# Group Norm for Learning Structured SVMs with Unstructured Latent Variables

Daozheng Chen*
UMD, College Park
dchen@cs.umd.edu

Dhruv Batra
Virginia Tech
dbatra@vt.edu

William T. Freeman
MIT CSAIL
billf@mit.edu

## Abstract

*Latent variables models have been applied to a number of computer vision problems. However, the complexity of the latent space is typically left as a free design choice. A larger latent space results in a more expressive model, but such models are prone to overfitting and are slower to perform inference with. The goal of this paper is to regularize the complexity of the latent space and* learn *which hidden states are really relevant for prediction. Specifically, we propose using group-sparsity-inducing regularizers such as $\ell_1$-$\ell_2$ to estimate the parameters of Structured SVMs with unstructured latent variables. Our experiments on digit recognition and object detection show that our approach is indeed able to control the complexity of latent space without any significant loss in accuracy of the learnt model.*

## 1. Introduction

Fully supervised algorithms are a useful but perhaps an unnatural abstraction for computer vision problems. In reality, we almost never have complete supervision – there are always some variables relevant to the problem that not annotated in our datasets. For example, consider the task of training a person detector. Standard benchmarks only provide bounding box annotations indicating the presence of people. However, people tend to be highly articulated objects and in order to detect a person, it is often essential to reason about the pose of the person in terms of configuration of parts: *i.e*. location of head, torso, limbs – all quantities not labelled in the dataset.

**Latent variable models** provide an ideal abstraction for such situations. They allow for modelling of interaction between the observed data (*e.g*. image features) and latent or hidden variables not observed in the training data (*e.g*. location of body parts). These hidden variables may provide a low-dimensional embedding of the input or help set up a mixture model where complex input-output dependencies are composed of simpler ones. For example, in the mixture of Deformable-Part Model (DPM) [14], latent variables are part locations & mixture component ids that allow modelling of multiple plausible articulations of the object. In handwritten digit recognition, deformations of digit images, such as rotation, can be modelled as latent variables to improve recognition accuracy [27, 18]. In document retrieval, the total ranking order of all documents related to a query can be modeled as a latent variable to help produce a higher number relevant documents in top $k$ returned results.

**Problem.** Unfortunately, training latent variable models is notoriously problematic since it typically involves a difficult non-convex optimization problem. Common algorithms for solving these problems, Expectation-Maximization (EM) [8] and the Concave-Convex Procedure (CCCP) [29, 27, 14], are known to be highly sensitive to initialization and prone to getting stuck in a poor local optimum. Standard techniques for mitigating the poor behaviour of these algorithms include multiple restarts with random initializations, smoothing the objective function, and annealing. Recently, Bengio *et al*. [4] and Kumar *et al*. [18] have presented curriculum learning schemes that train latent variable models in an easy-to-difficult manner, by initially pruning away difficult examples in the dataset.

**Goal.** At a high-level, our goal is to study the modelling-optimization tradeoff in designing latent variable models for computer vision problems. From a modelling perspective, we would like to design models with ever more complex latent variables, *e.g*. capture location of parts, their scale, orientation, appearance, *etc*. However, from an optimization perspective, complex models are more difficult to train than simpler ones; are more prone to getting stuck in a bad local minimum, ultimately resulting in poor generalization (often performing worse than simpler models). In most existing models, the complexity of the latent variable space (*e.g*. number of mixture components in a DPM [14]) is typically left as a free design choice that is hand-tuned. Thus, the question we seek to answer is: Is there a principled way to *learn* the complexity of the latent space?

**Overview.** In this paper, we address this question for a specific model – Structured SVMs with unstructured latent variables, *i.e*. linear models with an exponentially large (structured) output space, but an enumerable latent variable space. An example of such a model would be DPM [14] where the latent space indexes the mixture component ids, and thus the goal is to *learn* the number of mixtures in a DPM from data.

We propose the use of group-sparsity inducing norms

[*] Current affiliation: Yahoo! Inc.

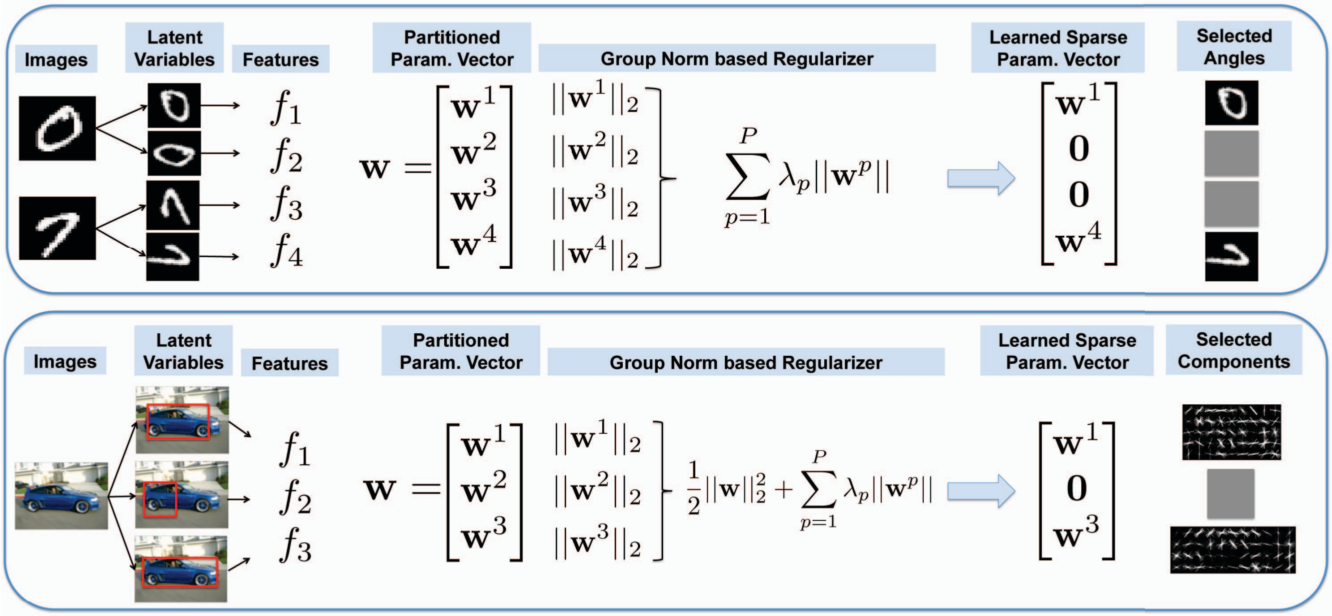
Images
Latent Variables
Features
Partitioned Param. Vector
Group Norm based Regularizer
Learned Sparse Param. Vector
Selected Angles
$f_1$
$f_2$
$f_3$
$f_4$
$\mathbf{w} = [\mathbf{w}^1; \mathbf{w}^2; \mathbf{w}^3; \mathbf{w}^4]$
$||\mathbf{w}^1||_2$ $||\mathbf{w}^2||_2$ $||\mathbf{w}^3||_2$ $||\mathbf{w}^4||_2$
$\sum_{p=1}^{P} \lambda_p ||\mathbf{w}^p||$
$[\mathbf{w}^1; 0; 0; \mathbf{w}^4]$
Images
Latent Variables
Features
Partitioned Param. Vector
Group Norm based Regularizer
Learned Sparse Param. Vector
Selected Components
$f_1$
$f_2$
$f_3$
$\mathbf{w} = [\mathbf{w}^1; \mathbf{w}^2; \mathbf{w}^3]$
$||\mathbf{w}^1||_2$ $||\mathbf{w}^2||_2$ $||\mathbf{w}^3||_2$
$\frac{1}{2}||\mathbf{w}||_2^2 + \sum_{p=1}^{P} \lambda_p ||\mathbf{w}^p||$
$[\mathbf{w}^1; 0; \mathbf{w}^3]$

Figure 1. Overview of our approach in the context of digit recognition (top) and object detection (bottom). In digit recognition, the latent variable indicates the rotation angle that must be corrected for before extracting features. For object detection, the latent variable is the component id in the mixture of deformable part models. The parameter vector is partitioned into groups corresponding to different latent states. Parameters for non-informative states become zero under such regularizers allowing us to select meaningful states for prediction.

like $\ell_1$-$\ell_2$ to estimate the parameters of such a model, thereby regularizing the complexity of the latent space. Group $\ell_1$-$\ell_2$ norm behaves like an $\ell_1$ norm at a group level and encourages groups of variables to be sparse. Specifically, we divide the latent variable state space into different groups, among which the group norm is induced. Since the group norm encourages group-sparsity, this allows simultaneous parameter estimation as well as state selection. Conceptually, this is an elegant solution since it gives the designer of a latent model tremendous flexibility in including plenty of latent variables without being concerned about the optimization issues – the group norm will automatically prune out latent variable states that are not helpful for prediction, while still utilizing all latent variables that have some informative states. Our approach is in a sense orthogonal to that of Bengio *et al.* [4] and Kumar *et al.* [18], in that they prune out difficult training examples to make the non-convex optimization easier, while we prune out difficult (or irrelevant) latent states.

We perform two sets of experiments: handwritten digit recognition on MNIST and object detection on the PASCAL VOC 2007 dataset [12]. Our first set of experiments show that our approach is indeed able to prune the complexity of latent space, resulting in a model that allows significantly *faster* inference at test time without drop in accuracy over a complete (non-sparse) model. Our second set of experiments show that our approach is able to learn a *better* model by adapting the complexity of the latent variable space to the category being trained.

Finally, our approach reuses almost all the existing Latent Structured SVM training machinery and is thus simple to incorporate in existing systems (*e.g.* DPM [14]).

## 2. Prior Work

**Latent Variable Models.** Latent variable models have been used to model observations in both generative and discriminative settings. In the generative setting, the goal is to explain the data with a low-dimensional latent structure. Mixture models like Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs) have a long history in applications such as speech recognition [24]. More recently, a number of discriminative latent models such as Hidden Conditional Random Field [23], Latent SVMs [14] and Latent Structured SVMs (LSSVMs) [27] have been proposed. These models have demonstrated success in a number of applications. They differ from generative models in the sense that the ultimate goal is prediction not explanation of the data. In both kinds of models, the parameter learning problem is non-convex and solved with techniques like Expectation-Maximization (EM) [8] and Concave-Convex Procedure (CCCP) [29, 27] respectively. Note that for all the models above, the latent variables and their state space are predefined and fixed. Our approach, on the other hand, aims for parameter estimation as well as discovery of meaningful latent variable states.

Related to this goal of discovery is the work of Chandrasekaran *et al.* [5], which attempts to identify the graphical model structure assuming that latent and observed variables are jointly Gaussian. Our work is different in that we are interested in prediction via a sparse latent model and not identification of such a model. Moreover, we do not make any Gaussian assumptions.

**Group Norms and Structured Sparsity.** There is a fairly mature body of work on $\ell_1$ regularization for sparse regression models [26, 6, 11]. Sparse coding with $\ell_1$ regularization has been successfully used to solve many problems in compressed sensing [10] and signal processing [20]. In the context of least-squares regression, group norm (*e.g.* $\ell_1 - \ell_2$) regularizers have been used [2, 28] to allow parameter estimation as well as selection of certain groups of variables. Bengio *et al.* [3] applied the group norm to build a word dictionary in bag-of-words document representations widely used in text, image, and video processing. Jia *et al.* [16] used the group norm to learn a latent space factorization into shared and private information. Recently, Jia *et al.* [15] used group norms for sparse representation based classification, for face recognition. A survey by Bach *et al.* [1] provides a comprehensive description of group norms and its applications. To the best of our knowledge, this is the first work to use structured norms in the context of latent variable selection in Latent (Structured) SVMs.

**Reducing Parameters in DPMs.** A number of recent works have looked at the broad problem of reducing the number of parameters in DPMs, by part-sharing [21], representing part filters as a linear combination of basis filters [22] or sparse-coding part filters [25]. We believe this is the first work to look at learning the number mixture components in a DPM.

The rest of this paper is organized as follows: Section 3 revisits Latent (Structured) SVMs; Section 4 describes our proposed group-norm modification; Section 5 describes how parameter learning can be performed in this model. Section 6 describes the two sets of experiments.

# 3. SVMs with Latent Variables

We begin by giving an overview of the Latent Structured SVM model and then specializing it to Latent Binary SVMs.

**Notation.** For any positive integer $n$, let $[n]$ be shorthand for the set $\{1, 2, \ldots, n\}$. We denote training data as $\mathcal{D} = \{(x_i, y_i) \mid i \in [n]\}$, where $x_i \in \mathcal{X}$ is the input feature vector, $y_i \in \mathcal{Y}$ is the (possibly structured) output label and $h_i \in \mathcal{H}$ is the latent variable for the $i^{th}$ datapoint. For example, in digit recognition, $x_i$ is the original image, $y_i \in \{0, 1, \ldots, 9\}$ is the true digit label and $h_i \in \{-60°, -45°, \ldots, 60°\}$ is the (deformation) rotation angle that must be corrected for before extracting features.

**Latent Structured SVMs (LSSVMs).** The linear prediction rule of LSSVMs is of the following form:

$$f(x) = \max_{(y,h) \in \mathcal{Y} \times \mathcal{H}} \quad \mathbf{w} \cdot \phi(x, y, h), \tag{1}$$

where $\phi(x, y, h)$ is the joint feature vector that encodes the relationship between the input, hidden and output variables, and $\mathbf{w}$ is the model parameter vector. In digit recognition, this joint feature vector is the vector representation of the image $x$ rotated by an angle corresponding to $h$. Let $\{\hat{y}_i(\mathbf{w}), \hat{h}_i(\mathbf{w})\} \triangleq \operatorname{argmax}_{(y,h) \in \mathcal{Y} \times \mathcal{H}} \mathbf{w} \cdot \phi(x_i, y, h)$ be the predicted output and latent variables for data-point $i$, written as a function of the parameter vector $\mathbf{w}$. A user-specified loss function $\Delta(y_i, \hat{y}_i(\mathbf{w}))$ measures the loss incurred for predicting $\hat{y}_i(\mathbf{w})$ for the $i^{th}$ sample, when the ground-truth label is $y_i$. Note that the loss function may additionally depend on the predicted latent variables, *i.e.* have the form $\Delta(y_i, \hat{y}_i(\mathbf{w}), \hat{h}_i(\mathbf{w}))$. The parameter vector $\mathbf{w}$ is learned by minimizing the (regularized) loss of the prediction on the training dataset $\mathcal{D}$. Unfortunately, this is a difficult optimization problem. Yu and Joachims [27] proposed minimizing an upper-bound on the loss and formulated the following optimization problem:

$$\min_{\mathbf{w}, \xi_i \geq 0} \quad \Omega(\mathbf{w}) + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{2a}$$

$$s.t. \quad \max_{h_i \in \mathcal{H}} \mathbf{w} \cdot \phi(x_i, y_i, h_i) - \mathbf{w} \cdot \phi(x_i, \bar{y}_i, \bar{h}_i) \geq \Delta(y_i, \bar{y}_i, \bar{h}_i) - \xi_i,$$

$$\forall (\bar{y}_i, \bar{h}_i) \in \mathcal{Y} \times \mathcal{H}, \;\; i \in [n]. \tag{2b}$$

where, the regularization term is $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. Intuitively, we can see that constraint (2b) tries to ensure that for each training instance $i$, the ground-truth $y_i$ and its best latent variable prediction $(\operatorname{argmax}_{h_i \in \mathcal{H}} \mathbf{w} \cdot \phi(x_i, y_i, h_i))$ have a higher score than all other labels and latent variable assignment pairs $(\bar{y}_i, \bar{h}_i)$ by a soft margin of $\Delta(y_i, \hat{y}_i, \hat{y}_i)$. Thus, high-loss configurations are forced to have a larger margin between them and the ground-truth. It can be shown that $\xi_i$ is an upper bound on the loss, *i.e.* $\xi_i \geq \Delta(y_i, \hat{y}_i(\mathbf{w}), \hat{h}_i(\mathbf{w}))$. More details can be found in [27].

**Latent SVMs (LSVMs).** The fairly general formulation of LSSVMs includes a number of interesting models as special cases. We describe one such instantiation, the deformable parts based Latent SVM model of Felzenszwalb *et al.* [14], which we use for one set of our experiments. In this model, $x_i$ are the HOG descriptors [7] computed at a particular sliding window location and scale in the image; $y_i \in \{+1, -1\}$ indicates presence or absence of a particular category in the window and $h_i$ indicates the mixture type of the deformable template and location and scale of root and part filters. The scoring function in this case can be reduced to the following form: $f(x) = \max_{h \in \mathcal{H}} \mathbf{w} \cdot \phi(x, h)$, where the joint feature vector $\phi(x, h)$ now does not depends on the label $y$. The loss function $\Delta$ is zero-one loss, and the learning problem looks more like a binary SVM:

$$\min_{\mathbf{w}, \xi_i \geq 0} \quad \Omega(\mathbf{w}) + \frac{C}{n} \sum_{i=1}^{n} \xi_i, \tag{3a}$$

$$s.t. \quad y_i f(x_i) \geq 1 - \xi_i, \qquad \forall i \in [n]. \tag{3b}$$

where $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. Constraints (3b) try to ensure that positive and negative training instances lie on different sides of the separation hyperplane with a soft margin of 1.

The next section describes our proposed group norm modification to the LSSVM and LSVM models, and Section 5 describes how parameter learning can be performed in the presence of this modification.

## 4. Inducing Group Norm for State Learning

We focus on models with an *enumerable* latent space, *i.e.* $\mathcal{H} = \{1, \cdots, P\}$ is the set of all possible latent configurations. Such a set of states can be the set of all possible rotation angles in digit recognition, or the set of mixture components in a DPM object detection model. Recall that our goal is to regularize the complexity of the latent space and learn which hidden states are really relevant for the prediction problem. To this end, we consider using the $\ell_1$-$\ell_2$ norm in the regularizer $\Omega(\mathbf{w})$ in problem (2) and (3) to learn meaningful latent states.

**Reformulating LSSVM Prediction Rule.** We start by describing a modification to the linear prediction rule in problem (2) that makes it easier to encode the group structure of latent states. Specifically, instead of learning a single weight vector $\mathbf{w}$, we now learn $P$ weight vectors $\{\mathbf{w}_h \mid h \in [P]\}$, one corresponding to each of the latent states. The modified linear prediction rule is given by $f(x) = \max_{y \in \mathcal{Y}, h \in [P]} \mathbf{w}_h \cdot \phi(x, y, h)$. We note that with appropriate zero-padding of the features, this model is equivalent to the original linear model. To see that, let $\mathbf{w} = [\mathbf{w}_1, \cdots, \mathbf{w}_P]$ be the concatenation of weight vectors from each group. We can also define new features:

$$\tilde{\phi}(x, y, h) = [\mathbf{0}_{n_1}, \mathbf{0}_{n_2}, \ldots, \phi(x, y, h), \ldots, \mathbf{0}_{n_P}], \qquad (4)$$

where $n_p$ is the length of $\mathbf{w}_p$. The above equation zeros pads the joint feature vectors such that only the weight vectors and features for the same group interact in the dot product: $\mathbf{w} \cdot \tilde{\phi}(x, y, h) = \sum_{p \neq h} \mathbf{0}_{n_p} \cdot \phi(x, y, p) + \mathbf{w}_h \cdot \phi(x, y, h)$.

The key reason for working with this representation is that parameters for each state are now represented separately and thus group $\ell_1$ regularization is possible over the state space. For any $q \in [1, \infty)$, an $\ell_1$-$\ell_q$ norm is given by $\Omega_{\mathcal{G}}(\mathbf{w}) = \sum_{p=1}^{P} \lambda_p \left\|\mathbf{w}_p\right\|_q$, where $\lambda_p \geq 0$ is the regularization weight for group $p$. Popular choices for $q$ are $\{2, \infty\}$ [1]. In our work, we only consider $q = 2$ and the regularizer is thus given as follows:

$$\Omega(\mathbf{w}) = \sum_{p=1}^{P} \lambda_p \left\|\mathbf{w}_p\right\|_2. \qquad (5)$$

We can see that within each group, the $\ell_2$ norm is used, which does not promote sparsity. At the group level, this norm behaves like the $\ell_1$ norm and thus induces group sparsity, *i.e.* the parameters of some groups are encouraged to be set completely to zero. Uninformative states will thus have sparse learned parameters. This gives us a natural way to select the latent states most useful for prediction.

In a manner similar to Elastic Nets [30], we can also use the group norm in combination with the $\ell_2$ norm:

$$\Omega(\mathbf{w}) = \frac{1}{2} \left\|\mathbf{w}\right\|_2^2 + \sum_{p=1}^{P} \lambda_p \|\mathbf{w}_p\|_2. \qquad (6)$$

Such a regularizer has the effect of both the original regularizer and the group norm. When $\lambda_p = 0$ for all $p$, the regularizer is reduced to the original form ($\ell_2$-norm). Group level sparsity can be induced when $\lambda_p$ is sufficiently large.

The next section describes the algorithm for parameter learning in LSSVMs with group norm regularizers.

## 5. LSSVM Training via Coordinate Descent

From an optimization perspective, both problems (2) and (3) can be viewed as minimizing a sum of convex and concave functions. Such problems are studied in the context of difference of convex programming and lend themselves to the concave-convex procedure (CCCP) [29, 27] and a similar coordinate descent approach of Felzenszwalb *et al.* [14]. We begin by rewriting problem (2) as minimization of difference of two convex functions:

$$\min_{\mathbf{w}} \; L(\mathbf{w}) \doteq \left[\Omega(\mathbf{w}) + \frac{C}{n} \sum_{i=1}^{n} \max\left\{0, f_i(\mathbf{w}) - g_i(\mathbf{w})\right\}\right], \qquad (7)$$

where $f_i(\mathbf{w}) = \max_{(\bar{y}_i, \bar{h}_i) \in \mathcal{Y} \times \mathcal{H}} [\mathbf{w}_{\bar{h}_i} \cdot \phi(x_i, \bar{y}_i, \bar{h}_i) + \Delta(y_i, \bar{y}_i, \bar{h}_i)]$ and $g_i(\mathbf{w}) = \max_{h_i \in H} \mathbf{w}_{h_i} \cdot \phi(x_i, y_i, h_i)$. We can see that $f_i(\mathbf{w})$ and $g_i(\mathbf{w})$ are both point-wise maximums of linear functions and thus convex. In order to minimize (7), we follow the approach of Felzenszwalb et al. [14] and minimize an upper bound on $L(\mathbf{w}) \leq L(\mathbf{w}, \{h_i\})$, which is the objective function with latent variables specified for the training data. Thus:

$$\min_{\mathbf{w}, \{h_i\}} \; L(\mathbf{w}, \{h_i\}) \doteq \left[\Omega(\mathbf{w}) + \frac{C}{n} \sum_{i=1}^{n} \max\left\{0, f_i(\mathbf{w}) - g_i(\mathbf{w}, h_i)\right\}\right], \qquad (8)$$

where $g_i(\mathbf{w}, h_i) = \mathbf{w}_{h_i} \cdot \phi(x_i, y_i, h_i)$. Intuitively, replacing $g_i(\mathbf{w})$ by $g_i(\mathbf{w}, h_i)$ implies that we enforce the margin not with respect to the best latent assignment for the ground-truth $y_i$, rather only the current latent assignment of $h_i$. Since $g_i(\mathbf{w}, h_i)$ is now simply a linear function in $\mathbf{w}$, $L(\mathbf{w}, \{h_i\})$ is the difference of a convex and a linear function, thus convex. In a manner similar to Felzenszwalb et al. [14], we follow a coordinate descent scheme. At iteration $t$, we first fix $\mathbf{w}^{(t)}$ and optimize $L(\mathbf{w}^{(t)}, \{h_i\})$ w.r.t. $\{h_i\}$. This is equivalent to computing:

$$h_i^{(t)} = \underset{h_i \in \mathcal{H}}{\operatorname{argmax}} \; \mathbf{w}_{h_i}^{(t)} \cdot \phi(x_i, y_i, h_i) \qquad \forall i \in [n]. \qquad (9)$$

This step is fairly straightforward and involves assigning the latent variables to their optimal states given the current setting of $\mathbf{w}^{(t)}$. Next, we fix $\{h_i^{(t)}\}$ and optimize w.r.t. $\mathbf{w}$. This is done via subgradient descent. The subgradient $\nabla L(\mathbf{w}, \{h_i^{(t)}\})$ is given by:

$$\nabla L(\mathbf{w}, \{h_i^{(t)}\}) = \nabla \Omega(\mathbf{w}) + \frac{C}{n} \sum_{i=1}^{n} m_i(\mathbf{w}, h_i^{(t)}), \qquad (10)$$

where $\nabla\Omega(\mathbf{w})$ is the subgradient of the regularizer:

$$\Bigg[\underbrace{\frac{\lambda_1 \mathbf{w}_1}{||\mathbf{w}_1||_2}}_{\text{Group 1}},\cdots,\underbrace{\frac{\lambda_P \mathbf{w}_P}{||\mathbf{w}_P||_2}}_{\text{Group } P}\Bigg], \tag{11}$$

and $m_i(\mathbf{w}, h_i^{(t)})$ is subgradient of the structured hinge loss:

$$\begin{cases} 0 & \text{if } f_i(\mathbf{w}) - g_i(\mathbf{w}, h_i^{(t)}) \leq 0 \\ \phi(x_i, \bar{y}_i^*, \bar{h}_i^*) - \phi(x_i, y_i, h_i^{(t)}) & \text{otherwise} \end{cases} \tag{12}$$

where $(\bar{y}_i^*, \bar{h}_i^*)$ is tuple that maximizes the loss-augmented score, *i.e.* $\mathrm{argmax}_{(\bar{y}_i,\bar{h}_i)\in\mathcal{Y}\times\mathcal{H}}[\mathbf{w}_{\bar{h}_i} \cdot \phi(x_i, \bar{y}_i, \bar{h}_i) + \Delta(y_i, \bar{y}_i, \bar{h}_i)]$. These two steps, *i.e.* fixing $\mathbf{w}^{(t)}$ and optimizing $L(\mathbf{w}^{(t)}, \{h_i\})$ w.r.t. $\{h_i\}$, and fixing $\{h_i^{(t)}\}$ to optimize w.r.t. $\mathbf{w}$, are repeated until the objective $L(\mathbf{w}, \{h_i\})$ converges. It can be shown that the algorithm always converges to a local minimum or a saddle point [29]. Algorithm 1 summarizes the entire algorithm.

The learning rate $\alpha_t$ is of important practical consideration. Following [17], we chose the learning rate at iteration $t$ to be $\alpha_t = \frac{1}{\eta_t+1}$, where $\eta_t$ is the number of times the objective value $L(\mathbf{w}, \{h_i^{(t)}\})$ has increased from one iteration of subgradient descent to the next. We found this learning rate to perform well in our experiments.

# 6. Experiment

We performed two sets of experiments: handwritten digit recognition on MNIST and object detection on the PASCAL VOC 2007 dataset [12]. Our results on the first set of experiments show that we are able to learn a sparse model that allows *faster* evaluation at test time, without drop in accuracy. The second set of experiments show that that we are able to learn a *better* model by adapting the complexity of the latent space to the category being trained.

## 6.1. Handwritten Digit Recognition

We closely follow the experimental setup of Kumar *et al.* [18], who proposed an LSSVM approach for this problem. Each digit is represented as a vector $x$ of pixel grayscale values. The goal is to predict the digit label $y \in \mathcal{Y} = \{0, 1, \cdots, 9\}$. Kumar *et al.* [18] showed that accuracy can be greatly improved by explicitly modeling (and correcting for) the rotational deformations present in each image. Specifically, they consider rotation as a hidden variable taking values in a set of 11 angles uniformly distributed from $-60°$ to $60°$, *i.e.* $h_i \in \mathcal{H} = \{-60°, -48°, \ldots, 60°\}$. The joint feature vector is $\phi(x, y, h) = \text{pca}(x_h)$, where $x_h$ is the image rotated by the angle corresponding to $h$ and $\text{pca}(x_h)$ is the 10-dimensional PCA projection of this rotated image. Examples of these rotated images are shown in Figure 1. We show that by inducing a group-norm over the parameters corresponding to each hidden state, only a few rotations are needed to achieve the recognition accuracy with the full set of angles.

**Algorithm 1** Group-Norm LSSVM Training

**Input:** $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, convergence criteria $\epsilon$, initialization $\mathbf{w}^{(0)}$.

```
$t \leftarrow 0$
repeat
    for $i = 1$ to $n$ do
        {#Find best latent assignment $h_i$.}
        $g_i \leftarrow \max_{h_i \in \mathcal{H}} \mathbf{w}_{h_i}^{(t)} \cdot \phi(x_i, y_i, h_i)$
        $h_i^{(t)} \leftarrow \mathrm{argmax}_{h_i \in \mathcal{H}} \mathbf{w}_{h_i}^{(t)} \cdot \phi(x_i, y_i, h_i)$
    end for
    repeat
        {#Optimize Over $\mathbf{w}$ with Subgradient Descent}
        for $i = 1$ to $n$ do {#Can pick a random element here if Stochastic Subgradient}
            $f_i \leftarrow \max_{(\bar{y}_i,\bar{h}_i)\in\mathcal{Y}\times\mathcal{H}} \Big[\mathbf{w}^{(t)} \cdot \phi(x_i, \bar{y}_i, \bar{h}_i) + \Delta(y_i, \bar{y}_i, \bar{h}_i)\Big]$, and obtain maximizer $(\bar{y}_i^*, \bar{h}_i^*)$
            $m_i \leftarrow 0$
            if $f_i - g_i > 0$ then
                $m_i \leftarrow \phi(x_i, \bar{y}_i^*, \bar{h}_i^*) - \phi(x_i, y_i, h_i^{(t)})$
            end if
        end for
        $\nabla L \leftarrow \nabla\Omega(\mathbf{w}^{(t)}) + \frac{C}{n}\sum_{i=1}^{n} m_i$
        $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t)} - \alpha_t \nabla L$
    until Convergence of Subgradient Descent
    $t \leftarrow t + 1$
until $\left|\frac{L(\mathbf{w}^{(t+1)}, \{h_i^{(t+1)}\}) - L(\mathbf{w}^{(t)}, \{h_i^{(t)}\})}{L(\mathbf{w}^{(t)}, \{h_i^{(t)}\})}\right| \leq \epsilon$
```

We work with the MNIST dataset [19], and perform binary classification on four difficult digit pairs (1-vs-7, 2-vs-7, 3-vs-8, 8-vs-9). The training data for each digit contains about 6000 images and the testing data contains approximately 1000 images. We use $\lambda_p = 1$ for each group in our experiment. We tried different values of C, and the prediction accuracies were fairly similar. We set $C = 1$.

Figure 2 shows the $\ell_2$-norms of the parameter vectors for different angles in the 4 digit-pair experiments. We can see that the $\ell_2$-norms for many angles are completely zero, and only a subset of angles actually remain to contribute to the final prediction. Our sparse model essentially selects 5, 8, 7, and 9 angles in total for digit pairs 1-7, 2-7, 3-8, and 8-9 respectively. This is a significant reduction from the hidden space of 22 angles per digit pair in the original model.

Table 1 shows prediction accuracy vs the number of angles used. For this table, we selected angles in descending order of the $\ell_2$-norms. As a baseline, we compare to uniform angle selection based on the original approach by Kumar *et al.* [18]. We can see that our approach achieves good performance very quickly. In fact, using only 8 angles for each digit pair, we can achieve prediction accuracies similar to those using the *entire* set of angles.

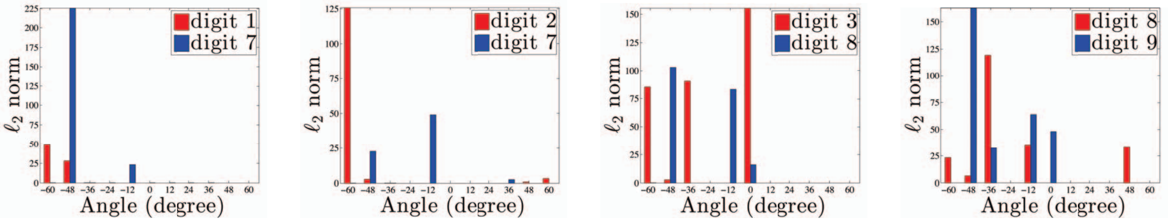

Figure 2. $\ell_2$ norm of the parameter vectors for different angles over the 4 digit pairs.

| Number of angles | | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Digit pair<br>1 vs 7 | Our approach | 0.863 | **0.988** | **0.988** | **0.988** | **0.988** | **0.988** | **0.988** | **0.988** | **0.988** | **0.988** | **0.988** |
| | Kumar *et al.* [18] | **0.961** | 0.962 | 0.959 | 0.959 | 0.962 | 0.964 | 0.963 | 0.962 | 0.961 | 0.964 | 0.945 |
| Digit pair<br>2 vs 7 | Our approach | 0.852 | **0.957** | **0.953** | **0.956** | **0.956** | **0.956** | **0.956** | **0.956** | **0.956** | **0.956** | **0.956** |
| | Kumar *et al.* [18] | **0.953** | 0.942 | 0.938 | 0.943 | 0.942 | 0.947 | 0.943 | 0.948 | 0.943 | 0.945 | 0.941 |
| Digit pair<br>3 vs 8 | Our approach | 0.499 | 0.719 | **0.923** | **0.923** | **0.923** | **0.923** | **0.923** | **0.923** | **0.923** | **0.923** | **0.923** |
| | Kumar *et al.* [18] | **0.882** | **0.876** | 0.901 | 0.890 | 0.905 | 0.901 | 0.901 | 0.903 | 0.899 | 0.904 | 0.916 |
| Digit pair<br>8 vs 9 | Our approach | 0.797 | 0.617 | 0.875 | **0.954** | **0.954** | **0.954** | **0.954** | **0.954** | **0.954** | **0.954** | **0.954** |
| | Kumar *et al.* [18] | **0.937** | **0.933** | **0.936** | 0.934 | 0.936 | 0.942 | 0.941 | 0.943 | 0.940 | 0.942 | 0.933 |

Table 1. Accuracy vs #angles for our approach and uniform selection based on the original approach by Kumar *et al.* [18]. We can see that our approach outperforms uniform selection and is able to quickly achieve accuracy comparable to the complete model (using all angles).

| Digit Pair | 1 vs 7 | 2 vs 7 | 3 vs 8 | 8 vs 9 |
|---|---|---|---|---|
| Our approach 8 angles | 6.5 | 7.0 | 5.4 | 6.0 |
| Kumar *et al.* [18] 22 angles | 22.3 | 24.1 | 23.7 | 23.1 |

Table 2. Running time (seconds) comparison of our approach and Kumar *et al.* [18]. Time reported is cputime on 64-bit 8-Core Intel i7 machine with 12GB RAM.

Running time for feature computation increases linearly with the number of angles chosen because the time to rotate an image and to perform PCA for each angle is about the same. Thus, as shown in Table 2, evaluation at test-time with our sparse method using 8 angles per digit-pair is 2.5-3 times faster than the non-sparse model without any or significant loss in accuracy.

## 6.2. Object Detection with Deformable Part Models

Next, we applied our approach to train a mixture of Deformable Part Models [14] for object detection. Each component in the mixture is a star-structured part model consisting of a root filter, part filters, and part displacements vectors. The score of a component at a particular location and scale in the image is defined as the sum of scores for root and part filters minus the deformation cost of placing part filters in the image. Each component is bilaterally symmetric and thus an $n$-component mixture really has $2n$-members. To detect the object instances, this model is scanned across different locations and scales in the image, followed by standard post-processing steps such as bounding box prediction, non-maximal suppression, and context rescoring. Details can be found in [14].

The goal of our experiments is to select and learn a *sparse* mixture model with only a subset of components. The motivation for doing this is our observation that as the number of components increases (from n=1 to n=6), the model generally overfits to the training data. For instance, Figure 3 shows the train and test accuracies vs number of components for the 'cat' category. Trends on other categories are provided in the supplemental materials. We can see that `test` accuracy increase at first then decreases while `trainval` accuracy increases monotonically, which is the classical sign of overfitting. Thus, We would like to learn the appropriate number components for each category. The naïve way of doing this would be cross-validation on $n$ for each category. However, this is computationally prohibitive and requires tuning 20 parameters. We show that using our approach we can learn the appropriate sparsity level for all classes via *a single parameter* $\lambda_p (= \lambda, \forall p)$.

We worked with release4 of the DPM system [13], and report results on the PASCAL VOC 2007 dataset [12]. VOC 2007 is smaller than the latest VOC 2012 dataset. However, ground-truth annotations for 2007 are available while evaluating results on 2012 requires submission to the evaluation server. Our experiments involve evaluating the effect of a large number of parameters and VOC discourages multiple submissions. Thus, following the 'best practices' guidelines, we report results on VOC 2007. We believe our approach should only work better with more data available.

From an implementation perspective [13], the parameter learning proceeds in three stages. Stage 1 learns the parameters of root filter for each component separately. Stage 2 concatenates the parameters of all components and learns a mixture model of root filters. Stage 3 adds the part and displacement parameters for each component and learns the final mixture model. Stages 2 and 3 use the coordinate descent approach described in Section 5. We applied the group norm to stage 2 of the learning process so that we can select a subset of components to do learning in stage 3. We initialized the mixture model with 6 components.

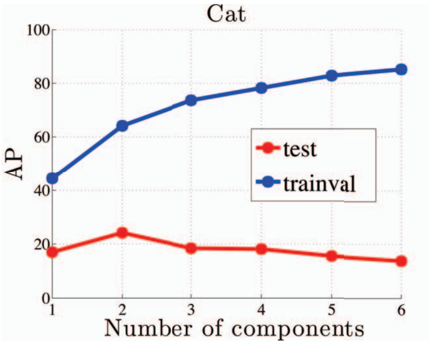

Figure 3. Accuracy of the Felzenszwalb *et al.* detector [14] vs the number of components for the 'cat' category, trained on VOC 2007 `trainval` and tested on `test`. Only root+part accuracies are shown (no bounding box prediction or context rescoring is performed). We can see that `test` accuracies increase at first then decrease while `trainval` accuracies increase monotonically, which is a classical sign of overfitting.

| $\lambda_p$ | 0.08 | 0.10 | 0.12 | 0.14 | 0.16 |
|---|---|---|---|---|---|
| Mean AP | 25.1 | 25.2 | **25.8** | 23.9 | 23.3 |
| Avg. #components | 4.8 | 3.9 | **3.1** | 2.4 | 2.0 |

Table 3. Results on VOC2007 `val` using models trained on `train`. The results are based on root+part model only (no bounding box prediction or context rescoring is performed). We can see that $\lambda_p = 0.12$ achieves the highest mean average precision and with 3.1 non-sparse components on average (out of 6).

**Effect of $\lambda_p$.** Table 3 shows the mean average precision and the average number of non-sparse components as a function of $\lambda_p$ for 2007 `val` set, using a model trained on `train`. We can see that $\lambda_p = 0.12$ gives the highest mean average precision and produces a reasonable level of sparsity on average (across categories). Results for individual categories are in the supplement. We use this optimal setting of $\lambda_p$ for our experiments on `test`.

**Qualitative Example.** Figure 4 shows a qualitative example of the model before and after Stage 2 training with the group norm. Some components that are very similar to others are removed by the group norm.

**Quantitative Results.** Table 4 summarizes the results on VOC 2007 `test`, with models trained on `trainval`. We can see that as $n$ increases, (standard) DPM accuracies for some categories like bottle monotonically decrease, while those for cat and horse reach their peak somewhere in the middle. However, the mean accuracies initially increase from $n = 1$ to $n = 2$, but then stagnate. This behavior empirically verifies our hypothesis that a single setting of number of components is a suboptimal choice. Our approach is able to pick out a fairly non-uniform sparsity pattern across the categories, performing the best in 7 out of 20 categories with a mean average precision of 32.5. This is better than *all other settings* in which $n$ is fixed. On average, 3.6 out of 6 components remains in the final models. This is significantly lower than the original 6 components for each category, resulting in 40% faster detection at test time. The training time of our approach for each object category is similar to that by the original approach with the same number of components because Stage 3 dominates the training process. We also ran the bounding box prediction and context rescoring steps the sake of completeness, and observe similar trends.

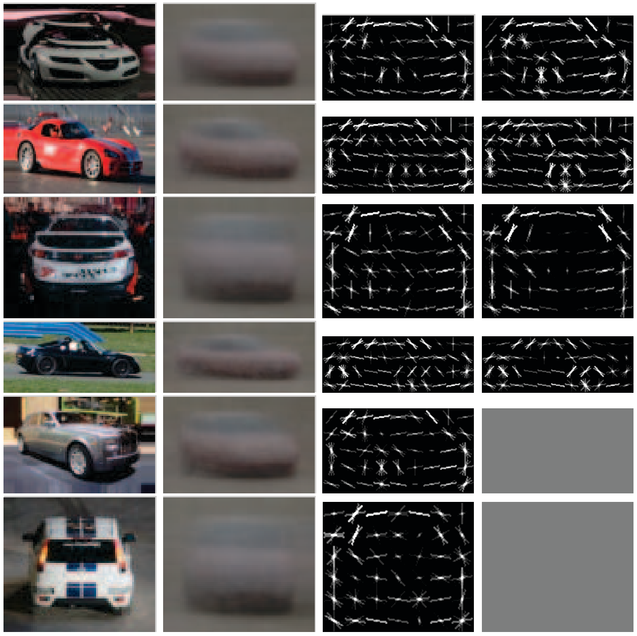

Figure 4. The mixture of root model before and after the training by $\ell_1$-$\ell_2$ norm for the car category. Each row corresponds to a component. First column shows a positive example from the component. Second column shows the average image for the component. Third and fourth columns show the root filter of the component before and after training respectively. A complete gray image indicates the filter is sparse. We can see that the components 4,5 were very similar components 2,3 and thus were removed.

## 7. Summary

We address the problem of estimating the parameters of latent variable models as well as discovering meaningful states for the latent variables. This allows us to control the model complexity and speed up inference time. We address this problem in the context of SVMs with structured output variables and unstructured (enumerable) latent variables via an $\ell_1$-$\ell_2$ group norm regularization. Our experiments on handwritten digit recognition show that our approach is able to effectively reduce the size of latent variable state space and thus reduce the inference time with no loss of accuracy compared to using the full latent state space. Our experiment on object detection shows that we are able to adapt the number of components to the category being learnt and achieve higher detection performance.

In this work, we build on the standard DPM framework [14], which typically uses $n = 3$ mixture components. Recent work on *visual subcategories* [9] has argued that a larger number of mixture components can lead to improved

| **root+part** | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbik | pers | plant | sheep | sofa | train | tv | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n=1 | 24.3 | 49.5 | 8.2 | 6.5 | 27.8 | 45.2 | 51.5 | 17.0 | 19.2 | 22.9 | 22.2 | 5.1 | 49.8 | 37.9 | 33.6 | 6.5 | 13.6 | 30.4 | 34.2 | 42.7 | 27.4 |
| n=2 | 31.7 | 56.0 | 10.3 | 11.2 | 27.5 | 52.0 | 53.8 | **24.2** | 21.1 | 26.6 | 22.9 | 10.6 | **59.6** | 44.1 | 39.5 | 13.6 | 18.1 | 29.1 | 44.0 | 42.0 | 31.9 |
| n=3 | 29.6 | 57.3 | 10.1 | 17.1 | 25.2 | 47.8 | **55.0** | 18.4 | 21.6 | 24.7 | 23.3 | 11.2 | 57.6 | 46.5 | **42.1** | 12.2 | 18.6 | **31.9** | 44.5 | 40.9 | 31.8 |
| n=4 | 31.8 | 57.2 | 10.1 | 14.7 | 24.3 | 50.0 | 54.1 | 18.2 | 20.4 | 24.8 | 19.3 | 11.0 | 57.0 | 40.2 | 38.1 | 12.8 | 22.8 | 28.4 | 46.6 | 40.0 | 31.1 |
| n=5 | **32.2** | **57.6** | 10.4 | **17.2** | 23.2 | **54.5** | 54.0 | 15.6 | 19.6 | 24.2 | 25.1 | 11.3 | 56.2 | **47.8** | 39.3 | 12.0 | 18.5 | 30.9 | **48.7** | 39.8 | 31.9 |
| n=6 | 30.5 | 56.7 | **11.0** | 16.2 | 22.1 | 49.7 | 54.1 | 13.9 | 19.6 | 21.7 | 21.4 | 11.2 | 55.7 | 46.8 | 38.5 | 8.3 | **23.6** | 26.0 | 43.9 | 40.8 | 30.6 |
| ours ($\lambda_p = 0.12$) | 30.8 | 56.3 | 9.4 | 15.5 | **28.4** | 52.4 | 54.5 | 19.8 | **21.9** | **30.1** | **28.0** | **11.3** | 57.1 | 45.7 | 38.6 | **14.7** | 15.4 | 31.8 | 44.5 | **43.2** | **32.5** |
| comp. num. | 4 | 6 | 1 | 2 | 2 | 3 | 4 | 5 | 2 | 4 | 3 | 5 | 4 | 4 | 6 | 4 | 1 | 2 | 6 | 3 | 3.6 |
| **bbox** | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbik | pers | plant | sheep | sofa | train | tv | mean |
| n=1 | 25.0 | 50.2 | 8.3 | 6.5 | 28.3 | 45.6 | 54.5 | 17.2 | 19.6 | 22.8 | 23.1 | 5.2 | 50.4 | 37.9 | 33.8 | 6.8 | 14.9 | 31.1 | 35.0 | **44.0** | 28.0 |
| n=2 | 31.0 | 58.5 | 10.2 | 10.7 | 27.4 | 52.6 | 56.2 | **26.1** | 21.5 | 26.6 | 22.3 | 10.8 | **61.6** | 45.3 | 40.0 | 13.5 | 17.9 | 29.9 | 45.4 | 42.6 | 32.5 |
| n=3 | 28.9 | 59.5 | 10.0 | 15.2 | 25.5 | 49.6 | **57.9** | 19.3 | **22.4** | 25.2 | 23.3 | 11.1 | 56.8 | 46.6 | **41.9** | 12.2 | 17.8 | **33.6** | 45.1 | 41.6 | 32.2 |
| n=4 | 32.1 | **59.6** | 10.2 | 15.3 | 24.6 | 51.9 | 57.5 | 18.5 | 20.2 | 25.1 | 17.3 | 11.0 | 57.4 | 43.0 | 36.6 | 12.5 | 22.5 | 28.0 | 46.6 | 41.7 | 31.6 |
| n=5 | 31.0 | 58.5 | 10.4 | **17.7** | 23.5 | **54.9** | 57.6 | 17.3 | 19.5 | 22.6 | 24.7 | 11.1 | 57.6 | **49.2** | 39.8 | 11.6 | 18.4 | 32.3 | **47.1** | 40.8 | 32.3 |
| n=6 | 29.6 | 56.1 | **10.9** | 15.3 | 21.8 | 50.5 | 57.1 | 15.3 | 20.2 | 19.8 | 21.0 | 11.4 | 55.7 | 45.5 | 38.5 | 10.3 | **23.6** | 25.4 | 42.6 | 41.6 | 30.6 |
| ours ($\lambda_p = 0.12$) | **33.6** | 57.6 | 9.4 | 15.5 | **28.9** | 51.7 | 55.3 | 20.2 | 22.1 | **30.4** | **28.9** | **11.5** | 58.1 | 46.4 | 38.8 | **14.1** | 16.2 | 32.3 | 45.6 | 43.8 | **33.0** |
| **context** | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbik | pers | plant | sheep | sofa | train | tv | mean |
| n=1 | 27.9 | 52.0 | 11.2 | 9.8 | **29.7** | 47.7 | 56.3 | 23.6 | 20.8 | 25.0 | 26.4 | 13.3 | 53.4 | 42.2 | 35.5 | 9.0 | 15.7 | 34.4 | 38.3 | **45.8** | 30.9 |
| n=2 | 33.2 | 60.5 | 12.5 | 10.8 | 28.6 | 52.7 | 58.0 | **30.9** | 22.9 | 28.7 | 26.3 | 13.2 | **65.3** | 47.6 | 42.6 | 15.6 | 19.9 | 33.1 | 49.3 | 44.2 | 34.8 |
| n=3 | 31.1 | **61.6** | 11.9 | 17.3 | 27.1 | 49.0 | **59.6** | 22.9 | 23.0 | 26.7 | 24.5 | 12.9 | 60.2 | 49.5 | **43.2** | 13.5 | 18.9 | **36.4** | 49.2 | 43.0 | 34.1 |
| n=4 | 35.2 | 59.4 | 12.4 | 16.7 | 25.4 | 51.9 | 59.1 | 20.2 | 21.3 | 26.3 | 18.8 | 12.7 | 60.3 | 46.2 | 38.3 | 14.5 | 21.4 | 30.6 | **51.1** | 43.1 | 33.2 |
| n=5 | 32.6 | 58.4 | **12.7** | 17.9 | 25.1 | **55.8** | 58.9 | 17.5 | 19.9 | 24.7 | 21.4 | 13.2 | 61.0 | **51.6** | 42.4 | 13.5 | 19.4 | 32.8 | 49.8 | 41.5 | 33.5 |
| n=6 | 30.8 | 57.1 | 12.1 | 15.7 | 21.6 | 49.1 | 57.7 | 19.1 | 20.6 | 22.3 | 18.8 | 13.3 | 58.3 | 47.1 | 41.4 | 12.0 | **22.9** | 28.0 | 43.0 | 42.2 | 31.7 |
| ours ($\lambda_p = 0.12$) | **35.8** | 59.9 | 10.3 | **18.1** | 29.3 | 53.4 | 56.3 | 25.3 | **23.5** | **30.6** | **31.0** | **13.9** | 60.5 | 48.9 | 41.1 | **16.1** | 17.3 | 35.3 | 49.5 | 45.7 | **35.1** |

Table 4. Accuracies on VOC 2007 `test` with models trained on `trainval`. Evaluation is performed based on 1) root and part (root+part) filters, 2) bounding box (bbox) prediction, and 3) context rescoring. The last row in the root+part table shows the number of non-sparse components out of a total of 6 components being trained.

performance. The task of selecting an appropriate number of subcategories for each category becomes even more crucial in this context. We are investigating these directions.

**Acknowledgements.** We gratefully acknowledge funding support from Quanta Computer and Google Grants. Part of this work was done while DC and DB were at TTI-Chicago. We thank Micah Kimo Johnson for helpful discussions.

## References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 2012.

[2] S. Bakin. *Adaptive Regression and Model Selection in Data Mining Problems*. Australian National University, 1999.

[3] S. Bengio, F. Pereira, Y. Singer, and D. Strelow. Group sparse coding. In *NIPS*, 2009.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[5] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Latent variable graphical model selection via convex optimization. *Annals of Statistics*, 2012.

[6] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 2001.

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

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.

[9] S. K. Divvala, A. A. Efros, and M. Hebert. How important are deformable parts in the deformable parts model? *CoRR*, abs/1206.3714, 2012.

[10] D. L. Donoho. Compressed sensing. *Info. Theory*, 2006.

[11] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 2004.

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

[13] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://www.cs.brown.edu/∼pff/latent-release4/.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[15] K. Jia, T.-H. Chan, and Y. Ma. Robust and practical face recognition via structured sparsity. In *ECCV*, 2012.

[16] Y. Jia, M. Salzmann, and T. Darrell. Factorized latent spaces with structured sparsity. In *NIPS*, 2010.

[17] T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. Dual decomposition for parsing with non-projective head automata. In *EMNLP*, 2010.

[18] P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

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

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

[22] H. Pirsiavash and D. Ramanan. Steerable part models. In *CVPR*, 2012.

[23] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS*, 2004.

[24] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

[25] H. O. 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.

[26] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 1996.

[27] C. J. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009.

[28] M. Yuan, M. Yuan, Y. Lin, and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 2006.

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

[30] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal Of The Royal Statistical Society Series B*, 2005.