



RV Educational Institutions[®]
RV College of Engineering[®]

Autonomous
Institution Affiliated
to Visvesvaraya
Technological
University, Belagavi

Approved by AICTE,
New Delhi, Accredited
By NAAC, Bengaluru
And NBA, New Delhi

Go, change the world

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS



**Machine Learning
18MCA343**

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

MASTER OF COMPUTER APPLICATIONS

By:

**Aditya Sukhwai
USN:1RV19MCA02**

**Juliouish Kumar Das
USN:1RV19MCA37**

**Under the Guidance
of**

**Dr. Andhe Dharani
Professor and Director**

2020-2021

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059



CERTIFICATE

Certified that the Assignment titled **“SNAP N GRAB (Food Recommendation System)”** carried out by **Aditya Sukhwal, Julioush Kumar Das**, USN: **1RV19MCA02, 1RV19MCA37**, bonafide students of **RV College of Engineering, Bengaluru** submitted in partial fulfilment for the award of **Master of Computer Applications** of **RV College of Engineering, Bengaluru** affiliated to **Visvesvaraya Technological University, Belagavi** during the year **2020-21**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the partial academic requirement in respect of the course Machine Learning 18MCA343.

Faculty- In charge

Dr. Andhe Dharani,

Professor and Director,

Department of MCA

RVCE, Bengaluru –

560059

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the success of any work would be incomplete unless we mention the name of the people, who made it possible, whose constant guidance and encouragement served a beacon light and served our effort with success.

We express our whole-hearted gratitude to Dr. K.N. Subramanya, Principal, and R.V. College of Engineering for providing me an opportunity and support to complete the project.

We express our special thanks to Dr. Andhe Dharani., Professor and Director, Department of MCA, R.V. College of Engineering for her constant support and guidance.

On a moral personal note, our deepest appreciation and gratitude to my beloved family and friends, who have been a fountain of inspiration and have provided unrelenting encouragement and support.

Date:

Aditya Sukhwal

1RV19MCA02

Julioush KumarDas

1RV19MCA37

TABLE OF CONTENTS

| SL.NO | DESCRIPTION | PAGE NO |
|-------|--|---------|
| 1.1 | Literature Survey | 1 |
| 1.2 | Tools and technologies used | 2 |
| 1.3 | Data and feature sets considered | 3 |
| 1.4 | Algorithm analysed | 4 |
| 2.1 | Libraries / functions used - details | 7 |
| 2.2 | Model implementation – code part of the algorithms | 10 |
| 2.3 | Performance evaluation and analysis of the model | 13 |
| 2.4 | Screenshots with explanation | 14 |
| 3 | Bibliography | 22 |
| 4 | Outcomes | 23 |

LITERATURE SURVEY

1.1. INTRODUCTION

Snap n Grab is a web application that tracks the user's food intake by pictures. We use state-of-the-art deep learning techniques to recognize dishes, making instant nutrition estimates from the user's meals. In today's world one of the biggest problems, we all homoserines face is related tour health issues, and health issues are directly related to the kind of food consumption we humans have, we do not map out our nutritional value. Using Machine and deep learning we can easily solve this issue to aware people about the nutrition value of the food they are consuming in their daily life. Our aim is to make life of people easier and healthier with appropriate use of the technology, It is not an easy task to develop such a complicated application to make all things go in smooth flow, problem which we faced during this project was how to determine the speed with efficiency, to find which food picture is taken by the user. There is huge image dataset of food 100 pictures, which had to be classified and indexed in order to predict the nutritional value of the food, which picture was taken by the user.

If we want to enjoy junk food once in a while but are concerned about the impact on our health, take a look at our overall health habits. Are we exercising regularly and eating plenty of nutritious foods such as vegetables, fruit, legumes, fish, nuts and seeds, and whole grains? When it comes to our health, it seems we can “get away with” the occasional junk food more easily when we follow a healthy lifestyle most of the time. Eating fast food more than twice a week doubled the incidence of insulin resistance, a risk factor for diabetes, in the 2005 study published in “Lancet,” and a 2010 Harvard report linked sweetened soft drinks with an increased risk of Type 2 diabetes and heart disease. Additionally, the risk of stroke may be related to the number of fast-food restaurants in a neighborhood, according to a study published in the “Annals of Neurology” in 2009. The study found the risk of stroke increased by 1 percent for every fast-food restaurant in a Texas neighborhood. Fast food is loaded with sodium, which increases the risk of high blood pressure and stroke. To reduce fast-food health risk a person needs to look into the nutritional value of that food, that is the only way to determine the healthy future for anybody.

Aim of our project is to help people in easiest way to make them aware about the diet they are having in their life. Nutrients that are needed in large amounts are called macronutrients. There are three classes of macronutrients: carbohydrates, fats and proteins. Macronutrients are carbon-based compounds that can be metabolically processed into cellular energy through changes in their chemical bonds. The chemical energy is converted into cellular energy known as ATP, that is utilized by the body to perform work and conduct basic functions, and by keeping these macronutrients in check we all can develop a healthy lifestyle.

1.1.2 Motivation

According to the **World Health Organization**, worldwide obesity has nearly tripled since 1975. In the United States, almost 75% of the population is overweight and more than half of the population is obese (**OECD**). Today, many diseases that were previously thought as hereditary are now shown to be seen connected to biological disfunction related to nutrition.

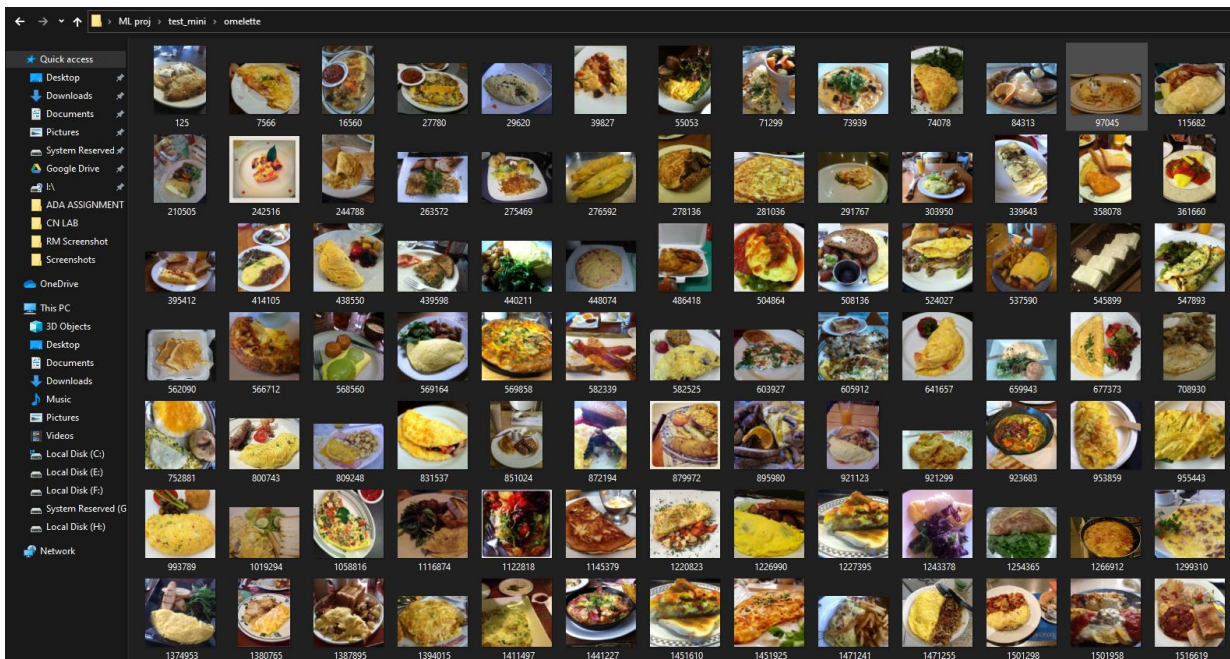
Although being healthy and eating better is something the vast majority of the population want, doing so usually requires great effort and organization. The lack of an easy and simple way to track nutrition information about the food you eat can easily lead to low engagement. By providing a very easy and fun way to keep track of what the user eat, we can largely improve engagement, and directly attack on of the largest health problems in the world.

1.2 Tools and Technology Used.

| CARRIED OUT BY | DATASET USED | TOOL USED | TECHNIQUE USED | ACCURACY RATE |
|--------------------|--|-----------|--------------------------|---------------|
| Aditya Sukhwal | Kaggle Machine Learning Repository 101 food | Python | KNN | 97.802197 |
| Julioush Kumar Das | Kaggle Machine Learning Repository | Python | kNN(k-Nearest Neighbour) | 98.24561 |

1.3 Data and feature sets considered:

- We have used Food 101(food images) to train and test our model. It is around 10.4GB taken from Kaggle (Dataset Repository).It contains images of food, organized by type of food. It was used in the Paper "Food-101 – Mining Discriminative Components with Random Forests. It's a good (large dataset) for testing computer vision techniques.
- We have used 70% of the dataset for training and 30% of the dataset for testing the proposed models.
- The first goal is to be able to automatically classify an unknown image using the dataset, but beyond this there are a number of possibilities for looking at what regions / image components are important for making classifications, identify new types of food as combinations of existing tags, build object detectors which can find similar objects in a full scene.



1000 high resolution images with labels

1.3.1 Features Set Consideration for Nutritional Value :

- We have considered 8 features for Dataset containing the nutritional values or macros of the desired food, which are the name of the food, energy contain of the food, carbohydrates present in the food, amount of sugars present in the food, protein, fat, fiber and cholesterol.
- These features of a food item are present in every food item which we eat, the difference is some food items have more healthy macros like the protein while others have more sugars and foods, that's how a food is defined by its nutritional values
- For example in our dataset omelet 100gm of omelet has 240Kcal, 14gm protein, 4.6gm carbohydrates, 0.7 gram fibers, 18.4gm fat and 0.8mg cholesterol.

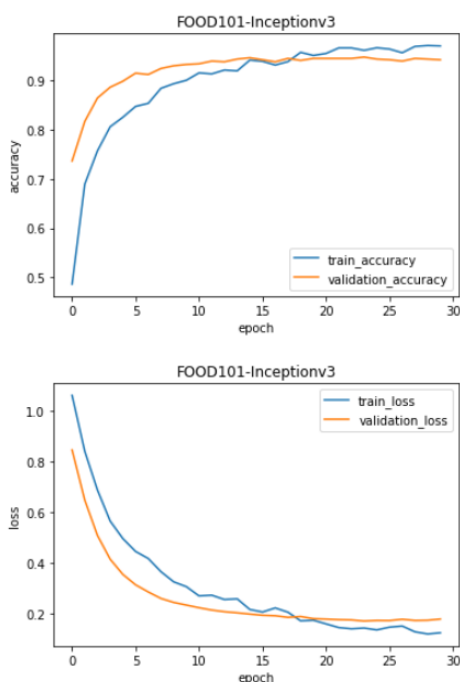
</

1.4 Algorithm analyzed

1.4.1 K-Nearest Neighbours(kNN)

- The kNN algorithm is a simple but extremely powerful classification algorithm. The name of the algorithm originates from the underlying philosophy of kNN – i.e., people having similar background or mindset tend to stay close to each other. In other words, neighbours in a locality have a similar background. In the same way, as a part of the kNN algorithm, the unknown and unlabelled data which comes for a prediction problem is judged on the basis of the training data set elements which are similar to the unknown element. So, the class label of the unknown element is assigned on the basis of the class labels of the similar training data set elements (metaphorically can be considered as neighbors of the unknown element).
- kNN algorithm, the class label of the test data elements is decided by the class label of the training data elements which are neighboring, i.e., similar in nature. Though there are many measures of similarity, the most common approach adopted by kNN to measure similarity between two data elements is Euclidean distance. Considering a very simple data set having two features (say f and f), Euclidean distance between two data elements d and d can be measured by:

```
In [15]: plot_accuracy(history, 'FOOD101-Inceptionv3')
         plot_loss(history, 'FOOD101-Inceptionv3')
```



$$\text{Euclidean distance} = \sqrt{(f_{11} - f_{12})^2 + (f_{21} - f_{22})^2}$$

where f_{11} = value of feature f_1 for data element d_1

f_{12} = value of feature f_1 for data element d_2

f_{21} = value of feature f_2 for data element d_1

f_{22} = value of feature f_2 for data element d_2

Strengths of the kNN algorithm

- Extremely simple algorithm – easy to understand
- Very effective in certain situations, e.g., for recommender system design.
- Very fast or almost no time required for the training phase

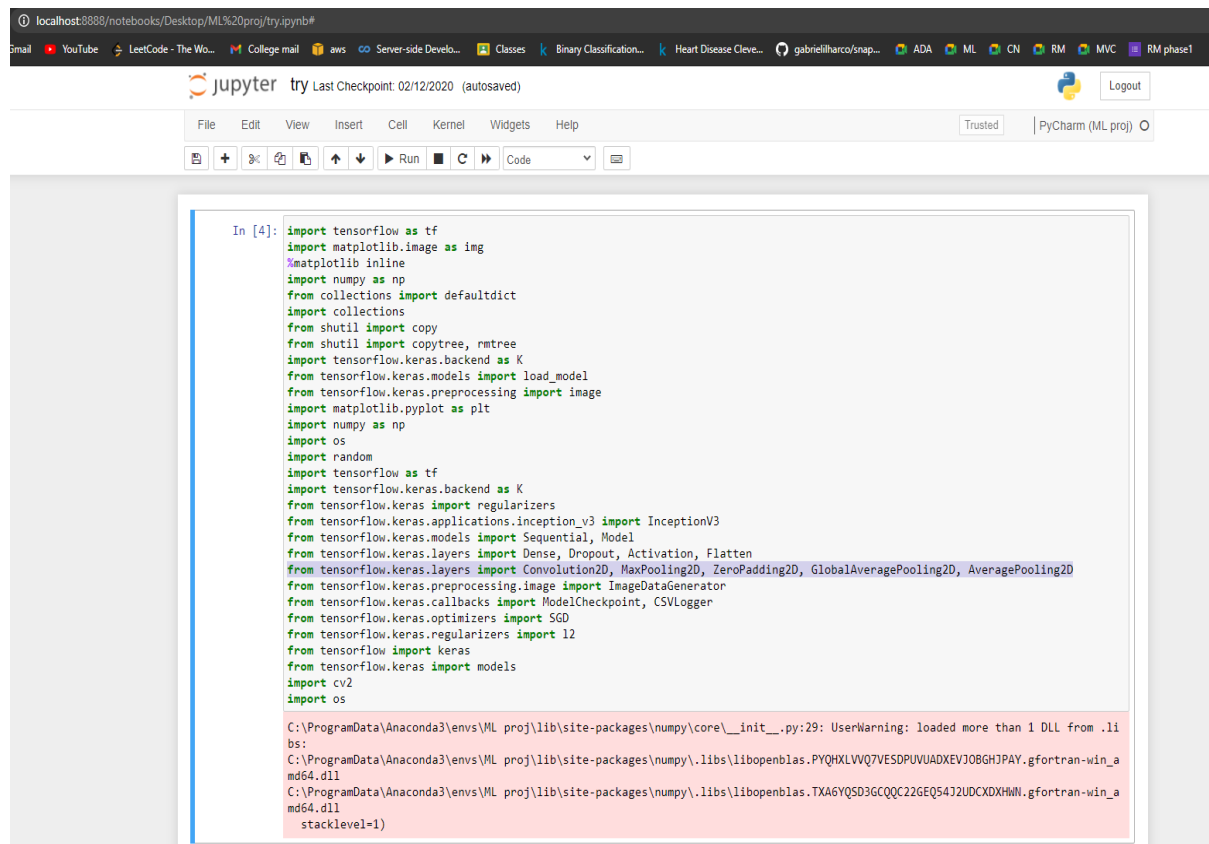
1.4.2 Weaknesses of the kNN algorithm

- Does not learn anything in the real sense. Classification is done completely on the basis of the training data. So, it has a heavy reliance on the training data. If the training data does not represent the problem domain comprehensively.
- The algorithm fails to make an effective classification. Because there is no model trained in real sense and the classification is done completely on the basis of the training data, the classification process is very slow. Also, a large amount of computational space is required to load the training data for classification.

1.4.3 Application of the kNN algorithm

- One of the most popular areas in machine learning where the kNN algorithm is widely adopted is recommender systems. As we know, recommender systems recommend users different items which are similar to a particular item that the user seems to like.
- The liking pattern may be revealed from past purchases or browsing history and the similar items are identified using the kNN algorithm. Another area where there is widespread adoption of kNN is searching documents/ contents similar to a given document/content. This is a core area under information retrieval and is known as concept search.

2.1 Libraries / functions used – details



```
In [4]: import tensorflow as tf
import matplotlib.image as img
import matplotlib inline
import numpy as np
from collections import defaultdict
import collections
from shutil import copy
from shutil import copytree, rmtree
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
import os
import random
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D, AveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow import keras
from tensorflow.keras import models
import cv2
import os

C:\ProgramData\Anaconda3\envs\ML proj\lib\site-packages\numpy\core\_init_.py:29: UserWarning: loaded more than 1 DLL from .libs:
C:\ProgramData\Anaconda3\envs\ML proj\lib\site-packages\numpy\.libs\libopenblas.PYQHXLVWQ7VESDPUVUADXEKJ0BGHJPAY.gfortran-win_a
md64.dll
C:\ProgramData\Anaconda3\envs\ML proj\lib\site-packages\numpy\.libs\libopenblas.TXA6VQSD3GCQC22GEQ54J2UDCXDXHWN.gfortran-win_a
md64.dll
stacklevel=1)
```

Tensor flow:

- TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models. See the TensorFlow documentation for complete details on the broader TensorFlow system.
- TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use the low-level APIs to create and explore new machine learning algorithms. In this class, you will use a high-level API named `tf.keras` to define and train machine learning models and to make predictions. `tf.keras` is the TensorFlow variant of the open-source Keras API.

Matplotlib:

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.
-

Numpy:

- NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Shutil:

- **Shutil module** in Python provides many functions of high-level operations on files and collections of files. It comes under Python's standard utility modules. This module helps in automating process of copying and removal of files and directories.
- *shutil.copy()* method in Python is used to copy the content of *source* file to *destination* file or directory. It also preserves the file's permission mode but other metadata of the file like the file's creation and modification times is not preserved.

OS module:

- This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see `open()`, if you want to manipulate paths, see the `os.path` module, and if you want to read all the lines in all the files on the command line see the file input module. For creating temporary files and directories see the `tempfile` module, and for high-level file and directory handling see the `shutil` module.

Random Module:

- The functions supplied by this module are actually bound methods of a hidden instance of the random. Random class. You can instantiate your own instances of Random to get generators that don't share state.
- Class Random can also be subclassed if you want to use a different basic generator of your own devising: in that case, override the random(), seed(), getstate(), and setstate() methods. Optionally, a new generator can supply a getrandbits() method — this allows randrange() to produce selections over an arbitrarily large range.
- The random module also provides the System Random class which uses the system function os.urandom() to generate random numbers from sources provided by the operating system.

2.2 Model implementation – code part of the algorithms

#Modules imported

```
import tensorflow as tf
import matplotlib.image as img
%matplotlib inline
import numpy as np
from collections import defaultdict
import collections
from shutil import copy
from shutil import copytree, rmtree
import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
import os
import random
```

```

import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D,
GlobalAveragePooling2D, AveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow import keras
from tensorflow.keras import models
import cv2
import os

```

Helper function to download data and extract

```

def get_data_extract():
    if "food-101" in os.listdir():
        print("Dataset already exists")
    else:
        print("Downloading the data...")
        !wget http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz
        print("Dataset downloaded!")
        print("Extracting data..")
        !tar xzvf food-101.tar.gz
        print("Extraction done!")

```

Helper method to split dataset into train and test folders

```

def prepare_data(filepath, src, dest):
    classes_images = defaultdict(list)
    with open(filepath, 'r') as txt:
        paths = [read.strip() for read in txt.readlines()]
        for p in paths:
            food = p.split('/')
            classes_images[food[0]].append(food[1] + '.jpg')

```

```

for food in classes_images.keys():
    print("\nCopying images into ",food)
    if not os.path.exists(os.path.join(dest,food)):
        os.makedirs(os.path.join(dest,food))
    for i in classes_images[food]:
        copy(os.path.join(src,food,i), os.path.join(dest,food,i))
print("Copying Done!")

```

Visualize the data, showing one image per class from 101 classes

```

rows = 17
cols = 6
fig, ax = plt.subplots(rows, cols, figsize=(25,25))
fig.suptitle("Showing one random image from each class", y=1.05, fontsize=24) # Adding
y=1.05, fontsize=24 helped me fix the supitle overlapping with axes issue
data_dir = r"C:\Users\adity\Desktop\ML proj\food-101\images"
foods_sorted = sorted(os.listdir(data_dir))
food_id = 0
for i in range(rows):
    for j in range(cols):
        try:
            food_selected = foods_sorted[food_id]
            food_id += 1
        except:
            break
        if food_selected == '.DS_Store':
            continue
        food_selected_images = os.listdir(os.path.join(data_dir,food_selected)) # returns the list of
all files present in each food category
        food_selected_random = np.random.choice(food_selected_images) # picks one food item
from the list as choice, takes a list and returns one random item
        img = plt.imread(os.path.join(data_dir,food_selected, food_selected_random))
        ax[i][j].imshow(img)
        ax[i][j].set_title(food_selected, pad = 10)

plt.setp(ax, xticks=[],yticks=[])
plt.tight_layout()

```

Create the training and testing dataset

#Training data = Is the subset of our data used to train our

model.

```
food_list = ['apple_pie','pizza','omelette']
```

```
src_train = r'C:\Users\adity\Desktop\ML proj\food-101\train'
```

```
dest_train = 'train_mini'
```

```
print("Creating train data folder with new classes")
```

```
dataset_mini(food_list, src_train, dest_train)
```

#Testing data = Is the subset of our data that the model hasn't seen before.

```
src_test = r'C:\Users\adity\Desktop\ML proj\food-101\test'
```

```
dest_test = 'test_mini'
```

```
print("Creating test data folder with new classes")
```

```
dataset_mini(food_list, src_test, dest_test)
```

#Model_training

```
K.clear_session()
```

```
n_classes = 3
```

```
img_width, img_height = 299, 299
```

```
train_data_dir = 'train_mini'
```

```
validation_data_dir = 'test_mini'
```

```
nb_train_samples = 2250 #75750
```

```
nb_validation_samples = 750 #25250
```

```
batch_size = 16
```

```
train_datagen = ImageDataGenerator(  
    rescale=1. / 255,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True)
```

```
test_datagen = ImageDataGenerator(rescale=1. / 255)
```

```
train_generator = train_datagen.flow_from_directory(
```



```

train_data_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

inception = InceptionV3(weights='imagenet', include_top=False)
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(128,activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(3,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)

model = Model(inputs=inception.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy',
metrics=['accuracy'])
checkpointer = ModelCheckpoint(filepath='best_model_3class.hdf5', verbose=1,
save_best_only=True)
csv_logger = CSVLogger('history_3class.log')

history = model.fit_generator(train_generator,
    steps_per_epoch = nb_train_samples // batch_size,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
    epochs=30,
    verbose=1,
    callbacks=[csv_logger, checkpointer])

model.save('model_trained_3class.hdf5')

#Generating classes index for clusters

class_map_3 = train_generator.class_indices
class_map_3

2.3 Performance evaluation and analysis of the model

#Plotting accuracy graph

```

```

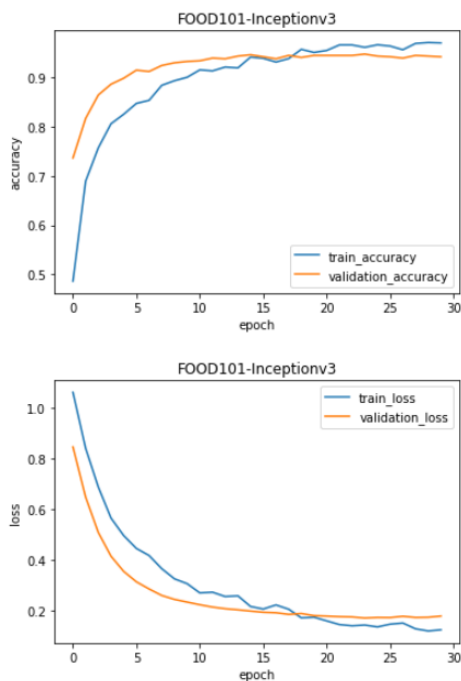
def plot_accuracy(history,title):
    plt.title(title)
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train_accuracy', 'validation_accuracy'], loc='best')
    plt.show()
def plot_loss(history,title):
    plt.title(title)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train_loss', 'validation_loss'], loc='best')
    plt.show()
plot_accuracy(history,'FOOD101-Inceptionv3')
plot_loss(history,'FOOD101-Inceptionv3')

```

```

In [15]: plot_accuracy(history,'FOOD101-Inceptionv3')
         plot_loss(history,'FOOD101-Inceptionv3')

```



Loading the best saved model to make predictions

```

K.clear_session()
model_best = load_model('best_model_3class.hdf5',compile = False)

```

#Predicting class

```
def predict_class(model, images, show = True):  
    for img in images:  
        img = image.load_img(img, target_size=(299, 299))  
        img = image.img_to_array(img)  
        img = np.expand_dims(img, axis=0)  
        img /= 255.  
  
        pred = model.predict(img)  
        index = np.argmax(pred)  
        food_list.sort()  
        pred_value = food_list[index]  
        if show:  
            plt.imshow(img[0])  
            plt.axis('off')  
            plt.title(pred_value)  
            plt.show()
```

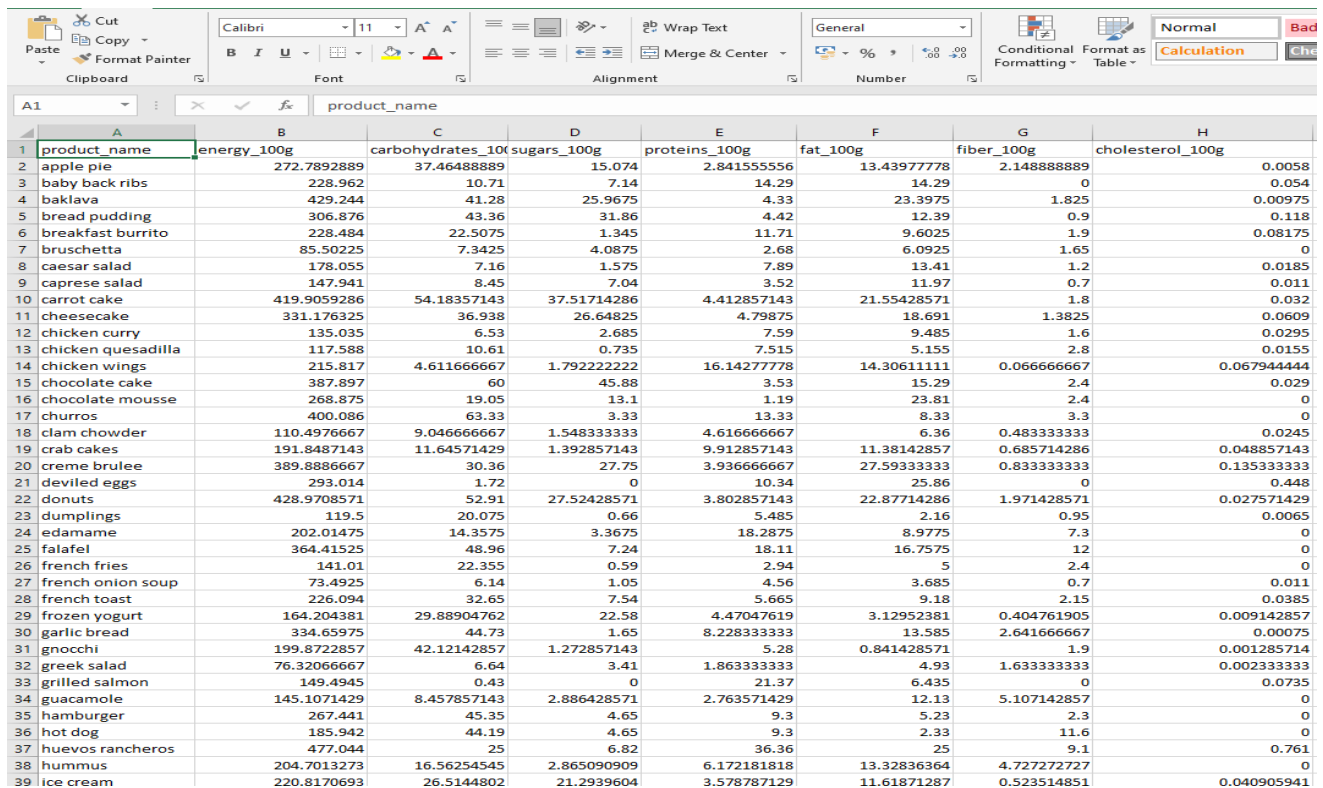
Make a list of downloaded images and test the trained model

Defining path to predict and check whether the image matches with the predicted class or not

```
images = []  
images.append(r'C:\Users\adity\Desktop\applepie.png')  
images.append(r'C:\Users\adity\Desktop\pizza1.jpeg')  
images.append(r'C:\Users\adity\Desktop\omelette1.jpeg')  
predict_class(model_best, images, True)
```

2.4 Screenshots with explanation

Step 1: Download Nutrition values dataset from Kaggle:



| | A | B | C | D | E | F | G | H |
|----|--------------------|-------------|--------------------|-------------|---------------|-------------|-------------|------------------|
| | product_name | energy_100g | carbohydrates_100g | sugars_100g | proteins_100g | fat_100g | fiber_100g | cholesterol_100g |
| 1 | apple pie | 272.7892889 | 37.46488889 | 15.074 | 2.841555556 | 13.43977778 | 2.148888889 | 0.0058 |
| 2 | baby back ribs | 228.962 | 10.71 | 7.14 | 14.29 | 14.29 | 0 | 0.054 |
| 3 | baklava | 429.244 | 41.28 | 25.9675 | 4.33 | 23.3975 | 1.825 | 0.00975 |
| 4 | bread pudding | 306.876 | 43.36 | 31.86 | 4.42 | 12.39 | 0.9 | 0.118 |
| 5 | breakfast burrito | 228.484 | 22.5075 | 1.345 | 11.71 | 9.6025 | 1.9 | 0.08175 |
| 6 | bruschetta | 85.50225 | 7.3425 | 4.0875 | 2.68 | 6.0925 | 1.65 | 0 |
| 7 | caesar salad | 178.055 | 7.16 | 1.575 | 7.89 | 13.41 | 1.2 | 0.0185 |
| 8 | caprese salad | 147.941 | 8.45 | 7.04 | 3.52 | 11.97 | 0.7 | 0.011 |
| 9 | carrot cake | 419.9059286 | 54.18357143 | 37.51714286 | 4.412857143 | 21.55428571 | 1.8 | 0.032 |
| 10 | cheesecake | 331.176325 | 36.938 | 26.64825 | 4.79875 | 18.691 | 1.3825 | 0.0609 |
| 11 | chicken curry | 135.035 | 6.53 | 2.685 | 7.59 | 9.485 | 1.6 | 0.0295 |
| 12 | chicken quesadilla | 117.588 | 10.61 | 0.735 | 7.515 | 5.155 | 2.8 | 0.0155 |
| 13 | chicken wings | 215.817 | 4.611666667 | 1.792222222 | 16.14277778 | 14.30611111 | 0.066666667 | 0.067944444 |
| 14 | chocolate cake | 387.897 | 60 | 45.88 | 3.53 | 15.29 | 2.4 | 0.029 |
| 15 | chocolate mousse | 268.875 | 19.05 | 13.1 | 1.19 | 23.81 | 2.4 | 0 |
| 16 | churros | 400.086 | 63.33 | 3.33 | 13.33 | 8.33 | 3.3 | 0 |
| 17 | clam chowder | 110.4976667 | 9.046666667 | 1.548333333 | 4.616666667 | 6.36 | 0.483333333 | 0.0245 |
| 18 | crab cakes | 191.8487143 | 11.64571429 | 1.392857143 | 9.912857143 | 11.38142857 | 0.685714286 | 0.048857143 |
| 19 | creme brulee | 389.8886667 | 30.36 | 27.75 | 3.936666667 | 27.59333333 | 0.833333333 | 0.135333333 |
| 20 | deviled eggs | 293.014 | 1.72 | 0 | 10.34 | 25.86 | 0 | 0.448 |
| 21 | donuts | 428.9708571 | 52.91 | 27.52428571 | 3.802857143 | 22.87714286 | 1.971428571 | 0.027571429 |
| 22 | dumplings | 119.5 | 20.075 | 0.66 | 5.485 | 2.16 | 0.95 | 0.0065 |
| 23 | edamame | 202.01475 | 14.3575 | 3.3675 | 18.2875 | 8.9775 | 7.3 | 0 |
| 24 | falafel | 364.41525 | 48.96 | 7.24 | 18.11 | 16.7575 | 12 | 0 |
| 25 | french fries | 141.01 | 22.355 | 0.59 | 2.94 | 5 | 2.4 | 0 |
| 26 | french onion soup | 73.4925 | 6.14 | 1.05 | 4.56 | 3.685 | 0.7 | 0.011 |
| 27 | french toast | 226.094 | 32.65 | 7.54 | 5.665 | 9.18 | 2.15 | 0.0385 |
| 28 | frozen yogurt | 164.204381 | 29.88904762 | 22.58 | 4.47047619 | 3.12952381 | 0.404761905 | 0.009142857 |
| 29 | garlic bread | 334.65975 | 44.73 | 1.65 | 8.228333333 | 13.585 | 2.641666667 | 0.00075 |
| 30 | gnocchi | 199.8722857 | 42.12142857 | 1.272857143 | 5.28 | 0.841428571 | 1.9 | 0.001285714 |
| 31 | greek salad | 76.32066667 | 6.64 | 3.41 | 1.863333333 | 4.93 | 1.633333333 | 0.002333333 |
| 32 | grilled salmon | 149.4945 | 0.43 | 0 | 21.37 | 6.435 | 0 | 0.0735 |
| 33 | guacamole | 145.1071429 | 8.457857143 | 2.886428571 | 2.763571429 | 12.13 | 5.107142857 | 0 |
| 34 | hamburger | 267.441 | 45.35 | 4.65 | 9.3 | 5.23 | 2.3 | 0 |
| 35 | hot dog | 185.942 | 44.19 | 4.65 | 9.3 | 2.33 | 11.6 | 0 |
| 36 | huevos rancheros | 477.044 | 25 | 6.82 | 36.36 | 25 | 9.1 | 0.761 |
| 37 | hummus | 204.7013273 | 16.56254545 | 2.865090909 | 6.172181818 | 13.32836364 | 4.727272727 | 0 |
| 38 | ice cream | 220.8170693 | 26.5144802 | 21.2939604 | 3.578787129 | 11.61871287 | 0.523514851 | 0.040905941 |

Step 2: Download Food-101 dataset from Kaggle:



Step:3 Creating training and testing dataset:

```
jupyter try Last Checkpoint: 02/12/2020 (autosaved) PyCharm (ML proj) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted

In [8]: # Helper method to create train_mini and test_mini data samples
def dataset_mini(food_list, src, dest):
    if os.path.exists(dest):
        rmtree(dest) # removing dataset_mini(if it already exists) folders so that we will have only the classes that we want
    os.makedirs(dest)
    for food_item in food_list :
        print("Copying images into",food_item)
        copytree(os.path.join(src,food_item), os.path.join(dest,food_item))

In [9]: food_list = ['apple_pie','pizza','omelette']
src_train = r'C:\Users\adity\Desktop\ML proj\food-101\train'
dest_train = 'train_mini'
src_test = r'C:\Users\adity\Desktop\ML proj\food-101\test'
dest_test = 'test_mini'

In [10]: print("Creating train data folder with new classes")
dataset_mini(food_list, src_train, dest_train)

Creating train data folder with new classes
Copying images into apple_pie
Copying images into pizza
Copying images into omelette

In [11]: print("Creating test data folder with new classes")
dataset_mini(food_list, src_test, dest_test)

Creating test data folder with new classes
Copying images into apple_pie
Copying images into pizza
Copying images into omelette
```

Step:4 Training the model:

It took 35 hours+ to train the model with the images of 3 classes, there were total 750 images, 250 images for each class, omelet, apple pie and pizza.

```
inception = InceptionV3(weights='imagenet', include_top=False)
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(3, kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)

model = Model(inputs=inception.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
checkpointer = ModelCheckpoint(filepath='best_model_3class.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('history_3class.log')

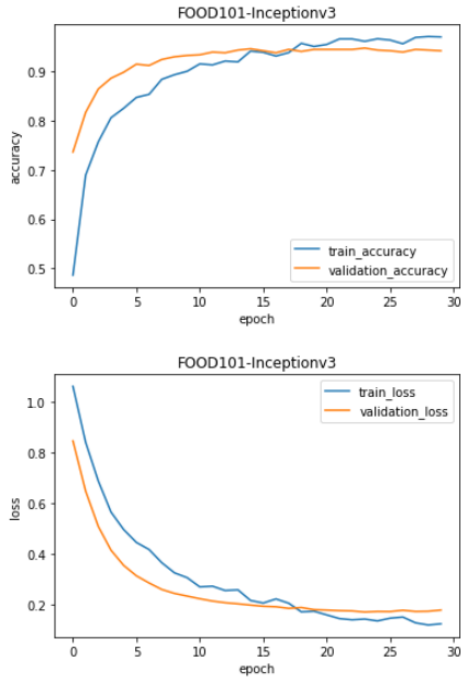
history = model.fit_generator(train_generator,
                             steps_per_epoch = nb_train_samples // batch_size,
                             validation_data=validation_generator,
                             validation_steps=nb_validation_samples // batch_size,
                             epochs=30,
                             verbose=1,
                             callbacks=[csv_logger, checkpointer])

model.save('model_trained_3class.hdf5')
```

```
Found 2250 images belonging to 3 classes.
Found 750 images belonging to 3 classes.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\envs\ML proj\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling
VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future ve
rsion.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Epoch 1/30
139/140 [=====>.] - ETA: 13s - loss: 1.0597 - acc: 0.4865
Epoch 00001: val_loss improved from inf to 0.84501, saving model to best_model_3class.hdf5
140/140 [=====] - 2177s 16s/step - loss: 1.0598 - acc: 0.4866 - val_loss: 0.8450 - val_acc: 0.7364
Epoch 2/30
139/140 [=====>.] - ETA: 12s - loss: 0.8402 - acc: 0.6889
Epoch 00002: val_loss improved from 0.84501 to 0.64766, saving model to best_model_3class.hdf5
140/140 [=====] - 1994s 14s/step - loss: 0.8396 - acc: 0.6893 - val_loss: 0.6477 - val_acc: 0.8166
Epoch 3/30
139/140 [=====>.] - ETA: 12s - loss: 0.6862 - acc: 0.7583
Epoch 00003: val_loss improved from 0.64766 to 0.50851, saving model to best_model_3class.hdf5
140/140 [=====] - 2032s 15s/step - loss: 0.6873 - acc: 0.7569 - val_loss: 0.5085 - val_acc: 0.8641
Epoch 4/30
```

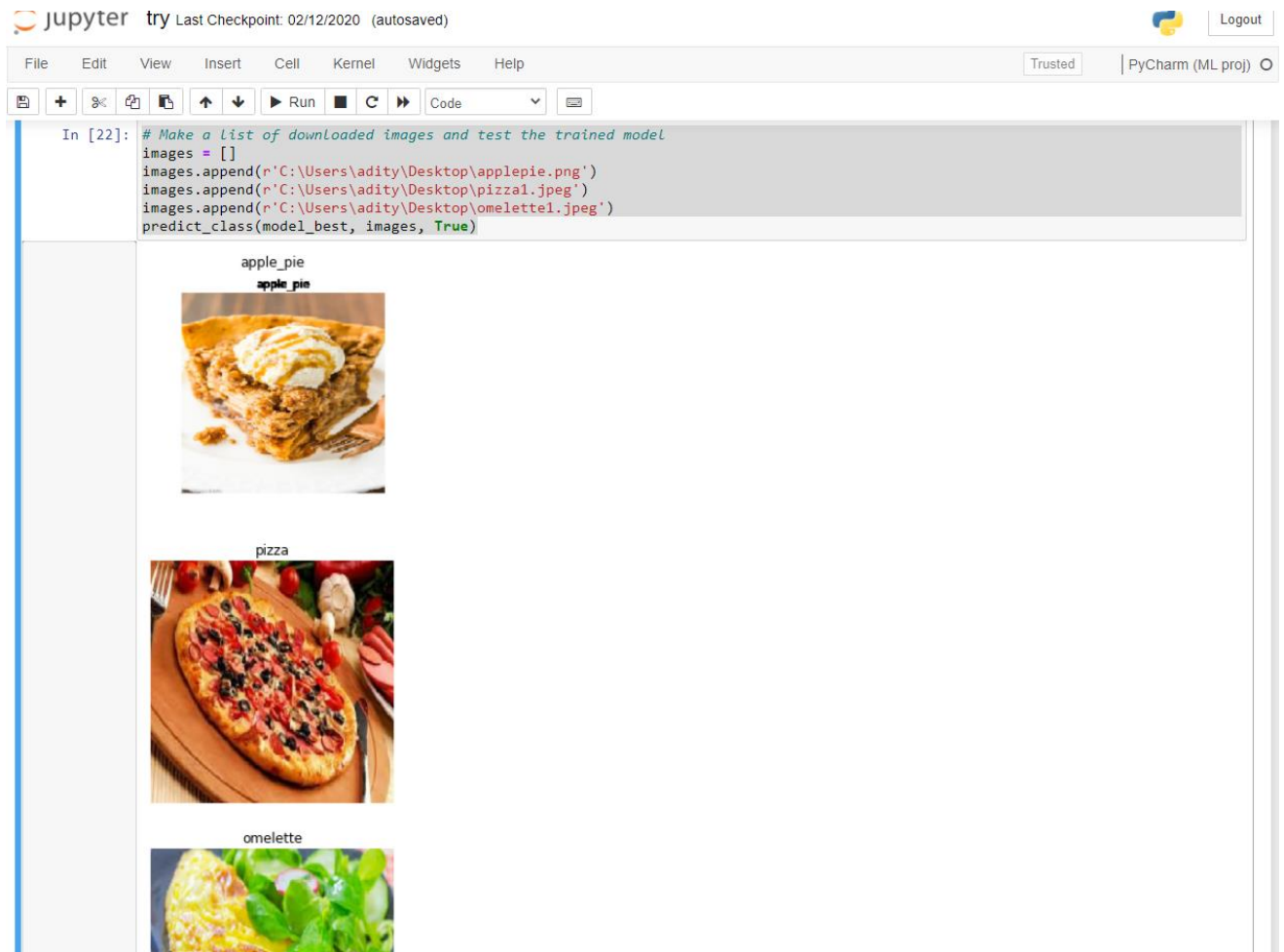
Step 5: Plotting accuracy graph using matplotlib for trained model.

```
In [15]: plot_accuracy(history, 'FOOD101-Inceptionv3')  
plot_loss(history, 'FOOD101-Inceptionv3')
```



The accuracy of our trained model with test dataset came out to be 97.3% accurate with respect to given test dataset for the prediction of food picture. While the predicted loss for distorted image came out to be less than $<0.2\%$, which was better than our expectations. With such high accuracy this model has proved itself to be very valuable in terms of time, efficiency and accuracy.

Step 6: Predicting the image with model by using downloaded images.



In this final stage, we give a path for a described picture as shown, and then the prediction about the food present in the picture is generated. In case if the food picture doesn't match with any of the classified food image the output is not generated for that picture.

3) Bibliography:

<https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow/toolkit>

<https://www.analyticsvidhya.com/blog/2019/08/3-techniques-extract-features-from-image-data-machine-learning-python/>

<https://www.kaggle.com/theimgclist/multiclass-food-classification-using-tensorflow>

<https://www.kaggle.com/dansbecker/food-101>

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>