# Geolocalized Modeling for Dish Recognition

Ruihan Xu, Luis Herranz, *Member, IEEE*, Shuqiang Jiang, *Senior Member, IEEE*, Shuang Wang, Xinhang Song, Ramesh Jain, *Fellow, IEEE*

***Abstract*—Food-related photos have become increasingly popular, due to social networks, food recommendation and dietary assessment systems. Reliable annotation is essential in those systems, but unconstrained automatic food recognition is still not accurate enough. Most works focus on exploiting only the visual content while ignoring the context. To address this limitation, in this paper we explore leveraging geolocation and external information about restaurants to simplify the classification problem. We propose a framework incorporating discriminative classification in geolocalized settings and introduce the concept of geolocalized models, which in our scenario are trained locally at each restaurant location. In particular, we propose two strategies to implement this framework: geolocalized voting and combinations of bundled classifiers. Both models show promising performance, and the later is particularly efficient and scalable. We collected a restaurant-oriented food dataset with food images, dish tags and restaurant-level information, such as the menu and geolocation. Experiments on this dataset show that exploiting geolocation improves around 30% the recognition performance, and geolocalized models contribute with an additional 3˜8% absolute gain, while can be trained up to five times faster.**

***Index Terms*—food recognition, geolocation, image tagging, mobile applications**

## I. Introduction

Eating is essential for human life, both from personal and socio-cultural perspectives. Thus, food is connected to many aspects and activities in daily life, including health, culture, leisure and social events. For instance, one new trend is sharing dining-out experiences on photo-enabled social networks. In fact, people are increasingly interested in discovering and sharing new cuisines, and knowing more about different aspects of the food they consume. Another popular application is keeping a personal log of daily meals and food intake.

Food photos are popular, but in general users annotate them poorly, either with rather useless tags (e.g. "today's lunch", "delicious"), not accurate or generic tags (e.g. "Italian food", "yellow rice") and even wrong tags. In fact, this is not surprising, as accurate photo annotation requires specific domain knowledge and manual textual input is time-consuming and prone to typos. Thus, automatic annotation from a photo taken with the smartphone is much more convenient for the user, and automatic tags are more accurate and useful for retrieval applications. Reliable automatic food recognition can enable countless functionalities in these systems. Examples include automatic photo tagging, image-based retrieval (e.g. recipe, dietary properties), recommendation (e.g. food, recipe, restaurant).

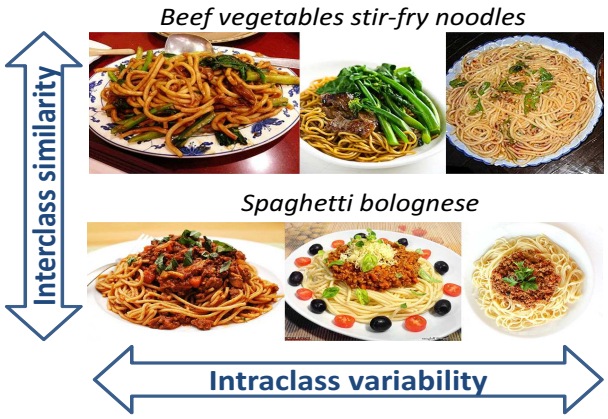



Figure 1. Examples illustrating similarities and variabilities for two different dish classes.

Many works on food recognition have been proposed in recent years based on different visual representations [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], although most of them are limited to a few food classes in controlled settings. Accurate food recognition from only visual information is still a challenge. In contrast to objects, food items are highly deformable and with high intra-class variability, e.g. different cooking styles and seasonings will lead to different appearances of the same food. Moreover, different foods share many ingredients and often differences between some food classes are almost imperceptible (see Fig. 1).

In order to address complex recognition problems, humans leverage prior and contextual knowledge. Often, automatic systems also incorporate context and external knowledge to solve a simpler problem. For instance, a tagging system can exploit internal and external context, by exploiting the personal tagging history and other users' tags in similar photos [11]. Personal interests and social circles are also helpful contexts [12]. Mobile phones can capture rich contextual information, in particular they can estimate their geographic location (i.e. geolocation) from GPS and mobile networks via their location services. The paradigmatic application is touristic or urban landmark recognition [13], [14], [15] with smartphones. Typically, using image retrieval techniques, the most similar images are retrieved, but only considering images whose geolocation is in geographic neighborhood. Classifiers can be used instead and the candidate classes are limited to those a few candidates based on the geolocation (i.e. *shortlist* approach [13]), although the nature of landmarks and buildings (rigid

R.Xu, L.Herranz, S.Jiang, S.Wang and X.Song are with Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China, e-mail: {ruihan.xu, luis.herranz, shuqiang.jiang, shuang.wang, xinhang.song}@vipl.ict.ac.cn
R.Jain is with University of California, Irvine, CA 92697, USA, email: jain@ics.uci.edu

objects, and intrinsically invariant) usually makes retrieval-based approaches with geometric verification more effective. The advantage of exploiting contextual information is twofold: simplifies the problem making it easier to solve, and reduces the computational complexity.

Similarly, we could exploit context for food recognition. In particular, in this paper we focus on a particular yet common scenario: *dish recognition in restaurants* (we use the term *dish* to emphasize that it is related to restaurants, and also more specific than the term food). This scenario has contextual information we can exploit, since the ingredients, cooking style and presentation of dishes and which dishes (i.e. menu) are very restaurant-specific, and restaurants are also naturally linked to a geolocation. Therefore, we propose exploiting information about the geographic location of the photo and about the dishes in the menu of the restaurants near that location, including user contributed images of those dishes (crawled from online review websites, such as Yelp or Dianping).

Due to the intrinsic characteristics of food images, we prefer using explicit discriminative classification, rather than a retrieval-based approach (more suitable for landmarks). We first adapt the shortlist approach to our scenario. Using a global classifier trained over all the dish classes, during test we geolocalize the problem by finding the restaurants within the geographic neighborhood of the test image and selecting only the dishes in their menus as potential classes for the test image. As we will see later, this approach has some limitations, related with the mismatch between the test settings (query-dependent) and the training settings (global), that is, models learned for the whole database are used to discriminate over a query-dependent subset of classes.

We address the test-training mismatch problem by introducing the concepts of geolocalized models and the related geolocalized training and model combination. By geolocalizing models the complexity of each model is lower, and the training classes and training data are more similar to those found for a particular query image. Then, for a particular query image, geolocalized models of candidate restaurants are combined. For dish recognition in restaurants, dish models are localized in the corresponding restaurants, and trained to discriminate between the dish classes in the menus of the same and neighboring restaurants. We describe two specific strategies to implement this framework and analyze their characteristics and performance. Experiments show that these strategies not only improve the recognition accuracy but also the efficiency.

In summary, motivated by the problem of dish recognition in restaurants, the main contributions of this paper are:

- We analyze the problem of discriminative classification in geolocalized settings, and propose *geolocalized modeling* as an effective framework.
- In addition to an adaptation of the shortlist approach, we propose two strategies to implement geolocalized models: *geolocalized voting* and *combinations of bundled classifiers* (CBC).
- We collected a specific restaurant-oriented food dataset, including restaurant-specific information such as menus and geolocation.

The rest of this paper is organized as follows. Section II reviews the related work. Sections III and IV introduce the proposed framework and the two implementation strategies. Experiments and conclusions are presented in Sections V and VI.

## II. Related Work

### A. Food recognition

Previous works on dish recognition are mainly based on analyzing the visual appearance. Some works address food recognition using conventional visual features trying to capture the global appearance of the food. Joutou and Yanai [6] proposed an automatic food image recognition system based on multiple kernel learning (MKL), which integrates several kinds of image features (e.g. color, texture, SIFT) learning an optimal linear combination of feature-specific kernels. Hoashi *et al.* [7] extended the system proposed in [6] with more image features and food classes. Maruyama *et al.* [8] improved the recognition accuracy by incrementally updating the classifier based on a Bayesian network. Zong *et al.* [2] proposed to exploit the structure of the food object which is represented as the spatial distribution of the local textural structures and encoded using shape context. Kawano *et al.*[9] compute Fisher vectors over HOG patches to develop a real-time mobile food recognition system. Recently, they extended the system to 256 food categories [10].

Other works consider food as a certain combination of different components (ingredients). Yang *et al.* [1] proposed an American fast food recognition system by using pairwise local features, which effectively captures important shape characteristics and spatial relationships between food ingredients. Dietcam [3] analyzes a meal by taking several images (or a short video), estimating the volume of each food items and finally estimating the caloric intake. The recognition accuracy is increased through modeling food geometric locations and a joint probability model. Zhang [16] proposed to classify plates of food to the correct cuisine using attribute-based classification, where the ingredients are considered attributes of a plate of food. Kawano *et al.* [5] built an interactive real-time food recognition system. First, the user locates the meal with a bounding box and then each food item region is segmented with GrabCut [17]. A color histogram and SURF-based bag-of-features are then extracted and fed into SVM classifiers. Finally, some works [18], [4] focus on multiple dish recognition. In this work we focused on close-up photos of dishes, which are also the common type we find in social networks.

Most existing works in food recognition have been focused only on content-based analysis, not considering any contextual information. In contrast, our work explores the potential of geographical context to improve the recognition accuracy.

### B. Image recognition exploiting geolocation

Previous works exploiting geographical information to help visual recognition mainly target landmarks (e.g. the Eiffel Tower) and location-specific concepts (e.g. lions in the forest). Ji *et al.* [19] classify them into four groups: (1) geographical

location recognition, (2) landmark mining, (3) tourism recommendation and (4) 3D scene modeling and city navigation. Yap *et al.* [13] have made a comparative study of mobile-based landmark recognition, in which content classifiers are offline trained and context is used to shortlist several candidate landmarks, then content analysis is performed for recognition (for convenience we refer to this approach as *shortlist*). Chen *et al. [20]* score the images in database using a vocabulary tree trained on SIFT descriptors, and geographically distant landmarks are excluded using GPS coordinates associated with the query image, then approximate nearest neighbors (ANN) is applied to find the nearest feature vectors within the candidates. Yaegashi and Yanai [21] exploited geotags for photo recognition by including two types of geographical information: raw values of latitude and longitude and visual features extracted from aerial photos around the geotagged location. In [22], these two kinds of features are combined using MKL. Zheng *et al.* [23] mined and modeled worldwide landmarks by using agglomerative hierarchical clustering on the geotag coordinates. More related works can be found in recent surveys [24], [19].

Not only the geolocation can be used to recognize images, but images themselves can be useful to estimate the geolocation. Li et al. [25], estimate the unknown geolocation of an image by searching visually similar images in a large set of geotagged images.

To the best of our knowledge, geographical location has not been exploited for food image recognition, and most geocontext-based image recognition works deal with outdoor photos such as landmark or urban. This type of images, in contrast to food images, are relatively invariant (apart from extrinsic factors such as illumination, viewpoint, etc). Besides, most methods are variants of retrieving similar images to the query (perhaps with some nearest neighbors classification) after some geocontext-based filtering to shortlist the candidate images. However, these approaches may not work well on the more deformable food images, especially when there are very few images for each class, thus more discriminative methods are necessary.

## III. Discriminative classification in geolocalized settings

### A. Dish recognition in restaurants

In contrast to generic food recognition, *dish recognition in restaurants* emphasizes two elements. First, the problem is localized, that is, we assume the user (and consequently the photo) is located inside a restaurant. Second, we use the more specific term *dish* rather than food to emphasize the relation with the menu of a restaurant. In general, accurate dish recognition is very challenging, since the combined number of classes can be very large. Variations in the ingredients and different cooking and presentations used in different restaurants can cause a large visual variability for the same dish, while accidental similarity between non-related dishes causes inter-class similarity (see Fig. 1). And these problems become more significant in larger datasets with more restaurants and dishes.

In food recognition the objective is to identify the class $s$ of an input image, represented by some visual descriptor $\mathbf{x}$. This is achieved using a visual classifier, that we represent as $p(s|\mathbf{x})$. For dish recognition in restaurants, in addition to the visual classifier, we have access to the menu of the restaurant and to the geographic location of both the restaurants and the image. Thus, a query to the system is a pair $Q = (\boldsymbol{\Psi}_q, \mathbf{x})$, where $\boldsymbol{\Psi}_q = (\lambda_q, \phi_q)$ are the geographical coordinates (estimated by the location services of the mobile device), with $\lambda_q$ and $\phi_q$ denoting latitude and longitude. Similarly, a restaurant $k$ is modeled as a pair $R_k = (\boldsymbol{\Psi_k}, M_k)$, representing its menu $M_k$ (i.e. the dish classes found in that particular restaurant) and its geographic location $\boldsymbol{\Psi_k} = (\lambda_k, \phi_k)$. For simplicity we project the location onto a local coordinate system (with origin at the average coordinates of the dataset), and use local coordinates $\boldsymbol{\varphi}_k = (u_k, v_k)$ for the restaurant location, and $\boldsymbol{\varphi}_q = (u_q, v_q)$ for the query image. The restaurant database contains $K$ restaurants with a combined total of $D = \left|\bigcup_{k=1}^{K} M_k\right|$ dishes. The menu is represented as $M_k = \{s_1, \ldots, s_{D_k}\}$, where $s_i$ is the $i$-th dish in the restaurant menu $M_k$, with $D_k$ different dishes.

While dish recognition is a very complex problem, contextual knowledge about menus and geolocations can be very helpful to simplify the problem and boost recognition performance. However, the way this information is incorporated and the type of models used can have a significant impact on both efficiency and accuracy.

### B. Naive strategy: shortlist approach

The simplest way to improve the performance of a (global) classifier by including geolocation is to use the query geolocation to *reduce the candidate classes*. The adaptation to the case of dish recognition in restaurants is straightforward. The candidates are those dish classes in the menus of the restaurants in the geographic neighborhood of the query. We refer to this method as the *shortlist* approach [13]. Given the local coordinates $\boldsymbol{\varphi}_q$ and the visual feature $\mathbf{x}$, predicting the dish is equivalent to finding the dish with maximum probability among the candidates

$$s^* = \arg\max_{s \in G_q} p(s|\mathbf{x}) \tag{1}$$

where the set of candidate restaurants $H_q$ and candidate dish classes $G_q$ for a query are obtained as

$$\begin{aligned} H_q &= H\left(\boldsymbol{\varphi}_q, \epsilon\right) \\ &= \left\{k \mid \left\|\boldsymbol{\varphi}_k - \boldsymbol{\varphi}_q\right\| \leq \epsilon, \forall k = 1, \ldots, K\right\} \\ G_q &= \bigcup_{k \in H_q} M_k \end{aligned} \tag{2}$$

where $\epsilon$ is the maximum distance from the candidate restaurants to the query image.

It is trivial to see that discriminating between fewer classes makes the average accuracy to increase because the new problem is simpler. Note that we do not adapt the classifier to the query's features, but to its context (e.g. geolocation). Although useful, this naïve strategy still uses a global model. Thus, the

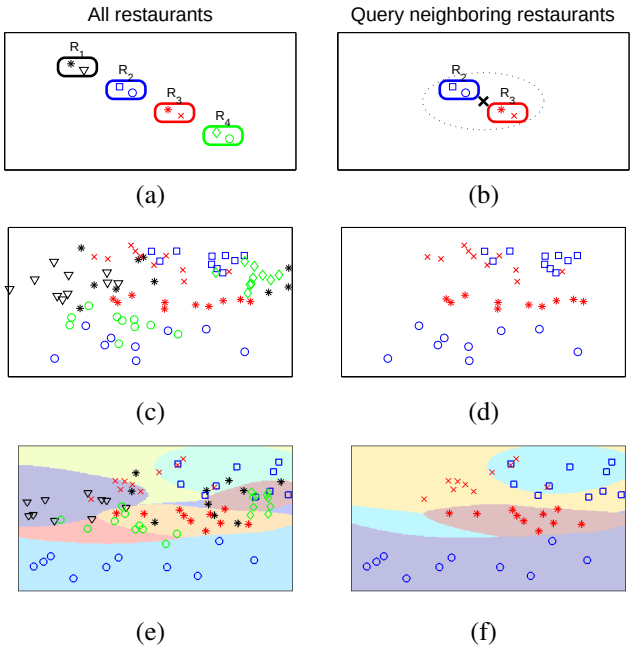



Figure 2. Toy example using a global model in geolocalized settings: (a) restaurants in the dataset, (b) restaurants near the query image and candidate classes, (c) training data, (d) geolocalized training data (i.e. only candidate classes), (e) test data and decision regions for all the classes, (f) test data and decision regions for the candidate classes.

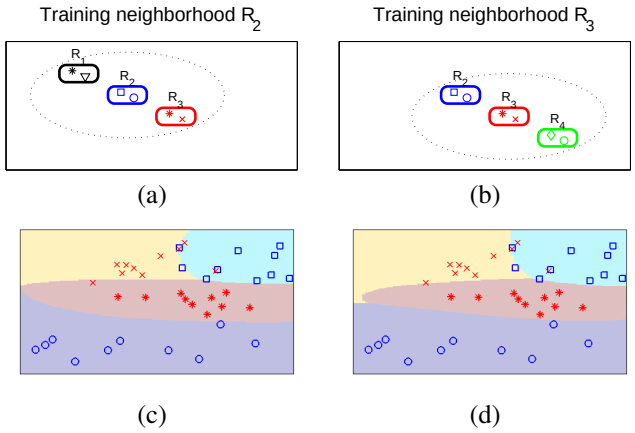



Figure 3. Different restaurant neighborhoods in the toy example: (a) training for restaurant $R_2$, (b) training for restaurant $R_3$, (c) decision regions for an optimal geolocalized model (retrained only with the classes in the query neighborhood), (d) decision regions using CBC (see Section IV-B).

model is not adapted to the test settings of each query, only adapting the scores of the classifiers (or probabilities). We illustrate the limitations of this strategy with a toy example. We consider four restaurants $R_1$, $R_2$, $R_3$ and $R_4$ with two dishes each, distributed geographically as shown in Fig. 2a. The training data of the different dishes are represented in Fig. 2c in a two dimensional feature space. A multi-class Support Vector Machine (SVM) is trained with this training data. The decision regions and some test samples are shown in Fig. 2e. Now, let us consider a query image and its estimated geolocation. Due to the error in the location, the device finds $R_2$ and $R_3$ as candidate restaurants, and thus reduces the potential classes to four candidate classes (see Fig. 2b). Classifying into four classes is simpler than classifying into eight. We see that after discarding some classes in this example the candidate classes are relatively easy to separate. However, while the global classifier can classify correctly most of the test samples, there are still some misclassification and the decision regions are more complex than necessary for this simple problem (see Fig. 2f).

### C. Test-training mismatch in geolocalized settings

The shortlist approach is an example of *classification in geolocalized settings*, in which the classification process is modified by geolocation information *at query time*. The reason why the global model in the toy example cannot discriminate properly between the candidate classes is because it has been trained to discriminate between the original classes. We refer to this problem as *test-training mismatch*, because classes and data involved during training are different from the classes and data involved during test after adapting the classifier to the query. Even ignoring classifiers from classes that are not included in the candidate set (i.e. shortlist) does not guarantee that the classifier will discriminate properly between the remaining classes, since the remaining classifiers were also trained with negative samples from the discarded classes. Thus, those negative samples from the discarded classes introduce certain training noise and bias the models.

A better way would be adapting not only the classes to the query, but also adapting the training data. Lazy classifiers not requiring explicit training, such as $k$-NN, implicitly adapt the training data when used in geolocalized settings. For this reason they are very common in landmark recognition. However, discriminative classifiers require explicit training. Adapting the training data for each query implies that the model has to be retrained for each query. As the problem settings (classes and training data) depend on the particular geographic location of the query image, the resulting classifier is optimal for that query. The resulting classifier discriminates better the data, with more robust decision regions (see Fig. 3c).

### D. Geolocalized modeling framework

While being the optimal case, retraining models for each query is not feasible in practice, due to the high training cost. However, between this optimal but impractical case and the less accurate global models, there are alternatives which can exploit geolocalization during both training and test time. Thus, we propose *geolocalized modeling* to address the test-training mismatch problem, where models are trained to discriminate only against those classes likely to be candidates simultaneously in the query.

The key difference with generic food or dish recognition is that we are not interested in solving the complex problem of modeling all types of dishes and their possible variations, but in simpler problems where we need to discriminate between fewer candidate classes *given the geolocation and restaurant information*. Thus, while a generic dish classifier is a complex model $p(s|\mathbf{x})$, we aim at simpler models $p(s|\mathbf{x}, \boldsymbol{\varphi}_q; \{R\})$ ( $\{R\}$ represents the information about restaurants) that can

discriminate better only within the candidate dishes for a given query.

The proposed framework is shown in Fig. 4, adapted to the scenario of dish recognition in restaurants. In particular, we use restaurants as geographical anchors for candidate classes. We first construct a database of restaurants including geographical locations and menus. This information is obtained from restaurant review websites, which also include images of the corresponding dishes. Only dishes with images are considered. Then, using these images, we train geolocalized models for each dish and store them. Each model is related to a particular geolocation.

During test time, the particular geolocation of the query defines a neighborhood with some candidate restaurants. For each query, the corresponding geolocalized models are selected and combined into a new classifier adapted to the query. In the next section we describe two strategies to implement this approach. Note that the shortlist approach can also be considered a special case, with a single global model adapted during query time.

## IV. Implementing geolocalized models

In this section we propose two strategies to implement geolocalized modeling. The basic idea is to train dish models to discriminate against the rest of the dishes in the menu and in the menus of neighboring restaurants. Fig. 3a and b show the geolocalized training neighborhood and classes using two different restaurants data. An appropriate geolocalized training and model combination at query time can closely approximate the ideal case without the cost of retraining the classifier (compare Fig. 3c and d). However, the particular strategy to implement geolocalized models has impact on the training and test costs and on the scalability.

### *A. Geolocalized strategy 1: pairwise models and geolocalized voting*

One way to implement multiclass classifiers (e.g. SVM) is training multiple binary pairwise classifiers (also known as one-against-one or OAO classifiers). Then, the input feature is classified by all of them, and the outputs of these classifers are combined and the result is decided. A simple yet widely used method is voting [26], where each binary classifier votes for either one of the two classes it can discriminate. The predicted class is simply the class with more votes.

We can observe that pairwise classifiers, in principle, do not suffer from the problem of training with data from unrelated classes as only the two classes involved are used for training. We can easily obtain a geolocalized version just discarding the set of classes out of the candidate set still keeps *all* the pairwise combinations of the candidate classes. Besides, each of these models was trained only with data from the two classes involved, both belonging to the candidate set. This is equivalent to re-training the same OAO candidate-specific classifiers for each query, but without actually performing any costly training procedure, just selecting models. Thus, here the strategy consists of *selecting pairwise classifiers*.

**Algorithm 1** Geolocalized voting.

**Input:** Visual feature vector $\mathbf{x}$, phone local coordinates $\boldsymbol{\varphi}_q$ and pairwise classifiers models $f_{ks,lp}(\mathbf{x})$
**Output:** Predicted dish class $s^*$

```
Find the set of candidate classes C(H_q) from φ_q using (2) and (3)
for (k, s) ∈ C(H_q) do
    votes[k, s] = 0 ∀(k, s)
end for
for (k, s) ∈ C(H_q) do
    for (l, p) ∈ C(H_q), (l, p) ≠ (k, s) do
        if f_{ks,lp}(x) ≥ 0 then
            votes[k, s] = votes[k, s] + 1
        else
            votes[l, p] = votes[l, p] + 1
        end if
    end for
end for
Find (k*, s*) = arg max_{(k,s)} votes[k, s]
return s*
```

*1) Classifier selection and combination:* In our scenario (see Fig. 5), a pairwise classifier $f_{ks,lp} : X \mapsto \mathbb{R}$ discriminates between the dish $s \in M_k$ of restaurant $k$ and the dish $p \in M_l$ of restaurant $l$. In a geolocalized setting, for a set of candidate restaurants $H_q$ we find the combinations of candidate classes as

$$C(H_q) = \{(k, s) \mid s \in M_k, \forall k \in H_q\} \tag{3}$$

Then we adopt a voting strategy [26] to combine the results of the binary classifiers. This is equivalent to the geolocalized voting algorithm shown in Algorithm 1.

*2) Geolocalized training:* Each binary classifier requires much fewer training samples (from only two classes) than one-against-all (OAA) classifiers, making training a particular pairwise model very efficient. However, the number of pairwise models is typically much larger, as they have to model all the pairwise dish combinations. In particular, for $K$ restaurants with an average number of dishes $\bar{D}$, there are $K\bar{D}(K\bar{D}-1)/2$ pairwise models. This is the main limitation of this framework, in particular for a large number of restaurants and large menus, as the pool of classifiers would require to train and store a large number of models, with a significant impact on training cost and memory requirements. Another drawback is that, during test, all these pairwise classifiers must be evaluated for each query image.

Fortunately, most of the pairwise models will never be used in practice, as the geographic distance between most restaurants is too large, so their dishes will never simultaneously be candidate classes. Using this fact, we can design a sparse pool of classifiers by considering only the pairwise relations between neighboring restaurants as

$$\mathcal{L}(\gamma) = \{f_{ks,lp} \mid \|\boldsymbol{\varphi}_k - \boldsymbol{\varphi}_l\| \leq \gamma, \forall s \in M_k \quad \forall p \in M_l, \forall k, l = 1, \ldots, K\} \tag{4}$$

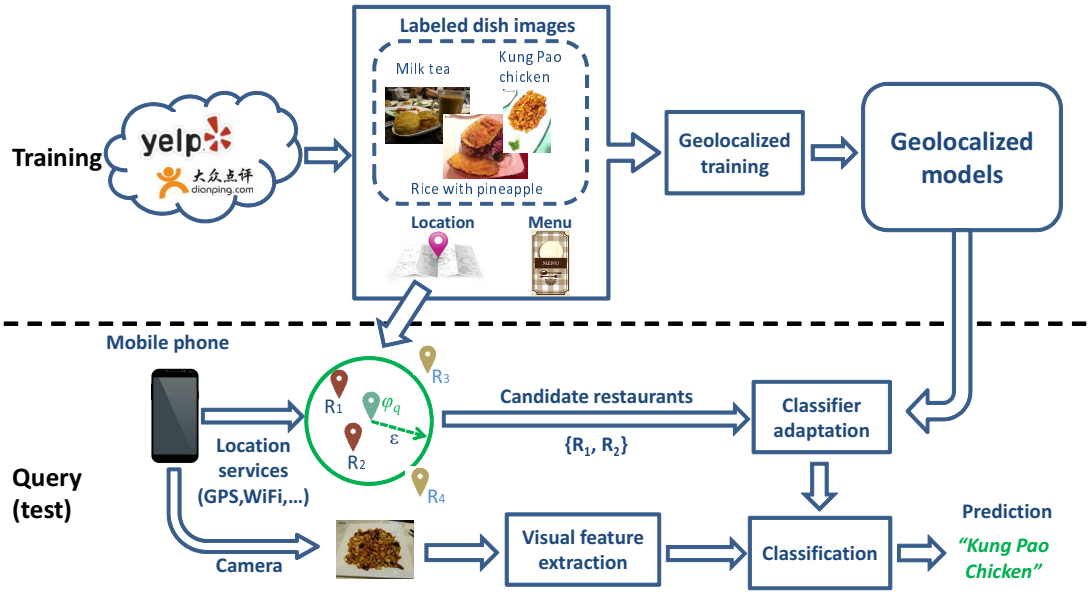



Figure 4. Overview of the framework for dish recognition with geolocalized models.

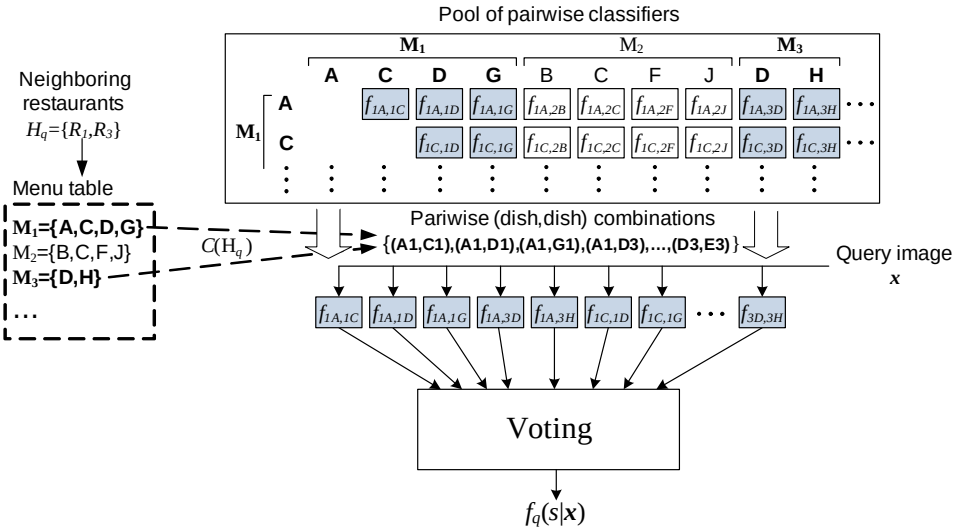



Figure 5. Classification using pairwise dish models. In this figure, *A1* denotes the dish A in the menu $M_1$ of restaurant $R_1$.

where $\gamma$ is the radius of the training neighborhood. The prediction would be obtained using Algorithm 1, but only considering the models in $\mathcal{L}(\gamma)$. Fig. 6 illustrates how the parameter $\gamma$ controls the sparsity of the classifier pool. Note that if $\gamma$ is too small, some pairwise combinations may be ignored, and some required pairwise models would not be included in the pool. Assuming a fixed $\epsilon$ for the radius of the candidate area, setting $\gamma \geq 2\varepsilon$ guarantees that all the required pairwise models are included in the pool (i.e. worst case scenario: two restaurants in the opposite sides of the circumference of radius $\epsilon$) .

Although this sparse pool of classifiers reduces significantly the cost during training, in many cases (e.g. dense areas with many restaurants, large menus) the number of classes may still be large and a larger number (quadratic) of pairwise classifiers need to be evaluated to classify a query image. This leads to a significant time cost during test.

### *B. Geolocalized strategy 2: combination of bundled classifiers (CBC)*

We also investigate geolocalized one-against-all models, which have the advantage that the number of binary classifiers is linear with the number of classes.

*1) Classifier selection and combination:* We can design a geolocalized version of OAA models by learning models locally and then combining them depenging on each query. In particular we use restaurants as local anchors, so each restaurant and its dishes are linked to a particular geographic location. For a particular restaurant $k$, we train a binary classifier for each dish in its menu. All the dishes are treated as part of the same bundle (i.e. the restaurant), that is why we refer to them as *bundled classifiers*.

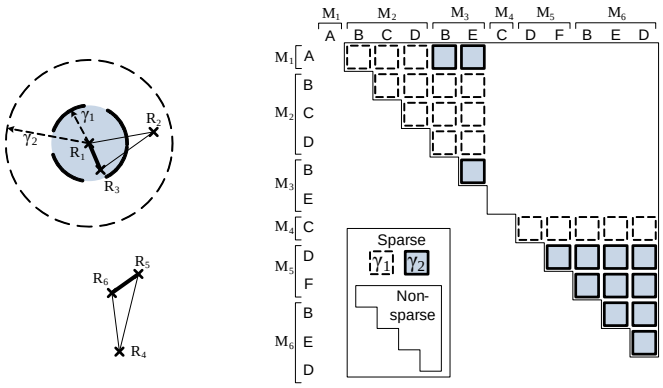



Figure 6. Neighboring restaurants graph (left) and sparse pool of models (right). A smaller radius $\gamma$ results in fewer pairwise combinations, making the pool more sparse.

During test time, the classifier is adapted to the query depending on the combination of models from neighboring restaurants as illustrated in Fig. 7a. Here the strategy is to *select and combine bundles of classifiers* (i.e. restaurants) based on their geolocations. For a particular query with geolocation $\boldsymbol{\varphi}_q$ the predicted dish is obtained as

$$s^* = \underset{s \in M_k, k \in H_q}{\arg\max}\ p_{k,s}(\mathbf{x}) \tag{5}$$

where $p_{k,s}(\mathbf{x})$ is the probability that $\mathbf{x}$ is of class $s$ of restaurant $k$ and $H_q$ is the set of neighboring restaurants obtained from (2).

*2) Geolocalized training:* Now we need to learn $K$ geolocalized models $p_{k,s}(\mathbf{x})$, one per each bundle $k$. In the first approach, we can assume that bundles are independent and thus we only need to train $p_{k,s}(\mathbf{x})$ to discriminate between classes in the menu $M_k$ of restaurant $k$ (see Fig. 7b). For each class $s \in M_k$, we learn a binary classifier $f_{k,s}(\mathbf{x})$. Typically, as in the case of SVMs, multiclass classification is implemented by selecting the class whose binary classifier has the maximum score. An important assumption is that the scores of the binary classifiers are comparable. This is reasonable for each bundle of classifiers, which has been trained with the same settings and with the same classes and training samples.

However, during query time multiple bundles are combined. Different bundles have different training settings and training data, so the scores of their binary classifiers are not directly comparable. Thus, we convert the scores $f_{k,s}(\mathbf{x})$ to probabilities by fitting a logistic function (still a binary function for a given $k$ and $s$)

$$\begin{aligned} p_{k,s}(\mathbf{x}) &= \frac{1}{1+\exp\left(A_{k,s} f_{k,s}(\mathbf{x}) + B_{k,s}\right)} \\ &= \sigma\left(-A_{k,s} f_{k,s}(\mathbf{x}) - B_{k,s}\right) \end{aligned} \tag{6}$$

where the parameters $A_{k,s}$ and $B_{k,s}$ are learned using Platt's scaling method [27], and $\sigma()$ is the logistic function. The resulting models $p_{k,s}(\mathbf{x})$ are stored in the pool of binary geolocalized models.

Discriminative classifiers trained only with classes in one bundle cannot discriminate properly between classes in other bundles. Thus, we propose including also neighboring restaurants when training geolocalized models (see Fig. 8a). For the bundle $k$, the training set of neighboring bundles $H_k$ is obtained as

$$H_k = H(\boldsymbol{\varphi}_k, \gamma) = \{l \mid \|\boldsymbol{\varphi}_k - \boldsymbol{\varphi}_l\| \leq \gamma, \forall l = 1, \ldots, K\} \tag{7}$$

An OAA classifier is learned for all the classes in $H_k$ (e.g. ten classes in Fig. 7), and Platt's scaling is also performed including neighboring classes and data. Note that neighboring restaurants are used to increase the discriminative capability, but only the classifiers in the related bundle are kept. Classifiers for other neighboring restaurants are obtained by geolocalizing the training on their coordinates.

We found that this slightly different way to train geolocalized classifiers is important, increasing the accuracy significantly. The reason is not only the increased capability to discriminate between inter-bundle classes, but also a better calibration of the probabilities $p_{k,s}(\mathbf{x})$ across bundles. The overlap of different training neighborhoods also helps probabilities $p_{k,s}(\mathbf{x})$ from different bundles to be more comparable in (5). Although this calibration is not exact, in practice this method achieves good performance for a reasonable $\gamma$.

One drawback of including neighboring restaurants is that the number of classifiers to be trained also increases significantly. Although they are not used during query time, the additional classifiers need to be trained, increasing the overall training time with the CBC approach. We found that in practice it is not necessary to train one model for each additional neighboring class, but training a single model combining them (see Fig. 8b) is much faster and also achieves better accuracy. In the experiments, we verify this hypothesis empirically (see Fig. 13). This additional model can be seen as a rejection model $f_{k,s \notin M_k}(\mathbf{x})$, related with the probability that the dish comes from a neighboring restaurant rather than from the restaurant $k$ itself.

This framework has an acceptable scalability for both restaurants and dishes. If a new restaurant is included in the database it only requires to train a new geolocalized model, and (optionally) retraining the models of its neighbors. In the case of a new dish in the menu, it would require to retrain all the dish models in the menu but only with images from that restaurant and its neighbors. In addition, since fewer models are selected, this scheme is very discriminative for dishes within the same restaurant, and also discriminative for dishes across neighboring restaurants, while simultaneously keeping a low time cost compared with geolocalized pairwise models.

### C. Efficiency and scalability

Another important aspect of the geolocalized modeling framework for practical applications is the complexity. Table I compares the complexity of the different approaches for both training and test stages. We include non-geolocalized models (*global*) as reference. An important difference between the different methods is how they implement multiclass classification. We typically need more pairwise models, but each of them is trained faster, as it uses fewer training samples.

The training cost is critical to address large scale problems. In general, the total cost of training both global and pairwise

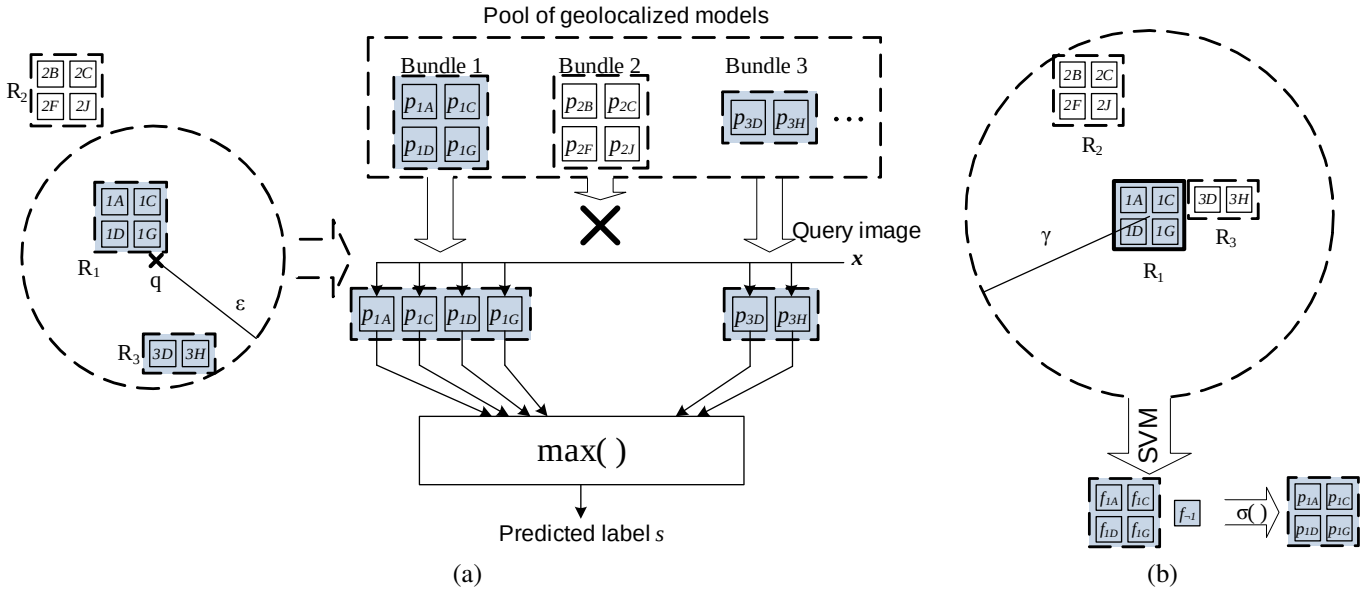



Figure 7. Classification using bundled classifiers: (a) classifier selection and combination, and (b) geolocalized training of a bundle. Notation: $f_{1,A}(\mathbf{x})$ and $p(s = A|k = 1, \mathbf{x})$ are represented as $f_{1A}$ and $p_{1A}$, respectively.

Table I
COMPARISON OF STRATEGIES FOR GEOLOCALIZED CLASSIFICATION. COST IS FOR THE CASE OF SVM CLASSIFIERS.

| Approach | ...selects/discards... | Classifiers | Number of classifiers | Training cost per classifier | Training cost of adding a new Dish | Training cost of adding a new Restaurant | Test cost per image |
|---|---|---|---|---|---|---|---|
| Global | - | OAA | $O\left(K\bar{D}\right)$ | $O\left(K\bar{D}N\right)$ | $O\left(K\bar{D}^2N\right)$ | $O\left(K\bar{D}^2N\right)$ | $O\left(K\bar{D}\right)$ |
| Shortlist | classes | OAA | $O\left(K\bar{D}\right)$ | $O\left(K\bar{D}N\right)$ | $O\left(K\bar{D}^2N\right)$ | $O\left(K\bar{D}^2N\right)$ | $O\left(e\bar{D}\right)$ |
| Geolocalized voting | pairwise classifiers | OAO | $O\left(cK\bar{D}^2\right)$ | $O\left(N\right)$ | $O\left(c\bar{D}N\right)$ | $O\left(c\bar{D}^2N\right)$ | $O\left(e^2\bar{D}^2\right)$ |
| CBC | bundled classifiers | OAA | $O\left(K\bar{D}\right)$ | $O\left(c\bar{D}N\right)$ | $O\left(c\bar{D}^2N\right)$ | $O\left(c\bar{D}^2N\right)$ | $O\left(e\bar{D}\right)$ |

**OAA**: one-against-all, **OAO**: one-against-one (pairwise), $K$: number of restaurants, $\bar{D}$: average number of dishes per restaurant, $c$: average number of nearest neighbor restaurants in the sparse graph construction, $N$: number of images per dish class and per restaurant, $e$ : average number of candidate restaurants.

classifiers is $O\left(K^2\bar{D}^2N\right)$, where $\bar{D}$ is the average number of dishes per restaurant and $N$ the number of training images per dish. When the models are geolocalized (i.e. geolocalized voting and CBC), we only train $c$ nearest restaurants ($c$ is indirectly related with $\gamma$) instead of $K$, which reduces the training cost significantly.

Another practical problem is scalability, i.e., how to update the dish models when the restaurant database is updated with new dishes or restaurants. In principle, global would require retraining again all the models in the database. Similarly, CBC requires training a new model for the particular restaurant, or retraining when a new dish is added. This is significantly faster than global. In this sense, geolocalized pairwise models are more suitable, as they only would need to retrain a smaller subset related with the new dishes, and can reuse all the models already available.

However, the most critical difference between geolocalized voting and CBC is related with the testing stage. Pairwise models have a quadratic dependency on $\bar{D}$ and the average number of candidate restaurants $e$ (indirectly related with $\epsilon$), while the number of models in OAA classifiers grows linearly. This makes CBC more suitable when the number of classes is large and when a low classification delay is required.

Table II
OVERALL STATISTICS OF THE DATASET (6 CITIES).

| Cities | #restaurants | #dishes | | #images | |
|---|---|---|---|---|---|
| | | total | per restaurant | total | per dish |
| Beijing | 187 | 1173 | 6.27 | 45541 | 38.82 |
| Shanghai | 198 | 1253 | 6.33 | 37590 | 30.00 |
| Tianjin | 78 | 435 | 5.58 | 10811 | 24.85 |
| Nanjing | 64 | 328 | 5.13 | 7895 | 24.07 |
| Hangzhou | 62 | 371 | 5.98 | 9124 | 24.59 |
| Guangzhou | 57 | 272 | 4.77 | 6543 | 24.06 |

## V. EXPERIMENTS

### A. Dish images in restaurants dataset

Although some food datasets are available, none of them is restaurant-centric, and thus they do not include neither geolocation nor menu information. For this reason we collected our own data to evaluate the proposed methods in the dish recognition in restaurants scenario. We gathered data about restaurants in six cities from an online restaurant directory and review site[1]. For simplicity, we focus on the data collected from Beijing to illustrate the structure of our dataset and provide more details.

Each restaurant in our dataset includes its location, the list of dishes (i.e. menu) and a number of photos of each dish.

[1]http://www.dianping.com

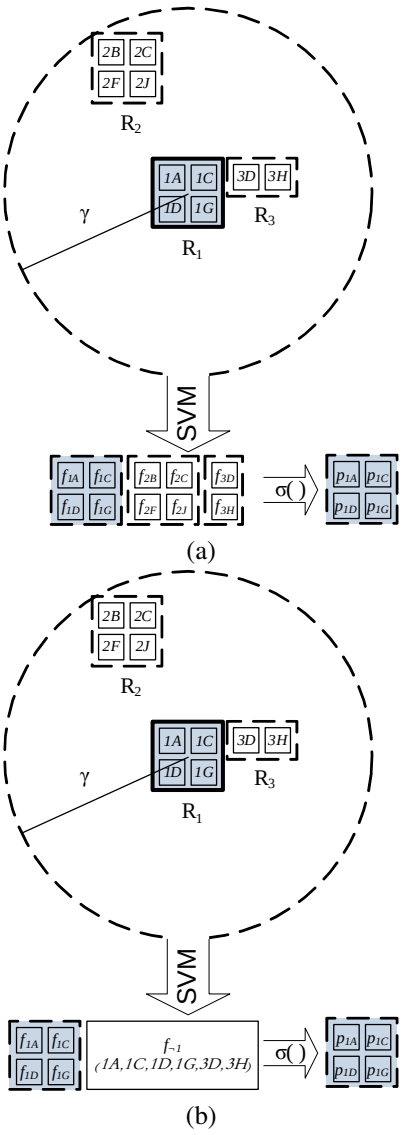



Figure 8. Geolocalized training of a bundle of classifiers with neighboring bundles: (a) each neighboring class independently, (b) all neighboring classes together in a joint rejection model. Notation: $f_{1,A}(\mathbf{x})$, $p(s = A|k = 1, \mathbf{x})$ and $f_{1,s\notin M_1}(\mathbf{x})$ are represented as $f_{1A}$, $p_{1A}$ and $f_{\neg 1}$, respectively.

We discarded restaurants with less than 3 dishes in the menu and fewer than 15 images per dish. Table II shows the overall statistics of the datasets[2]. For instance, focusing on the Beijing dataset, it contains 187 restaurants with a combined 1173 dish classes (701 unique dish classes). Fig. 9 shows some examples of restaurants and dishes in that dataset, illustrating the hierarchical organization of the images. The number of dishes per restaurant and images per dish vary significantly across the dataset (see Fig. 10). The geographical distribution of the restaurants in Beijing is shown in Fig. 11.

### B. Experimental settings

**Features.** Images are represented using the deep convolutional activation feature (DeCAF) proposed in [28]. Deep models have recently been applied to large-scale visual recognition tasks, including food recognition [29]. In our case we have very few images per class to train the deep model, case in which fully-supervised deep architectures tend to overfit dramatically [28]. In order to avoid overfitting, we just rely on deep features learned from a large object recognition dataset (i.e. ImageNet). In this way, DeCAF features can be learned to have sufficient representational power and generalization ability to be applied to new tasks which have too few training examples. Using the same network as in [28] we extract a 40960 dimensional features.

**Geolocation information.** We use the coordinates of the restaurants obtained from a web map service. As test data is collected in the same way, we lack more specific geolocation for the dish images other than the same coordinates of the restaurant. In this case, restaurants are modeled as points, neglecting their actual spatial extension (dish photos could be taken at any particular location within the restaurant). A simple yet more realistic model to evaluate the proposed methods is considering restaurants as square blocks (in our case we use $L \times L = 20 \times 20$ square meters). In addition we model the error of the mobile location service as isotropic Gaussian noise with $\sigma = 50$ meters to obtain the simulated query geolocation. Thus we simulate the location of a test image as

$$(u, v) = (u_R + {}^{L}/_{2} \times n_{U1} + n_{LOC1}, v_R + {}^{L}/_{2} \times n_{U2} + n_{LOC2})$$

where $(u_R, v_R)$ are the coordinates of the restaurant, $n_{U1}$ and $n_{U2}$ are sampled randomly from a uniform distribution $\mathcal{U}(-1, 1)$ and $(n_{G1}, n_{G2})$ from a bidimensional normal distribution $\mathcal{N}\left(0, \sigma^2_{LOC}\mathbf{I}\right)$. An example of simulated query geolocations is shown in Fig. 12, where different restaurants and their corresponding queries are shown in different colors. Note that (unknown) true geolocations lie within the boundaries of the restaurant, while the estimation given by the smartphone is much more scattered due to the noise.

**Binary classifiers.** We implemented the proposed approach for the particular case of SVM classifiers. We used the Liblinear library [30], considering its efficiency for high dimensional features. We randomly split the dataset selecting 10 images per dish class for training, and another 10 for testing (if not enough, the remaining images in the class, with a minimum of 5 images).

**Dish models.** We evaluated the three geolocalized methods described in the paper: *shortlist*, *geolocalized voting* and *CBC*. For the last two methods we use $\gamma = 400$ meters for the training neighborhood.

**Baselines.** We also compared with the global classifier ignoring geolocation (*global*), and with a simple yet representative retrieval-based method (*geolocalized kNN*). The latter first finds the training images in the geographic neighborhood of the query and then uses $k$-NN in the feature space. In Section V-C3 we evaluate this method over SIFT features encoded with a vocabulary tree [31], which is a widely used for landmark recognition.

### C. Recognition accuracy

In this section we first focus on the Beijing dataset for more detailed evaluation. In a later section we compare the

[2]http://vipl.ict.ac.cn/isia/datasets_dish/index.html

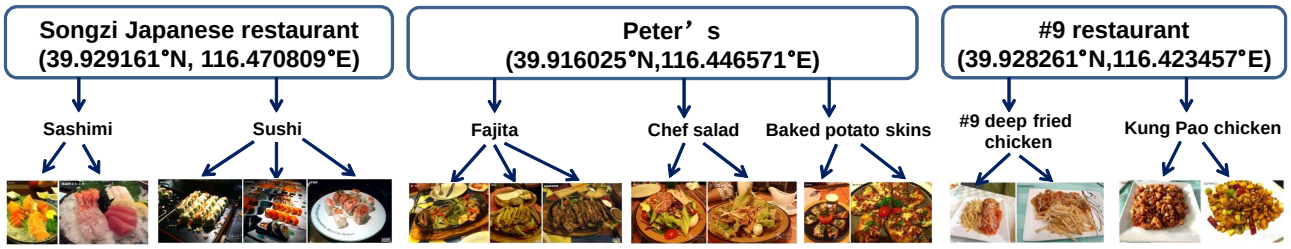



Figure 9. Examples of restaurants and dish photos in the dataset (Beijing). Each restaurant has a menu and a geolocation, and each item in the menu has several images.

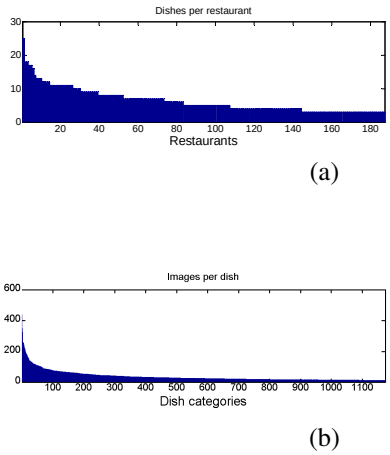



Figure 10. Characteristics of the dataset (Beijing): (a) number of dish classes per restaurant, (b) number of images per dish class.


Figure 11. Geographic distribution of the restaurants included in the experiments (Beijing).

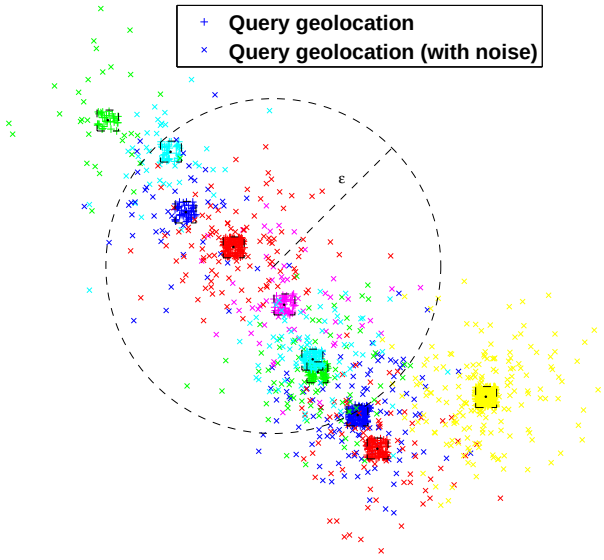


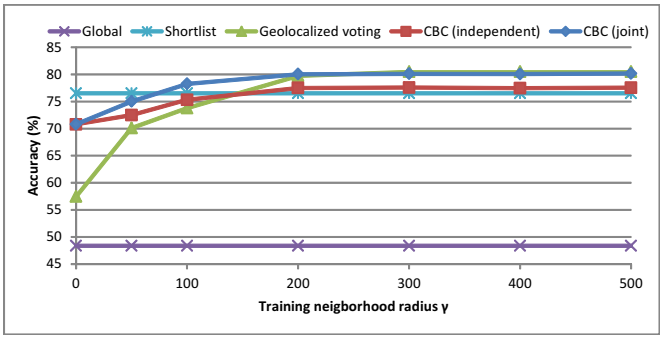

Figure 12. Example of geographical data used in the experiments, including several restaurants and the simulated query geolocations ($\sigma_{LOC} = 50$ meters, $\epsilon = 200$ meters).

performance over the datasets collected from other cities.

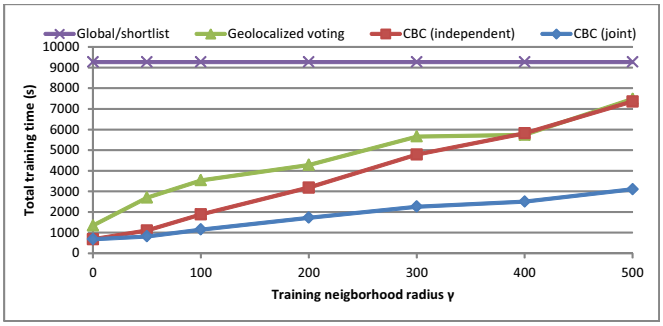
*1) Training radius:* We first evaluate the impact of the training radius $\gamma$ over the accuracy and efficiency, for a fixed $\epsilon = 150$ meters. A value of $\gamma = 0$ corresponds to training classifiers to discriminate only between the dishes in the menu. We see that in this case for both *geolocalized voting* and *CBC* the accuracy is not optimal (see Fig. 13a). The accuracy increases as the training radius increases and includes neighboring restaurants. On the other hand, the more restaurants we include, the more classifiers that need to be trained, so the training time also increases (see Fig. 13b).

For *CBC*, we also compared the two variants to include classes from neighboring restaurants (see Fig. 7): all classes independently, and a joint rejection model. We observed that both accuracy and training time are better when training a joint rejection model instead of many independent ones. For that reason we use this approach for the rest of the experiments.

*2) Search radius:* The performance of the proposed system (both accuracy and efficiency) greatly depends on the number of candidate restaurants found in the geographical neighborhood, which in practice depends on many factors. An important parameter is the search radius $\epsilon$. A low value can ignore the right restaurant, while a too large value can increase the computational cost and reduce the effectivity of geolocalized models. Fig. 14a compares the histograms of candidate restaurants found in the search neighborhood. Bars of different colors correspond to different values of $\epsilon$. The most frequent case is finding only one candidate, and cases with more than five restaurants are very rare. However, a

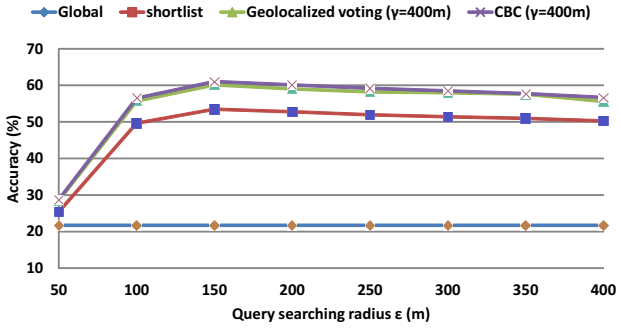



(a)

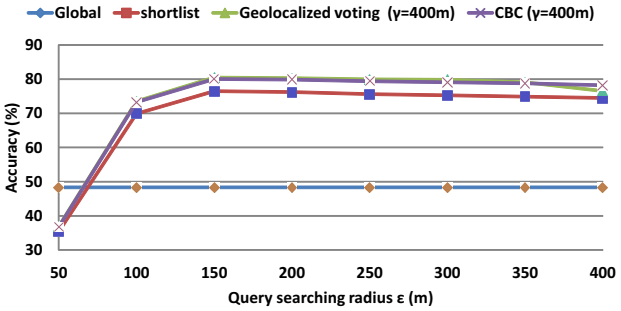



(b)

Figure 13. Accuracy for different training radius $\gamma$: (a) accuracy, and (b) training time.



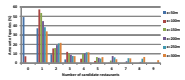

(a)



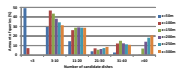

(b)

Figure 15. Accuracy for different search radius $\epsilon$: (a) LLC, and (b) DeCAF.

small value of $\epsilon$, such as 50 meters, results in not finding any candidate restaurant (due to the noise added in our geolocation model) in more than 45% of the times, which will degrade the performance significantly. In most cases, even few restaurants can lead to several tens of dish classes, as shown in Fig. 14b.



(a)



(b)

Figure 14. Distribution of queries according to the number of candidate (a) restaurants and (b) dishes found for different $\epsilon$. Best viewed in color.

Closely related with the number of candidate restaurants, the accuracy in this case follows a similar trend with $\epsilon$ (see Fig. 15). We can notice that if that radius is too small (e.g. 50 meters), the accuracy drops due to the lack of candidate restaurants. In the following experiments, we empirically set $\epsilon = 150$ meters which is a reasonable value for good performance and good efficiency.

*3) Features:* In addition to deep features, here we also evaluate the proposed geolocalized models for other four types of features. We include three BoW representations: kernel descriptors [32], locality-constrained linear coding (LLC) [33] and Fisher vectors [34], encoded with a dictionary of 10K words. We also include SIFT descriptors combined with a vocabulary tree for faster retrieval of local features.

The results are shown in Table III (and in Fig. 15 for LLC and DeCAF). It can be observed that geolocation contributes significantly to the recognition performance, since considering geolocation improves the accuracy of global models by roughly 30%. Among the evaluated features, *geolocalized kNN* has worse performance than *shortlist*, while still performing better than global models not considering geolocation. However, this does not hold for DeCAF, where the accuracy drops showing that kNN in high dimensional spaces can degrade its performance if the metric is not suitable. However, learning metrics in such high dimensional spaces is complex and very costly. In contrast, DeCAF is very powerful when used with SVM, case in which even a global classifier has better performance than *geolocalized kNN*. This also suggests that, in contrast to landmark recognition, discriminative classification may be more suitable for food recognition than retrieval-based classification.

The proposed two approaches have similar performance, consistently improving the accuracy. The best performing feature is DeCAF with *CBC*, with an absolute gain over *shortlist*

Table III
ACCURACY (%) FOR DIFFERENT FEATURES.

| Feature | Dim | Global | Geolocalized $k$NN $k=11$ | Geolocalized $k$NN $k=21$ | Shortlist | Geolocalized voting | CBC |
|---|---|---|---|---|---|---|---|
| SIFT | 11,1K | 7.28 | 17.17 | 20.11 | 38.59 | 44.64 | **45.13** |
| KDES | 42K | 16.78 | 17.36 | 17.39 | 48.30 | 51.42 | **54.29** |
| LLC | 21.5K | 21.72 | 44.36 | 40.78 | 53.45 | 59.88 | **60.92** |
| FishVec | 65.5K | 26.41 | 47.12 | 44.46 | 56.79 | 65.41 | **65.44** |
| DeCAF | 41K | 48.35 | 21.25 | 18.82 | 76.51 | 79.74 | **80.05** |

of 3.5%. The other features benefit more from geolocalized models, with larger absolute gains in accuracy around 5˜8%, with *CBC* having slightly better performance than *geolocalized voting*.

Table IV shows some examples of difficult dish images classified using the proposed methods. In general we observe that global models and geolocalized $k$NN cannot detect the exact label, although sometimes predict similar categories (e.g. dumplings). However, *geolocalized voting* and *CBC* can find the exact categories.

### D. Computational cost

We also evaluated the training and test cost of the different models, and the results are summarized in Table V. A first observation is that the number of support vectors per model in *global* and *shortlist* is much higher than the geolocalized models, leading to overcomplex models in geolocalized queries. Global models are very slow to train, while the proposed two models are much faster.

The key difference between *geolocalized voting* and *CBC* is related with the test stage, and in particular to the way multiclass classification is implemented. The OAA classifiers in *CBC* require very little time during test, and are faster than global models, and *geolocalized kNN*. In contrast, *geolocalized voting* is significantly slower due to a much larger number of pairwise models required to evaluate for each query image. This makes *CBC* more suitable than *geolocalized voting* when low latency is required and computational resources are limited.

### E. Other cities

Fig. 16 shows the performance of our models on the datasets of the 6 cities, using DeCAF as feature. In all the cases the two proposed geolocalized models outperform *global, shortlist* and *geolocalized kNN*. This also shows their effectivity with datasets of different sizes (see Table II). We can also observe that for larger datasets (e.g. Beijing, Shanghai) the gain due to geolocalized classification is larger. In those cases, the number of candidate classes for a query is smaller compared with the total number of classes, and the benefit of geolocalizing the models is higher. Note that future larger datasets (with more restaurants and more dish classes) will benefit more from this effect. The test time for each query image is also shown in Fig. 16b. In general, *CBC* shows a very promising performance, while keeping the complexity low.

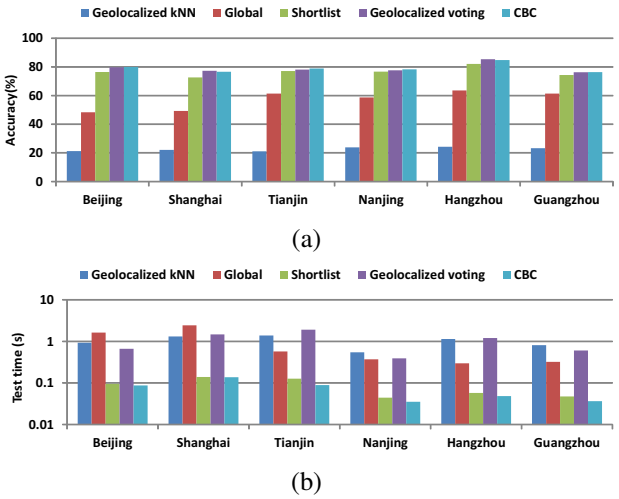



Figure 16. Evaluation in data from different cities (DeCAF): (a) average accuracy, and (b) test time per query.

## VI. CONCLUSIONS

Unrestricted dish recognition is a very challenging problem, and addressing the problem only considering visual information is very hard, even for humans. In general, but particularly in this domain, contextual information can significantly improve performance over visual-only approaches. By exploiting geolocation and user-contributed information about restaurants, we can effectively simplify the problem from thousands to tens of candidate classes. In contrast to most approaches using geolocation for image recognition, we do not use retrieval techniques but explicit discriminative classification. However, we also showed that a naive implementation of this idea (i.e. shortlist) is not optimal due to the mismatch between test and training settings, leading to limitations in both accuracy and efficiency. We analyzed the problem and proposed geolocalized classification to address these limitations. Thus we can achieve better recognition and faster training and prediction, in particular with the proposed CBC strategy.

In this work we focused on close-up photos of dishes. Future work will consider photos with multiple dishes and larger datasets including data from more countries and a wider range of cuisines.

## REFERENCES

[1] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar, "Food recognition using statistics of pairwise local features," in *International Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2249–2256.

[2] Z. Zong, D. T. Nguyen, P. Ogunbona, and W. Li, "On the combination of local texture and global structure for food classification," in *International Symposium on Multimedia*, 2010, pp. 204–211.

[3] F. Kong and J. Tan, "Dietcam: Regular shape food recognition with a camera phone," in *International Conference on Body Sensor Networks*, 2011, pp. 127–132.

[4] Y. Matsuda, H. Hoashi, and K. Yanai, "Recognition of multiple-food images by detecting candidate regions," in *International Conference on Multimedia and Expo*, 2012, pp. 25–30.

[5] Y. Kawano and K. Yanai, "Real-time mobile food recognition system," in *International Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 1–7.

[6] T. Joutou and K. Yanai, "A food image recognition system with multiple kernel learning," in *International Conference on Image Processing*, 2009, pp. 285–288.

[7] H. Hoashi, T. Joutou, and K. Yanai, "Image recognition of 85 food categories by feature fusion," in *International Symposium on Multimedia*, 2010, pp. 296–301.

Table IV
EXAMPLES OF LABELS PREDICTED BY THE DIFFERENT STRATEGIES. CORRECT PREDICTIONS ARE SHOWN IN BOLDFACE.

| Method |  |  |  |  |  |
|---|---|---|---|---|---|
| Ground truth | *Pot Stewed Beef* | *Kanokwan Fried Crab* | *BBQ Pork* | *Seafood Dumpling* | *Cream Caramel* |
| Geolocalized $k$NN | Honey Cake | Rice With Pineapple | Tofu With Pork Ribs | Three Delicacies Dumpling | Milk Tea |
| Global | Gross blood Mong | Crab Tofu | Gross blood Mong | Assorted Dumpling | Mango Pudding |
| Shortlist | **Borsch** | Rice With Pineapple | Gross blood Mong | Assorted Dumpling | Milk Tea |
| Geolocalized voting | **Borsch** | **Kanokwan Fried Crab** | **BBQ Pork** | **Seafood Dumpling** | **Cream Caramel** |
| CBC | **Pot Stewed Beef** | **Kanokwan Fried Crab** | **BBQ Pork** | **Seafood Dumpling** | **Cream Caramel** |

Table V
COMPLEXITY AND TIME COSTS FOR THE DIFFERENT MODELS.

| Method | #models | #SVs /model | Training time (min) | | Test time/image (s) | |
|---|---|---|---|---|---|---|
| | $\gamma = 400$ | $\gamma = 400$ | $\gamma = 0$ | $\gamma = 400$ | $\epsilon = 50$, $\gamma = 400$ | $\epsilon = 150$, $\gamma = 400$ |
| Geolocalized $k$NN ($k = 11$) | - | - | - | - | 0.2738 | 0.9273 |
| Global | 701 | 132.82 | 154.51 | | 1.6285 | |
| Shortlist | 701 | 132.82 | 154.51 | | 0.0435 | 0.0951 |
| Geolocalized voting | 15251 | 16.00 | 22.36 | 93.07 | 0.1508 | 0.6620 |
| CBC | 1314 | 60.41 | **11.34** | **41.73** | **0.0350** | **0.0869** |

Experiments performed in an AMD Optero 3.1 GHz with a memory of 48 GB (using one core).


[8] Y. Maruyama, G. C. de Silva, T. Yamasaki, and K. Aizawa, "Personalization of food image analysis," in *International Conference on Virtual Systems and Multimedia*, 2010, pp. 75–78.

[9] Y. Kawano and K. Yanai, "Foodcam: A real-time mobile food recognition system employing fisher vector," in *International Conference on Multimedia Modeling*, 2014, pp. 369–373.

[10] ——, "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation," in *European Conference on Computer Vision Workshops*, 2014.

[11] X. Qian, X. Liu, C. Zheng, Y. Du, and X. Hou, "Tagging photos using users' vocabularies," *Neurocomputing*, vol. 111, pp. 144–153, Jul. 2013.

[12] X. Qian, H. Feng, G. Zhao, and T. Mei, "Personalized recommendation combining user interest and social circle," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1763–1777, July 2014.

[13] K.-H. Yap, T. Chen, Z. Li, and K. Wu, "A comparative study of mobile-based landmark recognition techniques," *IEEE Intelligent Systems*, vol. 25, no. 1, pp. 48–57, Jan 2010.

[14] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. Reznik, "Mobile visual search: Architectures, technologies, and the emerging mpeg standard," *IEEE MultiMedia*, vol. 18, no. 3, pp. 86–94, March 2011.

[15] Z. Li and K.-H. Yap, "Content and context boosting for mobile landmark recognition," *IEEE Signal Processing Letters*, vol. 19, no. 8, pp. 459–462, Aug 2012.

[16] M. M. Zhang, "Identifying the cuisine of a plate of food," University of California San Diego, Tech. Rep., 2011.

[17] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[18] Y. Matsuda and K. Yanai, "Multiple-food recognition considering co-occurrence employing manifold ranking," in *International Conference on Pattern Recognition*, 2012, pp. 2017–2020.

[19] R. Ji, Y. Gao, W. Liu, X. Xie, Q. Tian, and X. Li, "When location meets social multimedia: A survey on vision-based recognition and mining for geo-social multimedia analytics," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 2, 2013.

[20] D. Chen, G. Baatz, K. Koser, S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk, "City-scale landmark identification on mobile devices," in *International Conference on Computer Vision and Pattern Recognition*, June 2011, pp. 737–744.

[21] K. Yaegashi and K. Yanai, "Can geotags help image recognition?" in *Advances in Image and Video Technology*, 2009, pp. 361–373.

[22] ——, "Geotagged image recognition by combining three different kinds of geolocation features," in *Asian Conference on Computer Vision*, 2010.

[23] Y. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T. Chua, and H. Neven, "Tour the world: building a web-scale landmark recognition engine," in *International Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1085–1092.

[24] J. Luo, D. Joshi, J. Yu, and A. Gallagher, "Geotagging in multimedia and computer vision-a survey," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 187–211, 2011.

[25] J. Li, X. Qian, Y. Y. Tang, L. Yang, and T. Mei, "Gps estimation for places of interest from social users' uploaded photos," *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 2058–2071, 2013.

[26] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[27] J. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers*, 1999, pp. 61–74.

[28] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *International Conference on Machine Learning*, 2014.

[29] Y. Kawano and K. Yanai, "Food image recognition with deep convolutional features," in *UbiComp*, 2014, pp. 589–593.

[30] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[31] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *International Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2161–2168.

[32] L. Bo, X. Ren, and D. Fox, "Kernel descriptors for visual recognition," in *Neural Information Processing Systems*, vol. 1, no. 2, 2010, p. 3.

[33] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *International Conference on Computer Vision and Pattern Recognition*, 2010.

[34] F. Perronnin, J. Sanchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision*, 2010.