Image Representations for Ranking and Classification

Josip Krapac

UNIVERSITÉ DE CAEN BASSE-NORMANDIE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE CAEN BASSE-NORMANDIE

Spécialité : Informatique et Applications

préparée au Laboratoire GREYC

dans le cadre de l'École Doctorale Sciences de Structure, Information, Matière et Matèriaux

présentée et soutenue publiquement

par

Josip Krapac

le 11 juillet 2011

Image Representations for Ranking and Classification

Reprèsentations d'images pour la recherche et la classification d'images

---

Directeurs de thèse : Frédéric Jurie et Jakob Verbeek

JURY

Éric Gaussier	*Université Joseph Fourier*	Examinateur
Tinne Tuytelaars	*Université catholique de Louvain*	Rapporteur
Matthieu Cord	*Université Pierre et Marie Curie 7*	Rapporteur
Jakob Verbeek	*INRIA Grenoble Rhône-Alpes*	Examinateur
Frédéric Jurie	*Universié de Caen Basse-Normandie*	Examinateur

Abstract

This thesis concerns the tasks of image re-ranking and image classification. These tasks are solved by learning statistical models given a representation of visual content of the image and a similarity measure between images. Here we aim to improve performance of the tasks by extending the bag-of-words image representation, while using existing statistical models and similarity measures between images.

We adapt the image representation according to a given task. First we explore the task of *image re-ranking*, whose goal is to re-order the images retrieved by a text query such that images relevant to a query are ranked above non-relavant ones. Inspired by text re-ranking methods we developed a *query-relative* image representation that depends on the visual content of the image, but also on the query used to retrieve it. Next, we adapt the representation for the task of *image classification*, which aims to assign one or more labels to an image that is related to the content of the image. We have adapted the representation by learning a visual vocabulary specifically for the classification task. We have also introduced a new representation that encodes the information about spatial layout of image parts in much more compact manner than currently used representations that encode the spatial layout.

All developed image representations are compact, fast to construct and already perform very good with linear models. We show marked improvements on several standard and challenging datasets with respect to state-of-art-methods. For image classification and image re-ranking tasks we have shown that adapting the representation to the task improves the performance.

Keywords

Image representation • Image re-ranking • Image classification • Visual vocabulary learning • Spatial layout.

Résumé

Cette thèse se concerne avec de tâches de la recherche et la classification d'images. Ces tâches sont résolues par l'apprentissage des modèles statistiques donnée une représentation du contenu visuel de l'image et une mesure de ressemblance entre les images. Ici nous visons à améliorer les performances du tâches en étendant le sac-de-mots représentation de l'image, tout en utilisant modèles statistiques et des mesures de similarité entre les images déjà existants.

Nous adaptons la représentation d'image en fonction d'une tâche donnée. Nous avons d'abord explorer la tâche de *reclassement d'images*, en contexte de la recherche d'images, dont le but est de trier les images récupérées par une requête textuelle afin que les images pertinentes pour ce requête sont classés au-dessus les autres images. Inspiré par le méthodes de reclassement de documents textuelles nous avons développé une représentation qui dépend du contenu visuel de l'image, mais egalement sur la requête textuelle utilisée pour récupérer l'image. Ensuite, nous adaptons la représentation pour la tâche de *classification d'images*, qui vise à attribuer une ou plusieurs étiquettes d'une image liée à la contenu visuel de l'image. Nous avons adaptée de la représentation en apprenant un vocabulaire visuel, spécifiquement pour la tâche de classification. Nous avons également introduit une nouvelle représentation qui encode les informations sur la disposition spatiale des parties d'image, de manière beaucoup plus compacte que les représentations actuellement utilisés pour codage de l'agencement spatial.

Toutes les représentations développées sont compacts, rapides à construire et obtient bons resultats en utilisent des modèles linéaires. Nous montrons des améliorations sur plusieurs bases des images complexes en comparation avec des méthodes de l'état de l'art. Pour les tâches de recherche et classification d'images nous avons montré que l'adaptation de la représentation à la tâche améliore les performances.

Mots-clés

Representation d'image • Recherche d'images • Classification d'images • Apprentisage de vocabulaire visuel • Agencement spatial

Contents

1

Introduction

In Section 1.1 of this chapter we explain our motivation for the work presented in this thesis, which concerns image representations. In Section 1.2 we define our objectives, and in Section 1.3 we give an overview of image representations used for image classification. In Section 1.4 we conclude the chapter with an overview of contributions presented in this thesis. In the remaining chapters we focus on each of the contributions, giving in each chapter an overview the related work and details of our contribution, presenting the results of experimental evaluation along with comments and perspectives. We conclude the thesis in Chapter 5 where we give a brief review of each contribution and comment possible extensions and applications of the proposed methods.

1.1 Motivation

Over the past decade digital photo cameras became ubiquitous, the network bandwidth has increased, and image compression techniques improved. These advances led to an explosion of digital images embedded in web-pages and blogs, and the advent of dedicated photo-sharing sites like Flickr and Picassa created a platform that enabled even more images to be published online. The number of images available online continues to grow with expansion of social networks that allow sharing of images: according to the Time magazine more than 130.000 photos are uploaded each minute on Facebook. In order to allow humans to use this vast and ever increasing collection of images, *e.g.* to search for images containing objects or persons, or to organize them into topics, the images need to be indexed by semantically meaningful terms. But annotating images is tedious task, and although there is a number of ways one can provide textual description of the image[1], the majority of the users still do not annotate the images with semantically meaningful terms.

[1]HTML language provides a way of describing images with text, EXIF meta-data contains information when and where the photo is taken and some photo-sharing services even provide tools to annotate image regions

Semantic indexing of a personal image collection can be performed manually, but indexing of big, web-scale image collection is a significant challenge. One solution is to divide the problem into the small parts that can be solved manually. For example, the ESP Game crowd-sources the problem of semantic indexing of images by presenting it as an on-line game, while Flickr Groups provide an administered way of organizing images into topics. But even under an assumption that humans do a perfect job when semantically annotating images, in the sense that they all agree on annotations for an image, manual effort alone is not enough to index all images available online because the number of images to be annotated increases too rapidly. Annotating a subset of images is not an option because we do not know in advance what images will be searched for. We would like to do the semantic indexing automatically, but there is no simple relation between low level image content as represented in a machine and its description by semantically meaningful terms. [Smeulders *et al.*, 2000] call the absence of this relation the *semantic gap*.

To learn statistical models that relate image content and semantic annotations we need annotated images. Therefore, although manual indexing of images can not provide the solution to the problem, it can help in building statistical models, by providing the annotations for the images which are used to learn the model. Learning the model assumes that the image content is *represented* in a machine. A good image representation should encode all the *relevant* information about the visual content of the image. What information in the image is considered relevant depends on the task. We illustrate this using an example in Figure 1.1. In discriminating between images of cities and beaches color is relevant feature, so using *e.g.* global color histograms as an image representation is a good idea. However, discriminating between images of cows and horses using the same features is a difficult, because color information is not relevant for that task: color is not a discriminating feature of cows and horses, and usually they share the environment in which we encounter them. Therefore using simple features might be good enough for easy tasks, but as complexity of the problem increases the need for more sophisticated image representations arises.

1.2 Objectives

In this thesis we explore different ways of representing image content, with special emphasis on image representations for classification and re-ranking tasks. Next, we define these tasks and briefly describe how to learn models for these tasks, as well as how to perform these tasks given the image representation and the learned model.

Image classification The goal of image classification is to assign an image to one or more sematic categories based on its content. In the binary classification setting the goal is to learn the model of an object that, given the image, answers

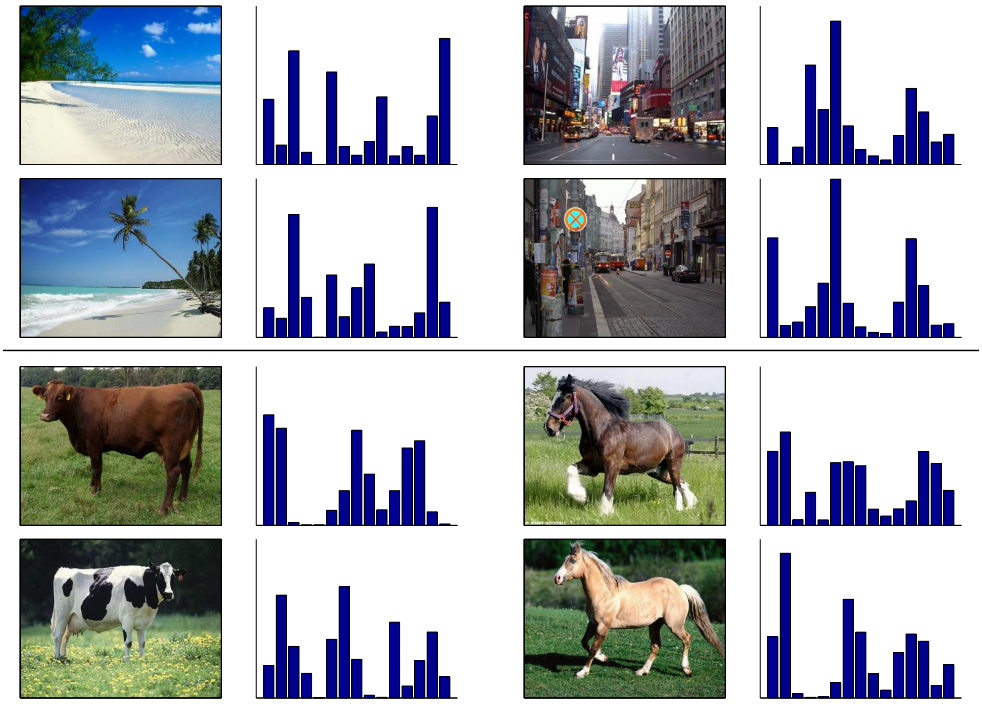

***Figure 1.1**: Representation of image with a global color histogram. Color histogram is a good image representation if we want to discriminate between images of beaches and cities, but not if the task is to discriminate between images of cows and horses.*

the question: "Is the object present in the image?". In more complex, multi-class classification scenario the model is learned to answer the question: "Which object is present in the image?". Image classification with multiple labels, called also *image annotation* answers to the question: "Which objects are present in the image?", by learning multiple binary classification models.

Image re-ranking Given the images retrieved by a text query, using *e.g.* an image web-search engine, the goal of image re-ranking is to sort the retrieved images so that the ones relevant to the query are ranked higher than the ones that are not, using the visual content of the image.

Given an image representation, statistical models that relate the image features to semantically meaningful terms can be learned. The model f is determined by parameters $\mathbf{w}$, so to learn a model means to infer the parameters using which the model well predicts the training labels from the training image features. Given an image $\mathbf{x}$, the learned model generates a score $f(\mathbf{x}, \mathbf{w})$ which can be used either to rank images by sorting the scores, or to classify the images by comparing their scores with a threshold. An example of such models are the support vector machines [Vapnik, 1998], commonly used in state-of-the-art image classification methods. Admittedly,

other models are possible (*e.g.* rule-based models), but since the choice of the model is orthogonal to contributions presented here we use only statistical models recognized to be both simple and efficient.

1.3 Image representations for classification

One of the first attempts to abstract the content of the image was to use textual meta-data associated with images (*e.g.* EXIF meta-data, image file and folder names, HTML tags within web-pages, user-provided tags *etc.*) to represent the images via text meta-data. The majority of image search engines rely on such image representation to retrieve and rank images given a text query. However, as text meta-data is generated often without the intention of annotating the images with semantically meaningful information it can be incorrect, ambiguous, incomplete, or simply missing, so the assumption that text meta-data is related to image content is not always valid. Therefore, to improve the quality of image retrieval and ranking, the visual content of the image must be taken into account.

1.3.1 Global image representations

The first image representations were *global*. These representations aggregate local appearance attributes into image-wide, global color, shape [Jain & Vailaya, 1996] or texture [Manjunath & Ma, 1996] features. They are compact, fast to build and invariant to layout of image parts, but their discriminative power is limited. When aggregating simple pixel features like gradient orientation of image intensity or pixel color, over the whole image into global image representation, like histogram of gradient orientations or color histogram, the influence of each pixel feature on the image feature can be small. This is disadvantageous for some tasks, *e.g.* for detecting if objects like cars or bicycles are present in the image, because the influence of the object features can be drowned by the background clutter. The effect is even more conspicuous when the object in the image is small or occluded. These representations are a good encoding of an image content when individual pixel features are already informative for the task, like in the case of using color to discriminate between images of beaches and cities in Figure 1.1.

1.3.2 Local image representations

To overcome the limitations of global image representations, the image is represented by a set of regions. These representations are therefore called *local*. Since only a

fraction of regions is influenced by the occlusions and background clutter, the local image representations are more robust with respect to these effects.

Here we will concentrate on group of image representations based on bag-of-words approach, first introduced in [Sivic & Zisserman, 2003, Csurka *et al.*, 2004]. Although many object representations can also be used to represent the whole image via *e.g.* histograms of oriented gradients [Dalal & Triggs, 2005], shape segments [Opelt *et al.*, 2006, Ferrari *et al.*, 2010], constellation models [Weber *et al.*, 2000, Fergus *et al.*, 2003], latent SVM [Felzenszwalb *et al.*, 2010], *etc.* we will deal only with bag-of-words image representations as they are very efficient and have been the state-of-the-art in image classification from their introduction. We next describe the creation of local image representations based on the bag-of-words approach.

The local image representations can be described by a succession following stages: region selection, region appearance description, region appearance coding, and derivation of the image features from the set of region appearance codes by spatial pooling. The schematic overview of these stages is given in Figure 1.2. We now describe each of these stage in more detail.

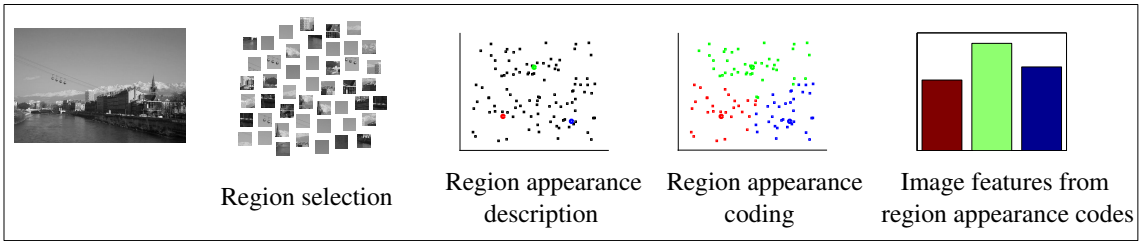

***Figure 1.2**: Shematic overview of stages for deriving local image representation on example of bag-of-words image representation*

Region selection The question of how to select regions from the image is important, since the region selection influences the image representation: different sets of regions yield different image representations. One of the first works that uses local image representation, employs an image segmentation algorithm to select image regions [Barnard *et al.*, 2003]. That option is attractive because it allows each region to be described with its shape [Belongie *et al.*, 2002], in addition to color and texture. However, since few segmentation algorithms produce stable regions[2], and are computationally intensive, this option has not gained a lot of attention.

The idea of using interest points detectors to select regions was borrowed from the field of wide-baseline stereo matching. Interest points detectors are constructed to

[2]In the sense that small change in image can cause big change in the number of regions, and their shape.

detect structures like corners [Harris & Stephens, 1988], blobs [Lindeberg, 1998] and ridges [Steger, 1998]. The rectangular region around detected point (also called *patch*) is represented by a region appearance descriptor. The main advantage when using interest point detectors is that structures are detected at their intrinsic scale using the principle of automatic scale selection [Lindeberg, 1998]. This allows description of regions by the local features that are invariant to local affine transformations [Mikolajczyk & Schmid, 2004]. The use of interest points detectors implicitly assumes that some image regions are not important for image representation, an assumption that seems intuitively valid. However, since they are not learned for the task at hand, but were developed for wide-base stereo matching, there is no guarantee that the selected regions will yield image representation that is optimal for solving a given image classification or re-ranking task.

This drawback of interest point detectors is the main motivation behind dense region sampling: if the image is completely covered by regions, the whole image can be reconstructed from the set of selected regions, and therefore no information is lost. The image representation then encodes the complete image content, and determination of importance of each region is left to the subsequent, task-specific stages. It has been shown that for some image classification tasks selecting regions from a regular grid, outperforms the use of interest point detectors [Winn *et al.*, 2005]. [Nowak *et al.*, 2006] sample the regions randomly from fixed spatial distribution, and in [Moosmann *et al.*, 2008] this approach is extended by dynamically updating the spatial distribution from which regions are sampled.

Region appearance description The appearance of a selected region is described by a set of region appearance attributes, called *local descriptor* or *local feature*. These descriptors can be similar to the global image descriptors, but they are usually handcrafted to have some geometric and photometric invariances, *e.g.* to be invariant to non-uniform illumination changes and geometric distortions caused by varying object pose. To achieve photometric invariance the response of filter banks which describe local texture has been used [Leung & Malik, 1999, Ojala *et al.*, 2002]. The invariance to geometric distortions is additionally addressed using spatial binning as in SIFT [Lowe, 2004], spatial blurring [Berg & Malik, 2001] or by projecting spatial distribution of filter responses to a subspace as in PCA-SIFT [Ke & Sukthankar, 2004]. Region color [van de Weijer *et al.*, 2007] and grayscale values [Obdrzalek & Matas, 2002, Kadir & Brady, 2001] have also been used to describe region appearances.

Region appearance coding The region appearance descriptors are coded using a set of prototype vectors. A prototype is called *visual word*, and the set of visual words used to code the region descriptors is called *visual dictionary*. The visual dictionary is used to express the content of the image, just like the text words are used to convey

the content of text document. However, when dealing with coding of image content, the visual words are not given, but need to be determined. Therefore, two important questions need to be answered: how to obtain a visual dictionary, and how to code a feature using a given visual dictionary. The methods can be divided in two groups, based on the approaches used to answer these questions. The ones that derive visual dictionary and code the features having in mind the reconstruction of the descriptor as a goal, are called here *reconstruction-guided coding methods*. *Prediction-guided coding methods* aim at prediction of the region label from the region descriptor.

Reconstruction-guided methods code the descriptor so it can be reconstructed with minimal error using a given visual dictionary. Early approaches obtained visual dictionaries by k-means clustering of training set of local features and coded the region by the index of the closest visual word [Sivic & Zisserman, 2003, Csurka *et al.*, 2004]. [Nistér & Stewénius, 2006] use hierarchical k-means to reduce the computational complexity of coding each region from $O(N)$ to $O(log(N))$, where N is the number of visual words in dictionary. When using k-means to quantize the descriptor space the quantization cells adapt to the distribution of training descriptors. This is sub-optimal when this distribution is non-uniform, because less frequent features are have bigger reconstruction error. To overcome this [Jurie & Triggs, 2005] use mean-shift clustering algorithm [Comaniciu *et al.*, 2000] to obtain visual words.

As shown in [Boiman *et al.*, 2008], the main drawback of these approaches is that each feature is coded by a single reconstruction coefficient: an index of closest visual words. In that case the reconstruction error is big unless a large number of visual words is used. To reduce the reconstruction error [Philbin *et al.*, 2008] and [van Gemert *et al.*, 2010a] soft-assign a feature to multiple visual words, so the appearance of each region is coded by multiple reconstruction coefficients. The same idea led [Yang *et al.*, 2009, Wang *et al.*, 2010] and [Boureau *et al.*, 2010a] to code the feature with coefficients of linear combination of visual words for which the reconstruction error is minimized, under sparsity [Yang *et al.*, 2009] or locality constraints [Wang *et al.*, 2010]. In the Fisher vector approach of [Perronnin & Dance, 2007] each feature is coded using the gradients with respect to the parameters of underlying generative model, specifically, mixture-of-Gaussians whose parameters are learned by EM algorithm on training set of local features.

In prediction-guided coding the visual dictionary construction is guided by minimizing the prediction of class label associated with the feature. To that end [Moosmann *et al.*, 2008] learn the randomized forests to predict class labels from labeled set of region descriptors. The label of each training region descriptor is inherited from the label of the image the region is sampled from. The growing of the randomized tree is guided by minimization of region mis-classification error. The leaves of the tree are visual words that correspond to the quantization cells of descriptor space. Several randomized trees are used, so each feature is coded by multiple coefficients, where each corresponds to the index of the leaf in the tree. In [Perronnin, 2008] the initial

dictionary is constructed with a goal of feature reconstruction, and then the obtained visual words are adapted using the class labels, so the adapted visual words depend on the class. The feature is coded using both initial, universal dictionary and the class-specific one.

Derivation of image features from the set region appearance codes by spatial pooling The majority of models used for image classification assume that image is represented by a vector space equipped with a dot product. To embed the set of features into a vector space, the set of coefficients that code region appearance features is aggregated into a vector of fixed size that represents the visual content of the image. The aggregation is performed per visual word, so that the coefficients corresponding to a visual word are aggregated over all regions sampled from the image. The case when image representation is derived as a sum of feature coefficients is called *sum aggregation*. Sum aggregation is performed by bag-of-words methods of [Sivic & Zisserman, 2003, Csurka *et al.*, 2004, Nistér & Stewénius, 2006, Jurie & Triggs, 2005] and in the Fisher vector approach of [Perronnin & Dance, 2007]. In *max aggregation* the image features are derived as maximum of feature coefficients corresponding a visual word. *Max aggregation* was introduced by [Riesenhuber & Poggio, 1999], whose goal was to model the responses of cells in visual cortex of primates. [Boureau *et al.*, 2010b] showed that this kind of aggregation better separates the image features corresponding to visual words that have low probability of being active, and that aggregating all features is sub-optimal. Max aggregation is used also by [Yang *et al.*, 2009] and [Wang *et al.*, 2010], and in [Moosmann *et al.*, 2007] where binary vectors are used to describe the image.

The image representations are finally normalized, such that all image vectors have equal L_1 or L_2 norm. When using bag-of-words methods L_1 normalization is used so image representation can be interpreted as multinomial over visual words called *bag-of-words histogram*.

1.3.3 Image layout

For some tasks the layout of the image parts is an informative feature, *e.g.* for scene classification where task is to discriminate between the types of scenes like “bedroom” or “store”. In that case the layout needs to be coded into the image representation. The dominant approach is the Spatial Pyramid Matching (SPM) approach of [Lazebnik *et al.*, 2006]. The image plane divided into the several parts, *e.g.* by quad-tree, and the feature coefficients are aggregated over these parts. The final image representation is obtained by concatenating the representation of the image parts. The same idea was used in [Perronnin *et al.*, 2010b] to enrich the Fisher vector representation with spatial layout information and in [Bosch *et al.*, 2007b] where quantized

gradient orientations are aggregated over image parts. The relative importance of the parts depends on the task, and [Bosch *et al.*, 2007b] explored learning the per-task weights for the parts.

The GIST descriptor of [Oliva & Torralba, 2001] is a global image representation that takes into account the image layout by characterizing image properties like "naturalness" or "openness", measured by spatial distribution of specifically designed image filter responses.

1.4 Contributions

Here we briefly describe the main contributions we present in the following chapters of this thesis.

In Chapter 2 we describe the image representation developed for the task of image re-ranking, where the goal is to to sort image retrieved by the text query taking into account their visual content. The developed image representation depends not only on the visual content of the retrieved image, but also on the content of other images retrieved along with it, using the same text query. This enables us to learn a single relevance model that, once learned, can be used to re-rank the images according to the relevance to queries not seen during model training. The relevance models learned from *query-relative* image representation perform better than *query-specific models* [Schroff *et al.*, 2007] that are trained for each query.

The majority of the methods for image classification and ranking rely on image representations that are learned in an unsupervised manner. In Chapter 3 we describe an image representation that is tailored for classification of images belonging to a specific class. We incrementally build visual dictionary and update the classification model by alternating between growing a clustering tree whose leaves correspond to visual words, and learning linear classification model. As opposed to [Moosmann *et al.*, 2008] who builds the random forest to minimize region mis-classification, we build the tree to minimize image mis-classification. The resulting image representation is very compact, fast to create and it gives excellent performance using linear classifiers which are fast to compute and have limited memory requirements.

State-of-the-art methods for image classification either ignore the layout of image regions to describe the image as in *e.g.* bag-of-words representation, or fix layout of parts and encode the appearance of each part, as in spatial pyramid of [Lazebnik *et al.*, 2006], where a quad-tree was used to fix the layout of image parts. In Chapter 4 we propose a new method that is more flexible, by using the Fisher vector principle to code the layout of image regions attributed to visual words. The resulting representation is much more compact, while achieving the same or better level of performance compared to using spatial pyramids to code the fixed spatial layout.

2

Query-relative Features for Image Re-ranking

Contents

2.1 Introduction

In this chapter we address the problem of re-ranking images retrieved by a web search engine, given the textual query. This work has been published in [Krapac *et al.*, 2010] and submitted to IEEE Transactions on PAMI.

Automatically finding images relevant to a textual query remains a very challenging task. To reduce the computational cost, most current commercial image web search engines rank images based on how the text associated with the images relates to the search query used, without taking into account the contents of the images.

Recently, significant effort has been invested in improving image search performance by taking into account visual information as well as text [Ben-Haim *et al.*, 2006, Berg & Forsyth, 2006, Fergus *et al.*, 2005, Fritz & Schiele, 2008, Morsillo *et al.*, 2009, Schroff *et al.*, 2007]. Most existing methods rely on a common framework: first a textual query is used to retrieve a noisy set of images using a text-based search engine (in some approaches, this initial set of images is filtered by removing drawings and other non-photographic images), and then a classifier, specific to the given query, is

learned from this image set, and used to re-rank the images. Although some promising results have been obtained, these methods share the disadvantage that a separate image re-ranking model must be learned for each query, either on-line or off-line. The computation time required, and the large number of possible queries, make this unsuitable for web-scale image search applications.

In this chapter we present an image re-ranking method, based on textual and visual features, that does not require learning a separate model for every new query. The model parameters are shared across all queries and learned once, as opposed to learned per-query. Our approach is inspired by text-based image search techniques [Frankel *et al.*, 1997, Lin *et al.*, 2003]. These methods compute a relevance score for an image by weighting various meta-data fields where the query terms can appear, such as the web page title and image filename. For a particular query, each image can be represented with a binary query-relative feature vector. Each feature codes for absence/presence in a particular text field, and its meaning is query-relative since the value of each feature depends on the match between the text field and the query terms. Query-relative text features might include, *e.g.* a single 'search term' feature shared across queries, which could be used to represent the occurrence of the word 'elephant' if we are searching for items relevant to the search term 'elephant', or 'giraffe' if we are searching for 'giraffe'.

If we wish to take into account the image content, the situation is more complex than when using text meta-data alone, because image content and textual query terms can not be matched directly. We overcome this problem by introducing query-relative visual features as follows. The training stage makes use of sets of images retrieved by a text-based image search engine for a set of queries, annotated with ground-truth information about whether each image is relevant to the query terms for which it was retrieved. Each image's visual content is represented by a histogram of visual word counts. The query-relative visual features for an image are computed with respect to the visual contents of all images retrieved by the text query. To obtain query-relative visual features we compute the mean visual word histogram for all images retrieved by text-based search for a given query, and group the visual words by how frequently they occur for this query compared to their overall frequency. We can then create a new image representation for each image retrieved by the query in terms of query-relative visual features, where for example one feature represents the visual words which occur most often for this query compared to other queries.

After computing the new representation for each image, it becomes possible to train a universal classifier, common to all search queries, using feature vectors obtained by concatenating the query-relative text and image features. The learned classifier can be used to re-rank images for new queries without additional training, because while the exact image or text content represented by each feature depends on the search query for which it was computed, a feature's meaning in relation to the query relevance is always the same.

Until now there has been no large public data set appropriate for evaluating image re-ranking methods' performance on the images found by text-based search engines. The data sets that exist [Fergus *et al.*, 2004, Li *et al.*, 2007, Schroff *et al.*, 2007] contain images for only a few classes and/or lack meta-data associated with the images. Therefore, we introduce a large public data set of 353 search queries [1] for which we provide the top-ranked images returned by an image search-engine along with the associated meta-data. The images retrieved by the search engine are embedded into web-pages, so the meta-data corresponds to parts of HTML code of the web-page: the text surrounding the image HTML tag on the web page, image filename, URLs, page titles, *etc.*. Each image has a ground-truth relevance label, indicating whether or not it is relevant to the query.

In the remainder of this chapter we give an overview of related work in Section 2.2. In Section 2.3 we present our method for representing text and images by query-relative features. Section 2.4 describes experiments performed on the new data set, giving an experimental validation of our approach and comparison to query-specific classifiers, while in Section 2.5 we summarize the contributions and comment on perspectives.

2.2 Related work

Web image search can be seen as a specific application for the more general problems of image classification and object detection. While the best classification and detection results are obtained by training class-specific models on clean, manually curated data (*e.g.* [Everingham *et al.*, 2009]), in recent years there has been much interest in making use of the larger quantities of noisy data available on the web. The tags applied to images on photo-sharing websites such as Flickr [Li *et al.*, 2009, Wnuk & Soatto, 2008, Wang *et al.*, 2009, Perronnin *et al.*, 2010b] or the text associated with images on general web pages, can be treated as noisy image annotations. Besides providing more data for specific classes of interest, this makes it possible to deal with classes for which no clean annotated data is available, to answer unrestricted search queries submitted by system users. A popular approach is to use an existing text-based search engine to gather hundreds to thousands of images which are potentially relevant to a query, then filter the results taking into account the image contents. The initial textual retrieval can quickly locate images whose meta-data references the query terms using pre-computed indexes, while the additional computational cost of taking into account visual information can be restricted to the classifier used for re-ranking in the second stage [Fergus *et al.*, 2004, Li *et al.*, 2007, Fergus *et al.*, 2005, Berg & Forsyth, 2006, Jing & Baluja, 2008]. The main challenge in learning a classifier for re-ranking is to cope with irrelevant images in the image set used to learn

[1] Available at: `http://lear.inrialpes.fr/~krapac/webqueries/webqueries.html`

the classifier, becuase it is silently assumed that the classifier learning is sufficiently robust to noise in annotations. Once learned, the system needs to predict, from the distribution of the noisy set of images, which visual information corresponds to the search query, and therefore which images should be ranked higher based on its occurrence. This is similar to outlier detection, but in the current setting the majority of the retrieved images may be outliers, and the inliers can be diverse: for example, the relevant images for a query "New York" may contain images of the Statue of Liberty but also ones of Times Square. The filtering method must determine which images are more likely to belong to the class of interest, based on the overall distribution of the contents of images and their meta-data.

Several approaches have been developed using generative models to re-rank the images returned from a search engine. Constellation models are used in [Fergus *et al.*, 2004], where an initial model is trained on all images and RANSAC is used to iteratively remove outliers. In [Yanai & Barnard, 2005] Gaussian mixture models are trained on features of image regions as well as LSI vectors coding the words surrounding the images. The model is refined using EM and used to determine which regions of the images retrieved for the query are most likely to correspond to the query object. Topic models such as PLSA [Hofmann, 2001] and LDA [Blei *et al.*, 2003] have also been used by several authors. In [Fergus *et al.*, 2005], PLSA models are used as well as variants that encode some of the spatial layout of objects. Ranking of images is done on the basis of one of the learnt topics, selected using the topic mixing proportions of the first few images returned by the search engine. Hierarchical Dirichlet processes, an extension of LDA that does not require the number of topics to be fixed, were used in [Li *et al.*, 2007]. The model is learned in an incremental manner, where at each iteration several images positively classified by the current model are added to the training set. Topic models are also used in [Fritz & Schiele, 2008], but are applied on a lower level by modeling gradient orientations in a spatial grid over the image. Images are represented by the estimated topic mixing proportions. To rank the images k-means is applied to the mixing weight representation, and images in bigger clusters are ranked higher. A clustering approach was also proposed in [Ben-Haim *et al.*, 2006], where images are first segmented into coherent regions and clustered based on HSV histograms using mean shift.The largest cluster found is assumed to contain the object of interest, and images are ranked by their distance to the center of this cluster.

A semi-supervised method using LDA was used in [Berg & Forsyth, 2006] to retrieve images of animals from the web. They learned an LDA model for the 100 word context around images found using Google, and manually selected the topics that correspond to the query. In a second step local image features are compared between images, and scores are determined by matches to images associated with good topics. The final ranking of images is based on a sum of scores obtained from visual feature matching and the LDA model. A different semi-supervised learning approach was

taken in [Morsillo *et al.*, 2009]. Here a hybrid generative-discriminative model is learned based on a small set of positive images provided by a user. A discriminative model predicts the image relevance given its visual word histogram. Given the image relevance, a generative model is defined over binary variables that code the presence of the query term in the image filename and URL, and in text near the image on the web page.

Discriminative classification was used in [Schroff *et al.*, 2011], where images are filtered in a three-step process. First, a visual classifier is applied to remove drawings, logos, etc. Next, a generative model over textual features is used to rank the images. They use simple meta-data features defined relative to the search query – for example, does the search query appear in the web page title? – to decide an initial image ranking. The model used for this initial ranking can be trained from whatever annotated data is available, then used on new queries without additional training. In the last step, a support vector machine classifier is trained on the visual content of the images, treating the images that are ranked high based on the textual metadata as positive examples, and a random set of other images as negative examples. The final image ranking is determined by the SVM classification scores.

The query-relative features we describe below are a way to enable transfer learning between annotated image class data and previously unseen classes. The closest related works in that respect are the ones of [Li *et al.*, 2006, Tommasi & Caputo, 2009], where learning procedure takes into account the knowledge about already learned categories when learning the model for the new category. In [Quattoni *et al.*, 2007] a large number of weakly-labelled images is used to derive compact semantic image representation, which can be used to learn new concepts using very small number of training examples. Nevertheless, all these approaches require at least one labeled training example per class to learn the class model.

We note that all this related work is based on learning specific models for each query, except for the text-based models used in [Frankel *et al.*, 1997, Schroff *et al.*, 2007] which are trained on a set of queries and applied to others. In our work we explore the approach of learning query-independent classifiers, but applied to visual features, as well as to more complex textual features. w

2.3 Query-relative features

We wish to train a classifier to re-rank the images returned by a text-based search engine. To *re-rank images* means to filter search engine results taking into account the image contents as well. In this section we describe how to construct 'query-relative' text and image features which enable us to train a single, generic classifier which can be used for re-ranking the images retrieved by different queries.

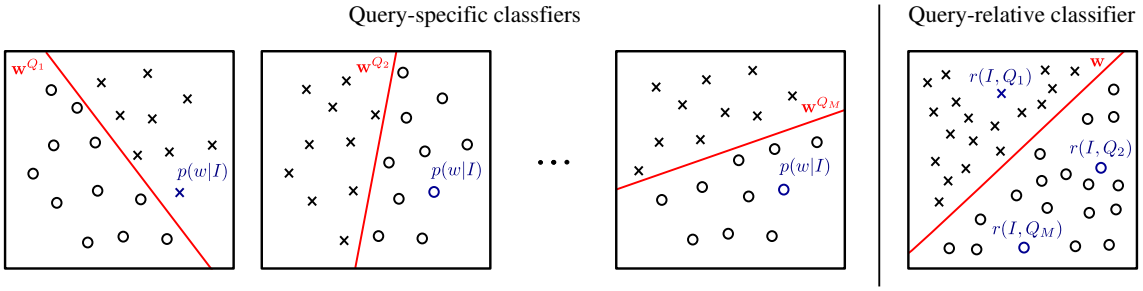

***Figure 2.1**: Query-independent image representations, e.g. bag-of-words histograms $p(w|I)$, require that separate, query-specific classifier $\mathbf{w}^Q$ is learned for each query Q. Representing image by query-relative features $r(I,Q)$, which take into account both the contents of the image I and the query Q used to retrieve them, we can learn a single, query-relative model $\mathbf{w}$ which does not depend on the query and therefore generalizes to new queries.*

The query-relative features for a retrieved document depend on statistical properties of the document I, but also on the statistical properties of the query Q. In the remainder of this chapter we will refer the set of images retreived by the textual query as the *query set*. In Figure 2.1 we demonstrate the advantages of query-relative: when using query-independent image representation the query-specific information is captured by the learned relevance model $\mathbf{w}^Q$ specific to the query, but when using a query-relative representation the query-specific information is used to *modify* the query-independent image content representation, which allows learning of the general, query-independent relevance model $\mathbf{w}$. Therefore the same document, retrieved by three different queries has three different query-relative representations. By their construction query-relative features characterize statistical properties which are shared across queries. This allows learning of single classifier which captures the relation between the query-relative features and relevance of the document for the query terms, using data annotated with relevance labels. The classifier can then be used to derive relevance scores for new queries, where the query-relative features for documents retrieved by new query are decided from the noisy set of retrieved documents.

In our experiments, all supervised learning is based on these query-relative features which are extracted in a fast, unsupervised, manner. This allows us to learn a generic model once, using whatever annotated data is available, and then use it to make relevance predictions for new queries.

Next, we describe the query-relative text features and introduce query-relative text *context features* which enable us to define query-relative visual features. We then show how to construct query-relative features by establishing the correspondence between different queries, either by sorting, or by binning the query-independent features in a

query-dependent way. We compare these two approaches to construct query-relative features and elaborate on several improvements of the basic approach.

2.3.1 Textual features

Our basic query-relative text features follow [Frankel *et al.*, 1997, Schroff *et al.*, 2007]. Nine binary features indicate the presence or absence of the query terms in various fields associated with the image: the text surrounding the image (10 words before and 10 words after), the image alternative text, the web-page title, the hostname, directory, and filename of the web-page that contains the link to the image, and the hostname, directory, and filename of the file that contains the image.

These base features are first supplemented with nine partial match features, each of which is active if some of the query terms, but not all, are present in the relevant field.

We further add context features which represent in more detail the text related to the image, beyond simple presence or absence of the query terms. We divide the image textual annotation in three parts: the text surrounding the image, the image alternative text, and words in the web-page title. For each of the three parts of text annotation we define contextual features by computing word histograms using all the images in the query set. Two words are considered the same if their lower-cased forms match, after removing punctuation. We ignore the query terms, as well as words included in a stop word list.

To create contextual features, we first calculate the normalized word histogram for each query Q, which we can view as a distribution $P(w|Q)$. Next, this query-specific set of words is used to first *establish the correspondence* between query-independent features from different queries (*e.g.* between the words that describe image content in different queries), and then to *transform them into query-relative features*. The correspondence between query-independent features can be performed by sorting the words by their frequency, which can be seen as query-specific feature selection. Table 2.1 shows the context words with the highest $P(w|Q)$ for some example search queries. The other way to establish correspondence between query-independent features is to group words by their relative frequency in the query, by binning $P(w|Q)$. The words that fall into the same histogram bin have the same meaning relative to the query.

Establishing correspondence by sorting In Table 2.1 are some examples of most frequent words, sorted by frequency. The first feature is active for an image if the most common word in the query set context appears in that image's context, the second feature relates to the presence of the second most common context word, and so on.

Search query	Ten most frequent context words
Arc de Triomphe	paris, hotel, etoile, champs, elysees, france, napoleon, carrousel, night, eiffel
avion de ligne	747, a320, boeing, airbus, 400, aircraft, 200, flight, air, airlines
Big Ben	london, parliament, westminster, tower, clock, houses, time, england, uk, world
Musée d'Orsay	orsay, musee, paris, louvre, museum, france, day, rue, monet, seine
lincoln memorial	washington, abraham, war, dc, president, monument, korean, jefferson, world, national
Logo Manchester united	club, football, kingdom, fc, uk, england, supporters, fa, website, item
Machu Picchu	peru, inca, city, trail, cuzco, travel, cusco, ruins, day, lost
mickey	disney, mouse, walt, world, minnie, price, friends, treasures, see, magic
paul mc cartney	mccartney, john, lennon, concert, creation, beatles, dvd, george, music, chaos
piazza san marco	venice, italy, st, square, basilica, venezia, mark, campanile, di, tower

***Table 2.1**: Ten words with highest $P(w|Q)$, in order of decreasing probability, for some example search queries from [Krapac et al., 2010].*

Given the histogram of textual word counts $\mathbf{t}_i$ for each image i, we sort the words indexed by k such that

$$\sum_i t_{i,k} \geq \sum_i t_{i,(k+1)}. \tag{2.1}$$

The ordered histogram of word counts is used to define context features, where the kth binary feature represents the presence or absence of the kth most common word in this source of context for this query:

$$T_{i,k} = \begin{cases} 1 & \text{if } t_{i,k} \geq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

The context features which are sorted earlier are more informative for classification than the later ones, so we can trim these features down to the first N elements without a large loss in performance.

Establishing correspondence by binning The sort operation is computationally expensive for high-dimensional input features, and has to be run on the test data histogram when answering search queries, not only on the training data when learning the model weights. We can reduce the computational complexity by grouping visual words, by binning $p(w|Q)$ instead of sorting it. We assign each word w_k to one of N bins, by comparing the word's probability $P(w_k|Q)$ with bin thresholds $b_1 \dots b_{N-1}$:

$$B_k = \begin{cases} 1 & \text{if } P(w_k|Q) <= b_1 \\ 2 & \text{if } b_1 < P(w_k|Q) <= b_2 \\ \dots & \\ N & \text{if } b_{N-1} < P(w_k|Q) \end{cases}. \tag{2.3}$$

Once we have determined which bin each word that appears in the query data's context is assigned to, we calculate the bin value $\beta_{i,j}$ for each bin j for each image i's context, as the sum of the word counts $t_{i,k}$ for all words k assigned to that bin:

$$\beta_{i,j} = \sum_k \delta_{j,B_k} t_{i,k}, \tag{2.4}$$

where $\delta_{B_k,j}$ is one when $B_k = j$ and zero otherwise. We can do this separately for the image's alternative text, the web page title, and the short context text fragment supplied for each image, or we can merge these three sources of text.

These contextual features can be understood as a form of pseudo-relevance feedback, used in [Lavrenko & Croft, 2001, Lin *et al.*, 2003]. Taking the base features, partial match features, and N contextual features for each text field, we have $9 + 9 + 3N$ features.

2.3.2 Visual features

For visual data we cannot directly construct binary features for presence or absence of the query terms, like the base text features, as there is no direct correspondence between query terms and image appearance. However, as with the contextual text features, we can find which visual words are strongly associated with the query set, and define a set of visual features to represent their presence or absence in a given image.

Our visual features follow the same query-relative design as the contextual text features. We first create a visual word histogram representation for the visual content of each image. Within a search engine system, the visual word histograms can be computed once, when the image is first crawled.

After calculating normalized visual word histograms for all images, we begin computing query-relative visual features by calculating the mean visual word histogram over the images in the query set, which we can view as the distribution $P(w|Q)$. We also compute the equivalent normalized visual word histogram for a large training set of generic images; if we take a good sample of images for different possible search queries, we can view this as the prior distribution over visual words, $P(w)$.

Like words in text, different visual words do not have uniform prior probability. For example, visual words which correspond to textureless image regions or to simple image gradients appear more often, while some visual words that relate to specific image structure are quite rare. We therefore either sort visual words or assign them to bins not according to $P(w_k|Q)$ but according to the ratio between $P(w_k|Q)$ and $P(w_k)$:

$$\rho(Q)_k = \frac{P(w_k|Q)}{P(w_k)} \propto P(Q|w_k), \text{ for fixed } Q. \tag{2.5}$$

We did not use the same ratio for text, because the text features in the data sets used give only a few words of textual context, and generally very short phrases for the web page title and image alternative text, so the word histogram of the meta-data for each image is extremely sparse. Meanwhile, for text, ignoring the words on a stop word list was an easy and fast way to discard uninformative features. For visual words, we do not have a stop word list, but discounting the word frequencies by $P(w)$ plays a similar role.

Binning the features from (2.5) we expect to learn larger weights for bins corresponding to visual words which are unusually common or unusually rare within the images for a query. For text, the value for each bin was the sum of the word counts for all words assigned to that bin. The count for a word in the meta-data histogram for an image was usually zero or one, and the histograms were extremely sparse. Since we have richer histograms for the visual content, we do not use the raw counts but $r_{i,k}$:

$$r_{i,k} = \frac{P(w_k|I_i)}{P(w_k|Q)}. \tag{2.6}$$

This converts query-independent features (visual word histograms) to query-relative features using query-specific information (query mean histogram), so that feature values calculated for different queries have the same meaning. This can be seen as query-specific normalization.

Sorting visual words gives us an ordered set of query-relative visual features, where the kth feature relates to the visual word kth-most related to this query. Using this ordering of the visual features, we compared three ways of representing each visual word's presence or absence: the visual word's normalized count for this image,

$P(w_k|I_i)$, the ratio between its normalized count for this image and its mean normalized count within this group of images, $r_{i,k}$, as in Equation 2.6, and a binary version of this ratio, thresholded at 1:

$$V_{i,k} = \begin{cases} 1 & \text{if } r_{i,k} \geq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

The typical counts for the most related visual words vary considerably across different image classes, so making the features binary may give better performance when we use the image features to learn a generic model in a query-relative manner, even though it discards some information.

We cannot usefully perform the same kind of binary conversion for the query-relative features created using binning: taking only binary summaries over a small number of bins would give the features very low representational power, while making the feature values binary before summing the bin contents would not achieve the same purpose of making the final query-relative features' magnitudes consistent between queries.

2.3.3 Comparison of query-relative representations by sorting and by binning

If we consider the textual context for different queries, we can consider the sorting procedure as defining an alignment between the terms that appear, as shown in the 'input' and 'sorting' columns of Figure 2.2. The first-sorted feature from a test query is aligned with the first-sorted features from the training queries, the second-sorted feature with the second-sorted features from the training queries, and so on. A single shared weight will be learned for all the features from different queries which are aligned with each other. Although the weights for different queries correspond to different words, they have similar functions relative to the different queries.

Query relative features by sorting and binning can be both represented as linear transformation of input features, where parameters of linear transformation $\mathbf{Q}$ depend on the query:

$$r(I,Q) = \mathbf{Q}\, p(w|I) \tag{2.8}$$

Figure 2.3 shows that sorting and binning differ only in the linear transformation $\mathbf{Q}$ used to transform input features. Since the computational complexity of sorting is bigger than the one of binning, the binning procedure is much faster.

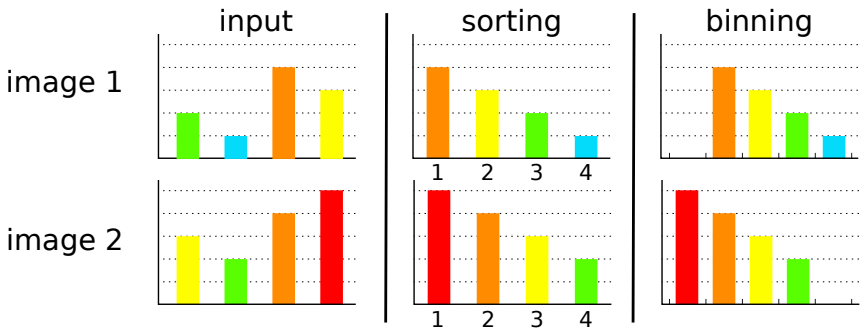

***Figure 2.2**: Illustration of sorting versus binning for feature alignment, to decide which features from different classes to learn shared weights for.*

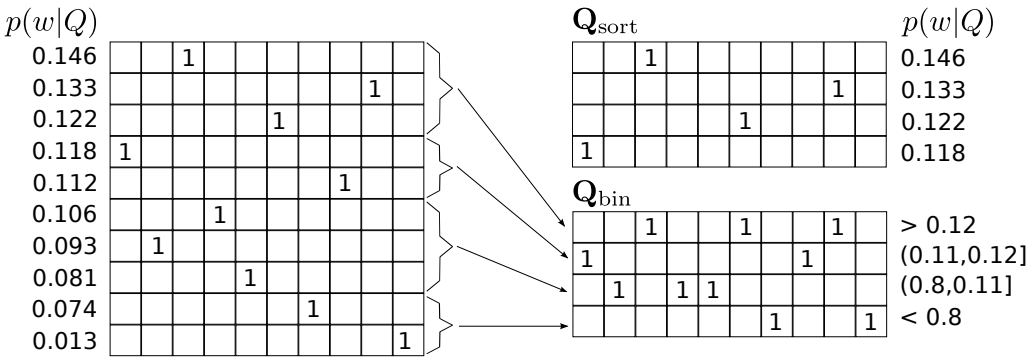

***Figure 2.3**: Comparison of linear transformations of input features for query-relative features obtained by sorting and by binning. On the left hand side the permutation matrix is displayed, obtained by sorting $p(w|Q)$, where rows and columns correspond to words. Linear transformation of input features in the case of sorting $\mathbf{Q}_{sort}$ corresponds to first N rows of permutation matrix on left hand side. Linear transformation of input features in the case of binning $\mathbf{Q}_{bin}$ corresponds to sum of rows of the permutation matrix on left hand side, where the grouping of words is defined by bin thresholds.*

The 'sorting' column in Figure 2.2 reveals another potential drawback in sorting procedure: words with quite different context frequencies may aligned by sorting, unless the input distributions precisely mirror each other. It might be better to instead align the features according to frequency bins, as shown in the 'binning' column of Figure 2.2, as the appropriate weight for each word is likely to depend on the its frequency rather than directly on how it sorts compared to other words.

Our experiments in Section 2.4 show that using $\rho(Q)_k$ to assign features to bins, and summing $r_{i,k}$ within each bin, gives better performance than aligning features by sorting. In the remainder of the section we focus on construction of query-relative features by binning and propose several extensions of the described basic approach.

2.3.4 Multi-dimensional binning

The query-relative visual word features described above work by comparing a visual word's frequency in the query set with its general frequency, to decide a bin, then by comparing the visual word's frequency in a single image with its frequency in the query set, to decide how much to add to that bin's value. It would be possible to represent the same information through bins in two-dimensions, with one axis representing the ratio in Equation 2.5 and the other representing the ratio in Equation 2.6, incrementing a bin's value by one for each feature assigned to it. By instead separating out the three distinct terms from Equations 2.5 and 2.6 into a three-dimensional binning, we can create features which give the potential for a richer structure of weights to be learned. This way the counts of the features allocated to each bin are used, rather than sums of feature values, and the model is free to learn flat weights across different bins if some variation between feature values is not informative.

We calculate three distributions: $P(w)$, estimated from all the training data, $P(w|Q)$, estimated from the images in the query set, and $P(w|I)$, estimated from a single image of interest. We create the query-relative representation for an image by giving each visual word a single vote in the space $(P(w|I),\ P(w|Q),\ P(w))$, then counting the number of votes within each bin of a three-dimensional grid. The sum across all the bins will be the number of visual words.

Under the single-dimensional binning, for example, all features with the same ratio between $P(w|Q)$ and $P(w)$ are assigned to the same bin, but it might be appropriate to learn a different weight for a feature depending on the absolute frequencies in question – if the feature is still very rare then an increase in its frequency might not be significant.

With multi-dimensional binning, very little computation is required to classify images for a new class. $P(w)$ and $P(w|I)$ can be pre-computed, then only these steps are required when a new search query is received from a user:

- Estimate $P(w|Q)$ from the query set.
- For each image, iterate over the visual words to create a compact query-relative feature descriptor by incrementing the $(P(w|I),\ P(w|Q),\ P(w))$ bins.
- Use the obtained query-relative image representation to derive the score for the image using a previously learned classifier.
- Sort the images according to assigned relevance scores.

2.3.5 Non-parametric query model

One possible objection to the procedure described above is that we assume that the mean visual word histogram over the images in the query is a good representation of the histograms for the individual images. If there are many outliers in the images returned for the query, or, worse, if there are multiple distinct clusters, this may not hold. We can avoid this assumption by non-parametric modeling of $P(w|Q)$.

Above, to represent an image I we loop over the visual words w in the vocabulary and increment the 3D histogram cell corresponding to the 3D frequency tuple $(P(w|I),\ P(w|Q),\ P(w))$. In the non-parametric case, we will instead increment multiple cells, for each image I_i in the query set. We use the tuple $(P(w|I),\ P(w|I_i),\ P(w))$ to determine the cell, and increment it by $P(w|I_i)/\sum_{I_j \in Q} P(w|I_j)$. This way we better model queries with multi-modal visual word distributions and also decrease the influence of outliers on the query model.

In Section 2.4 we will examine whether the additional computational cost of using the non-parametric query model is repaid by improvements in performance.

2.3.6 Bin thresholds

Before features can be allocated to the appropriate bins and the sum for each bin calculated, we need to decide what will go into each bin. We set up the bins as a rectangular grid in the appropriate input space, but with potentially unequally spaced bin thresholds.

Specifically, for each dimension we take a sample of the values to be binned, and choose bin thresholds which put approximately equal amounts of data into each bin. Making the bins have approximately equal probability in this way maximizes the information encoded by a set of bin allocations. For example, to create four bins we set bin thresholds at the lower quartile, median, and upper quartile of the sample. Each quantity that determines a dimension of binning, such as $\rho(Q)_k$ or $P(w_k)$, has independent bin thresholds, but for each quantity a single set of bin thresholds are precalculated and shared across all search queries.

2.4 Experimental evaluation

In this section we evaluate our method's performance on two data sets: a new data set of images for 353 search queries, and the Google images data from [Schroff *et al.*, 2007]. We compare combinations of the query-relative features described in previous section. In each case we train a logistic regression classifier for each query from the

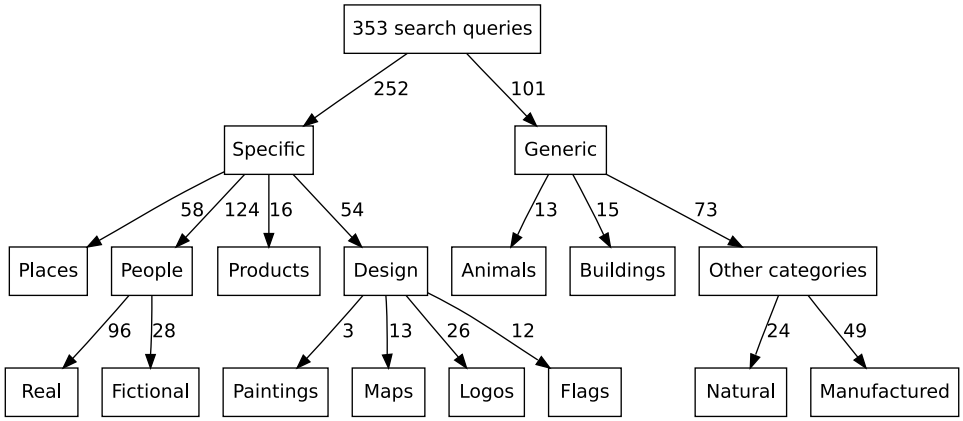

***Figure 2.4**: Partial hierarchy of search queries in the 353 query data set, to illustrate the distribution of query types.*

annotated data for all the other queries. Images are then ranked according to their classification scores.

2.4.1 353 query data set

We introduced a new large public data set of 353 image search queries in [Krapac *et al.*, 2010]. The data set consists of the images found by the text-based Exalead image search engine for 353 different search queries, their textual meta-data, and manual binary annotations saying which images are relevant to the query. Previous image re-ranking data sets [Fergus *et al.*, 2004, Li *et al.*, 2007, Schroff *et al.*, 2007] contain images for only a few classes, and in most cases provide image files without their corresponding meta-data.

For each of the 353 search queries, the data set includes the original textual query, the top-ranked images found by the web search engine, and an annotation file for each image. For 80% of queries there are more than 200 images; in total there are 71478 images in the data set. The annotation files contain manual binary labels for image relevance to the search query, and other meta-data obtained from the web: the image URL, the URL of the web page where it was found, the page title, the image's alternative text, the 10 words before the image on the web page, and the 10 words after. The images themselves have been scaled to fit within a 150×150 pixel square while maintaining the original aspect ratio – web search engine databases typically only include thumbnail images, as storing the full original images is too expensive. The fraction of relevant images for each query is 44%, comparable to the 39% found by [Schroff *et al.*, 2007] for Google image search.

Figure 2.4 shows a partial hierarchy for the search queries in the data set, to illustrate the distribution of different query types. This hierarchy was obtained by manual in-

spection of the queries in the data set, and is not used in our experiments. Around 70% of the queries relate to specific object or scenes, while the other 30% relate to generic categories of objects or scenes. Almost half of the specific queries are for images of people or fictional characters. The rest are for famous landmarks, specific products such as games consoles and football team shirts, and two-dimensional designs such as logos and flags. Almost half of the generic queries are for images of manufactured items, such as different types of vehicle and musical instruments. The largest other groups in the generic category queries are for different species of animals and different types of buildings. The remainder of the generic queries are for other types of natural objects or scenes, such as various landmarks and different types of fruit.

2.4.2 Model training

In our main experiments below we hold out each query in turn and train a binary logistic discriminant classifier from annotated data for all other queries.[2] Query-relative features of relevant images are used as positive examples, and query-relative features of irrelevant images are used as negative examples. We use the learnt model to rank images for the held-out query by the probability that they are relevant according to the model. Note that in a practical application the logistic discriminant parameters would only need to be learnt once, and could then be re-used for any new query. For methods which take a long time to train we used 10-fold cross-validation for evaluation.

We extracted local image descriptors on a dense multi-scale grid, and normalized so that the L2 norm is equal to one. Each local region is described by a 16-dimensional histogram of oriented gradients. Preliminary experiments showed that this performed better than 128-dimensional SIFT.

Our image representation is based on local appearance and position histograms. Local appearance is quantized using hierarchical k-means clustering as in [Nistér & Stewénius, 2006], learnt from a set of patches sampled from all images in the dataset, with 11 levels of quantisation, with the branching factor $k = 2$ at each level, yielding 2048 visual words at the finest quantisation level. For position quantisation we used quad-trees [Lazebnik *et al.*, 2006], with three levels of quantisation, yielding 64 spatial cells at the finest quantisation level. For each combination of appearance and position quantisation an image is described by an appearance-position histogram: each bin in the joint histogram represents a patch appearance at a certain position in the image. For appearance quantisation, only the levels with at least 128 quantisation bins (levels 7–11) are used, as at lower levels the bins are not sufficiently

[2] We also tried support vector machines instead of logistic discriminant classifiers; the results were almost the same but training was slower.

	Overall	LP	HP	SEP	SEG
Number of queries	353	25	25	25	25
Fraction relevant	44.3	12.1	78.0	52.5	31.5

***Table 2.2**: Data set properties by group.*

discriminative. The image is represented by a concatenation of all 14 appearance-position histograms with dimensionality less than 10000; the histograms with higher dimensionality are extremely sparse. In our experiments, the binning or sorting is performed after the appearance-position histograms from different levels of quantisation have been concatenated, resulting in 42368-dimensional histogram for each image. Therefore the query-relative image features may mix features from different levels of appearance and position quantisation.

We also add colour features as another feature channel. We take a downsampled 32×32 pixel version of each image, and assign each pixel's colour to one of 100 'colour words' according to a $4 \times 5 \times 5$ grid over CIELAB colour space. The resulting histograms of colour words are then normalised, and processed in the same way as the histograms of visual words to create query-relative colour features.

2.4.3 Evaluation

We evaluate a model's re-ranking performance by calculating the average precision (AP) for the scores it assigns to the images for each query, and taking the mean across all queries. These results can be compared with the precision of the queries and with the mean average precision of the search engine's rankings.

To allow a more detailed evaluation, we chose four groups of queries with extreme behaviour on the search engine, which uses an approach focused on textual cues:

- Low Precision (LP): 25 queries where the search engine's retrieval by text query performs worst, e.g. 'will smith', 'rugby pitch', 'bass guitar', 'mont blanc', 'jack black'
- High Precision (HP): 25 queries where the search engine's retrieval by text query performs best, e.g. 'batman', 'aerial photography', 'shrek', 'pantheon rome', 'brazil flag'
- Search Engine Poor (SEP): 25 queries where the search engine improves least over random ordering of the query set, e.g. 'clipart', 'cloud', 'flag', 'car'
- Search Engine Good (SEG): 25 queries where the search engine improves most over random ordering, e.g. 'rugby pitch', 'tennis court', 'golf course', 'ben stiller', 'dustin hoffman'.

Note that there is some overlap between these groups. In our results tables below, as well as the overall mean average precision on all 353 queries, we show the mean average precision for queries in each of these groups of 25 queries. The fraction of relevant images, averaged over the queries in each group, is shown in Table 2.2; this gives an indication of the difficulty of each group. For random ranking the AP score is close to the fraction of relevant images, while for the perfect ranking the AP score is 100.

2.4.4 Ranking images by textual features

In our first set of experiments we examine the image ranking performance of our textual features. Table 2.3 shows the mean average precision achieved by the Exalead web search engine, and by our text-only models using different feature sets. Note that we do not have access to the ranking system used by the Exalead search engine, and it may use additional features beyond those in the data set annotation files, such as click-through data or PageRank-like scores.

We start with the base features, similar to those used in [Frankel *et al.*, 1997, Schroff *et al.*, 2007], representing the presence or absence of the query terms themselves in various sources of text; these words are then excluded from consideration when creating the context features. The set of nine base features supplemented with nine partial match features forms 18 features referred as "base features". We then add an increasing numbers of context features obtained using binning or sorting to create query-relative features. Only single-dimensional binning is used for the text context features, where we have at most 20 words of context for each image, and cannot make a good estimate of $P(w|I)$. The overall performance increases as more context features are added, but with a diminishing gain per additional feature.

Our text features alone beat the performance of the search engine. The methods perform differently for different queries, as can be seen looking at the extreme groups in Table 2.3. We beat the search engine on the 'high precision' group where text retrieval performs best, and do slightly worse on the 'low precision' group where we do not expect a text-only algorithm to do well. Although we do worse on the 'search engine good' group where their algorithm works best, we perform significantly better on the 'search engine poor' group.

Keeping the sources of text context (the text surrounding the image, the image's alternative text, and the web page title) separate gives better performance than merging the three sources. Using 20 binned features per source we reach a better mean average precision than achieved by the search engine or by using sorting to create query-relative features with 100 features per source, and the performance increases if more bins are used.

mAP × 100	Dimensions	Overall	LP	HP	SEP	SEG
Base features	18	54.8	23.5	82.4	60.3	51.2
Sorting	+ 10 × 3 = 48	56.5	22.6	83.4	61.6	53.3
	+ 20 × 3 = 78	56.5	24.0	83.0	61.4	54.4
	+ 50 × 3 = 168	56.8	22.7	83.6	62.2	53.2
	+ 100 × 3 = 318	57.0	24.3	84.1	62.4	54.8
Binning merged	+ 30 = 48	56.7	24.3	82.3	61.6	52.2
	+ 60 = 78	57.0	25.1	82.6	61.8	52.9
	+ 120 = 138	56.7	23.9	83.2	60.8	53.3
	+ 180 = 198	56.6	22.7	83.2	61.6	53.0
Binning separate	+ 10 × 3 = 48	56.9	24.4	83.1	62.7	52.1
	+ 20 × 3 = 78	57.5	25.8	83.1	61.6	53.5
	+ 40 × 3 = 128	58.2	26.5	84.0	**63.3**	54.1
	+ 60 × 3 = 198	**58.4**	26.3	**84.9**	63.2	55.3
Search engine	N/A	56.9	**26.8**	83.0	49.5	**63.4**

***Table 2.3**: Mean average precision achieved with text features, on the 353 query data.*

2.4.5 Ranking images by visual features

mAP × 100	Overall	LP	HP	SEP	SEG
re-ordered histogram bins $[P(w_k\|I_i)]$	60.5	21.2	89.0	70.3	53.5
+ query-specific normalization $[r_{i,k}]$	59.7	21.0	87.6	67.7	53.1
+ binarization $[V_{i,k}]$	**64.4**	23.8	**90.7**	**72.0**	57.8
Search engine	56.9	**26.8**	83.0	49.5	**63.4**

***Table 2.4**: Performance of different feature representations, using visual features of dimensionality 200, without text features.*

In Table 2.4 we compare the performance of the different query-relative visual feature representations using sorting, described in Section 2.3.2. The first set of features, $P(w_k|I_i)$, uses the visual word histograms reordered by relative frequency of visual words in the query set compared to frequency of visual words in a general set of images. The second set of features, $r_{i,k}$, uses the ratio between the visual word frequency in an image and its frequency in the query set. The third set of features, $V_{i,k}$, thresholds these ratios at one to give binary indicators. The increase of the performance when we move to binary features is probably because the relative frequencies of visual words vary considerably between queries. In the following experiments we always use these binary query-relative visual features when using sorting to create query-relative features.

mAP × 100	Dimensions	Overall	LP	HP	SEP	SEG
Sorting	50	61.1	20.3	90.4	70.0	52.2
	100	63.1	21.9	90.6	71.3	56.2
	200	64.4	23.8	90.7	72.0	57.8
	400	64.9	24.1	91.0	71.9	58.4
Binning	10	62.9	27.5	87.5	71.8	52.3
	20	63.0	27.4	87.9	72.1	52.8
	40	63.1	27.4	88.2	72.2	53.0
	60	63.1	27.3	88.1	72.4	52.8
Multi-dimensional binning	$3 \times 3 \times 3 = 27$	68.2	30.4	91.9	**76.1**	60.6
	$4 \times 4 \times 4 = 64$	68.5	30.5	92.2	75.1	59.8
	$5 \times 5 \times 5 = 125$	68.6	30.8	91.9	75.2	60.7
	$6 \times 6 \times 6 = 216$	68.7	**31.4**	93.0	75.2	60.3
Multi-dimensional binning with non-parametric $P(w\|Q)$	$3 \times 3 \times 3 = 27$	68.3	28.7	93.3	75.9	59.9
	$4 \times 4 \times 4 = 64$	**68.9**	25.3	93.7	**76.1**	61.3
	$5 \times 5 \times 5 = 125$	68.6	25.6	93.6	75.9	60.7
	$6 \times 6 \times 6 = 216$	68.3	26.2	**93.8**	75.9	61.0
Search engine	N/A	56.9	26.8	83.0	49.5	**63.4**

***Table 2.5**: Mean average precision achieved with texture features (without text or colour features), on the 353 query data.*

Table 2.5 compares the performance of different query-relative features, without text features. Adding more visual features increases the overall performance, but with diminishing gains. The overall performance is better than that achieved by text features alone.

Looking at the extreme groups, we find that the 'low precision' group is also hard to rank using visual features, and that the 'high precision' group gives even better results using image features: the extremely low and comparatively high precision of the data for the queries in these two groups also hinders and aids ranking when visual features are used. Visual features give us a reliable advantage over the search engine's performance on the 'search engine poor' group, and also beat its results on the 'search engine good' group where the search engine gave the biggest improvement over random ordering.

The binning approach is competitive with sorting, while using a much smaller number of features (e.g. 20 features rather than hundreds). Moving to multi-dimensional binning, there is a significant increase in overall mean average precision, with an increase in performance seen on all the highlighted groups.

The last group of experiments shows that using non-parametric model of $P(w|Q)$ gives slightly better results in the best case, but also worse results in others. The best results are achieved using non-parametric query models, but the improvement is not

large, or reliable across parameter choices. The results on the 'high precision' group consistently improve, but the 'low precision' results are worse, while on the other extreme groups there is a general but unreliable improvement. Since non-parametric modeling is more computationally expensive, in the remainder of the chapter we will use the query-relative features where $P(w|Q)$ is estimated from the mean visual word histogram of images in the query set.

mAP × 100	Dimensions	Overall	LP	HP	SEP	SEG
Binning	10	55.6	18.5	86.6	65.6	43.4
	20	55.8	19.1	86.5	65.8	43.5
	40	55.8	19.1	86.2	66.8	43.2
	60	55.9	19.1	86.3	66.3	43.1
Multi-dimensional binning	$3 \times 3 \times 3 = 27$	56.4	22.4	86.9	68.1	46.9
	$4 \times 4 \times 4 = 64$	57.1	21.2	87.8	68.9	45.5
	$5 \times 5 \times 5 = 125$	57.7	21.8	87.4	67.7	47.2
	$6 \times 6 \times 6 = 216$	**58.2**	23.6	**88.2**	**69.3**	47.7
Search engine	N/A	56.9	**26.8**	83.0	49.5	**63.4**

***Table 2.6**: Mean average precision achieved with colour features (without text or visual word features), on the 353 query data.*

Table 2.6 shows the performance of our method using only colour features. The mean average precision is comparable to that achieved using base and contextual text features. As with the texture features, performance increases when multi-dimensional binning is used. Note that the colour features used here are much less computationally expensive than the texture features, so using colour features alone for re-ranking might be interesting in a low resource scenario.

2.4.6 Combining textual and visual features

In Table 2.7 we look at the performance of various combinations of textual and visual features. We combine between 50 and 400 visual features with the 18 base and partial match text features and between 20 and 100 additional features for each text field. Using larger numbers of features gives better performance, though the performance is not much worse with the smaller feature sets. The single-dimensional binning results are comparable to those achieve using sorting with many more features – better on some subsets, worse on others. Moving to multi-dimensional binning shows superior performance across all the subsets. Adding colour features provides relatively little additional information, but performance increases overall and on three of the four subsets. The larger feature sets (still of relatively low dimensionality) beat the search engine on all four groups of extreme queries. The best overall mean average precision

mAP × 100	Dimensions			Overall	LP	HP	SEP	SEG
	Textual	Visual	All					
Sorting Visual = Texture	20 × 3	50	128	64.3	25.6	90.4	70.6	60.3
	20 × 3	100	178	65.7	26.9	90.7	71.9	62.5
	20 × 3	200	278	66.8	29.1	90.9	72.7	63.7
	20 × 3	400	478	67.3	28.9	91.2	73.1	65.0
	100 × 3	400	718	67.3	29.2	91.3	73.7	65.5
Multi-dimensional binning Visual = Texture	60 × 3	4 × 4 × 4	262	70.3	34.0	92.7	75.6	65.8
	60 × 3	5 × 5 × 5	323	70.5	35.0	92.4	75.3	66.9
	60 × 3	6 × 6 × 6	414	70.6	36.4	93.1	75.7	**67.9**
Multi-dimensional binning Visual = Texture + Color	60 × 3	2 × (4 × 4 × 4)	326	71.2	36.6	93.2	77.3	66.8
	60 × 3	2 × (5 × 5 × 5)	448	71.5	35.9	92.9	76.8	66.9
	60 × 3	2 × (6 × 6 × 6)	630	**71.6**	**37.5**	**93.7**	**77.7**	67.7
Search engine	N/A			56.9	26.8	83.0	49.5	63.4

***Table 2.7**: Mean average precision on "353 queries" dataset, achieved by combining query-relative features from multiple channes: text, texture and color.*

shown is almost 15% better than the search engine, and more than 13% better than our best result using textual features alone.

For the remaining experiments we consider only the model that uses 60 × 3 binning for text features and 6 × 6 × 6 multi-dimensional binning for visual features, that corresponds to the best-performing experiment in Table 2.7.

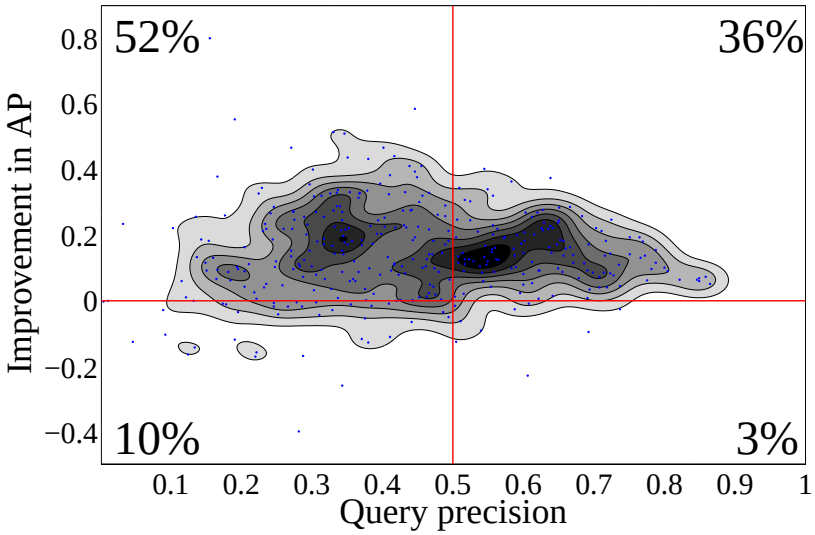

***Figure 2.5**: Improvement in AP using query-relative features over the search engine, as a function of the fraction of relevant images in the query set.*

In Figure 2.5 we show per query differences in performance of our method that uses query-relative feature obtained by binning and the search engine performance, with respect to the precision of the query sets. These results show that for about 62% of the queries the precision in the query set is under 50%, giving the insight into the difficulty of the re-ranking problem. For about 88% of all queries our model outperforms the search engine. It might seem that our method will only work when the majority of the images in query set are relevant, but for 52% of all queries we improve results despite a query set precision under 50%.

To understand why the method still works in such cases, note that the irrelevant images tend to be more diverse than the relevant ones. Even if only a minority of the images is relevant, the visual words in relevant images can still be the most frequent ones, as compared to the database statistics.

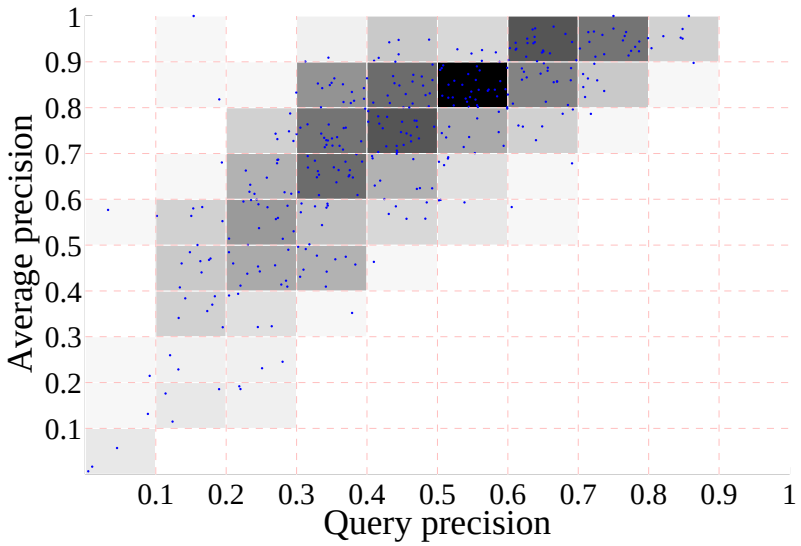

***Figure 2.6**: Average precision using query-relative features, plotted against the precision of the data provided for each query.*

Figure 2.6 shows how the average precision achieved by the model varies according to the precision of the query data used as input. We see that even for queries with less than 20% of the input data relevant to the query, the mean average precision is about 40%, while the mean average precision for queries with between 20% and 40% relevant input data is more than 60%. The mean average precision increases to almost 80% for queries with between 40% and 60% relevant input data.

The results presented so far measure at how well the method performs in terms of mean average precision. Since many web search users only look at the top-ranked search results, it is also interesting to look at the precision achieved at small recall levels across different queries. Figure 2.7 shows the distribution of the precision of

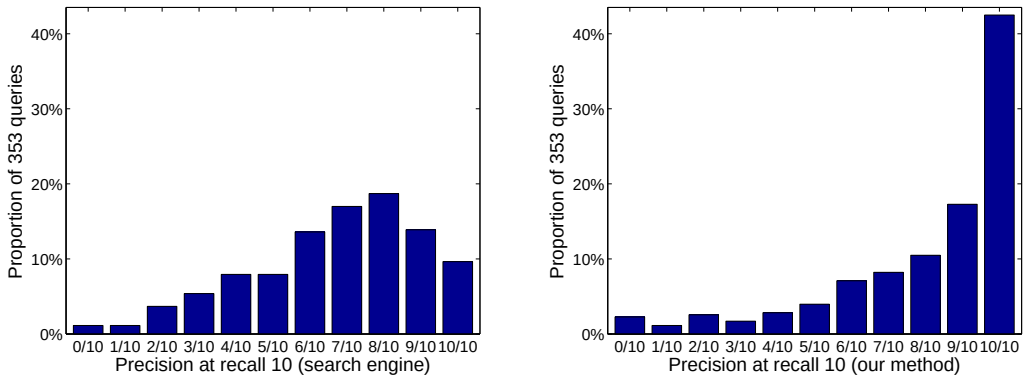

***Figure 2.7**: Proportion of 353 queries with a given precision on the first 10 images recalled, using the search engine, and using our method.*

the first 10 images recalled for each query by the search engine and by our method. The most common precision for the search engine is 8/10, achieved on 66 queries (19%), while the most common precision for our method is 10/10, achieved on 150 of the 353 queries (43%). The search engine only gives a precision of 9/10 or 10/10 on the first 10 images recalled for 24% of the test queries, while our method gives a precision of 9/10 or 10/10 for 60% of the queries.

Figure 2.8 shows the top-ranked images for some example queries from the 353 query data set where our method performs well despite low precision input data. For each query the images are ordered left to right. Our method does well on these queries, down-ranking irrelevant images, except on 'will smith' where it fails due to extremely low precision input data (less than 1%). In this case the model seems instead to detect portraits.

2.4.7 Comparison with query-specific classifiers

Motivated by very good results of our method to search engine's results, we compared the performance of our method to a state-of-art method for image re-ranking [Schroff *et al.*, 2007]. We trained a support vector machine specific for each query in the data set. Visual features are the histograms of visual word counts, the same ones used to derive our query-relative visual features. We used a RBF kernel with χ^2 distance measure, where the bandwidth of the exponential kernel is set to average χ^2 distance between training examples. Following the approach of [Schroff *et al.*, 2007] a query-specific classifier is learnt from noisy data. The images in the query set were treated as noisy positive training examples, while 1000 negative examples were sampled randomly from other queries. The regularisation parameter of SVM was set to 1 for all queries.

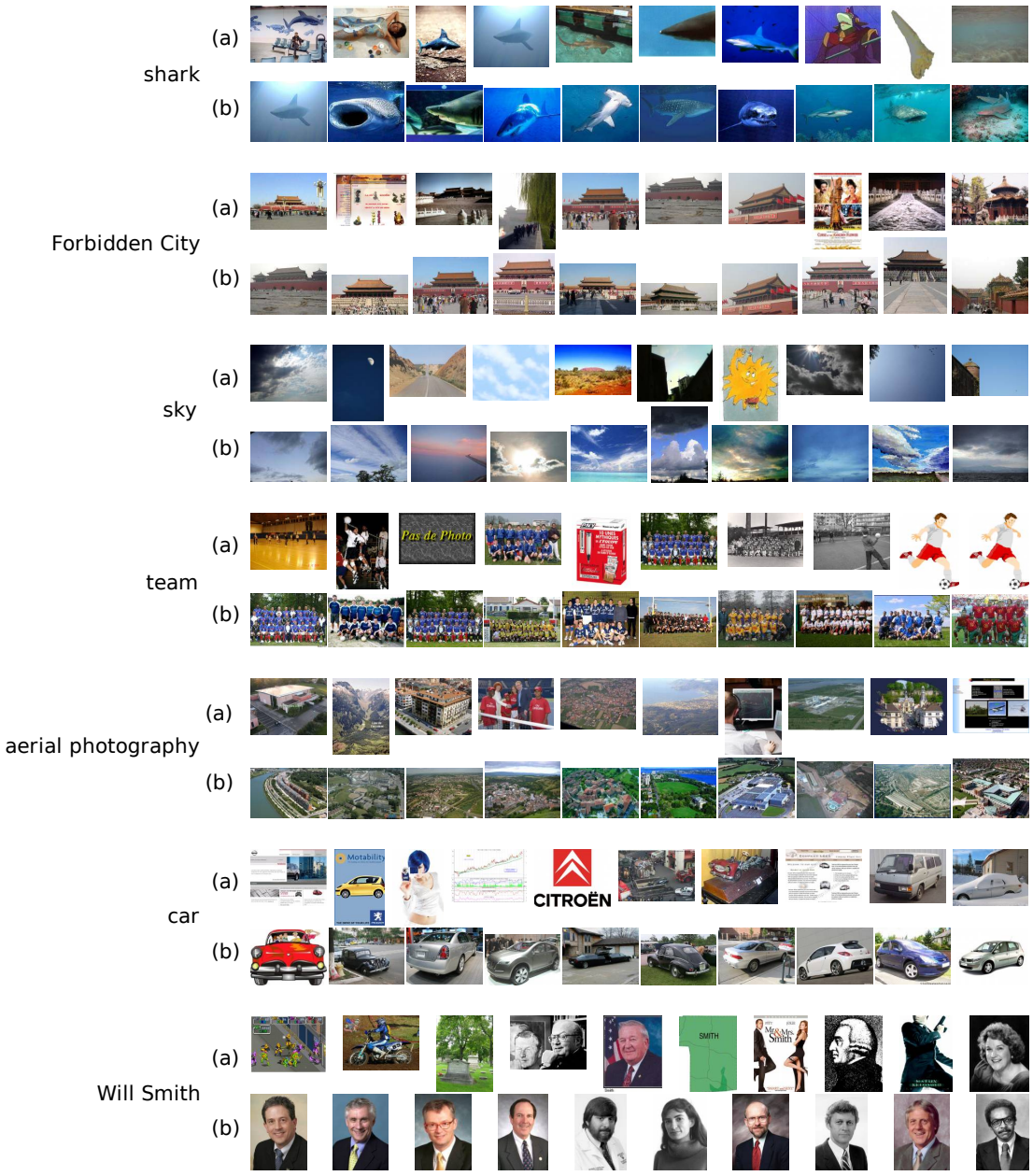

***Figure 2.8**: Top-ranked images, left to right, for some example search queries from the 353 query data: (a) search engine's ranking, (b) our ranking. The features perform well despite noisy input: shark (input precision 33.5%), Forbidden City (35.4%), sky (33.6%), team (35.0%), aerial photography (72.4%), car (45.5%), although they fail in the case of extremely low input precision: Will Smith (1%)*

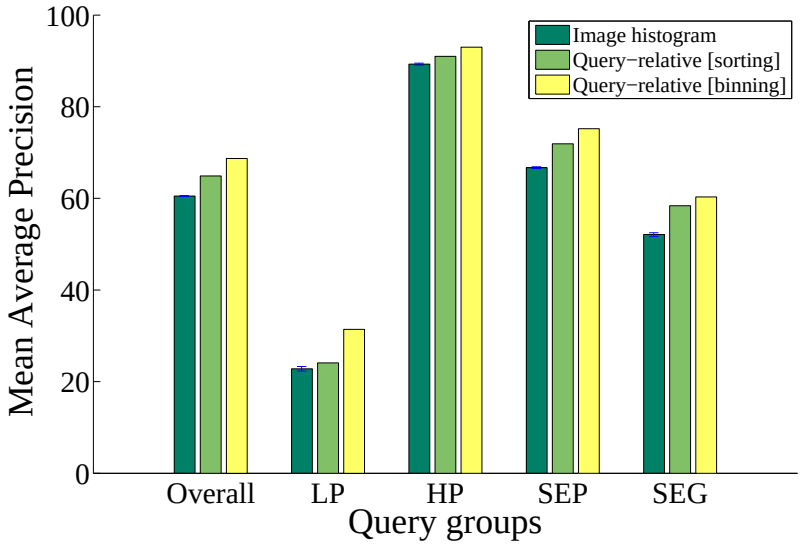

***Figure 2.9**: Comparison of query-relative texture features with query-independent image representation (BoW image histogram). Classifiers trained using query-independent representation are query-specific. The results for the query-specific classifiers using BoW histograms depend on the random set of negative images used to train the classifier. The results for query-specific classifiers are means and standard deviations of mAP across five runs.*

In Figure 2.9 the performance of query-specific classifiers using query-independent features (BoW image histograms) is compared to performance of query-relative features. Image histograms are obtained by concatenating 14 appearance-position histograms with dimensionality less than 10000, resulting in 42368-dimensional histogram for each image. Perhaps suprisingly, query-specific classifiers give worse performance than generic query-relative classifiers. The results could be enhanced by reducing the dimensioanlity of image histogram, since each query-specific classifier is learnt from only about 1300 training examples. However, it should be taken into account that although our query-relative features are derived from the same representation that is used directly by SVM, we do not suffer from over-fitting.

The SVM performance could also be enhanced by optimising the parameter settings per class. [Schroff *et al.*, 2007] use ten-fold cross validation to learn appropriate parameter settings for each class, while in our experiments we used a single set of parameters for all classes. The best parameters will depend, among other factors, on the proportion of irrelevant images in the input data, which varies across queries.

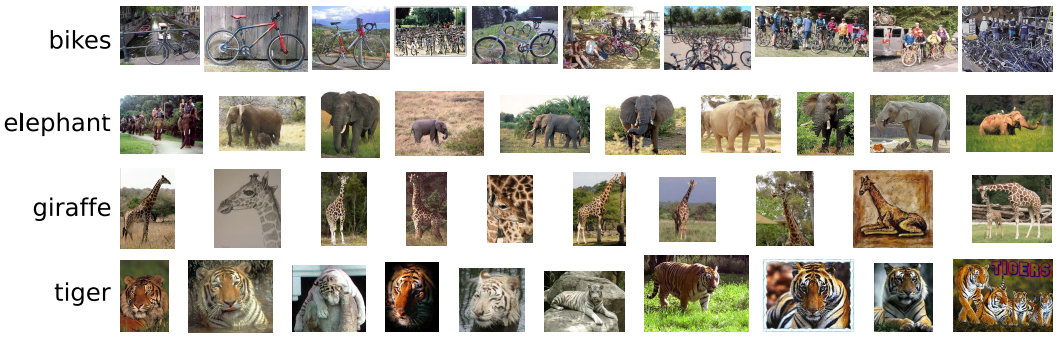

***Figure 2.10**: Top-ranked images, left to right, for some classes from the Google images data set using the* 60×3 *text feature,* $4 \times 4 \times 4$ *visual word and colour feature model.*

2.4.8 Google images data set

[Schroff *et al.*, 2007] introduced a data set of images obtained from Google image search for 18 object classes, along with textual meta-data. The images have been manually annotated to say whether or not they contain an object of the relevant class. We use the same filtered subset of this database as used in [Schroff *et al.*, 2011], containing only images which were automatically classified as 'non-abstract': photographs rather than, for example, line art. As with the previous data set, we test on each class using a classifier trained on all the other classes. Since this data set contains fewer classes, the training data is more limited in this case.

Table 2.8 shows the performance of our method on this data using various combinations of text, visual word, and colour features, as well as the best performing experiment from [Schroff *et al.*, 2011], and the best performance achieved there for each class. The query-relative features from multi-dimensional binning achieve better precision than query-specific classifiers that use the same set of parameters for training all query-specific models. The performance of query-specific classifiers is increased when the parameters are tuned per class. However, with added qurey-relative colour features our results are just 1.2% short of the best ones obtained by [Schroff *et al.*, 2011]. Our results on the 'beaver' class are poor, because the data set for this class is dominated by images corresponding to another meaning of the search term.

Figure 2.10 shows the top-ranked images for some example classes from the Google images data, for a model trained using 60×3 text feature, $4 \times 4 \times 4$ visual word features and $4 \times 4 \times 4$ colour features.

P15% × 100	Visual dimensions	Mean	airplane	beaver	bikes	boat	camel	car	dolphin	elephant	giraffe	guitar	horse	kangaroo	motorbikes	penguin	shark	tiger	wristwatch	zebra
Multi-dimensional binning Text = 60 × 3 Visual = Texture	4 × 4 × 4	71.2	66.7	6.3	77.1	64.9	65.2	94.6	67.2	87.1	91.0	73.2	65.9	58.8	77.3	75.6	60.8	82.5	79.3	87.5
	5 × 5 × 5	72.2	68.1	6.6	82.2	62.3	72.3	95.9	65.0	88.5	87.1	72.4	66.5	59.5	80.7	76.4	63.9	**87.0**	77.0	88.7
	6 × 6 × 6	70.4	68.1	6.3	79.6	60.6	72.3	**97.9**	65.6	85.2	77.1	69.7	64.8	60.2	77.3	76.4	60.8	81.0	79.9	84.0
Multi-dimensional binning Text = 60 × 3 Visual = Texture + Color	2 × (4 × 4 × 4)	73.9	63.6	6.8	82.2	**67.3**	68.2	95.9	67.8	88.5	**93.1**	**78.0**	69.0	66.7	**86.0**	78.2	66.0	85.5	77.0	91.3
	2 × (5 × 5 × 5)	73.5	**71.1**	7.0	**84.6**	64.9	**75.3**	95.9	63.0	**89.8**	88.3	73.6	69.3	66.7	80.7	76.4	67.4	85.5	75.4	87.5
	2 × (6 × 6 × 6)	72.5	68.6	6.4	80.0	61.2	70.9	97.2	64.0	87.1	80.7	**78.0**	67.7	**71.4**	78.0	82.9	66.0	83.9	80.5	80.8
[Schroff *et al.*, 2011] Text + Visual (HOG+BoW)	best shared	70.6	51.3	42.3	68.2	60.3	63.2	91.1	**69.7**	78.7	88.7	66.3	70.1	53.9	76.6	**90.2**	66.1	50.2	92.5	91.7
	best per class	**75.1**	63.5	**46.4**	71.0	66.2	63.2	93.5	**69.7**	84.8	88.7	67.3	**71.2**	56.2	84.6	**90.2**	**82.9**	62.6	**93.2**	**96.8**

Table 2.8: *Precision at recall 15% achieved by methods on the Google image data from [Schroff et al., 2011]*

2.5 Conclusion

In this chapter we have introduced query-relative features that can be used to train generic classifiers, and rank images for previously unseen search queries without additional model training. The features combine textual information about the occurrence of the query terms and other words found to be related to the query, and visual information derived from a visual histogram image representation. Previous state-of-the-art approaches required computationally expensive model training for every new search query, and so were unsuitable for real-world web search applications, while the query-relative features proposed here require very little computation when a new search query is received, making them practical to use (see Section 2.3.4). We showed that query-relative features obtained by binning outperform ones obtained by sorting, in addition being also cheaper to compute.

We applied our image re-ranking method to the top-ranked images returned by a web search engine. We found that our query-relative models gave a considerable improvement over the raw search engine ranking, showing an increase in mean average precision of almost 15%. Surprisingly, we also found that our model significantly outperforms state-of-the-art approach that requires training a visual classifier per query. In addition, we showed performance on the 18 Google images classes are comparable to the best results achieved in [Schroff *et al.*, 2011] by more computationally expensive class-specific classifiers.

We also present a new public data set of images returned by a web search engine for 353 search queries, along with their associated meta-data, and ground-truth annotations for all images. The intention of this data set is to facilitate further progress in improving image search.

3

Learning Tree-structured Quantizers for Image Classification

Contents

3.1 Introduction

In this chapter we discuss the image representations for image categorization and propose a method for learning the features for image categorization. This work has been submitted to British Machine Vision Conference 2011.

As explained in Chapter 1, the goal of image categorization is to assign labels to an image that relate to the image content at a semantically meaningful level. For example, the labels can indicate the presence of object categories, such as *cars*, or *bicycles*, or they can refer to scene types such as *indoor, city, beach*, *etc.*. Over the last decade significant progress has been made in this area. Most of the current state-of-the-art systems are based on the bag-of-words image representation, a processing pipeline tracing back to [Csurka *et al.*, 2004] which consists of the following steps. (i) Local image regions are sampled, using interest points or from a regular grid. (ii) The appearance of regions is described using features with some degree of photometric invariance, *e.g.* using SIFT [Lowe, 2004]. (iii) Features are quantized into a finite vocabulary, *e.g.* obtained by k-means clustering and each feature is coded by its quantization index. (iv) The overall image content is described by aggregating the

quantization indices of all regions into a normalized frequency histogram. (v) The image histograms are used to train a classifier, *e.g.* a SVM.

The various stages of this pipeline have been intensively studied, and we refer to Section 3.2 for pointers to representative work in this area. However, it is important to note that generally processing steps (i)-(iv) of the pipeline are completely unsupervised; only in step (v) the image labels are used to train the classifiers. Although in some cases it might be advantageous to have a single image representation that can be used to address different tasks, the fact that the representation is not optimized for the task means that it is suboptimal in the sense that it less discriminatively and/or less succinctly captures the relevant image content.

Recently, effort has been put into optimizing quantizers for categorization tasks, see *e.g.* [Boureau *et al.*, 2010a, Lazebnik & Raginsky, 2009, Lian *et al.*, 2010, Yang *et al.*, 2010b, 2008, Moosmann *et al.*, 2008, Zhang *et al.*, 2009]. Following this line of research, we build on the tree-structured quantizers explored earlier in [Moosmann *et al.*, 2008]. Despite the computational efficiency of the approach, one of big limitations of [Moosmann *et al.*, 2008] is that the trees are learned to classify local image regions, with labels inherited from the image. Therefore the image region classification is used as a proxy for the true objective of image categorization. In this chapter, contrarily to this previous approach, we use decision trees to perform quantization of local appearances, but construct them in a manner to directly optimize categorization performance of the whole image.

In Section 3.3 we describe how we learn tree-structured quantizers using a randomized greedy forward-selection process: in each step we enlarge the decision-tree by splitting one of the existing nodes in a way that maximally improves the categorization performance. We train the model to perform ordinal regression: defined on image pairs, the loss is more expressive than the loss defined on images, and therefore is more suited to the incremental manner in which we build the clusterers, as opposed to [Lazebnik & Raginsky, 2009, Yang *et al.*, 2008] that adapt an initial k-means quantizer to be more discriminative by minimizing image classification loss using gradient-based methods. Therefore these methods will always end on the local optimum that depends on the initial quantization and may require less succint representations to achieve similar performance, or may be less discriminative for a fixed representation size.

In Section 3.4 we present experimental results on two challenging public image categorization benchmarks, we evaluate our approach and compare to unsupervised quantizers and the method of [Moosmann *et al.*, 2008]. The results show that our approach outperforms these alternatives, while producing more compact image representations. We also find that our approach benefits from using ensembles of trees, or "forests", and more so than using k-means or the method of [Moosmann *et al.*, 2008]. We summarize our conclusions in Section 3.5.

3.2 Related work

As pointed out in the introduction, most current state-of-the-art systems for image classification follow the bag-of-words framework [Csurka *et al.*, 2004] and rely on the processing pipeline described above. For the stages (i), (ii), and (v), a consensus seems to be reached, *i.e.* the use of dense sampling [Nowak *et al.*, 2006], the representation of local features by SIFT like descriptors [Zhang *et al.*, 2007], and the combination of different non-linear kernels [Varma & Ray, 2007], possibly using explicit embeddings [Maji & Berg, 2009]. Two intermediate stages (iii) quantization, and (iv) aggregation by spatial pooling – the key ingredients of a good image representation – have been the topic of active research within the last two years.

The construction of the visual vocabulary is often considered as pure vector quantization, done with k-means [Csurka *et al.*, 2004, Leung & Malik, 2001, Sivic & Zisserman, 2003] or in a soft manner with Gaussian mixture models [Philbin *et al.*, 2008, van Gemert *et al.*, 2010b]. Some authors have also noticed that the way the quantization is done can influence the performance of the classification algorithm. In [Jurie & Triggs, 2005] it was observed that the clusters are strongly unbalanced, which makes k-means focus on clustering the most frequent descriptors, that are in general less discriminative. An algorithm similar to mean-shift was proposed to overcome this limitation. For similar reasons, [Leibe *et al.*, 2006] suggests to use agglomerative clustering which gives more compact clusters, while being more robust to outliers. Hierarchical clustering is also very efficient in terms of the computational cost to assign features to visual words [Nistér & Stewénius, 2006]. In conclusion, the results shown in [Jurie & Triggs, 2005, Leibe *et al.*, 2006] demonstrate that in terms of classification performance the k-means quantizer is clearly not optimal.

More recently, methods have started to appear that design vocabularies in order to optimize classification performance, typically by merging visual words from an existing visual vocabulary [Fulkerson *et al.*, 2008, Winn *et al.*, 2005]. The drawbacks of these methods are that they rely on a large initial vocabulary, which itself is not optimal with respect to the criterion to be optimized, and they still require computationally costly assignment of features to a large number of visual words.

One of the first papers to propose optimizing the discriminative power of the dictionary is [Perronnin, 2008], which combines general and class-specific dictionaries. While [Perronnin, 2008] and [Winn *et al.*, 2005] optimize the vocabulary to ensure that the image representations as a whole are discriminative, other recent approaches optimized the visual words independently [Lazebnik & Raginsky, 2009, Mairal *et al.*, 2008, Moosmann *et al.*, 2008], usually by maximizing mutual information between category labels and the visual words. A drawback of these methods is that they aim to find visual words that can discriminate the class labels of local features (inherited from the images from which they are sampled), while the true objective is to be

able to discriminate images (as a whole) based on their distribution of visual word assignments of the patches sampled from them.

A recent group of promising methods tries to directly optimize the visual vocabulary to minimize the image classification loss [Boureau *et al.*, 2010a, Lian *et al.*, 2010, Yang *et al.*, 2010b, 2008, Zhang *et al.*, 2009]. One way to do this is to use boosting-like approaches as in [Yang *et al.*, 2008], which unifies quantizer generation with classifier training. Each image feature is encoded by a sequence of "visual bits" that are optimized iteratively for each category, based on the classifiers' performance using previous visual bits on the training data. Similarly, [Zhang *et al.*, 2009] proposes a framework for learning multiple non-redundant vocabularies, each being learned in sequence to extract the discriminative information not captured by preceding ones, and their corresponding classifiers. In [Boureau *et al.*, 2010a, Lian *et al.*, 2010, Yang *et al.*, 2010b] the dictionary and the classification model are learned jointly. In their formulation these problems are not jointly convex, but each sub-problem is, so they alternate between updating the classification model and updating the dictionary. These approaches differ in the way the feature is represented, quantization vs. sparse coding, in the aggregation method, averaging vs. max-pooling, and the loss used for image classification: (squared) hinge vs. logistic. All these methods, however, initialize their dictionary based on the reconstruction errors of the patch descriptors, therefore they will converge to local optima, which depend on the initialization.

Optimizing over the space of tree-structured quantizers that we employ in our work is hard because of the non-differentiable relationship between tree parameters and the used loss. However, our sampling strategy shows to be very effective in practice. Like [Boureau *et al.*, 2010a, Lian *et al.*, 2010, Yang *et al.*, 2010b], our approach alternates between learning classifiers and updating the quantizer. However, we incrementally grow trees to quantize the feature space — the leafs of the tree representing the visual words. By optimizing the quantizer in a coarse-to-fine manner we are less sensitive to a specific initialization of the quantizer.

3.3 Learning tree-structured quantizers

Below we present the learning algorithm for our quantizers in Section 3.3.1, the split selection criterion in Section 3.3.2, and how to leverage ensembles of quantizers in Section 3.3.3. We use $X_i = \{x_{ij}\}_{j=1}^{N_i}$ to denote the set of N_i local feature vectors x_{ij} extracted from the i-th image.

3.3.1 Randomized greedy tree construction

To quantize or partition the feature space we use binary decision trees, in which each non-leaf node n has an associated split criterion $f(x;\theta_n,\tau_n)$ that determines for a given feature vector x whether it will proceed to the left child if $f(x;\theta_n,\tau_n)<0$, or to the right child otherwise. In particular we will use axis-aligned splits that threshold one element of the feature vector, *i.e.* we have $f(x;\theta_n,\tau_n)=x^\top\theta_n-\tau_n$, and θ_n is all-zero except one entry which equals 1. Clearly, other types of split functions are possible, *e.g.* linear functions without restrictions on θ, or generalized linear functions. Each node of the tree corresponds to a part of feature space: the root is associated with the complete feature space, child nodes being associated with a subset of the space associated with the parent. The set of leaf-nodes forms a partitioning of the complete feature space into non-overlapping subsets. Given a quantization tree with L leaf-nodes, an image X_i is then represented by an l_1 normalized histogram $h_i\in[0,1]^L$ that codes in dimension d the fraction of the N_i image regions associated with the d-th leaf.

To learn the trees suited for image categorization task we propose to follow a randomized greedy forward-selection procedure. We start with a trivial tree with just the root-node. Then, we expand the existing tree iteratively by adding two child-nodes to an existing leaf-node, thereby refining the current partitioning of the feature space. At each step, we sample T candidate splits per iteration, by uniformly selecting one of the existing leafs, and determine θ_n by sampling uniformly a feature dimension to split. Given these, we sample a region descriptor from descriptor set associated with selected leaf and and use its value on the selected dimension as the threshold τ_n. We evaluate each of the tentative expansions, and accept the best one using a criterion described below.

3.3.2 Evaluating splits for image categorization

The procedure outlined above is essentially the same as the one used in decision tree learning, where the leafs of the tree are used to classify the feature vectors processed by the tree. In that case, the branches of the tree can be grown independently since the design of the tree under one node will not affect classification performance of feature vectors that are assigned to its sibling node. Splits are typically evaluated using the information-gain criterion [Geurts *et al.*, 2006]. In Appendix A we show that information-gain criterion correponds to maximum likelihood learning for patch classification. Remember that our goal is not to use the leafs of the tree to classify individual feature vectors x_{ij}, but rather to classify images represented by sets of feature vectors $X_i=\{x_{ij}\}_{j=1}^{N_i}$ using a histogram h_i of leaf-assignments over the set. Therefore, in our case the branches of the tree can not be grown independently, and

we should use a criterion that evaluates a split directly by its impact on the image categorization performance. We illustrate the difference between the two approaches in Figure 3.1.

For each candidate expansion of the current tree, we update the image representations h_i by replacing the entry corresponding to the parent node by the histogram entries obtained for the two new child nodes. We then learn a score function over the new image representation, and evaluate it based on its performance on a validation set of images. We use linear score functions, as we have to evaluate many tentative splits: $T \times L$ in total, where L is the desired number of leaf-nodes in the final tree, equaling the number of expansion iterations.

Like much recent work on image categorization, we evaluate performance in terms of average precision (AP), rather than the classification rate for a given threshold on the score. Since we aim to optimize ranking performance, rather than classification performance, we learn a score function for ordinal regression where the goal is to ensure that for each pair of a positive and a negative image the score of the positive one is larger than the score of the negative one by some margin. If the score difference between a positive image h_i^+ and a negative image h_j^- is smaller than 1 we suffer the loss ξ_{ij}, otherwise $\xi_{ij} = 0$. In addition we use an ℓ_2 regularization term, which then leads to the following optimization problem [Joachims, 2005]:

$$\min_{w,\xi_{ij}\geq 0} \quad \frac{1}{2} w^\top w + C \sum_{i,j} \xi_{ij}, \tag{3.1}$$

$$\text{s.t.} \quad \forall_{i,j}: \quad w^\top (h_i^+ - h_j^-) \geq 1 - \xi_{ij}, \tag{3.2}$$

where i ranges over indices of positive examples, and j over indices of negative examples.

The number of constraints and slack variables is quadratic in number of images, but only a small fraction of all constraints is actually active — up to 15% in our experiments. The solution of the optimization problem can be obtained by cutting-plane methods [Kelley, 1960] that minimize the cost function subject to a subset of the constraints, and iteratively add constraints that are violated by the current solution. In practice, we initially minimize the cost function subject to a set of 1,000 randomly selected constraints, and the sequentially add all violated constraints and re-solve. Typically, all constraints are satisfied within three iterations.

Once a score function has been learned for each sampled expansion of the current tree, the T sampled splits are evaluated by the loss obtained on a set of validation images.

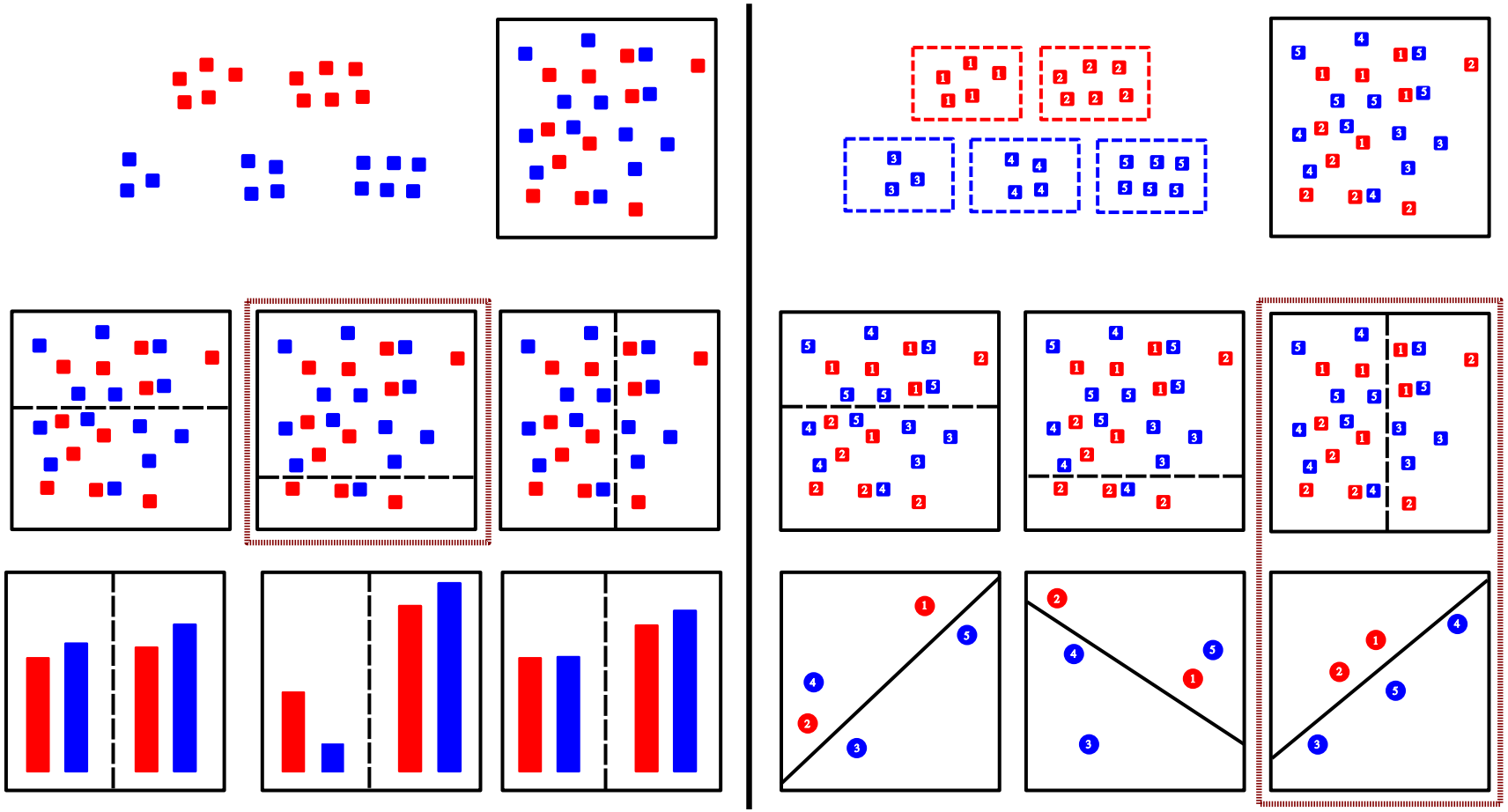

***Figure 3.1**: Illustration of tree quantization approach using approach of [Moosmann et al., 2008] (left) and our approach (right). First row: training image regions and its descriptors in two-dimensional feature space, for the selected node. Numbers denote the indices of the images the regions are sampled from. Positive training examples are denoted red, and negative by blue color. Second row: three tentative splits of feature space for the selected node. Third row left: maximum likelihood prediction of patch class for given the tentative splits. The second split is selected because it produces children which yeild maximal reduction of uncertaininty about patch class labels and therefore minimizes the patch classification loss. Third row right: two-dimensional image BoW histograms induced by tentative splits, along with the learned image classifier. Each split genertes two new dimensions in the histogram that are used to classify the images. The third split is selected because it minimizes the image classification loss.*

3.3.3 Learning ensemble of quantization trees

Due to the random and greedy nature of the tree construction algorithm, there are no guarantees that the above construction algorithm will find the optimal tree. Even when exhaustively considering all possible splits, which would be computationally infeasible, the split that currently improves the model most might not be optimal in combination with subsequent refinement of the quantization.

To address this problem, we have experimented with expanding nodes by more than two children, *i.e.* with a small tree, and evaluating the current split based on the future improvement that could be obtained by incorporating this split. Similar procedures have been used in decision tree learning before [Roizman & Last, 2006]. However, we did not observe significant improvements when employing this procedure.

Instead, we have found that using a "forest" of several trees does give significant improvements, as has also been observed before for randomized decision trees [Breiman, 2001]. Here we learn each tree independently, and then concatenate the histogram representations obtained using each tree to obtain our final image representation. Combining K trees of L leafs each we obtain a final representation of size $K \times L$. In addition to improving the results, using ensembles also significantly reduces the variance of performance.

3.4 Experimental evaluation

In this section we evaluate our approach and compare it to k-means quantization, and the tree quantizers of [Moosmann *et al.*, 2008]. In Section 3.4.1 we describe the data sets we use in the experiments, feature extraction, and implementation details. We present results for the Graz-02 data set in Section 3.4.2, and those for the 15-scenes data set in Section 3.4.3.

3.4.1 Data sets, features, and implementation details

The Graz-02 data set [Opelt & Pinz, 2005] contains 1476 images displaying instances of three objects classes: bicycles, cars, and people. In addition background images which contain none of these categories but show scenes similar to the ones that display the objects are included, Strong intra-class variations due to different category instances, *e.g.* different car models, are represented in the data set. Large variations in pose and scale make recognition challenging. As in [Fulkerson *et al.*, 2008], we use odd numbered images from each class for learning the quantizers and classification models, and we evaluate our model on the remaining images.

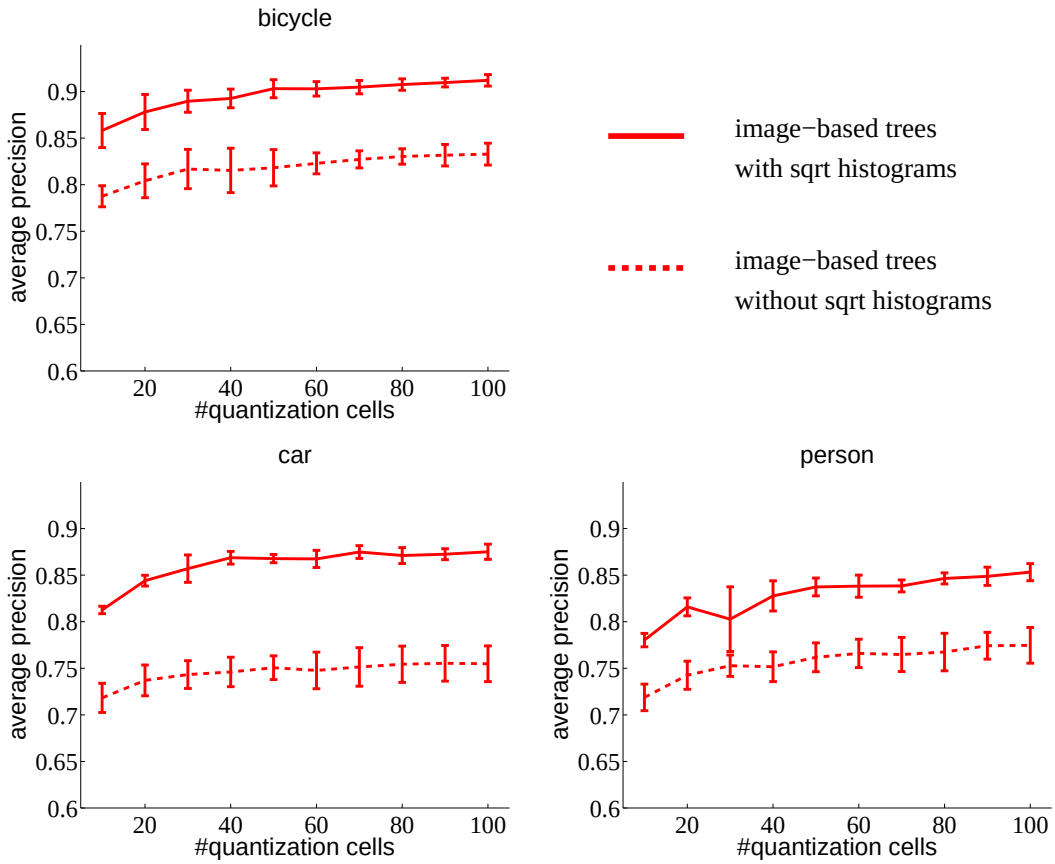

***Figure 3.2**: Performance for the classes in Graz-02, using image-based trees with $K = 10$ quantizers with a varying number of cells L. Square-rooting visual word histograms significantly improves the performance.*

The 15-scenes data set [Lazebnik *et al.*, 2006] contains 4485 images of various scene categories. Small inter-class variations between several indoor scenes are a major challenge of this dataset, *e.g. bedroom* vs. *livingroom*. As in [Lazebnik *et al.*, 2006] we use 100 randomly selected images per class for training, and use the remaining ones for evaluation. We repeat the procedure 10 times and report mean and standard deviation of average precision.

In all experiments we use the same image features. We sample regions of 20×20 pixels from a regular grid, spaced by 10 pixels. This is done at the original image scale, but also at four down-sampled versions of the image, rescaling it each time by a factor 1.2. For each patch we compute a SIFT descriptor [Lowe, 2004] of 128 dimensions. When training the linear score functions, we use element-wise square-rooted visual word histograms instead of the original ones. In [Perronnin *et al.*, 2010a] it was shown that this leads to significant performance increases for linear classifiers over bag-of-word histograms generated by k-means, while maintaining the efficiency of

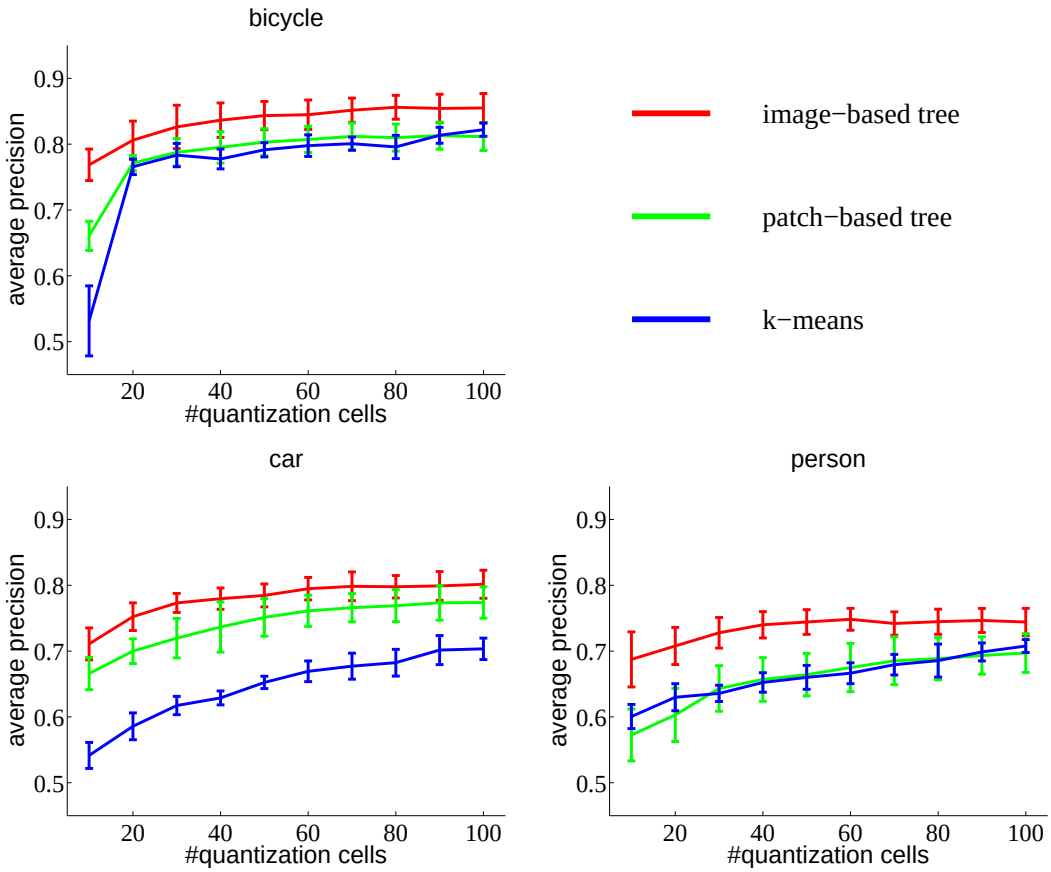

***Figure 3.3**: Performance for the classes in Graz-02, using one quantizer with a varying number of cells L using our image-based trees (red), the method of [Moosmann et al., 2008] (green), and k-means (blue).*

linear classification. We found that the same holds for histograms generated by tree structured quantizers, as shown in Figure 3.2.

When learning our quantization trees, we sample $T = 100$ splits at each iteration. Since we evaluate performance in terms of average precision, we determine the regularization parameter C in Equation 3.1 by 5-fold cross-validation to maximize average precision. Recall that we also use validation to prevent overfitting when growing the trees: for each tree we split the training data into two parts. We train the ordinal regression models on the first part, but select the split which minimizes the loss on the second part.

In our implementation of patch-based tree quantizers of [Moosmann *et al.*, 2008], we fix the desired number of leafs L in advance. When constructing the tree, we keep a priority queue of nodes to expand, ordered by the information gain that can be obtained by expanding them, and grow the tree in a best-first manner.

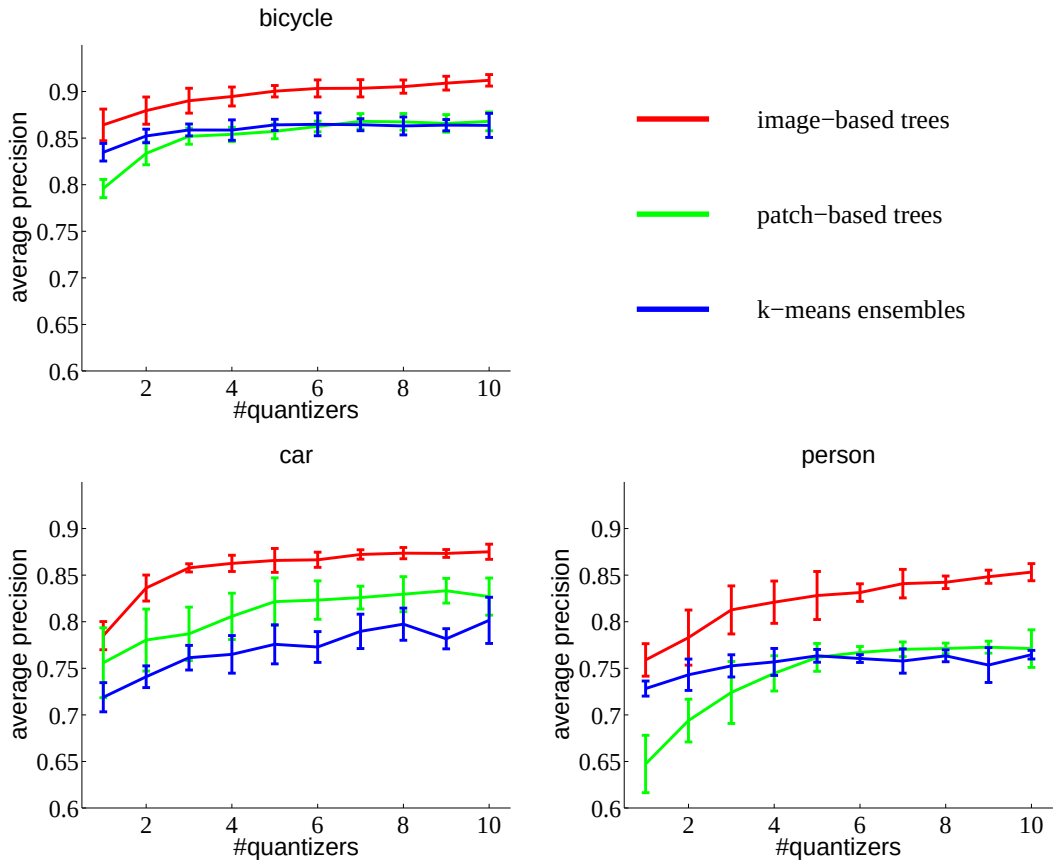

Figure 3.4: Performance for Graz-02 using $L = 100$ cells, and varying the number of quantizers K.

3.4.2 Results on the Graz-02 dataset

In our first set of experiments we consider performance as a function of the number of quantization cells L, ranging from 10 up to 100, with $K = 1$. Since k-means depends on its initialization, and the tree-based methods on the random selection of splits, we show the average and standard deviation of performance over 10 experiments.

From results in Figure 3.3 we can see that our image-based trees give best results for all classes and all numbers of quantization cells. The performance of k-means and patch-based trees are comparable to each other on the classes *bicycle* and *person*. To achieve similar performance our method needs much fewer quantization cells: on all classes using only 30-dimensional histograms our method outperforms the others when using up to 100-dimensional histograms.

In the second set of results in the Figure 3.4 we look at performance as a function of the number of quantizers K that is combined in an ensemble, in each case using $L = 100$ cells. We see that the performance of tree-structured quantizers is improved on all three classes. For k-means vocabulary ensembles are less effective, in particular

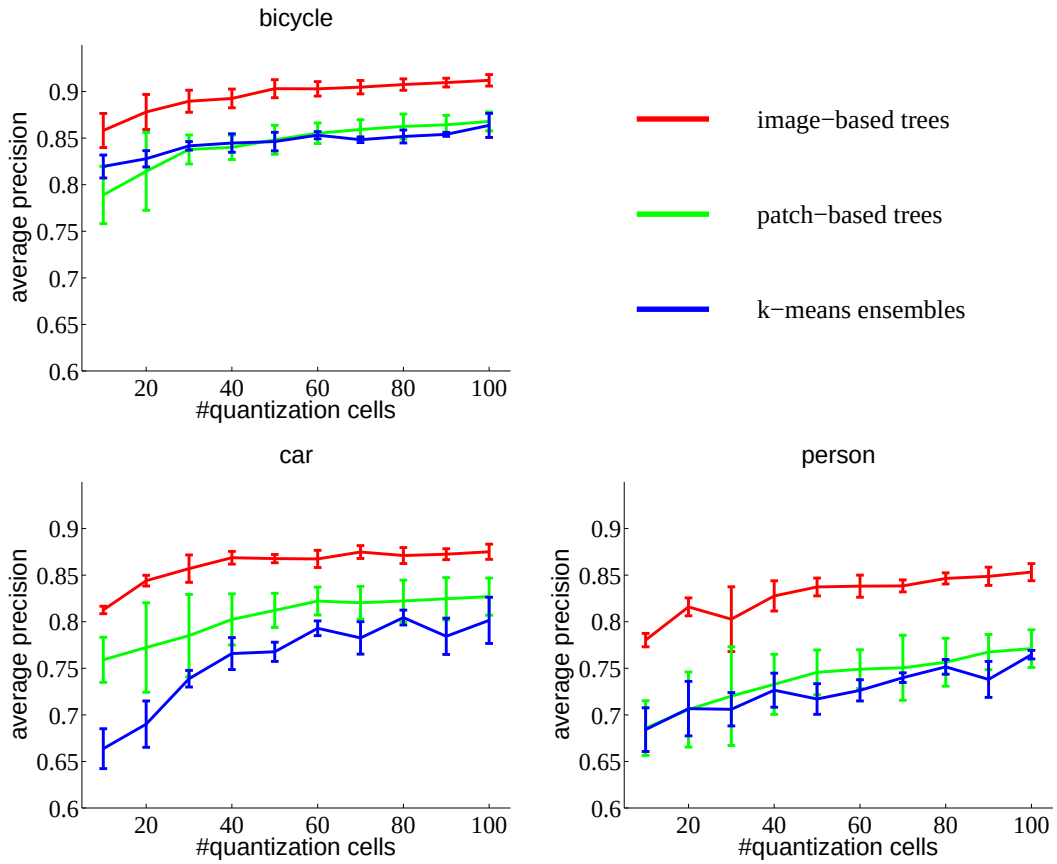

***Figure 3.5**: Performance for Graz-02 using ensembles of $K = 10$ quantizers while varying the number of cells L.*

for the class *bicycle*. For this class, k-means and our image-based trees perform similar with a single quantizer, but our method outperforms k-means by 7% AP when using $K = 10$ quantizers.

In the third set of results in the Figure 3.5 we again consider performance as a function of the number of quantization cells L, this time for ensembles of $K = 10$ quantizers. The results are similar to those of Figure 3.3: our method achieves very good performance using few quantization cells. Using only $L = 10$ cells our method outperforms k-means with $L = 100$ cells by a large margin. Our quantizers also significantly outperform those of [Moosmann *et al.*, 2008].

In Table 3.1, we compare ensembles of shallow patch-based and image-based trees with results obtained with much larger k-means vocabularies. Only for one class and using vocabularies of 4000 cells, the k-means results improve over the image-based trees.

In Figure 3.6five positive images with lowest scores and five negative images with highest scores are shown, for the best performing configuration ($K = 10$, $L = 100$)

AP×100	K	L	bicycle	car	person
image-based tree (ours)	10	100	**91.2 ± 0.6**	**87.5 ± 0.8**	85.3 ± 0.9
patch-based tree Moosmann *et al.* [2008]	10	100	86.8 ± 1.0	82.7 ± 2.0	77.1 ± 2.0
k-means	1	1000	88.3 ± 1.9	81.1 ± 0.8	83.1 ± 0.7
k-means	1	2000	90.0 ± 0.3	83.1 ± 0.6	85.6 ± 0.4
k-means	1	4000	90.7 ± 0.3	84.8 ± 1.2	**87.3 ± 0.4**

***Table 3.1**: Comparison to larger k-means vocabularies on Graz-02.*

using our model. The main problems in case of positive images are change in image orientation, object occlusions and small object size. To gain more insight into nature of top-ranked negative images we show in Figure 3.7 *score maps*: for each pixel the patches that contain it are assigned to leaves of the tree and associated with the corresponding weight of the learned model. The value for each pixel is obtained as average of the assigned weights. Only top 10% pixels with highest scores are displayed. This way we can identify the parts of the image that contribute the most to positive score of the image. We see that in the case of class *bicycle* the presence of wired structures or facade textures causes confusion because they locally resemble the parts of the bicycles. It is interesting to note that the same image belonging to the background class is among top five negative images for both class *bicycle* and class *car*. However, from score maps we can see that in the case of class *bicycle* the confusion is due to presence of the motorbike, while in case of class *car* it is because some cars are actually present in the image. In Figure 3.8 we show score maps for top five images for each class. All images are positive, and the most heavily weigted patches actually correspond to parts of object whose model is learnt.

3.4.3 Results on the 15-scenes dataset

In Figure 3.9 we present graphs similar to those presented before, in this case showing the mean average precision over all 15 classes. The standard deviation is measured over the mAP values obtained using ten random training sets. Here, using a single quantizer (left panel) our image-based trees give better performance than the other quantizers using $L < 50$ cells, and slightly worse using $L > 50$ cells. For this data set we also observe (middle panel) that the k-means vocabularies benefit least from ensembles: using more than two qunatizers has a marginal impact on performance. The tree-based quantizers, on the other hand, continue to improve with the number of quantizers in the ensembles. Using ensembles of $K = 10$ quantizers, and varying the number of cells L (right panel), we see that the k-means quantizer quickly improves with the number of cells, but its performance clearly remains behind as compared to that of the tree-based quantizers.

In Figure 3.10 we show a more detailed comparison of performance in terms of AP for all 15 classes, using ensembles of $K = 10$ quantizers and $L = 100$ cells. For most

***Figure 3.6**: Five positive images with lowest scores (first row) and five negative images with highest scores (second row) for classes in Graz-02 dataset. Under each image the rectangle displays the proportion of positive images (blue) and relative position of image in the ranking (red).*

***Figure 3.7**: Five negative images with highest scores (first row) and their score maps (second row) for classes in Graz-02 dataset. The score map displays the parts of the image most heavily weighted for the class prediction.*

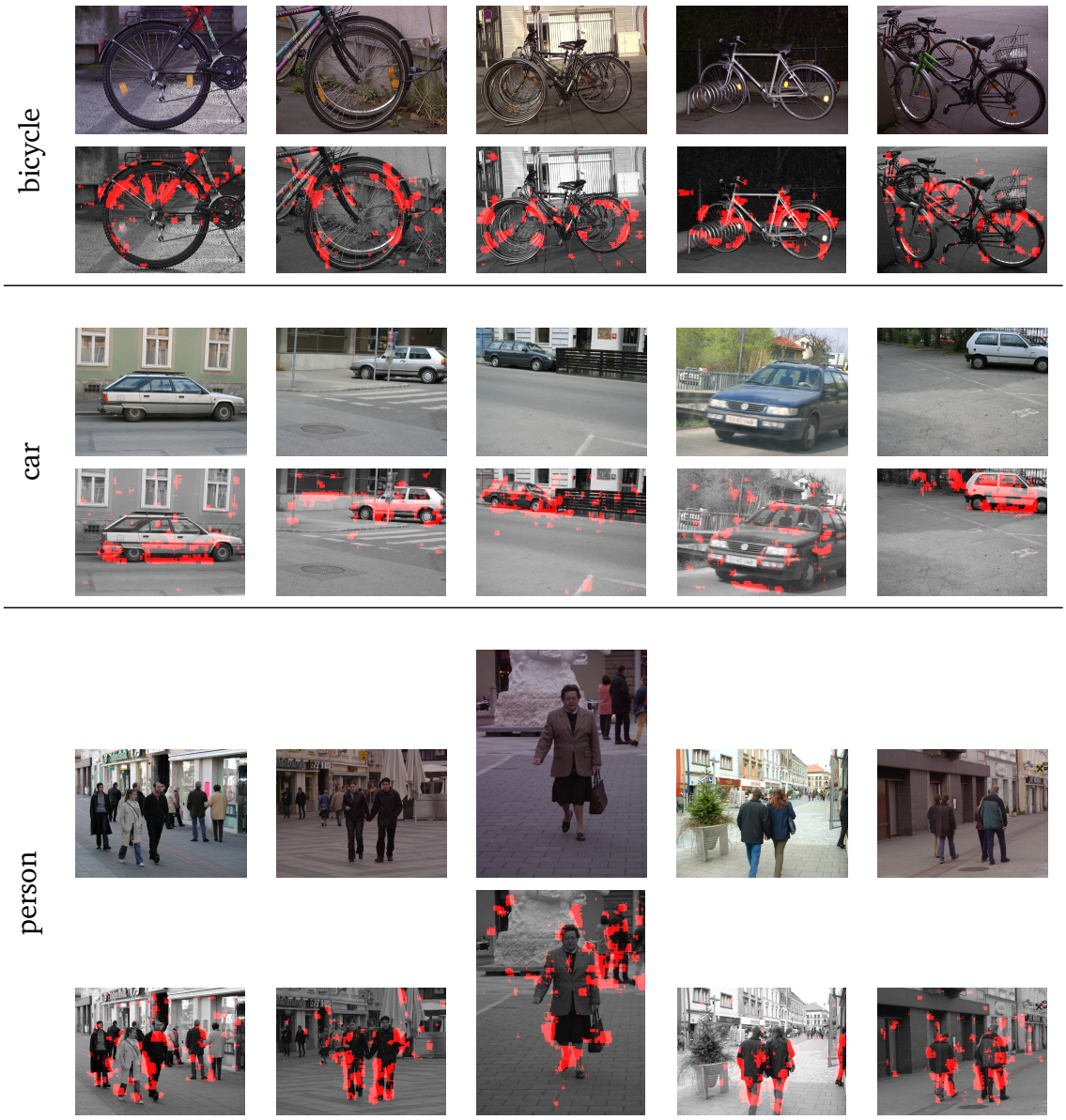

Figure 3.8: *Five images with highest scores (first row) and their score maps (second row) for classes in Graz-02 dataset. For all classes the top five images are positive. The image parts most heavily weighted for the class prediction correspond to locations of objects in the image.*

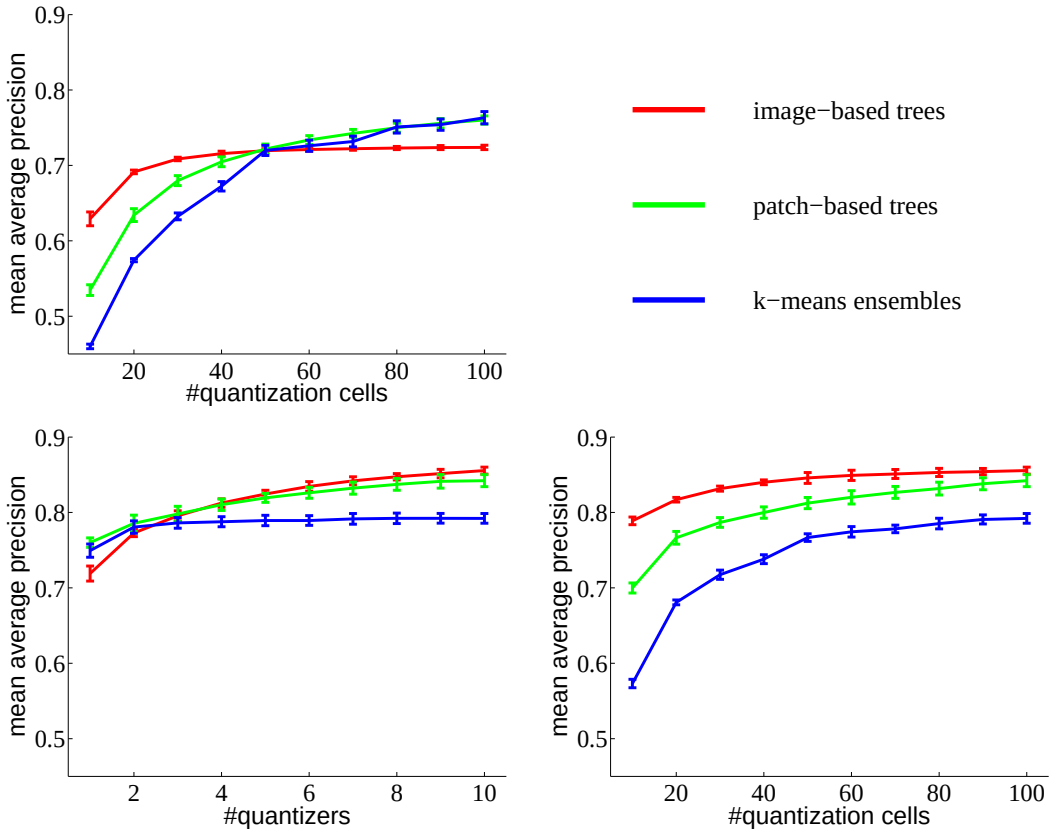

***Figure 3.9**: Performance in mAP for the 15-scenes data set: (left) using a single quantizer, and varying the number of cells L, (middle) using $L = 100$, and varying the number of quantizers K, (right) using $K = 10$ quantizers in the ensemble, and varying the number of cells L.*

classes image-based and tree-based quantizers perform similar, but larger improvements are observed for the more difficult classes such as *bedroom, livingroom*, and *opencountry*. On all classes k-means quantization gives the worst results.

The difference between patch-based trees of [Moosmann *et al.*, 2008] and our image-based trees on this dataset are smaller than on the Graz-02 dataset. This might be explained by the fact that in Graz-02 the different object classes appear against similar backgrounds, where in the 15-scenes data set the complete image contains discriminative information. Therefore, on Graz-02 when learning patch-based decision trees there will be many background patches, *e.g.* of buildings, that appear in all classes. Hence the decision tree will spend a large part of its capacity to separate buildings appearing as background of cars from buildings that appear as background of bicycles. The image-based trees do not suffer from this problem, since they are optimized for image classification, and it suffices to isolate a few informative features of the object class to obtain good classification results. On the 15-scenes data set, however, the class specific features are not hidden in a large pool of background features, and

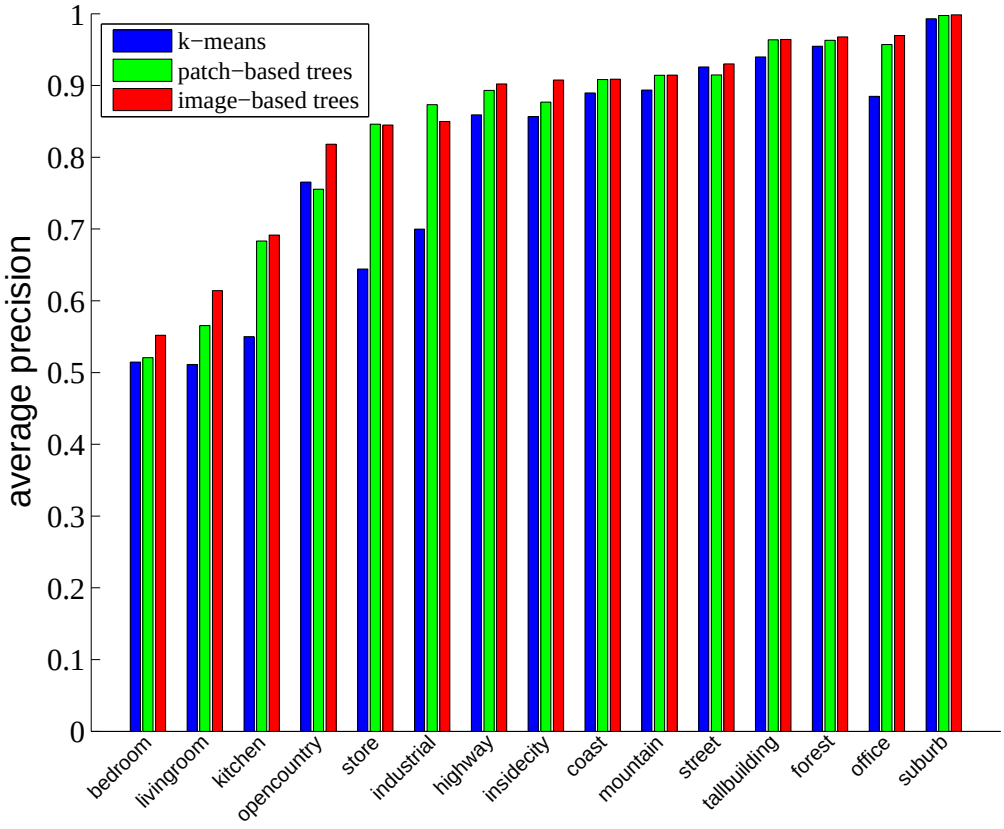

***Figure 3.10**: Per class performance comparison on 15-scenes dataset.*

the patch-based trees also learn effective quantizers, without modeling the various backgrounds.

In order to compare our results to others that report multi-class classification accuracies we trained multi-class logistic regression models as follows. First, using one half of the training data, we learn a forest of K trees for each of the $C = 15$ classes in a one-versus-all manner as before. Then, we concatenate the K histograms of size L of all classes, yielding a representation of size $C \times K \times L$. We then learn a multi-class logistic discriminant classifier over this representation using all training data.

In Table 3.2 we report for various configurations of k-means, patch-based trees, and our image-based trees the recognition rates, and include mAP values for reference. Due to the quite regular spatial layout of the 15-scenes classes, the use of spatial pyramids [Lazebnik *et al.*, 2006] might lead to better performance. For sake of clarity, however, we did not use them here. When using $K = 10, L = 10$ we obtain results comparable to k-means with $L = 1000$ cells, and using $K = 10$ and $L = 100$ we improve our accuracy by 3.6% to $83.6\% \pm 0.6$, This is clearly better than using k-means, and also better than the result of 81.4 ± 0.5 reported in [Lazebnik *et al.*, 2006] which introduced spatial pyramids over k-means histograms. Using $K = 10$ and $L =$

	K	L	accuraccy	mAP
k-means	10	10	59.7 ± 0.4	57.3 ± 0.6
patch-based tree	10	10	79.0 ± 0.9	70.0 ± 0.7
image-based tree	10	10	80.0 ± 0.9	78.9 ± 0.5
k-means	10	100	73.7 ± 0.8	79.2 ± 0.7
patch-based tree	10	100	**83.9 ± 0.6**	84.2 ± 0.8
image-based tree	10	100	**83.6 ± 0.6**	**85.6 ± 0.5**
k-means	1	1000	80.5 ± 0.7	84.0 ± 0.8
Lian *et al.* [Lian *et al.*, 2010]	105	50	78.1 ± 0.7	–

***Table 3.2**: Summary of the results on 15-scenes dataset.*

100, our results compare favorably to those reported in [Lian *et al.*, 2010], which are obtained by learning quantizers for multi-class classification, and combining $K = 105$ quantizers with $L = 50$ cells. Using ensembles of very shallow trees, $K = 10$ and $L = 10$, our results are already better than theirs.

Finally, we point out that even when an equal number of histogram cells is used, the histogram generation process requires several orders of magnitude less operations in our method as compared to k-means. For k-means, we need to compute distances to the $K \times L$ centers, each computed by D operations for a D dimensional descriptor; for SIFT descriptors with $D = 128$, $K = 1$ and $L = 1000$ this adds up to 128,000 operations per image patch. Using our tree-structured quantizers, we only need to compute approximately $\log L$ thresholds for each of the K trees, one threshold on each level of the tree. For $K = 10$ and $L = 100$ this results in applying only 70 thresholds.

3.5 Conclusion

We have introduced a method to learn tree-structured quantizers using a randomized greedy forward selection procedure, aimed at maximizing image classification performance. We found that combining several such trees in an ensemble leads to significant performance increases. The assignment of descriptors to histogram entries is much faster in the tree-structured quantizers than in k-means, since each node of the tree only applies a threshold on a single dimension of the descriptor. Our experimental results on two data sets show that our method improves over k-means quantization, and trees learned in a patch-based manner.

In future work we want to address the design of vocabularies that are shared across different classification tasks, both in multi-class settings as in settings with multiple binary labels. Secondly, we want to explore an approach where a forest of several trees is grown concurrently, so that they are trained to be mutually complementary.

4

Modeling Spatial Layout with Fisher Vectors for Image Categorization

Contents

4.1 Introduction

In this chapter we adress image representations that encode the spatial layout of image regions. We propose a new image representation that generalizes the state-of-the-art method for encoding the spatial layout of the regions. Our representation is much more compact than alternatives, yielding the same performance for image categorization task using simple linear classifiers. This work has been submitted to International Conference on Computer Vision 2011.

As already mentioned in the previous chapter, image categorization aims to determine the presence of objects in images, or to recognize them as particular scene types such as *city*, *mountain*, or *beach*. Current state-of-the-art image categorization systems use bag-of-word image representations, pioneered in [Csurka *et al.*, 2004, Sivic & Zisserman, 2003]. Using this approach the content of the image is represented by global statistics of the appearance of local image regions. First, image regions are sampled from the image, either using a regular grid, in a randomized manner [Nowak *et al.*, 2006], or using interest point detectors [Zhang *et al.*, 2007]. Each region is then described using a feature vector, *e.g.* SIFT [Lowe, 2004] or color histograms [van de

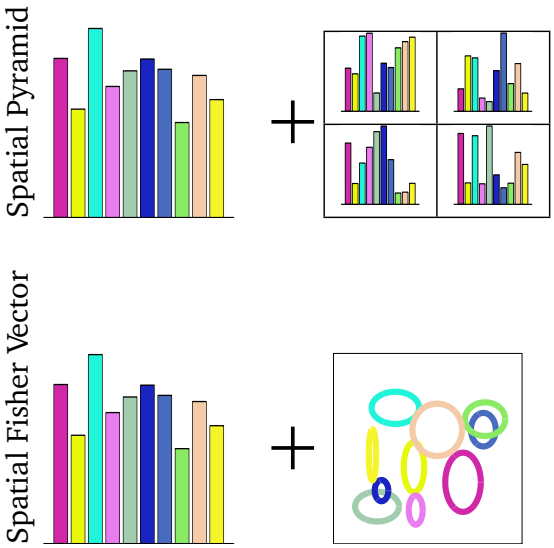

***Figure 4.1**: The spatial pyramid image representation concatenates visual word histograms of the complete image and spatial cells. Our spatial Fisher vector representation models spatial layout by the mean and variance of the occurrences of each visual word.*

Weijer & Schmid, 2006]. A visual vocabulary is then learned using k-means or a mixture of Gaussians (MoG), modeling patches sampled from many training images. The visual vocabulary quantizes the feature space into different cells, and region features are assigned to these cells. Therefore, each region feature is described by quantization index, either by hard-assigning the feature to quatization cell, or using a soft-assigment to a component of a MoG model. The patch assignments are then averaged to obtain the global image representation, which is essentially a histogram with as many bins as visual words, where each bin gives the number of patches assigned to that visual word. This way the image represented by a set of regions is embedded into vector space in which image classification is performed, by learning a classifier.

Several extensions to the basic bag-of-words image representation have been proposed; we will discuss the most relevant ones in detail in the next section. A recent extension to the bag-of-words model is the Fisher kernel image representation [Perronnin & Dance, 2007]. Instead of only storing the average (soft-)assign of patches to visual words, the first and second order moments of patch feature distribution for patches assigned to each visual word are also stored. This means that, for a descriptor of size D and K visual words, the image representation is of size $K(1+2D)$. Since much more information is stored per visual word, a much smaller number of visual words can be used for a given level of categorization performance, which has a clear computational advantage.

Another extension is the spatial pyramid representation of [Lazebnik *et al.*, 2006] which captures the information about the spatial layout of the image by computing bag-of-word histograms over different regions of the image, and concatenating these to form the final representation. Using K visual words and C spatial cells results in image representation of size KC. The same idea applied to the Fisher kernel image representation [Perronnin *et al.*, 2010b], leads to a representation of size $KC(1+2D)$. This representation has been proven to be effective, in particular when the image categories exhibit characteristic layouts, as in the case of the scene recognition. For object categorization this idea is also effective because even though the objects may appear anywhere in the image, the scenes in which they appear may still have strong layout patterns.

In this chapter we propose an alternative method to encode spatial layout information, based on the Fisher kernel principle [Jaakkola & Haussler, 1999], that previously was only used to encode the appearance information [Perronnin & Dance, 2007]. We model the spatial location of the image regions assigned to each visual words using MoG models, and compute the Fisher kernel representation from these models. Compared to using spatial pyramids, we obtain representations that are smaller, while not degrading performance. Using bag-of-word for appearance, our representations are smaller and achieve better performance on 15-Scenes and PASCAL VOC 2007 data sets, using linear classifiers, as compared to using the spatial pyramid representation with the non-linear intersection kernel. When we use the Fisher kernel principle to encode both spatial and appearance information we obtain very effective image representations: they are efficiently computed as compared to using k-means vocabularies, and are much more compact as compared to applying spatial pyramids to Fisher kernel appearance models, while acheiving similar performance. See Figure 4.1 for a schematic comparison of our approach to spatial pyramids.

In the next section we discuss the most relevant related work, and then present our image representations in Section 4.3. We present extensive experimental results in Section 4.4, comparing different variants of our image representations to alternatives from the literature. Finally, we present our conclusions in Section 4.5.

4.2 Related work

Because of its effectiveness, the bag-of-words (BoW) model has become one of the most popular representations for image categorization since its introduction in the seminal papers [Csurka *et al.*, 2004, Sivic & Zisserman, 2003]. Subsequent research has focused on overcoming its two intrinsic limitations, namely (a) the computational cost of the assignment of local features to visual words, and (b) the lack of information on the spatial layout of the local features.

Quantization issues and codebook compactness. Performance of the BoW model has been unanimously reported to increase with the size of the dictionary [Csurka *et al.*, 2004, Sivic & Zisserman, 2003, van Gemert *et al.*, 2010a] and the number of regions sampled from images [Nowak *et al.*, 2006]. Typically, vocabularies of several thousands codewords are used, and thousands of regions are densely sampled from the images. Assigning local features to their nearest visual word is computationally very expensive, as it scales as the product of the number of visual words, the number of regions, and the local feature dimensionality. These issues have been addressed by different authors, *e.g.* [Nistér & Stewénius, 2006] proposed a hierarchical k-means framework (scaling logarithmically with the number of codewords), while [Philbin *et al.*, 2007] introduced an approximate k-means algorithm better suited to the use of large vocabularies. Random forests, because of their hierarchical structure, are also good candidates for handling large visual vocabularies [Bosch *et al.*, 2007a, Moosmann *et al.*, 2007].

Nevertheless, the simplest way to reduce the time spent in assigning features to visual words is certainly to make the vocabulary smaller, of course without losing performance. Different authors have tried to build compact discriminative vocabularies [López-Sastre *et al.*, 2011, Moosmann *et al.*, 2007, Yang *et al.*, 2008], *i.e.* vocabularies that are specialized in representing the differences between categories. One of the most convincing approaches is the one by Perronnin et al. [Perronnin *et al.*, 2006]. However, these vocabularies are not universal since they have to be rebuilt each time a new category is added, which is a severe drawback.

On the other hand, when the vocabularies are more compact, the information lost in the quantization process becomes more important, in particular when using hard assigment [van Gemert *et al.*, 2010a]. The amount of discriminative information is considerably reduced due to the rough quantization of the feature space, as clearly shown by [Boiman *et al.*, 2008] who propose to compute direct image-to-class distances without descriptor quantization. The loss of information can be compensated by assigning descriptors to multiple visual words, as suggested by [Philbin *et al.*, 2008, van Gemert *et al.*, 2010a]. The assignment can also be guided by sparsity constraints [Yang *et al.*, 2009] or locality constraints [Wang *et al.*, 2010]. However, these approaches again require large codebooks, *e.g.* 2048 visual words in [Wang *et al.*, 2010].

Regarding the production of compact vocabularies, one of the most appealing approach is the one proposed in [Perronnin & Dance, 2007]. They have suggested to use the Fisher kernel framework [Jaakkola & Haussler, 1999], whose high dimensional gradient representation contains much more information than a histogram representation, resulting in informative and compact vocabularies.

Spatial information. The BoW representation is a histogram of vector quantized local appearances, and the spatial layout of the appearances is completely ignored.

Clearly, the spatial information may convey useful cues for image categorization, and at least two different ways to encode spatial information have been explored: based on pairwise positions of features, and using absolute positions.

Considering pairs of spatially close image regions is probably the most intuitive way to incorporate spatial information. Visual word "bigrams" are considered in [Savarese *et al.*, 2006], by forming a bag-of-word representation over spatially neighboring image regions. Others have proposed a more efficient feature selection method based on boosting which progressively mines higher-order spatial features [Liu *et al.*, 2008], and [Morioka & Satoh, 2010] proposes joint feature space clustering to build a compact local pairwise codebook. Distinctive spatial configurations of visual words can also be discovered by data mining techniques, such as frequent itemsets [Quack *et al.*, 2007].

In addition to pairwise relationships, images often have spatial biases: the composition of the pictures of particular object or scene category typically share common layout properties. Therefore, embedding the global positions of the features in the image is effective in many cases. Spatial Pyramid Matching (SPM) [Lazebnik *et al.*, 2006] exploits this property by partitioning the image into increasingly finer cells and concatenating the BoW histograms of the cells. This strategy is used in most of the state-of-the-art approaches, see *e.g.* [Perronnin *et al.*, 2010b, Yang *et al.*, 2010a]. In [Bosch *et al.*, 2007b] SPM is further improved by learning a weighting of the levels of the SPM representation on a validation set. The idea of implicitly representing spatial information by weighting image cells based on their discriminative power was explored earlier in the context of facial expression recognition in [Shinohara & Otsu, 2004], where linear discriminant analysis was used to find a weighting of the spatial cells. In addition to global spatial information, they also used local auto-correlation measurements to include local spatial information. Recently, a similar strategy was applied to address image categorization in [Harada *et al.*, 2010], which yielded results comparable to the state-of-the-art on the 15-Scenes data set.

More closely related to our work, [Zhou *et al.*, 2009] models regions appearances with a mixture of Gaussian (MoG) density, and uses the posterior over visual words for the image regions to form so called "Gaussian maps". Then then apply SPM to encode the spatial occurrence of visual words in the image. Our approach is similar, as we also use a MoG to model the region appearances and also incorporate spatial layout based on coding the region locations of each visual word. However, different from their approach, we use the more efficient Fisher kernel [Jaakkola & Haussler, 1999, Perronnin & Dance, 2007] approach to jointly code appearance and spatial layout, giving efficient, compact, and discriminative image representations.

4.3 Fisher kernels to encode spatial layout

In this section we present our models to encode both the spatial layout of local image features and their visual appearance. In Section 4.3.1 we start by rephrasing the common bag-of-word (BoW) image representation as a simple probabilistic model, for which we derive a Fisher vector representation. We then extend this model in Section 4.3.2 by including a simple Gaussian location model, and further extend the spatial model to a mixture of Gaussians (MoG) in Section 4.3.3. We integrate our spatial models with MoG appearance models in Section 4.3.4, essentially combining Fisher vector representations for both appearance and spatial layout. Finally, we consider normalization of the Fisher vectors in Section 4.3.5, and we compare the different models we introduced to spatial pyramid image representations in Section 4.3.6.

4.3.1 A generative model view on bag-of-words

The BoW image representation uses k-means to quantize the space of patch appearances, for each patch $\boldsymbol{x}_n$ we use $w_n \in \{1,\ldots,K\}$ to denote the index of the k-means center that is closest to $\boldsymbol{x}_n$, among the K centers. The trivial probabilistic model over the quantization indices is just a multinomial $\boldsymbol{\pi}$, and the likelihood of observing the k-th quantization index is given by $p(w_n = k) = \pi_k$. The parameters of this multinomial are fitted from the data used to learn the k-means quantizer, and are simply given by the fraction of the patches assigned to each visual word.

To apply the Fisher kernel framework [Jaakkola & Haussler, 1999], we consider the average log-likelihood of the N patches in an image, given by

$$\mathscr{L} = \frac{1}{N}\sum_n \ln p(w_n). \tag{4.1}$$

The average is taken to achieve invariance *w.r.t.* number of patches N in the image. We parameterize the multinomial using a softmax by defining $\pi_k = \exp\alpha_k / \sum_j \exp\alpha_j$, which by construction satisfies the constraints $\pi_k \geq 0$, and $\sum \pi_k = 1$ for any setting of the α_k. The gradient is then given by

$$\frac{\partial \mathscr{L}}{\partial \alpha_k} = h_k - \pi_k, \tag{4.2}$$

where h_k is the frequency of the k-th visual word in the image, *i.e.* its count devided by N.

We recognize this gradient of size K as the standard bag-of-word frequency minus the multinomial parameters, which have been learned from the vocabulary training data. By shifting BoW histograms h_k by π_k the average BoW histogram is centered to the origin.

4.3.2 A simple Gaussian spatial model

We extend the appearance-only bag-of-words model by introducing a Gaussian location model per visual word. Each image patch is represented as the tuple $\boldsymbol{f} = (w, \boldsymbol{l})$, where w is the quantization index and $\boldsymbol{l}$ gives the spatial location of the patch in the image. We define a generative model over appearance-location tuples as

$$p(\boldsymbol{f}) = p(w)p(\boldsymbol{l}|w), \tag{4.3}$$
$$p(w = k) = \pi_k, \tag{4.4}$$
$$p(\boldsymbol{l}|w = k) = \mathcal{N}(\boldsymbol{l}; \boldsymbol{m}_k, \boldsymbol{S}_k), \tag{4.5}$$

where $\mathcal{N}(\cdot; \boldsymbol{m}_k, \boldsymbol{S}_k)$ denotes the Gaussian location model with mean $\boldsymbol{m}_k$ and covariance matrix $\boldsymbol{S}_k$ associated with the k-th visual word. The location models can be learned trivially by computing the mean and variance of the spatial coordinates of image patches assigned to the k-th visual word in the vocabulary training data.

We assume that variables that describe the patch position are uncorrelated so we use diagonal covariance matrices. The gradient of the log-likelihood of a patch $\boldsymbol{f}_n$ is

$$\frac{\partial \ln p(\boldsymbol{f}_n)}{\partial \alpha_k} = q_{nk} - \pi_k, \tag{4.6}$$
$$\frac{\partial \ln p(\boldsymbol{f}_n)}{\partial \boldsymbol{m}_k} = q_{nk}\boldsymbol{S}_k^{-1}\boldsymbol{l}_{nk}, \tag{4.7}$$
$$\frac{\partial \ln p(\boldsymbol{f}_n)}{\partial \boldsymbol{S}_k^{-1}} = q_{nk}\left(\boldsymbol{S}_k - \boldsymbol{l}_{nk}^2\right)/2, \tag{4.8}$$

where $q_{nk} = 1$ if $w_n = k$ and $q_{nk} = 0$ otherwise, $\boldsymbol{l}_{nk} \equiv \boldsymbol{l}_n - \boldsymbol{m}_k$, and $\boldsymbol{l}_{nk}^2$ denotes the element-wise square. The last equation gives the gradient w.r.t. the diagonal of the inverse covariance matrix. We slightly abuse the notation: in Equation 4.8 $\boldsymbol{S}_k^{-1}$ and $\boldsymbol{S}_k$ denote the vector of diagonal elements of the (inverse) covariance matrix.

By averaging the gradients over all patches in an image, this yields an image descriptor of size $K(1 + 2d)$, where $d = 2$ is the dimension of the location $\boldsymbol{l}$. For each visual word we have 1 element for the gradient w.r.t. the α_k, and 4 for the gradient w.r.t. the spatial mean $\boldsymbol{m}_k$ and variance $\boldsymbol{S}_k$.

4.3.3 A spatial mixture of Gaussian model

We extend the spatial model by using an MoG distribution over the patch locations instead of a single Gaussian, *i.e.* we replace Equation 4.5 with

$$p(\boldsymbol{l}|w = k) = \sum_{c=1}^{C} \theta_{kc}\mathcal{N}(\boldsymbol{l}; \boldsymbol{m}_{kc}, \boldsymbol{S}_{kc}), \tag{4.9}$$

using a mixture of C Gaussians to model the spatial locations of the patches per visual word. We define the mixing weights again using the softmax as $\theta_{kc} = \exp \beta_{kc} / \sum_j \exp \beta_{kj}$. The spatial model of each visual word can be learned using the EM algorithm [Bishop, 2006] from the patch locations associated with each visual word.

The gradient w.r.t. the α_k remains as in Equation 4.6, but for the location model parameters we obtain

$$\frac{\partial \ln p(\boldsymbol{f}_n)}{\partial \beta_{kc}} = q_{nk} \left(r_{nkc} - \theta_{kc} \right), \tag{4.10}$$

$$\frac{\partial \ln p(\boldsymbol{f}_n)}{\partial \boldsymbol{m}_{kc}} = q_{nk} r_{nkc} \boldsymbol{S}_{kc}^{-1} \boldsymbol{l}_{nkc}, \tag{4.11}$$

$$\frac{\partial \ln p(\boldsymbol{f}_n)}{\partial \boldsymbol{S}_{kc}^{-1}} = q_{nk} r_{nkc} \left(\boldsymbol{S}_{kc} - \boldsymbol{l}_{nkc}^2 \right) / 2, \tag{4.12}$$

where $\boldsymbol{l}_{nkc} = \boldsymbol{l}_n - \boldsymbol{m}_{kc}$ and $r_{nkc} = p(c|\boldsymbol{l}_n, w_n = k) = \theta_{kc} \mathcal{N}(\boldsymbol{l}_n; \boldsymbol{m}_{kc}, \boldsymbol{S}_{kc}) / p(\boldsymbol{l}_n | w_n = k)$. The r_{nkc} can be interpreted as a "spatial soft-assign" of patches of visual word k to the spatial mixture components. The image representation has size $K + KC(1 + 2d)$, K dimensions for the appearance part, and $KC(1 + 2d)$ for the spatial layout.

4.3.4 Mixture of Gaussians appearance models

We now combine the ideas from the previous section with a mixture of Gaussians (MoG) model for the patch appearances, and use Fisher vectors to obtain the image representations. The parameters of the models defined in this section can all be learned using the EM algorithm.

Appearance-only Fisher vector image representation. First, we define the appearance-only model as in [Perronnin & Dance, 2007]; the patch appearances $\boldsymbol{x} \in \mathbb{R}^D$ are modeled as

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}|w = k) \tag{4.13}$$

$$p(\boldsymbol{x}|w = k) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.14}$$

where π_k denotes the mixing weight of the kth Gaussian in the mixture, defined using the softmax as above. Similarly to the spatial models, for the appearance models we also use diagonal covariance matrices, therefore the appearance representation has size $K(1 + 2D)$.

Redefining q_{nk} to denote the posterior $p(w_n = k|\boldsymbol{x}_n)$, or responsibility, and $\boldsymbol{x}_{nk}$ to denote $\boldsymbol{x}_n - \boldsymbol{\mu}_k$, the gradients of the log-likelihood for a single patch are

$$\frac{\partial \ln p(\boldsymbol{x}_n)}{\partial \alpha_k} = q_{nk} - \pi_k, \tag{4.15}$$

$$\frac{\partial \ln p(\boldsymbol{x}_n)}{\partial \boldsymbol{\mu}_k} = q_{nk}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x}_{nk}, \tag{4.16}$$

$$\frac{\partial \ln p(\boldsymbol{x}_n)}{\partial \boldsymbol{\Sigma}_k^{-1}} = q_{nk}\left(\boldsymbol{\Sigma}_k - \boldsymbol{x}_{nk}^2\right)/2. \tag{4.17}$$

The image representation is obtained by averaging these gradients over all patches in the image. This representation has the computational advantage that we can use smaller number of visual words, since the appearance per visual word is coded more precisely [Perronnin & Dance, 2007].

Gaussian spatial models with MoG for appearance. When we include a single Gaussian spatial model, the appearance-location tuple $\boldsymbol{f} = (\boldsymbol{x}, \boldsymbol{l})$ is modeled as

$$p(\boldsymbol{f}) = \sum_k \pi_k p(\boldsymbol{x}|w = k)p(\boldsymbol{l}|w = k), \tag{4.18}$$

where $p(\boldsymbol{l}|w = k)$ is defined as in Equation 4.5, and $p(\boldsymbol{x}|w = k)$ as in Equation 4.14.

If we redefine $q_{nk} = p(w_n = k|\boldsymbol{x}_n, \boldsymbol{l}_n)$, the gradients with respect to the $\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the same as in Equation 4.15–Equation 4.17, and those for the $\boldsymbol{m}_k, \boldsymbol{S}_k$ are the same as in Equation 4.7–Equation 4.8, albeit using the current definition of q_{nk}. The image representation has size $K(1 + 2D + 2d)$ in this case. Note that since the patch descriptor $\boldsymbol{x}$ is generally high dimensional, *e.g.* 128 for SIFT, the additional $2d = 4$ dimensions increase the representation size only slightly as compared to the MoG appearance-only model.

Using MoG spatial models with MoG for appearance. In this case we use the model of Equation 4.18, with the MoG spatial model $p(\boldsymbol{l}|w = k)$ of Equation 4.9. The model now has $K(1 + 2D)$ parameters for the appearance model, and $KC(1 + 2d)$ for the spatial models. So in total we have $K(1 + 2D) + KC(1 + 2d)$ parameters.

The gradients with respect to the appearance parameters $\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ remain as in (4.15)—(4.17). For spatial parameters $\beta_{kc}, \boldsymbol{m}_{kc}, \boldsymbol{S}_{kc}$ the gradients are the same as in Equations (4.10)—(4.12) using the current definition of q_{nk}.

4.3.5 Normalization

The Fisher kernel framework of [Jaakkola & Haussler, 1999] requires multiplication of the gradient vectors with $\boldsymbol{F}^{-1/2}$ where $\boldsymbol{F} = \mathbb{E}_x\left[\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^\top\right]$ is the Fisher information matrix and $\boldsymbol{g}(\boldsymbol{x})$ denotes the gradient vector. Instead of using an analytical approximation as in [Perronnin & Dance, 2007], we use the observation that the multiplication with $\boldsymbol{F}^{-1/2}$ corresponds to a whitening of the gradient vectors [Bishop, 2006]. Based on the patches used for vocabulary construction, we compute the mean and variance on each dimension of the gradient vectors to obtain an additive and multiplicative normalizer, so that the normalized gradient vectors are zero-mean and unit-variance. The motivation for the assumption of diagonal $\boldsymbol{F}$ is similar as in [Perronnin & Dance, 2007]: in some cases $\boldsymbol{F}$ may be quite large, which renders computation of $\boldsymbol{F}^{-1/2}$ very costly, especially when using large K and when modeling spatial layout with SPM. For example, with $K = 200$, $C = 5$, and $D = 64$, when using SPM and Fisher vectors for apperance, the dimensions of matrix $\boldsymbol{F}$ would be 129.000×129.000. It requires 66 Gb of memory to store such matrix, while the time required for inversion of such big matrix makes diagonal approximation a necessity.

4.3.6 Discussion and comparison to SPM

We summarize the models we have presented in this section in Figure 4.2 and compare the proposed models to generative model of spatial pyramids (SPM). The Fisher vector representation for an image is given by the *gradients* of generative model parameters *w.r.t.* to the log-likelihood the image data, at the parameters that maximize the log-likelihood of the training data. Therefore it also takes into account a large training data set via the learned parameters of the generative model. To see how this influences the representation consider Equation 4.2. While BoW histograms h_k have big components for visual words k which are more frequent in the image, the Fisher vector image representation for the same generative model has big components for the visual words which are more frequent in the image than on averge. An empty bin for a visual word in BoW histogram means that no region descriptors are assigned to the visual word, while in case of Fisher vectors zero value for a component corresponding to a visual word k means that the visual word is equally frequent in image as it is on average. Therefore the Fisher vector representation of the same size as BoW histogram is richer because it takes into account more information. When using SFV for coding spatial layout and FV for coding the appearance each patch will contribute to $2D$ dimensions for appearance and $2d$ for its position, while in BoW-SPM each patch only contributes to one spatial cell per level of a quad-tree. Therefore, the BoW-SPM codes are sparser, and less efficiently use the representation dimensions.

In Table 4.1 we give the representation size for each of them, and compare them to the sizes obtained using spatial pyramids (SPM) [Lazebnik *et al.*, 2006, Perronnin

spatial	appearance k-means	appearance MoG
None	K	$K(1+2D)$
Gauss.	$K+K2d$	$K(1+2D)+K2d$
MoG	$K+KC(1+2d)$	$K(1+2D)+KC(1+2d)$
SPM	KC	$KC(1+2D)$

***Table 4.1**: Comparison of representation size for the different models, using either k-means or a MoG for appearance, and either no spatial model, a single Gaussian, or a MoG.*

et al., 2010b] that concatenate appearance representations obtained over C spatial cells. We use C to denote either the number of components in our spatial MoG, or the total number of cells in the SPM representation.

Comparing SPM to our MoG spatial model in combination with k-means for appearance, we see that our representation adds $2d = 4$ numbers for each visual word (K) and spatial cell C. The size of the MoG model with $C = 1$ equals the SPM model with $C = 5$.

Comparing SPM to our MoG spatial model with MoG appearance models, we see that our model yields a much more compact representation. Where SPM concatenates C appearance Fisher vectors of size $K(1+2D)$, our representation uses a single appearance Fisher vector of size $K(1+2D)$ and adds KC spatial Fisher vectors of size $(1+2d) = 5$. For a typical setting of $K = 200, D = 64, C = 5$, the SPM representation is 129.000, while our MoG spatial-appearance model yields a descriptor of size 30.800: more than 4 times smaller. When using $C = 21$ the sizes would be 541.800 and 46.800 respectively, and our descriptor is more than 11 times smaller.

To compute our representation we have to compute the appearance soft-assign, and the spatial soft-assign per visual word. So, the only additional cost is to compute a spatial soft-assign per visual word, which costs $O(KCd)$. Since the appearance soft assign has cost $O(KD)$ per patch (regardless of k-means or MoG model), and since the descriptor dimension is typically much larger then the number of spatial cells, *i.e.* $D \gg C$, we can state that in general the computational cost is less than doubled. For example, when $D = 64$, $C = 1$ and $d = 2$ the spatial assign is 32 times faster than the appearance assign.

4.4 Experimental evaluation

Feature extraction and vocabulary learning. In all experiments we follow the same feature extraction process. We sample image patches on a regular spatial grid,

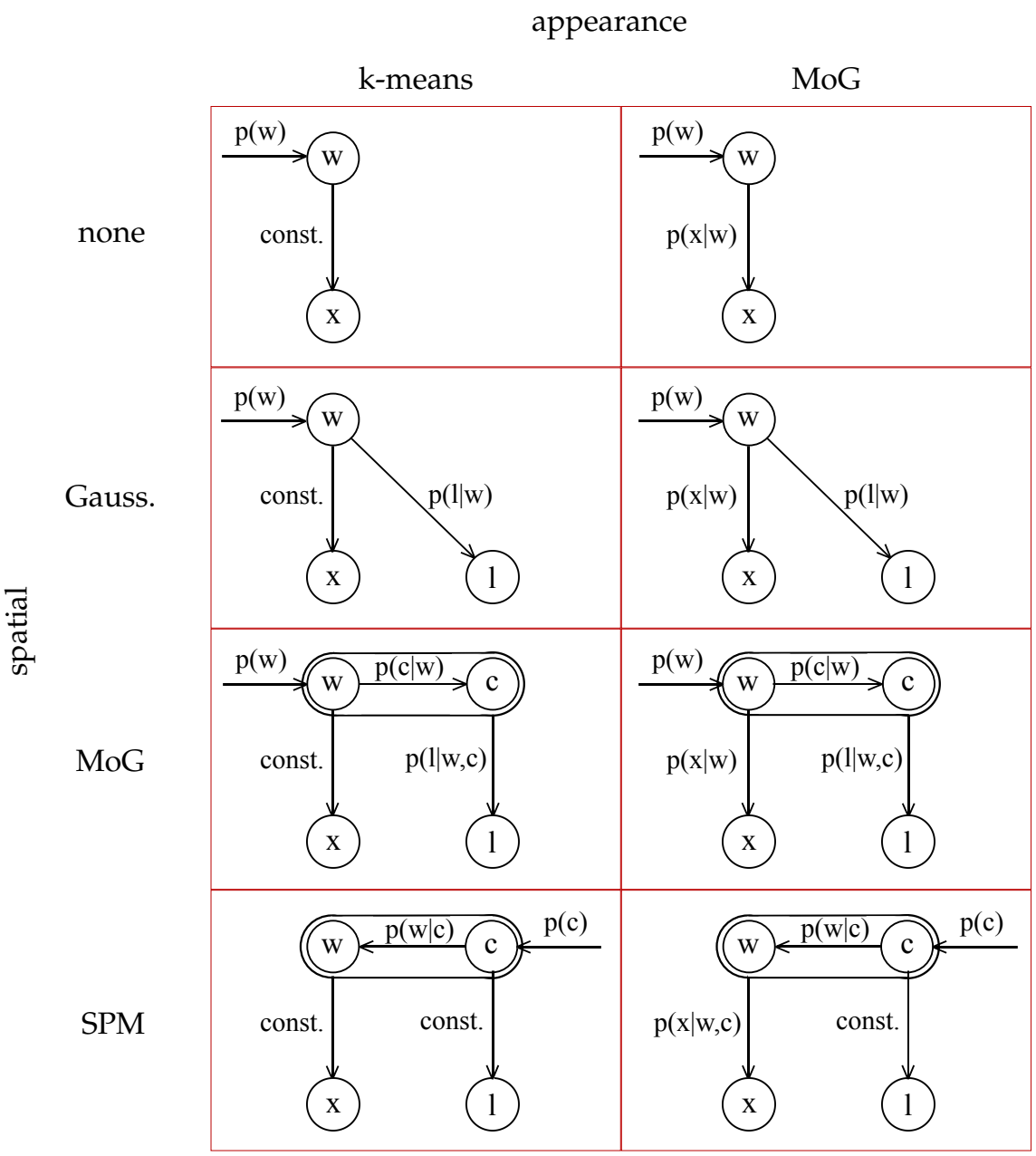

***Figure 4.2**: Different models of appearance and position, either using k-means or a MoG for appearance, for various alternatives of modeling spatial layout. From the last two rows we see that in our case (third row) the position $\mathbf{l}$ is generated by combination of spatial cell c and visual word k, while in case of SPM (last row) the appearance descriptor $\boldsymbol{x}$ is generated by the combination of k and c. Since $D \gg d$ our representation is much smaller. Additionally, our representation performs soft-assign of the region descriptors to the components of the learned spatial layout model, while SPM performs hard-assign to a fixed quad-tree.*

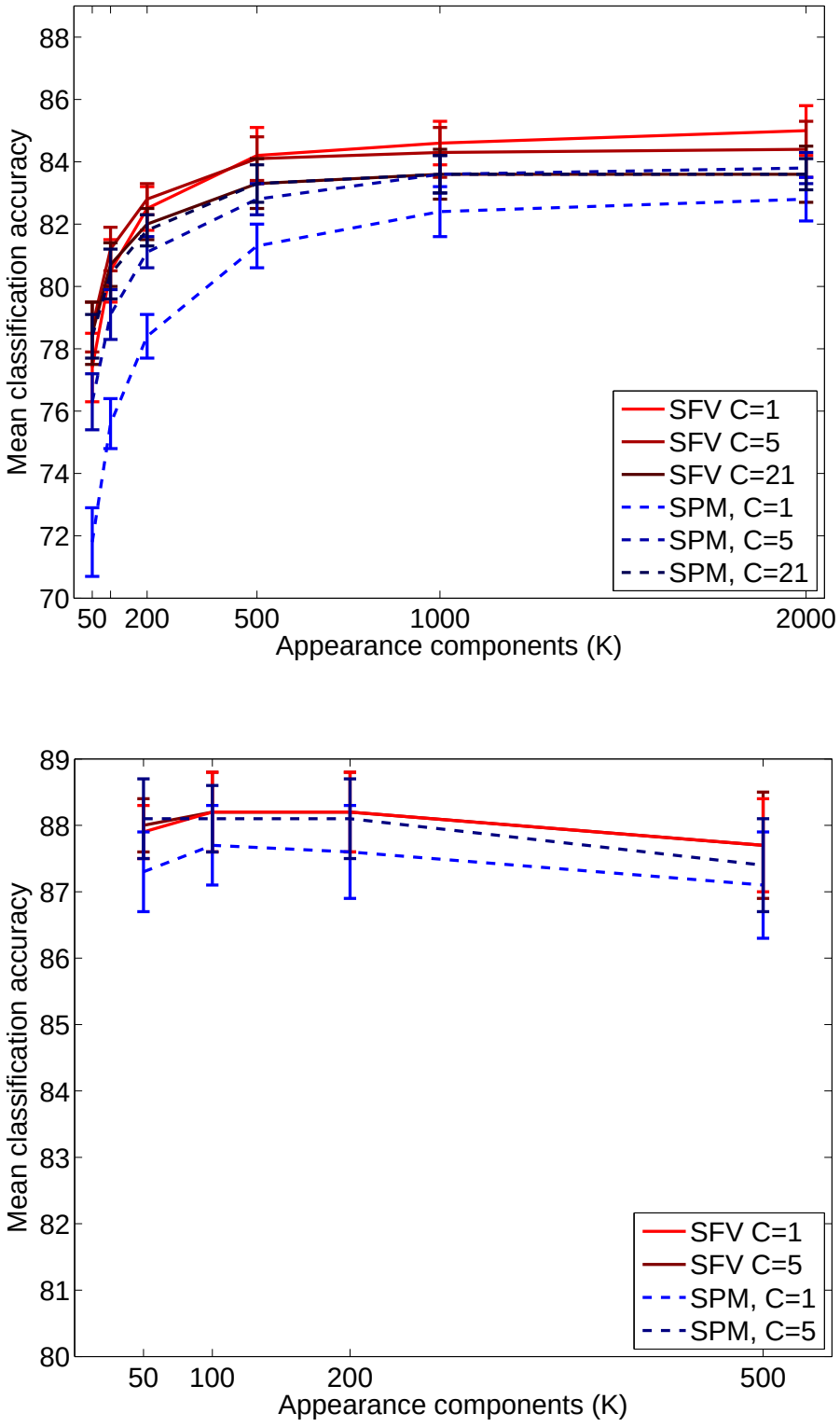

***Figure 4.3**: Using 15-Scenes data set to compare Spatial Fisher Vectors (SFV, solid curves) to Spatial Pyramids (SPM, dashed curves) for coding spatial layout, when using bag-of-words for coding appearance (top), and when using Fisher vector for coding appearance (bottom).*

with step-size half of the patch-size, over 8 scales separated by a factor 1.2. At the finest scale we use a patch-size of 20 and 16 pixels for the 15-Scenes and PASCAL VOC 2007 data sets, respectively. We compute 128 dimensional SIFT descriptors, which we project to 64 dimensions using PCA. This is done to reduce the image representation when using Fisher vectors to code appearance. Because of the statistics of natural images and way the SIFT descriptor performs spatial binning, we expect that the intrinsic dimensionality of the SIFT descriptors is lower than the original one, because the components of SIFT descriptors are highly correlated. By using PCA we decorrelate SIFT dimensions globally, therefore better fitting our modelling assumption that the components are uncorrelated locally, which is assumed by diagonal form of covaraince matrices for appearance model components. In [Perronnin & Dance, 2007, Zhou *et al.*, 2009] PCA is also used to compress SIFT features from 128 to 64 dimensions. The k-means and MoG appearance models, as well as PCA subspace, are learned using a random sample of 500.000 patches from the training images.

Construction of spatial models. Once the appearance models are learned we can learn the spatial models, either Gaussian or MoG, using the patches assigned to each visual word. However, in initial experiments we found that without loss of performance we can also use a fixed spatial model shared across all visual words ($m_c = m_{kc}, S_c = S_{kc}$).

Using $C = 1$ spatial Gaussian we set the mean and variances to match the first and second order moment of the uniform distribution over the unit square. Using $C = 5$ components we complement the global Gaussian with four Gaussians, each matching the first and second order moments of the four quadrants of the unit square. Similarly we add 16 Gaussians matching the uniform distribution of the 16 regular cells of the unit square. The mixing weights are set so that they sum to the same value per level. Note that the spatial model resembles the structure of the SPM in this case. The main differences are that we store spatial first and second order moments of the patches assigned to each spatial component, and that we use a spatial soft-assign.

Compared representations. In our experiments we compare the representations summarized in Table 4.1. We test SPM representations up to three levels; at the first level we have only $C = 1$ spatial cell which does not encode any spatial information. Using the first two levels we have $C = 5$ spatial cells, and using all three levels we have $C = 21$ spatial cells. When using Fisher vectors for appearance, we do not include $C = 21$ since then the image representation becomes very large, without increasing performance. For our Spartial Fisher Vector (SFV) representations we use $C = \{1, 5\}$ Gaussian components; using more components did not improve performance.

Classifier training and evaluation. For all image representations we learn a linear classifier over the Fisher vector representations, and include the L2 and power nor-

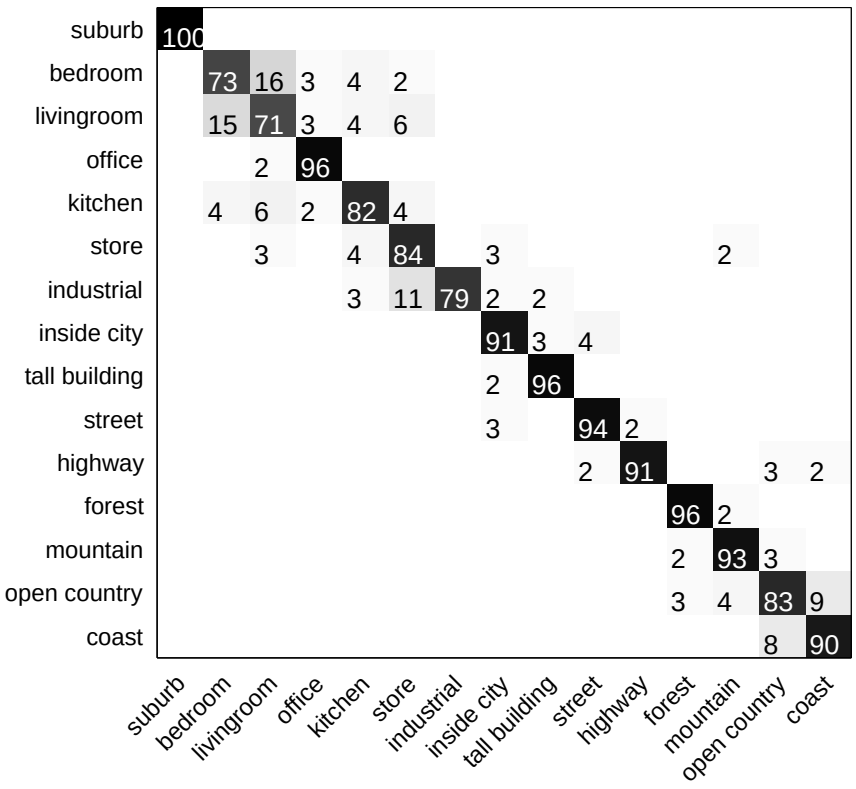

Figure 4.4*: Normalized confusion matrix for 15-Scenes dataset (the rows are the true classes), we only show figures larger than one.*

malizations of [Perronnin *et al.*, 2010b]. For a fair comparison, we use the histogram intersection kernel [Lazebnik *et al.*, 2006] when using BoW+SPM representations, since these seem to be optimal for that representation. We follow standard evaluation measures for used datasets. For the 15-Scenes data set we learn multi-class logistic discriminant models, and report classification accuracy measured as the fraction of correctly clasified test images. For PASCAL VOC 2007 we learn a binary SVM classifier per class, and report the mean of the per-class average precision (mAP) values.

Experimental results for the 15-Scenes dataset. The 15-Scenes data set [Lazebnik *et al.*, 2006] contains 4485 images of the categories *bedroom, suburb, industrial, kitchen, living room, coast, forest, highway, inside city, mountain, open country, street, tall building, office*, and *store*. We use the standard setup for this data set, using 10 random splits of the data into a train set of 100 images per class, and using the rest as test data. We then average the classification accuracy over the test/train splits.

In Figure 4.3 we show the classification accuracies as a function of the vocabulary size K. Using k-means to encode appearance (left panel) we see that large vocabularies ($K \geq 1000$) yield the best performance, and that our Spatial Fisher Vector

***Figure 4.5**: Mis-classified images with highest scores for six classes in 15-Scenes dataset. Each row corresponds to the predicted class, while the true class is annotated below each image.*

		SPM			SFV	
		1	5	21	1	5
BOW	50	29.1	37.0	41.4	35.1	37.9
	100	33.8	40.4	44.1	39.8	41.7
	200	38.1	44.1	47.1	43.5	45.1
	500	42.7	47.7	49.9	47.5	48.9
	1000	45.9	50.1	51.5	50.1	50.8
	2000	48.0	51.1	52.3	52.3	52.9
Fisher vector	50	54.1	55.8		55.4	50.2
	100	55.0	56.5		56.1	55.6
	200	55.5	56.7		56.5	56.1
	500	55.5	56.5		56.6	56.3

***Table 4.2**: PASCAL VOC 2007: comparison of spatial pyramids (SPM) with with $C = \{1, 5, 21\}$ cells (left) to Spatial Fisher vectors (SFV) with $C = \{1, 5\}$ spatial cells (right) for coding spatial layout. Using bag-of-words (BOW) for coding appearance (top), and using Fisher vector for coding appearance (bottom).*

	50	100	200	500
Early fusion	54.9	55.4	55.6	55.6
SFV	55.4	56.1	56.5	56.6

***Table 4.3**: PASCAL VOC 2007: comparison of Fisher vectors obtained from MoG learned from concatenation of patch appearance and position descriptors (top row) to Spatial Fisher vectors (SFV) with $C = 1$ spatial cells, using Fisher vector for coding appearance (bottom row), when varying number of apperance components K. The representations have the same sizes.*

representation with $C = 1$ outperforms all others, achieving 85.0 ± 0.8 accuracy. The size of our representation is in this case $K + K2d = 10.000$, which is the same as the size of the best SPM model with $C = 5$ which uses a non-linear kernel an achieves 83.8 ± 0.5. Our results are remarkably good for a bag-of-word image appearance models in combination with linear classifiers.

When using Fisher vectors for appearance (right panel) performance is generally much higher (note difference in axis scaling). In this case our Spatial Fisher Vector representation with $C = 1$ and $K = 100$ achieves best performance at $88.2\% \pm 0.6$, which is comparable to using SPM with $C = 5$ cells ($88.1\% \pm 0.5$). Note that our representation is much smaller, $K(1 + 2D + 2d) = 13.300$ dimensions, than using SPM: $KC(1 + 2D) = 64.500$ dimensions. We also noticed that performance saturates or drops when using vocabularies larger than 100 to 200 visual words. This consistent with the observations made by [Perronnin & Dance, 2007].

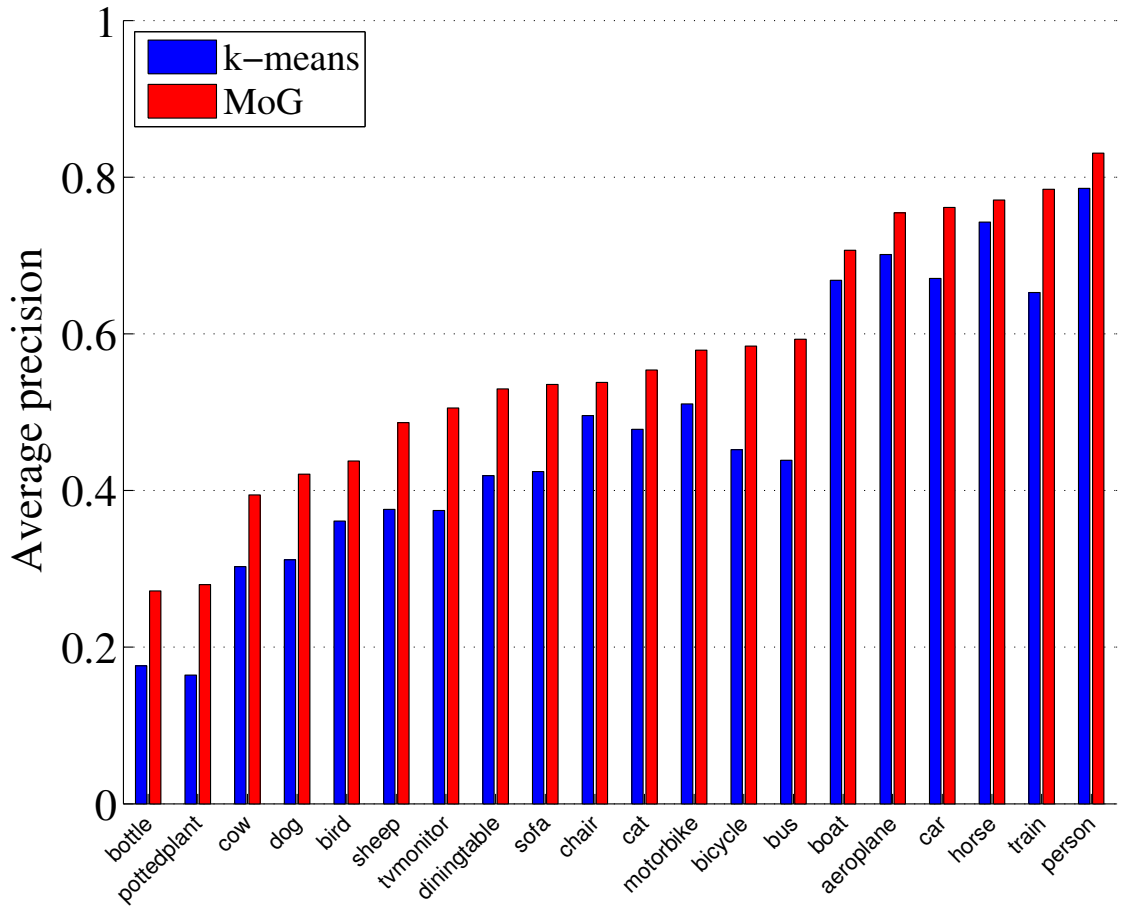

***Figure 4.6**: Per class comparison of SFV using $K = 500$ visual words and $L = 1$ spatial cells.*

Our results with only $K = 200$ visual words are on par with the current state-of-the-art of 88.1% reported in [Xiao *et al.*, 2010]. While we only use SIFT descriptors, [Xiao *et al.*, 2010] combines 14 different low-level image features; when using only SIFT [Xiao *et al.*, 2010] reports 81.2% using a BoW+SPM representation and intersection kernels.

In Figure 4.4 we show the confusion matrix we obtain with our best model. We see that the similar scenes are confused: the majority of classification errors are due to confusion of indoor scenes, followed by the group of classes that depicts the urban architecture (classes *insidecity*,*tallbuilding*,*street*,*highway*) and finally the outdoor scenes displaying countryside.

In Figure 4.5 we show the confused images with the highest scores, for six classes with prediction accuracy less than 90%. For the majority of confused images the true class the image displays spatial layout of image parts similar to the one of the predicted

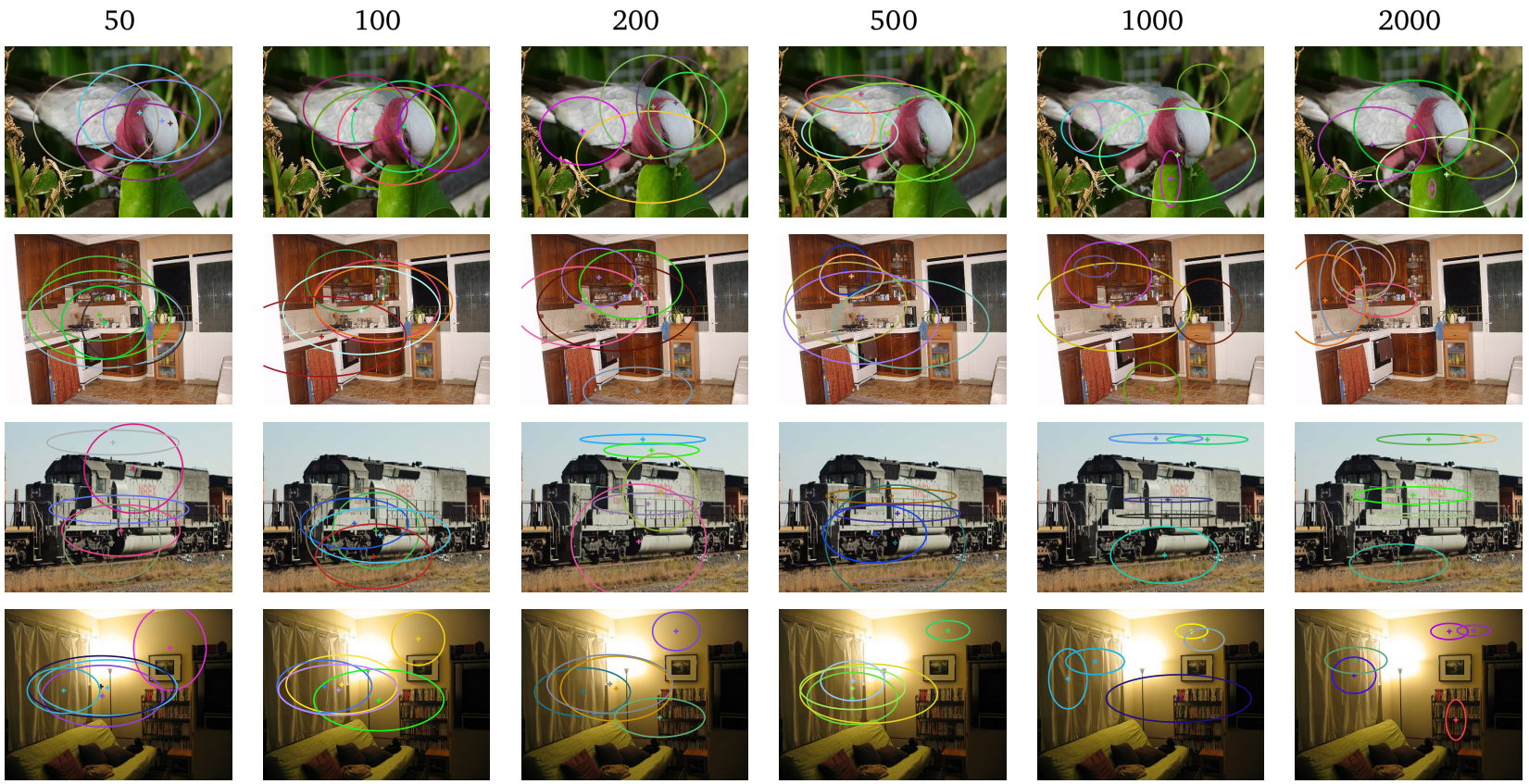

***Figure 4.7**: Ellipsoids display the spatial distribution of patches assigned to a visual word. Five visual words with the biggest difference in frequency of occurrence in image with respect to its average frequency are displayed (biggest $\frac{\partial \mathscr{L}}{\partial \alpha_k}$). Columns correspond to different vocabulary sizes. As the vocabulary size grows, the uncertainity about position of patches assigned to a visual word diminishes. In the extreme case, when a single feature is assigned to visual word, there is no uncertainity about position of the patch, while many SPM cells would be needed to code it with such high precision.*

class. Images confused between classes *bedroom* and *livingromm* have very similar spatial layout, with the only difference that images of bedrooms display beds, while images of living rooms display sofas.

Experimental results for PASCAL VOC 2007. The PASCAL VOC 2007 data set contains 9963 images, annotated for presence of 20 different object categories. We have used the 5011 images in the train and validation sets to train our models, and evaluate them on the 4952 test images.

In Table 4.2 we show the mAP scores for different vocabulary sizes. When using bag-of-word appearance models (top), we observe that our Spatial Fisher vector representations with $C = 1$ and a linear classifier yield performance comparable to using $C = 5$ cells with SPM and intersection kernel. The best performance of 52.9% is obtained using spatial Fisher vectors with $C = 5$ components, and 2000 visual words. The best SPM results of 52.3% are obtained using $C = 21$ cells, and $K = 2000$. As for the 15-Scenes data set, using Fisher vectors for appearance (bottom) improves the results, to a maximum of 56.6% using SFV with a single Gaussian, and for SPM the best results are 56.7% using $C = 5$ cells. Again, our representation is much smaller, using $K = 200, C = 1$ the SFV has size $K(1+2D+2d) = 26.600$, while using SPM with $K = 200, C = 5$ yields a $KC(1 + 2D) = 129.000$ dimensional image representation.

Our results are comparable to those in [Perronnin *et al.*, 2010b], which reports 55.3% using SPM with $C = 1, K = 256$, our results with SPM and $C = 1, K = 200$ are 55.5%. They reported 58.3% using SPM with $C = 8$ cells, which uses the complete image, the four quadrants, and using 3 horizontal strips, which we did not explore here.

In Table 4.3 we concatenate the appearance vectors $\boldsymbol{x}$ and the location vector $\boldsymbol{l}$ into a single vector of dimension $D + d$, and compute Fisher vectors for an MoG model learned on the concatenated vectors. Using K mixture components this yields a descriptor of size $K(1 + 2(D + d))$, which is the same as using SFV with $C = 1$. For all vocabulary sizes SFV outperforms early fusion of appearance and location vectors. With an early fusion of appearance and position information the patches of the same appearance that occur at different locations in the image can be assigned to different components of the joint appearance-position generative model. Therefore, given the fixed number of components K in order to model the position of the patches the precision with which its appearance is coded has to be sacrificed, while SFV with the same number of components and the same dimensionality does not suffer from this trade-off, because the appearance and the position are generated by separate models.

In Figure 4.6 we compare per class performance of SFV when using k-means and MoG to code appearance. Both models are using the same number of components to code the appearance ($K = 500$) and layout ($C = 1$). The complexity of representation creation is equal, but the dimensionality of k-means histograms with SFV coding of spatial layout is much smaller since $D \gg d$. For each class coding appearance using

Fisher vectors outperforms BoW histograms, the difference being bigger for classes with low classification performance. The difference in performance is also particularly big for classes that display complex structures, like *bus*, *bicycle* and *train*, probably because the finer structures become indistinguishable due to quantization.

Discussion. We observe that the performance of our representation increases with vocabulary size. Since less patches are assigned per visual word as the number of visual words grows, our spatial Fisher vectors —even with a single spatial component— are able to accurately describe the positions of patches assigned to each visual word. This effect is illustrated in Figure 4.7.

To represent spatial layout with the same accuraccy, SPM has to use many spatial quantization cells. However, this results in very big image representations, that are more likely lead to overfitting.

This effect could also explain results in Table 4.2 and Figure 4.3. When using small number of visual words the gain by adding more spatial components is significant, but this gain diminishes as we increase the number of visual words.

4.5 Discussion and conclusion

We have introduced Spatial Fisher Vectors as a new method to encode spatial information for image categorization. In SFV, spatial "cells" are adapted to the patch positions, unlike the rigid structure of spatial pyramid cells. Our representation has two clear benefits. When combined with bag-of-words appearance models, our representation used with *linear* classifiers gives similar or better results than SPMs with nonlinear intersection kernel classifiers, for comparable size of the representation. When we combine our model with Fisher vector coding of appearance, we also obtain similar or better results compared to SPM. In this case the advantage of SFV is that the image descriptors are roughly four times more compact, reducing requirements on disk storage, memory, and classifier training time by a factor four. In future work we want to further explore the Fisher kernel framework using more advance generative models to encode the appearance and spatial content of images.

5

Summary and Conclusions

In this thesis we have addressed the problem of representing images in a way that allows learning of models that capture semantic meaning of their visual content. Experimental evidence presented in this thesis, as well as in the literature shows that for a given task the choice of image representation significantly influences the performance. Therefore, the question of image representation is essential to computer vision. We have proposed image representations that are adapted to the problem being addressed. Specifically, we have dealt with image representations for image re-ranking and classification.

We next review each of our contributions in the light of goals stated in introduction, summarize the advances presented in previous chapters and comment on some insights which show prospective directions for future work.

5.1 Query-relative features for image re-ranking

Summary. Motivated by query-relative representation of text accompanying images in web-pages, used in image search engines to rank the images according to the relevance to query terms, we have proposed in Chapter 2 a way how to construct query-relative representations of visual content of an image. This representation does not depend only on the visual content of the image but depends also on the query used to retrieve the image, via the distribution of visual representations of a set of images retrieved by the textual query. We used this representation for re-ranking of images retrieved by an image search engine given the textual query. We showed that taking into account visual content of the images significantly improves performance compared to text-only re-ranking. However, the main advantage of this representation is that it allows learning of a single relevance model using data annotated with relevance labels. This approach outperforms the query-specific models [Schroff *et al.*, 2007] which are learned from noisy training examples. Given the same bag-of-words image representation the construction of the query-relative representation using binning and ranking

of images using the learned model is faster than learning and applying query-specific classifier. Since the query-relative features for different modalities have the same meaning the combination of different modalities is achieved by simple concatenation of query-relative representations from different modalities. We have shown that the combination of multiple modalities significantly improves the results.

Future work. The results we obtained using our query-relative features are very encouraging. We would like to relate the insights about query-relative representation to ideas about zero-shot learning [Palatucci *et al.*, 2009] that would allow better understanding and extension of the method.

Since users usually consider only top-ranked images visual diversity of relevant retrieved images is desired. Currently, with our method it is possible that top-ranked images are similar. There are several options to promote the diversity of image ranking. One option is to cluster the retrieved images. Then we can rank the clusters instead of the images, where each cluster is represented by an average representation of images in the cluster, *e.g.* an average BoW histogram. Another option is to rank the images inside the cluster and interleave the rankings from different clusters, similar to [Douze *et al.*, 2009] where rankings from visual and textual cues are simply interleaved with a goal of promoting diversity. Finally, we could simply collapse near duplicates that are close to each other in the ranking.

5.2 Learning tree-structured quantizers for image classification

Summary. The majority of methods for image categorization use representations that are produced in an unsupervised manner, without taking into account class labels of training images. For example, BoW histograms use quantization cells obtained by k-means clustering of local feature descriptors. To improve the discriminative power of image representations, the class labels of training images can be used to construct class-dependent image representation. One group of methods constructs the quantizers to predict the class labels of local feature descriptors, *e.g.* [Moosmann *et al.*, 2008] which learns the tree-structured quantizer from local feature descriptors with the labels inherited from the images they are sampled from. Since the goal is image classification, these methods optimize the quantizer for a related, but different task. Recently a new group of methods appeared which constructs quantizers in a way that directly targets improving image classification performance. In Chapter 3 we have introduced one such method. The main advantage of our method is fast image representation creation using tree-structured quantizers. Using a forest of such quantizers we have obtained very good results using very compact image representation and

simple linear classifiers. Our approach outperforms image representation derived in completely unsupervised manner, as well as image representation optimized for local feature classification, using the same dimensionality of image representation and the same types of classifiers.

Future work. Although our split sampling strategy is very effective it is unlikely that the optimal quantizer parameters could be found by simple sampling, because the space of quantizer parameters is very big. One option is to use more advanced sampling methods or global optimization methods like genetic algorithms or simulated annealing. However, these alternatives are quite slow. Another option is to change the loss function so it becomes differentiable. In that case the sampling of quantizer parameters could be replaced with update of parameters using the gradients of quantizer parameters *w.r.t.* image classification loss. To obtain a function which is differentiable *w.r.t.* to the used loss we could employ a soft branching (*e.g.* using a sigmoidal function) instead of hard assignment of data point in parent node to one of the children.

By growing trees independently some features could be correlated. We would like to explore the possibility of growing the trees in such manner that the learned image features, that correspond to the leaves of the tree, are independent. This would yield more succinct representations. One way to achieve this is to allow inner nodes of the tree to be splitted several times. Every split of the internal node creates a new tree which differs from the already created ones only in the sub-tree whose root node is the splitted inner node. Also, by splitting the inner nodes multiple times we might avoid that tree learning procedure gets stuck in local minima.

Our current model supports only binary classification, so a quantizer is learned per task. We would like to extend the method to handle multi-class and multi-label problems. The extension is relatively straightforward, but would allow learning a quantizers that are shared across several tasks.

5.3 Modeling spatial layout with Fisher vectors for image categorization

Summary. In Chapter 4 we have shown how to interpret existing state-of-art image representation as Fisher vectors *w.r.t.* generative models and proposed a more compact way of representing both the appearance and spatial layout of image patches which represent the image. Compared to use of spatial pyramid (SPM) of [Lazebnik *et al.*, 2006] the advantage of our method is a compact and data adaptive representation. Instead of using a fixed set of regular spatial cells, we capture the spatial

distribution of occurrences of a visual word in the image by its first and second order moments.

Future work. Currently we first learn a PCA subspace of local features and then learn the MoG model for appearance model in this subspace, to maximize the likelihood of projected local features. Instead of learning the PCA subspace and generative model separately, we could couple these steps and learn jointly the PCA subspace and the parameters of MoG model to maximize the likelihood of the original local features.

Here we have used a simple generative model that generates only individual local features that describe the appearance and position of a patch. We could use more complex models that generate sets of local features, which capture the relation between patches' appearances and their local region layouts, by *e.g.* modeling the pairs of neighbouring patches.

Fisher vectors have been also used for image indexing Perronnin *et al.* [2010a]. Spatial Fisher vectors can be also used for that task. Since they directly take into account spatial layout, they can be helpful in reducing the burden of computationally expensive geometrical verification steps in image retrieval. Since our SFV is more compact than SPM, it lends itself better to large scale applications where the limitations on the memory space used per image are more severe.

The extension of Spatial Fisher vectors to representation of videos is straightforward. In this case the generative model captures the spatio-temporal layout of video shots. The dimensionality of video representation vector that capture spatio-temporal relations used currently in video categorization is quite big, since the majority of the methods use a fixed spatio-temporal grid, similar to quad-tree used by SPM. Since Spatial Fisher vectors use adaptive grid, this information could be coded in more compact way using Spatial Fisher vectors.

5.4 Conclusion

As stated in the introduction, since the digital cameras became ubiquitous the number of digital images available increases rapidly. Therefore the time required to process them in order to bring the decisions related to their content becomes increasingly important. To address this issue we have focused our efforts on image representations that are fast to construct and that are compact. Additionally, all our models are linear classifiers which are fast both in learning and application phase, and have significantly smaller memory requirements than the non-linear classifiers usually employed for image categorization.

As the interest of computer vision community starts to shift towards large-scale image classification, the use of image representations that are fast to construct and models that are fast to learn and apply will be indispensable.

We hope that contributions that are presented in this thesis are a step towards this goal.

A

Learning the tree to classify local descriptors

In this appendix we show that chosing the split to maximize mutual information beteen the child nodes and data labels is equvivalent to learning the parameters of the tree-structured quantized to maximize the (log-)likelihood of data labels. Therefore chosing the split that maximizes the mutual information is equvivalent to choice of the split that maximizes the (log-)likelihood of the data class labels.

We denote by $p_{kc} = p(c|k)$ multinomial for class labels c correspodiing to leaf k.

Given the tree structure we want to find the parameters of leaf multinomials p_{kc} that maximize the log-likelihood of labels $\mathscr{Y} = \{y_i\}_{i=1}^N$ given the data $\mathscr{X} = \{\boldsymbol{x}_i\}_{i=1}^N$:

$$\mathscr{L}(\mathscr{Y}|\mathscr{X}) = \log \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i) = \sum_{i=1}^{N} \ln p(y_i|\boldsymbol{x}_i) \tag{A.1}$$

Next, we assign data points $\boldsymbol{x}_i$ to leaf nodes ℓ_k according to the tree structure. We can then group the points assigned to same leaf ℓ_k, and express $\mathscr{L}$ as sum of log-likelihood over K leaves:

$$\mathscr{L} = \sum_{k=1}^{K} \sum_{i \in \ell_k} \ln p(y_i|\ell_k) \tag{A.2}$$

Then, for each leaf and class we rewrite previous expression as:

$$\mathscr{L} = \sum_{k=1}^{K} \sum_{c=1}^{C} n_{kc} \ln p_{kc} \tag{A.3}$$

$$= \sum_{k=1}^{L} n_k \sum_{c=1}^{C} \frac{n_{kc}}{n_k} \ln p_{kc}, \tag{A.4}$$

where n_{kc} denotes the number of data points of class c assigned to leaf k, and n_k denotes the number of patches assigned to leaf k.

We have now expressed the log-likelihood of class labels $\mathcal{Y}$ given data $\mathcal{X}$ and tree structure, via parameters of leaf multinomials p_{kc}. We want to find the parameters of leaf multinomials that maximize this log-likelihood, under constraint that multinomial in each leaf node is a probability distribution. Therefore we have to minimize the follwing Lagrangian:

$$L = \mathcal{L} - \sum_{k=1}^{L} \lambda_k \left(\sum_{c=1}^{C} p_{ck} - 1 \right) \tag{A.5}$$

Setting derivatives of Lagrangian *w.r.t.* multinomial parameters to zero:

$$\frac{\partial L}{\partial p_{ck}} = \frac{n_{ck}}{n_k} \frac{1}{p_{ck}} - \lambda_k = 0 \tag{A.6}$$

$$\frac{n_{ck}}{n_k} = p_{ck} \lambda_k \tag{A.7}$$

The above holds for all c, we can obtain λ_k by summing right and left side of Equation A.7 over c:

$$\lambda_k \sum_{c=1}^{C} p_{ck} = \frac{1}{n_k} \sum_{c=1}^{C} n_{ck} \tag{A.8}$$

$$\lambda_k = 1 \tag{A.9}$$

So we find that the parameters of the optimal multinomial are given by $p_{ck} = \frac{n_{ck}}{n_k}$. Therefore the log-likelihood with the optimal multinomial is:

$$\mathcal{L} = \sum_{k=1}^{L} n_k \sum_{c=1}^{C} p_{ck} \ln p_{ck} = -\sum_{k=1}^{L} n_k H(p_k), \tag{A.10}$$

where by $H(p_k)$ denotes the entropy of the multinomial p_k.

Every tentative node split is evaluated by its contribution to increase of log-likelihood:

$$\mathcal{L}_{\text{new}} - \mathcal{L}_{\text{old}} = n_{\text{parent}} H\left(p_{\text{parent}}\right) - \sum_{k=1}^{2} n_{\text{child}k} H\left(p_{\text{child}k}\right). \tag{A.11}$$

From Equation A.11 we see that this split selection creterion based on maximum likelihood of data labels is the same as the one that whose goal is maximization of mutual information between data labels and children nodes.

B

Rapport de thèse

B.1 Motivation

Au cours des dix dernières années, les appareils photo numériques sont devenus omniprésents, la bande passante du réseau a augmenté, et les techniques de compression d'image ont été améliorées. Ces avancées ont donné lieu à une explosion du nombre d'images numériques intégrées dans des pages web et de blogs. L'avènement des sites dédiés au partage de photos, comme Flickr et Picasa, a permis à encore plus d'images d'être publiées en ligne. Le nombre d'images disponibles ne cesse de croître avec l'expansion des réseaux sociaux qui permettent le partage des images: selon le "Time magazine", plus de 130.000 photos sont téléchargées chaque minute sur Facebook. Afin de permettre l'utilisation de cette vaste et toujours croissante collection d'images, par exemple pour rechercher des images contenant des objets ou des personnes, ou de les organiser en thèmes, les images doivent être indexées par des termes liés à leur contenu. Mais l'annotation des images est une tâche fastidieuse, et bien qu'il y ait un certain nombre de façons pour fournir une description textuelle de l'image, la majorité des utilisateurs n'annotent pas encore les images avec des termes liés à leur contenu.

L'indexation sémantique d'une collection d'images personnelle peut être effectuée manuellement, mais l'indexation de grandes collections d'images à l'échelle du Web est un grand défi. Une solution est de diviser le problème en petites tâches qui peuvent être résolues manuellement. Par exemple, le jeu ESP présente le problème de l'indexation sémantique des images comme un jeu en ligne, tandis que les groupes Flickr permettent d'associer des tags avec les images et d'organiser les images en groupes. Mais même dans l'hypothèse où les annotateurs font un travail parfait, dans le sens où ils sont tous d'accord sur les annotations pour une image, l'effort manuel ne suffit pas pour indexer toutes les images disponibles en ligne parce que le nombre d'images sans annotations augmente trop rapidement. L'annotation d'un sous-ensemble d'images n'est pas une option parce que nous ne savons pas à l'avance

quelles images seront recherchées. Nous aimerions faire de l'indexation sémantique automatique, mais il n'y a pas de relation simple entre le niveau de l'image représentée dans une machine et sa description par des termes liés à leur contenu. [Smeulders *et al.*, 2000] appelle l'absence de cette relation "le fossé sémantique".

Pour apprendre des modèles statistiques qui concernent le contenu des images et des annotations sémantiques nous avons besoin d'images annotées. Par conséquent, bien que l'indexation manuelle des images ne peut pas fournir la solution à ce problème, l'indexation manuelle peut aider dans la construction de modèles statistiques, en fournissant des annotations pour les images qui sont utilisées dans l'apprentissage de modèle. L'apprentissage de modèle suppose que le contenu de l'image est *représenté* dans une machine. Une bonne représentation d'image doit coder toutes les informations *pertinentes* du contenu visuel de l'image. Les informations de l'image considérées comme pertinentes dépendent de la tâche. Nous illustrons cela en utilisant un exemple dans la Figure B.1. En discriminant entre les images des villes et des plages, la couleur est une caractéristique pertinente, donc utiliser par exemple les histogrammes globaux de couleur comme une représentation de l'image est une bonne idée. Cependant, la discrimination entre des images de vaches et de chevaux en utilisant les même représentation est une tâche difficile, car les informations de couleur ne sont pas pertinentes pour cette tâche: la couleur n'est pas une caractéristique discriminante des vaches et des chevaux, et habituellement ils partagent l'environnement dans lequel nous les rencontrons. Par conséquent l'utilisation des représentations simples pourraient être assez bon pour des tâches faciles, mais avec l'augmentation de la complexité du problème on a besoin de représentations d'image plus sophistiquées.

B.2 Les objectifs

Dans cette thèse nous explorons différentes façons de représenter le contenu des images, avec un accent particulier sur les représentations d'images pour la classification et le reclassement des images. Ensuite, nous définissons ces tâches et décrivons brièvement la façon d'apprendre des modèles pour ces tâches, ainsi que la façon d'exécuter ces tâches étant donné la représentation d'image et le modèle appris.

Classification d'image L'objectif de la classification d'image est d'assigner une image à une ou plusieurs catégories sémantiques en fonction de son contenu. Dans le cas de la classification binaire, l'objectif est d'apprendre le modèle d'un objet qui, compte tenu de l'image, répond à la question: "L'objet est-il présent dans l'image?". Le modèle de classification multi-classe a pour but de répondre à la question: "Quel objet est présent dans l'image?". Le classement d'images avec plusieurs étiquettes, appelé également "annotation des images", répond à

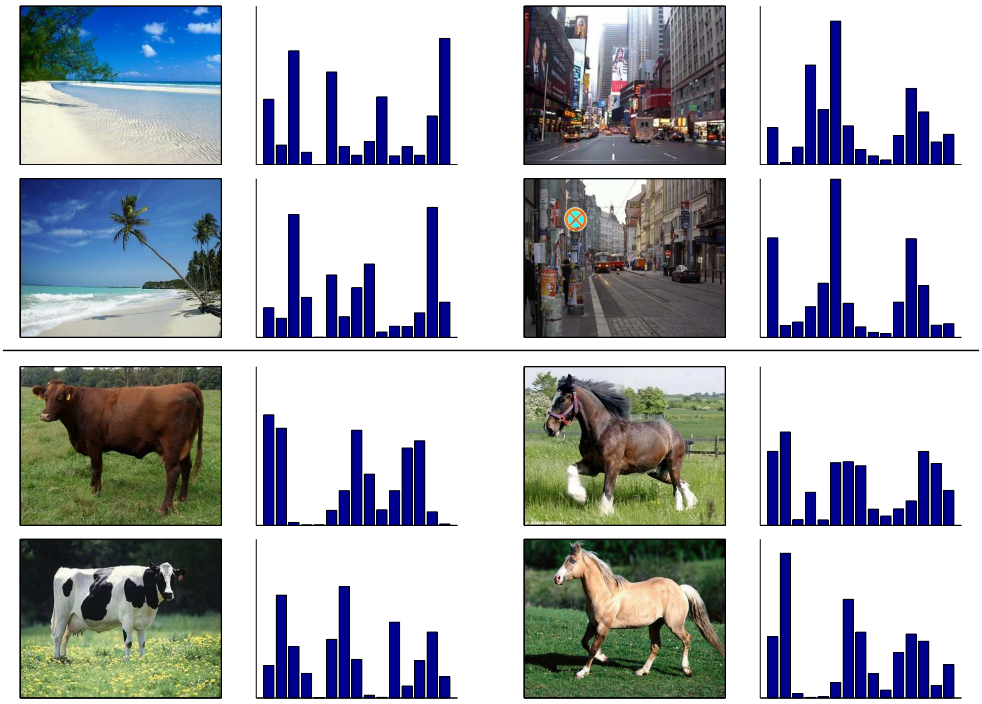

***Figure B.1**: Le représentation de contenu visuel des images avec l'histogramme global de couleur. Cette représentation est suffisante pour le tâche de discrimination entre des images des villes et des plages, mais elle n'est pas assez bonne si on veux séparer les images des chevaux et des vaches.*

la question: "Quels objets sont présents dans l'image?", en apprenant conjointement plusieurs modèles de classification binaire.

Reclassement d'image Compte tenu des images récupérées par une requête de texte, en utilisant par exemple un moteur de recherche d'images, l'objectif du reclassement d'image est d'utiliser le contenu visuel de l'image pour trier les images récupérées, de telle sorte que celles qui sont pertinentes par rapport à la requête sont classées avant celles qui ne le sont pas.

Les modèles statistiques qui relient le contenu visuel d'une image donnée à sa description sémantique peuvent être appris à partir d'un ensemble d'images annotées par une étiquette de formation. Un modèle f est déterminé par des paramètres $\mathbf{w}$. Ainsi, apprendre un modèle revient à déduire les paramètres $\mathbf{w}$ pour lesquelles le modèle prédit bien les étiquettes de formation à partir des caractéristiques des images d'entraînement. Etant donné une image $\mathbf{x}$, le modèle génère un score $f(\mathbf{x}, \mathbf{w})$ qui peut être utilisé soit pour reclasser les images en les triant, soit pour classer les images en comparant leurs scores avec une valeur prédefinie. Un exemple de ces modèles sont les Support Vector Machines [Vapnik, 1998], couramment utilisées dans les méthodes

de l'état de l'art de classification d'images. Certes, d'autres modèles sont possibles (modèles à base de règles par exemple), mais puisque le choix du modèle est orthogonal aux contributions présentées ici, nous n'utilisons que des modèles statistiques reconnus pour être à la fois simples et efficaces.

B.3 Les représentations d'images pour la classification

Une des premières tentatives d'abstraction du contenu visuel de l'image a été d'utiliser des méta-données textuelles, associées aux images (par exemple les méta-données EXIF, le nom de fichier de l'image et le nom du dossier, les tags HTML dans les pages web, les tags fournis par l'utilisateur, etc) pour représenter les images via du texte. La majorité des moteurs de recherche d'images s'appuient sur une telle représentation pour récupérer et classer les images par rapport à une requête textuelle. Cependant, comme les méta-données sont générées souvent sans l'intention d'annoter les images avec des informations qui sont sémantiquement significatives pour le contenu des images, elles peuvent être erronées, ambiguës, incomplètes, ou tout simplement absentes. C'est pourquoi l'hypothèse que le texte des méta-données est lié au contenu de l'image n'est pas toujours valide. Par conséquent, afin d'améliorer la qualité de la recherche et du classement des images, le contenu visuel de l'image doit être pris en compte.

B.3.1 Les représentations globales d'image

Les premières représentations d'images utilisées ont été "globales" . Ces représentations agrègent des attributs locaux de couleur, de forme ou de texture dans des caractéristiques globales d'image [Jain & Vailaya, 1996, Manjunath & Ma, 1996]. Elles sont compactes, rapides à construire et invariantes à la disposition des parties d'image, mais leur pouvoir discriminant est limité. En effet, l'influence de chaque caractéristique de pixel sur la représentation totale de l'image peut être petite. Cette situation est défavorable pour certaines tâches, par exemple pour déduire si des objets comme les voitures ou les vélos sont présents dans l'image, parce que l'influence des caractéristiques de l'objet peut être noyée par le fond d'image. L'effet est encore plus évident lorsque l'objet dans l'image est petit ou occlus. Ces représentations sont un bonne encodage d'un contenu d'image lorsque les caractéristiques individuelles des pixels sont déjà informatives pour la tâche, comme dans le cas de l'utilisation de la couleur pour discriminer entre des images de plages et de villes (*c.f.* Figure B.1).

B.3.2 Les représentations locales d'image

Pour surmonter les limitations des représentations globales, l'image peut être représentée par un ensemble de régions. Ces représentations sont donc appelés "locales". Comme seule une fraction de régions est influencée par les occultations et le fouillis de fond, les représentations d'images locales sont plus robustes à l'égard de ces effets. Ici nous allons nous concentrer sur le groupe de représentations d'images basées sur les "sac-de-mots", d'abord introduit dans [Sivic & Zisserman, 2003, Csurka *et al.*, 2004]. Bien que de nombreuses représentations d'objet peuvent également être utilisées pour représenter l'ensemble de l'image (par exemple par histogrammes de gradients orientés [Dalal & Triggs, 2005], la forme des segments [Opelt *et al.*, 2006, Ferrari *et al.*, 2010], des modèles en constellation, [Weber *et al.*, 2000, Fergus *et al.*, 2003] ou avec une SVM latente [Felzenszwalb *et al.*, 2010]), nous ne traitera que des représentations d'images par sac-de-mots car elles sont très efficaces et ont été l'état de l'art dans la classification d'images. Nous allons ensuite décrire la création de représentations locales d'image basées sur les sac-de-mots.

La représentation locale d'image peut être décrite par une succession d'étapes: la sélection des régions, la description d'apparence de région, le codage d'apparence de région, et la dérivation des caractéristiques d'image de l'ensemble des codes des régions par l'agrégation spatiale. L'aperçu schématique de ces étapes est donné à la Figure B.2. Nous allons maintenant décrire chacune de ces étapes plus en détail.

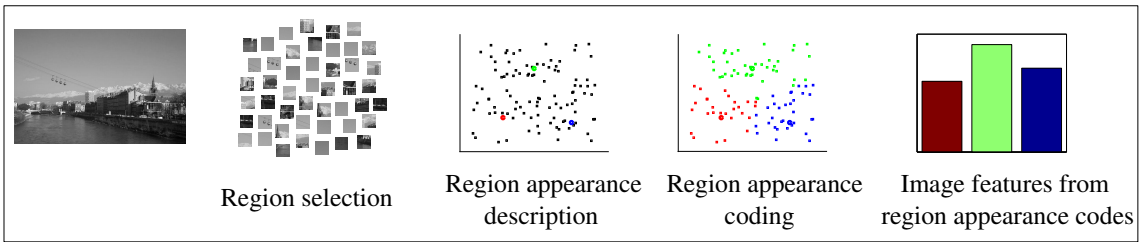

***Figure B.2**: Aperçu schématique des étapes de calculation la représentation locale de l'image: l'exemple du sac-de-mots représentation de l'image.*

La sélection des régions La façon dont on sélectionne des régions de l'image est importante, puisque la sélection de la région influence la représentation de l'image : différents ensembles de régions mènent à différentes représentations de l'image. Une des premières oeuvres qui utilise la représentation locale de l'image utilise un algorithme de segmentation d'image pour sélectionner des régions d'image [Barnard *et al.*, 2003]. Cette option est intéressante car elle permet à chaque région d'être décrite avec sa forme [Belongie *et al.*, 2002], en plus de la couleur et la texture. Cependant, comme les algorithmes de segmentation ne produisent pas des régions

stables, et qu'ils nécessitent des calculs intensifs, cette option n'a pas gagné beaucoup d'attention.

L'idée d'utiliser des détecteurs de points d'intérêt pour sélectionner les régions a été emprunté au domaine de la correspondance stéréoscopique. Les détecteurs de points d'intérêt sont construits pour détecter des structures comme les coins [Harris & Stephens, 1988], les blobs [Lindeberg, 1998] et les arêtes [Steger, 1998]. La région rectangulaire autour du point détecté (aussi appelé "patch") est représentée par un descripteur de l'apparence. Utiliser des détecteurs de points d'intérêt a l'avantage principal de détecter les structures à leur échelle intrinsèque, par le principe de sélection d'échelle automatique [Lindeberg, 1998]. Cela permet la description des régions par des caractéristiques locales, qui sont invariantes aux transformations affines locales [Mikolajczyk & Schmid, 2004]. L'utilisation de détecteurs de points d'intérêt suppose implicitement que certaines régions de l'image ne sont pas importantes pour la représentation de l'image, une hypothèse qui semble intuitivement valide. Toutefois, étant donné qu'elles ne sont pas construites pour la tâche à accomplir, il n'y a aucune garantie que les régions sélectionnées donneront une représentation d'image optimale pour la classification ou le reclassement d'images.

Cet inconvénient des détecteurs de points d'intérêt est la motivation principale derrière l'échantillonnage dense de régions : si l'image est complètement couverte par les régions, l'image peut être reconstruite à partir de l'ensemble des régions sélectionnées, et par conséquent aucune information n'est perdue. La représentation de l'image encode le contenu de l'image complète, et la détermination de l'importance de chaque région est laissée à des étapes ultérieures. Il a été démontré que pour certaines tâches de classification d'image, la sélection de régions à partir d'une grille régulière surpasse l'utilisation de détecteurs de points d'intérêt [Winn *et al.*, 2005]. [Nowak *et al.*, 2006] effectuent l'échantillonnage de régions au hasard à partir d'une distribution spatiale fixée, et dans [Moosmann *et al.*, 2008], cette approche a été étendue par la mise à jour de la distribution spatiale.

La description de l'apparence des régions L'apparition d'une région sélectionnée est décrite par un ensemble d'attributs d'apparence de région, appelé "descripteur local" ou "caractéristique locale". Ces descripteurs peuvent être semblables aux représentations globales d'images, mais ils sont généralement construits manuellement pour avoir certaines invariances géométriques et photométriques, par exemple l'invariance aux changements d'illumination non-uniforme et les distorsions géométriques causées par différentes poses d'objets. Pour atteindre l'invariance de la réponse photométrique, des bancs de filtres décrivant la texture locale ont été utilisés [Leung & Malik, 1999, Ojala *et al.*, 2002]. L'invariance aux distorsions géométriques est par ailleurs adressée à l'aide de binning spatiale comme dans SIFT [Lowe, 2004], brouillant spatiale [Berg & Malik, 2001] ou en projetant la distribution spatiale des

réponses de filtre à un sous-espace, comme dans PCA-SIFT [Ke & Sukthankar, 2004]. La couleur de la région [van de Weijer *et al.*, 2007] et ses caractéristiques basées sur l'intensité des pixels [Obdrzalek & Matas, 2002, Kadir & Brady, 2001] ont également été utilisés pour décrire l'apparence des régions.

Le codage de l'apparence des régions Les descripteurs d'apparence de région sont codés à l'aide d'un ensemble de vecteurs prototypes. Un prototype est appelé "mot visuel", et l'ensemble des mots visuels utilisés pour coder les descripteurs de région est appelé "dictionnaire visuel". Le dictionnaire visuel est utilisé pour exprimer le contenu de l'image, tout comme les mots du texte sont utilisés pour exprimer le contenu du document texte. Toutefois, lorsqu'il s'agit de codage de contenu de l'image, les mots visuels ne sont pas donnés, mais doivent être déterminés. Par conséquent, deux questions importantes doivent être posées : comment obtenir un dictionnaire visuel, et comment coder une caractéristique locale à l'aide d'un dictionnaire visuel donné. Les méthodes peuvent être divisées en deux groupes, basés sur les approches utilisées pour répondre à ces questions. Celles qui apprennent le dictionnaire visuel et le code des caractéristiques locales d'une manière non-supervisée, ayant comme but la reconstruction des descripteurs sont appelées ici "les méthodes de codage guidées par la reconstruction du contenu visuel d'image". On appelle "méthodes de codage guidées par la prédiction de l'étiquette d'image" les méthodes supervisées, qui apprennent le dictionnaire visuel en tenant compte de l'étiquette des images d'entrainement.

Les méthodes guidées par la reconstruction ont pour but de trouver le dictionnaire visuel qui permet de reconstruire les descripteurs avec un minimum d'erreur. Les premières approches ont obtenu le dictionnaire visuel par l'algorithme des k-moyennes et ont codé la région par l'indice du mot visuel le plus proche[Sivic & Zisserman, 2003, Csurka *et al.*, 2004]. [Nistér & Stewénius, 2006] utilisent l'algorithme de k-moyennes hiérarchique afin de réduire la complexité algorithmique de codage de chaque région de $O(n)$ à $O(log(N))$, où N est le nombre de mots dans le dictionnaire visuel. Lors de l'utilisation des k-moyennes pour quantifier l'espace des descripteurs des cellules de quantification s'adaptent à la distribution des descripteurs de formation. Ceci est sous-optimal lorsque cette distribution est non uniforme, et par conséquent des caractéristiques qui sont moins fréquentes ont plus d'erreur de reconstruction. Pour surmonter ce problème, [Jurie & Triggs, 2005] utilisent l'algorithme "mean shift" [Comaniciu *et al.*, 2000] pour obtenir les mots visuels.

Comme montré dans [Boiman *et al.*, 2008], le principal inconvénient de ces approches est que chaque caractéristique locale est codée par un simple coefficient de reconstruction: l'indice du mot visuel le plus proche. Dans ce cas, l'erreur de reconstruction est grande, sauf si un grand nombre de mots visuels sont utilisés. Pour réduire l'erreur de reconstruction, [Philbin *et al.*, 2008] et [van Gemert *et al.*, 2010a] pondèrent l'attribution d'une caractéristique à plusieurs mots visuels, de sorte que l'apparence

de chaque région est codée par de multiples coefficients de reconstruction. La même idée a conduit [Yang *et al.*, 2009, Wang *et al.*, 2010] et [Boureau *et al.*, 2010a] à coder les caractéristiques locales avec des coefficients de combinaison linéaire de mots visuels, pour lesquels l'erreur de reconstruction est minimisée, sous des contraintes de parcimonie [Yang *et al.*, 2009] ou des contraintes de localité [Wang *et al.*, 2010]. Dans l'approche vectorielle Fisher [Perronnin & Dance, 2007], chaque caractéristique locale est codée en utilisant les gradients par rapport aux paramètres du modèle génératif, précisément, le mélange de gaussiennes dont les paramètres sont appris par l'algorithme EM à partir de l'ensemble des caractéristiques locales d'entrainement.

Les méthodes guidées par la reconstruction minimisent l'erreur de prédiction de l'étiquette de classe associée soit à la caractéristique soit à la image. [Moosmann *et al.*, 2008] apprennent les forêts randomisées pour prédire les étiquettes de classe de la caractéristique à partir des descripteurs de la région. L'étiquette de chacun des descripteurs de région de formation est héritée de l'étiquette de l'image de la région à partir de laquelle elle a été échantillonnée. La croissance de l'arbre aléatoire est guidée par la minimisation de l'erreur de classification de région. Les feuilles de l'arbre sont des mots visuels qui correspondent aux cellules de quantification de l'espace du descripteur. Plusieurs arbres sont utilisés, de sorte que chaque élément est codé par des coefficients multiples, où chaque coefficient correspond à l'indice de la feuille dans l'arbre. Dans [Perronnin, 2008], le dictionnaire initial est construit avec un but de reconstruction de caractéristiques, puis les mots visuels obtenus sont adaptés en utilisant les étiquettes de classe. La caractéristique est codée en utilisant à la fois le dictionnaire universel et les dictionnaires adaptés spécifiquement pour chaque classe.

Dérivation des caractéristiques de l'image à partir des codes des régions par agrégation spatiale La majorité des modèles utilisés pour la classification d'images supposent que l'image est représentée par un espace vectoriel muni d'un produit scalaire. Pour enchâsser l'ensemble des caractéristiques locales dans un espace vectoriel, l'ensemble des coefficients qui codent la région sont agrégés dans un vecteur de taille fixe qui représente le contenu visuel de l'image. L'agrégation est effectuée par mot visuel, de sorte que les coefficients correspondant à un mot visuel sont agrégés sur l'ensemble de codes représentant les régions d'image. Le cas où la représentation d'image est calculée comme la somme des coefficients de fonction est appelé "agrégation par somme". L'agrégation par somme est effectuée par sac-de-mots, comme dans la méthode de [Sivic & Zisserman, 2003, Csurka *et al.*, 2004, Nistér & Stewénius, 2006, Jurie & Triggs, 2005] et l'approche des vecteurs de Fisher [Perronnin & Dance, 2007]. Dans "l'agrégation par maximum", les caractéristiques d'image sont dérivées comme le maximum des codes correspondant au mot visuel. Cette agrégation a été introduite par [Riesenhuber & Poggio, 1999], dont l'objectif était de modéliser les réponses des cellules dans le cortex visuel des primates. [Boureau *et al.*, 2010b] ont

montré que ce type d'agrégation sépare mieux les caractéristiques d'image correspondant aux mots visuels qui ont une faible probabilité d'être actifs, et que l'agrégation de toutes les régions est sous-optimale. L'agrégation par maximum est également utilisée par [Yang *et al.*, 2009, Wang *et al.*, 2010], et [Moosmann *et al.*, 2007] où les vecteurs binaires sont utilisés pour décrire l'image.

Les représentations d'image sont enfin normalisées, de sorte que les normes L1 ou L2 de tous les vecteurs de l'image sont égales. Lorsque l'on utilise l'approche sac-de-mots, le résultat de la normalisation L1 du vecteur de l'image est un vecteur qui peut être interprété comme une distribution multinomiale sur les mots visuels, appelé également histogramme par "sac-de-mots".

B.3.3 Disposition spatiale

Pour certaines tâches, l'agencement des parties d'image est une caractéristique informative, comme par exemple pour la classification de scène, qui consiste à distinguer entre les types de scène (comme "chambre" ou "magasin"). Dans ce cas, la disposition spatiale doit être codée dans la représentation de l'image. L'approche dominante est l'appariement pyramide spatiale (SPM), approche de [Lazebnik *et al.*, 2006]. L'image divisée en plusieurs sous-images, par exemple en quad-arbre, et la fonction des coefficients sont agrégées sur ces sous-images. L'image finale représentation est obtenue par la concaténation de la représentation des sous-images. La même idée a été utilisée dans [Perronnin *et al.*, 2010b] pour enrichir la représentation du vecteur de Fisher avec l'information de disposition spatiale, et dans [Bosch *et al.*, 2007b] où les orientations du gradient sont agrégées sur les sous-images. L'importance relative des sous-images dépend de la tâche. [Bosch *et al.*, 2007b] a exploré l'apprentissage des poids par tâche pour les sous-images.

Le descripteur GIST [Oliva & Torralba, 2001] est une représentation de l'image globale qui caractérise les propriétés de l'image comme "naturel" ou "d'ouverture", en mesurant la distribution spatiale des réponses des filtres d'image spécialement conçus.

B.4 Contributions

Ici, nous décrivons brièvement les principales contributions que nous présentons dans les chapitres de cette thèse.

Dans le Chapter 2, nous décrivons la représentation d'image développée pour la tâche du reclassement d'image, où le but est de trier les images récupérées par la requête texte en tenant compte de leur contenu visuel. La représentation de l'image développée dépend non seulement du contenu visuel de l'image récupérée, mais aussi du contenu des autres images récupérées avec elle, en utilisant la même requête textuelle.

Cela nous permet d'apprendre un modèle de pertinence unique qui, une fois acquis, peut être utilisé pour reclasser les images par pertinence par rapport aux requêtes ignorées lors de l'entraînement du modèle. Les modèles de pertinence appris à partir de cette représentation relative à un requête obtiennent une meilleure performance que les modèles spécifiquement appris pour chaque requête [Schroff *et al.*, 2007] .

La majorité des méthodes de classification d'images se basent sur une représentation d'image apprise de manière non-supervisée. Dans le Chapter 3, nous décrivons une représentation d'image adaptée pour la classification d'images appartenant à une classe spécifique. On construit progressivement le dictionnaire visuel en alternant entre la croissance d'un arbre dont les feuilles correspondent aux mots visuels, et l'apprentissage du modèle de classification linéaire. Par opposition à [Moosmann *et al.*, 2008] qui construit la forêt aléatoire afin de minimiser une classification erronée de région, nous construisons l'arbre afin de minimiser les erreurs de classification d'images. La représentation de l'image résultante est très compacte, rapide à créer, et elle donne d'excellentes performances en utilisant des classifieurs linéaires qui sont rapides à calculer et qui ont des exigences de mémoire limitées.

Les méthodes de l'état de l'art pour la classification d'image ignorent souvent la disposition des régions d'image. Pour décrire la disposition spatiale en utilisant la représentation sac-de-mots, la disposition des sous-images est fixe, comme dans la pyramide spatiale [Lazebnik *et al.*, 2006] où un quad-arbre détermine la disposition des sous-images. Dans le Chapter 4, nous proposons une nouvelle méthode qui est plus souple, en utilisant le principe du vecteur de Fisher pour coder l'aménagement des régions d'image attribuées aux mots visuels. La représentation résultant est beaucoup plus compacte, tout en obtenant un niveau de performance supérieur ou égal à celui des pyramides spatiales.

Bibliography

Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D., & Jordan, M. (2003). Matching words and pictures. *JMLR*, *3*, 1107–1135.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*(4), 509–522.

Ben-Haim, N., Babenko, B., & Belongie, S. (2006). Improving Web-based Image Search via Content Based Clustering. *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*.

Berg, A. C. & Malik, J. (2001). Geometric blur for template matching. In *CVPR*.

Berg, T. & Forsyth, D. (2006). Animals on the web. In *CVPR*.

Bishop, C. (2006). *Pattern recognition and machine learning*. Spinger-Verlag.

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet allocation. *JMLR*, *3*.

Boiman, O., Shechtman, E., & Irani, M. (2008). In defense of nearest-neighbor based image classification. In *CVPR*.

Bosch, A., Zisserman, A., & Munoz, X. (2007a). Image classification using random forests and ferns. In *ICCV*.

Bosch, A., Zisserman, A., & Munoz, X. (2007b). Representing shape with a spatial pyramid kernel. In *CIVR*.

Boureau, Y., Bach, F., Le Cun, Y., & Ponce, J. (2010a). Learning mid-level features for recognition. In *CVPR*.

Boureau, Y.-L., Ponce, J., & Lecun, Y. (2010b). A theoretical analysis of feature pooling in visual recognition. In *ICML*.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Comaniciu, D., Ramesh, V., & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *CVPR*. 142–149.

Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004). Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*.

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

Douze, M., Guillaumin, M., Mensink, T., Schmid, C., & Verbeek, J. (2009). INRIA-LEARs participation to ImageCLEF 2009.

Everingham, M., Van Gool, L., Williams, C., Winn, J., & Zisserman, A. (2009). The PASCAL Visual Object Classes Challenge 2009 results.

Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *PAMI*, *32*(9), 1627–1645.

Fergus, R., Fei-Fei, L., Perona, P., & Zisserman, A. (2005). Learning object categories from Google's image search. In *ICCV*.

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *CVPR*.

Fergus, R., Perona, P., & Zisserman, A. (2004). A visual category filter for Google images. In *ECCV*.

Ferrari, V., Jurie, F., & Schmid, C. (2010). From images to shape models for object detection. *IJCV*, *87*(3), 284–303.

Frankel, C., Swain, M., & Athitsos, V. (1997). WebSeer: an image search engine for the world wide web. In *CVPR*.

Fritz, M. & Schiele, B. (2008). Decomposition, discovery and detection of visual categories using topic models. In *CVPR*.

Fulkerson, B., Vedaldi, A., & Soatto, S. (2008). Localizing objects with smart dictionaries. In *ECCV*.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *36*(1), 3–42.

Harada, T., Nakayama, H., & Kuniyoshi, Y. (2010). Improving local descriptors by embedding global and local spatial information. In *ECCV*.

Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Proc. of Fourth Alvey Vision Conf.*.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, *42*(1/2).

Jaakkola, T. & Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *NIPS*.

Jain, A. K. & Vailaya, A. (1996). Image retrieval using color and shape. *Pattern Recognition*.

Jing, Y. & Baluja, S. (2008). Pagerank for product image search. In *Proceedings of the 17th International Conference on World Wide Web*.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *ICML*.

Jurie, F. & Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *ICCV*.

Kadir, T. & Brady, M. (2001). Saliency, scale and image description. *Int. J. Comput. Vision*, *45*(2), 83–105.

Ke, Y. & Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. In *CVPR*.

Kelley, J. (1960). The cutting-plane method for solving convex programs. *SIAM Journal on Control and Optimization*.

Krapac, J., Allan, M., Verbeek, J., & Jurie, F. (2010). Improving web-image search results using query-relative classifiers. In *CVPR*.

Lavrenko, V. & Croft, W. (2001). Relevance based language models. In *ACM SIGIR*.

Lazebnik, S. & Raginsky, M. (2009). Supervised learning of quantizer codebooks by information loss minimization. *PAMI*, *31*(7), 1294–1309.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*.

Leibe, B., Mikolajczyk, K., & Schiele, B. (2006). Efficient clustering and matching for object class recognition. In *BMVC*.

Leung, T. & Malik, J. (1999). Recognizing surfaces using three-dimensional textons. In *ICCV*. Los Alamitos, CA: IEEE, vol. 2, 1010–1017.

Leung, T. & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, *43*(1), 29–44.

Li, F.-F., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *PAMI*, *28*(4), 594–611.

Li, L.-J., Wang, G., & Li, F.-F. (2007). OPTIMOL: automatic object picture collection via incremental model learning. In *CVPR*.

Li, X., Snoek, C., & Worring, M. (2009). Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, *11*(7).

Lian, X., Li, Z., Lu, B., & Zhang, L. (2010). Max-margin dictionary learning for multiclass image categorization. In *ECCV*.

Lin, W.-H., Jin, R., & Hauptmann, A. G. (2003). Web image retrieval re-ranking with relevance model. In *Web Intelligence*. 242–248.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *Int. J. Comput. Vision*, *30*(2), 79–116.

Liu, D., Hua, G., Viola, P., & Chen, T. (2008). Integrated feature selection and higher-order spatial feature extraction for object categorization. In *CVPR*.

López-Sastre, R., Tuytelaars, T., Acevedo-Rodríguez, F., & Maldonado-Bascón, S. (2011). Towards a more discriminative and semantic visual vocabulary. *CVIU*, *115*(3), 415–425.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, *60*(2), 91–110.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2008). Supervised dictionary learning. In *NIPS*.

Maji, S. & Berg, A. (2009). Max-margin additive models for detection. In *ICCV*.

Manjunath, B. S. & Ma, W. Y. (1996). Texture features for browsing and retrieval of image data. *PAMI*, *18*(8), 837–842.

Mikolajczyk, K. & Schmid, C. (2004). Scale and affine invariant interest point detectors. *IJCV*, *60*(1), 63–86.

Moosmann, F., Nowak, E., & Jurie, F. (2008). Randomized clustering forests for image classification. *PAMI*, *30*(9), 1632–1646.

Moosmann, F., Triggs, B., & Jurie, F. (2007). Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*.

Morioka, N. & Satoh, S. (2010). Building compact local pairwise codebook with joint feature space clustering. In *ECCV*.

Morsillo, N., Pal, C., & Nelson, R. (2009). Semi-supervised learning of visual classifiers from web images and text. In *IJCAI*.

Nistér, D. & Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*.

Nowak, E., Jurie, F., & Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *ECCV*.

Obdrzalek, S. & Matas, J. (2002). Object recognition using local affine frames on distinguished regions. In *BMVC*.

Ojala, T., Pietikäinen, M., & Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, *24*(7), 971–987.

Oliva, A. & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*, 145–175.

Opelt, A. & Pinz, A. (2005). Object localization with boosting and weak supervision for generic object recognition. In *SCIA*.

Opelt, A., Pinz, A., & Zisserman, A. (2006). A boundary-fragment-model for object detection. In *ECCV*.

Palatucci, M., Pomerleau, D., Hinton, G., & Mitchell, T. (2009). Zero-shot learning with semantic output codes. In *NIPS*.

Perronnin, F. (2008). Universal and adapted vocabularies for generic visual categorization. *PAMI*, *30*(7), 1243–1256.

Perronnin, F. & Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *CVPR*.

Perronnin, F., Dance, C., Csurka, G., & Bressan, M. (2006). Adapted vocabularies for generic visual categorization. In *ECCV*.

Perronnin, F., Sánchez, J., & Liu, Y. (2010a). Large-scale image categorization with explicit data embedding. In *CVPR*.

Perronnin, F., Sanchez, J., & Mensink, T. (2010b). Improving the fisher kernel for large-scale image classification. In *ECCV*.

Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *CVPR*.

Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*.

Quack, T., Ferrari, V., Leibe, B., & van Gool, L. (2007). Efficient mining of frequent and distinctive feature configurations. In *ICCV*.

Quattoni, A., Collins, M., & Darrell, T. (2007). Learning visual representations using images with captions. In *CVPR*.

Riesenhuber, M. & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*, 1019–1025.

Roizman, M. & Last, M. (2006). Look-ahead mechanism integration in decision tree induction models. In *Advances in Web Intelligence and Data Mining*.

Savarese, S., Winn, J., & Criminisi, A. (2006). Discriminative object class models of appearance and shape by correlatons. In *CVPR*.

Schroff, F., Criminisi, A., & Zisserman, A. (2007). Harvesting image databases from the web. In *ICCV*.

Schroff, F., Criminisi, A., & Zisserman, A. (2011). Harvesting image databases from the web. *PAMI*, *TODO*, TODO.

Shinohara, Y. & Otsu, N. (2004). Facial expression recognition using fisher weight maps. *Automatic Face and Gesture Recognition*.

Sivic, J. & Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *ICCV*.

Smeulders, A., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. *PAMI*, *22*(12), 1349–1380.

Steger, C. (1998). An unbiased detector of curvilinear structures. *PAMI*, *20*(2), 113–125.

Tommasi, T. & Caputo, B. (2009). The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *BMVC*.

van de Weijer, J. & Schmid, C. (2006). Coloring local feature extraction. In *ECCV*.

van de Weijer, J., Schmid, C., & Verbeek, J. (2007). Learning color names from real-world images. In *CVPR*.

van Gemert, J., Snoek, C., Veenman, C., Smeulders, A., & Geusebroek, J.-M. (2010a). Comparing compact codebooks for visual categorization. *CVIU*, *114*(4), 450–462.

van Gemert, J., Veenman, C., Smeulders, A., & Geusebroek, J.-M. (2010b). Visual word ambiguity. *PAMI*, *32*(7), 1271–1283.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

Varma, M. & Ray, D. (2007). Learning the discriminative power-invariance trade-off. In *ICCV*.

Wang, G., Hoiem, D., & Forsyth, D. (2009). Building text features for object image classification. In *CVPR*.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. In *CVPR*.

Weber, M., Welling, M., & Perona, P. (2000). Unsupervised learning of models for recognition. In *ECCV*.

Winn, J., Criminisi, A., & Minka, T. (2005). Object categorization by learned universal visual dictionary. In *ICCV*. 1800–1807.

Wnuk, K. & Soatto, S. (2008). Filtering internet image search results towards keyword based category recognition. In *CVPR*.

Xiao, J., Hays, J., Ehinger, K., Oliva, A., & Torralba, A. (2010). SUN database: Large scale scene recognition from abbey to zoo. In *CVPR*.

Yanai, K. & Barnard, K. (2005). Probabilistic web image gathering. In *ACM MIR*.

Yang, J., Yu, K., Gong, Y., & Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*.

Yang, J., Yu, K., & Huang, T. (2010a). Efficient highly over-complete sparse coding using a mixture model. In *ECCV*.

Yang, J., Yu, K., & Huang, T. (2010b). Supervised translation-invariant sparse coding. In *CVPR*.

Yang, L., Jin, R., Sukthankar, R., & Jurie, F. (2008). Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*.

Zhang, J., Marszałek, M., Lazebnik, S., & Schmid, C. (2007). Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, *73*(2), 213–238.

Zhang, W., Surve, A., Fern, X., & Dietterich, T. (2009). Learning non-redundant codebooks for classifying complex objects. In *ICML*.

Zhou, X., Cui, N., Li, Z., Liang, F., & Huang, T. (2009). Hierarchical Gaussianization for image classification. In *ICCV*.