HUMAN ACTION RECOGNITION USING DIVERSE DATA AND SELF-PACED LEARNING

STANFORD UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Beyang Liu
September 2011

Vision without action is a daydream. Action without vision is a nightmare.

– Japanese proverb

The important thing is the diversity available on the Web.

– Tim Berners-Lee

When I see an adult on a bicycle, I do not despair for the future of the human race.

– H.G. Wells

Abstract

Human action recognition in still images is a challenging task in computer vision. Existing action recognition systems require a large amount of highly annotated training data. Acquiring such data is expensive and time consuming. Ideally, we would like to take advantage of cheaper and more plentiful sources of weakly annotated images. We present an action recognition model within the framework of latent stuctural support vector machines (LSVM) that can be trained on *diverse data*, i.e., data with heterogeneous levels of annotation. We train our action model on a combination of highly annotated images with person and object ground-truth and weakly labeled images obtained from Google Images where only the action is known.

We experiment with two training algorithms for LSVMs – the convex-concave procedure (CCCP) and self-paced learning (SPL) – and show that SPL takes better advantage of additional weakly labeled data, outerperforming CCCP. Finally, we show that our model achieves state-of-the-art performance on the task of action recognition for two action classes on a challenging dataset.

Acknowledgements

I thank my advisor, Professor Daphne Koller, for her guidance and support throughout my undergraduate years. She has been a continual source of inspiration and her mentorship has been critical to my development as a fledgling researcher in artificial intelligence and computer vision. I thank M. Pawan Kumar and Benjamin Packer for their patience, support, and guidance over the entire course of this project. They contributed countless hours of their time, providing crucial advice and direction without which this work could not have been brought to fruition. It was both a privilege and a great pleasure to work with them. Additional thanks goes to Professor Stephen Gould, who, as a PhD student, inspired and mentored me during my first undergraduate research project. Finally, I thank Andrew Duchi for being a great source of feedback and lively discussion and Mary McDevitt for proofreading and copy editing an early draft of this thesis.

Contents

List of Figures

Chapter 1

Introduction

Action recognition is a crucial area of research in the development of intelligent computer vision. A truly intelligent vision system must not only be capable of discerning what entities (e.g., people, objects) are present in a scene, but also what actions are being performed by and on those entities. To do so, it can take into account contextual information, such as the location of the object being acted upon and the location of the person performing the action. It must be capable of capturing the relationships between these interrelated variables and reasoning about them robustly in natural environments with high clutter and occlusion.

A model that adequately captures such a complex system will almost certainly be sophisticated and nuanced. Therefore, it is desirable to fit the model to as large a training set as possible. Obtaining a large amount of completely labeled data, however, is time-consuming and expensive. To this end, we would like to take advantage of *diverse data*, i.e., datasets that include not only fully labeled data but also additional data of varying levels of annotation. To make use of this additional, partially labeled data, we need to model the portions of the label that are potentially hidden at training time. A good framework in which to do so is that of the latent support vector machine (LSVM).

Using the LSVM framework, we construct a human action recognition model that can detect the presence of 3 different action classes – riding bike, riding horse, and using computer. Each action is associated with a specific object – bike, horse, and

computer monitor – and our model incorporates the presence of such objects and their spatial relationship to the person conducting the action. We then experiment with two state-of-the-art training algorithms for LSVMs, the Concave-Convex Procedure (CCCP) and Self-Paced Learning (SPL), and compare results on the PASCAL Action Classification dataset. Our results demonstrate the benefit of using a context-aware model that takes advantage of diverse data, as well as the superiority of the SPL algorithm.

Chapter 2

Related Work

Previous work on human action recognition has utilized a variety of data types, including single image, video, and 3D motion-capture. In the domain of video and motion-capture, many methods have applied motif discovery algorithms to time series data [13, 3, 1, 16]. Key issues here include identifying motifs in noisy data and discovering, in an unsupervised fashion, characteristic motifs that are robust to variations in length, frequency, and spatial deformation. These methods vary in model complexity. Wang, *et al.* 2009 [14] proposed a model that incorporates action as a latent topic variable and uses a simple bag-of-words representation of features for each image frame. Niebles, *et al.* 2008 [9] defined a more sophisticated hierarchical model that incorporates the spatial relationship between human parts (pose detection) to classify objectless actions, such as waving and walking. These methods, of course, require data to be gathered over time and each frame of the data must often be painstakingly labeled with the location of objects and human parts. The human brain, however, can discern actions without temporal data, given only single images, and it is this more constricted setting that we choose to explore here.

In the domain of still-image action recognition, previous work has produced contextual models that attempt to model the relationship between various components present in a scene. Rabinovich, *et al.* 2007 [10] defined a Conditional Markov Rankom

field that models relationships between general scene entities, including but not limited to people and objects, in order to improve object detection and semantic image segmentation. Other efforts specifically targeted at action recognition have constructed more sophisticated hierarchical models that explicitly incorporate object and pose location. Karlinsky, *et al.* 2010 [7] defined a two-step approach that first detects body parts and then extracts features from the region around the detected parts. The spatial relationship of the features to their corresponding parts is taken into account in the action inference.

Other works define unified models that reason jointly about action, human pose, and object location. Gupta, *et al.* 2009 [5] proposed a Bayesian model over the scene type, human pose, and object location. In particular, the spatial relationship between human and object was modeled using shape context features that capture the angle and distance between the two. Yao, *et al.* 2010 [15] presented a pictorial structure model that simultaneously models human pose, object location, and action detection on a dataset of six sports. The benefit obtained by using such contextual methods is reciprocal; not only does pose and object context strengthen the accuracy of the action detector, the explicit modeling of action leads to better object and pose detection, as well.

A severe limitation of such approaches, however, is that the incorporation of a wide range of contextual information requires finely annotated training data (e.g., bounding boxes for each object and for each human body part). Because such models include many inter-correlated variables, a large amount of such data is needed. In the present work, we present an action recognition model that can take advantage of large amounts of weakly labeled data in addition to highly annotated data. Moreover, because human pose is not a strong cue for the set of actions we consider (the pose associated with riding a horse, for example, is very similar to that associated with riding a bike), we propose a simpler human-object interaction model that is not reliant on fine-grain human parts labels. The strength and robustness of such an approach will be demonstrated by evaluating on a dataset that contains high levels of clutter and occlusion in a natural setting where objects and humans vary greatly in size and perspective.

Chapter 3

Background

3.1 Support Vector Machines

Given a data example with observed variables $\boldsymbol{x} \in \mathbf{R}^d$, we wish to classify the example by assigning a label, $y \in \{-1, 1\}$. (By convention, $y = 1$ denotes a "positive" example and $y = -1$, a "negative".) A linear classifier is a predictor of the form:

$$y^{\text{pred}} = \text{sign}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})) \tag{3.1}$$

Here, $\boldsymbol{w} \in \mathbf{R}^n$ parameterizes the classifier and $\boldsymbol{\phi} : \mathbf{R}^d \to \mathbf{R}^n$ is a feature function, which maps the observed variables to a feature vector in $\mathbf{R}^n$.

Training is the process of fitting the parameter vector $\boldsymbol{w}$ to a set of training examples, typically by optimizing an objective function defined over the training set. Let $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$ be a set of m training examples, each of which is comprised of an observed data vector $\boldsymbol{x}_i$ and a ground-truth label y_i. A support vector machine (SVM) is a linear classifier whose training procedure is

$$\underset{\boldsymbol{w}}{\text{minimize}} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + \frac{C}{m} \sum_{i=1}^m \max(0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)) \tag{3.2}$$

The first term is a quadratic regularization term on $\boldsymbol{w}$ that prevents overfitting and the second term is a convex upper-bound on the training set accuracy, $\sum_{i=1}^m \mathbf{1}\{y_i =$

$\text{sign}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i))\}$. C is a meta-parameter that trades off the importance of the regularization term against high training set accuracy.

The support vector machine is a powerful formalism used in many binary classification applications in computer vision and machine learning, including object detection, image segmentation, and spam filtering.

3.1.1 Geometric Margin

The support vector machine has an elegant geometric interpretation related to finding the separating hyperplane with minimal geometric margin between the sets of positive and negative training examples in the feature space. The separating hyperplane is $\{\boldsymbol{x} : \boldsymbol{w}^\top \boldsymbol{x} = 0\}$, and the geometric margin is the maximum distance that this hyperplane can be translated in the direction $\boldsymbol{w}$ or $-\boldsymbol{w}$ before it touches a positive or negative example. The minimum geometric margin problem can be formulated as

$$\text{minimize} \ \ \frac{1}{2} \|\boldsymbol{w}\|_2^2 \tag{3.3}$$

$$\text{subject to} \ \ y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) \geq 1, \ \text{for } i = 1, ..., m \tag{3.4}$$

Note that this imposes the hard constraint that $\boldsymbol{w}$ define a hyperplane that separates all positive examples from all negatives. This constraint can be relaxed to allow violations in exchange for incurring a penalty in the objective. This relaxation yields the following problem:

$$\text{minimize} \ \ \frac{1}{2} \|\boldsymbol{w}\|_2^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \tag{3.5}$$

$$\text{subject to} \ \ y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) \geq 1 - \xi_i \qquad \text{for } i = 1, ..., m \tag{3.6}$$

$$\xi_i \geq 0 \qquad \text{for } i = 1, ..., m \tag{3.7}$$

Here, the ξ_i's (also known as *slack variables*) define the penalty incurred by each example for violating the margin constraint $y_i \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_i) \geq 1$. This is known as the L1 soft-margin formulation of the SVM and is equivalent to the unconstrained SVM problem given in Eq. (3.2). The geometric intuition provided by this perspective of

SVMs will become important in the discussion of structural and latent SVMs in the sequel.

3.2 Structural SVM

The structural support vector machine (SSVM) is a generalization of the standard binary support vector machine to multidimensional output spaces, where the label, $\boldsymbol{y}$, is a vector whose elements can be discrete- or even continuous-valued. Within the SSVM framework, we have a linear predictor of the form:

$$\boldsymbol{y}^{\text{pred}} = \underset{\boldsymbol{y}}{\text{argmax}}\, \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) \tag{3.8}$$

As in the binary SVM case, $\boldsymbol{x}$ is the observed data vector. Note that the feature function, $\boldsymbol{\psi}$, is now a function of both $\boldsymbol{x}$ and $\boldsymbol{y}$, rather than of $\boldsymbol{x}$ alone. The expression $-\boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})$ is sometimes called the energy of the label $\boldsymbol{y}$, and the inference process defined above is known as energy minimization or MAP inference.

The SSVM training objective is

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m}[\max_{\boldsymbol{y}}(\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y})) - \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i)] \tag{3.9}$$

Here, $\Delta(\boldsymbol{y}_i, \boldsymbol{y})$ is a loss function, which captures the penalty associated with mislabeling $\boldsymbol{y}_i$ as $\boldsymbol{y}$. It must be nonnegative and satisfy the condition that the correct labeling yields a penalty of zero, i.e., $\Delta(z, z) = 0, \forall z \in \text{dom}(\boldsymbol{y})$. The second term in the objective ($\sum_{i=1}^{m}[\max_{\boldsymbol{y}}(\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y})) - \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i)]$) is a convex upper bound on the total empirical loss over the training set ($\sum_{i=1}^{m} \Delta(\boldsymbol{y}_i, \boldsymbol{y}_i^{\text{pred}}(\boldsymbol{w}))$, where $\boldsymbol{y}_i^{\text{pred}}(\boldsymbol{w}) = \text{argmax}_{\boldsymbol{y}}\, \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y})$). The overall SSVM training objective is convex and there exist several efficient methods for solving it [6, 11, 12].

As in the case of the binary SVM, the SSVM can be reformulated to include

constraints and slack variables. The training problem can be rewritten as

$$\text{minimize } \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \tag{3.10}$$

$$\text{subject to } \xi_i \geq \max_{\boldsymbol{y}}(\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y})) - \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{3.11}$$

As in the binary SVM case, ξ_i is the margin constraint violation penalty incurred by each example in the training set. Note that each margin constraint (Eq. (3.11)) can be rewritten as a set of linear constraints:

$$\xi_i \geq (\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y})) - \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i), \forall \boldsymbol{y} \in \text{dom}(\boldsymbol{y}) \tag{3.12}$$

Clearly, there will always be (at least) one linear constraint for which this inequality is tight (otherwise, we could reduce the objective by lowering ξ_i by a small amount). The constraint for which the inequality is tight is also known as the "most violated constraint" associated with example i. Computing the most violated constraint is a crucial subroutine in learning the SSVM.

SSVM Learning

The formulation of the SSVM objective with linear constraints (Eq. (3.10), Eq. (3.12)) is an LP. However, the number of linear constraints is $n|\,\text{dom}(\boldsymbol{y})|$. This can be intractably large, so instead the problem is typically treated in its unconstrained form (Eq. (3.9)) and optimized using the cutting plane method described in Tsochantaridis, *et al.* 2004 [12]. This cutting plane method exploits the sparsity and structure of the problem to reduce the number of constraints needed to be considered. This method requires us to specify how to compute the most violated constraint for each example, which can be done using a procedure known as "loss-augmented inference":

$$\boldsymbol{y}_i^{\text{pred}} = \operatorname*{argmax}_{\boldsymbol{y}} \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) + \Delta(\boldsymbol{y}_i, \boldsymbol{y})$$

Note that this is identical to the standard inference procedure, except that we add the loss into the inference objective.

We will not delve into the particulars of this method here, but mention briefly that the method operates by finding a series of cutting-planes that successively reduce the set of potentially optimal values of $\boldsymbol{w}$ until this set is within ϵ of the solution. At each iteration, the set of most violated constraints yields a subgradient, $\boldsymbol{g}$, for the objective function at the current value of $\boldsymbol{w}$. The subgradient, in turn, defines the cutting-plane for this iteration.

Relationship to binary SVM

The binary SVM can be written within the framework of the SSVM as follows:

$$\boldsymbol{y} \in \{-1, 1\} \tag{3.13}$$

$$\Delta(\boldsymbol{y}, \boldsymbol{z}) = \mathbf{1}\{\boldsymbol{y} \neq \boldsymbol{z}\} \tag{3.14}$$

$$\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \boldsymbol{\phi}(\boldsymbol{x}), & \boldsymbol{y} = 1 \\ \mathbf{0}, & \boldsymbol{y} = -1 \end{cases} \tag{3.15}$$

3.3 Latent Structural SVM

Latent structural SVMs (LSVM) extend the SSVM framework to incorporate *latent* (or hidden) variables. These variables comprise part of the label $\boldsymbol{y}$ and may not be observed at training time. An immediate benefit of this extension is that it can simplify relationships among the observed variables by allowing such relationships to depend on the value of the hidden variable.

Additionally, the use of hidden variables enables the use of diverse data in training. In the non-latent SSVM problem, it is assumed that each training instance is labeled with the same level of annotation, encompassing the entire label vector $\boldsymbol{y}$. In a training set of diverse data, some types of annotation will be provided for all training examples, but other types of annotation will be available for only a subset of training examples. We can incorporate the sometimes-available annotations as latent variables within the LSVM framework. The ability to incorporate diverse datasets into the training regime is valuable, because it is often the case that only a small set of fully annotated data is available (due to the time and/or expense of labeling such data). In

contrast, there may be a much larger source of partially annotated data. We would like to be able to make use of both the limited supply of fully annotated data as well as the large amount of coarser annotations, and the LSVM framework elegantly incorporates such diverse sources of data into a single, unified model.

In the LSVM framework, the inference procedure remains the same as before. At training time, however, we decompose the label into two parts, $\boldsymbol{y} = (\boldsymbol{a}, \boldsymbol{h})$. The annotation, $\boldsymbol{a}$, is the observed portion of the label at training time while the latent variable $\boldsymbol{h}$ is the hidden portion. The training problem is:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m}\max_{\boldsymbol{a},\boldsymbol{h}}(\Delta(\boldsymbol{a}_i, \boldsymbol{a}, \boldsymbol{h}) + \boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}, \boldsymbol{h})) - \frac{C}{m}\sum_{i=1}^{m}\max_{\boldsymbol{h}}\boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}_i, \boldsymbol{h}) \tag{3.16}$$

As in the case of SSVMs, the LSVM training objective is an upper bound on the empirical loss of the dataset, $\sum_{i=1}^{m}\Delta(\boldsymbol{a}_i, \boldsymbol{a}_i^{\text{pred}}, \boldsymbol{h}_i^{\text{pred}})$.

3.3.1 LSVM Learning via CCCP

Unlike the training problems of the binary SVM and structural SVM, the LSVM training problem is not convex. Rather, the objective is the difference of two convex functions. Because it is not convex, it is intractable to solve exactly. We can, however, minimize a convex upper-bound on the objective by iteratively approximating the concave portion of the objective with an affine expression and then optimizing the resulting convex function.

This method is known as the concave-convex procedure (CCCP). Concretely, we have an iterative algorithm with two alternating steps. The first step imputes the latent variables given the ground-truth annotation of each training example, yielding a complete label ($\boldsymbol{y}$) for each example. The second step then solves a standard SSVM problem, using the labels consisting of the ground-truth annotations and the imputed latent variables. To use this method, we therefore have to specify two subroutines. The first is the impute latent variable routine ($\boldsymbol{h}_i^* = \text{argmax}_{\boldsymbol{h}\in\mathcal{H}}\,\boldsymbol{w}^\top\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{h})$, where $\mathcal{H} = \text{dom}(\boldsymbol{h})$) and the second is the loss-augmented inference routine associated with solving the standard SSVM. It can be shown that CCCP converges to a local minimum

Algorithm 1 CCCP

Input: $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}, \boldsymbol{w}_0, \epsilon$

1: $t \leftarrow 0$
2: **repeat**
3: For each training example, i, set $\boldsymbol{h}_i^* = \text{argmax}_{\boldsymbol{h} \in \mathcal{H}} \boldsymbol{w}_t^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{h})$.
4: Compute $\boldsymbol{w}_{t+1}$ by fixing the latent portion of $\boldsymbol{y}_i$ to $\boldsymbol{h}_i^*$ and solving the corresponding SSVM problem. In other words, set $\boldsymbol{w}_{t+1}$ to minimize Eq. (3.9).
5: $t \leftarrow t + 1$
6: **until** objective function (Eq. (3.16)) cannot be decreased below tolerance ϵ.

or saddle-point of the objective [17].

3.3.2 LSVM Learning via Self-Paced Learning

CCCP approximately solves a non-convex problem by iteratively computing a convex upper-bound of the objective and minimizing this bound. Like other methods that use this approach (such as the well-known Expectation Maximization meta-algorithm), CCCP is susceptible to converging to bad local minima. Previous work has sought to address these issues in various ways. A common and straightforward approach is to randomly initialize multiple runs of CCCP and choose the best solution out of these [17]. However, this method is computationally expensive.

Bengio *et al.* [2] proposed an alternative method called curriculum learning. Curriculum learning is inspired by the way humans are taught in school, in which easy material is introduced first and only later are harder topics covered. Analogously, curriculum learning chooses easy examples from the training set to fit the model parameters in initial iterations of the training procedure. In subsequent iterations, it includes more and more of the training set until all examples are included and the algorithm reaches convergence.

The main difficulty of curriculum learning is choosing and computing the metric by which training examples are categorized as "easy" or "hard." In basic applications, it may be straightforward for a human to produce these labels. In real-world applications, however, there is usually no clear-cut distinction and even if there were, it is likely the case that human intuition for what is facile or difficult is not the best

metric to use.

Kumar *et al.* [8] proposed the framework of self-paced learning to address this problem. In self-paced learning, the "hardness" of training example i is defined to be the slack variable, ξ_i, associated with the example. When the slack variable is zero, this means that the example is correctly classified with significant confidence (thus, easy). When it is nonzero, this means that the example is either misclassified or properly classified but close to the decision boundary (i.e., close enough to violate the constraint $0 \geq \max_{\boldsymbol{y}}(\Delta(\boldsymbol{y}_i, \boldsymbol{y}) + \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y})) - \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i))$. The slack is thus a good measure of "hardness" from the classifier's point of view. Using the slack as the hardness score, we can classify examples as "easy" or "hard" (alternatively "valid" or "invalid") by comparing each example's slack value against a chosen threshold.

With this metric in hand, the algorithm proceeds as follows. In the first step, as before, the latent variables for each example are imputed. A hardness threshold, which defines the criterion for categorizing examples as easy or hard, is then set. The second step consists of an *alternate convex search*, which alternates between choosing a valid set of examples (using the current hardness threshold) and then solving the SSVM restricted to those examples. This iterates until convergence. Over time, the hardness threshold is annealed, causing more and more examples to be included in the valid set, until finally, all examples are included in the valid set (at which point, the SPL iterations become equivalent to those of CCCP).

Algorithm 2 Self-paced learning (with slack as hardness metric)

Input: $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}, \boldsymbol{w}_0, \epsilon$

1: $t \leftarrow 0, K \leftarrow K_0$
2: **repeat**
3: For each training example, i, set $\boldsymbol{h}_i^* = \text{argmax}_{\boldsymbol{h} \in \mathcal{H}} \boldsymbol{w}_t^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{h})$.
4: Compute $\boldsymbol{w}_{t+1}$ using **alternate convex search** to solve the relaxed objective $\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^m v_i\xi_i - \frac{1}{K}\sum_{i=1}^m v_i$ subject to the constraints of the regular SSVM problem and $v \in \{0, 1\}$.
5: $t \leftarrow t + 1, K \leftarrow K/\mu$
6: **until** $\forall i, v_i = 1$ and the objective function (Eq. (3.16)) cannot be decreased below tolerance ϵ.

Algorithm 3 Alternate Convex Search

Input: $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}, \boldsymbol{w}_0, \epsilon$

1: **repeat**

2: For each training example, i, set $v_i = \begin{cases} 0 & \xi_i > \frac{1}{K} \\ 1 & \text{otherwise} \end{cases}$.

3: Solve the SSVM over all examples with $v_i = 1$.

4: **until** the objective, $\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m} v_i\xi_i - \frac{1}{K}\sum_{i=1}^{m} v_i$, has converged

Since the last iterations of self-paced learning are equivalent to CCCP, it is guaranteed that this method converges to a local saddle-point or minimum of the LSVM training objective. In practice, to obtain a reasonable initial model for SPL, we can initialize $\boldsymbol{w}$ by running 3 iterations of CCCP.

Chapter 4

Object Context Action Recognition Model

4.1 Overview

We treat the problem as a one-versus-all binary classification problem. For each action class, we train a classifier to detect the presence of that particular action, treating instances of all other actions as negative examples. Alternatively, we could have formulated the problem as a multi-class structural SVM. We choose the binary formulation for several reasons. First, in the multiclass formulation, a fixed set of action classes must be set prior to training. If the desired set of detectable actions changes later, the classifier must be retrained. In the binary case, we simply need to train additional binary classifiers if we wish to consider additional actions. Furthermore, the multiclass formulation assumes that the actions are mutually exclusive. While this assumption holds true for the Pascal data set, one can easily envision a multitasking setting where one person performs multiple actions (phoning while riding a bike, for example). Related to this is the fact that we would like to output a score indicating our confidence in the presence of each action class. This would allow us to vary the decision criterion and compute a full precision-recall curve. In the binary case, we can simply use the score outputted by the classifier for the positive class. Since the score for the negative case is always zero, the score of the positive

case can be used as a measure of confidence across all examples. In the multi-class SSVM, however, the score does not lend itself to this interpretation. We only expect that the score of the ground-truth action be greater than the score of all other actions (assuming there is a single ground-truth action). Thus, there is no common reference point, like the zero score negative case, that would allow us to calibrate scores across different examples. Moreover, there is even less of a clear interpretation for the scores of the lower-scoring action classes; the training objective places no constraint on the ordering of the scores of the non-ground-truth actions.

Note, however, that even though we formulate the task of action detection as a binary classification problem, we still cast our model as a (latent) structural SVM, rather than a simple binary SVM. This is because the label, $\boldsymbol{y}$, that the model outputs at inference time contains not only the binary action presence variable but also additional (potentially hidden) variables, such as object and person location, that are also part of the complete label.

4.2 Notation

For each action class, we define a LSVM model as follows. Each example is a potential action instance, i.e., an instance of a person performing the action of interest. Let $\boldsymbol{x}$ be the observed data gathered from the image and let $\boldsymbol{y}$ be the label we wish to predict for this example. The label $\boldsymbol{y}$ indicates the presence or absence of the action class associated with the model and, if the action is present, the location of the person performing the action and the location of the object upon which the person is acting.

We denote the joint feature vector over $\boldsymbol{x}$ and $\boldsymbol{y}$ as $\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})$. The energy of a predicted label $\boldsymbol{y}$ is equal to $-\boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{w}$ parameterizes the model. The best label is obtained by MAP inference, in which we minimize the energy over $\boldsymbol{y}$ to obtain $\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})$.

Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{a}_i)\}_{i=1}^m$ denote the training set, where $\boldsymbol{x}_i$ is the observed variables of the action instance and $\boldsymbol{a}_i$ is the corresponding annotation given at training time. In a fully supervised training regime, each training example is annotated with the full label $\boldsymbol{y}$ (i.e., $\boldsymbol{a} = \boldsymbol{y}$). In the case of learning from diverse data, the training set

is split into subsets, each of which has its own level of annotation. Note, however, that the annotation $\boldsymbol{a}$ always indicates whether the example is positive or negative. For each instance $\boldsymbol{x}_i$ with annotation $\boldsymbol{a}_i$, we also specify a set of latent (or hidden) variables $\boldsymbol{h}_i$ such that $(\boldsymbol{a}_i, \boldsymbol{h}_i)$ defines the label $\boldsymbol{y}_i$ for the instance.

Let the loss function be $\Delta(\boldsymbol{a}_i, \boldsymbol{a}, \boldsymbol{h})$. This captures the desired penalty associated with assigning a training example the label $(\boldsymbol{a}, \boldsymbol{h})$ when the ground-truth annotation is $\boldsymbol{a}_i$.

4.3 Model Definition

At inference time, the model associated with a particular action (riding bike, riding horse, or using computer) is given an image and returns an indicator of whether or not the action is present and (if the action is present) the locations of the person and object that comprise the action and the type of the object (bicycle, motorbike[1], horse, or computer monitor). Various parts of the label may be given at inference time (the PASCAL Action Classification Challenge, for example, specifies that the location of the person is given). In this case, the model will make use of the given information and infer the unknown remainder of the label.

We define the loss function to be the zero-one loss over the action presence:

$$\Delta(\boldsymbol{a}_i, \boldsymbol{a}, \boldsymbol{h}) = \begin{cases} 1 & \text{if } \boldsymbol{a}_i \text{ and } \boldsymbol{a} \text{ agree on the presence of the action} \\ 0 & \text{otherwise} \end{cases}$$

Note that this means we do not incur a penalty if the action is predicted correctly, even if the predicted object and/or person is incorrect.

We cast our model within the framework of the LSVM described above. We define $\boldsymbol{x}$, the observed data, to be the object and person detector scores at each possible location in the image. In practice, we use an off-the-shelf sliding window object detector to compute these scores for a comprehensive sample of bounding box

[1]The decision to have 2 object classes associated with riding a bike was due to the availability of two separate object detectors for bicycle and motorbike

locations at different scales. This yields a list of person and object candidates. Then, for tractability reasons, we consider only the 10 highest-scoring person candidates and the 2,000 highest-scoring object candidates.

The specific object detector used for both people and action objects in our experiments was the deformable parts object detector from Felzenszwalb, *et al.* 2008 [4]. In addition to outputting an object detection score, this detector also outputs auxiliary scores, which we utilize in our model. These auxiliary scores exist because the detector breaks each object class model into several different sub-models or *components* and outputs a score for each component (the overall detection score is the max of the component scores). The motivation behind doing so is that human-specified object categories (e.g., "car") can often be split into distinctive sub-categories (e.g., "car from front," "car from side"). Because these sub-categories are often very different from each other in appearance, it makes sense to learn a separate model for each, rather than burden a single model with the task of simultaneously recognizing all of the sub-categories. In our action model, we include not only the overall person detection score outputted by the object detector, but also the detection score associated with each of the person components. Intuitively, each person component will be associated with a different person sub-category, which may correlate with pose (e.g., sitting versus riding), which in turn may correlate with different action classes (e.g., a sitting pose is more consistent with using a computer than a riding pose). By default, when the deformable parts detector is trained, each object component is initialized using a subset of the training data based upon a clustering of the aspect ratio of the ground-truth object bounding boxes. In addition to using the component scores from this initialization, we can also initialize the components with training subsets based on the ground-truth action. Note that the training data used to train the person detector does not contain action annotations. Thus, we cannot directly partition the person detector training set by action class. As a substitute, we compute the mean person HOG descriptor for each action class over the *action* training data (for which the action label is provided) and then cluster the person detector training data based on the L2 distance from the mean person HOG descriptor of each action. Initializing the components using these clusters, we retrain the person detector, and then use it

1.	Object score
2.	Person score
3.	Person component scores
4.	Shape context
5.	Relative size

Figure 4.1: List of features used in object-context model

to produce action-based component scores for each example.

Recall that the label $\boldsymbol{y}$ includes an indicator variable for the presence of the action (1 for a positive example of the action of interest, -1 for a negative example). If the action is present, then $\boldsymbol{y}$ also includes the person location, object type, and object location of the action example[2]. Given $\boldsymbol{y}$ and $\boldsymbol{x}$, we define a joint feature vector $\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{y})$ that includes the features in Figure 4.1 in a quadratic kernel. These features are described in more detail below:

1. The object score is the score outputted by the object detector for the object class associated with the action of interest. This is the score for the object detection at the location specified in $\boldsymbol{y}$.

2. The person score is the score outputted by the person detector for the person at the location specified in $\boldsymbol{y}$.

3. The person component scores include both the aspect-ratio-based component scores and the action-based component scores from the deformable parts person detector. These provide a human-pose-based cue for the action class.

4. The shape context features consist of an indicator vector that indicates the discretized relative angle and distance between person and object. There are 7 angle and 7 object bins ($7 \times 7 = 49$ shape bins overall), and the indicator is bilinearly interpolated (making it a soft indicator).

5. The relative size features consist of a soft (linearly interpolated) indicator vector of the relative size of the object to the person, discretized into 7 bins.

[2]Note that object type is determined by action type.

The last two sets of features capture the spatial relationship between person and object. To compute the shape context vector, we first compute the angle, θ, between the center of the person bounding box and the center of the object bounding box. We then compute the relative distance, d, from the object to the person. This is the raw pixel distance between the center of the two bounding boxes divided by the square-root of the person bounding box area. The raw values of (θ, d) will fall into one of 49 bins (7 for θ, 7 for d)[3]. We construct the (soft) indicator vector over these bins, $f \in \mathbf{R}^{49}$, smoothing with bilinear interpolation. We apply the same method to compute a soft indicator vector for relative size (discretized into 7 bins), which is the ratio of object to person bounding box area.

4.4 Learning

4.4.1 Diverse Data

Our training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{a}_i)\}_{i=1}^m$ consists of action instances with 3 levels of annotations, which we call "high," "medium," and "low." In the high level, we have $\boldsymbol{a} = \boldsymbol{y}$. In other words, we are given a bounding box around the person, the action label, a bounding box around the object (if applicable), and the object type. For the medium level, we have that $\boldsymbol{a}$ is the person location and action type, but the object remains hidden. That is, $\boldsymbol{h}$ is the object location and type. In the low level of annotation, we are given only an image and an action label indicating that the action is present in the image. In this case $\boldsymbol{h}$ includes the person location as well as the object location and type.

4.4.2 Imputing Latent Variables

The first step of both CCCP and SPL requires imputing the latent variables for the set of training examples. Given the data and the annotation, we impute the latent

[3]Bin boundaries were determined by visualizing the values of (θ, d) over the training set on a 2D scatter plot. The number of bins was determined via empirical cross-validation.

variable to minimize the energy, i.e.,

$$\boldsymbol{h}_i^* = \max_{\boldsymbol{h}} \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}_i, \boldsymbol{h}) \tag{4.1}$$

For training examples with the high level of annotation (i.e., both the ground-truth object and person are given), $\boldsymbol{h}$ is empty, so this step is trivial. Note that if all training examples are annotated at this level, the training procedure reduces to that of a standard (non-latent) structural SVM.

For training examples with medium annotation (i.e., the object is missing), we generate a list of object candidates using the deformable parts object detector trained on the 4 object classes. (Note that these object candidates never change, so we can pre-compute and cache them for the duration of the entire training procedure.) To produce $\boldsymbol{h}_i^*$, we iterate through all candidates of the object type associated with the action label specified by $\boldsymbol{a}_i$. For the low-annotation examples, we use the object detector to generate a list of person candidates, as well. To produce $\boldsymbol{h}_i^*$, we consider every person-object candidate pair where the object is consistent with the action label.

4.4.3 Most Violated Constraint

Having imputed the latent variables $\boldsymbol{h}$, we must solve the resulting (non-latent) structural SVM. To do so, we must specify the loss-augmented inference procedure (i.e., compute the most violated constraint for each training example). Recall that the standard constrained optimization problem associated with the SSVM with imputed latent variables, $\boldsymbol{h}^*$, is

$$\text{minimize} \quad \frac{1}{2} \|\boldsymbol{w}\|_2^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \tag{4.2}$$

$$\text{subject to} \quad \xi_i \geq \sum_{i=1}^{m} \Delta(\boldsymbol{a}_i, \boldsymbol{a}, \boldsymbol{h}) + \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}, bh) - \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}_i, \boldsymbol{h}_i^*), \quad \forall \boldsymbol{a}, \boldsymbol{h} \tag{4.3}$$

In most SSVM applications, the loss is non-zero if the label is incorrect. In our case, however, the loss can be zero even if the object and person label are incorrect, as long as the action label is correct. In other words, we care only about predicting the correct action label, rather than the complete label. The object and person label are merely means to an end. Consequently, blindly optimizing the vanilla SSVM training objective yields unwanted constraints. In particular, we do not want to declare constraints over ξ_i associated with labels $(\boldsymbol{a}, \boldsymbol{h})$ for which $(\boldsymbol{a}, \boldsymbol{h}) \neq (\boldsymbol{a}_i, \boldsymbol{h}_i^*)$ **and** $\text{action}(\boldsymbol{a}) = \text{action}(\boldsymbol{a}_i)$. That is, we do not care if the classifier gets the object and person wrong, as long as it correctly predicts whether the action is present or not. (Note that we still want the constraint for $(\boldsymbol{a}, \boldsymbol{h}) = (\boldsymbol{a}_i, \boldsymbol{h}_i^*)$, which effectively ensures $\xi_i \geq 0$.) Eliminating these constraints yields the following training problem:

$$\text{minimize} \quad \frac{1}{2} \|\boldsymbol{w}\|_2^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i \tag{4.4}$$

$$\text{subject to} \quad \xi_i \geq \sum_{i=1}^{m} \Delta(\boldsymbol{a}_i, \boldsymbol{a}, \boldsymbol{h}) + \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}, \boldsymbol{h}) - \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}_i, \boldsymbol{h}_i^*), \tag{4.5}$$

$$\forall \boldsymbol{a}, \boldsymbol{h} \in \{\boldsymbol{a}_i, \boldsymbol{h}_i^*\} \cup \{\boldsymbol{a}, \boldsymbol{h} : \text{action}(\boldsymbol{a}) \neq \text{action}(\boldsymbol{a}_i)\} \tag{4.6}$$

As with the standard SSVM problem, we can rewrite this in unconstrained form and optimize the unconstrained objective with the cutting-plane method described in the Background chapter. This method requires specifying a procedure for finding the most violated constraint for each training example, which we now describe. As stated before, the most violated constraint can be computed using a loss-augmented inference procedure. Given our modified set of constraints, this corresponds to the following step:

$$(\hat{\boldsymbol{a}}, \hat{\boldsymbol{h}}) := \underset{\{\boldsymbol{a}_i, \boldsymbol{h}^*\} \cup \{\boldsymbol{a}, \boldsymbol{h} : \text{action}(\boldsymbol{a}) \neq \text{action}(\boldsymbol{a}_i)\}}{\text{argmax}} (\Delta(\boldsymbol{a}_i, \boldsymbol{a}, \boldsymbol{h}) + \boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{a}, \boldsymbol{h})) \tag{4.7}$$

In practice, we compute this argmax by searching over all person-object candidate pairs in the image that are consistent with the annotation $\boldsymbol{a}_i$. This search yields a

label $\hat{\boldsymbol{y}} = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{h}})$, which is the label that generates the most violated constraint for the training example.

4.4.4 Self-Paced Learning

Choosing valid examples

Recall that in the standard self-paced learning setting, each iteration (after latent variables have been imputed) requires optimizing the objective

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m} v_i\xi_i - \frac{1}{K}\sum_{i=1}^{m} v_i$$

Here, K is the self-paced learning weight, which controls how many training examples are used to train the relaxed SSVM objective ($(1/K)(C/m)$ is the slack threshold below which examples are included, so a higher K means fewer examples will be included). Note that this selection criterion looks only at the slack value of a training example given the current model and is agnostic to the ground-truth annotation of the training example. This setting is fine if each label is equally "hard" – that is, if applying this threshold includes about an equal fraction of examples of each label into the valid set. However, if one label is easier to classify than the other, then applying this threshold will result in fitting the model exclusively to the "easy" labels and ignoring the "hard" ones.

In our case, the two labels we wish to distinguish are the positive (action present) and negative (action not present) cases. In practice, we found that the initial valid set included exclusively negative examples, ultimately leading to convergence to a poor local minimum (in which the accuracy of the model is high for negative examples but low for positive examples, resulting in low average precision). To rectify this issue, we maintain a separate threshold for each label class (that is, the selection criterion is different for positive and negative examples). The new objective is

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m} v_i\xi_i - \frac{1}{K_{\text{neg}}}\sum_{i:\text{neg}} v_i - \frac{1}{K_{\text{pos}}}\sum_{i:\text{pos}} v_i$$

We initialize these thresholds so that the initial valid set includes the same fraction of positive and negative examples. In subsequent iterations, we anneal these thresholds at the same rate, conducting the following update

$$K_{\text{neg}} := K_{\text{neg}}/\mu$$
$$K_{\text{pos}} := K_{\text{pos}}/\mu$$

4.5 Meta-parameters

The LSVM training procedure requires the following meta-parameters to be specified: the slack penalty coefficient (C), the initial SPL weights ($K_{\text{neg}}, K_{\text{pos}}$), and the SPL annealing factor (μ). Following the example of Kumar, *et al.* [8], we set the initial SPL weights such that half of the positive and negative examples are included in the initial SPL iteration. We set the annealing factor μ to 5.0, which yields a reasonable convergence rate of the objective. Finally, we set C using cross-validation by training a fully supervised model on the PASCAL training set (with object annotations) and validating on the validation set.

Chapter 5

Results and Discussion

5.1 Data

The PASCAL VOC 2010 Action Classification dataset contains 908 images and 1,221 action instances, roughly evenly split across 9 action classes (phoning, playing an instrument, reading, riding a bike, riding a horse, running, taking a photo, using a computer, and walking). The set is partitioned into a test set of 454 images that contain 613 action instances, a training set of 226 images containing 301 action instances, and a validation set of 228 images with 307 action instances. The images contain a high amount of clutter and occlusion, as well as a high variance in setting and point of view. These attributes make the dataset particularly challenging.

In addition, we also incorporate the use of 130 weakly labeled images obtained from Google Image Search. These images are labeled only with the action present (i.e., no person and no object location). There are 26 weak images of riding bicycle, 35 of riding horse, 32 of running, and 37 of walking. (Note that there are no images of computer use in the weak image set, and therefore we do not expect the additional weak data to improve the "using computer" action model.)

Though the dataset includes additional action classes, we only train models for riding bicycle, riding horse, and using computer. (The entire dataset is still utilized, however, because the other action classes serve as negative examples.) We excluded running and walking because they do not explicitly involve objects. Furthermore, we

limited our focus to the remaining action classes for which we could obtain reasonable object detection candidates. The decision to focus on the three actions mentioned above was thus motivated by the limitations of the object detector (discussed in the following section).

The Pascal validation set was used for model selection. After the model parameters were fixed, final results were produced by training on all of the PASCAL 2010 Action training and validation data (and also the weakly labeled images, when applicable) and evaluating on the PASCAL test set.

5.2 Object Detector

As mentioned earlier, we utilized the deformable parts object detector to generate detection scores for objects and people. As mentioned earlier, we utilize three types of scores from this detector – (1) the overall object detection score, (2) the aspect-ratio-based component scores of the default model (the max of these is the overall object detection score), and (3) the action-based component scores. Scores (1) and (2) were obtained from the off-the-shelf person model (i.e., we did not retrain on any additional images from the action dataset or elsewhere). Obtaining score (3) required retraining the model with the new component initializations. This new model was trained on the same data as the off-the-shelf model (the PASCAL Object and INRIA datasets).

For tractability purposes, our action model considers only the top 2,000 object candidates outputted by the object detector. The recall of this set of candidates over ground-truth objects in the PASCAL Action training and validation set is reported in Figure 5.1. The recall of the candidate objects set is a limiting factor on the performance of our model, because we rely on object presence and position as a primary cue for action recognition. We focus on action classes for which there is a strong object context cue – namely, riding bike, riding horse, and using computer.

We hypothesize that the poor performance of the object detector on the remaining object classes was due to discrepancies between the appearance of objects in the datasets used to train the object detector and the appearance of objects in the Pascal

Figure 5.1: Recall of the top 2,000 object detections outputted by the deformable parts object detector. A data example was marked a true positive if the overlap (intersection over union) between a detection and the ground-truth object bounding box was greater than 0.5.

Bicycle	0.936508
Motorbike	0.952381
Horse	1.000000
Computer monitor	0.857143
Phone	0.481481
Camera	0.661290
Instrument	0.610000
Book	0.754717

Action dataset. In particular, the data used to train the object detector included more detailed close-up shots while the objects in the Pascal Action data were often small, especially for small object types such as phone and camera. Other object classes were probably too generic (e.g., "musical instrument" and "book," which encompassed newspapers and magazines in addition to actual books), even for the deformable parts object detector to handle. (Recall that this object detector models each object class as a collection of more homogeneous sub-classes.)

5.3 Experiments

5.3.1 Training Set Notation

Recall that we have two sources of training data – the PASCAL Action data (training and validation set) and the set of images obtained from Google Images. These data sources feature different levels of annotation. The raw PASCAL data is labeled with the ground-truth action and person. Furthermore, to incorporate more detailed information into the training procedure, we also labeled the PASCAL images with ground-truth objects. The images from Google are labeled only with the ground-truth action. For ease of understanding, we henceforth refer to the PASCAL Action data (training *and* validation set) as the "PASCAL" set. We refer to the PASCAL

data augmented with ground-truth object labels as the "PASCAL(Object)" set. We refer to the set of weakly labeled images from Google Images as the "Weak" set. We use the plus sign to denote the union of two training sets (e.g., "PASCAL+Weak" denotes the combination of the standard PASCAL training and validation data with the weakly labeled data).

5.3.2 Overview

To obtain a good baseline for comparison, we train a simple LSVM model whose only feature is the object score (in a quadratic kernel). This model therefore predicts an action label by considering only the strength of the best object detection in the image. We trained the baseline model on both the PASCAL(Object) set and the PASCAL(Object)+Weak set. In the former case, training is fully supervised and there are no latent variables to impute, so the choice of CCCP versus SPL is moot (both reduce to solving a standard SSVM). In the latter case, we use CCCP as our baseline LSVM learning procedure.

Our full LSVM object context model was tested on a variety of datasets, using both CCCP and SPL as learning procedures. Full results are presented in Figures 5.2 and 5.3. These results demonstrate the following points:

1. LSVM models are capable of taking advantage of additional, weakly labeled data.

2. The additional features in the full object context model allow it to perform better than the baseline.

3. SPL performs better than CCCP, on average.

4. SPL outperforms CCCP incrementally more when additional weakly labeled data is added.

In the following sections, we justify the above conclusions and discuss the results in further detail.

Average Precision			
Model	Riding Bike	Riding Horse	Using Computer
Baseline, PASCAL(Object)	.717	.802	.332
Baseline, PASCAL(Object)+Weak, CCCP	.725	.810	.292
Previous State-of-the-Art, PASCAL	.810	.897	**.641**
PASCAL(Object)	.821	.885	**.475**
PASCAL, CCCP	.838	.790	.444
PASCAL+Weak, CCCP	.828	.902	.420
PASCAL, SPL	.839	.898	.414
PASCAL+Weak, SPL	**.861**	**.911**	.472

Figure 5.2: Average precision on the Pascal test set for different models. Yellow indicates the best for each action class. Cyan indicates the best of our models that used only the Pascal training and validation data to train (i.e., no weak images).

5.3.3 Object Context Model vs. Baseline

The precision-recall curves of Figures 5.4 and 5.5 clearly demonstrate that the object context model outperforms the baseline. This demonstrates that detecting object presence alone is insufficient to accurately model the presence of action. Intuitively, for example, an action model that does not consider the spatial relationship of the object to the human cannot distinguish between a parked bike and a bike being ridden.

Moreover, the use of such object-person spatial context features can mitigate the impact of weak false positives outputted by the object detector. The probability of the object detector outputting a false positive in a location that is consistent with the corresponding action is much lower than the probability of a false positive being detected anywhere in the image. In the case of computer monitors especially, the detector is susceptible to misclassifying other rectangular objects, such as picture frames and windows, as monitors. Such objects are common in everyday scenes, but less frequently are they positioned directly next to a person's head (the location consistent with a monitor participating in the using computer action).

We also note that there is significant variation in the relative difficulty of the

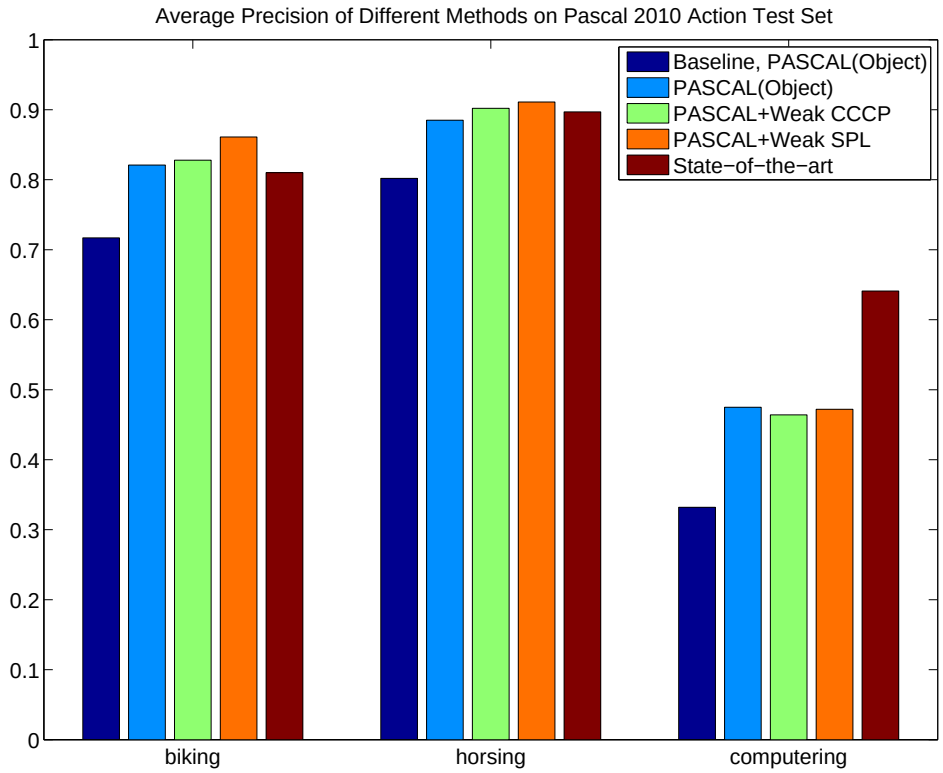
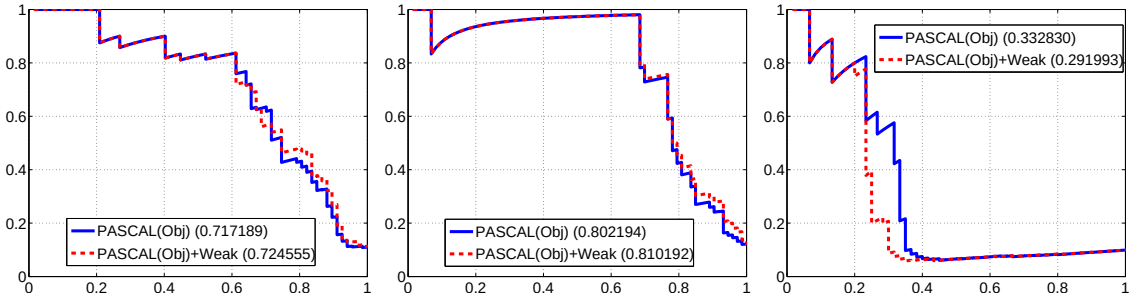
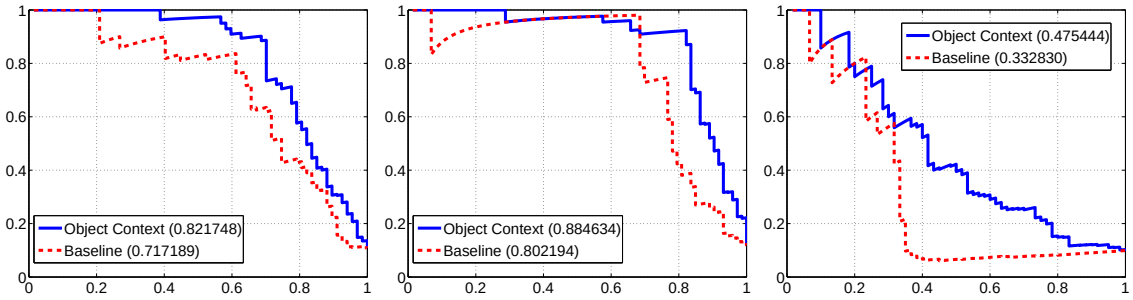


Figure 5.3: The results in Figure 5.2 presented graphically. The "diverse data" bars are the max of the corresponding "PASCAL+Weak" and "PASCAL(Object)+Weak" entries in the table.



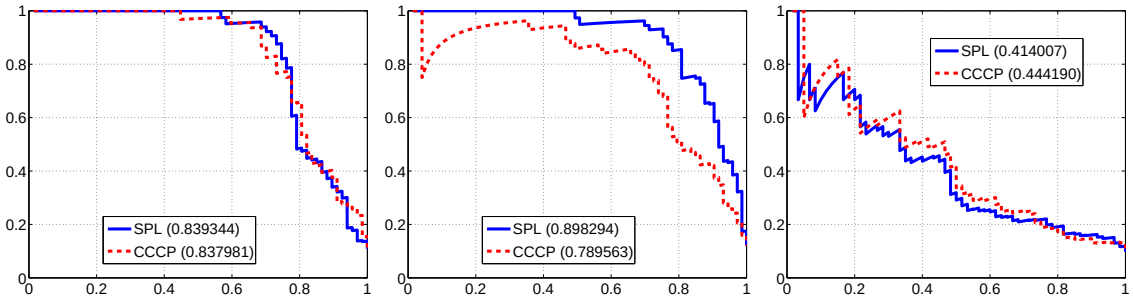
(a) Riding bike (b) Riding horse (c) Using computer

Figure 5.4: Precision vs. recall for baseline classifier, trained on both PASCAL(Object) and PASCAL(Object)+Weak datasets.

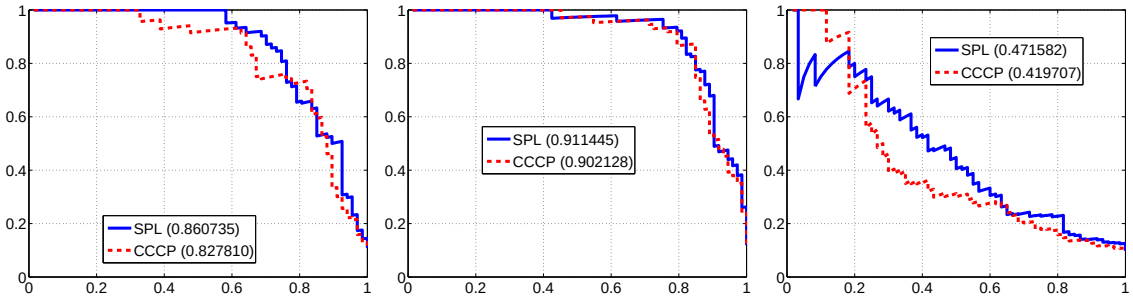


(a) Riding bike (b) Riding horse (c) Using computer

Figure 5.5: Precision vs. recall for the object context model, trained on the combination of the training and validation sets.



(a) PASCAL (riding bike, riding horse, using computer)



(b) PASCAL+Weak (riding bike, riding horse, using computer)

Figure 5.6: Comparison of CCCP vs. SPL in training the latent object context model on the PASCAL and PASCAL+Weak datasets.

three action classes. We achieve higher average precision on the actions associated with easy-to-detect objects, but achieve larger gains over the baseline for actions with

more difficult objects. Bicycles and horses are complex objects with multiple well-defined parts (e.g., wheels, handlebars, legs). Computer monitors, on the other hand, are very simple rectangular objects. This simplicity makes them easily confusable with many other simple rectangular objects such as windows, picture frames, newspapers, and even the backs of chairs. Thus, the performance of the object detector is much higher for bicycles and horses than monitors. This discrepancy has a large impact on the performance of the action recognition model, which relies heavily on the object detection. On the other hand, however, the baseline model is completely reliant on the object detection while the object context model can overcome bad detections in some cases for the reasons mentioned above. Thus, the improvement achieved by using the full model is higher for the harder using-computer action class.

5.3.4 Diverse Data and SPL vs. CCCP

The performance of the baseline classifier illustrates that even a simple LSVM model can benefit from the inclusion of additional weakly labeled data (Figure 5.4). The results show that the use of additional weakly annotated training data improves the baseline model on the "riding bicycle" and "riding horse" actions. The baseline performs worse on the using computer class, but recall that the weak image set lacks computer images, so we do not expect it to enable an improvement in this action class.

Though it shows minor improvement with diverse data, the baseline model is too simplistic to take full advantage of the additional training data. With the full object context model, the gains are generally more pronounced, as we would expect for a more sophisticated approach. Figure 5.6 compares the performance of the object context model trained using SPL versus CCCP. For "riding horse" and "riding bike," we see an across-the-board improvement using self-paced learning. For "using computer," CCCP performs better with the standard PASCAL training data, but SPL performs better with the complete set of diverse data. In fact, in the case of riding bike and using computer, the performance of CCCP actually drops when the weakly annotated data is included, due to convergence to a bad local optimum.

5.3.5 Error Analysis

Qualitative analysis revealed that in many cases, the action model outputted a false positive when the object detection was especially strong, even though the object detecton was not located in a position consistent with the action. For example, if there was a very strong horse detection in an image, a person in the image would likely be labeled as riding that particular horse, even if the horse was located far away from the person. This problem is rooted in the fact that, in our model, object detection strength and relative spatial location are modeled as independent features. One way to address this issue would be to introduce features that jointly capture the value of the object detection and the relative location of the object to the person. In other words, we could extend the shape context features to include a dimension for object detection score. Note that this would increase the number of shape-context features by a factor of the number of buckets associated with the object score dimension. An alternative solution that avoids this increase in feature quantity would be to use the original shape-context buckets, but weight the feature values by the object score (instead of using the standard soft-indicator feature value).

We note, however, that both these solutions may ultimately hurt the overall performance of the model on the PASCAL test set. This is due to the fact that there are many images in the dataset in which multiple people are performing the same action (e.g., a group of people riding bicycles). In these images, some people or objects are often small in size or partially included. However, there is usually one object that is relatively large and completely visible and in many cases the model infers that this object partakes in multiple action instances (e.g., it infers that multiple people are all riding the same horse). While semantically incorrect, this "error" does yield the correct action label (see Figure 5.7).

5.4 Future Work

Our model can be directly applied to additional action classes. An obvious next step would be to evaluate on the remaining 6 PASCAL Action classes (phoning,

Figure 5.7: Example of an image where action is inferred correctly, but the object location is incorrect.

playing instrument, reading, running, taking photo, walking). To enable the model to perform well on these action classes, the quality of the object detections for the corresponding object classes must be improved. Retraining the object detector on an object dataset that is more representative of the level of appearance detail of objects in the action dataset as well as increasing the bounding box search scale of the object detector to better capture smaller objects such as phones and cameras should significantly improve the recall of the object detector. For objectless actions such as running and walking, the deformable parts component scores for the person detection, which are currently incorporated as features in our action model, can provide cues related to human pose. Additional pose features, such as the scores and locations of individual parts in the deformable parts model (as opposed to just the score and location of the overall person), can also be experimented with. Alternatively, pose can be incorporated explicitly into the model as in Yao, *et al.* 2010 [15] and Gupta *et al.* 2009 [5]. The benefits of using diverse data would become even more important in this scenario, as a fully supervised training regime would require annotating a large number of images with individual human parts.

Appendix A

Qualitative Results

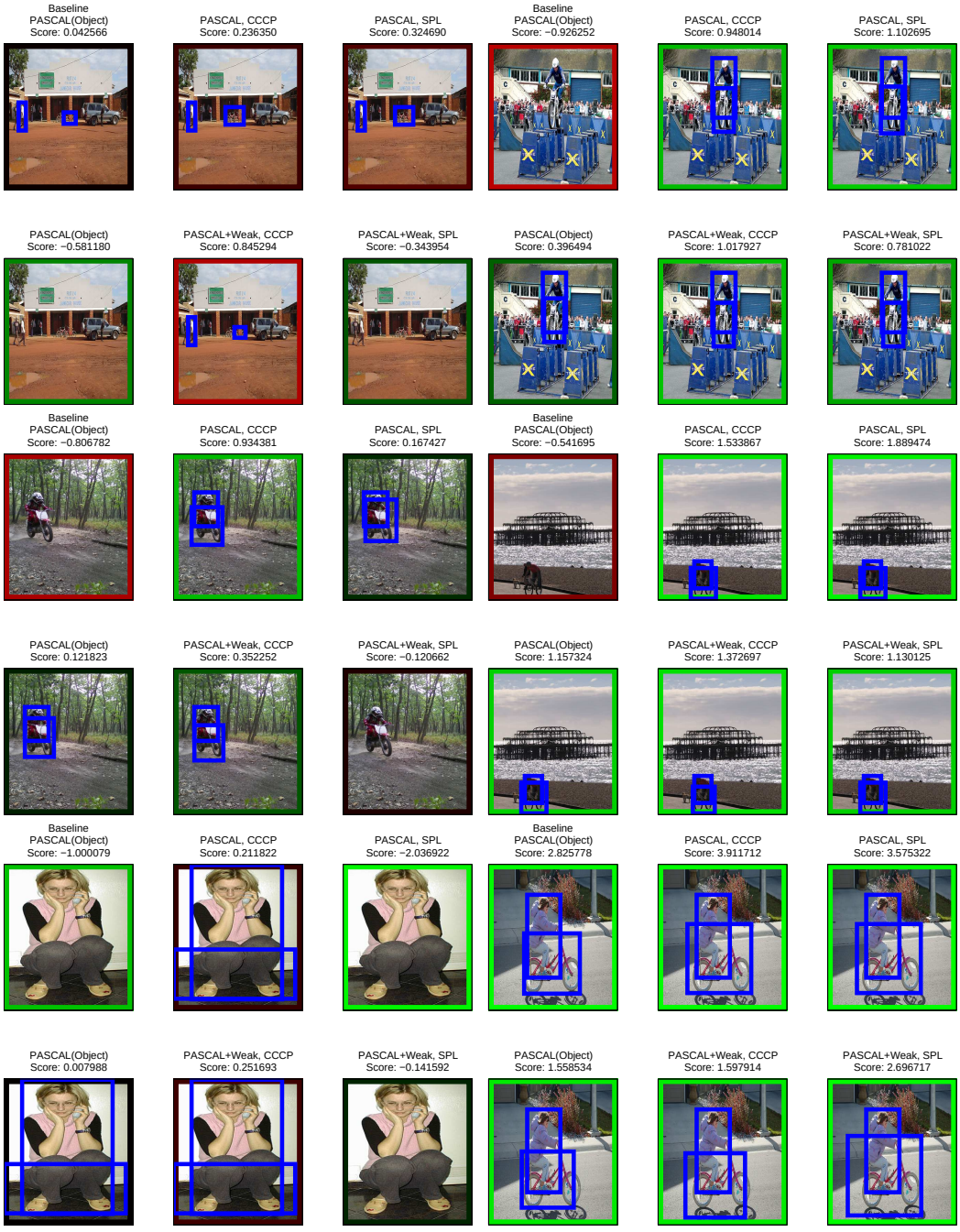
This appendix contains selected qualitative results on the 2010 PASCAL Action test set with models trained on different training sets with different training algorithms (see Figure A.1). The outline of the image indicates whether the action prediction is correct. Green denotes correct, red incorrect; a brighter color indicates greater confidence in the prediction (i.e., an action score further away from 0). If the action was detected, then blue boxes show the location predicted for the person and object. Note that the person bounding box is given at inference time.

Baseline PASCAL(Object)	Object Context PASCAL CCCP	Object Context PASCAL SPL
Object Context PASCAL(Object)	Object Context PASCAL+Weak CCCP	Object Context PASCAL+Weak SPL

Figure A.1: Format of qualitative results displayed in this section. Note that training on PASCAL(Object) is a fully supervised problem and therefore does not require an approximate technique like CCCP or SPL


Baseline
PASCAL(Object)
Score: −0.216290
PASCAL, CCCP
Score: −0.978807
PASCAL, SPL
Score: 4.153873
Baseline
PASCAL(Object)
Score: −0.216290
PASCAL, CCCP
Score: −0.975870
PASCAL, SPL
Score: 0.937526
PASCAL(Object)
Score: 2.033677
PASCAL+Weak, CCCP
Score: 3.271145
PASCAL+Weak, SPL
Score: 3.446858
PASCAL(Object)
Score: 0.372091
PASCAL+Weak, CCCP
Score: 0.348355
PASCAL+Weak, SPL
Score: 0.556367
Baseline
PASCAL(Object)
Score: −0.739181
PASCAL, CCCP
Score: −0.983399
PASCAL, SPL
Score: 1.080084
Baseline
PASCAL(Object)
Score: −0.817139
PASCAL, CCCP
Score: −0.987308
PASCAL, SPL
Score: 2.179703
PASCAL(Object)
Score: 0.363339
PASCAL+Weak, CCCP
Score: −0.336292
PASCAL+Weak, SPL
Score: 1.062826
PASCAL(Object)
Score: 0.928011
PASCAL+Weak, CCCP
Score: 0.430282
PASCAL+Weak, SPL
Score: 0.886822
Baseline
PASCAL(Object)
Score: −0.765883
PASCAL, CCCP
Score: −0.982394
PASCAL, SPL
Score: 2.513462
Baseline
PASCAL(Object)
Score: −0.801628
PASCAL, CCCP
Score: −0.996848
PASCAL, SPL
Score: −0.430242
PASCAL(Object)
Score: 0.877366
PASCAL+Weak, CCCP
Score: 1.343630
PASCAL+Weak, SPL
Score: 2.160387
PASCAL(Object)
Score: 0.020251
PASCAL+Weak, CCCP
Score: −0.148894
PASCAL+Weak, SPL
Score: 1.144067


Baseline
PASCAL(Object)
Score: 3.087141
PASCAL, CCCP
Score: −0.819469
PASCAL, SPL
Score: 4.758886
Baseline
PASCAL(Object)
Score: −0.100671
PASCAL, CCCP
Score: −0.967043
PASCAL, SPL
Score: 3.932318
PASCAL(Object)
Score: 3.431386
PASCAL+Weak, CCCP
Score: 4.525849
PASCAL+Weak, SPL
Score: 5.019714
PASCAL(Object)
Score: 1.870171
PASCAL+Weak, CCCP
Score: 3.513284
PASCAL+Weak, SPL
Score: 3.606131
Baseline
PASCAL(Object)
Score: −0.261634
PASCAL, CCCP
Score: −0.977774
PASCAL, SPL
Score: 0.670785
PASCAL(Object)
Score: 0.084832
PASCAL+Weak, CCCP
Score: 0.318313
PASCAL+Weak, SPL
Score: 0.449417


Baseline
PASCAL(Object)
Score: 0.042566
PASCAL, CCCP
Score: 0.236350
PASCAL, SPL
Score: 0.324690
Baseline
PASCAL(Object)
Score: −0.926252
PASCAL, CCCP
Score: 0.948014
PASCAL, SPL
Score: 1.102695
PASCAL(Object)
Score: −0.581180
PASCAL+Weak, CCCP
Score: 0.845294
PASCAL+Weak, SPL
Score: −0.343954
PASCAL(Object)
Score: 0.396494
PASCAL+Weak, CCCP
Score: 1.017927
PASCAL+Weak, SPL
Score: 0.781022
Baseline
PASCAL(Object)
Score: −0.806782
PASCAL, CCCP
Score: 0.934381
PASCAL, SPL
Score: 0.167427
Baseline
PASCAL(Object)
Score: −0.541695
PASCAL, CCCP
Score: 1.533867
PASCAL, SPL
Score: 1.889474
PASCAL(Object)
Score: 0.121823
PASCAL+Weak, CCCP
Score: 0.352252
PASCAL+Weak, SPL
Score: −0.120662
PASCAL(Object)
Score: 1.157324
PASCAL+Weak, CCCP
Score: 1.372697
PASCAL+Weak, SPL
Score: 1.130125
Baseline
PASCAL(Object)
Score: −1.000079
PASCAL, CCCP
Score: 0.211822
PASCAL, SPL
Score: −2.036922
Baseline
PASCAL(Object)
Score: 2.825778
PASCAL, CCCP
Score: 3.911712
PASCAL, SPL
Score: 3.575322
PASCAL(Object)
Score: 0.007988
PASCAL+Weak, CCCP
Score: 0.251693
PASCAL+Weak, SPL
Score: −0.141592
PASCAL(Object)
Score: 1.558534
PASCAL+Weak, CCCP
Score: 1.597914
PASCAL+Weak, SPL
Score: 2.696717


Baseline
PASCAL(Object)
Score: −0.997621
PASCAL, CCCP
Score: 1.352738
PASCAL, SPL
Score: 1.338077
Baseline
PASCAL(Object)
Score: −0.984462
PASCAL, CCCP
Score: 3.088346
PASCAL, SPL
Score: 4.191919
PASCAL(Object)
Score: −0.856014
PASCAL+Weak, CCCP
Score: −1.004020
PASCAL+Weak, SPL
Score: 0.453436
PASCAL(Object)
Score: −0.443325
PASCAL+Weak, CCCP
Score: −0.600879
PASCAL+Weak, SPL
Score: 2.151762
Baseline
PASCAL(Object)
Score: −0.995433
PASCAL, CCCP
Score: 1.766945
PASCAL, SPL
Score: 2.443711
Baseline
PASCAL(Object)
Score: −0.999243
PASCAL, CCCP
Score: 1.655263
PASCAL, SPL
Score: 0.564572
PASCAL(Object)
Score: −0.880569
PASCAL+Weak, CCCP
Score: −0.833452
PASCAL+Weak, SPL
Score: 1.347215
PASCAL(Object)
Score: −0.913670
PASCAL+Weak, CCCP
Score: −1.008499
PASCAL+Weak, SPL
Score: 0.908217
Baseline
PASCAL(Object)
Score: −0.994624
PASCAL, CCCP
Score: 0.825880
PASCAL, SPL
Score: 2.130987
PASCAL(Object)
Score: −0.761251
PASCAL+Weak, CCCP
Score: −0.818254
PASCAL+Weak, SPL
Score: 0.954748

Bibliography

[1] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin. Motion-motif graphs. In *SIGGRAPH Symposium on Computer Animation*, 2008.

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

[3] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *KDD*, 2003.

[4] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[5] A. Gupta, A. Kembhavi, and L. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. In *PAMI*, 2009.

[6] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training for structural SVMs. *Machine Learning*, 77(1), 2009.

[7] L. Karlinsky, M. Dinerstein, and S. Ullman. Using body-anchored priors for identifying actions in single images. In *NIPS*, 2010.

[8] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems 23*. 2010.

[9] J. Niebles, H. Wang, and L. Fei-fei. A hierarchical model of shape and appearance for human action classification. In *IJCV*, 2008.

[10] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007.

[11] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.

[12] I. Tsochantaridis, T. Hofmann, Y. Altun, and T. Joachims. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[13] A. Vahdatpour, N. Amini, and M. Sarrafzadeh. Toward unsupervised activity discovery using multi-dimensional motif detection in time series. In *IJCAI*, 2009.

[14] Yang Wang and Greg Mori. Human action recognition by semilatent topic models. *PAMI*, 2009.

[15] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[16] B. Yao and S. Zhu. Learning deformable action templates from cluttered videos. In *ICCV*, 2009.

[17] C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.