Semi-supervised Learning in Gigantic Image Collections

Rob Fergus,
Courant Institute, NYU,
715 Broadway,
New York, NY 10003
fergus@cs.nyu.edu

Yair Weiss,
School of Computer Science,
Hebrew University,
91904, Jerusalem, Israel
yweiss@huji.ac.il

Antonio Torralba,
CSAIL, EECS, MIT,
32 Vassar St.,
Cambridge, MA 02139
torralba@csail.mit.edu

Abstract

With the advent of the Internet it is now possible to collect hundreds of millions of images. These images come with varying degrees of label information. "Clean labels" can be manually obtained on a small fraction, "noisy labels" may be extracted automatically from surrounding text, while for most images there are no labels at all. Semi-supervised learning is a principled framework for combining these different label sources. However, it scales polynomially with the number of images, making it impractical for use on gigantic collections with hundreds of millions of images and thousands of classes.

In this paper we show how to utilize recent results in machine learning to obtain highly efficient approximations for semi-supervised learning. Specifically, we use the convergence of the eigenvectors of the normalized graph Laplacian to eigenfunctions of weighted Laplace-Beltrami operators. We combine this with a label sharing framework obtained from Wordnet to propagate label information to classes lacking manual annotations. Our algorithm enables us to apply semi-supervised learning to a database of 80 million images with 74 thousand classes.

1. Introduction

Gigantic quantities of visual imagery are present on the web and in off-line databases. Effective techniques for searching and labeling this ocean of images and video must address two conflicting problems: (i) the techniques to understand the visual content of an image and (ii) the ability to scale to millions of billions of images or video frames. Both aspects have received significant attention from researchers, the former being addressed by recent work on object and scene recognition, while the latter is the focus of the content-based image retrieval community (CBIR) [6].

A key issue pertaining to both aspects of the problem is the diversity of label information accompanying real world image data. A variety of collaborative and online annotation efforts have attempted to build large collections of human labeled images, ranging from simple image classifications, to bounding-boxes and precise pixel-level segmentation [17, 22, 25]. While impressive, these manual efforts have no hope of scaling to the many billions of images on the Internet. However, even though most images on the web lack human annotation, they often have some kind of noisy

Figure 1. Two examples of images from Internet search engines being re-ranked by our approach, according to the probability of belonging to the categories "turboprop" and "pony" respectively. The images are part of a dataset of 80 million [21], for which we have labels on only 64,185, spread over 386 classes out of 74,569 in the dataset. For the two classes used in this example, no labels exist and our algorithm operates by diffusing the labels from other classes through the entire 80 million images using both the density structure of the data and also the semantic relationships between categories. Our approach is able to do this efficiently, taking around 1 minute ($\sim$0.75μs/image) on a large PC.

label gleaned from nearby text or from the image filename and often this gives a strong cue about the content of the image. Finally, there are images where we have no information beyond the pixels themselves. Semi-supervised learning (SSL) methods are designed to handle this spectrum of label information [26, 27]. They rely on the density structure of the data itself to propagate known labels to areas lacking annotations, and provide a natural way to incorporate labeling uncertainty. However, to model the density of the data, each point must measure its proximity to every other. This requires polynomial time – prohibitive for large-scale problems.

In this paper, we introduce a semi-supervised learning scheme that is linear in the number of images, enabling us to tackle very large scale problems. Building on recent results in spectral graph theory, we efficiently construct accurate numerical approximations to the eigenvectors of the normalized graph Laplacian. Using these approximations, we can easily propagate labels through huge collections of images.

A second contribution is the integration of a method for sharing labels between classes into the SSL scheme. Many existing approaches to recognition and learning treat classes independently from one another. In large-scale problems, there will be many thousands of label classes, many of which will be similar to one another. In this regime, being able to transfer labels between classes is vital, since the expected number of labels per class will be small. By leveraging labels from nearby classes, we can boost the effective amount of label information available.

Bringing these two contributions together, we are able to propagate a limited set of manually provided labels through very large collections of images from the Internet, correcting the noisy labels obtained from non-visual sources such as surrounding text.

1.1. Related Work

Cleaning up Internet image data has been explored by several authors: Berg *et al.* [3], Fergus *et al.* [8], Li *et al.* [14], Vijayanarasimhan *et al.* [23], amongst others. Unlike our approach, these methods operate independently on each class and would be problematic to scale to millions or billions of images. A related group of techniques use active labeling, where the user is in the loop, e.g. [11].

Semi-supervised learning is a rapidly growing sub-field of machine learning, dealing with datasets that have a large number of unlabeled points and a much smaller number of labeled points (see [4] for a recent overview). The most popular approaches are based on the graph Laplacian (e.g. [26, 27] and there has been much theoretical work devoted to the asymptotics of these Laplacians [2, 5, 15].

The learning and use of a class hierarchy has been recently explored by a number of authors [10, 20, 28]. Unlike these approaches, we use a pre-determined hierarchy for our class-sharing. Ideally we would also learn the hierarchy, but it is challenging to do so when operating on gigantic image collections.

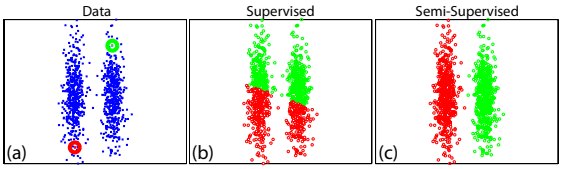

Figure 2. Comparison of supervised and semi-supervised learning on toy data. Semi-supervised learning seeks functions that are smooth with respect to the input density.

2. Semi-supervised Learning

We start by introducing semi-supervised learning in a graph setting and then describe an approximation that reduces the learning time from polynomial to linear in the number of images. Fig. 2 illustrates the semi supervised learning problem. Following the notations of Zhu *et al.* [27], we are given a labeled dataset of input-output pairs $(X_l, Y_l) = \{(x_1, y_1), ..., (x_l, y_l)\}$ and an unlabeled dataset $X_u = \{x_{l+1}, ..., x_n\}$. Thus in Fig. 2(a) we are given two labeled points and 500 unlabeled points. Fig. 2(b) shows the output of a nearest neighbor classifier on the unlabeled points. The purely supervised solution ignores the apparent clustering suggested by the data.

In order to use the unlabeled data, we form a graph $G = (V, E)$ where the vertices V are the datapoints $x_1, ..., x_n$, and the edges E are represented by an $n \times n$ matrix W. Entry W_{ij} is the edge weight between nodes i, j and a common practice is to set $W_{ij} = \exp(-\|x_i - x_j\|/2\epsilon^2)$. Let D be a diagonal matrix whose diagonal elements are given by $D_{ii} = \sum_j W_{ij}$, the combinatorial graph Laplacian is defined as $L = D - W$, which is also called the unnormalized Laplacian. The normalized graph Laplacian is defined as $\mathcal{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$.

In graph-based semi-supervised learning, the graph Laplacian L is used to define a smoothness operator that takes into account the unlabeled data. Intuitively, the main idea of semi-supervised learning is to find functions f which agree with the labeled data but are also *smooth* with respect to the graph. The smoothness is measured by the graph Laplacian:

$$f^T L f = \frac{1}{2} \sum_{i,j} W_{ij} \left(f(i) - f(j)\right)^2$$

Of course simply minimizing smoothness can be achieved by the trivial solution $f = 1$, but in semi-supervised learning, we minimize a combination of the smoothness and the training loss. For squared error training loss, this is simply:

$$\begin{aligned} J(f) &= f^T L F + \sum_{i=1}^{l} \lambda (f(i) - y_i)^2 \\ &= f^T L f + (f-y)^T \Lambda (f-y) \end{aligned}$$

where Λ is a diagonal matrix whose diagonal elements are $\Lambda_{ii} = \lambda$ if i is a labeled point and $\Lambda_{ii} = 0$ for unlabeled points. The minimizer is of course a solution to $(L+\Lambda)f = \Lambda y$. Fig. 2(c) shows the semi-supervised solution. Although the supervised solution is actually smoother if one ignores the unlabeled data, the semi-supervised solution is much smoother when the unlabeled data is taken into account. In the semi-supervised solution, neighboring points in the graph have similar labels, while in the supervised solution, neighboring points in the graph can have very different labels.

Although the solution can be given in closed form for the squared error loss, note that it requires solving an $n \times n$ system of linear equations. For large n this poses serious problems with computation time and robustness. But as suggested in [4, 18, 27], the dimension of the problem can be reduced dramatically by only working with a small number of eigenvectors of the Laplacian (see also [13] for the same idea within the context of matting).

Let Φ_i, σ_i be the eigenvectors and eigenvalues of the graph Laplacian L. Note that the smoothness of an eigenvector Φ_i is simply $\Phi_i^T L \Phi_i = \sigma_i$ so that eigenvectors with smaller eigenvalues are smoother. Since any vector in R^n can be written $f = \sum_i \alpha_i \Phi_i$, the smoothness of a vector is simply $\sum_i \alpha_i^2 \sigma_i$ so that smooth vectors will be linear combinations of the eigenvectors with small eigenvalues.

Fig. 3(top) shows the three *generalized* eigenvectors of the Laplacian (solutions to $L\phi = \sigma D\phi$) with smallest eigenvalue, for the toy data. As pointed out by Shi and Malik [19], eigenvectors of the unnormalized graph Laplacian have a tendency to cut out isolated points in the data (since there are very weak edges between an isolated point and its neighbors in the graph). We therefore use the generalized eigenvectors henceforth.

Since smooth vectors will be linear combinations of the eigenvectors with small eigenvalues, we can significantly reduce the dimension of f by requiring it to be of the form $f = U\alpha$ where U is a $k \times n$ matrix whose columns are the k eigenvectors with smallest eigenvalue. We now have:

$$J(\alpha) = \alpha^T \Sigma \alpha + (U\alpha - y)^T \Lambda (U\alpha - y)$$

The minimizing α is now a solution to the $k \times k$ system of equations:

$$(\Sigma + U^T \Lambda U)\alpha = U^T \Lambda y \qquad (1)$$

2.1. From Eigenvectors to Eigenfunctions

Given the eigenvectors of the graph Laplacian, we can now solve the semi-supervised problem in a reduced dimensional space. But to find the eigenvectors in the first place, we need to diagonalize a $n \times n$ matrix. How can we efficiently calculate the eigenvectors as the number of unlabeled points increases?

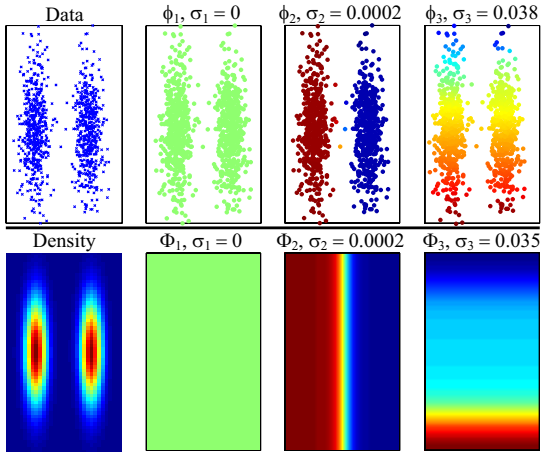

Figure 3. **Top:** The three generalized eigenvectors of the graph Laplacian, for the toy data. Note that the semi-supervised solution can be written as a linear combination of these eigenvectors (in this case, the second eigenvector is enough). Using generalized eigenvectors (or equivalently normalized Laplacians) increases robustness of the first eigenvectors, compared to using the un-normalized eigenvectors. **Bottom:** The 2D density of the toy data, and the associated smoothness eigenfunctions defined by that density. The plots use the Matlab jet colormap.

We follow [24] in assuming the data $x_i \in R^d$ are samples from a distribution $p(x)$ and analyzing the eigenfunctions of the smoothness operator defined by $p(x)$. Fig. 3(bottom) shows the density in two dimensions for the toy data. This density defines a weighted smoothness operator on any function $F(x)$ defined on R^d which we will denote by $L_p(F)$:

$$L_p(F) = \frac{1}{2} \int (F(x_1) - F(x_2))^2 W(x_1, x_2) p(x_1) p(x_2) dx_1 x_2$$

with $W(x_1, x_2) = \exp(-\|x_1 - x_2\|/2\epsilon^2)$.

Just as the graph Laplacian defined eigenvectors of increasing smoothness, the smoothness operator will define eigenfunctions of increasing smoothness. We define the first eigenfunction of $L_P(f)$ by a minimization problem:

$$\Phi_1 = \arg \min_{F: \int F^2(x) p(x) D(x) dx = 1} L_p(F)$$

where $D(x) = \int_{x_2} W(x, x_2) p(x2) dx2$. Note that the first eigenfunction will always be the trivial function $\Phi(x) = 1$ since it has maximal smoothness $L_P(1) = 0$. The second eigenfunction of $L_p(f)$ minimizes the same problem, with the additional constraint that $\int F(x)\Phi_1(x)D(x)p(x)dx = 0$. More generally, the kth eigenfunction minimizes $L_p(f)$ under additional constraints that $\int F(x)\Phi_l(x)p(x)D(x)dx = 0$ for all $l < k$. The eigenvalue of an eigenfunction Φ_k is simply its smoothness $\sigma_k = L_p(\Phi_k)$. Fig. 3(bottom) shows the first three

eigenfunctions corresponding to the density of the toy data. Similar to the eigenvectors of the graph Laplacian, the second eigenfunction reveals the natural clustering of the data. In order to avoid dividing by zero, we add a small constant to the density. Note that the eigenvalue of the eigenfunctions is similar to the eigenvalue of the discrete generalized eigenvector.

How are these eigenfunctions related to the eigenvectors of the Laplacian? It is easy to see that as $n \to \infty$, $\frac{1}{n^2} f^T L f = \frac{1}{2} \sum_{i,j} W_{ij} \left(f(i) - f(j)\right)^2$ will approach $L_p(F)$, and $\frac{1}{n} \sum_i f^2(i) D(i,i)$ will approach $\int F^2(x) D(x) p(x) dx$ so that the minimization problems that define the eigenvectors approach the problems that define the eigenfunctions as $n \to \infty$. Thus under suitable convergence conditions, the eigenfunctions can be seen as the limit of the eigenvectors as the number of points goes to infinity [1, 2, 5, 15].

For certain parametric probability functions (e.g. uniform, Gaussian) the eigenfunctions can be calculated analytically [15, 24]. Thus for these cases, there is a tremendous advantage in estimating $p(x)$ and calculating the eigenfunctions from $p(x)$ rather than attempting to estimate the eigenvectors directly. For example, consider a problem with 80 million datapoints sampled from a 32 dimensional Gaussian. Instead of diagonalizing an 80 million by 80 million matrix, we can simply estimate a 32×32 covariance matrix and get analytical eigenfunctions.

In low dimensions, we can calculate the eigenfunction numerically by discretizing the density. Let g be the eigenfunction values at a set of discrete points, then g satisfies:

$$P(\tilde{D} - \tilde{W})Pg = \sigma P \hat{D} g \tag{2}$$

where $\tilde{W}$ is the affinity between the discrete points, P is a diagonal matrix whose diagonal elements give the density at the discrete points, and $\tilde{D}$ is a diagonal matrix whose diagonal elements are the sum of $\tilde{W}$, and $\hat{D}$ is a diagonal matrix whose diagonal elements are the sum of $P\tilde{W}$. This method was used to calculate the eigenfunctions in Fig. 3(bottom).

Instead of assuming that $p(x)$ has a simple, parametric form, we will use a more modest assumption, that $p(x)$ has a product form. Specifically, we assume that if we rotate the data $s = Rx$ then $p(s) = p(s_1)p(s_2)\cdots p(s_d)$. This assumption allows us to calculate the eigenfunctions of L_p using only the marginal distributions $p(s_i)$.

Observation: Assume $p(s) = p(s_1)p(s_2)\cdots p(s_d)$. Let p_k be the marginal distribution of a single coordinate in s. Let $\Phi_i(s_k)$ be an eigenfunction of L_{p_k} with eigenvalue σ_i, then $\Phi_i(s) = \Phi_i(s_k)$ is also an eigenfunction of L_p with the same eigenvalue σ_i.

Proof: This follows from the observation in [15, 24] that for separable distributions, the eigenfunctions are also separable.

This observation motivates the following algorithm:

- Find a rotation of the data R, so that $s = Rx$ are as independent as possible.
- For each "independent" component s_k, use a histogram to approximate the density $p(s_k)$.
- Given the approximated density $p(s_k)$, solve numerically for eigenfunctions and eigenvalues of L_{p_k} using Eqn. 2. As discussed above, this can be done by solving a generalized eigenvalue problem for a $B \times B$ matrix, where B is the number of bins in the histogram.
- Order the eigenfunctions from all components by increasing eigenvalue.

This algorithm will recover eigenfunctions of L_p, which depend only on a single coordinate. As discussed in [24], products of these eigenfunctions for different coordinates are also eigenfunctions, but we will assume the semi-supervised solution is a linear combination of only the single-coordinate eigenfunctions.

By choosing the k eigenfunctions with smallest eigenvalue we now have k functions $\Phi_k(x)$ whose value is given at a set of discrete points for each coordinate. We then use linear interpolation to interpolate $\Phi(x)$ at each of the labeled points x_l. This allows us to solve Eqn. 1 in time that is *independent of the number of unlabeled points*.

Although this algorithm has a number of approximate steps, it should be noted that if the "independent" components are indeed independent, and if the semi-supervised solution is only a linear combination of the single-coordinate eigenfunctions, then this algorithm will exactly recover the semi-supervised solution as $n \to \infty$. Consider again a dataset of 80 million points in 32 dimensions and assume 100 bins per dimension. If the independent components $s = Rx$ are indeed independent, then this algorithm will exactly recover the semi-supervised solution by solving 32 100×100 generalized eigenvector problems and a single $k \times k$ least squares problem. In contrast, directly estimating the eigenvectors of the graph Laplacian will require diagonalizing an 80 million by 80 million matrix.

3. Semantic Sharing

Extending the algorithm to the multi-class scenario is straightforward. In a multi-class problem, the labels will be held in an $n \times c$ binary matrix Y, replacing y in Eqn. 1 (c being the number of classes). In the multi-class problem we solve for the $n \times c$ matrix $\mathcal{F} = [f_1, \ldots, f_c]$, using the eigenfunctions and Eqn. 1. This naive extension of the 1-class algorithm has an important limitation: the labels for each class are still propagated independently. Therefore, in order to achieve reasonable performance, we will need human annotations for a subset of the images for all the classes. Although providing training data is practical in situations with few classes, on large-scale problems with thousands of classes, a huge number of human annotations will

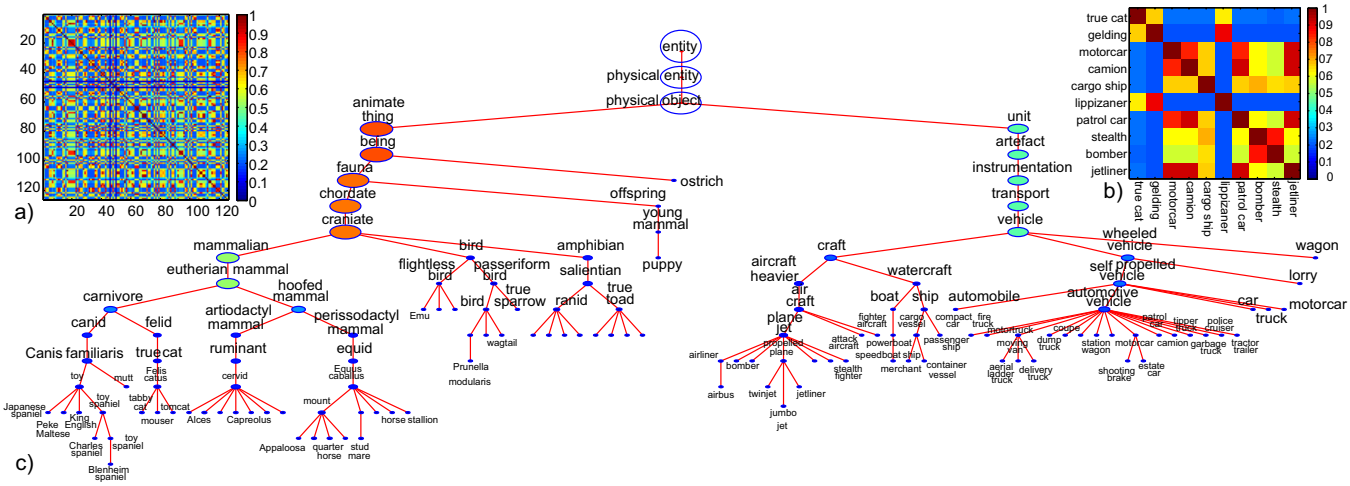

Figure 4. Wordnet sub-tree for 126 classes used in our experiments. The associated semantic affinity matrix A is shown in **(a)**, along with a closeup of 10 randomly chosen rows and columns in **(b)**.

be required. In this section we describe how we can modify the multi-class problem in order to transfer information across classes.

If we have many thousands of categories, the images in many of them will be visually similar and thus labeled examples in one might be expected to help in learning related classes. This requires some notion of the visual distance between classes and we use Wordnet [7] for this task. In doing so, we are approximating visual similarity with semantic similarity, as defined by Wordnet. In order to compute the semantic distance between two classes we use a tree defined by Wordnet (see Fig. 4(c)). The semantic distance S_{ij} between classes i and j (which are nodes in the tree) is defined as the number of nodes shared by their two parent branches, divided by the length of the longest of the two branches. We construct a sparse semantic affinity matrix $A = \exp(-\kappa(1-S))$, with $\kappa = 10$ for all the experiments in this paper. For the class "airbus", the nearest semantic classes are: "airliner" (0.49), "monoplane" (0.24), "dive bomber" (0.24), "twinjet" (0.24), "jumbo jet" (0.24), and "boat" (0.03). A visualization of A and a closeup are shown in Fig. 4(a) and (b). We share labels by replacing Y in our SSL scheme with YA, noting that this only alters the positive examples in each class.

4. Experiments

In this section we describe experiments to illustrate the performance and scalability of our approach. Most of the results will be reported on a portion of the Tiny Images database [21], in combination with the CIFAR-10 label set (beta version)[1]. This data is diverse and highly variable, having been collected directly from Internet search engines. The set of labels allows us to accurately measure the performance of our algorithm, while using data typical of the large-scale Internet settings for which our algorithm is designed. We start by describing the image features that we will use and briefly provide results on the Caltech 256 dataset [9], chosen to enable direct comparisons with other approaches.

4.1. Image Features

For the experiments in this paper we use global image descriptors to represent the entire image (there is no attempt to localize the objects within the images). Global descriptors have been shown to provide state of the art performance in challenging datasets such as Caltech 256 [9]. Here we illustrate the power of our approach on a standard dataset.

In Fig. 5(a) we show the performance of our SSL scheme on the Caltech 256 dataset. Our goal is to demonstrate the performance of our learning approach, compared to existing methods such as SVMs, once a suitable set of features have been computed for each image. Thus it is the relative performance between approaches that matters, rather than absolute performance which is largely dependent on the choice of features.

For Caltech 256, our chosen feature representation is a combination of a Gist descriptor and a Bag-of-Words descriptor, reduced to 1024 dimensions with PCA. The eigenfunction approach requires a matrix R that rotates the features so they become maximally independent. In our experiments, we found that the PCA components (which are of course uncorrelated but not independent) already had very low mutual information so we did not use an additional rotation. k=2048 eigenfunctions were used in the eigenfunction approach, while the SVM comparison uses an RBF kernel, tuned to give best performance. The eigenfunction approach consistently beats the SVM, and both approaches give a respectable absolute performance, compared to exist-

[1] Collected by Alex Krizhevsky and Vinod Nair.

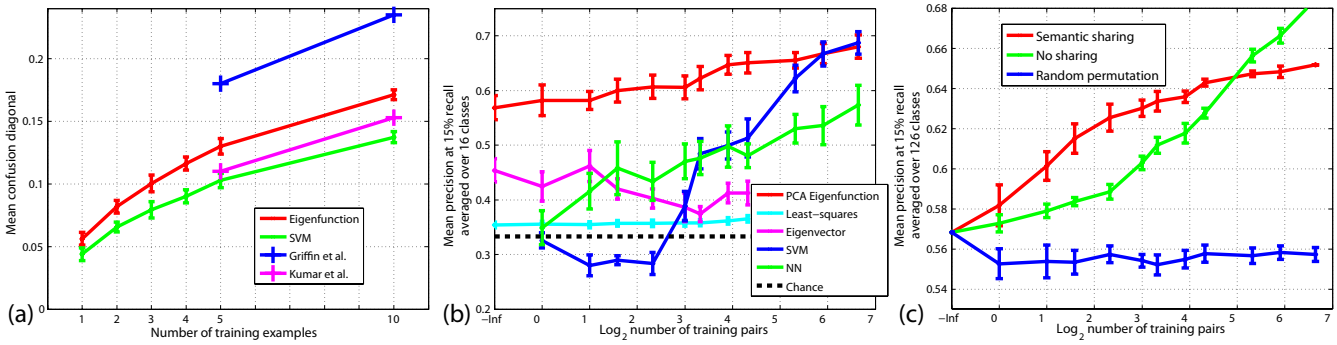

Figure 5. **(a)**: Evaluation of the eigenfunction approach on Caltech 256, compared to an SVM trained on the same descriptors and other published methods: Griffin et al. [9] and Kumar and Sminchisescu [12]. Tiny image data: **(b)**: Performance of different learning schemes as the number of training examples is increased. -Inf corresponds to the unsupervised case (0 examples). Tiny image data: **(c)**: Performance for different sharing strategies as the number of training examples is increased, using all 126 classes. Note the improvement in performance for small numbers of training examples when the Wordnet sharing matrix is used.

ing approaches.

The Caltech 256 dataset is not well adapted to our algorithm: there are relatively few categories (only 256) and the classes are very distinct, leaving very little room for transfer learning across classes. Therefore, in the rest of the paper we will show the power of our approach by deploying it on the Tiny Images dataset which is many orders of magnitude larger. For the rest of the experiments in the paper, we simplify our descriptor, using a single 384-dimensional global gist descriptor [16] to represent each image[2].

4.2. Experiments with 126 Categories and 63,000 Images

The CIFAR-10 labels comprise human image-level annotations (+ve/-ve) for 10 distinct classes in the Tiny Images dataset. Each class is made up of a number of keywords (the query term used to gather the images from the search engine), subordinate to the class as given by the Wordnet tree structure. We select the sub-set of 126 keywords from the CIFAR set which had at least 200 positive labels and 300 negative labels, giving a total of 63,000 images. The gist descriptor for each image is mapped down to a 64D space using PCA. For experiments using our eigenfunction approach, we computed a fixed set of k=256 eigenfunctions on the entire set of training data in the 64D space[3]. These keywords and their semantic relationship to one another are shown in Fig. 4. For each keyword, we randomly choose a fixed test-set of 100 positive and 200 negative examples, reflecting the typical signal-to-noise ratio found in images from Internet search engines. Note that we do not make use of any rank information. The training examples consist of +ve/-ve pairs drawn from the remaining pool of 100 positive/negative images for each keyword.

Each image has a noisy label (the keyword used to query the search engine). We assume that all true +ve instances of a class (e.g. "cat") are contained with those images having the noisy label cat[4]. Mixed in with these true +ve examples will be many non-cat images which also possess the noisy label cat. We use our scheme to propagate labels from training examples of cats, and semantically related classes, to the test examples of cat images. By assigning higher probability (values in f) to the genuine cat images, we are able to re-rank the images.

For approaches that require explicit formation of the affinity matrix, we calculated the distance between the 64D image descriptors using $\epsilon = 0.2$. All approaches used $\lambda = 1000$. To evaluate performance, we chose to measure the precision at a low recall rate of 15%, this being a sensible operating point when dealing with huge collections of data. Thus, chance level performance would be a precision of 33%. For experiments we averaged results over 10 different runs, each with different random train/test draws, and with different subsets of classes.

In our first set of experiments, shown in Fig. 5(b), we compare our eigenfunction approach using semantic sharing to a variety of alternative learning schemes. We use 16 different classes drawn randomly from the 126, and vary the number of training pairs used from 0 up to 100. Our eigenfunction approach performs well, even with relatively few training examples, or in an unsupervised setting. We also use two baseline approaches: (i) Nearest-Neighbor and (ii) RBF kernel SVM, with kernel width ϵ. Both these require at least one training pair per class. The SVM approach badly over-fits the data for small numbers of training examples, but performs well with sufficient data. We also test a range of SSL approaches. The exact least-squares approach ($\mathcal{F} = (L + \Lambda)^{-1}\Lambda Y A$) and eigenvector approach (Eqn. 1)

[2]The majority of the performance in the Caltech 256 experiments comes from the Gist descriptor. Furthermore, a bag-of-words representation is not practical to compute on a tiny image.

[3]Note that the construction of the eigenfunctions does not make use of any class label information.

[4]This means that we will not be able to recover a cat image with the noisy label dog, for example.

perform less well than our eigenfunction method, and also have the problem that the affinity matrix W becomes too big when a large amount of training data is used. Thus results cannot be computed for these cases. For example, for the 126 class 100 training pair case, a 63,000 by 63,000 matrix would need to be inverted. In Fig. 6(right) we explore how the number of eigenfunctions used in the PCA variant affects performance as the number of training examples is varied. The performance is fairly stable above 128 eigenfunctions (i.e. on average 2 per dimension), although some mild over-fitting seems to occur for small numbers of training examples when a very large number is used.

In Fig. 8 we show results on 10 keywords, whose performance ranges from poor to very good. The results were obtained using all 126 classes, 100 training pairs per class and semantic sharing turned on. Note that the human labeling at times is somewhat arbitrary and selective. For example, the Appaloosa horse images show many incorrect labels high in the cleaned-up results, yet the images are valid horses.

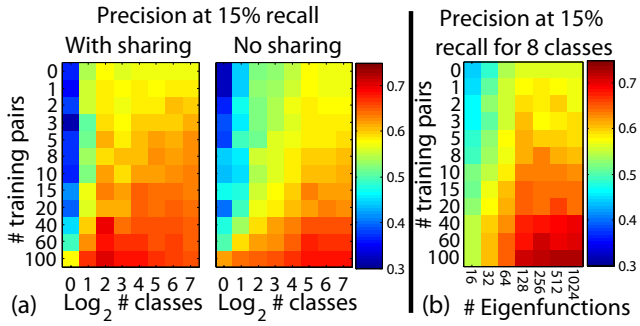

Figure 6. **(a)**: The effects of using the semantic sharing matrix are apparent, particularly for few training examples. However, in both cases, performance improves as the number of classes increases and the number of training examples increases. **(b)**: Performance as the number of eigenfunctions and training examples is varied for 8 classes.

4.2.1 Semantic Sharing

In Fig. 5(c) we explore the effects of semantic sharing with our eigenfunction approach. The application of the semantic affinity matrix can be seen to help performance when only a few training examples are present. However, when sufficient data is available, it does marginally impair performance. If the semantic matrix is randomly permuted (but with the diagonal fixed to be 1), then this hinders performance. Hence the semantic matrix must reflect the relationship between classes if it is to be effective. In Fig. 6, we perform a more systematic exploration of the effects of sharing using the eigenfunction approach. Both the number of classes and number of images are varied, and the performance recorded with and without the semantic affinity matrix. In both cases, the performance improves as more data is used (i.e. more classes) and also as more training examples are available. However, the affinity matrix assists performance for small numbers of training examples.

The sharing behavior can be used to effectively learn classes for which we have zero training examples. In Fig. 7, we explore what happens when we allocate 0 training images to one particular class (the left-out class) from the set of 126, while using 100 training pairs for the remaining 125 classes. When the sharing matrix is not used, the performance of the left-out class drops significantly, relative to its performance when training data is available (i.e. the point for each left-out class falls below the diagonal). But when sharing is used, the drop in performance is relatively small, all points being just below the diagonal.

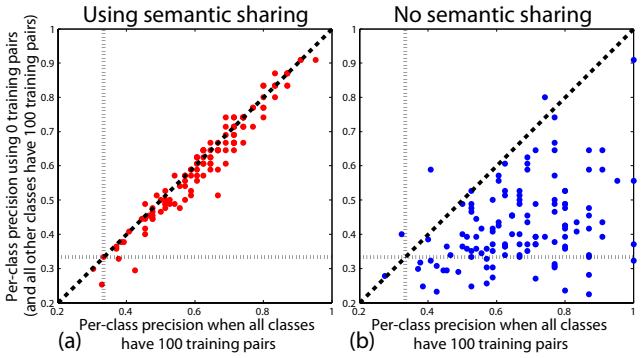

Figure 7. An exploration of the performance with 0 training examples for a single class, if all the other classes have 100 training pairs. **(a)**: By using the sharing matrix A, we can obtain a good performance by transferring labels from semantically similar classes. **(b)**: Without it, the performance drops significantly.

4.3. Experiments with 74,569 Categories and 79,302,017 Images

Our final experiment applies the eigenfunction approach to the whole of the Tiny Images dataset. The gist descriptor for each image is mapped down to a 32D space using PCA and k=48 eigenfunctions were used. We use all 445,954 CIFAR labels (64,185 of which are +ve) covering 386 keywords and propagate them to the rest of the 79,302,017 images using a 74,569 by 74,569 semantic affinity matrix (there are 74,569 keywords in the Tiny Images dataset). We use the noisy labels in the same manner as Section 4.2. Having precomputed the eigenfunctions, we can re-rank any chosen keyword by solving Eqn. 1, which takes around 1 minute in Matlab on a large PC (equivalent to $\sim$0.75μs per image). Motivated by Fig. 7, we show in Fig. 1 the approach operating on two classes for which no CIFAR labels exist, obtaining qualitatively good results. However, this ability is limited to keywords relatively close to the 386 keywords for which we have labels – an issue which will be alleviated with a more diverse set of labels. More examples may be found in the supplementary material.

5. Discussion

We have proposed a novel semi-supervised learning scheme that is linear in the number of images, and then

Figure 8. Test images from 10 keywords drawn from the 126 keyword sub-set with manual annotations. The border of each image indicates its label (used for evaluation purposes only) with respect to the keyword, green = +ve, red = -ve. The top row shows the initial ranking of the data, while the bottom row shows the re-ranking of our approach trained on 126 classes with 100 training pairs/classes.

demonstrated it on challenging datasets, including one of 80 million images. The approach can easily be parallelized making it practical for Internet-scale image collections. Currently the noisy labels are used in a simple way. Making better use of them, as well as incorporating other cues such as image rank, might allow our scheme to operate on classes further from those with labels.

References

[1] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian based manifold methods. *Journal of Computer and System Sciences*, 2007. 4

[2] Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. In *NIPS*, volume 16, pages 2197–2219, 2004. 2, 4

[3] T. Berg and D. Forsyth. Animals on the web. In *CVPR*, pages 1463–1470, 2006. 2

[4] O. Chapelle, B. Schlkopf, and A. Z. editors. *Semi-Supervised Learning*. MIT Press, Cambridge, Mass., 2006. 2, 3

[5] R. R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusion as a tool for harmonic analysis and structure definition of data, part i: Diffusion maps. *PNAS*. 2, 4

[6] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 2008. 1

[7] C. Fellbaum. *Wordnet: An Electronic Lexical Database*. Bradford Books, 1998. 5

[8] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *ICCV*, pages 1816–1823. 2

[9] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 5, 6

[10] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. 2008. 2

[11] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *CVPR*, 2007. 2

[12] A. Kumar and C. Sminchisescu. Support kernel machines for object recognition. In *CVPR*, 2007. 6

[13] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. *IEEE PAMI*, 30(10):1699–1712, 2008. 3

[14] L. J. Li, G. Wang, and L. Fei-Fei. Optimol: automatic object picture collection via incremental model learning. In *CVPR*, 2007. 2

[15] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21:113–127, 2006. 2, 4

[16] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001. 6

[17] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77. 1

[18] B. Schoelkopf and A. Smola. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press,, 2002. 3

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22:888–905, 2000. 3

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

[21] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *IEEE PAMI*, 30(11):1958–1970, November 2008. 1, 5

[22] L. van Ahn. The ESP game, 2006. 1

[23] S. Vijayanarasimhan and K. Grauman. Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *CVPR*, 2008. 2

[24] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. 3, 4

[25] B. Yao, X. Yang, and S. C. Zhu. Introduction to a large scale general purpose ground truth dataset: methodology, annotation tool, and benchmarks. In *EMMCVPR*, 2007. 1

[26] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *NIPS*. MIT Press, Cambridge, MA, 2004. 2

[27] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *In ICML*, pages 912–919, 2003. 2, 3

[28] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. In *ICCV*, 2007. 2