Food Recognition using Ingredient-Level Features

Jay Baxter
Massachusetts Institute of Technology
jbaxter@mit.edu

Abstract

Food recognition is a difficult problem, because unlike objects like cars, faces, or pedestrians, food is deformable and exhibits high intra-class variation. This paper considers the approach of analyzing a food item at the pixel-level by classifying each pixel as a certain ingredient, and then using statistics and spatial relationships between those pixel ingredient labels as features in an SVM classifier. We experimented with multiple variations on past methods, and found that using pixel ingredient labels to identify food greatly increases classification accuracy, but at the expense of higher computational cost.

1. Introduction

Food recognition has only become a fairly popular topic in the last few years, which is largely a result of the quickly growing number of people who routinely take pictures of their food with cell phone cameras before eating it. The main application of food recognition is to create a nutritional information phone application that is able to analyze these pictures of food and deduce the nutritional content of the food eaten. The most critical and difficult step of this process is recognizing the type of food in the picture: once the type of food is determined, estimating quantity and nutritional information is much easier. Quantity can be determined by asking participants to include a thumb in their picture or have the food a fixed distance away from the camera. Nutritional information can be looked up in official databases. Therefore, this paper addresses the problem of food recognition: determining what type of food is in the picture, given that we know that the input picture is of food, and that the food is the main focus of the picture. We assume that the background is rather plain, like a plain tabletop.

Even with such a restricted problem domain, food recognition is a very hard problem. Unlike other types of objects where object recognition has been more successful, such as faces, cars, and pedestrians, food is very deformable and has high intra-class variation, as is shown in Figure 1. Pasta is deformable, meaning that it is amorphous. The definition of pasta has nothing to do with shape, and only has to do with its ingredients and method of preparation. Ingredients and method of preparation manifest themselves in the visible features shape, color, and texture. However, even in a type of food we often think of as having a rigid shape and structure, a Big Mac, there is high intra-class variation: sometimes the meat is hidden underneath the bun, sometimes the cheese isn't visible, sometimes the lettuce is hidden, etc.

In this paper, we address these problems by trying to identify the ingredients that make up a food item. Humans describe food items and their differences in terms of the ingredients they contain and the way the ingredients are arranged, so it makes intuitive sense that we may be able to glean extra information by extracting features at an intermediate ingredient level instead of just as the original pixel level. After labeling the ingredients in the image, we extract features that describe the relatives quantities of each ingredient (which is useful for distinguishing a salad from a hamburger, because a salad has much more lettuce) as well as the spatial relationships between the ingredients (which is useful for determining a Big Mac from a normal hamburger, since we expect to see bread-meat-bread-meat-bread instead of bread-meat-bread).

2. Related Work

This paper relies heavily on past work by Yang et al. [11] on identifying food using pairwise statistics of local features, which was the paper that introduced the idea of randomly sampling ingredient pixels from the food and using histograms of statistics such as distance and orientation as features in an SVM classifier.

Pixel-level ingredients labels are also used as intermediate features in [12], but that paper focused only on the global ingredient histogram instead of the pairwise statistics.

Other work has considered combining both global and local features, such as global histograms, which is an approach that I take in this paper [1].

Less related work on food recognition has focused on extracting many different features, such as bag of SIFT and

Figure 1. Food recognition is a hard problem: deformable objects and high intra-class variation.

textons [5]. Joutou and Yanai combined these features into one classifier with multi-kernel learning [6].

Kong and Tan showed that food can be classified much better with multiple viewpoints [7], but this work is only applicable to a cell phone application if the user takes video of her food instead of just a picture.

In this paper, we focus on building off of Yang et al.'s approach in [11] by integrating ideas from these other papers.

3. Baseline Methods

To compare to our other methods (and to combine with them later to achieve even greater performance), standard object recognition techniques were used.

3.1. Bag of SIFT

The SIFT descriptor [8], when combined in a bag of words model, has been shown to perform well as a feature for image classification [4]. In this approach, the first step is to generate a discrete dictionary of SIFT features, which is obtained by running k-means clustering. Each SIFT descriptor is matched to the closest cluster center. Then, each image is described by the frequencies of each SIFT cluster in that image. Classification is performed in this paper with a multi-class SVM with χ^2 or histogram intersection kernel.

3.2. RGB Histogram

One other common feature is the RGB histogram, which is constructed by dividing the color space up into bins, and placing each pixel in the bin that its RGB values fall into. I divide each color evenly from 0-255 into four bins, 0-63, 64-127, 128-191, and 192-255, which results in a 64-dimensional histogram, which is again classified using an SVM.

On the PFID dataset, where images have very clean white backgrounds, the RGB histogram is able to effectively capture the size of the food, since no other pixels besides the background pixels are white enough to fall into that bin. As a result, RGB histograms work much better on this dataset than they would without the food-detection problem already solved for us in advance.

4. Ingredient Segmentation

In order to consider ingredient-level features, the most important step is labeling each pixel as a specific ingredient. To accomplish this task, I used semantic texton forests (STF) [10] because they have been shown to work well on difficult segmentation datasets, and because Yang et al. showed that they give reasonable performance on ingredient segmentation with relatively little training data [11]. I

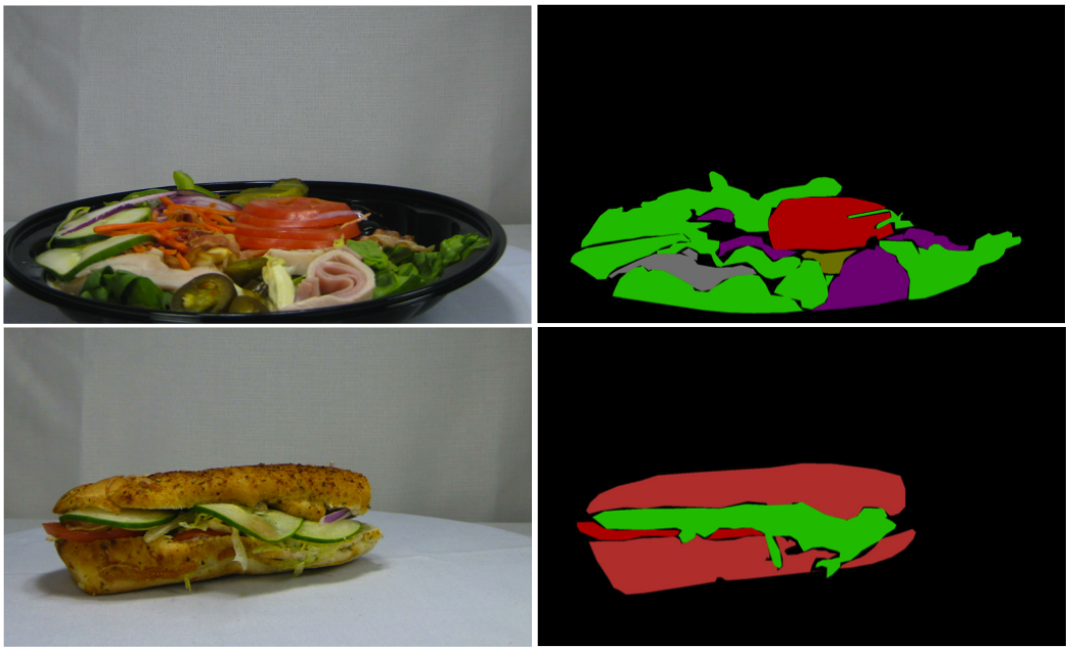

Figure 2. The first two rows show hand-labeled ingredient labels, used as training data for the Semantic Texton Forest.

Figure 3. On the left are sample input pictures to the trained STF, and on the right are output visualizations shown such that each pixel is colored the color of the ingredient that it is most likely to be, according to our segmentation. In the Big Mac, the meat and lettuce are perfectly identified, and most of the bread was identified correctly. However, there is a fair amount of noise: part of the bun is labeled as bacon (purple). Clearly, these ingredient labels can be quite noisy, and any improvements in the ingredient labeling algorithm will cause corresponding improvements in this food recognition classifier.

hand labeled the pixel-level ingredients for 18 images that were used as training data for the STF (which was enough to cover every ingredient at least 3 times), and then ran the trained STF segmented on the rest of the images (see Figure 3 for sample images). Even with 18 training images the algorithm performs well enough, but future work should in-

vestigate the improvements that may occur with more training data.

I needed to hand-select the ingredients that STF would be trained to detect. I found that 8 ingredients (not including the background) were sufficient to fully describe the types of ingredients found in PFID (note that these ingredients differ from the ingredients used by Yang et al. [11] - they are much more easily distinguishable. In particular, Yang et al. used some ingredients that I couldn't even distinguish by eye in any of the images in this dataset (egg and cheese) or that only appeared in a few instances (pork). The ingredients I selected, with an emphasis on making sure their appearances are sufficiently different, were beef, white meat (chicken/turkey/etc.), bread, green vegetables, tomatoes, cheese, and chocolate. In my preliminary tests, I found that STF gave better looking results with these ingredients, although I did not have time for a full quantitative evaluation of how ingredient selection affects classification accuracy (each full run takes of STF takes over a day on this dataset), so that would be interesting to explore in future work.

STF assigns a probability distribution to each pixel that describes the probability that that pixel is a certain ingredient. Typically, background pixels are very confidently labeled as background ($> 99\%$), and food items are more commonly labeled with confidences between 30 and 60 percent. We use these soft-labels in the next step.

5. Ingredient Features

The heard of this food recognition algorithm is extracting histogram features from the probabilistic ingredient labels that we got as output from STF.

5.1. Global Ingredient Histogram

The most naive approach is to construct an ingredient histogram that simply represents the proportion of food that is of a given ingredient type. We have nine bins - one for each ingredient, plus background. This approach captures a large amount of the information that we get from the ingredient labels.

5.2. Pairwise Ingredient Features

The way we extract local features is to examine every pair of points in the food (excluding the background), extract features from each pair, and add them up in a histogram. However, since the images in PFID are quite large, examining every pair of points is not computationally feasible. Instead, I sampled 500 pixels randomly from the food item and computed the pairwise statistics between them. It should be noted that even when only using 500 pixels, these computations take on the order of days for all images in the dataset with a modern desktop computer. For a single image, processing can be done in about 2 minutes, which makes this approach applicable only when instantaneous feedback is not necessary.

5.2.1 Pairwise Distance

We compute the base-2 logarithm of the pairwise distance between points instead of the absolute distance. While unintuitive, this logarithm will give us scale invariance, as I describe in the Invariance subsection, and empirically, it does not decrease its usefulness as a feature. This feature allows us to capture whether there are small pieces of bread scattered around, like croutons in a salad, or lots of bread pixels all in one giant blob. The results are stored in an 8x8x12 histogram, where there is one bin for each non-background ingredient, and 12 bins for different thresholds of distance. Since pixels have probabilistic ingredient labels, we weight each entry in the histogram by the probability that each point in the pair is a given ingredient. For example, if the first point has a 70% of being bread and the second has a 30% change of being bread, and the distance between them is 10, then we add 0.21 ($70\% \times 30\%$) to the (bread, bread, 10) entry of the histogram. Experimenting with different weighting functions is a way that this work could be extended, but this one worked well and is intuitive.

5.2.2 Pairwise Orientation

We compute the angle between every pair of randomly sampled points, and quantize the resulting angle into 12 distinct bands, each spanning 15 degrees (since the relationship between points needs to be symmetric, there are 180 total degrees instead of 360). Like the pairwise distance feature, the resulting histogram is an 8x8x12 histogram, and each entry is multiplied by the probabilities of both the pixel ingredient probabilities. This feature can capture details like whether an ingredient is all in a long line. In that case, the orientations would all be clustered in one bin for a certain ingredient.

5.2.3 Pairwise Midpoint Ingredient Labels

Unlike the previous two pairwise features, this feature is an 8x8x8 histogram, where each dimension represents the ingredient label of one of three points: the two randomly sampled points, and their midpoint. The same weighting is used as in the previous two pairwise features. The goal of this feature is to capture the spatial relationships that characterize some of the more structured food items, like sandwiches. We would expect there to be a number of bread - meat - bread pairs in a sandwich. However, in practice, this feature proves to be one of the most useless, and does a particularly bad job recognizing sandwiches (the RGB histogram does better). It doesn't work as well as other methods because it relies more heavily on accurate pixel ingredient labels (the

bread and meat labels need to be correct), and the meat in a sandwich is often very small compared to the rest of the sandwich, so if we randomly sample pairs of points, there are very few examples where we actually see bread - meat - bread patterns.

5.2.4 Invariance

I used the clever histogram normalization techniques used by Yang et al. [11] where we compute the mode of the pairwise distance and orientation, and then recenter those histograms in the appropriate direction so that it is centered on the mode. This technique gives rotation invariance, because now rotating the image wouldn't change the orientation histogram. Additionally, since the distance is actually the log of the distance, a scale change of the image only shifts the distance histogram, which will be re-centered, therefore giving scale invariance.

6. SVM Classifier

We use an SVM classifier with the histogram intersection kernel from [9], and the implementation provided by the authors, which is built on top of libsvm [2]. There are many kernels for comparing histograms. The two most prevalent are the χ^2 kernel and the histogram intersection kernel. For histograms a and b with $a_i > 0$ and $b_i > 0 \, \forall i$, the histogram intersection kernel is defined as:

$$K(a,b) = \sum_{i=1}^{n} min(a_i, b_i) \tag{1}$$

The χ^2 kernel is defined as:

$$K(a,b) = \exp\left(\sum_{i=1}^{n} \frac{(a_i - b_i)^2}{a_i + b_i}\right) \tag{2}$$

In these experiments, I used the histogram intersection kernel, but similar results would be obtained with a χ^2 kernel, or likely even with a linear kernel if the data is appropriately scaled.

7. Dataset

It is very difficult to compare results between different methods in the field of food recognition because there is no standard dataset. The Pittsburgh Fast Food Image Dataset (PFID) [3] was an attempt at standardizing a food recognition dataset. This dataset contains 101 different fast food items (e.g. McDonald's Big Mac and Wendy's Baconator), where each food item was collected on 3 different days, and for each of those 3 instances of a food item there are 6 pictures of the food item from different orientations (each 60 degrees apart) on a white table with a white backdrop. The dataset, therefore, is designed for evaluation of food recognition, since the food is essentially already detected.

However, PFID has not been widely used since its release, partially because the data is not in a ready-to-use format: it requires a significant amount of cleaning up. I had to go through the entire dataset by hand and remove certain food items that weren't really food (some pictures were of soft drink cups), and remove all food items that had less than 18 images.

Additionally, the dataset is too difficult. Even humans are not able to easily categorize some similar food items, as shown in Figure 4.

Therefore, I manually divided the food items up into seven larger categories: sandwiches, salads/sides, chicken, breads/pastries, donuts, bagels, and pizza. These categories are easily distinguishable by humans, and therefore make for a more realistic food recognition challenge. However, some of the food did not fit well into these categories (there was one instance of a granola bar, for example), so I had to remove those outliers. In the end, there were 88 different food types and 7 major categories.

8. Results

Features	88-Cat. Acc.	7-Cat. Acc.
GIH	36.8	69.1
RGB	35.5	71.2
MID	31.8	71.2
OR	37.7	73.7
DIST	36.1	75.4
DIST + GIH	40.5	76.3
OR + GIH	40.5	76.4
RGB + GIH	36.6	76.5
MID + GIH	37.5	75.5
DIST + MID + GIH	40.7	76.7
MID + OR + GIH	40.4	77.0
All except RGB	40.4	77.0
DIST + OR + GIH	40.7	77.0
All	37.5	81.2

Figure 5. Classification results using an SVM with various image features, sorted by increasing accuracy on 7 categories. RGB means RGB histogram, GIH means global ingredient histogram, MID means pairwise midpoint pixel label histogram, OR means pairwise orientation histogram, DIST means pairwise distance histogram.

I used the experimental setup proposed for PFID by Chen et al. [3] where we perform 3-fold cross validation over the images, where each fold contains the 6 pictures of every food item obtained on the same day. Since each food was obtained on three separate days, each fold contains six pictures of each food item.

Using the experimental setup described above, we found that the bag of SIFT gives 13% accuracy on the 88-category

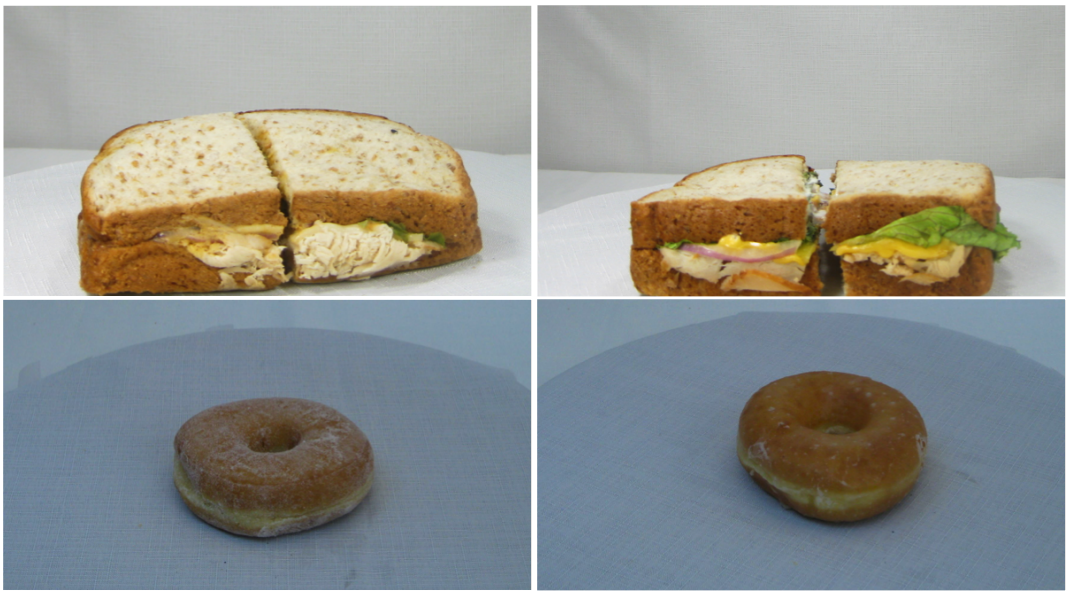

Figure 4. Example images from the Pittsburgh Fast Food Image dataset. On top are Arby's turkey and swiss sandwich and Arby's turkey and ranch sandwich, and on the bottom are Dunkin Donuts' sugar raised donut, and Dunkin Donuts' glazed donut. It is incredibly hard for even a human to differentiate between the two Arby's sandwiches, so I spent most of my attention optimizing for classification among the 7 major categories (e.g. sandwich and donut) instead of at the individual food item level.

data and 59% accuracy on the 7-category data, which is by far the worst of any method I tried. The performance of the RGB histogram is significantly better than what was found in [11] and [3], possibly because their implementation ignored the background data, and possibly because of differences in how we cleaned up the PFID data (Yang et al. [11] reduced the dataset to 61 specific food categories instead of 88). Either way, background information is critical because it provides food size information, and should not be ignored.

The results show that we can get a 10% improvement in accuracy, from the simple RGB histogram's 71.2% accuracy on the 7 major food categories to the combination of all methods, which gets 81.2%. It is reassuring that using all the features together provides better accuracy than any other combination of features, telling us that every feature captures some information that another one doesn't.

However, there is obviously a certain type of information that these features are not able to catch. My hypothesis is that this is because ingredients themselves can't describe an entire food item: if an object is all bread (as is often the case with donuts, pastries, and bagels, which are roughly the same size), then how do we tell them apart? While the approach in this paper uses both shape and texture data, we use texture to classify ingredients, and the shape between the ingredients. Shape and texture are both very important features to consider at the food-level instead of just the ingredient level, and I think that future research should explore those opportunities.

One failure of this approach is its inability to recognize sandwiches. The entire point of the midpoint feature was to capture structured foods like sandwiches, but this classifier frequently confuses sandwiches with pastries, donuts, and bagels. Future works needs to find a better way to encode the characteristic structure of a sandwich, without losing the ability to classify amorphous foods like salads that this method has successfully done.

9. Conclusion and Future Work

Using histogram features to represent ingredient-level spatial features provides better performance than other baseline methods, and significantly better performance than the simple bag of SIFT model, which loses lots of critical information about food.

The main limitation of the approach presented in this paper is that it assumes that the food is easily distinguishable from the background. Therefore, the problem of food recognition still needs to be separately considered, and should be approached in any future work.

Possibilities for future work include finding a better ingredient-level feature to recognize sandwiches, labeling more ground-truth ingredient data to improve the segmentation performance of STF, and incorporating features that consider image-level texture instead of only using texture to classify ingredients.

In conclusion, this method gives a significant improve-

ment in classification accuracy over baseline methods, but the pairwise features can take 1-2 minutes to compute on a modern computer. If that time delay is acceptable, this method would be easy to integrate into a mobile phone application.

References

[1] M. Bosch, F. Zhu, N. Khanna, C. Boushey, and E. Delp. Combining global and local features for food identification in dietary assessment. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 1789–1792. IEEE, 2011.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. Pfid: Pittsburgh fast-food image dataset. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 289–292. IEEE, 2009.

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22, 2004.

[5] H. Hoashi, T. Joutou, and K. Yanai. Image recognition of 85 food categories by feature fusion. In *Multimedia (ISM), 2010 IEEE International Symposium on*, pages 296–301. IEEE, 2010.

[6] T. Joutou and K. Yanai. A food image recognition system with multiple kernel learning. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 285–288. IEEE, 2009.

[7] F. Kong and J. Tan. Dietcam: Regular shape food recognition with a camera phone. In *Body Sensor Networks (BSN), 2011 International Conference on*, pages 127–132. IEEE, 2011.

[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[9] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[10] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[11] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar. Food recognition using statistics of pairwise local features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2249–2256. IEEE, 2010.

[12] M. Zhang. Identifying the cuisine of a plate of food.