Learning with Parsimony for Large Scale Object Detection and Discovery

by

Hyun Oh Song

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2014

Learning with Parsimony for Large Scale Object Detection and Discovery

Abstract

Learning with Parsimony for Large Scale Object Detection and Discovery

by

Hyun Oh Song

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Approximately 85% of internet traffic is estimated to be visual data. Conventional object detection algorithms are not yet suitable to harness this unconstrained, massive visual data because they require laborious bounding box annotations for training and large scale inference is infeasibly slow due to model complexity. In this thesis, I present two instantiations of model parsimony for large scale object detection and discovery. For model inference, I present sparselet models which significantly reduce model inference complexity by utilizing a shared representation, reconstruction sparsity, and parallelism to enable real-time multiclass object detection with deformable part models at 5Hz with almost no decrease in task performance. For model learning, I present a framework for training object detectors using only one-bit image level annotations of object presence without any instance level annotations (i.e. bounding boxes). This framework provides approximately 50% relative improvement in localization accuracy (as measured by average precision) over the current state of the art weakly supervised learning methods on standard benchmark datasets.

To my parents.

Contents

List of Figures

List of Tables

Acknowledgments

I am grateful for all the wonderful experience and the people I met in Berkeley. First, I would like to thank Trevor Darrell for being a great advisor, mentor, and leader. Trevor encouraged me to identify the big picture and be an independent researcher. I am grateful to my thesis committee members, Jitendra Malik, Bruno Olshausen, and Alyosha Efros for the feedback and guidance. I would also like to thank and acknowledge that my Ph.D. research have been greatly influenced by postdoctoral scholars Ross Girshick and Stefanie Jegelka. I am also grateful to Pedro Felzenszwalb for sharing his deep insights through Skype meetings.

Thanks Michael Jordan, Martin Wainwright, Richard Karp, Alex Smola, and Dan Klein for the greatest classes I've ever taken.

Thanks to my collaborators Tim Althoff, Mario Fritz, Christopher Geyer, Chunhui Gu, Zaid Harchaoui, Yong Jae Lee, Julien Mairal, and Stefan Zickler.

Thanks to fellow SDH denizens and group members, Pablo Arbelaez, Jon Barron, Jeff Donahue, Panna Felsen, Georgia Gkioxari, Dave Golland, Chunhui Gu, Sergio Guadarrama, Saurabh Gupta, Bharath Hariharan, Lisa Ann Hendricks, Judy Hoffman, Yangqing Jia, Allie Janoch, Abhishek Kar, Sergey Karayev, Jon Long, Trevor Owens, Evan Shelhamer, Eric Tzeng, Oriol Vinyals, Ning Zhang.

Finally, special thanks to my family for all the love and support.

My Ph.D. research has been generously supported by Samsung Scholarship Foundation.

Chapter 1

Introduction

One of the central problems in artificial intelligence is giving machines the ability to understand and analyze visual input data. Although humans can effortlessly solve the task of understanding the scene and describing what objects are where, the task performance still remains unsatisfactory for machines. After several AI winters[99] have passed, the research community recently have shown encouraging progress due to ever growing quantity of visual data and improving computing performance. However, many existing algorithms still suffer from scalability with the size of the data and rely on laborious human supervision. This thesis proposes a step towards learning and making inferences for large scale object detection and discovery with *parsimony*. First instantiation of parsimony proposes a compressed discriminative model learned with inference complexity parsimony for computational efficiency. Second instantiation of parsimony addresses the question of how to train models with human supervision parsimony.

Domains of modest complexity typically have hundreds to thousands of categories, and as one considers unconstrained search problems, the space of possible categories becomes practically unlimited. In this regard, conventional recognition algorithms become infeasibly slow due to model complexity. This thesis first shows an efficient discriminative machinery which significantly reduces the inference complexity by compressing the model structure with almost no loss in accuracy while providing speed ups as well as easy to analyze guarantees.

Furthermore, the classical paradigm for learning object recognition models is to have human annotators label each object instance in all training images with a bounding box. However, this exhaustive labeling is extremely costly and is not scalable with the size of the data. Furthermore, the increasing prominence of sparsely and noisily labeled data nurtures a growing demand for learning models that can cope with a minimal amount of supervision. The massive amount of textually tagged visual data (for example, images queried from web image search as in Figure 1.1) inspires a challenging research problem of whether we can still train object detection models with image level object presence/absence labels only. Concretely, the goal is to use weakly labelled images to learn to localize the target image in previously

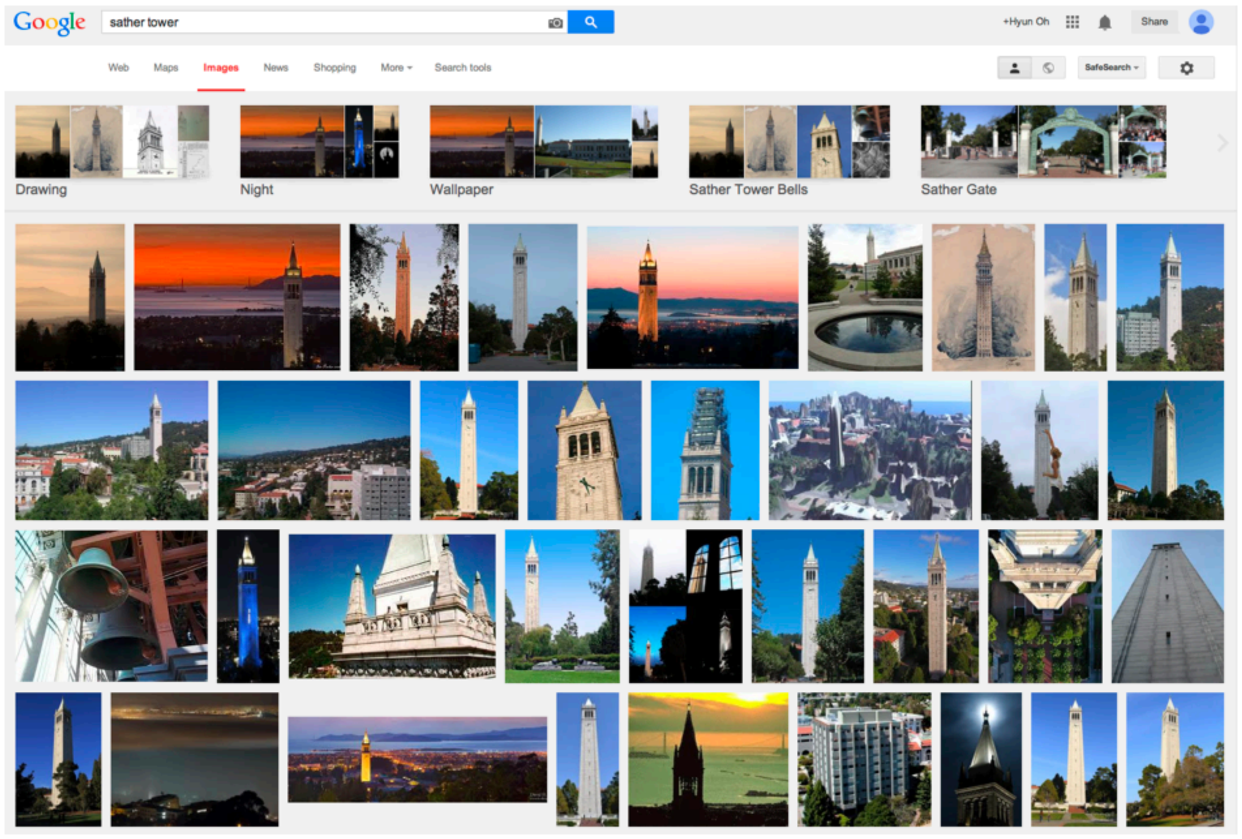

Figure 1.1: Google image search results with keyword "sather tower". Majority of the retrieved images contain the tower *somewhere* in the image.

unseen test images. Recent work[97, 33, 72, 82, 87] has explored learning methods that decreasingly rely on strong supervision. This thesis also proposes a model that can learn from weakly labeled images (human supervision parsimony) with theoretical performance guarantees.

1.1 Outline

Chapter 2 provides a self contained introduction to machine learning algorithms this thesis builds on and also a brief overview of object detection. This chapter is aimed to introduce basic concepts and intuition particularly on dictionary learning and sparse coding, submodularity, and constrained submodular maximization.

Chapter 3 addresses the question about how to learn *compressed discriminative models* for efficient for multi-class, multi-convolutional inference. This chapter describes a framework that simultaneously utilizes shared representation, reconstruction sparsity, and parallelism to enable real-time multiclass object detection with deformable part models at 5Hz on a laptop computer with almost no decrease in task performance. The framework is trained in standard structured output prediction formulation and is generically applicable for speeding up object recognition systems where the computational bottleneck is in multiclass, multi-convolutional inference. The experiments demonstrate the efficiency and task performance of our method on PASCAL VOC 2007, subset of ImageNet, Caltech101 and Caltech256 dataset. The source code for the sparselet model is available on Github at https://github.com/rksltnl/sparselet-release1

Chapter 4 focuses on the problem of learning to localize objects with *minimal supervision*. This chapter proposes a new method that achieves this goal with access to only image-level labels of whether the objects are present or not (such as images in figure 1.1). The approach combines a discriminative submodular cover problem for automatically discovering a set of positive object windows with a smoothed latent SVM formulation. The latter allows us to leverage efficient quasi-Newton optimization techniques. The experiments demonstrate that the proposed approach provides a 50% relative improvement in mean average precision over the current state-of-the-art on PASCAL VOC 2007 detection. The source code is available on Github at https://github.com/rksltnl/Song-ICML2014-release1

Chapter 5 provides a novel extension to the previous chapter and propose an approach that automatically identifies discriminative configurations of visual patterns that are characteristic of a given object class. This chapter formulates the problem as a constrained submodular optimization problem and demonstrate the benefits of the discovered configurations in remedying mislocalizations and finding informative positive and negative training examples. Together, these lead to state-of-the-art weakly-supervised detection results on the PASCAL VOC dataset. The source code will be made available on Github.

Chapter 6 concludes this thesis with future works and concluding remarks.

Chapter 2

Preliminary

This chapter gives a self contained introduction to some of the fundamental machine learning algorithms this thesis builds upon and also a brief overview on object detection.

2.1 Notations

This thesis uses plain lower case letter x for scalars, bold lower case letter $\mathbf{x}$ for vectors, capital letter X for matrices or sets. The $\mathbb{I}\left[\cdot\right]$ operator denotes the indicator function which is 1 if its argument is true and 0 otherwise. The $\mathrm{diag}\left(\cdot\right)$ operator create a diagonal matrix with it's input vector on the diagonal entries. The $\langle\cdot,\cdot\rangle$ operator denotes matrix inner product. Finally, the $\odot$ operator denotes the element wise matrix Hadamard product.

2.2 Overview of object detection and evaluation

Given an input image, the goal of object detection is to produce an output describing *what objects are where* [40]. Concretely, the task is to train discriminative models (i.e. person detector) which can localize all target objects (i.e. people) with tight bounding boxes on previously unseen images. Figure 2.1 illustrates an example person detection image. Orange boxes and the numbers above the boxes represent inferred locations and the corresponding confidence scores. Purple boxes represent ground truth boxes (typically obtained from human annotators) for evaluating the learned models.

Evaluating object detection performance is done by computing average precision(AP) based on detections on previously unseen held out dataset. Figure 2.2 illustrates this process. First, all the detections are sorted by the confidence score. Then, each detection boxes are checked to see if there are any nearby ground truth boxes with significant overlap and are assigned binary $\{\text{correct}, \text{incorrect}\}$ labels. Concretely, given a predicted detection box B_p, a nearest ground truth box B_{gt}, and overlap threshold θ_{th} (convention is to use $\theta_{th} = 0.5$),

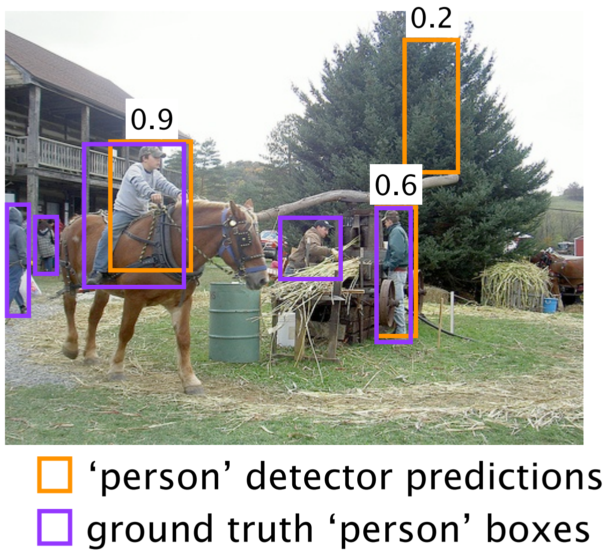

Figure 2.1: Example person detection result on a previously unseen test image. (Figure reproduced with permission from Ross Girshick)

the binary label is computed as follows:

$$\text{isCorrect}\left(B_p\right) = \mathbb{I}\left[\frac{\text{area}\left(B_p \cap B_{gt}\right)}{\text{area}\left(B_p \cup B_{gt}\right)} \geq \theta_{th}\right]$$

Then, thresholding at multiple confidence values, one can compute the corresponding precision(P) and recall(R). For a concrete example, in figure 2.2 at the first threshold (the first black vertical bar in between the rider detection and the sheep detection), the precision is 1.0 and the recall is $\frac{1}{\text{Number of people in the dataset}}$. Computing the precision and recall values at multiple threshold values create the precision recall curve as shown in figure 2.2. The final evaluation metric is then the area under the PR curve which is called average precision (AP).

2.3 Dictionary learning and sparse coding

Sparse models have appealing characteristics in that they are more interpretable and cheaper to use due to compactness. The objective of dictionary learning and sparse coding problem is to learn a dictionary of bases, $D = [\mathbf{d}_1, \ldots, \mathbf{d}_m]$ which can represent input data $\{\mathbf{x}\}_{i=1}^n$ via sparse linear combination of the dictionary elements with minimum reconstruction error. Equation 2.1 shows formal optimization problem. Note that the dictionary can either be overcomplete $m > d$ or undercomplete $m < d$ depending on the application. Few example applications for overcomplete dictionary are image denoising, super-resolution, inpainting

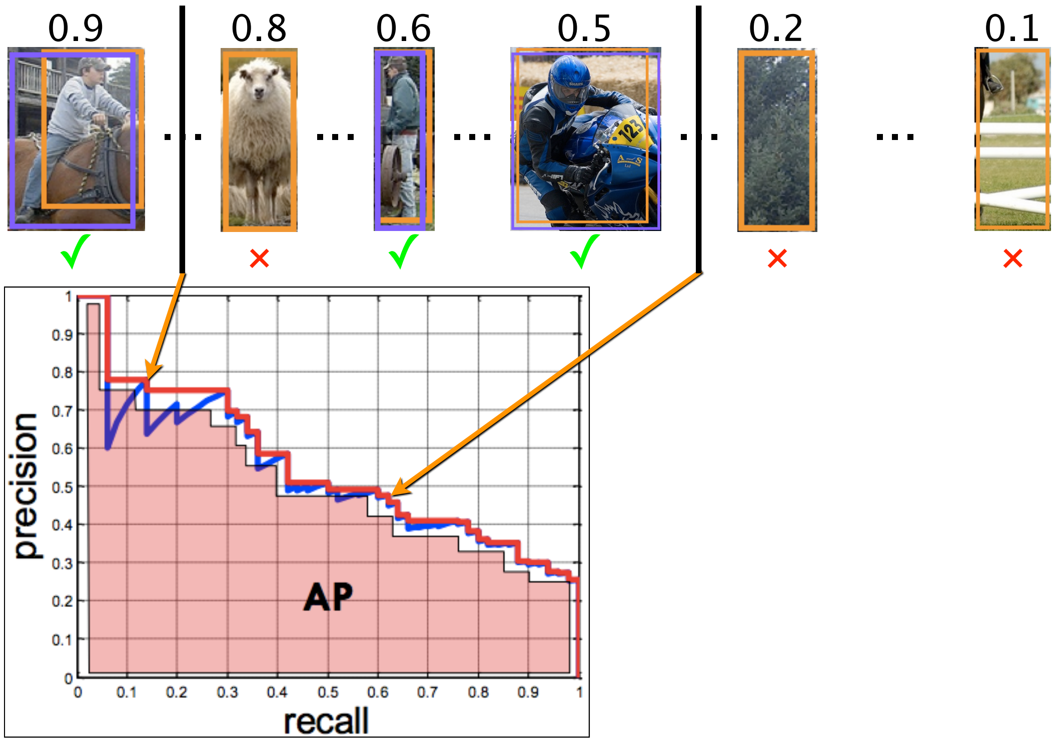

Figure 2.2: Precision-recall curve and average precision for the example detection image Fig. 2.1. The black vertical bars represent each confidence thresholds. Average precision(AP) is the area under the precision-recall curve. Mean average precision (mAP) is the result of averaging AP over different categories. (Figure reproduced with permission from Ross Girshick)

(see [64] for more details). This thesis will make use of undercomplete dictionary so as to compress the data into compact representation to harness computational efficiency. More details on the link between undercomplete dictionary and sparse codes versus the computational efficiency are in chapter 3.

$$\begin{aligned} \min_{\alpha_{ij},\mathbf{d}_j} \quad & \sum_{i=1}^{n} ||\mathbf{x}_i - \sum_{j=1}^{m} \alpha_{ij}\mathbf{d}_j||_2^2 \\ \text{subject to} \quad & ||\boldsymbol{\alpha}_i||_0 \le \lambda_0,\ \forall i = 1,\dots,n \\ & ||\mathbf{d}_j||_2^2 \ \le 1,\ \forall j = 1,\dots,m \end{aligned} \tag{2.1}$$

Although the above optimization is NP-hard, greedy algorithms such as orthogonal matching pursuit algorithm (OMP) [65] can be used to efficiently compute an approximate solution.

OMP iteratively estimates the optimal matching projections of the input signal onto the dictionary D. The above optimization problem is convex with respect to D if $\boldsymbol{\alpha}_i$ is fixed, and so we can optimize the objective in a coordinate descent fashion by iterating between updating $\boldsymbol{\alpha}_i$ while fixing D and vice versa. For our experiments we use the online dictionary learning algorithm from [64].

2.4 Submodularity

This section introduces fundamental concepts and algorithms on submodular maximization which we build upon in Chapters 4 and 5. A lot of the material in this section is borrowed from [53].

Definition 1 (Submodularity)**.** *A set function $F : 2^V \to \mathbb{R}$ which takes as input a subset $S \subseteq V$ of finite ground set V and assigns a function value $F(S)$ is called submodular if $\forall A \subseteq B \subseteq V$ and $k \in V \setminus B$, it holds that $F(A \cup \{k\}) - F(A) \geq F(B \cup \{k\}) - F(B)$. Or equivalently, a function $F : 2^V \to \mathbb{R}$ is submodular if $\forall A, B \subseteq V$, $F(A \cap B) + F(A \cup B) \leq F(A) + F(B)$.*

Note that the latter definition with set union and intersections does not require the a set to be a subset of another set.

Definition 2 (Monotonicity)**.** *A set function $F : 2^V \to \mathbb{R}$ is monotone if $\forall A \subseteq B \subseteq V$, it holds that $F(A) \leq F(B)$.*

Maximizing a general (non-monotone) submodular function without constraints is NP-hard (Max-cut is a special case). Also, maximizing a monotone submodular function is NP-hard for many constraints, even for cardinality constraints, $\max_{|S| \leq k} F(S)$. A monotone submodular function can be approximately maximized via a greedy algorithm. This algorithm subsequently selects an element from the ground set that has the maximal marginal gain with respect to the current selection. The algorithm approximately maximizes the problem by greedily selecting elements from the ground set which has the best marginal gain. Concretely, the algorithm proceeds by selecting

$$S_i := S_{i-1} \ \cup \ \left\{ \underset{\{e\} \notin S_{i-1}}{\operatorname{argmax}} F\left(S_{i-1} \cup \{e\}\right) - F\left(S_{i-1}\right) \right\},$$

until k elements have been chosen.

When the function is not only submodular but also nonnegative monotone, a result from [68] guarantees the approximation factor of about 63%. That is, denoting S^* as the maximizer and $S^\dagger$ as the greedy solution, $F\left(S^*\right) \geq (1 - 1/e)\, F\left(S^\dagger\right)$. We now introduce a richer class of constraints beyond the cardinality constraint through the notion of matroids and the corresponding approximation guarantees.

Definition 3 (Matroid)**.** *A matroid $(\mathcal{V}, \mathcal{I}_k)$ consists of a ground set $\mathcal{V}$ and a family $\mathcal{I}_k \subseteq 2^{\mathcal{V}}$ of "independent sets" that satisfy three axioms: (1) $\emptyset \in \mathcal{I}_k$; (2) downward closedness: if $S \in \mathcal{I}_k$ then $T \in \mathcal{I}_k$ for all $T \subseteq S$; and (3) the exchange property: if $S, T \in \mathcal{I}_k$ and $|S| < |T|$, then there is an element $v \in T \setminus S$ such that $S \cup \{v\} \in \mathcal{I}_k$.*

A "matroid constraint" means that we demand that the solution must be in $\mathcal{I}$, i.e., the selected set is an independent set in the matroid. A few examples of matroid constraints include:

Partition matroid Suppose the ground set is partitioned into non overlapping groups, $V = \bigcup_i G_i,\ G_i \bigcap_i G_j = \emptyset\ \forall i, j$. A partition matroid constraints the number elements which can be chosen from each group G_i. Formally, $\mathcal{I} = \{S \subseteq V|\ \ |S \cap G_i| \leq k, \forall\ i = 1, \ldots, p\}$.

Graphic matroid Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an arbitrary graph. Graphic matroid constraints the choice of subset of edges in the graph such that the selected nodes must not contain any cycles and thus must be a tree or forrest. Formally, $\mathcal{I} = \{S \subseteq V|\ \ S \text{ does not have a cycle}\}$.

Linear matroid Let V be n column vectors $\mathbf{v}_i \in \mathbb{R}^m$ where $n >> m$. Now, picking a collection of column vectors linearly independent of each other can be written as a matroid constraint. In this case, we use a linear matroid with $\mathcal{I} = \{S = v_1, \ldots, v_{|S|}|\ \ rank(S) = |S|\}$.

In case of matroid constraints, the maximization problem $\max_{S \in \mathcal{I}} F(S)$ can be approximately maximized with the greedy algorithm.

$$S_i := S_{i-1} \bigcup \left\{ \underset{\{e\} \notin S_{i-1}\ :\ S_{i-1} \cup \{e\} \in \mathcal{I}}{\operatorname{argmax}} F\left(S_{i-1} \cup \{e\}\right) - F\left(S_{i-1}\right) \right\},$$

which progressively selects elements from the ground set with the largest marginal gain until no more feasible elements can be added. The following theorem [7] provides the approximation guarantee for arbitrary number of matroid intersections.

Theorem 1 (Matroid intersection [7, 53])**.** *Supposed $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_p)$ are p matroids, and $\mathcal{I} = \cap_i \mathcal{I}_i$. That is $\mathcal{I}$ consists of all subsets of V that are independent in all p matroids. Even though $(V, \mathcal{I})$ is not generally a matroid anymore, the greedy algorithm is guaranteed to produce a solution so that $f(S_G) \geq \frac{1}{p+1} \max_{S \in \mathcal{I}} f(S)$.*

Example: bipartite matching
The following example shows that matching in a bipartite graph can be expressed as intersection of two matroid constraints. Fig. 2.3 shows an example bipartite graph $\mathcal{G} = \{\mathcal{U}, \mathcal{V}, \mathcal{E}\}$. A set of edges $S \subseteq \mathcal{E}$ form a matching if each node is adjacent to at most one edge in S. Formally, this can be written as intersection of two partition matroids with set cardinality n, m for sets $\mathcal{U}, \mathcal{V}$ respectively.

$$\begin{aligned}
&\mathcal{M} = \{\mathcal{E}, \mathcal{I}\},\ \mathcal{I} = \mathcal{I}_{\mathcal{U}} \cap \mathcal{I}_{\mathcal{V}} \\
&\mathcal{I}_{\mathcal{U}} = \{S \subseteq \mathcal{E} |\ \ |S \cap \mathcal{E}_i^{\mathcal{U}}| \leq 1,\ \forall\ i = 1, \dots n\} \\
&\mathcal{I}_{\mathcal{V}} = \{S \subseteq \mathcal{E} |\ \ |S \cap \mathcal{E}_i^{\mathcal{V}}| \leq 1,\ \forall\ i = 1, \dots m\},
\end{aligned} \tag{2.2}$$

where $\mathcal{E}_i^{\mathcal{U}}$ denotes the set of edges with left end point in the node u_i and $\mathcal{E}_i^{\mathcal{V}}$ denote the set of edges with the right end point in the node v_i.

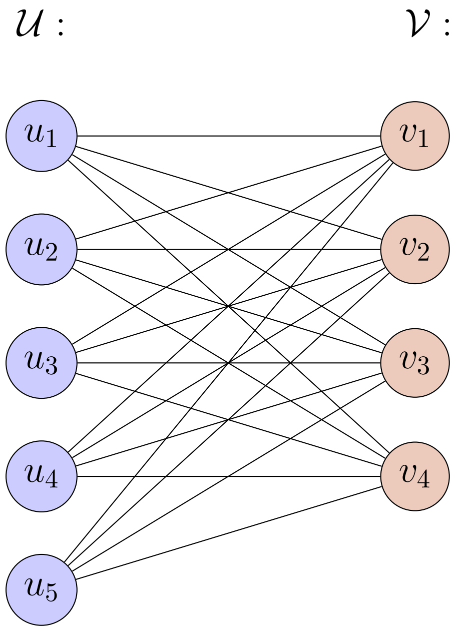

Figure 2.3: Example bipartite graph $\mathcal{G} = \{\mathcal{U}, \mathcal{V}, \mathcal{E}\}$

Lemma 3 in Chapter 5 uses these results to prove a matroid intersection bound on a graph.

Chapter 3

Generalized sparselet models for real-time multiclass object recognition

3.1 Introduction

Real-time category level recognition is a core requirement for visual competence in everyday environments. Domains of modest complexity typically have hundred to thousands of categories, and as one considers unconstrained search problems, the space of possible categories becomes practically unlimited. On top of that, modern object models [31, 5] consists of mixture of hundreds to thousands of object filters.

As the number of categories grows, individual models are increasingly likely to become redundant. In the case of part-based models this redundancy can be exploited by constructing models with shared parts. In this regard, shared intermediate representations are highly appealing due to their potential for gains in computational and statistical efficiency. These representations appear under a variety of guises, such as steerable filter banks [37], low-rank approximations for collaborative filtering [79], and shared part models for object detection [105, 71, 44].

Recently, sparselets [86, 41] were introduced as a new shared intermediate representation for multiclass object detection with deformable part models (DPMs) [31] by me. In this application, each sparselet can be thought of as a small, generic part (*e.g.*, a corner or edge) that is shared between all object categories. The parts of a DPM, for any class, are then constructed by tiling sparse linear combinations ("activations") of the sparselet mini-parts. With this representation, sparselets reconstruct approximate part responses using a sparse matrix-vector product instead of exhaustive convolutions.

In contrast to standard applications of sparse coding, where features are encoded as sparse combinations of overcomplete dictionary elements, sparselet models learn a dictionary of *model parameters*, and the models themselves are encoded as sparse combinations of dictionary elements. This leads to a compression of the models that can be exploited to speed-up computation.

The computational efficiency gains of this approach were demonstrated in a GPU sparselets implementation of DPM detection that outperformed a baseline GPU implementation by a factor of 3x to 5x, and outperformed the CPU version of the cascade algorithm in [32] by a factor of 15x, with almost no loss in detection average precision. The sparsity level used in this construction naturally trades off a decrease in detection accuracy for greater speed. However, the reconstructive method for learning activations proposed in [86] is brittle, and pushing slightly beyond these speedup factors leads to a substantial loss in detection accuracy.

This chapter also helps unify sparselets with the steerable part models of [74]. The fundamental differences between the two methods lies in how they accelerate inference and how they are trained. Steerable part models use a small part dictionary with dense linear combinations and discriminative training, whereas sparselets use a larger dictionary with sparse linear combination, and a reconstructive error training paradigm. With regard to dictionary size and linear combination density, the two approaches can be viewed as operating at different points within the same algorithm design space. The remaining difference, then, lies in the training method. This chapter unifies the two approaches by showing how to train sparselet activations discriminatively, or alternately, how to train steering coefficients sparsely.

3.2 Related Work

Our work is related to three strands of active research: (1) part sharing with compositional models [90, 35, 105, 71, 44], (2) sparse coding and dictionary learning [54, 64, 63], and (3) modeling and learning with low-rank approximations [37, 66, 100]. None of these methods, however, simultaneously exploit shared interclass information *and* discriminative sparsity learning to speed up inference while maintaining task performance.
A preliminary version of our system was described in [86, 41]. First we introduce the notion of generalized sparselets in structured output prediction problem [91, 88] and analyze the computational gains in efficiency. Also, we formulate a discriminative sparselet activation training framework and several regularization schemes that lead to improved sparsity and task performance. We experimentally demonstrate that the proposed sparselet activation learning algorithm substantially outperforms reconstructive sparselets and generalizes to previously unseen object categories.

This chapter is structured as follows. In Sec. 3.3, we start with a brief overview of sparselets [86] and formulate structured output prediction with generalized sparselets [41]. In Sec. 3.4, we describe how discriminative sparselet activation training fits into the framework and discuss several regularization methods for sparse activation learning. In Sec. 3.5, we discuss important applications of the proposed approach to multiclass object detection with mixtures of deformable part models [31] and to multiclass image classification. Before we conclude in Sec. 3.7, we provide experimental results on multiclass object detection and multiclass image classification problems in Sec. 3.6.

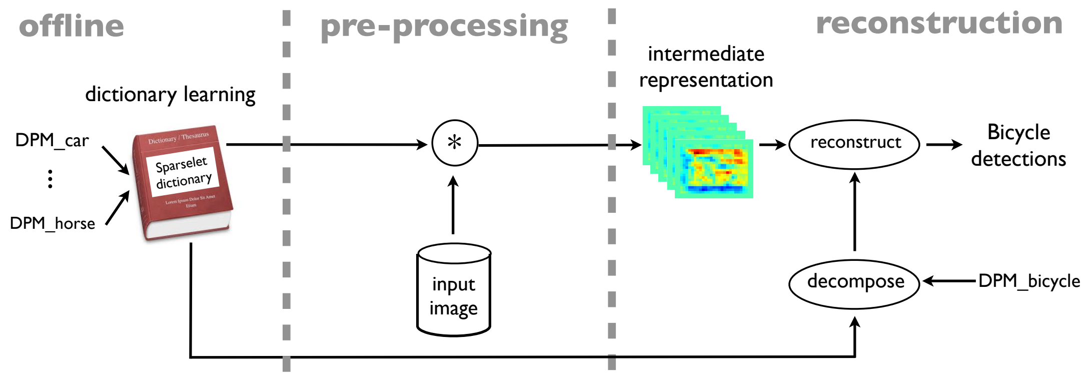

Figure 3.1: System concept.

3.3 Sparselets

In general, convolution of a feature pyramid with thousands of object model filters becomes the major computational bottleneck in multiclass object detection tasks. *Sparselet model* tackles this problem by learning a dictionary of "universal" object models that generalizes to previously unseen classes and expressing filter convolutions stage as sparse linear combinations of sparselet convolutions. Fig. 3.1 illustrates the concept in three stages. Also, the sparselet dictionary size is independent on number of classes and the speedup offered by the method approaches the ratio between number of classes and the sparsity in reconstruction weights as number of classes increases.

Sparse reconstruction of object models

Formally, a *sparselet model* is completely defined by a dictionary $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_d]$ in $\mathbb{R}^{m\times d}$, where each column $\mathbf{s}_i$ in $\mathbb{R}^m$ is called a *sparselet*. We formulate the following optimization problem to derive sparselet models which reconstruct linear object filters $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_K]$ in $\mathbb{R}^{m\times K}$ pooled from a set of training models via sparse linear combination.

$$\begin{aligned} &\min_{\alpha_{ij},\mathbf{s}_j} \sum_{i=1}^{K} ||\mathbf{w}_i - \sum_{j=1}^{d} \alpha_{ij}\mathbf{s}_j||_2^2 \\ &\text{subject to} \quad ||\boldsymbol{\alpha_i}||_0 \leq \lambda_0 \quad \forall i = 1, ..., K \\ &\qquad\qquad\quad ||\mathbf{s}_j||_2^2 \leq 1 \quad \forall j = 1, ..., d \end{aligned} \tag{3.1}$$

Although the above optimization is NP-hard, greedy algorithms such as orthogonal matching pursuit algorithm (OMP) [65] can be used to efficiently compute an approximate solution. OMP iteratively estimates the optimal matching projections of the input signal

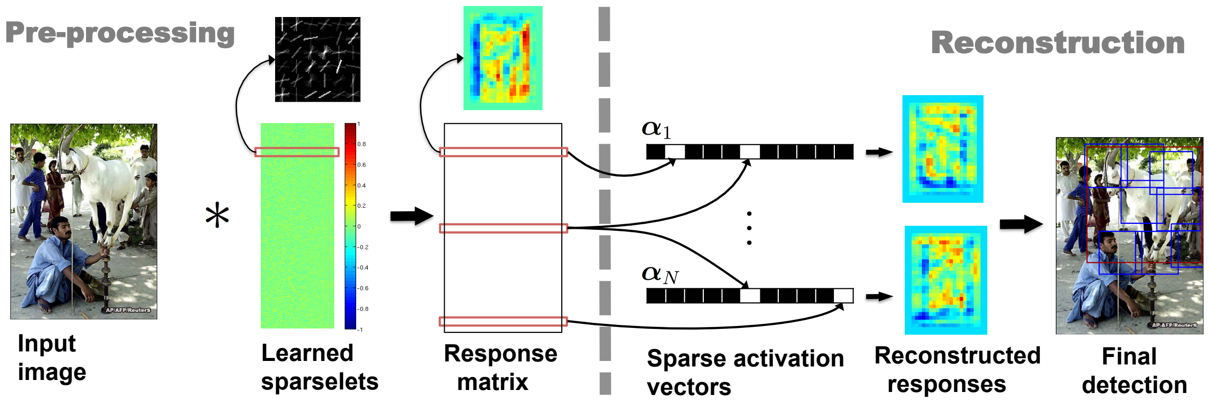

Figure 3.2: Overview diagram of object detection with sparselets. Once we evaluate the image with learned sparselets, the reconstruction phase can be done via efficient sparse matrix vector multiplications.

onto the dictionary $\mathbf{S}$. The above optimization problem is convex with respect to $\mathbf{S}$ if $\boldsymbol{\alpha}_i$ is fixed, and so we can optimize the objective in a coordinate descent fashion by iterating between updating $\boldsymbol{\alpha}_i$ while fixing $\mathbf{S}$ and vice versa. For our experiments we use the online dictionary learning algorithm from [64].

Precomputation and efficient reconstruction

We can precompute convolutions for all sparselets, and by linearity of convolution we can then use the activation vectors estimated for a target object detector to approximate the convolution response we would have obtained from convolution with the original filters. Denoting the feature pyramid of an image as $\boldsymbol{\Psi}$, we have

$$\boldsymbol{\Psi} * \mathbf{w}_i \approx \boldsymbol{\Psi} * \sum_j \alpha_{ij}\mathbf{s}_i = \sum_j \alpha_{ij}\left(\boldsymbol{\Psi} * \mathbf{s}_i\right) \tag{3.2}$$

Concretely, we can recover individual part filter responses via sparse matrix multiplication (or lookups) with the activation vector replacing the heavy convolution operation as shown in Eqn. 3.3:

$$\begin{bmatrix} \text{---}\ \boldsymbol{\Psi} * \mathbf{w}_1\ \text{---} \\ \text{---}\ \boldsymbol{\Psi} * \mathbf{w}_2\ \text{---} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \text{---}\ \boldsymbol{\Psi} * \mathbf{w}_K\ \text{---} \end{bmatrix} \approx \begin{bmatrix} \text{---}\ \boldsymbol{\alpha}_1\ \text{---} \\ \text{---}\ \boldsymbol{\alpha}_2\ \text{---} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \text{---}\ \boldsymbol{\alpha}_K\ \text{---} \end{bmatrix} \begin{bmatrix} \text{---}\ \boldsymbol{\Psi} * \mathbf{s}_1\ \text{---} \\ \text{---}\ \boldsymbol{\Psi} * \mathbf{s}_2\ \text{---} \\ \vdots \\ \text{---}\ \boldsymbol{\Psi} * \mathbf{s}_d\ \text{---} \end{bmatrix} \tag{3.3}$$

where the first matrix is the matrix of sparse activation vectors and the second matrix is a matrix of all sparselet responses. The sparselet responses $\boldsymbol{\Psi} * \mathbf{s}_i$ are independent of any filter, and thus their cost can be amortized over all filters from all object models. Figure 3.2 shows the overview of the approach. In the remainder of this section we present a novel generalization of this technique. First, we illustrate how to generalize sparselets for simple multiclass linear classifiers, and then for any linear structured output prediction model.

Generalized sparselets for structured output prediction

Consider a set of K linear classifiers parameterized by the weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K$ each in $\mathbb{R}^n$. An input feature vector $\mathbf{x} \in \mathbb{R}^n$ is assigned to a class $f_{\mathbf{w}}(\mathbf{x}) \in \{1, \ldots, K\}$ according to the rule

$$f_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{k \in \{1,\ldots,K\}} \mathbf{w}_k^\intercal \mathbf{x}. \tag{3.4}$$

Our objective is to reduce the computational cost of computing Eq. 3.4.

We begin by partitioning each parameter vector $\mathbf{w}_k$ into several m-dimensional *blocks*. A block is a subvector of parameters chosen so that the set of all blocks admits a sparse representation over $\mathbf{S}$. Concretely, in the examples that follow blocks will be chosen to be fragments of part filters in a deformable part model (see Fig. 3.4), or simply contiguous subvectors of the parameters in a bag-of-visual-words classifier. For clarity, we will assume that $n = pm$ for some positive integer p. We can rewrite each linear classifier in terms of its blocks, $\mathbf{b}_{ki}$ in $\mathbb{R}^m$, such that $\mathbf{w}_k = (\mathbf{b}_{k1}^\intercal, \ldots, \mathbf{b}_{kp}^\intercal)^\intercal$. Similarly, we can partition an input feature vector into m-dimensional subvectors, $\mathbf{c}_i$ in $\mathbb{R}^m$, such that $\mathbf{x} = (\mathbf{c}_1^\intercal, \ldots, \mathbf{c}_p^\intercal)^\intercal$.

Given a sparselet model $\mathbf{S}$, we can approximate any vector $\mathbf{b} \in \mathbb{R}^m$ as a sparse linear combination of the sparselets in $\mathbf{S}$

$$\mathbf{b} \approx \mathbf{S}\boldsymbol{\alpha} = \sum_{\substack{i=1,\\ \alpha_i \neq 0}}^{d} \alpha_i \mathbf{s}_i, \tag{3.5}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)^\intercal \in \mathbb{R}^d$ is a *sparselet activation vector* for $\mathbf{b}$. The quality of the approximation depends on the fixed dictionary and the chosen activation vector. Now, the dot product in Eq. 3.4 can be approximated as

$$\begin{aligned}\mathbf{w}_k^\intercal \mathbf{x} &= (\mathbf{b}_{k1}^\intercal, \ldots, \mathbf{b}_{kp}^\intercal)(\mathbf{c}_1^\intercal, \ldots, \mathbf{c}_p^\intercal)^\intercal \\ &= \sum_{i=1}^{p} \mathbf{b}_{ki}^\intercal \mathbf{c}_i \approx \sum_{i=1}^{p} (\mathbf{S}\boldsymbol{\alpha}_{ki})^\intercal \mathbf{c}_i = \sum_{i=1}^{p} \boldsymbol{\alpha}_{ki}^\intercal (\mathbf{S}^\intercal \mathbf{c}_i). \end{aligned} \tag{3.6}$$

We note two important properties of Eq. 3.6: (1) the *sparselet responses* $\mathbf{S}^\intercal \mathbf{c}_i$ are independent of any particular classifier, and (2) the subsequent product with $\boldsymbol{\alpha}_{ki}$ can be computed efficiently by accessing only the nonzero elements of $\boldsymbol{\alpha}_{ki}$. In the following, let λ_0 be the average number of nonzero elements in each $\boldsymbol{\alpha}_{ki}$.

Multiclass classification is a special case of structured output prediction. To complete the description of generalized sparselets for structured output prediction, consider the linear discriminant function

$$f_{\mathbf{w}}(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, \mathbf{w}^\intercal \boldsymbol{\Phi}(\mathbf{x}, \mathbf{y}) \tag{3.7}$$

where the input $\mathbf{x}$ comes from an arbitrary input space $\mathcal{X}$, and $f_{\mathbf{w}}$ outputs an element from the label space $\mathcal{Y}$. In the following subsection, we give concrete examples of how the weight vector $\mathbf{w}$ is partitioned into *blocks* and analyze the computational cost for three scenarios: multiclass classification, multiclass convolutional classifiers and part based models.

Computational cost analysis

We can analyze generalized sparselets for multiclass classification by looking at the cost of computing $\mathbf{b}_{ki}^\intercal \mathbf{c}_i$ for a single block i and for all classes k. The original classifiers require Km additions and multiplications. The generalized sparselet approach has a shared cost of dm operations for computing the sparselet responses, $\mathbf{r}_i = \mathbf{S}^\intercal \mathbf{c}_i$, and a cost of $K\lambda_0$ operations for computing $\boldsymbol{\alpha}_{ki}^\intercal \mathbf{r}_i$ for all classes. The overall speedup is thus $Km/(dm + K\lambda_0)$. To make this value large, the dictionary size d should be much smaller than the number of classes K, and the average number of nonzero coefficients in the activation vectors should be much less than the sparselet size m. As the number of classes becomes large, the cost of computing sparselet responses becomes fully amortized which leads to a maximum theoretical speedup of m/λ_0 [86]. This emphasizes the importance of a sparse representation, in contrast, for example, to the dense steering coefficients in [74]. This analysis shows that generalized sparselets are most applicable to multiclass problems with a large number of classes. This is a regime of growing interest, especially in computer vision as exemplified by new datasets such as ImageNet [14], which includes more than 10,000 categories [15]. In Sec. 3.6 we show results on the Caltech-{101,256} [27, 46] datasets demonstrating that even with only one or two hundred classes generalized sparselets can accelerate simple linear classifiers.

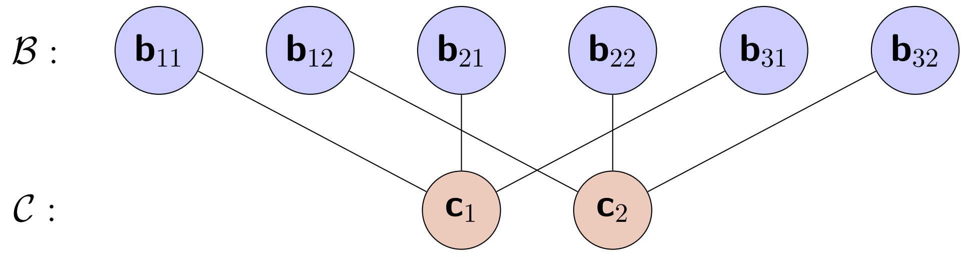

Figure 3.3: Computation graph for a multiclass problem with $K = 3$. Let the sparselet size be m and the number of blocks be $p = 2$. We define $\mathbf{w} = (\mathbf{w}_1^\intercal, \mathbf{w}_2^\intercal, \mathbf{w}_3^\intercal)^\intercal$ in $\mathbb{R}^{Kpm}$. Each per-class classifier $\mathbf{w}_k$ in $\mathbb{R}^{pm}$ is partitioned into p blocks such that $\mathbf{w}_k = (\mathbf{b}_{k1}^\intercal, \mathbf{b}_{k2}^\intercal)^\intercal$. An input vector $\mathbf{x}$ in $\mathbb{R}^{pm}$ is partitioned into subvectors such that $\mathbf{x} = (\mathbf{c}_1^\intercal, \mathbf{c}_2^\intercal)^\intercal$. The feature map $\mathbf{\Phi}(\mathbf{x}, k)$ in $\mathbb{R}^{Kpm}$ is defined as: $\mathbf{\Phi}(\mathbf{x}, 1) = (\mathbf{x}^\intercal, 0, \ldots, 0)^\intercal$; $\mathbf{\Phi}(\mathbf{x}, 2) = (0, \ldots, 0, \mathbf{x}^\intercal, 0, \ldots, 0)^\intercal$; $\mathbf{\Phi}(\mathbf{x}, 3) = (0, \ldots, 0, \mathbf{x}^\intercal)^\intercal$. The edges in the graph encode the dot products computed while solving $\operatorname*{argmax}_{k \in \{1,2,3\}} \mathbf{w}^\intercal \mathbf{\Phi}(\mathbf{x}, k)$.

To generalize the analysis to the structured prediction setting, we rewrite the speedup as $Qm/(dm + Q\lambda_0)$, where Q is defined to be *the number of times a given subvector of feature values is multiplied with a unique parameter block*. Intuitively, Q counts the number of times the intermediate sparselet response is reused while solving the argmax in Eq. 3.7. The value of Q depends on the specific feature map $\mathbf{\Phi}(x, y)$ and inference algorithm.

For clarity, we will assume that $n = pm$ for some integer p, where m is the length of each sparselet $\mathbf{s}_i$ in the sparselet dictionary $\mathbf{S}$. This assumption can be removed with simple modifications to the discussion that follows. We partition $\mathbf{w}$ into a set of blocks $\mathbf{b}_i$ in $\mathbb{R}^m$ such that $\mathbf{w} = (\mathbf{b}_1^\intercal, \ldots, \mathbf{b}_p^\intercal)^\intercal$.

Let $\mathcal{A}$ be an algorithm such that $\mathcal{A}(\mathbf{w}, x)$ computes $f_{\mathbf{w}}(x)$ — *i.e.*, it solves the argmax in Eq. 3.7. We are going to build a bipartite graph $\mathcal{G} = (\mathcal{B} \cup \mathcal{C}, \mathcal{E})$ that represents certain computations performed by $\mathcal{A}$. The graph depends on $\mathcal{A}$'s inputs $\mathbf{w}$ and x, but to lighten notation we will omit this dependence.

Each node in $\mathcal{G}$ corresponds to a vector in $\mathbb{R}^m$. With a slight abuse of notation we will label each node with the vector that it is in correspondence with. Similarly, we will label the edges with a pair of vectors (*i.e.*, nodes), each in $\mathbb{R}^m$. We define the first set of disconnected nodes in $\mathcal{G}$ to be the set of all blocks in $\mathbf{w}$: $\mathcal{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_p\}$. We will define the second set of disconnected nodes, $\mathcal{C}$, next.

Any algorithm that computes Eq. 3.7 will perform some number of computations of the form $\mathbf{b}^\intercal\mathbf{c}$, for a block $\mathbf{b} \in \mathcal{B}$ and some vector $\mathbf{c} \in \mathbb{R}^m$. The vectors $\mathbf{c}$ appearing in these computations are most likely subvectors of $\mathbf{\Phi}(x, y)$ arising from various values of y. The graph $\mathcal{G}$ is going to represent all unique computations of this form. Conceptually, we can construct $\mathcal{C}$ by running the algorithm $\mathcal{A}$ and adding each *unique* vector $\mathbf{c}$ that appears in a computation of the form $\mathbf{b}^\intercal\mathbf{c}$ to $\mathcal{C}$. The edge set $\mathcal{E}$ connects a node $\mathbf{b} \in \mathcal{B}$ to a node in $\mathbf{c} \in \mathcal{C}$

if and only if $\mathcal{A}$ performs the computation $\mathbf{b}^{\intercal}\mathbf{c}$. For a specific algorithm $\mathcal{A}$, we can construct $\mathcal{G}$ analytically. An example graph for a multiclass classification problem is given in Fig. 3.3.

Graph $\mathcal{G}$'s edges encode exactly all of the computations of the form $\mathbf{b}^{\intercal}\mathbf{c}$ and therefore we can use it to analyze the computational costs of $\mathcal{A}$ with and without generalized sparselets.

Obviously, not all of the computation performed by $\mathcal{A}$ are of the form captured by the graph. For example, when generalized distance transforms are used by $\mathcal{A}$ to solve in the computation of Eq. 3.7 for deformable part models, the cost of computing the distance transforms is outside of the scope of $\mathcal{G}$ (and outside the application of sparselets). We let the quantity $T(\mathbf{w}, x)$ account for all computational costs not represented in $\mathcal{G}$.

We are now ready to write the number of operations performed by $\mathcal{A}(\mathbf{w}, x)$. First, without sparselets we have

$$T_{\text{Original}}(\mathbf{w}, x) = T(\mathbf{w}, x) + m \sum_{\mathbf{c} \in \mathcal{C}} \deg(\mathbf{c}), \tag{3.8}$$

where $\deg(\mathbf{v})$ is the degree of a node $\mathbf{v}$ in $\mathcal{G}$. The second term in Eq. 3.8 accounts for the m additions and multiplications that are performed when computing $\mathbf{b}^{\intercal}\mathbf{c}$ for a pair of nodes $(\mathbf{b}, \mathbf{c}) \in \mathcal{E}$.

When sparselets are applied, the cost becomes

$$T_{\text{Sparselets}}(\mathbf{w}, x) = T(\mathbf{w}, x) + dm|\mathcal{C}| + \lambda_0 \sum_{\mathbf{c} \in \mathcal{C}} \deg(\mathbf{c}), \tag{3.9}$$

The second term in Eq. 3.9 accounts for the cost of precomputing the sparselet responses, $\mathbf{r} = \mathbf{S}^{\intercal}\mathbf{c}$ (cost dm), for each node in $\mathcal{C}$. The third term accounts for the sparse dot product $\boldsymbol{\alpha}(\mathbf{b})^{\intercal}\mathbf{r}$ (cost λ_0) computed for each pair $(\mathbf{b}, \mathbf{c}) \in \mathcal{E}$, where $\boldsymbol{\alpha}(\mathbf{b})$ is the sparselet activation vector for $\mathbf{b}$.

The speedup is the ratio $T_{\text{Original}}/T_{\text{sparselets}}$.

$$\frac{T(\mathbf{w}, x) + m \sum_{i=1}^{|\mathcal{C}|} \deg(\mathbf{c}_j)}{T(\mathbf{w}, x) + dm|\mathcal{C}| + \lambda_0 \sum_{i=1}^{|\mathcal{C}|} \deg(\mathbf{c}_j)} \tag{3.10}$$

In all of the examples we consider in this chapter, the degree of each node in $\mathcal{C}$ is a single constant: $\deg(\mathbf{c}) = Q$ for all $\mathbf{c} \in \mathcal{C}$. In this case, the speedup simplifies to the following.

$$\frac{T(\mathbf{w}, x) + Q|\mathcal{C}|m}{T(\mathbf{w}, x) + dm|\mathcal{C}| + Q|\mathcal{C}|\lambda_0} \tag{3.11}$$

If we narrow our scope to only consider the speedup restricted to the operations of $\mathcal{A}$ affected by sparselets, we can ignore the $T(\mathbf{w}, x)$ terms and note that the $|\mathcal{C}|$ factors cancel.

$$\frac{Qm}{dm + Q\lambda_0} \tag{3.12}$$

This narrowing is justified in the multiclass classification case (with K classes) where the cost $T(\mathbf{w}, x)$ amounts to computing the maximum value of K numbers, which is negligible compared to the other terms. We also observe that $Q = K$, yielding the following speedup.

$$\frac{Km}{dm + K\lambda_0} \tag{3.13}$$

The computation graph for a simple multiclass example with $K = 3$ is given in Fig. 3.3. For more intuition, we consider two examples below.

Multiclass convolutional classifiers. Consider the multiclass setting and let $\mathbf{w} = (\mathbf{w}_1^\intercal, \ldots, \mathbf{w}_K^\intercal)^\intercal$ in $\mathbb{R}^{Kn}$. As before, each $\mathbf{w}_k$ is partitioned into p blocks. But now, instead of an n-dimensional input feature vector, consider larger input vectors $\mathbf{x} \in \mathbb{R}^q$, $q \gg n$, and the feature map $\mathbf{\Phi}(\mathbf{x}, (k, y)) = (0, \ldots, 0, \mathbf{x}_{y:n}^\intercal, 0, \ldots, 0)^\intercal$. We write $\mathbf{x}_{y:n}$ to denote the length-n subvector of $\mathbf{x}$ starting at position y. This subvector is placed into the k-th "slot" of $\mathbf{\Phi}$ (corresponding to the slot for $\mathbf{w}_k$ in $\mathbf{w}$). The label space $\mathcal{Y}$ consists of all valid (class, position) pairs (k, y). This setup is equivalent to the problem of searching for the subvector of $\mathbf{x}$ that has maximum correlation with a weight vector in $\{\mathbf{w}_k\}$. A concrete example of this is multiclass object detection with Dalal and Triggs style scanning window detectors [12]. In contrast to the non-convolutional multiclass setting, now each block of $\mathbf{w}$ must be multiplied with each subvector of $\mathbf{x}$ while scanning for the maximum response (imagine "sliding" each $\mathbf{w}_k$ over $\mathbf{x}$ while computing a dot product at each position), and thus $Q = Kp$.

Part-based models. Another common situation that leads to a large Q value is when $\mathbf{w}$ parameterizes a set of "parts" and $f_{\mathbf{w}}(x)$ computes the optimal assignment of the parts to locations y in the input x. For example, a location y might be a position in a sentence or an image. In this problem setting, there is a (typically very large) pool of feature vectors, where each vector in the pool describes one location in x. The feature map $\mathbf{\Phi}(x, y)$ acts on a label y by installing the selected subset of local feature vectors into the appropriate slots of $\mathbf{\Phi}$. These problems typically also involve pairwise interactions between the labels assigned to some pairs of parts. When these interactions form a tree, dynamic programming can be used to efficiently compute the optimal label assignments. In the dynamic programming algorithm, the dot product between each part model and each local feature vector must be evaluated. As a concrete example, consider the deformable part models of [31]. For this model, the dynamic programming algorithm implicitly generates the large set of local feature vectors through the convolution of each part with a histogram of oriented gradients (HOG) feature image [12, 31]. Given object detectors for K classes, each with N parts, each of which is partitioned into p blocks, this model and algorithm result in $Q = KNp$. The part-based structure of this problem increases sparselet response reuse by a factor of N.

3.4 Discriminative activation of generalized sparselets

Throughout the rest of this chapter we consider linear models defined by parameter vectors that are partitioned into K slots: $\mathbf{w} = (\mathbf{w}_1^\intercal, \ldots, \mathbf{w}_K^\intercal)^\intercal$. In the multiclass setting, slots correspond to the individual classifiers. More generally, slots might be structures like the filters in a deformable part model. Generalized sparselets may be applied to any subset of the slots. For a slot $\mathbf{w}_k$ to which sparselets are applied, it is further partitioned into p_k blocks: $\mathbf{w}_k = (\mathbf{b}_{k1}^\intercal, \ldots, \mathbf{b}_{kp_k}^\intercal)^\intercal$. The $\{\mathbf{w}_k\}$ may have different dimensions, as long as each is a multiple of the sparselet dimension m.

In [86], the task of learning the sparselet model $\mathbf{S}$ from a training set of parameter blocks $\{\mathbf{b}_{ki}\}$ was naturally posed as a sparse coding dictionary learning problem [54, 64]. The objective was to find a dictionary $\mathbf{S}$ and activation vectors $\{\boldsymbol{\alpha}_{ki}\}$ that minimize reconstruction error, subject to an ℓ_0-pseudo-norm sparsity constraint on each activation vector. Then, given the learned dictionary $\mathbf{S}$, the activation vectors for a model $\mathbf{w}$ (either previously unseen or from the training set) were learned by minimizing reconstruction error, subject to the same sparsity constraint.

The experimental results in [86] show that task performance (average precision for object detection) quickly degrades to undesirable levels as the activation vectors are made increasingly sparse. This result is intuitive given the reconstructive activation vector learning method used in [86]. When reconstruction error is low (*i.e.* low sparsity), the original decision boundary of the model is roughly preserved. However, as sparsity increases, and the reconstruction error becomes larger, the decision boundary of the reconstructed model changes in an uncontrolled way and may no longer be discriminative for the target task.

Our solution is to replace reconstruction error with a discriminative objective. To do this (assuming a fixed dictionary), we propose to rewrite the original optimization problem used for training the linear model in terms of sparselet responses, which now act as training features, and the activation vectors, which now act as the model parameters. To achieve sparsity, we augment this new objective function with a sparsity inducing regularizer. As we show below, the somewhat obvious choice of ℓ_1 regularization leads to unsatisfactory results, motivating the development of an alternative approach.

Learning discriminative activation vectors

Here we consider learning the activation vectors for a predictor $\mathbf{w}$ in the structural SVM (SSVM) framework [91, 88]. The SSVM training equation is

$$\begin{aligned}\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \\ \frac{1}{M}\sum_{i=1}^{M} \max_{\hat{y}\in\mathcal{Y}} \left(\mathbf{w}^\intercal \boldsymbol{\Phi}(x_i, \hat{y}) + \Delta(y_i, \hat{y})\right) - \mathbf{w}^\intercal \boldsymbol{\Phi}(x_i, y_i),\end{aligned} \tag{3.14}$$

where $\Delta(y, y')$ is a loss function. Given a fixed sparselet model $\mathbf{S}$, we can rewrite Eq. 3.14 in terms of the activation vector parameters and sparselet responses. For clarity, assume the slots of $\mathbf{w}$ have been arranged so that slots 1 through s are represented with sparselets, and slots $s+1$ through K are not.[1] For each slot $\mathbf{w}_k = (\mathbf{b}_{k1}^\intercal, \ldots, \mathbf{b}_{kp_k}^\intercal)^\intercal$ that is represented by sparselets, we define a corresponding activation parameter vector $\boldsymbol{\alpha}_k = (\boldsymbol{\alpha}_{k1}^\intercal, \ldots, \boldsymbol{\alpha}_{kp_k}^\intercal)^\intercal \in \mathbb{R}^{p_k d}$. Let $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1^\intercal, \ldots, \boldsymbol{\alpha}_s^\intercal)^\intercal$ and $\tilde{\mathbf{w}} = (\mathbf{w}_{s+1}^\intercal, \ldots, \mathbf{w}_K^\intercal)^\intercal$, and define the new model parameter vector $\boldsymbol{\beta} = (\boldsymbol{\alpha}^\intercal, \tilde{\mathbf{w}}^\intercal)^\intercal$.

We transform the feature vector in a similar manner. For a feature vector slot $\boldsymbol{\Phi}_k(x, y) = (\mathbf{c}_1^\intercal, \ldots, \mathbf{c}_{p_k}^\intercal)^\intercal$ that will be represented by sparselets, we transform the features into sparselet responses: $\tilde{\boldsymbol{\Phi}}_k(x, y) = (\mathbf{c}_1^\intercal\mathbf{S}, \ldots, \mathbf{c}_{p_k}^\intercal\mathbf{S})^\intercal \in \mathbb{R}^{p_k d}$. The fully transformed feature vector is $\tilde{\boldsymbol{\Phi}}(x, y) = (\tilde{\boldsymbol{\Phi}}_1^\intercal(x, y), \ldots, \tilde{\boldsymbol{\Phi}}_s^\intercal(x, y), \boldsymbol{\Phi}_{s+1}^\intercal(x, y), \ldots, \boldsymbol{\Phi}_K^\intercal(x, y))^\intercal$. The resulting objective is

$$\begin{aligned}\boldsymbol{\beta}^* = \operatorname*{argmin}_{\boldsymbol{\beta}} R(\boldsymbol{\alpha}) &+ \frac{\lambda}{2}\|\tilde{\mathbf{w}}\|_2^2 \\ &+ \frac{1}{M}\sum_{i=1}^{M} \max_{\hat{y}\in\mathcal{Y}} \Big(\boldsymbol{\beta}^\intercal\tilde{\boldsymbol{\Phi}}(x_i, \hat{y}) + \Delta(y_i, \hat{y})\Big) - \boldsymbol{\beta}^\intercal\tilde{\boldsymbol{\Phi}}(x_i, y_i),\end{aligned} \tag{3.15}$$

where $R(\boldsymbol{\alpha})$ is a regularizer applied to the activation vectors.

Inducing sparsity

We consider three sparsity inducing regularizers R.

I. **L**asso penalty [89]
$R_{\text{Lasso}}(\boldsymbol{\alpha}) = \lambda_1\|\boldsymbol{\alpha}\|_1$

II. **E**lastic net penalty [106]
$R_{\text{EN}}(\boldsymbol{\alpha}) = \lambda_1\|\boldsymbol{\alpha}\|_1 + \lambda_2\|\boldsymbol{\alpha}\|_2^2$

III. **C**ombined ℓ_0 and ℓ_2 penalty
$R_{0,2}(\boldsymbol{\alpha}) = \lambda_2\|\boldsymbol{\alpha}\|_2^2$ subject to $\|\boldsymbol{\alpha}\|_0 \le \lambda_0$

The first two regularizers lead to convex optimization problems, however the third does not. We consider two alternative methods for approximately minimizing Eq. 3.15 when $R(\boldsymbol{\alpha}) = R_{0,2}(\boldsymbol{\alpha})$. Both of these methods employ a two step process. In the first step, a subset of the activation coefficients is selected to satisfy the constraint $\|\boldsymbol{\alpha}\|_0 \le \lambda_0$. In the second step, the selection of nonzero variables is fixed (thus satisfying the sparsity constraint) and the resulting convex optimization problem is solved. We consider the following variable selection strategies.

[1]This flexibility lets us leave slots where sparselets don't make sense unchanged, *e.g.* a bias parameter slot.

III-A. **O**vershoot, rank, and threshold (ORT). In this method, we first apply either R_{Lasso} or R_{EN} with λ_1 set to *overshoot* the target number of nonzero variables λ_0. We then rank the nonzero activation coefficients by their magnitudes and select the λ_0 variables with the largest magnitudes. Each variable in the selected variable set's complement is thresholded to zero.

III-B. **O**rthogonal matching pursuit (OMP). In this method, we select the nonzero variables by minimizing the reconstruction error between parameter blocks and their sparse coding approximation subject to the constraint $\|\boldsymbol{\alpha}\|_0 \leq \lambda_0$. In practice, we use orthogonal matching pursuit [65] as implemented in SPAMS software package [64]. This produces the same initial set of activation vectors as the baseline method [86]. However, we then learn the selected variables discriminatively according to Eq. 3.15.

3.5 Application of generalized sparselets

We first focus on the application of our novel sparselet activation vector learning approach to object detection with mixtures of deformable part models [31] in order to facilitate direct comparison with the results in [86]. In brief, the deformable part model (DPM) from [31] is specified by a root filter that models the global appearance of an object class and a set of N part filters that capture local appearance. The part filters are attached to the root filter by flexible "springs" that allow the model to match the image with a deformed arrangement of parts. In practice, several DPMs are combined into one mixture model to better represent more extreme variation in object class appearance.

A DPM is matched to an image by maximizing a score function over latent variables z. Let $z = (c, \rho_0, \ldots, \rho_N)$ specify a mixture component $c \in \{1, \ldots, C\}$, root filter location ρ_0, and part filter locations $\rho_1, \ldots, \rho_N$ for a model with C components and N part filters. The score function can be written as

$$\begin{aligned} \text{score}(x, z) = w_c &+ \sum_{i=0}^{N} \mathbf{w}_{ci}^{\intercal} \boldsymbol{\psi}_{ci}(x, \rho_i) \\ &+ \sum_{i=1}^{N} \mathbf{d}_{ci}^{\intercal} \boldsymbol{\delta}_{ci}(\rho_0, \rho_i) = \mathbf{w}^{\intercal} \boldsymbol{\Phi}(x, z), \end{aligned} \tag{3.16}$$

where $\mathbf{w}_{ci}$ are the weights in filter i of component c, $\mathbf{d}_{ci}$ are the quadratic deformation parameters specifying the stiffness of the spring connecting the root filter and part filter i of component c, and w_c is a score bias. The feature functions $\boldsymbol{\psi}_{ci}(x, \rho_i)$ and $\boldsymbol{\delta}_{ci}(\rho_0, \rho_i)$ are local image features (HOG) and deformation features, respectively. The score can be written as a single dot production between

$$\begin{aligned} \mathbf{w} = (&w_1, \ldots, w_C, \mathbf{w}_{10}^{\intercal}, \ldots, \mathbf{w}_{1N}^{\intercal}, \ldots, \mathbf{w}_{C0}^{\intercal}, \ldots, \mathbf{w}_{CN}^{\intercal}, \\ &\mathbf{d}_{11}^{\intercal}, \ldots, \mathbf{d}_{1N}^{\intercal}, \ldots, \mathbf{d}_{C1}^{\intercal}, \ldots, \mathbf{d}_{CN}^{\intercal})^{\intercal} \end{aligned} \tag{3.17}$$

and a sparse cumulative feature vector $\mathbf{\Phi}(x, z)$ that is laid out with the same slots as $\mathbf{w}$.

We apply sparselets to *all* filter slots of $\mathbf{w}$, *i.e.*, the $\{\mathbf{w}_{ci}\}$. The part filters all have the same 6×6 shape, but the root filters, both within a mixture model and across classes, have a variety of dimensions. Unlike [86] and [74] we decompose the root filters, not just the part filters. To do this, we employ 3×3 sparselets and pad the root filters with an extra one or two rows and columns, as needed, to ensure that their dimensions are multiples of 3. Summed over the models for all 20 object classes [30] in the PASCAL VOC 2007 dataset [23], there are a total of 4954 3×3 subfilters. In our experiments below, we represent *all* of these subfilters by sparse linear combinations of only 256 sparselets — effectively achieving more than an order of magnitude reduction in the number of model parameters. The HOG image features are 32-dimensional, leading to a sparselet size of $m = 288$. Our dictionary is thus undercomplete — which is desirable from a runtime perspective. Our experimental results confirm that the sparselets spans a sufficient subspace to represent the subfilters in the 20 PASCAL classes (Sec. 3.6), as well as to generalize to previously unseen classes from the ImageNet dataset (Sec. 3.6). Our DPM sparselets are visualized in Fig. 3.4.

Latent SVM

The DPMs in [31] are learned by optimizing a latent SVM (LSVM):

$$\begin{aligned} \mathbf{w}^* = &\operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \\ &\frac{1}{M}\sum_{i=1}^{M} \max\left(0, 1 - y_i \max_{z \in \mathcal{Z}(x_i)} \mathbf{w}^\intercal \mathbf{\Phi}(x_i, z)\right). \end{aligned} \tag{3.18}$$

The objective function in Eq. 3.18 is not convex in $\mathbf{w}$ and in practice a local optimum is found by coordinate descent on an auxiliary function that upper bounds Eq. 3.18 (see [31] for details). The coordinate descent algorithm alternates between two steps. In the first step, the set $\mathcal{Z}(x_i)$ is made singleton — for each *positive* example — by setting its only member to be an optimal latent value assignment for example x_i. This step results in a convex optimization problem that has the same form as a structural SVM. It is therefore straightforward to apply discriminative activation learning to a LSVM: we follow the same coordinate descent scheme and apply the SSVM problem transformation from Sec. 3.4 to the LSVM's convex optimization subproblem.

Our implementation is based on the `voc-release4` source code from [30]. To optimize the transformed objective function Eq. 3.15 when $R(\boldsymbol{\alpha})$ is either $R_{\text{Lasso}}(\boldsymbol{\alpha})$ or $R_{\text{EN}}(\boldsymbol{\alpha})$, we modified the default stochastic subgradient descent (SGD) code to implement the truncated gradient descent update of Langford *et al.* [57]. This method achieves actual sparsity by shrinking parameters and then truncating small values every few SGD iterations.

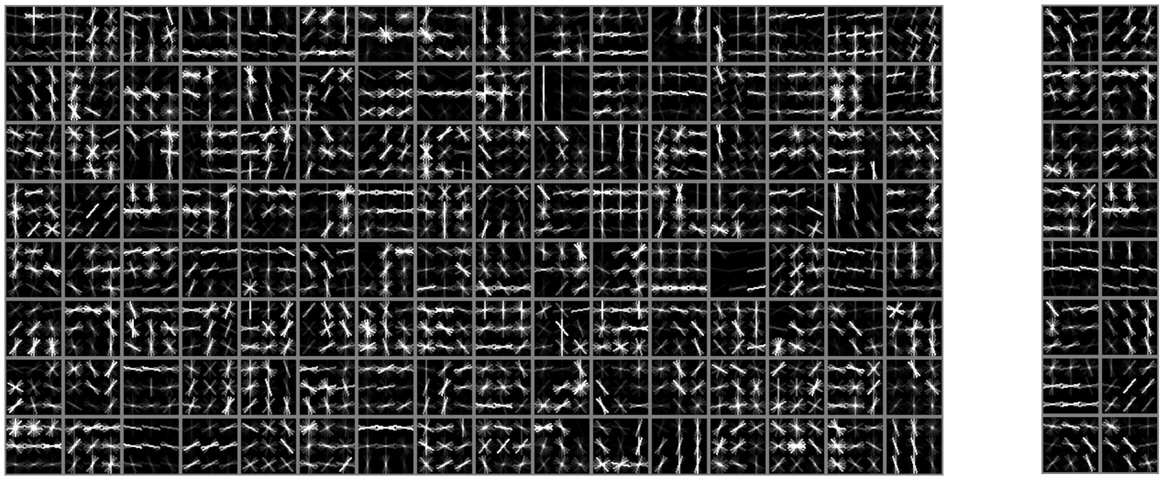

Figure 3.4: (Left) 128 of the 256 sparselets learned from 20 DPMs trained on the PASCAL VOC 2007 dataset. (Right) The top 16 sparselets activated for the motorbike category.

Visualizing learned DPM sparselets

Each DPM sparselet can be visualized as a 3×3 filter. In Fig. 3.4 (left) we show the positive weights of 128 of the 256 sparselets that we learned from DPMs for the 20 classes from the PASCAL VOC 2007 dataset. Regular structures, such as horizontal, vertical, and diagonal edges, as well as arcs and corners, are visible. We can order the sparselets activated for a particular category model by sorting them by the magnitude of their activation coefficients. Fig. 3.4 (right) shows the top 16 sparselets for the motorbike category. Some of the activated sparselets resemble circular fragments of wheels.

Image classification

To illustrate generalized sparselets applicability beyond DPMs, we evaluated our approach on the Caltech-101 [27] (102 classes, including background) and Caltech-256 (257 classes) [46] datasets. Since our aim is not state-of-the-art accuracy, but rather to demonstrate our learning method, we implemented sparselets atop a basic, publicly available image classification framework. Specifically, we used the `phow_caltech101` method included in VLFeat [93]. This approach trains one-against-all linear SVMs using bag-of-visual-words features (600 word vocabulary, 2×2 and 4×4 spatial pooling, and an approximate χ^2 feature map [94]). In Sec. 3.6 we experiment with two block sizes, $m \in \{100, 200\}$. These values of m lead to 36720 (or 18360) blocks in total for the 102 Caltech-101 classifiers, and 92520 (or 46260) blocks in total for the 257 Caltech-256 classifiers. We represent all of these blocks as sparse linear combinations of $d = 40$ sparselets.

3.6 Experiments

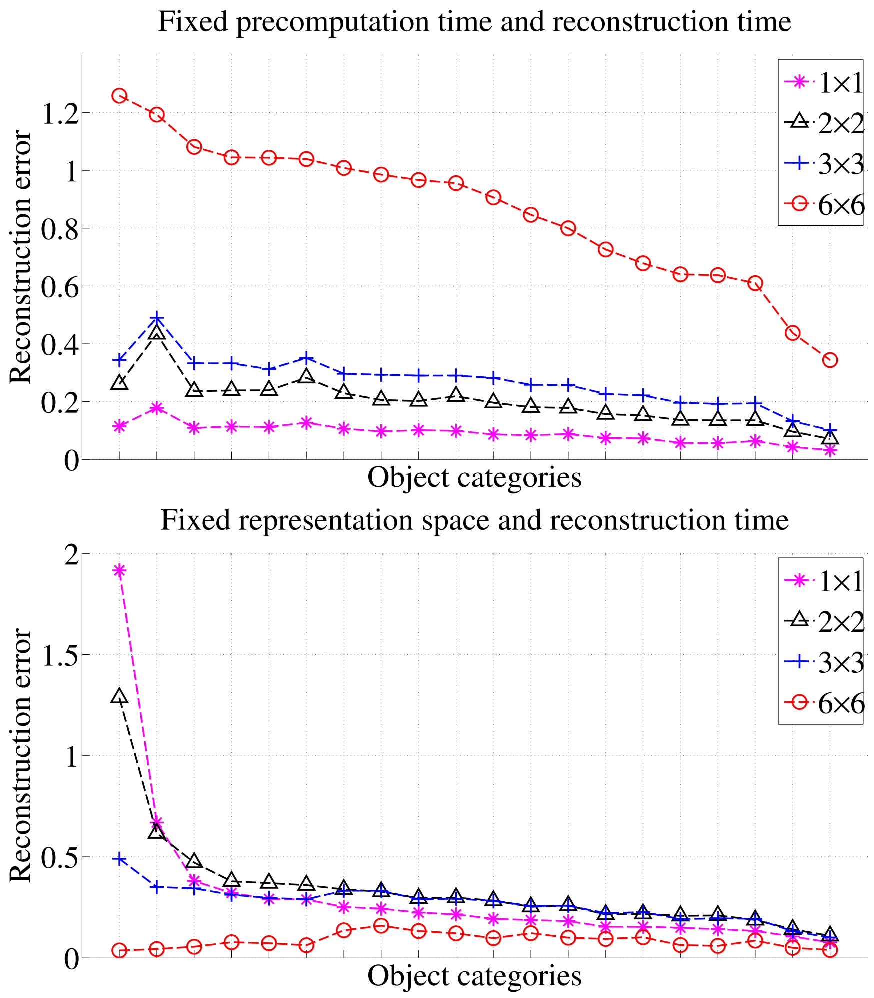

Figure 3.5: Reconstruction error for all 20 object categories from PASCAL 2007 dataset as sparselet parameters are varied. The precomputation time is fixed in the top figure and the representation space is fixed on the bottom. Object categories are sorted by the reconstruction error by 6 × 6 in the top figure and by 1 × 1 in the bottom figure.

We performed four sets of experiments, one with dictionary construction, one with model reconstruction, two with multiclass object detection, and the last one with multiclass image classification. The first experiment evaluates the sensitivity of the sparselet dictionary learned from random subset of object classes. The second experiment was designed to evaluate the effect of sparselet block size when the precomputation time is fixed versus when the representation space is fixed. The third experiment compares each of the regularization methods described in Sec. 3.4. The fourth experiment was designed to evaluate how well a set of sparselets learned on one set of models generalizes to a new set of models when learning the activation vectors in our discriminative framework.

Sensitivity to heterogeneity of object classes used in dictionary construction

To test how sensitive the learned dictionary of sparselets is with respect to the set of object classes used for the dictionary construction, we designed the following experiment on PASCAL VOC 2007 [23] dataset. For a test object class, we ran five trials of constructing the dictionary from five randomly chosen object models excluding the test object class (five different dictionaries per class). Empirically, (Table 3.1), a dictionary learned from randomly chosen subset of the object classes shows negligible loss in average precision compared to the dictionary learned from all the classes with insignificant standard deviation. This also shows the dictionary learned on a subset of object classes generalizes to previously unseen object classes.

Effect of different sparselet block sizes

In practice we can divide a part filter into smaller subfilters before computing the sparselet representation. The subfilter size (which equals the sparselet size) determines certain runtime and memory tradeoffs. Let F be a $h_F \times w_F \times l$ filter, and let the sparselet size be $h_s \times w_s \times l$. We require that h_s and w_s are divisors of h_F and w_F, respectively, and divide F into an $h_F/h_s \times w_F/w_s$ array of tiled subfilters. We approximate (or "reconstruct") a filter response by summing over approximate subfilter responses.

Given precomputed sparselet responses, reconstructing the response to F requires at most $\lambda_0(h_F/h_s)(w_F/w_s)$ operations. Low-cost approximation is essential, so we fix the reconstruction budget for F at $\lambda_0(h_F/h_s)(w_F/w_s) \leq B_R$. Within this budget, we can use fewer, smaller sparselets, or more, larger sparselets. We consider two other budget constraints. *Precomputation time* B_P: convolving an input with the entire sparselet dictionary requires $h_s w_s l d$ operations. For a fixed budget $h_s w_s l d \leq B_P$, we can use more, smaller sparselets, or fewer, larger sparselets. *Representation space* B_S: the space required to store the intermediate representation is proportional to the dictionary size d.

We evaluated the filter reconstruction errors for 4 different subfilter sizes. For these experiments we fixed the reconstruction budget $\lambda_0(h_F/h_s)(w_F/w_s) = 112$ by setting λ_0 to be $\{112, 28, 13, 3\}$ for subfilter size of $\{6\times6,\ 3\times3,\ 2\times2,\ 1\times1\}$. Fig. 3.5 (left) shows the result as the subfilter sizes vary while both the precomputation time and reconstruction time budget is fixed. We set the dictionary size $d = \{128, 512, 1152, 4608\}$ for $\{6 \times 6,\ 3 \times 3,\ 2 \times 2,\ 1 \times 1\}$ sized sparselets to fix the precomputation budget $h_s w_s d = 4608$.

For fixed reconstruction and precomputation budgets B_R and B_P, we studied the effect of varying sparselet size. Empirically, filter reconstruction error always decreases as we decrease sparselet size. When there are not too many classes, the precomputation time is not fully amortized and we would like to make B_P small. For a fixed, small B_P we minimize reconstruction error by setting h_s and w_s to small values as shown is Fig. 3.5 (top). However, as we make the sparselets smaller, d grows, possibly making the representation space budget B_S too large. In our experiments, we balance memory usage with sparselet size by setting h_s and w_s to 3.

When precomputation is amortized, minimizing precomputation time is less important. However, in this case we are still concerned with keeping the intermediate representation reasonably small. Fig. 3.5 (bottom) shows the results as the subfilter sizes vary while both the representation space and reconstruction time budget is fixed. We fixed the dictionary size $d = 512$ for this experiment. By fixing the response and representation space budgets, we observe that using more, larger sparselets minimizes reconstruction error (at the expense of requiring a larger precomputation budget) as shown in Fig. 3.5 (bottom).

Comparison of regularization methods

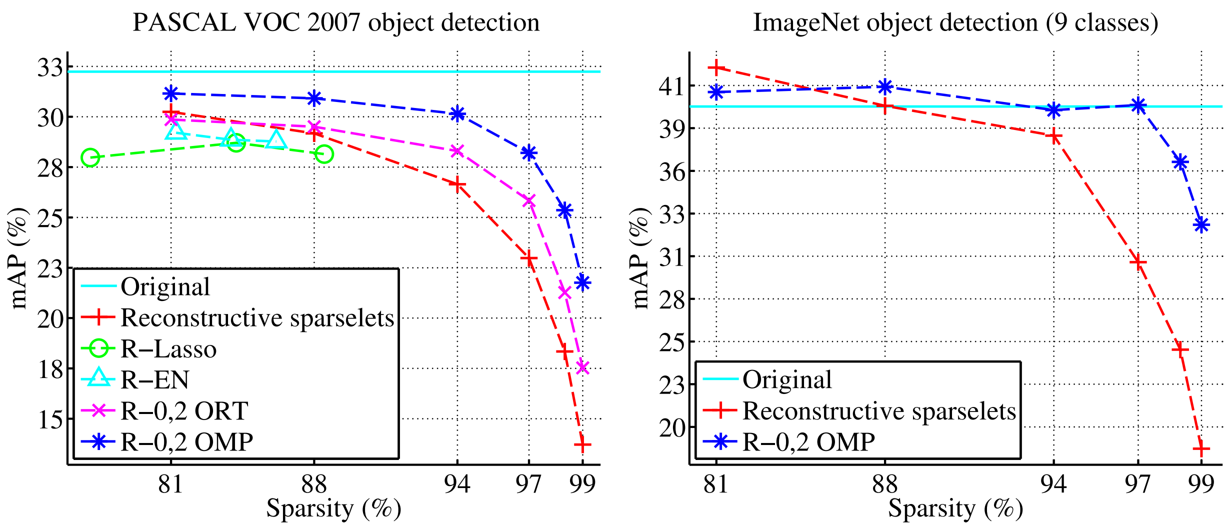

Figure 3.6: Mean average precision (mAP) *vs.* sparsity for object detection on the PASCAL 2007 dataset (left) and for 9 classes from ImageNet (right). The dictionary learned from the PASCAL detectors was used for the novel ImageNet classes. "Original" is the original linear model; "Reconstructive sparselets" is the baseline method from [86]; the remaining methods correspond to discriminative learning [41] with each of the regularizers described in Sec. 3.4.

We evaluated the reconstructive sparselets [86] and the discriminatively trained activation vectors [41] on the PASCAL VOC 2007 dataset [23]. Fig. 3.6 (left) shows the mean average precision (mAP) at various activation vector sparsity levels. We set the sparsity regularization constant λ_1 to $\{0.010, 0.015, 0.020\}$ for the lasso penalty ("R-Lasso") and to $\{0.025, 0.030, 0.035\}$ for the elastic net penalty ("R-EN"). For the combined ℓ_0 and ℓ_2 penalty, λ_0 was set to $\{48, 32, 16, 8, 4, 2\}$.

The ℓ_1-based regularization methods were very difficult to tune. Adjusting λ_1 to hit a desired sparsity level requires an expensive grid search. Additionally, the ratio between hinge-loss and the regularization term varied significantly between different classes, leading to a wide range of sparsity levels. Ultimately, these methods also underperformed in terms of mAP. Combined ℓ_0 and ℓ_2 regularization ("R–0,2 ORT" and "R–0,2 OMP"), in contrast, produces exactly the desired sparsity level and outperforms all other methods by a large margin. One interesting observation is that the mAP margin grows as the activation vectors become increasingly sparse.

Universality and generalization to previously unseen categories

To test the hypothesis that our learned dictionary of sparselets, in conjunction with the proposed discriminative activation training, are "universal" and generalize well, we used the sparselet dictionary learned from 20 PASCAL classes and evaluated detection performance on novel classes from the ImageNet [14] dataset. We selected 9 categories (sailboat, bread, cake, candle, fish, goat, jeep, scissors and tire) that have substantial appearance changes from the PASCAL classes. Fig. 3.6 (right) shows that our method generalizes well to novel classes and maintains competitive detection performance even in the high sparsity regime.

Image classification with generalized sparselets

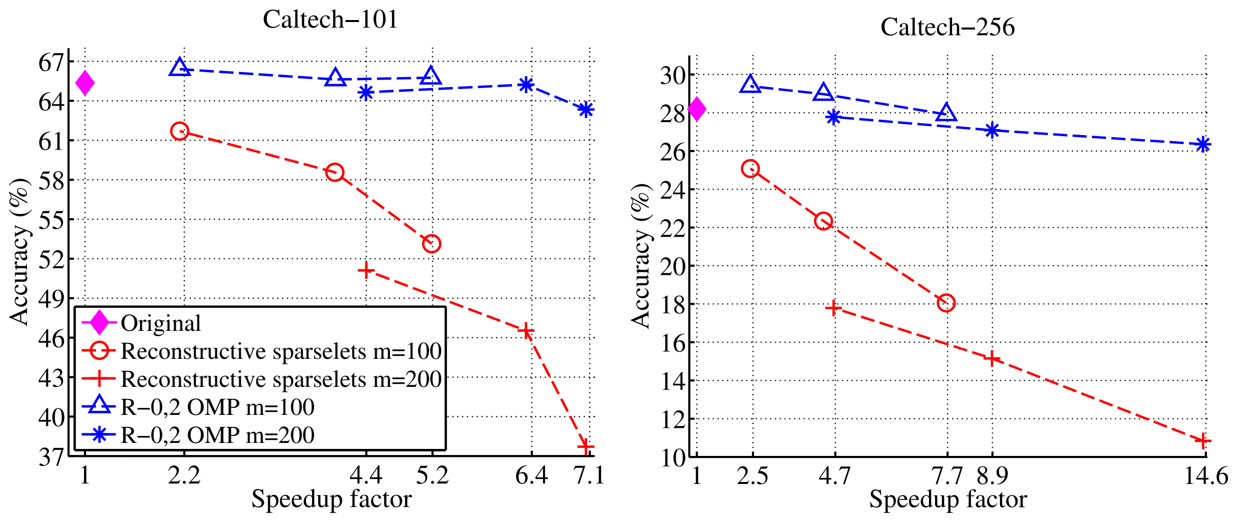

Figure 3.7: Average classification accuracy *vs.* speedup factor for Caltech-{101,256}.

Fig. 3.7 compares classification accuracy versus speedup factor (averaged over 6 machines with different CPU types). Generalized sparselets consistently provide a good speedup, however only the discriminatively trained sparselet activation models provide high accuracy,

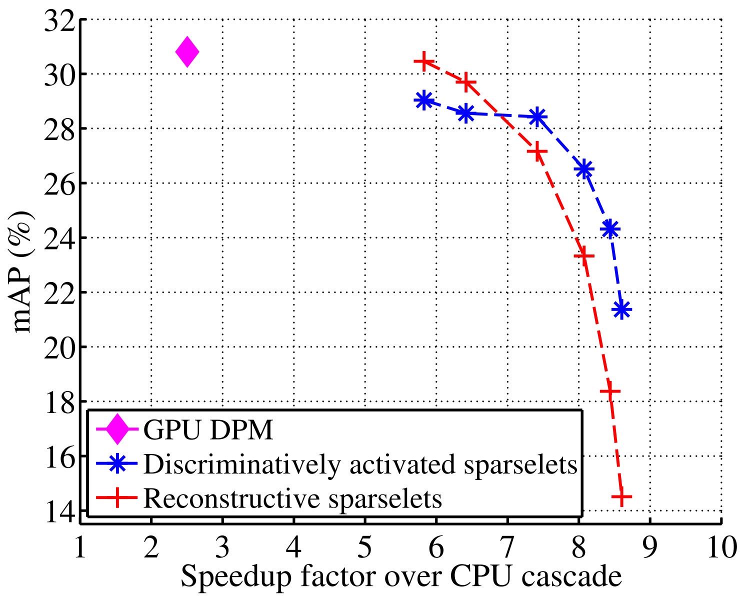

Figure 3.8: Run time comparison for DPM implementation on GPU, reconstructive sparselets and discriminatively activated sparselets in contrast to CPU cascade.

occasionally besting the original classifiers. In these experiments, we used a fixed dictionary size $d = 40$. We explored two block sizes $m = 100$ or 200. Each curve shows results at three sparsity levels: 0.6, 0.8, and 0.9. We trained and tested with 15 images per class on both datasets. As predicted by our cost analysis, increasing the class count (from 102 to 257) magnifies the speedup factor.

Run time experiments

We performed two sets of experiments to measure the wall clock runtime performance with and without GPU.

GPU experiment Fig. 3.8 shows the relative comparisons for DPM implementation on GPU, reconstructive sparselets and discriminatively activated sparselets. For sparselets, dictionary size K was set to 256 and the sparsity parameter λ_0 was varied in the following range $\{48, 32, 16, 8, 4, 2\}$ which corresponds to $\{81, 88, 94, 97, 98, 99\}\%$ sparsity respectively. As a reference for comparison, CPU cascade took 8.4547e+04 seconds to detection all 20 classes on all 4952 PASCAL VOC2007 test images (about 17 seconds per image).

In all the GPU experiments, detection thresholds were automatically adjusted at runtime via binary search to deliver 5000 ± 10 detections per object class, per frame. This was done to ensure sufficient detection coverage for approaching each object category's maximum-recall point, while limiting memory consumption and runtime of the non-maximum suppression algorithm to a known upper bound.

CPU cascade experiment was performed on a quad-core Intel Core i5-2400 CPU @ 3.10GHz with 8GB of RAM. GPU experiments were conducted on NVIDIA GeForce GTX 580 with 3GB of memory.

Single core CPU experiment On a single core CPU experiment (Intel Core i7) with 8GB memory, we compare our method against our efficient baseline implementation of DPM which utilizes SSE floating point SIMD instructions.

The sparsity level $\{81, 88, 94, 97, 98, 99\}\%$ resulted in $\{2.63, 3.99, 10.92, 15.81, 19.90, 22.57\}$ times speedup in filter convolution stage and $\{1.83, 2.25, 3.16, 3.39, 3.51, 3.54\}$ times speedup in end-to-end detection of 20 PASCAL classes per image. The variance of the experiments over test images were insignificant. The wall clock processing time for the baseline DPM code was 46.64 and 59.77 seconds per image per core respectively to evaluate the 20 class detectors. The end-to-end speedup factor for the cascade method with respect to the baseline DPM code was 3.04.

Even though the convolution stage is substantially accelerated via sparselets, other stages of the DPM framework (i.e. distance transform, zero padding filter responses) upper bounds the maximum possible end-to-end detection speedup (if the convolution stage and zero padding convolution responses takes 0 seconds) to be about 4X per core. However, our implementation of sparselets nearly reaches maximum possible speedup (up to 3.5X). In both GPU and CPU implementations, discriminatively activated sparselets significantly outperformed reconstructive sparselets in the high speedup, high sparsity regime.

For completeness, we plan to maintain a project webpage with demo videos, benchmark wall clock time results, and the source code at the following link:

http://www.eecs.berkeley.edu/∼song/sparselets/

3.7 Conclusion

We described an efficient object recognition model that simultaneously utilizes model redundancy and reconstruction sparsity to enable real-time multiclass object recognition. We also showed that the framework generalizes to any structured output prediction problem. The experimental result show that fixed number of sparselets learned from one dataset generalizes to novel objects from other datasets. This allows for reusing the pretrained sparselets with various other designs of activation vectors. In the future, we would like to design more flexible activation vectors to enable computational complexity logarithmic in number of object classes.

	Average Precision				
	Mean	**STD**	**Max**	**Min**	**Full Dict**
aeroplane	0.2922	0.0065	0.3024	0.2851	0.2966
bicycle	0.5542	0.0133	0.5747	0.5430	0.5688
bird	0.0690	0.0365	0.0948	0.0143	0.0343
boat	0.1236	0.0075	0.1352	0.1175	0.1394
bottle	0.1995	0.0104	0.2170	0.1903	0.2298
bus	0.4788	0.0129	0.4883	0.4581	0.5009
car	0.5479	0.0036	0.5526	0.5436	0.5640
cat	0.1296	0.0287	0.1631	0.0870	0.1432
chair	0.2008	0.0062	0.2075	0.1934	0.2057
cow	0.2226	0.0028	0.2258	0.2196	0.2384
diningtable	0.2136	0.0068	0.2215	0.2051	0.2381
dog	0.0574	0.0287	0.1054	0.0344	0.0590
horse	0.5408	0.0036	0.5442	0.5367	0.5542
motorbike	0.4611	0.0105	0.4721	0.4446	0.4724
person	0.3538	0.0080	0.3622	0.3459	0.3834
pottedplant	0.1048	0.0084	0.1163	0.0954	0.1127
sheep	0.1435	0.0148	0.1575	0.1236	0.1622
sofa	0.2940	0.0262	0.3144	0.2584	0.3191
train	0.4205	0.0102	0.4319	0.4039	0.4481
tvmonitor	0.3884	0.0059	0.3966	0.3826	0.3791

Table 3.1: Statistics of average precision for all 20 classes over five trials of constructing the dictionary from five randomly chosen classes (five different dictionaries per class). The last column (Full Dict) denotes the result when all 20 classes were used to construct the dictionary.

Chapter 4

On learning to localize objects with minimal supervision

4.1 Introduction

The classical paradigm for learning object detection models starts by annotating each object instance, in all training images, with a bounding box. However, this exhaustive labeling approach is costly and error prone for large-scale datasets. The massive amount of textually annotated visual data available online inspires a different, more challenging, research problem. Can weakly-labeled imagery, without bounding boxes, be used to reliably train object detectors?

In this alternative paradigm, the goal is to learn to localize objects with minimal supervision [96, 98]. We focus on the case where the learner has access to binary image labels that encode whether an image contains the target object or not, without access to any instance level annotations (i.e., bounding boxes).

Our approach starts by reducing the set of possible image locations that contain the object of interest from millions to thousands per image, using the selective search window proposal technique introduced by [92]. Then, we formulate a discriminative submodular cover algorithm to discover an initial set of image windows that are likely to contain the target object. After training a detection model with this initial set, we refine the detector using a novel smoothed formulation of latent SVM [2, 31]. We employ recently introduced object detection features, based on deep convolutional neural networks [21, 43], to represent the window proposals for clustering and detector training.

Compared to prior work on weakly-supervised detector training, we show substantial improvements on the standard evaluation metric (detection average precision on PASCAL VOC). Quantitatively, our approach achieves a 50% relative improvement in mean average precision over the current state-of-the-art for weakly-supervised learning.

4.2 Related work

Our work is related to three active research areas: (1) weakly-supervised learning, (2) unsupervised discovery of mid-level visual elements, and (3) co-segmentation.

We build on a number of previous approaches for training object detectors from weakly-labeled data. In nearly all cases, the task is formulated as a multiple instance learning (MIL) problem [62]. In this formulation, the learner has access to an image-level label indicating the presence or absence of the target class, but not its location (if it is present). The challenge faced by the learner is to find the sliver of signal present in the positive images, but absent from the negative images. The implicit assumption is that this signal will correspond to the positive class.

Although there have been recent works on convex relaxations [61, 48], most MIL algorithms start from an initialization and then perform some form of local optimization. Early efforts, such as [96, 98, 39, 34, 11, 10, 8], focused on datasets with strong object-in-the-center biases (e.g. Caltech-101). This simplified setting enabled clarity and focus on the MIL formulation, image features, and classifier design, but masked the vexing problem of finding a good initialization in data where such helpful biases are absent.

More recent work, such as [83, 82], attempts to learn detectors, or simply automatically generate bounding box annotations from much more challenging datasets such as PASCAL VOC [24]. In this data regime, focusing on initialization is crucial and carefully designed heuristics, such as shrinking bounding boxes [78], are often employed.

Recent literature on unsupervised mid-level visual element discovery [19, 80, 22, 50, 76] uses weak labels to discover visual elements that occur commonly in positive images but not in negative images. Discovered visual element representation were shown to successfully provide discriminative information in classifying images into scene types. The most recent work [18] presents a discriminative mode seeking formulation and draws connections between discovery and mean-shift algorithms [38].

The problem of finding common structure is related to the challenging setting of co-segmentation [77, 49, 1], which is the unsupervised segmentation of an object that is present in multiple images. While in this chapter we do not address pixel-level segmentation, we employ ideas from co-segmentation: the intuition behind our submodular cover framework in Section 4.4 is shared with CoSand [52]. Finally, submodular covering ideas have recently been applied to (active) filtering of hypothesis after running a detector, and without the discriminative flavor we propose [4, 9].

4.3 Problem formulation

Our goal is to learn a detector for a visual category from a set of images, each with a binary label. We model an image as a set of overlapping rectangular windows and follow a standard approach to detection: reduce the problem of detection to the problem of binary classification of image windows. However, at training time we are only given image-level labels, which leads to a classic multiple instance learning (MIL) problem. We can think of each image as a "bag" of instances (rectangular windows) and the binary image label $y = 1$ specifies that the bag contains at least one instance of the target category. The label $y = -1$ specifies that the image contains no instances of the category. During training, no instance labels are available.

MIL problems are typically solved (locally) by finding a local minimum of a non-convex objective function, such as MI-SVM [2]. In practice, the quality of the local solution depends heavily on the quality of the initialization. We therefore focus extensively on finding a good initialization. In Section 4.4, we develop an initialization method by formulating a discriminative set multicover problem that can be solved approximately with a greedy algorithm. This initialization, without further MIL refinement, already produces good object detectors, validating our approach. However, we can further improve these detectors by optimizing the MIL objective. We explore two alternative MIL objectives in Section 4.5. The first is the standard Latent SVM (equivalently MI-SVM) objective function, which can be optimized by coordinate descent on an auxiliary objective that upper-bounds the LSVM objective. The second method is a novel technique that smoothes the Latent SVM objective and can be solved more directly with unconstrained smooth optimization techniques, such as L-BFGS [70]. Our experimental results show modest improvements from our smoothed LSVM formulation on a variety of MIL datasets.

4.4 Finding objects via submodular cover

Learning with LSVM is a chicken and egg problem: The model weights are needed to infer latent annotations, but the latent annotations are needed to estimate the model weights. To initialize this process, we approximately identify jointly present objects in a weakly supervised manner. The experiments show a significant effect from this initialization. Our procedure implements two essential assumptions: (i) the correct boxes are similar, in an appropriate feature space, across positive images (or there are few modes), and (ii) the correct boxes do not occur in the negative images. In short, in the similarity graph of all boxes we seek dense subgraphs that only span the positive images. Finding such subgraphs is a nontrivial combinatorial optimization problem.

The problem of finding and encoding a jointly present signal in images is an old one, and has been addressed by clustering, minimum description length priors, and the concept of exemplar [13, 59, 67, 52]. These approaches share the idea that a small number of exemplars or clusters should well encode the shared information we are interested in. We formalize this intuition as a flexible *submodular cover* problem. However, we also have label information at hand that can help identify correct boxes. We therefore integrate into our covering framework the relevance for positively versus negatively labeled images, generalizing ideas from [19]. This combination allows us to find multiple modes of the object appearance distribution.

Let $\mathcal{P}$ be the set of all positive images. Each image contains a set $\mathcal{B}_I = \{b_1, \dots, b_m\}$ of candidate bounding boxes generated from selective search region proposals [92]. In practice,

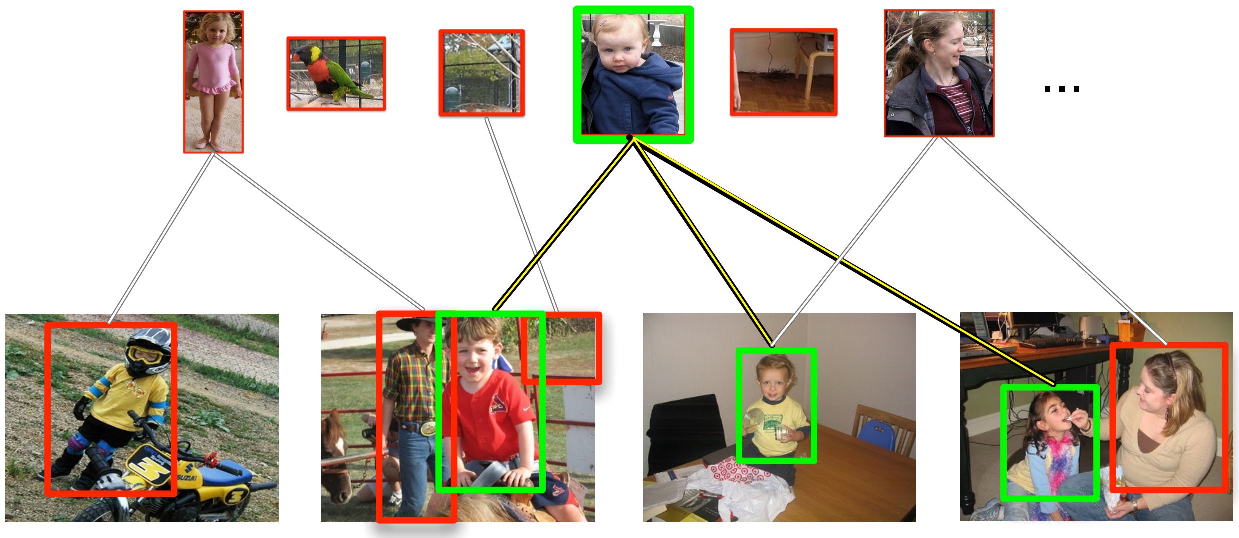

Figure 4.1: Illustration of the graph $\mathcal{G}$ with $\mathcal{V}$ (top row) and $\mathcal{U}$ (bottom row). Each box $b \in \mathcal{V}$ is connected to its closest neighbors from positive images (one from each image). Non-discriminative boxes occur in all images equally, and may not even have any boxes from positive images among their closest neighbors – and consequently no connections to $\mathcal{U}$. Picking the green-framed box v in $\mathcal{V}$ "covers" its (green) highlighted neighbors $\Gamma(b)$.

there are about 2000 region proposal boxes per image and about 5000 training images in the PASCAL VOC dataset. Ultimately, we will define a function $F(S)$ on sets S of boxes that measures how well the set S represents $\mathcal{P}$. For each box b, we find its nearest neighbor box in each (positive *and negative*) image. We sort the set $\mathcal{N}(b)$ of all such neighbors of b in increasing order by their distance to b. This can be done in parallel. We will define a graph using these nearest neighbors that allows us to optimize for a small set of boxes S that are (i) *relevant* (occur in many positive images); (ii) *discriminative* (dissimilar to the boxes in the negative images); and (iii) *complementary* (capture multiple modes).

We construct a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{U}, \mathcal{E})$ whose nodes $\mathcal{V}$ and $\mathcal{U}$ are all boxes occurring in $\mathcal{P}$ (each b occurs once in $\mathcal{V}$ and once in $\mathcal{U}$). The nodes in $\mathcal{U}$ are partitioned into groups $\mathcal{B}_I$: $\mathcal{B}_I$ contains all boxes from image $I \in \mathcal{P}$. The edges $\mathcal{E}$ are formed by connecting each node (box) $b \in \mathcal{V}$ to its top k neighbors in $\mathcal{N}(b) \subseteq \mathcal{U}$ from positive images. Figure 4.1 illustrates the graph. Connecting only to the top k neighbors (instead of all) implements discriminativeness: the neighbors must compete. If b occurs in positively and negatively labeled images equally, then many top-k closest neighbors in $\mathcal{N}(b)$ stem from negative images. Consequently, b will not be connected to many nodes (boxes from $\mathcal{P}$) in $\mathcal{G}$. We denote the neighborhood of a set of nodes $S \subseteq \mathcal{V}$ by $\Gamma(S) = \{b \in \mathcal{U} \mid \exists (v, b) \in \mathcal{E} \text{ with } v \in S\}$.

Let $S \subseteq \mathcal{V}$ denote a set of selected boxes. We define a *covering score* $\text{cov}_{I,t}(S)$ for each I that is determined by a *covering threshold* t and a scalar, nondecreasing concave function

$g : \mathbb{R}_+ \to \mathbb{R}_+$:

$$\mathrm{cov}_{I,t}(S) = g(\min\{t,\ |\Gamma(S) \cap \mathcal{B}_I|\}). \tag{4.1}$$

This score measures how many boxes in $\mathcal{B}_I$ are neighbors of S and thus "covered". We gain from covering up to t boxes from $\mathcal{B}_I$ – anything beyond that is considered redundant. The *total covering score* of a set $S \subseteq \mathcal{V}$ is then

$$F(S) = \sum\nolimits_{I \in \mathcal{P}} \mathrm{cov}_{I,t}(S). \tag{4.2}$$

The threshold t balances relevance and complementarity: let, for simplicity, $g = \mathrm{id}$. If $t = 1$, then a set that maximizes $\mathrm{cov}_{I,t}(S)$ contains boxes from many different images, and few from a single image. The selected neighborhoods are very complementary, but some of them may not be very relevant and cover outliers. If t is large, then any additionally covered box yields a gain, and the best boxes $b \in \mathcal{V}$ are those with the largest degree. A box has large degree if many of its closest neighbors in $\mathcal{N}(b)$ are from positive images. This also means b is discriminative and relevant for $\mathcal{P}$.

Lemma 1. *The function $F : 2^{\mathcal{V}} \to \mathbb{R}_+$ defined in Equation 4.2 is nondecreasing and submodular.*

A set function is *submodular* if it satisfies *diminishing marginal returns*: for all v and $S \subseteq T \subseteq \mathcal{V} \setminus \{v\}$, it holds that $F(S \cup \{v\}) - F(S) \geq F(T \cup \{v\}) - F(T)$.

Proof. First, the function $S \mapsto |\Gamma(S) \cap \mathcal{B}_I|$ is a covering function and thus submodular: let $S \subset T \subseteq \mathcal{V} \setminus b$. Then $\Gamma(S) \subseteq \Gamma(T)$ and therefore

$$\begin{aligned} |\Gamma(T \cup \{b\})| - |\Gamma(T)| &= |\Gamma(b) \setminus \Gamma(T)| && (4.3)\\ &\leq |\Gamma(b) \setminus \Gamma(S)| && (4.4)\\ &= |\Gamma(S \cup \{b\})| - |\Gamma(S)|. && (4.5) \end{aligned}$$

The same holds when intersecting with $\mathcal{B}_I$. Thus, $\mathrm{cov}_{t,I}(S)$ is a nondecreasing concave function of a submodular function and therefore submodular. Finally, F is a sum of submodular functions and hence also submodular. Monotonicity is obvious. □

We aim to select a representative subset $S \subseteq \mathcal{V}$ with minimum cardinality:

$$\min_{S \subseteq \mathcal{V}} |S| \quad \text{s.t.}\ \ F(S) \geq \alpha F(\mathcal{V}) \tag{4.6}$$

for $\alpha \in (0, 1]$. We optimize this via a greedy algorithm: let $S_0 = \emptyset$ and, in each step τ, add the node v that maximizes the marginal gain $F(S_\tau \cup \{v\}) - F(S_\tau)$.

Lemma 2. *The greedy algorithm solves Problem 4.6 within an approximation factor of* $1 + \log\left(\frac{kg(1)}{g(t)-g(t-1)}\right) = O(\log k)$.

Lemma 3 says that the algorithm returns a set $\widehat{S}$ with $F(\widehat{S}) \geq \alpha F(\mathcal{V})$ and $|\widehat{S}| \leq O(\log k)|S^*|$, where S^* is an optimal solution. This result follows from the analysis by [101] (Thm. 1) adapted to our setting. To get a better intuition for the formulation 4.6 we list some special cases:

Min-cost cover. With $t = 1$ and $g(a) = a$ being the identity, Problem 4.6 becomes a min-cost cover problem. Such straightforward covering formulations have been used for filtering after running a detector [4].

Maximum relevance. A minimum-cost cover merely focuses on complementarity of the selected nodes S, which may include rare outliers. At the other extreme (t large), we would merely select by the number of neighbors ([19] choose one single $\mathcal{N}(b)$ that way).

Multi-cover. To smoothly move between the two extremes, one may choose $t > 1$ and g to be sub-linear. This trades off representation, relevance, and discriminativeness.

In Figure 5.5, we visualize top 5 nearest neighbors with positive labels in the first chosen cluster S_1 for all 20 classes on the PASCAL VOC data. Our experiments in Section 4.6 show the benefits of our framework. Potentially, the results might improve even further when using the complementary mode shifts of [18] as a pre-selection step before covering.

4.5 Iterative refinement with latent variables

In this section, we review the latent SVM formulation, and we propose a simple smoothing technique enabling us to use classical techniques for unconstrained smooth optimization. Figure 4.3 illustrates our multiple instance learning analogy for object detection with one-bit labels.

Review of latent SVM

For a binary classification problem, the latent SVM formulation consists of learning a decision function involving a maximization step over a discrete set of configurations $\mathcal{Z}$. Given a data point $\mathbf{x}$ in $\mathbb{R}^p$ that we want to classify, and some learned model parameters $\mathbf{w}$ in $\mathbb{R}^d$, we select a label y in $\{-1, +1\}$ as follows:

$$y = \operatorname{sign}\left(\max_{\mathbf{z}\in\mathcal{Z}} \mathbf{w}^\intercal \phi(\mathbf{x}, \mathbf{z})\right), \tag{4.7}$$

where $\mathbf{z}$ is called a "latent variable" chosen among the set $\mathcal{Z}$. For object detection, $\mathcal{Z}$ is typically a set of bounding boxes, and maximizing over $\mathcal{Z}$ amounts to finding a bounding box containing the object. In deformable part models [31], the set $\mathcal{Z}$ contains all possible part configurations, each part being associated to a position in the image. The resulting set $\mathcal{Z}$ has exponential size, but (4.7) can be solved efficiently with dynamic programming techniques for particular choices of ϕ.

Learning the model parameters $\mathbf{w}$ is more involved than solving a simple SVM problem. We are given some training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where the vectors $\mathbf{x}_i$ are in $\mathbb{R}^p$ and the scalars

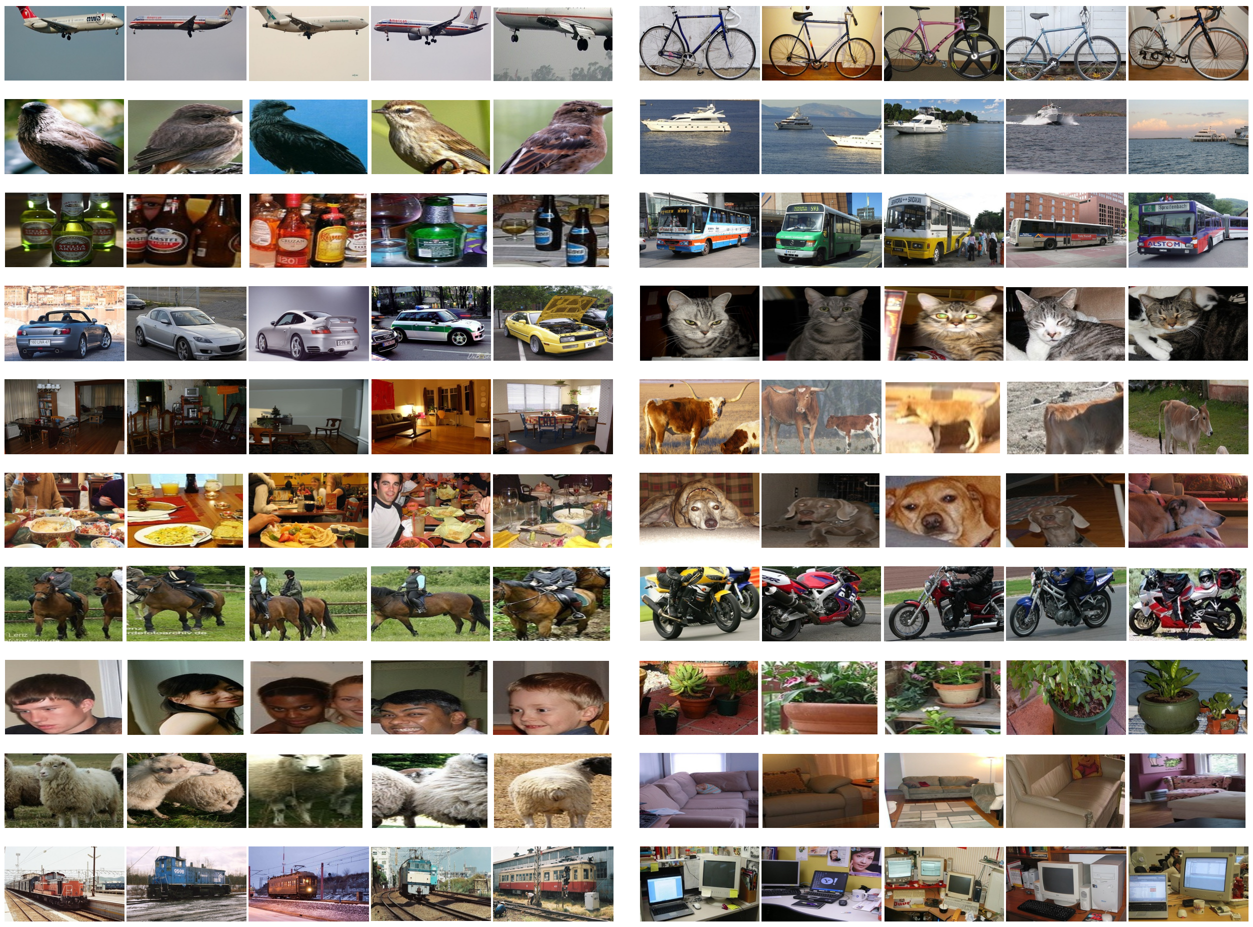
Figure 4.2: Visualizations of top 5 nearest neighbor proposal boxes with positive labels in the first cluster, S_1 for all 20 classes in PASCAL VOC dataset. From left to right, aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, plant, sheep, sofa, train, and tvmonitor.

y_i are binary labels in $\{1, -1\}$. Then, the latent SVM formulation becomes

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \ell\left(y_i,\ \max_{\mathbf{z}\in\mathcal{Z}} \mathbf{w}^\intercal \phi(\mathbf{x}_i, \mathbf{z})\right), \tag{4.8}$$

where $\ell : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is the hinge loss defined as $\ell(y,\hat{y}) = \max(0, 1 - y\hat{y})$, which encourages the decision function for each training example to be the same as the corresponding label. Similarly, other loss functions can be used such as the logistic or squared hinge loss.

Problem (4.8) is nonconvex and nonsmooth, making it hard to tackle. A classical technique to obtain an approximate solution is to use a difference of convex (DC) programming technique, called concave-convex procedure [103, 102]. We remark that the part of (4.8) corresponding to negative examples is convex with respect to $\mathbf{w}$. It is indeed easy to show that

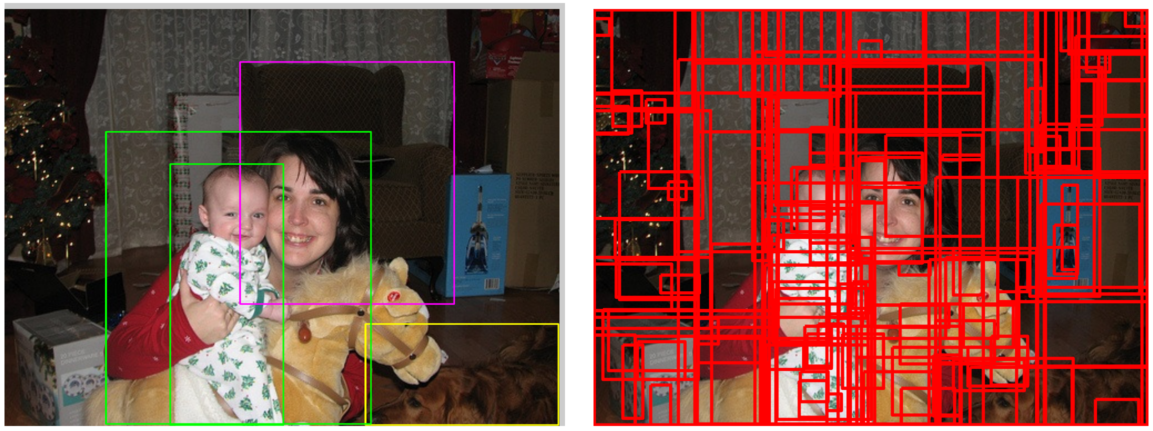

Figure 4.3: In the refinement stage, we formulate a multiple instance learning bag per image and bag instances correspond to each window proposals from selective search. Binary bag labels correspond to image-level annotations of whether the target object exists in the image or not. (Left) ground truth bounding boxes color coded with category labels. green: person, yellow: dog, and magenta: sofa, (Right) visualization of 100 random subset of window proposals.

each corresponding term can be written as a pointwise maximum of convex functions, and is thus convex [see 6]: when $y_i = -1$, $\ell\left(y_i, \max_{\mathbf{z}\in\mathcal{Z}} \mathbf{w}^\intercal \phi(\mathbf{x}_i, \mathbf{z})\right) = \max_{\mathbf{z}\in\mathcal{Z}} \ell(y_i, \mathbf{w}^\intercal \phi(\mathbf{x}_i, \mathbf{x}))$. On the other hand, the part corresponding to positive examples is concave, making the objective (4.8) suitable to DC programming. Even though such a procedure does not have any theoretical guarantee about the quality of the optimization, it monotonically decreases the value of the objective and performs relatively well when the problem is well initialized [31].

We propose a *smooth formulation* of latent SVM, with two main motives. First, smoothing the objective function of latent SVM allows the use of efficient second-order optimization algorithms such as quasi-Newton [70] that can leverage curvature information to speed up convergence. Second, as we show later, smoothing the latent SVM boils down to considering the top-N configurations in the maximization step in place of the top-1 configuration in the regular latent SVM. As a result, the smooth latent SVM training becomes more robust to unreliable configurations in the early stages, since a larger set of plausible configurations is considered at each maximization step.

Smooth formulation of LSVM

In the objective (4.8), the hinge loss can be easily replaced by a smooth alternative, e.g., squared hinge, or logistic loss. However, the non-smooth points induced by the following functions are more difficult to handle

$$f_{\mathbf{x}_i}(\mathbf{w}) := \max_{\mathbf{z}\in\mathcal{Z}} \mathbf{w}^\intercal \phi(\mathbf{x}_i, \mathbf{z}). \tag{4.9}$$

We propose to use a smoothing technique studied by [69] for convex functions.

Nesterov's smoothing technique We only recall here the simpler form of Nesterov's results that is relevant for our purpose. Consider a non-smooth function that can be written in the following form:

$$g(\mathbf{w}) := \max_{\mathbf{u}\in\Delta} \langle \mathbf{A}\mathbf{w}, \mathbf{u}\rangle \,, \tag{4.10}$$

where $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{A}$ is in $\mathbb{R}^{m\times d}$, and Δ denotes the probability simplex, $\Delta = \{\mathbf{x} : \sum_{i=1}^m x_i = 1, x_i \geq 0\}$. Smoothing here consists of adding a strongly convex function ω in the maximization problem

$$g_\mu(\mathbf{w}) := \max_{\mathbf{u}\in\Delta} \Big[\langle \mathbf{A}\mathbf{w}, \mathbf{u}\rangle - \frac{\mu}{2}\omega(\mathbf{u}) \Big] \,. \tag{4.11}$$

The resulting function g_μ is differentiable for all $\mu > 0$, and its gradient is

$$\nabla g_\mu(\mathbf{w}) = \mathbf{A}^\intercal \mathbf{u}^\star(\mathbf{w}), \tag{4.12}$$

where $\mathbf{u}^\star(\mathbf{w})$ is the unique solution of (4.11). The parameter μ controls the amount of smoothing. Clearly, $g_\mu(\mathbf{w}) \to g(\mathbf{w})$ for all $\mathbf{w} \in \mathcal{W}$ as $\mu \to 0$. As [69] shows, for a given target approximation accuracy ϵ, there is an optimal amount of smoothing $\mu(\epsilon)$ that can be derived from a convex optimization perspective using the strong convexity parameter of $\omega(\cdot)$ on Δ and the (usually unknown) Lipschitz constant of g. In the experiments, we shall simply learn the parameter μ from data.

Smoothing the latent SVM We now apply Nesterov's smoothing technique to the latent SVM objective function. As we shall see, the smoothed objective takes a simple form, which can be efficiently computed in the latent SVM framework. Furthermore, smoothing latent SVM implicitly models uncertainty in the selection of the best configuration $\mathbf{z}$ in $\mathcal{Z}$, as shown by [56] for a different smoothing scheme.

In order to smooth the functions $f_{\mathbf{x}_i}$ defined in (4.9), we first notice that

$$f_{\mathbf{x}_i}(\mathbf{w}) = \max_{\mathbf{u}\in\Delta} \langle \mathbf{A}_{\mathbf{x}_i}\mathbf{w}, \mathbf{u}\rangle, \tag{4.13}$$

where $\mathbf{A}_{\mathbf{x}_i}$ is a matrix of size $|\mathcal{Z}| \times d$ such that the j-th row of $\mathbf{A}_{\mathbf{x}_i}$ is the feature vector $\phi(\mathbf{x}_i, \mathbf{z}_j)$ and $\mathbf{z}_j$ is the j-th element of $\mathcal{Z}$. Considering any strongly convex function ω and parameter $\mu > 0$, the smoothed latent SVM objective is obtained by replacing in (4.8)

- the functions $f_{\mathbf{x}_i}$ by their smoothed counterparts $f_{\mathbf{x}_i,\mu}$ obtained by applying (4.11) to (4.13);
- the non-smooth hinge-loss function l by any smooth loss.

Objective and gradient evaluations An important issue remains the computational tractability of the new formulation in terms of objective and gradient evaluations, in order to use quasi-Newton optimization techniques. The choice of the strongly convex function ω is crucial in this respect.

There are two functions known to be strongly convex on the simplex: i) the Euclidean norm, ii) the entropy. In the case of the Euclidean-norm $\omega(\mathbf{u}) = \|\mathbf{u}\|_2^2$, it turns out that the smoothed counterpart can be efficiently computed using a projection on the simplex, as shown below.

$$\mathbf{u}^\star(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{u}\in\Delta} \left\| \frac{1}{\mu}\mathbf{A}\mathbf{w} - \mathbf{u} \right\|_2^2, \tag{4.14}$$

where $\mathbf{u}^\star(\mathbf{w})$ is the solution of (4.11). Computing $\mathbf{A}\mathbf{w}$ requires a priori $O(|\mathcal{Z}|d)$ operations. The projection can be computed in $O(|\mathcal{Z}|)$ [see, e.g., 3]. Once $\mathbf{u}^\star$ is obtained, computing the gradient requires $O(d\|\mathbf{u}^\star\|_0)$ operations, where $\|\mathbf{u}^\star\|_0$ is the number of non-zero entries in $\mathbf{u}^\star$.

When the set $\mathcal{Z}$ is large, these complexities can be improved by leveraging two properties. First, the projection on the simplex is known to produce sparse solutions, the smoothing parameter μ controlling the sparsity of $\mathbf{u}^\star$; second, the projection preserves the order of the variables. As a result, the following heuristic can be justified. Assume that for some $N < |\mathcal{Z}|$, we can obtain the top-N entries of $\mathbf{A}\mathbf{w}$ without exhaustively exploring $\mathcal{Z}$. Then, performing the projection on these reduced set of N variables yields a vector $\mathbf{u}'$ which can be shown to be optimal for the original problem (4.14) whenever $\|\mathbf{u}'\|_0 < N$. In other words, whenever N is large enough and μ small enough, computing the gradient of $f_{\mathbf{x}_i,\mu}$ can be done in $O(Nd)$ operations. We use this heuristic in all our experiments.

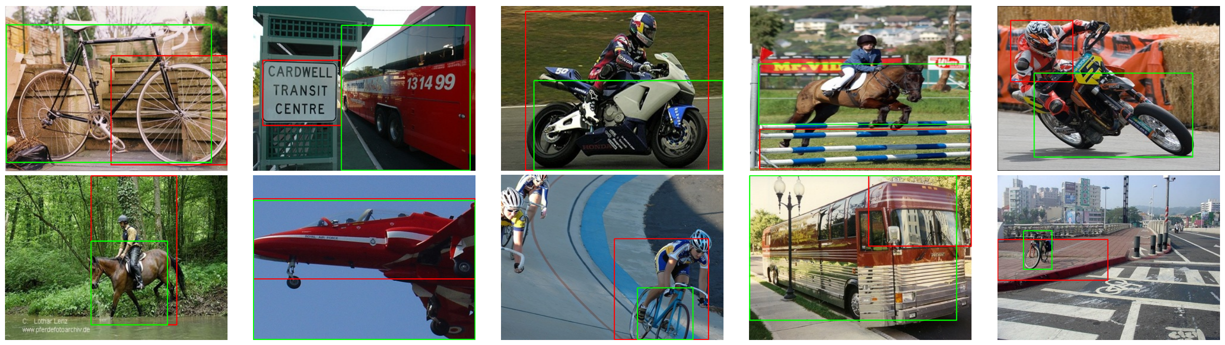

Figure 4.4: Visualization of some common failure cases of constructed positive windows by[82] vs our method. Red bounding boxes are constructed positive windows from [82]. Green bounding boxes are constructed positive windows from our method.

4.6 Experiments

We performed two sets of experiments, one on a multiple instance learning dataset [2] and the other on the PASCAL VOC 2007 data [23]. The first experiment was designed to com-

Dataset	LSVM w/o bias	SLSVM w/o bias	LSVM w/ bias	SLSVM w/ bias
musk1	70.8 ± 14.4	80.3 ± 10.3	81.7 ± 14.5	79.2 ± 13.4
musk2	51.0 ± 10.9	79.5 ± 10.4	80.5 ± 9.9	84.3 ± 11.4
fox	51.5 ± 7.5	63.0 ± 11.8	57.0 ± 8.9	61.0 ± 12.6
elephant	81.5 ± 6.3	88.0 ± 6.7	81.5 ± 4.1	87.0 ± 6.3
tiger	79.5 ± 8.6	85.5 ± 6.4	86.0 ± 9.1	87.5 ± 7.9
trec1	94.3 ± 2.9	95.5 ± 2.6	95.3 ± 3.0	95.3 ± 2.8
trec2	69.0 ± 6.8	83.0 ± 6.5	86.5 ± 5.7	83.8 ± 7.4
trec3	77.5 ± 5.8	90.0 ± 5.8	85.5 ± 6.3	86.0 ± 6.5
trec4	77.3 ± 8.0	85.0 ± 5.1	85.3 ± 3.6	86.3 ± 5.2
trec7	74.5 ± 9.8	83.8 ± 4.0	82.5 ± 7.0	81.5 ± 5.8
trec9	66.8 ± 5.0	70.3 ± 5.7	68.8 ± 8.0	71.5 ± 6.4
trec10	71.0 ± 9.9	84.3 ± 5.4	80.8 ± 6.6	82.8 ± 7.3

Table 4.1: 10 fold average and standard deviation of the test accuracy on MIL dataset. The two methods start from the same initialization introduced in [2]

Method	aeroplane		bicycle		boat		bus		horse		motorbike		mAP
	left	right	left	right	left	right	left	right	left	right	left	right	
[16]	9.1	23.6	33.4	49.4	0.0	0.0	0.0	16.4	9.6	9.1	20.9	16.1	16.0
[72]	7.5	21.1	38.5	44.8	0.3	0.5	0.0	0.3	45.9	17.3	43.8	27.2	20.8
[17]	5.3	18.1	48.6	61.6	0.0	0.0	0.0	16.4	29.1	14.1	47.7	16.2	21.4
[78]	30.8		25.0		3.6		26.0		21.3		29.9		22.8
[82] with our features	23.2		15.4		5.1		2.0		6.2		17.4		11.6
Cover + SVM	23.4		43.5		8.1		33.9		24.7		40.2		29.0
Cover + LSVM	28.2		47.2		9.6		34.7		25.2		39.8		30.8

Table 4.2: Detection average precision (%) on PASCAL VOC 2007-6x2 test set. First three baseline methods report results limited to left and right subcategories of the objects.

VOC2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	pson	plant	sheep	sofa	train	tv	mAP
[83]	13.4	44.0	3.1	3.1	0.0	31.2	43.9	7.1	0.1	9.3	9.9	1.5	29.4	38.3	4.6	0.1	0.4	3.8	34.2	0.0	13.9
Cover + SVM	23.4	43.5	22.4	8.1	6.2	33.9	33.8	30.4	0.1	17.9	11.5	17.1	24.7	40.2	2.4	14.8	21.4	15.1	31.9	6.2	20.3
Cover + LSVM	28.2	47.2	17.6	9.6	6.5	34.7	35.5	31.5	0.3	21.7	13.2	20.7	25.2	39.8	12.6	18.6	21.2	18.6	31.7	10.2	22.2
Cover + SLSVM	27.6	41.9	19.7	9.1	10.4	35.8	39.1	33.6	0.6	20.9	10.0	27.7	29.4	39.2	9.1	19.3	20.5	17.1	35.6	7.1	22.7

Table 4.3: Detection average precision (%) on full PASCAL VOC 2007 test set.

pare the multiple instance learning bag classification performance of LSVM with Smooth LSVM (SLSVM). The second experiment evaluates detection accuracy (measured in average precision) of our framework in comparison to baselines.

Multiple instance learning datasets

We evaluated our method in Section 5 on standard multiple instance learning datasets [2]. For preprocessing, we centered each feature dimension and ℓ_2 normalize the data. For fair comparison with [2], we use the same initialization, where the initial weight vector is obtained by training an SVM with all the negative instances and bag-averaged positive instances. For this experiment, we performed 10 fold cross validation on C and μ. Table 4.1 shows the experimental results. Without the bias, our method significantly performs better than LSVM method and with the bias, our method shows modest improvement in most cases.

Weakly-supervised object detection

To implement our weakly-supervised detection system we need suitable image features for computing the nearest neighbors of each image window in Section 4.4 and for learning object detectors. We use the recently proposed R-CNN [43] detection framework to compute features on image windows in both cases. Specifically, we use the convolutional neural network (CNN) distributed with DeCAF [21], which is trained on the ImageNet ILSVRC 2012 dataset (using only image-level annotations). We avoid using the better performing CNN that is fine-tuned on PASCAL data, as described in [43], because fine-tuning requires instance-level annotations.

We report detection accuracy as average precision on the standard benchmark dataset for object detection, PASCAL VOC 2007 *test* [23]. We compare to five different baseline methods that learn object detectors with limited annotations. Note that other baseline methods use additional information besides the one-bit image-level annotations. [16, 17] use a set of 799 images with bounding box annotations as meta-training data. In addition to bounding box annotations, [16, 17, 72] use extra instance level annotations such as *pose*, *difficult* and *truncated*. [82, 78] use *difficult* instance annotations but not *pose* or *truncated*. First, we report the detection average precision on 6 subsets of classes in table 4.2 to compare with [16, 17, 72].

To evaluate the efficacy of our initialization, we compare it to the state-of-the-art algorithm recently proposed by [82]. Their method constructs a set of positive windows by looping over each positive image and picking the instance that has the maximum distance to its nearest neighbor over all negative instances (and thus the name negative data *mining* algorithm). For a fair comparison, we used the same window proposals, the same features [43], the same L2 distance metric, and the same PASCAL 2007 detection evaluation criteria. The class mean average precision for the mining algorithm was 11.6% compared to 29.0% obtained by our initialization procedure. Figure 4.4 visualizes some command failure modes in our implementation of [82]. Since the negative mining method does not take into account the similarity among positive windows (in contrast to our method) our intuition is that the method is less robust to intra-class variations and background clutter. Therefore, it often latches onto background objects (i.e. hurdle in horse images, street signs in bus images), onto parts of the full objects (i.e. wheels of bicycles), or merges two different objects (i.e. rider and motorcycle). It is worth noting that [72, 82] use the CorLoc metric[1] as the evaluation metric to report results on PASCAL *test* set. In contrast, in our experiments, we exactly follow the PASCAL VOC evaluation protocol (and use the PASCAL VOC devkit scoring software) and report detection average precision.

Table 5.1 shows the detection result on the full PASCAL 2007 dataset. There are two baseline methods [83, 78] which report the result on the full dataset. Unfortunately, we were not able to obtain the per-class average precision data from the authors of [78] except the class mean average precision (mAP) of 15.0%. As shown in Table 5.1, the initial detector model trained from the constructed set of positive windows already produces good object detectors but we can provide further improvement by optimizing the MIL objective.

4.7 Conclusion

We developed a framework for learning to localize objects with one-bit object presence labels. Our results show that the proposed framework can construct a set of positive windows to train initial detection models and improve the models with the refinement optimization method. We achieve state-of-the-art performance for object detection with minimal supervision on the standard benchmark object detection dataset. Source code will be available on the author's website.

[1]CorLoc was proposed by [16] to evaluate the detection results on PASCAL *train* set

Chapter 5

Weakly-supervised discovery of visual pattern configurations

5.1 Introduction

The growing amount of sparsely and noisily labeled image data promotes the need for robustly learning detection methods that can cope with a minimal amount of supervision. A prominent example of this scenario is the abundant availability of labels at the image level (i.e., whether a certain object is present in the image or not), while detailed annotations of the exact location of the object are tedious and expensive and, consequently, scarce. Learning methods that handle image-level labels circumvent the need for such detailed annotations and therefore have the potential to effectively utilize the massive and ever-growing textually annotated visual data available on the Web. In addition, such weakly supervised methods can be more robust than fully supervised ones if the detailed annotations are noisy or ill-defined.

Motivated by these developments, recent work has explored learning methods that decreasingly rely on strong supervision. Early ideas for weakly supervised detection [97, 33] paved the way by successfully learning part-based object models, albeit on simple object-centric datasets (e.g., Caltech-101). Since then, a number of approaches [72, 82, 87] have attempted to learn models on more realistic and challenging data sets that feature large intra-category appearance variations and background clutter. To cope with those difficulties, these methods typically generate multiple candidate regions and retain the ones that occur most frequently in the positively labeled images. However, due to intra-category variations and deformations, the identified (single) patches often correspond to only a part of the object, such as a human face instead of the entire body. Such mislocalizations are a frequent problem for weakly supervised detection methods. Figure 5.4 illustrates some examples.

Mislocalization and too large or too small bounding boxes are problematic in two respects. First, they obviously affect the accuracy of the detection method. Detection is commonly phrased as multiple instance learning and addressed with non-convex optimization methods that alternatingly guess the location of the objects as positive examples (since the true location is unknown) and train a detector based on those guesses. Such methods are therefore heavily influenced by good initial localizations. Second, a common approach is to train the detector in stages, while adding informative "hard" negative examples to the training data. If we are not given accurate true object localizations in the training data, these hard examples must be derived from the detections identified in earlier rounds, and these initial detections may only use image-level annotations. The higher the accuracy of the initial localizations, the more informative is the augmented training data – and this is key to the accuracy of the final learned model.

In this work, we address the issue of mislocalizations by identifying characteristic, discriminative *configurations* of multiple patches (rather than a single one). This part-based approach is motivated by the observation that automatically discovered single "discriminative" patches often correspond to object parts. In addition, wrong background patches (e.g., patches of water or sky) occur throughout the positive images, but do not re-occur in typical configurations. In particular, we propose an effective method that takes as input a set of images with labels of the form "the object is present"/"not present", and automatically identifies characteristic part configurations of the given object.

To identify such co-occurrences, we use two main criteria. First, useful patches are *discriminative*, i.e., they occur in many positively labeled images, but not in the negatively labeled ones. To identify such patches, we use a discriminative covering formulation similar to the previous chapter. However, our end goal is to discover multiple patches in each image that represent different parts, i.e., they may be close but may not be overlapping too much. In covering formulations, one may discourage overlaps by saying that for two overlapping regions, one "covers" the other, i.e., they are treated as identical. But this is a transitive relation, and the density of possible regions in detection would imply that all regions are identical, strongly discouraging the selection of more than one part per image. Partial covers face the challenge of scale invariance. Hence, we take a different approach and formulate an independence constraint. This second criterion ensures that we select regions that may be close but non-redundant and not fully overlapping. We show that this constrained selection problem corresponds to maximizing a submodular function subject to a matroid intersection constraint, which leads to approximation algorithms with theoretical worst-case bounds. Given candidate parts identified by those two criteria, we effectively find frequently co-occurring configurations that take into account relative position, scale and viewpoint.

We demonstrate multiple benefits of the discovered configurations. First, we observe that combinations of patches can produce more accurate spatial coverage of the full object, especially when the most discriminative pattern corresponds to an object part. Second, any overlapping region between the co-occurring visual patterns is likely to cover a part (but not the full) of the object of interest (see intersecting regions between green and yellow boxes in Figure 5.5); thus, they can be used to generate very informative hard negatives for training, and those can reduce localization errors at test time.

In short, our main contribution is a novel weakly-supervised object detection method that automatically discovers frequent configurations of discriminative visual patterns and exploits them for training more robust object detectors. In our experiments on the challenging PASCAL VOC dataset, we find the inclusion of our discriminative, automatically detected configurations to outperform all state-of-the-art methods.

5.2 Related work

Weakly-supervised object detection. Training object detectors is usually done in a fully-supervised fashion using tight bounding box annotations that cover the object of interest (e.g., [29]). To reduce laborious bounding box annotation costs, recent weakly-supervised approaches [97, 33, 72, 82, 87] train detectors using binary object-presence labels without any object-location information.

Early efforts [97, 33] focused on simple datasets that have a single prominent object in each image (e.g., Caltech-101). More recent approaches [72, 82, 87] focus on the more challenging PASCAL dataset, which contains multiple objects in each image and large intra-category appearance variations. Of these, the previous chapter achieve state-of-the-art results by finding discriminative image patches that occur frequently in the positive images but rarely in the negative images using deep Convolutional Neural Network (CNN) features [55] and a submodular cover formulation. We use a similar approach to identify discriminative patches. But, contrary to the previous chapter who assume patches to contain entire objects, we allow patches to contain full objects or merely object *parts*. Thus, we aim to automatically piece together those patches to produce better full-object estimates. To this end, we augment the covering formulation and identify patches that are both representative and explicitly mutually different. We will see that this leads to more robust object estimates and further allows our system to intelligently select "hard negatives" (mislocalized objects), both of which improve detection performance.

Visual data mining. Existing approaches discover high-level object categories [85, 45, 25], mid-level patches [81, 19, 51], or low-level foreground features [58] by grouping similar visual patterns (i.e., images, patches, or contours) according to their texture, color, shape, etc. Recent methods [19, 51] use weakly-supervised labels to discover discriminative visual patterns. We use related ideas, but formulate the problem as a submodular optimization over matroids, which leads to approximation algorithms with theoretical worst-case guarantees. Covering formulations have also been used in [4, 9], but after running a trained object detector. An alternative discriminative approach, but less scalable than covering, uses spectral methods [107].

Modeling co-occurring visual patterns. Modeling the spatial and geometric relationship between co-occurring visual patterns (objects or object-parts) often improves visual recognition performance [97, 33, 84, 75, 104, 58, 29, 26, 81, 60]. Co-occurring patterns are usually represented as doublets [81], higher-order constellations [97, 33] or star-shaped models [29]. Among these, our work is most inspired by [97, 33], which learns part-based models using only weak-supervision. However, we use more informative features and a different formulation, and show results on more difficult datasets. Our work is also related by [60], which discovers high-level object compositions ("visual phrases" [26]) using ground-truth bounding box annotations. In contrast to [60], we aim to discover part compositions to represent full objects and do so without any bounding box annotations.

5.3 Approach

Our goal is to find a discriminative set of parts or patches that co-occur in many of the positively labeled images in the same configuration. We address this goal in two steps. First, we find a set of patches that are discriminative, i.e., they tend to occur mainly in positive images. Second, we use an efficient approach to find co-occurring configurations of pairs of such patches. Our approach easily extends beyond pairs; for simplicity and to retain configurations that occur frequently enough, we here restrict ourselves to pairs.

Discriminative candidate patches. For identifying discriminative patches, we begin with a construction similar to that of the previous chapter. Let $\mathcal{P}$ be the set of positively labeled images. Each image I contains candidate boxes $\{b_{I,1}, \ldots, b_{I,m}\}$ found via selective search [92]. For each $b_{I,i}$, we find its closest matching neighbor $b_{I',j}$ in each other image I' (regardless of the image label). The K closest of those neighbors form the neighborhood $\mathcal{N}(b_{I,i})$, the remaining ones are discarded.

Discriminative patches will have neighborhoods mainly within images in $\mathcal{P}$, i.e., if $\mathcal{B}(\mathcal{P})$ is the set of all patches from images in $\mathcal{P}$, then $\mathcal{N}(b) \cap \mathcal{B}(\mathcal{P}) \approx K$. To identify a small, diverse and representative set of such patches, like the previous chapter, we construct a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, where both $\mathcal{U}$ and $\mathcal{V}$ contain copies of $\mathcal{B}(\mathcal{P})$. Each patch $b \in \mathcal{V}$ is connected to the copy of its nearest neighbors in $\mathcal{U}$ – these will be K or less, depending on whether the K nearest neighbors of b occur in $\mathcal{B}(\mathcal{P})$ or in negatively labeled images. The most representative patches maximize the covering function

$$F(S) = |\Gamma(S)|, \tag{5.1}$$

where $\Gamma(S) = \{u \in \mathcal{U} \mid (b, u) \in \mathcal{E} \text{ for some } b \in S\}$ is the neighborhood of $S \subseteq \mathcal{V}$ in the bipartite graph. Figure 5.1 shows a cartoon illustration. The function F is monotone and submodular, and the C maximizing elements (for a given C) can be selected greedily [68].

However, if we aim to find part configurations, we must select multiple, jointly informative patches per image. Patches selected to merely maximize coverage can still be redundant, since the most frequently occurring ones are often highly overlapping. A straightforward modification would be to treat highly overlapping patches as identical. This identification would still admit a submodular covering model as in Equation 5.1. But, in our case, the candidate patches are very densely packed in the image, and, by transitivity, we would have to make all of them identical. In consequence, this would completely rule out the selection of more than one patch in an image and thereby prohibit the discovery of any co-occurring configurations.

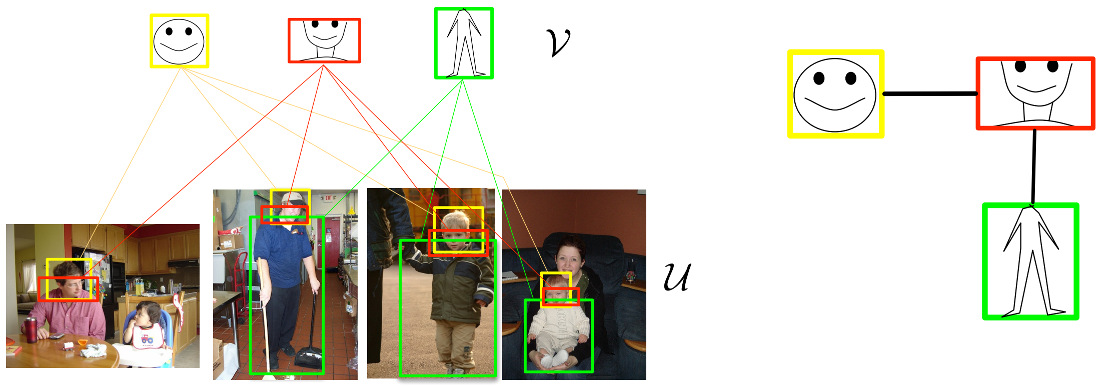

Figure 5.1: Left: bipartite graph $\mathcal{G}$ that defines the utility function F and identifies discriminativeness; right: graph $\mathcal{G}_C$ that defines the diversifying independence constraints $\mathcal{M}$. We may pick C_1 (yellow) and C_3 (green) together, but not C_2 (red) with any of those, since it is redundant. If we identify overlapping patches in $\mathcal{G}$ and thus the covering F, then we would only ever pick one of C_1, C_2 and C_3, and no characteristic configurations could be identified.

Therefore, we take a different approach, differing from the previous chapter whose goal is to identify single patches, and not part-based configurations. We constrain our selection such that no two patches $b, b' \in \mathcal{V}$ can be picked whose neighborhoods overlap by more than a fraction of θ. By overlap, we mean that the patches in the neighborhoods of b, b' overlap significantly (they need not be identical). This notion of diversity is reminiscent of NMS and similar to that in [19], but we here phrase and analyze it as a constrained submodular optimization problem. Our constraint can be expressed in terms of a different graph $\mathcal{G}_C = (\mathcal{V}, \mathcal{E}_C)$ with nodes $\mathcal{V}$. In $\mathcal{G}_C$, there is an edge between b and b' if their neighborhoods overlap prohibitively, as illustrated in Figure 5.1. Our family of feasible solutions is

$$\mathcal{M} = \{S \subseteq V \mid \forall\, b, b' \in S \text{ there is no edge } (b, b') \in \mathcal{E}_C\}. \tag{5.2}$$

In other words, $\mathcal{M}$ is the family of all independent sets in $\mathcal{G}_C$. We aim to maximize

$$\max_{S \subseteq \mathcal{V}} F(S) \quad \text{s.t. } S \in \mathcal{M}. \tag{5.3}$$

This problem is NP-hard. We solve it approximately via the following greedy algorithm. Begin with $S^0 = \emptyset$, and, in iteration t, add $b \in \operatorname{argmax}_{b \in \mathcal{V} \setminus S} |\Gamma(b) \setminus \Gamma(S^{t-1})|$. As we add b, we delete all of b's neighbors in $\mathcal{G}_C$ from $\mathcal{V}$. We continue until $\mathcal{V} = \emptyset$. If the neighborhoods $\Gamma(b)$ are disjoint, then this algorithm amounts to the following simplified scheme: we first sort all $b \in \mathcal{V}$ in non-increasing order by their degree $\Gamma(b)$, i.e., their number of neighbors in $\mathcal{B}(\mathcal{P})$, and visit them in this order. We always add the currently highest b in the list to S, then delete it from the list, and with it all its immediate (overlapping) neighbors. The following lemma states an approximation factor for the greedy algorithm, where Δ is the maximum degree of any node in $\mathcal{G}_C$.

Lemma 3. *The solution S_g returned by the greedy algorithm is a $1/(\Delta+2)$ approximation for Problem 5.2: $F(S_g) \geq \frac{1}{\Delta+2}F(S^*)$. If $\Gamma(b) \cap \Gamma(b') = \emptyset$ for all $b, b' \in \mathcal{V}$, then the worst-case approximation factor is $1/(\Delta+1)$.*

The proof relies on phrasing $\mathcal{M}$ as an intersection of matroids.

Definition 4 (Matroid)**.** *A matroid $(\mathcal{V}, \mathcal{I}_k)$ consists of a ground set $\mathcal{V}$ and a family $\mathcal{I}_k \subseteq 2^{\mathcal{V}}$ of "independent sets" that satisfy three axioms: (1) $\emptyset \in \mathcal{I}_k$; (2) downward closedness: if $S \in \mathcal{I}_k$ then $T \in \mathcal{I}_k$ for all $T \subseteq S$; and (3) the exchange property: if $S, T \in \mathcal{I}_k$ and $|S| < |T|$, then there is an element $v \in T \setminus S$ such that $S \cup \{v\} \in \mathcal{I}_k$.*

Proof. (Lemma 3) We will argue that Problem 5.2 is the problem of maximizing a monotone submodular function subject to the constraint that the solution lies in the intersection of $\Delta+1$ matroids. With this insight, the approximation factor of the greedy algorithm for submodular F follows from [36] and that for non-intersecting $\Gamma(b)$ from [47], since in the latter case the problem is that of finding a maximum weight vector in the intersection of $\Delta+1$ matroids.

It remains to argue that $\mathcal{M}$ is an intersection of matroids. Our matroids will be partition matroids (over the ground set $\mathcal{V}$) whose independent sets are of the form $\mathcal{M}_k = \{S \mid |S \cap e| \leq 1, \text{ for all } e \in E_k\}$. To define those, we partition the edges in $\mathcal{G}_C$ into disjoint sets E_k, i.e., no two edges in E_k share a common node. The E_k can be found by an edge coloring – one E_k and $\mathcal{M}_k$ for each color k. By Vizing's theorem [95], we need at most $\Delta+1$ colors. The matroid $\mathcal{M}_k$ demands that for each edge $e \in E_k$, we may only select one of its adjacent nodes. All matroids together say that for any edge $e \in \mathcal{E}$, we may only select one of the adjacent nodes, and that is the constraint in Equation 5.2, i.e. $\mathcal{M} = \bigcap_{k=1}^{\Delta+1} \mathcal{M}_k$. We do not ever need to explicitly compute E_k and $\mathcal{M}_k$; all we need to do is check membership in the intersection, and this is equivalent to checking whether a set S is an independent set in $\mathcal{G}_C$, which is done by the deletions in the algorithm. □

From the constrained greedy algorithm, we obtain a set $S \subset \mathcal{V}$ of discriminative patches. Together with its neighborhood $\Gamma(b)$, each patch $b \in \mathcal{V}$ forms a representative cluster. Figure 5.2 shows some example patches derived from the labels "aeroplane" and "motorbike". The discovered patches intuitively look like "parts" of the objects, and are frequent but sufficiently different.

Finding frequent configurations. The next step is to find frequent configurations of co-occurring clusters, e.g., the head patch of a person on top of the torso patch, or a bicycle with visible wheels. A "configuration" consists of patches from two clusters C_i, C_j, their relative location, and their viewpoint and scale. In practice, we give preference to pairs that by themselves are very relevant and maximize a weighted combination of co-occurrence count and coverage $\max\{\Gamma(C_i), \Gamma(C_j)\}$.

All possible configurations of all pairs of patches amount to too many to explicitly write down and count. Instead, we follow an efficient procedure for finding frequent configurations. Our approach is inspired by [60], but does not require any supervision. We first

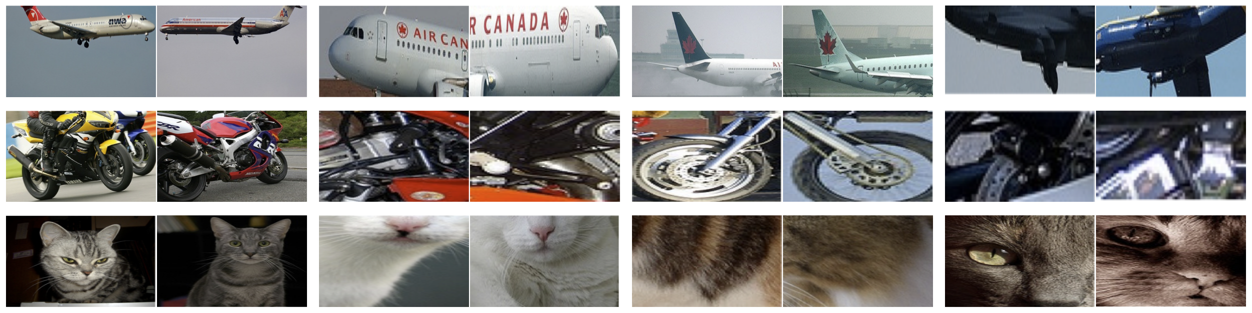
Figure 5.2: Examples of discovered patch "clusters" for aeroplane, motorbike, and cat. The discovered patches intuitively look like object parts, and are frequent but sufficiently different.

find configurations that occur in at least two images. To do so, we consider each pair of images I_1, I_2 that have at least two co-occurring clusters. For each correspondence of cluster patches across the images, we find a corresponding transform operation (translation, rescale, viewpoint change). This results in a point in a 4D transform space, for each cluster correspondence. We quantize this space into B bins. Our candidate configurations will be pairs of cluster correspondences $((b_{I_1,1}, b_{I_2,1}), (b_{I_1,2}, b_{I_2,2})) \in (C_i \times C_i), (C_j \times C_j)$ that fall in the same bin, i.e., share the same transform, and have the same relative location. Between a given pair of images, there can be multiple such pairs of correspondences. We keep track of those via a multi-graph $\mathcal{G}_P = (\mathcal{P}, \mathcal{E}_P)$ that has a node for each image $I \in \mathcal{P}$. For each correspondence $((b_{I_1,1}, b_{I_2,1}), (b_{I_1,2}, b_{I_2,2})) \in (C_i \times C_i), (C_j \times C_j)$, we draw an edge (I_1, I_2) and label it by the clusters C_i, C_j and the common relative position. As a result, there can be multiple edges (I_1, I_j) in $\mathcal{G}_P$ with different edge labels.

The most frequently occurring configuration can now be read out by finding the largest connected component in $\mathcal{G}_P$ induced by retaining only edges with the same label. We use the largest component(s) as the characteristic configurations for a given image label (object class). If the component is very small, then there is not enough information to determine co-occurrences, and we simply use the most frequent single cluster. (This may also be determined by a statistical test.) The final single "correct" localization will be the smallest bounding box that contains the full configuration.

Discovering mislocalized hard negatives. Discovering frequent configurations can not only lead to better localization estimates of the full object, but they can also be used to generate mislocalized estimates as "hard negatives" when training the object detector. We exploit this idea as follows. Let b_1, b_2 be a discovered configuration within a given image. These patches typically constitute parts or, in some cases, a part and the full object. Our foreground estimate is the smallest box that includes both b_1 and b_2. Hence, any region within the foreground estimate that does not overlap simultaneously with both b_1 and b_2 will capture only a *fragment* of the foreground object. We extract the four largest such rectangular regions (see white boxes in Figure 5.3) as "hard" negative examples.

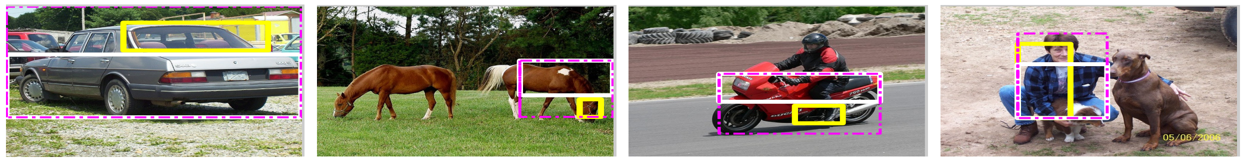

Figure 5.3: Automatically discovered foreground estimation box (magenta), hard negative (white), and the patch (yellow) that induced the hard negative. Note that we are only showing the largest one out of (up to) four hard negatives per image.

Specifically, we parameterize any rectangular region with $[x^l, x^r, y^t, y^b]$, i.e., its x-left, x-right, y-top, and y-bottom coordinate values. Let the bounding box of b_i be $[x_i^l, x_i^r, y_i^t, y_i^b]$, and foreground estimate $[x_f^l, x_f^r, y_f^t, y_f^b]$, and let $x^l = \max(x_1^l, x_2^l)$, $x^r = \min(x_1^r, x_2^r)$, $y^t = \max(y_1^t, y_2^t)$, $y^b = \min(y_1^b, y_2^b)$. We generate four hard negatives: $[x_f^l, x^l, y_f^b, y_f^t]$, $[x^r, x_f^r, y_f^b, y_f^t]$, $[x_f^l, x_f^r, y_f^t, y^t]$, $[x_f^l, x_f^r, y^b, y_f^b]$. If either b_1 or b_2 is very small in size relative to the foreground, the resulting hard negatives can have high overlap with the foreground, which will introduce undesirable noise (false negatives) when training the detector. Thus, we shrink any hard negative that overlaps with the foreground estimate by more than 50%, until its overlap is 50% (we adjust the boundary that does not coincide with any of the foreground estimation boundaries).

Finally, it is important to note that simply taking arbitrary rectangular regions that overlap with the foreground estimation box by some threshold will not always generate useful hard negatives. If the overlap threshold is too low, the selected regions will be uninformative, and if the overlap threshold is too high, the selected regions will cover too much of the foreground. Our approach selects informative hard negatives more robustly by ruling out the overlapping region between the configuration patches, which is very likely be part of the foreground object.

Mining positives and training the detector. While the discovered configurations typically lead to better foreground localization, their absolute count can be relatively low compared to the total number of positive images. This is due to inaccuracies in the initial patch discovery stage: for a frequent configuration to be discovered, both of its patches must be accurately found. Thus, we also mine additional positives from the set of remaining positive images $\mathcal{P}'$ that did not produce any of the discovered configurations.

To do so, we train an initial object detector, using the foreground estimates derived from our discovered configurations as positive examples, and the corresponding discovered hard negative regions as negatives. In addition, we mine negative examples as in [29]. We run the detector on all selective search regions in $\mathcal{P}$ and retain the region with the highest detector score as an additional positive training example. Our final detector is trained on this augmented training data, and iteratively improved by latent SVM (LSVM) updates (see [29, 87] for details).

5.4 Experiments

In this section, we analyze (1) detection performance of the models trained with the discovered configurations, and (2) impact of the discovered hard negatives on detection performance.

Implementation details. We employ a recent region based detection framework [42, 87] and use the same fc7 features from the CNN model [20] on region proposals [92] throughout the experiment. For discriminative patch discovery, we use $K = |\mathcal{P}|/2, \theta = K/20$. For correspondence detection, we discretize the 4D transform space of $\{x$: relative horizontal shift, y: relative vertical shift, s: relative scale, p: relative aspect ratio$\}$ with $\Delta x = 30\ px, \Delta y = 30\ px, \Delta s = 1\ px/px, \Delta p = 1\ px/px$. We choose this binning scheme by visually examining few qualitative examples so that scale, aspect ratio agreement between the two paired instances are more strict, while their translation agreement is more loose in order to handle deformable objects. More details regarding be transform space binning can be found in [73].

Discovered configurations. Figure 5.5 qualitatively illustrates discovered configurations (green and yellow boxes) and foreground estimates (magenta boxes) that have high degree in graph $\mathcal{G}_P$ for all classes in the PASCAL dataset. Our method consistently finds meaningful combinations such as a wheel and body of bicycles, face and torso of people, locomotive basement and upper body parts of trains/buses, window and body frame of cars, etc. Some failures include cases where the algorithm latches onto different objects co-occurring in consistent configurations such as the lamp and sofa combination (right column, second row from the bottom in Figure 5.5).

Weakly-supervised object detection. Following the evaluation protocol of the PASCAL VOC dataset, we report detection results on the PASCAL *test* set using detection average precision. For a direct comparison with the state-of-the-art weakly-supervised object detection method the previous chapter, we do not use the extra instance level annotations such as *pose, difficult, truncated* and restrict the supervision to the image level object presence annotations. Table 5.1 compares our detection results against two baseline methods [83, 87] which report the result on the full dataset. As shown in Table 5.1, our method improves detection performance on the majority of the classes (consistent improvement on rigid man-made object classes). It is worth noting that our method shows significant improvement on the person class (arguably most important category in the PASCAL dataset). Figure 5.4 shows some example high scoring detection results on the test set.

Impact of discovered hard negatives. To analyze the effect of our discovered hard negatives, we compare to two baseline cases: (1) not adding any negative examples from positives images (2) adding image regions around the foreground estimate as conventionally implemented in fully supervised object detection algorithms [28, 42]. We use the criterion from [42], where all image regions in positive images with overlap score (intersection area over union area with respect to foreground regions) less than 0.3 are used as "neighboring" negative image regions on positive images. Table 5.2 shows the effect of our hard negative examples in terms of detection average precision, for all classes (mAP). The experiment

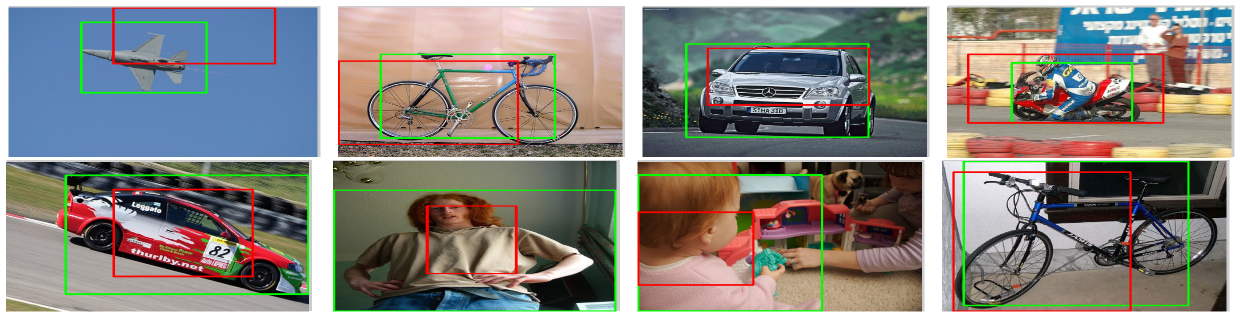
Figure 5.4: Example detections on test set. Green: our method, Red: the previous chapter

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	pson	plant	sheep	sofa	train	tv	mAP
[83]	13.4	44.0	3.1	3.1	0.0	31.2	43.9	7.1	0.1	9.3	9.9	1.5	**29.4**	38.3	4.6	0.1	0.4	3.8	34.2	0.0	13.9
the previous chapter	27.6	41.9	19.7	9.1	10.4	35.8	39.1	**33.6**	0.6	20.9	10.0	**27.7**	**29.4**	39.2	9.1	**19.3**	20.5	**17.1**	35.6	7.1	22.7
ours + SVM	31.9	47.0	21.9	8.7	4.9	34.4	41.8	25.6	0.3	19.5	**14.2**	23.0	27.8	38.7	**21.2**	17.6	**26.9**	12.8	**40.1**	9.2	23.4
ours + LSVM	**36.3**	**47.6**	**23.3**	**12.3**	**11.1**	**36.0**	**46.6**	25.4	**0.7**	**23.5**	12.5	23.5	27.9	**40.9**	14.8	19.2	24.2	**17.1**	37.7	**11.6**	**24.6**

Table 5.1: Detection average precision (%) on full PASCAL VOC 2007 test set.

	w/o hard negatives	neighboring hard negatives	discovered hard negatives
ours + SVM	22.5	22.2	**23.4**
ours + LSVM	23.7	23.9	**24.6**

Table 5.2: Effect of our hard negative examples on full PASCAL VOC 2007 test set.

shows that adding “neighboring negative regions” does not lead to noticeable improvement over not adding any negative regions from positive images, while adding our automatically discovered hard negative regions improves the detection performance more substantially.

5.5 Conclusion

We presented a novel weakly-supervised object detection method that discovers frequent configurations of discriminative visual patterns. We showed that the discovered configurations provide more accurate spatial coverage of the full object and provide a way to generate useful hard negatives. Together, these lead to state-of-the-art weakly-supervised detection results on the challenging PASCAL VOC dataset.

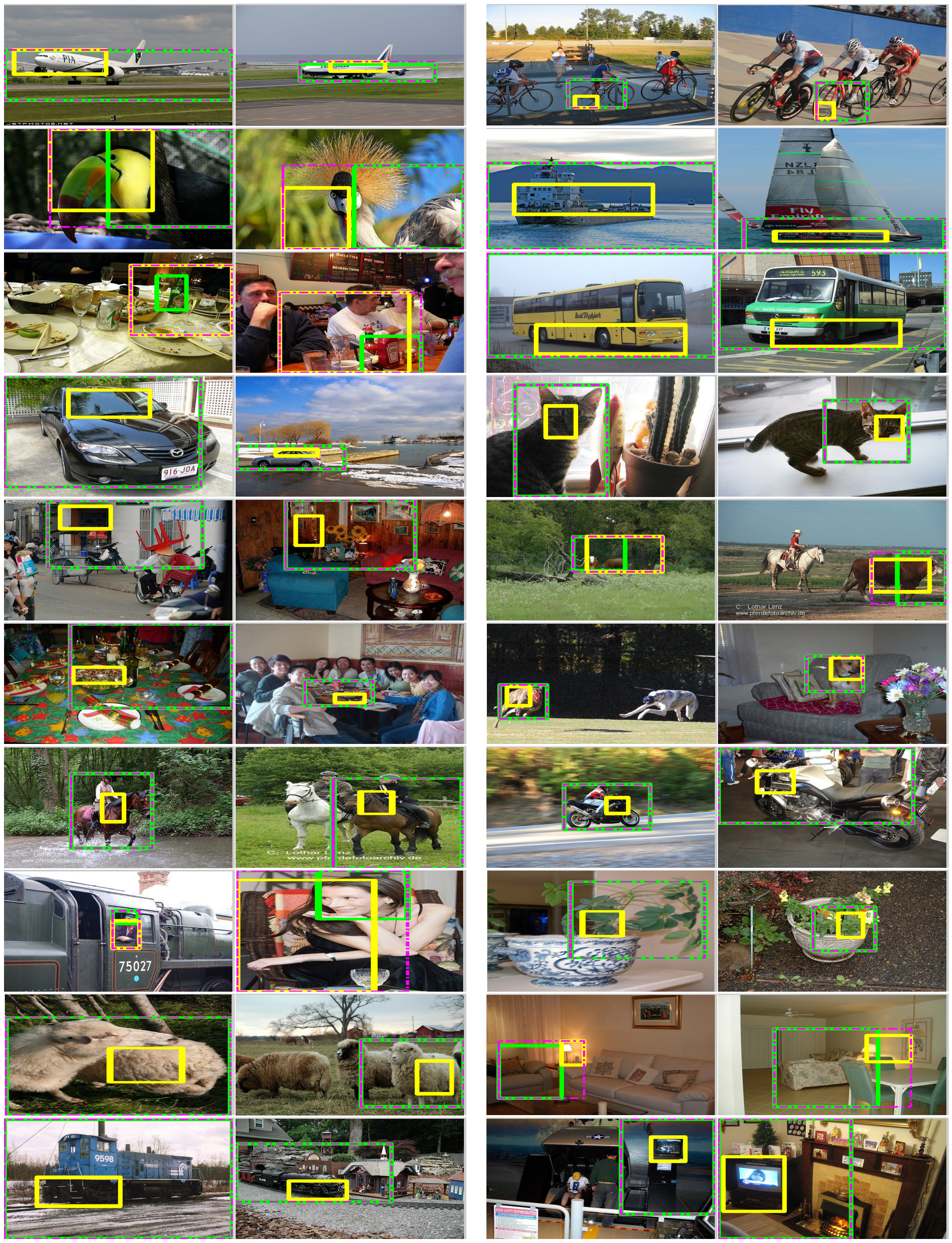

Figure 5.5: Example configurations that have high degree in graph $\mathcal{G}_P$. The green and yellow boxes show the discovered discriminative visual parts, and the magenta box shows the bounding box that tightly fits their configuration.

Chapter 6

Conclusion

This thesis presented a framework for learning and making inferences with parsimony for large scale object detection and discovery. This work not only shows the theoretical analysis and guarantees about the presented models but also demonstrates the state of the art empirical results on challenging benchmark datasets. For inference, I've presented sparselet models to address the model complexity parsimony for efficient multiclass, multi-convolutional inference. For learning, I've presented detection models to address human supervision parsimony. The proposed model demonstrates state of the art detection performance with minimal supervision. I plan to maintain the code repository on Github at https://github.com/rksltnl.

In future research, I plan to expand my work on efficient convolutional classifiers to other areas of machine learning and computer vision, and also build on my experience in learning object detectors with large scale unconstrained data. First, deep learning methods with convolutional neural networks (DCNN) recently have shown promising results on benchmark object recognition datasets. However, it is often required to run days if not weeks of computation on GPU architectures to train DCNN models. The computational bottleneck in the DCNN framework is in multi-layer convolution with thousands of filters. I plan to approach this computational burden along the lines of the *sparselet* concept: learn a discriminatively trained family of matrices with special structures (i.e. low rank and sparse, Toeplitz, Hadamard, etc) which are more computationally tractable as the complexity of the training model increases. The second line of future research is about designing efficient and practical algorithms for dealing with massive amount of negative data. This is a relaxed setting from the fully unsupervised setting, where we still work with the smaller set of fully labelled data for positives but utilize the large scale unconstrained data for negatives. The classical solution on handling large scale negative data in computer vision and machine learning community comes under the guise of streaming based data mining or bootstrapping. Some of the drawbacks of these classical approaches include restrictions on the choice of loss functions, lack of analysis on practical convergence criteria. I plan to explore a theoretical connection between streaming based data mining and generalized importance sampling.

Bibliography

[1] B. Alexe, T. Deselaers, and V. Ferrari. “Classcut for Unsupervised Class Segmentation”. In: *ECCV*. 2010.

[2] S Andrews, I Tsochantaridis, and T Hofmann. “Support vector machines for multiple-instance learning”. In: *NIPS*. 2003.

[3] F. Bach et al. “Optimization with Sparsity-Inducing Penalties”. In: *Foundations and Trends in Machine Learning* 4.1 (2012), pp. 1–106.

[4] O. Barinova, V. Lempitsky, and P. Kohli. “On Detection of Multiple Object Instances using Hough Transforms”. In: *IEEE TPAMI* (2012).

[5] L. Bourdev and J. Malik. “Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations”. In: *ICCV*. IEEE. 2009, pp. 1365–1372.

[6] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] Calinescu et al. “Maximizing a submodular set function subject to a matroid constraint”. In: *SIAM Journal on Computing* (2012).

[8] X. Chen, A. Shrivastava, and A. Gupta and. “NEIL: Extracting Visual Knowledge from Web Data”. In: *ICCV*. 2013.

[9] Y. Chen et al. “Active Detection via Adaptive Submodularity”. In: *ICML*. 2014.

[10] O. Chum and A. Zisserman. “An exemplar model for learning object classes”. In: *CVPR*. 2007.

[11] D. Crandall and D. Huttenlocher. “Weakly supervised learning of part-based spatial models for visual object recognition”. In: *ECCV*. 2006.

[12] N. Dalal and B. Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *CVPR*. IEEE. 2005.

[13] T. Darrell, S. Sclaroff, and A. Pentland. “Segmentation by Minimal Description”. In: *ICCV*. 1990.

[14] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR*. IEEE. 2009.

[15] J. Deng et al. “What does classifying more than 10,000 image categories tell us?” In: *ECCV*. 2010.

[16] T. Deselaers, B. Alex, and V. Ferrari. “Localizing objects while learning their appearance”. In: *ECCV*. 2010.

[17] T. Deselaers, B. Alex, and V. Ferrari. “Weakly supervised localization and learning with generic knowledge”. In: *IJCV* (2012).

[18] C. Doersch, A. Gupta, and A. Efros. “Mid-level visual element discovery as discriminative mode seeking”. In: *NIPS*. 2013.

[19] C. Doersch et al. “What makes Paris look like Paris?” In: *SIGGRAPH*. 2012.

[20] J. Donahue et al. “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”. In: *arXiv e-prints* arXiv:1310.1531 [cs.CV] (2013).

[21] J. Donahue et al. “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”. In: *ICML*. 2014.

[22] I. Endres, K. Shih, and D. Hoeim. “Learning collections of part models for object recognition”. In: *CVPR*. 2013.

[23] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/ index.html.

[24] M. Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[25] A. Faktor and M. Irani. “Clustering by Composition Unsupervised Discovery of Image Categories”. In: *ECCV*. 2012.

[26] A. Farhadi and A. Sadeghi. “Recognition Using Visual Phrases”. In: *CVPR*. 2011.

[27] L. Fei-Fei, R. Fergus, and P. Perona. “One-shot learning of object categories”. In: *IEEE TPAMI* 28.4 (2006), pp. 594–611.

[28] P. Felzenszwalb, D. McAllester, and D. Ramanan. “A Discriminatively Trained, Multiscale, Deformable Part Model”. In: *CVPR*. 2008.

[29] P. Felzenszwalb et al. “Object Detection with Discriminatively Trained Part Based Models”. In: *IEEE TPAMI* 32.9 (2010).

[30] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. *Discriminatively Trained Deformable Part Models, Release 4*. http://people.cs.uchicago.edu/~pff/latent-release4/.

[31] P. F. Felzenszwalb et al. “Object Detection with Discriminatively Trained Part Based Models”. In: *IEEE TPAMI* 32.9 (2010), pp. 1627–1645.

[32] P.F. Felzenszwalb, R.B. Girshick, and D. McAllester. “Cascade object detection with deformable part models”. In: *CVPR*. 2010.

[33] R. Fergus, P. Perona, and A. Zisserman. “Object class recognition by unsupervised scale-invariant learning”. In: *CVPR*. 2003.

[34] R. Fergus, P. Perona, and A. Zisserman. "Weakly supervised scale-invariant learning of models for visual recognition". In: *IJCV* (2007).

[35] S. Fidler, M. Boben, and A. Leonardis. "Learning Hierarchical Compositional Representations of Object Structure". In: *Object Categorization: Computer and Human Vision Perspectives*. Ed. by Sven Dickinson et al. Cambridge University Press, 2009.

[36] M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. "An analysis of approximations for maximizing submodular set functions - II". In: *Math. Prog. Study* 8 (1978), pp. 73–87.

[37] W.T. Freeman and E.H. Adelson. "The design and use of steerable filters". In: *IEEE TPAMI* 13.9 (1991), pp. 891–906.

[38] K. Fukunaga and L. Hostetler. "The estimation of the gradient of a density function, with applications in pattern recognition". In: *Information Theory* (1975).

[39] C. Galleguillos et al. "Weakly supervised object localization with stable segmentations". In: *ECCV*. 2008.

[40] R. Girshick. "From Rigid Templates to Grammars: Object Detection with Structured Models". PhD thesis. University of Chicago, 2012.

[41] R. Girshick, H. Song, and T. Darrell. "Discriminatively Activated Sparselets". In: *ICML*. 2013.

[42] R. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *arXiv e-prints* (2013).

[43] R. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CVPR*. 2014.

[44] R.B Girshick, P.F. Felzenszwalb, and D. McAllester. "Object detection with grammar models". In: *NIPS*. 2011.

[45] K. Grauman and T. Darrell. "Unsupervised learning of categories from sets of partially matching image features". In: *CVPR*. 2006.

[46] G. Griffin, A. Holub, and P. Perona. *Caltech-256 Object Category Dataset*. Tech. rep. 7694. California Institute of Technology, 2007. URL: `http://authors.library.caltech.edu/7694`.

[47] T.A. Jenkyns. "The efficacy of the "greedy" algorithm". In: *Proc. of 7th South Eastern Conference on Combinatorics, Graph Theory and Computing*. 1976, pp. 341–350.

[48] A. Joulin and F. Bach. "A convex relaxation for weakly supervised classifiers". In: *ICML*. 2012.

[49] A. Joulin, F. Bach, and J. Ponce. "Discriminative clustering for image co-segmentation". In: *CVPR*. 2010.

[50] M. Juneja et al. "Blocks that shout: Distinctive parts for scene classification". In: *CVPR*. 2013.

[51] M. Juneja et al. "Blocks that Shout: Distinctive Parts for Scene Classification". In: *CVPR*. 2013.

[52] G. Kim et al. "Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion". In: *ICCV*. 2011.

[53] A. Krause and D. Golovin. "Submodular Function Maximization". In: *Chapter in Tractability: Practical Approaches to Hard Problems (to appear)* (2014).

[54] K. Kreutz-Delgado et al. "Dictionary learning algorithms for sparse representation". In: *Neural computation* 15.2 (2003), pp. 349–396.

[55] A. Krizhevsky and I. Sutskever G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.

[56] P Kumar, B Packer, and D Koller. "Modeling Latent Variable Uncertainty for Loss-based Learning". In: *ICML*. 2012.

[57] J. Langford, L. Li, and T. Zhang. "Sparse online learning via truncated gradient". In: *JMRL* 10 (2009), pp. 777–801.

[58] Y. J. Lee and K. Grauman. "Foreground Focus: Unsupervised Learning From Partially Matching Images". In: *IJCV* 85 (2009).

[59] B. Leibe, A. Leonardis, and B. Schiele. "Combined Object Categorization and Segmentation with an Implicit Shape Model". In: *Wkshp on Statistical Learning in Computer Vision*. 2004.

[60] C. Li, D. Parikh, and T. Chen. "Automatic Discovery of Groups of Objects for Scene Understanding". In: *CVPR*. 2012.

[61] Y. Li et al. "Convex and Scalable Weakly Labeled SVMs". In: *ICML*. 2013.

[62] P.M. Long and L. Tan. "PAC learning axis aligned rectangles with respect to product distributions from multiple-instance examples". In: *Proc. Comp. Learning Theory*. 1996.

[63] J. Mairal, F. Bach, and J. Ponce. "Task-Driven Dictionary Learning". In: *IEEE TPAMI* 32.4 (2012).

[64] J. Mairal et al. "Online Dictionary Learning for Sparse Coding". In: *ICML*. ACM. 2009.

[65] Stphane Mallat and Zhifeng Zhang. "Matching Pursuit With Time-Frequency Dictionaries". In: *IEEE Trans. on Signal Processing* 41 (1993), pp. 3397–3415.

[66] R. Manduchi, P. Perona, and D. Shy. "Efficient deformable filter banks". In: *IEEE Trans. on Signal Processing* 46.4 (1998), pp. 1168–1173.

[67] K. Micolajczyk, G. Leibe, and B. Schiele. "Multiple Object Class Detection with a Generative Model". In: *CVPR*. 2006.

[68] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical Programming* 14.1 (1978), pp. 265–294.

[69] Y Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical Programming* 103.1 (2005).

[70] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, 1999.

[71] P. Ott and M. Everingham. "Shared parts for deformable part-based models". In: *CVPR.* IEEE. 2011, pp. 1513–1520.

[72] M. Pandey and S. Lazebnik. "Scene recognition and weakly supervised object localization with deformable part-based models". In: *ICCV.* 2011.

[73] D. Parikh, C. L. Zitnick, and T. Chen. "From Appearance to Context-Based Recognition: Dense Labeling in Small Images". In: *CVPR.* 2008.

[74] H. Pirsiavash and D. Ramanan. "Steerable Part Models". In: *CVPR.* IEEE. 2012.

[75] T. Quack et al. "Efficient Mining of Frequent and Distinctive Feature Configurations". In: *ICCV.* 2007.

[76] M. Raptis, I. Kokkinos, and S. Soatto. "Discovering discriminative action parts from mid-level video representations". In: *CVPR.* 2012.

[77] C. Rother et al. "Cosegmentation of Image Pairs by Histogram Matching Incorporating a Global Constraint into MRFs". In: *CVPR.* 2006.

[78] O. Russakovsky et al. "Object-centric spatial pooling for image classification". In: *ECCV.* 2012.

[79] B. Sarwar et al. "Application of Dimensionality Reduction in Recommender SystemsA Case Study". In: *Proc. ACM WebKDD Workshop.* 2000.

[80] S. Singh, A. Gupta, and A. Efros. "Unsupervised discovery of mid-level discriminative patches". In: *ECCV.* 2012.

[81] S. Singh, A. Gupta, and A. A. Efros. "Unsupervised Discovery of Mid-Level Discriminative Patches". In: *ECCV.* 2012.

[82] P. Siva, C. Russell, and T. Xiang. "In defence of negative mining for annotating weakly labelled data". In: *ECCV.* 2012.

[83] P. Siva and T. Xiang. "Weakly supervised object detector learning with model drift detection". In: *ICCV.* 2011.

[84] J. Sivic and A. Zisserman. "Video Data Mining Using Configurations of Viewpoint Invariant Regions". In: *CVPR.* 2004.

[85] J. Sivic et al. "Discovering Object Categories in Image Collections". In: *ICCV.* 2005.

[86] H. Song et al. "Sparselet Models for Efficient Multiclass Object Detection". In: *ECCV.* Springer-Verlag. 2012.

[87] H. O. Song et al. “On learning to localize objects with minimal supervision”. In: *ICML*. 2014.

[88] B. Taskar, C. Guestrin, and D. Koller. “Max-Margin Markov Networks”. In: *NIPS*. 2003.

[89] R. Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society Series B* (1996), pp. 267–288.

[90] A. Torralba, K.P. Murphy, and W.T. Freeman. “Sharing visual features for multiclass and multiview object detection”. In: *IEEE TPAMI* 29.5 (2007), pp. 854–869.

[91] I. Tsochantaridis et al. “Large margin methods for structured and interdependent output variables”. In: *JMRL* 6.2 (2006), pp. 1453–1484.

[92] J. Uijlings et al. “Selective search for object recognition”. In: *IJCV*. 2013.

[93] A. Vedaldi and B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. `http://www.vlfeat.org/`. 2008.

[94] A. Vedaldi and A. Zisserman. “Efficient Additive Kernels via Explicit Feature Maps”. In: *IEEE TPAMI* 34.3 (2011).

[95] V.G. Vizing. “On an estimate of the chromatic class of a p-graph”. In: *Diskret. Analiz.* 3 (1964), pp. 25–30.

[96] M. Weber, M. Welling, and P. Perona. “Towards automatic discovery of object categories”. In: *CVPR*. 2000.

[97] M. Weber, M. Welling, and P. Perona. “Unsupervised Learning of Models for Recognition”. In: *ECCV*. 2000.

[98] M. Weber, M. Welling, and P. Perona. “Unsupervised learning of models for recognition”. In: *ECCV*. 2000.

[99] Wikipedia. *AI winter — Wikipedia,* [Online; accessed 22-July-2014]. 2014. URL: `http://en.wikipedia.org/wiki/AI_winter`.

[100] Lior Wolf, Hueihan Jhuang, and Tamir Hazan. “Modeling Appearances with Low-Rank SVM”. In: *CVPR*. 2007.

[101] L. Wolsey. “An analysis of the greedy algorithm for the submodular set covering problem”. In: *Combinatorica* 2 (1982), pp. 385–393.

[102] C.N. Yu and T Joachims. “Learning Structural SVMs with Latent Variables”. In: *ICML*. 2009.

[103] A.L. Yuille and A. Rangarajan. “The Concave-Convex Procedure”. In: *Neural Computation* 15.4 (2003), pp. 915–936.

[104] Y. Zhang and T. Chen. “Efficient Kernels for Identifying Unbounded-order Spatial Features”. In: *CVPR*. 2009.

[105] L.L. Zhu et al. “Part and Appearance Sharing: Recursive Compositional Models for Multi-View Multi-Object Detection”. In: *CVPR*. IEEE. 2010, pp. 1919–1926.

[106] H. Zou and T. Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society Series B* (2005), pp. 301–320.

[107] James Zou et al. “Contrastive Learning Using Spectral Methods”. In: *NIPS*. 2013.