Mobile Multi-Food Recognition Using Deep Learning

PARISA POULADZADEH, University of Ottawa
SHERVIN SHIRMOHAMMADI, University of Ottawa, Canada, and Istanbul Sehir Univesrity, Trukey

In this article, we propose a mobile food recognition system that uses the picture of the food, taken by the user's mobile device, to recognize multiple food items in the same meal, such as steak and potatoes on the same plate, to estimate the calorie and nutrition of the meal. To speed up and make the process more accurate, the user is asked to quickly identify the general area of the food by drawing a bounding circle on the food picture by touching the screen. The system then uses image processing and computational intelligence for food item recognition. The advantage of recognizing items, instead of the whole meal, is that the system can be trained with only single item food images. At the training stage, we first use region proposal algorithms to generate candidate regions and extract the convolutional neural network (CNN) features of all regions. Second, we perform region mining to select positive regions for each food category using maximum cover by our proposed submodular optimization method. At the testing stage, we first generate a set of candidate regions. For each region, a classification score is computed based on its extracted CNN features and predicted food names of the selected regions. Since fast response is one of the important parameters for the user who wants to eat the meal, certain heavy computational parts of the application are offloaded to the cloud. Hence, the processes of food recognition and calorie estimation are performed in cloud server. Our experiments, conducted with the FooDD dataset, show an average recall rate of 90.98%, precision rate of 93.05%, and accuracy of 94.11% compared to 50.8% to 88% accuracy of other existing food recognition systems.

CCS Concepts: • **Computer systems organization → Real-time operating systems**

Additional Key Words and Phrases: Mobile food recognition, deep learning, cloud computing

ACM Reference Format:
Parisa Pouladzadeh and Shervin Shirmohammadi. 2017. Mobile multi-food recognition using deep learning. ACM Trans. Multimedia Comput. Commun. Appl. 13, 3s, Article 36 (August 2017), 21 pages.
DOI: http://dx.doi.org/10.1145/3063592

1. INTRODUCTION

Healthy eating, including counting calories and nutrition, is increasingly becoming more important for most consumers everywhere in the world. According to a study organized by Canadian Obesity Network [www.obesitynetwork.ca], around one in four adults and one in 10 children in Canada are now living with obesity. This means roughly six million Canadians are affected by this condition, far more than people living with diabetes, heart disease, arthritis, chronic lung disease, or cancer. Similar trends exist all over the world [http://www.who.int], leading to calls for designing personal assistive systems to help consumers manage their eating. This is why, in recent years,

Authors' addresses: P. Pouladzadeh, Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER Lab), School of Electrical Engineering and Computer Science, University of Ottawa, Canada; email: ppoul081@uottawa.ca; S. Shirmohammadi, Distributed and Collaborative Virtual Environments Research Laboratory (DISCOVER Lab), School of Electrical Engineering and Computer Science, University of Ottawa, Canada; email: shervin@discover.uottawa.ca; and Multimedia Systems Laboratory, Istanbul Sehir University, Istanbul, Canada; email: shervinshirmohammadi@sehir.edu.tr.

DOI: http://dx.doi.org/10.1145/3063592

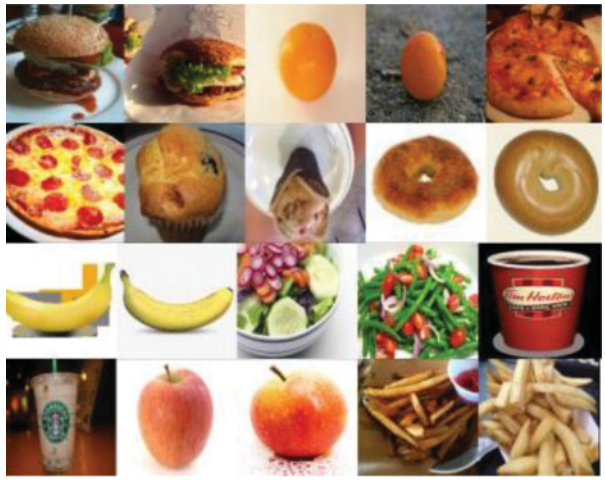


Fig. 1. Examples of single-item foods.

we have seen an explosion of health-related mobile applications. Most of these mobile applications automatically record physical activities carried out and calories burnt each day. Some of them also calculate the calorie intake. A survey of existing methods for mobile calorie intake measurement, which shows the pros and cons of the related works, is published in pouladzadeh and Shirmohammadi [2016]. Among these, apps that run on mobile devices seem to be the most practical ones, and they are the subject of this work.

Previously, we proposed such mobile health applications in Pouladzadeh et al. [2016] that were able to recognize single-item foods, by capturing the image of the food and further calculating the total number of calories in the plate of food. In those works, in the first phase, we made use of four feature vectors: size, shape, color, and texture, during the feature extraction phase. In the second phase, we used a deep neural network as a classifier. Once we trained the deep neural network with food images from the FooDD dataset [Pouladzadeh et al. 2015], the system then generated a model file. To make implementation practical, the image processing, classification, and recognition steps were performed in the cloud [Peddi et al. 2015].

While the above system worked better than existing ones, similar to other previous work, it assumed that the food image contained only one food item, as shown in Figure 1. In other words, it only works for single-item food.

However, in real-life scenarios, a food image of a meal mostly includes multiple food items; that is, it is multi-item food as shown in Figure 2.

Despite the importance of multi-item food recognition, there are only a few research articles on this problem, as reviewed in the related work section below. However, similar to most previous work, they still use the classical visual features like SURF and color histogram instead of Convolution Neural Networks features. All in all, food recognition is a challenging task. First, collecting a huge dataset for training the system accurately is important but difficult. Second, there can be significant intra-class variations in the observed food items. For example, same food can have multiple visual appearances. Finally, presence of occlusions around food items adds extra complexity for recognition, as the same food might be served in a bowl or wrapped within a paper cover.

In this article, we propose a system for automatically detecting multi-item foods and estimating the calories from a given food image. Such a model can then be ported to mobile devices, serving as a way to automatically record calorie intake. The rest of the

Fig. 2. Multi-item food.

article is organized as follows: In Section 2, we present the related work. Section 3 gives our proposed system. In Section 4, we describe the implementation and experimental results. Finally, the conclusion is presented in Section 5.

2. RELATED WORK

The task to recognize what the food is in one image could be categorized into several problems. Most existing research work in food recognition assumes that only one food item is present in each image. Here, we review some of these works. In Bossard et al. [2014], researchers created a food database named Food 101 containing 101 different food categories. For each category, there were 1,000 images. First, they extracted the color features for super-pixels on each image. Then, all the super-pixels were clustered into groups using Random Forest based on their respective Fisher vector encoded feature vectors to obtain discriminative components across all the images, and they achieved a test accuracy rate of 50.8%. Kagaya et al. [2014] used a trained Convolutional Neural Network (CNN) for both food and non-food detection. They built a food database of 170,000 images containing 10 popular food items. Using sixfold cross-validation, the optimal CNN hyper parameters related to the number of layers, the pre-processing, and the training were decided. Experiments showed that CNN outperformed all the other baseline classical approaches by achieving an average accuracy rate of 73.7%. In Zhu et al. [2015], researchers proposed a method for dietary assessment to automatically identify and locate food in a variety of images. Two concepts were combined in their algorithm. First, a set of segmented objects were partitioned into similar object classes based on their features, and for solving the problem, they applied different segmentation methods. Second, automatic segmented regions were classified using a multichannel feature classification system, and SVM was used as their classifier. The final decision was obtained by combining class decisions from individual feature.

Unfortunately, few research projects have been conducted on multiple-food recognition. In Matsuda et al. [2012], the authors presented the first work ever done on multiple-food recognition. First, candidate regions generated by the whole image, deformable part model, circle detector, and JSEG region segmentation were integrated to create a candidate set of bounding-box regions. Second, various kinds of image

features—including a bag of features (BoF), a histogram of oriented gradient (HOG), Gabor texture features, and color histograms—were computed for each bounding box region in the candidate set. The evaluation values of each region belonging to all the given classes were computed using trained SVM classifiers by multiple kernel learning. They trained the SVM models on their own dataset, which contained 9,132 food images of 100 different food categories; an accuracy rate of 65.6% was reported on their test set. In Kawano and Yanai [2013], a mobile application called FoodCam was developed for real-time multiple food recognition. In this system, users needed to draw bounding circle over food items on the screen. This bounding circle was adjusted to the food region by a GrubCut-based segmentation method. For each segmented region within the bounding circle, Fisher Vector encoded feature vectors of color histograms, and histograms of gradients were computed and passed into SVM classifiers to predict the food item. An accuracy rate of 51.9% was reported on their test set.

The authors of Zhang et al. [2015] developed a mobile application for multiple-food recognition of 15 food categories on the phone without any user intervention. First, the user took a photo of a food plate, and a cropped 400×400 food image was uploaded to the server for food recognition. On the server side, the food image was first segmented into possible salient regions, and these regions were further grouped based on the similarity of their color and their HOG and SIFT feature vectors. Normally, a typical food image yielded about 100 salient segment regions. They collected 2,000 training images for 15 classes with a mobile phone camera, since they had discovered that the model trained with images downloaded from the Internet could not generalize well for images taken on mobile phones. They trained a linear multiple-class SVM classifier for each class using the Fisher vector encoded feature vectors (including SIFT and color features) of salient regions. They reported a top-1 accuracy rate of 85% when detecting 15 different kinds of foods in their experiments.

More recently, Zhang et al. [2016] proposed a multi-task system that can identify dish types, food ingredients, and cooking methods from food images with deep convolutional neural networks. They built up a dataset of 360 classes of different foods. To reduce the noises of the data, which was collected from the Internet, outlier images were detected and eliminated through a one-class SVM trained with deep convolutional features. They simultaneously trained a dish identifier, a cooking method recognizer, and a multi-label ingredient detector. They share a few low-level layers in the deep network architecture. The proposed framework shows higher accuracy than traditional method with handcrafted features.

In Akbari Fard et al. [2016], authors introduce an automatic way for detecting and recognizing the fruits in an image to enable keeping track of daily intake automatically using images taken by the user. The proposed method uses state-of-the-art deep-learning techniques for feature extraction and classification. Deep learning methods, especially convolutional neural networks, have been widely used for a variety of classification problems and have achieved promising results. The model has achieved an accuracy of 75% in the task of classification of 43 different types of fruit.

In Singla et al. [2016], authors report experiments on food/non-food classification and food recognition using a GoogLeNet model based on deep convolutional neural network. The experiments were conducted on two image datasets, where the images were collected from existing image datasets, social media, and imaging devices such as smart phone and wearable cameras. Experimental results show 83.6% accuracy for the food category recognition. They mention the main reason for not achieving a higher recognition accuracy on certain types of food images is the complex mixture of food items in the image and the high visual similarities between some images across categories. Shimoda and Yanai [2016] propose an intermediate approach between the traditional proposal approach and the fully convolutional approach. They especially propose a

method that generates high food-ness regions by fully convolutional networks and back-propagation-based approach with training food images gathered from the Web. In their work, they achieved reduced computational cost while keeping high quality for food detection. In Su et al. [2016], they propose a game theoretic resource allocation scheme for media cloud to allocate resource to mobile social users through brokers. In their work, a good framework of resource allocation among media cloud, brokers, and mobile social users is presented. Media cloud can dynamically determine the price of the resource and allocate its resource to brokers.

Finally, in Dehais et al. [2016], authors propose a method to detect and segment the food dishes in an image. The method combines region growing and merging techniques with deep CNN-based food border detection. A semi-automatic version of the method is also presented that improves the result with minimal user input. The proposed methods are trained and tested on non-overlapping subsets of a food image database, including 821 images, taken under challenging conditions and annotated manually. The automatic and semi-automatic dish segmentation methods reached average accuracies of 88% and 92%, respectively.

3. OUR PROPOSED SYSTEM

Recognizing multiple food items and categorizing them is a very challenging task. Our approach is to learn about a detector for each food category from a set of training images with only binary labels indicating whether the image contains this food category or not. We model an image as a set of overlapping rectangular windows. The binary image label $y = 1$ specifies that the image contains at least one instance of the target category, and the label $y = -1$ specifies that the image contains no instances of the category. Since instance labels are not provided, we perform region mining with submodular cover optimization to find the positive instances, which are rectangular windows containing the target food category. With the positive instances, we can train a classifier for each food category. In this section, we propose an algorithm that jointly detects food item combinations on a plate. In this algorithm, each of the training images contains only a single food item, while the testing images may contain multiple food items. Our proposed algorithm will predict the labels of food items and estimate the amounts of calories represented on the plate. The diagram of the algorithm is shown in Figure 3. The following is a detailed description of each step of our proposed algorithm and a brief explanation of the reason why certain methods and parameters have been chosen.

3.1. Bounding Circle

In the first step, for having a more accurate system, after taking a picture with the mobile device, it is better to remove the background from the main objects as much as possible and then save the new image. To do so, we ask the user to draw a bounding circle around the plate of the food, shown in Figure 4, to make it simpler for the next processing steps. We should clarify that bounding circle means removing the background area from the image and only keeping the food plate. So, for multiple food items, the bounding will happen only once for the whole plate, and there is no need to circle each food item separately.

3.2. Sending Data to Cloud

To reduce the processing time and battery usage for the mobile device, we have done our processing in the cloud. In this step, we use the Elastic map reduce (Amazon EMR), which is described in detail in Peddi et al. [2015]. The method further uses Hadoop, to distribute the data and process it across resizable Amazon EC2 instances. Hadoop uses a distributed processing architecture called Map Reduce, in which a task is mapped to a set of servers for processing. The results of the computation performed by those

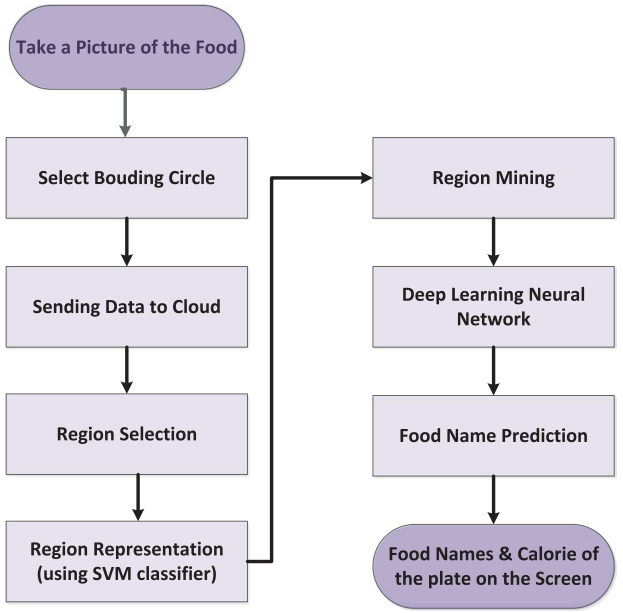


Fig. 3. Block Diagram of Our Proposed Food Recognition System.

Fig. 4. Bounding circle.

servers are then reduced to a single output set. One node, designated as the master node, controls the distribution of tasks to the slave nodes. The overview of the model is shown in Figure 5.

As shown in Figure 5, initially a request is sent to Amazon EMR to start the cluster. Once Amazon EMR creates the Hadoop cluster with a master instance group and core instance group, the master node is added to the master instance group. The slave node is then added to the core instance group. Our system deals with a huge set of images and processes them to derive results in various formats. Managing such a bulk of data can become a tedious job. To overcome this, we perform the processing and the storage in a cloud-based platform. Using the cloud platform not only satisfies the computational complexity constraints of our system but also helps achieving higher

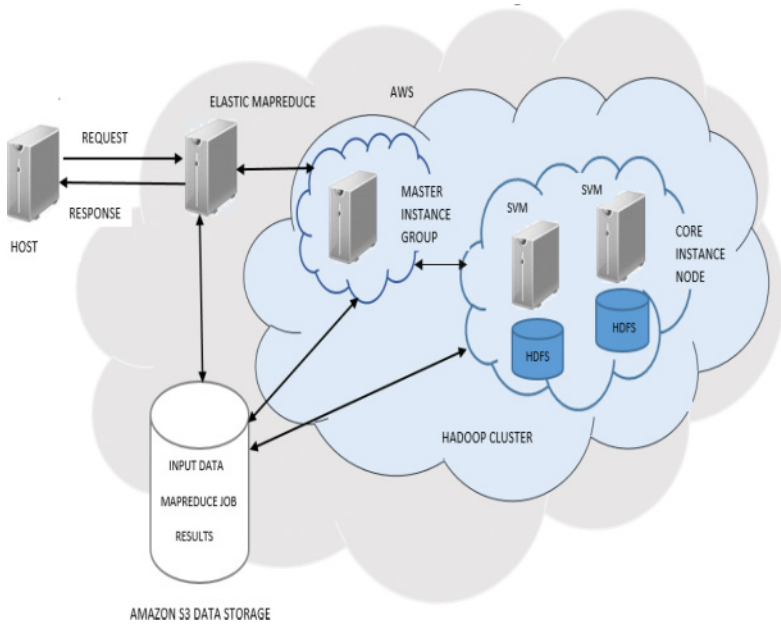


Fig. 5. Hadoop Integration on Amazon EC2/AWS. HDFS: Hadoop Distributed File System.

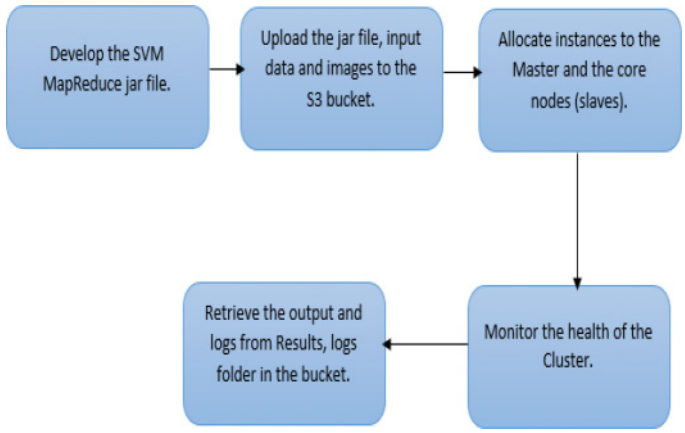


Fig. 6. Implementation of Map Reduce.

accuracy for food classification. The data, including the images, non-trained, trained, and non-tested results, logs, and the Map Reduce job, are all stored in the Amazon S3 bucket, which is an Internet storage service that is secure, reliable, scalable, fast, and inexpensive. Amazon S3 stores data as objects within buckets. An object is composed of a file and optionally metadata that describes the file. To store the file, one must upload the file to the bucket. In our system, we create a bucket named Food Recognition-test and made four subfolders in the bucket. The job folder contains the SVM Map Reduce jar file, the data folder contains the input file (svm_data.train), and logs folder saves the log info whenever the Map Reduce job is run. The implementation algorithm is shown in Figure 6.

As shown in Figure 6, the Map Reduce implementation has five stages. In the first stage, we develop the SVM Map Reduce jar file. This includes the mapper class, the reduce class, the combiner class, and the main Map Reduce java class. Since the segmentation is done with the help of Support Vector Model, we have implemented it with SVM training and testing classes, which we further call from the main Map Reduce class. In the second stage, we store the data and the jobs in the bucket. In the third step, we provision an Amazon EMR cluster, wherein the instance size of the node directly impacts the performance of the cluster. This further depends on the workload being CPU intensive or disk (I/O) intensive or memory intensive. For memory intensive jobs, m1.xlarge or one of the m2 family instance sizes have enough memory and CPU power to perform the work [http://media.amazonwebservices.com]. In our case, since we ran the job with fewer samples, we assumed that it would take lower memory and lower CPU. After the cluster is provisioned, we launch the job workflow, which further includes provisioning cluster, running bootstrap action, running the job flow, and shutting down the cluster. In the fourth step, once the job flow is started, we can use the cluster monitoring interface in AWS (Amazon Web Services) to further monitor the cluster status, Map Reduce job, node status, IO, and the hardware configuration. Finally, in the fifth step, the results are stored in the results folder, and the logs are stored in the logs folder of the bucket. We should mention that the main processing part of the method happens at the server side, where the resource/power is not limited due to Cloud scalability. The server side is responsible for training, detection, and recognition phases. The thin client side is only responsible for capturing the image, sending it and receiving the results, which does not need huge computation.

3.3. Region Collection

In this step, we use Selective Search [Fujishige 2005] to generate region proposals for multiple-food detection, and the algorithm we have chosen is described below. Selective Search has been broadly used as a detection method by many object detectors. The purpose of this method is to group super-pixels to generate a hierarchy of small to large regions. Usually, Selective Search can generate more than 1,000 regions for a single image. We further filtered the candidate regions based on their size and aspect ratio. Regions were removed if they were too small (i.e., if the area was less than 1/100 of the original image) or if they had a ratio of length to width that was either too large (i.e., larger than 20) or too small (i.e., less than 1/20 of the original image). Because of this simple, logical filtering process, we successfully decreased the number of proposed regions per image from about 1,000 to about 150. This decrease significantly improved the speed and accuracy of the subsequent procedures.

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

ALGORITHM 1: Region Selection [Fujishige 2005]

Generate initial regions.
//Generate as many regions as possible and each of them belongs to one object.
Merge similar regions into larger ones, until only one region is left.
Merge criteria based on the color and texture similarities.
//The similarities are measured by the similarity metric, which is defined as follows:
Choose the color space from RGB, HSV and Lab color spaces.
Compute the HOG-like features for texture similarity measurement.
Compute size component in the similarity metric.
Measures the similarity of shape between two regions.
The function of the similarity = linear combination of all above metrics.
Output regions contain candidate objects.

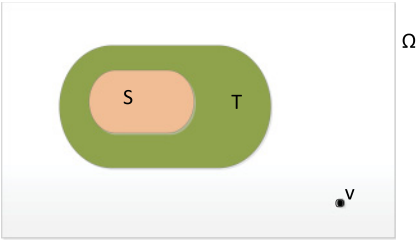


Fig. 7. Submodular Set Function.

3.4. Region Description

Extracting features to represent the regions in the featured space is an important component of object detection. In our experiments, we use the Caffe [Jia et al. 2014] deep-learning framework to compute the feature vectors of the proposed regions. Then the pixel values of the region are subtracted from the mean and then normalized to reach between [0:0, 1:0]. The normalized pixels' values are then forward propagated through the neural network, and a 4,096-dimension feature is extracted by using the output of the last convolutional layer in the model. Since we decreased the number of proposed regions from about 1,000 to about 150 per image, after extracting features for all these regions, a feature matrix of $150 \times 4{,}096$ is computed for each image. This feature matrix is used as the input data for the region mining process.

3.5. Region Mining

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

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

(a) S can well represent the category.
(b) S is small enough for fast training.

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

A set function $F : 2^{\Omega} \rightarrow R$ is submodular if it satisfies diminishing gains [Awolsey. 1982]. That is, for any

$S \subseteq T \subseteq \Omega \backslash \{v\}$, it holds in Equation (1):

$$F(T \cup \{v\} - F(T) \leq F(S \cup \{v\}) - F(S), \forall v \in \Omega. \quad (1)$$

We define a covering score $C_{I,t}(S)$ for each image I. The score is determined by a covering threshold tand the number of regions from R_I that are in the set of neighbors

of S, that is, N(S), which is described in Equation (2).

$$C_{I,t}(S) = \min\{t, |N(S) \cap R_I|\} \tag{2}$$

The covering score $C_{I,t}(S)$ measures how many regions in R_I are the near neighbors of S. If there are many regions in R_I that are the near neighbors of S, then set S can well represent the image I, and we say that set S covers image I. The covering score F(S) from Equation (3) for the whole set S is then defined as sum of scores on all positive images P of that category:

$$F(S) = \sum_{I \in P} C_{I,t}(S). \tag{3}$$

Our target is to find a set of regions S from R that minimizes Equation (3), while having enough representation ability. We want regions in S to come from each of the positive images and at the same time to keep S as small as possible. This is indirectly controlled by the threshold parameter t. If t is small, then regions in S will come from regions from many different images and the set S is relatively small. On the other hand, if t is large, there will be many more regions in the set, which covers each image more.

To minimize F(S), we want regions in S with the most number of edges from positive images. An edge e exists between R and I if the region r_i from R is the nearest neighbors of the region in I. That means the function F(S) is a submodular function and can be optimized via a greedy algorithm. This is because function $2^{\Omega} \rightarrow R$ defined in Equation (3) is a submodular function.

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

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

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

$$\begin{aligned} \left|N(T \cup \{r\}\right| - |N(T)| &= |N(r) \backslash N(T)| \\ \left|N(T \cup \{r\}\right| - |N(T)| &\leq |N(r) \backslash N(S)| \\ \left|N(T \cup \{r\}\right| - |N(T)| &= |N(S) \cup \{r\}| - |N(S)| \end{aligned} \tag{5}$$

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

Optimization: Our goal is to select a representative minimum subset $S \subseteq R$, which can be stated as in Equation (6):

$$\min |S| \text{ (for } S \subseteq R), F(s) \leq \propto F(R), \text{ for } \propto \in (0, 1) \tag{6}$$

For this, we have used the algorithm discussed in Awolsey [1982]. Let $S_0 = 0$ and in each following iteration step I, add the region r that can cover the largest number of uncovered regions. More formally, add the region that could maximize the marginal gain $F(S_I \cup \{r\} - F(S_I)$.

Our region representation method is faster than existing ones, as explained below:

In our region representation method, we have chosen a greedy algorithm with optimization. As we know, in a greedy algorithm in each step, we should choose a set that contains the greatest number of uncovered elements. As an example, if we have a set A = {1,2,3,4,5,6,7,8,9,10} and subset of S = {S_1 = {1,2,3,4,5,6}, S_2 = {1,2,3,8,9}, S_3 = {4,5,6,9,10}, S_4 = {7,8,9}, S_5 = {10}}, in the normal Greedy algorithm, in the first step, S_1 has six uncovered elements and is better than all other five sets. So, first, we select S_1, and cover C will be {{1,2,3,4,5,6}}. In the second step, we should check other

uncovered sets. S_4 has three uncovered elements, and S_2 has two uncovered elements. So, the second set will be S_4. in the third step, S_5 has one uncovered element and S_2 has no uncovered elements, so we select S_5 as the selected set and the C = {{1,2,3,4,5,6}, {7,8,9},{10}}. Now C has all elements with the size = 3.

In our modified method and using the same example, we choose a step-size equal to two. So, the algorithm in each step selects two sets. This means that in the first step candidates are {(S_1, S_2)), (S_1, S_3), (S_1, S_4), (S_1, S_5), (S_2, S_3), (S_2, S_4), (S_3, S_4), (S_3, S_5), (S_4, S_5)}. Among all candidates, (S_2, S_3) are better, since they have ten uncovered items. So, in the first step, C = {{1,2,3,8,9}, {4,5,6,9,10}}. So far, all elements are covered by C, so the size C will be equal to two. So, the C cover is smaller than the one calculated by the usual greedy algorithm. As such, in our modified method the region representation step is faster, since it computes a smaller cover than the cover computed by the greedy algorithm.

3.6. Deep Learning

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

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

$$\mathrm{p(labe \mid an\ input\ image)}. \tag{7}$$

The standard way to model a neuron's output f as an activation function of its input x is either a hyperbolic or a sigmoid function, shown, respectively, in Equations (8) and (9):

$$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}), \tag{8}$$

$$\mathrm{sigmoid}(x) = 1/(1 + e^{-x}). \tag{9}$$

But we use the term rectified linear unit (ReLU) to refer to a unit in a neural net that uses the activation function$\max(0; x)$. As compared to the above-mentioned activation functions (hyperbolic or sigmoid), deep convolutional neural networks with ReLUs train several times faster [Krizhevsky et al. 2012].

To better understand the deep neural network operation in our system, consider this example. Let's say we choose greyscale images that are 28 by 28 pixels in size as input. If we use the notation x to denote the training input, then there will be 28∗28 = 784 input neurons. Each entry in the neuron represents the grey value for a single pixel in the image. We'll denote the corresponding desired output by $(y = y(x))$. What we'd like is an algorithm that lets us find weights and biases so the output from the network approximates $y(x)$ for all training inputs x. So, if we have a training image set of food images and we want to classify the food type as Apple during the learning phase, we could possibly achieve that by tweaking the weight and bias values. To quantify how well we're achieving this goal, we define a cost function [Zeiler et al. 2013]:

$$C(w, b) \equiv 1/2 \sum_x \| y(x) - a \|^2 . \tag{10}$$

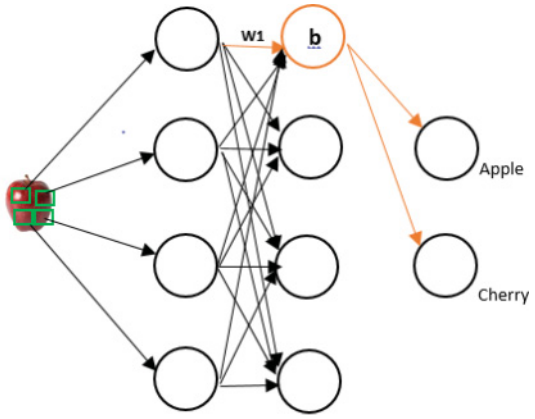


Fig. 8. An example showing implementation of Stochastic Gradient Descent.

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

We'll do that using the well-known algorithm stochastic gradient descent. By using a smooth cost function like the quadratic cost, it turns out to be easy to figure out how to make small changes in the weights and biases to get an improvement in the cost [LeCun et al. 1998]. Hence, we will be able to tweak the weights and bias to get the output closer to the desired output, during the learning phase. Hence, our goal is to train the neural network to find weights and biases, which minimize the quadratic cost function C (w,b).

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

The stochastic gradient descent works by randomly picking out a small number m of randomly chosen training inputs (for example, 10 images from the original set of 100 images). We'll label those random training inputs $X_1, X_2, \ldots, X_m$. Then, stochastic gradient descent works by picking out a randomly chosen mini-batch of training inputs, and training with those, where the weights and bias is computed by:

$$\begin{aligned} w_k &\rightarrow w'_k = w_k - \eta/m \sum_j (\partial C/\partial w_k) \\ b_{bl} &\rightarrow b'_l = b_l - \eta/m \sum_j (\partial C/\partial b_{bl}), \end{aligned} \tag{11}$$

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

The algorithm in [http://neuralnetworksanddeeplearning.com] is a way of implementing stochastic gradient descent, which is shown in Figure 8 and the backward propagation to compute the weight and bias for smaller cost function. Training the deep neural network in this way, will allow us to make the necessary changes to the

weight and bias, without majorly significantly effecting the output of the network and giving us the desired results. For example, this algorithm will help us to tweak the weights(w) and bias(b) during the learning phase, in a way we can finally determine the output as one of the two (apple or cherry), without effecting the rest of the food classes. Delta changes in either the weights or the bias will change the result from one food class to the other. As shown in the diagram below, considering we have taken the color feature into account, any changes in the weight of w_1 or bias b would make small changes to the final results, which in this case between apple and cherry (having almost the same color features) will alter the eventual result. If the probability of the image is $p > 0.5$ toward Apple, then it would be classified as Apple and the same is the case with any food type. Algorithm 2 describes the method step by step.

ALGORITHM 2: Implementing stochastic gradient

1: *Input = training examples.*
2: *For each training example x.*
3: *// Feedforward.*
4: *For each* $l = 2, 3, \ldots, L$ *compute* $z^l = w^l a^{l-1} + b^l$ *and* $a^l = \sigma(z^l)$.
5: *//Output error.*
6: *Compute* $\delta^l = \nabla_a^C \odot \sigma'(z^l)$.
7: *// Backpropagate the error.*
8: *For each* $l = L-1, L-2, \ldots, 2$ *compute* $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
9: *For each l = L, L-1 . . . 2, update the weight.*
10: *// L = the layer number,* δ^l *= the output error, bl = the bias,* w^l *= the weight and C = the cost.*

3.6.2. Implementing the Deep Neural Network. In this section, we provide the details of the deep neural network used in our system. The first step in our approach is to generate a pre-trained model file with the help of the CNN network. This is performed by initially capturing a set of images of one particular class (e.g., 50 images of class apple) and then labeling them with an object name-set (object being apple). So, it does not categorize them as part of the apple class. Once the model file is generated from the training, we load it into the application and test it against the images captured and submitted by the user. The system then performs the image recognition process and generates a list of probabilities against the label name. The label with the highest probability is prompted to the user in the dialog circle, to confirm the object name. Once the object name is confirmed, the system performs the calorie computation part by calculating the size of the food item with respect to the user's finger, which is placed next to the food object in the plate as shown in Figure 12. Since the dimension of the user's finger is known, the system can compute the corresponding dimension of the food object. These dimensions are then used in calculating the volume of the food object, which is further mapped to the corresponding calorie value of the food object [Pouladzadeh et al. 2016]. The correctly detected images will then be considered as the set of relevant (positive) images, which are used to train the system. In the second step of the training, we re-train the system with the set of negative images (images that do not contain the relevant object). In our case, we trained the system with the background images. Figure 9 illustrates the above-mentioned process.

For having better accuracy in food recognition, we created a new model for our multiple food recognition. For food image feature extraction and classification, deep belief networks (DBN), which are based on the concept of deep learning, allow us to achieve higher accuracy and better generalization, even with small datasets. This is a well-known property of deep learning. It further gives us the flexibility to integrate its feature extraction model with other linear classifiers, including SVM. Hence, for the

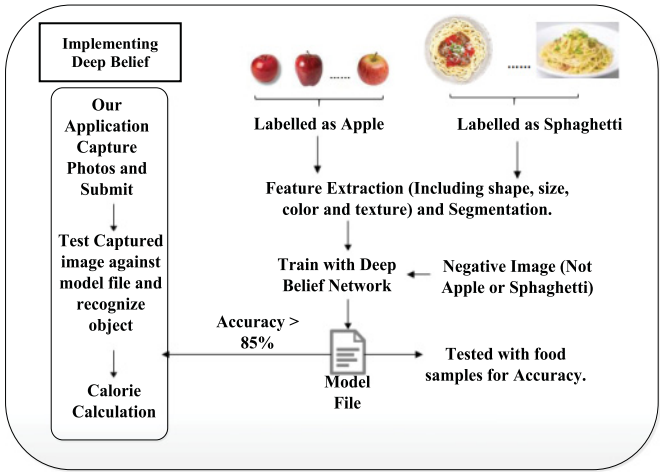


Fig. 9. Implementation of Deep Belief Network.

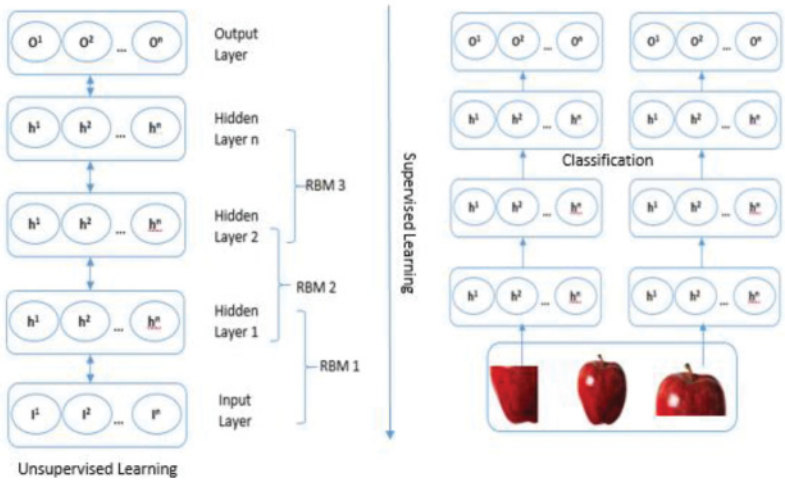


Fig. 10. Deep Belief Network Model for Food Recognition.

food recognition part, we use DBN, wherein we initially perform unsupervised training without labeling 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 [3]. Training the deep neural network in this way will allow us to make the necessary changes to the weight and bias, from which we obtain the desired results. We label these random training inputs X1, X2, . . . , Xm. The stochastic gradient descent algorithm then selects a randomly chosen mini-batch of training inputs and trains with those. As shown in Figure 10, DBN is a hierarchical model with many hidden layers in the network. Two hidden layers combine to form the Restricted Boltzmann Machine (RBM), which further form stack of RBMs. Each layer, including the hidden layers, contain a set of neurons i_k being the input vectors, h_k being the hidden neuron, and Ok being the output neuron of the kth layer in the figure.

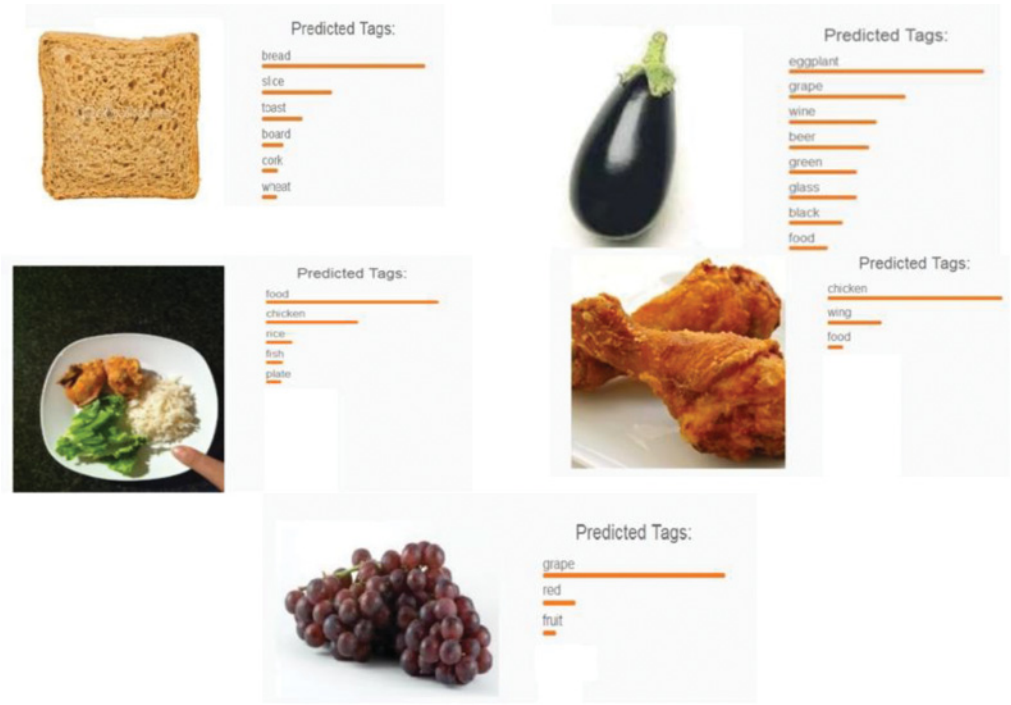


Fig. 11. Result of food recognition.

As part of the feature extraction process, the unsupervised and supervised learning (backward propagation) together generate the desired feature vectors that are used by the classifiers (feed forward propagation). Hence, we derive the high-level features from the low-level features further. The prediction is based on the probabilistic model, where P(labe | n input image) is the probability of the match with the corresponding label set.

3.7. Food Name Prediction and Calorie Extraction

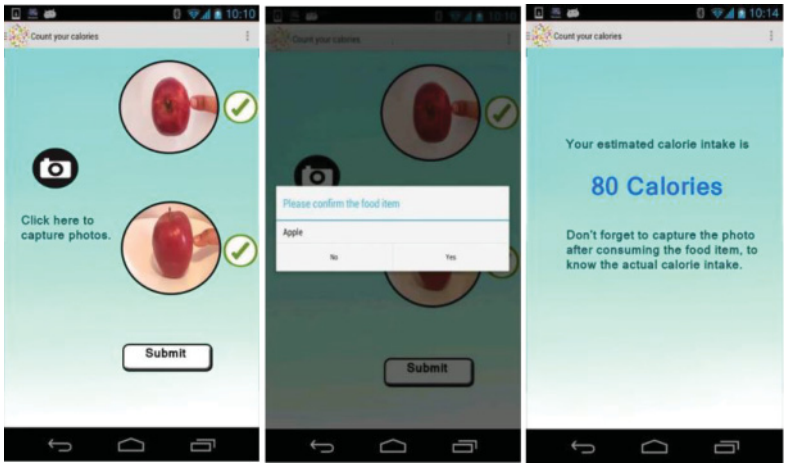
The recognition step will generate a list of food items ordered in occurrence probability. This means that for each food item, we may have three or four predicted tags. A sample output of the recognition step is shown in Figure 11.

The proposed system will choose the food item with the highest probability. Providing several food items comes from the general concept behind deep learning. In other words, any deep learning algorithm provides the list of items and their corresponding probability. This concept applies to our method as well; however, in many cases the probability of one or two names are so high that the user only needs to select from a very short list of food items.

After recognizing the food names, we can estimate the calories in the food plate as described in Pouladzadeh et al. [2014].

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

When the user clicks to start our system, it will redirect him/her to the camera. The user then captures the images of the food from two angles: the top view, which enables the application to extract the food portions, and the side view, which enables the application to analyze the depth of the food item, which is needed to calculate the mass and



a b c

Fig. 12. (a) Photos uploaded. (b) Confirm food type. (c) Estimated calorie value.

subsequently the calories [Pouladzadeh et al. 2014], as shown in Figure 12(a). Then the user draws the bounding circle, and when the user hits the submit button, the application will prompt to confirm the food type, as shown in Figure 12(b). If the food type suggested by the application is correct, the user then clicks on "Yes"; if not, the user clicks on "No." If the user clicks "Yes," the application will then display the estimated calorie value of the food type, as shown in Figure 12(c). If the user clicks on "No," the application will prompt the user to start again.

This section explains how we trained and evaluated the proposed system on a manually collected food dataset. A detailed description of the results is also given.

4.1. Dataset

The food dataset used for our experiments is FooDD [Pouladzadeh et al. 2015], composed of a training set and a test set. For training, we collected 7,000 images of 30 categories of food. Each one of the training images only contains one food item, which is shown in Figure 13.

4.2. Recognition Results

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

4.2.1. Average Accuracy, Recall, and Precision. Recall rate measures the ability to detect food items and precision rate measures how accurate the detection is. Equation computes the recall rate Equation (12), and the precision rate is computed by Equation (13) as follows:

$$\text{Recall} = \frac{\text{num. of correctly detected items in the category}}{\text{ground truth num. of items in the category}}, \tag{12}$$

$$\text{Precision} = \frac{\text{num. of correctly detected items in the category}}{\text{num. of predicted items in the category}} \tag{13}$$

Fig. 13. Examples of training images (left) and testing images (right). The training dataset contains images with only a single food item and the testing dataset contains images with multiple food items.

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

Our experiments show that the proposed method described in Section 3 works well in recognizing and predicting the food items in the image. Also, we have achieved a higher accuracy rate (94.11%) compared to the 50.8–88% accuracy rate of related systems that currently exist. Region mining is an important step in object detection; it discovers discriminative regions that help distinguish each type of food from the others. Regions that belong to the background or would possible confuse the classifier were discarded.

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

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

Table I. Food Recognition Accuracy

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

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

4.2.2. Comparison Experiment. To evaluate the significance of the region mining procedure in object detection, we conduct a comparison experiment, which uses the whole image region to train multiple food detectors without any region mining involved during training. An average recall rate of 90.98% and precision rate of 93.05% is obtained by the comparison method. Compared to the results obtained with region mining, it has a lower precision rate. The classifiers trained without region mining produce low precision rate, because many false predictions are computed.

4.2.3. Difficulties. The biggest problem that the application may encounter is having too much variance in the food appearance. For example, the color, size, and shape of

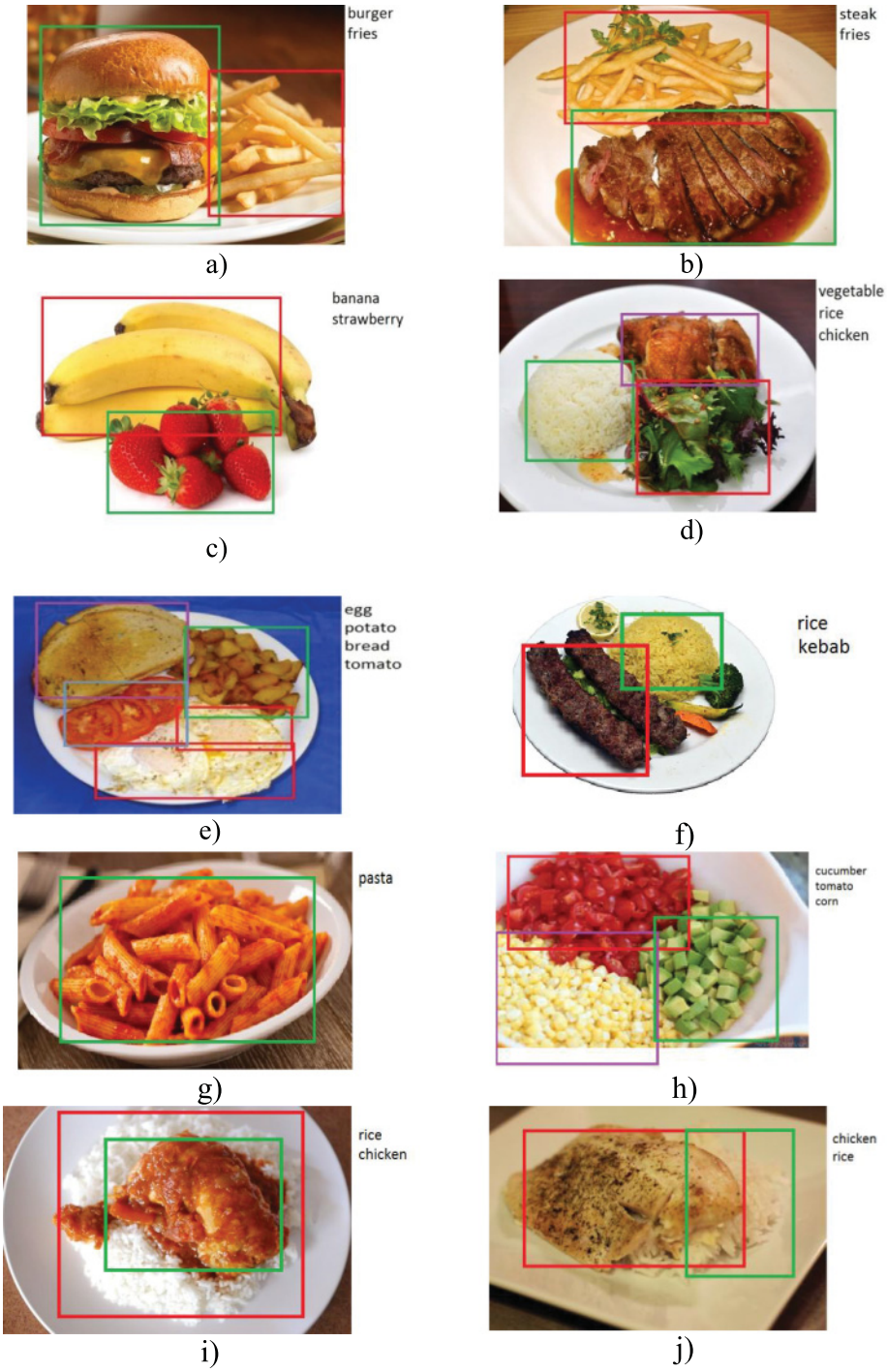


Fig. 14. Detection Results: (a)–(i) correct recognition, (j) wrong recognition.

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

Figure 14 shows some successful recognition by our proposed method. Although food items vary a lot in appearance, size, and occlusion, our proposed method correctly recognizes and localizes most of the samples. However, our proposed algorithm also generates inaccurate detection on some of the test images.

5. CONCLUSIONS

In this article, we proposed a Deep Learning method for detecting multiple food items from food pictures taken with mobile devices. Our experiments were conducted on a dataset of 30 food categories, where images containing only a single food item were used as the training set, and images containing multiple and mixed food items were used as the test set. Our proposed method learns from single food item images using various techniques, including CNN features, region mining, and classifier training. We showed that, in our system, the combination of those methods provides a powerful instrument for attaining a high level of accuracy of food recognition. Using our system, an average recall rate of 90.98%, precision rate of 93.05%, and accuracy of 94.11% compared to 50.8% to 88% accuracy of other existing food recognition systems can be achieved.

REFERENCES

www.obesitynetwork.ca.

http://www.who.int.

Parisa Pouladzadeh, Shervin Shirmohammadi, and Abdulsalam Yassine. 2016. You are what you eat: So, measure what you eat. *IEEE Instrum. Meas. Mag.* 19, 1, 9–15.

Parisa Pouladzadeh, Pallavi Kuhad, Sri Vijay Bharat Peddi, Abdulsalam Yassine, and Shervin Shirmohammadi. 2016. Calorie measurement and food classification using deep learning neural network In *Proceedings of the IEEE International Conference on Instrumentation and Measurement Technology (I2MTC'16)*.

P. Pouladzadeh, A. Yassine, and S. Shirmohammadi. 2015 FooDD: Food detection dataset for calorie measurement using food images, in new trends in image analysis and processing. In *Proceedings of the ICIAP 2015 Workshops*, V. Murino, E. Puppo, D. Sona, M. Cristani, and C. Sansone (eds.). Lecture Notes in Computer Science, Springer, Vol. 9281, 441–448. DOI:10.1007/978-3-319-23222-5_54

Sri Vijay Bharat Peddi, Abdulsalam Yassine, and Shervin Shirmohammadi. 2015. Cloud based virtualization for a calorie measurement e-health mobile application. In *Proceedings of the 2015 International Conference on Multimedia and Expo Workshops (ICME'15)*. 1–6

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *Proceeding of the European Conference on Computer Vision–(ECCV'14)*. 446–461.

Yuji Matsuda, Hajime Hoashi, and Keiji Yanai. 2012. Recognition of multiple-food images by detecting candidate regions. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'12)*. 25–30.

Yoshihiro Kawano and Katsuki Yanai. 2013. Real-time mobile food recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'13)*. 1–7.

Weiyu Zhang, Qian Yu, Behjat Siddiquie, Ajay Divakaran, and Harpreet Sawhney. 2015. Food recognition and nutrition estimation on a smartphone. *J. Diabetes Sci. Technol.* 9, 3, 525–533.

Satoru Fujishige. 2005. *Submodular Functions and Optimization*, Vol. 58. Elsevier.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Cafe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*. 675–678.

Laurence Awolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* 2, 4, 385–393.

Fengqing Zhu, Marc Bosch, Nitin Khanna, and Carol J. Boushey. 2015. Multiple hypotheses image segmentation and classification with application to dietary assessment. *IEEE J. Biomed. Health Informat.* 19, 1, 377–389.

Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. 2014. Food detection and recognition using convolutional neural network. In *Proceedings of the ACM International Conference on Multimedia*, 1085–1088.

Kiyoharu Aizawa and Makoto Ogawa. 2015. FoodLog: Multimedia tool for healthcare applications. *IEEE MultiMed.* 22, 2, 4–9.

Sosuke Amano, Kiyoharu Aizawa, and Makoto Ogawa. 2015. Food category representatives: Extracting categories from meal names in food recordings and recipe data. In *Proceedings of the IEEE International Conference on Multimedia Big Data.* 48–55. DOI:10.1109/BigMM.2015.54

Colin Ware. 2008. Toward a perceptual theory of flow visualization. *IEEE Comput. Graph. Appl.* 28, 2 (2008), 6–11. DOI:http://dx.doi.org/10.1109/MCG.2008.39

Xi-Jin Zhang, Yi-Fan Lu, and Song-Hai Zhang. 2016. Multi-task learning for food identification and analysis with deep convolutional neural networks. *J. Comput. Sci. Technol*. 31, 3, 489–500.

Morteza Akbari Fard, Hamed Haddadi, and Alireza Tavakoli Targhi. 2016. Fruits and vegetables calorie counter using, convolutional neural networks. In *ACM Dig. Health*, 121–122.

Ashutosh Singla, Lin Yuan, and Touradj Ebrahimi. 2016. Food/non-food image classification and food categorization using pre-trained googlenet model. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management.* 3–11.

Wataru Shimoda Keiji Yanai. 2016. Foodness proposal for multiple food detection by training of single food images. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*. 13–21.

Joachim Dehais, Marios Anthimopoulos, and Stavroula Mougiakakou. 2016. Food image segmentation for dietary assessment. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management.* 23–28.

A. Krizhevsky, I. Sutskever, and G. Hinton, 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of Neural Information Processing Systems (NIPS'12)*.

M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean and G. E. Hinton. 2013. Onrectied linear units for speech processing. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP'13)*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradientbased learning applied to document recognition. *Proc. IEEE*, 86, 11: 2278–2324.

Parisa Pouladzadeh, Shervin Shirmohammadi, and Rana Almaghrabi. 2014. Measuring calorie and nutrition from food image. *IEEE Trans. Instrument. Measure.* 63, 8, 1947–1956.

Z. Su, Q. XU, M. Fei, and M. Dong. 2016. Game theoretic resource allocation in media cloud with mobile social users. *IEEE Trans. Multimed. (TMM)* 18, 8, 1650–1660.

Received October 2016; revised February 2017; accepted March 2017