DISCRIMINATIVE LATENT VARIABLE MODELS FOR VISUAL RECOGNITION

by

Weilong Yang

B.Eng., Southeast University (China), 2007

M.Sc., Simon Fraser University, 2010

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2012

APPROVAL

Name: Weilong Yang

Degree: Doctor of Philosophy

Title of Thesis: Discriminative Latent Variable Models for Visual Recognition

Examining Committee: Dr. Oliver Schulte, Associate Professor
Chair

Dr. Greg Mori, Senior Supervisor
Associate Professor

Dr. Mohamed Hefeeda, Supervisor
Associate Professor

Dr. George Toderici, Supervisor
Senior Software Engineer, Google Research

Dr. Ghassan Hamarneh, SFU Examiner
Associate Professor

Dr. Deva Ramanan, External Examiner
Associate Professor of Computer Science
University of California at Irvine

Date Approved: Sept., 26, 2012

Partial Copyright Licence

Abstract

Visual Recognition is a central problem in computer vision, and it has numerous potential applications in many different fields, such as robotics, human computer interaction, and entertainment.

In this dissertation, we propose two discriminative latent variable models for handling challenging visual recognition problems. In particular, we use latent variables to capture and model various prior knowledge in the training data. In the first model, we address the problem of recognizing human actions from still images. We jointly consider both poses and actions in a unified framework, and treat human poses as latent variables. The learning of this model follows the framework of latent SVM. Secondly, we propose another latent variable model to address the problem of automated tag learning on YouTube videos. In particular, we address the semantic variations (sub-tags) of the videos which have the same tag. In the model, each video is assumed to be associated with a sub-tag label, and we treat this sub-tag label as latent information. This model is trained using a latent learning framework based on LogitBoost, which jointly considers both the latent sub-tag label and the tag label.

Moreover, we propose a novel discriminative latent learning framework, *kernel latent SVM*, which combines the benefit of latent SVM and kernel methods. The framework of kernel latent SVM is general enough to be applied in many applications of visual recognition. It is also able to handle complex latent variables with interdependent structures using composite kernels.

Acknowledgments

I would like to express my gratitude to my senior supervisor, Prof. Greg Mori. I consider myself very lucky to have been his student. I am grateful not only for his constant encouragement and guidance throughout this research, but also for his patience, kindness and integrity.

Thanks to Prof. Yang Wang for his "unofficial" mentoring. Yang has introduced me to the area of discriminative learning with latent variables, and given me numerous help with my research. Yang is a great mentor, and he is also exceptionally knowledgeable and smart. It is a pleasure to have known him and worked with him.

Thanks to my manager at Google Research, Dr. George Toderici, for giving me the great opportunity to work on "Google-scale"computer vision problems. It was a wonderful experience to work with George and many other world-class engineers at Google.

Thanks to Prof. Mohamed Hefeeda, Prof. Ghassan Hamarneh, Prof. Deva Ramanan, and Prof. Oliver Schulte for serving on my thesis committee and providing constructive suggestions on my research.

Thanks to all my collaborators, labmates, and friends in SFU Vision and Media Lab. Especially thanks to Tian Lan, Guang-Tong Zhou, Arash Vahdat, Peng Peng, Kevin Cannons, and Nataliya Shapovalova for their support and suggestions on my research.

Thanks to the circle of my "Epic Bros", George Toderici, Annie Toderici, Dat Chu, Sean O'Malley, Tianhong Fang, Yaming Qiu, and Steve Urban, for the encouragement and bless.

This thesis is impossible without the many years of support and sacrifice from my family. Lastly, I want to thank my fiancée, Yujuan Xie, for her love and support.

Contents

Approval ii

Partial Copyright License iii

Abstract iv

Acknowledgments v

Contents vi

List of Tables ix

List of Figures x

1 Introduction 1

1.1 Dissertation Overview 4

2 Previous Work: Latent SVM 6

2.1 Formulation 6

2.2 Learning 8

2.2.1 Concave-Convex Procedure 8

2.2.2 Non-Convex Cutting Plane Training 9

2.2.3 Self-paced Learning 10

2.3 Related Models 11

2.3.1 Multiple Instance Learning 11

2.3.2 hCRF 12

2.4 Summary 13

3 Previous Work: Latent SVM in Visual Recognition **15**
3.1 Object Detection . . . 15
3.1.1 Latent Parts . . . 17
3.1.2 Latent Viewpoint . . . 19
3.2 Object Classification . . . 19
3.2.1 Latent Cropping . . . 20
3.2.2 Latent Attribute . . . 21
3.3 Human Activity Recognition . . . 22
3.3.1 Latent Patch . . . 23
3.3.2 Latent Interaction . . . 25
3.4 Other work . . . 27
3.5 Summary . . . 28

4 Action Recognition with Latent Pose **30**
4.1 Overview . . . 31
4.2 Pose Representation . . . 34
4.3 Model Formulation . . . 35
4.4 Learning and Inference . . . 41
4.4.1 Inference . . . 41
4.4.2 Learning . . . 41
4.4.3 Computational Complexity . . . 43
4.5 Experiments . . . 44
4.5.1 Visualization of Latent Pose . . . 47
4.6 Summary . . . 49

5 Kernel Latent SVM **50**
5.1 Preliminaries . . . 51
5.1.1 Binary Latent SVM . . . 51
5.1.2 Dual form with fixed h on positive examples . . . 52
5.2 Learning . . . 54
5.2.1 Optimization over $\{h_i\}$ . . . 55
5.2.2 Composite Kernels . . . 57
5.3 Experiments . . . 58
5.3.1 Object Classification with Latent Localization . . . 58

5.3.2 Object Classification with Latent Subcategory 60
5.3.3 Recognition of Object Interaction 63
5.4 Summary 64

6 Video Tagging with Latent Sub-tag **67**
6.1 Overview 67
6.2 Related Work 70
6.3 Model Formulation 71
6.4 Learning 72
6.4.1 Implementation Details 73
6.4.2 Computational Complexity 75
6.4.3 Initialization by Cowatch Features 76
6.5 Features 77
6.6 Experiments 79
6.6.1 Evaluation Measures 79
6.6.2 Results 81
6.7 Summary 83

7 Conclusion and Future Work **86**

Bibliography **88**

List of Tables

4.1 Results on the still image action dataset. We report both overall and mean per class accuracies due to the class imbalance. 46
4.2 Results on the Youtube dataset. We report both overall and mean per class accuracies due to the class imbalance. 46

5.1 Results on the mammal dataset. We show the mean/std of classification accuracies over five rounds of experiments. 60
5.2 Results on CIFAR10 Dataset. We show the mean/std of classification accuracies over five folds of experiments. Each fold uses a different batch of the training data. 62
5.3 Results on object interaction dataset. For the approaches using latent variables, we show the mean/std of classification accuracies over five folds of experiments. 64

6.1 The summary of the sub-tag labels for the 15 tags used in our experiments. Note that our algorithm cannot assign the semantic meaning label to each sub-tag. For the purpose of illustration, we manually assign a word label to each sub-tag by summarizing the video clusters obtained from cowatch initialization. For some sub-tags which are difficult to assign a meaningful word label, we simply use the cluster number to represent them. 80
6.2 Precision@1000 for tags “transformers” and “bike”. We compare our method to the baseline, and a method that runs the first step of our learning procedure once. 83

List of Figures

1.1 Examples of visual recognition tasks. The goal of object detection is to localize the instances of a particular object in the image, while object classification is only required to predict whether an instance of an object is present in the image. The goal of human action recognition is to identify the actions performed by the person in the image or the video. 2
1.2 Illustration of different levels of human supervision involved in the training of a car classifier. 3

3.1 Figure (a) is from [86]. Figure (b) and (c) are from PASCAL VOC Challenges [21]. 16
3.2 The red bounding boxes illustrate the detection results of a deformable part model trained for person. The small blue bounding boxes show the latent part locations obtained by inference (Eq. 2.5). This model consists of a root template (a), and a set of part templates (b). (c) visualizes a spatial model for the location of each part relative to the root. These figures are from [26]. 18
3.3 The visualization of latent crop examples. These figures are from [33]. 21
3.4 Visualization of an attribute relation graph. An edge on the graph represents a strong co-occurrence relationship. This figure is from [90]. 23
3.5 (a) illustrates the high-level motivation of the model in [91] which is to represent a human action by a large scale template feature (e.g. optical flow feature) and a set of local "parts". (b) is the graphical illustration of the model used in [91]. (c) visualizes the latent patches learned during training. Patches are colored based on their part labels. These figures are from [91]. . . 24
3.6 Graphical illustrations of the model used in [45]. The dashed lines indicate that the structure of latent variables is latent. These figures are from [45]. . . 26

4.1 Illustration of our proposed approach. Our goal is to infer the action label of a still image. We treat the pose of the person in the image as "latent variables" in our system. The "pose" is learned in a way that is directly tied to action classification. 32
4.2 Difference between previous work and ours. (Top) Previous work typically approaches pose estimation and action recognition as two separate problems, and uses the output of the former as the input to the latter. (Bottom) We treat pose estimation and action recognition as an single problem, and learn everything in an integrated framework. 34
4.3 Visualization of the poselets for a walking image. Ground-truth skeleton is overlayed on image. Examples of poselets for each part are shown. 36
4.4 Sample images of the still image action dataset [37], and the ground truth pose annotation. The locations of 14 joints have been annotated on each action image. (a) Running; (b) Walking; (c) Playing Golf; (d) Sitting; (e) Dancing. 36
4.5 Examples of poselets for each part from the running action. Each row corresponds to one poselet. The last column is the visualization of the filters for each poselet learned from SVM + HOG. 37
4.6 The four part star structured model. We divide the pose into four parts: legs, left-arm, right-arm, and upper-body. 38
4.7 Confusion matrices of the classification results on the still image action dataset: (a) baseline (b) our approach. Horizontal rows are ground truths, and vertical columns are predictions. 45
4.8 Confusion matrix of the classification results of our approach on the Youtube dataset. Horizontal rows are ground truths, and vertical columns are predictions. 46
4.9 Example visualizations of the latent poses on test images. For each action, we manually select some good estimation examples and bad examples. The action for each row (from top) is running, walking, palying golf, sitting and dancing respectively. 48

5.1 Visualization of how the latent variable (i.e. object location) changes during the learning. The red bounding box corresponds to the initial object location. The blue bounding box corresponds to the object location after the learning. 60
5.2 Visualization of some testing examples from the "bird" (a) and "boat" (b) categories. Each row corresponds to a subcategory. We can see that visually similar images are grouped into the same subcategory. 62
5.3 Visualization of how latent variables (i.e. object locations) change during the learning. The top image is from the "person riding a bicycle" category, and the bottom image is from the "person next to a car" category. Yellow bounding boxes corresponds to the initial object locations. The blue bounding boxes correspond to the object locations after the learning. 65

6.1 For the rather specific tag "transformers", the videos of this tag have large variations in video types and contexts. Those videos can be roughly grouped as video games, two different types of animations, toys, and movies. We use the notation "sub-tags" to refer to those semantic variants and treat them as latent information in our learning framework. 68
6.2 The sub-tag initialization results for the "bike" tag. In total, we generate four sub-tags: "mountain bike", "falling from bike", "pocket bike", and "motorbike". Note that our algorithm only cluster the candidate videos into four clusters and we manually assign a meaningful word label to each cluster. . . 77
6.3 Precision at K curves for both baseline and our method on the 50 million YouTube video dataset. We incremental increase K from 100 to 1000. 82
6.4 Visualizations of the sub-tag labels of the testing videos for tag "transformers". 84
6.5 Visualizations of the sub-tag labels of the testing videos for tag "bike". . . . 84

Chapter 1

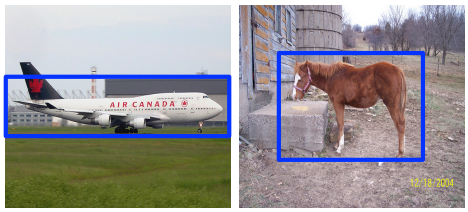
Introduction

Recognition is a central problem in computer vision, and its goal is to learn visual categories and then recognize the new instances from those categories. For example, visual recognition seeks the answers to questions such as "Is this a car?" or "What is this object?". Typical visual recognition tasks include image classification, object detection and classification, human action recognition, etc. Fig. 1.1 illustrates a few examples of visual recognition tasks. A working visual recognition system can have numerous applications in many different fields. For example, face detection has been integrated into many digital cameras to enhance the autofocus. As one of the dominant technologies in biometrics, face recognition has been applied not only in many security systems, but also in the digital photo organizers such as Picasa[1] and iPhoto[2]. The technology of human pose estimation using Kinect sensors [71] is another good example, which achieves a great success in human computer interaction and turns the concept of the controller-free game console into reality. Besides, visual recognition algorithms also play an important role in many image search and video retrieval systems.

The success of visual recognition systems builds on advanced techniques in the machine learning literature, such as AdaBoost, Support Vector Machine (SVM), and Random Forest. Most of these learning approaches for addressing the recognition problems follow the same pipeline which consists of two major steps. First, we need to collect a set of training examples (e.g. images or videos) of given categories. Depending on the details of different applications, each training example needs to be associated with a certain type of label. The second step

[1] http://picasa.google.com

[2] http://www.apple.com/ilife/iphoto



(a) Object Detection

(b) Object Classification (c) Human Action Recognition

Figure 1.1: Examples of visual recognition tasks. The goal of object detection is to localize the instances of a particular object in the image, while object classification is only required to predict whether an instance of an object is present in the image. The goal of human action recognition is to identify the actions performed by the person in the image or the video.

is to learn a model from the training set that can make predictions for the category labels of new examples. The first step of the above pipeline is also known as a process of preparing the training set, which requires manually labeling the training images/videos. For a specific task, e.g. training a car classifier, it may involve different levels of human supervision. For example, as shown in the left image of Fig. 1.2, in a weakly supervised setting, the car image is associated with a category label "car". This labeling is rather "weak" since it lacks a lot of useful information such as the exact location of the car, the viewpoint (e.g. side-view or front-view), the color or the shape of the car. Whereas "strong" supervision would mark the bounding box or the contour of the car, and even provide a list of descriptive keywords (i.e. attributes) about the image. While the strong labels tend to be more informative and less ambiguous, they are also very expensive to obtain, especially when we have thousands of images. Moreover, developing a system which can combine different types of labeling and maximize the benefits of such strong supervision is also challenging. Therefore, when designing a method for visual recognition, one must consider the trade-offs between the cost of manual labeling and the advantages that it can bring to the learning process.

Thanks to powerful image search engines from Google and Microsoft, a large scale image

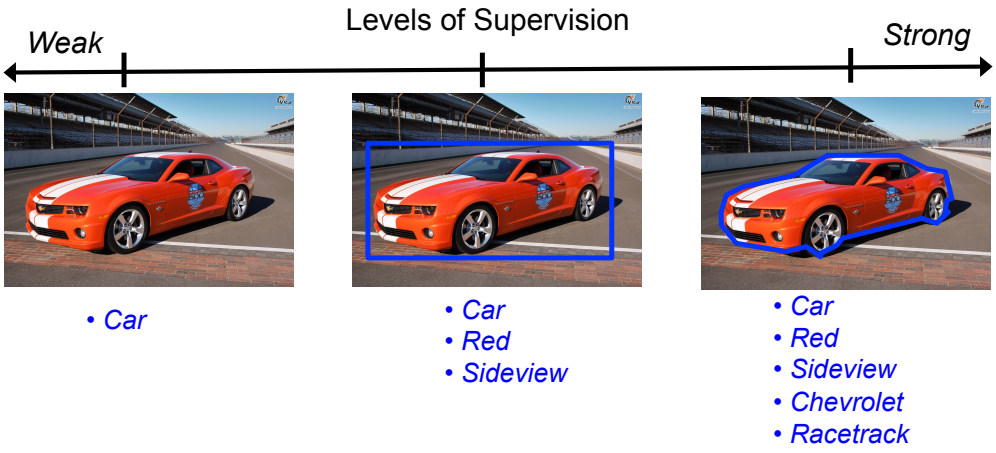


Figure 1.2: Illustration of different levels of human supervision involved in the training of a car classifier.

dataset can be easily collected using keyword search on the Web. However, even after the pruning process by the dataset creators or Amazon Mechanical Turk[3], there are still significant intra-category variations and many noisy samples. Typical examples of such datasets are ImageNet [16] and 80 Million TinyImages [78], both of which consist of millions of images and thousands of categories. However, the images are only *weakly labeled* by the category name without further detailed annotations.

In the literature, one possible approach for addressing weakly labeled data is to introduce *latent variables* into the training, and treat the missing labels as latent variables. Latent variables refer to the variable that are not directly observed but can be inferred from other observed variables. Latent variables have been widely used in many generative probabilistic models such as latent topic models and hidden Markov models. The constellation model [28] is one of the successful generative models using latent variables for addressing the problem of object classification. However, latent variables are less explored in discriminative models. Note that it is still a debatable topic in the machine learning community whether generative or discriminative models are superior. But in practice, discriminative approaches tend to produce better performance for most recognition problems. There is some recent work which explores the use of latent variables in discriminative models. Quattoni et al. [62] include latent variables in conditional random fields and propose the hidden Conditional

[3]https://www.mturk.com/mturk/welcome

Random Fields (hCRF). Another representative work is the latent SVM which extends the SVM framework for handling latent variables. It is proposed by Felzenszwalb et al. [26] for training a deformable part model (DPM) and achieves excellent performance in object detection.

In this dissertation, we mainly follow the framework of latent SVM, and focus on the learning of discriminative latent variable models for a variety of visual recognition problems. We not only adopt latent variable models for dealing with weakly labeled data, but also generalize them to address the problems of *recognition with auxiliary labels*. In contrast to weakly labeled data, the training data in those problems is associated with certain additional information. For example, in Chapter 4, we consider the task of human action recognition from still images, and each training image is associated with additional human pose labeling. In this case, the main task is human action recognition, and human poses are considered as auxiliary labels.

The major contributions of this dissertation can be briefly summarized as two discriminative latent variable models and one general latent learning framework: 1) We propose a latent variable model for the problem of action recognition from still images that jointly learns the actions and poses in a unified framework; 2) We propose the *kernel latent SVM* framework which combines the benefits of latent SVM and kernel methods; 3) We propose another latent variable model for addressing the semantic variations of YouTube videos, and we evaluate this model on a very large-scale dataset consisting of over *50 million* YouTube videos.

1.1 Dissertation Overview

The rest of this dissertation is organized as follows:

Chapter 2 reviews one of the most representative discriminative latent learning frameworks – latent SVM [26]. In the literature, latent SVM has been used to address a variety of challenging problems in visual recognition. The work presented in this dissertation is also mainly based on the latent SVM. In Chapter 2, we first introduce the general formulations of latent SVM, then review its two most related models, hidden Conditional Random Fields (hCRF) and MI-SVM in Multiple Instance Learning (MIL).

Chapter 3 reviews a few representative approaches using latent SVM. We mainly cover three sub-fields of visual recognition in this chapter: object detection, object classification,

and human activity recognition.

Chapter 4 proposes a discriminative latent variable model for addressing the problem of human action recognition from still images. In this model, we treat the pose of the person in the image as latent variables that will help with recognition. Different from other work that learns separate systems for pose estimation and action recognition, then combines them in an ad-hoc fashion, our model is trained in an integrated fashion that jointly considers poses and actions in a unified framework using latent SVM. This work has been published in [95] at IEEE Conference in Computer Vision and Pattern Recognition (CVPR) 2010.

Chapter 5 presents *kernel latent SVM* – a new learning framework that combines the benefits of latent SVM and kernel methods. We develop an iterative training algorithm to learn the model parameters, and demonstrate the effectiveness of kernel latent SVM using three different applications in visual recognition. Our framework is very general and it can be applied to solve a wide range of applications in computer vision. This work has been accepted to The Neural Information Processing Systems Conference (NIPS) 2012 [96].

Chapter 6 focuses on the problem of content-based automated tag learning on YouTube videos. We propose another latent variable model for addressing the semantic variations (sub-tags) of the videos which have the same tag. In this model, each video is assumed to be associated with a sub-tag label, and we treat this sub-tag label as latent information. This model is trained by a novel latent learning framework based on LogitBoost, which jointly considers both the latent sub-tag label and the tag label. We use cowatch information to initialize the learning process. This work has been published in [94] at IEEE Conference in Computer Vision and Pattern Recognition (CVPR) 2011.

Chapter 7 concludes this dissertation and discusses the future work.

Chapter 2

Previous Work: Latent SVM

Latent Support Vector Machine (*latent SVM*) is one of the most representative discriminative latent learning frameworks. It inherits the advantages of max-margin learning from SVM, and includes latent variables in the learning process. Due to its elegance and flexibility, latent SVM has been successfully applied to many visual recognition problems, and latent variables are defined in a variety of ways. Before reviewing these applications in Chapter 3, we first introduce the formulation of latent SVM in Section 2.1. We consider structured latent variables and assume there are certain dependencies among the latent variables of a training example. As in other learning algorithms with latent variables, latent SVM is a non-convex problem. We review three commonly used training algorithms of latent SVM in Section 2.2. Lastly, we compare latent SVM with other discriminative latent learning frameworks such as hCRF and MI-SVM in Section 2.3.

2.1 Formulation

We assume a data instance is in the form of $(\mathbf{x}, \mathbf{h}, y)$, where $\mathbf{x}$ is the observed variable and y is the class label. $\mathbf{h}$ is the latent variable that is associated with each instance $\mathbf{x}$. For example, say we want to learn a "car" model from a set of positive images containing cars and a set of negative images without cars. We know there is a car somewhere in a positive image, but we do not know its exact location. In this case, $\mathbf{h}$ can be used to represent the unobserved location of the car in the image.

We consider structured latent variables and define $\mathbf{h}$ as a vector, i.e. $\mathbf{h} = (z_1, z_2, ..)$, where z_i is the i-th component of $\mathbf{h}$ and takes its value from a discrete set $\mathcal{H}$ of possible

labels (i.e. $z_i \in \mathcal{H}$). For structured latent variables, it is assumed that there are certain dependencies between some pairs of (z_i, z_j). We can use an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to capture the structure of latent variables, where a vertex $i \in \mathcal{V}$ corresponds to the label z_i, and an edge $(i, j) \in \mathcal{E}$ corresponds to the dependency between z_i and z_j. In the original latent SVM work [26], only binary classification is considered, i.e. $y \in \{+1, -1\}$. Wang and Mori [91] extend latent SVM to multi-class classification. Yu and Joachims [97] further extend it to a more general formulation with structured output. In this chapter, we review the formulation of latent SVM under the multi-class classification setting, i.e. $y \in \mathcal{Y}$ and $\mathcal{Y}$ is a set of discrete class labels. As in other discriminative learning frameworks, we are interested in learning a discriminative function $f_{\mathbf{w}} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ over an instance $\mathbf{x}$ and its class label y, where $f_{\mathbf{w}}$ is parameterized by $\mathbf{w}$. During testing, we can predict the class label y^* of an input instance as: $y^* = \arg\max_{y \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{x}, y)$. The scoring function $f_{\mathbf{w}}(\mathbf{x}, y)$ is assumed to take the following form:

$$f_{\mathbf{w}}(\mathbf{x}, y) = \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y), \tag{2.1}$$

where $\Phi(\mathbf{x}, \mathbf{h}, y)$ is a feature vector depending on the instance $\mathbf{x}$, the latent variable $\mathbf{h}$, and the class label y. The maximizing over $\mathbf{h}$ in Eq. 2.1 has a very appealing interpretation. For example, in the above "car model" example, the optimal $\mathbf{h}^*$ for the image $\mathbf{x}$ (i.e. $\mathbf{h}^* = \arg\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$) will be the exact location of the car in the image under an ideal situation. One alternative way to define the scoring function $f_{\mathbf{w}}(\mathbf{x}, y)$ is summing over $\mathbf{h}$ instead of maximizing over $\mathbf{h}$. Summing over $\mathbf{h}$ corresponds the marginalization rule in probabilistic models. We will compare these two approaches in Section 2.3.2.

Depending on different applications, the model formulation $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ can be defined in different ways. However, in general, given that latent variable $\mathbf{h}$ is constrained by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ usually consists of unary terms and pairwise terms. Here, we give an example form of $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ as follows:

$$\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y) = \sum_{j \in \mathcal{V}} w_1^\top \phi_1(\mathbf{x}, z_j, y) + \sum_{(i,j) \in \mathcal{E}} w_2^\top \phi_2(z_i, z_j, y) \tag{2.2}$$

We emphasize that the form of $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ is not limited to Eq. 2.2. It can be very flexible as long as there exists an efficient inference algorithm for Eq. 2.1.

2.2 Learning

Given a set of N training examples $\{\mathbf{x}_i, y_i\}_{i=1}^N$, our goal is to learn the model parameter $\mathbf{w}$ that can be used to assign the class label y to an unseen instance $\mathbf{x}$. Note again that each training instance is associated with the latent variable $\mathbf{h}_i$. Similar to classical SVMs, the optimization in latent SVM is a quadratic program [91] as follows:

$$\begin{aligned} &\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ &\text{s.t.} \quad f_{\mathbf{w}}(\mathbf{x}_i, y) - f_{\mathbf{w}}(\mathbf{x}_i, y_i) \leq \xi_i - 1, \quad \forall i, \quad \forall y \neq y_i \\ &\qquad \xi_i \geq 0, \quad \forall i, \end{aligned} \tag{2.3}$$

where $\{\xi_i\}$ are the slack variables for handling soft margins, and C is the trade-off parameter. By replacing $f_{\mathbf{w}}(\mathbf{x}, y)$ with $\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$, we obtain the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ &\text{s.t.} \quad \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}, y) - \max_{\mathbf{h}'} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}', y_i) \leq \xi_i - \Delta(y, y_i), \quad \forall i, \quad \forall y \\ &\qquad \xi_i \geq 0, \quad \forall i \\ &\text{where,} \quad \Delta(y, y_i) = \begin{cases} 1 & \text{if } y \neq y_i \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.4}$$

In the formulation, since the maximum of a set of convex functions is convex, the first component in the constraint (i.e. $\max_{\mathbf{h}} \mathbf{w}^\top \Phi(x_i, \mathbf{h}, y)$) is convex w.r.t. $\mathbf{w}$. However, the second component (i.e. $-\max_{\mathbf{h}'} \mathbf{w}^\top \Phi(x_i, \mathbf{h}', y_i)$) is concave because of the negative sign. Therefore, the learning problem presented in Eq. 2.4 is a non-convex optimization problem. Unlike other non-convex problems, latent SVM has a nice property of *semi-convexity*. If we fix the latent variable $\mathbf{h}'$ to a single choice for the instance pair of $(\mathbf{x}_i, y_i)$, the optimization problem of Eq. 2.4 becomes convex. In Section 2.2.1 and Section 2.2.2, we will introduce two general approaches of solving the optimization problem of Eq. 2.4. Although neither of these two approaches would guarantee a global minimum or a good local minimum, both of them ensure convergence.

2.2.1 Concave-Convex Procedure

The *semi-convexity* property of the latent SVM allows an intuitive iterative algorithm that alternates between inferring latent variable $\mathbf{h}'$ for the instance of $(\mathbf{x}_i, y_i)$ and optimizing the

model parameter $\mathbf{w}$. These two major steps are summarized as follows:

1. Holding $\mathbf{w}, \xi$ fixed, compute the optimal latent variable $\mathbf{h}'$ for every instance pair of $(\mathbf{x}_i, y_i)$:

$$\mathbf{h}_{i,y_i} = \arg\max_{\mathbf{h}'} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}', y_i) \tag{2.5}$$

2. Holding $\mathbf{h}_{i,y_i}$ fixed, compute the optimal w, ξ as follows:

$$\begin{aligned} &\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ &\text{s.t.} \quad \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}, y) - \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}_{i,y_i}, y_i) \leq \xi_i - \Delta(y, y_i), \quad \forall i, \quad \forall y \\ &\qquad \xi_i \geq 0, \quad \forall i \end{aligned} \tag{2.6}$$

If $\mathbf{h}$ takes its value from a discrete set, Eq. 2.5 can be solved by simply enumerating all possible values of $\mathbf{h}$. However, if latent variables are structured (i.e. $\mathbf{h} = (z_1, z_2, ..., z_M)$), the complexity of this naive enumerating approach is $O(M^K)$ when z_i takes its value from a discrete set of K possible choices (i.e. $z_i \in \mathcal{H}$ and $|\mathcal{H}| = K$). This is obviously too expensive. Fortunately, in most of approaches using latent SVM, the structured latent variables are assumed to form a tree and then there exist efficient inference algorithms for solving Eq. 2.5, such as dynamic programming.

As we discussed earlier, if holding $\mathbf{h}_{i,y_i}$ fixed, the optimization problem in Eq. 2.6 is a convex quadratic program. Since it is not in a standard form of multi-class SVMs, we cannot solve it by off-the-shelf SVM solvers (e.g. LibSVM [11] or SVMLight [38]). Wang and Mori [91] develop a solver for Eq. 2.6 based on the cutting-plane method and decomposed dual optimization.

Interestingly, Yu and Joachims [97] show that this iterative algorithm is essentially a concave-convex procedure algorithm. It is shown in [98] that the CCCP algorithm is guaranteed to decrease the value of the objective function in each iteration and converge to a local minimum. For simplicity, we will refer to this iterative algorithm as CCCP in the following discussion.

2.2.2 Non-Convex Cutting Plane Training

The learning problem in Eq. 2.4 can also be solved by a non-convex cutting plane algorithm in [19], which is an extension of the popular convex cutting plane algorithm [39] for learning

structural SVM [1]. Due to its ease of use, this algorithm has been adopted in several different applications of latent SVM [95, 90].

First, we write Eq. 2.4 as the following unconstrained formulation:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i R_i(\mathbf{w}) \quad \text{where,}$$

$$R_i(\mathbf{w}) = \max_y \left(\Delta(y, y_i) + \max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}, y)\right) - \max_{\mathbf{h}'} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}', y_i) \tag{2.7}$$

The learning algorithm in [19] aims to iteratively build an increasingly accurate piecewise quadratic approximation of Eq. 2.4 based on the "sub-gradient"[1] $\partial_{\mathbf{w}}\left(\sum_i R_i(\mathbf{w})\right)$ [19]. The key issue is how to compute the sub-gradient $\partial_{\mathbf{w}}\left(\sum_i R_i(\mathbf{w})\right)$. Since

$$\partial_{\mathbf{w}}\left(\sum_i R_i(\mathbf{w})\right) = \sum_i \partial_{\mathbf{w}} R_i(\mathbf{w}), \tag{2.8}$$

all we need to do is to figure out how to compute $\partial_{\mathbf{w}} R_i(\mathbf{w})$. We define:

$$(y^*, \mathbf{h}^*) = \arg\max_{y,\mathbf{h}} \Delta(y, y_i) + \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}, y) \tag{2.9}$$

$$\widehat{\mathbf{h}}' = \arg\max_{\mathbf{h}'} \mathbf{w}^T \Phi(\mathbf{x}_i, \mathbf{h}', y_i) \tag{2.10}$$

It can be shown that $\partial_{\mathbf{w}} R_i(\mathbf{w})$ can be calculated as

$$\partial_{\mathbf{w}} R_i(\mathbf{w}) = \Phi(\mathbf{x}_i, \mathbf{h}^*, y^*) - \Phi(\mathbf{x}_i, \widehat{\mathbf{h}}', y_i) \tag{2.11}$$

The computing of the subgradient $\partial_{\mathbf{w}} R_i(\mathbf{w})$ involves solving two inference problems in Eq. 2.9 and Eq. 2.10. This is similar to the CCCP learning algorithm (Section 2.2.1) which also requires the solving of inference problems over latent variable $\mathbf{h}$.

2.2.3 Self-paced Learning

The learning of latent SVM is sensitive to initialization and often gets stuck in a bad local optimum. This is a common problem for most non-convex optimization algorithms. In practice, to avoid bad local optima, one needs to run the learning algorithm many times with different initializations, and then choose the model parameters which produce the best performance on a validation set. However, this scheme is computationally expensive.

[1] We follow the notation of "sub-gradient" in [19]. It is used to compute the cutting plane which is a "local" lower-bound of the non-convex objective function.

To addresses these issues, Kumar et al. [42] propose a self-paced learning algorithm for the training of latent SVM. This algorithm is inspired by human learning. For example, children are first taught with simple and easy concepts, and then the hard ones. Similarly, in self-paced learning, training examples are grouped into "easy" examples and "hard" examples. An example is an "easy" one if its correct output can be predicted easily, i.e. it has a higher decision score (above some threshold). Self-paced learning of latent SVM can be considered as a variant of the CCCP learning algorithm in Section 2.2.1. In self-paced learning, the step 2 (Eq. 2.6) is slightly modified and is only performed on the "easy" examples. During the iterations, by adjusting the threshold, more examples will be considered as "easy" and introduced into the training until the entire training set is used. Kumar et al. [42] have demonstrated that this self-paced learning algorithm outperforms the CCCP algorithm on several visual recognition applications. Similar idea has been adopted in [99] which adds training data incrementally at each iteration.

2.3 Related Models

There are many approaches in the literature which are related to the latent SVM. For example, there is a line of work on using latent topic models for visual recognition based on "bag-of-words" representation, such as pLSA [34], LDA [6], etc. The latent topic models are generative and probabilistic so they are very different to the latent SVM. In this section, instead of going through all the learning frameworks with latent variables, we focus on the two most related work to latent SVM: Multiple Instance Learning (MIL) [2] and hidden Conditional Random Fields (hCRF) [62].

2.3.1 Multiple Instance Learning

In the setting of Multiple Instance Learning (MIL), the training examples come in as "bags". There is at least one example that is positive in a positive bag, while in a negative bag all examples must be negative. It has many applications in visual recognition, such as object detection [84], object tracking [4], etc. Similar to latent SVM, an appealing characteristic of MIL is that the "multiple instance" setting fits with most of visual recognition problems using weakly labeled data. For example, the "car model" example we used in Section 2.1 can also be interpreted as a MIL problem. A positive "car" image is a positive bag which consists of a number of image regions, among which there is at least one region that contains

a car. Note again that the ground-truth locations of the cars are not provided on the training images. A negative "car" image is a negative bag which does not contain any car region.

A number of researchers have modified classical discriminative learning approaches to perform MIL. Viola et al. [84] propose MILBoost which is an extension of AnyBoost framework for MIL problem. Andrews et al. [2] formulate MIL as a max-margin problem and propose the MI-SVM. In MI-SVM, the decision score for a positive bag is defined as the maximum score of the instances in this bag, and the goal of MI-SVM is to maximize the margin between bags instead of instances. The formulation of MI-SVM is defined as follows:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_I \xi_I \\ &\text{s.t. } \forall I,\ Y_I \max_{i\in I}(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_I,\ \xi_I \geq 0 \end{aligned} \tag{2.12}$$

where Y_I denotes the label for the I-th bag, and $\mathbf{x}_i$ denotes the i-th instance in the bag I. Note that this formulation of MI-SVM is for binary classification so that $Y_I \in \{+1, -1\}$. It is interesting to notice that Eq. 2.12 is very similar to the original latent SVM formulation [26]. Indeed, as noted in [26], the latent SVM formulation in [26] is equivalent to the MI-SVM. However, we believe latent variables in latent SVM are much easier to be defined than the "bag-instance" setting in MIL. Moreover, latent variables can be very flexible and have richer representations. For example, in visual recognition, latent variables are often structured (e.g. a tree structure), which are difficult to be formulated in the MIL setting.

2.3.2 hCRF

Quattoni et al. [62] propose the hidden Conditional Random Fields (hCRF) for object recognition. hCRF is a discriminative probabilistic model and it extends the Conditional Random Fields (CRF) model by incorporating hidden variables. Note that the term "hidden variable" in hCRF is equivalent to the "latent variable" in latent SVM. Both of them refer to the variables which are not observed during training. As a discriminative model, hCRF models the conditional distribution $p(y|\mathbf{x})$ directly which can be expressed as follows:

$$p(y|\mathbf{x};\theta) = \sum_{\mathbf{h}} p(y,\mathbf{h}|\mathbf{x};\theta) = \frac{\sum_{\mathbf{h}} \exp(\Phi(y,\mathbf{h},\mathbf{x};\theta))}{\sum_{y',\mathbf{h}} \exp(\Phi(y',\mathbf{h},\mathbf{x};\theta))} \tag{2.13}$$

where θ are the model parameters, and $\Phi(y,\mathbf{h},\mathbf{x};\theta)$ is the potential function parameterized by θ. The potential function in hCRF is defined in a similar way to the one in latent SVM,

and it consists of both unary terms and binary terms if the latent variable $\mathbf{h}$ is constrained by a graph $\mathcal{E}$.

The model parameters θ can be learned by minimizing the following objective function, which is equivalent to maximizing the conditional log likelihood:

$$L(\theta) = -\sum_i \log p(y_i|\mathbf{x}_i;\theta) + \frac{||\theta||^2}{2\sigma^2} \tag{2.14}$$

$$= -\sum_i \log \sum_{\mathbf{h}} \exp(\Phi(y_i, \mathbf{h}, \mathbf{x}_i;\theta)) + \sum_i \log \sum_{y',\mathbf{h}} \exp(\Phi(y', \mathbf{h}, \mathbf{x}_i;\theta)) + \frac{||\theta||^2}{2\sigma^2}, \tag{2.15}$$

where $\frac{||\theta||^2}{2\sigma^2}$ is the log of the Gaussian prior, i.e. $p(\theta) \backsim \exp(\frac{1}{2\sigma^2}||\theta||^2)$. The first term in Eq. 2.15 is concave and the other two terms are convex. So, similar to latent SVM, the optimization problem in Eq. 2.15 is not convex. In [62], Quattoni et al. use the gradient descent approach to find its locally optimal solutions.

Both hCRF and latent SVM are discriminative models with latent variables. As noted in [91], we can think of multi-class SVM as a "max-margin" version of hCRF. Wang and Mori discuss the connections between hCRF and latent SVM in [91]. First, the training criteria of latent SVM and hCRF are different. The goal of latent SVM is to maximize the margin but hCRF is to maximize the conditional log likelihood. Besides, the most major difference between these two approaches lies in their different ways of incorporating the latent variables. In hCRF, due to its probabilistic nature, the rule of summing over $\mathbf{h}$ is adopted (as shown in Eq. 2.13). In contrast, latent SVM requires the maximizing over $\mathbf{h}$ (as shown in Eq. 2.1). It is discussed in [91] that in practice, maximizing over $\mathbf{h}$ is more suitable for visual recognition problems. The intuition behind the summing over $\mathbf{h}$ (Eq. 2.13) is that the learning algorithm would push the probability of incorrect labeling of $\mathbf{h}$ close to zeros and thus the probability of correct labeling of $\mathbf{h}$ will contribute more to $p(y|\mathbf{x};\theta)$. However, this is rather an ideal situation. In practice, for an example $\mathbf{x}$, the choices of its latent variables can be exponentially large, among which only a small number of choices are "correct" ones. Then, the summing over $p(y, \mathbf{h}|\mathbf{x};\theta)$ may still be dominated by the incorrect $\mathbf{h}$'s, which is obviously not desirable.

2.4 Summary

In this chapter, we have provided an overview of latent SVM. Here, we summarize a few key characteristics of latent SVM as follows:

- Latent SVM is not convex. A proper initialization is necessary for achieving good performance. Alternatively, one can use some learning tricks, e.g. self-paced learning.
- Latent SVM is essentially a variant formulation of MI-SVM in multiple instance learning (MIL). However, defining latent variables for most computer vision problems using weakly labeled data is much more natural than converting them to MIL problems.
- A multi-class latent SVM can be considered as a max-margin version of hCRF. The maximizing over $\mathbf{h}$ in latent SVM is more suitable for most visual recognition applications than summing over $\mathbf{h}$.

Chapter 3

Previous Work: Latent SVM in Visual Recognition

In this chapter, we review the approaches in the literature which adopt the latent SVM framework for addressing challenging problems in visual recognition. Although these approaches follow the same learning framework, they can be differentiated by 1) the semantic meaning of latent variables; and 2) the model formulation (i.e. $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$). We will focus on these two aspects and review a few representative approaches using latent SVM. We mainly cover three sub-fields of visual recognition: objection detection (Section 3.1), object classification (Section 3.2), and human activity recognition (Section 3.3). We emphasize that latent SVM is not limited to these three applications, and it has been used in a variety of applications, which will be briefly summarized in Section 3.4.

3.1 Object Detection

Object detection is one of the most challenging problems in computer vision. The goal of object detection is to answer the question of "where are the instances of a particular object in the image (if any)?". Object detection has a wide range of applications in the real world. One successful example is face detection which can be considered as a special case of object detection. Although to a certain extent, face detection is a "solved" problem, the detection of generic objects (e.g. person, car, bicycle, dog, etc.) remains challenging. In this section, we first summarize the challenges of detecting objects. We emphasize that most of these

(a) Shape variations (b) Suboptimal labeling (c) Viewpoint variation

Figure 3.1: Figure (a) is from [86]. Figure (b) and (c) are from PASCAL VOC Challenges [21].

challenges also exist in other applications, such as object classification and human action recognition.

Shape variation: One of the major challenges of object detection is the significant variations in appearance, which include not only illumination and viewpoint variations, but also non-rigid deformations in the shape of the object. For example, people in the pictures are often in very different postures (Fig. 3.1(a)). The linear models (e.g. HOG feature + linear SVM [14]) cannot effectively capture these variations, and they usually only work well when a person is in an upright posture. To handle the variations in the shape of objects, one popular approach is to model the deformable configuration of object parts using pictorial structures [27]. However, it requires the detailed labeling of object parts, which is not available for most object detection tasks.

Suboptimal labeling: In the setting of object detection, images are only *manually* labeled with bounding boxes which cover the objects of interest. Although these bounding boxes are treated as "ground-truth", they are usually suboptimal. For example, as shown in Fig. 3.1(b), the bounding boxes correctly locate the two persons in the image, but clearly the persons in the bounding boxes are not well aligned. It is believed that suboptimal labeling can result in inferior training, and an alignment preprocessing is often necessary. For example, in most face detection/recognition systems, it is a common strategy to first pre-process the face images and align the faces by eye positions.

Viewpoint variations: At present, the general goal of object detection is detecting objects from static 2D images. The projection process of an object from a 3D world to a 2D image plane often results in appearance variations. Compared with the shape variation, the variations caused by the viewpoints or the object poses are much more significant. For

example, as shown in Fig. 3.1(c), the front-view car looks very different from the side-view car. This type of variation can be easily addressed if we have the annotations of the viewpoints or the objects' poses in 3D. A successful example is the multi-view face detection system [50] which trains a separate detector for each specified head pose (e.g. frontal face, profile face, etc.). However, for the detection of generic objects, it is difficult to collect accurate annotations of viewpoints or objects' 3D poses.

So far we have summarized three major challenges of object detection. In fact, all these challenges are not independent from each other and they are rather coupled together. For example, the viewpoint variations would lead to more challenging shape variations or occlusions. Having more complete and elaborate annotations can greatly help us address these challenges, but again it will be difficult and expensive to collect such annotations. In the following sections, we will discuss the literature which adopts the latent SVM to deal with these challenges.

3.1.1 Latent Parts

Felzenszwalb et al. [26] propose a deformable part model (DPM) for object detection. This model achieves the top performance on the PASCAL VOC object detection competition [21] and it has been widely adopted. The DPM builds on the Histogram of Oriented Gradient (HOG) features [14]. Given a set of training images annotated with bounding boxes around each object instance, the training of DPM is a typical binary classification problem. For example, for training a "car" detector, the image regions containing the cars are the positive examples, while other image regions without cars are the negative examples. As we discussed earlier, the bounding boxes provided along with the training set are often not optimal. Therefore, the locations of the optimal bounding boxes are treated as latent variables. Moreover, to address the shape variations, a star-structured pictorial structure model is used in DPM. Unlike other approaches using the pictorial structure, the locations of object parts in DPM are considered as latent, and they will be implicitly inferred during learning.

More formally, the latent variable for an object is defined as $\mathbf{h} = (z_0, z_1..., z_n)$, where z_0 denotes the location of the optimal bounding box. $(z_1, ..., z_n)$ denote the locations of the n parts. It is assumed that latent variable $\mathbf{h}$ is constrained by a star structure where z_0 is the root. Note that here, for ease of understanding, we modify the notations in [26], and try to make them consistent with the ones in Chapter 2. The potential function $w^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ is

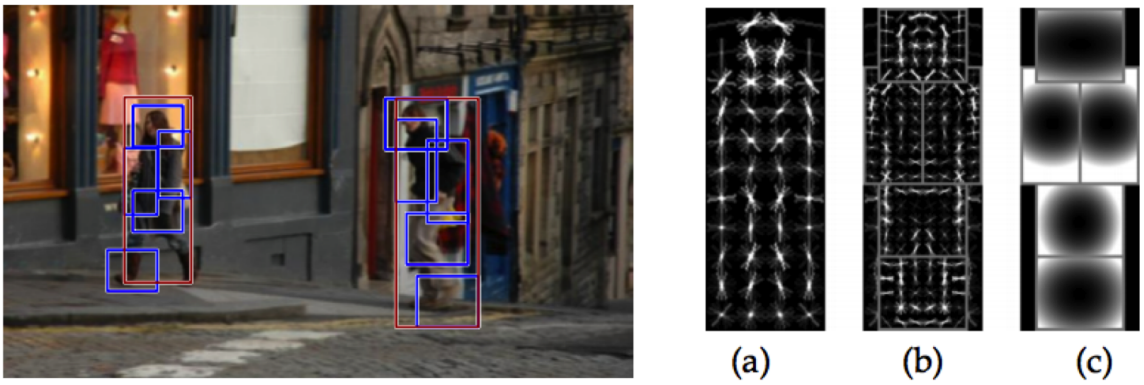


Figure 3.2: The red bounding boxes illustrate the detection results of a deformable part model trained for person. The small blue bounding boxes show the latent part locations obtained by inference (Eq. 2.5). This model consists of a root template (a), and a set of part templates (b). (c) visualizes a spatial model for the location of each part relative to the root. These figures are from [26].

defined as follows:

$$\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y) = w_0^\top \phi(\mathbf{x}, z_0) + \sum_{i=1}^{n} w_i^\top \phi(\mathbf{x}, z_i) + \sum_{i=1}^{n} d_i \phi_d(z_i, z_0) + b, \tag{3.1}$$

where b is the bias term. $\phi(\mathbf{x}, z_i)$ represents the features extracted from location z_i for the i-th part. $\phi_d(z_i, z_0)$ models the relative displacement of the i-th part with respect to the root z_0. The model parameters $\mathbf{w}$ are simply the concatenation of the parameters in all the factors, i.e. $\mathbf{w} = (w_0, w_1, ..., w_n, d_1, ..., d_n, b)$. The training of DPM is performed by latent SVM and it is solved by the CCCP algorithm described in Section 2.2.1. Because of the star structure, the inference of the latent variable $\mathbf{h}$ (Eq. 2.5) can be efficiently computed using dynamic programming and generalized distance transforms (min-convolutions). Example detection results and a part model trained for person detection are visualized in Fig. 3.2.

In DPM, the star structure of latent variables can be considered as a 2-layer structure including a root node and several part nodes. Zhu et al. [99] extend it to a 3-layer structure. In this "deep structure" model, the first layer is the root node (i.e. the optimal bounding box). The second layer consists of 9 child nodes (i.e. parts) in a 3×3 grid layout. Each node in the second layer is associated with 4 child nodes (i.e. parts) from the third layers. They demonstrate the performance of DPM can be improved by simply adding an extra layer in the structure of latent variables. To accelerate the training, an incremental concave-convex procedure (iCCP) is proposed in [99] which adds training data incrementally at each

iteration and thus reduces the training cost.

3.1.2 Latent Viewpoint

In [26], Felzenswalb et al. use mixture models to address the large viewpoint variations. In their experiments, each mixture model consists of only two components, and each component corresponds to a viewpoint. Gu and Ren [31] demonstrate that the mixture model representation in DPM is able to handle a large number of components (viewpoints). There is an interesting interpretation of mixture models by latent variables. In [31], each object instance is associated with a viewpoint label h, where $h \in \mathcal{H}$ and $\mathcal{H}$ is the set of all possible viewpoints. Note that different from the structured latent variables we introduced in Chapter 2, here the latent variable is a simple discrete value. The potential function $\mathbf{w}^\top \Phi(\mathbf{x}, h, y)$ is defined as follows:

$$\mathbf{w}^\top \Phi(\mathbf{x}, h, y) = \sum_{a \in \mathcal{H}} \mathbf{1}_a(h) \cdot w_a^\top \phi(\mathbf{x}, y) \tag{3.2}$$

where $\mathbf{1}_a(h)$ is an indicator that takes the value 1 if $h = a$, and 0 otherwise. $\phi(\mathbf{x}, y)$ represents the HOG feature extracted from the example $\mathbf{x}$. Similar to [26], Gu and Ren [31] choose the CCCP algorithm to train the latent SVM models. Note that the latent variable h in [31] is a discrete value, so the inference of the latent variable is performed by simply enumerating every choice in the set of $\mathcal{H}$.

In the setting of latent SVM, the viewpoints are considered as latent information. However, in the real world, it might be provided on training data (e.g. 3DObject dataset [66]). Gu and Ren [31] also extend the framework of latent SVM to address the following two scenarios: 1) h is observed (i.e. ground-truth viewpoint labeling is provided); 2) h is partially observed (i.e. a subset of training examples has ground-truth viewpoint labeling). It demonstrates the flexibility of latent SVM framework for handling the problem of recognizing with auxiliary labels (e.g. viewpoints).

3.2 Object Classification

The goal of object classification is to predict whether an instance of a particular object class exists in the image. It is interesting to notice that if object detection is solved, object classification is solved. For example, if we know there is a car in the right-upper corner

of the image, then this image definitely contains a car. However, given the challenges we list in Section 3.1, object detection is still a largely unsolved problem, so it is questionable whether the output of any object detection algorithm is reliable enough to be directly used for object classification. Because of the special relationship between *detection* and *classification*, object classification faces the same challenges as object detection. At present, most successful object classification methods are still "global" methods based on the bag-of-features (BOF) extracted from the whole image [21]. This is mainly because the BOF is robust to appearance variations and incorporates scene contextual information.

In this section, we will review two interesting systems for classifying objects with latent variables. We emphasize that the latent variables in these two systems are semantically meaningful. In [33], the locations of objects in the image are considered as latent variables, while in [90] latent variables are the object attributes. It has been experimentally demonstrated that the performance of object classification is improved by using latent variables.

3.2.1 Latent Cropping

As we discussed earlier, most of object classification approaches are "global" approaches based on the bag-of-feature representation and treat each image region equally important. However, consider the bicycle image in Fig. 3.3(a). The bicycle takes only a small portion of the image. Although the "road" in the image may provide a certain amount of contextual information, the "grass" seems irrelevant to the bicycle. Therefore, it is counterintuitive to include all image regions in the training of an object classification model and treat them equally important.

To address this issue, one possible approach is to first localize the most discriminative (or salient) region on the image, then perform object classification on this region instead of the whole image. However, this approach is rather difficult. For example, it is not clear what type of region would be discriminative enough for the purpose of object classification. There is no ground-truth labeling of these discriminative regions either. Instead, Bilen et al. [33] propose to treat the location of the most discriminative region on each image as a latent variable. This latent variable essentially specifies a cropping operation which determines a bounding box covering the most discriminative region on the image. The latent variable $\mathbf{h}$ is defined as the location of the discriminative region on the image, and the feature vector $\Phi(\mathbf{x}, \mathbf{h}, y)$ is the BOF extracted from the image region $\mathbf{h}$. During testing, the inference of $\mathbf{h}$ is performed by enumerating all possible regions on the image $\mathbf{x}$.

Figure 3.3: The visualization of latent crop examples. These figures are from [33].

Fig. 3.3 shows examples of the learned latent variables (i.e. the location of the most discriminative region). We can see that all discriminative regions cover the target objects and a bounding box can include more than one object.

3.2.2 Latent Attribute

The goal of object classification is also known as *object naming* (e.g. "Is this an image of a dog or cat?"). Recently, there is a trend in visual recognition which shifts the goal of recognition from naming to describing. In the task of "object describing", each image is not only assigned with an object class, but also a list of attributes. Object attributes are in line with human intuition about describing an object. Typical attributes can be color (e.g. "red"), shape (e.g. "round"), materials (e.g. "furry"), etc. There is an interesting question arising: "can attributes help recognition?". Farhadi et al. [24] propose a two-stage classification process by first training a number of attribute classifiers, then using their outputs to classify objects. They show that the learning of attributes can help categorize objects, especially when only a few training examples are available.

In the two-stage classification framework of [24], the learning of object attributes and classes are treated as two separate learning problems. One limitation of this approach is that the learning of object attributes is not directly tied to the end goal of object classification, and thus the output of the attribute prediction might not be reliable. Wang and Mori [90] introduce a discriminative model which jointly learns object attributes and classes. In particular, they characterize the problem as a *recognition problem with auxiliary labels*, and treat the attributes as auxiliary labels. The training data consists of both ground-truth object class labels and attribute labels. But testing images do not have the ground-truth attribute labels. The end goal of the learning system is object classification, so the loss functions are only defined over the class label y. This model is trained in the framework of

latent SVM. To ensure the consistency between training and testing, the auxiliary attribute labels on the training examples are treated as latent variables, so that the learning algorithm would not "overly trust" the attribute information which is missing during testing.

More formally, the attributes of an object are denoted as $\mathbf{h} = (z_1, z_2, ..., z_K)$, where $z_k \in \{0, 1\}$ indicates whether the k-th attribute presents in the image. For example, if the k-th attribute is "red", $z_k = 1$ means this object is "red", while $z_k = 0$ means it is not. Similar to other discriminative frameworks with latent variables, there are certain dependencies (e.g. co-occurrence) between some pairs of attributes (z_j, z_k), which are captured by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. This graph is named as "attribute relation graph", and an example graph is shown in Fig. 3.4. The $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ is defined as follows:

$$\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y) = w_y^\top \phi(\mathbf{x}) + \sum_{j \in \mathcal{V}} w_{z_j}^\top \varphi(\mathbf{x}) + \sum_{j \in \mathcal{V}} w_{y,z_j}^\top \omega(\mathbf{x}) + \sum_{(j,k) \in \mathcal{E}} w_{j,k}^\top \psi(z_j, z_k) + \sum_{j \in \mathcal{V}} \nu_{y,z_j} \quad (3.3)$$

Now we briefly describe each potential function in Eq. 3.3. First, $w_y^\top \phi(\mathbf{x})$ is a standard linear model for object classification without considering attributes. Both $\sum_{j \in \mathcal{V}} w_{z_j}^\top \varphi(\mathbf{x})$ and $\sum_{j \in \mathcal{V}} w_{y,z_j}^\top \omega(\mathbf{x})$ are linear models trained to predict the label (0 or 1) of j-th attribute for the image $\mathbf{x}$. The only difference is that the latter one is the class-specific attribute model and its parameters w_{y,z_j} can be considered as a template for the j-th attribute to take the label z_j if the object class is y. The third potential function $\sum_{(j,k) \in \mathcal{E}} w_{j,k}^\top \psi(z_j, z_k)$ measures the dependencies between the j-th and the k-th attributes, and ν_{y,z_j} captures the compatibility between object class y and the j-th attribute z_j. Although this model (Eq. 3.3) is rather complicated, it successfully integrates image features $\mathbf{x}$, the latent attribute labels $\mathbf{h}$, and the object class label y in a unified framework which outperforms the two-stage framework in [24].

This paper [90] shares the same motivation as our work that will be presented in Chapter 4, that is to learn a unified framework for the recognition problems with auxiliary labels. In Chapter 4, we address the problem of human action recognition and consider the human poses as the auxiliary labels.

3.3 Human Activity Recognition

Human activity recognition is one of the most popular topics in visual recognition. A lot of work has been done in recognizing human activity either from still images or from video sequences. Much work focuses on recognizing actions performed by a single person. In

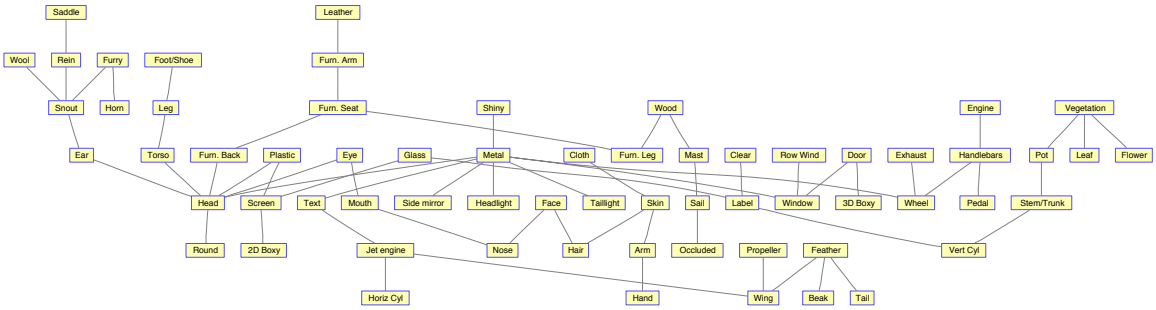


Figure 3.4: Visualization of an attribute relation graph. An edge on the graph represents a strong co-occurrence relationship. This figure is from [90].

recent years, recognizing and analyzing group activities are increasingly attracting more attention. In general, the goal of human activity recognition is to classify an image or a video sequence into one of several pre-defined categories based on the actions performed by the people in the image or the video. Activity recognition is also a very challenging problem. One of the major challenges is the intra-class variation in action. For example, for the same "walking" action, different people may perform it very differently (e.g. with different strides or speed). A successful human activity recognition system also need to cope with other challenges, such as viewpoint variation, occlusion, and clutter background.

In this section, we will review two approaches addressing human activity recognition using latent variables. One approach [91] focuses on recognizing actions performed by a single person, and the other one [45] focuses on group activity recognition. In both approaches, it is assumed the image frame has been pre-processed so the persons in the image have been localized.

3.3.1 Latent Patch

In visual recognition, the major feature representations include global template, bag-of-words, and part-based models. Inspired by [26], Wang and Mori [91] propose a principled approach to combine the global template features and the part model using latent SVM. The high-level overview of this model is illustrated in Fig. 3.5(a). Different to the pictorial structure model in [26], the part model in [91] is based on the local patches and it models the human action as a flexible constellation of parts conditioned on the appearances of local

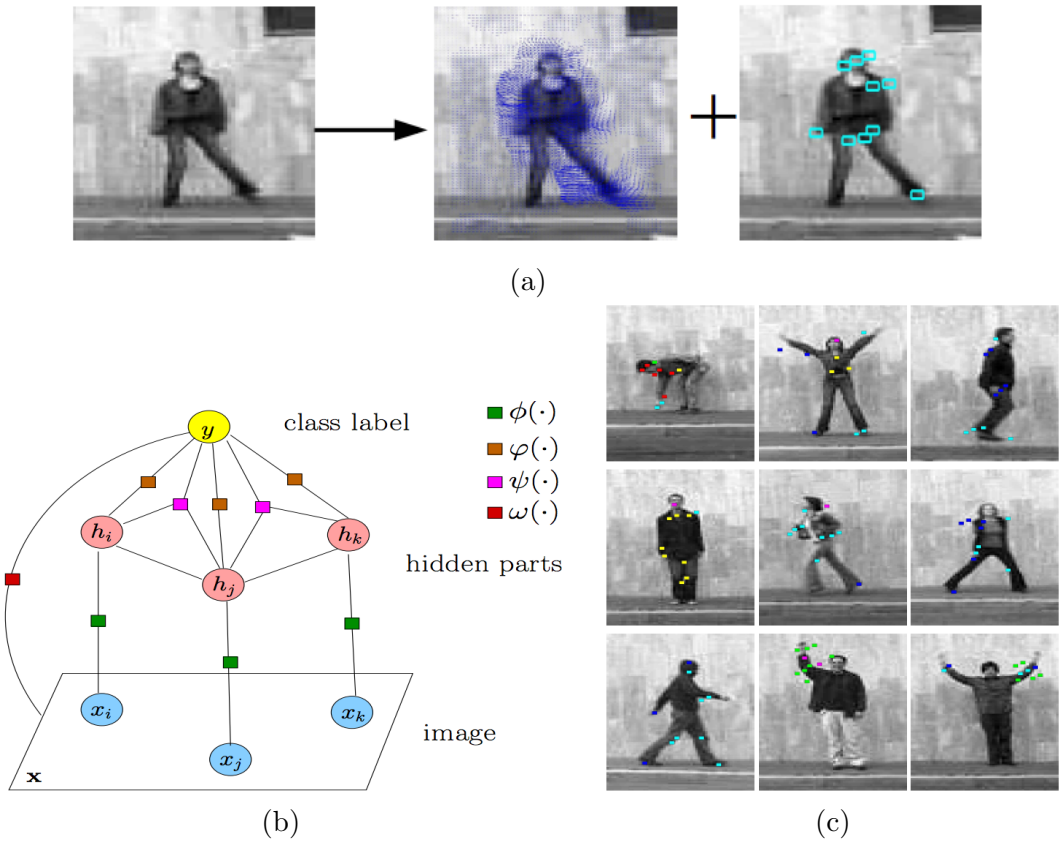


Figure 3.5: (a) illustrates the high-level motivation of the model in [91] which is to represent a human action by a large scale template feature (e.g. optical flow feature) and a set of local "parts". (b) is the graphical illustration of the model used in [91]. (c) visualizes the latent patches learned during training. Patches are colored based on their part labels. These figures are from [91].

patches. In the model, each local patch is assigned with a "part label" which indicates certain motion patterns of this patch. For example, the patches on the "jumping" person are often assigned with the "part labels" corresponding to the "moving down" or "moving up" patterns. The "part labels" of the patches are treated as latent information.

The training example is assumed to take the form of $\mathbf{x} = (s_0, s_1, ..., s_m)$, where s_0 is the motion feature extracted from the whole frame, s_i is the feature vector extracted from the i-th patch. Each example is associated with a vector of latent variables $\mathbf{h} = (h_1, h_2, .., h_m)$, where h_i denotes the "part label" of the i-th patch and it takes a discrete value from a set $\mathcal{H}$ of possible "part labels"(i.e. $h_i \in \mathcal{H}$). Again, $(h_1, h_2, .., h_m)$ forms a tree structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$,

which is obtained by running a minimum spanning tree (MST) algorithm over the patches. The model is graphically illustrated in Fig. 3.5. The potential function $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$ is defined as follow:

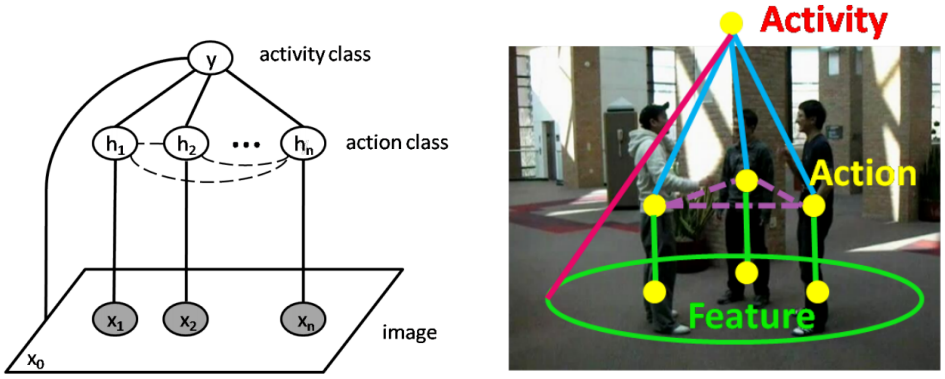
$$\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y) = w_0^\top \phi_0(y, s_0) + \sum_{j \in \mathcal{V}} w_1^\top \phi_1(s_j, h_j) + \sum_{j \in \mathcal{V}} w_2^\top \phi_x(h_j, y) + \sum_{(j,k) \in \mathcal{E}} w_3^\top \phi_3(h_j, h_k, y) \quad (3.4)$$

$w_0^\top \phi_0(y, s_0)$ is the root model which models the compatibility of the global feature s_0 extracted from the whole frame and the class label y. The second potential function $\sum_{j \in \mathcal{V}} w_1^\top \phi_1(s_j, h_j)$ captures the compatibility between latent part label h_j and the local feature s_j on the j-th patch. $\sum_{j \in \mathcal{V}} w_2^\top \phi_x(h_j, y)$ measures the compatibility of the class label y and the latent part label h_j, and $\sum_{(j,k) \in \mathcal{E}} w_3^\top \phi_3(h_j, h_k, y)$ models the dependencies between part labels h_j and h_k.

3.3.2 Latent Interaction

Lan et al. [45] focus on recognizing the activities of a group of people. It is believed that the action of an individual in a group is rarely performed in isolation. There are often strong correlations between the actions of the people in the same group. For example, the activity of "queueing" involves a group of people, and the persons in this group usually perform the same action, i.e. standing and facing to the same direction. In [45], Lan et al. propose a discriminative framework using latent SVM, which jointly models the group activity, the individual person actions, and the person-person interactions.

Assuming there are in total m persons found in the image I, the feature extracted from image I is in the form of $\mathbf{x} = (x_0, x_1, .., x_m)$, where x_i is the feature vector extracted from the i-th person. The collective actions of all the persons in the image are denoted as $\mathbf{h} = (h_1, h_2, .., h_m)$, and h_i is the action label of the i-th person. Moreover, it is assumed that the persons in the same group are interacted so that there are dependencies between the pair of actions (h_j, h_k). An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is used to capture the structure of the action labels $(h_1, h_2, ..., h_m)$. Fig. 3.6 shows the graphical representation of the model. Comparing Fig. 3.6 with Fig. 3.5, one may notice that the model and notations in [45] are very similar the ones in [91], except that the $\mathbf{h}$ in [45] are the action labels of the individuals in the image rather than the part labels. However, there is a significant difference. In [91], $\mathbf{h}$ are considered as latent and the graph $\mathcal{G}$ (i.e. a tree structure generated by MST) is predefined. But here, the action labels $\mathbf{h}$ are assumed to be provided on the training images



(a) (b)

Figure 3.6: Graphical illustrations of the model used in [45]. The dashed lines indicate that the structure of latent variables is latent. These figures are from [45].

so that they are not latent during training. Instead, the structure (i.e. $\mathcal{G}$) of action labels is treated as a latent variable. This is one of the major contributions of [45]. By treating the $\mathcal{G}$ as a latent variable, the person-person interaction becomes adaptive and it achieves better performance than a fixed graph.

Given a set of N training examples $\{(\mathbf{x}_i, \mathbf{h}_i, y)\}_{i=1}^{N}$,the model parameters can be learned by solving the following latent SVM formulation:

$$\begin{aligned} &\min_{\mathbf{w},\xi\geq 0,\mathcal{G}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\ &\text{s.t.} \quad \max_{\mathcal{G}_y}\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}, y; \mathcal{G}_y) - \max_{\mathcal{G}_{y_i}} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}_i, y_i; \mathcal{G}_{y_i}) \leq \xi_i - \Delta(y, y_i), \quad \forall i, \quad \forall y \end{aligned} \tag{3.5}$$

where $\mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}_i, y_i; \mathcal{G}_{y_i})$ denotes a potential function in which the action labels $\mathbf{h}_i$ are constrained by a particular graph $\mathcal{G}_{y_i}$. Lan et al. [45] shows that Eq. 3.5 can be solved using the non-convex cutting plane training algorithm described in Section 2.2.2.

As mentioned earlier, in the setting of [45], the action labels $\mathbf{h}$ are *observed* during training. However, this auxiliary labeling is not provided on the new testing image. This is a very common scenario since the algorithms are usually expected to be fully automatic during testing without using any manual labeling. Therefore, for a new testing image, we need infer both the action labels $\mathbf{h}$ and their structure $\mathcal{G}_y$:

$$f_{\mathbf{w}}(\mathbf{x}, y) = \max_{\mathcal{G}_y}\max_{\mathbf{h}} \mathbf{w}^\top \Phi(\mathbf{x}_i, \mathbf{h}, y; \mathcal{G}_y) \tag{3.6}$$

A coordinate ascent style algorithm is proposed in [45] for solving the optimization problem in Eq. 3.6 by alternating between the inference of $\mathbf{h}$ and the inference of $\mathcal{G}_y$.

Lastly, we emphasize that the learning approach in [45] is very different from most of the previous work using latent structured models. It does not predefine any structure for the hidden layer but rather treats it as a latent variable. Again, this learning approach demonstrates the flexibility of the latent SVM framework.

3.4 Other work

In the above three sections, we have reviewed the details of a few interesting approaches which use latent SVM to address the challenging problems in visual recognition. In this section, we will briefly go through other work which follows the same latent SVM framework, but defines different latent variables.

In object detection, to handle the occlusion and truncation on the object instances, Vedaldi and Zisserman [82] assign a binary latent variable h on each HOG cell. For example, $h_j = 1$ means that the j-th cell of the HOG descriptor is visible, and $h_j = 0$ means this cell is either occluded or truncated. Pandey and Lazebnik [59] directly apply the deformable part models [26] in both scene recognition and weakly supervised object localization. For the weakly supervised object location, it is assumed that the ground-truth bounding-box labeling is not provided on the training set, and it is then treated as the latent variable. A similar part-based model is proposed by Parizi et al. [60]. They represent each scene image as a set of region models (i.e. "parts"), and each image patch is associated with a latent region label.

In human activity recognition, Niebles et al. [54] consider an activity video as a composition of several shorter video segments which represent simpler actions. The displacements of these video segments are the latent variables and they are assumed to form a chain structure. Liu et al. [51] introduce the concept of attribute to human action recognition. They use a similar model as [90] and treat the attribute labels as latent variables. Vahdat et al. [81] focus on recognizing the interactions between two persons, and each action is modeled as a sequence of 5 key poses. Both the displacements of key poses in the videos and the choices of key poses (i.e. "which key pose we should use to represent the action at a certain time?") are treated as latent variables.

Wang and Mori [88] develop a discriminative latent model for image tagging. Unlike

many previous work where latent variables are a vector of discrete labels, the latent variables in [88] is a matching matrix which maps the image regions to their corresponding tags. Kumar et al. [44] address the semantic image segmentation which aims to assign each pixel a class label (e.g. "road", "tree", etc.). It is assumed in [44] that only bounding boxes or image-level labels are provided on the training images, and the pixel-wise segmentations are treated as latent variables. In [44], the self-paced learning algorithm (described in Section 2.2.3) is used for training the latent SVM. Wang et al. [85] address the problem of indoor scene understanding from a single image, and they focus on recovering the layouts of major faces (e.g. floor, ceiling and walls). Except these major faces, most indoor scenes also contain many cluttered objects (e.g. furniture and decorations). The layouts of these clutters are treated as latent variables. Sharma et al. [70] address a series of image classification problems using latent SVM, and treat the saliency maps of the images as latent variables. Their approach is motivated by the observations that different areas (e.g. patches) on the image do not contribute equally for image classification.

3.5 Summary

In this chapter, we show that latent SVM has been used to address various visual recognition problems, such as object detection, object classification, human action recognition, and image segmentation. The latent variables in those approaches are defined in a variety of forms and have different semantic meanings. Here, we summarize the latent variables in the literature as the following three types:

1. Missing labels. As we mentioned earlier, the latent variable model can formulate the missing labels as latent variables, and thus address the intra-category variations arising in the weakly labeled data. For example, in this chapter, we show that latent variables have been used to model the locations of object parts [26] and the viewpoints of objects [31] for the task of object detection. Those labels are missing from the training data and thus treated as latent variables.

2. Auxiliary labels. This form of latent variables is associated with a class of problems – *recognition with auxiliary labels*. For example, [90] and [51] address the problems of object classification and action recognition respectively, but both of them treat the attributes as auxiliary labels. In their problem setting, the auxiliary labels are

observed during training but remain latent during testing. The focus of this line of work is to develop unified systems that jointly consider the target category label (e.g. object class or action class) and the auxiliary label (e.g. attributes).

3. Hidden structure. This is best exemplified by [45] as we discussed in Section 3.3.2. Instead of predefining the structure of the middle-layer representation (i.e. the interactions between the individual persons), the structure itself is treated as latent information which will be automatically inferred during the learning process.

Chapter 4

Action Recognition with Latent Pose

In this chapter, we address the problem of recognizing human actions from still images. In particular, we consider the figure-centric representation. The images are preprocessed so that the person is localized in the center of the image. Our training data consists of images with ground-truth action labels (e.g. "walking", "running", "sitting", etc.) and pose annotations (i.e. the joint positions of human body). Our goal is to train a model which can predict the action label of the testing image, for which ground-truth pose annotations are not provided. Different from other work that learns separate systems for pose estimation and action recognition, then combines them in an ad-hoc fashion, we propose a novel latent variable model that jointly considers the human actions and poses. This model is trained in an integrated fashion using latent SVM, and the learning objective is designed to directly exploit the pose information for action recognition.

The rest of this chapter is organized as follows. We first give an overview of our approach in Section 4.1. We describe the human pose representation – an action-specific variant of the "poselet" [9] in Section 4.2. The model formulation and the learning algorithm are described in Section 4.3 and Section 4.4 respectively. Section 4.5 presents the experimental results which demonstrate that by inferring the latent poses, the results of action recognition can be improved. Section 4.6 concludes this chapter.

4.1 Overview

Consider the two images shown in Fig. 4.1 (left). Even though only still images are given, we as humans can still perceive the actions (walking, playing golf) conveyed by those images. The primary goal of this work is to recognize actions from still images. In still images, the information about the action label of an image mainly comes from the pose, i.e. the configuration of body parts, of the person in the image. However, not all body parts are equally important for differentiating various actions. Consider the poses shown in Fig. 4.1 (middle). The configurations of torso, head and legs are quite similar for both walking and playing golf. The main difference for these two actions in terms of the pose is the configuration of the arms. For example, "playing golf" seems to have very distinctive V-shaped arms, while "walking" seems to have two arms hanging on the side. A standard pose estimator tries to find the correct locations of all the body parts. The novelty of our work is that we do not need to correctly infer complete pose configuration in order to do action recognition. In the example of "walking" versus "playing golf", as long as we get correct locations of the arms, we can correctly recognize the action, even if the locations of other body parts are incorrect. The challenge is how to learn a system that is aware of the importance of different body parts, so it can focus on the arms when trying to differentiate between "walking" and "playing golf". We introduce a novel model that jointly learns poses and actions in a principled framework.

Human action recognition is an extremely important and active research area in computer vision, due to its wide range of applications, e.g. surveillance, entertainment, human-computer interaction, image and video search, etc. Space constraints do not allow an extensive review of the field, but a comprehensive survey is available in [30]. Most of the work in this field focuses on recognizing actions from videos [47, 56, 68] using motion cues, and a significant amount of progress has been made in the past few years. Action recognition from still images, on the other hand, has not been widely studied. We believe analyzing actions from still images is important. Progress made here can be directly applied to videos. There are also applications that directly require understanding still images of human actions, e.g. news/sports image retrieval and analysis.

Not surprisingly, recognizing human actions from still images is considerably more challenging than video sequences. In videos, the motion cue provides a rich source of information for differentiating various actions. But in still images, the only information we can rely on

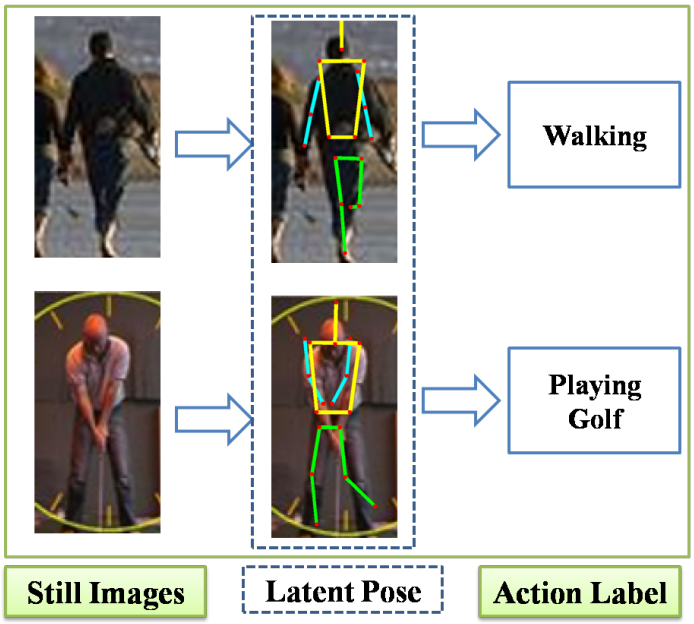


Figure 4.1: Illustration of our proposed approach. Our goal is to infer the action label of a still image. We treat the pose of the person in the image as "latent variables" in our system. The "pose" is learned in a way that is directly tied to action classification.

is the shape (or the pose) of the person in an image. Previous work mainly focuses on building good representations for shapes and poses of people in images. Wang et al. [86] cluster different human poses using distances calculated from deformable shape matching. Thurau and Hlaváč [76] represent actions using histograms of pose primitives computed by non-negative matrix factorization. Ikizler et al. [36] recognize actions using a descriptor based on histograms of oriented rectangles. Ikizler-Cinbis et al. [37] learn actions from web images using HOG descriptors [14]. A limitation of these approaches is that they all assume an image representation based on global templates, i.e. an image is represented by a feature descriptor extracted from the whole image. This representation has been made popular due to its success in pedestrian detection, in particular the work on histogram of oriented gradient (HOG) by Dalal and Triggs [14]. This representation might be appropriate for pedestrian detection, since most pedestrians are upright. So it might be helpful to represent all the pedestrians using a global template. But when it comes to action recognition, global templates are not flexible enough to represent the huge amount of variations for an action.

For example, consider the images of the "playing golf" action in Fig. 4.4. It is hard to imagine that a single global template can capture all the pose variations of this action. Recently, Felzenszwalb et al. [25] show that part-based representations can better capture the pose variations of an object, hence outperform global template representations. In our approach, we operationalize on the same intuition and demonstrate that part-based representations are useful for action recognition in still images as well. A major difference of our work from [25] is that we have ground-truth labeling of the pose on the training data, i.e. our "parts" are semantically meaningful.

Another important goal of our approach is to bridge the gap between *human action recognition* and *human pose estimation*. Those are two closely related research problems. If we can reliably estimate the pose of a person, we can use this information to recognize the action. However, in the literature, they are typically touted as two separate research problems and there has been only very little work on combining them together. There is some work on trying to combine these two problems in a cascade way, e.g. by building an action recognition system on top of the output of a pose estimation system. For example, Ramanan and Forsyth [64] annotate and synthesize human actions in 3D by track people in 2D and match the track to an annotated motion capture dataset. Their work uses videos rather than still images, but the general idea is similar. Ferrari et al. [29] retrieve TV shots containing a particular 2D human pose by first estimating the human pose, then searching shots based on a feature vector extracted from the pose. But it has been difficult to establish the value of pose estimation for action recognition in this cascade manner, mainly because pose estimation is still a largely unsolved problem. It is questionable whether the output of any pose estimation algorithm is reliable enough to be directly used for action recognition.

In this chapter, we propose a novel way of combining action recognition and pose estimation together to achieve the end goal of action recognition. Our work is different from previous work in two perspectives. First, instead of representing the human pose as the configuration of kinematic body parts [63], e.g. upper-limb, lower-limb, head, etc, we choose to use an exemplar-based pose representation, "poselet". This notation of "poselet" is first proposed in [9] and used to denote a set of patches with similar 3D pose configuration. For the purpose of action recognition, we further restrict those patches not only to have similar configuration, but also from the same action class. Second, as illustrated by the diagram in Fig. 4.2 (top), previous work typically treats pose estimation and action recognition as two separate learning problems, and uses the output of a pose estimation algorithm as the

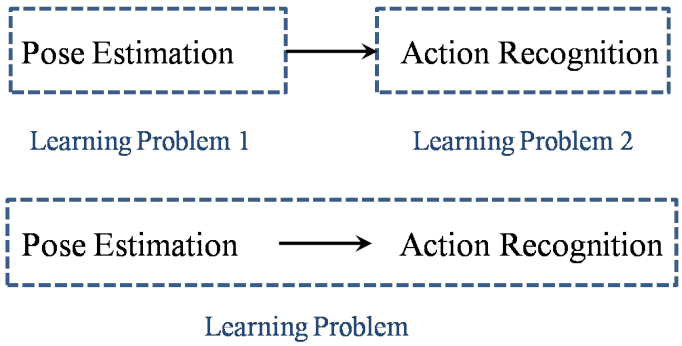


Figure 4.2: Difference between previous work and ours. (Top) Previous work typically approaches pose estimation and action recognition as two separate problems, and uses the output of the former as the input to the latter. (Bottom) We treat pose estimation and action recognition as an single problem, and learn everything in an integrated framework.

input of an action recognition system [29, 36]. As pointed out earlier, the problem with this approach is that the output of the pose estimation is typically not reliable. Instead, as illustrated by the diagram in Fig. 4.2 (bottom), we treat pose estimation and action recognition as two components of a single learning problem, and jointly learn the whole system in an integrated manner. But our learning objective is designed in a way that allows pose information to help action classification.

The high-level idea of our proposed approach can be seen from Fig. 4.1. Our goal is to infer the action label of a still image. We treat the pose of the person as intermediate information useful for recognizing the action. But instead of trying to infer the pose correctly using a pose estimation algorithm, we treat the pose as *latent variables* in the whole system. Compared with previous work on exploiting pose for recognition [64, 29], the "pose" in our system is learned in a way that is directly tied to our end goal of action classification.

4.2 Pose Representation

In our model, we treat human pose as latent information and use it to assist the task of action recognition. Since we do not aim to obtain good pose estimation results in the end, the latent pose in our approach is not restricted to any specific type of pose representation. Because our focus is action recognition, we decide to choose a coarse exemplar-based pose representation. It is an action-specific variant of the "poselet" proposed in [9]. In this chapter, we use the notation of "poselet" to refer to a set of patches not only with similar pose configuration,

but also from the same action class. Fig. 4.3 illustrates the four poselets of a walking image. As we can see, the poselet normally covers more than one semantically meaningful part in terms of limbs and thus it is distinct from the background. So, the detection of poselets is more reliable than limb detection, especially with cluttered backgrounds.

In [9], a dataset is built where the joint positions of each human image are labeled in 3D space via a 2D-3D lifting procedure. We simply annotate the joint positions of human body in the 2D image space, as shown in Fig. 4.4. From the pose annotation, we can easily collect a set of patches with similar pose configuration. Based on the intuition that action-specific parts contain more discriminative information, we decide to select the poselets per action. For example, we would like to select a number of poselets from running-legs, or walking arms. The procedure of poselet selection for a particular action (e.g. running) is as follows: 1. We first divide the human pose annotation of the running images into four parts, legs, left-arm, right-arm, and upper-body; 2. We cluster the joints on each part into several clusters based their normalized x and y coordinates; 3. We remove clusters with very few examples; 4. Based on the pose clusters, we crop the corresponding patches from the images and form a set of poselets for the running action. Representative poselets from the running action are shown in Fig. 4.5. As we can see, among each poselet the appearance of each patch looks different, but they have very similar semantic meaning. As pointed in [9], this is also one advantage of using poselets. We repeat this process for other actions and obtain 90 poselets in total in the end.

In order to detect the presence of each poselet, we train a classifier for each poselet. We use the standard linear SVM and the histograms of oriented gradients feature proposed by Dalal and Triggs [14]. The positive examples are the patches from each poselet cluster. The negative examples are randomly selected from images which have the different action label to the positive examples. For example, when we train the classifier for one of "running-legs" poselets, we select the negative examples from all other action categories except for the running action. The learned running poselet templates are visualized in the last column in Fig. 4.5.

4.3 Model Formulation

Let I be an image containing a person. We consider a figure-centric representation where I only contains one person centered in the middle of the image. This representation can

Figure 4.3: Visualization of the poselets for a walking image. Ground-truth skeleton is overlayed on image. Examples of poselets for each part are shown.

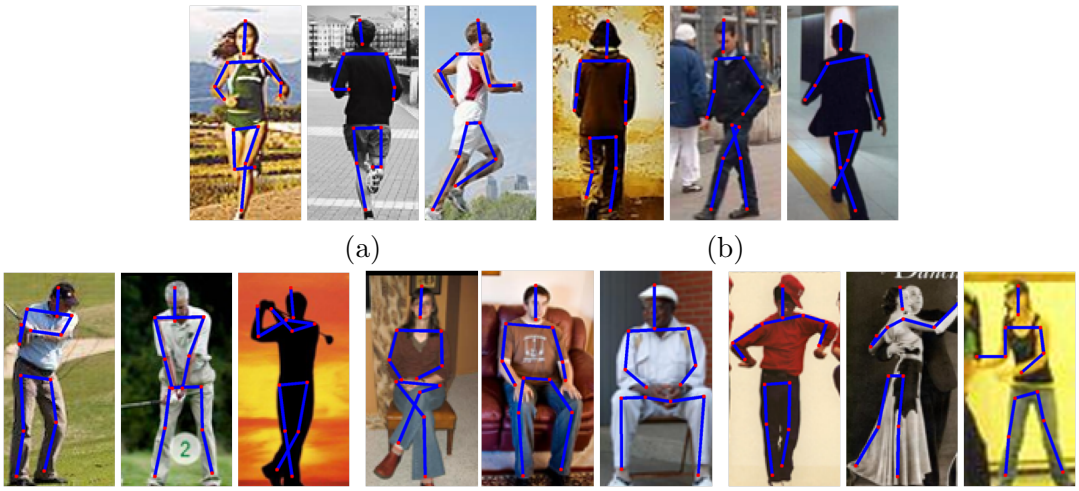


Figure 4.4: Sample images of the still image action dataset [37], and the ground truth pose annotation. The locations of 14 joints have been annotated on each action image. (a) Running; (b) Walking; (c) Playing Golf; (d) Sitting; (e) Dancing.

Figure 4.5: Examples of poselets for each part from the running action. Each row corresponds to one poselet. The last column is the visualization of the filters for each poselet learned from SVM + HOG.

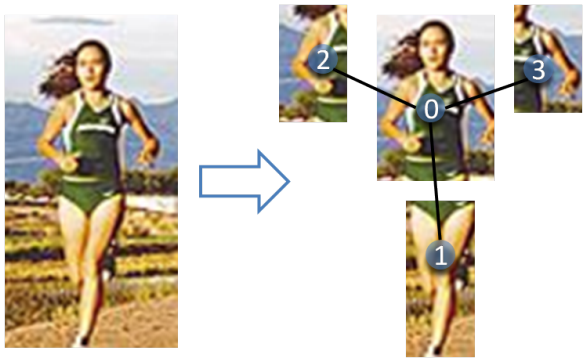


Figure 4.6: The four part star structured model. We divide the pose into four parts: legs, left-arm, right-arm, and upper-body.

be obtained from a standard pedestrian detection system. Let Y be the action label of the person, and L be the pose of the person. We denote L as $L = (l_0, l_1, ..., l_{K-1})$, where K is the number of parts. In our model, we choose $K = 4$ corresponding to upper-body, legs, left-arm, and right-arm. The configuration of the k-th part l_k is represented as $l_k = (x_k, y_k, z_k)$, where (x_k, y_k) indicates the (x, y) locations of the k-th part in the image, and $z_k \in \mathcal{Z}_k$ is the index of the chosen poselet for the k-th part. We have used $\mathcal{Z}_k$ to denote the poselet set corresponding to the part k. In our model, we use $|\mathcal{Z}_k|$ as 26, 20, 20, 24 for the four parts: legs, left-arm, right-arm, and upper-body, based on our clustering results.

Similar to the standard pictorial structure models [27, 63] in human pose estimation, we use an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to constrain the configuration of the pose L. Usually the kinematic tree of the human body is used. A vertex $j \in \mathcal{V}$ corresponds to the configuration l_j of the j-th part, and an edge $(j, k) \in \mathcal{E}$ indicates the dependency between two connected parts l_j and l_k. In this work, we use a simple four part star structured model, as shown in Fig. 4.6. The upper-body part is the root node of $\mathcal{G}$ and other parts are connected to the root node. We emphasize that our algorithm is not limited to the four part star structure and can be easily generalized to other types of tree structures.

Our training data consists of images with ground-truth labels of their action classes and poses (i.e. (x, y) location of each part and its chosen poselet). The ground-truth poselet of a part is obtained by tracing back the poselet cluster membership of this part. Given a set of N training examples $\{(I^{(n)}, L^{(n)}, Y^{(n)})\}_{n=1}^{N}$, our goal is to learn a model that can be used to assign the class label Y to an unseen test image I. Note that during testing, we do not know the ground-truth pose L of the test image I.

We are interested in learning a discriminative function $H : \mathcal{I} \times \mathcal{Y} \rightarrow \mathbb{R}$ over an image I and its class label Y, where H is parameterized by Θ. During testing, we can predict the class label Y^* of an input image I as:

$$Y^* = \arg\max_{Y \in \mathcal{Y}} H(I, Y; \Theta) \tag{4.1}$$

We assume $H(I, Y; \Theta)$ takes the following form:

$$H(I, Y; \Theta) = \max_{L} \Theta^T \Psi(I, L, Y) \tag{4.2}$$

where $\Psi(I, L, Y)$ is a feature vector depending on the image I, its pose configuration L and its class label Y. We define $\Theta^T \Psi(I, L, Y)$ as follows:

$$\Theta^T \Psi(I, L, Y) = \sum_{j \in \mathcal{V}} \alpha_j^T \phi(I, l_j, Y) + \sum_{(i,j) \in \mathcal{E}} \beta_{jk}^T \psi(l_j, l_k, Y) + \eta^T \omega(l_0, Y) + \gamma^T \varphi(I, Y) \tag{4.3}$$

The model parameters Θ are simply the concatenation of the parameters in all the factors, i.e. $\Theta = \{\alpha_j : j \in \mathcal{V}\} \cup \{\beta_{j,k} : (j,k) \in \mathcal{E}\} \cup \{\gamma\}$. The details of the potential functions in Eq. 4.3 are described below.

Part appearance potential $\alpha_j^T \phi(I, l_j, Y)$: This potential function models the compatibility between the action class label Y, the configuration $l_j = (x_j, y_j, z_j)$ of the j-th part, and the appearance of the image patch extracted from the location (x_j, y_j). It is parameterized as:

$$\alpha_j^T \phi(I, l_j, Y) = \sum_{a \in \mathcal{Y}} \sum_{b \in \mathcal{Z}_j} \alpha_{jab}^T \cdot 1_a(Y) \cdot 1_b(z_j) \cdot f(I(l_j)) \tag{4.4}$$

where $1_a(X)$ is an indicator that takes the value 1 if $X = a$, and 0 otherwise. We use $f(I(l_j))$ to denote the feature vector extracted from the patch defined by $l_j = (x_j, y_j, z_j)$ in the image I. The poselet set for the j-th part is denoted as $\mathcal{Z}_j$. The parameter α_{jab} represents a template for the j-th part if the action label is a and the chosen poselet for the j-th part is b.

Instead of keeping $f(I(l_j))$ as a high dimensional vector, we simply use the output of a SVM classifier trained on a particular poselet as the single feature. We append a constant 1 to $f(I(l_j))$ to learn a model with a bias term. In other words, let $f_{ab}(I(l_j))$ be the score of the SVM trained with action a and poselet b. Then the parameterization can be re-written as:

$$\alpha_j^T \phi(I, l_j, Y) = \sum_{a \in \mathcal{Y}} \sum_{b \in \mathcal{Z}_j} \alpha_{jab}^T \cdot 1_a(Y) \cdot 1_b(z_j) \cdot [f_{ab}(I(l_j)); 1] \tag{4.5}$$

This trick greatly speeds up our learning algorithm. Similar tricks are used in [17].

Pairwise potential $\beta_{jk}^T \psi(l_j, l_k, Y)$: This potential function represents the dependency between the j-th and the k-th part, for a given class label Y. Similar to [63], we use discrete binning to model the spatial relations between parts. We define this potential function as

$$\beta_{jk}^T \psi(l_j, l_k, Y) = \sum_{a \in \mathcal{Y}} \beta_{jka}^T \cdot \text{bin}(l_j - l_k) \cdot 1_a(Y) \tag{4.6}$$

where $\text{bin}(l_j - l_k)$ is a feature vector that bins the relative location of the j-th part with respect to the k-th part according to the (x, y) component of l_j and l_k. Hence $\text{bin}(l_j - l_k)$ is a sparse vector of all zeros with a single one for the occupied bin. Here β_{jka} is a model parameter that favors certain relative bins for the j-th part with respect to the k-th part for the action class label a.

Root location potential $\eta^T \omega(l_0, Y)$: This potential function models the compatibility between the action class label Y and the root location. Here l_0 denotes the configuration of the "root" part, i.e. upper-body in our case. It is parameterized as:

$$\eta^T \omega(l_0, Y) = \sum_{a \in \mathcal{Y}} \eta_a^T \cdot \text{bin}(l_0) \cdot 1_a(Y) \tag{4.7}$$

We discretize the image grid into $h \times w$ spatial bins, and $\omega(l_0)$ is a length $h \times w$ sparse vector of all zeros with a single one for the spatial bin occupied by the root part. The parameter η_a favors certain bins (possibly those in the middle of the image) for the location of the root part for the action label a. For example, for the running and walking actions, the root part may appear in the upper-middle part of the image with high probability, while for the sitting or playing golf action, the root part may appear in the center-middle or lower-middle part of the image. This potential function deals with different root locations for different actions. It also allows us to handle the unreliability caused by the human detection system.

Global action potential $\gamma^T \varphi(I, Y)$: This potential function represents a global template model for action recognition from still images without considering the pose configuration. It is parameterized as follows:

$$\gamma^T \varphi(I, Y) = \sum_{a \in \mathcal{Y}} \gamma_a^T \cdot 1_a(Y) \cdot f(I) \tag{4.8}$$

where $f(I)$ is a feature vector extracted from the whole image I. The parameter γ_a is a template for the action class a. This potential function measures the compatibility between the model parameter γ and the combination of image observation $f(I)$ and its class label

Y. Similar to the part appearance model, we represent $f(I)$ as a vector of outputs of a multi-class SVM classifier.

4.4 Learning and Inference

We now describe how to infer the action label Y given the model parameters Θ (Sec. 4.4.1), and how to learn the model parameters from a set of training data (Section 4.4.2)

4.4.1 Inference

Given the model parameters and a test image I, we can enumerate all the possible action labels $Y \in \mathcal{Y}$ and predict the action label Y^* of I according to Eq. 4.1. For a fixed Y, we need to solve an inference problem of finding the best pose L^{best} as follows:

$$\begin{aligned} L^{\text{best}} &= \arg\max_L \Theta^T \Psi(I, L, Y) \\ &= \arg\max_L \Big(\sum_{j \in \mathcal{V}} \alpha_j^T \phi(I, l_j, Y) + \sum_{(i,j) \in \mathcal{E}} \beta_{jk}^T \psi(l_j, l_k, Y) + \eta^T \omega(l_0, Y) \Big) \end{aligned} \tag{4.9}$$

Note for a fixed Y, the global action potential function is a constant and has nothing to do with the pose L, so we omit it from above equation. Since we assume a star model on L, the inference problem in Eq. 4.9 can be efficiently solved via dynamic programming.

4.4.2 Learning

Now we describe how to train the model parameters Θ from N training examples $\{I^n, L^n, Y^n\}_{n=1}^N$. If we assume the pose L is unobserved on the training data, we can learn Θ using the latent SVM formulation [25, 87] as follows:

$$\begin{aligned} \min_{\Theta, \xi^n \geq 0} \quad & \Theta^T \Theta + C \sum_n \xi^n \\ \text{s.t.} \quad & \underbrace{\max_L \Theta^T \Psi(I^n, L, Y^n)}_{H(I^n, Y^n; \Theta)} - \underbrace{\max_L \Theta^\top \Psi(I^n, L, Y)}_{H(I^n, Y; \Theta)} \\ & \geq \Delta(Y, Y^n) - \xi^n, \quad \forall n, \quad \forall Y \in \mathcal{Y} \end{aligned} \tag{4.10}$$

where $\Delta(Y, Y^n)$ is a function measuring the loss incurred by classifying the example I^n to be Y, while the true class label is Y^n. We use the 0-1 loss defined as follows:

$$\Delta(Y, Y^n) = \begin{cases} 1 & \text{if } Y \neq Y^n \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

The constraint in Eq. 4.10 specifies the following intuition. For the n-th training example, we want the score $H(I^n, Y; \Theta) = \arg\max_L \Theta^T \Psi(I^n, L, Y)$ to be high when Y is the true class label, i.e. $Y = Y^n$. In particular, we want the score $H(I^n, Y^n; \Theta)$ to be higher than the score associated with any hypothesized class label $H(I^n, Y; \Theta)$ by 1 if $Y \neq Y^n$.

Now since L is observed on training data, one possible way to learn Θ is to plug-in the ground-truth pose in Eq. 4.10, i.e. optimize the following problem:

$$\begin{aligned} \min_{\Theta, \xi^n \geq 0} \quad & \Theta^T \Theta + C \sum_n \xi^n \\ \text{s.t.} \quad & \Theta^T \Psi(I^n, L^n, Y^n) - \max_L \Theta^\top \Psi(I^n, L, Y) \\ & \geq \Delta(Y, Y^n) - \xi^n, \quad \forall n, \quad \forall Y \in \mathcal{Y} \end{aligned} \tag{4.12}$$

Our initial attempt of using Eq. 4.12 suggests it does not perform as well as Eq. 4.10. We believe it is because the learning objective in Eq. 4.12 assumes that we will have access to the correct pose estimation at run-time. This is unrealistic. On the other hand, the learning objective in Eq. 4.10 mimics the situation at run-time, when we are faced with a new image without the ground-truth pose. So we will use the formulation in Eq. 4.10 from now on. But we would like to point out that Eq. 4.10 does not ignore the ground-truth pose information on the training data. That information has been implicitly built into the features which are represented as outputs of SVM classifiers. Those SVM classifiers are learned using the ground-truth information of the pose on the training data.

The training problem in Eq. 4.10 can be solved by the non-convex cutting plane algorithm in [19], which is an extension of the popular convex cutting plane algorithm [39] for learning structural SVM [1]. We briefly outline the algorithm here.

Consider the following unconstrained formulation which is equivalent to Eq. 4.10:

$$\begin{aligned} & \Theta = \arg\min_\Theta \Theta^T \Theta + C \sum_n R_n(\Theta) \quad \text{where} \\ & R_n(\Theta) = \max_Y \left(\Delta(Y, Y^n) + H(I^n, Y; \Theta)\right) - H(I^n, Y^n; \Theta) \end{aligned} \tag{4.13}$$

In a nutshell, the learning algorithm in [19] iteratively builds an increasingly accurate piecewise quadratic approximation of Eq. 4.13 based on the subgradient $\partial_\Theta \left(\sum_n R_n(\Theta)\right)$. It can be shown that the subgradient $\partial_\Theta \left(\sum_n R_n(\Theta)\right)$ is related to the most-violated constraint of Eq. 4.10. So in essence, the algorithm iteratively adds the most-violated constraint of Eq. 4.10 and solves a piecewise quadratic approximation at each iteration. It has been

proved that only a small number of constraints are needed in order to achieve an reasonably accurate approximation to the original problem [1].

Now the key issue is how to compute the subgradient $\partial_\Theta \left(\sum_n R_n(\Theta)\right)$. Since

$$\partial_\Theta \left(\sum_n R_n(\Theta) \right) = \sum_n \partial_\Theta R_n(\Theta) \tag{4.14}$$

all we need to do it to figure out how to compute $\partial_\Theta R_n(\Theta)$. Let us define:

$$(Y^*, L^*) = \arg\max_{Y,L} \Delta(Y, Y^n) + \Theta^T \Psi(I^n, L, Y) \tag{4.15}$$

$$\widehat{L} = \arg\max_{L} \Theta^T \Psi(I^n, L, Y^n) \tag{4.16}$$

Then, it can be shown that $\partial_\Theta R_n(\Theta)$ can be calculated as

$$\partial_\Theta R_n(\Theta) = \Psi(I^n, L^*, Y^*) - \Psi(I^n, \widehat{L}, Y^n) \tag{4.17}$$

4.4.3 Computational Complexity

The learning algorithm of our model is an iterative method. The computation of each iteration is dominated by the calculation of the subgradient $\partial_\Theta \left(\sum_n R_n(\Theta)\right)$. In particular, it involves solving two inference problems in Eq. 4.15 and Eq. 4.16 for each training example. The inference on $\arg\max_Y$ is easy, since the number of possible choices of Y is small (e.g. $|\mathcal{Y}| = 5$ in our case). So we can simply enumerate all possible $Y \in \mathcal{Y}$ with the complexity $O(|\mathcal{Y}|)$.

As mentioned in Section 4.4.1, because we assume the pose L is constrained by a star structure, the inference on $\arg\max_L$ (Eq. 4.9 and Eq. 4.16) can be efficiently solved via dynamic programming. Consider the situation that we have K body parts in the model. Each body part is associated with Z poselets, and it can be displaced on M possible locations in the image. The computational complexity of the inference on $\arg\max_L$ is $O(KZ^2M^2)$. The efficiency of this inference problem can be further improved using the generalized distance transform [27]. In our experiments, we set M as a relatively small value, e.g. the size of relative location binning $\text{bin}(l_j - l_k)$ is only 32×15, so we are able to solve the inference problems efficiently even without using the generalized distance transform. The inference for a fixed Y on an image only takes $0.015s$ by our MATLAB/MEX implementation.

In each iteration of our learning algorithm, we need to solve the inference problem (Eq. 4.15 and Eq. 4.16) for every training example, so the model might take a very long time to train

when we have a large-scale training set. To further improve the training efficiency, we can easily parallelize the computing of inferences over a number of CPUs, and each CPU takes care of a subset of training examples.

4.5 Experiments

We first test our algorithm on the still image action dataset collected by Ikizler-Cinbis et al. [37]. This dataset consists of still images from five action categories: running, walking, playing golf, sitting, and dancing. The images of this dataset are downloaded from the Internet. So there are a lot of pose variations and cluttered backgrounds in the dataset. In total, there are 2458 images in the dataset. We further increase the size and pose variability of the dataset by mirror-flipping all the images. Most of actions in the dataset do not have axial symmetry. For example, running-to-left and running-to-right appear very different in the image. So mirror-flipping makes the dataset more diverse. We manually annotate the pose with 14 joints on the human body on all the images in the dataset, as shown in Fig. 4.4. We select 1/3 of the images from each action category to form the training set, and the rest of the images are used for testing. We also ensure the testing set does not contain the mirror-flipped version of any training image. Since we focus on action classification rather than human detection, we simply normalize each image into the same size and put the human figure in the center of the image, based on the pose annotation information.

At the training stage, we create 90 poselets in total from the training set following the method described in Section 4.2. For each poselet, we train an SVM classifier based on the HOG descriptors extracted from image patches at the ground-truth locations of the corresponding poselet in the training set. Examples of trained templates for running poselets are visualized in the last column of Fig. 4.5. We compare our approach with a multi-classification SVM with HOG descriptors as the baseline. Note that the outputs of this baseline are also used to model the global action potential function in our approach. The confusion matrices of the baseline and our method on the testing set are shown in Fig. 4.7 (a),(b). Table 4.1 summarizes the comparison between our result and the baseline. Since the testing set is imbalanced, e.g. the number of running examples are more than twice of the playing-golf and sitting examples, we report both overall and mean per class accuracies. For both overall and mean per class accuracies, our method outperforms the baseline.

Figure 4.7: Confusion matrices of the classification results on the still image action dataset: (a) baseline (b) our approach. Horizontal rows are ground truths, and vertical columns are predictions.

We also apply our trained model on a Youtube video dataset originally collected by Niebles et al. [55][1]. Ikizler-Cinbis et al. [37] have annotated 11 videos of this dataset. The action of each human figure on each frame has been annotated by one of the five action categories. The bounding box information of the human figure returned by a standard human detection algorithm is also provided by Ikizler-Cinbis et al. [37]. In total, there are 777 human figures. We normalize each human figure into the same size based on the bounding box information and then run our model, which is trained from the still image dataset. To show the generalization power of our method, we use exactly the same model learned from our previous experiment on the still image action dataset without any re-training on the Youtube dataset. The confusion matrix of our method is given in Fig. 4.8. Table 4.2 shows the comparison of our method with the baseline SVM classifier on HOG features trained from the same training set. Our method performs much better in terms of both overall and mean per-class accuracies. Our results are lower than the best results without temporal smoothing reported in [37]. This is likely because the method in [37] uses an additional step of perturbing the bounding box on the training set to account for the errors of human localization. If we use the same trick in our method, the performance will probably improve as well.

[1]http://vision.stanford.edu/projects/extractingPeople.html

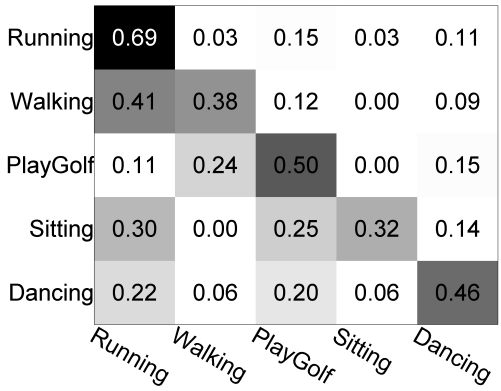


Figure 4.8: Confusion matrix of the classification results of our approach on the Youtube dataset. Horizontal rows are ground truths, and vertical columns are predictions.

method	overall	mean per-class
Baseline	56.45	52.46
Our approach	**61.07**	**62.09**

Table 4.1: Results on the still image action dataset. We report both overall and mean per class accuracies due to the class imbalance.

method	overall	mean per-class
Baseline	46.98	40.52
Our approach	**50.58**	**46.73**
[37] (MultiSVM)	59.35	N/A
[37] (Best)	63.61	N/A

Table 4.2: Results on the Youtube dataset. We report both overall and mean per class accuracies due to the class imbalance.

4.5.1 Visualization of Latent Pose

We can also visualize the latent poses learned by our model. But first, we need to point out that our model is trained for *action classification*, not *pose estimation*. We simply treat the pose as latent information in the model that can help solve the action classification task. Since our model is not directly optimized for pose estimation, we do not expect to get pose estimation results that are "good" in the usual sense. When measuring the performance of a pose estimation, people typically examine how closely the localized body parts (torso, arm, legs, etc) match the ground-truth locations of the parts in the image. But since our final goal is action classification, we are not aiming to correctly localize the body parts, but rather focus on localizing the body parts that are *useful for action classification*.

Fig. 4.9 shows the visualization of the latent poses superimposed on the original images. For the k-th part with $l_k = (x_k, y_k, z_k)$, we place the chosen poselet z_k at the location (x_k, y_k) in the image. The skeleton used for a particular poselet is obtained from the cluster center of the joint locations of the corresponding poselet. In terms of pose estimation in the usual sense, those results are not accurate. However, we can make several interesting observations. In the "sitting" action, our model almost always correctly localizes the legs. In particular, it mostly chooses the poselet that corresponds to the "A" shaped-legs (e.g. first two images in the fourth row) or the triangle-shaped legs (e.g. the third image in the fourth row). It turns out the legs of a person are extremely distinctive for the "sitting" action. So our model "learns" to focus on localizing the legs for the sitting action, in particular, our model learns that the "A" shaped-legs and the triangle-shaped legs are most discriminative for the sitting action. For the sitting action, the localized arms are far from their correct locations. From the standard pose estimation point of view, this is considered as a failure case. But for our application, this is fine since we are not aiming to correctly localize all the parts. Our model will learn not to use the localizations of the arms to recognize the sitting action. Another example is the "walking" action (the images in the second row). For this action, our model almost always correctly localizes the arms hanging on the two sides of the torso, even on the bad examples. This is because "hanging arms" is a very distinctive poselet for the walking action. So our model learns to focus on this particular part for walking, without getting distracted by other parts.

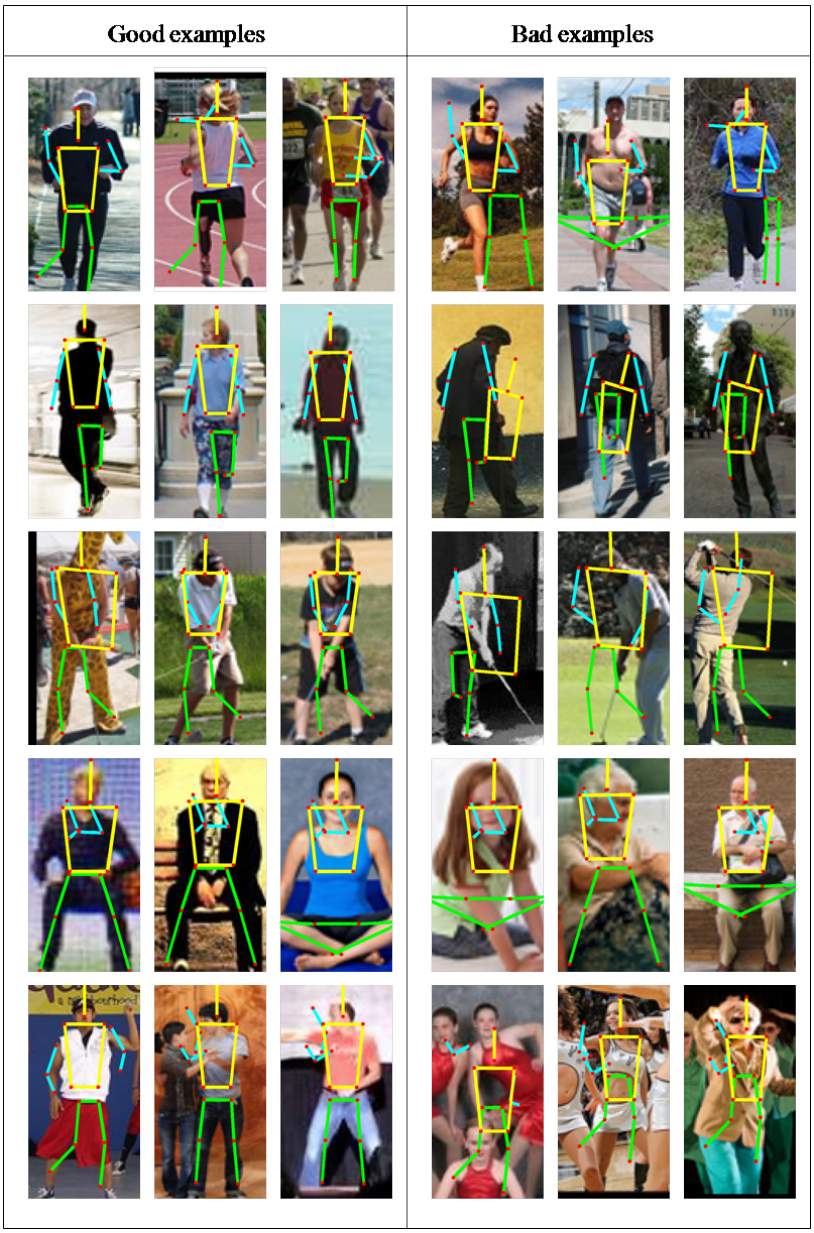


Figure 4.9: Example visualizations of the latent poses on test images. For each action, we manually select some good estimation examples and bad examples. The action for each row (from top) is running, walking, palying golf, sitting and dancing respectively.

4.6 Summary

We have presented a model that integrates action recognition and pose estimation. The main novelty of our model is that although we consider these two problems together, our end goal is action recognition, and we treat the pose information as latent variables in the model. The pose is directly learned in a way that is tied to action recognition. This is very different from other work that learns a pose estimation system separately, then uses the output of the pose estimation to train an action recognition system. Our experimental results demonstrate that by inferring the latent pose, we can improve the final action recognition results.

Chapter 5

Kernel Latent SVM

In previous chapters, we show that latent SVM (LSVM) is a class of powerful tools that have been successfully applied to many applications in computer vision. However, a limitation of latent SVM is that it relies on linear models (i.e. $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{h}, y)$). For many computer vision tasks, linear models are suboptimal and nonlinear models learned with kernels typically perform much better. Therefore it is desirable to develop the kernel version of latent SVM.

Latent SVM and kernel methods represent two different, yet complementary approaches for learning classification models that are more expressive than linear classifiers. They both have their own advantages and limitations. The latent variables in latent SVM can often have some intuitive and semantic meanings. As a result, it is usually easy to adapt latent SVM to capture various prior knowledge about the training data in various applications. Examples of latent variables in the literature include part locations in object detection [26], subcategories in video annotation [94], object localization in image classification [42], etc. However, latent SVM is essentially a parametric model. So the capacity of these types of models is limited by the parametric form. In contrast, kernel methods are non-parametric models. The model complexity is implicitly determined by the number of support vectors. Since the number of support vectors can vary depending on the training data, kernel methods can adapt their model complexity to fit the data.

In this chapter, we propose *kernel latent SVM (KLSVM)* – a new learning framework that combines latent SVMs and kernel methods. As a result, KLSVM has the benefits of both approaches. On one hand, the latent variables in KLSVM can be something intuitive and semantically meaningful. On the other hand, KLSVM is nonparametric in nature, since the decision boundary is defined implicitly by support vectors. We demonstrate KLSVM

on three applications in visual recognition: 1) object classification with latent localization; 2) object classification with latent subcategories; 3) recognition of object interactions.

The rest of this chapter is organized as follows. First, we review the binary latent SVM and derive its dual form with fixed latent variable h in Section 5.1. The formulation of KLSVM and an iterative training algorithm are described in Section 5.2. Section 5.3 presents the experimental results of KLSVM on three different visual recognition applications, and Section 5.4 concludes this chapter.

5.1 Preliminaries

In this section, we introduce some background on latent SVM and on the dual form of SVMs used for deriving kernel SVMs. Our proposed model in Section 5.2 will build upon these two ideas.

5.1.1 Binary Latent SVM

In Chapter 2, we introduce the general formulation of latent SVM for the multi-class classification, which follow the form of multi-class SVM proposed by Crammer and Singer [13]. In this chapter, we consider binary classification for simplicity, i.e. $y \in \{+1, -1\}$. Note that multi-class classification can be easily converted to binary classification, e.g. using one-vs-all or one-vs-one strategy. Here, we briefly review the binary latent SVM.

In binary latent SVM, a data instance is assumed in the form of $(\mathbf{x}, h, y)$, where $\mathbf{x}$ is the observed variable and y is the class label. Each instance is also associated with a latent variable h. Again, we adopt the example of learning a "car" model from a set of positive images containing cars and a set of negative images without cars. We know there is a car somewhere in a positive image, but we do not know its exact location. In this case, h can be used to represent the unobserved location of the car in the image. To simplify the notation, we also assume the latent variable h takes its value from a discrete set of labels $h \in \mathcal{H}$. However, our formulation is general. We will show how to deal with more complex h in Section 5.2.2 and in one of the experiments (Section 5.3.3).

In latent SVM, the scoring function of sample $\mathbf{x}$ is defined as $f_{\mathbf{w}}(\mathbf{x}) = \max_h \mathbf{w}^\top \phi(\mathbf{x}, h)$, where $\phi(\mathbf{x}, h)$ is the feature vector defined for the pair of $(\mathbf{x}, h)$. More formally, we are given M positive samples $\{\mathbf{x}_i\}_{i=1}^{M}$, and N negative samples $\{\mathbf{x}_j\}_{j=M+1}^{M+N}$. The learning of binary

latent SVM needs to solve the following quadratic program:

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C_1 \sum_i \xi_i + C_2 \sum_{j,h} \xi_{j,h} \tag{5.1a}$$

$$\text{s.t.} \quad \max_{h_i} \mathbf{w}^\top \phi(\mathbf{x}_i, h_i) \geq 1 - \xi_i, \quad \forall i \in \{1, 2, ..., M\}, \tag{5.1b}$$

$$-\mathbf{w}^\top \phi(\mathbf{x}_j, h) \geq 1 - \xi_{j,h} \quad \forall j \in \{M+1, M+2, ..., M+N\}, \forall h \in \mathcal{H} \tag{5.1c}$$

$$\xi_i \geq 0, \quad \xi_{j,h} \geq 0 \quad \forall i, \ \forall j, \ \forall h \in \mathcal{H} \tag{5.1d}$$

Similar to standard SVMs, $\{\xi_i\}$ and $\{\xi_{j,h}\}$ are the slack variables for handling soft margins. The first constraint (i.e. Eq. 5.1b) is imposed on the positive examples, while the second constraint (i.e. Eq. 5.1c) corresponds to the negative examples. Since we assume h's are not observed on negative images, we need to enumerate all possible values for h's in Eq. 5.1c. Intuitively, this means every image patch from a negative image (i.e. non-car image) is not a car. Because of the latent variables, latent SVM is essentially a non-convex optimization problem. However, the learning problem in Eq. 5.1 becomes convex once the latent variable h is fixed for positive examples. We can solve this problem by an iterative algorithm that alternates between inferring h on positive examples and optimizing the model parameter $\mathbf{w}$.

5.1.2 Dual form with fixed h on positive examples

Due to its nature of non-convexity, it is not straightforward to derive the dual form for the latent SVM. Therefore, as a starting point, we first consider a simpler scenario assuming h is fixed on the positive training examples. As previously mentioned, the latent SVM is then relaxed to a convex problem with this assumption. Note that we will relax this assumption in Sec. 5.2. In the above "car model" example, this means that we have the ground-truth bounding boxes of the cars in each image. The form of the positive samples become $\{\mathbf{x}_i, h_i\}_{i=1}^M$. After fixing the h on positive examples, the quadratic program in Eq. 5.1 changes to the following formulation:

$$\mathcal{P}(\mathbf{w}^*) = \min_{\mathbf{w},\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C_1 \sum_i \xi_i + C_2 \sum_{j,h} \xi_{j,h} \tag{5.2a}$$

$$\text{s.t.} \quad \mathbf{w}^\top \phi(\mathbf{x}_i, h_i) \geq 1 - \xi_i, \quad \forall i \in \{1, 2, ..., M\}, \tag{5.2b}$$

$$-\mathbf{w}^\top \phi(\mathbf{x}_j, h) \geq 1 - \xi_{j,h} \quad \forall j \in \{M+1, M+2, ..., M+N\}, \forall h \in \mathcal{H} \tag{5.2c}$$

$$\xi_i \geq 0, \quad \xi_{j,h} \geq 0 \quad \forall i, \ \forall j, \ \forall h \in \mathcal{H} \tag{5.2d}$$

It is easy to show that Eq. 5.2 is convex. Similar to the dual form of standard SVMs, we can derive the dual form of Eq. 5.2 as follows:

$$\mathcal{D}(\alpha^*, \beta^*) = \max_{\alpha,\beta} \sum_i \alpha_i + \sum_j \sum_h \beta_{j,h} - \frac{1}{2}||\sum_i \alpha_i \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h}\phi(\mathbf{x}_j, h)||^2 \quad (5.3a)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C_1, \quad \forall i; \quad 0 \leq \beta_{j,h} \leq C_2, \quad \forall j, \ \forall h \in \mathcal{H} \quad (5.3b)$$

The optimal primal parameters $\mathbf{w}^*$ for Eq. 5.2 and the optimal dual parameters (α^*, β^*) for Eq. 5.3 are related as follows:

$$\mathbf{w}^* = \sum_i \alpha_i^* \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h}^* \phi(\mathbf{x}_j, h) \quad (5.4)$$

Let us define λ to be the concatenations of $\{\alpha_i : \forall i\}$ and $\{\beta_{j,h} : \forall j, \forall h \in \mathcal{H}\}$, so $|\lambda| = M + N \times |\mathcal{H}|$. Let Ψ be a $|\lambda| \times D$ matrix where D is the dimension of $\phi(\mathbf{x}, h)$. Ψ is obtained by stacking together $\{\phi(\mathbf{x}_i, h_i) : \forall i\}$ and $\{-\phi(\mathbf{x}_j, h) : \forall j, \forall h \in \mathcal{H}\}$. We also define $Q = \Psi\Psi^\top$ and 1 to be a vector of all 1's. Then Eq. 5.3a can be rewritten as (we omit the linear constraints on λ for simplicity):

$$\max_\lambda \lambda^\top \cdot 1 - \frac{1}{2}\lambda^\top Q \lambda \quad (5.5)$$

The advantage of working with the dual form in Eq. 5.5 is that it only involves a so-called kernel matrix Q. Each entry of Q is a dot-product of two vectors in the form of $\phi(\mathbf{x}, h)^\top \phi(\mathbf{x}', h')$. We can replace the dot-product with any other kernel functions in the form of $k(\phi(\mathbf{x}, h), \phi(\mathbf{x}', h'))$ to get nonlinear classifiers [10]. The scoring function for the testing images $\mathbf{x}^{new}$ can be kernelized as follows:

$$f(\mathbf{x}^{new}) = \max_{h^{new}} \left(\sum_i \alpha_i^* k(\phi(\mathbf{x}_i, h_i), \phi(\mathbf{x}^{new}, h^{new})) - \sum_j \sum_h \beta_{j,h}^* k(\phi(\mathbf{x}_j, h), \phi(\mathbf{x}^{new}, h^{new})) \right) \quad (5.6)$$

Another important, yet often overlooked fact is that the optimal values of the two quadratic programs in Eqs. 5.2 and 5.3 have some specific meanings. They correspond to the inverse of the (soft) margin of the resultant SVM classifier [46, 93]:

$$\mathcal{P}(\mathbf{w}^*) = \mathcal{D}(\alpha^*, \beta^*) = \frac{1}{\text{SVM (soft) margin}}. \quad (5.7)$$

In the next section, we will exploit this fact to develop the kernel latent support vector machines.

5.2 Learning

Now we assume the variables $\{h_i\}_{i=1}^M$ on the positive training examples are unobserved. If the scoring function used for classification is in the form of $f(\mathbf{x}) = \max_h \mathbf{w}^\top \phi(\mathbf{x}, h)$, we can use the LSVM formulation [26, 97] to learn the model parameters $\mathbf{w}$. As mentioned earlier, the limitation of LSVM is the linearity assumption of $\mathbf{w}^\top \phi(\mathbf{x}, h)$. In this section, we propose kernel latent SVM (KLSVM) – a new latent variable learning method that only requires a kernel function $K(\mathbf{x}, h, \mathbf{x}', h')$ between a pair of $(\mathbf{x}, h)$ and $(\mathbf{x}', h')$.

Note that when $\{h_i\}_{i=1}^M$ are observed on the positive training examples, we can plug them in Eq. 5.3 to learn a nonlinear kernelized decision function that separates the positive and negative examples. When $\{h_i\}_{i=1}^M$ are latent, an intuitive thing to do is to find the labeling of $\{h_i\}_{i=1}^M$ so that when we plug them in and solve for Eq. 5.3, the resultant nonlinear decision function separates the two classes as widely as possible. In other words, we look for a set of $\{h_i^*\}$ which can further maximize the SVM margin (equivalent to minimizing $\mathcal{D}(\alpha^*, \beta^*)$). The same intuition was previously used to develop the max-margin clustering method in [93]. Using this intuition, we write the optimal function value of the dual form as $\mathcal{D}(\alpha^*, \beta^*, \{h_i\})$ since now it implicitly depends on the labeling $\{h_i\}$. We can jointly find the labeling $\{h_i\}$ and solve for (α^*, β^*) by the following optimization problem:

$$\min_{\{h_i\}} \mathcal{D}(\alpha^*, \beta^*, \{h_i\}) \tag{5.8a}$$

$$= \min_{\{h_i\}} \max_{\alpha,\beta} \quad \sum_i \alpha_i + \sum_j \sum_h \beta_{j,h} - \frac{1}{2} || \sum_i \alpha_i \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h} \phi(\mathbf{x}_j, h) ||^2 \tag{5.8b}$$

$$\text{s.t.} \;\; 0 \leq \alpha_i \leq C_1, \quad \forall i; \qquad 0 \leq \beta_{j,h} \leq C_2, \quad \forall j, \; \forall h \in \mathcal{H} \tag{5.8c}$$

The most straightforward way of solving Eq. 5.8 is to optimize $\mathcal{D}(\alpha^*, \beta^*, \{h_i\})$ for every possible combinations of values for $\{h_i\}$, and then take the minimum. When h_i takes its value from a discrete set of K possible choices (i.e. $|\mathcal{H}| = K$), this naive approach needs to solve M^K quadratic programs. This is obviously too expensive. Instead, we use the following iterative algorithm to solve Eq. 5.8 .

- Fix α and β, compute the optimal $\{h_i\}^*$ by

$$\{h_i\}^* = \arg\max_{\{h_i\}} \frac{1}{2} || \sum_i \alpha_i \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h} \phi(\mathbf{x}_j, h) ||^2 \tag{5.9}$$

- Fix $\{h_i\}$, compute the optimal (α^*, β^*) by

$$(\alpha^*, \beta^*) = \arg\max_{\alpha,\beta} \left\{ \sum_i \alpha_i + \sum_j \sum_h \beta_{j,h} - \frac{1}{2} || \sum_i \alpha_i \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h} \phi(\mathbf{x}_j, h) ||^2 \right\} \tag{5.10}$$

The optimization problem in Eq. 5.10 is a quadratic program similar to that of a standard dual SVM. As a result, Eq. 5.10 can be kernelized as Eq. 5.5 and solved using standard dual solver in regular SVMs, and its complexity is estimated to be quadratic in the number of training examples [8]. In Sec. 5.2.1, we describe how to kernelize and solve the optimization problem in Eq. 5.9.

5.2.1 Optimization over $\{h_i\}$

The complexity of a simple enumeration approach for solving Eq. 5.9 is again $O(M^K)$, which is clearly too expensive for practical purposes. Instead, we solve it iteratively using an algorithm similar to co-ordinate ascent. Within an iteration, we choose one positive training example t. We update h_t while fixing h_i for all $i \neq t$. The optimal h_t^* can be computed as follows:

$$h_t^* = \arg\max_{h_t} ||\alpha_t \phi(\mathbf{x}_t, h_t) + \sum_{i:i\neq t} \alpha_i \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h} \phi(\mathbf{x}_j, h)||^2 \tag{5.11a}$$

$$\Leftrightarrow \arg\max_{h_t} ||\alpha_t \phi(\mathbf{x}_t, h_t)||^2 + 2 \left(\sum_{i:i\neq t} \alpha_i \phi(\mathbf{x}_i, h_i) - \sum_j \sum_h \beta_{j,h} \phi(\mathbf{x}_j, h) \right)^\top \alpha_t \phi(\mathbf{x}_t, h_t) \tag{5.11b}$$

By replacing the dot-product $\phi(\mathbf{x}, h)^\top \phi(\mathbf{x}', h')$ with a kernel function $k(\phi(\mathbf{x}, h), \phi(\mathbf{x}', h'))$, we obtain the kernerlized version of Eq. 5.11(b) as follows

$$\begin{aligned} h_t^* = \arg\max_{h_t} \quad & \alpha_t \alpha_t k(\phi(\mathbf{x}_t, h_t), \phi(\mathbf{x}_t, h_t)) + 2 \sum_{i:i\neq t} \alpha_i \alpha_t k(\phi(\mathbf{x}_i, h_i), \phi(\mathbf{x}_t, h_t)) \\ & -2 \sum_j \sum_h \beta_{j,h} \alpha_t k(\phi(\mathbf{x}_j, h), \phi(\mathbf{x}_t, h_t)) \end{aligned} \tag{5.12}$$

It is interesting to notice that if the t-th example is not a support vector (i.e. $\alpha_t = 0$), the function value of Eq. 5.12 will be zero regardless of the value of h_t. This means in KLSVM we can improve the training efficiency by only performing Eq. 5.12 on positive examples

corresponding to support vectors. For other positive examples (non-support vectors), we can simply set their latent variables the same as the previous iteration. Note that in LSVM, the inference during training needs to be performed on every positive example. For discrete latent variables, the complexity of the optimization problem over $\{h_i\}$ in KLSVM is only $O(S_pK)$, where S_p is the number of positive support vectors and K is the number of possible choices of latent variables (i.e. $|\mathcal{H}| = K$).

Connection to LSVM: When a linear kernel is used, the inference problem (Eq. 5.11) has a very interesting connection to LSVM in [26]. Recall that for linear kernels, the model parameters w and dual variables (α, β) are related by Eq. 5.4. Then Eq. 5.11 becomes:

$$h_t^* = \arg\max_{h_t} ||\alpha_t \phi(\mathbf{x}_t, h_t)||^2 + 2\left(\mathbf{w} - \alpha_t \phi(\mathbf{x}_t, h_t^{\text{old}})\right)^\top \alpha_t \phi(\mathbf{x}_t, h_t) \tag{5.13a}$$

$$\Leftrightarrow \arg\max_{h_t} \alpha_t \mathbf{w}^\top \phi(\mathbf{x}_t, h_t) + \frac{1}{2}\alpha_t^2 ||\phi(\mathbf{x}_t, h_t)||^2 - \alpha_t^2 \phi(\mathbf{x}_t, h_t^{\text{old}})^\top \phi(\mathbf{x}_t, h_t) \tag{5.13b}$$

where h_t^{old} is the value of latent variable of the t-th example in the previous iteration. Let us consider the situation when $\alpha_t \neq 0$ and the feature vector $\phi(\mathbf{x}, h)$ is l_2 normalized, which is commonly used in computer vision. In this case, $\alpha_t^2 \phi(\mathbf{x}_t, h_t)^\top \phi(\mathbf{x}_t, h_t)$ is a constant, and we have $\phi(\mathbf{x}_t, h_t^{\text{old}})^\top \phi(\mathbf{x}_t, h_t^{\text{old}}) > \phi(\mathbf{x}_t, h_t^{\text{old}})^\top \phi(\mathbf{x}_t, h_t)$ if $h_t \neq h_t^{\text{old}}$. Then Eq. 5.13 is equivalent to:

$$h_t^* = \arg\max_{h_t} \mathbf{w}^\top \phi(\mathbf{x}_t, h_t) - \alpha_t \phi(\mathbf{x}_t, h_t^{\text{old}})^\top \phi(\mathbf{x}_t, h_t) \tag{5.14}$$

Eq. 5.14 is very similar to the inference problem in LSVM, i.e., $h_t^* = \arg\max_{h_t} \mathbf{w}^\top \phi(\mathbf{x}_t, h_t)$, but with an extra term $\alpha_t \phi(\mathbf{x}_t, h_t^{\text{old}})^\top \phi(\mathbf{x}_t, h_t)$ which penalizes the choice of h_t for being the same value as previous iteration h_t^{old}. This has a very appealing intuitive interpretation. If the t-th positive example is a support vector, the latent variable h^{old} from previous iteration causes this example to lie very close to (or even on the wrong side) the decision boundary, i.e. the example is not well-separated. During the current iteration, the second term in Eq. 5.14 penalizes h^{old} to be chosen again since we already know the example will not be well-separated if we choose h^{old} again. The amount of penalty depends on the magnitudes of α_t and $\phi(\mathbf{x}_t, h_t^{old})^\top \phi(\mathbf{x}_t, h_t)$. We can interpret α_t as how "bad" h_t^{old} is, and $\phi(\mathbf{x}_t, h_t^{old})^\top \phi(\mathbf{x}_t, h_t)$ as how close h_t is to h_t^{old}. Eq. 5.14 penalizes the new h_t^* to be "close" to "bad" h_t^{old}.

5.2.2 Composite Kernels

So far we have assumed that the latent variable h takes its value from a discrete set of labels. Given a pair of $(\mathbf{x}, h)$ and $(\mathbf{x}', h')$, the types of kernel function $k(\mathbf{x}, h; \mathbf{x}', h')$ we can choose from are still limited to a handful of standard kernels (e.g. Gaussian, RBF, HIK, etc). In this section, we consider more interesting cases where h involves some complex structures. This will give us two important benefits. First of all, it allows us to exploit structural information in the latent variables. This is in analog to structured output learning (e.g. [75, 79]). More importantly, it gives us more flexibility to construct new kernel functions by composing from simple kernels.

Before we proceed, let us first motivate the composite kernel with an example application. Suppose we want to detect some complex person-object interaction (e.g. "person riding a bike") in an image. One possible solution is to detect persons and bikes in an image, then combine the results by taking into account of their relationship (i.e. "riding"). Imagine we already have kernel functions corresponding to some components (e.g. person, bike) of the interaction. In the following, we will show how to compose a new kernel for the "person riding a bike" classifier from those components.

We denote the latent variable using $\vec{h}$ to emphasize that now it is a vector instead of a single discrete value. We denote it as $\vec{h} = (z_1, z_2, ...)$, where z_u is the u-th component of $\vec{h}$ and takes its value from a discrete set of possible labels. For the structured latent variable, it is assumed that there are certain dependencies between some pairs of (z_u, z_v). We can use an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to capture the structure of the latent variable, where a vertex $u \in \mathcal{V}$ corresponds to the label z_u, and an edge $(u, v) \in \mathcal{E}$ corresponds to the dependency between z_u and z_v. As a concrete example, consider the "person riding a bike" recognition problem. The latent variable in this case has two components $\vec{h} = (z_{person}, z_{bike})$ corresponding to the location of person and bike, respectively. On the training data, we have access to the ground-truth bounding box of "person riding a bike" as a whole, but not the exact location of "person" or "bike" within the bounding box. So $\vec{h}$ is latent in this application. The edge connecting z_{person} and z_{bike} captures the relationship (e.g. "riding on", "next to", etc.) between these two objects.

Suppose we already have kernel functions corresponding to the vertices and edges in the graph, we can then define the composite kernel as the summation of the kernels over all the

vertices and edges.

$$K(\Phi(\mathbf{x},\vec{h}),\Phi(\mathbf{x}',\vec{h}')) = \sum_{u\in\mathcal{V}} k_u(\phi(\mathbf{x},z_u),\phi(\mathbf{x}',z'_u)) + \sum_{(u,v)\in\mathcal{E}} k_{uv}(\psi(\mathbf{x},z_u,z_v),\psi(\mathbf{x}',z'_u,z'_v)) \tag{5.15}$$

It is also possible for Eq. 5.15 to include kernels defined on higher-order cliques in the graph, as long as we have some pre-defined kernel functions for them.

5.3 Experiments

We evaluate KLSVM in three different applications of visual recognition. Each application has a different type of latent variables. For these applications, we will show that KLSVM outperforms both the linear LSVM [26] and the regular kernel SVM. Note that we implement the learning of linear LSVM by ourselves using the same iterative algorithm as the one in [26].

5.3.1 Object Classification with Latent Localization

Problem and Dataset: We consider object classification with image-level supervision. Our training data only have image-level labels indicating the presence/absence of each object category in an image. The exact object location in the image is not provided and is considered as the latent variable h in our formulation. We define the feature vector $\phi(x,h)$ as the HOG feature extracted from the image at location h. During testing, the inference of h is performed by enumerating all possible locations of the image.

We evaluate our algorithm on the mammal dataset [42] which consists of 6 mammal categories. There are about 45 images per category. For each category, we use half of the images for training and the remaining half for testing. We assume the object size is the same for the images of the same category, which is a reasonable assumption for this dataset. This dataset was used to evaluate the linear LSVM in [42].

Results: We compare our algorithm with linear LSVM. To demonstrate the benefit of using latent variables, we also compare with two simple baselines using linear and kernel SVMs based on bag-of-features (BOF) extracted from the whole image (i.e. without latent variables). For both baselines, we aggregate the quantized HOG features densely sampled from the whole image. Then, the features are fed into the standard linear SVM and kernel SVM respectively. We use the histogram intersection kernel (HIK) [53] since it has been

proved to be successful for vision applications, and efficient learning/inference algorithms exist for this kernel.

We run the experiments for five rounds. In each round, we randomly split the images from each category into training and testing sets. For both linear LSVM and KLSVM, we initialize the latent variable at the center location of each image and we set $C_1 = C_2 = 1$. For both algorithms, we use one-versus-one classification scheme. we use HIK in KLSVM. We use both linear dot-product kernel and HIK in KLSVM. Table 5.1 summarizes the mean and standard deviations of the classification accuracies over five rounds of experiments. Fig. 5.1 shows examples of how the latent variables change on some training images during the learning of the KLSVM (using HIK). For each training image, the location of the object (latent variable h) is initialized to the center of the image. After the learning algorithm terminates, the latent variables accurately locate the objects.

Across all experiments, both linear LSVM and KLSVM achieve significantly better results than approaches using BOF features from the whole image. This is intuitively reasonable since most of images on this dataset share very similar scenes. So BOF feature without latent variables cannot capture the subtle differences between each category. Table 5.1 shows that both KLSVM with linear kernel and KLSVM with HIK significantly outperform linear LSVM. Furthermore, we find out that in the learning of linear LSVM, the latent variables (i.e. object location) on positive images barely move from their initial positions during iterations. But in KLSVM (for both linear kernel and HIK), the latent variables are usually able to locate the objects in first a few iterations, as shown in Fig. 5.1. This would suggest that KLSVM is better at avoiding the bad local minimum. Note that the learning algorithm of KLSVM with linear kernel is very similar to the linear LSVM. The difference is that KLSVM with linear kernel uses Eq. 5.13 for the inference, while linear LSVM uses $h_t^* = \arg\max_{h_t} \mathbf{w}^\top \phi(\mathbf{x}_t, h_t)$. In Section 5.2.1, we show that Eq. 5.13 contains an extra penalty term which penalizes the choice of h_t for being the same value as previous iteration. Therefore, we believe this penalty term is a key factor in KLSVM for avoiding the bad local minimum and achieving better performance in this task. Table 5.1 also shows that the KLSVM using linear kernel achieves the similar performance to the KLSVM using HIK. We believe it is reasonable since the mammal dataset is relatively simple and small (only 45 images per category). If the objects in the image can be localized accurately, it is possible that the benefits of using HIK are not well demonstrated on such a small dataset.

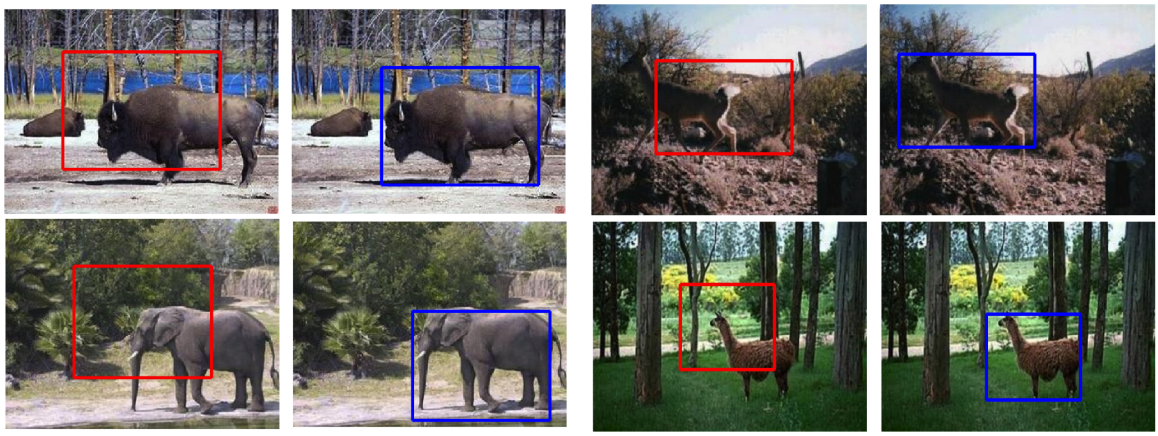


Figure 5.1: Visualization of how the latent variable (i.e. object location) changes during the learning. The red bounding box corresponds to the initial object location. The blue bounding box corresponds to the object location after the learning.

Method	linear SVM	kernel SVM	linear LSVM	KLSVM (linear)	KLSVM (HIK)
Acc (%)	45.57 ± 4.23	50.53 ± 6.53	75.07 ± 4.18	$\mathbf{84.80 \pm 5.37}$	$\mathbf{84.49 \pm 3.63}$

Table 5.1: Results on the mammal dataset. We show the mean/std of classification accuracies over five rounds of experiments.

5.3.2 Object Classification with Latent Subcategory

Problem and Dataset: Our second application is also on object classification. But here we consider a different type of latent variable. Objects within a category usually have a lot of intra-class variations. For example, consider the images for the "bird" category shown in the left column of Fig. 5.2. Even though they are examples of the same category, they still exhibit very large appearance variations. It is usually very difficult to learn a single "bird" model that captures all those variations. One way to handle the intra-class variation is to split the "bird" category into several subcategories. Examples within a subcategory will be more visually similar than across all subcategories. Here we use the latent variable h to indicate the subcategory an image belongs to. If a training image belongs to the class c, its subcategory label h takes value from a set $\mathcal{H}_c$ of subcategory labels corresponding to the c-th class. Note that subcategories are latent on the training data, so they may or may not have semantic meanings.

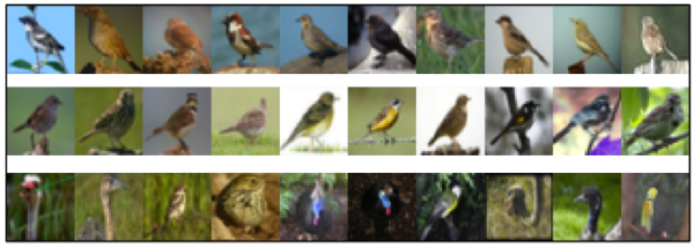
The feature vector $\phi(x, h)$ is defined as a sparse vector whose feature dimension is $|\mathcal{H}_c|$ times of the dimension of $\phi(x)$, where $\phi(x)$ is the HOG descriptor extracted from the image x.

In the experiments, we set $|\mathcal{H}_c| = 3$ for all c's. Then we can define $\phi(x, h = 1) = (\phi(x); \mathbf{0}; \mathbf{0})$, $\phi(x, h = 2) = (\mathbf{0}; \phi(x); \mathbf{0})$, and so on. Similar models have been proposed to address the viewpoint changing in object detection [31] and semantic variations in YouTube video tagging [94].

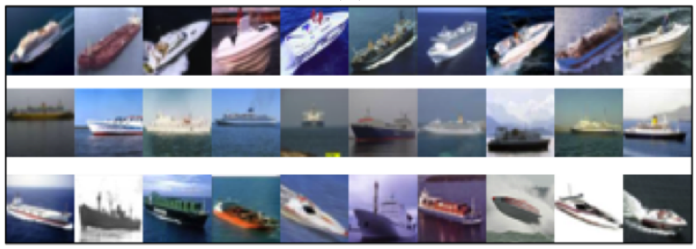
We use the CIFAR10 [41] dataset in our experiment. It consists of images from ten classes including airplane, automobile, bird, cat, etc. The training set has been divided into five batches and each batch contains 10000 images. There are in total 10000 test images.

Results: Again we compare with three baselines: linear LSVM, non-latent linear SVM, non-latent kernel SVM. We use HIK in the non-latent kernel SVM, and we use both linear dot-product kernel and HIK in KLSVM. For non-latent approaches, we simply feed feature vector $\phi(x)$ to SVMs without using any latent variable.

We run the experiments in five folds. Each fold use a different training batch but the same testing batch. We set $C_1 = C_2 = 0.01$ for all the experiments and initialize the subcategory labels of training images by k-means clustering. Table 5.2 summarizes the results. Again, KLSVM outperforms other baseline approaches. It is interesting to note that both linear LSVM and KLSVM outperform their non-latent counterparts, which demonstrates the effectiveness of using latent subcategories in object classification. However, the KLSVM using linear kernel does not outperform the linear LSVM. This result seems to be in contradiction to what we have observed in Section 5.3.1. We believe it is because the number of subcategories is too small, e.g. $|\mathcal{H}_c| = 3$. For example, for the t-th example, if the value of its latent variable in previous iteration equals to 1, i.e. $h_t^{old} = 1$, which causes this example to lie very close to the decision boundary, the penalty term (see Eq. 5.14) will reward the choice of h_t in current iteration for being either 2 or 3. However, since there are only two choices for h_t, it is possible that $h_t = 2$ or $h_t = 3$ might not be the optimal choice for the t-th example. As noted in [18], the number of subcategories is an important parameter in the latent subcategory learning, and it may significantly influence the classification performance. Therefore, we believe that the performance of KLSVM would be further improved if we set the number of subcategories to a larger value. We visualize examples of the correctly classified testing images from the "bird" and "boat" categories in Fig. 5.2. Images on the same row are assigned the same subcategory labels. We can see that visually similar images are automatically grouped into the same subcategory.



(a)



(b)

Figure 5.2: Visualization of some testing examples from the "bird" (a) and "boat" (b) categories. Each row corresponds to a subcategory. We can see that visually similar images are grouped into the same subcategory.

Method	linear SVM	linear LSVM	kernel SVM (HIK)	KLSVM (linear)	KLSVM (HIK)
Acc (%)	50.69 ± 0.38	53.13 ± 0.63	52.98 ± 0.22	$\mathbf{52.0 \pm 0.53}$	$\mathbf{55.17 \pm 0.27}$

Table 5.2: Results on CIFAR10 Dataset. We show the mean/std of classification accuracies over five folds of experiments. Each fold uses a different batch of the training data.

5.3.3 Recognition of Object Interaction

Problem and Dataset: Finally, we consider an application where the latent variable is more complex and requires the composite kernel introduced in Sec. 5.2.2. We would like to recognize complex interactions between two objects (also called "visual phrases" [65]) in static images. We build a dataset consisting of four object interaction classes, i.e. "person riding a bicycle", "person next to a bicycle", "person next to a car" and "bicycle next to a car" based on the visual phrase dataset in [65]. Each class contains 86∼116 images. Each image is only associated with one of the four object interaction label. There is no ground-truth bounding box information for each object. We use 40 images from each class for training and the rest for testing.

Our approach: We treat the locations of objects as latent variables. For example, when learning the model for "person riding a bicycle", we treat the locations of "person" and "bicycle" as latent variables. In this example, each image is associated with latent variables $\vec{h} = (z_1, z_2)$, where z_1 denotes the location of the "person" and z_2 denotes the location of the "bicycle". To reduce the search space of inference, we first apply off-the-shelf "person" and "bicycle" detectors [26] on each image. For each object, we generate five candidate bounding boxes which form a set $\mathcal{Z}_i$, i.e. $|\mathcal{Z}_1| = |\mathcal{Z}_2| = 5$ and $z_i \in \mathcal{Z}_i$. Then, the inference of $\vec{h}$ is performed by enumerating 25 combinations of z_1 and z_2. We also assume there are certain dependencies between the pair of (z_1, z_2). Then the kernel between two images can be defined as follows:

$$K(\Phi(\mathbf{x}, \vec{h}), \Phi(\mathbf{x}', \vec{h}')) = \sum_{u=\{1,2\}} k_u\left(\phi(\mathbf{x}, z_u), \phi(\mathbf{x}', z'_u)\right) + k_p\left(\psi(z_1, z_2), \psi(z'_1, z'_2)\right) \quad (5.16)$$

We define $\phi(\mathbf{x}, z_u)$ as the bag-of-features (BOF) extracted from the bounding box z_u in the image x. For each bounding box, we split the region uniformly into four equal quadrants. Then we compute the bag-of-features for each quadrant by aggregating quantized HOG features. The final feature vector is the concatenation of these four bag-of-features histograms. This feature representation is similar to the spatial pyramid feature representation. In our experiment, we choose HIK for $k_u(\cdot)$. The kernel $k_p(\cdot)$ captures the spatial relationship between z_1 and z_2 such as above, below, overlapping, next-to, near, and far. Here $\psi(z_1, z_2)$ is a sparse binary vector and its k-th element is set to 1 if the corresponding k-th relation is satisfied between bounding boxes z_1 and z_2. Note that $k_p(\cdot)$ does not depend on the images. Similar representation has been used in [17]. We define $k_p(\cdot)$ as a simple linear kernel.

Method	BOF + linear SVM	BOF + kernel SVM	linear LSVM	KLSVM
Acc(%)	42.92	58.46	46.33 ± 1.4	$\mathbf{66.42 \pm 0.99}$

Table 5.3: Results on object interaction dataset. For the approaches using latent variables, we show the mean/std of classification accuracies over five folds of experiments.

Results: We compare with the simple BOF + linear SVM, and BOF + kernel SVM approaches. These two baselines use the same BOF feature representation as our approach except that the features are extracted from the whole image. We choose the HIK in the kernel SVM. Note that this is a strong baseline since [15] has shown that a similar pyramid feature representation with kernel SVM achieves top performances on the task of person-object interaction recognition. The other baseline is the standard linear LSVM, in which we build the feature vector $\phi(x, h)$ by simply concatenating both unary features and pairwise features, i.e. $\phi(x, h) = [\phi(x, z_1); \phi(x, z_2); \psi(z_1, z_2)]$. Again, we set $C_1 = C_2 = 1$ for all experiments. We run the experiments for five rounds for approaches using latent variables. In each round, we randomly initialize the choices of z_1 and z_2. Table 5.3 summarizes the results. The kernel latent SVM that uses HIK for $k_u(\cdot)$ achieves the best performance.

Fig. 5.3 shows examples of how the latent variables change on some training images during the learning of the KLSVM. For each training image, both latent variables z_1 and z_2 are randomly initialized to one of five candidate bounding boxes. As we can see, the initial bounding boxes can accurately locate the target objects but their spatial relations are different to ground-truth labels. After learning algorithm terminates, the latent variables not only locate the target objects, but more importantly they also capture the correct spatial relationship between objects.

5.4 Summary

We have proposed kernel latent SVM – a new learning framework that combines the benefits of LSVM and kernel methods. Our learning framework is very general. The latent variables can not only be a single discrete value, but also be more complex values with interdependent structures. Our experimental results on three different applications in visual recognition demonstrate that KLSVM outperforms using LSVM or using kernel methods

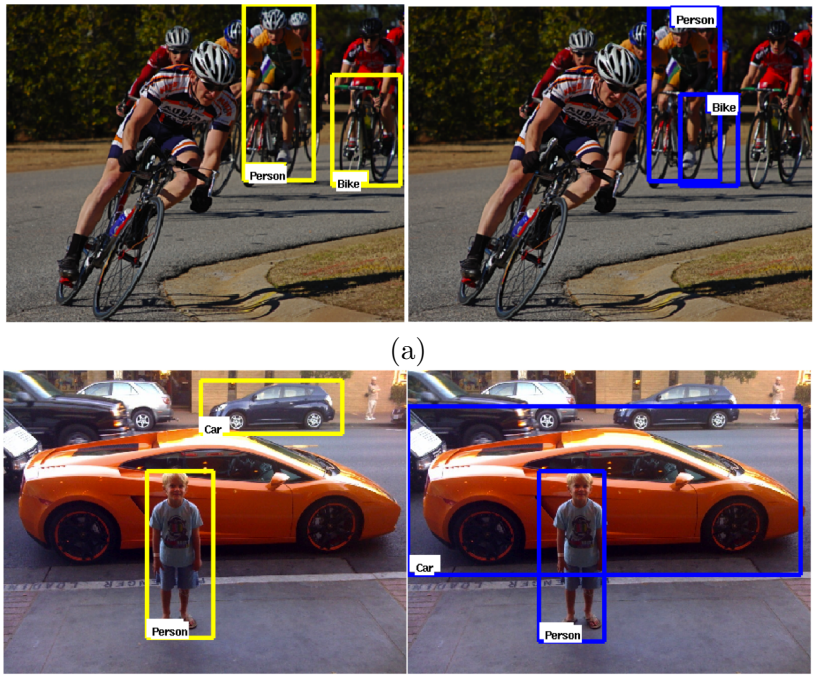


Figure 5.3: Visualization of how latent variables (i.e. object locations) change during the learning. The top image is from the "person riding a bicycle" category, and the bottom image is from the "person next to a car" category. Yellow bounding boxes corresponds to the initial object locations. The blue bounding boxes correspond to the object locations after the learning.

alone. We believe our work will open the possibility of constructing more powerful and expressive prediction models for visual recognition. As future work, we plan to further investigate applications of this framework. We would also like to develop new learning/inference algorithms that allow us to handle more complex forms of latent variables.

Chapter 6

Video Tagging with Latent Sub-tag

In this chapter, we consider the problem of content-based automated tag learning. In particular, we address semantic variations (*sub-tags*) of the tag using the latent variable model. Each video in the training set is assumed to be associated with a sub-tag label, and we treat this sub-tag label as latent information. A latent learning framework based on LogitBoost is proposed, which jointly considers both the tag label and the latent sub-tag label. The latent sub-tag information is exploited in our framework to assist the learning of our end goal, i.e. tag prediction.

The rest of this chapter is organized as follows. An overview of our approach is provided in Section 6.1, followed by a brief review of "tag" related work in Section 6.2. We give the model formulation in Section 6.3, and the learning algrithm in Section 6.4. The details of the video features are introduced in Section 6.5. Lastly, the experimental results on a large-scale testing video set are presented in Section 6.6. Section 6.7 concludes this chapter.

6.1 Overview

On the Internet, the term "tag" refers to keywords assigned to an article, image, or video. With the rapid development of social sharing websites (i.e. Flickr, Picasa, and YouTube), the tags help organize, browse and search relevant items within these massive multimedia collections. In this chapter, we are particularly interested in the problem of content-based automated tag learning/prediction. Given a video, an automated tag learning/prediction system would be able to predict the tags associated to the video based on its content. Such a system would ensure that videos are properly tagged, and therefore enforce uniformity in

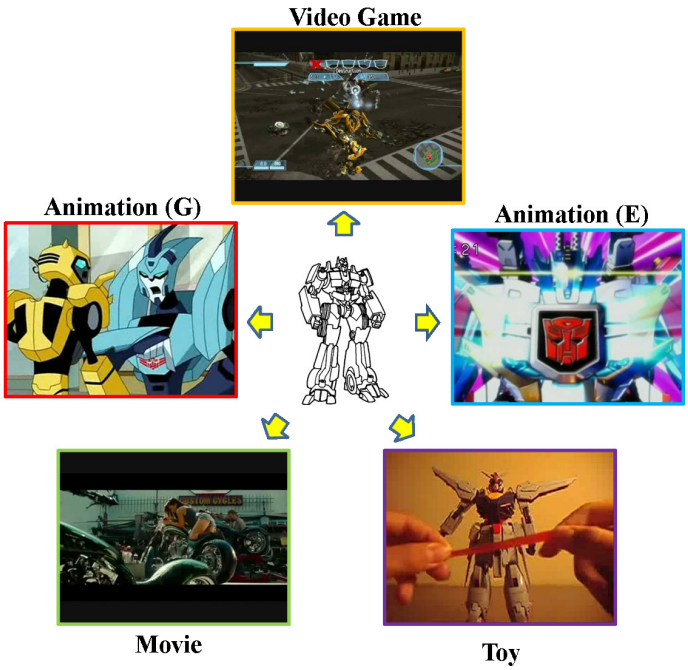


Figure 6.1: For the rather specific tag "transformers", the videos of this tag have large variations in video types and contexts. Those videos can be roughly grouped as video games, two different types of animations, toys, and movies. We use the notation "sub-tags" to refer to those semantic variants and treat them as latent information in our learning framework.

tagging. Uniformity is useful since users tend to not tag every possible aspect of the media, leading to a large number of undertagged objects. Having a uniform, automated tagging system would allow users to specify queries based on the known tag vocabulary and have an increased confidence that the results retrieved should be consistent, and more complete than otherwise possible, while allowing currently undertagged media to be considered for retrieval. Another application of the tag learning could be verifying that the users have typed a specific tag with the intent of annotating the video as opposed to simply spamming the index of the search engine.

Our work is under a similar framework to [3]. We generate a set of tags by counting the unique user-supplied tags and n-grams in the title for each video on YouTube. For each given tag, a classifier model is trained over a positive training set containing the videos ($20K$-$100K$) whose user-provided metadata text contains the given tag. The negative training set is sampled randomly from videos which do not contain the tag in their metadata. In this

way, both of the tag set and the training data are automatically generated from the online video corpora. In particular, since the tag set is generated from user-provided title/tags, the tags can better reflect the overall user tagging behaviors compared to a manually pre-defined tag set. In the tag set, we observe that some tags are too general (e.g. "video", "music", "youtube"), and some others are too specific (e.g. the name of the video uploader). Those tags do not contain any useful information. Therefore we discard those tags which are either too frequent or too infrequent.

After examining the remaining tags, we observe that for most of tags, there is a large variation within the videos which have such a tag, though those tags are rather specific. Interestingly, for a particular tag, the variation of its associated videos is not trivial, and those videos can be clustered into a few groups which may have some semantic meanings. For example, as shown in Fig. 6.1, the videos with tag "transformers" consists of "video games", "animations", "toys" and "movies". Although all those videos contain the tag "transformers", the video types and contexts are different. Another example is the tag "bike", which is a simple object. As shown in Fig. 6.2, the "bike" videos may consist of videos about "mountain bike", "falling from bike", "pocket bike", and "motorbike". The variations among "bike" videos include scene variation (mountain *vs.* road), object variation (bicycle *vs.* motorbike *vs.* pocket bike), and event variation (falling from bike *vs.* riding a bike). In the context of video analysis, we suspect that off-the-shelf classifiers cannot handle these non-trivial variations.

In this work, we use the notation of *sub-tag* to refer to the semantic variants of the given tag label. In particular, for a given tag, we define a sub-tag as a group of videos which not only have such a tag, but also share the same theme which is usually semantically interpretable. We propose a novel framework that can jointly learn both tag label and its sub-tag labels in a unified framework. We believe introducing the sub-tag label to the learning process will help to capture more of the variations of the tag, thereby improving the quality of the final classifier.

Sub-tags are conceptually related to a hierarchy in which the parent tag would have a general meaning and the child sub-tag is more specific. However, unlike such hierarchies (e.g. ImageNet or WordNet), the sub-tag does not necessarily need to be related by a relationship which can be expressed in terms of rigid ontology. Besides, it is very difficult and costly to define (or name) the sub-tag set for all possible tag labels, especially for the YouTube videos. Naming sub-tags requires a comprehensive knowledge of video types, and

user behaviors on YouTube. Moreover, in a dynamic video collection, more and more new tags may be invented and become popular, while the meanings of existing tags may vary as well.

The main contributions of the work presented in this chapter are two-fold. First, instead of explicitly defining or naming the sub-tags, we propose to treat sub-tags as latent information and use it to assist the task of tag learning. Inspired by latent SVM [26], we propose a novel latent learning framework based on LogitBoost, which jointly considers both the latent sub-tag label and the tag label. In LogitBoost, we use decision stumps as the weak learner, which introduce the non-linearity into the model and also allow us to handle complex feature vectors that consist of different feature types. To deal with the noisy samples in our training data, we use a bootstrapping scheme which can be naturally combined into our learning framework. In each iteration, the model is only trained on a training subset that contains "trustworthy" positive samples. Secondly, likewise to other non-convex learning problems, a proper initialization of the latent sub-tag label is very crucial to our learning framework. We propose to use the cowatch-based clustering scheme for initialization. The similarity between two videos is measured by cowatch statistics. Conceptually, if two videos are watched one after the other in a short period of time (cowatched), they are usually related and likely to be similar. However, our learning framework is general, and it is not limited to this type of initialization scheme.

6.2 Related Work

In the computer vision literature, we can roughly categorize the "tag"-related work into two lines: (1) leveraging the tagged image/video content on the Internet to improve the performance of learning-based image or object recognition systems; (2) content-based image/video tag prediction.

With the help of image search engines and photo hosting websites, researchers built more and more computer vision datasets by downloading images or videos, e.g. ImageNet [16], and 80 Million Tiny Images [78]. To decrease the label noise, additional human annotations are employed in most datasets to prune out the noisy samples. It is interesting to note that in [78], a reasonable recognition performance is obtained despite the high labeling noise. Fan et al. [22] also show that more effective classifiers can be obtained after pruning out the noisy tags by an image clustering approach based on both visual and tagging similarities

between images. Hwang and Grauman [35] assume that the prominence of an object in an image can be revealed by its order of mention in the tag list. They show that leveraging the information of tag ordering can improve the performance of object detection. In our approach, we also train our tag models on the videos collected from YouTube. For simplicity, we train a model for each tag and ignore the structure information among tags.

Recently, there has been an increased interest in automated image/video tagging. A variety of methods are proposed for image tagging, e.g. kernel canonical correlation analysis [74], hierarchical generative models [49], latent SVM [89], etc. There is also a lot of research work about video tagging (e.g. [73]), and the proposed methods have been evaluated on the TRECVID dataset for the task of semantic indexing. A list of papers can be found in [57]. A few systems are built for the automated YouTube video tagging based on video content. Ulges et al. [80] predict the conceptual tags for the keyframes of the video by the combination of three different classifiers. Toderici et al. [77] and Aradhye et al. [3] learn the tags of a video using AdaBoost based on both video and audio features. Our work is based on [77, 3], but we are more interested in the problem of semantic variations (sub-tags) in the YouTube videos that share the same tag. A latent learning framework is proposed to address this issue and we treat sub-tag labels as latent variables in our system.

6.3 Model Formulation

Given a tag label z and a training set $\mathcal{T}$ which consists of N training samples $\mathcal{T} = \{(V^n, y^n)\}_{n=1}^N$, our goal is to learn a scoring function for the tag label z: $f_z(V)$, which can return a confidence score of z being assigned to the video V. $y \in \{+1, -1\}$ takes value 1 if the video V has the tag label z, and -1 otherwise. In our model, each video is associated with a sub-tag label h, where $h \in \mathcal{H}_z$ and $\mathcal{H}_z$ is the set of sub-tags for the original tag z. We treat h as latent information since we do not have sub-tag labels during training. We assume the scoring function takes the following form:

$$\begin{aligned} f_z(V) &= \max_{h \in \mathcal{H}_z} \Psi(V, h; \mathcal{F}_z); \\ \Psi(V, h; \mathcal{F}_z) &= \sum_{b \in \mathcal{H}_z} 1_b(h) \cdot F_b(V), \end{aligned} \tag{6.1}$$

where $1_b(h)$ is an indicator that takes the value 1 if $h = b$, and 0 otherwise. $F_b(\cdot)$ is a LogitBoost [67] classifier learned from the training samples which share the same sub-tag

label $h = b$. We denote $\mathcal{F}_z$ as a set of LogitBoost classifiers for the tag z, i.e. $\mathcal{F}_z = \{F_b : b \in \mathcal{H}_z\}$. The details concerning training $\mathcal{F}_z$ will be discussed in Section 6.4.

Given $\mathcal{F}_z$ and a testing video V, we need to solve an inference problem of finding the best sub-tag label h^* as follows:

$$h^* = \arg\max_{h \in \mathcal{H}_z} \Psi(V, h; \mathcal{F}_z). \tag{6.2}$$

This inference problem can be easily solved by enumerating all the possible sub-tag labels $h \in \mathcal{H}_z$. We are more interested in improving the performance of video classification at tag level, though we do obtain the sub-tag label h^* as a by-product for each testing video.

6.4 Learning

Different from other learning algorithms with structured output, we simply assume each tag label is independent of each other and we train a model for each tag label. The optimal $\mathcal{F}_z^*$ could be learned as follows:

$$\mathcal{F}_z^* = \arg\min_{\mathcal{F}_z} \sum_n^N l(y^n, \max_{h \in \mathcal{H}_z} \Psi(V^n, h; \mathcal{F}_z)), \tag{6.3}$$

where $l(\cdot)$ is the loss function. We use the convex logistic loss $l(y, \Psi) = \log(1 + \exp(-2y\Psi))$ in our learning algorithm. Similar to the latent SVM [26], this problem is not convex but we could use an iterative algorithm to solve it by alternating the estimation of latent variable h and the optimization of $\mathcal{F}_z$. This iterative procedure can be summarized as follows:

1. Holding $\mathcal{F}_z$ fixed, compute the latent variable h^n for each training sample as follows:

$$h^n = \arg\max_{h \in \mathcal{H}_z} \Psi(V, h; \mathcal{F}_z); \tag{6.4}$$

2. Holding h^n fixed, optimize $\mathcal{F}_z$ by solving the following problem:

$$\mathcal{F}_z^* = \arg\min_{\mathcal{F}_z} \sum_n^N l(y^n, \Psi(V^n, h^n; \mathcal{F}_z)). \tag{6.5}$$

Eq. 6.4 can be easily solved by enumerating all possible $h \in \mathcal{H}_z$. Note that in Eq. 6.5, the latent variable h has been fixed to a single choice. $\mathcal{F}_z$ is a set of classifiers $\mathcal{F}_z = \{F_b : b \in \mathcal{H}_z\}$.

The optimization problem in Eq. 6.5 can be written as

$$L = \min_{\mathcal{F}_z} \sum_{b \in \mathcal{H}_z} \sum_{n:h^n=b} l(y^n, \Psi(V^n, h^n; \mathcal{F}_z))$$
$$= \sum_{b \in \mathcal{H}_z} \min_{F_b} \sum_{n:h^n=b} l(y^n, F_b(V^n)). \quad (6.6)$$

Then, Eq. 6.5 can be solved by minimizing $L(F_b) = \sum_{n:h^n=b} l(y^n, F_b(V^n)) \ \forall b \in \mathcal{H}_z$ independently. In practice, this can be simply achieved by a regular LogitBoost solver which learns a classifier $F_b(\cdot)$ over the training samples whose latent sub-tag label $h^n = b$, i.e. $\{(V^n, y^n)\}_{n:h^n=b}$.

6.4.1 Implementation Details

LogitBoost: The implementation of LogitBoost used in the optimization of $\mathcal{F}_z$ uses decision stumps as the weak learner. This allows it to handle complex feature vectors that consist of different feature categories, and also introduce the non-linearity to our model. However, one disadvantage of this boosting algorithm is that it takes a long time to train. It is particularly undesirable if we use boosting in an iterative training algorithm. To address this issue, in the training of $F_b(\cdot)$, we first run a linear SVM to filter out the feature dimensions that are assigned small weights. In other words, we use linear SVM as a feature selection step to select a subset of discriminative features. Then LogitBoost will be only used on this subset of features. This trick can significantly speed up the training process. Similar tricks are also introduced in [32]. To further improve the efficiency, for each LogitBoost classifier, we use only 256 decision stumps as weak learners.

Reweighting of classifiers: Due to the above feature selection trick and the early stopping scheme (only 256 stumps), the learned LogitBoost classifiers may have not converged. For the same testing example, it is possible that a well converged classifier will output a higher decision score than a classifier which has not converged, though both classifiers would classify this example as positive. This may cause problems when estimating the latent variable (Eq. 6.4), since we need to compare the decision scores from different LogitBoost classifiers. To address this issue, we train a linear SVM to calibrate the decision scores from different classifiers with respect to the hinge-loss $l(y^n, \mathbf{x}^n) = \max(0, 1 - y^n(\mathbf{w}^T\mathbf{x}^n))$, where $\mathbf{x}^n$ is the feature vector of V^n and $\mathbf{w}$ is the model parameters. $\mathbf{x}^n$ is represented as a sparse vector, and $|\mathbf{x}^n| = |\mathcal{H}_z|$. If the latent variable of V^n is equal to b, then the b-th element in $\mathbf{x}^n$ is equal to the decision score $F_b(V^n)$, and the rest of the elements are all 0.

The linear SVM is trained over all pairs of (V^n, y^n) in the training set. After the training, the scoring function in Eq. 6.4 can be re-written as $\Psi(V, h; \mathcal{F}_z) = \sum_{b \in \mathcal{H}_z} w_b \cdot 1_b(h) \cdot F_b(V)$, where w_b is the b-th element in $\mathbf{w}$. This calibration step is motivated by the mixture-model representation in [26]. The intuition is to re-weight each classifier based on its discriminative ability over the tag label y.

Bootstrapping: In this work, the tag labels of the training samples are extracted from user-provided meta-data, such as the video title, and user-provided tags. These tag labels are often irrelevant to the video content. In particular, we observe that the "hard" positive examples during learning are usually the videos with noisy training labels. Including those noisy video examples will likely deteriorate the learning performance. Instead, we would like to "remove" those positive samples that have the lowest decision scores and thus more likely to be outliers. In our learning algorithm, we maintain a set of training samples $\mathcal{S}$ that is a subset of the entire training set $\mathcal{T}$, i.e., $\mathcal{S} \subset \mathcal{T}$. In each iteration of our learning algorithm, instead of using all training samples, we optimize the model $\mathcal{F}_z$ only on the training subset $\mathcal{S}$ which contains the most "trustworthy" positive examples. Since we have a very large number of negative samples ($100K$) that is difficult to fit into the memory, in each iteration, we only include the "hard" negative samples into the training subset $\mathcal{S}$.

Learning procedure: First, we initialize the training set $\mathcal{S}$ and the latent variable h^n of each training sample in $\mathcal{S}$ (Section 6.4.3). Based on the latent sub-tag label, we can denote $\mathcal{S} = \cup_{b \in \mathcal{H}_z} \mathcal{S}_b$, where $\mathcal{S}_b$ contains the training samples that have the sub-tag label b. Then, we repeat the following steps for a fixed number of iterations.

1. Train a LogitBoost classifier $F_b(\cdot)$ for each sub-tag label $b \in \mathcal{H}_z$ over the training subset $\mathcal{S}_b$, then obtaining the model $\mathcal{F}_z = \{F_b : b \in \mathcal{H}_z\}$. We train a linear SVM to re-weight those classifiers;

2. Holding $\mathcal{F}_z$ fixed, compute the latent variable for every training sample in the entire training set $\mathcal{T}$ by Eq. 6.4. The decision score of each sample can be computed as $s^n = \Psi(V^n, h^n; \mathcal{F}_z)$. Similarly, we denote $\mathcal{T} = \cup_{b \in \mathcal{H}_z} \mathcal{T}_b$, where $\mathcal{T}_b$ contains the training samples which have the sub-tag label b;

3. Update the training subset $\mathcal{S} = \cup_{b \in \mathcal{H}_z} \mathcal{S}_b$, which we can rewrite as $\mathcal{S}_b = \mathcal{S}_b^{pos} \cup \mathcal{S}_b^{neg}$, where superscripts denote positive and negative subsets respectively. We re-construct $\mathcal{S}_b^{pos}$ by using the top k samples in $\mathcal{T}_b^{pos}$ with the largest decision scores. Similarly,

we re-construct S_b^{neg} by using the top k samples (hard negatives) from $\mathcal{T}_b^{neg}$. k is a predefined parameter;

In step 3, due to the fact that the dimension of our features is relatively large, in order to feed every training sample into the memory in step 1, we tune the parameter k so that the size $|S| \leq 20K$ and we incrementally increase the size of S during the iterative learning process.

The learning procedure described above is an algorithmic bootstrapping approach for generating a clean positive training set. We run this procedure for a fixed number of iterations in the absence of convergence guarantees, but find it effective experimentally. With the help of the training subset $\mathcal{S}$, our approach learns $\mathcal{F}_z$ on a training subset which contains the most "trustworthy" positive samples and thus it is likely more tolerant to label noise.

6.4.2 Computational Complexity

In our iterative learning algorithm, the complexity of each iteration involves the following three major factors: (1) the training of linear SVM for feature selection; (2) the training of LogitBoost classifiers; and (3) the computing of the latent variable for every training example.

For the training of linear SVM, we use the LIBLINEAR [23] package with the complexity of $O(NT)$, where N is the number of training examples and T is the number of feature dimensions. LIBLINEAR can achieve fast convergence and it is very efficient for training large-scale problems. In contrast, the training of LogitBoost is computational expensive. Traditional techniques for training a boosting classifier usually run in $O(MNT \log N)$, where M is the number of selected weak classifiers. In our implementation, by using a caching strategy proposed by Wu et al. [92], we are able to reduce the training time to $O(MNT)$, with a pre-calculation of time $O(NT \log N)$. Lastly, the complexity of inferring the latent variable for one training example is only $O(M)$. But since we have $20K$ - $100K$ training examples for each tag, the overall computation needed for computing latent variables is also very expensive. In our implementation, we parallelize the computing of latent variables over a number of CPUs, and each CPU takes a subset of training examples.

6.4.3 Initialization by Cowatch Features

For a non-convex problem, the initialization is usually very important. In this chapter, we treat the sub-tag label of a video as latent information. Intuitively, we would like to assign the same sub-tag label to the videos which are not just visually similar but also have a very strong semantic similarity. In our approach, we use video cowatch statistics [5] for the sub-tag initialization. Cowatch statistics are generated by measuring the occurrence frequency of two videos in the same viewing session. In other words, if two videos are watched one after the other by users, there will be a high cowatch connection between them. Note that it is common that users will watch similar videos in a short period of time (i.e. a viewing session). Cowatch statistics is a reliable tool to measure the video similarity since it combines the votes from millions of users. We believe the feature obtained from cowatch statistics is also a helpful cue to disambiguate different sub-tags. For example, users are more likely to watch a "mountain bike" video followed by another "mountain bike" video, and less likely followed by a "pocket bike" video.

The procedure of using cowatch statistics for initialization is summarized as follows:

Step 1: From the positive training set of the tag label z, we randomly sample N seed videos with the tag z. In our experiments, we set $N = 3000$. We generate a cowatch video list C_l^n for each sampled video. We remove from C_l^n the videos that do not contain the tag label z. Then, we combine those cowatch lists as a video set $C_s = \cup_{n=1}^{N} C_l^n$. Each video can be represented as a sparse vector v^n, and $|v^n| = |C_s|$. The value of i-th element in v^n is 1 if the corresponding i-th video of C_s also exists in C_l^n, and 0 otherwise.

Step 2: The cowatch lists of similar videos are likely to overlap. To generate the initial sub-tag label set, we simply cluster the set $\{v^n\}_{n=1}^{N}$ into K clusters using *k-means*. The distance between sparse feature vectors is computed by using the L_1 distance. Clusters with too few samples will be discarded and we obtain $K' \leq K$ clusters in the end. Then, we merge the cowatch video list C_l^n of each sampled video to its corresponding cluster. The video set C_s is generated by combining the cowatch lists of N randomly sampled videos. It is possible that for some videos in C_s, they may appear in only one of the N co-watch lists. Those videos are more likely to be irrelevant to the tag label z, so we remove those videos from the K' clusters.

Step 3: We can build the sub-tag set for tag z as $\mathcal{H}_z = \{0, ..., K' - 1\}$. For the purposes of initialization, we use the cluster label of each video as its initial sub-tag label h^n. Note

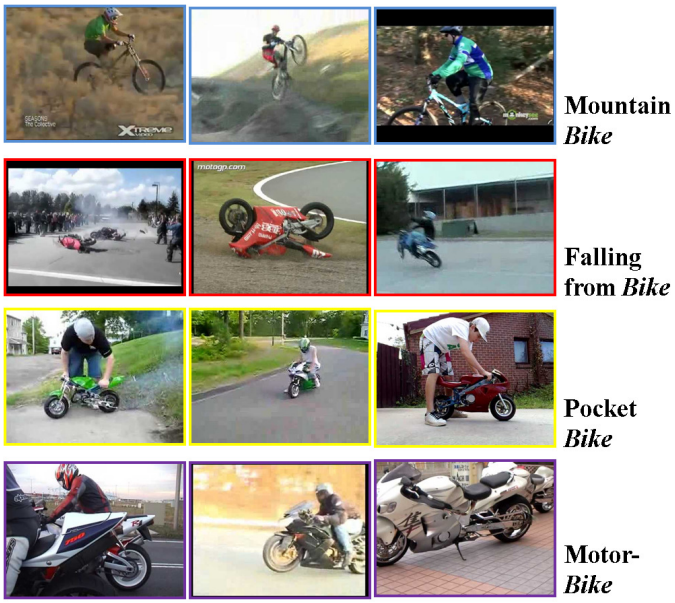


Figure 6.2: The sub-tag initialization results for the "bike" tag. In total, we generate four sub-tags: "mountain bike", "falling from bike", "pocket bike", and "motorbike". Note that our algorithm only cluster the candidate videos into four clusters and we manually assign a meaningful word label to each cluster.

that the generation of this sub-tag set is an unsupervised process. Our algorithm cannot automatically assign a semantic meaningful label to each sub-tag. Then, the videos from cluster b will form the initial training subset $\mathcal{S}_b^{pos}$, and we randomly sample a large number of negative samples to form the $\mathcal{S}_b^{neg}$, for all $b \in \mathcal{H}_z$.

The example initialization results for the tag label "bike" are depicted in Fig. 6.2. Each initialization cluster has a very unique semantic meaning. Note that the cowatch feature is only used for initialization purpose. It is also possible to consider the cowatch feature as a part of the video feature in our learning algorithm. However, the reliability of cowatch information depends on view counts of the video. In practice, most of testing videos would be the newly-uploaded videos, which have no viewing history.

6.5 Features

In order to tackle the problem of tag/concept/action learning many researchers [73, 20, 80, 7] have decided to use solely visual features. In the spirit of [77], we decide to use auditory

features in addition to visual features. We believe that audio gives an important cue in video analysis, especially for tag prediction. The audio features contribute to detect concepts that are very difficult to define visually, e.g. cat versus dog. Cats and dogs not only appear in similar contexts, but they are also very hard to disambiguate by only visual features. However, a dog's bark sound may help us distinguish between a dog video and a cat video.

We extract a variety of features from each video, which allow to capture various aspects of the video. The features can be categorized into three groups: frame features, motion features, and auditory features. For each type of feature, we use the standard bag-of-word representation. Each feature will be represented as a histogram by vector quantizing the feature descriptors. The histogram is normalized so that the sum of all the bin values is 1. The final feature vector of each video is the concatenation of the histograms from each feature. Its dimension is fixed for videos with different length, and the maximum number of non-zero dimensions for a video is 12439.

Frame features: This group of features consists of the histograms of oriented gradients (HOG) feature, color histogram, texton, and a face counter. For the HOG feature, at each frame pixel location, we extract a 1800-dimensional feature descriptor, which is the concatenation of HOG [14] in a 10×10 surrounding window. The raw descriptors are then collected into a bag-of-words representation by quantizing them using a randomized decision tree similar to [72]. In addition, we also compute a Hue-Saturation color histogram [48] and a Texton histogram [58] for every frame. Lastly, we run a face detector [83] over every frame and count the number of faces in each frame. This simple face counter provides an easy way to discriminate between videos containing human faces and those which do not. Note that all those frame-based features are represented as histograms, which are further pooled over the entire video using mean-pooling.

Motion features: In order to compute motion features we employ the cuboid interest point detector [20]. We extract spatio-temporal volumes around all the detected interest points. From each cuboid we extract two types of descriptors: (1) We concatenate the normalized pixel values to a vector and apply PCA to reduce the dimensionality to 256. (2) We first split each slice of the cuboid into 2×2 cells. Then, we concatenate all HOG descriptors of cells in the cuboid to a vector. Similarly, PCA is applied to reduce the dimensionality to 256. Both descriptors are further quantized using their corresponding codebooks.

Auditory features: We choose two widely-used audio features in addition to visual features: mel-frequency cepstral coefficients (MFCC) [12] and stabilized auditory images

(SAI) [52].

6.6 Experiments

We evaluate our method on a large-scale video dataset which consists of about 50 million YouTube videos. We only use a very small portion for training, and remaining videos are used for testing. For ease of evaluation, we arbitrarily selected 15 tags: *"bike"*, *"boat"*, *"card"*, *"dog"*, *"explosion"*, *"flower"*, *"helicopter"*, *"horse"*, *"kitchen"*, *"mountain"*, *"protest"*, *"robot"*, *"running"*, *"stadium"*, *"transformers"*. In terms of categories, this tag set contains "animals", "objects", "actions", "scenes", and "events". For each tag, we consider a set of $20K$-$100K$ videos which contains the given tag as the potential positive training set, and we randomly select around $100K$ videos as the negative training set. The rest of videos are used for testing. For each tag, we create a sub-tag set following the method described in Section 6.4.3. For illustration purposes, we summarize the sub-tag set of each tag in Table 6.1 and manually assign a semantic word to each sub-tag.

6.6.1 Evaluation Measures

For every given tag, we train a classifier model, which we apply on each video in the testing set (none of the videos in this set were used during training). A decision score is computed for each video using Eq. 6.1. If we had the ground-truth tag label of the 50 million testing videos, we could use the decision scores to compute the ROC or precision-recall curves. However, it is tremendously costly to accurately annotate that many videos, even when using low-cost online crowdsourced marketplaces (e.g. Amazon's Mechanical Turk). An alternative way is to randomly sample a relatively small number of testing videos then manually annotate those videos and only use them in the evaluation. We argue that this scheme is not fair, because the randomly sampled set would either be too small to fairly represent all the video categories on YouTube, or it would be too large to be practical.

To evaluate our approach, we choose the precision at K (precision@K) measure, which has been widely adopted in information retrieval. In practice, video retrieval and ranking are also important applications of automated video tagging system. For each tag, we first apply the trained model onto whole testing set (50M). We rank the videos in the testing set by their decision scores. Then, we only annotate the videos with top K decision scores. The precision at K is computed as precision@$K = |\{\text{relevant videos}\}|/K$. In this way, *for each*

Tag	**Sub-tags**
bike	mountain *bike*, falling from *bike*, pocket *bike*, motor*bike*
boat	building a *boat*, jet *boat*, *boat* accident
card	*Card* Captor Sakura (animation), *card* collection, making a greeting *card*, *card* trick
dog	*dog*1, *dog*2, *dog*3, *dog*4
explosion	bomb *explosion*, *explosion*2, building implosion
flower	*paper flower (howto)*, *flower (plant)*, *flower*3
helicopter	remote controlled (RC) *helicopter*1, RC *helicopter*2, *helicopter* crash, military *helicopter*
horse	*horse* jumping, *horse* reining, *horse*3
kitchen	*kitchen* remodeling, *kitchen*2, cooking show
mountain	*mountain* creek, *mountain* biking, *mountain* driving, *mountain* climbing
protest	riot police, *protest* in Iran, *protest*3, *protest* in Thailand
robot	industry robot, *robot*2, Super *Robot* Wars (video game)
running	free *running*, *running* tips, *running* back (football skill)
stadium	soccer *stadium*, football *stadium*, baseball *stadium*
transformers	video game, animation1, animation2, movie, toy

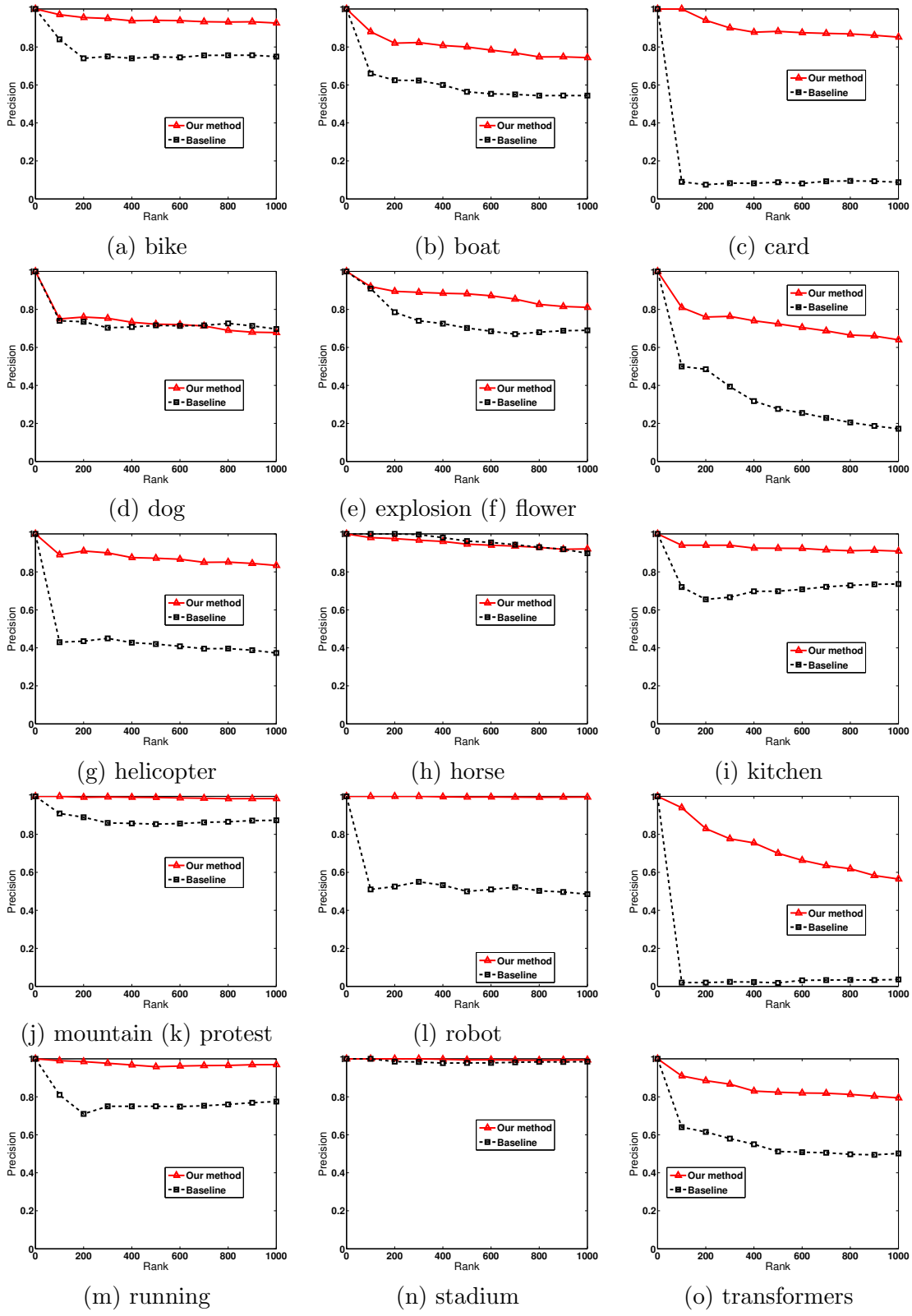
Table 6.1: The summary of the sub-tag labels for the 15 tags used in our experiments. Note that our algorithm cannot assign the semantic meaning label to each sub-tag. For the purpose of illustration, we manually assign a word label to each sub-tag by summarizing the video clusters obtained from cowatch initialization. For some sub-tags which are difficult to assign a meaningful word label, we simply use the cluster number to represent them.

tag, the evaluation only requires the annotations of K videos. We choose $K = 1000$, giving us $15,000$ videos to annotate. Compared to 50 million videos, this number is negligible but it is far more acceptable in terms of annotation cost. This measurement is particularly suitable for the situation in which the testing set is dynamic, as in the case in which new videos are added to the collection over time. For each update of the testing set, we only need to annotate the new videos that appear in the top-K rank list.

6.6.2 Results

We compare our method to the video tag learning approach described in [77]. In order to make the comparison fair, we use the exactly the same features (Section 6.5) for both the baseline and our method. The average precision@1000 of the baseline is 57.36%, and our method is 84.14%. We observed an improvement of 46% on the average precision@1000. Fig. 6.3 shows the precision at K curves of both our method and the baseline. As we can see, our method significantly outperforms the baseline on 12 tags, and we achieve similar results on the tags "dog", "horse", and "stadium". Both our method and the baseline achieve very good performance on the tags of "stadium" and "horse". Most of "dog" videos uploaded to YouTube are usually shot from consumer level hand-held cameras. These videos have very limited variation, and thus our method performs similarly to the baseline on this tag. After manually examining the cowatch initialization results, as shown in Table 6.1, the sub-tags of "dog" do not have any semantic meaning. This observation demonstrates a limitation of our approach: our approach can barely improve the performance for the unambiguous tags. This is what we expect since for the unambiguous tags, the latent sub-tag information has little semantic meaning and thus its contribution is very limited.

As shown in Table 6.1, most of sub-tag sets generated from cowatch initialization are semantically meaningful. One interesting question is whether the iterative process of our algorithm really contribute to the performance improvement. To answer this question, we compare our approach with a method that is only trained on the initial training subset obtained by the cowatch initialization. This method essentially only runs the first step of our learning procedure (Section 6.4.1) once. Due to the cost of annotation, we only run this experiment on two tags: "transformers" and "bike". For comparison purposes, we compute precision@1000 as shown in Table 6.2. We can see that our method outperforms the method using only initial training subset. This is reasonable because in the initial training subset, the latent sub-tags are generated from the cowatch information and are not tied



(a) bike (b) boat (c) card

(d) dog (e) explosion (f) flower

(g) helicopter (h) horse (i) kitchen

(j) mountain (k) protest (l) robot

(m) running (n) stadium (o) transformers

Figure 6.3: Precision at K curves for both baseline and our method on the 50 million YouTube video dataset. We incremental increase K from 100 to 1000.

Tag	Baseline	Initialization only	Our approach
transformers	50.2%	33.1%	**79.4%**
bike	74.8%	87.4%	**92.5%**

Table 6.2: Precision@1000 for tags "transformers" and "bike". We compare our method to the baseline, and a method that runs the first step of our learning procedure once.

with our end goal of tag learning. Interestingly, on the "transformers", the method using only initialization is worse than the baseline. We believe it is due to the high complexity of the "transformers" videos, and the initialization of the sub-tag set cannot properly capture the variance of the "transformers" tag.

Given a tag label z and a testing video, besides a decision score of tag z being assigned to the testing video, we can also infer the sub-tag label for the testing video by Eq. 6.2. We visualize the testing videos with top scores from each sub-tag in Figs. 6.4,6.5. Two interesting observations can be made. In Fig. 6.4 ("transformers"), some videos under sub-tag h_0 (video game) are classified as h_4 (movie). In the cowatch initialization of tag "bike", most videos of sub-tag h_2 are about "pocket bike". However, as shown in Fig. 6.5, the testing videos under sub-tag h_2 are mostly "motorbike". Those videos seem to be "misclassified" with respect to sub-tag label, but they are all related videos to the tag label "bike", which is exactly what we expect. As we pointed out in Section 6.3, the latent sub-tag label is only a by-product of our model. Our model is only optimized for tag-level classification, so we do not aim to obtain good sub-tag level classification results. Therefore, although some video do not have an "accurate" sub-tag label, they are assigned the correct tag labels. Note that the "misclassified" and "accurate" are determined by comparing the testing videos with initialization results. Sub-tag information is latent so we cannot measure the performance of sub-tag level classification.

6.7 Summary

We have studied the problem of semantic variations in the videos which share the same tag. We have named those semantic variants as sub-tags, which were treated as latent variables and used to assist the task of tag learning. A general latent learning framework was proposed to jointly model the tag label and its related latent sub-tags. We have presented a clustering approach based on cowatch information to initialize the latent sub-tag labels in our learning

(a) h_0 (video game)

(b) h_1 (animation1) (c) h_2 (animation2)

(d) h_3 (toy) (e) h_4 (movie)

Figure 6.4: Visualizations of the sub-tag labels of the testing videos for tag "transformers".

(a) h_0 (mountain bike) (b) h_1 (falling from bike)

(c) h_2 (pocket bike) (d) h_3 (motorbike)

Figure 6.5: Visualizations of the sub-tag labels of the testing videos for tag "bike".

framework. By running experiments on the testing set which consists of about 50 million YouTube videos, we have demonstrated that our approach significantly outperformed the baselines, with an improvement of over 46% on the average precision@1000.

Chapter 7

Conclusion and Future Work

In this dissertation, we have presented two discriminative latent variable models and one general latent learning framework for addressing a series of challenging problems in visual recognition. A common theme of the proposed approaches in this dissertation is that they all involve learning with *latent variables*. In the first model, we jointly consider human actions and poses in a unified system, and treat human poses as latent variables. In the second model, we address the semantic variations of YouTube videos and consider the sub-tag label as latent information. Moreover, we propose a novel latent learning framework – *kernel latent SVM* which combines the benefits of latent SVM and kernel methods. The proposed approaches demonstrate the flexibility and effectiveness of discriminative latent variable models for addressing a variety of problems in computer vision.

We believe the work presented in this dissertation will lead to many interesting directions for future research. We will briefly highlight two directions for future research as follows.

Convex Relaxation: One limitation of latent SVM is the non-convexity. Although there are three different algorithms available (Chapter 2) for training the latent SVM, all of them can only find a local optimum and they are sensitive to initialization due to the non-convex nature of latent SVM. There is also no theoretical guarantee that a local optimum can produce good recognition performance. Moreover, in each iteration of those algorithms, inference of the latent variable is required for each positive training example. Thus, the training of latent SVM is usually much slower than its non-latent counterpart.

One possible direction is *convex relaxation* which aims to relax a non-convex optimization problem to a convex one and then solve it using standard quadratic program solvers.

Quadrianto et al. [61] propose a convex relaxation algorithm for a mixture of regression models. Joulin and Bach [40] propose a simple and general framework to solve a series of weakly supervised learning problems by convex relaxation, such as multiple instance learning and discriminative clustering. Those algorithms could be good starting points for solving latent SVM through convex relaxation.

Loss Function with Latent Variables: In latent variable models, the training instance is often in the form of $(\mathbf{x}, \mathbf{h}, y)$. The loss functions in latent variable models only depend on the class label y without considering the latent variable $\mathbf{h}$. For example, in the multi-class latent SVM we presented in Chapter 2, the loss function is the standard $0/1$ loss, i.e. $\Delta(y_i, y) = 1_{[y_i \neq y]}$. Therefore, the inferring of latent variables on testing examples might not be accurate. We believe it will limit the applicability of latent variable models to the situations where accurate predictions of latent variables on testing examples are required.

One interesting direction for dealing with this issue is to include the latent variables in the loss function, e.g. changing the form of loss function to $\Delta(y_i, \mathbf{h}_i, y, \mathbf{h})$. However, due to the fact that the latent variables are not observed during training, this modification will not be trivial. Kumar et al. [43] approach this problem by introducing a conditional distribution to model the uncertainty of latent variables in the loss function. Similar motivations are also adopted by Shapovalova et al. [69] and they add an additional loss function to ensure the pairwise similarity of latent variables across the training set. We believe these two approaches are good starting points for considering latent variables in the loss functions.

Latent variable models are a class of powerful tools and we believe they will open the possibility of constructing more powerful and expressive approaches for visual recognition.

Bibliography

[1] Yasemin Altun, Thomas Hofmann, and Ioannis Tsochantaridis. SVM learning for interdependent and structured output spaces. In G. Bakir, T. Hofman, B. Scholkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, editors, *Machine Learning with Structured Outputs*. MIT Press, 2006.

[2] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 561–568. MIT Press, 2003.

[3] Hrishikesh Aradhye, George Toderici, and Jay Yagnik. Video2text: Learning to annotate video content. In *ICDMW '09: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, pages 144–151, Washington, DC, USA, 2009. IEEE Computer Society.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, August 2011.

[5] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for YouTube: taking random walks through the view graph. In *Proceeding of the 17th international conference on World Wide Web*, pages 895–904. ACM, 2008.

[6] David M. Blei and Jon D. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2008.

[7] D. Borth, A. Ulges, and T.M. Breuel. Relevance filtering meets active learning: improving web-based concept detectors. In *Proceedings of the international conference on Multimedia information retrieval*, pages 25–34. ACM, 2010.

[8] Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors. *Large Scale Kernel Machines*. MIT Press, Cambridge, MA., 2007.

[9] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors training using 3d human pose annotations. In *IEEE International Conference on Computer Vision*, 2009.

[10] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[12] G. Chechik, E. Ie, M. Rehn, S. Bengio, and D. Lyon. Large-scale content-based audio retrieval from text queries. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 105–112. ACM, 2008.

[13] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[14] Navneet Dalal and Bill Triggs. Histogram of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[15] Vincent Delaitre, Ivan Laptev, and Josef Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *British Machine Vision Conference*, 2010.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[17] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for multi-class object layout. In *IEEE International Conference on Computer Vision*, 2009.

[18] Santosh Divvala, Alexei A. Efros, and Martial Hebert. How important are Deformable Parts in the Deformable Parts Model? . In *Parts and Attributes Workshop at ECCV*, 2012.

[19] Trinh-Minh-Tri Do and Thierry Artieres. Large margin training for hidden markov models with partially observed states. In *International Conference on Machine Learning*, 2009.

[20] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV'05 Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[22] Jianping Fan, Yi Shen, Ning Zhou, and Yuli Gao. Harvesting large-scale weakly-tagged image databases from the web. In *cvpr*, pages 802–809, jun. 2010.

[23] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 2008.

[24] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[25] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[26] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1672–1645, 2010.

[27] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, January 2005.

[28] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[29] Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. Pose search: retrieving people using their pose. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[30] David A. Forsyth, Okan Arikan, Leslie Ikemoto, James O'Brien, and Deva Ramanan. Computational studies of human motion: Part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3):77–254, July 2006.

[31] Chunhui Gu and Xiaofeng Ren. Discriminative mixture-of-templates for viewpoint classification. In *European Conference on Computer Vision*, 2010.

[32] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[33] Vinay Namboodiri Hakan Bilen and Luc Van Gool. Object and action classification with latent variables. In *Proceedings of the British Machine Vision Conference*, pages 17.1–17.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.17.

[34] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of Twenty-Second Annual International Conference on Research and Development in Information Retrieval(SIGIR)*, pages 50–57, 1999.

[35] S.J. Hwang and K. Grauman. Reading between the lines: Object localization using implicit cues from image tags. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2971–2978. IEEE, 2010.

[36] Nazli Ikizler, R. Gokberk Cinbis, Selen Pehlivan, and Pinar Duygulu. Recognizing actions from still images. In *International Conference on Pattern Recognition*, 2008.

[37] Nazli Ikizler-Cinbis, R. Gokberk Cinbis, and Stan Sclaroff. Learning actions from the web. In *IEEE International Conference on Computer Vision*, 2009.

[38] Thorsten Joachims. Making Large-Scale SVM Learning Practical. In Bernhard Schölkopf, Christopher J.C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, USA, 1999. MIT Press.

[39] Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 2008.

[40] Armand Joulin and Francis Bach. A convex relaxation for weakly supervised classifiers. In *International Conference on Machine Learning (ICML)*, 2012.

[41] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

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

[43] M. Pawan Kumar, Ben Packer, and Daphne Koller. Modeling latent variable uncertainty for loss-based learning. In *International Conference on Machine Learning*, 2012.

[44] M. Pawan Kumar, Haithem Turki, Dan Preston, and Daphne Koller. Learning specific-class segmentation from diverse data. In *IEEE International Conference on Computer Vision*, 2011.

[45] Tian Lan, Yang Wang, Weilong Yang, Stephen N. Robinovitch, and Greg Mori. Discriminative latent models for recognizing contextual group activities. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[46] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent Rl Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:24–72, 2004.

[47] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[48] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[49] Li-Jia Li, Richard Socher, and Li Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[50] Stan Z. Li and ZhenQiu Zhang. Floatboost learning and statistical face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1112–1123, September 2004.

[51] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

[52] R.F. Lyon, M. Rehn, S. Bengio, T.C. Walters, and G. Chechik. Sound retrieval and ranking using sparse auditory representations. *Neural computation*, 22(9):2390–2416, 2010.

[53] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[54] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European Conference on Computer Vision*, 2010.

[55] Juan Carlos Niebles, Bohyung Han, Andras Ferencz, and Li Fei-Fei. Extracting moving people from internet videos. In *European Conference on Computer Vision*, 2008.

[56] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *British Machine Vision Conference*, volume 3, pages 1249–1258, 2006.

[57] NIST. Trecvid notebook papers, 2010.

[58] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168. IEEE, 2006.

[59] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *IEEE International Conference on Computer Vision*, pages 1307–1314, 2011.

[60] Sobhan Naderi Parizi, John Oberlin, and Pedro F. Felzenszawlb. Reconfigurable Models for Scene Recognition. In *Computer Vision and Pattern Recognition*, Providence, Rhode Island, États-Unis, June 2012.

[61] Novi Quadrianto, Tiberio Caetano, John Lim, and Dale Schuurmans. Convex relaxation of mixture regression with efficient algorithms. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 1491–1499, 2009.

[62] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, Cambridge, MA, 2005.

[63] Deva Ramanan. Learning to parse images of articulated bodies. In *Advances in Neural Information Processing Systems*, volume 19, pages 1129–1136, 2007.

[64] Deva Ramanan and David A. Forsyth. Automatic annotation of everyday movements. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[65] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

[66] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision*, 2007.

[67] R. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*. Springer, 2004.

[68] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local SVM approach. In *IEEE International Conference on Pattern Recognition*, volume 3, pages 32–36, 2004.

[69] Nataliya Shapovalova, Arash Vahdat, Kevin Cannons, Tian Lan, and Greg Mori. Similarity constrained latent support vector machine: An application to weakly supervised action classification. In *European Conference on Computer Vision*, 2012.

[70] Gaurav Sharma, Frédéric Jurie, and Cordelia Schmid. Discriminative Spatial Saliency for Image Classification. In *Computer Vision and Pattern Recognition*, Providence, Rhode Island, États-Unis, June 2012.

[71] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

[72] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[73] C.G.M. Snoek, KEA van de Sande, O. de Rooij, B. Huurnink, JC van Gemert, JRR Uijlings, J. He, X. Li, I. Everts, V. Nedovic, et al. The MediaMill TRECVID 2008 semantic video search engine. In *Proceedings of the TRECVID Workshop*, 2008.

[74] Richard Socher and Li Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[75] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[76] Christian Thurau and Václav Hlaváč. Pose primitive based human action recognition in videos or still images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[77] G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, and J. Yagnik. Finding meaning on youtube: Tag recommendation and category discovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3447–3454, 2010.

[78] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *IEEE PAMI*, 30(11):1958–1970, November 2008.

[79] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[80] Adrian Ulges, Markus Koch, Damian Borth, and Thomas M. Breuel. Tubetagger - youtube-based concept detection. In *ICDM Workshops*, pages 190–195, 2009.

[81] Arash Vahdat, Bo Gao, Mani Ranjbar, and Greg Mori. A discriminative key pose sequence model for recognizing human interactions. In *Eleventh IEEE International Workshop on Visual Surveillance*, 2011.

[82] Andrea Vedaldi and Andrew Zisserman. Structured output regression for detection with partial truncation. In *Advances in Neural Information Processing Systems*. MIT Press, 2009.

[83] Paul Viola and Michael Jones. Robust real-time object detection. In *Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling(SCTV)*, 2001.

[84] Paul Viola, John C. Platt, and Cha Zhang. Multiple instance boosting for object recognition. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006.

[85] Huayan Wang, Stephen Gould, and Daphne Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *European Conference on Computer Vision*, 2010.

[86] Yang Wang, Hao Jiang, Mark S. Drew, Ze-Nian Li, and Greg Mori. Unsupervised discovery of action classes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[87] Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[88] Yang Wang and Greg Mori. A discriminative latent model of image region and object tag correspondence. In *Advances in Neural Information Processing Systems*. MIT Press, 2010.

[89] Yang Wang and Greg Mori. A discriminative latent model of image region and object tag correspondence. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[90] Yang Wang and Greg Mori. A discriminative latent model of object classes and attributes. In *European Conference on Computer Vision*, 2010.

[91] Yang Wang and Greg Mori. Hidden part models for human action recognition: Probabilistic versus max-margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1310–1323, 2011.

[92] Jianxin Wu, S. Charles Brubaker, Matthew D. Mullin, and James M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(3):369–382, March 2008.

[93] Linli Xu, James Neufeldand, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 1537–1544. MIT Press, Cambridge, MA, 2005.

[94] Weilong Yang and George Toderici. Discriminative tag learning on youtube videos with latent sub-tags. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2011.

[95] Weilong Yang, Yang Wang, and Greg Mori. Recognizing human actions from still images with latent poses. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[96] Weilong Yang, Yang Wang, Arash Vahdat, and Greg Mori. Kernel Latent SVM for Visual Recognition. In *Neural Information Processing Systems (NIPS)*, 2012.

[97] Chun-Nam Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *International Conference on Machine Learning*, 2009.

[98] Alan Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[99] Long Zhu, Yuanhao Chen, Alan Yuille, and William Freeman. Latent hierarchical structural learning for object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.