Lecture 16: Learning Object Models for Detection

A.L. Yuille

March 18, 2012

1 Introduction

This lecture describes how to learn models for detecting objects. These models explicitly represent objects parts and allow the spatial relations between these parts to vary. This differs from the methods we used to model faces and text which did not represent parts and instead relied on features that were invariant to these spatial variations. But many objects have large spatial variations and it is not possible to find invariant features. Moreover, modeling the subparts explicitly enables us to detect them and hence parse the image to detect the positions of the parts (i.e. say where different parts of the object are – and not simply detect the object).

2 Standard Learning of Object Models without Hidden variables

Previous lectures gave models where the objects are represented by deformable parts. These are represented by graphical models $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the state variables $\{z_\mu : \mu \in \mathcal{V}\}$ represent the positions of the parts and the edges $(\mu, \nu) \in \mathcal{E}$ specify the nodes which are directly related. We use x to represent the input image.

The models are specified by a conditional distribution:

$$P(z|x;\lambda) = \frac{1}{Z[\lambda, x]} \exp\{-E(z,x)\},$$
$$\text{with} E(z,x) = \sum_\mu \lambda_\mu \cdot \phi(z_\mu, x) + \sum_{(\mu,\nu)\in\mathcal{E}} \lambda_{\mu\nu} \cdot \phi(z_\mu, z_\nu, x). \quad (1)$$

The unary terms $\lambda_\mu \cdot \phi(z_\mu, x)$ give local evidence for the state z_μ of part μ. The binary terms $\lambda_{\mu\nu} \cdot \phi(z_\mu, z_\nu, x)$ specify the relationships between the states z_μ, z_ν of pairs of parts. These binary terms may be dependent on the image x in which case they are sometimes called data-dependent priors. Often, however, the binary terms are independent of x and so there are similar to priors (but not identical – explain why.)

If we have training examples – $\mathcal{D} = \{(z^a, x^a) : a \in \mathcal{D}\}$ – then we can learn the parameters λ of the distribution $P(z|x;\lambda)$ by maximum likelihood (ML) – i.e. by maximizing $\prod_{a\in\mathcal{D}} P(z^a|x^a;\lambda)$ with respect to λ. There are only two difficulties: (I) Computational – (a) can we perform inference on the model to estimate quantities like $z^* = \arg\max_z P(z|z;\lambda)$? (note that inference is a pre-requisite for learning), and (b) can we compute, or approximate, the normalization constant $Z[\lambda, x]$ and other properties? (II) Data – do we have enough training data to learn the model (i.e. so that it generalizes to novel data)? This depends on the number of parameters λ – and so we can adjust the complexity of the model to the amount of data available (e.g., by imposing regularization constraints on the λ).

One way to simply the learning is to use machine learning methods. This can be thought of in two ways: (i) re-formulate the problem as classification and drop the probabilistic interpretation, and (ii) treat the machine learning formulation as an approximation to the probabilistic formulation. This viewpoint has been developed by several authors – see handout by Yuille and He – who stress the many advantages of the rich conceptual structure provided by probability theory in conjunction with the computational practicality of machine learning methods. But here we will concentrate on the machine learning formulation.

The machine learning formulation drops the probabilistic formulation and specifies the learning task as being to find parameters λ to obtain an estimate of the state variables z:

$$z^* = \arg\min_z \sum_\mu \lambda_\mu \cdot \phi(z_\mu, x) + \sum_{(\mu,\nu)\in\mathcal{E}} \lambda_{\mu\nu} \cdot \phi(z_\mu, z_\nu, x), \tag{2}$$

so that this estimator minimizes a criterion which combines the empirical loss $L(z, z^a)$ with a regularization term $1/2|\lambda|^2$ (which helps give generalization). This criterion is given by:

$$\begin{aligned} 1/2|\lambda|^2 + C \sum_{a\in D} \min_z \{ \sum_{\mu\in\mathcal{V}} \lambda_\mu \cdot \phi(z_\mu, x^a) + \sum_{(\mu,\nu)\in\mathcal{E}} \lambda_{\mu\nu} \cdot \phi(z_\mu, z_\nu, x^a) + L(z, z^a) \} \\ -C \sum_{a\in D} \sum_{\mu\in\mathcal{V}} \lambda_\mu \cdot \phi(z^a_\mu, x^a) + \sum_{(\mu,\nu)\in\mathcal{E}} \lambda_{\mu\nu} \cdot \phi(z^a_\mu, z^a_\nu, x^a). \end{aligned} \tag{3}$$

This is a generalization of the max-margin criterion for binary classification. It is called the structure max-margin criterion (citations). It is a convex function which can be minimized by two different types of algorithms. The first is an *online* method which selects data at random from $\mathcal{D}$ and performs an iteration of steepest descent on this quantity (removing the summations) – the *structure perceptron algorithm* (Collins) – is an approximation to this procedure. The second is to reformulate this problem in terms of primal and dual problem (see paper handout for lecture 18, the discussion in Yuille and He, and many other sources). This leads to an algorithm that is similar to that used for max-margin classification (i.e. support vector machines).

Note: the criterion in equation (3) is usually obtained as a convex upper bound (convex in λ) of the empirical loss $\sum_{a\in\mathcal{D}}$ plus a regularization term $1/2|\lambda|^2$. But it can also be related (e.g., Yuille and He handout) as an approximation to more standard probabilistic estimation of the parameters of regression models.

The handout to lecture 18 will give vision examples of the use of these machine learning methods for learning the parameters of models of objects – this include AND/OR graph models of baseball players. (Need citations to more standard work on learning probabilistic models of objects).

Another class of learning problem – which we will discuss later in this lecture – is when the model has hidden variables. In one common form – the only information is whether an object is present in an image window or not. In this case, the state variables for the positions of parts are hidden variables (this also requires introducing a probability model for the data if the object is not present). Formally this can be addressed by using the EM algorithm to deal with the hidden variables. But, in practice, people often use machine learning methods – such as multiple instance learning and a simplification known as latent SVM. These are described in a later section of the lecture.

3 Other Topics

This section describes other topics which we did not have time for (will be written up in later drafts).

Firstly, what are the cues (i.e. the $\phi(.)$ that should be used for these models? One solution is to specify a dictionary of possible $\phi(.)$ and then allow the learning process to select which $\phi(.)$ using a sparsity criterion (e.g., an $\lambda|$ term). But, as discussed in this AdaBoost lecture, this only postpones the problem to the choice of the dictionary (and the dictionary for faces is different than the dictionary for text). There are types of visual cues that have been empirically been shown to be useful (examples in the next section) and considerable efforts to learn dictionaries (citations), but there is no satisfactory solution at present. Only general criteria – e.g., the cues should be specific to the objects (and object parts) but invariant to nuisance parameters (see lecture 3).

Secondly, how do the condition models $P(z|x)$ relate to generative models $P(x|z)$ and $p(z)$? Generative models are able to generate samples of the objects – by stochastic sampling from $P(x|z)$ and $p(z)$. They have many other desirable properties (see the next topic) but they suffer from some big problems. Most importantly, the space of all images is astronomically big – nobody, in any research discipline, has managed to define probability distributions over spaces of this enormous size. The ability to do this in general would require the ability to understand and model all the patterns that occur in natural images – this is an exciting but a very difficult endeavour. Moreover, at present, computer vision lacks the technical machinery to model images expect in restricted circumstances. There are

limited exceptions – e.g., active appearance models work for certain objects under restricted circumstances, lambertian reflectance models coupled with three-dimensional shape models can also deal with certain types of objects, there are also models of restricted classes of texture.

It is considerably easier to have generative models for image features. For example, there are several models that are generative for sparse features points (e.g., Caltech work, L. Zhu et al, others). These are simpler because they only need to put distributions over the positions and attributes of the interest points. Hence they do not provide generative models of the full images. Moreover, if we sample from the image to obtain interest points with attributes then it is not certain they there is a consistent image which has these attributes (well, if the interest points are far enough apart then it is almost certain that there is). Suppose, for example, we can sample the first derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ independently – then these is no guarantee that these samples will satisfy necessary consistency constraints to correspond to a well-defined image (like the integrability constraint). Technically learning should involve computing, or estimating, the g-factor (Coughlan and Yuille).

Thirdly, how to learn the graph structure of the model? The current formulation assumes that the graph structure of the model – the nodes and the edges – is known and only the parameters need to be estimated. This formulation does include some ability to learn the graph edges – by including in the dictionaries potentials which allow us to link different nodes to form edges. The generative approach gives a principled way to learn models of objects and images without specifying the graph structure. The basic strategy is simple – defined a large class of graphical models (with different graph structures and parameters) – and selecting between them to pick the best model that describes the data. This involves evaluating $P(\mathcal{D}|\text{model}_1, \lambda_1), ..., (\mathcal{D}|\text{model}_N, \lambda_N)$. In practice, this is difficult because there are an extremely (exponentially) large set of models that we need to consider. Current strategies involve building models out of elementary components and growing them (e.g. by AND, or OR operations). The ideas that models can be constructed in this manner relates to the *compositionality conjecture* and it at the heart of stochastic grammars. Searching through the enormous space of possible models can be done by a greedy strategy – e.g., pick the best model and grow it – or by picking several different models in parallel and growing all of them.

4 Hierarchical Part Models and Latent SVM

The remainder of this lecture concentrates on a special class of models that have been successful on the Pascal Object detection challenge. This section introduces the hierarchical structure of the models, the types of image cues ϕ's that are used, and the latent SVM learning algorithm.

In these models, objects are represented as a mixture of hierarchical models. A hierarchical model is illustrated in figure (1). Several models (i.e. a mixture) are used for each object to deal with the different viewpoints. Each hierarchical model represents the object in terms of parts – but these parts are not "semantic parts", so there do not correspond to the legs of the horse, the head, or the torso. The parts are allowed to model to deal with the spatial deformations that can happen at different scales of the object.

The image cues $\phi(.)$ are chosen to represent the different possible appearances of the object and to use those factors that distinguish it from the background. Appearance patterns can be roughly classified into two classes: (i) structural (e.g., the head of a cat) which can be roughly described by the intensity edges and their spatial relations (e.g. by a HOG), and (ii) textural (e.g., the fur of a cat) which can be modeled by Bag of Words – i.e. histograms of image features or words. Here the words will be based on SIFT features.

Each hierarchical model is a graphical model without closed loops. So we can use dynamic programming to perform inference and estimate its state variables. We can perform inference separately for each hierarchical model (i.e. each mixture component) separately and then select the model that has the lowest energy score.

4.1 The Hierarchical Model

Each hierarchy is a 3 layer tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node represents a part. The number of nodes depends on the application. In one example (L. Zhu, Y. Chen, A. Yuille and W. Freeman – CVPR 2010) there are a total of 46 nodes: 1 root node representing the entire object, 9 part nodes (at the second level), and 36 sub-part nodes (at the third level). The position of a node $\mu \in \mathcal{V}$ is represented by a state variable z_μ. The potentials $\lambda_\mu \cdot \phi(z_\mu, x)$ relates the position of the part to cues (HOG and SIFT features) in the image. The graph edges $(\mu, \nu) \in \mathcal{E}$ relate the root node to all

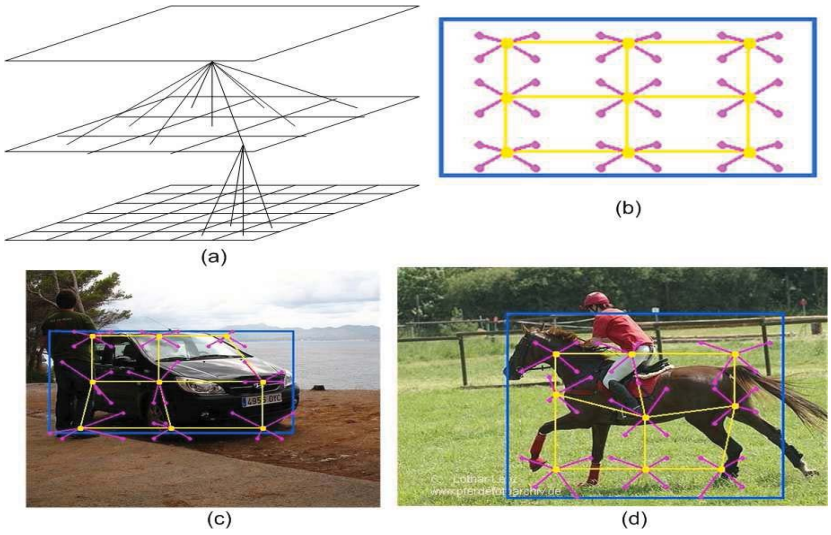

Figure 1: A 3-layer model (a), the structure has three layers with nodes in simple grid layouts (b), the nodes represent parts which move to adapt to the object (c,d).

nine nodes at the second level, and each second level node is related to four nodes at the third level. The potential defined over the graph edges – $\lambda_{\mu\nu} \cdot \phi(z_\mu, z_\nu)$ – specify statistical constraints for the relative positions of the parts and subparts.

Each model can be formally expressed as probability model of form given in equation (1). In particular, the energy function can be expressed in terms of a sum of unary terms (depending on the states of the nodes) and binary terms (depending on the states of pairs of nodes). The nodes can be re-expressed in terms of different level – i.e. $\mathcal{V} = \mathcal{V}_1 \bigcup \mathcal{V}_2 \bigcup \mathcal{V}_3$ where $\mathcal{V}_1$ is the root node, $\mathcal{V}_2$ is the nine level-2 nodes, and $\mathcal{V}_3$ are the thirty six level-3 nodes. Graph edges exist between nodes at adjacent levels. (Write down the full model next time).

The basic ideas of the inference and learning algorithms do not depend on the details of the model – only on the fact that each models can be expressed as a graph without closed loops and the energy is a linear sum of features ϕ's weighted by parameters λ's.

First we will describe the features/cues and then we will describe how latent SVM can be applied to learn the parameters λ.

4.2 The features

The binary features $\phi(z_\mu, z_\nu)$ are defined over the graph edges and relate the spatial positions of the nodes (they do not depend on the image x). We express $z_\mu = (u_\mu, v_\mu)$ in terms of its components in the x and y directions. Then $\phi(z_\mu, z_\nu) = (\Delta u, \Delta v, \Delta u^2, \Delta v^2)$, where $\Delta u = (u_\mu - u_\nu)$ and $\Delta v = (v_\mu, v_\nu)$. These statistics are quadratic in the relative positions of the two parts, and hence correspond to a Gaussian distribution on the relative positions of the parts (if we expressed the model probabilistically).

The unary features $\phi(z_\mu, x)$ are defined as follows. Each node μ corresponds to a region $\mathcal{D}(z_\mu)$ of the image. For the root node, this region consists of $W \times H$ cells of size 8×8 pixels: $\mathcal{D} = \bigcup_{w,h=1,1}^{W,H} \mathcal{D}_{w,h}$. For nodes in layer-2, each region is composed of $2/3W \times 2/3H$ cells, while layer-3 nodes have $1/3W \times 1/3H$ cells. This is illustrated in figure (1).

The unary features are of two types, which model the edges and the internal appearance of the objects respectively.

The first type of features, for edges, are modeling using HOG's (Histograms of Gradients). For each cell we compute a HOG, which is a 31-dimensional vector (containing 9 contrast sensitive features, eighteen contrast sensitive features, and four summations). HOGs are invariant to small spatial changes of the object and lighting and other

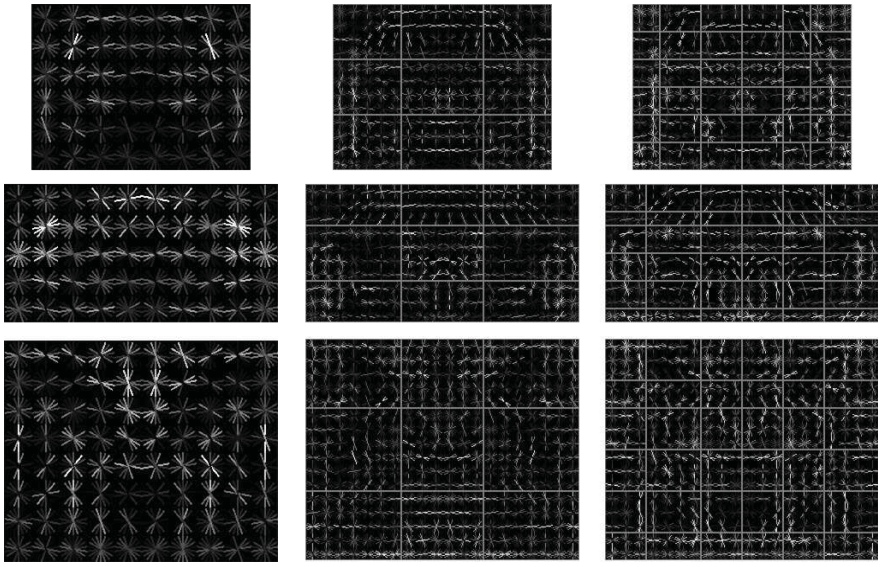

Figure 2: Some models (appearance only) learnt from the PASCAL 2007 dataset. The first 2 rows are the car models from two views and the last row is a horse model. Three columns show the weights in each orientation of the HOG cells at the root node, level-2 and level-3. Each cell consists of 8×8 pixels. The models look semantically meaningful. The weights along the object boundary are high. The features at different layers capture object appearance in a coarse-to-fine way. The features at lower levels capture more detailed appearance (e.g. the horse legs at the 3rd layer look brighter.

nuisance variables (see third lecture). This gives a potential of form $\phi_{HOG}(z_\mu) = \{\phi_{HOG;w,h}(z_\mu) : w, h\}$ – i.e. this concatenates the HOG responses in each cell. There are lambda variables for each cell – hence $\lambda_{HOG} = \{\lambda_{HOG,w,h} : w, h\}$. Hence if $W = 10, H = 5$, then there is a 50×31 HOG feature vector for each node. Figure (2) represents the parameters λ learnt for some parts and illustrates that the larger λ's tend to capture the edge structure of the object approximately (this will not be true for highly deformable objects like cats).

The second type of features are intended to capture the appearance of regions of the images. These features are Bag of Words constructed as follows. On all 8×8 cells in the dataset (including all object and all the background) we compute the SIFT descriptor (see third lecture). We do k-means clustering of these SIFT responses to obtain M centers (each $M = 60$). This gives a dictionary of M words – corresponding to the centers obtained by clustering. Each SIFT descriptor, computed on an 8×8 cell, can be assigned to a single a word (i.e. the cluster center it is closest to). Then a node containing $W \times H$ cells is represented by the histogram of the words that occur in its cells. The λ_{BAG} parameter for the node is an M dimensional vector. Examples of these histograms are shown in figure (3).

5 Hierarchical Models and Latent Structural SVM

The goal is to detect whether an object with class label y is present in an image region x. The model has latent variables $h = (V, z)$ (i.e. not specified in the training set), where V labels the mixture component and z specifies the positions of the graph nodes.

The model is specified by a function $[w \cdot \Phi(x, y, h)]$ where w is a set of parameters (to be learnt) and Φ is a feature vector. Φ has two types of terms: (i) Appearance terms $\Phi_A(x, y, h)$ which relate features of the image x to object classes y, components V and node positions z. (ii) Shape terms $\Phi_S(y, h)$ which specify the relationships between the positions of different nodes and which are independent of the image x. These potentials were described in the previous sections.

The *inference task* is to compute:

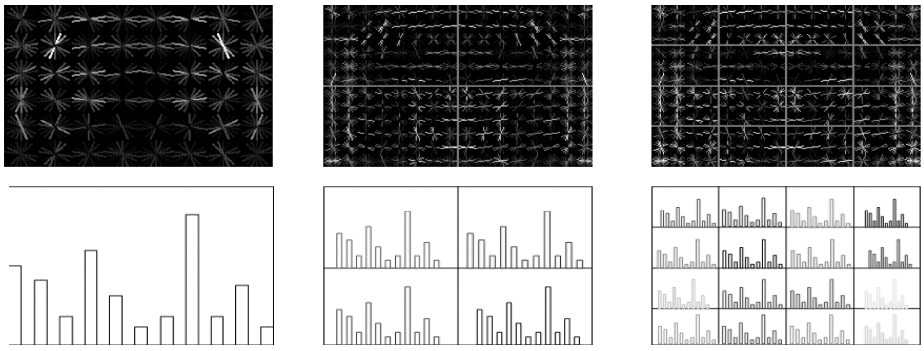

Figure 3: Appearance features. The top three panels show the Histogram of Oriented Gradients (HOGs). The bottom three panels show the Histogram Of Words (HOWs) extracted within different cells. The visual words are formed by using SIFT descriptors. In this case, HOWs are calculated using the first shape mask (regular rectangle). There columns from left to right correspond to the top to bottom levels of the active hierarchy.

$$F_\lambda(x) = \underset{y,h}{\text{argmax}}[\lambda \cdot \Phi(x, y, h)] \tag{4}$$

The *learning task* is to estimate the optimal parameters λ from a set of training data (x_1, y_1, h_1),...,(x_N, y_N, h_N) $\in \mathcal{X} \times \mathcal{Y} \times \mathcal{H}$.

We formulate the learning task as latent structural SVM learning. The object labels $\{y_i\}$ of the image regions $\{x_i\}$ are known but the latent variables $\{h_i\}$ are unknown (recall that the latent variables are the mask positions $\vec{p}$ and the model component V). The task is to find the weights λ which minimize an objective function $L(\lambda)$:

$$L(\lambda) = \frac{1}{2}||\lambda||^2 + C\sum_{i=1}^{N}\left[\max_{y,h}[\lambda \cdot \Phi_{i,y,h} + L_{i,y,h}] - \max_{h}[\lambda \cdot \Phi_{i,y_i,h}]\right] \tag{5}$$

where C is a fixed number, $\Phi_{i,y,h} = \Phi(x_i, y, h)$ and $L_{i,y,h} = L(y_i, y, h)$ is a loss function. For our object detection problem $L(y_i, y, h) = 1$, if $y_i = y$, and $L(y_i, y, h) = 0$ if $y_i \neq y$.

Solving the optimization problem in equation (5) is difficult because the objective function $L(\lambda)$ is non-convex (because the fourth term $-\max_h[\lambda \cdot \Phi_{i,y_i,h}]$ is a concave function of λ). Following Yu and Joachims we use the Concave-Convex Procedure (CCCP) (Yuille) which is guaranteed to converge at least to a local optimum. We briefly describe CCCP and its application to latent SVMs in section 5.1.

The kernel trick is used for the Bag of Words models – using a kernel like radial basis functions, which gives a similarity between image regions which have similar histograms of words. We did not use a kernel for the HOG (edge) features, or for the spatial relationship terms.

5.1 Optimization by CCCP

Learning the parameters λ of the model requires solving the optimization problem specified in equation (5). Following Yu and Joachims we express the objective function $L(\lambda) = f(\lambda) - g(\lambda)$ where $f(.)$ and $g(.)$ are convex functions given by:

$$f(\lambda) = \left[\frac{1}{2}||\lambda||^2 + C\sum_{i=1}^{N}\max_{y,h}[\lambda \cdot \Phi_{i,y,h} + L_{i,y,h}]\right] \tag{6}$$

$$g(\lambda) = -\left[C\sum_{i=1}^{N}\max_{h}[\lambda \cdot \Phi_{i,y_i,h}]\right] \tag{7}$$

The Concave-Convex Procedure (CCCP) (Yuille 2001) is an iterative algorithm which converges to a local minimum of $L(\lambda) = f(\lambda) - g(\lambda)$. When $f(.)$ and $g(.)$ take the forms specified by equation (6), then CCCP reduces to two steps which estimate the latent variables and the model parameters in turn (analogous to the two steps of the EM algorithm):

Step (1): Estimate the latent variables h by the best estimates given the current values of the parameters λ: $h^* = (V^*, \vec{p}^*)$.

Step (2): Apply structural SVM learning to estimate the parameters λ using the current estimates of the latent variables h:

$$\min_{\lambda} \frac{1}{2}||\lambda||^2 + C\sum_{i=1}^{N}\left[\max_{y,h}[\lambda \cdot \Phi_{i,y,h} + L_{i,y,h}] - \lambda \cdot \Phi_{i,y_i,h_i^*}\right] \tag{8}$$

We perform this structural SVM learning by the cutting plane method to solve equation (8).

A variant of CCCP, called *incremental CCCP* (iCCCP), has the advantage of less training data (L. Zhu et al. 2010).

5.2 Detection: Dynamic Programming

The inference task is to estimate $F_\lambda(x) = \text{argmax}_{y,h}[\lambda \cdot \Phi(x, y, h)]$ as specified by equation (4). The parameters λ and the input image region x are given. Inference is used both to detect objects after the parameters λ have been learnt and also to estimate the latent variables during learning (Step 2 of CCCP).

The task is to estimate $(y^*, h^*) = \text{argmax}_{y,h}[\lambda \cdot \Phi(x, y, h)]$. The main challenge is to perform inference over the node positions z since the remaining variables y, V take only a small number of values. Our strategy is to estimate the $\vec{p}$ by dynamic programming for all possible states of V and for $y = +1$, and then take the maximum. From now on we fix y, V and concentrate on z.

First, we obtain a set of values of the root node $z_1 = (u_1, v_1)$ by exhaustive search over all subwindows at different scales of the pyramid. Next, for each location (u_1, v_1) of the root node we use dynamic programming to determine the best best configuration z of the remaining nodes. To do this we use the recursive procedure:

$$F(x, z_a) = \sum_{b \in Ch(a)} \max_{z_b}\{F(x, z_b) + w \cdot \Phi_S(z_a, z_b)\} + \lambda \cdot \Phi_A(x, z_a) \tag{9}$$

where $F(x, z_a)$ is the max score of a subtree with root node a. The recursion terminates at the leaf nodes b where $F(x, z_b) = \Phi_A(x, z_b)$. This enables us to efficiently estimate the configurations z which maximize the discriminant function $F(x, z_1) = \max_z \lambda \cdot \Phi(x, z)$ for each V and for $y = +1$.

The bounding box determined by the position (u_1, v_1) of the root node and the corresponding level of the image pyramid is output as an object detection if the score $F(x, z_1) > 0$ – if $F(x, y) \leq 0$ we set $y = -1$.

5.3 Experimental Results

Some results of this model are shown in figure (4). A more sophisticated version of this approach – using more cells, more mixture components, better clustering to get words, soft assignments – was second in the Pascal object detection challenge in 2010 and first equal in 2011.

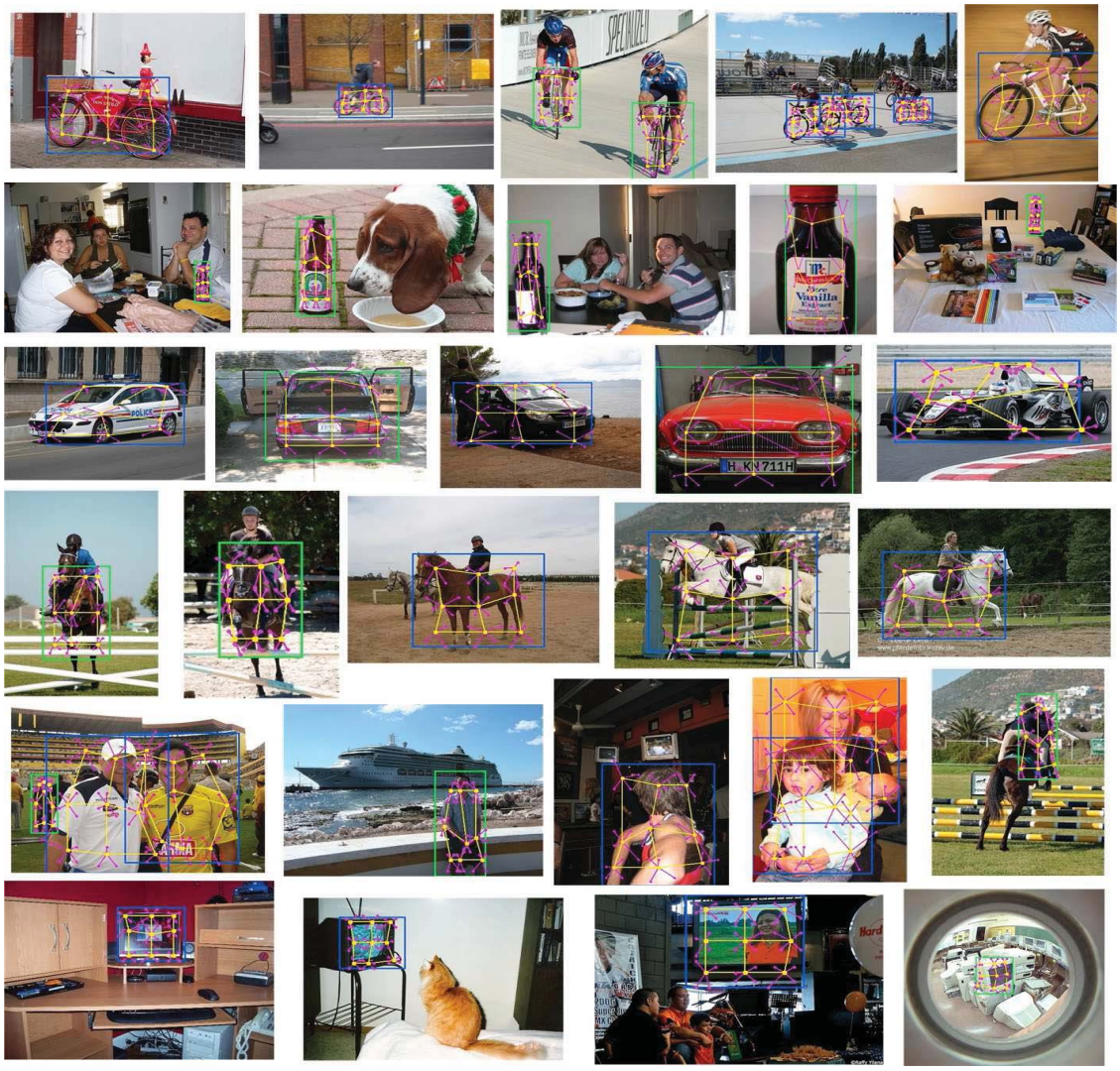

Figure 4: Examples on Pascal.