##### FOOD CALORIE DETECTION AND ESTIMATION

##### A PROJECT REPORT

##### (15CS752–Software Application Development Lab)

###### ***Submitted by***

**J.B.ARVIND BALAAJEE(201804011)**

**J.JOYAL HARIS(201804038)**

**S.MADHESH RAJ(201804050)**

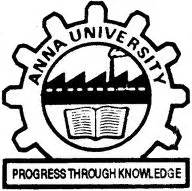
***in partial fulfillment for the award of the degree***

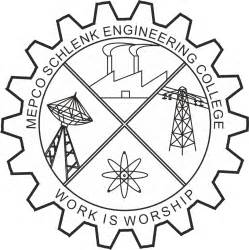
***of***

**BACHELOR OF ENGINEERING**

*in*

# **COMPUTER SCIENCE AND ENGINEERING**



**MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI (AUTONOMOUS)**

**ANNA UNIVERSITY: CHENNAI 600 025**

**NOVEMBER 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“FOOD CALORIE DETECTION AND**

**ESTIMATION”** is the bonafide work of “**J.B.ARVINDBALAAJEE(201804011),**

**J.JOYAL HARIS(201804038), S.MADHESH RAJ(201804050)”** who carried out the

project work in 15CS752 – Software Application Development Lab.

SIGNATURE SIGNATURE

**Dr.J.Rajasekar M.E,Ph.D Mr.N.Umakanth M.Tech,(Ph.D)**

Sr. Prof. and Head of the Department Assitant Professor(Sr.Grade),

Dept.of Computer Science and Engg. Dept. of Computer Science and Engg.

Mepco Schlenk Engg. College (Autonomous), Mepco Schlenk Engg. College (Autonomous),

Sivakasi, Virudhunagar Dt.-626 005 Sivakasi, Sivakasi,Virudhunagar Dt.-626 005

Submitted to Viva-Voce examination held on .......... **/11/ 2021**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

The ease with which food is being delivered at our doorsteps has led to an outbreak of a major chronic

disease known as obesity. As the necessity of the food arose among people, the apprehension related to

their diet also simultaneously increased. In this paper we propose a calorie measurement system

whereby the user is made to upload the image of food item and as a result, number of calories present in

the uploaded food image will be predicted. It is a multi-task system which also displays the weekly

statistics on how much calorie is consumed by the user and how more/less calories must be consumed to

avoid obesity related diseases such as heart attack, cancer etc. We built a dataset of food images

collected from existing datasets to detect complex images consisting of 20 classes and each class

containing 500 images each. We have curated our own Convolutional Neural Network architecture of 6

layers to extract the features and classify the images. Our experimental results on food recognition

showed 78.7% testing accuracy with 93.29% training accuracy.

Software designed to accurately estimate food calories from still images could help users and health

professionals more efficiently identify dietary patterns and food choices associated with health and

health risks. However, calorie estimation from images is difficult, and no publicly available software

can do so accurately while minimizing the burden associated with data collection and analysis.

**ACKNOWLEDGEMENT**

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavours.

We extend our sincere gratitude to our college management and Principal

**Dr.S.Arivazhagan M.E., Ph.D.,** for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete out project.

We would like to extend our heartfelt gratitude to **Dr.J.Rajasekar M.E.,Ph.D.,** Senior Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving me the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would like to extend our gratitude to **Mr.C.Balasubramanian M.Tech.,(Ph.D.,)** Assistant Professor (Sr. Grade), Department of Computer Science and Engineering, Mepco Schlenk Engineering College , for being our Project Coordinator and for his moral support and suggestions.

We would also like to extend our gratitude and sincere thanks to

**Mr.N.Umakanth M.E. ,(Ph.D)**Assistant Professor (Sr. Grade), Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Guide. She has put her valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

We would also like to extend our gratitude and sincere thanks to

**Mr. C.Soundararajan B.E.**,Teamlead,Devops,SRE(2012 - 2016 Batch Alumni of CSE),Software Engineer in L&T Info Tech,Chennai for being our Project Guide. He has put his valuable experience and expertise in directing, and helps us to complete our project with his industrial experience.

Our Sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards our beloved **family and our friends** for their support which made the project a successful one.

# **TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **CHAPTERNO. TITLE** | **PAGE NO.** |
| **ABSTRACT** | **3** |
| **1. INTRODUCTION** | **5** |
| 1.1 Product Perspective | 5 |
| 1.2 Product Description | 5 |
| **2. METHODOLOGY** | **6** |
| 2.1 Functional Requirements | 6 |
| 2.2 Non Functional Requirements | 9 |
| 2.3 Interface Requirements | 10 |
| **3. DESIGN** | **11** |
| 3.1 Architectural Design | 11 |
| 3.2 Data Set | 15 |
| 3.3 Low Level Design | 17 |
| **4. SYSTEM IMPLEMENTATION** | **18** |
| **5. TEST CASES AND TEST RESULTS** | **25** |
| **6. CONCLUSION** | **29** |
| **7. APPENDIX** | **29** |

**CHAPTER 1**

**INTRODUCTION**

* 1. **PRODUCT PERSPECTIVE**

The main purpose of this Food calorie detection system is that to detect the food and to show its corresponding calorie value. The user will show the Food infront the camera and this software will detect the food and displays the food name along with its calorie

* 1. **PRODUCT DESCRIPTION**

We use various image processing and classification techniques to identify the food, calculate the volume and the nutritional content. A mixture of methods including canny edge detection, watershed segmentation, morphological operators and Otsu’s method were used to segment the food item to obtain the contour of the fruit and the contour of the thumb. We use the thumb finger for calibration purposes. The thumb is placed next to the dish while clicking the photo and this thumb gives us the estimate of the real-life size of the food item and helps estimate volume accurately.

Once the food item is extracted from the image, we extract the feature vector of the image for training/testing purposes. The feature vector is calculated by using hsv histogram for color features, Gabor filters for texture features, and hu moments for shape, and the area for size. The feature vector is a 95x1 dimensional vector. We used Support Vector Machine model for training the images using our 95-dimensional feature vector. We obtained an accuracy of 94% in the classification of food item.

**CHAPTER 2**

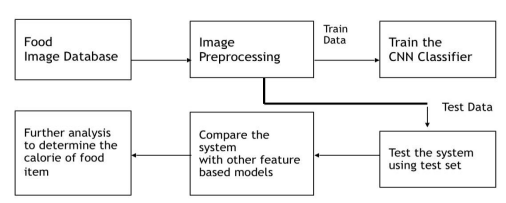
**METHODOLOGY**

**2.1 FUNCTIONAL REQUIREMENTS**

**2.1.1 Food Calorie Detection:**

In this method we detect the food based on their size, shape, color and texture.

A system is developed which detects the food from the given input. In addition to this, the system also helps to estimate the calorie intake of the food. The Proposed flow diagram is depicted in the figure 1



**Figure 1 BLOCK DIAGRAM FOR IMAGE PROCESSING**

The steps involved in image processing are preprocessing and neural network training and

from this, the trained model is obtained which will classify any supplied image based on the

trained dataset. The proposed methodology allows for automatic food detection and calorie

estimation. The system consists of four stages:

1. Image Acquisition and Preprocessing

2. Neural Network Training

3. Image segmentation

4. Calorie Estimation

**2.1.2 Image Acquisition and Preprocessing:**

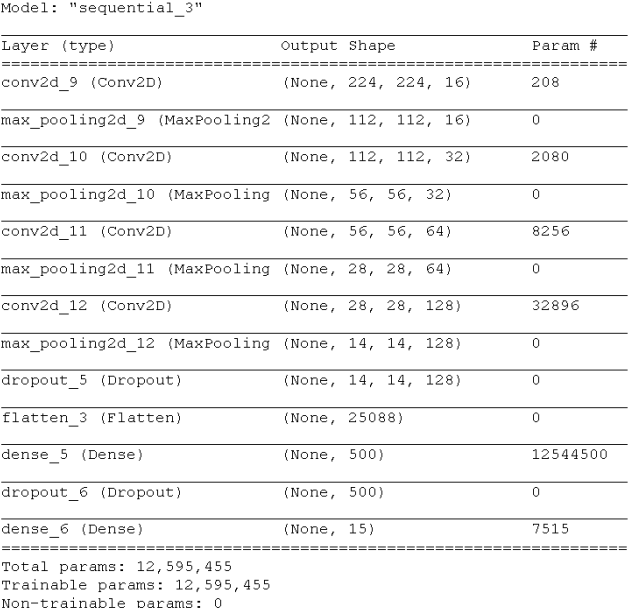
Fruits 360 dataset [11] is used which contains 90483 images of 131 fruits and vegetables. For our project we consider 15 types of fruits and vegetables. The dimensions of the images are 100x100 which are resized to 224\*224.

After resizing the images are converted to 4D tensors with the shape of (1, 224, 224, 3) and can then be passed to the Convolutional Neural Network for learning

**2.1.3 Neural Network Training:**

The base of the model being made is deep learning and the technique used is convolution neural networks (CNN), the model is being developed from scratch. CNN also called ConvNet is a class of deep neural networks which is based on shared-weights architecture and translation invariance characteristics. In CNN, the network employs a mathematical operation called Convolution. Convolution is a specialized kind of linear operation. A CNN consists of an input and an output layer with the multiple hidden layers between input and output layer.

The model now takes the input array of dimension 4 and processes it further through the network. The main aim was to give perfect result with less errors so the number of filters were increased. First we started with 16 filters and kept padding as same as it may help not to lose the data and then the MaxPooling layer with pool\_size=2 to reduce our data width wise. Then to go deep inside and classify, the number of filters were increased to 128 in following sequence 16,32,64,128 and droupout layer were used to reduce the chances of overfitting. The flattering layer is used to connect CNN layer to fully connected flattering layer. Then the hidden layers are connected having ’relu’ activation function. The end layer of the model is dense layer of 15 nodes as we have 15 types of food which uses Softmax function’ which gives the probabilities of the type of fruit in the Image. The Sequential CNN model is compiled with a Root Mean Square Propagation (RMS Prop) optimizer which uses moving average of squared gradients which normalizes itself. Also, the model is compiled with the ‘categorial\_crossentropy’ loss.

****

**Figure 2 MODEL SUMMARY**

**2.1.4 Image Segmentation:**

Segmentation is the process of partitioning a digital image into multiple segments with the aim to simplify or change the representation of the image into something that is more meaningful and easier to analyze. An adaptive thresholding is used to convert an image consisting of gray scale pixels to just black and white scale pixels. Usually a pixel value of 0 represents white and the value 255 represents black with the numbers from 1 to 254 representing different gray levels. A Contour based segmentation is done by calculating the number of contours and by finding the biggest contour. The biggest contour corresponds to the plate and the food. An interface was used that allowed users to segment the food portion using a polygonal tool and the application also detected the one-centimeter square using Hue Saturation Value (HSV) color thresholds to measure food portion area. Once the image has been convert to HSV to remove plate and fruit pixels, we obtain fruit pixels and then it is converted to binary. Later we have implemented two fundamental morphological operations that is erosion and dilation. First erosion has been applied to remove the pixels from object boundaries and then dilation has been implemented to add the pixels to the object and the area of the food has been calculated. Finally, we then have image of just the fruit (in this case apple). For different foods different set of morphological operations need to be performed multiple times in order to get the food region. To calculate area we convert the pixels to area in cm square. We do this by dividing the area with the skin area and multiple it with a constant which is found by trail and error.

**2.1.5 Calorie Estimation:**

Once area of the food has been found, the volume of the food has been calculated considering the different shape of the food. In our case we have used an apple. First Radius is obtained by dividing it with pi (3.14). Then volume of sphere is obtained using the formula

### NON FUNCTIONAL REQUIREMENTS

### 2.2.1 Performance and response time

The system should have high performance rate when executing user’s input and should able to provide feedback or response within a short time span.

### 2.2.2 Technical Feasibility

As mentioned earlier, the task of coding and debugging was made easier by using CNN for the model. As the only webcam to capture images and much GUI is not required, the app construction coding was also not a very difficult task. As every segment will be coded individually and separately, the risk assessment for each segment is easy

### 2.2.3 Ease of use

Considered the level of the knowledge possessed by the user of this system,

a simple hand gestures should be developed to make it easy to understand and required less training.

### 2.2.4 Error handling

Should be considerably minimized and an appropriate error message that guides the user to recover from an error should be provided and validation of users input should be highly essential and also the standard time taken to recover from an error should be minimum time.

### 2.2.5 Operational Feasibility

The user just has to show the object infront of camera. The camera has to be in good quality which can easily captured and processed. Hence, the controls are simple and basic. The efficiency of the project is based upon a few factors, namely, the model and its individual accuracy, the front end and its latency, the inter-connectivity. All these components can again be controlled individually and hence accuracy of the project as a whole can be easily controller.

**2.3 INTERFACE REQUIREMENTS**

**2.3.1 Software Interfaces**

Technology Implemented : Apache Server

Language Used : Python

Module Used : OpenCV, CNN

Dataset : Food Image Dataset

**2.3.2 Hardware Interfaces**

Processor : AMD or Higher Version.

Operating System : Windows 8/ Windows 10/ Window 9/Linux

RAM : Minimum 256 MB, 2GB recommended

Hardware Devices : Keyboard with mouse and camra

Hard disk : 10GB or More

Display : Standard Output Display

**2.3.3 System interface**

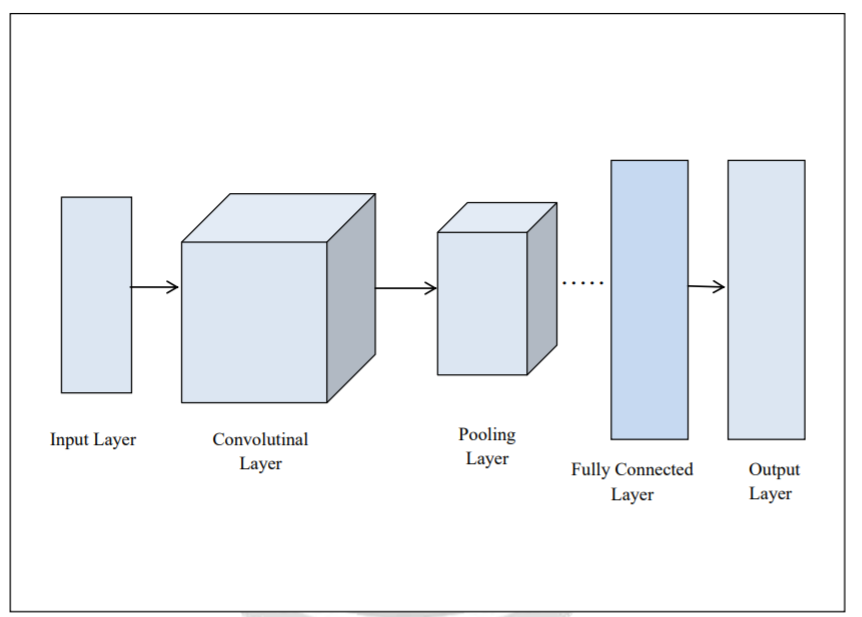
The software will acess the camera and take the picture for detection.

**CHAPTER 3**

**DESIGN**

**3.1 ARCHITECTURAL DESIGN**

**3.1.1Arichtectural Diagram**

****

**Fig-3: Layers of a Convolutional Neural Network**

**3.1.2Network Architecture**

The convolutional neural network used in this paper consists of three convolution zlayers

and three pooling layers followed by two fully connected layers. The input to the network is

a 227 × 227 × 3 image and the output is a distribution over 43 labels in the model. The first

convolution layer has 48 filters of size 11 × 11. The second convolution layer has 128 filters

of size 5 × 5 and finally, the last convolution layer has 128 filters of size 3 × 3.

Convolutional Neural Networks are the most widely used types of artificial neural

networks. CNNs have successfully been used in image and video recognition voice

recognition and signal processing recommender systems and natural language processing.

The main property of a convolutional neural network is its sparse connectivity. Each neuron

in a CNN layer is connected to a subset of neurons in an adjacent layer by a set of weights

as a result, a spatially local correlation exists around each neuron. These weights are shared

between different neurons in a layer and represent the correlation filters



**Figure 4 Neural Network**

**3.1.3 Preprocessing**

A few preprocessing steps were conducted before the training phase to ensure

reliable results. We resized all of the images in the database to the width and height of 256 pixels

without preserving the aspect ratio. Then the resized images were cropped to the size of 227 × 227

with random offsets

All the components that are needed to build a CNN: Convolution, ReLU and Pooling. Here the

output of max pooling is fed into the classifier which is usually a multi-layer perceptron layer. In

CNNs these layers are used more than once i.e. Convolution ->ReLU -> Max-Pool -> Convolution

->ReLU -> Max-Pool and so on. Now for the classification part, fully connected layer is used

which involves ReLU->Dense->Soft-max and so on.

Throughout the study Convolutional Neural Network is used to justify images of fruits containing 4

different classes and achieved accuracy of 99.89%. A block diagram is a short road map for that

graphically represents how the data moves through the existing system. The block diagram shown

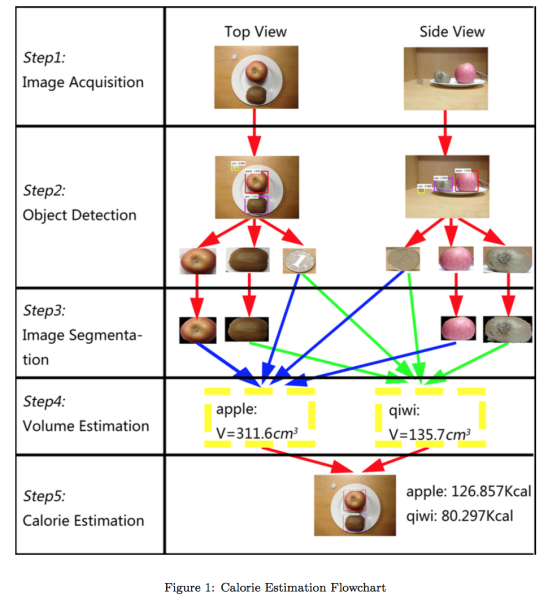
in figure is used in design process. The block diagram provides facilitating communication

between us and user. It shows the input and output information i.e. what kinds of information will

be input to and output from the system, where the data will come from and go to, and where the

data will be stored. However it does not show information about the timing of processes but shows

the work procedure of the processes.



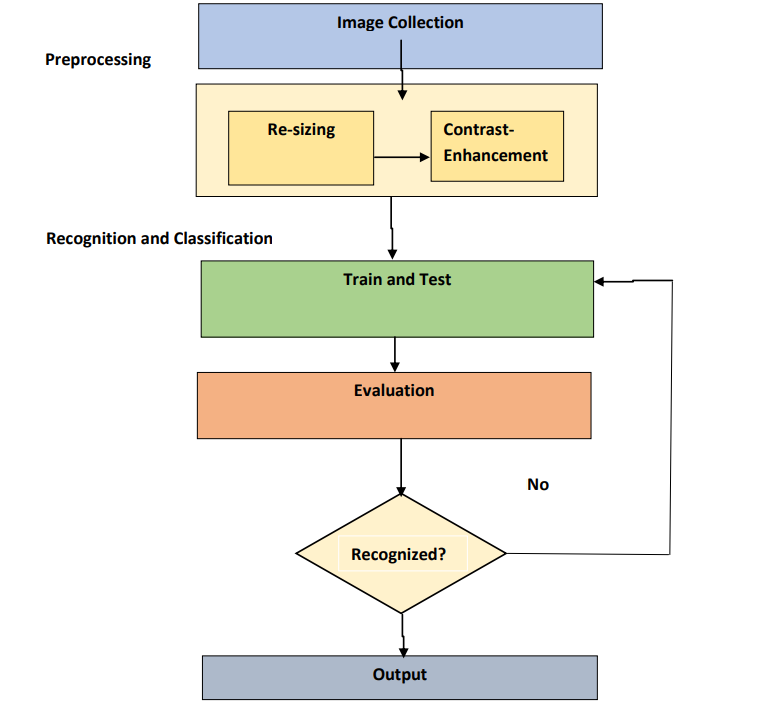
**Figure5 displays the details of the dataset**

As the collection process of the data by smart phone the images were in different shape and

sizes and training a convolutional neural network on crude pictures will most likely lead to

terrible classification exhibitions. So the images are resized into square shape (256 x 256 pixel)

and reduced unnecessary object from the images



**Figure 6Block Diagram of proposed System**

* 1. **DATA SET**
     1. **Dataset Description**

The dataset created by the images captured by smart phone. This dataset of fruits

which is categorized into four classes. A challenging data set of 4 fruits categories,

with 2403 real world images in total are introduced. The images were collected from

different fruit shops with various angles. It incorporates different yet in addition

outwardly and semantically comparative fruit classes where each class consists of

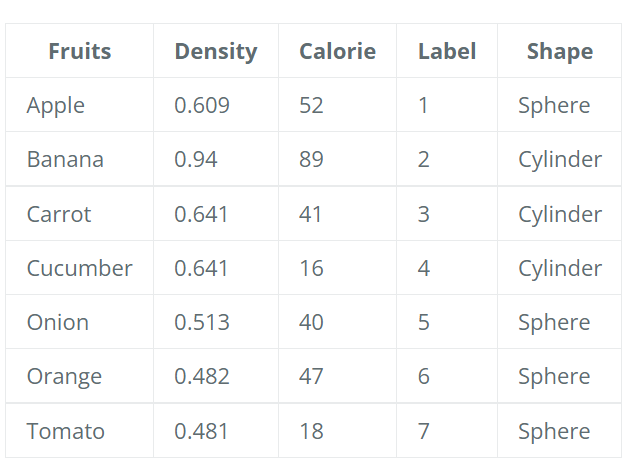
565 of image among which 100 are manually reviewed test images and 465 are

training images.

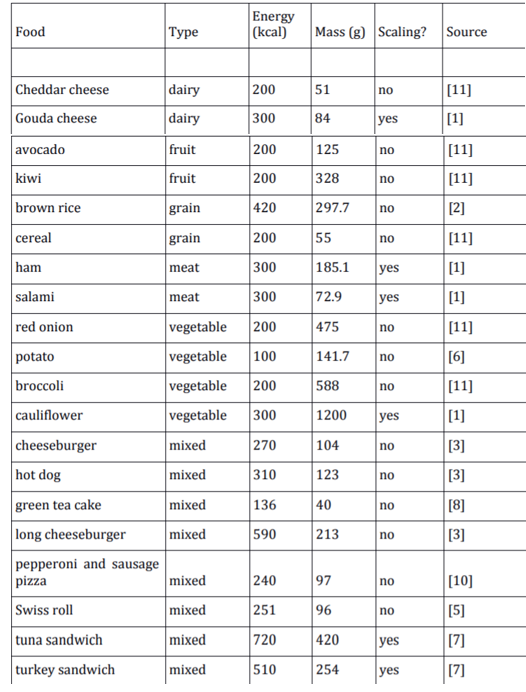


**Figure 7 Different Foods**

* + - 1. **Food Dataset**

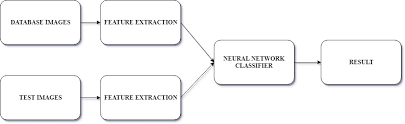
****

**Figure 8.1 Food data set**

****

**Figure 8.2 Food Data Set**

* 1. **LOW LEVEL DESIGN**

****

**Figure 8.3 Low Level Design**

**CHAPTER 4**

**4.PROCESSING**

4.1 **Architecture**

This particular model is designed to recognize some fruits those are

familiar to us. Layer like Convolution is used followed by pooling layer. Also dense layer

and a few regularization strategies such as batch normalization along with dropout are

utilized to design this model. First and Second layer are convolution layer having a size

of filter 32 and size of kernel 3. Layer 1 is considered to be the input layer which asks for

the size 32x32 of RGB channel. It also used same padding and 1 as stride. Both layers use

ReLU (1) activation and have the same property for padding and stride. Output of layer 2

connected with max pooling layer. It has the pool size of 2 and stride 2. Consider this one as

layer 3 which is connected to another convolution layer 4 of filter 64 and kernel size of 3.

Layer 4 also holds the same property as layer 2 except the filter size. Another max pooling

layer 5 having the pool size of 2 is attached to it. It also has the stride of 2. ReLU(X) =

MAX( 0, X) (1) Layer 6 a batch normalization layer is placed after layer 5. Batch

normalization enables to uses the higher learning rate and makes the learning process quicker. Layer 7 is also a convolution with 128 filter size. Except for the filter size other

thing remains the same as the other convolution layer. Layer 7 is attached to a max pooling

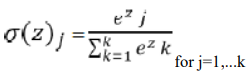
layer 8 having the pool size of 2. After all of these 8 layers are placed, the outcome is

smoothing into an array and undergo a dense layer which is considered as layer 9 with 256

concealed units and normalized with half (50%) dropout. And all the previous flow is

associated with a dense layer 10 with 4 units which is fully connected with SoftMax (2)

actuation. And this is how the model is built.



for j=1,...k (2) Algorithm for the proposed model

Set ADAM(Rate of learning = 0.0001) 2. For 30 iterations in all batch

do:

{ i. Set first convolutional layer Convolution

1(Filter =32, Kernel Size=3, Stride=1, Padding=SAME, Activation=relu)

ii. Rearrange first convolutional layer as second Convolution 1(Filter =32, Kernel Size=3 Stride=1, Padding=SAME, Activation=relu)

iii. MaxPool 1(Pool Size=2, Stride=2) a. do Dropout (Rate=25%)

iv. Set third convolutional layer Convolution 1(Filter =64, Kernel Size=3, Stride=1, Padding=SAME, Activation=relu)

v. MaxPool 2(Pool Size=2, Stride=2) vi. do Batch Normalization()

vii. Set fourth convolutional layer Convolution 1

(Filter =128, Kernel Size=3, Stride=1, Padding=SAME, Activation=relu)

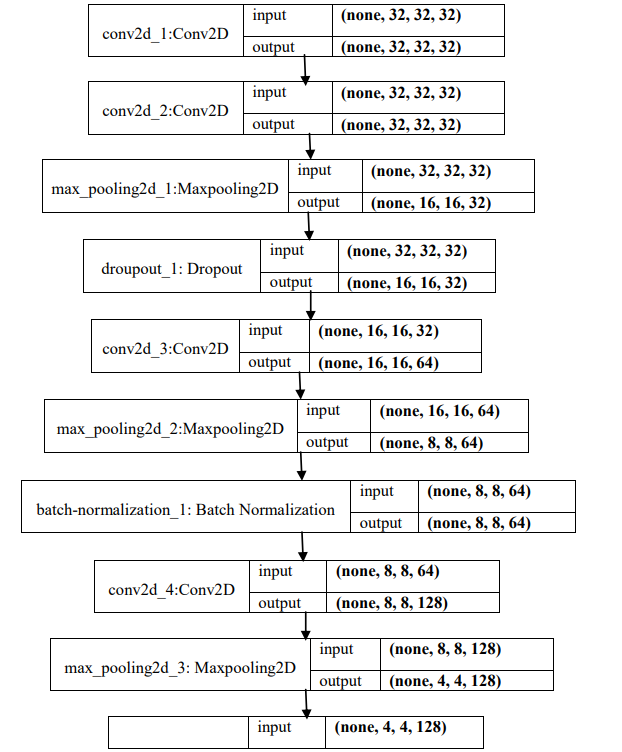
vii. MaxPool 3(Pool Size=2, Stride=2) b. do Dropout (Rate=25%)

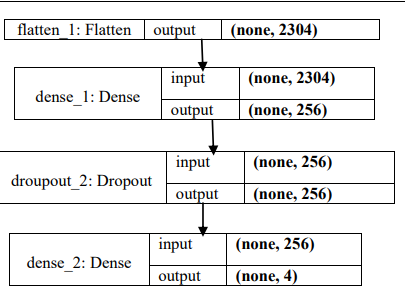
viii. for fully connected layer do { i. Flatten(Units=2304) j

Dense(Units=256,Activation=relu)

c. do Dropout (Rate=50%) k. Dense(Units=4,Activation=softmax) } }

end for





**Figure 9 Architecture For Food Recoginition**

**4.2 Optimizer and Learning rate**

For minimizing the error of a model optimizer plays a bigger role. Adam optimizer is

used in this scenario. It replaced classical stochastic gradient descent method which updates

network weights iteratively in training data. For its better execution, it is generally utilized by PC

vision researchers. In this model Adam optimizer is used with the learning rate of 0.0001.

Categorical cross entropy function is used to calculate the error. Ongoing exploration

demonstrates that cross entropy shows some quality rather than the other functions out there like

classification mistake and also mean squared mistake etc. . As out model is a multi-class



classification it is the fittest choice for us. Call for training a convolutional neural network,

Learning rate plays a huge role. The classification will be more perfect if the rate of learning is

lower. However optimizer will set aside more exertion to accomplish the global optima reducing

the loss. Apart from that higher learning rate may not be the best for the accuracy.



Achieve the desired goal become harder. The automatic learning rate reduction method is used here

to defeat this test The learning rate is set to 0.0001 toward the starting which is naturally

dropped by checking the accuracy of validation.

**4.3 Data augmentation**

There is a popular theory goes around and that is the more data you have the better performance

you get. As a result data augmentation was built to produce more data artificially by handling some

operations. For this fruit recognition model augmentation can play a huge role which is beyond

imagination. Here, the size of data can be certainly multiply by twice. For data augmentation, each

image is rotated by degree of 40, shifted the width and height by 20% randomly, rescaled and

zooms by 20%, flipped horizontally and shear with the range of 20%.



**Figure 10 Augmented Image**

**4.4 Training, Testing and the Validation of the model**

To find out the performance of the model, separate training data, testing data and validation data is

created. The training dataset is used to train the model. During training time for checking the model

performance validation set is used which helped tuning the hyper-parameters of the model. The test

data is used to finding out the performance of final model. The dataset has total 2403 food images.

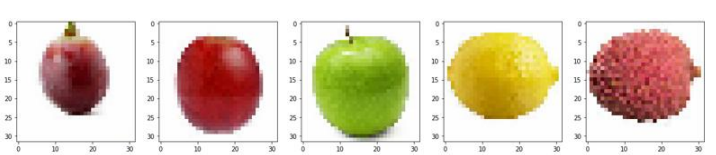
Around 25% of images (569 in total) used for validation and 75% of images (1834 in total) used to

train the model. After the training is completed, random images of fruit 5 in total is used. The

validation data consisted of various fruit pictures. Test images look like below



**Figure 11 Validation Data**

****

**Figure 12 Computer vision of Validation Data**

To check the system validation, the images of validation data is used here. In this testing the model

predicts the data which class it belongs to. During this type of testing, all of the images of

validation data have to be cropped in 256×256 sizes.

**4.5 Model performance**

Subsequent to running 30 epochs the proposed model gained the accuracy of 99.84% for the

training dataset that was created and 99.89% on the validation dataset. Completing the training

session the test on random images went pretty well. The model was able to accomplish a very

successive rate. Breaking down the outcome and confusion Matrix it can be assumed that the

performance of this model is acceptable for these kinds of fruits. The over-all performance of the

model is illustrated in figure 5.3. In figure 5.3(a), it indicates the training loss and validation loss of

overall model performance. A very plain graph indicates the minimize loss for both training and

testing of the model. Figure 5.3(b) shows the training accuracy and validation accuracy of the

overall model performance.

**4.6 Model Summary**

After training and testing the proposed model, the model can be summarized. The visualization of

the model summary is given in figure 5.5. The figure also shows the architecture of the proposed

model which includes a lot of layers to implement the model. The convolution and max-pooling

layers are used in feature extraction part and the dense and soft-max layers are used as the fully

connected layer.

**4.7 Outcome of the Model**

In the proposed system, a model is introduced to recognize fruits from images.

During this type of work, a machine learning approach has developed to establish the model.

In this study, a dataset of fruits of 4 classes is introduced for recognition. To perform the task of

the model, Convolutional

Neural Networks (CNNs) is used which was developed to perform on machine learning

approaches. This model is able to get accuracy of 99.89% which proved that the performance of

this model to recognize fruits from images is more advanced.

The high accuracy of the model shows that CNN is very suitable for this kind of fruit recognition

and also found a great algorithm for CNN which has implemented successfully for recognition of

fruits. The optimized CNN's hyper parameters showed that CNN significantly improved the fruit

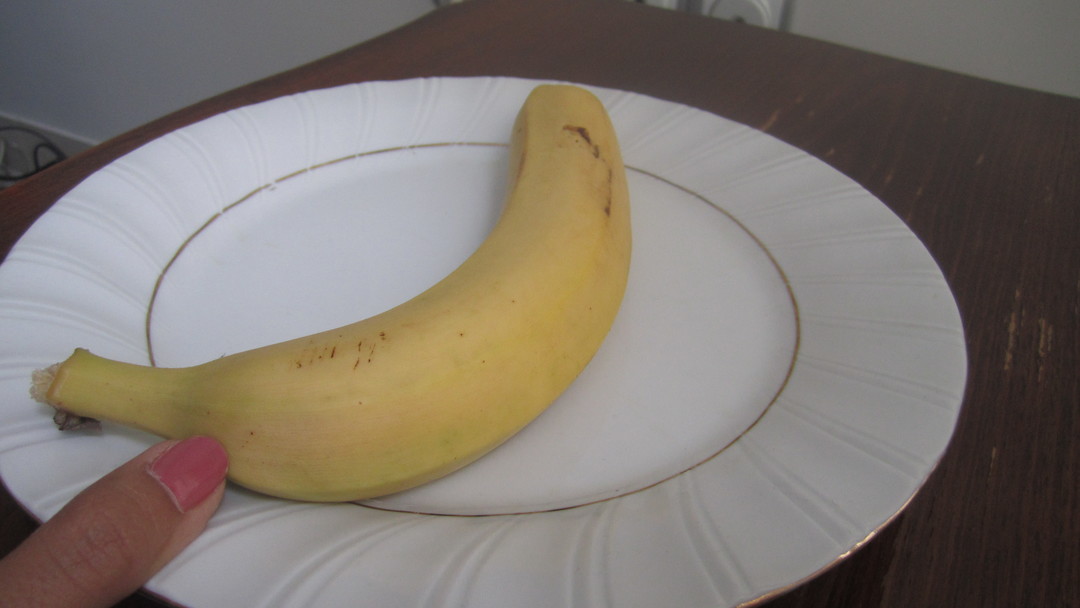
recognition accuracy compared with a conventional method using a support vector machine (SVM)

with hand-crafted features.

**CHAPTER 5**

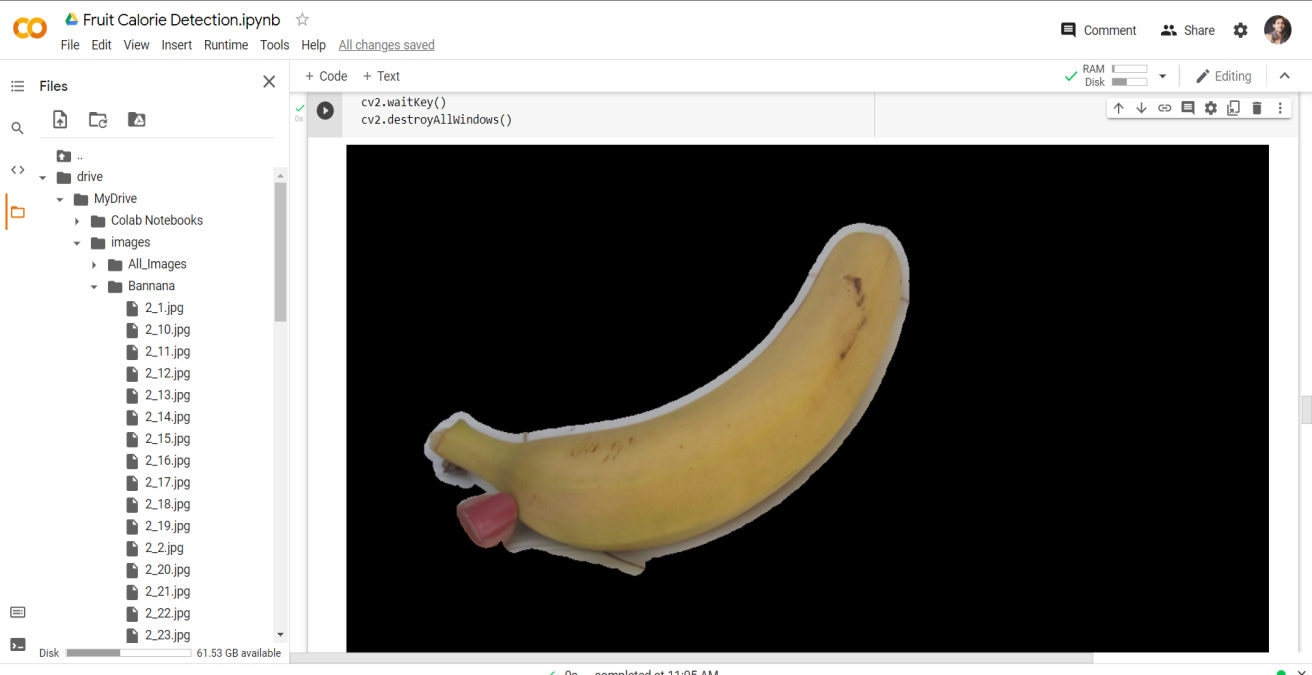
**TEST CASES AND TEST RESULTS**

* 1. **TEST CASES AND RESULTS**

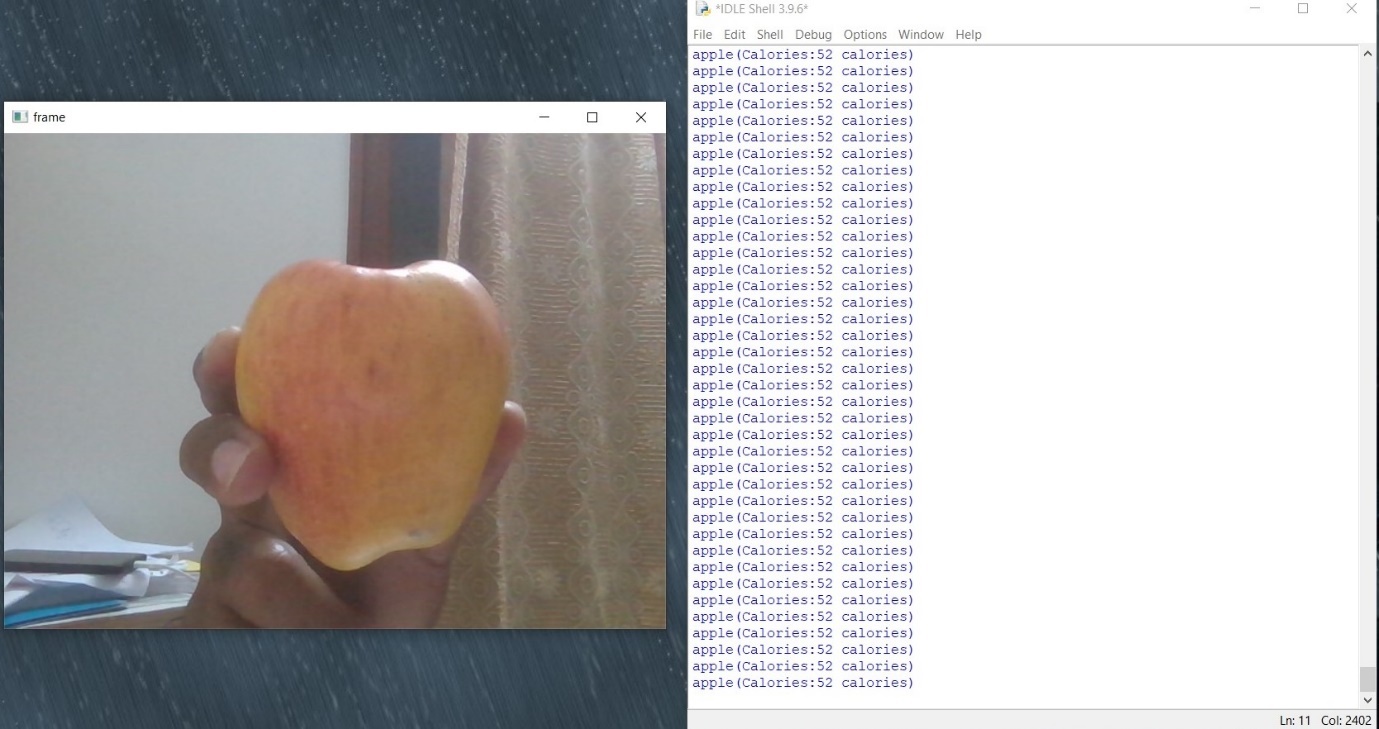
**

**Figure 13 Original Image with calibaration Point**

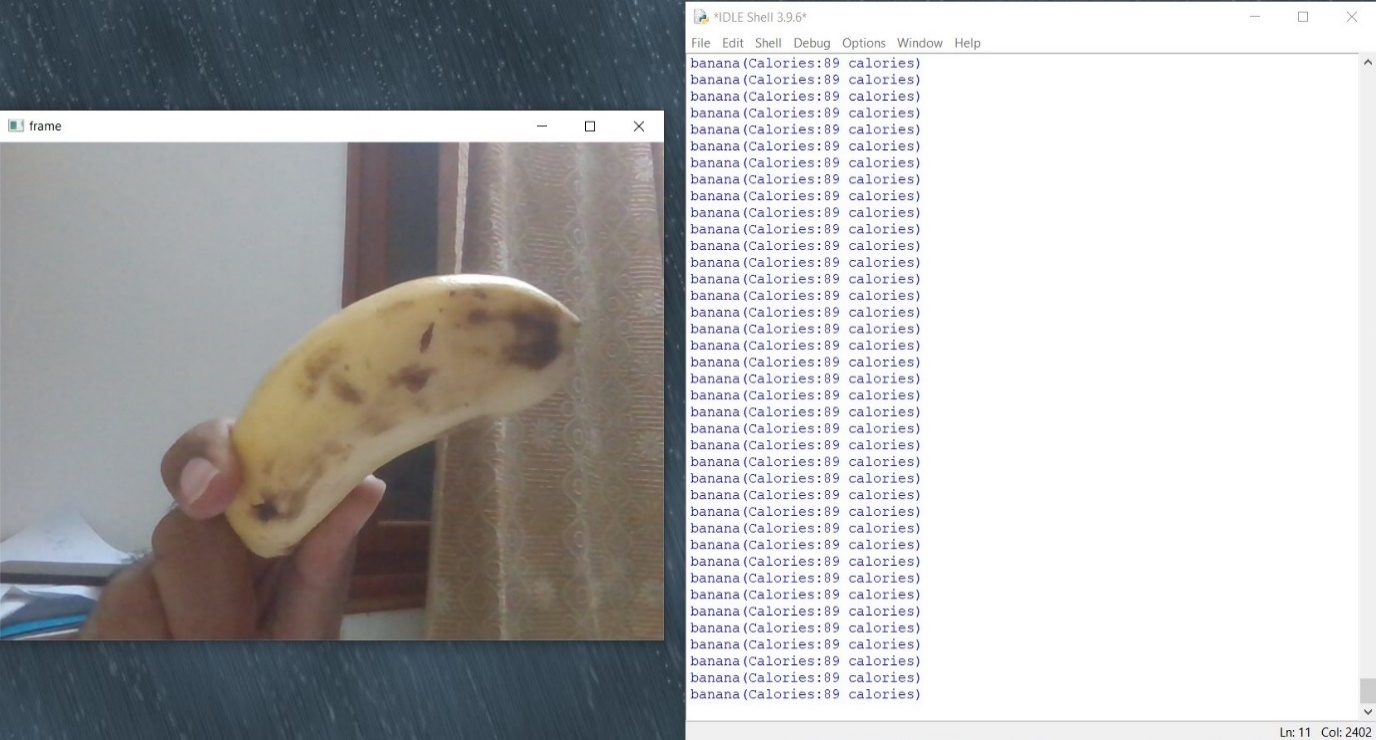
**Figure 14 Gray Scale Image**



**Figure 15 Detection Image of Banana**

****

**Figure 16 Apple Detected with Calorie Value**

****

**Figure 17 Banana Detected with Calorie Value**

### 5.1.1 Test Cases and Results for detect and estimate the calorie of apple

|  |  |
| --- | --- |
| **Test Case ID** | T01 |
| **Test Case Description:** | To detect the apple |
| **Test Data:** | Hold the apple infornt of camera |
| **Expected output:** | The apple will be detected and estimates the calories |
| **Test Result:** | PASS |

### Test Cases and Results for detect and estimate the calorie of banana

|  |  |
| --- | --- |
| **Test Case ID** | T02 |
| **Test Case Description:** | To detect the banana |
| **Test Data:** | Hold the banana infornt of camera |
| **Expected output:** | The banana will be detected and estimates the calories |
| **Test Result:** | PASS |

**CHAPTER 6**

**6.1 Conclusion**

In this software, we detected and estimate calorie of food by showing the food infront

of camera by Faster R-CNN.In this experiment, we created the fruit photo dataset. In

addition, we applied this food detector to food calorie estimation to determine the calorie of

single fruit. we also create a dataset of calorie annotated fruit images. As a results, we

estimated food calorie by capturing the food through camera. As future work, we plan to

implement multi-task CNN of the food calorie estimation and the food detection. We

expectthat the accuracy of each task improves by multi-task learning. We also plan to crop

higher-resolution images corresponding to the bounding boxes estimated by Faster R-CNN,

and use them for more accurate calorie estimation.

**6.2 Future Enhancements**

* To implement the mutli task CNN for food calorie detection and estimation to detect multiple food in the plate.
* To develop it into android app.

**CHAPTER 7**

**APPENDIX**

**SOURCE CODE**

**7.1 Detection code**

import numpy as np

from keras.preprocessing import image

import cv2

import os

import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

training\_set = train\_datagen.flow\_from\_directory('E:\\python cnn\\python cnn\\train',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

test\_datagen = ImageDataGenerator(rescale = 1./255)

test\_set = test\_datagen.flow\_from\_directory('E:\\python cnn\\python cnn\\test',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

print("Image Processing.......Compleated")

cnn = tf.keras.models.Sequential()

print("Building Neural Network.....")

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu', input\_shape=[64, 64, 3]))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

cnn.add(tf.keras.layers.Conv2D(filters=32, kernel\_size=3, activation='relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2, strides=2))

cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

cnn.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

print("Training cnn")

cnn.fit(x = training\_set, validation\_data = test\_set, epochs = 25)

cnn.save\_weights("w.h5")

#cnn.load\_weights("w.h5")

vid = cv2.VideoCapture(0)

print("Camera connection successfully established")

i = 0

while(True):

r, frame = vid.read()

cv2.imshow('frame', frame)

cv2.imwrite('E:\\python cnn\\python cnn\\final'+str(i)+".jpg", frame)

test\_image = image.load\_img('E:\\python cnn\\python cnn\\final'+str(i)+".jpg", target\_size

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = cnn.predict(test\_image)

training\_set.class\_indices

if result[0][0] ==1:

print("banana(Calories:89 calories)")

if result[0][0] == 0:

print("apple(Calories:52 calories)")

os.remove('E:\\python cnn\\python cnn\\final'+str(i)+".jpg")

i = i + 1

if cv2.waitKey(1) & 0xFF == ord('q'):

break

vid.release()

cv2.destroyAllWindows()

**7.2 Calorie Calculation**

import cv2

import numpy as np

import sys

density\_dict = { 1:0.609, 2:0.94, 3:0.577, 4:0.641, 5:1.151, 6:0.482, 7:0.513, 8:0.641, 9:0.481, 10:0.641, 11:0.521, 12:0.881, 13:0.228, 14:0.650 }

calorie\_dict = { 1:52, 2:89, 3:92, 4:41, 5:360, 6:47, 7:40, 8:158, 9:18, 10:16, 11:50, 12:61, 13:31, 14:30 }

skin\_multiplier = 5\*2.3

def getCalorie(label, volume):

calorie = calorie\_dict[int(label)]

if (volume == None):

return None, None, calorie

density = density\_dict[int(label)]

mass = volume\*density\*1.0

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

return mass, calorie\_tot, calorie

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

area\_fruit = (area/skin\_area)\*skin\_multiplier

label = int(label)

volume = 100

if label == 1 or label == 9 or label == 7 or label == 6 or label==12:

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 == 10 or (label == 4 and area\_fruit > 30):

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) or (label==5) or (label==11):

volume = area\_fruit\*0.5

if (label==8) or (label==14) or (label==3) or (label==13):

volume = None

return volume

**7.3 Feature moments**

import numpy as np

import cv2

import sys

import os

def getShapeFeatures(img):

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

moments = cv2.moments(contours[0])

hu = cv2.HuMoments(moments)

feature = []

for i in hu:

feature.append(i[0])

M=max(feature)

m=min(feature)

feature =list(map(lambda x: x \* 2, feature))

feature = (feature - M - m)/(M - m);

mean=np.mean(feature)

dev=np.std(feature)

feature = (feature - mean)/dev;

return feature

if \_\_name\_\_ == '\_\_main\_\_':

img = cv2.imread(sys.argv[1])

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

mask = cv2.inRange(img, 80, 255)

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

h = getShapeFeatures(img1)

print(h)

cv2\_imshow(img1)

cv2.waitKey()

cv2.destroyAllWindows()

**7.4 Feature Gabor**

import numpy as np

import cv2

from multiprocessing.pool import ThreadPool

def build\_filters():

filters = []

ksize = 31

for theta in np.arange(0, np.pi, np.pi / 8):

for wav in [ 8.0, 13.0]:

for ar in [0.8, 2.0]:

kern = cv2.getGaborKernel((ksize, ksize), 5.0, theta, wav, ar, 0, ktype=cv2.CV\_32F)

filters.append(kern)

cv2\_imshow(filters[9])

return filters

def process\_threaded(img, filters, threadn = 8):

accum = np.zeros\_like(img)

def f(kern):

return cv2.filter2D(img, cv2.CV\_8UC3, kern)

pool = ThreadPool(processes=threadn)

for fimg in pool.imap\_unordered(f, filters):

np.maximum(accum, fimg, accum)

return accum

def EnergySum(img):

mean, dev = cv2.meanStdDev(img)

return mean[0][0], dev[0][0]

def process(img, filters):

feature = []

accum = np.zeros\_like(img)

for kern in filters:

fimg = cv2.filter2D(img, cv2.CV\_8UC3, kern)

a,b = EnergySum(fimg)

feature.append(a)

feature.append(b)

np.maximum(accum, fimg, accum)

M = max(feature)

m = min(feature)

feature = list(map(lambda x: x \* 2, feature))

feature = (feature - M - m)/(M - m);

mean=np.mean(feature)

dev=np.std(feature)

feature = (feature - mean)/dev;

return feature

def getTextureFeature(img):

filters = build\_filters()

gray\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow("",gray\_image)

res1 = process(gray\_image, filters)

return res1

if \_\_name\_\_ == '\_\_main\_\_':

import sys

print (\_\_doc\_\_)

try: img\_fn = sys.argv[1]

except: img\_fn = 'test.JPG'

img = cv2.imread(img\_fn)

print (getTextureFeature(img)

cv2.waitKey(33)

cv2.destroyAllWindows()

**7.5 Feature color**

import cv2

import math

import sys

import numpy as np

def getColorFeature(img):

featurevec = []

img\_hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

h,s,v = cv2.split(img\_hsv)

hsvHist = [[[0 for \_ in range(2)] for \_ in range(2)] for \_ in range(6)]

hist = cv2.calcHist([img\_hsv], [0, 1, 2], None, [6,2,2], [0, 180, 0, 256, 0, 256])

for i in range(6):

for j in range(2):

for k in range(2):

featurevec.append(hist[i][j][k])

feature=featurevec[1:]

M = max(feature)

m = min(feature)

feature = list(map(lambda x: x \* 2, feature))

feature = (feature - M - m)/(M - m);

mean=np.mean(feature)

dev=np.std(feature)

feature = (feature - mean)/dev;

return feature

if \_\_name\_\_ == '\_\_main\_\_':

img = cv2.imread("/content/drive/MyDrive/images/All\_Images/10\_16.jpg")

featureVector = getColorFeature(img)

print(featureVector)

cv2.waitKey(0)

cv2.destroyAllWindows()

**7.6 Image Segmentation**

import cv2

import numpy as np

import sys

import os

def getAreaOfFood(img1):

#img1 = cv2.resize(img1, (110,110))

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

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

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

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

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)

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

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

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

mask\_plate = cv2.inRange(hsv\_img, np.array([0,0,100]), np.array([255,90,255]))

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

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

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

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

not\_skin = cv2.bitwise\_not(skin);

fruit = cv2.bitwise\_and(fruit\_skin,fruit\_skin,mask = not\_skin)

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

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

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3))

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

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

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)

kernel2 = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(13,13))

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

res = cv2.bitwise\_and(fruit\_bin,fruit\_bin,mask = mask\_fruit2)

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

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

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)

skin2 = skin - mask\_fruit2

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3))

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

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

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)

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)

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

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

skin\_area = cv2.contourArea(box)

return fruit\_area, mask\_fruit2, fruit\_final, skin\_area, fruit\_contour, pix\_to\_cm\_multiplier

if \_\_name\_\_ == '\_\_main\_\_':

img1 = cv2.imread(sys.argv[1])

area, bin\_fruit, img\_fruit, skin\_area, fruit\_contour, pix\_to\_cm\_multiplier = getAreaOfFood(img1)

cv2.imshow("",img\_fruit)

cv2.waitKey()

cv2.destroyAllWindows()

**7.7 Create Feature**

from 2feature\_moments import getShapeFeatures

from 3feature\_gabor import \*

from 4feature\_color import getColorFeature

from img\_seg import \*

def createFeature(img):

feature = []

areaFruit, binaryImg, colourImg, areaSkin, fruitContour, pix\_to\_cm\_multiplier = getAreaOfFood(img)

color = getColorFeature(colourImg)

texture = getTextureFeature(colourImg)

shape = getShapeFeatures(binaryImg)

for i in color:

feature.append(i)

for i in texture:

feature.append(i)

for i in shape:

feature.append(i)

M = max(feature)

m = min(feature)

feature =list(map(lambda x: x \* 2, feature))

feature = (feature - M - m)/(M - m)

mean=np.mean(feature)

dev=np.std(feature)

feature = (feature - mean)/dev;

return feature, areaFruit, areaSkin, fruitContour, pix\_to\_cm\_multiplier

def readFeatureImg(filename):

img = cv2.imread(filename)

f, farea, skinarea, fcont, pix\_to\_cm = createFeature(img)

return f, farea, skinarea, fcont, pix\_to\_cm

if \_\_name\_\_ == '\_\_main\_\_':

import sys

f = readFeatureImg(sys.argv[1])

print (f, len(f))

**7.8 REFERENCE**

https://www.pantechsolutions.net/food-recognition-using-opencv-and-python/

https://www.ijert.org/fruit-recognition-system-for-calorie-management

https://www.lftechnology.com/blog/ai/image-calorie-estimation-deep-learning/