

# **ANALYSIS OF CONVOLUTIONAL NEURAL NETWORK ON INDIAN FOOD CLASSIFICATION AND ESTIMATION OF CALORIES**

*Report submitted to SASTRA Deemed to be University  
As per the requirement for the course*

## **CSE300: MINI PROJECT**

*Submitted by*

**Aravind P**

**(125003425, B. Tech Computer Science and Engineering)**

**Vigneswar A**

**(125003455, B. Tech Computer Science and Engineering)**

**Goutham Karthick S**

**(125003491, B. Tech Computer Science and Engineering)**

**May 2024**



**SASTRA**  
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION  
**DEEMED TO BE UNIVERSITY**  
(U/S 3 of the UGC Act, 1956)



**THINK MERIT | THINK TRANSPARENCY | THINK SASTRA**

**SCHOOL OF COMPUTING**

**THANJAVUR, TAMIL NADU, INDIA – 613 401**



# SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

## SCHOOL OF COMPUTING

THANJAVUR – 613 401

### Bonafide Certificate

This is to certify that the report titled “**Analysis of Convolutional Neural Networks on Indian food detection and estimation of calories**” submitted as a requirement for the course, **CSE300 : MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. Aravind P (125003425., B. Tech Computer Science and Engineering)**, **Mr. Vigneswar A (125003455, B. Tech Computer Science and Engineering)** and **Mr. Goutham Karthick S (125003491, B. Tech Computer Science and Engineering)** during the academic year 2022-23, in the School of Computing, under my supervision.

**Signature of Project Supervisor :**

**Name with Affiliation :**

**Date :**

Mini Project *Viva voice* held on \_\_\_\_\_

**Examiner 1**

**Examiner 2**

## ACKNOWLEDGEMENTS

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K. Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Algeswaran**, Associate Dean, Students Welfare

Our guide **Mrs. Ramya S**, Assistant Professor – III, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through this project.

## List of Figures

Fig No.	Title	Page No.
1.1	Inception V3 neural network Architecture	2
1.2	VGGNet16 neural network Architecture	3
1.3	Basic CNN Architecture	3
4.1	Model accuracy and Model loss graph for InceptionV3 model	24
4.2	Prediction of food item by InceptionV3 model	24
4.3	Confusion matrix for InceptionV3 model	25
4.4	Model accuracy and model loss graph for VGGNet16 model	25
4.5	Prediction of food item by VGGNet16 model.	26
4.6	Confusion matrix for VGGNet16	26
4.7	Model accuracy and model loss graph for Basic CNN model	27
4.8	Prediction of food item by Basic CNN model	27
4.9	Confusion matrix for Basic CNN model	28
4.10	Image before applying Otsu's segmentation technique	28
4.11	Image after applying Otsu's segmentation technique	28
4.12	Calorie Estimation of multiple foods in an image	29

## List of Tables

Table No.	Title	Page No.
4.1	Comparison of accuracy of different models	29

## Abbreviations

RGB	Red Green Blue
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Networks
Adam	Adaptive Moment estimation
OpenCV	Open Computer Vision
VGGNet	Visual Geometry Group Network

## Notations

### English Symbols (in alphabetical order)

Cal	Calorie
D	Dimension

## ABSTRACT

The aim of this project is to develop Deep Learning model using CNN and OpenCV which identifies Indian dishes and evaluate their calories. The food dataset for this project was prepared by collecting images from various online sources and subjecting them to thorough preprocessing for quality enhancement to increase the accuracy and removing irrelevant images for relevance.

An InceptionV3 model is used for classification of the foods. The model exhibited high accuracy in classifying the food by achieving 99.84% accuracy on the training set and 92.66% on the testing set. Additionally, the model has employed Otsu's thresholding algorithm to isolate different food items in the image. Using segmented images, the volume of each food item is approximately calculated. Then the amount of calorie in each food item is calculated using the volume of each food item. This approach holds significant promise for addressing dietary concerns and promoting healthier eating habits within the context of India's prevailing health challenges.

**KEY WORDS:** Convolutional Neural Network, Optimization, Image Segmentation, OpenCV, Machine Learning, Calorie detection.

## Table of Contents

<b>Title</b>	<b>Page No.</b>
Bonafide Certificate	ii
Acknowledgements	iii
List of Figures and Tables	iv
Abbreviations	v
Notations	vi
Abstract	vii
1. Summary of the base paper	1
2. Merits and Demerits of the base paper	5
3. Source Code	7
4. Output Snapshots	24
5. Conclusion and Future Plans	30
6. References	31
7. Appendix -Base Paper	32



# CHAPTER 1

## SUMMARY OF THE BASE PAPER

<b>Title</b>	:	Analysis of Convolutional Neural Networks on Indian food detection and estimation of calories
<b>Publisher</b>	:	ScienceDirect
<b>Year</b>	:	2022
<b>Journal name</b>	:	ScienceDirect
<b>DOI</b>	:	10.1016/j.matpr.2022.03.122
<b>Base paper URL</b>	:	<a href="https://www.sciencedirect.com/science/article/pii/S2214785322014444?via%3Dihub">https://www.sciencedirect.com/science/article/pii/S2214785322014444?via%3Dihub</a>

The main contributions of the base paper are:

- Collecting the images of Indian cuisine foods and preparing a dataset.
- Training a food classification model to classify the interested food items.
- Applying image segmentation techniques to the image to isolate different food items.
- Calculating calorie of each food from given parameters.

Our method consists of 4 major steps:

### 1.1 DATASET PREPARATION

A dataset has been created by collecting images of Indian food sources from various online platforms. The dataset underwent thorough preprocessing techniques to ensure quality and relevance including the removal of low-quality images and irrelevant images. Finally, the food dataset consists of 11 classes, each consisting of 1000 images.

### 1.2. FOOD DETECTION

#### 1.2.1 Data preprocessing

The dataset is divided into training, testing, and validation dataset in the percentage of 56.25%, 25%, and 18.75%, respectively. Before training the training dataset, the images are preprocessed by normalizing them. Normalization of the image is done by scaling the pixel values between 0 and 1. This can be achieved by dividing the pixel value by the maximum value (255). The height and width of the image are 299.

#### 1.2.1. Model training

For detecting the food in the given image, Convolutional neural network models are

employed. In this paper, InceptionV3, VGGNet16 and the basic CNN model are utilized. The preprocessed dataset is given as input to the model.

For InceptionV3 and VGGNet16, the customized top layers are built. The customized top layers include a global average pooling layer, a dense layer with the ReLU activation function, a dropout layer and a dense layer with the softmax activation function.

The global average pooling layer performs downsampling by calculating mean of the height and width dimensions of the input. The dense layer is a layer of neurons which receive input from the neurons of the previous layer. Based on input received, neuron computes weight and passes it as input to the next dense layer.

The ReLU activation function gets computed weight as input. If the input is negative then it gives output 0; otherwise, it gives the same weight as output. This enables the neural network to introduce non-linearity, which allows it to learn more complex patterns.

The dropout layer is used to drop a random set of neurons. By doing this, the network becomes less sensitive to specific weights and is forced to learn more features from the images. This helps to prevent overfitting and enhances the ability of the model to generalize unseen data.

A dense layer with the softmax function acts as an output layer. In this layer, the softmax function takes unnormalized predictions/logits from the above layer as input. Logits are scores assigned to each class in the network. The SoftMax functions convert these logits to probabilities representing the likelihood of each class. This work makes the SoftMax function to be suitable for multi-classification model.

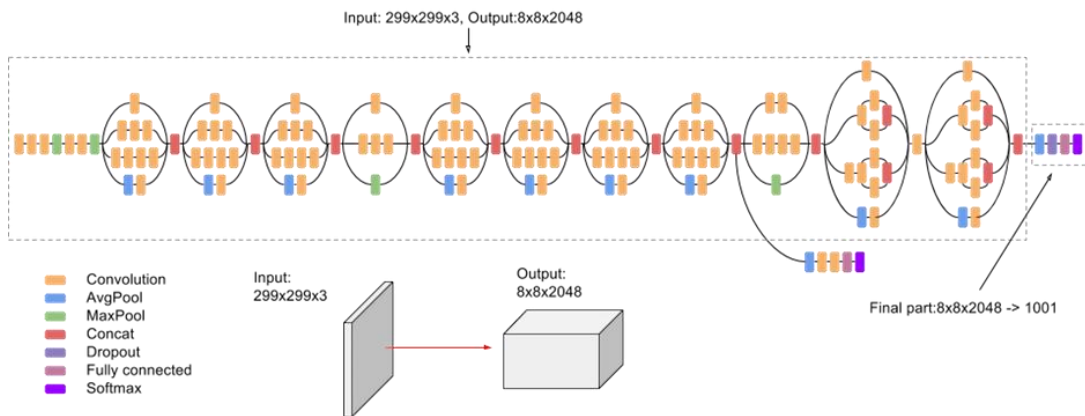


Fig 1.1 InceptionV3 Architecture

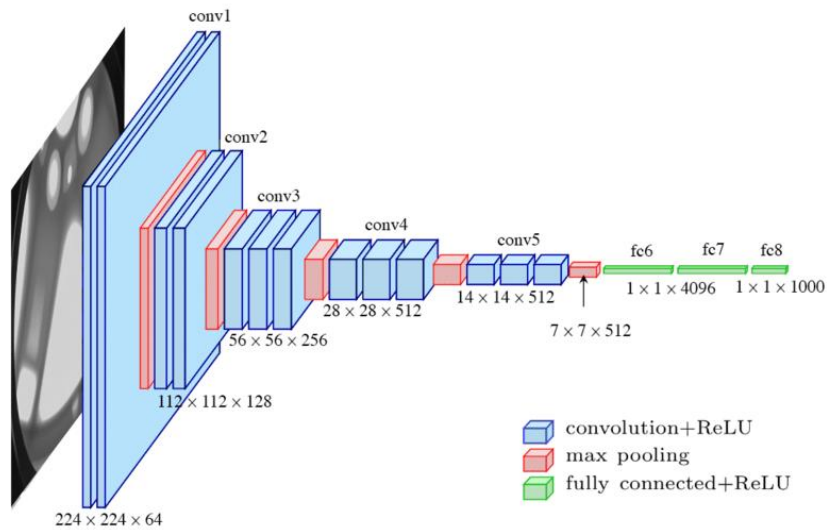


Fig 1.2 VGGNet16 Architecture

The Basic CNN model consists of 7 layers which includes 3 convolutional layers, 2 max pooling layers, 1 flatten layer and 2 dense layers.

The convolutional layer applies a set of filters or kernels to input data. A filter is defined as the small matrix of weights. Each filter is moved across the image. At each position, the filter is applied to the local region of the input data which produces a single-valued output. The ReLU function is applied to output of this layer to introduce non-linearity.

The max pooling layer is used to reduce the spatial dimensions of the input features. This is achieved by taking the maximum value within each local region. Thus, the max pooling layer preserves significantly important features and also reduce the computational complexity.

The flatten layer is used to transform the multi-dimensional input to 1D array. The output of this layer is given as input to the dense layer with softmax activation function. So that, the predictions are converted into probabilities of likelihood of each class.

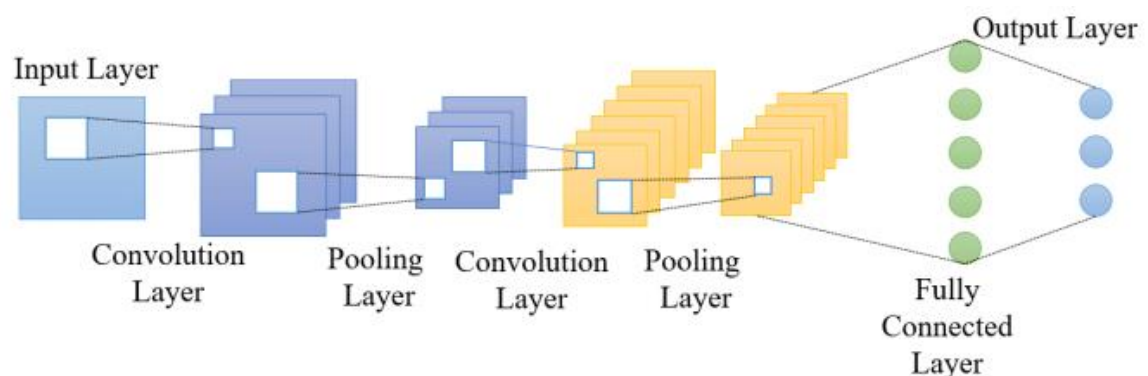


Fig 1.4 Basic CNN Architecture

After this, the model is compiled using Adam optimizer to configure the learning process before training. The Adam optimizer adjusts the weights and biases by computing adaptive learning rates using mean and variance which helps to minimize the loss function during the training process. Once the training is completed, the training and validation accuracy of each model is evaluated and plotted as graphs. Then the model is used to evaluate the test dataset and test accuracy is computed. After this model gets ready for the food classification.

### 1.3. IMAGE SEGMENTATION

Image segmentation is a process of partitioning an image into multiple segments which are based on colour, texture and intensity. Image segmentation is done using OpenCV. First the image is converted into grayscale. Then the threshold of the image is calculated using Otsu's thresholding algorithm. Otsu's thresholding algorithm searches for the threshold which minimizes the intra-class variance. Intra-class variance is defined as weighted sum of variances of the two classes.

Inverted threshold also found by applying bitwise not operation to the threshold. Inverted threshold is used to enhance the segmentation process in the image containing various food items with different intensities. The contours are drawn using threshold and inverted threshold separately then they are merged. The bounding box algorithm is employed to eliminate unnecessary contours. Then the contours of the various food are saved as image in the directory.

### 1.5. CALORIE PREDICTION

After applying image segmentation to the image, volume of the food is estimated from the image using the formula

$$volume = area * conversion\ factor * thickness$$

where area is the area of the largest contour (food item) in the image, conversion factor is used to convert pixel to centimeters and thickness is the height of the image.

Then the calorie of the food is estimated by

$$calorie = volume * density * caloric\ density$$

where volume is volume of the food in the image, density is the density of the given food. The density and caloric density of the foods is taken from the internet. The calculated calorie of the food is represented in units of Cal.

## CHAPTER 2

### MERITS AND DEMERITS OF THE BASE PAPER

#### LITERATURE SURVEY:

There is various paper proposed for image classification, image segmentation. Some of them are listed below mentioning the merits and demerits of the proposed method over each of the existing methods.

- **“An Instance Segmentation Approach to Food Calorie Estimation Using Mask R-CNN”** by Parth Poply et al. This paper proposes a method to estimate calories using OpenCV, and Mask-R-CNN model is used for image segmentation. The disadvantage of this paper is that the mask-R-CNN require more resources for training and inference which limits its scalability and utilization in real-time applications.
- **“Food Recognition and Calorie Measurement using Image Processing and Convolutional Neural Network”** by V. Hemalatha Reddy et al. This paper proposes a caloric value estimation app. It requires the user to feed the image to the application with the food, which in turn calculate the caloric value. Additionally, it also displays the weekly statistics of number of calories consumed by users and how to prevent obesity. The potential disadvantage of the paper is that accuracy of the calorie estimation heavily depends on the quality of the image such as lighting conditions, camera angle etc.,
- **“Nutrition5k: Towards Automatic Nutritional Understanding of Generic Food”** by Quin Thames et al. A primary dataset of actual real food items with their in-depth high-quality images and acme precision nutrition content interpretation is developed. The disadvantage of this paper is that the model needs more processing time required to analyse the nutritional content of the food.
- **“CNN-Based Food Image Segmentation Without Pixel-Wise Annotation”** by Wataru Shimoda et al. This paper proposes a method for image segmentation that eliminates pixel wise interpretation by generating food clusters using selective search and bounding box clustering. This is further enhanced by back propagation based on saliency map estimation with a CNN model fine-tuned with the UEC-FOOD100 dataset. The potential disadvantage of this paper is the complexity of these algorithms which takes longer to process the image, and the UEC-FOOD100 dataset contains only a limited number of foods. So, this model is not suitable for diverse food items.

## **MERITS AND DEMERITS**

### **Merits:**

- The paper uses OpenCV for image segmentation, which is optimized for speed and efficiency.
- InceptionV3 has been pre-trained on large datasets, which enables effective transfer learning where the model can be fine-tuned on datasets to achieve high performance.
- Otsu's thresholding technique automatically selects the optimal threshold value without requiring parameter tuning.

### **Demerits:**

- A large dataset that contains images with minimum noise is needed as input for the model to exhibit good accuracy.
- Otsu's thresholding algorithm is sensitive to noise as it assumes only foreground and background with distinct intensity distributions. If the image contains noise, then it may lead to poor segmentation results.

## CHAPTER 3

### SOURCE CODE

#### 3.1. DATA PREPROCESSING

delete\_duplicate.py

```
import os
from PIL import Image
import imagehash
```

```
def find_duplicate_images(directory):
    hash_dict = {}
    duplicates = []
    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)
        if os.path.isfile(file_path):
            with open(file_path, 'rb') as f:
                try:
                    img_hash = imagehash.average_hash(Image.open(f))
                    if img_hash in hash_dict:
                        hash_dict[img_hash].append(file_path)
                    else:
                        hash_dict[img_hash] = [file_path]
                except Exception as e:
                    print(f"Error processing {file_path}: {e}")
    for img_hash, files in hash_dict.items():
        if len(files) > 1:
            duplicates.append(files)
    return duplicates
```

```
def delete_duplicates(duplicates):
    for group in duplicates:
        for file_path in group[1:]:
            os.remove(file_path)
            print(f"Deleted {file_path}")
```

```

if __name__ == "__main__":
    duplicates = find_duplicate_images(r"Final_Data_Set\paruppu-sambar-rice")
    if duplicates:
        print("Duplicate images found:")
        for group in duplicates:
            print("\n".join(group))
        delete = input("Do you want to delete the duplicates? (yes/no): ").lower()
        if delete == 'y':
            delete_duplicates(duplicates)
            print("Duplicates deleted successfully.")
        else:
            print("Duplicates not deleted.")
    else:
        print("No duplicate images found.")

```

### 3.2. MODEL TRAINING

#### InceptionV3\_model\_training.ipynb

```

import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

```

```

train_data_dir = r"FoodDataSet\train"
valid_data_dir = r"FoodDataSet\test"
test_data_dir = r"FoodDataSet\val"

```



```

datagen = ImageDataGenerator(
    rescale = 1./255,
)
train_generator = datagen.flow_from_directory(train_data_dir,
                                              target_size = (299, 299),
                                              batch_size = 32,
                                              class_mode = 'categorical')
valid_generator = datagen.flow_from_directory(valid_data_dir,
                                              target_size = (299, 299),
                                              batch_size = 32,
                                              class_mode = 'categorical')
test_generator = datagen.flow_from_directory(test_data_dir,
                                              target_size = (299, 299),
                                              batch_size = 32,
                                              class_mode = 'categorical')

```

```

from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3

base_model = InceptionV3(weights='imagenet', include_top=False)
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.3)(x)
predictions = Dense(11, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
optimizer=Adam(0.0001)
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

```

```

epochs = 30
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data = valid_generator
)
model_name = 'inception_v3_adam_updated.keras'
model.save(model_name, save_format='keras')

```

```
from tensorflow.keras.preprocessing import image
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

model = tf.keras.models.load_model('inception_v3_adam_updated.keras')
```

```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')
```

```
import matplotlib.pyplot as plt
def plot_training_history(history):
    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.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
```

```
plot_training_history(history)
```

```
# Store classes name to show in prediction result
class_map = train_generator.class_indices
classes = []
for key in class_map.keys():
    classes.append(key)
```

```

from tensorflow.keras.preprocessing import image
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
model = tf.keras.models.load_model('inception_v3_adam_updated.keras')
def predict_image(filename, model):
    img_ = image.load_img(filename, target_size=(299, 299))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255
    prediction = model.predict(img_processed)
    index = np.argmax(prediction)
    plt.title("Prediction - {}".format(str(classes[index]).title()), size=18, color='red')
    plt.imshow(img_array)
predict_image(r'C:\Users\prasa\OneDrive\Desktop\MiniProject\Testing_Set\paniyaaram.png',
model)

```

```

import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
data_directory = r"C:/Users/prasa/OneDrive/Desktop/MiniProject/FoodDataSet/test"
datagen = ImageDataGenerator(rescale=1./255)
batch_size = 32 # Adjust batch size as needed
subset_generator = datagen.flow_from_directory(
    directory=data_directory,
    target_size=(299, 299),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)
predictions = model.predict(subset_generator)
predicted_labels = np.argmax(predictions, axis=1)
conf_matrix = confusion_matrix(subset_generator.classes, predicted_labels)

```

```

def plot_confusion_matrix(conf_matrix, class_names):
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()

class_names = list(subset_generator.class_indices.keys())

```



```
plot_confusion_matrix(conf_matrix, class_names)
```

## VGGNet16\_model\_training.ipynb



```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam
import h5py as h5py
```




```
train_data_dir = r"FoodDataSet\train"
valid_data_dir = r"FoodDataSet\test"
test_data_dir = r"FoodDataSet\val"
```



```
datagen = ImageDataGenerator(
    rescale = 1./255,
)
train_generator = datagen.flow_from_directory(train_data_dir,
                                              target_size = (224, 224),
                                              batch_size = 32,
                                              class_mode = 'categorical')
valid_generator = datagen.flow_from_directory(valid_data_dir,
                                              target_size = (224, 224),
                                              batch_size = 32,
                                              class_mode = 'categorical')
test_generator = datagen.flow_from_directory(test_data_dir,
                                              target_size = (224, 224),
                                              batch_size = 32,
                                              class_mode = 'categorical')
```






```
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16

base_model = VGG16(weights='imagenet', include_top=False)
base_model.trainable = False
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.3)(x)
predictions = Dense(11, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
optimizer = Adam(0.0001)
model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```



```
epochs = 30
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data = valid_generator,
)
model.save('food_detection_vgg_updated.keras')
```



```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')
```

```
import matplotlib.pyplot as plt
def plot_training_history(history):
    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.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
```

```
plot_training_history(history)
```

```
# Store classes name to show in prediction result
class_map = train_generator.class_indices
classes = []
for key in class_map.keys():
    classes.append(key)
```



```
from tensorflow.keras.preprocessing import image
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
model = tf.keras.models.load_model('food_detection_vgg_updated.keras')
threshold = 0.8
def predict_image(filename, model):
    img_ = image.load_img(filename, target_size=(299, 299))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255
    prediction = model.predict(img_processed)
    index = np.argmax(prediction)
    confidence = prediction[0][index]
    print(confidence)
    if confidence >= threshold:
        plt.title("Prediction - {}".format(str(classes[index]).title()), size=18, color='red')
    else:
        plt.title("Prediction - Unknown", size=18, color='red')
```



```
predict_image(r"C:\Users\prasa\OneDrive\Desktop\MiniProject\FoodDataSet\test\ulundu_
vadai\augmented_similar_image_125_0_804.jpg", model)
```



```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
data_directory = r"C:/Users/prasa/OneDrive/Desktop/MiniProject/FoodDataSet/test"
datagen = ImageDataGenerator(rescale=1./255)
batch_size = 32 # Adjust batch size as needed
subset_generator = datagen.flow_from_directory(
    directory=data_directory,
    target_size=(299, 299),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)
predictions = model.predict(subset_generator)
predicted_labels = np.argmax(predictions, axis=1)
conf_matrix = confusion_matrix(subset_generator.classes, predicted_labels)
```

```
def plot_confusion_matrix(conf_matrix, class_names):
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()

class_names = list(subset_generator.class_indices.keys())
```

```
plot_confusion_matrix(conf_matrix, class_names)
```

## BasicCNN\_model\_training.ipynb

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

```
train_data_dir = r"FoodDataSet\train"
valid_data_dir = r"FoodDataSet\test"
test_data_dir = r"FoodDataSet\val"
```



```

datagen = ImageDataGenerator(
    rescale = 1./255,
)
train_generator = datagen.flow_from_directory(train_data_dir,
                                              target_size = (64, 64),
                                              batch_size = 32,
                                              class_mode = 'categorical')
valid_generator = datagen.flow_from_directory(valid_data_dir,
                                              target_size = (64, 64),
                                              batch_size = 32,
                                              class_mode = 'categorical')
test_generator = datagen.flow_from_directory(test_data_dir,
                                              target_size = (64, 64),
                                              batch_size = 32,
                                              class_mode = 'categorical')

```

```

def basic_cnn(input_shape, num_classes):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3,3), activation="relu", input_shape=input_shape))
    model.add(layers.MaxPooling2D((2,2)))
    model.add(layers.Conv2D(64, (3,3), activation="relu"))
    model.add(layers.MaxPooling2D((2,2)))
    model.add(layers.Conv2D(64, (3,3), activation="relu"))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.4))
    model.add(layers.Dense(64, activation="relu"))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model

input_shape = (64, 64, 3)
num_classes = 11
model = basic_cnn(input_shape, num_classes)
adam = Adam(learning_rate=0.0001)
model.compile(optimizer=adam, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

```

```

epochs = 30
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data = valid_generator,
)
model.save('food_detection_Basic_CNN.keras')

```



```
test_loss, test_accuracy = model.evaluate(test_generator)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')
```



```
import matplotlib.pyplot as plt
def plot_training_history(history):
    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.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
```



```
plot_training_history(history)
```



```
# Store classes name to show in prediction result
class_map = train_generator.class_indices
classes = []
for key in class_map.keys():
    classes.append(key)
```



```
from tensorflow.keras.preprocessing import image
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
model = tf.keras.models.load_model('food_detection_Basic_CNN.keras')
threshold = 0.8
def predict_image(filename, model):
    img_ = image.load_img(filename, target_size=(64, 64))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255
    prediction = model.predict(img_processed)
    index = np.argmax(prediction)
    confidence = prediction[0][index]
    print(confidence)
    if confidence >= threshold:
        plt.title("Prediction - {}".format(str(classes[index]).title()), size=18, color='red')
    else:
        plt.title("Prediction - Unknown", size=18, color='red')
```



```
predict_image(r"C:\Users\prasa\OneDrive\Desktop\MiniProject\FoodDataSet\test\ulundu_
vadai\augmented_similar_image_125_0_804.jpg", model)
```



```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator
data_directory = r"C:/Users/prasa/OneDrive/Desktop/MiniProject/FoodDataSet/test"
datagen = ImageDataGenerator(rescale=1./255)
batch_size = 32 # Adjust batch size as needed
subset_generator = datagen.flow_from_directory(
    directory=data_directory,
    target_size=(64, 64),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)
predictions = model.predict(subset_generator)
predicted_labels = np.argmax(predictions, axis=1)
conf_matrix = confusion_matrix(subset_generator.classes, predicted_labels)
```

```
def plot_confusion_matrix(conf_matrix, class_names):
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()

class_names = list(subset_generator.class_indices.keys())
```

```
plot_confusion_matrix(conf_matrix, class_names)
```

### 3.3. IMAGE SEGMENTATION

#### Image\_segmentation.ipynb

```
import cv2
import matplotlib.pyplot as plt
import os
```

```
orig_image = cv2.imread(r"calorie_prediction\omelette.jpeg")

gray = cv2.cvtColor(orig_image, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
thresh_inverted = cv2.bitwise_not(thresh)
contours_original, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contours_inverted, _ = cv2.findContours(thresh_inverted, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contours = contours_original + contours_inverted
contours = sorted(contours, key=cv2.contourArea, reverse=True)
output_folder = "otsu_segmentation"
os.makedirs(output_folder, exist_ok=True)
```





```
for file in os.listdir(output_folder):
    file_path = os.path.join(output_folder, file)
    os.remove(file_path)
existing_rects = []
offset = 10
```



```
for idx, contour in enumerate(contours):
    x, y, w, h = cv2.boundingRect(contour)
    if w > 0.9 * orig_image.shape[1] or h > 0.9 * orig_image.shape[0] or w < 0.15 *
orig_image.shape[1] or h < 0.15 * orig_image.shape[0]:
        continue
    overlap = False
    for rect in existing_rects:
        if (x + w + offset >= rect[0] and y + h + offset >= rect[1] and x - offset <=
rect[0] + rect[2] and y - offset <= rect[1] + rect[3]) or \
            (rect[0] + rect[2] + offset >= x and rect[1] + rect[3] + offset >= y and rect[0]
- offset <= x + w and rect[1] - offset <= y + h):
            overlap = True
            break
    if overlap:
        continue
    output_path = os.path.join(output_folder, f"cropped_{idx}.jpg")
    cv2.imwrite(output_path, orig_image[max(y - offset, 0):min(y + h + offset,
orig_image.shape[0]),max(x - offset, 0):min(x + w + offset, orig_image.shape[1])])
    existing_rects.append((x - offset, y - offset, w + 2 * offset, h + 2 * offset))
```



```
plt.imshow(thresh, cmap='gray')
plt.show()
```

### 3.4. CALORIE PREDICTION

calorie\_prediction.ipynb



```
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing import image
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```

model = tf.keras.models.load_model(r'food_recognition_inception_V3_adam.keras')
classes = ['Idli', 'Omelette', 'boiled_egg', 'chicken_fry', 'curd_rice', 'fish_fry',
'paniyaram', 'poriyal', 'puli_sadham', 'sambar-rice', 'ulundu_vadai']

```

```

def predict_image(filename):
    img_ = image.load_img(filename, target_size=(299, 299))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255
    prediction = model.predict(img_processed)
    index = np.argmax(prediction)
    return str(classes[index]).lower()

```

```

constant = {
    'boiled_egg': (156, 0.1, 2.5),
    'chicken_varuval': (246, 0.5, 2.5),
    'curd_rice': (130, 1, 2),
    'fish_varuval': (228, 0.5, 1.8),
    'idli': (135, 1.5, 0.7),
    'omelette': (123, 1.75, 0.4),
    'paniyaram': (232, 0.5, 3),
    'paruppu_sambar_saadham': (103, 0.5, 0.5),
    'poriyal': (228, 0.5, 3),
    'puli_sadham': (209, 0.5, 1.5),
    'ulundu_vadai': (177, 0.5, 3)
}

conversion_factor = 0.00001
total_calorie = 0

```

```

for file in os.listdir('otsu_segmentation'):
    image_path = os.path.join('otsu_segmentation', file)
    segmented_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    _, thresh = cv2.threshold(segmented_image, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    largest_contour = max(contours, key=cv2.contourArea)
    area = cv2.contourArea(largest_contour)
    image_class = predict_image(image_path)
    caloric_density, mass_density, thickness = constant[image_class]
    volume = area * thickness * conversion_factor
    calorie = (volume * caloric_density * mass_density)
    total_calorie += calorie
    print(f"Estimated calorie of {image_class}: {calorie:.2f} cal")

plt.imshow(cv2.cvtColor(orig_image, cv2.COLOR_BGR2RGB))
print(f"Total calorie: {total_calorie:.2f} cal")

```

## CHAPTER 4

### OUTPUT SNAPSHOTS

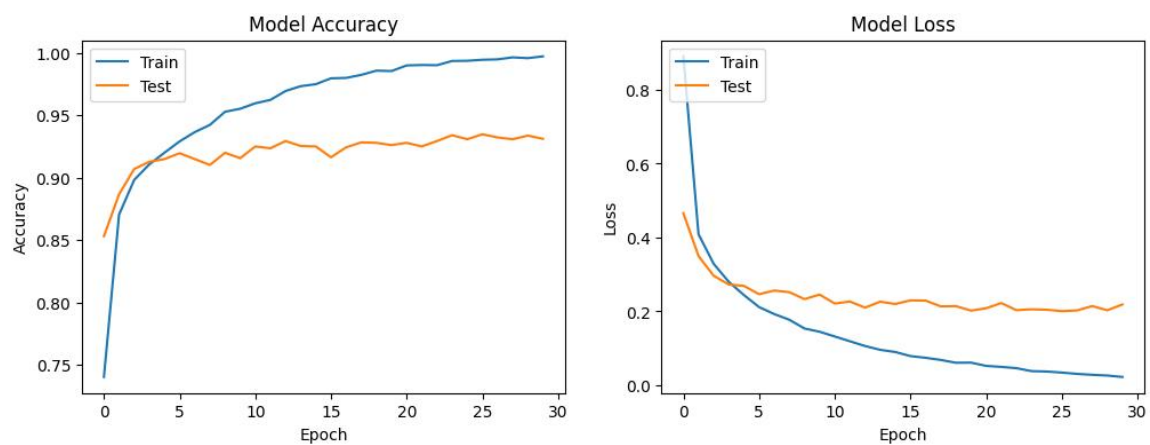


Fig 4.1. Model Accuracy and Model Loss for InceptionV3 model

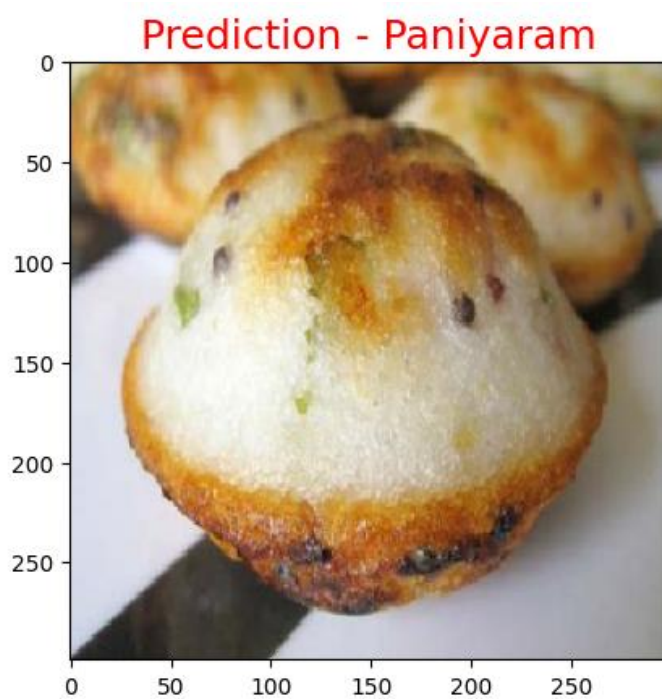


Fig 4.2. Prediction of Food item by InceptionV3



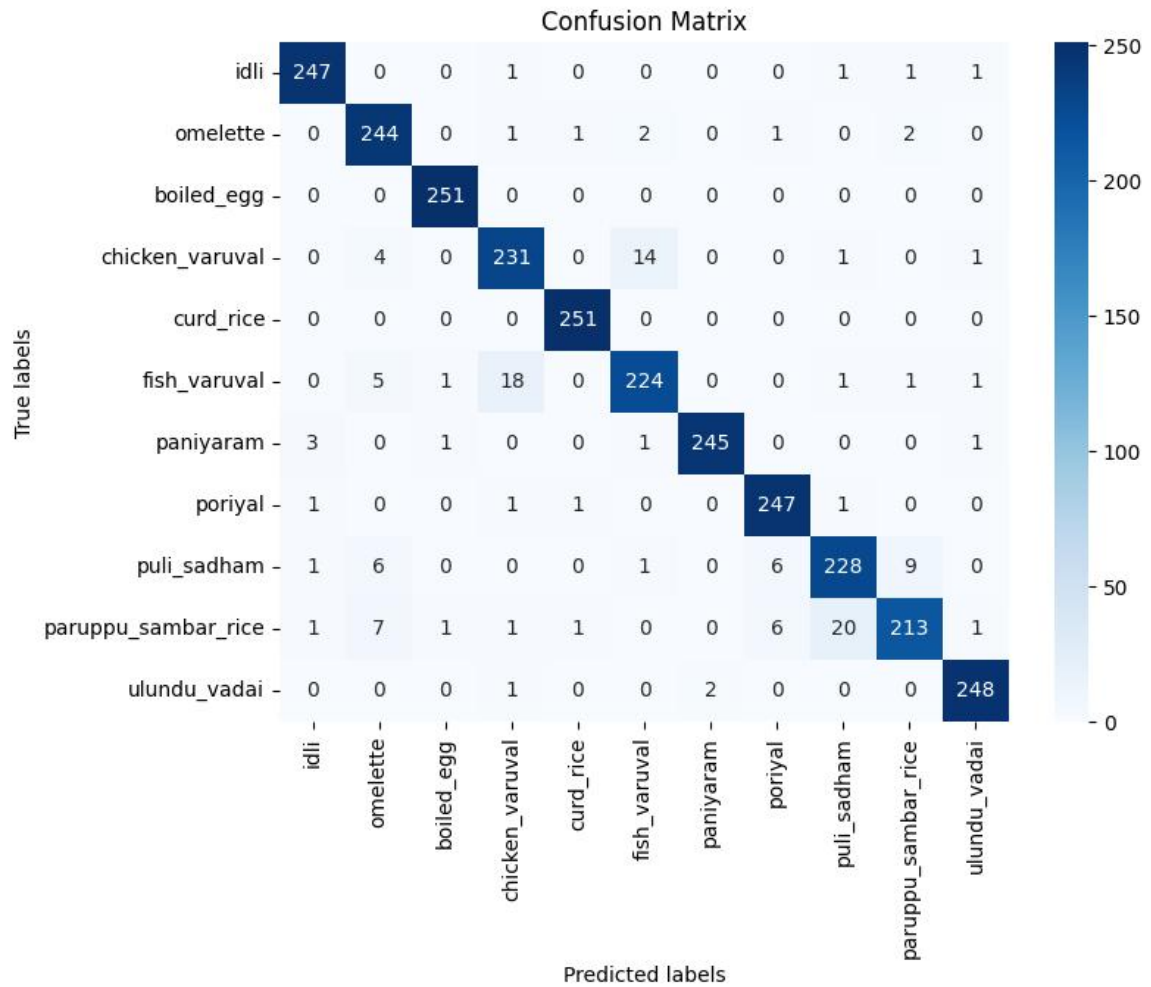


Fig 4.3 Confusion matrix for InceptionV3 model

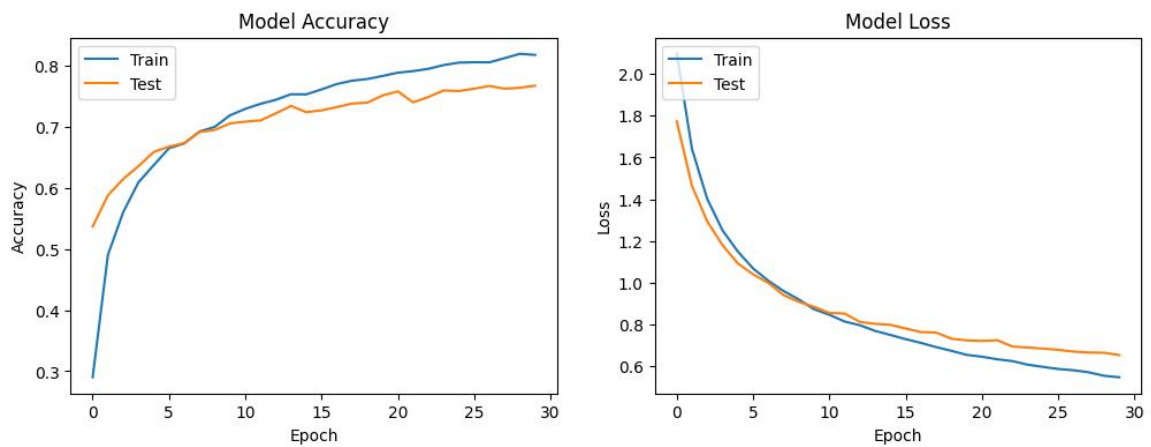


Fig 4.4. Model Accuracy and Model Loss for VGGNet16 model



Fig 4.5. Prediction of Food item by VGGNet16

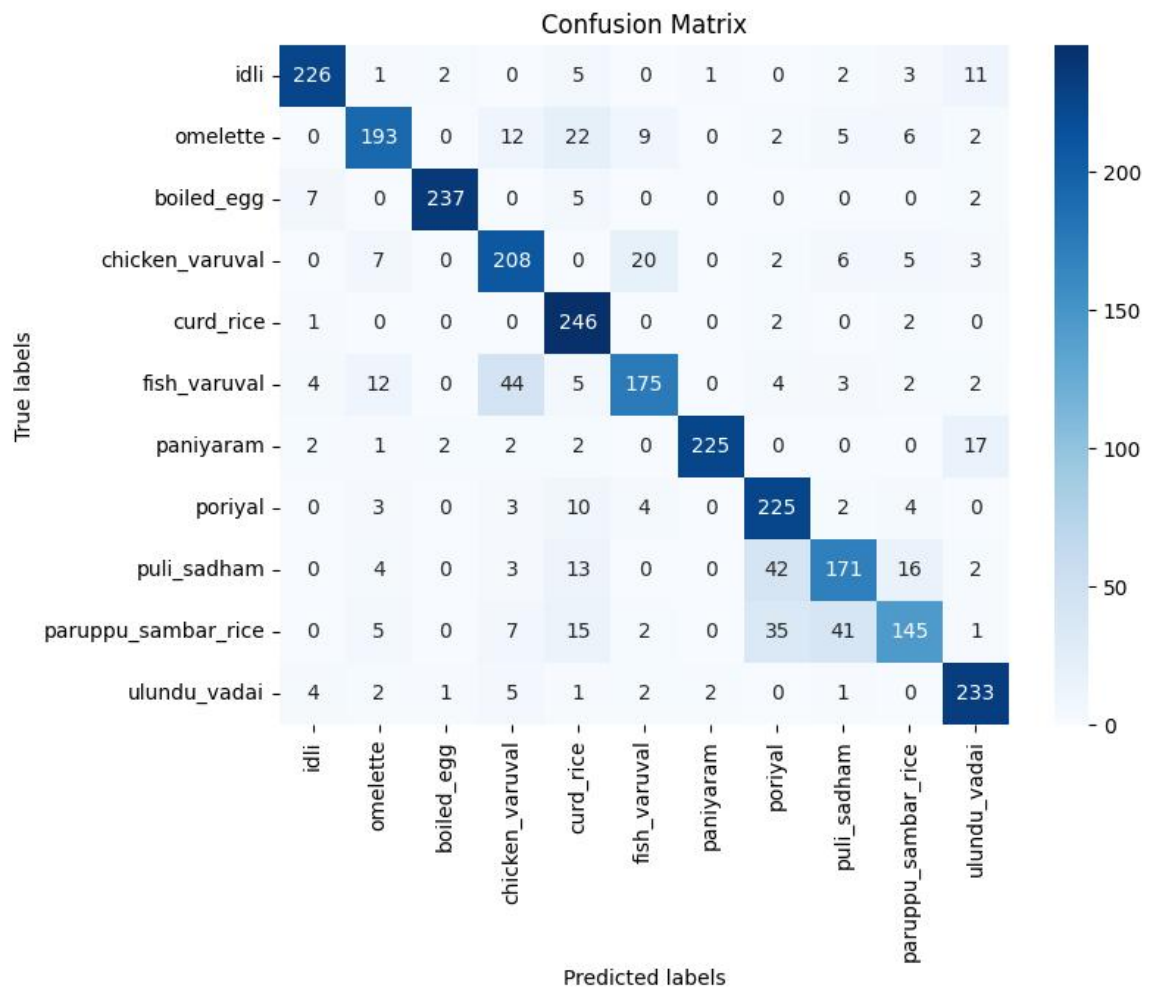


Fig 4.6 Confusion matrix for VGGNet16 model.

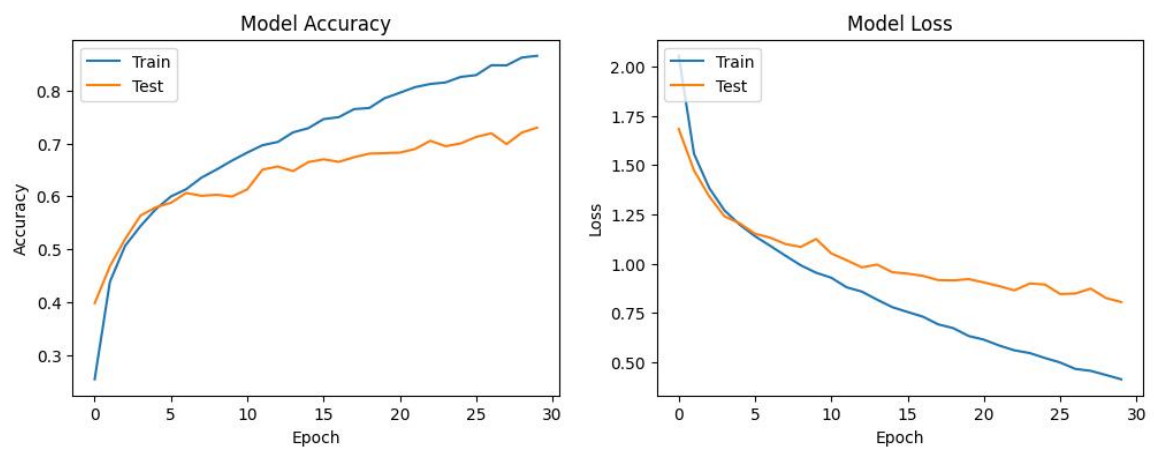


Fig 4.7. Model Accuracy and Model Loss for Basic CNN model

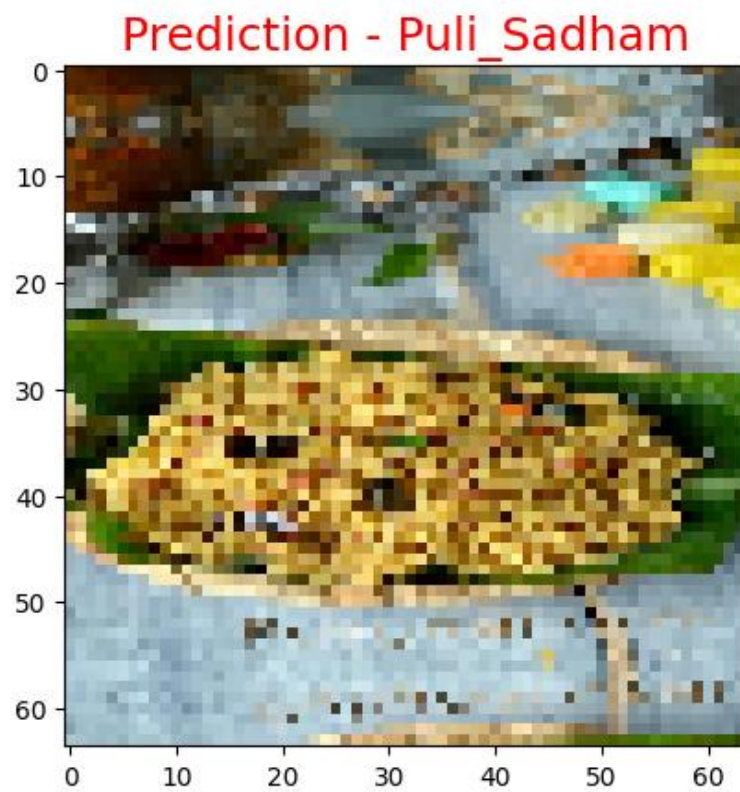


Fig 4.8. Prediction of food by Basic CNN model

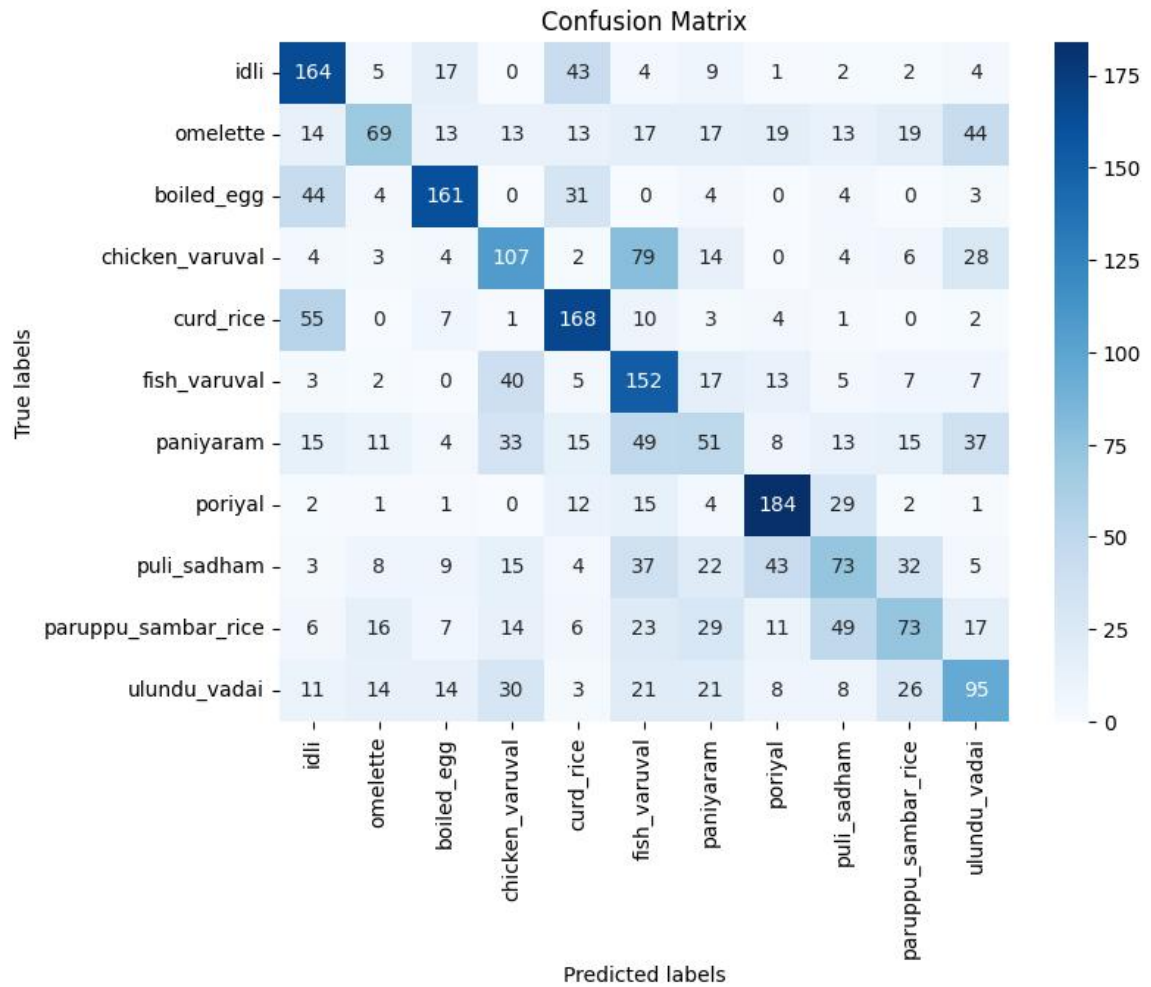


Fig 4.9 Confusion Matrix for Basic CNN model



Fig 4.10. Image before applying Otsu's Segmentation technique

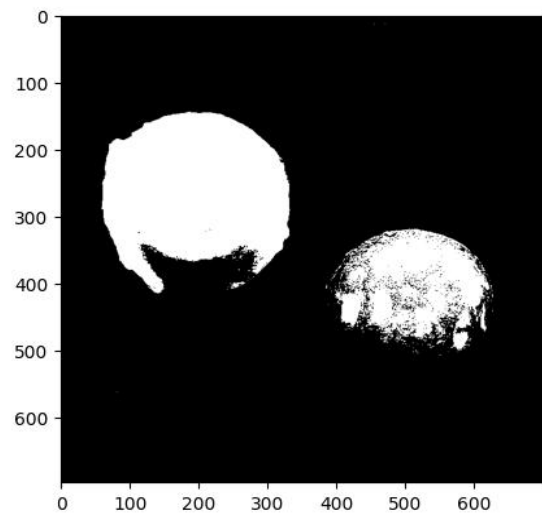


Fig 4.11 Image after applying Otsu's Segmentation Technique.



Fig 4.12 Calorie estimation of multiple foods in an image.

Trained Model	Training Accuracy	Testing Accuracy	Total Parameters
InceptionV3	99.84%	92.66%	23,912,235
VGGNet16	82.49%	78.47%	15,251,275
Basic CNN	84.65%	73.55%	646,923

Table 4.1 Comparison of accuracy of Trained Models.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE PLANS**

In this project, the food detection has been done using different CNN models. InceptionV3 classified the Indian foods better than VGGNet16 and Basic CNN models with the training accuracy of 99.84% and testing accuracy of 92.13%. Using contour-based outlining techniques in OpenCV, the boundaries were drawn precisely which helped us to isolate the various food items in the image. The volume of each food item is computed using segmented images. The calorie of each food is calculated using its volume.

As the future work, we can expand the food dataset to detect various dishes from various cuisine because it is restricted only to the Indian cuisine. For enhancing image segmentation, a powerful outlining technique can be employed to segment the image more accurately. We can build a user friendly app to identify the food items and computing their calories instantly.

## **CHAPTER 6**

### **REFERENCES**

1. **InceptionV3 algorithm:**
  - a. <https://arxiv.org/pdf/1512.00567.pdf>
2. **VGGNet16 algorithm:**
  - a. <https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide>
3. **Basic CNN algorithm:**
  - a. <https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/>
4. **Image Segmentation:**
  - a. <https://www.sciencedirect.com/science/article/pii/S1110016820303215>
  - b. <https://learnopencv.com/otsu-thresholding-with-opencv/>
5. **Volume estimation:**
  - a. <https://acadpubl.eu/jsi/2017-114-7-ICPCIT-2017/articles/12/35.pdf>

## CHAPTER 7

### APPENDIX

#### BASE PAPER

Suriyakrishnan Sathish, S. Ashwin b, Md. Abdul Quadir, L.K. Pavithra, " Analysis of Convolutional Neural Networks on Indian food detection and estimation of calorie " in *IEEE Transactions on Multimedia*, Volume 62, Part 7 2022, Pages 4665-4670

**DOI:** 10.1016/j.matpr.2022.03.122

**Keywords:** { Food detection, Deep learning, Calorie estimation, OpenCV, Indian cuisine},

**URL:**

<https://www.sciencedirect.com/science/article/pii/S2214785322014444?via%3Dihub>