8-1-2014

Methods for efficient object categorization, detection, scene recognition, and image search

Alessandro Bergamo
Dartmouth College

METHODS FOR EFFICIENT OBJECT CATEGORIZATION, DETECTION, SCENE RECOGNITION, AND IMAGE SEARCH

DARTMOUTH COMPUTER SCIENCE TECHNICAL REPORT TR2014-764

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Alessandro Bergamo

DARTMOUTH COLLEGE

Hanover, New Hampshire

August 2014

Examining Committee:

(chair) Lorenzo Torresani

Dragomir Anguelov

Chris Bailey-Kellogg

Devin J. Balkcom

F. Jon Kull, Ph.D.
Dean of Graduate Studies

Abstract

In the past few years there has been a tremendous growth in the usage of digital images. Users can now access millions of photos, a fact that poses the need of having methods that can efficiently and effectively search the visual information of interest. In this thesis, we propose methods to learn image representations to compactly represent a large collection of images, enabling accurate image recognition with linear classification models which offer the advantage of being efficient to both train and test. The entries of our descriptors are the output of a set of basis classifiers evaluated on the image, which capture the presence or absence of a set of high-level visual concepts. We propose two different techniques to automatically discover the visual concepts and learn the basis classifiers from a given labeled dataset of pictures, producing descriptors that highly-discriminate the original categories of the dataset. We empirically show that these descriptors are able to encode new unseen pictures, and produce state-of-the-art results in conjunct with cheap linear classifiers. We describe several strategies to aggregate the outputs of basis classifiers evaluated on multiple subwindows of the image in order to handle cases when the photo contains multiple objects and large amounts of clutter. We extend this framework for the task of object detection, where the goal is to spatially localize an object within an image. We use the output of a collection of detectors trained in an offline stage as features for new detection problems, showing competitive results with the current state of the art.

Since generating rich manual annotations for an image dataset is a crucial limit of modern methods in object localization and detection, in this thesis we also propose a method to automatically generate training data for an object detector in a weakly-supervised fashion, yielding considerable savings in human annotation efforts. We show that our automatically-generated regions can be used to train object detectors with recognition results remarkably close to those obtained by training on manually annotated bounding boxes.

To my beautiful wife, Niusha, for being a light in the dark, for your wisdom, your patience, and your infinite love.

Acknowledgment

I want to first and foremost thank my wife, Niusha, who has been more than supportive during my dedication and long hours put into my PhD. I could have not made it without you.

I want to also thank my entire family, for being proud of me and having supported me every day: Mamma, Papa, Laura, Francesca, Hossein, Giti, and Donya.

I also want to thank my advisor Lorenzo Torresani, for having supervised me all these years. Thank you for persuading me into doing a PhD. I am happy you did.

I want to express my gratitude to my thesis committee members. Chris Bailey-Kellogg, and Devin J. Balkcom, thank you to have followed my work from the beginning. Dragomir Anguelov, you have been a tremendous mentor to me. I hope to work with you again soon.

Finally, a special thanks goes to my friends and collaborators Mohammad Rastegari and Chen Fang, with whom our friendship has extended beyond the walls of my PhD, and Sudipta Sinha, who has been a wonderful guru. I learned a lot from all of you.

Contents

1 Introduction **1**

2 Related work **11**

3 Descriptors for object categorization **17**

3.1 Introduction 17

3.2 General framework 18

3.3 PiCoDes 22

3.4 Meta-Classes 27

3.5 Experiments 31

3.5.1 Datasets 31

3.5.2 Low-level descriptors 32

3.5.3 Learning classifier-based descriptors 36

3.5.4 Evaluation setup 39

3.5.5 Experiments on Caltech 256 39

4 Descriptors for scene recognition **60**

4.1 Introduction 60

4.2 Methods 61

4.3 Experiments 65

4.3.1 Datasets 65

5 Descriptors for object detection **70**

5.1 Introduction 70

5.2 DetClassemes 72

5.2.1 Modeling our descriptor 72

5.2.2 Learning the descriptor 74

5.2.3 Utilize our descriptor 74

5.3 Experiments 75

5.3.1 Implementation of our descriptor 75

5.3.2 Experiments on PASCAL 2007 76

6 Weakly-supervised object detection **79**

6.1 Introduction 79

6.2 Self-taught Object Localization 82

6.2.1 Input Grayout 82

6.2.2 Agglomerative Clustering 84

6.3 Weakly-Supervised Detection using STL 86

6.4 Experiments 87

6.4.1 Self-taught Localization 87

6.5 Qualitative results 92

6.5.1 Weakly-supervised Detection 93

7 Software **102**

7.1 vlg_extractor 102

7.2 LIBLINEAR_bitmap 103

Appendix **107**

A Support Vector Machine **108**

B Approximated feature map **111**

C Multiple-kernel combiner **113**

D Object detector model **115**

References **116**

List of Figures

1.1 Application: image search 2

1.2 Text limitations for describing visual information 3

1.3 Application: search in personal photo collections 5

3.1 Visualization of PiCoDes projections 22

3.2 Visualization of the Meta-Class tree 30

3.3 Example images from Caltech-256 33

3.4 Example images from ImageNet 34

3.5 Accuracy versus PCA-compression trade off 38

3.6 Recognition results on Caltech-256 50

3.7 Recognition results on Caltech-256 51

3.8 Recognition results on ILSVRC-2010 53

3.9 Recognition results on ILSVRC 2010 54

3.10 Recognition results on ILSVRC-2010 55

3.11 Meta-Class node levels usage 56

3.12 Different types of Meta-Class trees 57

3.13 Accuracy versus compactness trade off 58

3.14 Compactness versus recognition time 58

3.15 Object-class search results on ILSVRC-2010 59

4.1 Example images from MIT-67 62

4.2 Example images from SUN-397 63

5.1 Detection results on PASCAL-2007 78

6.1 Gray-out technique 80

6.2 Comparison bounding-box proposal methods 89

6.3 Comparison bounding-box proposal methods on the benchmark ILSVRC-2012-LOC 90

6.4 Comparison bounding-box proposal methods 90

6.5 Gray-out technique: successes 98

6.6 Gray-out technique: successes 99

6.7 Gray-out technique: failures 100

6.8 Weakly-supervised detection results on ILSVRC-2012-LOC-200rnd . . 101

List of Tables

3.1 List of image descriptors 36

3.2 Highest and lowest active nodes in the Meta-Class tree 52

3.3 Recognition results on ILSVRC-2010 53

4.1 Categorization results on PASCAL-2007 66

4.2 Recognition results on SUN 397 67

4.3 Recognition results on SUN-397 69

5.1 Detection results on PASCAL-2007 76

6.1 Highest-scoring bounding boxes 94

6.2 Highest-scoring bounding boxes 95

6.3 Weakly-supervised object detection: best and worst classes 97

Chapter 1

Introduction

Nowadays digital images are everywhere, and in large quantity. Advances in camera and storage technologies have made the acquisition and saving of pictures simple and convenient for the common user, who now keeps thousands of documents instead of a few dozen as before. Moreover, web-based platforms for media sharing like *Flickr*, *Facebook*, or *Google+* allow the users to share and have access to millions of visual documents. Online commerce services like *Amazon* or *eBay* all provide pictures of their listed items, in order to allow users to purchase what they like visually. We argue that one of the biggest challenges from a user standpoint is how to search the visual information of interest, in an effective and efficient fashion.

In this thesis we consider the problem of efficient *visual object-category search by a set of examples*. The task is stated as the following: a user defines a query by providing a small collection of images while an automatic system retrieves the relevant images from a large image database. We do not set any constraints about the number or type of query images. Furthermore, the database may contain millions of pictures; nevertheless, we want to perform an accurate search very efficiently.

We can think of several real scenarios why this type of search can be useful. In

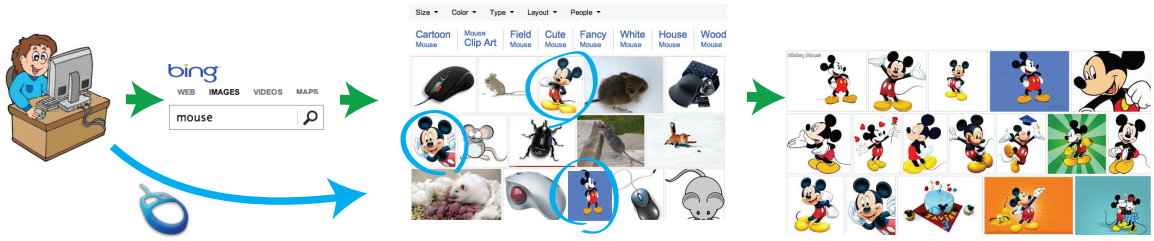


Figure 1.1: Application usage (1): advanced filtering of the pictures retrieved by a classic image-search engine.
The user enters the textual query "mouse" and performs an initial search. The text description is vague, therefore there is a large variety of results: computer input devices, animals, cartoon characters, clipart, etc. The user now selects a few images that best characterize his needs, and the system will filter the results in real-time according to his selection. This multiple selection allows the user to specify a general or more specific new visual category, as he desires. Current filtering options based on size, color, and image type are too simple for this task and non utilitarian.

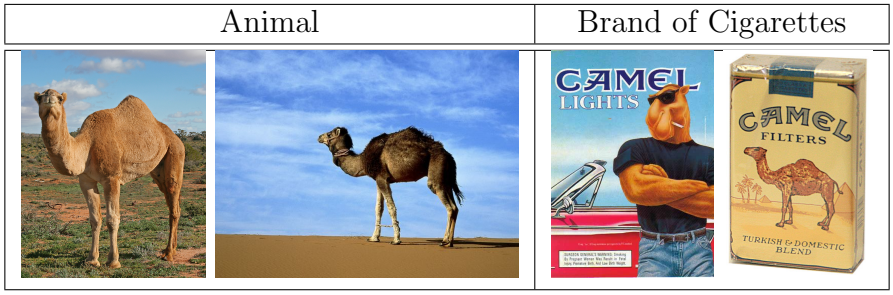
Fig. 1.1 a user is trying to search for an object of interest using an image search engine and providing a text description. We can see that the textual information is ambiguous, and can refer to a large variety of visual content.

Even if a precise text description was provided, the search might not return the desired results, because it may happen that a list of words can be expanded in several visual concepts. For instance the word "hawksbill" denotes both a crag in Arkansas as well as a type of sea turtle. As another example, "camel" can be referred to either the animal or a brand of cigarettes. These pictures may have a little bit in common visually, and form multiple visual clusters as shown in Figure 1.2.

These examples all show that taking the actual visual content of the image into account is of critical importance in order to achieve satisfactory results. In this thesis we want to study methods that allow the user to select a few images that better *visually* describe his needs (see Fig. 1.1, center), and the search engine will either re-rank the previously gathered pictures, or perform a new search from scratch using only the *visual* query provided by the user.

Another interesting application scenario that motivates this thesis, is *visual search*

(a) Hawksbill



(b) Camel

Figure 1.2: Text is very limiting to describe visual information. For instance the word "hawksbill" denotes both a crag in Arkansas as well as a type of sea turtle. The word "camel" can be referred to as the animal or a brand of cigarettes.

in personal photo collections. Users typically store large amounts of personal photographs, a fact that poses non-trivial challenges in searching the photos of interest. The automatic organization of the pictures based on shooting time or manual tagging is not effective in the long term and/or as the collection grows. Automated tagging of pictures is limited to simple, pre-defined categories. The user rarely has time to manually tag the pictures but even if one does, it is not clear which tags will be useful in future needs. Automated tagging of pictures is limited to pre-defined categories that don't go beyond simple concepts (e.g. "mountain", "beach", etc..). In summary, the interactive nature of searching and browsing large image collections calls for the ability to allow users to define their own personal query categories to be recognized. As described in Fig. 1.3, the systems we propose in this thesis avoid any offline labeling and simply allows searching the information only when it is needed.

We believe that defining a *visual* query consisting of multiple images carries much

more expressiveness than a query with a single image. For instance, in the example in Fig. 1.1 if we had selected only the image with the white background we could expect the search engine to gather only images with Mickey Mouse in the foreground and a clean white background, performing almost an object-instance search. Instead, while selecting images with different backgrounds we are implicitly telling the system that that background is variable. This is the peculiarity and key-difference from all other search systems founded on content-based retrieval [58, 15], where the query consists of a single image.

Note that while specifying a small collection of images, the user defines implicitly an arbitrary visual category. However, we need to take into consideration that the system is not aware of this novel category until at query time; this makes the system substantially different from classic recognition methods which assume a fixed set of classes [19, 65].

Moreover, traditional recognition techniques do not pose any constraint regarding the efficiency in learning and testing the model, whereas in our case they are crucial. The user expects results in real-time, thus we require our system to be able to learn the *novel-class object classifier* on-the-fly, and evaluate it on a large-scale image collection accurately and very efficiently. We also require the images to be described with very compact descriptors in order to be able to keep millions of images in memory for faster processing. This is in contrast with the direction taken from most popular object categorization benchmarks [40, 8], that measures the performance of recognition systems solely in terms of classification accuracy over a *predefined* set of classes, and without the consideration of the computational costs.

In the first part of this thesis we consider the problem of designing a system that can address these requirements: our goal is to develop an approach that enables accurate *real-time recognition* of arbitrary categories in gigantic image collections, where

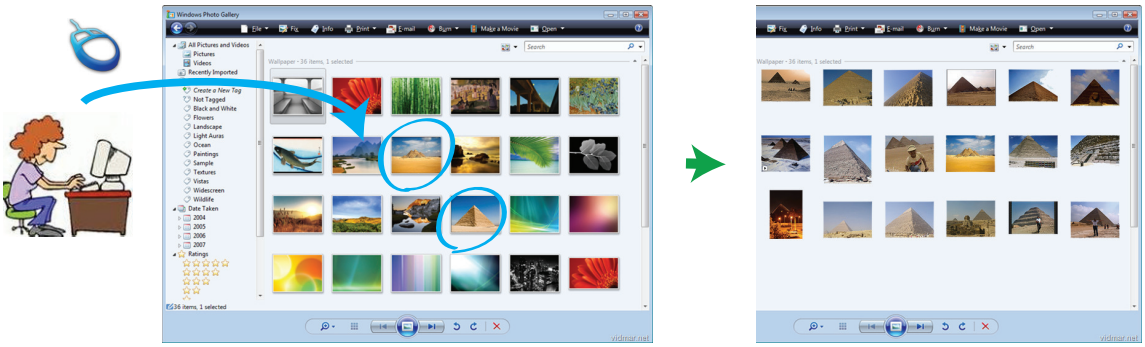


Figure 1.3: Application usage (2): Search in personal photo collections. Bob has tons of pictures of vacations in his personal computer and recently he was in Egypt. He wants to see all the pictures that contains the pyramids so he just clicks on a couple of images and our system will search all the other images containing the pyramids for him in real-time.

the *classes are not defined in advance*, i.e, they are not known at the time of the creation of the database. We propose to achieve this goal by means of image descriptors designed to enable good recognition accuracy with simple linear classifiers, which can be trained efficiently and can be tested in just a few seconds even on databases containing millions of images. Rather than optimizing classification accuracy for a fixed set of classes, our aim is to learn a general image representation which can be used to describe and recognize arbitrary categories, even novel classes not present in the training set used to learn the descriptor.

We follow the framework introduced in [74], where the authors propose to use the outputs of a predefined set of nonlinear classifiers as entries of the image descriptor, which are evaluated on low-level features computed from the photo. This implies that a simple linear classification model applied to this descriptor effectively implements a nonlinear function of the original low-level features. As demonstrated in recent literature on object categorization [36], these nonlinearities are critical to achieve good categorization accuracy with low-level features. However, the advantage of this approach is that our classification model, albeit nonlinear in the low-level features, remains linear in our descriptor and thus it enables efficient training and

testing. In other words, the nonlinear classifiers implementing our features are used as a classification basis to recognize new categories via linear models. Based on this intuition, we refer to our features as *basis classes*. The final classifier for a novel class is obtained by linearly combining the outputs of the nonlinear classifiers, which we can pre-compute and store for every image in the database, thus enabling efficient novel-class recognition even in large datasets. Note that in [74] the authors propose to use a *hand-selected* set of object classes from the real-world as basis classes; in this case each basis classifier is simply trained as a traditional object classification model optimized to recognize that particular *base class*.

In this thesis, we investigate and propose in chapter 3 several strategies to *automatically* discover a suitable set of discriminative *basis classes* and so to train a more effective set of basis classifiers. We indeed demonstrate that a better classifier-based representation can be obtained by learning the basis classes as general "abstract categories" optimized to yield useful features for linear models rather than hand-defining them as real object classes. We propose two distinct strategies to learn automatically the basis classes. The first one is called PiCoDes (appeared first in [10]) and is presented in section 3.3. This approach optimizes the basis classifiers for linear classification, i.e., it trains them to produce good recognition accuracy when used as features with linear models. The second strategy is called Meta-Class (appeared first in [9]) and is presented in section 3.4. This method constrains each basis class to be a super-category obtained by grouping a set of object classes such that, collectively, are easy to distinguish from other sets of categories. We show that our feature vectors provide better accuracy compared to the existing work on learning compact image codes, including [74]. On the Caltech 256 benchmark, a simple linear SVM trained on our representation outperforms the state-of-the-art LP-β classifier [36] trained on the same low-level features used to learn our descriptor. On the 2010 ImageNet Challenge

(ILSVRC-2010) database, linear classification with our features achieves recognition accuracy only 10.3% lower than the winner of the competition [53], whose computational cost for training and testing is several orders of magnitude higher compared to our approach, which is a system that was trained for a week using a powerful cluster of machines, a specialized hardware architecture for memory sharing, and a file system capable of handling terabytes of data. Instead, our approach allows us to fit the entire ILSVRC-2010 training and testing set in the RAM of a standard computer and produce results within a day using a budget PC.

In order to render the descriptor capable of handling the presence of multiple objects and clutter typical of these datasets, we propose in chapter 4 methods to apply our basis classifiers on many subwindows of the photo. We describe and evaluate several strategies to aggregate the features produced from multiple subwindows into a single compact descriptor that can be used by linear classification models. We show that the resulting descriptors yield a significant boost in accuracy compared to feature vectors built from basis classifiers evaluated on the whole image and are on par with specialized object detectors tuned on the test classes. However, the training and testing of linear classifiers using our descriptors have considerably lower computational cost. We present in this thesis the first comprehensive evaluation of classifier-based descriptors on object detection and scene recognition datasets, including the challenging PASCAL-VOC-2007, MIT 67, and SUN 397 benchmarks.

In chapter 5 of this thesis, we also studied the more general problem of object detection. In other words, the task is to detect the presence of the objects of interest, as well as detecting the bounding boxes surrounding them. Object detection provides several benefits over holistic classification, including the ability to localize objects in the image, as well as robustness to irrelevant visual elements, such as uninformative background, clutter, or the presence of other objects. Traditionally, the object de-

tection problem has been tackled by means of subwindow recognition [41, 32], where a classifier is evaluated on an exhaustive set of candidate subwindows yielding high scores for those containing the objects of interest. Given the large number of subwindows to consider, these approaches can be used in practice only with features that are extremely fast to extract (e.g. Histograms of Oriented Gradients (HOG) [14]), since the typical high-dimensionality of image representations poses non-trivial challenges in their storage.

In this work we further increase the sophistication of these object detection models by replacing conventional sub-window representations with higher-level visual information learned during an offline training stage. We propose to describe each candidate subwindow in terms of the output of a set of pre-trained object detectors. Given a new object class to detect, we simply train a linear classifier on this representation, thus using the pre-learned object detectors as a basis for detection of new categories. Our results on the challenging PASCAL-VOC-2007 dataset move towards the best published results in the literature, while being conceptually much simpler to implement compared to the most currently dominant approaches involving part-based objects [32].

We notice that while whole-image classifiers can be trained with image examples labeled merely with class information (e.g. "chair" or "pedestrian"), detectors require richer labels consisting of manual selections specifying the region or the bounding box containing the target object in each individual image example. Unfortunately, such detailed annotations are expensive and time-consuming to acquire. This effectively limits the applicability of object detectors to scenarios involving only few categories (*e.g.*, [38]). Furthermore, these manual selections are often rather subjective and noisy, and being so, they do not provide optimal ground truth regions for training detectors.

In chapter 6 of this thesis, we introduce a novel approach that leverages on deep convolutional networks [47] trained for whole-image recognition to localize objects in images without additional human supervision, i.e., without using any ground-truth bounding boxes for training. The key idea is to analyze the change in the recognition scores when artificially graying out the color pixels of different regions of the image. We observe that graying out a region that contains an object of a certain category, typically causes a significant drop in recognition for that class. This intuition is embedded into an agglomerative clustering technique that generates self-taught localization hypotheses. For a small number of hypotheses, our object localization scheme greatly outperforms prior subwindow proposal methods in terms of both recall and precision [1] .

These subwindows are used as training data to learn a conventional object detector, like the one described in the appendix D. The detectors are weakly-supervised, as we effectively replace the traditional manually-annotated bounding boxes with regions automatically estimated from training images annotated only with whole-image object-class labels. This framework enables scalable training of object detectors at a much reduced human cost, since no manual annotation of regions is needed. We demonstrate that these weakly-supervised detectors achieve recognition accuracy surprisingly close to that obtained by using ground-truth bounding boxes as training data.

This thesis is organized as follows: In chapter 2 we introduce all related prior work relevant for all the topics treated in this thesis. We will discuss all the alternative methods we considered that could be potentially used to tackle the problems discussed in the introduction.

[1] Informally, the *recall* can be viewed as the probability that each object in the image is covered by at least one subwindow, whereas *precision* is the chance that a subwindow covers an object.

Then, in chapter 3 we introduce methods for learning compact global image representations: PiCoDes presented in section 3.3, and Meta-Class in section 3.4. In section 3.5 we present an extensive set of experiments for whole-image categorization on both medium and large scale benchmarks.

Next, in chapter 4 we propose several extensions for these methods to make the descriptor capable of handling the presence of multiple objects and clutter in the pictures. In section 4.3 we propose an exhaustive set of experiments showing the advantages in terms of accuracy of these extensions, providing state-of-the-art results on challenging benchmarks like MIT 67, SUN 397, PASCAL-VOC-2007.

Later, in chapter 5 we extend this framework to tackle the object detection task, showing in section 5.2 how to learn a bank of detectors offline, whose outputs will be used as features. In section 5.3 we experimentally show how these features can be used for a new detection task, showing improvements on the benchmark PASCAL-VOC-2007.

In chapter 6 we present a framework to automatically generate bounding box training data for an object detector, in a weakly-supervised fashion by making use solely of the whole-image class label annotation. We show in section 6.4 that these weakly-supervised detectors yield precision close to the one obtained by making use of human-annotated data. Lastly, in chapter we finally conclude this thesis and discuss the possible future work.

Chapter 2

Related work

The problem of object class recognition in large datasets has been the subject of much recent work. While nonlinear classifiers (see appendix A) are recognized as state-of-the-art in terms of categorization accuracy [36, 47], they are difficult to scale to large training sets. Thus, much more efficient linear models (see appendix A) are typically adopted in recognition settings involving a large number of object classes, with many image examples per class [19]. As a result, much work in the last few years has focused on methods to retain high recognition accuracy even with linear classifiers. We can loosely divide these methods in three categories.

The first category comprises techniques to approximate nonlinear kernel distances via explicit feature maps [55, 79]. For many popular kernels in computer vision, these methods provide analytical mappings to higher-dimensional feature spaces where inner products approximate the kernel distance. This permits to achieve results comparable to those of the best nonlinear classifiers with simple linear models. However, these methods are typically applied to hand-crafted features that are already high-dimensional and they map them to spaces of further increased dimensionality (≥ 3 times as large, depending on the implementation). As a result, they pose high stor-

age costs in large-scale settings. We presented the method introduced in [79] in more detail in the appendix B.

A second line of work [65] involves the use of vectors containing a very large number of features (up to several millions) so as to obtain a high degree of linear separability. The idea is similar to that of explicit feature maps, with the difference that these high-dimensional signatures are not produced with the goal of approximating kernel distances between lower-dimensional features but rather to yield higher accuracy with linear models. In order to be able to keep large datasets in memory with such representations, the vectors are typically stored in compressed form and then decompressed quickly and one at a time during training and testing [65, 44]. An exception is the work of Lin et al. [53] where the high storage and I/O costs caused by their high-dimensional descriptor were absorbed by a large system infrastructure consisting of Apache Hadoop to distribute computation and storage over many machines.

Finally, the third strand of related work involves the use of image descriptors encoding categorical information as features; the image is represented in terms of its closeness to a set of basis object classes [82, 74, 16] or as the response map to a set of detectors [52]. Even linear models applied to these high-level representations have been shown to produce good categorization accuracy. These descriptors can be viewed as generalizing attributes [49, 30, 50], which are semantic characteristics selected by humans as associated with the classes to recognize.

The approaches presented in this thesis in chapter 3 are closely related to this third line of work, as they all represent images in terms of the outputs of classifiers learned for a set of basis classes. However, while the prior work made use of hand-selected categories to define the base classes, our work automatically builds an optimized collection of categories for the given objective. Section 3.2 will describe in detail the differences between our image descriptors and the work introduced in [74].

We note that this line of work is also evocative of the use of attributes [49, 30, 50] which are fully-supervised classifiers trained to recognize certain properties in the image such as "has beak", "near water". While attributes have been used as features for recognition in specialized domains (e.g., animal recognition [50] or face identification [49]), we demonstrate that by choosing a large and varied set of object categories as basis classes, it is possible to employ the resulting descriptor as an effective universal feature representation for general object categorization. Furthermore, we show that this feature vector can be binarized with little loss of recognition accuracy to produce a compact binary code that allows even gigantic image collections to be kept in memory for more efficient testing.

In this thesis we also consider how to extend classifier-based descriptors to yield good recognition accuracy even for photos containing multiple objects and clutter. Operating in these scenarios requires factoring out the background and the irrelevant visual elements, while representing the multiple objects in the scene. Our approach is inspired by Li et al. [52] who have proposed to use the localized outputs of object detectors as image features. The advantage of this representation (called ObjectBank) is that it encodes spatial information. Furthermore, object detectors are more robust to clutter and uninformative background than classifiers evaluated on the entire image. However, their representation is very high-dimensional (the ObjectBank descriptor includes over 40,000 real-valued entries) and as such is not adequate for large-scale recognition. In chapter 4 we will present and test several strategies to aggregate the outputs of basis classifiers evaluated on multiple subwindows in a single *low-dimensional* descriptor.

Deep networks [7] have very recently gained great popularity due to their outstanding performance on different tasks, including large-scale image categorization [48, 24, 69], face verification [73], video recognition [46], and speech recognition [22].

Note that PiCoDes, the method introduced in section 3.3 can be viewed as implementing a form of deep belief network [43], where low-level features are pumped through a set of learned nonlinear functions organized in layers. However, our approach differs from traditional deep learning methods in terms of the training objective, the optimization algorithm, the type of nonlinearities, and also in our use of the output layer as a representation for subsequent recognition.

The problem of object detection has been traditionally approached as the task of exhaustive sub-image recognition [41, 32]: for every category of interest, a classifier is evaluated at every possible rectangular subwindow of the image, thus performing a brute-force sliding window search. In order to maintain the computation manageable despite the large number of subwindows to consider, these approaches are constrained to use features that are extremely fast to extract. Representative efficient sub-image descriptors include the Histograms of Oriented Gradients (HOG) [14] and the Haar features [80] which can be calculated in constant time at every location by using integral images.

Recently, a few authors [2, 75] have introduced the idea of efficiently identifying inside the image the rectangular subwindows that are most likely to contain objects, regardless of their class. Particularly the method of selective search (SS) originally proposed in [75] shows a recall (fraction of the true objects that are identified by the method) approaching 97% for a small number of candidate subwindows (on average about 1500 per image). This desirable property, coupled with the efficiency of their algorithm, implies that relatively few subwindows need to be considered to accurately localize and recognize objects. In turn, this enables the practical application of sophisticated features and object detection models, which instead would be prohibitive in a traditional sliding-window scenario. For example, the system of [75] achieves state-of-the-art results by training a nonlinear SVM on a spatial pyramid

of histograms computed from 3 distinct local appearance descriptors. Despite the complexity of this model, the computational cost of recognition remains low if the classifier is applied only to the 1500 candidate sub-images rather being exhaustively evaluated over all possible subwindows.

Leveraging the prior work on candidate subwindow proposal methods, we propose in chapter 5 a detector-based subwindow descriptor. The approaches presented in [74] and [52] are similar in spirit to ours. Similary to [74] we use the output of non-linear classifiers as features. However, while in [74] the basis classifiers are trained for whole-image classification, in our work we use proper image detectors to better capture the locality of the objects, leveraging the detector model described in the appendix D. Similarly with [52] we use the response of object detectors as features. However, the work of [52] is designed and tackles a scene recognition task, whereas we are focusing on efficient object detection.

In the context of both object localization and detection, several researchers have attempted to apply deep networks [38, 66, 72, 26]. In [38], a convolutional network [48] is fine-tuned on ground truth bounding boxes and then applied to classify subwindows generated by the region proposal algorithm of Uijlings et al. [75]. In [66, 26, 72] a convolutional network is trained to directly perform regression on the vector-space of all bounding boxes of an image in order to avoid the high computational cost of traditional sliding window or region proposal approaches. These deep networks have shown promising results compared to standard detection schemes relying on hand-crafted features [32, 75]. However, nearly of all them require manually-annotated ground truth bounding boxes as training data. In contrast, our method presented in chapter 6 populates the images in the training set with automatically-generated bounding boxes, which can be exploited for the learning of arbitrary detectors.

The subwindow proposal methods [3, 75] mentioned before, focus on generating

bounding boxes in order to maximize recall, thus are typically employed at testing time to replace the classic but computationally expensive sliding window approach. However, these subwindow proposals cannot be used in lieu of ground truth bounding boxes to train a detector because of their low precision (i.e., presence of many false positives). In addition, most of these proposal techniques [3] require ground truth bounding boxes during training, thus effectively increasing the amount of manual annotations needed to train a recognition system. The method we introduced in chapter 6 can also be viewed as a subwindow proposal method but it provides precision much superior to that of prior methods: detectors trained on our automatically-generated bounding boxes perform nearly on par with detectors learned from ground-truth annotations.

Most weakly-supervised object detection methods [23, 13, 68, 70] aim at jointly learning and inferring both the class and the position of the objects. In chapter 6 we do not attempt to estimate both class and location within a single training stage. Instead, we leverage on powerful deep network trained with only class labels to auto-generate training bounding boxes for weakly-supervised detection.

Chapter 3

Descriptors for object categorization

3.1 Introduction

In this chapter we present two compact global image representations, that enable accurate *real-time recognition* of arbitrary categories in gigantic image collection, and are designed to be used with simple linear classifiers, which can be trained efficiently and can be tested in just a few seconds even on databases containing millions of images. Rather than optimizing classification accuracy for a fixed set of classes, our aim is to learn a general image representation which can be used to describe and recognize arbitrary categories, even novel classes not present in the training set used to learn the descriptor.

We made publicly available a software called VLG_EXTRACTOR for Linux, Windows, and Mac to extract the descriptors introduced in this chapter. Please refer to the chapter 7.1 more more details.

3.2 General framework

In this section we introduce our general framework of classifier-based image descriptors. Similarly as introduced in [74], at a high-level, our approach involves representing an image $\boldsymbol{x}$ as a C-dimensional vector $\boldsymbol{h}(\boldsymbol{x})$, where the c-th entry is the output of a classifier h_c evaluated on $\boldsymbol{x}$:

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} h_1(\boldsymbol{x}) \\ \vdots \\ h_C(\boldsymbol{x}) \end{bmatrix} \tag{3.1}$$

The classifiers $h_{1...C}$ (the basis classifiers) are learned during an offline stage from a large labeled dataset of images $\mathcal{D}^S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i$ denotes the i-th image in the database and $y_i \in \{1, \ldots, K\}$ indicates the class label of the object present in the photo. The database $\mathcal{D}^S$ represents our general visual knowledge-base; it should be very large and ideally include all possible visual concepts of the world. Our approaches analyze $\mathcal{D}^S$ to automatically extract the C visual classifiers $h_{1...C}$ to be subsequently used as universal image features for class recognition. We call these C visual features *basis classes*. The work proposed in [74] uses as basis classes a pre-defined set of real word classes, whereas in our work the basis classes are abstract categories, i.e., visual categories that do not necessarily exist in the real-world but that are useful to represent and discriminate the K classes in $\mathcal{D}^S$. This is a critical difference, which lead to non-trivial descriptor learning methods. In the work of [74] indeed, the image representation is constrained to have dimensionality equal to the number of categories of the offline training set $\mathcal{D}^S$ (thus $C = K$). In our work instead, we decouple the number of original classes from the descriptor dimensionality, that can be specified by the user. Note that we encode in the descriptor all the visual

information carried by $\mathcal{D}^S$, and use it to extract a descriptor from a new image that potentially will belong to a visual category not present in $\mathcal{D}^S$.

Before describing how we define and learn the basis classifiers, we want to discuss the motivation behind this representation. Intuitively, our descriptor represents an image in terms of its visual closeness to the set of C basis classes. We propose to extract this descriptor for every image $\boldsymbol{x}$ in the database where we want to perform the final recognition. As shown in our experiments, these classifier-based representation provides us with two fundamental advantages:

1. Compactness. Only C entries are are needed to describe each image. By limiting C to relatively small values (say, a few thousands), we can store even large image collections in memory (rather than on the hard-drive) for more efficient recognition.

2. High accuracy with linear models. This representation enables good classification accuracy even with simple linear models of the form $\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{x})$. Intuitively this happens because a linear combination of these C features implements a powerful weighted sum of C classifiers, which reuses the basis classes as building blocks to recognize novel categories.

Let us now consider the problem of how to define the basis classifiers. We propose to train our basis classifiers on a low-level representation of the image: given an image $\boldsymbol{x}$ we first extract a set of M low-level features $\{\boldsymbol{f}_m(\boldsymbol{x})\}_{m=1}^M$, capturing different visual cues, such as the color distribution, or the spatial layout in the image (our low-level features include common descriptors such as SIFT [54] and GIST [59]; see sec.3.5.2 for further details). In order to obtain a powerful image representation, we want our basis classifiers $h_{1...C}$ to be *nonlinear* functions of features $\{\boldsymbol{f}_m(\boldsymbol{x})\}$. This is motivated by the observation that several recent papers have shown that such nonlinearities

are crucially necessary to achieve good classification accuracy (see, e.g., [36, 74]). However, at the same time we want to define a representation that is efficient to compute. This implies that the basis classifiers $h_{1...C}$ must be fast to evaluate. In order to achieve this twofold purpose, we lift each feature vector $\boldsymbol{f}_m(\boldsymbol{x}) \in \mathbb{R}^{d_m}$ to a higher-dimensional feature space via the explicit map proposed in [79] and described in the appendix B, so that inner products in this space approximate well a predefined nonlinear kernel distance $K_m()$ over the original features $\boldsymbol{f}_m(\boldsymbol{x})$. In other words, this trick allows us to approximate a computationally-expensive non-linear classifier based on kernels, with a linear classifier that makes the overall accuracy of the model drop by only a few percentage points, but allowing us to train the basis classifiers in a considerably reduced amount of time, but most important the time needed to extract our image descriptors of a give image is orders of magnitude smaller.

The concatenation of the M lifted-up features produces a vector $\Psi(\boldsymbol{x})$ of dimensionality $D = \sum_{m=1}^{M} d_m(2r+1)$:

$$\Psi(\boldsymbol{x}) = \begin{bmatrix} \Psi_1(\boldsymbol{f}_1(\boldsymbol{x})) \\ \vdots \\ \Psi_M(\boldsymbol{f}_M(\boldsymbol{x})) \end{bmatrix} \tag{3.2}$$

Using this formulation based on explicit feature maps, we implement each basis classifier h_c as a linear combination of approximate kernels parameterized by a single projection vector $\boldsymbol{a}_c$:

$$h_c(\boldsymbol{x}) = \tau_c(\boldsymbol{a}_c^\top [\Psi(\boldsymbol{f}(\boldsymbol{x})); 1]) \tag{3.3}$$

where τ_c is a function to either quantize or scale the classifier output. The constant value 1 is appended to the vector $\Psi(\boldsymbol{x})$ to implement a bias coefficient without explicitly keeping track of it. We can then stack together the parameters of all basis

classifiers in a single matrix:

$$\boldsymbol{A} = [\boldsymbol{a}_1 \dots \boldsymbol{a}_C]$$

For notational convenience, we set the following equivalences: $h_c(\boldsymbol{x}) \equiv h(\boldsymbol{x}; \boldsymbol{a}_c) = \tau_c(\boldsymbol{a}_c^\top [\Psi(\boldsymbol{f}(\boldsymbol{x})); 1])$ and $\boldsymbol{h}(\boldsymbol{x}) \equiv \boldsymbol{h}(\boldsymbol{x}; \boldsymbol{A})$.

We can now even more clearly recognize the two above-mentioned advantages enabled by our descriptor, i.e, compactness and high accuracy with linear models. Specifically, this formulation implies that a simple linear classification model $\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{x})$ trained for a new class effectively implements an approximate linear combination of multiple low-level kernel distances, using our basis classifiers as individual components:

$$\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{x}) = \sum_{c=1}^{C} \theta_c \tau_c(\boldsymbol{a}_c^\top [\Psi(\boldsymbol{f}(\boldsymbol{x})); 1]) \tag{3.4}$$

The great advantage is that the model $\boldsymbol{\theta}$, albeit nonlinear in the low-level features, remains linear in our descriptor and thus can be efficiently trained and tested. Moreover, note that while the extraction of the low-level features $\{\boldsymbol{f}_m(\boldsymbol{x})\}_{m=1}^M$ is needed to calculate the descriptor $\boldsymbol{h}(\boldsymbol{x})$, the storage of these low-level features is not necessary for the subsequent recognition. This implies that only the compact feature vectors $\boldsymbol{h}(\boldsymbol{x})$ need to be stored in the database.

Note that this approach can also be viewed as a dimensional reduction method: the function $\boldsymbol{h}$ indeed maps the high-dimensional descriptor represented by Eq. 3.2, to a space of much lower dimensionality, i.e. $C \ll D$. In the experimental section we will show that such representation is able to retain or even enrich the original representation, allowing efficient linear classifiers to achieve state-of-the-art classification accuracy on several challenging benchmarks.

In the next sections we propose different methods to discover the basis classes and learn the basis classifiers from a given database $\mathcal{D}^S$.

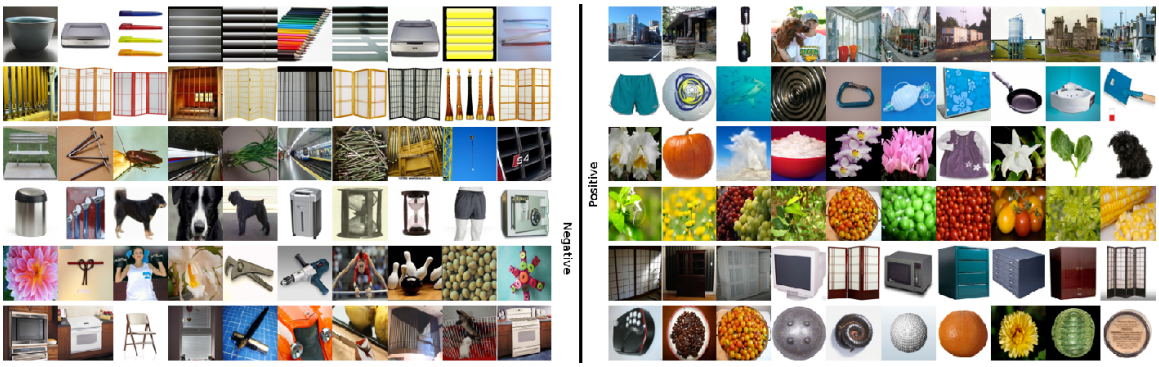


Figure 3.1: Visualization of PiCoDes. Each row of images illustrates a particular bit in our 128-bit descriptor. These 6 randomly chosen bits are shown as follows: for bit c, all images are sorted by non-binarized classifier outputs $\boldsymbol{a}_c^\top \boldsymbol{x}$ and the 10 smallest and largest are presented on each row. Note that $\boldsymbol{a}_c$ is defined only up to sign, so the patterns to which the bits are specialized may appear in either the "positive" or "negative" columns.

3.3 PiCoDes

The idea of the PiCoDes approach is quite simple: given that we want to use basis classifiers as features with linear models, we design our learning objective to enforce that linear combinations of these basis classifiers must yield good accuracy with respect to our training set. In other words, rather than hand-designing the basis classes as done in [74], we learn abstract categories aimed at optimizing linear classification when they are used as features. This learning objective decouples the number of training classes from the target dimensionality of the binary descriptor and thus it allows us to optimize our descriptor for any arbitrary length. Furthermore, it enables direct optimization of the learning parameters with respect to the desired quantization function τ_c, avoiding the suboptimal use of quantizers applied as a post-processing.

Figure 3.1 provides a visualization of a few basis classifiers learned by our training procedure. This figure suggests that our learned features describe the image in terms of binary visually-interpretable properties corresponding, e.g., to particular shape,

texture or color patterns.

We first introduced this approach in [10]. We called these features PiCoDes, which stands for "Picture Codes" but also "Pico-Descriptors". In order to obtain a highly compact descriptor, we optimize the PiCoDes features to be *binary*, i.e., $h(\boldsymbol{x}; \boldsymbol{a}_c) = \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x} > 0]$ (note that this is the same form as in Eq. 3.1 with only difference that here we choose $\tau_c(z) = \mathbf{1}[z > 0]$).

Our work improves the technique presented in [74] in the following aspects:

- Our basis classifiers are defined formally and learned jointly by optimizing an objective that is directly related to their actual usage as features at application time (see Eq. 3.4). In the prior work instead the learning of the basis classifiers is completely disjoint from their actual usage.

- The basis classes are *not* hand-selected real object classes, but automatically discovered from a given labeled training set. In the prior work instead, the choice of classes to use is left to the designer of the descriptor but there is no well-defined criterion to determine the best set of categories.

- We directly optimize to learn a binary descriptor, whereas prior work applies a quantization scheme as a post-processing and is not considered in the learning objective of the basis classifiers.

Learning the basis classifiers

In order to learn the PiCoDes basis classifiers, we use a binary encoding of the labels for the training examples in $\mathcal{D}^S$: for each example i, we define $\boldsymbol{y}_i \in \{-1, +1\}^K$ to be a vector describing its category label, with $y_{ik} = +1$ iff the i-th example belongs to class k. As seen shortly, this labeling will allow us to implement a form of "one-vs-the-rest" strategy for training the basis classifiers. We will continue to let C stand

for the dimensionality (i.e., number of bits) of our code.

We then define our c-th basis classifier to be a boolean function of the form of the Eq. 3.3, i.e. a thresholded *nonlinear* projection of the original low-level features $\boldsymbol{f}$, parameterized by $\boldsymbol{a}_c \in \mathbf{R}^n$. We want to optimize the parameter matrix $\boldsymbol{A}$ so that linear combinations of the resulting basis classifiers yield good categorization accuracy on $\mathcal{D}^S$. In order to formalize this objective, for each training class k we introduce auxiliary parameters $(\boldsymbol{w}_k, b_k)$, which define a linear classifier operating on the PiCoDes features and distinguishing the k-th class from all the others. Thus, we state our overall objective as *joint* optimization over our representation (parameterized by $\boldsymbol{A}$) and the auxiliary parameters defining the K linear classifiers so as to obtain the best possible linear classification accuracy on $\mathcal{D}^S$.

Specifically, we use a large-margin formulation which trades off between a small classification error and a large margin when using the output bits of the basis classifiers as features in a one-versus-all linear SVM:

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2}\|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell\left[y_{i,k}(b_k + \boldsymbol{w}_k^\top \boldsymbol{h}(\boldsymbol{x}_i; \boldsymbol{A}) \right] \right\} \tag{3.5}$$

where $\ell[\cdot]$ is the traditional hinge loss function. Expanding, we get

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2}\|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell\left[y_{i,k}(b_k + \sum_{c=1}^{C} w_{kc} \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0]) \right] \right\}$$

We propose to jointly learn the linear SVMs $(\boldsymbol{w}_k, b_k)$ and the parameters $\boldsymbol{a}_c$ of the basis classifiers using the method described below.

Optimization

Minimizing this objective requires optimization directly for binary features, which is difficult and nonconvex. Thus, we use an alternation scheme implementing block-coordinate descent. We alternate between the two following steps:

1. Learn classifiers.

We fix $\boldsymbol{A}$ and optimize the objective with respect to $\boldsymbol{w}$ and $\boldsymbol{b}$ jointly. This optimization is convex and equivalent to traditional linear SVM learning.

2. Learn PiCoDes

Given the current values of $\boldsymbol{w}$ and $\boldsymbol{b}$, we minimize the objective with respect to $\boldsymbol{A}$ by updating one basis-classifier at a time.

First, we rewrite our objective function, i.e., eq. 3.5, in expanded form:

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2}\|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell\left[y_{ik}(b_k + \sum_{c=1}^{C} w_{kc}\mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0]) \right] \right\}.$$

Fixing the parameters $\boldsymbol{w}_{1..K}, \boldsymbol{b}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{c-1}, \boldsymbol{a}_{c+1}, \ldots, \boldsymbol{a}_C$ and minimizing the function above with respect to $\boldsymbol{a}_c$, is equivalent to minimizing the following objective:

$$E'(\boldsymbol{a}_c) = \sum_{k=1}^{K} \sum_{i=1}^{N} \ell\left[y_{ik} w_{kc} \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0] + y_{ik} b_k + \sum_{c' \neq c} y_{ik} w_{kc'} \mathbf{1}[\boldsymbol{a}_{c'}^T \boldsymbol{x}_i > 0] \right].$$

Let us define $\alpha_{ikc} \equiv y_{ik} w_{kc}$, and $\beta_{ikc} \equiv (y_{ik} b_k + \sum_{c' \neq c} y_{ik} w_{kc'} \mathbf{1}[\boldsymbol{a}_{c'}^T \boldsymbol{x}_i > 0])$. Then, we can rewrite the objective as follows:

$$\begin{aligned} E'(\boldsymbol{a}_c) &= \sum_{k=1}^{K} \sum_{i=1}^{N} \ell\left[\alpha_{ikc} \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0] + \beta_{ikc} \right] \\ &= \sum_{i=1}^{N} \left[\mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0] \sum_{k=1}^{K} \ell(\alpha_{ikc} + \beta_{ikc})(1 - \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0]) \sum_{k=1}^{K} \ell(\beta_{ikc}) \right] \end{aligned}$$

$$= \sum_{i=1}^{N} \left[\mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0] \sum_{k=1}^{K} \ell(\alpha_{ikc} + \beta_{ikc}) - \ell(\beta_{ikc}) \right] + const \ .$$

Finally, it can be seen that optimizing this objective is equivalent to minimizing

$$E(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \mathbf{1}[z_i \boldsymbol{a}_c^T \boldsymbol{x}_i > 0] + const \tag{3.6}$$

Unfortunately, this objective is not convex and not trivial to optimize. Thus, we replace it with the following convex upper bound defined in terms of the hinge function ℓ:

$$\hat{E}(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \ell(z_i \boldsymbol{a}_c^T \boldsymbol{x}_i) \tag{3.7}$$

This objective can be efficiently globally optimized using an LP solver. However, note that the objective is equivalent to the one of a SVM without regularizer and individually-weighted slack variables. In practice, we had optimized this formulation with the "Weights for data instances" version of LIBLINEAR [29], setting the hyper-parameter to a very high value, thus canceling the effect of the regularizer.

Note that we have also experimented with several other optimization methods, including Stochastic Gradient Descent (SGD) applied to a modified version of our objective, where we replaced the binarization function $h(\boldsymbol{x}; \boldsymbol{a}_c) = \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x} > 0]$ with the sigmoid function $\sigma(\boldsymbol{x}; \boldsymbol{a}_c) = 1/(1 + \exp(-\frac{2}{T} \boldsymbol{a}_c^T \boldsymbol{x}))$ to relax the problem. This type of minimization is similar to that traditionally used in neural networks, with the difference that here we are optimizing a large-margin multiclass objective with respect to our PiCoDes representation. However, the issue with this minimization strategy is that it no longer optimizes for pure *binary* features, which is the way we intend to use them at application time: after learning, we want to replace back $\sigma(\boldsymbol{x}; \boldsymbol{a}_c)$ with $h(\boldsymbol{x}; \boldsymbol{a}_c)$ to obtain binary descriptors. In practice, we found that for this reason codes

optimized in this fashion performed much worse than those directly learned via the coordinate descent procedure described in section 3.3 of the paper.

3.4 Meta-Classes

We have seen that the PiCoDes approach described in the previous section provides principled learning of abstract basis classes explicitly optimized for good accuracy with linear models, but it requires a computationally expensive minimization, which in practice can be run only for very compact dimensionalities (our largest PiCoDes contain 2048 bits and required several weeks to be learned). We now introduce a descriptor-learning algorithm that combines the advantages of having a scalable learning method, and still the automatic discovery of abstract basis-classes yielding good recognition when used as features with linear models. We originally presented this approach in [9].

We refer to the basis classes learned by this method as "meta-classes". Intuitively, we want our meta-class classifiers to be "repeatable" (i.e., they should produce similar outputs on images of the same object category) and to capture properties of the image that are useful for categorization. We formalize this intuition by defining each meta-class to be a subset of object classes in the training set. Specifically, we hierarchically partition the set of training object classes such that each meta-class subset can be easily recognized from the others. This criterion forces the classifiers trained on the meta-classes to be repeatable. At the same time, since the meta-classes are superclasses of the original training categories, by definition the classifiers trained on them will capture common visual properties shared by similar classes while being effective to discriminate visually-dissimilar object classes.

Each basis classifier h_c is implemented as a LP-β classifier [36], while the func-

tion τ_c performs either a hard or soft thresholding of the input value. The hyphotesis h_c is learned from a training set $\mathcal{D}_c^S$ containing the images of meta-class c as positive examples, and of other meta-classes as negative images. Note that the *meta-classes* are abstract basis classes, as they are not necessarily present in the real-world.

Discovering the meta-classes

In this section we describe the procedure to discover the meta-classes. Our method is an instance of the algorithm for label tree learning described in [6] [1] . This algorithm learns a tree-structure of classifiers (the label tree) and was proposed to speed up categorization in settings where the number of classes is very large. Instead, here we use the label tree training procedure to learn meta-classes, i.e, sets of classes that can be easily recognized from others. We provide below a review of the label tree algorithm, contextualized for our objective.

Let $\ell_{\mathcal{D}}$ be the set of distinct class labels in the training set $\mathcal{D}^S$, i.e. $\ell_{\mathcal{D}} \equiv \{1, \ldots, K\}$. The label tree is generated in a top-down fashion starting from the root of the tree. Each node has associated a set of object class labels. The label set of the root node is set equal to $\ell_{\mathcal{D}}$. Let us now consider a node with label set ℓ. We now describe how to generate its two children [2] . The two children define a partition of the label set of the parent: if we denote with ℓ^L and ℓ^R the label sets of the two children, then we want $\ell^L \cup \ell^R = \ell$ and $\ell^L \cap \ell^R = \emptyset$. Ideally, we want to choose the partition $\{\ell^L, \ell^R\}$ so that a binary classifier $h_{(\ell^L, \ell^R)}(\boldsymbol{x})$ trained to distinguish these two meta-classes makes as few mistakes as possible. In order to select the best classifier, we should train a classifier for each of the possible $(|\ell|(|\ell|-1)/2-1)$ partitions

[1] Note that other label-tree learning methods, such as [35, 21], could also be applied to our task of meta-class training.

[2] Although the label tree can have arbitrary branching factor at each node, in our work we use binary trees. We tried to learn a label tree with branching factor equal to three, and obtained similar performances.

of ℓ, but this operation is prohibitively expensive. Instead, we take inspiration from the work of [6] and we use the confusion matrix of one-vs-the-rest classifiers learned for the individual object classes to determine a good partition of ℓ: intuitively, our goal is to include classes that tend to be confused with each other in the same label subset. More formally, let $\hat{h}_1, \ldots, \hat{h}_{|\ell_\mathcal{D}|}$ be the one-vs-the-rest LP-β classifiers learned for the individual object classes using the training set $\mathcal{D}^S$. Let $A \in \mathbb{R}^{|\ell_\mathcal{D}| \times |\ell_\mathcal{D}|}$ be the confusion matrix of these classifiers evaluated on a separate validation set $\mathcal{D}^{\text{val}} \subset \mathcal{D}^S$: A_{ij} gives the number of samples of class i in $\mathcal{D}^{\text{val}}$ that have been predicted to belong to class j (we assume the winner-take-all strategy for multiclass classification). Since this matrix is not symmetric in general, we compute its symmetrized version as $B = (A + A^T)/2$. Then, for each node we propose to partition its label set ℓ into the subsets $\ell^L \subset \ell$, $\ell^R \equiv \ell - \ell^L$ that maximize the following objective:

$$E(\ell) = \sum_{i,j \in \ell^L} B_{ij} + \sum_{p,q \in \ell^R} B_{pq} \ . \tag{3.8}$$

The objective encourages to include in the same subset classes that are difficult to tell apart, thus favoring the creation of meta-classes containing common visual properties. At the same time, maximizing this objective will tend to produce meta-classes ℓ^L, ℓ^R that are easy to separate from each other. Let I^ℓ denote the indexes of training examples having class labels in ℓ and we form the labeled set $\mathcal{D}_{(\ell^L, \ell^R)} = \{(\boldsymbol{x}_i, +1) : i \in I^{\ell^L}\} \cup \{(\boldsymbol{x}_i, -1) : i \in I^{\ell^R}\}$ We repeat this process recursively on each node until it contains a single class label, i.e., $|\ell| = 1$, producing in total $\tilde{C}$ labeled sets.

Note that optimization of eq. 3.8 can be formulated as a graph partitioning problem [81]. We compute the solution ℓ^L by applying spectral clustering [57] to the matrix B: this is equivalent to solving a relaxed, normalized version of eq. 3.8 that penalizes unbalanced partitions. We repeat this process recursively on each node until

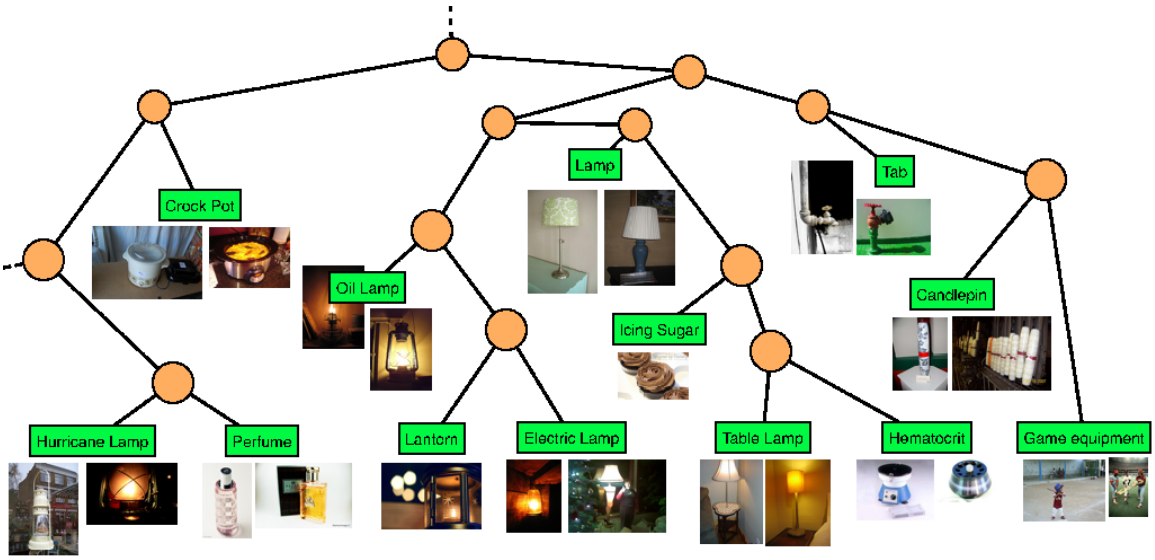


Figure 3.2: *Meta-class tree.* This figure shows a small portion of the tree learned by the *meta-class* approach described in Sec 3.4. The rectangular nodes represent the leaves of the tree, associated with real categories of the *ImageNet* dataset. The inner round nodes represent the meta-classes automatically learned by our method, which tends to group together categories that are difficult to tell apart.

it contains a single class label, i.e., $|\ell| = 1$.

Fig. 3.2 shows a portion of our meta-class tree learned from the *ImageNet* dataset [19]. As expected, we found that our meta-classes tend to group together object classes that are visually similar although not necessarily semantically related (e.g., lantern and electric lamp but also hurricane lamp and perfume).

Learning the meta-class classifiers

The learning of the basis classifiers is now straightforward: at each node ℓ of the label tree we train an LP-β classifier (see appendix C) on the binary split $\{\ell^L, \ell^R\}$. Specifically, let I^ℓ denote the indexes of training examples having class labels in ℓ. Then, we form the labeled set $\mathcal{D}_{(\ell^L,\ell^R)} = \{(\boldsymbol{x}_i, +1) : i \in I^{\ell^L}\} \cup \{(\boldsymbol{x}_i, -1) : i \in I^{\ell^R}\}$ and use it to train meta-class classifier $h_{(\ell^L,\ell^R)}(\boldsymbol{x})$. This procedure yields in total $\tilde{C}$ basis classifiers, where $\tilde{C}$ is the number of inner nodes of the label tree. Note

that this learning is embarrassingly parallelizable as we can learn the hypothesis independently. We also include in the descriptor the outputs of the one-vs-the-rest classifiers $\hat{h}_1, \ldots, \hat{h}_{|\ell_{\mathcal{D}}|}$, as we have found that this improves the final performance of the classifier. The image descriptor $\boldsymbol{h}(\boldsymbol{x})$ is then a C-dimensional vector, where $C = \tilde{C} + K$. We refer to the classifiers $\hat{h}_1, \ldots, \hat{h}_{|\ell_{\mathcal{D}}|}$ as "classemes", as it represents in practice a modified more efficient version of the method proposed in [74].

We experiment with two versions of the function τ in Eq. 3.1, thus giving rise to the two following variants of the meta-class descriptor:

- MC: We convert the raw score of each LP-β classifier into a probabilistic output by means of a sigmoid function, i.e., we set $\tau_c(z) = 1/(1 + \exp(-\alpha_c z + \beta_c))$. We learn the parameters of the sigmoid by means of Platt's scaling procedure [62], using the validation set $\mathcal{D}^{\text{val}}$. We found that this sigmoidal normalization yields a large boost in the accuracy of the final classifier trained on this representation, probably as it makes the range of classification scores more homogeneous and reduces outlier values.

- MC-BIT: We create a binary vector by setting set $\tau_c(z) = \mathbf{1}[z > 0], \forall c \in \{1, \ldots, C\}$.

3.5 Experiments

3.5.1 Datasets

In this work we make use of several datasets, for both learning and testing our descriptors. Here we summarize briefly their characteristics:

- Caltech 256 [40]: Popular benchmark for object categorization with $\sim$ 30K images

partitioned into 256 visual categories. Each photo contains a single centered object. Some example images are shown in figure 3.3.

- ImageNet [19] (Spring 2010 release): Large-scale image dataset consisting of more than 15K object categories and 11M pictures. Most images contain a single object. ImageNet is based on a collection of visual concepts, that are organized hierarchically according to WordNet[31]. Each concept, also called "synset", is described by a set of words or word phrases (only nouns). The images have been manually annotated, and most of them contain a single object. Some example images are shown in figure 3.4.

- ILSVRC-2010 [8]: Large-scale benchmark for object categorization. It is a subset of the ImageNet dataset: it contains 1000 categories and 1.2M images. The training set contains a variable number of examples per class, from a minimum of 619 to a maximum of 3047 yielding in total about 1.2M examples. The validation and test sets have 50 and 150 images per category, respectively.

3.5.2 Low-level descriptors

As previously described in Sec. 3.2, our descriptors are built upon a set of low-level features. For the features that are based on the aggregation of local descriptors, we also exploit the Spatial Pyramid method [51] to encode weak geometry into the representation. In particular we use a pyramid (SP) of L layers, and for each layer $l \in \{0, \ldots, L-1\}$ we partition the image into a $2^l \times 2^l$ grid of cells and we extract a low-level feature vector from each cell. We use the following low-level features:

- *Color GIST* [59]: we first resized the images to 32×32 pixels, without maintaining the aspect ratio, and then we compute the orientation histograms on a 4×4 grid.

Figure 3.3: Example images from the Caltech 256 dataset for the category "dog"

We use 3 scales with the number or orientation per scale being 8, 8, 4.

- *Oriented HOG* [14] (4 SP layers): Histogram of Oriented Gradients computed using 20 bins.

- *Unoriented HOG* [14] (4 SP layers): an histogram of unoriented gradients quantized into 40 bins.

- *SSIM* [67] (3 SP layers): we compute a histogram by extracting the 30-dimensional

Figure 3.4: Example images from the ImageNet dataset for the category "dog"

self-similarity descriptor every 5 pixels, and by quantizing it into 300 cluster centroids obtained from K-means.

- *SIFT* [54] (3 SP layers): We extract the *SIFT* [54] features for our descriptor according to the following pipeline. We first convert each image to gray-scale, then we normalize the contrast by forcing the 0.01% of lightest and darkest pixels to be mapped to white and black respectively, and linearly rescaling the values in between. All images exceeding 786,432 pixels of resolution are downsized to this maximum

value while keeping the aspect ratio. The 128-dimensional SIFT descriptors are computed from the interest points returned by a DoG detector [78]. We finally compute a Bag-Of-Word histogram of these descriptors, using a K-means vocabulary of 500 words. The 128-dimensional SIFT descriptors are computed from the interest points returned by a *Difference of Gaussians* (*DoG*) detector [78]. We finally compute a Bag-Of-Word histogram of these descriptors, using a K-means vocabulary of 500 words.

In our work we treat each pyramid layer as if it was a separate low-level feature vector. The concatenation of all these $M = 15$ feature vectors, yields a $22,860$-dimensional vector. As described in Sec. 3.2 we make use of the explicit map proposed in [79] to perform efficient non-linear classification with an approximated Intersection Kernel. The map is a function $\Psi(\boldsymbol{f}; r, L, \gamma)$ that takes a feature vector $\boldsymbol{f}$ as input and is parametrized by r (number of samples), L (period of the sampling), and γ (normalization of the kernel). We set $r = 1$ for all the features, producing feature vectors three times as big as the original vectors, and $\gamma = 1$ as suggested in [79]. For each feature vector, we select the parameter L by grid search, minimizing the error between the exact kernel distance K, and the approximated one, i.e. $\min_L \sum_{i,j=1}^{N} \left| \left\langle \Psi(\boldsymbol{f}_i), \Psi(\boldsymbol{f}_j) \right\rangle - K(\boldsymbol{f}_i, \boldsymbol{f}_j) \right|$ computed on a validation set consisting of $N = 2560$ images randomly sampled from the Caltech 256 dataset [40]. The concatenation of all these mapped low-level feature vectors form the vector Ψ introduced in Eq. 3.2, which for this implementation has dimensionality $D = 68,580$. In this chapter we will refer to this descriptor as Psi. Finally, we want to emphasize that our approach is obviously not constrained to work only with the particular choice of low-level features described above. Therefore, it could easily take advantage of more powerful low-level features, such as the recently introduced Fisher Vectors [65], which have been shown to lead to state-of-the-art results in object categorization.

Name	**Dimens.**	**Storage size per image**
Our descriptors		
PSI (Sec. 3.5.2)	68580	268 KB
PICODES (Sec. 3.3)	2048 (bin)	0.25 KB
MC (Sec. 3.4)	15232	60 KB
MC-BIT (Sec. 3.4)	15232 (bin)	1.9 KB
MC-LSH (Sec. 3.4)	200 K (bin)	25 KB
X+SPCAT L0L1 (Sec. 4.2)	5×	5×
X+SPLPOOL L0L1 (Sec. 4.2)	2×	2×
X+OBJPOOL (Sec. 4.2)	2×	2×
Prior work		
Torresani et al. 2010, [74] (CLASSEMES)	2659	11 KB
Torresani et al. 2010, [74] (CLASSEMES-BIT)	2659 (bin)	0.33 KB
Gong et al. 2011, [39] (ITQ)	2048 (bin)	0.25 KB
Gong et al. 2011 [39] (CCA-ITQ)	2048 (bin)	0.25 KB
Li et al. 2010 [52] (OBJECTBANK)	44.6 K	175 KB
Lin et al. 2011 [53]	1.1 M	4.5 MB
Sanchez et al. 2011 [65] (FV)	1 M	4 MB
Harzallah et al. 2009 [41]	69.3 K	4.78 MB
Song et al. 2011 [71]	192 K	750 KB
Elfiky et al. 2012 [25]	18 K	71 KB
Xiao et al. 2010 [83]	40 K	155 KB
IFK [61]	262,144	1 MB

Table 3.1: Descriptors considered in our comparison. This table presents: the name of the descriptor and the section where it is introduced; the native dimensionality and whether the descriptor is binary; the required memory to store a single image descriptor.

3.5.3 Learning classifier-based descriptors

In this section we describe in detail how we implemented the classifier-based descriptors proposed in sections 3.3 and 3.4. A summary of the descriptors described in this section and others from prior work is provided in Table 3.1.

PiCoDes

We built the training set $\mathcal{D}^S$ from 2659 randomly selected ImageNet synsets using 30 images per category, for a total of $\sim$ 80K images. In order to avoid pre-learning the test classes in the descriptor, we avoided picking as training synsets categories belonging to the ILSVRC2010 dataset or related to Caltech 256 (we performed sub-string matching comparison between the synset tags and the Caltech 256 keywords, removing in total 711 ImageNet classes). This allows us to evaluate our descriptor in a scenario where each test class to recognize is effectively novel, i.e., not present in the training set used to learn the descriptor.

Note that the learning procedure described in Sec. 3.3 is not easily parallelizable, and it requires continuous access to the high-dimensional image descriptors $\Psi(\boldsymbol{x})$ of Eq. 3.2, which are very costly in terms of storage (see Psi in Tab. 3.1). Thus, in practice, we compress down the vectors via PCA, producing a vector of 6415 dimensions. More formally, for the PiCoDes learning, we set $\Psi(\boldsymbol{x}) \equiv \mathbf{P}\Psi(\boldsymbol{x})$ where $\mathbf{P}$ contains the top 6415 PCA components. The dimensionality of 6415 was chosen based on a preliminary experiment of multiclass classification on Caltech 256, which is shown in figure 3.5. We used a linear SVM trained on image descriptors $\Psi(\boldsymbol{x})$ of varying dimensionality. This study showed that the vector of 6415 dimensions causes only a small drop in accuracy ($\sim$ 1%) compared to the full 68K-dimensional feature vector.

Given the training set $\mathcal{D}^S$ we can train the PiCoDes descriptor as described in Sec. 3.3, performing 15 iterations. For each target dimensionality C, we learned multiple descriptors using different values for the hyper-parameter λ, and kept the descriptor that gave the lowest multiclass error on a validation set consisting of 5 images per class.

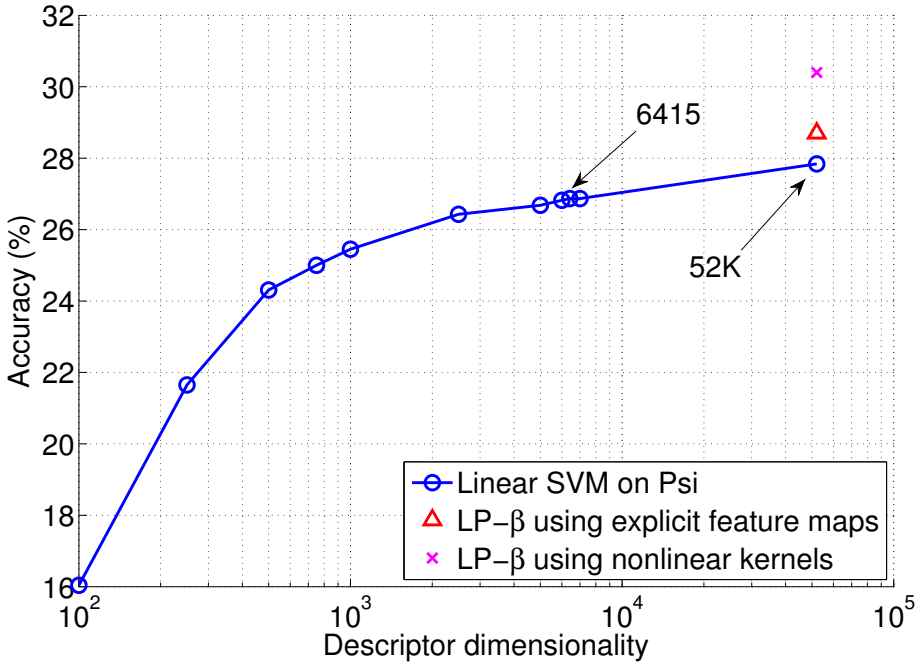


Figure 3.5: The accuracy versus compactness trade off. The benchmark is Caltech256, using 10 examples per class. The pink cross shows the multiclass categorization accuracy achieved by an LP-β classifier using exact kernel distances; the red triangle is the accuracy of an LP-β classifier that uses "lifted-up" features to approximate kernel distances; the blue line shows accuracy of a linear SVM trained on PCA projections of the lifted-up features, as a function of the number of PCA dimensions.

Meta-Classes

We formed the training set $\mathcal{D}^S$ from 8000 randomly sampled synsets from the ImageNet dataset, using at most 1000 examples per category. We also created a validation set $\mathcal{D}^{\text{val}}$ with the same categories and 80 examples per class. As done for PiCoDes, we selected the training classes such that the synsets of these categories do not contain any of the Caltech 256 or ILSVRC2010 class labels, so as to avoid "pre-learning" the test classes during the feature-training stage. We learned the MC and MC-BIT descriptors following the procedure described in Sec. 3.4, using the validation set to compute the confusion matrix and the two parameters of the sigmoid function for

each meta-class. Since the basis classifiers can be learned independently, we use a cluster of multiple machines to reduce the training time.

3.5.4 Evaluation setup

The next sections will describe the experimental evaluations that we have done using our descriptors, as well as other popular image representations. All the experiments involve the usage of a classifier to predict the correct object/scene label of an image, and we decided to use the very efficient linear SVM model (with the only exception of LP-β). In cases of multi-class categorization, we used the 1-vs-the-rest strategy and performed the prediction using the winner-take-all strategy. The SVM hyperparameter is selected using either 5-fold cross validation or using the validation set when available. The calculate the accuracy as the mean of the diagonal of the confusion matrix.

3.5.5 Experiments on Caltech 256

We present experiments obtained with several image descriptors on the challenging Caltech 256 benchmark. We follow the standard approach of learning the classifier for different number of training examples per class $\{5, 10, \ldots, 50\}$. We evaluate the model on a test set consisting of 25 examples per class. We compare our descriptors PiCoDes, mc and mc-bit with the following methods:

- Psi: the concatenation of the mapped low-level features (introduced in Sec. 3.5.2). This baseline is interesting to consider as it shows the accuracy that can be obtained by directly training the final classifier on the low-level image representation that we have used to learn our descriptors.

- LSPM and KSPM: these baseline methods are the Spatial Pyramid Matching (SPM) method [51] with a linear and Chi-square kernel respectively, and SIFT features, as reported from the work of [84].

- ITQ: the embedding method introduced in [39], which learns a binary code by directly minimizing the quantization error of mapping the input data to vertices of the binary hypercube. As training data for the learning we use 2560 images from the Caltech 256 dataset: the samples are converted into PSI descriptors by using the same low-level features, feature mapping, and PCA projection that we have adopted for PiCoDes. We have tried different PCA subspace dimensionality; here we report the results obtained with the setting yielding the best final accuracy. We learn a ITQ descriptor of 2048 bits in order to compare it with PICODES.

- CCA-ITQ. Same as *ITQ* but instead of PCA we use Canonical Correlation Analysis (CCA) (as suggested in [39]), which performs discriminative dimensionality reduction.

- CLASSEMES: the descriptor introduced in [74]. The entries of this feature image vector are the outputs of C classifiers trained to predict the presence or absence of C basis classes, which are manually-specified. Note that the original work of [74] makes use standard kernel SVM (see the appendix A). We re-implemented this work by making use of the approximated feature map described in appendix B, which makes the feature extraction orders of magnitude faster. Moreover, in order to make fair comparisons, we adopted the same low-level features we used to train our descriptors PICODES and MC.

- CLASSEMES-BIT: Binarized version of the descriptor CLASSEMES, obtained by

thresholding the output of the basis classifiers at zero.

- ObjectBank: the descriptor introduced in [52] which encodes into a single vector both semantic and spatial information of objects detected in the input image, using a set of pre-trained detectors.

- IFK Improved Fisher Kernel, as reported in [61].

- LP-β : this denotes the variant of the LP-β [36] multiple-kernel combiner described in the appendix C, based on the same low-level features used by PiCoDes and mc (see Sec. 3.5.2).

Figure 3.6 shows the multi-class recognition accuracy of the different approaches as a function of the number of training examples per class. We can see that the simple LSPM baseline performs very poorly with a linear model; note that incorporating the Chi-square kernel helps a lot but at the cost of the usage of an expensive non-linear classifier. Nevertheless the performance of both the methods is lower then all the other methods. The semantic descriptors classemes and classemes-bit outperform these baselines while having much lower storage cost. Note that the differences in performance between classemes and classemes-bit is negligible on this benchmark, probably indicating that the accuracy of the classifier is saturated at this dimensionality and cannot exploit the additional information provided by the continuous data. ObjectBank performs moderately well but much worse than our descriptors. PiCoDes outperforms classemes, showing the improvements generated by our proposed objective, which explicitly optimizes for linear classification, that is the actual usage of the descriptors in this test. For the same dimensionality and storage cost, PiCoDes outperform also ITQ and CCA-ITQ. The methods Psi and LP-β use multiple features and nonlinear kernels (albeit in approximate form)

and hence their recognition accuracies are among the best in the literature. However, for small numbers of training examples per class, PiCoDes surprisingly matches LP-β . Furthermore, note from Table 3.1 that the storage cost is 2 order of magnitude higher than PiCoDes. The IFK method exploits only SIFT features but it perform very well thanks to a more robust encoding; however this technique produces 1MB-descriptors, which are 4 order of magnitudes larger than PiCoDes, making this approach unsuitable for large-scale scenarios where both storage and computational costs are critical.

Finally, we can see that our descriptors MC and MC-BIT greatly outperform all the other representations; note that the binarized version (MC-BIT) is only 1% worse than the real-valued descriptor (MC) while being 32 times more compact (the storage size for MC-BIT is less than 2KB per image). Moreover we use a simple linear model, which enables efficient training and recognition, and the storage requirement for these descriptors is only a few KBytes per image.

In figure 3.7 we study the accuracy of our descriptors and other binary codes as a function of the number of bits. We tested different representations on Caltech 256, using 10 training examples per class and 25 for testing. The classification model is again a 1-vs-all linear SVM for all methods, with the exception of LP-β.

Note that for our PiCoDes descriptor we could learn many representations, each optimized for a different number of bits, thus setting $C \in \{128, 192, 256, 512, 1024, 2048\}$. Our descriptor was found to be the best compact representation for this task, confirming the findings of our previous evaluation. We observed that for very small descriptor dimensionalities, CCA-ITQ performs slightly better than PiCoDes (e.g., by a margin of 2% for 128 bits). Note however that in our tests the CCA transformation was given the advantage of using the Caltech 256 classes as training data.

Experiments on ILSVRC 2010

Qualitative results The goal of this evaluation is to provide some qualitative understanding regarding which *basis classes* of our MC descriptor are found to be more relevant to describe a few given images. In other words, we want to see which *basis classifiers* fire for some given images. For this experiment, we used a few pictures from the validation set of ILSVRC-2010, and we extracted the MC-1VSALL descriptor, which consist only of the entries corresponding with the 8000 one-vs-the-rest classeme classifiers (the leafs of our label tree). Note that each entry is associated with a real ImageNet category, intuitively measuring how much likely the image contains the an object of that category. Table 3.2 visualizes the ten highest and ten lowest entries among the 8,000 values of the descriptor. This visualization shows that this selection of 8,000 categories generalizes well to novel classes, and the highest scoring classifiers do not need to be necessarily semantically related to the visual category of the test images (fact also noted in [74]).

Categorization We now present results for the multiclass recognition task on the ILSVRC 2010 dataset. Again, we remind the reader that these classes are disjoint from the training categories used to learn the descriptors PICODES, MC, and MC-BIT.

The large size of this database poses new challenges and issues that are not present in smaller databases, as already noted in [18, 65, 53]. Yet, our binary PICODES, and MC-BIT features render the learning on this database relatively fast to accomplish even on a budget PC, as we can represent these descriptors using a single bit per dimension, storing the entire ILSVRC2010 training set in a few GigaBytes of memory. This allows us to use efficient software for batch training [4] of linear SVM: we have had

[4]Note that in this scenario we could have also used a SVM formulation that supports online learning, like the ones proposed in [11] or [1]. These formulations are based on Stochastic Gradient Descent (SGD), and substantially have the great advantace that they need to access one example at a

success with the package LIBLINEAR [29], properly modified to support non-sparse binary input data, multi-threading, negative sampling, different strategies to weight the slack variables. Please refer to section 7.2 for more details.

Again for all the experiments involving our descriptors we use the one-vs-the-rest strategy to perform multiclass classification. To further speed-up the learning we reduce the negative training set by sampling $n = 150$ examples per class. According to a set of preliminary experiments, this sampling causes only a negligible drop in accuracy (less than 1%). Training a single one-vs-the-rest linear SVM using our MC-BIT descriptor takes on average 50 seconds. So the entire multiclass training for ILSVRC2010 can be accomplished in 14 hours on a single-core computer. In practice the learning can be made highly parallel when multiple cores and machines are available. As a comparison, the winning system of the ILSVRC2010 challenge [53] required a week of training with a powerful cluster of machines and specialized hardware.

In table 3.3 we show the results of the analysis we made with our descriptors and other approaches reported in the literature on the ILSVRC2010 dataset.

The MC-LSH method is just a compressed form of MC, obtained using LSH [5, 37] with $200K$ binary random projections, thus reducing the storage for the MC from 60 KB to 25KB per image (see Table 3.1). We apply LSH as follows: we first reduce the dimensionality of the data via PCA, producing a 9000-dimensional descriptor. We center the vector by subtracting the mean (which has been calculated using a validation set), and we apply 200,000 random projections drawn from a Gaussian distribution with 0 mean and unit variance. Figure 3.8 provides an experimental justification for the number of random projections used to generate the MC-LSH descriptor. The

time during the training phase, so avoiding to keep the entire training set in memory. However, they introduce additional hyperparameters, like the learning rate, which makes the learning procedure more complicated and hard to control. We believed that for our use-case a batch method would have been better as simpler to use.

plot shows that 200K dimensions are sufficient to maintain the discriminative power of the original descriptor (the compression causes a negligible drop in accuracy, $< 1\%$ according to our evaluation using the original real-valued MC descriptors on the same ILSVRC 2010 benchmark).

We can notice that the performances of the descriptor MC-LSH is remarkably superior to the binary versions MC-BIT, yielding an improvement +5.44%. This indicates that the real-valued descriptors are more informative than the corresponding binary versions, but this additional expressiveness is exhibited only in large-scale scenarios (again, we remind the reader that on the small Caltech 256 dataset, no significant improvement was obtained using real-valued descriptors). In particular MC-LSH achieves a recognition rate of 42.15% and a top-5 accuracy of 64.01% (top-5 accuracy is the traditional performance measure of ILSVRC 2010). The systems of [65] and [53] are based on very high-dimensional image signatures and linear classifiers. Note that although these approaches provides better accuracy, storage requirements and prediction times are orders of magnitude more costly and clearly inapplicable in our motivational large scale scenarios. The embedding method proposed in [39], and the descriptor CLASSEMES-BIT proposed in [74] produce representations that are comparable in storage size and recognition time to PICODES, but they yield lower accuracy.

Is the meta-class tree useful? In this paragraph, we want to answer the following question in regard or our descriptors MC and MC-BIT: does the meta-class tree provide any advantage over a flat hierarchy composed by only the original classes (i.e. the leaves of our tree)? The following experiments analyze this question from different perspectives, finally showing that there is a benefit in using our tree-based representation.

We first calculated the multiclass recognition on the ILSVRC 2010 dataset achieved with the individual subcomponents of MC-BIT:

- MC-BIT-TREE, which consists of the 7232 meta-class classifiers learned for the inner nodes of the label tree;
- MC-BIT-1VSALL, which contains the outputs of the 8000 one-vs-the-rest classifiers.

MC-BIT-TREE yields 33.87% of Top-1 accuracy, which is clearly superior to MC-BIT-1VSALL that produces only 30.64% of accuracy. More in detail, figure 3.9 shows the results achieved with the individual subcomponents of MC-BIT as a function of the descriptor dimensionality, showing that the grouping of classes performed by the label tree produces features that lead to better generalization on novel classes.

We also applied Recursive Feature Elimination [12] on the *full* descriptor MC-BIT, and in figure 3.10 we show the accuracy as a function of the descriptor dimensionality. Also in this case, the feature selection method chooses more features corresponding to abstract meta-classes (inner nodes) than real object classes (leaves) of the learned tree.

We also investigated which levels of the meta-class tree are mostly used. In figure 3.11 we show the number of retained nodes as a function of the tree dept, when the MC-BIT descriptor dimensionality is $2,000$ (obtained by the same feature elimination procedure). The majority of the selected nodes are in the intermediate levels, indicating that the information provided by the meta-classes is more important than single-class nodes (leaves).

These experiments clearly indicate that the grouping of classes performed by the label tree learning produces features that lead to better generalization on novel classes.

However, the complete MC-BIT descriptor yields even higher accuracy (36.71%), suggesting that there is value in using both subcomponents.

Different types of meta-class trees We note that the work of [16] showed that a hand-constructed semantic hierarchy provides semantic knowledge that can be exploited to define an effective metric for retrieval. Therefore a natural question we asked to ourself is: how do our learned meta-classes compare to the semantic-classes of the ImageNet hierarchy?

For this experiment, we pruned the original ImageNet tree so as to leave only the nodes corresponding to the 8,000 synsets used to train our MC descriptor. This produced a new semantic tree with 1,528 internal nodes. As before, we then trained a binary classifier for each of these semantic classes and obtained a new binary descriptor of 1,528 features. As shown in Fig. 3.12, we found that these "semantic meta-classes" yield an accuracy of 18.37% on ILSVRC 2010. However, a random selection of 1,528 features from our MC-BIT descriptor performs much better, yielding an average accuracy of 21.14% (the average is computed over 10 random selections of 1,528 features). This suggests that meta-class features automatically learned by considering visual relations between classes are more effective than attributes based on human-defined notions of semantic similarity.

Trade-offs between accuracy, speed, and compactness Figure 3.13 shows the trade-off between accuracy and storage for different image signatures. Note that our descriptors are the most compact ones while producing near state-of-the-art classification accuracy.

Figure 3.14 shows the storage-speed envelope of modern image descriptors. The plot shows how our descriptors are among the most compact and fast ones. Note

that the times reported here do not include the feature extraction time, since in our motivating application of object-class search feature extraction is performed during an offline stage, when the search index is created. However, we also point out that the time needed to extract our meta-class descriptor is actually in the same order of magnitude as those of other commonly used features for categorization, including Fisher Vectors [65]. For example, according to our experiments on a few full-scale images taken from PASCAL VOC 2007, the extraction of Fisher Vectors takes roughly 1 second per image if sparse SIFT features are used, but it is about 8 seconds if dense SIFT (single scale at every pixel) are used (as in Perronnin, CVPR 2012 [65]). By comparison, extraction of our complete MC descriptor (i.e., including computation of the low-level features) takes on average about 5 seconds. All timing experiments were done on the same machine using a single core.

Object-class search We present here results for our motivating problem: fast novel-class recognition in a large-scale database. For this experiment we use again the ILSVRC2010 data set. However, this time for each class we learn a binary linear SVM using as positive examples all the images of that class in the ILSVRC2010 training set; as negative examples we use 4995 images obtained by sampling 5 images from each of the other 999 categories. Then we use the classifier to rank the 150,000 images of the test set. We measure performance in terms of mean precision at K, i.e., the average proportion of images of the relevant class in the top-K. Note that for each class, the database contains only 150 positive examples and 149,850 distractors from the other classes. Figure 3.15 shows the accuracy obtained with MC-BIT, which on this task outperforms by 15% CLASSEMES-BIT [74]. We also compared our descriptors to 2048-bit codes learned by ITQ [39] and LSH (we ran these methods on a compressed version of the representation PSI obtained via PCA). It can be seen that, for the same

target dimensionality, all of our methods outperform these baselines.

While the systems described in [65, 53] achieve higher multiclass recognition accuracy than our method on ILSVRC2010 where the classes are predefined (see Sec. 3.5.5), we point out that these approaches are not scalable in the context of real-time object-class search in large databases. Table 3.3 (column three) reports the storage required by different methods for a database containing 10M images. We can see that the methods proposed in [65] and [53] require to store high-dimensional real-valued vectors, making these approaches clearly not scalable in real scenarios.

In addition, our system is also outperforming the other competing methods in terms of recognition time. The last column of Table 3.3 shows the average search time per image for a single object-class query. Our methods are by far the fastest. The system proposed by [65], which was relatively scalable in terms of storage, is the slowest one. Our approach provides a 10-fold or greater speedup over these systems and is the only one computing results in times acceptable for interactive search. We also note that sparse retrieval models and top-k ranking methods [64] could be used with our binary code to achieve further speedups on the problem of class-search.

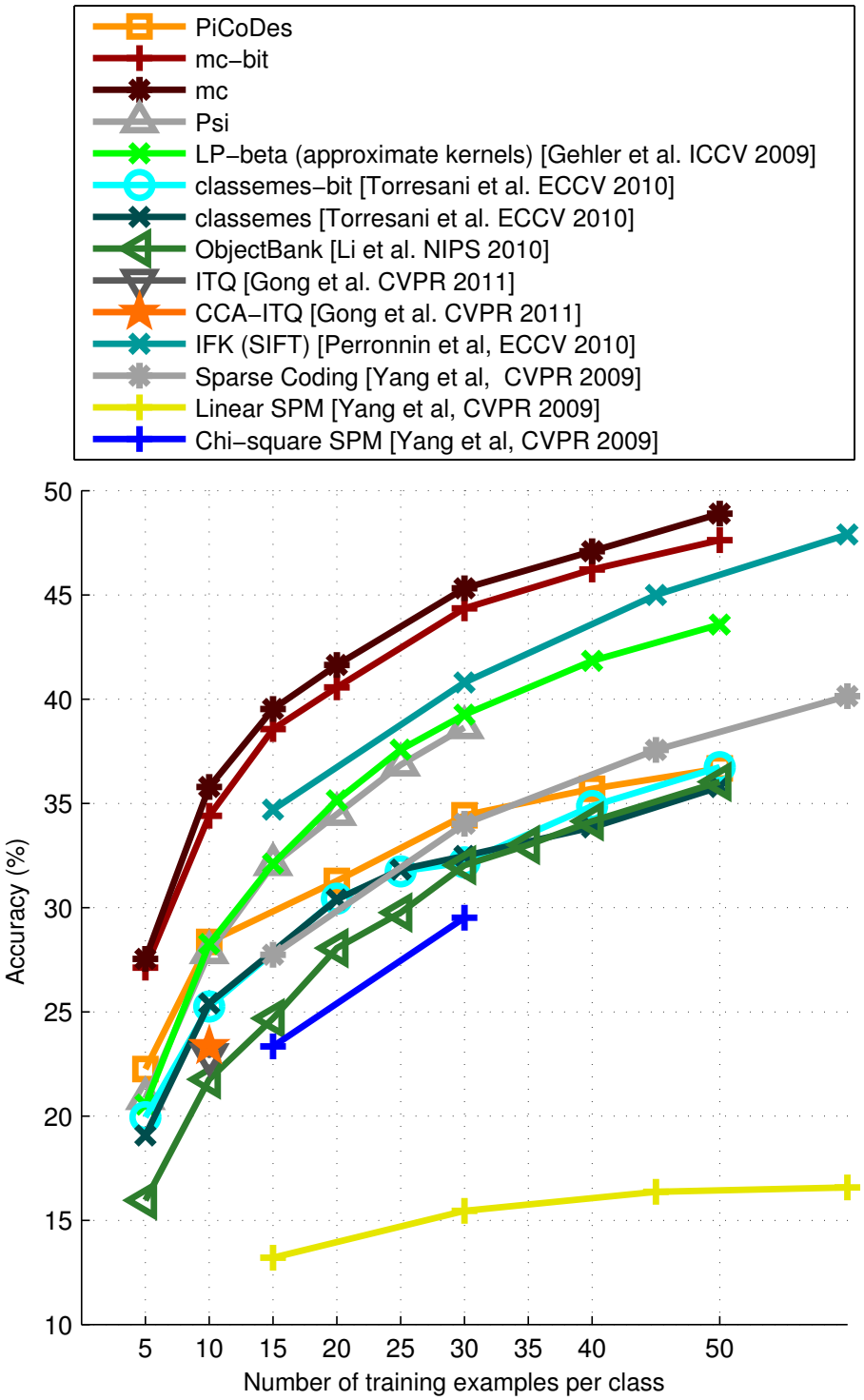


Figure 3.6: Multi-class recognition on Caltech 256 using different image representations. The classification model is a linear SVM (except for LP-β). The accuracy is plotted as a function of the training set size.

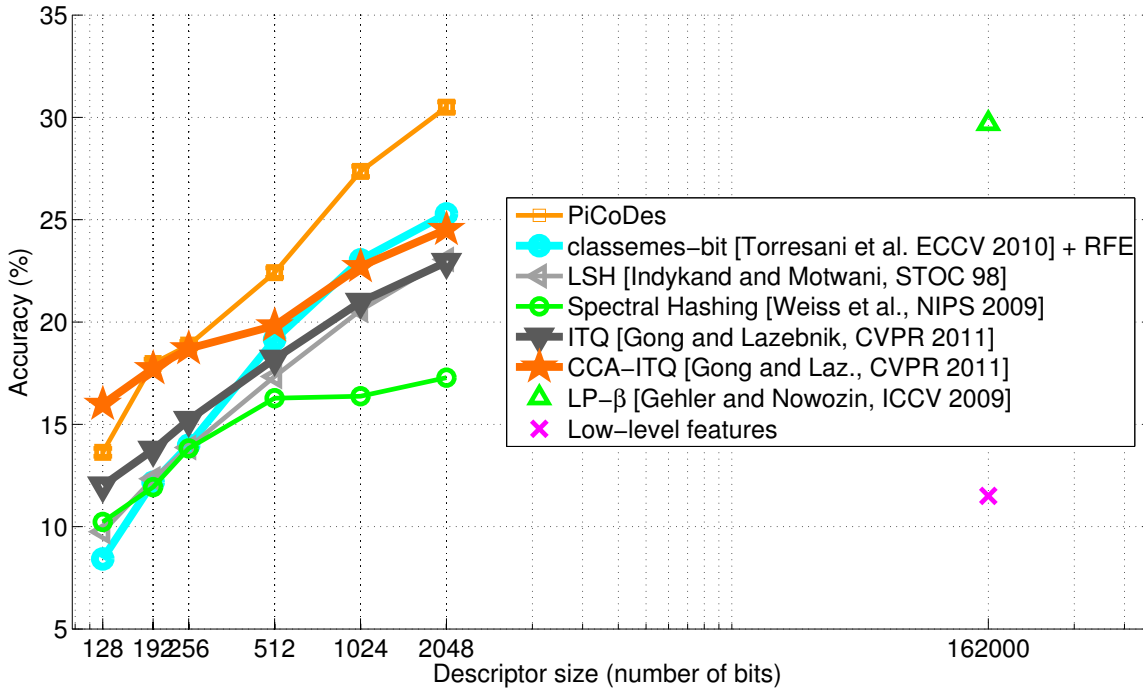


Figure 3.7: Multiclass categorization accuracy on Caltech256 using different binary codes, as a function of the number of bits. PiCoDes outperform all the other compact codes. PiCoDes of 2048 bits match the accuracy of the state-of-the-art LP-β classifier.

Image	**Highest-scoring mc-1vsAll**	**Lowest scoring mc-1vsAll**
	0.99 - wreck 0.99 - beach 0.99 - overcast 0.99 - coral bean tree 0.99 - waterside 0.99 - grey whale 0.99 - gallery 0.99 - shore 0.99 - strand 0.99 - red buckeye	3e-10 - Helvella acetabulum 4e-09 - service club 5e-09 - compressor 6e-09 - stuffed peppers 1e-08 - swamp sunflower 1e-08 - gasoline engine 1e-08 - banquet 2e-08 - spice 2e-08 - traction engine 5e-08 - automobile engine
	0.96 - peregrine falcon 0.96 - double knit 0.95 - snood 0.95 - leatherjacket 0.94 - pillbox 0.94 - cloth cap 0.94 - sweater 0.93 - macrame 0.93 - bolero 0.93 - civilian clothing	2e-3 - van 3e-3 - kaffir boom 3e-3 - Link trainer 3e-3 - hatch 4e-3 - giant petrel 4e-3 - passenger van 4e-3 - press 6e-3 - horsebox 6e-3 - seeder 6e-3 - bubble-jet printer
	0.97 - workroom 0.97 - mustard 0.96 - wine bottle 0.96 - pop bottle 0.96 - brass 0.95 - bier 0.95 - salon 0.95 - musician 0.95 - wine 0.95 - laundry basket	6e-5 - killifish 1e-3 - eagle ray 1e-3 - harrow 1e-3 - duckbill 2e-3 - water flea 2e-3 - ascidian 2e-3 - badminton equipment 2e-3 - tufted puffin 2e-3 - slope 3e-3 - keeshond

Table 3.2: Visualization showing the highest and lowest entries for our MC-1VSALL descriptor, calculated on a few test images. Please refer to the text for further details.

Method	Top-1 accuracy (%)	Storage 10M images (GB)	Recognition time (μs)
PICODES	22.66	2.38	5.14
MC-BIT	36.71	18	38.23
MC-LSH	42.15	232	501.97
Lin et al. 2011 [53]	52.9	43,945	796.60
Sanchez et al. 2011 [65] (FV)	n/a	39,041	603.1
Sanchez et al. 2011 [65] (FV+PQ)	54.3	1,220	2131.0
Gong et al. 2011, [39] (ITQ)	21.24	2.38	5.14
CLASSEMES-BIT	22.15	3.09	6.67
CLASSEMES	29.95	99.05	1.53

Table 3.3: Object class recognition on ILSVRC 2010. For each descriptor, we report: the top-1 accuracy on the test set; the storage size of an hypothetical database consisting of 10 million images; the average time required to evaluate a linear classifier on a single image [3] , without including the feature-extraction time. All the methods use linear SVM as classifier.

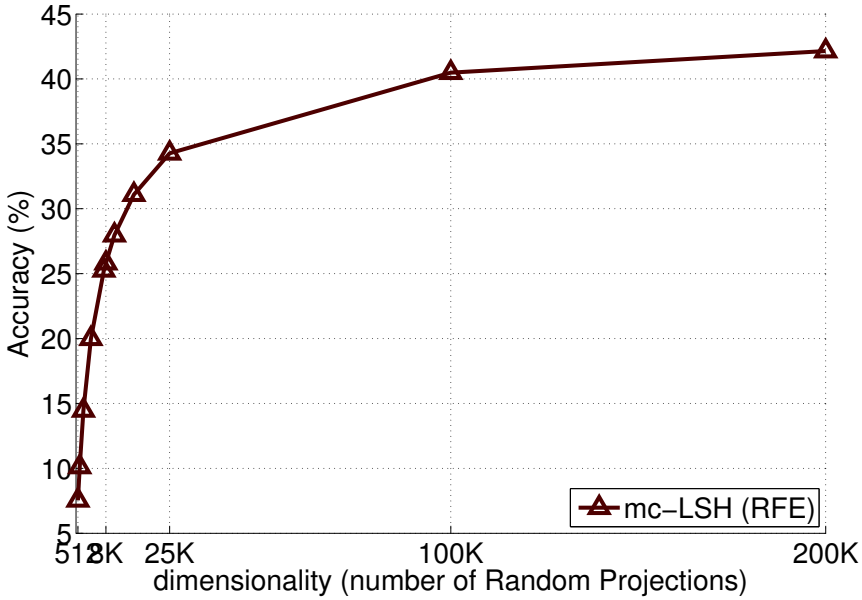


Figure 3.8: Multiclass recognition accuracy on ILSVRC2010 as a function of the number of random projections used to generate the MC-LSH descriptor. We use Recursive Feature Elimination to reduce the dimensionality of the descriptor. The plot shows that 200K dimensions provide a significant compression of the original descriptor, at a negligible drop in accuracy (note that the axes use the log scale). See the text for more details.

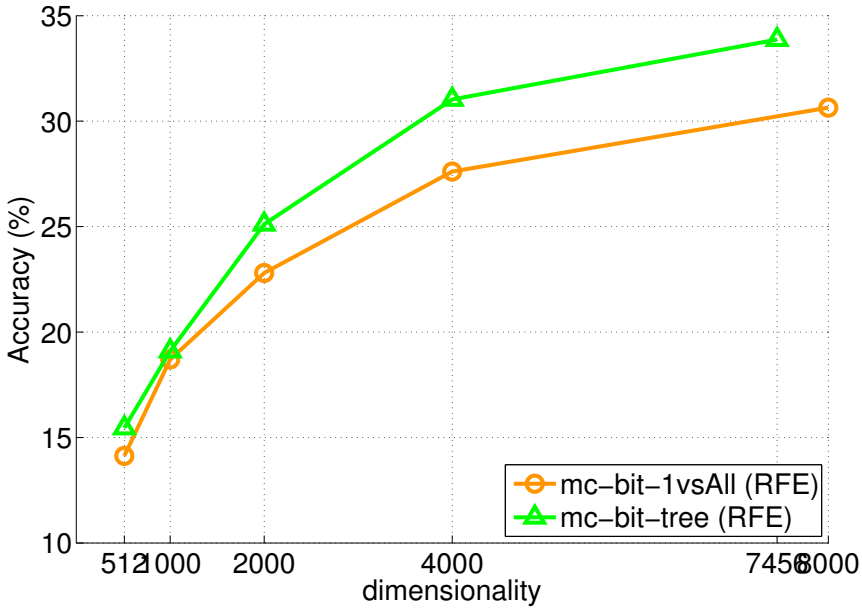


Figure 3.9: Multiclass recognition accuracy on ILSVRC2010 as a function of the dimensionality of the image descriptor. We use Recursive Feature Elimination to reduce the dimensionality of all descriptors. The two curves correspond to the results achieved with the individual subcomponents of MC-BIT: "MC-BIT-TREE", which consists of the 7232 meta-class classifiers learned for the inner nodes of the label tree, and "MC-BIT-1VSALL", which contains the outputs of the 8000 one-vs-the-rest classeme classifiers (the leaves of the tree). Note that the accuracy obtained using only the inner node features is clearly superior to the one generated by only the leaves-features.

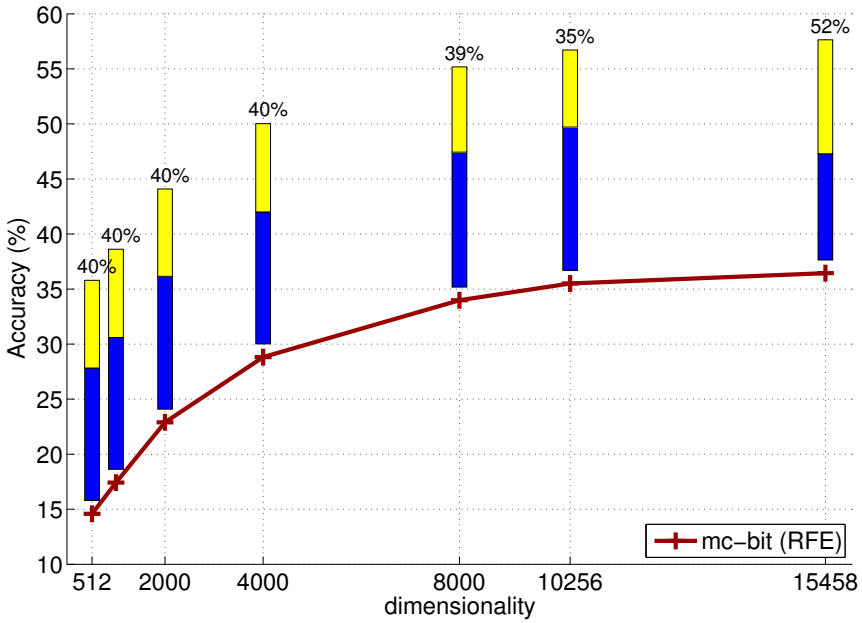


Figure 3.10: Multiclass recognition accuracy as a function of MC-BIT dimensionality on ILSVRC2010. We use Recursive Feature Elimination [12] to reduce the dimensionality of our MC-BIT descriptor. The percentage at each dimensionality indicates the proportion of classeme features retained in the descriptor. Although initially the full descriptor contains more classemes than meta-classes, the majority of features selected at each step are meta-classes.

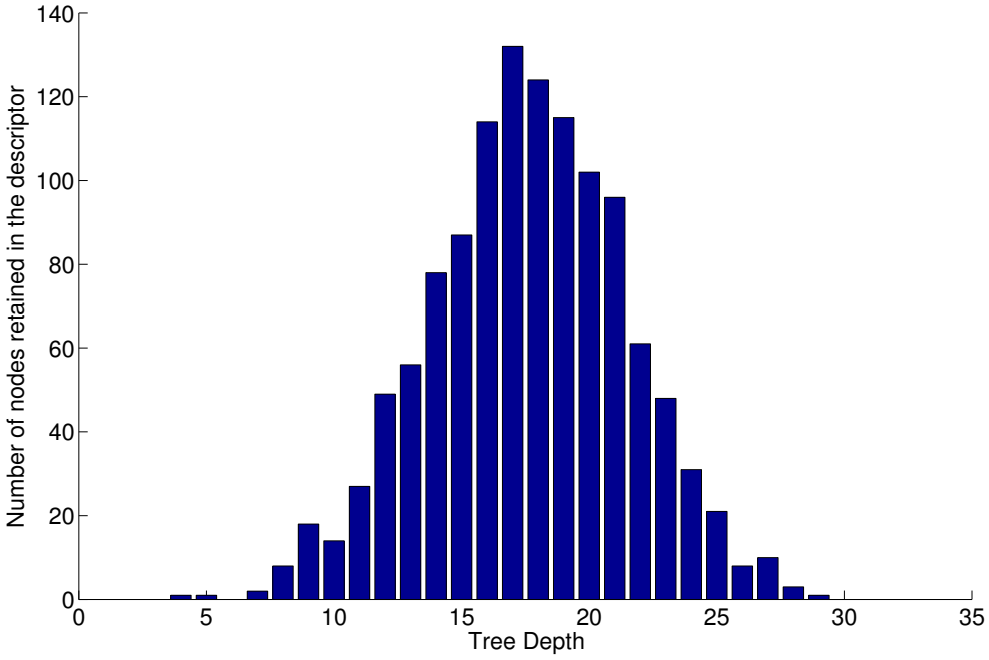


Figure 3.11: Number of retained nodes as a function of the tree dept, when the MC-BIT descriptor dimensionality is $2,000$, obtained by a Recursive Feature Elimination [12] procedure applied to the original vector.

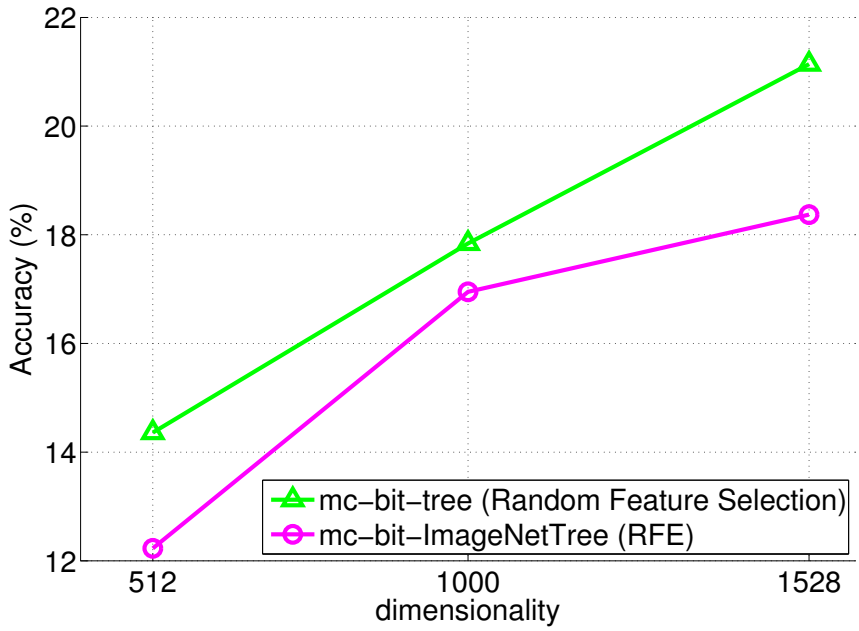


Figure 3.12: Comparison between label trees on ILSVRC2010. The curve for MC-BIT-TREE shows the accuracy achieved with the inner-node features of MC-BIT (the 7232 abstract meta-classes in our learned tree). The MC-BIT-IMAGENETTREE curve corresponds to the inner nodes of the ImageNet semantic tree (we use a pruned version of this tree, containing only the 8,000 synsets that we used to train our MC descriptor). We used Random Feature Selection for the MC-BIT-TREE descriptor, and Recursive Feature Elimination for MC-BIT-IMAGENETTREE. Note that the accuracy obtained with our learned label tree is clearly superior to the one generated using the semantic ImageNet tree (despite we use for this last one a more powerful feature selection algorithm, Recursive Feature Elimination instead of Random Feature Selection).

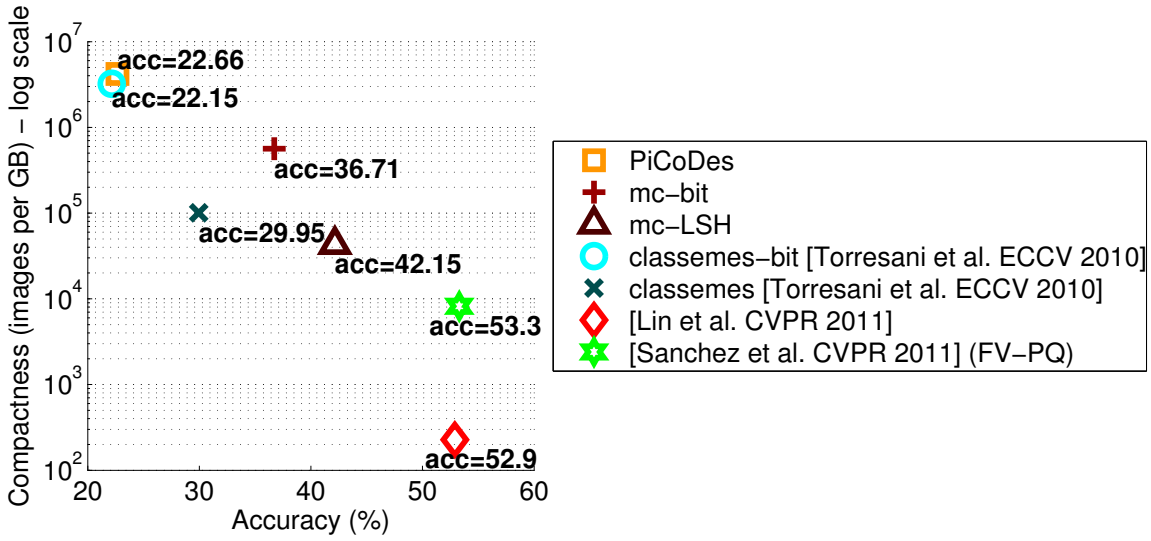


Figure 3.13: Accuracy versus compactness of different image descriptors. On both axes, higher is better (note the logarithmic axes). The number next to each point indicates the multi-class accuracy obtained on the benchmark ILSVRC 2010.

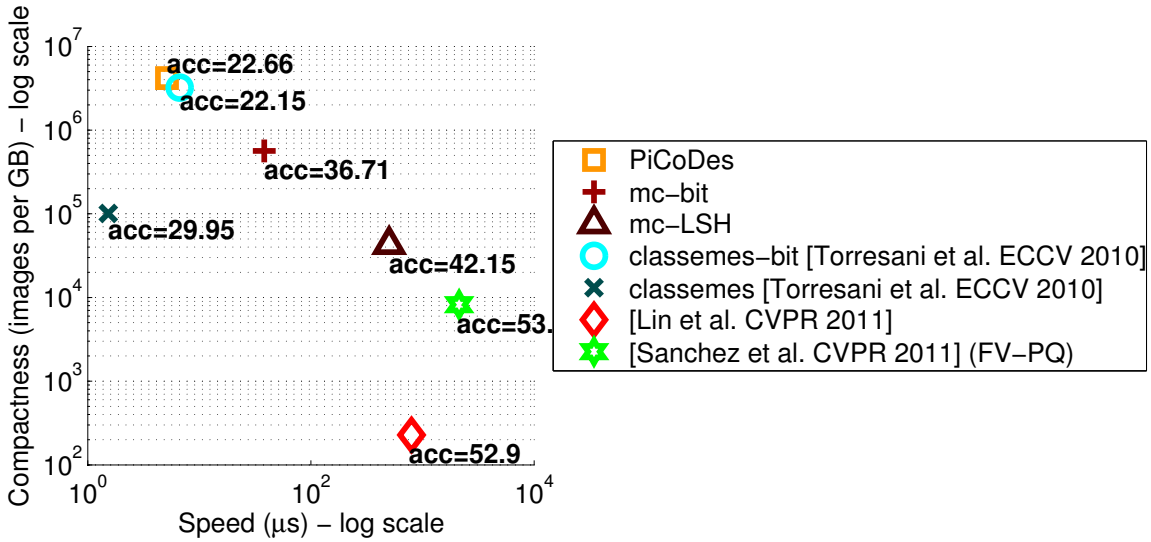


Figure 3.14: Compactness versus recognition time for different image descriptors. Note that the recognition time does not include the feature extraction time (see text). The number next to each point indicates the multi-class accuracy obtained on the benchmark ILSVRC 2010.

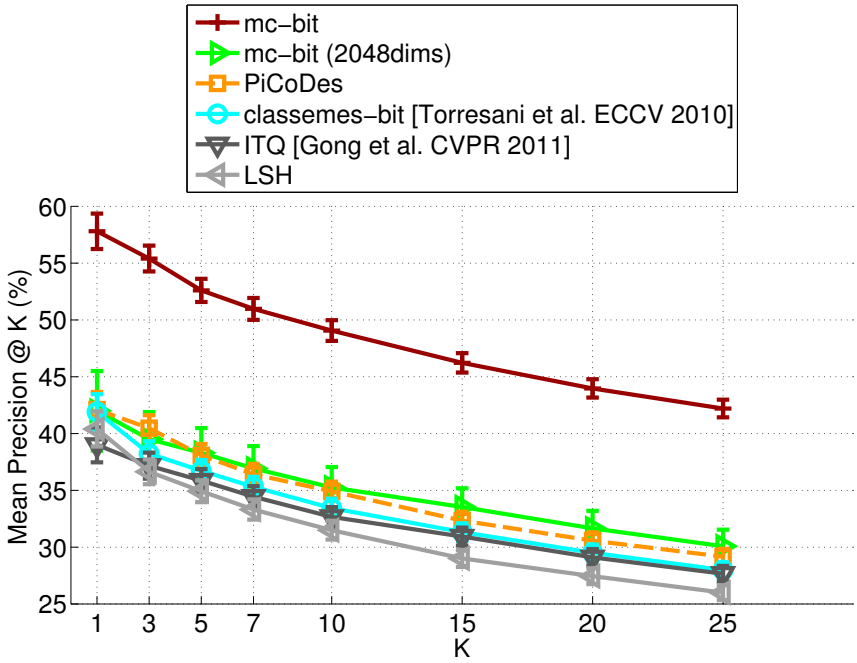


Figure 3.15: Object-class search on ILSVRC-2012: precision in retrieving images of a novel class from a dataset of 150,000 photos. For each query, the true positives are only 0.1% of the database. The classification model is a linear SVM.

Chapter 4

Descriptors for scene recognition

4.1 Introduction

In the previous chapter, our final goal was to perform efficient full-image recognition in large image collections. For this task, there was an inherent assumption that each image contains essentially a single object of interest. This is reflected by most of the standard benchmarks of the fields. For instance, as we can see from figures 3.3 and 3.4, pictures of the Caltech 256 and ILSVRC-2012 are associated with a single label, representing a single type of object. We also observe that in case of Caltech 256, the objects are typically centered and occupy most of the area of the image.

The framework described in Sec. 3.2 describes methods to compute *global* classifier-based descriptors, i.e., feature vectors where each individual entry is the output of a classifier evaluated over the *entire* image. While our experiments show that these representations produce high accuracy on the task of whole-image classification, they are clearly not suitable for recognition when the object of interest occupies only a small region of the photo. In this chapter we describe how to extend our framework to encode local information in the descriptor so as to handle cases when the image

contains small or multiple objects.

We present three different strategies to capture local information. Each of these local encoding methods can be applied to any of the descriptors described in Sec. 3.3, and 3.4. At a high-level, each local encoding method is defined by a splitting function s that decomposes the image into a set of M sub-images, and a combiner m that aggregates the M vectors produced by extracting our classifier-based descriptor $\boldsymbol{h}$ from the individual sub-images. More formally, let $\mathcal{X}$ be the space of all images. Thus, the splitting function $s : \mathcal{X} \rightarrow \mathcal{X}^M$ takes an image $\boldsymbol{x} \in \mathcal{X}$ as input and outputs M rectangular sub-images $\{\boldsymbol{x}^{(m)}\}_{m=1}^{M}$. Then, we define our final global image descriptor $\bar{\boldsymbol{h}}(\boldsymbol{x})$ capturing local information as $\bar{\boldsymbol{h}}(\boldsymbol{x}) = m(\boldsymbol{h}(\boldsymbol{x}^{(1)}), \ldots, \boldsymbol{h}(\boldsymbol{x}^{(M)}))$.

In the next subsections we describe several choices of functions s and m, giving rise to different $\bar{\boldsymbol{h}}(\boldsymbol{x})$.

4.2 Methods

SPCAT: Spatial Pyramid + Concatenation

This is an instantiation of the *spatial pyramid* method proposed in [51] where the function s decomposes the image into a hierarchical partition of rectangular sub-images. The pyramid consists of L layers, where the first layer (layer 0) is the image itself. Each layer spatially subdivides each sub-image of the previous layer into a grid of 2x2 sub-images of equal size. Hence the total number of rectangular sub-images defined by the pyramid is $M = \sum_{l=0}^{L-1} 4^l$.

The combiner m takes as input the set of descriptors computed from the individual sub-images of the pyramid, and concatenates them together:

$$\bar{\boldsymbol{h}}(\boldsymbol{x}) = \left[\boldsymbol{h}(\boldsymbol{x}^{(1)})^\top, \ldots, \boldsymbol{h}(\boldsymbol{x}^{(M)})^\top\right]^\top \ .$$

Figure 4.1: Example images taken from the MIT 67 dataset for the category "Art Studio"

The final descriptor $\bar{\boldsymbol{h}}$ has dimensionality $C \cdot M$, where C is the length of the individual descriptors associated to the regions (as defined in Sec. 3.2).

SPLPOOL: Spatial Pyramid + Layer Pooling

In this strategy the function s still implements a Spatial Pyramid. Instead, the combiner function m now concatenates descriptors obtained by pooling (or aggregating) the feature vectors within each layer.

Figure 4.2: Example images from the dataset SUN 397 for the category "Veterinarians office"

Specifically, let $\boldsymbol{x}^{(l,g)}$ be the g-th sub-image of the l-th layer. In case of *binary* descriptors (PiCoDes, mc-bit, and classemes-bit i.e. the descriptor proposed in [74]) we keep the final features binary by performing max-pooling within each

layer:

$$\bar{\boldsymbol{h}}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}) \\ \max_{g=1,\dots,4} \boldsymbol{h}(\boldsymbol{x}^{(1,g)}) \\ \vdots \\ \max_{g=1,\dots,4^{L-1}} \boldsymbol{h}(\boldsymbol{x}^{(L-1,g)}) \end{bmatrix}$$

where the function max computes the component-wise maximum. In case of *real-valued* descriptors (MC, and CLASSEMES [74]), instead we average the feature values within each layer (thus replacing max with the sample mean of each feature). We also tried max-pooling for the real-valued descriptor but we found empirically that this yields lower accuracy. Note that with this local encoding strategy, the final vector $\bar{\boldsymbol{h}}(\boldsymbol{x})$ has dimensionality $L \cdot C$.

OBJPOOL: Objectness + Pooling

The encoding methods of subsections 4.2 and 4.2 are limited by the fact that the subdivision of the image is fixed, and does not take into account the actual locations of the objects in the photo. Intuitively, we would like the function s to split the image into regions containing objects so as to encode more relevant sub-images

To this end, we propose to implement s as a class-generic object detector producing a candidate set of regions that are likely to contain an object. For this purpose, we define the function s to return the M rectangular sub-images $\{\boldsymbol{x}^{(m)}\}_{m=1}^{M}$ that have the greatest *ObjectNess* measure [4]. Note that that *ObjectNess* method provides also the probability a given subwindow contains a generic object; we tried to exploit this information, by using the subwindows whose have such probability above a certain threshold thus discarding irrelevant regions, but on a preliminary set of experiments it did not provide any significant help.

Then, the combiner m performs average pooling, i.e., computes the average of

each feature entry over all M sub-images. Even in this case we tried max-pooling, but again we found this strategy to produce inferior results compared to average pooling. We append the resulting descriptor to the feature vector computed from the entire image:

$$\bar{\boldsymbol{h}}(\boldsymbol{x}) = \left[\boldsymbol{h}(\boldsymbol{x})^\top, \quad \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{h}(\boldsymbol{x}^{(m)})^\top \right]^\top .$$

Thus, the final dimensionality is in this case $2 \cdot C$.

4.3 Experiments

4.3.1 Datasets

In this work we make use of several datasets, for both learning and testing our descriptors. Here we summarize briefly their characteristics:

- PASCAL VOC 2007 [27]: Benchmark for object detection and classification including 20 object categories and 9,963 images. Despite the small size of this dataset, the classification is very challenging as each image might contain multiple objects whose positions and scales vary greatly. The images have been manually annotated with rectangular regions containing objects. There are a total of 24,640 annotated objects. The database is divided into training, validation and test sets. Folllowing the PASCAL VOC specifications, we assess the quality of our method independently for each class, using *Average Precision* (AP) as measure. We then average the AP of the classes, ending up with a single scalar number, mAP.

- MIT 67 [63]: Benchmark for indoor scene recognition. It consists of 67 indoor scenes (e.g. corridors, bookstores) for a total of 15,620 images, each containing multiple objects. This dataset is challenging as the scenes are characterized by the objects

Method	mAP
PiCoDes	0.437
PiCoDes + SPLPOOL L0L1	0.455
mc-bit	0.527
mc-bit + SPLPOOL L0L1	0.53
mc	0.532
mc + OBJPOOL	0.55
classemes-bit [74]	0.427
classemes-bit [74] + SPLPOOL L0L1	0.452
classemes [74]	0.438
classemes [74] + SPLPOOL L0L1	0.447
Li et al. 2010 [52] (ObjectBank)	0.452
Harzallah et al. 2009 [42]∗	0.635
Song et al. 2011 [71]∗	0.705

Table 4.1: Object-class categorization results obtained on PASCAL-VOC-2007 using our descriptors and other methods in the literature. The performance measure is the mean of the Average Precision. Note that the methods marked with ∗ make additional use of ground truth bounding boxes for training the model. For our descriptors, the classification model is a linear SVM.

contained in the scene. The dataset is split into training and test sets, with 80 and 20 examples per class respectively. Figure 4.1 shows some example images.

- SUN 397 [83]: Large-scale benchmark for indoor/outdoor scene recognition. It contains 397 scene categories for a total number of 108,754 images. The dataset is divided into training and test sets with 50 examples for each class. Figure 4.2 shows some example images.

PASCAL 2007

We now present categorization results on PASCAL 2007. To the best of our knowledge, this is the first comprehensive analysis of the recognition accuracy of classifier-based descriptors on a *detection* dataset, which includes images containing multiple objects whose position and scale varies greatly. For this reason we tested our de-

Method	Accuracy
MC-2048dims	44.6
MC-2048dims + SPCAT L0L1	46.9
MC-2048dims + SPLPOOL L0L1L2	49.6
MC-2048dims + OBJPOOL	49.6
MC + OBJPOOL	55.9
Elfiky et al. 2012 [25]	48.9
Nakayama et al. 2010 [56]	45.5
Pandey et al. 2011 [60]	43.1
Li et al. 2010 [52] (ObjectBank)	37.6

Table 4.2: Scene recognition on MIT 2007 using our descriptors and other methods in the literature. For our image representations, the classification model is a linear SVM. Our descriptor MC +OBJPOOL outperforms all prior methods on this test.

scriptors using the local-encoding extensions described in Sec. 4.2. We implemented SPLPOOL L0L1 using a pyramid of two levels, and the extension OBJPOOL using the 25 subwindows with the greatest ObjectNess score. We tried additional pyramid levels as well as increasing the number of ObjectNess subwindows, but we did not see a significant improvement in accuracy. Table 4.1 summarizes the results in terms of mAP. Despite the simplicity of the linear classification model that we are using, we can see that our descriptors yield good accuracy while enabling extremely efficient prediction. Also for this benchmark MC and MC-BIT are the best performing ones, followed by PiCoDes, CLASSEMES [74] and CLASSEMES-BIT [74] respectively. Moreover note that all the proposed local-encoding strategies boost the accuracies of the raw descriptors. In particular OBJPOOL produces the best results while only doubling the storage cost.

MIT 67

In this section we present experiments performed on the MIT 67 benchmark. As said before this dataset is composed of images containing indoor-scenes, therefore incor-

porating multiple objects. We tested our descriptor MC and the variants described in Sec. 4.2. Specifically: for a first set of experiments, we tested the descriptor MC-2048dims created by selecting from MC the 2048 most active features according to the criterion of Recursive Feature Elimination [12] (RFE). The feature selection was performed using ILSVRC 2010, by removing at each iteration 50% of the features. This lower-dimensional descriptor reduced the computational requirements and allowed us to easily perform an extensive set of evaluations. Table 4.2 summarizes the results of our experiments, and includes the recognition rates of several other methods presented in the literature. We can see that the plain descriptor MC-2048dims is already very competitive, yielding accuracy close to other state-of-the-art methods. All the local encoding variants of Sec. 4.2 are able to boost the accuracy. In particular the method MC-2048dims+SPLPOOL L0L1L2 is superior to MC-2048dims+SPCAT L0L1 while producing a smaller descriptor; MC-2048dims+OBJPOOL is the best method as it produces a descriptor that is only twice as big as the original one and it yields the best recognition accuracy (+2.5% over MC-2048dims). Finally the full-dimensional MC +OBJPOOL yields an accuracy of 56% that, to the best of our knowledge, is the best published result for this benchmark.

SUN 397

To conclude, we present experiments on the large-scale scene recognition benchmark SUN 397. We tested our binary descriptors PICODES and MC-BIT, and the real-valued MC. We compare our representations against the binary descriptor CLASSEMES-BIT introduced in [74]. We use an efficient linear SVM A as usual for all our experiments involving these descriptors.

Table 4.3 shows our results. As already noticed in our prior evaluations, PICODES outperforms CLASSEMES-BIT, and the larger MC-BIT is the best performing

Method	Accuracy
PiCoDes	27.1
MC-BIT	34.8
MC	36.8
CLASSEMES-BIT [74]	17.6
Xiao et al. 2010 [83]	38.00

Table 4.3: Scene recognition on SUN 397 using our descriptors and other methods in the literature. The classification model for our image representations is a linear SVM.

one. The accuracy obtained with MC approaches the results provided by [83] which is a multiple-kernel combiner with 15 types of features, and thus orders of magnitudes more expensive to train and test and requiring much higher storage size.

Chapter 5

Descriptors for object detection

5.1 Introduction

Object recognition is one of the fundamental open challenges of computer vision, which may take two subtly-different forms: *whole-image classification* and *detection*. In the previous Chapter 3 of this thesis we dealt with the problem of *whole-image classification*, where the goal was to categorize a holistic representation of the image. In this chapter, we want to study the problem of *object detection*, which instead aims at detect the presence and localize objects in the image. Object detection provides several other benefits over holistic classification, besides having the ability to localize objects in the picture. It provides robustness to irrelevant visual elements, such as uninformative background, clutter or the presence of other objects.

The problem of object detection has been traditionally approached as the task of exhaustive sub-image recognition [41, 32]: for every category of interest, a classifier is evaluated at every possible rectangular subwindow of the image, thus performing a brute-force sliding window search. In order to maintain the computation manageable despite the large number of subwindows to consider, these approaches are constrained

to use features that are extremely fast to extract. Representative efficient sub-image descriptors include the Histograms of Oriented Gradients (HOG) [14] and the Haar features [80] which can be calculated in constant time at every location by using integral images [80].

Recently, a few authors [2, 75] have introduced the idea of efficiently identifying inside the image the rectangular subwindows that are most likely to contain objects, regardless of their class. Particularly the method of *Selective Search* (SS) originally proposed in [75] shows a recall (fraction of the true objects that are identified by the method) approaching 97% for a small number of candidate subwindows (on average about 1500 per image). This desirable property, coupled with the efficiency of their algorithm, implies that relatively few subwindows need to be considered to accurately localize and recognize objects. In turn, this enables the practical application of sophisticated features and object detection models, which instead would be prohibitive in a traditional sliding-window scenario. For example, the system of [75] achieves state-of-the-art results by training a nonlinear SVM on a spatial pyramid of histograms computed from 3 distinct local appearance descriptors. Despite the complexity of this model, the computational cost of recognition remains low if the classifier is applied only to the 1500 candidate sub-images rather being exhaustively evaluated over all possible subwindows.

In this work we increase further the sophistication of these object detection models by replacing bag-of-word features with higher-level visual concepts learned during an offline training stage. We propose to describe each candidate subwindow in terms of the outputs of a set of pre-trained object detectors. Given a new object class to detect, we simply train a linear classifier on this representation, thus using the pre-learned object detectors as a basis for detection of new categories. This idea is akin to the use of attributes [49, 30, 50] or classifier-based descriptors [82, 74, 16], which

have gained popularity for the task of whole-image recognition. Our contribution is to demonstrate that these representations can be successfully extended to the problem of object detection and enable efficient localization of objects when used in conjunction with methods for selective subwindow search. Our results on the challenging PASCAL-VOC-2007 dataset approach the best published results in the literature. However, we want to point out that our features are learned from image examples taken from an independent dataset (ImageNet) and thus our representation is general and not specifically tuned for the test classes of thePASCAL-VOC-2007 benchmark. Furthermore, our approach involves simple linear classification of mid-level descriptors obtained from low-level bag-of-word features. This conceptual simplicity means that our method is much easier to implement than most currently dominant approaches in detection, such as the high-performing part-based model described in [32], which uses spatially-sensitive pictorial structures to represent objects.

5.2 DetClassemes

5.2.1 Modeling our descriptor

In this section we introduce in detail our framework. The new proposed descriptor is learned from a labeled dataset of images $\mathcal{D} = \{(\boldsymbol{x}_1, y_1, G_1, S_1), ..., (\boldsymbol{x}_N, y_N, G_N, S_N)\}$. Here $\boldsymbol{x}_i$ denotes the i-th image in the database which contain n_i objects of a particular visual category labeled by the scalar $y_i \in \{1, ..., C\}$. The matrix $G_i \in \mathbb{N}^{4\times n_i}$ contains the ground truth bounding box locations associated with the objects, and $S_i \in \mathbb{N}^{4\times s_i}$ contains the location of the top s_i bounding boxes retrieved by the *Selective Search* salient detector introduced by *van de Sande* [75]. We will refer to these subwindows as *SS bounding boxes*.

We indicate with $\boldsymbol{h}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), ..., h_C(\boldsymbol{x})]^\top$ our C-dimensional descriptor associated with the subwindow $\boldsymbol{x}$. Each entry $h_c(\boldsymbol{x})$ is the maximum-pooled output of a set of binary classifiers, each trained to detect a particular visual object by leveraging a specific visual feature. More formally, we first describe a subwindow $\boldsymbol{x}$ with a fixed set of visual low-level features $\{\boldsymbol{f}_m(\boldsymbol{x})\}_{m=1}^M$, which are introduced in section 5.3.1. For each visual category c of the offline dataset $\mathcal{D}$, we then learn a set of classifiers $\{h_c^m\}_{m=1}^M$ each trained on a different feature m. We finally perform max-pooling, hence relying on the feature that provides the highest score: $h_c(\boldsymbol{x}) = \max_{m\in\{1...M\}} h_c^m(\boldsymbol{f}_m(\boldsymbol{x}))\}$. We use an efficient SVM as classifier, using the approximated Intersection Kernel map introduced in [79] and described in the Appendix B to obtain a more robust non-linear boundary. More in detail, we lift-up each feature vector $\boldsymbol{f}_m(\boldsymbol{x})$ in a high-dimensional feature space by mean of the feature map Ψ, and then learn a linear SVM in this augmented space (see the appendix for details).

In addition, we produce a probability score by learning the parameters of a sigmoid squashing function σ by mean of the method proposed in [62]. This scaling acts as a sort of feature regularization and it has been shown to boost the performance in tasks while object-recognition [9] or similarity-search when used with attribute-based descriptors [16]. In summary, each entry $h_c(\boldsymbol{x})$ is an instance of the following efficient model:

$$h_c(\boldsymbol{x}) = \max_{m\in\{1...M\}} \{\sigma_{c,m}(\boldsymbol{w}_{c,m}\Psi(\boldsymbol{f}_m(\boldsymbol{x})) + b_{c,m})\}$$

where $\sigma(z) = 1/(1 + e^{(\alpha z+\beta)})$ is the sigmoidal function, each parameterized by the scalars α and β.

We will refer to our descriptor $\boldsymbol{h}(\boldsymbol{x})$ as DetClassemes.

5.2.2 Learning the descriptor

In this section we provide the details of the training procedure for our descriptor DetClassemes. The hypothesis h_c^m are trained using all the ground-truth subwindows containing the object c as posive examples, and a varying, random subset of candidate subwindows belonging to images not containing the c object as negative examples. Each hypothesis is trained using the procedure described in the appendix D, which involves training the model multiple times while adding hard-negative examples. The negative set of examples used to train the first model is composed by 10 randomly-sampled SS bounding boxes from each positive image, which overlap by 20-50% with the ground-truth positive subwindows. In addition, we randomly sample one SS bounding box for each negative image. We perform a single iteration of hard-negative mining. The hyperparamter of the SVM is chosen from a candidate list by 5-fold-cross validation, by selecting the one that produces the highest cross-validation Average Precision.

Note that this stage of training the parameters for $\boldsymbol{h}$ is done in an offline stage, and does not need to be repeated for a new given problem, as described in the following section.

5.2.3 Utilize our descriptor

After the training of the parameters for our descriptor $\boldsymbol{h}$ is complete, we can now exploit this new representation to tackle and efficiently solve an object localization task. Given a new dataset $\mathcal{D}^{test}$, we extract our low-dimensional descriptors from both the GT and SS subwindows. We then train a very efficient linear SVM using this new representation as input, while leveraging the iterative procedure of hard-negative mining introduced in the appendix D to make the model more robust. We found that

the model successfully converges in 3 iterations. Also here the SVM hyperparameter is chosen from a candidate list by 5-fold-cross validation, by selecting the one that produces the highest cross-validation Average Precision. This particular model selection follows naturally given the task of object detection, where the quality of the scoring value is more important than the accuracy in classifying the subwindows.

Given a test image, the evaluation is performed by first applying the model to all the *SS* subwindows. The scores are then sorted, while greedily removing subwindows that overlap for more than 30% with a subwindow with an higher score. The presence of the object of interested can be inferred by obtaining a boolean value by binarizing the scores according to a specific threshold. However, we vary the threshold to calculate precision-recall curves, as normal practice in an object-detection task.

5.3 Experiments

5.3.1 Implementation of our descriptor

As introduced in section 5.2 we represent an image with a set of low-level features $\{\boldsymbol{f}_m(\boldsymbol{x})\}_{m=1}^{M}$, each of which capturing a different visual clue. More in details we set $M = 3$ and extract the local descriptors SIFT [54], Opponent-SIFT [76] and RGB-SIFT [78], at every pixel of the image. These local visual vectors are finally quantized in a Bag-of-Word vector, and we use a Spatial Pyramid [51] to encode some weak visual spatial information into the descriptor. For SIFT we use a K-means quantizer with 4000 clusters and a 4-level pyramid, whereas for OpponentSIFT and RGBSIFT we use Hierarchical-K-means with 3300 centers and a 3-level pyramid. All the dictionaries have been trained from a random subset of patches (between 1 and 5 million) extracted from the training and validation set of the dataset PASCAL-VOC-

Class	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbik	pers	plant	sheep	sofa	train	tv
Our	28.9	**43.0**	43.0	**16.0**	**18.0**	11.0	**40.0**	45.0	**32.0**	14.0	**25.0**	31.0	**22.0**	31.0	39.0	24.0	12.0	**26.0**	**29.0**	38.0	39.0
[75]	25.2	35.0	39.0	6.0	13.0	5.0	32.0	42.0	29.0	11.0	11.0	31.0	17.0	33.0	44.0	23.0	7.0	23.0	22.0	41.0	39.0
[33]	29.0	32.8	56.8	2.5	16.8	28.5	39.7	51.6	21.3	17.9	18.5	25.9	8.8	49.2	41.2	36.8	14.6	16.2	24.4	39.2	39.1

Table 5.1: Average Precision results obtained with our descriptor and other state-of-the-art object detection pipelines on the benchmark PASCAL-VOC-2007. Our method was able to match the complex DPM model introduced in [32] while being more efficient and conceptually much simpler.

2007. We used the library VLFeat [78] for the feature extraction and quantization. Each low-level feature vector is finally lifted-up using the approximated feature map $\Psi_m : \mathbb{R}^{d_m} \rightarrow \mathbb{R}^{d_m(2r+1)}$ proposed in [79] and explained in the appendix B, in order to approximate an Intersection Kernel. We set $r = 1$ that results in a vector three times bigger. We define the vector Psi made of the concatenation of the three lifted-up BoW descriptors, which yields a 565,000-dimensional vector.

The training set used to train our descriptor is composed by randomly selecting $C = 3380$ categories from Fall-2011 release of the dataset ImageNet [19]. The selected categories contain a variable number of labeled bounding boxes, ranging from 50 to 600 per class. We compose a negative set by randomly selecting $20,280$ images. We finally learn the parameters of our descriptor using the procedure described in section 5.2.2.

Once this step is done, we can extract our descriptor from any subwindows from any given image. The next section will show how employ our feature vector for an object localization task.

5.3.2 Experiments on PASCAL 2007

Object detection In this section we compare our proposed framework with other competing methods on the PASCAL-VOC-2007 benchmark [28], already introduced in section 4.3.1. We remind the reader that this dataset contains twenty visual categories, for a total of 9,963 images. Each image contains multiple objects belonging

to different visual categories, for a total of 24,640 ground truth bounding boxes.

In this experiment the task is to predict from a given test image the presence or absence of an object of a specific category. If the image contains the object, the system must return the location of the bounding box surrounding the object. We consider as true positive a predicted bounding box whose relative overlap with the ground truth is at least 50%. In particular, we use the standard *Average Precision* criteria of PASCAL VOC [28], which also penalizes multiple predictions on the same location.

Table 5.1 shows the results of our evaluations. The *SS+BOW* entry is our implementation of the method introduced in [75]. Note that the authors of [75] reports 0.33 mAP on the same benchmark. A controlling experiment showed that the gap is due to minor differences in the SIFT features used. More in detail, the method introduced in [75] uses a variant of SIFT features introduced in [76], whereas we use the conventional SIFT version available with the package *VlFeat* [78].

This comparison shows that our descriptor outperforms the original low-level representation Psi, while being 167 more compact. Note that in our implementation, the vector Psi is very sparse, due to the fact that many entries are the results of a BoW encoding strategy applied to a sparse set of local descriptors [1]. If we take into account the sparsity, our descriptor is still approximately 20 times more compact. Note that despite the simplicity of our method, we were able to match the *context* version of the complex state-of-the-art DPM model introduced in [32].

Effect of number of categories In this experiment we want to quantify the effect that the number of ImageNet categories have to the final impact of the detection

[1] More in detail, as explained in section 3.5.2, we use the *DoG* interest-point detector [78], which yields less than 1,000 keypoints. Empirically, many of these points are very similar, therefore the BoW quantization creates a very sparse histogram. Finally, the approximated feature map described in the appendix B preserves the sparsity as it maps each zero element to zero.

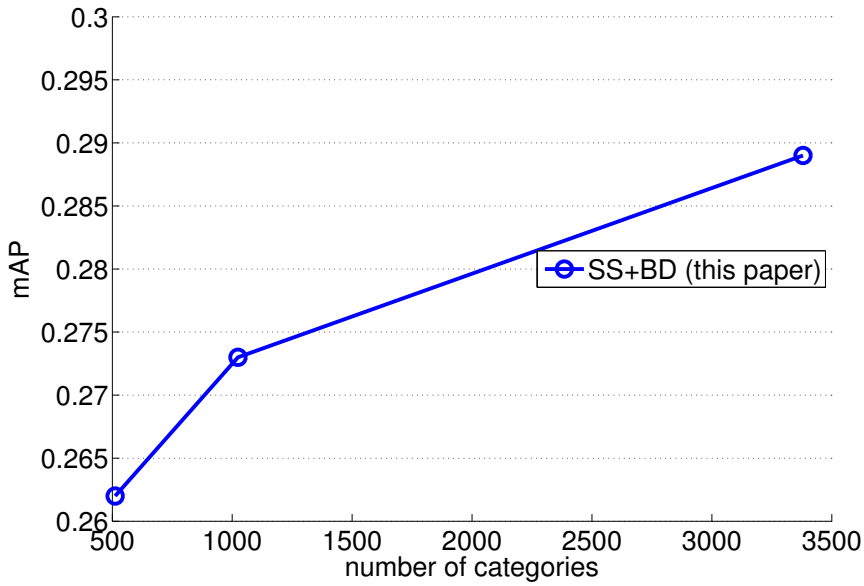


Figure 5.1: mean Average Precision obtained for the object detection task on PASCAL-VOC-2007 with RFE, using our descriptor DetClassemes

system. We start from the 3380-dimensional descriptor and apply Recursive Feature Elimination [12] to remove the associated basis detectors. The results are shown in Figure 5.1. This evaluation suggests that the proposed framework is very sensible to the training set size, as a greater number of training classes corresponds to a increasing in the final performance of the system.

Chapter 6

Weakly-supervised object detection

6.1 Introduction

In this thesis we study the general problem of *image recognition*. In chapters 3 and 4 we dealt with the particular task of *whole-image classification*, where the goal was to categorize a holistic representation of the image. In chapter 5 we considered the task of *object detection*, which instead aims at detect the presence and localize objects in the image. The reader might have noticed that while whole-image classifiers can be trained with image examples labeled merely with class information (e.g. "chair" or "pedestrian"), detectors require richer labels consisting of manual selections specifying the region or the bounding box containing the target object in each individual image example. Unfortunately, such detailed annotations are expensive and time-consuming to acquire. In general, while many efforts have been made to generate annotations using crowdsourcing (e.g. [20] for fine-grained recognition), we are convinced that it is difficult to scale human annotations to millions of categories, given the high-specific knowledge required to understand the visual content of certain fine-grained categories. In summary, we believe that generating rich manual annotations for an

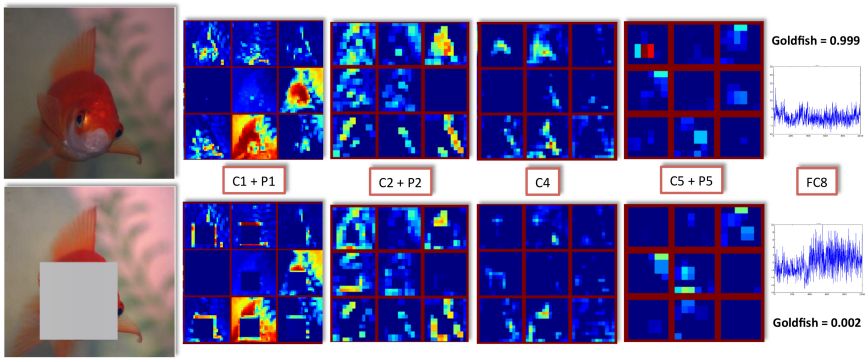


Figure 6.1: The original image (first row), a grayed-out version of the image (second row) and the respective outputs of different layers of the deep convolutional network [48]. For each layer, a 3×3 grid of convolutional kernel responses are shown (C=convolution, P=pooling, FC=fully-connected). The final classification score for the goldfish class is reported at the far right. Better seen in color.

image dataset is a crucial limit of the current state of the art in object localization and detection, which effectively limits the applicability of detectors to scenarios involving only few categories (e.g., [38]). Furthermore, these manual selections are often rather subjective and noisy, and as such they do not provide optimal ground truth regions for training detectors.

Note that this is a limit also for the approach introduced in chapter 5, which requires a large amount of bounding box annotations. Moreover the approach introduced in section 4.2 is partially affected as well, as it uses the ObjectNess measure which made use of bounding box information at training time.

In this chapter, we[1] introduce *self-taught object localization*, a novel approach to localize objects with minimal human supervision by leveraging the output of a whole-image classifier, trained without object location information but only full-image labels.

The key idea is to analyze how the recognition score of the classifier varies as we artificially "gray-out" (i.e. obscure) different regions in the image. Intuitively,

[1]This project has been developed in collaboration with Loris Bazzani

we expect that when the region containing the object is obscured the whole-image classification score will drop significantly. Fig. 6.1 shows how the partial graying out of the input is propagated through the deep network and how this affects the recognition score. We combine this idea with a hierarchical clustering technique that merges regions according to their relative drop in classification score in addition to criteria of spatial vicinity and size of the segments. This produces for each image a set of subwindows that are deemed likely to contain the object.

In this thesis we also investigate the use of these weakly-supervised-generated subwindows as training data to learn a pool of weakly-supervised object detectors, which will focus on recognizing the regions most likely to contain the object, rather than the full image. By doing so, we effectively replace the traditional manually-annotated bounding boxes with regions automatically estimated from training images annotated only with object-class labels. We want to stress that, unlike manual region annotations, category labels are easy to obtain even for a large number of training images. In summary, this framework enables scalable training of object detectors at a much reduced human cost, since no manual annotation of regions is needed. Besides the reduced labeling effort, we demonstrate in our experiments that detectors trained on our subwindow proposals achieve recognition accuracy surprisingly close to that obtained using ground-truth bounding boxes as training data.

A critical aspect of our approach is the choice of whole-image classifier used to bootstrap the training of the detector via self-taught localization. In this work we choose to utilize deep networks since, although they perform holistic image recognition, they have been shown to be impressively accurate even in presence of clutter and multiple objects [48]. In addition, deep networks are particularly suited for our strategy because they operate directly on pixels, which we gray out, in contrast to other models (such as those based on bag of visual words) that discard the spatial

information. Another advantage of deep networks is that the intermediate features learned for whole-image classification can be reutilized as the representation for the subwindow proposals used to train the object detectors.

In section 6.2 we introduce our method to automatically generate bounding-boxes containing the objects of interest, by making use only of the whole-image label. In section 6.3 we present how to use these bounding boxes to train a weakly-supervised object detector. In section 6.4 we will present experiments on the benchmark [17] showing a great improvements in terms of precision and recall with respect to the state of the art in subwindow proposals for object localization. Finally, we demonstrate that the subwindows automatically-generated by our approach can be directly used as positive training examples to learn object detectors without any additional human supervision. Our results on 200 classes of ILSVRC2012 are close to those obtained with detectors trained on manually annotated bounding boxes.

6.2 Self-taught Object Localization

The aim of Self-Taught Localization (in brief STL) is to generate bounding boxes that are very likely to contain objects. The proposed approach relies on the idea of graying out regions of an image provided as input to a deep network (Sec. 6.2.1). The drop in recognition score caused by the graying out is embedded into an agglomerative clustering method which merges regions for object localization (Sec. 6.2.2).

6.2.1 Input Grayout

Let us assume to have a deep network $f : \mathbb{R}^N \mapsto \mathbb{R}^C$ that maps an image $\mathbf{x} \in \mathbb{R}^N$ of N pixels to a vector $\mathbf{y} \in \mathbb{R}^C$ of C classes. The class vector is defined as $\mathbf{y} = [y_1, y_2, \ldots, y_C]$, where y_i corresponds to the classification score of the i-th class.

We propose to gray out the input image $\mathbf{x}$ by replacing the pixel values in a given rectangular region of the image $\mathbf{b} = [b_x, b_y, w, h] \in \mathbb{N}^4$ with the 3-dimensional vector $\mathbf{g}$ (one dimension for each image channel), where b_x and b_y are the x and y coordinates and w and h are the width and height, respectively. The graying vector $\mathbf{g}$ is learned from a training set as the mean value of the individual image channels[2]. We denote the function that grays out the image $\mathbf{x}$ given the region $\mathbf{b}$ using the vector $\mathbf{g}$ as $h_g : \mathbb{R}^N \times \mathbb{N}^4 \mapsto \mathbb{R}^N$. Please note that the output of the function is again an image.

We then define the variation in classification score of the image $\mathbf{x}$ subject to the graying out of a bounding box $\mathbf{b}$ as the output value of function $\delta_f : \mathbb{R}^N \times \mathbb{N}^4 \mapsto \mathbb{R}^C$ defined as

$$\delta_f(\mathbf{x}, \mathbf{b}) = \max(f(\mathbf{x}) - f(h_g(\mathbf{x}, \mathbf{b})), 0) \tag{6.1}$$

where the max and the difference operators are applied component-wise. This function compares the classification scores of the original image to those of the grayed-out image. Intuitively, if the difference for the c-th class is large, it means that the grayed-out region is very discriminative for that class. Therefore the image is likely to contain an object of class c in that position.

In this work, we define two versions of "drop in classification score", depending on the availability of the class label for the image. When the ground truth class label c of $\mathbf{x}$ is provided, we use drop function $d_{\text{CL}} : \mathbb{R}^N \times \mathbb{N}^4 \mapsto \mathbb{R}$ defined as $d_{\text{CL}}(\mathbf{x}, \mathbf{b}) = \delta_f(\mathbf{x}, \mathbf{b})^T \mathbb{I}_c$, where $\mathbb{I}_c \in \mathbb{N}^C$ is an indicator vector with 1 at the c-th position and zeros elsewhere. This drop function enables us to generate *class-specific* window proposals in order to populate a training set with bounding boxes likely to contain instances of class c. We denote the method that uses d_{CL} as STL-CL.

[2]The mean value is a color close to gray, this motivates the name of the procedure, that is called grayout.

If the class information is not available, for example when testing a detector, we use the top C_I classes predicted by the whole-image classifier to define $d_{\mathrm{U}} : \mathbb{R}^N \times \mathbb{N}^4 \mapsto \mathbb{R}$ as $d_{\mathrm{U}}(\mathbf{x}, \mathbf{b}) = \delta_f(\mathbf{x}, \mathbf{b})^T \mathbb{I}_{\mathrm{top}}$, where $\mathbb{I}_{\mathrm{top}} \in \mathbb{N}^C$ is an indicator vector with 1s at the C_I top predictions for the image $\mathbf{x}$ and zeros elsewhere. Since the function is not using the class label, the setup is *unsupervised* and as a consequence *class-agnostic*. The STL method that uses d_{U} is named STL-U.

As deep convolutional network f we use the model introduced in [48] which has been proven to be very effective for image classification. Even though our input grayout idea is general and can be use with arbitrary classifiers, here we evaluate it using deep networks. In this context, the action of replacing regions of the image with the learned mean RGB value corresponds to zeroing out parts of the actual input of the network. Fig. 6.1 shows how the deactivation of the grayed-out input units is propagated through the different layers of the network.

In the next section, we describe how the drop in classification score is used into an agglomerative clustering method to generate window proposals.

6.2.2 Agglomerative Clustering

The initialization point of the proposed agglomerative clustering is a set of K rectangular regions $\{\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_K\}$ generated for an image $\mathbf{x}$ using the image segmentation method proposed in [34]. Our aim is to gray-out regions of the image containing pixels having coherent appearance. Note that in our approach we gray out rectangular bounding boxes enclosing segments rather than the segments themselves. The reason is that if we grayed out the segments, the shape information of the segment would be passed to the network, which could use it for recognition. Instead, we consider the rectangular region enclosing each segment, so that the edges of the segment are

removed from consideration and the drop in classification is more significant.

The goal of the agglomerative clustering is to fuse regions and generate windows that are likely to contain an object. We propose an iterative method that greedily compares the available regions, and merges the two regions that maximize the similarity function discussed below at each iteration (Eq. 6.2). This procedure terminates when only one region is left and it covers the whole image. The similarity score of the regions for an image are then sorted, removing the ones that overlap for more than 50% with a subwindow with an higher score.

We define the concept of similarity between regions using three terms capturing the following intuitions: two bounding boxes are likely to contain parts of the same object if 1) they have a diverse drop in classification score, 2) have similar sizes and 3) are not far apart. These three insights are implemented in the following similarity function for region $\mathbf{b}_i$ and $\mathbf{b}_j$:

$$s(\mathbf{b}_i, \mathbf{b}_j, \mathbf{x}) = \alpha_1(d_k(\mathbf{x}, \mathbf{b_i}) - d_k(\mathbf{x}, \mathbf{b}_j)) + \alpha_2 s_{\text{size}}(\mathbf{b}_i, \mathbf{b_j}, \mathbf{x}) + \alpha_3 s_{\text{fill}}(\mathbf{b}_i, \mathbf{b}_j, \mathbf{x}) \quad (6.2)$$

where the index $k \in \{\text{CL}, \text{U}\}$ in the first term selects STL-CL or STL-U presented in Sec. 6.2.1. The last two terms terms $s_{\text{size}} : \mathbb{N}^4 \times \mathbb{N}^4 \times \mathbb{R}^N \mapsto \mathbb{R}$ and $s_{\text{fill}} : \mathbb{N}^4 \times \mathbb{N}^4 \times \mathbb{R}^N \mapsto \mathbb{R}$ are borrowed from [75] and defined as:

$$s_{\text{size}} = 1 - \frac{\text{size}(\mathbf{b}_i) - \text{size}(\mathbf{b}_j)}{\text{size}(\mathbf{x})} \quad \text{and} \quad s_{\text{fill}} = 1 - \frac{\text{size}(\mathbf{b}_{i,j}) - \text{size}(\mathbf{b}_i) - \text{size}(\mathbf{b}_j)}{\text{size}(\mathbf{x})} \quad (6.3)$$

where $\mathbf{b}_{i,j}$ is the bounding box that contains $\mathbf{b}_i$ and $\mathbf{b}_j$. In practice, s_{size} encourages to merge small regions and s_{fill} favours regions that are close to each other. The overall effect of Eq. 6.2 is that it locally combines regions similar in appearance that

are more and more diverse in terms of drop in classification score.

There are many advantages of the proposed similarity with respect to [75]. First of all, it does not rely on hand-engineered features (like SIFT), because we exploit the features learned by the deep network. Moreover, our similarity exploits the discriminant power of the deep convolutional network enabling our method to generate class-specific window proposals. Yet, our method can work also in the class-agnostic regime like [75] (by using d_{U} instead of d_{CL}).

6.3 Weakly-Supervised Detection using STL

We propose a weakly-supervised detection approach relying on our self-taught localizer. The idea is to exploit the bounding boxes generate by STL (Sec. 6.2) as positive examples when training a pool of object detectors, thus eliminating the need for ground truth annotations. Let us consider the training of an object detector for class c.

Our desired object detector is a function that takes a subwindow $\mathbf{b}_{i,j}$, extracts a meaningful feature representation, and maps it to a score measuring the confidence that the bounding box contains an object of class c. In spirit similar to [38], the feature representation that our detectors are trained on is the last fully-connected layer (before the soft-max) of the deep convolutional network. We refer the reader to [38] for additional details.

Our object detector is iteratively trained using the hard negative mining procedure introduce in [75] and described in details in the appendix D. The initial training set considers as positive examples the $\mathbf{B}_i$ bounding boxes produced by STL-CL on training images of class c (i.e., we use the class label information for localization of positive examples). The negative set is built by selecting the Selective Search [75]

bounding boxes that overlap less than 30% with any $\mathbf{B}_i$ from the positive images, and one randomly-chosen selective search bounding box from each negative image.

At testing time, each detector is tested on the SS subwindows of a given image, the detection scores are sorted and a procedure of Non Maximum Suppression is then applied.

6.4 Experiments

The experiments were performed to assess the contributions of the proposed method for self-taught localization (Sec. 6.4.1) and weakly-supervised detection (Sec. 6.5.1) on challenging datasets making comparisons with state-of-the-art methods.

6.4.1 Self-taught Localization

Given a test image, the task is to propose a set of bounding boxes that enclose the objects of interest with high probability. We consider as True Positive a bounding box whose Intersection-Over-Union overlap with the ground truth is at least 50%, as commonly done for the PASCAL benchmark [27]. The performance is then measured using the *per-class* recall and precision, following the evaluation protocol introduced in [75].

In our experiments, we used the Convolutional Neural Network software *Caffe* [45] for classification, and the model trained on ILSVRC-2012 provided by the authors. Note that the training of the network does not make use of bounding-box annotations.

Given an image, we gray-out the input pixels and perform the agglomerative clustering as described before, outputting a set of bounding boxes likely to contain all the objects of interest. We compare the proposed STL technique introduced in Sec. 6.2 with other competing methods on the following benchmarks:

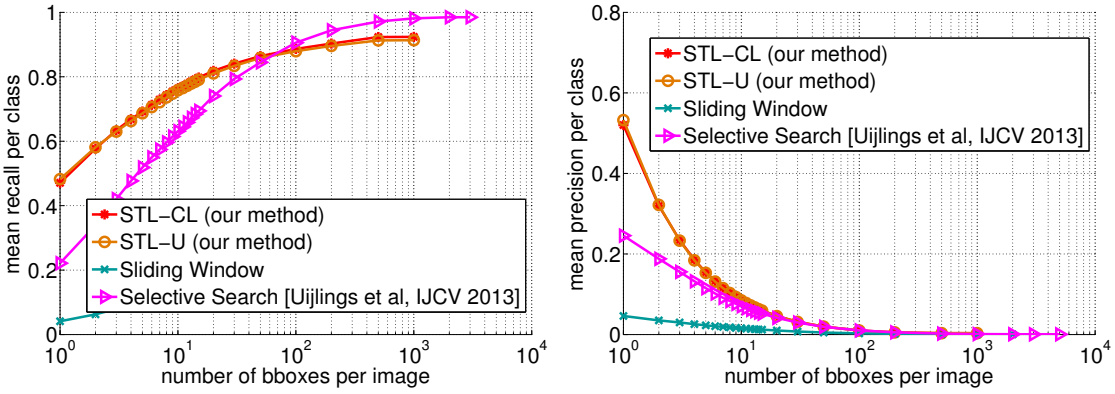
- PASCAL-VOC-2007 [27], which was introduced in section 4.3.1. We remind that reader that it contains twenty visual categories, for a total of 9,963 images divided into training-validation and testing splits. Each image contains multiple objects belonging to different categories at different positions and scales, for a total of 24,640 ground truth bounding boxes.

- ILSVRC-2012-LOC [17], which is a large-scale benchmark for visual object localization containing 1000 categories. Note that this is a subset of the dataset introduced in section 3.5.1, which is the dataset used to train our classification network. The training set contains $544,546$ images with $619,207$ annotated bounding boxes. The validation set contains $50K$ images for a total of $76,750$ annotated bounding boxes. We will name ILSVRC-2012-LOC-200rnd a subset of 200 randomly-chosen categories that will be also used for the weakly-supervised detection task in the next section.

STL is compared against Selective Search [75] (*fast* version) , state-of-the-art class-agnostic bounding boxes proposal methods. One baseline is also the conventional Sliding Window method: a *fixed* set of rectangular windows is slid and evaluated by computing the confidence score as the sum of the classification scores of the top-5 classes predicted by the deep network. The set of subwindows is generated by sliding a square box across the image, using 7 different scales. This produces a number of subwindows that is comparable to the one produced by our method, thus yielding to a similar computational cost. This comparison is important as it shows the performance obtained by using a bounding box *sampling* strategy, as opposed as having a system that *generates* subwindows.

Figure 6.2 (a, b) show the results in terms of recall and precision on ILSVRC-2012-LOC-200rnd (training) and ILSVRC-2012-LOC (validation) set respectively. Note that while ILSVRC-2012-LOC-200rnd (training) includes images that were used as

(a) ILSVRC-2012-LOC-200rnd (training set, 200 classes)



(b) ILSVRC-2012-LOC-200rnd (validation set, 200 classes)

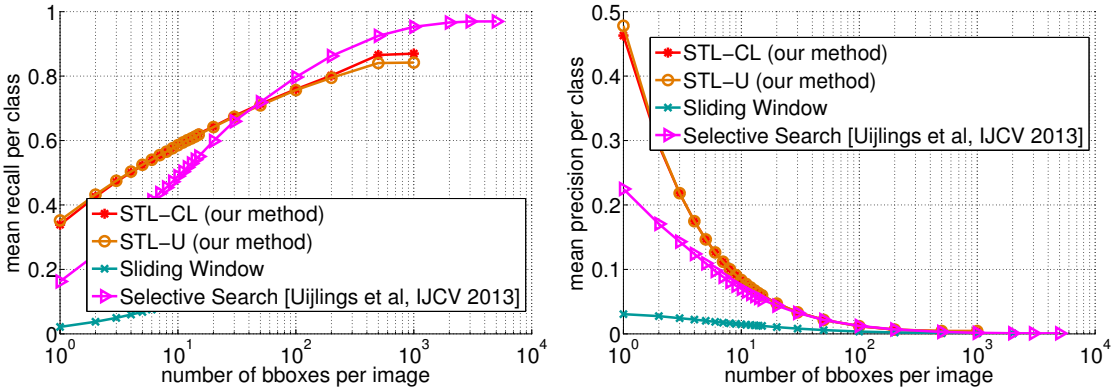


Figure 6.2: Comparison of different bounding-box proposal methods on the benchmark ILSVRC-2012-LOC-200rnd (200 classes). The left column reports the mean recall per class, as a function of the number of proposed subwindows. The right column reports the mean precision per class. The *first row* shows the evaluation for ILSVRC-2012-LOC-200rnd (training), which is part of the data used to train the deep network. Note that both our STL-CL and STL-U outperform all the competitors for the first 100 subwindows. The *second row* contains the evaluations on ILSVRC-2012-LOC (validation), which shows the capability of our methods to generalize to unseen images.

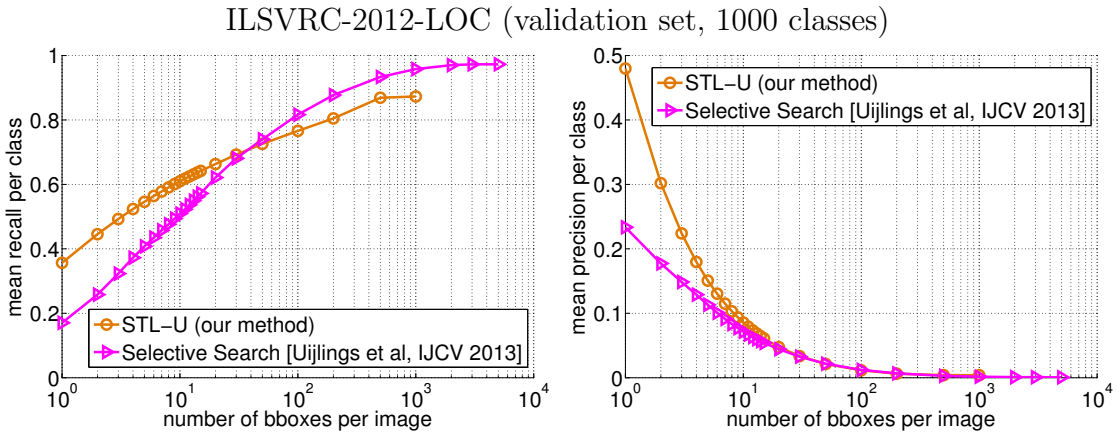


Figure 6.3: Comparison of different bounding-box proposal methods on ILSVRC-2012-LOC-200rnd (validation set, 200 classes). The left column reports the mean recall per class, as a function of the number of proposed subwindows. The right column reports the mean precision per class. The evaluation shows the capability of our methods to generalize on unseen images.

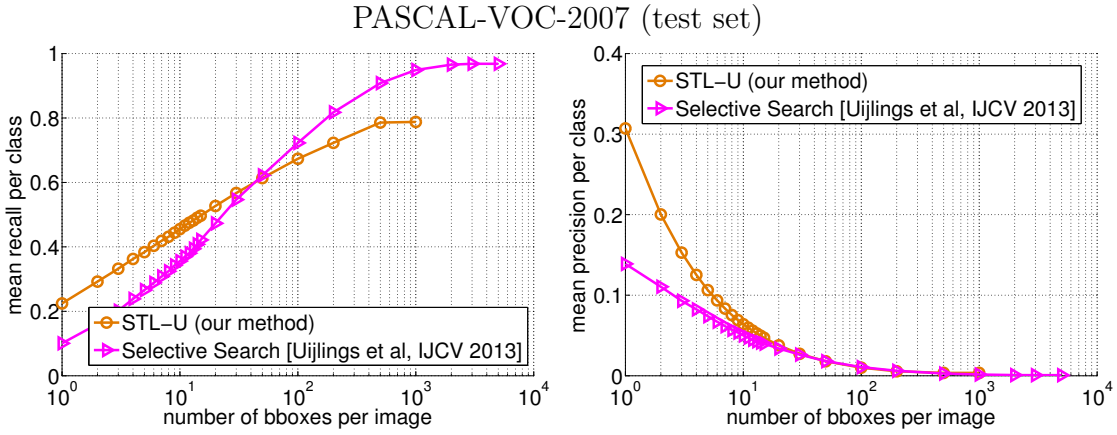


Figure 6.4: Comparison of different bounding-box proposal methods on the benchmark PASCAL-VOC-2007. The left column reports the mean recall per class, as a function of the number of proposed subwindows. The right column reports the mean precision per class. The evaluation shows the capability of our methods to generalize on unseen images.

training examples to learn the deep network, ILSVRC-2012-LOC (validation) does not and thus it is useful to assess the ability of STL to work on new images not seen during the training of the whole-image classifier. We can see that our method significantly outperforms all the other methods for the first 100 proposed subwindows (+33% relative to SS for 5 bounding boxes). The Sliding Window approach performs very poorly (−60% in absolute value for 5 bounding boxes) demonstrating the need for a method that generates bounding boxes of appropriate positions and shapes rather than based on a fixed grid and scale. Note that the performance of using the class label of the image (STL-CL) and not using it (STL-U) is negligible. This indicates that our STL approach works equally well when not given the class label information.

Fig. 6.3 shows performance on ILSVRC-2012-LOC (validation), i.e., on images not used for the training of the deep network. Our methods is the best also in this case, indicating that it naturally generalize to unseen examples. Note that the results on the validation sets of ILSVRC-2012-LOC and ILSVRC-2012-LOC-200rnd are comparable, showing that the random selection of the 200 classes represent well the full dataset.

The methods SS was designed to obtain high recall when using a large number of proposals, which is a necessary property at testing time. However it yields precision not sufficiently high to train a detector. In contrast, our proposed method is designed to produce both high precision and high recall when using a small number of proposals. Although SS shows higher recall when using more than 100 bounding boxes, STL remains competitive even at this regime. Furthermore STL is by far the best in a scenario involving having a small number of proposed subwindows.

We finally show the capability of our method to generalize to unseen datasets and *classes*, using the popular benchmark PASCAL-VOC-2007 [27]. The images of this benchmark have very different statistics than the ones present in ILSVRC-2012-

LOC, as each image can contain objects belonging to different categories. Moreover, we point out that our classification network was neither trained nor fine-tuned on this dataset. Nevertheless, our method is able to generalize to this new scenario, as shown in Fig. 6.4, outperforming again all the compared methods. Finally we notice again that the results of the two proposed methods STL-CL and STL-U are almost identical. This remarkable result shows that STL-U can perform well in the unsupervised setup, as the categories of PASCAL-VOC-2007 are disjoint from those of ILSVRC-2012-LOC.

First of all, we report an additional experiment of the self-taught localization method on ILSVRC-2012-LOC-200rnd. We tested both STL-CL and STL-U against Selective Search and Sliding Window. Figures 6.2 and 6.3 show that that difference in performance between the datasets ILSVRC-2012-LOC-200rnd and ILSVRC-2012-LOC is negligible. This demonstrates that 1) the randomly-selected 200 classes represent in a proper way the difficulty of the 1000-classes dataset and again 2) the proposed method is able to generalize to unseen images.

6.5 Qualitative results

We show some qualitative examples of the effect of the gray-out operation on images in Figure 6.5, 6.6 and 6.7. Each row reports the original image and each of the 5 convolutional layers of the network along with the last fully-connected layer (last column) For the obfuscated image, usually right after the original image, we also show the drop of the recognition score.

In Fig. 6.5, we can see some cases where the proposed method success. In Fig. 6.6, some harder example where the drop is not very significant . Finally, Fig. 6.7 shows the examples where our method is prone to fail (in fact the drop in recognition is not

high).

Moreover, we show in Tables 6.1, 6.2 the highest-scoring bounding box for some images of the dataset ILSVRC-2012-LOC-200rnd (training), for different bounding box proposal methods. We notice how the quality of our STL-CL bounding boxes are better than the SS ones. However, our method does not always work in presence of multiple objects, or when the object gets very confused with the background.

6.5.1 Weakly-supervised Detection

In this section we analyze the effect of training a set of object detectors using the proposed self-taught localization subwindows. To this aim, we trained 200 detectors (one for each class in ILSVRC-2012-LOC-200rnd), each having a training set with 50 positive and $4,975$ negative images (25 examples from each negative class). The test set is composed by $10,000$ images of the ILSVRC-2012-LOC validation set. The detector uses as feature representation the layer *fc7* (the last fully-connected layer) of the deep convolutional network used in [38]. Note that unlike [38], we do not use the context surrounding the bounding boxes (padding).

We carried out three experiments. The proposed approach is trained in the self-taught setup with the top-K bounding boxes generated by STL-CL for the positive images. We set $K = 3$ to obtain a recall/precision trade-off by qualitatively looking at the results in Figure 6.2. We also try training the detector using as positive boxes the top-K windows generated by Selective Search [75] on each positive image. Finally, we report the results of the detectors trained using the ground truth bounding boxes in the fully-supervised setup. This represents an upper bound in terms of accuracy of the self-taught setup. Note that our method exploits the class labels provided during training to generate the bounding boxes, but no ground truth location information is

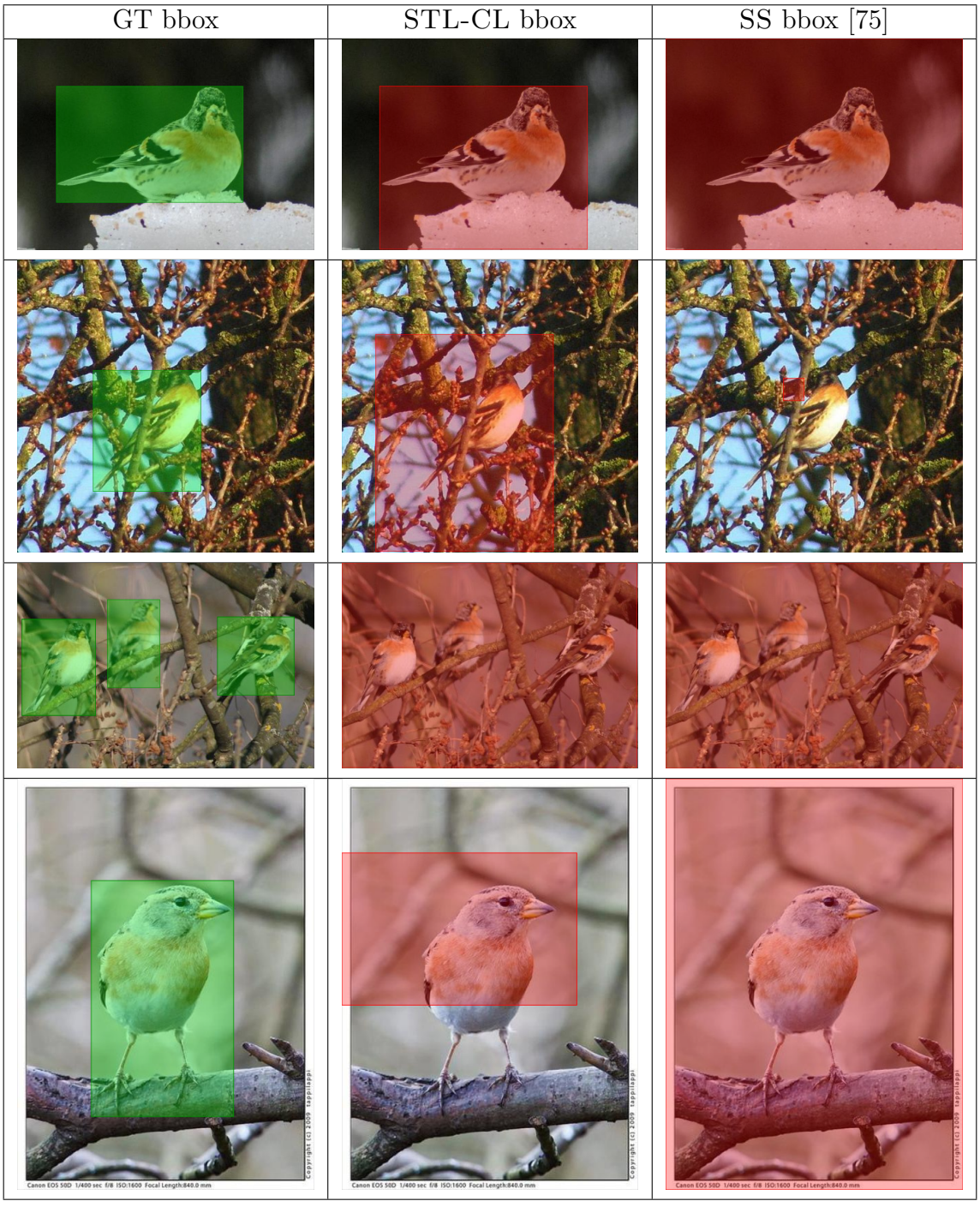


Table 6.1: Highest-scoring bounding boxes generated by different methods for the class n01530575 of the dataset ILSVRC-2012-LOC-200rnd, for a few selected images.

.

Table 6.2: Highest-scoring bounding boxes generated by different methods for the class n01693334 of the dataset ILSVRC-2012-LOC-200rnd, for a few selected images.

.

used in the self-taught setup.

In Fig. 6.8 (left), we report the results in terms of Average Precision (AP, x-axis) for each of the 200 classes (y-axis) comparing the fully-supervised setup with the proposed method. For better visualization, the classes are sorted according to the AP values of the fully-supervised method. For many classes our proposed method achieves accuracy similar to that obtained when using ground truth data. For some classes the drop in average precision is significant, while for other classes STL-CL performs even better the fully supervised method.

Fig. 6.8 (right) shows the difference in average precisions between STL-CL and Selective Search [75] (both used in the weakly-supervised detection setup). The figure shows that our method outperforms Selective Search for more than 120 classes and that for some the difference is significant (greater than 10% in absolute terms). The mAP (mean AP over all classes) is 20.06% for STL-CL and 18.31% for SS.

Table 6.3 shows the best-10 and worst-5 classes for each method along with the mean average precision (mAP) across *all* the 200 classes (last row). We notice that 7 out of 10 best categories are shared between the detectors trained on the ground truth annotations and our STL-CL. This shows that our subwindows do not differ much in nature to the fully-annotated ones. In terms of mAP, STL-CL shows a relative drop in performance of only 21% with respect to the fully supervised method, which is a remarkable result given that our method uses only class labels. Our method yields a relative improvement of 9.5% over Selective Search.

Ground truth boxes	STL-CL	SS [75]
leopard = 65.28	leopard = 67.07	leopard = 59.29
Crock Pot = 62.60	koala = 54.30	car mirror = 50.86
teapot = 59.12	teapot = 53.59	koala = 49.23
admiral = 58.55	car mirror = 51.97	admiral = 44.96
car mirror = 58.28	admiral = 50.07	giant panda = 44.19
koala = 55.63	orangutan = 49.99	crib = 41.21
cabbage butterfly = 54.73	Crock Pot = 48.39	bullfrog = 41.00
frilled lizard = 52.10	pickup = 47.61	maze = 40.67
police van = 51.86	giant panda = 44.18	orangutan = 40.66
giant panda = 51.68	frilled lizard = 44.08	whiskey jug = 38.27
reel = 2.17	*pop bottle = 1.09*	*screwdriver = 0.65*
Windsor tie = 2.16	*swimming trunks = 0.89*	*muzzle = 0.51*
steel drum = 1.43	*nipple = 0.69*	*flute = 0.22*
matchstick = 1.05	*punching bag = 0.37*	*swimming trunks = 0.21*
flagpole = 0.71	*flute = 0.27*	*steel drum = 0.20*
flute = 0.65	*croquet ball = 0.15*	*croquet ball = 0.20*
punching bag = 0.63	*steel drum = 0.10*	*punching bag = 0.12*
hair spray = 0.54	*basketball = 0.07*	*hair spray = 0.03*
screwdriver = 0.41	*hair spray = 0.03*	*basketball = 0.01*
nail = 0.10	*nail = 0.03*	*pole = 0.01*
pole = 0.05	*pole = 0.02*	*nail = 0.01*
mAP = 25.40	mAP = 20.06	mAP = 18.31

Table 6.3: Each column contains the best classes (blue color), and the *worst classes* (red color) for the object-detectors listed at the top, all trained on ILSVRC-2012-LOC-200rnd. Average Precision (%) is provided for each class. The mAP is calculated as the mean across all 200 classes.

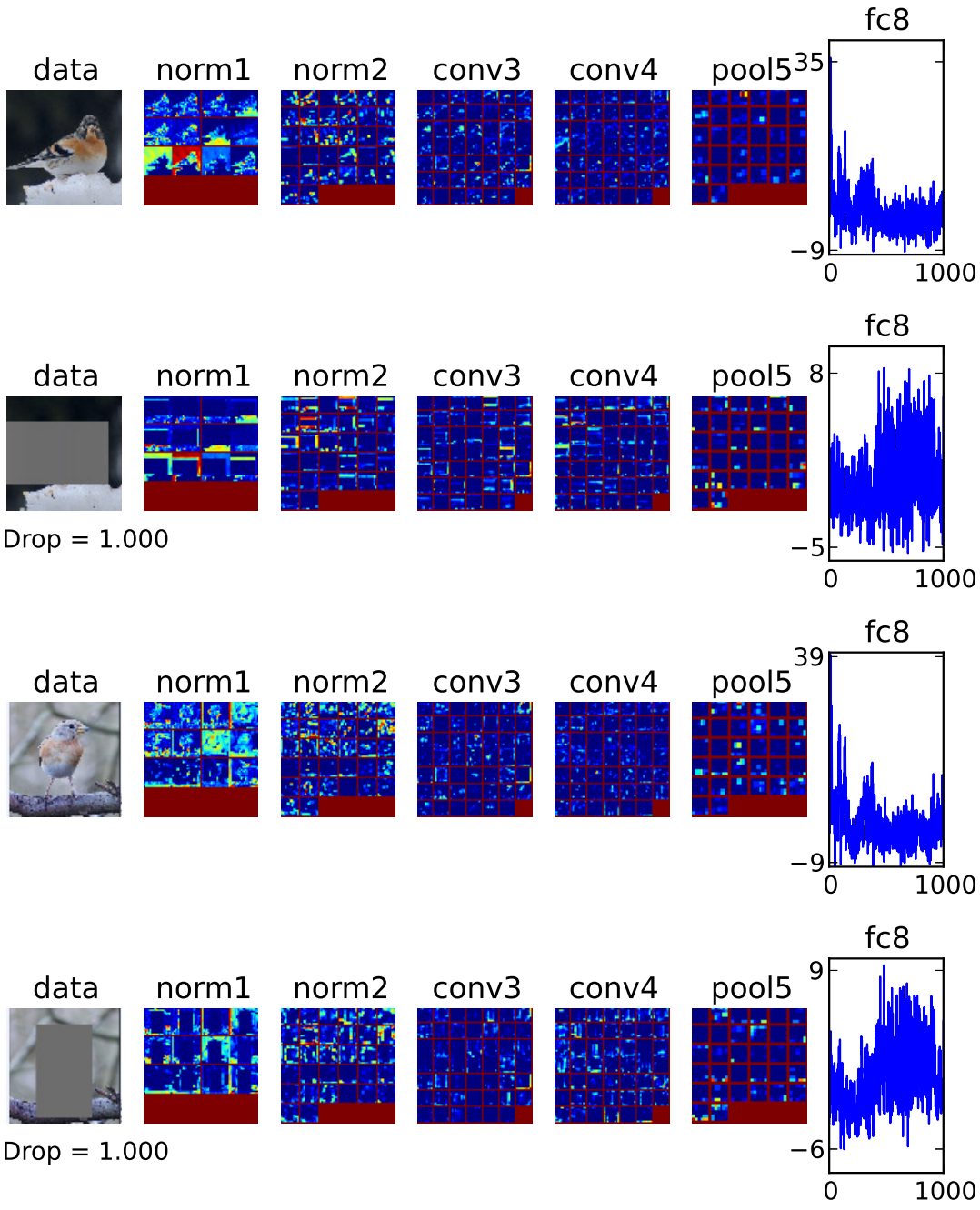


Figure 6.5: Easy successful examples

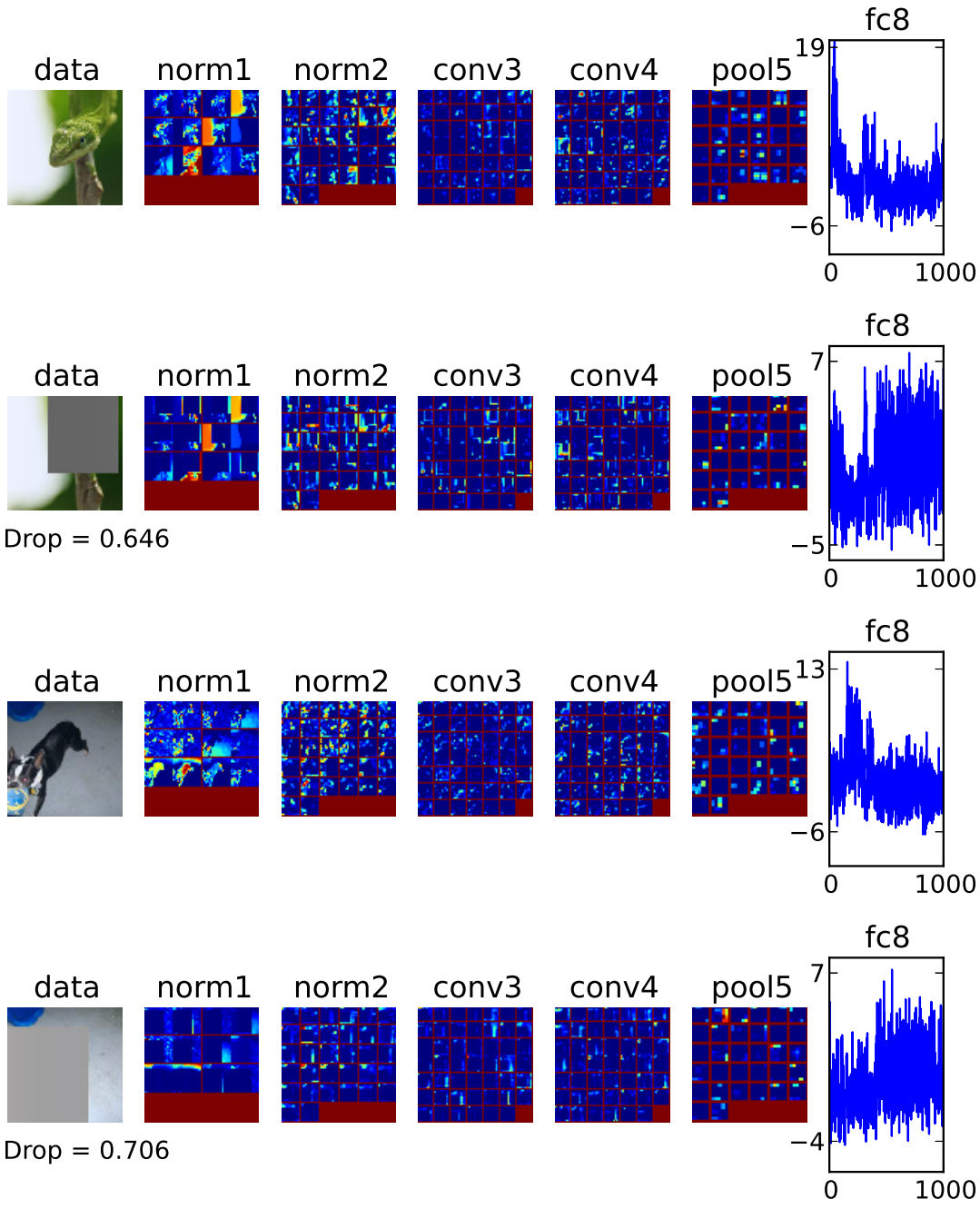


Figure 6.6: Harder successful example

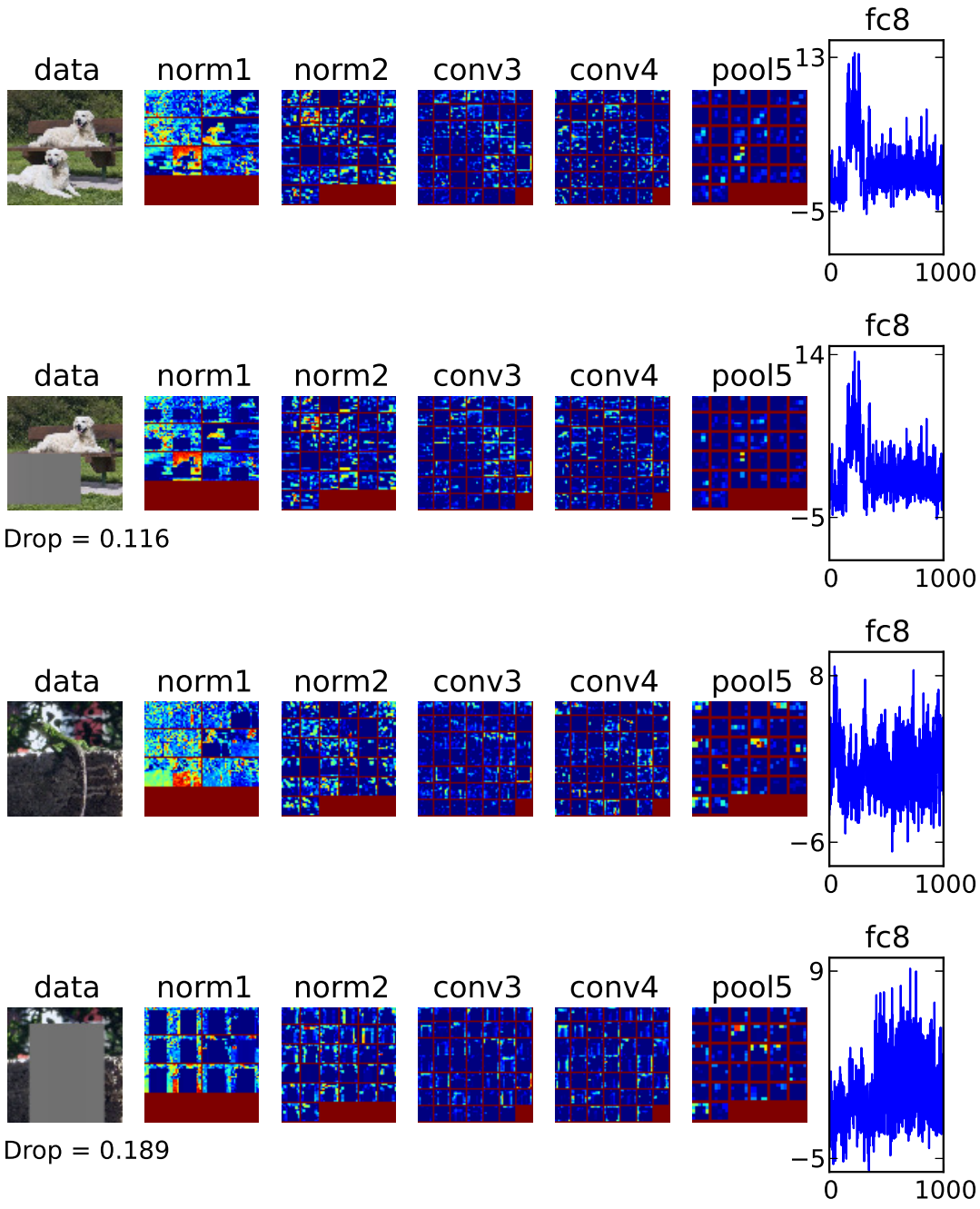


Figure 6.7: Failure examples

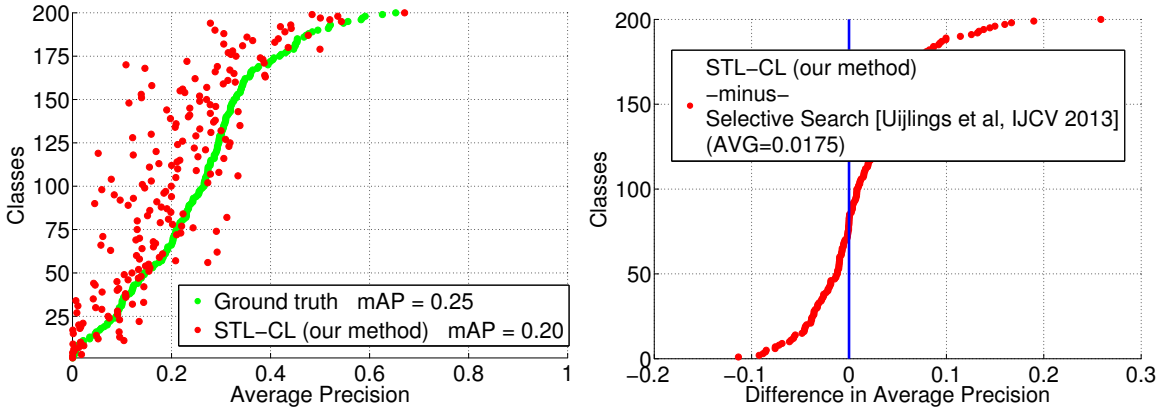


Figure 6.8: Results for weakly-supervised object detection. We trained detectors for the 200 categories of ILSVRC-2012-LOC-200rnd using as positive examples either 1) the ground truth bounding boxes, 2) our STL-CL boxes (automatically generated by making use only of the class labels), or 3) those generated by SS. The left plot shows the average precision of the proposed method STL-CL and the one that uses the ground truth manually-annotated bounding boxes. The right plot reports the difference in average precision for each class between STL-CL and Selective Search [75].

Chapter 7

Software

7.1 vlg_extractor

We[1] wrote a software named VLG_EXTRACTOR, that extracts our image descriptors PiCoDes and Meta-Class introduced in Chapter 3 of this thesis, as well as the third-part descriptor CLASSEMES presented in [74].

The implementation of [74] is order of magnitude more efficient than the one proposed in the original paper, as we make use of approximated feature maps to realize the kernel trick (see the appendix B).

The software supports several images types (Jpeg, Png, Tiff, and others) and it is available for Microsoft Windows, GNU/Linux and Mac OSX. It has been written in C/C++ using GCC, OpenCV for image processing routines, and BLAS for fast matrix calculus.

The software is available at the following link: `http://vlg.cs.dartmouth.edu/projects/vlg_extractor/`

[1] We thank Chen Fang for help in implementing the feature extractions modules described in Sec. 3.5.2

7.2 LIBLINEAR_bitmap

The large-scale experiments presented in section 3.5.5 required non-trivial investigations for a proper training methodology of the linear SVM classifier.

As motivational application, consider the following real use-case. We want to represent using our descriptor MC-BIT the 1.2 million training images of the dataset ILSVRC-2010. To the best of our knowledge, the best state-of-the-art software package for batch linear SVM learning is LIBLINEAR [29], which is incredibly fast and supports sparse data. It can be shown that empirically our MC-BIT descriptor has approximately 30% of non-zero elements. However, if we would have used the software as it is, we would need more than 81 GB of memory[2] , which greatly exceeds that capacity of our hardware. Instead, we notice that if we would have used a dense bitmap to represent the data (1 bit for each value), the storage would be only 2 GB which can be fit in any modern low-budget computer[3] .

Therefore we modified LIBLINEAR [29] to efficiently support the following type of data:

- the original data sparse format, i.e. sparse double-precision floating point data;
- non-sparse binary data, organized as a dense bitmap;
- non-sparse single-precision floating point data, organized as a matrix.

Many other features have been added, such as multi-core supports via PThreads, several 1-vs-all classification models, down-sampling of the training data, polynomial kernel, b-bit Minwise Hashing as the compression method. It has been tested

[2]We have that on a 64-bit machine each non-zero element requires 16 bytes: 8 for the data itself, and 8 for the step. Therefore we have: $(1.2e6 * 15232 * 0.30 * 16)/2^{30} \approx 81.71$ GB

[3]In this case we have: $(1.2e6 * 15232/8.0)/2^{30} \approx 2.12$ GB

on a database with 1.2M of training examples and 100K dimensions on low-budget computers. Written in C/C++ using GCC and PThreads.

`http://www.cs.dartmouth.edu/~aleb/#Code`

Conclusions

In this thesis we studied and made progress in the field of *image recognition*. We have first presented in chapter 3 two novel image descriptors, which measure the closeness of a given image to a set of high-level visual concepts, called basis classes, that are automatically learned. We implement this similarity measure using the outputs of non-linear classifiers trained on an offline labeled dataset. Our descriptors are designed to be very compact, yet they achieve state-of-the art accuracy even with simple linear classifiers. We showed compelling results for tasks of object recognition and novel object-class search. Thank to the compactness and the rich visual information incorporated into the descriptors, the proposed framework enables real-time object-class training and search in databases containing millions of images with good accuracy. The software for the extraction of all our descriptors is publicly available.

In chapter 4 we also extended this framework to aggregate the outputs of the basis classifiers evaluated on subwindows of the image into a single feature vector, thus rendering the descriptor more robust to clutter and multiple objects. We showed that this modified descriptors produce great results for object categorization in images containing multiple objects at different scales and positions, using cheap linear models. We also successfully tackled the task of scene recognition, yielding state-of-the-art results.

We proposed in chapter 5 a new descriptor suitable for the task of object detection,

using the output of a bank of object detectors trained in an offline stage as features to represent the content of bounding boxes. We successfully apply this idea showing improvements on a standard detection benchmark in terms of precision, while lowering the storage requirements.

We pointed out that generating manual bounding box annotations for many training images has been a crucial limit of state-of-the-art methods for object localization and detection. In this regard, we presented in chapter 6 a self-taught localization method that leverages the discriminant power of deep convolutional networks to determine the most likely bounding boxes containing the objects of interest. We tested the proposed methodology on the task of object localization and window proposal using different datasets and showed that it consistently outperforms the current state-of-the-art. We have also shown that detectors trained on localization hypotheses automatically generated by our method achieve performance nearly comparable to those produced when training on manual bounding boxes.

This thesis proposes many different paths to explore for future work. The descriptors proposed in Ch. 3 and 5 are currently limited in using hand-designed low-level features. Moreover, the proposed learning methods are flats, i.e. a single layer of representation is actually learned. These two shortcomings could be tackled by making use of the recently-proposed deep convolutional networks, which elaborate raw pixels while providing an hierarchical representation.

The aggregation methods in Ch. 4 are limited to fixed pyramids and number of Objectness subwindows. In particular the pooling stage could potentially benefit by knowing the likelihood of a subwindow containing an object. Moreover, the combination of the different proposed methods has not been studied, and could bring great improvement in the final performances.

For the methods presented in Ch. 6 there is the need to understand the behavior

of the deep network when it is fine-tuned on new datasets and tasks, since this could potentially bring additional performances.

Finally, we could make use of the method proposed in Ch. 6 to generate training data for the descriptor proposed in Ch. 5, thus learning a powerful descriptor for object detection making use solely of full-image annotations.

Appendix A

Support Vector Machine

Support Vector Machines [77] is a learning framework suitable for many tasks, like classification, as treated in this section. For the fully-supervised binary classification problem, each example $\boldsymbol{x} \in \mathbb{R}^d$ is associated with a label $y \in \{+1, -1\}$ that could be either "positive" or "negative". The goal is to realize a function $g(\boldsymbol{x})$ of the form: $g : \mathbb{R}^d \longrightarrow \mathbb{R}$ which takes an example $\boldsymbol{x}$ as input and outputs a positive score if the example belongs to the positive class, and negative otherwise.

For the *linear SVM* model, the decision function $g(\boldsymbol{x})$ is realized by a simple linear projection of the example $\boldsymbol{x}$ on the hyperplane $\boldsymbol{w}$:

$$g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b \tag{A.1}$$

Note that the addition of the bias term b could be embedded in the linear projection by simply appending a unitary value to the feature vectors $\boldsymbol{x} \leftarrow [\boldsymbol{x}^\top 1]^\top$, and so just realize $g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$.

Given a training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, we want to learn a hyperplane $\boldsymbol{w}$ so that it discriminates well the two classes. More in detail, in the SVM framework the

goal is to minimize the empirical mis-classification error while maximizing the margin, i.e. the distance between the closest correctly-classified examples to the hyperplane. This is achieved by minimizing the following objective function:

$$\min_{\boldsymbol{w},\xi_1,\ldots,\xi_N} \frac{1}{2}\|\boldsymbol{w}\|_2 + C\sum_{i=1}^{N}\xi_i \tag{A.2}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

In the objective above, we have that the variable ξ_i is greater than zero if the example $\boldsymbol{x}_i$ is mis-classified, and zero otherwise. It can be shown that the first term $\|\boldsymbol{w}\|_2$ maximizes the margin (see [77] for more details), while implementing a regularization method, which encourages simple hyperplanes in order to avoid overfitting. The hyperparameter C controls the trade-off of the importance of the regularizer versus the empirical error.

The objective function is convex, and the unique global minimum can be found by optimizing either the primal (eq. A.2), or the dual formulation (see [77]). We note [77] that in the dual objective function, the feature vectors $\{\boldsymbol{x}_i\}_{i=1}^{N}$ appear *only* in form of dot-product with other feature vectors, i.e. $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ for some i, j. This particular form allows the usage of the *kernel trick*, which allows the realization of non-linear decision boundaries. In simplified terms, the kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ realizes the dot-product between two examples in an high-dimensional space (which is never constructed in practice). We can then substitute every dot-product of examples in the SVM dual formulation with a kernel function. In order to obtain a linear model, we use a linear kernel i.e. $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. Non-linear kernels instead, produce non-linear decision boundaries. The decision function for the dual formulation

becomes [77]:

$$g(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{A.3}$$

which consists of a weighted sum over all the training examples, where $\alpha_i \in \mathbb{R}$. We note a great disadvantage of this kernel formulation: the evaluation time is bounded by the number of training examples, whereas the linear model of Eq. A.1 is not.

Appendix B

Approximated feature map

This appendix introduces briefly the method proposed in [79], i.e. *"Efficient Additive Kernels via Explicit Feature Maps", Vedaldi and Zisserman, PAMI 2011.*

The work of [79] presents a method to approximate an homogeneous additive kernel K (see the appendix A for general notions about what is a kernel). Briefly, and in simplified terms, the authors proposed a finite-dimensional function Ψ, which takes a d-dimensional vector as input, and projects it to a higher dimensional space:

$$\Psi : \mathbb{R}^d \longrightarrow \mathbb{R}^{d(2r+1)} \quad r \in \mathbb{N}^0 \tag{B.1}$$

such that the non-linear kernel distance in the original feature space is approximated:

$$K(\boldsymbol{x}, \boldsymbol{y}) \approx \langle \Psi(\boldsymbol{x}), \Psi(\boldsymbol{y}) \rangle \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$$

The great advantage of this method is that it allows to approximate a non-linear model based on an homogeneous kernel, by a simple linear model learned on the top of the features $\Psi(\boldsymbol{x})$. This is particularly useful when dealing in the SVM framework, as a linear SVM is extremely more fast to learn and test than a non-linear one. In

particular at test time given an example $\boldsymbol{x}$, the complexity of a classic non-linear kernel machine $g(\boldsymbol{x})$ is bounded by the number of support vectors, whereas with the approximated feature maps we just have to realize the mapping Ψ and perform a dot-product:

$$g(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) \approx \langle \boldsymbol{w}, \Psi(\boldsymbol{x}) \rangle$$

We note that in this thesis we have always used the Intersection Kernel, which is an additive homogeneous kernel. Moreover, in practice the mapping Ψ is very fast to compute, and produce a very good approximation when $r = 1$ in eq. B.1 (thus producing feature vectors three times as big as the original ones).

Appendix C

Multiple-kernel combiner

This section introduces a variant of the LP-β [36] model, which makes use of approximated feature maps.

The LP-β classifier is defined as a linear combination of M non-linear kernel SVMs, each trained on a different low-level feature vector $\boldsymbol{f}_m(\boldsymbol{x})$. It has been shown to yield state-of-the-art results on several whole-image categorization benchmarks. Note that while [36, 74] employ exact kernels to render the classifier nonlinear, in this thesis we use approximate kernel distances by adopting the explicit feature map described in the Appendix B. This trick allows us to approximate the traditional LP-β classifier, which is computationally very expensive to train, with a linear combination of *linear* classifiers, which can be learned much more efficiently. We found that approximating the kernel distance via explicit maps decreases the overall accuracy of the LP-β classifier by only a few percentage points, but it allows us to speed up the training procedure by several orders of magnitude.

Formally, the LP-β binary classification model $h(\boldsymbol{x})$ is defined as:

$$h(\boldsymbol{x}) = \tau\left(\sum_{m=1}^{M} \beta_m \left[\boldsymbol{w}_m^T \Psi_m(\boldsymbol{f}_m(\boldsymbol{x})) + b_m\right]\right) \tag{C.1}$$

Note that by re-arranging the terms in Eq. C.1, we can expresses h in the same form as Eq. 3.3. Following the customary training scheme of LP-β, we first learn the parameters $\{\boldsymbol{w}_m, b_m\}$ for each feature m independently by training the hypothesis $h_m(\boldsymbol{x}) = \left[\boldsymbol{w}_m^T \Psi_m(\boldsymbol{f}_m(\boldsymbol{x})) + b_m\right]$ using the traditional SVM large-margin objective on a training set. In a second step, we optimize over parameter vector $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_m]^T$ subject to the constraint $\sum_m \beta_m = 1$. This last optimization can be shown to reduce to a simple linear program (see [36] for further details).

Appendix D

Object detector model

This section briefly introduces the object detector model used in the work of [75], i.e. *"Selective search for object recognition", Uijlings, van de Sande et al., IJCV 2013.*

Let us consider the training of an object detector for class c. Let us suppose to have a dataset of M training images $\mathcal{D}_c = \{(\mathbf{x}_i, \mathbf{B}_i, \mathbf{y}_i)\}_{i=1}^{M}$, where $\mathbf{x}_i$ is the i-th image, $\mathbf{B}_i = \{\mathbf{b}_{i,j}\}_{j=1}^{K_i}$ is a set of K_i bounding boxes of the i-th image and $\mathbf{y}_i = \{\mathbf{y}_{i,j}\}_{j=1}^{K_i}$ contains the corresponding binary labels for detection, with $\mathbf{y}_{i,j} = 1$ denoting a box containing an object of class c, while $\mathbf{y}_{i,j} = -1$ indicates a candidate box. The candidate bounding boxes might be, for instance, the ones produced by the method introduced in [75]. We say an image is "positive" if it contains at least a positive bounding box, and "negative" otherwise.

Our desired object detector is a function that takes a subwindow $\mathbf{b}_{i,j}$, extracts a meaningful vectorized feature representation, and maps it to a score measuring the confidence that the bounding box contains an object of class c. This function is implemented with a linear SVM [29], automatically choosing the hyper-parameter from a candidate list by mean of 5-fold-cross validation, selecting the one that produces the highest cross-validation Average Precision. This particular model selection follows

naturally given the task of object detection, where the quality of the scoring value is more important than the accuracy in classifying the subwindows.

The procedure introduced in [75] involves training the model multiple times while performing mining of hard negative examples as technique to reinforce the model. More in detail, we compose an initial training set by considering all the $\{\mathbf{b}_{i,j} \forall i, j | \mathbf{y}_{i,j} = 1\}$ bounding boxes as positive examples. The negative set of examples is built by selecting the candidate bounding boxes that overlap less than 70% (this is a parameter that might change) with any $\mathbf{b}_{i,j}$ from the positive images, and one randomly-chosen candidate bounding box from each negative image.

We then perform a stage of hard-negative data mining, where the initial model is evaluated over all the candidate bounding boxes of the negative images of the training set, augmenting the negative set by adding for each negative image the bounding box with the highest positive score. We finally re-train the model with this augmented training set.

Given a test image, the evaluation is performed by first applying the model to all the candidate subwindows. The scores are then sorted, while greedily removing subwindows that overlap for more than 30% (this parameter might change) with a subwindow with an higher score. The presence or absence of the object of interested could be inferred by binarizing the scores according to a specific threshold.

Bibliography

[1] Akata, Z., Perronnin, F., Harchaoui, Z., and Schmid, C. Good practice in large-scale learning for image classification. *IEEE Trans. Pattern Anal. Mach. Intell. 36*, 3 (2014), 507–520.

[2] Alexe, B., Deselaers, T., and Ferrari, V. What is an object? In *CVPR* (2010), pp. 73–80.

[3] Alexe, B., Deselaers, T., and Ferrari, V. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 11 (2012), 2189–2202.

[4] Alexe, B., Deselaers, T., and Ferrari, V. Measuring the objectness of image windows. *IEEE PAMI 34*, 11 (2012), 2189–2202.

[5] Andoni, A., and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM 51*, 1 (2008), 117–122.

[6] Bengio, S., Weston, J., and Grangier, D. Label embedding trees for large multi-class tasks. In *NIPS* (2010).

[7] Bengio, Y., and Courville, A. Deep learning of representations. In *Handbook on Neural Information Processing*. Springer, 2013, pp. 1–28.

[8] Berg, A., Deng, J., and Fei-Fei, L. Large scale visual recognition challenge, 2010. http://www.image-net.org/challenges/LSVRC/2010/.

[9] Bergamo, A., and Torresani, L. Meta-class features for large-scale object categorization on a budget. *CVPR 0* (2012), 3085–3092.

[10] Bergamo, A., Torresani, L., and Fitzgibbon, A. W. Picodes: Learning a compact code for novel-category recognition. In *NIPS* (2011), pp. 2088–2096.

[11] Bottou, L., and LeCun, Y. Large scale online learning. In *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, Cambridge, MA, 2004.

[12] Chapelle, O., and Keerthi, S. S. Multi-class feature selection with support vector machines. *Proc. Am. Stat. Ass.* (2008).

[13] Cinbis, R. G., Verbeek, J., and Schmid, C. Multi-fold MIL Training for Weakly Supervised Object Localization. In *IEEE Conference on Computer Vision & Pattern Recognition* (2014).

[14] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *CVPR* (2005).

[15] Datta, R., Joshi, D., Li, J., and Wang, J. Z. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys 40*, 2 (2008), 1–60.

[16] Deng, J., Berg, A., and Li, F.-F. Hierarchical semantic indexing for large scale image retrieval. In *CVPR* (2011).

[17] Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., and Li, F.-F. Large scale visual recognition challenge, 2012. http://www.image-net.org/challenges/LSVRC/2012/.

[18] Deng, J., Berg, A. C., Li, K., and Li, F.-F. What does classifying more than 10, 000 image categories tell us? In *ECCV* (2010).

[19] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR* (2009).

[20] Deng, J., Krause, J., and Li, F.-F. Fine-grained crowdsourcing for fine-grained recognition. In *CVPR* (2013), pp. 580–587.

[21] Deng, J., Satheesh, S., Berg, A. C., and Li, F.-F. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS* (2011), pp. 567–575.

[22] Deng, L., Hinton, G., and Kingsbury, B. New types of deep neural network learning for speech recognition and related applications: An overview. In *ICASSP* (2013), pp. 8599–8603.

[23] Deselaers, T., Alexe, B., and Ferrari, V. Weakly supervised localization and learning with generic knowledge. *Int. J. Comput. Vision 100*, 3 (Dec. 2012), 275–293.

[24] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML* (2014).

[25] Elfiky, N. M., Gonzàlez, J., and Roca, F. X. Compact and adaptive spatial pyramids for scene recognition. *Image Vision Comput. 30*, 8 (2012), 492–500.

[26] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. Scalable object detection using deep neural networks. In *IEEE CVPR* (2014).

[27] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[28] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results.

[29] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. Liblinear: A library for large linear classification. *JMLR 9* (2008).

[30] Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. Describing objects by their attributes. In *CVPR* (2009).

[31] Fellbaum, C., Ed. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, illustrated edition ed. The MIT Press, May 1998.

[32] Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 32*, 9 (Sept 2010), 1627–1645.

[33] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (2010), 1627–1645.

[34] Felzenszwalb, P. F., and Huttenlocher, D. P. Efficient graph-based image segmentation. *Int. J. Comput. Vision 59*, 2 (Sept. 2004), 167–181.

[35] Gao, T., and Koller, D. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV* (2011).

[36] Gehler, P., and Nowozin, S. On feature combination for multiclass object classification. In *ICCV* (2009).

[37] Gionis, A., Indyk, P., and Motwani, R. Similarity search in high dimensions via hashing. In *VLDB* (1999), pp. 518–529.

[38] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE CVPR* (2014).

[39] Gong, Y., and Lazebnik, S. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR* (2011), pp. 817–824.

[40] Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Tech. Rep. 7694, California Institute of Technology, 2007.

[41] Harzallah, H., Jurie, F., and Schmid, C. Combining efficient object localization and image classification. In *ICCV* (2009), pp. 237–244.

[42] Harzallah, H., Jurie, F., and Schmid, C. Combining efficient object localization and image classification. In *ICCV* (2009), pp. 237–244.

[43] Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation 18*, 7 (2006), 1527–1554.

[44] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., and Schmid, C. Aggregating local image descriptors into compact codes. *IEEE Trans. on PAMI* (2011).

[45] Jia, Y. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[46] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *CVPR* (2014).

[47] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1106–1114.

[48] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1106–1114.

[49] Kumar, N., Berg, A., Belhumeur, P., and Nayar, S. Attribute and simile classifiers for face verification. In *ICCV* (2009).

[50] Lampert, C. H., Nickisch, H., and Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR* (2009).

[51] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR* (2006).

[52] Li, L., Su, H., Xing, E., and Fei-Fei, L. Object Bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS* (2010).

[53] Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L., and Huang, T. S. Large-scale image classification: Fast feature extraction and svm training. In *CVPR* (2011).

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

[55] Maji, S., and Berg, A. C. Max-margin additive classifiers for detection. In *ICCV* (2009).

[56] Nakayama, H., Harada, T., and Kuniyoshi, Y. Global gaussian approach for scene categorization using information geometry. In *CVPR* (2010), IEEE, pp. 2336–2343.

[57] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *NIPS* (2001).

[58] Nistér, D., and Stewénius, H. Scalable recognition with a vocabulary tree. In *Proc. CVPR* (2006), pp. 2161–2168.

[59] Oliva, A., and Torralba, A. Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research 155* (2006).

[60] Pandey, M., and Lazebnik, S. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV* (2011), pp. 1307–1314.

[61] Perronnin, F., Sánchez, J., and Mensink, T. Improving the fisher kernel for large-scale image classification. In *ECCV (4)* (2010), pp. 143–156.

[62] Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers* (1999), MIT Press.

[63] Quattoni, A., and Torralba, A. Recognizing indoor scenes. In *CVPR* (2009), pp. 413–420.

[64] Rastegari, M., Fang, C., and Torresani, L. Scalable object-class retrieval with approximate and top-k ranking. In *ICCV* (2011).

[65] Sánchez, J., and Perronnin, F. High-dimensional signature compression for large-scale image classification. In *CVPR* (2011), pp. 1665–1672.

[66] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR* (2014).

[67] Shechtman, E., and Irani, M. Matching local self-similarities across images and videos. In *CVPR* (2007).

[68] Shi, Z., Hospedales, T. M., and Xiang, T. Bayesian joint topic modelling for weakly supervised object localisation. In *International Conference on Computer Vision (ICCV)* (2013).

[69] Simonyan, K., Vedaldi, A., and Zisserman, A. Deep fisher networks for large-scale image classification. In *NIPS* (2013), pp. 163–171.

[70] Song, H. O., Girshick, R., Jegelka, S., Mairal, J., Harchaoui, Z., and Darrell, T. One-bit object detection: On learning to localize objects with minimal supervision. In *ICML* (2014).

[71] Song, Z., Chen, Q., Huang, Z., Hua, Y., and Yan, S. Contextualizing object detection and classification. In *CVPR* (2011), pp. 1585–1592.

[72] Szegedy, C., Toshev, A., and Erhan, D. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems* (2013), pp. 2553–2561.

[73] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. DeepFace: Closing the gap to human-level performance in face verification. In *CVPR* (2014).

[74] Torresani, L., Szummer, M., and Fitzgibbon, A. Efficient object category recognition using classemes. In *ECCV* (2010).

[75] Uijlings, J., van de Sande, K., Gevers, T., and Smeulders, A. Selective search for object recognition. *International journal of computer vision 104*, 2 (2013), 154–171.

[76] van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. Evaluation of color descriptors for object and scene recognition. In *CVPR* (2008).

[77] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

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

[79] Vedaldi, A., and Zisserman, A. Efficient additive kernels via explicit feature maps. *PAMI* (2011).

[80] Viola, P. A., and Jones, M. J. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)* (2001), pp. 511–518.

[81] von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing 17*, 4 (Dec. 2007), 395–416.

[82] Wang, G., Hoiem, D., and Forsyth, D. Learning image similarity from flickr using stochastic intersection kernel machines. In *ICCV* (2009).

[83] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR* (2010), pp. 3485–3492.

[84] Yang, J., Yu, K., Gong, Y., and Huang, T. S. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR* (2009), pp. 1794–1801.