A Cloud-Assisted Mobile Food Recognition System

By

Parisa Pouladzadeh

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the Doctor of Philosophy degree

in

Electrical and Computer Engineering

Ottawa-Carleton Institute for
Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

Accurate instruments and methods for measuring food and energy intake are crucial in the battle against obesity. Providing users and patients with convenient, intelligent solutions that help them measure their food intake and collect dietary information is valuable for long-term prevention and successful treatment programs. In this thesis, we propose an assistive calorie measurement system to help patients and doctors succeed in their fight against diet-related health conditions. Our proposed system runs as an application on smartphones that automatically measures the calorie intake based on a picture of the food taken by the user.

The key functions of our application involve image segmentation, image processing, and food classification and recognition. Client-side devices (e.g., smartphones, tablets) do have some limitations in their ability to handle the time-sensitive and computationally intensive algorithms required for our application. The image processing and classification algorithms used for food recognition and calorie measurement consume device batteries quickly, which is inconvenient for the user. It is also very challenging for client-side devices to scale the large amount of data and images that is needed for food recognition. The entire process is time consuming, inefficient, and frustrating for the users, and may deter them from using the application.

In this thesis, we address these challenges by proposing a cloud-assisted mobile food recognition system. Our results show that the accuracy of the recognition step within this cloud-assisted application, compared to Support Vector Machine (SVM), is increased in single food portions, non-mixed plates, and mixed plates of food. In addition, by applying a deep neural network, the accuracy of food recognition in single food portions is increased to 100%.

However, in real-life scenarios, it is common for a food image to contain more than one food item, which reduces recognition accuracy. To solve this problem, we propose a novel

method that detects both food item combinations and their locations in the image with minimal supervision. In our dataset, images with only one food item are used as the training dataset, while images with multiple food items are used as the test dataset. In the training stage, we began by using region proposal algorithms to generate candidate regions and extract the convolutional neural network (CNN) features of all regions. Second, we performed region mining to select positive regions for each food category using maximum cover in our proposed submodular optimization method. In the testing stage, we began by generating a set of candidate regions. For each region, a classification score was computed based on its extracted CNN features and the application predicted food names of the selected regions. Our experiments, conducted with the FooDD dataset, show an average recall rate of 90.98%, a precision rate of 93.05%, and an accuracy rate of 94.11%, compared to 50.8%–88% accuracy in other existing food recognition systems.

The analysis and implementation of the proposed system are described in the following chapters.

Acknowledgments

First and foremost I would like to express my deepest gratitude to my supervisor, Professor Shervin Shirmohammadi, for assigning this project to me which is intended to help people with diabetes and to improve health system. Without his support, the completion of this thesis would not be possible.

I had the opportunity to work as part of a great team during the course of my thesis. The people involved in the project gave me a better perspective and new knowledge. I would like to thank all of my colleagues in the DISCOVER lab, especially Dr. Abdulsalam Yassine, for providing valuable information and help.

Finally, I owe my loving thankfulness to my family and I thank my husband, Mehdi, for his kind help and patience.

Table of Contents

List of Figures

List of Tables

Glossary of Terms

2D	Two Dimensional Space
3D	Three Dimensional Space
MCC	Mobile cloud-computing
NN	Neural Network
FIVR	Visual and Voice Recogniser
NIR	Near Infrared
PDA	personal digital assistive
ReLU	rectified linear unit
PDE	Partial Differential Equation
RFID	Radio Frequency Identification
RBF	Radial Basis Function
RGB	Red, Green and Blue Color Space
SVM	Support Vector Machine
WHO	World Health Organization
CNN	Convolutional Neural Network CNN
BOF	Bag of Features
HOG	Histogram of Oriented Gradient
DBN	Deep Belief Networks

Chapter 1 Introduction

1.1 Motivation

Obesity in adults and children is considered a global epidemic [1]. Overweight and obesity are defined as abnormal or excessive fat accumulation that may impair health. Body mass index (BMI) is a simple index of weight-for-height that is commonly used to classify overweight and obesity in adults. For adults, WHO defines overweight and obesity as follows:

- Overweight is a BMI greater than or equal to 25.
- Obesity is a BMI greater than or equal to 30.

The main cause of obesity is a combination of excessive food consumption and a lack of physical exercise [2]. As people are becoming used to a sedentary lifestyle, they are involuntarily unaware of their food energy intake. There is overwhelming evidence that metabolic complications, which are caused by obesity, increase the risks for developing adverse health consequences, such as diabetes, high blood pressure, dyslipidemia, and hypertension [1]. People in general understand the links between diet and health. In fact, there is widespread nutritional information and guidelines available to patients at their fingertips. However, such information has not prevented diet-related illnesses or helped patients to eat healthily. In most cases, people find it difficult to examine all the information about nutrition and dietary choices. Furthermore, people are oblivious about measuring or controlling their daily calorie intake due to the lack of nutritional knowledge, regular eating patterns, or self-control. Empowering patients with an effective long-term solution requires novel mechanisms that help them make permanent changes to their dietary quality and calorie

intake.

Furthermore, the main cause of obesity is the imbalance between the amount of food and energy consumed by the individuals [3]. Therefore, to lose weight in a healthy way, as well as to maintain a healthy weight for normal people, daily food intake must be measured. In fact, all existing obesity treatment techniques require the patient to record all food consumed per day to compare food intake to consumed energy. However, in most cases, unfortunately, patients face difficulties in estimating and measuring food intake due to the self-denial of the problem, lack of nutritional information, the manual process of writing down this information (which is tiresome and can be forgotten), and other reasons. Hence, a semi-automatic monitoring system to record and measure the amount of calories consumed in a meal would be of great help to not only patients and dietitians in the treatment of obesity, but also the average calorie-conscious person. Indeed, a number of food intake measuring methods have been developed in the last few years. However, most of these systems have drawbacks, such as usage difficulties or large calculation errors.

1.2 Research Problem

The widespread availability of “smart” mobile telephones (“smart phones”), along with their improved memory capacity, network connectivity, and faster processors, enables these devices to be used in health care scenarios. A dietary assessment application for a mobile phone offers a unique mechanism for collecting dietary information. Measuring accurate dietary intake is considered an open research problem in the nutrition and health fields.

In this thesis, we will investigate the following problems:

1. Smart phones have limitations in handling time-sensitive and computationally demanding algorithms.

2. Different segmentation and algorithms used in food recognition and calorie measurement applications consume device batteries quickly.
3. It is difficult for smart phones to manage the large amount of data and images required for food recognition.
4. Detection and recognition of multiple food items is challenging for smart phones.

1.3 Proposed Solution

To resolve the drawbacks mentioned above, we propose a system in which we can automatically estimate the amount of food consumed at a meal based on images taken by a mobile device user. Each food item's image is segmented and identified, and its number of calories is estimated. "Before" meal and "after" meal images can be used to estimate food intake. Using this information, the nutrients consumed can be determined using a food composition database. Our system uses more than 7000 images for food classification, image segmentation identification, and calorie measurement. Images are taken under different conditions, such as with different cameras, lighting, and angles. We also use a variety of foods, such as solid and liquid foods, and mixed and non-mixed foods. The proposed system first uses the SVM method for object classification. The SVM method is known for its robust and accurate classifications.

The processes of the segmentation and classification of food images are known to be complex and computationally intensive. Therefore, a high-power processor is needed to run some of the sophisticated segmentation methods, and to ensure that other SVM kernels achieve accurate results. Furthermore, our system is intended to be available to users anywhere they find themselves enjoying a meal, making an online-based system for food recognition crucial for users' convenience. We use mobile cloud computing (MCC) for our proposed system, as it is targeted

mostly to mobile devices. MCC uses a combination of cloud computing and mobile networks to provide benefits to mobile users. Using MCC not only solves the computational complexity constraint of our system, but it also delivers more accurate results. In other words, since the cloud has access to all other client data, it can update its food database more easily, leading to results that are more accurate.

The details of our proposed cloud-based food recognition system and its implementation are presented in Chapter 4. The results of the experiments show that the proposed system surpasses existing studies in several aspects, such as food image segmentation, classification, identification, and calorie measurement.

In the next step, to develop a more accurate system for recognizing food portions, we focus on the implementation of deep learning as a means for providing food classification and recognition, which are imperative for calorie measurement systems. We demonstrate that this method provides a powerful instrument that significantly increases the accuracy of food classification and recognition in our system. We further increase the accuracy rate of food recognition by applying deep learning neural network algorithms. Deep learning is an emerging approach within the machine learning research community. Deep learning algorithms have been proposed in recent years to move machine learning systems toward the discovery of multiple levels of representation.

Also, we aim to solve the task of multiple food items recognition and detection using CNN features and weakly supervised learning.

1.4 Contributions

In our proposed system, we aim to use smart phones as monitoring tools, as they are widely accessible and easy to use. Our system makes the following contributions:

- For recognizing multiple food items, we perform region mining with submodular cover optimization to find the positive instances, which are represented by rectangular windows containing the target food category.
- We propose a deep neural network model for our application. Our proposed CNN serves as a backbone to the application, and handles the training and testing requests at the top layers without affecting the central layers. We customize the top layer of the deep neural network, which allows us to embed the functionality of the application easily and retain the top levels of the network to spot the relevant images, even in low-powered mobile devices.
- We propose a cloud-based SVM system for food categorization and calorie measurements.
- We have designed a mechanism by which we periodically update the MapReduce SVM model. In doing so, we ensure the system is periodically trained so that it can correct any inaccuracies that may have occurred during the classification phase.
- We have designed a method to apply a graph-cut segmentation for our method to counter food regions more precisely.
- We have designed a method for applying a Gabor filter to the texture segmentation of food images. To do this, a bank of Gabor filters with different desired orientations and wavelengths are applied to an image. Texture plays an important role in identifying different food portions.

- In our proposed system, we have designed a method that uses different methods of segmentation, which are mentioned in the thesis, to extract color, texture, size, and shape features.
- In our food image segmentation, classification, identification, and calorie measurement system, we use a variety of foods, such as solid or liquid foods, and mixed or non-mixed foods. Since there is not any complete food dataset already in existence, we created a dataset of food images that contains more than 7000 images. These images are captured under different conditions, such as with different cameras, lighting, and angles. This dataset is free to be used by public.

1.5 Publications Resulting from this Work

Journal paper:

[1] Parisa Pouladzadeh, Shervin Shirmohammadi, "Real-time Mobile Multi-item Food Recognition Using Deep Learning", Submitted to ACM Transactions on Multimedia Computing Communications and Applications , minor revision, revised version submitted, 2017.

[2] Sri Vijay Bharat Peddi, Pallavi Kuhad, Abdulsalam Yassine, Parisa Pouladzadeh , Shervin Shirmohammadi, Ali Asghar Nazari Shireh, "An Intelligent Cloud-Based Data Processing Broker for Mobile e-Health Multimedia Applications" Future Generation Computer Systems, Elsevier, p.p .71–86, 2016.

[3] Parisa Pouladzadeh, Sri Vijay Bharat Peddi, Pallavi Kuhad, Abdulsalam Yassine, Shervin Shirmohammadi, "A virtualization mechanism for real-time multimedia-assisted mobile food recognition application in cloud computing", Cluster Compute, p.p. 1099-1110, 2015.

[4] Parisa Pouladzadeh, Shervin Shirmohammadi, Abdulsalam Yassine. 2016. You are what you eat: So measure what you eat. IEEE Instrum. Meas. Mag. 19(1):9-15, 2015.

[5] Parisa Pouladzadeh, Shervin Shirmohammadi, and Rana Almaghrabi, “Measuring Calorie and Nutrition from Food Image”, IEEE Transactions on Instrumentation & Measurement, Vol.63, No.8, p.p. 1947 – 1956, August 2014.

[6] P. Pouladzadeh, S. Shirmohammadi, A. Bakirov, and A. Bulut, “Cloud-Based SVM for Food Categorization”, Multimedia Tools and Applications, Springer, Published online, June 2014.

E-Letter:

[1] P. Pouladzadeh, A. Bakirov, S. Shirmohammadi, and A. Bulut, “Utilizing the Cloud for Image-Based Food Recognition”, IEEE COMSOC Multimedia Communications Technical Committee E-Letter, Vol. 8, No. 6, pp. 16-18, 2013.

Conference papers:

[1] Parisa Pouladzadeh, Pallavi Kuhad, Sri Vijay Bharat Peddi, Abdulsalam Yassine, Shervin Shirmohammadi. "Calorie Measurement and Food Classification Using Deep Learning Neural Network", IEEE International Conference on Instrumentation and Measurement Technology (I2MTC), 2016.

[2] P. Pouladzadeh, A. Yassine, and S. Shirmohammadi. "FooDD: Food Detection Dataset for Calorie Measurement Using Food Images", in New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Lecture Notes in Computer Science, Springer, Volume 9281, ISBN: 978-3-319-23221-8, 441-448, 2015.

[3] P.Pouladzadeh, S.Shirmohammadi, A.Yassine, “Using Graph Cut Segmentation for Food Calorie Measurement”, IEEE International Symposium on Medical Measurements and applications, p.p.1-6, Lisbon, June 2014.

[4] Parisa Pouladzadeh, Pallavi Kuhad, Sri Vijay Bharat Peddi, Abdulsalam Yassine, Shervin Shirmohammadi “Mobile Cloud Based Food Calorie Measurement” The 4th International IEEE Workshop on Multimedia Services and Technologies for E-Health (MUST-EH),p.p. 1-6- ICME, Chengdu,July 2014.

[5] Parisa Pouladzadeh, Pallavi Kuhad, Sri Vijay Bharat Peddi, Abdulsalam Yassine, Shervin Shirmohammadi, “A Map Reduce Parallel Classifier for Cloud Based Food Recognition”, International Conference on Next Generation Computing and Communication Technologies [ICNGCCT], Dubai, 23-24 April 2014.

[6] Parisa Pouladzadeh, Shervin Shirmohammadi, Tarik Arici, “Intelligent SVM based food intake measurement system”, IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), p.p.87-92, 15-17 July 2013.

1.6 Organization

The organization of the thesis is as follows. In chapter 2, some of the related works are explained. Technology Background of the system is explained in chapter 3. In chapter 4, the proposed system for food recognition is depicted. Chapter 5 represents the implementation and evaluation steps. Finally the conclusion and future work is described in chapter 6 and 7 respectively.

Chapter 2 Related Work

In this section, a review of some of the most popular dietary assessment methods is provided. The objective here is to describe the advantages and major drawbacks of those methods. This will demonstrate the significance of our mobile food recognition system, which can be used for population and clinical based studies to improve the understanding of diet. We have divided them to four different methods which are Traditional Clinical Methods, Smart Environment, Smartphone Based System, and Dedicated Portable System. Large portions of this chapter are almost verbatim based on the magazine paper [4].

2.1 Traditional Clinical Methods

In 24-hour dietary recall method, the respondent is asked to remember and report all food consumed in the previous 24 hours. The recall is normally prepared through an in person or telephone interview. The interview usually needs specific probes to help the respondent remember all foods consumed in the day. In this method, the interviewer investigates daily reports to help the patient getting a better program for the other days [5]. While helpful this method has a major drawback related to underreporting. In [6], Wang et al. has been shown that features such as obesity, gender, and education, seeming health status, age and ethnicity are underreported. In [7], Harnack et al. also found that important information about food portions are underreported. Underreporting of food intake is discussed in other studies such as [8] . It has been observed that portion sizes have grown considerably in the past 20 to 30 years [7][8], this may be a contributor to underreporting. Obviously, there is a need for methods to capture accurate portion sizes as well as to collect accurate dietary information.

2.2 Smart Environments

The idea of the smart kitchen has also appeared in Pei-Yu Chi et al. works [9]. In this approach, researchers designed a smart kitchen called the Calorie-Aware Kitchen, including cameras to increase the awareness of choosing healthy food and the amount of calories in prepared food. The kitchen includes a camera overhead to capture images of the food preparation process, while there are sensors connected to the counter and stove to measure all the ingredients and cover most of the places inside the kitchen. This will result in immediate feedback for the user with a suggestion for the suitable amount of calories intake. The major downsides to this approach are the limited usage, the smart kitchen being linked to a certain place and the inability for the smart kitchen to be used outside the home. Moreover, some configuration mistakes could happen, such as mistaking the types of cooking oil or meat due to the similarity between food categories.

Nishimura and Kuroda advanced a wearable sensor system by using a microphone [10]. On the other hand, because of applying intelligent technologies in the dietary intake assessments, Chang et al. in [11] created a dietary aware dining table for diary intake consumption. They used radio frequency identification (RFID) as a surface sensor to gauge the type of food eaten and integrated existing scales on a dining table to measure food weight. But there are many drawbacks related to this technique, including the difficulty in using it in several locations and the complexity of attaching the RFID tag to each served food.

2.3 Smartphone Based System

Recently, food intake measuring systems that rely on image capture and analysis became commonly used all over the world due to the great advances in cell-phone camera resolutions, computer programs, network connectivity and image processing analysis. Food recognition is a hard case in object recognition. The difficulty of this problem is mainly due to

the fact that food portions come in different sizes and shapes. Also, food portions can be single items, multiple items, or mixed food such as salad, soup, etc. Consequently, in some food an image taken from a whole plate of mixed food is impossible for traditional pattern recognition techniques to accurately discriminate among them with a high degree of success. Another issue is related to processing time since most recognition and segmentation algorithms are computationally intensive and require faster processors and higher memory. Generally, food portions recognition is divided into two main groups: Segmentation and Recognition.

The ideal output of the image segmentation operation is to group those pixels in the image which share certain visual characteristics that are perceptually meaningful to human observers. This is a difficult problem as humans use all sorts of tricks to perform this task. Various methods of segmentation were used in different food applications, such as: Color and Texture Segmentation, K-mean Clustering, Graph Cut Based Segmentation.

Also, in this stage, the extracted features are classified in order to recognize each food portion by applying different classification methods such as SVM, Neural Network, and Deep Neural Network. Also by using Cloud space and applying all these methods in the virtual space, the accuracy and time processing will have a huge increase in all food applications. Figure 2-1, shows the different methods of segmentation and recognition which are applied on food calorie measurement applications. In the following sections we will see applications with using each of these methods. Some related applications and methods will be described below.

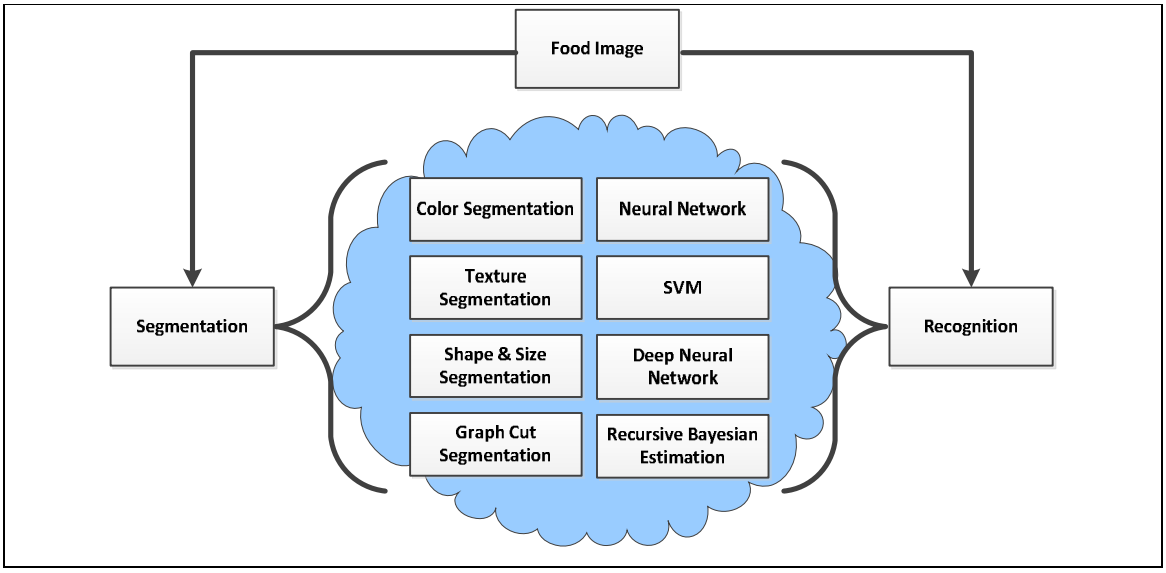


Figure 2-1 Variety of different Architecture for different calorie measurement applications

2.3.1 Meal Snap – Calorie Counting Magic

Meal Snap [12], lets you take pictures of the meals you eat, and then magically tells you what food was in your meal. But, it seems like a person who stands behind the app and send the results. Because it takes long time and every time the result is the same as the previous one. In this application, Food detection method is manual and the disadvantage is latency varies. This means that some time it takes too long to receive the result on our cell phone (some times around 24 hours). Figure 2-2 shows the application.

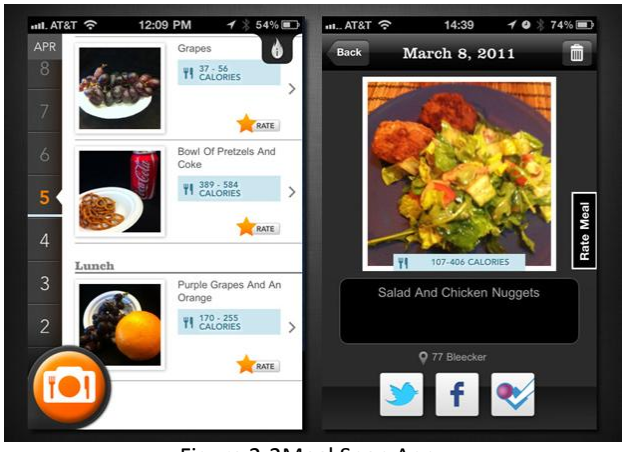


Figure 2-2Meal Snap App.

2.3.2 Carbs & Cals

The Carbs and Cals app [13], doesn't have image processing part. The latency is fast but the problem is, it doesn't contain a vast range of food and only contains over 2,500 photos of popular food and drink items.

The application provides a very easy comparison of up to 6 portions of each food. Simply tap the one closest to the portion on your plate to reveal the larger photo, which is shown in Figure 2-3. For foods of which you are likely to eat several (e.g. biscuits, slices of pizza, etc.), there is a multiple wheel for entering the amount of the number of items you are consuming. Foods are added to a meal, and the total carbs and calories are automatically calculated and displayed at the bottom of the meal. The main weaknesses in this device is that users do not have the opportunity to measure different types of food rather than the types stored in the database. Moreover, the device does not count the amount of food intake which might be different from the amount of food already stored in the data.

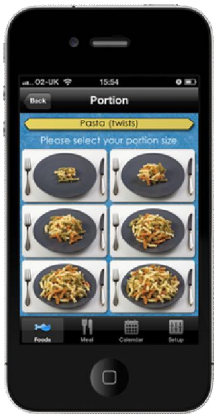


Figure 2-3 Carbs and Cabs App.

2.3.3 Mobile Weight Loss Tracker & Carb Counter

Barcode scanners in smartphones are also used for the purpose of calorie measurement [14] . This Mobile application is one such tool which gets nutritional information on grocery

items, restaurant meals, and over 1,600 Atkins recipes and products. Search by keyword or scan the UPC label. The method covers only limited food database. The Menu is shown in Figure 2-4.

Figure 2-4 Weight Loss Tracker App.

2.3.4 Food Scanner [15]

Similarly, Food Scanner [15] allows you to use your iPhone's camera to scan UPC barcodes on the foods you eat, although it uses Daily Burn Tracker's nutrition database of about 200,000 food products, and powered by Red Laser technology (for scanning barcode). It suffers from the same problem as Mobile Weight Loss Tracker & Carb Counter; i.e., it cannot measure anything that doesn't have a barcode. Nevertheless, for situation where the food has barcode and the barcode is available in the mobile system's database, it can be useful. It is shown in Figure 2-5.

Figure 2-5 Food Scanner App.

2.3.5 Other applications and methods

In [16], Jie Yand et al. created a computer program method to identify fast-food intake from video of eating by using a wearable camera. In this method, numbers of captured images

from a fast-food restaurant are compared to images stored in a computer database. Researchers applied the camera in three different locations with 101 food types. The idea is useful but it covers only some images from fast food and not more.

Related to this model, Jieun Kin in [17], proposed a method for automatically estimating the amount of a given nutrient contained in a commercial food. The method applies when no part of any ingredient is removed in the preparation process. First, the system automatically bound the amount of each ingredient used to prepare the food using the information provided on its label along with the nutrition information for at least some of the ingredients. Then utilize the Simplex algorithm to refine these bounds on the nutrient content.

By applying different methods of image processing and segmentation algorithms, food recognition applications were able to increase their accuracy and time processing. In [18], Young et al. used image processing techniques to measure and analyze large food portions. A similar approach is also reported in [19], where the idea is to take pictures of the food, and based on a calibration card located inside the picture as a measurement pattern, the size of the food portions is calculated. In this study, the food is manually identified with the help of nutritional information retrieved from a database. Then, the calories are calculated for each picture and finally the complete set of information is stored in different database in the research facility. In this case, based on the known size of the calibration card, the portions can be translated into real life size, and the calculations are closely related with the real caloric content of each food. Martin et al. [20], proposed a system where the user captures the images, with the calibration card also, then the images are sent to a research center to be analyzed. This will, of course, come with its own shortcoming of offline data processing.

In addition to the above mentioned studies, researchers apply classification and segmentation methods on food images to achieve higher recognition accuracy and lower processing time. For example, Takeda, F., et al. in [21] used the Neural Network (NN) to develop

a method to extract food intake from an image; in this approach, they captured a photo of several dishes in a tray before and after eating. Particularly, an image of the whole tray is captured first. Then this image will be converted to a binary image by using threshold values, and a small image of the food will be extracted from the tray image. Due to the previous procedures, the system will identify all information related to the image such as length, width and shape. All previous information will transfer to the NN. After getting the results, researchers applied them to a computer simulation program to compare the information and analyse the results. This method is also difficult for the user to follow. The user must capture the photo in a tray to extract the shape of the food. Moreover, the food image needs to be analysed by the computer, which is impractical for everyday usage.

In [22] Fengqing Zhu et.al. proposed a method for dietary assessment to automatically identify and locate food in a variety of images. Two concepts were combined in their algorithm. First, a set of segmented objects partitioned into similar object classes based on their features; for solving the idea they have applied different segmentation method, second, automatic segmented regions were classified using a multichannel feature classification system. They have used SVM as their classifier. The final decision is obtained by combining class decisions from individual feature, but the processing will run on the mobile device which is not affordable for users.

. Also Jay Baxter In [23], mentioned that Food recognition is a difficult problem, because unlike objects like cars, faces, or pedestrians, food is deformable and exhibits high intra-class variation. This paper considers the approach of analyzing a food item at the pixel level by classifying each pixel as a certain ingredient, and then using statistics and spatial relationships between those pixel ingredient labels as features in an SVM classifier. Results shows that using pixel ingredient labels to identify food greatly increases classification accuracy, but at the expense of higher computational cost.

In [24], Kawano et al. propose a mobile food recognition system. The purposes of which are estimating calorie and nutritious of foods and recording a user's eating habits. Since all the processes on image recognition performed on a smartphone, the system does not need to send images to a server and runs on an ordinary smartphone in a real-time way. To recognize food items, a user draws bounding boxes by touching the screen first, and then the system starts food item recognition within the indicated bounding boxes. To recognize them more accurately, they segment each food item region by Gruph Cut, color histogram and SURFbased bag-of-features, and finally classify it into one of the fifty food categories with linear SVM and fast $\chi 2$ kernel. In addition, the system estimates the direction of food regions where the higher SVM output score is expected to be obtained, show it as an arrow on the screen in order to ask a user to move a smartphone camera. This recognition process is performed repeatedly about once a second. In the experiments, they have achieved the 81.55% classification rate for the top 5 category candidates when the ground-truth bounding boxes are given. But in their work they didn't show their datasets and didn't mention that this accuracy percentage is only for single food or both single and multiple food items and also the application is run on the mobile devices.

Yu Wang in [25] have developed a dietary assessment system that uses food images captured by a mobile device. Food identification is a crucial component of the system. Achieving a high classification rates is challenging due to the large number of food categories and variability in food appearance. They employ recursive Bayesian estimation to incrementally learn from a person's eating history. Results show an improvement of food classification accuracy by 11%.

2.3.5.1 Cloud Examples

The above mentioned methods are computationally demanding and require processing resources beyond what a smart phone's memory can handle. For this reason, several studies introduced cloud resources as a means for offloading the process of image segmentation and classification. The cloud resources not only allow for achieving higher accuracy, but also can lower the processing time while processing huge image datasets.

Some existing systems which used cloud computing models and machine learning are mentioned here. Low and et al. [26], extended Graph Lab framework to support dynamic and parallel computation of graphs on cloud. The study implements extensions to pipelined locking and data versioning to avoid network congestion and latency. Graph Lab framework successfully deployed on large Amazon EC2 cluster to run performance tests. In [27] , Grzegorz et al., were implemented an experimental Page Ranking system, called Pregel. The results show that Distributed Graph Lab performs better than Hadoop 20 to 60 times. Kraska et al. [28], introduced a new distributed machine learning system called MLBase. MLBase allows researchers to declare machine learning problems in very simple way and implements this algorithm in distributed and highly scalable manner without extensive systems knowledge. The optimizer in the framework converts ML jobs into an artificial learning plan and returns best answer to the user by improving result iteratively in the background. Alessandro Mazzei in [29],have used Cloud Computing and proposes a software architecture for automatic diet management and recipes analysis.

2.3.5.2 Deep Learning Examples

Convolutional Neural Network (CNN) or (Deep Learning) is also used to identify different food portions more accurately and easily. In [30], they apply a (CNN) to the tasks of detecting and recognizing food images. Because of the wide diversity of types of food, image recognition of

food items is generally very difficult. However, as deep learning has been shown recently to be a very powerful image recognition technique, and CNN is a state-of-the-art approach to deep learning. They applied CNN to the tasks of food detection and recognition through parameter optimization. They constructed a dataset of the most frequent food items in a publicly available food-logging system, and used it to evaluate recognition performance. CNN showed significantly higher accuracy than did traditional SVM based methods with handcrafted features. In [34], Christodoulidis et al. propose a method for the recognition of already segmented food items in meal images. The method uses a 6-layer deep convolutional neural network to classify food image patches. For each food item, overlapping patches are extracted and classified and the class with the majority of votes is assigned to it. Experiments on a manually annotated dataset with 573 food items justified the choice of the involved components and proved the effectiveness of the proposed system yielding an overall accuracy of 84.9%. Hokuto Kagaya in [31], investigate food classification of images. They used three different datasets of images: (1) images they collected from Instagram, (2) Food-101 and Caltech-256 dataset (3) dataset they used in [4]. They investigated the combinations of training and testing using the all three of them. As a result, they achieved high accuracy 96%, 95% and 99% in the three datasets respectively. The problem of this work is, it only can find the food item and doesn't calculate the calorie of the food. It means that it cannot calculate the size of the food portion which is needed for the calorie estimation.

In [32], system, also use food images taken with the built-in camera of a smartphone. But, authors go one step further by implementing graph cut segmentation and deep learning algorithms as a means of improving the accuracy of food classification and recognition system. Furthermore, the processing of the images and the calorie measurement results are provided to the user instantly. This means that the system is convenience to use and well suited to be a long-term solution for users. Furthermore, system uses cloud-based virtualization where an

emulation of the smartphone environment allows the application to run virtually on the remote server in the cloud. They have used dataset which is published in [33]. The comparison table of some food recognition system which is published in [4] has written below.

Paper	Architecture	Acquisition	Segmentation Techniques	Classification Techniques	Accuracy	Analysis Summary
[19]	Mobile	1- Picture, using calibration card located inside the picture	-	-	-	They need card everywhere, and the picture sent to a research center to be analyzed. This will, of course, come with its own shortcoming of offline data processing
[23]	Mobile	1 Picture	Bounding Box	Linear SVM	81.55%	Using only linear SVM segmentation.
[16]	Mobile	Video	SIFT package	-	73%	Focuses on recognizing foods not deciding food portions and only used database of 101 foods from 9food restaurants in USA
[24]	Mobile	1-Picture	Bounding Box	Linear SVM + Fisher Vector	79.2%	Using linear SVM and fisher vector together, but still the accuracy of the system needs to be increased.
[21]	Mobile	1-Picture, Dish Image in the tray	Information such as width, length, diameter and shape	NN	70%	Food sometimes causes miss-recognition of the correct position of the extracted dish image, It is good that they newly add food rejection algorithm to the dish extraction method
[22]	Mobile	Picture	Normalized cut, Salient Region Detection, Fast Rejection	KNN and SVM	Not Compared	Good segmentation and classification method, but Not using servers so the processing time will be too long
[25]	Mobile	Picture	Graph based segmentation	KNN and SVM	Improved 11% in average	They employ recursive Bayesian estimation to incrementally learn from a person's eating history
[22]	Mobile-Server	1-Picture	color, texture and SIFT	SVM	61.34%	50 kinds of foods, but number of picture is not enough for good training and they achieve a low accuracy.
[35]	Mobile-Server	1-Picture	Bag-of-features (BoF), Color histogram, Gabor features and gradient histogram	MKL	62.5%	Complete and accurate segmentation part and they implemented a prototype system to recognize food images taken by cellular-phone cameras, but the accuracy of the system is not good enough.
[71][72][82] [83]	Mobile-Server	2-Pictures, User Thumb	K-mean clustering, Color-Texture Segmentation	SVM+ RBF Kernel	92.21%	Complete and accurate segmentation steps and RBF SVM method and achieved reasonable accuracy which is good for the user.

2.3.5.3 Multiple Food Items

Unfortunately, few research projects have been conducted on multiple-food recognition. In [35] , Matsuda et al presented the first work ever done on multiple-food recognition. First, candidate regions generated by the whole image, deformable part model, circle detector and JSEG region segmentation were integrated to create a candidate set of bounding-box regions. Second, various kinds of image features—including a bag of features (BoF), a histogram of oriented gradient (HOG), Gabor texture features and color histograms—were computed for each bounding box region in the candidate set. The evaluation values of each region belonging to all the given classes were computed using trained SVM classifiers by multiple kernel learning. They trained the SVM models on their own dataset, which contained 9,132 food images of 100 different food categories; an accuracy rate of 65.6% was reported on their test set. In [36], Kawano and Yanai, developed a mobile application called FoodCam for real-time multiple food recognition. In this system, users needed to draw bounding circle over food items on the screen. This bounding circle was adjusted to the food region by a GrubCut-based segmentation method. For each segmented region within the bounding circle, Fisher Vector encoded feature vectors of color histograms, and histograms of gradients were computed and passed into SVM classifiers to predict the food item. An accuracy rate of 51.9% was reported on their test set.

Zhang et al. [37] developed a mobile application for multiple-food recognition of 15 food categories on the phone without any user intervention. First, the user took a photo of a food plate, and a cropped 400x400 food image was uploaded to the server for food recognition. On the server side, the food image was first segmented into possible salient regions, and these regions were further grouped based on the similarity of their color and their HOG and SIFT feature vectors. Normally, a typical food image yielded about 100 salient segment regions. They collected 2,000 training images for 15 classes with a mobile phone camera, since they had

discovered that the model trained with images downloaded from the Internet could not generalize well for images taken on mobile phones. They trained a linear multiple-class SVM classifier for each class using the Fisher vector encoded feature vectors (including SIFT and color features) of salient regions. They reported a top-1 accuracy rate of 85% when detecting 15 different kinds of foods in their experiments.

More recently, zhang et al [38] proposed a multi-task system that can identify dish types, food ingredients, and cooking methods from food images with deep convolutional neural networks. They built up a dataset of 360 classes of different foods. To reduce the noises of the data, which was collected from the Internet, outlier images were detected and eliminated through a one-class SVM trained with deep convolutional features. They simultaneously trained a dish identifier, a cooking method recognizer, and a multi-label ingredient detector. They share a few low-level layers in the deep network architecture. The proposed framework shows higher accuracy than traditional method with handcrafted features.

In [39], Akbari Fard et al. introduce an automatic way for detecting and recognizing the fruits in an image in order to enable keeping track of daily intake automatically using images taken by the user. The proposed method uses state of the art deep-learning techniques for feature extraction and classification. Deep learning methods, especially convolutional neural networks, have been widely used for a variety of classification problems and have achieved promising results. The model has achieved an accuracy of 75% in the task of classification of 43 different types of fruit.

In [40], Singla et.al. report experiments on food/non-food classification and food recognition using a GoogLeNet model based on deep convolutional neural network. The experiments were conducted on two image datasets, where the images were collected from existing image datasets, social media, and imaging devices such as smart phone and wearable cameras. Experimental results show 83.6% accuracy for the food category recognition. They

mention the main reason for not achieving a higher recognition accuracy on certain types of food images is the complex mixture of food items in the image and the high visual similarities between some images across categories.

Shimoda and Yanai, [41] propose an intermediate approach between the traditional proposal approach and the fully convolutional approach. Especially they propose a method which generates high food-ness regions by fully convolutional networks and back-propagation based approach with training food images gathered from the Web. In their work they achieved reduced computational cost while keeping high quality for food detection.

Finally, in [42], Dehais et al. propose a method to detect and segment the food dishes in an image. The method combines region growing and merging techniques with deep CNN-based food border detection. A semi-automatic version of the method is also presented that improves the result with minimal user input. The proposed methods are trained and tested on non-overlapping subsets of a food image database including 821 images, taken under challenging conditions and annotated manually. The automatic and semi-automatic dish segmentation methods reached average accuracies of 88% and 92%, respectively.

Another method in which authors have tried to increase the accuracy of the results is described in [43].Ruihan Xu et al explore leveraging geolocation and external information about restaurants to simplify the classification problem. They have collected a restaurant-oriented food dataset with food images, dish tags, and restaurant-level information, such as the menu and geolocation. Experiments on this dataset show that exploiting geolocation improves around 30% the recognition performance, and geolocalized models contribute with an additional 3–8% absolute gain, while they can be trained up to five times faster.

2.4 Dedicated Portable System

Other food calorie measurement type which we include them in dedicated portable systems. In [46] they have a system which automatically read calories consumed by using bioimpedance to measure reading the glucose in the user's cells, although scientists seriously doubt the validity of its approach of using bioimpedance only, and the system has not been properly evaluated on a wide scale. Another clear disadvantage of this system is that it only measures the calories after the user has already eaten the food, which is too late for obesity patients who need to know the amount of calories before they eat the food.

Near infrared (NIR) spectroscopy has also been recently proposed to determine food's composition, with [47] and Consumer [48] ,as commercial examples already available in the market. These systems can tell to the user, for example, the amount of saturated fats per 100 grams in the food, which, when coupled with a good quality lookup database of food ingredient spectra, can give information in terms of calories and nutrition. However, these tools cannot measure the weight and amount of each food ingredient. As such, going from the "fats per 100 grams" value to actual calories in the food is not trivial for the users. In addition, transparent liquids cannot be measured with NIR spectroscopy unless special containers are used.

Chapter 3 Background

In this chapter, we will briefly cover technologies behind our system, Different methods of Image segmentation and Classification. Furthermore, we briefly talk about Cloud, CloudSVM, Hadoop-MapReduce, which we have applied them in this thesis. Then, in chapter4, we will show how we used these technologies in our proposed system.

3.1 Image Segmentation

The ideal output of the image segmentation operation is to group those pixels in the image which share certain visual characteristics that are perceptually meaningful to human observers. This is a difficult problem as humans use all sorts of tricks to perform this task. In this subsection we provide some of the most important segmentation tools that are used in our thesis.

3.1.1 Color and Texture Segmentation

Color and texture are fundamental characteristics of natural images, and play an important role in visual perception. Although color has been used in identifying objects for many years, texture is another active topic in segmentation. In the recent research into food recognition areas, different features of color and texture are combined together in order to measure food nutrition more accurately [49].

Defining what a texture is or how to differentiate between two textures is a complex task. A possible reason for this difficulty is that humans are not consciously aware of what their eyes or

minds do when they identify a texture; the visual system seems to do it automatically. This automatic feature of the visual system is called pre-attentive vision, and this is where texture segmentation occurs [50].

In the early days of the perception research field, it was thought that the human visual system is capable of computing the Fourier Transform of an image and identifying textures based on their power range. However, this view has been mostly ignored, and the texton-based theory of texture perception is more widely accepted.

The objective of texture segmentation is to identify different uniform textural regions in an image. A texture segmentation example is shown in Figure 3-1. The image on the left is the original image consisting of three uniform textural regions. The image on the right presents an "ideal" segmentation result. In other words, the segmentation process produces an image of the same size as the original textural image, in which different colors represent different uniform textural regions in the original image.

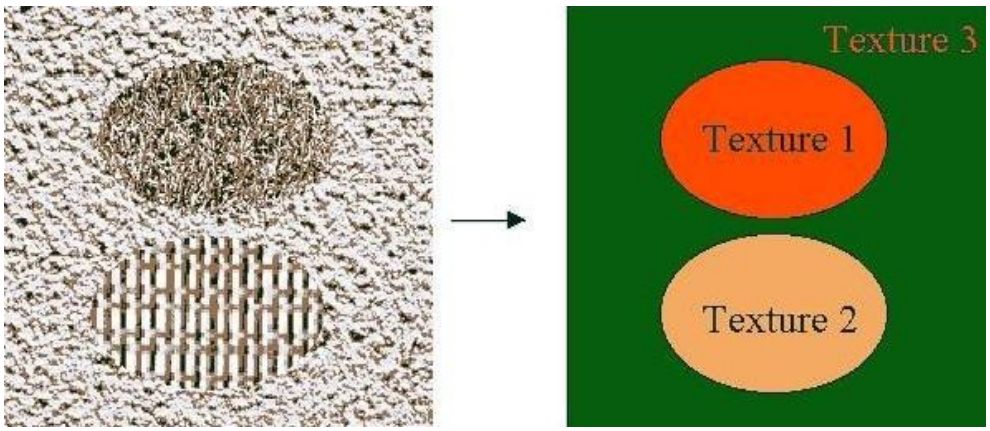


Figure 3-1 Texture segmentation. Left: original image, right: ideal segmentation result

There are several approaches for texture segmentation. Commonly, the segmentation process consists of two major steps, namely feature vector extraction and feature vector clustering. In feature vector extraction, several textural features are extracted from localized regions in the image. All features extracted from the same localized region comprise a single feature vector. In

feature vector clustering, the feature vectors extracted from all localized regions are grouped into several groups. Ideally, all vectors belonging to the same group should correspond to visually similar local texture regions. Later in this section we will introduce the Gabor filter which has been used by numerous researchers in texture analysis problems [51][52].

3.1.1.1 Gabor Filter

The Gabor function, which is mainly used for texture detection, can be defined as shown in formula (3-1) [53].

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left\{-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}\right\}\cos(2\pi f_0+\varphi) \quad (3\text{-}1)$$

The Gabor filter has some directional possessions, as an example when $\sigma_x > \sigma_y$ the filter will look to be extended to the horizontal axis. The filter works as a band-pass filter by imagining the horizontal axis with a central frequency of f_0. Similarly, by imagining the vertical axis, it works as a low-pass filter.

If the Gabor filter has an orientation of θ, the following transformations can be done:

$$x' = x\cos\theta + y\sin\theta \quad (3\text{-}2)$$

$$y' = -x\sin\theta + y\cos\theta \quad (3\text{-}3)$$

Some Gabor filter examples for different σ_x, σ_y and f_0 parameters are shown in Figure 3-2. The gray-level intensity of the surface plots is related to the value of the filter mask at the particular mask location (gray is 0, low and high intensities correspond to negative and positive mask values, respectively) .

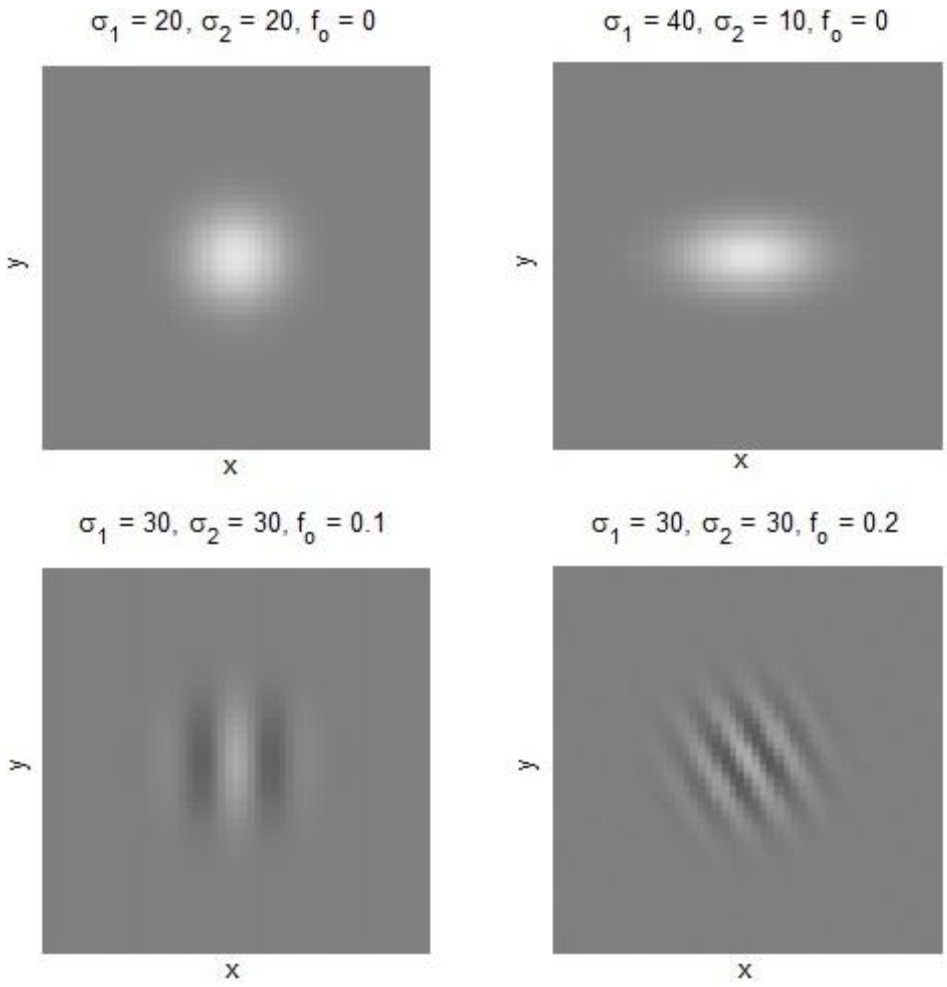


Figure 3-2 Examples of applying Gabor filter with various parameters.

3.1.2 K-mean Clustering

The K-means algorithm is used for clustering data points or vectors into a number of clusters.

In other words, it categorizes the pixels into a number of clusters, where each cluster represents a specific texture. For example, assuming a total of R vectors, $\mathrm{x_r} = 1,2,\ldots \mathrm{R}$, for an image. The goal of the K-means algorithm is to group or cluster the R vectors into K groups. The algorithm of K-means is as follows:

1. Initialize K-cluster centroids (C_k) randomly.

A total of K vectors, $C_k = 1,2,3 \ldots, k$ are chosen randomly, $x_r = x_{rL}, L = 1,2, \ldots L$.

The vectors are called centroids. Each centroid is the typical vector of its own group.

2. Assign each sample to the nearest centroid.

Each vector x_r is assigned to one of the K groups. A usually used distance measure is the Euclidean distance which is shown in (3-4).

$$E^2\{x_r, C_k\} = (x_{r1} - c_{k1})^2 + (x_{r2} - c_{k2})^2 + \cdots + (x_{rL} - c_{kL})^2 \quad (3\text{-}4)$$

3. Calculate centroids (means) of K-clusters.

4. If centroids are unchanged, done. Otherwise, go to step 2.

3.1.3 Graph-Based Segmentation

In this section, we introduce the graph cut method, which is used to establish the graph with the given image for segmentation. Let G=<V, E> be an undirected graph where V is a series of vertices and E is the graph edge which connects every two neighboring vertices. The vertex V is composed of two different kinds of nodes (vertices). The first kind of vertices is neighborhood nodes, which correspond to the pixels and the other kind of vertices are called terminal nodes, which consist of s (source) and t (sink). This type of graph is also called s-t graph where, in the image s node usually represents the object while t node denotes the background. In this graph, there are also two types of edges. The first type of edges is called n-links, which connects the neighboring pixels within the image. And the second type of edges is called t-links, which connects the terminal nodes with the neighborhood nodes. Each edge is assigned with a non-negative weight denoted as w_e, which is also named as cost. A cut is a subset of edges E, which can be denoted as C and expressed as $C \subset E$. The cost of the cut $|C|$ is the sum of the weights on edges C, which is expressed as follows.

$$|C| = \sum_{e \in C} W_e \tag{3-5}$$

A minimum cut is the cut that has the minimum cost called min-cut and it can be achieved by finding the maximum flow, which is verified in [54] to be equivalent to the max-flow. The max-flow/min-cut algorithm developed by [55] can be used to get the minimum cut for the s-t graph. Thus, the graph is divided by this cut and the nodes are separated into two disjoint subsets S and T where s∈S, t ∈T and S ∪T = V. The two subsets correspond to the foreground and background in the image segmentation. An illustration of this graph is depicted in Figure 3-3.

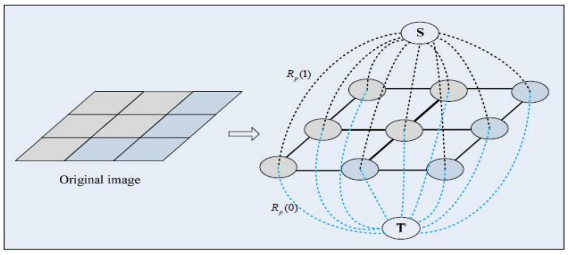


Figure 3-3.Illustration of s-t graph. The image pixels correspond to the neighbor nodes in the graph. The solid lines in the graph n-links and the dotted lines are t-links.

Image segmentation can be regarded as pixel labeling problems. The label of the object (s-node) is set to be 1 while that of the background (t-node) is given to be 0 and this process can be achieved by minimizing the energy-function through minimum graph cut. In order to make the segmentation reasonable, the cut should be occurred at the boundary between the object and the background. Namely, at the object boundary, the energy (cut) should be minimized. Let$\mathrm{L} = \{\mathrm{l}_1, \mathrm{l}_2, \mathrm{l}_3, \ldots \mathrm{l}_\mathrm{i}, \ldots \mathrm{l}_\mathrm{p}\}$, where p is the number of the pixels in the image and $\mathrm{l}_\mathrm{i} \in \{0,1)$, thus, the set L is divided into 2 parts and the pixels labeled with 1 belong to object while others are grouped into background. The energy function is defined in equation (2), which can be minimized by the min-cut in the s-t graph [55].

$$E(L) = \propto R(L) + B(L) \tag{3-6}$$

where, R(L) is called regional term, which incorporates the regional information into the

segmentation and B(L) is called boundary term, which incorporates the boundary constraint into segmentation, $\mathrm{E(L)}$ is the relative importance factor between regional and boundary term. When $\propto$ is set to be 0, it means that the regional information is ignored and only considering the boundary information. In the energy function in equation (3-6), the regional term is defined as follows:

$$R(L) = \sum_{p \in P} R_p(l_p) \tag{3-7}$$

Where, $\mathrm{R_p(l_p)}$is the penalty for assigning the label $\mathrm{l_p}$ to pixel p. The weight of $\mathrm{R_p(l_p)}$can be obtained by comparing the intensity of pixel p with the given histogram (intensity model) of the object and background. The weight of the t-links is defined in the following equations [54][55].

$$R_p(1) = -lnPr(I_p|'obj') \tag{3-8}$$

$$R_p(0) = -lnPr(I_p|'bkg') \tag{3-9}$$

From equation (3-8),(3-9), we can see that when $\mathrm{Pr(I_p|'obj')}$ is larger than $\mathrm{Pr(I_p|'bkg')}$ will be smaller than $\mathrm{R_p(0)}$. This means when the pixel is more likely to be the object, the penalty for grouping that pixel into object should be smaller which reduces the energy in equation (3-6). Thus, when all of the pixels are correctly separated into two subsets, the regional term is minimized. B (L) in equation (3-6) is the boundary term, which is defined in the following equation.

$$B(L) = \sum_{\{p,q\} \in N} B_{<p,q>} . \delta(I_p, I_q) \tag{3-10}$$

Where p, q is neighboring pixels and $\delta(\mathrm{I_p}, \mathrm{I_q})$ is defined as:

$$\delta(I_p, I_q) = \begin{cases} 1 \; if \; I_p = I_q \\ 0 \; if \; I_p \neq I_q \end{cases} \tag{3-11}$$

For the regional constraint, it can be interpreted as assigning labels $\mathrm{I_p}, \mathrm{I_q}$ to neighboring pixels. When the neighboring pixels have the same labels, the penalty is 0 which means that the regional term would only sum the penalty at the segmented boundary. For the term $\mathrm{B_{<p,q>}}$, it is

defined to be a non-increasing function of $\left|\mathrm{I_p} - \mathrm{I_q}\right|$ as follows:

$$B_{<p,q>} \propto exp(-\frac{(I_p - I_q)^2}{2\sigma^2}) \tag{3-12}$$

Where σ can be viewed as camera noise. When the intensity of two neighboring pixel is very similar, the penalty is very high. Otherwise, it is low. Thus, when the energy function obtains a minimum value, it is more likely occurred at the object boundary. The study in [55] shows that the minimized energy value can be computed by the min-cut through max-flow. Consequently, the minimum energy problem is converted into the graph cut problem. In order to get a reasonable segmentation result, the assignment of the weight in the s-t graph is very important. The weight of the s-t graph is given as following.

$$Weight = \begin{cases} B_{<p,q>} & \{p,q\} \in Neiboring\ pixel \\ \propto . R_p(0) & for\ edge\ \{p,s\} \\ \propto . R_p(1) & for\ edge\ \{p,T\} \end{cases} \tag{3-13}$$

Equation (3-13) can also be explained as such, in the s-t graph, when the intensity of the pixel is inclined to be the object, the weight between this pixel and the s-node will be larger than that between the pixel and the t-node, which means that the cut is more likely occurred at the edge with smaller weight. For the neighboring pixels, when their intensity is very similar, the weight is very big, which is not likely to be separated by the cut. Thus, when the minimum cut is achieved from the s-t graph, the location of the cut is close to the object boundary. The implementation of the graph cut can be fulfilled by the max-flow/min-cut as described in [54][55]. In Figure 3-4, we illustrate the graph cut for a 3×3 image segmentation. The thickness of the edge denotes the magnitude of the weight.

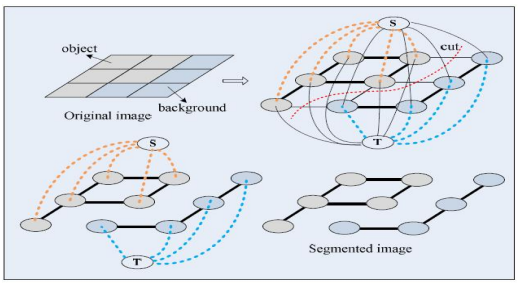


Figure 3-4 Illustration of graph cut for image segmentation

3.2 Deep Learning

A neural network computes a differentiable function of its input. For example, our application computes the probability of the match of the input image with the corresponding label set,

$$\mathrm{p(label \mid an\ input\ image)} \tag{3-14}$$

The standard way to model a neuron's output f as an activation function of its input x is either a hyperbolic function

$$\tanh(\mathrm{x}) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \text{ or} \tag{3-15}$$

$$\mathrm{sigmoid(x)} = 1/(1 + e^{-x})$$

But we use the term rectified linear unit (ReLU) to refer to unit in a neural net that use the activation function $\max(0; \mathrm{x})$. As compared to the above mentioned activation functions (hyperbolic or sigmoid), deep convolutional neural networks with ReLUs trains several times faster [56].

For understanding the deep neural network, we can refer to this example. Consider we choose the grayscale images which are 28 by 28 pixels in size as input. If we will use the notation x to denote the training input, then there would be 28*28= 784 input neurons. Each entry in the neuron represents the grey value for a single pixel in the image. We'll denote the corresponding desired output by$y = y(x)$. What we'd like is an algorithm which lets us find weights and biases so that the

output from the network approximates $y(x)$ for all training inputs x. So if we have a training image set of food images and we want to classify the food type to Apple during the learning phase, we could possibly achieve that by tweaking the weight and bias values. To quantify how well we're achieving this goal we define a cost function[57]:

$$\mathrm{C(w,b)} \equiv 1/2 \sum_x \| \mathrm{y(x)} - \mathrm{a} \|^2 \tag{3-16}$$

Here, w denotes the collection of all weights in the network, and b all the biases, a is the vector of outputs from the network when x is input, and the sum is over all training inputs, x. In other words, we want to find a set of weights and biases which make the cost as small as possible.

We'll do that using an algorithm known as stochastic gradient descent. By using a smooth cost function like the quadratic cost it turns out to be easy to figure out how to make small changes in the weights and biases so as to get an improvement in the cost[56][59]. Hence we will able to tweak the weights and bias to get the output closer to the desired output, during the learning phase. Hence our goal is to train the neural network is to find weights and biases which minimize the quadratic cost function C(w,b).

The idea is to use gradient descent to find the weights $\mathrm{w_k}$ and biases $\mathrm{b_l}$ which minimize the cost C. The gradient vector $\nabla \mathrm{C}$ has corresponding components $\partial \mathrm{C}/ \partial \mathrm{w_k}$ and$\partial \mathrm{C}/\mathrm{b_l}$. The stochastic gradient descent can be used to speed up learning by estimating the gradient $\nabla \mathrm{C}$ by computing $\nabla\ \mathrm{C_x}$ for a small sample of randomly chosen training inputs. By averaging over this small sample it turns out that we can quickly get a good estimate of the true gradient$\nabla\ \mathrm{C}$, and this helps speed up gradient descent, and thus learning.

The stochastic gradient descent works by randomly picking out a small number m of randomly chosen training inputs (for example 10 images from the original set of 100 images). We'll

label those random training inputs$X_1, X_2, \ldots, X_m$. Then stochastic gradient descent works by picking out a randomly chosen mini-batch of training inputs, and training with those, where the weights and bias is computed by:

$$w_k \rightarrow w_k^{'} = w_k - \eta/m \sum_j (\partial C/\partial w_k)$$
$$b_{bl} \rightarrow b_l^{'} = b_l - \eta/m \sum_j (\partial C/\partial b_{bl}) \quad (3\text{-}17)$$

Where the sums are over all the training examples X_j in the current mini-batch and η is the learning rate. Then we pick out another randomly chosen mini-batch and train with those, until we've exhausted the training inputs. The back-propagation algorithm is the fast way of computing the gradient of the cost function.

Algorithm in [58], is a way of implementing stochastic gradient descent and the backward propagation to compute the weight and bias for smaller cost function. Training the deep neural network in this way, will allow us to make the necessary changes to the weight and bias, without majorly effecting the output of the network and giving us the desired results. For example, this algorithm will help us to weak the weights(w) and bias(b) during the learning phase, in a way we can finally determine the output as one of the two (apple or cherry), without effecting the rest of the food classes. Delta changes in either the weights or the bias will change the result from one food class to the other. As shown in the below diagram, considering we have taken the color feature into account, any changes in the weight of w_1 or bias b would make the small changes to the final results, which in this case between apple and cherry (having almost same color features) will alter the eventual result. If the probability of the image (p>0.5 towards Apple), it would be classified as Apple and same is the case with any food type.

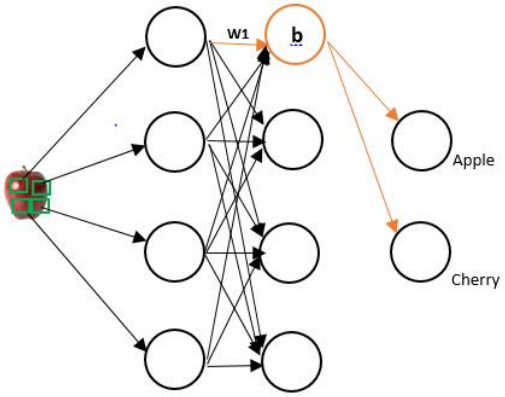


Figure 3-5 An example showing implementation of Stochastic Gradient Descent

Algorithm implementing stochastic gradient descent[58]:

1. Input a set of training examples
2. For each training example x: Set the corresponding input activation a^l, and perform the following steps:
   - **Feedforward**: For each $l = 2,3,\dots,L$ compute $z^l = w^l a^{l-1} + \mathrm{b}^l$ and $a^l = \sigma(\mathrm{z}^l)$.
   - **Output error, $\boldsymbol{\delta^l}$**: Compute the vector $\boldsymbol{\delta^l} = \nabla_{\mathrm{a}}^{C} \odot \sigma'(\mathrm{z}^l)$.
   - **Backpropagate the error**: For each $l = L-1, L-2, \dots, 2$ compute $\boldsymbol{\delta^l} = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
3. Gradient descent: For each l=L, L-1…2, update the weight according to [58].

* Where L denoting the layer number, δ^{l} being the output error, bl being the bias, w^{l} being the weight and C being the cost.

3.3 Cloud

Cloud computing involves deploying groups of remote servers and software networks that allow data storage and online access to computer services or resources. Clouds can be classified as public, private or hybrid.

- **Private:** Private cloud services are delivered from a business data center to internal users. This model offers flexibility and convenience, while preserving management, control and security.

- **Public:** In the public cloud model, a third-party provider delivers the cloud service over the Internet. Public cloud services are sold on-demand, typically by the minute or the hour. Customers only pay for the CPU cycles, storage or bandwidth they consume. Leading public cloud providers include Amazon Web Services (AWS), Microsoft Azure, IBM/SoftLayer and Google Compute Engine.
- **Hybrid:** Hybrid cloud is a combination of public cloud services and on-premises private cloud with automation between the two. Companies can run workloads or sensitive applications on the private cloud while using the public cloud for bursty workloads that must scale on-demand. The goal of hybrid cloud is to create automated and scalable environment which takes advantage of all that a public cloud infrastructure can provide, while still maintaining control over mission-critical data.

Although cloud computing has changed over time, it has always been divided into three broad service categories:

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as service (SaaS)

3.3.1 IaaS

IaaS providers such as AWS supply a virtual server instance and storage, as well as application program interfaces (APIs) that let users migrate workloads to a virtual machine (VM). Users have an allocated storage capacity and start, stop, access and configure the VM and storage

as desired. IaaS providers offer small, medium, large, extra-large, and memory- or compute-optimized instances, in addition to customized instances, for various workload needs.

3.3.2 PaaS

In the PaaS model, providers host development tools on their infrastructures. Users access those tools over the Internet using APIs, Web portals or gateway software. PaaS is used for general software development and many PaaS providers will host the software after it's developed. Common PaaS providers include Salesforce.com's Force.com, Amazon Elastic Beanstalkand Google App Engine.

3.3.3 SaaS

SaaS is a distribution model that delivers software applications over the Internet; these are often called Web services. Users can access SaaS applications and services from any location using a computer or mobile device that has Internet access.

3.4 Hadoop-MapReduce

Apache Hadoop is a set of algorithms for distributed storage and distributed processing of very large data sets (Big Data) on computer clusters built from hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures of machines are common place and thus should be automatically handled in software by the framework.

The core of Apache Hadoop consists of a storage part (Hadoop Distributed File System (HDFS)) and a processing part (MapReduce). Hadoop splits files into large blocks (default 64MB or 128MB) and distributes the blocks amongst the nodes in the cluster. To process the data, Hadoop Map/Reduce transfers code to nodes that have the required data, which the nodes then process in parallel. This

approach takes advantage of data locality to allow the data to be processed faster and more efficiently via distributed processing.

A MapReduce program is collected of a Map that implements filtering and organizing and a Reduce that makes an immediate operation. The other name for MapReduce System nominated as framework that has the ability running the various tasks in parallel, managing all communications and data transfers between the various parts of the system. Also allows developers to write programs to process huge number of formless data in parallel across a distributed cluster of individual computers [60][61].

The model is inspired by the map and reduces functions usually used in functional programming. Basically a Map Reduce executes three basic operations as follows: The first task is Map function that processes in a parallel manner by each node without transferring any data to other nodes. In second operation, processed data by Map function is repartitioned across all nodes of the cluster. Lastly, Reduce task is executed in a parallel manner by each node with partitioned data [60].An overview of the Map Reduce system is shows in Figure 3-6.

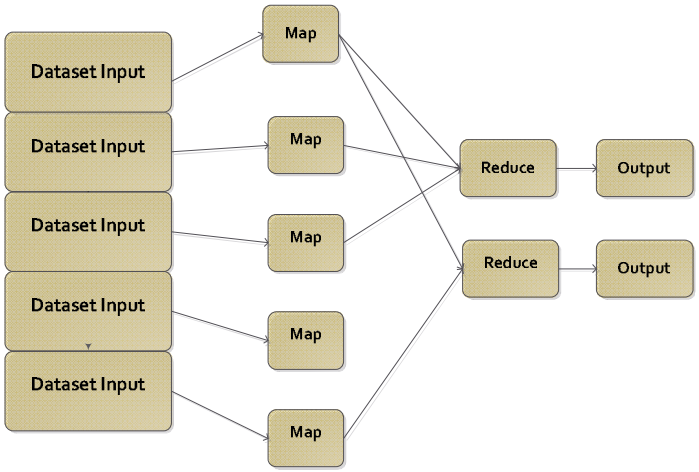


Figure 3-6 Overview of Map Reduce System [59]

3.5 Detection Region Proposal Methods

To detect objects in an image, candidate regions where objects are expected to exist are usually extracted to guide the object detection. Region proposal methods can generate a small number of semantic regions where objects are most likely present, thus avoiding an exhaustive sliding window search across the whole image. Good region proposal methods can propose high quality regions, all of which contain potential target objects. Meanwhile, the number of generated regions is also relatively small so that we can mine the regions effectively. There are many region proposal methods. A method based on sliding window searches is a primitive way to propose regions, as this approach exhaustively searches across the whole image.

In multiple-scale detection, the number of proposed windows grows in order of magnitude based on the different sizes of detection windows.

Another approach to alleviating the computing burden is to propose regions only on objects of interest. We observe that objects of interest share common visual properties most of time. Based on this observation, a much smaller number of regions that have high probability to contain objects of interest can be proposed to be candidate regions. Current state-of-the-art object detectors often employ region proposal methods to speed up the search for objects.

These approaches (multiple scale detection and object of interest detection) are more efficient than the sliding window approach, as fewer windows are examined. Currently, there are two categories of region proposal methods [62]. One category is window scoring, and the most popular method within this category is Objectness. The other category is grouping method, and the most popular algorithm within this category is known as SelectiveSearch [44].

Objectness was proposed by Alexe et al. [63] in 2010. It is among the earliest well-known region proposal methods. In this method, the salient locations in an image are used for initial proposals. Each of the salient regions are then scored based on the probability of containing an object[64]. The score estimation is based on a combination of multiple pieces of information such as color contrast, edge, location and size, saliency, and its overlap with superpixels.

SelectiveSearch was proposed by Uijlings et al. [44]. This method liberally merges similar regions together to generate candidate regions. Since the author designed similarity functions manually, there are no learned parameters. Because of its high recall rate and no customized parameters, it is the most widely-used region proposal method, and the most recent object detection methods use SelectiveSearch as the region proposal method.

Chapter 4 Proposed System

In this section, we will discuss our proposed system and categorized it into two parts: single and multiple food items.

4.1 Single Food Item

Single food item means the food image contained only one food item as shown in Figure 4-1.

Figure 4-1 Single-item Food

The algorithm for recognizing the food item is shown in Figure 4-2. To describe them precisely, we have divided them to subsections which are:

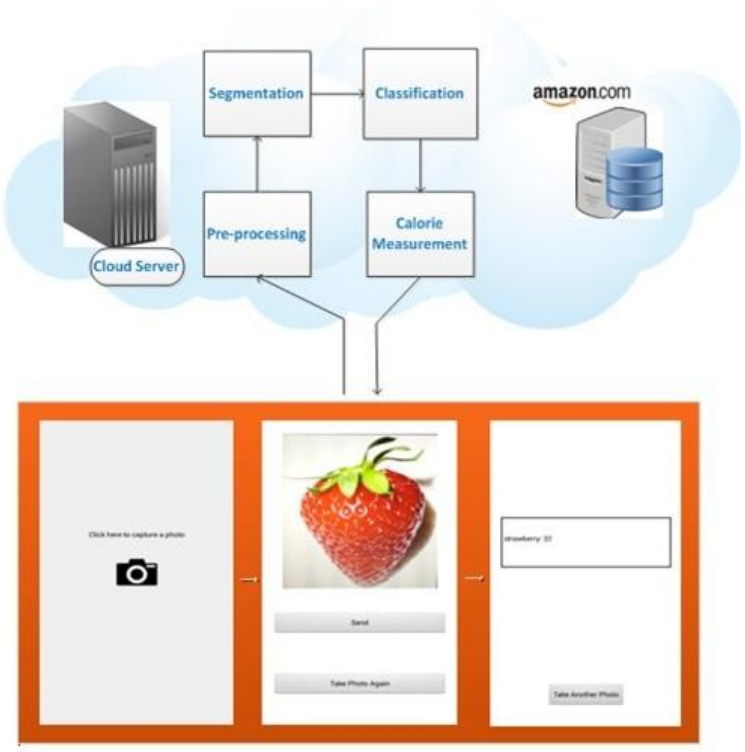


Figure 4-2 Proposed System

4.1.1 Pre-processing

First of all, in order to have accurate results for our segmentation, a simple transformation must be performed on the image to change the image size into standard format. To do so, the size of each image will be compared with standard size categorizes. If the image size is not compatible with any size category, some cropping or padding techniques will be applied to the image. We have defined one size category, i.e. 970×720 for simplicity. Larger images will be adjusted to this size, before performing any image processing technique.

4.1.2 Segmentation

In Figure 4-3, we provide a block diagram of the proposed system. The system consists of the following stages: Image acquisition and pre-processing, image segmentation, classification, and measurements.

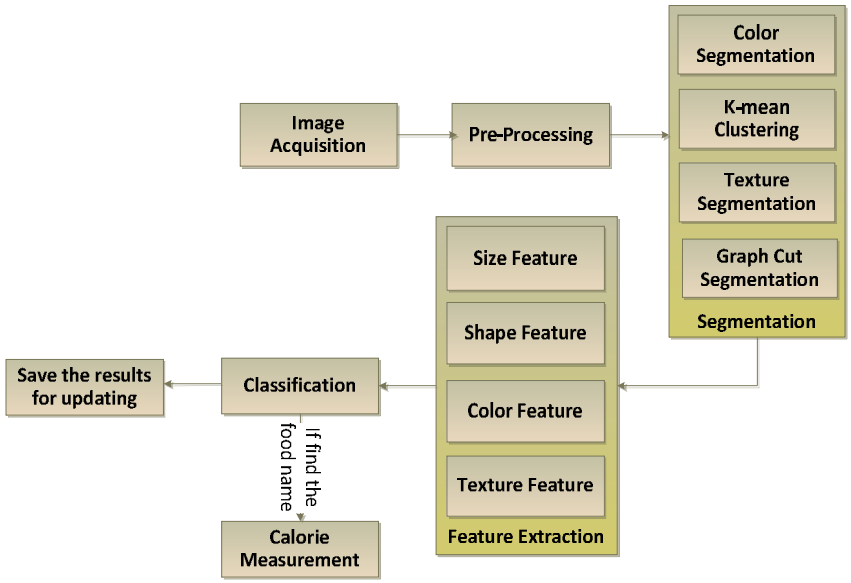


Figure 4-3 Image Analysis System

At the high level, the system works as follows: The user captures three pictures of the food with his/her thumb on a suitable position on the dish so the picture will not only contain the food item, but also the user's thumb, which is used for size calibration. The first two images are taken before the food consumption, one from the top view that will enable us to extract the portions and its corresponding areas, the other from the side of the dish, to analyze the height of the food items inside the dish is shown in Figure 4-4.

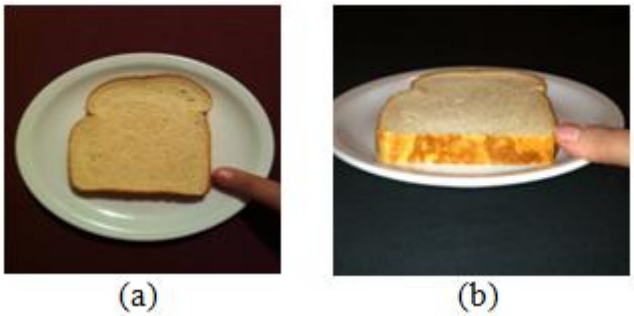


Figure 4-4 a) Image from Top, b) Image from Side

With these two measurements, we can obtain a better approximation for the volume, and its translation to calories and nutritional facts. The third picture must be taken at the end of food intake, to subtract from the calculations the food not consumed by the patient. The

technique of using the thumb in a photo captured has an important usage in our system, because the thumb is considered as a standard for calculating the dimensions of the food items. Compared to the previous measuring method such as PDAs and the calibration card, thumb is more flexible, controllable and stable standard, giving to the patient the freedom to use the application without the need to carry around uncommon equipment or in this case measurement patterns. As an alternative to the thumb (for disabled patients who might not have a thumb), the user can pace a coin inside the image, so the system will use this coin instead of the finger, to translate the portions of the food from the picture size into real life size. The system is designed to store the patient's thumb size during its one-time calibration process. Once the food is recognized and the application suggests the type of food, the user is responsible to accept or correct the type of food from the application in the mobile device. In the following subsection we will explain each step in more detail.

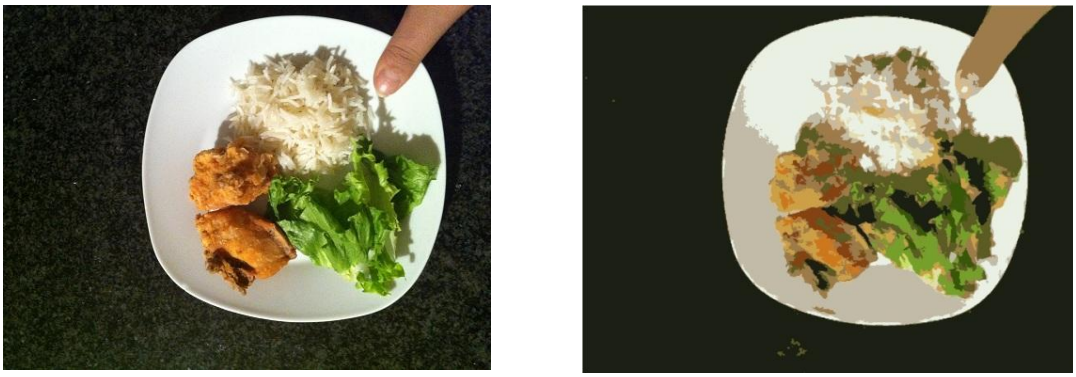
At the segmentation step, each image is analyzed to extract various segments of the food portion. We paid significant attention to the segmentation mechanism design to ensure that images are processed appropriately. Particularly, we have used color segmentation, k-mean clustering, and texture segmentation tools. In this subsection, we show how these steps and the tools used lead to an accurate food separation scheme.

4.1.2.1 K-mean Clustering

There are many clustering algorithms in the open literature such as mean shift and k-means. The mean shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. The k-mean algorithm iteratively computes the mean of a set of clusters, until it converges to a stable set of cluster samples. In gray-scale images, areas are typically modeled as uniform intensity areas. Segmentation algorithms employ some form of Euclidean distance

measure to determine pixel similarity either on a spatially local basis or on a global color basis. For color image processing, the clustering algorithms operate in complex multidimensional spaces. Because of the added complexity of needing three variables to represent color pixels, the issue of region segmentation in color images is not as well defined as for gray-scale images.

In the segmentation step of this thesis, we focus more on creating regions of similar color. This means that the choice of distance measure becomes very important since similarity depends very much on how distances between colors are being measured. In this case, all approaches found in the literature use some form of Euclidean distance to determine similarity between two color pixels Figure 4-5 shows the result of this clustering scheme on a sample food image. As the figure shows, different colors are clustered appropriately.



a) Original images b) Color clustered images

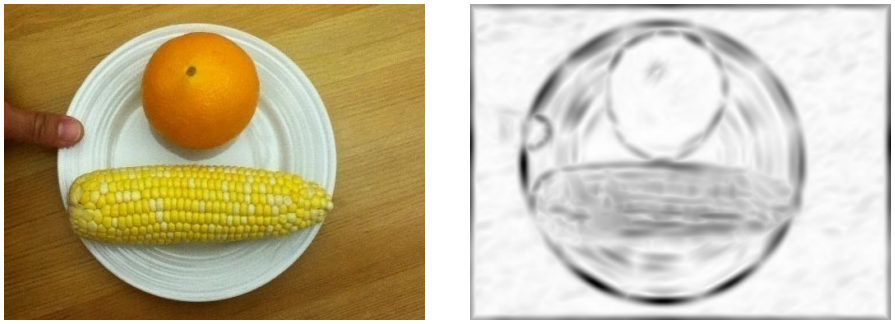
Figure 4-5 Color clustering results

4.1.2.2 Texture Segmentation

To obtain more accurate results in the segmentation stage, we added texture segmentation to the method. For texture features, we used a Gabor filter to measure local texture properties in the frequency domain. The Gabor filter describes properties related to the local power spectrum of a signal and has been used for texture analysis [65]. A Gabor impulse response in the spatial domain consists of a sinusoidal plane wave of some orientation and frequency modulated by a two-dimensional Gaussian envelope and is given by equation (4-1).

$$h(x,y) = \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)\right]\cos\left(2\pi Ux + \varphi\right) \quad (4\text{-}1)$$

It is suitable for our use, where texture features are obtained by subjecting each image to a Gabor filtering operation in a window around each pixel and then estimating the mean and the standard deviation of the energy of the filtered image. A Gabor filter-bank consists of Gabor filters with Gaussians of several sizes modulated by sinusoidal plane waves of different orientations from the same Gabor-root filter as defined in (4-1). In our implementation, a bank of Gabor filters with six different desired orientations and five wavelengths are applied to the image. Furthermore, we have included the spatial coordinates of the pixels as two additional features that we have in our segmentation to get an accurate result in this part. The outcome of each of these Gabor filters is a two-dimensional array, with the same size of input image. The sum of all elements in one such array is a number that represents the matching orientation and spatial frequency of the input image. The results of applying a Gabor filter on food images are shown in Figure 4-6.



c) Original images d) Gabor filtered images

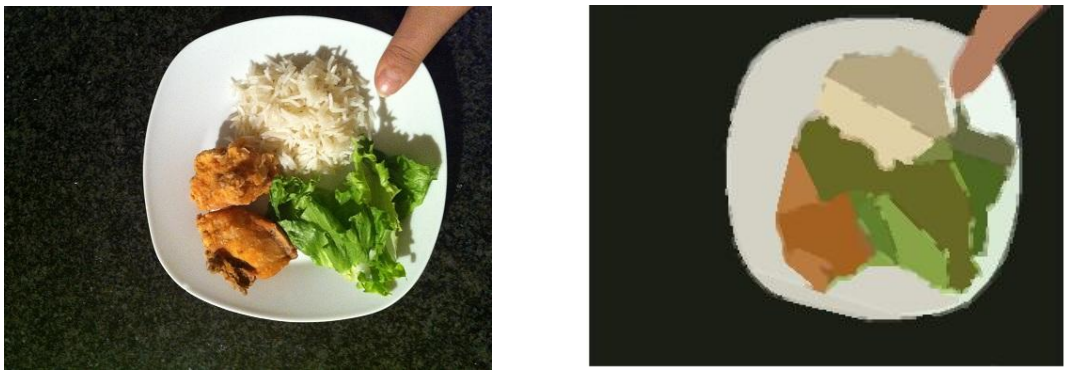
Figure 4-6 Gabor filter results

4.1.2.3 Graph Cut Segmentation

Before performing the segmentation on the image, the user captures the picture of the food with her thumb on a suitable position on the dish so that the picture will not only contain the food item, but also the user's thumb, which is used for size calibration. To use graph cut segmentation, it is important to determine the features of a good graph that will extracted from

an image. These three properties are most three steps of our graph cut based food image segmentation method.

First, it should be robust; i.e., if the image is somewhat distorted, the graph should not be deeply changed. In graph cut, each pixel of the image is mapped onto a vertex in a graph. Neighboring pixels are connected by weighted edges where the weight is determined based on a predefined energy function. In the normalized cut approach, the cut cost is determined by the fraction of the total edge connections to all the vertices in the graph. Second, it should also have good algorithmic properties. This means that the graph, when drawn, is actually a symbolic representation of the image; for instance, the boundaries between the regions should match with the edges of the graph. Third, we would like to be able to rebuild an image from the graph and for this new image to be a good compression of the initial image. In other terms, the loss due to the extraction process should be minimal. An example is shown in Figure 4-7.



a) Original images b) Graph cut images

Figure 4-7 Graph cut results

4.1.3 Classification

In this stage, the extracted features are classified in order to recognize each food portion. For this purpose, we used two different classifiers: SVM and Deep learning neural network, which are popular techniques used for data classification. A classification task usually

involves training and testing data, which consist of some data instances. Each instance in the training set contains one class label and several features.

4.1.3.1 SVM

Once the food items are segmented and their features are extracted, the next step is to identify the food items using statistical pattern recognition techniques. Afterwards, the food item has to be classified, using SVM mechanism [66][67]. SVM is one of the popular techniques used for data classification. A classification task usually involves training and testing data which consist of some data instances. Each instance in the training set contains one class label and several features. The goal of SVM is to produce a model that predicts target value of data instances in the testing set, which is given only by their attributes. In our model, we use the radial basis function (RBF) kernel, which maps samples into a higher dimensional space in a non-linear manner. Unlike the linear kernels, the RBF kernel is well suited for the cases in which the relation between class labels and attributes is nonlinear. In our proposed method, the feature vectors of SVM contain 5 texture features, 5 color features, 3 shape features, and 5 size features. The feature vectors of each food item, extracted during the segmentation phase, will be used as the training vectors of SVM. For increasing the accuracy, after the SVM module has determined each food portion type, the system can optionally interact with the user to verify the kind of food portions. For instance, it can show a picture of the food to the user, annotated with what it believes are the portion types, such as chicken, meet, vegetable, etc. The user can then confirm or change the food type. This changes the system from an automatic one into a semi-automatic one; however, it will increase the accuracy of the system.

4.1.3.2 Deep Learning Neural Network

In this section, we provide the details of the deep neural network method used in our system. The first step in our approach is to generate a pre-trained model file with the help of CNN network. This is performed by initially capturing a set of images of one particular class (e.g. 50 images of apple class) and then labeling them with object name-set (object being apple). These images will be considered as the set of relevant (positive) images and are used to train the system. In the second step of the training, we re-train the system with the set of negative images (images that do not contain the relevant object). In our case, we trained the system with the background images, so it does not categorize them as part of the apple class. Once the model file is generated from the training, we load it into the application and test it against the images captured and submitted by the user. The system then performs the image recognition process and generates a list of probabilities against the label name. The label with the highest probability is prompted to the user in the dialog box, to confirm the object name. Once the object name is confirmed, the system performs the calorie computation part by calculating the size of the food item with respect to the finger in the frame. It finally prints the output to the user with the required calorie. Figure 4-8, illustrates the above mentioned process. We trained the system using the deep neural network model by [70], with various classes of food samples. Figure 4-9 shows an example for an apple class. The system is trained with the apple in the plate, from various angles and with different apples, so that the system gets trained with different models of the apple class. These would be the positive image frames or the relevant image frames.

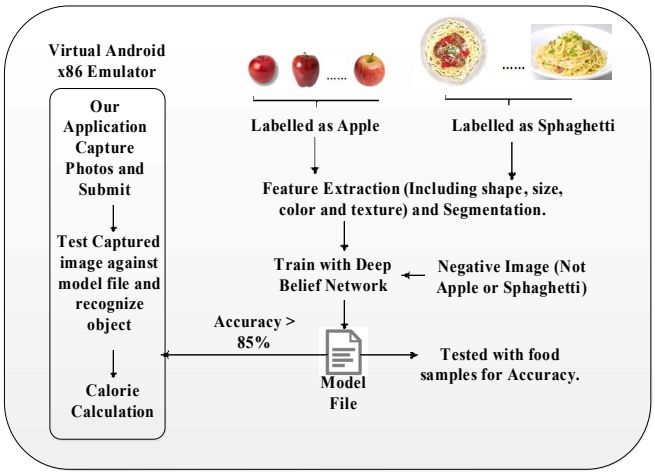


Figure 4-8 Implementation of Deep Belief Network in the Android Application

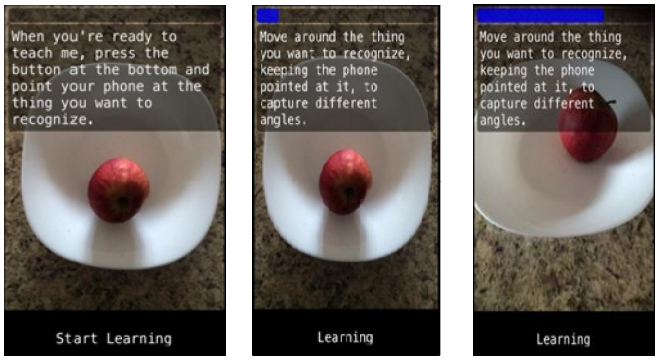


Figure 4-9 Training the Deep Neural Network with Apple Class

Once the system is trained with the positive sample, we then re-train the system with the negative samples. By negative samples, we mean the objects that should not be recognized. In our scenario, we took frames of an empty plate, which should not be learned as part of the background image frame. Also, we used a different object (e.g. eggplant on the plate) and some other background image frames, as part of the negative samples. After training the system with negative samples, the system generates a model file, which is further used to recognize objects in the application. Figure 4-10 shows an example of training the system with negative samples.

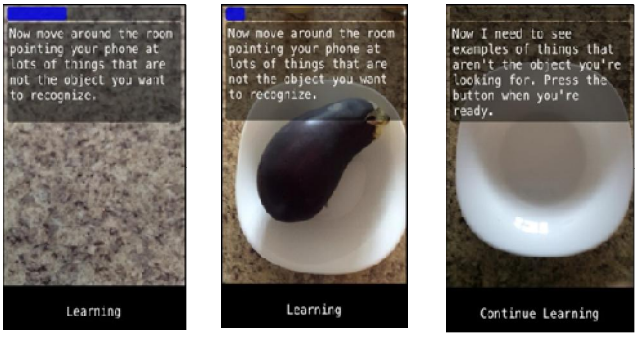


Figure 4-10 Training the Deep Neural Network with Eggplant Class

4.1.4 Cloud

In this section, we describe the MapReduced SVM method for cloud based food recognition.

In order to design a Map Reduced SVM [68], we used the Elastic map reduce (Amazon EMR). The method further uses Hadoop, to distribute the data and process it across resizable size Amazon EC2 instances. Hadoop uses a distributed processing architecture called the Map Reduce in which a task is mapped to a set of servers for processing. The results of the computation performed by those servers were then reduced down to a single output set. One node, designated as the master node, controls the distribution of tasks to the slave nodes. The overview of the model is shown in Figure 4-11.

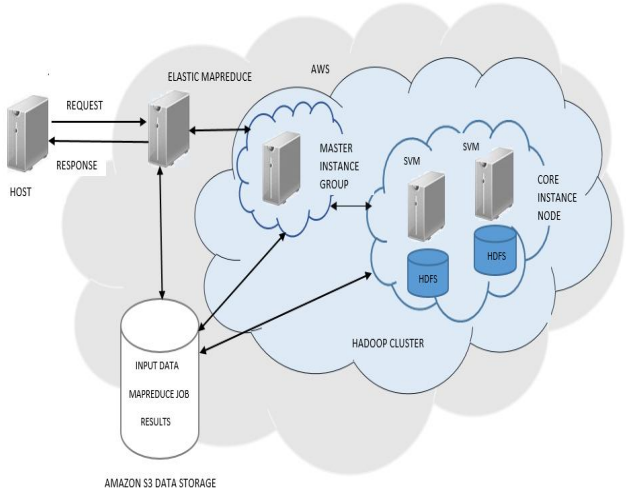


Figure 4-11 Hadoop Integration on Amazon EC2/AWS

As shown in Figure 4-11, initially a request is sent to Amazon EMR to start the cluster. Once Amazon EMR creates the Hadoop cluster with a master instance group and core instance group, the master node is added to the master instance group. The slave node is then added to the core instance group. Our system deals with huge set of images and processes them to derive results in various formats. Managing such bulk of data can become a tedious job. To overcome such issue, we perform the processing and the storage in a cloud based platform. Using the cloud platform not only satisfies the computational complexity constraints of our system but also helps achieving higher accuracy for food classification. The data including the images, non-trained, trained and non-tested results, logs and the Map Reduce job are all stored in the Amazon S3 bucket. Amazon S3 is an Internet storage service. It is secure, reliable, scalable, fast and inexpensive storage platform, which makes it easier for us to manage the bulk data. Amazon S3 stores data as objects within buckets. An object is composed of a file and optionally metadata that describes the file. To store the file, one must upload the file to the bucket. In our system, we create a bucket named Food Recognition-test and made four subfolders in the bucket. The job folder contains the SVM Map Reduce jar file, the data folder contains the input file (svm_data.train) and logs folder saves the log info whenever the Map Reduce job is run. The implementation algorithm is shown in Figure 4-12.

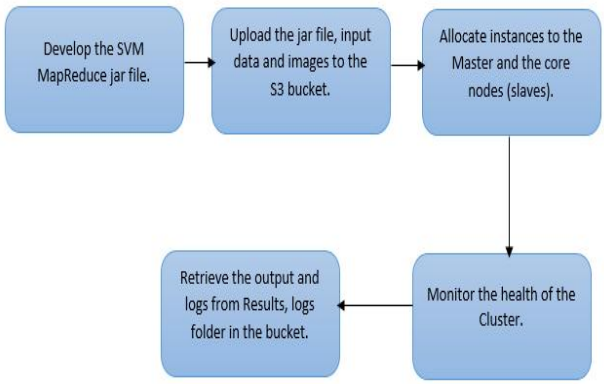


Figure 4-12 Implementation of Map Reduce

As shown in the Figure 4-12, the Map Reduce implementation has 5 stages. In the first stage, we develop the SVM Map Reduce jar file. It includes the mapper class, the reduce class, the combiner class, and the main Map Reduce java class. Since the segmentation is done with the help of Support Vector Model, we have implemented it with SVM training and testing class, which we further call from the main Map Reduce class. In the second stage, we store the data and the jobs in the bucket. In the third step, we provision an Amazon EMR cluster, wherein the instance size of the node directly impacts the performance of the cluster. This further depends on the workload being CPU intensive or disk (I/O) intensive or memory intensive. For memory intensive jobs, m1.xlarge or one of the m2 family instance sizes have enough memory and CPU power to perform the work [69]. In our case, since we ran the job with fewer samples, we assumed that it would take lower memory and lower CPU. After the cluster is provisioned, we launch the job workflow, which further includes provisioning cluster, running bootstrap action, running the job flow and shutting down the cluster. In the fourth step, once the job flow is started, we can use the cluster monitoring interface in AWS (Amazon Web Services) to further monitor the cluster status, Map Reduce job, node status, IO and the hardware configuration. Finally, in fifth step, the results are stored in the results folder, and the logs are stored in the logs folder of the bucket. In the next section, we present the details of the implementation of our system.

4.1.4.1 Proposed Measurement Method

CloudSVM is built on the SVM and implemented using the Hadoop implementation of MapReduce. The implementation of Map Reduce for the SVM model can be categorized into the following steps: First, statistics computation for features (color, size, shape etc.) and class objects. Second, transform the sample by implementing the SVM model, after that, computing statistics for new feature space and finally distributing the new samples and training the model

in a random order with the reducer function. The SVM model is implemented in parallel with the help of Map Reduce mechanism wherein each instance is trained with a SVM model. The support vector of each subSVM are taken as input of next layer subSVM [69]. The non-support vectors are filtered with subSVMs. Furthermore, CloudSVM is a MapReduce based SVM training algorithm that runs in parallel on multiple computers with Hadoop.

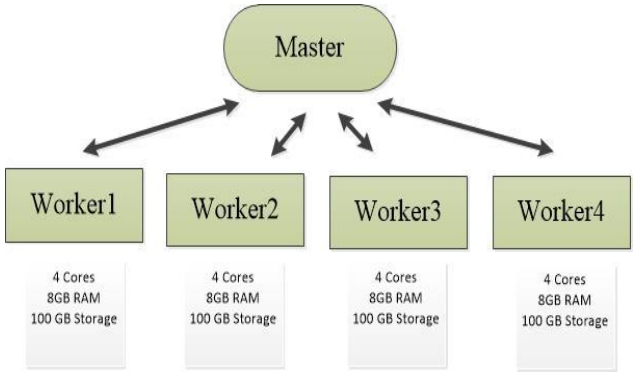


Figure 4-13 Cluster Configuration

4.1.4.2 Building SVM in parallel and in batch from scratch

In order to classify input images into different food categories, we use a support vector machine based classifier. For building an SVM and for scaling the actual build task with increasing number of images, we utilize a cluster of workers mastered by one head node; the model is shown in Figure 4-13. Our cluster contains five Nodes. One of these nodes is called "Master" and the remaining nodes are called Workers. Each and every node in our cluster has the same resource configuration: 4 core CPU, 8 GB RAM memory and 100 GB disk space. Each machine runs Apache Hadoop version 1.0.4. Apache Hadoop is an actual implementation of MapReduce task execution framework [80], which is primarily used for efficiently processing very large datasets in parallel and offline. A Hadoop-based framework mainly consists of two main components: (1) Hadoop Distributed File System (HDFS) and (2) Hadoop MapReduce. HDFS is a distributed, fault-tolerant, and highly scalable file system. It requires a namenode that intercepts, accepts, and serve all file access requests. It commands an army of datanodes that

are used to store the actual data blocks. On the cluster, we installed an HDFS headed by the master as the namenode and the workers acting as the datanodes.

A MapReduce job consists of a map task and an accompanying reduce task. Tasks operate on data that is stored in the HDFS. For a given map task, a core responsible to execute the task at any of the workers independently processes a data partition of size equal to the HDFS block size. Each map task runs in serial in itself. Once all map tasks are finished, the interim results produced by the map tasks are co-located around a key identifier at a destination node, where they are to be reduced to a final result by a reduce task. In order to build an SVM on Hadoop, we used a cascade-SVM implementation. The image features are stored on our HDFS cluster. In order to test the scalability .of building the final SVM, we ran a test on 250K, 500K, 750K, 1M and 1,5M images respectively. The results are shown in Table 4-1.

Table 4-1 Runtime for building a cascade-SVM on a cluster of 5 machines

Total number of images in the training dataset.	250K	500K	750K	1M	1.5M
Time required to build SVM distributed (in min).	3.5 mins	5.1 mins	19.3 mins	51.7 mins	66.4 mins

4.1.4.3 Maintaining an SVM online with incoming new images

Instead of building an SVM from scratch that takes more than an hour on 1.5M images, we defer the build task as long as the SVM accuracy is above a pre-determined threshold. This threshold dictates when it is acceptable to continue updating an already built SVM model with new image features and class labels online. The main algorithm that drives this aspect is outlined below in Algorithm 1.

Algorithm 1

Let S denote the set of images S accrued so far in the HDFS. We use a certain percentage of S for training denoted by L and the remaining part is used for testing denoted by T. Note that S satisfies

S = L + T.

A. Build a cascade-SVM denoted by M using the images in L.
B. Test M on the set of images T. Measure the classification accuracy, x as Benchmark.
C. With every new incoming image I,

C.1. Add the image to the set of training images, i.e., L = L + I

C.2. Incrementally update M using I to obtain M'.

C.3. Test M' on T again and measure its classification accuracy as y.

C.4. If x > y, then retrain a new SVM from scratch on L + I, obtain model M''.

C.5. Test M'' on T again and measure its classification accuracy as z.

C.5.1 If z > x, use M'' else if z < x, use M for predictions.

C.6 If x < y, use M' for image classifications.

**In the above algorithm we assumed x to be the threshold for Degree of Inaccuracy (DoI), it can be adjusted based on the accuracy constraints of the system [75].*

4.2 Multiple Food Recognition

In real-life scenarios, a food image of a meal mostly includes multiple food items, as shown in Figure 4-14.

Figure 4-14 Multi-item Food

Recognizing multiple food items and categorizing them in a single food image is a challenging task. Our goal is to learn a detector for each food category from a set of training images with only binary labels indicating whether the image contains this food category or not. We model an image as a set of overlapping rectangular windows. The binary image label y = 1 specifies that the image contains at least one instance of the target category; the label y = -1 specifies that the image does not contain instances of the category. Since instance labels are not provided, we perform region mining with submodular cover optimization to find positive instances, which are rectangular windows containing the target food category. With these positive instances, we can train a classifier for each food category.

In this section, we propose an algorithm which jointly detects food item combinations in the plate. In this algorithm, each training image contains only a single food item, while the testing images may contain multiple food items. Our proposed algorithm will predict the labels of food items and estimate the calorie of the plate. The diagram of the algorithm is shown in Figure 4-15.

We describe each step of our proposed algorithm in detail and offer a brief motivation for why certain methods and parameters have been chosen.

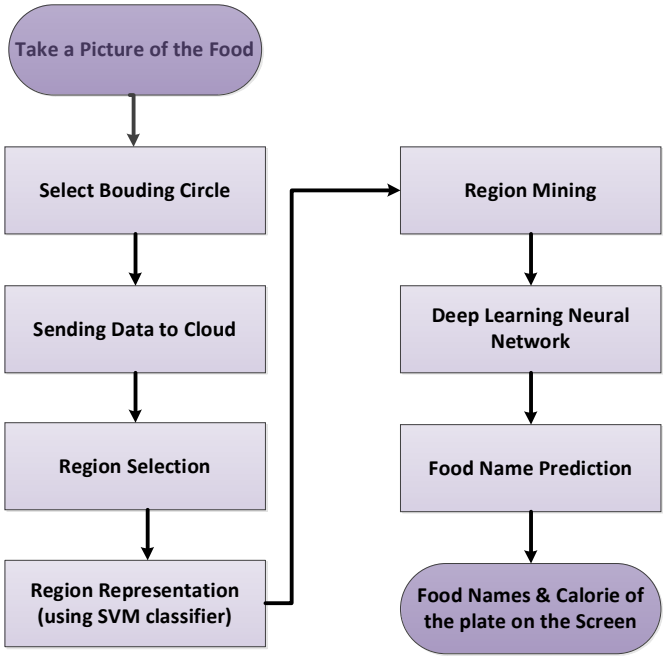


Figure 4-15 Block Diagram of Our Proposed Food Recognition System

4.2.1 Bounding Circle

In the first step, for having a more accurate system, after taking a picture with the mobile device, it is better to remove the background from the main objects as much as possible and then save the new image. To do so, we ask the user to draw a bounding circle around the plate of the food, sown in Figure 4-16, to make it simpler for the next processing steps.

Figure 4-16 Bounding circle

4.2.2 Sending Data to Cloud

The process of sending data to cloud is the same as the one depicted in section 4.1.4, so we don't repeat it here and refer the reader to that section for more information.

4.2.3 Region Selection

In this step, we used Selective Search [76] to generate region proposals for multiple-food detection, and the algorithm we chose is described below.

Selective Search has been broadly used as a detection proposal method by many object detectors. The idea of this method is grouping superpixels to generate a hierarchy of small to large regions. Usually, more than 1,000 regions can be produced for a single image by Selective Search. We further filter the candidate regions based on their size and aspect ratio. Regions that are either too small (area is less than 1/100 of the original image) or the ratio of length to width is too large (greater than 20) or too small (less than 1/20) are removed. With this simple and reasonable filtering, we successfully decrease the number of proposed regions per image from

about 1,000 to about 150. This significantly improves the speed and accuracy of the subsequent procedures.

The processing procedures of this algorithm are shown in Algorithm-1:

Algorithm-1: Region Selection [76[` q

---

1. Generate initial regions.

2. //Generate as many regions as possible and each of them belongs to one object.

3. Merge similar regions into larger ones, until only one region is left.

4. Merge criteria based on the color and texture similarities.

5. //The similarities are measured by the similarity metric, which is defined as follows:

6. Choose the color space from RGB, HSV and Lab color spaces.

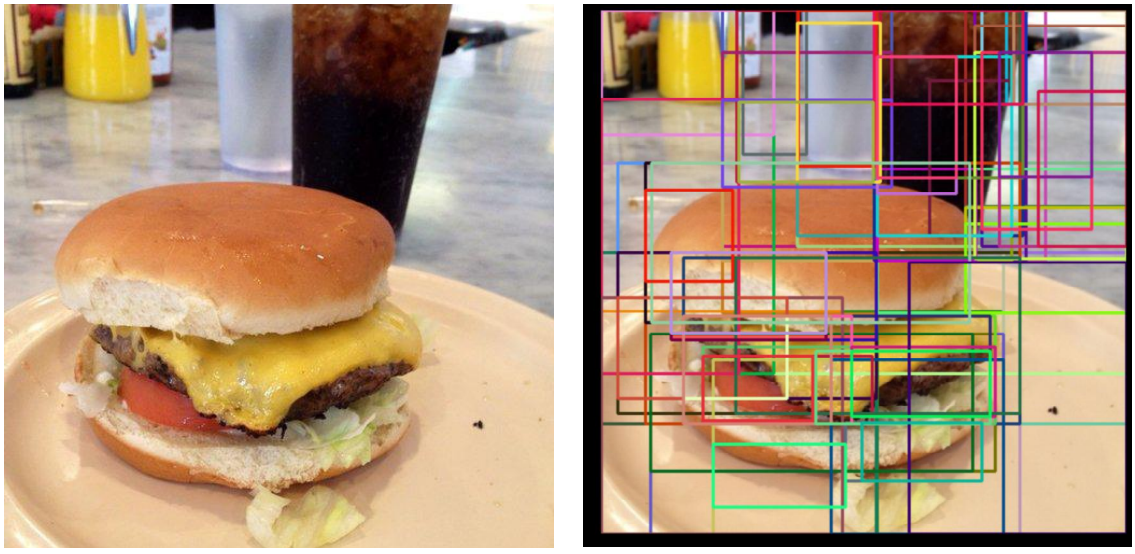
7. Compute the HOG-like features for texture similarity measurement.

8. Compute size component in the similarity metric.

9. Measures the similarity of shape between two regions.

10. The function of the similarity = linear combination of all above metrics.

11. Output regions contain candidate objects.



(a). The Food image (b). Proposed region windows by SelectiveSearch

Figure 4-17 Example of a food image (left) with proposed regions computed by SelectiveSearch(right)

4.2.4 Region Representation

Extracting features to represent the regions in the featured space is an important component of object detection. In our experiments, we use the [70] deep learning framework to compute the feature vectors of the proposed regions. Then the pixel values of the region are subtracted from the mean and then normalized to reach between [0.0, 1.0]. The normalized pixel's values are then forward propagated through the neural network, and a 4,096-dimension feature is extracted by using the output of the last convolutional layer in the model. Since we decreased the number of proposed regions from about 1,000 to about 150 per image, after extracting features for all these regions, a feature matrix of 150 x 4,096 is computed for each image. This feature matrix is used as the input data for the region mining process.

4.2.5 Region Mining

Although the SelectiveSearch algorithm does propose many regions that contain objects, most of those regions do not contain the target objects we want to detect, or else they just contain part of the target objects. Region mining aims to find two types of regions from all the proposed regions of training images. For positive regions, based on the fact that different proposed regions have different kinds of discrimination toward the target object class, a region mining procedure evaluates the discrimination in order to discover the positive regions that would best differentiate the target object class from the backgrounds. Hard negative regions consist of two parts: the background regions from positive images belong to the same class, and the regions from negative images belong to other classes. Hard negative regions are samples that are similar to positive regions and are falsely detected by the classifiers.

After we get the scores of each region, we want to select a set of regions $S \subseteq R$, such that:

a. S can well represent the category.

b. S is small enough for fast training.

The most important problem with submodularity is how many regions are needed to represent one food category. The representation margin diminishes by adding one more region to set S compared to previously added regions. We define the function N(S) as the neighbors of a set of regions $S \subseteq R$, shown in Figure 4-18.

A set function $F: 2^{\Omega} \rightarrow R$ is submodular if it satisfies diminishing gains [77]. That is (4-2), for any $S \subseteq T \subseteq \Omega \backslash \{v\}$, it holds in equation (4-2):

$$F(T \cup \{v\} - F(T) \leq F(S \cup \{v\}) - F(S), \quad \forall v \in \Omega \tag{4-2}$$

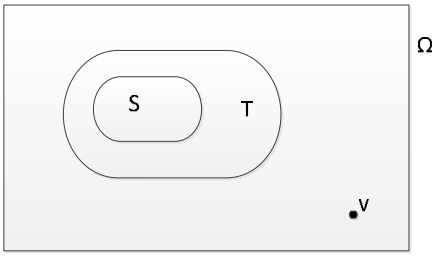


Figure 4-18 Submodular Set Function

We define a covering score $C_{I,t}(S)$ for each image I. The score is determined by a covering threshold t and the number of regions from R_I that are in the set of neighbors of S, that is, $N(S)$, which is described in equation (4-3).

$$C_{I,t}(S) = \min\{t, |N(S) \cap R_I| \tag{4-3}$$

The covering score $C_{I,t}(S)$ measures how many regions in R_I are the near neighbors of S. If there are many regions in R_I which are the near neighbors ofS, then set S can well represent the image I, and we say that set S covers image I. The covering score $F(S)$ from equation (4-4) for the whole set S is then defined as sum of scores on all positive images P of that category.

$$F(S) = \sum_{I \in P} C_{I,t}(S) \tag{4-4}$$

Our target is to find a set of regions S from R that minimizes equation (4-4), while having enough representation ability. We want regions in S to come from each of the positive images, and at

the same time to keep S as small as possible. This is indirectly controlled by the threshold parametert. If t is small, then regions in S will come from regions from many different images and the set S is relatively small. On the other hand, if t is large, there will be many more regions in the set, which covers each image more.

To minimize $F(S)$, we want regions in S with the most number of edges from positive images. An edge e exists between R and I if the region r_i from R is the nearest neighbors of the region in$I$. That means the function $F(S)$ is a submodular function and can be optimized via a greedy algorithm. This is because function $2^{\Omega} \rightarrow R$ defined in Equation (4-4) is a submodular function.

A set function that satisfies decreasing gains is submodular. That is, for $v \in \Omega$ and $S \subseteq T \subseteq \Omega$ it holds $F(S \cup \{v\} - F(S) \leq F(T \cup \{v\}) - F(T)$.

Let $Cov = |N(S) \cap R_I|$, then function Cov is a covering function. For $S \subseteq T \subseteq \Omega \backslash r$, we can have equations (4-5) and (4-6).

$$N(S) \subseteq N(T) \tag{4-5}$$

$$|N(T \cup \{r\}| - |N(T)| = |N(r) \backslash N(T)|$$
$$|N(T \cup \{r\}| - |N(T)| \leq |N(r) \backslash N(S)| \tag{4-6}$$
$$|N(T \cup \{r\}| - |N(T)| = |N(S) \cup \{r\}| - |N(S)|$$

Thus function Cov is a non-decreasing submodular function. Function $F(S)$ is the sum over Cov, thus also is a non-decreasing submodular function.

Optimization: Our goal is to select a representative minimum subset $S \subseteq R$, which can be stated as in equation (4-7) :

$$\min|S| \ (\text{for } S \subseteq R) \text{ , } F(s) \leq \propto F(R), \text{for } \propto \in (0,1) \tag{4-7}$$

For this, we have used the algorithm discussed in [77]. Let $S_0 = 0$ and in each following iteration step I, add the region r that can cover the largest number of uncovered regions. More formally, add the region that could maximize the marginal gain$F(S_I \cup \{r\} - F(S_I)$.

Figure 4-19 shows the top-10 regions selected from the greedy algorithm for the burger training images with the $\propto = 0.95$.

Figure 4-19 The top-10 regions in set S for the training dataset of the burger category

4.2.6 Deep Learning

As mentioned above, we discussed deep learning algorithm in section 4.1.3.2, but for having better accuracy in food recognition we created new model for our multiple food recognition. For food image feature extraction and classification, deep belief networks (DBN), which are based on the concept of deep learning, allow us to achieve higher accuracy and better generalization, even with small datasets. This is a well-known property of deep learning. It further gives us the flexibility to integrate its feature extraction model with other linear classifiers, including SVM. Hence, for food recognition part, we use DBN, wherein we initially perform unsupervised training without labelling the images, which further assist the deep learning network to find the suitable parameters (determines the hidden nodes and the edge parameters) that is further used as part of the supervised learning to achieve accurate results. The back-propagation algorithm as part of the supervised learning is used to compute the gradient of the cost function quickly [78]. Training the deep neural network in this way will allow us to make the necessary changes to the weight and bias, from which we obtain the desired results. We label these

random training inputs X1, X2,..., Xm. The stochastic gradient descent algorithm then selects a randomly chosen mini-batch of training inputs, and trains with those. As shown in Figure 4-20, DBN is a hierarchical model with many hidden layers in the network. Two hidden layers combine to form the RBM (Restricted Boltzmann Machine) which further form stack of RBMs. Each layer, including the hidden layers, contain a set of neurons, i^k being the input vectors, h^k being the hidden neuron and O^k being the output neuron of the k^{th} layer in the figure. As part of the feature extraction process, the unsupervised and supervised learning (backward propagation) together generate the desired feature vectors that are used by the classifiers (feed forward propagation). Hence, we derive the high level features from the low level features further. The prediction is based on the probabilistic model, where $\mathrm{p(label|n\ input\ image)}$ is the probability of the match with the corresponding label set.

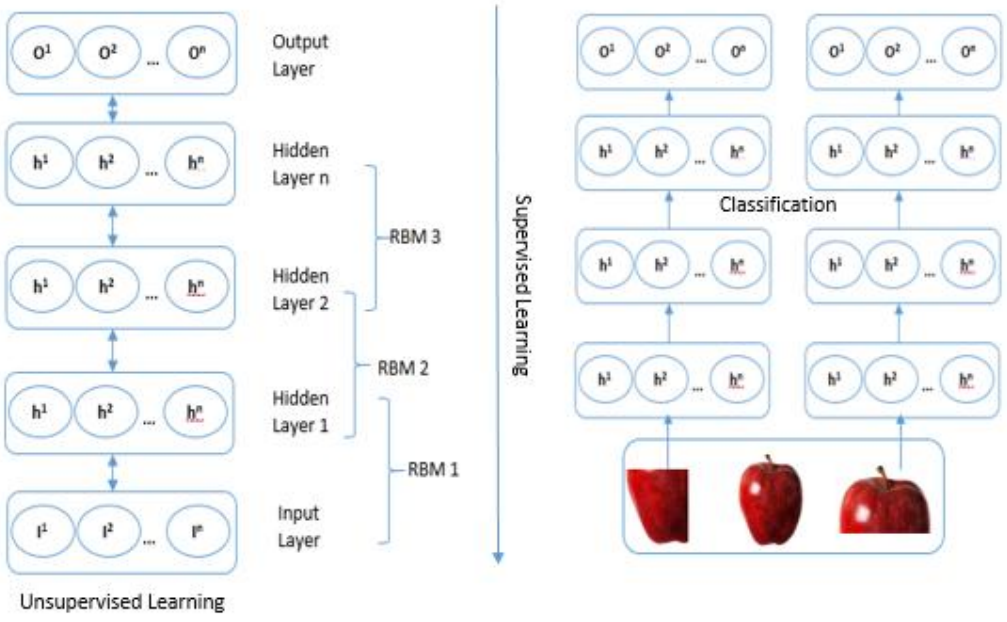


Figure 4-20 Deep Belief Network Model for Food Recognition

4.2.7 Food Name Prediction and Calorie Extraction

The recognition step will generate a list of food items ordered in occurrence probability. This means that for each food item we may have 3 or 4 predicted tags. The proposed system will

choose the food item with the highest probability. After recognizing the food names, we can estimate the calories in the food plate as described in [79].

Chapter 5 Implementation and Results

In this chapter, we describe how we train and evaluate the proposed method on a manually collected food dataset. Detailed experiment setup is given in the following sections.

5.1 Food Dataset

The collection of the food images in our dataset, we divided the food images into two different collections; single food portions and mixed food portions. We took into consideration important factors that affect the accuracy of our results. Specifically, we used a variety of the following components:

- **Camera**
  The camera itself will have an effect on the results in terms of its lens, hardware, and software. As such, we used three different cameras for our experiments, consisting of Canon SD1400, iPhone 4, and Samsung S4 cameras.
- **Lighting**
  Lighting and illumination is one of the important parameters which affect the system outcome because illumination directly affects the image segmentation algorithm, which in turn affects the rest of the algorithms. To take this into account, we put the same plate in three different locations with different illuminations and we took pictures.
- **Shooting Angle**
  Another effective parameter is the angle of photography; we have chosen three different angles which are approximately 30, 90, and 150 degrees from the plate of food

for all pictures. This means that for each plate in 3 different lighting locations we have also taken 3 pictures from different angles.

- **White Plate**
  For all images we have considered a white plate to ignore the background of the images. By using white plate, food segmentation and food recognition will be easier to perform.

- **Thumb**
  The thumb of the user and its placement on the plate are also shown in Figure 5-1. There is a one-time calibration process for the thumb, which is used as a size reference to measure the real-life size of food portions. Compared to the calibration methods of similar systems, using the thumb is more flexible, controllable, and reliable. For users with thumb disability or amputated thumbs, another finger or a coin can be used instead, the latter still more ubiquitous than special plates or cards used in other systems.

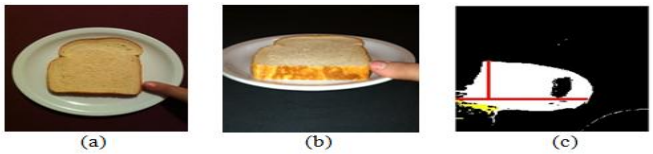


Figure 5-1 (a, b) Test images with food and thumb (c) Calculation of the thumb dimensions

In the dataset, the images are divided into 6 categories considering the capturing device, background, and lighting condition. For example, images in category 1 are captured with a Samsung camera, within a light environment with a white background, and from different shooting angles.

Table 5-1 Different Food Categories

Category	Camera	Background	Lighting
1	Samsung-S4	White	Light
2	Samsung-S4	White	Dark
3	IOS-4	White	Light
4	IOS-4	White	Dark
5	CanonSD1400	White	Light
6	CanonSD1400	White	Dark

The food dataset used for our experiments is composed of a training set and a test set. For training, we collected 7000 images of 30 categories of food. Each one of the training images only contains one food item which is shown in Figure 5-2.

Figure 5-2 Examples of training (left) images in the food dataset and testing (right) images. The training dataset contains images with only single food item and the testing dataset contains images with multiple food items.

5.2 Single Food Item

5.2.1 Applying Graph cut Segmentation and SVM classifier

In the first step, we applied color and texture segmentation in [71]-[73]. We applied our method to three different categories of food: single food, non-mixed food, and mixed food. We achieved better accuracy for non-mixed food, and our method had problems detecting some mixed foods. For example, the appearance of food portions in a mixed food may change as they get mixed, making it harder to extract different food portions. Furthermore, the size of food portions in different mixed foods are not similar, hence the method fails to segment food portions properly. To solve these problems, we have applied graph cut segmentation to improve our segmentation step. We also trained the system with more mixed food to get more accurate results.

We have chosen a data set that comprises of 15 different categories of food and fruits. These food and fruit images are divided into training and testing sets, in which around 50% of the fruit images from each group is used to train the system, and the remaining images serve as the testing set.

As Figure 5-3 shows, applying graph cut segmentation produces better region boundaries in our segmentation phase, which helps us to extract more accurate features from the segmentation phase.

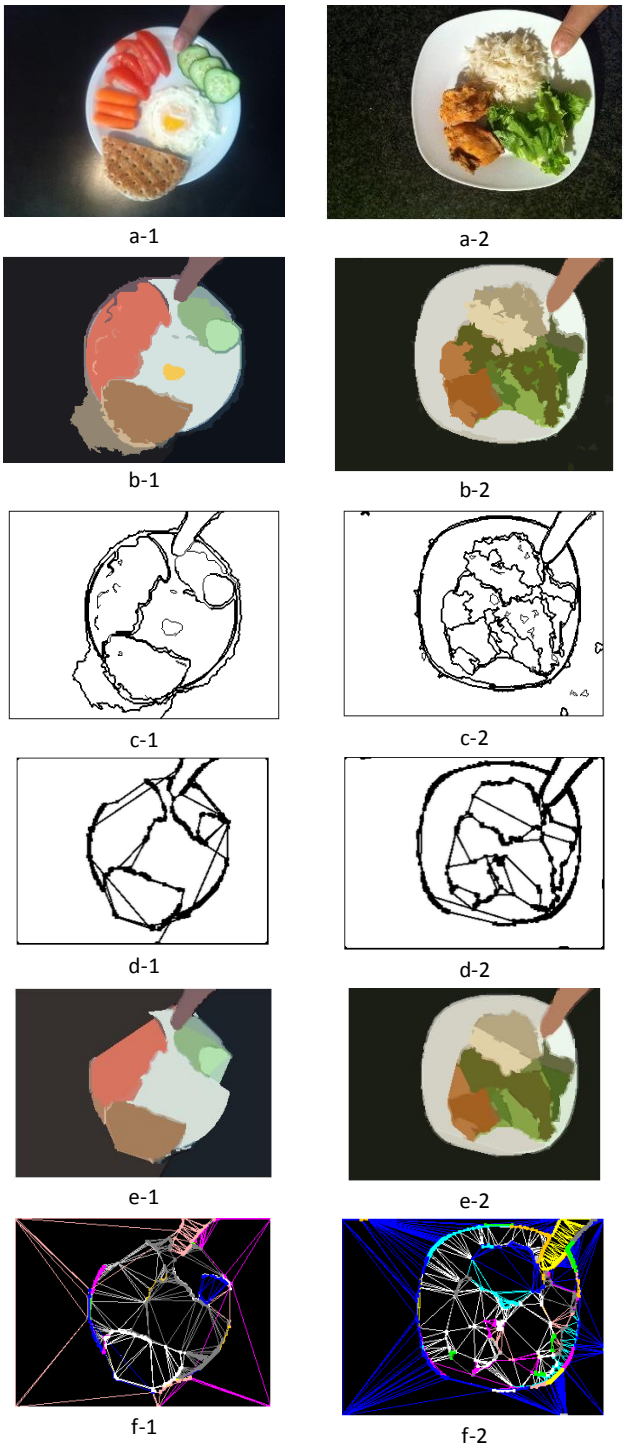


Figure 5-3a-1,a-2: Original Image, b-1,b-2: Image reconstruction from segmentation, c-1,c-2: Segmentation boundaries, d-1,d-2: Image of the graph, e-1,e-2: Image reconstruction from the graph, f-1,f-2: Region based triangulation of the graph

The results of this step feed into our SVM model to recognize each food portion. As can be seen in Table 5-2, we have an approximately 3% increase in our single food classification accuracy

in comparison to our previous model [73]. In addition to single foods, we applied our method to two other categories of food: non-mixed food (e.g. steak and fries), and mixed food (e.g. curry or soup). The resulting SVM's accuracy is approximately 90% and 75%, respectively. In comparison to our previous model [73], we achieved a 5% and 15% increase in accuracy for non-mixed and mixed food, respectively.

Table 5-2 Food Recognition Accuracy for Single Food

No.	*Food items*	*Recognition Rate (%)*	
		Using color-texture segmentation	*Using graph-cut, color-texture segmentation*
1	Red Apple	97.64	100
2	Orange	95.59	97.5
3	Corn	94.85	96
4	Tomato	89.56	95
5	Carrot	99.79	100
6	Bread	98.39	99
7	Pasta	94.75	98
8	Sauce	88.78	92
9	Chicken	86.55	89
10	Egg	77.53	83
11	Cheese	97.47	97
12	Meat	95.73	96
13	Onion	89.99	93
14	Beans	98.68	98
15	Fish	77.7	85
16	Banana	97.65	97
17	Green Apple	97.99	97
18	Cucumber	97.65	98
19	Lettuce	77.55	85
20	Grapes	95.7	95
21	Potato	88.56	89
22	Tangerine	97.59	99
23	Chocolate Cake	88.19	85
24	Caramel Cake	85.29	85
25	Rice	94.85	94
26	Green Pepper	97.99	98
27	Strawberry	83.47	98
28	Cooked Vegetable	92.62	96
29	Cabbage	77.55	100
30	Blueberry	83.47	95
	Total average	**92.21**	**95**

5.2.2 Applying Cloud Base SVM

A number of experiments were carried out to identify the accuracy and performance of the map reduced SVM on food classification and compares them with the SVM. We have applied these two methods on different categories of food, named single food portion and food plate. In the following sub-sections, we will explain our simulation settings and outcomes for each of these food categories.

5.2.2.1 Single food portion

First, we calculated the accuracy of our system performance on different single food portions, including various fruits and single pieces of food. For the SVM approach, we chose 100 images for the testing phase and 100 images for the training phase. The results of the LIBSVM method are shown in the third column of Table 5-3.

In another simulation, we applied the updating method by engaging the MapReduce SVM, which follows Algorithm 1 from Chapter 4, for updating purposes. The accuracy results of this method for single food portions are shown in fourth column of Table 5-3. The total average shows that we have increased the accuracy in different food portions, which are around 3% in a limited amount of food. In the following method, we will also see a huge increase in the accuracy of non-mixed and mixed food.

Table 5-3 Accuracy results of single food for LIBSVM and Map reduced SVM methods

No.	*Food items*	*Recognition Rate (%)*	
		Using All Features (10 fold cross validation)	*Using All Features (updating data-base periodically*
1	Red Apple	97.64	92.11
2	Orange	95.59	91.32
3	Corn	94.85	98.2
4	Tomato	89.56	93.82
5	Carrot	99.79	93
6	Bread	98.39	93.5
7	Pasta	94.75	98.42
8	Sauce	88.78	92.14
9	Chicken	86.55	90.12
10	Egg	77.53	92.53
11	Cheese	97.47	93.43
12	Meat	95.73	97.73
13	Onion	89.99	90
14	Beans	98.68	96.75
15	Fish	77.7	83.13
16	Banana	97.65	99.1
17	Green Apple	97.99	100
18	Cucumber	97.65	100
19	Lettuce	77.55	92
20	Grapes	95.7	97
21	Potato	88.56	95
22	Tangerine	97.59	98.58
23	Chocolate Cake	88.19	94.22
24	Caramel Cake	85.29	94.15
25	Rice	94.85	100
26	Green Pepper	97.99	98
27	Strawberry	83.47	90.48
28	Cooked Vegetable	92.62	96.5
29	Cabbage	77.55	89
30	Blueberry	83.47	92.4
Total Average		91.304	94.5

5.2.2.2 Food plate

In order to identify the accuracy of the afore-mentioned methods on food plates, we considered two different categories of food; mixed and non-mixed. Some examples are shown in Figure 5-4. In our database, we have around 3000 non-mixed and 500 mixed plates of food. For non-mixed food, we made three groups of images, containing 1000, 2000, and 3000 images,

respectively. Then, we kept 1000 images for testing purposes for each group. The system is trained with LIBSVM using half of the remaining images in each group.

The simulation results for non-mixed food are shown in Figure 5-5. As shown in this figure, the cloud SVM method outperforms the SVM method in all image categories. Furthermore, the accuracy increases as we acquire more images in the training phase. We have also evaluated SVM and cloud SVM methods on 500 mixed foods [75]. As the results in Figure 5-5 show, although the overall accuracy is lower than the non-mixed food category, we gained around 20% higher accuracy than the SVM approach.

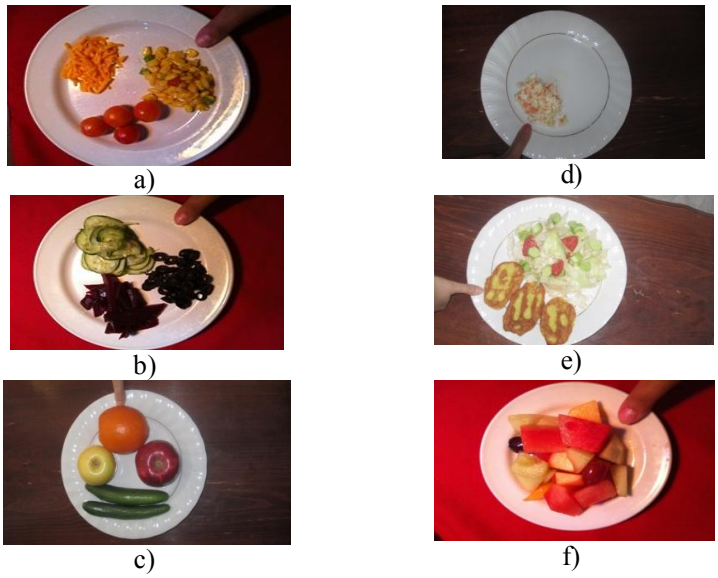


Figure 5-4 Non-mixed food (left) and mixed food (right)

The measurement of the mass of the food needs to be improved to achieve higher accuracy. This can be achieved by:

- Better estimation of the area of each food portion, which can be improved using more accurate segmentation methods.
- Coming up with an approach to measure the depth of the food more accurately, instead of assuming that the depth is uniform throughout the food portion's area, which is what we assume now.

- All of our simulations are performed on white plates with a smooth texture. We need to expand our work to various plates with different shapes, textures and colors.

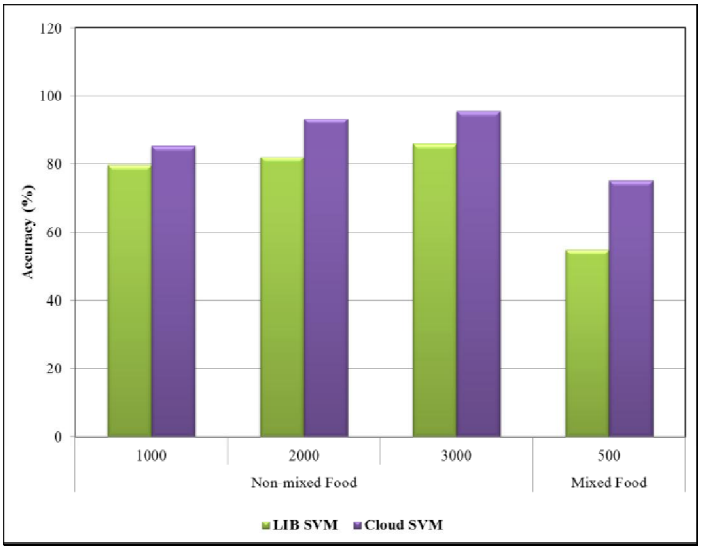


Figure 5-5 Accuracy results of Non-mixed and Mixed food plate

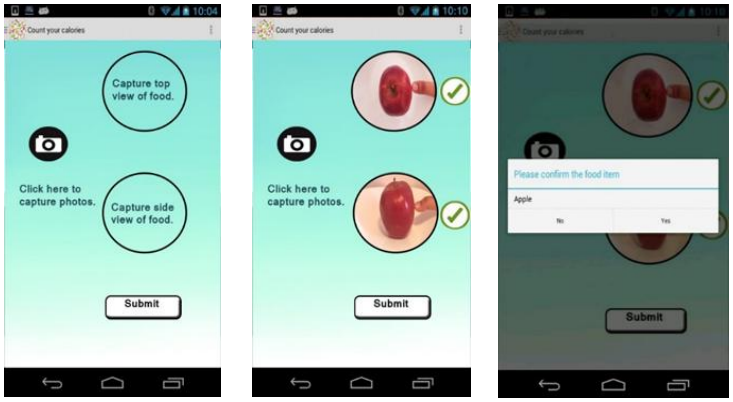
5.2.3 Applying Deep Neural Network

In this section, we combine the graph cut segmentation and deep neural network algorithms. The combination of these two methods allows us to improve the accuracy of our food classification and recognition. By recognizing the food portions with these two models and by knowing the size and shape of the food portions from the graph cut algorithm, we will have a chance to calculate the whole food portion calories.

Before the implementation of the image recognition algorithm in the Android application, the first step in our approach is to generate a pre-trained model file with the help of the CNN network. We performed this step by capturing a set of initial images of one particular class (e.g. 50 images of apple class), and then labeling them with the object name set (e.g. “apple”). These images are considered the set of relevant (positive) images. After the image sets are captured, the system is trained with these images, and then the system is re-trained with the set of negative images (images that do not contain the relevant objects). In our case, we trained the

system with the background images, so that it did not recognize them or categorize them as part of the image class. Once the model file is generated from the training, we load it into the Android application and test it against the images captured and submitted by the user.

The application works as follows: once the user clicks on the "Expected calorie intake" button in the Android application, it redirects the user to a new page. The user then captures the images twice; a top view, which enables the application to extract the food portions, and a side view, which enables the application to analyze the height of the food item in the dish (Figure 5-6-a). The thick beside the photographs indicates that the photos have been captured, and the user can now click the submit button (Figure 5-6-b). After the user clicks the submit button at the bottom of the screen, the application will initiate the image recognition algorithm, which is based on the deep neural network. The system then performs the image recognition process and generates the list of probabilities against the label name.



(a) (b) (c)

Figure 5-6 User Interface of the Android Application

The label with the highest probability is prompted to the user in a dialog box so that he or she can confirm the object name (Figure 5-6-c). The dialog box will also prompt the user to confirm the food type.

Figure 5-7 Calorie measurement

So far, our data set comprises of 30 different categories of food and fruits. As Table 5-4 and Figure 5-8 show, applying graph cut segmentation and the deep neural network algorithm produces better recognition. Our system could recognize the food portions very accurately in three seconds. The results of Table 5-4 show that we have 100% accuracy in our single food portions.

The application will select labels for food images based on the probability of occurrence. This means that for each food item, we may have three or four predicted tags. In this situation, the system will display the food item with the highest probability. Some results are shown in Figure 5-8. At the end of this process, the user will see the name and calories of the food on the monitor [74].

Table 5-4 Accuracy of Recognition in Single Food

		Recognition Rate (%)		
No.	***Food items***	***Using color-texture segmentation***	***Using graph-cut, color-texture segmentation***	***Using Deep Neural Network Method***
1	Red Apple	97.64	100	100
2	Orange	95.59	97.5	100
3	Corn	94.85	96	100
4	Tomato	89.56	95	100
5	Carrot	99.79	100	100
6	Bread	98.39	99	100
7	Pasta	94.75	98	100
8	Sauce	88.78	92	100
9	Chicken	86.55	89	100
10	Egg	77.53	83	100
11	Cheese	97.47	97	100
12	Meat	95.73	96	100
13	Onion	89.99	93	100
14	Beans	98.68	98	100
15	Fish	77.7	85	100
16	Banana	97.65	97	100
17	Green Apple	97.99	97	100
18	Cucumber	97.65	98	100
19	Lettuce	77.55	85	100
20	Grapes	95.7	95	100
21	Potato	88.56	89	100
22	Tangerine	97.59	99	100
23	Chocolate Cake	88.19	85	100
24	Caramel Cake	85.29	85	100
25	Rice	94.85	94	100
26	Green Pepper	97.99	98	100
27	Strawberry	83.47	98	100
28	Cooked Vegetable	92.62	96	100
29	Cabbage	77.55	100	100
30	Blueberry	83.47	95	100
Total average		92.21	95	100

Figure 5-8 Result of food recognition

5.3 Multiple Food Item

5.3.1 Recognition Results

In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant. Recall (also known as sensitivity) is the fraction of relevant instances that are retrieved. Both precision and recall are, therefore, based on an understanding and measure of relevance.

5.3.2 Average Recall and Precision

Recall rate measures the ability to detect food items and precision rate measures how accurate the detection is. Equation computes the recall rate (5-1), and the precision rate is computed by Equation (5-2) as follows:

$$\text{Recall} = \frac{\text{num.of correctly detected items in the category}}{\text{ground truth num.of items in the category}} \quad (5\text{-}1)$$

$$\text{Precision} = \frac{\text{num.of correctly detected items in the category}}{\text{num.of predicted items in the category}} \quad (5\text{-}2)$$

Following the proposed processing pipeline, we train multiple food detectors on the training images and test their performance on the test dataset. An average recall rate of 90.98 percent and a precision rate of 93.05 percent is obtained on the test dataset. Detailed recall rate and precision rate for each food category are listed in Table 5-5.

Table 5-5 Food Recognition Accuracy

	Recognition Rate (%)			
N	Food items	Recall	Precision	Accuracy
1	Red Apple	93.64	96	96
2	Orange	95.59	97.5	98
3	Corn	84.85	80	85
4	Tomato	89.56	97	96
5	Carrot	93.25	98	98
6	Bread	98.39	89	90
7	Pasta	94.75	98	98
8	Sauce	88.78	92	93
9	Chicken	86.55	89	88
10	Egg	81.22	87	90
11	Cheese	95.12	97	97
12	Meat	95.73	96	96.5
13	Onion	89.99	93	95
14	Beans	98.68	95	97
15	Fish	77.7	85	88
16	Banana	97.65	97	97
17	Green Apple	97.99	97	97
18	Cucumber	97.65	98	98
19	Lettuce	77.55	85	88
20	Grapes	95.7	95	95
21	Potato	88.56	89	91
22	Tangerine	97.59	99	99
23	Chocolate Cake	88.19	85	90
24	Caramel Cake	85.29	85	88
25	Rice	94.85	94	94
26	Green Pepper	97.99	98	98
27	Strawberry	85	95	97
28	Cooked Vegetable	92.62	96	96
29	Cabbage	80	91	92
30	Blueberry	89	98	98
Total average		90.98	93.05	94.11

Also the comparison of precision, recall and accuracy of the model is shown in Figure 5-9. As the graph shows the accuracy of the system in each category is higher than two other methods.

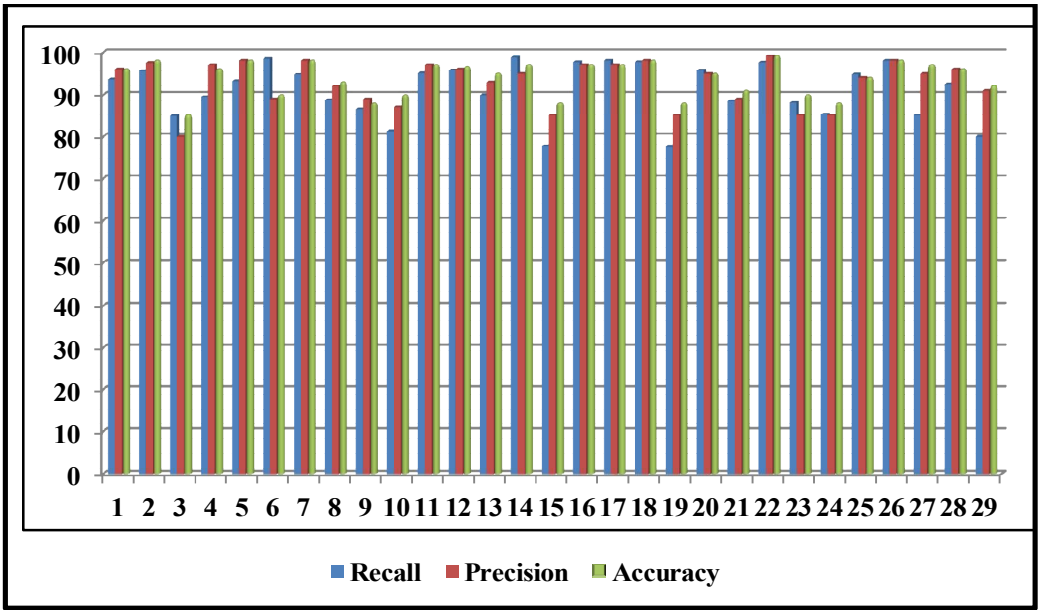


Figure 5-9 Some successful recognition by our proposed method.

Our experiments show that the proposed method described in section 4.2 works well in recognizing and predicting the food items in the image. Also, we have achieved a higher accuracy rate (94.11%) compared to the 50.8–88% accuracy rate of related systems that currently exist. Region mining is an important step in object detection; it discovers discriminative regions that help distinguish each type of food from the others. Regions that belong to the background or would possible confuse the classifier were discarded.

Region mining is an important step in object detection. It discovers discriminative regions which help distinguish each type of food from the others. Regions that belong to the background or would possibly confuse the classifier are discarded.

Region mining improves the accuracy of classifiers for each food category because the classifiers are trained on carefully selected positive samples. On the other hand, when training classifiers by using the whole image as the positive region without region mining, the precision accuracy rate is very low. This is because the individual classifier for each category trained on the whole image is not accurate. Thus, there is much false positive detection at the testing phase.

Mining the hard negative regions is another important step in our proposed method. This processing step solves the problem of imbalanced positive and negative samples at training stage. Thus, it helps improve the recall rate of our detection. classifiers trained with the data most of which are negative examples will predict “NO” most of the times, resulting in very low recall rate. The step of mining the hard negatives only selects the hardest samples from the negative sample set and thus can balance the positive/negative samples in the training set.

5.3.3 Comparison Experiment

To evaluate the significance of region mining procedure in object detection, we conduct a comparison experiment which uses the whole image region to train multiple food detectors without any region mining involved during training.

An average recall rate of 90.98% and precision rate of 93.05% is obtained by the comparison method. Compared to the results obtained with region mining, it has a lower precision rate. The classifiers trained without region mining produce low precision rate because many false predictions are computed.

5.3.4 Response Time

Mostly, it depends on the image whether it is single or multiple foods.

By proposing a dynamic cloud allocation mechanism [86] to handle parallel image processing requests from different users, the system is able to automatically add the cloud instance in real-time by gauging the system resource which are:

- Number of available cloud instances
- The total number of incoming parallel food processing requests
- By reallocating the cloud instances when not in use

The overall respond time will be less than 70 seconds. so, in order to ensure efficient utilization of cloud resources and at the same time ensure the overall response time of the processing the image remains less than 70 seconds, the cloud nodes had to be strategically increased and even decreased when not in use. It is shown in Figure 5-10 and Figure 5-11.

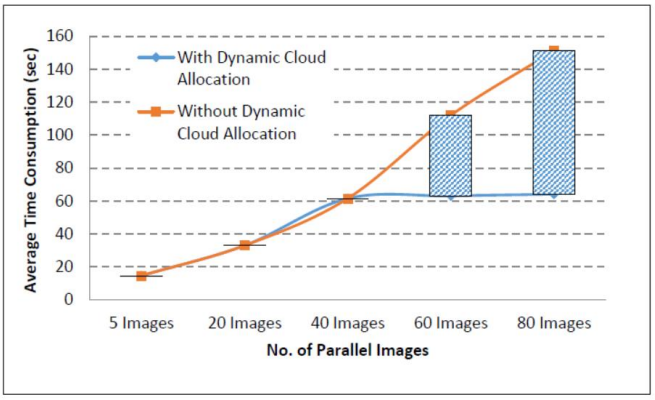


Figure 5-10 parallel Images average time consumption

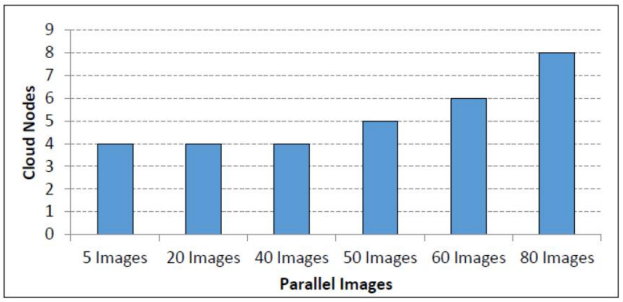


Figure 5-11comparison between the cloud nodes and parallel image request (Avg. time<70)

5.3.5 Difficulties

The biggest problem that the application may encounter is having too much variance in the food appearance. For example, the color, size and shape of cookies vary too much. In some images, cookies appear to have a round shape and a dark brown color, while others appear to have a square shape and a black chocolate color. The region mining procedure in our proposed pipeline is based on the assumption that object regions of images belonging to the same class will cluster closely together in feature space. If the appearance varies too much, it is difficult to obtain meaningful positive regions from region mining.

Figure 5-12 shows some successful recognition by our proposed method. Although food items vary a lot in appearance, size and occlusion, our proposed method correctly recognizes and localizes most of the samples. However, our proposed algorithm also generates inaccurate detection on some of the test images.

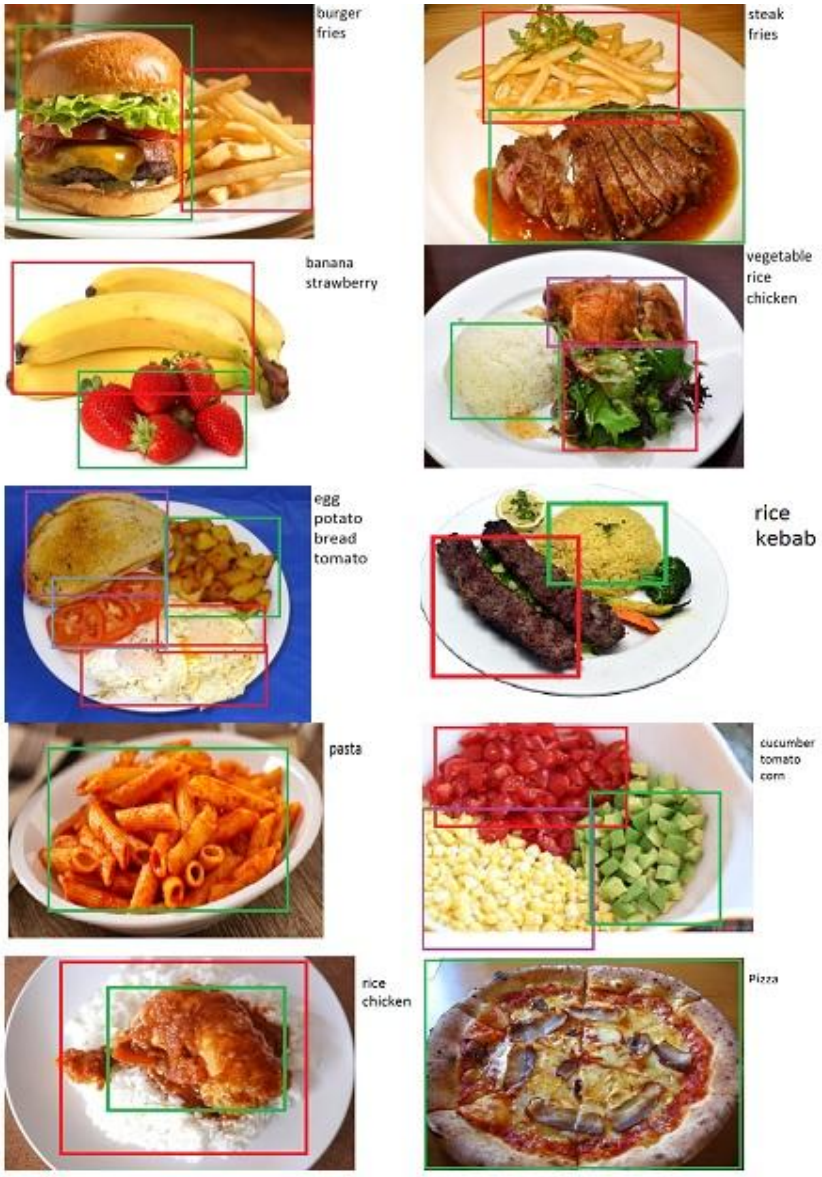


Figure 5-12 Detection Results

Chapter 6 Conclusion

In this thesis, we proposed a deep learning method for detecting single and multiple food items from food pictures taken with mobile devices. Our experiments were conducted with a dataset of 30 food categories, in which images containing only a single food item were used as the training set, and images containing single, multiple, and mixed food items were used as the test set. Our proposed method learns from single food item images using various techniques, including CNN features, region mining, and classifier training. We demonstrated that our system uses a combination of these methods to provide a powerful instrument for attaining a high level of food recognition accuracy.

We employed a rather unique combination of graph cut segmentation and deep learning neural networks as a means of accurately classifying and recognizing food items. We showed that the combination of those two methods can achieve 100% food recognition accuracy in our system. We implemented a virtualization approach in the application, which allows us to benefit from cloud-based resources. Also, by applying the region mining method on multiple food items, we achieved an average recall rate of 90.98%, precision rate of 93.05%, and accuracy rate of 94.11%.

Chapter 7 Future Work

In our research thus far, deep neural networks have been effectively applied to a single and multiple feature. Our plan in the future is

- Propose a mobile application and ask users to test it and apply their comments.
- Propose an application for single and multi-objects in videos.

7.1 Deep Learning on Video

From a practical view, videos are significantly more difficult to collect and store compared to images. Nonetheless, the issues that motivate us to go through this methodology include:

- From a user's perspective, it would be boring and time consuming to take pictures of different food items one by one. Instead, the user can capture a video of everything he or she wants to eat or drink and see the number of calories in the whole meal.
- For mixed foods or in conditions when some ingredients are invisible or absorbed by other items after the cooking process (e.g. butter or oil), capturing a video would lead to a more accurate result. In other words, when the user wants to prepare food, he/she can take a video of all ingredients and identify the total number of calories instead of finding the number of calories after cooking. In this way, the system can achieve more accurate calorie results.

Therefore, we are collecting our datasets of food to prepare a dataset for our algorithm and prepare the test case for application. Different works that have been conducted regarding video

classification estimate the standard approach to video classification involves three major stages: first, local visual features that describe a region of the video must be extracted. The second stage involves the recognition of natural human actions in diverse and realistic video settings, including recognition of the video and the sharing of common problems with object recognition in images. Both tasks must deal with significant intra-class variations and background muddle [84]. Finally, a classifier such as SVM is trained on the words of the images to extract among the visual classes of interest.

CNNs [85]are a biologically inspired class of deep learning models that replace all three stages with a single neural network that is trained from raw pixel values to classifier outputs.

From a modeling viewpoint, we are interested in applying the CNN method to the video and trying to recognize the object. As thus far we have done much work on food recognition, we are going to apply the method of video capturing food to extract each food from different frames.

From a computational viewpoint, CNNs require long periods of training to optimize effectively the millions of parameters. This difficulty happens when extending the connectivity of the architecture in time, because the network must process not just one image but several frames of video at a time.

We must consider these parameters in video object recognition:

7.1.1 Time Information Fusion in CNNs

We must investigate several approaches to fusing information. We can first consider a baseline single frame CNN and then discuss its extensions in time according to different types of fusion.

7.1.2 Learning

We must take advantage of preprocessing to reduce the effect of over fitting and we must apply it to all frames of the same clips.

References

[1] M. Bosch, I. Woo, S. Kim, C. Boushey, D. Ebert, and E. Delp F. Zhu, "The use of mobile devices in aiding dietary assessment and evaluation," IEEE Journal of Selected Topics in Signal Processing, vol. 4, no. 4, pp. 756-766, August 2010.

[2] M. Carrol, L. Curtin, M. Lamb, and K. Flegal C. Ogden, "Prevalence of high body mass index in us children and adolescents" Journal of the American Medical Association, vol. 303, no. 3, pp. 242-249, January 2010.

[3] Yu Deng, Shiyin Qin, and Yunjie Wu, "An Automatic Food Recognition Algorithm with both, Image Processing and Photonics for Agricultural Engineering," vol. 7489 , 2009.

[4] Parisa Pouladzadeh, Shervin Shirmohammadi, Abdulsalam Yassine. "You are what you eat: So measure what you eat". IEEE Instrum. Meas. Mag. 19(1):9-15, 2015.

[5] Y. C. Probst and L. C. Tapsell, "Overview of computerized computerized dietary assessment programs for research and practice in nutrition education," J. Nutr. Educ. Behav, pp. 20–26, 2005.

[6] D. H. Wang, D. H. Kogashiwa, and S. Kira, "Development of a new instrument for evaluating individuals' dietary intakes," J. Am. Diet. Assoc. 106, pp. 1588–1593, 2006.

[7] L. Harnack, L. Steffen, D. Arnett, S. Gao, and R. Luepker, "Accuracy of estimation of large food portions," J.Am.Diet.Assoc, vol. 104, pp. 804–806, 2004.

[8] R. Johnson, R. Soultanakis, and D. Matthews, "Literacy and body fatness are associated with underreporting of energy intake in US low-income women using the multiple-pass 24-hour recall: a doubly labeled water study," J.Am.Diet.Assoc, vol. 98, pp. 1136–1140, 1998.

[9] Pei-Yu Chi, Jen-Hao Chen, Hao-Hua Chu, & Jin-Ling Lo, "Enabling Calorie-Aware Cooking in a Smart Kitchen," 3rd International Conference on Persuasive Technology, 2008.

[10] J. Nishimura and T. Kuroda, "Human Action Recognition Using Wireless Wearable In-Ear Microphone ," IEEJ, Vol.131, No.9, Sec.C, pp.1570-1576, 2011.

[11] K. Chang et al., "The diet-aware dining table: Observing dietary behaviors over a tabletop surface," Proceedings of Pervasive Computing , 4th International Conference, pp. 366–382, May 2006.

[12] http://mealsnap.com

[13] http://www.carbsandcals.com/apps

[14] http://www.atkins.com/Free-Tools/Mobile-App.aspx

[15] http://tracker.dailyburn.com/foodscanner

[16] Jie Yang Wen Wu, "Fast Food Recognition From Videos of Eating for Calorie Estimation," Intl. Conf. on Multimedia and Expo, IEEE, pp. 1210–3, 2009 .

[17] Jieun Kim, Mireille Boutin, “Estimating the Nutrient Content of Commercial Foods from their Label Using Numerical Optimization”, in New Trends in Image Analysis and Processing - ICIAP Workshops, pp 309-316, 2015.

[18] L. Young and M. Nestle, "The contribution of expanding portion sizes to the us obesity epidemic," American Journal of Public Health, vol. 92, pp. 246-249, 2002.

[19] R. Patterson, A. Kristal, and C. Cheney S. Rebro, "The effect of keeping food records on eating patterns," Journal of the American Dietetic Association, vol. 98, pp. 1163-1165, 1998.

[20] Martin C. K., Kaya S. and Gunturk B. K, “Quantification of food intake using food image analysis”, International Conference of IEEE Engineering in Medicine and Biology Society, pp. 6869-6872, 2009.

[21] Takeda, F., et al., “Dish extraction method with neural network for food intake measuring system on medical use”, Computational Intelligence for Measurement Systems and Applications, July 29-31, pp. 56-59 , 2003.

[22] Fengqing Zhu, Marc Bosch, Nitin Khanna, Carol J. Boushey, “Multiple Hypotheses Image Segmentation and Classification With Application to Dietary Assessment”, IEEE Journal of Biomedical and Health Informatics, Vol. 19, NO. 1,pp. 377- 389, January 2015.

[23] http://jaybaxter.net/6869_food_project.pdf

[24] Yoshiyuki Kawano, Keiji Yanai, “ FoodCam: A Real-Time Mobile Food Recognition System Employing Fisher Vector ”, Springer International Publishing Switzerland, pp. 369–373, 2014.

[25] Yu Wang, Ye He, Fengqing Zhu, Carol Boushey and Edward Delp, “The Use of Temporal Information in Food Image Analysis”, in New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Lecture Notes in Computer Science, Springer, Volume 9281, 2015, ISBN: 978-3-319-23221-8, pp 317-325.

[26] Yucheng, et al. Low, "Distributed GraphLab: A framework for machine learning and data mining in the cloud," Proceedings of the VLDB Endowment 5.8 , pp. 716-727, 2012.

[27] Grzegorz, et al. Malewicz, "Pregel: a system for large-scale graph processing," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010.

[28] A. Talwalkar, J.Duchi, R. Griffith, M. Franklin, M.I. Jordan T. Kraska, "MLbase: A Distributed Machine Learning System," in Conference on Innovative Data Systems Research, 2013.

[29] Alessandro MazzeiAffiliated , Luca Anselma, Franco De Michieli, Andrea Bolioli, Matteo Casu, Jelle Gerbrandy, Ivan Lunardi, “Mobile Computing and Artificial Intelligence for Diet Management”, in New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, V. Murino,

E. Puppo, D. Sona, M. Cristani, and C. Sansone, Lecture Notes in Computer Science, Springer, Volume 9281, 2015, ISBN: 978-3-319-23221-8, pp 342-349.

[30] http://dx.doi.org/10.1145/2647868.2654970.

[31] Hokuto Kagaya , Kiyoharu Aizawa, “Highly Accurate Food/Non-Food Image Classification Based on a Deep Convolutional Neural Network” , in New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Lecture Notes in Computer Science, Springer, Volume 9281, 2015, ISBN: 978-3-319-23221-8, pp 350-357.

[32] http://link.springer.com/article/10.1007/s10586-015-0468-2

[33] P. Pouladzadeh, A. Yassine, and S. Shirmohammadi, “FooDD: Food Detection Dataset for Calorie Measurement Using Food Images”, in New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Lecture Notes in Computer Science, Springer, Volume 9281, 2015, ISBN: 978-3-319-23221-8, pp 441-448

[34] Stergios Christodoulidis, Marios Anthimopoulos, Stavroula Mougiakakou, “Food Recognition for Dietary Assessment Using Deep Convolutional Neural Networks”, in New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone, Lecture Notes in Computer Science, Springer, Volume 9281, 2015, ISBN: 978-3-319-23221-8, pp 458-465.

[35] Yuji Matsuda, Hajime Hoashi, and Keiji Yanai. 2012. Recognition of multiple-food images by detecting candidate regions, IEEE International Conference on Multimedia and Expo (ICME), 25–30.

[36] Yoshihiro Kawano and Katsuki Yanai. 2013. Real-time mobile food recognition system. In IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 1–7.

[37] Weiyu Zhang, Qian Yu, Behjat Siddiquie, Ajay Divakaran, and Harpreet Sawhney, Food recognition and nutrition estimation on a smartphone. Journal of diabetes science and technology, 9(3),pp. 525–533, 2015.

[38] Xi-Jin Zhang, Yi-Fan Lu, Song-Hai Zhang, Multi-Task Learning for Food Identification and Analysis with Deep Convolutional Neural Networks, Journal of Computer Science and Technology, Volume 31, Issue 3, pp. 489–500, 2016.

[39] Morteza Akbari Fard, Hamed Haddadi, Alireza Tavakoli Targhi, Fruits and Vegetables Calorie Counter Using, Convolutional Neural Networks, In ACM Digital Health, pp. 121-122, 2016.

[40] Ashutosh Singla, Lin Yuan, Touradj Ebrahimi, Food/Non-food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model, In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management,p.p. 3-11, 2016.

[41] Wataru Shimoda Keiji Yanai, Foodness Proposal for Multiple Food Detection by Training of Single Food Images, In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management, pp. 13-21, 2016.

[42] Joachim Dehais, Marios Anthimopoulos, Stavroula Mougiakakou, Food Image Segmentation for Dietary Assessment, In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management, pp. 23-28, 2016.

[43] Ruihan Xu, Luis Herranz, Shuqiang Jiang, Geolocalized Modeling for Dish Recognition, IEEE Transactions on Multimedia, Vol. 17, NO. 8, August 2015.

[44] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. International Journal of Computer Vision, 104(2):154–171, 2013.

[45] Alex Krizhevsky, Ilya Sutskever, and Geo_rey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[46] http://healbe.com/

[47] http://www.tellspecopedia.com

[48] http://www.consumerphysics.com/myscio

[49] A. Jain and G. Healey, "A multiscale representation including opponent color features for texture recognition," IEEE Transactions on Image Processing, vol. 7, no. 1, pp. 124-128, 1998.

[50] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters."Communications of the ACM 51.1, pp.107-113,2008.

[51] P. Kruizinga, N. Petkov, and S. E. Grigorescu, "Comparison of texture features," in Proceedings of the 10th International Conference on Image Analysis and Processing, Washington DC, USA, September 1999, pp. 142–148.

[52] S. E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on Gabor filters," IEEE Trans. on Image Processing, vol. 11 (10), pp. 1160–1167, 2002.

[53] J. Zhang, T. Tan, and L. Ma, "Invariant texture segmentation via circular Gabor filters," Proceedings of the 16th IAPR International Conference on Pattern Recognition (ICPR), vol. II, p. 901—904, 2002.

[54] Y. B. Yuri, G. F. Lea, “Graph Cuts and Efficient N-D Image Segmentation,” International Journal of Computer Vision, vol.70, no.2, pp.109-131, 2006.

[55] Y. Boykov, V. Kolmogorov, “An experimental comparison of mincut/max-flow algorrithms for energy minimization in vision,” IEEE transaction PAMI, vol.26, no.9. pp. 1124-1137,2004.

[56] Krizhevsky, A., Sutskever, I., and Hinton, G. on “ImageNet classification with deep convolutional neural networks.” in NIPS, 2012.

[57] Zeiler, M.D., Ranzato, M., Monga, R., Mao, M., Yang,K., Le, Q.V., Nguyen, P., Senior, A., Vanhoucke V., Dean J. and Hinton, G.E., “OnRectied Linear Units for Speech Processing”, in ICASSP , 2013.

[58] http://neuralnetworksanddeeplearning.com/chap2.html

[59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradientbased learning applied to document recognition”, Proceedings of the IEEE, 86(11):2278–2324, 1998.

[60] F. Ozgur Catak1and M. Erdal Balaban, “CloudSVM: Training an SVM Classifier in Cloud Computing Systems”, Proceedings of the 2012 international conference on Pervasive Computing and the Networked World, pp.57-68, Springer-Verlag Berlin, 2013.

[61] http://en.wikipedia.org/wiki/MapReduce.

[62] Jan Hosang, Rodrigo Benenson, Piotr Doll´ar, and Bernt Schiele. What makes for effective detection proposals? arXiv preprint arXiv:1502.05082, 2015.

[63] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 73–80. IEEE, 2010.

[64] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34(11):2189–2202, 2012.

[65] P. Kruizinga, N. Petkov, and S. E. Grigorescu, "Comparison of texture features," in Proceedings of the 10th International Conference on Image Analysis and Processing, Washington DC, USA, September 1999, pp. 142–148.

[66] C. C. Chang and C. J. Lin, "LIBSVM : a library for support vector machines," ACM Transactions on Intelligent Systems and Technology 2:27:1–27:27, 2011.

[67] T. Mitchell, Machine Learning, McGraw-Hill Computer Science Series., 1997.

[68] M.Victor Jose S.Sathya, "Application of Hadoop MapReduce Technique to Virtual Database System Design," Processing of ICETECT, vol. 11, no. 9, pp. 892-896, 2011.

[69] http://media.amazonwebservices.com/AWS_Amazon_EMR_Best_Practices.pdf

[70] https://www.jetpac.com/

[71] G. Villalobos, R. Almaghrabi, P. Pouladzadeh, and S. Shirmohammadi, “An Image Processing Approach for Calorie Intake Measurement,” in Proc. IEEE Symposium on Medical Measurement and Applications, pp. 1-5, 2012.

[72] P. Pouladzadeh, G. Villalobos, R. Almaghrabi, and S. Shirmohammadi, “A Novel SVM Based Food Recognition Method for Calorie Measurement Applications,” in Proc. International Workshop on Interactive Ambient Intelligence Multimedia Environments, in Proc. IEEE International Conference on Multimedia and Expo, pp. 495 – 498, 2012.

[73] R. Almaghrabi, G. Villalobos, P. Pouladzadeh, and S. Shirmohammadi, “A Novel Method for Measuring Nutrition Intake Based on Food Image,” in Proc. IEEE International Instrumentation and Measurement Technology Conference, pp. 366 – 370, 2012.

[74] Parisa Pouladzadeh, Shervin Shirmohammadi, and Rana Almaghrabi, “Measuring Calorie and Nutrition from Food Image”, IEEE Transactions on Instrumentation & Measurement, Vol.63, No.8, p.p. 1947 – 1956, August 2014.

[75] P. Pouladzadeh, S. Shirmohammadi, A. Bakirov, and A. Bulut, “Cloud-Based SVM for Food Categorization”, Multimedia Tools and Applications, Springer, Published online, June 2014.

[76] Satoru Fujishige. Submodular functions and optimization, volume 58. Elsevier, 2005.

[77] Laurence AWolsey, "An analysis of the greedy algorithm for the submodular set covering problem". Combinatorica, 2(4):385–393, 1982.

[78] Krizhevsky, A., Sutskever, I., and Hinton, G. on “ImageNet classification with deep convolutional neural networks.” in NIPS, 2012.

[79] Parisa Pouladzadeh, “An Image Processing and Pattern Analysis Approach for Food Recognition”, Master’s thesis, University Of Ottawa, 2012.

[80] F. Ozgur Catak1and M. Erdal Balaban, “CloudSVM: Training an SVM Classifier in Cloud Computing Systems”, Proceeding in ICPCA/SWS'12 Proceedings of the 2012 international conference on Pervasive Computing and the Networked World, pp.57-68, Springer-Verlag Berlin, 2013.

[81] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradientbased learning applied to document recognition”, Proceedings of the IEEE, 86(11):2278–2324, 1998.

[82] Parisa Pouladzadeh, Shervin Shirmohammadi, and Rana Almaghrabi, “Measuring Calorie and Nutrition from Food Image”, IEEE Transactions on Instrumentation & Measurement, Vol.63, No.8, p.p. 1947 – 1956, August 2014.

[83] P.Pouladzadeh, S.Shirmohammadi, A.Yassine, “Using Graph Cut Segmentation for Food Calorie Measurement”, IEEE International Symposium on Medical Measurements and applications, p.p.1-6, Lisbon, June 2014.

[84] H.Wang, M. M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al., “Evaluation of local spatio-temporal features for action recognition”, In BMVC, 2009.

[85] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradientbased learning applied to document recognition”, Proceedings of the IEEE, 86(11):2278–2324, 1998.

[86] Sri Vijay Bharat Peddi, Pallavi Kuhad, Abdulsalam Yassine, Parisa Pouladzadeh , Shervin Shirmohammadi, Ali Asghar Nazari Shireh, "An Intelligent Cloud-Based Data Processing Broker

for Mobile e-Health Multimedia Applications" Future Generation Computer Systems, Elsevier, p.p .71–86, 2016.