Unsupervised Learning of Stochastic AND-OR Templates for Object Modeling

Zhangzhang Si and Song-Chun Zhu
Deptartment of Statistics, UCLA
{zzsi,sczhu}@stat.ucla.edu

Abstract

This paper presents a framework for unsupervised learning of a hierarchical generative image model called AND-OR Template (AOT) for visual objects. The AOT includes: (1) hierarchical composition as "AND" nodes, (2) deformation of parts as continuous "OR" nodes, and (3) multiple ways of composition as discrete "OR" nodes. These AND/OR nodes form the hierarchical visual dictionary. We show that both the structure and parameters of the AOT model can be learned in an unsupervised way from example images using an information projection principle. The learning algorithm consists two steps: i) a recursive Block-Pursuit *procedure to learn the hierarchical dictionary of primitives, parts and objects, which form leaf nodes, AND nodes and structural OR nodes and ii) a* Graph-Compression *operation to minimize model structure for better generalizability, which produce additional OR nodes across the compositional hierarchy. We investigate the conditions under which the learning algorithm can identify, (i.e. recover) an underlying AOT that generates the data, and evaluate the performance of our learning algorithm through both artificial and real examples.*

1. Introduction

Deformable templates ([3, 5, 14, 4]) and compositional hierarchy ([8, 9, 6, 15, 13, 11]) are widely used in visual object modeling to account for structural variations and shared parts among categories. Furthermore, generative image grammar [16, 2, 7, 1] is introduced to computer vision to facilitate robust statistical modeling of images using AND nodes for hierarchical composition, and OR nodes for deformation and alternate ways of composition.

The main issues of learning the image AND-OR template include the following: i) identifying a hierarchical dictionary of visual parts and objects; ii) deep mixing of AND, OR nodes; iii) Parts accuracy and shape; iv) Detailed coasse-to-fine detection of objects. Among them a most important issue with learning visual dictionaries is the pervasive ambiguity in identifying which elements should be grouped as a visual part. For example, it is hard to determine where to segregate the animal face or horse body into constituent parts. This has been mentioned by previous work such as [15].

The learning of hierarchical visual dictionaries has been explored in a series of recent work, where a hierarchy of meaningful visual parts are learned (or mined) from various image features, such as image primitives [6], segmented image regions [13], interest points [12, 15] and histogram of gradients [4]. Different learning algorithms have been used: discriminative criterion is followed in [4]; data mining heuristics are adopted in [6, 13]; maximum likelihood learning is used in [12] where a hierarchical latent Dirichlet process is assumed to generate interest point descriptors extracted from the image, and a Grammar-Markov model is used in [15]. The above methods have demonstrated the usefulness of the learned structures mainly through good classification performance.

To analyze the merits and limitations of the above methods, we characterize them by five aspects: i) compositional hierarchy, ii) unsupervised learning, iii) deep mixing of AND/OR nodes, iv) fully generative, v) probabilistic model. In [6, 13, 12, 15], a compositional hierarchy is learned by unsupervised learning. However, the OR nodes important for structural variations are largely omitted or oversimplified. And there is no deep mixing of AND/OR nodes, *i.e.* OR nodes across all levels of compositional hierarchy. Another major limitation of the above methods is that they are not fully generative to the level of image pixels. For example, in [12] a local Bag-of-words model summarizes local geometry and appearance as a pooled histogram, but detailed object deformation is discarded during computation of histograms. As a result, it is difficult to visualize the learned structures to ensure they are semantically meaningful. The state-of-art for object detection is achieved by [4], where a part-based latent SVMs model is trained using multi-scale HoG feature. It is able to model certain amount of object articulation, but the localization of object boundaries is imprecise because of the local histogram pooling in HoG. Another drawback of the model is that it requires tens and thousands of features and negative training examples.

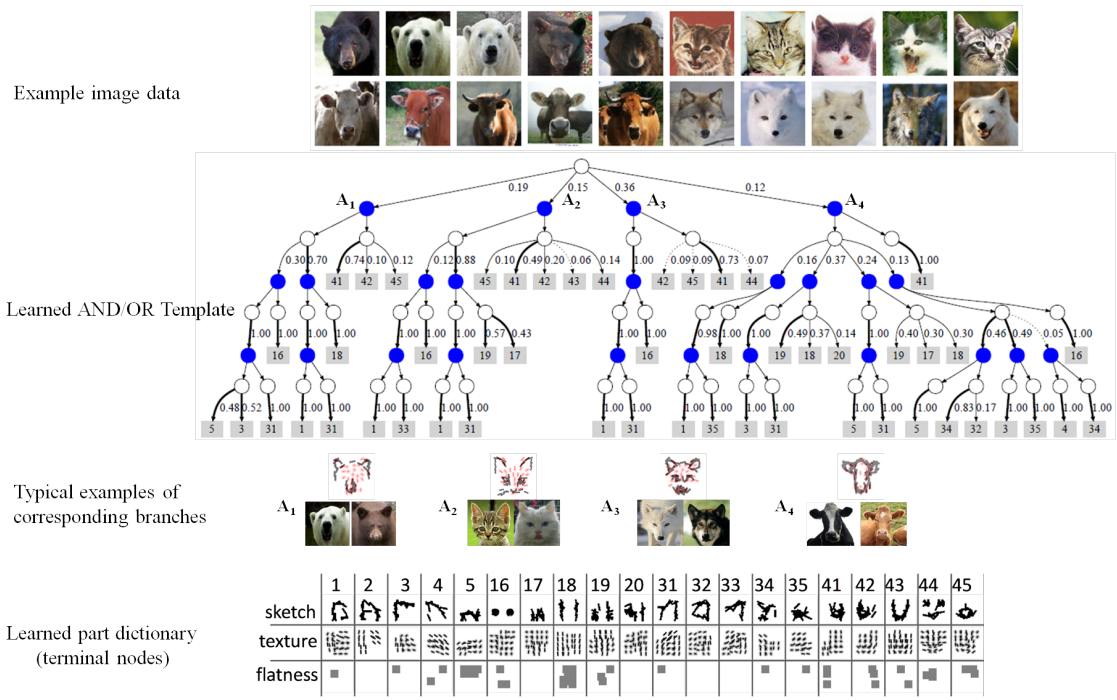


Figure 1. An AND-OR Template learned from 320 animal face images of four categories, with no manual labeling. Shaded circles denote AND nodes, which are combinations of terminal nodes. Empty circles means discrete OR nodes. Shaded rectangles are terminal nodes (a special case of AND nodes). Each terminal node is associated with a continuous OR node which account for its local geometric perturbation. For clarity we removed the continuous OR nodes and OR branches with probability less than 0.05.

We propose to learn hierarchical structures based on the generative model of active basis [14], where a small number of Gabor basis elements are generated by a shape template before they are linearly combined to generate the observed image. Using the active basis model, one can easily perform model checking for learned structures, without going through tedious classification experiments. In [14] hierarchical templates are not automatically learned.

In order to fulfill the full promise of the image AND-OR grammar models, we propose an unsupervised learning framework for learning the AND-OR Templates (AOT) of visual objects. As an example, Figure 1 shows a learned AOT from 320 animal face images without manual labeling. The solid circles denote AND nodes and hollow ones denote OR nodes. The branching probabilities are also shown on each OR node. The rectangles denote terminal nodes that index the entries of animal facial features (e.g. eyes, ears). These AND/OR nodes form a hierarchical dictionary of visual parts for fully generative image representation.

The learning algorithm carries out in two steps: i) Block-Pursuit for dictionaries of reusable parts. In a data matrix formed by collected features of positive training examples, a rectangular *block* in the data matrix is a set of common components (columns) shared by a set of examples (rows). The shared components form a *template* of the block members. The area of the block indicates its significance and pursuing large homogeneous blocks directly links to the information projection principle [17, 14]. The learned blocks then form a higher level of dictionary, which is used to produce another data matrix. This carries on recursively until a hierarchy of dictionary can be obtained.

ii) Graph-Compression on the AND-OR Template. Graph compression takes the learned dictionary as input and produces a more compact stochastic AND-OR Template by reducing the model complexity. In particular the number of free parameters (*i.e.* degree of freedom) is minimized. We apply a compression operator on the AND-OR Template, which can happen in two cases: (1) merging OR nodes with similar branching probabilities; (2) restructuring the graph by sharing parts (e.g. $(A \cap B) \cup (A \cap C) \Rightarrow A \cap (B \cup C)$).

The contributions of our paper are as follows: i) We learn AND and OR nodes, where OR nodes include continuous OR (deformation) and discrete OR (structural variation). ii) We adopt a principled learning framework under MLE and information projection. iii) We study the identifiability of parts and the factors that influence the identifiability.

2. 1D example: learning AOT from text

To study the identifiability issue, we study a 1D exmaple where we know the underlying AOT as ground truth that generates the training data. As shown in Fig.2, a stochastic AND-OR Template is used as the true generating model

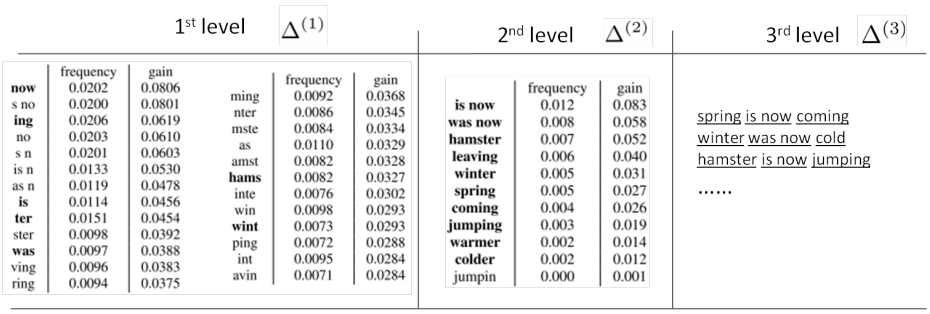


Figure 3. **Left:** The learned dictionary $\Delta^{(1)}$ for three/four letter groupings (white space is included). We only show the top ones, together with their frequencies and information gains side by side, up to a constant multiple. **Middle:** The learned dictionary $\Delta^{(2)}$ for words composed by entries in the children dictionary $\Delta^{(1)}$. **Right:** The learned dictionary for sentences as combinations of $\Delta^{(2)}$ entries.

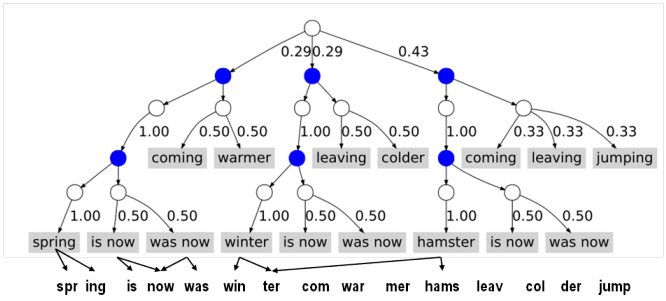


Figure 2. A stochastic AND-OR template for generating a sentence composed by three parts: subject + linking verb + adjective/present participle, such as "hamster is now jumping". Shaded circles denote AND nodes. Empty circles means OR nodes. Shaded rectangles are terminal nodes. Certain configurations are not allowed, such as *spring is now jumping*.

for sentences composed of three parts: subject + linking verb + adjective/present participle (*e.g.* winter is now leaving). The OR nodes introduce mixture components into the model, so the AOT in general is cannot be factorized. The three parts are correlated, so not all combinations of the three parts are admissible. For example, the combination *spring is now jumping* is not allowed. Each part is in turn composed of a prefix and a postfix. Each part, prefix/postfix and letter can be occluded by random letters with a small probability (0.01). Finally, random letters of varying lengths are inserted between the three parts of the sentence.

2.1. The data and data matrix

Table 1 shows several example strings generated by this underlying AOT. Our goal is to learn an AOT from the sampled example strings, and compare the learned AOT with the underlying one in Fig.2 to study the effectiveness of the learning algorithm in identifying its parts and composite structures. What Table 1 presents can be considered as a data matrix, where each row represents one example and each column represents one feature or component. Although different rows can be of various lengths, they can be treated as sharing the same length by padding nuisance symbols to shorter strings.

Table 1. String examples.

1. nkfnwkn**spring**yzxyxu**was now**jvzea**warmer**tgprh
2. oqsdq bov**hamster**iwxwo**was now**tdxtzbyc**coming**bjxp
3. lhtuwbcdzfz**hamster**aquo **is now**zgoclu**jumping**mmqrlu
4. jlmzzrsl**winter**vmqdle**is now**napla**leaving**douggkwh

2.2. Recursive block pursuit

By shuffling the data matrix, one can align the strings such that regular patterns coincide in their locations inside the string. We first identify frequent substrings of length l ($l = 3$ or 4), such as "ing", "ster", as significant blocks in the data matrix. These blocks are selected into the first level dictionary $\Delta^{(1)}$ (Fig.3).

Once $\Delta^{(1)}$ is learned, the strings are re-encoded using the entries in $\Delta^{(1)}$. We then construct a new data matrix by collecting co-occurrences of $\Delta^{(1)}$ entries. As a result, frequent combinations such as "spr"+"ing" are identified as significant blocks and selected into the second level *word* dictionary $\Delta^{(2)}$. An entry in the word level dictionary covers 6 to 8 letters. The word dictionary contain many duplicate or overlapping entries, such as "hamster" and "amster". The nuance entries like "amster" are pruned by a greedy procedure of finding best matching and local inhibition. In the end, only high frequency words remain in the top of $\Delta^{(2)}$ (Fig.3). Notice that compared to $\Delta^{(1)}$, $\Delta^{(2)}$ contains much less ambiguity. Finally the level 3 dictionary (sentences) $\Delta^{(3)} = \{$"spring is now coming", $\cdots\}$ is easily obtained by identifying frequent combinations of words.

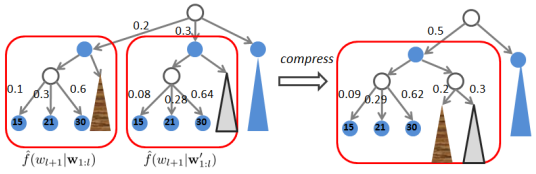
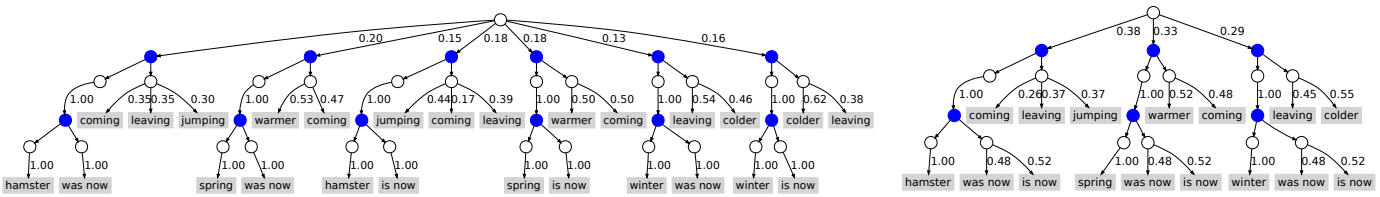


Figure 5. Compression on AOT by sharing probabilities.

2.3. Minimal structure via graph compression

To begin with, an AOT is naïvely constructed which has one giant OR node that branches over all entries in $\Delta^{(3)}$ (*i.e.*



(a) memorization AND-OR Template

(b) compressed AND-OR Template

Figure 4. **Left:** the memorization AoT. **Right:** the compressed AoT. Both are obtained from the same 100 training sequences sampled from the underlying AoT in Figure 2. The merging parameter α is 0.05. The memorization AoT has 13 free parameters. The compressed AoT has only 9 free parameters and successfully recovers the structure of true underlying AoT in Figure 2.

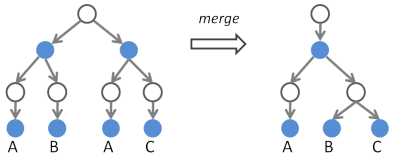


Figure 6. Compression on AOT by sharing parts.

all observed combinations of $\Delta^{(2)}$ entries). Then iteratively we apply a compression procedure illustrated in Fig.5.

If two OR nodes share the same parent OR node and similar branching distribution on children nodes, then they are merged into one, and the branching probabilities are re-estimated. In the mean time, the OR node is "pushed down" into the adjacent level. This is a lossy compression because the branching probabilities are modified and the number of free parameters are reduced. The compression operator is controlled by a parameter $\alpha \in [0, 1]$. $\alpha = 1$ corresponds to no compression at all, which results in the most complicated AOT that memorizes training data. In addition, as a pre-processing step we also compress model structure by sharing parts in a similar fashion to combining like terms (Fig.6). This is a lossless compression. With $\alpha = 0.05$, the compressed AOT in Fig.4 successfully recovers the true underlying AOT in Fig. 2.

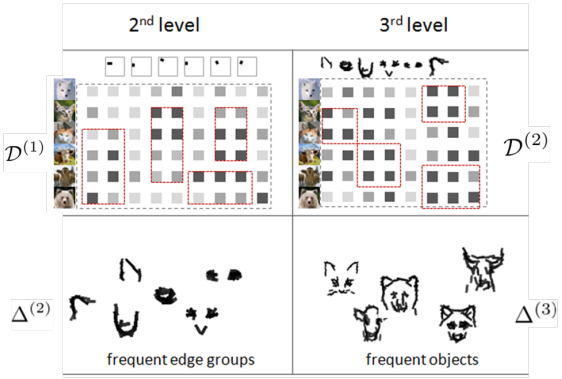


Figure 7. Block pursuit on images.

3. Learning AND-OR Templates on images

Given input images $\mathcal{X}_+ = \{\mathbf{I}_1, ..., \mathbf{I}_N\}$ as positive examples of multiple unlabelled categories, and another image set $\mathcal{X}_-$ as generic natural images, our objective is to pursue a series of probabilistic models

$$q(\mathbf{I}) = p_0(\mathbf{I}) \to p_1(\mathbf{I}) \to \cdots p(\mathbf{I}) \approx f(\mathbf{I})$$

to approximate the underlying distribution f, starting with q being a distribution governing $\mathcal{X}_-$.

We specify the image alphabet as a set of atomic feature prototypes $\Sigma_{\text{image}} = \{\text{primitives}\} \cup \{\text{textures}\} \cup \{\text{flatness}\}$. The image $\mathbf{I}$ is broken down into small image patches $\{\mathbf{I}_{\Lambda_s}\}$ where $\{\Lambda_s\}$ are local regions. For each feature prototype we measure a one-dimensional response $r_j(\mathbf{I}_{\Lambda_s})$, which indicates how likely the image patch belongs to the prototype. Here j indexes both feature prototype and image patch. The data matrix $\mathcal{D}^{(1)}$ at first level is formed with the features $\{r_j : j = 1, \cdots, D\}$ as columns and image examples $\{\mathbf{I}_n : n = 1, ..., N\}$ as rows.

Originally, the entries in the data matrix are explained by the marginal background distribution $q(r_j)$, $j = 1, ..., D$, that are pooled from a large number of generic natural images. The baseline log-likelihood for the data matrix is:

$$L_0 = \log q(\mathcal{X}_+) = \sum_{n=1}^{N} \sum_{j=1}^{D} \log q(r_j(\mathbf{I}_{n,\Lambda_j}))$$

By the information projection principle [10, 17], we have a step-wise procedure

$$\begin{aligned} p_k^* &= \arg\min \quad \mathcal{K}(p_j | p_{j-1}) \\ s.t. \quad & E_{p_j}[r_j(\mathbf{I}_{\Lambda_j})] = E_f[r_j(\mathbf{I}_{\Lambda_j})] \end{aligned}$$

where $\mathcal{K}$ denotes the Kullback-Leibler divergence. Intuitively, the updated model match more and more marginal statistics with the observed training examples. The model p after K iterations have the following log-linear form:

$$p(\mathbf{I}) = q(\mathbf{I}) \prod_{j=1}^{K} \left[\frac{1}{z_j} \exp\{\lambda_j r_j(\mathbf{I}_{\Lambda_j})\} \right] \tag{1}$$

where λ_j is the parameter for feature r_j and z_j is the individual normalization constant determined by λ_j. The resulting image log-likelihood is

$$L = \log p(\mathcal{X}_+) = L_0 + \sum_{n=1}^{N}\sum_{j=1}^{K}\left(\lambda_j r_j(\mathbf{I}_{n,\Lambda_j}) - \log z_j\right)$$

We divide the selected features $r_1, ..., r_K$ into a dictionary of blocks $\Delta = \{\mathcal{B}_1, ..., \mathcal{B}_M\}$ in the data matrix, which maximally explain the image data. The overall objective function of learning AOT can be formulated as a penalized maximum likelihood criterion:

$$\begin{aligned}\psi &= L - L_0 - \gamma \cdot \text{Complexity(AOT)} \\ &= \sum_{m=1}^{M} \sum_{\substack{n \in \text{rows}(\mathcal{B}_m) \\ j \in \text{cols}(\mathcal{B}_m)}} \left(\lambda_{m,j} r_j(\mathbf{I}_{n,\Lambda_j}) - \log z_{m,j}\right) \\ &\quad -\gamma \cdot \text{Complexity(AOT)} \end{aligned} \tag{2}$$

where rows(·) and cols(·) denote the selected rows and columns of block $\mathcal{B}_m$, $\{\lambda_{m,j}\}$ are parameters of image likelihood models for different blocks, and $\{z_{m,j}\}$ are individual normalizing constants determined by $\{\lambda_{m,j}\}$. The scalar γ controls the tradeoff between the likelihood and penalty terms.

The optimization procedure contains two steps: i) block pursuit that focuses on the first term of Eq.(2); ii) graph compression that focuses on the second term of Eq.(2).

3.1. Recursive block pursuit

At each step, we pursue one block (or a few blocks at the same time) that maximally increases ψ in Eq.(2), *i.e.* that has the largest information gain:

$$\mathcal{B}_m^* = \arg\max_{\mathcal{B}_m} \text{IG}(B_m)$$

where

$$\text{IG}(B_m) = \sum_{\substack{n \in \text{rows}(\mathcal{B}_m) \\ j \in \text{cols}(\mathcal{B}_m)}} \lambda_{m,j} r_j(\mathbf{I}_{n,\Lambda_j}) - \log z_{m,j}$$

With the rows of each block being hidden variables, the block pursuit is essentially an EM algorithm with variable selection. In particular, identifying the columns of a block $\mathcal{B}_m$ is equivalent to learning or updating the shared template for the block, carried out in the M (Maximization) step. It is a variable selection problem, which is addressed under the information projection principle [14]. One can also interpret the learned templates as words in a visual dictionary. In the E (Expectation) step, the rows of each block are identified. The assignment of rows to blocks can be hard assignments with nearest neighbor, or soft assignments with posterior probabilities. We tried two initialization methods of the EM algorithm: starting from a block that contains a single image patch, or random initialization. Both initialization methods work reasonably well.

The block pursuit is carried out recursively from atomic image features to parts, and from parts to objects (Fig. 7):

$$\Delta^{(1)} \rightarrow \mathcal{D}^{(1)} \rightarrow \Delta^{(2)} \rightarrow \mathcal{D}^{(2)} \rightarrow \Delta^{(3)}$$

where $\Delta^{(1)}, \Delta^{(2)}, \Delta^{(3)}$ are three levels of visual dictionaries and $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}$ are two levels of data matrices. The block pursuit procedure is terminated when no more blocks with significant information gain is found.

To account for deformation, we use continuous OR nodes as local geometric transforms of atomic image features in the same way as [14]. We assume the local geometric transforms are independent of each other, and they follow a uniform distribution over an affine neighborhood. To impute the hidden perturbations, we use their MAP (maximum a posteriori) estimations. Computationally, this is implemented by a simple local maximization on feature maps:

$$r_j^{\text{LMAX}} \triangleq \max_{j' \in \partial j} r_{j'}(\mathbf{I}),$$

where ∂j denotes the neighborhood around feature j. We also allow local translation and rotation of each block $\mathcal{B}_m$. And in the E step of block pursuit, we impute not only the rows of the blocks, but also the geometric transform of $\mathcal{B}_m$ on each image $\mathbf{I}_n$.

3.2. Graph compression

The penalty term $Complexity(AOT)$ is defined as the number of AND/OR nodes in AOT. Initially we form a giant AND-OR Template that has one root OR node with each unique co-appearance configuration as one branch. We first compress the AOT by sharing parts (a lossless compression). Then we apply the compression procedure illustrated in Fig. 5. For each pair of OR nodes, if they share the same parent OR node and similar branching distribution on children nodes, then they are merged, and the branching probabilities are re-estimated. This results in the change in log-likelihood $\delta L < 0$ and the change in model complexity (number of nodes) $\delta C < 0$. We decide to merge two OR branches if

$$\delta L - \gamma \cdot \delta C > 0 \tag{3}$$

i.e. the reduction in complexity outweighs the loss of log-likelihood.

Re-parameterization of the γ factor. Directly optimizing requires fine tuning of γ parameter, and the optimal value of $\gamma \in [0, +\infty)$ is very sensitive to the training data. We provide a robust re-parameterization of γ using another parameter $\alpha \in [0, 1]$. Observing that Eq.3.2 is essentially testing whether two distributions are the same, we

propose to use the χ^2 test with significance level $1-\alpha$ (where $\alpha \in [0,1]$) to approximately implement the decision in Eq.(3.2). If the branching probabilities of the two OR nodes are : $(a_1, ..., a_M)$ and $(b_1, ..., b_M)$ with $\sum_i a_i = 1, \sum_i b_i = 1, a_i > 0, b_i > 0, \forall i$, then the χ^2 test statistic is computed as $\chi^2 = \sum_i (a_i - b_i)^2 / a_i^2$. We compare this value to $F_{M-1,1-\alpha}$ which can be looked up in the F-table, and if $\chi^2 < F_{M-1,1-\alpha}$ then we decide merge these two OR nodes. In the experiments, we use α as the control parameter for model complexity instead of γ.

4. Experiment

4.1. The synthesized 1D example

In this experiment, we are interested in studying the factors that influence model identifiability: i) n: training sample size; ii) s: which is used to control the average length of random letters inserted between two words in the underlying AOT. When s is small, words are hard to separate, which results in larger ambiguity. and iii) α: the parameter of graph compression which implies the model complexity. n and s are parameters of training data, and α is a parameter of the learning algorithm.

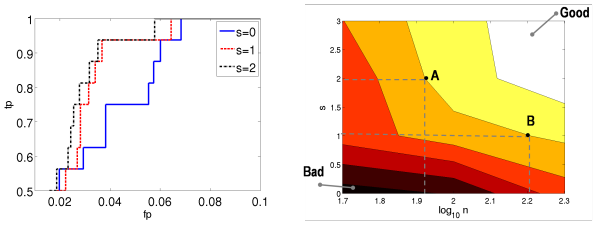


Figure 8. The effect of the separation parameter s and training sample size n on learned dictionary. **Left:** ROC curves for different separation parameters. **Right:** AUC as a function of separation s and sample size n.

Evaluating the learned dictionary of AOT. We compare the underlying "true" dictionary Δ_{true} to the learned dictionary Δ. We use the ROC curve and AUC (area under ROC curve) to evaluate the difference between manually labelled ground-truth Δ_{true} and the learned dictionary Δ. Fig. 8 (left) plots three ROC curves for three different values of s for sample size $n = 100$. After repeating this for different n, we obtain a series of ROC comparisons. To summarize this, Fig. 8 (right) shows the isolines of AUC as a function of two variables: s the separation parameter, and n the training sample size. Take the two points A, B as an example, when the separation decreases by 1 from A to B, we need about twice ($10^{0.3}$) as many training examples to achieve the same AUC.

Evaluating on the graph topology and branching probabilities of the learned AOT. Another important factor is the parameter α which controls model complexity. We set α to different values and compress the sentence-level dictionary $\Delta^{(3)}$ into a compact AND-OR Template, and compute the distance between the learned model AOT and the underlying model AOT* shown in Fig. 2. We use the Kullback-Leibler divergence as the distance between AND-OR Templates, which is estimated by Monte-Carlo method.

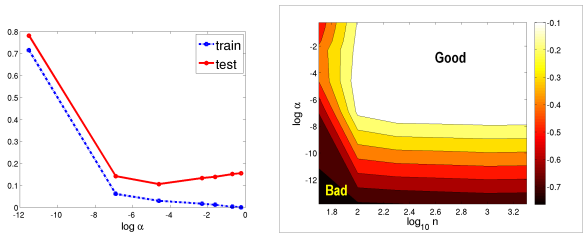


Figure 9. The effect of the model complexity α and training sample size n on model generalizability. **Left:** Error of learned model (KL divergence) as a function of model complexity α, plotted on training and testing data respectively. **Right:** KL divergence as a function of n and α.

To investigate the effect of parameter α and training sample size n on the model generalizability, we perform repeated cross validations. The result is shown in Fig. 9 (left). The horizontal axis is the logarithm of α which is sampled at seven points, and the vertical axis is the KL divergence between the learned model AOT and the true model AOT* from which training data (denoted as the empirical distribution $\hat{f}$) is sampled. Fig. 9 (right) shows at what sample size and what α values can we successfully recover the generating grammar. In the white convex region the grammar is recoverable (up to a tolerance of 0.1 bit).

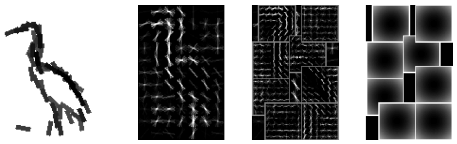
4.2. Object detection using AOT

Detection of AOT in a cluttered image is performed by a recursive SUM-MAX procedure[14]. We test the detection of AOT for three object categories: egret, deer and side-view bikes in PASCAL VOC 2007, in comparison to the state-of-art latent SVMs [4]. For all the three categories, we use a small training set of around 20 images and a larger and more challenging set of testing examples (Table 2). For the training images, the rough location and scale of object are known. For testing images, we are not given the scales and locations of the objects, and a lot of clutter is present.

Table 2. Training and testing sizes for the detection experiment.

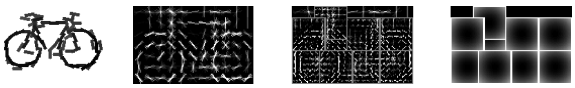
	egret	deer	bike
train	25	15	20
test	67	128	161

Fig. 10 and Fig. 11 are the learned object templates for the three categories using AOT and the part-based latent SVMs. In contrast to the discriminatively trained system [4] which uses a large number of features ($O(10^4)$), the AOT learned for each category only contains less than 100 features. It takes less than 10 minutes to learn an AOT with



(a) AOT (b) LSVM

Figure 10. Templates learned for egret.



(a) AOT (b) LSVM

Figure 11. Templates learned for VOC bike.

around 20 positive example images. While for latent SVMs, going over a large number of negative examples in each optimization iteration takes heavy computation. So despite the highly optimized implementation of the training algorithm, it takes much more time (several hours) to train a model.

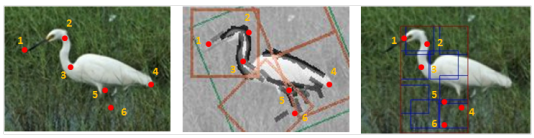


Figure 12. Evaluating object localization with key points.

To evaluate the detection accuracy, one may use the object-level bounding boxes provided by the PASCAL VOC benchmark. But it tells little about detailed correspondence between the template and the image. We propose to evaluate the localization of not only objects, but also parts and pixel-level key points. This is done using manually selected key points easily identified by a human labeler. Fig. 12 shows the ground truth key point labelings (6 key points), along with detected key points by AOT and by latent SVMs. The key point labels are used only in evaluating the detection results, and not in training of either models. For AOT, we associate each key point to the nearest edge element in the template, and record the most likely location of the key point relative to that edge element. For the latent SVMs, we associate each key point to the nearest rectangular part and record the most likely location of the key point relative to that part. We then propagate this information through the matched template to locate key points in the testing image.

To numerically measure the performance of localization, we use an imprecision-recall curve. In this curve, the horizontal axis is the tolerance for normalized displacement (or localization error) $\sqrt{(\Delta x)^2 + (\Delta y)^2}$ by dividing the object size. We restrict the range to be $[0, 1]$ for convenience. The vertical axis is the recall rate (between 0 and 1), *i.e.* the percentage of correctly detected points that fall within the specified displacement tolerance. As we tolerate more displacement, the recall rate increases. We use the area under curve (AUC) to measure the average recall rate.

Fig. 13 shows the imprecision-recall curves for 6 key points (tip of beak, joint of head and neck, joint of neck and body, tail, top of standing leg, and bottom of standing leg) of egret. We also show the curves for 4 parts (head, neck, body, and leg) and the whole object. To get the curves for parts and object, the displacement of the part is computed by averaging the displacements of key points associated with that part; and the displacement of object is computed by averaging the displacements of all key points. Our model performs localization consistently better than the state-of-art latent SVMs, for all the parts, key points, and all the displacement tolerances. Table 3 provides a numerical comparison for egret, deer and VOC bikes, using the area under curve (AUC) measure. The part and key point AUCs are computed by averaging over curves of all parts and key points.

Table 3. AUCs for localization of object, parts and key points.

	object		part		keypoint	
	AOT	LSVM	AOT	LSVM	AOT	LSVM
egret	.93	.80	.88	.76	.88	.73
deer	.93	.83	.91	.79	.90	.75
bike	.78	.76	.70	.66	.68	.61

In Fig. 14 and 15 we show the detection results on some testing images of egret and side-view bikes in VOC2007. From these examples we can see that the AOT can locate the object boundary and inner structures with a higher precision, which leads to more accurate localization overall.

4.3. Unsupervised learning of AOT

Fig. 1 shows the learned AND-OR Template from 320 animal face images of four categories: bear, cat, wolf and cow, without any manual labeling. At the bottom we show the top blocks ranked by their frequencies. For this experiment, we also incorporated the orientation histogram and flatness features both defined on the Gabor response map.

5. Conclusion

Under the information projection principle, the identifiable parts together with deformation and structural variation, make a strong image model that can be reliably learned from examples. The proposed AOT model improves upon state-of-art models in both generative ability and discriminative object detection tasks.

Acknowledgement. We thank Dr. Ying Nian Wu and the reviewers for the insightful suggestions. Our work is supported by NSF IIS-1018751, NSF CNS 1028381 and ONR MURI N00014-10-1-0933.

References

[1] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. 1

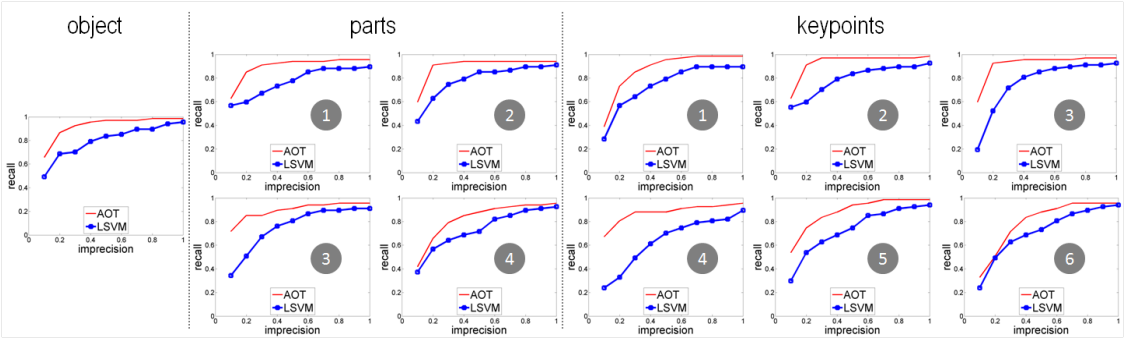


Figure 13. Egret: the quantitative performance measure for localization of object, parts and key points.

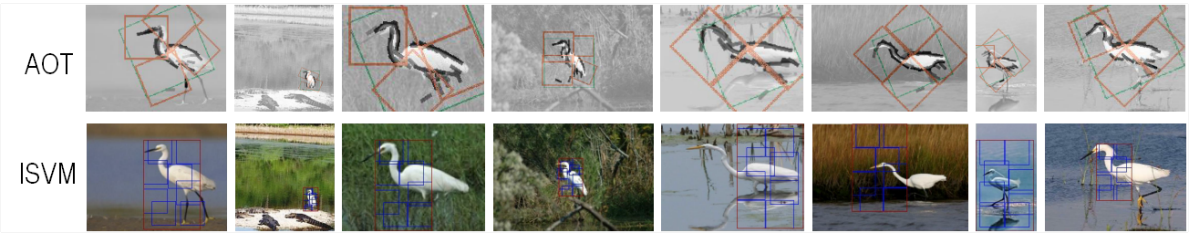


Figure 14. Detection results on Egrets (only showing the top detection per image). Best viewed in color.

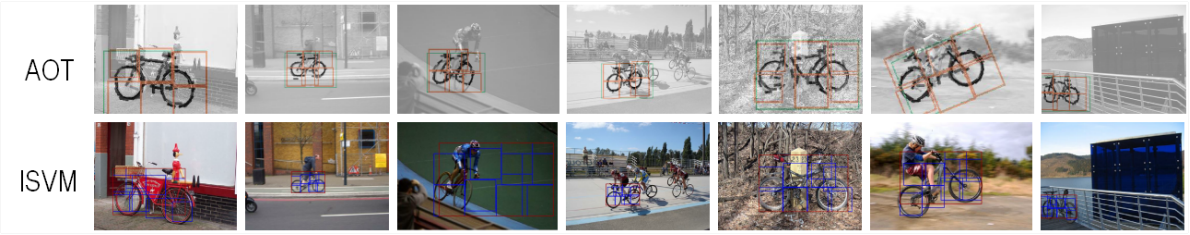


Figure 15. Detection results on side-view bikes (only showing the top detection per image). Best viewed in color.

[2] H. Chen, Z. Xu, Z. Liu, and S.-C. Zhu. Composite templates for cloth modeling and sketching. In *CVPR*, 2006. 1

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *TPAMI*, 23(6):681–685, 2001. 1

[4] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010. 1, 6

[5] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *IJCV*, 71(3):273–303, March 2007. 1

[6] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*, 2007. 1

[7] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009. 1

[8] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006. 1

[9] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, 2006. 1

[10] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *TPAMI*, 1997. 4

[11] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Object recognition with cortex-like mechanisms. *TPAMI*, 29:411–426, 2007. 1

[12] E. B. Sudderth, A. B. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed objects and parts. *IJCV*, 77(1-3):291–330, 2008. 1

[13] S. Todorovic and N. Ahuja. Unsupervised category modeling, recognition, and segmentation in images. *TPAMI*, 30(12):2158–2174, 2008. 1

[14] Y. N. Wu, Z. Si, H. Gong, and S.-C. Zhu. Learning active basis model for object detection and recognition. *IJCV*, 2009. 1, 2, 5, 6

[15] L. Zhu, Y. Chen, and A. Yuille. Unsupervised learning of probabilistic grammar-markov models for object categories. *TPAMI*, 2009. 1

[16] S. C. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006. 1

[17] S.-C. Zhu, Y. N. Wu, and D. B. Mumford. Minimax entropy principle and its applications to texture modeling. *Neural Computation*, 9(8)(2):1627–1660, 1997. 2, 4