VISION BASED CALORIES COUNTER

BY

YEOH KOK SIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

MAY 2018

REPORT STATUS DECLARATION FORM

Title: _____

_____

_____

Academic Session: _____

I _____

(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.

Verified by,

_____ _____

(Author's signature) (Supervisor's signature)

Address:

_____

_____ _____

_____ Supervisor's name

Date: _____ **Date**: _____

VISION BASED CALORIES COUNTER

BY

YEOH KOK SIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

MAY 2018

DECLARATION OF ORIGINALITY

I declare that this report entitled "**VISION BASED CALORIES COUNTER**" 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 : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Lau Phooi Yee who has given me this bright opportunity to engage in this project. Her willingness to spare her time in assisting and providing guidance to me are truly appreciated. It is my first attempt in the field of deep learning and my first step to establish my career in the field.

Finally, I wish to thank my family for their support and encouragement throughout the time.

ABSTRACT

This project proposes a framework to compute calorie from classified dessert in an Android application based on deep learning. Transfer learning was being applied in order to train a convolutional neural network model that is able to identify a total of 30 kinds of dessert including both western and local. The main benefits of deep learning is that it eliminates the need of feature engineering and proven to excel in performance when compared to traditional image classification techniques. Nonetheless, TensorFlow Inference API made it possible for trained model to be used in local mobile device that usually has restrictions in terms of memory, computing power and storage space. This project used MobileNetV2 that was pre-trained on ImageNet dataset. Since ImageNet dataset consist of 1000 classes, these learnt features can be reused in solving other similar problems. Last few layers of MobileNetV2 was being trained while the others were being frozen to preserve the learnt general features. Besides, last layer was modified to classify 30 classes instead of 1000. The model was being trained on new datasets consisting 30 classes of dessert images. Then, the final trained model are bundled into the Android application and interacted through TensorflowInferenceInterface API. Every time a raw image captured from phone camera, it is processed and feed into the model to get predictions. The application have two modes, photo and real-time that can be achieved by applying Camera 2 API. Lastly, the final prediction is determined by class with highest score and its relevant information are retrieved from a SQLite database created. After computation of calories, the results are displayed to user. Finally, the project model achieved an accuracy of approximately 95.25%.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
BMI	Body Mass Index
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HTML5	Hypertext Markup Language revision 5
KNN	K-Nearest Neighbors
RAM	Random Access Memory
RGB	Red, Green and Blue
RNN	Recurrent Neural Network
SURF	Speeded Up Robust Feature
SVM	Support Vector Machine
VRAM	Video Random Access Memory

CHAPTER 1 INTRODUCTION

1.1 Problem statement

Problems of overweight and obesity in Malaysia is worsening (Economist Intelligence Unit, 2017) and it is directly related to uncontrolled calorie intake. Desserts could be the one to be blamed. The psychological sugar-craving (Conason, 2012) is the reason of desserts high popularity. While people are enjoying their desserts, they might not be aware of the high calorie content of it. Desserts could be calorie bombs and if they are being over-consumed, it will have a negative impact on our body such as obesity and high blood pressure due to the high saturation of sugar. Moreover, some local or traditional desserts information are not very well documented and therefore their details including calorie content are hard to be identified.

1.2 Background and motivation

Malaysia has been long associated with problems of obesity and overweight. According to a recent study from Economist Intelligence Unit (2017), Malaysia topped the chart of overweight rate among other South-East Asian countries with a rate of 13.3%. Likewise, the same trend could be seen in the chart of obesity rate where Malaysia leads with a rate of 38.5%. The charts of overweight and obesity prevalence among several ASEAN countries are shown in **Figure 1-1** and **Figure 1-2**. Likewise, the overall increase in number of obese people from 2010 to 2014 is estimated to be 27% (Economist

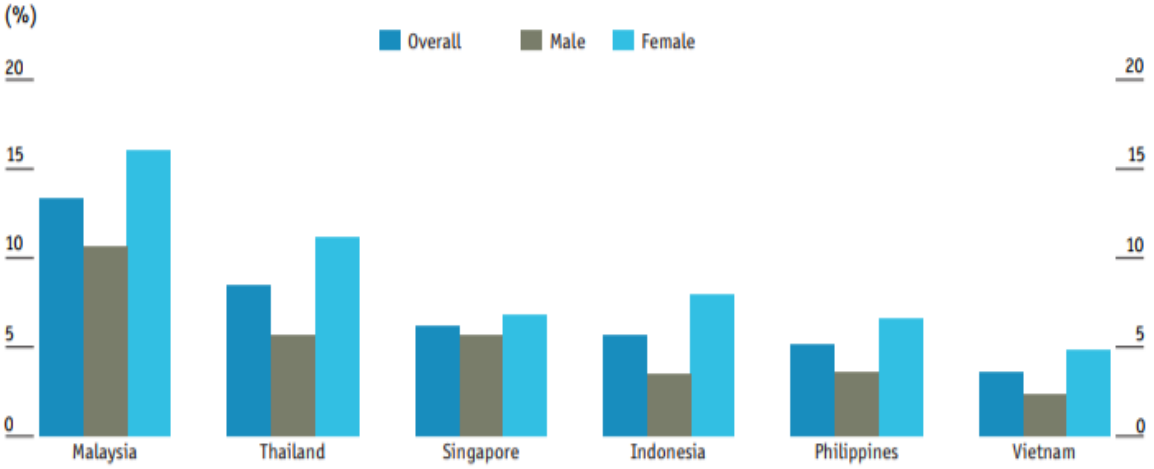


Figure 1-1 Obesity prevalence (BMI >= 30) age-standardised adjusted estimates, adults > 18 (Economist Intelligence Unit, 2017)

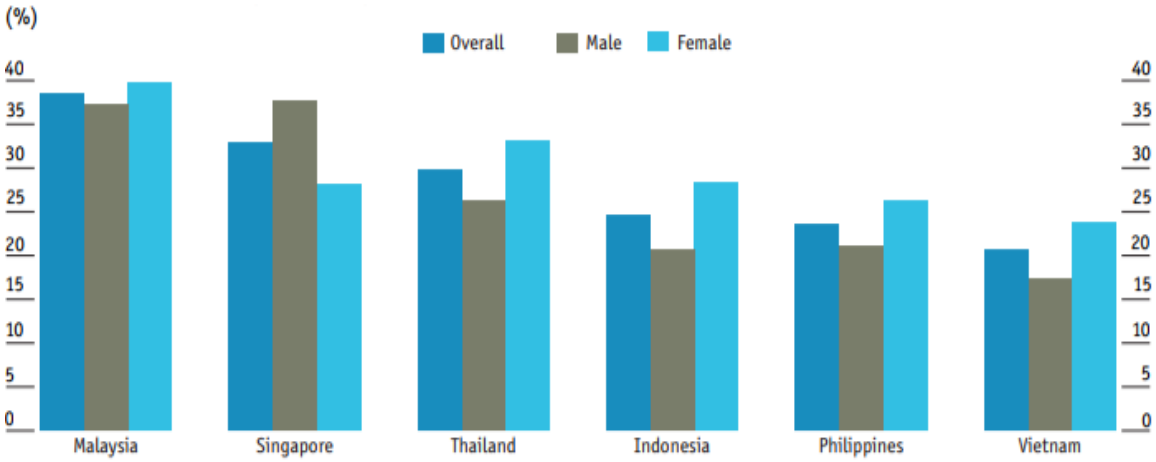


Figure 1-2 Overweight (BMI >= 25) age-standardised adjusted estimates, adults > 18 (Economist Intelligence Unit, 2017)

Intelligence Unit, 2017). In fact, Malaysia is currently catching up with western countries such as United States and United Kingdom in terms of obesity and overweight rates among the adult population while leading other South-East Asian countries. A table of obese adult percentage is shown in **Table 1-1**. These heartbreaking statistics is not only a warning sign to Malaysian people but also among people in other countries.

Table 1-1 Obesity prevalence in ASEAN country sample and selected other countries (Economist Intelligence Unit, 2017)

ASEAN country sample			
Country	**Percentage obese adults (%)**		**Increase in number of obese people (%), 2010 - 2014**
	2010	**2014**	
Indonesia	4.3%	5.7%	33%
Malaysia	10.5%	13.3%	27%
Philippines	4.1%	5.1%	24%
Singapore	5%	6.2%	24%
Thailand	6.7%	8.5%	27%
Vietnam	2.6%	3.6%	38%
Comparator countries			
Japan	2.9%	3.3%	14%
South Korea	4.2%	5.8%	38%
United Kingdom	25.5%	28.1%	10%
United States	31.2%	33.7%	8%

Body mass index (BMI) can be calculated by the formula *BMI = (Weight in kg / Height in $m^{2)}$*. According to World Health Organization (2016), BMI >= 25kg/m^2 is considered overweight and BMI >= 30kg/m^2 is considered obese for adults aged above 18. An example of BMI range and their categories is shown in **Figure 1-3**. As stated by Haslam and James (2014), obesity could be linked to serious diseases such as cardiovascular disease, diabetes, and cancer. These are serious problems that need be paid attention as not only it has direct impact on life expectancy but also quality of life.

Figure 1-3 Body mass index categorised range (Categorised Body Mass Index, 2017)

One of the factor contributing to overweight and obesity problems is directly related to excessive calorie intake. Not to miss out that insufficient calorie intake could also lead to underweight problems which is also undesired. According to Gunnars (2017), the daily calorie intake for average male is 2,500 calories while 2,000 for average female. However, these standards might not be suitable for anyone as there are several factors need to be considered aside from gender. These factors are age, height, weight and also activity level which indicates how often a person exercises in a week. Therefore, daily calorie intake for each person will be different and in order to maintain a good health, people must be aware of their daily calorie intake and manage it properly.

Moving on, Malaysia is a highly multicultural country and there are great variety of food available. However, most of the Malaysian's favourite food are actually high-

calorie such as nasi lemak and roti canai as seen in **Figure 1-4** and **Figure 1-5**. This is because of the high content of fats and sugar. For instance, according to MyFitnessPal (2017), a pack of nasi lemak with serving size of 230 grams contains 348 calories while

Figure 1-4 Nasi lemak (Satu Johor, 2016)

Figure 1-5 Roti canai (Plain Roti Canai, 2017)

single slice of plain roti canai consists of 302 calories. Both of the dishes are shown in Figure 5 and 6 respectively. Of course, these dishes would normally not be served alone. There is other food like peanuts, eggs and drinks to be considered.

Regardless of that, there are also various types of traditional dessert available. Indeed, the sweetness of dessert comes from its high amount of sugar content. Despite being a great source of energy, excessive sugar intake will be converted into body fats in order to be stored. As an example, traditional desserts are usually made up of coconut oil, rice and sugar. Thus, mixing up these ingredients contribute to a lot of calorie content. Some of the examples of Malaysian dessert are seri muka, appam balik which can be seen

Figure 1-6 Appam balik (Auria, 2012)

Figure 1-7 Kuih seri muka (Law, 2016)

in **Figure 1-6** and **Figure 1-7**. For instance, a piece of appam balik with service size of 120 grams and a piece of kuih seri muka with service size of 99 grams would be 290 and 192 calories respectively. All in all, regardless of traditional or western dessert, their similarity is always its high amount of sugar content.

According to World Health Organization (2003), daily sugar intake should not exceed 10% of total energy which means it should be within 50 grams. Following by American Heart Association (2009) which states that daily added sugar intake should be within 100 calories for females and 150 calories for males. However, regardless to rising price of sugar, Malaysian sugar intake increased by 33% between early 1960 and 2005. This could be associated with Malaysian's deep affection towards dessert. In addition, certain western desserts are also calorie bombs. For example, according to (Johnson et al., 2009), one NLEA serving of a plain, commercially prepared cheesecake as seen in **Figure 1-8** weights 125 grams and has 401 calories. Not just that, most of the western desserts are made up of ingredients such as butter, eggs etc. that contains heavy calorie.

Figure 1-8 Cheesecake (Johnson, 2013)

Historically, there had been several proposed solutions to counter the problem of overweight and obesity. One of the proposed system could be personalised messaging system based on crowdsourcing as proposed by Moorhead et al. (2015). It applies the idea of crowdsourcing as a mobile application where food photographs are taken and uploaded for a group of nutritionist expert to estimate the calorie content. Then, Hariadi et al. (2015)

proposed a food nutrition information system as a mobile application with the help of an application programming interface (API). The food photograph taken are sent to a server for food classification and retrieve nutritional information through the API. Similar to previous approach, Tammachat and Pantuwong (2014) implement the system as a service on the web instead. This allows device independency hence any device with a browser are able to access the system. Not to miss out that, there is also a real-time mobile food recognition system as proposed by Kawano and Yanai (2013). It allows user to acquire food calorie content as quick as possible regardless of internet connection.

Recent rapid progression in development of smartphone hardware had made performing heavy computation in the device locally possible. Besides, deep learning had been the state-of-the-art in the field of image classification as it excels in accuracy when compared to the others. At the same time, smartphones are getting more common and affordable for everyone. By applying deep learning, a mobile device could be a useful tool to solve the problems of overweight and obesity. It provides convenience for people of all age to maintain their daily calorie intake and also gain better understanding towards their favourite dessert. In a nut shell, deep learning can be implemented into a mobile application that is capable of identifying food and estimating calories based on inputs from phone camera itself.

This project is to propose a mobile application framework to estimate food calorie in two modes, based on a photograph or in real time. It will use apply deep learning instead of classical or traditional machine learning methods due to recent advancement of deep learning in terms of accuracy and reliability in the field of image classification. According to Caspi (2017), 'Deep neural networks are the first family of machine learning that do not require manual feature engineering. Instead, high-level features were learnt from raw data. Besides, raw pixels are fed in to the network where 20-30 percent improvement in accuracy was seen in most computer vision benchmark'. With that being said, Convolutional Neural Network (CNN) will be used in this project as opposed to Support Vector Machine (SVM) which most of the past approaches were built on for food classification.

1.3 Project objectives

The project's main objective is to propose a framework to estimate food calorie for different kind of desserts by applying deep learning. It focus on well-known desserts which also includes local or traditional desserts. Second objective is to develop a functional prototype for the proposed framework as an Android mobile application. The prototype will have photograph and video modes that can deliver results in real-time.

1.4 Proposed approach / study

The project applied deep learning as a backbone of the framework to classify frames and used the predicted results to compute dessert calories. Besides, the project also applied popular technique named transfer learning where pre-trained model was used as a starting point for model training instead of re-designing a new model architecture. Nonetheless, only last few layers of the model was being trained in order to preserve and reuse features that were being learnt from previous dataset (ImageNet). Then, dessert information including names and base calories were built and embedded into application to allow offline usage. Each frames captured from phone camera will be used as input to the model and the highest output scores will be used to compute calories.

1.5 Highlights of achievements

- The prototype managed to correctly classify between 30 different kinds of dessert presented in front of camera and telling its relevant calorie
- The model achieved an estimated accuracy of approximately 95.25%
- The speed of model is fast enough which took approximately 450 milliseconds per frame which simulates real-time although it can still be improved
- The size of final prototype application package is small enough (~150MB) which extends its compatibility across different types of device models

1.6 Report Organization

The details of this project are shown in the following chapters. In Chapter 2, some literature reviews and experiments are done. Then, project overview and flows are shown

in Chapter 3. Furthermore, more information such as methodologies, tools etc. are listed in Chapter 4. Finally, Chapter 5 concludes the project and suggested some possible future works that can be done.

CHAPTER 2 LITERATURE REVIEW

There is a total of six papers being reviewed. Among them are existing systems and solutions to tackle the problem of difficulty in acquiring food calorie. Indeed, these papers focus on estimating food calorie based on a food photograph. The first paper discussed about applying the idea of crowdsourcing as a mobile application for user to upload food photograph in order for nutritionist experts to estimate its calorie content. The second paper proposed a system that uses API to acquire food nutritional value after successful food classification. The third paper proposed the calorie estimation system as a web service. The fourth paper introduces real-time food calorie estimation. Finally, the fifth and sixth paper applies Convolutional Neural Network (CNN) for food and traffic sign classification respectively.

2.1 Smart Food: Crowdsourcing of experts in nutrition and non-experts in identifying calories of meals using smartphone as a potential tool contributing to obesity prevention and management

Moorhead et al. (2015) proposed a solution to solve the problem by developing a personalised messaging system for a mobile application that is based on crowdsourcing, where calorie content of meal is determined by a group of nutritionist experts based on a photograph given. The architecture of the proposed application is shown below as **Figure 2-1**.

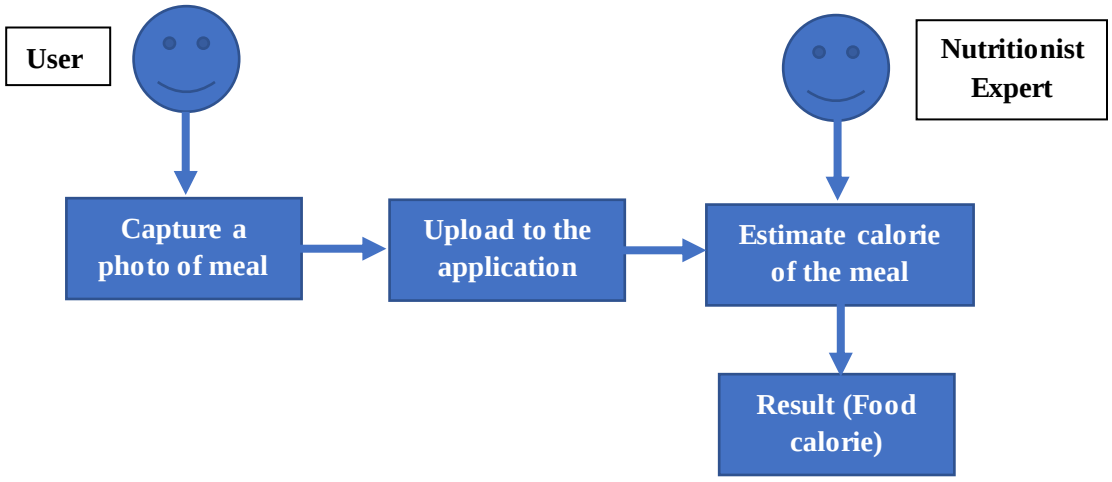


Figure 2-1 Framework of the SmartFood

This paper attempt to prove the benefit of crowdsourcing and how it could be used for an application that could help to prevent and manage obesity. To do that, an experiment is done on a group of 12 nutritionist experts and 12 non-experts by using online survey. Each group are required to estimate calories for 15 meals for twice, so there was a total of 720 estimates being made *(15 meals * 24 persons * 2 instances)*.

The results from the experiment shown that the group of nutritionist experts' calorie estimation outperformed group of non-experts in terms of percentage differences between estimated calorie count and actual calorie count, which were +8% and +55% respectively. Besides, the studies also found that the accuracy of calorie estimation from a crowd is higher than it is from an individual.

According to Ranard et al. (2013), 'crowdsourcing is an approach to accomplishing a task by opening up its completion to broad sections of the public'. That statement correlates with the proposed solution with one of its strength is that it allows users to get access to nutritionist and perform food logging in a much simpler way. Other than estimating calorie content of food taken from photograph, nutritionist could also provide opinions and suggestions to user regarding to calorie intake. This allow user to have better understanding on food nutrition. Also, Wooldford et al. (2010) concludes that 'computerised tailored text messaging is a feasible adjunct to multidisciplinary obesity treatment'. That statement further strengthen that the proposed solution has potential to be realised.

However, there exist a few weaknesses from the proposed solution. Firstly, the idea of crowdsourcing is inefficient and time-consuming. As being said, user would capture a photograph of their meal, upload to the application and wait for someone to reply with an estimated result. There is no guaranteed time of arrival for the answers as it only depends on willingness for someone to spend their time in computing the results and posting a reply. Secondly, the results could be inaccurate as any of the experts are able to reply with an answer and uncertainties might arise. The overall process relies on lots of manual input and calculation which is undesired and inconvenient. Besides, the answers might be inconsistent among each other. Thus, results from computing the mean from the answers will be affected and enlarges the error ratio. Nonetheless, the method proposed solely relies

on presence of nutritionist experts. The application would not function if there is absent or lack of nutritionist experts accessing it.

In a nut shell, this paper proposed a solution that utilize crowdsourcing to achieve its goal to manage and prevent obesity. The solution stressed about estimating food calorie by human manually based on photograph taken. However, certain limitations exist in this solution as is over-reliance on human interaction with the system where it depends mostly on manual input. To overcome this problem, an automated calorie estimation system will be a better way for users as it is much simpler and convenient to use.

2.2 Design and Implementation of Food Nutrition Information System using SURF and FatSecret API

Hariadi et al. (2015) proposed a food nutrition information system as a mobile application. The proposed application will take a food photograph as input and output its nutrition information to user. Among the returned nutrition information includes calorie, fat, carbohydrate and protein per serving. For the architecture used, user first take a photograph of their food. Then, the photograph is being sent to a server that will try to recognize the food and identify its name. Firstly, Speeded Up Robust Features (SURF) (Bay et. Al, 2008) is used in the process of image recognition. Next, the classified name is used to get nutrition information via FatSecret API (FatSecret Platform API, 2017) and the

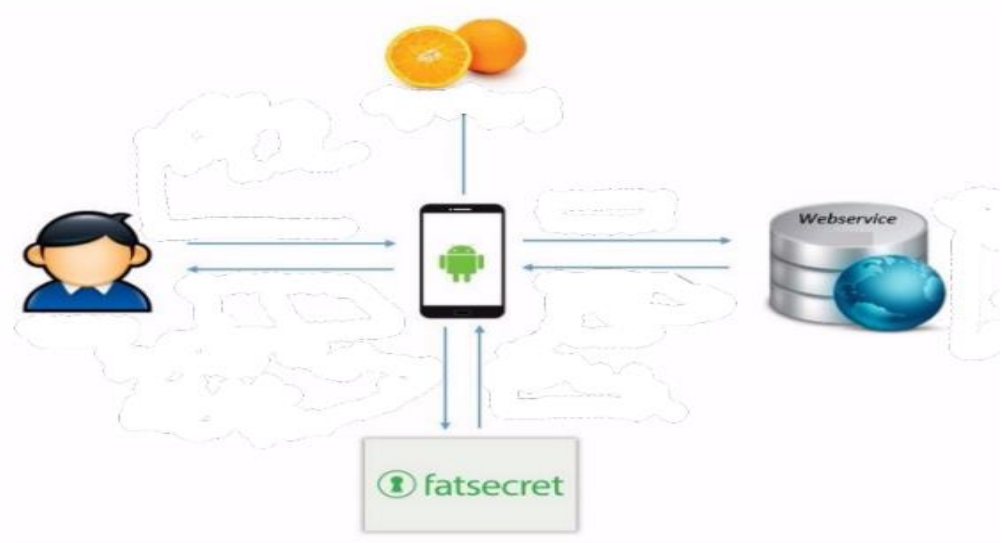


Figure 2-2 Framework of Food Nutrition Information System (Hariadi, 2015, pg. 181)

result is then displayed to user. The framework for the proposed application is shown below as **Figure 2-2**.

Using SURF, the keypoint features are extracted from image and compare with dataset by using Hessian matrix (Bay, 2008, pg.3) as shown in **Equation 1** below. In addition, all of the symbol descriptions are shown in **Table 2-1**.

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \quad \textbf{(1)}$$

Table 2-1 Description of symbols for Hessian matrix

Symbol	Description
x	a point (x, y) in the image
σ	scale
$L_{xx}(x, \sigma)$	convolution of Gaussian second order derivative

Then, these keypoints will be used as the image descriptor. An example of an image of lemon with its keypoints is shown in **Figure 2-3** below.

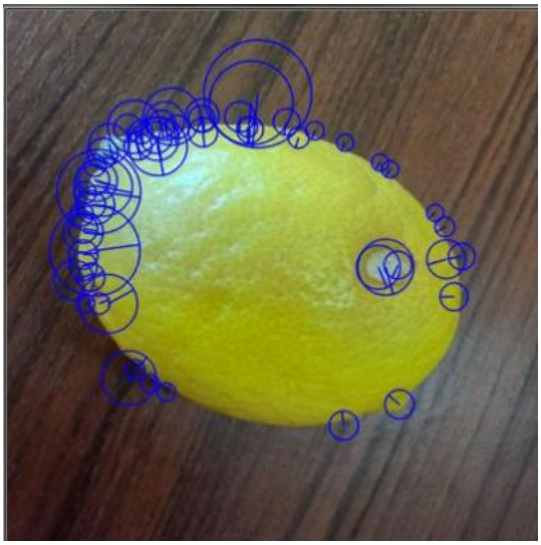


Figure 2-3 An image of lemon with its keypoints (Hariadi et al. 2015)

Finally, the similarity of keypoints is used to classify the objects compared with the dataset by calculating the Euclidean distance between two vectors.

An experiment is conducted on correct recognition rate of 10 different objects such as orange, banana etc. 20 unique photographs of each object are captured and tested with the application. In the end, the experiment achieved an average accuracy of 92% which is satisfiable.

One of the strength of the proposed system is the usage of FatSecret API (FatSecret Platform API, 2017) which provide access to their food and nutrition database. It provides complete access to their large database of food and its nutritional information. Also, the information retrieved are guaranteed to be accurate as all of the data are verified. Secondly, since that food classification are done in a server rather than the mobile device itself, this reduce the computational burden on the device. Thus, the phone specification requirements are significantly lowered as the application doesn't need to perform any heavy computation. Moreover, SURF is simple to implement, fast and requires smaller training datasets compared to Convolutional Neural Network (CNN) which is used in the project.

However, the system mostly relies on server to function properly. This prevents the system from being interactive due to absence of internet connection or communication delays. Likewise, the server used for image processing and food classification is a single point of failure. This renders the system to be non-functional in case the server fails. Then, there is no image segmentation performed in the server. This might reduce the overall accuracy of food classification as the background of the food image is not removed. An example of image segmentation can be seen in **Figure 2-4**.

Figure 2-4 Left image is the original image. Right image has its background removed by image segmentation

Nonetheless, FatSecret API is not an entirely free of charge service. There are several versions available and the basic version offers only up to 5000 API calls per day as well as imposing throttling limit (FatSecret Platform API, 2017). That means that the application will not be able to retrieve any calorie information after maximum limit of API calls is reached.

To conclude that, the proposed system is tackled the problem of obesity by simplifying the process of acquiring food calorie that is to estimate food calorie from a photograph taken. Next, the system is unable to function under bad or absence of internet connection. Regarding to improvement to be done, it should be able to function regardless of internet connection and perform image processing such as noise removal and image segmentation to increase the accuracy of food classification.

2.3 Calories Analysis of Food Intake Using Image Recognition

Tammachat and Pantuwong (2014) proposed a system that estimate calorie based on a food photograph taken. As discussed previously, several approaches to tackle the problem of obesity regarding to calorie intake is being proposed. Instead of applying the idea of crowdsourcing which is proposed by Moorhead et al. (2015) or acquiring information via an API as proposed by Hariadi et al. (2015), Tammachat and Pantuwong (2014) takes a different approach by applying the concept of calorie estimating as a service on the web.

The framework of the system is shown in **Figure 2-5** below. User will upload their

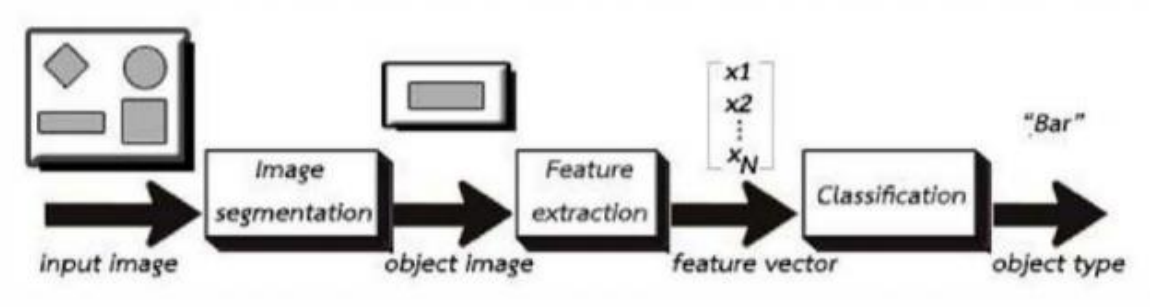


Figure 2-5 Framework of the proposed system by Tammachat and Pantuwong (2014)

food photograph onto the web server using browser. Then, the food portion is segmented

Figure 2-6 Example of image segmentation based on texture. Flow of the process is from left to right. (Tammachat and Pantuwong 2014)

out from the whole image based on texture segmentation. An assumption that table and dish have no texture is being made. One of the example of image segmentation is shown in **Figure 2-6** below. Furthermore, the segmented image is cropped as seen in last image from **Figure 2-6**. Then, feature vectors are extracted by Bag-of-Features (BOF), Segmentation based Fractal Texture Analysis (SFTA), and Colour Histogram. Finally, the feature vectors are used for food classification by Support Vector Machine (SVM) model with pre-trained datasets. Feature vectors with dimensions of 1x500, 1x24 and 1x30 are constructed respectively. Finally, a feature vector of 1x554 dimension *(500 (BOF) + 24(SFTA) + 30 (Colour Histogram))* is created for each input image.

An experiment is performed on two kinds of SVM model. First model where constructed by food images grouped by food type achieved an average accuracy of 70%. The second model where constructed by food images grouped by range of calories achieved a slightly unsatisfying result with average accuracy of 46.4%. The paper concludes that first model is indeed better in terms of performance against second model.

One of the strength of the proposed system is being device independent. This is because it implemented as a service on website using HTML5 instead of a mobile application. Be it a desktop or smartphone, almost any devices can access the service via a browser. Furthermore, device requirements will not a problem since that all image processing operations and calculations are performed in a server. Nevertheless, the system will also record down user's calorie intake behavior and try to provide suggestions to use.

Despite all of the strengths that this system has, the system is not functional without internet connection. This is because all of the processes are done in a server. In the cases of network breakdown or server breakdown, it will render the whole system unfunctional.

All in all, the proposed system tackled the problem of obesity by estimating calorie based on a photograph uploaded by user. Instead of performing food classification using SVM, Convolutional Neural Network (CNN) could be used to solve the problem of food classification. Since there is no manual extraction of feature vectors. It is more general and able to be applied to any kind of food. A CNN will be trained using TensorFlow in the proposed project. Then, the model will be retrained through a process called Transfer Learning to fit on the project's aim, which is to classify and estimate dessert's calorie.

2.4 Real-time Mobile Food Recognition System

Kawano and Yanai (2013) proposed a mobile food recognition system. The system is capable to estimating food calorie by simply pointing device camera towards it. Users are required to draw a bounding box around each food portion for the activate calorie estimation. Nonetheless, a slider is provided for user to indicate the size of the food portion. A screenshot of the proposed system is shown in **Figure 2-7** below.

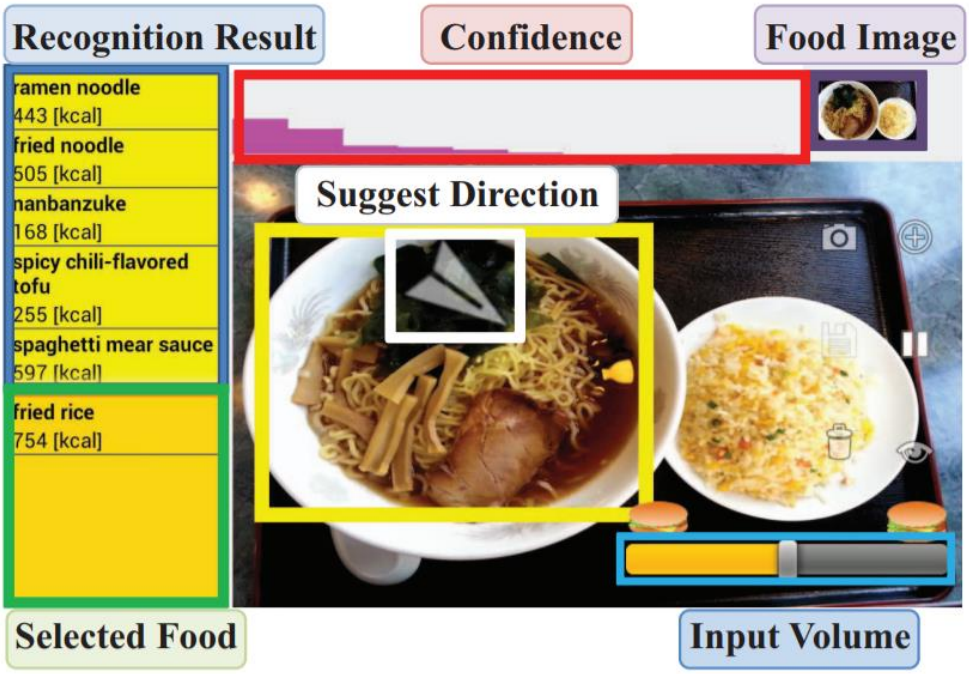


Figure 2-7 Screenshot of the proposed system (Kawano and Yanai, 2013)

First of all, the proposed system harness use of GrabCut (Rother, Kolmogorov and Blake, 2004) to perform food portion segmentation. It segments out food portion from image background based on bounding boxed drawn by user. Then 50 categories of food

are trained by Linear Support Vector Machine (SVM). Next, Colour Histogram (576 dimensions) and Bag-of-SURF (500 dimensions) are used to construct feature vectors from image. These feature vectors are then used to perform food classification using Linear SVM which is defined as inner product of two vectors. **Equation 2** and **Equation 3** are both equivalent Linear SVM formulas and the weighing factor *w* of input vector is calculated using **Equation 4**. All of the symbol descriptions are shown in **Table 2-2**.

An experiment performed on the proposed system achieved a satisfying result of 81.55% correct classification rate with top five candidates of detected food.

$$f(x) = \sum_{i=1}^{M} y_i a_i K(x, x_i) + b = \sum_{i=1}^{M} y_i a_i < x, x_i > + b \quad \textbf{(2)}$$

$$f(x) = < \sum_{i=1}^{M} y_i a_i x_i, x > + b = < w, x > + b \quad \textbf{(3)}$$

$$w = w^+ + w^- \quad \textbf{(4)}$$

Table 2-2 Symbols with descriptions for Linear SVM formula

Symbol	Description
x	Input vector
x_i	Support vector
$f(x)$	Output SVM score
$yi \in \{+1, -1\}$	Class label
a_i	Weight of support vector
b	Bias vector
M	Number of support vector

One of the strength of the proposed system is real time food classification, the system is capable of performing the action in real time, that is by simply point phone camera towards the food. Next, the system does not rely on any internet connection to function properly. This is because all image processing and recognition task are done in the device locally. As being said, users are able to acquire food calorie as quick as possible. To continue, the idea of using video instead of photograph allows user to locate correct food easily. Since the process of food recognition is repeated every second, user only need

to continuously search for a new position to point their camera until the correct food is identified.

On the other hand, the system does impose a few weaknesses. For an instance, performance could be a problem when there are too many dishes present in a single frame. This is because GrabCut (Rother, Kolmogorov and Blake, 2004) is resource intensive, drawing too much bounding boxes at once will burdens the CPU of the device. Secondly, the system is not very autonomous. It displays top five candidates for the food detected and users are required to select the right food manually. Furthermore, users are required to draw bounding boxes around the food portion before the calculation of calorie activates. Of course, this is due to the system is not focus on only a single dish but more than one, so it requires assistance from user to locate the food. Likewise, the usage of GrabCut (Rother, Kolmogorov and Blake, 2004) is also another reason for it as the algorithm requires a bounding box to be drawn to determine background (table) and foreground (food portion).

Overall, the proposed system is creative and innovative in tackling the problem of obesity and calorie intake management. Not only it is able to detect more than one food at a time but also do it in real-time at the cost of performance. However, certain useful techniques proposed in the system such as GrabCut (Rother, Kolmogorov and Blake, 2004) could be applied to the proposed project and the performance can be improved by restricting detection to single food at a time.

2.5 Fruits and Vegetables Calorie Counter Using Convolutional Neural Networks

Akbari Fard et al. (2016) proposed a fruit and vegetables calorie counter system as mobile application. Convolutional Neural Network (CNN) is used as food classifier. The goal of the system is to detect and classify type of food based on a photograph and retrieve its relevant calorie content. The framework of the proposed system is shown in **Figure 2-8**.

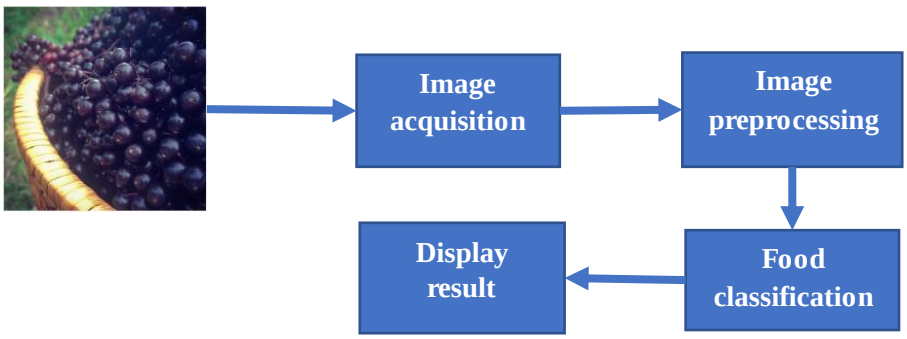


Figure 2-8 Framework for Fruits and Vegetables Calorie Counter (Akbari Fard et. al, 2016)

Firstly, test images were collected from ImageNet dataset where each of the images were pre-labeled. As a result, a dataset of 49,626 images spanning across 43 categories were handpicked. Among them, 80% of the images were used in training the CNN model while the rest were used for testing purposes. The features used for training are a combination of colour, texture, size and shape. The CNN consists of 3 convolution layers, 3 pooling layers and 2 fully connected layers. The distribution of filters can be seen in **Table 2-3**.

Table 2-3 Number of filters and Filter sizes for each convolution layer

Convolution Layer	Number of filter	Filter size
First	48	11 x 11
Second	128	5 x 5
Third	128	3 x 3

Nonetheless, several preprocessing steps were performed on every image before they were used for CNN model training. These includes resizing each of them to dimensions of 256 x 256. Then, the resized images were cropped to dimensions of 227 x 227 with random offsets. These steps were performed to increase the accuracy of the food classification.

An experiment is conducted to identify the accuracy of the CNN model on fruit and vegetable classification. Two tests were conducted to determine the percentage of correct food classification in first prediction and percentage of correct food classification among

top five predictions. The results are shown in **Table 2-4**. Overall, the experiment achieved an average accuracy of 75% for Top-5 test and 45% for Top-1 test.

Table 2-4 Results of percentage for Top-1 and Top-5 correct classification

Food name	Top-1 (%)	Top-5 (%)
Apple	13	53
Strawberry	61	83
Blueberry	32	69
Peach	40	74
Pear	20	58

The strength of the proposed system is the use of CNN as food classifier together with its huge image datasets which is 49,626 images used for training of the CNN model. This is because CNN requires more training images to improve its accuracy compared to Support Vector Machine (SVM). The second strength is its framework simplicity which benefits the environment of a mobile device. Moreover, the system does not perform any communications with server which implies that it will work regardless of internet connection. This is because all operations including image classification are done locally in the device itself.

However, the system achieved a poor accuracy with only 45% of correct classification in Top-1 prediction. This might be caused by lack of image segmentation as the background of food are not being removed may have an impact on the results of classification. Furthermore, cropping the image with random offset might also introduces inaccuracy during classification. It should be standardised on every image.

2.6 Image classification system based on Deep Learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes

Bruno and Osorio (2017) proposed an image classification system that is able to recognize traffic signals by applying deep learning. The main goal of the system is to increase road safety by usage of autonomous and semi-autonomous robotic vehicles. The

paper adopted TensorFlow (Tensorflow, 2018), an open source software library for machine intelligence to implement deep learning onto traffic signs classification.

Firstly, 21000 images which composed of 21 classes of German traffic sign was collected. Then, the image dataset was randomly split into 70% for as training set and remaining 30% for testing set for performance evaluation. The image dataset were converted from Portable Pixmap Format (PPM) to Joint Photographic Experts Group (JPEG). Image resolution were ranged between 15x15 and 256x256. Few examples of images are shown in figure below. The algorithm for detection is Slide Window where template are slide into input image to generate clippings. Then, the snippets are sent to Deep Convolutional Neural Network (DCCN) for classification. **Figure 2-9** showcase some examples extracted from the image dataset.

Figure 2-9 Examples of traffic signs from image dataset (Bruno and Osorio, 2017)

An experiment is conducted by shuffling the dataset before splitting into training and testing set for 10 iterations. All training are performed only with CPU. The experiment achieved a minimum accuracy of 94% and averaged at 97.24% of correct classification for segmented and tagged images. On the other hand, the system achieved an average of 75% accuracy for untagged images.

Nonetheless, one of the main benefits of deep learning its ability to accept raw input images and perform feature extraction on its own to learn from it. Compared to classic

machine learning, feature engineering needs to be done manually where learning is shallow. Besides, deep learning scales better with complex problems. As an example, a Support Vector Machine (SVM) usually require features in real-valued vectors while a Convolutional Neural Network (CNN) is end-to-end, which it has to ability to adapt to the problem it is solving. In general, CNN performed better with higher accuracy when compared to classic machine learning techniques.

On the other hand, deep learning techniques such as CNN do possess some disadvantages. Firstly, it require a lot of training data in order to perform well. Then, it is associated with high computational costs where time taken to train a model is significantly long. Nonetheless, its benefits outweighs the cons.

In a nutshell, the rising popularity of deep learning had raised interest of people into it and once again raised another challenges in the field of image classification for others. Its usefulness can certainly be applied in many other ways.

2.7 Programming Language

Python is one of the most flexible language that can be used for various purposes (All About Web, 2018). There are various useful libraries that are available for artificial intelligence and machine learning. Besides, it also serves as a great backend scientific computing (Codementor, 2018). For an instance, Scikit-learn is a popular library that is used for data mining and analysis and Numpy for scientific computing. These advantages had made Python to become the most popular language for machine learning and it will improve the efficiency of development and quality of the project.

2.8 Technology

TensorFlow, the open-source machine learning libraries is available for Python, Java and C, it is a tool designed to build deep neural network models (TensorFlow, 2018). Using TensorFlow, a built model can be deployed for usage on other platform including mobile platform, Android. This opened up wide opportunity for many new ideas to be implemented as it is impossible to perform machine learning in mobile platform before this without connecting to cloud (Greene, 2018). Besides, TensorFlow also supports

deployment of computation across multiple CPU or GPU easily. This helps to save a lot of training time. Moreover, as compared to other machine learning library such as Theano (Deeplearning.net, 2018), TensorFlow compile times is much better. Nonetheless, there is also a tool named TensorBoard that allow visualization of graph and performance easily.

One of the other popular deep learning libraries is Keras (Keras.io, 2018). It's much simpler and user friendlier compared to Tensorflow. The other benefits of Keras is its modularity. A complex neural network can be built in just a few lines of codes with Keras. However, TensorFlow provides much more flexibility than Keras as it offers more advanced operation. This means that TensorFlow provides more control over the network. Furthermore, Tensorflow stands out as it support for multi-platform provides more use case of neural network compared to others such as Theano and Keras.

2.9 Image classifying techniques

2.9.1 K-Nearest Neighbors (KNN)

KNN is a simple image classification technique that tries to find a predefined number of training samples, the k-nearest neighbours that have the closest distance to a new sample. (Scikit-learn.org, 2018). The first step of KNN is to memorize all the images data and their labels. Then, a distance function is used to measure distance between two images. The common distance function are L1 and L2 distance which are also known as Manhattan and Euclidean distance respectively as shown in **Equation 5 and 6** below (K Nearest Neighbors, 2018).

$$(l_1, l_2) = \sum_p \left| l_1^p - l_2^p \right| \qquad \textbf{(5)}$$

$$(l_1, l_2) = \sqrt{\sum_p (l_1^p - l_2^p)^2} \qquad \textbf{(6)}$$

The distance between each input samples are compared to every training samples and k number of training samples with closest distance to input samples is obtained and followed by their labels. A majority voting is performed on the predictions to get final predicted label for each input sample. In this case, the hyperparameter is value of k as it determines

how many predictions to be considered in the voting. An example of majority voting can be seen in **Table 2-5** below. In this case, donut will be the final predicted label.

Table 2-5 Sample data to showcase majority voting

Labels	No. of occurrence in predictions
Donut	5
Egg tart	2
Mooncake	3

The strength of KNN is its simplicity to implement as the algorithm is pretty straightforward while the weakness of KNN is its high computational cost as the size of the dataset grows larger. This will cause fast training as its just saving all data and slow prediction as it needs to compute distances between every training samples against every input samples.

2.9.2 Multiclass Support Vector Machine (SVM) Linear Classifier

To start with Multiclass SVM, it requires a loss function to tell how good the classifier is at predicting input dataset (Rosebrock, 2018). The score function and loss function are as shown in **Equation 7, 8 and 9** below. All the descriptions of symbols from the equations are in **Table 2-6** as shown below.

$$f(x, W) = Wx \quad \textbf{(7)}$$

$$L = \frac{1}{N} \sum_i L_i(f(x_i),\ y_i) \quad \textbf{(8)}$$

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1) \quad \textbf{(9)}$$

Table 2-6 Symbols with Description for Multiclass SVM loss formulas

Symbol	Description
W	Weight
x	Input samples
N	Number of samples

L_i	Loss for sample i
s_j, s_{yi}	Scores of true and predicted class computed from $f(x, W)$

In **Equation 7**, we compute the scores, s of each input samples. Then compare true class score against every predicted class score as seen in **Equation 9**. The true class score, s_y should be larger than predicted class score, s_j by margin of 1 to incure 0 loss, also known as hinge loss (Ipfs.io, 2018). Otherwise, set it to 0 as shown in Equation to prevent sum of negative loss. Lastly, simply divide by total number of samples to get overall loss as shown in **Equation 8**. To improve generalization and prevent overfitting, a regularization term is added to the computed loss usually (Chioka.in, 2018). The most common examples are L1 and L2 regularization, the formulas are shown in **Equation 10 and 11** below. L1 is just sum of weight while L2 is just sum of square of weight.

$$\mathrm{R}(W) = \lambda \sum_{i=1}^{k} |w_i| \qquad \textbf{(10)}$$

$$\mathrm{R}(W) = \lambda \sum_{i=1}^{k} w_i^2 \qquad \textbf{(11)}$$

Lastly, an optimization algorithm such as gradient descent, adam etc. is used to minimize the loss during the training session. The general gradient descent algorithm is shown in **Equation 12** below. Basically, it is just updating the weight by subtracting derivative term. A learning rate is used to control the speed of convergence towards local minima (Kdnuggets.com, 2018).

$$w^{(t+1)} = w^{(t)} - \alpha \frac{df(w^{(t)})}{dw} \qquad \textbf{(12)}$$

Table 2-7 Symbols with Description for Gradient Descent Algorithm

Symbol	Description
α	Learning rate
$\frac{df(w^{(t)})}{dw}$	Derivative term

One of the strength of multiclass SVM lies on its ability of generalization that creates simpler model. However, there are more hyperparameters to be considered and it is crucial to the performance of the classifier.

2.9.3 Deep Learning

According to Chen et al. (2018), Deep learning had gain huge success and applicable to wide area of applications such as voice recognition, image classification and games. It takes advantage of increasing computational power and availability of data nowadays. Deep learning actually inspired the way human brain works. Figure shows an artificial neuron and biological neuron which are very similar. Both of them work similarly

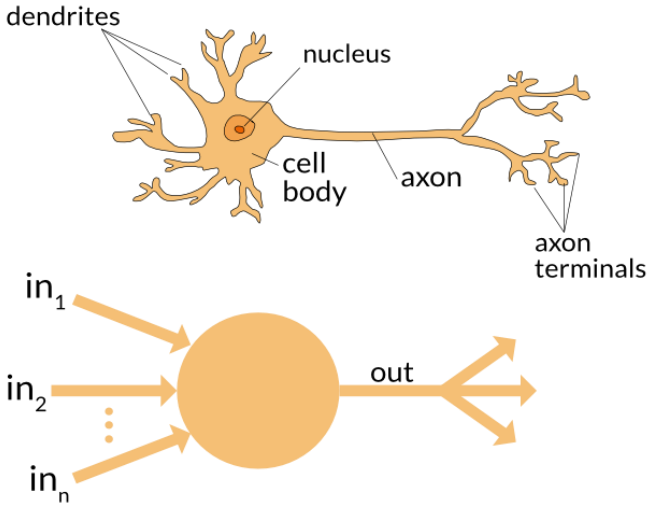


Figure 2-10 Biological and artificial neuron, also known as perceptron (Perceptron, 2016)

by taking in inputs, process them and send out the output. According to Applied Go (2016), neuron receives multiple inputs and process it into one output. Each input is multiplied by a weight and add to a bias. During the training phase, the neural network adjust the weights and biases accordingly through a process called backward propagation using gradient descent (Brilliant.org, 2018) where calculations of gradient proceeds backwards from last layer to the first. This indicates that the neural network is learning from its data through minimizing the loss function.

There are many types of deep learning, Convolutional Neural Network is one of the most popular one. CNN is built up by a sequence of layers that transform an input to output

with some differentiable function (Cs231n, 2018). The common layers in CNN are convolution layer, pooling layer and fully connected layer. First of all, Convolution layer is essentially a set of filters where each filter detects a feature. Its theory are similar to pooling where a sliding window scan across the inputs while each capture different features such as colour, contours, edges etc. **Figure 2-11** shows an example of the convolution layer. Then, another common layer is Fully Connected layer. In this layer, every neurons from one layer connects to the other layer. An illustration of Fully Connected layer is shown in **Figure 2-12** below.

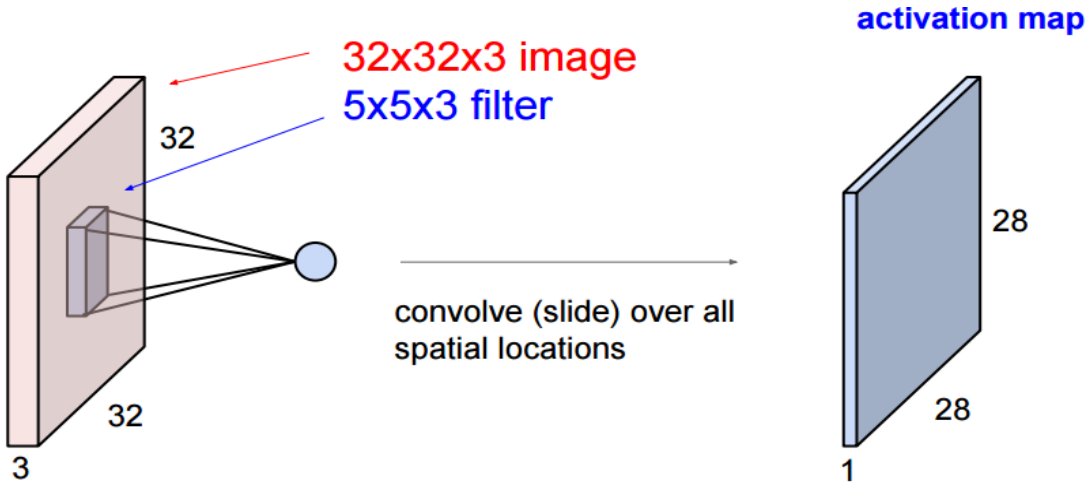


Figure 2-11 An example of convolution layer (Main actor the convolution layer, 2018)

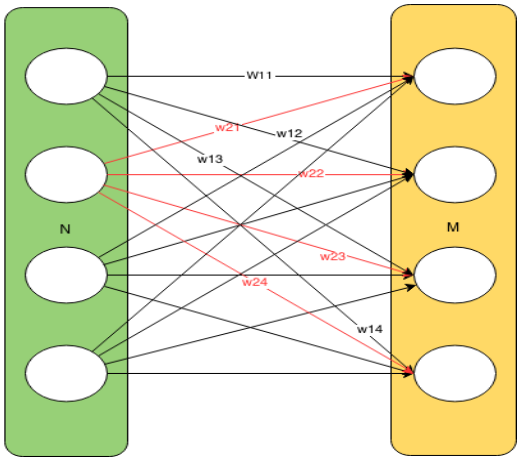


Figure 2-12 An example of fully connected layer (Pennlio, 2014)

Besides that, sometimes overfitting can happen, this is the case where the network is too closely fit to the training set. It needs to be simpler and general in order to solve

problems outside the training data. Therefore, dropout layer can help in this situation. This improves generalization as the model drops some of the neurons causing them to be deactivated (Santos, 2018). An example of illustrating the neural network with dropout layer is shown in Figure 2-13 below.

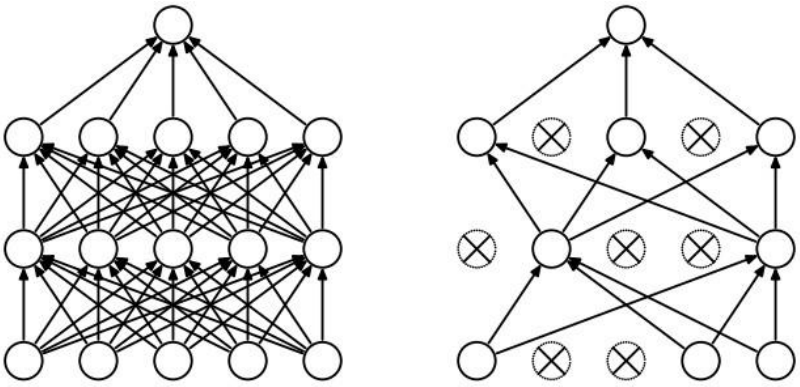


Figure 2-13 Neural network before and after dropout layer (Santos, 2018)

Aside from that, there is also another popular deep learning techniques known as Recurrent Neural Network (RNN). The idea behind RNN is the use sequential information while the term 'recurrent' means that it perform the same task for every element in the sequence (Britz, 2015). The formula for Vanilla RNN can be seen in **Equation 13** and **14** below (Karpathy, 2015). The symbols and descriptions for it are shown in **Table 2-8**.

$$h_t = f_w(h_{t-1}, x_t) \qquad \textbf{(13)}$$

$$y_t = W_{hy} h_t \qquad \textbf{(14)}$$

Table 2-8 Symbols and descriptions for Vanilla RNN

Symbol	Description
h_t, h_{t-1}	New and old hidden state
f_w	Some function with parameters W
x_t	Input vector at some time step
y_t	Output vector

Since that RNN does not take in fixed size of input and produce fixed size of output like CNN did, it is ideal for sequential data such as text generation, speech recognition and machine translation. On the other hand, CNN is more ideal for classification task such as image classification.

2.10 Transfer learning

Transfer learning is a machine learning method where knowledge learnt for a task is being reused on another task (Brownlee, 2017). In fact, very few people actually train an entire convolutional neural network, CNN from scratch (Cs231n, 2018). One of the reason is that training a good network requires a huge amount of datasets, time and resources. Instead, transfer learning had been a popular approach in deep learning where it could speed up training and improves performance.

According to Torrey and Shavlik (2009), humans also applies transfer learning between different tasks. For instance, whenever we faces a new task, we will recognize and apply relevant knowledge where we acquired from previous experience. To better understand the idea of transfer learning, Urban (2017) explained it in terms of transfer of knowledge through inventions of language during tribal times. The impact of transfer learning is significant throughout the entire human history as shown in **Figure 2-14** and **Figure 2-15** below.

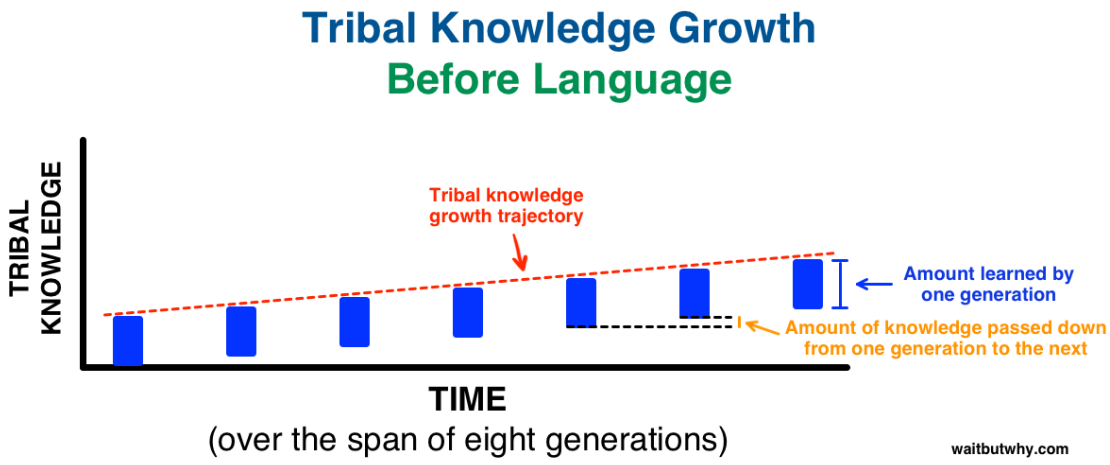


Figure 2-14 Tribal knowledge growth before language (Urban, 2017)

As seen in **Figure 2-14** where it depicts times before language was being invented, knowledge growth among generations are slow and insignificant. This is because each generations must learn and acquire knowledge by themselves from scratch. Followed by **Figure 2-15** where it depicts times after language was being invented as shown below.

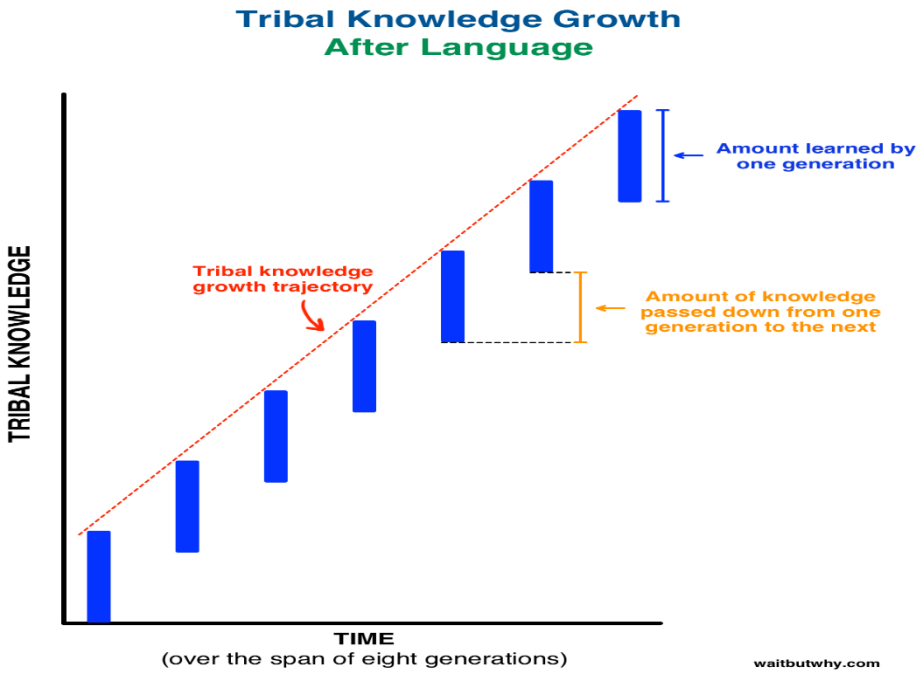


Figure 1-15 Tribal knowledge growth after language (Urban, 2017)

As seen in **Figure 2-15**, the knowledge growth became significant among generations. This can be explained where knowledge was being passed down from the previous generation to the next. Tribes did not need to learn knowledge from scratch anymore and they could apply knowledge learnt by previous generations as a starting point and further increase more knowledge acquired into them instead.

In other words, transfer learning is an optimization, it saves time and increase performance at the same time (Brownlee, 2017).

2.10.1 Pre-trained model approach

In real-world practices, there are many popular models published by others that were trained to solve certain problems. Instead of defining a brand new architecture and train it from scratch, these pre-trained models could be used as a starting point (Urban, 2017). According to Torrey and Shavlik (2009), the benefits of transfer learning can be summarized into the **Figure 2-16** as shown below. The figure explains the difference between training a network with and without transfer learning.

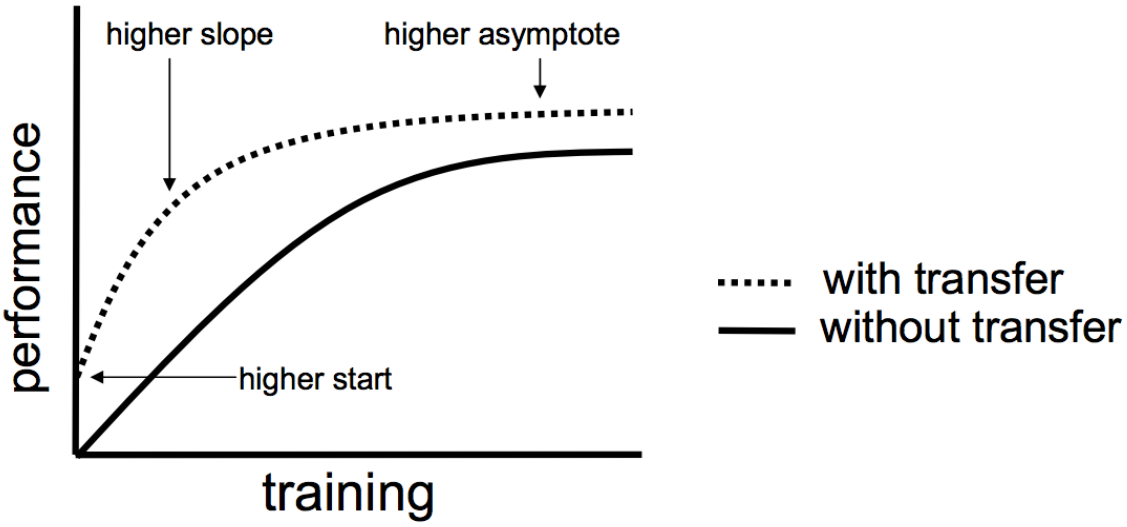


Figure 2-16 Benefits of transfer learning during training phase (Torrey & Shavlik, 2009)

While there are not a specific way to perform a transfer learning, there are a few standard approaches around. According to Urban (2017), the approaches could be using only the model architecture and train from scratch on our own dataset, using the model as a feature extractor where only output layer is being trained and training only a few layers while the rest are being froze.

Since the earlier layers in a CNN learns about general features such as edges, textures etc. These layers are being frozen while the last few layers where classifying happens are being trained. By doing this way, features learnt from previous task are actually being transferred and applied to a different task. The total number of trainable parameters also decreases significantly. Transfer learning not only saves time but also improves performance significantly.

However, transfer learning using pre-trained models does not necessarily always produce a desired results. This could happen due to the gap difference between the tasks where the model was being trained on. An experiment was performed to evaluate the benefits of transfer learning as written in **Chapter 2.12.2.** Overall, transfer learning provides an alternative way to improve performance and the project would definitely benefits from this technique.

2.11 Review of several pre-trained models

This chapter reviews those pre-trained model architecture that were evaluated in the experiment as written in **Chapter 2.12.2**. As all of these models were pre-trained on ImageNet dataset which consist of 1000 classes. The diversity of the classes increase the reusability of the features learnt to be applied on other task.

2.11.1 ResNet (Residual network)

According to He et al. (2016), deep convolutional neural network (CNN) has led to several breakthroughs in image classification recently. Evidence shown that deep networks is crucial to these amazing results. Hence, there were a lot of new models that were built on the concept of deep networks. However, there exist problems of degradation as deeper network starts converging as seen in **Figure 2-17**. Based on **Figure 2-17** as shown, training

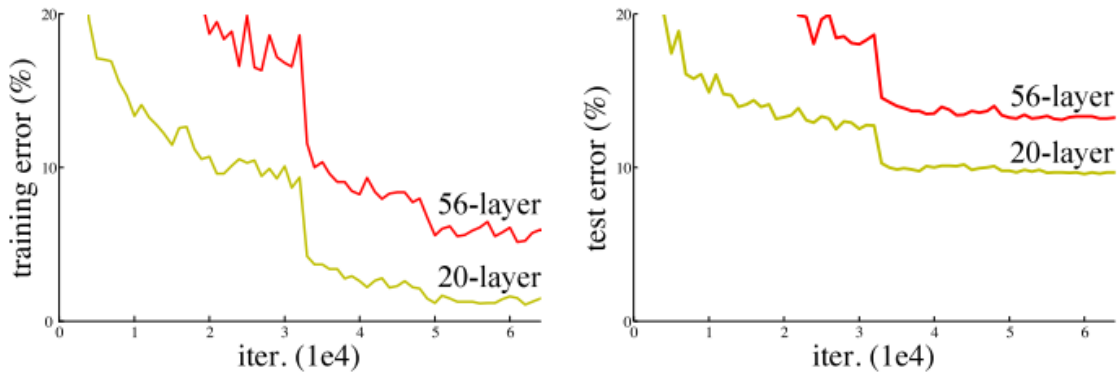


Figure 2-17 Graph of training and testing error from 20-layer and 56-layer plain networks on CIFAR-10 dataset. Higher test error in deeper networks was resulted from higher training error as seen (He et al., 2016)

error is higher for deeper networks. This was not caused by overfitting issues as testing error is also higher for deeper networks. However, it was also not caused by vanishing gradient as batch normalization was used in plain networks.

As the depth of network increases, the network performance somehow got saturated and even degraded. He et al. (2016) argued that increasing network depth does not work by simply stacking layers together. Training error of a deeper network should be lower than training error from its shallower counterpart. This idea served as a motivation for residual networks.

The core idea of residual networks or ResNet is to address the degradation problem by introducing shortcut connections that allows the network to pass earlier information to deeper layers. This is named as a residual block as shown in **Figure 2-18**. Given that *H(x)*

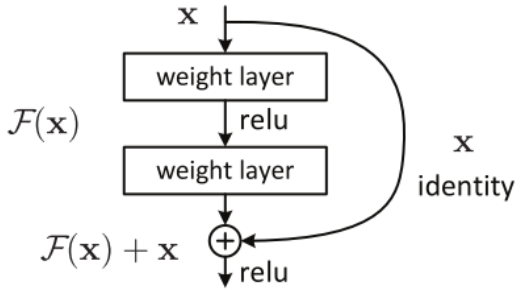


Figure 2-18 Residual block (He et al., 2016)

= F(x) + x, He et al. (2016) argued that a residual mapping *F(x)* is easier to be optimized compared to the original, unreferenced mapping *H(x)*. For instance, by comparing a plain block where y = F(x) to a residual block where y = F(x) + x, it is hard to learn F(x) = x and make y = x an identity mapping compared to residual block where we can just make F(x) to be 0 and make y = x an identity mapping. To sum that up, this reformulation solves the issue of performance degradation problem by allowing added layers in deep networks to be constructed as identity mappings whenever it is optimal so that also fulfils the statement where a deeper model should have training error no greater than its shallower counterpart.

He et al. (2016) strengthen their arguments by comparing the performance of residual network, ResNet to plain networks as seen in **Figure 2-19**. The problems of degradation was well addressed.

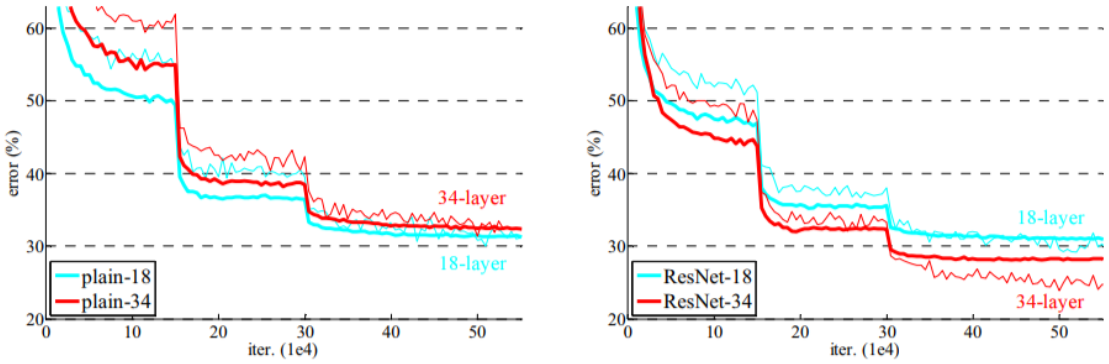


Figure 2-19 Training error of plain vs residual networks on ImageNet datasets (He et al., 2016)

There are several variations of ResNet with different number of layers such as ResNet-18, 34, 50, 101 and 152. To improve training efficiency for deeper version of ResNet such as ResNet-50, bottleneck design was introduced (He et al., 2016). The difference between a regular residual block and a bottleneck block is shown in **Figure 2-20**. Basically, the purpose of 1x1 layers are to reduce and increase dimensions thus leaving 3x3 layer a bottleneck with smaller input and output dimensions. As shortcut was connected to the two high-dimensional ends, time complexity was prevented to be doubled.

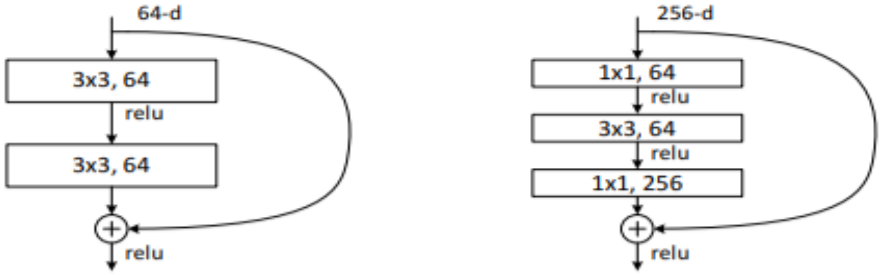


Figure 2-20 A building block (left) and a bottleneck building block (right) (He et al., 2016)

2.11.2 GoogleNet (Inception)

In the year 2014, Szegedy et al. proposed a new efficient deep neural network architecture codenamed Inception. The term 'deep' was used in two different meanings. The first one being introduction of new level of organization with Inception module. The later simply means increased depth of network.

According to Szegedy et al. (2014), increasing network depth and width are the most straightforward ways to improve a deep neural network performance. However, Szegedy et al. (2014) argued that there are drawbacks by doing so. Firstly, increased network size equals to increased number of parameters which may cause the network more prone to overfitting. Secondly, it also dramatically increases the use of computational resources. This is the result of uniform increase in number of filters in all layers as the number of parameters will be increased quadratically. In order to solve these issues, Szegedy et al. (2014) proposed removal of fully connected layers where it also resonates with Hebbian pricinple which states that 'neurons that fire together, wire together'.

The core idea of Inception architecture is to find optimal local construction and repeat it spatially. According to Szegedy et al. (2014), the goal of an Inception module is to design a good local network topology and then stack these modules on top of each other. **Figure 2-21** shows an example of Inception module. As mentioned by Szegedy et al.

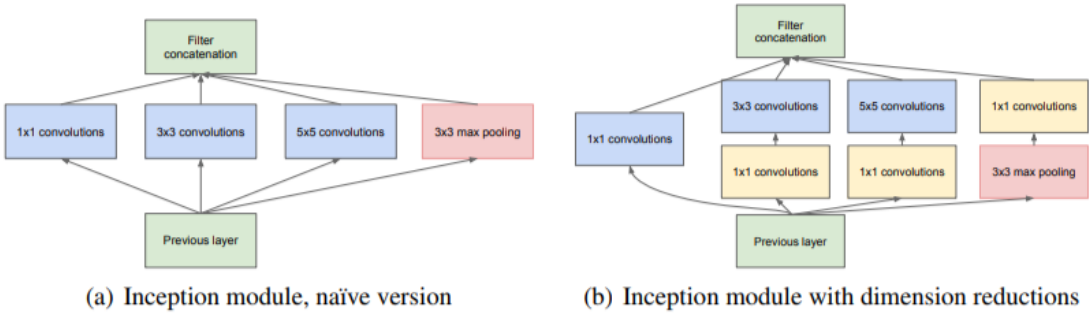


Figure 2-21 Inception module (Szegedy et al., 2014)

(2014), the problem lies within Inception module, naïve version is that 5x5 convolutions can be prohibitively expensive on top of a convolutional layer with large number of filters. This leads to generation of second idea which can be seen in **Figure 2-21 (b)**, the problems can be solved by applying dimensionality reduction and projections whenever computational requirements will be intensive. Basically, the 1x1 convolutions are introduced for dimensionality reductions before other expensive 3x3 or 5x5 convolutions and also allows use of activation functions. Furthermore, the number of filters of these 1x1 convolutions can be used to control the number of output channels.

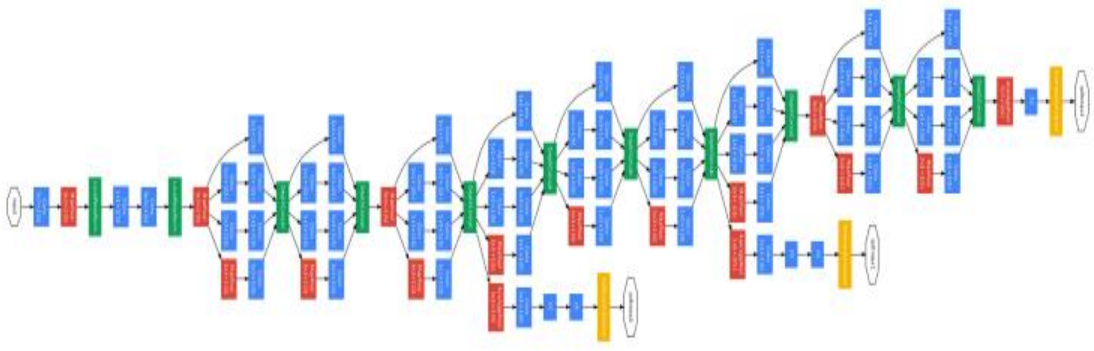


Figure 2-22 GoogleNet (Szegedy et al., 2014)

Moreover, the GoogleNet or Inception as shown in **Figure 2-22** consist of 22 layers with stacking of nine Inception modules. There is only a final fully connected layer and the top fully connected layer is being replaced with a global average pooling layer. As argued by Szegedy et al. (2014), fully connected layers caused deep neural network prone to overfitting issues as it increases the number of parameters of the network. As a result, Inception has 12 times fewer parameters than AlexNet and lower computation resources.

To sum that up, GoogleNet focus on the idea of dimensionality reduction and removal of fully connected layers to reduce number of parameters. In the year of 2015, Szegedy et al. proposed to further improve the network training time by factorizing convolutions with large filter size in later version of Inception networks. The idea is to replace larger convolution with multiple smaller convolutions which will produce lesser number of parameters but at the same time still maintains the same input size and output depth. For instance, Szegedy et al. (2015) proposed replacing 5x5 convolution with two 3x3 convolutions. A comparison can be seen in **Figure 2-23**. The efficient use of factorizing convolutions reduced a lot of parameters to be optimized from the networks

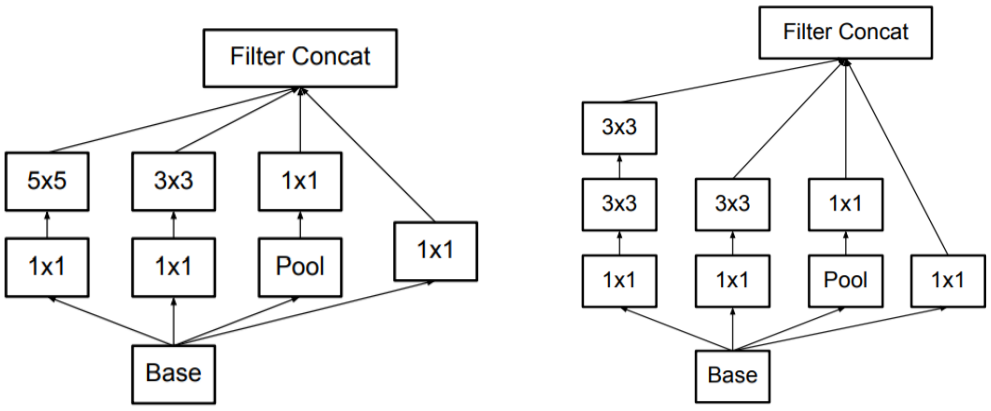


Figure 2-23 Original Inception module (left) and new Inception module (right)

and cut down the cost of computation and yet still achieve the state-of-the-art results where InceptionV3 achieved 21.2% for top-1 and 5.6% for top-5 error during single crop evaluation on the ILSVR 2012 classification.

2.11.3 Inception-ResNet

According to Szegedy et al. (2016), Inception architecture had shown to achieve great performance at low computational cost while residual networks with a more traditional architecture had achieved state-of-the-art performance in 2015 ILSVRC challenge. Hence, this serves as a motivation for the proposal for combination of both networks.

The core idea of Inception-ResNet was exactly same as its name, to combine both Inception and ResNet architecture as discussed previously. This allows Inception to exploit the benefits of residual networks while still able to retain its computational efficiency. **Figure 2-24** showcase the comparison between a regular residual connection and the

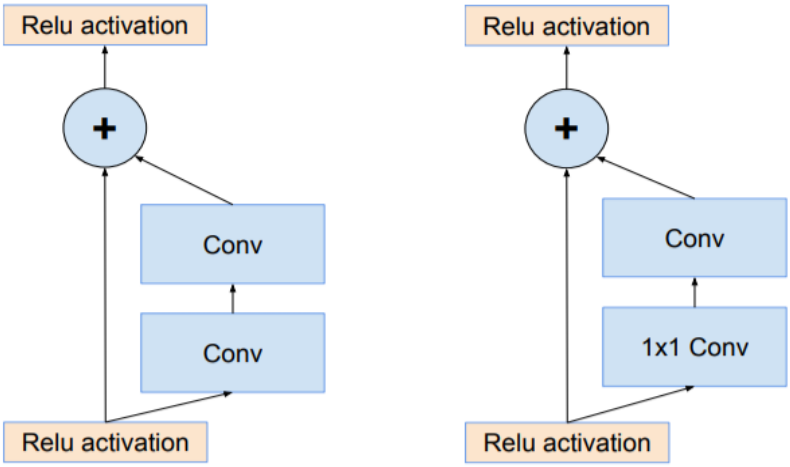


Figure 2-24 Original residual connections (left) and Optimized residual connections (right) (Szegedy et al., 2016)

optimized version which was presented in Inception-ResNet. The optimized version was named Residual Inception block instead. In addition, Inception-ResNet used batch normalization only on top of the traditional layers but not summations. Aside of that, an experiment was done to compare between the performance of top-1 error between Inception-v3 and Inception-ResNet-v1. The results were shown in **Figure 2-25** where Inception-ResNet clearly outperforms the traditional Inception network. Despite faster training of Inception-ResNet, it achieved a slightly poorer final accuracy compared to its traditional counterpart.

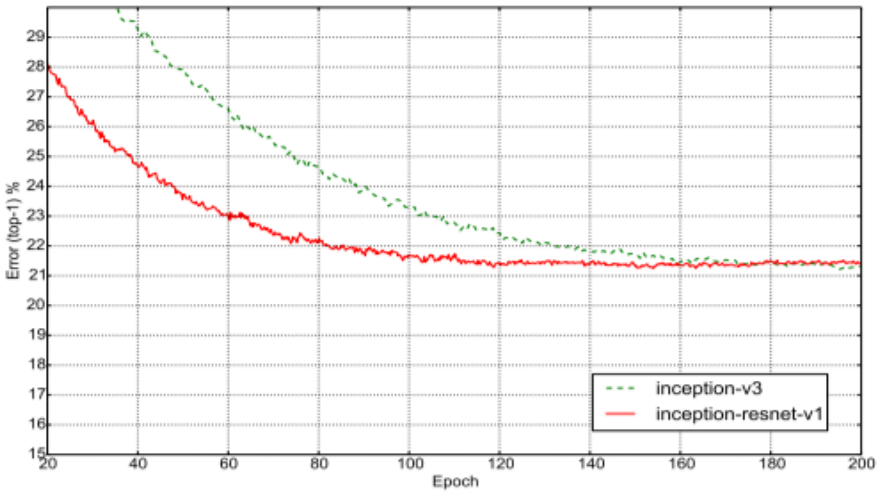


Figure 2-25 Top-1 error of pure Inception-v3 vs Inception-Resnet-v1 of similar computational cost (Szegedy et al., 2016)

2.11.4 MobileNet

According to Howard et al. (2017), the current trend in deep neural networks is to go deeper and become more complicated. However, the improvements in accuracy does not make the network more efficient in terms of size and speed. In fact, real world applications requires the recognition task to be performed in timely fashion on a computationally limited platform. For instance of recent popular real world applications are augmented reality, self-driving vehicles, robotics etc. Hence, this serves as a motivation for proposal of MobileNets.

The core idea of MobileNets is to build networks an efficient network with low latency and size which can be easily embedded into mobile or other embedding devices. In other words, model speed size are being heavily focused. First of all, MobileNets are based on a form of factorized convolution named depthwise separable convolution. Basically, a depthwise separable convolution factorizes a standard convolution into a depthwise convolution and a 1x1 convolution, also named pointwise convolution. The first layer is for filtering while the latter is for combining. **Figure 2-26** illustrate the process of factorizing a standard convolution into a depthwise convolution and a pointwise convolution. As a result of this factorization, computation and model size had been drastically reduced.

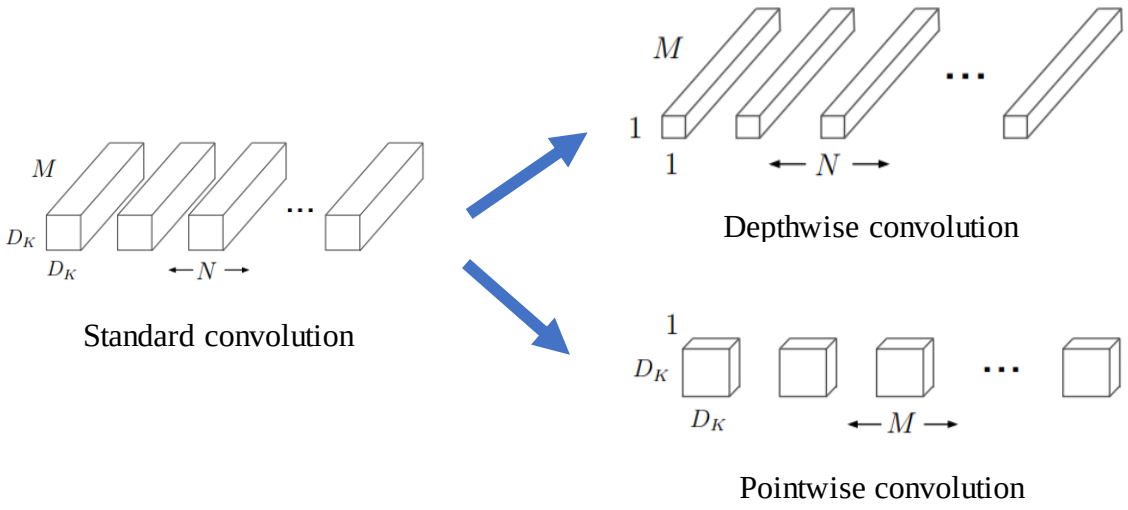


Figure 2-26 Standard convolution factorized into two separate layers named depthwise convolution and pointwise convolution (Howard et al., 2017)

The idea of depthwise separable convolution is to reduce the computational cost comparing to a standard convolution. A comparison of computation cost for a standard convolution and a depthwise separable convolution is shown in **Equation 15** and **16**.

$$D_K * D_K * M * N * D_F * D_F \quad \textbf{(15)}$$

$$D_K * D_K * M * D_F * D_F + M * N * D_F * D_F \quad \textbf{(16)}$$

Table 2-9 Description of symbols for computation cost (Howard et al., 2017)

Symbols	Description
D_K	Convolution kernel size
D_F	Spatial width & height of input
M	No. of channels of input
N	No. of channels of output

Howard et al. (2017) used depthwise separable convolution to break the interaction between size of kernel and amount of output channels. Depthwise convolutions are very computationally efficient compared to a standard convolution. However, a depthwise convolution only filter input channels and does not combine them to create new features. This is where pointwise convolution comes into play in order to generate these new

features. As a result, MobileNet uses 3x3 depthwise which reduced computational cost by 8 to 9 times comparing to a standard convolution. With that being said, the drawback is that the accuracy is slightly worse than standard convolution. Beside the last fully connected layer, all layers in MobileNet are accompanied with a batchnorm and ReLU nonlinearity. A comparison of standard convolution layer with a depthwise separable convolution is shown in **Figure 2-27**.

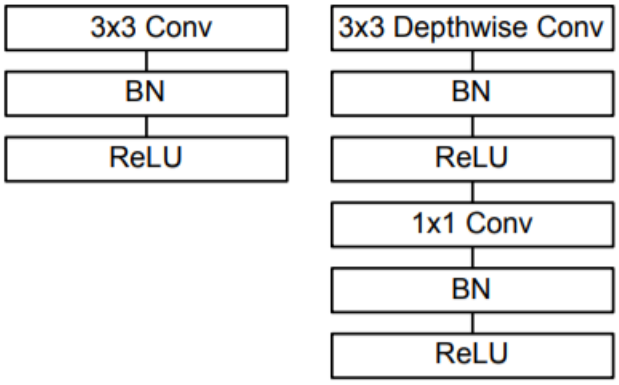


Figure 2-27 Standard convolutional layer followed by batchnorm and ReLU (left) vs depthwise separable convolution followed by batchnorm and ReLU (right) (Howard et al., 2017)

While the MobileNet architecture consisting of 28 layers by counting depthwise and pointwise convolutional layers as separated layers is already small and fast enough, it could still be made even smaller and faster. To achieve that, Howard et al. (2017) introduced a parameter known as width multiplier, *a*. The sole purpose of this parameter is thinning a network at each layer uniformly. With that being said, the new computational cost of a depthwise separable convolution with width multiplier is as **Equation 17**.

$$D_K * D_K * aM * D_F * D_F + aM * aN * D_F * D_F \qquad \textbf{(17)}$$

The baseline MobileNet has $a = 1$ while $a < 1$ are reduced MobileNets. According to Howard et al. (2017), the width multiplier can reduce the number of parameters quadratically and computational cost by roughly a^2. Following width multiplier is another hyperparameter known as resolution multiplier, *p*. This multiplier is applied to input image thus change the computational cost again as shown in **Equation 18**.

$$D_K * D_K * aM * pD_F * pD_F + aM * aN * pD_F * pD_F \qquad \textbf{(18)}$$

The same rules applies for resolution multiplier as $p = 1$ for baseline MobileNets and $p < 1$ for reduced MobileNets. As a result, resolution multiplier can reduce computational cost by p^2. Because of the resolution multiplier is set implicitly, the input resolution of network is 128, 160, 192 and 224.

In the year of 2018, Sandler et al. (2018) proposed MobileNetV2 as the successor to the original MobileNet as proposed in previous year. Same as the goal of its predecessor which is to introduce a new architecture that is specially designed for mobile and resource constrained environments. Sandler et al. (2018) tries to decrease the number of operations and memory required while retaining the same accuracy by introducing novel layer module, the inverted residual with linear bottleneck. This new module reduces the memory footprint during inference significantly in mobile environment.

First of all, MobileNetV2 retains the use of 3x3 depthwise separable convolution as in the first MobileNet. However, there are some changes to it. An addition 1x1 layer before the depthwise convolution while the pointwise layer now act as projection or bottleneck layer. The newly added 1x1 layer will act the exactly opposite to the pointwise convolution which is to expand dimension before depthwise convolution. Secondly, residual connection was introduced to the same block which work just like those in ResNet. The existence of residual connections aids the flow of gradient during backward propagation. The modified version of the original depthwise separable convolution block is now named as residual bottleneck block. An example of a residual bottleneck block is shown in **Figure 2-28** as compared to the original block.

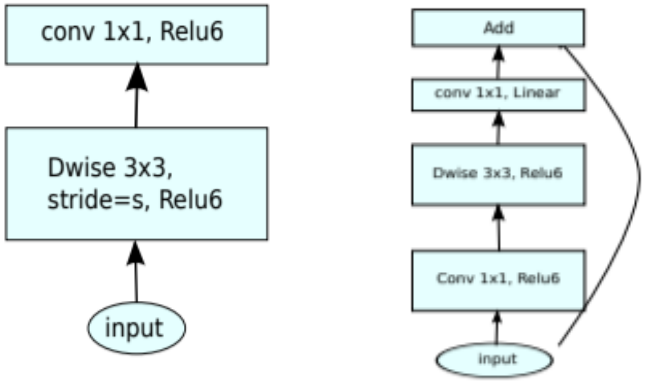


Figure 2-28 Comparison between original depthwise separable convolution block vs residual bottleneck block (Sandler et al., 2018)

As mentioned by Sandler et al. (2018), the motivation behind these changes were due to the smaller the tensor, the fewer the multiplications the convolutional layers have to do. Thus, the lower the computational cost. Sandler et al. (2018) concluded with stating MobileNetV2 achieving better results than its predecessor in terms of accuracy and computation time and memory efficiency as shown in **Table 2-10** and **Table 2-11**.

Table 2-10 Comparison of top-1 accuracy and computation time between MobileNet and MobileNetV2 (Sandler et al., 2018)

Network	Top-1 accuracy	CPU
MobileNet	70.6%	113ms
MobileNetV2	70.2%	75ms

Table 2-11 Comparison of memory efficiency (max number of channels/memory in Kb that needs to be materialized at each spatial resolutions) between MobileNet and MobileNetV2 (Sandler et al., 2018)

Size	**MobileNet**	**MobileNetV2**
112x112	1/O(1)	1/O(1)
56x56	128/800	32/200
28x28	256/400	64/100
14x14	512/200	160/62
7x7	1024/199	320/32
1x1	1024/2	1280/2
max	800K	200K

2.11.5 VGG

In the year of 2015, Simonyan and Zisserman proposed a deep convolutional neural network named VGG. The architecture of VGG nets are simple as its just stacking of convolutional layers followed by three fully connected layers.

The idea of VGG networks was to use small filters in convolution to allow growth in depth. VGG have 2 variants which are VGG-16 and VGG-19 where the integer represents the number of layers. Moreover, VGG uses small 3x3 filters in convolution while the number of filters gradually increases down the network. At the top of the network was three fully connected layers where the first two have 4096 channels each while the latter has 1000 channels for classification. In order to preserve spatial resolution, stride and

paddings of 1 are used in every convolutional layers. Furthermore, a total of 5 maxpooling layers were used with 2x2 filters and stride of 2.

Comparing to AlexNet which only have 8 layers, VGG nets have more than double of layers and also number of parameters. For instance, AlexNet consist of 60M parameters while a VGG-16 consist of up to 138M parameters. The overly huge number of parameters may cause the network prone to overfitting issues during training.

2.11.6 Xception

In the year of 2017, Chollet (2017) proposed a network named Xception which was named after 'Extreme Inception'. The network was inspired by Inception-v3 and attempted to improve on it. In order to achieve that, Chollet (2017) proposed to map cross-channel correlations by using a 1x1 convolution followed by mapping the spatial correlations of each output channels separately. This version of Inception module is very similar to a depthwise separable convolution as discussed in MobileNets.

Basically, the architecture of Xception was just replacement of Inception module with depthwise separable convolution. An example comparison of a simplified Inception module with an 'extreme' version of Inception module as proposed by Chollet (2017) is shown in **Figure 2-29**.

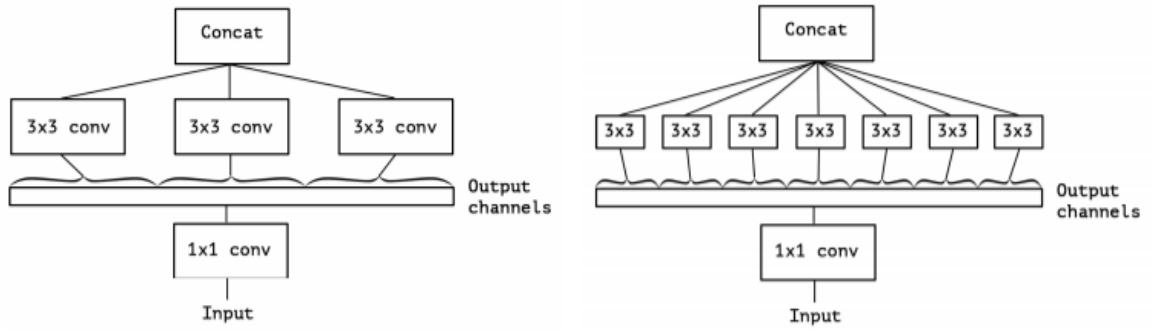


Figure 2-29 Simplified Inception module (left) vs 'extreme' version of Inception module (right) (Chollet, 2017)

As mentioned by Chollet (2017), the first difference between 'extreme' version of Inception module and an original Inception module is the order of operation. In depthwise separable convolution, channel-wise spatial convolution was performed first followed by 1x1 convolution while Inception module performs 1x1 convolution first. The second

difference is the presence of non-linearity functions. In depthwise-separable convolution, there are usually no non-linearity implemented while ReLU non-linearity was implemented after the operations in Inception module.

As mentioned earlier, Xception architecture was based entirely on depthwise separable convolution layers or 'extreme' version of Inception module with residual connections. There were a total of 36 convolutional layers structured into 14 modules where all consist of residual connections.

According to experiment performed by Chollet (2017), Xception managed to slightly outperforms Inception-v3 in both top-1 and top-5 accuracies on ImageNet dataset as shown in **Table 2-12**. Besides, the comparison of size and speed between both architectures were also shown in **Table 2-13** also shows that Xception slightly outperformed Inception-v3.

Table 2-12 Performance comparison between Inception-v3 and Xception on ImageNet dataset (Chollet, 2017)

	Top-1 accuracy	**Top-5 accuracy**
Inception-v3	0.782	0.941
Xception	0.790	0.945

Table 2-13 Size and speed comparison between Inception-v3 and Xception on ImageNet dataset (Chollet, 2017)

	Parameter count	**Steps / seconds**
Inception-v3	23,626,728	31
Xception	22,855,952	28

2.11.7 NASNet

According to Le and Zoph (2017), the Google team had successfully built several models and applied it in many different ways from machine translation to speech recognition and image recognition. However, behind all of these amazing work were a huge team of engineers and scientists. The process of designing and experimenting machine learning models is a painfully time-consuming task. Instead, Le and Zoph (2017)

proposed to automate the process of designing machine learning models which motivates the creation of AutoML. Basically, AutoML is a controller neural net which can propose a brand new model architecture. Then, feedbacks or the resulting child model accuracy are used to inform AutoML how to improve the proposals of new model architecture. The process can be summarized as **Figure 2-30**.

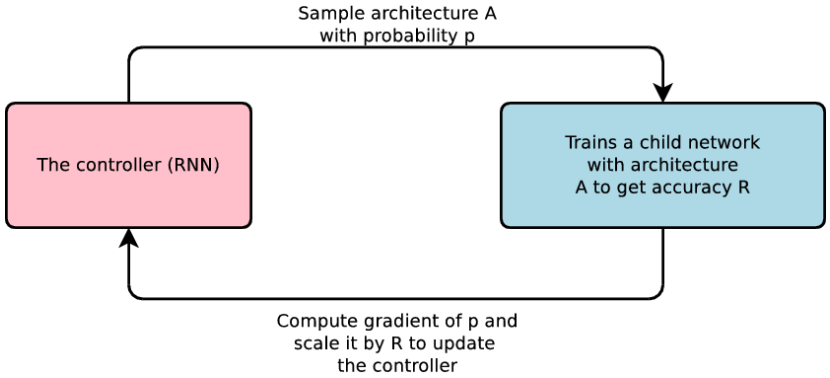


Figure 2-30 Process of automating creation of machine learning models using AutoML (Le & Zoph, 2017)

However, Le and Zoph (2017) found that AutoML was able to design model that were on par with human experts designed model on small dataset like CIFAR-10. Therefore, this serves as motivation for the proposal to modify AutoML approach to perform better on a large dataset like ImageNet (Zoph et al., 2018).

In fact, the term NAS stands for Neural Architecture Search and it was used in AutoML approach. In simpler words, NAS framework uses reinforcement learning to optimize architecture configurations. As mentioned by Zoph et al. (2018), directly applying NAS onto a large dataset would be computationally expensive. Instead, they proposed to search for a best layers in small dataset with AutoML then transfer the learned architecture to a large dataset. This transferability was achieved through designing a search space named NASNet search space. The purpose of this search space is to allow complexity of architecture to be independent of input images size and network depth.

As a result of the proposed approach (Zoph et al., 2018), among the created models, NASNet was created and determined to be the best. In order to build scalable architecture, AutoML tries to predict a generic convolution cell which can be stacked. There are two types of cell named normal cell and reduction cell. Both cells are shown in **Figure 2-31**.

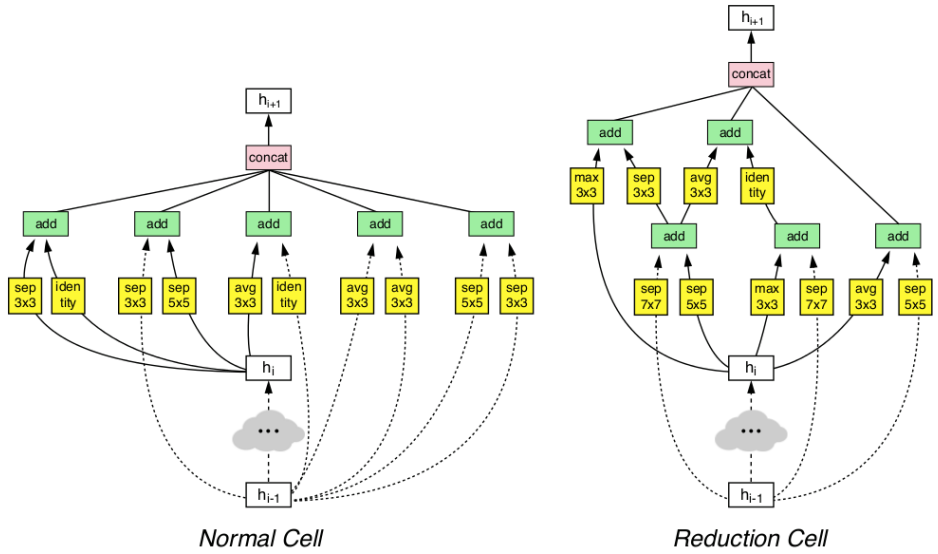


Figure 2-31 Normal cell and Reduction cell in NASNet (Zoph et al., 2018)

As the names goes, the first convolution cell returns feature map of same dimension while the latter returns a feature map where sizes are reduced by factor of two. An example architecture for the created architectures for CIFAR-10 and ImageNet is illustrated in **Figure 2-32**. As seen, ImageNet architecture has more reduction cells than the others.

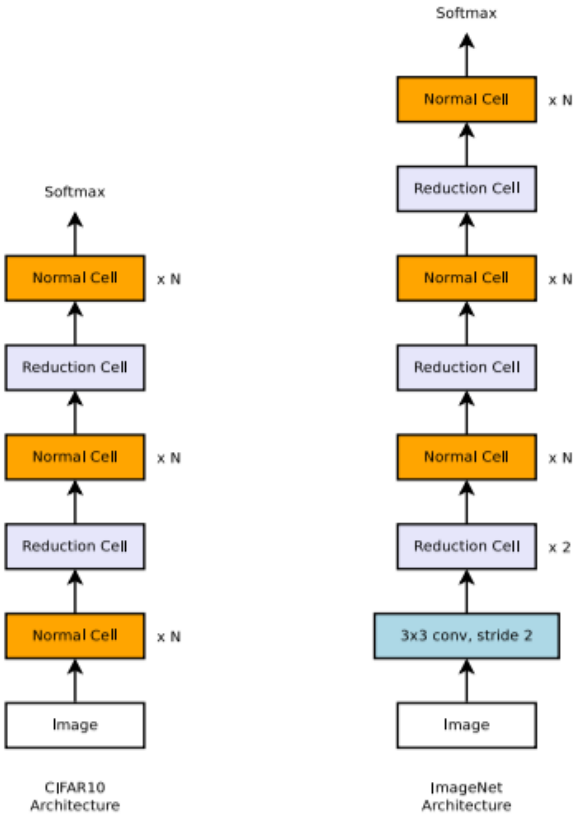


Figure 2-32 Scalable architecture with Normal and Reduction cells (Zoph et al., 2018)

As a result, NASNet surpassed the state-of-the-art results and achieved Top-1 and Top-5 errors of 82.7% and 96.2% on ImageNet dataset in May of 2018 (Hulstaert, 2018).

2.11.8 DenseNet

According to Huang et al. (2016), DenseNet stands for Dense Convolutional Network which connect each layer to every other layer in a feed-forward fashion. The main difference between a traditional convolutional network and DenseNet is the number of connections. For instance, there are ***L*** connections for ***L*** layers in a traditional convolutional network while there are ***L(L+1) / 2*** connections in DenseNet. In other words, each layer receives feature maps of all preceding layers as input in a DenseNet. As mentioned by Huang et al. (2016), the motivation behind this was to solve vanishing gradient problem, reduce number of parameters, promotes feature reuse and strengthen feature propagation.

Figure 2-33 showcase a DenseNet with three dense block. The introduction of

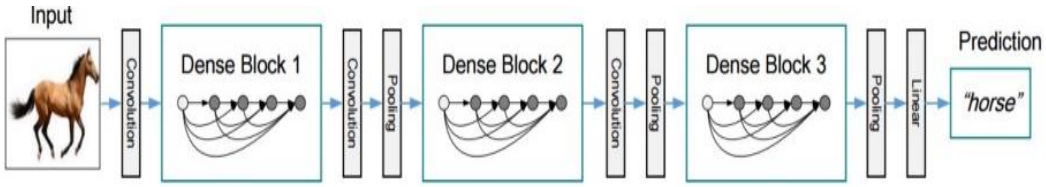


Figure 2-33 Example of a DenseNet with three dense block (Huang et al., 2016)

dense block promotes feature reuse from previous layers, thus reducing number of parameters compared to a traditional convolutional network. This gives better flow of information between layers as illustrated in **Equation 19**.

$$\boldsymbol{x_l = H_l([x_o, x_1, \dots, x_{l-1}])} \qquad \textbf{(19)}$$

Table 2-14 Description of symbols for Equation 19 (Huang et al., 2016)

Symbols	Description
x_l	Feature maps of current layer
H_l	Function
$[x_0, x_1, \dots x_{l-1}]$	Concatenation of feature maps of previous layers

To sum that all up, DenseNet basically took inspiration of traditional convolutional network and improve on it by introducing reuse of feature maps from previous layers. Not only this promotes information flow but also reduce the number of parameters significantly.

2.12 Experiments

2.12.1 Comparison of performance for several image classification techniques

An experiment is conducted for 4 times with different image classification techniques Multiclass SVM, FCNN, CNN and KNN. An image dataset containing 5 classes of dessert with 200 images each. **Figure 2-34** shows the sample image for each classes from the dataset.

Apam Balik

Cookie

Donut

Egg Tart

Mooncake

Figure 2-34 Examples of image from each classes of dessert in the experiment dataset

The performance of each classification technique is determined by their accuracy of prediction on unseen data, the testing set. All of the experiment were performed on the same dataset for consistency. The dataset were first split into training and test set at ratio of 8:2. Then, the process of hyperparameter tuning, a process to search for the good parameters for the network is performed by in 10 trials of random search on learning rate and regularization. Besides, K-Fold cross validation is also done to obtain the average accuracy for each random hyperparameter combinations with 30 epochs each fold. During K-Fold cross validation, training set was split into 5 folds where each fold will take turn to be the validation set. The hyperparameter combinations that achieve the highest validation accuracy is assumed to be the best suit. Finally, the training and testing accuracies are

obtained by using training and testing sets accordingly. This process is repeated for 5 times and the average accuracy is calculated from the results are shown in **Table 2-15** below.

Table 2-15 Results obtained from the experiment. Overall, there are K-fold cross validation is done on every trails out of 10 trials of hyperparameter tuning

Technique	**Average training accuracies from K-Fold cross validation (5 folds) in 5 trials**					**Average accuracy**
	1st	**2nd**	**3rd**	**4th**	**5th**	
KNN	-	-	-	-	-	-
Multiclass SVM	0.655000	0.687500	0.670000	0.685000	0.676250	0.674750
FCNN	0.361250	0.415000	0.398750	0.388750	0.388750	0.390500
CNN	0.698750	0.738750	0.763750	0.725000	0.763750	0.738000
Technique	**Average testing accuracies from K-Fold cross validation (5 folds) in 5 trials**					**Average accuracy**
	1st	**2nd**	**3rd**	**4th**	**5th**	
KNN	0.390000	0.390000	0.390000	0.390000	0.390000	0.390000
Multiclass SVM	0.485000	0.440000	0.435000	0.440000	0.425000	0.445000
FCNN	0.345000	0.330000	0.330000	0.330000	0.320000	0.331000
CNN	0.535000	0.515000	0.480000	0.525000	0.490000	0.509000

For KNN, the result for training set is not done because the training set is essentially just saving all data. In other words, there are no learning mechanism in KNN. It is redundant to test with training set as it is equivalent to comparing same dataset. Hyperparameter tuning is done using grid search where 10 possible K values such as 1, 2, 5 until 100. The result from grid search remains to be the same as 100 is selected to be the best. The final testing set accuracy of 39% is achieved. This is expected as KNN is just simple image matching, while the experiment dataset is diverse in terms of camera angle, colors etc. KNN may perform well if the datasets are similar. Training in KNN is relatively fast as its just saving data, O(1) while prediction is really slow where the time taken becomes longer as size of dataset grow, giving O(N). This is because every input is being matched against every trained data which is time consuming.

For Multiclass SVM, hyperparameter tuning is performed on both learning rate and regularization term using random search. Overall, the results obtained are quite consistent. However in SVM, the regularization term is quite crucial as it can affect the performance of the network as it punishes misclassification more severely. Due to fairness issue, every experiment is limited to 10 trails of parameter search. When compared to KNN, Multiclass SVM performed much better than KNN with an average testing accuracy of 44.50%. This is because Multiclass SVM does learn the weight from data training with gradient descent and the addition of regularization term improves generalization and the performance of the network on unseen data.

For deep learning by using fully connected neural network, this is a standard fully connected neural network performed poorly when compared to the others. This can be caused by the low volume of dataset. The number of parameters are much larger than the input. Not to miss out that fully connected neural network is very inefficient to train as the number of parameters is huge. In this case, the first layer has 9,440,256 parameters (3072 * 3072 + 3072 biases). It is too expensive to perform only fully connected network. This is unfavourable when the size of the image dataset grows larger.

For deep learning with a simple CNN, a standard CNN is built with addition of Convolution layer, MaxPooling layer and a Dropout layer followed by a Fully Connected layer and output layer. CNN was the best performer in this experiment with a final testing accuracy of 50.90%. As mentioned previously, one of the cons of CNN is that a huge amount of dataset is required for training a good network. It is a clear winner when it comes to predicting unseen data. With that being said, this experiment is considered success as the results collected are satisfied with the experiment settings.

In a nut shell, KNN is unfit to be an image classifier for a complex problems where there are more dataset classes. Same goes for a fully connected neural network, the computational cost is too expensive to perform which is unfavourable for use in a mobile platform. Ultimately, CNN is being chosen over Multiclass SVM as its superior performance and the area of improvement are wider. With a larger dataset and proper hyperparameter tuning, CNN can reach state-of-the-art results which is very useful and applicable to this project needs.

2.12.2 Experiment and evaluation on several models through transfer learning

An experiment of transfer learning was carried out with a total of 12 popular models that are pre-trained on ImageNet dataset which consist of 1000 classes. Since that it is a resource intensive task, the experiment was performed in a cloud platform named FloydHub (Floydhub, 2018). The specifications of GPU used is an Nvidia Tesla K80 with 12GB of VRAM. All experiments were carried out under same environment. The results are recorded in **Table 2-16** as shown.

Some of the settings are listed as below:

- Total classes: 10
- Size of dataset: 2000
- Epochs: 20
- Batch size: 128

Table 2-16 Results obtained from the experiment across 12 models including total number of parameters, size of trained model and its accuracy

	No. of params	Size (MB)	Accuracy (%)
ResNet50	23,608,202	90.6	79.75
InceptionV3	21,813,029	84.2	74.00
InceptionResNetV2	54,352,106	209.0	74.25
MobileNet	3,239,114	12.6	76.50
MobileNetV2	2,270,794	9.2	74.75
VGG16	134,301,514	512.0	82.50
VGG19	139,611,210	532.0	83.75
Xception	20,881,970	80.1	77.75
NASNetMobile	4,275,001	19.5	49.75
DenseNet121	7,047,754	28.0	78.50
DenseNet169	12,659,530	49.9	81.50
DenseNet201	18,341,194	71.9	85.00

Since that there were not much significant differences in accuracies between multiple attempts, the results were only for once as reference and guidance in selecting model. Then, each model was then embedded into a mobile device to examine the speed of model (delay in milliseconds for getting model results). The mobile device used remained unchanged throughout the experiment to ensure results consistency. As the model

to be selected is to be used in mobile devices which might consist certain restrictions in terms of memory usage, storage and processing power, model size and speed of inference will be prioritised over accuracy. The results are recorded in **Table 2-17** as shown below.

Table 2-17 Model performance in terms of speed (in milliseconds)

	1st test	2nd test	3rd test	Average
ResNet50	1581	1542	1617	1580
InceptionV3	2275	2209	2244	2243
InceptionResNetV2	-	-	-	-
MobileNet	528	552	509	530
MobileNetV2	419	512	493	475
VGG16	-	-	-	-
VGG19	-	-	-	-
Xception	2785	2670	2830	2762
NASNetMobile	1642	1456	1657	1585
DenseNet121	1675	1796	1738	1736
DenseNet169	3336	2768	2790	2965
DenseNet201	3968	3436	3416	3607

Throughout the experiment, there are some models simply crashed the mobile application because the mobile device ran out of memory and this situation is unfavorable as it should be compatible with as much devices as possible. Furthermore, some models such as VGG16 & VGG19 are way too large in size (500+ MB) and they are not suitable to be used in the case although the experiment achieved higher accuracy with them. Therefore, they were counted out of the selection process.

As expected, MobileNets outperformed others in terms of speed as they are specifically designed to be lightweight for use in mobile devices. MobileNetV2 is slightly better than its predecessor combined with its very small model size, speed and a satisfiable performance. It was chosen as it seems to be the best fit for the project. The experiment was then continued with increased dessert classes to 20 and proceeded with transfer learning on MobileNetV2 and achieved an accuracy of **73%** initially.

In order to further improve the performance, popular techniques such as image augmentation was applied as it helps when datasets are limited (Raj, 2018). More training data were added using image augmentation. One of the example of image augmentation outcome is as shown in **Figure 2-35** below.

Figure 2-35 Original image (left) and augmented image (right)

By adding more training data with random image augmentation (ex. rotation, flipping, shifting etc.) and experimenting with different parameters, the experiment ended up with an accuracy of **86.25%** for 10 classes of desserts and it proven that image augmentation is a useful technique to improve model performance.

CHAPTER 3 SYSTEM DESIGN

3.1 Project overview

3.1.1 Data preparation

The first part of project was to collect, process and prepare datasets required for later phase. In this project, images consisting of 30 kinds of desserts including both local and western were collected from the internet. Then, these images were filtered out manually. The purpose of this step is to remove images that are irrelevant and it is crucial in order to achieve a good results. Moving on, the dataset consisting of 4000 images is ready for use. However, the number of images was still insufficient to train a good model. In this case, image augmentation can be used in order to create a larger datasets which would improve the network performance (Raj, 2018). For instance, the image augmentation done was random flipping, zooming, shifting etc. of original image as shown in **Figure 3-1**. After that process, the augmented dataset now consist of 18000 images which is 5 times larger than its original ones. With this in mind, the new augmented images should only be used in training phase.

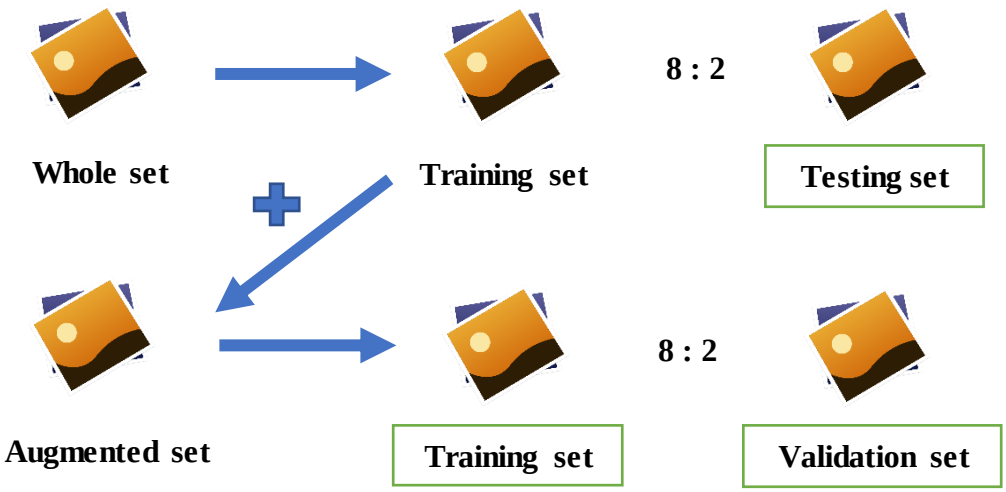


Figure 3-1 Summary of data preparation

As shown in **Figure 3-1** above, the original image dataset consisting of 4000 images were split at 8:2 ratio as training and testing set respectively. Then, the augmented

images were then appended to the training set followed by a split at 8:2 ratio as training and validation set. Finally, three datasets named training, testing and validation were prepared and ready for use in later.

On the other hand, all image labels are one-hot encoded as shown in **Table 3-1** below. Basically, the process is to convert labels to binary format instead of integers to represent each dessert classes.

Let (donut, ice cream, cookies) = (0, 1, 2)

Table 3-1 Example results of one-hot encoding

Label (class)	One-hot encoding
0	(1, 0, 0)
1	(0, 1, 0)
2	(0, 0, 1)

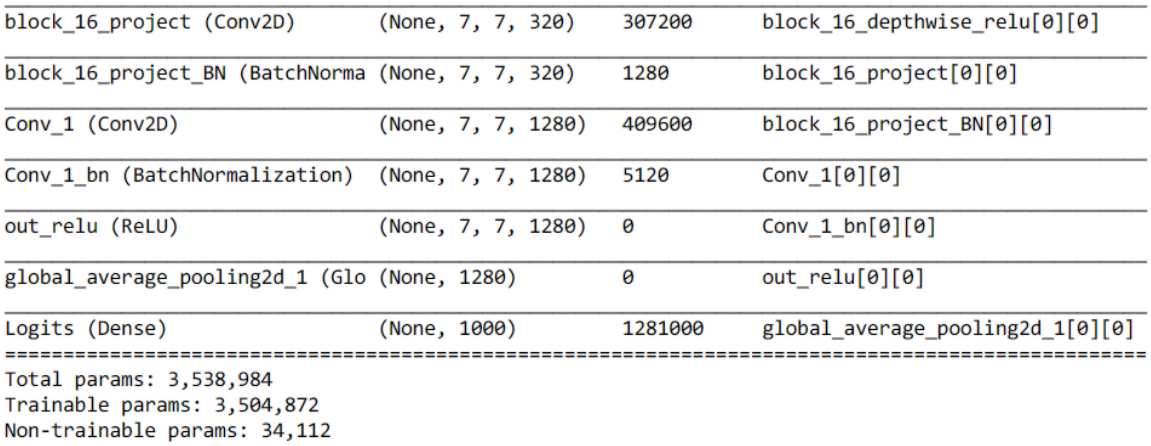
3.1.2 Fine-tuning MobileNetV2

The second part of the project was to select an existing model that is suitable for use in the project's case. In previous session, a CNN was defined and trained from scratch. However, the performance was unfavourable as the experiment only managed to achieve an estimated accuracy of 50%. It is time-consuming and resource intensive to define and train a good model from scratch. Therefore, approach of transfer learning was applied. However, there are many models available and a model selection is required. In order to select the best model for the project, an experiment was carried out as written in **Chapter 2.12.2** where a total of 12 popular models that have high reputation were evaluated in terms of their performance, size and inference speed in mobile devices.

The experiment served as a guidance in model selection and MobileNetV2 stood out among others and was deemed to be the best fit for the project. The published paper for MobileNetV2 was studied in order to gain better understanding as written in **Chapter 2.11.4**. The weights and model architecture were downloaded from Keras API. Then, only few of the uppermost layers were being trained while the rest of the layers were all frozen.

This helps as the earlier layers learnt simple and general features and these knowledge can be transferred to solve other problems. As mentioned in **Chapter 2.12.2**, the model manage to achieve an estimated accuracy of 85.86% approximately with 10 kinds of dessert. This further strengthen the benefits of transfer learning.

Using MobileNetV2, the initial total trainable parameters is 3,504,872 as seen in **Figure 3-2**. As the MobileNetV2 was pre-trained on ImageNet dataset which consist of

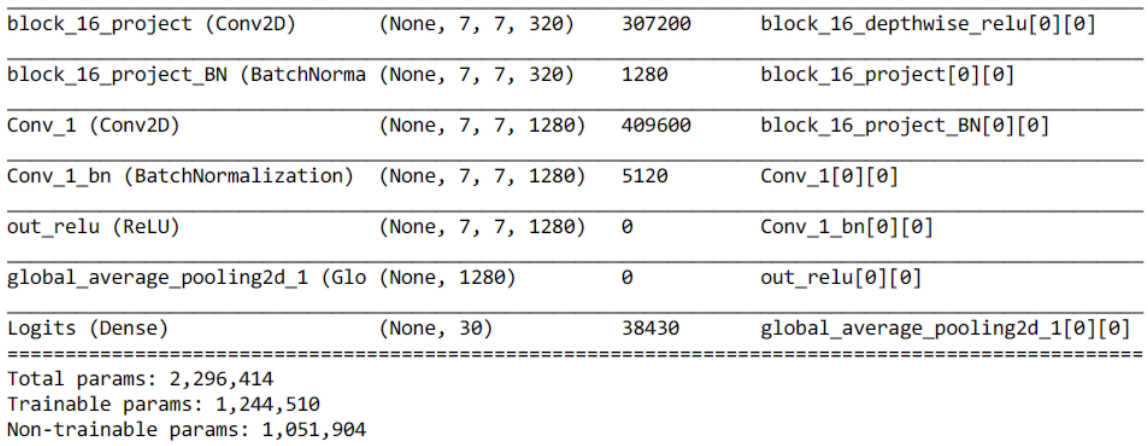


block_16_project (Conv2D)	(None, 7, 7, 320)	307200	block_16_depthwise_relu[0][0]
block_16_project_BN (BatchNorma	(None, 7, 7, 320)	1280	block_16_project[0][0]
Conv_1 (Conv2D)	(None, 7, 7, 1280)	409600	block_16_project_BN[0][0]
Conv_1_bn (BatchNormalization)	(None, 7, 7, 1280)	5120	Conv_1[0][0]
out_relu (ReLU)	(None, 7, 7, 1280)	0	Conv_1_bn[0][0]
global_average_pooling2d_1 (Glo	(None, 1280)	0	out_relu[0][0]
Logits (Dense)	(None, 1000)	1281000	global_average_pooling2d_1[0][0]

Total params: 3,538,984
Trainable params: 3,504,872
Non-trainable params: 34,112

Figure 3-2 Screenshot of last few layers of MobileNetV2 architecture using Keras API before modification

1000 classes, the first step was to modify the last layer with the number of classes the project will have. In this case, the project will be covering 30 classes of desserts. The diversity of features learnt from ImageNet can be reused to solve different problems. In order to achieve that, the earlier layers of the architecture must be frozen and not be trained on new datasets. This is because the features learnt in earlier layers are general and not as



block_16_project (Conv2D)	(None, 7, 7, 320)	307200	block_16_depthwise_relu[0][0]
block_16_project_BN (BatchNorma	(None, 7, 7, 320)	1280	block_16_project[0][0]
Conv_1 (Conv2D)	(None, 7, 7, 1280)	409600	block_16_project_BN[0][0]
Conv_1_bn (BatchNormalization)	(None, 7, 7, 1280)	5120	Conv_1[0][0]
out_relu (ReLU)	(None, 7, 7, 1280)	0	Conv_1_bn[0][0]
global_average_pooling2d_1 (Glo	(None, 1280)	0	out_relu[0][0]
Logits (Dense)	(None, 30)	38430	global_average_pooling2d_1[0][0]

Total params: 2,296,414
Trainable params: 1,244,510
Non-trainable params: 1,051,904

Figure 3-3 Screenshot of last few layers of MobileNetV2 architecture using Keras API after modification

specific as those learnt from last few layers. As a result of several attempts, the last 6 layers were being unfroze (kept their weights trainable) while the rest were all frozen. This reduces the total trainable parameters to 1,244,510 as seen in **Figure 3-3**.

Nonetheless, all the layers configurations were kept default. At this stage, the model should be ready to be trained on the new dessert datasets that were prepared.

3.1.3 Training phase

The training phase is a time-consuming process and it requires heavy computing power. First of all, all images were resized into resolution of 224x224 with 3 colour channels (RGB format) each. This is the default input resolution as recommended by MobileNetV2. Then, training and validation sets that were prepared previously were being used accordingly during the training session.

In addition, there will be times where training loss started to increase and accuracy degrades after certain epochs of steady improvements. An example of this situation was shown in **Figure 3-4**. This can be solved by applying a technique called early stopping.

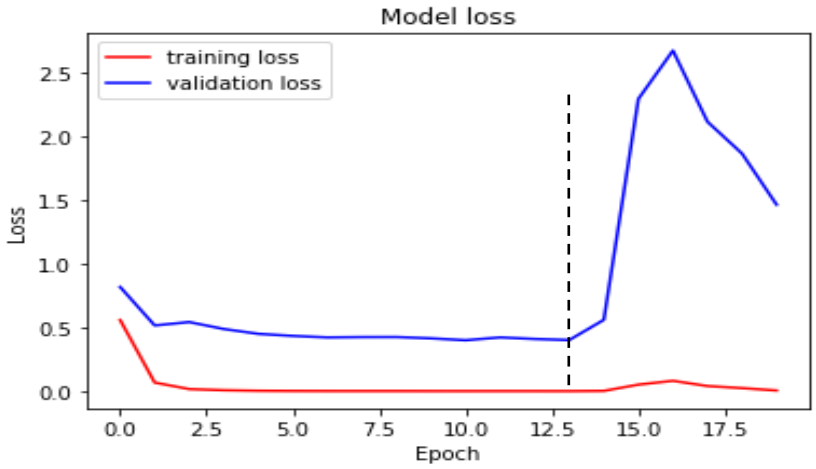


Figure 3-4 Model loss stopped improving after steady decrement

Criteria can be made to stop the training process once the training loss stopped improving such as stop training after no improvements were being seen after 2 epochs. This helps to secure the best model from training phase.

3.1.4 Performance evaluation and hyperparameter tuning

After the training phase, the model accuracy was evaluated with testing set. The hyperparameters were being fine-tuned in order to optimize the performance. Some of them including learning rates, batch size etc. Besides, the number of layers to unfreeze as seen in **Table 3-3** would also have a significant impact on the outcome of model performance. The process of performance evaluation and hyperparameter tuning were repeated until a satisfied model accuracy was achieved.

3.1.5 Deployment of model

After a satisfied model accuracy is achieved, the model architecture together with its weights were all frozen, saved and converted into constants using TensorFlow library. The model is then bundled into assets folder of the Android application and it is ready for inference.

3.1.6 Building SQLite database

A SQLite database was built to store dessert information. This database was also bundled into the application package. This allows the application to function even without internet connection. Any updates to database would just refer to version number in SQLite. The database will consist of information such as dessert name and calorie. Moreover, it will only be initialized for once during the first startup of application. **Table 3-2** shows the dessert database built with SQLite.

Table 3-2 Dessert database

Dessert
Name
Calorie
Standard serving size (grams)

3.1.7 Capturing image and inference

During usage of the application, the mobile device camera will act as the input to the model. Every time a frame is ready, it is resized and process into proper format with channel at last. Then, the bitmap is fed into the model. The output results will be an array

of prediction scores. This classifies the dessert type captured from the camera. The array was sorted and the class with highest score will be assumed the detected dessert type. Its relevant information will then be queried from the SQLite database and computation of calories based on several standard serving sizes will be done. Finally, the output will be displayed to user.

3.2 Project flow and summary

For realization of the project, the flow of the project can be summarized into two parts as shown in **Figure 3-5** and **Figure 3-6** below. In short, first part was carried out in a desktop environment while the latter in a mobile device.

3.2.1 Part one (in desktop)

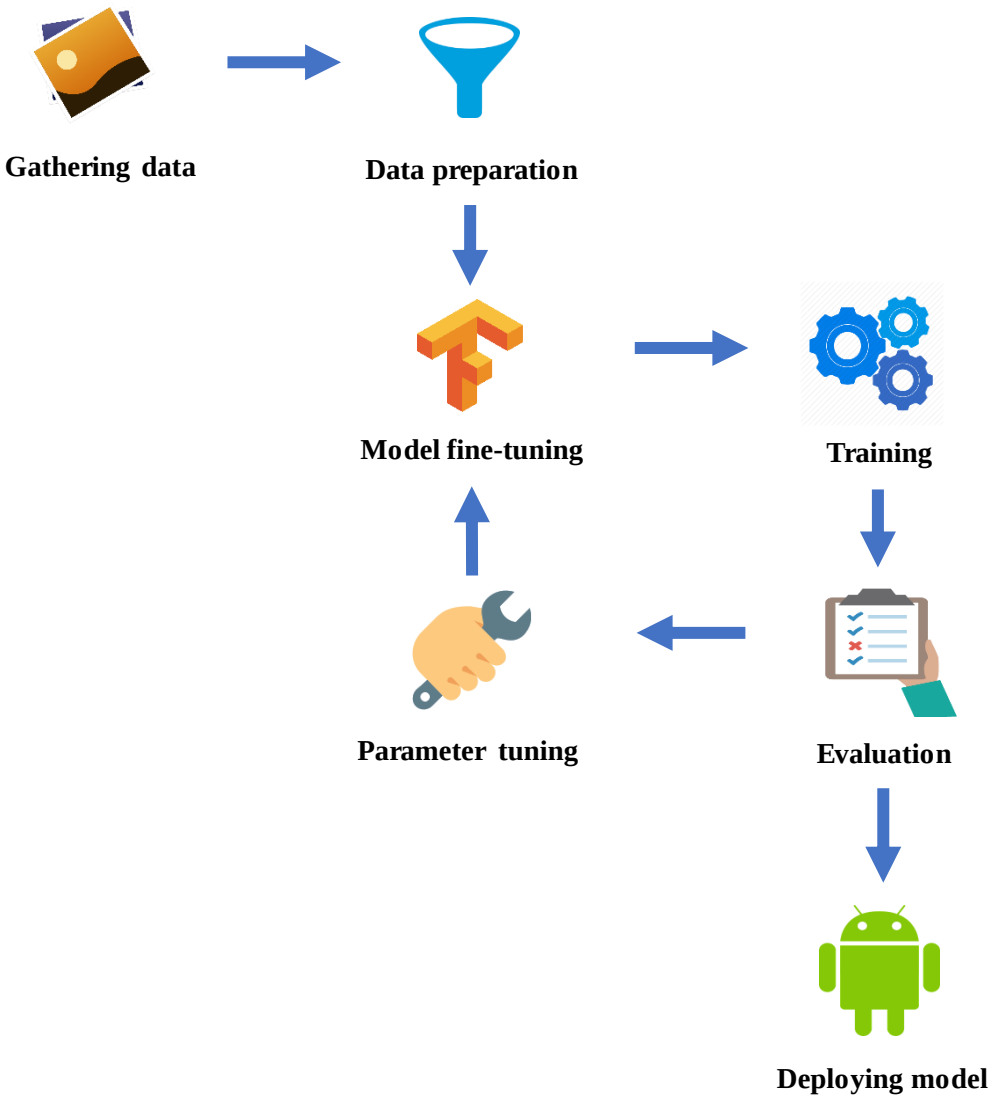


Figure 3-5 Overview of system design for building an image classifier

3.2.2 Part two (in mobile device)

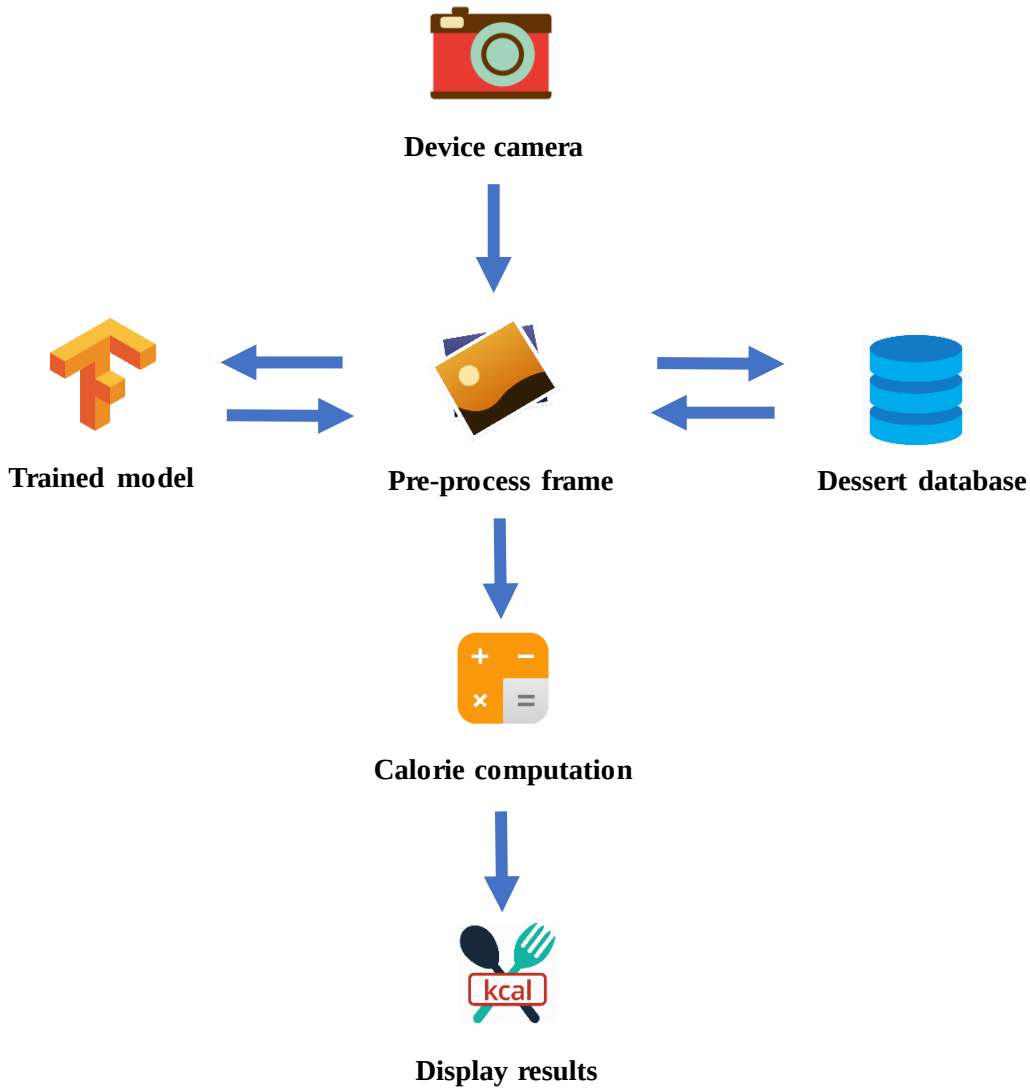


Figure 3-6 Overview of system design for the mobile application

3.3 Detailed flow of mobile application

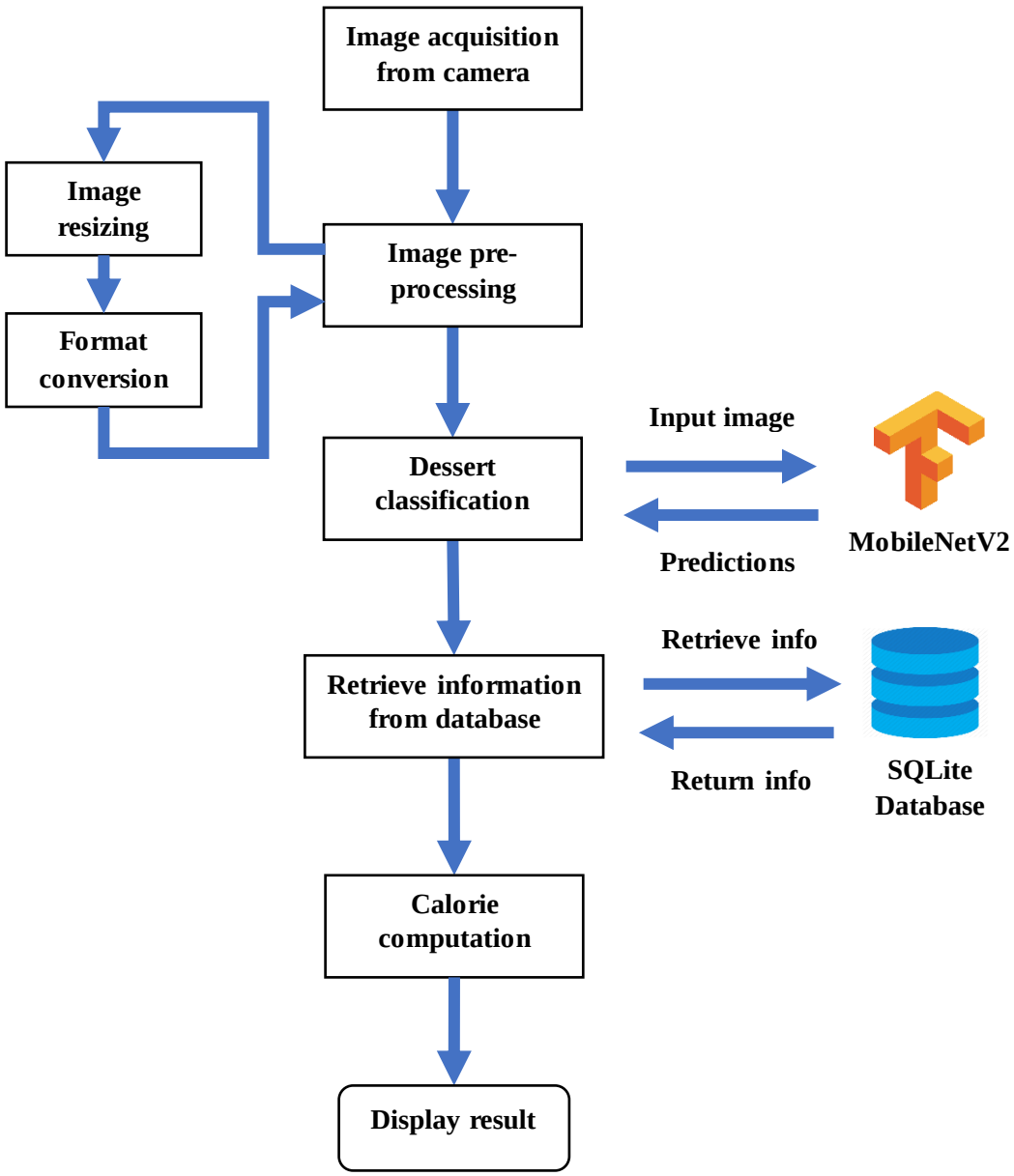


Figure 3-7 Detailed flow of the mobile dessert calorie counter application

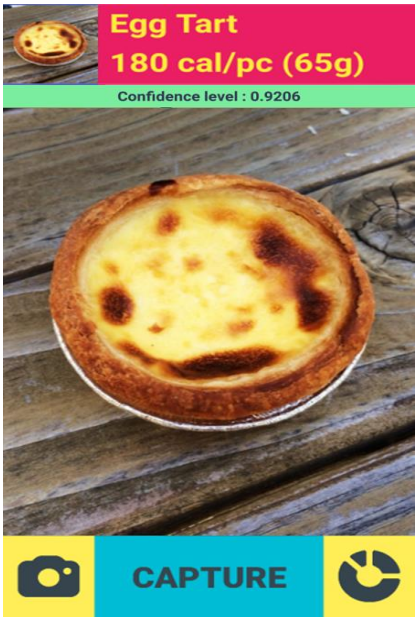


Figure 3-8 Example of good camera position

Step 1: Image acquisition

- For the best result, user should try to locate and position the target food within the camera frame.
- The whole portion of the food should be present within the frame without any blockage.
- The phone should be held at approximately 45 degrees towards the food placed on a horizontal surface and located the food so that it is just within the frame.
- An example of a good camera position is shown in **Figure 3-8**.

Step 2: Image pre-processing

- Every time a new image frame is acquired, certain pre-process need to be done before it is ready for inference.
- To start with that, each image frame is first resized into resolution of 224x224 according to the default set by MobileNetV2.
- Then, the array is arranged so that image channels are ordered last. The desired image format will be ordered by batch size, image width, image height and number of colour channels.
- Finally, the array is flatten into a two dimensional array.

Step 3: Dessert classification

- The trained TensorFlow model will be used to perform dessert classification.
- This is achieved through the use of TensorFlow Inference Inteface API which acts as a bridge between TensorFlow model and the Android application.
- The pre-processed bitmap image is fed into the model for inference and the output predictions will be an array of scores.
- The scores are sorted and the dessert class who scored the highest will be the final output

Step 4: Retrieve information from database

- Since the dessert classes in the SQLite database is ordered the same as the model output, the index of classes who scored highest will be used to query for its relevant information

Step 5: Calorie computation

- The base dessert calorie of the predicted dessert is obtained.
- For several serving sizes, the calorie are computed accordingly and the final results are displayed to user

3.4 Flow inside MobileNetV2

Figure 3-9 is an example bottleneck residual block used in MobileNetV2. Basically, the block is built from several convolutional layers. Each projection (bottleneck) layer were followed by batch normalization while the rest of the convolution layers were all followed by batch normalization and ReLU non-linearity. The overall MobileNetV2 architecture is built from stacking and repetition of these blocks together with their residual connections as shown in **Figure 3-10**.

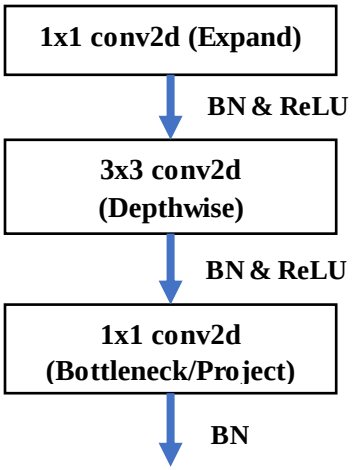


Figure 3-9 An example of bottleneck residual block in MobileNetV2

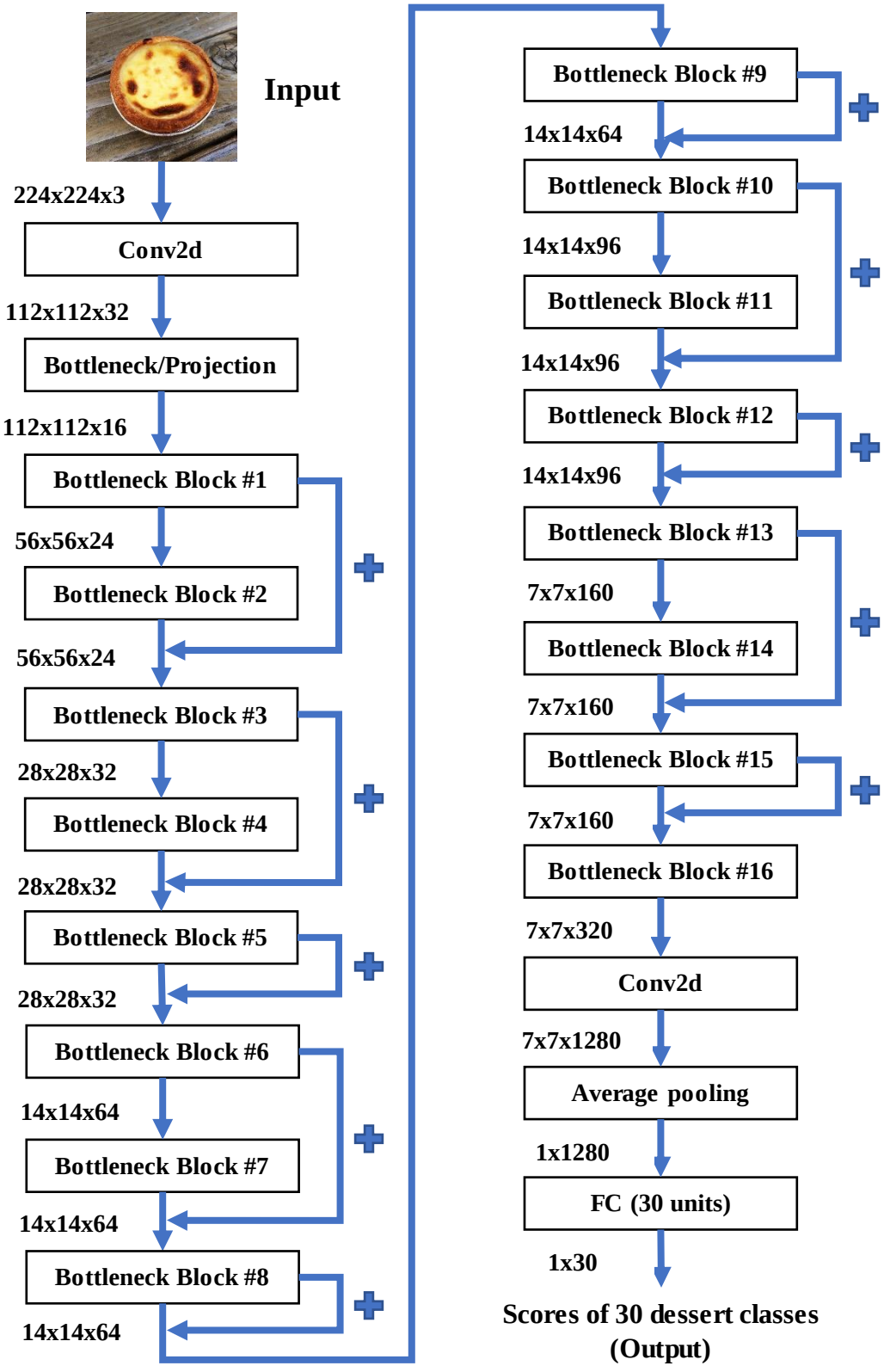


Figure 3-10 Flow of information inside MobileNetV2 showing the output dimension changes and residual connections going up the network

CHAPTER 4 METHODOLOGY, TOOL & TESTING

4.1 Methodologies

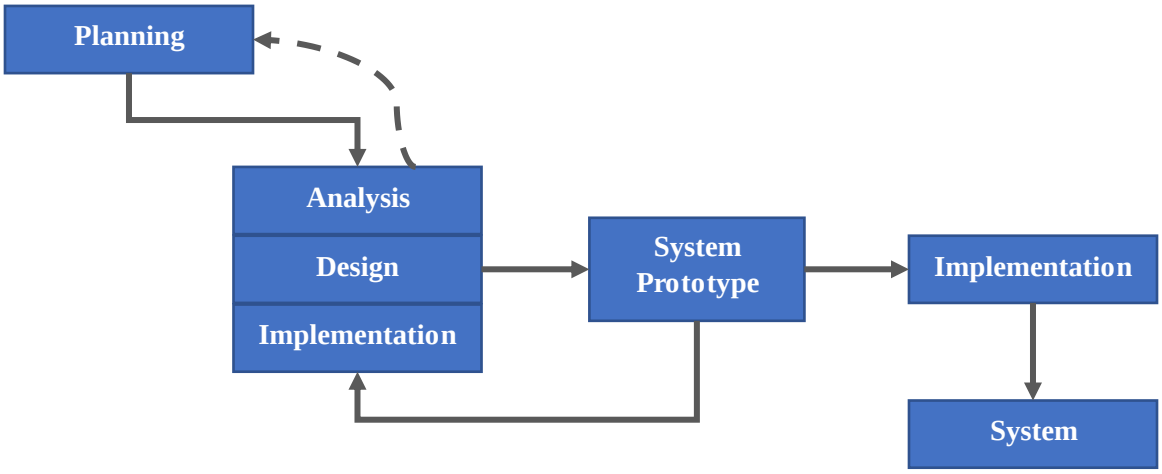


Figure 4-1 Prototyping methodology (The WritePass Journal, 2018)

Planning Phase

In the planning phase, the project is determined to take a duration of 2 long semesters which is around 10 months to complete.

Analysis Phase

In analysis phase, the problem statement, project objectives and scopes are identified. The project background is reviewed together with some researches performed to understand the strengths and weakness of previous approaches towards the problem.

Design Phase

In design phase, the overall structure and flow of the system is determined. This includes the structure of the convolutional neural network model, design of mobile application user interface and database structure. Besides, the hardware requirements are also identified.

Implementation Phase

In implementation phase, the actual coding is carried out according to the design chosen.

System Prototype

System prototypes are created to acquire feedback of its performance. The iterations of analysis, design and implementation are repeated to create several prototypes for evaluation until a satisfied result is obtained.

4.2 Tools and software

- Python 3.6
- Java
- Jupyter notebook

  Great support of many programming languages, produce rich interactive output and nice visualization of data (Project Jupyter, 2018)
- OpenCV for Python

  Open source computer vision software library (OpenCV team, 2018)
- Android Studio 3

  An office IDE designed by to build apps for every single Android devices (Android Studio, 2018)
- SQLite Database

  A self-contained, high-reliability, fully featured and one of the most used database engine (SQLite, 2018)
- TensorFlow

  Open source machine learning framework (TensorFlow, 2018)
- TensorBoard

  Visualization tool to visualize graph for better understanding and debugging (TensorBoard Visualizing Learning, 2018)
- Keras

  High-level API that runs on TensorFlow backend allows for more efficient and consistent implementation (Keras, 2018)
- FloydHub

  A popular cloud platform designed for training and deploying deep learning models (FloydHub, 2018)

4.3 User requirements

- Android version higher than 5.0 or API level 21

According to minimum user requirement for use of camera2 API (Android 5.0 API, 2018)

- Minimum available storage of 150MB

  According to compiled application package (.apk)

4.4 Implementation and testing

The trained model on 30 kinds of dessert was being tested on some dessert images to verify its accuracies. Some examples on the model confidence level on its classification was being shown in **Figure 4-2**. Overall, the model performed well under situation between desserts that have similar appearance. However, there are circumstances where the model wrongly classified dessert when only partial portion of dessert was present. This was expected as the majority of images in training dataset are captured in full portion.

Figure 4-2 Example of model performance on desserts with similar appearance

In order to search for the best number of layers in MobileNetV2 to unfreeze or made trainable. A test was performed and results were recorded into **Table 3-3**.

Table 3-3 No. of layers made trainable vs test accuracy

No. of layers made trainable	**Test accuracy (%)**
Freeze layers [, : -3]	87.00
Freeze layers [, : -4]	85.50
Freeze layers [, : -5]	91.00
Freeze layers[, : -13] (Unfreeze 1 bottleneck residual block)	93.33
Freeze layers[, : -22] (Unfreeze 2 bottleneck residual blocks)	95.25
Freeze layers[, : -31] (Unfreeze 3 bottleneck residual blocks)	95.08
Freeze layers[, : -40] (Unfreeze 4 bottleneck residual blocks)	77.92

As seen from Table, the performance started to degrade after unfreezing certain amount of layers. An assumption was being made the trend will continue as so. As a result of the test, the fifth settings was being applied to the project where only the last 22 layers in MobileNetV2 were being made trainable.

CHAPTER 5 CONCLUSION

5.1 Project review, discussions & conclusions

Uncontrolled and imbalanced calorie intake are both main sources of problem that are causing overweight and obesity in Malaysia. Dessert played an important part of the causes of this due to its high sugar content that leads to high calorie. However, these problems are not being taken seriously enough from the people. Aside from lack of awareness of the problem, one of the cause could be difficulty in acquiring food calorie values as some local or traditional desserts are not very well documented. Traditional ways of food logging is inefficient and discouraging as it requires a lot of effort to perform.

Recent rise in popularity of deep learning had raised interest among many people in the field of machine learning. Not only it outperforms traditional machine learning techniques, it has a wide area of application in different fields. At the same time, rapid progression of smartphone development had made them much more affordable and increased capability to perform heavy computation locally.

This project proposed a mobile application that applied deep learning to acquire dessert calorie. Not only it provides an easier way to access information of food but could also serves as an educational tool for people to learn more about desserts. In a nutshell, the project aims to promote a healthy lifestyle by raising awareness among people towards the importance of balanced calorie intake. Hence, it can reduce the problem that the country is currently facing. Overall, the project have a potential to grow into a more usable tool as the classes of dessert that it is able to identify increases.

5.2 Novelties and contributions achieved

The project could be used in many cases. For instance, foreigners or tourist could use it to easily learn about local desserts without extensively digging up resources throughout the internet. While it could be served as an educational tool, it can also be used as a tool to manage calorie intake. As the country's alarming problem of obesity and overweight are arising, normal people could use it as a guidance to manage their calorie intake. Besides, this project could raise public awareness towards the impact of dessert

towards our daily calorie intake. Moreover, this project could possibly eliminate traditional ways of food logging as there is no need to search for food information manually anymore. This project is unique as it also addressed local dessert where most of them are not very well documented and there are not any tool of its kind in the market yet.

5.3 Future Work

There are many improvements that could be added to enhance the project in many aspects. One of them could be implementation of object detection so that multiple dessert objects can be detected and sum of calories can be computed instead. Moreover, model performance can be further improved in terms of accuracy and dessert variety. This would help to extend the usability of the project. Nonetheless, the implementation of application can also be further optimised especially the frames per second during real-time mode in order to produce a smoother user experience.

REFERENCES

Akbari Fard, M., Hadadi, H. and Tavakoli Targhi, A. 2016, 'Fruits and Vegetables Calorie Counter Using Convolutional Neural Networks', *Proceedings of the 6th International Conference on Digital Health Conference - DH '16*, pp.121-122.

All About Web 2018, *How Flexible is Python?*. Available from: <http://www.allaboutweb.biz/how-flexible-is-python/>. [02 March 2018].

Android 5.0 APIs 2018, Available from: <https://developer.android.com/about/versions/android-5.0.html>. [02 March 2018].

Android Studio 2018, *The Official IDE For Android*. Available from: <https://developer.android.com/studio/index.html>. [03 March 2018].

Auria, 2012, *Appom Balik*. Available from: <http://auriasmalaysiankitchen.com/wp-content/uploads/2012/06/IMG_05611.jpg>. [14 August 2017].

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. 2008, 'Speeded-Up Robust Features (SURF)', *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346-35.

Brilliant.org 2018, *Backpropagation | Brilliant Math & Science Wiki*. Available from: <https://brilliant.org/wiki/backpropagation/>. [03 March 2018].

Britz, D. 2015, *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. Available from: <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>. [03 March 2018].

Britz, D. 2015, *Understanding Convolutional Neural Networks for NLP*. Available from: <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for- nlp/>. [11 August 2017].

Brownlee, J. 2017, *A Gentle Introduction to Transfer Learning for Deep Learning*. Available from: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>. [20 July 2018].

Bruno, D. and Osorio, F. 2017, 'Image classification system based on deep learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes', *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pp. 1-6.

Caspi, G. 2017, *What's the difference between deep learning and machine learning*. Available from: < https://betanews.com/2016/12/12/deep-learning-vs-machine-learning/>. [02 February 2018].

Categorised Body Mass Index, 2017. Available from: <http://www.weightofthenation.org/wp- content/uploads/2016/02/bmi-overweight-obesity-men-and-women.png?3890d0>. [14 August 2017].

Chioka.in. 2018, *Differences between L1 and L2 as Loss Function and Regularization*. Available from: <http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/>. [03 March 2018].

Chollet, F. 2017, 'Xception: Deep Learning with Depthwise Separable Convolutions', pp. 1-8.

Codementor 2018, *Best Programming Language For Me in 2016*. Available from: <http://www.bestprogramminglanguagefor.me/why-learn-python>. [02 March 2018].

Conason, A. 2018, *Sugar Addiction: Part Deux*. Available from: <https://www.psychologytoday.com/us/blog/eating-mindfully/201210/sugar-addiction-part-deux>. [21 July 2018].

Cs231n.github.io 2018, *CS231n Convolutional Neural Networks for Visual Recognition*. Available from: <http://cs231n.github.io/convolutional-networks/>. [03 March 2018].

Cs231n.github.io. 2018, *CS231n Convolutional Neural Networks for Visual Recognition*. Available from: <http://cs231n.github.io/transfer-learning/>. [20 July 2018].

Deeplearning.net 2018, *Welcome — Theano 1.0.0 documentation*. Available from: <http://deeplearning.net/software/theano/>. [03 March 2018].

Deshpande, A. 2016, *A Beginner's Guide To Understanding Convolutional Neural Networks Part 2*. Available from: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>. [03 March 2018].

Economist Intelligence Unit 2017, *Tackling obesity in ASEAN Prevalence, impact, and guidance on interventions*. Available from: < https://www.eiu.com/public/topical_report.aspx?campaignid=ObesityInASEAN>.

FatSecret Platform API, 2017. Available from: <https://platform.fatsecret.com/api/>. [12 August 2017].

Floydhub 2018, *FloydHub - Deep Learning Platform - Cloud GPU*. Available from: <https://www.floydhub.com/>. [21 July 2018].

Greene, T. 2018, *Google brings on-device machine learning to mobile with TensorFlow Lite*. Available from: <https://thenextweb.com/artificial-intelligence/2017/11/15/google-brings-on-device-machine-learning-to-mobile-with-tensorflow-lite/>. [03 March 2018].

Gunnars, K. 2017, *How Many Calories Should You Eat Per Day to Lose Weight?*. Available from: <http://www.healthline.com/nutrition/how-many-calories-per-day#section2>. [Accessed 14 August 2017].

Hariadi, R. R., Khotimah, W. N. and Wiyono, E. A. 2015, 'Design and implementation of food nutrition information system using SURF and FatSecret API', *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, pp. 181-183.

Haslam, D. and James, W. 2005, 'Obesity', *The Lancet*, vol. 366, no. 9492, pp.1197-1209.

He, K., Zhang, X., Ren, S., Sun, J. 2016, 'Deep Residual Learning for Image Recognition', *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-777.

How to Retrain Inception's Final Layer for New Categories, 2017. Available from: < https://www.tensorflow.org/tutorials/image_retraining>. [17 August 2017].

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. 2017, 'MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications', pp. 1-9.

Huang, G., Liu, Z., Maaten, L.V.D., Weinberger, K.Q. 2016, 'Densely Connected Convolutional Networks', pp. 1-9.

Hulstaert, L. 2018, *Going deep into image classification*. Available from: < https://towardsdatascience.com/an-overview-of-image-classification-networks-3fb4ff6fa61b>. [27 July 2018].

Image Recognition, 2017. Available from: < https://www.tensorflow.org/tutorials/image_recognition>. [17 August 2017].

Ipfs.io. 2018, *Hinge loss*. Available from: <https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Hinge_loss.html>. [03 March 2018].

Johnson, J., 2013, *Cheesecake*. Available from: <http://img.aws.livestrongcdn.com/ls-article-image-400/cpi.studiod.com/www_livestrong_com/photos.demandstudios.com/getty/article/185/132/medfrd3804_XS.jpg>. [16 August 2017].

Johnson, R., Appel, L., Brands, M., Howard, B., Lefevre, M., Lustig, R., Sacks, F., Steffen, L. and Wylie-Rosett, J. 2009, 'Dietary Sugars Intake and Cardiovascular Health: A Scientific Statement From the American Heart Association', *Circulation*, vol. 120, no. 11, pp. 1011-1020.

K Nearest Neighbors – Classification, 2018. Available from: <http://www.saedsayad.com/k_nearest_neighbors.htm>. [03 March 2018].

Karpathy 2018, *The Unreasonable Effectiveness of Recurrent Neural Networks*. Available from: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>. [03 March 2018].

Kawano, Y. and Yanai, K. 2013, 'Real-Time Mobile Food Recognition System', *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1-7.

Kdnuggets.com 2018, *Keep it simple! How to understand Gradient Descent algorithm*. Available from: <https://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html>. [04 March 2018].

Keras 2018, *Keras Documentation*. Available from: <https://keras.io/>. [03 March 2018].

Keras 2018, *Applications - Keras Documentation*. Available from: <https://keras.io/applications/>. [21 July 2018].

Lau, S. 2017, *Image Augmentation for Deep Learning – Towards Data Science*. Available from: <https://towardsdatascience.com/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2>. [10 July 2018].

Law, S.P., 2016, *Kuih Seri Muka*. Available from: <http://i.hungrygowhere.com/cms/a1/a6/16/6b/20023671/resipi-kuih-seri-muka_566x424_fillbg_fde0c1fc25.jpg>. [14 August 2017].

Le, Q., Zoph, B. 2017, *Using Machine Learning to Explore Neural Network Architecture*. Available from: <https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html>. [27 July 2018].

Moorhead, R., Bond, R. and Zheng, H. 2015, 'Smart food: Crowdsourcing of experts in nutrition and non-experts in identifying calories of meals using smartphone as a potential tool contributing to obesity prevention and management', *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1777-1779.

OpenCV team 2018, *About*. Available from: <https://opencv.org/about.html>. [03 March 2018].

Pennlio 2014, *Fully Connected Neural Network*. Available from: <https://pennlio.files.wordpress.com/2014/04/nn-mxe.png?w=402&h=443&zoom=2>. [03 March 2018].

Perceptron, 2016. Available from: <https://appliedgo.net/media/perceptron/neuron.png>. [03 March 2018]

Plain Roti Canai, 2017. Available from: <http://www.thestar.com.my/~/media/online/2016/03/05/14/51/roti-canai.ashx/?w=620&h=413&crop=1&hash=7C6E98334F403C575A822CE179B734625FB24154>. [14 August 2017].

Python.org 2018, *Welcome to Python.org*. Available from: <https://www.python.org/>. [02 March 2018].

Project Jupyter 2018. Available from: <http://jupyter.org/>. [03 March 2018].

Raj, B. 2018, *Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2*. Available from: <https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced>. [07 July 2018].

Ranard, B. L., Yoonhee, P., Meisel, Z. F., Asch, D. A., Hill, S. S., Becker, L. B., Seymour, A.K., Merchant, R.M. 2013, 'Crowd sourcingHarnessing the Masses to Advance Health and Medicine, a Systematic Review', *Journal of General Internal Medicine*, vol. 29, no. 1, pp. 187-203.

Rosebrock, A. 2018, *Multi-class SVM Loss - PyImageSearch*. Available from: <https://www.pyimagesearch.com/2016/09/05/multi-class-svm-loss/>. [03 Mar. 2018].

Rother, C., Kolmogorov, V. and Blake, A. 2004, 'GrabCut: interactive foreground extraction using iterated graph cuts', *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309-314.

Satu Johor, 2016, *Nasi Lemak*. Available from: <http://www.satujohor.net/wp-content/uploads/2016/10/img_0050.jpg>. [14 August 2017].

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C. 2018, 'MobileNetV2: Inverted Residuals and Linear Bottlenecks', pp. 1-8.

Scikit-learn.org. 2018, *1.6. Nearest Neighbors — scikit-learn 0.19.1 documentation.* Available from: <http://scikit-learn.org/stable/modules/neighbors.html>. [03 March 2018].

Simonyan, K., Zisserman, A. 2015, 'Very Deep Convolutional Networks For Large-Scale Image Recognition', pp. 1-14.

Sqlite 2018, *SQLite Home Page*. Available from: <https://www.sqlite.org/index.html>. [03 March 2018].

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A. 2016, 'Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning', pp.1-12.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. 2015, 'Going deeper with convolutions'. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 1-9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. 2016, 'Rethinking the inception architecture for computer vision'. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Tammachat, N. and Pantuwong, N. 2014, 'Calories analysis of food intake using image recognition', *2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 1-4.

TensorFlow 2018, *TensorFlow*. Available from: <https://www.tensorflow.org/>. [03 Mar. 2018].

TensorBoard: Visualizing Learning 2018, Available from: <https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard>. [03 March 2018].

The WritePass Journal 2018, *Prototyping Methodology – The WritePass Journal*. Available from: <https://writepass.com/journal/2012/12/evaluation-of-mendix-as-a-rapid-application-development-rad-tool/prototyping-methodology/>. [3 March 2018].

Torrey, L., and Shavlik, J. 2009, 'Transfer learning', *Handbook of Research on Machine Learning Applications*, vol. 3, pp. 17-35.

Urban, T. 2018, *Neuralink and the Brain's Magical Future - Wait But Why*. Available from: <https://waitbutwhy.com/2017/04/neuralink.html>. [20 July 2018].

Vanilla Cake, 2017. Available from: <https://dessertswithbenefits.com/wp-content/uploads/2014/07/Healthy-Low-Carb-Gluten-Free-Vanilla-Cake-e1488056067949.jpg>. [15 August 2017].

Walia, A.S. 2017, *The Vanishing Gradient Problem – Anish Singh Walia – Medium*. Available from: <https://medium.com/@anishsingh20/the-vanishing-gradient-problem-48ae7f501257>. [24 July 2018].

Wicke, M, 2016, *Inception v3 architecture*. Available from: < https://github.com/tensorflow/models/blob/master/inception/g3doc/inception_v3_architec ture.png>. [17 August 2017].

Woolford, S. J., Clark, S. J., Strecher, V.J., Resnicow, K. 2010, 'Tailored mobile phone text messages as an adjunct to obesity treatment for adolescents', Journal of Telemedicine and Telecare vol. 16, no. 8, pp. 458-461.

World Health Organization 2016, *Obesity and Overweight*. Available from: <http://www.who.int/mediacentre/factsheets/fs311/en/>. [15 August 2017].

Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V. 2018, 'Learning Transferable Architectures for Scalable Image Recognition', pp. 1-14.