A Joint Learning Framework for Attribute Models and Object Descriptions

Dhruv Mahajan Sundararajan Sellamanickam Vinod Nair
Yahoo! Labs, Bangalore, India
{dkm,ssrajan,vnair}@yahoo-inc.com

Abstract

We present a new approach to learning attribute-based descriptions of objects. Unlike earlier works, we do not assume that the descriptions are hand-labeled. Instead, our approach jointly *learns both the attribute classifiers and the descriptions from data. By incorporating class information into the attribute classifier learning, we get an attribute-level representation that generalizes well to both unseen examples of known classes and unseen classes. We consider two different settings, one with unlabeled images available for learning, and another without. The former corresponds to a novel transductive setting where the unlabeled images can come from* new *classes. Results from* Animals with Attributes *and* a-Yahoo, a-Pascal *benchmark datasets show that the learned representations give similar or even better accuracy than the hand-labeled descriptions.*

1. Introduction

Attribute-based object recognition (e.g. [6, 9]) aims to go beyond simply *naming* an object in an image to *describing* it by its high-level characteristics. Given an image of, say, a zebra, binary classifiers are applied to the image to predict the presence or absence of semantic attributes like *stripes*, *four legs*, *hooves*, etc. (along with various non-zebra attributes). Their outputs form a binary vector description of the object, which can then be used to predict the class label. The set of attributes used in a particular application is typically designed by hand to be broad enough to describe the objects of interest.

Since objects share attributes (e.g. many animals have *four legs*), images for only a subset of objects may be needed to train the attribute classifiers. What is learned about an attribute from one class can be used by other classes. This allows for a new kind of generalization: a previously *unseen* object class can be recognized at test time knowing only which attributes it has. Such a capability can be used to build a system that initially learns on a small set of labeled objects and then gradually grows the catalogue of objects it can recognize as it sees more images, with human supervision limited to simply naming the new objects.

Lampert et al. [9] and Farhadi et al. [6] train attribute classifiers on an attribute-labeled dataset consisting of pairs of images and corresponding attribute values. Both use hand-labeled attribute values to describe objects. For example, the Direct Attribute Prediction (DAP) model [9] (see Figure 2(a)) uses a single *attribute vector* per class as the description for the class. Given a test image, the model compares the attribute classifier outputs against the attribute vectors of the known classes to predict the object label. This has several drawbacks:

1. Semantic attributes need not always be detectable from images. This can happen because they are either non-visual (e.g. *smelly* and *smart* from *Animals with Attributes* dataset [9]), or the image representation does not contain sufficient information to detect them (e.g. detecting a colour attribute from a grayscale image).

2. The attributes may not be discriminative. The most intuitive visual attributes we pick to describe, say, cats and dogs, are unlikely to be good for distinguishing the two classes.

3. The attributes can be redundant. For example, Pubfig [8] defines the attributes *Senior* and *Youth* to describe faces, which are anti-correlated and therefore redundant.

4. Human labeling error can produce incorrect descriptions of objects. For example, the class *Collie* in the *Animals with Attributes* dataset has the attribute *black* set as absent, yet most images in that class actually contain prominent black patches. Setting attribute values by hand is more ambiguous and requires more careful judgments than assigning a class label, so attribute values are more prone to error.

We aim to overcome these limitations by *automatically learning from data* the attribute vectors for the object classes (both known and unseen), instead of hand-specifying them. The idea is to rely on human supervision to define a useful 'vocabulary' for describing objects, but the descriptions themselves should be learned from data. To do this, we *jointly* learn the attribute classifiers and the attribute vectors. Thus we allow the potentially more reliable class label information to influence the learning of the attribute classifiers. The joint optimization helps in getting a better attribute-level representation, by converting unde-

tectable and redundant attributes into discriminative ones, while retaining the useful semantic attributes.

The basic version of our algorithm takes as input 1) a set of attributes, 2) positive and negative training examples for each of the binary attribute classifiers, and 3) training images of various objects with *known* class labels (but not their attribute labels). Note that labeled datasets with known classes either already exist or are relatively easier to collect (e.g., by using results for an image search on an object name like *Zebra*). After learning, the algorithm outputs one attribute vector per class. We then extend the basic version so that, given unlabeled images from a pre-specified number of unseen classes[1], it outputs an attribute-based description for each of those classes as well. Results show that the solution generalizes well both 1) on the unseen examples of the known classes and 2) on unseen classes.

Automatically learning the attribute vectors also removes the manual effort needed to specify them. This is particularly beneficial when we want to build a system that can recognize thousands of classes.

1.1. Contributions

1. We present a general framework for jointly learning both attribute classifiers and attribute vectors for known classes (but unknown attributes) (Sections 3 and 4.1).

2. We show that class-level information helps in learning a better attribute-level representation. We present results for the **Animals with Attributes (AwA)** and **a-Yahoo, a-Pascal** datasets (Section 5). We get similar or better classification accuracy compared to when both the classes and their attribute vectors are known, while still generalizing well to unseen classes.

3. We extend our framework to a novel transductive setting where we have unlabeled images from *new* classes also (unlike traditional transductive setting) and additionally learn attribute vectors and labels for these classes (Sections 3 and 4.2). We get significant improvement in accuracy (about 5% for **AwA** and 8% for **a-Yahoo, a-Pascal**).

2. Related Work

There has been a recent surge in interest in attribute-based object recognition [7, 6, 9, 8, 5, 1, 11]. Farhadi et al. [6] and Lampert et al. [9] present a number of innovative applications that demonstrate the usefulness of semantic attributes. Both of these papers look at the problem of detecting unseen classes at test time, but as mentioned before, they use hand-specified attribute based descriptions, whereas we learn them. Farhadi et al. describe a procedure for explicitly learning a set of *discriminative* attributes (in addition to a pre-defined set of semantic ones). It considers various random two-way splits of classes and trains binary classifiers to distinguish them. Such an approach still does not take into account whether attributes are redundant, so the final set of attributes may not be compact.

Dietterich and Bakiri [4] present algorithms for learning Error-Correcting Output Codes (ECOCs) for a multi-class problem. ECOCs were originally proposed as a purely top-down way of learning a binary encoding of class labels. The encoding does not have any pre-defined semantic interpretation. Note that ECOCs do not consider unseen classes. Subsequent papers by others have considered designing codes that depend not only on the class label, but has bottom-up dependencies on the input as well (e.g. [10]). Our work can be seen as combining ideas from ECOCs with semantic attributes to learn binary encodings both for classes with labeled examples as well as unseen classes.

A number of papers deal with not only detecting attributes, but also localizing them within the image [7, 5, 1]. This is not the focus of our work, although our approach can be modified to incorporate it.

Tuytelaars et al. [12] describe an empirical evaluation of various unsupervised learning methods for discovering object classes; but here we use supervision with the expectation that it will allow better generalization to unseen classes.

3. Overview of Our Approach

Our goal is to learn the attribute classifiers and attribute vectors jointly. We have two settings: 1) supervised learning and 2) transductive learning. In both settings, we have a set of positive and negative examples for training each of the attribute classifiers (Figure 1). The two settings differ in what kind of training data is available for the joint optimization. Table 1 shows the notation used in the paper.

Supervised Learning: Figure 1(b) shows the inputs in this case. A class-labeled dataset (*dolphin*, *deer* in Figure 1(b)) is available during training. Note that the mapping from attributes to classes are not known at the beginning of the learning (denoted by the *missing links*). Here, the goal is to learn this mapping ($\mathbf{R}_K$), and the attribute classifiers ($\mathbf{W}$). Our formulation (section 4.1) addresses this problem by jointly optimizing $\mathbf{R}_K$ and $\mathbf{W}$ with some attribute vector constraints. Constraints ensure that 1) the attribute vectors discriminate the classes well, and 2) the attributes are not redundant. We use two constraints, namely, *row separation* and *column separation* on the rows and columns of the matrix $\mathbf{R}_K$. Row separation measures the *Hamming* distance between the attribute vectors – maintaining some mean distance helps in identifying attribute vectors that discriminate the classes well. Column separation measures the correlation among the attribute vectors – setting the correlation to 0 removes redundancy. *After learning*, $\mathbf{R}_K$ is known, as indicated by the links in Figure 1(c).

When the unlabeled data from new, unseen classes become available, we can learn the attribute vectors for those

[1] Neither class nor attribute labels are given for these classes.

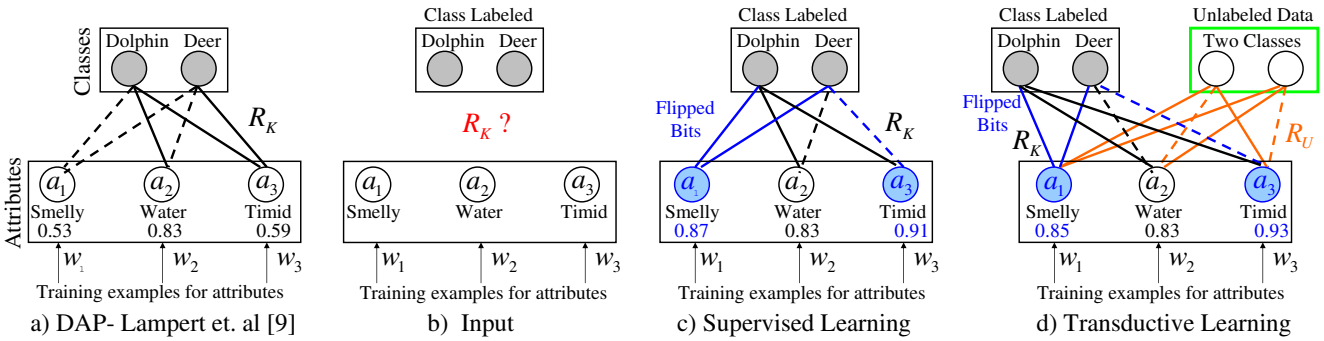


Figure 1. DAP Model (a) and our learning setups (b-d). a) The DAP model assumes that the attribute vectors ($\mathbf{R}_K$) are known as shown by solid line (1) and dashed line (0). Computed AUC values corresponding to some attributes (e.g.,*Smelly*) are shown. b) Class labeled and attribute labeled data needed by our approach (with unknown $\mathbf{R}_K$) are shown. c) After learning, $\mathbf{R}_K$ is known. Retained semantic attribute and undetectable semantic attributes that got flipped (blue) are shown. d) Additional unlabeled data (green box) and unknown attribute vectors ($\mathbf{R}_U$) for the unseen classes are also shown.

$\mathbf{x}$: Feature vector of an image
y: Class label of an image
N_a: Number of attributes
$a_{i,j}$: j-th attribute value of i-th image ($a_{i,j} \in \{0,1\}$)
$\mathcal{T}_A = \{(\mathbf{x}_{i,j}, a_{i,j}) : i = 1,\ldots,N_j, j = 1,\ldots,N_a\}$ *attribute labeled dataset*
N_j: Number of examples to train j-th attribute classifier
$\mathbf{w}_j$: Model weight vector of j-th classifier
$\mathbf{W} = \{\mathbf{w}_j : j = 1,\ldots,N_a\}$: Full attribute classifier models
$\mathcal{T}_C = \{(\tilde{\mathbf{x}}_i, y_i) : i = 1,\ldots,N_{\tilde{x}}\}$: *class labeled dataset*
$\mathcal{Y}_K, \mathcal{Y}_U$ (known, unseen/unknown set of classes)
$N_{\tilde{x}}$: Number of examples in $\mathcal{T}_C$
c_y: Number of examples with label y
$\mathcal{T}_U = \{\bar{\mathbf{x}}_i : i = 1,\ldots,N_{\bar{x}}\}$: *unlabeled dataset*
$\mathbf{r}_y$: Binary attribute vector for a class y ($r_{y,j}$:j-th element)
$\mathbf{R}_K, \mathbf{R}_U$: Matrices - each row is transpose of $\mathbf{r}_y, y \in \mathcal{Y}_K, \mathcal{Y}_U$
$\bar{\mathbf{r}}_j$: A column vector in $\mathbf{R}_K$ or $\mathbf{R}_U$

Table 1. Notations

classes from the outputs of the attribute classifiers using a mixture of Bernoulli model [2].

Discussion: In our approach, the joint optimization exerts an additional *top-down* influence from the class labels to the attribute-level representation (Figure 1(c)). This is in contrast to the purely *bottom-up* methods which learn the attribute classifiers using only attribute-labeled data, without any class-level information [6, 9] (Figure 1(a)). Our combined top-down and bottom-up approach has the desirable characteristic of *retaining discriminative semantic attributes* and *flipping attributes that are not useful in discriminating the classes*. Some of the non-detectable *semantic* attributes like *timid* and *smelly* are flipped (highlighted in blue in Figure 1(c)), while visually meaningful attributes like *water* are retained. Note that the Area Under ROC (AUC) of the flipped attribute classifiers are higher. This characteristic helps in getting good generalization performance on both seen and unseen classes, *even in the unknown* $\mathbf{R}_K$ *case*.

Transductive Learning: Figure 1(d) shows this setting. In addition to the inputs of the supervised setting, we also use unlabeled data from unknown classes (green box in Figure 1(d)) during training. We assume that the known classes and unknown classes do not overlap[2]. This case is of high importance since the main goal in the attribute based object recognition framework is to generalize well on *completely* unseen classes. We distinguish between unknown classes and unseen classes: unseen classes are not used during training, while unknown classes are used during training but without class label information. Again, the mapping from attributes to classes $\mathbf{R}_K$ and $\mathbf{R}_U$ (for unknown classes) are not known at the beginning of the learning.

Our approach takes the conventional transductive learning one step further by *additionally learning the attribute vectors for the unknown classes* along with the assignment of unlabeled examples to clusters. Thus, the mapping between the outputs of the attribute classifiers and unknown classes become known (see Figure 1(d)). We jointly optimize $\mathbf{W}$, $\mathbf{R}_K$ and $\mathbf{R}_U$ with attribute constraints (discussed earlier). Learning $\mathbf{R}_U$ jointly with $\mathbf{W}$ and $\mathbf{R}_K$ is far more powerful in getting good accuracy on the unlabeled data ($\mathcal{T}_U$), compared to learning the attribute vectors ($\mathbf{R}_U$) separately after supervised learning. However, this is possible only when the unlabeled data is available during training.

Since the classes are unknown ($\mathcal{Y}_U$), the unlabeled data ($\mathcal{T}_U$) is grouped into multiple clusters after learning, with each cluster identified by an attribute vector in $\mathbf{R}_U$. While the attribute vector of a cluster gives an object description of the images present in the cluster, manual labeling is required to associate a cluster with a class label such as Tiger. This can be done efficiently by looking at a few images per cluster and assigning a label at the cluster level, rather than at the image level.

It is well-known that label constraints are important to get good solutions in transductive and semi-supervised

[2]The general case of the unseen classes containing a subset of known classes can be handled by our approach with some modifications in our formulations and will be presented in a future work.

learning settings. As constraints, we assume that the number of classes in $\mathcal{Y}_U$ is known and place constraints on the cluster sizes to prevent highly skewed clusters.

4. Formulations and Optimization

4.1. Supervised Learning

Given $\mathcal{T}_A$ and $\mathcal{T}_C$, the goal is to *jointly* learn the attribute classifiers ($\mathbf{W}$) and the attribute vectors ($\mathbf{R}_K$). We solve this learning problem by optimizing the following objective function $G_s(\mathbf{W}, \mathbf{R}_K; \mathcal{T}_A, \mathcal{T}_C)$:

$$G_a(\mathbf{W}; \mathcal{T}_A) + \lambda_c \mathcal{L}_C(\mathbf{W}, \mathbf{R}_K; \mathcal{T}_C) + \lambda_r \mathcal{L}_r(\mathbf{R}_K) \quad (1)$$

where λ_c and λ_r are regularization constants. We explain these terms below. *Henceforth, we show the dependency on the datasets only when a term is introduced; in other places, we suppress it for brevity.*

L2-Regularized Loss Function $G_a(\mathbf{W}; \mathcal{T}_A)$: This term is given by: $G_a(\mathbf{W}; \mathcal{T}_A) = \frac{1}{2}\sum_{j=1}^{N_a} ||\mathbf{w}_j||^2 - \frac{\lambda_a}{N_a}\sum_{j=1}^{N_a}\frac{1}{N_{x,j}}\sum_{i=1}^{N_{x,j}} \log p(a_{i,j}|\mathbf{x}_{i,j}; \mathbf{w}_j)$ where λ_a is a regularization constant. We use the logistic regression model, $p(a|\mathbf{x}; \mathbf{w}) = \frac{1}{1+\exp(-a\mathbf{w}^T\mathbf{x})}$[3] with linear kernel.[4]

Predictive Likelihood Term $\mathcal{L}_C(\mathbf{W}, \mathbf{R}_K)$: Before we define this term, we define a *class predictive likelihood score* for each example, given the attribute vector ($\mathbf{r}_y$). This score is computed using the attribute classifiers and is given by: $F(\mathbf{r}_y; \mathbf{x}, \mathbf{W}) = \sum_{j=1}^{N_a} \log p(a_j = r_{y,j}|\mathbf{x}, \mathbf{w}_j)$; this can also be written as $\sum_{j=1}^{N_a} r_{y,j} \log p(a_j = 1|\mathbf{x}, \mathbf{w}_j) + (1 - r_{y,j}) \log p(a_j = -1|\mathbf{x}, \mathbf{w}_j)$. Furthermore, using this score we can find the Maximum A Posteriori (MAP) label estimate as: $\hat{y} = \arg\max_{y \in \mathbf{R}_K} F(\mathbf{r}_y; \mathbf{x}, \mathbf{W})$. Then, we have:

$$\mathcal{L}_C(\mathbf{W}, \mathbf{R}_K) = -\frac{1}{N_{\tilde{x}} N_a} \sum_{y \in \mathcal{Y}_K} \sum_{i:y_i = y} F(\mathbf{r}_y; \tilde{\mathbf{x}}_i, \mathbf{W}). \quad (2)$$

This term facilitates joint learning of $\mathbf{R}_K$ and $\mathbf{W}$.

Attribute Vector Regularization Term $\mathcal{L}_r(\mathbf{R}_K)$: The role of this term is to enforce the row and column separation attribute constraints by regularizing the attribute matrix $\mathbf{R}_K$. We assume that the distances among the classes (row separation) and correlations (column separation) among the attribute vectors come from Gaussian distributions. Then, it is sufficient to specify the desired means and variances for the row and column separations (denoted by (μ_r, σ_r^2) and (μ_c, σ_c^2)). We define $\mathcal{L}_r(\mathbf{R}_K)$ as a sum of two terms:

ROW SEPARATION TERM: This terms measures the Kullback-Leibler (KL) divergence between the reference Gaussian distribution with mean and variance (μ_r, σ_r^2), and empirical Gaussian distribution with mean and variance $(\hat{\mu}_r, \hat{\sigma}_r^2)$ computed using distances between the rows of $\mathbf{R}_K$. We compute the distance between every pair of attribute vectors (y, y') as $\eta_{y,y'} = \mathbf{r}_y^T(\mathbf{1} - \mathbf{r}_{y'}) + (\mathbf{1} - \mathbf{r}_y)^T\mathbf{r}_{y'}$, $y, y' \in \mathcal{Y}_K$ and $y \neq y'$, where $\mathbf{1}$ is a vector of all ones; note that $\eta_{y,y'} = \eta_{y',y}$.

COLUMN SEPARATION TERM: This term is similar to the row separation KL-divergence term. Here, we compute $(\hat{\mu}_c)$ and $(\hat{\sigma}_c^2)$ for the column separation using the distances $\gamma_{i,j} = (2\bar{\mathbf{r}}_\mathbf{i} - \mathbf{1})^T(2\bar{\mathbf{r}}_\mathbf{j} - 1), i, j = 1, \ldots, N_a$ and $i \neq j$. Note that we capture both positive and negative correlations.

4.2. Transductive Learning

In this formulation, given $\mathcal{T}_U$ (in addition to $\mathcal{T}_A, \mathcal{T}_C$), the goal is to *jointly* learn $\mathbf{W}$, $\mathbf{R}_K$ and $\mathbf{R}_U$. Furthermore, we group the examples in $\mathcal{T}_U$ into clusters. To address this learning problem, we propose to optimize the following objective function $G_t(\mathbf{W}, \mathbf{R}_K, \mathbf{R}_U, \mathbf{Q}; \mathcal{T}_A, \mathcal{T}_C, \mathcal{T}_U)$:

$$G_s(\mathbf{W}, \mathbf{R}_K) + \lambda_u \mathcal{L}_U(\mathbf{W}, \mathbf{R}_U, \mathbf{Q}; \mathcal{T}_U) + \lambda_l \mathcal{L}_l(\mathbf{Q}) \quad (3)$$

where λ_u and λ_l are regularization constants, $\mathbf{Q}$ denotes a set of variables $\{q_{i,y} : i = 1, \ldots, N_{\bar{x}}, \forall y \in \mathcal{Y}_U\}$; $q_{i,y}$ is an indicator variable such that $q_{i,y} = 1$ when i-th example belongs to the cluster with label y and 0 otherwise. The first term is nothing but (1) except that we modify the attribute regularization term $\mathcal{L}_r(\mathbf{R}_K)$ by including $\mathbf{R}_U$ in the computation of the means and variances of the row and column separation. We define the remaining terms below.

Predictive Likelihood Term $\mathcal{L}_U(\mathbf{W}, \mathbf{R}_U, \mathbf{Q})$: Using $\mathbf{Q}$, we define this term as a generalized version of (2):

$$\mathcal{L}_U(\mathbf{W}, \mathbf{R}_U, \mathbf{Q}) = -\frac{1}{N_{\bar{x}} N_a} \sum_{i=1}^{N_{\bar{x}}} \sum_{y \in \mathcal{Y}_U} q_{i,y} F(\mathbf{r}_y; \bar{\mathbf{x}}_i, \mathbf{W}) \quad (4)$$

and it also plays the role of learning the attribute classifiers. Note that (4) is similar to (2) when Q is known.

Label Regularization Term $\mathcal{L}_l(\mathbf{Q})$: As mentioned earlier, we assume that the number of classes in $\mathcal{Y}_U$ is known. Furthermore, we use lower and upper bound constraints such as: $n_l \leq \bar{c}_y \leq n_u, \forall y$ where $\bar{c}_y$ denotes the number of examples with class label y; n_l, n_u denote some lower and upper bounds, and $\sum_{y \in \mathcal{Y}_U} \bar{c}_y = N_{\bar{x}}$. However, instead of solving a constrained optimization problem, we enforce the above constraints by adding $\mathcal{L}_l(\mathbf{Q})$ defined as:

$$\sum_{y \in \mathcal{Y}_U} \max(0, (\sum_i q_{i,y} - n_u))^2 + \max(0, (n_l - \sum_i q_{i,y}))^2 \quad (5)$$

Thus, the squared terms penalize the solution when the constraints are violated.

4.3. Optimization

Supervised Learning: In order to optimize the objective function $G_s(\mathbf{W}, \mathbf{R}_K)$ (see (1)), we perform an alternating

[3] Note that there is a conversion from {0,1} to {-1,+1} done here.

[4] Note that our formulation can handle nonlinear kernels as well.

optimization of $\mathbf{W}$ and $\mathbf{R}_K$. In the first step, $\mathbf{W}$ is optimized keeping $\mathbf{R}_K$ fixed. In the second step, $\mathbf{R}_K$ is optimized keeping $\mathbf{W}$ fixed. We convert the problem of discrete optimization over $\mathbf{R}_K$ to a continuous optimization problem (see appendix). We use standard conjugate gradient algorithm to optimize in each step and repeat until there is no significant improvement (i.e., the improvement is less than a user defined parameter ϵ). The overall algorithm is given in Algorithm 1.

Transductive Learning: As earlier, we convert the discrete optimization problems in $\mathbf{R}_U$, and also $\mathbf{Q}$ into continuous optimization problems (see appendix). We optimize $\mathbf{R}_U$, $\mathbf{Q}$ using the alternating optimization technique. We indicate only the changes in the steps of the algorithm 1. We initialize $\mathbf{Q}$ randomly, but the assignments satisfy the clustering constraints. When $\mathbf{Q}$ is known, we just have the supervised learning problem; therefore, the steps (5)-(7) are applied to find both $\mathbf{R}_K$ and $\mathbf{R}_U$ with the modified objective function:$\lambda_c\mathcal{L}_C(\mathbf{W},\mathbf{R}_K)+\lambda_u\mathcal{L}_U(\mathbf{W},\mathbf{R}_U,\mathbf{Q})+\mathcal{L}_r(\mathbf{R}_K,\mathbf{R}_U)$. Then, keeping $\mathbf{R}_U$ and $\mathbf{W}$, we optimize $\lambda_u\mathcal{L}_U(\mathbf{W},\mathbf{R}_U,\mathbf{Q})+\mathcal{L}_l(\mathbf{Q})$ over $\mathbf{Q}$; this step is introduced between the steps (7) and (8). Finally, keeping $\mathbf{R}_K$, $\mathbf{R}_U$ and $\mathbf{Q}$ fixed, we modify the step (8) to optimize $G_t(\mathbf{W},\mathbf{R}_K,\mathbf{R}_U,\mathbf{Q})$ over $\mathbf{W}$ (see (4)).

Data: $\mathcal{T}_A,\mathcal{T}_C$; Parameters:$\{\lambda_a,\lambda_c,\lambda_r,\lambda_e\},\epsilon$
Result: $\mathbf{W},\mathbf{R}_K$

begin
  Initialize $\mathbf{W}=\mathbf{0}$; $t=1$
  Optimize $G_a(\mathbf{W})$ over $\mathbf{W}$
  while *t=1* **do**
    Initialize
    $\mathbf{r}_{j,y}=\frac{1}{c_y}\sum_{i:y_i=y}p(a_{i,j}=1|\tilde{\mathbf{x}}_i;\mathbf{w}_j),\forall j,y$
    $g_{old}=G_s(\mathbf{W},\mathbf{R}_K)$
    Keeping $\mathbf{W}$ fixed, optimize
    $\lambda_c\mathcal{L}_C(\mathbf{W},\mathbf{R}_K)+\mathcal{L}_r(\mathbf{R}_K)$ over $\mathbf{R}_K$
    Keeping $\mathbf{R}_K$ fixed, optimize $G_s(\mathbf{W},\mathbf{R}_K)$ over $\mathbf{W}$
    $g_{new}=G_s(\mathbf{W},\mathbf{R}_K)$
    if $|g_{new}-g_{old}|<\epsilon$ **then**
      $t=0$
    end
  end
end

Algorithm 1: Alternating Optimization Algorithm

5. Experimental Evaluation

We demonstrate the effectiveness of our approach using two benchmark datasets. We compare our methods with different baselines. We make several interesting observations from the learned attribute vectors. We also study the importance of attribute vector and label regularizations.

5.1. Experimental Setup

Datasets:

Animals with Attributes (AwA): This dataset created by Lampert et al. [9] contains 30,475 images of 50 animal classes. Each class is described by a set of 85 attributes such as *black, blue, striped* etc. We use the same 10 classes used by [9] as the unlabeled dataset ($\mathcal{T}_U$). To assess the ability of our methods to learn different attribute vectors, we create 4 different random partitions of the attribute labeled dataset ($\mathcal{T}_A$) and the class labeled dataset ($\mathcal{T}_C$) from the remaining 40 classes. We use 30 classes to create $\mathcal{T}_A$ and 10 classes for $\mathcal{T}_C$. The generated partitions are referred as *Cut1* to *Cut4*. For testing on the known classes, we used 70% and 30% train-test split.

a-Pascal, a-Yahoo: This dataset created by Farhadi et. al [6] has two sets of classes. **a-Pascal** consists of 20 classes with 6340 training images, and 6355 test images collected from PASCAL VOC 2008 challenge. **a-Yahoo** dataset has 12 classes (different from **a-Pascal**) with images collected from Yahoo images. The images are semantically described by a set of 64 attributes such as *metal, wheel* etc.

We split the **a-Pascal** data in two parts: 12 classes (*bicycle, person, sofa, cow, dog, dining table, bird, potted plant, cat, boat, car, aeroplane*) are used as $\mathcal{T}_A$, and remaining 8 classes (*bottle, chair, motorbike, bus, horse, tv monitor, train and sheep*) are used as $\mathcal{T}_C$. As in [6], we use the **a-Yahoo** dataset as $\mathcal{T}_U$. Since the attribute vectors are given at the image level (rather than at the class level), we compute $\mathbf{R}_K$ and $\mathbf{R}_U$ as the median of attribute predictions for the images in each class.

Features: For the **AwA** dataset, we use the precomputed PHOG features [3] provided by [9]. For the **a-Yahoo** and **a-Pascal** datasets, we use the precomputed 9751 dimensional features provided by [6]. These features are extracted from information on color, texture, visual words, and edges.

Parameter Settings: In the supervised setting (Section 4.1), there are four regularization parameters: $\lambda_a,\lambda_c,\lambda_r$ and λ_e (entropy regularization). Although there are four parameters, we found that tuning only two of them (λ_a and λ_e) was sufficient. We set λ_a using a validation set while learning the classifier models $\mathbf{W}$ using T_A only (i.e., the first term in (1)). We used a validation set to find the best value for λ_r in the interval $[0,4]$[5]. We set the other parameters as: $\lambda_c=\lambda_r$ and $\lambda_e=\frac{\lambda_c}{10}$. For the transductive setting there are two more parameters: λ_u,λ_l. Again, we used a validation set to find the best values for (λ_u,λ_r) in $[0,4]\times[0,4]$ over an 8×8 grid. We set the label regularization constant λ_l to $15\lambda_u$.

Evaluation Criteria: We evaluate the performance of different methods on several metrics.

[5]Since both attribute labeled term $G_a(\mathbf{W};\mathcal{T}_A)$ and predictive likelihood term $\mathcal{L}_C(\mathbf{W},\mathbf{R}_K)$ are log likelihood terms, this range works well.

	$\mathbf{W}$ Training			$\mathcal{T}_C$ (Testing)	$\mathcal{T}_U$ (Testing)
Method	$\mathcal{T}_A$	$\mathbf{R}_K$	$\mathbf{R}_U$	$\mathbf{R}_K$	$\mathbf{R}_U$
CLB1	✓	*	-	✓	✓
CLB2	✓	✓	-	✓	✓
CLB3	✓	✓	-	✓	Learned $\mathbf{R}_U$ (BMM)
SL	✓	X	-	Learned $\mathbf{R}_K$	Learned $\mathbf{R}_U$ (BMM)
TL	✓	X	X	Learned $\mathbf{R}_K$	Learned $\mathbf{R}_U$

Table 2. Methods for comparison. ✓ and X indicate that the designer specified attribute vectors are known and unknown respectively. $*$ and $-$ indicate that $\mathcal{T}_C$ and $\mathcal{T}_U$ is not used respectively. SL: Supervised Learning and TL: Transductive Learning. BMM: Bernoulli Mixture Models.

(1) For the attribute classifiers ($\mathbf{W}$), we use Area under ROC curve (AUC) as reported in other works.

(2) To evaluate the generalization performance on the known classes ($\mathcal{Y}_K$), we measure the accuracy on the unseen examples with the MAP label estimates.

(3) We make several observations from the attribute matrix $\mathbf{R}_K$ (learned using our supervised learning method) in terms of visually detectable semantic attributes, non-discriminative and discriminative attributes.

(4) To evaluate the performance on the unknown classes, we measure the accuracy on $\mathcal{T}_U$. For the baselines (Table 2) with known $\mathbf{R}_U$ we use the MAP label estimates. For the unknown $\mathbf{R}_U$ scenario, we learn $\mathbf{R}_U$ using Bernoulli mixture models (BMM) with the number of clusters set equal to the number of unseen classes. To measure the accuracy after the cluster assignments are made using the learned attribute vectors ($\mathbf{R}_U$), we use the Hungarian algorithm to match the clusters with the classes ($\mathcal{Y}_U$).

Baselines for Comparison: Table 2 shows three baselines - CLB1, CLB2, CLB3 along with our methods. In CLB1, the attribute classifiers $\mathbf{W}$ are learned using only $\mathcal{T}_A$. During testing phase, we assume that $\mathbf{R}_K$ and $\mathbf{R}_U$ are known. In CLB2, $\mathcal{T}_C$ is also used along with known $\mathbf{R}_K$ during training. The difference between CLB2 and CLB3 is only during testing on $\mathcal{T}_U$. While we use the designer specified class attribute matrix ($\mathbf{R}_U$) in CLB2, we use $\mathbf{R}_U$ learned using BMM in CLB3. Note that CLB2 and CLB3 are hard baselines since $\mathbf{R}_K$ is known during training.

5.2. Experimental Results

In this section, we compare various methods and provide key insights into the attribute vector ($\mathbf{R}_K$) learning part of our method. We also present the performance results with and without attribute vector and label regularizations.

5.2.1 Supervised Learning

Generalization Performance on Known Classes ($\mathcal{Y}_K$): Table 3 shows the classification accuracy of different methods on unseen examples of the known classes. Since CLB1 does not use $\mathcal{T}_C$ to train $\mathbf{W}$, its performance is not good. CLB2 improves over CLB1 significantly. This is expected since the examples belonging to $\mathcal{T}_C$ are used during training and evaluation is done on the same set of classes (though on unseen examples). As mentioned earlier, CLB2 and CLB3 differ only during testing on the unlabeled data. Our supervised learning (SL) method has similar or even better accuracies than the hard baseline CLB2. Note that for the **a-Pascal** dataset, the performance is better by more than 5%. This experiment clearly demonstrates that our method with the attribute vector learning component generalizes well on the known classes and performs comparable or better than the methods that make use of $\mathbf{R}_K$.

Dataset	CLB1	CLB2/CLB3	SL	TL
AwA (*Cut1*)	30.92	54.00	**54.62**	52.6
AwA (*Cut2*)	21.51	44.90	**46.43**	44.61
AwA (*Cut3*)	18.77	51.40	**52.04**	51.85
AwA (*Cut4*)	23.02	49.50	**50.83**	48.88
a-Pascal	30.83	62.01	**67.33**	65.33

Table 3. Classification accuracy (%) on the unseen examples of known classes. CLB2 and CLB3 are same while testing on $\mathcal{T}_C$ with known $\mathbf{R}_K$.

Dataset	CLB2	CLB3	SL	TL
AwA (*Cut1*)	26.17	26.8	*28.28*	**33.61**
AwA (*Cut1*)	24.89	*26.34*	25.57	**31.00**
AwA (*Cut1*)	27.02	27.12	*28.96*	**32.33**
AwA (*Cut1*)	22.00	*27.65*	27.48	**33.34**
a-Yahoo	28.14	28.78	*29.77*	**37.93**

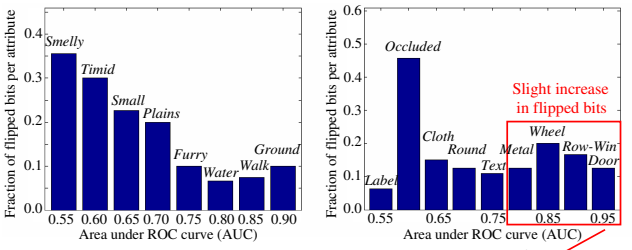
Table 4. Classification accuracy (%) on the unlabeled data ($\mathcal{T}_U$) with unseen/unknown classes. Unseen classes case: CLB2, CLB3 and SL do not use $\mathcal{T}_U$. Unknown classes case: TL uses $\mathcal{T}_U$ without class label information.

Generalization Performance on Unseen Classes ($\mathcal{Y}_U$): Table 4 shows the classification accuracy on the unlabeled data with unseen/unknown classes. Note that CLB2, CLB3 and SL do not use the unlabeled data during training. Therefore, evaluation on the unlabeled data means evaluation on the *unseen* classes. The performance of all the methods (first three columns) are similar. CLB2 is a hard baseline since it assumes that $\mathbf{R}_U$ is known during testing. This result demonstrates that the attribute classifiers ($\mathbf{W}$) learned by our supervised learning method generalizes well *even on the unseen classes*.

Gain from Joint Learning: We compared our joint learning framework with stage-wise learning in which attribute classifers and object descriptions are learned separately. For the **AwA** dataset, the joint learning gave an average improvement of 10% over the four cuts; while on the **Yahoo-Pascal** dataset, the improvement was 8%. The corresponding improvement for the case of unseen classes was about

	AwA				a-Pascal
	Cut1	*Cut2*	*Cut3*	*Cut4*	
Flips (%)	21.41	24.46	26.47	23.74	11.51
AUC ($\mathcal{T}_C$) (CLB1)	0.64	0.62	0.61	0.66	0.70
AUC ($\mathcal{T}_C$) (SL)	0.84	0.78	0.82	0.84	0.91
Acc. ($\mathcal{T}_C$) (%) (Semantic)	52.32	40.48	41.37	48.22	65.48
Acc. ($\mathcal{T}_C$) (%) (Full)	54.62	46.43	52.04	50.83	67.33

Table 5. Analysis of $\mathbf{R_K}$



a) ***AwA*** – Cut 1 b) ***a – Pascal***

	Foot	Leg	Row Wind	Door
Bus	0.00	0.00	0.89	0.47
Horse	0.72	0.86	0.00	0.00
Train	0.00	0.00	0.50	0.49
Sheep	0.42	0.63	0.00	0.00

c) Fraction of positive labels per class for flipped attributes in red box (***a – Pascal***)

Figure 2. Analysis of $\mathbf{R_K}$

3% for both the datasets.

Observations from Attribute Vector Learning: To get insights into the attribute vector learning aspects of our method, we measured several metrics and they are reported in Table 5. The first row shows the percentage of bits flipped (i.e., that got changed) in the known $\mathbf{R_K}$. The average percentage of flips is around 24% over the different cuts of the **AwA** dataset. For the **a-Pascal** dataset, the percentage is 11.52%. Note that our method retains the majority of the bits since many semantic attributes are visually detectable, thereby ensuring good generalization on unseen classes.

To get more insights, we analyzed the fraction of bits that got flipped as a function of attributes. The results are presented in Figure 2a and b for the two datasets. We computed the AUC score of each attribute classifier (trained using $\mathcal{T}_A$ and tested on $\mathcal{T}_C$). The following observations can be made from the figure. (1) The AUC score for the attributes like *Smelly*, *Timid*, etc. is less. The AUC plots shown in Lampert et al.[9] also have similar behaviors. (2) The average number of flips is significantly higher for the attributes with low AUC scores (more prominently seen in the **AwA** dataset). Thus, visually non-detectable attributes get modified by our method during the attribute vector learning.

	AwA				a-Pascal
	Cut1	*Cut2*	*Cut3*	*Cut4*	
WOC	49.58	42.60	44.51	48.99	60.31
CS Only	53.00	45.85	45.89	48.17	63.44
RS Only	54.71	45.80	48.21	51.30	67.33
CS + RS	54.62	46.43	52.04	50.83	67.33

Table 6. Accuracy on known classes $\mathcal{Y}_K$ with and without attribute vector regularization (WOC: Without Constraints, CS: Column Separation, RS: Row Separation).

On the other hand, visually recognizable semantic attributes such as *Water* are retained with very few bits of flip or no flip at all across the classes, ensuring good generalization performance on the unseen classes. (3) On the **a - Pascal** dataset (Figure 2b), there is a slight increase in the fraction of flips for some attributes with high AUC values (see the red box). To understand this behavior, we looked at the attribute values of examples as a function of classes. Figure 2c shows the fraction of positive labels for some of the attributes and classes. Note that many values are close to 0.50. For example, depending upon the viewing direction, the attribute *door* might not be visible for the classes *bus* and *train*. Such attributes are not discriminative[6]. Therefore, they get flipped.

We measured the average AUC score of the attribute classifiers on both ($\mathcal{T}_A$ and $\mathcal{T}_C$) with and without attribute vector learning. The second and third rows in Table 5 show the scores on $\mathcal{T}_C$. Note that by flipping the attributes with low AUC scores, the AUC scores have improved significantly (by around 0.2). This suggests that several discriminative attributes have been identified. This has some effect on the AUC scores computed on $\mathcal{T}_A$; however, we observed that the drop was quite small (around 0.02). To evaluate the importance of the flipped attributes we compared the classification accuracy on $\mathcal{T}_C$ with and without these attributes. We predicted the class labels with the full learned $\mathbf{R}_K$, and after removing the attributes (columns) with more than 25% flipped bits. On comparing the last two rows, we see that these attributes are important since they give significant improvement (even more than 5% in a couple of cases).

Importance of Attribute Regularization: Table 6 shows the classification accuracy results on the known classes $\mathcal{Y}_K$ with and without attribute regularization. On comparing the first and last row, we see that significant improvements of 4.5% and 7.0% are achieved on the **AwA** and **a-Pascal** datasets respectively. If we include only column separation regularization (second row), the performance improves significantly. We observed that the average decrease in the correlation was around 22% (with the maximum decrease of 32%) for the **AwA**(*Cut2*) dataset. Similarly, adding row

[6]To use such attributes we need multiple attribute vector representation per class.

separation regularization (third row) improves the accuracy by maintaining sufficient mean *Hamming* distance between the classes. As the results show, the row separation regularization is sufficient in many cases. However, for the **AwA**(*Cut3*) dataset, the column separation regularization helps in improving the accuracy by around 4%. Thus, it is useful to have column separation regularization since its inclusion gives significant improvement whenever possible (while maintaining the same performance in other cases).

5.2.2 Transductive Setting

Performance on Known and Unknown Classes: The last column in Table 4 shows the performance of our TL method on the unlabeled data ($\mathcal{T}_U$)[7]. Significant accuracy improvements are clearly seen on all the datasets. For different cuts of **AwA** dataset, the average improvement is around 5.3%. On the **a-Yahoo** dataset, the improvement is $> 8\%$ and the performance is even 6% better than the feature selection approach proposed by Farhadi et. al [6] with known $\mathbf{R}_U$. The last column in Table 3 indicates the accuracy of the TL method on the unseen examples of the known classes. Although there is approximately $1-2\%$ degradation compared to our SL method, its generalization is quite good. These results clearly demonstrate the effectiveness of the TL method in the attribute based object recognition framework.

Label Regularization: We compared the performance with and without label regularization. On the **a-Yahoo** dataset, the accuracy improved from 29.88% to 37.83% with the label regularization term. We observed that highly skewed clusters were formed without regularization. For example, the classes *zebra, wolf, donkey* have several common semantic attributes and only two clusters (instead of four) were formed with images mainly belonging to these classes. For the **AwA** dataset, the attributes for the classes are relatively far apart. Hence, the accuracy did not change much.

6. Conclusions and Future Work

We proposed an approach to learn attribute-based object descriptions jointly with the attribute classifiers. We use class label information to improve the attribute-level representation by converting undetectable and non-discriminative attributes into discriminative ones. We showed that 1) it is possible to learn good descriptions from data while generalizing well to unseen classes, and 2) including images from unseen classes into the learning significantly improves the accuracy on those classes. Future extensions include learning more expressive representations, such as allowing multiple descriptions per class (e.g. using a mixture model), and reducing human supervision even further by automatically discovering semantic attributes.

[7] Note that $\mathcal{T}_U$ is used during training, but without the class label information. Therefore, evaluation on $\mathcal{T}_U$ means evaluation on *unknown* classes (in contrast to the *unseen* classes).

References

[1] T. L. Berg, A. C. Berg, and J. Shih. Automatic attribute discovery and characterization from noisy web data. In *ECCV*, ECCV'10, pages 663–676, Berlin, Heidelberg, 2010. Springer-Verlag. 2

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 3

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, pages 401–408, 2007. 5

[4] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 2

[5] A. Farhadi, I. Endres, and D. Hoiem. Attribute-centric recognition for cross-category generalization. *CVPR*, 0:2352–2359, 2010. 2

[6] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing Objects by their Attributes. In *CVPR*, 2009. 1, 2, 3, 5, 8

[7] V. Ferrari and A. Zisserman. Learning visual attributes. In *NIPS*, Dec. 2007. 2

[8] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *ICCV*, Oct 2009. 1, 2

[9] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to Detect Unseen Object Classes by Between-Class Attribute Tranfser. In *CVPR*, 2009. 1, 2, 3, 5, 7

[10] O. Pujol, P. Radeva, and J. Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *PAMI*, 28:1007–1012, 2006. 2

[11] Y. Su, M. Allan, and F. Jurie. Improving object classification using semantic attributes. In *Proceedings of the British Machine Vision Conference*, pages 26.1–26.10, 2010. 2

[12] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine. Unsupervised object discovery: A comparison. *IJCV*, 88:284–302, June 2010. 2

APPENDIX

Optimization over $\mathbf{R}_K$: We optimize (1) over $\mathbf{R}_K$ by solving an unconstrained optimization problem using the transformation: $r_{y,j} = \frac{1}{1+exp(-d_{y,j})}$ and optimizing on the variables $d_{y,j}, \forall y, j$ over $r_{y,j} \in [0,1], \forall y \in \mathcal{Y}_K, j = 1, \ldots, N_a$. To push the solution to the boundary, we add an entropy regularization term. That is, we add $-\lambda_e \sum_{y,j} r_{y,j} \log r_{y,j} + (1 - r_{y,j}) \log(1 - r_{y,j})$ to the KL-divergence terms that penalize violations of the row and column separation constraints. Here, λ_e is an entropy regularization constant.

Optimization over Q: We define $q_{i,y} = \frac{\exp(\frac{e_{i,y}}{T})}{\sum_{y \in \mathcal{Y}_U} \exp(\frac{e_{i,y}}{T})}$ and optimize over the variables $\{e_{i,y}, \forall i, y\}$. For small value of T, the maximum element in $\{q_{i,y}, \forall y\}$ gets a value close to 1, and we used $T = 0.1$ in our experiments. Thus, $\bar{c}_y \approx \sum_{i=1}^{N_{\bar{x}}} q_{i,y}$ and $\sum_{y \in \mathcal{Y}_U} \bar{c}_y = N_{\bar{x}}$.