Harvesting Image Databases from the Web

F. Schroff
Dept. of Engineering Science
University of Oxford, UK

A. Criminisi
Microsoft Research Ltd.
Cambridge, UK

A. Zisserman
Dept. of Engineering Science
University of Oxford, UK

Abstract

The objective of this work[1] *is to automatically generate a large number of images for a specified object class (for example, penguin). A multi-modal approach employing both text, meta data and visual features is used to gather many, high-quality images from the web.*

Candidate images are obtained by a text based web search querying on the object identifier (the word penguin). The web pages and the images they contain are downloaded. The task is then to remove irrelevant images and re-rank the remainder. First, the images are re-ranked using a Bayes posterior estimator trained on the text surrounding the image and meta data features (such as the image alternative tag, image title tag, and image filename). No visual information is used at this stage. Second, the top-ranked images are used as (noisy) training data and a SVM visual classifier is learnt to improve the ranking further. The principal novelty is in combining text/meta-data and visual features in order to achieve a completely automatic ranking of the images.

Examples are given for a selection of animals (e.g. camels, sharks, penguins), vehicles (cars, airplanes, bikes) and other classes (guitar, wristwatch), totalling 18 classes. The results are assessed by precision/recall curves on ground truth annotated data and by comparison to previous approaches including those of Berg et al. *[5] (on an additional six classes) and Fergus* et al. *[9].*

1. Introduction

The availability of image databases has proved invaluable for training and testing object class models during the recent surge of interest in object recognition. However, producing such databases containing a large number of images and with high precision is still an arduous manual task. Image search engines apparently provide an effortless route, but currently are limited by poor precision of the returned images and restrictions on the total number of images provided. For example, with Google Image Search the precision is as low as 32% on one of the classes tested here (shark) and averages at 39%, and downloads are restricted to 1000 images.

Fergus *et al*. [9, 10], Lin *et al*. [16] and Li *et al*. [15] dealt with the precision problem by re-ranking the images downloaded from an image search. The method in [9] involved visual clustering of the images by using probabilistic Latent Semantic Analysis (pLSA) [12] over a visual vocabulary. [15] used a Hierarchical Dirichlet Process instead of pLSA. Lin *et al*. [16] re-ranked using the text on the original page from which the image was obtained. However, for all three methods the yield is limited by the restriction on the total number of images provided by the image search.

Berg *et al*. [5] overcome the download restriction by starting from a *web* search instead of an *image* search. This search can generate thousands of images. Their method then proceeds in two stages: first, topics are discovered based on words occurring on the web pages using Latent Dirichlet Allocation (LDA) [6] on *text* only. Image clusters for each topic are formed by selecting images where nearby text is top ranked by the topic. A user then partitions the clusters into positive and negative for the class. Second, images and the associated text from these clusters are used as exemplars to train a classifier based on voting on visual (shape, colour, texture) and text features. The classifier is then used to re-rank the downloaded dataset. Note, the user labelling of clusters avoids the problem of polysemy, as well as providing good training data for the classifier. The method succeeds in achieving a greater yield, but at the cost of manual intervention.

Our objective in this work is to harvest a large number of images of a particular class *automatically*, and to achieve this with high precision. Our motivation is to provide training databases so that a new object model can be learnt effortlessly. Following [5] we also use web search to obtain a large pool of images and the web pages that contain them. The low precision does not allow us to learn a class model from such images using vision alone. The challenge then is how best to combine text, meta-data and visual information in order to achieve the best image re-ranking.

The two main contributions are: first, we show in section 3 that meta-data and text attributes on the web page

[1]This work was supported in part by Microsoft Research through the European PhD Scholarship Programme and by EU project CLASS.

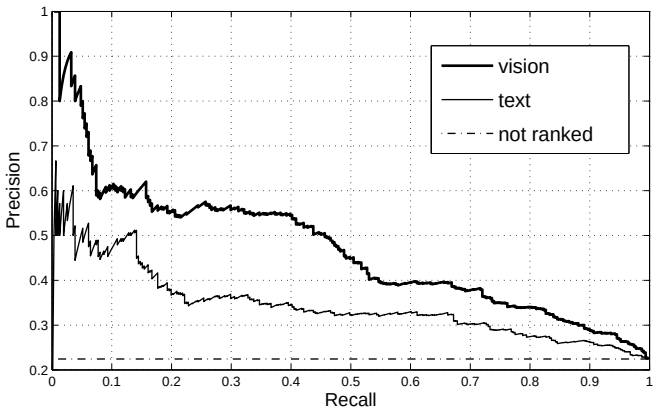

Figure 1. **Text & visual ranking vs. unranked-baseline:** precision recall plot for the text re-ranking, the visual ranking trained on the text ranking, and the unranked images, for the "shark" query.

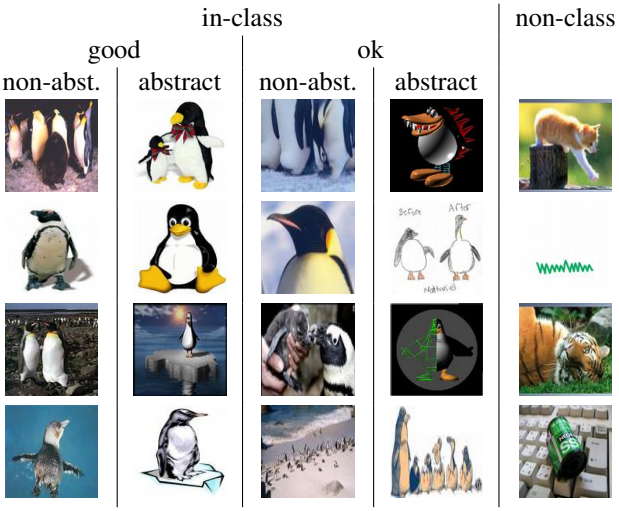

Figure 2. **Image annotations:** Example images corresponding to annotation categories for the class penguin.

containing the image provide a useful estimate of the probability that the image is in class, and thence can be used to successfully rank images in the downloaded pool. Second, we show in section 4 that this probability is sufficient to provide (noisy) training data for a visual classifier, and that this classifier delivers a superior re-ranking to that produced by text alone. Figure 1 visualises this two stage improvement over the initially downloaded images (not-ranked). The class independent text ranker significantly improves this baseline and is itself improved by quite a margin when the vision based ranker (trained on the text ranker results) is employed.

Others have used text and images together, however in a slightly different setting. For example, Barnard *et al*. [2] use ground truth annotated images as opposed to noisy annotation stemming from web pages, as in our case. Other work of Berg *et al*. [4] uses text from the Internet, but focused on identifying a specific class rather than general object classes.

We show in section 5 that our automatic method achieves superior ranking results to those produced by the method of Berg *et al*. [5] and also to that of Google Image Search.

2. The Databases

This section describes the methods for downloading the initial pool of images (together with associated meta-data) from the Internet, and the initial filtering that is applied. For the purposes of training classifiers and for assessing precision and recall the downloaded images are annotated manually for 18 classes: airplane (ap), beaver (bv), bikes (bk), boat (bt), camel (cm), car (cr), dolphin (dp), elephant (ep), giraffe (gf), guitar (gr), horse (hs), kangaroo (kg), motorbikes (mb), penguin (pg), shark (sk), tiger (tr), wristwatch (ww), zebra (zb).

Data collection. We compare three different approaches to downloading images from the web. The first approach, named `WebSearch`, submits the query word to Google web search and all images that are linked within the returned web pages are downloaded. Google limits the number of returned web pages to 1000, but many of the web pages contain multiple images, so in this manner thousands of images are obtained. The second approach, `ImageSearch`, starts from Google image search (rather than web search). Google image search limits the number of returned images to 1000, but here each of the returned images is treated as a "seed" – further images are downloaded from the web page from where the seed image originated. The third approach, `GoogleImages`, includes only the images directly returned by Google image search (a subset of those returned by `ImageSearch`). The query can consist of a single word or more specific descriptions such as "penguin animal" or "penguin OR penguins". Images smaller than 120×120 are discarded. In addition to the images, text surrounding the image HTML tag is downloaded together with other meta-data such as the image filename.

Ground truth annotation. In a similar manner to Fergus *et al*. [9], images are divided into three categories:

in-class-good: Images that contain one or many class instances in a clearly visible way (without major occlusion, lighting deterioration or background clutter and of sufficient size).

in-class-ok: Images that show parts of a class instance, or obfuscated views of the object due to lighting, clutter, occlusion and the like.

non-class: Images not belonging to *in-class*.

The good and ok sets are further divided into two subclasses:

abstract: Images that don't look like realistic natural images (*e.g*. drawings, non realistic paintings, comics, casts or statues).

non-abstract: Images not belonging to the previous class.

Example annotations for the class penguin are shown in

Service	in-class	non-class	precision
WebSearch	8773	25252	26%
ImageSearch	5963	135432	4%
GoogleImages	4416	6766	39%

Table 1. **Statistics by source**: The statistics of downloaded images for different retrieval techniques.

figure 2. As is usual in annotation there are ambiguous cases, *e.g.* deciding when occlusion is sufficiently severe to classify as *ok* rather than *good*, or when the objects are too small. Note, the *abstract* vs. *non-abstract* categorisation is not general but is suitable for the object classes we consider in this paper. For example, it would not be useful if the class of interest was "graph" or "statue" or a similar more abstract category.

Table 1 details the statistics for each of the three retrieval techniques (WebSearch, ImageSearch and GoogleImages). Note that some images are common between the methods. ImageSearch gives a very low precision (only about 4%) and is not used for the harvesting experiments. Only WebSearch and GoogleImages are used, and their images are merged into one dataset per object class. Table 2 lists the 18 categories downloaded and the corresponding statistics for *in-class* and *non-class* images. The overall precision of the images downloaded for all 18 classes is about 29%.

Due to the great diversity of images available on the Internet and because of how we retrieve the images, it is difficult to make general observations on how these databases look. However, it is clear that polysemy affects the returned images. Interestingly, this is not a problem that could be predicted directly from the English word, since most of the classes we search for don't have direct polysemous meanings, i.e. they are not polysemous in the sense of "bank" (as in place to get money, or river bank) for example. It is rather that the words correspond to brands or product names ("leopard tank") or team names (the NHL ice hockey team "San Jose Sharks") or are used as attributes ("tiger shark"). Apart from that, the *in-class* images occur in almost all variations imaginable, as sharks crashed into houses or other oddities. Even though context [22] can clearly be important in re-ranking the images (*e.g.* camel and kangaroo in desert-like images), it will have its limitations due to the variety of occurrences of the object.

2.1. Removing Drawings & Symbolic Images

Since we are mostly interested in building databases for natural image recognition, we ideally would like to remove all *abstract* images from the downloaded images. However, separating *abstract* images from all others automatically is very challenging for classifiers based on visual features. Instead we tackle the easier visual task of removing drawings & symbolic images. These include: comics, graphs, plots, maps, charts, drawings and sketches, where the images can be fairly simply characterised by their visual features (see

	downloaded images			after filtering		
Class	in-cl.	non-cl.	prec.	in-cl.	non-cl.	prec.
airplane (ap)	843	1720	32.89%	531	975	35.26%
beaver (bv)	201	3156	5.99%	149	2126	6.55%
bikes (bk)	1236	1963	38.64%	909	1057	46.24%
boat (bt)	861	2170	28.41%	726	1310	35.66%
camel (cm)	492	1910	20.48%	393	1329	22.82%
car (cr)	1125	1045	51.84%	949	554	63.14%
dolphin (dp)	663	1544	30.04%	448	917	32.82%
elephant (ep)	660	1835	26.45%	548	1175	31.80%
giraffe (gf)	779	1433	35.22%	641	792	44.73%
guitar (gr)	1261	1993	38.75%	903	821	52.38%
horse (hs)	963	1986	32.66%	809	1143	41.44%
kangaroo (kg)	295	1886	13.53%	240	1144	17.34%
motorbikes (mb)	704	981	41.78%	571	593	49.05%
penguin (pg)	664	1484	30.91%	391	784	33.28%
shark (sk)	522	1771	22.76%	311	1075	22.44%
tiger (tr)	333	2114	13.61%	275	1262	17.89%
wristwatch (ww)	916	982	48.26%	658	478	57.92%
zebra (wb)	427	1718	19.91%	351	985	26.27%
total	12945	31691	**29.00%**	9803	18520	**34.61%**

Table 2. **Image class statistics** of the original downloaded images using WebSearch&GoogleImages only, and after applying the drawing&symbolic images removal filter.

below). Example images are shown in figure 3. Their removal significantly reduces the number of *non-class* images improving the resulting precision of the object class datasets as shown in table 2 (overall precision goes from 29% to 35%). Filtering out such images also has the aim of removing this type of *abstract* image from the *in-class* images.

Learning the filter. We train a radial basis function Support Vector Machine (SVM) on a hand labelled dataset (examples in figure 3). After the initial training no further user interaction is required. In order to obtain this dataset, images were downloaded using ImageSearch with one level of recursion (*i.e.* web pages linked from "seed" web pages are also used) with queries such as "sketch" or "drawing" or "draft". The goal was to retrieve many images and then select suitable training images manually. The resulting dataset consists of approximately 1400 drawings&symbolic images, and 2000 non drawings&symbolic images.

Three simple visual only features are used: (i) a colour-histogram; (ii) a histogram of the L2-norm of the gradient; (iii) a histogram of the angles $(0 \ldots \pi)$ weighted by the L2-norm of the corresponding gradient. In all cases 1000 equally spaced bins are used. The motivation behind this choice of features is that drawings&symbolic images are characterised by sharp edges in certain orientations and or a distinctive colour distribution (*e.g.* only few colours in large

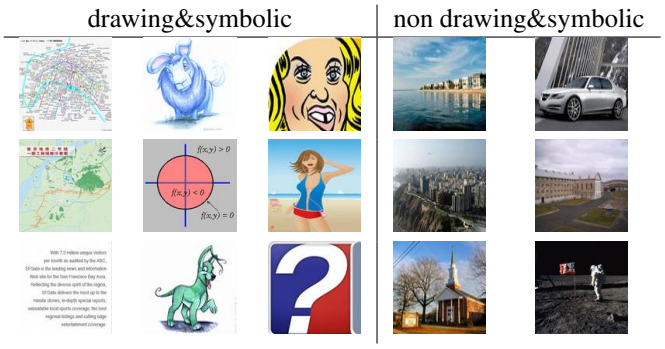

Figure 3. **Drawings&symbolic images:** Examples of positive and negative training images.

areas). The method achieves around 90% classification accuracy on the drawings&symbolic images database (using two-fold cross-validation).

This classifier is applied to the entire downloaded image dataset to filter out drawing&symbolic images, before further processing. The total number of images that are removed for each class is shown in table 2. In total 42% of *non-class* images are removed over all classes. The remaining images are those used in our experiments. As well as successfully removing *non-class* images, the filter also succeeds in removing an average of 60% (123 images) *in-class abstract* images, with a range between 45% (for motorbikes, 40 images) and 85% (for wristwatch, 11 images). There is some loss of the desired *in-class non-abstract* images, with on average 13% (90 images) removed, though particular classes lose a relatively high percentage (28% for shark and wristwatch). Even though this seems to be a high loss the precision of the resulting datasets is improved in all cases except for the class shark.

3. Ranking on Textual Features

We now describe the re-ranking of the returned images based on text and meta-data alone. Here we follow and extend the method proposed by Frankel *et al*. [11] in using a set of textual attributes whose presence is a strong indication of the image content.

Textual features. We use seven features from the text and HTML-tags on the web page: *contextR, context10, filedir, filename, imagealt, imagetitle, websitetitle*.

Filedir, *filename* and *websitetitle* are self-explanatory. *Context10* includes the ten words on either side of the image-link. *ContextR* describes the words on the web-page between eleven and 50 words away from the image-link. *Imagealt* and *imagetitle* refer to the "alt" and "title" tag of the image-tag. The features are intended to be conditionally independent, given the image content (we address this independence below). It is difficult to compare directly with the features in [11], since no precise definition of the features actually used is given.

Context here is defined by the HTML source, not by the rendered page since the latter depends on screen resolution and browser type and is an expensive operation. In the text processing a standard stop list [20] and the Porter stemmer [21] are used. In addition HTML-tags and domain specific stop words (such as "html" or " ") are ignored.

We also experimented with a number of other features, such as the image MIME type ('gif', 'jpeg' *etc*.), but found that they did not help discrimination.

Image ranking. Using these seven textual features, the goal is to re-rank the retrieved images. Each feature is treated as binary: "True" if it contains the query word (*e.g*. penguin) and "False" otherwise. The seven features define a binary feature vector for each image $\mathbf{a} = (a_1, \ldots, a_7)$, and the ranking is then based on the posterior probability, $P(y = \textit{in-class}|\mathbf{a})$, of the image being *in-class*, where $y \in \{\textit{in-class}, \textit{non-class}\}$ is the class-label of an image. We learn a class *independent* ranker in order to re-rank the images based on those seven features; *i.e*. the ranker is not learnt or tuned for each class separately, but is learnt once and can then be applied to any new class.

A simple Bayesian posterior estimation $P(y|\mathbf{a}) = P(\mathbf{a}|y)P(y)/P(\mathbf{a})$ is used, with

$$P(\mathbf{a}|y) = P(a_1, \ldots, a_4|y) \prod_{5}^{7} P(a_i|y) \qquad (1)$$

where $P(a_1, \ldots, a_4|y)$ is the joint probability of the first four textual features (*contextR, context10, filedir, filename*). This choice resulted from a comparison of several different factorisations of the likelihood $P(\mathbf{a}|y)$, including naïve Bayes [8] $P(\mathbf{a}|y) = \prod_{i=1}^{7} P(a_i|y)$. The mixed model (1) gave slightly better performance than other factorisations, and reflects the fact that the first four features are less independent from each other than the remaining three.

3.1. Text re-ranking results

Images are ranked using the text based posterior $P(y|\mathbf{a})$ to provide an ordering. We assess the performance by reporting precision at various points of recall as well as average precision. To re-rank images for one particular class (*e.g*. penguin) we do not employ the ground truth data for that class. Instead, we train the Bayes classifier (specifically we learn $P(\mathbf{a}|y)$, $P(y)$ and $P(\mathbf{a})$) using all available annotations, *except* the class we want to re-rank. This way we evaluate performance as a *completely automatic class independent* image ranker; *i.e*. for any new and unknown class, the images can be re-ranked without *ever* using labelled ground truth knowledge of that class.

Figure 4 shows the precision-recall curves for selected classes and table 3 (text) gives the precision at 15% recall for all classes. It can clearly be seen that precision is highly increased at the lower recall levels, compared to the average precision of table 2.

A separate set of experiments was carried out to measure how the performance of the text ranker varies with the

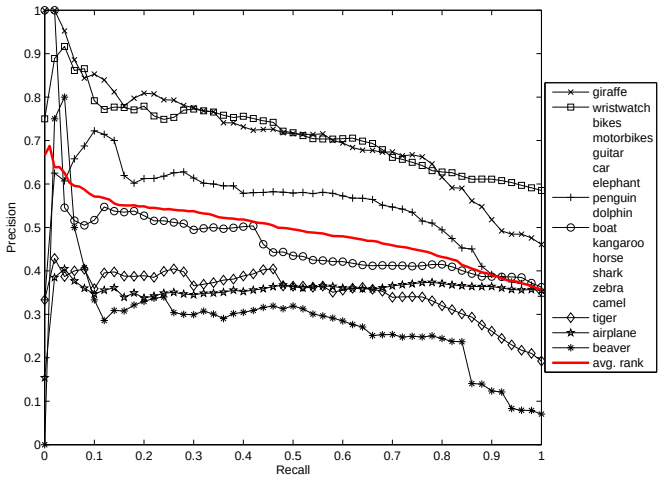

Figure 4. **Text based re-ranking:** precision vs. recall estimated for each class with *abstract* images considered *in-class*. The labels are shown in decreasing order of precision at 15% recall. The recall precision curves are only shown for selected classes for clarity. The average over *all* 18 classes is also shown.

number and choice of classes used for training. Ideally we would like to compare $P(\mathbf{a}|y)$, $P(y)$ and $P(\mathbf{a})$ learnt using different numbers of training classes. However, given our goal of ranking images we instead compare these probabilities indirectly by assessing precision at 15% recall. We find that the performance is almost unaltered by the choice of training classes provided more than five classes (chosen randomly) are used for training.

Discussion. As can be seen from figure 4 our text-re-ranker performs well on average, and significantly improves the precision up to quite a high recall level. In section 4 we will show that this is sufficient to train a visual classifier. For some classes the text ranker performs very well (*e.g.* wristwatch, giraffe) for others it performs rather poorly (*e.g.* airplane, beaver, camel, tiger). Visual inspection of the highly ranked "outlier" (*non-class*) images in the text ranked lists gives some explanation for these performances. Classes that perform well (wristwatch, giraffe) generally have outliers that are unrelated to each other. In contrast for the classes that perform poorly the outlier images are related and result from lack of discriminativity of the query word – for example for airplanes there are images of airplane food, airports, toy airplanes, paper airplanes, airplane interiors, and advertisements with comic airplanes. Other classes suffer from the type of polysemy described in section 2: for camels there are brand and cigarette related outliers; and for tiger there is the attribute problem with images of helicopters, tanks, fish (sharks), boxing, golf, stones, and butterflies. Despite these problems we are able to train a good visual classifier for most classes (see next section), with the main exception of beaver.

4. Ranking on Visual Features

The text re-ranking of section 3 associates a posterior probability with each image as to whether it contains the query class or not. The problem we are now faced with is how to use this information to train a visual classifier which would improve the ranking further. The problem is one of training from noisy data: we need to decide which images to use for positive and negative training data and how to select a validation set in order to optimise the parameters of the classifier.

We first describe the visual features used, and then how the classifier is trained.

Visual features. We follow the approach of [9] and use a variety of region detectors with a common visual vocabulary. All images are first resized to 300 pixels in width. Regions are detected using: difference of Gaussians, Multiscale-Harris [18], Kadir's saliency operator [14], and points sampled from Canny edge points. Each image region is represented as a 72 dimensional SIFT [17] descriptor. A separate vocabulary consisting of 100 visual words is learnt for each detector using k-means, and these vocabularies are then combined into a single one of 400 words. Finally, the descriptor of each region is assigned to the vocabulary. The software for the detectors is obtained from [23]. Fuller implementation details are given in [9], and are reproduced in our implementation.

Text based training of a visual classifier. At this point we can select n_+ positive training images from the top of the text ranked list, or those that have a posterior probability above some threshold, but a subset of these positive images will be "noisy", i.e. will not be *in-class*. Table 3 (text) gives an idea of the noise from the proportion of outliers. It averages at 45% if $n_+ = 100$. However, we can assume that the *non-class* images are *not* visually consistent – an assumption verified to some extent by the results in section 4.1. The case of negative images is more favourable: we select n_- images at random from all downloaded images (*i.e.* from all 18 classes, tens of thousands of images) and the chances of any image being of a particular class is very low. We did not choose to select the n_- images from the low ranked images of the text-ranker output, because the probability of finding *in-class* images there is higher than finding them in the set of *all* downloaded images.

Given this situation we choose to use a SVM classifier, since it has the potential to train despite noise in the data. The SVM training minimizes the following sum [19]:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_+ \sum_{i:y_i=1} \xi_i + C_- \sum_{j:y_j=-1} \xi_j \quad (2)$$

$$\text{subject to} \quad y_l \left(\mathbf{w}^T \Phi(\mathbf{x}_l) + b\right) \geq 1 - \xi_l, \quad (3)$$

$$\xi_l \geq 0, l = 1, \ldots, (n_+ + n_-)\,. \quad (4)$$

Where $\mathbf{x}_l$ are the training vectors and $y_l \in \{1, -1\}$ the class

1. Download images and meta-data for new class (*e.g.* "lion") using `WebSearch` &`GoogleImages` (section 2).
2. Filter images: remove drawings&symbolic images (section 2.1).
3. Rank images based on text-attributes using the Bayes classifier (section 3).
4. Train visual SVM classifier on text-ranked images (section 4).
5. Rank all images from 1. using the visual classifier.

Figure 5. **Overview of the text+vision (t+v) image harvesting algorithm.**

labels. C_+ and C_- are the false classification penalties for the positive and negative images, with $\boldsymbol{\xi}$ being the corresponding slack-variables.

To implement the SVM we use the publicly-available SVM$^{\text{light}}$ software [13] (with the option to remove inconsistent training data enabled). Given two input images I_i and I_j and their corresponding normalized histograms of visual words S_i and S_j this implementation uses the following χ^2, radial basis function (RBF) kernel: $K(S_i, S_j) = exp(-\gamma \cdot \chi^2(S_i, S_j))$ [24]. With γ the free kernel parameter.

Thus, γ, C_+ and C_- are the three parameters that can be varied. The optimal value for these parameters is obtained by training the SVM using ten-fold cross validation. We require a performance measure for the cross-validation and use precision at 15% recall – the SVM re-ranks the validation images and we measure precision on this. Note, we do not use the ground truth at any stage of training.

Sometimes C_+ and C_- are used to correct unbalanced training data [19]. In our case, however, the SVM is very sensitive to these parameters, probably due to the huge amount of noise in the data, and the optimal value does not directly correspond to the ratio of positive to negative images.

Finally, the trained SVM is used to re-rank the filtered image set based on the SVM classification score. The entire image harvesting algorithm is summarised in figure 5.

4.1. Results for textual/visual image ranking

In this section we evaluate different combinations of training and testing. If not stated otherwise the text+vision system of figure 5 was used. Results are given in table 3 for various choices of n_+ and n_-. For each choice, five different random selections are made for the sets used in the ten-fold cross-validation, and mean and standard deviation are reported. The clear improvement brought by the visual classifier over the text based ranking for most classes is obvious. Figure 6 shows example high ranked images using text+vision.

We first investigate how the classification performance is affected by the choice of n_+ and n_-. It can be seen that increasing the n_- tends to improve performance. It is, however, difficult to select optimal values for n_+ and n_- since these numbers are very class dependent.

We next determine how much the performance is affected by the noise in the training data by training the SVM on ground truth positive data, i.e. instead of selecting n_+ images from the text ranked images, we select n_+ *in-class* images using the ground truth labelling. We find that the ground truth performs, as to be expected, better for most classes and similar or even worse on those classes where the text+vision system performs well.

As a baseline comparison, we investigate performance if no text re-ranking is used, but the n_+ images are sampled uniformly from the filtered images. If the text re-ranking works well, and hence provides good training data, then text+vision improves over the baseline. In cases where the text ranking does not perform well the baseline can even outperform text+vision. This is due to the fact, that bad text ranking can provide visually consistent training data, that does *not* show the expected class (*e.g.* for airplanes it contains many images showing: airplane food, inside airplanes/airports, taken out of the window of an airplane). However, the uniformly sampled images still consist of about 35% *in-class* images (table 2) and the n_- are very unlikely to contain *in-class* images. Thus, the SVM based classifier is shown to be very noise insensitive and well suited for this task.

Discussion. We also investigated two alternative visual classifiers, pLSA (as used as a visual classifier by Fergus *et al.* [9]) and feature selection. Both showed inferior performance to the SVM. The pLSA appeared to suffer from the high variability in the appearance of in-class images and the high levels of training noise. For feature selection our intention was to find discriminative visual words and then use these in a Bayesian framework. The discriminative visual words were obtained based on the likelihood ratio of a visual word occurring in the foreground to background images [7]. However, probably due to the large variation in both the foreground and background images, together with the noisy training data, we weren't able to match the performance of the SVM ranker.

We found that combining the vision and text ranked lists using Borda count [1] or similar methods gave a slight improvement on average, but results were very class dependent.

5. Comparison with other approaches

We compare our algorithm with three other approaches. In all cases results are given for $n_+ = 250, n_- = 1000$, as this is the most stable setting (see table 3). Again we report mean and standard deviation over five runs of ten-fold cross-validation.

Comparison with Google image search. Here we re-rank the images downloaded with `GoogleImages` in the fully automatic fashion described earlier. Comparative results between our approach and `GoogleImages`

prec. 15%	ap	bv	bk	bt	cm	cr	dp	ep	gf	gr	hs	kg	mb	pg	sk	tr	ww	zb	avg.	std
text	36.9	24.5	69.8	53.9	46.0	63.3	72.2	66.0	77.7	62.7	57.4	47.8	68.5	62.2	41.7	39.2	79.2	40.7	56.1	
t+v (250/250)	54.0	36.4	64.8	60.9	53.5	78.5	61.9	77.5	86.5	65.4	60.6	53.5	78.2	55.5	56.8	45.7	81.6	96.8	64.9	9.3 (2.5)
t+v (150/500)	59.1	33.8	69.7	63.4	56.8	91.8	61.6	70.7	83.2	67.3	67.0	50.0	77.4	70.4	69.3	59.6	86.3	85.5	67.9	8.3 (2.0)
t+v (250/500)	58.6	35.0	64.6	65.1	47.8	79.0	60.1	75.4	86.9	51.7	68.4	50.2	77.7	69.7	64.3	39.8	90.4	87.9	65.2	8.8 (2.0)
t+v (150/1000)	61.4	30.6	65.5	61.3	53.8	92.0	67.7	85.8	86.6	66.1	65.7	61.6	82.2	72.4	64.3	46.9	91.8	80.5	68.7	8.1 (2.1)
t+v (250/1000)	63.5	32.3	65.9	62.4	51.6	93.0	61.7	80.2	87.8	62.6	71.2	45.5	84.6	69.6	64.9	53.0	86.6	95.8	68.4	7.9 (1.9)
gt (250/1000)	86.4	–	80.4	73.3	77.7	96.7	63.5	96.1	83.5	73.0	82.6	83.6	79.8	82.8	80.6	87.1	91.2	96.7	83.2	9.3 (2.6)
Basel. (250/1000)	64.2	12.5	65.1	58.0	46.9	76.7	55.6	53.3	86.7	61.1	66.2	36.1	71.3	60.4	56.3	56.7	87.0	81.4	60.9	10.2 (2.6)

Table 3. **Comparison of precision at 15% recall**: 'text' refers to text re-ranking alone; 't+v' is text+vision re-ranking using different training ratios n_+/n_-; 'gt' is ground truth (only positive images) training of the visual classifier; and 'Basel.' is the baseline, where the visual classifier is trained on $n_+ = 250$ images uniformly sampled from the filtered images of one class, instead of the text re-ranked images, and $n_- = 500$ background images as before. The second last column (avg.) gives the average over all classes. The last column states the mean of the classwise standard deviations over five runs of cross-validation, as well as the standard deviation of the means over all classes, in parantheses.

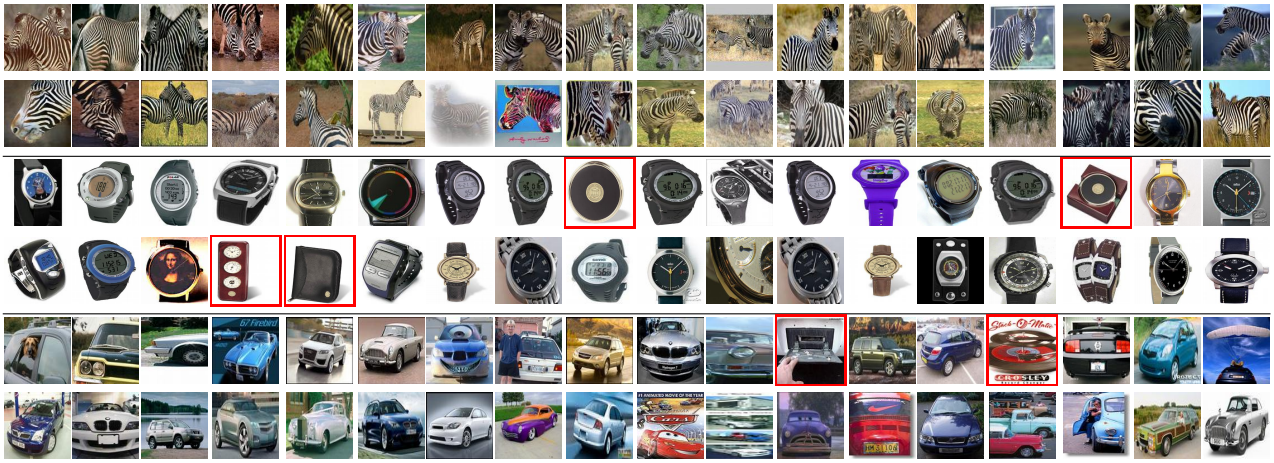

Figure 6. **Top ranked 36 images** of zebra, wristwatch and car using the text+vision algorithm of figure 5. Red boxes indicate false positives.

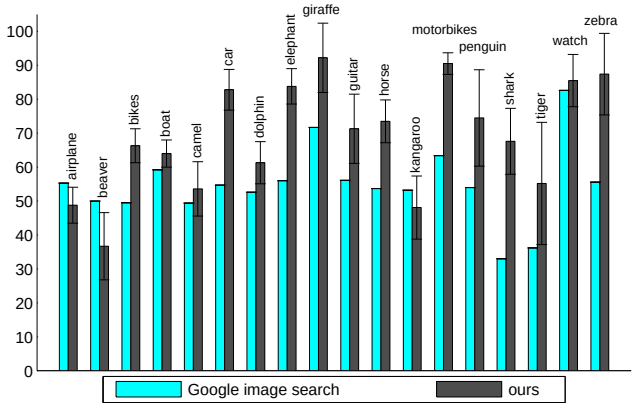

Figure 7. **Comparison with Google image search.** Precision at 100 image recall.

are shown in figure 7. As can be observed, our approach achieves higher average precision for 15 out of 18 classes with only airplane, beaver and kangaroo being outperformed by the Google results. Those are cases where our text ranker doesn't perform that well, which increases the noise in the training data and thus explains the decreased visual performance.

	airplane	guitar	leopard	motorbike	wristwatch
our	45 ± 5	72 ± 11	72 ± 6	81 ± 9	97 ± 4
our ($\not\oslash$k)	35 ± 4	29 ± 4	50 ± 5	63 ± 8	93 ± 7
[9] ($\not\oslash$k)	57	50	59	71	88
Google ($\not\oslash$k)	50	30	41	46	70

Table 4. **Comparison with Fergus *et al.* [9]**: Average precision at 15% recall with one standard deviation. The images are from Google image search and were provided by the author. ($\not\oslash$k) uses the same annotation as [9]. The first row of the table treats Fergus' *ok* class as *in-class*, unlike [9].

Comparison with Fergus *et al.* [9]. In this experiment we re-rank the Google images provided by [9]. It is difficult to directly compare results in [9] to the text+vision algorithm, as [9] treats *ok* images as non-class, whereas our system is not tuned to distinguish *good* from *ok* images. Due to this our system performs slightly worse than [9], when measured only on *good* images. However, it still outperforms Google image search on most classes even in this case. Table 4 also shows (first row) the results when *ok* images from the [9] data are treated as in-class. As expected the performance increases significantly for all classes.

Comparison with Berg *et al.* [5]. Here we run our visual ranking system on the dataset provided by [5]. In order to do so we downloaded an additional set of six classes

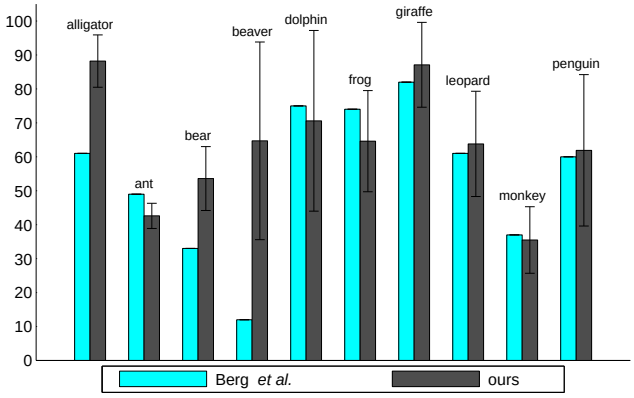

Figure 8. **Comparison with Berg *et al.* [5].** Precision at 100-image recall level for the 10 animal classes made available by the authors of [5]. Note that our automatic algorithm is superior in many cases, even though the method of [5] involves manual intervention.

(alligator, ant, bear, frog, leopard, monkey) for which no manual annotation was obtained. Figure 8 compares the results reported in [5] to re-ranking of the test images available from [3] using our visual classifier. Note that we are training on our set of images which might stem from a different distribution than the Berg test set. We compare with the "classification on test data" category of [5], not to their "final dataset" which includes ground truth from their manual step. Their provided ground truth, which treats *abstract* images as *non-class*, was used. Note that our automatic algorithm produces results comparable or superior to those of Berg *et al.*, although their algorithm requires manual intervention.

6. Conclusion

This paper has proposed an *automatic* algorithm for harvesting the web and gathering hundreds of images of a given query class. Thorough quantitative evaluation has shown that the proposed algorithm performs similarly to state of the art systems such as [9], while outperforming both the widely used Google Image Search and recent techniques which rely on manual intervention [5].

The algorithm does not rely on the high precision of top returned images *e.g.* from Google Image Search. Such images play a crucial role in [9, 15], and future work could take advantage of this precision.

Polysemy, a problem with no automatic solution (currently), does affect our results. However, this paper improves our understanding of the problem in its different forms.

References

[1] J. Aslam and M. Montague. Models for metasearch. In *SIGIR*, pages 276–284, New York, NY, USA, 2001. ACM Press.

[2] K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. Jordan. Matching words and pictures. *J. Machine Learning Research*, 3:1107–1135, Feb 2003.

[3] T. Berg. Animals on the web dataset. http://www.cs.berkeley.edu/ millert/animalDataset/.

[4] T. Berg, A. Berg, J. Edwards, M. Mair, R. White, Y. Teh, E. Learned-Miller, and D. Forsyth. Names and Faces in the News. In *Proc. CVPR*, 2004.

[5] T. L. Berg and D. A. Forsyth. Animals on the web. In *Proc. CVPR*, 2006.

[6] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *J. Machine Learning Research*, 3:993–1022, Jan 2003.

[7] G. Dorkó and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proc. ICCV*, 2003.

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2001.

[9] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *Proc. ICCV*, 2005.

[10] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *Proc. ECCV*. Springer-Verlag, May 2004.

[11] C. Frankel, M. J. Swain, and Athitsos V. Webseer: An image search engine for the world wide web. In *Proc. CVPR*, 1997.

[12] T Hofmann. Probabilistic latent semantic analysis. In *UAI*, 1999.

[13] T. Joachims. SVM$^{\text{light}}$. http://svmlight.joachims.org/.

[14] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proc. ECCV*. Springer-Verlag, May 2004.

[15] J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In *Proc. CVPR*, 2007.

[16] W.-H. Lin, R. Jin, and A. Hauptmann. Web Image Retrieval Re-Ranking with Relevance Model. In *ICWI*, 2003.

[17] D. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, pages 1150–1157, Sep 1999.

[18] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Proc. ECCV*. Springer-Verlag, May 2004.

[19] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. In *ICML*, 1999.

[20] Onix. ONIX Text Retrieval Toolkit. http://www.lextek.com/manuals/onix/stopwords1.html.

[21] M. Porter, R. Boulton, and A. Macfarlane. The English (Porter2) stemming algorithm.

[22] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):153–167, 2003.

[23] VGG. Affine Covariant Features. http://www.robots.ox.ac.uk/ vgg/research/affine/index.html.

[24] J. Zhang, M. Marszalek, S. Lazebnik, and Schmid C. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 2007.