淡江大學

資訊創新與科技學系

==

108 學年度專題實作書面報告

利用影像辨識加快自助餐結帳流程

Image Recognition Approach for

Expediting Chinese Cafeteria Checkout Process

學生：	吳柏霆	405850420
	鄒亞微	405850206
	湯智天	405856047
指導教授：	張峯誠	博士

中華民國 108 年 12 月

摘要

在淡江蘭陽的日常生活中，每當接近用餐時間時，學校自助餐總是大排長龍。不管是學生、教授以及行政同仁們，都會在這車水馬龍的隊伍中耽擱一會兒。然而這短短的十分鐘、十五分鐘，對於緊接下一堂有行程的人們，會造成極大的不便。俗話說：「時間就是金錢」，我們希望透過這個專題將現有的結帳流程，結合影像辨識的技術，減少大家排隊的時間。在這個專題中，我們將使用現有的影像辨識模型結構，導入遷移學習的概念，訓練出足以辨識自助餐便當菜的模型。透過安裝在結帳檯上方的攝影機，將捕捉到顧客結帳餐盤的畫面輸入至已訓練好的模型中，達到自動計算結帳金額及顯示出該餐的營養資訊。我們使用了五百張拍攝的自助餐照片訓練模型，在訓練時達到了平均 90%的準確率(mAP)，並且在實測中準確率(mAP)也達到了將近 70%。我們相信將我們的專題應用在淡江蘭陽的學校自助餐中，能夠有效率的減少大家排隊等待的時間。

Abstract

One of the common running themes in modern day Chinese cafeterias is the hold up in foot traffic in queueing due to food checkout. As the famous phrase goes: “time is money”, we believe this issue can be addressed by introducing real time image recognition techniques during food checkout process. By using a webcam to provide live video feed to our model, we are able to eliminate the need to perform manual price calculations. Our model uses YOLOv3 architecture as its base with its training carried out five cumulative times in groups of one-hundred images with its first training starting at 100 and its last at 500. Initial results show that while losses are high, mAP (mean average precision) comes out low. It is noted that by further feeding the dataset with images, model’s losses can be lowered while rasing mAP to a higher percentage. As of this writing, total losses are in the twenties with mAP around 90%. Our data shows that with a high degree of precision given from our trained model, by applying it to a Chinese cafeteria food setting such as those found in a college or university campus, expediting the queue can be a realistic possibility.

Acknowledgement

Special mention goes to our instructor Professor Chang for accompanying us throughout these years. His continued patience and dedication toward teaching his respective subject matter are generally agreed to be one of the best here at Tamkang University. We specifically appreciate that Professor Chang takes the time to digest the information beforehand as well as taking the necessary strides needed to teach the materials that his class is listed for. For this we express our heartfelt thanks: Thank you.

We would also like to extend our thanks and appreciation to the kitchen staff, they are the ones keeping the school fed and running. For us in this particular year, we would like to express our gratitude towards the ladies working at Feng-Lin Restaurant, more specifically, Feng-Lin and her older sister. Throughout the course of this project, by the powers that be, we have eaten and consumed what seem to us a lifetime's worth of your food. Thank you for accompanying us along the way and providing us the food for us to provide data for.

Finally, we'd like to express our appreciation to the general staff here at TKU Lanyang Campus. We thank you for providing the platform, framework, and support for us students to stand upon. Thank you for keeping this steady foundational rock, that is the campus, in the same condition it was as we first arrived four years ago.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

In this chapter we will be explaining our main motivation and resulting goal for our project. Section 1.1 outlines, in a comedic and somewhat over exaggerated fashion, what a typical day is like here in the cafeteria at Tamkang University Lanyang Campus producing our motivation and goals for our project. Following that, Section 1.2 briefly details the overall structure, or more specific purpose of each chapter of this paper.

1.1 Motivation and Project Goal

Imagine yourself this: You woke up to a packed and busy schedule, that schedule so filled with activity. In order to fulfill it, you had to skip out on breakfast. What’s more, having started the day off on an empty stomach, your essentially forced to endure the grind while whittling the hours away. One could even say that you are literally running on fumes. And with the tank that is your stomach running past empty, the only sense of relief is found within a fleeting one hour’s worth of lunch break. So then, the best way to celebrate the trek through the proverbial desert of mundaneness than seeking solace in the oasis is called the “campus cafeteria.” Here we are, and ironically, just like an oasis, the moment you arrive, the further away it seems.

Figure 1.1. The waiting line is extended to the collecting area during mealtime.

One of the most prized time and place in our days on campus is unarguably lunch time in the cafeterias. It is this period of time where we as individuals, not as students or staff members, can truly wind down on campus. More importantly, it is this period of time we are allotted to consume actual food items, in order to fuel our body for whatever activity we may plan on doing. It’s the halfway point of the day, and unlike breakfast in the morning hours, where foot traffic sparse and activity low, lunch in the cafeteria, with its high volumes of activity, gives us a precious platform to socialize and communicate with one another. Unfortunately, anyone who has resided on campus for any extended period of time would know, arguably one of the biggest downsides with dining in the campus cafeteria is the time spent waiting. There is a famous three worded phrase which resonates with all of us: “time is money”. Taking that to heart, our motivation for our project is the dreaded queue.

By examining the cafeteria periodically throughout the day, we concluded that it is during the rush hours (or typically during the general meal times) that queueing mostly occur. It is during this timeframe when the orders pile up and hands run short. Most importantly, we would like to specify that it is mostly the case when high volumes of customers, waiting in line during the food checkout process does this queuing situation be at its absolute worse.

Below is a quick survey we conducted on Dec. 9, 2019 from the timeframe of 11:00-13:00 and 17:00-19:00 in Feng-Lin cafeteria in order to illustrate our point. Note only transactions from ordering cafeteria food are recorded. Direct order does not count and has been omitted. Info. Lunch: 47 servings. Dinner: 38 servings. For the sake of brevity this chart combines both.

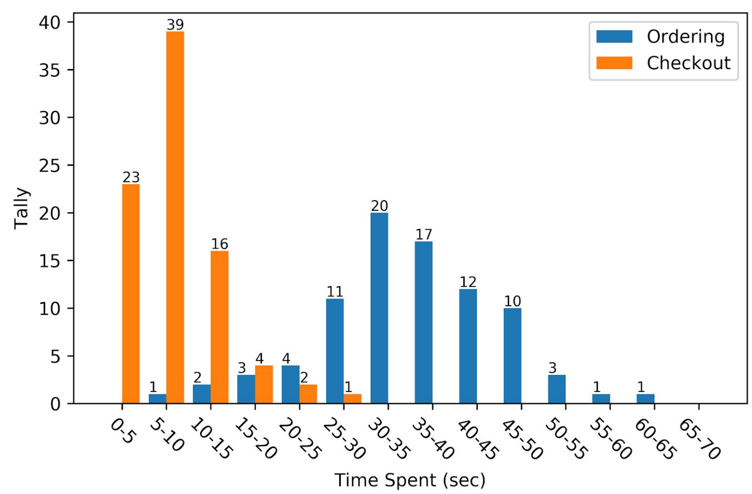


Figure 1.2. Survey of the queuing time during check out.

So we ask ourselves this seemingly innocent question: How do we address this? And the answer we arrived at is simple: As senior CS students here at Lanyang Campus, we figured the most fitting way to address this challenge is by utilizing innovative technology. This brings us to our project goal: Developing a system that will expedite the queueing during food checkout in Chinese cafeterias.

1.2 Organization

Including this chapter, this report has been divided into 7 specific chapters. In sequential order, they are Introduction, Background Information, System Analysis, System Design, Implementation and Demonstration, Discussion, and lastly Conclusions.

As some of the terminology used throughout the paper is relatively new, the Background Information is here to provide brief yet concise explanation on them. While in System Analysis we primarily focus on User Stories which deals mainly with different individual input and use them to assist in the formation of what our project's system structure ought to become. Next comes the System Design, this chapter focuses on the various integral and essential components used in building up our proposed system. After which follows the chapter on Implementation and Demonstration, in this chapter we mostly go over the various exercises we conducted using our system and illustrate our findings. Lastly, in the Conclusions we discuss our overall thoughts on our system, such as the possibilities in regards to future improvements on the system and what generally what we've learned and accomplished in creating our system.

Chapter 2 Background Information

In this chapter, some terminologies that are frequently used in this report will be explained. We believe that by providing these background information, readers should possess adequate domain knowledge to read through this report comfortably.

2.1 Image Recognition

Image recognition, in the context of computer vision, is the ability of computers to identify objects, places, people, writing and actions in images through the use of algorithm and the concept of machine learning [1]. In this section, we describe some terminologies and the concept of image recognition we are using in our system.

2.1.1 Image Classification

One of the basic approaches in image recognition is image classification, which classifies a given image into one of the predetermined categories.

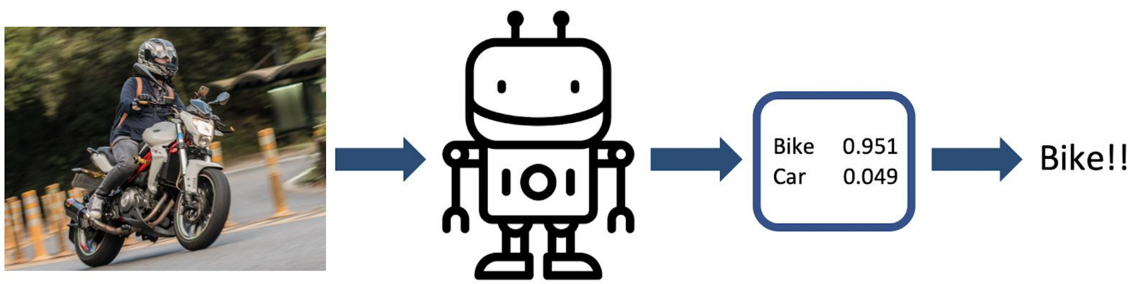


Figure 2.1. Image classification model classifies the given image.

2.1.2 Object Detection

Object detection is another image recognition approach that deals with identifying and locating objects of certain classes in the image [2].

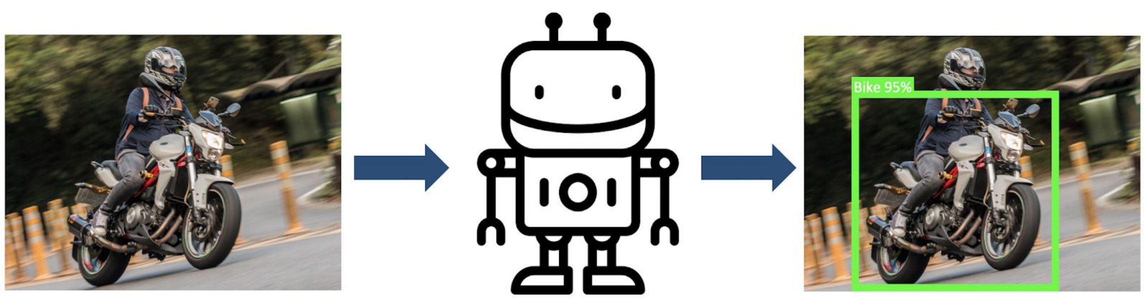


Figure 2.2. Object detection model locates the specific object.

2.1.3 Bounding Box

Bounding box is a layered box that frames and encompasses an object in its entirety. It is usually the output of all kinds of the object detection algorithm.

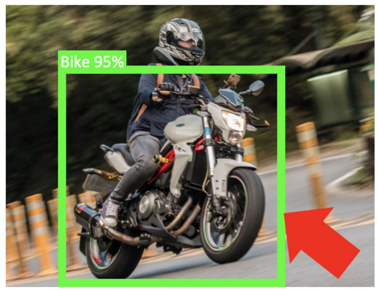


Figure 2.3. The bounding box.

2.1.4 Grid

Grid, as it will be mentioned in Sec. 4.1.3, represents a network of lines that cross each other to form a series of small boxes. The model we adopt in this report performs object detection in every box.

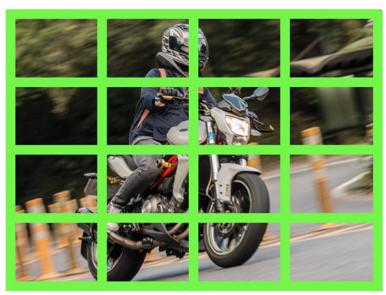


Figure 2.4. A 4x4 grid over an image.

2.2 YOLO (You Only Look Once)

YOLO (You Only Look Once) is a well-known algorithm for object detection proposed by Joseph Redmon in 2015. Compared to other algorithms, YOLO uses a totally different way to process the object detection. It takes an image as an input, and returns the position of the bounding box [Sec. 2.1.3] as well as the confidence being that category.

2.2.1 Structure of YOLOv3

The model structure of YOLOv3 is shown in Fig. 2.8. There are a total of 106 layers inside the network, which is very tremendous and complicated. To further understand the network architecture, we will first explain the terminologies and concepts used in YOLOv3 network.

2.2.1.1 Convolutional Layer and Filters

Convolutional layer and filters play an important role in the field of image recognition. They are responsible for retrieving features from an image. As Fig. 2.5 shows, convolutional layer takes an input such as an image, in this case is a 7-by-7 matrix(I), and computes the inner product with its filter(K). The filter slides through the entire matrix from left to right and from top to bottom in order to compute the resulting matrix(I*K). Figure 2.6 demonstrates an image being convoluted with a 3-by-3 filter.

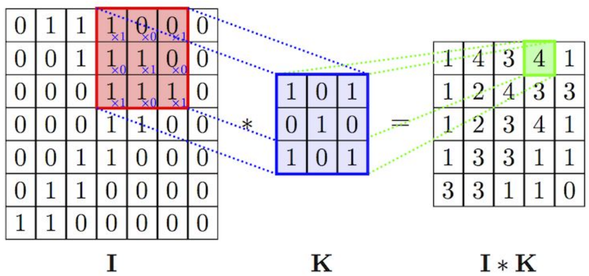


Figure 2.5. The convolutional layer [3].

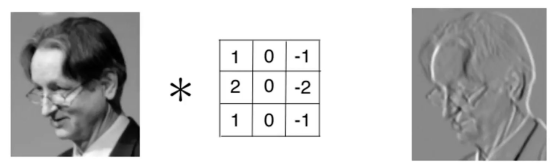


Figure 2.6. The Filter [4].

2.2.1.2 Residual Block

Usually, a layer is fed only to the following layer in traditional neural networks. However, with an implementation of residual block, a layer will not only be fed to the next layer but also the layer after the next. In this way, some of the un-convoluted features can be preserved and referenced by the layer after next.

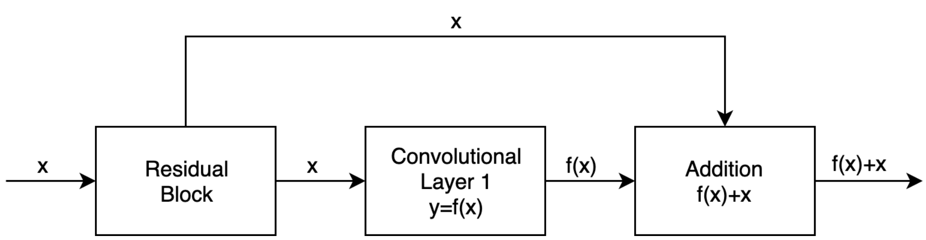


Figure 2.7. Concept of Residual Block.

2.2.1.3 Analysis of Structure

The entire YOLOv3 structure can be divided into two parts: base network and detection network. Base network is used to extract features given an input while detection network is used to predict outputs based on the features extracted from base network.

The left part of Fig. 2.8 is the base network just mentioned. It is the main part of feature extraction in YOLOv3. The blue part of Fig. 2.8 is the detection network in YOLOv3.

As the base network goes deeper, small objects become harder to be detected due to the features of small objects being convoluted too many times. To solve this problem, the detection network of YOLOv3 is connected in three levels from its base network. The first detection is performed to detect small objects in the early stage; the third detection is performed to detect large object using the features outputted from the very last stage of the base network.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	416 × 416
	Convolutional	64	3 × 3/2	208 × 208
1 ×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			208 × 208
	Convolutional	128	3 × 3/2	104 × 104
2 ×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			104 × 104
	Convolutional	256	3 × 3/2	52 × 52
8 ×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			52 × 52
	Convolutional	512	3 × 3/2	26 × 26
8 ×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			26 × 26
	Convolutional	1024	3 × 3/2	13 × 13
4 ×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			13 × 13

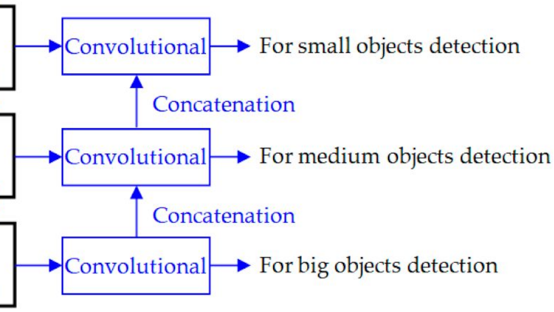


Figure 2.8. The network architecture of YOLOv3 [5].

2.2.2 Evaluation Indicators

There are two common indicators that are adopted to evaluate YOLOv3 model in our system. We will describe the concepts of the two indicators in this section.

2.2.2.1 Loss

‘Loss’ is an indicator of measuring the failure prediction of a model. In YOLOv3, ‘loss’ is the sum of classification loss, localization loss and confidence loss.

- Classification loss is the failure of predicting an object into a wrong category.
- Localization loss is the bias between the correct location and the predicted location of the model when predicting an object.
- Confidence loss is the lack-of-confidence while predicting objects of a model. It is calculated by *(1 - confidence)*.

2.2.2.2 mean Average Precision (mAP)

mAP is an indicator of evaluating how precise a model is. It is calculated by averaging the precision of each category. The actual formula of calculating the precision of each category varies based on different implementation of framework, so we only briefly introduce the concept, but commonly, the range of mAP is usually between 0.0 to 1.0.

2.3 Transfer Learning

Transfer learning is a popular method in machine learning because it allows us to build accurate models in a timesaving way [6]. In other words, transfer learning is a way to build upon something that has already been established. The advantage of this is not having to perform training and learning patterns from scratch. This in effect saves a considerable amount of time and resources one would usually spend had they done the simulations from the beginning.

2.4 Back Propagation

Back Propagation is a learning algorithm to compute gradient descent [7] with respect to weights. Desired outputs are compared to achieved system outputs, and then the systems are tuned by adjusting the connection weights in order to narrow the difference between the two as much as possible. The algorithm gets its name because the weights are updated backwards, from output to input. [8]

2.5 COCO Dataset

COCO dataset is a collection of data cleaned up by Microsoft Company. In order to promote the development of object detection, Microsoft collects vast amounts of images which was taken in a scene of a naturally occuring daily life, and reorganizes those images to provide developers a standard dataset for image recognition.

COCO dataset contains images of 91 categories of objects, and those objects can easily be recognized. With a total of 2.5 million labeled objects in 328 thousands of images, the creation of the dataset was built from crowd of workers for object classification, object detection and instance segmentation.

2.6 Google Colaboratory

Google Colaboratory (also known as Google Colab) is a free service provided by Google for education or research purposes involving machine learning. The sole requirement in using Google Colab is to have a Google account. The graphical user interface of Google Colab is almost the same as an online version of Jupyter Notebook [9], and once the service is turned on, users can directly write Python code, which is an efficient way for users to establish the environment. Above all else, Google Colab offers users a free GPU service for calculating huge and complicated operations in training machine learning models.

Chapter 3 System Analysis

Having created the goal in Sec. 1.1, we first list out all user stories we touched upon in Sec. 3.1. Going into Sec. 3.2, we concretize the user stories to more implementable requirements. Finally, come up with an initial architecture in Sec. 3.3.

3.1 User Stories

To expedite the queueing during the rush hour in the campus cafeteria, we propose a real time image recognition techniques during the checkout process. Below are some scenarios that our system attempts to resolve.

3.1.1 Customer

A. As a customer, I want to put the plate on the table and have it instantly learn the total amount of money needed to pay, so that I can leave the waiting queue as soon as possible.
B. As a customer, I want to learn not only the total amount of money needed to be paid but its nutrition facts as well, so that I may more easily keep track of my diet.

3.1.2 Entree Serving Staff

A. As an entree serving staff, I want to monitor the correctness of identified food on the plate so that I can ensure the calculated amount is always accurate.

3.1.3 Cafeteria Owner

A. As a cafeteria owner, I want to update the image recognition system easily so that it can detect newly launched entrees.
B. As a cafeteria owner, I want to store all the transaction record so that I can quantify the popularity of each entree.
C. As a cafeteria owner, I want to record all the transaction time so that I can prepare the entrees with adequate amount during the time frame that its purchase volume is at its highest.
D. As a cafeteria owner, I want to run the image recognition system on embedded devices so that the volume of the whole system is small for installation.

3.2 Functional & Non-Functional Requirements

After acquiring user stories from each perspective, we detail the user stories listed above into more concrete functional and non-functional requirements.

Table 3.1
Functional Requirements

User Story	Functional Requirements	Acceptance Criterias
3.1.1.B	• Create image recognition model to recognize entrees. • Pull up the pre-stored price of each detected entree and sum them up. • Pull up the pre-stored nutrition facts of each detected entree and sum them up.	• The correct amount of money to pay is displayed. • The correct nutrition facts is displayed.
3.1.2.A	• Display bounding boxes [Sec. 2.1.3] with its category around every recognized entree.	• Bounding boxes are displayed with vivid colors and entree names.
3.1.3.B	• Create a database for storing receipt. • Create a save-receipt function to store the receipt in the database once a plate of entrees is detected.	• Purposely put a plate of entrees under the webcam and examine the record in the database.
3.1.3.C	• Add another field called timestamp for the receipt table. • Modify save-receipt function to save the current timestamp too.	• Examine the records in the database should contain their stored time.

Table 3.2
Non-Functional Requirements

User Story	Quality Attribute	Non-Functional Requirement
3.1.1.A	Performance	• The time to place the plate under the camera for the system for system to display the total amount paid should be less than 2 second.
3.1.3.A	Maintainability	• Modularize the system in order to achieve low coupling, so the image recognition model, entree categories, prices, nutrition facts is alterable.
3.1.3.D	Portability	• Program should be compatible with Raspberry Pi and Google Coral, so that inference time can be lowered.

3.3 Initial Architecture

By splitting the high-level user stories down to factual requirements, we are then able to build up the initial architecture for our system. The system first inputs an image extracted from the webcam, and passes it down to the image recognition model. The image recognition model is in charge of detecting entrees as well as their general locations. After having detected these entrees, the system pulls out the corresponding price and nutritional facts from the config file and sums them up respectively. Next, the system draws bounding boxes around the detected entrees and prints out the price and nutritional facts. At the same time, the system also saves a copy of the receipt including entrees with their prices and the purchase time into the database. Finally, from the system, the image is then outputted with all the information prescribed.

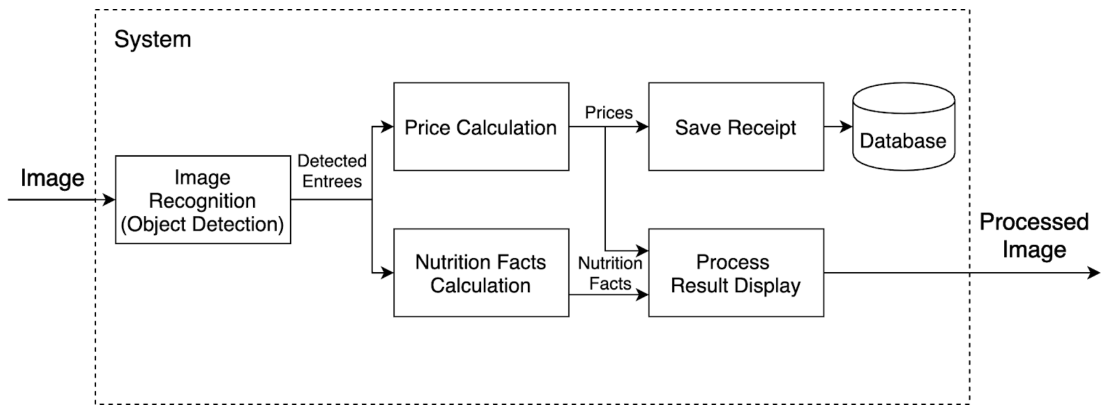


Figure 3.1. Initial architecture diagram.

Chapter 4 System Design

In this chapter, we describe how we designed our system according to the architecture we made in Sec. 3.3. The system design is divided into five parts. These being Image Recognition in Sec. 4.1, Price Calculation in Sec. 4.2, Nutrition Facts Calculation in Sec. 4.3, Process Result Display in Sec. 4.4 and lastly, Saving Receipt in Sec. 4.5.

4.1 Image Recognition

There have been some prominent image recognition model architectures existed in the field of machine learning. In following the motto “to stand on the shoulders of giants”, we will apply a technique called transfer learning to train on a pre-existing model but having it learning the entrees. In this section, we elaborate on the details during each step.

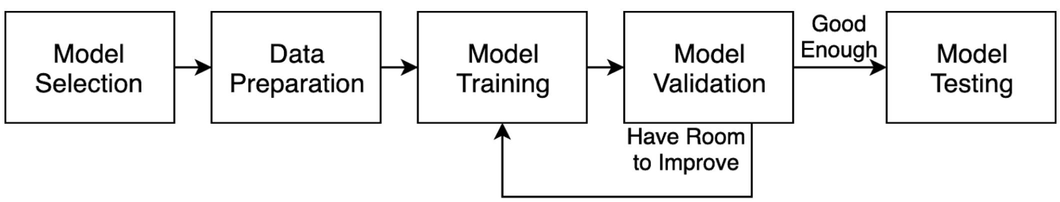


Figure 4.1. The process of building up an image recognition model to detect entrees.

4.1.1 Model Selection

We compare four variations of popular image recognition models which are Faster R-CNN, SSD, R-FCN and YOLO. Finally, we weigh each between all four and finally choose the third version of YOLO (YOLOv3) to become our image recognition model structure.

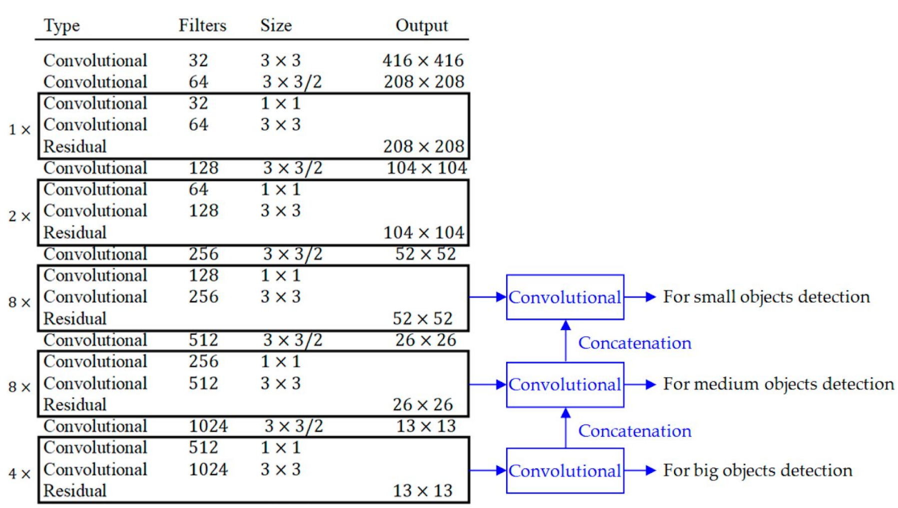


Figure 4.2. The network architecture of YOLOv3.

4.1.2 Data Preparation

Image recognition model takes full color images as its input data. Before moving on to the model training process, we first illustrate how we collect the data, preprocess the raw data and split the preprocessed data into piles for different purpose in training.

4.1.2.1 Data Collection

Due to development time constraints, we limit the amount of food entrees down to twelve most popular ones and make categories out of them. The categories are:

- Carrot Eggs
- Chicken Nuggets
- Chinese Cabbage
- Chinese Sausages
- Curry
- Fried Chicken
- Fried Dumplings
- Fried Egg
- Mung Bean Sprouts
- Rice
- Triangle Hash Browns
- Water Spinach

We routinely take sample pictures of the plate of food with those twelve, taking pictures from the plate's four sides and from the top from student cafeteria. In the effort to simulate different lighting condition for the dataset, we also take pictures with differing levels of brightness.

4.1.2.2 Data Preprocessing

Once the data was collected, we manually pre-processed it by having any displayed entrees boxed and labelled using a program called labelImg [10]. The labelImg will save an additional file that records the coordinates of all the labeled entrees.

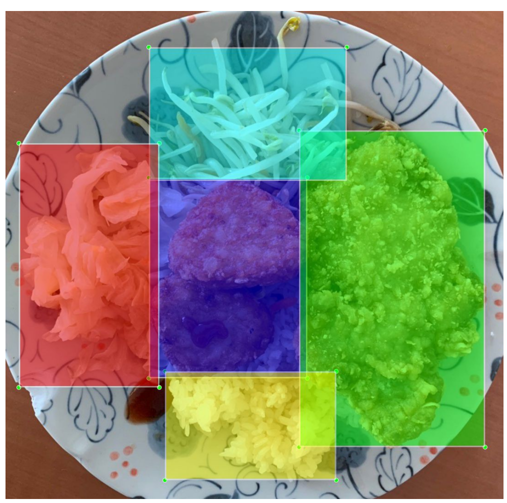


Figure 4.3. A preprocessed image with all entrees being labeled.

4.1.2.3 Splitting Data

After all the collected data was labeled, we split the data into three piles, these being the training dataset, validation dataset, and testing dataset. The training dataset is for training purposes that allows the model to learn the patterns of entrees within this pile; The validation dataset is used to examine whether the model is well trained or should continue advance training after performing each training; the testing dataset is utilized to test the accuracy of a model after it is believed well-trained by verified by the validation dataset.

4.1.3 Model Training

Instead of training from scratch, we apply transfer learning [Sec. 2.3] onto the system. This takes a model already trained for recognizing objects in COCO dataset [Sec. 2.5], and uses it as a starting point to train to recognize our entrees. In this way the training process can be sped up manifolds as there are pre-trained filters in the convolutional layers [Sec. 2.2.1.1] in the model.

The three output layers from the model are rebuilt with the exclusive shape of the twelve, so that the output shape of the modified model will fit with the twelve entrees.

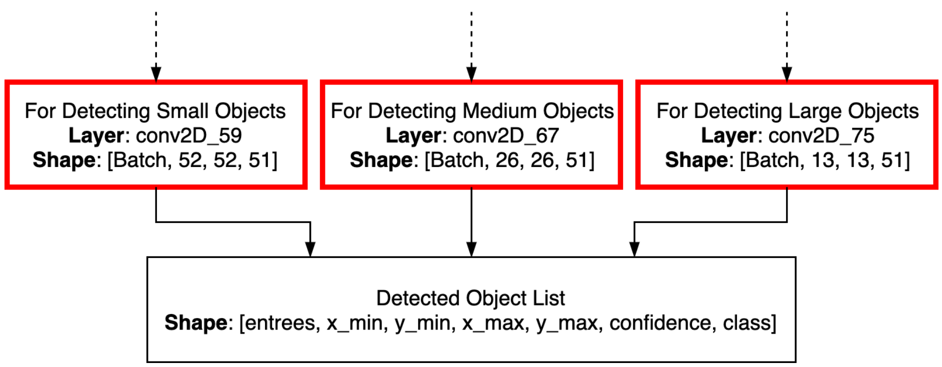


Figure 4.4. The three rebuilt output layers.

Being more specific, the adjusted three output shapes, ‘Shape: [Batch, x, y, z]’, where the parameters x and y are noted by 52, 26 and 13, represent the amount of grids [Sec. 2.1.4] contained in the layers respectively. The first output layer contains the most amount of grids at 52x52, the second contains the half that at 26x26, and the third output layer contains half of the second at 13x13 grids. This means that given the amount of grids in each layer, the coverage of each grid per layer becomes increasingly larger. In other words, this means that the first layer has the least amount of coverage in the grids which is ideal and meant for detecting smaller objects whereas the third layer with the most amount of coverage is ideal and meant for detecting larger objects.

To explain the z parameter, note that in each batch, each x and y contains 3 bounding boxes with 17 values, this being 3x17 in short explaining the number 51. By default all YOLO based model structures contain 5 values spontaneously, which represent the coordinal x-min, y-min, x-max, y-max and confidence for detecting an object. The remaining 12 values indicate our twelve categories used to define the entrees.

The entire training process is carried out by recurring feeding the images from training dataset in batches into the model and back propagating [Sec. 2.4] the correct location of the entrees for the model to update its parameter in the filter of each convolutional layer.

4.1.4 Model Validation

Whenever the training process trains through one round of the entire training dataset, model validation is performed to examine whether the model still has room to improve. The indicator so-called ‘loss’ [Sec. 2.2.2.1] should continue descending until it starts to plateau. This means the learning of the model is saturated.

4.1.5 Model Testing

As soon as the training is done, we will use the testing dataset to test out the model. The indicator ‘mAP’ [Sec. 2.2.2.2] will be used to evaluate the precision of the model. The higher means the model is extraordinary in recognizing the entrees from the images that it has never seen before.

4.2 Price Calculation

We survey the price per serving of the twelve entrees (see Appx. A) and store them into a config file. Whenever the model detects the entrees from a given image. The system pulls up the pre-stored prices from the config file and sum them up.

4.3 Nutrition Facts Calculation

There are two steps to calculate the nutrition facts for a customer ingest from that dish. First we have to obtain the nutrition facts of a particular weight or piece of an entree. Next, we have to measure the average weight per serving of that entree. After all, we can multiply them together to get the total amount of nutrition facts being consumed. The nutrition facts of a particular weight or piece of an entree and the average weight per serving of that entree will also be stored in the config file.

4.4 Process Result Display

As soon as the system gets the output from the price calculation process and the nutrition facts calculation process, it will render the entrees, nutrition facts and prices on the right hand side of the image with green font in black background, and this processed image will be displayed on the monitor during the checkout desk.

4.5 Saving Receipt

Besides displaying the receipt (calculation results) on the monitor, we also store a copy of the receipt into the database for further analysis. Below is the structure of the designed database tables.

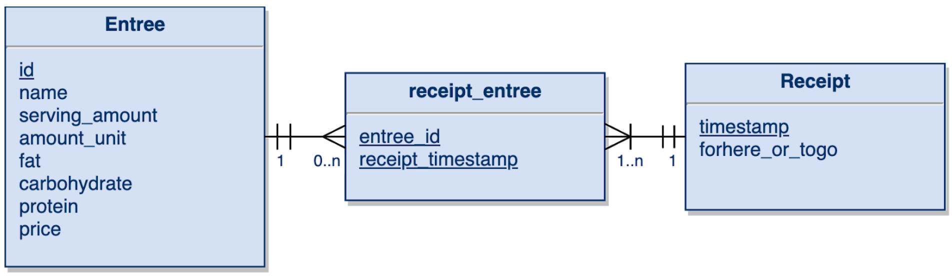


Fig. 4.5. The ERD for receipt storage in database.

Chapter 5 Implementation and Demonstration

After the system is well-designed from the last chapter, in this chapter, we will be implementing the design by first train the image recognition model for detection entrees in Sec. 5.1, evaluate the well-trained model in Sec. 5.2, complete price calculation and nutrition facts calculation in Sec. 5.3, and eventually, demonstrate the system in Sec. 5.4.

5.1 Training the Image Recognition Model

To perform training on a YOLO-structured model, there are some existing framework can be used. The major difference between them is that they are written in different programming languages or implemented with different libraries. We choose the frameworks coded by qqwweee, AlexeyAB and experiencor to work with since they are the top 3 most popular frameworks in GitHub.

5.1.1 qqwweee / keras-yolo3

This framework implements YOLO by two famous machine learning libraries in Python called Tensorflow and Keras. The maintenance was left uncared for since last year so it currently only supports Tensorflow version 1.6.0 along with Keras version 2.1.5. We followed the provided instructions throughout the training, however, this yielded no effective results.

In the first figure below, the training loss and validation loss plateaued, which indicates that learning is saturated. Unfortunately, when we fed a test image into the model, the model predicted entrees everywhere (see Fig. 5.2). If we look into details of these bounding boxes, almost all of them carry low confidences at nearly zero.

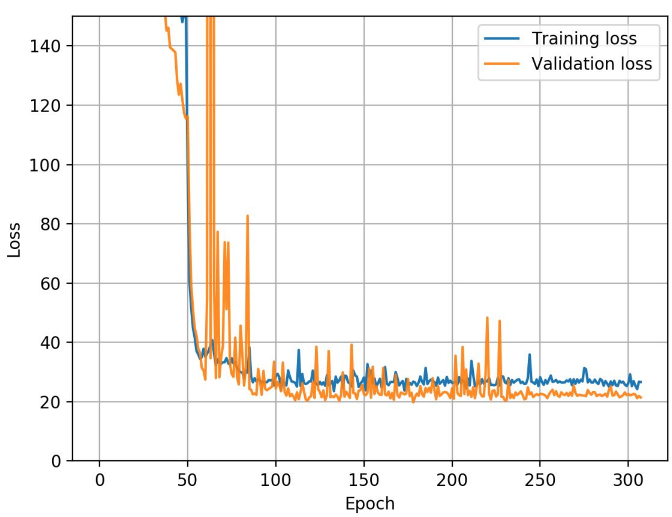


Figure 5.1. Training and validation losses while training with qqwweee framework.

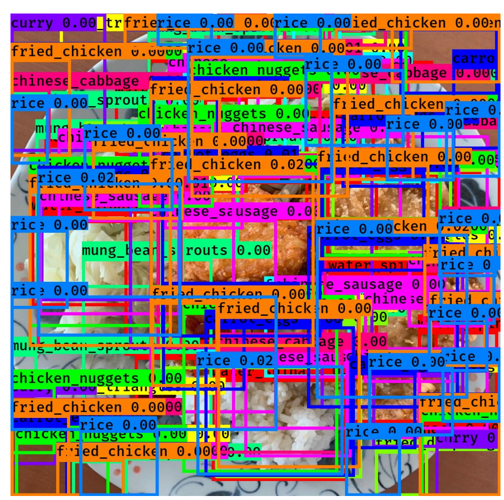


Figure 5.2. Testing the model with an image from testing dataset.

There were no additional improvement even though we readjusted the hyper-parameters in different ways. We also studied all the relevant issues opened in the GitHub relating to this framework and noted that we were not the only ones experiencing this particular issue. Ultimately, we decided on using another framework in order to implement our YOLO model.

5.1.2 AlexeyAB / darknet

The second framework we tried was Darknet which was written in C language by Joseph Redmon, the original YOLO author. To run the training with this framework, we had to build the C-language source code first. It was inconvenient that we must rebuild it again and again as long as we readjusted the hyper-parameters of the training.

Our training results show that the training loss decreases to 1 after around 1,400 iterations, with the mAP calculated on validation dataset fluctuates between 90% to 94% after 1,400 iterations (see Fig 5.3), which is thought to be precise enough to stop the training.

We then evaluated the model with the testing dataset. The mAP calculated from testing dataset is not as ideal as validation mAP, which is roughly between 34% to 35%. Finally, we decided to check out other frameworks to see if there is another better solution.

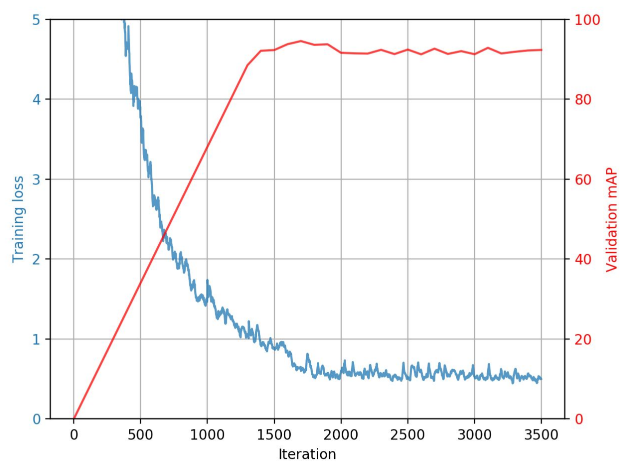


Figure 5.3. Training loss and validation mAP while training YOLO with Darkent.

5.1.3 experiencor / keras-yolo3

The third framework we tried is from the author experencor. The author's framework is based on Python with TensorFlow and Keras libraries too. But one noticeable difference we find out is that it updates the parameter of the model after training five times the entire training dataset in one epoch. (Usually, one epoch is interpreted as training through an entire training dataset once and updates the parameter once.)

In addition, we also change the way we trained by splitting the training dataset to 5 piles with 100 images each and training the model by incrementally adding new piles each time with the model set to terminate itself once the latest five validation loss rate converges (see Fig. 5.4).

This time, the result is outperformed that the mAP calculated from testing dataset (see Fig. 5.5) is nearly 70%, which is twice larger than the result from previous darknet framework. We of course adopt this framework and the trained model to continue building up our system.

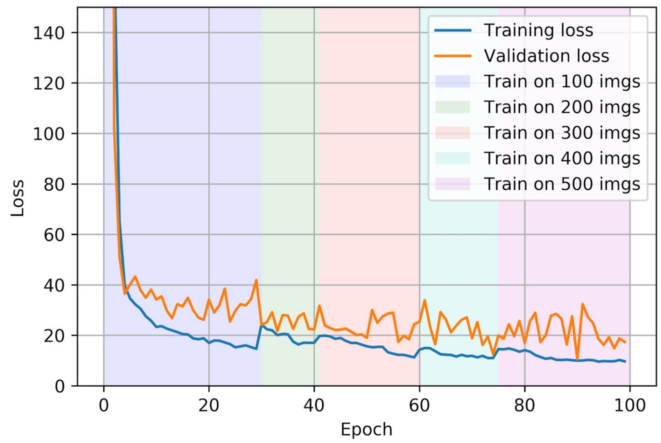
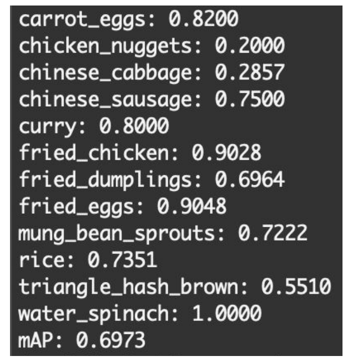


Figure 5.4. Training and validation loss while training with experiencor's framework.



```
carrot_eggs: 0.8200
chicken_nuggets: 0.2000
chinese_cabbage: 0.2857
chinese_sausage: 0.7500
curry: 0.8000
fried_chicken: 0.9028
fried_dumplings: 0.6964
fried_eggs: 0.9048
mung_bean_sprouts: 0.7222
rice: 0.7351
triangle_hash_brown: 0.5510
water_spinach: 1.0000
mAP: 0.6973
```

Figure 5.5. The mAP calculated from testing dataset.

5.2 Model Evaluation After Training

Besides the indicators training loss, validation loss and mAP from testing dataset, we also evaluated the model by the overall correctness of detecting every entree given a dish. The reason for this is to simulate the circumstances in actual checkout process that whenever any entree is falsely identified, the total price of the meal and nutrition facts are undoubtedly being calculated incorrectly. Below is the table showing the evaluation performed on testing dataset.

Table 5.1
Model Evaluation with Testing Dataset

Dish	Actual Existed Entrees	Model Detected Entrees	Pass/Fail
Appx. D Dish 1	• Rice • Triangle Hash Browns	• Rice • Triangle Hash Browns	Pass
Appx. D Dish 2	• Fried Chicken • Mung Bean Sprouts • Triangle Hash Browns • Water Spinach	• Fried Chicken • Mung Bean Sprouts • Water Spinach	Fail
Appx. D Dish 3	• Fried Chicken • Fried Dumplings • Rice • Water Spinach	• Fried Dumplings • Triangle Hash Browns • Water Spinach	Fail

Appx. D Dish 4	• Fried Chicken • Fried Dumplings • Rice • Water Spinach	• Fried Chicken • Fried Dumplings • Rice • Water Spinach	Pass
Appx. D Dish 5	• Chinese Cabbage • Chinese Sausages • Fried Chicken • Mung Bean Sprouts • Water Spinach	• Chinese Sausages • Fried Chicken • Mung Bean Sprouts • Water Spinach	Fail
Appx. D Dish 6	• Carrot Eggs • Chicken Nuggets • Fried Egg	• Carrot Eggs • Chicken Nuggets • Fried Egg	Pass
Appx. D Dish 7	• Fried Dumplings	• Fried Dumplings	Pass
Appx. D Dish 8	• Fried Dumplings	• Fried Dumplings	Pass
Appx. D Dish 9	• Fried Dumplings • Fried Egg • Rice • Triangle Hash Browns • Water Spinach	• Fried Dumplings • Fried Egg • Rice • Triangle Hash Browns • Water Spinach	Pass
Appx. D Dish 10	• Fried Dumplings • Fried Egg • Rice • Triangle Hash Browns • Water Spinach	• Fried Dumplings • Fried Egg • Rice • Triangle Hash Browns • Water Spinach	Pass
Appx. D Dish 11	• Fried Egg • Rice • Triangle Hash Browns • Water Spinach	• Fried Egg • Rice • Triangle Hash Browns • Water Spinach	Pass
Appx. D Dish 12	• Carrot Eggs • Fried Chicken • Fried Egg • Rice	• Carrot Eggs • Fried Chicken • Fried Egg • Rice	Pass
Appx. D Dish 13	• Carrot Eggs • Fried Chicken • Fried Egg • Rice	• Carrot Eggs • Fried Chicken • Fried Egg • Rice	Pass
Appx. D Dish 14	• Chinese Cabbage • Fried Chicken • Fried Egg	• Fried Chicken • Fried Egg • Rice	Fail

	• Rice • Triangle Hash Browns	• Triangle Hash Browns	
Appx. D Dish 15	• Chinese Cabbage • Fried Chicken • Fried Egg • Rice • Triangle Hash Browns	• Chinese Cabbage • Fried Chicken • Fried Egg • Rice • Triangle Hash Browns	Pass
Appx. D Dish 16	• Fried Egg • Rice • Water Spinach	• Fried Egg • Rice • Water Spinach	Pass
Appx. D Dish 17	• Fried Egg • Rice • Water Spinach	• Fried Egg • Rice • Water Spinach	Pass
Appx. D Dish 18	• Fried Egg • Rice • Water Spinach	• Fried Egg • Rice • Water Spinach	Pass
Appx. D Dish 19	• Carrot Eggs • Curry • Fried Egg • Water Spinach	• Carrot Eggs • Curry • Fried Egg • Water Spinach	Pass
Appx. D Dish 20	• Carrot Eggs • Curry • Fried Egg • Water Spinach	• Curry • Fried Egg • Water Spinach	Fail
Appx. D Dish 21	• Carrot Eggs • Curry • Fried Egg • Water Spinach	• Carrot Eggs • Curry • Fried Egg • Water Spinach	Pass
Appx. D Dish 22	• Chinese Cabbage • Chinese Sausages • Fried Chicken • Mung Bean Sprouts • Water Spinach	• Chinese Sausages • Fried Chicken • Mung Bean Sprouts • Water Spinach	Fail
Appx. D Dish 23	• Chinese Cabbage • Chinese Sausages • Fried Chicken • Mung Bean Sprouts • Water Spinach	• Fried Chicken • Mung Bean Sprouts • Water Spinach	Fail
Appx. D Dish 24	• Chinese Cabbage • Chinese Sausages • Fried Chicken	• Chinese Cabbage • Chinese Sausages • Fried Chicken	Pass

	• Mung Bean Sprouts • Water Spinach	• Mung Bean Sprouts • Water Spinach	
Appx. D Dish 25	• Water Spinach	• Fried Dumplings • Water Spinach	Fail
Appx. D Dish 26	• Water Spinach	• Fried Dumplings • Water Spinach	Fail
Appx. D Dish 27	• Fried Egg • Rice • Water Spinach	• Rice • Water Spinach	Fail
Appx. D Dish 28	• Chinese Cabbage • Mung Bean Sprouts	• Mung Bean Sprouts	Fail
Appx. D Dish 29	• Fried Chicken • Fried Egg • Water Spinach	• Fried Chicken • Fried Egg • Water Spinach	Pass
Appx. D Dish 30	• Curry • Fried Chicken	• Curry • Fried Chicken	Pass
Appx. D Dish 31	• Curry • Fried Chicken • Fried Egg • Rice • Water Spinach	• Curry • Fried Chicken • Fried Egg • Rice • Water Spinach	Pass
Appx. D Dish 32	• Carrot Eggs • Chicken Nuggets • Fried Egg	• Carrot Eggs • Chicken Nuggets • Fried Egg	Pass
Appx. D Dish 33	• Carrot Eggs • Chicken Nuggets • Fried Egg	• Carrot Eggs • Fried Egg	Fail
Appx. D Dish 34	• Carrot Eggs • Chicken Nuggets • Fried Egg	• Carrot Eggs • Fried Egg	Fail
Appx. D Dish 35	• Carrot Eggs • Chicken Nuggets	• Carrot Eggs • Chicken Nuggets	Pass

From the table above, we can see that there are 22 dishes being correctly detected meanwhile 13 dishes did not. The overall correctness of detecting every entree given a dish is roughly 63% (22/35). If we look into details of these falsely detected dishes, most of them are lacking one or two entrees rather than misclassifying entrees. Therefore, we believe that as we gather more and more data into training, this issue can be resolved.

5.3 Price Calculation and Nutrition Facts Calculation

For the calculation of the total price of the meal, we asked the staff at Feng-Lin restaurant the price for each individual entree. As for obtaining the nutrition facts of each entree, we referenced an app called “MyFitnessPal” [11]. This app was used to provide us the grams of carbohydrate, fat and protein (see Fig. 5.6) based on the weight. Next, we measured the weight of the entree per serving, using food scale that the cafeteria staff usually serves around and take the estimated average of entree for each (see Appx. A-C). For example, the weight of mung bean sprouts is approximately between 70 grams to 80 grams as Fig. 5.7 shows. With the weight of food and the information of nutrition per 100 grams, we can easily calculate the nutrition and calories of the meal.

Figure 5.6. Nutrition facts of mung bean sprouts provided by MyFitnessPal.

Figure 5.7. The calculated weight of mung bean sprouts from Feng-Lin cafeteria.

After getting all the prices and the nutrition facts, we store them into the config file used in our system. Whenever the webcam detects the entrees, our system retrieves the prices and the nutrition facts from the config file. They will then be calculated in order to display the results on the monitor.

5.4 Demonstration

After successfully making up our system, we performed several trial runs at the student restaurant. Figure 5.8 demonstrates an image taken by the webcam being fed to the system in real time. With Fig. 5.9 showing the prices and the nutrition of facts of each entree detected displayed. With our system, customers can easily learn the total amount of money to pay as well as how many fat, calories, and proteins they will consume.

Figure 5.8. The setup of our system in testing.

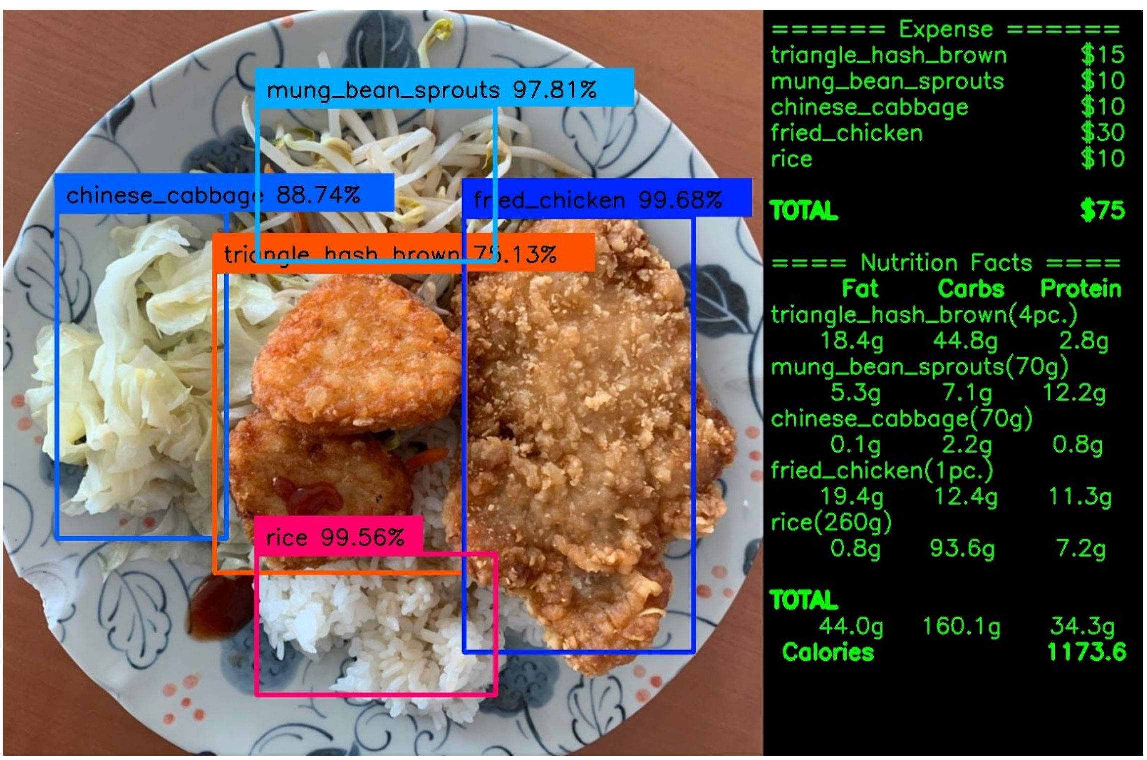


Figure 5.9. A screenshot from the system showing prices and nutrition facts.

Chapter 6 Discussion

In this chapter we will discuss some setbacks we encountered during implementation so that anyone who wishes to rebuild the system would not fall in the same trap.

6.1 The Way We Label Images

In the initial stages of training, we labelled Chinese sausages as many single pieces as possible. However, we find an interesting result that the model starts to draw bounding boxes around the single pieces even though they are adjacent even overlapped to each other, which is not what we aimed for (see Fig. 6.1). We only care about the category of entrees appear on the plate rather than how many pieces there are assuming the serving staff caters the same amount of pieces per serving on the plate.

We then changed the way we preprocess the data images by labeling them into a large singular box so that as long as they are adjoining they will all be labeled. The resulting test are as expected with the model only drawing one bounding box around a serving of Chinese sausages (see Fig. 6.2).

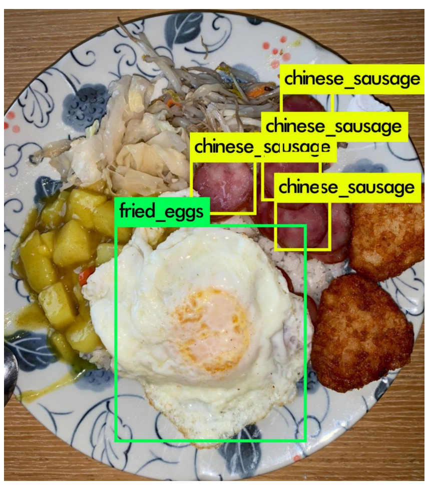


Figure 6.1. Chinese sausages are predicted individually.

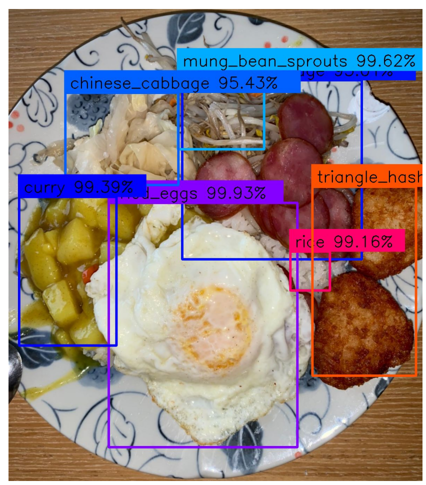


Figure 6.2. Chinese sausages are predicted as per serving.

6.2 CPU Speed - Google Colab

In the process of training our image recognition model, we used the student computer inside the department lab (CL519) at first. Unfortunately, due to the tremendous amount of floating point calculations during the back propagation step of the model, the computing power from the CPU is not enough. It takes an entire week to train the model only for about 1,000 iterations while running the darknet framework.

Fortunately, Google provides service called Colab [Sec. 2.6], a platform with free GPU service, for machine learning developers to train experimental models on cloud. With Colab, the training of 1,000 iterations using darknet framework only takes a night to complete. The graphical user interface is as friendly as Jupyter Notebook, so does the usage.

However, you can not both have your cake and eat it as there are still some restrictions with the use of Colab. The environments will be withdrawn periodically after running around twelve hours. In order to overcome this obstacle, we mount the Google Drive cloud disk with the environment so that we can save the partially trained model into it as to prevent loss.

Chapter 7 Conclusions

All good things must come to an end. In the last chapter, we will wrap up this project in Sec. 7.1, and list out the future works that we can improve our system in Sec. 7.2.

7.1 Conclusion

Although this system is still in the concept phase with the simplest prototype being demonstrated, we believe that the implementation of the system can positively reduce the waiting queue during chinese cafetera food checkout. We also believe that by addressing the two biggest crux: time and resources, we should be capable of establishing a more efficient and accurate model. Currently, image recognition is a prevalent field of study, and we are confident that given a few more years, coupled with the ever growing rate at which hardware develops, we should be more than capable in further optimizing our current system.

7.2 Future Work

Due to time constraints, we are not able to complete some of the requirements mentioned in Chapter 3. However, we do design them in Chapter 4 so the requirements are indeed feasible. Below are the parts being halfway done.

7.2.1 Storing and Analyzing Transaction Record

As the analysis of the requirements listed in Sec. 3.1.3, storing transaction records in the database is more flexible for retrieving and being further analyzed in the future. The analysis of the customer activity with those transaction records is helpful for the owner of the cafeteria. In the future, we will implement the functions of storing and analysis of those transaction records.

7.2.2 USB Accelerator

Since we are unable to run our system in embedded devices due to the great amount of floating point calculations inside the model in this period, we are willing to transplant our system onto it in the future. To achieve the goal, there is an existing product called Google Coral can be adopted. Google Coral is a tailor made accelerator for machine learning that allows embedded devices delegate the great amount of calculations to it to calculate. We believe by applying it, our system can be run in embedded devices.

7.2.3 Graphical User Interface

The graphical user interface of our system is very simple, which prints only the information with green font on the black background (see Fig. 5.10). It’s better to design an aesthetic graphical user interface for making our system more professional and high-quality.

References

[1] "image recognition" May, 2017. Accessed on: Dec. 18, 2019. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/image-recognition

[2] Prakhar Ganesh, "Object Detection : Simplified" Aug. 13, 2019. Accessed on: Dec. 18, 2019. [Online]. Available: https://towardsdatascience.com/object-detection-simplified-e07aa3830954

[3] Soham Chatterjee, "Different Kinds of Convolutional Filters" Dec. 20, 2017. Accessed on: Dec. 18, 2019. [Online]. Available: https://www.saama.com/blog/different-kinds-convolutional-filters/

[4] Yeh James, "[資料分析&機器學習] 第5.1講: 卷積神經網絡介紹(Convolutional Neural Network)" Dec. 25, 2017. Accessed on: Dec. 18, 2019. [Online]. Available: https://medium.com/jameslearningnote/資料分析-機器學習-第5-1講-卷積神經網絡介紹-convolutional-neural-network-4f8249d65d4f

[5] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, arXiv:1804.02767v1. [Online]. Available: https://arxiv.org/abs/1804.02767

[6] Waseem Rawat, Zenghui Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review" Sep. 2017. Accessed on: Dec. 18, 2019. [Online]. Available: https://www.saama.com/blog/different-kinds-convolutional-filters/

[7] "Gradient descent" Dec. 15, 2019. Accessed on: Dec. 18, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Gradient_descent

[8] "backpropagation algorithm" Nov., 2015. Accessed on: Dec. 18, 2019. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/backpropagation-algorithm

[9] "Project Jupyter" Dec. 10, 2019. Accessed on: Dec. 18, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Project_Jupyter

[10] "LabelImg" Aug. 18, 2019. Accessed on: Dec. 18, 2019. [Online]. Available: https://github.com/tzutalin/labelImg

[11] "MyFitnessPal" Dec. 11, 2019. Accessed on: Dec. 18, 2019. [Online]. Available: https://en.wikipedia.org/wiki/MyFitnessPal

Appendix A

Prices and Nutrition Facts of the Twelve Entrees

Entree	Amount per serving	Fat(g)	Carbohydrate(g)	Protein(g)	Price(NTD)
Carrot Eggs	70g	1.9	4.2	3.2	10
Chicken Nuggets	4pc.	12.0	12.0	9.0	15
Chinese Cabbage	70g	0.1	2.2	0.8	10
Chinese Sausages	24g (6pc.)	7.2	3.8	3.8	10
Curry	170g	7.1	16.5	10.7	10
Fried Chicken	1pc.	19.4	12.4	11.3	30
Fried Dumplings	4pc.	16.0	20.0	8.0	15
Fried Egg	1pc.	15.0	1.4	13.5	10
Mung Bean Sprouts	70g	5.3	7.1	12.2	10
Rice	260g	0.8	93.6	7.2	10
Triangle Hash Browns	4pc.	18.4	44.8	2.8	15
Water Spinach	70g	2.3	2.9	2.6	10

Source: 1. Nutrition facts are calculated from Appendix B and Appendix C and their amount per serving are approximated to the tens place. 2. Prices are documented from Feng-Lin Cafeteria (2019/12).

Appendix B

Sources of Nutrition Facts

胡萝卜炒鸡蛋 100 g 58 cal 36% 6 g Carbs 37% 2.7 g Fat 27% 4.5 g Protein Serving Size 100 g Number of Servings 1	4 Pc. Chicken Nugget Mcdonald's, 65 g 190 cal 25% 12 g Carbs 56% 12 g Fat 19% 9 g Protein Serving Size 65 g Number of Servings 1	Cabbage Chinese Cabbage, 100 g 16 cal 66% 3.2 g Carbs 9% 0.2 g Fat 25% 1.2 g Protein Serving Size 100 g Number of Servings 1
Carrot Eggs	Chicken Nuggets	Chinese Cabbage
香腸 黑橋牌, 100 克 398 cal 16% 16 g Carbs 68% 30 g Fat 16% 16 g Protein Serving Size 100 克 Number of Servings 1	Curry Curry, 100 gramm 102 cal 38% 9.7 g Carbs 37% 4.2 g Fat 25% 6.3 g Protein Serving Size 100 gramm Number of Servings 1	香脆雞腿排 卜蜂, 1 片 269 cal 18% 12.4 g Carbs 65% 19.4 g Fat 17% 11.3 g Protein Serving Size 1 片 Number of Servings 1
Chinese Sausages	Curry	Fried Chicken
Dumpling - Pan Fried Chinese, 1 dumpling 62 cal 31% 5 g Carbs 56% 4 g Fat 13% 2 g Protein Serving Size 1 dumpling Number of Servings 1	荷包蛋（油煎） 100 g 199 cal 3% 1.4 g Carbs 69% 15 g Fat 28% 13.5 g Protein Serving Size 100 g Number of Servings 1	炒豆芽 炒豆芽, 100 g 175 cal 23% 10.1 g Carbs 38% 7.6 g Fat 39% 17.4 g Protein Serving Size 100 g Number of Servings 1
Fried Dumplings	Fried Egg	Mung Bean Sprouts
白飯 白飯, 170 g 269 cal 91% 61.2 g Carbs 2% 0.5 g Fat 7% 4.7 g Protein Serving Size 170 g Number of Servings 1	三角薯餅 Generic, 56 g 90 cal 50% 11.2 g Carbs 47% 4.6 g Fat 3% 0.7 g Protein Serving Size 56 g Number of Servings 1	炒空心菜 100 g 48 cal 27% 4.1 g Carbs 49% 3.3 g Fat 24% 3.7 g Protein Serving Size 100 g Number of Servings 1
Rice	Triangle Hash Browns	Water Spinach

Source: MyFitnessPal (2019/12).

Appendix C

Measured Weight of Entrees

Entree	Entree put on Food Scale	Weight
Carrot Eggs		73g
Chinese Cabbage		70g

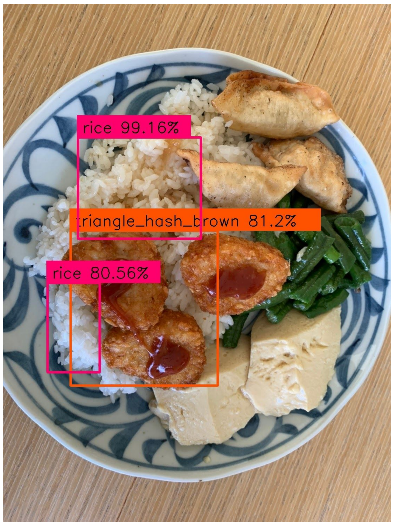
Chinese Sausages		24g (6pc.)
Curry		171g
Mung Bean Sprouts		72g

Rice		264g
Water Spinach		72g

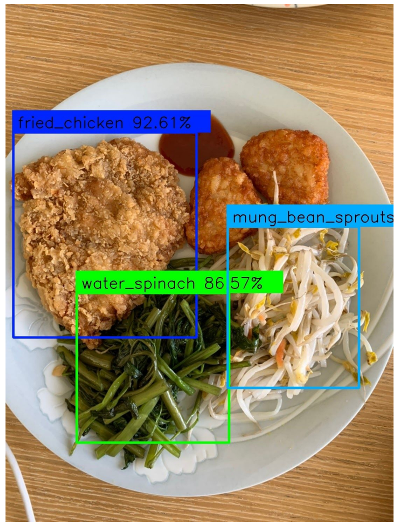
Source: Feng-Lin Cafeteria at TKU Lanyang Campus (2019/12).

Appendix D

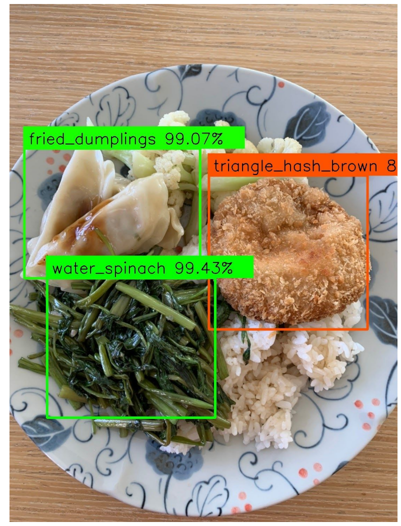
Model Perform Detection on Each Individual Dish in Testing Dataset



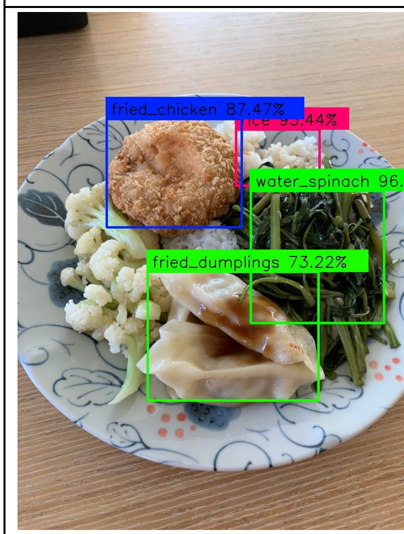
Dish 1



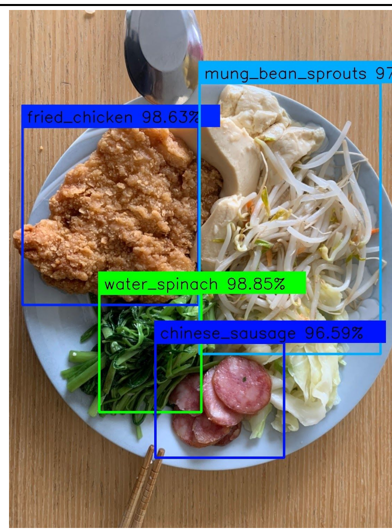
Dish 2



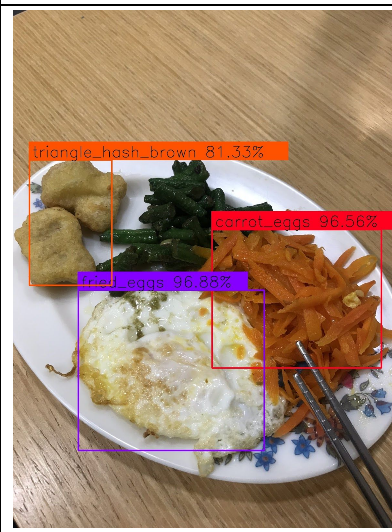
Dish 3



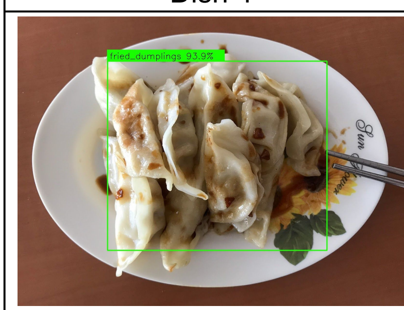
Dish 4



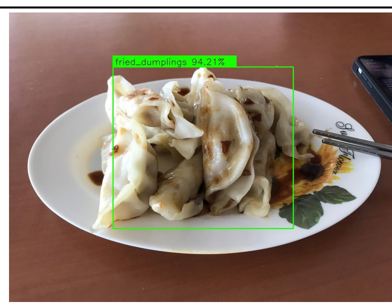
Dish 5



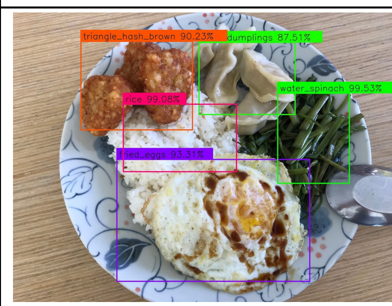
Dish 6



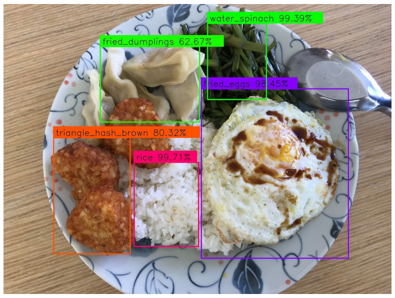
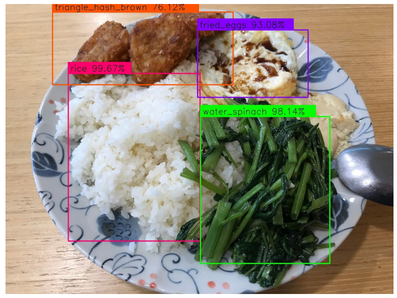
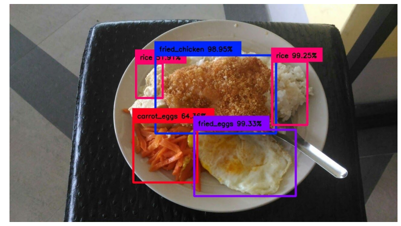
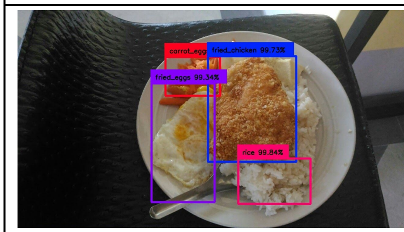
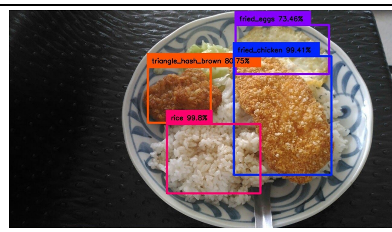
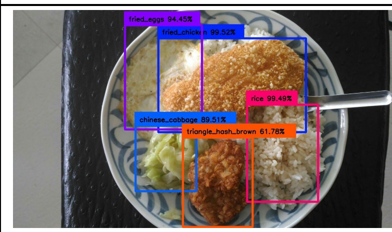
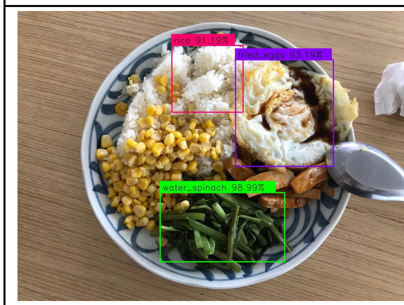
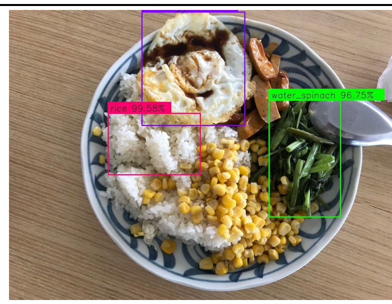
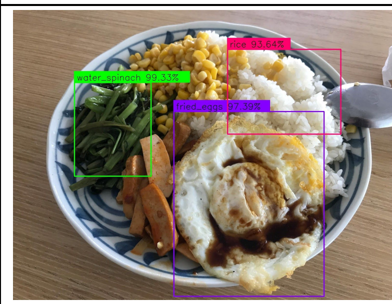
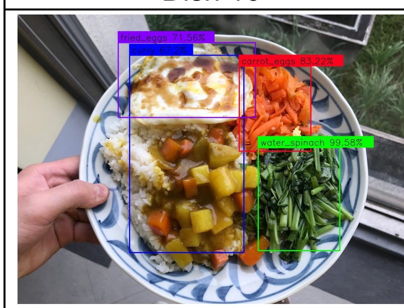
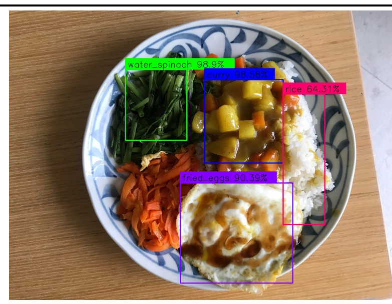
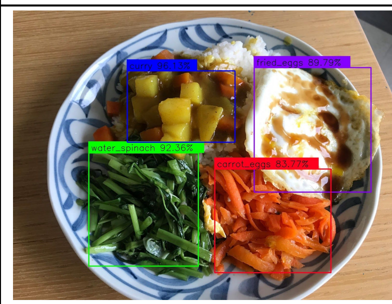
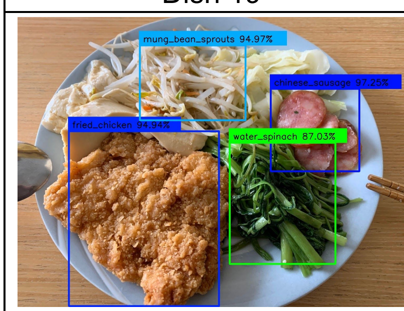
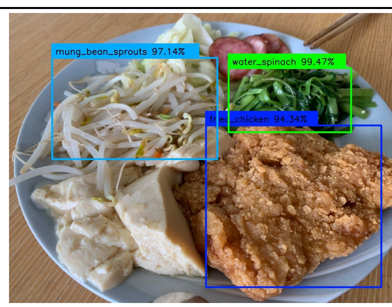
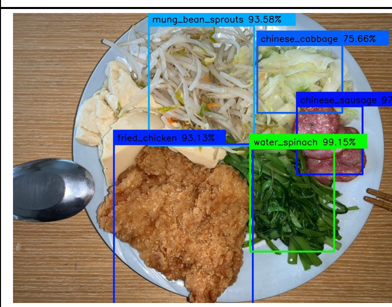
Dish 7

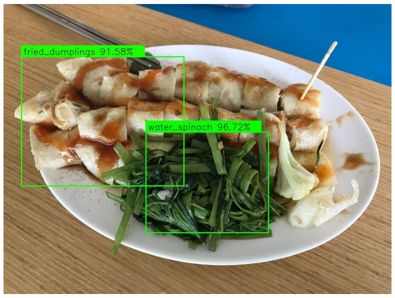


Dish 8

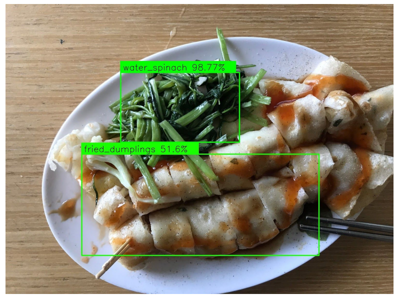


Dish 9

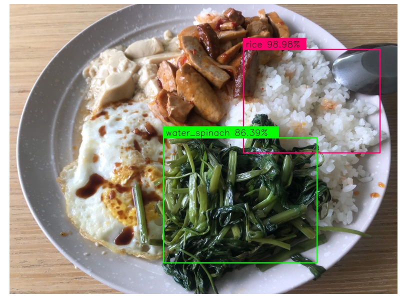
		
Dish 10	Dish 11	Dish 12
		
Dish 13	Dish 14	Dish 15
		
Dish 16	Dish 17	Dish 18
		
Dish 19	Dish 20	Dish 21
		
Dish 22	Dish 23	Dish 24



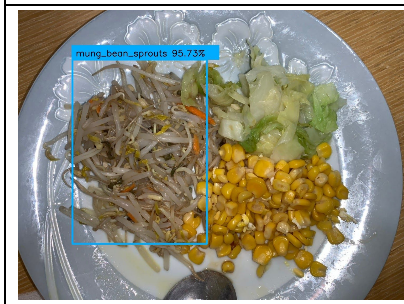
Dish 25



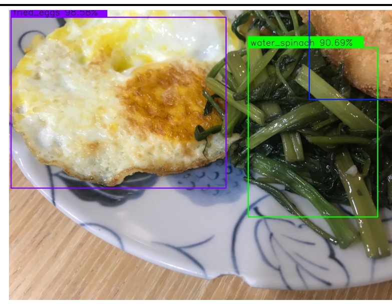
Dish 26



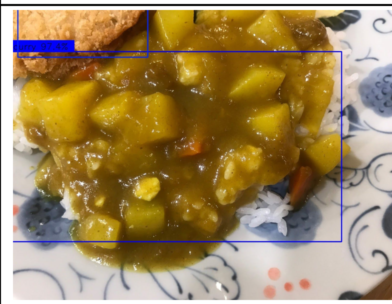
Dish 27



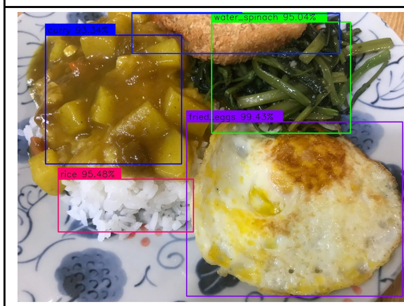
Dish 28



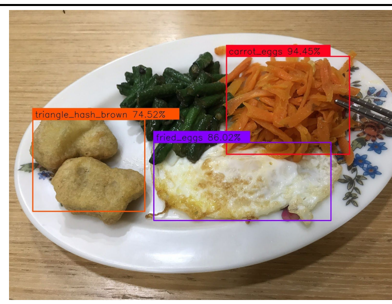
Dish 29



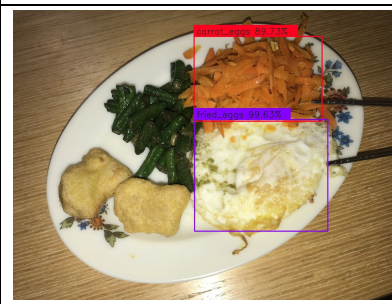
Dish 30



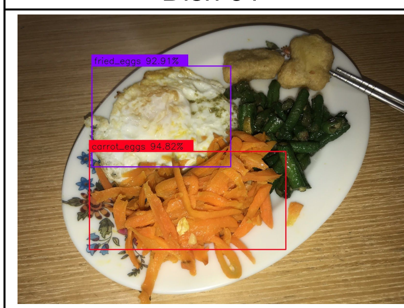
Dish 31



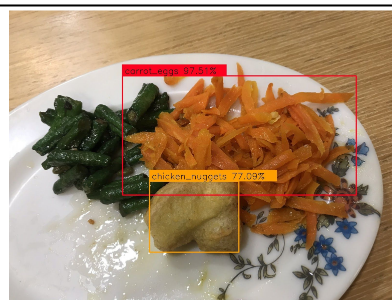
Dish 32



Dish 33



Dish 34



Dish 35