信息工程学院《计算机视觉》课程

综 合 实 验 报 告



实验题目：花卉识别系统的设计与实现

实验人员：计算机1502班 20151012826许江龙

指导老师：何进荣

2018年5月30日

目录

[花卉识别系统的设计与实现](#_Toc27687_WPSOffice_Level1) [3](#_Toc27687_WPSOffice_Level1)

[1.花卉识别的应用意义](#_Toc22242_WPSOffice_Level1) [3](#_Toc22242_WPSOffice_Level1)

[2.实验数据](#_Toc11143_WPSOffice_Level1) [3](#_Toc11143_WPSOffice_Level1)

[3. 实验原理与基础知识介绍](#_Toc25319_WPSOffice_Level1) [3](#_Toc25319_WPSOffice_Level1)

[4. 实验设置](#_Toc11021_WPSOffice_Level1) [5](#_Toc11021_WPSOffice_Level1)

[5. 方法对比与结果评价](#_Toc7480_WPSOffice_Level1) [6](#_Toc7480_WPSOffice_Level1)

[6. 源代码](#_Toc26259_WPSOffice_Level1) [7](#_Toc26259_WPSOffice_Level1)

[7. 参考资料](#_Toc27616_WPSOffice_Level1) [15](#_Toc27616_WPSOffice_Level1)

# 花卉识别系统的设计与实现

## 1.花卉识别的应用意义

深度学习是人工智能领域近年来最火热的话题之一，但是对于个人来说，以往想要玩转深度学习除了要具备高超的编程技巧，还需要有海量的数据和强劲的硬件。不过 TensorFlow 和 Keras 等框架的出现大大降低了编程的复杂度。

而将深度学习的应用到图像识别上是十分普遍且有趣、有很大前景的一项工作。我做的小项目是基于TensorFlow的CNN进行花卉的种类识别。前景十分广阔。例如在各种园艺节上用该系统快速的识别不同种类的花卉，从而节省时间，极大地提高了工作效率。同时在初高中进行深度学习基本知识的普及过程中也可以用该系统进行演示。演示的过程十分简单、易上手。原理也较清晰，初高中生理解起来也较为容易。所以，该系统有比较广阔的应用意义。

## 2.实验数据

实验数据是五种不同种类的花朵，分别是：菊花、百合、玫瑰、太阳花、蒲公英。这几种花都是十分常见的花朵。我的数据集是在网上下载，下载地址是[http://download.tensorflow.org/example\_images/flower\_photos.tgz](http://download.tensorflow.org/example_images/flower_photos.tgz" \t "_blank)

## 3. 实验原理与基础知识介绍

##### 3.1相关原理介绍

该系统是利用深度学习的卷积神经网络CNN，首先介绍下卷积神经网络CNN，首先介绍下什么是卷积神经网络，卷积神经网络（Convolutional Neural Network,CNN）是一种[前馈神经网络](https://baike.baidu.com/item/%E5%89%8D%E9%A6%88%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C" \t "_blank)，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。  它包括卷积层(convolutional layer)和池化层(pooling layer)。

卷积神经网络是近年发展起来，并引起广泛重视的一种高效识别方法。20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks-简称CNN）。现在，CNN已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用。 K.Fukushima在1980年提出的新识别机是卷积神经网络的第一个实现网络。随后，更多的科研工作者对该网络进行了改进。其中，具有代表性的研究成果是Alexander和Taylor提出的“改进认知机”，该方法综合了各种改进方法的优点并避免了耗时的误差反向传播。

一般地，CNN的基本结构包括两层，其一为特征提取层，每个神经元的输入与前一层的局部接受域相连，并提取该局部的特征。一旦该局部特征被提取后，它与其它特征间的位置关系也随之确定下来；其二是特征映射层，网络的每个计算层由多个特征映射组成，每个特征映射是一个平面，平面上所有神经元的权值相等。特征映射结构采用影响函数核小的sigmoid函数作为卷积网络的激活函数，使得特征映射具有位移不变性。此外，由于一个映射面上的神经元共享权值，因而减少了网络自由参数的个数。卷积神经网络中的每一个卷积层都紧跟着一个用来求局部平均与二次提取的计算层，这种特有的两次特征提取结构减小了特征分辨率。

CNN主要用来识别位移、缩放及其他形式扭曲不变性的二维图形。由于CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。

##### 3.2可能用到的相关算法介绍

|  |
| --- |
| **算法1**. 直方图均衡化算法 |
| 输入：原始图像  输出：均衡化后的图像  过程：  第1步：列出原始图灰度级f，f = 0,1,2……,7。  第2步：列出原始直方图。  第3步：利用公式 计算累积直方图。  第4步：利用公式 g = int[(L-1)]取整。  第5步：确定映射关系（f->g）。  第6步：计算出新的直方图。 |

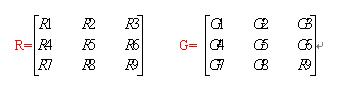
|  |
| --- |
| **算法2**. 非线性平滑滤波（中值滤波）算法 |
| 输入：原始图像  输出：中值滤波后的图像  过程：主要运用到模板运算。  第1步：将模板在图中漫游，并将模板中心与图中某个像素位置重合。  第2步：读取模板下各个对应像素的灰度值。  第3步：将这些灰度值从小到大排成一列。  第4步：找出灰度值的中值。  第5步：将这个灰度值赋给对应模板中心位置的像素。 |

|  |
| --- |
| **算法3**. Canny算子 |
| 输入：原始图像  输出：检测后的图像  过程：   1、对传入的彩色图片二值化  2、对二值化后的图片进行高斯滤波  3、使用sobel算子对滤波之后的图形分别求x，y方向上的梯度  4、计算出梯度幅值  5、对梯度图像做非极大抑制  6、对非极大抑制后的图像做双阈值处理  7、对双阈值处理后的图像进行连接操作（处理阈值中间的点） |

##### 3.3使用模板处理图像的过程：

模板：矩阵方块，其数学含义是一种卷积运算。

卷积运算：可看作是加权求和的过程，使用到的图像区域中的每个像素分别于卷积核(权矩阵)的每个元素对应相乘，所有乘积之和作为区域中心像素的新值。  
卷积核：卷积时使用到的权用一个矩阵表示，该矩阵与使用的图像区域大小相同，其行、列都是奇数，是一个权矩阵。

卷积示例：  
 3 \* 3 的像素区域R与卷积核G的卷积运算：  
R5(中心像素)=R1G1 + R2G2 + R3G3 + R4G4 + R5G5 + R6G6 + R7G7 + R8G8 + R9G9  
     

3.4模板处理图像的问题：  
边界问题：当处理图像边界像素时，卷积核与图像使用区域不能匹配，卷积核的中心与边界像素点对应，卷积运算将出现问题。  
处理办法： A． 忽略边界像素，即处理后的图像将丢掉这些像素。  
     B． 保留原边界像素，即copy边界像素到处理后的图像。

## 4. 实验设置

本项目是基于TensorFlow的CNN花卉识别应用，所有代码都是通过Python编写，需要用到opencv和TensorFlow框架。所以本项目的运行环境如下：

操作系统：Windows7

内存：12G

硬盘：1000G

处理器：core-i7

Anaconda3的Python开源版本+TensoFlow框架+OpenCv

Python图像界面处理的PyQt5

## 5. 方法对比与结果评价

如下所示是本文用到的方法进行的花卉识别结果展示，具体运行过程在演示视频可查看，首先尽心训练集的学习，经过4层卷积层、4层池化层、2层全连接层的操作后，得到具体的训练模型。然后调用具体的训练模型进行花卉识别。通过损失函数的分析可以看到，最终的准确率在70%左右，效果较好。



图5.1结果展示

而采用迁移学习调用Inception-v3模型对本文中的花卉数据集分类准确率在95%左右。主要的原因在于本文的CNN模型较于简单，而且花卉数据集本身就比mnist手写数字数据集分类难度就要大一点，同样的模型在mnist手写数字的识别上准确率要比花卉数据集准确率高不少。

       本文的CNN模型完全可以通过增大模型复杂度或者改参数调试以及对图像进行预处理来提高准确率，由于水平有限，只能做到这里。但后续肯定还会进行优化处理，使得准确率能显著地提高。

## 6. 源代码

##### 6.1训练集Train.py：

from skimage import io,transform

import glob

import os

import tensorflow as tf

import numpy as np

import time

path='F:/Python/ComputerVision/CvData/flower\_photo/'

model\_path='F:/Python/ComputerVision/CvData/flowers/model.ckpt'

w=100

h=100

c=3

def read\_img(path):

cate=[path+x for x in os.listdir(path) if os.path.isdir(path+x)]

imgs=[]

labels=[]

for idx,folder in enumerate(cate):

for im in glob.glob(folder+'/\*.jpg'):

print('reading the images:%s'%(im))

img=io.imread(im)

img=transform.resize(img,(w,h))

imgs.append(img)

labels.append(idx)

return np.asarray(imgs,np.float32),np.asarray(labels,np.int32)

data,label=read\_img(path)

num\_example=data.shape[0]

arr=np.arange(num\_example)

np.random.shuffle(arr)

data=data[arr]

label=label[arr]

ratio=0.8

s=np.int(num\_example\*ratio)

x\_train=data[:s]

y\_train=label[:s]

x\_val=data[s:]

y\_val=label[s:]

x=tf.placeholder(tf.float32,shape=[None,w,h,c],name='x')

y\_=tf.placeholder(tf.int32,shape=[None,],name='y\_')

def inference(input\_tensor, train, regularizer):

with tf.variable\_scope('layer1-conv1'):

conv1\_weights = tf.get\_variable("weight",[5,5,3,32],initializer=tf.truncated\_normal\_initializer(stddev=0.1))

conv1\_biases = tf.get\_variable("bias", [32], initializer=tf.constant\_initializer(0.0))

conv1 = tf.nn.conv2d(input\_tensor, conv1\_weights, strides=[1, 1, 1, 1], padding='SAME')

relu1 = tf.nn.relu(tf.nn.bias\_add(conv1, conv1\_biases))

with tf.name\_scope("layer2-pool1"):

pool1 = tf.nn.max\_pool(relu1, ksize = [1,2,2,1],strides=[1,2,2,1],padding="VALID")

with tf.variable\_scope("layer3-conv2"):

conv2\_weights = tf.get\_variable("weight",[5,5,32,64],initializer=tf.truncated\_normal\_initializer(stddev=0.1))

conv2\_biases = tf.get\_variable("bias", [64], initializer=tf.constant\_initializer(0.0))

conv2 = tf.nn.conv2d(pool1, conv2\_weights, strides=[1, 1, 1, 1], padding='SAME')

relu2 = tf.nn.relu(tf.nn.bias\_add(conv2, conv2\_biases))

with tf.name\_scope("layer4-pool2"):

pool2 = tf.nn.max\_pool(relu2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

with tf.variable\_scope("layer5-conv3"):

conv3\_weights = tf.get\_variable("weight",[3,3,64,128],initializer=tf.truncated\_normal\_initializer(stddev=0.1))

conv3\_biases = tf.get\_variable("bias", [128], initializer=tf.constant\_initializer(0.0))

conv3 = tf.nn.conv2d(pool2, conv3\_weights, strides=[1, 1, 1, 1], padding='SAME')

relu3 = tf.nn.relu(tf.nn.bias\_add(conv3, conv3\_biases))

with tf.name\_scope("layer6-pool3"):

pool3 = tf.nn.max\_pool(relu3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

with tf.variable\_scope("layer7-conv4"):

conv4\_weights = tf.get\_variable("weight",[3,3,128,128],initializer=tf.truncated\_normal\_initializer(stddev=0.1))

conv4\_biases = tf.get\_variable("bias", [128], initializer=tf.constant\_initializer(0.0))

conv4 = tf.nn.conv2d(pool3, conv4\_weights, strides=[1, 1, 1, 1], padding='SAME')

relu4 = tf.nn.relu(tf.nn.bias\_add(conv4, conv4\_biases))

with tf.name\_scope("layer8-pool4"):

pool4 = tf.nn.max\_pool(relu4, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')

nodes = 6\*6\*128

reshaped = tf.reshape(pool4,[-1,nodes])

with tf.variable\_scope('layer9-fc1'):

fc1\_weights = tf.get\_variable("weight", [nodes, 1024],

initializer=tf.truncated\_normal\_initializer(stddev=0.1))

if regularizer != None: tf.add\_to\_collection('losses', regularizer(fc1\_weights))

fc1\_biases = tf.get\_variable("bias", [1024], initializer=tf.constant\_initializer(0.1))

fc1 = tf.nn.relu(tf.matmul(reshaped, fc1\_weights) + fc1\_biases)

if train: fc1 = tf.nn.dropout(fc1, 0.5)

with tf.variable\_scope('layer10-fc2'):

fc2\_weights = tf.get\_variable("weight", [1024, 512],

initializer=tf.truncated\_normal\_initializer(stddev=0.1))

if regularizer != None: tf.add\_to\_collection('losses', regularizer(fc2\_weights))

fc2\_biases = tf.get\_variable("bias", [512], initializer=tf.constant\_initializer(0.1))

fc2 = tf.nn.relu(tf.matmul(fc1, fc2\_weights) + fc2\_biases)

if train: fc2 = tf.nn.dropout(fc2, 0.5)

with tf.variable\_scope('layer11-fc3'):

fc3\_weights = tf.get\_variable("weight", [512, 5],

initializer=tf.truncated\_normal\_initializer(stddev=0.1))

if regularizer != None: tf.add\_to\_collection('losses', regularizer(fc3\_weights))

fc3\_biases = tf.get\_variable("bias", [5], initializer=tf.constant\_initializer(0.1))

logit = tf.matmul(fc2, fc3\_weights) + fc3\_biases

return logit

regularizer = tf.contrib.layers.l2\_regularizer(0.0001)

logits = inference(x,False,regularizer)

b = tf.constant(value=1,dtype=tf.float32)

logits\_eval = tf.multiply(logits,b,name='logits\_eval')

loss=tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits(logits=logits, labels=y\_)

train\_op=tf.train.AdamOptimizer(learning\_rate=0.001).minimize(loss)

correct\_prediction = tf.equal(tf.cast(tf.argmax(logits,1),tf.int32), y\_)

acc= tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

def minibatches(inputs=None, targets=None, batch\_size=None, shuffle=False):

assert len(inputs) == len(targets)

if shuffle:

indices = np.arange(len(inputs))

np.random.shuffle(indices)

for start\_idx in range(0, len(inputs) - batch\_size + 1, batch\_size):

if shuffle:

excerpt = indices[start\_idx:start\_idx + batch\_size]

else:

excerpt = slice(start\_idx, start\_idx + batch\_size)

yield inputs[excerpt], targets[excerpt]

n\_epoch=10

batch\_size=64

saver=tf.train.Saver()

sess=tf.Session()

sess.run(tf.global\_variables\_initializer())

for epoch in range(n\_epoch):

start\_time = time.time()

#training

train\_loss, train\_acc, n\_batch = 0, 0, 0

for x\_train\_a, y\_train\_a in minibatches(x\_train, y\_train, batch\_size, shuffle=True):

\_,err,ac=sess.run([train\_op,loss,acc], feed\_dict={x: x\_train\_a, y\_: y\_train\_a})

train\_loss += err; train\_acc += ac; n\_batch += 1

print(" train loss: %f" % (np.sum(train\_loss)/ n\_batch))

print(" train acc: %f" % (np.sum(train\_acc)/ n\_batch))

#validation

val\_loss, val\_acc, n\_batch = 0, 0, 0

for x\_val\_a, y\_val\_a in minibatches(x\_val, y\_val, batch\_size, shuffle=False):

err, ac = sess.run([loss,acc], feed\_dict={x: x\_val\_a, y\_: y\_val\_a})

val\_loss += err; val\_acc += ac; n\_batch += 1

print(" validation loss: %f" % (np.sum(val\_loss)/ n\_batch))

print(" validation acc: %f" % (np.sum(val\_acc)/ n\_batch))

saver.save(sess,model\_path)

sess.close()

##### 6.2调用训练模型进行花卉识别MainWindow.py：

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QFileDialog

from PyQt5.QtGui import QImage, QPixmap

class Ui\_MainWindow(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

super(Ui\_MainWindow,self).\_\_init\_\_()

self.setupUi(self)

self.retranslateUi(self)

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(751, 393)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.pushButton = QtWidgets.QPushButton(self.centralwidget)

self.pushButton.setGeometry(QtCore.QRect(620, 50, 93, 28))

self.pushButton.setObjectName("pushButton")

self.layoutWidget = QtWidgets.QWidget(self.centralwidget)

self.layoutWidget.setGeometry(QtCore.QRect(90, 260, 491, 51))

self.layoutWidget.setObjectName("layoutWidget")

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout(self.layoutWidget)

self.horizontalLayout\_3.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

self.label\_6 = QtWidgets.QLabel(self.layoutWidget)

self.label\_6.setText("")

self.label\_6.setObjectName("label\_6")

self.horizontalLayout\_3.addWidget(self.label\_6)

self.label\_5 = QtWidgets.QLabel(self.layoutWidget)

self.label\_5.setText("")

self.label\_5.setObjectName("label\_5")

self.horizontalLayout\_3.addWidget(self.label\_5)

self.label\_7 = QtWidgets.QLabel(self.layoutWidget)

self.label\_7.setText("")

self.label\_7.setObjectName("label\_7")

self.horizontalLayout\_3.addWidget(self.label\_7)

self.label\_8 = QtWidgets.QLabel(self.layoutWidget)

self.label\_8.setText("")

self.label\_8.setObjectName("label\_8")

self.horizontalLayout\_3.addWidget(self.label\_8)

self.widget = QtWidgets.QWidget(self.centralwidget)

self.widget.setGeometry(QtCore.QRect(90, 50, 491, 30))

self.widget.setObjectName("widget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.widget)

self.horizontalLayout.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout.setObjectName("horizontalLayout")

self.pushButton\_1 = QtWidgets.QPushButton(self.widget)

self.pushButton\_1.setObjectName("pushButton\_1")

self.horizontalLayout.addWidget(self.pushButton\_1)

self.pushButton\_2 = QtWidgets.QPushButton(self.widget)

self.pushButton\_2.setObjectName("pushButton\_2")

self.horizontalLayout.addWidget(self.pushButton\_2)

self.pushButton\_3 = QtWidgets.QPushButton(self.widget)

self.pushButton\_3.setObjectName("pushButton\_3")

self.horizontalLayout.addWidget(self.pushButton\_3)

self.pushButton\_4 = QtWidgets.QPushButton(self.widget)

self.pushButton\_4.setObjectName("pushButton\_4")

self.horizontalLayout.addWidget(self.pushButton\_4)

self.widget1 = QtWidgets.QWidget(self.centralwidget)

self.widget1.setGeometry(QtCore.QRect(90, 110, 491, 121))

self.widget1.setObjectName("widget1")

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.widget1)

self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

self.label\_1 = QtWidgets.QLabel(self.widget1)

self.label\_1.setText("")

self.label\_1.setObjectName("label\_1")

self.horizontalLayout\_2.addWidget(self.label\_1)

self.label\_2 = QtWidgets.QLabel(self.widget1)

self.label\_2.setText("")

self.label\_2.setObjectName("label\_2")

self.horizontalLayout\_2.addWidget(self.label\_2)

self.label\_3 = QtWidgets.QLabel(self.widget1)

self.label\_3.setText("")

self.label\_3.setObjectName("label\_3")

self.horizontalLayout\_2.addWidget(self.label\_3)

self.label\_4 = QtWidgets.QLabel(self.widget1)

self.label\_4.setText("")

self.label\_4.setObjectName("label\_4")

self.horizontalLayout\_2.addWidget(self.label\_4)

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 751, 26))

self.menubar.setObjectName("menubar")

MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

self.pushButton\_1.clicked.connect(self.choose\_file\_1)

self.pushButton\_2.clicked.connect(self.choose\_file\_2)

self.pushButton\_3.clicked.connect(self.choose\_file\_3)

self.pushButton\_4.clicked.connect(self.choose\_file\_4)

self.pushButton.clicked.connect(self.run)

def choose\_file(self):

filename, \_ = QFileDialog.getOpenFileName(self,'选择文件','','\*')

return filename

def show\_image(self, filename, label):

image = QImage(filename)

image = image.scaled(label.width(), label.height())

label.setPixmap(QPixmap.fromImage(image))

def choose\_file\_1(self):

filename = self.choose\_file()

self.show\_image(filename, self.label\_1)

self.path1 = filename

def choose\_file\_2(self):

filename = self.choose\_file()

self.show\_image(filename, self.label\_2)

self.path2 = filename

def choose\_file\_3(self):

filename = self.choose\_file()

self.show\_image(filename, self.label\_3)

self.path3 = filename

def choose\_file\_4(self):

filename = self.choose\_file()

self.show\_image(filename, self.label\_4)

self.path4 = filename

def run(self):

from skimage import io,transform

import tensorflow as tf

import numpy as np

flower\_dict = {0:'dasiy',1:'dandelion',2:'roses',3:'sunflowers',4:'tulips'}

w=100

h=100

c=3

def read\_one\_image(path):

img = io.imread(path)

img = transform.resize(img,(w,h))

return np.asarray(img)

with tf.Session() as sess:

data = []

data1 = read\_one\_image(self.path1)

data2 = read\_one\_image(self.path2)

data3 = read\_one\_image(self.path3)

data4 = read\_one\_image(self.path4)

data.append(data1)

data.append(data2)

data.append(data3)

data.append(data4)

saver = tf.train.import\_meta\_graph('F:/Python/ComputerVision/CvData/flowers/model.ckpt.meta')

saver.restore(sess,tf.train.latest\_checkpoint('F:/Python/ComputerVision/CvData/flowers/'))

graph = tf.get\_default\_graph()

x = graph.get\_tensor\_by\_name("x:0")

feed\_dict = {x:data}

logits = graph.get\_tensor\_by\_name("logits\_eval:0")

classification\_result = sess.run(logits,feed\_dict)

#print(classification\_result)

#print(tf.argmax(classification\_result,1).eval())

output = []

output = tf.argmax(classification\_result,1).eval()

for i in range(len(output)):

print("Number",i+1,"Flowers:"+flower\_dict[output[i]])

self.label\_6.setText(flower\_dict[output[0]])

self.label\_5.setText(flower\_dict[output[1]])

self.label\_7.setText(flower\_dict[output[2]])

self.label\_8.setText(flower\_dict[output[3]])

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.pushButton.setText(\_translate("MainWindow", "识别"))

self.pushButton\_1.setText(\_translate("MainWindow", "选择图片1"))

self.pushButton\_2.setText(\_translate("MainWindow", "选择图片2"))

self.pushButton\_3.setText(\_translate("MainWindow", "选择图片3"))

self.pushButton\_4.setText(\_translate("MainWindow", "选择图片4"))

##### 6.3 主函数main.py

import sys

from PyQt5.QtWidgets import QApplication , QMainWindow

from PyQt5 import QtCore, QtWidgets

from MainWindow import Ui\_MainWindow

if \_\_name\_\_ == "\_\_main\_\_":

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()

sys.exit(app.exec\_())

## 参考资料

https://www.cnblogs.com/denny402/p/5853538.html

https://blog.csdn.net/zh\_jnu/article/details/54342856

https://blog.csdn.net/XJY104165/article/details/78215545?locationNum=7&fps=1