

**ACCURATE CALORIE COUNTING ALGORITHM FOR CAREFULLY PLATED  
DISHES**

**BY**

**WOON ZHENG LI**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF BUSINESS INFORMATION SYSTEM (HONS)**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**JAN 2020**

UNIVERSITI TUNKU ABDUL RAHMAN

**REPORT STATUS DECLARATION FORM**


**Title:** ACCURATE CALORIE COUNTING FOR  
CAREFULLY PLATED DISHES

**Academic Session:** JAN 2020

I WOON ZHENG LI  
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

  
\_\_\_\_\_  
(Author's signature)

Verified by,  
  
\_\_\_\_\_  
(Supervisor's signature)

**Address:**  
56, LORONG SRI KUANTAN 25,  
25250 Kuantan,  
PAHANG DARUL MAKMUR

DR. Aun YiChiet  
Supervisor's name

**Date:** 19 April 2020

**Date:** 24 April 2020

**ACCURATE CALORIE COUNTING ALGORITHM FOR CAREFULLY PLATED  
DISHES**

BY

WOON ZHENG LI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

**BACHELOR OF BUSINESS INFORMATION SYSTEM (HONS)**


Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2020

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**ACCURATE CALORIE COUNTING ALGORITHM FOR CAREFULLY PLATED DISHES**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  \_\_\_\_\_

Name : WOON ZHENG LI

Date : 19 April 2020

## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Aun YiChiet who has given me this bright opportunity to engage in a machine learning project. A million thanks to you. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

# ABSTRACT

Calorie tracking technology has taken great strides in a health-oriented society. Sophisticated calorie counters employ artificial intelligence (A.I.) for pervasive and accurate calorie estimation. The general consensus in this domain is to strive for general purpose algorithms which aim to support for more variants of food and cuisine types' recognition. These state of the art methods are tuned for accurate food recognition but do not address for accurate food portion detection. In this project, an accurate calorie counter is designed to consider the amount of ingredients used in a non-occlusive plated environment. The intuition is that food ingredients; despite their underlying calories values; would further contribute significantly and differently to overall calories when varying sizes, slices, portion, pieces or units of the ingredients are used. The outcome of the project is a portion-aware calorie counter that accurately estimate the calorie on a plate depending on the amount of food served.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>ii</b>
<b>ACKNOWLEDGEMENT</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>v</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xi</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Background	1
1.2 Motivation and Problem Statement	2
1.3 Project Scope	3
1.4 Project Objectives	3
1.4.1 Sub-Objective	4
1.4.2 Measureable objective	5
1.5 Impact, Significance and Contribution	5
1.6 Background Information	6
1.7 Achievement	7
1.8 Report Organization	8

<b>CHAPTER 2</b>	<b>LITERATURE REVIEW</b>	<b>9</b>
2.1	Status and Potential	9
2.2	Application Review	9
2.2.1	Calories In Food	9
2.2.2	Lose It!	11
2.2.3	Foodvisor	13
2.2.4	Argus	14
2.2.5	Yazio	15
2.3	Data Collection	17
2.4	Comparison of model in existing applications with proposed model	20
2.5	Model Review	21
2.5.1	Faster-RCNN	21
2.5.2	R-FCN	22
2.5.3	SSD	22
2.5.4	Performance Evaluation Metric for COCO Dataset	22
<b>CHAPTER 3</b>	<b>System Design</b>	<b>27</b>
3.1	Design Process and Workflow	27
3.2	Timeline	43
3.3	Implementation Issues and Challenges	43
3.4	System Workflow	44
<b>CHAPTER 4</b>	<b>Discussion</b>	<b>46</b>
4.1	Technologies Involved	46



4.2	Methodology	47
<b>CHAPTER 5</b>	<b>Implementation and Testing</b>	<b>49</b>
5.1	Sample Results	49
5.2	Case Studies	54
5.2.1	Correlation of accuracy and meal complexity	54
5.2.2	Performance Evaluation	56
<b>CHAPTER 6</b>	<b>Conclusion</b>	<b>58</b>
6.1	Project Review	58
6.2	Further Improvement	59
<b>REFERENCES</b>		<b>60</b>
<b>FINAL YEAR PROJECT WEEKLY REPORT</b>		<b>61</b>
<b>POSTER</b>		<b>73</b>
<b>TURNITIN SIMILARITY REPORT</b>		<b>74</b>
<b>SIMILARITY REPORT</b>		<b>75</b>
<b>CHECKLIST FOR FYP2 THESIS SUBMISSION</b>		<b>76</b>

## **LIST OF TABLES**

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.1	Summary of features in existing applications	20
Table 5.1	Average accuracy of case studies 5-2-1	54
Table 5.2	Average accuracy of case studies 5-2-2	56

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.1	Interface of Calories in Food	10
Figure 2.2	Interface of Lose It!	12
Figure 2.3	Interface of Foodvisor	13
Figure 2.4	Interface of Argus	14
Figure 2.5	Interface of Yazio	16
Figure 2.6	Calories counting result from Argus, Yazio and Foodvisor	17
Figure 2.7	Calories counting result from Argus, Yazio and Foodvisor	18
Figure 2.8	List of popular detection model	21
Figure 2.9	Graphical Concept of IoU	23
Figure 2.10	Graphical image for ground truth box and bbox prediction	24
Figure 2.11	Computation value for mAP calculation example	24
Figure 2.12	Computation value for mAP calculation example	25
Figure 2.13	Precision-recall curve	25
Figure 2.14	List of popular detection model	26
Figure 3.1	List of public datasets	28
Figure 3.2	Sample data for training	31
Figure 3.3	Samples label	32
Figure 3.4	Label data .csv file	32
Figure 3.5	Class Name for label map	33
Figure 3.6	Class ID name for training file	34
Figure 3.7	Class Number and Input Path Labels	35
Figure 3.8	Training process	36
Figure 3.9	Tensorboard graph	37
Figure 3.10	Nutritional information for each class	38
Figure 3.11	Code to generate boxes and count for inference model	39

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 3.12	Jupyter Notebook path modification	40
Figure 3.13	Web application interface design	41
Figure 3.14	Threshold, detection and value output for web-based app	42
Figure 3.15	Gantt chart	43
Figure 3.16	Workflow of training process	44
Figure 3.17	Workflow for web application	45
Figure 4.1	Prototype methodology	48
Figure 5.1	Result sample 1	49
Figure 5.2	Result sample 2	50
Figure 5.3	Result sample 3	50
Figure 5.4	Result sample 4	51
Figure 5.5	Result sample 5	51
Figure 5.6	Result sample 6	52
Figure 5.7	Result sample 7	53
Figure 5.8	Result sample 8	53

## LIST OF ABBREVIATIONS

<i>API</i>	Application program interface
<i>CPU</i>	Central Processing Unit
<i>CNN</i>	Convolutional Neural Networks
<i>GPU</i>	Graphic Processing Unit
<i>FASTER-RCNN</i>	Faster Region Convolutional Neural Networks
<i>AP</i>	Average Precision
<i>mAP</i>	Mean Average Precision

# **Chapter 1 : Introduction**

## **1-1 Background**

Calorie counting is a process where it uses multiple algorithm such as object detection and is linked to a database where the calorie information of each food and ingredients is stored in it. When a user take pictures of their meal, it will detects the food and automatically count the total calories that is about to be consumed for the users. This is applied in many healthy diet applications to help users to keep track of the numbers of calories consumed in the food so that one will not eat too much.

Since healthy diet is popularly concerned in this modern world. These application able to help the individuals to maintain and improve their overall health by providing relevance information such as calories, nutritional facts as well as keeping track of their food intake. One biggest flaws that these applications have made are that they do not have object counting algorithm. This will affect the counting process and will not able to provide an accurate result during the process. The information are too generalized where the calories are counted in a fixed portion or the user have to manually select their quantities of it, but most of them would not prefer to gone through these hassle. The aim for this project is to further enhance these applications in the form of accuracy in counting food calories by using object detection and object counting so that each of every ingredients and food to be consumed by the user is determined and provide a precise result for them.

Others similar calories counter applications are also studied and reviewed in order to be compared and to have a better understanding in the features that they provide. As for the development of this project, we will be using extreme programming methodology since it requires constant improvement and tries in order to test the accuracy and get a desired result in counting.

## 1-2 Motivation and Problem Statement

There is several major health problems in society today and Obesity is one of the main issues. It has increased for nearly three times as much compared to in year 1975. In 2016, 39% of the adults who are aged 18 years old and above were overweight, and 13% of them were obese (WHO, 2018). It is associated with diseases like cardiovascular, hypertension and also menstrual problems.

Over time, people have been become more conscious about their diet and attempted to have calories control over the years. People have better self-conscious in taking care of these issues and eat healthier with a proper diet plan. Calorie counting is a common technique used to calculate their energy taken from one's food consumption. Many are used for the purpose of losing, gaining and maintaining weight. In the past, people have been using traditional ways in calorie counting where they estimate the portion of their meals and then do a look up for the number of calories in the calories listing book. But this method is pretty much inaccurate since people required do estimation for their food portion and the process is very time consuming where they need to look for different listing from the book.

As for now, people have been using modern ways to do calories counter. Many health-based applications have featured this function in it, where they provide a calories information database for the users and they can search manually through it to do the food listing, then the application will summarize and do the calculation for the counting. Nutritional information will also be listed to let the users to know more. Moreover, some applications allow users to capture a picture of their meal, algorithm is then used to process the image and automatically detect the objects, listing down all the nutritional facts and calories information related to the food in a generalized portion.

Although these applications are able to do the basic calories counting, but the result is based on a generalized portion and the amount can be customized by the users. This will lead to inaccurate result since the portion itself is selected by the users. Users might not know how much is the proper amount of food they are consuming, the calculations for calories counting could be wrong. However, this can be improved and enhanced using object counting algorithm. The idea here is to classify each of the class into different kind of portion with an appropriate calories information, then the applications should able to do quantities counting from the image taken by the user. Therefore, the algorithm is able to segment the portion size of the food itself and then determine the correct quantities, and

finally provide a more accurate and reliable calories information based on the values. This is aimed to further enhance calories counting by improving the accuracy of the result and avoid the hassle of having users to manually do a look up in the listing, the process could be shortened and assist them in planning a more reliable diet.

### **1-3 Project Scope**

The end product will be an improvement plan and developing a model to further enhance the calories counter feature for simple plated dishes. It should consist of a robust detection algorithm where it can detect different sizing and portion of the ingredients captured in image form correctly. It also able to do quantities counting as well to provide an accurate result at the end of the calculations. This project also come with relevant complete trained model in order to test the functionality and the accuracy of it.

### **1-4 Project Objective**

The first objective is to design an object counting algorithm derived from crowd counting technique for food portion recognition. The algorithm should able to count the total quantities of the ingredient in the image right after classifying them. The value will be later used to calculate an accurate total calories for the food in the image.

The second objective is to fine-tune the neural network (CNN) in terms of image intensity, dataset quality, data augmentation, optimising epoch and network layers for a more accurate counting. There are a few factors such as image intensity, angle of the image as well as labelling techniques that will affect the outcome. Therefore, multiple approaches is being used to determine which techniques should be selected as best practices in the process of training the model to obtain the most accurate and reliable result.

The third objective is to integrate object occlusion detection technique for detecting non-overt ingredients on a plate. Usually food images comes with objects that overlaps each other and this situation affect the detection process. It leads to problematic situation and low reusability where users have to retake an image with different angle. Object



occlusion techniques will further enhance the detection algorithm to be able to detect overlapped objects.

The fourth objective is to develop a script for automating the overall calorie counting for a plate with composite ingredient. The script required to automate the process of calorie counting, by multiplying respective calories of their classes with the total number of quantities detected in the image.

The fifth objective is to establish a suitable database of calories information for each of the class, so that the model can retrieve the information during the inference run and display the appropriate calories corresponding to the count. The database should be developed in a way that it is easy to link between the model as well as storing and retrieving only necessary information to use.

The last objective is to deploy the trained detection model into a web based application with simple interface. The interface should consist of live capture feature which enables users to input images using live capture and run the inference on the input for detection, counting and retrieving calories information. It will then display all the necessary information after running the inference at the backend.

### **1-4-1 Sub-Objectives**

There are some sub-objectives that need to be achieved. These objectives are related and will affect the implementation and success of the main objective.

- A) The sub-objective for this project would be to improve the accuracy and reliability of the calories counting. Accurate calories counting and more efficient process can greatly increase the reusability of this function that is widely used in health-based applications. As a result, an accurate calories counter could change the perception of peoples' way in calculating calories, high reusability could lead them in giving more concern on their calories intake by constantly keeping track of it.
- B) Another sub-objectives also aim to reduce the error and detection failure in the object-detection algorithm to avoid users' frustration and increase its reusability.

- C) To find out a suitable approach for accurate calories counting algorithm for simple plated meal. In our case, we would like to find an appropriate approach to segment all the classes into smaller portion in order to achieve accurate calories counting.

### **1-4-2 Measureable Objectives**

The project is to improve the accuracy for calories counter. This project is strived to increase at least 15 percent of the accuracy in calories counting compared to the results from other health-based applications available in the market. This model is hoped to be used widely in health-based applications and increase its reusability by 10 percent so that users can keep continue using this feature to keep track of their diet.

### **1-5 Impact, Significance and Contribution**

This project will be beneficial to those who have concern on their diet and would like to always keeping track of their calorie intake. Accurate calorie counter can assist people in maintaining a good healthy weight, monitoring and regulating their daily intake can keep them in fit. Accurate calorie counter can also help in planning a good plan in reducing risk in health issues such as obese and underweight, it helps you to maintain at a healthy weight where it helps to achieve more important goal of keeping health problems at bay. Moreover, it helps in training peoples' discipline when they start counting their daily calories intake, making them to know more about unhealthy eating habits and make effort in quitting and replace these bad habits with a healthier one.

This could impact and lead the society towards a healthier practice, giving more concern to their diet problems and possible issues that they could encounter if they do not take care of it. Overall, it will increase the reusability of the feature as it reduces detection failure, a more robust and accurate calculations along with the help me quantities counting.

## **1-6 Background Information**

There are multiple fields of knowledge to be involved to carry out this project. The main fields that are required are Convolutional Neural Network (CNN), Python (Programming Language), Digital image processing and Machine Learning.

Python is a high level programming language that serves general purposes. It uses a notable significant amount of whitespace to emphasize on its code readability. It is object oriented aimed to help the users to write logical code for projects scaled small to large in clarity. It was release in the late year 1980s and is now one of the most used programming languages. It is a multi-paradigm programming language that supports functional programming and aspect-oriented programming. Since we are doing machine learning, python is more suitable since it offers more concise and readable code. It also have extensive selection of libraries and frameworks to support artificial intelligence algorithm. It also consist of great community and popularity where there are many users and materials out there to be referred.

Next, Convolutional Neural Network is used in deep learning to analyse about visual imagery. CNN uses different approach towards regularization, they uses advantage in the hierarchical pattern in data and assemble a much more complex patterns using smaller and simpler patterns. Therefore, CNNs are considered on a lower extreme compared on the scale of connectedness and its complexity.

Digital image processing uses the computer algorithm to do image processing on digital images. It consist of much wider range of algorithm to be used into the input data and avoid issues such as distortion of voice and signal during the process. It can be modelled in the form of multidimensional systems.

Machine Learning is the study between algorithms and scientific study that is used by computers devices to perform task without using instructions, they uses the patterns and inference which considered as part of artificial intelligence to build model on top of sample data, also known as training data. Machine Learning can make decisions and predictions without being programmed to do the task. It is widely used in variety of applications involved data mining, which focus on data analysis and unsupervised learning.

## **1-7 Achievement**

Highlights and objective achieve through this project all as listed below:

- Developed an object-detection model for clean plated meals
- Integrated counting algorithm in the object-detection model
- Integrated the model to retrieve information for calories information when running the inference model
- Build an interface to run the inference
- Enabled live-capture feature for users to input for the inference run
- Able to display necessary information quickly after running inference through live-capture

## **1-8 Report Organization**

The report is organized with different chapters each with specific contents. There are a total of 6 chapters in this report. Chapter 1 is basically the introduction to this project work, discussing on the project background, our objectives, problem statement and also the motivation to start the project. All the achievements done in this project is summarized in the achievement section. Chapter 2 is for the literature review content where we review existing applications to determine current technologies used and approaches for calorie counter application. In this, we will determine what work could be done to improve the existing technology and using better approach to enhance the usability and performance. Also, we will be reviewing the existing models for detection so we could compare and decide the suitable model to be used for our project. Chapter 3 is discussing all the details on the development of the project. To provide all the necessary details and the steps so that someone allowed to rebuild the system with ease. In our case, we will be discussing on the training process to develop model for back-end to building front-end interface required to run the application. Chapter 4 is regarding to the design specification, discussing on the methodology used and the tools required to build the project. Lastly, Chapter 6 is the conclusion that we have obtained during the project development and also describing what can be achieved or to have future improvements or development.

## **Chapter 2: Literature Review**

### **2-1 Status and Potential**

With a lot of health-based applications out in the market, these applications seem to have a positive impact and be beneficial for the society, especially those who are health concerned. According to a survey done by (Krebs and Duncan, 2015), 65.5% of their participants in the survey opened their health application at least once per day and 44.4% of them used for 1-10 minutes. People have been using health-based applications to keep track of their food intake, recording calories intake and checking out the nutritional facts. However, according to (Lyles and co-authors, 2011), part of their patients do not find that mobile applications are useful for them as they have to learn to use smartphones which has affected their overall experience with mobile applications.

Overall, there is an increasing number of people starting to use these health-based applications on mobile phones in their daily life and it is proven to be beneficial and useful to them, having a positive impact on their healthcare.

### **2-2 Application Review**

In order to determine what are the existing functions and features that are available in other calorie counting/fitness applications, these applications are being reviewed. This will help to identify issues within the system and some useful features can be taken into consideration in improving them. Below are the applications that have been reviewed :

#### **2-2-1 Calories in Food**

The applications have the below features :

- Search feature
- Sorting function
- Calories and nutritional facts reference

- Multiple languages to select

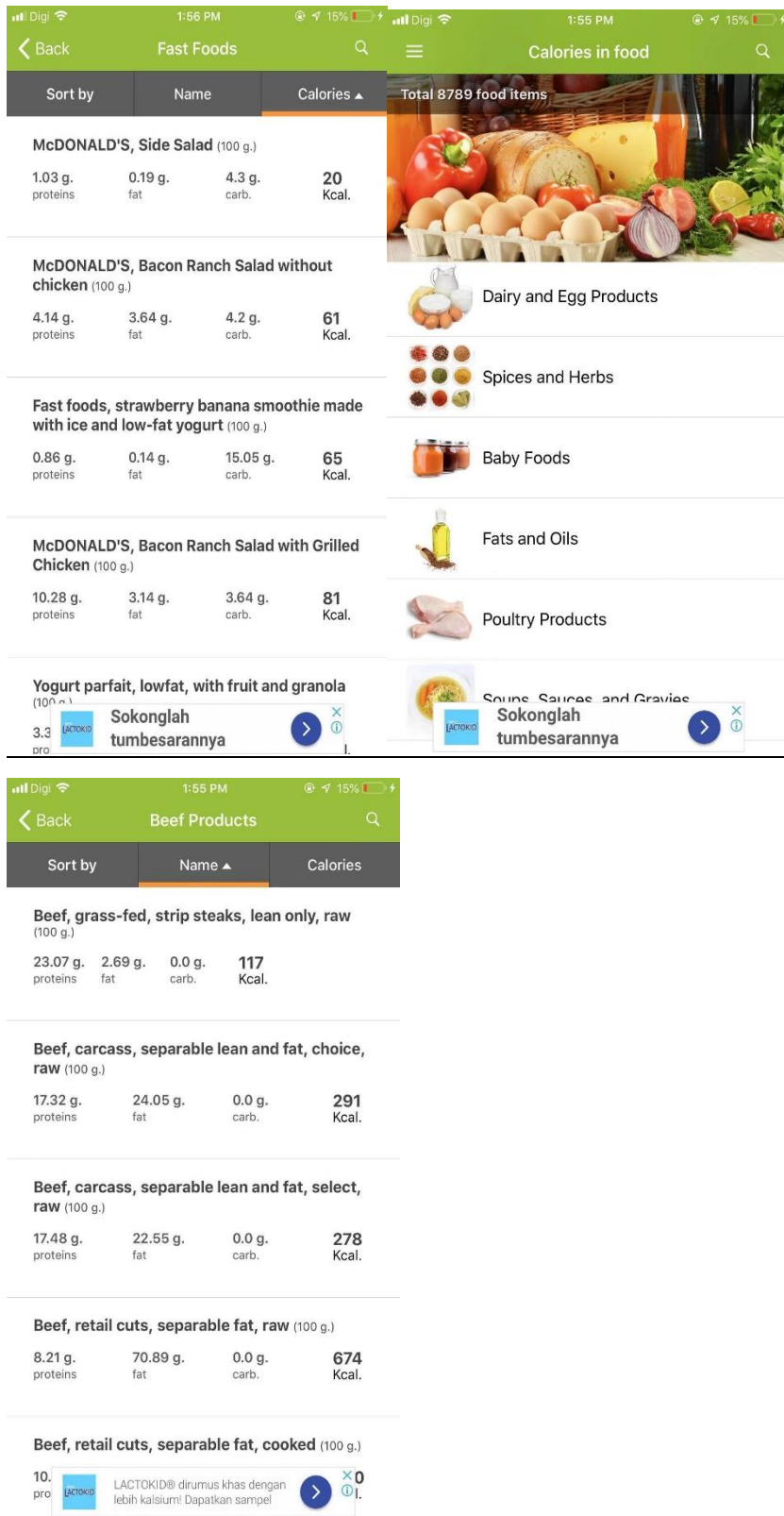


Figure 2-1: Interface of Calories in Food

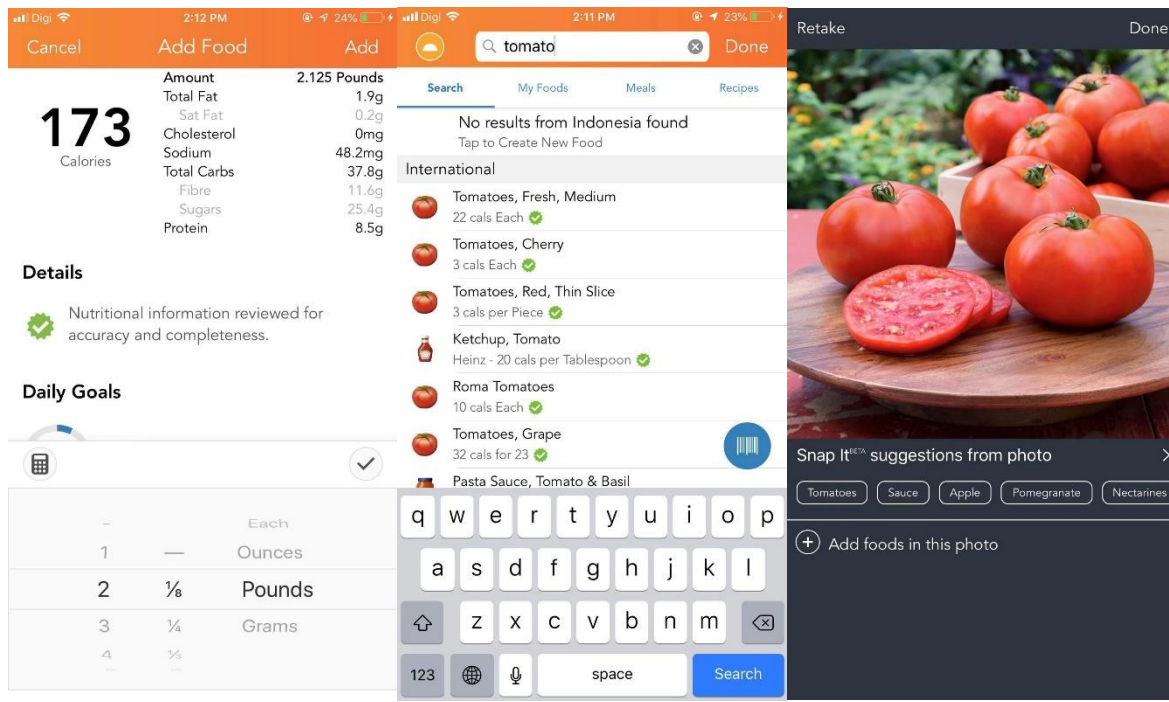
Calories in food is a very simple application that allows user to check calories for their food but had no any other functions. It basically provide 3 nutritional facts for each food and its total calories for a fixed portion of 100g. It also gives option to user whether to sort by name or calories from ascending or either descending order. This application also allows user to use the search function to quickly access to the information that they are looking for. However, this applications does not allow the user to customize any of their information. Usually one will able to customize the size of its portion, and the application will automatically calculate the precise calories for their food. Moreover, the name for each of these ingredient could be lengthy and it is very difficult to read. Users also have to manually search and calculate their calories since it does not provide any feature that can shorten the process such as detecting and recognizing the ingredient through images or camera.

### **2-2-2 Lose It!**

Application has the below features:

- Challenges and Bonus Feature
- Social and Chat Function
- Weight Goal Function
- Calories Reference
- Food Tracking (Taking photos and recording food intake)
- Nutrition Tips
- Search function
- Customizable Portion and Recipes
- Exercise Browser
- Barcode scanner





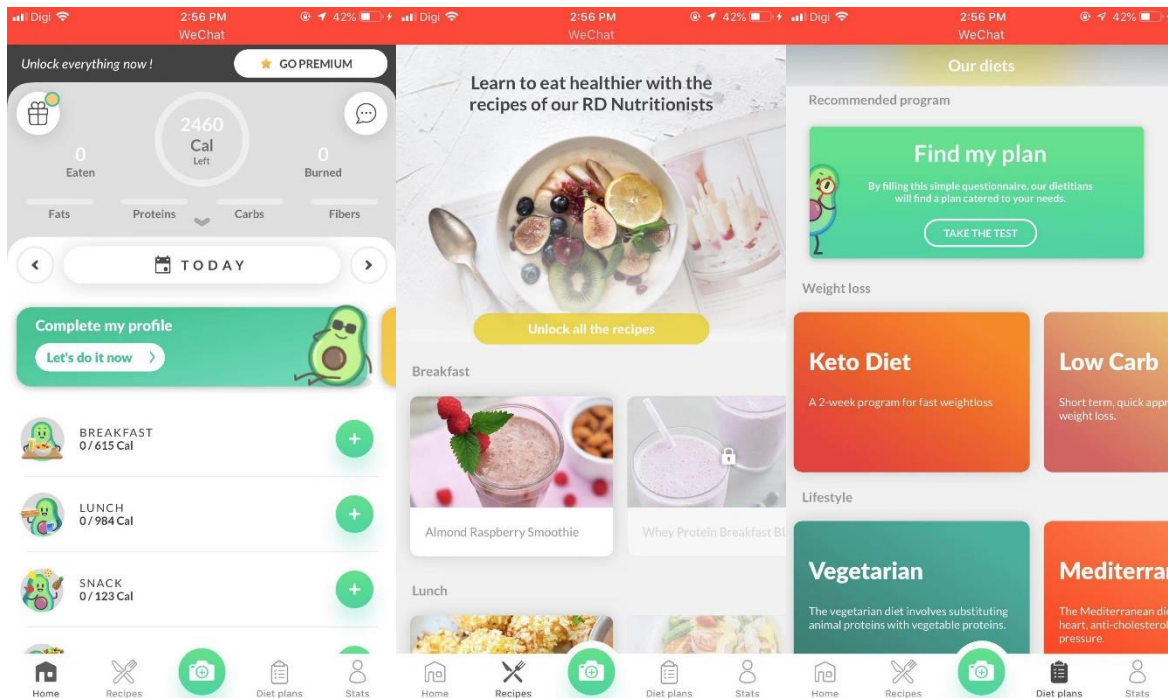
**Figure 2-2: Interface of Lose It!**

Lose It! is a popular application that is used by many users. It basically emphasize on weight goals and calories intake. It consists of multiple challenges that can keeps the users motivated for the purpose of losing their weight or even gaining. It also records the calories intake of the users and is shown through a progress bar in the main interface of the applications so user can easily keep track of it. User can additionally add in some of their food photos for everyday meal and is recorded in the cloud, user can refer back to the past activities if they desire. Moreover, the applications allow user to access to social media function which allows them to post their activity on their profile, adding their friends through email and can instant message their friends through one to one or group chat. Last but not least, one most interesting function that this application features would be the barcode scanner, it allows the camera to scan through the barcode then the nutritional facts of the item is immediately shown to the user. However, the application has a complex interface and does not detect the image precisely, it does give suggested items in the image but do not immediately give user the calories information, the user have to press the item manually and is directed to the calories database. This process would be time consuming for the users.

## 2-2-3 Foodvisor

The application has the below feature:

- Calories Tracker
- Recipes Guide
- Diet Planner
- Dashboard (Graph)
- Personal Consultation
- Food detection (camera)
- Weight monitoring



**Figure 2-3: Interface of Foodvisor**

Foodvisor is one of the most accurate application that exist in the market for food detection to check calories. It provides users the recipe guides and is updated every day as a reference for cooking, it list down the nutritional information, ingredients required and directions for cooking. It also provides different categories of diet plan so that users can plan their diet accordingly to their style, the diet plan is synced with the preferred recipes for that particular category. It also provide a clean interface design and a dashboard to provide a clear information of their daily progress. This is the only application that came

across and automatically provide an estimated calories after detecting the food through an images. However, the detections have flaws such as detection failure and unable to recognize the food accurately, there are several tries where I could not detect the food and still have to access to the database manually to select the item.

## 2-2-4 Argus

The application has the below features:

- Heart rate checker
- Steps tracker
- Tips and tricks
- Friends & followers
- Food and water tracker
- Challenges and goals

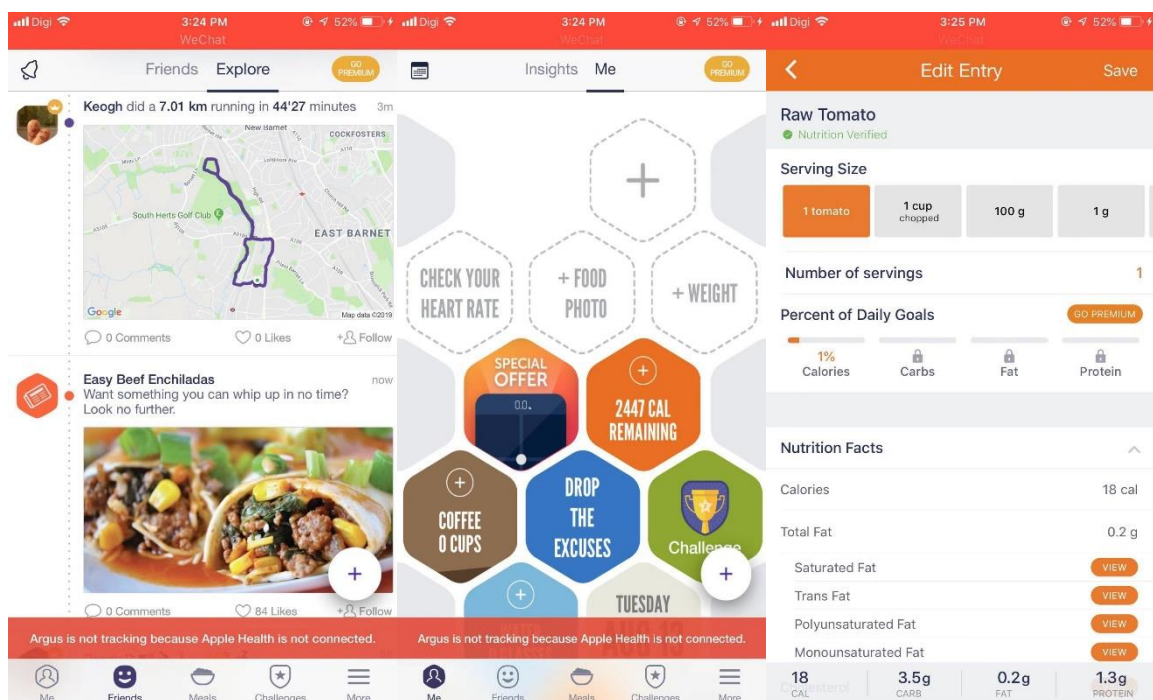


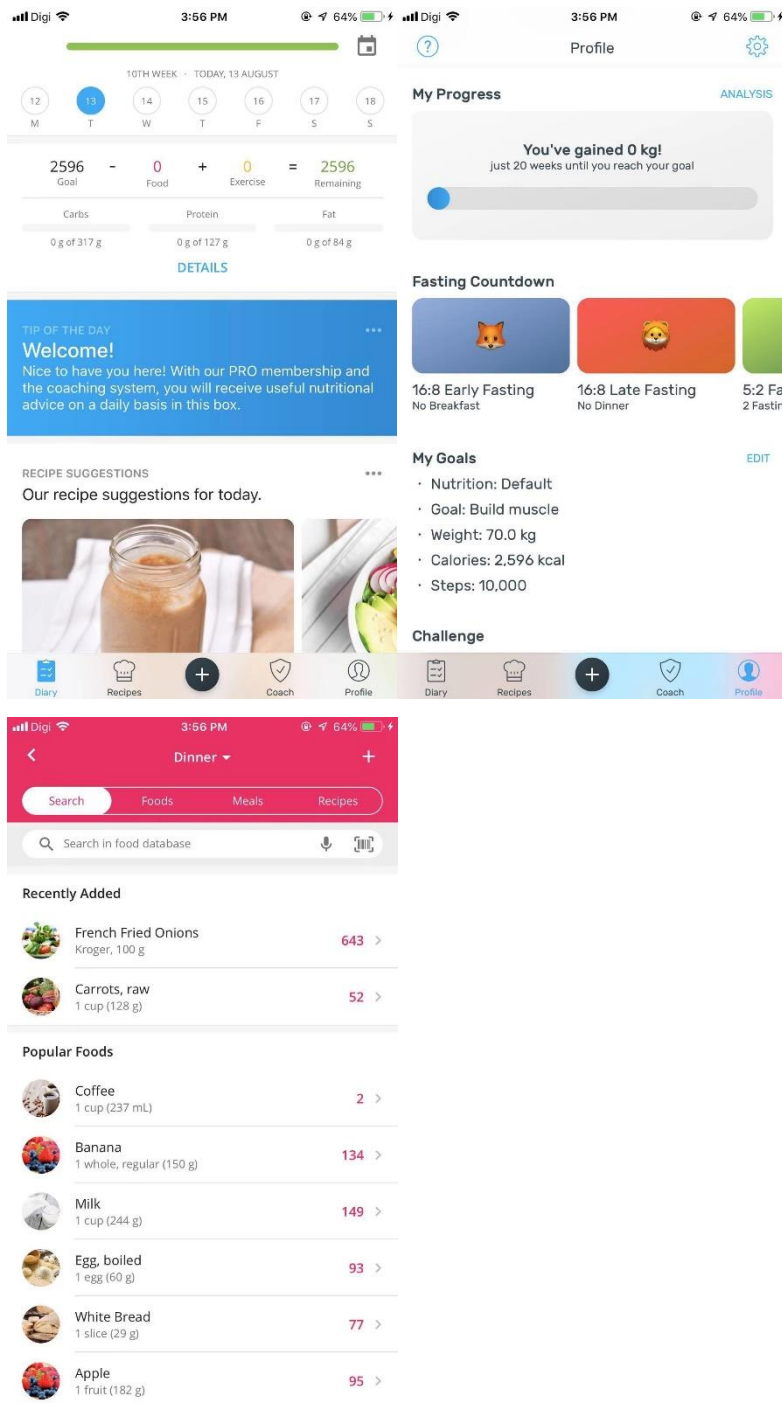
Figure 2-4: Interface of Argus

This application features some unique function like heart rate checker. It can be synced with Apple Health to track the overall progress of your health. It also can check your heart rate by manually placing a finger on the camera. It allows users to keep track of their food and water intake, providing meal plans and recipes to suit their eating style. It also consist of a social platform where users are able to share their fitness achievement, their sleep log and their daily activities. The applications provide some challenges and goals to keep users motivated in continuous effort for self-improvement and health concern. As for calories tracking, users are allowed to take photo and suggested food item list will be shown. Then it navigate the users to the database to select the correct portion then finally calculates the calories. There could be improvement where the process of tracking can be shortened and calculate calories for the users without having to navigate them and to select manually.

### **2-2-5 Yazio**

The applications has the below feature:

- Calories and nutrition tracker
- Recipes guide
- Coaching
- Sync function with other health applications
- Water tracker
- Health measurement (glucose level, blood pressure, weight)



**Figure 2-5: Interface of Yazio**

Yazio first requires setting information from the users such as their weight, height, age and their exercising intensity. Then it analyse the suitable profile for the users and customize for them. It display the nutritional intake required in progress bars and have recipes suggestion daily. It also record down their water intake and track their daily steps. It contains a recipes tab where users can search for their desired recipes, it contains the

nutritional facts and preparing direction information in each recipes. It also contains coach function where they assignment a professional assistance to help users to keep track of their health progress and planning their diet. As for calories counting, users have to manually navigate and customize their serving size to calculate the calories. However, it does not feature automation on the calculation process. Recipes and Coach function are only available for those who subscribed annually or monthly to their services for quite a value.

## 2-3 Data Collection

Calorie Table													
	ARGUS	YAZIO	FoodVisor		ARGUS	YAZIO	FoodVisor		ARGUS	YAZIO	FoodVisor		
<b>Ingredients</b>				<b>Tofu</b>				<b>Sweet Potato</b>					
Serving Size				Raw (100g)	76	76	120	Raw (100g)	86	86	79		
				Raw (300g)	228	228	360	Raw (300g)	258	258	237		
<b>Carrot</b>													
Raw (100g)	41	41	28	Raw (1cup)	188	188		Sliced (1cup)	210				
Raw (300g)	123	123	84	Raw (3cups)	564	564		Sliced (3cup)	630				
Sliced (1 unit)	50	60		<b>Zucchini</b>				Cubed (1 cup)	114	133			
Sliced (3 units)	150	180		Raw (100g)	17	17		Cubed (3 cup)	342	399			
				Raw (300g)	51	51							
Chopped (1 cup)	52	60		Chopped (1cup)	21			<b>Peanut</b>					
Chopped (3 cups)	156	180		Chopped (3cups)	63			Raw (100g)	567	567	622		
								Raw (300g)	1701	1701	1866		
<b>Red Tomato</b>				Sliced (1cup)	19			Raw (1 cup)	828	828			
Raw (100g)	16	18	16	Sliced (3cups)	57			Raw (3 cups)	2483	2483			
Raw (300g)	48	54	48										
				Medium 'whole (1unit)	33		36						
Chopped (1cup)	25	50		Medium 'whole (3units)	99		108	<b>Green beans</b>					
Chopped (3cups)	75	150						Raw (100g)	27	31	33		
				Sliced (1 unit)	2		2	Raw (300g)	81	93	99		
Crushed (1cup)	57	88		Sliced (3 units)	6		6						
Crushed (3cups)	171	264						Raw (1 cup)	40	25			
				<b>Green peas</b>				Raw (3 cups)	120	75			
Sliced (1 unit)	2	4	4	Raw (100g)	81	81	71						
Sliced (3 units)	6	12	12	Raw (300g)	243	243	213	<b>Corns</b>					
								Raw (100g)	86	86	100		
<b>Potato</b>				Raw (1 Cup)	117	117		Raw (300g)	258	258	300		
Raw (100g)	77	77	130	Raw (3cups)	351	351							
Raw (300g)	231	231	390					Raw (1 ear)	88	123	90		
				<b>Pumpkin</b>				Raw (3 ear)	264	369	270		
Mashed (1cup)	237	215		Raw (100g)	26	26	14	Raw (1cup)	125	125			
Mashed (3cups)	711	645		Raw (300g)	78	78	42	Raw (3cups)	375	375			
<b>Egg</b>				Raw (1 Cup)	30	30		<b>Broccoli</b>					
Raw (100g)	147	143		Raw (3cups)	90	90		Raw (100g)	34	34	29		
Raw (300g)	441	429						Raw (300g)	102	102	87		
Boiled (100g)	155	155	134					Raw (1cup)	30	62			
Boiled (300g)	465	465	402					Raw (3cups)	90	186			
								Raw (1 stalk)	11		43		
								Raw (3 stalks)	33		129		

**Figure 2-6 shows the calories counting result from Argus, Yazio and Foodvisor for multiple types of basic ingredients**

	ARGUS	YAZIO	FoodVisor			ARGUS	YAZIO	FoodVisor
<b>Enoki Mushroom</b>					<b>Asparagus</b>			
Raw (100g)	37	37	45		Raw (100g)	20	20	30
Raw (300g)	111	111	135		Raw (300g)	60	60	90
Sliced (1 cup)	24				Raw (1cup)	27	27	
Sliced (3 cup)	72				Raw (3cups)	81	81	
Whole large (1unit)	2	2			Raw (1 spear)	3	2	4
Whole large (3units)	6	6			Raw (3 spears)	9	6	12
					<b>Salmon</b>			
<b>Capsicum</b>					Raw (100g)	127	208	181
Raw (100g)	20	26	32		Raw (300g)	381	624	543
Raw (300g)	60	78	96		Cooked (100g)	178	206	217
Chopped (1 Cup)	30				Cooked (300g)	534	618	651
Chopped (3 Cups)	60				Smoked (100g)	117	152	169
Sliced (1 Cup)	18		12		Smoked (300g)	351	456	507
Sliced (3 Cups)	54		36		<b>Onion</b>			
Medium whole (1unit)	24	51	32		Raw (100g)	32	40	43
Medium whole (3units)	72	153	96		Raw (300g)	96	120	129
					Chopped (1 Cup)	32	64	
<b>Squid</b>					Chopped (3 Cups)	66	192	
Raw (100g)	92	92	80		<b>Cauliflower</b>			
Raw (300g)	276	276	240		Raw (100g)	25	25	28
Serving (1unit)	78		120		Raw (300g)	75	75	84
Serving (3units)	234		360		Chopped (1cup)	27	27	
					Chopped (3cups)	81	81	
<b>Octopus</b>					Medium whole (1u)	147		161
Raw (100g)	82	82	86		Medium whole (3u)	441		483
Raw (300g)	246	246	258		Floweret (1 unit)	3	3	3
					Floweret (3 units)	9	9	9

**Figure 2-7 shows the calories counting result from Argus, Yazio and Foodvisor for multiple types of basic ingredients**

The data collections are done using applications that features camera food detection and calories tracking to test its functionality and its preciseness on the result. The samples that were used to test are using simple and basic ingredients that are most probably available in the database for all of these applications to ensure comparison could be done. Images of these ingredients were prepared to be used during the testing session. Since Yazio and Argus does not include automation during the calculation process. All the ingredients have to manually selected by the users after the detection at the suggestion list, if the list is unrelated, users have to manually navigate to the calories database to search for the name. Portion and serving size are also manually selected by users since they do not have object

counting or portion detection in their algorithm. This process could be very time consuming as users basically go through the process manually. Meanwhile, Foodvisor is much better as they provide the nutritional information right after the application recognize the food from the images. However, its algorithm are not able to detect some of the object with ease. Many tries resulted to detection failure and ended up users still have to navigate to its databases or retaking the photo with an angle where it can recognize it better.

22 ingredients were selected to obtain the calories result from these 3 applications. We then segment it into the lowest basic unit which is 1 and a multiply of 3. We aimed to test if the applications can recognize the portion and serving size of the food in the image, the end result is that they do not involve algorithm in counting. Every time a user upload or take a picture of their food, the users were prompted to select their desire serving size manually after detecting the object type. Thus, these applications does not automate counting but is customizable to change to serving size. As for the calories result, majority of them can be seen as inconsistent between the applications since the predefined serving size that is saved in their databases are not the same. Some of the data were contributed by third parties and is verified by the developer, which serves in different serving size. This lead to a situation where users will ended up getting different result from their actual meal, affecting their diet plan. Moreover, some serving size in certain food type were not able to be found in some of these applications. For example, if the users were to find the calories of a cup of chopped onion, the application will not able to provide the information needed by the users since it is only available in the unit of grams. The blank unit in the excel file indicates that the application does not have that particular units of measurement for that food in their databases.



## 2-4 Comparison of model used in Existing Applications with Proposed Model

Model	Object detection	Object counting	Time consumption	Accuracy
HealthifyMe	No	No	Very High	Moderate
Calories in Food	No	No	Very High	Low
Lose It!	Yes	No	High	Moderate
Calorie Mama	Yes	No	High	Moderate
Foodvisor	Yes	No	Moderate	Moderate
Argus	Yes	No	Moderate	Moderate
Yazio	No	No	Very High	Moderate
<b>Proposed Model</b>	Yes	Yes	Low	High

**Table 2-1: Summary of features in existing applications**

The existing applications are compared in the terms of the use of object detection, object counting algorithm, the consumption of the time used to acquire the result and their accuracy in these applications. The object detection is where the applications allows users to upload an image or taking a photos to automate the process of searching the food information instead of a manual way of doing it.

As for the object counting, it could be used to automate the process of determining the portion and serving size of the users' meal. For the existing applications, the serving portion is customizable to be selected by them and minority of them do not provide this function. The current existing databases all have predefined values for their measurement, so the end result that delivers to the user might be inaccurate to their actual values.

The time consumption is the time taken for users to acquire their result in doing calories counting, the existing applications take moderate to high time consumption as users were required to manually search for their food one by one to add up the calories, some were automated like Foodvisor which gives a calorie value after detecting the object without having the users to navigate to the other page. Overall, it is a hassle for users to calculate just calories by having them navigate here and there.

The accuracy would be the main aspect of evaluating these function. The accuracy in the table represents the accurateness of the application in object detection and delivering

an accurate calories counting result that is near with the actual values. The object detection function still have much room to be improvised as the current applications have a high chance unable to detect the object from different angle. The picture taken must be clear and precise at a direct angle to ensure that the object could be recognized. As for the calories part, the predefined serving size could be varies and availability of unit of measurement is low to meet users' requirement.

## 2-5 Model Review

Object detection has been widely explored in different challenges like COCO Challenge, ILSVRC, and PASCAL VOC and other variety of methods have been proposed in the past. CNNs are becoming very popular and is leading techniques for object detection, we are about to compare three convolutional object detectors which are Faster-RCNN, R-FCN as well as SSD. These detectors use different feature extractor such as ResNet and MobileNet.

Object detector	Feature extractor	COCO mAP	Test time (sec/img)
Faster R-CNN	ResNet-101	30	1.627
Faster R-CNN	Inception v2	28	1.121
R-FCN	ResNet-101	32	1.912
SSD	MobileNet	22	<b>0.688</b>
SSD	Inception v2	24	0.903

**Figure 2-8 show a list of popular model and their performance for model detection**

### 2-5-1 Faster Region-Based Convolutional Neural Network (Faster-RCNN)

It is a state-of-art method that are used because it improve the speed of R-CNN and Fast R-CNN by generating the region proposals by a Region Proposal Network(RPN) instead of using hand-crafted features. It involves two steps: (1) RPN takes the input image and then extract its image features using CNN. They are used in predict object proposals and corresponding objectness scores. In the training of RPNs, the system utilizes anchors using different scales and ratios to generate the region proposals according to the CNN features. Then, the region proposals and features maps are transferred to the Region of Interest (ROI) Pooling Layer to crop and resize the feature maps. Then, it is fed into fully connected layers

of the CNN. It aims to predict the class probabilities and its corresponding rectangular bounding boxes for the labels. A regressor refines the position of each rectangular proposals and outputs its final coordinate of the predicted bounding boxes. It shares the convolutional features with the detection network to reduce computational costs of the region proposal method.

### **2-5-2 Region-Based Fully Convolutional Network (R-FCN)**

R-FCN applies a different approach compared to Fast-RCNN and Faster-RCNN which use subnetwork per-region several hundred times. It is a fully convolutional network and it is based on the image convolution computation. It introduces position-sensitive score maps that encodes the position as for that a relative spatial position. It solves the problem between translation-invariance in object detection. In this implementation, it uses less memory required in the computational process. The ROIs are extracted using RPN and are classified as background or object classes. In the last stage of R-FCN, there will be a position-sensitive ROI pooling layer. It will combine the outputs of the final convolutional layer and produce a score for each ROI.

### **2-5-3 Single Shot Multibox Detector**

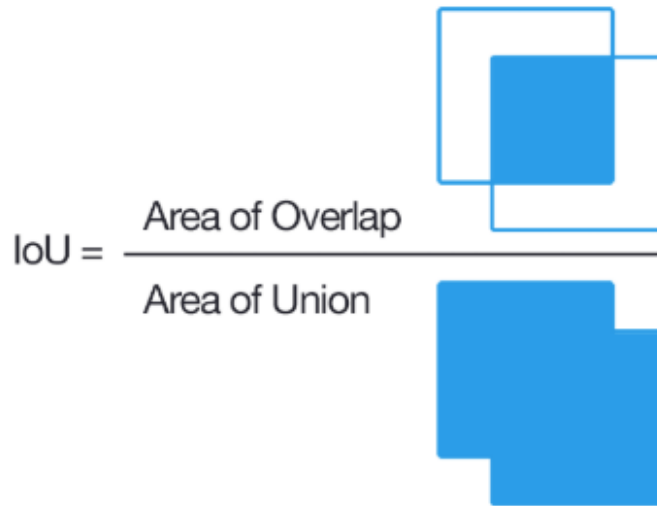
It is another approach that are similar to RPN of Faster-RCNN which separates the output region of the bounding boxes into a set of default boxes according to different aspect ratios and scales for each feature map position. The detector will merge prediction that will resulted from the feature maps in a different resolution. It uses additional convolutional layers that will have direct influence to its final position of the box and their confidence scores. However, size of the extra layers is very small and it can only hold a very limited information of objects which are smaller. Therefore, it will have very poor performance on small objects. It also saves computational time by skipping proposal generation and combine process into a single unified network.

### **2-5-4 Performance Evaluation Metrics for COCO dataset**

AP (Average precision) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. Average precision computes the average precision value for recall value over 0 to 1.

### **IoU ( Intersection over Union )**

To decide whether the prediction made is correct or not, IoU is used. A threshold is defined for IoU as the intersection between the predicted box and actual box divided by their union. It is considered as true positive if  $\text{IoU} > \text{threshold}$  and vice versa.



**Figure 2-9 show the graphical concept of IoU**

### **Precision and Recall**

Recall is actually the true positive rate, for example out of all actual positive, how many are true positive predictions made. While precision is the positive prediction value, for out of all the positive predictions, how many are true positive.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\# \text{ ground truths}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\# \text{ predictions}}$$

### mAP (mean Average Precision)

Calculation of AP ( average precision ) per class is required to calculate mAP. Considering the images containing ground truth box (green) and bbox predictions (red) for a particular class.

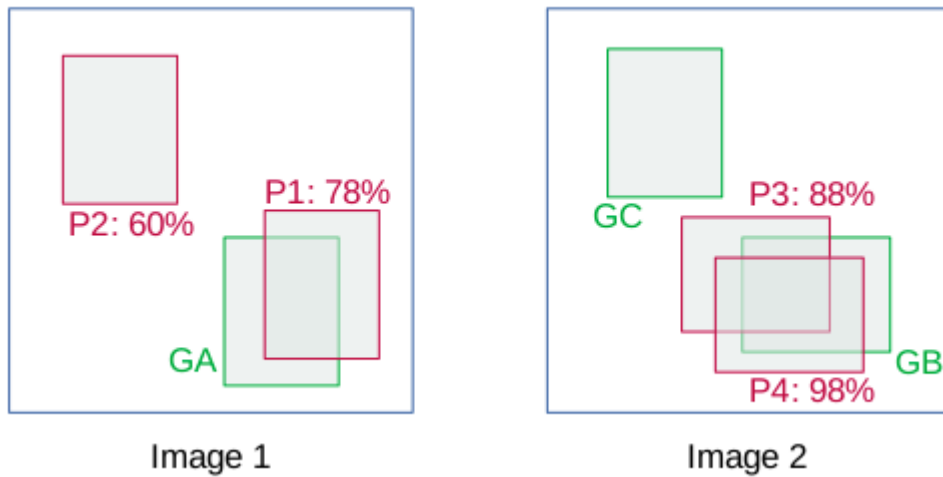


Figure 2-10 Graphical image for ground truth box and bbox prediction

The details of the bboxes are shown in the image below:

Image	Detection	Ground Truth	Confidence	IoU
Image 1	P1	GA	78%	> 0.5
Image 1	P2	-	60%	< 0.5
Image 2	P3	GB	88%	> 0.5
Image 2	P4	GB	98%	> 0.5
Image 2	-	GC	-	-

Figure 2-11 Computation value for the mAP calculation example

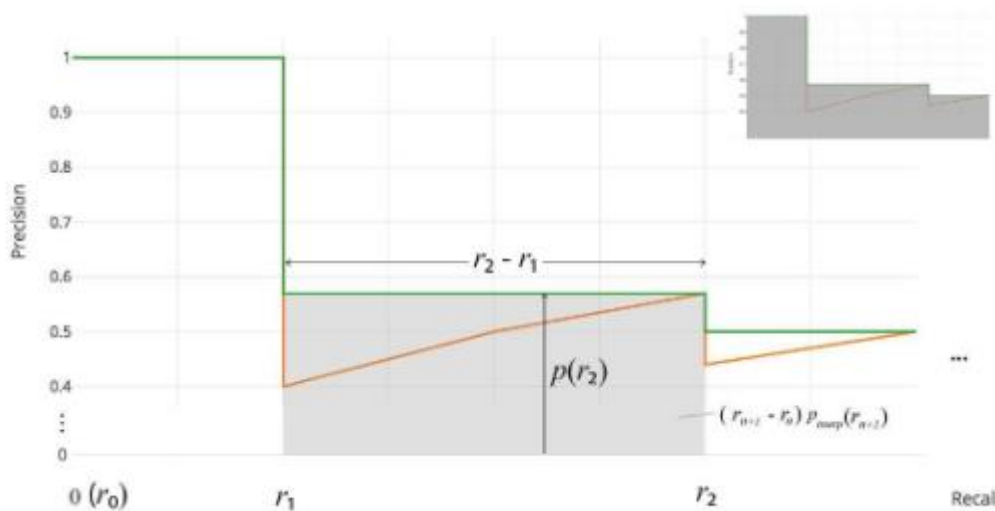
For this example, we set the threshold  $>0.5$ . Which means any prediction made is true positive (TP) if  $\text{IoU} > 0.5$ . If there are more than one detection for a single object, the detection with highest confidence is considered as TP, the rest will be assumed as FP.

Image	Detection	Confidence	IoU	Ground Truth	TP/FP	Acc TP	Acc FP	Precision	Recall
Image 2	P4	98%	> 0.5	GB	TP	1	0	1	0.33
Image 2	P3	88%	> 0.5	GB	FP	1	1	0.5	0.33
Image 1	P1	78%	> 0.5	GA	TP	2	1	0.67	0.67
Image 1	P2	60%	< 0.5	-	FP	2	2	0.5	0.67

**Figure 2-12 Computation value for mAP calculation example**

Calculate both precision and recall values. For example P4, Precision =  $1/(1+0) = 1$  and Recall =  $1/3 = 0.33$ . These values will be used to plot to get a precision-recall curve. The area under the curve would be Average Precision(AP).

In the latest approach to compute AP, it samples all the curve at all unique recall values ( $r_1, r_2, \dots$ ), whenever the maximum of precision values drop. We are measuring the exact value that falls under the precision-recall curve after the zigzags are removed.



**Figure 2-13 Precision-recall curve**

No approximation or interpolation is needed and instead of sampling only 11 points in the old approach, it samples  $p(r_i)$  whenever it drops and compute AP.

$$AP = \sum (r_{n+1} - r_n) p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

Now, we just use all AP per class, mAP is the averaged AP over all object categories.

As there were plenty of models to choose from, some models (such as SSD-mobileNet model) have architecture that allows fast detection but with less accuracy, while some model such as Faster-RCNN gives more accuracy but with slower detection. We decided to go for Faster-RCNN that provides more accuracy as it was important in our case. We need accurate detection for calorie counter application where the system have to detect multiple objects within an image input so that it is able to satisfy users' requirements and the objective of the application. Although models like SSD MobileNet can be implemented in smaller device such as mobile phone, in our case where users are required to input a static image and not live detection, Faster-RCNN is definitely more beneficial for such project.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
<a href="#">ssd_mobilenet_v1_coco</a>	30	21	Boxes
<a href="#">ssd_mobilenet_v1_0.75_depth_coco</a> ☆	26	18	Boxes
<a href="#">ssd_mobilenet_v1_quantized_coco</a> ☆	29	18	Boxes
<a href="#">ssd_mobilenet_v1_0.75_depth_quantized_coco</a> ☆	29	16	Boxes
<a href="#">ssd_mobilenet_v1_ppn_coco</a> ☆	26	20	Boxes
<a href="#">ssd_mobilenet_v1_fpn_coco</a> ☆	56	32	Boxes
<a href="#">ssd_resnet_50_fpn_coco</a> ☆	76	35	Boxes
<a href="#">ssd_mobilenet_v2_coco</a>	31	22	Boxes
<a href="#">ssd_mobilenet_v2_quantized_coco</a>	29	22	Boxes
<a href="#">ssdlite_mobilenet_v2_coco</a>	27	22	Boxes
<a href="#">ssd_inception_v2_coco</a>	42	24	Boxes
<a href="#">faster_rcnn_inception_v2_coco</a>	58	28	Boxes

**Figure 2-14 List of popular model for object-detection**

## **Chapter 3: System Design**

### **3.1 Design Process and Workflow**



For this project, there are several steps required to be done starting from scratch in order to train our object detection model for food ingredients that can be further integrated with calories counting. From data gathering to deploying and testing, every step is crucial and must be handled precisely to avoid any model failure and error.

#### **Setting up tensorflow and environment path**

The TensorFlow Object Detection API requires using the specific directory structure provided in its GitHub repository. We will need to create a virtual environment for our project via command prompt, installing Tensorflow into the environment. It also requires several additional Python packages, specific additions to the PATH and PYTHONPATH variables, and a few extra setup commands to get everything set up to run or train an object detection model.



## Datasets

Food images taken by camera devices, are widely used in variety of proposed system like classification, object-detection and also food recognition. Detection and classification of food ingredients through the images is a key process for calorie measurement systems used to treat chronicle diseases like diabetes and obesity. There are quite a few public datasets that are made ready to be retrieved for model training in the research community.

Dataset	Classes	Total Images	Source	Food-type
<b>ETHZ Food-101</b> [7]	101	101,000	foodspotting.com	Misc.
<b>UPMC Food-101</b> [26]	101	90,840	Web	Misc.
<b>Food50</b> [16]	50	5000	Web	Misc.
<b>Food85</b> [15]	85	8500	Web	Misc.
<b>CHO-Diabetes</b> [4]	6	5000	Web	Misc.
<b>Bettadapura et al.</b> [5]	75	4350	Web, smartphone	Misc.
<b>UEC256</b> [18]	256	at least <b>100</b> per class	Web	Japanese
<b>ChineseFoodNet</b> [10]	208	185,628	Web	Chinese
<b>NutriNet dataset</b> [22]	520	225,953	Web	Central European
<b>Food-251</b>	<b>251</b>	<b>158,846</b>	<b>Web</b>	<b>Misc.</b>

**Figure 3-1: Public datasets**

For example, the Food-101 dataset, the author have collected 101 categorize of different type of fast food like burgers, salad, pizza and etc. However, the collection only contain fast food images that are usually complex and having a lot of ingredients occluded in the images. It introduces 101 food categories, with 101,000 images which are mostly mixed food. UEC-256 consist of 256 categories of food images comes with bounding box indicating the location of the label. However, the dataset contain mostly Japanese cuisine only. While ChineseFoodNet consist of Chinese food items only. The available dataset on the research community mostly have the feature of:

1. Dataset contain both mixed and non-mixed food
2. Food images are labelled in a general way
3. Some images are cleaned and contain no noise (background)

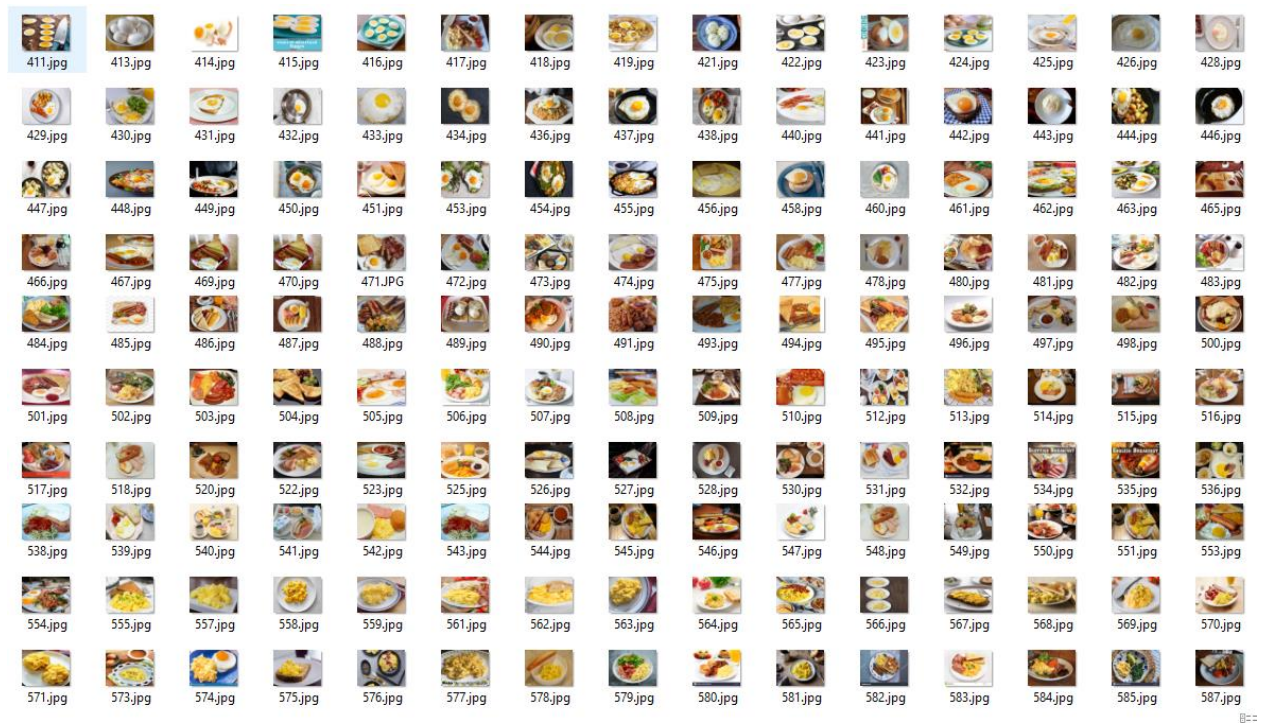
However, in our specific calorie counting project, we needed a custom dataset that contain food images that comes in non-mixed food portion. Since our approach in this project is to integrate the counting process as well as segmenting the food portion into smaller size so that our algorithm can allow the calorie counting to estimate the volume of the food in order to provide a much more accurate result, the images should come in different portion of food ingredients as well as in different light intensity and background so that it allows us to train the model to detect with better accuracy and efficiently. The existing dataset oppose to the interest of our project since many of the dataset contain mixed food. Mixed food is complex in a way that many ingredients are occluded by each other, this will lead to inaccurate detection since it is not visible. Calorie counting for mixed food should uses different volume estimation approach like determining the scale and area of the food and calculate the values using general portion adjusted by the serving size detected by the algorithm. But in our case, our approach are used to detect mainly non-mixed food which are mostly visible and not occluded in majority parts. Since we are segmenting the food portion ranged from small to large serving, we differentiate it by the label of the class. One kind of food ingredient should be segmented into different serving portion so it when the system detects, it will retrieve the calories information according to the class. Since many of the public dataset comes with pre-labelled images that are not appropriate for this project, I decided to create my own custom dataset which are suitable to be used in our approach by taking subset of the public dataset which meets the requirement and re-labelled and also sources from image platform like Google Image as well as taking own data samples using smartphone. The custom dataset included total of 888 images which separated into ratio of 8:2 into train and test data respectively. The dataset contains single and multiple labels with their respective classes. The main feature of our dataset are:

1. Images with different light intensity and background
2. Non-mixed food type (single and multiple quantity)
3. Labelled with customized class for the model
4. Multiple angle for the same food item

## **Gathering data**

In order to train a good model classifier, Tensorflow requires more than hundreds or even thousand of images of certain object to ensure the end result is robust. It should include any random objects or background that comes with the object that you required to train. The background should be available in multiple kind of intensity and light condition. The object should come with different kind of angle as well. There should also be some images then come with object that is partially blocked, overlapping with something or just halfway in the picture, but not majority occluded.

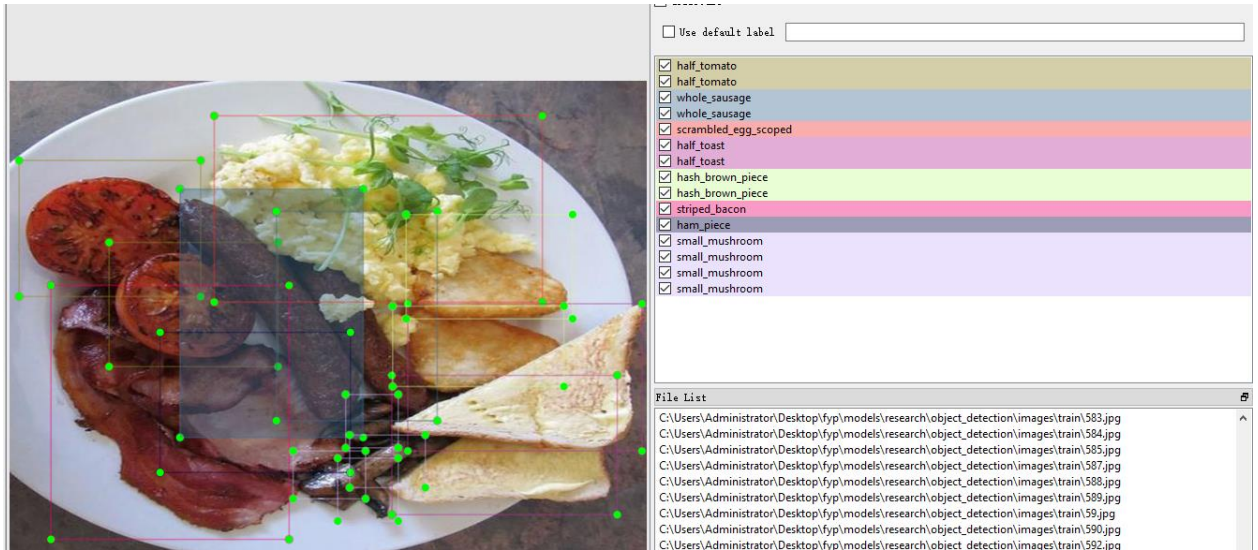
As for this classifier, I am planning to train 17 classes that are mostly ingredients that available in English breakfast, due to the reason with time constraint and having minimum hardware requirement as we cannot train too many different food classes on the given time period, the main objective is to test how well and accurate the algorithm can calculate the calories values of the plated meal. As English breakfast is part of the non-mixed food that are mostly countable and allows easier volume estimation which is suitable with our approach. Using further segmentation into appropriate portion serving units so that these classifier could be used in calorie counting and are able to determine its portion precisely. However, not all of the ingredients are segmented by quantity unit but only those ingredients that easily countable. As for ingredient that are not easily countable such as baked beans and scrambled egg, they are segmented into portion like one or two scope to estimate their appropriate calories values. The sample images were taken from subset of public dataset, Google Image and taken by own smartphone. The images were selected carefully with the requirement that the photos should be in clear condition, partly occluded with other objects, or comes with random non-desired objects in a different light intensity. There are total images of 888 used for the data training. Each of the picture is resized to 800x600 resolution in order to ensure their size is consistent and reducing the training time. Higher resolution images might result in difficult and more time consumption in training the data. Approximately 80% of the images goes to the train file that are used for the training process while the rest of the 20% will be transferred to the test file which will be used to validate the results.



**Figure 3-2: Sample data for training**

## Labelling

With all the images being collected, we will have to label each of the desired object in the sample images. We will be using the graphic annotation tool LabelImg for labelling purpose. It is a handy tool that is simple to use, we just have to create a rectangle reactbox to cover the part of our desired object. If there is multiple desired object in one image, labelling should include most of the objects in the image. Repeat the steps for every images in the train folder and test folder. It will save a .xml file the contains labelling data for every images, it will be used later to generate a .record file for both train and test folder which will act as the inputs for the training process, the file contains all of the coordinates information of the image features in the labelled images.



**Figure 3-3: Labelling samples**

### Generating TFrecords

Since the labelling data is generated in .xml file. We have to convert the format into .csv file so it could be used to generate into .record file. We will be using the `xml_to_csv.py` and `generate.tfrecord.py` scripts from Dat Tran's Racoon Detector dataset with some modification done to work with our directory structure. We will run some code in python to convert the generated .xml file into .csv format. Now we will have `label.csv` and `test.csv` files. The excel file should contains the class and labelling data.

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	1.jpg	800	600	full_tomato	75	28	737	568
3	10.jpg	800	600	full_tomato	72	52	737	547
4	11.jpg	800	600	full_tomato	73	6	733	599
5	111.jpg	800	600	full_tomato	152	1	399	185
6	111.jpg	800	600	full_tomato	165	152	441	361
7	111.jpg	800	600	full_tomato	146	313	430	547
8	111.jpg	800	600	full_tomato	359	109	630	319
9	111.jpg	800	600	full_tomato	353	284	638	482
10	111.jpg	800	600	full_tomato	412	447	666	600
11	112.jpg	800	600	full_tomato	91	262	385	556
12	112.jpg	800	600	full_tomato	243	50	523	356
13	112.jpg	800	600	full_tomato	422	249	696	570
14	113.jpg	800	600	tomato_slice	151	191	500	600
15	113.jpg	800	600	tomato_slice	83	69	420	421
16	113.jpg	800	600	tomato_slice	51	4	363	226

**Figure 3-4: Labelled data in .csv format**

Modifications were done to the `generate_tfrecord.py` file to ensure it contains our number of classes and their class names. The return values are important because they let the system know what values to return when certain classes are detected, the values will let the system know what the values belong to so it could output the result according to the IDs in `labelmap.pbtxt` file.

```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'full_tomato':
        return 1
    elif row_label == 'half_tomato':
        return 2
    elif row_label == 'tomato_slice':
        return 3
    elif row_label == 'half_boiled_egg':
        return 4
    elif row_label == 'whole_boiled_egg':
        return 5
    elif row_label == 'whole_sausage':
        return 6
    elif row_label == 'whole_toast':
        return 7
    elif row_label == 'scrambled_egg_scoped':
        return 8
    elif row_label == 'ham_piece':
        return 9
    elif row_label == 'baked_beans_scoped':
        return 10
    elif row_label == 'small_mushroom':
        return 11
    elif row_label == 'whole_fried_egg':
        return 12
    elif row_label == 'striped_bacon':
        return 13
    elif row_label == 'half_toast':
        return 14
    elif row_label == 'hash_brown_piece':
        return 15
    elif row_label == 'cocktail_sausage':
        return 16
    elif row_label == 'large_mushroom':
        return 17
    else:
        return None
```

**Figure 3-5: Class Name for Label Map**

Then, using our previous `.csv` labelling file, we can start generating a `train.record` and `test.record` file which will be later used to train the new object detection classifier.

## Configuring Training

We have to create a label map and modify part of the code in the configuration file. The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. We will be using Visual Studio Code to build a label map. It is important to include all classes that are within the training in this label file so that it can reference to these labels to output the result.

```
item {
  id: 1
  name: 'full_tomato'
}
item {
  id: 2
  name: 'half_tomato'
}
item {
  id: 3
  name: 'tomato_slice'
}
item {
  id: 4
  name: 'half_boiled_egg'
}
item {
  id: 5
  name: 'whole_boiled_egg'
}
item {
  id: 6
  name: 'whole_sausage'
}
item {
  id: 7
  name: 'whole_toast'
}
item {
  id: 8
  name: 'scrambled_egg_scoped'
}
item {
  id: 9
  name: 'ham_piece'
}
item {
  id: 10
  name: 'baked_beans_scoped'
}
item {
  id: 11
  name: 'small_mushroom'
}
item {
  id: 12
  name: 'whole_fried_egg'
}
item {
  id: 13
  name: 'striped_bacon'
}
item {
  id: 14
  name: 'half_toast'
}
item {
  id: 15
  name: 'hash_brown_piece'
}
item {
  id: 16
  name: 'cocktail_sausage'
}
item {
  id: 17
  name: 'large_mushroom'
}
```

**Figure 3-6: Class ID and Name for Training File**

Next, we will download the model which is trained on the COCO dataset. COCO stands for Common Objects in Context, this dataset contains around 330K labeled images. The model selection is important as we need to make an important tradeoff between speed and

accuracy. Depending on the system requirement that is used to train the model, the correct model must be selected. In our case, we will be using `faster_rcnn_inception_v2_coco` dataset. Modification must be made to the configuration file in order to train our model, first we change the total number of classes required for our project, then set the paths for `fine_tune_checkpoint`, `input_path` as well as `label_path`.

```
model {
  faster_rcnn {
    num_classes: 17
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
  }
}
```

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "C:/Users/Administrator/Desktop/fyp/models/research/object_detection/train.record"
  }
  label_map_path: "C:/Users/Administrator/Desktop/fyp/models/research/object_detection/training/labelmap.pbtxt"
}
```

```
eval_input_reader: {
  tf_record_input_reader {
    input_path: "C:/Users/Administrator/Desktop/fyp/models/research/object_detection/test.record"
  }
  label_map_path: "C:/Users/Administrator/Desktop/fyp/models/research/object_detection/training/labelmap.pbtxt"
  shuffle: false
  num_readers: 1
}
```

**Figure 3-7: Class Number and Input Path Labels**



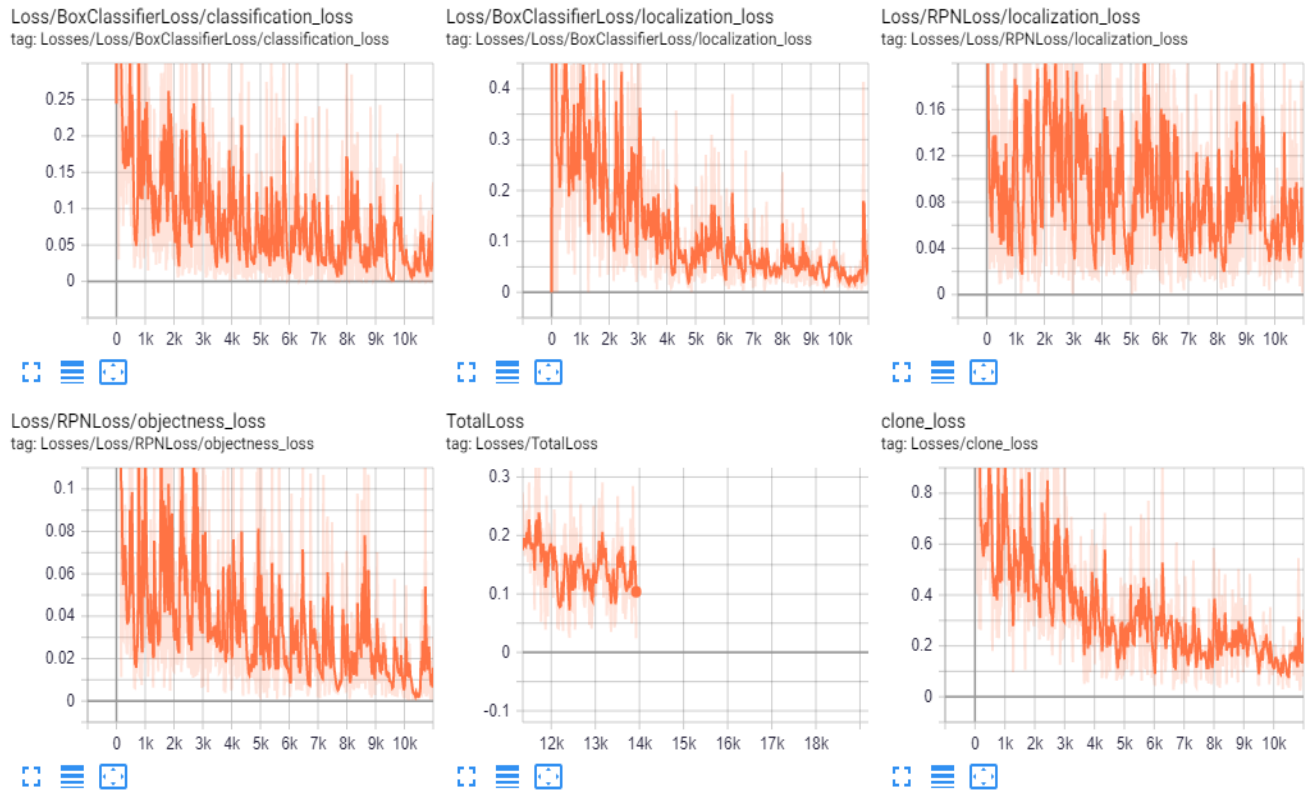
## Training the model

After all the steps done, we can begin training data using the train.py file available in the API. In our case, we use the config file for Faster-RCNN model specifically since it will be our fine-tuned model for this training process.

```
01:00.0, compute capability: 6.1)
INFO:tensorflow:Restoring parameters from C:/tensorflow1/models/research/object_detection/faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt
INFO:tensorflow:Starting Session.
INFO:tensorflow:Saving checkpoint to path training/model.ckpt
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 1: loss = 2.6708 (5.383 sec/step)
INFO:tensorflow:global step 2: loss = 3.0352 (0.251 sec/step)
INFO:tensorflow:global step 3: loss = 3.4884 (0.204 sec/step)
INFO:tensorflow:global step 4: loss = 2.9733 (0.193 sec/step)
INFO:tensorflow:global step 5: loss = 2.2184 (0.191 sec/step)
INFO:tensorflow:global step 6: loss = 2.0321 (0.554 sec/step)
INFO:tensorflow:global step 7: loss = 2.0424 (0.211 sec/step)
INFO:tensorflow:global step 8: loss = 2.0252 (0.208 sec/step)
INFO:tensorflow:global step 9: loss = 2.0053 (0.194 sec/step)
INFO:tensorflow:global step 10: loss = 1.3622 (0.193 sec/step)
INFO:tensorflow:global step 11: loss = 1.8027 (0.197 sec/step)
INFO:tensorflow:global step 12: loss = 1.2485 (0.196 sec/step)
INFO:tensorflow:global step 13: loss = 1.0712 (0.193 sec/step)
INFO:tensorflow:global step 14: loss = 1.6604 (0.189 sec/step)
INFO:tensorflow:global step 15: loss = 1.2657 (0.192 sec/step)
INFO:tensorflow:global step 16: loss = 1.4351 (0.193 sec/step)
INFO:tensorflow:global step 17: loss = 1.2152 (0.192 sec/step)
INFO:tensorflow:global step 18: loss = 1.1165 (0.197 sec/step)
INFO:tensorflow:global step 19: loss = 1.6557 (0.192 sec/step)
INFO:tensorflow:global step 20: loss = 1.7777 (0.200 sec/step)
```

**Figure 3-8: Training process**

The prompt should show each step done along with their loss values. The loss values should begin from the highest then slowly drop to lower values as it progresses. The model is recommended to be trained until the loss values have dropped to a consistency of 0.05 or below. The data is recorded every 5 mins and will be logged to Tensorboard. The loss function indicates how far the predicted values are deviated from the actual values in the training data. We change the model weights to make the loss minimum and that is what training is all about.



**Figure 3-9: Tensorboard showing graph of value losses during training**

### Exporting Inference Graph

After the training process, we will be using the export python file to generate a frozen inference graph (.pb file) for the latest model file that we have trained. This file will contain the object detection classifier.



## Adding calories value database

Since we need to calculate the calories of each of the food type, we need to have a database ready to store all the values needed required for the calculation. Database will contain all the calories information and ready to be retrieved by the API when running the inference, each of the objects detected will trigger the API to retrieve its calories information and sum up together after the counting process. In our case, we are using the USDA National Nutrient Database for standard reference in building our database because it is a gold standard that many related field application has been using them to build nutritional databases, as we have very customized and specific terms and portion segmentation for our model, we will not be using other existing nutrition database to be merged with our model.

```
dict1 = {'full_tomato':24,'half_tomato':12,'tomato_slice':3,'half_boiled_egg':39, 'whole_boiled_egg':78,'whole_sausage':283,'whole_toast':50,  
        'scrambled_egg_scoped':91,'ham_piece':60, 'baked_beans_scoped':119,'small_mushroom':2,'whole_fried_egg':90,'striped_bacon':43,  
        'half_toast':25,'hash_brown_piece':160,'cocktail_sausage':30,'large_mushroom':25}
```

**Figure 3-10 Nutritional information for each classes in the model**

The counting and retrieving nutritional information of the food detected will be performed all at once after running the inference. The nutritional information will be retrieved based on the classes detected and sum up all the values based on the counting and the inference will print the values to the users.

```

# Draw all boxes onto image.
box_count = 0
for box, color in box_to_color_map.items():
    box_count = box_count + 1;
    ymin, xmin, ymax, xmax = box
    if instance_masks is not None:
        draw_mask_on_image_array(
            image,
            box_to_instance_masks_map[box],
            color=color
        )
    if instance_boundaries is not None:
        draw_mask_on_image_array(
            image,
            box_to_instance_boundaries_map[box],
            color='red',
            alpha=1.0
        )
    draw_bounding_box_on_image_array(
        image,
        ymin,
        xmin,
        ymax,
        xmax,
        color=color,
        thickness=line_thickness,
        display_str_list=box_to_display_str_map[box],
        use_normalized_coordinates=use_normalized_coordinates)
    if keypoints is not None:
        draw_keypoints_on_image_array(
            image,
            box_to_keypoints_map[box],
            color=color,
            radius=line_thickness / 2,
            use_normalized_coordinates=use_normalized_coordinates)

print(dict)
getCalorie(dict)
return image

```

**Figure 3-11 Code to generate boxes for inference detection and count**

Running the inference will draw all the boxes on the object detected as well as displaying their confidence level and classes name. Inference will print the classes name, their calories, the number of count as well as displaying the image that has been used to run the inference.

## Model Testing

Using Jupyter Notebook, we can run the application for testing. Modify the cell with appropriate number of classes and set the path to the generated inference graph. Then, set the correct paths for the training and label map. Then the application is already to run with test images for classification. For testing, each set of model trained is evaluated and test the accuracy and preciseness of the detection, if the model is not robust then the more dataset is added into it and retrained for better result. This steps is repeated until we are able to obtain a satisfying result.

```
In [4]: # What model to download.
MODEL_NAME = 'inference_graph'

# Path to frozen detection graph. This is the actual model that is used for the object detection.
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('training', 'labelmap.pbtxt')

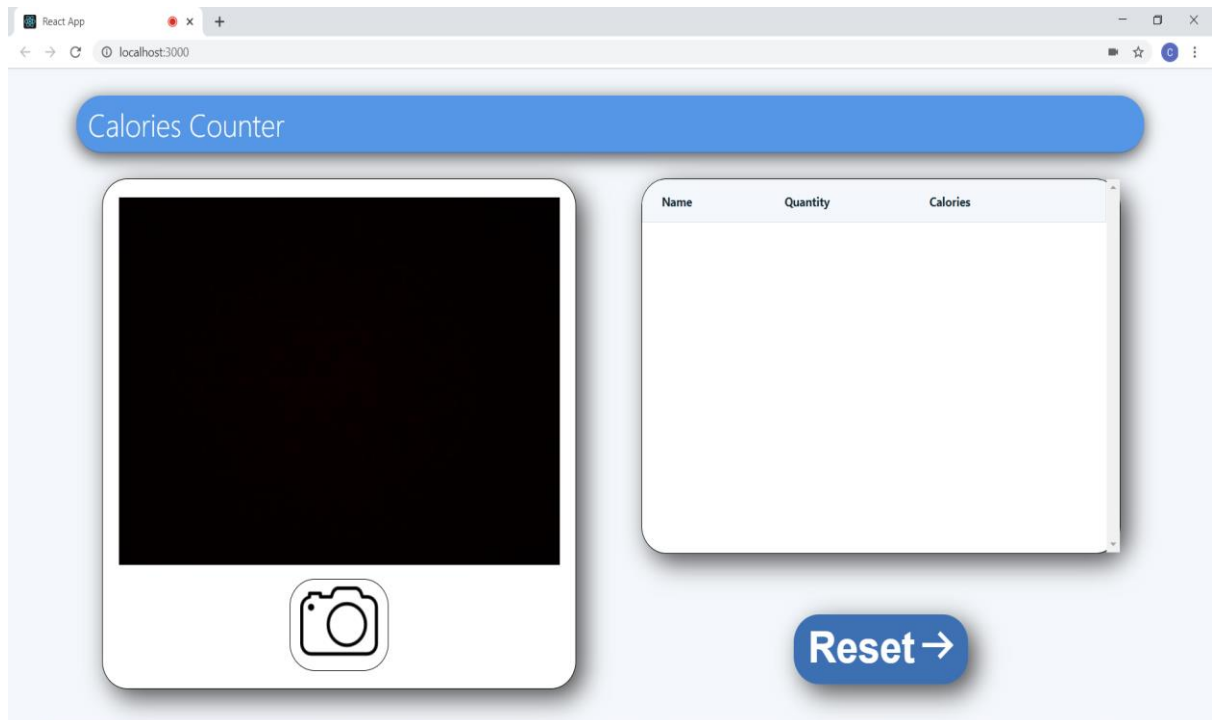
NUM_CLASSES = 17
```

**Figure 3-12: Modification of path in Jupyter Notebook**

## Web-based application development

An simple interface for calorie counting is built in order to ease the process to run an inference for user. In this application, we will be featuring a live capture using any camera devices that allows users to live capture photos of their meal and directly input it to run the inference detection. We will be using react.js, an open-source Javascript library that provide fast and simple process to build the interface, it is suitable for such single paged application because it allows users to easily change the data without having to reload the page. In our case, we need our interface to have a webcam container and a table to display mainly our labels, quantity in the detection as well as the total calories for each classes. Then , we will have one reset button to allow user to reset the output for the detection.

Basically, our interface looks like this:



**Figure 3-13 Interface design for the detection**

### Deploying Model into Web App

In this application, we will need to have frontend for display. Then, we requires the backend system be ready to run the detection inference when user input data in the interface. The trained model is required to put into the backend directory of the folder. Using python script, we can start the backend system easily where we specified the model directory. In our case, we will be running the backend and frontend system at two different port which are port:5000 and port:3000. When user live capture an image input, it uses POST to send data to the backend system to process, where the inference will run the input and output result back to the frontend. If user capture subsequent images without resetting the output, the system will continue to append result into the table and updated automatically.

We have to set a path to integrate the system to the inference graph.

```
dir_path = os.path.dirname(os.path.realpath(__file__))  
MODEL_DETECT_PATH = dir_path + '/model/frozen_inference_graph'
```

Setting the back-end system to retrieve the image captured to load in as input to run the inference graph. Threshold is set at confidence = 0.80, which means only object that are detected more than 80% confidence will be considered as true positive for the prediction, else ignored. The system will store the values for each item in an array which later to be displayed in the interface table.

```

# Load in an image to object detect and preprocess it
img_data = getI420FromBase64(request.data)
x_input = np.expand_dims(img_data, axis=0) # add an extra dimension.

# Setting initial detection time, so execution time can be calculated.
t_det = time.time()

# Get the predictions (output of the softmax) for this image
tf_results_det = sess_det.run([output_tensor_det,detection_boxes,detection_scores,detection_num], {input_tensor_det : x_input})

dt_det = time.time() - t_det
app.logger.info("Execution time: %0.2f" % (dt_det * 1000.))

# Different results arrays
predictions_det = tf_results_det[0]
prediction_scores_det=tf_results_det[2]
prediction_boxes_det=tf_results_det[1]
prediction_num_det=tf_results_det[3]

# print("-----")
# print(predictions_det)
# print(prediction_scores_det)
# print("-----")

threshold = 0.80

num=int(prediction_num_det)
predict_list=predictions_det[0].astype(int).tolist()
scores = prediction_scores_det[0]
label=[]

for i in range(num):
    new_item = {}
    if scores[i] > threshold:
        prediction_label = str(predict_list[i])
        obj_name = menu['item'][prediction_label]
        obj_price = menu['value'][prediction_label]

        new_item = {'id': randint(0, 100000),
                    'name': obj_name,
                    'quantity': 1,
                    'value': obj_price}

        label.append(new_item)

print("number and list of items that above the threshold")
print(len(label))
print(label)
return jsonify(label)

```

**Figure 3-14 Threshold, detection and value output for web-based interface**

### 3-2 Timeline

The gantt chart shows the overall timeline that is required to finish the project in time from start to the end of it. It is divided into different phases with each being the pre-requisites to the next activity.

PROJECT NAME	PROJECT DURATION	PROJECT START DATE	PROJECT END DATE																		
FYP Project II	15 Weeks	13-Jan-20	17-Apr-20	TASK ID	TASK DESCRIPTION	TASK DURATION	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15
1 Overall Progress																					
Development																					
Meet up with supervisor																					
2 Research																					
Model evaluation																					
Database information retrieval																					
Training Model Evaluation																					
Development Approach																					
3 Data Collecting																					
Calories data																					
Training images collection																					
Test images collection																					
4 Development																					
Building nutritional database																					
Image labelling																					
Model training																					
Model testing																					
Interface development																					
Interface testing																					
5 Documentation																					
Report writing																					
6 Presentation and Demostration																					
Planning and preparation																					
Presentation																					

Figure 3-15: Gantt Chart

### 3-3 Implementation Issues and Challenges

Since this project involves image processing that requires high processing power to perform, it usually requires high-end hardware like high processing power of graphic cards, RAM and processor. In our case, we are using Nvidia Geforce GTX750-Ti, 8 gigabytes of RAM and an i5-4440 CPU processor running in Windows 8.1 and 64-bit operating system, which barely meet the requirement of a typical image processing device. This eventually slow down our overall training progress compared to other high-end device. This causes the overall training process to be very time consuming. Moreover, installation of Tensorflow could be difficult as it requires some specific version of software to be used like CUDA 10.0 and CUDNN. During the process, there could be many errors that occur that required to be solved. There are also issue on choosing the suitable approach and model to use for accurate calories counting, there should have different approaches on detecting



a more complex and occluded meals but in our case, we will be using a more simple approach in counting a less complex and occluded meal like English breakfast that are countable for accuracy purpose, to calculate complex meal that are hardly countable needed to use another approach like volume estimation for accuracy purposes.

### 3-4 System Workflow

The project flow shall consist of two part mainly divided to training process and running inference at the web-based application.

The training process:

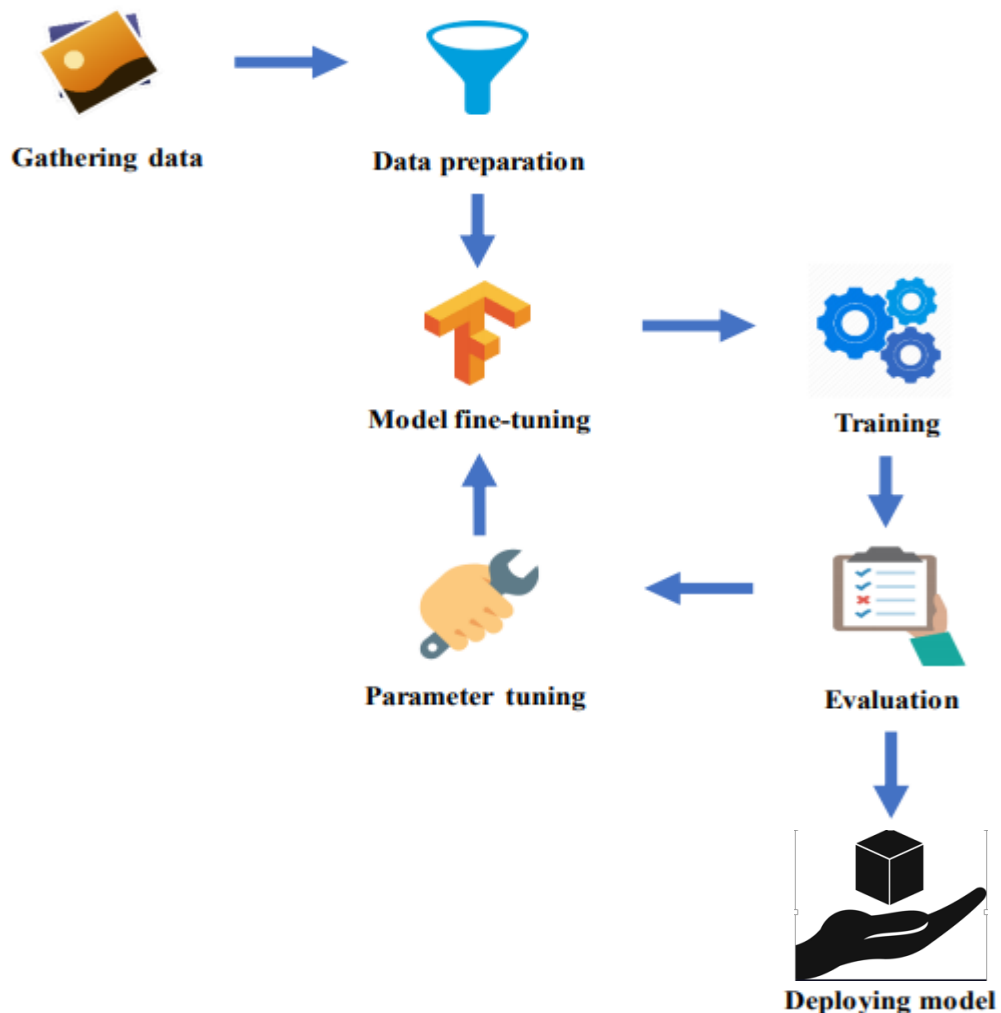


Figure 3-16 Workflow of training process

The web application process:

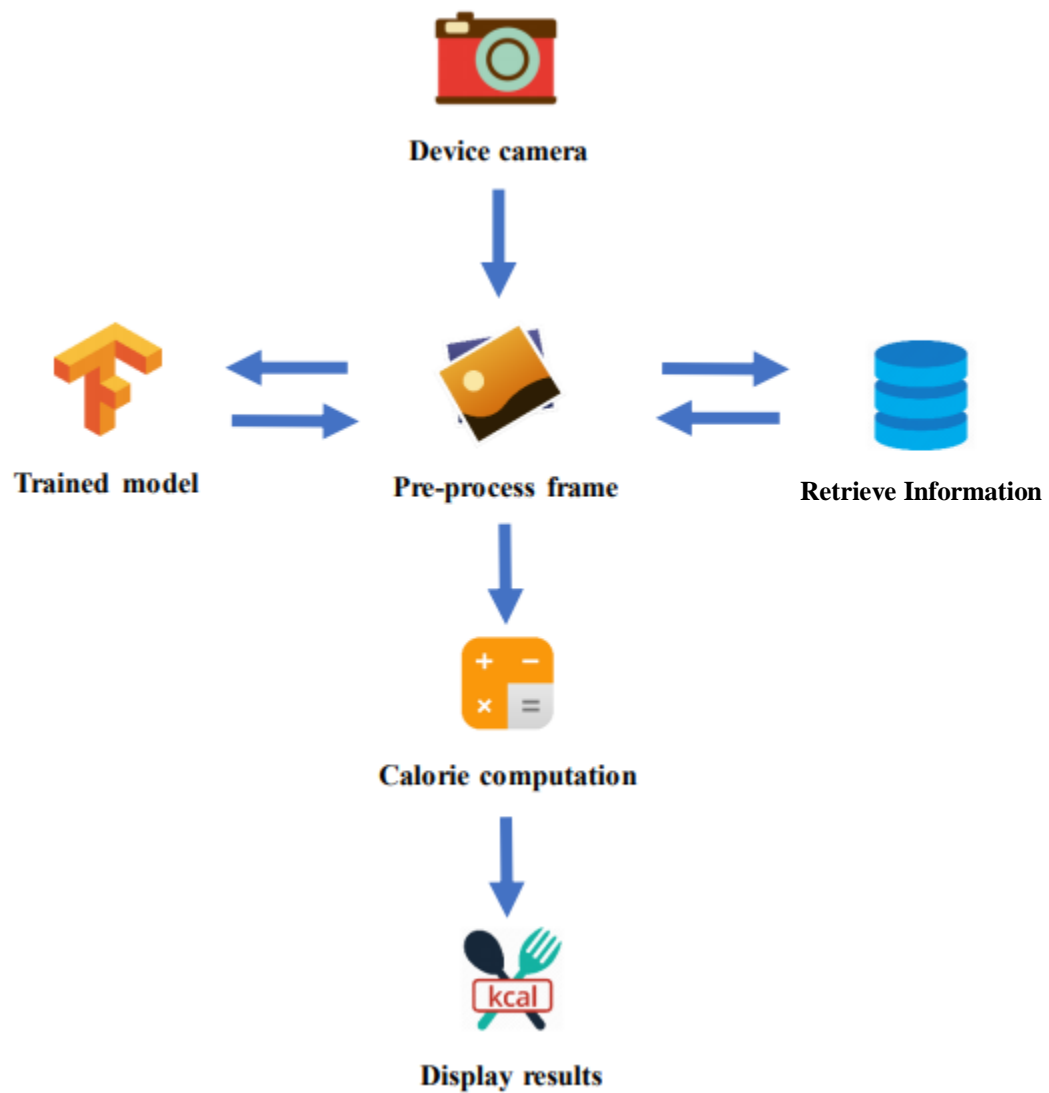


Figure 3-17 Workflow for web application

## **Chapter 4: Discussion**

### **4-1 Technologies Involved**

#### **TENSORFLOW**

It is an open source software library that is developed by the Google team used for dataflow and various programming across a range of tasks. It is a symbolic math library that are mostly used for machine learning projects like neural networks. Tensorflow ease the process of the training as it could provides various library functions that are needed. It requires CPU and GPU power in order the run the training task and it is also able to run inference graph required for the model detection.

#### **CUDA and cuDNN**

Cuda is the language/API for programming on the graphic card. cuDNN is the library for deep neural nets built using CUDA. It is a GPU-accelerated library of primitives for deep neural networks. It provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization and activation layers. Since our project requires to run on GPU, it is essential to be used in this project to train models.

#### **ANACONDA**

Anaconda is an open source software that is mainly focus on Python and R languages to be used in scientific computing, predictive analysis and machine learning applications. It helps users to simplify the management of the package and the deployment processes. It is highly popular and have been widely used, it comes with more than 1500 packages and a virtual environment manager. It includes a Anaconda Navigator which is a desktop graphical user interface that allows users to launch applications and manage the package. We will be using the Anaconda prompt to execute the python code to setup, train and access to applications like Jupyter Notebook for the purpose of running the object detection and obtaining the results.

## **VISUAL STUDIO CODE**

It is a source-code editor that is introduced by Microsoft. The applications mainly serve the purpose of writing code and debugging. It allows user to customize their preferences and install available extensions for extra functionality. It can be used with wide variety of different languages. In this case, we will need Visual Studio Code to create and write file for our labelling, as well as editing the configuration file for the training process. Also, we will be developing a script for both front-end and back-end system to enable it to run the inference for object-detection with an interface, display all the necessary information to the end-users.

## **LABELIMG**

This application serve the purpose as a graphical image annotation tool and is able to label images with object bounding boxes. It allows users to label each of their object in an image then assign a class variable to them. We will be needing this tool to label each of our sample data, thus classifying them. The end product will deliver a xml file that contains the information of the labels.

## **REACT**

The application allow users to create an interactive UIs for webpages with supported library that ease the development process. React has a high performance and it is able to update when render the webpages component for each data changed. This allows us to develop more easily since developer do not have to re-run the application when updating the code. Also, since we are doing live capture detection, any input and output data occur will make changes directly to the webpage, allowing fast performance for the detection and results.

## **4-2 Methodology**

### **XP Programming**

To build up the system , XP Programming is used as the methodology. It is part of the agile methodology and it emphasize business results first and takes an incremental “to get something started” approach in building the product using continuous testing and improvement (Maniuk,2016). It advocates frequent releases in short development cycles and is intended to

improve productivity and introduces checkpoints at which new customer requirements can be adopted.

### Prototyping methodology

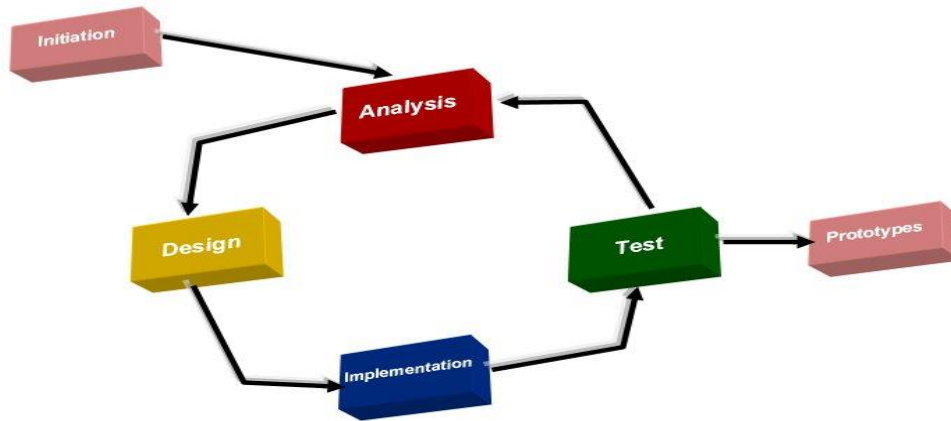


Figure 4-1 Prototype methodology

#### Planning Phase

The project is planned to take place for total of 2 semester which is approximately 9-10 months time to complete the whole system.

#### Analysis Phase

All problem statement and objective all stated down. The feasibility of the project and the area of scope is determined before developing the project. This is to make sure project could be completed and is within the time constant. Also, analysing the model and approaches to be used in the project.

#### Design Phase

Designing on how to build the model and the system, using approaches and methods that we analysed earlier to determine how we want the system to look like and its functionality. Also, determining the structure of the model, database and also interface.

#### Implementation Phase

The model is being trained to be ready to use and web application interface is built. The back-end and front-end system have to be ready to be deployed for testing.

#### Prototyping

The system is being prototyped and testing is carried out to determine its performance and evaluating the model. If the result is not satisfying, it will jump back to previous phase for necessary modification and repeat the testing until a satisfied result is being produced.

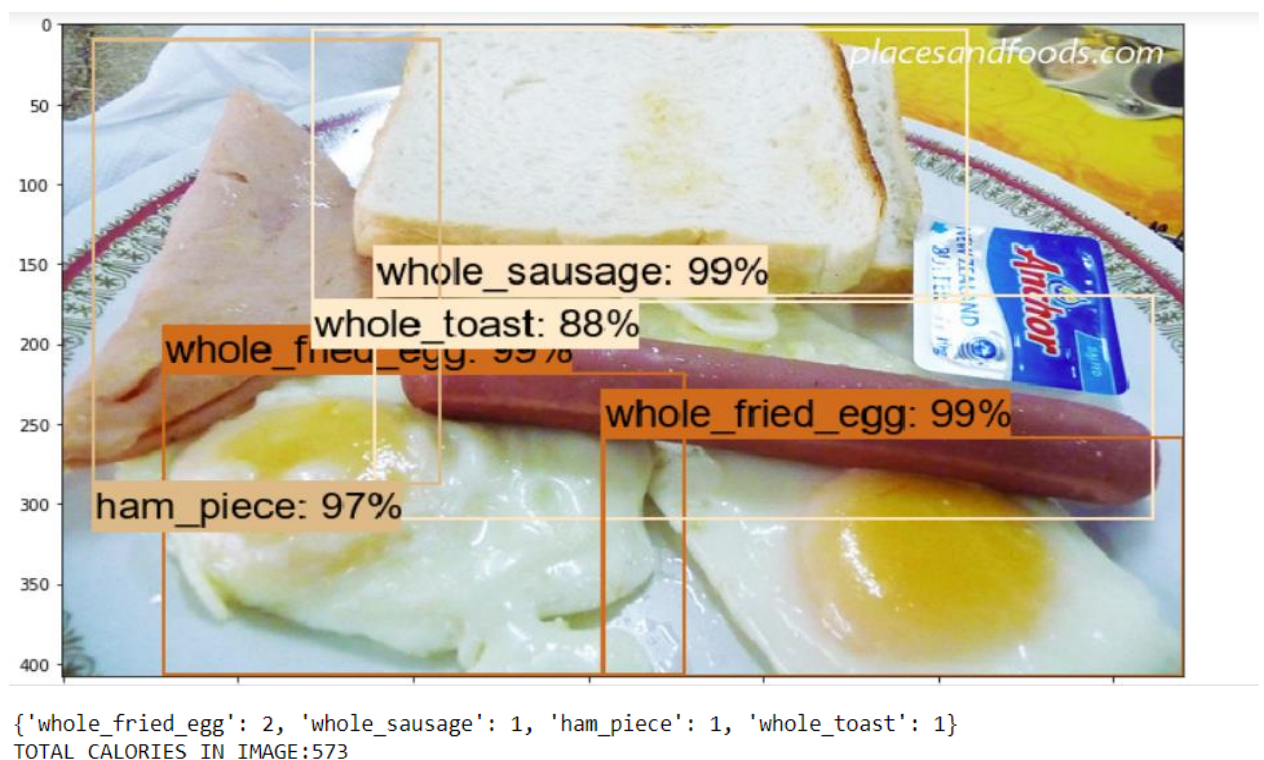
## Chapter 5: Implementation and Testing

### 5-1 Results

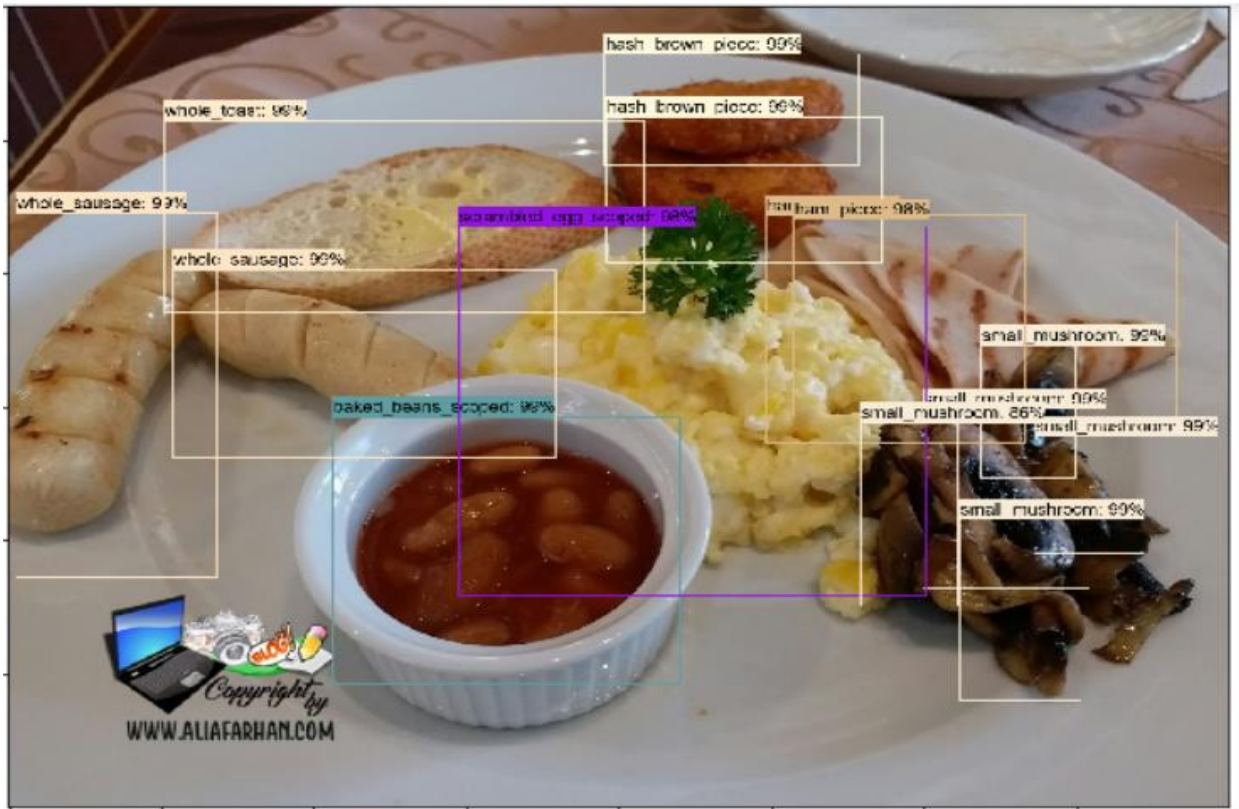
Some sample images are taken from online source to test the detection of the inference trained, the images taken mostly consist of all the food components that are mostly classified for the detection model. There are a few hypothesis made for which the model might fail to make prediction of:

- 1) The majority part of the object are occluded.
- 2) The object are not clean plated and have a lot of complexity for the model to recognize the objects.
- 3) Picture resolution is low and object is very blur to be detected.

Below are the test run samples for the detection (running inference on pre-captured image) :

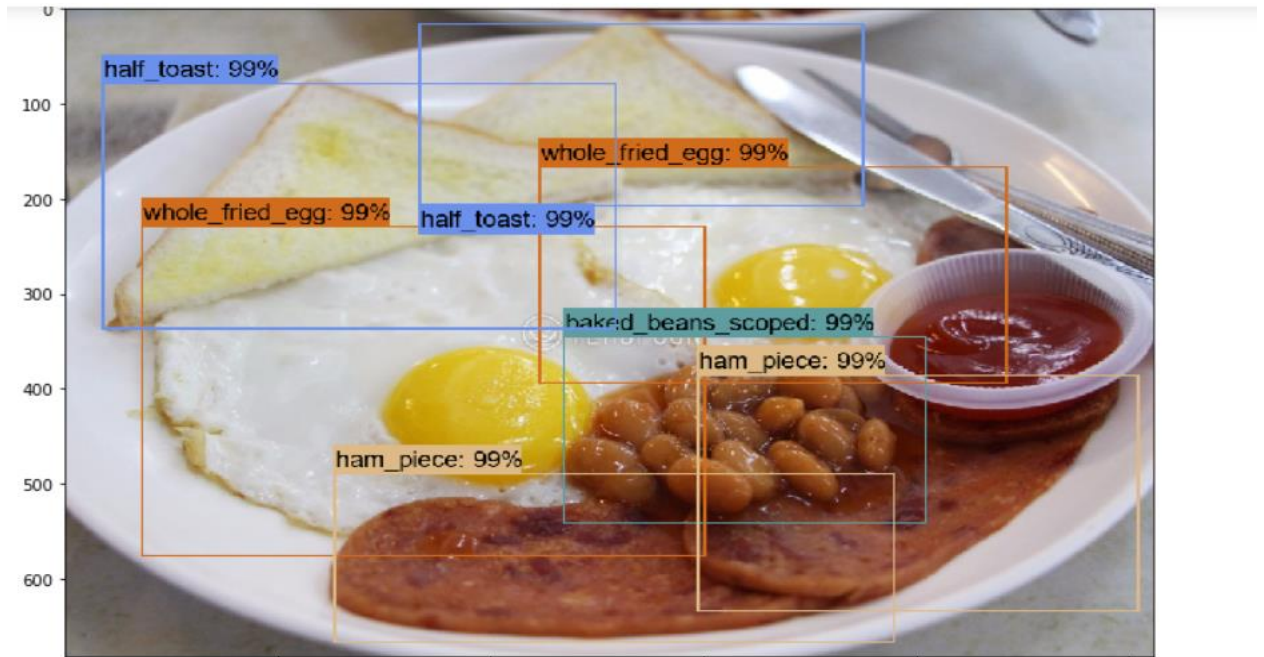


**Figure 5-1: Result sample 1**



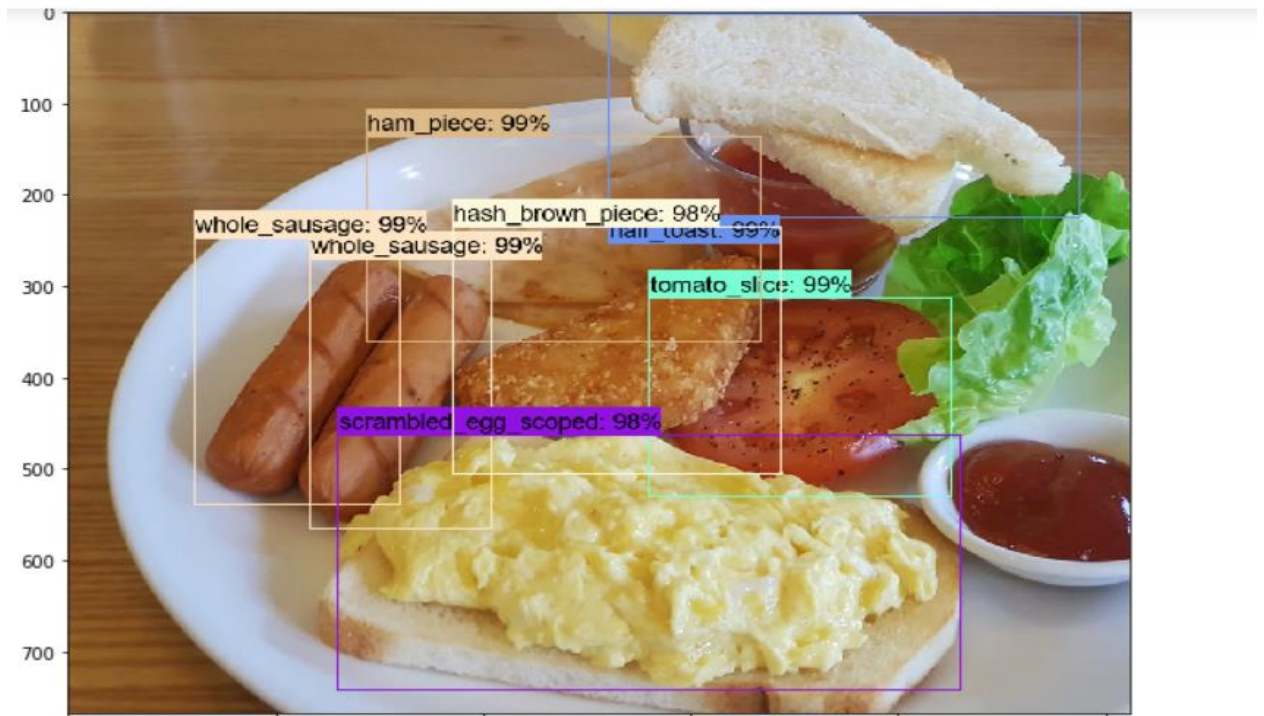
```
{'baked_beans_scoped': 1, 'hash_brown_piece': 2, 'small_mushroom': 5, 'ham_piece': 2, 'whole_sausage': 2, 'whole_toast': 1, 'scrambled_egg_scoped': 1}
TOTAL CALORIES IN IMAGE:1276
```

**Figure 5-2: Result sample 2**



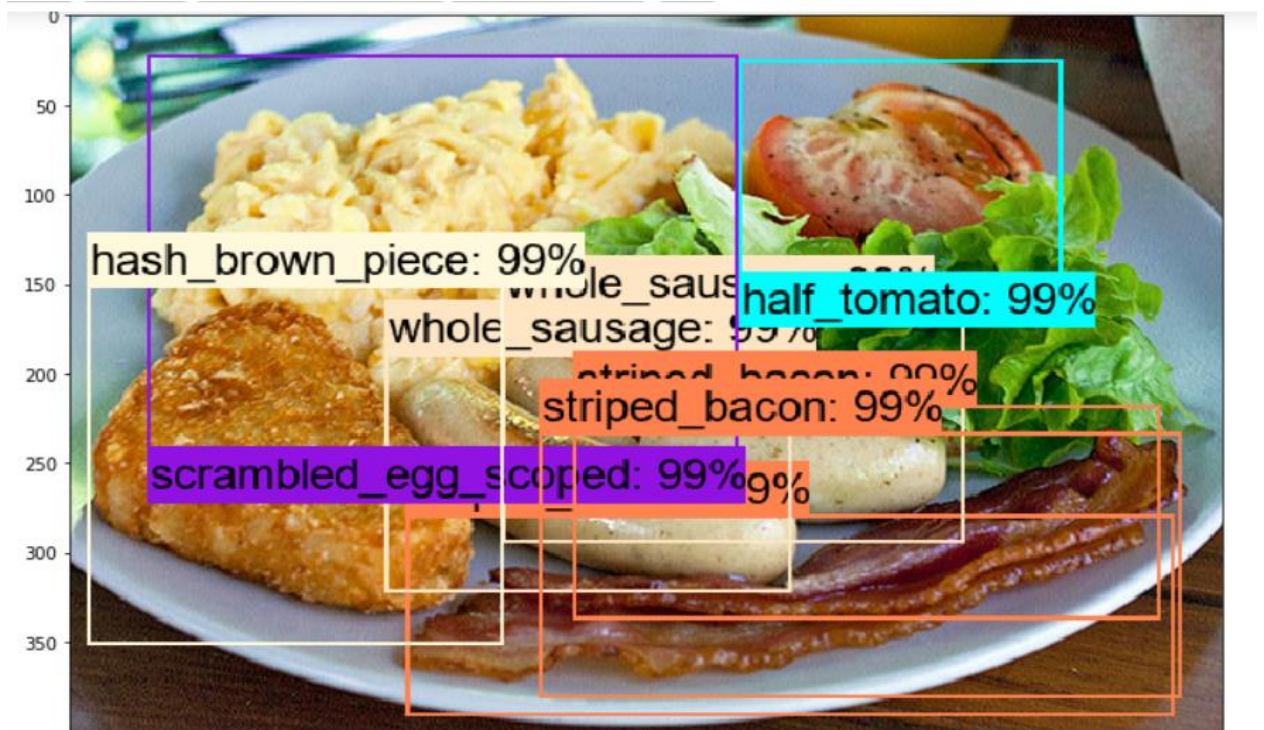
```
{'whole_fried_egg': 2, 'baked_beans_scoped': 1, 'half_toast': 2, 'ham_piece': 2}
TOTAL CALORIES IN IMAGE:469
```

**Figure 5-3: Result sample 3**



```
{'tomato_slice': 1, 'ham_piece': 1, 'whole_sausage': 2, 'half_toast': 1, 'scrambled_egg_scoped': 1,
'hash_brown_piece': 1}
TOTAL CALORIES IN IMAGE:905
```

**Figure 5-4: Result sample 4**



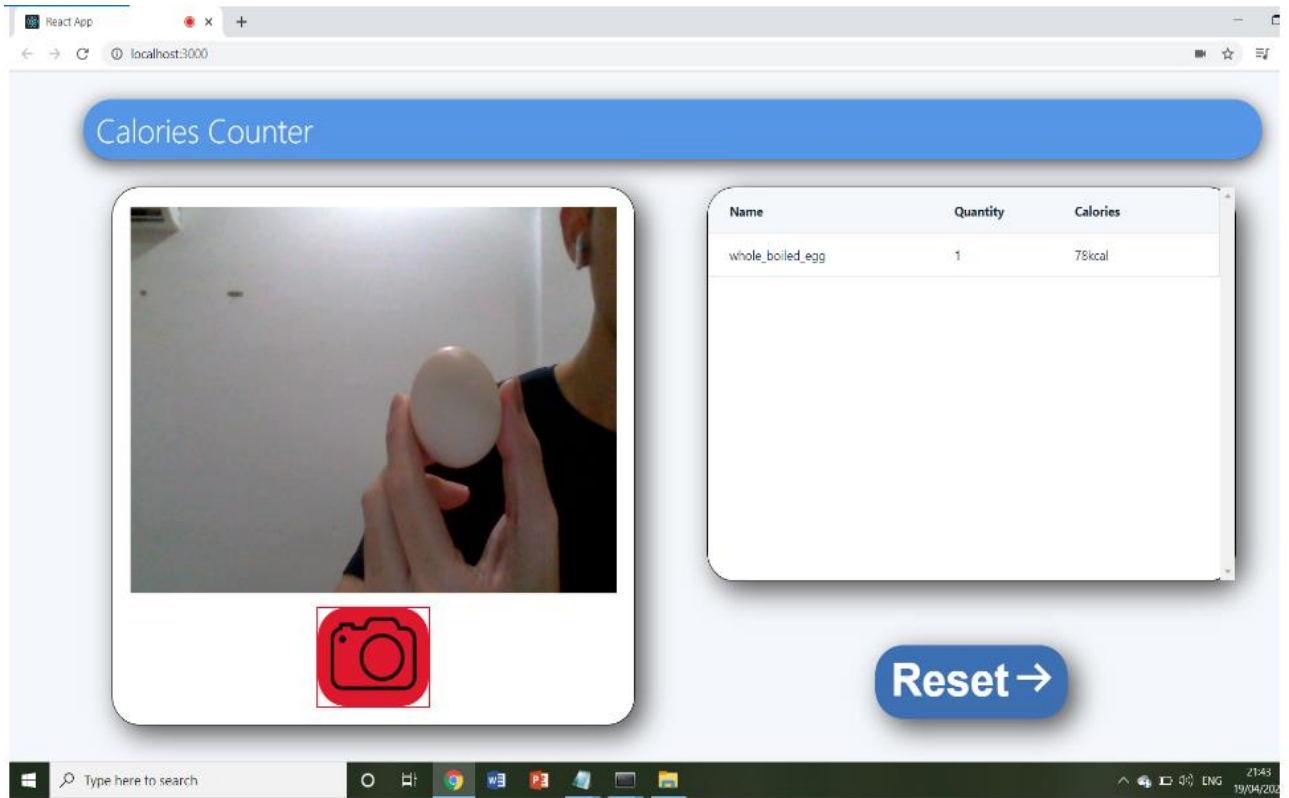
```
{'striped_bacon': 3, 'whole_sausage': 2, 'scrambled_egg_scoped': 1, 'hash_brown_piece': 1, 'half_toma
to': 1}
TOTAL CALORIES IN IMAGE:958
```

**Figure 5-5: Result sample 5**



Now, we try to test run inference on the interface that we have developed.

Testing the functionality of object detection for the inference model:



**Figure 5-6: Result sample 6**

Testing the model detection for plated meal on the live-capture, however using test set images displayed on a mobile phone since I do not have a real plated meal for the detection test :

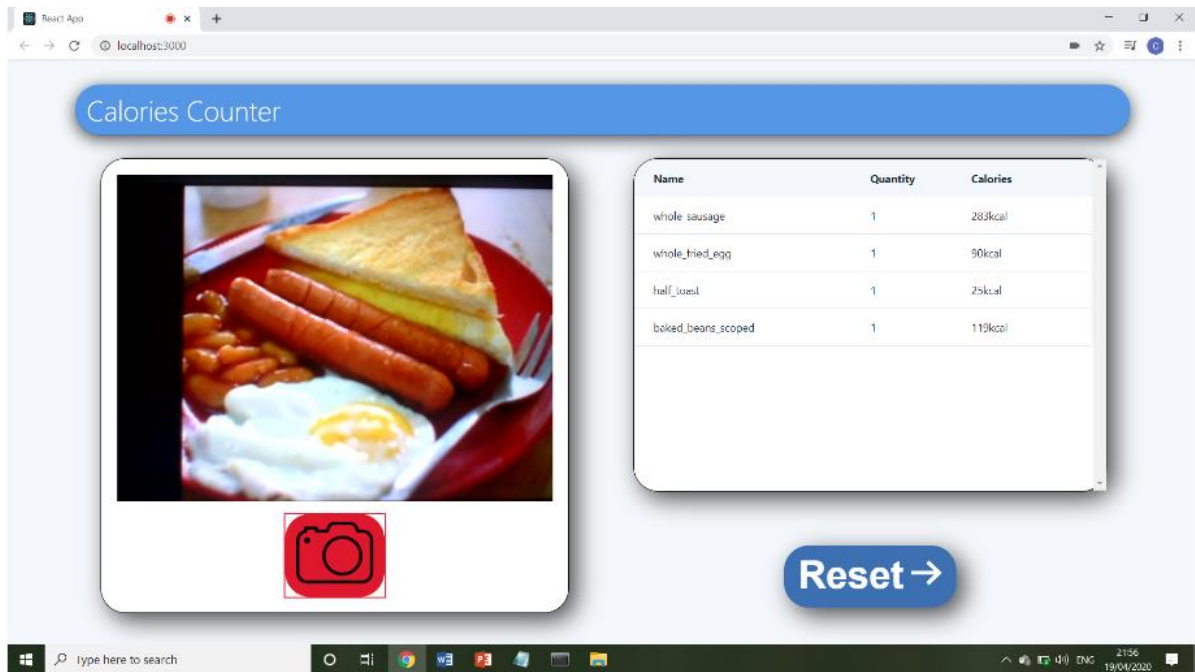


Figure 5-7: Result sample 7

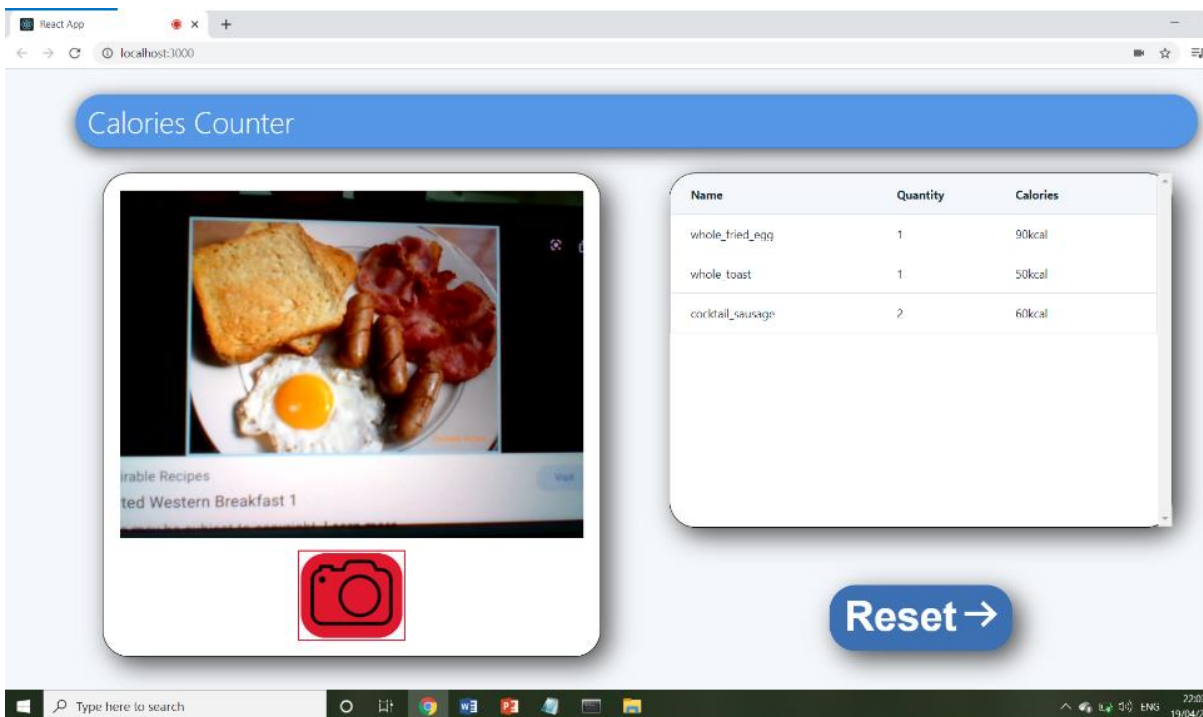


Figure 5-8: Result sample 8

## 5-2 Case Studies

### 5-2-1 Correlation of accuracy and meal complexity

Below is the case studies conducted to test the accuracy of the algorithm in counting the classes in images format. The detection box is set on condition at a confidence level of more than  $>0.5$ , which means detection box will only be shown for that particular object if the algorithm is more than 50% confidence in determining that object belongs to a certain class. Three different case studies is conducted using different number of classes in an image. The image should include overlapping objects and good number of quantities to ensure the object occlusion detection works and is able to count the total amount of that particular objects throughout the process. A total images of 10 is used for each case study depending on how many classes they need to include. Different images set is used for different case studies. Then, the accuracy is evaluated depending how well the detection works on each picture, then averaged as our results for these case studies.

Case Studies	Class Used	Average accuracy (Full tomato)	Average accuracy (Half tomato)	Average accuracy (Tomato Slice)
1	Full tomato	80%	-	-
2	Full and half tomato	82%	74%	-
3	Full, half and tomato slices	78%	72%	70%

**Table 5-1: Average accuracy of case studies 5-2-1**

The end result for this studies have proved that our algorithm works fairly well in different complexity of images. The first case study that is aim to detect only full tomato achieve an accuracy of 80%. Meanwhile, the other two case studies also shown that the detection for each classes has an average of 70% and above. The average accuracy of half

tomato and tomato slices would be slightly lower since the top part of these classes look fairly similar. It could be affected by factor like viewing angle of that object in the images. In conclusion, but comparing with other existing algorithm that is used in most of the calories counting applications out in the market, these algorithm could help us to simplify the process of calories counting by automating it and giving a more reliable result by counting the calories depending on the serving amount of their meal specifically instead of just giving a more generalized values.

The algorithm generally can detect more objects in an image. But in our case, we are looking for a more robust and accurate detection, therefore a minimum threshold of confidence interval is set at more than 0.5. With this we can reduce incorrect detection and exclude unrelated objects during the process. The model can be further fine-tuned by adding more labels and training data set.

### 5-2-2 Performance Evaluation

Referring back to Chapter 2, there is a list of applications that we have reviewed on their detection algorithm and functionality. Since we are developing the same concept with enhanced features and uses new algorithm to compute the calories values. We want to evaluate the functionality, accuracy, time consumption between our detection model and these application. 10 images of clearly visible of non-mixed food within scope of trained model is picked from the data test set to evaluate their performance in each feature. Since the model classes from each application might differs to each other, only the similar food classes and portion is being taken into consideration during the evaluation. The test is using live capture image that are non-occluded under the same light intensity, angle and position to test so there will be no variation on the test images and to avoid bias results in the evaluation. A stopwatch is being used to record the time taken between taking the picture and displaying the result. Below are the test results :

Model	Object detection	Object counting	Volume estimation	Multiple Detection	Detection Time consumption (average)	Total time consumption to compute calorie
HealthifyMe	No	Manual	Manual	N/A	N/A	Very High
Calories in Food	No	Manual	Manual	N/A	N/A	Very High
Yazio	No	Manual	Manual	N/A	N/A	Very High
Lose It!	Yes	Manual	Manual	No	3.58s	High
Calorie Mama	Yes	Manual	Manual	No	3.36s	High
Foodvisor	Yes	Manual	Manual	Yes	2.67s	Moderate
Argus	Yes	Manual	Manual	Yes	1.65s	Moderate
<b>Proposed Model</b>	Yes	Automated	Automated	Yes	1.62s	Low

**Table 5-2: Result of case studies 5-2-2**

Note:

Very High = > more than 3minutes

High = 1 – 3 minutes

Moderate = 10seconds - < 1 minute

Low = less than 10seconds

The table result is based on the sample images that are being used for the evaluation. The result like time consumption for detection and calories computational may increase depending on the complexity of the plated meal. For applications like HealthifyMe, Calories in Food and Yazio are having significant increase in time consumption to compute calories is because they do not provide the feature of object detection, users are required to manually search and select for each of the class and its respective volume and quantities. This definitely lengthen the process for the computational as it is time consuming to search items one by one especially for those meals with large quantity and with different volume. For LoseIt! and CalorieMama, they do provide object-detection feature but however, they do not have multiple detection in the algorithm, the results is provided with individual classes along with other suggestions, users allowed to add items along with the volume and quantity of the plated meal without having to search the classes. In Foodvisor and Argus, the calorie computational is reasonably fast. They included multiple detection in their algorithm which directly list down all the food items in the list without having users to search the database. However, users are still required to define the volume and quantities for each of the item. As for our proposed algorithm, each of the process mentioned in the table is automated, meaning users do not have to input anything other than taking a picture for the detection. Once picture is taken, the algorithm will directly compute all the volume estimation and quantities of the food item in the image, the algorithm also come with multiple detection which allow the detection model to list down all the items in the plated meal. This conclude that the algorithm that automated these processes will significantly ease the process of calorie computation and increase the efficiency of it. Furthermore, it will allows us to compute a more precise calorie result which is closer to the actual result in the real world by breaking down the classes into different segments.

## **Chapter 6: Conclusion**

### **6-1 Project Review**

Calories Counter has been widely used by users who are generally more health-concerned in the current society. This is applied in many healthy diet applications to help users to keep track of the numbers of calories consumed in the food so that one will not eat too much. There are many approaches that have been done to develop such function. However, majority of these functions involves complex process like manual counting and does not give an accurate and precise result.

Problems such as object occlusion detection failure and algorithm does not address for accurate food portion will eventually lead to incorrect counting resulting user unable to plan a proper diet for themselves. The aim for this project is to further enhance these algorithm in the form of improving accuracy in counting food calories by using object detection that involves object occlusion techniques and object counting so that each of every ingredients and food to be consumed by the user is determined and provide a precise result for them.

Object counting algorithm derived from crowd counting technique for food portion recognition. The algorithm should able to count the ingredients after classification. The model is fined tuned with different image intensity, optimizing epoch and integrated with object occlusion detection technique for detecting non-overt ingredients on a plate to detect overlapped objects. A script to automate to automate the process of calorie counting, by multiplying respective calories of their classes with the total number of quantities detected in the image, is developed.

The detection is also able to retrieve calories information for their respective classes from the database and output as a result for the user. The user are able to know how much calories are consumed within the plated meal. This will help user to achieve and keep track of their nutritional goals.

Lastly, the model is able to be deployed into a simple interface to ease the process of running inference for users. A simple interface is built to contain a live-capture feature which allows users to capture plated meal directly using camera/webcam to run inference quickly to obtain the output for classes, quantities as well as calories information.

## 6-2 Further Improvement

Since the project only covers a simple plated meal with less complexity and occlusion. This does not address all the problems for calories counting. There are a wide variety of foods to be covered in calories counting applications. Different food contain different complexity and approaches to be handled in the detection. For example, mixed-food with more complexity that are hardly countable and usually occluded by many other objects should use a different approach rather than the method used in this project, it is suggested to use different algorithm like volume estimation for accurate counting purposes. In all, different kind of food should be handled in different way for the detection, so that if calories counter can covers all suitable algorithm for the right kind of meal, it will definitely addresses the problem of accuracy in this field.

Also, to improve the accuracy of the detection and increase its coverage for food type, the model should be trained and fine-tuned continuously with more data and different classes. Due to time constraint and minimal hardware requirement, the model is adjusted with amount of data and classes that could be handled in the given time. However, adding more data or using method like crowdsourcing to further enhance the model will definitely improve its accuracy and coverage for the detection model.

Other than deploying to web-based application, the project could also deployed as a mobile application. However, currently the only model that support object detection in mobile application is MobileNet. MobileNet gives much more speed to train and run the inference but it has noticeable slow down in accuracy compared to Faster-RCNN. It is still a drawback as calorie counting requires accuracy to achieve the purposes.



## REFERENCE

- Hulstaert, L. (2018). *Going deep into detection*. Retrieved from <https://towardsdatascience.com/going-deep-into-object-detection-bed442d92b34>
- Hui, J. (2018). *Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)*. Retrieved from [https://medium.com/@jonathan\\_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359](https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359)
- Hui, J. (2018). mAP (mean Average Precision) for Object Detection. Retrieved from [https://medium.com/@jonathan\\_hui/map-mean-average-precision-for-object-detection-45c121a31173](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)
- Hao, G. (2017). *Faster R-CNN explained*. Retrieved from <https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8>
- Katsuta, E. (2019). *Creating a Simple App With React.js*. Retrieved from <https://medium.com/better-programming/creating-a-simple-app-with-react-js-f6aa88998952>
- Krebs, P. & Duncan, D.T. (2015). *Health App Use Among US Mobile Phone Owners: A National Survey*. Retrieved from <https://www.ncbi.nlm.nih.gov/pubmed/26537656>
- Maniuk, I. (2016). *Life Cycle of Extreme Programming*. Retrieved from <https://hygger.io/blog/life-cycle-of-extreme-programming/>
- Mwiti, D. (2019). *A 2019 Guide to Object Detection*. Retrieved from <https://heartbeat.fritz.ai/a-2019-guide-to-object-detection-9509987954c3>
- World Health Organization, (2018). *Obesity and Overweight*. Retrieved from <http://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight>

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 1
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Changes to project scope.

## 2. WORK TO BE DONE

Research on methods to establish database for the model.

## 3. PROBLEMS ENCOUNTERED

No problems.

## 4. SELF EVALUATION OF THE PROGRESS

Done some research content from internet.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 2
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Determining feasible solution.

## 2. WORK TO BE DONE

Review feasible solutions.

## 3. PROBLEMS ENCOUNTERED

Model is customized and existing database is not suitable for model.

## 4. SELF EVALUATION OF THE PROGRESS

Gain ideas on developing the project.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 3
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Comparison of fine-tuned model.

## 2. WORK TO BE DONE

Discuss important factors to be reviewed and choose one for development.

## 3. PROBLEMS ENCOUNTERED

No problems.

## 4. SELF EVALUATION OF THE PROGRESS

The comparison is done.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 4
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Discuss tools and approaches to develop functions.

## 2. WORK TO BE DONE

Review existing tool and approaches which are suitable to be use.

## 3. PROBLEMS ENCOUNTERED

Decision on using suitable approach to develop database and retrieving data.

## 4. SELF EVALUATION OF THE PROGRESS

Understanding the advantages and suitability for each approaches.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 5
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Deciding tools and requirements to build an interface for the model.

## 2. WORK TO BE DONE

Discussion and decision of suitable tools and necessary requirement to run model interface.

## 3. PROBLEMS ENCOUNTERED

No problems.

## 4. SELF EVALUATION OF THE PROGRESS

Researched and decided suitable tool.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 6
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Installation of required software and understand how to use them.

## 2. WORK TO BE DONE

Solving errors.

## 3. PROBLEMS ENCOUNTERED

Several installation errors encountered.

## 4. SELF EVALUATION OF THE PROGRESS

The error is solved.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 7
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Deciding dataset and techniques to be used.

## 2. WORK TO BE DONE

Decide dataset to be included for training.

## 3. PROBLEMS ENCOUNTERED

No problems

## 4. SELF EVALUATION OF THE PROGRESS

Ideas on dataset to use.



Supervisor's signature



Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 8
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Executing the training process.

## 2. WORK TO BE DONE

Discussion on the dataset training procedure

## 3. PROBLEMS ENCOUNTERED

Hardware with low requirement slowing down the process.

## 4. SELF EVALUATION OF THE PROGRESS

Obtained trained dataset.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 9
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Reviewing result of trained data.

## 2. WORK TO BE DONE

Discussion on methods to improve dataset.

## 3. PROBLEMS ENCOUNTERED

Result is not satisfying.

## 4. SELF EVALUATION OF THE PROGRESS

Retrained the data.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 10
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Review on new trained dataset.

## 2. WORK TO BE DONE

Discussion on the new trained dataset.

## 3. PROBLEMS ENCOUNTERED

No problems

## 4. SELF EVALUATION OF THE PROGRESS

Revamp is done.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 11
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Discussion regarding to the documentation report.

## 2. WORK TO BE DONE

Modify and add content to the report.

## 3. PROBLEMS ENCOUNTERED

No problems

## 4. SELF EVALUATION OF THE PROGRESS

Report is done.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> JAN 2020	<b>Study week no.:</b> Week 12
<b>Student Name &amp; ID:</b> Woon Zheng Li 1605571	
<b>Supervisor:</b> Dr. Aun YiChiet	
<b>Project Title:</b> Accurate calorie counting algorithm for carefully plated dishes	

## 1. WORK DONE

Review a complete report and overall development project.

## 2. WORK TO BE DONE

Further enhance and improve the algorithm.

## 3. PROBLEMS ENCOUNTERED

No problems.

## 4. SELF EVALUATION OF THE PROGRESS

Overall progress nearly complete.



Supervisor's signature



Student's signature

# POSTER

## Problem:

Major health problems has been increasing significantly in recent years. More people are conscious about their diet and attempt to control calories. People are mostly using modern calorie counting application existed in market today. However, these application generate result based on a generalized portion without counting algorithm and sometimes manually selected by user, causing inefficiency and incorrect calories count which mislead users into incorrect control of their diet.



## Project Scope:

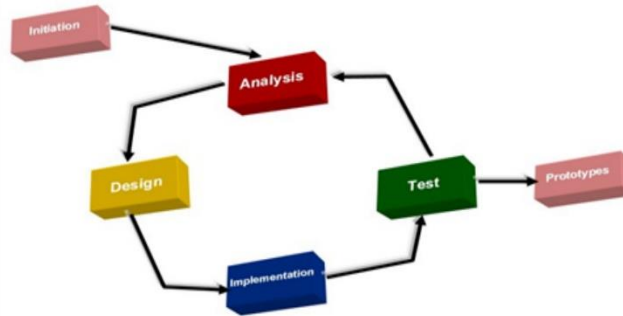
An improvement plan aims to develop a accurate calorie counting algorithm for clean and simple plated dishes that are able to detect the quantity and the portion serving of the meal to achieve a more accurate calories counter result.

# Accurate calorie counting algorithm for carefully plated dishes

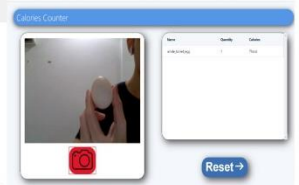
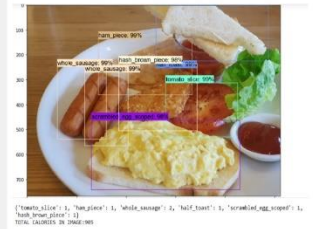
By Woon Zheng Li

## Methodology

Protype methodology is being used. System being prototyped and testing is carried out to determine its performance and evaluation to the model. If the result is not satisfying, it will revert back to previous phase for necessary modification and process is repeated a satisfied result is being produced.



## Result:



## Main Objective

- Develop a detection model for clean plated dishes components
- Integrate counting algorithm for the model
- Able to retrieve nutritional information for output
- Building an interface to run the inference

# TURNITIN SIMILARITY REPORT

**ACCURATE CALORIE COUNTING ALGORITHM FOR CAREFULLY PLATED DISHES**  
 BY  
 WOON ZHENG LI

Match Overview

12%

Match 1 of 17

30	Submitted to Leeds Be... Student Paper	<1% >
31	Submitted to Oxford Br... Student Paper	<1% >
32	neurorobotics.me Internet Source	<1% >
33	Submitted to University... Student Paper	<1% >
34	Submitted to University... Student Paper	<1% >
35	scholar.lib.vt.edu Internet Source	<1% >
36	Submitted to Coventry ... Student Paper	<1% >
37	"Biometric Recognition"... Publication	<1% >

## FYP2-report

### ORIGINALITY REPORT

12%	6%	5%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

1	<a href="http://eprints.utar.edu.my" style="color: #f00; font-weight: bold;">eprints.utar.edu.my</a> <small>Internet Source</small>	4%
2	<span style="color: #f00; font-weight: bold;">Elif Baykal, Hulya Dogan, Mustafa Emre Ercin, Safak Ersoz, Murat Ekinci. "Modern convolutional object detectors for nuclei detection on pleural effusion cytology images", Multimedia Tools and Applications, 2019</span> <small>Publication</small>	2%
3	<a href="http://www.newportus.com" style="color: #f00; font-weight: bold;">www.newportus.com</a> <small>Internet Source</small>	1%
4	<a href="http://medium.com" style="color: #f00; font-weight: bold;">medium.com</a> <small>Internet Source</small>	<1%
5	<span style="color: #f00; font-weight: bold;">Submitted to Higher Education Commission Pakistan</span> <small>Student Paper</small>	<1%
6	<a href="http://www.mmrpc.org" style="color: #f00; font-weight: bold;">www.mmrpc.org</a> <small>Internet Source</small>	<1%
7	<span style="color: #f00; font-weight: bold;">Submitted to Heriot-Watt University</span> <small>Student Paper</small>	<1%



**FACULTY OF INFORMATION AND COMMUNICATION  
TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	WOON ZHENG LI
<b>ID Number(s)</b>	1605571
<b>Programme / Course</b>	BUSINESS INFORMATION SYSTEM
<b>Title of Final Year Project</b>	ACCURATE CALORIE COUNTING ALGORITHM FOR CAREFULLY PLATED DISHES

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>12</u> %</b>  <b>Similarity by source</b> Internet Sources: <u>6</u> % Publications: <u>5</u> % Student Papers: <u>6</u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>1</u></b>	In text citation
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> <b>(i) Overall similarity index is 20% and below, and</b> <b>(ii) Matching of individual sources listed must be less than 3% each, and</b> <b>(iii) Matching texts in continuous block must not exceed 8 words</b> <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

***Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.***

\_\_\_\_\_  
Signature of Supervisor

Name: DR. Aun YiChiet

Date: 24 April 2020

\_\_\_\_\_  
Signature of Co-Supervisor

Name: \_\_\_\_\_

Date: \_\_\_\_\_





**UNIVERSITI TUNKU ABDUL RAHMAN**



**FACULTY OF INFORMATION & COMMUNICATION  
TECHNOLOGY (KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	16ACB05571
Student Name	Woon Zheng Li
Supervisor Name	Dr. Aun YiChiet

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
/	Front Cover
/	Signed Report Status Declaration Form
/	Title Page
/	Signed form of the Declaration of Originality
/	Acknowledgement
/	Abstract
/	Table of Contents
/	List of Figures (if applicable)
/	List of Tables (if applicable)
	List of Symbols (if applicable)
/	List of Abbreviations (if applicable)
/	Chapters / Content
/	Bibliography (or References)
/	All references in bibliography are cited in the thesis, especially in the chapter of literature review
/	Appendices (if applicable)
/	Poster
/	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

\*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <p align="center"></p> <p>_____ (Signature of Student) Date: 19 April 2020</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <p align="center"></p> <p>_____ (Signature of Supervisor) Date: 24 April 2020</p>
---	--

