人工智能实践课程报告

基于深度学习的手势数字识别设计

学　　生：鲜阳

学　　号：320085404113

专　　业：电子信息

班　　级：控制工程 2020.1

指导教师：谭飞

四川轻化工大学自动化与信息工程学院

2022 年 02 月

基于深度学习的手势数字识别设计

摘要：手势识别是在计算机视觉领域中受到广泛关注的问题，早在深度神经网络崛起前，研究者和程序员们就对该问题进行了一定的探索，但传统的算法比较复杂，而且随着针对目标的变化，算法也会随之改变。近年来，深度神经网络得到广泛应用，在计算机视觉领域更是如此，其特点是不需要人工设计算法进行特征提取，且对不同种类的手势有普适性。因此，本项目基于 python 的 TensorFlow，训练了一个卷积神经网络进行手势数字的识别。

关键词：手势数字识别，神经网络，python

Design of Gesture Digit Recognition Based on Deep Learning

ABSTRACT： Gesture recognition is a problem that has received extensive attention in the field of computer vision. As early as before the rise of deep neural networks, researchers and programmers have explored this problem to a certain extent, but traditional algorithms are more complex, and with the development of target changes, the algorithm will also change. In recent years, deep neural network has been widely used, especially in the field of computer vision. Therefore, based on python's TensorFlow, this project trains a convolutional neural network to recognize gesture digits.

Keywords: Gesture digit recognition, Neural network, Python

目录

第一章 绪 论

1.1 研究工作的背景与意义

手势是人类活动中最为简单的一种非语言交流方式，因其简便直观、目的性强等特点，被广泛应用在人机交互、智能家居、增强现实等领域。例如，华为公司于 2019 年发布的华为 mate30 手机可实现“隔空手势握拳”截屏操作；百度 AI 平台推出手势识别 API，可在线识别“拳头”等 24 种常见手势。随着计算机硬件能力的提升以及深度学习的不断发展，基于计算机视觉的手势识别成为当下科学研究的重点领域之一。常见的手势数字分类识别技术有深度卷积神经网络、支持向量机、朴素贝叶斯等方法。

1.2 本项目的结构安排

本文的章节结构安排如下四部分：
方案设计：主要介绍了手势数字识别的各个步骤。
软件实现：介绍了实现所需的电脑环境和程序文件功能。
调试过程：进行了识别结果展示以及代码编写中的一些问题。
经验感受：记录了心得体会和后续工作展望。

第二章 方案设计

2.1 数据录入

通过调用笔记本电脑的摄像头进行数据集的录入。对摄像头获得的图像做轮廓检测和背景消除，并在保存前调整图像大小以便于网络输入。

2.2 CNN 编写

编写了一个简单的网络来实现其功能，使用 9 层的卷积神经网络，卷积核大小为 2，激活层使用 relu 函数，每一个卷积层后面接一层最大池化层，最后设置两层全连接层。

```
x = input_data(shape=[None, 89, 100, 1], name='input')
conv1 = conv_2d(x, 32, 2, activation='relu')
pool1 = max_pool_2d(conv1, 2)
conv2 = conv_2d(pool1, 64, 2, activation='relu')
pool2 = max_pool_2d(conv2, 2)
conv3 = conv_2d(pool2, 128, 2, activation='relu')
pool3 = max_pool_2d(conv3, 2)
conv4 = conv_2d(pool3, 256, 2, activation='relu')
pool4 = max_pool_2d(conv4, 2)
conv5 = conv_2d(pool4, 256, 2, activation='relu')
pool5 = max_pool_2d(conv5, 2)
conv6 = conv_2d(pool5, 128, 2, activation='relu')
pool6 = max_pool_2d(conv6, 2)
conv7 = conv_2d(pool6, 64, 2, activation='relu')
pool7 = max_pool_2d(conv7, 2)
full1 = fully_connected(pool7, 1000, activation='relu')
convnet = dropout(full1, 0.75)
full2 = fully_connected(convnet, 6, activation='softmax')
convnet = regression(full2, optimizer='adam', learning_rate=0.001,
    loss='categorical_crossentropy', name='regression')
model = tflearn.DNN(convnet, tensorboard_verbose=0)
```

2.3 参数设定

数据集：训练集分为 6 部分，每部分各 1000 张；测试集同理但取 100 张。
网络：n_epoch=50，batch_size=64, learing_rate=1e-4, optimizer='adam', loss='categorical_crossentropy'

2.4 训练

将所有训练数据存放到一个名为 loadedImages 的列表中，创建相应的输出向量。同时将测试数据存放到 testImages 列表中，并在 testLabels 中标注对应的标签数字。在训练前打乱训练集次序，运行下面语句进行训练。

```
model.fit(loadedImages, outputVectors, n_epoch=50,
          validation_set=(testImages, testLabels),
          snapshot_step=100, show_metric=True, run_id='convnet_coursera')
```

然后再保存模型:

```
model.save("TrainedModel/GestureRecogModel.tfl")
```

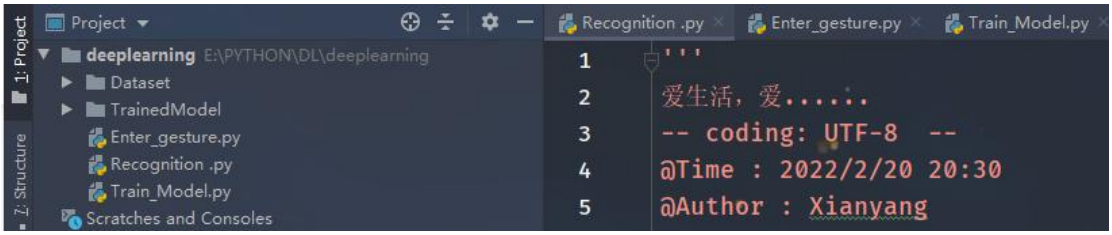
第三章 软件实现

3.1 软件环境

使用 Pycharm 软件平台及相关的 python 第三方包:
Tensorflow
TfLearn
Cv2
Numpy
PIL
Imutils

3.2 文件功能

文件夹如下描述:

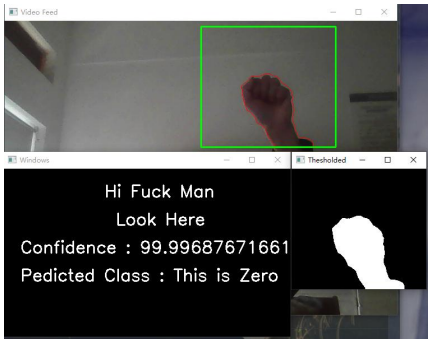


1.Dataset 为自己录入的数据集，包含了 0（拳头），1，2，3，4，5（手掌）的训练数据和测试数据，数量分别为 1000、100，大小为 100*89，png 格式，灰度图。
2.TrainedModel 为训练后网络模型保存的位置。
3.Enter_gesture.py 为数据录入文件, 执行即可将手势录入并保存到 Dataset 中相应的文件夹内，重新录入复杂的手势时需要更改相应的路径。
4. Train_Model.py 执行该文件将读取训练集和数据集，送到网络中训练，并将得到的模型保存到 TrainedModel 中。
5. Recognition.py 执行该文件将打开摄像头获取手势，从 TrainedModel 中加载训练好的模型，并在窗口中显示识别结果。

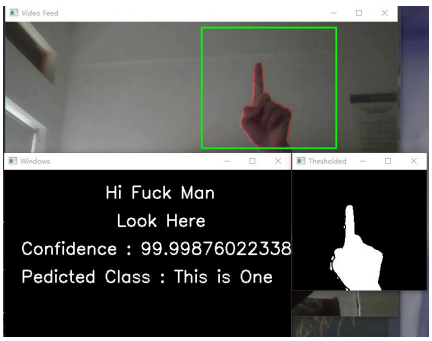
第四章 调试过程

4.1 识别结果

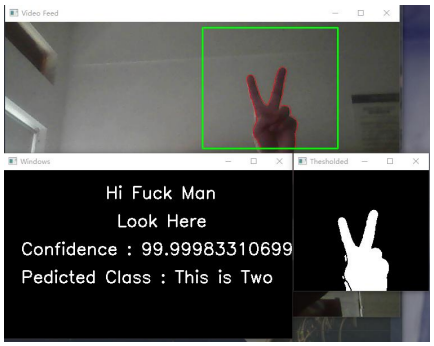
以下展示了手势数字 0-5 的识别结果。



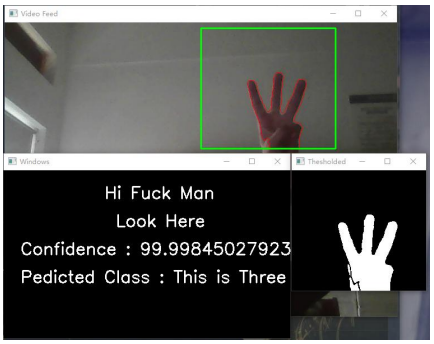
图一



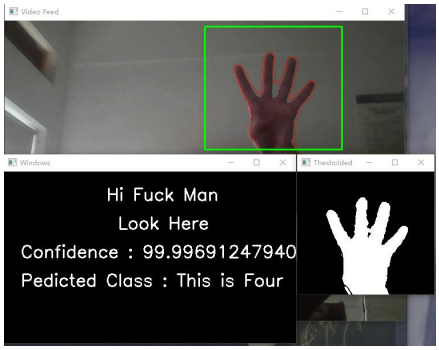
图二



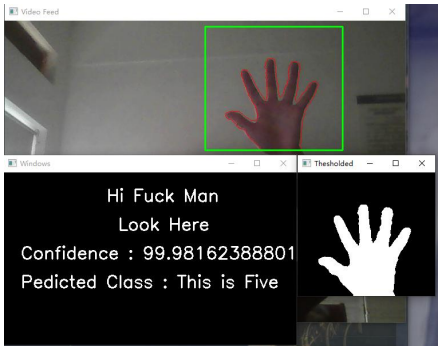
图三



图四



图五



图六

4.2 训练结果

```
# Shuffle Training Data
loadedImages, outputVectors = shuffle(loadedImages, outputVectors, random_state=0)

# Train model
model.fit(loadedImages, outputVectors, n_epoch=50,
          validation_set = (testImages, testLabels),
          snapshot_step=100, show_metric=True, run_id='convnet_coursera')

model.save("TrainedModel/GestureRecogModel.tfl")
```

```
Training Step: 4699  | total loss: [1m[32m0.00073[0m[0m | time: 2.832s
| Adam | epoch: 050 | loss: 0.00073 - acc: 1.0000 -- iter: 5952/6000
Training Step: 4700  | total loss: [1m[32m0.00068[0m[0m | time: 3.867s
| Adam | epoch: 050 | loss: 0.00068 - acc: 1.0000 | val_loss: 0.00011 - val_acc: 1.0000 -- iter: 6000/6000
--
```

4.3 代码重构

为减少代码冗余，经常会从相关文件中进行某些函数的调用，而在 Recognition.py 文件中，是进行实时识别的，除了加载先前训练好的模型文件，还要对摄像头获得的图像进行处理，而 resizeImage、run_avg、segment 起到了该作用。而这三个函数在 Enter_gesture.py 中也有用到，当 from Enter_gesture import resizeImage,run_avg,segment 后，程序并不会报错，但在运行时会出现摄像头窗口闪退一次，再重新打开的 BUG，而后续实时识别摄像头窗口区域会出现阴影且不消失，而影响到识别，这个问题还未解决。

第五章 经验感受

5.1 心得体会

一、使用当前的网络模型进行识别不太标准的手势时，识别率会一定程度的下降，甚至出现错误。解决办法是录入更多数量的数据重新进行训练，尽大可能的包含多种不标准的手势。

二、代码重构需要一定的技术含量，即使是很简单的导入操作，也可能会造成 BUG。

三、基于深度学习的方法训练网络比传统的识别方法简单高效，在计算机视觉领域有很大的优势。

5.2 后续工作展望

1.尽量解决第四章代码重构里出现的 bug。
2.另外争取录入像 ok、点赞、双手组合等较复杂一些的手势作为训练集，丰富识别功能。

附　录

程序和数据如下压缩包: