An Intelligent Cloud-Based Data Processing Broker for Mobile e-Health Multimedia Applications

Sri Vijay Bharat Peddi[1], Pallavi Kuhad[1], Abdulsalam Yassine[1], Parisa Pouladzadeh[1]
Shervin Shirmohammadi[1,2], Ali Asghar Nazari Shirehjini[3]

[1] DISCOVER Lab, Department of Electrical Engineering and Computer Science, University of Ottawa
[2] College of Engineering and Natural Sciences, Istanbul Şehir University, Turkey
[3] Sharif University of Technology, Tehran, Iran

Abstract

Mobile e-health applications such as pulse-rate detection, blood pressure monitoring, calorie measurement, activity tracking, and others are in great demand especially in countries where shortages in qualified healthcare professionals is a challenge confronting healthcare providers. Mobile e-health applications provide users and healthcare practitioners with an insightful way to check users/patients' status and monitor their daily activities. Our goal is to build a mobile e-health calorie based application that can classify the food objects in the plate and further compute the overall calorie of each food object with reduced response time without affecting the accuracy of the classification and calorie computation of the food object. To achieve reduced response time and improved accuracy, certain heavy computational parts of the application could be offloaded to the cloud, however, existing cloud offloading strategies would not be able to fully utilize the unique characteristics of food objects including: food object categories (single, multiple and mixed food object), food ingredients, food specific features (textures, shape, color and size) that could be integrated into the cloud computing model to provide more accurate and improved time specific results. Also, leveraging increased number of cloud resources is not only expensive, but also not a feasible solution. In this paper, we propose an intelligent cloud-broker mechanism where the food classification and the calorie computation steps are strategically integrated in the cloud model in a manner that it fully utilizes the unique food characteristics. We also propose a dynamic cloud allocation mechanism that takes decisions on allocating and de-allocating cloud instances in real-time for ensuring the average response time stays within a predefined threshold. We further demonstrate various scenarios to explain the workflow of the cloud components including: segmentation, deep learning, indexing food images, decision making algorithms, calorie computation, scheduling management and second level ingredient testing as part of the proposed cloud broker model. The implementation results of our system showed that the proposed cloud broker results in a 45% gain in the overall time taken to process the images in the cloud. With the use of dynamic cloud allocation mechanism, we were able to reduce the average time consumption by 77.21 % when 60 images were processed in parallel.

Keywords: Food Recognition, e-Health Application, Deep Learning, Central Cloud Broker, Mixed Food Recognition, Single Food Recognition, Decision Algorithm, Dynamic Cloud Allocation.

1. Introduction

Mobile devices have become an indispensable gadget for many people, not just as a communication medium, but also as e-health applications to measure and process pulse-rate, blood pressure, calorie measurement, activity tracking, etc. Because mobile phone ownership and both the number and the complexity of health applications are likely to increase, the potential for mobile technology based health interventions to help populations is expanding in ways that previously were not possible [1]. These applications are in great demand especially in countries where shortages in qualified healthcare professionals is a challenge confronting healthcare providers. Mobile e-health applications provide users and healthcare practitioners with an insightful way to determine the users/patients' status and their daily activities such as waking up times, exercising, eating habits, sleeping patterns etc. All these personal activities could be easily tracked and processed with the use of mobile applications. However, there are several technical challenges that encumber wide adoption of e-health mobile applications as described in [21] [28]. Amongst those challenges is the limited processing power of the mobile device. Even with the enhanced capabilities of today's smartphones, like more and improved mobile sensors, better processing capacity in terms of quad cores and octa core chipsets, more physical storage and RAM, and intensive data processing, e-health applications can consume the resources of a mobile device over its limits, making the application unfeasible or unacceptable, from a quality of experience perspective, to run on the mobile device. This is especially true for multimedia e-health applications that use images, video, computer vision, or other computationally intensive media. To overcome this limitation, certain heavy computational parts of the mobile application could be offloaded to the cloud, which has the necessary processing power and resources to provide satisfactory performance for the mobile application and its features. But, the decision of offloading computationally intensive tasks to cloud instances is not trivial. Cloud resources must be allocated using an intelligent mechanism that dispatches the processing tasks in real-time while ensuring the minimum processing time possible. Furthermore, the cost of allocating cloud resources increases as the number of instances increases, hence, the mechanism should be able to dynamically free the resources when the average processing time of the resources stays within a predefined threshold.

In this paper, we propose an intelligent cloud-based data processing broker mechanism for mobile e-health multimedia applications, and we demonstrate its feasibility and performance by integrating it with our specific e-health mobile application, Eat Health Stay Health (EHSH). EHSH includes food image processing, deep learning for food recognition and classification, and calorie estimation. These data processing functions are computationally intensive and require resources that most current mobile devices cannot provide. To overcome the computationally intensive resource problem, offloading a portion or the application entirely to cloud would take the load away from mobile device and improve the performance and resource consumption of the mobile device [5] [6]. To overcome the computationally intensive resource problem, offloading a portion or the application entirely to cloud would take the load away from mobile device and improve the performance and resource consumption of the mobile device [5] [6]. However, in our mobile e-health application, the algorithm includes certain unique food characteristics which, if efficiently utilized would be improving the overall system performance, achieving scalability as well as ensuring optimal use of cloud resources. The unique food characteristics include mainly the food object category, food ingredients and food specific features. To incorporate these food characteristics in the cloud model it is important to understand why and how these characteristics can help improve the performance of our system. Analyzing the food category characteristics, we observe that there are mainly three food categories that we look to incorporate: Single, multiple and mixed food objects. Out of these three, the segmentation of single food objects would result in one food object which would further be processed by the cloud broker to classify. Then, it is followed by calorie computation, which contributes

to overall time taken to process the single food image (T_s). Multiple food objects on the other hand can include more than two objects on the plate and hence segmentation of this will result in X number of images which would have to be processed to classify and compute the number of calories for each of the X food items. Hence, the overall time to compute for multiple food objects (T_m) would be $T_m = \sum_1^X t_d + t_c$, where t_d is the time taken to classify the food object using deep learning algorithm and t_c is time taken to compute the calorie. Considering the total multiple food objects m > 1, then $T_m > T_s$. In case of mixed food objects, we propose a second level ingredient algorithm in section 4.2 which results in two levels of deep learning algorithm. Hence, the overall time taken to process the mixed food object T_{mx} will be greater than T_s due to the additional second level of processing. Taking the overall time into consideration it is essential to include a scheduling and decision making algorithm that takes these food categories in an independent queue and processes them accordingly.

Our intelligent decision making algorithm of the proposed cloud-based broker decides on how to distribute the processing of the aforementioned functions based on a number of factors such as the number of active user connections, statistics of the broker (CPU utilization, memory utilization, queue status, etc.), and historical data such as the average time needed for running the deep learning algorithm for a single food object. The aim of our proposed broker is to significantly improve the scalability of the mobile application by introducing queuing management, indexing food images, allocating cloud resources, and other components that would smartly manage multiple food images at the same time, and provide more accurate calorie estimation for the food image sent by a specific user.

The main contributions of this paper are as follows:

- We propose an efficient cloud broker model that incorporates intelligent mechanism to decide where and when to offload the processing of food images to cloud instances. This is rather significant for our mobile e-health application given the complex nature of food images. The processing of these images involves computationally intensive tasks such as segmentation algorithms and deep learning for food object recognition. The broker uses decision making algorithms to collect performance metrics at regular intervals allowing the broker to make the choice of cloud instance to offload the computation tasks.
- We propose an intelligent scheduling mechanism which allows the broker to provide optimal utilization of the cloud resources and improve the overall response time of processing the food images. The implementation results of our system show that the proposed mechanism results in a 45% gain in the overall time taken to process the images in the cloud.
- We proposed a dynamic cloud allocation mechanism to handle parallel image processing requests from different users. By introducing this mechanism, the system is able to automatically add the cloud instance in real-time by gauging the system resource: number of available cloud instances, the queue length and the total number of incoming parallel food processing requests for achieving the pre-defined threshold value for the response time. Also, by deallocating the cloud instances when not in use, the system is able to avoid under-utilization of cloud resources. This will ensure that the system will not add additional resources exponentially, but rather gauge the requirement and accordingly take decision to add and remove the cloud resources. This is particularly important and unique because it ensures scalability when handling multiple mobile device application requests and different food category recognition requests. The proposed dynamic allocation mechanism along with the queue management strategy would smartly manage multiple food images, at the same time, and maintain the consistency of the system. Our results show that we were able to reduce the average time consumption by 77.21 % when 60 images were processed in parallel

- We implemented the system and tested scenarios of scalability in terms of the number of users connecting to the cloud. Our system shows consistent calorie measurements results even when we increasing the number of food objects. From the user perspective it is important to provide a single correct result; otherwise scenarios of incorrect sequence of the recognized food objects or the calorie measurement results will adversely impact the user acceptance of the application.

The rest of the paper is organized as follows: Next section provides background information about our application EHSH followed by the related work in section 3. In section 4, we present the design details and the architecture of the proposed mechanism. . In section 5, we discuss the proposed cloud broker and its various components, followed by Implementation Results in section 6 and lastly section 7, which includes the conclusion and future work.

2. Background on EHSH

EHSH is able to recognize a food object, the image of which is captured by the user's smartphone, and can estimate the total number of calories in the food object. For recognizing the food portions in the meal, we proposed two approaches in [2][3] and [4]. In our first approach [2], we implemented a Support Vector Machine (SVM) based parallel classifier, which was implemented in parallel in multiple cloud instances. In our second approach, we used deep learning [4], which resulted in higher accuracy as compared to [2][3]. With the use of deep learning, the information of each pixel in the image was stored in different neurons, which were further connected to each other in the network. The network comprised of several layers, and each layer contained a set of neurons. The part that made the deep learning network different compared to other neural networks, was the inclusion of multiple hidden layers in the system.

In our implementation, we made sure the image processing steps are performed in the cloud, in order to remove the overhead of processing from the mobile device [4]. For computing the calorie value of the food, we proposed two different techniques, each of which was running in the cloud too. For the first technique in [2][3], we utilized the user's finger as a calibration reference for computing the size of food portions and their ensuing calorie amounts. The user had to place his finger near the food each time the photo was captured. For the second technique, we proposed an auto-calibration approach [4] that did not require the use of a reference object to calculate the food dimension and its ensuing calorie content.

From the above, we can see that food recognition and calorie computation are essential components of EHSH. However, due to the resource constraints in the mobile device, implementing them in a smartphone was not viable option, since the response time was very high, and the application consumed a large percentage of the smartphone's resources, limiting the usability of the application and the smartphone. To overcome these challenges, we used the cloud to offload a bulk of the processing tasks .. In [2][3], we used MapReduce as a parallel classifier in the cloud to implement SVM on each of the cloud instances. As part of the SVM implementation in Amazon Web Services (AWS), we performed the map and reduce based training and testing tasks on multiple EC2 instances. Although the MapReduce model yielded the desired results for larger training sets, the initial time taken to distribute the data and process it across different Amazon EC2 instances was considerably higher for smaller training sets of data. Based on the results in [6], we observed that with the increase in cloud instances (from a t2.micro one cloud instance to an x3.large four cloud instances), the overall response time got reduced. In a scenario where higher number of users is accessing EHSH in parallel, adding more cloud instances on demand would suffice the demand for cloud resources, but the initial time taken to allocate the cloud instances would adversely affect the overall response time. Hence the scalability aspect was still a challenge that had to be addressed.

To address this challenge, we proposed a decision making mechanism [5] that would efficiently utilize the cloud resources. As shown in Figure 1, when multiple users are accessing EHSH in parallel, the system first performs segmentation to gauge the number of food objects on the plate. Based on the total number of food objects, the decision mechanism will allocate the segmented food objects to the remaining cloud instances.

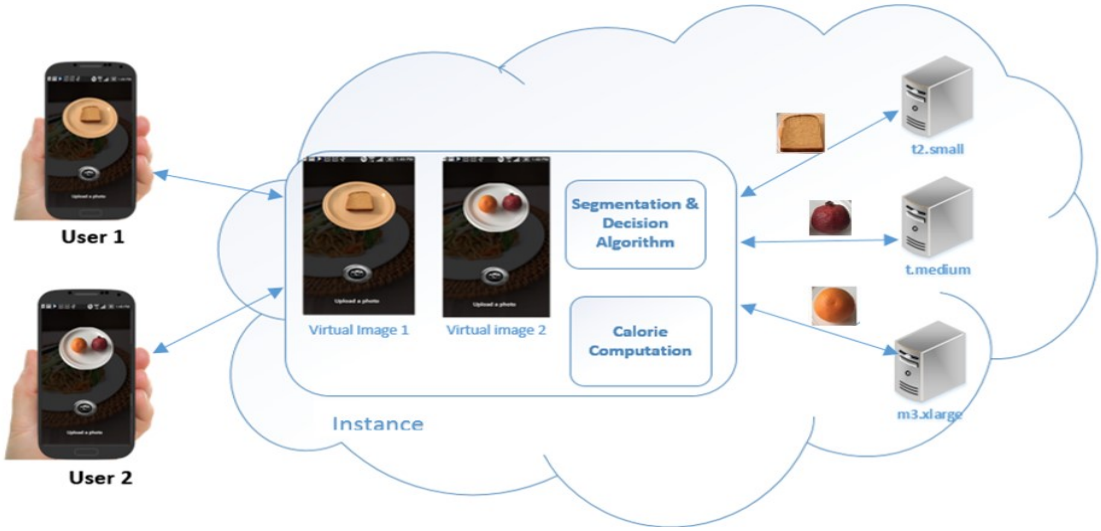

Figure 1: Intelligent Decision Making Mechanism [5]

However, the decision making algorithm in [5] has two weaknesses: 1- it uses performance checks and needs to be further optimized to efficiently allocate the food images to the remaining cloud instances, and 2- it needs to be further extended for each food category of single (e.g. apple, banana, pineapple, orange etc.), multiple (rice with chicken and salad on the same plate) and mixed food objects (pizza with mixed ingredients like pepporoni, chicken, pepper, olives etc.). Each of these food categories has different processing requirements. For example, single food items need two stages: the deep learning (feature extraction and classification) stage and the calorie measurement stage. On the other hand, multiple food has to initially be segmented to obtain the food objects from the plate, and then perform deep learning and classification on each of the segmented food images and finally integrate these results to give a single calorie output to the user. Similarly for mixed food objects, there are additional steps introduced to firstly recognize the food object with deep learning and further recognizing the ingredients of the food objects with second level of deep learning and calorie measurement steps. The processing of these food categories was not considered in previous work as the decision algorithm was only designed to dispatch the tasks based on the total number of food objects. Since for each food category the steps involved for food recognition and calorie measurement are different, the cloud broker must employ a completely new decision algorithm to accommodate those differences efficiently while offloading the processing to other cloud instances.

In this paper, the cloud broker and the newly proposed decision algorithm are completely different from our work in [4][5][6]. The objective of the new design is twofold: The first is to handle the processing tasks of each food category while ensuring efficient utilization of the cloud resources. The second is to reduce the overall response time without having to add new cloud resources in case of non-dynamic cloud allocation model. The concept proposed as part of this paper specifically focuses on the scalability aspect

of the system, taking into account each of the 3 food categories while trying to address the issue. This is entirely different from [5] and [6] which focused on the cloud virtualization where the goal was improving the application performance of the mobile application by virtually swapping the mobile session with the cloud session.

3. Related Work

Recently, because virtualization techniques enable cloud computing environments to remotely run services for mobile devices, computation offloading has attracted significant interest from researchers [16]. Computation offloading extends a mobile device's capabilities when running intensive computational services such as e-health apps. However, seamless computation offloading from mobile devices to the cloud is not trivial and requires optimal offloading and scheduling decisions. These decisions must take into account certain parameters such as computation capability, server loads, response time and the amounts of exchanging data between mobile devices and cloud servers [17]. In this section, we investigate some of the previous studies in this area that are closely related to our work.

The work presented in [18] proposes two offloading schemes for mobile health applications called self-reliant multi-cloud offloading system and multi-cloud offloading system. The study shows that self-reliant multi-cloud offloading systems can provide stability but with high communication cost, while multi-cloud offloading systems reduce communication cost but are less stable. But the study does not provide any mechanism regarding the offloading decision and the performance of the system is not analyzed. The work proposed by [19] introduces a task-level adaptive framework for real-time streaming of data for healthcare applications. Although the study assumes that some decision mechanism is in place for offloading, it focuses only on the usage effectiveness of cloud servers. The work in [20] proposes a Primary Node-based architecture for typical telehealth service using the cloud, which takes into account both storage and delivery efficiency. The study presents an algorithm to predict and allocate future bandwidth of all virtual machines during urgent e-health services. The authors in [21] propose a scheme to process part of the data processing locally on the node (e.g. mobile device), while other parts could be offloaded to the cloud. In the proposed design, the authors consider computation offloading from the cloud to the wireless nodes (and vice versa) as an effective mechanism for energy optimization. The study considers mobile e-health and proposes a multi-layered optimization approach that combines computational offloading with optimization mechanisms.

There are also several studies proposing computational offloading schemes that are relevant, although they do not target e-heath applications specifically. For example, the work in [11] makes use of the Ibis client-server infrastructure, wherein the computation intensive work is offloaded from the Android smartphone to the cloud. MAUI [7] makes use of the Virtual Machine (VM) Migration and the traditional client server model, wherein the computation part from the mobile device is offloaded by program partitioning. The problem with this approach is that a smaller programming model is offloaded, which puts additional burden on the programmer if the entire application has to be offloaded. In [8] they propose a transient customization of cloudlet infrastructure using hardware VM technology wherein A VM cleanly encapsulates and separates the transient guest software environment from the cloudlet infrastructure's permanent host software environment. With [8], the overall response time consists of the time it takes for the VM to start and stop, each time a request is made to the cloud server. Both [9] and [10] are client-server Android based models, but differ in their implementation. The work in [10] is implemented as a map-reduce system where the computing intensive tasks are offloaded to other mobile devices, while [9] is implemented as a single client server cloud system.

Most of the above work in the area of decision making algorithm focus on deciding whether or not to offload the bulk of processing to cloud. As compared to above models, our model focuses on the decision mechanism in the cloud, to decide on which cloud instance to offload depending on the performance of the cloud and further ensuring the consistency of food categories as described in subsection 1.1. Although the majority of the related work focus on the general concept of mobile cloud computing and its wide application in the area of decision application, there is still a need of a decision making mechanism specific to mobile e-health and its integration with various food categories (single, multiple and mixed food) since the steps for food recognition and calorie measurement are different for each food category.

Parallel to the above work, there are several studies that focus on the scalability aspect of processing tasks in the cloud. For example, Ranabahu et al. [27] identified the lack of scaling strategy in Cloud middleware. They propose a horizontally replicating best practice that includes a load balancing layer, an application server layer and database layer [28]. The challenging aspect while trying to handle the scalability aspect in cloud is the limitation of the cloud resources provided by standard cloud service provider. The factor that is common to all cloud based e-health mobile application is the effect on response time of the mobile user while ensuring the system is scalable handling multiple requests. Our work most closely resembles [28], wherein they propose a real-time health monitoring mobile application, which collects ECG data in real-time and performs ECG beat analysis and dissemination of personal ECG health data to patients and health-care professionals. The high processing component in their application was the heartbeat waveform analysis, and in order to address multiple requests they also performed horizontal scalability testing (ability to system to expand resource pool to accommodate heavier load). However, since they are handling only one data category (ECG data) the scheduling mechanism was straightforward and depended on only two factors: Number of requests in the queue and the requests observed over a time span. Also, they do not include the decision making algorithm, as to which cloud instance to dispatch the task to and make use of Aneka [29] for handling scalability requests in there system. On the other hand, we developed a scalability model which includes scheduling and queue management mechanisms which is different for each food category (single, multiple and mixed food). Also we designed two decision algorithms, one for classifying the food categories and the other to classify and allocate/deallocate cloud instances in real-time. The work in [28] observed 40% improvement when the system had 80 parallel requests, as compared to 135.93% improvement in our system for the same set of parallel requests. In [30], another medical healthcare application is proposed that makes use of parallel image processing for image datasets from brain tumor specimens and developed pipeline analysis for study of brain cancers on hybrid cluster. The authors in [30], performed image analysis by breaking the image into rectangular tiles and performing segmentation and feature extraction process on each tile independently. The multi-node parallelization strategy, wherein each process (segmentation and feature extraction), is assigned to different worker nodes. For scheduling strategy, the study adopted Priority and First Come First Served (FCFS) scheduling strategy. For FCFS, the scalability assessment of [30] in terms of execution time had 1.7x improvement when performed over various cloud nodes. Although, we also used similar scheduling techniques (FCFS scheduling) and the scheduling algorithms were specific to various food categories and also used feature extraction and classification based on deep belief networks (DBN) [23]-[26], which are based on the concept of deep learning, allow us to achieve higher accuracy and better generalization, even with small datasets.

In table 1, we summarize the main differences between our work and existing studies in order to highlight our contribution and novelty. We provide the comparison based on: offloading technique, problem addressed, decision making consideration, scalability, and application description and implementation model.

Table 1: Comparisons among Mobile Cloud Computing Models

Methodologies	Implementation	OS	Offload Technique	App Description	Goals	Decision Making	Scalable
Cuckoo [11]	Ibis Middleware	Android, Ibis	Local & Remote Implementation	Image Recognition App	Minimize Energy Usage	Yes	No
MAUI [7]	Program Partitioning	Windows Mobile, Windows 7	Local & Remote Implementation	Face Recognition, Voice Translation App.	Response Time, Energy Saving & CPU Utilization.	Yes	No
Cloudlets [8]	Cloudlets	Nokia with Maemo Linux, Linux OS	Remote Implementation	Linux App(AbiWord, PathFind, SnapFind)	Response Time	No	Yes
GEM Cloud[9]	Client-Server Model	Android, Linux	Remote Implementation	Not Mentioned	Response Time & Energy Consumption	No	No
Hyrax [10]	Hadoop MapReduce	Android	Remote Implementation	Sort & Word Count App.	Response Time	No	Yes
ECG Analysis[28]	Aneka[29] Middleware	All Wearable Devices.	Remote Implementation.	ECG-Heart Beat Waveform Analysis	Response Time	No	Yes
Brain-Tumor Analysis[30]	Multi-Node Parallelization, Keeneland [31]	Not Mentioned.	Remote Implementation.	Brain tumor specimen's analysis.	Response Time, Speedup Estimation, Efficiency.	Yes	Yes
EHSH (proposed)	Cloud Based Virtualization	Android, Android x86, Linux	Local & Remote Implementation (Cloud Broker)	Food Recognition, Calorie Measurement App.	Response Time, Battery & CPU Utilization	Yes	Yes

4. EHSH Design

To understand our cloud broker, it is imperative to first understand how our food classification and calorie measurement system works. In this section, we give a brief explanation of how EHSH detects food and measures their calorie.

4.1 Single and Multiple Food Objects

In Figure 2, we provide a block diagram of our Single and Multiple Food Object recognition and calorie measurement system. The system consists of the following stages: Image acquisition and pre-processing, image segmentation, classification, and calorie measurements.

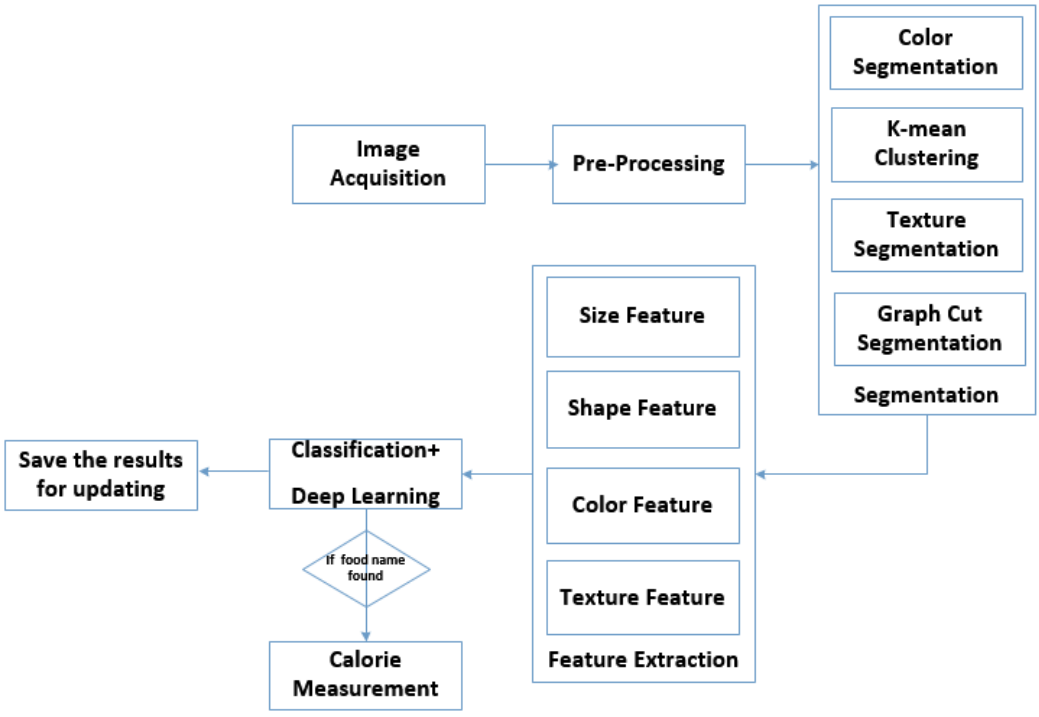

Figure 2: Image Analysis System

When the system receives an image of food, taken by the user's smartphone, it does preprocessing, and then applies different methods such as color segmentation, K-mean clustering, Texture segmentation and graph cut segmentation, described in [15], for segmentation and for extracting various food portions and further saving them as a single food image. Then, it applies image classification to each extracted food portion to find out what food they are; e.g., chicken, fries, rice and cucumber for Figure 3, etc. To ensure the classification is correct, it shows a menu to the user to confirm all detected food portions and to correct them if there are any missed recognitions. Once the user confirms or corrects the food portions, the calorie calculation method will be applied to each food portion and the result is shown on the screen, as shown in Figure 3 wherein each food portion is bounded by Region Of Interest (ROI).

Figure 3: Segmentation for Multiple Food Objects on a Plate

4.2 Mixed Food Objects

For computing calories for mixed foods, the system undergoes two levels of ingredient recognition. In the first level, the food object is tested against a generic model file that is trained with all mixed food objects, to recognize the food as a whole. In the second level, the food object is tested against a model file that is trained with all the ingredients of the specific food that was recognized in the first level. Let's elaborate with an example. As shown in Figure 4, the image of a pizza is fed to the first level, and undergoes segmentation and feature extraction, which are followed by testing it against the generic trained model. When the specific food object is recognized, pizza in this case, the image is analyzed to determine the size of the food and its corresponding calorie value. Then a rough calorie value is determined, 2000 Cal in this example. Then, the image again goes through a 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 ingredients. In this level, the model file is trained with a set of 100 images of each of peppers, pepperoni, chicken pieces, mushrooms, and other pizza ingredients. When the ingredients are classified, their calorie values will be mapped against the values stored in the database. The calorie value obtained from level 1 (in this case 2000 Cal) is an approximate value for the average pizza without considering its ingredients, and the calorie value obtained from level 2 (in this case 2400 Cal) will be more accurate, as it takes into account the ingredients. Hence, the results of the second level testing will be the food object and its overall calorie value, 2400 Cal in this case.

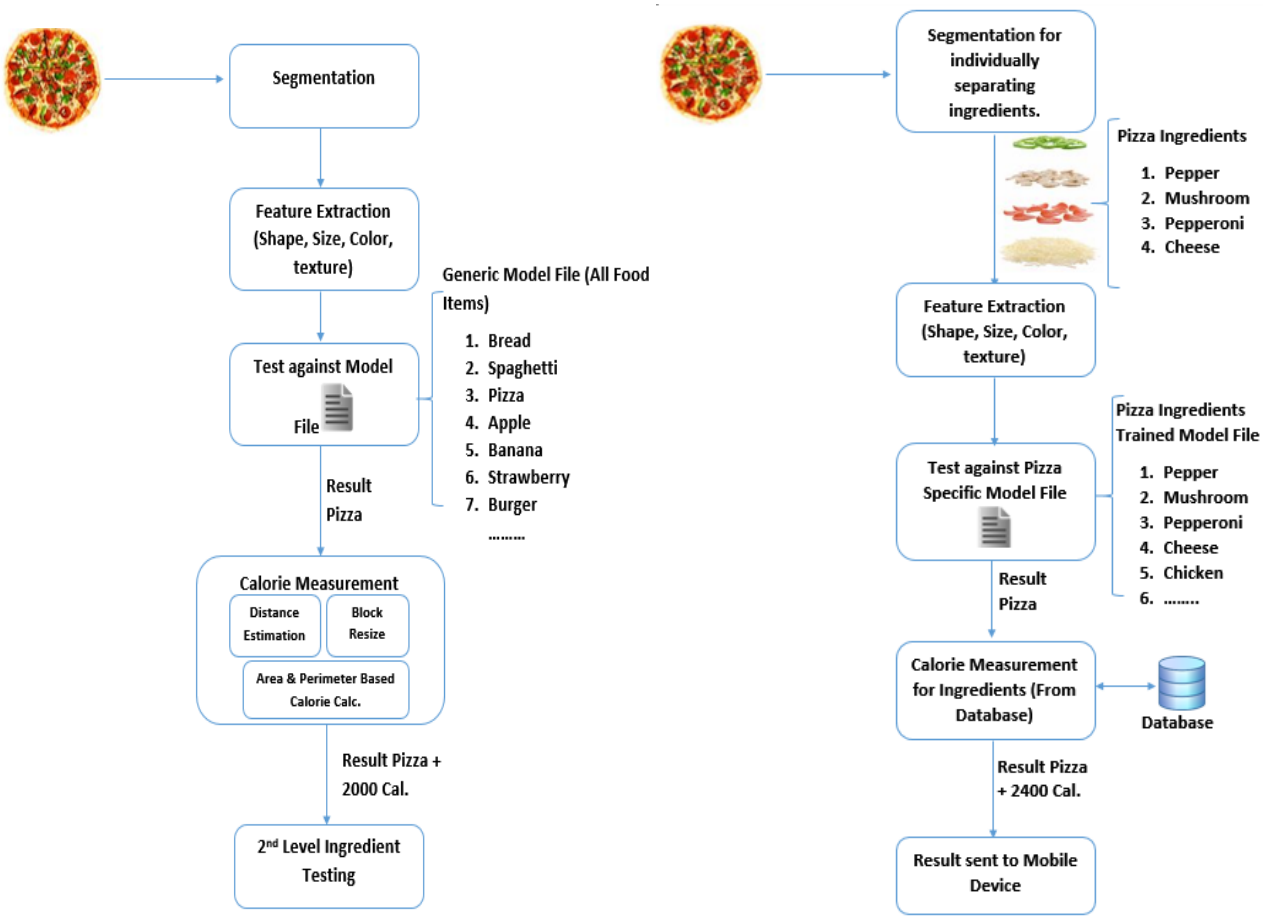

Figure 4: First (left) and Second (right) level ingredient testing of the Mixed Food Objects (pizza) [12]

4.3 Deep Belief Network Based Feature Extraction and Classification

For food image feature extraction and classification, deep belief networks (DBN) [23]-[26], which are based on the concept of deep learning, allow us to achieve higher accuracy and better generalization, even with small datasets. This is a well-known property of deep learning. It further gives us the flexibility to integrate its feature extraction model with other linear classifiers, including SVM. Hence, for EHSH's food recognition part, we use DBN, wherein we initially perform unsupervised training without labelling the images, which further assist the deep learning network to find the suitable parameters (determines the hidden nodes and the edge parameters) that is further used as part of the supervised learning to achieve accurate results. The back-propagation algorithm as part of the supervised learning is used to compute the gradient of the cost function quickly [13]. Training the deep neural network in this way will allow us to make the necessary changes to the weight and bias, from which we obtain the desired results. We label these random training inputs X1, X2, …, Xm. The stochastic gradient descent algorithm then selects a randomly chosen mini-batch of training inputs [14], and trains with those. The weights and biases are further modified to accurately train the system [12]. As shown in Figure 5, DBN is a hierarchical model with many hidden layers in the network. Two hidden layers combine to form the RBM (Restricted Boltzmann Machine) which further form a stack of RBMs. Each layer, including the hidden layers, contain a set of neurons, i^k being the input vectors, h^k being the hidden neuron and O^k being the output neuron of the k^{th} layer in the figure. As part of the feature extraction process, the unsupervised and supervised learning (backward propagation) together generate the desired feature vectors that are used by

the classifiers (feed forward propagation). Hence, we derive the high level features from the low level features further. The prediction is based on the probabilistic model [10], where p(label|n input image) is the probability of the match with the corresponding label set.

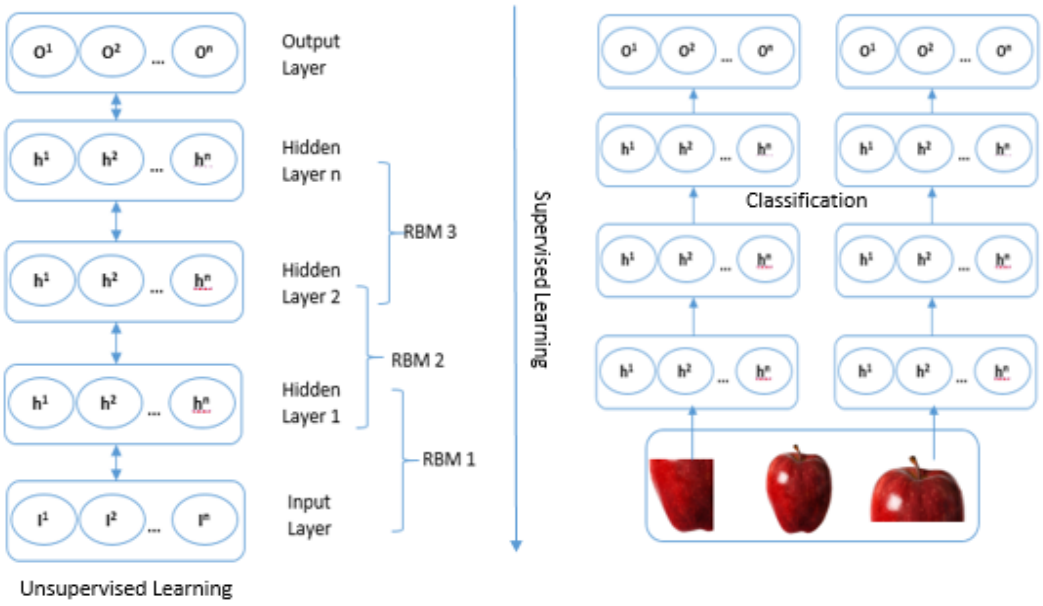

Figure 5: Deep Belief Network Model for Food Recognition

4.4 Calorie Computation

For computing calories, we have proposed an auto-calibration approach in [4] which makes use of the distance between the user and food to calibrate the food dimensions. It further makes use of mobile sensors like accelerometer and magnetic field sensor to compute the distance between the food object and the person capturing the image. This computed distance is used to calibrate the dimensions, without using the user's thumb discussed earlier. The distance value acts like a reference for calculating the block dimension which is further picked from the database in the cloud. The block dimensions themselves are used to calculate the area of the food portion, based on which the volume, hence mass, and finally the calorie value of the food portion is calculated.

5. Proposed Cloud Broker

In this section, we describe the proposed cloud-based broker mechanism for handling single, multiple, and mixed foods. We specifically describe how the broker handles multiple and concurrent requests from a large number of users. The decision making algorithm of the broker is also described in this section.

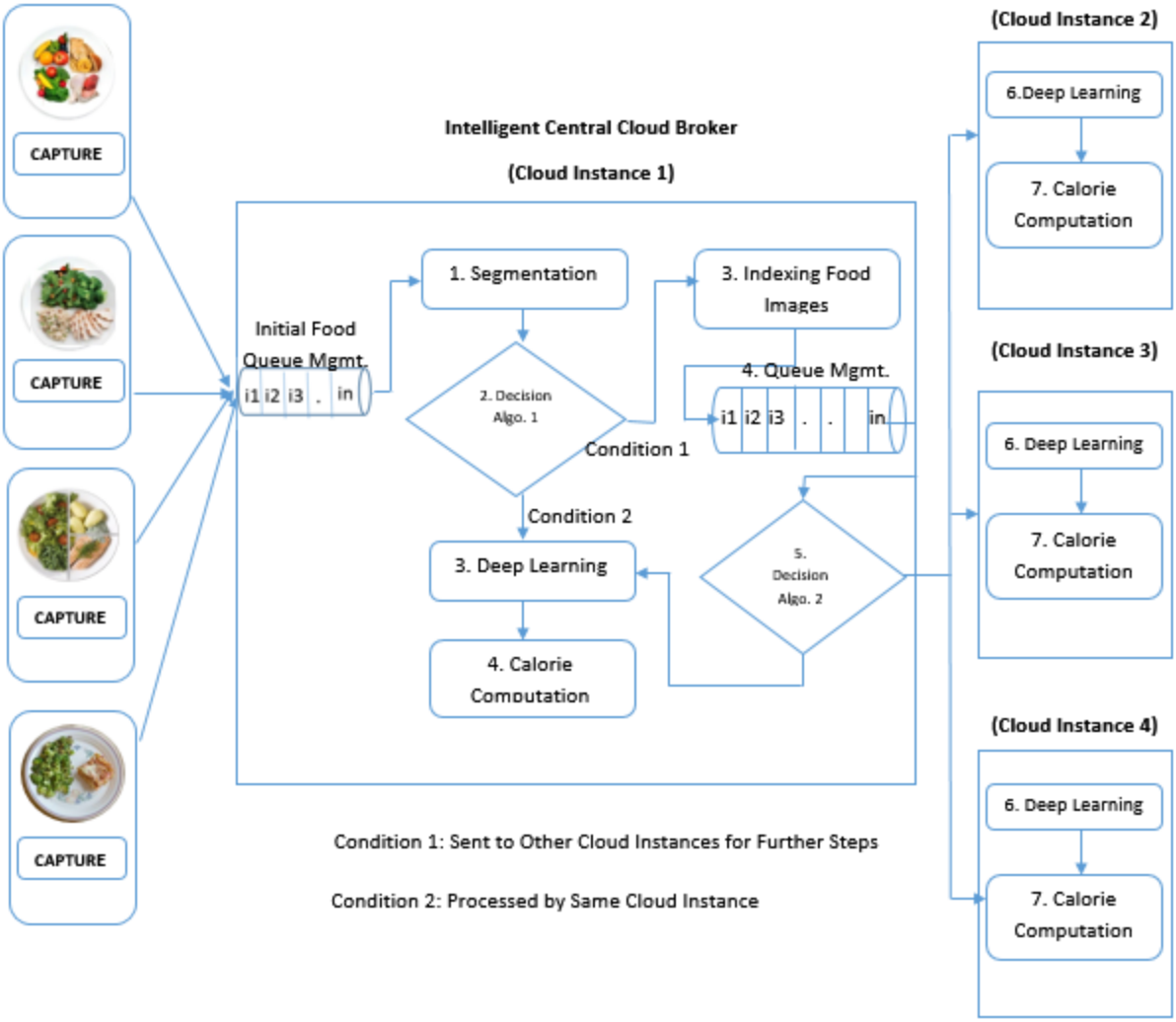

Figure 6: Intelligent Cloud Broker Architecture for Single & Multiple Food Objects

5.1 Cloud Broker Mechanism for Handling Single and Multiple Food Objects

Figure 6 describes the cloud broker mechanism for handling single and multiple food objects. Initially the users send the food object images from their mobile devices to the cloud broker, wherein they get stored in the first queue. The initial queue management sequentially dispatches the images for segmentation. In this scenario, the first step for the cloud broker would be to perform segmentation of the food portions, to determine if it is a single or multiple food object. The first step (segmentation) is imperative in determining the food category and hence all the food objects have to sequentially go through this process in order for the cloud broker to efficiently allocate them to other cloud instances for further parallel processing. Also to avoid failures as part of the initial queue, we implemented a caching mechanism wherein the cache will back up the images in the queue for a fixed time-span (in our scenario it is 1 hour) after which will refresh the queue to add newer images.

In the case of single food, the cloud broker will initiate Decision Algorithm 1, described in section 4.2 below, to decide whether the image processing is to be performed locally or on a separate cloud server. This is decided based on the number of active user connections, statistics of the broker (CPU Utilization, Memory Utilization, Queue Status, etc.), and historical data (average time consumption for running deep

learning for single food object, etc.). Once it decides to implement locally, it performs the remaining steps of Deep Learning and Calorie Computation on the cloud broker itself. Based on Decision Algo.1, if it decides to run the remaining steps on remote cloud servers, it will then index the food objects. It does this to maintain consistency for the food objects. After indexing the food images, it sends the images to queue management. The queue management maintains the queue of the food image, prioritizes them, and decides to offload the images, sequentially to other cloud servers. The offloading part is handled by Decision Algorithm 2, described in section 4.3 below, whose primary goal would be to assess statistics of the remote cloud servers and efficiently offload the food images to reduce the overall time consumption. After the image is offloaded to the remote cloud server, the remaining steps of deep learning and calorie computation are performed.

As shown in the Figure 6 and described above, two decision making algorithms are running on the cloud broker. Decision algorithm 1 is responsible for analyzing the performance of the local cloud broker and checking if it is within the specified threshold levels. Decision algorithm 2 on the other hand, is responsible for analyzing the performance of local server and further comparing it against the remaining 3 cloud instances. Let us now describe these two algorithms in details.

5.2 Decision Algorithm 1

The decision algorithm 1 determines if the image could processed locally or be dispatched to the queue management component, from where the images are further sent to the cloud instances. It firstly checks for disk space utilization followed by CPU utilization of the local cloud broker and whether it is below the threshold value (initially set to 85%). If the disk space utilization is not within the threshold of 85%, it will reject further incoming images, and reclaim space by removing the processed images from the server. The reason for setting the threshold value for the CPU Utilization to 85% is to allow the cloud broker to perform optimally without affecting the queue length. If it becomes 100 %, then the cloud broker will fail to function, which might result in stockpile of images in the queue management component, further effecting the overall response time. If the CPU utilization, is however greater than 85%, then it will send the image to queue management for further processing. But if the CPU utilization is below the threshold level, it will further check for memory utilization. Once the memory utilization is below the threshold level too, the local cloud broker will process the image by running deep learning to classify the food object, followed by calorie computation.

5.3 Decision Algorithm 2

Decision algorithm 2 is responsible for deciding the cloud node (instance) among the list of nodes (including the cloud broker) that would be take the least response time in terms of image processing and calorie computation. As shown in Algorithm 2, it will decide based on the following parameters: CPU Utilization (CU), Memory Utilization (MU) and the historical data. The historical data helps the cloud broker estimate the approximate time that the node my take and further assist the broker in choosing the cloud broker with the least timing.

Algorithm 2: Decision Algorithm 2

1: CPU Utilization for node c_x → CUc_x
2: Memory Utilization for node c_x → MUc_x
3: Estimated Time taken for node c_x → Tc_x;
4: for image i_k from images i_1, i_2, i_3, … i_k ... i_n do
5: CUc_1 ← check CU for node c_1;
6: if CUc_1 < THRESHLD then

```
7:              MUc1 ← check MU for node c1;
8:              if MUc1 < THRESHLD then
9:                      dispatch (ik) to node c1; break;
10:                     initiateDeepLearning (ik , c1);
11:     else
12:
13:             check the estimated Time Tc1 for CUc1;
14:             for each node ck do
15:                     CUck ← check CU for node ck;
16:                     check the estimated Time Tck for CUck; break;
17:                     MUck ← check MU for node ck;
18:                     check the estimated Time Tck2 for MUck; break;
19:                     if Tck > Tck2 then
20:                             Tcf ← Tck;
21:                     else
22:                             Tcf ← Tck2;          //where Tcf is the Desired Time
23:                     if Tc1 < Tcf then
24:                             dispatch (ik) to node c1;
25:                             initiateDeepLearning (ik ,c1); break;
26:                     else
27:                             dispatch (ik) to node ck;
28:                             initiateDeepLearning (ik , ck); break;
29:             end for
30:     break;
31: end for
```

Based on the historical data, we were able to estimate the expected time that the cloud instance might take, depending on CPU and memory utilization. For computing the expected time taken by the cloud instances, we perform a linear regression model between, firstly, CPU utilization (shown in Figure 7) and secondly memory Utilization (shown in Figure 8), against the real time taken by the cloud instances to perform deep learning and calorie estimation steps as shown in Figure 2.

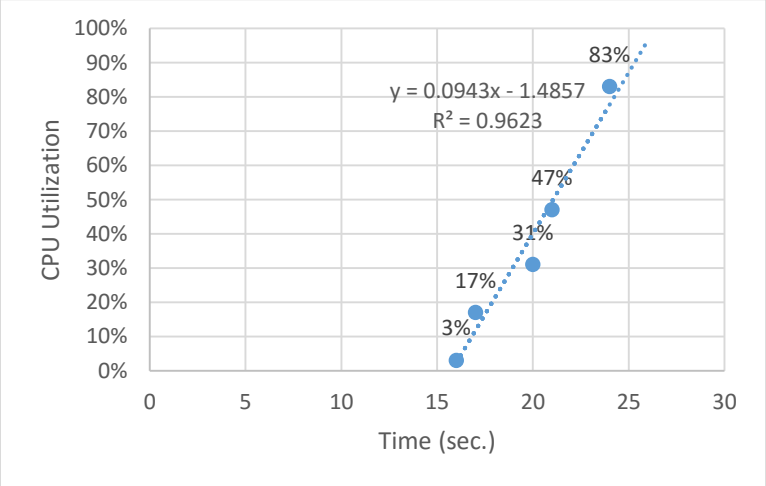

Figure 7: Linear Regression Model of CPU Utilization vs. time taken to process each image.

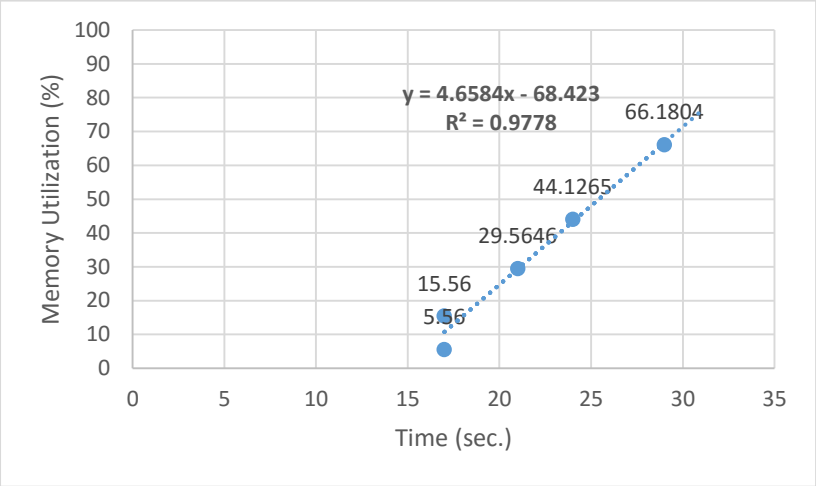

Figure 8: Linear Regression Model of Memory Utilization vs. Time Taken to Process each image.

We were hence be able to determine the correlation coefficient from Figure 7 and

Figure 8 that would further enable us to determine the expected time for the cloud instances to perform the image processing steps in different CPU and memory utilization scenarios. We notice that the R^2 for Memory Utilization is 0.9778, which is better as compared to the CPU Utilization (0.9623). Hence the readings from memory utilization are more reliable, due to which we firstly check the CPU Utilization followed by Memory Utilization in Decision Algo.2. As shown in Decision Algo. 2, we would further compare the two expected times as a result of both the linear regression models and select the largest of the two models.

5.4 Cloud Broker Mechanism for Handling Mixed Food Objects

For mixed food objects, the cloud broker uses the two level ingredient testing algorithm described in section 3.3.

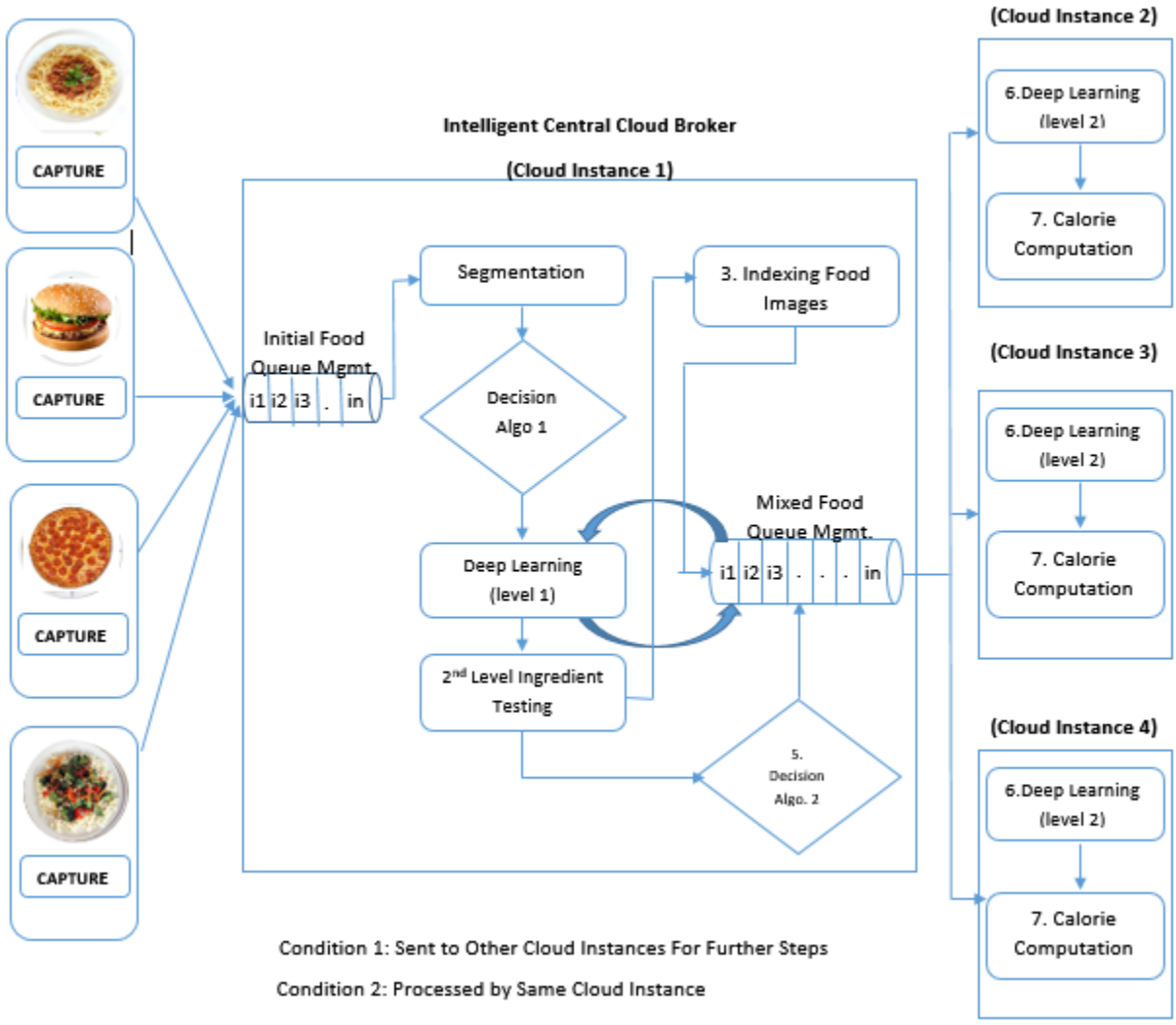

Figure 9: Intelligent Cloud Broker Architecture for Mixed Food Objects

The cloud broker sequentially pick images from the queue management (as shown in Figure 9), performs the first level of ingredient testing algorithm on the first image, and pass the result to Decision Algo.1, which would further offload it to other cloud instances for implementing the second level of ingredient testing algorithm. During the time the second level is being implemented by the remaining cloud servers, the cloud broker would pick the 2nd image and start the first level classification step. In this manner, the cloud broker would be efficiently handle the processing of the mixed food objects in the cloud.

5.5 Indexing Food Images

Indexing the food images is an essential part of the system, as it ensures the segmented food object images are consistently tagged. Indexing the food objects means to tag the segmented food images with a sequence number, to ensure the food images are processed in the same sequence and the results are

further retrieved in the same manner. For example, multiple food objects on the same plate, including orange and apple will be tagged with 1a and 1b, indicating that 1 is referring to user 1, a refers to the orange, and b to the apple, both for user 1. This also ensures the aforementioned consistency at the time of integration of the calorie and food item information from different cloud instances. The calorie value along with the classified food object name is sent as one cohesive answer to the mobile device.

For mixed food objects, the indexing of the food images ensures that the values from level 1 and level 2 are consistently tagged, in order to be indexed at the time of information retrieval from different cloud instances.

5.6 Scheduling and Queue Management

Once decision algorithm 1 decides to offload the target image, the image is indexed after which it is placed in the queue. The queue management component stores the images in the queue and further dispatches them to decision algorithm 2. The images are dispatched in the batch of 10 images, wherein they are further sent to the remote cloud instances for processing. The images are dispatched based on the First Come First Served (FCFS) scheduling mechanism, wherein the images that were sent to the queue first, will be the first to be dispatched to the cloud instances.

1. Dynamic Cloud Allocation Mechanism

Based on the current model, in order to address the issue of scalability when parallel food processing requests is received by the cloud broker, based on decision algorithm 1 and decision algorithm 2, the broker makes decision to dispatch the food images to the remaining three cloud instances. However, from the results we concluded that after 60 images, the system reached to its threshold of efficiently processing the food images and the time increased cumulatively. One way to address the issue was to add more cloud instances to the current architecture for improving performance. But adding more cloud instances, would lead to under-utilization of the cloud resources and large cost for maintenance during off-peak time, when less parallel requests are received by the broker. We hence modified our decision algorithm 2 and incorporated a technique wherein the system was able to dynamically add and remove cloud instances whenever the overall time of processing was severely impacted. In this manner, the system was intelligent to decide on when and how many cloud instances would be needed to process the food images on a given time instance. The ability of the system to take intelligent decision to add more cloud instances other than the existing 3 cloud instances and to further remove the instances to ensure the overall time taken to process the food images when sent in parallel, remains consistent (the condition that avg. time is greater than 70 seconds) as shown in Figure 3. This approach allows us to achieve the scalability in the system to handle multiple requests when sent in parallel. In our experimental setup, cloud instances were added when the total number of parallel requests were 50, 60 and 80 parallel image processing requests in order to ensure the overall response time was less than 70 seconds.

2. Scheduling and Queue Management

We restructured the scheduling algorithm to incorporate priority scheduling along with the existing FCFS scheduling and also explained 3 independent food queues (Single, Multiple and Mixed Food Queues). Since the overall time for calorie computation and deep learning algorithm for single food object processing is less, all the single food object images will be sent to single food queue to cloud instance 1. For the multiple food queue, since the queue included the segmented food objects, the total calorie value of multiple food object can only be sent when all the indexed food items are processed. For example, if there were 3 food objects in plate 1 (1a, 1b and 1c), the final calorie results will only be sent to the user when each of the food objects (1a, 1b and 1c) has been received from cloud instance. For mixed food

objects, since the images in the queue were part of the second iteration, the overall time taken to process the single food object was ranging from 8-10 seconds which was less than processing a single food object.

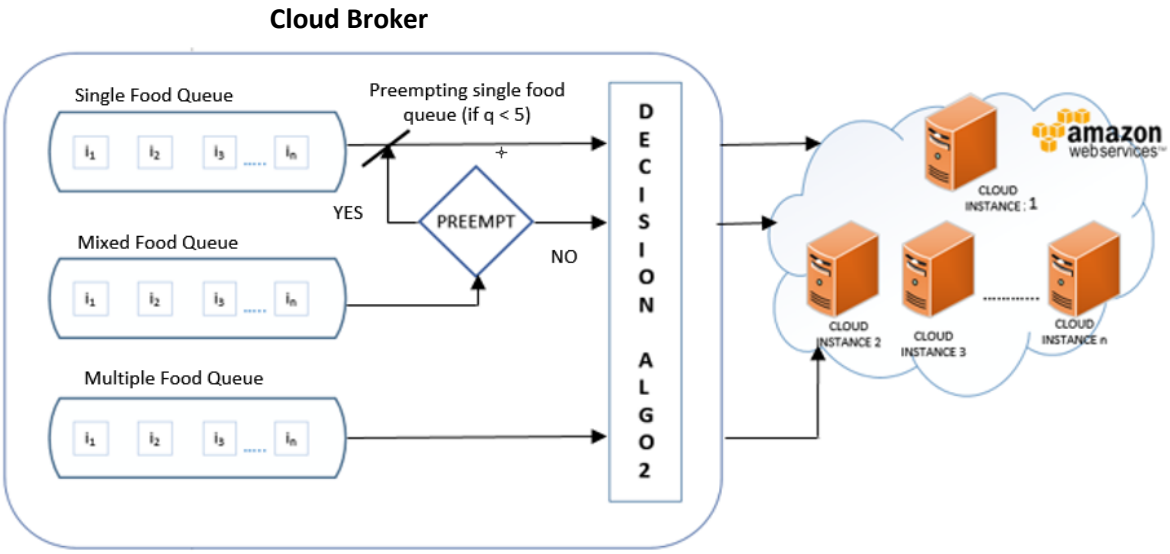

Figure 10: Scheduling & Queue Management

Since the time taken to process the second iteration of mixed food image is less (considering the smaller trained model file M2 when compared to generic model file (M1)), the system will preempt the single food queue in order to give higher priority to mixed food image over the single food image only if the single food queue is less than 5 food images. The reason for this condition was, due to the cloud instance ability to process images in batch of 5 images in parallel. If the condition is not met, then single food image will be given priority over the mixed food image and will be processed with the dedicated cloud instance 1, and the mixed food image will be processed as part of the existing methodology wherein it would wait for remaining cloud instances to finish the current batch processing.

To provide a better understanding of the queue management, we proposed a queue management algorithm which includes scenario for each of the queues including: Initial Queue Qi where the food images are sequentially dispatched to the segmentation algorithm, Single Queue Qs which includes the single food object after segmentation, multiple queue Qm which includes more than one and less than four food objects after segmentation and mixed queue Qmx which includes food objects in the second level of the ingredient testing.

Algorithm 3: Queue Management

1: Segmentation of image i_k → $S(i_k)$
2: Initiating Decision Algorithm 1 for image i_k → decisionAlgo1(i_k)
3: Count of Total number of ROI from Segmentation $S(i_k)$→ count ($S(i_k)$)
4: Level of the Image i_k processed in the 2nd Level Ingredient Testing → level (i_k)
5. Initial Queue, Single Food Queue, Multiple Food Queue, Mixed Food Queue → Q_i, Q_s, Q_m, Q_{mx}
6. Image I_k pre-empting Single Queue Q_s→ preempt ($Q_s(i_k)$)
7: for image ik from images i1, i2, i3, … ik ... in do
8: QueueM(i_k) {
9: Switch (Q){
10: Case (Q_i,): segmentation (i_k)
11: decisionAlgo1(i_k);
12: if (count ($S(i_k)$) = = 1) then
13: dispatch (i_k) to queue (Q_s)
14: else if (count ($S(i_k)$) > 1 && count ($S(i_k)$) < 4)
15: for image i_k from images i1, i2, i3, … ikn ... in do
16: i_{kn} ← indexFoodImage(i_k)
17: dispatch (i_{kn}) to queue (Q_m)
18: end for
19: else if (level(i_k) = = 2)
20: dispatch (i_k) to queue (Q_{mx})
21: Case (Q_s): append (i_k, Q_s)
22: if (j(i_k) = end)
23: decisionAlgo2(i_k)
24: else wait($Q_i(i_k)$)
25: Case (Q_m): append (i_k, Q_m)
26: if (j(i_k) = end)
27: decisionAlgo2(i_k)
28: else wait ($Q_i(i_k)$)
29: Case (Q_{mx}): append (i_k, Q_m)
30: if (j(i_k) = end && length (Q_s) < 5)
31: preempt ($Q_s(i_k)$)
32: else if (j(i_k) = end && length (Q_s) > 5)
33: decisionAlgo2(i_k)
34: else wait($Q_{mx}(i_k)$)
35: default (Q_i,): wait ($Q_i(i_k)$)
36: }
37: end for

Once the images are dispatched from the queue to the remote cloud server, the images are processed in parallel on multiple threads. On each thread, the cloud instance runs the deep learning algorithm on the image and further computes the calorie content based on the auto-calibration method. Processing the images inside the queue in parallel ensures that the queue is never stockpiled with many images, further ensuring the desired response time from the system.

6. Implementation Results

In this section, we report on the experimental setup and results. For the experimental setup, we had different cloud instances setups for the training and the testing the food images. We will also explain the simulations performed on the central cloud broker and the setup we used in order to test parallel processing of the intelligent cloud broker.

Training Setup

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) [12]. 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, and Scipy on top of our Ubuntu 14.04 cloud instance and further made it suitable to deploy the training models [12].

Testing Setup

The tests were performed on multiple cloud instances, which were further integrated with the central cloud broker for improving the overall timing results for parallel processing of food images. 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 image [12]. We further installed HDF5 and OpenCV on t2.micro instance [12].

For running the systems shown in Figure 6 and Figure 9, we used t2.medium as the central cloud broker, which was further integrated to three other cloud instances (t2.micro) in the cloud provided by AWS. For achieving parallel processing from multiple mobile phones, we had established 5 virtual emulators, with Android x86 operating system, and 5 mobile devices running our EHSH application. We used the concept of cloud based virtualization (explained in [6]), for achieving multiple virtual Android emulators. As we known from [6], Android x86 emulator is faster compared to hardware assisted Android emulator. The smartphone that we used was the Samsung Galaxy S4, which has been upgraded to Android 4.4.2 KitKat. The Android x86 emulator used was an Android 4.4 KitKat image. The configuration of our physical device Samsung Galaxy S4 included an octa-core processor and a 1.6 GHz Quad + 1.2 GHz Quad PU speed with 2GB of RAM [22].

Results

As shown in Figure 11, we initially compute the total time taken for each of the components described in Figure 6 and Figure 9. This includes the various components of EHSH: food recognition (deep learning feature extraction and deep learning classification) and calorie measurement (distance based auto-calibrated calorie computation) along with different components of the intelligent cloud broker (Decision Algo.1 and Decision Algo.2 explained in sections 4.2 and 4.3). Based on Figure 11 we are able to note the overall time taken for each of the mentioned food categories (single food object, multiple food objects and mixed food object). We can observe that Deep Learning feature extraction for each food category consumes a maximum time of 42.92% of the overall time for processing single food objects. The same is applicable to other food categories wherein it takes, 44.84%, 43.32% and 47.80% of the overall time for Multiple Food Object (2 and 3 Food Objects) and Mixed Food Objects.

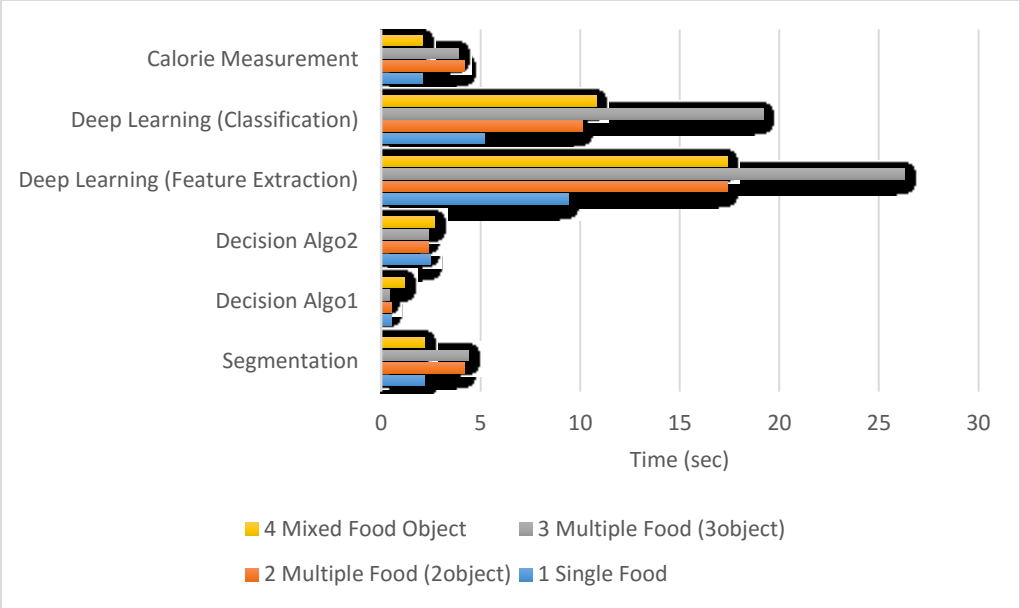

Figure 11: Time Taken for each Food Category in various EHSH Components

Decision Algo.1, which decides whether or not to process the food objects locally, consumes the least amount of time with as little as 3.29 % of the overall time taken in the case of Mixed Food object. From this, we can note that with the inclusion of Decision Algo.1 into the overall system, the time will not be significantly impacted. Instead, it helps in lowering the time during parallel processing, which can be seen in Figure 11. The same could be observed with Decision Algo.2, which also takes less time compared to the other components except for Decision Algo.1, which could be explained by the fact that Decision Algo.2 checks the viability of each and every cloud instance (including the cloud broker) and determines where to offload the food image for processing. It hence takes approximately twice as much time as Decision Algo. 1. Also from Figure 11, we can observe that the time taken for segmentation for multiple food objects increases with increasing number of food objects, which could be explained by the fact that additional ROIs have to be determined and further segmented by the segmentation algorithm. Also the time taken for the Mixed Food Objects is approximately twice taken by single food objects, due to the additional level explained in Section 3.2.

Figure 12, describes the time taken by different Image sets, when processed in parallel in the cloud. With the integration of the central cloud broker into the system, we are able to test the scalability of the system. As shown in Figure 12, the time taken to process the initial 5 images in each case remained approximately similar with 14.9 seconds. With the addition of 15 more images, we observe an increase of 16.67% in the average overall time consumption of each of the remaining images. The time further increased by 40% (as compared to 20 images in parallel) for 40 images running in parallel. For the last 60 images processed in parallel, the increase was almost to 37.5 % over the previous images.

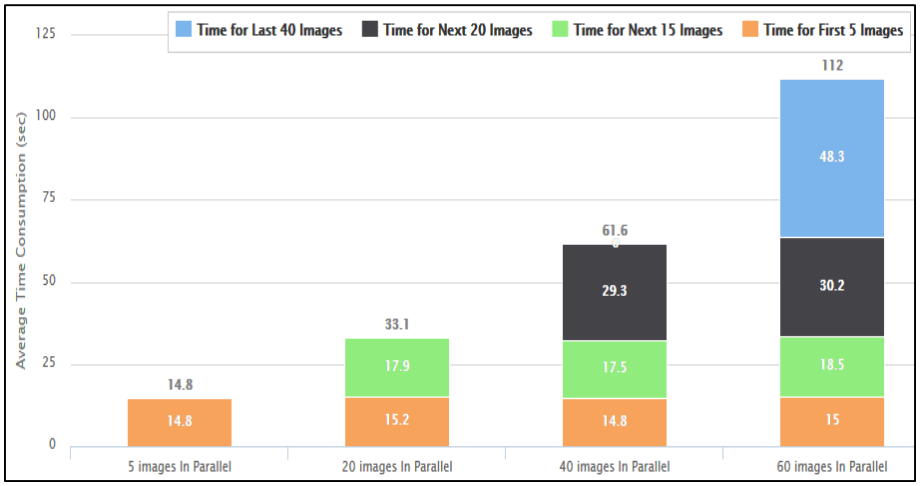

Figure 12: Average Time Consumption for Parallel Processing of Images by the Cloud Broker (without dynamic cloud allocation)

Hence with the use of three t.micro cloud instances, the central cloud broker was able to efficiently maintain the average time of the food image processing, till 40 images, where the total time was 61.6 seconds. However, with further increase in the number of images to be processed in parallel, the broker was unable to mitigate the time and instead got increased in a cumulative manner in a non-dynamic cloud allocation mechanism. But, comparing these results to the scenario where the cloud broker was not used at all, we see a difference of 45% faster overall time taken to process in the case of 20 images when processed in parallel.

Based on these results, we can interpret that the intelligent cloud broker was essential in achieving a 45% gain in the overall time taken to process the images in parallel in the cloud. However, without the dynamic cloud allocation model, the cloud broker doesn't optimally perform in case of parallel images greater than 60, wherein the overall time increases. Hence, as discussed in section 4.6 (Scheduling and Queue Management), we have introduced the dynamic cloud allocation mechanism which allows the system to intelligently add the cloud instances when the system gauges that the average time consumption for processing parallel images will be greater than 70 seconds. Also, in order to address the issue of under-utilization of these cloud resources, the system also decides on removing the cloud instances, when not in use. The graph (Figure 13) below compares the total parallel image requests with the average time consumption (sec.), with and without the dynamic cloud allocation mechanism. The results obtained from without dynamic cloud allocation model is based on Figure 12 (Average time consumption for parallel processing of images by cloud broker) in our paper. The time taken to process parallel 5 images took 14.9 seconds in both the scenarios (with and without dynamic cloud allocation).

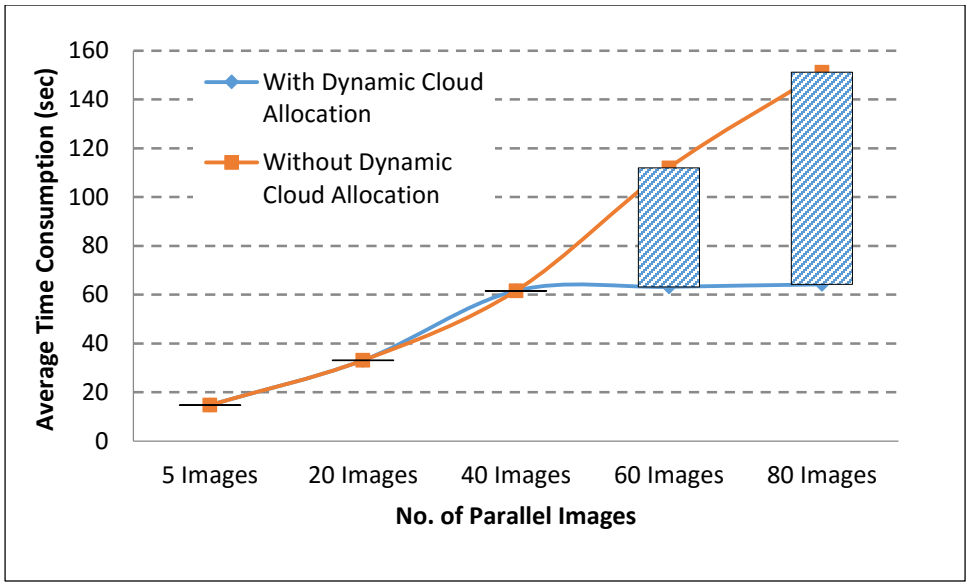

Figure 13: Parallel Images v/s Average Time Consumption (sec.)

The time again remains the same to process 40 images wherein it took 61.6 seconds to process 40 images. Based on our requirements, our goal was to ensure the overall time to process the image, calorie computation of the food image and returning the result back to user's mobile should be less than 70 seconds. With the use of dynamic cloud allocation mechanism, we were able to bring down the average time consumption by 77.21 % when 60 images were processed in parallel. The overall time average further improved for 80 images, wherein the 135.93% improvement was observed when compared to non-dynamic cloud allocation system.

Also, in order to ensure efficient utilization of cloud resources and at the same time ensure the overall response time of the processing the image remains less than 70 seconds, the cloud nodes had to be strategically increased and even decreased when not in use. The graph Figure 14 below specifies the cloud nodes required for the processing a set of parallel image requests. For processing 5 to 40 parallel image requests we observed that system used 1 Cloud Broker and 3 Micro Cloud instances making it 4 cloud nodes to ensure the time remains less than 70 seconds. However the cloud nodes had to be increased to 5 and 6 respectively to process 50 and 60 images.

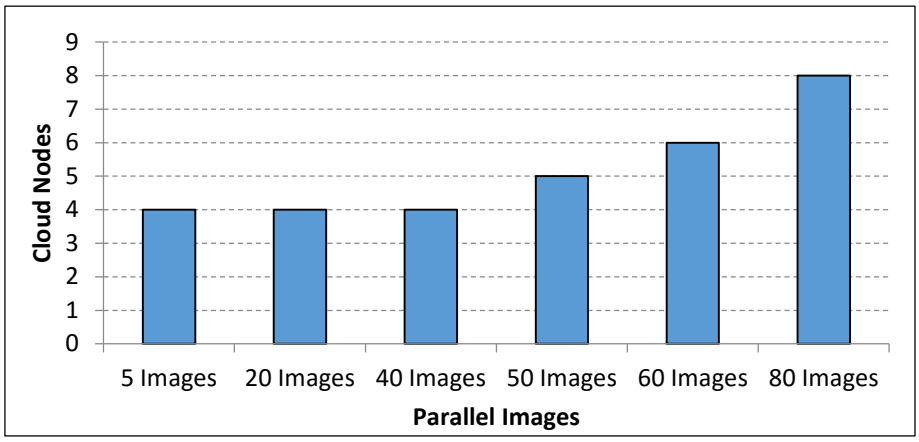

Figure 14: Comparison between the number of cloud nodes and the parallel image request (Avg. time < 70 sec.).

As shown in Table 2, in [32] which is also a food based image recognitions system, they build a Hadoop based cloud model and observed 77.38% improvement in response time when processing the image requests in parallel. In our proposed EHSH system, we observed an improvement of 77.21% and 135.93% while processing 40 and 60 images in parallel. These improvements are a result of our dynamic cloud allocation model combined with the efficient queue management algorithm that intelligently handled the incoming request reducing the overall response time.

Table 2: Comparisons with e-health Models

e_Health Models	**Cloud Computing Model**	**Processing Time Improvements**	**Improvement Observed**
Food Image Recognition [32]	Pervasive Cloud Computing, hadoop	77.38%	Parallel Images processing
EHSH Food Recognition	Decision Making, Dynamic Cloud	77.21%, 135.93%	Parallel Image Processing

7. Conclusions and Future Work

In this paper, we have introduced an intelligent cloud broker for EHSH, supporting different food categories of Single Food Object, Multiple Food Object and Mixed Food Object. We have further introduced a 2 level ingredient testing approach for the mixed food category. We then introduced two decision algorithms (Decision Algo. 1 and Decision Algo. 2) that enabled the cloud broker to make decisions of whether or not to offload to other cloud instances and if so to which cloud instance in particular. We discussed the various components of the intelligent cloud broker, including the food recognition components (deep learning feature extraction and deep learning classification model), calorie measurement, indexing the food images, scheduling and queue management. Finally, scalability, consistency, and the performance of the system in terms of response time was studied and presented. The proposed model helped us achieve an improved performance and efficient utilization of the cloud resources on one hand and also to address the scalability aspect of the system, wherein the system was

able to process parallel requests coming from different users. For processing parallel requests, if the system continuously adds more cloud instances it would lead to increased cost and under-utilization of resources. Hence, we introduced dynamic resource allocation mechanism, wherein the system will strategically add and remove the cloud instances depending on the resource requirement. This will ensure that the system will not add additional resources exponentially, but rather gauge the requirement and accordingly take decision to add and remove the cloud resources.

As part of the future work, we would like to evaluate the proposed dynamic cloud allocation mechanism for improving the overall accuracy of the system and the the impact of adding more GPU as against CPU cores in cloud environment, as the deep learning algorithm is severely impacted by the GPU processing, which we used in the training phase. Although our cloud broker model is specifically designed for the various food categories and is the first of its kind, we would however like to compare it to other models for non-food categories on common metrics.

8. References

[1] C. Lister, JH. West, B. Cannon, T. Sax, D. Brodegard, 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.

[2] P. Pouladzadeh, P. Kuhad, S.V.B. Peddi, A. Yassine, and S. Shirmohammadi, Mobile Cloud Based Food Calorie Measurement, Proc. IEEE International Workshop on Multimedia Services and Technologies for E-health, in Proc. IEEE International Conference on Multimedia and Expo, Chengdu, China, July 14-18 2014, 6 pages.

[3] P. Pouladzadeh, S.V.B. Peddi, P. Kuhad, and S. Shirmohammadi, A MapReduce Parallel Classifier for Cloud-Based Food Recognition, Proc. International Conference on Next Generation Computing and Communication Technologies, Dubai, UAE, April 23-24 2014, pp. 142-147.

[4] P. Kuhad, A. Yassine, and S. Shirmohammadi, Using Distance Estimation and Deep Learning to Simplify Calibration in Food Calorie Measurement, Proc. IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, Shenzhen, China, June 12-14 2015, 6 pages.

[5] S.V.B. Peddi, A. Yassine, and S. Shirmohammadi, Cloud Based Virtualization for A Calorie Measurement E-Health Mobile Application, Proc. IEEE International Workshop on Multimedia Services and Technologies for E-health, in Proc. IEEE International Conference on Multimedia and Expo, Torino, Italy, June 29 - July 3 2015, 6 pages.

[6] P. Pouladzadeh, S.V.B. Peddi, P. Kuhad, A. Yassine, and S. Shirmohammadi, A Virtualization Mechanism for Real-Time Multimedia-Assisted Mobile Food Recognition in Cloud Computing, *Cluster Computing*, Springer, 13 pages, accepted June 1 2015, to appear.

[7] E. Cuervo, et al. MAUI: making smartphones last longer with code offload. Proceedings of the 8th international conference on Mobile systems, applications, and services. ACM, 2010.

[8] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, The case for vm-based cloudlets in mobile computing, Pervasive Computing, IEEE , vol. 8, no. 4, pp. 14 –23, oct.-dec. 2009.

[9] H.B.W Heinzelman, C.A. Janssen and J.Shi Mobile Computing-A green computing resource, Wireless Communications and Networking Conference (WCNC), pp 4451-4456, 7-10 April 2013.

[10] E. E. Marinelli, Hyrax: Cloud computing on mobile devices using mapreduce, Master's thesis, Carnegie Mellon University, 2009.

[11] Kemp, R, et al. Cuckoo: a computation offloading framework for smartphones. Mobile Computing, Applications, and Services. Springer Berlin Heidelberg, 2012. 59-79.

[12] P Kuhad, A Deep Learning and Auto-Calibration Approach for Food Recognition and Calorie Estimation in Mobile e-Health, Master's thesis, University Of Ottawa, 2015, http://hdl.handle.net/10393/32455.

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

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

[15] P. Pouladzadeh, S. Shirmohammadi, and A. Yassine, Using Graph Cut Segmentation for Food Calorie Measurement, Proc. IEEE Symposium on Medical Measurements and Applications, Lisbon, Portugal, June 11-12 2014, 6 pages.

[16] Kumar, Karthik, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. Mobile Networks and Applications, vol. 18, no. 1, pp. 129-140, 2013

[17] S. Deng, L. Huang, J. Taheri, A.Y. Zomaya, Computation Offloading for Service Workflow in Mobile Cloud Computing, Parallel and Distributed Systems, IEEE Transactions on, vol.PP, no.99, pp.1,1 2015, doi: 10.1109/TPDS.2014.2381640 2015

[18] H. Wu, Q. Wang and K. Wolter Mobile Healthcare Systems with Multi-cloud Offloading Proc. of the 14th IEEE International Conference on Mobile Data Management, June 2013, pp. 188-193

[19] F.Zhang, J. Cao, S. U. Khan, K. Li, K. Hwang A task-level adaptive MapReduce framework for real-time streaming data in healthcare applications Future Generation Computer Systems Vol.43–44 (2015) pp. 149–160, 2015

[20] J. Wang , M. Qiub , B. Guoa High reliable real-time bandwidth scheduling for virtual machines with hidden Markov predicting in telehealth platform, Future Generation Computer Systems Volume 49, Pages 68–76, 2015

[21] M. Zapater, P. Arroba, J. L. Ayala, J. M. Moya, K. Olcoz, A novel energy-driven computing paradigm for e-health scenarios, Future Generation Computer Systems, Vol. 34, pages 138–154, 2014

[22] S.V.B Peddi, Cloud Computing Frameworks for Food Recognition from Images, Master's thesis, University Of Ottawa, 2015, http://hdl.handle.net/10393/32450

[23] G. Hinton, S. Osindero, and Y. Teh, A fast learning algorithm for deep belief nets, Neural Comput., vol. 18, no. 7, pp. 1527–1554, 2006.

[24] G. Hinton and R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science, vol. 313, no. 5786, pp. 504–507, 2006.

[25] Y. Bengio, Learning deep architectures for AI, Foundat. Trends® in Mach. Learn., vol. 2, no. 1, pp. 1–127, 2009.

[26] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, Greedy layerwise training of deep networks," Proc. Adv. Neural Inf. Process. Syst., vol. 19, pp. 153–161, 2007.

[27] A. Ranabahu, M. Maximilien, A best practice model for cloud middleware systems, in: Proceedings of the Best Practices in Cloud Computing: Designing for the Cloud, ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA, Orlando, FL, USA, 2009.

[28] Pandey, S., Voorsluys, W., Niu, S., Khandoker, A., & Buyya, R. (2012). An autonomic cloud environment for hosting ECG data analysis services. Future Generation Computer Systems, 28(1), 147-154.

[29] C. Vecchiola, X. Chu, R. Buyya, High Speed and Large Scale Scientific Computing, IOS Press, ISBN: 978-1-60750-073-5, 2009, (Chapter). Aneka: A Software Platform for.NET-Based Cloud Computing, pp. 267–295.

[30] Teodoro, G., Pan, T. F., Kurc, T. M., Kong, J., Cooper, L., Podhorszki, N & Saltz, J. H. (2013, May). High-throughput analysis of large microscopy image datasets on cpu-gpu cluster platforms. In Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on (pp. 103-114). IEEE.

[31] Vetter JS, Glassbrook R, Dongarra J, Schwan K, Loftis B, McNally S, Meredith J, Rogers J, Roth P, Spafford K, Yalamanchili S. Keeneland: Bringing Heterogeneous GPU Computing to the Computational Science Community. Computing in Science and Engineering. 2011;13

[32] Duan, Pengcheng, et al. Food Image Recognition Using Pervasive Cloud Computing. Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing. IEEE, 2013.

[34] M. Shamim Hossain, Cloud-Supported Cyber–Physical Localization Framework for Patients Monitoring, IEEE Systems J., Sep 2015.

[35] Hassan M. M., Cost-effective resource provisioning for multimedia cloud-based e-health systems. Multimedia Tools and Applications, Vol. 74 (14), pp. pp 5225-5241, 2015

[36] M. Masud, et al, Data Interoperability and Multimedia Content Management in e-health Systems, IEEE Trans. Inf. Technol. Biomed. Vol. 16, No. 6, pp. 1015-1023, Nov. 2012

[37] Hassan, M.M., Song, B., Almogren, A., Hossain, M.S., Alamri, A., Alnuem, M., Monowar, M.M. and Hossain, M.A., Efficient Virtual Machine Resource Management for Media Cloud Computing. KSII Transactions on Internet and Information Systems (TIIS), 8(5), pp.1567-1587, 2014

[38] M. Shamim Hossain, and G. Muhammad, Cloud-Assisted Industrial Internet of Things (IIoT)-enabled Framework for Health Monitoring," Elsevier Computer Networks, 2016