Incorporating Structural Alternatives and Sharing into Hierarchy for Multiclass Object Recognition and Detection

Anonymous CVPR submission

Paper ID 447

Abstract

This paper proposes a reconfigurable model to recognize and detect multiclass (or multiview) objects with large variation in appearance. Compared with well acknowledged hierarchical models, we study two advanced capabilities in hierarchy for object modeling: (i)"switch" variables(i.e. or-nodes) for specifying alternative compositions, and (ii) making local classifiers (i.e. leaf-nodes) shared among different classes. These capabilities enable us to account well for structural variabilities while preserving the model compact. Our model, in the form of an And-Or Graph, comprises four layers: a batch of leaf-nodes with collaborative edges in bottom for localizing object parts; the or-nodes over bottom to activate their children leaf-nodes; the and-nodes to classify objects as a whole; one root-node on the top for switching multiclass classification, which is also an or-node. For model training, we present an EM-type algorithm, namely dynamical structural optimization (DSO), to iteratively determine the structural configuration, (e.g., leaf-node generation associated with their parent or-nodes and shared across other classes), along with optimizing multi-layer parameters. The proposed method is valid on challenging databases, e.g., PASCAL VOC 2007 and UIUC-People, and it achieves state-of-the-arts performance.

1. Introduction

Object recognition is an area of active research in computer vision, and its performance has been improved substantially in recent years [6, 26, 18, 10, 4, 16, 14]. The objective of this work is to develop a novel hierarchical and reconfigurable model for multiclass object recognition, in the form of an And-Or graph representation, as Fig. 1 illustrates. We study two following issues that are often ignored or over-simplified in previous works.

Model reconfigurability. One key challenge in object modeling is to capture the large object variation in appearance and view/pose. Some recently proposed deformable part-based models [6, 18] handle this challenge by using hierarchical and contextual compositions, and achieve remarkable progresses. However, the structural configurations of these models are mainly fixed, e.g., the number of part detectors and the ways of composition. Inspired by And-Or graph models in [27, 8, 22], we develop the "switch variables", namely or-nodes, to specify alternative compositions in hierarchy. In detection, the or-nodes are used to activated its children leaf-nodes (i.e. local classifiers), accounting for intraclass variance. It worths mentioning that the association of or-nodes with its children leaf-nodes can be automatically determined in model training. In Fig. 1, the sheep head is localized by the leaf-node that is activated by its parent or-node.

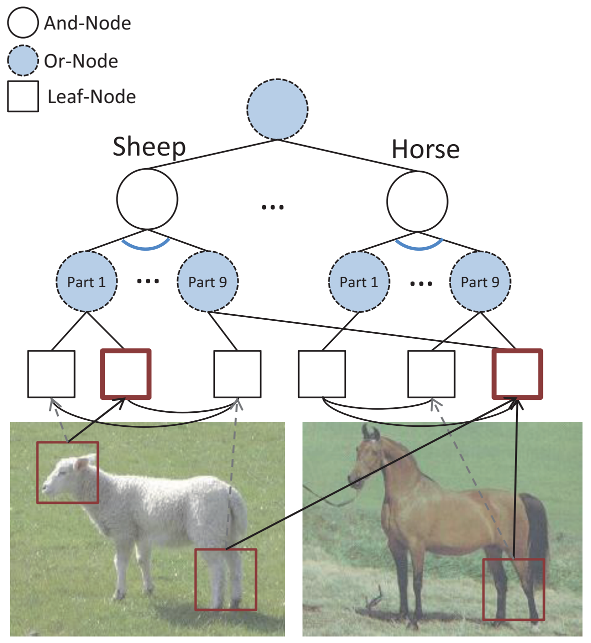

Figure 1. An example of the proposed 4-layer And-Or graph model for multiclass object recognition. Parts of the model for sheep and horse are shown. The squares in bottom represent the leaf-nodes, which can be shared among different classes(e.g. the leaf-node for localizing legs are shared between sheep and horse). The or-nodes over bottom are used to activate their children leaf-nodes, tackling the local ambiguities.

Model sharing. In the context of multiclass object recognition, existing systems commonly treat different classes as unrelated entities, (by [15]). According to acknowledged studies [19, 17, 14], sharing information among different classes can boost model performance in general and alleviate the requirement of a large amount of training data. Recently, Salakhutdinov et al. [15] propose a learning-to-share framework that allows rare objects to borrow statistical strength from other related classes, and demonstrate impressive results. It inspires us to make structure shared in the And-Or graph model, for adapting the task of multiclass recognition. In our method, the leaf-nodes are sharable among different classes so that we keep the model compact to represent multiple object categories. For example, in Fig. 1, the part of feet in category horse and sheep have similar appearances, and thus can be both detected by the leaf-node shared across the two classes.

The key contribution of this work is a novel And-Or graph model for multiclass object recognition, by addressing the both above issues. Without loss of generality, we define our four layered model, as Fig.1 illustrates. The leaf-nodes (denoted by squares) in the bottom are discriminative classifiers for detecting object parts. The or-nodes (denoted by dashed circles) over in the third layer are used to activate one of its children leaf-nodes in detection, which are allowed to slightly perturb for capturing deformations. The and-nodes (denoted by solid circles) in the second layer are global classifiers for object classes. The root-node at top is for switching multiclass recognition, which is also an or-node. In addition, we define the collaborative edges (denoted by curve connections) to encode intraclass (part-level) relations, and interclass contexts are modeled in the similar way as the edges connect the and-nodes also.

One non-trivial problem in model training is to automatically determine the model structure without requiring elaborate supervision and initialization. In our method, we propose a novel algorithm for this problem, namely Dynamical Structural Optimization (DSO), motivated by the recently proposed structural optimization methods [23, 12]. It is designed in the EM-type iterating with three steps. (i) Estimate model latent variables for optimization, according to parameters from the previous iteration. (ii) Reconfigure the model structure by clustering. In this step, we produce leaf-nodes associated with their parent or-nodes and make leaf-nodes shared across classes. (iii) Check the acceptance for the newly generated model structure, and update the model parameters.

Due to large variance among classes, it would be intractable to train the classes altogether by pooling all samples from different classes into a bag. In this work, we first partition all classes into several groups by a data-driven approach, in order to reduce the computational complexity for model sharing. Then we train the models for object classes in each group. For example, we can easily decide to put sheep and horses into one group and train the multiclass model by sharing. Afterwards, the trained models for all groups are further combined into the complete one, by reweighing parameters of all the models. And the collaborative edges are also learned during this step.

2. Related Work

Traditional multiclass object detectors are trained in a one-vs-all manner, where each object category are trained independently. These methods often rely on large amount of training data. A pioneer work [19] is proposed to learn shared features among classes and improve the classifier in both effectiveness and efficiency. Opelt et al. [13] further incorporate the incremental learning with classifier sharing. To discover hierarchical structures of object categories, the Hierarchical Latent Dirichlet Allocation (hLDA) model is presented in [17]. The efficiency of multiclass object recognition can be significantly improved by integrating taxonomies with object hierarchy [7, 9].

To tackle realistic challenges in object recognition, many deformable part-based methods are developed by latent structural learning recently [6, 26, 18, 16], and demonstrate very promising results. These models are also extended to multiclass recognition and detection [4, 15, 14]. For example, Razavi et al. [14] present the multiclass Hough Forest combing with the part-based models; Desai et al. [4] further incorporate the context information into hierarchy, and predict a structured labeling for each image during detections. However, the commonly used part-based models are often defined in a tree structure, whose configurations are fixed during the learning and detection, and may have problems on handling objects with large appearance and structure variations.

And-Or graph models are first proposed for modeling complex visual patterns by Zhu and Mumford [25]. Its general idea, using And/Or nodes to account for structural compositions and variabilities in hierarchy, has been applied in several vision tasks, e.g., human parsing [22, 27] and scene understanding [24]. These approaches often require supervised learning or manually initialization. Fidler et al. [8] propose to train the And-Or graph for multiclass shape-based detection in a generative way, and extensively discuss the learning strategies. Motivated by these works, we propose an alternating way to discriminatively train the And-Or graph model for multiclass object recognition, and achieve superior performance on very challenging databases, e.g., PASCAL VOC 2007 and UIUC-People.

3. And-Or Graph Model

Our multiclass object model is constructed in the form of an And-Or graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ contains three types of nodes, $\mathcal{E}$ represents the collaborative edges. The root-node is indexed as 0, indicating the switch among classes. The and-nodes are indexed by $r = 1, ..., m$, each representing one category. For each and-node, there are 9 or-nodes arranged in a layout of 3×3 blocks to represent object parts, and we index all the or-nodes as $j = m+1, ..., 10m$. The leaf-nodes in the fourth layer are indexed by $i = 10m+1, ..., 10m+n$, where n is the leaf-node number dynamically adjusted during training. For notation simplicity, we define that $m' = 10m+1, n' = 10m+n$, and $i \in ch(j)$ indexes a child node of node j. The details of our model are presented as follows.

Sharable Leaf-node: The leaf-nodes $L_i, i = m', ..., n'$ are local classifiers for object parts, and they can be shared among different classes. Specifically, if a leaf-node is affiliated to the j-th or-node, it is also possible to be shared by the or-nodes in other classes indexed by $j + 9 \times k$, where $k \in \{1, 2...\}$. We denote the location of leaf-node L_i as P_i, which is determined by its parent or-node activating L_i during inference. The response of L_i is defined as,

$$R_i^l(X, P_i) = \omega_i^l \cdot \phi^l(X, P_i). \tag{1}$$

In our implementation, a HOG [3] pyramid is built across different image scales as in [6]. $\phi^l(X, P_i)$ is the HOG feature extracted from image X at position P_i, and ω_i^l is a parameter vector.

Or-node: The or-nodes $U_j, j = m+1, ..., 10m$ in the third layer are "switch" variables to select (activate) their children. For each leaf-node L_i, we define an variable $V_i \in \{0, 1\}$ to represent the activation during inference. An indicator vector is then composed for each or-node U_j: $\mathbf{V}_j = (V_{i_1}, V_{i_2}, ...)$, where $i_k \in ch(j)$ and $\|\mathbf{V}_j\| = 1/0$. Note that $||\mathbf{V}_j|| = 1$ only when one of the leaf-nodes is activated under U_j. The response of U_j is thus defined as,

$$R_j^u(X, P_j, \mathbf{V}_j) = \sum_{i \in ch(j)} R_i^l(X, P_j) \cdot V_i, \tag{2}$$

where P_j denotes the position of U_j, and it is allowed to perturb slightly during inference. We define a feature for object deformation as $\phi^s(P_r, P_j) = (dx, dy, dx^2, dy^2)$, where (dx, dy) represents the displacement of U_j relative to its anchor position that is determined by the position of its parent P_r. The response of the deformation is defined as,

$$R_j^s(P_r, P_j) = \omega_j^s \cdot \phi^s(P_r, P_j). \tag{3}$$

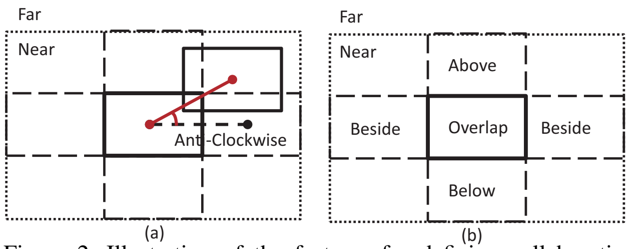

Figure 2. Illustration of the features for defining collaborative edges. (a) shows feature $\psi^l(P_i, P_{i'})$ between leaf-nodes; (b) shows feature $\psi^a(P_r, P_{r'})$ between and-nodes.

And-node: The and-nodes $A_r, r = 1, ..., m$ are global classifiers for objects. Suppose A_r is placed at P_r during detection, we extract the HOG feature for the and-node $\phi^a(X, P_r)$ at half the resolution of the feature extracted for leaf-nodes. We define the response for A_r with its parameters ω_r^a, as,

$$R_r^a(X, P_r) = \omega_r^a \cdot \phi^a(X, P_r). \tag{4}$$

Root-node: The root-node on the top is an or-node for switching different classes, i.e. choosing its children and-nodes. Similarly with defining the or-nodes, for each and-node A_r, we also define the activation for it as $V_r \in \{0, 1\}$, and the indicator vector for root-node is $\mathbf{V}_0 = (V_1, ..., V_m)$ and $\|\mathbf{V}_0\| = 1$, i.e, only one children is selected.

Collaborative edge: There are two types of collaborative edges in our model, representing the spatial co-occurrence between different leaf-nodes as well as between different and-nodes. For the collaborative edges between leaf-nodes, we introduce a 4-bin binary feature $\psi^l(P_i, P_{i'})$. Each bin of $\psi^l(P_i, P_{i'})$ represents one of the relations: *clockwise, anti-clockwise, near* and *far* between two leaf-nodes L_i and $L_{i'}$. As Fig. 2(a) illustrates, the bold rectangle in the middle represents the location of L_i. If the center of $L_{i'}$ is localized in the dotted rectangle, it is *near* the L_i, otherwise it is *far* from L_i. We connect the initial centers of L_i and $L_{i'}$ with the dashed line and the red line represents their layout after accounting for deformation. Then we use two bins to indicate either *clockwise* or *anti-clockwise* for the angle between the dashed line and the red line. We thus define the response of the collaborative edge between two leaf-nodes as,

$$\Gamma_{i,i'}^l(P_i, P_{i'}) = \alpha_{i,i'}^l \cdot \psi^l(P_i, P_{i'}), \tag{5}$$

where $\alpha_{i,i'}^l$ is a 4-bin parameter vector. Motivated by [4], we define a 6-bin binary feature $\psi^a(P_r, P_{r'})$ representing the contextual relations: *above, below, beside, overlap, near* and *far* between two objects. As Fig. 2(b) illustrates, the bold rectangle in the middle represents the window of A_r. And the dashed and dotted rectangles represent the bins to be set as 1 if the center of $A_{r'}$ is inside. The response of the collaborative edge between two and-nodes is defined as,

$$\Gamma_{r,r'}^a(P_r, P_{r'}) = \alpha_{r,r'}^a \cdot \psi^a(P_r, P_{r'}), \tag{6}$$

where $\alpha_{r,r'}^a$ is a 6-bin parameter vector. In practice, we only connect the two leaf-nodes whose parent or-nodes are adjacent to each other in spatial domain. And the and-nodes are connected across classes.

4. Inference

Given an image, the task for inference is to localize all the multiclass objects with the model. For simplicity, we notate the vector of selections for and-nodes together with leaf-nodes as $\mathbb{V} = \langle V_1, ..., V_m, V_{m'}, ..., V_{n'} \rangle$, and the vector of placements as $\mathbb{P} = \langle P_1, ..., P_{10m} \rangle$.

A subgraph of the And-Or graph, rooted at one of the and-nodes, can be regarded as a detector for one class. For each subgraph, we compute its scores by sliding the detection sub-window at different positions and scales of the image. It is a procedure integrating the local testing and binding testing as follows.

Local testing: For a subgraph model rooted at A_r (i.e., $V_r = 1$) and placed at P_r of the image, we assume a hypothesis $\mathbb{V}$ for leaf-node selections. Then the placement of each part can be obtained by incorporating Eq.(2) and Eq.(3):

$$\begin{aligned}\widetilde{P}_j &= \max_{P_j}(R^u_j(X, P_j, \mathbf{V}_j) - R^s_j(P_r, P_j)) \\ &= \max_{P_j}(\sum_{i \in ch(j)} R^l_i(X, P_j) \cdot V_i - R^s_j(P_r, P_j)),\end{aligned} \tag{7}$$

where $R^l_i(X, P_j)$ represents the leaf-node response, and we can share these responses among different classes by calculating them at the beginning of inference. Then the score of local testing is calculated as:

$$S^l_r(X, \widetilde{\mathbb{P}}, \mathbb{V}) = \sum_{j \in ch(r)} (R^u_j(X, \widetilde{P}_j, \mathbf{V}_j) - R^s_j(P_r, \widetilde{P}_j)). \tag{8}$$

Binding testing: We obtain the response over the and-node $R^a_r(X, P_r)$ with Eq.(4). And for each hypothesis $\mathbb{V}$, we compute the scores of intra-class contextual relations between the selected leaf-nodes via Eq.(5). Then the binding score is calculated as:

$$S^a_r(X, \widetilde{\mathbb{P}}, \mathbb{V}) = R^a_r(X, P_r) + \sum_{i,i'=m'}^{n'} \Gamma^l_{i,i'}(\widetilde{P}_i, \widetilde{P}_{i'}) \cdot V_i \cdot V_{i'}, \tag{9}$$

where the leaf-node location is set as $\widetilde{P}_i = \widetilde{P}_j$ for $i \in ch(j), ||\mathbf{V}_j|| = 1$. By integrating these two procedures, we select the best $\mathbb{V}$ as the score of detection via the subgraph rooted at A_r:

$$S^g(X, r, P_r) = \max_{\mathbb{V}}(S^l_r(X, \widetilde{\mathbb{P}}, \mathbb{V}) + S^a_r(X, \widetilde{\mathbb{P}}, \mathbb{V})). \tag{10}$$

After the detections for all subgraphs, we can represent the image as a collection of K scored sub-windows, overlapping at different scales. Our objective is to label them with $Y = \{y_1, ..., y_K\}$, where $y_i \in \{1, ..., m\}$ represents object classes and $y_i = -1$ the background. The multiclass detection score in the image can be defined by combining Eq.(10) and Eq.(6),

$$S(X) = \max_{Y}(\sum_{k=1}^{K} S^g(X, y_k, P^k) + \sum_{k,k'=1}^{K} \Gamma^a_{y_k, y_{k'}}(P^k, P^{k'})), \tag{11}$$

where P^k indicates the position of the kth sub-window of the image. The optimization of Eq.(11) can be solved by the greedy forward search mentioned in [4]. We define a instance set INS $= \{(k, y)\}$ indicating that $y_k = y$ for $\forall(k, y) \in$ INS and $y_k = 0$ otherwise. Then the greedy method is performed as Algorithm 1.

5. Dynamical And-Or Graph Learning

The training of our And-Or graph is a two stages procedure: (i) estimating model structure (without edges) and parameters for each object group; (ii) combining models and learning collaborative edges.

Algorithm 1 Greedy Forward Inference

```
Input:
  The detections scores $S^g(X, r, P_r)$ for image sub-windows.
Output:
  Instance set INS, detection score $S$.
Initialization:
  INS = {}, $S = 0$, $\delta(k, y) = S^g(X, y, P^k)$ for $k \in \{1, ..., K\}$.
repeat
  1. $(\widetilde{k}, \widetilde{y}) = \text{argmax}_{(k,y)} \delta(k, y)$ for $\forall(k, y) \notin$ INS.
  2. INS = INS $\cup$ $(\widetilde{k}, \widetilde{y})$.
  3. $S = S + \delta(\widetilde{k}, \widetilde{y})$.
  4. $\delta(k, y) = \delta(k, y) + \Gamma^a_{y,\widetilde{y}}(P^k, P^{\widetilde{k}}) + \Gamma^a_{\widetilde{y},y}(P^{\widetilde{k}}, P^k)$.
until $\delta(\widetilde{k}, \widetilde{y}) \leq 0$ and $S$ stops increasing.
```

To reduce the computational cost for model sharing, we first divide the object classes into several groups as a data-driven initialization, and train the multiclass model for each group. Afterwards, we combine the trained models together to construct the final And-Or graph model.

The learning for stage (i) is an EM-type procedure incorporating structure reconfiguration and parameter estimation. During each iteration, our algorithm dynamically create and remove leaf-nodes associated with their parent or-nodes, and share leaf-nodes among classes. More precisely, a leaf-node is created to better handle the intra-class variance (Fig. 3(b)); A leaf-node is removed if there is another similar one (Fig. 3(d)); A leaf-node is shared as it can capture the similar appearances for other classes (Fig. 3(c)).

5.1. Data Driven Initialization

Suppose the number of all classes is M, we partition them into several groups as a data driven initialization for training. The partition is based on the similarity between two classes, and we calculate the similarity as follows.

(I) We first learn a two-layer deformable part-based model [6] $\mathcal{T}^k = \{T^k_i\}$ for all classes, where T^k_i represents one part classifier for k-th class. And we apply $\mathcal{T}^k$ to perform detection on the positive training samples in every class. During the detection, each T^k_i extracts a set of image patches from different samples, and we group these patches into a cluster Ω^k_i. Note that the size of image patches detected by T^k_i is (h^k_i, w^k_i). For all Ω^k_i, we further merge them into a few new sets, each of which contains image patches of similar size (h^k_i, w^k_i). In each of the new sets, we describe the image patches with the HOG descriptor and group them into several clusters by using ISODATA algorithm with Euclidean distance.

(II) Afterwards, a matrix $\mathcal{M}$ is defined to represent the similarity between M classes. In each set of image patches, if there are patches from class j and k falling into the same cluster, we set $\mathcal{M}(j, k) \leftarrow \mathcal{M}(j, k) + 1$. Two classes j and k are assumed to share their models if $\mathcal{M}(j, k) > \sigma$, where σ is a threshold set as $M/3$ empirically.

(III) Based on the calculated $\mathcal{M}$, we assign the classes

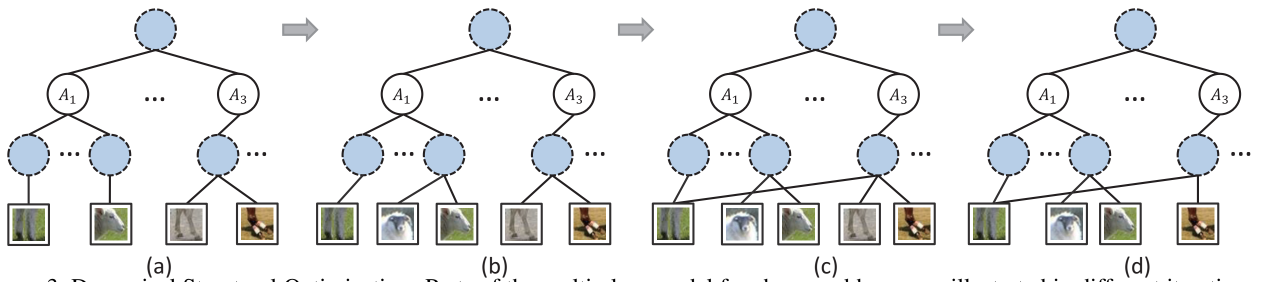

Figure 3. Dynamical Structural Optimization. Parts of the multi-class model for sheep and horse are illustrated in different iterations. (a) The model structure after the first iteration; (b) A new leaf-node is created to recognize the head of sheep; (c) A leaf-node for sheep leg is shared with the horse; (d) A leaf-node for horse leg is removed.

that are possibly shared into the same group $\mathbb{S}$. We thus obtain a few groups as $\{\mathbb{S}_1, ..., \mathbb{S}_c\}$. We denote that each $\mathbb{S}$ has $|\mathbb{S}|$ classes, and we discuss the training method for each $\mathbb{S}$ in the following section.

5.2. Optimization Formulation

Given an object group $\mathbb{S}$, we train a multiclass model without collaborative edges, which is a procedure integrating structure reconfiguration and parameter estimation. Suppose there are a set of N training samples (X_1, y_1),...,(X_N, y_N) in $\mathbb{S}$, where X is the image, $y \in \{1, ..., |\mathbb{S}|\}$ labels the object classes, and $y = -1$ labels the background. At the beginning of training, we initialize the multiclass model with $m = |\mathbb{S}|$ and-nodes and one leaf-node for each or-node. The detection score of this model can be represented as the maximization of Eq.(10) over m and-nodes, by setting edge parameters to zero,

$$S^t(X) = \max_{1 \le r \le m} S^g(X, r, P_r)$$
$$= \max_{\mathbb{P},\mathbb{V}} (\sum_{i=m'}^{n'} \omega_i^l \cdot \phi^l(X, P_i) \cdot V_i - \sum_{r=1}^{m} \sum_{j \in ch(r)} \omega_j^s \cdot \phi^s(P_r, P_j) \cdot ||\mathbf{V}_j||$$
$$+ \sum_{r=1}^{m} \omega_r^a \cdot \phi^a(X, P_r) \cdot V_r), \tag{12}$$

where the first two terms represent the response of local testings, and the last term is the and-node response. For simplicity, we refer $H = (\mathbb{P}, \mathbb{V})$ as the latent variables, then we redefine Eq.(12) in a discriminative form as,

$$S^\omega(X) = argmax_{(y,H)}(\omega \cdot \phi(X, y, H)), \tag{13}$$

where ω includes the complete model parameters of current model, $\phi(X, y, H)$ is defined as,

$$\phi(X, y, H) = \begin{cases} \phi(X, H) & \text{if } V_y = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{14}$$

and $\phi(X, H)$ is the overall feature vector.

The function (13) can be learned by applying structural SVM with latent variables,

$$\min_\omega \frac{1}{2}\|\omega\|^2 + C\sum_{k=1}^{N}[\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y))$$
$$- \max_H(\omega \cdot \phi(X_k, y_k, H))], \tag{15}$$

where C is a penalty parameter set as 0.005 empirically, and we define the loss function $\mathcal{L}(y_k, y) = 0$ when $y_k = y$, and $\mathcal{L}(y_k, y) = 1$ if $y_k \neq y$. In recent works [26, 11], the CCCP [23] method is applied to solve the non-convex optimization, which provides an iterative approach to achieve a local minima. However, in these methods, the model structure configuration is assumed to be fixed, e.g., without or-nodes. Motivated by these works, we propose a Dynamical Structural Optimization (DSO) method to train out model.

5.3. Dynamical Structural Optimization

To optimize the objective Eq.(15), we transform it into a concave and convex form following [26],

$$\min_\omega[\frac{1}{2}\|\omega\|^2 + C\sum_{k=1}^{N}\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y))]$$
$$- C\sum_{k=1}^{N}\max_H(\omega \cdot \phi(X_k, y_k, H)) \tag{16}$$
$$= \min_\omega[f(\omega) - g(\omega)], \tag{17}$$

where the first two terms in (16) are represented by $f(\omega)$ and $g(\omega)$ is the other term. Then we present our 3-step Dynamical Structural Optimization method as follows.

(I) Suppose we are in the iteration t, and ω_t is the parameter vector updated in the previous iteration. We first find a hyperplane q_t to upper bound $-g(\omega)$ in (17),

$$-g(\omega) \le -g(\omega_t) + (\omega - \omega_t) \cdot q_t, \forall\omega. \tag{18}$$

We calculate q_t by finding the optimal latent variables $\widetilde{H}_k = argmax_H(\omega_t \cdot \phi(X_k, y_k, H))$. That is, we apply the current model to perform detections on the training samples, and the hyperplane is constructed as $q_t = -C\sum_{k=1}^{N}\phi(X_k, y_k, \widetilde{H}_k)$.

(II) We adjust the model by structural reconfiguration and sharing, and it is performed on each one of the 9 object parts over m classes, independently. Given a variable vector $\widetilde{H}_k$ for a sample, we can obtain the activation of leaf-nodes and the image patches detected via them. For each leaf-node L_i, we group the patches detected via it from all samples into a cluster Ω_i, and the size of these patches is (h_i, w_i).

We index the nine object parts by $j (m+1 \le j \le m+9)$. For the j-th part, we pool the clusters whose corresponding leaf-nodes are associated to or-nodes $U_{j+9k} (0 \le k < m)$ from m classes together. Then these clusters are further merged into a few new sets, each of which contains patches of similar size (h_i, w_i). For each new set, we describe the image patches with HOG descriptor and perform clustering on them by applying ISODATA with Euclidean distance.

After the clustering, the leaf-nodes are reconfigured as: If a cluster is newly generated, we create a new leaf-node accordingly; we remove a leaf-node if there are few image patches in the corresponding cluster. For a cluster Ω_i, if there are images patches localized by U_j(in step(**I**)), we associate the leaf-node L_i to U_j. Thus L_i is shared by different classes for different associations.

The feature vector ϕ of each sample is also adjusted according to the clustering result. Recall that the HOG vector of an image patch is part of ϕ, and the patches in the same cluster are represented with the same bins in ϕ. We present a toy example in Fig. 4 for illustration. The sub-vector $\langle\phi_5, ..., \phi_8\rangle$ of sample X_3 is grouped from one cluster to another; then the feature bins are moved from $\langle\phi_5, ..., \phi_8\rangle$ to $\langle\phi_1, ..., \phi_4\rangle$, as (a) and (c) shows. We define the new feature vector for each sample after clustering as $\phi^d(X_k, y_k, \widetilde{H}_k)$, then the new hyperplane in step **(I)** is reconstructed as $q_t^d = -C\sum_{k=1}^{N}\phi^d(X_k, y_k, \widetilde{H}_k)$.

(III) With the current model structure and q_t^d we can learn the model parameters by solving,

$$\omega_t^d = argmin_\omega(f(\omega) + \omega \cdot q_t^d). \tag{19}$$

By substituting $f(\omega)$ with the first two terms defined in Eq.(16), we can re-write Eq.(19) as,

$$\min_\omega \frac{1}{2}\|\omega\|^2 + C\sum_{k=1}^{N}[\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y)) - \omega \cdot \phi^d(X_k, y_k, \widetilde{H}_k)]. \tag{20}$$

The optimization of Eq.(20) can be solved by standard structural SVM. After that, we can calculate the energy of the objective by $E(\omega_t^d) = f(\omega_t^d) - g(\omega_t^d)$.

If $E(\omega_t^d) < E(\omega_t)$, we accept the new model structure and have $\omega_{t+1} = \omega_t^d$. Otherwise, we keep the model configuration as it is in the previous iteration, and continue to perform parameter optimization without structure reconfiguration as Eq.(19): $\omega_{t+1} = argmin_\omega(f(\omega) + \omega \cdot q_t)$.

In this way, we ensure the optimization objective in Eq.(17) continuing to decrease in iterations. Thus, the algorithm keeps iterating until the objective converges.

5.4. Model Combination

After training the multiclass models for each object group in $\{\mathbb{S}_1, ..., \mathbb{S}_c\}$, we combine them together into a complete one for all the object categories. Intuitively, the root-nodes from each group are first merged into the final top root-node, so that the original and-nodes are all associated to the new root-node. Then we introduce a n' dimension vector $\beta = (\beta_1, ..., \beta_{n'})$ to re-weight the parameters for the newly generated model. Meanwhile, the collaborative edges defined in Eq.(5) and Eq.(6) are trained as well.

For simplicity, we shorten the responses for leaf-node, or-node deformation and and-node as $R_i^l(k) = R_i^l(X, P_i^k)$,

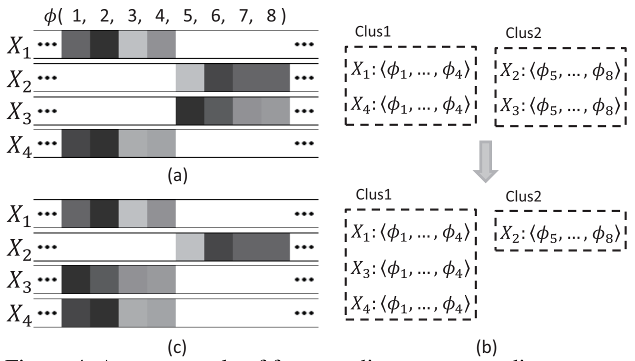

Figure 4. A toy example of feature adjustment according to structural clustering. Parts of 4 feature vectors associated to two different leaf-nodes are presented. (a)shows the feature vectors generated after Step (**I**), whose value is indicated by the intensities of bins; (b)shows the structural re-clustering: The feature $\langle\phi_5, ..., \phi_8\rangle$ of X_3 are moved from Cluster 2 to Cluster 1; (c)updates the feature vectors according to clustering results.

$R_j^s(k) = R_j^s(P^k, P_j^k)$ and $R_{y_k}^a = R_{y_k}^a(X, P^k)$. Given an image X, the objective function $S(X)$ of multiclass recognition defined in Eq.(11) is reformulated as,

$$\max_Y \sum_{k=1}^{K}[\sum_{i=m'}^{n'}\beta_i \cdot R_i^l(k) \cdot V_i^k - \sum_{j\in ch(y_k)}\beta_j \cdot R_j^s(k) + \beta_{y_k} \cdot R_{y_k}^a + \sum_{i,i'=m'}^{n'}\alpha_{i,i'}^l \cdot \psi^l(P_i^k, P_{i'}^k) \cdot V_i^k V_{i'}^k + \sum_{k'=1}^{K}\alpha_{y_k,y_{k'}}^a \cdot \psi^a(P^k, P^{k'})],$$

where the first two terms represent the local testing score, the next two represent the binding testing score, and the last one accounts for edge responses between and-nodes.

For the training, we collect a set of images containing multiclass objects, each of which is labeled with $Y = \{y_1, ..., y_K\}$. Given each image, we first obtain the latent variables $\widetilde{H}_k$ with Eq.(13) by fixing y_k, and the responses for each part are derived meanwhile. We can then use the responses $R_i^l(k)$, $R_j^s(k)$ and $R_{y_k}^a$ as part of the input feature, and train the parameters β, α^l and α^a by standard structural SVM. Here the loss function for training is defined as $\mathcal{L}'(Y, Y') = \mathcal{K} - tp$, where $\mathcal{K}$ indicates the number of objects in groundtruth Y, and tp is the number of true positives in Y' according to Y.

Afterwards, the parameters for leaf-nodes $(A_i, i = m', ..., n')$, or-node deformations$(U_j, j = m+1, ..., 10m)$ and and-nodes$(A_r, r = 1, ..., m)$ are re-weighed as: $\omega_i^l = \beta_i \cdot \omega_i^l$, $\omega_j^s = \beta_j \cdot \omega_j^s$ and $\omega_r^a = \beta_r \cdot \omega_r^a$.

6. Experiments

We evaluate our method on two challenging datasets: UIUC people [20] and PASCAL VOC 2007 [5].

Dataset and Setting. The UIUC people dataset contains 593 images(296 for training, 297 for testing), and most of them contain one person playing badminton. For PASCAL VOC 2007 dataset, there are 9963 images of 20 object cat-

	Ours(full)	Ours-3	[22]	[1]	[6]	[2]
Accuracy	**0.845**	0.818	0.668	0.506	0.486	0.458

Table 1. Detection accuracies on UIUC people dataset.

egories with 5011 images for training and 4952 images for testing. In both datasets, we represent each object category with two views, i.e. each object category is specified by two and-nodes in our model. Hence, we perform 2-class recognition on UIUC people dataset, and 40-class recognition on PASCAL VOC 2007 dataset. During evaluation, we adopt PASCAL Challenge criterion: a detection is considered as correct only if the intersection over union with the groundtruth bounding-box is at least 50%. All our experiments are carried out on a PC with Core Duo 3.0 GHZ CPU and 16GB memory. We denote our fully implemented model as "Ours(full)", since we will simplify the model in different settings for empirical study.

6.1. Experimental Results

UIUC people dataset. For model training, it takes 11 iterations and around 6 hours to converge in optimization. And the time for detection on a image is about 5 seconds. We compare our model with the state-of-the-arts human detectors [22, 1, 2, 6], some of which used manually labeled model. The detection accuracy is calculated as [22]: only the detection with the highest score on the image is considered. As Table. 1 reports, our approach reaches the detection accuracy of 84.5%, outperforming other methods. Moreover, we demonstrate the advantage of our model for handing object variations in detection in Fig. 6. We visualize the detectors generated by our trained model in the form of HOG patterns. The detectors for and-nodes and leaf-nodes are shown in Fig. 6 (a). Note that some of the leaf-nodes are shared for capturing similar appearances. Two detectors, composed by 9 activated leaf-nodes, are visualized in Fig. 6 (b). The two detectors are generated when recognizing the images beside them. The results show that our model can generate alterable detectors to adapt diverse object appearances and poses.

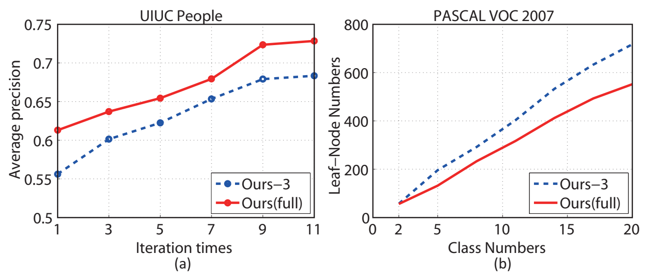

Figure 5. Extensive experiments for discussion. "Our-3" indicates a simplified model without sharing leaf-nodes. (a) shows the APs on UIUC people dataset. (b) represents the leaf-node numbers with the increasing of object categories on PASCAL VOC 2007 dataset.

PASCAL VOC 2007 dataset. To train the 40-class model on the database, it takes 25 $\sim$ 30 iterations in 30 $\sim$ 34 hours. On average, it takes 92 seconds for detecting all 20 classes of objects on one input image. We then calculate the average precision (AP) to evaluate our method. As shown in Table. 2, our method achieves the mean AP(mAP) of 34.7%, which is highly competitive to the state-of-the-arts methods: 29.0% [10], 29.2% [14], 29.6% [26], 32.1% [21] and 26.8% [6]. We also notice that there is a significant improvement achieving mAP of 37.7% [18] recently, by employing multi-kernels classification into detection.

6.2. Evaluation for Model Sharing

To analyze the effectiveness of sharing leaf-nodes, we disable the process for model sharing in training so that we obtain the non-sharing And-Or graph model, named "Ours-3" on UIUC people dataset. As Table. 1 reports, "Ours-3" achieve detection accuracy of 81.8%, 2.7% less than the fully implemented model. We also compare the APs of these two models in Fig. 5 (a), in which the APs are visualized with the increasing of iteration numbers for model training. Each AP for a specific iteration number is obtained by testing the model that is trained by the number of iterations. And the APs of "Ours(full)" and "Ours-3" achieve 72.8% and 68.3%, respectively, after 11 iterations.

We further evaluate the benefits of collaborative edges in our model. We simplify the model by setting the parameters of collaborative edges to zero. Two models are generated under this setting, denoted by "Ours-1" and "Ours-2", by turning on/off the leaf-nodes sharing among classes, respectively. We execute this experiment on PASCAL VOC 2007 dataset. As Table. 2 reports, the mAPs of "Ours-1" and "Ours-2" achieve 33.6% and 31.8%, respectively.

We consider the model complexity, represented by the number of leaf-nodes, could be effectively reduced by model sharing. Thus, we also present an experiment to show the numbers of leaf-nodes in model training, with the increasing of object categories, in Fig. 5(b). Precisely, we obtain 552 leaf-nodes for 20 object categories on PASCAL VOC 2007 dataset, less than 717 leaf-nodes by "Ours-3" model.

7. Conclusion

This paper introduces a novel method for multiclass object detection and recognition, in the form of And-Or graph. Our model is shown to handle well the challenges in large variance object recognition. Moreover, we also illustrate the benefits of information sharing among classes, which leads to a more compact and better model. Since our learning method(SDO) is very general, it can be extended to many other vision tasks.

References

[1] M. Andriluka, S. Roth, and B. Schiele, Pictorial structures revisited: People detection and articulated pose estimation, In *CVPR*, 2009. 7

[2] L. Bourdev, S. Maji, T. Brox, and J. Malik, Detecting people using mutually consistent poselet activations, In *ECCV*, 2010. 7

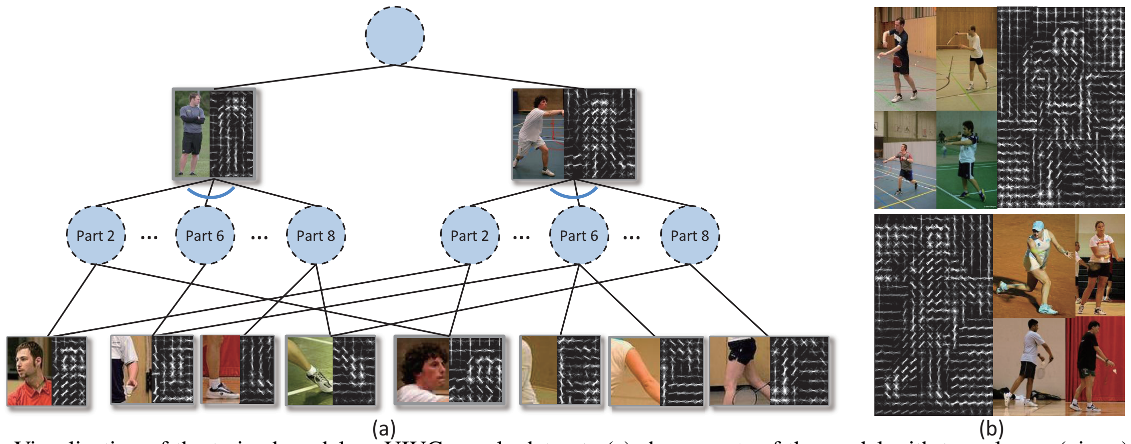

Figure 6. Visualization of the trained model on UIUC people dataset. (a) shows parts of the model with two classes (views), in which we visualize the detectors (in the form of HOG patterns) for the and-nodes and leaf-nodes based on the learned parameters, alone with the example images recognized by the model. (b) visualizes two detectors that are composed by 9 activated leaf-nodes. The two detectors are generated, respectively, when recognizing the images beside them. The detectors for and-nodes are not visualized here.

	plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	Avg.
Ours(full)	32.5	**60.1**	11.1	**16.0**	**31.0**	**50.9**	**59.0**	26.1	**21.2**	26.5	25.4	16.4	**61.7**	**48.3**	**42.2**	**16.1**	28.2	30.1	44.6	46.3	**34.7**
Ours-1	31.4	58.4	10.8	15.7	28.3	49.3	58.3	25.4	20.4	25.5	**25.7**	16.3	58.9	46.0	40.6	15.5	26.3	**30.3**	42.2	46.4	33.6
Ours-2	30.4	59.5	7.0	13.8	28.1	47.3	58.0	21.9	20.6	25.8	21.5	13.1	56.7	43.9	38.2	13.9	21.9	29.6	38.8	46.5	31.8
MC [10]	33.4	37.0	15.0	15.0	22.6	43.1	49.3	**32.8**	11.5	**35.8**	17.8	16.3	43.6	38.2	29.8	11.6	**33.3**	23.5	30.2	39.6	29.0
HF [14]	26.0	56.0	10.0	11.0	21.0	47.0	50.0	16.0	19.0	23.0	20.0	12.0	51.0	45.0	37.0	12.0	17.0	29.0	41.0	38.0	29.2
LEO [26]	29.4	55.8	9.4	14.3	28.6	44.0	51.3	21.3	20.0	19.3	25.2	12.5	50.4	38.4	36.6	15.1	19.7	25.1	36.8	39.3	29.6
MKL [21]	**37.6**	47.8	**15.3**	15.3	21.9	50.7	50.6	30.0	17.3	33.0	22.5	**21.5**	51.2	45.5	23.3	12.4	23.9	28.5	**45.3**	**48.5**	32.1
UoCTTI [6]	29.0	54.6	0.6	13.4	26.2	39.4	46.4	16.1	16.3	16.5	24.5	5.0	43.6	37.8	35.0	8.8	17.3	21.6	34.0	39.0	26.8

Table 2. Results on PASCAL VOC 2007.

[3] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, In *CVPR*, pages 886-893, 2005. 3

[4] C. Desai, D. Ramanan, and C. C. Fowlkes, Discriminative models for multi-class object layout, *Int'l J. of Computer Vision*, 2011. 1, 2, 3, 4

[5] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, The pascal visual object classes (voc) challenge, *Int'l J. of Computer Vision*, 88:303-338, 2010. 6

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, Object detection with discriminatively trained part based models, *IEEE TPAMI*, 2010. 1, 2, 3, 4, 7, 8

[7] S. Fidler, M. Boben, and A. Leonardis, A coarse-to-fine taxonomy of constellations for fast multi-class object detection, In *ECCV*, 2010. 2

[8] S. Fidler, M. Boben, and A. Leonardis, Evaluating multi-class learning strategies in a generative hierarchical framework for object detection, In *NIPS*, 2009. 1, 2

[9] G. Griffin and P. Perona, Learning and using taxonomies for fast visual categorization, In *CVPR*, 2008. 2

[10] C. Gu, P. Arbelaez, Y. Lin, K. Yu, and J. Malik, Multi-component models for object detection, In *ECCV*, 2012. 1, 7, 8

[11] C.-N. J. Yu and T. Joachims, Learning structural svms with latent variables, In *ICML*, 2009. 5

[12] L. Lin, X. Wang, W. Yang, and J.H. Lai, Learning contour-fragment-based shape model with And-Or tree representation, In *CVPR*, 2012. 2

[13] A. Opelt, A. Pinz, and A. Zisserman, Incremental learning of object detectors using a visual shape alphabet, In *CVPR*, 2006. 2

[14] N. Razavi, J. Gall, and L. V. Gool, Scalable multi-class object detection, In *CVPR*, 2011. 1, 2, 7, 8

[15] R. Salakhutdinov, A. Torralba, and J. Tenenbaum, Learning to share visual appearance for multiclass object detection, In *CVPR*, 2011. 2

[16] P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele, Discriminative structure learning of hierarchical representations for object detection, *CVPR*, 2009. 1, 2

[17] J. Sivic, B. C. Russell, A. Zisserman, W. T. Freeman, and A. A. Efros, Unsupervised discovery of visual object class hierarchies, In *CVPR* 2008. 2

[18] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, Contextualizing object detection and classification, In *CVPR*, 2011. 1, 2, 7

[19] A. Torralba, K. P. Murphy, and W. T. Freeman, Sharing visual features for multiclass and multiview object detection, *IEEE TPAMI*, 29(5):854-869, 2007. 2

[20] D. Tran and D. Forsyth, Improved human parsing with a full relational model, In *ECCV*, 2010. 6

[21] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, Multiple kernels for object detection, In *ICCV*, 2009. 7, 8

[22] Y. Wang, D. Tran, and Z. Liao, Learning hierarchical poselets for human parsing, In *CVPR*, 2011. 1, 2, 7

[23] A. Yuille and A. Rangarajan, The concave-convex procedure(cccp), In *NIPS*, pages 1033-1040, 2001. 2, 5

[24] Y.B. Zhao and S.C. Zhu, Image parsing via stochastic scene grammar, In *NIPS*, 2011. 2

[25] S.C. Zhu and D. Mumford, A stochastic grammar of images, *Foundations and Trends in Computer Graphics and Vision*, 2(4): 259-362, 2006. 2

[26] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, Latent hierarchical structural learning for object detection, In *CVPR*, 2010. 1, 2, 5, 7, 8

[27] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. Yuille, max margin AND/OR graph learning for parsing the human body, In *CVPR*, 2008. 1, 2