# Discovering Attributes for Food Description and Recognition

**Student: Jennifer Nguyen**
**Supervisor: Dr. Gabriel Brostow**
MSc Computational Statistics and Machine Learning
September 2013

This report is submitted as part requirement for the MSc Degree in Computational Statistics and Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text.



Department of Computer Science
University College London

# Abstract

Automated food recognition has great value in terms of health benefits, as it provides a more efficient and accurate method of recording an individual's diet. It is the high variability within the same food groups and yet, the subtle distinctions between different food groups that makes food recognition a difficult task.

In this thesis, we evaluated and compared three popular image descriptors for their performance on two challenging food data sets: 50 Chinese Foods and ImageNet Foods, the latter of which we collected from ImageNet. We also proposed a method to automatically discover food-related attributes to give rise to a recipe-based food descriptor. Our food descriptors were able to improve the performance of the baseline descriptors by encoding attributes such as ingredients, cooking method, and nutritional content. We also demonstrated that our food descriptors are capable of knowledge transfer for zero-shot learning.

## Acknowledgements

I am grateful to the following people who have made this thesis possible and an enjoyable experience:

- My supervisor, Dr. Gabriel Browstow, for introducing me to the exciting field of computer vision and helping me navigate it to find answers to my questions.
- Oisin Mac Aodha, for taking the time to help me develop my ideas.
- The Prism Group, for their advice and suggestions when I needed help.
- My good friends Kasra Khalaj, Emily Kirsch, Margie McKerron, and Sharon Zheng for proofreading this thesis and providing constructive feedback.
- My parents for their continual support of my endeavours and who inspire me to do my best.

Jennifer

# Contents

# List of Figures

# List of Tables

## Chapter 1

# Introduction

Studies have shown that there is a link between an individual's food intake and chronic diseases such as cancer, diabetes, and obesity [6, 37, 16]. To better study the relationship between food and these diseases, health care practitioners suggest that individuals keep a food diary, documenting what they eat each day and how much. Health care practitioners can then analyze the nutritional content of an individual's diet and elucidate relationships between certain foods and food-related diseases. From these insights, they can develop more effective and preventative interventions to mitigate these diseases.

## 1.1 The need for automated food recognition

Current methods to record one's daily food intake include traditional pen and paper. Alternatively, there are numerous smartphone applications that allow users to enter the name of the food and how many servings they ate (e.g., Lose It!™, Food Tracker™, Calorie Counter™). The application retrieves the nutritional information from a database and computes the total caloric intake. Manually logging every food item, however, can become onerous in the long term. Furthermore, self-reported methods can be inaccurate as humans tend to underestimate their food consumption [21, 43].

As a remedy, Martin et al. developed the Remote Food Photography Method whereby users upload a before and after photo of their meal to a research centre [33]. The images are analyzed by registered dieticians for nutritional content. While this approach is more accurate than self-reports, it requires a human expert to perform the nutritional analysis, which can be costly in terms of time and money.

Another approach by Noronha et al. uses crowd-sourcing instead of an expert which achieves comparable results [36]. The Platemate crowd-sourcing system works in a similar manner. Users upload images of their meal to a server, which are analyzed by turkers using Amazon Mechanical Turk. The turkers identify the food items in the image and quantify the amount present. The final results are then returned to the user. The disadvantage is that the entire process takes on average 94 minutes to complete and costs $1.40 [36].

In summary, current diet recording methods are onerous, inaccurate, costly, inefficient, or a

combination of all of the above. As such, our goal is to develop a system that can automatically and accurately recognize food items from images using computer vision techniques. We hope to eliminate the need of a human expert while maintaining the same accuracy in a more cost-effective and timely manner.

## 1.2 Recognizing food is challenging

Training a computer vision system to recognize food items is challenging for many reasons. Even though the class space is restricted to only food, there are nevertheless many classes of food to identify. It is infeasible to train the system with every existing food item. However, work has been done to develop methods to classify novel classes, i.e., classes that the system has not been trained to recognize [3, 2, 17, 30, 39].

Another difficulty is the inherent variability of foods. Figure 1.1 presents a typical case of a food class having variable appearance. While all images are of a hamburger, they all look different because hamburgers are customizable using different ingredients, which is true for all other foods. Furthermore, the images differ due to lighting conditions, camera angle, and background. A good vision system should be robust to these variances and recognize that all these images show a hamburger.

Figure 1.1. Example of intra-class variability: food is inherently variable even within the same class, which makes recognition difficult as seen here with these different versions of a hamburger.

While foods of the same class can look different, there are some foods of different classes that look similar. Figure 1.2 illustrates three such examples. The foods look the same in shape, texture, and/or colour but they belong to different classes. Unlike the previous difficulty, where the foods share common attributes because they belong to the same class, these foods share common attributes and yet belong to different classes. This problem is more visually challenging because the differences between these classes are subtle and often times, not visually obvious.

In this thesis, we propose a method to address these challenges.

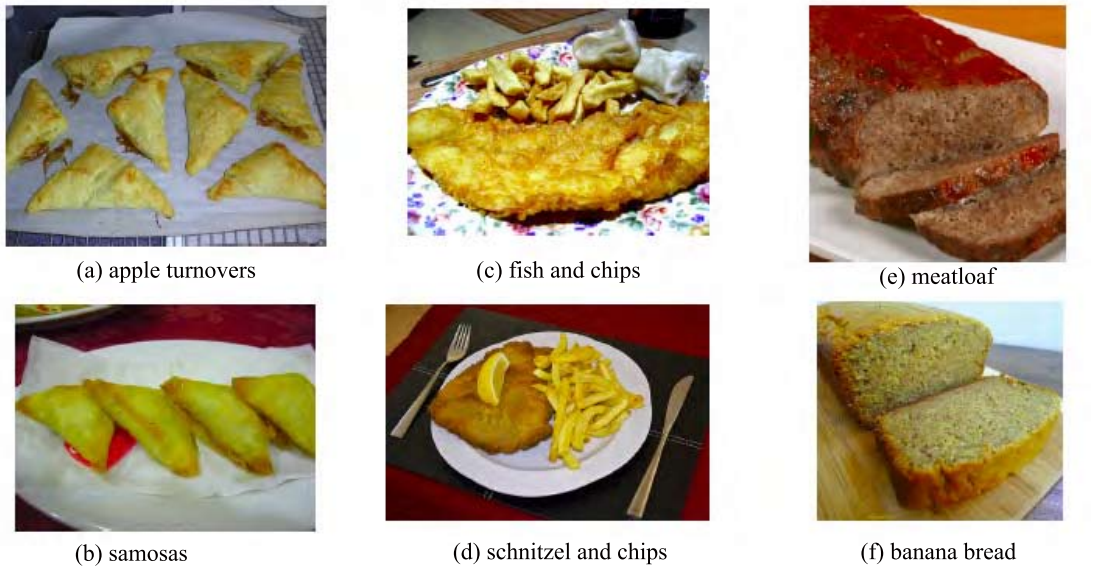



Figure 1.2. Example of inter-class similarity: foods that look similar but belong to different classes poses another challenge for food recognition.

## 1.3 Scope and Goals

To overcome the challenges discussed in Section 1.2, image descriptors need to better capture the finer details of food items (inter-class variability), while still maintaining a view of the bigger picture (intra-class variability). With these objectives in mind, the goals of this thesis are to:

1. Explore and compare the performance of state-of-the-art object recognition descriptors on food images representative of a real-world setting.
2. Automatically discover food-specific attributes to semantically describe a food image via support vector machines and random forests.
3. Integrate these semantic descriptions with the general object recognition descriptors in (1) to specialize for food recognition.

While past work in the field of food recognition has focused on improving recognition accuracy using low-level visual features (cf. Section 2.1), our work differs in that we design a descriptor to improve not only recognition but to describe foods in a way understandable by humans. Even though we work with the assumption that the set of classes is fixed, our food descriptors are capable of inferring the composition of unseen food classes which can be used for zero-shot learning (cf. Section 5.1).

## 1.4 Thesis Overview

Having studied the strengths and weaknesses of the baseline descriptors, we achieved the second and third goals by proposing a fine-grained recognition approach using information from online

recipes. By leveraging the embedded semantic information in the image labels, we constructed a set of recipe-based attributes and used these to describe a food image in a more high-level manner.

The remainder of this thesis is organized as follows. Chapter 2 presents an overview of related works with respect to food recognition, fine-grained recognition, and attribute-based classification. Chapter 3 describes the food data sets and the methodology used to test our descriptors. Chapter 4 explains the various baseline descriptors we used and compares their performance on the data sets. Chapter 5 presents our proposed recipe-enhanced food descriptor and compares the performance of these descriptors to the baseline descriptors. Lastly, Chapter 6 concludes the thesis and suggests ideas for future work.

## Chapter 2

# Related Work

In this chapter we present a literature review of the research related to our work. Section 2.1 discusses our main research interest: the progress in food classification thus far. We also provide a survey of works related to fine-grained recognition (Section 2.2) and attribute-based classification (Section 2.3), two areas that we draw ideas from to achieve our overall goal of improved food recognition.

## 2.1 Food Recognition

The field of food recognition can be considered as a specialized object recognition problem. While it is a fairly new area of research, much work has been done to identify food items and also estimate its the nutritional content [34, 27, 28, 7].

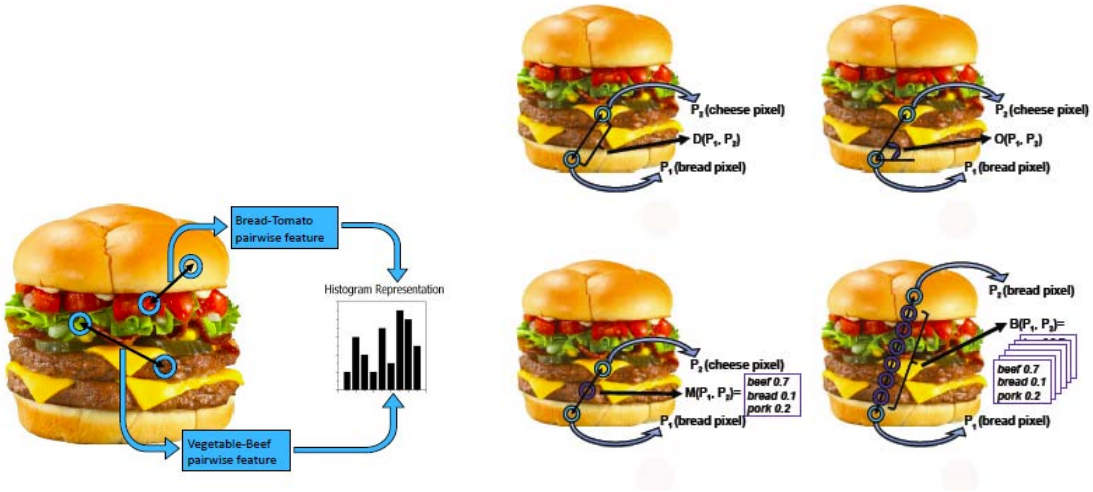
To achieve either goal, work on food recognition has primarily focused on clever feature extraction. Chen et al. collected the Pittsburgh Fast Food Image (PFI) data set, a collection of fast food images of 101 different foods from 11 fast food restaurants, and used it to benchmark the performance of two simple baseline algorithms [7]. The first method used colour histograms as the only feature while the second method used scale-invariant feature transform (SIFT) descriptors. Both methods used support vector machines (SVMs) as the classifier. The performances of the two baselines were not high, but their work showed that SIFT descriptors were more informative in recognizing food items than colour histograms, recognizing twice as many items [7]. Nevertheless, the authors agreed that there is room for improvement with respect to either method. In our work, we combined both features along with additional features to improve recognition accuracy. We also tested our method on data sets that are more diverse and reflective of an average person's diet.

A limitation of the previous method is that it failed to capture spatial relationships between features, which can be important because of the regular composition of some foods. In one of the earliest food recognition works, Yang et al. addressed this shortcoming in their modified bag of features (BoF) model, in which they extracted SIFT descriptors to use with a $K$-nearest neighbours ($K$-NN) classifier [47]. Instead of extracting the SIFT descriptors from the image directly, they first segmented the images into visually similar regions and then computed the

features from each segment to obtain groups of related features. Each segment of the image is classified and the final classification is the one that is the most agreed upon across all segments. Using this approach, they were able to achieve 81.25% accuracy for four food categories: hamburger, ice cream, fries, and cola. However, their method was tested on a very conservative data set and is not expected to generalize well to images with multiple food items.

While the previous method preserved the spatial relationship of features in the same segment, it did not account for the spatial relationships *between* segments. In response, Yang et al. developed a more robust feature representation that captured the geometric orientation of ingredients without the need for segmentation [49]. In their method, they assigned a sample of pixels with a soft-labelling, i.e., a distribution over eight ingredients based on the patch around the pixel using a semantic texton forest (cf. Figure 2.1.a). Using this initial labelling, they computed pairwise statistics between pixels such as distance, orientation, midpoint category, between-pair category, distance and orientation, and orientation and midpoint category (cf. Figure 2.1.b). Each feature is aggregated into a multi-dimensional histogram and used as input into an SVM classifier.

Their novel features achieved a 28.2% accuracy for 61 food items of the PFI dataset. It was found that the joint orientation and midpoint category statistic was the most useful in food categorization, achieving twice the performance of the baseline colour histogram; showing that spatial relationships are significant for food recognition. In our work, we also leveraged ingredient information but unlike their ingredients which are fixed and known *a priori*, we learned the ingredients to adapt to the data. We also incorporated this information to describe foods by their ingredients so the results are more interpretable.




(a) Pairwise local features

(b) Distance, orientation, midpoint category, and between-pair category pairwise local features (clockwise from top-left)

Figure 2.1. Extracting pairwise features for food recognition: Distributions of ingredients between pairs of patches (a) are used to compute local statistics (b).[49].

In contrast, Bosch et al. concentrated on computing global features in addition to local features [6]. After images were segmented to identify the food item boundary, global features such as colour statistics and entropy statistics were computed to get colour and texture information. To extract local features, a Difference of Gaussians filter was applied to detect interest points. Local features including Tamura perceptual features (a type of texture feature), SIFT descriptors, and Haar wavelets were then computed from a patch of pixels surrounding the interest point. Each feature was used to train independent classifiers where the final classification was done by taking the majority vote. For the global features, classification was done using SVMs whereas for the local features, a BoF model with a $K$-NN classifier was used. Unlike in the PFI results [7], it was found that colour-related features were the most useful in discriminating food items, suggesting that the important features vary with the data. In our work, we also used a BoF model with similar features. Since there is likely to be some correlation between the features, we only used one SVM to account for this correlation instead of training independent SVMs for each feature.

Joutou et al. [26] and Hoashi et al. [24] extracted similar features but used a different classification approach. Instead of using each feature separately like Bosch et al. , they used Multiple Kernel Learning (MKL) in which each feature was associated with a separate kernel and linearly combined to form one overall classifier. For each food class, they trained a separate MKL classifier so that the features were weighted accordingly for that particular class. Hence, the most useful feature for a class was automatically learned. Their method achieved 61.3% and 62.6% accuracy for 50 and 85 food categories respectively. In addition, unlike the previous works, the images used were retrieved from the Internet and not taken in a controlled setting, which is more reflective of real life situations. Our work also involved combining features of different types. In addition to visual features as in [6, 26, 24], we also added semantic features.

Chen et al. computed similar features from a data set of Chinese food images that they collected [8]. Instead of using SIFT features with a dictionary of words learned through K-means clustering, the dictionary was constructed using sparse coding. The idea behind sparse coding is that an image can be represented by a linear combination of a few "words"/features from the dictionary. Therefore, each image can be represented in a more compact manner. Each feature was used to train a separate multi-class SVM classifier and combined using the Adaboost algorithm as opposed to MKL. Their classifier was able to achieve a performance of 68.3% accuracy for 50 food categories. This is one of the data sets we used in our experiments because of its variety and availability. On an individual food category basis, however, there are some items that still have low recall, like arepas and croissants [8], most likely because they are confused with other foods. This observation further emphasizes the need for a more fine-grained approach to discriminate between these classes as we will discuss in Section 2.2.

Another challenge of food recognition is that there is typically more than one food item associated with an image. Kong et al. addressed this issue by building a model using multiple

images of a food scene [29]. For each image of the same instance, interest points were detected and features were extracted and classified. The interest points from the images were matched using epipolar geometry between the images to determine their geometric location. As such, food items were better segmented by using the classes of the interest points and their locations. For example if two interest points belong to the same class but are located far apart, it is most likely that they belong to different food items. Finally, the existence of a food item was modelled by the joint probability of that food item across the images. While this method proved to perform better than baseline methods for identifying multiple food items, it required multiple images which is not only burdensome for the user but also requires more training time.

## 2.2 Fine-Grained Recognition

It has been acknowledged that human recognition is hierarchical in nature [42]. Objects higher in the hierarchy represent more general/superordinate classes (e.g., food, animals, plants, etc.), followed by basic level classes (e.g., apple, bird, flower, etc.), and finally, subordinate level classes (e.g., Granny Smith apple, Royal Gala apple, sparrow, blue jay, etc.) [42]. Superordinate classes are distinguished by their function whereas basic level classes are distinguished by their structure. Subordinate classes are in turn, distinguished by their part attributes, e.g., Granny Smith apples have green skin while Royal Gala apples have red skin.

In humans, the basic level classes are learned before the subordinate classes, which is what the works on food recognition have achieved so far. However, these traditional object recognition methods are too general to distinguish the subtle differences between subordinate classes, such as arepas and croissants mentioned in [8] or visually similar foods in Figure 1.2. For this more sophisticated level of distinction, fine-grained recognition methods are required. In the following two sections, we discuss some of the approaches of fine-grained recognition and how it is applicable to food recognition.

Bar-Hillel and Weinshall were among the first to acknowledge that human recognition is hierarchical and believe computers should follow the same strategy. They proposed a part-based, generative model for each basic level class, where objects were modelled as a set of relative parts [23]. Each part was represented by an image feature that captured its appearance, scale, and location relative to a reference point. The descriptor for each part was then concatenated to form a fixed, ordered feature vector. Therefore, there is a class-specific feature vector for each basic level class. The advantage of this approach is that only models for basic levels need to be learned, which generalizes well when new subordinate classes are introduced. However, training a generative model is computationally intensive, especially when there are many basic level classes. This framework also implies that basic classes are independent from each other, when there can be useful relationships between these classes. In our work, we also represented food items by "part" attributes but we used a feature vector that generalizes to all basic classes.

Another challenge of fine-grained recognition is finding the distinguishing feature between two subordinate classes. One approach to finding these features is by random sampling. For

example, Yao et al. used a dense sampling approach where they sampled image patches of varying sizes to extract visual features [51]. However, this approach resulted in a large number of redundant features as many of the patches overlapped. In addition, some of the patches may not contain discriminative features for classification. The authors addressed these two problems by employing a random forest framework where each node of a tree is an SVM classifier trained on a sampled image patch or pair of patches. The tree was grown by selecting the best image patch used to split the node. Consequently, only a subset of features were used and the most discriminative patches were kept. Furthermore, each tree was grown using a subset of training images to minimize the correlation between them.

The previous method relied on a dictionary of visual words to extract the features from each image patch. Using a dictionary can result in loss of information that is needed for fine-grained discrimination [50]. Yao et al. avoided this issue by proposing an image representation that is codebook-free [50]. In their approach, they randomly generated templates from training images. Next, a template matching algorithm was used to match regions of a test image with these templates to create a response map. The response maps were then aggregated into a feature vector using max-pooling. In this way, the continuous template matching scores captures the subtle details of an image that is sometimes lost with a quantized, codebook-based representation. However, the same issue of redundancy and non-discriminative features arose as the templates frequently overlapped. To solve this problem, the authors trained several classifiers on a random subset of feature dimensions and combined them using a bagging algorithm to avoid overfitting.

A weakness of these two methods is that a lot of unnecessary templates were generated, leading to non-discriminative features and complicated classifiers to resolve these features. Deng et al. followed this work with a similar approach of template matching by involving a human-in-the-loop to help locate the discriminative regions [13]. In their method, they designed a game whereby users were asked to classify a blurred image of a bird. Users could reveal small parts of the image to help with classification by clicking on a region, but were penalized as more regions were revealed. Their system made note of all the clicked regions that helped the user to correctly classify the bird and used these as the discriminating templates. Another approach by Yang et al. learned a template model that specified the discriminating object parts and their spatial relations [48]. The spatial template model was then fitted to a test image before feature extraction. In doing so, templates were not naïvely sampled. Figure 2.2 illustrates an overview of this template alignment approach.

While these template-based approaches have been shown to work well for classifying birds, dogs, and humans, these objects share a consistent structure with distinctive parts that make template-matching a feasible option. The same does not apply for foods however; a pizza does not have the same structure as a hamburger for instance. To apply a template matching approach for food, a specific set of templates would be required for each "family" of foods. Consequently, we chose to use an alternative fine-grained approach for the context of food.

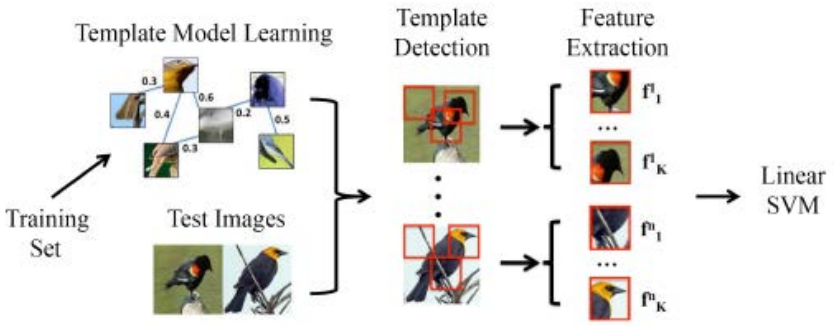



Figure 2.2. Unsupervised template learning for fine-grained object recognition: A spatial template model is learned from training images and is fitted to a test image before feature extraction. [48]

## 2.3 Attribute-Based Classification

The previous methods were all example-based learning methods where a classifier was trained using many instances of a class. Another strategy for fine-grained recognition that is better suited for food is attribute-based classification. If the right attributes are used, attribute-based classification can be superior to example-based classification since attributes are more meaningful than low-level features [17, 30, 40]. Since foods are typically described by their ingredients, cooking technique, and nutritional content, these qualities serve as good attributes to enhance our classification strategy. In this section, we describe the literature on attribute learning and classification.

Ferrari and Zisserman suggested that object recognition can be improved if computers can learn to describe the objects in a high-level manner like humans [19]. They proposed a probabilistic generative model to learn attributes like colours, shapes, and patterns, e.g., stripes. Their model was learnt by maximizing the likelihood over image segments. The presence of an attribute at test time was inferred by computing this likelihood. The attributes in their model were general mid-level attributes that can be obtained using simpler low-level features like colour histograms and shape context descriptors. In contrast, the attributes we used are high-level features that were designed for food recognition specifically.

Our work is more closely related to that of Lampert et al. [30] and Farhadi et al. [17]. In both works and ours, individual attribute classifiers were learned for each attribute $a$. In [30], they used a probabilistic classifier to obtain the posterior probability of attributes $p(a|x)$ and used these for class-level classification. In comparison, Farhadi et al. used a logistic regression model but they also included a feature selection step when training their attribute classifiers to avoid learning a correlated attribute [17]. In our work, we experimented with both SVM and random forests classifiers. SVM classifiers are known to have high performance but random forests have an inherent feature selection step that can mitigate correlation amongst features. Our work also differed in that we discovered the attributes without the need for domain experts

whereas the attributes in both of these works were pre-defined.

Other works that discovered attributes automatically include Parikh and Grauman [40] and Duan et al. [14]. In [40], the authors constructed a set of attributes that are discriminative and interpretable by first finding features that separated the data well. Then by visualizing the feature, they asked a human-in-the-loop to suggest a name for it. This procedure was done iteratively so that each new attribute increased the overall predictive performance. Duan et al. used a similar approach but they discovered their attributes for local regions as opposed to whole images [14]. In addition, they used a recommender system to select the most meaningful attribute to present to the human annotator.

Both methods required a human-in-the-loop to label the attribute which takes time and can be inconsistent. Our method of discovering attributes made use of easily available web content that ensured consistency, much like the work of Berg et al. [1]. In their work, they have a collection of unlabelled images and associated text descriptions mined from an online shopping website. The descriptions contained possible attributes which were ranked by their "visualness" and "localizability", i.e., were the attributes visible and could they be attributed to a particular region of the object? A limitation of their method was that the text descriptions contained superfluous terms that were not useful for recognition (cf. Fig 2.3). In comparison, our attributes come from recipes which are strict descriptions of the foods. Furthermore, our attributes go beyond visual attributes as we also incorporated information from the recipe directions and nutritional information to infer cooking technique and nutrient content.


Figure 2.3. Automatic attribute discovery from noisy web data: images along with a description of the object mined from an online shopping site. The text contains phrases that can be used as attributes but also contains superfluous terms that are not helpful for characterizing the object[1].

# Chapter 3

# Data Sets and Methodology

In this chapter, we describe the data sets used to evaluate the various image descriptors presented in Chapters 4 and 5 and mention the difficulties of each data set. We also outline the evaluation methodology used throughout our work.

## 3.1 Data Sets

### 50Food

The first data set was collected by Chen et al. [8] and is available to download from their webpage.[1] It contains 50 food classes with 100 images for each class. The images were collected from the Internet with most classes consisting of Chinese foods. Figure 3.5 presents an example each food class for reference. Overall, the images are fairly consistent within each class. The food item is centred in the picture and is not occluded by other objects. However, there are some classes like donuts where the number of instances of the item varies, which may pose a challenge because the scale of the object is not consistent (cf. Figure 3.1a,b). Another challenge is that some classes are composed of other classes. For example, rice chicken is made of rice and chicken, so it would be interesting how the classifiers handle these examples. Furthermore, there are some classes that are highly similar even though they are made of different ingredients, such as mapotofu and gongbao chicken (cf. Figure 3.1c,d).

(a) Single instance of a donut (b) Multiple donuts (c) Gongbao Chicken (d) MapoTofu

Figure 3.1. Difficulties of the 50Food data set: the scale and number of instances is inconsistent within a food class (a-b). In other cases, different foods are very similar in appearance (c-d).

[1] http://www.cmlab.csie.ntu.edu.tw/project/food/

### ImageNet

The second data set is a subset of the ImageNet image database [12], built to mirror the hierarchical structure of the lexical graph WordNet [18]. For each concept (*synset*) in WordNet, ImageNet contains approximately 500-1000 images. The images were collected from the Internet using various image search engines and verified by Turkers to ensure they illustrated the particular synset. Of the 230 leaf node food classes in ImageNet, we randomly selected 115 classes and downloaded the images for each class from the ImageNet database. Images less than 20Kb in size were removed and the remainder were randomly sampled to get 100 images per class. Figure 3.6 shows the 115 classes in the data set. We also created a subset of the ImageNet food data set by randomly selecting 10 classes to evaluate the baseline methods on a small scale and to also evaluate food recognition by humans (cf. Section 4.1). These 10 classes are identified in Figure 3.6.


(a) lasagne (b) lasagne (c) lasagne (d) lasagne

(e) kebab (f) fruit drink (g) fruit salad (h) milk

Figure 3.2. Difficulties of the ImageNet food data set: Images are noisy, e.g., (a) is a close-up of lasagne, (b) is an unfinished lasagne dish, (c) is a lasagne mix, (d) is a pizza labelled as lasagne. Many images are not properly centered as in (e)-(h).

These data sets were chosen because they are reflective of everyday images that users would take. While both are challenging, the ImageNet data set is a more difficult data set than the 50 Food data set. Not only are there more classes but the images sometimes contain objects other than the food of interest or are mistakenly labelled (cf. Figure 3.2d). In some cases, the food item represents a small percentage of the image (cf. Figure 3.2e-g). With more classes, there are also frequent visual overlap between classes, such as pottage and bisque, and couscous and tabbouleh; classes that differ by a few ingredients or just by cooking technique (cf. Figure 3.3).

(a) pottage (b) bisque (c) couscous (d) tabbouleh

Figure 3.3. Visually similar food classes of ImageNet: (a) pottage vs. (b) bisque and (c) couscous vs. (d) tabbouleh

### Preprocessing of Images

To ensure that all images are of comparable size and to keep the feature extraction consistent, all images were rescaled so that the smallest dimension is 200 pixels. Other than rescaling the images, we did not perform any other preprocessing techniques such as cropping the image for the region of interest. Table 3.1 displays a summary of the data sets used in this thesis.

Table 3.1. Data Set Statistics

| Data Set | No. Classes | Images per Class | Total No. of Images |
|---|---|---|---|
| 50 Food | 50 | 100 | 5000 |
| ImageNet 10 | 10 | 100 | 1000 |
| ImageNet 115 | 115 | 100 | 11 500 |

## 3.2 Evaluation Methodology

### Classifier

As mentioned in the scope and goals section of this thesis (cf. Section 1.3), our focus is on innovating the feature extraction process as opposed to the classifier. For this reason, we chose to use support vector machines (SVM) as our main classifier for its ease of implementation and training.

We provide a brief formulation of SVM binary classification for easy reference. For further detail regarding SVMs, please see [9]. We have a set of $N$ images $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_n \in \mathbb{R}^d$ is the feature vector for image $n$ and $y_n \in \{-1, 1\}$ is the corresponding class label. The solution to the SVM finds the best hyperplane $H_{\mathbf{w},b}$ that separates the negative and positive examples, i.e.,

$$H_{\mathbf{w},b} = \{x \in \mathbb{R}^d : y_n(\mathbf{w}^T\mathbf{x}_n + b) > 0, n = 1, \ldots, N\},$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. The best hyperplane is the one that has the maximum margin $\rho(\mathbf{w}, b)$, defined as the distance of the closest example $\mathbf{x}_n$ to $H_{\mathbf{w},b}$, i.e.,

$$\rho_{\mathcal{S}}(\mathbf{w}, b) = \min_{n=1}^{N} \left\{ \frac{|\mathbf{w}^T\mathbf{x}_n + b|}{||\mathbf{w}||} \right\}.$$

Figure 3.4 illustrates this concept.

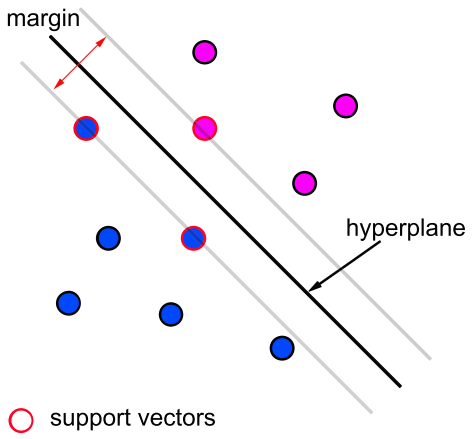



Figure 3.4. The support vector machine solution finds the separating hyperplane with the widest margin. Support vectors are data points that lie on the margin.

To maximize $\rho_{\mathcal{S}}(\mathbf{w}, b)$, the SVM solves the primal optimization problem

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + R\sum_{n=1}^{N}\xi_n$$

$$\text{subject to } y_n(\mathbf{w}^T\mathbf{x_n} + b) \geq 1 - \xi_n$$

$$\xi_n \geq 0, n = 1, \ldots, N$$

where $R$ is a regularization parameter and $\xi_n$ are slack variables that provide a trade-off between misclassification error and the complexity of the model.

When $N << d$, it is more efficient to solve the dual problem:

$$\max_{\alpha} -\frac{1}{2}\alpha^T K\alpha + \sum_{n=1}^{N}\alpha_n$$

$$\text{subject to } \sum_{n=1}^{N} y_n\alpha_n = 0$$

$$0 \leq \alpha_n \leq R, n = 1, \ldots, N$$

where $K$ is the Gram matrix of inner products, $K_{ij} := \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. In the dual formulation, it is possible to replace the entries of $K$ with a kernel function $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$. A popular kernel that works well with non-linearly separable data is the radial basis function (RBF):

$$k(\mathbf{x}, \mathbf{x}') := \exp(-\gamma|\mathbf{x} - \mathbf{x}'|^2).$$

The parameter $\gamma$ controls how close an example $\mathbf{x}'$ has to be to a training example $\mathbf{x}$ for the value of $k(\mathbf{x}, \mathbf{x}')$ to be high; the larger $\gamma$ is, the closer $\mathbf{x}'$ needs to be to $\mathbf{x}$ to be affected.

The classification of a new image $\mathbf{x}$ is

$$h(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} y_n\alpha_n K(\mathbf{x_n}, \mathbf{x}) + b\right).$$

For multi-class classification with $C$ classes, one approach is to train $C$ one-vs-rest classifiers. Another approach is to train $\binom{C}{2}$ one-vs-one classifiers. Since it has been noted that a one-vs-one strategy works better than one-vs-rest [11], we adopted the latter strategy. We implemented the multi-class SVM classifier using the SVM module from the `scikit-learn` machine learning library[41].

## Classifier Training and Evaluation

To train the SVM classifiers, we conducted preliminary experiments to find the optimal kernel and parameters $R$ and $\gamma$ with respect to each data set and image descriptor. We created a training set consisting of 80 images from each class and reserved the remaining 20 images for testing. Through five-fold cross-validation on the training set, we found the best kernel (linear/RBF), $R \in \{0.001, 0.01, 1, 10, 100\}$, and $\gamma \in \{0.01, 0.001, 0.0001\}$ for each data set and descriptor. For all subsequent experiments using the same data set and descriptor, we used these optimal parameters to train the classifier.

The tuned classifiers were used to predict the classes of the testing set and evaluated based on accuracy $acc$, precision $p$, recall $r$, and $F1$ score. Letting $TP_c$, $FN_c$, $FP_c$ denote the number of true positives, false negatives, and false positives for class $c$, the accuracy and $F1$ score for class $c$ are defined as

$$acc = \frac{\sum_{c=1}^{C} TP_c}{|\mathcal{S}|}$$

$$F1(c) = 2\frac{p(c)r(c)}{p(c) + r(c)} \qquad c = 1, \ldots, C$$

where

$$p(c) = \frac{TP_c}{TP_c + FP_c}$$

and

$$r(c) = \frac{TP_c}{TP_c + FN_c}$$

are precision and recall for class $c$.

The results were averaged over five random training/testing splits of the data (80% training/20% testing) where the classifier was trained using the optimal parameters learned from our preliminary experiments.

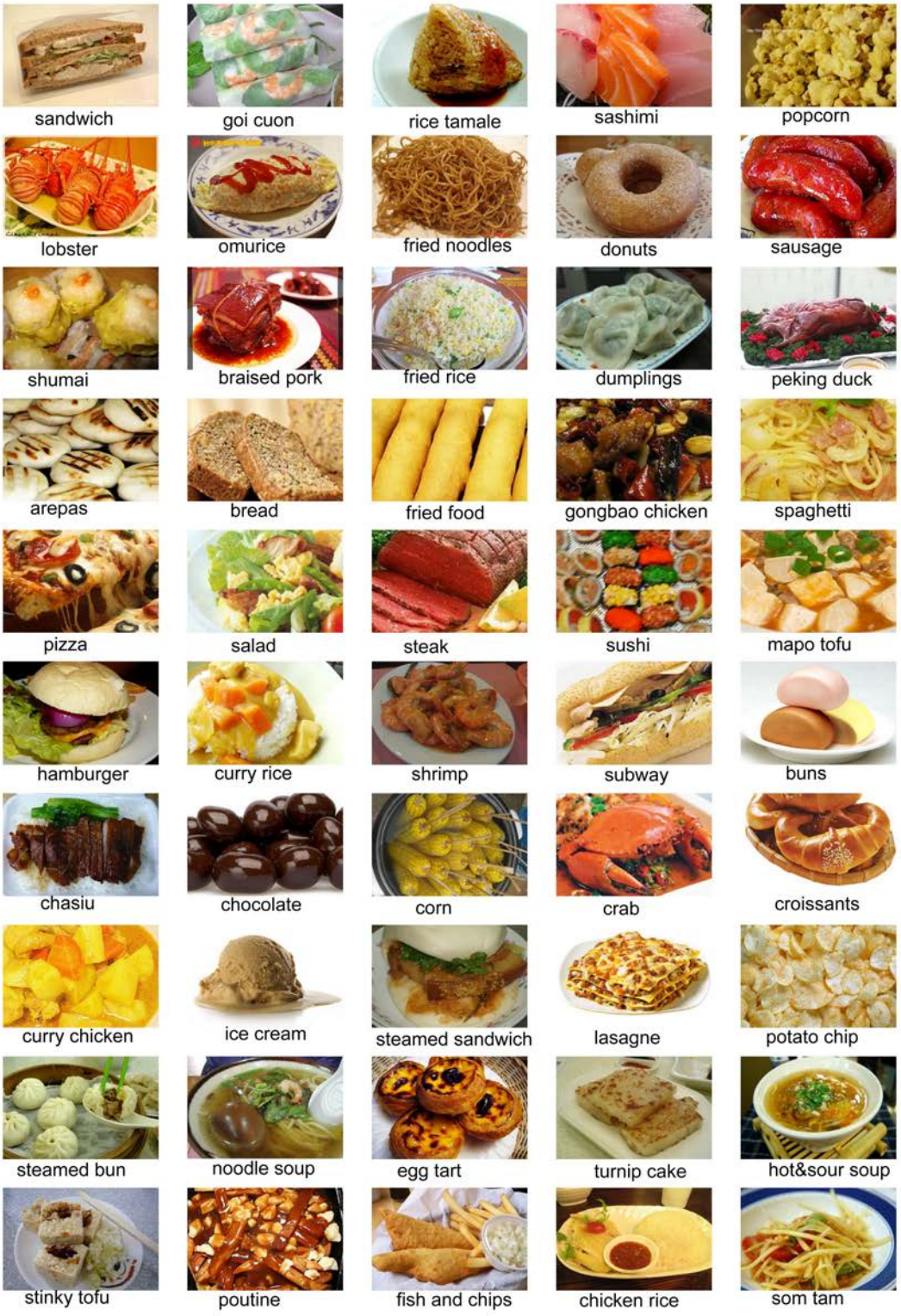



Figure 3.5. 50 Food Data Set [8]

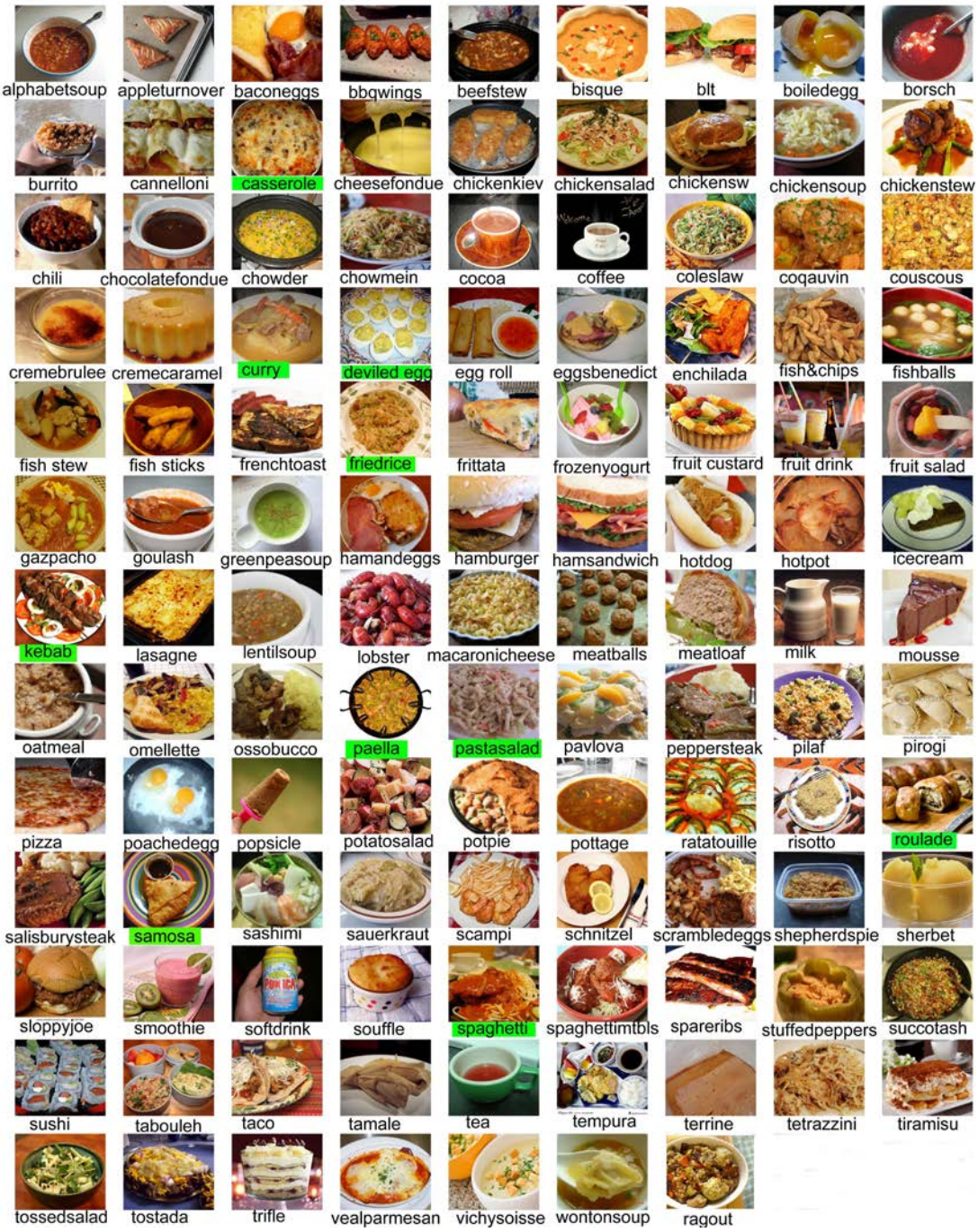



Figure 3.6. ImageNet115 Food Data Set[12]

(Classes of ImageNet10 are highlighted in green)

## Chapter 4

# Baseline Descriptors

As mentioned in Chapter 2, there are many techniques available for object recognition. This chapter describes the baseline methods we evaluated on our food images and highlights their strengths and weaknesses. Through experimental evaluation, we determined the best baseline for food recognition.

## 4.1 Human Classification

It has been studied that humans are excellent at remembering thousands of objects, recognizing at least 30,000 classes [4]. As a preliminary study, we were curious to know how well humans can recognizing foods. Using the ImageNet10 data set, we created an online questionnaire that was administered through Google Docs. Forty Facebook friends of the author volunteered to take part in the study. In the first part of the questionnaire, the participant was presented with 10 randomly selected images of each food class. After having studied the images, the participant was presented with a set of new images, one randomly selected image from each class and asked to name the food in the photo.

To keep the questionnaire simple for the participant and to ensure that answers stayed within the set of food classes, we chose to use a multiple choice format for the quiz. In addition to the correct answer, the other three choices were selected at random from the remaining nine classes. This protocol was used to avoid making the quiz too easy or difficult. A sample of the quiz is included in Appendix A for reference.

The results of our study showed that humans have an accuracy of 95.6% at recognizing the 10 foods in this set. We realize that the quiz was not a comprehensive test of human food recognition. For one, only one test image was presented for each class to keep the quiz less onerous for the participant. Second, using a multiple choice format did simplify the task by narrowing the options to four possible classes. Nevertheless, the results give an approximate upper bound that recognition systems can strive toward. The study also revealed that while humans are good at recognizing most foods, some foods are more difficult to recognize than others, as seen in the $F1$ scores in Table 4.1. In the study, paella was the most challenging food to classify. It was often confused with fried rice and curry, two dishes that are similar to paella

in appearance.

Table 4.1. Human Food Classification Scores (scores are averaged over 40 participant responses)

| Class | Precision | Recall | F1 Score | Class | Precision | Recall | F1 Score |
|---|---|---|---|---|---|---|---|
| Roulade | 1.00 | 1.00 | 1.00 | Samosa | 1.00 | 0.97 | 0.99 |
| Spaghetti | 1.00 | 1.00 | 1.00 | Fried Rice | 1.00 | 0.92 | 0.96 |
| Kebab | 1.00 | 1.00 | 1.00 | Casserole | 0.85 | 1.00 | 0.92 |
| Deviled Egg | 1.00 | 1.00 | 1.00 | Curry | 0.90 | 0.95 | 0.93 |
| Pasta Salad | 0.98 | 1.00 | 0.99 | Paella | 0.85 | 0.72 | 0.78 |

## 4.2 Bag of Features

Our first computer baseline is the popular *Bag of Features* (BoF) descriptor, which represents the distribution of visual features of an image. The intuition is that images of the same class will have similar feature distributions, descriptors that can be used to train a classifier. The BoF descriptor starts by extracting low-level features from the images and uses them to construct a dictionary of visual "words" via a clustering algorithm. Once the dictionary is created, the distribution of visual features can be computed for each image to obtain the BoF descriptor. Figure 4.1 illustrates an overview of this pipeline.

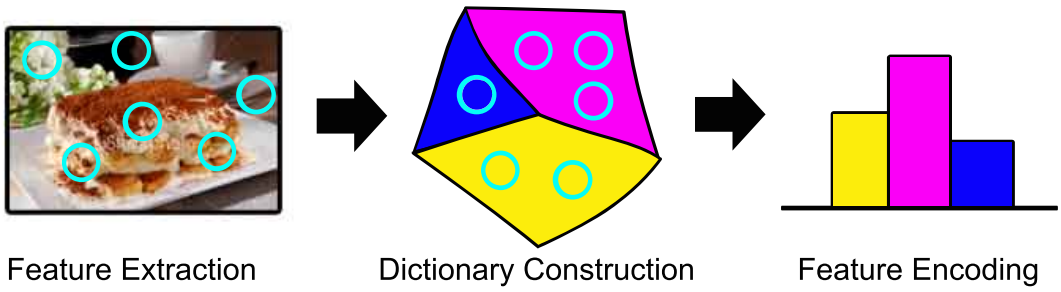
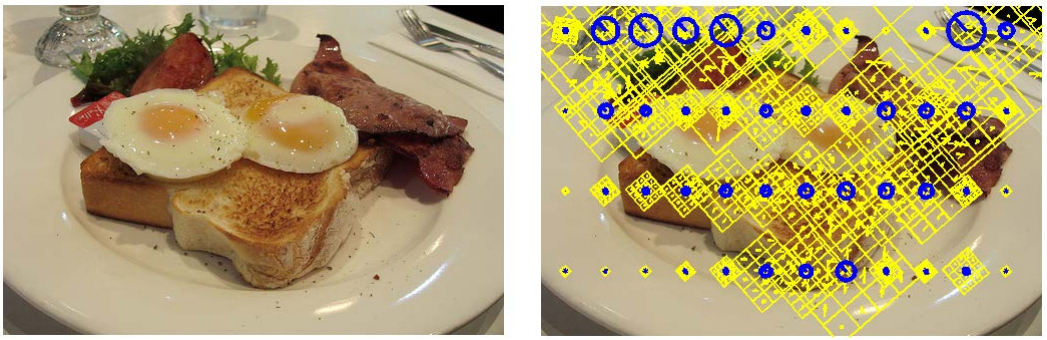



Figure 4.1. Overview of Bag of Features pipeline: visual features are extracted from the image, which are discretized using a clustering algorithm to determine the feature distribution. The distribution is used as input for a classifier like SVM.

Here, we used $K$-means as the clustering method. The features used in the clustering algorithm were from 10 randomly sampled images for each class. The choice of $K$ was chosen among $\{100, 200, 300, 400, 500, 600, 1000, 2000, 3000, 4000, 5000, 6000\}$ to maximize the cross-validation score during the training stage. We chose to implement this descriptor as our first baseline because it has been shown to perform reasonably well on other food data sets with relatively little effort [6, 7, 8, 24, 47]. In addition, it provides a delineating lower bound of food recognition performance using a basic descriptor.

Another advantage of the BoF descriptor is that it is flexible in terms of the choice of visual features used. Since food is typically described by its colour, shape, and texture, we extracted the following features to capture these visual cues: colour scale invariant feature transform (cSIFT) descriptors, colour histograms, histogram of oriented gradients (HOG), and texture information.

**cSIFT:** SIFT descriptors have been shown to perform well on recognition tasks as they are invariant to contrast, intensity and small transformations [31]. They summarize the gradient information of a 16×16 pixel region around a key point by computing the gradient orientation and amplitude. The 16×16 patch was divided into 4×4 blocks where an 8-bin histogram of gradient orientation was computed for each block, with bin entries weighted by the gradient amplitude. The 16 histograms were concatenated and normalized to form a 128-dimensional vector. For colour SIFT, the same procedure was done for each colour channel resulting in a 384-dimensional feature vector. In our work, SIFT descriptors were computed at points on a regular grid spaced 40 pixels using the `vlfeat` toolbox [45]. Figure 4.2 shows the SIFT descriptors (yellow) computed at each key point (blue) for a typical image. A weakness of using a regular grid to compute SIFT descriptors was that descriptors for non-food pixels were also included.



(a) Original food image (b) SIFT key points (blue) and descriptors (yellow)

Figure 4.2. SIFT desciptors were computed at points on a regular grid 40 pixels apart. The size of the blue circles corresponds to the scale at which the descriptor was computed.

**Colour Histograms:** Colour plays an important role in visual recognition and for food recognition especially as foods are easily distinguished by colour alone [7]. To capture colour information, colour histograms were computed for an image by discretizing pixel intensities into $n$ bins. This step was done for each RGB channel to obtain a $3\times n$-dimensional feature vector. Local colour information was also obtained by dividing the image into a 4×4 grid and computing colour histograms for each cell. The resulting feature vector was $17\times3\times n$-dimensional. The number of bins $n \in \{16, 32, 64\}$ was decided through cross-validation.

**HOG:** Another important quality of food is its shape. To this end, we extracted shape information by first applying a Canny edge detector to get a binarized image of edges. HOG

descriptors were then computed for each 8×8 pixel cell as described in [10]. The orientation was discretized into nine bins where each bin entry is the number of edges that fall within that orientation range. Then for each 2×2 block of cells, the four histograms were normalized and concatenated to form a block descriptor. Finally, all block descriptors were concatenated to obtain the final HOG descriptor. As with the SIFT descriptors, we also used the `vlfeat` toolbox to compute these descriptors [45].

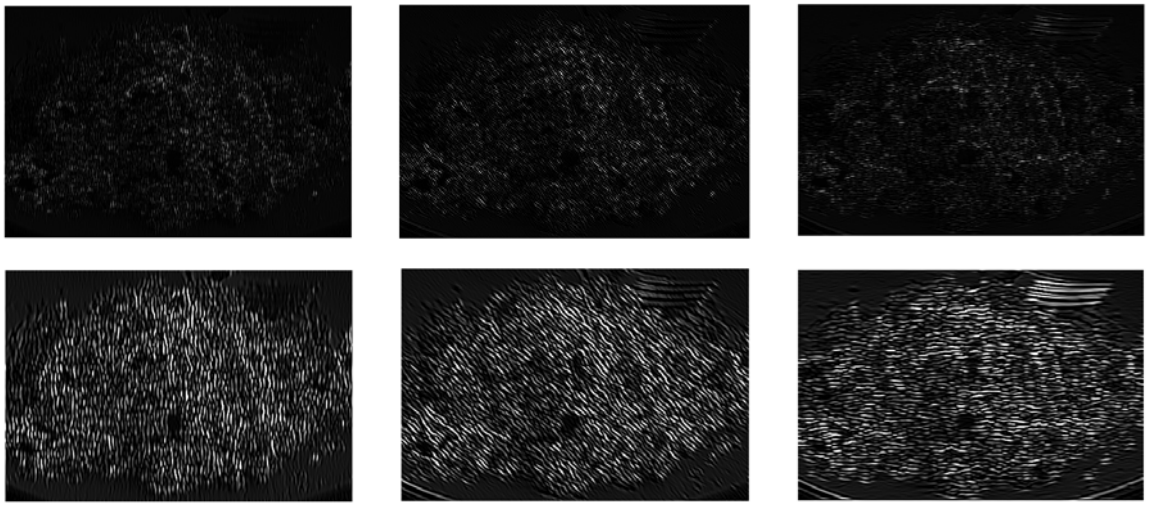
(a) Original food image (b) Binarized edge image (c) HOG features

Figure 4.3. Shape information was captured by applying a Canny edge detector to image (a) to isolate edges (b). HOG descriptors were then computed using the binarized edge image (c).

**Texture:** Finally, texture information was captured by filtering each image with a bank of Gabor filters to accentuate the texture of the food. Here, we used Gabor filters with three different scales $\{3, 5, 7\}$ and five orientations $\{0°, 45°, 90°, 135°, 180°\}$. Figure 4.4 shows the effect of varying the scale and orientation of each Gabor filter. Increasing scale captured coarser texture information while varying orientation highlighted textures at different angles. For each scale and orientation the texture was summarized by computing the entropy $E = -\mathbf{p}^T \log_2(\mathbf{p})$, where $\mathbf{p}$ is a 256-bin histogram of pixel intensities of the filtered image. Local texture was also computed by dividing the image into 4×4 grid and computing the entropy for each cell. The final texture feature vector was $No.Scales \times No.Orientations \times 17$ dimensions long.

The final BoF descriptor was formed by concatenating the cSIFT descriptor, colour histogram, HOG descriptor, and texture descriptor. Table 4.2 summarizes the dimension of the individual features for each data set. Figure 4.5 shows the performance of each feature on its own and for the full BoF descriptor. Here we see that colour histograms were the most discriminative of the individual features for food recognition, only underperforming the full BoF descriptor by 3.5% at the most, while texture was the least discriminative. We also see that the ImageNet115 data set was the most challenging, achieving the highest performance of 6.1% with just colour histograms, which was approximately five times less than that of the 50Food and ImageNet10 data sets. Nevertheless, with 115 classes, this result is still better than chance. Another observation is that the classification accuracy is not additive by combining features as they are likely correlated.

(a) Original food image



(b) Gabor filtered images

Figure 4.4. Gabor filters were applied to food images to extract texture. Increasing the scale captured coarser texture (top row vs. bottom row) while varying the orientation captured texture elements at different angles (left to right).

## 4.3 PiCoDes

Our second baseline is the Picture Codes (PiCoDes) descriptor, a compact descriptor developed by Bergamo et al. for large scale object recognition [3]. It is motivated by the problem of novel class recognition where a typical task is to retrieve images of an unseen class. At query time, the user supplies the system with a small collection of example images and the classifier would therefore have to quickly learn on this small training set. Since this step is done online, the descriptors need to be compact yet discriminative for the images to be retrieved in a timely and accurate fashion.

The PiCoDes descriptor addresses these requirements by encoding the binary output of a set of basis classifiers. That is, for an image $\mathbf{x} \in \mathbb{R}^d$, the PiCoDes descriptor is represented by

$$\mathbf{h}(\mathbf{x}; \mathbf{A}) = [h_1(\mathbf{x}; \mathbf{a}_1), \ldots, h_C(\mathbf{x}; \mathbf{a}_C)]$$

where the basis classifiers $h_c(\mathbf{x}; \mathbf{a}_c) = \mathbb{I}[\mathbf{a}_c^T \psi(\mathbf{x}) > 0] \in \{0, 1\}$ are parametrized by $\mathbf{a}_c$. Hence,

Table 4.2. Optimal BoF Dimensions Determined by 5-Fold Cross-Validation

| **Feature** | **Feature Dimensions** | | |
|---|---|---|---|
| | ImageNet10 | 50Food | ImageNet115 |
| cSIFT ($K$) | 500 | 1000 | 300 |
| Colour ($n$) | 32 | 64 | 32 |
| HOG ($K$) | 100 | 100 | 100 |
| Texture | 255 | 255 | 255 |
| **Full BoF** | **887** | **1419** | **687** |

the descriptor is compact and discriminative due to the implicit incorporation of the $h_c$ basis classifiers.

To learn the parameters $\mathbf{a}_c$, the classifiers were trained on low-level features (pyramid of histogram of oriented gradients (PHOG)[5], SIFT[31], GIST[38], and spatial pyramid of self-similarity descriptors[44]) which are mapped to a higher dimension using a feature map $\psi : \mathbb{R}^d \to \mathbb{R}^{2(dr+1)}$ adopted from [46].[1] The feature map $\psi$ was chosen so that it approximated a non-linear kernel distance, i.e., $\langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle \approx K(\mathbf{x}_i, \mathbf{x}_j)$. The advantage of using $\psi$ instead of $K$ is so a linear classifier can be learned, which is less expensive to train than a non-linear classifier. The parameters $\mathbf{a}_c, c = 1, \ldots, C$ were jointly solved to minimize the misclassification error of the training set.

We chose this descriptor as one of our baselines in part because of its low overhead in terms of computation and storage, which allowed for fast training, but also for its high recognition performance on difficult data sets like Caltech256[22] and ImageNet[12], of which our ImageNet food data sets are a subset. For a PiCoDes descriptor of maximum dimension $C = 2048$, the accuracy on the Caltech256 data set matched the performance of the state-of-the-art non-linear LP-$\beta$ classifier of Gehler et al. [20]. This feature was particularly important for our work because our recipe-based descriptor was developed as an extension to the baseline descriptor and consequently depended on its predictive performance. The design of the PiCoDes descriptor is also similar to our recipe descriptor as both encode the outputs of a set of binary classifiers and should compliment each other well. Another strength of the PiCoDes descriptor is that the features are mid-level visual cues which are more in line with human recognition, as opposed to the low-level features of the BoF descriptor.

To extract PiCoDes descriptors, we used the software available from the authors' website.[2] Based on the authors' reported results, we used a descriptor size of $C = 2048$ for maximum performance. $C = 2048$ is the largest feasible descriptor size because the parameters $\mathbf{a}_c, c = 1, \ldots, C$ are learned jointly through an expensive optimization algorithm. Another limitation is that it is unclear what what the individual basis $h_c$ classifiers are classifying.

[1] $r$ is a tuning parameter set to 1 in the authors' implementation.

[2] `vlg.cs.dartmouth.edu/picodes`


Figure 4.5. Bag of Features classification accuracy. Results are shown for each feature individually and for the full BoF descriptor. Of the individual features, colour histograms were the most discriminative. Note that features are not additive as they are likely correlated.

## 4.4 Meta-Class

Our third baseline, the Meta-Class descriptor[2] is in the same line as PiCoDes, where each dimension of the descriptor is an output of a non-linear classifier. However, instead of classifying arbitrary attributes, the classifiers were trained to predict if an image belongs to a *meta-class*. A *meta-class* is an abstract super-class that covers a set of visually similar classes. They were learned by hierarchically partitioning the confusion matrix derived from the class-level classifiers using spectral clustering [35]. The intuition is that by partitioning the confusion matrix like so, classes that are often confused and hence visually similar are grouped together in a meta-class. The result is a visual class hierarchy where each node in the hierarchy represents a meta-class.

The classifiers themselves are a linear combination of SVM classifiers trained on $m = 1, \ldots, M$ visual features; the same features as in PiCoDes. That is,

$$h_c(\mathbf{x}) = \sum_{m=1}^{M} \beta_{m,c} h_{m,c}$$

where $h_{m,c} = \mathbf{w}_{m,c}^T \psi(\mathbf{x}) + b_{m,c}$ is an SVM that measures how well an image belongs to meta-class $c$ with respect to feature $m$ and $\psi$ is a feature map as in the PiCoDes formulation. The overall classifier $h_c$ combined the results of all $m$ SVM classifiers much like the LP-$\beta$ classifier [20] but

it used a kernel approximation instead of a non-linear kernel[46].

We chose to use Meta-Class descriptor as a baseline method for the same reasons as PiCoDes but also for its implicit embedding of the visual hierarchy. Deng et al. noted that in their analysis of the ImageNet data set that hierarchies were important when working with large scale data, like our ImageNet115 food data set, as there were frequent visual overlaps between classes[11]. A hierarchy helps to reduce the confusion between these visually similar classes which we have already noticed through our human food recognition study (cf. Section 4.1). Marszalek and Schmid[32] and Hwang et al. [25] have shown that incorporating a hierarchy did improve recognition, at least for their non-food data sets. Unlike these works, however, Meta-Class incorporates the hierarchy in the descriptor as opposed to the classifier, allowing for the use of a simple, off-the-shelf classifier.

We used the software available from the authors' website (the same as PiCoDes) to compute Meta-Class descriptors for our food data sets. Since the performance of the continuous-valued and binary-valued version were comparable[2], we chose to use the binarized version of the Meta-Class descriptor. The advantage of the binary version was that it required less memory and was quicker to train a classifier. For both the PiCoDes and Meta-Class descriptors, the individual basis and meta-class classifiers were pre-trained on a subset of images from ImageNet, which may or may not have included the ImageNet food images. As such, the accuracy of any food classifier trained on these descriptors ultimately depends on the training images used to train these basis/meta-class classifiers.

## 4.5 Results and Analysis

In this section, we present an analysis of the performance of each baseline descriptor for the three food data sets.

### Overall Classification Performance

From Figure 4.6, we see that the best descriptor is Meta-Class, which consistently outperformed PiCoDes by ∼5% across all data sets. Since both descriptors were trained on the same low-level visual features, this result is evidence that a hierarchical approach does help to improve recognition. Furthermore, Meta-Class was able to outperform PiCoDes using just a linear kernel whereas PiCoDes required a RBF kernel for optimal performance (cf. Table 4.3). We also note that even with five times the number of classes, both the Meta-Class and PiCoDes descriptors performed better on the 50Food data set than the ImageNet10. A reason for this behaviour is because the ImageNet images are not as clean as the 50Food images, where sometimes the food item is not the main subject in the photo (cf. Figure 3.2).

### Learning Rate

We also studied the learning rate of each descriptor. Figure 4.7 shows the learning curve for each descriptor by varying the number of training examples per class and evaluating the same test set of 20 images per class. Again, we see that the Meta-Class consistently has a faster

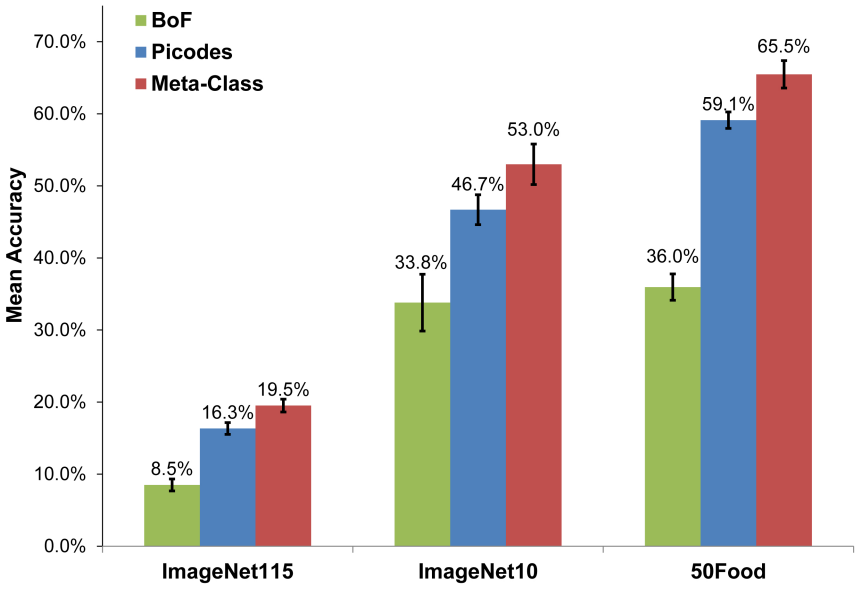
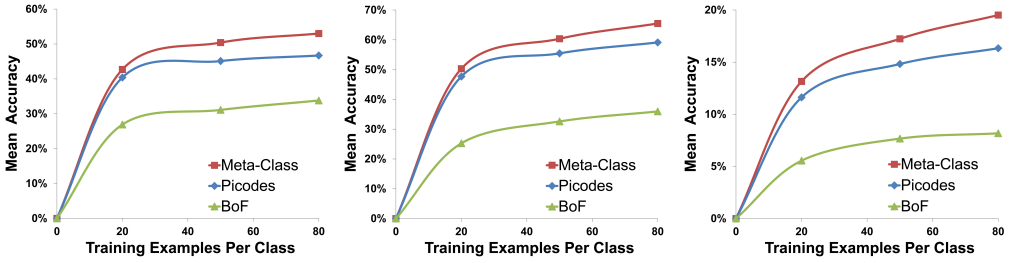



Figure 4.6. Baseline classification performance. Results are averaged over 5 random 80%/20% training/testing splits of the data.

learning rate than PiCoDes and BoF. At 20 training examples, the accuracy of PiCoDes and Meta-Class are comparable but as the number of training examples increases, the difference in accuracy between the two descriptors also increases. Based on the trajectory of the learning curves for the 50Food and ImageNet115 data sets, it is not surprising that with more classes more training examples are needed before the recognition accuracy plateaus.




(a) ImageNet10 (b) 50Food (c) ImageNet115

Figure 4.7. Classificaton Performance vs. Number of Training Images. Results are averaged over 5 random training/testing splits of the data. The classifier was evaluated on a test set with 20 images per class.

## Individual Food Classification

On an individual food class basis, Meta-Class achieved the best F1-score for 5/10 ImageNet10 classes (cf.Table 4.4), 36/50 50Food classes (cf.Appendix B:Table B.1), and 70/115 Ima-

Table 4.3. Optimal SVM Parameters Determined by 5-Fold Cross-Validation
(see Section 3.2 for further details)

| **Descriptor** | **Data Set** | **Kernel** | $R$ | $\gamma$ |
|---|---|---|---|---|
| BoF | ImageNet10 | RBF | 10 | 0.0001 |
| | 50Food | Linear | 1 | N/A |
| | ImageNet115 | Linear | 0.001 | N/A |
| PiCoDes | ImageNet10 | RBF | 10 | 0.001 |
| | 50Food | RBF | 10 | 0.001 |
| | ImageNet115 | RBF | 10 | 0.001 |
| Meta-Class | ImageNet10 | Linear | 1 | N/A |
| | 50Food | Linear | 1 | N/A |
| | ImageNet115 | Linear | 0.01 | N/A |

geNet115 classes (cf.Appendix B:Tables B.2/B.3). Judging from the many zero F1-scores in Tables B.2/B.3, we observed that as the number of classes increase, it becomes much harder to recognize all food classes. Note that the Meta-Class F1-scores for curry and fried rice dropped to 0 for the ImageNet115 data set when it was 0.41 and 0.24 respectively for the ImageNet10 data set, suggesting that the confusion between these similar foods becomes more extreme when the label space increases. It is also interesting that the worst five recognized foods of the ImageNet10 data set by these computer vision systems mirrored the result of our human food recognition study (cf. Table 4.1). A reason for this pattern is because unlike the easily recognized foods like spaghetti and deviled eggs, which are distinctive in colour and/or shape, curry, paella, and fried rice are made of similar ingredients making them hard to differentiate one from the other.

Table 4.4. ImageNet10 F1-Scores by Food Class
(Best results are bolded)

| **Class** | **Meta-Class** | **PiCoDes** | **BoF** | **Class** | **Meta-Class** | **PiCoDes** | **BoF** |
|---|---|---|---|---|---|---|---|
| Spaghetti | **0.75** | 0.67 | 0.29 | Casserole | **0.59** | 0.58 | 0.23 |
| Deviled Egg | 0.62 | **0.71** | 0.51 | Samosa | **0.53** | 0.33 | 0.30 |
| Roulade | **0.67** | 0.36 | 0.52 | Curry | **0.41** | 0.39 | 0.29 |
| Kebab | 0.44 | 0.34 | **0.65** | Paella | 0.38 | **0.39** | 0.33 |
| Pasta Salad | 0.47 | **0.63** | 0.31 | Fried Rice | 0.24 | **0.30** | 0.20 |

Figure 4.8 shows examples of the misclassified images made by PiCoDes, the best descriptor for two of these three most difficult to recognize food classes. The mistakes made by the classifier are understandable as these foods look very similar, especially the fried rice and curry images. To the untrained eye, one could confuse one image for the other. The paella image also looks

like curry because it contains a lot of liquid, hence the misclassification. It is also possible that the true labels are incorrect as we have noted earlier (cf. Figure 3.2).

(a) Image of fried rice confused for paella

(b) Image of paella confused for curry

(c) Image of curry confused for fried rice

Figure 4.8. Top three hardest to recognize foods of ImageNet10 across all baselines. These are mistakes made by PiCoDes, despite having the best F1 scores for these foods.

Based on these results, we have established that Meta-Class is the best descriptor for all food data sets, achieving the highest overall accuracy and on an individual food class basis as well. However, there are some food classes that Meta-Class was unable to distinguish well. In the next chapter, we propose a solution to address this shortcoming.

## Chapter 5

# Recipe-Based Food Descriptor

To motivate the formulation of our recipe-based food (RBF) descriptor, we examined some of the misclassified foods made by the baseline descriptors. Since Meta-Class had the best performance out of the three baselines, we only show mistakes made by this descriptor. Tables 5.1 and 5.2 shows the top five most confusing foods for the 50Food and ImageNet115 data sets (we do not show results for ImageNet10 since it is a subset of ImageNet115). The tables show images mistaken for the label of the image in the second column and vice versa. Comparing the images in the two columns, we see that the pairs of confused images are similar in appearance. Nevertheless, they differ by two important aspects: the ingredients they contain and the way they are prepared, aspects that are not easily captured by a visual descriptor. This insight inspired us to design a fine-grained descriptor that is able to capture these minute but discriminating details.

As discussed in Sections 2.2 and 2.3, two common approaches for fine-grained recognition are template-based or attribute-based. In previous works, template-based approaches have worked well for distinguishing images of different bird species and people performing actions [13, 51, 50, 48]. Template-based descriptors perform well because these objects have distinctive parts like beaks, tails, arms, legs, etc. that are easily matched with a template. Moreover, these objects typically have only one or two of these parts. In contrast, foods are not made of a few parts *per se*, but are composed of several ingredients, some of which are not visible. Accordingly, our solution is to design an attribute-based fine-grained descriptor.

Unlike other objects, food is unique in that information like the ingredients and method of preparation of a dish are available from recipe websites. Hence, the recipe determines the attributes as opposed to being arbitrarily selected. To define the set of attributes, we used the labels of the training images to query a recipe website for information on each dish and used this information to design our RBF descriptor. We then used the baseline descriptors and augmented them with the RBF descriptor to train a multi-class SVM classifier. Figure 5.1 illustrates a high-level overview of this classification pipeline.

Table 5.1. 50Food Top 5 Most Confusing Foods: The first column has an instance of an image misclassified for the label of the second column and vice versa.

| Food Class 1 | Food Class 2 | No.Mistakes/40 |
|---|---|---|
|  |  | 8 |
| Croissants | Donuts | |
|  |  | 6 |
| Egg Tart | Donut | |
|  |  | 6 |
| Sausage | Fried Food | |
|  |  | 4 |
| Braised Pork | Peking Duck | |
|  |  | 3 |
| Ice Cream | Buns | |

Table 5.2. ImageNet115 Top 5 Most Confusing Foods: The first column has an instance of an image misclassified for the label of the second column and vice versa.

| Food Class 1 | Food Class 2 | No.Mistakes/40 |
|---|---|---|
|  |  | 7 |
| Bisque | Pottage | |
|  |  | 8 |
| Fruit Drink | Smoothie | |
|  |  | 6 |
| BBQ Wings | Spareribs | |
|  |  | 5 |
| Potato Salad | Chicken Salad | |
|  |  | 5 |
| Milk | Vichysoisse | |

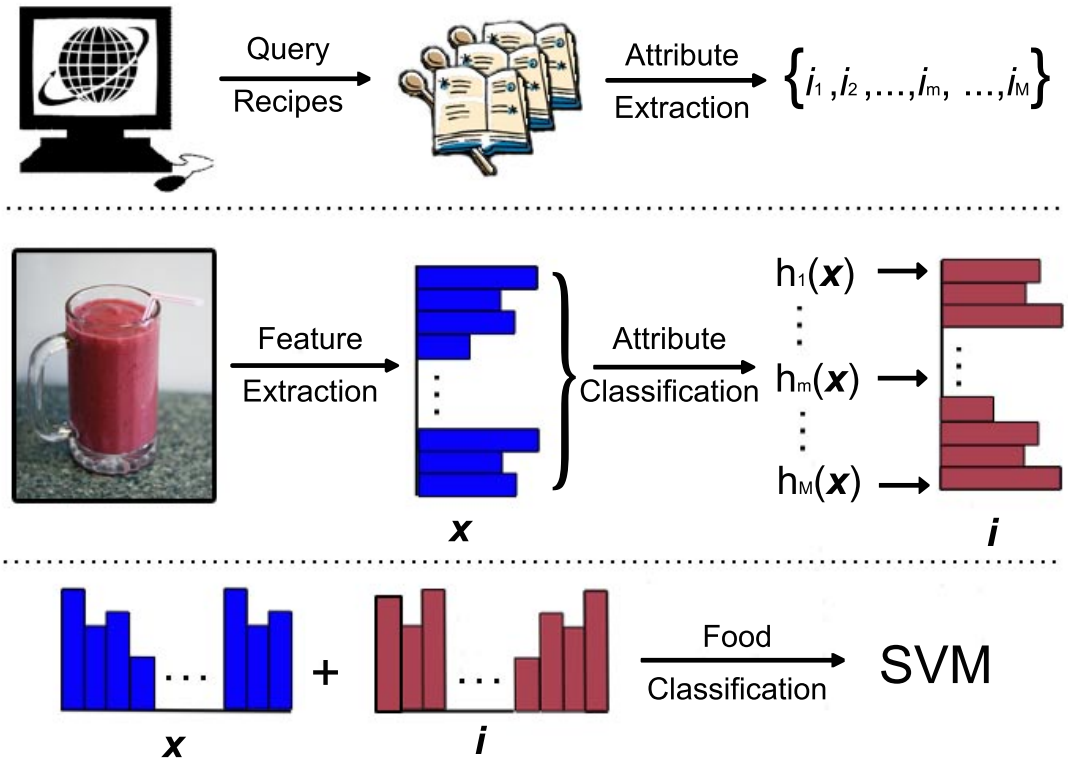



Figure 5.1. Overview of Food Recognition Pipeline. Top Row: Training labels were used to query the Internet for recipes, from which food attributes were extracted. Middle Row: Features were extracted from a test image to form a visual descriptor, which were used as input to a set of trained attribute classifiers. The attribute classifiers computed the recipe-based food (RBF) descriptor. Bottom Row: The visual descriptor and the RBF descriptor were used for food recognition.

## 5.1 Descriptor Definition

More formally, let $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ be the set of $N$ training images where $\mathbf{x}_n \in \mathbb{R}^d$ is the feature vector for image $n$ and $y_n \in \{1, \ldots, C\}$ is the corresponding class label. Each food class is described based on a set of recipe-based attributes $\mathcal{I}(c)$ defined as follows:

**Definition 5.1. Class-Level Recipe Attributes** $\mathcal{I}(c)$

$$\mathcal{I}(c) = \{i_{c,1}, \ldots, i_{c,M}\}$$

where $i_{c,m}$ is the $m$-th attribute as determined by querying the recipe for class $c$. For example, $i_{c,m}$ can be a scalar representing the quantity of an ingredient or a binary value indicating if a particular cooking method is used for class $c$. For simplicity, we let $i_{c,m}$ be a binary measure, i.e., $i_{c,m} \in \{0, 1\}$. Therefore, this representation implies that for all attributes in $\mathcal{I}(c)$, $i_{c,m} = 1 \, \forall m$.

By letting the class-level recipes dictate the set of attributes, as opposed to arbitrary selection, we ensured the attributes are relevant for classification. Hence, the set of attributes adapts to the data set accordingly.

**Definition 5.2. Global Recipe Attributes** It follows that the global set of attributes with respect to the set of images $\mathcal{S}$ is the union of all class-level attributes, i.e.,

$$\mathcal{I} = \cup_{c=1}^{C} \mathcal{I}(c) = \{i_1, \ldots, i_M\}.$$

We choose to drop the class subscript $c$ for this global set of attributes since it is possible that some attributes are shared across several classes. The RBF descriptor for image $n$ is thus

**Definition 5.3. Recipe-Based Food Descriptor (training image)**

$$\mathbf{i}_n = [i_1, \ldots, i_M].$$

For a training image, $\mathbf{i}_n$ is easy to compute because $y$ is given, i.e.,

$$i_m = \begin{cases} 1, & i_m \in \mathcal{I}(y) \\ 0, & \text{otherwise} \end{cases} \quad \forall m = 1, \ldots, M.$$

For a testing image, $y$ is unknown and we have to infer the presence of $i_m$, which leads us to our attribute classification problem definition.

**Problem 5.1.** ***Attribute Classification*** *Given the feature vectors for the set of training images $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and their computed RBF descriptors $\{\mathbf{i}_1, \ldots, \mathbf{i}_N\}$, we wish to learn a set of binary attribute classifiers $\mathcal{H} = \{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}$ to predict the presence of an attribute $i_m$ for a test image $\mathbf{x}'$.*

*Remark.* If $i_m \in \mathbb{R}$, then Problem 5.1 becomes a regression problem.

The intuition is that the visual features $\mathbf{x}$ correlate with an attribute $i$, a semantic attribute that can be visualized. Therefore, it is appropriate to use the already computed descriptors $\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ to train $h_m(\mathbf{x})$. Having learned the set of attribute classifiers $\mathcal{H}$, the RBF descriptor for a test image $\mathbf{x}'$ is:

**Definition 5.4. Recipe-Based Food Descriptor (test image)**

$$\mathbf{i}_n = [h_1(\mathbf{x}'), \ldots, h_M(\mathbf{x}')].$$

The RBF descriptor $\mathbf{i}$ is then concatenated with the visual feature descriptor $\mathbf{x}$ to train the multi-class food classifier.

## 5.2 Implementation Details

In this section, we describe the process of constructing the set of recipe attributes $\mathcal{I}$ and training the attribute classifiers $\mathcal{H}$ to compute the RBF descriptors, and how these are used to train the multi-class food classifiers.

### 5.2.1 Constructing the Recipe Attributes $\mathcal{I}$

We start by constructing the individual class-level recipe attributes $\mathcal{I}(c)$. These are created by using the class labels $\mathcal{C} = \{1, \ldots, C\}$ to query a website for recipes related to $c$. For each class $c$, we retrieved the attributes from the top 10 most relevant recipes and added it to $\mathcal{I}(c)$, ignoring duplicate attributes. To construct $\mathcal{I}$, we simply took the union of all $\mathcal{I}(c)$ as in Definition 5.2.

More specifically, we used *AllRecipes.com* because of its large and diverse collection of recipes and for its easily parsable HTML structure. Figure 5.2 shows a screenshot of the structure of a typical recipe from *AllRecipes.com*. Each recipe is divided into three main sections: ingredients, directions, and nutritional information. Since these sections provide different "views" of a food, we created three different sets of recipe attributes: $\mathcal{I}_{ing}, \mathcal{I}_{dir}, \mathcal{I}_{nutr}$.

For $\mathcal{I}_{ing}$, the attributes $i$ were simply the ingredients in the ingredients list. For $\mathcal{I}_{dir}$, we parsed the directions of a recipe searching for key cooking terms. Table 5.3 contains a list of the stop words we used for potential attributes.[1] For $\mathcal{I}_{nutr}$, the attributes were $\{calories, carbohydrates, cholesterol, fat, fiber, protein, sodium\}$ as this set was static from class to class. We also used the percent daily value for each nutrient as the attribute value. Table 5.4 summarizes the number of attributes for each data set by attribute type.

Table 5.3. Cooking Key Words: Terms used to search the directions of a recipe for potential attributes.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| bake | barbecue | baste | beat | blanch | blend | boil | broil | caramelize |
| chop | clarify | cream | cure | deglaze | degrease | dice | dissolve | dredge |
| drizzle | dust | fillet | flake | flambe | fold | fricassee | fry | garnish |
| glaze | grate | grill | grind | julienne | knead | marinate | mince | mix |
| pan-broil | pan-fry | pare | peel | pickle | poach | puree | reduce | render |
| roast | saute | scallop | score | sear | shred | sift | simmer | skim |
| steam | steep | stew | stir | toss | truss | whip | | |

Table 5.4. Number of Attributes Mined from *AllRecipes.com*

| **Attribute Type** | **ImageNet10** | **50Food** | **ImageNet115** |
|---|---|---|---|
| Nutrient | 7 | 7 | 7 |
| Directions | 44 | 52 | 54 |
| Ingredients | 435 | 981 | 1698 |

[1] Stop words were taken from a cooking glossary `http://www.d.umn.edu/~alphanu/cookery/glossary_cooking.html`.

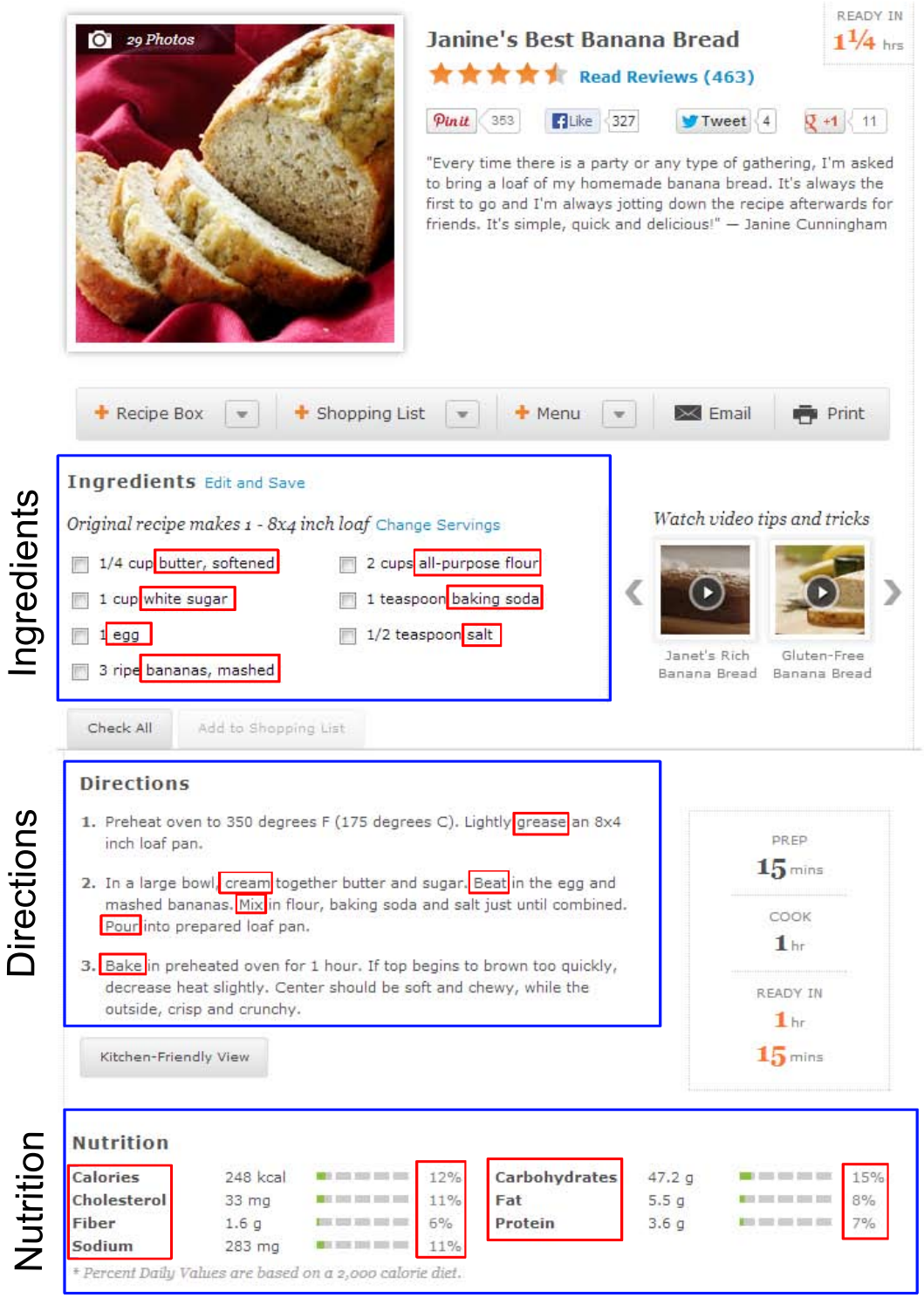



Figure 5.2. Screenshot of a recipe from *AllRecipes.com*: We scraped a recipe for its ingredients, directions, and nutritional information (highlighted in red) to discover attributes for a food class.

### 5.2.2 Training the Attribute Classifiers $\mathcal{H}$

Having created the recipe attribute sets $\mathcal{I}_{ing}, \mathcal{I}_{dir}, \mathcal{I}_{nutr}$, we proceeded to train the individual attribute classifiers $h_m(\mathbf{x})$. (For $\mathcal{I}_{nutr}$ we used regression to predict percent daily nutrient values.) To train each classifier $h_m$, we used the set of training images and attribute labels $\mathcal{S}(i_m) = \{(\mathbf{x}_1, \mathbf{i}_1[m]), \ldots, (\mathbf{x}_N, \mathbf{i}_N[m])\}$, where $\mathbf{x}_n$ was one of the three baseline descriptors and $\mathbf{i}_n[m]$ was the value of the $m$-th attribute of the RBF descriptor $\mathbf{i}$ for the $n$-th image as defined in Definition 5.3.

Since our attribute classification framework can be used with any classifier, we experimented with both SVM and random forests to determine the best method. While SVM classifiers are known for their generally good predictive performance, random forests have a built-in feature selection procedure. This is important as noted by Farhadi et al. [17] to avoid training a correlated attribute. Furthermore, random forests are capable of performing multi-output classification as in [15]. The advantage is that we only need to train one overall attribute classifier instead of $M$ independent SVM classifiers. It simultaneously predicts all attributes, while taking into account the correlation between them.

For the SVM attribute classifiers, each $h_m$ was tuned using five-fold cross-validation to pick the best kernel, its associated parameters, and the regularization parameter $R$, similar to the methodology outlined in Section 3.2. The RBF descriptor $\mathbf{i}$ for a test image is computed by evaluating each $h_m$ for $\mathbf{x}'$ as in Definition 5.4.

For the multi-output random forest, we used the Extremely Randomized Tree module with multiple outputs from the `scikit-learn` library to train our random forest [41]. The random forest was tuned by selecting the optimal number of trees $n_{tree} \in \{1, 25, 50\}$, maximum tree depth $depth_{max} \in \{5, 15\}$, and the maximum number of features to use at each split $feat_{max} \in \{\log_2(d), 0.5d, 0.75d\}$. The minimum number of samples per leaf was set to 15 and the minimum number of samples required for a split was set to 10. The RBF descriptor $\mathbf{i}$ for a test image is computed by evaluating the random forest with $\mathbf{x}'$.

### 5.2.3 Training the Food Classifiers

To train the food classifiers, we concatenated the baseline visual descriptors and the RBF descriptors to form an enhanced descriptor

$$\mathbf{z} = [\mathbf{x},\ \mathbf{i}_{ing},\ \mathbf{i}_{dir},\ \mathbf{i}_{nutr}].$$

Thus, our new training set is $\mathcal{Z} = \{(\mathbf{z}_1, y_1), \ldots, (\mathbf{z}_N, y_N)\}$, which we used to train a collection of one-versus-one class-level SVM classifiers as described in Section 3.2. In our experiments, we tested the performance of all three baseline descriptors enhanced with the RBF descriptor. Table 5.5 shows the parameters for our tuned classifiers. Note that even though we augmented the Meta-Class descriptors with features of a different type, the best classifier is still linear.

Since the RBF descriptor for each training image of the same class is identical (cf. Definition 5.3), it leads to overfitting. Therefore in practice, we computed $\mathbf{i}$ by uniformly sampling

attributes from $\mathcal{I}(c)$ for the training set.

Table 5.5. Optimal SVM Parameters Determined by 5-Fold Cross-Validation for Enhanced Descriptors (see Section 3.2 for further details)

| **Descriptor** | **Data Set** | **Kernel** | $R$ | $\gamma$ |
|---|---|---|---|---|
| BoF | ImageNet10 | RBF | 10 | 0.0001 |
| | 50Food | Linear | 0.001 | N/A |
| | ImageNet115 | RBF | 100 | 0.001 |
| PiCoDes | ImageNet10 | Linear | 0.01 | N/A |
| | 50Food | RBF | 100 | 0.0001 |
| | ImageNet115 | Linear | 1 | N/A |
| Meta-Class | ImageNet10 | Linear | 0.01 | N/A |
| | 50Food | Linear | 0.01 | N/A |
| | ImageNet115 | Linear | 0.01 | N/A |

## 5.3 Results and Analysis

In this section, we present an analysis of the recognition performance of the enhanced baseline descriptors. Please note that due to limited resources, results are based on one test set.

### Comparison of SVM and Random Forest Attribute Classifiers

We first compared the effect of using SVM attribute classifiers $\mathcal{H}_{SVM}$ vs. a random forest attribute classifier $\mathcal{H}_{RF}$ on food recognition. Table 5.6 compares the accuracy of the class-level SVM food classifiers using the enhanced food image descriptors $\mathbf{z} = [\mathbf{x},\ \mathbf{i}_{ing},\ \mathbf{i}_{dir},\ \mathbf{i}_{nutr}]$, where $\mathbf{i}_{ing}, \mathbf{i}_{dir}, \mathbf{i}_{nutr}$ are computed using the trained SVM attribute classifiers or the random forest classifier. The performance between SVM and RF attribute classifiers was comparable but SVMs slightly outperformed RF. Therefore, for the remainder of the analysis we only present results for the SVM attribute classifiers. However, if one is willing to compromise a few percentage points in accuracy for speed, a multi-output RF is the better choice as only one classifier is needed for all attributes.

Table 5.6. Classification Accuracy using SVM vs. Random Forest Attribute Classifiers (Best results bolded)

| **Data Set** | **BoF** | | **PiCoDes** | | **Meta-Class** | |
|---|---|---|---|---|---|---|
| | $\mathcal{H}_{RF}$ | $\mathcal{H}_{SVM}$ | $\mathcal{H}_{RF}$ | $\mathcal{H}_{SVM}$ | $\mathcal{H}_{RF}$ | $\mathcal{H}_{SVM}$ |
| ImageNet115 | **7.1** | 6.9 | 15.2 | **15.5** | **19.7** | 19.5 |
| ImageNet10 | 35.0 | **36.0** | 47.5 | **48.5** | 50.5 | **53.0** |
| 50Food | 33.8 | **33.9** | **56.3** | 53.9 | 64.1 | **65.7** |

### Classification with RBF Descriptors

We then compared the classification performance of the baseline descriptors in Chapter 4 when enhanced with the RBF descriptors. We also evaluated with ground truth RBF descriptors to determine the maximum accuracy possible with these enhanced descriptors, i.e., we assumed the class labels $y_n$ of the test images are known to compute $\mathbf{i}_n$ as in Definition 5.3.

From Table 5.7, recognition was improved when augmenting the baseline descriptors with the RBF descriptors compared to using the basic baseline descriptors. The improvement was the best when using Meta-Class descriptors to train the attribute classifiers, a reflection of its highly discriminative features as concluded in the previous chapter. The improvement was even greater, however, when using the ground truth RBF descriptors. This evidence suggests that the better described the food in terms of its attributes, the better the recognition.

Table 5.7. Classification Accuracy of Baseline vs. Enhanced Baseline Descriptors: Results are shown for classification with baseline descriptors (BL), enhanced baseline descriptors (+RBF), and enhanced baseline descriptors with ground-truth attributes (+GT). (Best results for each data set bolded.)

| **Data Set** | **BoF** | | | **PiCoDes** | | | **Meta-Class** | | |
|---|---|---|---|---|---|---|---|---|---|
| | BL | +RBF | +GT | BL | +RBF | +GT | BL | +RBF | +GT |
| ImageNet115 | 5.2 | 6.9 | 8.8 | 16.2 | 15.5 | 32.7 | 19.0 | **19.5** | 22.9 |
| ImageNet10 | 36.5 | 36.0 | 36.5 | 47.0 | 48.5 | 66.0 | 51.0 | **53.0** | 57.0 |
| 50Food | 33.0 | 33.9 | 33.8 | 58.6 | 53.9 | 70.1 | 64.3 | **65.7** | 67.6 |

Since recipe enhanced Meta-Class descriptors have the best overall classification accuracy, we repeated the experiment over five different training/testing splits of the data to determine its robustness. Table 5.8 shows that the enhanced Meta-Class descriptors consistently outperformed the basic Meta-Class descriptors across all data sets. In addition, the accuracy of both the basic and enhanced Meta-Class descriptors were better than Chen et al. [8] result of 62.7% on the 50Food data set when they used a multi-class SVM as the classifier with their descriptor.

Table 5.8. Comparison of Meta-Class and Enhanced Meta-Class Descriptors (Accuracy averaged over five training/testing splits)

| **Data Set** | **Accuracy (± SE)** | |
|---|---|---|
| | MC | +RBF |
| ImageNet115 | 19.5 ± 0.4 | **19.8 ± 0.4** |
| ImageNet10 | 53.0 ± 1.2 | **53.9 ± 0.6** |
| 50Food | 65.5 ± 0.8 | **66.5 ± 0.6** |

On an individual food class basis, the enhanced Meta-Class descriptors obtained equal or

greater recall scores compared to the basic Meta-Class descriptors for 6/10 ImageNet10 classes (cf.Table 5.9), 40/50 50Food classes (cf.Appendix B:Table B.4/B.5), and 85/115 ImageNet115 classes (cf.Appendix B:Tables B.6-B.9). It also achieved greater precision, recall, and F1-scores on average across all three data sets.

Table 5.9. ImageNet10: Comparison of Meta-Class (MC) vs. MetaClass+RBF (+RBF) Descriptors by Class (Best results bolded)

| **Class** | **Recall** | | **Precision** | | **F1-Score** | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| spaghetti | 0.75 | **0.90** | **0.75** | 0.58 | **0.75** | 0.71 |
| deviled eggs | 0.65 | **0.75** | 0.59 | **0.60** | 0.62 | **0.67** |
| roulade | **0.70** | 0.65 | 0.64 | **0.76** | 0.67 | **0.70** |
| samosa | 0.50 | **0.55** | **0.56** | 0.44 | **0.53** | 0.49 |
| casserole | **0.55** | 0.50 | **0.65** | 0.62 | **0.59** | 0.56 |
| curry | 0.45 | **0.45** | 0.38 | **0.50** | 0.41 | **0.47** |
| pasta salad | **0.50** | 0.40 | 0.43 | **0.57** | **0.47** | **0.47** |
| fried rice | 0.25 | **0.40** | 0.24 | **0.36** | 0.24 | **0.38** |
| kebab | **0.40** | 0.35 | **0.50** | **0.58** | **0.44** | **0.44** |
| paella | 0.35 | **0.35** | **0.41** | 0.35 | **0.38** | 0.35 |
| **Average** | 0.51 | **0.53** | 0.52 | **0.54** | 0.51 | **0.52** |

## Classification with Single Attribute Types

We studied the performance of the Meta-Class descriptors augmented with just one of the $\mathbf{i}_{ing}, \mathbf{i}_{dir}$, and $\mathbf{i}_{nutr}$ descriptors compared with the combined RBF descriptor. Figure 5.3 shows a breakdown of the improvement of the enhanced Meta-Class over the basic Meta-Class descriptors by attribute type. Comparing the three attributes (ingredients, directions, and nutrients), ingredients and directions showed the most improvement, while nutrients did not show any improvement at all or performed even worse than the baseline. This may be because there are more ingredient attributes than directional or nutrient attributes, or because ingredients are more discriminative at describing foods. Interestingly, combining all three descriptors did not necessarily make for the best overall descriptor as in the case of ImageNet10 and ImageNet115 where directions and ingredients alone achieved the best accuracy. We also see the improvement decreased for the 50Food and ImageNet115 data sets, a reflection of their size and difficulty.

We also investigated the classification performance using just the RBF descriptors exclusively. Figure 5.4 shows the accuracy of the food classifiers trained on the RBF descriptors using only ingredients, directions, or nutrient attributes. From the figure, we can make three observations: the more discriminative the visual descriptor $\mathbf{x}$, the better predictive performance of the attribute classifiers $\mathcal{H}$, which in turn resulted in more discriminative attribute descriptors.

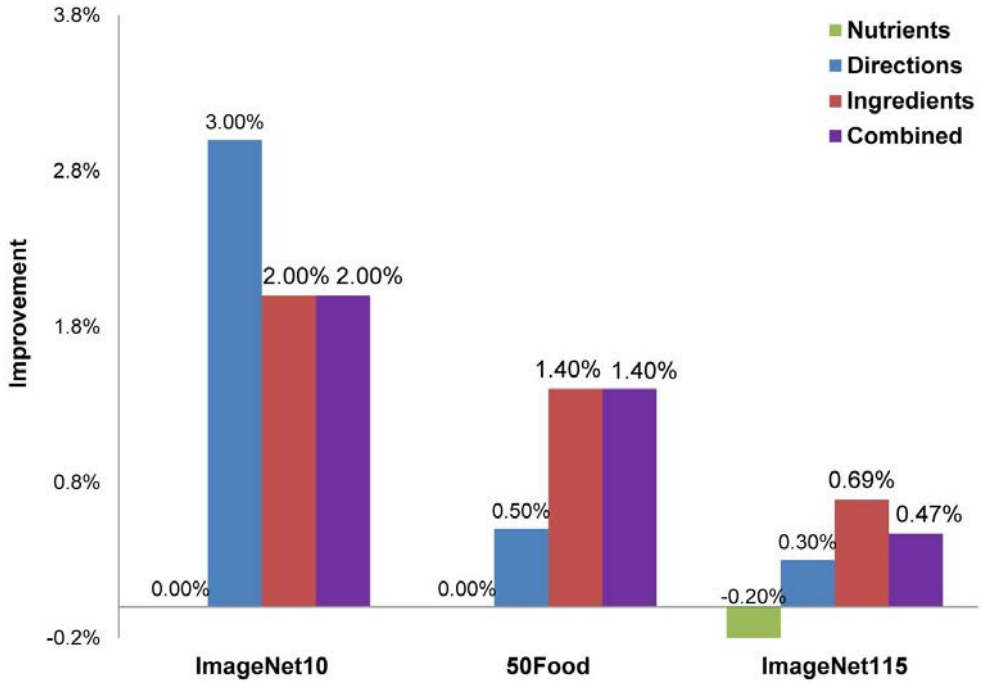


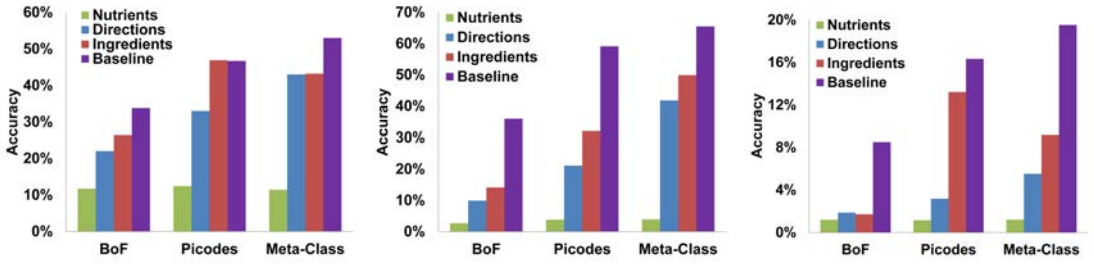

Figure 5.3. Improvement of Recipe Enhanced Meta-Class Over Basic Meta-Class by Attribute Type

Meta-Class descriptors, which we established was the best baseline, gave rise to the most discriminative RBF descriptors. Secondly, ingredient attributes were better for classification than cooking direction attributes, which were better than nutrient attributes, mirroring the result in Figure 5.3. Lastly, while attribute descriptors alone have good predictive performance, they did not do as well as the baseline descriptors, suggesting that both visual and semantic cues are needed for optimal recognition.

In addition, when attributes alone were used for classification, confusion was made with respect to the particular attribute type. For example, when ingredient attributes were used, the most misclassified foods were ones that share similar ingredients (cf. Table 5.10) or when cooking directions were used, the confused foods were ones that are prepared similarly (cf. Table 5.11). A useful application of this by-product would be to use the attribute descriptors to recommend dishes with similar ingredients or that are prepared similarly based on only a food image.

### Application: Predicting Novel Foods

An advantage of an attribute-based descriptor is that the results are easily interpretable. Table 5.12 shows the top five most significant ingredients ranked by the SVM feature weights $\mathbf{w}$ for the classes of ImageNet10. The results make sense; for example, it is natural to add curry powder to curry but not casserole. The weights also reveal novel features that otherwise would not be noticed through traditional object recognition methods, such as the poultry seasoning that is important for roulade. Furthermore, we can use information about these ingredients to




(a) ImageNet10 (b) 50Food (c) ImageNet115

Figure 5.4. Classification Accuracy with Attributes Only: Food classifiers are trained using just recipe-based food descriptors by attribute type. Baseline performance from Figure 4.6 provided here for comparison.

predict novel food categories without having to re-train the classifier.

As a rudimentary zero-shot learning demonstration, we used the attribute classifiers trained on the ImageNet10 data set to predict the top 10 ingredients (ranked by the value of the SVM decision function) of five novel dishes. The predicted ingredients were entered into Google Image Search to see if the novel dish was among the search results. Figures 5.5−5.9 shows the outcome of our demonstration. Since using all ingredients as search terms gave us noisy results, we refined the ingredients list to get more reasonable results. For most cases, Google recalled the particular dish using just the ingredients as search terms. The one case where knowledge transfer failed is Figure 5.8) because our training set did not include desserts, and hence was not trained to identify dessert-related ingredients.

(a) Novel Test Image

diced sweet **onion**

**dried basil**

butter, cubed

**dried thyme**

peeled and deveined shrimp

**tomato puree**

**dried oregano**

garlic salt

lemon, juiced

andouille **sausage**

(b) Top Predicted Ingredients
(search terms bolded)


(c) Google Image Search Results (Hits highlighted in red)

Figure 5.5. Zero-Shot Learning Demonstration: Pizza

(a) Novel Test Image

processed cheese spread with jalapeno

shredded Colby **cheese**

margarine

hash brown **potatoes**

French fried onions

condensed French onion soup

quick-cooking grits

cornflakes cereal

rutabagas

country **gravy** mix

(b) Top Predicted Ingredients

(search terms bolded)

(c) Google Image Search Results (Hits highlighted in red)

Figure 5.6. Zero-Shot Learning Demonstration: Poutine

(a) Novel Test Image

masa harina

Pecorino **cheese**

Genoa **salami**, thinly sliced

German stone ground mustard

**sauerkraut**, drained

**salsa**

top round beef cutlets

whole bone-in turkey breast with skin

dill **pickle**

**spinach**, drained

(b) Top Predicted Ingredients (search terms bolded)


(c) Google Image Search Results (Hits highlighted in red)

Figure 5.7. Zero-Shot Learning Demonstration: Sandwich

(a) Novel Test Image

hash brown **potatoes**

**white corn**, drained

**margarine**

processed **cheese**

quick-cooking **grits**

**cornflakes** cereal

French fried **onions**

yellow squash, sliced

dry corn muffin mix

shredded Colby cheese

(b) Top Predicted Ingredients
(search terms bolded)

(c) Google Image Search Results (Hits highlighted in red)

Figure 5.8. Zero-Shot Learning Demonstration: Egg Tart

(a) Novel Test Image

frozen **puff pastry**

**apple juice**

**oil for deep frying**

beef broth

ajwain seeds

large wonton wrappers

russet **potatoes**

phyllo dough

Kikkoman Thai Red Curry Sauce

dry onion soup mix

(b) Top Predicted Ingredients
(search terms bolded)

(c) Google Image Search Results (Hits highlighted in red)

Figure 5.9. Zero-Shot Learning Demonstration: Fish & Chips

Table 5.10. 50Food: Top 5 Most Confusing Foods by Ingredients

| Food Class 1 | Food Class 2 | No.Mistakes/40 |
|---|---|---|
| Mapo Tofu | Stinky Tofu | 14 |
| Arepas | Buns | 7 |
| Croissants | Buns | 6 |
| Curry Rice | Curry Chicken | 6 |
| Spaghetti | Lasagne | 5 |

Table 5.11. 50Food: Top 5 Most Confusing Foods by Cooking Directions

| Food Class 1 | Food Class 2 | No.Mistakes/40 |
|---|---|---|
| Fried Noodle | Spaghetti | 15 |
| Hot & Sour Soup | Noodle Soup | 14 |
| Corn | Lasagne | 8 |
| Steak | Spaghetti | 7 |
| Gongbao Chicken | Spaghetti | 6 |

Table 5.12. Top 5 Significant Ingredients for ImageNet10 Classes by SVM Feature Weights (Negative weights indicate the ingredient is not needed for the food dish)

| **Food Class** | **Ingredient** | **Weight** | **Food Class** | **Ingredient** | **Weight** |
|---|---|---|---|---|---|
| | spaghetti | −0.795 | | meat tenderizer | 0.839 |
| | curry powder | −0.639 | | skewers | 0.752 |
| casserole | hash brown potatoes | 0.638 | kebab | dried apricots | 0.617 |
| | milk | 0.536 | | chicken breasts | 0.500 |
| | processed cheese | 0.498 | | jalepeno peppers | 0.487 |
| | curry powder | 0.960 | | spinach | 0.538 |
| | curry paste | 0.544 | | bread crumbs | 0.538 |
| curry | dried lentils | 0.514 | roulade | stuffing mix | 0.397 |
| | ground lamb | −0.499 | | poultry seasoning | 0.397 |
| | black pepper to taste | −0.463 | | turkey breast | 0.397 |
| | ranch dressing | 0.962 | | beef broth | 0.656 |
| | mayonnaise | 0.519 | | tamarind extract | 0.625 |
| deviled eggs | eggs, lightly beaten | 0.516 | samosa | flour | 0.604 |
| | spaghetti | −0.475 | | canola oil | 0.581 |
| | prepared mustard | 0.440 | | tumeric | 0.513 |
| | cooked white rice | 0.614 | | rotini pasta | 0.612 |
| | soy sauce | 0.602 | | cucumber | 0.502 |
| fried rice | green onion, chopped | 0.562 | pasta salad | baby spinach | 0.485 |
| | cooking oil | 0.495 | | seashell pasta | 0.483 |
| | eggs, lightly beaten | 0.444 | | grape tomatoes | 0.460 |
| | saffron threads | 0.599 | | spaghetti | 1.639 |
| | chicken broth | 0.467 | | chicken stock | −0.262 |
| paella | uncooked white rice | 0.391 | spaghetti | artichoke hearts | −0.262 |
| | large shrimp | 0.348 | | white onion | −0.262 |
| | eggs, lightly beaten | −0.327 | | green beans | −0.262 |

## Chapter 6

# Conclusion and Future Work

The ability to recognize foods automatically has great value in terms of health benefits. Food recognition is difficult, however, because of its inherent variability within the same classes and yet the subtle distinctions between different classes. In this thesis, we evaluated the recognition performance of three object recognition descriptors: Bag of Features, PiCoDes, and Meta-Class. Of the three, we established that Meta-Class was the best descriptor for food recognition with respect to the food data sets used in our evaluations. In addition to having the best performance, Meta-Class only required a simple, linear SVM classifier to achieve optimal performance, which is advantageous in terms of classifier training.

Despite showing good predictive performance, the baseline descriptors had difficulty distinguishing visually similar foods. As a solution, we proposed a new descriptor designed specifically for food to enhance the baseline methods. Our recipe-based descriptors were able to improve the performance of the baseline descriptors by describing foods in terms of its ingredients, cooking method, and nutritional content. Accordingly, users can easily interpret the results leading to potential applications like identifying gluten-free or nut-free products for those with special dietary restrictions. Furthermore, it allows for knowledge-transfer to novel classes for zero-shot learning.

## 6.1 Future Work

One of the limitations of our method is that we did not have ground truth images to train our attribute classifiers. Consequently we may have been training for attributes that were not present in the images. Since *AllRecipes.com* has images for each of its recipes, we can use these images to pre-train the attribute classifiers. We can also maximize the attribute classifiers' predictive performance by finding better ways of feature selection to ensure they correlate well with the attributes. Alternatively, we can prune the attribute set to ensure attributes are visual qualities that can be detected.

In our work, we only focused on predicting the presence or absence of an ingredient attribute. We can extend this by exploring ways to predict the quantity of each ingredient so that nutritional content can be better inferred. Another improvement is to use natural language

processing to automatically learn the cooking attributes from the recipes' directions instead of using a pre-defined set of stop words. In this way, the attributes are better suited to the data. We are also interested in seeing how humans can help improve food recognition by exploring active-learning techniques.

## Appendix A

# Food Questionnaire

The following are the training images and quiz used in the questionnaire administered in Section 4.1.


Example 1
Example 2
Example 3
Example 4
Example 5
Casserole
Curry
Deviled Egg
Fried Rice
Kebab
Roulade
Samosa
Pasta Salad
Paella
Spaghetti

## 2. Quiz

For each image, select the name of the food item that best describes the dish. Please note that it is possible that a food item can be the correct answer for more than one question.

*Skip to question 1.*

1. **1.** *

*Mark only one oval.*

- ◯ Pasta Salad
- ◯ Paella
- ◯ Curry
- ◯ Kebab

2. **2.** *

*Mark only one oval.*

- ◯ Samosa
- ◯ Curry
- ◯ Deviled Egg
- ◯ Pasta Salad

3. **3.** *
*Mark only one oval.*
- Fried Rice
- Pasta Salad
- Spaghetti
- Samosa

4. **4.** *
*Mark only one oval.*
- Deviled Egg
- Paella
- Curry
- Casserole

(a)

5. **5.** *
*Mark only one oval.*
- Roulade
- Curry
- Samosa
- Deviled Egg

6. **6.** *
*Mark only one oval.*
- Spaghetti
- Fried Rice
- Paella
- Deviled Egg

(b)

7. **7.** *
*Mark only one oval.*
- Casserole
- Roulade
- Curry
- Pasta Salad

8. **8.** *
*Mark only one oval.*
- Curry
- Paella
- Fried Rice
- Kebab

(c)

9. **9.** *
*Mark only one oval.*
- Pasta Salad
- Deviled Egg
- Casserole
- Kebab

10. **10.** *
*Mark only one oval.*
- Deviled Egg
- Kebab
- Paella
- Curry

(d)

## Appendix B

# Full Results for 50Food and ImageNet115

The following tables are the classification scores mentioned in Sections 4.5 and 5.3 for individual classes of the 50Food and ImageNet115 data sets.

Table B.1. 50Food F1-Scores by Food Class

(Best results for each class are bolded)

| | **Meta-Class** | **PiCoDes** | **BoF** | | **Meta-Class** | **PiCoDes** | **BoF** |
|---|---|---|---|---|---|---|---|
| Corn | **0.87** | 0.85 | 0.40 | BraisedPork | **0.65** | 0.60 | 0.33 |
| Chocolate | **0.84** | 0.86 | 0.65 | Bread | **0.65** | 0.59 | 0.20 |
| Dumplings | **0.83** | 0.82 | 0.43 | Sausage | **0.63** | 0.50 | 0.19 |
| Hamburger | **0.83** | 0.70 | 0.30 | Salad | 0.62 | **0.63** | 0.44 |
| Sashimi | **0.83** | 0.68 | 0.50 | Pizza | **0.62** | 0.50 | 0.39 |
| Popcorn | **0.81** | 0.79 | 0.60 | TurnipCake | 0.60 | **0.71** | 0.35 |
| SteamedBun | **0.80** | 0.78 | 0.63 | IceCream | **0.60** | 0.43 | 0.23 |
| Steak | **0.80** | 0.73 | 0.55 | FriedFood | 0.58 | **0.63** | 0.52 |
| MapoTofu | **0.79** | 0.77 | 0.67 | Sandwich | 0.58 | **0.60** | 0.43 |
| SteamedSandwich | **0.77** | 0.68 | 0.55 | Subway | 0.58 | **0.59** | 0.32 |
| Hot&SourSoup | **0.75** | 0.71 | 0.48 | CurryChicken | **0.57** | 0.37 | 0.23 |
| Poutine | **0.75** | 0.46 | 0.32 | Chasiu | **0.56** | 0.50 | 0.30 |
| FriedRice | 0.74 | **0.75** | 0.43 | Omurice | **0.56** | 0.47 | 0.43 |
| GongbaoChicken | **0.74** | 0.73 | 0.41 | SomTam | 0.53 | **0.58** | 0.06 |
| RiceTamale | **0.74** | 0.56 | 0.24 | EggTart | **0.53** | 0.42 | 0.21 |
| PotatoChip | 0.73 | **0.74** | 0.36 | Spaghetti | **0.52** | 0.50 | 0.24 |
| Shumai | 0.72 | **0.74** | 0.47 | Shrimp | **0.51** | 0.47 | 0.11 |
| NoodleSoup | **0.72** | 0.56 | 0.35 | GoiCuon | **0.50** | 0.49 | 0.19 |
| CurryRice | **0.71** | 0.60 | 0.22 | Crab | **0.49** | 0.47 | 0.27 |
| Lasagne | **0.71** | 0.53 | 0.33 | Lobster | **0.47** | 0.44 | 0.12 |
| Sushi | 0.70 | **0.74** | 0.17 | ChickenRice | **0.45** | 0.41 | 0.22 |
| StinkyTofu | **0.70** | 0.40 | 0.22 | Donut | 0.38 | **0.41** | 0.39 |
| FriedNoodle | **0.68** | 0.65 | 0.44 | Croissants | 0.30 | **0.32** | 0.05 |
| PekingDuck | **0.67** | 0.65 | 0.23 | Arepas | 0.27 | **0.34** | 0.19 |
| Buns | **0.67** | 0.51 | 0.31 | Fish&Chips | **0.25** | 0.23 | 0.11 |

Table B.2. ImageNet115 F1-Scores by Food Class

(Best results for each class are bolded)

| | **Meta-Class** | **PiCoDes** | **BoF** | | **Meta-Class** | **PiCoDes** | **BoF** |
|---|---|---|---|---|---|---|---|
| tiramisu | **0.57** | 0.55 | 0.11 | milk | 0.25 | **0.30** | 0.15 |
| cremecaramel | **0.50** | 0.49 | 0.21 | hotdog | 0.25 | **0.27** | 0.00 |
| trifle | **0.48** | 0.36 | 0.06 | ossobucco | **0.25** | 0.19 | 0.04 |
| softdrink | **0.47** | **0.47** | 0.06 | pottage | **0.24** | 0.22 | 0.11 |
| deviledegg | **0.46** | 0.32 | 0.12 | smoothie | **0.24** | 0.11 | 0.10 |
| pavlova | 0.44 | **0.46** | 0.30 | tamale | 0.23 | **0.25** | 0.05 |
| borsch | **0.43** | 0.32 | 0.00 | potatosalad | **0.23** | 0.14 | 0.08 |
| popsicle | **0.42** | 0.37 | 0.17 | roulade | 0.22 | **0.24** | 0.06 |
| coffee | **0.42** | 0.24 | 0.12 | goulash | 0.22 | **0.24** | 0.05 |
| cremebrulee | **0.41** | 0.38 | 0.11 | scrambledegg | **0.22** | 0.18 | 0.00 |
| cheesefondue | **0.38** | 0.34 | 0.11 | poachedegg | **0.22** | 0.05 | 0.00 |
| chowmein | **0.38** | 0.17 | 0.09 | chickensalad | **0.21** | 0.13 | 0.00 |
| tabbouleh | **0.36** | 0.26 | 0.06 | boiledegg | **0.21** | 0.12 | 0.00 |
| vichysoisse | **0.35** | 0.25 | 0.16 | sashimi | **0.21** | 0.05 | 0.17 |
| fruitdrink | **0.34** | 0.29 | 0.00 | sauerkraut | 0.20 | **0.26** | 0.00 |
| alphabetsoup | **0.33** | **0.33** | 0.00 | paella | 0.20 | **0.21** | 0.09 |
| blt | 0.32 | **0.40** | 0.00 | chowder | **0.20** | 0.18 | 0.00 |
| macaroni&cheese | **0.30** | 0.18 | 0.12 | chili | **0.20** | 0.16 | 0.13 |
| oatmeal | 0.29 | **0.30** | 0.10 | succotash | **0.20** | 0.12 | 0.09 |
| tetrazzini | **0.29** | 0.18 | 0.04 | sheppherdspie | **0.20** | 0.10 | 0.17 |
| chocolatefondue | **0.29** | 0.12 | 0.06 | fish&chips | **0.19** | 0.15 | 0.09 |
| pastasalad | **0.28** | 0.27 | 0.10 | tea | **0.19** | 0.06 | 0.06 |
| appleturnover | **0.27** | 0.26 | 0.05 | casserole | **0.18** | 0.16 | 0.00 |
| spaghetti | **0.27** | 0.22 | 0.00 | hotpot | **0.18** | 0.13 | 0.05 |
| wontonsoup | 0.26 | **0.32** | 0.14 | risotto | **0.18** | 0.13 | 0.00 |
| souffle | **0.26** | 0.24 | 0.05 | fruitcustard | **0.18** | 0.11 | 0.00 |
| sushi | **0.26** | 0.24 | 0.05 | coleslaw | **0.18** | 0.10 | 0.05 |
| frozenyogurt | **0.26** | 0.23 | 0.10 | frittata | **0.17** | 0.19 | 0.04 |
| couscous | **0.26** | 0.19 | 0.10 | mousse | **0.17** | **0.17** | 0.06 |

Table B.3. ImageNet115 F1-Scores by Food Class
(continued from Table B.2)

| | **Meta-Class** | **PiCoDes** | **BoF** | | **Meta-Class** | **PiCoDes** | **BoF** |
|---|---|---|---|---|---|---|---|
| pizza | **0.17** | **0.17** | 0.00 | samosa | 0.07 | **0.09** | 0.04 |
| cocoa | **0.17** | 0.14 | 0.00 | fishstew | **0.07** | 0.06 | 0.00 |
| eggsbenedict | **0.17** | 0.04 | 0.12 | hamburger | 0.06 | **0.22** | 0.05 |
| schnitzel | 0.16 | **0.19** | 0.00 | enchilada | 0.06 | **0.07** | 0.00 |
| potpie | **0.16** | 0.05 | 0.09 | stuffedpeppers | 0.06 | **0.07** | 0.00 |
| lentilsoup | **0.16** | 0.05 | 0.04 | taco | **0.06** | 0.05 | 0.00 |
| fishballs | **0.15** | 0.07 | 0.10 | vealparmesan | 0.06 | 0.00 | **0.08** |
| terrine | **0.15** | 0.00 | 0.00 | chickenstew | **0.06** | 0.00 | 0.00 |
| sherbert | 0.14 | **0.23** | 0.00 | ragout | **0.06** | 0.00 | 0.00 |
| greenpeasoup | **0.14** | 0.11 | 0.07 | baconandeggs | 0.05 | **0.16** | 0.05 |
| scampi | **0.14** | 0.10 | 0.04 | cannelloni | 0.05 | **0.11** | 0.05 |
| pilaf | **0.14** | 0.05 | 0.00 | omelette | 0.05 | **0.10** | 0.07 |
| spaghettimeatballs | 0.13 | **0.21** | 0.00 | lasagne | 0.05 | **0.08** | 0.00 |
| lobster | 0.13 | **0.15** | 0.00 | peppersteak | 0.05 | **0.06** | **0.06** |
| beefstew | **0.13** | 0.10 | 0.00 | chickensoup | **0.05** | **0.05** | 0.00 |
| fishstick | **0.13** | 0.07 | 0.00 | hamsandwich | **0.05** | **0.05** | 0.00 |
| chickensandwich | **0.13** | 0.06 | 0.11 | tossedsalad | **0.05** | 0.00 | **0.05** |
| kebab | **0.13** | 0.00 | 0.05 | burrito | **0.04** | 0.00 | 0.00 |
| ratatouille | **0.12** | 0.11 | 0.00 | bisque | 0.00 | **0.12** | 0.06 |
| pirogi | **0.12** | 0.00 | 0.00 | eggroll | 0.00 | **0.06** | 0.00 |
| tostada | **0.11** | 0.09 | 0.05 | salisburysteak | 0.00 | **0.06** | 0.00 |
| spareribs | **0.11** | 0.00 | 0.05 | friedrice | 0.00 | **0.05** | **0.05** |
| chickenkiev | 0.10 | **0.15** | 0.08 | hamandeggs | 0.00 | **0.05** | **0.05** |
| fruitsalad | 0.10 | **0.12** | 0.00 | curry | 0.00 | **0.05** | 0.04 |
| coqauvin | **0.10** | 0.00 | 0.00 | sloppyjoe | 0.00 | 0.00 | **0.06** |
| bbqwings | 0.09 | **0.17** | 0.15 | gazpacho | 0.00 | 0.00 | **0.05** |
| frenchtoast | 0.08 | **0.16** | 0.00 | tempura | 0.00 | 0.00 | **0.04** |
| meatloaf | 0.08 | **0.12** | 0.00 | meatballs | 0.00 | 0.00 | 0.00 |
| icecream | 0.07 | **0.10** | 0.00 | | | | |

Table B.4. 50Food: Comparison of Meta-Class (MC) vs. MetaClass+RBF (+RBF) Descriptors by Class (Best results are bolded))

| **Class** | **Recall** | | **Precision** | | **F1-Score** | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| Chocolate | **0.95** | **0.95** | **0.76** | 0.73 | **0.84** | 0.83 |
| Steak | 0.80 | **0.90** | 0.80 | **0.82** | 0.80 | **0.86** |
| Dumplings | 0.85 | **0.90** | **0.81** | 0.75 | **0.83** | 0.82 |
| SteamedBun | **0.90** | **0.90** | **0.72** | **0.72** | **0.80** | **0.80** |
| Hamburger | **0.85** | **0.85** | 0.81 | **0.89** | 0.83 | **0.87** |
| Corn | **0.85** | **0.85** | **0.89** | 0.85 | **0.87** | 0.85 |
| Sashimi | **0.85** | **0.85** | 0.81 | **0.85** | 0.83 | **0.85** |
| Popcorn | **0.85** | **0.85** | 0.77 | **0.77** | **0.81** | **0.81** |
| SteamedSandwich | 0.75 | **0.85** | **0.79** | 0.74 | 0.77 | **0.79** |
| GongbaoChicken | **0.85** | **0.85** | 0.65 | **0.68** | 0.74 | **0.76** |
| Sushi | 0.75 | **0.85** | 0.65 | **0.68** | 0.70 | **0.76** |
| Bread | 0.65 | **0.80** | 0.65 | **0.84** | 0.65 | **0.82** |
| PotatoChip | 0.75 | **0.80** | **0.71** | 0.70 | 0.73 | **0.74** |
| Salad | **0.80** | **0.80** | 0.50 | **0.59** | 0.62 | **0.68** |
| NoodleSoup | 0.70 | **0.75** | 0.74 | **0.75** | 0.72 | **0.75** |
| MapoTofu | **0.75** | **0.75** | **0.83** | 0.68 | **0.79** | 0.71 |
| BraisedPork | **0.75** | **0.75** | 0.58 | **0.65** | 0.65 | **0.70** |
| Lasagne | **0.75** | **0.75** | **0.68** | 0.62 | **0.71** | 0.68 |
| StinkyTofu | **0.75** | **0.75** | **0.65** | 0.62 | **0.70** | 0.68 |
| FriedFood | **0.75** | **0.75** | 0.47 | **0.54** | 0.58 | **0.63** |
| FriedRice | **0.70** | **0.70** | **0.78** | **0.78** | **0.74** | **0.74** |
| Hot&SourSoup | **0.75** | 0.70 | 0.75 | **0.78** | **0.75** | 0.74 |
| Poutine | **0.75** | 0.70 | 0.75 | **0.78** | **0.75** | 0.74 |
| Shumai | 0.65 | **0.70** | **0.81** | 0.74 | **0.72** | **0.72** |
| CurryRice | **0.75** | 0.70 | 0.68 | **0.70** | **0.71** | 0.70 |
| FriedNoodle | **0.70** | **0.70** | **0.67** | 0.64 | **0.68** | 0.67 |
| Sausage | 0.55 | **0.65** | 0.73 | **0.81** | 0.63 | **0.72** |
| RiceTamale | **0.65** | **0.65** | **0.87** | 0.68 | **0.74** | 0.67 |
| Subway | 0.55 | **0.65** | 0.61 | **0.68** | 0.58 | **0.67** |
| GoiCuon | 0.55 | **0.65** | 0.46 | **0.50** | 0.50 | **0.57** |
| Omurice | 0.45 | **0.60** | **0.75** | **0.75** | 0.56 | **0.67** |
| TurnipCake | **0.60** | **0.60** | 0.60 | **0.67** | 0.60 | **0.63** |
| Buns | **0.70** | 0.60 | **0.64** | 0.60 | **0.67** | 0.60 |

Table B.5. Continued from Table B.4

| **Class** | **Recall** | | **Precision** | | **F1-Score** | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| PekingDuck | **0.65** | 0.60 | **0.68** | 0.60 | **0.67** | 0.60 |
| Sandwich | **0.65** | 0.60 | 0.52 | **0.57** | 0.58 | **0.59** |
| Chasiu | **0.55** | **0.55** | 0.58 | **0.65** | 0.56 | **0.59** |
| CurryChicken | 0.50 | **0.55** | **0.67** | 0.65 | 0.57 | **0.59** |
| Pizza | **0.60** | 0.55 | 0.63 | **0.65** | **0.62** | 0.59 |
| ChickenRice | 0.50 | **0.55** | 0.42 | **0.61** | 0.45 | **0.58** |
| EggTart | 0.50 | **0.55** | **0.56** | 0.52 | 0.53 | **0.54** |
| SomTam | **0.50** | **0.50** | 0.56 | **0.59** | 0.53 | **0.54** |
| IceCream | **0.60** | 0.45 | 0.60 | **0.64** | **0.60** | 0.53 |
| Shrimp | **0.45** | **0.45** | **0.60** | 0.56 | **0.51** | 0.50 |
| Crab | **0.50** | 0.45 | 0.48 | **0.50** | 0.49 | **0.47** |
| Spaghetti | **0.40** | **0.40** | **0.73** | **0.73** | **0.52** | **0.52** |
| Lobster | **0.40** | **0.40** | **0.57** | 0.50 | **0.47** | 0.44 |
| Donut | **0.40** | **0.40** | **0.36** | **0.36** | **0.38** | **0.38** |
| Arepas | 0.25 | **0.35** | **0.29** | 0.26 | 0.27 | **0.30** |
| Croissants | **0.25** | **0.25** | 0.38 | **0.50** | 0.30 | **0.33** |
| Fish&Chips | **0.20** | 0.15 | 0.33 | **0.38** | **0.25** | 0.21 |
| **Average** | 0.64 | **0.66** | 0.65 | **0.66** | 0.64 | **0.65** |

Table B.6. ImageNet115: Comparison of Meta-Class (MC) vs. MetaClass+RBF (+RBF) Descriptors by Class (Best results are bolded)

| Class | Recall | | Precision | | F1-Score | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| cremecaramel | **0.70** | **0.70** | 0.39 | **0.42** | 0.50 | **0.53** |
| softdrink | **0.65** | **0.65** | **0.37** | **0.37** | **0.47** | **0.47** |
| chowmein | 0.45 | **0.60** | 0.33 | **0.41** | 0.38 | **0.49** |
| pavlova | 0.55 | **0.60** | **0.37** | 0.35 | **0.44** | **0.44** |
| popsicle | 0.60 | **0.60** | **0.32** | **0.32** | **0.42** | **0.42** |
| borsch | 0.45 | **0.50** | 0.41 | **0.53** | 0.43 | **0.51** |
| tiramisu | **0.60** | 0.50 | **0.55** | 0.48 | **0.57** | 0.49 |
| fruitdrink | 0.40 | **0.45** | 0.30 | **0.35** | 0.34 | **0.39** |
| tabbouleh | 0.35 | **0.45** | **0.37** | 0.31 | 0.36 | **0.37** |
| blt | 0.40 | **0.45** | 0.27 | **0.29** | 0.32 | **0.35** |
| chocolatefondue | 0.40 | **0.45** | **0.23** | **0.23** | 0.29 | **0.31** |
| deviledegg | **0.40** | **0.40** | **0.53** | 0.44 | **0.46** | 0.42 |
| spaghetti | 0.25 | **0.40** | 0.29 | **0.36** | 0.27 | **0.38** |
| macaronicheese | 0.35 | **0.40** | 0.26 | **0.31** | 0.30 | **0.35** |
| cheesefondue | **0.45** | 0.40 | **0.33** | 0.30 | **0.38** | 0.34 |
| trifle | **0.40** | 0.35 | **0.62** | 0.54 | **0.48** | 0.42 |
| coffee | **0.40** | 0.35 | **0.44** | 0.37 | **0.42** | 0.36 |
| couscous | 0.30 | **0.35** | **0.23** | 0.21 | **0.26** | **0.26** |
| hotdog | 0.30 | **0.35** | **0.21** | **0.21** | 0.25 | **0.26** |
| potatosalad | 0.30 | **0.35** | **0.18** | 0.17 | **0.23** | **0.23** |
| ossobucco | **0.35** | **0.35** | **0.19** | 0.15 | **0.25** | 0.21 |
| smoothie | 0.25 | **0.30** | 0.23 | **0.35** | 0.24 | **0.32** |
| frozenyogurt | **0.30** | **0.30** | 0.22 | **0.32** | 0.26 | **0.31** |
| cremebrulee | **0.40** | 0.30 | **0.42** | 0.30 | **0.41** | 0.30 |
| alphabetsoup | **0.35** | 0.30 | **0.30** | 0.27 | **0.33** | 0.29 |
| vichysoisse | **0.35** | 0.30 | **0.35** | 0.27 | **0.35** | 0.29 |
| chickensalad | **0.30** | **0.30** | 0.16 | **0.21** | 0.21 | **0.24** |
| boiledegg | 0.25 | **0.30** | **0.19** | 0.18 | 0.21 | **0.22** |
| wontonsoup | **0.35** | 0.30 | **0.21** | 0.15 | **0.26** | 0.20 |
| pastasalad | **0.30** | 0.25 | 0.26 | **0.36** | 0.28 | **0.29** |
| roulade | 0.20 | **0.25** | 0.24 | **0.31** | 0.22 | **0.28** |
| tamale | 0.20 | **0.25** | 0.27 | **0.31** | 0.23 | **0.28** |
| sushi | **0.25** | **0.25** | **0.28** | 0.26 | **0.26** | **0.26** |
| souffle | **0.25** | **0.25** | **0.26** | 0.24 | **0.26** | 0.24 |

Table B.7. Continued from Table B.6

| Class | Recall | | Precision | | F1-Score | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| goulash | 0.20 | **0.25** | **0.25** | 0.23 | 0.22 | **0.24** |
| milk | **0.25** | **0.25** | **0.25** | 0.22 | **0.25** | 0.23 |
| hotpot | 0.20 | **0.25** | 0.16 | **0.20** | 0.18 | **0.22** |
| casserole | 0.20 | **0.25** | 0.16 | **0.19** | 0.18 | **0.22** |
| sashimi | **0.25** | **0.25** | **0.19** | 0.17 | **0.21** | 0.20 |
| appleturnover | **0.30** | 0.25 | **0.25** | 0.15 | **0.27** | 0.19 |
| scrambledegg | 0.20 | **0.25** | **0.25** | 0.13 | **0.22** | 0.17 |
| lentilsoup | 0.20 | **0.25** | **0.14** | 0.12 | 0.16 | **0.17** |
| tea | 0.15 | **0.20** | 0.25 | **0.40** | 0.19 | **0.27** |
| risotto | 0.15 | **0.20** | 0.23 | **0.36** | 0.18 | **0.26** |
| oatmeal | **0.35** | 0.20 | 0.25 | **0.31** | **0.29** | 0.24 |
| fishballs | 0.15 | **0.20** | 0.15 | **0.25** | 0.15 | **0.22** |
| chowder | **0.20** | **0.20** | 0.20 | **0.22** | 0.20 | **0.21** |
| fruitcustard | **0.20** | **0.20** | 0.17 | **0.22** | 0.18 | **0.21** |
| schnitzel | 0.15 | **0.20** | 0.18 | **0.21** | 0.16 | **0.21** |
| succotash | 0.15 | **0.20** | **0.30** | 0.21 | 0.20 | **0.21** |
| sloppyjoe | 0.00 | **0.20** | 0.00 | **0.17** | 0.00 | **0.19** |
| scampi | 0.15 | **0.20** | 0.13 | **0.14** | 0.14 | **0.16** |
| eggsbenedict | **0.25** | 0.20 | **0.13** | 0.11 | **0.17** | 0.15 |
| lobster | 0.15 | **0.20** | **0.12** | 0.11 | 0.13 | **0.14** |
| fishstick | 0.10 | **0.15** | 0.20 | **0.27** | 0.13 | **0.19** |
| pilaf | **0.15** | **0.15** | 0.13 | **0.21** | 0.14 | **0.18** |
| tetrazzini | **0.25** | 0.15 | **0.33** | 0.21 | **0.29** | 0.18 |
| chili | **0.20** | 0.15 | **0.20** | **0.20** | **0.20** | 0.17 |
| mousse | **0.15** | **0.15** | **0.20** | **0.20** | **0.17** | **0.17** |
| ratatouille | 0.10 | **0.15** | 0.14 | **0.20** | 0.12 | **0.17** |
| fishandchips | **0.20** | 0.15 | 0.17 | **0.19** | **0.19** | 0.17 |
| paella | **0.20** | 0.15 | **0.19** | **0.19** | **0.20** | 0.17 |
| pizza | **0.15** | **0.15** | **0.19** | **0.19** | **0.17** | **0.17** |
| cocoa | **0.15** | **0.15** | **0.19** | 0.18 | **0.17** | 0.16 |
| tostada | 0.10 | **0.15** | 0.13 | **0.18** | 0.11 | **0.16** |
| chickensoup | 0.05 | **0.15** | 0.06 | **0.15** | 0.05 | **0.15** |
| potpie | **0.20** | 0.15 | **0.14** | 0.14 | **0.16** | 0.15 |
| sauerkraut | **0.20** | 0.15 | **0.19** | 0.14 | **0.20** | 0.14 |
| frittata | **0.15** | **0.15** | **0.19** | 0.12 | **0.17** | 0.13 |

Table B.8. Continued from Table B.7

| Class | Recall | | Precision | | F1-Score | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| kebab | **0.15** | **0.15** | **0.12** | **0.12** | **0.13** | **0.13** |
| meatloaf | 0.10 | **0.15** | 0.07 | **0.10** | 0.08 | **0.12** |
| fruitsalad | **0.10** | **0.10** | 0.11 | **0.29** | 0.10 | **0.15** |
| coleslaw | **0.15** | 0.10 | **0.23** | 0.22 | **0.18** | 0.14 |
| greenpeasoup | **0.10** | **0.10** | **0.22** | **0.22** | **0.14** | **0.14** |
| terrine | **0.10** | **0.10** | **0.29** | 0.20 | **0.15** | 0.13 |
| chickensandwich | **0.10** | **0.10** | **0.18** | **0.18** | **0.13** | **0.13** |
| spaghettimeatballs | **0.10** | **0.10** | **0.20** | 0.18 | **0.13** | **0.13** |
| sherbert | **0.15** | 0.10 | 0.14 | **0.15** | **0.14** | 0.12 |
| sheppherdspie | **0.20** | 0.10 | **0.20** | 0.14 | **0.20** | 0.12 |
| hamburger | 0.05 | **0.10** | 0.07 | **0.13** | 0.06 | **0.11** |
| pirogi | **0.10** | **0.10** | **0.14** | 0.13 | **0.12** | 0.11 |
| icecream | 0.05 | **0.10** | 0.10 | **0.12** | 0.07 | **0.11** |
| poachedegg | **0.25** | 0.10 | **0.19** | 0.11 | **0.22** | 0.11 |
| bbqwings | **0.10** | **0.10** | **0.08** | **0.08** | **0.09** | **0.09** |
| chickenkiev | **0.10** | **0.10** | **0.10** | 0.08 | **0.10** | 0.09 |
| coqauvin | **0.10** | **0.10** | **0.10** | 0.08 | **0.10** | 0.09 |
| frenchtoast | 0.10 | **0.10** | 0.07 | **0.08** | 0.08 | **0.09** |
| omelette | 0.05 | **0.10** | 0.04 | **0.06** | 0.05 | **0.08** |
| meatballs | 0.00 | **0.05** | 0.00 | **0.17** | 0.00 | **0.08** |
| chickenstew | **0.05** | **0.05** | 0.06 | **0.14** | 0.06 | **0.07** |
| beefstew | **0.10** | 0.05 | **0.18** | 0.12 | **0.13** | 0.07 |
| eggroll | 0.00 | **0.05** | 0.00 | **0.11** | 0.00 | **0.07** |
| gazpacho | 0.00 | **0.05** | 0.00 | **0.11** | 0.00 | **0.07** |
| samosa | **0.05** | **0.05** | **0.10** | 0.09 | **0.07** | 0.06 |
| vealparmesan | **0.05** | **0.05** | 0.07 | **0.09** | **0.06** | **0.06** |
| enchilada | **0.05** | **0.05** | **0.09** | 0.08 | **0.06** | **0.06** |
| peppersteak | **0.05** | **0.05** | 0.05 | **0.07** | 0.05 | **0.06** |
| stuffedpeppers | **0.05** | **0.05** | **0.09** | 0.07 | **0.06** | **0.06** |
| baconandeggs | **0.05** | **0.05** | 0.05 | **0.06** | 0.05 | **0.06** |
| taco | **0.05** | **0.05** | **0.07** | 0.06 | **0.06** | **0.06** |
| cannelloni | **0.05** | **0.05** | **0.05** | 0.04 | **0.05** | 0.05 |
| lasagne | **0.05** | **0.05** | **0.05** | 0.04 | **0.05** | 0.04 |
| bisque | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| burrito | **0.05** | 0.00 | **0.03** | 0.00 | **0.04** | 0.00 |

Table B.9. Continued from Table B.8

| Class | Recall | | Precision | | F1-Score | |
|---|---|---|---|---|---|---|
| | MC | +RBF | MC | +RBF | MC | +RBF |
| curry | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| fishstew | **0.05** | 0.00 | **0.10** | 0.00 | **0.07** | 0.00 |
| friedrice | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| hamandeggs | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| hamsandwich | **0.05** | 0.00 | **0.06** | 0.00 | **0.05** | 0.00 |
| pottage | **0.20** | 0.00 | **0.31** | 0.00 | **0.24** | 0.00 |
| ragout | **0.05** | 0.00 | **0.06** | 0.00 | **0.06** | 0.00 |
| salisburysteak | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| spareribs | **0.10** | 0.00 | **0.11** | 0.00 | **0.11** | 0.00 |
| tempura | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** |
| tossedsalad | **0.05** | 0.00 | **0.05** | 0.00 | **0.05** | 0.00 |
| **Average** | 0.19 | **0.20** | 0.18 | **0.19** | **0.18** | **0.18** |

# Appendix C

# Source Code

## C.1 Code to Generate Table 5.8

```
import numpy as np
import scipy.io
import pylab as pl

from sklearn import svm
from sklearn.metrics import roc_curve, auc, accuracy_score, f1_score
from sklearn.cross_validation import StratifiedKFold
from labels import *
import pickle
import multiprocessing
from functools import partial
from sklearn.grid_search import GridSearchCV
import time

################################################################################
# Attribute Classifiers
def trainAttributeClassifiers(attr,trainlabels,Recipes,fooddish,traindata):
    #find positive examples of attribute i
    attr_labels = np.zeros((len(trainlabels),),dtype=np.uint8)
    for j in xrange(len(fooddish)):
if attr in Recipes[fooddish[j]]:
    pos_idx = np.where(trainlabels==j)
    attr_labels[pos_idx[0]] = 1

    #split training data into k-folds
    kfold = StratifiedKFold(attr_labels,2)
    param_grid = [
      {'C': [0.001, 0.01, 1, 10, 100], 'gamma': [0.0], 'kernel': ['linear'], 'degree':[3]},
      {'C': [0.001, 0.01, 1, 10, 100], 'gamma': [0.01, 0.001, 0.0001], 'kernel': ['rbf'], 'degree':[3]},
```

```
     ]

    #cross-validate
    cv = GridSearchCV(estimator=svm.SVC(class_weight='auto'), param_grid=param_grid, cv=kfold)
    cv.fit(traindata,attr_labels)

    #fit best model
    attributeclassifier = svm.SVC(C=cv.best_params_['C'],
                                  kernel=cv.best_params_['kernel'],
                                  gamma = cv.best_params_['gamma'],
                                  degree=cv.best_params_['degree'],
                                  class_weight='auto')
    attributeclassifier.fit(traindata,attr_labels)
    return attributeclassifier

##############################################################################
if __name__=="__main__":

    # import data
#path to Matlab file
    dataset = "vlg_extractor/ImageNetSurveyMC/ImageNetSurveyMC"
    var=scipy.io.loadmat(dataset)
    traindata = var['X'].astype(np.float32)
    trainlabels = var['trainlabels'].flatten().astype(int)
    testdata = var['Xtest'].astype(np.float32)
    testlabels = var['testlabels'].flatten()
    X = np.concatenate((traindata,testdata),0)
    y = np.concatenate((trainlabels,testlabels),0)
    n_samples, n_features = X.shape
    del traindata, testdata, var

    # load Recipe Dictionary and food labels
    recipedict = recipeDict[0]
    fooddish = fooddish[0]
    I,R = pickle.load(file(recipedict,'r'))
    ingsorted = sorted(I.keys())[1:]

    # flatten recipes
    Recipes = {}
    for dish in fooddish:
recipes = R[dish].values()
ing = [r[0].keys() for r in recipes]
```

```
Recipes[dish] = set().union(*ing)

    ###############################################################################
    # Run classifier with cross-validation

    # Create training/testing splits
    cv = StratifiedKFold(y, n_folds=5)
    scores = np.zeros((2,5))
    scores2 = np.zeros((2,5))

    for k, (train, test) in enumerate(cv):
print time.asctime(time.localtime(time.time())), "Trial: ", k
#change these parameters accordingly
classifier = svm.SVC(C=1, kernel='linear')
classifier2 = svm.SVC(C=0.001, kernel='linear')

#fit baseline
classifier.fit(X[train], y[train])
scores[0,k]=accuracy_score(y[test],classifier.predict(X[test]))
scores[1,k]=f1_score(y[test],classifier.predict(X[test]))
del classifier
print time.asctime(time.localtime(time.time())), "BL fitted"

#train attribute classifiers
pool = multiprocessing.Pool()
AttributeClassifier = partial(trainAttributeClassifiers,
                              trainlabels=y[train],
                              Recipes=Recipes,
                              fooddish=fooddish,
                              traindata=X[train])
attributeclassifiers= pool.map(AttributeClassifier,ingsorted)
pool.close()
pool.join()

print time.asctime(time.localtime(time.time())), "Attribute classifiers trained"

# predict attributes
IngredientsTest = np.zeros((len(y[test]),len(ingsorted)),dtype=np.float32)
for i in xrange(len(attributeclassifiers)):
    IngredientsTest[:,i] = attributeclassifiers[i].predict(X[test])
XExtraTest = np.concatenate((X[test],IngredientsTest),1)
#uncomment this to save attribute features to disk
```

```
#pickle.dump(IngredientsTest,file("/".join(dataset.split('/')[0:2])+'/IngredientAttributes'+str(k)+'.npy','w'))
del attributeclassifiers

thresholds = [0,0.25,0.5,0.75,0.95]
acc = [None]*thresholds
f1 = [None]*thresholds
for t in xrange(len(thresholds)):
    #sample attributes
    IngredientsTrain = np.zeros((len(y[train]),len(ingsorted)),dtype=np.float32)
    for i in xrange(len(y[train])):
dish = fooddish[y[i]]
IngredientsTrain[i,:] = [1 if ing in Recipes[dish] and prob > thresholds[t]
                         else 0 for ing,prob in zip(ingsorted,np.random.random((len(ingsorted),)))]

    XExtraTrain = np.concatenate((X[train],IngredientsTrain),1)

    #fit enhanced baselines
    classifier2.fit(XExtraTrain,y[train])
    acc[t] = accuracy_score(y[test],classifier2.predict(XExtraTest))
    f1[t] = f1_score(y[test],classifier2.predict(XExtraTest))

scores2[0,k] = max(acc)
scores2[1,k] = max(f1)
print time.asctime(time.localtime(time.time())), "Enhanced classifier trained"
del classifier2

    # write out scores and standard errors
    f = file(dataset+"-CVScores.txt",'w')
    f.write("Method\tAccuracy\tF1\n")
    f.write("BL\t %0.3f (+/-%0.03f)\t" % (scores[0,:].mean(), scores[0,:].std()/sqrt(5)))
    f.write("%0.3f (+/-%0.03f)\n" % (scores[1,:].mean(), scores[1,:].std()/sqrt(5)))

    f.write("Attribute\t %0.3f (+/-%0.03f)\t" % (scores2[0,:].mean(), scores[0,:].std()/sqrt(5)))
    f.write("%0.3f (+/-%0.03f)\n" % (scores2[1,:].mean(), scores[1,:].std()/sqrt(5)))
    f.close()
```

## C.2 Code to Scrape *AllRecipes* for Attributes

```
from bs4 import BeautifulSoup as bs
import urllib2
import pickle
import numpy as np
import scipy.io
```

```
import random as rnd
from labels import *
import sys

# Ingredient Scraper for ingredients, cooking terms and nutritional info
def IngredientScraper(fooddish):
    #dictionary for ingredients
    I = {}
    #dictionary for food recipes
    R = {}

    website = 'http://allrecipes.com'
    for food in fooddish:
        R[food] = {}
        #search for recipes
        print food

resultspage = urllib2.urlopen("http://allrecipes.com/search/default.aspx?qt=k&wt="+food)
results = bs(resultspage)
for recipelinks in results.find_all('a',class_='title'):
    recipelink = recipelinks.get('href')
    #go to recipe page
    recipepage = urllib2.urlopen(website+recipelink)
    recipe = bs(recipepage)
    recipename = recipe.find('h1',id='itemTitle').text

    if recipename not in R[food]:
print "Recipe: ", recipename

# find ingredients for this recipe
ingredients = recipe.find_all('li', id='liIngredient')
#list containing ingredients, cookingterms, nutritionrating
R[food][recipename] = [{},[],[0]*7]
for ing in ingredients:
    ingid = ing.attrs['data-ingredientid']
    ingname = ing.find(id='lblIngName').text
    if ingid not in I:
I[ingid] = ingname
    amt=float(ing.attrs['data-grams'])
    R[food][recipename][0][ingid] = amt

#uncomment to normalize values
```

```
#m = sum(R[food][recipename][0].values())
#R[food][recipename][0]={ingid: R[food][recipename][0][ingid]/m for ingid in R[food][recipename][0].keys()}

#get cooking terms
directions = [step.text.lower() for step in recipe.find_all('span', class_='plaincharacterwrap break')]
R[food][recipename][1] = directions

#get nutrition
nutritionrating = recipe.find_all('ul', id='ulNutrient')
n = 0
for nutrient in nutritionrating:
    #category = nutrient.find('li',class_='categories').text
    R[food][recipename][2][n]=float(nutrient.find('li',id='divNutrientGradient').attrs['style'][6:-1])/100
    n += 1

    # save Recipe info to disk
    pickle.dump((I,R),file('AllRecipes.npy','w'))



if __name__=="__main__":
    #usage: python IngredientScraper.py datasetID
    #datasetID 0 = ImageNet10
    #          1 = 50Food
    #          2 = ImageNet115
    IngredientScraper(fooddish[int(sys.argv[1])])
```

Remaining code included in accompanied DVD

# References


[1] T. L. Berg, A. C. Berg, and J. Shih. Automatic attribute discovery and characterization from noisy web data. In *Computer Vision–ECCV 2010*, pages 663–676. Springer, 2010. vi, 11

[2] A. Bergamo and L. Torresani. Meta-class features for large-scale object categorization on a budget. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3085–3092. IEEE, 2012. 2, 25, 26

[3] A. Bergamo, L. Torresani, and A. W. Fitzgibbon. Picodes: Learning a compact code for novel-category recognition. In *Advances in Neural Information Processing Systems*, pages 2088–2096, 2011. 2, 23

[4] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 19

[5] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007. 24

[6] M. Bosch, F. Zhu, N. Khanna, C. J. Boushey, and E. J. 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. 1, 7, 20

[7] 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. 5, 7, 20, 21

[8] M.-Y. Chen, Y.-H. Yang, C.-J. Ho, S.-H. Wang, S.-M. Liu, E. Chang, C.-H. Yeh, and M. Ouhyoung. Automatic chinese food identification and quantity estimation. In *SIGGRAPH Asia 2012 Technical Briefs*, page 29. ACM, 2012. vi, 7, 8, 12, 17, 20, 39

[9] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995. 14

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 22

[11] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *Computer Vision–ECCV 2010*, pages 71–84. Springer, 2010. 16, 26

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. vi, 13, 18, 24

[13] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013. 9, 30

[14] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3474–3481. IEEE, 2012. 11

[15] M. Dumont, R. Marée, L. Wehenkel, and P. Geurts. Fast multi-class image annotation with random subwindows and multiple output randomized trees. In *VISAPP (2)*, pages 196–203, 2009. 37

[16] A. Fagot-Campagna, J. B. Saaddine, K. M. Flegal, and G. L. Beckles. Diabetes, impaired fasting glucose, and elevated hba1c in us adolescents: the third national health and nutrition examination survey. *Diabetes Care*, 24(5):834–837, 2001. 1

[17] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1778–1785. IEEE, 2009. 2, 10, 37

[18] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 13

[19] V. Ferrari and A. Zisserman. Learning visual attributes. In *Advances in neural information processing systems*, 2007. 10

[20] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221–228. IEEE, 2009. 24, 25

[21] A. H. Goris, M. S. Westerterp-Plantenga, and K. R. Westerterp. Undereating and underrecording of habitual food intake in obese men: selective underreporting of fat intake. *The American journal of clinical nutrition*, 71(1):130–134, 2000. 1

[22] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 24

[23] A. B. Hillel and D. Weinshall. Subordinate class recognition using relational object models. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, 19:73, 2007. 8

[24] 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. 7, 20

[25] S. J. Hwang, K. Grauman, and F. Sha. Semantic kernel forests from multiple taxonomies. In *Advances in Neural Information Processing Systems 25*, pages 1727–1735, 2012. 26

[26] 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

[27] K. Kitamura, C. de Silva, T. Yamasaki, and K. Aizawa. Image processing based approach to food balance analysis for personal food logging. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 625–630. IEEE, 2010. 5

[28] K. Kitamura, T. Yamasaki, and K. Aizawa. Foodlog: Capture, analysis and retrieval of personal food images via web. In *Proceedings of the ACM multimedia 2009 workshop on Multimedia for cooking and eating activities*, pages 23–30. ACM, 2009. 5

[29] 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

[30] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009. 2, 10

[31] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 21, 24

[32] M. Marszalek and C. Schmid. Semantic hierarchies for visual object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–7. IEEE, 2007. 26

[33] C. Martin, H. Han, S. Coulon, H. Allen, C. Champagne, and S. Anton. A novel method to remotely measure food intake of free-living people in real-time: The Remote Food Photography Method (RFPM). *British Journal of Nutrition*, 101:446–456, 2009. 1

[34] C. K. Martin, S. Kaya, and B. K. Gunturk. Quantification of food intake using food image analysis. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6869–6872. IEEE, 2009. 5

[35] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002. 25

[36] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 1–12. ACM, 2011. 1

[37] C. L. Ogden, M. D. Carroll, L. R. Curtin, M. M. Lamb, and K. M. Flegal. Prevalence of high body mass index in us children and adolescents, 2007-2008. *JAMA: the journal of the American Medical Association*, 303(3):242–249, 2010. 1

[38] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006. 24

[39] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009. 2

[40] D. Parikh and K. Grauman. Interactively building a discriminative vocabulary of nameable attributes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1681–1688. IEEE, 2011. 10, 11

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 16, 37

[42] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive psychology*, 8(3):382–439, 1976. 8

[43] D. A. Schoeller, L. G. Bandini, and W. H. Dietz. Inaccuracies in self-reported intake identified by comparison with the doubly labelled water method. *Canadian journal of physiology and pharmacology*, 68(7):941–949, 1990. 1

[44] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 24

[45] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 21, 22

[46] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):480–492, 2012. 24, 26

[47] L. Yang, N. Zheng, H. Cheng, J. D. Fernstrom, M. Sun, and J. Yang. Automatic dietary assessment from fast food categorization. In *IEEE 34th Annual Northeast Bioengineering Conference 2008*, 2008. 5, 20

[48] S. Yang, L. Bo, J. Wang, and L. Shapiro. Unsupervised template learning for fine-grained object recognition. In *Advances in Neural Information Processing Systems 25*, pages 3131–3139, 2012. vi, 9, 10, 30

[49] 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. vi, 6

[50] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3466–3473. IEEE, 2012. 9, 30

[51] B. Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1577–1584. IEEE, 2011. 9, 30