Unsupervised Image Ranking

Eva Hörster[*] , Malcolm Slaney, Marc'Aurelio Ranzato[†] , Kilian Weinberger
Yahoo! Research
Sunnyvale, CA 94089
hoerster@informatik.uni-augsburg.de, malcolm@ieee.org, ranzato@cs.toronto.edu, kilian@yahoo-inc.com

ABSTRACT

In the paper, we propose and test an unsupervised approach for image ranking. Prior solutions are based on image content and the similarity graph connecting images. We generalize this idea by directly estimating the likelihood of each photo in a feature space. We hypothesize the photos at the peaks of this distribution are the most likely photos for any given category and therefore these images are the most representative. Our approach is unsupervised and allows for various feature modalities. We demonstrate the effectiveness of our approach using both visual-content-based and tag-based features. The experimental evaluation shows that the presented model outperforms baseline approaches. Moreover, the performance of our method will only get better with time as more images move online and it is thus possible to build more detailed models based on the approach presented here.

1. INTRODUCTION

With the growth of user-generated content on the web, we have datasets of unprecedented size. Helping users find images in these large databases is an important problem. However, without external sources of information, such as links, ranking images via a search engine is hard.

Given a query term, we want to find the most relevant images in a large web-scale collection. The user enters a query term, much as is done in a text search. We return the images that we judge are most relevant. We solve this problem by using only the information in the picture and any supplied metadata such as tags. For now we only consider a single query term, but our approach is easily extended to more than one term.

Our approach is based on a simple hypothesis. There are billions of diverse photographs on the web (see Figure 1).[1] We assume that the most common images are the most relevant images related to the query term. Because many different people captured very similar images, we believe these are the most interesting and relevant shots. For instance, if there are many images of the Golden Gate Bridge from a similar angle and under similar lighting conditions, we assume that these pictures capture a very relevant shot. In other words, many people agree that this is an interesting shot by virtue of the fact that many photographers have put the same or similar photos into an online collection. In essence, people are voting on their favorite images of the query term by their pictures (i.e. camera clicks).

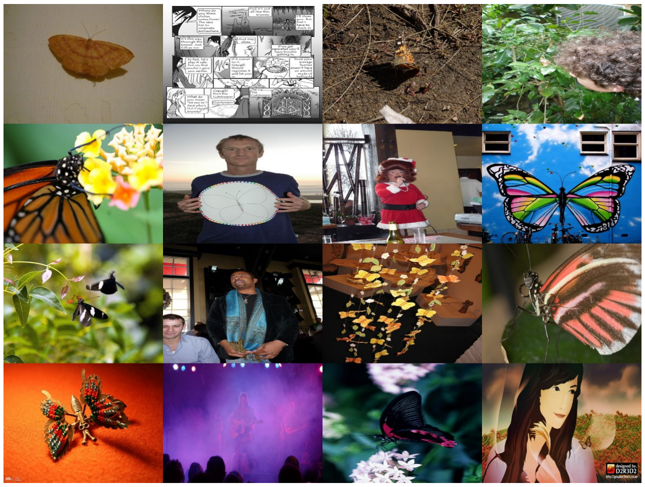
Figure 1: Modern image collections are diverse. These images show the range of images in Flickr$^{\text{TM}}$labeled *butterfly*.

Moreover, in our approach the most representative images are not only based on the image content but also based on other sources of information such as tags, date and time, location. For instance, if most people tag a shot of the *Colosseum* with *Italy* and *Rome*, then we use this common metadata agreement to improve our relevance calculation.

In order to determine the most likely image given a query term, we learn a simple, probabilistic model using an unsupervised approach. We call our approach unsupervised because we do not consider external factors such as web links, search click-through results or other sources of user feedback. This model is then used to find the images whose content and metadata give us the highest probabilities.

In contrast to previous work, we do not consider only certain classes of images such as landmarks [17] or objects [1, 2]; we describe a general framework for all kinds of images such as objects,

[*] Currently at the Univ. of Augsburg.

[†] Currently at NYU.

[1] Images shown in this paper were published using the Creative Common License and their attribution is available at http://research.yahoo.com/files/2009-004_hoerster.pdf.

places, events, etc. However, we exclude searches for images of a particular person and for current events. We believe faces should be solved differently, i.e. it may require face-recognition techniques, because human perception is sensitive to faces. Current events are not amenable to our approach because there is so little content — the one best shot might be iconic because it was captured by only one photographer.

In this paper, we demonstrate the effectiveness of our approach by using two types of features for describing our images in the database: visual features based on the image content; and semantic features based on user-defined tags. In general, our model can be used with any combination of features, perhaps from multiple modalities.

Both image and text (or tag) data are noisy signals. Image features are often designed to be invariant to common image changes, but they will never be perfect. Therefore, we must accommodate a wide range of image modifications. Similarly, user-supplied tags associated with images are both good and bad. Users specify a tag for a photo for many different reasons, thus giving us important, but noisy, semantic information. The tags are good because they give us information about semantic content that is hard for machines to infer, but tags are noisy because their use is so capricious. A person going to Tokyo might label all the photos from his trip with the word *Tokyo*, even though some of the pictures show people eating, or the inside of a meeting room. For this one person, these pictures all represent the concept *Tokyo*. We extract good information from these noisy signals by building statistical models from a large number of images.

1.1 Contributions

Our work describes improvements to three areas of the science of multimedia ranking: models, image features and text features.

- We propose a general model based on image probability and show that it is an appropriate way to find highly relevant images. Moreover, our approach is based on feature densities and thus it easily scales to databases with millions of images, especially compared to previous techniques that compute pair-wise similarity.

- Our proposed model allows a wide range of features to describe the images. We demonstrate the effectiveness of our model using two kinds of features, visual and textual features based on tags. In contrast to most previous works where only visual features are used, we get better image ranking when adding textual features (tags) to our model.

- We use visual and textual features based on deep networks. The visual features allow us to better describe highly variable image categories compared to previous works based on SIFT features, which mainly concentrate on landmark images. Also, our features are fast to compute due to their feed-forward architecture and thus they are appropriate for describing very large image collections.

The paper is organized as follows. We describe our probabilistic model in Section 3 and the features used in our approach in Section 4. Section 5 gives implementation details. Our results are in Section 5. But first we wish to describe previous work and show how it fits into a more general framework.

2. RELATED WORK

There are a number of previous papers that address the issue of finding iconic or most representative images from a collection of photos. One approach builds a graph connecting similar images and then uses either eigenvector similarity [14] or spectral clustering [4] to find the images that are at the "center" of the graph. Similarity in these approaches is computed on an item-by-item basis using feature detectors such as SIFT.

Another technique builds clusters of images and then use either intra-cluster similarity [17] or cluster centroids [1, 2] to find the most representative images. Berg's [2] work in particular extends the clustering idea by finding images that have a clear foreground object and thus are more likely to represent "good" images. Works by Hsu describe pseudo-relevance feedback to improve search ranking using both cluster-based similarity [11] and graph-based similarity [12]. Our work takes a more direct approach for ranking.

A number of works have studied the scene-summarization problem [26, 19, 28]. These works aim to find canonical images of (unchanging) landmarks by matching features between images. The University of Washington work [26], in particular, aims to discover the different viewpoints by clustering the images and choosing the best image in viewpoint space. Similarly to the work described here, they build a probabilistic model of image distributions in all possible dimensions.

We believe the unsupervised approach described in this paper is a unifying metaphor for choosing most relevant images, and the earlier (excellent) works are different approximations for the overarching model we present here. Both tightly coupled graphs of photos and image-cluster centroids are found near the peaks in a probability distribution. Moreover, the previous works mostly determine their ranking based only on the image content and rarely take other modalities into account. We demonstrate our approach can be used with different kinds of features, visual features as well as semantic features derived from tags.

There are also supervised approaches that aim to train a classifier for ranking images. Such approaches need clean, labeled training data and output the images with the highest classification score, i.e. the ones furthest from the SVM margin [6, 25]. Other systems use feedback from users' searches and then learn the results that are most likely to be selected [15]. Our approach uses none of this information — just the frequency of each type of image. In a real system, one might initially use an unsupervised approach and then augment it with supervised data, i.e. click data, as it becomes available.

Our approach is similar to an idea proposed by Hua [13]. We generalize her approach, apply it to multimedia, and show how it works with real queries.

3. MODEL

We hypothesize that the most representative images are the most likely images related to the query term. In order to determine the most likely images related to a query term, we build a model for this term. In our model, people are essentially voting on their favorite images of *Paris* or *purple kittens* by the number of images they upload and tag by those names, thus our approach can only work when we have a large collection of images, all independently taken my many different people.

To build the model, we start with a broad and all-inclusive set of images that satisfies the query. In our implementation, we derive this set of images automatically based on the tags associated with the images. We simply use all images that have been tagged by their authors with the query term. We can enhance this approach by using query-expansion techniqes. Note, our image set is derived automatically, i.e. the image set contains a significant number of noisy images not necessarily showing the desired image content due to the subjectivity and ambuguity of tags. Besides these images that are related to the query term, we have no additional data about the query or which images are preferred.

Next, we assume that we find the most representative images by looking for the peaks in a simple, probabilistic model of the image set's distribution. Thus, the model for images tagged with the query term t is given by

$$P(I_j|t) = P(f_j^1, f_j^2, f_j^3, ..., f_j^n|t), \tag{1}$$

where f_j^i is the i-th feature that describes the content or something about how the image I_j was collected. The simplicity of this equation belies the art involved in our approach.

In order to represent our image, we need to choose a set of features, denoted by $f^1, ..., f^n$, that accurately reflect the available images. Here we could use any type of feature, based on one modality or based on several metadata cues as well as the image content. Having defined our feature set, we need to build the model that captures the distribution of the images related to the query term in this feature space. Therefore, we learn the probabilities of our model by estimating a non-parameteric density for all the different modalities we measure from the (noisy) image set.

Having computed our model, we define the rank R of an image I_j to be inversely proportional to the probability of the current image I_j given the model for the query term t:

$$R \propto 1/P(I_j|t). \tag{2}$$

We can apply the model to all images in our database or we can compute the rank only for a pre-filtered set of images likely showing the desired content. In our system, we use the latter approach and we only consider images for ranking that have the query term associated as a tag. This is reasonable as in a large scale image collection because we are interested in obtaining high precision, i.e. the retrieved images should show highliy relevant content, whereas high recall, i.e. finding all relevant images, is not important due to the large amount of available images.

There is a dichotomy between the approach in this paper, based on likelihood, and an approach based on classification, which is often based on a calculation of the posterior probability. The two expressions are related through Bayes rule. The posterior is equal to

$$P(w_i|x) = P(x|w_i)P(w_i)/P(x), \tag{3}$$

where w_i is one of the class labels, x is a (vector) representation of the image, and $P(\cdot)$ is the probability of the class or an image occurring with this feature value. The likelihood is written as

$$P(x|w_i) = P(w_i|x)P(x)/P(w_i). \tag{4}$$

These two quantities have peaks in different locations.

We illustrate the difference with a simple three-class example. Figure 2 shows the total probability for the sum of three Gaussian distributions, one for each class of data, separated by 120 degrees. An 'x' marks the center of each distribution. The right side of Figure 2 shows the posterior probability distribution for class w_1. The overall shape is triangular, correctly indicating that in the region between the two lines objects are most likely to come from class w_1. We again used an 'x' to mark the peak of the w_1 likelihood function. But an '*' marks the point where the posterior probability grows to 0.99. The posterior is maximum well to the right of the likelihood peak. If you want to be *sure* that you have not made an error, then one should choose a point at $(0, \infty)$ because a point far to the right is most likely to be class w_1.

The difference between the posterior and the likelihood can be seen in how the two expressions use the data distribution $P(x)$. The likelihood calculation uses the overall distribution in the numerator of the equation, causing the maximum to be reached where there is a lot of data. Conversely, the posterior calculation puts the data distribution in the denominator. Thus, when there is little data, $P(x)$ is small, and the posterior is forced high. A decision based on the posterior is safe, but it won't necessarily be a common image.

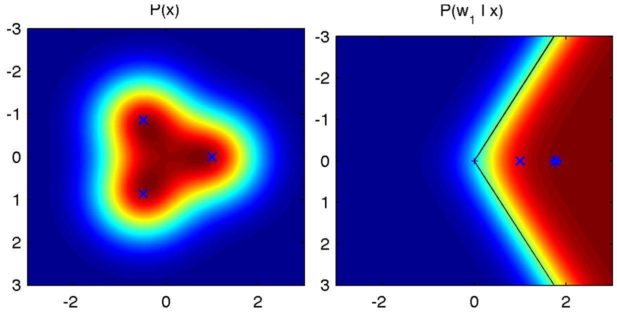

Figure 2: The difference between likelihood and posterior. The left figure shows the overall probability distribution of the data. The right figure shows the posterior probability distribution for class w_1. The posterior increases to the right. The maximum of the likelihood is marked with a $+$ while a conservative estimate for the maximum of the posterior is marked with a '*'.

3.1 Approximation

Even on the web, getting enough data to train a full model as proposed in Eq. 1 is prohibitive. Thus, we might assume that each feature is statistically independent of the others so we can write the image probability as

$$P(I_j|t) = \prod_i P(f_j^i|t). \tag{5}$$

Note, we often have multi-dimensional features and we treat each dimension as an independent feature so we only need to compute one-dimensional density estimates.

If we have enough data and find correlations between two or more features (or feature dimensions), we get better accuracy by building joint probability models of these features/dimensions. Such a division of the entire model into smaller submodels and assuming independency between those models is less restrictive than assuming independence between each and every metadata dimension but less expensive than computing a joint distribution for all of the cues. We cannot hope to have enough data to model the full distribution, but can take into account feature inter-dependencies to jointly model related variables. Finally, the choice of features is arbitrary and one can balance the information in different dimensions by learning or setting a weight per dimension. The most general form of the probability model is

$$P(I_j|t) \propto \prod_i [P(F_j^i|t)]^{\alpha_i}, \tag{6}$$

where $P(F_j^i)$ is a full model of the probability of feature set i and α_i is a weighting factor. We set the α_is to 1 in our implementation.

4. FEATURES

To demonstrate the utility of the our above presented approach, we consider two kinds of features: one based on the pixels in the image (see Section 4.1); and the other based on the tags supplied by the user to describe the content or other properties of the image (see Section 4.2.) There are interesting peaks in other metadata modalities such as location and time [27], but including those into the model is the subject of future work.

Both types of features are based on feed-forward networks, whose parameters are trained off-line from hundreds of thousands of web images. We believe that it is important to learn the parameters for feature computation from data, not only the model representing the images itself. Images from the web are diverse and seldom contain only a single object; thus, object detectors trained on clean, labeled databases are not an option. Our chosen features are not the only answer, but they share a common framework by approximating the data in a low-dimensional space with minimum reconstruction error. Most importantly, the dimensionality of the feature describing an image is fixed, so that the 42nd dimension always represents, for example, a particular kind of texture in the upper-left corner of the image. Once the parameters of the networks are trained and fixed, the feature computation for novel images is inexpensive. It is important to compute the features representing the different image modalities efficiently by using a bottom-up system because we cannot wait seconds to process an image. Lastly it should be noted that our features use a non-linear representation because linear projections can't efficiently encode images; linear systems can capture only 2nd order statistics.

In the sub-sections that follow we describe the image features (Section 4.1), the text features (Section 4.2) and then show how we reliably estimate the density function of the images in the (potentially joint) feature space.

4.1 Pixel Features

In order to learn locally, shift-invariant, sparse representations, we combine two recently proposed algorithms [23, 16]. As stated above, there are alternative features that could we used as well, such as GIST [21] or SIFT [20] features in combination with a bag-of-words model. However, features based on a feed-forward network are computationally cheap and we believe that they are appropriate for modeling various kinds of highly variable image categories.

First, we describe the baseline sparse-coding algorithm. Second, we propose a simple extension to learn representations that are not only sparse but also locally shift invariant. Spatial invariance is desirable because we are not interested in the exact location of objects. And finally, we explain how we produce feature hierarchies by stacking these models. We use these features because they perform well in an image classification task (see Section 5.2).

Olshausen and Field [22] proposed a sparse coding model. Given a vectorized input image patch $I \in R^M$, we seek the code $Z \in R^N$ (with possibly $N > M$) that can reconstruct the input, the code is sparse and minimizes the following objective function:

$$L(I, Z; W_d) = \|I - W_d Z\|^2 + \lambda \sum_k |Z_k|, \qquad (7)$$

where $W_d \in R^{M \times N}$ is a matrix that we learn, and $\lambda \in R^+$ is a hyperparameter controlling the sparsity of the representation. We learn the matrix W_d with an on-line block-coordinate gradient-descent algorithm. Given a training-image patch: (1) we minimize the loss in Eq. 7 w.r.t. Z to produce the optimal sparse code; and (2) we update the parameters W_d by one step of gradient descent using the optimal sparse code, and we normalize the columns of W_d to 1. The re-normalization is necessary since the loss is trivially decreased by multiplying and dividing W_d and Z by the same factor. When we apply this algorithm to natural images, it learns features that resemble Gabor wavelets. The main problem with this code is the expense of using it. Computing the sparse code corresponding to an input image patch requires solving a convex but non-quadratic optimization problem. Although many optimization algorithms have been proposed in the literature [18, 3], the iterative procedures are prohibitively expensive when encoding whole images in large-scale web applications.

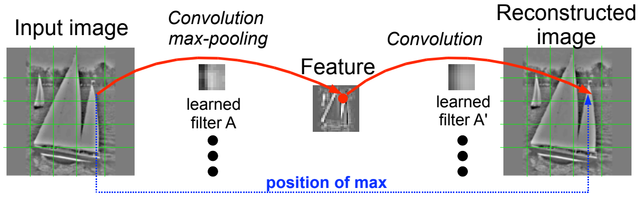

Figure 3: One stage of the image-coding/decoding pipeline. The feature we use to represent the image is shown in the middle of the pipeline. The transformation parameters that represent the location of the maximum in each window, are passed from coder to decoder because they represent spatial data we want the representation to ignore.

Therefore, we use a feed-forward approximation [16]. We train a feed-forward regressor to directly map input patches to sparse codes. We consider the class of $D \tanh(W_e I)$ functions where $\tanh$ is the hyperbolic tangent non-linearity, D is a diagonal matrix of coefficients, and W_e is a $N \times M$ matrix. Training consists of minimizing the squared reconstruction error between the output of this function and the optimal sparse codes w.r.t the parameters W_e and D. We perform the optimization after optimizing W_d, or jointly by adding this extra error term to the loss of Eq. 7:

$$L(I, Z; W_d, D, W_e) = \|I - W_d Z\|^2 + \lambda \sum_k |Z_k| + \|Z - D \tanh(W_e I)\|^2. \qquad (8)$$

Since the joint optimization is faster (because the inference step enjoys the initialization provided by the feed-forward regressor), we choose the latter optimization strategy. The training algorithm is the same one, alternating a minimization over Z and a parameter update step over (W_d, W_e, D). Note that the rows of matrix W_e can be interpreted as trainable filters that are applied to the input.

In order to make the codes not only sparse but also translation invariant over small spatial neighborhoods, we extend this algorithm by applying a trick [23]. The idea is to use the filters *convolutionally* over the input image patch (which is not vectorized and whose spatial resolution is larger than the support of the filters) and to take the maximum across non-overlapping windows. Clearly, the resulting code becomes invariant to translations within the corresponding window. The reconstruction is similar to before, and is done convolutionally as well. First, the code units are placed in the feature maps at the locations where the maxima where found, and then the resulting feature maps are convolved with the reconstruction filters, and finally summed up to produce the reconstruction of the input. Figure 3 shows the coder and decoder.

The learning algorithm remains unchanged when we add a spatially invariant aspect to the sparse code because both algorithms reconstruct the input while satisfying a sparsity constraint. In particular, these algorithms do not make any specific assumption on the input. Therefore, it can be replicated to build a feature hierarchy, analogous to the training scheme employed in deep learning methods [8]. The algorithm is first trained using image patches. Once we learn the filter banks, we use the feed-forward mapping function to directly predict approximatly sparse and locally shift-invariant codes to train another layer. We repeat the same greedy process for as many layers as desired. The resulting features are sparse, locally shift-invariant and they are produced by a simple feed-forward pass through several stages of convolution and max-pooling.

4.2 Semantic Features

To transform tags associated with an image to semantic image features, we use a bag-of-words (BOW) description in combination with a deep network. Our deep network uses multiple, non-linear, hidden layers and was introduced by Hinton et al. [10] in the context of modeling image patches. A modified version of the model was proposed by Salakhutdinov [24] for text documents. A deep network consists of multiple, non-linear, latent feature layers, each capturing the strong correlations of the feature activations in the level below.

This deep network computes, similar to the model for pixels described in the previous subsection, a low-dimensional representation from which the text documents represented by their BOW models can be reconstructed with low error. We apply this model here to the BOW description of the images' tags. The learning procedure for such a deep model consists of two stages. In the first stage, the pre-training, we compute an initialization based on restricted Boltzmann machines (RBM). The second stage refines the representation by using backpropagation.

RBMs provide a simple way to learn a single layer of hidden features without supervision. They consist of a layer of visible units that are connected to hidden units using symmetrically weighted connections. Note that a RBM does not have any visible-visible or hidden-hidden connections. We apply one step contrastive divergence [9] to learn the variables of a RBM, i.e. its weights and biases.

To extend this and construct a deep network, Hinton [10] proposes to learn additional layers of features by treating the hidden states of the lower-level RBM as the visible data for training a higher-level RBM, that learns the next layer of features. By repeating this greedy layer-by-layer training several times, we learn a deep model that is able to capture higher-order correlations between the input units. Note that the learning algorithm for pixels representation presented above uses a similar approach for learning a feature hierarchy. Here the outcome of a lower layer is also used as the input to learn another feature layer.

After pretraining for all layers, we further refine the parameters of the deep model. This is done by unrolling the layers to create an autoencoder [10]. Using the pretrained biases and weights as initializations, the backpropagation algorithm is used to fine-tune the parameters for optimal reconstruction of the input data, i.e. in our case the BOW descriptions based on the image tags.

The input vector from tags to such a deep network is a word-count vector. We first divide each entry of the respective vector by the total number of tags associated with the current image. This creates a discrete probability distribution over the finite tag vocabulary for each image. To model the probability distributions in the input layer, we use a softmax at the visible units in the first level RBM while its hidden units and also all other units in the deep network are binary. However the output units at the top level of the network are linear. We use the multi-class crossentropy error function to refine the weights and biases in the backpropagation algorithm.

Once the deep network is trained, we derive a low-dimensional representation of an image in the semantic space by applying the learned model to its BOW description and using its top-level unit values as its low-dimensional description. Note, the mapping from the word count vector, i.e. the basic tag description, to a high-level semantic feature only consists of a single matrix multiplication and single squashing function per network unit.

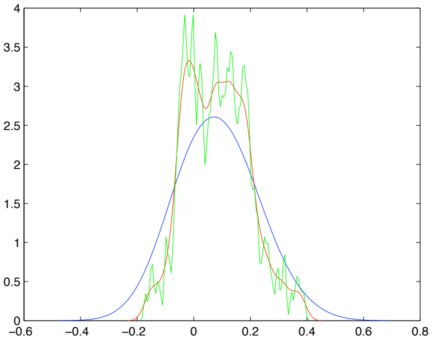

Figure 4: Three examples of density estimation: one too smooth (blue); one too detailed (green); and one which maximizes the likelihood of the test data (red).

4.3 Density Estimation

Having computed the features for each of the images in our collection, we perform non-parametric density estimation in order to derive the probabilities $P(f_j^i|t)$. We compute a one-dimensional probability density for each feature dimension using Parzen's windows [5]. For each feature (pixel or semantic), we use a Gaussian kernel and perform 10-fold cross validation to find the best kernel width. The goal of this step is to build a model of the data that accurately reflects the underlying probability distribution. We find the kernel variance that defines a likelihood model that best predicts the held-out test data. Figure 4 gives an example of this calculation. The distributions are often bimodal or skewed. The product of these distributions is a simple model of image likelihood as a function of the image features.

5. IMPLEMENTATION

5.1 Dataset

We consider the following 20 query terms for our evaluation: *baby, beetle, butterfly, carnival, chair, Christmas, CN Tower, coast, Colosseum, flower, forest, Golden Gate Bridge, highway, horse, mountain, sailboat, sheep, Statue of Liberty, sunset, wedding*. This list includes objects, landmarks, scenes, events and places.

We downloaded nearly 4.8 million public, geotagged Flickr images, all with at least one of the tags listed above. The number of images per category ranged from 3,700 to 683,000 images.

5.2 Pixel Feature Implementation

In this paper, we report the results of experiments using three layers of convolution and max-pooling to analyze each image. The input image is converted to YUV and down-sampled so that the longest side is 158 pixels long. The Y channel is high-pass filtered to remove changes in illumination. After downsampling, filtering and zero padding all image patches are 140×140 pixels. The filter banks of the first two stages have kernels of size 9×9, and the max-pooling is performed over 4×4 and 5×5 non-overlapping windows at the first and second stage, respectively. The third-stage filters have the same size support as one of the second stage feature maps, and therefore, no pooling is performed there. The number of feature maps is equal to 128 at the first stage, 512 at the second stage and 1024 at the third stage. The final feature vector is 1024-dimensional.

We trained our feature extractor with the unsupervised algorithm on natural image patches randomly selected from images downloaded from Flickr, and then validated the parameters of our extractor by testing its performance using the Caltech-256 object-classification task [7]. This allowed us to fine-tune the architecture of our network and the number of layers and nodes. Note that the Caltech-256 data gave us a separate validation set. However, it is important to build our feature extractor using Flickr images since its style is much less constrained than the Caltech-256 set and Flickr represents a larger dataset.

The validation of the parameters proceeded as follows after building the feature extractor as described above: we compute the features for each image of the Caltech-256 dataset; then we train a linear SVM classifier with 30 training samples per class. The average recognition accuracy was equal to 26% which is considered to be a good result — the state-of-the-art is in the mid-thirties [7].

5.3 Semantic Feature Implementation

To train our tag model, we sample a number of images from each class in our dataset (see Section 6) and use this set to learn the parameters of the deep network that can be used to compute features for each image independent of the current query term.

The first step is to choose a vocabulary. This is done by first listing all tags associated with at least one training image. Next we filter this list to keep only those tags that are used by a certain number of authors/owners — ten in our implementation. Additionally, we remove all tags containing numbers. To map singular and plural forms of objects into one single word we use a very simple scheme which pools words which are equal up to the letter 's' at the end of the word. A more sophisticated approach to building the vocabulary may be useful to consider for future work, as we also do not consider any type of translation of the words.

Our final vocabulary consisted of 3674 words. Having chosen the vocabulary we can represent each image by a word-count vector. The resulting representation is in most cases very sparse as users typically only use up to 5 words/expressions to tag an image.

The trained deep network consisted of three hidden layers with a 3674-1000-400-100 structure. Thus, we obtain a 100-dimensional semantic representation for each image. We used 84,000 images from all categories for learning the deep network, 25 iterations for pre-training each layer and 50 iterations to optimize the autoencoder.

5.4 Densities

For training the feature densities we randomly select 10,000 images per query term; if fewer images were availabe in our collection, we used less. Moreover, we only consider one (randomly chosen) image per owner per tag, as we do not want the densities biased towards the images of one photographer. This can easily happen as there are photographers who upload thousands of images all associated with one tag and all very similar but not really relevant for the tag.

In order to represent the estimated non-parametric densities, we evaluate them at 5,000 equally distributed points between the minimum and maximum feature value and store the found values. Normalizing their sum to 1 gives us discrete probabilities for each of those points. Thus, when performing ranking we map each feature to its nearest value in this dimension (a form of quantization) and derive directly the probability associated, this ensures fast computation of image ranks and relatively low computational effort.

We also implemented a cleaning algorithm to reduce the effects of common background images in our density calculation. This process removes images that are not surrounded by other images of the same class, i.e. with the same tag. For each class, we perform a nearest-neighbor calculation using 10,000 in-class and 10,000 background images. We keep an image if more than 50% of its neighbors are in the same class. Note, the number of training images for some query terms is less than 10,000 before filtering, thus making it harder in those cases to reach the required 50%.

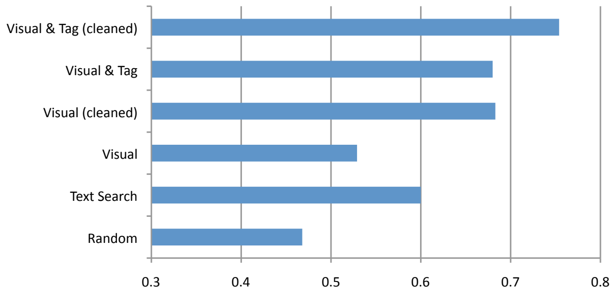

Figure 5: Mean scores for the top-15 images over all categories for our baseline method (random), a text-based search engine, and 4 different variants of the unsupervised ranking algorithm.

In this work, we do not address image diversity. But given a probabilistic model such as described here, one has the right representation so images can be chosen from all over the range of data, perhaps emphasizing the peaks. In Figure 8, we see images from different regions of feature space. When selecting the top ranked images for the query *beetle* according to the model, we find high-probability images in different areas representing the different meanings of the word, i.e. showing either the insect or the car.

6. RESULTS

To test our proposed ranking approach, we calculated likelihood models for each of 20 query terms. We then calculated the probability of each of our 4.8 million images using models built from pixel features only as well as models built using both image and semantic data, i.e. pixel and tag features. We compared the ranking performance of those models, with and without training data cleaning (see Section 5.4), to two different baseline measures: a random set of images from the respective category and images that were ranked highly by the Flickr search API (which uses text matching).

We evaluated the performance of our approach in a user study by: displaying the top 15 images, but only one image per author, for each of the 20 test queries; and asking six test users to judge the result images. In the test, subjects assign one point to an image they consider relevant to their query, give each somewhat relevant image 0.5 points, and give each irrelevant image 0 points. For each query, we report the mean score per image over all test users.

Figure 5 compares the mean scores over all queries of the different approaches. Using visual features alone gives better performance than a random set of images. Adding tags to the model improves the relevance scores, as did cleaning the models training data to remove images that were common to many query terms. Summarizing, when using both, visual and semantic, features, our proposed model outperforms the baseline approaches.

Table 1 summarizes the results as a function of the query. Images of objects with a low visual diversity, such as *Colosseum* and *Statue of Liberty* are easy. Classes such as *wedding* or *highway* are more difficult, even though we did well, because of the wide range of images. Many wedding pictures, for example, consist of groups of happy people. They were probably taken at weddings, but it is

	Random	Text Search	Unsupervised
baby	0.52	0.23	**0.76**
beetle	0.60	**0.83**	**0.83**
butterfly	0.75	0.78	**0.88**
carnival	0.28	0.43	**0.81**
chair	0.41	0.39	**0.79**
Christmas	0.21	0.26	**0.67**
CN Tower	0.46	0.65	**0.98**
coast	0.27	0.40	**0.76**
Colosseum	0.77	**0.99**	0.93
flower	0.77	**0.82**	0.71
forest	0.12	0.68	**0.79**
Golden Gate Bridge	0.56	**0.95**	0.64
highway	0.10	0.33	**0.37**
horse	0.29	**0.86**	0.72
mountain	0.27	0.61	**0.68**
sailboat	0.72	0.59	**0.77**
sheep	0.51	0.62	**0.79**
Statue of Liberty	0.42	0.56	**0.99**
sunset	**0.76**	0.63	0.62
wedding	**0.58**	0.41	**0.58**

Table 1: Detailed relevance scores for the different categories. The winning score for each category is shown in bold.

hard for humans to judge their relevance. Moreover, the quality of results for some queries such as *sunset*, *flower* or *butterfly* was good even for the random approach indicating that majority of the images tagged with this words were showing the desired content. Other query terms, e.g. *Christmas* or *forest*, are hard because they are less distinct; nevertheless, our approach gives good performance.

To illustrate results of our algorithm, we applied the learned models to a different set of recently uploaded images for each query from a set of Flickr images that are licensed under the Creative Commons License (and thus we can republish). We show the resulting top-16 images for the queries *chair*, *baby*, *beetle* and *Christmas* in Figures 6 to 9.

7. CONCLUSIONS

We demonstrate an unsupervised approach to determine image relevance. This approach works well, not only for images of landmarks, but also objects, scenes and events. We described features based on convolutional neural networks (for pixels) and related text features based on a deep network. We show how we use a probabilistic model of image distributions to find the most common images, even when we approximate the distribution with a product of independent dimensions. The experimental evaluation showed that our approach outperformed the baselines and users judged the results as most relevant when we used both visual and tag data to build our models.

This algorithm is important because it forms the first step in a general machine-learned ranking algorithm. We may use this algorithm before we have human relevance judgements, or for queries that are too rare for detailed human data. Most importantly, we believe the need and the accuracy of this approach will only improve as more images find their way online and it is thus possible to build more detailed models.

8. REFERENCES

[1] T. Berg and D. Forsyth. Animals on the web. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1463–1470, 2006.

[2] T. L. Berg and A. C. Berg. Finding iconic images. In *The 2nd Internet Vision Workshop at IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[3] J. Bioucas-Dias and M. Figueiredo. A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, 2007.

[4] D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world's photos. In *Proc. 18th International World Wide Web Conference*, 2009.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[6] D. Grangier and S. Bengio. A discriminative kernel-based approach to rank images from text queries. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1371–1384, Aug. 2008.

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

[8] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[9] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, 2002.

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[11] W. H. Hsu, L. S. Kennedy, and S.-F. Chang. Video search reranking via information bottleneck principle. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 35–44, New York, NY, USA, 2006. ACM.

[12] W. H. Hsu, L. S. Kennedy, and S.-F. Chang. Reranking methods for visual search. *Multimedia, IEEE*, 14(3):14–22, July-Sept. 2007.

[13] M. Hua, J. Pei, A. W.-C. Fu, X. Lin, and H. fung Leung. Efficiently answering top-k typicality queries on large databases. In *VLDB*, pages 890–901. ACM, 2007.

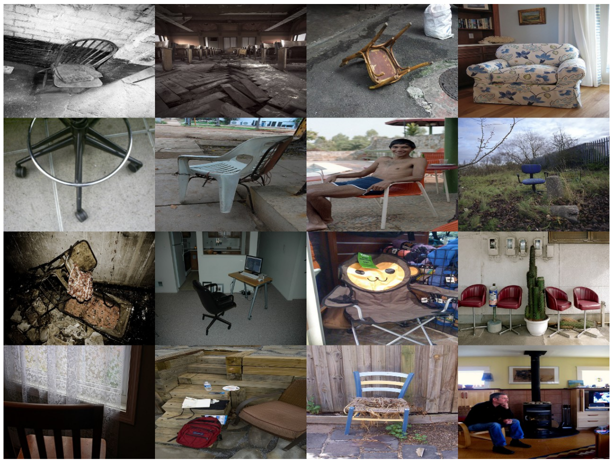

Figure 6: Sample results for the *chair* query for visual and tag model trained using cleaned data. This is a simple query with many complicated images, but all images show chairs.

[14] Y. Jing and S. Baluja. Pagerank for product image search. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 307–316, New York, NY, USA, 2008. ACM.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[16] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU, 2008. Tech Report CBLL-TR-2008-12-01.

[17] L. S. Kennedy and M. Naaman. Generating diverse and representative image search results for landmarks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 297–306, New York, NY, USA, 2008. ACM.

[18] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006.

[19] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *ECCV (1)*, volume 5302 of *Lecture Notes in Computer Science*, pages 427–440. Springer, 2008.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[21] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006.

[22] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[23] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[24] R. R. Salakhutdinov and G. E. Hinton. Semantic hashing. In *Proc. SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[25] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct. 2007.

[26] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *ICCV*, pages 1–8. IEEE, 2007.

[27] K. Q. Weinberger, M. Slaney, and R. van Zwol. Resolving tag ambiguity. In A. El-Saddik, S. Vuong, C. Griwodz, A. D. Bimbo, K. S. Candan, and A. Jaimes, editors, *ACM Multimedia*, pages 111–120. ACM, 2008.

[28] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: building a web-scale landmark recognition engine. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, Miami, Florida, U.S.A, June, 2009.

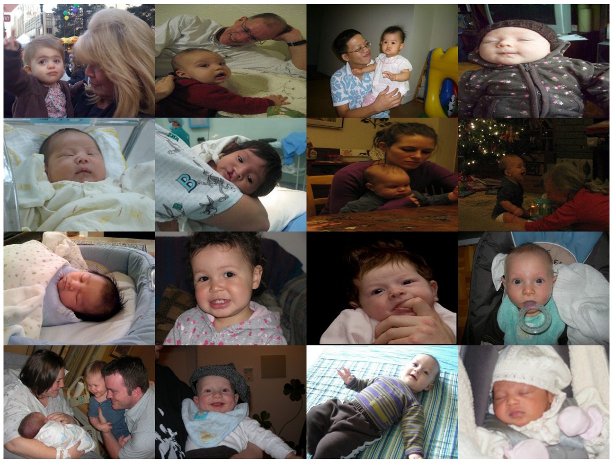

Figure 7: Sample results for the *baby* query for visual and tag model trained using cleaned data. Our approach works well at finding large round (baby) faces.

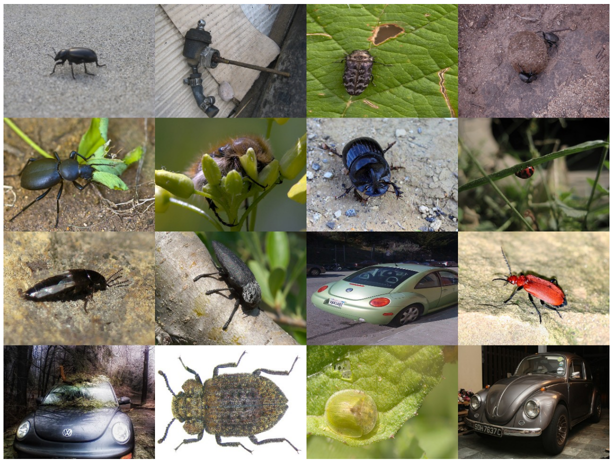

Figure 8: Sample results for the *beetle* query for visual and tag model trained using cleaned data. The unsupervised approach discovers both meanings of the word beetle.

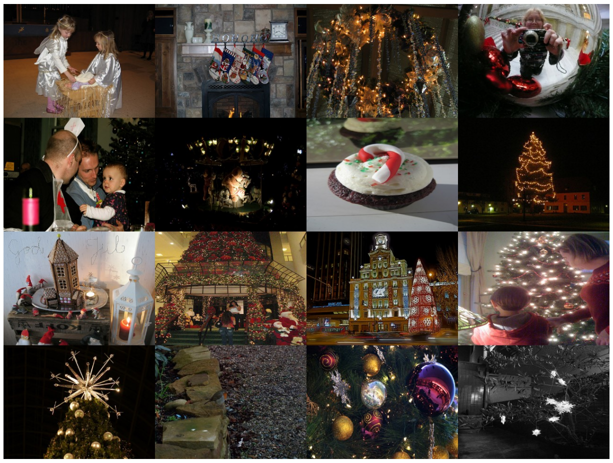

Figure 9: Sample results for the *Christmas* query for visual and tag model trained using cleaned data. The unsupervised approach produces good and varied results for events.