NTT-MIT Variable Viewpoint Reality Tech Report

Please do not distribute outside of NTT

Boosting Image Retrieval

Kinh Tieu and Paul Viola
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We present an approach for image retrieval using a very large number of highly selective features and efficient online learning. Our approach is predicated on the assumption that each image is generated by a sparse set of visual "causes" and that images which are visually similar share causes. We propose a mechanism for computing a very large number of highly selective features which capture some aspects of this causal structure (in our implementation there are over 45,000 highly selective features). At query time a user selects a few example images, and a technique known as "Boosting" is used to learn a classification function in this feature space. By construction, the boosting procedure learns a simple classifier which only relies upon 20 of the features. As a result a very large database of images can be scanned rapidly, perhaps 1,000,000 images per second. Finally we will describe a set of experiments performed using our retrieval system on a database of 3000 images.

1 Introduction

In the image retrieval task a user must search a database of many thousands, or millions, of images. User goals vary, in some cases the task is to find a particular image, in other cases any image from a class will do. The optimal interface would provide a very flexible query mechanism, perhaps through a natural language interface. In fact, many "stock photo houses" currently provide such an interface to their collections. Advertisers and publishers present a description of their requirements: "an image of the beach with athletic people playing volleyball". Human clerks then scan many images by hand using keywords.

Recently a large number of automated image retrieval systems have appeared [6, 10, 13, 8]. Rather than describe an image using text, in these systems an image query is described using a set of example images. In some of these systems a user's only interaction with the retrieval engine is through example images, in others the user is also asked to weight a set of "intuitive" features, such as color, texture and shape.

Image retrieval differs from the more common task of classification which includes tasks such as face detection and character recognition. In retrieval the number of potential image classes is extremely large and the number of example images is very small. For example, a user may wish to retrieve example images of "cars on the road" using perhaps three example images. Conventional machine learning methods, such as neural networks or support vector machines, are not well suited to this task because they often require a small number of classes and a large set of labeled data (see [15, 12] for example).

An effective solution to this problem hinges on the discovery of a simplifying structure in the distribution of images. A learning algorithm can then take advantage of this structure to learn an image class from a small number of examples.

When a human clerk is shown three example images containing "a car on the road", he concludes that other images must contain both "car" and "road". A photograph chosen at random from the Web might contain a "car", a "road", the "Eiffel Tower", the "Taj Mahal", or any one of a thousand other objects. But, while there are a very large number of objects which might be present in any one image, any particular image will contain at most a few of these objects. This is not unlike the structure of English text: there are over 100,000 possible English words, but any given

sentence will contain roughly 6 words. The clerk's actions in the above example are justified because the probability of a car and a road appearing by random chance in all three images is quite low.

The distribution of natural images is simplified by the fact that the objects which cause images are rare. In other words the causal structure of images is sparse.

Placed in this context, previous feature based retrieval approaches face a daunting task. There are usually just a few types of features used in such schemes, such as color, and oriented edges. These features are likely to appear in a large percentage of images. Since both the Eiffel Tower and the Taj Mahal have vertical edges, these features clearly cut across the boundaries of the causal structure. Learning the concept of "Eiffel Tower" from example images using these features will require the learning algorithm to stake out a complex region in this feature space. Many systems attempt to learn conjunctive concepts, such as a histogram. Stated simply a histogram encodes the relative frequencies of primitive properties (e.g., there are 1.3 times as many vertical edges than there are horizontal edges). Since it is likely that the background of an image will also contain vertical edges, these ratios are sensitive to changes in background.

In contrast we will define a very large set of highly selective visual features. A highly selective feature will respond to only a small percentage of images in the database – such a feature might return a large numerical value for only 5% of images. One could not hope to define such a large set of features by hand, instead an algorithm for automatically generating plausible features is given. Because these features are so rare, they are also very unlikely to occur at random in the background of an image.

Given a set of highly selective features query learning can be greatly simplified. Only a few features will respond to the set of example images. A learning algorithm which can rapidly select a set of 20 to 50 features which distinguishes these images is presented. The algorithm is an adaptation of "AdaBoost"[7]. After query learning, each image in the database can be evaluated rapidly by examining only 20 to 50 features. As a result over one million images can be scanned per second.

2 Creating Highly Selective Features

Highly selective features are a natural extension of the simple features used in other image database systems. Given a set of "first order" features such as oriented edges or color (see Figure 1), highly selective features measure how these first order features are geometrically related. By finding arrangements of first order features, a set of second order features can be defined. Arrangements of second order features form third order features.

The process starts out by first extracting a feature map for each type of simple feature (there are 25 simple linear features including "oriented edges", "center surround" and "bar" filters shown in Figure 1). Each features map is then rectified and down-sampled by two. The 25 feature maps are then used as the input to another round of feature extraction (yielding 25 x $25 = 625$ feature maps). The process is repeated again to yield 15,625 feature maps (over the red, green, and blue color channels, this yields 46,875 feature maps). Three levels of filtering were possible for the resolution of our images. Finally each feature map is summed to yield a single feature value.

Each level of processing discovers arrangements of features in the previous level. Thus a second order feature might be sensitive to diagonal arrangements of horizontal features – a feature visible as a staircase pattern. One example feature is shown in Figure 2. It might be called a tiger stripe feature. The first low-pass filter smoothes the image and removes high-frequency noise. The second order feature finds vertical edges. The third order feature detects a horizontal arrangement of these vertical edges. The feature map demonstrates the selectivity of a particular feature on the image of a tiger and a waterfall. Notice that the response in the final feature map is peaked over the tiger's stripes, while there is no discernible peak in the waterfall image. Figure 3 shows another filtering sequence that is more difficult to explain intuitively. Nevertheless it is selective for images of churchs and responds very weakly to the image of the field.

More formally these features are computed from an image as:

$$g_{i,j,k,c} = \sum_{pixels} M_{i,j,k,c} \tag{1}$$

where M is the feature map, i, j and k are indices over primitive features, and c indexes the different color channels of the image. The definition of M is:

$$M_i = \downarrow_2 (|f_i \otimes X|) \tag{2}$$

$$M_{i,j} = \downarrow_2 (|f_j \otimes M_i|) \tag{3}$$

$$M_{i,j,k} = \downarrow_2 (|f_k \otimes M_{i,j}|) \tag{4}$$

where X is the image, f_i is the ith filter and $\downarrow_2$ is the down-sample by two operation. Because the feature maps are down-sampled before the next level of filtering, the support of the filters over the image plane are effectively enlarged. This enables the features to cap-

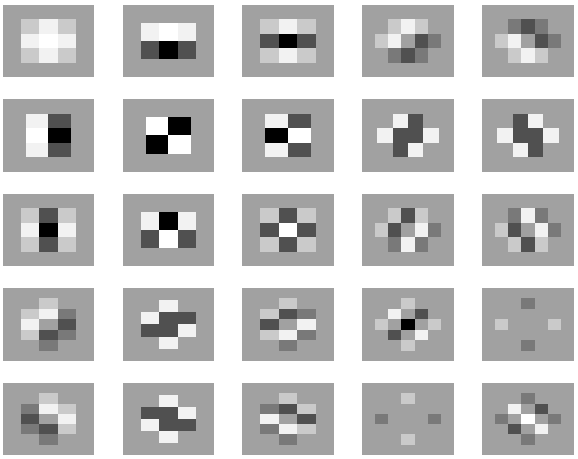

Figure 1: The 25 primitive filters used in computing the feature maps. In practice, these are efficiently computed with separable horizontal and vertical convolutions.

ture complex arrangements of low-level features (i.e. global structure).

We conjecture that these features do in fact reflect some of the sparse causal structure of the image formation process. One piece of evidence which supports this conclusion is the statistical distribution of the highly selective feature values. Evaluated across an image database containing 3000 images, these features are very sparse. The average kurtosis is approximately 8 and some of the features have a kurtosis as high as 120 (the Gaussian has a kurtosis of 3). Observing this type of distribution in a filter is extremely unusual and hence highly meaningful. It is well known that the response of certain linear filters (such as Laplacian or wavelet filters) are somewhat sparse and have higher kurtosis than Gaussian [1, 17]. The sparse response of the highly selective features is much more significant. Recall that each highly selective feature response is a summation across the entire image, and that the sum of a number of independent random variables tends quickly toward Gaussian. The pixel-wise kurtosis of the feature maps, before this summation, can be as high as 304 [1] . In contrast, the distribution of the summation of a rectified Laplacian filter across the image database is Gaussian.

[1] It is not unusual to observe high kurtosis in the distribution of a non-linear feature. For example one could easily square a variable with Gaussian distribution in order to yield a higher kurtosis. The highly selective features do *not* contain these sorts of non-linearities. At each level only the absolute value of the feature map is computed.

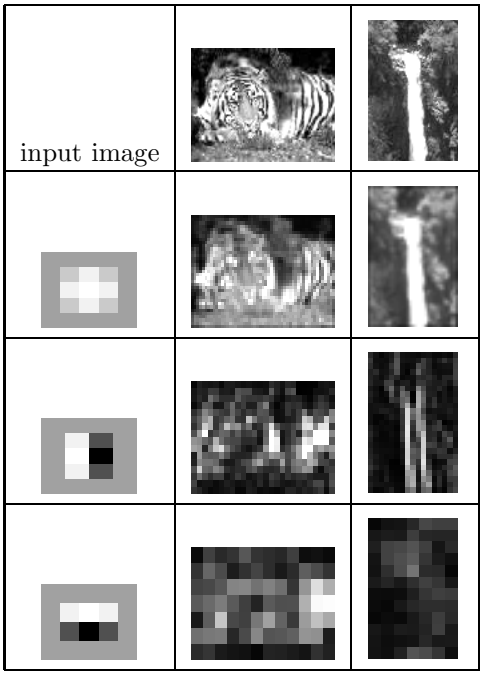

Figure 2: Responses of an image of a tiger and a waterfall to a particular filter sequence. The final feature map has a strong peak at the arrangement of the stripes on the body of the tiger, whereas there is a weak response to the waterfall image.

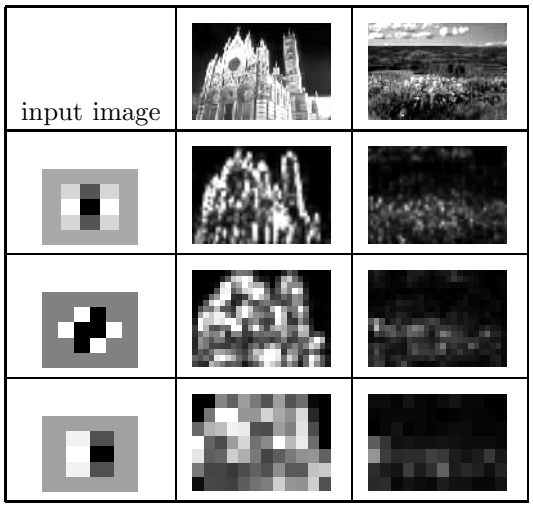

Figure 3: Responses of an image of a church and a field to a particular filter sequence. The final feature map selectively responds strongly to the church image and weakly to the field image.

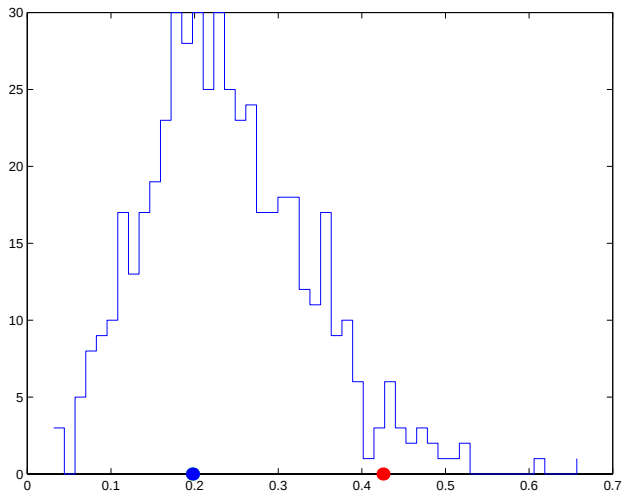

Figure 4: A histogram of the tiger stripe feature responses to a set of 500 images. The response for the tiger image (marked in red) is more than twice that of the waterfall image (marked in blue).

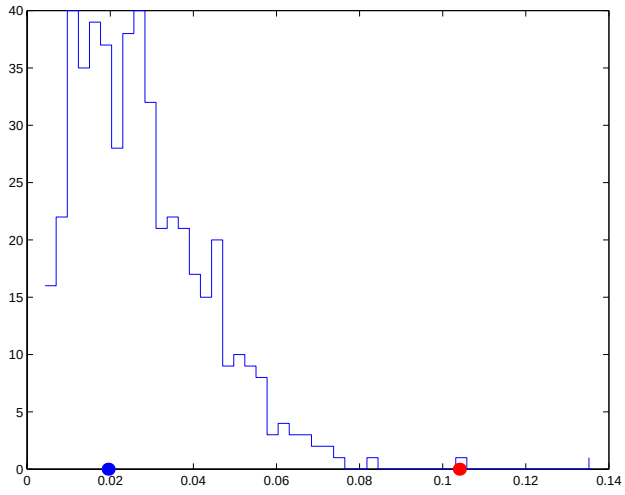

Figure 5: A histogram of feature responses to a set of 500 images. One particular image of a church has a strong response (marked in red), while an image of a field has a weak response (marked in blue). The histogram is super-Gaussian with a kurtosis of 7.8.

Figure 4 shows a histogram of the tiger stripe feature's response to 500 images. Notice that observation of the tiger's strong response (red point) would be considered much more statistically significant than observation of the waterfall's weak response (blue point). Figure 5 shows the histogram of a more kurtotic feature that responds strongly to images of churches and weakly to images of fields.

Further evidence for the significance of these highly selective features is a theorem from the projection pursuit literature, which states that random projections of a high dimensional random variable are almost always Gaussian [5]. This holds even when the high dimensional random variable does in fact have significant statistical structure, as is the case for natural images.

2.1 Relationship to Wavelets

There is a superficial similarity between the highly selective feature approach and retrieval based upon a set of wavelet coefficients [9]. In a wavelet approach images are represented and retrieved using their wavelet coefficients. Like our features, many of the wavelet coefficients of an image are close zero. But unlike our approach, these coefficients are very sensitive to changes in the image. For example, wavelet coefficients are very sensitive to a small shift in the image plane. A wavelet based approach is best thought of as a very efficient approximation to template matching (because distance in the image space is a simple function of the distance in wavelet feature space).

3 Query Learning with Boosting

At first it might seem that the introduction of tens of thousands of features could only make the query learning process infeasible. How can a problem which is difficult given ten to twenty features become tractable with 10,000? Two recent results in machine learning argue that this is not necessarily a terrible mistake: support vector machines (SVM) [3] and boosting [7]. Both approaches have been shown to generalize well in very high dimensional spaces because they maximize the margin between positive and negative examples. Boosting provides the closer fit to our problem because we can use it to greedily select a small number of features from a very large number of potential features.

In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple learning algorithm (e.g., it might be used to boost the performance of a simple perceptron). It does this by combining a collection of weak classification functions to form a stronger classifier. In the language of boosting the simple learning algorithm is called a

weak learner. So, for example the perceptron learning algorithm searches over the set of possible perceptrons and returns the perceptron with the lowest classification error. The learner is called weak because we do not expect any single perceptron to classify the training data well (perhaps the perceptron may only classify the training data correctly 51% of the time). In order for the weak learner to be boosted, it is called upon to solve a sequence of learning problems. In each subsequent problem examples are re-weighted in order to emphasize those which were incorrectly classified by the previous weak classifier. The final strong classifier is a weighted combination of weak classifiers.

One important goal for image database query learning is that the final classifier depend only on a small number of complex features. A classifier which depends on few features will be more efficient to evaluate on a very large database. In addition, a simple classifier which depends on few features will be more likely to generalize well.

In support of this goal, we design our weak learning algorithm to select the single highly selective feature along which the positive examples are most distinct from the negative examples. For each feature, the weak learner computes a Gaussian model for the positives and negatives, and returns the feature for which the two class Gaussian model is most effective. In practice no single feature can perform the classification task with 100% accuracy. Subsequent weak learners are forced to focus on the remaining errors through example re-weighting. In the experiments below, the algorithm is typically run for 20 iterations, yielding a strong classifier which depends upon 20 features. Table 1 shows the learning algorithm.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Train one hypothesis h_j for each feature j using w_t, with error $\epsilon_j = \Pr_i^{w_t}[h_j(x_i) \neq y_i]$.
  2. Choose $h_t(\cdot) = h_k(\cdot)$ such that $\forall j \neq k, \epsilon_k < \epsilon_j$ (i.e., the hypothesis with the lowest error). Let $\epsilon_t = \epsilon_k$.
  3. Update:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0, 1$ for example x_i classified correctly or incorrectly respectively, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$. Normalize $w_{t+1,i} \leftarrow \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}}$ so that w_{t+1} is a distribution.
- The final hypothesis is:
$$h(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Table 1: The boosting algorithm for learning a query online. T hypotheses are constructed each using a single feature. The final hypothesis is a weighted linear combination of the T hypotheses where the weights are inversely proportional to the training errors.

4 User Interface

The user interface of our query engine has two phases: an initial browsing phase, and a relevance feedback stage.

The user begins a new query by browsing the database to select a few positive examples. Users found that it was somewhat tedious to hand pick negative examples. Instead we randomly choose 100 images from the database to form a set of generic negative examples. This is a reasonable policy because the the number of images which satisfy any particular query is very small. Nevertheless, this policy for selecting negatives is somewhat risky because it is possible that the negative set may contain true positives. Typically we run AdaBoost for 20 iterations which is usually sufficient to achieve zero training error[2]. Since the number of positive training examples is typically smaller than the number of negative examples, we initially weight them higher so that the sum of the weights of the positives and negatives are equal. This encourages correct classification of the positive examples at the outset.

Each image in the database can then be classified by the strong classifier. Alternatively, since the strong classifier is itself a perceptron, the images can be ranked by their margin. The first goal of an image retrieval program is to present the user with useful images which are related to the query. Since the learning algorithm is most certain about images with a large

[2]One might be concerned that attaining zero training error would lead to poor performance on the test set. In fact for the boosting algorithm this is usually not the case, since the margin between the negatives and positives typically increases even after there is zero error on the training set [16].

positive margin, a set of these images are presented. Without further refinement this set of images often contains many false positives.

Retrieval results can be improved greatly if the user is given the opportunity to select new training examples [11, 4]. Recall that images are classified as positive if the final hypothesis (i.e., weighted combination of the weak learners) exceeds the AdaBoost threshold (i.e., decision boundary): $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$ (see Table 1). This defines a margin such that images highly above threshold are considered most positive while images well below threshold are labeled most negative.

Following a query, three sets of images are presented to the user: (*i*) test set images with large positive margin; *ii*) the randomly selected negative images which are close to the decision boundary; and (*iii*) test set images which are close to the boundary. The first set is intended to allow the user to select new negative training examples which are currently labeled as strongly positive. The landscape and leopard image in Figure 7 are obvious false positives that the user can add as negative examples. The second set allows the user to discard randomly chosen negatives which are not true negatives. The third set allows the user to refine the decision boundary by labeling examples which determine the margin. In Figure 7, the last row (green frame) contains three images of cars which are close to the decision boundary. The user can add these as positive examples to update the query.

In every case the final query is produced by running AdaBoost for 20 iterations. This yields a strong classifier which is a simple function of 20 complex features. Since image databases are very large, the computational complexity of the final classifier is a critical aspect of retrieval performance.

5 Results

Experimental verification of image retrieval systems is a very difficult task. There are few if any standard datasets, and there are no widely agreed upon evaluation metrics.

To test the retrieval performance of the system we constructed five classes of natural images (sunsets, lakes, waterfalls, fields, and mountains) using the Corel Stock Photo [3] image sets 1, 26, 27, 28, and 114 respectively [2, 14]. Each class contains 100 images.

Figure 6 shows the average recall and average precision for the five classes of natural images. Recall is the ratio of the number of relevant images returned to the total number of relevant images. Precision is the ratio of the number of relevant images returned to the total number of images returned.

[3]This publication includes images from the Corel Stock Photo images which are protected by the copyright laws of the U.S., Canada and elsewhere. Used under license.

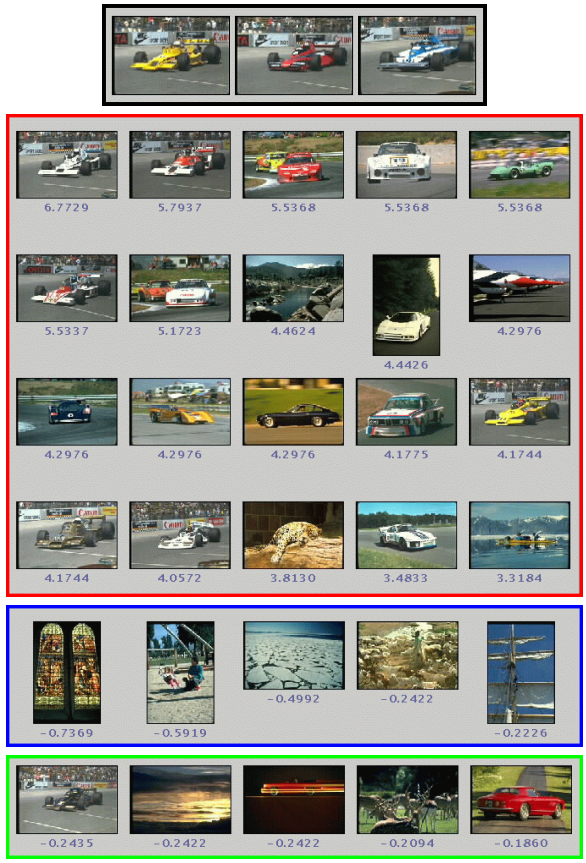

Figure 7: Race cars: The top row shows the positive examples followed by the top twenty retrieved images in the middle portion (red frame). The first row of the bottom portion (blue frame) shows the negative images in the training set which are close to the decision boundary. The second row (green frame) shows images in the test set which are near the boundary.

Figures 7 to 9 show results from queries for race cars, flowers, and waterfalls, cloudy skies, and jet planes in a 3000 image data set (using Corel image sets 1 to 30).

6 Summary and Conclusions

We have presented a framework for image retrieval based on representing images with a very large set of highly selective features. Queries are interactively learned online with a simple boosting algorithm. The selectivity of the features allow effective queries to be formulated using just a small set of features and a small number of training examples. This supports our observation of the "sparse" causal structure of images.

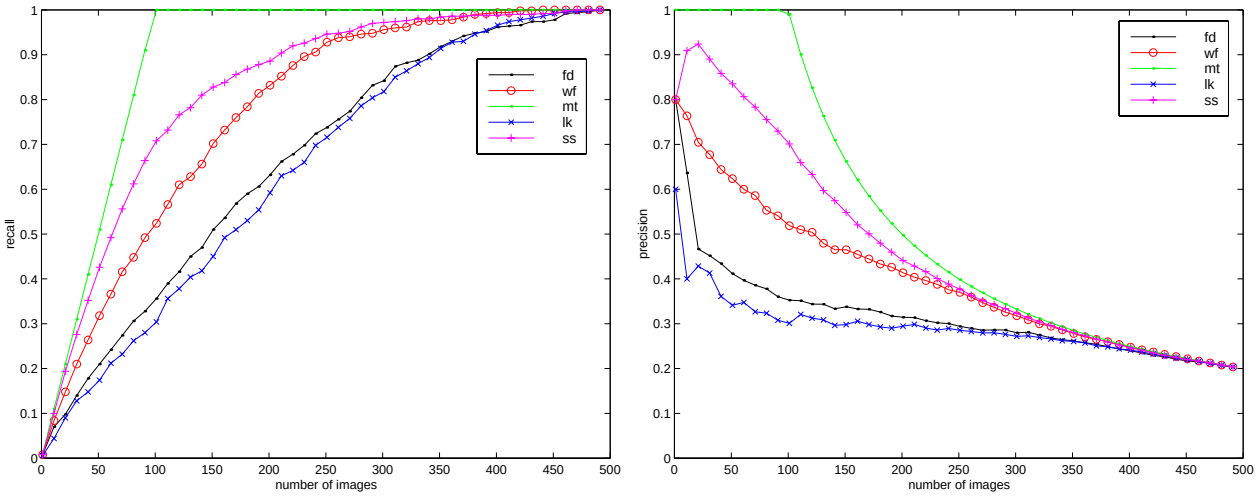

Figure 6: Average recall and average precision for the five classes of natural images. Essentially perfect performance was achieved on the mountains (mt) class. Closer inspection shows that the images in this class are fairly homogeneous especially when compared to images in the fields (fd) and lakes (lk) classes. These results were averaged over five trials, each using five randomly selected positive and four negative training examples (one from each negative class).

It also makes training the classifier simple, and retrieval on a large database efficient.

Acknowledgments

Thanks to John Fisher for helpful discussions. This work was supported in part by Nippon Telegraph and Telephone.

References

[1] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Comm.*, 31(4):432–540, 1983.

[2] Corel Corporation. Corel stock photo images. http://www.corel.com.

[3] C. Cortes and V. Vapnik. Support vector networks. *Mach. Learn.*, 20:1–25, 1995.

[4] I. Cox, M. Miller, T. Minka, and P. N. Yianilos. An optimized interaction strategy for bayesian relevance feedback. In *IEEE Comp. Vis. Patt. Recog. Conf.*, pages 553–558, 1998.

[5] P. Diaconis and D. Freedman. Asymptotics of graphical projection pursuit. *Ann. Stat.*, 12:793–815, 1984.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–32, 1995.

[7] Y. Freund and R. E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *J. Comp. & Sys. Sci.*, 55(1):119–139, 1997.

[8] J. Huang, S.R. Kumar, M. Mitra, W.J. Zhu, and R. Zabih. Image indexing using color correlograms. In *CVPR97*, pages 762–768, 1997.

[9] Charles E. Jacobs, Adam Finkelstein, and David H. Salesin. Fast multiresolution image querying. In *SIGGRAPH 95*, 1995.

[10] M. Kelly, T. M. Cannon, and D. R. Hush. Query by image example: the candid approach. *SPIE Stor. and Ret. Image & Video Databases III*, 2420:238–248, 1995.

[11] C. Nastar, M. Mitschke, and C. Meilhac. Efficient query refinement for image retrieval. In *IEEE Comp. Vis. Patt. Recog. Conf.*, pages 547–552, 1998.

[12] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *ICCV98*, pages 555–562, 1998.

[13] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: content-based manipulation of image databases. *Int. J. Comp. Vis.*, 18(3):233–254, 1996.

[14] A. L. Ratan and O. Maron. Multiple instance learning for natural scene classification. In *Int. Conf. Mach. Learn.*, pages 341–349, 1998.

[15] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Patt. Anal. Mach. Intell.*, 20(1):23–28, 1998.

[16] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Stat.*, 26(5):1651–1686, 1998.

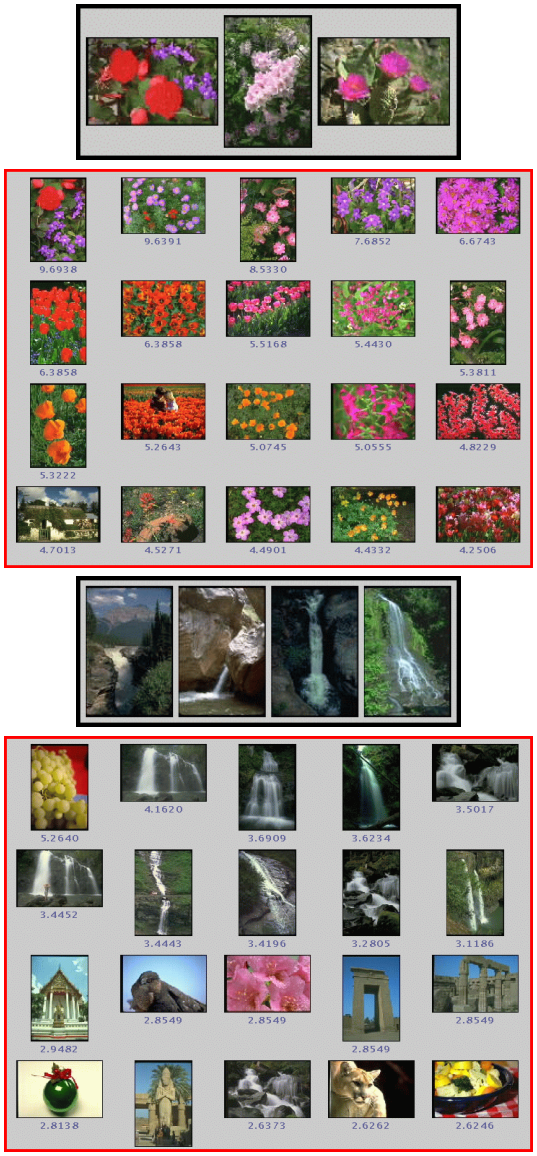

Figure 8: Flowers and waterfalls: The positive examples are shown first followed by the top twenty retrieved images.

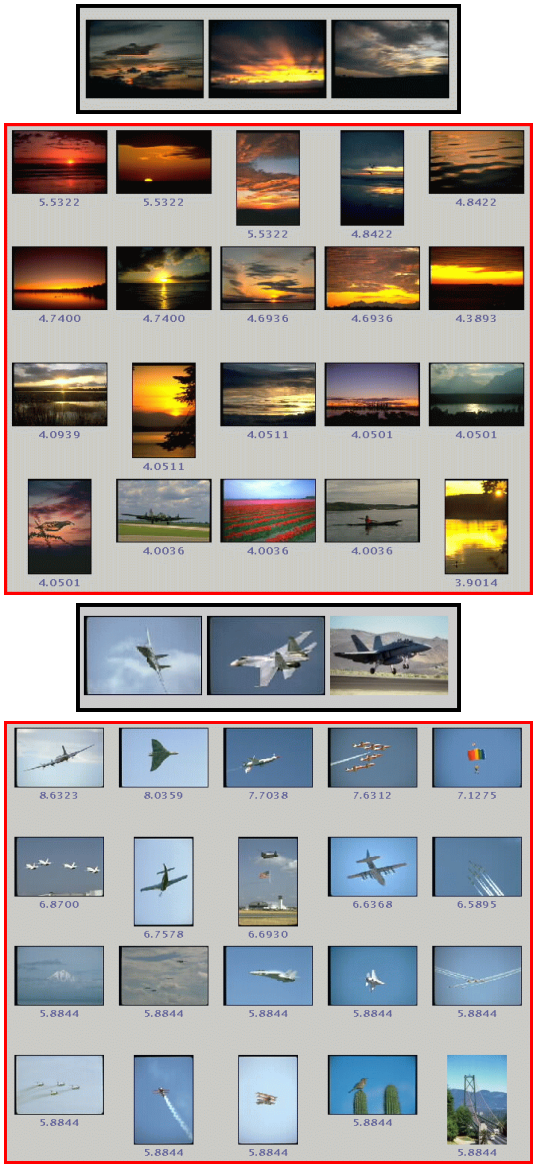

Figure 9: Cloudy skies and jets: The positive examples are shown first followed by the top twenty retrieved images.

[17] E P Simoncelli and E H Adelson. Noise removal via bayesian wavelet coring. In *3rd IEEE Int'l Conf on Image Processing*, 1996.