Sharing Features Between Objects and Their Attributes

Sung Ju Hwang[1], Fei Sha[2] and Kristen Grauman[1]
[1]Department of Computer Science
University of Texas at Austin
{sjhwang,grauman}@cs.utexas.edu
[2]Computer Science Department
University of Southern California
feisha@usc.edu

Abstract

Visual attributes expose human-defined semantics to object recognition models, but existing work largely restricts their influence to mid-level cues during classifier training. Rather than treat attributes as intermediate features, we consider how learning visual properties in concert *with object categories can regularize the models for both. Given a low-level visual feature space together with attribute- and object-labeled image data, we learn a shared lower-dimensional representation by optimizing a joint loss function that favors common sparsity patterns across both types of prediction tasks. We adopt a recent kernelized formulation of convex multi-task feature learning, in which one alternates between learning the common features and learning task-specific classifier parameters on top of those features. In this way, our approach discovers any structure among the image descriptors that is relevant to both tasks, and allows the top-down semantics to restrict the hypothesis space of the ultimate object classifiers. We validate the approach on datasets of animals and outdoor scenes, and show significant improvements over traditional multi-class object classifiers and direct attribute prediction models.*

1. Introduction

Visual attributes are human-understandable properties shared among object categories (e.g., "glassy", "has legs"), and are a compelling way to introduce high-level semantic knowledge into predictive models. Recent work shows that attributes are valuable in several interesting scenarios, ranging from description of generic images or unfamiliar objects [11, 9, 24], to zero-shot transfer learning [13], to intermediate features that aid in distinguishing people, objects, and scenes [12, 13, 9, 27].

Existing approaches to attribute-based recognition assume that the attributes' role is primarily to focus learning effort on properties that will be reusable for many categories of interest, and to elegantly integrate human knowledge into discriminative models. As such, attribute classifiers are learned independently from object classifiers, and then their predictions are treated as "mid-level" features that bridge low-level image features and high-level object classes. However, segregating supervision about attributes from supervision about objects may restrict their impact. In particular, in conventional models, even though attributes influence object predictions, the attribute-labeled training data does not directly introduce *new* information when discriminatively learning the objects.

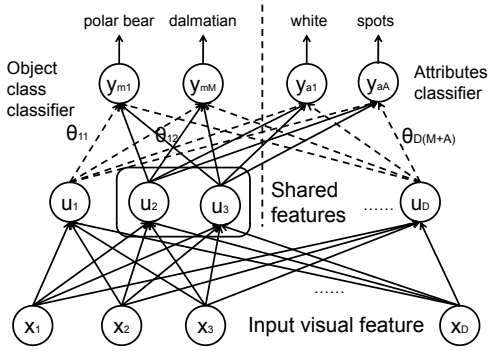


Figure 1. In our model, object categories and their human-defined visual attributes *share* a lower-dimensional representation (dashed lines indicate zero-valued connections), thereby allowing the attribute-level supervision to regularize the learned object models.

We explore how learning visual attributes *in concert* with object categories can strengthen recognition. The assumption is that both types of prediction tasks rely on some shared structure in the original image descriptor space. In other words, patterns among those generic visual properties that humans elect to name may reveal information about which low-level cues are valuable to object recognition—in the most general case, whether the objects of interest exhibit those attributes or not. Thus, rather than treat attributes as intermediate features, we propose an approach to discover this structure and learn a shared lower-dimensional representation amenable to discriminative models for either one (see Figure 1). In effect, we show how human-defined semantics (as revealed by attributes) can regularize training for object classifiers.

Given a low-level visual feature space together with attribute- and object-labeled image data, we learn a feature subspace for all labeling tasks based on a joint loss function that favors common sparsity. The optimization process alternates between regularizing towards shared features, and retraining task-specific classifiers based on those features. Our technique directly builds on a multi-task feature learning algorithm developed in [1], where it was applied to collaborative filtering of consumer data. To improve its scalability, we provide a more efficient kernelized implementation and linear algebra shortcuts for dealing with large matrices. Additionally, while in [1] all tasks are assumed to have the same label space, our setting entails non-overlapping label spaces (attributes, objects), for which feature-sharing is expected to be more challenging.

It is well-known that the success of multi-task learning or feature sharing hinges on the assumption that the input tasks are indeed related. Why should the assumption hold in our case? What makes attributes "special" as auxiliary tasks for object learning? Intuitively, their relation is intrinsic, since attributes are by definition shared among object categories. Many object-level distinctions can be made using a vocabulary of relevant properties, suggesting that a representation sufficient to distinguish the properties would also be relevant for the objects (e.g., a child learning to discriminate cows from other animals might focus on the visual properties a cow exclusively has but other animals do not). In fact, in early visual processing, it is known that the human visual system discovers some sparse coding using a feature "vocabulary" of low-level filters [17].

More abstractly, we expect that structure among a wide span of attribute classifiers could reveal information about which low-level features are valuable to human understanding of the visual world. That is, even attributes that are not relevant to distinguishing a particular object may still help to constrain the space of image descriptors suitable for higher-level recognition problems. Finally, there is a practical incentive for treating attributes as auxiliary tasks regarding supervision cost: for many attributes, knowing the real world object-attribute relationship is sufficient to transfer object-level image labels to attribute-level labels (i.e., all buildings are manmade, so if we have a labeled image of a building, it is also an image of the manmade attribute).[1]

In short, our contribution is threefold: 1) we design a method for feature sharing between object and attribute prediction tasks; 2) we verify with experiments on two datasets that feature sharing can offer noted improvements in accuracy for target object categorization tasks; and 3) we explore to what extent different attributes are useful for a target task, and provide some initial ideas for automatic selection of relevant attributes to limit training costs.

[1]This is the case for many binary attributes, but of course not all attributes (e.g., some bicycles are red, some are blue).

2. Related Work

Attributes and their applications: Recent work shows that attributes are useful in a variety of settings. First, they are independently useful to *describe* familiar and unfamiliar things (e.g., the leopard is spotted and furry, whether or not we know to call it a leopard [9, 11]), or to *search* through large image/video collections in semantic terms [24]. Second, they enable new *zero-shot learning* paradigms, where one can build an object model on the fly [13]. Third, they can serve as *mid-level features* to an object classification layer; having learned to predict the presence of each attribute, one can build supervised object models on top of those predictions [12, 13, 9, 27]. Usually attribute-object associations are manually specified, but some work explores ways to obtain them automatically [26, 6, 21]. Notably, nearly all models using attributes for recognition learn them independently.

Relating objects and attributes: The "indirect attribute prediction" model [13] offers a way to regularize attribute predictions based on object predictions; however, the attribute-object connections are set by human-given definitions, and so the two are not jointly learned. The novel multiple instance learning (MIL) approach in [25] jointly trains attribute and object detectors with weakly labeled data, with a constraint that both models should agree on localization (e.g., if an image is tagged "blue cap", both MIL classifiers should prefer to select positive training instances from the same location). In contrast, our data is strongly labeled, and our method influences the feature space construction, not training instance selection. The method in [27] integrates attribute- and object-based cues into a structured latent SVM model: the attribute labels are left as latent variables on the training data, and the objective is to minimize object prediction loss. In contrast, we show the value in discovering a single shared representation such that both attribute and object tasks can be predicted well. Thus, while [27] implicitly discovers object-attribute relationships, we exploit the two simultaneously as explicit tasks.

Transfer and multi-task learning: Transfer learning is a related way to regularize object learning. The focus has been on inter-class transfer between objects [10, 5, 22, 20], and jointly training detectors so that features are shared across objects [23, 3, 29], which lends to better efficiency during detection. In contrast, the proposed feature sharing spans multiple target label spaces, and our emphasis is generalization ability rather than efficiency. Multi-task learning [7] is often accomplished through feature sharing, and some vision work explores text [19, 14] or pattern matching [2] data as auxiliary tasks. We are the first to explore multi-task learning with attributes, which (relative to other sources of auxiliary tasks) has potential advantages of intrinsic task relevance and supervision "reuse", as discussed above.

3. Approach

We describe in detail the approach we take to learn shared features between objects and their attributes. Our work directly builds on a previous approach [1]. Being mindful of desired large-scale learning settings, however, we extend the method by providing faster and more scalable numerical techniques. Additionally, we adapt the models to handle classification tasks where the label sets are disparate.

We start by describing the basic setup for learning features from multiple tasks. Then we explain how the problem can be cast as convex optimization for both linear and kernel classifiers. Finally, we discuss extensions and improvements we have developed in order to apply the approach.

3.1. Basic Setup and Notation

There are two groups of classification tasks. We aim to improve *object* classification accuracy; thus, we refer to the objects as the *main* task, and the attribute classifiers as *auxiliary* tasks. Note that the two groups have different sets of labels.

We use multi-class support vector machines (SVMs) for the main task [8]. Let M denote the number of object classes, $\boldsymbol{x}_n \in \mathbb{R}^{\mathsf{D}}$ denote the n-th feature vector in the training data and y_n its class label. The multi-class SVM has M parameter vectors $\{\boldsymbol{w}_m^*\}_{m=1}^{\mathsf{M}}$, one for each class. In the most basic setting, we consider linear discriminants which are parameterized by $\boldsymbol{w}_m \in \mathbb{R}^{\mathsf{D}}$. Let $\boldsymbol{W}$ denote the matrix whose columns are $\boldsymbol{w}_m$. To identify $\boldsymbol{W}$, we minimize a loss function that maximizes the discriminant $\boldsymbol{w}_{y_n}^{\mathsf{T}}\boldsymbol{x}_n$,

$$\boldsymbol{W}^* = \arg\min \sum_n \ell(\{\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{x}_n\}_{m=1}^{\mathsf{M}}, y_n) + \gamma \sum_m \|\boldsymbol{w}_m\|_2^2$$

where $\gamma \geq 0$ is a tradefoff parameter that regularizes the model complexity, using the parameter's 2-norm.

For learning A auxiliary tasks, we use y_{na} to denote the label for the a-th auxiliary task and $\boldsymbol{w}_a$ for the corresponding model parameter. Our auxiliary tasks are binary classification of attributes. We use the squared hinge loss for these tasks. For simplicity, the notation assumes that both the main task and auxiliary tasks are trained on the same feature vectors. However, this is not mandatory, as we demonstrate in our results.

We use t ranging from 1 to $\mathsf{T} = (\mathsf{M} + \mathsf{A})$ to index all parameter vectors for the main and auxiliary tasks. To avoid unnecessary notation clutter, with a slight abuse, we use $\sum_{t=1}^{\mathsf{M}} \ell(\boldsymbol{w}_t^{\mathsf{T}}\boldsymbol{x}_n, y_{nt})$ in lieu of $\ell(\{\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{x}_n\}_{m=1}^{\mathsf{M}}, y_n)$, namely, the true object function for the main task.

3.2. Learning Shared Features via Regularization

Conventionally, all T parameters $\{\boldsymbol{w}_m\}_{t=1}^{\mathsf{T}}$ are learned by independently training $(1+\mathsf{A})$ classifiers. For linear discriminants such as $\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{x}_n$, the resulting parameter often reveals how effective features are. For instance, a zero-valued element w_{mi} indicates that the i-th feature of $\boldsymbol{x}_n$ does not play a role in classifying objects. Thus, intuitively, for related tasks, we expect their parameters to reveal similar sparsity patterns. Furthermore, we hypothesize that shared patterns will enable more effective parameter training—for example, reducing feature space dimensionality, thus improving classification performance. How can we identify such common patterns across tasks?

This desideratum is achieved in two steps. The first is to transform the original features to a shared feature space $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{x}_n \in \mathcal{U}$ for all tasks [1, 4]. The second step is to learn models in the space of $\mathcal{U}$ and promote a common sparsity pattern in the new parameters. Concretely, we express the discriminant in $\{\boldsymbol{\theta}_t\}$ such that $\boldsymbol{w}_t = \boldsymbol{U}\boldsymbol{\theta}_t$. Analogously to $\boldsymbol{W}$, we collect all $\boldsymbol{\theta}_t$ in $\boldsymbol{\Theta} \in \mathbb{R}^{\mathsf{D}\times\mathsf{T}}$. We jointly optimize all loss functions, but regularized with $\boldsymbol{\Theta}$'s $(2,1)$-norm,

$$\boldsymbol{\Theta}^*,\ \boldsymbol{U}^* = \arg\min \sum_t \sum_n \ell(\boldsymbol{\theta}_t^{\mathsf{T}}\boldsymbol{U}^{\mathsf{T}}\boldsymbol{x}_n, y_{nt}) + \gamma\|\boldsymbol{\Theta}\|_{2,1}^2 \tag{1}$$

The norm is given by $\|\boldsymbol{\Theta}\|_{2,1} = \sum_{d=1}^{D}\sqrt{\sum_t \theta_{td}^2}$. An important property of this norm is that it computes the 2-norm of parameter values in each dimension *across* tasks. Consequently, for any dimension d, the regularization attains the minimum if and only if the corresponding parameters are all zero: $\theta_{td} = 0$ for all t. Therefore, the regularization would choose the $\boldsymbol{\Theta}$ with the *smallest* number of *non-zero rows*.

The discriminant $\boldsymbol{\theta}_t^{\mathsf{T}}\boldsymbol{U}^{\mathsf{T}}\boldsymbol{x}_n$ depends only on nonzero elements of $\boldsymbol{\theta}_t$. Thus eq. (1) yields solutions that use a subset of features that are commonly effective for all tasks. Similar ideas have also been explored in other settings [28, 15].

The optimization of eq. (1) is challenging due to the non-smoothness of the regularization term. We next describe the alternating minimization algorithm proposed in [1].

3.3. Convex Optimization

The optimization algorithm of [1] starts by identifying eq.(1) with its equivalent form

$$\begin{aligned}\boldsymbol{W}^*,\ \boldsymbol{\Omega}^* = \arg\min &\sum_t \sum_n \ell(\boldsymbol{w}_t^{\mathsf{T}}\boldsymbol{x}_n, y_{nt}) \\ &+ \gamma \sum_t \boldsymbol{w}_t^{\mathsf{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{w}_t + \gamma\epsilon\,\mathsf{Trace}(\boldsymbol{\Omega}^{-1}),\end{aligned} \tag{2}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}}$ is constrained to be a positive definite matrix with bounded trace $\mathsf{Trace}(\boldsymbol{\Omega}) = 1$. $\epsilon \ll 1$ is a smoothing parameter for numerical stability and benign convergence properties (cf. Theorem 3 in [1]). $\boldsymbol{\Omega}$'s role can be understood more clearly by relating the solutions to the two problems eq. (1) and eq. (2):

$$\boldsymbol{W}^* = \boldsymbol{U}^*\,\boldsymbol{\Theta}^*,\ \boldsymbol{\Omega}^* = \boldsymbol{U}^*\,\mathsf{Diag}\left(\left\{\frac{\|\boldsymbol{\Theta}_d\|_2}{\|\boldsymbol{\Theta}\|_{2,1}}\right\}_{d=1}^{\mathsf{D}}\right)\boldsymbol{U}^{*\mathsf{T}} \tag{3}$$

Input: training data $(\boldsymbol{x}_n, \{y_{nt}\}), \epsilon, \gamma$
Output: $\boldsymbol{W}^*, \boldsymbol{\Omega}^*$
1: Initialize $\boldsymbol{\Omega}$ with a scaled identity matrix $\frac{1}{\mathsf{D}}\boldsymbol{I}$
2: **while** $\boldsymbol{W}$ still changes between two iterations **do**
3: Compute transformed variables according to eq. (6)
4: Solve $\hat{\boldsymbol{w}}_t$ according to eq. (5)
5: Compute $\boldsymbol{w}_t$ as $\boldsymbol{w}_t = \boldsymbol{\Omega}^{1/2}\hat{\boldsymbol{w}}_t$
6: Update $\boldsymbol{\Omega}$ according to eq. (7)
7: **end while**

Algorithm 1: Learning Shared Features for Linear Classifier [1]

where the operator $\mathsf{Diag}(\cdots)$ converts its D-element arguments as elements of a diagonal matrix. $\|\boldsymbol{\Theta}_d\|_2$ is the 2-norm of $\boldsymbol{\Theta}$'s d-th row: $\sqrt{\sum_t \theta_{td}^2}$. Intuitively, the diagonal measures relatively how much each row of $\boldsymbol{\Theta}$ is "non-zero". Therefore, the matrix $\boldsymbol{\Omega}$ measures relative effectiveness of each feature dimension.

We gain further insight by drawing an analogy to the maximum a posteriori (MAP) estimator when the prior distribution for the parameter $\boldsymbol{w}_t$ is a Gaussian $\mathcal{N}(\boldsymbol{w}_t \mid \boldsymbol{0}; \boldsymbol{\Sigma}^{-1})$. The regularization term of the MAP estimator is in the form $\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{w}_t$. Therefore, intuitively, $\boldsymbol{\Omega}$ functions as an estimator of the covariance structure, computed from all parameters $\boldsymbol{w}_t$ (or equivalently, $\boldsymbol{\theta}_t$), over all tasks.

Eq. (2) is computationally advantageous for it is a convex optimization. To solve it, we alternatively minimize over $\{\boldsymbol{w}_t\}$ and $\boldsymbol{\Omega}$ while holding the other fixed. When $\boldsymbol{\Omega}$ is fixed, each $\boldsymbol{w}_t$ can be identified as

$$\boldsymbol{w}_t^* = \arg\min \sum_n \ell(\boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{x}_n, y_{nt}) + \gamma \boldsymbol{w}_t^{\mathrm{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{w}_t \,. \quad (4)$$

With two simple variable substitutions, the optimization takes the standard form of ℓ_2-norm regularization:

$$\hat{\boldsymbol{w}}_t^* = \arg\min \sum_n \ell(\hat{\boldsymbol{w}}_t^{\mathrm{T}}\boldsymbol{z}_n, y_{nt}) + \gamma\|\hat{\boldsymbol{w}}_t\|_2^2, \quad (5)$$

$$\boldsymbol{z}_n \leftarrow \boldsymbol{\Omega}^{1/2}\boldsymbol{x}_n, \;\; \hat{\boldsymbol{w}}_t \leftarrow \boldsymbol{\Omega}^{-1/2}\boldsymbol{w}_t. \quad (6)$$

When the parameters $\{\boldsymbol{w}\}$ are fixed, the optimal $\boldsymbol{\Omega}$ that minimizes eq. (2) has a closed-form solution:

$$\boldsymbol{\Omega} = \frac{(\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} + \epsilon\boldsymbol{I})^{1/2}}{\mathbf{Trace}\left[(\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} + \epsilon\boldsymbol{I})^{1/2}\right]}. \quad (7)$$

The alternating minimization procedure monotonically decreases the objective function until the optimum solution is reached. Algorithm 1 lists the key steps. We set the hyperparameters γ and ϵ using a validation data set.

3.4. Extension to Kernel Classifiers

The feature learning framework can be extended to kernel-based nonlinear classifiers. We apply the kernel construction of [1]. Let $K(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$ denote the kernel function between two original feature vectors $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n'}$. The kernel induces a nonlinear feature mapping $\phi(\boldsymbol{x}_n) \in \mathcal{H} \subset \mathbb{R}^{\mathsf{H}}$. We perform feature learning in this new space $\mathcal{H}$.

To "kernelize", note that the optimal parameter $\boldsymbol{W} \in \mathbb{R}^{\mathsf{H}\times\mathsf{T}}$ for the models is a linear combination of (training) feature vectors. This can be understood intuitively by observing that eq. (5) is the standard formulation of an SVM; therefore the solution $\{\hat{\boldsymbol{w}}_t^*\}$ is a linear combination of feature vectors. The same statement is also true for $\boldsymbol{W}$, as the two are linearly related as in eq. (6).

It is computationally convenient to express $\boldsymbol{W}$ using the basis $\boldsymbol{V}$ of the feature space $\mathcal{H}$: $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{\alpha}$ (we have adopted a slightly different notation from [1] by adhering to the standard nomenclature in SVMs). We assume the number of basis vectors in $\boldsymbol{V}$ is $\mathsf{B} < \mathsf{N}$ where N is the total number of feature vectors. The matrix $\boldsymbol{\alpha}$ is the linear combination matrix, each column for a task. The basis $\boldsymbol{V}$ can be computed from the kernel matrix formed from training feature vectors, for instance, through eigendecomposition or Gram-Schmidt (G-S) orthogonalization. We use the latter technique for its slightly lower computational overhead. Concretely, we randomly choose B training feature vectors $\mathcal{S}$ and express the basis in the linear combination of those features, $\boldsymbol{V} = \boldsymbol{\Phi}_{\mathcal{S}}\boldsymbol{B}$, where the matrix $\boldsymbol{\Phi}_{\mathcal{S}}$'s columns are the nonlinear features computed from the chosen training instances. The matrix $\boldsymbol{B} \in \mathbb{R}^{\mathsf{B}\times\mathsf{B}}$ stores the linear combination coefficients, computed by the G-S process.

The parameter $\boldsymbol{W}$ is also linearly represented, as $\boldsymbol{W} = \boldsymbol{\Phi}_{\mathcal{S}}\boldsymbol{B}\boldsymbol{\alpha}$. Analogous to eq. (2), the optimal $\boldsymbol{\alpha}$ is then:

$$\begin{aligned}\boldsymbol{\alpha},\, \boldsymbol{\Omega}^* = \arg\min \sum_t \sum_n \ell(\boldsymbol{\alpha}_t^{\mathrm{T}}\boldsymbol{z}_n, y_{nt}) \\ + \gamma \sum_t \boldsymbol{\alpha}_t^{\mathrm{T}}\boldsymbol{\Omega}^{-1}\boldsymbol{\alpha}_t + \gamma\epsilon\,\mathsf{Trace}(\boldsymbol{\Omega}^{-1}).\end{aligned} \quad (8)$$

where $\boldsymbol{\alpha}_t$ is the t-th column of $\boldsymbol{\alpha}$. $\boldsymbol{z}_n = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{k}_{\mathcal{S}}(\boldsymbol{x}_n)$ is the transformed data, resulting from the linear discriminant in the feature space $\mathcal{H}$,

$$\boldsymbol{w}_t^{\mathrm{T}}\phi(\boldsymbol{x}_n) = (\boldsymbol{B}\boldsymbol{\alpha}_t)^{\mathrm{T}}\boldsymbol{\Phi}_{\mathcal{S}}^{\mathrm{T}}\phi(\boldsymbol{x}_n) = \boldsymbol{\alpha}_t^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{k}_{\mathcal{S}}(\boldsymbol{x}_n), \quad (9)$$

where the vector $\boldsymbol{k}_{\mathcal{S}}(\boldsymbol{x}_n) \in \mathbb{R}^{\mathsf{B}}$ consists of the elements of the kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_b) = \phi(\boldsymbol{x}_b)^{\mathrm{T}}\phi(\boldsymbol{x}_n)$.

The optimization problem eq. (8) is now readily solvable using techniques described previously. Key steps are given in Algorithm 2.

3.5. Other Extensions

We propose several additional extensions, addressing issues that naturally arise in our setting.

Modeling disparate sets of labels As opposed to [1], our main task and auxiliary tasks have different sets of labels and different types of loss functions. Thus, we use two regularizers, one for each group. In the linear classifier case,

Input: training data $(\boldsymbol{x}_n, \{y_{nt}\}), \epsilon, \gamma$, and B
Output: $\boldsymbol{\alpha}^*, \boldsymbol{\Omega}^*, \boldsymbol{B}$
1: Formulate kernel matrix $\boldsymbol{K}$
2: Compute the basis $\boldsymbol{B}, \mathcal{S} \leftarrow$ GRAM-SCHMIDT$(\boldsymbol{K}, \mathsf{B})$
3: Transform data according to eq. (9) and $\mathcal{S}$
4: $\boldsymbol{\alpha}^*, \boldsymbol{\Omega}^* \leftarrow$ ALGORITHM 1$((\boldsymbol{z}_n, \{y_{nt}\}), \epsilon, \gamma)$

Algorithm 2: Learning Features for a Kernel Classifier

our optimization takes the form,

$$\boldsymbol{W}^*,\ \boldsymbol{\Omega}^* = \arg\min \sum_t \sum_n \ell(\boldsymbol{w}_t^{\mathsf{T}} \boldsymbol{x}_n, y_{nt}) + \epsilon\, \mathsf{Trace}(\boldsymbol{\Omega}^{-1}) + \sum_{t=1}^{\mathsf{M}} \gamma_M \boldsymbol{w}_t^{\mathsf{T}} \boldsymbol{\Omega}^{-1} \boldsymbol{w}_t + \sum_{t=\mathsf{M}+1}^{\mathsf{T}} \gamma_A \boldsymbol{w}_t^{\mathsf{T}} \boldsymbol{\Omega}^{-1} \boldsymbol{w}_t \tag{10}$$

where γ_M is used for the main task and γ_A for auxiliary tasks. When γ_A is set to zero, the optimization learns shared features from parameters for all object classes, without attributes. We term this setup as "Sharing-Obj". When γ_M is constrained to be the same as γ_A, we recover eq (2).

Handling high-dimensional features The alternating minimization algorithm described in Section 3.3 depends on re-estimating $\boldsymbol{\Omega}$ and computing its square root $\boldsymbol{\Omega}^{1/2}$ with eq. (3) and eq. (6). For the high-dimensional features used in our setting, directly computing these quantities is costly. We exploit the low-rank property of $\boldsymbol{\Omega}$ to circumvent this challenge. Note that the matrix $\boldsymbol{W}$ has T columns and $\mathsf{D} \gg \mathsf{T}$ rows. Thus, $\boldsymbol{W}$ can be factorized with "thin" singular value decomposition: $\boldsymbol{W} = \boldsymbol{L}\boldsymbol{S}\boldsymbol{R}^{\mathsf{T}}$, where $\boldsymbol{L} \in \mathbb{R}^{\mathsf{D}\times\mathsf{T}}$ and $\boldsymbol{R} \in \mathbb{R}^{\mathsf{T}\times\mathsf{T}}$ are $\boldsymbol{W}$'s (partial) left and right eigenvectors. The diagonal matrix $\boldsymbol{S} \in \mathbb{R}^{\mathsf{T}\times\mathsf{T}}$ is composed of $\boldsymbol{W}$'s singular values $\{\sigma_i(\boldsymbol{W})\}_{i=1}^{\mathsf{T}}$. With some algebraic manipulation, we identify the eigenvalues of $\boldsymbol{\Omega}$:

$$\lambda_i(\boldsymbol{W}) = \left(\sqrt{\sigma_i^2(\boldsymbol{W}) + \epsilon}\right)/\rho,\ \ \lambda(\epsilon) = \sqrt{\epsilon}/\rho \tag{11}$$

$$\rho = \sum_{i=1}^{\mathsf{T}} \sqrt{\sigma^2(\boldsymbol{W}) + \epsilon} + \sqrt{\epsilon}\,[\mathsf{D} - \mathsf{T}]\,. \tag{12}$$

The eigenvectors in $\boldsymbol{L}$ and the subspace orthogonal to them span precisely $\boldsymbol{\Omega}$'s column space. This yields,

$$\boldsymbol{\Omega} = \boldsymbol{L}\,\mathsf{Diag}\left(\{\lambda_i(\boldsymbol{W})\}_{i=1}^{\mathsf{T}}\right)\boldsymbol{L}^{\mathsf{T}} + \lambda(\epsilon)(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}). \tag{13}$$

The matrix $\boldsymbol{\Omega}^{1/2}$ can be formulated similarly, replacing $\lambda_i(\boldsymbol{W})$ and $\lambda(\epsilon)$ with their square roots.

Choosing the kernel basis For the kernelized version, one needs to choose B basis vectors to expand the kernel feature space, as described in Section 3.4. We use two simple heuristics. We choose B large enough such that the performance of using the B basis vectors for *individual* task learning is close to the performance of our baseline system's. The individual task learning is set up as a linear classifier using the *transformed* feature vectors eq. (9), while the baseline system's are kernel-based nonlinear classifiers using the original features.

For the Gram-Schmidt process, we choose B/M feature vectors randomly from each of M classes. This gives balanced coverage of different features, and in practice works better than purely randomly selecting without taking object class into consideration.

4. Results

We validate our approach against relevant baselines, and report results on object categorization, the main target task.

Datasets We consider two datasets: the *Animals with Attributes* dataset (AWA) [13], and the *Outdoor Scene Recognition* dataset (OSR) [16]. AWA contains 30,475 images, 50 animal classes, and 85 attributes.[2] Each image is labeled by the animal and attributes present. OSR has 2,688 images, 8 scene classes, and 6 attributes as given in [16]: natural, open, perspective, size, diagonal plane, and depth. We asked another vision researcher to make the assignment from attributes to scenes. We apply random train-test splits, ensuring balance among object classes. Throughout, we use "object" to refer to an animal or scene.

Baselines We consider two baselines: 1) a traditional multi-class object recognition approach using an SVM with a χ^2 kernel computed on image features, which we refer to as **No sharing-Object**, or **NSO**, and 2) an approach that treats attributes as intermediate features, which we call **No sharing-Attribute**, or **NSA**. For NSA, we train SVMs on image features to predict attribute labels, and then treat their outputs as features to a multi-class logistic regression classifier. This baseline follows the basic direct attribute prediction (DAP) approach defined in [13]. We use LIBSVM.

Image features All methods use the same original image features. For AWA, we use the six (SIFT, rgSIFT, PHOG, SURF, LSS, RGB) provided with the dataset, each up to 2688-D. For OSR we generate 512-D Gist and 45-D LAB color histograms. We average the kernels computed over multiple feature types. Note that both datasets permit global descriptors, since there is one primary object of interest per image. To test with multi-object images, one would apply a window-based detector.

4.1. Impact of Sharing Features

First we evaluate the object recognition accuracy of our approach and the baselines. Our approach gets the same training images for both the attribute and object tasks. We form four training splits of increasing size (10% to 60%),

[2] For all methods, we use the 59 attributes exceeding 70% accuracy as reported in [13], since some are unpredictable from the given features.

	50-class Animals Dataset				8-class Scenes Dataset			
Method / % train data	10%	20%	40%	60%	10%	20%	40%	60%
No sharing-Obj. (NSO)	31.96	38.12	44.08	48.03	76.76	79.75	83.03	83.74
No sharing-Attr. (NSA)	31.03	35.61	41.12	43.59	57.77	58.98	60.50	60.78
Sharing-Obj. (Ours)	37.08	41.01	46.46	49.15	78.76	81.49	85.05	86.06
Sharing+Attr. (Ours)	36.73	42.60	47.70	50.94	78.09	81.62	85.89	87.01
% gain over NSO	**14.92%**	**11.75%**	**8.21%**	**6.06%**	**1.73%**	**2.34%**	**3.44%**	**3.90%**
% gain over NSA	**18.37%**	**19.63%**	**16.00%**	**16.86%**	**35.17%**	**38.39%**	**41.97%**	**43.16%**

Table 1. Accuracy on both datasets, as a function of training set size. Learning shared representations with our approach significantly improves generalization on the novel test set, and can be most pronounced when labeled training data is limited.

and reserve the rest for validation and testing (20% each). We demonstrate two variants of our approach: Sharing-Obj, where we learn a common representation for all object classes simultaneously, corresponding to $\gamma_A = 0$ in eq. (10), and Sharing+Attributes, where we learn the space for all objects and attributes, corresponding to $\gamma_A = \gamma_M$.

Table 1 shows the results. Our feature sharing approach offers significant improvements over both 'No sharing' baselines, and we obtain the best results when jointly learning with both the objects and attributes. The last two rows summarize gains of Sharing+Attributes over the baselines. Our improvements over the NSO baseline are perhaps most informative, since the general approach taken by NSO (multiple image features, kernel combination, non-linear SVM) is typical in state-of-the-art image recognition techniques.

While the margin between our Sharing-Object and Sharing+Attributes variants is smaller than the margin between not sharing at all versus sharing, the impact of attributes is clear and consistent. A one-tailed paired t-test on the 60% training split confirms that the accuracy gain with attribute tasks is statistically significant (for $\alpha = 5\%$ on AWA and $\alpha = 1\%$ on OSR). By separately tuning the γ_M and γ_A regularization weights, we expect even better performance; we simply let them be equal to save computation time.

Interestingly, on the larger AWA set, our gains are largest for smaller labeled data pools, supporting our claim that attribute feature sharing can have a beneficial regularization effect for object learning. This is an encouraging result, particularly since obtaining attribute labels on object-labeled data has minimal additional overhead for many attribute types, as discussed previously. Figure 2 visualizes the shared features over iterations, showing how we converge to a common sparse set.

Figure 3 breaks out the prediction accuracy per object category on both datasets. We improve accuracy for 33 of the 50 AWA classes, and yield correct predictions for some classes the baselines miss completely (e.g., `beaver`, `rat`). On OSR, the absolute accuracy is higher overall, due to the smaller multi-way decision. However, NSA suffers due to the insufficiency of the attribute vocabulary; it happens that the scenes `tallbuilding` and `insidecity` have exactly the same attribute definitions. In contrast, our approach accounts for attributes while still learning features sufficient to make the distinction.

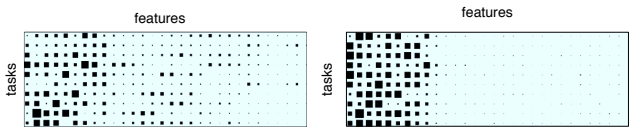


Figure 2. Hinton diagram of the matrix Θ in the initial and last iterations of Alg. 2. Each square is a matrix entry, and area reflects the entry's magnitude. For clarity only a partial matrix is shown, for the first 30 features (horizontally) and the first 10 object classes (vertically). The matrix at the last iteration is much sparser.

One might ask whether some *arbitrary* grouping of object classes into tasks might also have similar benefits. That is, are our gains due to the attributes' meaning, or could it be a sort of "error-correcting code" effect? To analyze this, we test a baseline where each object's attribute labels are randomly reassigned to other attributes, and then apply our method (for five such random assignments on the 60% training split). On OSR, we find this baseline offers no improvement over Sharing-Object (decreasing accuracy by 0.06). On AWA, the baseline improves over Sharing-Object (by 0.97 on average), but by less than sharing with real attributes (which increases accuracy by 1.79). This indicates the attribute semantics are indeed a factor in our method's success.[3]

In the remaining text, we report our results using Sharing+Attributes, and we focus on the AWA data, since it is $11x$ larger and has a richer set of attributes.

4.2. Impact of Disjoint Training Images

Our model is flexible to the source of object- and attribute-labeled data, and we can train the tasks on disjoint sets of images. This is relevant when one has a large set of existing attribute-labeled data, and wants to use it to regularize the training process for a new set of object models.

Thus, we next examine the impact of which images are used as the auxiliary attribute tasks to train the object clas-

[3]Looking closely at the AWA data, we see that the baseline's small gain made with randomly assigned attribute labels may be misleading. Because the classes are fine-grained, any random assignment of labels can overlap with meaningful attributes; the 85 attribute labels in AWA are certainly not exhaustive for the 50 animals.

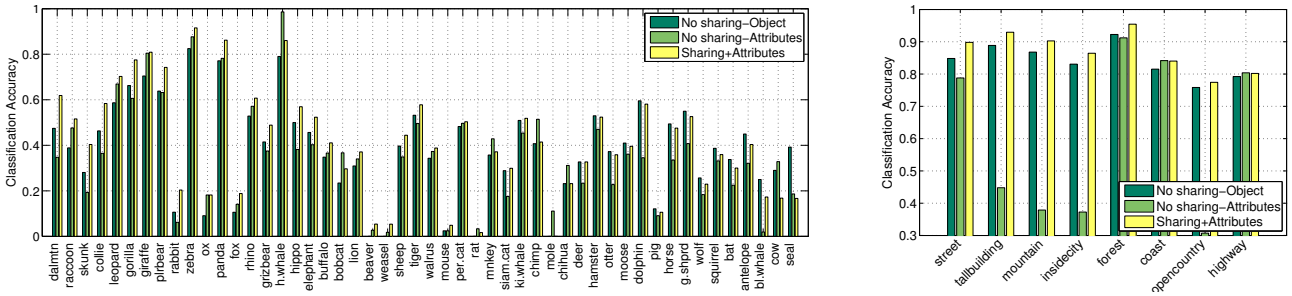


Figure 3. Accuracy on AWA (left) and OSR (right) classes. Our approach outperforms methods that learn objects (No sharing-Object) or attributes (No sharing-Attributes) independently.

	Image source for attributes		
Method	Same	Disjoint	All
No sharing-Object (NSO)	72.99	72.99	72.99
Sharing+Attribute	76.40	76.32	77.05
% gain	4.67%	4.56%	5.56%

Table 2. Object prediction accuracy as a function of which image pool is used for the attribute tasks, on the 10-class AWA subset.

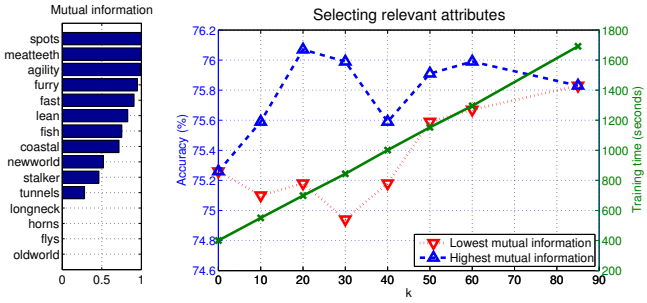


Figure 4. Left: Mutual information scores. Right: Object classification accuracy and training time as a function of the number of attribute tasks included.

sifiers. We select 10 classes (the same as [13]) to train the object classifiers, and test three variations for learning the attributes: 1) the *same* images used for the objects, 2) a *disjoint* set of images containing object classes outside of the 10, and 3) all images, the union of the previous two.

Table 2 shows the results. Interestingly, we see that our method performs similarly whether the attribute data overlaps or not (see first two columns). This suggests that the value of the attributes is not simply having deeper/stronger labels on the very same training examples; rather, it is the fact that we identify a common space where both types of labels are well predicted. The table also indicates that more attribute-labeled images is helpful (cf. last column).

4.3. Selecting Relevant Attributes

Having tested the impact of *which images* have attribute labels, next we consider the impact of *which attribute classes* are leveraged as auxiliary tasks. Presumably, not all attributes will benefit feature sharing, and—as usual in multi-task learning—some may be detrimental. Even if all attributes were relevant to some degree, we may want to be selective to save training costs.

Thus, we explore a simple form of automatic attribute selection in which we rank all attributes by their mutual information (MI) with the 10 animals[4]. Figure 4 (left) displays the computed MI, from the most informative attributes (e.g., "spots", which chimps and pigs lack, but leopards and pandas have) to the least (e.g., none of the 10 animals "fly").

Figure 4 (right) shows the impact of using the MI scores to select attributes for sharing. Both dotted curves denote our method, but one uses the k *most* informative attributes, and the other uses the k *least* informative attributes.[5] The most interesting cases are for lower values of k. (For higher values of k, the "most" and "least" sets overlap more, and they're identical at $k = 85$.)

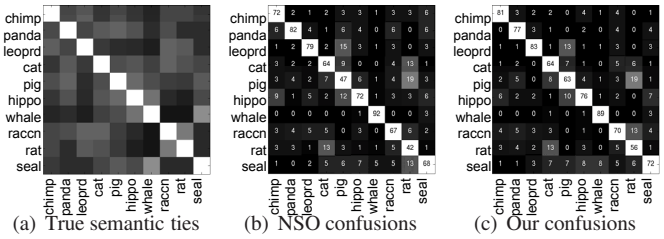
The results show that using the 20 attributes with the highest MI yields the best accuracy, while using the lowest 20 is slightly worse than using none whatsoever. Further, we see that more attribute classes do not necessarily always help. These findings plus the fact that training time increases linearly with k (see solid green line, right axis), suggest it is practical to choose intelligently. This result also shows the potential for performing task selection outside of the feature sharing learning procedure.

4.4. Semantically Meaningful Predictions

Finally, we analyze to what extent the semantics we introduce by jointly training objects and attributes are manifest in our method's predictions. Figure 5 compares the confusion matrices for our method (c) and NSO (b). To judge the "reasonableness" of their errors, in (a) we depict the true relationships between all pairs of the 10 objects. To obtain this matrix, we use human subjects' ratings collected in [18] about the relative strength of association between the 85 attributes and 50 objects in AWA. For each object, we create a vector of its 85 property "strengths", and then compute the pairwise χ^2 kernel values between all such vectors. Brighter boxes indicate greater true association in (a), and higher confusion in (b,c). Thus, if a method captures semantics well, its confusion matrix will look more like (a).

[4]chimp, panda, leopard, persian cat, hippo, whale, raccoon, rat, seal

[5]Note, we simply fix the γ and ϵ parameters for all cases, in order to see the effect of the attribute selection in isolation.



(a) True semantic ties (b) NSO confusions (c) Our confusions

Figure 5. Confusions made by the baseline (b) and our method (c) relative to human-given object relationships (a).

First, we notice that our method boosts accuracy for most classes, raising the mean diagonal from 66.9% to 68.9%. Second, we see that the pairs for which our method most reduces confusions (e.g., `pig` vs `rat`) are more distinctive semantically. On the flip side, some closely related pairs become confused by our method (e.g., `raccoon` vs `cat`). Figure 6 shows example animal category and attribute predictions, compared alongside NSO and NSA.

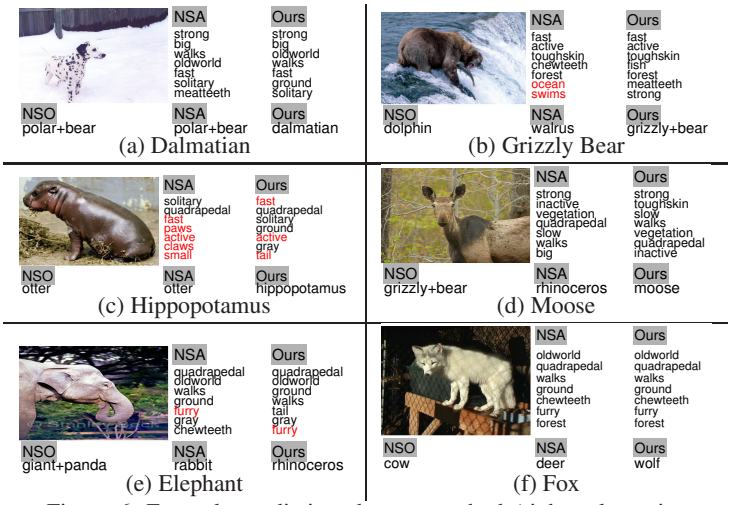


Figure 6. Example predictions by our method (right column in each), NSA (middle columns), and NSO (object prediction under each image). Attributes are those with 7 highest positive decision values, by ours or NSA (red attributes incorrect). (a)-(d) illustrate good results, and (e)-(f) show failure cases that highlight our method's tendency to make semantically meaningful errors.

5. Conclusions

This work shows that by learning a common feature space suitable to either attribute or object tasks, we can obtain noticeably stronger object recognition results. We demonstrated the proposed method's improved generalization accuracy and its potential to make more predictable errors in terms of human-defined semantics. In future work, we plan to continue our exploration of automatic task selection, and consider how to optimally combine the regularization per object and attribute.

Acknowledgements: This research is supported in part by NSF IIS-1065390, ONR ATL 141110105, and CSSG.

References

[1] M. P. A. Argyriou, T. Evgeniou. Convex Multi-task Feature Learning. *Machine Learning*, 73(3):243–272, 2008.
[2] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing. Training Hierarchical Feed-Forward Visual Recognition Models using Transfer Learning from Pseudo-Tasks. In *ECCV*, 2008.
[3] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering Shared Structures in Multiclass Classification. In *ICML*, 2007.
[4] R. Ando and T. Zhang. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *JMLR*, 6, 2005.
[5] E. Bart and S. Ullman. Cross-Generalization: Learning Novel Classes from a Single Example by Feature Replacement. In *CVPR*, 2005.
[6] T. Berg, A. Berg, and J. Shih. Automatic Attribute Discovery and Characterization from Noisy Web Data. In *ECCV*, 2010.
[7] R. Caruana. Multitask Learning. *Machine Learning*, 1997.
[8] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs. *JMLR*, 2001.
[9] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing Objects by their Attributes. In *CVPR*, 2009.
[10] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories. In *ICCV*, 2003.
[11] V. Ferrari and A. Zisserman. Learning Visual Attributes. In *NIPS*, 2007.
[12] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Attribute and Simile Classifiers for Face Verification. In *ICCV*, 2009.
[13] C. Lampert, H. Nickisch, and S. Harmeling. Learning to Detect Unseen Object Classes by Between-Class Attribute Transfer. In *CVPR*, 2009.
[14] N. Loeff and A. Farhadi. Scene Discovery by Matrix Factorization. In *ECCV*, 2008.
[15] G. Obozinksi, B. Taskar, and M. Jordan. Joint Covariate Selection and Joint Subspace Selection for Multiple Classification Problems. *Stat Comput*, 2009.
[16] A. Oliva and A. Torralba. Modeling the Shape of the Scene: a Holistic Representation of the Spatial Envelope. *IJCV*, 42(3), 2001.
[17] B. Olshausen and D. Field. Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, 381:607–609, 1996.
[18] D. Osherson, J. Stern, O. Wilkie, M. Stob, and E. Smith. Default Probability. *Cognitive Science*, 15(2), 1991.
[19] A. Quattoni, M. Collins, and T. Darrell. Learning Visual Representations Using Images with Captions. In *CVPR*, 2007.
[20] A. Quattoni, M. Collins, and T. Darrell. Transfer Learning for Image Classification with Sparse Prototype Representations. In *CVPR*, 2008.
[21] M. Rohrbach, M. Stark, G. Szrvas, I. Gurevych, and B. Schiele. What Helps Where and Why? Semantic Relatedness for Knowledge Transfer. In *CVPR*, 2010.
[22] M. Stark, M. Goesele, and B. Schiele. A Shape-based Object Class Model for Knowledge Transfer. In *ICCV*, 2009.
[23] A. Torralba and K. Murphy. Sharing Visual Features for Multiclass and Multiview Object Detection. *PAMI*, 29(5), 2007.
[24] D. Vaquero, R. Feris, L. Brown, and A. Hampapur. Attribute-based People Search in Surveillance Environments. In *WACV*, 2009.
[25] G. Wang and D. Forsyth. Joint Learning of Visual Attributes Object Classes and Visual Salience. In *ICCV*, 2009.
[26] J. Wang, K. Markert, and M. Everingham. Learning Models for Object Recognition from Natural Language Descriptions. In *BMVC*, 2009.
[27] Y. Wang and G. Mori. A Discriminative Latent Model of Object Classes and Attributes. In *ECCV*, 2010.
[28] M. Yuan and Y. Lin. Model Selection and Estimation in Regression with Grouped Variables. *Jrnl of the Roy Stat Soc*, 68(1):49–67, 2007.
[29] X.-T. Yuan and S. Yan. Visual Classification with Multi-Task Joint Sparse Representation. In *CVPR*, 2010.