Im2Calories: towards an automated mobile vision food diary

Austin Myers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban
Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, Kevin Murphy
amyers@umd.edu, (nickj, rathodv, kbanoop, gorban)@google.com
(nsilberman, sguada, gpapan, jonathanhuang, kpmurphy)@google.com

We present a system which can recognize the contents of your meal from a single image, and then predict its nutritional contents, such as calories. The simplest version assumes that the user is eating at a restaurant for which we know the menu. In this case, we can collect images offline to train a multi-label classifier. At run time, we apply the classifier (running on your phone) to predict which foods are present in your meal, and we lookup the corresponding nutritional facts. We apply this method to a new dataset of images from 23 different restaurants, using a CNN-based classifier, significantly outperforming previous work. The more challenging setting works outside of restaurants. In this case, we need to estimate the size of the foods, as well as their labels. This requires solving segmentation and depth / volume estimation from a single image. We present CNN-based approaches to these problems, with promising preliminary results.

1. Introduction

Many people are interested in tracking what they eat to help them achieve weight loss goals or manage their diabetes or food allergies. However, most current mobile apps (MyFitnessPal, LoseIt, etc) require manual data entry, which is tedious and time consuming. Consequently, most users do not use such apps for very long [9]. Furthermore, amateur self-reports of calorie intake typically have an error rate that exceeds 400 calories per day [31, 5].

Rather than rely purely on data entry, several approaches make use of a mobile camera to aid in this task. Cordeiro *et al*. [8] records a photo of the meal but does not perform any visual analysis of the image. Several previous approaches [23] [29] rely on an expert nutritionists to analyse the image offline (at the end of each day). Other approaches [26] [25] use crowd sourcing to interpret the image, in lieu of an expert. However, crowd sourcing is both costly and slow, which hinders widespread adoption.

Several existing works [39, 21, 37] do use computer vision algorithms to reason about meals but only work in laboratory conditions where the food items are well separated and the number of categories is small. Furthermore, most of these methods use traditional, hand-crafted visual features, and only use machine learning at the classification stage.

The holy grail is an automatic method for estimating the nutritional contents of a meal from one or more images. Unfortunately, even a perfect visual interpretation of the scene cannot perfectly predict what is inside many foods, e.g., a burrito. Consequently, an ideal system is one which correctly infers what is knowable and prompts the user for feedback on inherantly ambiguous components of a meal. Our goal is to minimize user effort for completing food diaries by offering smart "auto-complete" functionality, rather than complete automation.

In this paper, we take some initial steps towards such a system. Our approach utilizes several deep learning algorithms, tailored to run on a conventional mobile phone, trained to recognize food items and predict the nutrional contents meals from images taken "in the wild". We refer to this task as the "Im2Calories" problem, by analogy to the recent line of work on the "Im2Text" problem. It should be stressed, however, that we are interested in estimating various other properties of a meal (such as fat and carbohydrates) and not just calories.

We start by building on [1], who developed a system that can predict the calorie content of a meal from a single image. Their key insight (also made in [2]) was that the problem becomes much simpler if we restrict the setting to one where the user is eating in a restaurant whose menu is known. In this case, the problem reduces to one of detecting which items, out of the K possible items on the menu, the user has chosen. Each item typically has a standard serving size[1], and we typically know its nutritional contents.[2], whereas getting nutritional information for arbitrary cooked foods is much harder, as discussed in Section 7.

In Section 3, we show that by simply replacing the hand-crafted features used in [1] with a convolutional neural network (CNN), we can significantly improve performance,

[1] There may be variants, but these are typically few in number (e.g., small, medium or large fries). Hence we an treat these as different items.

[2] The US Food and Drug Administration has passed a law requiring all major chain restaurants to post the nutritional contents of their meals, starting in December 2016. See [15] for details.

both at labeling the foods and estimating total calories. We then extend their method from 3 restaurants to 23 restaurants, increasing the coverage from 41 food items to 2517. We show that the same basic multilabel classification system continues to work.

Unfortunately, it is hard to get enough images for all the menu items for all the restaurants in the world. And even if we could, this would not help us when the user is not eating at a restaurant. Therefore, in Section 4, we develop a set of 201 generic, restaurant-independent food tags. We then extend the existing public Food101 dataset [3] with these tags using crowdsourcing. We call the resulting dataset Food201-multilabel and plan to release it publicly.[3] We show that the same kind of CNN-based multi-label classifier also works fairly well on this new (larger and more challenging) dataset, although we found it necessary to perform some clustering of visually indistinguishable labels in order to get acceptable performance.

Of course, detecting the presence of a food item in an image is not sufficient, since most items can be "parameterized" in terms of size, toppings, etc. Note that this is true even in the restaurant setting. We could of course ask the user about these variants, but we would like to do as much as possible automatically.

To be able to perform such fine grained classification, we need to be able to localize the objects within the image and extract detailed features. We argue that a segmentation-based approach is more appropriate than the traditional bounding-box approach, since food items can have highly variable shape. In Section 5, we present the new Food201-segmented dataset, and our approach to semantic image segmentation. We show that leveraging the multilabel classifier from the earlier stage can help with segmentation, since it provides a form of "context".

Once we have segmented the foods, we can try to estimate their volume. To do this, we need to know the surface height of the foods above the plate. In Section 6, we present some preliminary results on estimating the depth of each pixel from a single RGB image, using a CNN. We then show promising results on estimating the volumes of foods.

In summary, this paper makes 3 main contributions. First, we develop a system that can recognize the contents of a restaurant meal much more accurately than the previous state of the art, and at a much larger scale. Second, we introduce a new dataset, Food201, and show how it can be used to train and test image tagging and segmentation systems. Third, we show some promising preliminary results on the challenging problem of mapping image to calories from images taken in the wild, in a non-restaurant setting. Our overall system is illustrated in Figure 1.

[3] See https://storage.googleapis.com/food201/food201.zip.

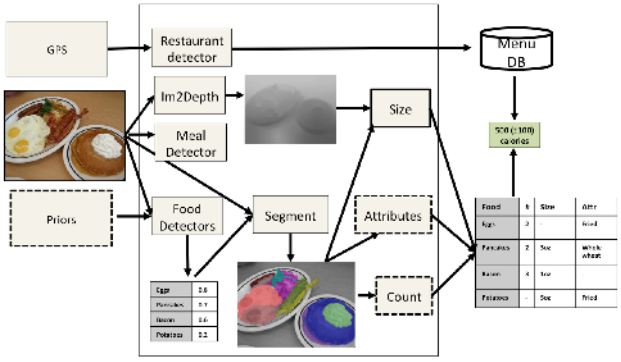

Figure 1. Illustration of the overall system. Dotted boxes denote components that have not yet been implemented. The input is one or more images, and optionally a GPS signal and user priors (e.g., concerning food preferences). The output is a food diary, and an estimated total calorie count (in cases where a suitable nutritional database is available, such as from a restaurant's menu).

Name	#Classes	#Train	#Test	Comments
Food101	101	75,750	25,250	[3]
Food101-Background	2	151,500	50,500	Food vs non-food
Food201-MultiLabel	201	35,242	15,132	Multi label
Food201-Segmented	201	10,100	2,525	Label per pixel
Restaurant	2517	75k	25k	Single label
Gfood-3d	-	150k	2471	Depth per pixel
Nfood-3d	-	-	1050	Depth per pixel

Table 1. Summary of the datasets used in this paper. The Restaurant dataset contains web images for 23 restaurants. Gfood-3d is from 50 Google meals. Nfood-3d is from 11 meals made with Nasco food replicas.

2. Meal detection

The first step in our pipeline is to determine if the image is an image of a meal at a "reasonable" distance from the camera. We can phrase this as a simple binary classification problem. To tackle this problem, we need to have a suitable dataset. We start by taking the Food101 dataset developed in [3], which contains 101 classes of food, 1000 images each (750 train, 250 test). Food101 is the largest publicly available dataset of food images we are aware of (see Table 1 for a comparison with some other public datasets).

The Food101 dataset is designed for multi-class classification. To make a dataset suitable for binary classification, we combined all the food classes into one generic "food" class, and then randomly extracted an equal number of images from the ImageNet challenge dataset [30] to create the "non-food" class.[4]

Each image is rescaled (if necessary) so that its maximum height or width is 512 pixels, but preserving the original aspect ratio. We call this new dataset the **Food101-**

[4] ImageNet actually contains 51 food classes, so we removed these images from the negative set.

Background dataset.

To train a classifier for this problem, we took the GoogLeNet CNN model from [34], which had been pre-trained on ImageNet, removed the final 1000-way softmax, and replaced it with a single logistic node. Then we fine tune the whole model on the Food101-Background dataset; this takes about 12 hours using a single Titan X GPU with 12 GB of memory. The final test set accuracy is 99.02%.

3. Restaurant-specific im2calories

Once we have determined that the image contains a meal, we try to analyze its contents. The first step is to determine which restaurant the user is in. In this paper, we use Google's Places API [27] for this. We then retrieve the menu of the nearest restaurant from the web,[5] parse the menu into a list of K food items, and retrieve images for each of these, either by performing web search (as in [2]) or by asking users to collect images (as in [1]).[6]

Once we have the dataset, we can train a classifier to map from image to label. Since the user may have multiple food items on their plate, it is better to use a multi-label classifier rather than using a multi-class classifier, which assumes the labels are mutually exclusive. Next we can estimate the set of foods that are present by picking a suitable threshold ϕ, and then computing $\mathcal{S} = \{k : p(y_k = 1|x) > \phi\}$, where $p(y_k = 1|x)$ is the probability that food k is present in image x. Finally, we can lookup each of these food items in our (restaurant specific) database, and then estimate the total calories as follows: $\hat{C} = \sum_{k \in \mathcal{S}} C_k$, where C_k is the calorie content of menu item k. Alternatively, to avoid having to specify the threshold ϕ, we can compute $\bar{C} = \sum_{k=1}^{K} p(y_k = 1|x) C_k$. We compare these methods below.

3.1. Experiments on MenuMatch dataset

To evaluate this approach, we used the dataset from [1], known as "MenuMatch". This consists of 646 images, tagged with 41 food items, taken from 3 restaurants. Unlike other datasets, each image in MenuMatch has a corresponding ground truth calorie estimate, computed by an expert nutritionist. In addition, each restaurant's menu has corresponding ground truth calorie content per item.

[1] used various hand-crafted features together with a linear SVM, trained in a one-vs-all fashion. Their best performing system achieved a mean average precision (mAP) of 51.2% on the test set. By contrast, we get much better results using a deep learning approach, as we explain below.

Method	Mean error	Mean absolute error
Baseline	-37.3 ± 3.9	239.9 ± 1.4
Meal Snap	-268.5 ± 13.3	330.9 ± 11.0
Menu Match	-21.0 ± 11.6	232.0 ± 7.2
$\hat{C}$	-31.90 ± 28.10	163.43 ± 16.32
$\bar{C}$	-25.35 ± 26.37	152.95 ± 15.61

Table 2. Errors in calorie estimation on the MenuMatch dataset. $\hat{C}$ and $\bar{C}$ are our methods. Numbers after the $\pm$ sign are standard errors estimated by 5-fold cross-validation. See text for details.

We took the GoogLeNet CNN model from [34], which had been pre-trained on ImageNet, removed the final 1000-way softmax, replaced it with a 101-way softmax, and fine-tuned the model on the Food101 dataset [3]. The resulting model has a classification accuracy on the Food101 test set of 79%, which is significantly better than the 50.76% reported by [3] (they used hand crafted features plus an SVM classifer using a spatial pyramid matching kernel).

Next, we took the model which was trained on Food101, removed the 101-way softmax, replaced it with 41 logistic nodes, and fine-tuned on the MenuMatch training set. (Pre-training on Food101 was necessary since the MenuMatch dataset is so small.) The resulting mAP on the test set is 81.4%, which is significantly higher than the best result of 51.2% reported in [1].

Finally, we wanted to assess the accuracy of calorie prediction. We compared 5 methods: the MenuMatch system of [1], the MealSnap app [25] that uses crowdsourcing, our method using $\hat{C}$, our method using $\bar{C}$, and finally, a baseline method, which simply computes the empirical mean of the calorie content of all meals from a specific restaurant. The results are shown in Table 2. We see that our system has considerably lower error than MenuMatch and the crowdsourced MealSnap app. (The unreliability of MealSnap was also noted in [26].) In fact, we see that MenuMatch barely beats the baseline approach of predicting the prior mean.

3.2. Scaling up to more restaurants

The MenuMatch results are based on 646 images of 41 food items from 3 restaurants. In this Section, we discuss our attempts to scale up these experiments.

First, we downloaded the menus for the top 25 restaurants in the USA, as ranked by sales.[7] We decided to drop "Pizza Hut" and "Chipotle", since gathering images for their menu items was tricky.[8] From the remaining 23

[5] This data is available for many US chain restaurants in semi-structured form from https://www.nutritionix.com.

[6] In practice, it is suprisingly complicated to parse the menus and retrieve relevant images. For example, the restaurant "16 handles" contains the folowing items: NSA Blueberry Tease, NSA Chocolate Eruption, NSA Strawberry Fields, and 65 other similar entries. You need to know that these are from the frozen yogurt section of the menu, and that NSA stands for "no sugar added", in order to make any sense of this data.

[7] Source: http://nrn.com/us-top-100/top-100-chains-us-sales.

[8] For example, "Pizza Hut" has menu items such as "chicken", "pepperoni", etc. But these are toppings for a pizza, not individual food items. Similarly, "Chipotle" lists "chicken", "beans", etc. But these are fillings for a burrito, not individual food items.

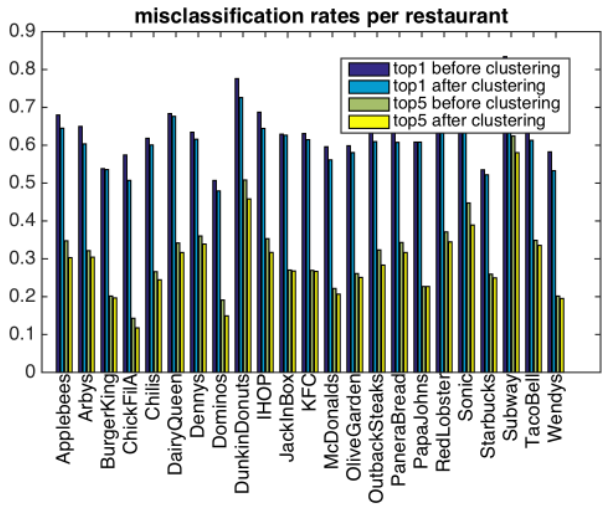

Figure 2. Top 1 and top 5 error rates on the test set for 23 different restaurants, before and after clustering the most confusable labels.

restaurants, we collected 4857 menu items. We manually removed drinks and other miscellaneous items, and then scraped 1.2 million images by issuing image search queries of the form "<restaurant name> <item name> (yelp | flickr | instagram | pinterest | foodspotting)". (These site restricts were chosen to increase the chances that we collected user-generated photos, rather than "official" photos from the restaurant itself, which tend to be too easy.) Of the scraped images, 270k were sent to Amazon Mechanical Turk for verification, producing a final set of 2517 menu items and 99k images. We call this the **Restaurant** dataset.

We then took our GoogleLeNet model, which had been trained on ImagetNet and then Food101, and replaced the 101-softmax with 2517 logistic nodes. We trained the final layer of this model on 75% of the data, and tested on the rest. We then computed the top 1 and top 5 error rates, averaged over the food items, for each of the 23 restaurants. The results are shown in Figure 2.

The top 1 error rate is quite high. This is because many food items are extremely similar, and it is hard, even for a human expert, to tell them apart. For example, McDonalds has the following items on its menu: Quarter Pounder Deluxe, Quarter Pounder Bacon Cheese, Quarter Pounder with Cheese, etc. (See Figure 3 for an illustration of these food items.)

To combat this issue, we computed the class confusion matrix on the training set for each restaurant, and then performed a very simple clustering of similar items. In particular, we computed the K nearest neighbor graph, in which we connected each label to the K other labels it is most often mapped to (which can include itself). We then computed the connected components to form a set of clusters.

Figure 3. The first two images are put into the same visual cluster, the third is kept distinct. Image sources: http://www.mcdonaldsmenu.mobi/beefburgersmenu/deluxequarterpounder/. http://www.mcdonaldsmenu.mobi/beefburgersmenu/baconcheesequarterpounder/. http://www.mcdonalds.com/us/en/food/product_nutrition.burgerssandwiches.7.quarter-pounder-with-cheese.html

We found qualitatively that using $K = 1$ gave a good tradeoff between merging the most confusable classes and overclustering. With $K = 1$, the number of clusters was about 0.9 times the original number of items; most clusters were singletons, but some had 2 or 3 items in them. Figure 3 gives an example of two clusters we created from the McDonald's menu; the first cluster contains two very visually similar items (Quarter Pounder Deluxe and Quarter Pounder Bacon Cheese), and the second cluster contains a visually distinctive item (Quarter Pounder with Cheese).

Finally, we evaluated performance of the classifier on the clustered test labels, by defining the probability of a cluster as the max probability of any label in the cluster. Figure 2 shows that, not surprisingly, the error rates decrease. In the future, we hope to try using a hierarchical classifier, which can tradeoff specificity with error rate c.f., [11].

4. Generic food detection

The results from Section 3 required images for each menu item from each restaurant. It is difficult to acquire such data, as previously remarked. To create a more generic dataset, we took half of the Food101 dataset (50k images), and asked raters on Mechanical Turk to name all the food items they could see in each image. We included the original class label as one of the choices, and manually created a list of commonly co-occuring foods as additional choices in the drop-down menu (e.g., eggs often co-occur with bacon). We also allowed raters to enter new terms in a text box. We used 2 raters per image. After manually merging synonymous terms, and removing terms that occurred less than 100 times, we ended up with a vocabulary of 201 labels. On average, each image had 1.9 labels (with a median of 1 and a max of 8).

We split the resulting dataset into 35,242 training images and 15,132 test images, in a way which is consistent with the Food101 train/ test split. We call this the **Food201-MultiLabel** dataset. See Table 1 for a summary of how this dataset compares to other public food datasets.

Next, we developed a multi-label classifier for this

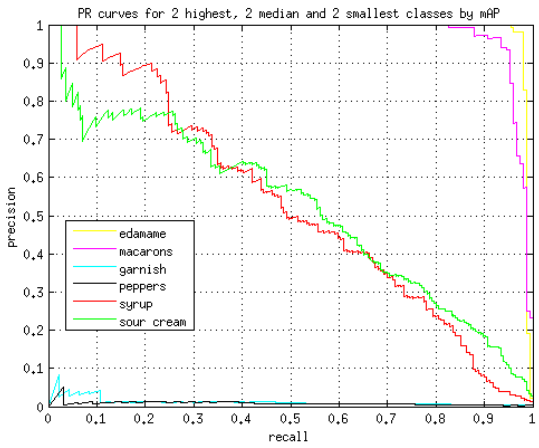

Figure 4. Precision-recall curves for 6 classes, ranging from best to worst, on the Food201-MultLabel dataset.

dataset, using the same method as in Section 3 (namely taking the GoogLeNet model, replacing the 101-way softmax with 201 logistic nodes). This takes about half a day to train on a single GPU. We then compute the average precision (area under the precision-recall curve) for each class, and average this over the classes, to compute the mean average precision (mAP).

The mAP is 0.8 for classes in Food 101, 0.2 for classes outside Food 101, and 0.5 overall. Not surprisingly, we do better on the original Food101 classes, since the new classes often correspond to side dishes or smaller food items, and are much less frequent in the training data. The top 3 classes were: edamame (0.987), macarons (0.976), hot and sour soup (0.956). The bottom 3 classes were: cream (0.015), garnish (0.014), peppers (0.010). Figure 4 shows the precision-recall curves for some of these classes.

5. Semantic image segmentation

In addition to predicting the presence of certain foods, it is useful to localize them in the image. Since most foods are amorphous, it is better to segment out the region corresponding to each food, rather than putting a bounding box around them. Such a segmented image will enable further analysis, such as counting and size estimation (see below), which is essential for nutrition estimation. We can also allow the user to interactively edit the estimated segmentation mask, in order to improve accuracy of the system (although we leave this to future work).

To train and evaluate semantic image segmentation systems, we took a subset of the Food201-MultiLabel dataset (12k images), and asked raters on a crowd computing platform to manually segment each of the food items that have been tagged (in an earlier stage) in that each image. We used 1 rater per image, and an internal tool that leverages grabcut to make this labeling process reasonably fast. Raters had the option of skipping foods that were too hard to segment. Thus some foods are not segmented, resulting in false negatives. We call this the **Food201-segmented** dataset.

Note that we are segmenting each class, as in the PASCAL VOC semantic segmentation challenge [13]. Arguably, instance-level segmentation (see e.g., [17]) would be more useful, since it distinguishes different instances of the same class, which would enable us to count instances.[9]. However, this is quite difficult. For example, consider distinguishing pancake 1 from pancake 2 in the food image in Figure 1: this is quite challenging, since the top pancake almost completely covers the bottom one. Furthemore, segmenting the eggs into 2 instances is even harder, since the boundary is not well defined. We leave instance-level segmentation to future work.

To tackle the segmentation problem, we use the "DeepLab" system from [6].[10] This model uses a CNN to provide the unary potentials of a CRF, and a fully connected graph to perform edge-sensitive label smoothing (as in bilateral filtering).

The model is initialized on ImageNet, and then fine-tuned on Food201-segmented, which takes about 1 day on a single GPU. For the 3 CRF parameters, which control the strength of the edge potentials, we use the parameter values from [6]; these were chosen by grid search, to minimize validation error on held-out training data.

The label set in the Food201-Segmented dataset is much larger than in the VOC challenge (201 labels instead of just 20). Consequently, the baseline model has a tendency to generate a lot of false positives. To improve performance, we take the probability vector $p(y_k = 1|x)$ computed by the multi-label classifier from Section 4, find the top 5 entries, and then create a binary mask vector, which is all 0s except for the top 5 labels, plus the background. We then multiply the per-pixel label distribution from the segmentation CNN by this sparse vector, before running the CRF smoothing. This provides a form of global image context and improves the results considerably (see below).

We show some sample results in Figure 5. Consider the last row, for example. We see that the context from the multilabel model helps by eliminating false positives (e.g., caprese salad and caesar salad). We also see that the ground truth is sometimes incomplete (e.g., the burrito is not actu-

[9] It is more natural for the user if the system records in their food diary that they ate 3 slices of pizza rather than, say, 120 ounces of pizza. It is also much easier for the user to interactively fix errors related to discrete counts than continuous volumes. Of course, this assumes we know what the size of each slice is. We leave further investigation to future work.

[10] At the time this paper was submitted, DeepLab was the best performing method on the PASCAL VOC challenge (see http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6). At the time of writing the camera ready version, the best performing method is an extension of DeepLab that was additionally trained on MS-COCO data.

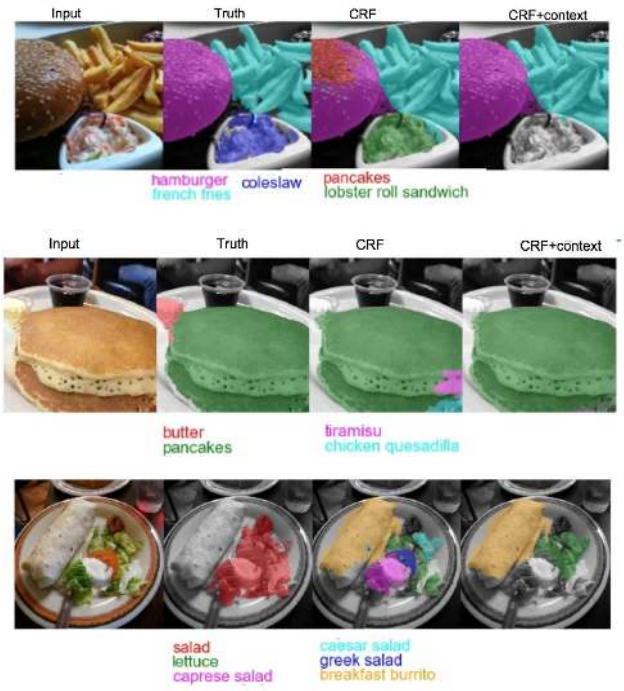

Figure 5. Examples of semantic image segmentation on some images from the Food201-Segmented test set. First column: original image. Second column: ground truth. Third column: predictions using CNN/CRF. Fouth column: predictions using CNN/CRF with multilabel context. Best viewed in color.

CRF?	Context?	Acc	Recall	IoU
0	0	0.71	0.30	0.19
1	0	0.74	0.32	0.22
0	1	0.74	0.32	0.23
1	1	0.76	0.33	0.25

Table 3. Performance on the Food101-Segmented test set.

ally labeled by the human). Finally, we see that the label space is somewhat arbitrary: the ground truth uses the term "salad", whereas the model predicts "lettuce". Currently we ignore any semantic similarities between the labels.

The performance of our system with and without the CRF, and with and without multilabel context, is shown in Table 3. The performance in terms of IoU is much lower than on the PASCAL VOC 2012 segmentation challenge. We believe this is due to several factors: (1) we have 201 foreground labels to predict, whereas VOC just has 20; (2) our labeled dataset suffers from incompleteness, which can result in correct predictions being erroneously being counted as false positives, as well as "polluting" the background class during training; (3) our task is arguably intrinsically harder, since the categories we wish to segment are more deformable and varied in their appearance than most of the VOC categories.

6. Volume estimation

Having segmented the foods, the next step is to estimate their physical size (3d volume).

We first predict the distance of every pixel from the camera, using the same CNN architecture as in [12] applied to a single RGB image. We trained our model on the NYU v2 RGBD dataset of indoor scenes, and then fine tuned it on a new 3d food dataset we collected which we call the **GFood3d** dataset (G for Google). This consists of short RGBD videos of 50 different meals from various Google cafes collected using the Intel RealSense F200 depth sensor.[11] In total, the dataset has about 150k frames. (Note that each RGB image has a corresponding depth image, but otherwise this is unlabeled data.)

On a test set of 2,471 images (recorded in a different set of Google cafes), we get an average relative error of 0.18 meters, which is slightly better to the performance reported in [12] on the NYU dataset. Figure 6 shows a qualitative example. We see that the CNN is able to predict the depth map fairly well, albeit at a low spatial resolution of 75 x 55. (For comparison with the F200 data, we upsample the predicted depth map).

The next step is to convert the depthmap into a voxel representation. To do this, we first detect the table surface using RANSAC. Next, we project each pixel into 3d space, exploiting the known intrinsic parameters of the F200 sensor. Finally, we tile the table surface with a 2d grid (using a cell size of 2mm x 2mm), and compute the average height of all the points in each cell, thus creating a "tower" of that height. For an example of the result of this process, see Figure 7.

Given a voxel representation of the food, and a segmentation mask, we can estimate the volume of each food item. To measure the accuracy of this approach, we purchased the "MyPlate" kit from Nasco.[12] This contains rubber replicas of 42 food items in standard sizes, and is used for training nutritionists. We verified the actual size of these items using the water displacement method (we found that the measured size was somewhat smaller than the quoted sizes). We then created 11 "meals" from these food replicas, and recorded images of them using the F200. Specifically, we put the meals on a turntable, and automatically took 100 images as the plate went through a full rotation. The resulting dataset has 1050 depth images. We call this the **NFood-3d** dataset (N for Nasco).

To measure performance of our system, we compute the absolute error in the volume estimate. We can estimate the volume using the true depth map (from F200) or the pre-

[11] The F200 has a spatial resolution of 1920 x 1080 pixels, and a sensing range of 0.2–1.2m. The Kinect2 camera has the same resolution, but a sensing range of 0.8–3.5m. Thus the F200 is a better choice for close up images.

[12] Source: http://www.enasco.com/product/WA29169HR.

Figure 6. (a) An image from the GFood-3d test set. (b) Depth recorded from RealSense RGBD sensor. (c) Depth predicted by CNN.

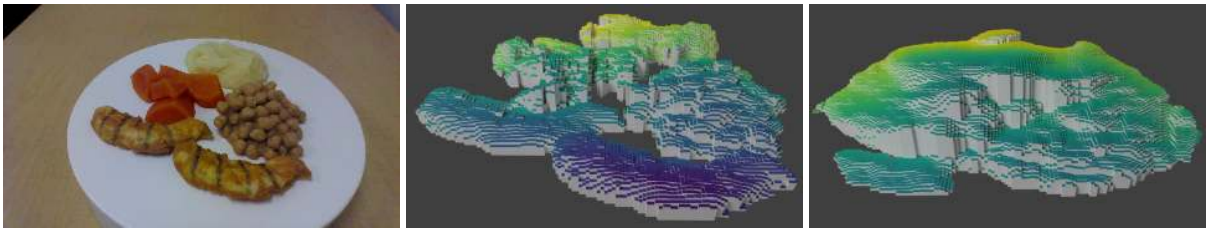

Figure 7. (a) An image from the NFood-3d dataset. (b) Voxel grid derived from RealSense depthmap. (c) Voxel grid estimated from CNN predicted depthmap. Size of grid cell is 2mm x 2mm.

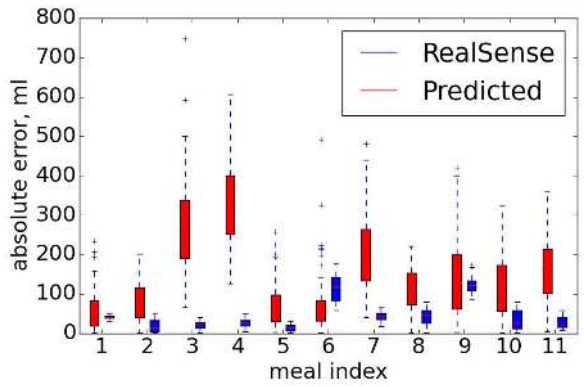

Figure 8. Absolute error in volume estimation (in ml) across the 11 meals in the NFood-3d dataset. Each meal has 100 images, taken from different viewing angles. The boxplots plot the distribution of errors across these 100 images.

dicted depth map (from CNN), and using the true segmentation mask (from human) or the predicted segmentation mask (from CNN). Unfortunately, we have found that our segmentation system does not work well on the NFood images, because their color and texture properties are too dissimilar to real food. However, we were able to compare the quality of using the true depth map and predicted depth map. The results are shown in Figure 8. We see that for most of the meals, our CNN volume predictor is quite accurate.

7. Calorie estimation

The final step is to map from the volume to the calorie content. This requires knowing the calorific density of each kind of food. The standard source for this is the USDA National Nutrient Database (NNDB). The latest version (May 2015) is Standard Release 27 [35], which lists nutritional facts about 8618 basic foods. However, we have noted a discrepancy of up to 35% in the calorie contents between the USDA NNDB and the numbers quoted by Nasco for their food replicas.[13]

A more serious problem is that the NNDB focuses on "raw" foods, rather than cooked meals. For example, NNDB contains information such as the calorie content of a pound of beef, but this does not help us estimate the calorie content of a cooked burger. For prepared foods, it is better to use the USDA Food and Nutrient Database for Dietary Studies (FNDDS) [16], which is derived from NNDB.[14] However, the calorie content can vary a lot depending on exactly how the food was prepared (e.g., grilling vs frying). Consequently, we do not yet have a broad coverage nutritional database for prepared foods, and therefore we have not yet been able to conduct an end-to-end test of our system outside of the restaurant setting.[15]

[13] According to http://www.enasco.com/product/WA29109HR, "nutrition data is provided by The Nutrition Company using their FoodWorks nutrient analysis software". Recall that these food replicas are used to train professional nutritionists, so the information cards that accompany them are designed to be as accurate as possible.

[14] For a detailed comparison of NDB-SR and FNDDS, see http://www.nutrientdataconf.org/PastConf/NDBC36/W-3_Montville_NNDC2012.pdf.

[15] Companies such as MyFitnessPal have developed much larger nutri-

Figure 9. Screenshot of the mobile app. Note that it is running in airplane mode (no internet connection).

8. Mobile app

The complete system is sketched in Figure 1. We have ported the restaurant detector, meal detector, and food detectors (the multi-label classifiers) to the Android mobile phone system. The app can classify any image in under 1 second. The memory footprint for the model is less than 40MB (using unquantized floats to represent the CNN weights). However, we have not yet ported the segmentation or depth estimation CNNs.

To use the app, the user takes a photo and then our system processes it. First the system determines if the image contains a meal, and if you are at a known restaurant. If so, it applies the multilabel classifiers. We sort the possible labels by their predicted probability, taking all those above a threshold of 0.5, and then truncating to the top 5, if necessary. We then show these labels to the user (see Figure 9 for a screenshot). The user can dismiss any labels that are incorrect. He/ she can also enter new labels, either by selecting from a pull-down menu (sorted in order of probability), or by entering free text. The user's image and labels are then optionally stored in the cloud for subsequent offline model retraining (although we have not yet implemented this).

9. Related work

There is a large number of related prior publications. Here we mention a few of them.

In terms of public datasets, we have already mentioned Food101 [3], which has 101 classes and 101k images. In addition, there are various smaller datasets, such as: PFID [7], which has 61 classes (of fast food) and 1098 images; UNICT-FD889 [14], which has 889 classes and 3853 images; and UECFOOD-100 [24], which has 100 classes (of Japanese food), and 9060 images.

tional databases using crowd sourcing [22], but this data is proprietary, and its quality is not guaranteed (since most casual users will not know the true nutritional content of their food).

In terms of classifiers, [3, 18] use SVMs for generic foods. [2] and [1] develop restaurant-specific multi-label classifiers. Some recent papers on food segmentation include [28, 33, 38, 37].

There are many papers that leverage structure from motion to develop a 3d model of the food, including [28, 21, 10, 36, 32]. However, all these methods require multiple images and calibration markers. In terms of single images, [4] use parametric shape models for a small set of food types (e.g., sphere for an apple), and [19] use a nearest neighbor regression method to map the 2d image area to physical mass of each food item.

There are very few papers that predict calories from images. [33] apply semantic image segmentation, and then train a support vector regression to map from the number of pixels of each class to the overall number of calories in the meal. [4, 19] calculate calories by multiplying the estimated volume of each food by the calorie density. [26] use crowdsourcing to estimate calories. [1] relies on the restaurant's menu having the calorie information.

10. Discussion

Besides its obvious practical use, the Im2Calories problem is also very interesting from the point of view of computer vision research. In particular, it requires solving various problems, such as: fine-grained recognition (to distinguish subtly different forms of food); hierarchical label spaces (to handle related labels); open-world recognition (to handle an unbounded number of class names); visual attribute recognition (to distinguish fried vs baked, or with or without mayo); instance segmentation; instance counting; amodal completion of occluded shapes [20]; depth estimation from a single image; information fusion from multiple images in real time, on-device; etc. We have tackled some of these problems, but it is clear that there is much more work to do. Nevertheless, we believe that even a partial solution to these problems could be of great value to people.

References

[1] O. Beijbom, N. Joshi, D. Morris, S. Saponas, and S. Khullar. Menu-match: restaurant-specific food logging from images. In *WACV*, 2015.

[2] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa. Leveraging context to support automated food recognition in restaurants. In *WACV*, pages 580–587, 2015.

[3] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101: Mining discriminative components with random forests. In *ECCV*, 2014.

[4] J. Chae, I. Woo, S. Kim, R. Maciejewski, F. Zhu, E. J. Delp, C. J. Boushey, and D. S. Ebert. Volume estimation using food specific shape templates in mobile image-based dietary assessment. In *Proc. SPIE*, 2011.

[5] C. Champagne, G. Bray, A. Kurtz, J. Montiero, E. Tucker, J. Voaufova, and J. Delany. Energy intake and energy expenditure: a controlled study comparing dietitians and non-dietitians. *J. Am. Diet. Assoc.*, 2002.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[7] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. PFID: Pittsburgh fast-food image dataset. In *ICIP*, pages 289–292, 2009.

[8] F. Cordeiro, E. Bales, E. Cherry, and J. Fogarty. Rethinking the mobile food journal: Exploring opportunities for lightweight Photo-Based capture. In *CHI*, 2015.

[9] F. Cordeiro, D. Epstein, E. Thomaz, E. Bales, A. K. Jagannathan, G. D. Abowd, and J. Fogarty. Barriers and negative nudges: Exploring challenges in food journaling. In *CHI*, 2015.

[10] J. Dehais, S. Shevchik, P. Diem, and S. G. Mougiakakou. Food volume computation for self dietary assessment applications. In *13th IEEE Conf. on Bioinfo. and Bioeng.*, pages 1–4, Nov. 2013.

[11] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *CVPR*, pages 3450–3457, June 2012.

[12] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common Multi-Scale convolutional architecture. *Arxiv*, 18 Nov. 2014.

[13] M. Everingham, S. M. Ali Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 25 June 2014.

[14] G. M. Farinella, D. Allegra, and F. Stanco. A benchmark dataset to study the representation of food images. In *ECCV Workshop Assistive CV*, 2014.

[15] FDA. www.fda.gov/Food/ IngredientsPackagingLabeling/LabelingNutrition/ucm248732.htm.

[16] USDA FNDDS. www.ars.usda.gov/ba/bhnrc/fsrg.

[17] B. Hariharan, P. Arbel, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[18] H. He, F. Kong, and J. Tan. DietCam: Multiview Food Recognition Using a MultiKernel SVM. *IEEE J. of Biomedical and Health Informatics*, 2015.

[19] Y. He, C. Xu, N. Khanna, C. J. Boushey, and E. J. Delp. Food image analysis: Segmentation, identification and weight estimation. In *ICME*, pages 1–6, July 2013.

[20] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Amodal completion and size constancy in natural scenes. In *Intl. Conf. on Computer Vision*, 2015.

[21] F. Kong and J. Tan. DietCam: Automatic dietary assessment with mobile camera phones. *Pervasive Mob. Comput.*, 8(1):147–163, Feb. 2012.

[22] R. Macmanus. How myfitnesspal became the king of diet trackers, 2015. readwrite.com/2015/02/23/trackers-myfitnesspal-excerpt.

[23] C. K. Martin, H. Han, S. M. Coulon, H. R. Allen, C. M. Champagne, and S. D. Anton. A novel method to remotely measure food intake of free-living individuals in real time: the remote food photography method. *British J. of Nutrition*, 101(03):446–456, 2009.

[24] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of Multiple-Food images by detecting candidate regions. In *ICME*, pages 25–30, July 2012.

[25] Mealsnap app. tracker.dailyburn.com/apps.

[26] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos. PlateMate: Crowdsourcing nutrition analysis from food photographs. In *Proc. Symp. User interface software and tech.*, 2011.

[27] Google places API. https://developers.google.com/places/.

[28] M. Puri, Z. Zhu, Q. Yu, A. Divakaran, and H. Sawhney. Recognition and volume estimation of food intake using a mobile device. In *WACV*, pages 1–8, Dec. 2009.

[29] Rise app. https://www.rise.us/.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575*, 2014.

[31] D. Schoeller, L. Bandini, and W. Dietz. Inaccuracies in self-reported intake identified by comparison with the doubly labelled water method. *Can. J. Physiol. Pharm.*, 1990.

[32] T. Stutz, R. Dinic, M. Domhardt, and S. Ginzinger. Can mobile augmented reality systems assist in portion estimation? a user study. In *Intl. Symp. Mixed and Aug. Reality*, pages 51–57, 2014.

[33] K. Sudo, K. Murasaki, J. Shimamura, and Y. Taniguchi. Estimating nutritional value from food images based on semantic segmentation. In *CEA workshop*, pages 571–576, 13 Sept. 2014.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[35] USDA National Nutrient Database for Standard Reference, Release 27 (revised). http://www.ars.usda.gov/ba/bhnrc/ndl.

[36] C. Xu, N. Khanna, C. B. Y. He, and A. Parra. Image-Based food volume estimation. In *CAE workshop*, 2013.

[37] W. Zhang, Q. Yu, B. Siddiquie, A. Divakaran, and H. Sawhney. 'Snap-n-eat': Food recognition and nutrition estimation on a smartphone. *J. Diabetes Science and Technology*, 9(3):525–533, 2015.

[38] F. Zhu, M. Bosch, N. Khanna, C. J. Boushey, and E. J. Delp. Multiple hypotheses image segmentation and classification with application to dietary assessment. *IEEE J. of Biomedical and Health Informatics*, 19(1):377–388, Jan. 2015.

[39] F. Zhu, M. Bosch, I. Woo, S. Kim, C. J. Boushey, D. S. Ebert, and E. J. Delp. The use of mobile devices in aiding dietary assessment and evaluation. *IEEE J. Sel. Top. Signal Process.*, 4(4):756–766, Aug. 2010.