Discriminatively Activated Sparselets

Ross Girshick* RBG@EECS.BERKELEY.EDU
Hyun Oh Song* SONG@EECS.BERKELEY.EDU
Trevor Darrell TREVOR@EECS.BERKELEY.EDU
University of California, Berkeley, Berkeley, CA 94720.

Abstract

Shared representations are highly appealing due to their potential for gains in computational and statistical efficiency. Compressing a shared representation leads to greater computational savings, but can also severely decrease performance on a target task. Recently, *sparselets* (Song et al., 2012) were introduced as a new shared intermediate representation for multiclass object detection with deformable part models (Felzenszwalb et al., 2010a), showing significant speedup factors, but with a large decrease in task performance. In this paper we describe a new training framework that learns which sparselets to activate in order to optimize a discriminative objective, leading to larger speedup factors with no decrease in task performance. We first reformulate sparselets in a general structured output prediction framework, then analyze when sparselets lead to computational efficiency gains, and lastly show experimental results on object detection and image classification tasks. Our experimental results demonstrate that discriminative activation substantially outperforms the previous reconstructive approach which, together with our structured output prediction formulation, make sparselets broadly applicable and significantly more effective.

1. Introduction

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 (Freeman & Adelson, 1991), low-rank approximations for collaborative filtering (Sarwar et al., 2000), and shared part models for object detection (Zhu et al., 2010; Ott & Everingham, 2011; Girshick et al., 2011).

Recently, sparselets (Song et al., 2012) were introduced as a new shared intermediate representation for multiclass object detection with deformable part models (DPMs) (Felzenszwalb et al., 2010a). 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 miniparts. 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, and outperformed the CPU version of the cascade algorithm in (Felzenszwalb et al., 2010b) 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 (Song et al., 2012) is brittle, and pushing slightly beyond these speedup factors leads to a substantial loss in detection accuracy.

This paper makes two contributions, the first of which remedies the brittleness of the approach in (Song et al., 2012). We formulate a new, discriminative framework for sparselet activation training that leads to greater speedup factors while maintaining high task performance. Second, (Song et al., 2012) introduced sparselets only in the context of DPMs, making their applicability limited. In this paper we show how to formulate sparselets in a generic structured output prediction (Taskar et al., 2003; Tsochantaridis et al., 2006) setting and then analyze when the structure of the problem leads to gains in computational efficiency. This new insight opens the path for the application of sparselets to a wide variety of problems. As an example, our

*These authors contributed equally.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

experiments demonstrate that sparselets are highly effective when applied to off-the-shelf image classifiers.

Our work is related to three strands of active research: (1) part sharing with compositional models (Torralba et al., 2007; Fidler et al., 2009; Zhu et al., 2010; Ott & Everingham, 2011; Girshick et al., 2011), (2) sparse coding and dictionary learning (Kreutz-Delgado et al., 2003; Mairal et al., 2009; 2012), and (3) modeling and learning with low-rank approximations (Freeman & Adelson, 1991; Manduchi et al., 1998; Wolf et al., 2007). None of these methods, however, simultaneously exploit shared interclass information *and* discriminative sparsity learning to speed up inference while maintaining task performance.

This paper also helps unify sparselets with the steerable part models of (Pirsiavash & Ramanan, 2012). These closely related approaches were both applied to DPMs and resulted in the same 3x speedup factor. 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 paper unifies the two approaches by showing how to train sparselet activations discriminatively, or alternately, how to train steering coefficients sparsely.

The paper is structured as follows. In Sec. 2, we start with a brief overview of sparselets (Song et al., 2012) and formulate structured output prediction with generalized sparselets. In Sec. 3, we describe how discriminative sparselet activation training fits into the framework and discuss several regularization methods for sparse activation learning. In Sec. 4, we discuss important applications of the proposed approach to multiclass object detection with mixtures of deformable part models (Felzenszwalb et al., 2010a) and to multiclass image classification. Before we conclude in Sec. 6, we provide experimental results on multiclass object detection and multiclass image classification problems in Sec. 5.

2. Generalized sparselets

In this section we introduce *generalized sparselets* — a general approach for speeding up inference in any linear structured output prediction model.

2.1. Sparselets reviewed

Sparselets were introduced in (Song et al., 2012) for the purpose of accelerating object detection with deformable part models (DPMs) (Felzenszwalb et al., 2010a). In brief, a *sparselet model* is completely specified by a dictionary $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_d]$ in $\mathbb{R}^{m \times d}$, where each column $\mathbf{s}_j$ in $\mathbb{R}^m$ is called a *sparselet*. Noting that the computational bottleneck of detection is convolution of a feature pyramid $\mathbf{\Psi}$ with a set of DPM part filters, $\{\mathbf{f}_i\}$, (Song et al., 2012) proposed to approximate each filter $\mathbf{f}_i$ as a sparse linear combination of sparselets, yielding: $\mathbf{\Psi} * \mathbf{f}_i \approx \mathbf{\Psi} * \sum_j \alpha_{ij} \mathbf{s}_j = \sum_j \alpha_{ij} (\mathbf{\Psi} * \mathbf{s}_j)$. The sparselet responses $\mathbf{\Psi} * \mathbf{s}_j$ are independent of any filter, and thus their cost can be amortized over all filters from all object models. 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.

2.2. Multiclass classification with generalized sparselets

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

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

Our objective is to reduce the computational cost of computing Eq. 1.

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 from all $\mathbf{w}_k$ 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. 1), 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 p 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 $\mathbf{S}$

$$\mathbf{b} \approx \mathbf{S}\boldsymbol{\alpha} = \sum_{\substack{i=1 \\ \alpha_i \neq 0}}^{d} \alpha_i \mathbf{s}_i, \tag{2}$$

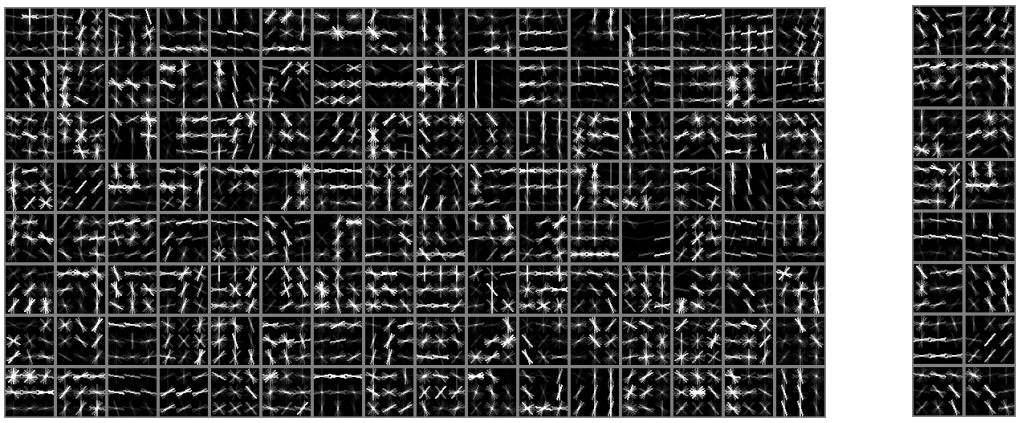

Figure 1. (Left) 128 of the 256 sparselets learned from 20 DPMs trained on the PASCAL VOC 2007 dataset. (Right) The top 16 sparselets activated for the motorbike category.

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. 1 can be approximated as

$$\begin{aligned}\mathbf{w}_k^\intercal \mathbf{x} &= (\mathbf{b}_{k1}^\intercal, \ldots, \mathbf{b}_{kp}^\intercal)(\mathbf{c}_1^\intercal, \ldots, \mathbf{c}_p^\intercal)^\intercal \\ &= \sum_{i=1}^{p} \mathbf{b}_{ki}^\intercal \mathbf{c}_i \approx \sum_{i=1}^{p} (\mathbf{S}\boldsymbol{\alpha}_{ki})^\intercal \mathbf{c}_i = \sum_{i=1}^{p} \boldsymbol{\alpha}_{ki}^\intercal (\mathbf{S}^\intercal \mathbf{c}_i). \quad (3)\end{aligned}$$

We note two important properties of Eq. 3: (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 $\boldsymbol{\alpha}_{ki}$.

Computational costs. 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 (Song et al., 2012). This emphasizes the importance of a sparse representation, in contrast, for example, to the dense steering coefficients in (Pirsiavash & Ramanan, 2012). 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 datasets such as ImageNet (Deng et al., 2009), which includes more than 10,000 categories (Deng et al., 2010). In Sec. 5.3 we show results on the Caltech-{101,256} (Fei-Fei et al., 2006; Griffin et al., 2007) datasets demonstrating that even with only one or two hundred classes generalized sparselets can accelerate simple linear classifiers.

2.3. Structured output prediction with generalized sparselets

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

$$f_{\mathbf{w}}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w}^\intercal \boldsymbol{\Phi}(x, y) \quad (4)$$

where the input x comes from an arbitrary input space $\mathcal{X}$, and $f_{\mathbf{w}}$ outputs an element from the label space $\mathcal{Y}$. As in the previous discussion, $\mathbf{w}$ is partitioned into blocks in a problem specific manner. The partition used in the multiclass setup is one concrete example. Given the partition, sparselets can be applied to each block in a straightforward extension of the multiclass case.

Computational costs. 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 unique parameter blocks that are multiplied with a distinct subvector of feature values.* Intuitively, Q counts the number of times the intermediate

sparselet response, for a distinct feature vector block, is reused while solving the argmax in Eq. 4. The value of Q depends on the specific feature map $\mathbf{\Phi}(x, y)$ and inference algorithm, and in general may vary across feature blocks. We point interested readers to a more detailed analysis in the supplementary material. 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 (Dalal & Triggs, 2005). 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 (Felzenszwalb et al., 2010a). 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 (Dalal & Triggs, 2005; Felzenszwalb et al., 2010a). 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. Discriminative activation of generalized sparselets

Throughout the rest of this paper 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 (Song et al., 2012), 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 (Kreutz-Delgado et al., 2003; Mairal et al., 2009). 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 (Song et al., 2012) 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 learning paradigm used in (Song et al., 2012): when reconstruction error is low (usually requiring 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 aug-

ment this new objective function with a sparsity inducing regularizer. As we show below, the obvious choice of ℓ_1 regularization leads to unsatisfactory results, motivating the development of an alternative approach.

3.1. Learning discriminative activation vectors

Here we consider learning the activation vectors for a predictor $\mathbf{w}$ in the structural SVM (SSVM) framework (Taskar et al., 2003; Tsochantaridis et al., 2006). The SSVM training equation is

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{M}\sum_{i=1}^{M} \max_{\hat{y}\in\mathcal{Y}} (\mathbf{w}^\intercal \mathbf{\Phi}(x_i, \hat{y}) + \Delta(y_i, \hat{y})) - \mathbf{w}^\intercal \mathbf{\Phi}(x_i, y_i), \tag{5}$$

where $\Delta(y, y')$ is a loss function. Given a fixed sparselet model $\mathbf{S}$, we can rewrite Eq. 5 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 $\mathbf{\Phi}_k(x, y) = (\mathbf{c}_{k1}^\intercal, \ldots, \mathbf{c}_{kp_k}^\intercal)^\intercal$ that will be represented by sparselets, we transform the features into sparselet responses: $\tilde{\mathbf{\Phi}}_k(x, y) = (\mathbf{c}_{k1}^\intercal \mathbf{S}, \ldots, \mathbf{c}_{kp_k}^\intercal \mathbf{S})^\intercal \in \mathbb{R}^{p_k d}$. The fully transformed feature vector is $\tilde{\mathbf{\Phi}}(x, y) = (\tilde{\mathbf{\Phi}}_1^\intercal(x, y), \ldots, \tilde{\mathbf{\Phi}}_s^\intercal(x, y), \mathbf{\Phi}_{s+1}^\intercal(x, y), \ldots, \mathbf{\Phi}_K^\intercal(x, y))^\intercal$. The resulting objective is

$$\boldsymbol{\beta}^* = \operatorname*{argmin}_{\boldsymbol{\beta}} R(\boldsymbol{\alpha}) + \frac{\lambda}{2}\|\tilde{\mathbf{w}}\|_2^2 + \frac{1}{M}\sum_{i=1}^{M} \max_{\hat{y}\in\mathcal{Y}} \left(\boldsymbol{\beta}^\intercal \tilde{\mathbf{\Phi}}(x_i, \hat{y}) + \Delta(y_i, \hat{y})\right) - \boldsymbol{\beta}^\intercal \tilde{\mathbf{\Phi}}(x_i, y_i), \tag{6}$$

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

3.2. Inducing sparsity

We consider three sparsity inducing regularizers R.

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

I. **Lasso penalty** (Tibshirani, 1996)
$R_{\text{Lasso}}(\boldsymbol{\alpha}) = \lambda_1\|\boldsymbol{\alpha}\|_1$

II. **Elastic net penalty** (Zou & Hastie, 2005)
$R_{\text{EN}}(\boldsymbol{\alpha}) = \lambda_1\|\boldsymbol{\alpha}\|_1 + \lambda_2\|\boldsymbol{\alpha}\|_2^2$

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

The first two regularizers lead to convex optimization problems, however the third does not. We consider two alternative methods for approximately minimizing Eq. 6 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 $\|\boldsymbol{\alpha}\|_0 \leq \lambda_0$. In the second step, the selection of nonzero variables is fixed, thus satisfying the sparsity constraint and resulting in convex optimization problem to solve. We consider the following variable selection strategies.

III-A. **Overshoot, 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. **Orthogonal 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 (Mallat & Zhang, 1993) as implemented in the SPAMS software package (Mairal et al., 2009). This produces the same initial set of activation vectors as the baseline method (Song et al., 2012). However, we then learn the selected variables discriminatively according to Eq. 6.

4. Implementation

We first focus on the application of our novel sparselet activation vector learning approach to object detection with mixtures of deformable part models (Felzenszwalb et al., 2010a) in order to facilitate direct comparison with the results in (Song et al., 2012). In brief, the deformable part model (DPM) from (Felzenszwalb et al., 2010a) 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

$$\begin{aligned}\text{score}(x, z) = w_c + \sum_{i=0}^{N} \mathbf{w}_{ci}^{\intercal} \boldsymbol{\psi}_{ci}(x, \rho_i) \\ + \sum_{i=1}^{N} \mathbf{d}_{ci}^{\intercal} \boldsymbol{\delta}_{ci}(\rho_0, \rho_i) = \mathbf{w}^{\intercal} \boldsymbol{\Phi}(x, z),\end{aligned} \tag{7}$$

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

$$\begin{aligned}\mathbf{w} = (w_1, \ldots, w_C, \mathbf{w}_{10}^{\intercal}, \ldots, \mathbf{w}_{1N}^{\intercal}, \ldots, \mathbf{w}_{C0}^{\intercal}, \ldots, \mathbf{w}_{CN}^{\intercal}, \\ \mathbf{d}_{11}^{\intercal}, \ldots, \mathbf{d}_{1N}^{\intercal}, \ldots, \mathbf{d}_{C1}^{\intercal}, \ldots, \mathbf{d}_{CN}^{\intercal})^{\intercal}\end{aligned} \tag{8}$$

and a sparse cumulative feature vector $\boldsymbol{\Phi}(x, z)$ 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 (Song et al., 2012) and (Pirsiavash & Ramanan, 2012) 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 (Felzenszwalb et al.) in the PASCAL VOC 2007 dataset (Everingham et al.), 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. 5.1), as well as to generalize to previously unseen classes from the ImageNet dataset (Sec. 5.2). Our DPM sparselets are visualized in Fig. 1.

4.1. Latent SVM

The DPMs in (Felzenszwalb et al., 2010a) are learned by optimizing a latent SVM (LSVM):

$$\begin{aligned}\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \\ \frac{1}{M}\sum_{i=1}^{M} \max\left(0, 1 - y_i \max_{z \in \mathcal{Z}(x_i)} \mathbf{w}^{\intercal}\boldsymbol{\Phi}(x_i, z)\right).\end{aligned} \tag{9}$$

The objective function in Eq. 9 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. 9 (see (Felzenszwalb et al., 2010a) 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. 3.1 to the LSVM's convex optimization subproblem.

Our implementation is based on the `voc-release4` source code from (Felzenszwalb et al.). To optimize the transformed objective function Eq. 6 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.* (Langford et al., 2009). This method achieves actual sparsity by shrinking parameters and then truncating small values every few SGD iterations.

4.2. Visualizing learned DPM sparselets

Each DPM sparselet can be visualized as a 3×3 filter. In Fig. 1 (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. 1 (right) shows the top 16 sparselets for the motorbike category. Some of the activated sparselets resemble circular fragments of wheels.

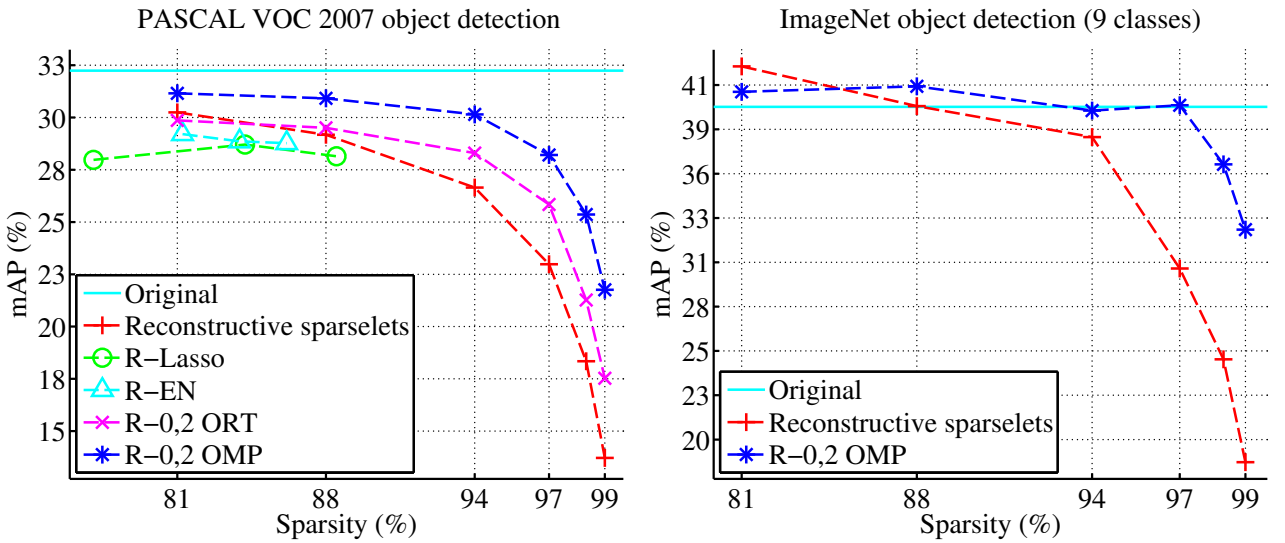

Figure 2. Mean average precision (mAP) *vs.* sparsity for object detection on the PASCAL 2007 dataset (left) and for 9 classes from ImageNet (right). Using the implementation in (Song et al., 2012), sparsity levels 81 ~ 99% correspond to speedup factors of 15 ~ 25x over the cascade (Felzenszwalb et al., 2010b) algorithm on CPU. 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 (Song et al., 2012); the remaining methods correspond to discriminative learning (our method) with each of the regularizers described in Sec. 3.2.

4.3. Image classification

To illustrate generalized sparselets applicability beyond DPMs, we evaluated our approach on the Caltech-101 (Fei-Fei et al., 2006) (102 classes, including background) and Caltech-256 (257 classes) (Griffin et al., 2007) 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 (Vedaldi & Fulkerson, 2008). 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 (Vedaldi & Zisserman, 2011)). In Sec. 5.3 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.

5. Experiments

We performed three sets of experiments, two with multiclass object detection and the third with multiclass image classification. The first experiment was designed to evaluate each of the regularization methods described in Sec. 3.2. The second 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. In the final experiment, we demonstrate generalized sparselets with their application to a standard multiclass image classification problem.

5.1. Comparison of regularization methods

We evaluated the baseline reconstructive sparselets (Song et al., 2012) and our discriminatively trained activation vectors on the PASCAL VOC 2007 dataset (Everingham et al.). Fig. 2 (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 de-

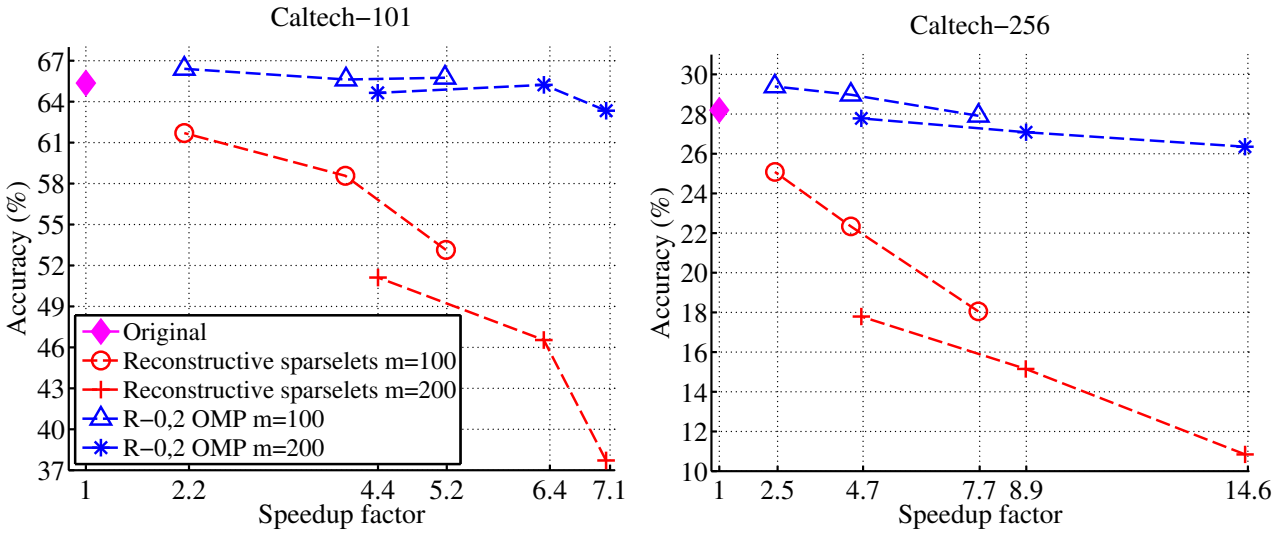

Figure 3. Average classification accuracy *vs.* speedup factor for Caltech-{101,256}.

sired 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.

5.2. 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 (Deng et al., 2009) 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. 2 (right) shows that our method generalizes well to novel classes and maintains competitive detection performance even in the high sparsity regime.

5.3. Image classification with generalized sparselets

Fig. 3 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, 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.

6. Conclusion

In this work, we generalize sparselets beyond deformable part models to any structured output prediction problem. We then show how to learn the sparselet activation vectors discriminatively using structural SVMs with sparsity constraints. Our experiments show that learning activation vectors discriminatively significantly outperforms reconstructive learning in benchmark object detection and classification tasks.

References

Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *CVPR*. IEEE, 2005.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*. IEEE, 2009.

Deng, J., Berg, A., Li, K., and Fei-Fei, L. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.

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

Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE TPAMI*, 28(4), 2006.

Felzenszwalb, P. F., Girshick, R. B., and McAllester, D. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/~pff/latent-release4/.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part based models. *IEEE TPAMI*, 32(9), 2010a.

Felzenszwalb, P.F., Girshick, R.B., and McAllester, D. Cascade object detection with deformable part models. In *CVPR*, 2010b.

Fidler, S., Boben, M., and Leonardis, A. Learning hierarchical compositional representations of object structure. In *Object Categorization: Computer and Human Vision Perspectives*. 2009.

Freeman, W.T. and Adelson, E.H. The design and use of steerable filters. *IEEE TPAMI*, 13(9), 1991.

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

Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

Kreutz-Delgado, K., Murray, J.F., Rao, B.D., Engan, K., Lee, T.W., and Sejnowski, T.J. Dictionary learning algorithms for sparse representation. *Neural computation*, 15(2), 2003.

Langford, J., Li, L., and Zhang, T. Sparse online learning via truncated gradient. *JMRL*, 10, 2009.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *ICML*. ACM, 2009.

Mairal, J., Bach, F., and Ponce, J. Task-driven dictionary learning. *IEEE TPAMI*, 32(4), 2012.

Mallat, Stphane and Zhang, Zhifeng. Matching pursuit with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, 41, 1993.

Manduchi, R., Perona, P., and Shy, D. Efficient deformable filter banks. *IEEE Trans. on Signal Processing*, 46(4), 1998.

Ott, P. and Everingham, M. Shared parts for deformable part-based models. In *CVPR*. IEEE, 2011.

Pirsiavash, H. and Ramanan, D. Steerable part models. In *CVPR*. IEEE, 2012.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Application of dimensionality reduction in recommender systemsa case study. In *Proc. ACM WebKDD Workshop*, 2000.

Song, H., Zickler, S., Althoff, T., Girshick, R., Fritz, M., Geyer, C., Felzenszwalb, F., and Darrell, T. Sparselet models for efficient multiclass object detection. In *ECCV*. Springer-Verlag, 2012.

Taskar, B., Guestrin, C., and Koller, D. Max-margin markov networks. In *NIPS*. 2003.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, 1996.

Torralba, A., Murphy, K.P., and Freeman, W.T. Sharing visual features for multiclass and multiview object detection. *IEEE TPAMI*, 29(5), 2007.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *JMRL*, 6(2), 2006.

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

Vedaldi, A. and Zisserman, A. Efficient additive kernels via explicit feature maps. *IEEE TPAMI*, 34(3), 2011.

Wolf, Lior, Jhuang, Hueihan, and Hazan, Tamir. Modeling appearances with low-rank svm. In *CVPR*, 2007.

Zhu, L.L., Chen, Y., Torralba, A., Freeman, W., and Yuille, A. Part and appearance sharing: Recursive compositional models for multi-view multi-object detection. In *CVPR*. IEEE, 2010.

Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 2005.