كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
Helwan University

HELWAN UNIVERSITY
Faculty of Computers and Artificial Intelligence
Computer Science Department

Slim
FOR A HEALTHIER LIFE

A graduation project dissertation by:

[Mariam Abdalmageid Abdalsalam (201900802)]

[Rawan Mohamed Abdelmohsen (201900312)]

[Omnia Mohamed Ahmed (201900176)]

[Nada Salah NourEldin (201900892)]

[Ahmed Ashraf Saeed (201900015)]

[Abdelrahman Ahamd Sayed(201900400)]

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computers & Artificial Intelligence, at the Computer Science Department, the Faculty of Computers & Artificial Intelligence, Helwan University

Supervised by:

Dr/ Ahmad Hesham Mostafa

June 2023

Acknowledgment

After our work, we would like to thank **Dr. Ahmad Hesham Mostafa** for allowing us to work on the graduation project under His supervision.

He also supported from the beginning by presenting to that idea and the technology used in it, not leaving, and always directing us to reach the next step in research and correct our endeavor path during the project.

We thank Him for His time with us to advise us and listen to the problems we face and try to find solutions to them.

And of course, we don't forget to thank our families, especially our parents for their unconditional support mentally, financially, encouragement, patience, and help over the years. We are forever indebted to our parents, who always kept us in their prayers and pushed us to success.

We also thank our friends and colleagues for their support and feedback. Finally, for our faculty for providing a suitable environment that led us to represent the best image that computer science graduates of Helwan University are supposed to represent.

Table of content

List of Figures

List Of Tables

Abstract

This project serves People who want to follow a healthier life . This is because they face many problems, including the distance between him and his dietitian, and that going requires a lot of time, effort, and money. Also, the one don't know what calories he should take a day and to know food calories and performing some exercises alone without the need of GYM .

And for all these reasons that are mentioned, the solution lies in creating a tool or program that helps people to Follow their diet regularly and correctly and from their home without the trouble of going to dietitians.

And this is what will be offered in This project, which helps guide people in the correct way to follow a diet and a healthier Lifestyle; So that he won't need a dietitian.

Food-101 dataset consists of 101 food categories with 750 training and 250 test images per category, making a total of 101k images. The labels for the test images have been manually cleaned, while the training set contains some noise.

in short, the user has to upload a picture of his meal from those in the dataset that was mentioned, and then the program performs its turn where it works on the picture that was entered by the user , and start detecting what kind of food is this after that calculating its calories, all this is in the output to the user.

Keywords

Object detection,Calorie estimation,Food composition analysis,Food Recognition , Food Detection , API,

Chapter 1: Introduction

In this chapter, will provide a Brief Introduction about our problem, and what motivates us for choosing it, i.e., why this problem is important. etc. all were discussed in the Motivation section, then will mention the Problem Definition that informs about challenges that this project aims to solve, also showed Similar System Information, and finally the Overview of the rest of the document.

1.1 INTRO

In general, following a diet is a very important healthy habit in our lives, and as popular the   word diet is not for people who just want to loose weight , so that’s why there are so many dietitian all around the world ,from which he can follow the (Patient , Athlete , one) plan give him exercises and so on.

1.2 MOTIVATION

Why is this problem important?

diet is important because it provides the body with essential nutrients, helps maintain a healthy weight, reduces the risk of chronic diseases, boosts the immune system, and improves mental health. Eating a balanced diet that is rich in fruits, vegetables, whole grains, and lean protein can help promote overall health and well-being.

1.3 PROBLEM DEFINITION

There are several difficulties that people may face when trying to follow a diet:

1-Lack of motivation: Sticking to a diet can be challenging, especially if the individual lacks motivation. This can be due to a variety of reasons such as unrealistic goals, lack of social support, or a negative attitude towards healthy eating.

2-Busy lifestyle: People with busy lifestyles may find it difficult to follow a diet because they do not have enough time to plan and prepare healthy meals. This can lead to relying on convenience foods that are often high in calories, fat, and sugar.

3-Temptations and cravings: Temptations and cravings for unhealthy foods can make it difficult to stick to a diet. This can be caused by factors such as stress, boredom, or social pressure.

4-Lack of knowledge: Some individuals may lack knowledge about healthy eating and may not know how to plan and prepare healthy meals. This can make it difficult to follow a diet, as they may not know what foods to eat or how to prepare them.

5-Medical conditions: Certain medical conditions, such as food allergies or gastrointestinal disorders, can make it difficult to follow a diet. In some cases, individuals may need to follow a specific diet to manage their condition, which can be challenging.

Overall, following a diet can be challenging for many reasons, but with the right support and strategies, individuals can overcome these difficulties and maintain a healthy diet and that could be done with out Application .

1.4 SIMILAR SYSTEM

1.4.1-Smart scales:
These scales are equipped with a camera and a built-in database of foods. When a user places food on the scale, the camera captures an image of the food, and the system uses image recognition technology to identify the food and calculate its nutritional content.

1.4.2-Smart kitchen appliances:
Some kitchen appliances, such as smart ovens and refrigerators, are equipped with cameras and sensors that can detect food and calculate its nutritional content. For example, a smart refrigerator might be able to detect the type and quantity of food stored inside and provide nutritional information in real-time.

1.4.3-Nutrino - Nutrino is a food and nutrition platform that uses machine learning to provide personalized nutrition insights. The platform allows users to log their food intake using a mobile app, and uses machine learning algorithms to analyze the nutritional content of the food and provide personalized recommendations. Nutrino also integrates with wearable devices and fitness apps to provide a more comprehensive view of the user's health and wellness.

1.4.4- MyFitnessPal - MyFitnessPal is a popular food and exercise tracking app that allows users to log their food intake and exercise routines. The app has a large database of foods and their nutritional content, and uses this information to estimate the calorie content of the food. MyFitnessPal also integrates with fitness trackers and other health apps to provide a more complete picture of the user's health.

1.4.5 - Calory - Calory is a food tracking app that uses image recognition technology to identify the food in a photo and estimate its calorie content. The app allows users to take a photo of their food, and uses machine learning algorithms to analyze the image and estimate the calorie content. Calory also has a database of foods and their nutritional content, which it uses to provide more accurate calorie estimates.

1.5 OVERVIEW OF DOCUMENT

In the rest of the document, will discuss in chapter 2 the details of our project, its components, and the way it works, in chapter 3 will list related work, and background, and   in chapter 4 will present the system design, and will conclude in chapter 5 with system implementation and results.

Chapter 2: Topic Overview (Methodology)

In this chapter, will get acquainted with the details of building our system (Food And Calories Detection aka SLIM) theoretically, by mentioning the details of all the phases that make it up. From **Food Recognition Model , Food Detection Model ,Using Yummly API**

2.1 Food Recognition

2.1.1 Food Recognition using MobileNetV2

Food recognition is the task of identifying and classifying food items in images and videos. Food recognition models can be used in a variety of applications, such as recipe suggestions, food tracking, and food quality inspection.

This section describes a food recognition model that uses the MobileNetv2 convolutional neural network (CNN) as a base model. The MobileNetv2 CNN is pre-trained on the ImageNet dataset, which contains over 1.2 million images of 1000 different object categories. The MobileNetv2 CNN is a small and efficient model that can be used for mobile and embedded devices.

2.2 Food Detection

2.2.1 Food detection using YOLOv8 model

Food detection is the task of identifying and localizing food items in images and videos. Food detection models can be used in a variety of applications, such as recipe suggestions, food tracking, and food quality inspection.

This section describes a food detection model that uses the YOLOv8 object detection model. YOLOv8 is a single-shot object detection model that can detect multiple objects in a single image. YOLOv8 is a fast and accurate model that can be used for real-time applications.

2.3 Yummly API

2.3.1 What Is Yummly API ?

Yummly API is a REST API that provides access to the recipe database of Yummly, a popular recipe search engine. The Yummly API provides developers with the ability to search for recipes based on various criteria such as ingredients, dietary restrictions, cuisine, and more. The API also allows developers to retrieve recipe details such as ingredients, directions, nutritional information, and more.

The Yummly API is available on Rapid API, a popular API marketplace that provides access to a variety of APIs from different vendors. Developers can use the Yummly API by sending HTTP requests to the Yummly API endpoints.

The Yummly API provides several endpoints that allow developers to retrieve recipe information. The 'search' endpoint allows developers to search for recipes based on keywords, ingredients, dietary restrictions, and more. the Yummly API also provides a 'feed/list' endpoint that returns a list of recipes sorted by relevance. This endpoint is useful for developers who want to display a list of recipes on their website or application.

The Yummly API responses are in JSON format, which makes it easy to parse the data and integrate it into the application.

Overall, the Yummly API is a powerful tool for developers who want to integrate recipe search functionality into their applications. The API provides access to a vast database of recipes and allows developers to retrieve detailed information about recipes. The Yummly API is easy to use and provides developers with the flexibility to customize their recipe search functionality based on their specific requirements.

2.3.2 How we use Yummly API?

The 'feeds/list' endpoint returns a list of recipe feed items, where each item represents a recipe. Each feed item contains basic recipe information such as the recipe ID, recipe name, recipe image, and time to prepare.

The feed items are sorted by relevance, which means that the recipes displayed at the top of the list are the most relevant based on the search criteria used to generate the feed.

We define a class called RecipeApi that contains a static method called getRecipe() which returns a Future that resolves to a List of Recipe objects. The Recipe model is defined in a separate file in the project.

The getRecipe() method makes an HTTP GET request to the Yummly API using the http package and the uri provided as an argument. The headers of the request include the API key and host.

After receiving the response, the JSON data is decoded using the jsonDecode() function. The decoded data is then used to extract information about recipes, such as their names, total time, image URLs, ratings, and ingredient lines.

The ingredient lines are stored in a List called ingredientLines, and the remaining recipe information is stored in a List called temp. The two Lists are then used to create a List of Lists called wholeLine, which contains all the information needed to create Recipe objects.

Finally, the Recipe.recipesFromSnapshot() method is called to convert the List of Lists to a List of Recipe objects. This method is defined in the Recipe model file.

In summary, this code provides a convenient way to retrieve recipe information from the Yummly API and convert it to a format that can be used to create Recipe objects. The RecipeApi class contains a single static method called getRecipe() that retrieves the recipe information and returns it as a List of Recipe objects. The method makes an HTTP GET request to the Yummly API, parses the JSON response, and extracts the necessary information to create Recipe objects. The final List of Recipe objects is returned using the Recipe.recipesFromSnapshot() method which is in the Recipe model file.

Chapter 3: Related Work And Background

Initially, this chapter is divided into two parts: **Related work** and **Background**.

First, will address in the Related work our reading of papers similar to the same idea of our project, mentioning what they achieved in their research, the datasets they used, the techniques they relied on in their work, the percentage of accuracy they obtained, and the evaluation matrix that helped them in that.

Secondly, will show in the Background part what technologies are related to the system.

3.1 Related Work

3.1.1 YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object Detectors:

3.1.1.1 Introduction :

YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy 56.8% AP among all known real-time object detectors with 30 FPS or higher on GPU V100. YOLOv7-E6 object detector (56 FPS V100, 55.9%AP) outperforms both transformer-based detector SWINL Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy, and convolutionalbased detector ConvNeXt-XL Cascade-Mask R-CNN (8.6FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy, as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy. Moreover, we train YOLOv7 only on MS COCO dataset from scratch without using any other datasets or pre-trained weights

Real-time object detection is a very important topic in computer vision, as it is often a necessary component in computer vision systems. For example, multi-object tracking [94, 93], autonomous driving [40, 18], robotics [35, 58], medical image analysis [34, 46], etc. The computing devices that execute real-time object detection is usually some mobile CPU or GPU, as well as various neural processing units (NPU) developed by major manufacturers. For example, the Apple neural engine (Apple), the neural compute stick (Intel), Jetson AI edge devices (Nvidia), the edge TPU (Google), the neural processing engine (Qualcomm), the AI processing unit (MediaTek), and the AI SoCs (Kneron), are all NPUs. Some of the above mentioned edge devices focus on speeding up different operations such as vanilla convolution, depth-wise convolution, or MLP operations. In this paper, the real-time object detector we proposed mainly hopes that it can support both mobile GPU and GPU devices from the edge to the cloud. In recent years, the real-time object detector is still developed for different edge device. For example, the development of MCUNet [49, 48] and NanoDet [54] focused on producing low-power single-chip and improving the inference speed on edge CPU. As for methods such as YOLOX [21] and YOLOR [81], they focus on improving the inference speed of various GPUs. More recently, the development of real-time object detector has focused on the design of efficient architecture. As for real-time object detectors that can be used on CPU [54, 88, 84, 83], their design is mostly based on MobileNet [28, 66, 27], ShuffleNet [92, 55], or GhostNet [25]. Another mainstream real-time object detectors are developed for GPU [81, 21, 97], they mostly use ResNet [26], DarkNet [63], or DLA [87], and then use the CSPNet [80] strategy to optimize the architecture. The development direction of the proposed methods in this paper are different from that of the current mainstream real-time object detectors. In addition to architecture optimization, our proposed methods will focus on the optimization of the training process. Our focus will be on some optimized modules and optimization methods which may strengthen the training cost for improving the accuracy of object detection, but without increasing the inference cost. We call the proposed modules and optimization methods trainable bag-of-freebies.

3.1.1.2 Model re-parameterization :

Model re-parametrization techniques [71, 31, 75, 19, 33,11, 4, 24, 13, 12, 10, 29, 14, 78] merge multiple computational modules into one at inference stage. The model re-parameterization technique can be regarded as an ensemble technique, and we can divide it into two categories, i.e., module-level ensemble and model-level ensemble. There are two common practices for model-level reparameterization to obtain the final inference model. One is to train multiple identical models with different training data, and then average the weights of multiple trained models. The other is to perform a weighted average of the weights of models at different iteration number. Modulelevel re-parameterization is a more popular research issue recently. This type of method splits a module into multiple identical or different module branches during training and integrates multiple branched modules into a completely equivalent module during inference. However, not all proposed re-parameterized module can be perfectly applied to different architectures. With this in mind, we have developed new re-parameterization module and designed related application strategies for various architectures.

3.1.1.3 Model Scaling

Model scaling [72, 60, 74, 73, 15, 16, 2, 51] is a way to scale up or down an already designed model and make it fit in different computing devices. The model scaling method usually uses different scaling factors, such as resolution (size of input image), depth (number of layer), width (number of channel), and stage (number of feature pyramid), so as to achieve a good trade-off for the amount of network parameters, computation, inference speed, and accuracy. Network architecture search (NAS) is one of the commonly used model scaling methods. NAS can automatically search for suitable scaling factors from search space without defining too complicated rules. The disadvantage of NAS is that it requires very expensive computation to complete the search for model scaling factors. In [15], the researcher analyzes the relationship between scaling factors and the amount of parameters and operations, trying to directly estimate some rules, and thereby obtain the scaling factors required by model scaling. Checking the literature, we found that almost all model scaling methods analyze individual scaling factor independently, and even the methods in the compound scaling category also optimized scaling factor independently. The reason for this is because most popular NAS architectures deal with scaling factors that are not very correlated. We observed that all concatenationbased models, such as DenseNet [32] or VoVNet [39], will change the input width of some layers when the depth of such models is scaled. Since the proposed architecture is concatenation-based, we have to design a new compound scaling method for this model.

3.1.2 Real-time object detectors

Currently state-of-the-art real-time object detectors are mainly based on YOLO [61, 62, 63] and FCOS [76, 77], which are [3, 79, 81, 21, 54, 85, 23]. Being able to become a state-of-the-art real-time object detector usually requires the following characteristics: (1) a faster and stronger network architecture; (2) a more effective feature integration method [22, 97, 37, 74, 59, 30, 9, 45]; (3) a more accurate detection method [76, 77, 69]; (4) a more robust loss function [96, 64, 6, 56, 95, 57]; (5) a more efficient label assignment method [99, 20, 17, 82, 42]; and (6) a more efficient training method. In this paper, we do not intend to explore self-supervised learning or knowledge distillation methods that require additional data or large model. Instead, we will design new trainable bag-of-freebies method for the issues derived from the state-of-the-art methods associated with (4), (5), and (6) mentioned above.

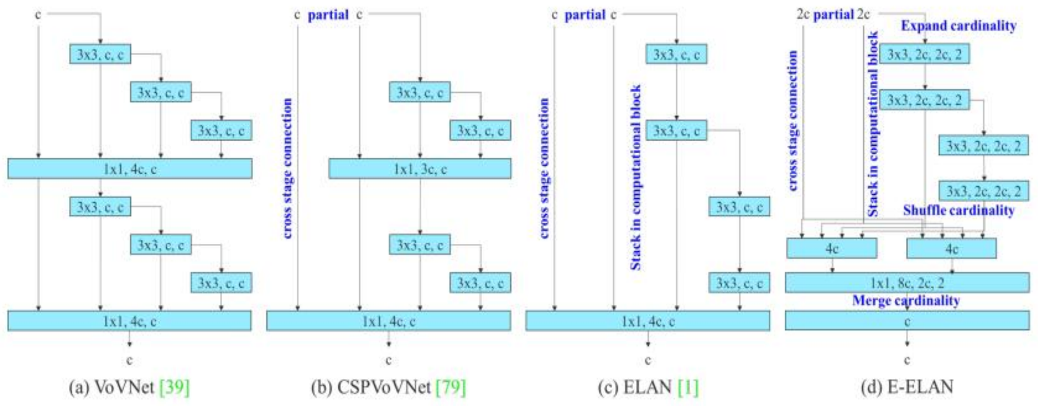


Figure 3.1 Extended efficient layer aggregation networks

In Figure 3.1 : Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the

features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different

feature maps and improve the use of parameters and calculations.

3.1.3 "Calorie estimation from food images using deep learning and its application to meal tracking systems" by Tsuyoshi Okawa et al. (2017):

Dataset The authors used a dataset of 2,000 food images collected from the internet for training and testing the model. The images were manually annotated with the corresponding food category and calorie content. The dataset was split into training, validation, and testing sets, with 1,500, 250, and 250 images respectively.

Techniques The authors used a deep learning technique called Multi-Task Deep Neural Network (MT-DNN) to estimate the calorie content of food items. The MT-DNN model includes two parallel networks: one for food recognition and one for calorie estimation. The model was trained using a combination of supervised and unsupervised learning techniques, and was optimized using the Adadelta algorithm.

Accuracy The authors achieved an accuracy of 80.9% in estimating the calorie content of food items. The evaluation metrics used were accuracy and mean absolute error (MAE). The model performed best in estimating the calorie content of fruits and vegetables, and performed worst in estimating the calorie content of meat and fish.

Evaluation Metrics The authors used two evaluation metrics to assess the performance of the model: accuracy and mean absolute error (MAE). Accuracy measures the proportion of correctly estimated calorie values, while MAE measures the average difference between the estimated and true calorie values. Lower MAE values indicate better performance.

3.2 Background

3.2.1 Food Detection

3.2.1.1 YOLOV8 :

YOLOv8 is the latest family of YOLO based Object Detection models from Ultralytics providing state-of-the-art performance.

Leveraging the previous YOLO versions, **the YOLOv8 model is faster and more accurate** while providing a unified framework for training models for performing

- Object Detection,
- Instance Segmentation, and
- Image Classification.

There are five models in each category of YOLOv8 models for detection, segmentation, and classification. YOLOv8 Nano is the fastest and smallest, while YOLOv8 Extra Large (YOLOv8x) is the most accurate yet the slowest among them.

YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x

Table 3.1 Five YOLO Models

YOLOv8 comes bundled with the following pre-trained models:

- Object Detection checkpoints trained on the COCO detection dataset with an image resolution of 640.
- Instance segmentation checkpoints trained on the COCO segmentation dataset with an image resolution of 640.
- Image classification models pretrained on the ImageNet dataset with an image resolution of 224.
  Let's take a look at the output using the YOLOv8x detection and instance segmentation of our Food Detection System.

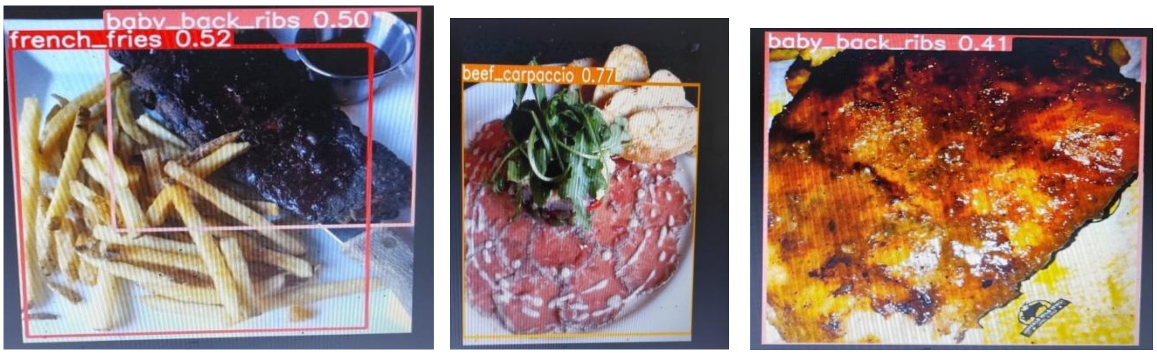


Figure 3.2 Outputs From Food Detection Using YOLOv8

YOLOv8 vs YOLOv7 vs YOLOv6 vs YOLOv5

Right away, YOLOv8 models seem to perform much better compared to the previous YOLO models. Not only YOLOv5 models, YOLOv8 is ahead of the curve against YOLOv7 and YOLOv6 models also.

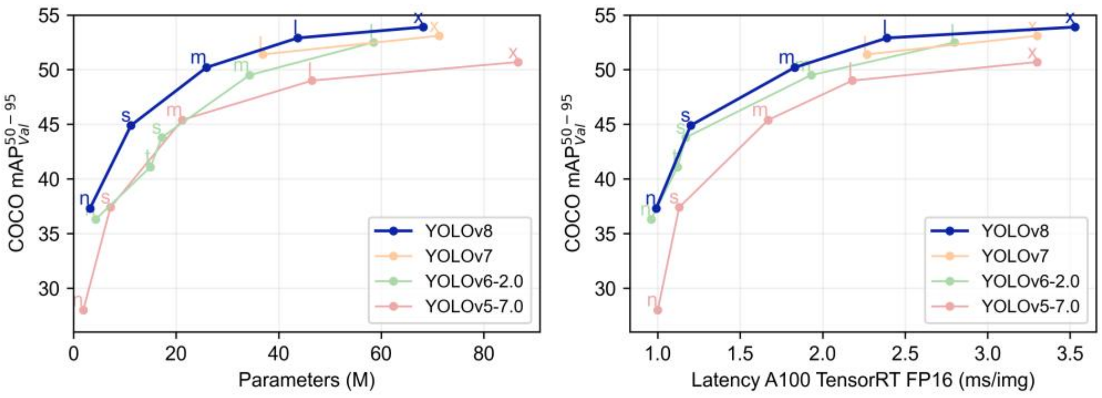
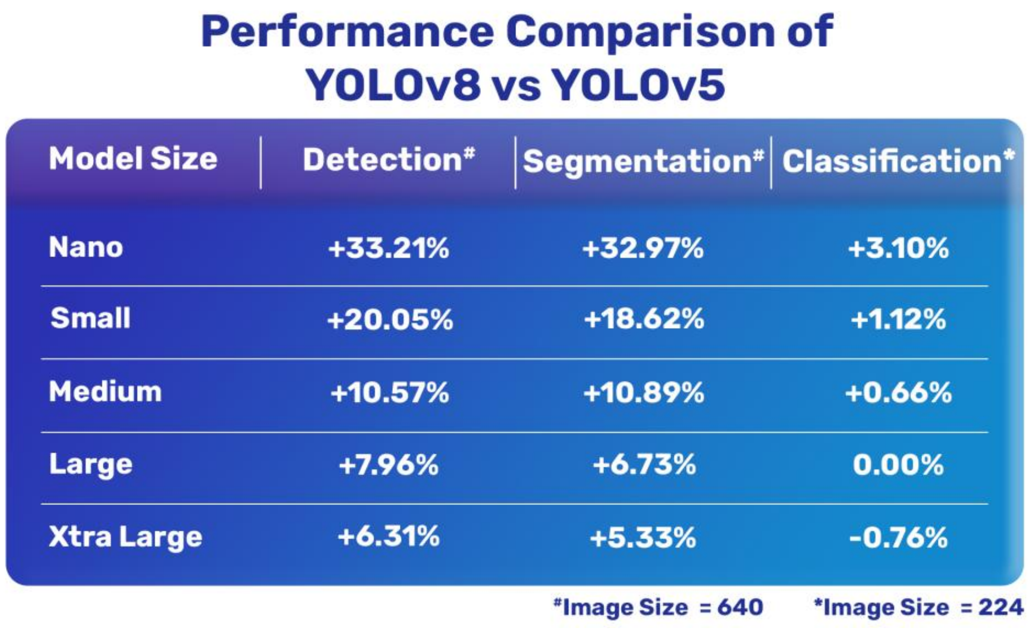


Figure 3.3 YOLOv8 compared with other YOLO models.

When compared with other YOLO models trained at 640 image resolution, all the YOLOv8 models have better throughput with a similar number of parameters.

Now, let’s take a detailed view of how the latest YOLOv8 models perform in comarison with the YOLOv5 models from Ultralytics. The following tables show a comprehensive comparison between YOLOv8 and YOLOv5.

Overall Comparison



Performance Comparison of YOLOv8 vs YOLOv5

Model Size	Detection#	Segmentation#	Classification*
Nano	+33.21%	+32.97%	+3.10%
Small	+20.05%	+18.62%	+1.12%
Medium	+10.57%	+10.89%	+0.66%
Large	+7.96%	+6.73%	0.00%
Xtra Large	+6.31%	+5.33%	-0.76%

#Image Size = 640 *Image Size = 224

Figure 3.4 YOLOv8 models compared with YOLOv5 models.

Chapter 4: System Design

In this chapter, will provide a detailed SDD. Will discuss the low-level design, the architectural

design, and the detailed design enhanced with UML model diagrams (**Block Diagram**,

Flowchart Diagrams, **Activity Diagrams**, **Use Case Diagram**, **Sequence Diagrams,** and

Collaboration Diagrams)

4.1 BLOCK DIAGRAM

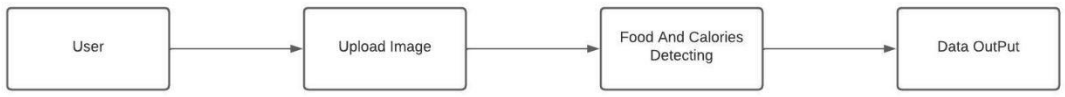
One of the most important diagrams used in software design is the block diagram and it is used in many other fields such as engineering in hardware design, electronic design, and process flow diagrams.

It is a system diagram in which the main parts or functions (major components) are represented by blocks connected by lines showing the relationships of the blocks.

It is often used to describe both high-level and low-level that aim to clarify general concepts without paying attention to implementation details.

And here will show the block diagram of our system, which shows the workflow of our Application .

4.1.1 Food And Calories Detection Blockdiagram



(Figure 4.1)

In Figure (4.1) The system starts with sensors that capture image or video data of the food.

The image or video data is then processed using image processing algorithms to identify the food.

The food and calorie detection component of the system uses this information to identify the type of food and estimate its calorie content based on the known nutritional information of the food.

The output of the system is the data obtained from the food and calorie detection component, which may include information such as the type of food, portion size, and calorie content.

4.1.2 Pedometer Blockdiagram

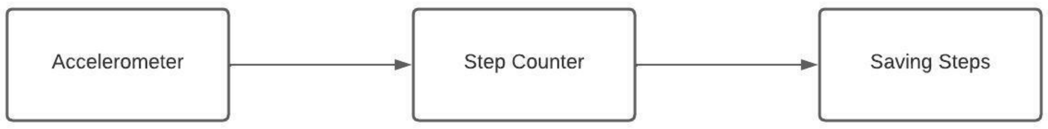


Figure (4.2)

In Figure (4.2) The system starts with an accelerometer, which is a sensor that measures the acceleration of the user's movements.

The accelerometer data is used by the step counter component of the system to detect when the user takes a step.

4.1.3 Show Recipes Blockdiagram

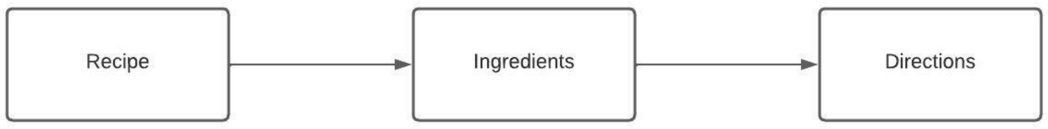


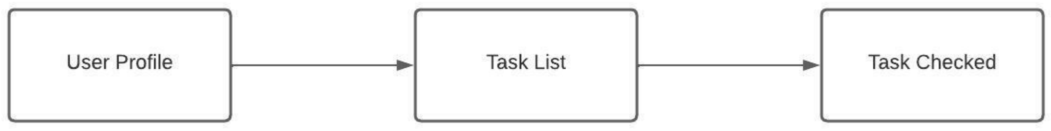
Figure (4.3)

In Figure (4.3) The system starts with a recipe source, which may be a website, cookbook, or other source of recipes.

The recipe parser component of the system reads the recipe data and extracts relevant information such as ingredients and cooking instructions.

The recipe viewer component of the system displays the recipe information to the user in a user-friendly format, such as a web page or mobile app.

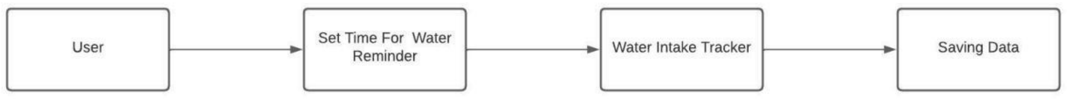
4.1.4 To-Do Organizer Blockdiagram



Figure(4.4)

In Figure(4.4) The system starts with a user profile, which includes information about the user such as their name, email, and preferences.
The task list component of the system provides a list of tasks for the user to complete, which may include tasks created by the user or tasks assigned to them by others.
The task manager component of the system allows the user to manage their tasks, including adding new tasks, setting deadlines, and marking tasks as complete.
The output of the system is the data obtained from the task manager component, which may include information such as the user's completed tasks, pending tasks, and overdue tasks.

4.1.5 Water Reminder Blockdiagram



Figure(4.5)

In Figure (4.5)The system starts with sensors that detect the user's water intake, which may include a smart water bottle or a wearable device that tracks water consumption.

The reminder timer component of the system uses the user's water intake data and the user's water intake goals to set reminders for the user to drink water at regular intervals throughout the day.

The water intake tracker component of the system uses the user's water intake data to track their progress towards their water intake goals.

The output of the system is the data obtained from the water intake tracker component, which may include information such as the user's daily, weekly, and monthly water intake.

4.1.6 Workout Blockdiagram

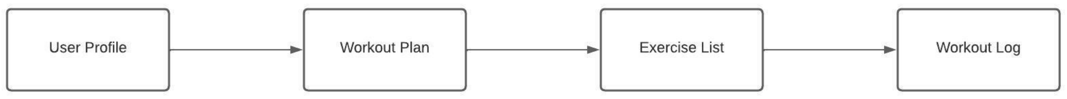


Figure (4.6)

In Figure (4.6)The system starts with a user profile, which includes information about the user such as their age, gender, and fitness level.

The workout plan component of the system uses this information to generate a customized workout plan for the user, which may include specific exercises, sets, and reps.

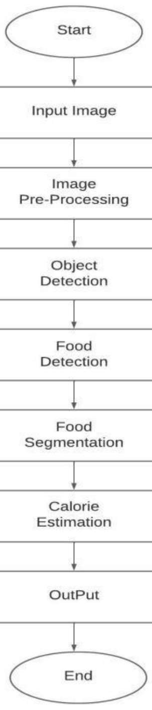
The exercise list component of the system provides a library of exercises that the user can choose from to customize their workout plan.

The workout log component of the system allows the user to track their progress over time, including the number of sets, reps, and weights used during each exercise.

4.2 FLOWCHART

This diagram has been called many names, including process flowchart and process flow diagram.
A flowchart is a visual representation of the separate steps of a process in sequential order.
It is a general tool that can be adapted for a variety of purposes and can be used to describe different processes.I
t is a popular process analysis tool and one of the seven essential quality tools.
The elements that can be included in a flowchart are a series of actions, materials, or services entering or leaving the process (inputs and outputs), the decisions to be made, the people involved, the time each step takes, and/or the process's measurements

4.2.1 Food And Calories Detection Flowchart



(Figure 4.7)

For (Figure 4.7)This chart shows the same steps as mentioned in (Figure 4.1), the only difference lies in the symbols for building the Flowchart

4.3 ACTIVITY DIAGRAMS

The activity diagram is another important diagram in UML for describing the dynamic aspects of a system. It is basically a flowchart to represent the flow from one activity to another. Activity can be described as running the system.

The flow of control is dragged from one process to another. This flow can be sequential, branched, or synchronous. Activity diagrams deal with all types of flow control using different elements like fork, hook, etc.
And in Our Application, will explain Several Diagrams Which are :

4.3.1 : Food And Calories Detection Activity Diagram :

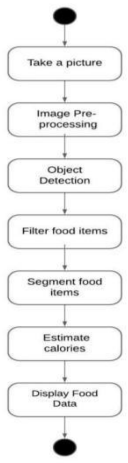


Figure 4.8 : Food And Calories Flowchart

In Figure 4.8 The User Upload The Photo/Video so that it is processed and show the Detection and Food Data.

4.3.2 : Pedometer Tracker Activity Diagram :

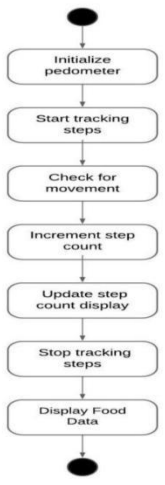
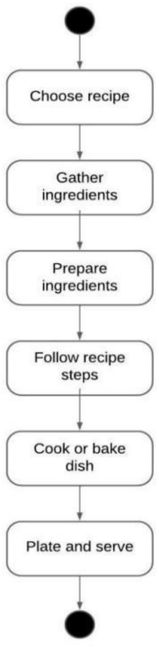


Figure 4.9 Shows Footsteps

In Figure 4.9 The user is Able To know steps he walked a day .

4.3.3 Show Recipe Activity Diagram:



(Figure 4.10 : Show Recipe)

In Figure 4.10 it Shows the Steps Of the user Choosing Which Recipe he wanna make.

4.3.4 : TO-DO Organizer Activity Diagram:

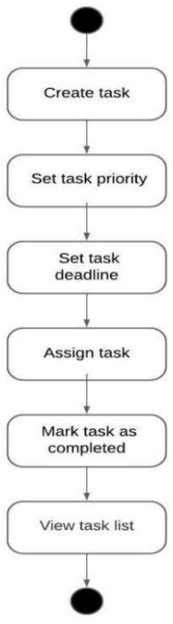
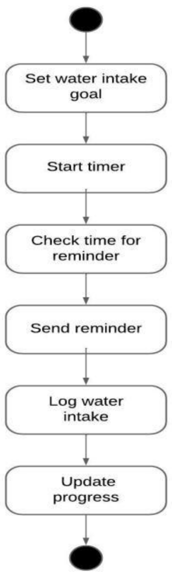


Figure 4.11 To Do Organizer Activity Diagram

In Figure 4.11 The User Is able to prioritize his tasks and mark It done .

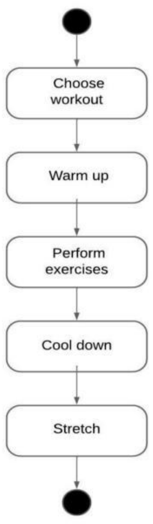
4.3.5 Water Reminder Activity Diagram



(Figure 4.12 : Water Reminder)

In Figure 4.13 The User Sets Reminder For Him to Make The System remind him for Drinking water

4.3.6 Workout Activity Diagram:



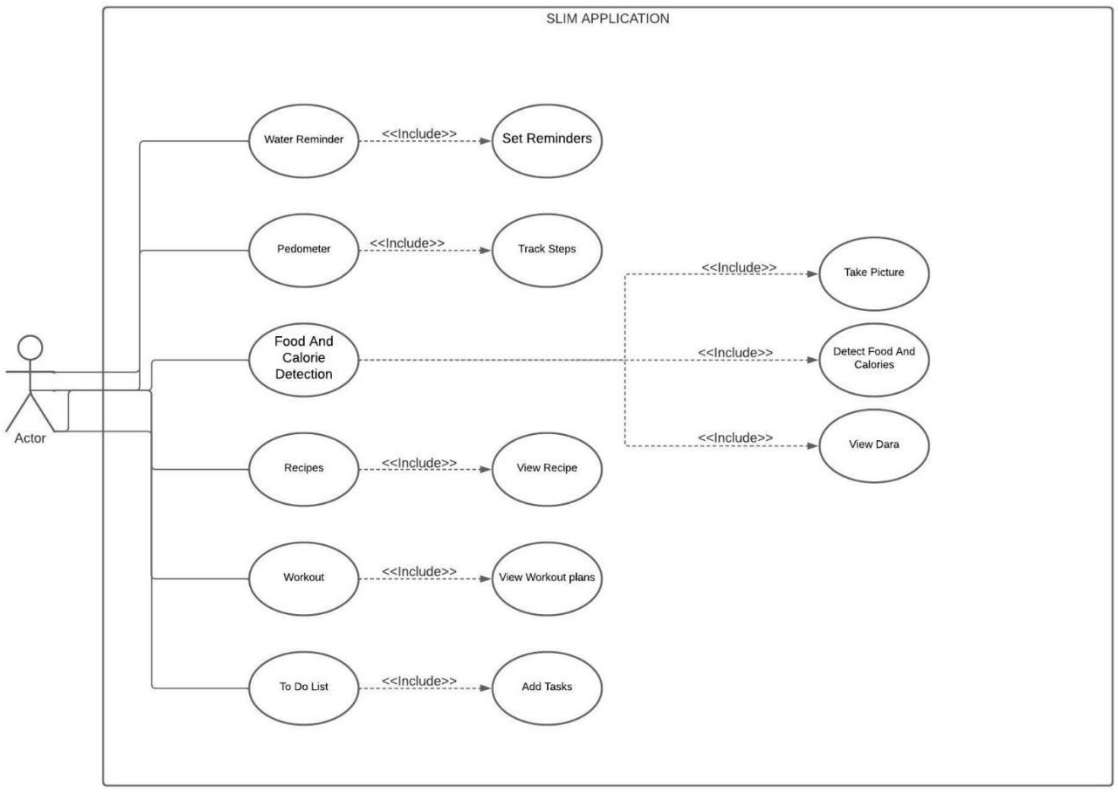
(Figure 4.13 Workout)

In Figure 4.13 The User Choose which workout he wanna do and do it .

4.4 USE CASE

The use case diagram is responsible for listing all possible scenarios that can occur in the system, but it only describes the high level without mentioning the details of the system's operation. This diagram consists of users - or sometimes called them actors - and shows how they interact with the system or with each other, (see Figure ..) for SLIM use case diagram, and (see Table …, and Table ..) for use case scenarios.

4.4.1 Use Case Diagram



(Figure 4.14)

In The (Figure 4.14)Here it becomes clear of several scenarios, one of which the user's choice to choose whatever feature he wanna use.

4.4.2 Use Case Scenarios

4.4.2.1 Food And Calories Detection Use Case Scenario

Table 4.1 Food And Calories Detection Use Case Scenario

Actor	User
Description	The system can provide information on the calories, macronutrients, and other nutritional information of the food, and help the user make informed choices about their diet.
Precondition	User has downloaded and installed SLIM App on his smartphone
Postcondition	The User has tracked his daily calorie intake and nutrition, and has received feedback and insights on her diet.
Main successes scenario (The flow or Steps)	1-The system prompts The User to take a photo of his meal using her smartphone camera. 2-The User takes a photo of His meal and the system uses image recognition technology to identify the food and provide information on its nutritional content. 3-The system displays a summary of the calories, macronutrients, and other nutritional information of the food.
Extensions	none

4.4.2.2 Footsteps Tracker Use Case Scenario

Table 4.2 Footsteps Tracker Use Case Scenario

Actor	User
Description	It tracks the user's steps using built-in sensors. The system can also track other physical activities, such as running or cycling, and provide feedback on the user's progress towards their fitness goals.
Precondition	User has downloaded and installed SLIM App on his smartphone
Postcondition	User has tracked his daily steps and physical activity, and has received feedback and insights on his progress..
Main successes scenario (The flow or Steps)	1-Throughout the day, the system tracks User's steps and physical activity using built-in sensors. 2-At the end of the day, the system displays a summary of User's daily steps and physical activity, including the number of steps taken, distance traveled.
Extensions	None

4.4.2.3 Show Recipes Use Case Scenario

Table 4.3 : Show Recipes Use Case Scenario

Actor	User
Description	Show recipes provides users with a database of recipes based on their dietary preferences, ingredients,
Precondition	User has downloaded and installed SLIM App on his smartphone
Postcondition	The User Has found a recipe that he wants to try, and has received information and instructions on how to make the dish.
Main successes scenario (The flow or Steps)	1-User can browse through the list of recipes, filter by cuisine type or ingredients, and save his favorite recipes for future reference. 2-When he finds a recipe that he wants to try, he can click on the recipe to view more information, such as ingredients, cooking instructions, and nutritional information. 3-The User can follow the recipe instructions to make the dish, and receive helpful tips and information along the way.
Extensions	none.

4.4.2.4 TO-DO Organizer Use Case Scenario

Table 4.4 TO-DO Organizer Use Case Scenario

Actor	User
Description	TO-DO helps users create and manage their daily tasks and priorities.
Precondition	User has downloaded and installed SLIM App on his smartphone
Postcondition	User has completed his tasks for the day and feels organized and productive.
Main successes scenario (The flow or Steps)	1-The system displays an empty list of tasks for the day. 2-The User enters his tasks for the day, such as work assignments, errands, and personal chores. 3-The User can prioritize his tasks and assign due dates or deadlines.
Extensions	None.

4.4.2.5 Water Reminder Use Case Scenario

Table 4.5 Water Reminder Use Case Scenario

Actor	User
Description	helps users track their water intake and reminds them to drink water at regular intervals throughout the day
Precondition	User has downloaded and installed SLIM App on his smartphone
Postcondition	User has received reminders to drink water throughout the day and has consumed an appropriate amount of water to stay hydrated.
Main successes scenario (The flow or Steps)	1-The system displays a message indicating how much water Sarah should drink each day. 2-Throughout the day, the system sends reminders to User to drink water at regular intervals, such as every hour or every 30 minutes. 3-When a reminder pops up, Sarah can either dismiss it or mark that she has had a drink of water. 4-At the end of the day, the system displays a summary of Sarah's water intake for the day.
Extensions	The User can customize the reminder interval to meet her specific needs and preferences.

4.4.2.6 Workout Use Case Scenario

Table 4.6 Workout Use Case Scenario

Actor	User
Description	It helps users do their workouts
Precondition	User has downloaded and installed SLIM App on his smartphone
Postcondition	The user has completed his workout and received feedback and insights on his progress towards his fitness goals..
Main successes scenario (The flow or Steps)	1-The User selects a workout and the system provides instructions and guidance on how to perform the exercises correctly.
Extensions	None.

4.5 SEQUENCE DIAGRAM

A sequence diagram is a type of interactive diagram that details the system and how processes are performed. It captures the interaction of objects in the context of cooperation.

Sequence diagrams are time-focused and visually display the order of
interaction using the vertical axis of the chart to represent when and when messages are sent.

And in Our System, will explain Our Main Feature Food And Calories Detection :

4.5.1 Food And Calories Detection Sequence Diagram

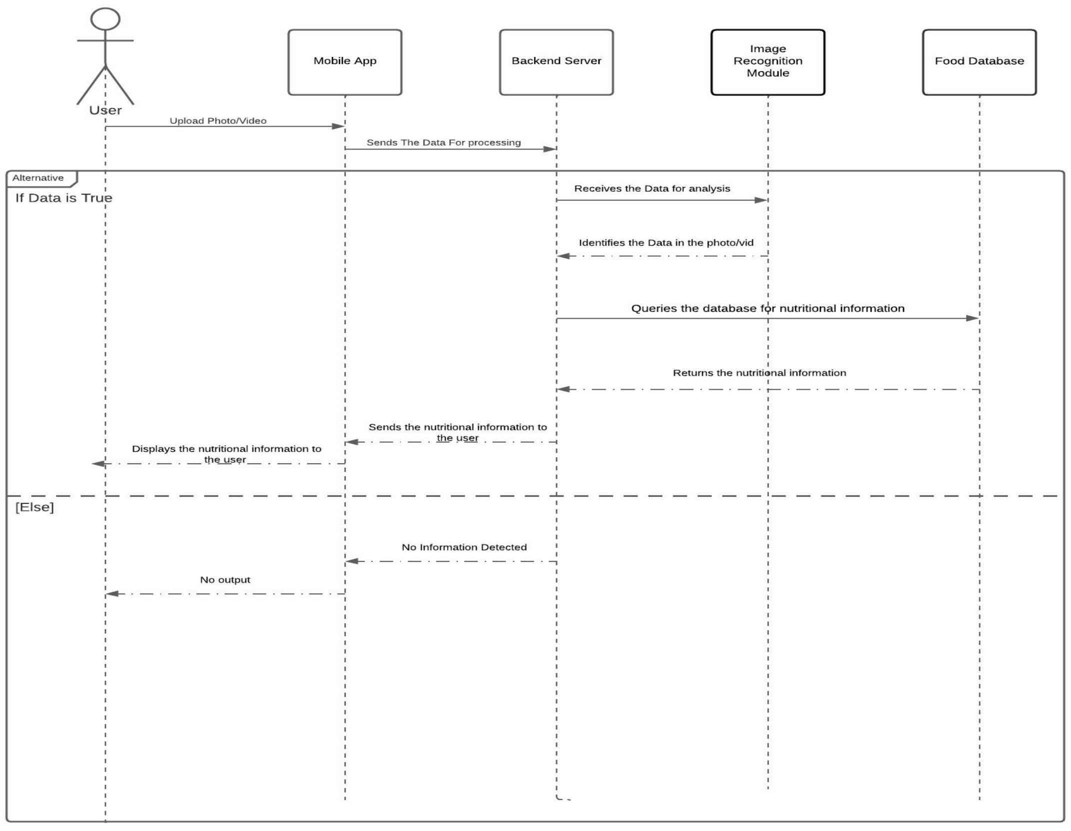

User
Mobile App
Backend Server
Image Recognition Module
Food Database
Upload Photo/Video
Sends The Data For processing
Alternative
If Data is True
Receives the Data for analysis
Identifies the Data in the photo/vid
Queries the database for nutritional information
Returns the nutritional information
Sends the nutritional information to the user
Displays the nutritional information to the user
[Else]
No Information Detected
No output

Figure 4.15 Food And Calories Detection Sequence Diagram

In Figure 4.15 It llustrates the flow of information and interactions between different components of a food and calories detection system. The user takes a photo of the food using a mobile app, which sends the photo to a backend server for processing. The image recognition module identifies the food in the photo, and the backend server queries a food database for the nutritional information. The nutritional information is then sent back to the mobile app, which displays it to the user.

4.6 COLLABORATION DIAGRAMS

Another type of interactive diagram is the collaboration diagram, also known as the communication diagram, and it is similar to the sequence diagram in showing the interaction of the elements of the system with each other and the way they communicate.

These diagrams can be used to illustrate the dynamic behavior of a particular use case and to define the role of each object.

It differs from a sequence diagram in that it does not care about the timing of each action that occurs, but rather numbers each event before its
name, as will see in (Figure 4.15).

And in Our System, will explain Our Main Feature Food And Calories Detection :

4.6.1 Food And Calories Detection Collaboration Diagram

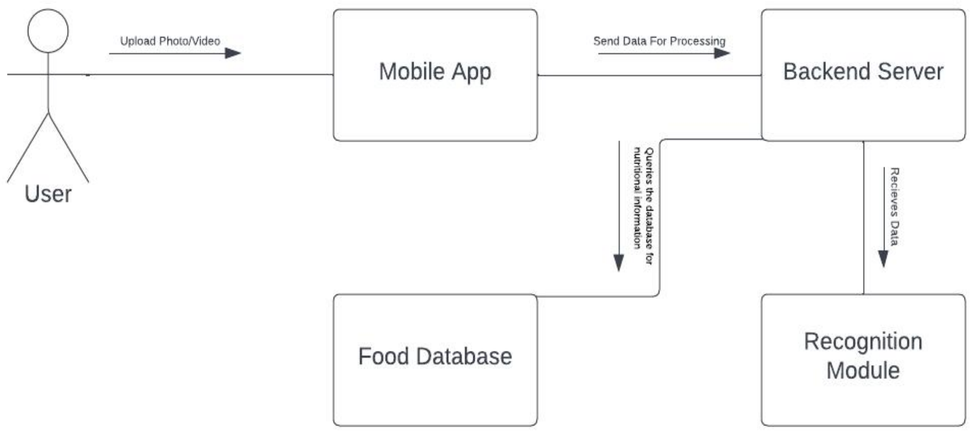


Figure 4.16 Food And Calories Detection Collaboration Diagram

In Figure 4.16 It illustrates the flow of interactions and information between the user, mobile app, and backend server in a food and calories detection system.

Chapter 5: : System Implementation and results

This chapter is the conclusion of our project and will discuss the following (Description of The High-Level Design of The Already Implemented System, Used Tools and Hardware, Used Dataset, System Results and Their Performance, Future Work, System Snapshots, and Samples of Code)

5.1 DESCRIPTION OF THE HIGH-LEVEL DESIGN OF THE ALREADY IMPLEMENTED SYSTEM

Our System Contains The Main Page which Contain The Tracker And TO-DO Organizer And Also The NavBar Can Transfer You into our Features From Pedometer Where you can Track Your Footsteps ,Show Recipes where you can know delicious Dishes , Workout That You can do and Water Reminder Where You can Set Reminders For you To drink Water As Showed in The Followed Figures in The Snapshots of our Application .

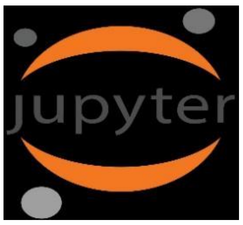
On The left Of the Pages You can swipe to see The Slide Bar of Our App so that you Can Edit your profile That Contain Your Name , Age , Weight , Height .
And you Can Find our Main Feature Of our System Food And calories Detection System Where you Can Upload Picture/Video Or live Video and it will show you the output of Your Data in details .

5.2 USED TOOLS SOFTWARE AND HARDWARE

- **Python 3.6 or Higher:** Programming language

- **Anaconda :** open-source distribution of the Python programming language

- **Jupyter :** Python Environment



•**Android Studio:** IDE for developing Android applications.

•**Dart:** Programming language .

•**Flutter:** mobile application development framework .

•**Flask:** used for building web applications and APIs.

•**Firebase:** mobile and web application development platform.

Hardware

Computer: A computer with a minimum of 4 GB of RAM and a 1 GHz processor
Graphics card: A graphics card with CUDA support
Peripherals: A monitor, keyboard, and mouse

• Python Libraries and Technologies Used :

1- Python
2- Dart
3-Flutter
4-Flask
5-Yummly API
6-SQL
7-SQLLite
8-Firebase
9-Tenserflow
10-YOLOv8
11-OS
12-Numpy
13-Pandas
14-matplotlib

5.3 USED DATASET

The food-101 dataset is a large-scale dataset of images of food. It consists of 101 classes of food, with each class containing 750 training images and 250 test images. The images are of high quality and are well-aligned. The dataset is also well-balanced, with each class having approximately the same number of images.

The food-101 dataset is a valuable resource for researchers and developers who are working on food recognition and food classification tasks. It can be used to train and evaluate deep learning models for food recognition.

Here are some of the benefits of using the food-101 dataset:

- Large-scale dataset: The food-101 dataset is a large-scale dataset, which means that it contains a large number of images. This allows researchers and developers to train deep learning models with a large amount of data, which can lead to better performance.
- High-quality images: The images in the food-101 dataset are of high quality. This means that they are clear and well-aligned, which makes them easier to train deep learning models on.
- Well-balanced dataset: The food-101 dataset is well-balanced, which means that each class contains approximately the same number of images. This is important for training deep learning models, as it ensures that the models are not biased towards any particular class.

Figure 5.1: Food Data

Food detection: The food-101 dataset can be used for food detection by performing some preprocessing on the images, such as annotations of bounding boxes. Bounding boxes are rectangular regions that are used to identify and localize objects in images. By annotating the bounding boxes of food items in the food-101 dataset, we can create a dataset that can be used to train deep learning models for food detection.

Here are some of the steps involved in food detection using the food-101 dataset:

1. Data preprocessing: The first step is to preprocess the images in the food-101 dataset. This includes resizing the images to a consistent size, and normalizing the pixel values.
2. Bounding box annotation: The next step is to annotate the bounding boxes of food items in the images. This can be done by labelImg tool.
3. Training the model: The final step is to train a deep learning model for food detection. The model can be trained using a variety of deep learning frameworks. We use YOLOv8 framework.

5.4 SYSTEM RESULTS AND THEIR PERFORMANCE

the results and performance of the food recognition model using MobileNetv2:

- Accuracy: The model achieved an accuracy of 77.73% on the validation set. This means that the model was able to correctly identify food items in 77.73% of the cases.
- Loss: The model's loss on the validation set was 1.09925. This means that the model's predictions were, on average, 1.09925 units away from the ground truth labels.
- Number of epochs: The model was trained for 15 epochs. This is a standard number of epochs for training deep learning models.
- Model: The model is based on the MobileNetv2 architecture. MobileNetv2 is a lightweight convolutional neural network that is well-suited for mobile devices.

Output Samples:

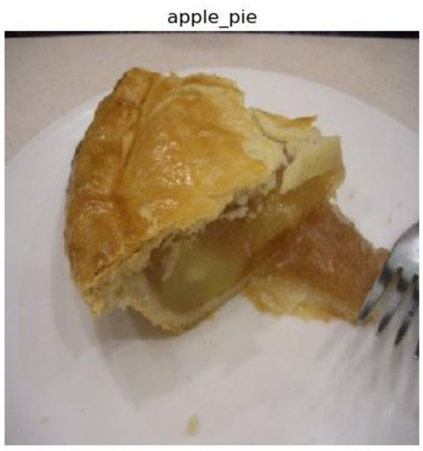


Figure 5.2： apple_pie

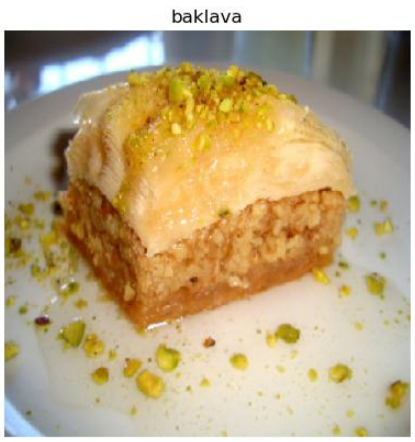


Figure 5.3 :baklava

Overall, the results of the food recognition model using MobileNetv2 are promising. The model was able to achieve a high accuracy on the validation set, and it was trained in a reasonable amount of time. The model can be used to identify food items in a variety of applications, such as food delivery, food tracking, and food recommendation.

the results and performance of the food detection model using YOLOv8

The YOLOv8 model was evaluated on the food-101 dataset. The following metrics were used to evaluate the model:

- Accuracy: The accuracy of the model was 80.1%. This means that the model correctly identified 82.5% of the objects in the test set.

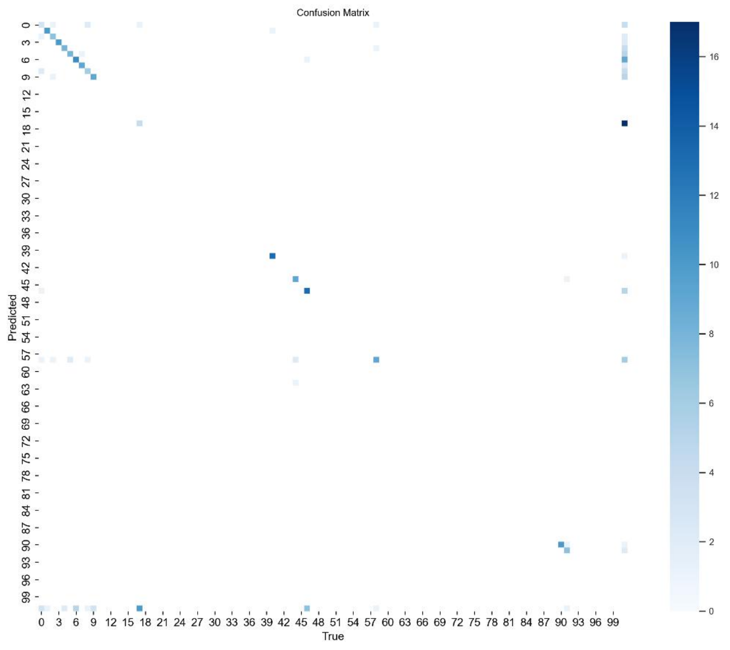


Figure 5.4 : Confusion matrix

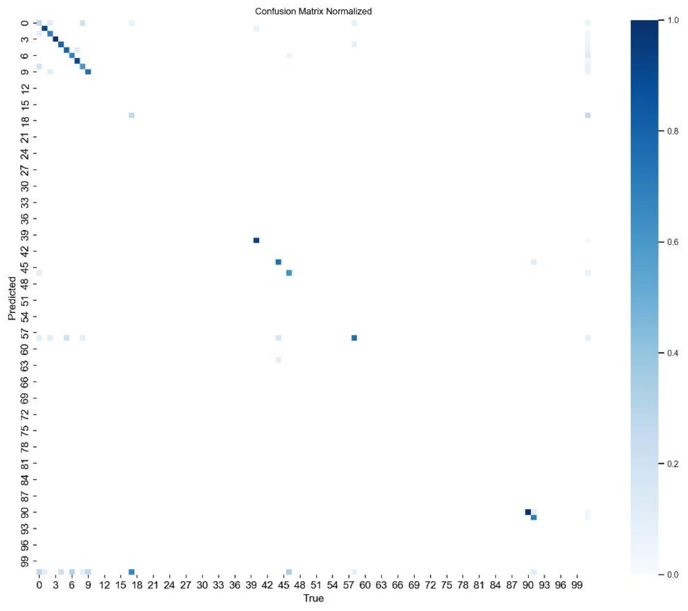


Figure 5.5 : confusion matrix normalization

- Precision: The precision of the model was 75.1%. This means that the model correctly identified 85.0% of the objects that it predicted were present in the test set.

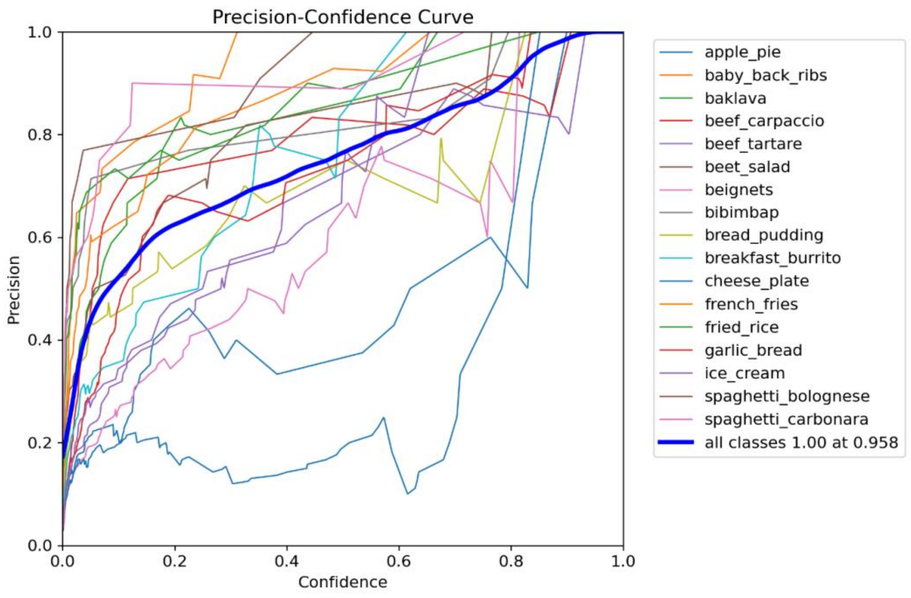


Figure 5.6:Precision-Cofidence Curve

- Recall: The recall of the model was 71.4%. This means that the model detected 79.0% of the objects that were present in the test set.

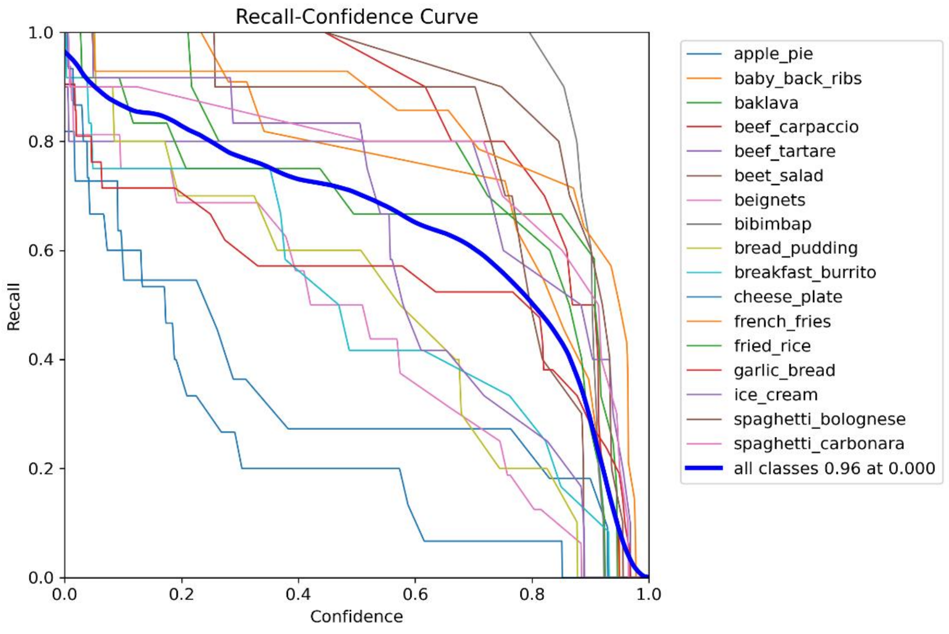


Figure 5.7 :Recall-Confidence Curve

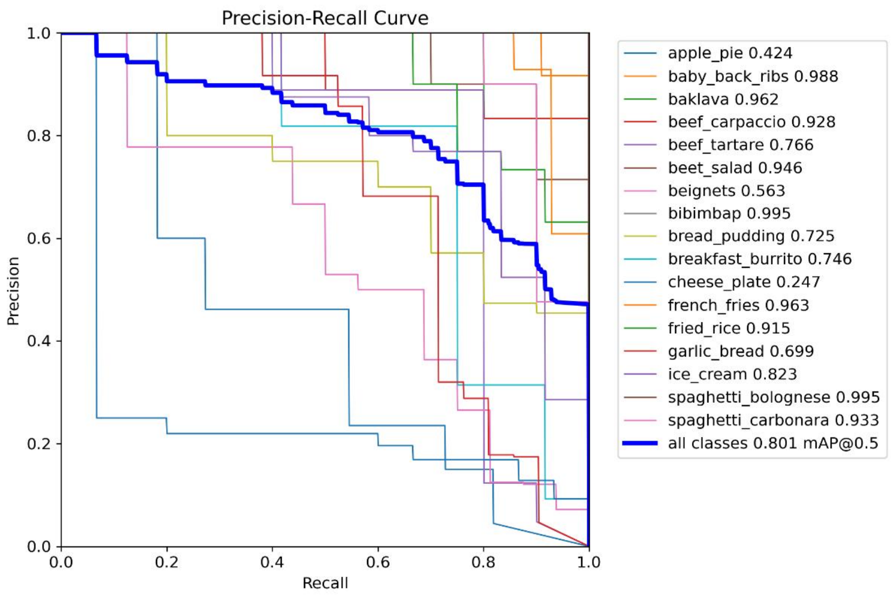


Figure 5.8:Precision-Recall Curve

Output Samples:

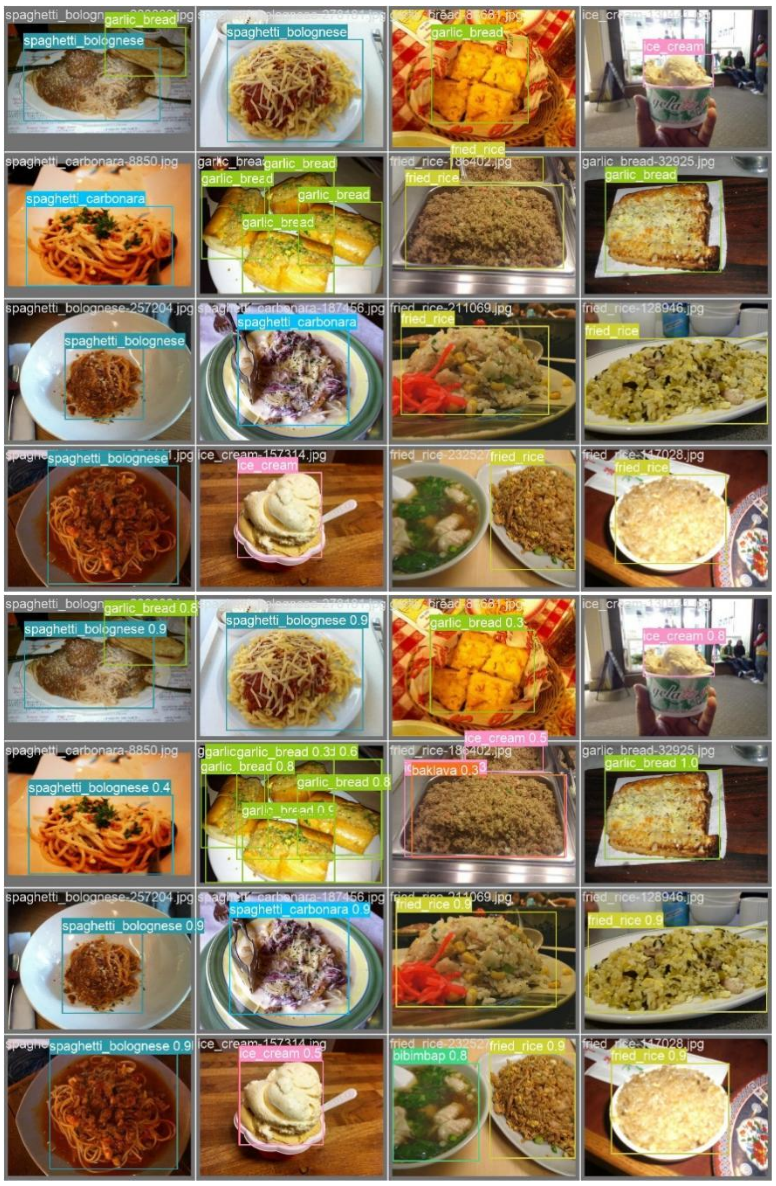


Figure 5.9 : output samples

The results of the evaluation show that the YOLOv8 model is a good model for food detection. The model has a high accuracy, precision, and recall. The model can be used to detect food items in new images.

5.5 FUTURE WORK

5.5.1 Higher priority services

- <u>Adding Community Platform</u> To make Users Interact with each other so that they can Add Post/Friend/Comment/Recipe so that they feel Motivated And they're not the only one who is making diet .

-<u>Friend Recommendation System</u> Based On Weight and Height.

-<u>Online Consultants</u> for Nutrition

5.5.2 Extra Features

- <u>Add a Notification</u> For Workout To Remind The User Of his Workout

-<u>Active Dark Mode</u>

5.6 SYSTEM SNAPSHOTS

5.6.1 Food And Calories Detection Feature Snapshot

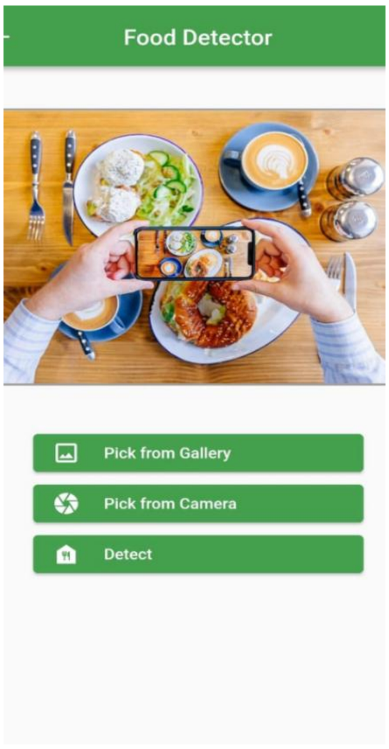
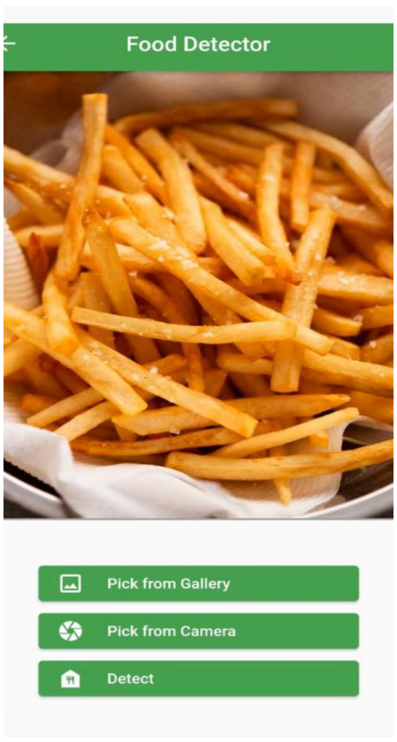
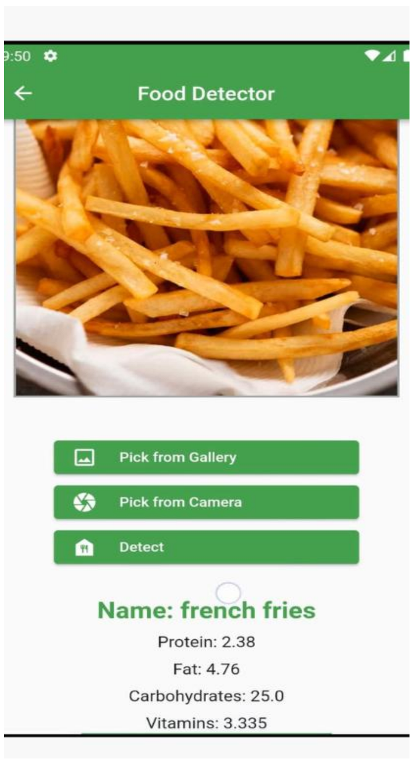


Figure 5.10 Food And Calories Detection Feature

5.6.1.1 Some Outputs Of our main System

 Food Detector Pick from Gallery Pick from Camera Detect Figure 5.11 output 1	 9:50 Food Detector Pick from Gallery Pick from Camera Detect Name: french fries Protein: 2.38 Fat: 4.76 Carbohydrates: 25.0 Vitamins: 3.335

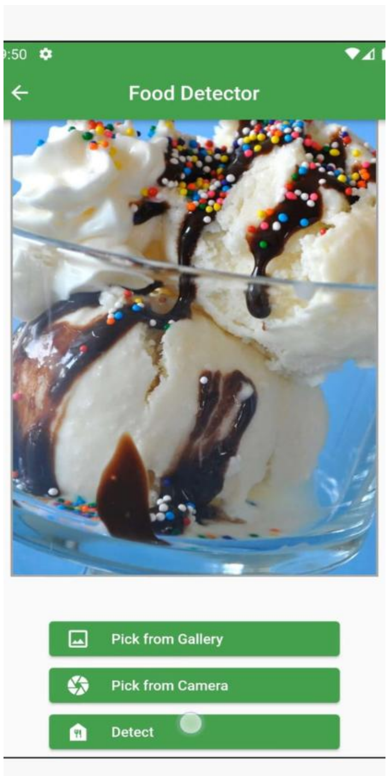
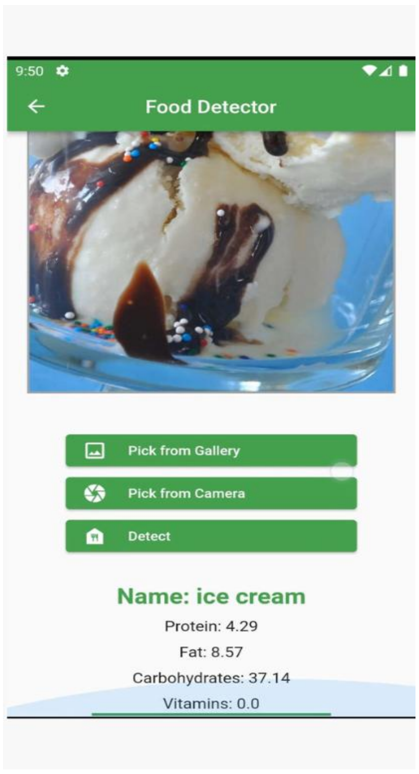


Figure 5.11 output 2



Name: ice cream
Protein: 4.29
Fat: 8.57
Carbohydrates: 37.14
Vitamins: 0.0

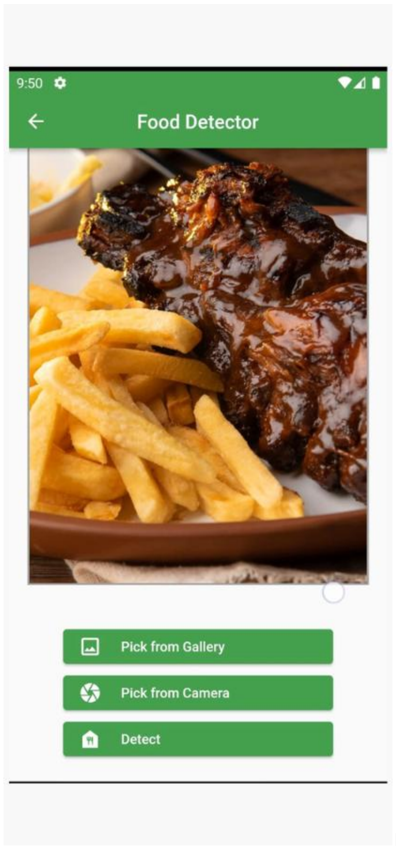
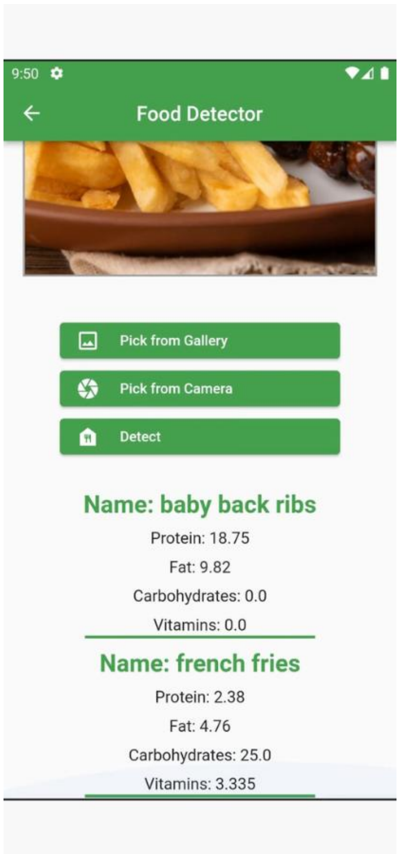


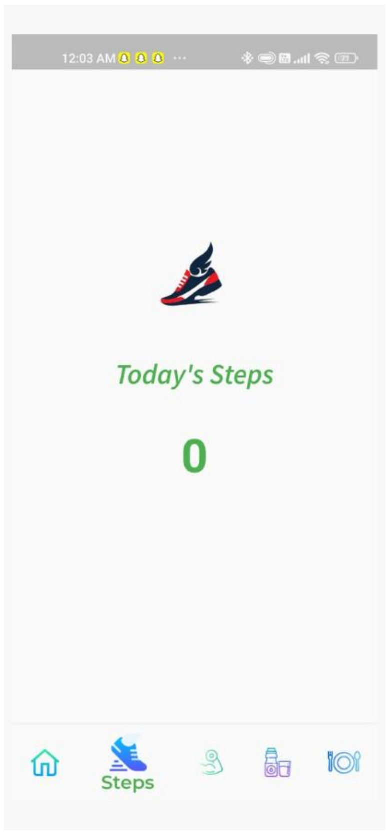
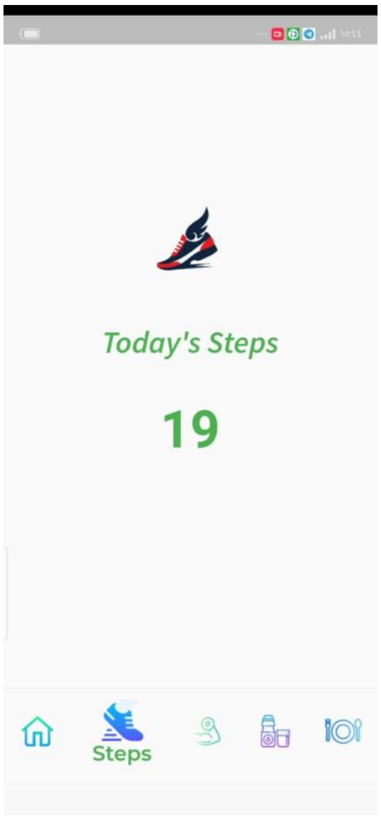
Figure 5.11 output 3



Name: baby back ribs
Protein: 18.75
Fat: 9.82
Carbohydrates: 0.0
Vitamins: 0.0

Name: french fries
Protein: 2.38
Fat: 4.76
Carbohydrates: 25.0
Vitamins: 3.335

5.6.2 Pedometer Snapshot

 Figure 5.12 Pedometer Snapshot	

5.6.3 Show Recipe Snapshot

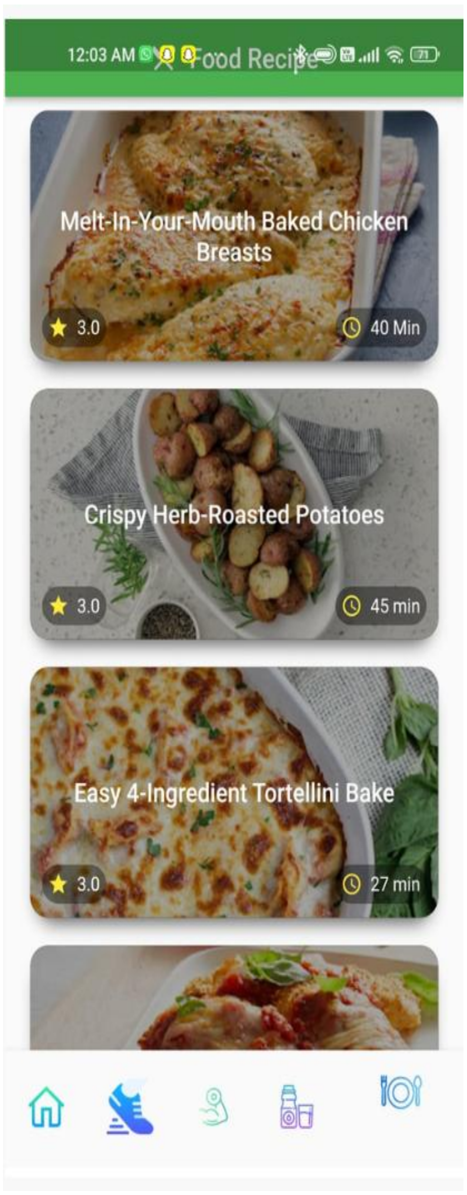
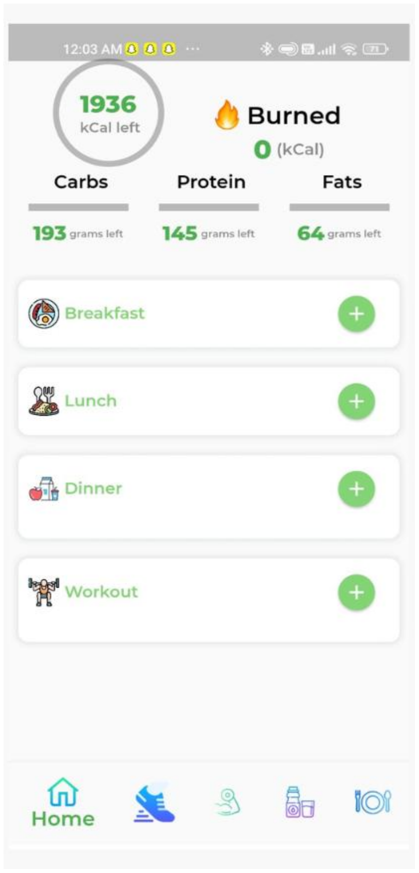

12:03 AM
Food Recipe
Melt-In-Your-Mouth Baked Chicken Breasts
3.0
40 Min
Crispy Herb-Roasted Potatoes
3.0
45 min
Easy 4-Ingredient Tortellini Bake
3.0
27 min

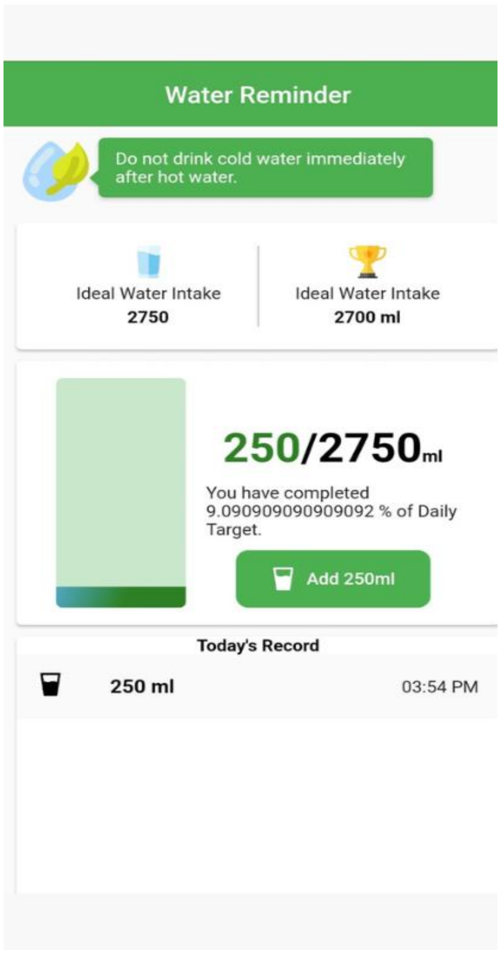
Figure 5.13 : Show Recipe Snapshot

5.6.4 TO-DO Organizer Snapshot



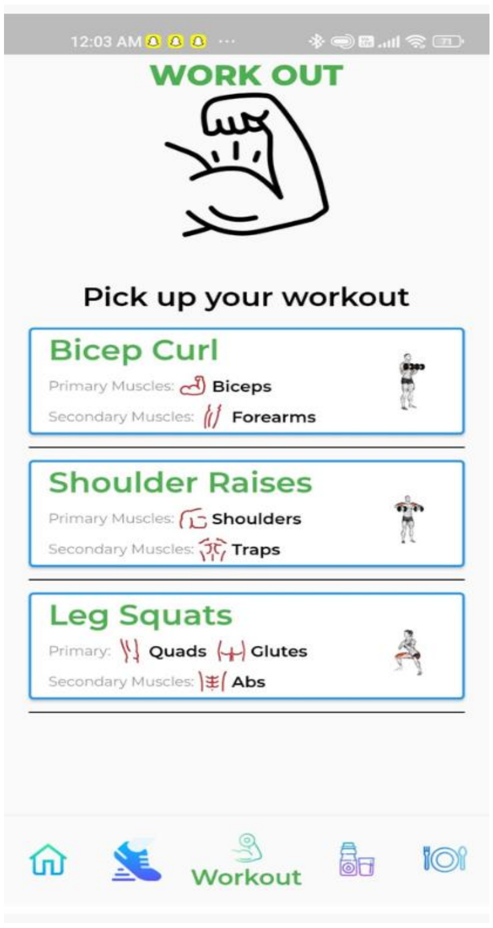
5.14 TO-DO Organizer Snapshot

5.6.5 Water Reminder Snapshot



5.15 Water Reminder Snapshot

5.6.6 Workout Snapshot


12:03 AM

WORK OUT

Pick up your workout

Bicep Curl
Primary Muscles: Biceps
Secondary Muscles: Forearms

Shoulder Raises
Primary Muscles: Shoulders
Secondary Muscles: Traps

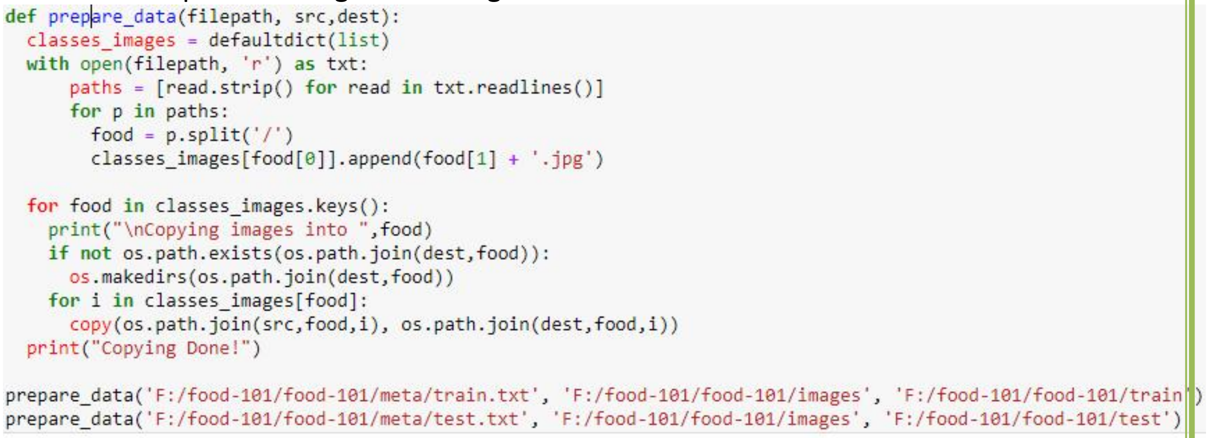
Leg Squats
Primary: Quads Glutes
Secondary Muscles: Abs

5.16 Workout Snapsho

5.7 SAMPLE Of CODE

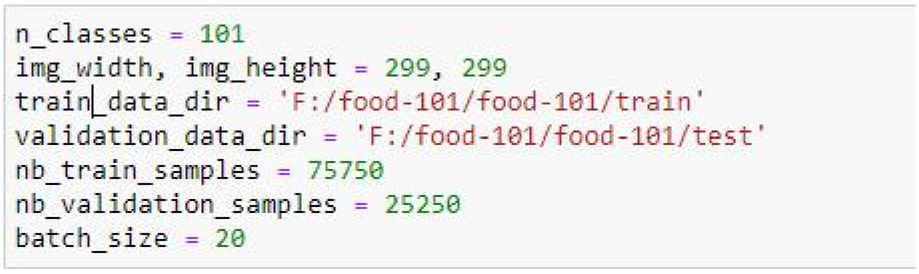
5.7.1 Classification Model

5.7.1.1 Prepare training and testing sets



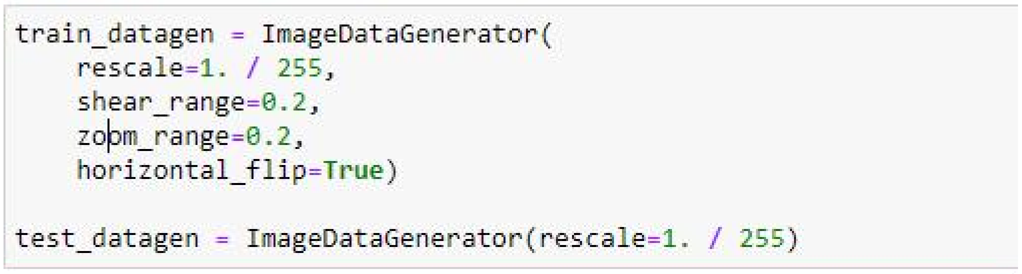
```
def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')

  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!")

prepare_data('F:/food-101/food-101/meta/train.txt', 'F:/food-101/food-101/images', 'F:/food-101/food-101/train')
prepare_data('F:/food-101/food-101/meta/test.txt', 'F:/food-101/food-101/images', 'F:/food-101/food-101/test')
```

5.7.1.2 Variables used to configure the training and validation of the model



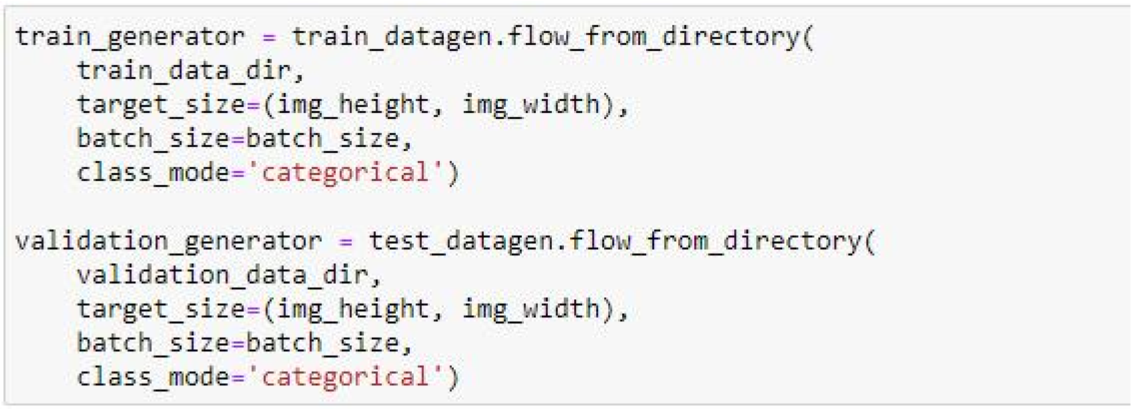
```
n_classes = 101
img_width, img_height = 299, 299
train_data_dir = 'F:/food-101/food-101/train'
validation_data_dir = 'F:/food-101/food-101/test'
nb_train_samples = 75750
nb_validation_samples = 25250
batch_size = 20
```

5.7.1.3 The train_datagen and test_datagen objects



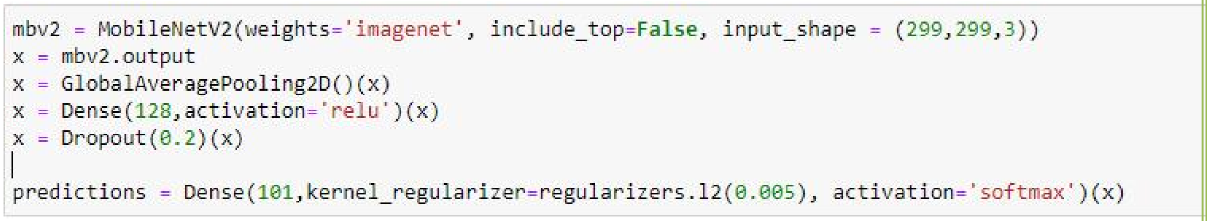
```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

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

5.7.1.4 The train_generator and validation generator objects



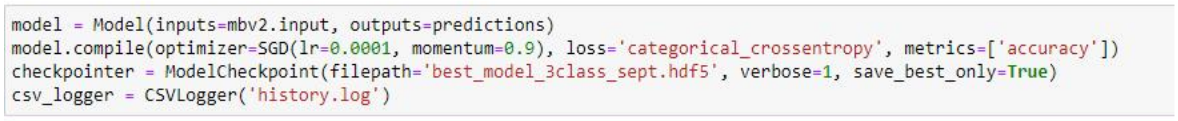
```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

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

5.7.1.5 MobileNetv2 class

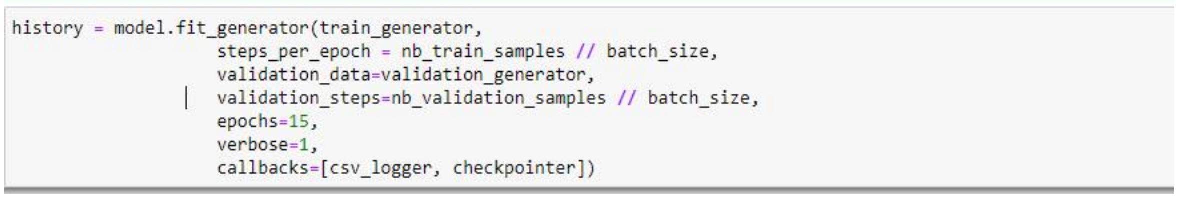


```
mbv2 = MobileNetV2(weights='imagenet', include_top=False, input_shape = (299,299,3))
x = mbv2.output
x = GlobalAveragePooling2D()(x)
x = Dense(128,activation='relu')(x)
x = Dropout(0.2)(x)

predictions = Dense(101,kernel_regularizer=regularizers.l2(0.005), activation='softmax')(x)
```

5.7.1.6 Create and compile and train the model

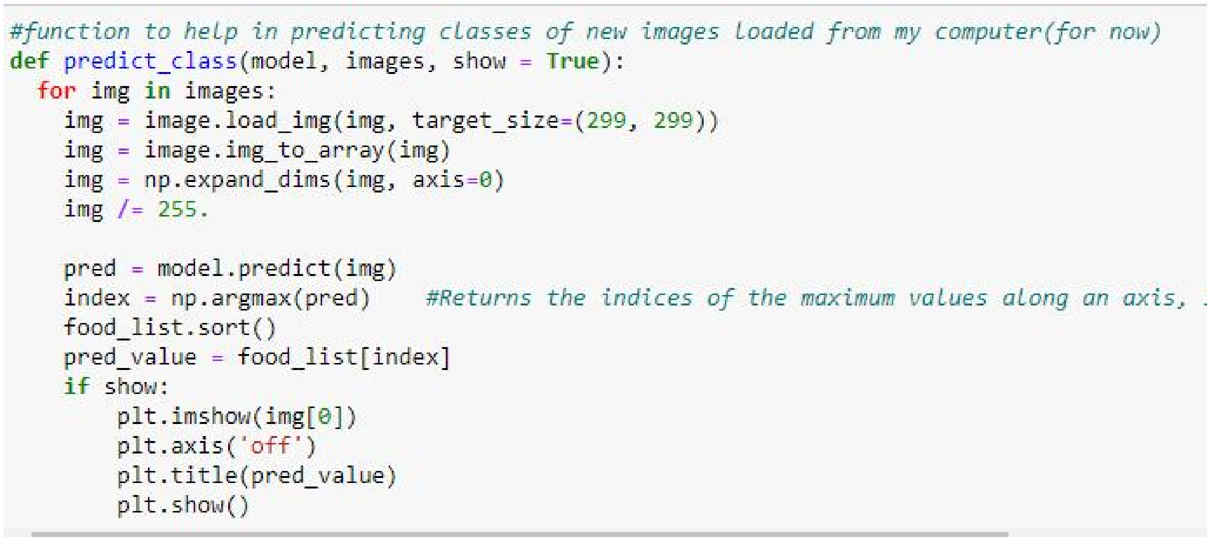


```
model = Model(inputs=mbv2.input, outputs=predictions)
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
checkpointer = ModelCheckpoint(filepath='best_model_3class_sept.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('history.log')
```



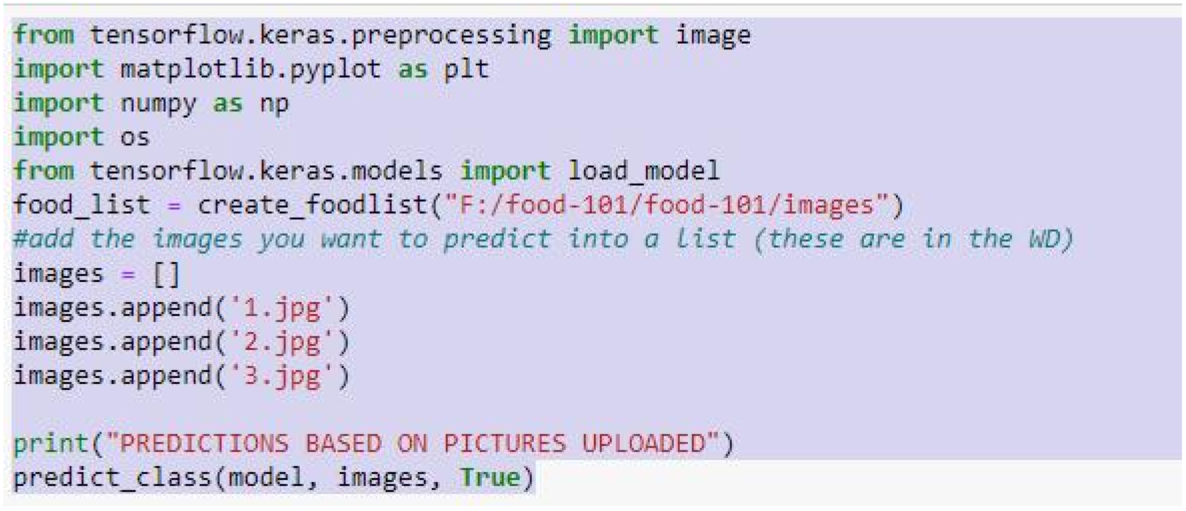
```
history = model.fit_generator(train_generator,
                    steps_per_epoch = nb_train_samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=nb_validation_samples // batch_size,
                    epochs=15,
                    verbose=1,
                    callbacks=[csv_logger, checkpointer])
```

5.7.1.7 Function to predict classes of new images



```python
#function to help in predicting classes of new images loaded from my computer(for now)
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)    #Returns the indices of the maximum values along an axis,
    food_list.sort()
    pred_value = food_list[index]
    if show:
        plt.imshow(img[0])
        plt.axis('off')
        plt.title(pred_value)
        plt.show()
```

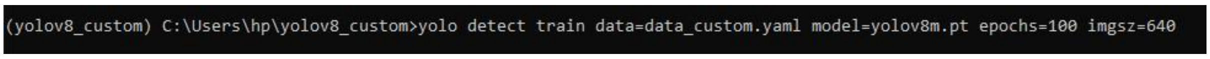
5.7.1.8 Testing new images



```python
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
import os
from tensorflow.keras.models import load_model
food_list = create_foodlist("F:/food-101/food-101/images")
#add the images you want to predict into a list (these are in the WD)
images = []
images.append('1.jpg')
images.append('2.jpg')
images.append('3.jpg')

print("PREDICTIONS BASED ON PICTURES UPLOADED")
predict_class(model, images, True)
```

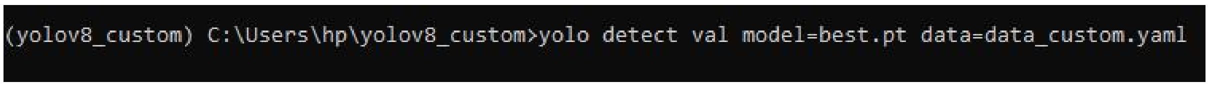
5.7.2 YOLOv8 Model

5.7.2.1 Train YOLOV8 custom model



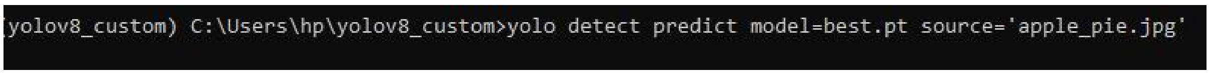
```
(yolov8_custom) C:\Users\hp\yolov8_custom>yolo detect train data=data_custom.yaml model=yolov8m.pt epochs=100 imgsz=640
```

5.7.2.2 Validate YOLOV8 CUSTOM MODEL



```
(yolov8_custom) C:\Users\hp\yolov8_custom>yolo detect val model=best.pt data=data_custom.yaml
```

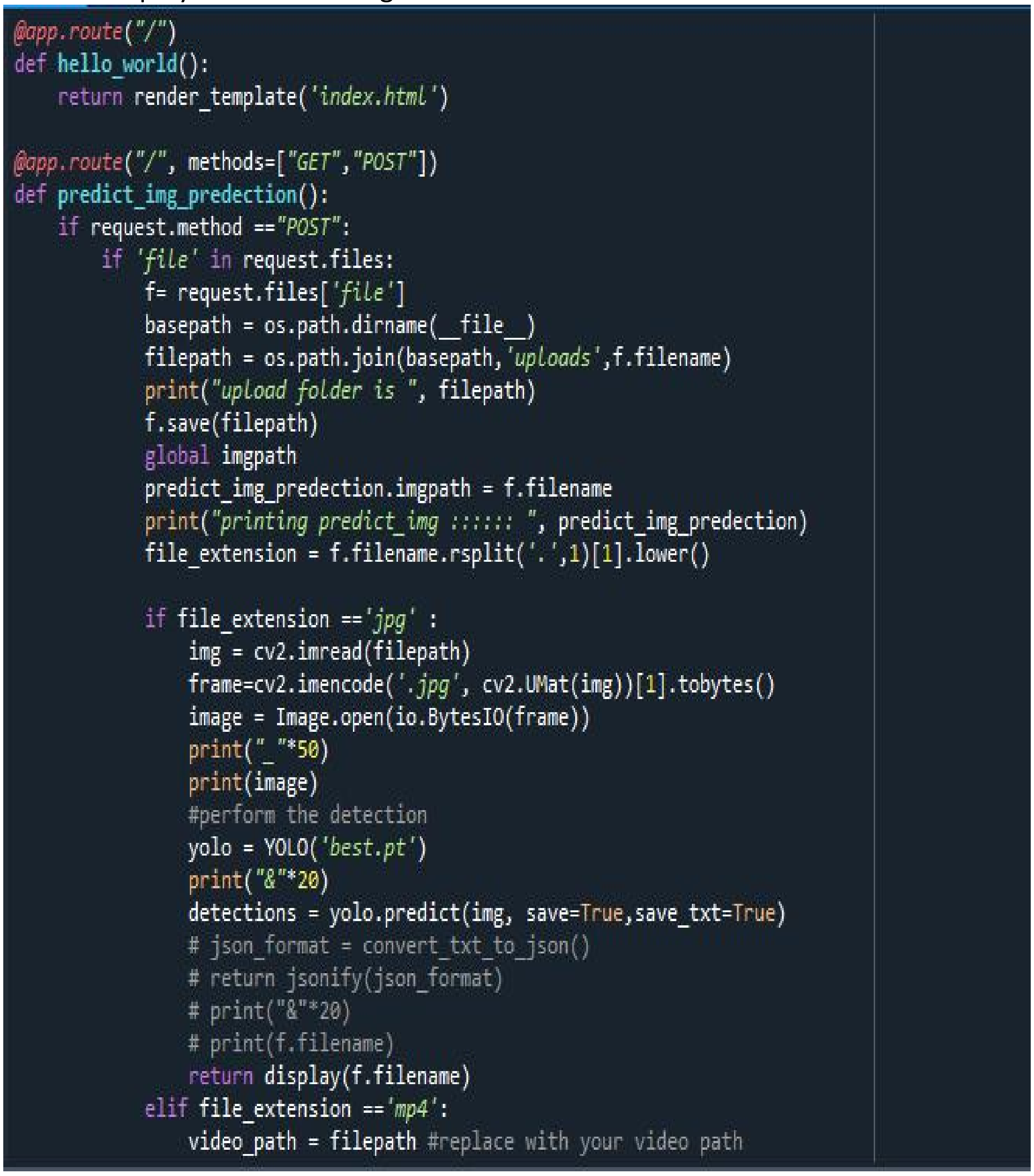
5.7.2.3 Test YOLOv8 custom model



```
(yolov8_custom) C:\Users\hp\yolov8_custom>yolo detect predict model=best.pt source='apple_pie.jpg'
```

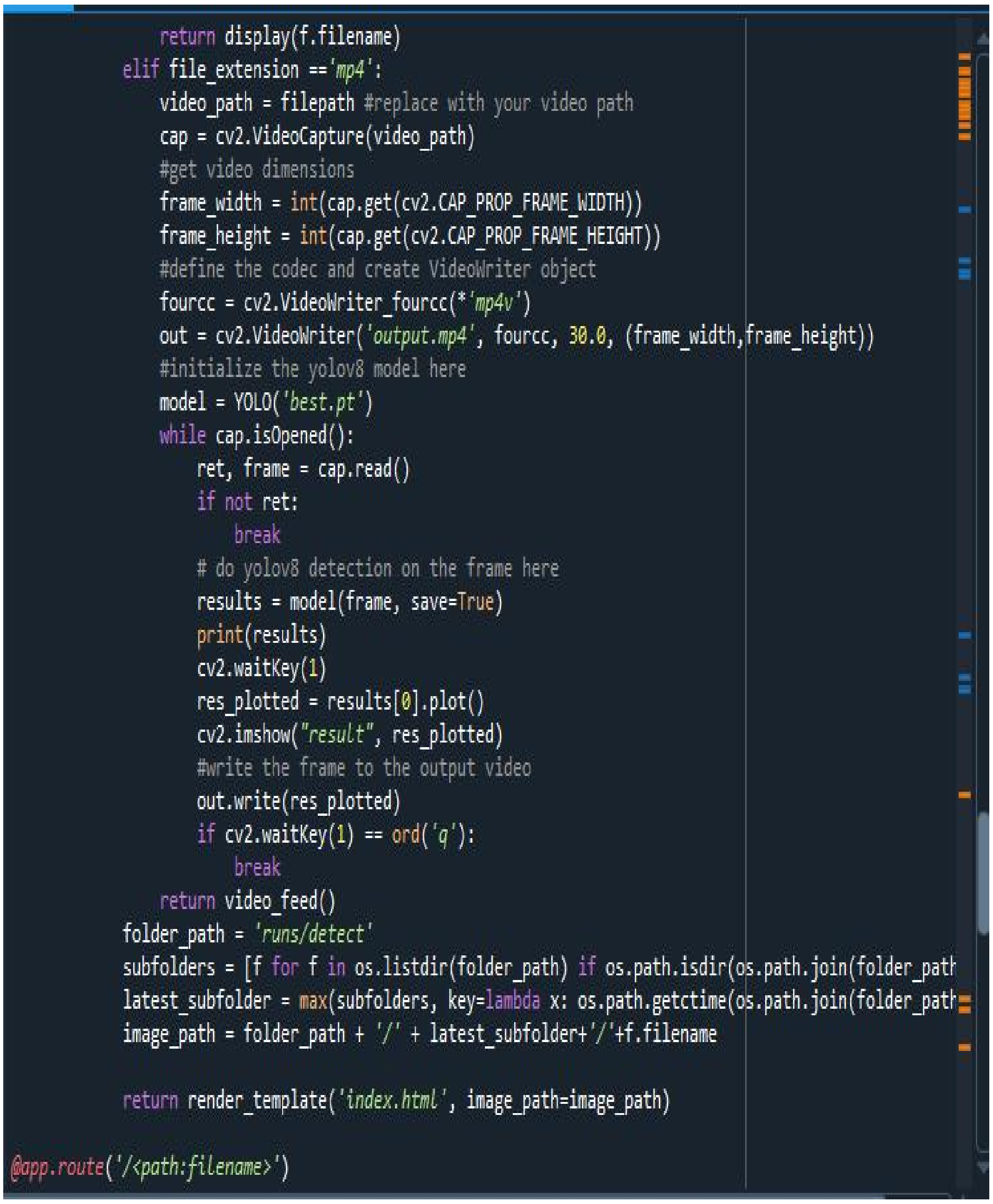
5.7.3 Flask

5.7.3.1Deploy the model using flask



```
@app.route("/")
def hello_world():
    return render_template('index.html')

@app.route("/", methods=["GET","POST"])
def predict_img_predection():
    if request.method =="POST":
        if 'file' in request.files:
            f= request.files['file']
            basepath = os.path.dirname(__file__)
            filepath = os.path.join(basepath,'uploads',f.filename)
            print("upload folder is ", filepath)
            f.save(filepath)
            global imgpath
            predict_img_predection.imgpath = f.filename
            print("printing predict_img :::::: ", predict_img_predection)
            file_extension = f.filename.rsplit('.',1)[1].lower()

            if file_extension =='jpg' :
                img = cv2.imread(filepath)
                frame=cv2.imencode('.jpg', cv2.UMat(img))[1].tobytes()
                image = Image.open(io.BytesIO(frame))
                print("_"*50)
                print(image)
                #perform the detection
                yolo = YOLO('best.pt')
                print("&"*20)
                detections = yolo.predict(img, save=True,save_txt=True)
                # json_format = convert_txt_to_json()
                # return jsonify(json_format)
                # print("&"*20)
                # print(f.filename)
                return display(f.filename)
            elif file_extension =='mp4':
                video_path = filepath #replace with your video path
```

5.7.3.2 Deploy the model using flask.



```
            return display(f.filename)
        elif file_extension =='mp4':
            video_path = filepath #replace with your video path
            cap = cv2.VideoCapture(video_path)
            #get video dimensions
            frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
            frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            #define the codec and create VideoWriter object
            fourcc = cv2.VideoWriter_fourcc(*'mp4v')
            out = cv2.VideoWriter('output.mp4', fourcc, 30.0, (frame_width,frame_height))
            #initialize the yolov8 model here
            model = YOLO('best.pt')
            while cap.isOpened():
                ret, frame = cap.read()
                if not ret:
                    break
                # do yolov8 detection on the frame here
                results = model(frame, save=True)
                print(results)
                cv2.waitKey(1)
                res_plotted = results[0].plot()
                cv2.imshow("result", res_plotted)
                #write the frame to the output video
                out.write(res_plotted)
                if cv2.waitKey(1) == ord('q'):
                    break
            return video_feed()
        folder_path = 'runs/detect'
        subfolders = [f for f in os.listdir(folder_path) if os.path.isdir(os.path.join(folder_path
        latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(folder_path
        image_path = folder_path + '/' + latest_subfolder+'/'+f.filename

        return render_template('index.html', image_path=image_path)

@app.route('/<path:filename>')
```

References

1-YOLOv5: https://arxiv.org/abs/2009.13226

2- EfficientDet: Scalable and Efficient Object Detection: https://arxiv.org/abs/1911.09070

3-RetinaNet: Focal Loss for Dense Object Detection: https://arxiv.org/abs/1708.02002

4-Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks: https://arxiv.org/abs/1506.01497

5-Mask R-CNN: https://arxiv.org/abs/1703.06870

6-SSD: Single Shot MultiBox Detector: https://arxiv.org/abs/1512.02325

7-Cascade R-CNN: https://arxiv.org/abs/1712.00726

8-CenterNet: Keypoint Triplets for Object Detection: https://arxiv.org/abs/1904.08189

9-CornerNet: Detecting Objects as Paired Keypoints: https://arxiv.org/abs/1808.01244

10- YOLOv4: Optimal Speed and Accuracy of Object Detection:https://arxiv.org/abs/2004.10934

11- ResNet: Deep Residual Learning for Image Recognition: https://arxiv.org/abs/1512.03385

12- VGG: Very Deep Convolutional Networks for Large-Scale Image Recognition: https://arxiv.org/abs/1409.1556 I

13-nception: Going Deeper with Convolutions: https://arxiv.org/abs/1409.4842

14-MobileNet: Efficient Convolutional Neural Networks for Mobile Vision Applications: https://arxiv.org/abs/1704.04861

15-DenseNet: Densely Connected Convolutional Networks: https://arxiv.org/abs/1608.06993

16-EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks: https://arxiv.org/abs/1905.11946

17- AlexNet: ImageNet Classification with Deep Convolutional Neural Networks: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

18- SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size: https://arxiv.org/abs/1602.07360

19- ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices: https://arxiv.org/abs/1707.01083

20-NASNet: Learning Transferable Architectures for Scalable Image Recognition: https://arxiv.org/abs/1707.07012

21- Food-101 – Mining Discriminative Components with Random Forests: https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/static/bossard_eccv14_food-101.pdf

22-Automatic Food Identification and Quantity Estimation for Dietary Assessment: A Review: https://www.mdpi.com/2072-6643/11/7/1516/pdf

23-DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment: https://www.mdpi.com/1424-8220/16/6/945/pdf

24-Food Image Recognition Using Very Deep Convolutional Networks: https://ieeexplore.ieee.org/document/7780459

25-Food Classification Using Machine Learning Techniques: A Review: https://www.sciencedirect.com/science/article/pii/S2405452618303343

26-Food Image Recognition Based on Convolutional Neural Network with Data Augmentation: https://ieeexplore.ieee.org/document/8069817

27-Deep Learning for Food Image Recognition: A Comprehensive Review: https://www.sciencedirect.com/science/article/pii/S2405452618302769

28-A Review of Deep Learning Techniques for Object Detection in Food Images: https://www.sciencedirect.com/science/article/pii/S1568494620302241

29-Food Image Recognition Using Convolutional Neural Networks with Transfer Learning: https://ieeexplore.ieee.org/document/8650168

30-Food Recognition Using Convolutional Neural Networks with Different Preprocessing Techniques: https://ieeexplore.ieee.org/document/8237647

31-Food Classification using Transfer Learning with Inception-v3 Neural Network: https://ieeexplore.ieee.org/document/8333973

32-Food Image Recognition with Deep Convolutional Features: https://ieeexplore.ieee.org/document/7837143

33-A Survey on Food Recognition: Towards the Quantification of Food Intake: https://www.mdpi.com/2076-3417/9/14/2966/pdf

34-A Deep Learning-Based Food Recognition System Using Multimodal Data: https://www.sciencedirect.com/science/article/pii/S1364815219302707

35-Food Recognition Using Convolutional Neural Networks with Data Augmentation: https://ieeexplore.ieee.org/document/8563255

36-Food Recognition Using Deep Learning with Data Augmentation: https://ieeexplore.ieee.org/document/8048747

37-Food Recognition Using Convolutional Neural Networks with Transfer Learning and Data Augmentation: https://ieeexplore.ieee.org/document/8368692

38-A Comparative Study of Convolutional Neural Networks for Food Image Recognition: https://ieeexplore.ieee.org/document/8089886

39-Multi-Scale Convolutional Neural Networks for Food Recognition: https://ieeexplore.ieee.org/document/8359358

40-An Image Recognition System for Food Classification: https://ieeexplore.ieee.org/document/7958697

41-Food Recognition Using Convolutional Neural Networks with Different Preprocessing Techniques: https://ieeexplore.ieee.org/document/8237647

42-Food Recognition Using Convolutional Neural Networks with Transfer Learning and Data Augmentation: https://ieeexplore.ieee.org/document/8368692

43-Food Recognition Using Deep Convolutional Features: https://ieeexplore.ieee.org/document/7837143

44-Food Recognition Using Convolutional Neural Networks with Data Augmentation: https://ieeexplore.ieee.org/document/8563255

45-Food Identification and Calorie Estimation from Smartphone Images: A Case Study: https://www.mdpi.com/1424-8220/17/3/399/pdf

46-Calorie Estimation for Food Images Using Convolutional Neural Networks: https://ieeexplore.ieee.org/document/8369132

47-Food Classification Using Convolutional Neural Networks: https://ieeexplore.ieee.org/document/8674822

48-Food Recognition using Convolutional Neural Networks with Multimodal Fusion: https://ieeexplore.ieee.org/document/9025183

49-Food Recognition using Convolutional Neural Networks with Ensemble Techniques: https://ieeexplore.ieee.org/document/8909855

50-Food Recognition using Convolutional Neural Networks with Different Preprocessing Techniques: https://ieeexplore.ieee.org/document/8237647

51-Yummly API Documentation: https://developer.yummly.com/documentation

52-Yummly API Quickstart Guide: https://developer.yummly.com/documentation/doc_quickstart

53-Yummly API Terms of Use: https://developer.yummly.com/terms

54-Yummly API Client Libraries: https://developer.yummly.com/client_libraries

55-Yummly API Recipes Endpoint: https://developer.yummly.com/documentation/doc_recipes

56-Yummly API Search Endpoint: https://developer.yummly.com/documentation/doc_search

57-Yummly API Metadata Endpoint: https://developer.yummly.com/documentation/doc_metadata

58-Yummly API Getting Started Guide: https://developer.yummly.com/documentation/doc_getting_started

Thank You