Training deformable part models with decorrelated features

Ross Girshick and Jitendra Malik
UC Berkeley
{rbg, malik}@eecs.berkeley.edu

Abstract

*In this paper, we show how to train a deformable part model (DPM) fast—typically in less than 20 minutes, or four times faster than the current fastest method—while maintaining high average precision on the PASCAL VOC datasets. At the core of our approach is "latent LDA," a novel generalization of linear discriminant analysis for learning latent variable models. Unlike latent SVM, latent LDA uses efficient closed-form updates and does not require an expensive search for hard negative examples. Our approach also acts as a springboard for a detailed experimental study of DPM training. We isolate and quantify the impact of key training factors for the first time (*e.g.*, How important are discriminative SVM filters? How important is joint parameter estimation? How many negative images are needed for training?). Our findings yield useful insights for researchers working with Markov random fields and part-based models, and have practical implications for speeding up tasks such as model selection.*

1. Introduction

Over the past several years deformable part models (DPMs) [12, 14] have emerged as one of the leading methods for object detection [10]. Significant strides have been made in accelerating DPM detection speed, further broadening their appeal. Popular approaches include cascades [11], coarse-to-fine processing [22], and sparse representations [23, 24]. In contemporaneous work, a hashing technique promises fast detection with *100,000* DPMs on a single workstation [6]. In stark contrast, relatively little attention has been paid to accelerating DPM *training*. Yet slow training is often a significant bottleneck in experimental research.

Training a DPM involves optimizing a latent SVM (LSVM). The LSVM objective function is not convex, and in practice a stationary point is found by solving a sequence of large-scale convex subproblems [1, 12]. This optimization heuristic can be slow, with most of the time spent searching for "hard negative" instances (a process commonly called *bootstrapping* or *data mining*).

Some of the aforementioned techniques can accelerate training via fast detection. However, these methods often involve approximations (such as hard thresholds in the DPM cascade [11]) that make learning unstable. In this paper we take a more direct approach: we develop techniques that accelerate training by *avoiding* most of the typically requisite data mining. Around this core goal we present:

- fast, approximate DPM training using *latent LDA*—a novel generalization of linear discriminant analysis;
- a detailed experimental study of what factors are important for training a DPM; and
- hybrid large-margin latent LDA training for learning DPMs 4x faster than the current fastest method [14], while maintaining high average precision.

Moreover, our approach is complementary to exact methods for speeding up detection (*e.g.*, [8]), making further acceleration possible.

Recently, Hariharan *et al*. [15] attacked the problem of accelerated training for rigid (non-part-based) object detectors. Their work introduced an efficient method for estimating the covariance matrix of $n_x \times n_y$ patches of histogram of oriented gradients (HOG) features [5]. Using this technique, they learned the parameters of a HOG filter by linear discriminant analysis (LDA) instead of the usual, comparatively slow, SVM training route. Their experiments on the INRIA pedestrian dataset [5] demonstrate that an LDA-HOG filter can produce good results, with an average precision (AP) of 75% *vs*. the 80% achieved by SVM-HOG. The main advantage of LDA-HOG over SVM-HOG is training speed; the former approach does not require searching for hard negative instances.

Can a similar technique accelerate DPM training, and if so, what is the training-time *vs*. detection-accuracy tradeoff? This paper answers these questions and at the same time presents a careful dissection of DPM training, providing new insight into which aspects are important for high detection accuracy.

Our approach re-envisions the LSVM optimization problem so that it involves only positive examples and coarse background statistics. Optimizing our proposed objective

involves alternating between imputing latent labels and updating model parameters, like an LSVM. However, when the latent labels are fixed, the optimal parameter vector can be solved for in closed form. To connect our optimization problem to LDA, we show that if the input features are whitened, then the optimal parameters form a set of LDA classifiers. We call this natural counterpart to latent SVM "latent LDA" (LLDA).

As foreshadowed by the INRIA pedestrian experiments described above, we find that LLDA DPMs achieve surprisingly good AP performance on the much more challenging PASCAL VOC datasets [10], *even though no hard negative examples are used*. Training an LLDA DPM with 600 examples takes less than 8 minutes on a six-core machine. For comparison, training the recently proposed exemplar SVM [17] with 600 examples takes about 4 hours on 100 cores, and yields a similar mean AP (18.0% *vs*. 19.8%, respectively).

However, there is still a sizable AP gap between DPMs trained with LLDA and LSVM. The second contribution of this paper is to experimentally analyze this difference. Our analysis breaks the AP gap down into two factors: generative LDA *vs*. discriminative SVM training; and joint *vs*. two-stage training [18, 20]. We find that each factor plays an important role, and in particular, two-stage training (*i.e*., learning each filter independently and then stitching them into a model) can dramatically underperform joint training. This finding can be interpreted more broadly as a cautionary tale showing that joint training of a Markov (or conditional) random field can turn an underperforming technique into one that is state-of-the-art.

Our analysis also uncovers a surprising result: even though object detection performance with LLDA DPMs trails behind LSVM, *they impute equally good latent labels*. These labels (*i.e*., a choice of DPM mixture component and filter positions for each positive example) are "good" in the sense that if they are used as ground truth in large-margin training, they yield performance equal to a DPM trained end-to-end with LSVM. This observation leads to our third contribution—a faster way to train DPMs with no loss in mean AP performance. We "warm start" LSVM training by replacing the usual sequence of convex subproblems with latent LDA. Then, after LLDA has converged, we solve *one* large-margin training problem with data mining. We call this hybrid large-margin / LLDA method LM-LLDA.

A simple, alternative approach to speed up training is to subsample the negative training images. Surprisingly, no previous work has studied how DPM detection accuracy varies as a function of the number of negative training examples (*cf.* [27]). We investigate this dependence both in the case of single HOG filter models on the INRIA dataset and with DPMs on the PASCAL VOC datasets. Interestingly, very few negative images are needed to get good performance. For INRIA, data mining from *just 64* negative images—instead of the customary 1218—yields 76.2% AP, which slightly outperforms LDA-HOG. Similar results hold true for DPMs. However, in LSVM training data mining is performed many times, once for each subproblem solved, making training slow even with subsampling. The LLDA warm start can be viewed as subsampling to the limit where no negative examples are used during all but the last iteration of training. Naturally, the two methods can be combined. Subsampling in the last iteration, after the warm start, allows us to train DPMs achieving high performance (32.3% mAP) with a median training time under 20 minutes (4x faster than [14]).

2. Training without negative examples

In this section, we develop latent LDA, an alternative to latent SVM that can quickly train DPMs without any hard negative examples. We begin by reviewing latent SVM. A more detailed account can be found in [12].

2.1. Latent SVM primer

Consider a set of labeled training examples $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, where each x_n comes from an input space $\mathcal{X}$ and y_n is a binary label in $\{-1, 1\}$. Each input element $x \in \mathcal{X}$ also has an unobserved label $z \in \mathcal{Z}(x)$, which is latent during both training and testing. Given a particular x, the sign of $f_{\mathbf{w}}(x) = \max_{z \in \mathcal{Z}(x)} \mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(x, z)$ is used to classify it and simultaneously predict its hidden label (by computing the argmax). Here, $\boldsymbol{\varphi}(x, z)$ is a vector-valued function that computes features for the pair (x, z). LSVM is a method for learning the parameters $\mathbf{w}$ of $f_{\mathbf{w}}$.

To ease our subsequent discussion of DPMs, we introduce some notational conventions that are well-suited for the DPM model parameterization. Without loss of generality, we assume $\mathbf{w}$ is divided into K "mixture components" such that $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_K)$. Correspondingly, a latent label is a pair $z = (k, h)$, where $k \in \{1, \ldots, K\}$ specifies a mixture component and h are additional latent labels. For a DPM, k identifies a pose or viewpoint component while h specifies the image position and scale placement of each filter used by component k. Finally, we assume the feature function $\boldsymbol{\varphi}(x, z)$ is block sparse, having one non-zero block with values $\boldsymbol{\varphi}_k(x, h)$ such that the property $\mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(x, (k, h)) = \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\varphi}_k(x, h)$ holds (see Eq. 14).

Recent releases of the DPM code use a variant of LSVM where the standard ℓ_2 regularization penalty is replaced by the "max-component" regularizer [13], which penalizes the mixture component $\mathbf{w}_k$ with the largest ℓ_2 norm. This objective function can be written as

$$\frac{1}{2} \max_k \|\mathbf{w}_k\|^2 + C \sum_{n=1}^{N} \max\{0,\ 1 - y_n f_{\mathbf{w}}(x_n)\}, \quad (1)$$

where C is the usual regularization tradeoff. For training sets that are not linearly separable, max-component regularization equalizes the norms of the components. This can be seen intuitively: if one component has a lower norm than the max, and has non-zero hinge loss, it can grow a bit without penalty in order to reduce its loss. Experimentally, this equalization stabilizes LSVM training by preventing mixture components from losing all of their examples and "evaporating" (*cf.* [16]).

The standard optimization heuristic for finding a stationary point of Eq. 1 is coordinate descent [1, 12]. The coordinate descent algorithm alternates between fixing a single labeling for each positive example (given a current estimate of $\mathbf{w}$), which results in a convex optimization subproblem, and then updating $\mathbf{w}$ to be the solution to that subproblem. These iterations are repeated until the process converges.

Unfortunately, the coordinate descent subproblems can be very time consuming to solve. For example, when training a DPM each subproblem requires densely scanning a large set of images in order to extract a small number of hard negative examples. This process takes around 5-10 seconds per image, and may require several passes over thousands of images before converging.

2.2. What can we do with only positive examples?

Let $\mathcal{P} = \{n : y_n = 1\}$ be the index set of positive examples in $\mathcal{D}$. We define the following optimization problem restricted to $\mathcal{P}$.

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\sum_{n \in \mathcal{P}} f_{\mathbf{w}}(x_n) \\ \text{s.t.} \quad & \|\mathbf{w}_k\|^2 = 1, \quad \forall k \end{aligned} \tag{2}$$

We've intentionally maintained a direct parallel between this objective and LSVM (Eq. 1): in place of max-component regularization, we constrain all components to have unit norm; we removed the per-example loss for the negative training instances; and for each positive example, the objective drives $\mathbf{w}$ towards a solution that gives at least one latent labeling a high score under $f_{\mathbf{w}}$.

As with LSVM, the objective function in Eq. 2 is not convex in $\mathbf{w}$ (due to the sum of piecewise-linear concave functions in the objective and the quadratic constraints). A stationary point can be found by applying the concave-convex procedure (CCCP) [26] to the Lagrangian of Eq. 2.

Each CCCP iteration has two steps. In step (1) of iteration t, we replace $f_{\mathbf{w}}(x_n)$ in Eq. 2 by $\mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(x_n, z_{n\mathbf{w}_{(t)}})$, where $z_{n\mathbf{w}_{(t)}} = \operatorname{argmax}_{z \in \mathcal{Z}(x_n)} \mathbf{w}_{(t)}^{\mathrm{T}} \boldsymbol{\varphi}(x_n, z)$ is the label imputed with the current model parameters $\mathbf{w}_{(t)}$. This substitution places a linear upper bound on $-f_{\mathbf{w}}(x_n)$ at $\mathbf{w}_{(t)}$ (up to a constant that does not depend on $\mathbf{w}$), and results in the linear objective

$$g(\mathbf{w}; \mathbf{w}_{(t)}) = -\sum_{n \in \mathcal{P}} \mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(x_n, z_{n\mathbf{w}_{(t)}}). \tag{3}$$

In step (2), we update the current solution according to

$$\mathbf{w}_{(t+1)} = \operatorname*{argmin}_{\mathbf{w}} g(\mathbf{w}; \mathbf{w}_{(t)}) \quad \text{s.t.} \quad \|\mathbf{w}_k\|^2 = 1, \quad \forall k. \tag{4}$$

This optimization problem has an efficiently computable closed-form solution. By adding a Lagrange multiplier λ_k for each constraint and equating the gradient with respect to each $\mathbf{w}_k$ to zero, we arrive at

$$\mathbf{w}_k \propto \sum_{n : k_{n\mathbf{w}_{(t)}} = k} \boldsymbol{\varphi}_k(x_n, h_{n\mathbf{w}_{(t)}}), \tag{5}$$

where we've explicitly written the current label predictions as $(k_{n\mathbf{w}_{(t)}}, h_{n\mathbf{w}_{(t)}}) := z_{n\mathbf{w}_{(t)}}$ and the summation is over examples assigned to component k. Computing $\mathbf{w}_k$ as the sum in Eq. 5 and then scaling it to unit ℓ_2 norm is equivalent to selecting λ_k to satisfy its constraint. These two CCCP steps are repeated until the linear upper bounds do not change (or the relative change in Eq. 2 is small). This process minimizes an upper bound on our original objective.

2.3. The LDA connection

The optimization algorithm just described is reminiscent of *spherical* k-means clustering [7]. In each iteration, we use fixed cluster "centers" $\mathbf{w}_k$ to infer latent cluster assignments (and, in our case, additional latent labels), and then with those fixed, we update the centers $\mathbf{w}_k$ to be the (normalized) means of the new clusters. At a first glance it seems doubtful that this algorithm will result in a reasonable detector. However, we show that by applying a simple whitening transformation to the training features, the algorithm's output has an appealing connection to linear discriminant analysis.

To achieve this goal we need to cheat slightly and compute some coarse statistics of all training examples (including those from the negative class). Specifically, we need the sample mean $\boldsymbol{\mu}$ of the negative examples and the sample covariance matrix $\mathbf{S}$ of the entire training set. Since, by assumption, our feature vectors are non-zero within only one component's span, computing these statistics for each mixture component independently is sufficient ($\mathbf{S}$ is block-diagonal, with blocks $\mathbf{S}_k$). In Section 3, we illustrate how $\boldsymbol{\mu}_k$ and $\mathbf{S}_k$ can be modeled and estimated efficiently in the case of DPMs. Our estimates will come in the form of basic building blocks that can synthesize the mean and covariance for a mixture component with any number of filters, each with any shape. Moreover, our estimates are class independent: in datasets with large class imbalance, such as any typical detection dataset, the negative examples' sample mean $\boldsymbol{\mu}$ can be computed from all examples, since the minority class lends a negligible contribution. Together, these observations allow us to compute $\boldsymbol{\mu}_k$ and $\mathbf{S}_k$ offline, once-and-for-all, for all classes.

Assuming for now that we have these statistics in hand, we apply a whitening transformation to (approximately) decorrelate the features of the positive examples:

$$\tilde{\boldsymbol{\varphi}}_k(x,h) = \mathbf{S}_k^{-\frac{1}{2}} \left(\boldsymbol{\varphi}_k(x,h) - \boldsymbol{\mu}_k\right). \tag{6}$$

By convention, variables topped with a tilde reside in the whitened feature space. Plugging the transformed features into the update for $\mathbf{w}_k$ (Eq. 5), we see that each model component $\tilde{\mathbf{w}}_k$ takes the form

$$\tilde{\mathbf{w}}_k = Z_k^{-1} \sum_{n:k_n=k} \tilde{\boldsymbol{\varphi}}_k(x_n,h_n) \tag{7}$$

$$= Z_k^{-1}\mathbf{S}_k^{-\frac{1}{2}} \sum_{n:k_n=k} \left(\boldsymbol{\varphi}_k(x_n,h_n) - \boldsymbol{\mu}_k\right), \tag{8}$$

where Z_k^{-1} scales $\tilde{\mathbf{w}}_k$ to unit length and the summation is over examples assigned to component k.

Given parameters $\tilde{\mathbf{w}}$ learned in the whitened space, an input x is classified according to $f_{\tilde{\mathbf{w}}}(x) = \max_{(k,h)\in\mathcal{Z}(x)} \tilde{\mathbf{w}}_k^{\mathrm{T}}\tilde{\boldsymbol{\varphi}}_k(x,h)$. Looking into the dot product, some basic algebra shows that

$$\tilde{\mathbf{w}}_k^{\mathrm{T}}\tilde{\boldsymbol{\varphi}}_k(x,h) = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\varphi}_k(x,h) + b_k, \tag{9}$$

where

$$\mathbf{w}_k = Z_k^{-1}\mathbf{S}_k^{-1} \sum_{n:k_n=k} \left(\boldsymbol{\varphi}(x_n,z_n) - \boldsymbol{\mu}_k\right) \tag{10}$$

$$b_k = -\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\mu}_k. \tag{11}$$

Recall that in standard two-class LDA we assume the classes have a shared covariance matrix $\mathbf{S}$, but with separate class means $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$. The general form of the resulting LDA classifier is $\mathbf{w} \propto \mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ [2]. Eq. 10 says a dot product with $\tilde{\mathbf{w}}_k$ in the whitened feature space is equivalent to a dot product with a vector $\mathbf{w}_k$—*which has the form of an LDA classifier*—in the unwhitened feature space, plus a bias (Eq. 11). Under this interpretation, the function $f_{\mathbf{w}}(x) = \max_{(k,h)\in\mathcal{Z}(x)} \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\varphi}_k(x,h)$ scores an example by picking the best LDA component classifier and latent label pair. Overall, the classifier can be thought of as the "latent LDA" counterpart to a latent SVM.

2.4. Large-margin LLDA (LM-LLDA)

One immediate application of latent LDA is as a fast initialization, or "warm start," for latent SVM training. We can use LLDA to quickly generate the unobserved labels (k_n, h_n) for each positive example, and then treat those labels as if they were the observed ground truth in the convex large-margin objective

$$\frac{1}{2}\max_k \|\mathbf{w}_k\|^2 + C\sum_{n\in\mathcal{P}} \left\{1 - \mathbf{w}_{k_n}^{\mathrm{T}}\boldsymbol{\varphi}_k(x_n,h_n)\right\}_+ \tag{12}$$

$$+ C\sum_{n\in\mathcal{N}} \left\{1 + \max_{z\in\mathcal{Z}(x_n)} \mathbf{w}^{\mathrm{T}}\boldsymbol{\varphi}(x_n,z)\right\}_+,$$

where $\mathcal{N} = \{n : y_n = -1\}$ is the index set of negative examples and $\{m\}_+$ denotes $\max\{0,m\}$. This objective is exactly the subproblem solved in coordinate descent for LSVM. Optionally, one could run multiple LSVM coordinate descent iterations after the warm start, similar to how expectation maximization is used to initialize a latent structural SVM in [21]. But our experiments in Section 4 show that minimizing Eq. 12 without further iterations already yields excellent results. We dub this method LM-LLDA.

3. Application to deformable part models

We now describe how LLDA and its whitening transformation can be applied to deformable part models. We begin with a brief overview of DPM weight vector and feature function parameterization.

DPM weight vector parameterization. A DPM is composed of K mixture components, each of which has a root filter and P part filters. Each part filter has a canonical offset ("anchor") relative to its root filter. During detection, the parts can shift relative to their anchor positions at a cost. This cost is modeled independently for each part as an axis-aligned, convex quadratic function. This structure is encoded in a model's weight vector, which is parameterized as $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_K)$, where each per-component weight vector $\mathbf{w}_k$ is composed of the parameter blocks:

$$\mathbf{w}_k = (\underbrace{\mathbf{f}_{k0},\ldots,\mathbf{f}_{kP}}_{\text{filter weights}}, \underbrace{\mathbf{d}_{k1},\ldots,\mathbf{d}_{kP}}_{\text{deformation weights}}, \underbrace{\beta_k}_{\text{bias}}). \tag{13}$$

DPM feature parameterization. Matching the weight vector parameterization, the feature function has the form

$$\boldsymbol{\varphi}\left(x,(k,h)\right) = (\,0,\ldots,0,\ \underbrace{\boldsymbol{\varphi}_k(x,h)}_{\text{span of }\mathbf{w}_k},\ 0,\ldots,0\,). \tag{14}$$

Each per-component feature function $\boldsymbol{\varphi}_k(x,h)$ contains: HOG features $\boldsymbol{\phi}_{kp}$ extracted from image x at the filter placements listed in h; and deformation features $\boldsymbol{\delta}_{kp}(h) = -(d_x^2,\ d_x,\ d_y^2,\ d_y)$, where (d_x, d_y) are displacements relative to the p-th part's anchor, yielding

$$\boldsymbol{\varphi}_k(x,h) = (\ \underbrace{\boldsymbol{\phi}_{k0}(x,h),\ldots,\boldsymbol{\phi}_{kP}(x,h)}_{\text{HOG features}}, \tag{15}$$

$$\underbrace{\boldsymbol{\delta}_{k1}(h),\ldots,\boldsymbol{\delta}_{kP}(h)}_{\text{deformation features}}, \underbrace{1}_{\text{bias}}\ ).$$

Modeling and estimating $\mathbf{S}_k$ and $\boldsymbol{\mu}_k$. In principle, we would like to compute

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}} \frac{1}{|\mathcal{H}_k(x_n)|} \sum_{h\in\mathcal{H}_k(x_n)} \boldsymbol{\varphi}_k(x_n,h) \tag{16}$$

$$\mathbf{S}_k = \frac{1}{|\mathcal{D}|}\sum_{n=1}^{N} \frac{1}{|\mathcal{H}_k(x_n)|} \sum_{h\in\mathcal{H}_k(x_n)} \mathring{\boldsymbol{\varphi}}_k(x_n,h)\mathring{\boldsymbol{\varphi}}_k(x_n,h)^{\mathrm{T}}$$

where $\mathcal{H}_k(x)$ specifies all valid filter configurations for x and $\mathring{\boldsymbol{\varphi}}_k(x,h) = \boldsymbol{\varphi}_k(x,h) - \boldsymbol{\mu}_k$ are centered features.

For a typical DPM, a full covariance matrix would have around 10^9 parameters, per component, which is clearly infeasible to work with. We sidestep this problem with the following modeling assumptions: (1) HOG features and deformation features are uncorrelated; (2) deformation features are uncorrelated across parts; and (3) HOG features are uncorrelated across parts. The last assumption is somewhat coarse since DPM parts are spatially adjacent and sometimes overlap. However it has a significant computational advantage: we can model the covariance matrix of each part independently. These assumptions give $\mathbf{S}_k$ a block-diagonal structure, which allows us to carry out the required matrix computations on relatively small matrices, block-by-block.

For blocks corresponding to HOG features $\boldsymbol{\phi}_{kp}$, we use Hariharan *et al.*'s [15] method for estimating the mean and covariance matrix of $n_x \times n_y$ rectangular patches of HOG features. Their approach assumes translational invariance, so all HOG cells separated by a fixed spatial offset share the same covariance parameters. These covariance "building blocks" are computed for all offsets within a fixed radius (*e.g.*, 10 cells), and can then be used to synthesize the diagonal blocks of $\mathbf{S}_k$ corresponding to the HOG features of each part. This flexibility decouples the estimation of the HOG covariance parameters from the structure (*i.e.*, number of filters and their shapes) of any particular mixture component. For $\boldsymbol{\mu}_k$, we follow [15] and use the mean of a single HOG cell, repeated the appropriate number of times. These parameters were estimated on PASCAL VOC 2010 trainval.

The deformation features $\boldsymbol{\delta}_{kp}$ are low-dimensional, but present a different challenge: unlike the natural images used for HOG feature statistics, there is no source of "natural" deformations for computing their background model. We take a simple approach to this problem: fix the deformation costs to $(0.01, 0, 0.01, 0)$ for all parts, and do not update them during LLDA training. These values are the initialization used in the DPM code [14].

The per-component biases β_k do not need to be modeled, and are set equal to the sum of the biases (Eq. 11) coming from each filter p in component k, *i.e.*, $\beta_k = \sum_{p=0}^{P} b_{kp}$.

4. Experimental results and analysis

We implemented our experiments on top of the publicly available DPM software [14] with one novel modification. By default the DPM code pads each level of a HOG feature pyramid with several cells that are set to zero. This padding allows filters to slide outside the image, enabling detection of partially truncated objects. Motivated by the intuition that a filter placed outside the image should have the same *expected* score as a filter placed in background, we pad with the *mean* HOG feature vector, instead of zero. This insight boosts the baseline system from 33.7% to 34.5% mAP on PASCAL 2007 (Table 1 LSVM-$\boldsymbol{\mu}$ *vs.* LSVM).[1] All timed experiments were performed on one reference machine with a six-core 3.2GHz Intel i7 processor. All design decisions were validated on PASCAL 2007, fixed, and then run once on PASCAL 2011. Source code will be available.

LLDA DPMs (LLDA-0). With these preliminaries in place we evaluate our first method, LLDA-0—pure latent LDA without any hard negative examples. Executing the same initialization routine as the default DPM training pipeline[2] (*i.e.*, clustering positives into three aspect ratios and automatically grouping each aspect into a left/right split), LLDA-0 achieves remarkably good performance, 18.0% mAP (Table 1), considering that training takes less than 8 minutes for a class with 600 examples on a single six-core machine. As a reference point, exemplar SVM [17] does only slightly better, 19.8% mAP, even though it makes extensive use of hard negatives and training 600 exemplars takes about 4 hours on a 100-core cluster. LLDA-0 DPMs are also competitive with the multi-component (MC) LDA (17.0% mAP) and exemplar LDA detectors (19.1% mAP) [15]. Both systems in [15] *rescore* LDA-HOG filter detections using a second-layer SVM trained with negative examples, and thus are not "pure" LDA methods.

4.1. Analyzing the LLDA-LSVM performance gap

Our LLDA-0 DPM is different from an LSVM DPM in four fundamental respects: (1) deformation costs are not learned; (2) filters are estimated by LDA; (3) furthermore, the filters are learned independently due to our modeling assumptions; and (4) the labels imputed by LLDA might differ from those imputed by LSVM. We designed experiments to isolate the effects of these differences using two tools: large-margin LLDA and two-stage training.

LM-LLDA (Section 2.4) naturally bridges the gap between LLDA and LSVM training, up to differences in imputed labels. Our second tool is two-stage training (see [20] for an overview), which allows us to hold subsets of DPM parameters fixed while optimizing the others in the large-margin phase of LM-LLDA (*i.e.*, minimizing Eq. 12). Originally designed as a technique to speed up training of graphical models [18], two-stage training is also an ideal tool for understanding what makes a model perform well. By holding strategic subsets of parameters fixed we can "interpolate" between LLDA and LSVM.

In the following experiments, we use the latent labels generated by LLDA-0. This implies each experiment uses *exactly* the same feature vectors for the positive examples (as determined by the LLDA labeling). We also fix a set of negative training images, with an average of 2300 per class.

[1] By default [14] includes a feature set to 1 inside the padding and 0 inside the image. This feature is designed to allow filters to learn a score bias for placements outside the image. Our results indicate it is insufficient.

[2] To be pedantic: we use LLDA (*not LSVM*) for all initialization steps.

	our methods					baselines			
	LLDA-0	LLDA-1	LLDA-2	LLDA-3	LSVM-μ	LSVM [14]	MC-LDA [15]	ELDA [15]	ESVM [17]
mAP '07 test (%)	18.0	24.4	30.7	**34.5**	**34.5**	33.7	17.0	19.1	19.8
mAP '11 val (%)	13.2	19.9	25.4	**28.6**	28.2	27.4	n/r	n/r	n/r
joint training?	no	no	no	yes	yes	yes	no	no	no
hard neg?	no	yes	yes	yes	yes	yes	yes	yes	yes

Table 1. Performance of our methods and baselines on PASCAL VOC 2007 *test* and 2011 *val* [10] in mean AP (n/r: not reported). Method summary. **LLDA-0**: latent LDA (no negative examples). **LLDA-1**: LLDA-0 followed by large-margin training of deformation costs, filter calibration weights, and biases. **LLDA-2**: same as the LLDA-1, but with independently trained SVM filters replacing the LDA filters. **LLDA-3**: LLDA-0 followed by large-margin training of *all* parameters (equivalent to **LM-LLDA**). **LSVM-μ**: [14] with mean HOG padding. **MC-LDA** and **ELDA**: multi-component and exemplar LDA from [15]. All results are without context rescoring [12, 15, 17].

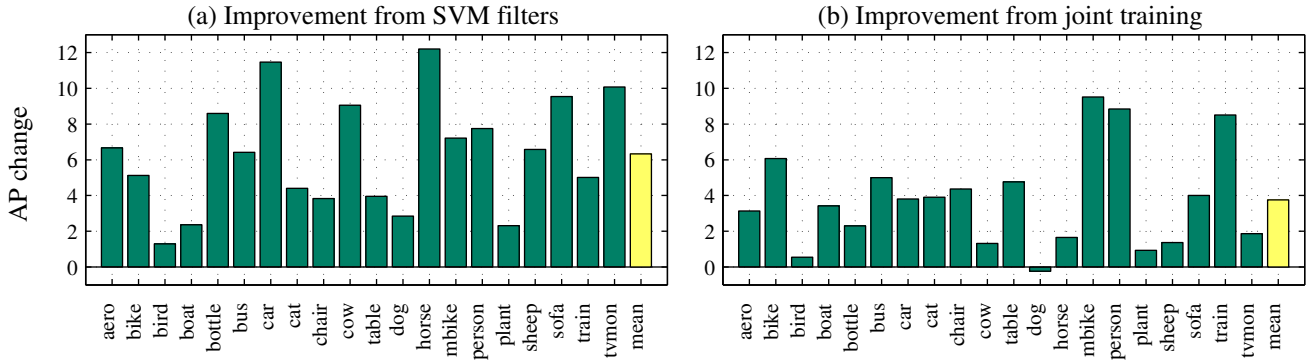

Figure 1. Absolute improvement in average precision from (a) SVM *vs*. LDA filters and (b) joint *vs*. two-stage (independent) training. The boost from discriminative filters is substantial, and likely greater than one might expect from experiments in [15]. Independent filter learning dramatically underperforms joint training, suggesting other part-based models (*e.g*., poselets [4]) may benefit from joint learning.

Calibrating deformations, filters, and biases (LLDA-1). In two-stage training of a graphical model, the unary potentials (*i.e*., a DPM's filters) are learned independently during a "first stage" of training. Then, keeping the unaries fixed, their real-valued outputs are used as features in a "second stage" model. Our second stage model is parameterized by one scalar multiplier per unary feature, deformation costs, and per-component biases. This two-stage approach allows us to adjust the relative magnitudes of the filters, without changing their directions, as well as to tune deformation costs and component biases.

Our unaries are the LLDA-0 filters. Discriminatively calibrating these fixed filters boosts mAP substantially from 18.0% to 24.4% (Table 1 LLDA-1). A more fine-grained ablation shows learning only the per-component biases yields 19.4% mAP; biases plus deformation costs gives 23.0% mAP; and biases plus filter multipliers produces 23.2% mAP. LLDA-1 is similar to the multi-component and exemplar LDA detectors since all systems use a second stage to discriminatively calibrate LDA-HOG filter responses, but achieves a much higher mAP due to the DPM's parts.

One subtle, but important point, is that we need to regularize each second stage unary multiplier such that its cost is equal to the cost of scaling the original filter. For a scalar α that multiplies the output of a fixed filter $\mathbf{f}$, we set the regularization penalty for α to $\|\mathbf{f}\|^2$. To verify this approach, we performed two-stage training using the *jointly* trained filters from the LSVM-μ models as unaries. With regularization penalties correctly set, two-stage training matches the original mAP (34.8% *vs*. 34.5%).

LDA *vs*. SVM filters (LLDA-2). We use the same two-stage training methodology as before, but this time we replace each LDA filter with a linear SVM. For training these SVMs, the positive examples come from the labels generated by LLDA-0 (thus the LDA and SVM filters use exactly the same positive feature vectors). The negative examples are all subwindows of the negative images. The second stage model parameterization is the same as in LLDA-1, making discriminative SVM filters the only difference. Table 1 shows that SVM filters boost mAP from 24.4% to 30.7%. The relative change is much larger than what was observed in the single filter experiments on INRIA pedestrians [15]. Figure 1(a) shows the per-class breakdown.

Independent *vs*. joint training (LLDA-3). Discriminative calibration and replacing LDA filters with SVM filters substantially closes the gap between LLDA and LSVM. These results suggest the remaining difference stems from either a less effective latent labeling of the positives or from

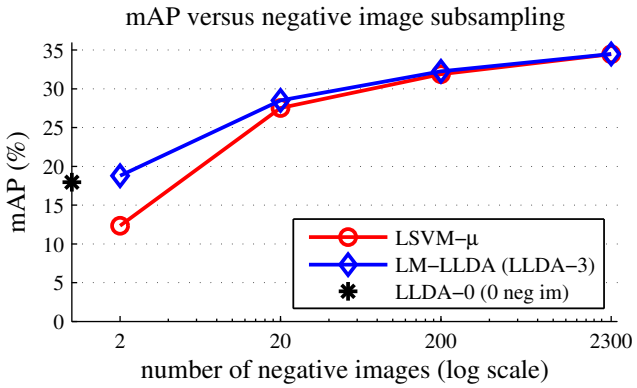

Figure 2. Mean average precision decays gracefully as the number of negative images decreases (note the log scale). LM-LLDA performs better than LSVM-μ when very few negative images are available. See text for discussion.

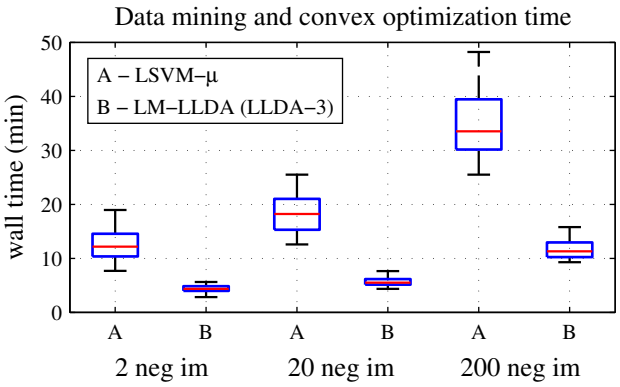

Figure 3. LM-LLDA spends one-third as much time on data mining and convex optimization compared to LSVM-μ across various levels of negative image subsampling. The boxes span the 25th, 50th, and 75th percentiles of the time distribution over 20 classes.

independent filter training. We can easily remove the independence assumption by optimizing *all* model parameters jointly (*i.e.*, one-stage instead of two-stage training when minimizing Eq. 12). Our final variant, LLDA-3, is therefore exactly the same as LM-LLDA.

Joint training completely closes the AP gap (Table 1 LLDA-3 and Figure 2). This implies that the latent labels imputed by LLDA-0 are equivalent, from a final mAP perspective, to the ones obtained by coordinate descent with LSVM. Figure 1(b) shows the improvement from joint training on a per-class basis. On average, joint parameter estimation boosts AP by 4 points, absolute. The difference is dramatic for some classes, such as motorbike and person, where the improvement is nearly 10 points. Our findings agree with Endres *et al.*'s [9] and Yang and Ramanan [25] and add a note of caution to the suggestion in [19] that joint and two-stage training perform equally well.

4.2. Subsampling negative images

LDA-HOG [15] and LLDA DPM can be seen as methods for training detectors in the limit of subsampling where no negative images are used. Surprisingly, the question of how detector accuracy varies with the number of negative training examples has not been carefully studied. The focus is typically on positive examples (*e.g.*, [27]).

Effect on AP. We first look at a Dalal and Triggs style detector on the INRIA pedestrian dataset [5]. Data mining from two negative images yields an AP of 29.6% (20.4%[3]). With eight images, AP ascends to 67.4% (7.1%). At 64 images, the detector surpasses LDA-HOG, achieving an AP of 76.2% (0.7%). AP slowly climbs to 80% while increasing the number of negative images 20-fold to the full set of 1218. In retrospect, the steep gradient of this curve and saturation at a small number of images is not surprising.

[3]Standard deviation over 20 draws of negative images.

The success of LDA-HOG implies that coarse background statistics, which the SVM efficiently extracts from a small set of images, are sufficient to get most of the way toward maximum performance on INRIA.

Curves showing a similar story for DPMs are readily visible in Figure 2. Using only 200 negative images—*an order of magnitude fewer than typically used*—results in 32.3% mAP for LM-LLDA and 31.9% mAP for LSVM-μ. Both are close to the maximum performance of 34.5% achieved using around 2300 negative images. Even with few negative images, mAP is relatively stable over random draws due to averaging (in contrast with the single-class INRIA experiments). At two negative images, LM-LLDA had a standard deviation of only 0.6% mAP over five samples.

Another notable trend is LM-LLDA gains further advantage over LSVM as the number of negative images decreases. Our hypothesis is that with very few (*e.g.*, 2) negative images LSVM training begins to impute noisy labels for the positive examples. The positive labels used by LM-LLDA, in contrast, are invariant to the number of negatives. A final pattern, which might not even appear worth mentioning at first, is that mAP increases with the number of negative examples. Moreover, this trend holds for each class on its own. However, this finding contradicts [3], where Blaschko *et al.* find that AP *decreases* as the number of negative examples increases when binary hinge loss is used instead of their proposed ranking loss. One conjecture is that the models in [3] did not have a learnable bias.

Effect on training time. Figure 3 shows that LM-LLDA reduces the time spent in data mining and convex optimization by a factor of three averaged over all classes. The rest of training is spent imputing labels for the positives, which takes the same amount of time in both cases. The overhead from LLDA parameter estimation is less than 10 seconds. End-to-end training time acceleration ranges from a factor

of 1.6x with two negative images to 2.0x with 200 negative images. Relative to the publicly available DPM code [14], which is currently the fastest option for training DPMs, we can learn models nearly 4x faster (with median training wall times of 19.9 minutes *vs*. 77.9 minutes) while maintaining a high mAP of 32.3%. Our results compare favorably with Vedaldi and Zisserman [24], for example, who speed up training by less than a factor of two, using product quantization, and have a final mAP of 27.7%.

5. Conclusion

Long training times are often a bottleneck in experimental research. Even modest improvements accelerate experimentation, and in turn, research progress. The four-fold speedup for DPM training achieved in this paper will allow researchers working to improve DPMs, using them as building blocks in a larger system, or applying them to new datasets, to iterate more quickly. Moreover, our latent LDA approach is general and applies to latent variable models beyond DPM. Latent LDA also provides the basis for deconstructing DPM training in order to gain experimental insight into part-based models. Notably, we find that joint parameter estimation is fundamental. Existing approaches that train parts independently, such as poselets [4], will likely benefit from joint learning. Secondly, we often waste significant time mining hard negative examples from excessively large image sets. Model selection, for instance, should be performed using a small set of negative images (while achieving nearly full AP performance), while only the final model should be trained on the full set, if at all.

Acknowledgments. This research was supported by MURI N000014-10-1-0933 and the MindsEye project.

References

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

[2] C. M. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006.

[3] M. Blaschko, A. Vedaldi, and A. Zisserman. Simultaneous object detection and ranking with weak supervision. In *NIPS*, 2010.

[4] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.

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

[6] T. Dean, J. Yagnik, M. Ruzon, M. Segal, J. Shlens, and S. Vijayanarasimhan. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013.

[7] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 2001.

[8] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *ECCV*, 2012.

[9] I. Endres, V. Srikumar, M.-W. Chang, and D. Hoiem. Learning shared body plans. In *CVPR*, 2012.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[11] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

[12] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

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

[14] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/~rbg/latent-release5/.

[15] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.

[16] M. Hoai and A. Zisserman. Discriminative subcategorization. In *CVPR*, 2013.

[17] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[18] A. McCallum and C. Sutton. Piecewise training of undirected models. In *UAI*, 2005.

[19] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in crf-based approaches to object class image segmentation. In *ECCV*, 2010.

[20] S. Nowozin and C. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 2011.

[21] S. N. Parizi, J. G. Oberlin, and P. F. Felzenszwalb. Reconfigurable models for scene recognition. In *CVPR*, 2012.

[22] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011.

[23] H. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. In *ECCV*, 2012.

[24] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

[25] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures-of-parts. *TPAMI*, 2012.

[26] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 2003.

[27] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? In *BMVC*, 2012.