自动水果识别与计价

Team: 果宝特工小分队

小组成员：洪依婷、卢诗颖、贾婷

指导老师：鲍杨

目录

一、项目简介：

1. 背景：

近年来，卷积神经网络（Convolutional Neural Networks）在计算机视觉上有着十分出色的表现，我们在现实生活中也或多或少都接触到相关的应用。计算机视觉作为深度学习的主要应用之一，具有很多应用场景，例如在无人自动驾驶、人脸识别等。我们小组此次的项目也正是基于此实现的。

为了满足无人化便利店及超市的需求，我们使用 Python，以 tensorflow 为环境基础实现图片的识别和分类功能，通过在 Github 上寻找相关图片集自行建立各种水果的数据库，搭建 CNN 模型进行机器学习，经过反复训练，完成识别新提供的水果照片并分类计价的功能针对日常生活中食品购买流程进行简化。此功能的实现将大大便利人们的消费过程，并为商家节省人力成本，具有巨大的商业价值。

2. 分工：

洪依婷：CNN 卷积神经网络搭建，主要负责识别学习模型板块。

贾婷：主要负责结果测试模型，构建 excel 文档并且生成自动计价功能。

卢诗颖：负责结果测试模型，整合两个模型的代码，结果可视化，调整参数以最终运行。

二、library 介绍：

➢ 关于 Tensorflow:

TensorFlow 是 Google 的开源深度学习库，你可以使用这个框架以及 Python 编程语言，构 建大量基于机器学习的应用程序。深度学习让我们能够以极高的准确性构建复杂的应用程序。 图像、视频、文本、音频等领域的问题，都可以通过深度学习解决。

Google 的 TensorFlow 图像识别系统是目前最准确的图像分类软件。所谓图像识别，就是 对图片中的内容进行识别，然而这并非对任意图片都能识别。只有被训练过的对象，系统才 能识别。

我们使用 TensorFlow 进行图像分类/物体识别。通过识别需要计价的水果品种，我们进一 步明确水果种类分类，并更加有效的实行水果分类和计价。

➢ 关于 PIL:

PIL(Python Image Library)是 python 的第三方图像处理库，可以进行图像归档、图像展示、图像处理等工作，它支持支持众多的 GUI 框架接口用于图像展示，功能强大且应用广泛。

我们使用 PIL 对拍摄的水果图片进行像素处理，进行检测水果的图片展示，便于结果的检验和展示。

三、代码解释

Part one：识别学习模型

1. 导入相关包

首先导入所需的包和模块文件(事先已 pip install)，主要有 os，sklearn，tensorflow，keras(主要用于机器学习)，numpy(主要用于数组计算)，random（实现随机选取图片训练和测试），skimage（进行图像数据的处理），

```
import matplotlib.pyplot as plt
import random
import numpy as np
import os
from sklearn.metrics import confusion_matrix, classification_report
from skimage import color,data,transform,io
from keras.optimizers import Adadelta
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from sklearn.utils import shuffle
```

2. 选取要进行训练与测试的数据

1）首先定义函数用于选取需要训练和测试的图片，将返回的数据以列表的形式存放，便于接下来对图像的处理。由于我们的数据集过于庞大，我们选择从其中选取 20 种水果进行水果分类与识别。同时，在这 20 种数据中随机选取 40 张图片进行训练与测试。（此处的参数 20 与 40 均可以根据自己的需要进行调整）

```
##由于数据集过大，所以先选取其中的部分进行训练和测试
##dir_path是图片所在的路径
##m是训练的图片种类
def load_pics(dir_path,m):
    images = []
    labels = []
    lab = os.listdir(dir_path)
    ##除去名为 .DS/STORE 的干扰
    def fun(n):
        return n.find('.D')==-1
        return n[0]!='.'

    new_list = filter(fun, lab)

    n = 0
    for l in new_list:
        if (n>=m):
            break
        image_file = os.listdir(dir_path +l)

        image_file = random.sample(list(filter(fun, image_file)),40) ##调整随机选取的图片数量提高
        for i in image_file:
            image_path = dir_path + l +'/'+i
            labels.append(int(n))
            images.append(io.imread(image_path))
        n += 1
    return images,labels
```

2) 建立水果种类的 excel 文件

在识别训练集的同时，生成 excel 文件并且将识别结果保存至 excel 中，便于后续输入水果价格以实现自动计价功能。

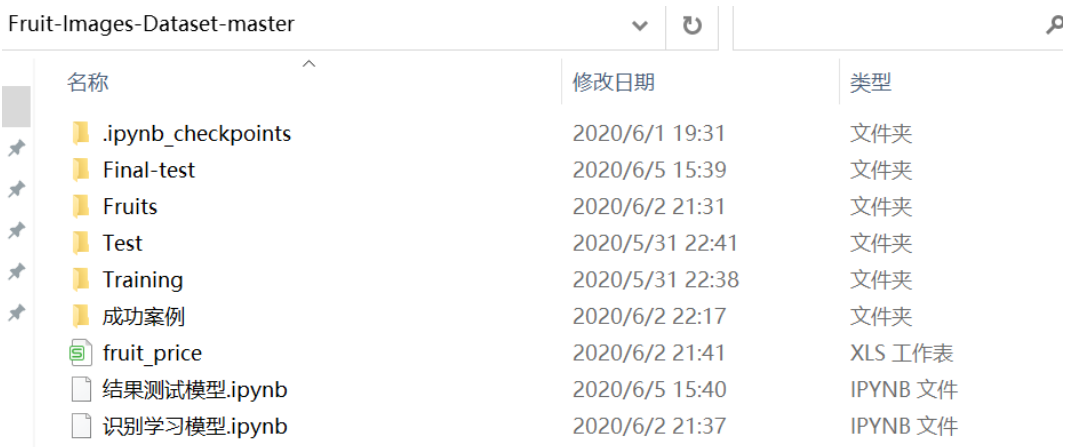
```
lab = os.listdir('/Users/apple/Desktop/Fruit-Images-Dataset-master/Test')
def fun(n):
    return n.find('.D')==-1
    return n[0]!='.'

new_list = filter(fun, lab)
kindlist  = []
for i in new_list:
    kindlist.append(i)
idx = [i for i in range(20)]
dict1 = dict(zip(idx,kindlist))

df = pd.DataFrame({
    "类别":kindlist[:20]
})

df.to_csv('/Users/apple/Desktop/Fruit-Images-Dataset-master/fruit_price.csv', encoding = 'utf-8')
```

3) 调用函数获取 training 和 test 数据。

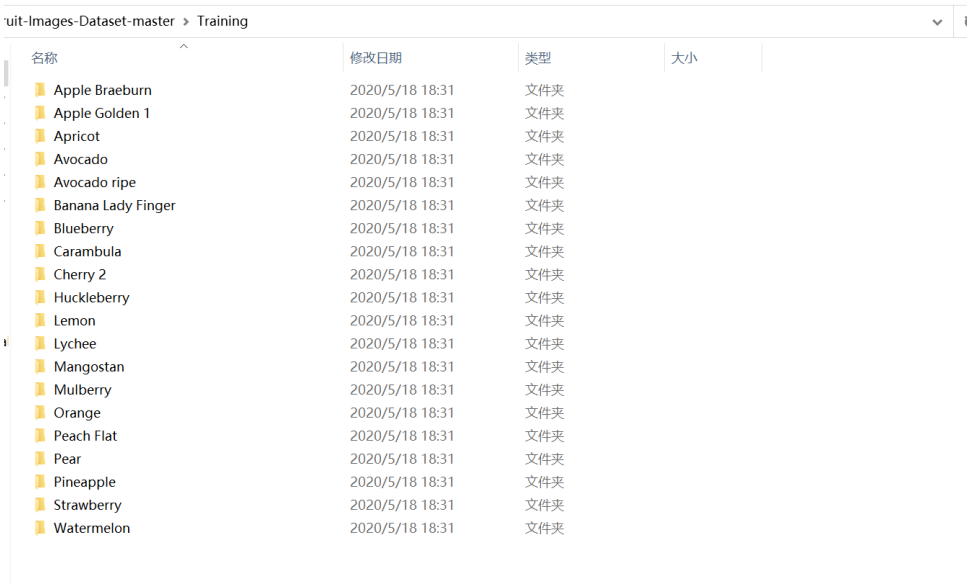
```
images_10,labels_10 = load_pics('/Users/apple/Desktop/Fruit-Images-Dataset-master/Training/',20)
```

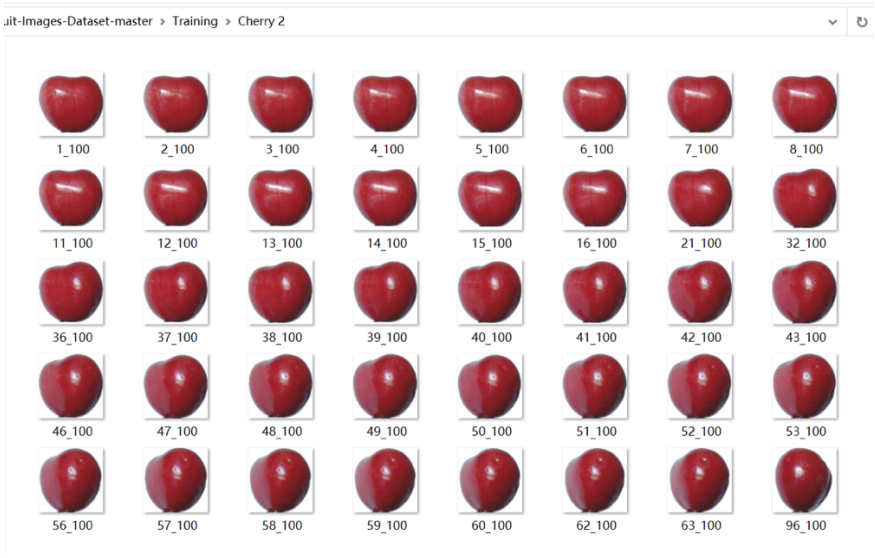
```
images_test_10, labels_test_10 = load_pics('/Users/apple/Desktop/Fruit-Images-Dataset-master/Test/',20)
```

4）我们的数据库展示：



Training 和 Test 文件夹里都包含 200 张左右水果图片以供机器识别训练。





3. 处理数据。

定义函数 prepare_data( )将数据数组化，并进行乱序操作。此处还采用了机器学习的常见操作——one hot 独热编码（独热编码：使用 N 位状态寄存器来对 N 个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候，其中只有一位有效）。

```
def prepare_data(images,labels,nums):
    ##转化为数组形式
    train_x = np.array(images)
    train_y = np.array(labels)

    indx = np.arange(0,train_y.shape[0])  ##生成序列
    index = shuffle(indx) ##将序列中所有元素随机排序
    train_x = train_x[indx]
    train_y = train_y[indx]
    train_y = keras.utils.to_categorical(train_y,nums) ##one-hot编码，使用N位状态寄存器来对N个状态进行编码，每个状态都由他独立
    return train_x,train_y

training_x,training_y = prepare_data(images_10,labels_10,20)
test_x,test_y = prepare_data(images_test_10,labels_test_10,20)
```

4. Tensorflow 卷积神经网络搭建

此处我们小组学习了李宏毅教授的相关视频课程，主要参考了 GoogleNe 的模型并进行一定的简化，搭建了两层卷积、池化层，三层全连接层。此模型是基于 tensorflow 的环境搭建的。为此，我们小组成员都自学了 tensorflow 相关用法。

下面是一些参数设置以及 feed 给神经网络的一些数据。

```
import tensorflow as tf
num_classes = 20    ##和labels种类数量一样
batch_size = 128    ##样本大小小组
kernel_h = kernel_w = 5
dropout = 0.8
depth_in = 3
depth_out1 = 64
depth_out2 = 128
image_size = training_x.shape[1]
num_training_sample = training_x.shape[0]
num_test_sample = test_x.shape[0]
```

```
##feed给神经网络的数据类型，shape
x = tf.placeholder(tf.float32,[None,100,100,3], name='input')
##feed给神经网络的标签数据类型
y = tf.placeholder(tf.float32,[None,num_classes], name='output')
##每个元素被保留的概率（防止过拟合）
keep_prob = tf.placeholder(tf.float32, name='keep_prob')

fla = int((image_size*image_size/16)*depth_out2)
```

1)定义一些卷积层和全连阶层的权重和 bias.

```
##定义各卷积层和全连接层的 weights&bias
Weights = {"conv1_weights":tf.Variable(tf.random_normal([kernel_h,kernel_w,depth_in,depth_out1])),\
           "conv2_weights":tf.Variable(tf.random_normal([kernel_h,kernel_w,depth_out1,depth_out2])),\
           "fully_connected_weights_1":tf.Variable(tf.random_normal([int((image_size*image_size/16)*depth_out2),1024])),\
           "fully_connected_weights_2":tf.Variable(tf.random_normal([1024,512])),\
           "output":tf.Variable(tf.random_normal([512,num_classes]))}

bias = {"conv1_bias":tf.Variable(tf.random_normal([depth_out1])),\
       "conv2_bias":tf.Variable(tf.random_normal([depth_out2])),\
       "fully_connected_bias1":tf.Variable(tf.random_normal([1024])),\
       "fully_connected_bias2":tf.Variable(tf.random_normal([512])),\
       "output":tf.Variable(tf.random_normal([num_classes]))}
```

2)接下来，我们开始搭建 CNN 网络。首先定义卷积层（convolution layer）、激活函数（此处采用 RELU 函数）、池化层(max-pooling layer)生成函数。

```
##定义convolution层和maxpooling层函数
def Conv2D(x,W,b,stride_set= 1):
    x = tf.nn.conv2d(x,W,strides=[1,1,1,1],padding = "SAME")
    x = tf.nn.bias_add(x,b)
    return tf.nn.relu(x)

def Max_pool2D(x,stride_set = 1):
    return tf.nn.max_pool(x,ksize=[1,2,2,1],strides=[1,2,2,1],padding = "SAME")
```

3)然后调用这两个函数构建新的函数 cnn_network，生成神经网络。

```
def cnn_network(x,weights,biases,dropout):
    ##第一层卷积、池化
    Conv1 = Conv2D(x,Weights["conv1_weights"],bias["conv1_bias"])   ## 100*100*64
    Conv1 = Max_pool2D(Conv1,2)  ##50*50*64

    ##第二层卷积、池化
    Conv2 = Conv2D(Conv1,Weights["conv2_weights"],bias["conv2_bias"])   ## 50*50*128
    Conv2 = Max_pool2D(Conv2,2)  ##25*25*128

    ##fully_connected layer1
    flatten = tf.reshape(Conv2,[-1,fla])   ##把它拍扁哈哈哈
    fully_connected1 = tf.add(tf.matmul(flatten,Weights["fully_connected_weights_1"]),bias["fully_connected_bias1"])
    fully_connected1 = tf.nn.relu(fully_connected1)  ##activate一下
    #print(flatten.get_shape())

    ##fully_connected layer2
    fully_connected2 = tf.add(tf.matmul(fully_connected1,Weights["fully_connected_weights_2"]),bias["fully_connected
    fully_connected2 = tf.nn.relu(fully_connected2)  ##activate一下

    ##dropout层，随机扔掉参数，防止过拟合
    fully_connected2 = tf.nn.dropout(fully_connected2,dropout)

    ##输出预测参数
    prediction = tf.add(tf.matmul(fully_connected2,Weights['output']),bias['output'])

    return prediction
```

4)调用函数得到预测值，并计算 loss。由于我们的研究问题是多分类问题，所以此处我们采用交叉熵函数进行损失计算。

```
prediction = cnn_network(x,Weights,bias,keep_prob)

predictions = tf.argmax(prediction, 1, name = "predictions")

##loss函数计算，此处采用交叉熵
cross_entropy_loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=prediction,labels=y))
```

5)为了解决计算过程中随机梯度下降所造成的陷入局部最小点而无法继续更新参数所带来的问题，我们加入优化器 AdamOptimizer,并进行多次调整，选取了一个相对较优的 learning_rate。

```
##优化器选择AdamOptimizer
optimizer = tf.train.AdamOptimizer(0.0009).minimize(cross_entropy_loss)  ##此处可更改参数进行训练提高accuracy
```

6) 调用函数评估模型。

```
##evaluate
correct_prediction = tf.equal(tf.argmax(prediction,1),tf.argmax(y,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))
```

7) 由于一次性喂给神经网络过多的数据不利于 accuracy 的提高，我们建立了函数 generate_small_data 将一次训练的数据分成 n 个小块。

```
##训练块数据生成器
def generate_small_data(inputs,batch_size):
    i = 0
    while True:
        small_data = inputs[i:(batch_size + i)]
        i += batch_size
        yield small_data
```

8) 接下来就正式运行神经网络并开始训练与测试。根据不同参数所得到的最终 accuracy，不断修改并选用其中最高的一组作为我们最终选取的参数。

```
with tf.Session() as sess:
    tf.global_variables_initializer().run()   ###初始化
    for i in range(100): ##此处可调整epoch提高accuracy
        training_x,training_y = prepare_data(images_10,labels_10,20) ##重新预处理数据
        training_x = generate_small_data(training_x,batch_size) ##生成图像块数据
        training_y = generate_small_data(training_y,batch_size) ##生成标签块数据
        for j in range(int(num_training_sample/batch_size)+1):
            xx = next(training_x)
            yy = next(training_y)

            ##准备验证数据
            validate_feed = {x:xx , y:yy , keep_prob: 0.8}
            if i % 1 == 0:
                sess.run(optimizer,feed_dict = validate_feed)
                loss,acc = sess.run([cross_entropy_loss , accuracy],feed_dict = {x:xx , y:yy , keep_prob: 0.8})
                print("EPOCH:", "%04d" % (i+1),"cost=","{:.9f}".format(loss),"Training accuracy","{:.5f}".format(acc
        checkpoint_path = os.path.join('/Users/apple/Desktop/Fruit-Images-Dataset-master/Fruits/', 'fruits.ckpt')
        saver = tf.train.Saver()
        saver.save(sess, checkpoint_path)
    print('Optimization Completed')

    ##准备测试数据
    test_x = test_x[0:400]
    test_y = test_y[0:400]
    test_feed = {x:test_x, y:test_y , keep_prob: 0.8}
    y0 = sess.run(prediction, feed_dict = test_feed)
    test_classes = np.argmax(y0,1)
    print("Testing Accuracy:", sess.run(accuracy, feed_dict = test_feed))
```

Part Two：结果测试模型

完成模型的训练之后，我们需要将模型与应用场景相结合。为此，我们构思了以下的应用场景：

1、设计一张水果单价表，存放所有参与训练的水果的单价，并可以通过预测结果调出其单价。

2、将一张新的图片输入到模型中，判断其所属水果类型并调取单价。

3、输入水果重量，与单价相乘，构成这一水果总价。

在代码中，我们首先先将训练好的 Tensorflow 模型参数与算法图保存至本地，避免每次都要重新训练而浪费时间。接着设计函数读取模型，将需要进行识别的图片路径传入。最后进行识别，通过对应机制将预测结果与单价表中的对应单价建立联系，并通过输入水果总重后得到该水果的总价。

1、导入相关包

主要就是 tensorflow、pandas 和 PIL 对识别学习模型进行导入和检测图片进行操作。

```
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import random
import os
import pandas as pd
import matplotlib.image as img

from keras.utils import np_utils
from keras.datasets import mnist
from keras.utils import to_categorical
from pandas import DataFrame
from PIL import Image,ImageGrab
```

2、定义选取并显示最终测试图片函数 selet_test_image

首先使用 matplotlib.image()功能读取进行测试用的图片，接着使用 PIL 中 Image.open()功能打开图片，ImageGrab 中 resize()功能对不符合模型所定义

的像素的图片进行缩放或扩大，再次使用 PIL 中 plt.imshow()、plt.show()功能读取并显示图片，方便后续结果检测。再来就是使用 np.array 处理图片，生成数据。

```
def selet_test_image(path):
    image = img.imread(path)
    image = Image.open(path)
    image = image.resize((100, 100),Image.ANTIALIAS)#图片缩放
    plt.imshow(image)
    plt.show()#显示选择的图片
    image = np.array(image).reshape(1,100,100,3)
    return image
```

3、定义测试函数 test

首先用 tensorflow 的功能导入识别学习模型，再定义了进行模型数据读取的变量 graph，通过导入 meta 文件的方法可以避免重新搭建网络，较为便捷。接着获取模型中的变量和数据，给模型喂食数据输出预测结果 result。通过之前读取类别和价格的表格，使用 result 给出的数值对类别和价格进行定位输出，由用户输入水果重量，机器显示出水果名称并且计算出总价。

```
def test(image_test):
    df = pd.read_csv('/Users/admin/Desktop/Fruit-Images-Dataset-master/fruit_price
    pred = 0
    #加载.meta文件构建图
    with tf.compat.v1.Session() as sess:
        saver = tf.compat.v1.train.import_meta_graph('/Users/admin/Desktop/Fruit-I
        saver.restore(sess, tf.train.latest_checkpoint(model_path))#导入最新的模型
        graph = tf.compat.v1.get_default_graph()

        #获取模型变量
        X = graph.get_tensor_by_name('input:0')
        keep_c_prob = graph.get_tensor_by_name('kb:0')
        logits = graph.get_tensor_by_name('predictions:0')
        result = sess.run(logits, feed_dict={X: image_test, keep_c_prob:0.8})
        for i in result:
            pred = i
        print("预测的水果为: ",df[df['Unnamed: 0']==pred]["类别"].values)
        weights = input("输入该水果的重量")
        total = float(weights) * float(df[df['Unnamed: 0']==pred]["单价"].values)
        print("这个水果的总价为",total,"元")
```

4、进行测试

最后就是调用函数和图片储存路径，进行结果测试。

```
got_image = selet_test_image(image_dir)
test(got_image)
```

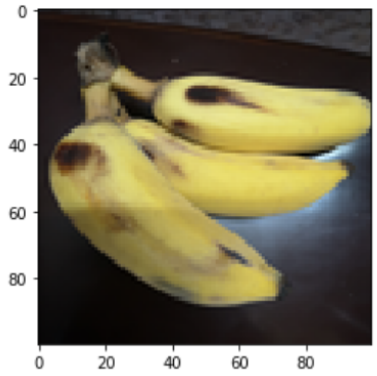
四、成果与展示

1. 最终我们所采用的识别学习模型准确率达到 90%以上。

```
EPOCH: 0148 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0149 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0149 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0149 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0149 cost= 1671491.000000000 Training accuracy 0.98438
EPOCH: 0149 cost= 234437.500000000 Training accuracy 0.99219
EPOCH: 0149 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0149 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0150 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0150 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0150 cost= 277790.000000000 Training accuracy 0.99219
EPOCH: 0150 cost= 1039133.000000000 Training accuracy 0.99219
EPOCH: 0150 cost= 1860397.000000000 Training accuracy 0.98438
EPOCH: 0150 cost= 0.000000000 Training accuracy 1.00000
EPOCH: 0150 cost= 0.000000000 Training accuracy 1.00000
Optimization Completed
Testing Accuracy: 0.9475
```

2. 当我们将一张待检测的图片
放入结果测试模型时，结果如下：

```
#进行测试
got_image = selet_test_image(image_dir)
test(got_image)
```




```
INFO:tensorflow:Restoring parameters from /Users/admin/Desktop/Fruit-Images-[
预测的水果为： ['Banana Lady Finger']
输入该水果的重量 1
这个水果的总价为 6.0 元
```

检测图片与“Banana Lady Finger”特征点相似最高，故显示预测的水果为“Banana Lady Finger”，输入水果重量，可计算得出水果总价。

五、局限性与优化设想：

本实验通过使用深度学习中卷积神经网络，通过 Tensorflow 实现了水果图片的识别。数据方面，通过下载 Github 上 Fruit-Images-Dataset 的数据集，获取到了大量质量优异且品种繁多的水果图片，之后训练模型，并实现了新水果图片的识别与出价。

但是，考虑到实际的应用背景、代码的简洁性和识别准确性，本项目具有可以进一步优化的可能，主要从数据、代码模型与实际功能三个角度。

1.数据方面：

此程序现阶段必须要把待检测的图片和图片库(已做的 database)放在一个文件夹里才能执行。我们希望以后能够通过学习 Java 等语言制作相关小程序，直接通过小程序实现识别计价功能，简化拍照后上传至本地电脑的过程。

同时，现阶段我们收集到的水果图片都是没有背景或者其他无噪音的优质图片，但实际应用中我们待预测的图片都会是从平常的摊位上拍照下来的图片，这些真实场景中的图片具有一定的噪音，如何让模型在不那么理想的图片中仍然识别到目标水果就成了问题，而这一点由于时间原因并没有在本次实验中直接表现，我们在代码中仅仅是简单地进行了剪裁处理。但我们展望可以利用代码将图片进行去噪、灰度处理等操作，提高识别的准确率。

2.代码模型：

由于我们的模型只是初步实现了一个完整但是比较简单的卷积神经网路，实际上没有过多去优化整个网络。据我们的了解，目前在计算机视觉领域，迁移学习在图像识别中具有很高的使用价值，在经历资料查阅，我们发现以下几种网络，可以为我们后续的优化提升中提供帮助。

框架	技术手段	结构特点
VGGNet	激活函数为 Relu，采用 dropout 技术，数据增强技术， 多 GPU 平行训练技术，使用1×1和3×3的小卷积核，分类器使用的 Softamx 回归	小卷积核使判决函数更具有判决性，具有更少的参数增加了非线性表达能力，网络结构更深，计算量更大。

ResNet	激活函数为 Relu，多 GPU 平行训练技术，引入残差块，平均池化，分类器使用的 Softamx 回归等技术	引入残差单元，通过直接将输入信息绕道传到输出，保护信息的完整性整个网络只需要学习输入、输出差别的那一部分，简化学习目标和难度。在一定程度上解决了信息传递的时候或多或少会存在信息丢失，损耗等问题。
DenseNet	激活函数为 Relu，多 GPU 平行训练技术，平均池化，分类器使用的 Softamx 回归等技术	由若干个 Dense Block 串联起来而得到的，在每个 Dense lock 之间有一个 Convolution+Pooling 的操作，DenseNet 通过连接操作来结合 feature map，并且每一层都与其他层有关系，解决了深层网络的梯度消失问题，加强了特征的传播，鼓励特征重用，减少了模型参数。

3.实际应用：

我们的展望是在实际应用时给电子称重器配备摄像头，用户将购买的水果放入指定区域，称重器利用传感器、摄像头拍照并检测重量，利用识别程序识别拍下图片的水果种类，再将传感器得到的数值输入程序，根据设置好的价格计算得到总价。

同时，我们未来也可以再进一步扩大数据库，以此推广到对其他消费品的识别，如蔬菜、零食、日化用品等等，形成更高的消费价值。

六、结论：

这一次的小组作业让我们积极地思考并发现了可应用 python 编程来解决的实际生活中的问题，将理论付诸于实践，进一步感受学科魅力。

在设计自动水果识别与计价这一程序的过程中，我们在已有的 python 编程基础上，大胆挑战深度学习领域：自学 github 上开源文件以及 youtube 上相关教学视频，研究 Tensorflow 识别图片的功能，了解有关卷积神经网络的相关知识.....期间经历了许多困难和瓶颈，我们通过向老师、助教寻求指导建议、网上查找相关资料和小组讨论探究，最终完成了“自动水果识别与计价”这一程序。

总而言之，通过本次项目，我们小组成员都收获颇丰，极大地提高了自己的编程能力以及自学能力。

最后，感谢鲍杨老师一学期以来的悉心指导！