, cv2.COLOR\_BGR2GRAY)  ret, img\_threshold = cv2.threshold(img\_gray, 127, 255, cv2.THRESH\_BINARY\_INV)  return img\_threshold  # 读取图像并显示  def read\_pic(path):  img = cv2.imread(path, cv2.IMREAD\_COLOR)  cv2.imshow('img', img)  cv2.waitKey(0)  img\_threshold = color\_input(img)  cv2.imshow('img\_threshold', img\_threshold)  cv2.waitKey(0)  if \_\_name\_\_ == '\_\_main\_\_':  read\_pic("pic.png")  运行以上代码，在read\_pic()函数中可读取与read\_pic.py同级目录下的pic.png图像文件并且显示出来。pic.png图像文件如图8-8所示，这是手写的一些数字，关掉显示框之后，调用color\_input()函数，并将读取的图像传递进去。    pic.png图像文件  在color\_input()函数中，完成RGB彩色图像向二值化图像的转换，并将转换后的二值化图像显示出来，转换后的二值化图像如图所示。    转换后的二值化图像  2．从摄像头获取图像  从摄像头获取图像并转换为二值化图，其基本操作与读取图像类似，但需要使用OpenCV调用摄像头。  【例8-2】 在mnist\_predict目录下新建文件，命名为camera.py，使用摄像头拍摄图像，处理为二值化图并显示，在PyCharm中编写以下代码。  import cv2  def start():  # 使用摄像头  cap = cv2.VideoCapture(0)  while (True):  # 读取一帧的图像  ret, frame = cap.read()  # 灰度化  img\_gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  ret, img\_threshold = cv2.threshold(img\_gray, 127, 255, cv2.THRESH\_BINARY\_INV)  cv2.imshow('img\_threshold', img\_threshold)  key = cv2.waitKey(30) & 0xff  if key == 27:  sys.exit(0)  # 释放摄像头  cap.release()  cv2.destroyAllWindows()  if \_\_name\_\_ == '\_\_main\_\_':  start()  start()函数可调用摄像头，捕捉并显示视频帧。 8.2.3 图像识别 使用训练好的模型识别新图像，根据图像来源不同，介绍两种不同的方式完成识别。  1．从图像文件中读取并识别  在识别之前，首先需要恢复保存的模型。在恢复模型之前，无需初始化变量，在恢复过程中会自动进行初始化保存的变量的操作。8.2.2小节实现了图像的读取和二值化的转换，输入预测代码后进行运算，完成识别。  【例8-3】 在mnist\_predict目录下新建文件，命名为predict\_pic.py，识别图像。  import os  import cv2  import numpy as np  import tensorflow as tf  os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  # 将输入的彩色图像转换为二值化图  def color\_input(endimg):  img\_gray = cv2.cvtColor(endimg, cv2.COLOR\_BGR2GRAY) # 灰度化  ret, img\_threshold = cv2.threshold(img\_gray, 127, 255, cv2.THRESH\_BINARY\_INV)  return img\_threshold  # 读取图像并显示  def read\_pic(path):  img = cv2.imread(path, cv2.IMREAD\_COLOR)  cv2.imshow('img', img)  cv2.waitKey(0)  img\_threshold = color\_input(img)  cv2.imshow('img\_threshold', img\_threshold)  cv2.waitKey(0)  return img\_threshold  if \_\_name\_\_ == '\_\_main\_\_':  with tf.Session() as sess:  saver = tf.train.import\_meta\_graph('model\_data/model.meta')  # 模型恢复  saver.restore(sess, 'model\_data/model')  graph = tf.get\_default\_graph()  # 获取变量  input\_x = sess.graph.get\_tensor\_by\_name("Mul:0")  y\_conv2 = sess.graph.get\_tensor\_by\_name("final\_result:0")  # 读取图像  img\_threshold = read\_pic("nine.png")  # 将图像进行缩放  im = cv2.resize(img\_threshold, (28, 28), interpolation=cv2.INTER\_CUBIC)  x\_img = np.reshape(im, [-1, 784])  # 识别  output = sess.run(y\_conv2, feed\_dict={input\_x: x\_img})  result = np.argmax(output)  print("识别结果为：{}".format(result))  使用saver.restore(sess,model\_path)函数实现模型恢复，根据标识符获得Tensor变量内容，将要识别的图像进行读取并进行缩放，读取到的图像如图所示。缩放的目的是为了保持处理后的图像与训练时的数据集图像大小一致，否则无法识别。    图8-10 读取到的图像  识别完成后，得到识别结果，如图所示。    识别结果（1）  2．从摄像头实时识别  从摄像头捕捉图像进行识别的基本过程与直接读取图像一致，通过实时图像的界面，可以将视频帧数、结果都显示出来。  【例8-4】 在目录下新建文件，命名为predict\_camera.py，完成识别。  首先导入需要的类，包括OpenCV、NumPy和TensorFlow。  import os  import cv2  import sys  import time  import numpy as np  import tensorflow as tf  os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'  接下来需要封装一个函数，作用是将输入的RGB图像转换为二值化图像，并将转换后的二值化图像返回。  # 将输入的彩色图像转换为二值化图  def color\_input(endimg):  img\_gray = cv2.cvtColor(endimg, cv2.COLOR\_BGR2GRAY)  ret, img\_threshold = cv2.threshold(img\_gray, 127, 255, cv2.THRESH\_BINARY\_INV)  return img\_threshold  然后恢复模型。实例化一个saver，并使用saver.restore()函数恢复模型，将得到的变量返回。  # 恢复模型并实例化saver  def restore\_model():  sess = tf.Session()  saver = tf.train.import\_meta\_graph('model\_data/model.meta')  # 使用saver.restore()函数模型恢复  saver.restore(sess, 'model\_data/model')  # 获取变量  input\_x = sess.graph.get\_tensor\_by\_name("Mul:0")  y\_conv2 = sess.graph.get\_tensor\_by\_name("final\_result:0")  return sess, input\_x, y\_conv2  接下来构建预测函数，将变量和二值化图像传入，将图像进行缩放，调用sess.run()函数实现预测，并将结果返回。  # 图像预测  def mnist\_predict(sess, input\_x, y\_conv2, img\_thre):  # 将图像进行缩放  im = cv2.resize(img\_thre, (28, 28), interpolation=cv2.INTER\_CUBIC)  x\_img = np.reshape(im, [-1, 784])  # 识别  output = sess.run(y\_conv2, feed\_dict={input\_x: x\_img})  result = np.argmax(output)  return result  最后在主函数中调用摄像头，调用模型恢复函数，使用cv2.putText()函数在显示的界面上显示识别结果、帧数等提示。  if \_\_name\_\_ == '\_\_main\_\_':  # 使用默认字体  font = cv2.FONT\_HERSHEY\_SIMPLEX  # 使用摄像头  cap = cv2.VideoCapture(0)  # 初始化用于计算fps的变量  fps = "FPS: ??"  start\_time = time.time()  counter = 0  # 调用模型恢复函数  sess, input\_x, y\_conv2 = restore\_model()  # 循环显示识别结果图像  while (True):  # 读取一帧的图像  ret, frame = cap.read()  cv2.rectangle(frame, (180, 100), (460, 380), (0, 255, 0), 2)  frame = cv2.putText(frame, 'Please', (0, 40), font, 1.2, (0, 255, 255), 2)  frame = cv2.putText(frame, 'Put the number in the box:', (0, 80), font, 1.2, (0, 255, 255), 2)  endimg = frame[100: 380, 180: 460]  endimg\_threshold = color\_input(endimg)  result = mnist\_predict(sess, input\_x, y\_conv2, endimg\_threshold)  counter += 1  if(time.time() - start\_time) > 1:  print("FPS: ", counter / (time.time() - start\_time))  counter = 0  start\_time = time.time()  cv2.putText(frame, "%d" % result, (460, 380), font, 3, (0, 0, 255), 2)  cv2.putText(frame, fps, (50, 120), font, 0.8, (0, 0, 255), 2)  cv2.imshow('Number Recognition', frame)  key = cv2.waitKey(30) & 0xff  if key == 27:  sys.exit(0)  sess.close()  # 释放摄像头  cap.release()  cv2.destroyAllWindows() 8.2.4 结果显示 使用代码画一个方形框，可以使结果更加直观。将摄像头对准手写的数字，将数字放在框里，调用识别函数完成识别，并将帧数FPS显示在方形框左上角，将结果显示在方形框右下角，识别结果如图所示。    识别结果（2） |
| **总结评价** | 本节课主要介绍了计算机视觉开发的几个方向，并以一个项目实例讲解了TensorFlow在手写数字数据集MNIST的图像识别领域中的应用。 |