

CSE4019 - Image Processing

Food Recognition and Calorie Estimation using CNN and Image Processing Techniques

By : -

P Bhargav Reddy - 19BCE1342

Lingam V V S N Datta Sai - 19BCE1373

B Jivitesh - 19BCE1275

Submitted to:-

Dr. Geetha S

**Abstract:**

Nowadays, regular intake of nutritious foods is important for maintaining a healthy eating routine and avoiding obesity within the physical body .. during this paper, a completely unique framework hooked into AI that naturally performs exact grouping of food pictures and gauges food credits. within the preparation section of the model system, it proposes a profound learning model that has a convolutional neural organization that orders food into explicit classifications. The principle reason for the proposed technique is to enhance the exactness of the pre-preparing model. The paper plans a model framework hooked into the customer worker model. The customer sends an image location solicitation and cycles it on the worker side. Support Vector Machine (SVM), Artificial Neural Network (ANN), and Convolution Neural Network (CNN) are the three classifiers that are modified to research the framework's enhanced precision. Exploring various avenues a few sorts of food groups, each with an outsized number of images, using AI (AI) to realize higher grouping precision.

**Problem Statement:**

The problem can be simply stated as, given a set of food images with calibration object thumb with the food name and an unlabeled set of food images from the same group of food, identify food and estimate food volume and calories intake.

**Introduction:**

Accurate and passive acquisition of dietary data from patients is important for a far better understanding of the etiology of obesity and development of effective weight management programs. Self-reporting is currently the most method for such data acquisition. However, studies have shown that data obtained by self-reporting seriously underestimate food intake and thus don't accurately reflect the important habitual behavior of people . Computer food recognition programs haven't yet been developed. during this paper, we present a study for recognizing foods from videos of eating, which are directly recorded in restaurants by an internet camera. From recognition results, our method then estimates food calories of intake.

The advantage of recognizing items, rather than the entire meal, is that the system are often trained with only single item food images. At the training stage, we first use region proposal algorithms to get candidate regions and extract the convolutional neural network (CNN) features of all regions. Second, we perform region mining to pick positive regions for every food category using maximum cover by our proposed submodular optimization method. At the testing stage, we first generate a group of candidate regions. for every region, a classification score is computed supported its extracted CNN features and predicted food names of the chosen regions

**Dataset:**

For a single food portion, we took several pairs of images by using smart phones; each group of images contains a top view and a side view of this food. For each image, there will be a plate and a finger pointing towards the object and no more than two foods in it. There shouldn't be two foods in the same image.

**Methodology:**

Calorie estimation method based on deep learning:

we use deep learning algorithms to recognize the types of food and apply image segmentation to identify the food’s contour in the photos. So as the side view. then, the volumes of each food is calculated based on the calibration objects in the images. In the end, the calorie of each food is obtained by searching the density table and a nutrition table.

Objection detection With Deep Learning Methods

We do not use semantic segmentation methods such as Fully Convolutional Networks (FCN) but choose to use Faster R-CNN. Faster R-CNN is a framework based on deep convolutional networks. It includes a Region Proposal Network (RPN) and an Object Detection Network. When we put an image with RGB channels as input, we will get a series of bounding boxes.

Volume Estimation And Calorie Calculation

According to the contours detected in the top view, the true size of a pixel is known. Similarly, we know the actual size of a pixel in the side view. Then we use different formulas to estimate the volume of each food.

**Result and Discussion:**

#image\_segment

import cv2

import numpy as np

import os

def getAreaOfFood(img1):

data=os.path.join(os.getcwd(),"images")

if os.path.exists(data):

print('folder exist for images at ',data)

else:

os.mkdir(data)

print('folder created for images at ',data)

cv2.imwrite('{}\\1 original image.jpg'.format(data),img1)

img = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

cv2.imwrite('{}\\2 original image BGR2GRAY.jpg'.format(data),img)

img\_filt = cv2.medianBlur( img, 5)

cv2.imwrite('{}\\3 img\_filt.jpg'.format(data),img\_filt)

img\_th = cv2.adaptiveThreshold(img\_filt,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,21,2)

cv2.imwrite('{}\\4 img\_th.jpg'.format(data),img\_th)

contours, hierarchy = cv2.findContours(img\_th, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE) #make change here

# find contours. sort. and find the biggest contour. the biggest contour corresponds to the plate and fruit.

mask = np.zeros(img.shape, np.uint8)

largest\_areas = sorted(contours, key=cv2.contourArea)

cv2.drawContours(mask, [largest\_areas[-1]], 0, (255,255,255,255), -1)

cv2.imwrite('{}\\5 mask.jpg'.format(data),mask)

img\_bigcontour = cv2.bitwise\_and(img1,img1,mask = mask)

cv2.imwrite('{}\\6 img\_bigcontour.jpg'.format(data),img\_bigcontour)

# convert to hsv. otsu threshold in s to remove plate

hsv\_img = cv2.cvtColor(img\_bigcontour, cv2.COLOR\_BGR2HSV)

cv2.imwrite('{}\\7 hsv\_img.jpg'.format(data),hsv\_img)

h,s,v = cv2.split(hsv\_img)

mask\_plate = cv2.inRange(hsv\_img, np.array([0,0,50]), np.array([200,90,250]))

cv2.imwrite('{}\\8 mask\_plate.jpg'.format(data),mask\_plate)

mask\_not\_plate = cv2.bitwise\_not(mask\_plate)

cv2.imwrite('{}\\9 mask\_not\_plate.jpg'.format(data),mask\_not\_plate)

fruit\_skin = cv2.bitwise\_and(img\_bigcontour,img\_bigcontour,mask = mask\_not\_plate)

cv2.imwrite('{}\\10 fruit\_skin.jpg'.format(data),fruit\_skin)

#convert to hsv to detect and remove skin pixels

hsv\_img = cv2.cvtColor(fruit\_skin, cv2.COLOR\_BGR2HSV)

cv2.imwrite('{}\\11 hsv\_img.jpg'.format(data),hsv\_img)

skin = cv2.inRange(hsv\_img, np.array([0,10,60]), np.array([10,160,255])) #Scalar(0, 10, 60), Scalar(20, 150, 255)

cv2.imwrite('{}\\12 skin.jpg'.format(data),skin)

not\_skin = cv2.bitwise\_not(skin); #invert skin and black

cv2.imwrite('{}\\13 not\_skin.jpg'.format(data),not\_skin)

fruit = cv2.bitwise\_and(fruit\_skin,fruit\_skin,mask = not\_skin) #get only fruit pixels

cv2.imwrite('{}\\14 fruit.jpg'.format(data),fruit)

fruit\_bw = cv2.cvtColor(fruit, cv2.COLOR\_BGR2GRAY)

cv2.imwrite('{}\\15 fruit\_bw.jpg'.format(data),fruit\_bw)

fruit\_bin = cv2.inRange(fruit\_bw, 10, 255) #binary of fruit

cv2.imwrite('{}\\16 fruit\_bw.jpg'.format(data),fruit\_bin)

#erode before finding contours

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))

erode\_fruit = cv2.erode(fruit\_bin,kernel,iterations = 1)

cv2.imwrite('{}\\17 erode\_fruit.jpg'.format(data),erode\_fruit)

#find largest contour since that will be the fruit

img\_th = cv2.adaptiveThreshold(erode\_fruit,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,11,2)

cv2.imwrite('{}\\18 img\_th.jpg'.format(data),img\_th)

contours, hierarchy = cv2.findContours(img\_th, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

mask\_fruit = np.zeros(fruit\_bin.shape, np.uint8)

largest\_areas = sorted(contours, key=cv2.contourArea)

cv2.drawContours(mask\_fruit, [largest\_areas[-2]], 0, (255,255,255), -1)

cv2.imwrite('{}\\19 mask\_fruit.jpg'.format(data),mask\_fruit)

#dilate now

kernel2 = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))

mask\_fruit2 = cv2.dilate(mask\_fruit,kernel2,iterations = 1)

cv2.imwrite('{}\\20 mask\_fruit2.jpg'.format(data),mask\_fruit2)

fruit\_final = cv2.bitwise\_and(img1,img1,mask = mask\_fruit2)

cv2.imwrite('{}\\21 fruit\_final.jpg'.format(data),fruit\_final)

#find area of fruit

img\_th = cv2.adaptiveThreshold(mask\_fruit2,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,11,2)

cv2.imwrite('{}\\22 img\_th.jpg'.format(data),img\_th)

contours, hierarchy = cv2.findContours(img\_th, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

largest\_areas = sorted(contours, key=cv2.contourArea)

fruit\_contour = largest\_areas[-2]

fruit\_area = cv2.contourArea(fruit\_contour)

#finding the area of skin. find area of biggest contour

skin2 = skin - mask\_fruit2

cv2.imwrite('{}\\23 skin2.jpg'.format(data),skin2)

#erode before finding contours

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(5,5))

skin\_e = cv2.erode(skin2,kernel,iterations = 1)

cv2.imwrite('{}\\24 skin\_e .jpg'.format(data),skin\_e )

img\_th = cv2.adaptiveThreshold(skin\_e,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,11,2)

cv2.imwrite('{}\\25 img\_th.jpg'.format(data),img\_th)

contours, hierarchy = cv2.findContours(img\_th, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

mask\_skin = np.zeros(skin.shape, np.uint8)

largest\_areas = sorted(contours, key=cv2.contourArea)

cv2.drawContours(mask\_skin, [largest\_areas[-2]], 0, (255,255,255), -1)

cv2.imwrite('{}\\26 mask\_skin.jpg'.format(data),mask\_skin)

skin\_rect = cv2.minAreaRect(largest\_areas[-2])

box = cv2.boxPoints(skin\_rect)

box = np.int0(box)

mask\_skin2 = np.zeros(skin.shape, np.uint8)

cv2.drawContours(mask\_skin2,[box],0,(255,255,255), -1)

cv2.imwrite('{}\\27 mask\_skin2.jpg'.format(data),mask\_skin2)

pix\_height = max(skin\_rect[1])

pix\_to\_cm\_multiplier = 5.0/pix\_height

skin\_area = cv2.contourArea(box)

return fruit\_area,fruit\_bin ,fruit\_final,skin\_area, fruit\_contour, pix\_to\_cm\_multiplier

#caleries

import cv2

import numpy as np

#density - gram / cm^3

density\_dict = { 1:0.609, 2:0.94, 3:0.641, 4:0.641,5:0.513, 6:0.482,7:0.481}

#kcal

calorie\_dict = { 1:52, 2:89, 3:41,4:16,5:40,6:47,7:18 }

#skin of photo to real multiplier

skin\_multiplier = 5\*2.3

def getCalorie(label, volume): #volume in cm^3

calorie = calorie\_dict[int(label)]

density = density\_dict[int(label)]

mass = volume\*density\*1.0

calorie\_tot = (calorie/100.0)\*mass

return mass, calorie\_tot, calorie #calorie per 100 grams

def getVolume(label, area, skin\_area, pix\_to\_cm\_multiplier, fruit\_contour):

area\_fruit = (area/skin\_area)\*skin\_multiplier #area in cm^2

label = int(label)

volume = 100

if label == 1 or label == 5 or label == 7 or label == 6 : #sphere-apple,tomato,orange,kiwi,onion

radius = np.sqrt(area\_fruit/np.pi)

volume = (4/3)\*np.pi\*radius\*radius\*radius

#print (area\_fruit, radius, volume, skin\_area)

if label == 2 or label == 4 or (label == 3 and area\_fruit > 30): #cylinder like banana, cucumber, carrot

fruit\_rect = cv2.minAreaRect(fruit\_contour)

height = max(fruit\_rect[1])\*pix\_to\_cm\_multiplier

radius = area\_fruit/(2.0\*height)

volume = np.pi\*radius\*radius\*height

if (label==4 and area\_fruit < 30) : # carrot

volume = area\_fruit\*0.5 #assuming width = 0.5 cm

return volume

def calories(result,img):

img\_path =img # "C:/Users/M Sc-2/Desktop/dataset/FooD/"+str(j)+"\_"+str(i)+".jpg"

fruit\_areas,final\_f,areaod,skin\_areas, fruit\_contours, pix\_cm = getAreaOfFood(img\_path)

volume = getVolume(result, fruit\_areas, skin\_areas, pix\_cm, fruit\_contours)

mass, cal, cal\_100 = getCalorie(result, volume)

fruit\_volumes=volume

fruit\_calories=cal

fruit\_calories\_100grams=cal\_100

fruit\_mass=mass

#print("\nfruit\_volumes",fruit\_volumes,"\nfruit\_calories",fruit\_calories,"\nruit\_calories\_100grams",fruit\_calories\_100grams,"\nfruit\_mass",fruit\_mass)

return fruit\_calories

#cnn\_model

import tflearn

from tflearn.layers.conv import conv\_2d, max\_pool\_2d

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.estimator import regression

import tensorflow as tf

def get\_model(IMG\_SIZE,no\_of\_fruits,LR):

try:

tf.reset\_default\_graph()

except:

print("tensorflow")

convnet = input\_data(shape=[None, IMG\_SIZE, IMG\_SIZE, 3], name='input')

convnet = conv\_2d(convnet, 32, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 64, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 128, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 64, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = conv\_2d(convnet, 32, 5, activation='relu')

convnet = max\_pool\_2d(convnet, 5)

convnet = fully\_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)

convnet = fully\_connected(convnet, no\_of\_fruits, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning\_rate=LR, loss='categorical\_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard\_dir='log')

return model

import os

import cv2

import numpy as np

IMG\_SIZE = 400

LR = 1e-3

no\_of\_fruits=7

MODEL\_NAME = '/content/drive/MyDrive/dip/model/Fruits\_dectector-0.001-5conv-basic.model'

model\_save\_at=os.path.join("/content/drive/MyDrive/dip/model",MODEL\_NAME)

model=get\_model(IMG\_SIZE,no\_of\_fruits,LR)

model.load(model\_save\_at)

labels=list(np.load('/content/drive/MyDrive/dip/labels.npy'))

# test\_data='/content/drive/MyDrive/dip/test\_image.JPG'

test\_data='/content/drive/MyDrive/dip/orange.png'

img=cv2.imread(test\_data)

img1=cv2.resize(img,(IMG\_SIZE,IMG\_SIZE))

model\_out=model.predict([img1])

result=np.argmax(model\_out)

name=labels[result]

cal=round(calories(result+1,img),2)

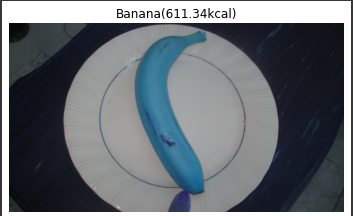
import matplotlib.pyplot as plt

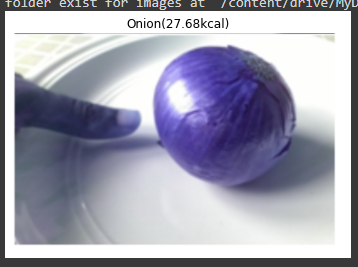
plt.imshow(img)

plt.title('{}({}kcal)'.format(name,cal))

plt.axis('off')

plt.show()







**Conclusion:**

In practice, the traditional models in machine learning are not attaining much accuracy when it comes to image classification. In this project, the CNN model is applied in image recognition. Much data augmentation and segmentation have to be performed as well and clean pixel values are not necessary for CNN as it on its own learn the generalized pattern required to identify and recognize new images. So using the CNN model, the accuracy is comparatively a lot higher than all other traditional models.

References:

[1] <https://arxiv.org/pdf/1705.07632.pdf>

[2] <https://mm.cs.uec.ac.jp/e/pub/conf17/171127ege_0.pdf>

[3] <https://www.researchgate.net/profile/Keiji-Yanai/publication/309128551_An_Automatic_Calorie_Estimation_System_of_Food_Images_on_a_Smartphone/links/605807a6458515e8345ff678/An-Automatic-Calorie-Estimation-System-of-Food-Images-on-a-Smartphone.pdf>

[4] <http://img.cs.uec.ac.jp/pub/conf17/171024ege_0.pdf>