Cluster Comput (2015) 18:1099–1110
DOI 10.1007/s10586-015-0468-2

A virtualization mechanism for real-time multimedia-assisted mobile food recognition application in cloud computing

Parisa Pouladzadeh[1,2] · **Sri Vijay Bharat Peddi**[1,2] · **Pallavi Kuhad**[1,2] · **Abdulsalam Yassine**[1,2] · **Shervin Shirmohammadi**[1,2]

Received: 26 September 2014 / Revised: 1 June 2015 / Accepted: 1 June 2015 / Published online: 9 September 2015

Abstract The integration of multimedia-assisted healthcare systems with could-computing services and mobile technologies has led to increased accessibility for healthcare providers and patients. Utilizing cloud computing infrastructures and virtualization technologies allows for the transformation of traditional healthcare systems that demand manual care and monitoring to more salient, automatic and cost effective systems. The goal of this paper is to develop a multimedia-assisted mobile healthcare application using cloud-computing virtualization technologies. We consider calorie measurement as an example healthcare application that can benefit from cloud-computing virtualization technology. The key functionalities of our application entail image segmentation, image processing and deep learning algorithms for food classification and recognition. Client side devices (e.g. smartphones, tablets etc.) have limitations in handling time sensitive and computationally intensive algorithms pertained to our application. Image processing and deep learning algorithms, used in food recognition and calorie measurement, consume devices' batteries quickly, which is inconvenient for the user. It is also very challenging for client side devices to scale for large number of data and images, as needed for food recognition. The entire process is time-consuming and inefficient and discomforting from users' perspective and may deter them from using the application. In this paper, we address these challenges by proposing a virtualization mechanism in cloud computing that utilizes the Android architecture. Android allows for parting an application into activities run by the front-end user and services run by the back-end tasks. In the proposed virtualization mechanism, we use both the hosted and the hypervisor models to publish our Android-based food recognition and calorie measurement application in the cloud. By so doing, the users of our application can control their virtual smartphone operations through a dedicated client application installed on their smartphones, while the processing of the application continue to run on the virtual Android image even if the user is disconnected due to any unexpected event. We have performed several experiments to validate our mechanism. Specifically, we have run our deep learning and image processing algorithms for food recognition on different configuration platforms on both the cloud and local server connected to the mobile. The results show that the accuracy of the system with the virtualization mechanism is more than 94.33 % compared to 87.16 % when we run the application locally. Also, with our virtualization mechanism the results are processed 49 % faster than the case of running the application locally.

Keywords Cloud computing · Virtualization · Healthcare · Food recognition · Multimedia

✉ Parisa Pouladzadeh
pr_pouladzadeh@yahoo.com;
ppouladzadeh@discover.uottawa.ca

Sri Vijay Bharat Peddi
vpedd013@uottawa.ca

Pallavi Kuhad
pkuha009@uottawa.ca

Abdulsalam Yassine
ayassine@discover.uottawa.ca

Shervin Shirmohammadi
shervin@discover.uottawa.ca

[1] Distributed and Collaborative Virtual Environments Research Laboratory, University of Ottawa, Ottawa, Canada

[2] Colleges of Engineering and Natural Sciences, Istanbul şehir University, Istanbul, Turkey

1 Introduction

The integration of multimedia-assisted healthcare systems with cloud-computing services and mobile technologies has led to increased accessibility for healthcare providers and patients [1]. Utilizing cloud computing infrastructures and virtualization technologies allows for the transformation of traditional healthcare systems that demand manual care and monitoring to more salient, automatic and cost effective systems [1,2]. With such paradigm, patients will not only have fast responses to health related inquiries without any disturbance to their daily routines, but also access to sophisticated back-end emergency processing centres, which at the end can help reduce healthcare costs [3]. Our goal in this paper is to develop a multimedia-assisted mobile healthcare application using cloud-computing virtualization technology. We consider food recognition and calorie measurement [4–6] as an example healthcare application that needs cloud-computing virtualization for efficient execution and mass deployment.

In our food recognition and calorie measurement application, the user takes a picture of the food using his smartphone and the application measures the amount of calorie intake automatically. The system enables the user/patient to obtain the measurement results of the food intake from the application, which simulates the calculation procedure performed by the dietician. The process entails several key functionalities such as the use of food pictures, image processing by graph cut segmentation, image processing and analysis, and deep learning algorithms for food classification and recognition. There are several challenges that make virtualization to be more effective in such type of real-time healthcare application. For example, existing client side devices (e.g. smartphones, tablets etc.) have limitations in adaptively allocating necessary computing resources in order to handle time sensitive and computationally intensive algorithms. Image processing and deep learning algorithms, essential for food recognition, consume devices' batteries quickly, which is inconvenient for the user. It is also very challenging for client side devices to scale for the large number of food data and images that are necessary to achieve high accuracy. The entire process is time-consuming and inefficient and discomforting from users' perspective and may deter them from using the application.

With cloud services the above challenges can be overcome. Users with mobile devises can rely on cloud resources to perform computationally intensive operations such as image segmentation, deep learning, data mining, and image processing. In this paper, we address the above challenges by proposing a virtualization mechanism in cloud computing that utilizes the Android architecture. Android allows for parting an application into activities run by the front-end user and services run by the back-end tasks. Generally, the approach for virtualization is either performed using a hosted or hypervisor architecture [7]. A hosted architecture installs and runs the virtualization layer as an application on top of an operating system and supports the broadcast range of hardware configuration. In contrast, hypervisor (bare-metal) architecture installs the virtualization layer directly on a clean x86-based system [8]. A hypervisor acts as a virtual machine manager that allows multiple operating systems to share a single hardware host. Each operating system appears to have the host's processor, memory, and other resources all to itself. However, the hypervisor is actually controlling the host processor and resources, allocating what are needed to each operating system in turn and making sure that the guest operating systems (virtual machines) cannot disrupt each other [8]. In this paper, we have implemented both models (details are in Sect. 3), wherein (i) we configured Android x86 image on top of VMware Elastic Sky Integrated (ESXi) for achieving Bare-Metal architecture and (ii) Android x86 image on top of VMware Workstation on the host operating system. We finally used the hosted architecture for publishing our Android-based food recognition and calorie measurement application in the cloud. The main contributions made in this paper are as follows:

- We propose a cloud-based virtualization mechanism for multimedia-assisted mobile food recognition application. Our mechanism allows users to control their virtual smartphone operations, through a dedicated client application installed on their smartphones, while the processing of the application continues to run on the virtual Android image even if the user is disconnected for any unexpected reason. With our mechanism, the mobile environment is emulated on the cloud in a manner that users always feel as if the application is being run on the smartphone, but in reality it is virtually running on the remote server in the cloud. This is rather significant to overcome the limited capability of smartphones to run intensive machine learning algorithms similar to those presented in this paper.
- Our model integrates virtualization mechanism of the multimedia-assisted mobile application with deep neural networks in cloud computing. In our system, we use deep convolutional neural networks (CNN) as a backbone of the application and the system handles the training and testing requests at the top layers, without affecting the central layers. This will allow us to enhance the accuracy of food recognition by adding a large data set of images in a cloud-based storage.
- Our experimental results of running the application on cloud servers show significant improvements when compared with experiments running on the local server. The results show that the rate of food recognition when we run the application in the cloud is more than 94.33 %

compared to 87.160 % when we run the application on a local server. Also, with our virtualization mechanism the results are processed up to 49 % faster than the case of running the application locally.

The rest of the paper is organized as follows: The related work is presented in the next section followed by the proposed system in Sect. 3. In Sect. 4, we present the implementation of the proposed virtualization mechanism for our food recognition and classification application. In Sect. 5, we provide the experimental results and finally in Sect. 6 we conclude the paper and provide directions for future work.

2 Related work

In this section, we present related work in two areas: Multimedia-assisted food recognition and classification systems, and virtualization mechanisms for healthcare applications using cloud computing. In particular, we examine a sample of both areas of studies that we believe to be representative and specifically related to our work. We also describe in this section the main advantages and drawbacks of existing work and the need for new mechanisms as proposed in this paper.

2.1 Multimedia-assisted food classification and recognition systems

The open literature describes several approaches that use multimedia mechanisms such as image processing to analyze food content, e.g. [9–14]. In [9] the authors propose a system that utilizes food images that are captured and stored by multiple users in a public Web service called FoodLog. Then, a dictionary dataset of 6512 images is formed including the calorie estimation. The images in the dictionary are used in dietary assessment approach. The accuracy in such approach for measuring calories is very low. In [9], a new 3D/2D model-to-image registration framework is presented for estimating food volume from a single-view 2D image containing a reference object. In this system, the food is segmented from the background image using morphological operations while the size of the food is estimated based on user-selected 3D shape model. In [11,12], a set of pictures is taken for before and after food consumption in order to recognize and classify the food and determine its size. In such method, the existence of a premeasured and predefined measurement pattern is used inside the images to translate the size in pixels of each portion. All these conditions can generate difficulties, which has been addressed by Martin et al.[13]. In [13], the authors proposed a system that captures the images and sends them to a research facility where the analysis and extraction are performed. The major disadvantage of such system is that it does not provide information to the users in real-time. There is a considerable delay in providing the information due to the offline processing of images. A smartphone based application for recording food intake is proposed in [14]. This system uses image analysis tools for identification and quantification of food consumption. However, it does not provide estimation of calorie intake and only process 2D images. Other various image processing and machine learning techniques applied in different steps of food recognition systems are discussed in [4–6].

2.2 Virtualization of healthcare applications in cloud computing

In the past few years several studies, such as those presented in [15–20] have proposed methods for virtualization of healthcare applications in cloud computing as an alternative underlying technology to overcome the limitation of existing healthcare system. For example the work presented [15] proposes a system called “MedCloud” which utilizes cloud-based technologies in conjunction with privacy and security rules for patients’ data storage. The authors in [16] propose mobile cloud for Assistive Healthcare (MoCAsH) as an infrastructure that makes use of the cloud computing as well as collaborative plans by deploying intelligent mobile agents, context-aware middleware, and collaborative protocol for efficient resource sharing and planning. It also addresses various quality-of-service issues concerning critical responses and energy consumption. The goal of the work in [17] is to develop a cloud-assisted mobile pervasive system with medical software as a service (SaaS). The mechanism of virtualization uses back-end real-time application server stack to store and manage patients’ health records. It focuses on deadline-critical real-time medical data which is generated by sensor-based medical devices, such as wireless electrocardiogram (ECG). In order for the system to handle time sensitive and mission-critical medical data in a public cloud computing infrastructure, it uses a real-time application server operated as a virtual machine. Virtualization of healthcare sensors is addressed in [18–20], which develop a virtual sensor for remote health monitoring applications. The system tries to overcome the discomfort issue of wearing blood pressure sensors during monitoring. The application is deployed in the cloud to ensure scalability, accessibility and flexibility. In such system, the physician or care-taker is notified if there is any deviation from the normal value for immediate attention.

In parallel to the above work, several mechanisms are proposed to help users conveniently use cloud-base healthcare systems. In [21], the authors build a “virtual network computing (VNC)” server and a VNC client for establishing the remote desktop connectivity between the mobile device

and the server. They were able to establish remote connection to mobile from the desktop and send the text messages from the desktop. The cuckoo framework [22] is based on the client server communication methodology, wherein the mobile device and the server communicate via remote procedure call (RPC) and remote method invocation (RMI). The cuckoo application could be run either remotely or locally and is based on Android platform. Another model, known as Ibis high performance programming system [23], is used as the basis for Cuckoo's communication component. However, in this system there is a decision making algorithm on how to offload the content to the cloud. Our model is different from [21,22,24] and [23] in several aspects as follows: It is designed based on the Android platform, which is not only designed for smartphone devices but also for the Android x86 emulator. It allows establishing remote connection from the physical device to the virtual machine. In our model we also support content off-loading to cloud, although it is Android based. Our system uses VNC instead of Ibis [23] and we support RPC and RMI protocols for performing remote operations. While Ibis provides remote access to the resources and acts as a middleware, it needs prior installation and may not be suitable for disconnected operations. VNC on the other hand is available for all platforms and easy to port on new platforms.

Our work is closely related to [24], as we share similar objectives of virtualizing the Android Application in android x86. In [24], the model, although designed for a different application than ours, is implemented based on virtual connection for Android. As there were multiple images running the same functionality, the processing time takes less than 1 second to open a PDF file in Android x86 images hosted in a data center, while the average time for opening the same PDF file takes 10 s. In our work, we propose a similar idea of mobile virtualization system which is based on Android x86 virtual machine. We use the hosted infrastructure for achieving virtualization with the intention of deploying our application in the cloud. A similar approach has also been proposed by [25], which is an Android application for performing object recognition, using the camera sensors of the smartphone. The implementation in [25] is based on the Ibis middleware and hence has the same drawbacks as the Cuckoo model which is also based on the client server communication methodology. Although we share some features with [24–26]our model is rather advantageous to both studies from the virtualization aspect. With virtualization, the virtual machine layer is located between the physical hardware and the operating system, which minimized the overhead of code generation and processing. Also virtualization supports multiple operating systems. In comparison to [9,11] and [12], Our approach provides much higher accuracy in almost real time. This means the user in our system does not need to wait for the result after he/she eats the meal, which could be too late to alert them about the amount of calorie in the plate. In the next section, we present the proposed system in details.

3 Proposed system

In this section, we discuss our proposed system. We first introduce the reader to the concept of virtualization in Sect. 3.1. Then, in Sect. 3.2, we provide details about our proposed model using Android virtualization mechanism (Fig. 1).

3.1 Android virtualization mechanism

Figure 2 shows a high level view of our virtualization model. In this model, we are able to run our Android application "eat healthy stay healthy (EHSH)" in the cloud with Android x86 image. An android based smartphone normally runs on Android ARM processor and the Android operating system is specifically built to be compatible with the ARM processor. The Android x86 emulator mimics all of the hardware and software features of a typical mobile device. It runs a full Android system stack, down to the kernel level that includes a set of preinstalled applications that can be accessed from the user's applications. For a smartphone, it emulates the mobile environment in a manner that the user always feels as if the application is being run on the Android smartphone, but in reality it is virtually running on the remote server (Android x86) with proper synchronization between the local client (smartphone) and the remote server.

Running the mobile application on virtual systems could be achieved through different approaches. The first approach is to implement the mobile application in the kernel layer (bare-metal architecture). Bare-metal architecture installs the virtualization layer directly on a clean x86-based system. A hypervisor being a virtual machine manager allows multiple operating systems to share a single hardware host. Each operating system appears to have the host's processor, memory, and other resources all to itself. However, the hypervisor is actually controlling the host processor and resources by allocating what is needed for each operating system in turn and making sure that the guest operating systems (virtual machines) cannot disrupt each other. The main challenge of this method is that Android supports only one display and keypad device since Android is mainly designed for smartphones. Hence we would not be able to enter any entries in the text field in our application. In contrast, a hosted architecture installs and runs the virtualization layer as an application on top of an operating system and supports the broadest range of hardware configuration.

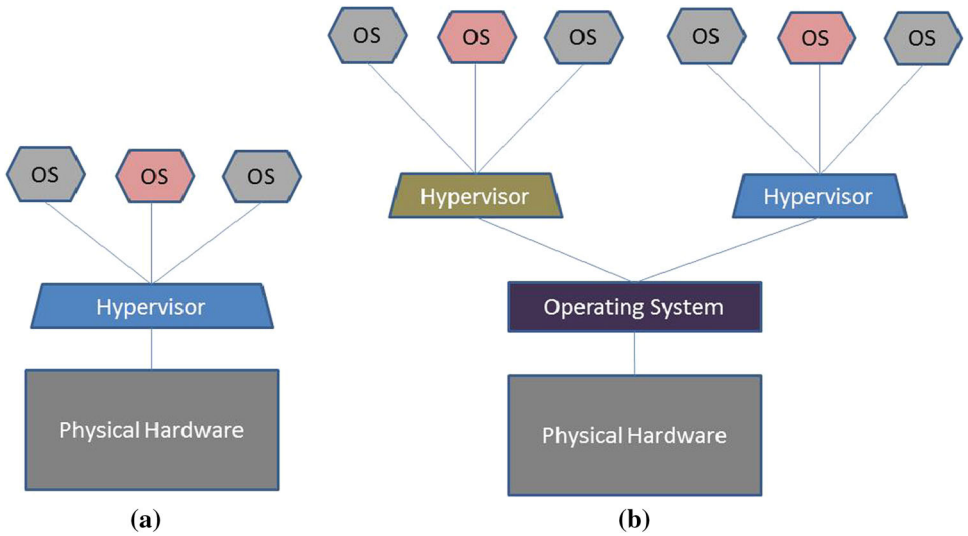

Fig. 1 Bare-metal **a** virtualization versus Hosted **b** virtualization

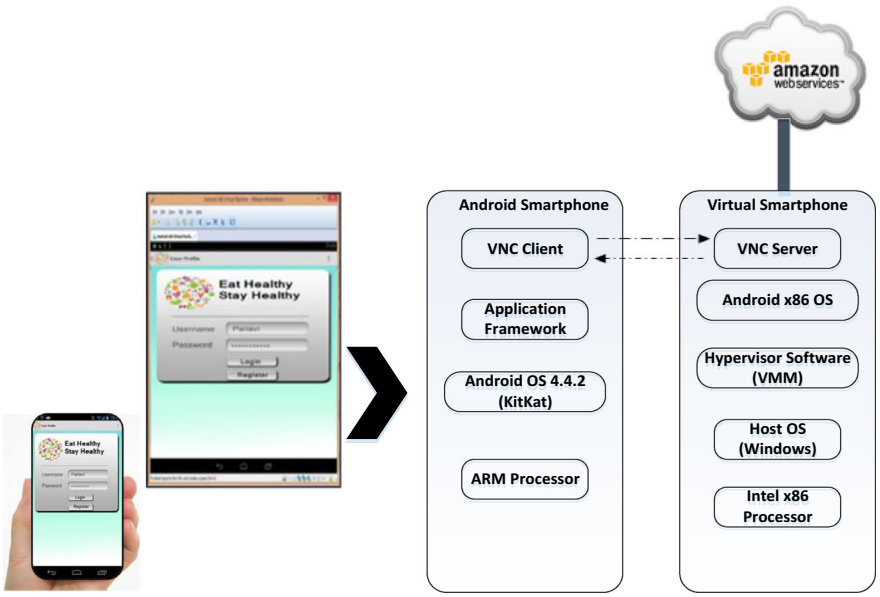

Fig. 2 Cloud-based virtualization on Android for eat healthy stay healthy application

3.2 System mechanism

Our model uses both the hosted and hypervisor models and we used the hosted architecture for publishing our Android application in the cloud, because the bare-metal architecture will not allow us to publish the virtual machine structure in the cloud. By so doing, the users of EHSH can control their virtual smartphone operations through a dedicated client application installed on their smartphones while the processing of the application continues to run on the virtual Android image even if the user is disconnected for any unexpected reason. For establishing remote connection from the Android physical device to the Android x86 image, we used VNC and installed the VNC server on the Android x86 processor and VNC client on the physical smartphone as shown in Fig. 2. "remote desktop protocol (RDP)" is semantic, which means it is aware of controls, fonts and other graphic primitives. So while rendering changes of screens across the network, for example between two consecutive frames, if the only change is that a button is added, then RDP will send the location of the button on the screen, size and colour. While in the case of VNC, actual images are sent over the network. VNC is available on all platforms while RDP is specially made for the Windows platform. VNC also supports file sharing and is platform-independent. The simplicity of the protocol makes it easy to port on new platforms. Hence we are using VNC for our model, as we have Android ARM processor on physical smartphone and Android x86 on a virtual machine. Furthermore, VNC provides flexibility to establish remote connection to Android. Also, in the bare-metal virtualization architecture, the mobile application is implemented in the kernel layer. The main challenge of this method is that,

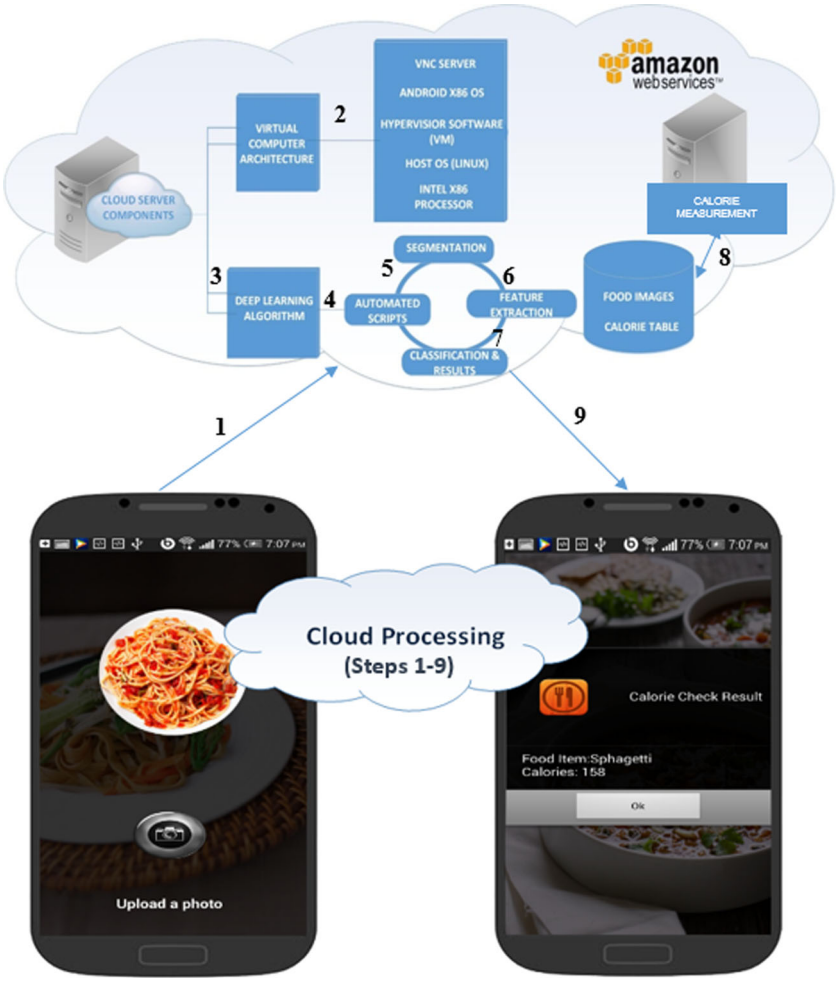

Fig. 3 Cloud-based virtualization on Android for EHSH application

Android supports only one display and keypad device since android is mainly designed for smartphone. Hence the user will not be able to enter any entries in the text field (especially in the login credential section) in our application. Hence we used the hosted architecture wherein this problem could be addressed. So, we have applied virtualization to the android application (EHSH) and the end result is that, we are able to establish a virtual cloud session on the mobile device as elaborated in Fig. 2. Apparently the login page is being displayed on the mobile screen and the user always feels as if the application is being run on the Android smartphone, but in reality it is virtually running on the remote server (Android x86) with proper synchronization between the local client (smartphone) and the remote server. We have used this methodology to implement our android application in the cloud, which is further explained in the next section.

4 Implementation

In this section, we describe the implementation and operation of our system. The aim of this section is to illustrate the use of VNC and Android platform for the purpose of visualization of our application. We also provide brief details of the use of image segmentation and deep learning based on our previous work [4–6].

We implement our system such that both the smartphone and the virtual machine's operating system have Android. The smartphone used in the implementation is Samsung Galaxy S4 running Android 4.4.2 (KitKat) while the Android x86 emulator has the Android 4.4 (Kitkat) image. We have stored the replica of our application in the Android x86 virtual machine hosted on Amazon Web services; see Fig. 3. The VNC client on the physical smartphone is used to connect to the VNC server on the virtual machine. Once the connection is established, VNC transmits all the events and Android images similar to streaming a video. For handling connections from multiple users, we have assigned a virtual image to each user. Hence the user session remains consistent and the user never realizes that the session is being run on the remote machine. Hardware virtualization (hypervisor) is between the physical hardware of the virtual machine and the Android x86 OS, which enables the deployment of a replica VM. To maintain consistency between the Android smart-

phone and the android x86 replica, any new file that has been added to the file system of the smartphone (from sources other than the Internet in the case of continuously connected operation) is also sent to the replica. Running a VNC client on the physical smartphone is used to connect to the VNC server on the virtual machine. Once this occurs, VNC transmits all the events and android images just like streaming a video. The first component of our system is the Virtual Computing Architecture, which contains all parts of the virtualization architecture. The second component is food image processing and recognition, which performs feature extraction, classification and calorie measurement. This process is fully automatic and does not require any intervention from the administrator or the user. After the features are extracted and the food image is classified, the result is compared with the database to generate the corresponding calorie of the food item.

The system makes use of a set of key functionalities such as the use of food pictures, image processing by segmentation, and image analysis. The details of the image segmentation and classification are provided in [4–6], and will not be repeated here. Furthermore, we apply deep learning neural networks to increase our food recognition accuracy[27,28]. We do this by initially capturing a set of images of one particular class and labelling them with an object name-set (object being Spaghetti, in Fig. 3). These would be our set of relevant (positive) images. After we have captured the image-sets we train the system with these images. As training takes place virtually on the server, we have the much needed processing power, so the system gets trained quickly (depending upon the number of images in a class). As part of the second step of training, we now re-train the system with the set of negative images (images that do not contain the relevant object), in our case we trained the system with the background images, so it does not categorize them as part of the mentioned class. Once the model file is generated from the training, we load it into the Android Application and test it against the images captured and submitted by the user. The system then performs the image recognition process and generates a list of the probabilities against the label name. The label with the highest probability is prompted to the user in the dialog box, to confirm the object name. Once the object name is confirmed, the system performs the calorie computation part as fully described in [29] and shows the computed calories to the user. The final result, containing the food item name and the corresponding calorie value are send to the android API.

5 Experimental results

This section reports on the experimental evaluation of the performance of the proposed model. We compare the results of running our application's algorithms such as deep learning, image segmentation, and image processing on three different configurations of cloud servers and a local server connected to the smartphone.

The setup of our testing is as follows: We used seven different food classes, each class containing 20 test images. For each image belonging to a class, we have recorded the recognition accuracy, recognition success, and the time, in seconds, of processing the results to the user. We ran the first set of experiment on a single instance cloud server (1 ECU and 2GB RAM) in which we have included the top 5 results of images. The average timings for the implementation of the above algorithm were recorded between 16 and 18 s. For most of the image classes, the recognition success rate was 18 out of 20 images with accuracy varying between 59.05 % and 100 %. Except for a couple of images, the recognition rate of most food items recorded a recognition rate above 95 %.

In the second setup, we performed the experiments on a cloud server with two instances each with 4 GB RAM and 3 ECUs. For this experiment, we have included the top 5 results for images. The average timings for the implementation of the above algorithm were recorded between 15 and 16 s. The recognition success rates were similar to the case of one instance cloud server and the accuracy rates were between 59.05 and 99.9487 %. In the third setup, we performed the experiment using 4 instances of the cloud server with 15GB RAM and 13 ECU. With this relatively powerful cloud server, there was significant improvement in the timings and the average processing time was 14.60 s.

The experiment was also performed on the local server with 1.7GHz and a shared 1.78 GB RAM. We have noticed a significant difference with respect to the time and the recognition results in this case. The average timing for the implementation of the algorithm ranged between 24 and 28 s, with varying results on each run. Also the recognition was 16 out of 20 images. The results of the processing time of the above experiments are shown in Table 1. As we can see, the time has improved significantly on all cloud configurations compared to the local server. As expected, the time improvement increased as the processing power of the cloud sever increased. For some food items, the boost in time processing reached to 50 % percent improvement. One aspect of the results that did not provide significant improvement is the accuracy of food recognition as we can see in Table 3. Even when running the system on the most powerful could server, there were no improvements. The reason for such results is the number of images used in the experiment. The training algorithm needs a much bigger data set to show the difference between the two experiments. This is a limitation on the current experiment; however, we are towards adding more images for further investigation. For comparing the accuracy of the system, We have run the method with deep learning

Table 1 Comparison results of the application processing time

Food name	Time (s)			
	Single instance cloud server (2GB and EBS storage)	Two instances cloud server (4GB and EBS storage)	Four instances cloud server (15GB and EBS storage)	Local Server (1.7 GB RAM)
Spaghetti	17.61	15.85	14.62	24.2
Burger	17.47	16.07	14.6	27.41
Bread Slice	17.55	15.28	14.675	27.09
Banana	17.92	16.36	14.68	26.71
Strawberry	17.6	15.7	14.6	27.39
Pineapple	17.46	15.14	14.68	27.2
Cucumber	16.60	15.03	14.63	28.77

Table 2 Percentage improvement of processing time

Food name	Percentage improvement of processing time over		
	One instance cloud server versus local server (%)	Two instances cloud server versus local server (%)	Four instances cloud server versus local server(%)
Spaghetti	27	35	40
Burger	36	41	47
Bread Slice	35	44	46
Banana	33	39	45
Strawberry	36	43	47
Pineapple	36	44	46
Cucumber	42	48	49

Table 3 Comparison of the accuracy percentage

Food name	Accuracy (%)	
	Cloud (15GB memory)	Local server
Spaghetti	0.93	0.73
Burger	0.90	0.82
Bread Slice	0.98	0.96
Banana	1	0.74
Strawberry	1	0.98
Pineapple	1	0.97
Cucumber	0.85	0.59
Average	0.94	0.87

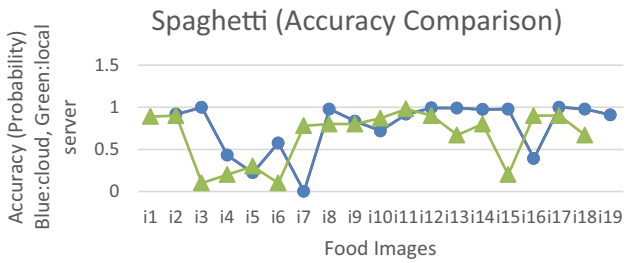

Fig. 4 Accuracy of cloud and local server on spaghetti

neural network on cloud and server and you can see the results in the table below. So by increasing the number of images we will get higher accuracy in cloud version. Also by updating the system periodically with unrecognized images, we will get better accuracy after a short time in cloud version in compare with local server (Table 2; Fig. 4).

Also, we have compared our time and accuracy results to other similar models. The models which we look to compare share the same goal of object recognition via mobile device and make use of the cloud computing models. However the approach they follow is different in terms of the recognition concept and even the cloud computing model. We have made two graphs, each including the comparison based on time and accuracy when comparing our model to other models.

In our simulations, the time taken (in seconds) is the overall time taken for the object recognition process (feature extraction, classification etc.) from the time the images are captured from the mobile device, till the point the final recognition results is displayed on the mobile screen. Matusiak [30]model, is a food object recognition model in a mobile phone application for visually impaired users. It makes use of color detection, object recognition module and light source detector, wherein the image resolution for recognition is

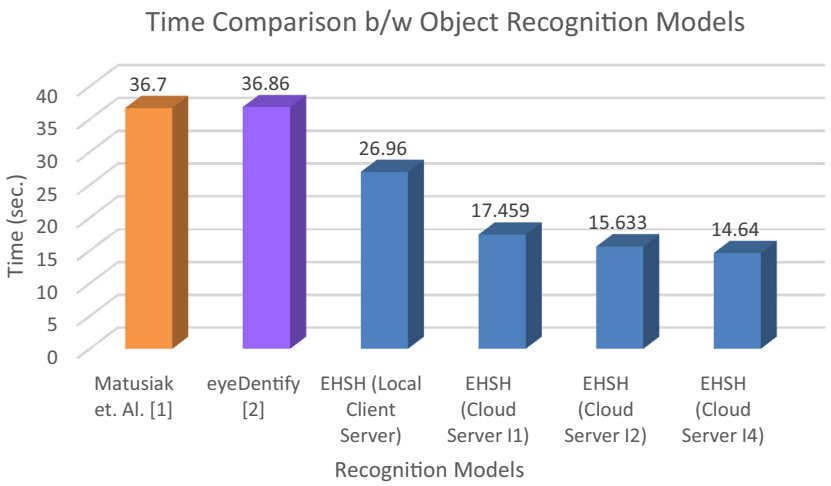

Fig. 5 Time Comparison across various recognition models

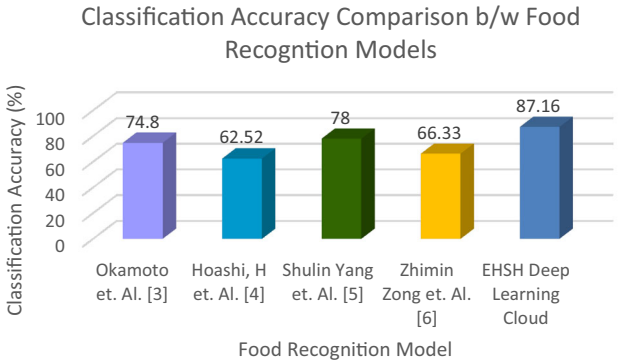

Fig. 6 Classification accuracy comparison between food recognition models

320 × 480 and is trained with the database of 30 food products. The object recognition time in this model is 36.7 seconds for each image. The application is implemented as a standalone application on the mobile device, without the use of any external server.

Eyedentify [31]was also designed to recognize objects. Similar to our method, the computation requirements for achieving the recognition task in [31] were not met by the smartphone alone. By using cyber foraging in which some computations were offloaded to the cloud, they were able to increase the application's responsiveness and accuracy and decrease the amount of energy used. In [31], Ibis middleware was used to build a distribution system, which was evaluated using eyeDentify, which performs object recognition. 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, this 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 by dividing them into circles, whereas our system processes images based on pixels. The circular region approach [31] misses the information not covered by the circles, whereas the pixel approach analyzes all the information in an image. Furthermore, in [31], 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. They had implemented it in a 8 node VU cluster wherein each cluster was dual CPU / dual core 2.4 GHz AMD OPetron 4 gb ram. It also ran on Linux Operating system. The overall time taken with the cyber foraging concept by [31], was 34.8 s.

As shown in Fig. 5, compared to [30] and [31], our model (EHSH) has an improved overall timing result. On the local client server model, the time taken for food object recognition was 26.96 s which further improved to 14.64 s with the use of the cloud based virtualization model.

Also, based on Fig. 6, we can observe that our model has an average classification accuracy of 87.16 % for the food objects. The existing food object recognition models like the Okamoto et. Al. model [32] has an overall accuracy of 74.8 %. In [32] they propose a mobile real-time eating action recognition system. They classify food items like meat, rice, pumpkin, bell pepper and carrot. Comparing [32] to our model, with the implantation of the deep learning in cloud based virtualization model, we are able to achieve 12.36 % better accuracy result with certain food objects even classified with an accuracy percentage of 100 % in single food portions. As shown in Fig. 2, we can observe similar accuracy improvements of 24.64 % with [33], 9.16 % with [34]and 20.83 % when compared to [35].

6 Conclusion and future work

This paper described a virtualization mechanism for real time multimedia-assisted mobile food recognition application in cloud computing. The key ideas and the overall system

implementation as well as experimental results to support the validity of the proposed mechanism were described. We were able to virtually run computationally intensive and time sensitive Android-based food recognition and classification application on the virtual system (Android x86) hosted in the cloud. With the proposed virtualization mechanism, we addressed several challenges related to the complex food recognition algorithm. Our experimental results of running the application on the cloud servers show significant improvements when compared with experiments running on the local server. The results show that the rate of food recognition when we run the application in the cloud servers is more than 94.33 % compared to 87.16 % when we run the application on a local server. Also, with our virtualization mechanism the results are processed up to 59 % faster than the case of running the application locally. One of the main contributions of the proposed system is that it allows users to control their virtual smartphone operations, through a dedicated client application installed on their smartphones, while the processing of the application continue to run on the virtual Android image even if the user is disconnected for any unexpected event. With our mechanism the mobile environment is emulated on the cloud in a manner that users always feel as if the application is being run on the smartphone, but in reality it is virtually running on the remote server in the cloud. This is rather significant to overcome the limited capability of smartphones to run intensive machine learning algorithms similar to those presented in this paper.

As for our plans for future work, we will be embedding the VNC server configurations inside our Android Application. Currently we are establishing the remote connection outside of our Android application, thereby allowing the user to access other applications. One of the main reasons for creating this unique VNC configuration being, there is a stage in our Android application wherein the user captures the image from the physical device. The captured images are further used for image recognition, which is processed remotely in the virtual machine. Currently the existing configuration does not re-establish the remote connection once broken, while capturing images. Hence the images are fetched from the gallery before the launch of the application. We will be addressing these issues in the near future. Another aspect of the results is related to the accuracy of food recognition as shown in Table 3. We have noticed that the accuracy percentage has improved marginally. Even when comparing the accuracy results with the most powerful could server. The reason for such results is the number of images used in the experiments. The training algorithm needs a much bigger data set to show the difference in the accuracy results. This is a limitation on the current experiments, however we are planning to increase our set of data images and perform further investigation to study the accuracy of the system.

References

1. Bamiah M., Brohi S., Chprat S., AB Manan J.L. : A study on significance of adopting cloud computing paradigm in healthcare sector. In: IEEE International Conference on Cloud Computing Technology, Applications and Management (ICCTAM), pp. 65–68, (2012)
2. Seoyoung, K., Jok-Soo, K., Soonwook, H., Yoonhee, K.: Towards effective sceince cloud provisioning for a large-scale high-throughput computing. Clust. Comput. J. **17**(4), 1157–1169 (2014). doi:10.1007/s10586-014-0371-2
3. Myoungjin, K., Seungho, H., Yun, C., Hanku, L., Hogyeon, C., Sungdae, H.: CloudDMSS robust hadoop-based multimedia streaming service architecture for a cloud computing environment. Clust. Comput. J. **17**(3), 605–628 (2014)
4. Pouladzadeh P., Bharat Peddi S.V., Kuhad P., Yassine A., Shirmohammadi S.: Mobile cloud based food calorie measurement. In: IEEE International Confernce on Multimedia and Expo Workshops (ICMEW), pp. 1–6, Chengdu, China (2014)
5. Pouladzadeh, P., Shirmohammadi, S., Bakirov, A., Bulut, A., Yassine, A.: Cloud-Based SVM for food categorization. Springer's J. Multimed. Tools Appl. (2014). doi:10.1007/s11042-014-2116-x
6. Pouladzadeh P., Shirmohammadi S., Yassine A.: Using graph cut segmentation for food calorie measurement. In: IEEE International Symposium on Medical Measurements and applications, pp. 1–6, Lisbon, (2014)
7. Hlavacek J., Bestak R.: Configuration of live migration for VoIP applications. In: MECHATRONIKA, 15th International Symposium, 5–7 Dec. pp. 1–4 (2012)
8. VMware "Understanding full virtualization, para- virtualization, and hardware Assist" White Paper Available at http://www.vmware.com/resources/techresources/1008. Accessed 14 Sept 2014
9. Kato Y., Suzuki T., Kobayashi K., Nakauchi Y.: A web application for an obesity prevention system based on individual lifestyle analysis. In: IEEE International conference on Systems, Man, and Cybernetics (SMC), pp. 1718–1723, Seoul (2012)
10. Chen H.C., Jia W., Li Z., Sun Y., Sun M.: 3D/2D model-to-image registration for quantitative dietary assessment. In: Bioengineering Conference (NEBEC), 38th Annual Northeast, pp. 95–96, 16–18 March (2012)
11. Sun M., Liu Q., Schmidt K., Yang J., Yao N., Fernstrom J.D., Fernstrom M.H., DeLany J.P., Sclabassi R.J.: IEEE International EMBS Conference on Determination of Food Portion Size by Image Processing, pp. 871–874, (2008)
12. Saeki, Y., Takeda, F.: Proposal of food intake measuring system in medical use and its discussion of practical capability. Knowl.-based Intell. Inf. Eng. Syst. **3683**, 1266–1273 (2005)
13. Martin, C.K., Kaya, S., Gunturk, B.K.: Quantification of food intake using food image analysis. Int. Conf. IEEE Eng. Med. Biol. Soc. **2009**, 6869–6872 (2009)
14. Zhua F., Mariappana A., Bousheyb C.J., Kerrd D., Lutesc K.D., Eberta D.S., Delp E.J.: Technology-assisted dietary assessment. Computational Imaging VI, Proceedings of SPIE-IS&T Electronic Imaging, SPIE vol. 6814, p. 681411 (2008)
15. Sobhy D., El-Sonbaty Y., Abou Elnasr M.: MedCloud heathcare cloud computing system. In: IEEE International Conference on Internet Technology and Secured Transactions, pp. 161–166 (2012)
16. Doan B.H. and Chen L.: Mobile cloud for assistive healthcare. IEEE Asia-Pacific Serices Computing Conference APSCC, pp. 325–332, (2010)
17. Yong, W.A., Cheng, A.M.K., Baek, J., Jo, M., Chen, H.: An auto-scaling mechinam for virtual resources to support mobile, pervasive, real-time healthcare applications in cloud computing. IEEE Netw. **27**(5), 62–68 (2013)

18. Harini M., Bhairavi K., Gopicharan R., Kirupa G., Vaidehi V.: Virtualization of healthcare sensors in cloud. IEEE International Conference on Recent Trends in Information Technology, pp. 663–667, (2013)
19. Hossain, M.S., Muhammad, G.: Cloud-based collaborative media service framework for health-care. Int. J. Distrib.Sens. Netw. (2014)
20. Hossain, M.S., Muhammad, G.:Cloud-assisted speech and face recognition framework for health monitoring. Mob. Netw. Appl., pp. 1–9 (2015)
21. Skurski, A., Swiercz, B.: VNC-based remote control for symbian OS smartphones. In: 16th International Conference on Mixed Design of Integrated Circuit and Systems, June 25–27, Lodi, Poland, MIXDES (2009)
22. Kemp, R., Palmer,N., Kielmann, T., Bal, H.: Cuckoo: a computation offloading framework for smartphones. In: Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, MobiCASE'10 (2012)
23. Van Nieuwpoort, R., Maassen, J., Wrzesińska, G., Hofman, R., Jacobs, C., Kielmann, T., Bal, H.: Ibis: a flexible and efficient java based grid programming environment. Concurr. Comput. **17**, 1079–1107 (2005)
24. Chen, E.Y., Itoh, M.: Virtual smartphone over IP. In: IEEE International Symposium on World of Wireless Mobile and Multimedia Networks (WoWMoM) (2010)
25. Kemp, R., Palmer, N., Kielmann, T., Seinstra, F., Drost, N., Maassen, J., Bal, H.E.: EyeDentify : multimedia cyber foraging from a smartphone. In: IEEE International Symposium on Multimedia (2009)
26. Alamri, A., Hassan, M.M., Hossain, M.A., Al-Qurishi, M., Aldukhayyil, Y., Shamim, M.: Evaluating the impact of a cloud-based serious game on obese people. Comput. Hum. Behav. **30**, 468–475 (2014)
27. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. (NIPS) (2012)
28. Srivastava, N., Salakhutdinov, R.: Multimodal learning with deep Boltzmann machines. In: Proceedings Neural Information and Processing System (2012)
29. Pouladzadeh, P., Shirmohammadi, S., Almaghrabi, R.: Measuring calorie and nutrition from food image. IEEE Trans. Instrum. Meas **63**(8), 1947–1956 (2014)
30. Matusiak, K., Skulimowski, P., Strumillo, P.: Object recognition in a mobile phone application for visually impaired users. In: The 6th International Conference on Human System Interaction (HSI), pp. 479–484, 6–8 June (2013)
31. Kemp, R., Palmer, N., Kielmann, T., Seinstra, F., Drost, N., Maassen, J., Bal, H.: eyeDentify: 11th IEEE International Symposium on Multimedia Cyber Foraging from a Smartphone, pp. 392–399, 14–16 Dec. (2009)
32. Okamoto, K., Yanai, K.: Real-time eating action recognition system on a smartphone. In: IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–6, 14–18 July 2014 doi:10.1109/ICMEW.2014.6890718
33. Hoashi, H., Joutou, T., Yanai, K.: Image recognition of 85 food categories by feature fusion. In: IEEE International Symposium on Multimedia (ISM), pp. 296–301, 13–15 Dec. 2010 doi:10.1109/ISM.2010.51
34. Yang, S., Chen, M., Pomerleau, D., Sukthankar, R.: Food recognition using statistics of pairwise local features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2249–2256, 13–18 June (2010)
35. Zong, Z., Nguyen, D.T., Ogunbona, P., Li, W.: On the combination of local texture and global structure for food classification. In: IEEE International Symposium on Multimedia (ISM), pp. 204–211, 13–15 Dec. (2010)

Parisa Pouladzadeh received her M.Sc from University of Ottawa in 2012, where her thesis was nominated for a best thesis award. Currently she is a Ph.D. student in the School of Electrical Engineering and Computer Science at the University of Ottawa, working on food recognition systems. Her other research interests include image processing, artificial intelligence and classification.

Sri Vijay Bharat Peddi was research assistant at the Discover Lab, School of Electrical Engineering and Computer Science at the University of Ottawa, Canada. He has completed his M.Sc degree in Electrical and Computer Engineering from University of Ottawa and is currently working in the Analytics division in IBM Canada. He has expertise in parallel computing, distributed cloud computing, mobile cloud computing and image recognition. He has previously worked as database administrator and has several years of experience working with cluster environments in Oracle and Sybase databases.

Pallavi Kuhad has completed her M.Sc degree in Electrical and Computer Engineering from University of Ottawa. She was a researcher at the School of Electrical Engineering and Computer Science of the University of Ottawa, Canada. She is currently working in Scientific Research & Experimental Development Team at Ernst & Young (EY). She has several publications to her name and specializes in the area of food recognition, deep learning, auto-calibration approaches for calorie measurement in food images and computer vision algorithms. She has previously worked on enterprise IT systems development and has 3 years of software industry experience working with enterprise software and networking technologies.

Abdulsalam Yassine received his Ph.D. and M.Sc. in Electrical and Computer Engineering from University of Ottawa–Canada in 2010 and 2004, respectively, his B.Sc. in Electrical Engineering from Beirut Arab University?Lebanon in 1993. He is currently a Post-doctoral fellow at DISCOVER Laboratory at the School of Electrical Engineering and Computer Science, University of Ottawa. Between 2001 and 2013, Dr. Yassine was member of the technical staff in the Wireless Communication Division at Nortel Networks and later at Alcatel-Lucent, Ottawa, Canada. His current research interests are mostly focused on Systems and Networks, Multimedia, Artificial Intelligence (AI), Smart Environments, and Smart Grids.

Shervin Shirmohammadi received the Ph.D. degree in Electrical Engineering from the University of Ottawa, Canada, where he is currently a Professor with the School of Electrical Engineering and Computer Science. He is Director of the Distributed and Collaborative Virtual Environment Research Laboratory, and affiliate member with the Multimedia Communications Research Laboratory, conducting research in gaming systems and virtual environments, video systems, and multimedia-assisted biomedical engineering. The results of his research have led to more than 250 publications, over 60 researchers trained at the postdoctoral, Ph.D, Master's, and research engineer levels, over 20 patents and technology transfers to the private sector, and a number of awards and prizes. He is the Associate Editor-in-Chief of the IEEE Instrumentation and Measurement magazine, Senior Associate Editor of ACM Transactions on Multimedia Computing, Communications, and Applications, an Associate Editor of IEEE Transactions on Instrumentation and Measurement, and was an Associate Editor of Springer's Journal of Multimedia Tools and Applications from 2004 to 2012. Dr. Shirmohammadi is a University of Ottawa Gold Medalist, a licensed Professional Engineer in Ontario, a Senior Member of IEEE, and a Lifetime Professional Member of the ACM.