***Intelligent Fruit Nutrient Analysis***

Libin Alex

**Aim:** To detect the calorie content in a fruit using CNN and Image Processing Techniques.

**Objective of the Work:**

1. **Address Nutritional Assessment Challenges**: Tackle the difficulty individuals face in accurately determining the nutritional content of their meals, which impacts dietary choices and health.
2. **Develop AI-Powered Solution**: Create an efficient, user-friendly system using AI to instantly provide sufficient nutritional information.
3. **Utilize CNN for Image Recognition**: Employ Convolutional Neural Networks to recognize food items and extract nutrient details.
4. **Volume Estimation Method**: Use thumb as a reference for calibrating images to estimate food volume accurately.
5. **Calculate Nutritional Facts**: Determine the mass and calorie content of food portions by matching measured volume with nutritional fact tables.
6. **Overcome Traditional Method Limitations**: Offer an alternative to time-consuming traditional nutritional analysis methods like manual logging or label reading.
7. **Enhance Dietary Decision-Making**: Empower users to make informed, healthier dietary choices with ease.
8. **Improve Model Generalization**: Address the need for larger, diverse datasets to enhance the AI model’s ability to generalize across various food types.
9. **Promote Healthier Lifestyles**: Contribute to public health by providing a tool that aids in combating obesity and diet-related health issues through informed dietary decisions.
10. **Future Directions:** Investigate the potential for integrating emerging technologies to refine the accuracy and user experience of nutritional analysis applications.

**Introduction:**

In today's fast-paced world, where health and nutrition are paramount for a well-rounded lifestyle and staving off health issues linked to diet, understanding the composition of our meals is pivotal. Despite the rising awareness surrounding the importance of nutrition, many individuals still struggle to decipher the nutritional makeup of their food. This dilemma often results in suboptimal dietary choices, contributing significantly to the global surge in obesity and related health conditions. The conventional methods of nutritional analysis, such as manual tracking or simply relying on food labels, are not only difficult but also ill-suited to the demands of modern life.

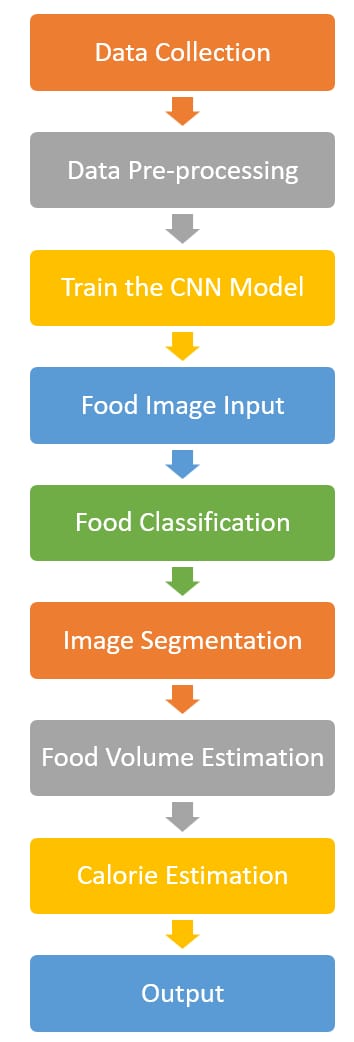
This is the key objective of our project "Intelligent Food Nutrient Analysis We’re using the strength of Artificial Intelligence, particularly the kind that can understand images, to build a system that’s easy to use and gives you an estimate of the calories in the food item. We’re aiming to create a solution that’s simple and fast. This way, people can easily make choices about their diet that are good for their health

Our journey commenced with a thorough exploration of cutting-edge research into methods for determining calorie content in food. This exploration yielded a plethora of fresh ideas and techniques to experiment with. Subsequently, we embarked on a quest to amass a diverse array of food images from various online sources and academic studies to train our AI model. Notably, we introduced the use of a thumb in these images to aid in estimating food portion sizes, a critical component in calorie calculation.

Following this, we assembled a specialized AI model known as a Convolutional Neural Network (CNN). This model comprises multiple layers that synergistically function to identify food items in images. Employing a range of sophisticated techniques, the CNN adeptly identifies features, navigates complex patterns, and compresses images to facilitate comprehension. Ultimately, the AI distinguishes between different types of food and quantifies their amounts, thus enabling accurate calorie estimation.

Our project exemplifies the transformative potential of AI in reshaping our approach to food and nutrition. By simplifying the process of selecting healthier foods, we strive to contribute to the creation of a healthier global populace. Continuously refining our system and addressing any encountered challenges, we remain steadfast in our commitment to leveraging AI to enhance public health and combat diseases stemming from poor dietary choices.

**Proposed Work:**



**Architecture Diagram for Fruit Calorie Estimation**

**Methodology for Calorie Estimation from Fruit Images**

**1. Image Collection and Preprocessing:** The initial phase involves gathering a diverse dataset of food images. These images are then pre-processed to facilitate efficient analysis. The preprocessing steps include:

* **Resizing** the images to a uniform dimension to ensure consistency.
* **Normalization** of pixel values to aid in the convergence of the neural network during training.

**2. Convolutional Neural Network (CNN) for Fruit Detection:** A CNN is employed to classify the fruit present in the images. The CNN model has the following layers:

1. **Input Layer:** Input layer takes the food image as input.
2. **Convolutional Layers:** Stack multiple convolutional layers to extract features from the input image. Each convolutional layer applies a set of filters to detect different patterns and features in the image.
3. **Activation Functions:** Apply activation function ReLU (Rectified Linear Unit) after each convolutional layer to introduce non-linearity and make the model more expressive.
4. **Pooling Layers:** We used pooling layers (MaxPooling) to reduce the spatial dimensions of the feature maps while retaining important information.
5. **Flattening Layer:** Then we flattened the output from the last convolutional layer into a 1-dimensional vector to feed into the fully connected layers.
6. **Fully Connected Layers:** Then we added one or more fully connected layers to learn complex patterns in the feature representations obtained from the convolutional layers.
7. **Output Layer:** Finally, the output layer predicted the probability distribution over different classes of food items. We used softmax activation for multi-class classification.

**3. Image Segmentation:** To isolate the food item from the background, image segmentation techniques are applied:

* Conversion of the image to **grayscale** to reduce complexity.
* Application of a **median blur** to smooth the image, reducing noise and details.
* Utilization of **adaptive thresholding** to distinguish the food item based on pixel intensity variations.
* Identification and extraction of the **largest contour**, which is presumed to be the food item.

**4. HSV Color Space Conversion:** The segmented image is converted to the HSV (Hue, Saturation, Value) color space to enhance the distinction between the food item and other objects such as plates or human skin:

* **Hue** represents the color type.
* **Saturation** indicates the vibrancy of the color.
* **Value** reflects the brightness of the color.

**5. Volume and Area Calculation:** The pixel area of the fruit and thumb is calculated using the contours obtained from the segmentation process. 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. The volume is then estimated based on the assumption that the food item is either spherical or cylindrical:

* **Area calculation** is performed by summing the areas of the pixels within the food item’s contour. We already know area of thumb, which is 5x2.3 cm². So, area per pixel is actual thumb area divided by pixel area of thumb.

From this actual food area can be calculated as: pixel area of food \* area per pixel.

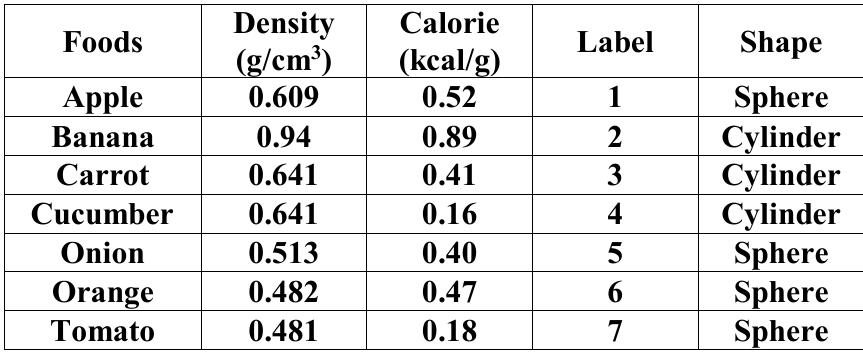
* **Volume estimation** involves applying geometric formulas such as

𝑉𝑠𝑝ℎ𝑒𝑟𝑒=4/3 𝜋𝑟3

𝑉𝑐𝑦𝑙𝑖𝑛𝑑𝑒𝑟=𝜋𝑟2ℎ

We calculated the values of r and h, and used these formulae to estimate the volume.

**6. Calorie Estimation:** The calorie content is estimated using the volume and a predefined dictionary containing the density and calorie information for various food types:



* **Mass estimation** is done by multiplying the volume by the food’s density.
* **Calorie calculation** is achieved by multiplying the mass by the calorie density, yielding the total calories and calories per 100 grams.

**7. Model Training and Prediction:** The CNN model is trained with labelled food images. The training process involves:

* **Feeding the pre-processed images** into the network.
* **Adjusting the model weights** based on the error between the predicted and actual labels using backpropagation.
* **Optimizing the model** with algorithms like Adam to minimize the loss function.

**8. Visualization and Display:** The trained model is then used to predict the type of food in new images. The results are displayed using a visualization library that presents:

* **The predicted food type** alongside the image.
* **Calorie content** of the food item, both total and per 100 grams.

This comprehensive methodology integrates advanced image processing techniques with machine learning algorithms to accurately estimate the calorie content of food items from images. The process is designed to be automated and scalable, allowing for the application to new images for consistent calorie estimation.

**Implementation:**

#image segmentation

import cv2

import numpy as np

def getAreaOfFood(img1):

img2 = cv2.cvtColor(img1, cv2.COLOR\_BGR2RGB)

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)

# 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)

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

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

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

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

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

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

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

#convert to hsv to detect and remove skin pixels

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

skin = cv2.inRange(hsv\_img, np.array([0,10,60]), np.array([10,120,255]))

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) #binary of fruit

#erode before finding contours

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

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

#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)

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)

#dilate now

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)

#find area of fruit

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)

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

skin2 = skin - mask\_fruit2

#erode before finding contours

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

# Calorie Calculation

import cv2

import numpy as np

# here we're using density, calorie per 100 gm of the fruit.

#density - gram / cm^3,

# Apple - 1, Banana - 2, Carrot - 3, Cucumber - 4, Onion - 5, Orange - 6, tomato - 7

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 }

#kcal

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 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 == 6 or label == 7 : #sphere - apple, onion, orange, tomato

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: #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

return volume

def calories(result,img):

img\_path =img

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("\nvolume of fruit(cm^3): ",fruit\_volumes,"\ncalories in 100 grams of this fruit(kcal): ",fruit\_calories\_100grams,"\nmass of fruit(gm): ",fruit\_mass,"\ntotal calories in the fruit(kcal): ",fruit\_calories)

return fruit\_calories

# Convolution Neural Network for Fruit Classification

import tensorflow as tf

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

input\_shape = (IMG\_SIZE, IMG\_SIZE, 3)

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input\_shape=input\_shape))

model.add(tf.keras.layers.MaxPooling2D((2, 2)))

model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))

model.add(tf.keras.layers.MaxPooling2D((2, 2)))

model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))

model.add(tf.keras.layers.MaxPooling2D((2, 2)))

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

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

model.add(tf.keras.layers.Dropout(0.5))

model.add(tf.keras.layers.Dense(no\_of\_fruits, activation='softmax'))

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

return model

# Training the CNN model

import numpy as np

import os # dealing with directories

from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.

import glob

import cv2

# Get the current working directory where the notebook resides

path = 'images'

IMG\_SIZE = 400

LR = 1e-3

MODEL\_NAME = 'Fruits\_dectector-{}-{}.model'.format(LR, '5conv-basic')

no\_of\_fruits = 7

percentage = 0.3

no\_of\_images = 100

def create\_train\_data(path):

images = []

labels = []

folders = os.listdir(path)[0:no\_of\_fruits]

for i in range(len(folders)):

label = [0 for i in range(no\_of\_fruits)]

label[i] = 1

print(folders[i])

k = 0

for j in glob.glob(os.path.join(path, folders[i], '\*.jpg')):

if k == no\_of\_images:

break

k += 1

img = cv2.imread(j)

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

images.append(np.array(img))

labels.append(np.array(label))

np.save('images\_{}\_{}\_{}.npz'.format(no\_of\_fruits, no\_of\_images, IMG\_SIZE), images)

np.save('labels\_{}\_{}\_{}.npz'.format(no\_of\_fruits, no\_of\_images, IMG\_SIZE), labels)

training\_data = list(zip(images, labels))

shuffle(training\_data)

return training\_data, folders

# Call the create\_train\_data function with the updated path

training\_data, labels = create\_train\_data(path)

# Split the training data into training and testing sets

size = int(len(training\_data) \* percentage)

train = training\_data[:-size]

test = training\_data[-size:]

X = np.array([i[0] for i in train]).reshape(-1, IMG\_SIZE, IMG\_SIZE, 3)

Y = [i[1] for i in train]

Y = np.array(Y)

test\_x = np.array([i[0] for i in test]).reshape(-1, IMG\_SIZE, IMG\_SIZE, 3)

test\_y = [i[1] for i in test]

test\_y = np.array(test\_y)

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

# Train the model with the training data

history = model.fit(X, Y, epochs=10, validation\_data=(test\_x, test\_y))

# model.fit(X, Y, epochs=10, validation\_data=(test\_x, test\_y), snapshot\_step=500, show\_metric=True, run\_id=MODEL\_NAME)

# Save the trained model

model\_save\_at=os.path.join("model",MODEL\_NAME)

model.save(model\_save\_at)

print("Model Save At",model\_save\_at)

import matplotlib.pyplot as plt

# Plot training & validation accuracy values

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

# Plot training & validation loss values

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.tight\_layout()

plt.show()

# demo

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

IMG\_SIZE = 400

LR = 1e-3

no\_of\_fruits = 7

MODEL\_NAME = 'Fruits\_dectector-{}-{}.model'.format(LR, '5conv-basic')

model\_path = os.path.join("model", MODEL\_NAME) # Save the model in the "model" directory

# Load the saved model

loaded\_model = tf.keras.models.load\_model(model\_path)

# Add labels

labels = ['Apple', 'Banana', 'Carrot', 'Cucumber', 'Onion', 'Orange', 'Tomato']

# Load and preprocess the test image

test\_data = '7.jpg'

img = cv2.imread(test\_data)

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

img\_processed = np.expand\_dims(img\_resized, axis=0)

# Make predictions

model\_out = loaded\_model.predict(img\_processed)

result = np.argmax(model\_out)

name = labels[result]

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

# Display the results

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

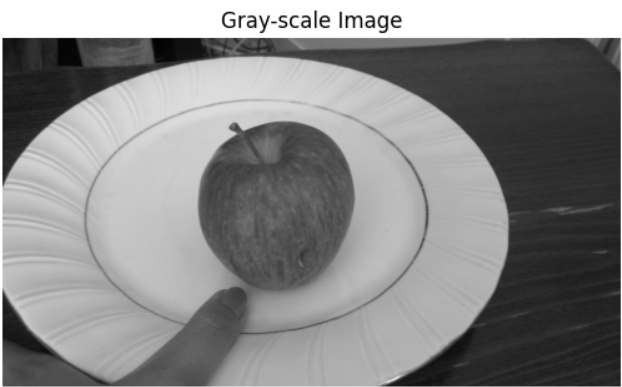
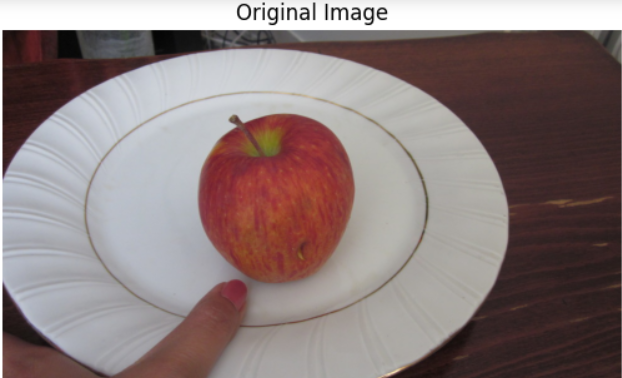
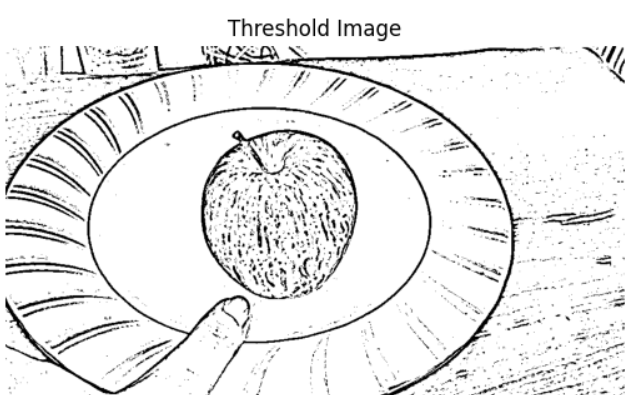
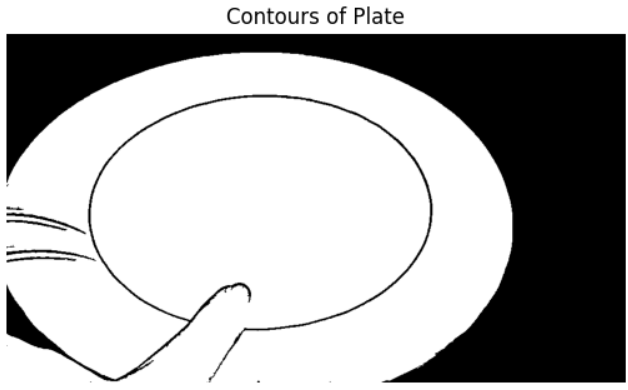
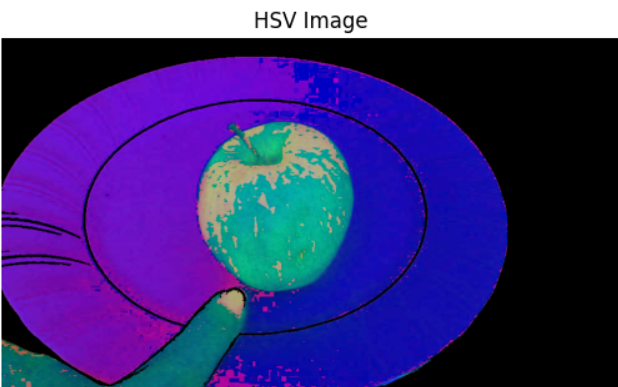
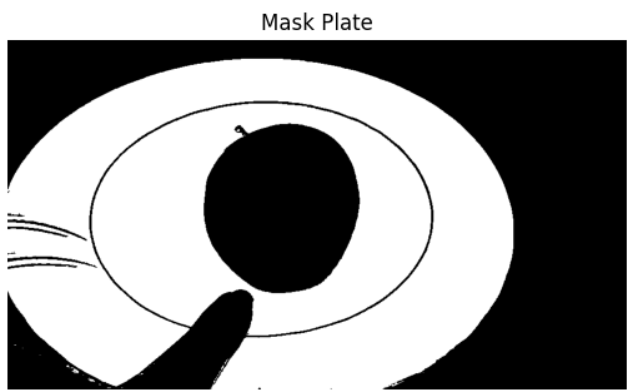
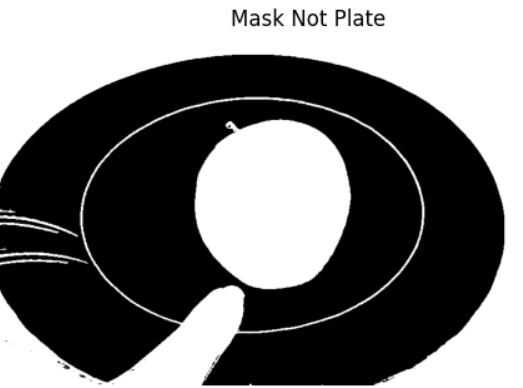
plt.title('{} ({} kcal)'.format(name, cal))

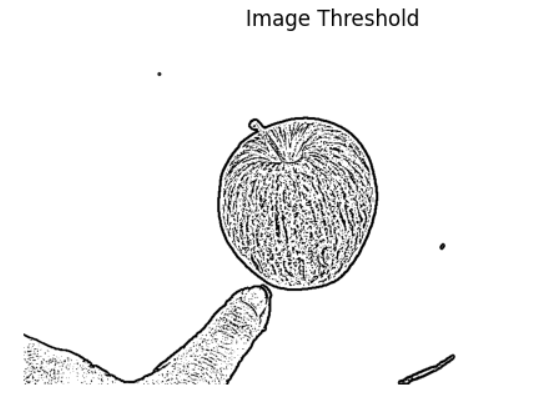
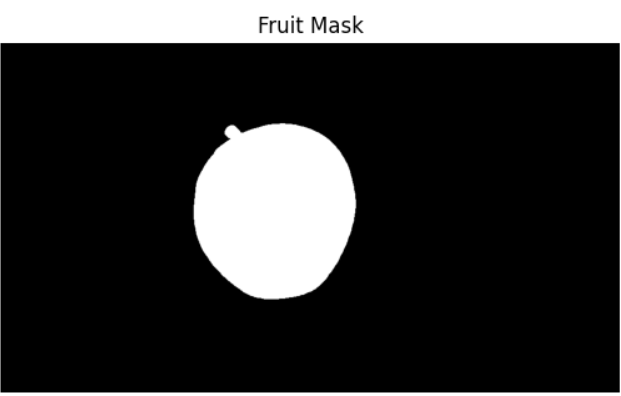
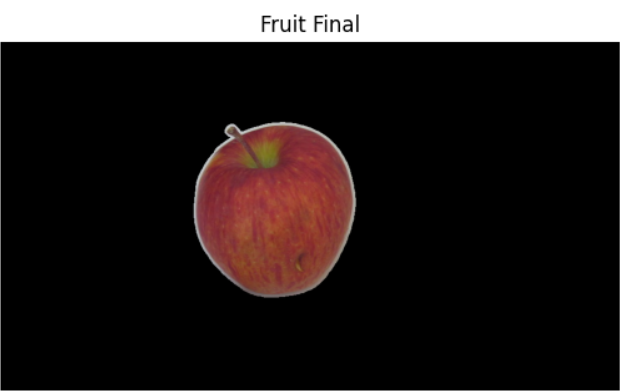
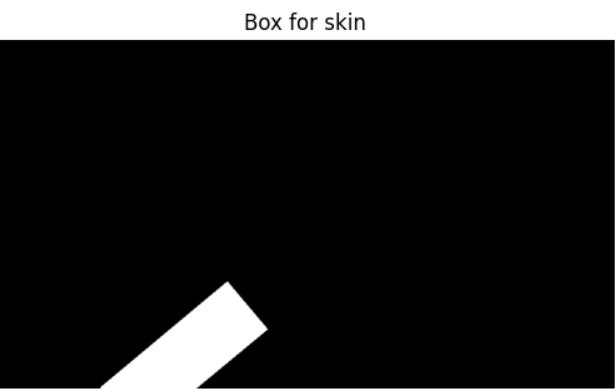
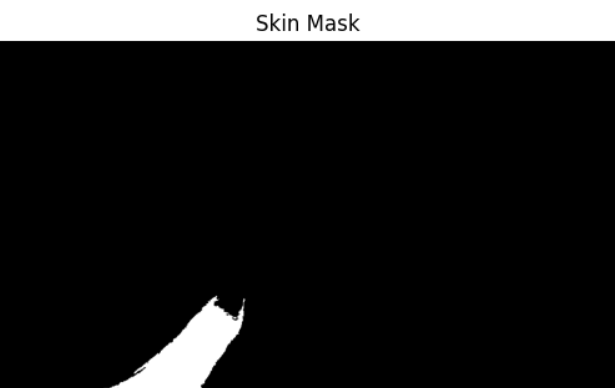
plt.axis('off')

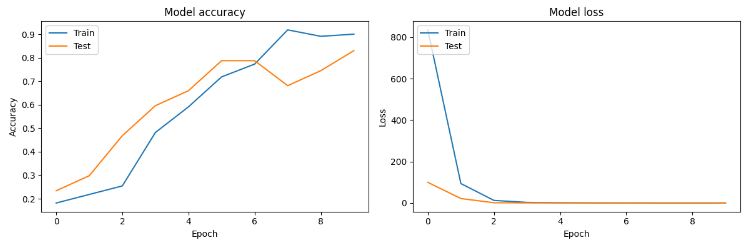
plt.show()

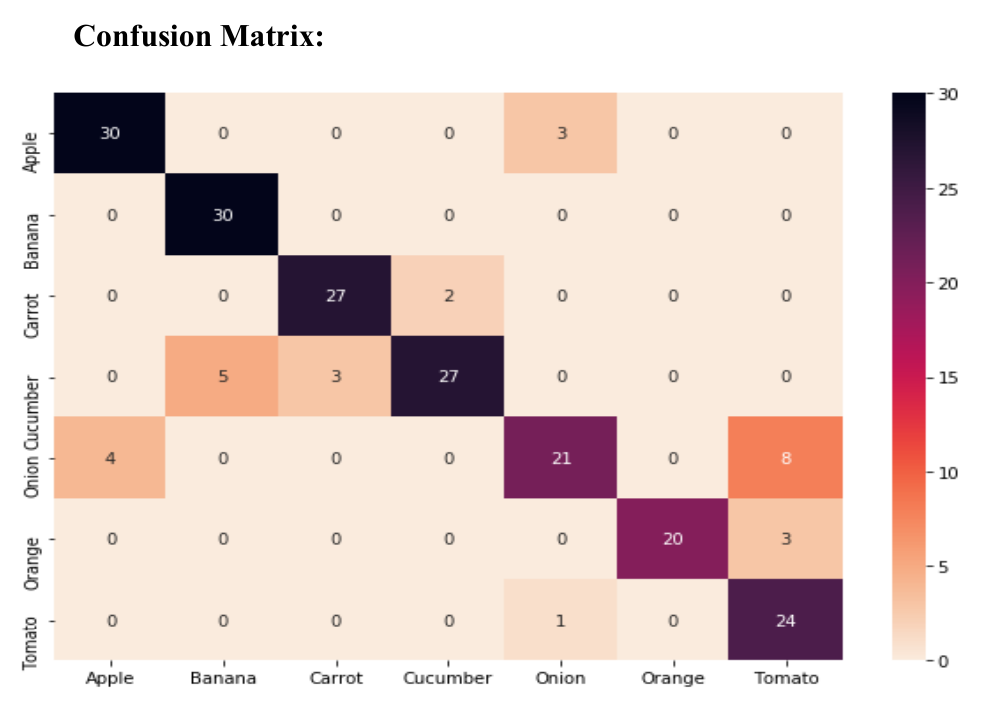
**Results:**

*Image Segmentation:*

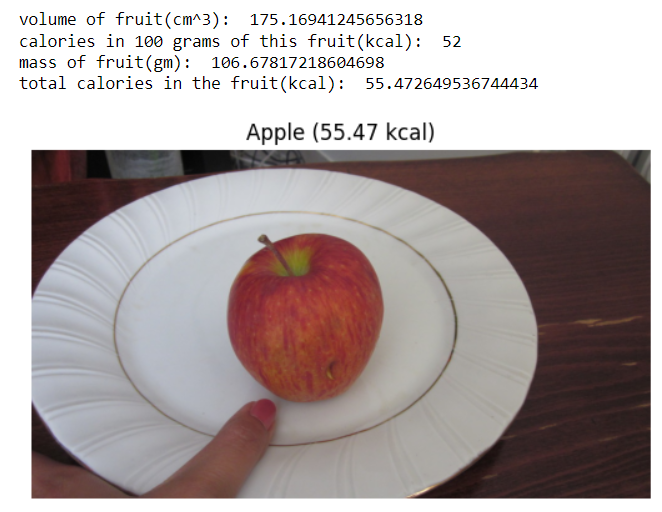
     

*Training the CNN Model:*  


**

*Demo:*



**Conclusion:**

Our model represents a significant step forward in leveraging AI technology to empower individuals to make informed and healthier dietary decisions. By combining image recognition, volume estimation, and nutritional analysis, our system offers a comprehensive and user-friendly solution to the challenge of assessing the nutritional content of food items accurately. Through further research and development, we can unlock the full potential of AI-driven solutions in promoting public health and combating diet-related diseases.