DIETARY ASSESSMENT USING DEEP LEARNING

A MINOR PROJECT REPORT

Submitted By

KARAN SHAH
[RA1811003010370]
SHUBH JHAWAR
[RA1811003010405]

Under the guidance of

MS. R ANITA

(Assistant Professor,Dept of Computer Science and Engineering)

In Partial fulfillment for the award of

the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M.Nagar,Kattankulathur,ChengalpattuDistrict

NOVEMBER 2021

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section3 of UGC Act , 1956)

BONAFIDE CERTIFICATE

Certified that 18CSP107L minor project report titled "**DIETARY ASSESSMENT USING DEEP LEARNING**"is the bonafide work of "**KARAN SHAH [RA1811003010370], SHUBH JHAWAR [RA1811003010370]**" who carried out the minor project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]

SIGNATURE

MS.R ANITA
GUIDE
Assistant Professor
Dept. of Computer Science and Engg

[illegible]

SIGNATURE

DR. M. PUSHPALATHA
HEAD OF THE DEPARTMENT
Professor
Dept.of Computing
Technologies

[illegible]

Signature of the Panel Head
DR. SUBALALITHA C. N
PANEL HEAD
Associate Professor
Dept. of Computing Technologies

ABSTRACT

Due to the 3rd industrial revolution , our lives have changed a lot in the last century. While this has brought many incredibly great improvement in our lives , it also affected us in negative way in many aspects ,which include food and lifestyle.People are not having physically active lifestyle that is necessary for them, instead they are spending huge portion of their day indoors in sitting position , spending time using different technologies ,which further harm their health.Many people are facing various health problems nowadays. People are not eating healthy food and preferring fast food and highly processed food that is readily available. But the main problem is ,people do not have accurate picture of what they are eating in their days , which is important to help people understand their daily eating habits, exploring nutrition patterns and maintain a healthy diet.So , we are planning a deep model based food recognition and dietary assessment system to study and analyze food items from daily meal images (e.g., captured by smartphone)We are using the photo taken using the smartphone and applying algorithm to recognize multi-item (food) images by detecting candidate regions and using deep convolutional neural network (CNN) for object classification.It then identifies each region of proposals by mapping them into feature maps, and classifies them into different food categories, as well as locating them in the original images. Finally, the system will analyze the nutritional ingredients based on the recognition results and generate a dietary assessment report by calculating the amount of calories, fat, carbohydrate and protein.Our goal is to recognize the food items accurately and generate the dietary assessment report efficiently, which will benefit the users with a clear insight of healthy dietary and guide their daily recipe to improve body health and wellness.We have achieved 92% accuracy in this project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER1

INTRODUCTION

1.1 Problem Statement

In the recent decade , humans have made tremendous improvement , when it comes to improving people's lives.Most peoples are now living in democracy.They are now richer than ever.Also there is much improvement in people's health. Due to inventions in the last 2 centuries , after the industrial revolution that began in the late 18th century , we have more than doubled life expectancy. According to the World Health Organization (WHO), the current global life expectancy in 2016 was 72.0 years, 74.2 years for females and 69.8 years for males.But with these there are also some issues that have arised in the recent decades.One of the most important is obesity.

According to WHO almost 20% of deaths worldwide are attributable to an unhealthy diet. 39% of adults aged 18 years and over were overweight in 2016, and 13% were obese. Most of the world's population live in countries where overweight and obesity kills more people than underweight.

If people knew how much calories their food contains, then this problem would be somewhat controlled.But in reality , it is about the people not knowing what's in their diet. If people could estimate their calorie intake using the images of their food, they can easily decide on the amount of calories they want to consume. An image-based Calorie estimator built using deep learning can be a convenient app to keep track of what an individual's diet plan contains.

1.2 Our Approach

In a changing society where people have become health conscious there has still no breakthroughs in a convinient way to measure your daily food intake, one that is hassle free and even a normal person can do it if he/she tries. We are taking a step closer to devising the discussed way.
Everything thorugh a few clicks on your phone with the help of advanced deep learning running in the background. This is a work in progress and so far we have created an efficient model with the help of 'transfer learning' and as a result we acquired phenomenal accuracy, this is due to the millions of pictures pre-trained in an industrial-level model.
Our immediate next step would be to train the model again with more and more data so that it gets better overtime. Alongside that, we hope to build a mobile-interface that uses 'openCV' and give results in real-time and also keep a detailed track of it, through a mobile app connected to a database.
Next up, will be taking a step further by doing data analysis on the collected data of all the users and come up with customised plans and suggestions for an individual user all with the help of machine learning.

1.3 Different Approaches

Many researchers have worked on this concept and come up with different solutions to face this challenge, their work is inspiring and intuitive. Some of these earlier approaches are as follows.
DeepFood: Deep Learning-based Food Image Recognition for Computer-aided Dietary Assessment [1] used deep learning-based approach, where they used CNNs to build a state-of-the-art model for food-image recognition and dietary assessment, which gives an accuracy ranging from 80%-94% when tested on different datasets. The model was iterated for about 300,000 times, gradually increasing the accuracy per iteration. They made a model consisting of 7 layers, it takes 32x32 grey-scale images and after computation a 10-class output is generated ranging from 0-9.

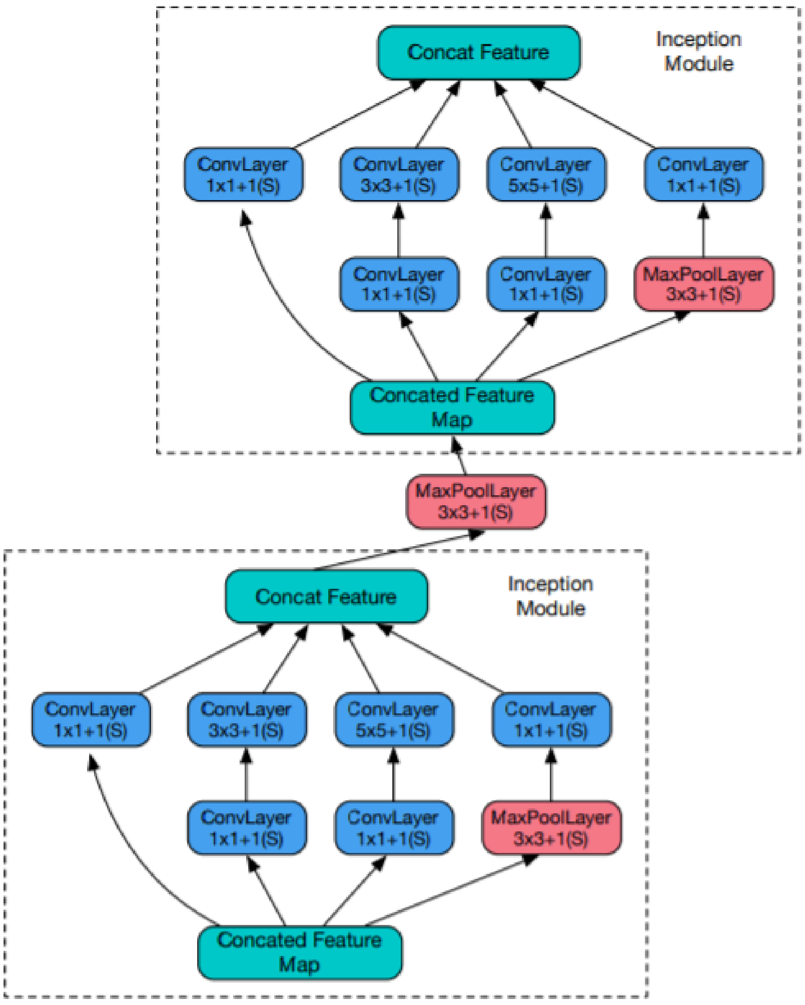


Fig 1.1 :Model architecture

Next up, is an article 'How I created Curry-AI' where the author collected his own data for different food items and then tried it on many things starting from transfer learning then a binary classification and lastly multi-class classification. At the end he got a training accuracy of 0.88 and validation accuracy of 0.59 on IndianFood31 dataset which consists of 85 different classes.

The model was not as powerful but the implementation inspired in the right direction, future scope of this article was to classify the dish then break it into its' ingredients in order to calculate the nutritional values and later it can be implemented in a mobile-based app.

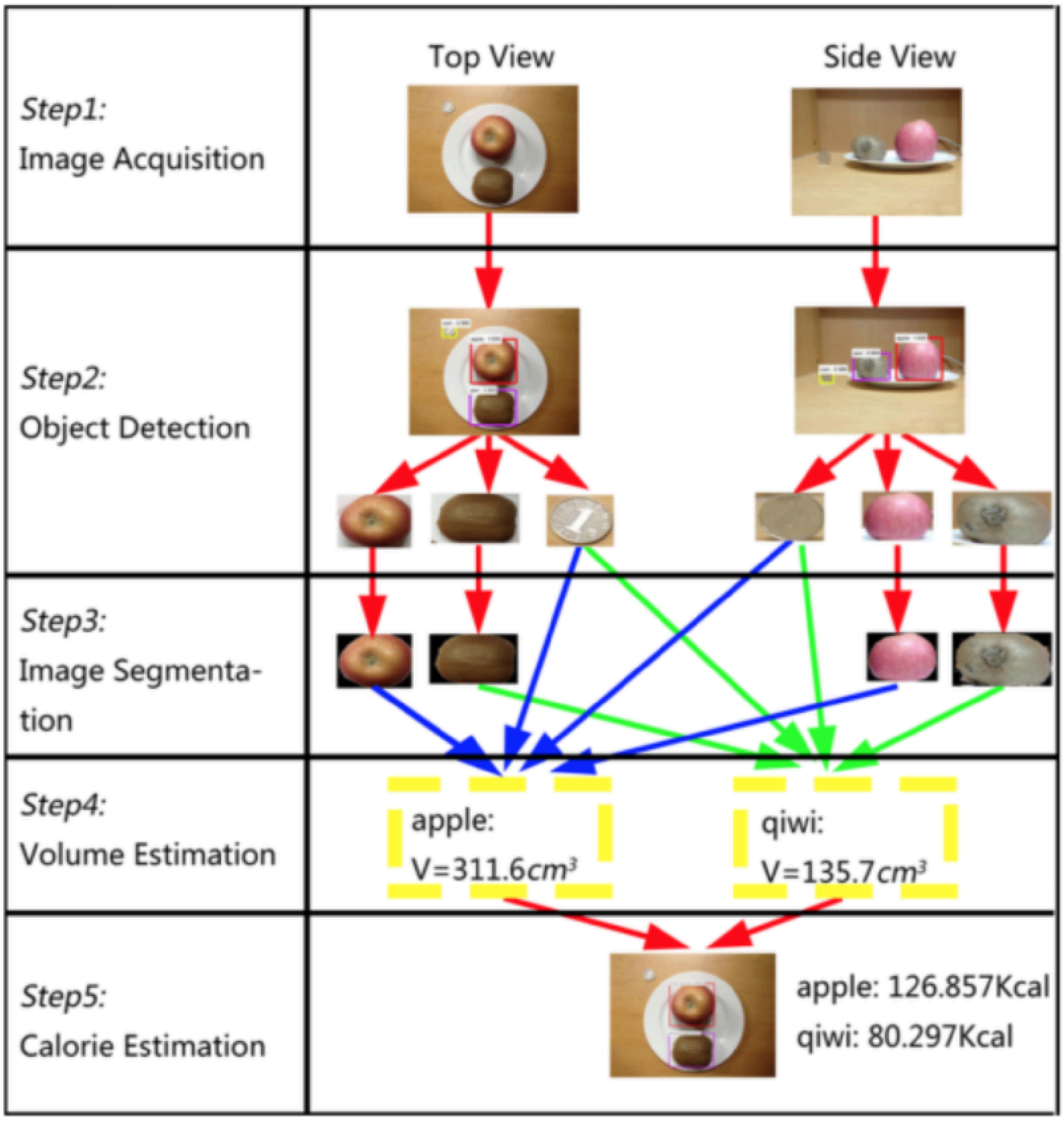


Figure 1.2 : Steps for different approach

Lastly, we looked at an article named 'Deep learning-Based food calorie Estimation method in Dietary Assessment', in this model the author used volume of the food item to calculate it's calories. Firstly, through a deep learning model food items are classified, then different individual items are segregated and their individual volume is estimated.

1.4 Software Requirement Specification

The software requirements to run this project are:

• **Operating System**: Windows 7/8/10/11 (incl. 64-bit, 32 bit) / Mac OS / Linux,

• **Language**: Python 3

Python is a high level programming language utilized on an extremely essential level for the most part accommodating programming. Guido van Rossum is credited to be the discoverer of Python in the year 1991. It has a development mental adequacy that underlines usually on points of view, for instance, code intelligence and strikingly uses immense whitespace. It is a reasonable programming language and is usually utilized over the globe for both little and gigantic degree programming. Python is regarded as an extraordinary choice by Engineers for Machine Learning (ML), Deep Learning (DL) and Artificial Intelligence (AI). It is like a way to have an expansive and clearing standard library. Positive conditions of Python which will help us in this assignment are:

• **IDE:** Jupyter Notebooks / Google Co-lab

Jupyter is a Python Integrated Development Environment (IDE). Jupyter allows us to program and run code in Python. It's a free, cross-platform IDE that runs on Anaconda.

Here , we have used Google Colab for our project.It is like Jupyter in online mode.There are some major advantages for using Google Colab.

- It is completely online . So we can access it from anywhere
- It does not require any computational power to run. Because it runs all the codes in Google server . Also , colab already provides free GPUs to use ,which is very fast compared to average computers today.
- As all the code is stored online , you just need a login credential to access the code.

Jupyter comes with a large number of built-in libraries that can be used, which can be used in Google Colab also.

- NumPy

- Pandas
- Matplotlib
- seaborn
- tensorflow
- keras
- cv2

Chapter 2

Literature Survey

Yang et al. proposed a method that calculates pairwise statistics between local features computed over a soft pixel level segmentation of the image into eight ingredient types. These statistics in a multi-dimensional histogram, which are then used as a feature vector for a discriminative classifier. The image is represented as the statistics of pairwise local features, known as pairwise feature distribution (PFD). Pairwise local feature distribution includes Soft labeling of pixels, Global Ingredient Representation (GIR) , Pairwise Features, Histogram representation for pairwise feature distribution, Histogram normalization. The method states that exploiting the spatial characteristics of food, in combination with statistical methods for pixel-level image labeling will enable the development of practical systems for food recognition.
Zong et al. proposes a food image classification method by means of local textural patterns and their global structure to describe the food image. The method uses a visual codebook of local textural patterns created by employing Scale Invariant Feature Transformation (SIFT) interest point detector using the Local Binary Pattern (LBP) feature. The global structure of the food object is represented as the spatial distribution of the local textural structures and encoded using shape context. By using shape context to represent the relative spatial relationship between codewords, the proposed method can accommodate deformations and transformations in the shape of food objects. But this technique does not incorporate view invariant texture features.
Kong et al.developed an automatic camera phone based multi-view food classifier named DietCam. DietCam uses a probabilistic method to separate every food from multiple images.The recognition accuracy is increased through an enhanced joint distribution from every viewpoint. First for classifying food items from the images, they detect and extract local feature points in every image and classify these features based on an existing feature database .They then increase the recognition accuracy through result verifications from multiple viewpoints. It considers the images are taken by three cameras at a synchronized time. A new technique has been introduced as perspective distance, which reflects the geometric relation between two features concerning their appearances in all the possible perspectives. It shows an accuracy of 84% for regular shaped food items.
Matsuda et al. proposed a two-step method for recognizing multiple food images by detecting regions of interest i.e. candidate regions using various methods and classifying them according to the features extracted. They detected several candidate region by integrating several results acquired from the region detected using (DPM), a circle detector and the JSEG region segmentation. And then applying feature extraction methods like including bag-of-features of SIFT and CSIFT with spatial pyramid (SPBoF), histogram of oriented gradient(HoG), and Gabor texture features and finally classify them according to SVM score. They estimated ten food candidates for multiple-food images in the descending order of the SVM scores. They have achieved the 55.8% classification rate.
Y. Kawano et al. built interactive and real time food recognition and recording them on the user 's smartphone. First, the user draws bounding boxes according to the region of interest for more accurate results they segmented the food image by Grub Cut, extracted a color histogram and SURF based bag-of-features. And finally classify them using a linear SVM with a fast $\chi2$ kernel for fast and accurate food recognition. They have achieved 81.55% classification rate for the top 5 candidates when ground-truth bounding boxes are given.
Joutou et al. proposed food recognition systems which are 50 kinds of common food items in Japan. They have proposed a method for recognizing food images by integrating various kinds of image features including SIFT-based bag-of-features, Gabor, and color histograms and classify it

into one of the given food categories with the trained MKL-SVM and they have achieved the 61.34% classification rate.
H. Hoashi et al. discussed new image features and food categories . As new image features added are, gradient histogram which can be regarded as a simple version of Histogram of Oriented Gradient (HoG), and added 35 new food categories to the existing 50 categories in the existing system. They have also described the proposed method based on feature fusion of various kinds of image features with MKL.
Y. Maruyama et al. consists of work based on the "FoodLog" system, where the user takes photos of the foods and uploads them to the system, and the system performs image processing to detect food images and determine the food balance. The foodlog system allows the user to correct the results of the system. They proposed a method to make use of the corrections made by the user by Naive Bayes which is one of the Bayesian networks It also investigates how to improve the accuracy by using user feedbacks. First, we compare the accuracy of the performance between SVM and Naive Bayes Then, making use of the user's corrections as feedback, the Bayesian network is updated to improve the performance.
Kawano et al. proposed the extension of an existing image dataset automatically leveraging existing categories and crowdsourcing. There are uncountable food categories, since foods are different from place to place. Dataset of one cultured food cannot be used for the Food detection system of another culture. This enables not only to build other -cultured food datasets based on an original food image dataset automatically, but also to save as much crowd -sourcing costs as possible. Basically, they focused on expansion on food image data sets to build a food dataset irrespective of food culture.
M. M. Zhang et al. used the idea of attribute-based classification, they classified plates of food to the exact cuisine by the country, using the ingredients as attributes for a plate of food. First recognized the ingredients, gave each ingredient a probability marker, and then used pairwise local features among the ingredients to find out the food category, by calculating the orientation, distance, and other properties between each pair of ingredients. To identify the cuisine using ingredients, attribute-based classification is used. First level includes use of Earth Mover's Distance as the low-level feature. EMD turns out to be problematic for dishes where only one ingredient is present. In the last level, to determine the cuisine category from the attributes, we use the attribute vectors to train the final classifier, with the cuisine category ID as ground truth which is the image's area ratio for the definite ingredient classifier, as extracted from the image's attribute vector & determine at intermediate level. Hence the final categorization of the cuisines is separate from the raw images, as it uses the intermediate high-level attributes layer to predict the results.
Rother et al. extends the graph-cut approach in three respects. First, developed a more powerful, iterative version of the optimization. Second, the power of the iterative algorithm is used to considerably simplify the user interaction needed for a given value of result. Third, a robust algorithm for "border matting" has been developed to estimate and at the same time the alpha-matte around an object boundary and the colors of foreground pixels. Graph-cut is used to accomplish robust segmentation even in camouflage, when foreground and background colors distributions are not well separated. Graph-cut method is failed in terms of user interactions, can occur in three cases: (i) regions of low contrast at the transition from foreground to background (ii) hide, in which the true foreground and background distributions overlap to some extent in colour space (iii) background material inside the user rectangle happens not to be effectively represented in the background region. Graph-cut is successful where the bounding rectangle on your own is a sufficient user interaction to enable foreground extraction to be accomplished automatically by GrabCut.
Y. Matsuda et al. proposed a method to identify multiple food items from one food image considering co occurrence statistics with the manifold ranking method. As in traditional method, first candidate regions detected after that image features extraction technique is applied to classify models trained by MKL & obtain the names of the top N food item candidates over the given

image. Some familiar combinations of food items such as “hamburger and french-fries” and “rice and miso-soup” & improbable combinations exist such as “sushi and hamburger” or “sashimi and french-fries” are observed. From these observations, co-occurrence statistics is try to enhance the performance of multiple-food image recognition. By considering co-occurrence statistics, we can reduce improbable combinations which are present in the higher ranked candidates. For multi-food recognition with co occurrence statistics, we use manifold ranking which is actually a re-ranking method to consider similarities between items

Chapter 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Sequence Diagram

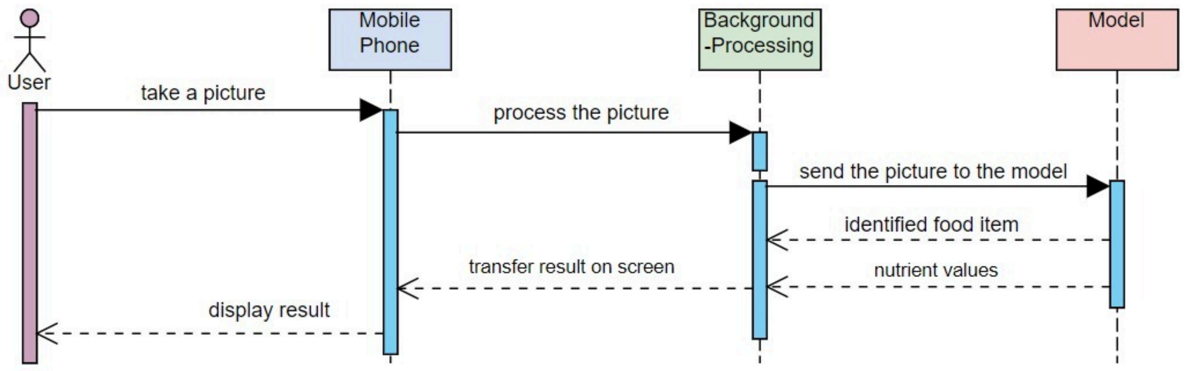


Figure 3.1 : Sequence Diagram

The above diagram is sequence diagram of our project.

A sequence diagram or system sequence diagram shows object interactions arranged in time sequence in the field of software engineering. A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

3.2 Data Flow Diagram

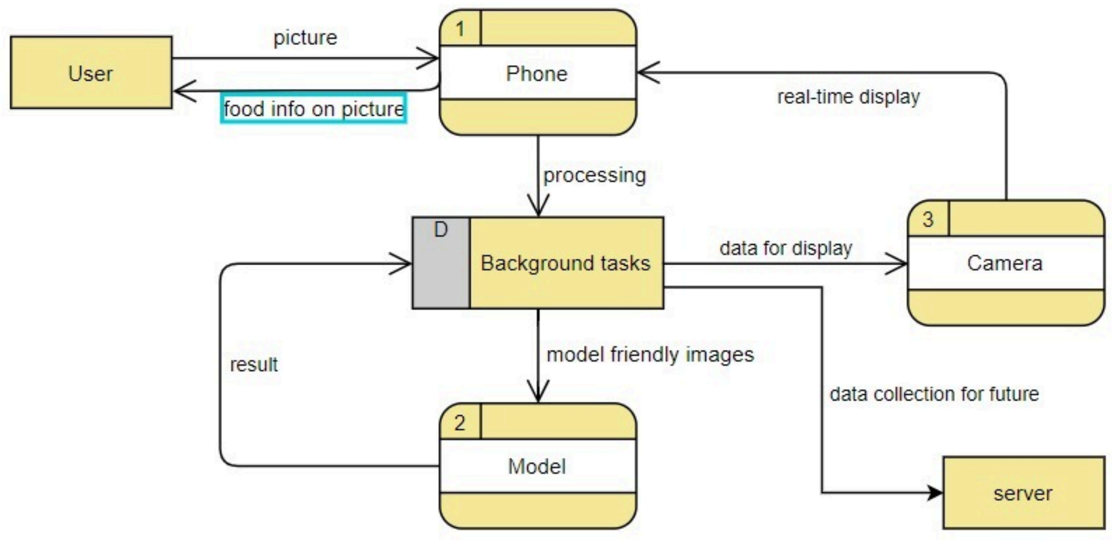


Figure 3.1 : Data Flow Diagram

The above is the Data flow diagram for our project.

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation

Chapter 4

Methodology

To implement the project , following 6 main steps are followed:

1. Data Collecting
2. Data Preprocessing
3. Model Definition
4. Model Training
5. Food Image Classification
6. Displaying the nutritional Value

4.1 Data Collecting

Here, we are taking the dataset from kaggle website.Dataset is called 'food20 dataset' , which is a food database of most popular indian food items.Of the all the dataset available for indian food items, it was among the largest and most used dataset .It originally has 20 different food items ,with each having 100 images each. From this, 10 food items are being selected for this project.
Here , each category has 100 images in total. They are divided into 2 types as a whole. First is the training set ,which is used to train our Deep Learning model. The second is a testing set , which is a validation dataset .A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's hyperparameters.Here each food item dataset is divided in 70:30 ratio for train_set and test_set respectively.

4.2 Data Preprocessing

We create a get_data function which will return a numpy array of the required data. First the images are read through opencv2 ,which is a popular python library used for real-time computer vision.Now images are converted into RGB format from BGR format.Now images are resized to 256x256 pixel format. After getting a numpy array, training and validating numpy arrays are created, normalised and reshaped, getting them ready for our model.For the model , we have created 4 numpy arrays. 1) x_train ,2) y_train ,3)x_val , 4) y_val. Here each of the arrays has 3 values for each pixel , having values between 0 and 1 , including both , which represent Red , Green and Blue colour's intensity.

4.3 Model Definition

For the model selection , we have a model based on Transfer Learning , called MobileNetV2. MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices.It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity.As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. We are also printing a model summary , which will show CNN Architecture with parameter count for the 'sequential' model .

4.4 Model Training

Now , we finally run our model. It is run for a total of 150 epochs with a base learning rate of 0.001. It will run x_train and y_train against validation datas , x_val and y_val respectively. For both parts of the model , it will calculate loss value and accuracy value and show it for each of the epochs. Also ,we are using Adam from the keras package.Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. Adam is relatively easy to configure where the default configuration parameters do well on most problems.We use Adam optimizer to update the parameter values with the backpropagation.After training the model, we plot the training and validation loss. We also analyze the accuracy score for the epoch.

4.5 Food Image Classification

After we have successfully created a model , we use it to classify our food items.After training the model on the training data , we run our model against the validation data for our food items. We have taken a total of 10 different food items.Not only that, each food item does not have any resemblance to others.

4.6 Displaying the Nutritional Value

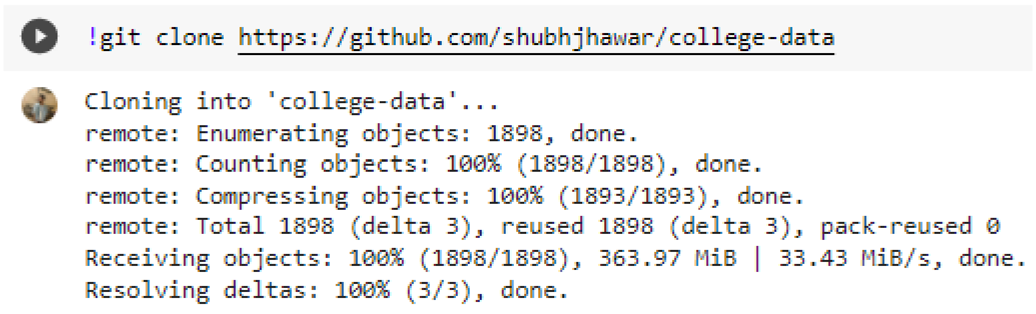
After we have classified the food image , we will show its nutritional value. For this, we have taken its nutritional value from a website called ‘nutritionix' which is a wide database of food and its nutritional facts. It contains many different nutrients , from which we will show only main , which is calories , fats , protein , carbohydrates and fats . For the final output , it will print out the name of the food item and its nutritional value for the user.

CHAPTER 5

CODING AND TESTING

5.1 Coding

Cloning the database from github and loading it into google colab.

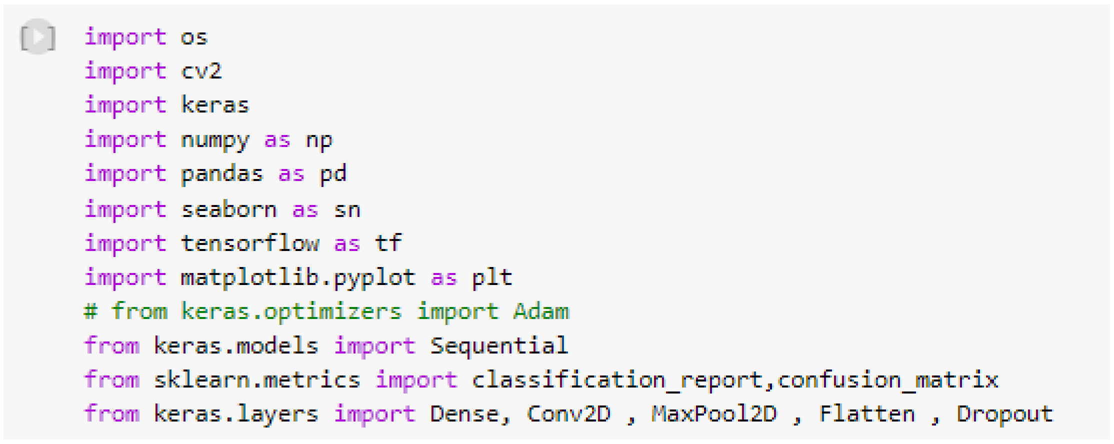


```
!git clone https://github.com/shubhjhawar/college-data
```

```
Cloning into 'college-data'...
remote: Enumerating objects: 1898, done.
remote: Counting objects: 100% (1898/1898), done.
remote: Compressing objects: 100% (1893/1893), done.
remote: Total 1898 (delta 3), reused 1898 (delta 3), pack-reused 0
Receiving objects: 100% (1898/1898), 363.97 MiB | 33.43 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Figure 5.1 : Cloning the dataset

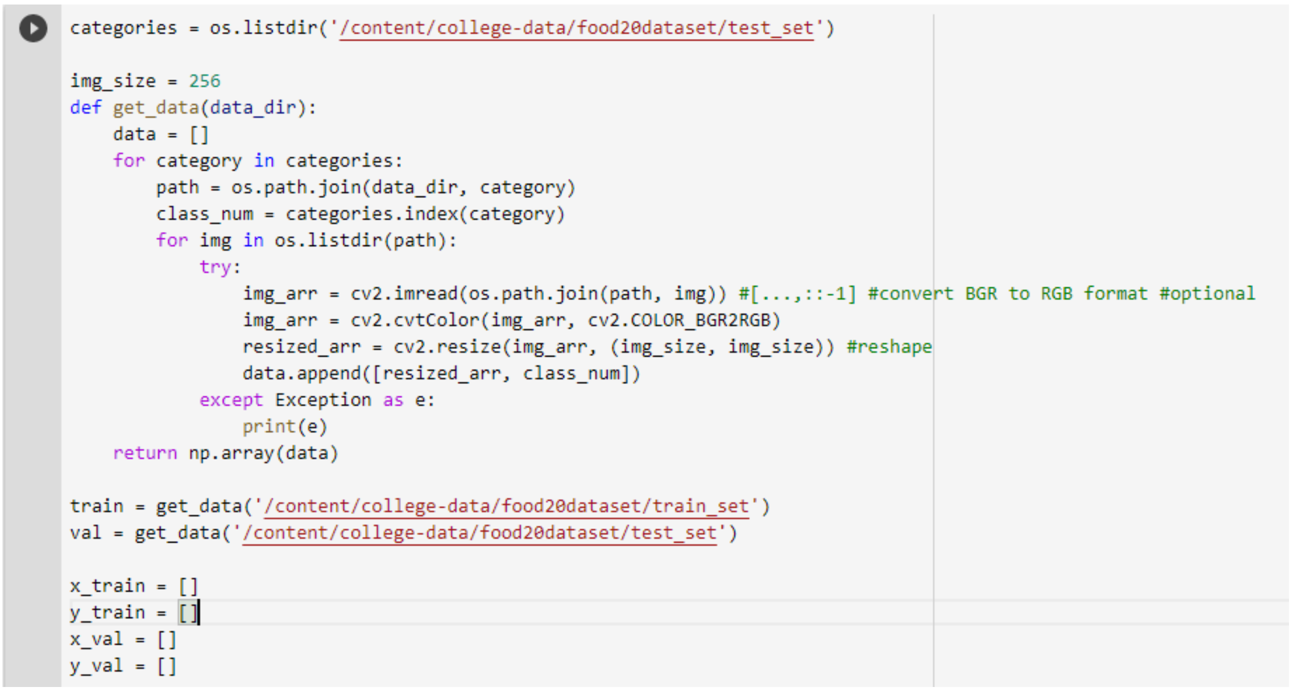
importing the required libraries



```
import os
import cv2
import keras
import numpy as np
import pandas as pd
import seaborn as sn
import tensorflow as tf
import matplotlib.pyplot as plt
# from keras.optimizers import Adam
from keras.models import Sequential
from sklearn.metrics import classification_report,confusion_matrix
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
```

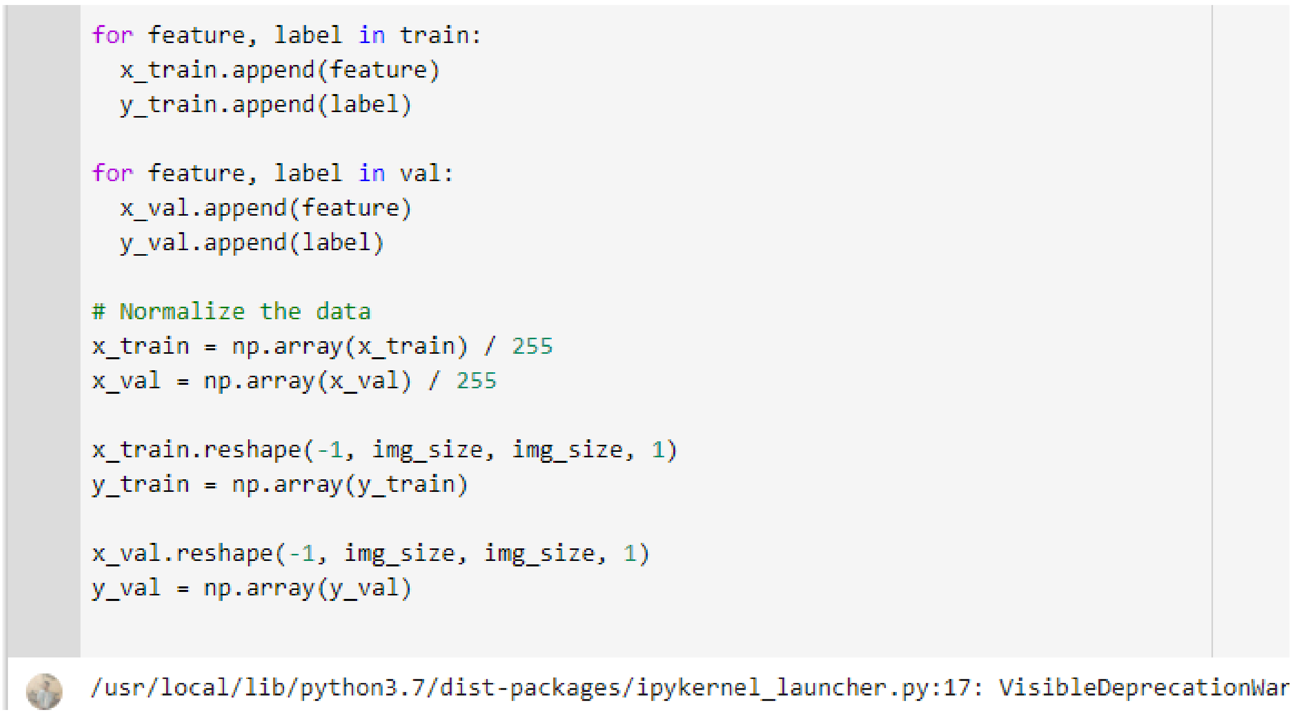
Figure 5.2: Importing the libraries

Preprocessing the image dataset (converting , resizing and normalizing) and converting them into numpy arrays



```
categories = os.listdir('/content/college-data/food20dataset/test_set')

img_size = 256
def get_data(data_dir):
    data = []
    for category in categories:
        path = os.path.join(data_dir, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img)) #[...,::-1] #convert BGR to RGB format #optional
                img_arr = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) #reshape
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)

train = get_data('/content/college-data/food20dataset/train_set')
val = get_data('/content/college-data/food20dataset/test_set')

x_train = []
y_train = []
x_val = []
y_val = []
```

Figure 5.3:Preprocessing the images

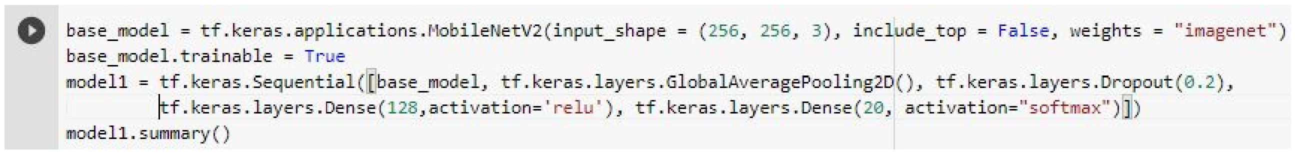


```
for feature, label in train:
  x_train.append(feature)
  y_train.append(label)

for feature, label in val:
  x_val.append(feature)
  y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255

x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: VisibleDeprecationWar
```

Figure 5.4: Converting images into numpy arrays

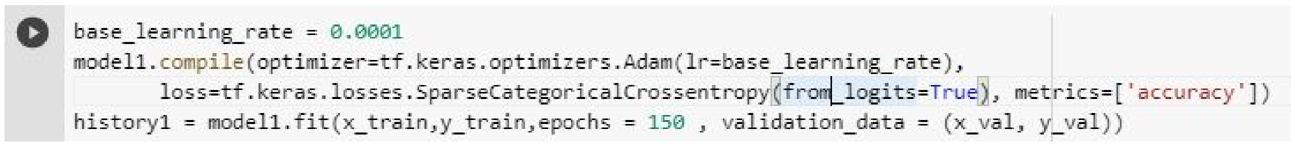
Defining the model



```
base_model = tf.keras.applications.MobileNetV2(input_shape = (256, 256, 3), include_top = False, weights = "imagenet")
base_model.trainable = True
model1 = tf.keras.Sequential([base_model, tf.keras.layers.GlobalAveragePooling2D(), tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(128,activation='relu'), tf.keras.layers.Dense(20, activation="softmax")])
model1.summary()
```

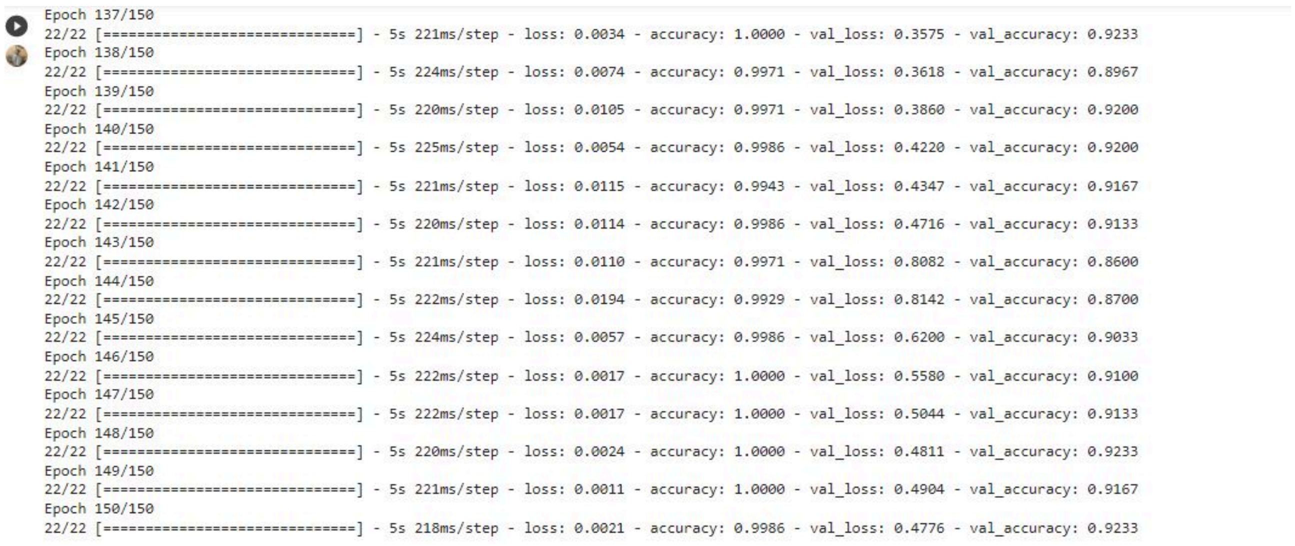
Figure 5.5: Model definition

Training the model for 150 epochs



```
base_learning_rate = 0.0001
model1.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
history1 = model1.fit(x_train,y_train,epochs = 150 , validation_data = (x_val, y_val))
```

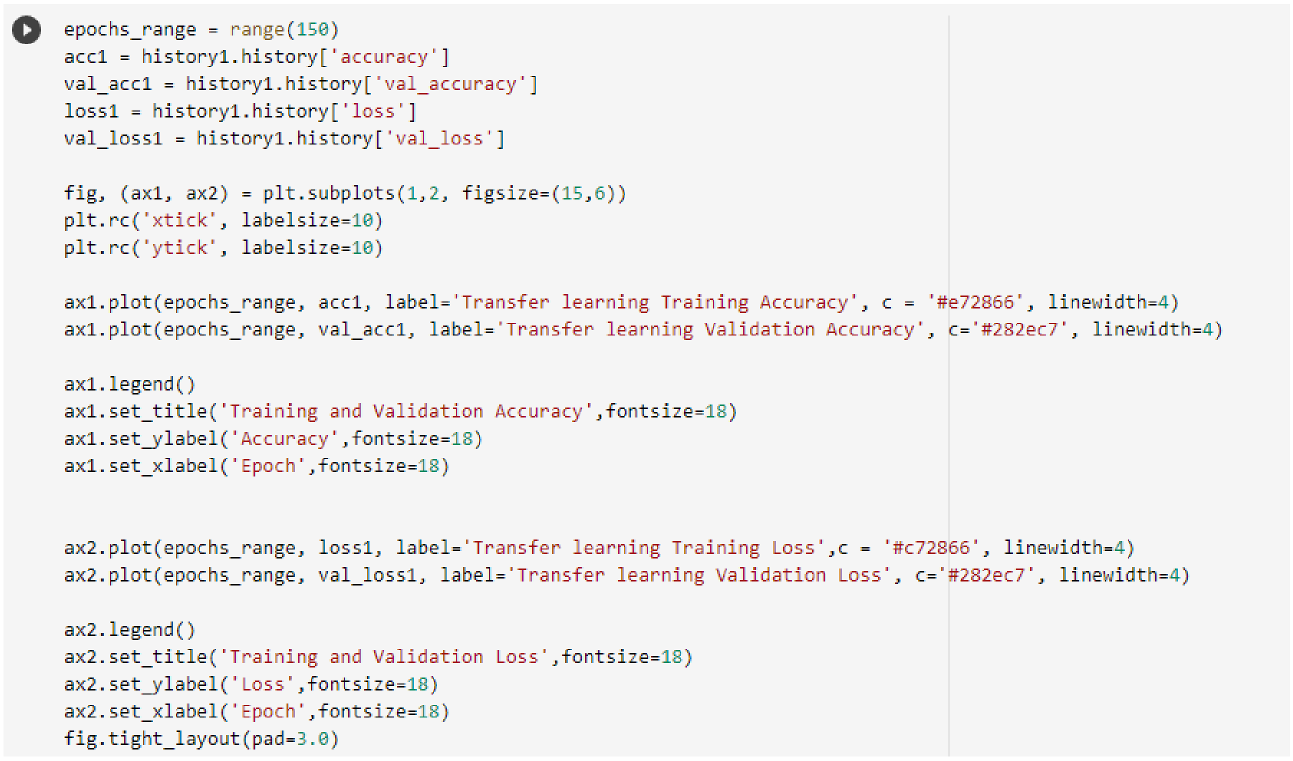
Figure 5.6: Model training



```
Epoch 137/150
22/22 [==============================] - 5s 221ms/step - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.3575 - val_accuracy: 0.9233
Epoch 138/150
22/22 [==============================] - 5s 224ms/step - loss: 0.0074 - accuracy: 0.9971 - val_loss: 0.3618 - val_accuracy: 0.8967
Epoch 139/150
22/22 [==============================] - 5s 220ms/step - loss: 0.0105 - accuracy: 0.9971 - val_loss: 0.3860 - val_accuracy: 0.9200
Epoch 140/150
22/22 [==============================] - 5s 225ms/step - loss: 0.0054 - accuracy: 0.9986 - val_loss: 0.4220 - val_accuracy: 0.9200
Epoch 141/150
22/22 [==============================] - 5s 221ms/step - loss: 0.0115 - accuracy: 0.9943 - val_loss: 0.4347 - val_accuracy: 0.9167
Epoch 142/150
22/22 [==============================] - 5s 220ms/step - loss: 0.0114 - accuracy: 0.9986 - val_loss: 0.4716 - val_accuracy: 0.9133
Epoch 143/150
22/22 [==============================] - 5s 221ms/step - loss: 0.0110 - accuracy: 0.9971 - val_loss: 0.8082 - val_accuracy: 0.8600
Epoch 144/150
22/22 [==============================] - 5s 222ms/step - loss: 0.0194 - accuracy: 0.9929 - val_loss: 0.8142 - val_accuracy: 0.8700
Epoch 145/150
22/22 [==============================] - 5s 224ms/step - loss: 0.0057 - accuracy: 0.9986 - val_loss: 0.6200 - val_accuracy: 0.9033
Epoch 146/150
22/22 [==============================] - 5s 222ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.5580 - val_accuracy: 0.9100
Epoch 147/150
22/22 [==============================] - 5s 222ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.5044 - val_accuracy: 0.9133
Epoch 148/150
22/22 [==============================] - 5s 220ms/step - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.4811 - val_accuracy: 0.9233
Epoch 149/150
22/22 [==============================] - 5s 221ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.4904 - val_accuracy: 0.9167
Epoch 150/150
22/22 [==============================] - 5s 218ms/step - loss: 0.0021 - accuracy: 0.9986 - val_loss: 0.4776 - val_accuracy: 0.9233
```

Figure 5.7 : Running the model for training

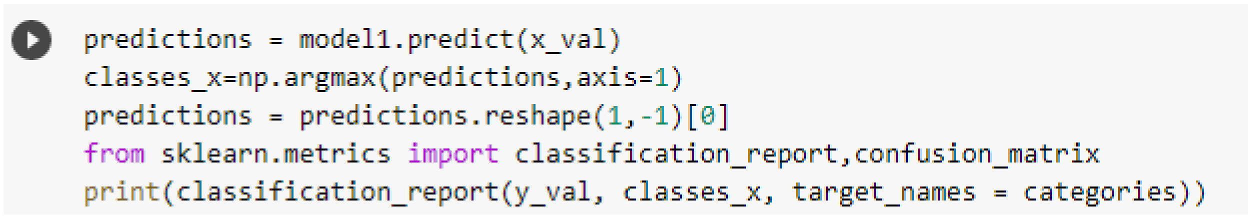
Plotting accuracy and loss for both the validation and training



```
epochs_range = range(150)
acc1 = history1.history['accuracy']
val_acc1 = history1.history['val_accuracy']
loss1 = history1.history['loss']
val_loss1 = history1.history['val_loss']

fig, (ax1, ax2) = plt.subplots(1,2, figsize=(15,6))
plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)

ax1.plot(epochs_range, acc1, label='Transfer learning Training Accuracy', c = '#e72866', linewidth=4)
ax1.plot(epochs_range, val_acc1, label='Transfer learning Validation Accuracy', c='#282ec7', linewidth=4)

ax1.legend()
ax1.set_title('Training and Validation Accuracy',fontsize=18)
ax1.set_ylabel('Accuracy',fontsize=18)
ax1.set_xlabel('Epoch',fontsize=18)


ax2.plot(epochs_range, loss1, label='Transfer learning Training Loss',c = '#c72866', linewidth=4)
ax2.plot(epochs_range, val_loss1, label='Transfer learning Validation Loss', c='#282ec7', linewidth=4)

ax2.legend()
ax2.set_title('Training and Validation Loss',fontsize=18)
ax2.set_ylabel('Loss',fontsize=18)
ax2.set_xlabel('Epoch',fontsize=18)
fig.tight_layout(pad=3.0)
```

Figure 5.8: Plotting the result of model

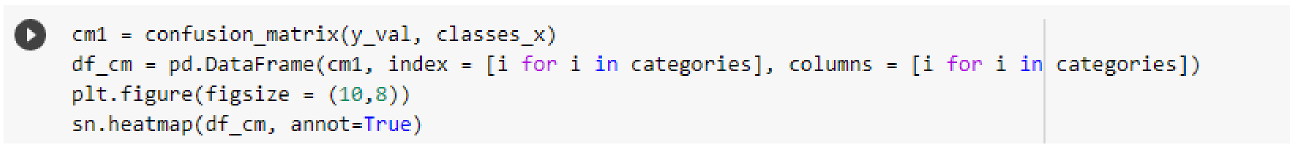
Printing Category wise Precision, Recall and F1-score.



```
predictions = model1.predict(x_val)
classes_x=np.argmax(predictions,axis=1)
predictions = predictions.reshape(1,-1)[0]
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_val, classes_x, target_names = categories))
```

Figure 5.9: Printing category wise precision , recall and F-1 score

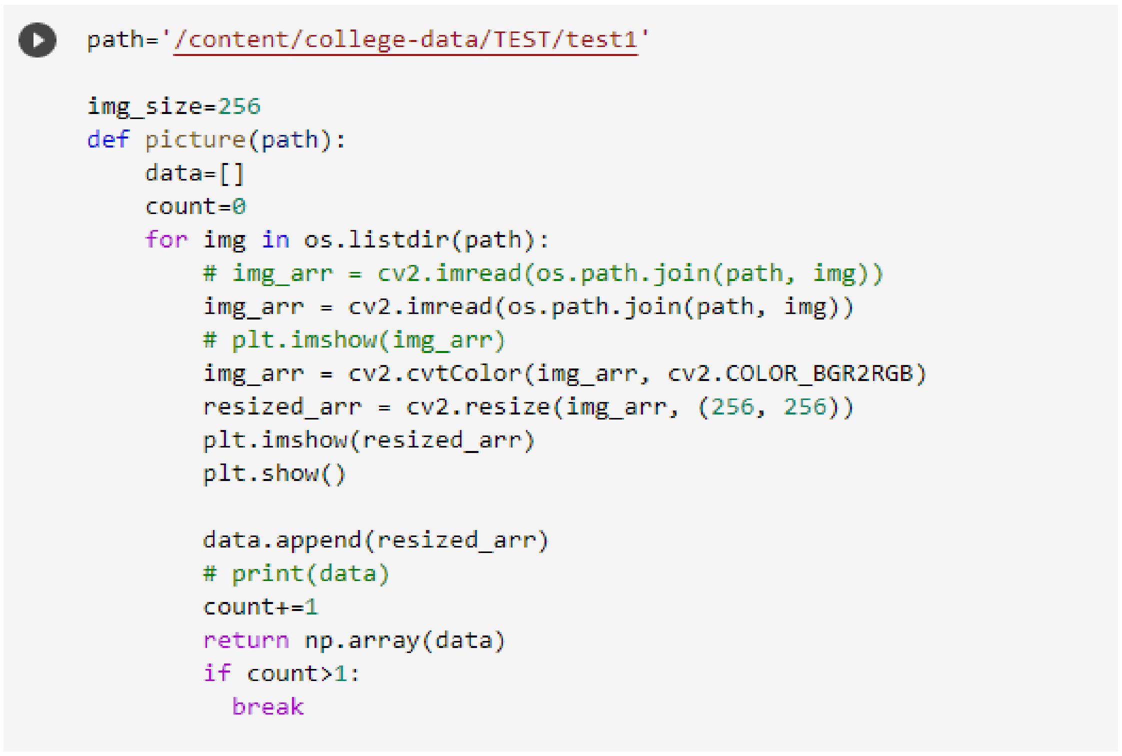
Showing the confusion matrix table representation



```
cm1 = confusion_matrix(y_val, classes_x)
df_cm = pd.DataFrame(cm1, index = [i for i in categories], columns = [i for i in categories])
plt.figure(figsize = (10,8))
sn.heatmap(df_cm, annot=True)
```

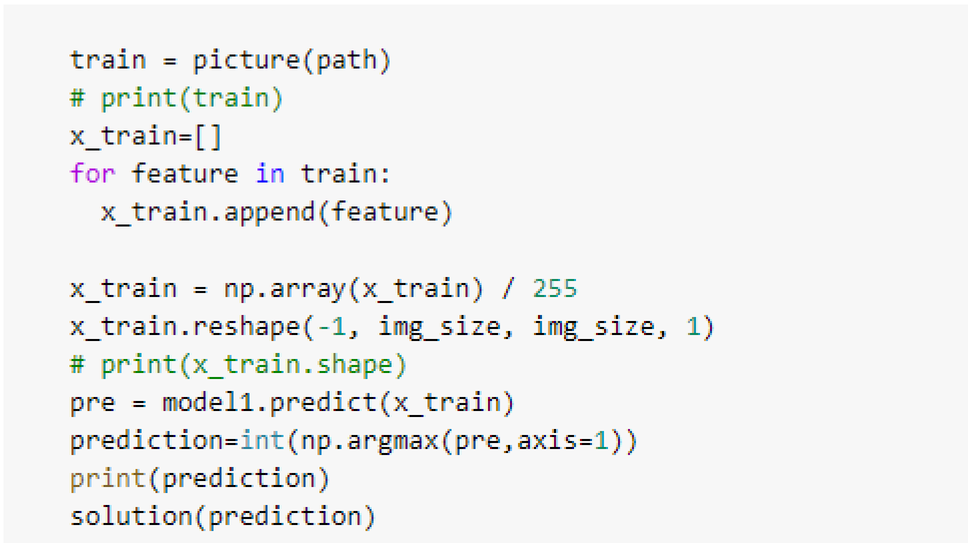
Figure 5.10: Printing confusion matrix

Testing our model using external images



```python
path='/content/college-data/TEST/test1'

img_size=256
def picture(path):
    data=[]
    count=0
    for img in os.listdir(path):
        # img_arr = cv2.imread(os.path.join(path, img))
        img_arr = cv2.imread(os.path.join(path, img))
        # plt.imshow(img_arr)
        img_arr = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
        resized_arr = cv2.resize(img_arr, (256, 256))
        plt.imshow(resized_arr)
        plt.show()

        data.append(resized_arr)
        # print(data)
        count+=1
        return np.array(data)
        if count>1:
          break
```

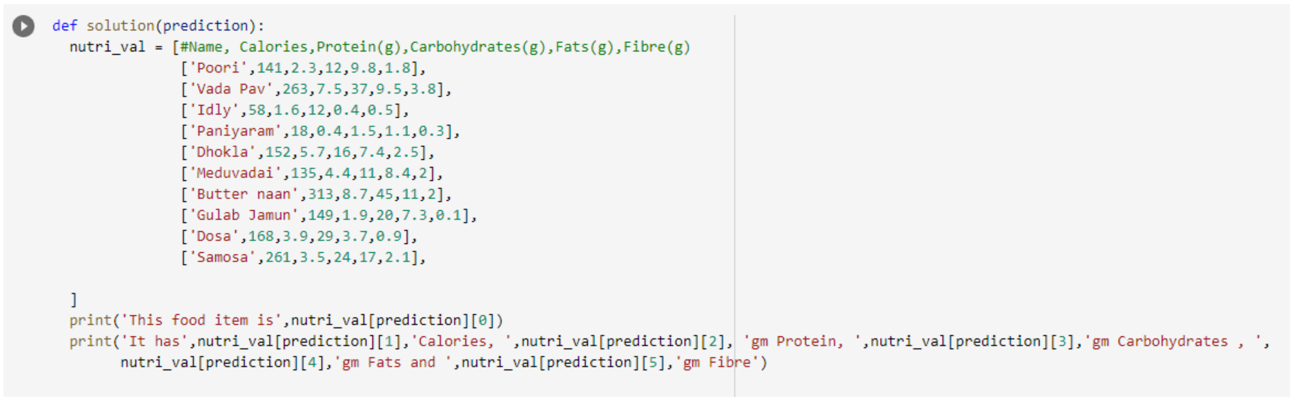
Figure 5.11: testing the model -1



```python
train = picture(path)
# print(train)
x_train=[]
for feature in train:
  x_train.append(feature)

x_train = np.array(x_train) / 255
x_train.reshape(-1, img_size, img_size, 1)
# print(x_train.shape)
pre = model1.predict(x_train)
prediction=int(np.argmax(pre,axis=1))
print(prediction)
solution(prediction)
```

Figure 5.12: testing the model -2

Adding the nutritional values and showing the final output with the food
name and its nutritional facts.



```
def solution(prediction):
  nutri_val = [#Name, Calories,Protein(g),Carbohydrates(g),Fats(g),Fibre(g)
               ['Poori',141,2.3,12,9.8,1.8],
               ['Vada Pav',263,7.5,37,9.5,3.8],
               ['Idly',58,1.6,12,0.4,0.5],
               ['Paniyaram',18,0.4,1.5,1.1,0.3],
               ['Dhokla',152,5.7,16,7.4,2.5],
               ['Meduvadai',135,4.4,11,8.4,2],
               ['Butter naan',313,8.7,45,11,2],
               ['Gulab Jamun',149,1.9,20,7.3,0.1],
               ['Dosa',168,3.9,29,3.7,0.9],
               ['Samosa',261,3.5,24,17,2.1],

  ]
  print('This food item is',nutri_val[prediction][0])
  print('It has',nutri_val[prediction][1],'Calories, ',nutri_val[prediction][2], 'gm Protein, ',nutri_val[prediction][3],'gm Carbohydrates , ',
        nutri_val[prediction][4],'gm Fats and ',nutri_val[prediction][5],'gm Fibre')
```

Figure 5.13: Adding the nutritional facts

TESTING

Software testing is an activity that checks in case the real outcomes match the normal outcomes and guarantees that the product framework is liberated from defects. It involves running a program or framework part to assess at least one property of interest.Software testing also aids in the detection of flaws, gaps, or missing requirements that are not in accordance with the actual requirements. It can be done manually or with the assistance of computerized devices.

5.1 Types of Testing

5.1.1 Functional Testing

Functional Testing is a kind of programming testing by which the framework is tried against the useful necessities/determinations. Capacities (or highlights) are tried by taking care of them input and analyzing the yield. Practical testing guarantees that the prerequisites are appropriately fulfilled by the application. This sort of testing isn't worried about how handling happens, but instead, with the aftereffects of handling. It mimics genuine framework use however doesn't make any framework structure suppositions. During utilitarian testing, Black Box Testing procedure is utilized in which the internal logic of the framework being tried isn't known to the analyzer. Functional testing is often carried out throughout the System Testing and Acceptance Testing stages.Functional testing usually entails the following steps:

- Determine which functions the software will be expected to fulfil.
- Create input data according to the function's requirements.
- Determine the output using the function's parameters.
- Carry out the test case.
- Compare the actual and anticipated results.

When the test conditions are created directly from user/business requirements, functional testing is more successful.

5.2.2 Non-functional Testing

A Type of Software testing to check non-practical viewpoints (usability, performance, reliability, etc.) of a product application is known as Non-functional Testing. It's intended to assess a system's readiness based on nonfunctional criteria that aren't covered by functional testing. Checking how many individuals can simultaneously login into a software is a good example of a non-functional test. Client satisfaction is affected by non-functional testing, which is just as important as functional testing. In previous test cycles, the Non Functional requirements were likewise not given the attention they deserved. However, this is no longer the case. They are currently generally significant as they think about all the application execution and security. This testing greatly affects applications with regards to the exhibition of the application under high client traffic. This testing guarantees that your application is steady and is ready to deal with load under outrageous conditions.

5.2.3 Regression Testing

A type of software testing that aims to guarantee that software modifications (enhancements or bug fixes) haven't had a negative impact is known as Regression Testing. Any code change has the potential to affect functionality that isn't directly related to the code, therefore regression testing is critical to ensure that repairing one thing doesn't break another. During regression testing, no new test cases are developed; instead, existing test cases are re-run. The act of returning or reverting to a previous place or state is known as regression. In a perfect world, a comprehensive regression test would be performed, but time and resources are often limited. In such circumstances, an impact analysis of the changes is required to identify sections of the software that are most likely to be affected by the change and have the most influence on users in the event of a malfunction, as well as focus testing around those areas. Regression test automation solutions are being used by an increasing number of firms and projects because of the scope and importance of regression testing.

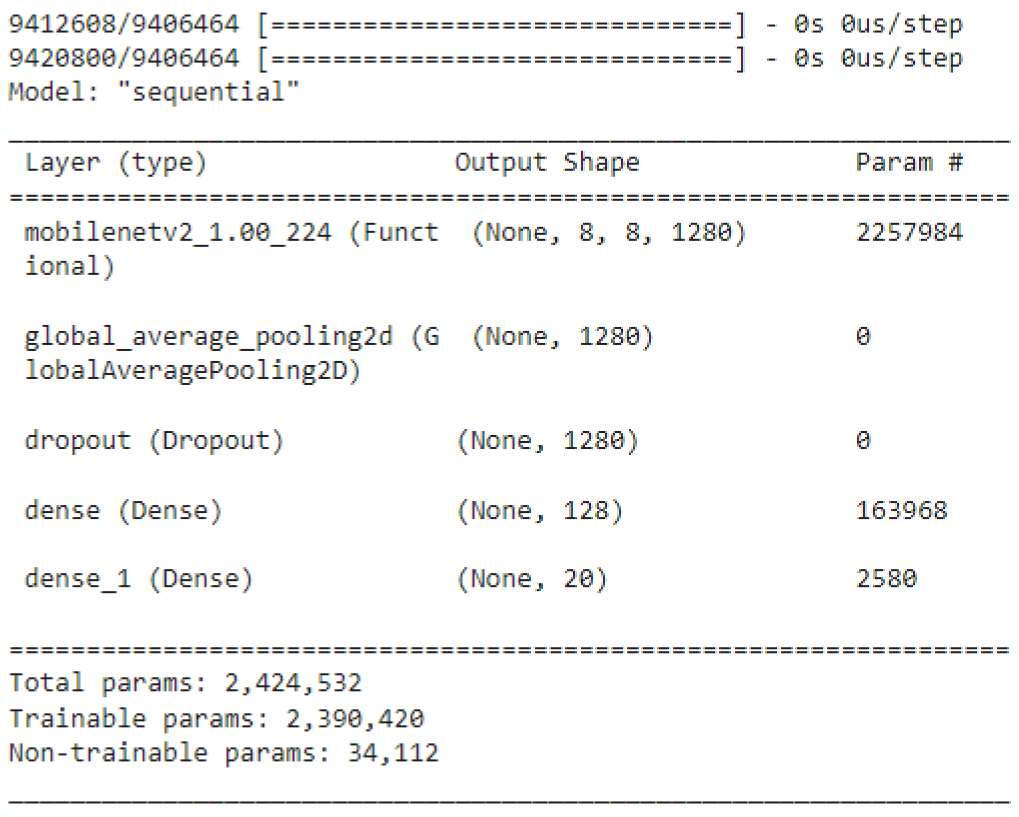
5.2.4 Performance Testing

The practice of determining the speed, responsiveness, and stability of a software program, network, computer, or device under demand is known as performance testing. Performance testing can consist of quantitative tests carried out in a lab or constrained scenarios carried out in the production environment. Processing speed, network bandwidth data transfer rate, workload efficiency, throughput, and reliability are all common criteria. An organisation can, for example, assess a program's reaction time when a user requests an activity or the number of millions of instructions per second (MIPS) that a mainframe can process.

Chapter 6

Result and Discussion

Result shows that our model has 92% accuracy , which is impressive considering relatively smaller dataset and quick computing resources required.



```
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 8, 8, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 128)               163968

 dense_1 (Dense)             (None, 20)                2580

=================================================================
Total params: 2,424,532
Trainable params: 2,390,420
Non-trainable params: 34,112
_________________________________________________________________
```

Figure 6.1: CNN architecture with parameters values

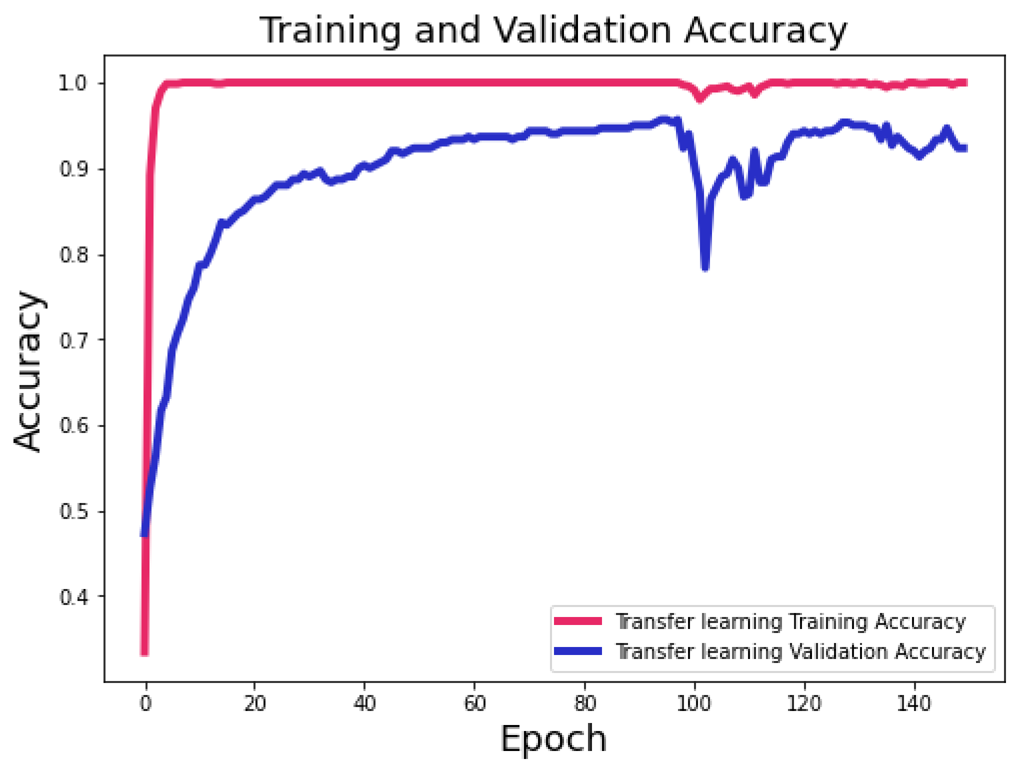


Figure 6.2: Training and Validation Accuracy graph

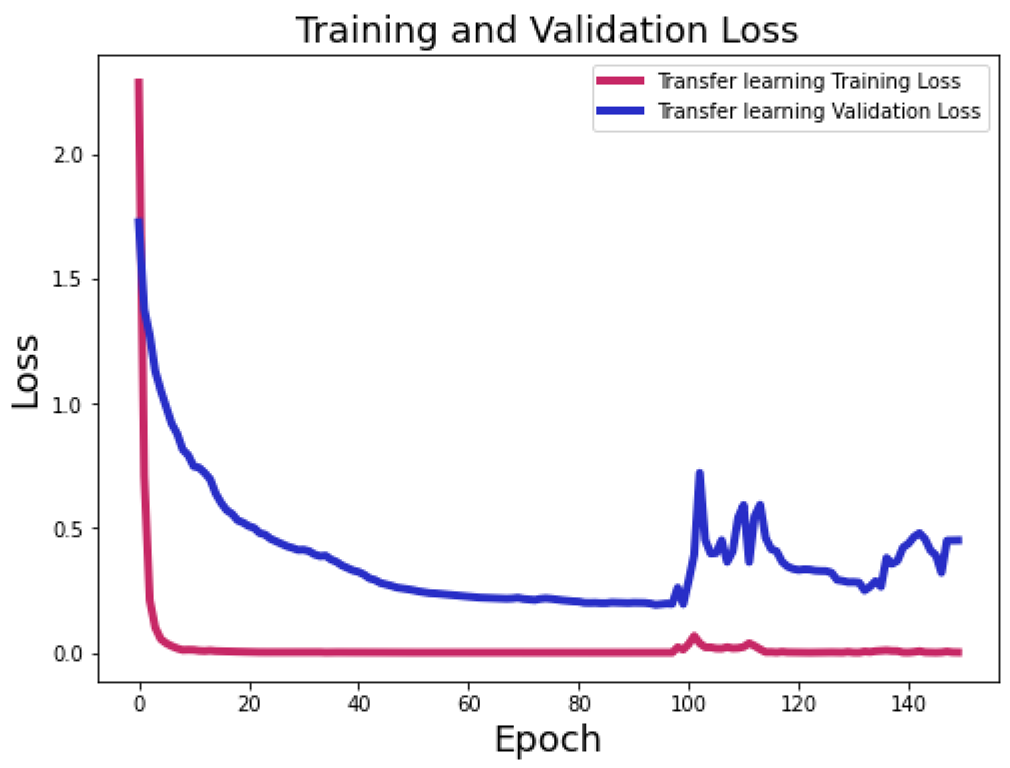
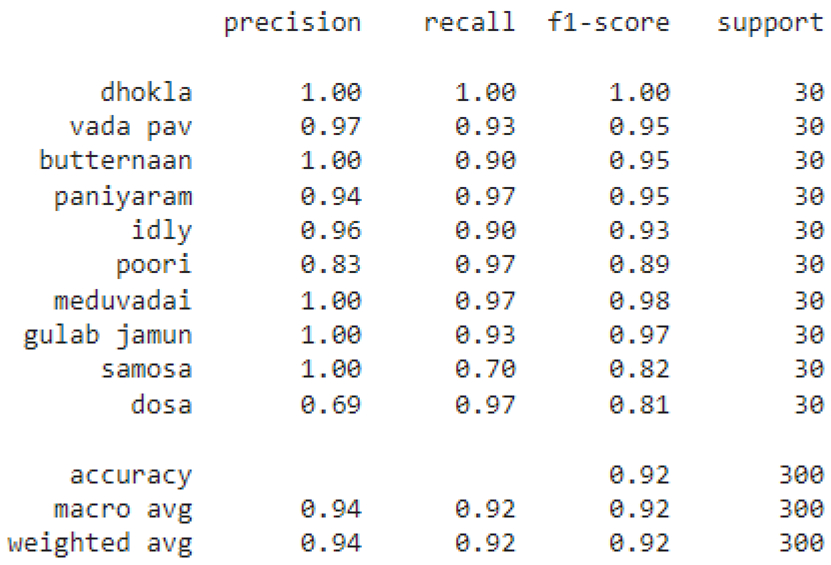


Figure 6.3: Training and Validation Loss graph



```
              precision    recall  f1-score   support

      dhokla       1.00      1.00      1.00        30
    vada pav       0.97      0.93      0.95        30
  butternaan       1.00      0.90      0.95        30
   paniyaram       0.94      0.97      0.95        30
        idly       0.96      0.90      0.93        30
       poori       0.83      0.97      0.89        30
   meduvadai       1.00      0.97      0.98        30
 gulab jamun       1.00      0.93      0.97        30
      samosa       1.00      0.70      0.82        30
        dosa       0.69      0.97      0.81        30

    accuracy                           0.92       300
   macro avg       0.94      0.92      0.92       300
weighted avg       0.94      0.92      0.92       300
```

Figure 6.4: Category wise Precision, Recall and F1-score.

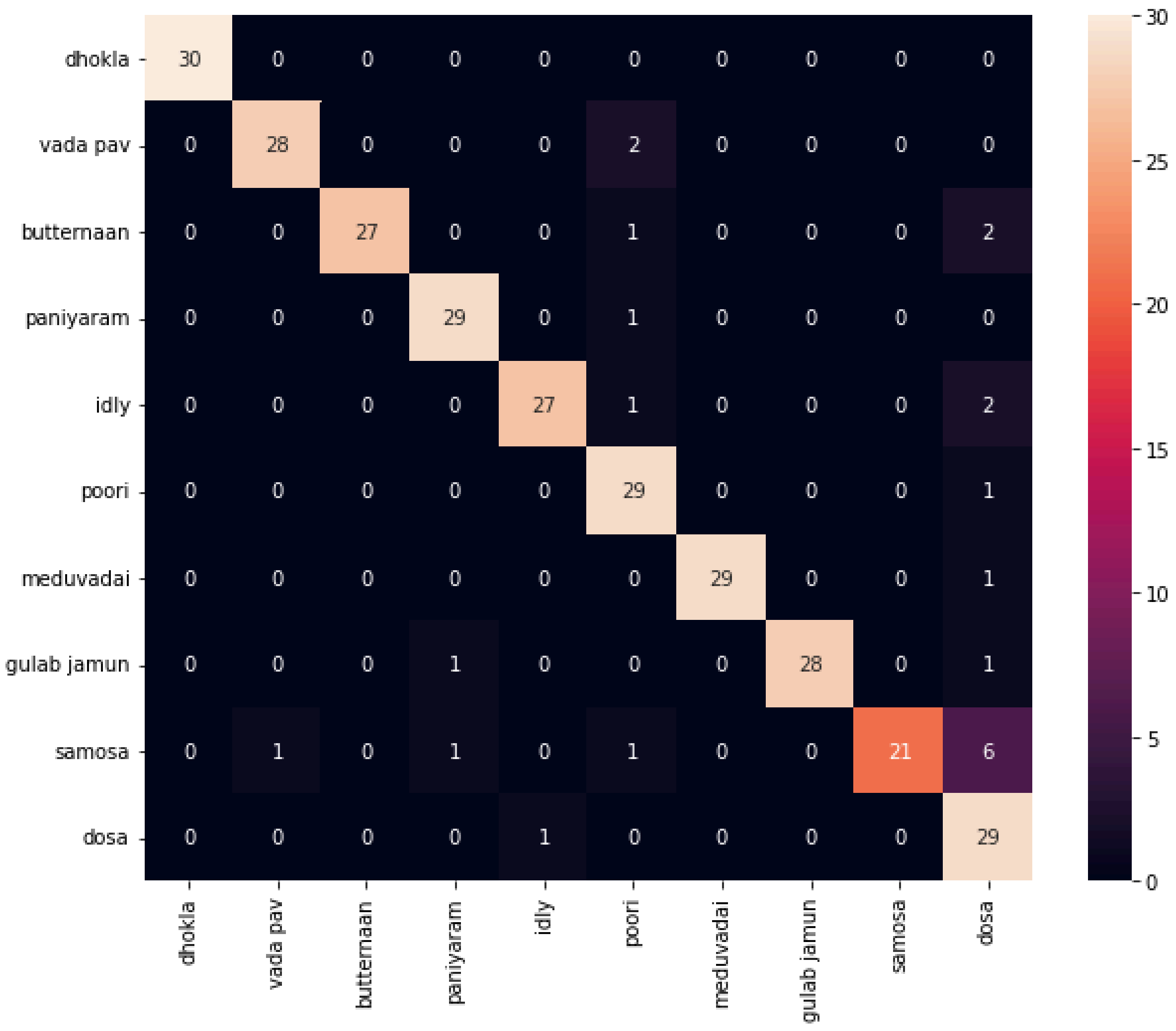
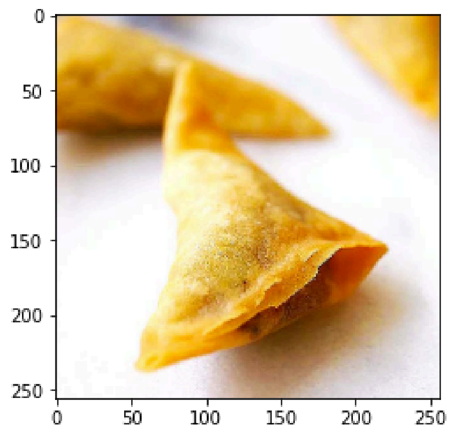


Figure 6.5 : Confusion Matrix



```
9
This food item is Samosa
It has 261 Calories,  3.5 gm Protein,  24 gm Carbohydrates ,  17 gm Fats and  2.1 gm Fibre
```

Figure 6.6: Output image

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

Throughout this project , we have learned a lot , not just technical aspects of the project , but also others .We have repeatedly found that there is still a great need for these kinds of projects. There exist relatively fewer projects and even rarer commercial projects that are still in beta phases and not being launched to a wider audience. No major company in dietary advice and assistance fields has publicly announced any such project. Moreover , despite having more than a billion people , such a solution for Indian food items is nearly non-existent even at the trial level or beta phases. Using newer deep learning technologies and developing a relatively accurate model will lead to a boom to the field.

For our project , despite having relatively good accuracy , there is still a lot more room for improvement. There are many ways for improvement. First we can add more food items , resembling the actual food habits of Indian people. We can also develop an improved algorithm with greater accuracy and lesser computational time. We can deploy this solution to web based solutions (app or website) , so people can use this service.Also we can give user to add food item missing from trained database , to further expand as per users need.We can also add a suggestion features , which will study user's daily and weekly eating habit ,and suggest adding and removing food items as per their personalized nutritional requirement decided by age , gender , BMI , weight , other illness and allergies etc.

References

[1] Yuzhen Lu, “Food Image Recognition by Using Convolutional Neural Networks (CNNs) ” , arxiv.org , 2016

[2]Landu Jiang; Bojia Qiu; Xue Liu; Chenxi Huang; Kunhui Lin , “DeepFood: Food Image Analysis and Dietary Assessment via Deep Model” , in IEEE,2020

[3] Chang Liu , Yu Cao, Yan Luo , Guanling Chen, Vinod Vokkarane, Yunsheng Ma , “DeepFood: Deep Learning-based Food Image Recognition for Computer-aided Dietary Assessment” ,arxiv.org ,2016

[4] Shashi Rekha,“Deep Indian Delicacy: Classification of Indian Food Images using Convolutional Neural Networks” , IJRASET, 2018

[5] https://analyticsindiamag.com/how-i-created-curryai-a-computer-vision-aided-indian-food-nutrition-calculator/

[6] https://medium.com/syncedreview/deep-learning-based-food-calorie-estimation-method-in-dietary-assessment-1e76a2acee7

[5] https://www.kaggle.com/cdart99/food20dataset