Joint learning of visual attributes, object classes and visual saliency

Gang Wang
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
gwang6@uiuc.edu

David Forsyth
Dept. of Computer Science
University of Illinois at Urbana-Champaign
daf@uiuc.edu

Abstract

We present a method to learn visual attributes (e.g. "red", "metal", "spotted") and object classes (e.g. "car", "dress", "umbrella") together. We assume images are labeled with category, but not location, of an instance. We estimate models with an iterative procedure: the current model is used to produce a saliency score, which, together with a homogeneity cue, identifies likely locations for the object (resp. attribute); then those locations are used to produce better models with multiple instance learning. Crucially, the object and attribute models must agree on the potential locations of an object. This means that the more accurate of the two models can guide the improvement of the less accurate model.

Our method is evaluated on two data sets of images of real scenes, one in which the attribute is color and the other in which it is material. We show that our joint learning produces improved detectors. We demonstrate generalization by detecting attribute-object pairs which do not appear in our training data. The iteration gives significant improvement in performance.

1. Introduction

Object class learning is now an extremely active area of vision research (important recent papers include [8, 7]). While current methods for identifying instances and objects drawn from particular categories are quite strong, little is known about how categories should be organized internally or with respect to one another there is no satisfactory notion of an attribute. Object attributes reveal themselves as modifiers in language ("blue", "spotted", and so on). Modifiers refer to properties that many categories of object can share, which suggests that within class variation of objects is structured and shared between categories. It should be possible to exploit this structure because many attributes, particularly those dealing with material or appearance or color, can be observed in images. Two recent papers learn visual attributes from weakly supervised images [9, 18], but neither links attribute to object class. The most relevant work is [6] and [13], which learn to describe objects by their visual attributes. In this paper, we present an approach to jointly learn detectors for object classes and for visual attributes.

We demonstrate two important properties by combining object classes and visual attributes. First, the combination appears to result in improved object and visual attribute detectors. This is because our training data demands strong responses from an attribute detector and an object detector at the same locations and scales. As is usual, we do not know where these combined responses should occur. However, the attribute detector and the object detector must agree on where the object is. For example, as Figure 2, if we have "red car", "yellow car", "yellow dress" and "red dress", the "yellow" detector needs to agree with both the "car" detector in some images and the "dress" detector in others; the "red" detector must agree with the "dress" detector in some images and the "car" detector in others. Even with a moderate pool of attributes and objects, these constraints become very strong indeed. Joint learning (Sec.2) results in much improved detectors, because the object (resp. attribute) detector is revealing some information about where the attribute (resp. object) detector should respond. Second, we can generalize: we show that learning to detect attributes and categories simultaneously allows us to find quite specific objects of a form for which we possess no training examples. For example, we could build a "green car" detector by combining "green" and "car" detectors despite having no "green car" training images. Our joint learning method is related to the well known co-training algorithm [2] by using two different "modalities". But co-training requires fully labeled data and cannot use the key cue that the attribute detector and the object detector much agree on the location and scale of the foreground region.

In order to learn the visual attribute and object class detectors from the weakly labeled images, we need to sample local regions from images. For simplicity, we assume the
regions are rectangular. We call them “windows” in this paper. There are a lot of candidate windows with different aspects and sizes and we cannot use all of them. We sample the “interesting” ones instead, using models of visual saliency and homogeneity. Visual saliency is usually defined as a bottom-up process [12, 19]. In this paper, we adopt the definition of [17]: saliency is the set of attributes that distinguish a concept the most from others. It is task dependent. In our task, the visual salient windows should represent the object classes or visual attributes better than other windows.

Pixels from the same object or visual attribute tend to have similar low level properties such as brightness and texture. This means a window with evidence for many boundaries within it is unlikely to surround an object or visual attribute. Windows with smaller average $p_b$ scores [16] should be more homogeneous and more likely to be objects, and so we prefer them.

We propose a new joint multiple instance learning algorithm to jointly learn the object classes and visual attributes on the sampled windows. Multiple instance learning is a variation of supervised learning, where we must learn from “bags” of examples; we know that bags are either wholly negative or contain at least one positive example, but we don’t know which examples are positive [1]. Our method is novel, because we can force two learners (for object class and for attribute) to cooperate on labelling instances a window that contains the object must also contain the attribute. Unlike Ferrari and Zisserman [9], we do not localize the attribute precisely, nor do we explicitly encode variation in appearance, but instead link the attribute, and so its location, to that of the object; the advantage of doing so is that attribute and object models can help train one another. Our model is discriminative. In comparison to the work [18], we consider texture and material attributes as well as color attributes. Our work is also related to [10], who show that joint learning of object classes (“noun”) and object relationships (“Prepositions”) results in improvement. In this paper, we jointly learn object classes (“noun”) and visual attribute ("adjective").

2. Approach

We have a set of weakly labeled training examples, where we know that some “visual attribute object” pair (such as “red car”) is present, but do not know where or at what scale. We do not necessarily assume that every pair is represented in the data set (Figure 2). We now aim to build a detector for each visual attribute and a detector for each object class.

Building an object class (resp. attribute) detector is a multiple instance learning task. Each image is a “bag” of windows at different scales and locations. If the image is labeled positive, then at least one window contains the object. If it is labeled negative, no window does. In principle, we could use this information to train a classifier, but doing so is difficult, because we do not know which window contains the object. We will show how using attributes and class together simplify the problem. We will use the mi-SVM framework [1], which we introduce here briefly. Assume we wish to learn a linear 1-vs-all classifier for the $m$th object class. Write the classifier coefficients as $w^m, b^m$. For the $i$th example image, we have a set of windows $r_{ij}$, each of which is associated with an unknown label $y_{ij}$ and represented as feature $f_{ij}$. If the example is positive, we write $Y_i = 1$, and this means that some window must contain the example and so some $y_{ij}$ is one. If it is negative, we write $Y_i = -1$, and all windows must be negative. Using this notation, an mi-SVM must solve the following optimization

![Figure 1. Training and test categories of the color object data set. Each row shows a color and each column shows an object class. Each color model is learned with all the images in the row and each object model is learned with all the images in the column. Color detectors and object detectors need to agree on the foreground location in intersection images. Two test categories shown in red square, “red cap” and “yellow flower”, are not trained in the training procedure. But we can obtain their models by combining the color models and object models learned from other categories.](image-url)
problem (the explanation of notations could be found in Table 1).

\[
\begin{align*}
\min_{y_{ij}^m} & \min_{w^m, b^m} \frac{1}{2} \|w^m\|^2 + C \sum_{ij} \xi_{ij}^m \\
\text{s.t.} \forall i,j : & y_{ij}^m (\langle w^m, f_{ij} \rangle + b^m) \geq 1 - \xi_{ij}^m \\
& \xi_{ij}^m \geq 0 \\
y_{ij}^m &= -1 \text{ if } Y_i^m = -1 \\
\max_j y_{ij}^m &= 1 \text{ if } Y_i^m = 1
\end{align*}
\]  

the problem is easy if the positive windows are known. Otherwise it is difficult, because of the max term, which makes this a mixed integer problem. In effect, the optimizer must determine which window in each bag is the positive example, which involves a search of a very large space if there are many windows per bag.

We have two methods to significantly improve the behaviour of the multiple instance learning process. First, we will reduce the search space by showing the multiple instance learner only a small set of interesting windows for each image (section 2.1). Second, we will exploit the important constraint that the attribute and the object need to be present in the same window (section 2.2).

2.1. Focusing the Search for Good Windows

We expect a good candidate window to be salient, because the object should “stick out”, in a way that depends on our model of classes and of attributes. We expect a good candidate window to be homogeneous, because objects tend to have coherent appearance. The presence of these cues means we do not need to look at every window in an image to determine which is the positive window. This yields an important simplification: we can use saliency and homogeneity to sample image windows, and ensure we see the most useful for building a classifier. This requires an iteration. In our approach (Figure 2), we iterate estimating a classifier for attribute (resp. class) with using saliency and homogeneity cues to sample candidate windows for training. The form of the saliency cue is changed by the classifier estimate, which is why the procedure works.

We must identify windows in bags that have a good chance of containing the object class or the attribute. Such windows are important both for positive bags (because they allow us to estimate which window is the positive one) and for negative bags (where they are the important negative examples). We use visual saliency and homogeneity to subsample the set of windows in the bag to emphasize interesting image windows. Saliency captures top-down cues from the current models of object (resp. attribute), while homogeneity captures the important bottom-up cue that objects tend to be coherent in appearance. Once we have saliency and homogeneity scores, we sample the windows in each image to produce a small bag of interesting windows. We have T candidate windows of arbitrary size and aspect ratio for each image. For the tth window, write \(o_t\) for the average object saliency value, \(v_t\) for the average visual attribute saliency value, and \(h_t\) for the homogeneity score. We now sample the tth window from the image with probability

\[
\sum_t o_t v_t h_t
\]

so that windows that are strongly salient and strongly homogeneous have a good chance of appearing in the next round of training.

Computing Saliency: For us, saliency is the attribute that distinguishes a concept most clearly from others [17]. This definition means that saliency depends on the set of objects to be recognized. Salient regions are those most likely to contain either the object or the visual attribute sought, conditioned on the current model. We use the method of [17] to learn saliency. Given a known set of object (resp. attribute) detectors, we sample a set of image patches on a regular grid, and classify them with the detectors. Any pixel inside a patch then gets the patches detector response, and we average where patches overlap. The result is two saliency maps for each image: one produced by the attribute detector and one produced by the class detector. Now, for each image window, we calculate the average saliency value, and use this map to sample windows.

Computing Homogeneity: Pixels from the same object or visual attribute tend to have similar low level properties
such as brightness and texture. This means a window with evidence for many strong boundaries within it is unlikely to surround an object or visual attribute. We represent this homogeneity property with the $p_b$ feature [16], which estimates likely a pixel is to be a boundary pixel. We prefer windows with a small average $p_b$, because this implies fewer significant boundaries within the window. For each window, we calculate $h = e^{-\gamma B}$ value as the measure for the homogeneity. $B$ denotes the average $p_b$ value of this window; $\gamma$ is a constant. Unlike visual saliency, homogeneity is not updated in the learning procedure.

2.2. Joint Multiple Instance Learning

Our multiple instance learning problem differs from the usual in an important way. Each image is a “bag” of windows. Inside each positive bag is at least one window that is a positive example for the object class. The same window is also a positive example for the attribute. This means that if the object class learner has found a likely positive example, it can help the attribute learner, and vice versa. To describe this process in more detail needs some notation, which appears in Table 1.

<table>
<thead>
<tr>
<th>$I_i$: the $i$th image</th>
<th>$r_{ij}$: the $j$th window in the $i$th image</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{ij}^o$: the object class feature of $r_{ij}$</td>
<td>$f_{ij}^v$: the visual attribute feature of $r_{ij}$</td>
</tr>
<tr>
<td>$Y_{ij}^m$: \begin{cases} 1 &amp; \text{if the $m$th object class is present in $I_i$} \ 0 &amp; \text{otherwise} \end{cases}</td>
<td>$L_i^n$: \begin{cases} 1 &amp; \text{if the $n$th visual attribute is present in $I_i$} \ 0 &amp; \text{otherwise} \end{cases}</td>
</tr>
<tr>
<td>$y_{ij}^m$: \begin{cases} 1 &amp; \text{if the $m$th object class is present in $r_{ij}$} \ 0 &amp; \text{otherwise} \end{cases}</td>
<td>$l_{ij}^n$: \begin{cases} 1 &amp; \text{if the $n$th visual attribute is present in $r_{ij}$} \ 0 &amp; \text{otherwise} \end{cases}</td>
</tr>
<tr>
<td>$w^m, b^m, \xi^m$: the SVM for the $m$th object class</td>
<td>$v^n, d^n, \eta^n$: the SVM for the $n$th visual attribute</td>
</tr>
</tbody>
</table>

Table 1: The notation for the learning algorithm

Each window $r_{ij}$ in the $i$’th image $I_i$ is represented with features relevant to object class $f_{ij}^o$ and features relevant to attribute $f_{ij}^v$. Detailed descriptions of these features appear in Sec.3. We know whether the image contains the $m$’th object class ($Y_{ij}^m$), but not which window contains it ($y_{ij}^m$). Our problem differs from the standard multiple instance learning problem, because there is a crucial extra cue. If a particular window is positive for an object class, it is also positive for that image’s attribute. This means that if $Y_{ij}^m + L_{ij}^n = 2$, then $y_{ij}^m = l_{ij}$ for each $j$. We can then write the problem as minimizing

$$\sum_m \frac{1}{2} ||w^m||^2 + C \sum_{ij} \xi_{ij}^m + \sum_n \frac{1}{2} ||v^n||^2 + C \sum_{ij} \eta_{ij}^n$$ \hspace{1cm} (7) subject to a set of constraints. Constraints are the usual mi-SVM constraints, and also the requirement that object class and attribute labels be consistent, yielding:

$$\forall i,j: y_{ij}^m (\langle w^m, f_{ij}^o \rangle + b^m) \geq 1 - \xi_{ij}^m$$ \hspace{1cm} (8)

$$\xi_{ij}^m \geq 0$$ \hspace{1cm} (9)

$$l_{ij}^n (\langle v^n, f_{ij}^v \rangle + d^n) \geq 1 - \eta_{ij}^n$$ \hspace{1cm} (10)

$$\eta_{ij}^n \geq 0$$ \hspace{1cm} (11)

$$y_{ij}^m = -1 \text{ if } Y_{ij}^m = -1$$ \hspace{1cm} (12)

$$\max_j y_{ij}^m = 1 \text{ if } Y_{ij}^m = 1$$ \hspace{1cm} (13)

$$l_{ij}^n = -1 \text{ if } L_{ij}^n = -1$$ \hspace{1cm} (14)

$$\max_j l_{ij}^n = 1 \text{ if } L_{ij}^n = 1$$ \hspace{1cm} (15)

$$y_{ij}^m = l_{ij}^n \text{ if } Y_{ij}^m + L_{ij}^n = 2$$ \hspace{1cm} (16)

This problem can be seen as two separate optimization problems, coupled by the instance labels. A natural solution is to iterate, predicting the instance labels with the current class (resp. attribute) classifiers, then retraining. In particular, when we compute the instance labels for a positive bag, we can require that the window that has the strongest sum of responses is the positive instance. This means that the class (resp. attribute) classifiers can guide one another in the search. The procedure is given as pseudocode in alg. 1.

Algorithm 1 The Pseudo-code for the optimization heuristic.

Initialize: $y_{ij}^m = Y_{ij}^m, l_{ij}^n = L_{ij}^n$ for each $i$ and $j$.

repeat

Compute SVM solution $\{w_m, b_m\}$ and $\{v_n, d_n\}$ for each object class $m$ and visual attribute $n$.

for every image $i$ do

if $Y_{ij}^m + L_{ij}^n = 2$ then

for each window $j$ do

Classification with the learned SVM detectors, outputting object and visual attribute labels: $y_{ij}^m$ and $l_{ij}^n$.

set $y_{ij}^m = \eta_{ij}^n = \sgn(y_{ij}^m + l_{ij}^n - 0.5)$

end for

if $\max_j y_{ij}^m < 0$ then

compute $j^* = \max_j (\langle w^m, f_{ij}^o \rangle + b^m + \langle v^n, f_{ij}^v \rangle + d^n)$

set $y_{ij}^m = l_{ij}^n = 1$

end if

end if

until $N$ times

540
3. Data sets and features

We have collected two data sets specifically for this problem, because existing data sets do not pair attributes and object properties. We collected the datasets by searching Google and Flickr with the “attributes + object” keywords, false positive images for each class are manually deleted. Each class has about 30 ~ 60 images. The first one focuses on color as a visual attribute. The data set includes sixteen visual attribute object pairs as shown in the test panel of Figure 2: “blue cap”, “blue car”, “blue pants”, “purple car”, “purple flower”, “red cap”, “red car”, “red dress”, “red flower”, “red pants”, “red umbrella”, “yellow cap”, “yellow car”, “yellow dress”, “yellow flower”, “yellow umbrella”. There are four colors (“blue”, “purple”, “red” and “yellow”) and six object classes (“cap”, “pants”, “car”, “dress”, “flower” and “umbrella”). The second data set is a material object data set and includes six visual attribute object pairs: “metal cabinet”, “metal cup”, “spotted dress”, “spotted leopard”, “striped dress”, “striped zebra” and “wooden cabinet”. There are four materials (“metal”, “spotted”, “striped”, “wooden”) and five object classes (“cabinet”, “cup”, “dress”, “leopard” and “zebra”).

Object class and visual attribute features are extracted from each window respectively. The spatial pyramid representation [14] is used to represent spatial structure in each window.

Color attribute data: For the color object data set, we use a color feature to represent visual attributes. We represent the windows using four scales, with the i-th level having $2^{i-1}$ cells. We write $(r, g, b)$ for the average RGB value for each window and represent each cell by a 6-dimensional vector:

$$[r, g, b, \frac{r-g}{r+g+b}, \frac{b-r}{r+g+b}, \frac{r+g+b}{255+3}] \quad (17)$$

Overall, each window is represented as a 180 ($(1+4+9+16)*6$) dimensional vector. For the color object data set, we use a coarse HOG [4] feature to represent object classes. We represent each window in three levels, and quantize the gradients into 9 bins in each cell. Overall, there are 126 dimensions.

Material attribute data: In the material data set, we use a texton feature [15] to represent visual attributes. We quantize the texture features to 200 clusters and represent each region as a 200-dimensional vector. We use the shape based gradient feature described in [3] as an object class feature. We use a pyramid with three levels and quantize the gradient at each cell to 30 bins, so the feature has 420 dimensions, and use $p_b$ [16] to find the edges.

Finally, we implement the SVM using SVMperf [11], with a linear kernel. The SVM parameter C is set to be 1000, which produces good performance in our experiments.

### Table 2

<table>
<thead>
<tr>
<th>Round</th>
<th>Cap</th>
<th>Car</th>
<th>Dress</th>
<th>Flower</th>
<th>Pants</th>
<th>Umbrella</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1</td>
<td>0.02/0.01</td>
<td>0.14/0.09</td>
<td>0.16/0.10</td>
<td><strong>0.13/0.08</strong></td>
<td>0.03/0.06</td>
<td>0.03/0.02</td>
</tr>
<tr>
<td>Round 2</td>
<td>0.08/0.02</td>
<td>0.14/0.09</td>
<td>0.18/0.06</td>
<td>0.11/0.10</td>
<td>0.05/0.05</td>
<td>0.05/0.03</td>
</tr>
<tr>
<td>Round 3</td>
<td><strong>0.11/0.02</strong></td>
<td><strong>0.17/0.11</strong></td>
<td><strong>0.18/0.06</strong></td>
<td><strong>0.12/0.13</strong></td>
<td><strong>0.06/0.05</strong></td>
<td><strong>0.06/0.03</strong></td>
</tr>
</tbody>
</table>

### Table 3

<table>
<thead>
<tr>
<th>Round</th>
<th>Blue</th>
<th>Purple</th>
<th>Red</th>
<th>Yellow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1</td>
<td>0.24/0.24</td>
<td>0.21/0.13</td>
<td>0.29/0.29</td>
<td>0.20/0.16</td>
</tr>
<tr>
<td>Round 2</td>
<td>0.28/0.29</td>
<td>0.23/0.15</td>
<td><strong>0.33/0.33</strong></td>
<td>0.22/0.17</td>
</tr>
<tr>
<td>Round 3</td>
<td><strong>0.30/0.29</strong></td>
<td><strong>0.27/0.18</strong></td>
<td>0.30/0.33</td>
<td><strong>0.25/0.19</strong></td>
</tr>
</tbody>
</table>

Table 2. The average precision (AP) score for each object class and color at each round. In each cell, the first number denotes the result with joint learning; the last number denotes the result with separate learning. Joint learning works better in most of the categories because it combines the object and visual attribute cues. The performance is improved by re-estimating the windows.

Table 3. The average precision (AP) score for detecting “visual attribute object” pair at the third round. This is achieved by adding the confidence values of visual attribute and object class detectors for each window to draw the precision-recall curve. In each cell, the first number denotes the result with joint learning; the last number denotes the result with separate learning. Two categories “Red cap” and “Yellow flower” don’t have training examples in the training procedure, we detect them by combining the visual attribute and object detectors learned from other categories. Empty cells contain no training and test images.
4. Experiments

In each data set, joint learning and separate learning methods are tested and compared on the same tasks. In joint learning, windows are sampled from images with both visual attribute and object class saliency maps, the detectors of visual attribute and object class are learned jointly in the joint multiple instance learning framework; in separate learning, the windows are sampled for visual attribute and object class separately, the detectors are learned separately with the standard multiple instance learning algorithm.

4.1. Color data set

We train the detectors with fourteen categories (as shown in the left panel of Figure 2). 25 training examples from each category are used. SVM detectors for four colors (“blue”, “purple”, “red” and “yellow”) as well as for six object classes (“cap”, “pants”, “dress”, “flower”, “car” and “umbrella”) are learned in separate learning and joint learning methods respectively. “red cup” and “yellow flower” are held out in the training procedure. We sample the windows for three times (We have implemented the fourth round, but find the performance doesn’t change much, so we show only three rounds in this paper). At each round, about 100 windows are sampled from each image. In the joint multiple instance learning heuristics, we iterate for 20 times at each round. Each round takes about 10 hours on the computer with 3.6G Hz CPU.

With the learned detectors, we do detection on the test images. This is the same as the detection task in the PASCAL challenge [5]. We use the standard sliding windows method to extract regions from the test images. The learned detectors are applied to classify the windows and output the confidence values. The detection procedure takes about 2 seconds for each test image. Then for each visual attribute or object class, the precision-recall curve can be drawn. As the PASCAL challenge, only the predicted boxes which overlap much with the ground truth boxes can be considered to be correct. The average precision (AP) value is used as the quantitative measure. The AP values for object classes and visual attributes are shown in Table 2. We can also do detection for the “visual attribute object” pair such as “red car”, the results are shown at Table 3. We also train object and color detectors in a fully supervised way. That means, we use the ground truth bounding boxes of the training images to find the foreground regions, and only use them as positive examples. But because we don’t have enough positive training examples, the resulting detectors don’t work well.

Another benefit of combining the visual attributes and object classes (though this is not brought by the joint learning) is that specific color objects which don’t have training examples such as “red cap” can also be detected by combining the color and object models learned from other categories. We test such generalization ability on two held out categories “red cap” and “yellow flower”. The AP values are shown in Table 3. The joint learning method still works better than separate learning in this task since it has better visual attributes and object classes detectors.

![Figure 3](image)

Figure 3. The saliency maps generated by the detectors produced in the second round. If the region is brighter, it is more likely to contain the object or the visual attribute. The first column shows the original images; the second column shows the saliency maps generated by the object classes detectors; the third column shows the saliency maps generated by the visual attribute detectors; the fourth column shows the saliency maps by combining the above two maps. By combining the two cues, we can get better estimate of the foreground regions.

4.2. Material dataset

In the material data set, we train on six categories: “metal cabinet”, “metal cup”, “spotted dress”, “spotted leopard”, “striped zebra” and “wooden cabinet”. The “striped dress” category is held out to test the generalization ability. After training, we get the detectors of “metal”, “spotted”, “striped”, “wooden” and “cup”, “leopard”, “dress”, “zebra”, “cabinet”.

<table>
<thead>
<tr>
<th>Image</th>
<th>Object class Saliency map</th>
<th>Visual attribute Saliency map</th>
<th>Combined Saliency map</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Image" /></td>
<td><img src="image" alt="Image" /></td>
<td><img src="image" alt="Image" /></td>
<td><img src="image" alt="Image" /></td>
</tr>
</tbody>
</table>
We perform detection experiments on this data set. As in the color object data set, we use the sliding windows for the detection, and the AP value is used to measure the performance. Since the feature extraction needs more time on this dataset, it takes about 15 seconds to process each test image. The detection performance of object classes and visual attributes is shown in Table 4. The joint learning outperforms separate learning in most categories. The performance is improved by re-sampling windows. The performance of detecting visual attribute object pair is shown in Table 5.

On the “striped dress” category, we combine “striped” and “dress” detectors learned from other categories to do detection.

The detection windows at each round are shown in Figure 5. We get better detection windows after re-sampling the windows.

5. Conclusion and future work

Our joint multiple instance learning process offers substantial increases in performance with few rounds of re-estimation. This is because the constraint that an attribute model and an object model should agree on how likely it is an object appears in a window offers powerful cues to the location of an instance in an image. Experiments suggest strongly that these improvements have to do with agreement, rather than with the increase in training data associated with pooling over attributes (resp. objects). This is most likely because agreement means the stronger model can guide the weaker to better possible locations. In this paper, only one attribute (modifier) is used in each image. But one object instance may have multiple types of visual attributes (For example, a can can be “red” and “metal”). A future direction is to exploit more attributes simultaneously.

Some visual attributes cannot be inferred only from objects appearance. Instead, multiple objects should be considered. For example, when we say some objects are “big”, we in fact compare them with other objects. More sophisticated relationship between objects and visual attributes should be considered in the future.

6. Acknowledgement

The authors would like to thank all the anonymous reviewers for their suggestions. This work was supported
Table 4. The average precision (AP) score for each object class and material at each round. In each cell, the first number denotes the result with joint learning; the last number denotes the result with separate learning. We see joint learning works better in most of the categories because it combines the object and visual attribute cues. The performance is improved by re-estimating the windows.

<table>
<thead>
<tr>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cabinet</td>
<td>Cup</td>
<td>Dress</td>
</tr>
<tr>
<td>0.20/0.13</td>
<td>0.03/0.03</td>
<td>0.10/0.07</td>
</tr>
<tr>
<td>Cup</td>
<td>0.04/0.02</td>
<td>0.05/0.05</td>
</tr>
<tr>
<td>Dressed</td>
<td>0.16/0.07</td>
<td>0.04/0.04</td>
</tr>
<tr>
<td>Spotted</td>
<td>0.04/0.09</td>
<td>0.09/0.05</td>
</tr>
<tr>
<td>Striped</td>
<td>0.30/0.27</td>
<td>0.07/0.04</td>
</tr>
<tr>
<td>Wooden</td>
<td>0.12/0.08</td>
<td>0.04/0.03</td>
</tr>
<tr>
<td>0.08/0.05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5. The average precision (AP) score for detecting “visual attribute object” pair at the third round. This is achieved by adding the confidence values of visual attribute and object class detectors for each window to draw the precision-recall curve. In each cell, the first number denotes the result with joint learning; the last number denotes the result with separate learning. The category “striped dress” doesn’t have training examples in the training procedure, we detect it by combining the visual attribute and object detectors learned from other categories. Empty cells contain no training and test images.

<table>
<thead>
<tr>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metal</td>
<td>Spotted</td>
<td>Striped</td>
</tr>
<tr>
<td>0.06/0.06</td>
<td>0.30/0.28</td>
<td>0.33/0.31</td>
</tr>
<tr>
<td>0.12/0.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spotted</td>
<td>0.07/0.06</td>
<td>0.28/0.29</td>
</tr>
<tr>
<td>0.35/0.33</td>
<td>0.12/0.09</td>
<td></td>
</tr>
<tr>
<td>Striped</td>
<td>0.07/0.06</td>
<td>0.29/0.32</td>
</tr>
<tr>
<td>0.06/0.06</td>
<td>0.33/0.32</td>
<td></td>
</tr>
<tr>
<td>Wooden</td>
<td>0.15/0.09</td>
<td></td>
</tr>
<tr>
<td>0.15/0.09</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

in part by the National Science Foundation under IIS - 0803603 and in part by the Office of Naval Research under N00014-01-1-0890 as part of the MURI program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation or the Office of Naval Research.

References


