A Deep Learning and Auto-Calibration Approach for Food Recognition and Calorie Estimation in Mobile e-Health

By

Pallavi Kuhad

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the degree of Masters of Applied Science in

Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

University of Ottawa

Abstract

High calorie intake has proved harmful worldwide, as it has led to many diseases. However, dieticians have deemed that a standard intake of number of calories is essential to maintain the right balance of calorie content in human body. In this thesis, we consider the category of tools that use image processing to recognize single and multiple mixed-food objects, namely Deep Learning and the Support Vector Machine (SVM). We propose a method for the fully automatic and user-friendly calibration of the sizes of food portions. This calibration is required to estimate the total number of calories in food portions. In this work, to compute the number of calories in the food object, we go beyond the finger-based calorie calibration method that has been used in the past, by automatically measuring the distance between the user and the food object. We implement a block resize method that uses the measured distance values along with the recognized food object name to further estimate calories. While measuring distance, the system also assists the user in real time to capture an image that enables the quick and accurate calculation of the number of calories in the food object. The experimental results showed that our method, which uses deep learning to analyze food objects, led to an improvement of 16.58% in terms of recognition, over the SVM-based method. Moreover, the block resize method showed that percentage error for calorie estimation was reduced to 3.64% as compared to 5% achieved in previous methods.

Acknowledgements

I am grateful to Prof. Shervin Shirmohammadi for giving me the opportunity to work on this exciting project. In particular, I would like to thank him for allowing me to be part of this great team. The professor has always provided prompt guidance on any issue, and he has always helped me by providing immediate feedback on my work. He has always accepted novel ideas and motivated me to think analytically. He has provided me with many opportunities to write papers for various conferences and scholarly journals. The professor's appraisal of our work has been critical to the acceptance of our papers for publication. I have always enjoyed working under him, and I hope to work with him in the future.

I am also very grateful to Dr. Abdulsalam Yassine for his guidance throughout this project. He has always encouraged me to explore new possibilities in our project. His great sense of humor has kept our project meetings very positive and full of energy. He has always been available to discuss my queries and has provided immediate suggestions for my work. I really enjoyed working with him.

In addition, Parisa Pouladzadeh and Sri Vijay have always been very helpful in sharing the workload, which has helped us work cohesively as a strong team. Parisa has been very supportive, especially in introducing me to various concepts when I was new to this project.

Finally, I would like to thank my Dad, Mom, and brother (Piyush) for trusting my decision to do Master's degree and supporting me throughout my life. My family has been my strength in both difficult and good times. My parents have been my friends and have always motivated me to work hard. I hope my work will make them proud. I also thank Vijay for being with me for the last nine years and for his love and support.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

There has been an unprecedented rise worldwide in overweightness and obesity, as well as in diseases they cause [1][2][3][4]. Thus, monitoring daily eating routines in order to avoid excess calorie intake has become an important issue in maintaining a good quality of life. Studies have shown that several diseases are linked to excessive calorie intake in humans. In [1], Pan found that breast, colon, and prostate cancers are caused by high calorie intake. High calorie intake was found to be only second to tobacco in directly causing cancer [2]. Moreover, a previous study found that a proper diet involving lower calorie intake helped the residents of Okinawa to increase their life expectancy and lower the risk of age-associated diseases [3]. High calorie values in food that is nutritionally poor led to systemic inflammation and reduced insulin sensitivity, as well as a cluster of metabolic abnormalities, including obesity, hypertension, dyslipidemia, and glucose intolerance [4]. In [5], a review of the literature showed that reducing the calorie intake lessened the risk of cancer in humans.

To assist people in tracking their calorie intake, efficient and convenient mobile applications have been developed that alert users to the number of calories they consume. Such mobile applications are becoming increasingly popular. Mobile applications have the capacity to provide the easy collection of personal health-related data and timely behavioral cues [6][7]. Additionally, research has focused on the benefits of mobile and

Internet technologies in reaching diverse populations, such as rural communities, in order to reduce health disparities [8][9] and promote health interventions [10].

Because mobile phone ownership and the number and complexity of health applications are likely to increase, the potential for technology-based health interventions to affect populations is expanding in ways that previously were not possible [11]. Based on study of health and fitness mobile applications [12], we concluded that over 45% of users of health and fitness applications use the Android platform, and 33% use iOS. Health and fitness mobile applications are broadly classified into three major categories: fitness and training applications, calorie counter applications that count the number of calories burnt during any physical activity, and healthy eating applications. Most of these applications focus on assisting users in monitoring their calorie consumption during not only workout sessions but also any physical activity.

Mobile health applications provide several technological solutions that also measure vital signs, such as heart rate, blood glucose level, blood pressure, body temperature, and brain activities. Prominent examples of applications are communication, information, and motivation tools, such as medication reminders, or tools that recommend ways to improve fitness and diet [13].

Despite the availability of mobile applications and fitness trackers that help the user by keeping track of the number of calories burned, there is still the need for an intelligent system that assists the user in keeping track of calories and reduces the risk for diseases caused by the excessive intake of calories.

1.2 Problem Statement

In this thesis, we focused on multiple aspects that contribute to building a mobile e-health application that would provide tools to monitor calorie intake and assist users to ensure a healthy diet. The system is able to recognize the food object on the user's plate and determine its calorie content. We will further discuss some of the problems with the existing approaches below.

The patient's dietary information helps doctors gain the information they need to diagnose problems, such as obesity- and weight-related issues. In the existing approach, patients report their daily diet to a dietician. The problem with this approach is that it provides an inaccurate measure of the sizes of food portions, which neglects the fundamental reason for inconsistencies in calorie calculation while maintaining dietary information.

One challenge of calorie computation is the ability to gauge the quantity of the real object in a two-dimensional image. Existing approaches require a reference object, such as a finger, coin, or scale, to calibrate the food dimensions. The problem with this approach is that it is cumbersome for the user to take a picture using a mobile phone while placing the reference object. If the dimensions of the reference object were recorded incorrectly, it would result in an inaccurate measurement of the dimensions of the food portion, thus further affecting the overall calorie value.

Another problem in computing calorie intake is that the quantity of the food object, which contributes to its overall calorie content, changes according to the position of the mobile device with respect to the food object. The user's distance from the food object

also directly affects the size ratio of the food object. In other words, the farther the user stands from the food object, the smaller it appears in the captured image, and the lower number of calories that are computed. However, distance measurement methods, which are critical in determining the dimensions of the food object, have not yet been proposed, and the existing approaches do not address this challenge in computing calories.

Furthermore, existing approaches compute the dimensions of the food portions by splitting them into blocks and then calculating the area of the individual blocks. However, in this approach, the problem is that the block size remains the same, irrespective of the distance from which the images are captured. This leads to the inaccurate measurement of the food portion size and to a higher ration of errors in the number of calories consumed. Hence, there is the need for an approach that uses the distance measurement method to determine the units of the blocks.

Currently, many computer vision algorithms are available for image recognition and face recognition, which are used to determine the class of the object by analyzing a two-dimensional image. However, the results of studies on deep learning have proved to be significantly more accurate in classifying these images. Therefore, an approach that uses deep learning methodology to recognize food images is required because they have unique characteristics distinguishes them from other classes of generic images.

The existing mobile e-health applications require frequent feedback and suggestions by the user for the system to classify accurately the food object and compute the calorie intake. In most applications, either the processing is too lengthy or frequent input from the user is required, such as the ingredients, the food category, or the unit

measurements required. Applications that require frequent user inputs are likely to make the user lose interest, thus undermining the entire user experience.

Another problem associated with calorie computation is that the existing approaches follow the same methodology to address all food objects, which leads to the inaccurate calculations of the number of calories. Each food class has unique characteristics that need to be considered when the total number of calories is calculated.

Another challenge in current applications for food recognition and calorie computation is distinguishing multiple food objects or mixtures of food objects on the plate. In order to differentiate the total number of food objects in the plate, each food object is treated separately. In addition, all calories are summed and displayed to the user because the system was designed to work automatically. Current approaches do not address this concern in computing the number of calories.

This thesis will address these issues and elaborate the approaches that we adopted to achieve the computation of the calorie values of food objects through the smartphone. We will compare them based on accuracy and processing time, which will enable us to understand the principles that underlie the methodologies and how they contribute to the overall results.

1.3 Thesis Objectives

To address the above research challenges, we proposed methodologies of calorie estimation and food object recognition that will enable the system to determine accurately the calorie content. To achieve this goal, we built a mobile e-health application

that enables the user to capture a photo of the target food object in order for the system to determine the calorie content.

An automated mobile application will not only assist the doctor with the required dietary information about the patient but also empower the patient to keep track of calorie intake while he or she is consuming the food object. This will remove the daily dependency of the patient on the dietician and provide the latter with information on a monthly or weekly basis. Hence, the patient is able to control his or her diet by not overindulging in food.

By using deep learning, this system will be able to recognize accurately the single food objects on a white plate. With regard to multiple food objects, the system will be able to segment them when they are placed separately on the plate. Several problems are associated with mixed food objects because many dissolvable contents (e.g., sugar, salt, oil, etc.) are used in the preparation of the mixed food dish. However, the content of single food objects (e.g., apple, banana, grapes, tomato, etc.) is homogeneous. Hence, the proposed system is limited because ingredients that are not visible and that are mixed in the food objects will not be recognized. The results demonstrate that most the food objects are independent of such variance because of dissolvable substances except food objects, such as caramel cake, chocolate cake and chicken. Although we are able to measure the calorie content of single food objects, mixed food still presents a challenge. Nevertheless, deep learning has helped in achieving higher accuracy in measuring mixed food objects, which is a considerable improvement over previous approaches. However,

the negligible difference in the calorie estimation of the dissolvable content used in the preparation of mixed food is a remaining challenge.

The current approach is able to recognize the food object and determine the calorie content based on distance measurement and the block resize method. Although our system is still unable to detect mixed content, in future work, we will focus on detecting each ingredients with respect to the type of food object.

The proposed system for determining calorie content is designed to run in a fully automated manner according to the prescribed steps of distance measurement, deep learning, and calorie computation. Hence, no human intervention is required in the classification stage, calorie estimation, or administration. Only in cases when the food object is not recognized by the system is the user prompted to specify the food class.

In the recognition of food objects, the current system follows the same model file that was trained to detect a set of food classes. However, in calorie computation, the steps of block resizing and distance measurement remain the same; the reference values of the blocks change based on the type of food object. This has enabled us to achieve higher accuracy in calorie computation.

1.4 Contribution of Thesis

- We proposed an automated mobile e-health application for the recognition of food objects by implementing deep learning, specifically for food-based images.
- We also proposed calibration methods, such as distance measurement for estimating the sizes of food portions and determining the number of calories they

contain. Our distance measurement method gauges the exact distance between the food object and the user.

- This application provides real-time feedback during the distance measurement, and the system suggests the ideal position to capture a photo of the food on the plate. This capability enables the user to have an accurate image, which means less delay in computing the number of calories. The system accurately determines the size of the food portion and calculates the calorie content.
- Using the new block resize method, the system uses the values obtained from the distance measurement method and resizes the block dimensions accordingly, further assisting it to calculate the total area and the perimeter of the food portions.

1.5 Research Publications

- Parisa Pouladzadeh, Pallavi Kuhad, Sri Vijay Bharat Peddi, Abdulsalam Yassine, and Shervin Shirmohammadi, "Mobile Cloud Based Food Calorie Measurement" paper presented to the 4th International IEEE Workshop on Multimedia Services and Technologies for E-health (MUST-EH 2014), July 14, 2014 Chengdu, China, in conjunction with IEEE ICME 2014.
- Parisa Pouladzadeh, Sri Vijay Bharat Peddi, Pallavi Kuhad, Shervin Shirmohammadi, "A Map Reduce Parallel Classifier for Cloud Based Food Recognition" in *International Conference on Next Generation Computing and Communication Technologies* [ICNGCCT]-2014, Dubai. pp. 142–147.

- Pallavi Kuhad, Abdulsalam Yassine and Shervin Shirmohammadi, "Using Distance Estimation and Deep Learning to Simplify Calibration in Food Calorie Measurement" in *Computational Intelligence and Virtual Environments for Measurement Systems and Applications* [CIVEMSA]-2015, Shenzhen, China.

Journal publications

- 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," *in Cluster Computing Journal 2015*.

1.6 Thesis Organization

This thesis is organized as follows. In Chapter 2, we discuss the background and related work focusing on the computer vision algorithms used in various healthcare-related applications. We also discuss related work in the area of mobile healthcare applications, with specific regard to health and fitness applications and their development over time. We then discuss in detail the basic architecture of the computer vision algorithm implemented for the food recognition purposes, which is the subject of this thesis. In Chapter 3, we introduce the support vector machine (SVM) methodology for classifying food objects, and we elaborate the deep learning model and its integration with cloud computing. In Chapter 4, we describe the implementation of these models and present distance measurement and block resize methodology, which are implemented as part of the calorie estimation method. In Chapter 5, we discuss the results with respect to each models. In Chapter 6, we summarize the work and explain about how the proposed

models contribute to enhancing the user's experience of the Smartphone application in terms of time, cost, resource consumption, and accuracy of the Eat Healthy Stay Healthy (EHSH) mobile application. We also make recommendations for future work.

Chapter 2

Background

2.1 Computer Vision Algorithms Used for Food Recognition

To achieve our core goal of recognizing the food object and calculating the number of calories, we implemented three models in our system, each of which is explained in detail below.

2.1.1 Support Vector Machine (SVM)

The support vector machine (SVM) method is used in several machine-learning applications, such as text classification, face classification, and so on [14] [15]. As described in [16], SVM classification approaches, which are based on hyper planes, are computationally concentrated processes. However, in SVM-based methods, the processing steps may be complicated, especially with large data sets, which is the case in our system. We used more than 3,000 images for food classification, image segmentation identification, and calorie estimation. To achieve food recognition, we proposed a mechanism by which we could periodically update the mapreduce SVM model. In so doing, we made sure that the system would be periodically trained to correct any inaccuracies that occurred during the classification phase.

However, SVMs use a shallow linear pattern separation model with a one-or-zero feature transformation layer when the kernel trick is used. Shallow architectures have been shown to be effective in solving many simple and well-constrained problems, but their

limited modeling and representational power can cause difficulties in complicated real-world applications involving natural signals, such as human speech, natural sound and language, natural images, and visual scenes. However, human information processing mechanisms (e.g., vision and audition) suggest the need for deep architectures to extract complex structures and build internal representations from rich sensory inputs [17]. Hence, we adopted the deep learning methodology, which was adapted from neural network research.

2.1.2 Neural Networks

Before implementing the food classification model based on deep neural networks, we tried working with simplistic neural network models. In our previous work, we have used SVM to accurately classify the food image. However, because of the shallow linear patter of SVM and the limitations that were addressed in the previous chapter, we intended to switch from SVM to a neural network model.

Researchers used neural networks as the recognition tool in capturing images from mobiles. A previous study [18] developed Andromaly, a behavioral-based detection framework for mobile devices. They used a neural network to classify the collected feature vectors from the mobile device as normal or abnormal (malicious). In addition, they were able to differentiate the datasets of games or tools based on the extracted feature vectors. They used a training set of games and tools feature vectors, which were collected during the activation of the tool or the game. In addition, during the testing phase, a different collection (the testing-set) containing feature vectors in both game and tool applications was classified by the trained classifier. Another model was developed in [19], which is an automatic number

plate recognition software used for detecting Vietnamese number plates. In [19], the U-ANPR used OpenCV on mobiles to process raw image data. They were able to locate the number plate within the captured image by using a neuron network to learn and recognize characters on number plates. They used neural_net.ser file, which is a pre-trained neural network file used for determining image label probability.

In our project, we used neural networks to train the mobile system to detect food images, test it, and input the results in our application for further calorie computation. Because we already had the required features extraction parameters (i.e., color, shape, texture etc.) in the existing system, we used the results to train the neural network.

We also created an Application Program Interface (API) to transfer image data from mobile to neural networks and sending the results back to the mobile application. The NBA Droid plugin enabled us to create the image recognition for the mobile application. By training the food samples using neural network and using this plugin to work with the neural networks, we created image recognition components. By using this pre-trained file to recognize the food samples captured with the camera in the smartphone, we were able to achieve the desired recognition. However, neural networks suffer from the same limitation as SVM does. It was a shallow neural network with a single hidden layer and failed to solve complex networking problems. It did not yield the desired results in terms of accuracy compared to the deep learning model. By using deep neural networks, we were able to train the system with hidden layers with the result that it performed much better than the shallow neural networks with a single layer. The reason, of course, was the ability of deep nets to build a complex hierarchy of concepts [20]. It allowed us to address deep levels of abstraction

in the layers and the feature points, which we would not have obtained by using the former models. Hence, we chose the deep learning network for all our future research work.

2.1.3 Deep learning network

Deep learning is a new area in machine learning. It has recently revolutionized many domains of signal and information processing, in not only speech and object recognition but also computer vision, natural language processing, and information retrieval [21].

The concept of the deep learning neural network model could further be explained with a food recognition example. It is based on the abstraction of models, wherein it breaks a problem into several sub-questions until it is able to define the object. Unlike the shallow computer vision algorithm, it uses multiple hidden layers. The image is analyzed, and each pixel acts as a source of information. The features of each pixel are analyzed at various levels in the network. *Figure 1* shows a deep learning model in which the network is trying to find the solution to classifying the food object as an apple. Based on how the network has been trained, it breaks this question into sub-questions to address the various features of the apple. In this case, the system asks questions about the color, texture, shape, and size of the food object. The questions could be further subdivided to achieve a deep level of abstraction.

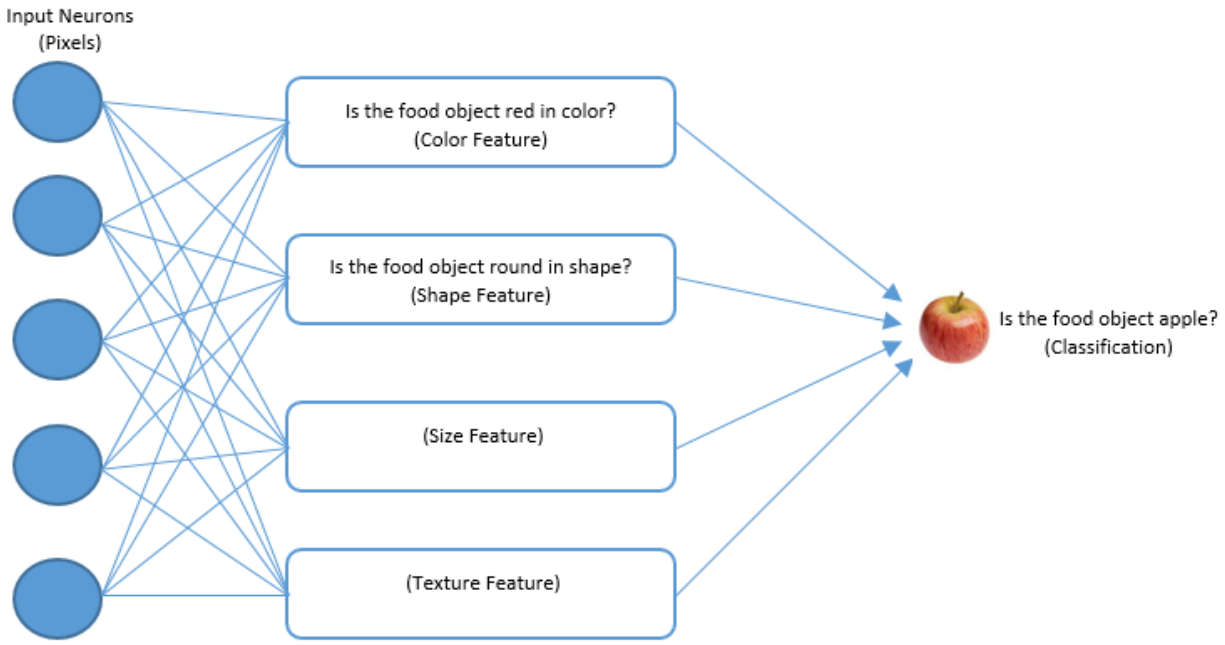


Figure 1: Concept of deep learning in the food recognition model

2.2 Android

Before we discuss the interaction of the Android application, Eat Healthy Stay Healthy, with deep learning and the calorie estimation method, we need to understand the basic principle of Android and the various sensors that are used during this process. Android is an open source platform that runs many applications through its operating system. Its middleware is specifically designed to suit the mobile phone's requirements. Android applications are Java based and compiled by the Dalvik Virtual machine.

The Android application consists of multiple activities and a broadcast receiver, which are responsible for handling the application implementation inside the device. In interacting with the system level applications, such as call receive and call reject, we used broadcast

receivers. For all other user-programmed applications, we used of the activity inside Android.

To implement the EHSH application in Android, we used three hardware sensors, which are integrated into almost all the latest Android smartphones. As shown in *Figure 2*, we used the accelerometer, magnetic field, and camera sensors as part of the hardware component. When the camera component is invoked in the Android application (EHSH), it interacts with the camera framework, which is a part of the application framework. The camera framework uses the camera driver to interact further with the camera hardware component. In closely observing the patterns of interaction of these sensors, we detected five levels of the Android architecture. Any Android application that needs to invoke the hardware of the mobile device needs to go through each of these layers. A similar interaction model was observed in the accelerometer and magnetic field sensors, in which the application first interacts with the sensor manager, which communicates with the sensor service and sensor HAL, which further interact with the corresponding drivers of the accelerometer and magnetic field at the Linux kernel level.

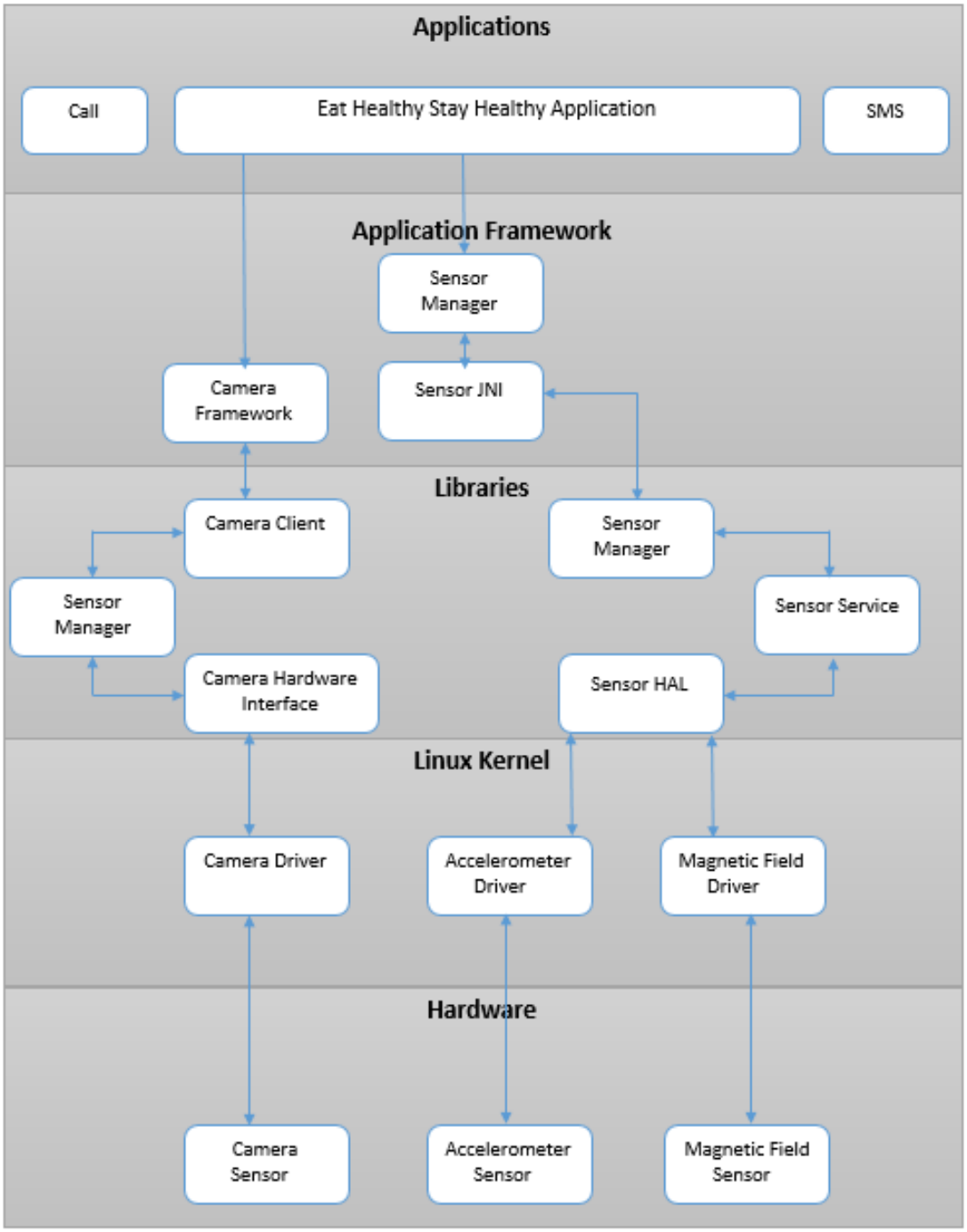


Figure 2: Android architecture and implementation of camera, accelerometer, and magnetic field sensors.

Chapter 3

Related Work

Calorie counting using a mobile phone application was investigated in [22]. They proposed a system in which the user's posture was monitored by mobile accelerometers to determine the activity performed by the user, further enabling him or her to estimate the number calories burned in activities. Although we used accelerometer sensors in our mobile application, we used the mobile device to measure the number of calories in the food portion consumed by the user. A previous study [23] used mobile sensors to detect human activities, such as walking, jogging, and running. The mobile sensor is a powerful tool, and it has helped many researchers to analyze the activity of users, further allowing them to calculate the number of calories burned. In our application, we used mobile sensors (accelerometers) to estimate the distance between the food object and the user taking the photograph. In [24], the dimensions of the food portion were measured using a 2-D image. Their system was based on the assumption that the dimensions of the circular plate were known, which, from the user's point of view, is not a viable option.

In [25], a mobile application, eyeDentify, was designed to recognize objects. As in our project, the computation requirements for achieving the recognition task were not met by the smartphone alone. By using cyber foraging in which some computations were offloaded to the cloud, they gained full control over the application parts. They were able to increase the application's responsiveness and accuracy and decrease the amount of energy used. In

[25], Ibis middleware was used to build a distribution system, which was evaluated using eyeDentify, which performs object recognition.

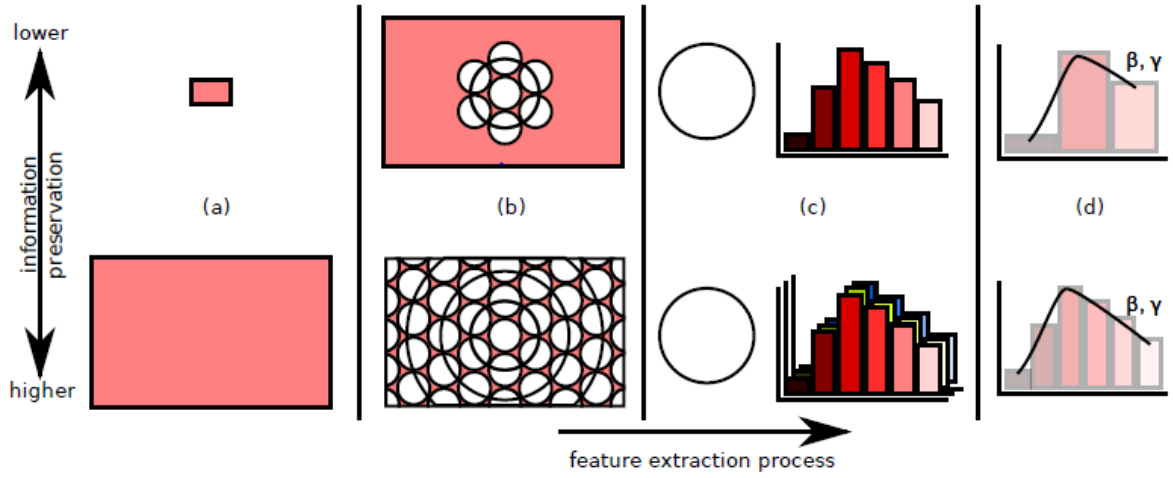


Figure 3: Image recognition process in eyeDentify [25]

In the learning mode, [25] the user takes a picture, enters the name, and stores it in the database. As shown in *Figure 3*, in training, the image is divided into circular areas, and color histograms are formed for each circular area inside the image. Based on the values of the histogram, as shown in *Figure 3*, the feature vectors determine the feature sets. It categorizes them into low, medium, and high accuracy fields. The problem with this type of approach is that it uses the Ibis distributed deployment system. Because the adapters for grid middleware in Amazon EC2 are still in progress, which prevents them from running the Ibis distributed environment in Amazon Web Services (AWS). However, in our system, we offloaded the computational part to multiple Amazon EC2 instances as part of AWS. In addition, Ibis analyzes the images [25] by dividing them into circles, whereas our system processes images based on pixels. The circular region approach [25] misses the information

not covered by the circles, whereas the pixel approach analyzes all the information in an image. Furthermore, in [25], the features used for recognition of objects were based on only the color feature, which is inefficient when the objects have the same colors. In contrast, our model is based on four feature sets.

Peerhood [26] also used middleware and cyber to develop the concept although a system is not available. In contrast, our system is applicable for Android operating system. In [26], a mobile application was designed for the purpose of barcode analysis. However, it requires a pre-started server running on computer resources.

3.1 Food Object Recognition and Calorie Computation

In [27], a 24-Hour Dietary Recall (24HR) method was proposed in which the daily food intake is listed by using a special format over a period of 24 hours. The patient is expected to recall all the food and beverages he or she consumed on the previous day, 24 hours prior to the interview. In this method, portion sizes are estimated using standardized cups and spoons. The recorded food amounts are converted into nutrient intake by using food composition tables. The self-dietary assessment of food portions and the number of calories consumed [27] has the major drawback that self-assessment is not accurate. It is not a feasible option for either the user or the dietician because it does not provide a clear record of calorie intake.

Another manual-method used to maintain dietary information is the Food Frequency Questionnaire (FFQ), which uses external verification based on double-labeled water and urinary nitrogen [28]. Although [28] focused on describing dietary patterns or food habits, it did not calculate calorie intake.

3.2 Analysis of Manual Dietary Methods

Previous methods developed in [27] and [28] were not able to determine accurately the amount of food consumed because no standardize approach was used for all food types. In addition, problems related to delays in reporting the food consumed and underreporting the size of food portions undermined the fundamental reason for maintaining dietary information. Both [27] and [28] relied on memory and required skilled interviewers who guessed the number of calories and nutrients the person had consumed. Furthermore, these manual methods did not quantify usual dietary intake, and they required complex calculations to estimate frequencies.

Therefore, there was a need for an intelligent system that reduced human intervention during the assessment and recording of dietary information and that was accurate in determining the number of calories consumed by the user.

In [29], a web-based application was proposed to detect whether the user's habits were considered risk factors for obesity. The application acquired and registered data about diet, exercise, sleep, and fat mass by using a web application and health information sensors. The major drawback of this system is that it is inconvenient for the user, and the learning process is troublesome. In addition, users must be adequately motivated to change their behavior.

In [30], a 2-sample statistical t-test feature extraction method was developed to classify fruit and analyze predictions. The method used pixels having features more prominent than others do. To achieve this goal, two performance criteria metrics were proposed to rank and reorder significant feature pixels. The problem with the proposed method is that the results were not generalizable because only three classes of fruit were included in the study.

In addition, [30] used different masks to reduce the dimensions of image space. However, the elimination of significant pixels caused missing information. Masking certain pixels that were not used in final processing led to the loss of critical information.

In [31], Food Log, a public web service, was used to capture and store the data of multiple users.. Image datasets of 6,512 dictionary and calorie-estimated images were formed. These images were used in dietary assessment. However, the overall accuracy of this system was significantly lower than in other systems.

In [32], the goal was to recognize food objects. They aimed to address the issue of deformed food objects by pairing the statistics of the ingredients with the spatial arrangement of the ingredients in order to obtain detailed statistics about the food object. In the spatial arrangement, they considered distance, orientation, midpoint category, orientation, and midpoint category.

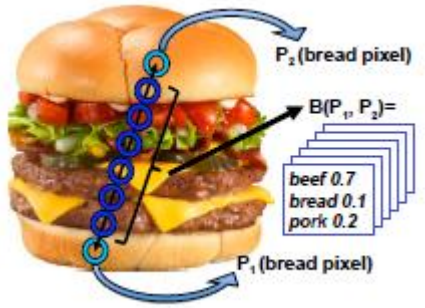


Figure 4: Pairing the ingredients' statistics to obtain the statistics of the food object [32]

In [32], the soft labeling of pixels was conducted by assigning them to nine categories: beef, chicken, pork, bread, vegetable, tomato/tomato sauce, cheese/butter, egg/other. The problem with this approach is that certain ingredients were occluded. In addition, variations

in the assembly of food objects could affect the accuracy of the overall food classification and the ingredients. This method relied on pixel-based paired spatial points, as shown in Figure 4. In the example of the burger shown in Figure 4, the spatial points are considered for beef, bread, and pork. In addition, because the histogram featured values passed into SVM, the learning is primarily supervised unlike deep learning, which has the option of both unsupervised and supervised learning.

Figure 5: Color features of all the ingredients inside the burger [32]

Another problem with [32] is that because all pixels are colored in the training images, the color feature alone does not provide a sufficiently accurate reading for classification, as shown in Figure 5. The same drawback was reflected in the classification results, which ranged from 49.7% to 78%, compared to the range 59.05% to 99.9% achieved by our system. The reason could be that unlike our system, the system proposed in [32] did not rely on other features, such as edge, color, texture, or key points. Instead, it was based on the statistical information between a selected pair of pixels.

In [33], an application was designed to recognize foods in videos of people eating in restaurants, which were taken with a web camera. Based on the video recordings, they also determined the calorie content of the food being consumed. In [33], the food database

included video recordings of 640 x 480 and 30 fps and four photos of food taken from different angles. Based on video recordings made in nine restaurants, the food database included burgers, sandwiches, salads, chickens, and drinks. The problem with this approach is that it was based only on fast foods and focused on food offered by chain restaurants, which limited its application to certain food types. The user would only be able to estimate the calories in the food from one of the nine restaurants (information that was already readily available). Another problem with this approach was that the food data, including the video and image data, was eventually processed in the laboratory. Therefore, the collection of data on the image processing tasks and the in-lab videos required manual inspection during the food image/video processing. In addition to calculate the calorie consumption, they simply referred to the restaurants' websites. Hence, if the restaurant were unknown, the computation of the number of calories in the food object would be inaccurate.

Chapter 4

Proposed Method

4.1 Deep Learning

In this section, we discuss the various methodologies we adopted in this thesis. As explained above, our primary goal was to recognize the food object and then measure its calorie content. To classifying the food object, we used deep learning, which helped us achieve high accuracy. We propose a new methodology in which we used a unique approach to estimate the distance from the food object and then calibrate the size of the image in order to measure the calorie content.

4.1.1 Deep learning method

We implemented the deep learning methodology, specific to food images, which enabled the system to identify food features based on color, contour, texture, and size, in order to classify the food object with accuracy. Although our model is based on the deep learning model by [35] and [36], the contribution has been mainly in terms of making it specific to recognize food objects, by training the deep learning network with different food classes image sets and generating food image specific trained model files and using it further to classify the food images.

Stochastic gradient descent (SGD) algorithms are the most efficient when the training set is large and redundant as is the case in most applications [34]. We were thus able to achieve highly accurate results during the classification stage compared to our previous

approaches, which used the support vector machine (SVM). In the deep learning method, there are two stages in the classification of the food object. After we trained our food images using the deep neural network, we generated the model file, performed segmentation, and extracted features that were further written in hidden layers in the deep network. The libraries and the model files of deep neural network used in this thesis are based on the work of Hinton [35] and Srivastava [36]). After customizing the top-level feature layer, we were able to generate the results of the classification.

1. Deep Neural Network

The deep neural network works according to the principle of the backward propagation algorithm. In this section, we will describe in detail the total number of layers, the number of hidden layers, the total number of input neurons, and the total number of output neurons. The design of deep neural network, which is shown in Figure 6, is based on a two-step process in which the first, pre-training step determines the hidden nodes and the edge parameters and the second step is the back-propagation, in which is the base and the weights are adjusted to achieve the desired classification results. This is further explained in the next section.

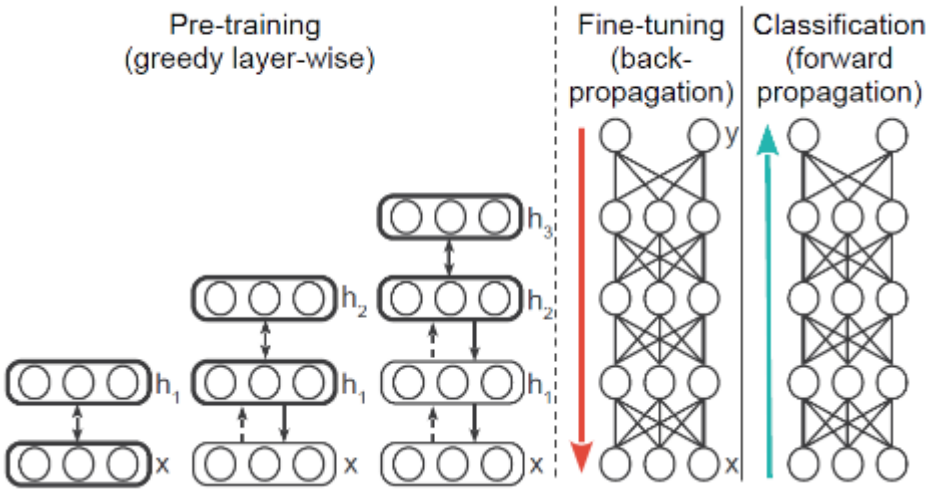


Figure 6: Design methodology of the deep neural network [21]

2. Training the Deep Neural Network

The neural network computes the differentiable function of its input. For example, our application computes the probability of a match of the input image with the corresponding label set, $p(label|n\ input\ image)$ [35]. The standard way of modeling a neuron's output f as an activation function of its input x is either a hyperbolic function:

$$tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \text{ or} \qquad \text{.....(1)}$$

$$sigmoid(x) = \frac{1}{(1 + e^{-x})} \text{ [35]} \qquad \text{.....(2)}$$

However, we used the term rectified linear unit (ReLU) to refer to a unit in a neural net that uses the activation function max(0;x) [35]. Compared to these activation functions (hyperbolic or sigmoid), the training of deep convolutional neural networks with ReLUs is relatively fast [37].

To clarify the deep neural network that we built, we refer to the following example: Consider that we choose as output greyscale images 28 by 28 pixels in size. If we use the notation x to denote the training input, then there are 28*28 = 784 input neurons. Each entry in the neuron represents the grey value of a single pixel in the image. We denote the corresponding desired output as y = y(x). We aim to train the network in a manner that enables us to find the weights and biases so that the output from the network approximates y(x) for all training inputs x. Therefore, if we have a training image set of food images, and we want to classify the food type as an apple during the learning phase, we could achieve this goal by adjusting the weight and bias values. To quantify how successful we are in achieving this goal, we define a cost function [38]:

$C(w,b) = \frac{1}{2}\sum x|y(x) - a|^2$ [39] .....(3)

where w denotes the collection of all weights in the network, b is all the biases, a is the vector of outputs from the network when x is input, and the sum is all training inputs x.

In other words, we want to find a set of weights and biases that reduces the cost as much as possible. We do that using an algorithm known as stochastic gradient descent, as described in [39]. By using a smooth cost function, such as the quadratic cost, it is easy to figure out how to make small changes in the weights and biases in order to reduce the cost. 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 to find the weights and biases that minimize the quadratic cost function C(w,b). The gradient descent algorithm is used to find the weights wk and biases bl that minimize the cost C. The gradient

vector ∇C has corresponding components $\partial C/\partial wk$ and $\partial C/\partial b$. The stochastic gradient descent will enable faster learning by estimating the gradient ∇C by computing ∇Cx in a small sample of randomly chosen training inputs. By averaging this small sample, we can quickly obtain a good estimate of the true gradient ∇C, which helps to speed up the gradient descent, and therefore learning [35].

The stochastic gradient descent algorithm works by randomly picking out a small number m of randomly chosen training inputs (e.g., 10 images from the original set of 100 images). We label these random training inputs X1,X2,…,Xm. The stochastic gradient descent algorithm then selects a randomly chosen mini-batch of training inputs [40], and trains with those. The weights and biases are computed as follows:

$$w_k \rightarrow w_k' = w_k - \eta/m\Sigma(\partial C/\partial w_k) \qquad \text{.....(4)}$$

$$b_l \rightarrow b_l' = b_l - \eta/m\Sigma(\partial C/\partial b_l) \qquad \text{.....(5)}$$

where, the sums are all the training examples Xj in the current mini-batch, and η is the learning rate.

We then choose another randomly chosen mini-batch of inputs and train with those until we have exhausted the training inputs. The back-propagation algorithm is used to compute the gradient of the cost function quickly [35]. 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. For example, this algorithm will help us adjust the weights (w) and bias (b) during the learning phase, such that we can finally determine the output as one of the effects of the output of network two (apple or cherry), without affecting 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 *Figure 7*, considering that we have taken into account the color feature, any changes in weight w1 or bias b would cause small changes in the results. In this case, changes in apple and cherry, which have similar color features, will alter the results. If the probability of the image (p >0.5 towards apple), it would be classified as apple, and the same would apply to any food type [38].

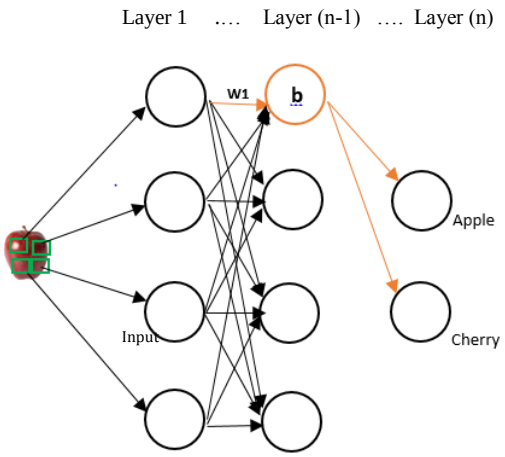


Figure 7: An example of implementation learning using the stochastic gradient descent algorithm [38]

3. Stochastic Gradient Descent (SGD) Algorithm

This algorithm implements stochastic gradient descent as follows [20]:

1. Input a set of training examples
2. **For each training example *x*:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:
    - **Feedforward:** For each $l = 2,3,\ldots,L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$
    - **Output error $\delta^{x,L}$:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$

- **Back propagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \eta/m \sum_x \delta^{x,l} (a^{x,l-1})^T$ and update the biases according to the rule $b^l \rightarrow b^l - \eta/m \sum_x \delta^{x,l}$.

Where, L is the layer number, $\boldsymbol{\delta^{x,L}}$ is the output error, b^l is the bias, wl is g the weight, and C is the cost.

4. Food Object Classification Based on the Deep Neural Network

When we have trained the deep neural network to detect food images pertaining to certain classes, the system then generates a model file. While integrating this system with a mobile device, we ensured that the image processing steps were performed in the cloud. This was done primarily to remove the overhead incurred by processing from the user's mobile device. We used the concept of cloud virtualization, which enabled us to create a replica of our mobile application to be run in cloud. The computing in cloud was performed using Amazon EC2 instances. When the user clicked on the photo of the food object, the image and other details (e.g., distance information and user information) were sent to the cloud for processing. Here, the image was first segmented to obtain the food portion, and the features (i.e., color, texture, size, and shape) were extracted from the target object. When these features were extracted, the image was then mapped against the model file, which provides the probability and food tag associated with the food object [38].

4.2 Calorie Estimation

4.2.1 Finger-based calorie measurement

In the finger-based calorie measurement, we used the thumb of the user and its placement on the plate, as shown in *Figure 7*. The thumb is measured by a one-time calibration process, which is used as a size reference to measure the real-life size of food portions [41]. Examples of food picture capturing and thumb isolation and measurement are shown in *Figure 7*. Compared to the calibration methods used in similar systems, the thumb was more flexible, controllable, and reliable. For users with a thumb disability or amputated thumbs, another finger or a coin could be used instead. The finger-based method is more widely used than the special plates or cards used in other systems.[41]

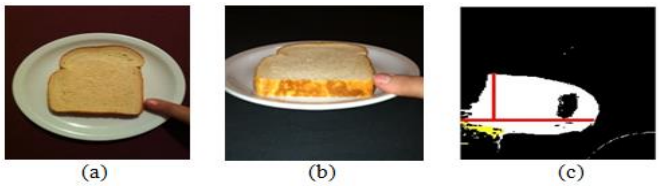


Figure 8: (a, b) Test images with food and thumb; (c) Calculation of the thumb dimensions [41]

Although our system achieved excellent results, the issue of calibration was bothersome because it was difficult for the user to take a photo with one hand on the mobile phone and the thumb on the other hand placed near the plate. In the next section, we address this issue by proposing a method to calculate the distance between the object on the plate and the person taking the image.

4.2.2 Distance measurement method for estimating calories

The application calculates the distance from the food object during the live camera feed when the user is about to capture the photo. The distance measurement technique has been specifically proposed to compute the distance from the food objects, which has not been proposed previously. During the registration phase, the user is prompted to enter his or height (in feet or cm), which is used later in measuring the distance of the food object from the user's phone [38].

We then calculated the mobile phone's orientation by using the rotation matrix. The values of the rotation matrix were obtained from the accelerometer sensor and magnetic field sensor values. As shown in *Figure 9*, the values returned by the rotation matrix contained the azimuth (rotation around the z-axis), pitch (rotation around the x axis), and roll (rotation around the y axis) [42]. To obtain the sensor's values, we first invoked the accelerometer sensor instance (API, which is TYPE_ACCELEROMETER in the sensor event class) by using the sensor manager in Android. A sensor event holds information, such as the sensor type, the time-stamp, accuracy, and the sensor data [42]. We used the same process to obtain the values from the magnetic field sensor. For our application, EHSH, we captured the food object images taken in landscape mode by the mobile phone's camera. The landscape mode provided a detailed overview of the food object on the plate, further helping us in calculating accurate dimensions of the food object. The coordinate system (*Figure 9*) was measured with respect to the mobile phone's screen in default orientation. The Android accelerometer sensor was used to measure the acceleration applied to the mobile device [38].

Hence, we were able to obtain the angle (radian) based on orientation of the phone to the target food object. We converted the value of the angle, originally in radian, to degrees (theta θ). Based on the angle value and the height entered by the user (e.g., 165 cm), we obtained the distance of the target food object from the mobile phone. After obtaining the angle, the system calculated the distance of the food object to the phone's camera, as shown in *Figure 10*.

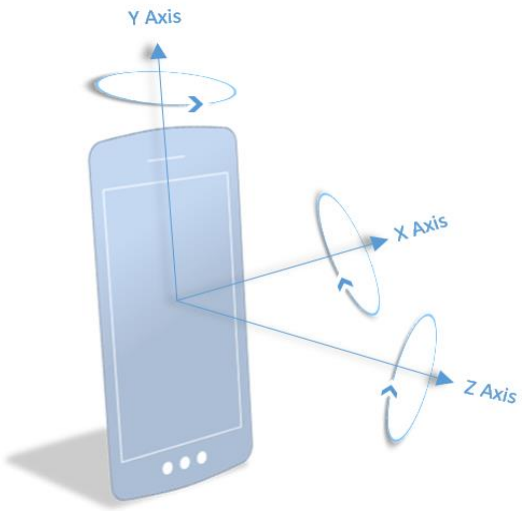


Figure 9: Coordinate system used to calculate the orientation axis of the Android phone

Based on the rotation matrix, we obtained the orientation angle (i.e., the angle at which the phone is placed in space) from the accelerometer and the magnetic field sensors (as explained above). After we calculated the orientation angle, we used the height of the person (h) to measure the distance (d). As shown in *Figure 10*, the height of the person (h) is given by the user while creating the user account. We assumed that in a normal scenario, the user

would use the phone's camera to capture the image of a plate that is placed on a stool at height (h/2), that is, half the height of the person. We were then able to calculate the computed height, which was further used to calculate the distance of the food object from the phone's camera. We used the following formula [38]:

$$Distance\ to\ calculate(d) = \left[\tan(Orientation\ angle(\theta)) \times \left(\left(Height\ of\ Person\ (h)\right) - \left(Height\ of\ Stool\ \left({}^{h}/_{2}\right)\right)\right)\right] \quad \text{.....(6)}$$

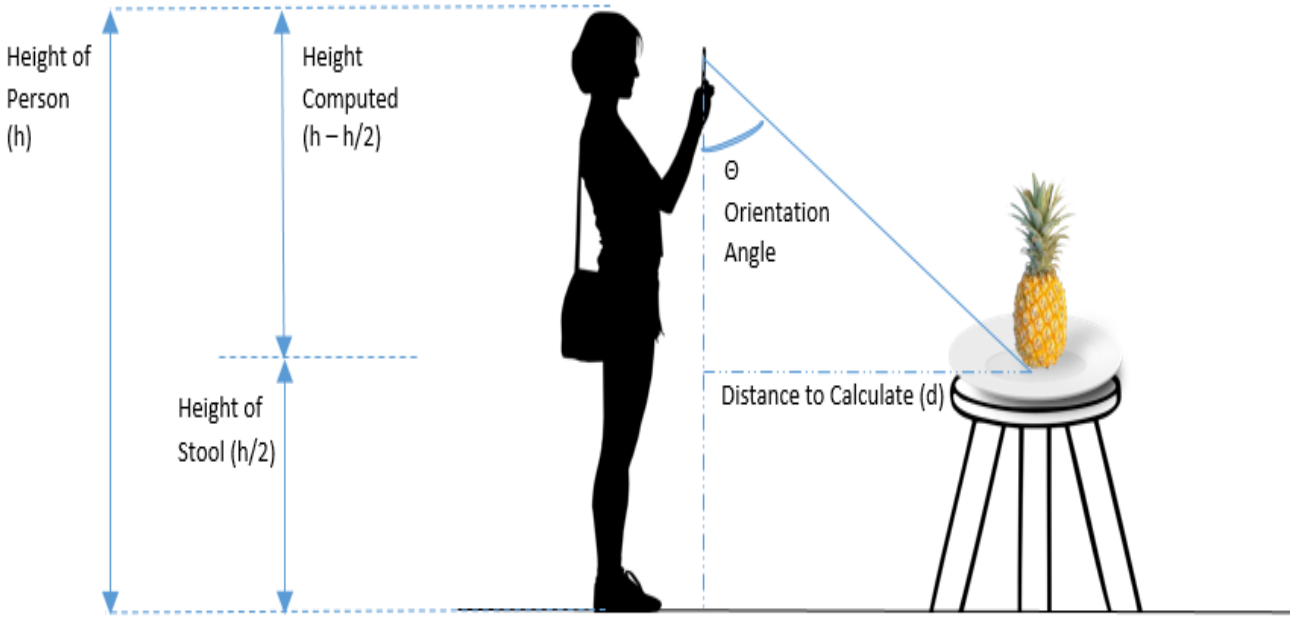


Figure 10: Calculating distance from the phone's camera to the target food object [38]

4.3 Comparison between SVM and Deep Learning Methods Used for Food Object Recognition

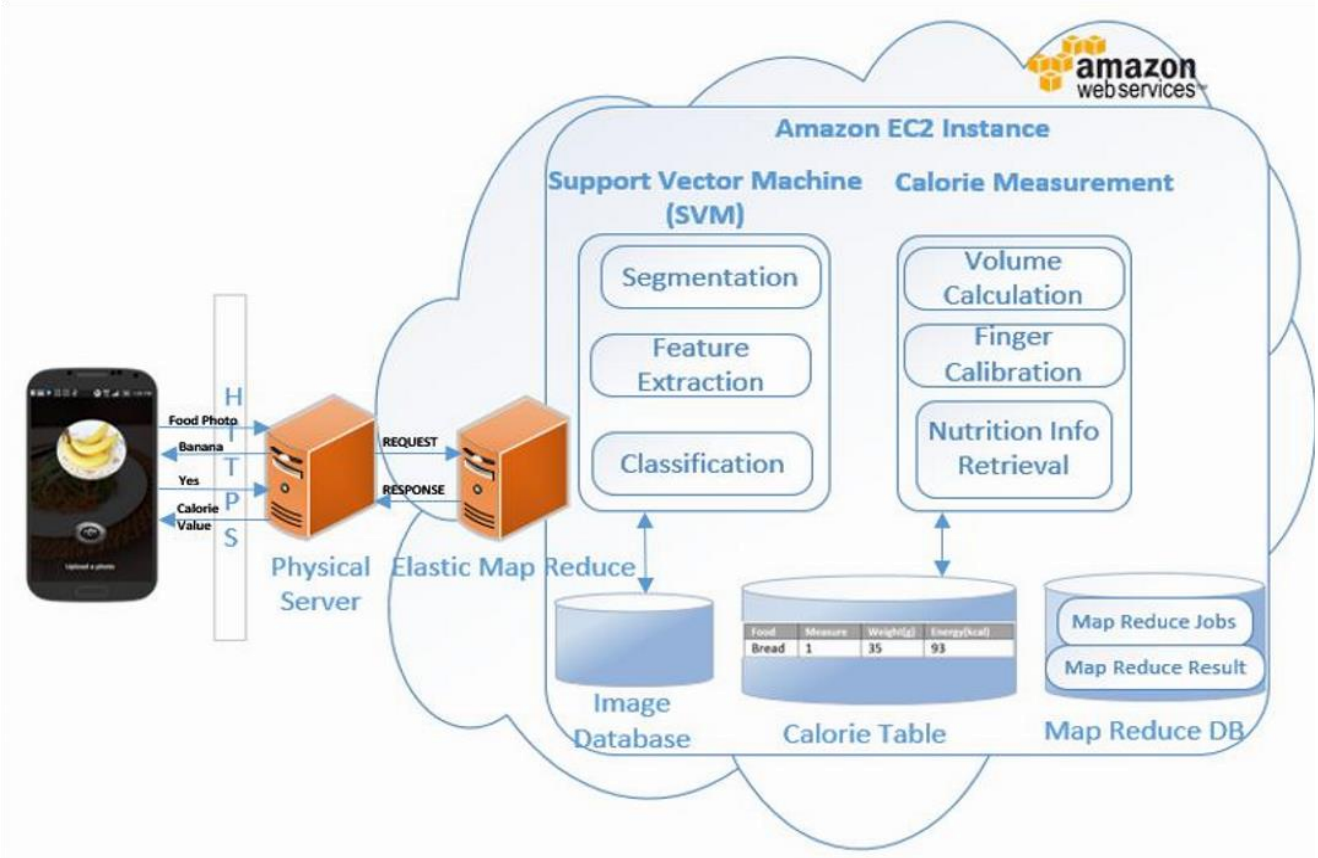


Figure 11: Architecture components of SVM-based implementation of the e-health application [38]

In our previous work [41], [43], and [48], we developed a mobile system that measured the number of calories of the food from the image in the user's smartphone. In that system, when the user captured the image of the food item on the plate, the image was sent to the cloud for food recognition and calorie computation. Food recognition was done by the SVM running in the cloud. The image was recognized, and the calorie details

matching the image were retrieved from the database, which also existed in the cloud. The results were then prompted back to the user's phone. In addition, for calibration, the user had to put his/her thumb near the food when the food picture is taken. The components of SVM based implementation of our e-health application are shown in *Figure **11***. We used SVM for image processing, MapReduce for the cloud model and volume calculation and finger calibration for the calorie calculation. The user uses a smartphone to connect to a server through http request and response, which further connects to the cloud server (in our case, an Amazon EC2 instance). We then used Elastic MapReduce for parallel computing in the Amazon cloud. [38]

Although our system achieved excellent results, the issue of calibration was bothersome because it was difficult for the user to take a photo with one hand on the mobile phone and the thumb on the other hand placed near the plate. In this work, we addressed this issue by proposing a method to calculate the distance between the object on the plate and the person taking the image. By using the distance calculation method, the user now does not have to keep the finger on the plate for calibration. [38]

Hence, we proposed the deep learning methodology to train and classify the food object accurately according to its corresponding label. With the help of mobile sensors, the system in real-time can gauge the distance from the to the food object. The system then records this value and sends it to the cloud with the image of the food. The image is then processed in the cloud with the help of virtualization, and the results (including the calorie value and the food object label) are sent back to the user via the mobile device. We also proposed a new method in which the application assists the user in real time in determining the ideal distance from

which the user must capture the photo. If the user fails to capture the photo from the this distance, the system will recalibrate the block size based on the distance measure. Hence, calibration will always be accurate, irrespective of the distance from which the user captures the photo [38].

The two approaches described above are used for recognizing food objects. In the first method, we introduced a new semi-automatic system that used SVM as a classification tool to assist dieticians in measuring the number of calories and daily nutrient intake for the treatment of obese and overweight patients. The second method, which was based on the deep learning model, helped us achieve accurate results. In addition, when integrated with the deep learning model, the application performs in a fully automated manner, in which the system sequentially follows all the steps described in *Figure 15* and gives the result of the calorie calculation. Compared to the SVM method, the deep learning method does not require human intervention, neither during the classification stage nor in calorie estimation, even from the perspective of administrative.

Chapter 5

Implementation

5.1 User Interface

Our system has a user-friendly and easy-to-understand interface, which was developed by using the Google Android platform. The system uses a two-step process by which the user is able do the following:

- Estimate the expected calorie intake (number of calories in food before consumption)
- Calculate the actual calorie intake (calculated by the leftover food on the plate)

As shown in *Figure 12*a, when the user logs in, he or she is redirected to a page that has two steps [43]:

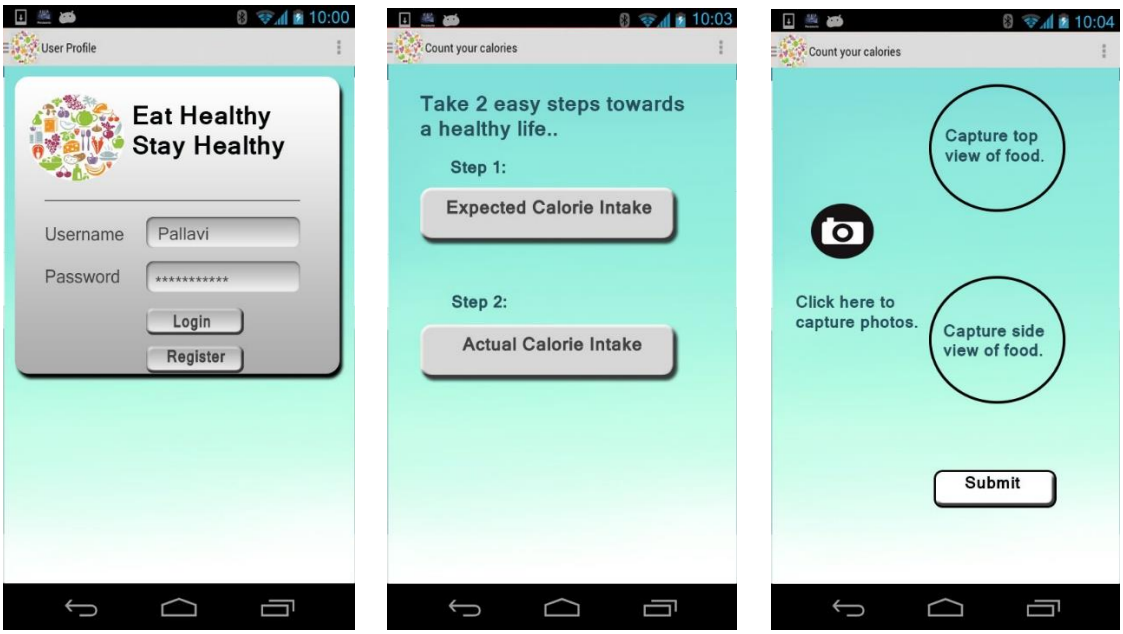
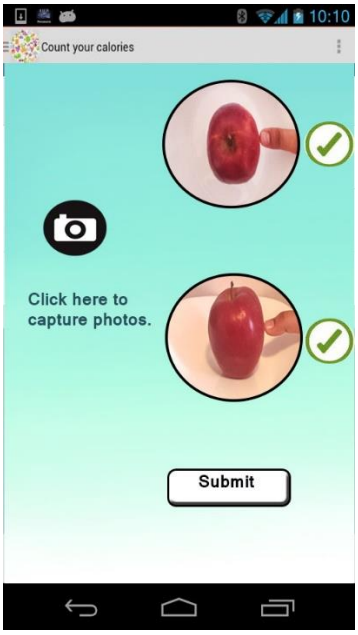


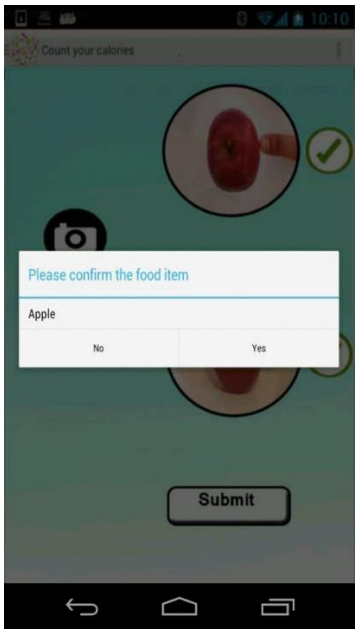
Figure 12: (a) User Login (b) Two Step Process (c) Capture Photos

Step 1

When the user clicks on “Expected Calorie Intake” the application will redirect him or her to a new page. The user then captures the images from two angles: the top view enables the application to extract the food portions; and the side view enables the application to analyze the height of the food item in the dish (*Figure 12*c). These photos should include the thumb of the user, which is used for size calibration. As an alternative to the thumb, the user can pace a coin inside the image, and the system will use this coin instead of the finger to translate the portions of the food from the image into real life size. The system is designed to store the patient’s thumb size during the one-time calibration process. The tick beside the photograph indicates that the photos have been captured, and the user can now click on the submit button, as shown in *Figure 13*a. When the user hits the submit button at the bottom of the screen, the application will prompt to confirm the food type, as shown in *Figure 13*b. If the food type suggested by the application is correct, the user then clicks on "Yes”; if not, the user clicks on the "No" button [43].

If the user clicks "Yes,” the application will then display the estimated calorie value of the food type, as shown in *Figure 13*c. If the user clicks on "No,” the application will prompt the user to enter the correct food type and further displays the estimated calorie value based on the type entered by the user. To begin step 2, the application then redirects the user to the main page of “Count your Calorie” [43].





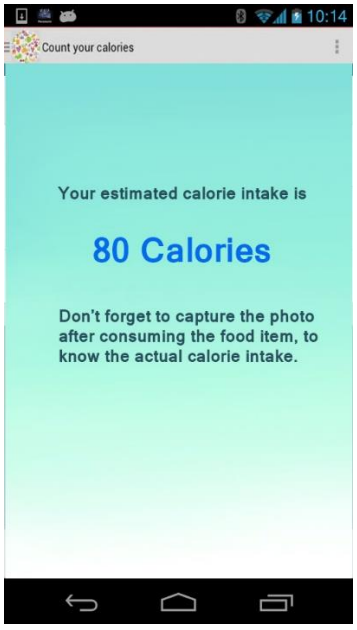


Figure 13a: Photos uploaded b: Confirm food type c: Estimated calorie value.

Step 2

Step 2 involves capturing the leftover food in the dish after the food is consumed. After the user finishes the meal, he or she clicks on the "Actual Calorie Intake" button. The process in Step 1 is repeated, but the user has to capture the photo of the finished meal only once. When the submit button is clicked by the user, the application displays the actual number of calories consumed. The net result is computed based on the estimated number of calories computed minus the number of calories in the leftover food on the plate. In the example shown, an apple was the food type, and its calorie value was computed. We took two snapshots of the apple, the top view and the side view (*Figure 13*). After we clicked on the submit button, the application was able to identify the image as an apple, and the user was prompted to confirm the food item (*Figure 13*). The application was able to compute the estimated value of the apple as 80 calories. We then consumed parts of the apple and clicked

on “Actual Calorie Intake.” We took only one snapshot of the leftover apple. The application was able to calculate the number of calories consumed by the user and display the result as 50 calories [43].

5.2 Food Database Creation

5.2.1 Food database based on the finger calibration method

In our system, some parameters may have affected the results. In collecting the food images in our dataset, we divided them into two categories: single food portions and mixed food portions. We took into consideration important factors that could affect the accuracy of our results. Specifically, we used a variety of the following factors:

a) Camera

The camera affects the results because of its lens, hardware, and software. We used three different cameras in our experiments: Canon SD1400, iPhone 4, and Samsung S4.

b) Lighting

Lighting is an important parameter because it directly affects image segmentation, which in turn affects the rest of the algorithms. To account for this factor, we took pictures of the same plate in three different locations with different lighting.

c) Shooting Angle

Another parameter is the angle of the photography. We chose three different angles for all pictures: approximately 30, 90, and 150 degrees from the plate of food. Hence, images were taken of each plate in three different lighting locations from three different angles.

d) White Plate

We used a white plate in all images to neutralize the potential effects of the background, which facilitated the tasks of food segmentation and food recognition.

e) Thumb

A one-time calibration process was used for the thumb, which was a reference to measure the real size of the food portions. Compared to the calibration methods used in similar systems, the thumb is more flexible, controllable, and reliable. For users with a disabled or amputated thumb, another finger or a coin could be used instead. The latter is more prevalent that the than special plates or cards used in other systems.

5.2.2 Food database based on distance measurement method

In this thesis, we propose a profound approach that is used to calculate the number of calories in food objects consumed by users. Our approach uses distance measurement and the block resize methodology to enable the accurate calculation of calories in the food portions on the plate. In this method, we broadly classified food objects into three categories.

5.2.2.1 Food object types

Users can capture images of food objects conveniently. However, we defined some guidelines that enable the system to classify the food object into one of the categories described below and to apply the calorie estimation method accordingly. To achieve this goal, the segmentation algorithm is used, which initially allows users to classify the food object and make smart decisions. In the training phase, we captured food images based on two categories.

- **Single food objects**

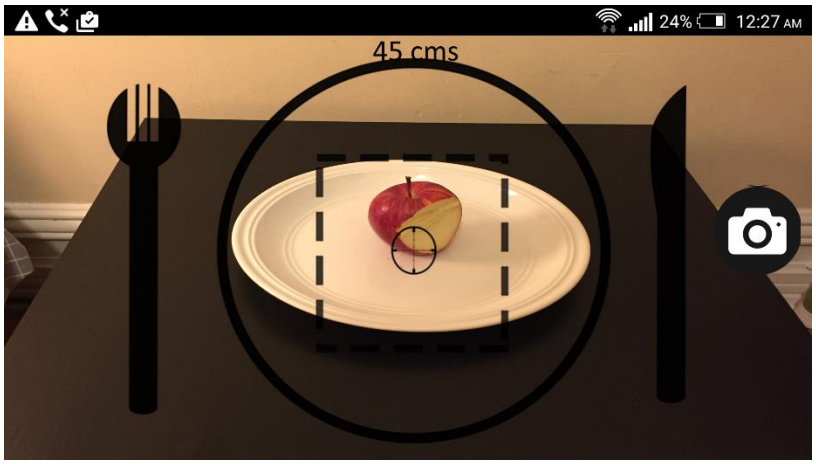
We trained the system using many images of single food objects. Initially, our focus was on recognizing the single food objects on the plate. As explained above, we trained the system using 200 images of each food class, specifically single food objects. For example, we trained the system with 200 images of an apple taken in different lighting conditions, distances, and angles. Since we used the distance measurement method, we could not use the standard food image database. We had to train the system using images that satisfied all conditions.

- **Single consumed food objects and multiple mixed food objects**

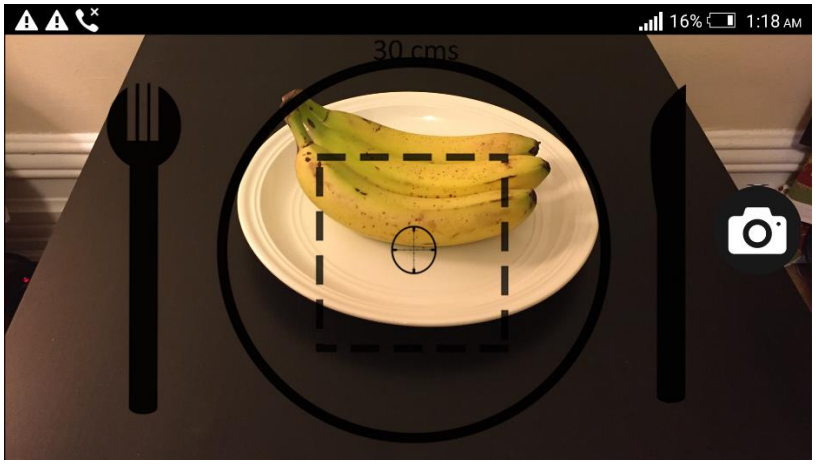
We have trained the system with images of food objects that were partially consumed and left on the same plate. For example, we trained the system with images of a half-eaten apple, banana, or other fruit taken from different angles and positions. We used the same training concept for the multiple food objects. We trained the single food objects on the plate, but during the testing phase, we used a segmentation algorithm to split the multiple food objects into single food objects and then tested them as single objects.

5.2.3 Distance from which food object images were captured

In order for the system to calibrate the distance between the food object and the current position of the phone, the altitude at which the phone is held must be determined. As explained in the previous section, at the start of the process the user enters his or her details (including height). Hence, while capturing the image of the food object, we were able to determine the distance of the food object from the user. During the training phase, we trained the system with images of the food object that were captured from various distances.



(a)



(b)

Figure 14: Photos shot from distances of 45 cm and 30 cm (a) apple (b) banana, respectively

To capture images of the food objects from specific distances, we placed the camera's crosshairs just below the food object, as shown in Figure 14. When the food object was placed at the desired distance and under the crosshairs, we captured the images of the food object.

5.3 Deep Learning Implementation

Deep Convolutional Neural Networks (CNN) allowed us the flexibility to retrain the top levels of the network to spot the relevant images, even on low-powered mobile phones. As part of the food image recognition approach used in the Android phone, we proposed a model in which we integrated our mobile application with the deep neural network. The CNN served as the backbone of the application. It handled the training and testing requests at the top layers without effecting the central layers. The high-level layers are important because they can be seen as adjectives that help the output layer make its final choice between different classes of image sets. These high-level layers are also useful for choosing between other classes that have not been trained.

We built the custom top layer in the deep neural network on the jetpac and libccv networks [44][45], which respond to the images they were trained to detect. This allowed us to embed the functionality in the Android application. The convolutional neural architecture was based on the model described by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton [35].

Settings for Experiments

Training Settings

For GPU processing we used the g2.2xlarge Amazon EC2 cloud instance wherein we installed CUDA SDK of version 6.5 on top of Ubuntu Server 14.04 LTS (HVM). After installing CUDA, we made sure to install Cudamat. For installing CUDA, we had to make sure of configuring the Nvidia driver on the cloud instance. We also installed Python 2.7, Numpy,

Scipy on top of our Ubuntu 14.04 cloud instance and further making it suitable to deploy the training the models.

Testing Settings

For testing the deep learning models, we did not install CUDA on top of t2.micro Amazon EC2 cloud instance as we performed feature extraction for every test images. We further installed HDF5 and OpenCV on t2.micro instance.

The first step in our approach was to generate a pre-trained model file with the help of the CNN network. We did this by first capturing a set of images in one particular class (e.g., 50 images of the apple class) and then labelling them with an object name set (e.g., apple). These were a set of relevant (positive) images. After we captured the image sets, we used them to train the system. Because the training took place virtually on the server, we had the required processing power, and the system was trained quickly (depending upon the number of images in a class). As part of the second step of training, we re-trained the system using the set of negative images (i.e., images that do not contain the relevant object). In our case, we trained the system using the background images, so it did not recognize them as part of the previous class. When the model file was generated from the training, we loaded it into the Android application and tested it against the captured images submitted by the user, as shown in *Figure 15*. The system then performed the image recognition process and generated a list of the probabilities against the label name. The label with the highest probability was prompted in the dialog box so the user could confirm the object name. When the object name was confirmed, the system performed the calorie computation by calculating

the size of the food item with respect to the finger in the frame. It then printed the output with the number of calories.

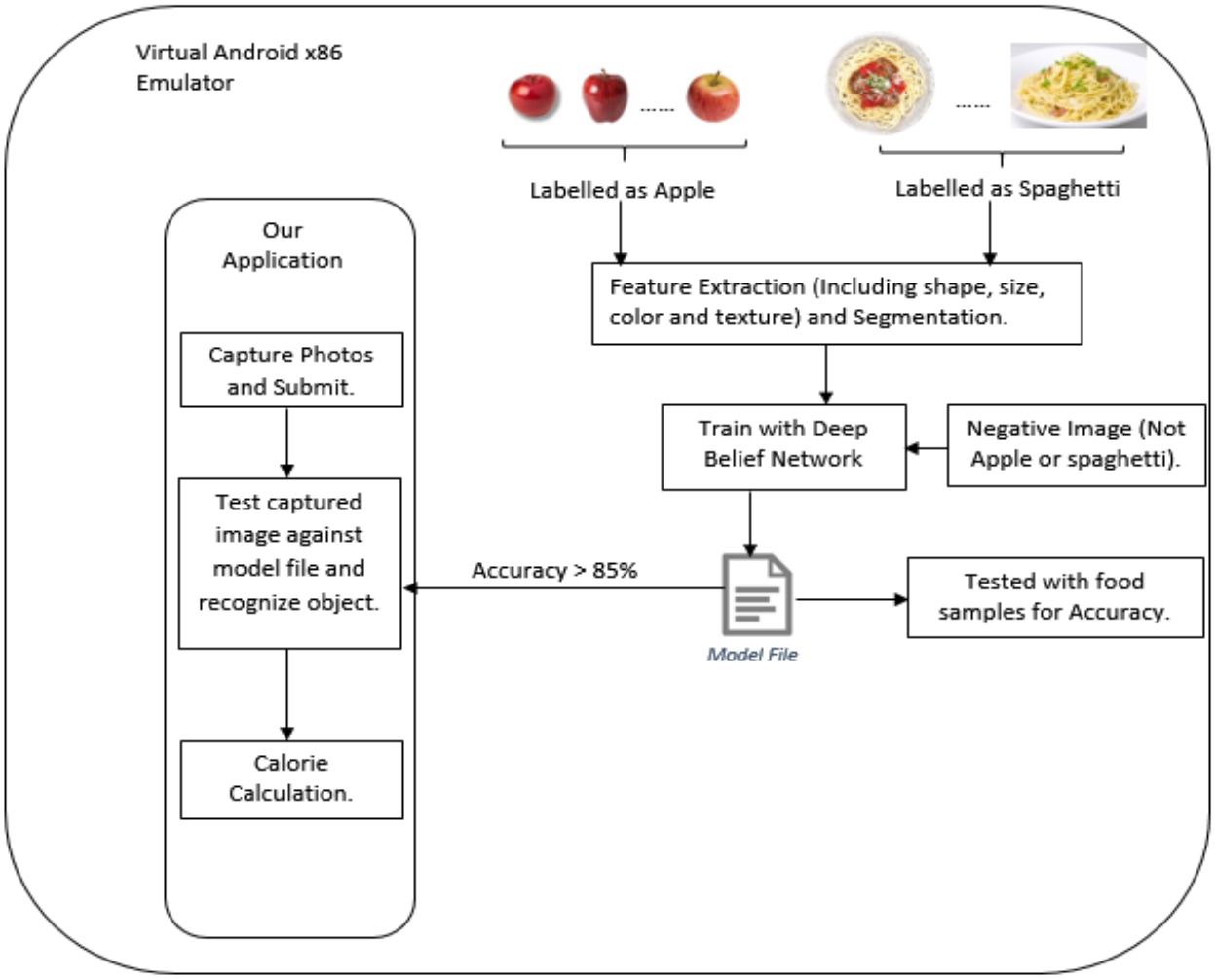


Figure 15: Implementation of the deep learning network in the Android application

5.4 Calorie Estimation

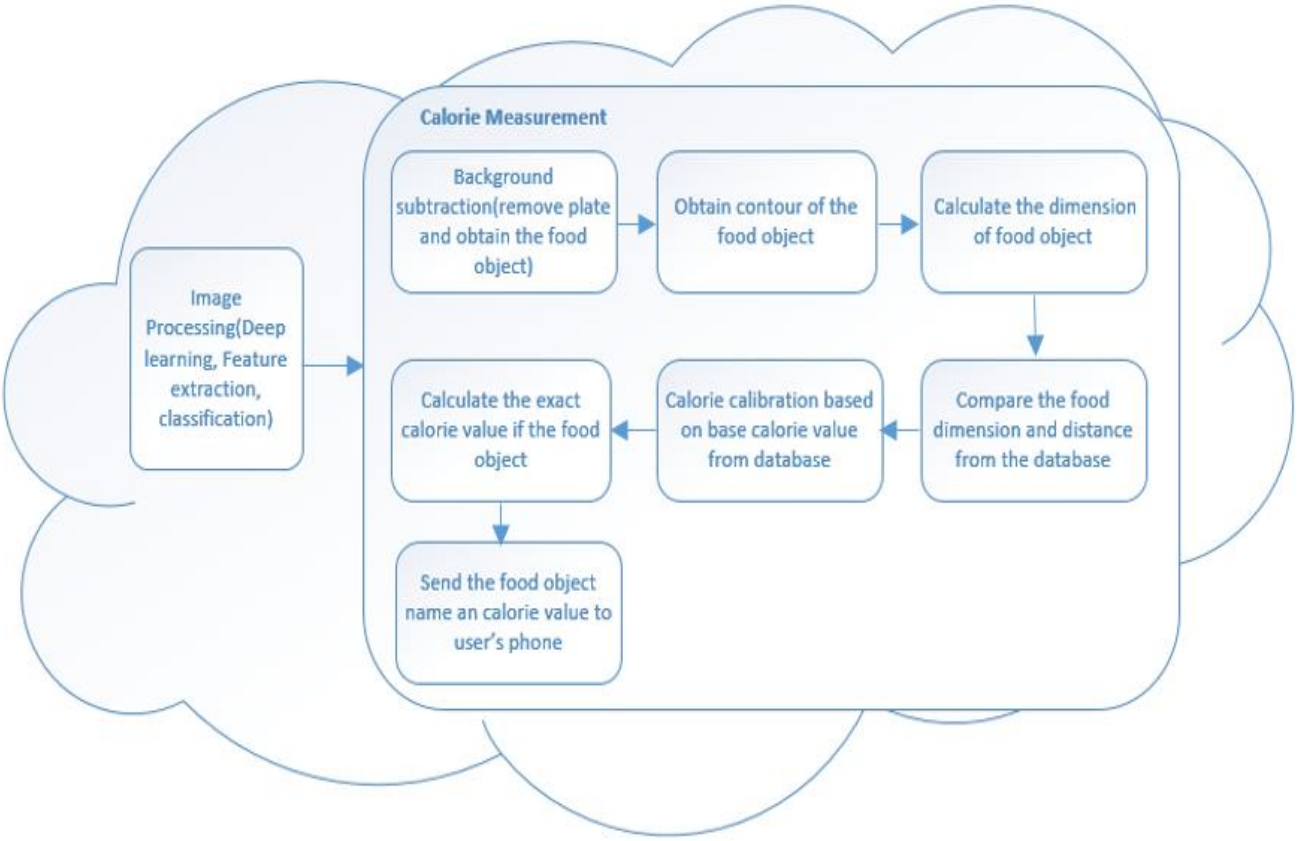


Figure 16: Cloud architecture comprising image processing and calorie estimation [38]

For performing calorie estimation, we have proposed the block resize method that uses the measured distance values along with the recognized food object name to further estimate calories.

As shown in *Figure 16*, the first step in calorie estimation is to access the food content and its quantity on the plate. We did this by obtaining the contour around the food object using the Canny edge detection algorithm [37]. When we obtained the contour, we were able to measure the food object and obtain the area of the closed contour around it. The arc length and contour were obtained using opencv functions [46][47], which helped determine the dimensions of the food object in pixels. We then performed a linear regression analysis on

the contour area and the perimeter (arc length) compared to the weight. We would have usually used a weighing machine to calculate the food quantity (e.g., in g or mgs). Because asking the user to weigh the food object on the plate was not feasible in the proposed method, we developed a novel methodology to access the weight of the food object present on the plate and further determine its calorie content.

In the current scenario, there is an image of the food object on the plate. Based on the results of the deep learning model, we also know what the food object is. The fact that we have recorded the distance that the camera was positioned from the food object gave us a crucial reference point. We can ascertain from these parameters that based on the image of the known food object, if we can calculate its dimensions according to the reference point, we would be able to obtain a quantifiable value of the food object, the unit of which would in centimeters or feet, rather than be grams or milligrams. We can then calibrate the weight of the food object on the plate (in grams or milligrams) and then obtain its calorie value. A main assumption of this method is that the user would always follow the instructions. For example, in our Android application, the user needs to take the photo from a specific distance, such as 50 cm. This might mitigate the usability of this application [38].

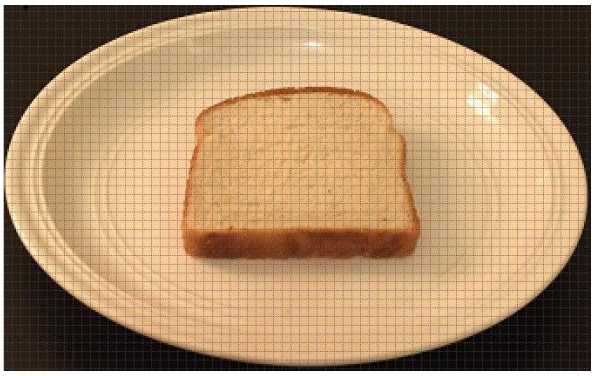


Figure 17: Bread from a distance of 45 cm [38]

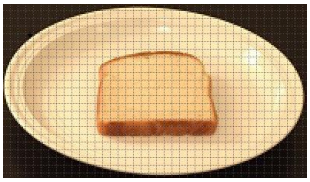


Figure 18: Bread from a distance of 80 cm [38]

Hence, to handle this problem, we proposed a distance calibration method that will enable the system to calculate accurately the calorie value, irrespective of the changes in the distance from which the image of the food object is captured. Our primary goal was to determine a constant as a reference point, with respect to the varying distances (d). We therefore created equally sized blocks on the image by using grid lines, as shown in *Figure 19*.

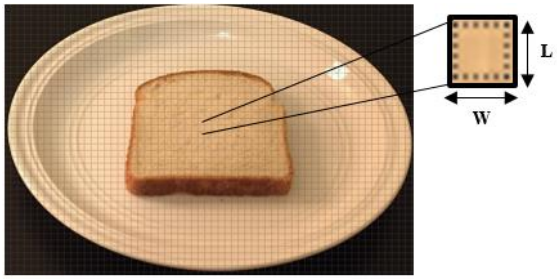


Figure 19: Bread segmented in blocks [38]

The block size remained the same in all images (irrespective of the distance from which the images were captured). *Figure 17* shows an image of a slice of bread taken from a distance of 45 cm. An image of the same slice of bread was taken from a distance of 80 cm (*Figure 18*). Because the block size was constant, the system then calculated the total number of blocks in images that were taken from varying distances. The above figure shows that the total number of blocks decreased as the distance increased (i.e., the distance from which the image was taken) [38].

The total number of blocks in the image of the slice of bread was 1,620 when the image was captured from a distance of 25 cm. This was reduced to 84 blocks when the image was captured from a distance of 90 cm. The lower value for the same food object (in this case, the slice of bread), would affect the final calorie calculation because the weight of the food object would be equated to the dimensions of the food object. This should not be the case because the food object remained the same throughout. Hence, we calibrated the scale of the block using real dimensions, block length (l) and width (w), which increased as the distance increased (d). This helped us achieve a scaled image. In other words, it helped us to recreate the image to appear as if it was shot from a specific distance. The graph in Figure 20 helped

us determine the block area with respect to the distance from which the photo was taken. The graph shows the exponential growth of the real area (cm^2) when the distance increased [38].

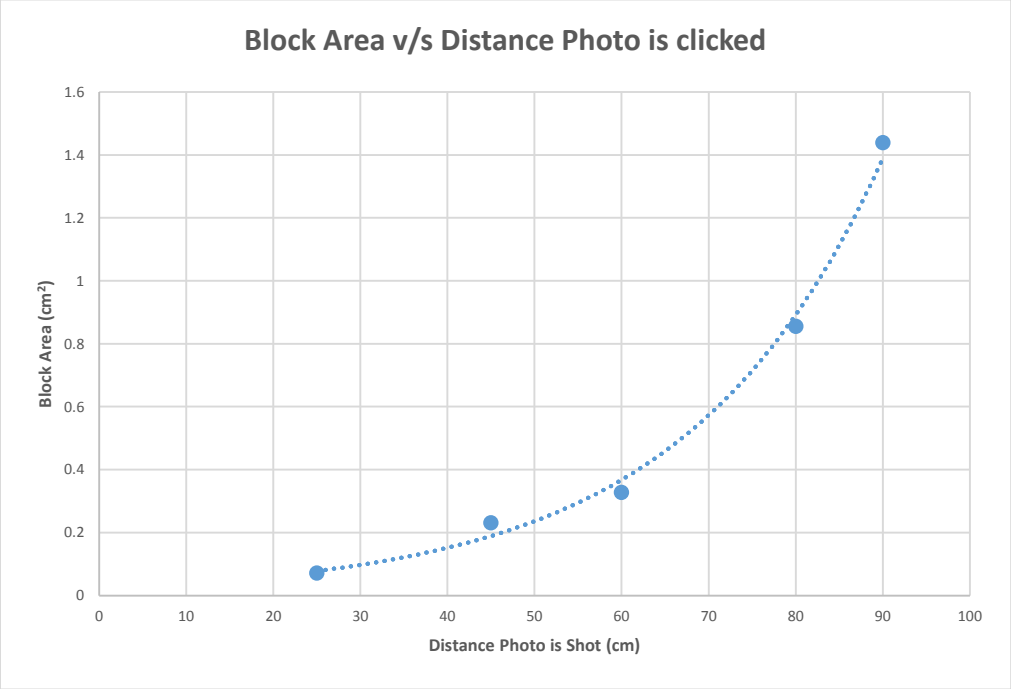


Figure 20: Relation between block area and distance photo is clicked [38]

If an image was taken from a distance of 60 cm, then the area of the blocks would be 0.328 cm^2 and the dimensions of the block would be 0.478 cm in width and 0.687 cm in length. Similarly, if an image was captured from a distance of 80 cm, then the area of the block would increase to 0.855 cm^2 and the dimensions of the block would increase to 0.687 cm in width and 1.1 cm in length, as shown in Figure 20.

Thus, the proposed method helped us to standardize the dimensions of the target image, irrespective of the distance from which the image was taken. When we calibrated the target image and resized it, we then processed the image to obtain the weight of the food item. In part of processing the image, our goal was to obtain the area (in pixels) and the perimeter (also in pixels) of the food portion. To acquire the area of the food portion, we detected the edges of the food portion. We used the Canny edge detector algorithm [37] to calculate the edges of the food portion, which further helped us to compute the closed area and the arc length (perimeter) of the food portion [38].

Chapter 6

Results

6.1 Calorie Computation Results

We proposed two models to compute the number of calories in the food objects on the plate (as described in section 5.4. Object recognition, however, was critical in achieving accurate calorie results. Based on the food type that is recognized by deep learning or the SVM classification algorithm, the system automatically starts calculating the number of calories in the food object.

6.1.1 Distance measurement results

As explained in section 5.2.3, the system uses the accelerometer and the magnetic field sensors to gauge the angle at which the mobile phone is positioned. This angle and the height of the user will further determine the distance of the food object kept on the plate in relation to the user's position. The most amazing aspect of the entire process is that the system does this automatically without the intervention of the user.

The distances were again considered during the steps in the calorie computation, in which the block size was considered according to the distance measured. As shown in Table 1, the bread slice was captured from distances of 25 cm, 45 cm, 60 cm, 80 cm, and 90 cm. The system then used the block resize methodology to calculate the number of blocks in each photos. This number is used in the calorie computation. Similar steps were taken to train the system to detect images of other food classes that were captured from various distances.

Table 1: Calculation of Total Blocks in the Bread Slice [38]

Food Object (Distance Photo was Clicked)	Total No of Blocks
Bread (Distance 25 cm)	1620
Bread (Distance 45 cm)	513
Bread (Distance 60 cm)	368
Bread (Distance 80 cm)	160
Bread (Distance 90 cm)	84

6.1.2 Calorie results from distance measurement

We performed a linear regression on the area and perimeter against the real weights of the food objects. This test enabled us to predict the best fitting slope or the regression line. Hence, given the area of the food object, it predicted the corresponding weight (in grams) followed by the number of calories (Cal). Based on the results of the tests, we predicted the number of calories in both scenarios (with perimeter and area). When we compared the number of calories computed in the area, as shown in *Figure 22*, we determined that the R square value or the coefficient correlation square value was 0.9709. This figure was higher than the square of the coefficient correlation value of 0.8412, which was obtained from the

comparison of the perimeter and the number of calories, as shown in *Figure 21*. The R square value amounted to the accuracy of 97% when the comparison was made with the contour area computed from the edges.

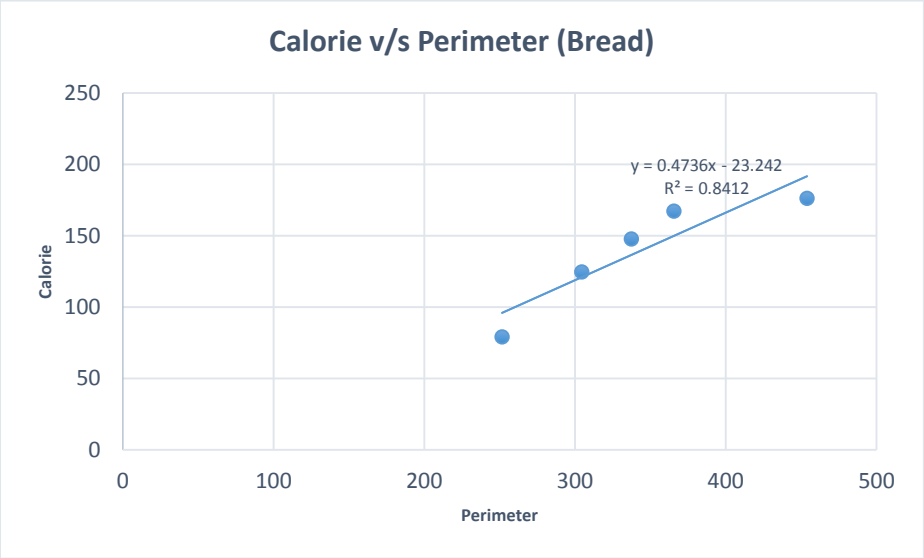


Figure 21: Linear regression on calories and perimeter (bread)

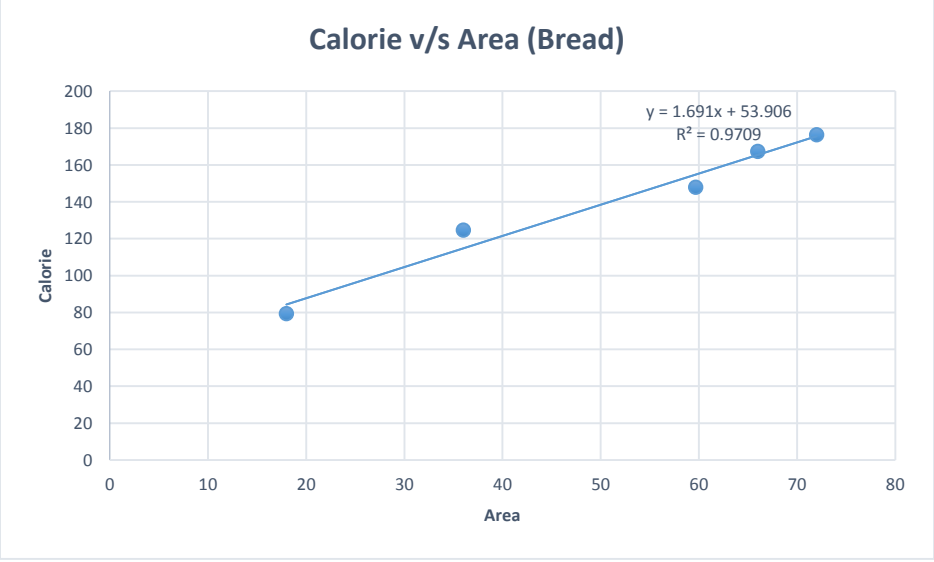


Figure 22: Linear regression on calories and area (bread)

We also estimated the error in computing the number of calories of differently sized bread slices. As **Table 2** shows, we determined the error percentage in estimated calories (C'') from the linear regression on area is minimal when compared to estimated calorie when obtained from perimeter. **Table 2** shows that in some instances, the difference in the number of calories was less than 0.65, compared with the actual number of calories. **Table 2** also shows that the overall error (%) was 3.64%, which was a significant improvement over previous methods of calorie computation, such as 5% in [50].

Table 2: Calculation of Calories in Bread

Area	Weight	Calorie (C)	Calorie '' (C'')	C-C''
66	188	167.32	165.62	1.7
59.66	166	147.74	154.12	-6.38
72	198	176.22	175.57	0.65
18	86	79.21	82.44	-3.23
36	140	124.6	113.63	11

Similar computations were performed on every food class to obtain the best fitting slope for estimating the number of calories. The reason that we considered different base values for each food class is that the computed area of the bread slice might have been greater than the area obtained from the banana, but it did not necessarily corroborate the linear relation between the area and the number of calories. Hence, each food class was scaled individually using different base values for the area.

6.2 Results of Accuracy in Food Recognition Using the Support Vector Machine (SVM) Method

SVM was used in food classification and was compared with the LIBSVM. Both categories were used as tools to classify the food objects.

Software and Development Environments

Table 3: Comparison between Libsvm and Cloud-Based Map in Reducing SVM

LIBSVM in Single Node	**Cloud Based Map Reduce SVM**
Hadoop Version: Hadoop 0.19.1, **SVM Type:** LibSVM-3.17, **Environment:** Eclipse Europa, Cygwin Terminal used for implementing the program on the single node cluster	**Hadoop Version:** Hadoop 1.0.3 **Hadoop Version** LibSVM-3.17 **Environment:** Eclipse Europa Cloud Based Environment: Amazon Web Services (AWS), Amazon EMR, Python, Ruby 1.8.7, RubyGems version 1.8, Amazon EC2, Amazon S3 and Amazon EMR for implementing the Map Reduce SVM program on the multiple node cluster in the cloud environment

For the non-mixed food, we first made three groups containing 1000, 2000, 3000 images, respectively. Second, we kept 1000 images in each group for testing. The system was trained with LIB SVM by using half of the remaining images in each group.

Map Reduce was implemented for the SVM model in the following steps: 1) compute statistics of features (color, size, shape, etc.) and class objects; 2) transform the sample by implementing the SVM model; 3) compute the statistics for the new feature space; 4) distribute the new samples and train the model in random order using the reducer function. The SVM model was implemented in parallel with the Map Reduce mechanism in which each instance was trained with a SVM model. The support vector of each subSVM was taken as the input of the next layer of subSVM [48] [49] [2]. In the cloud-based SVM model, the classification was implemented in parallel across multiple Amazon EC2 instances, which reduced the overall computing time required for large food samples and improved accuracy. These results are shown in *Figure 23*.

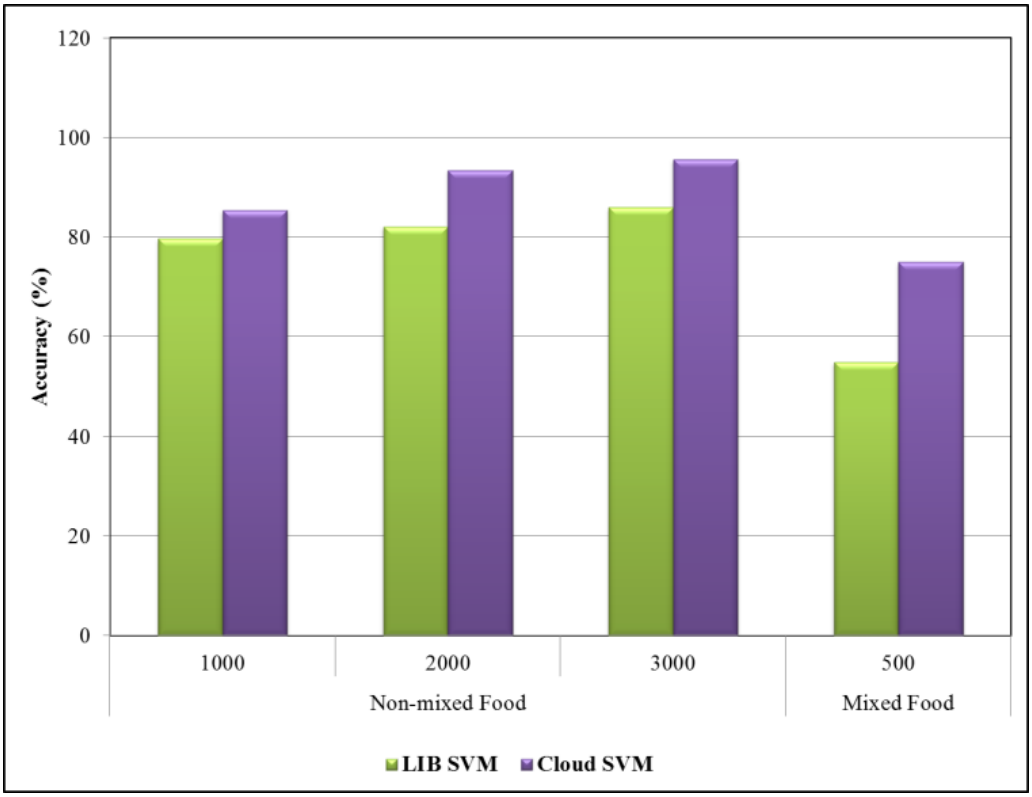


Figure 23: Results based on Cloud SVM and LIB SVM for non-mixed and mixed food objects [49]

The simulation results for non-mixed food are shown in *Figure 23***Error! Reference source not found.**. The Cloud SVM method outperformed the LIBSVM in all image categories. Furthermore, accuracy increased as more images were used in the training phase. We also evaluated LIBSVM and cloud SVM methods on 500 images of mixed food. As shown in *Figure 23*, although the results of the overall accuracy were lower than in the non-mixed food category, the accuracy was approximately 20% greater than that achieved using the LIBSVM approach.

6.3 Food Recognition Results for the Deep Learning Method

We compared the results of various components integrated in our application, such as deep learning, image segmentation, and image processing, with different parameters (accuracy and timing) of cloud servers and a local server connected to the smartphone.

The experimental setup was as follows: We used seven different food classes, each of which contained 40 test images. For each image belonging to a class, we recorded recognition accuracy, recognition success, and the time in seconds taken to process the results to the user.

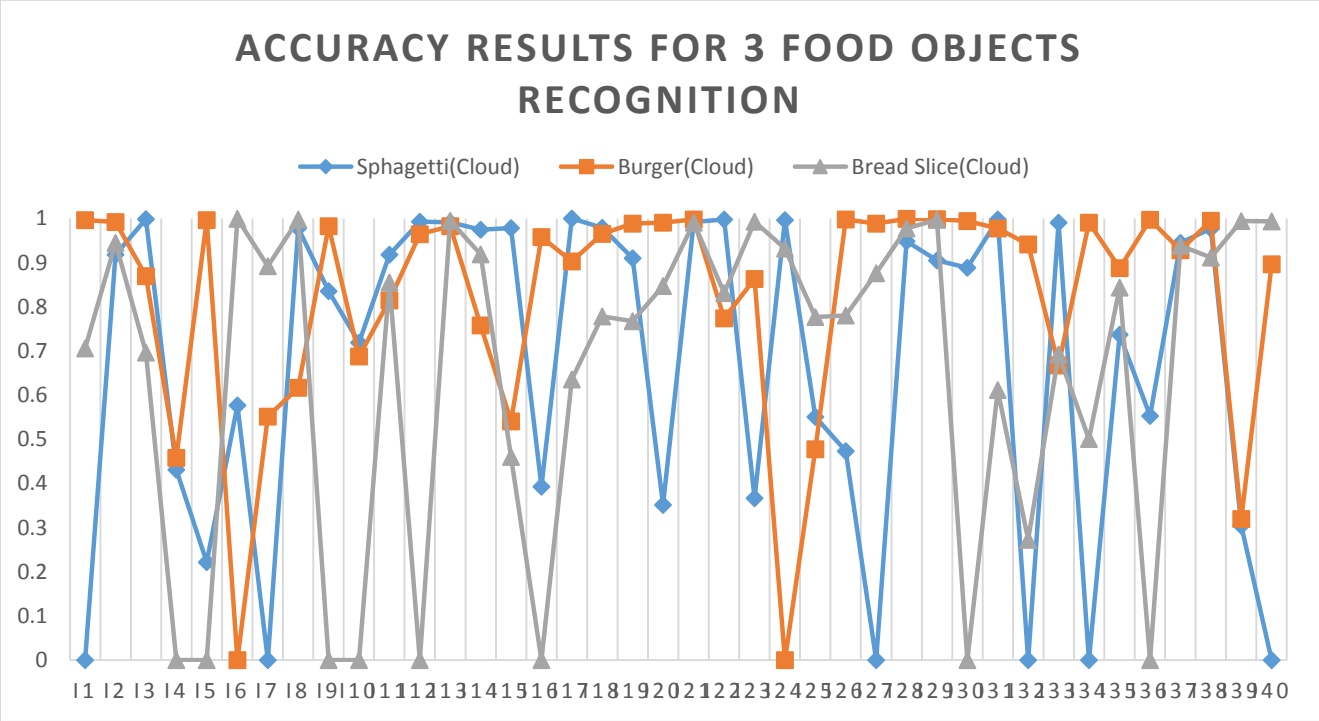


Figure 24: Graph of the recognition of three food objects in 40 images (Images against Probability for Recognition)

Analysis of the Results of Accuracy

Figure 24 shows the accuracy results for three food classes: spaghetti, burger, and bread slice. Accuracy was defined in terms of probability, as explained in section 4.1.1. Hence, it varied between 0 and 1. Spaghetti is a mixed food object. *Figure 24* shows five instances in which the system was unable to classify the food object. Moreover, the overall accuracy for this class had significant variations across all 40 images. As explained in section 5.2.2, spaghetti was classified as a mixed food object because of the various ingredients used in its preparation (e.g., carbonara, Bolognese, etc.) The ingredients of spaghetti include roasted red peppers, olives, garlic, broccoli, tomatoes, and meat (ham, shrimps, chicken, etc.) in varying amounts. Therefore, the accuracy results were not as consistent as the results for the other food objects were. The analysis of the bread slice showed eight instances in which the system was unable to classify this food object. Based on the analysis, we were able to determine that the feature values in the bread were lower than in the other food classes. Bread had fewer points for the texture feature. In contrast, the burger had higher results for food recognition.

Figure 25 shows the accuracy results for four food classes: strawberry, pineapple, cucumber, and banana. The accuracy results for the pineapple were consistent although it was not recognized in one instance. For the strawberry, the average accuracy was 0.941, and all the images were recognized. For the cucumber, the average probability of accuracy was 0.758.

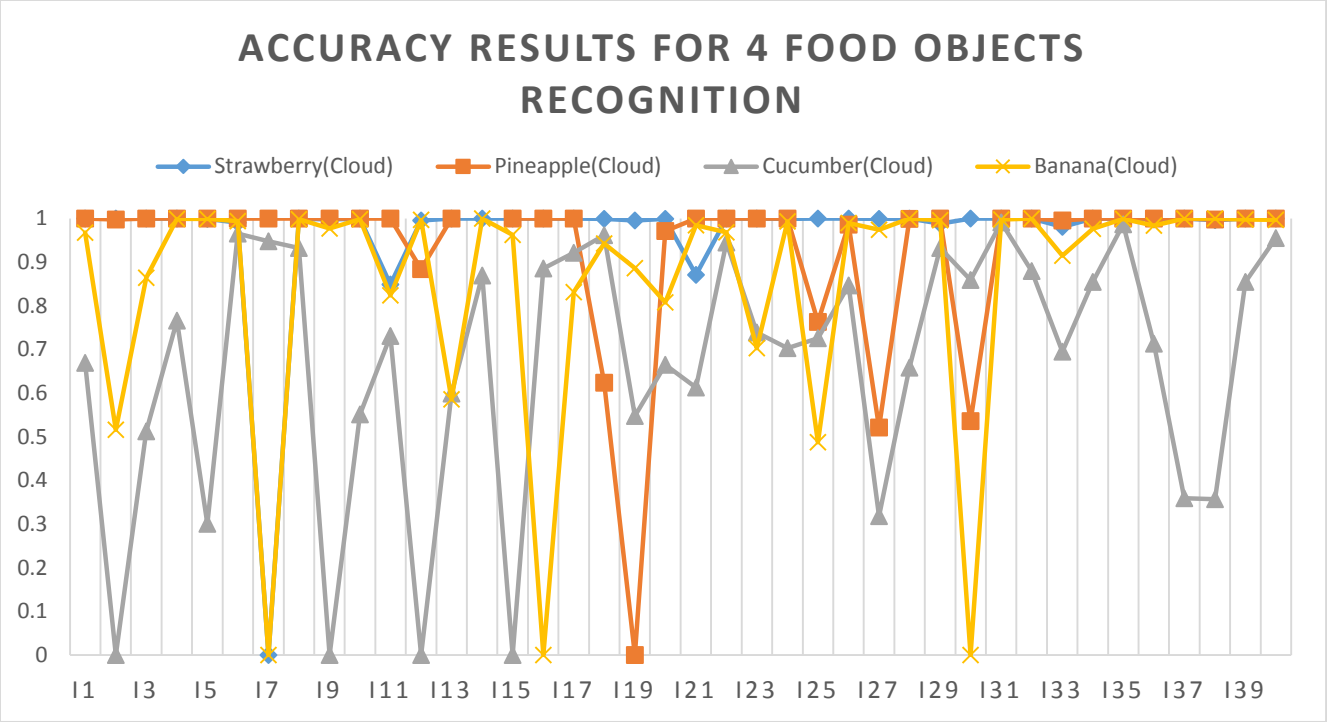


Figure 25: Graph of the recognition of four food objects in 40 images (Images I against Probability of Recognition).

The results show consistent results for accuracy in the mixed food objects. One approach that we used was to train the system to detect each sub-category of the food objects. As in the case of spaghetti, the food object itself was generic in terms of its ingredients; hence, the results were not consistent. If, however, we trained the system to detect the sub-classes of spaghetti, such as carbonara or Bolognese, the features would be different and more precise. A greater number of feature points also played a significant role in the accuracy of food recognition. In the case of the pineapple, the system displayed an accuracy of 0.941, which was better than that for the other food classes.

The timing parameter was also used to analyze our system. The results were critical for our mobile application. Because the user interacts with the mobile application and waits for

the system to process the image, the response time has to be mitigated. To achieve this, we offloaded the processing from the mobile device to the cloud, so different cloud servers handled the processing, which improved the response time.

Table 4: Timing Results for the Seven Food Classes

Results	Food Timing Results Across 3 Cloud Servers and 1 Local Server					
		Instance No	Instance Type	Configuration	Memory	Food Item Avg. Time(sec)
Timing Results for 7 Food Objects over 40 Images	Ubuntu Cloud Server 14.04 LTS (HVM), SSD Volume Type	1	t2.small	(Cloud)2.5 GHz, Intel Xeon Family and 1ECUs	2GB and EBS storage	Spaghetti 17.61 Burger 17.475 Bread Slice 17.55 Banana 17.92 Strawberry 17.6 Pineapple 17.46 Cucumber 16.6
		2	t.medium	(Cloud)2.5 GHz, Intel Xeon Family and 3ECUs	4GB and EBS storage	Spaghetti 15.85 Burger 16.075 Bread Slice 15.28 Banana 16.36 Strawberry 15.7 Pineapple 15.14 Cucumber 15.03
		4	m3.xlarge	(Cloud)2.5 GHz, Intel Xeon Family and 13ECUs	15GB and 2 x 40 (SSD)	Spaghetti 14.62 Burger 14.6 Bread Slice 14.675 Banana 14.68 Strawberry 14.6 Pineapple 14.68 Cucumber 14.63
	Ubuntu Local Server 14.04 LTS (HVM), SSD.	1	Local server.	(Local Server) 1.7 GHz, Intel i5 CPU	1.78 Gb Shared	Spaghetti 24.2 Burger 27.41 Bread Slice 27.09 Banana 26.71 Strawberry 27.39 Pineapple 27.2 Cucumber 28.77

Table 4 shows the timing results for each of the seven food classes described previously. We offloaded the image processing algorithms, such as deep learning, image segmentation and food image processing, to three types of cloud servers on Amazon Web Service (AWS) and compared them with the local serer processing. As shown in **Table 4**, when the processing was done on the local server, the average time taken to process each of the seven food classes was 26.96 seconds, which would have been unacceptable according to the user's expectation of fast results. However, by offloading the image content to cloud-based Amazon EC2 instances, we were able to reduce significantly the average response time to 17.45 seconds, an improvement of almost 35 %.

As shown in **Table 4**, the timing results further improved with the use of the heavy configuration cloud servers, t.medium and the m3. xlarge both of which had higher instances and memory compared to Amazon EC2. The average timing was 16.54 seconds when the food based image processing was done on the t.medium, which was an improvement of 5.21% over the t2.small instance. The average timing of the m3.xlarge instance was improved by 8.82%. The timing results of the latter were 15.911 seconds compared to the single instance cloud server.

6.4 User Interface Snapshots

The figure below shows user interface snapshots taken with our mobile application (Eat Healthy Stay Healthy). The results shown in **Table 5** are based on the three food classes: banana, burger and bread slice, which serve as examples of the classifications of other food objects.

Table 5: Calorie Results of Mobile Application (EHSH) for Three Food Classes

Food Class	Upload Photo Activity	Calorie Result Activity
Calorie Computation of Banana by the Mobile Application (EHSH)		
Calorie Computation of Banana by the Mobile Application (EHSH)		
Calorie Computation of Banana by the Mobile Application (EHSH)		

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we proposed a new approach for image processing technique used to recognize food objects. We also proposed a method for the fully automatic and user-friendly calibration of the dimensions of food portions. These dimensions were used to measure the weight of food portions and the number of calories they contained. This thesis incorporated various methodologies to develop the proposed system, which aimed to achieve highly accurate image recognition and calorie computation.

We also proposed a new method for measuring the number of calories in the food object and discussed some improvements over our previous work in measuring the number of calories. To compute the number of calories in the food object, we went beyond the existing finger-based calorie calibration method to develop a system that automatically computes the distance between the user and the food object. By determining the right combination of mobile and cloud computing, we obtained the estimated distance from the mobile device and used it to process the image in the cloud. This enabled us to process all the images on the same scale and to determine the calorie value of the food object. By using deep learning, we were able to extract the features of the food object and classify it accurately. We also examined the relation between the area and perimeter of the food object with the calorie

value. The findings showed that area was more accurate in determining the calorie values of the food objects used in this study.

The classification methodologies discussed in this thesis stressed the deep learning model, which was proposed specifically for the purpose of food recognition. We discussed the reasons for using the deep learning model instead of the Support Vector Machine (SVM), which we used in previous works. With the use of deep learning model, we were able to improve accuracy with respect to food recognition.

We addressed various aspects of food recognition, from single food objects to multiple mixed food objects, and we devised a methodology that would enable us to address these issues. Because we implemented a unique approach to computing the number of calories in the food object, we had to develop a specific food database that suited the requirements of calorie estimation.

When the number of calories was computed, we focused on the number of calories consumed by the user, and we used an initial calorie assessment to determine the final number of calories consumed by the user. The amazing aspect of this approach is that the entire process is automatic and does not require the user's intervention. We also proposed an approach to enable the system to provide real-time healthy food suggestions based on the type of food object that the user is consuming and his or her historical data. The user's historical data enabled the system to determine both harmful food choices and beneficial food choices.

7.2 Future Work

We believe the real dimensions of the food object could be obtained in real time by using an application on a mobile device. Currently, we are able to obtain the distance from the mobile device in real time. In future research, we will analyze more food classes and confirm the accuracy of the methodology presented in this thesis. By using distance measurement, we could also obtain the width and the height of the food object, which would help us to calculate the area and perimeter of the food object in real-time. This would allow us to determine accurately the number of calories in the food object. We also proposed a method for calculating the number of calories in mixed food objects, which we elaborate in the following section.

7.2.1 Calorie measurement in mixed food (proposed method)

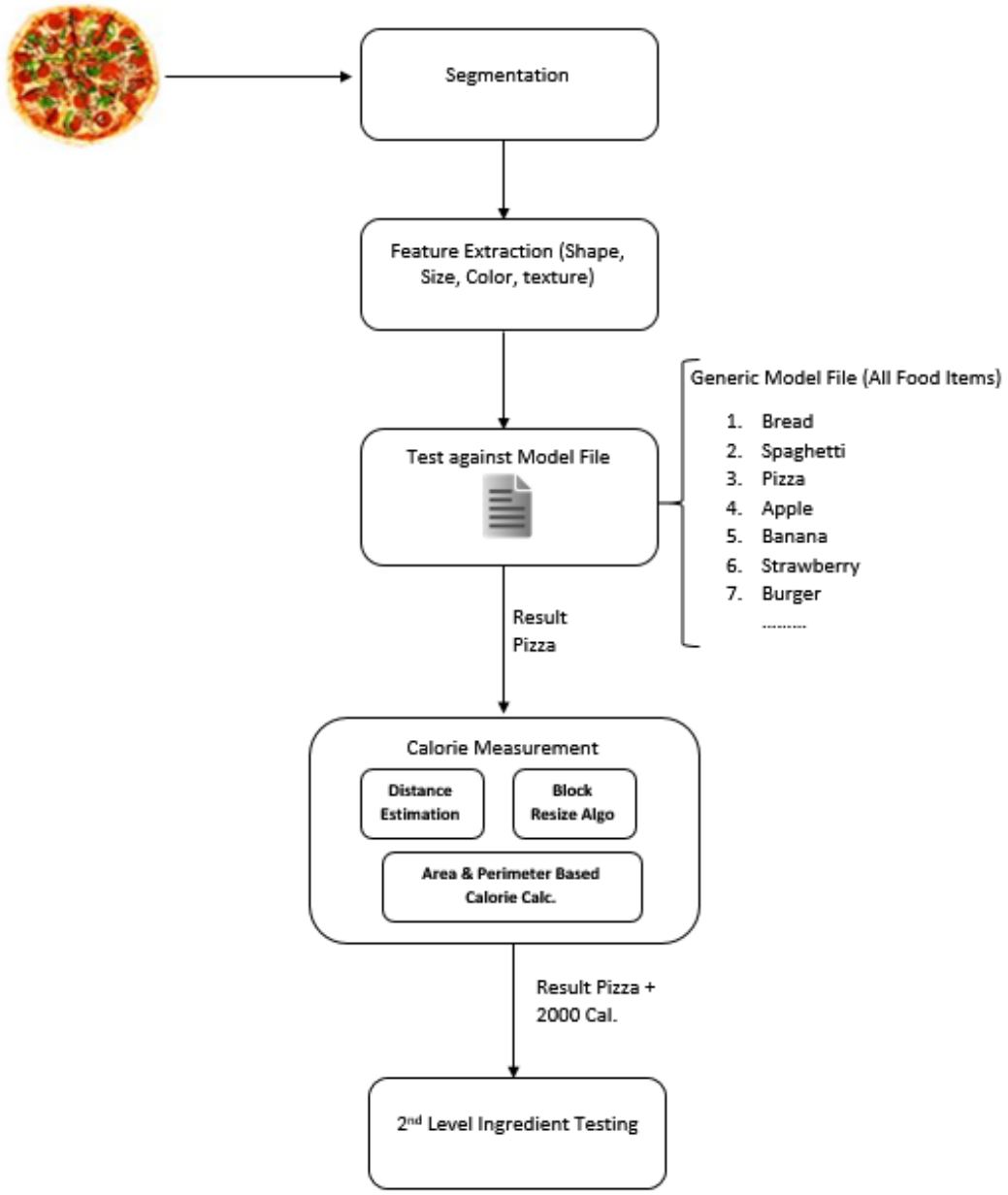


Figure 26: First level of food recognition and calorie computation

In the proposed method for the computation of the number of calories in mixed food objects, the system undergoes two levels of ingredient recognition in the same food object. In the first level, the food object is tested against the model file (which is trained with all the

food objects) to compute the overall calorie value of the food object. In the second level of ingredient recognition, the food object is tested against the model file, which is trained with all the ingredients in the recognized food object (e.g., pizza). As shown in *Figure 26*, the image of the food object (pizza) initially undergoes the image processing steps of segmentation and feature extraction, which are followed by testing it against the generic trained model file that has been trained to detect all the food objects. When the food object is recognized, the steps in calorie estimation are performed. Based on the distance measurement and the block resize method, the image is analyzed to determine accurately the exact size of the food portion and its corresponding calorie value. Once the final calorie value is determined, the image again goes through the second level of ingredient testing, where it is tested against the model file that is specifically trained to detect all ingredients in that food object (in this case, pizza). In the example, the image of the pizza goes through the second level of ingredient testing, as shown in *Figure 27*. It follows the same set of steps shown in *Figure 26*. The only difference is that the model file is trained with a set of images (100 images of each ingredient) of peppers, pepperoni, chicken pieces, mushrooms, etc. When the ingredients are classified, their differing calorie values will be mapped against the values stored in the database. Hence, the results of the second level testing will be the food object and its overall calorie value.

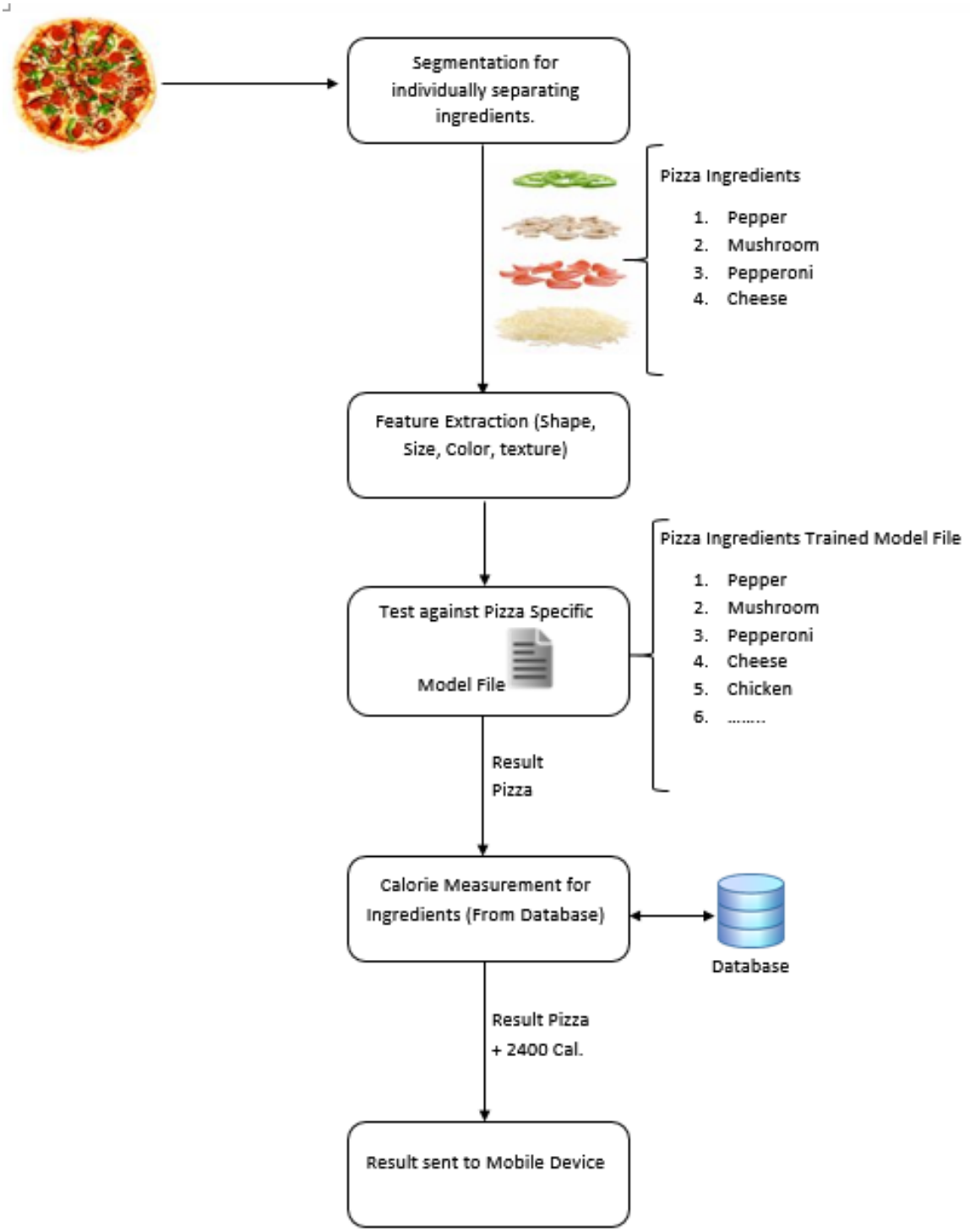


Figure 27: Second-level ingredient testing of the food object (pizza)

References

[1] Pan SY, DesMeules M. Energy intake, physical activity, energy balance, and cancer: epidemiologic evidence. Methods Mol Biol. 2009; 472:191215.

[2] Percik, R., and M. Stumvoll. Obesity and cancer. Experimental and clinical endocrinology and diabetes 117.10 (2009): 563.

[3] Craig, et al. The Okinawan diet: health implications of a low-calorie, nutrient-dense, antioxidant-rich dietary pattern low in glycemic load. Journal of the American College of Nutrition 28.sup4 (2009): 500S516S.

[4] Ford ES, Giles WH, Dietz WH Prevalence of the metabolic syndrome among US adults findings from the third National Health and Nutrition Examination Survey. JAMA 287:356359, 2002.

[5] Albanes D: Calorie intake, body weight, and cancer: a review,Nutr. Cancer, 9: 199-217, 1987.

[6] Waltz E. How I quantified myself. IEEE Spectr 2012 Sep;49(9):42-47.

[7] Wu W, Dasgupta S, Ramirez EE, Peterson C, Norman GJ. Classification accuracies of physical activities using smartphone motion sensors. J Med Internet Res 2012;14(5):e130

[8] Hampton T. Recent advances in mobile technology benefit global health, research, and care. JAMA 2012 May 16;307(19):2013-2014.

[9] Muñoz RF. Using evidence-based internet interventions to reduce health disparities worldwide. J Med Internet Res 2010;12(5):e60

[10] Kratzke C, Wilson S, Vilchis H. Reaching rural women: Breast cancer prevention information seeking behaviors and interest in Internet, cell phone, and text use. J Community Health 2013 Feb;38(1):54-61

[11] Lister C, West JH, Cannon B, Sax T, Brodegard D Just a Fad? Gamification in Health and Fitness Apps JMIR Serious Games 2014;2(2):e9 URL: http://games.jmir.org/2014/2/e9 DOI: 10.2196/games.3413 PMID: 25654660 PMCID: 4307823.

[12] http://www.slideshare.net/ruderfinnuk/ruder-finn-mhealth-report-2012.

[13] Green Paper on mobile Health

[14] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," Artif. Intell. Rev., vol. 29, no. 1, p. 63–92, 2008.

[15] A. Blanco, A. M. Ricket, and M. Martin-Merino, "Combining SVM classifiers for email anti-spam filtering," in Computational and Ambient Intelligence. Proceedings 9th International Work-Conference on Artificial Neural Networks, IWANN 2007, Berlin, Germany, 2007,pp. 903 - 10.

[16] B. Scholkopf and A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge,MA, USA: MIT Press, 2001.

[17] L. Deng and D. Yu. Deep Learning: Methods and Applications. Now Publishers Inc. Jan. 2014.

[18]. Asaf Shabtai and Yuval Elovici : "Applying Behavioral Detection on Android-Based Devices" in the Mobile Wireless Middleware, Operating Systems, and Applications Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 48, 2010, pp 235-249.

[19].Tuan Nguyen, Don Nguyen and Phu Nguyen: “UIT ANPR: towards and open framework for Automatic Number Plate Recognition on Smartphones” published in Proceeding of the 8th International Conference on Ubiquitous Information Management and Communication Article No. 113.

[20] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015. http://neuralnetworksanddeeplearning.com/chap1.html

[21]Nicholas D. Lane, Petko Georgiev,"Can Deep Learning Revolutionize Mobile Sensing?" at HotMobile '15 Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications Pages 117-122.

[22] Ryu N, Kawahara Y, Asami T (2008) A calorie count application for a mobile phone based on METS value. In: Proceedings of 5th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks, San Francisco, pp 583–584. -14

[23] E. Thammasat, “The statistical recognition of walking, jogging, and running using smartphone accelerometers,” in Biomedical Engineering International Conference (BMEiCON), 2013 6th, 2013, pp. 1–4.

[24] Yue Y, Jia W, Sun M. Measurement of food volume based on single 2-D image without conventional camera calibration. Proceedings of IEEE 34th Annual Conference on Engineering in Medicine and Biology; 28 August–1 September; San Diego, CA. 2012. pp. 2166–2169.

[25] Kemp, R.; Palmer, N.; Kielmann, T.; Seinstra, F.; Drost, N.; Maassen, J.; Bal, H., "eyeDentify: Multimedia Cyber Foraging from a Smartphone," Multimedia, 2009. ISM '09. 11th IEEE

International Symposium, vol., no., pp.392,399, 14-16 Dec. 2009 doi: 10.1109/ISM.2009.21

[26] T. Kallonen and J. Porras, “Use of distributed resources in mobile environment,” International Conference on Software in Telecommunications and Computer Networks , vol. 0, p. 281–285, 2006.

[27] L. Bandini, A. Must, H. Cyr, S. Anderson, J. Spadano and W. Dietz, “Longitudinal changes in the accuracy of reported energy intake in girls 10-15 y of age,” The American Journal of Clinical Nutrition, vol. 78, p.p. 480–484, 2003.

[28] W. Luo, H. Morrison, M. d. Groh, C. Waters, M. DesMeules, E. Jones-McLean, A.-M. Ugnat, S. Desjardins and M. L. a. Y. Ma, “The burden of adult obesity in Canada,” Chronic Diseases in Canada, vol. 27, no. 4, p.p. 135-144, 2007.

[29] Y. Kato, T. Suzuki, K. Kobayashi, Y. Nakauchi, "A web application for an obesity prevention system based on individual lifestyle analysis," IEEE International Conference on Systems, Man, and Cybernetics (SMC), p.p. 1718 - 1723, Oct.2012.

[30] Beaulieu, P.; Megherbi, D.B., "A study of the effect of feature reduction via statistically significant pixel selection on fruit object representation, classification, and machine learning prediction," Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2014 IEEE International Conference on , vol., no., pp.82,87, 5-7 May 2014 doi: 10.1109/CIVEMSA.2014.6841443.

[31] T. Miyazaki, G.C. De Silva, K. Aizawa, "Image-based Calorie Content Estimation for Dietary Assessment," IEEE International Symposium on Multimedia (ISM), pp.363-368, 5-7 Dec. 2011.

[32] Shulin Yang; Mei Chen; Pomerleau, D.; Sukthankar, R., "Food recognition using statistics of pairwise local features," Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on , vol., no., pp.2249,2256, 13-18 June 2010

[33] Wen Wu; Jie Yang, "Fast food recognition from videos of eating for calorie estimation," Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on , vol., no., pp.1210,1213, June 28 2009-July 3 2009 doi: 10.1109/ICME.2009.5202718.

[34] L. Bottou and Y. LeCun. Large scale online learning. In Proceedings of Neural Information Processing Systems (NIPS) . 2004.

[35] Krizhevsky, A., Sutskever, I., and Hinton, G. on ImageNet classification with deep convolutional neural networks. in NIPS2012.

[36] N. Srivastava and R. Salakhutdinov, and ldquo, Multimodal Learning with Deep Boltzmann Machines, Proc. Neural Information and Processing System, 2012

[37] John Canny, “A computational approach to edge detection“, Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI - 8(6):679–698, Nov. 1986.

[38] Pallavi Kuhad, Abdulsalam Yassine and Shervin Shirmohammadi, “Using Distance Estimation and Deep Learning to Simplify Calibration in Food Calorie Measurement” in Computational Intelligence and Virtual Environments for Measurement Systems and Applications [CIVEMSA]-2015, Shenzhen, China.

[39] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015. http://neuralnetworksanddeeplearning.com/chap2.html.

[40] G. E. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," in Technical report 2010-003, Machine Learning Group, University of Toronto, 2010.

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

[42]http://developer.android.com/reference/android/hardware/SensorEvent.html#values.

[43] Pallavi Kuhad, Sri Vijay Bharat Peddi, Parisa Pouladzadeh, Abdulsalam Yassine, and Shervin Shirmohammadi, “Mobile Cloud Based Food Calorie Measurement” in the 4th International IEEE Workshop on Multimedia Services and Technologies for E-health. (MUST-EH 2014) July 14, 2014 Chengdu, China, conjunction with IEEE ICME 2014.

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

[45] http://libccv.org/doc/doc-convnet/ ◊ For Libccv library for Deep neural network

[46]

http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=contourarea#contourarea

[47]

http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=arclength#arclength

[48] Parisa Pouladzadeh, Sri Vijay Bharat Peddi, Pallavi Kuhad, Shervin Shirmohammadi, “A Map Reduce Parallel Classifier for Cloud Based Food Recognition” in International Conference on Next Generation Computing and Communication Technologies [ICNGCCT]-2014, Dubai. pages 142-147.

[49] Sun, Zhanquan, and Geoffrey Fox. "Study on parallel SVM based on MapReduce." In International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 16-19. 2012.

[50] Rana Al-Maghrabi, “Measuring Food Volume and Nutritional Values from Food Images” in University of Ottawa, (http://www.ruor.uottawa.ca/handle/10393/26287).