Image Retrieval Using Boosting Algorithm

Yizheng Cai
Department of Computer Science
University of British Columbia
Vancouver, V6T 1Z4
Email:*yizhengc@cs.ubc.ca*
Homepage: *www.cs.ubc.ca/~yizhengc*

Abstract

This project uses an offline learning algorithm to get a highly efficient classifier for online image retrieval. The boosting algorithm is adopted for the learning process. It chooses a small number (10 in this project) of highly selective features from a very large feature set (there are totally over 45,000 features in the set in this project) and combine them together to get a strong classifier for retrieval. The computation of such strong classifier is very fast so that it can be applied to image retrieval on a very large dataset.

1 Introduction

For the task of image retrieval, the users would probably expect particular images to be retrieved when submitting a query. How close the retrieved images are to the users' expectations only depend on the query they submit. There are generally two ways of submitting these queries. One is by key words and the other is by sample images. Kinh Tieu and Paul Viola [1] have already used the boosting algorithm for image retrieval using sample image queries and the mechanism of relevant feedback. Their approach showed very impressive results. But, in most cases, it is difficult for the users to provide the sample images because they probably do not have one before getting the results. So, the key words queries are still necessary and also useful because it is easy and natural.

There are also two different approaches for key words queries. One is text based, which means there is text associated with each image in the database. So, the retrieval work is just to index the key words within the text, which is actually a text matching. This is what Google does for its online image retrieval. But, for more general image databases, there is no text associated with each image. So, the content based image retrieval is required. This is the right task of this project.

The motivation of this project is to take a closer look at the new boosting algorithm in the area of machine learning and test the effectiveness of the classifier trained by boosting algorithm. The general idea of this project is to do an offline training using boosting algorithm to get a classifier, which detect one specific object in the image, and associate each of the classifiers with a key word that represent the object. For example, if a user submits a query with the key word "lion", a pre-trained classifier,

which is associated with the word “lion”, will be invoked to scan the whole image database and return the positive images as the result. This approach can also accept more flexible queries like “adding” queries and “subtraction” queries. For example, the query “lion and tree” just need to invoke both the two classifiers for these two key words and get the images retrieved by both of them. For the “subtraction” query, “lion without tree”, the expected images are just the result of “lion” classifier subtracted by the results of “tree” classifier. In a word, with this approach, the image retrieval system can accept very flexible combined key words queries and this is quite useful for image retrieval. So, what this project focuses on is to construct one such classifier and test its efficiency. The main methodologies used in this project are based on the work done by Kinh Tieu and Paul Viola on boosting image retrieval [1].

The first part of this report will describe the methodology used in this project. The second part will be the implementation details. In the third part, the testing results will be presented with discussion. The last part will be the discussion and future work of this project.

2 Methodology

Before getting into the implementation details of the project, it is better to know how to build up the classifier and why to use boosting algorithm first.

2.1 Boosting Algorithm

How to find an effective classifier is obviously the main problem of classification. The simple classifiers, which might perform only slightly better than random guessing, are much easier to get rather than a sophisticated one. So, the motivation of the boosting algorithm is just to find a way to “boost” these simple classifiers into a much stronger one through learning. Among all the boosting algorithms, the AdaBoost, introduced by Schapire and Freund in 1995 [2], has solved many practical difficulties of the earlier boosting algorithms.

The algorithm with simple modification of AdaBoost procedure is used in this project. The pseudocode of this algorithm is given in Fig. 1. It is based on those shown in [1, 3] with little changes.

The cost function of AdaBoost is simply the training error of the final classifier. Given training data $(x_1, y_1),...,(x_m, y_m)$ and the final classifier $H(x)$, the error function can be defined as follows:

$$Error = \frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \tag{1}$$

The error is bounded as follows [4]:

$$\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) = \prod_t \left(\sum_i \omega_{t,i} \exp(-\alpha_t y_i h_t(x)) \right) \tag{2}$$

where $f(x) = \sum_t \alpha_t h_t(x)$, $H(x) = sign(f(x))$, $h_t(x)$ are weak classifiers and $\omega_{t,i}$ are the weights associated with each training image.

Eq. (2) indicates that the error can be rapidly reduced if α_t and h_t can be properly chosen in each round *t* to minimize the upper bound. It is also proved by Freund and Schapire [2] that AdaBoost can find the steepest path to reach the minimum error. With the growth of rounds, the error will exponentially decrease. From the pseudocode in Fig. 1, we can see, intuitively, that the boosting algorithm is just to find

a linear combination of a set of weak classifiers *h(x)* with corresponding weights to minimize the error. The weights α_t are the evaluation of the effectiveness of each weak classifier.

- Given training images $(x_1, y_1),...,(x_m, y_m)$ where $y_i = 0,1$ for negative and positive training images respectively.
- Initialize the weight $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0,1$ respectively, where m and l are the number of negative and positive images respectively.
- For t = 1,…..,T:
  1. Train one weak classifier h_j for each j using ω_t , with error $\varepsilon_j = \sum_i \omega_{t,i} \left| h_j(x_i) - y_i \right|$
  2. Choose $h_t(\cdot) = h_k(\cdot)$ such that $\forall j \neq k, \varepsilon_k < \varepsilon_j$ (i.e., the weak classifier with the lowest error). Let $\varepsilon_t = \varepsilon_k$.
  3. Update:
     $$\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0,1$ for training image x_i classified correctly or incorrectly respectively, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
  4. Normalize $\omega_{t+1,i} = \frac{\omega_{t+1,i}}{\sum_{j=1}^{n} \omega_{t+1,j}}$, so that ω_{t+1} is a distribution.
- The final strong classifier is:

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Figure 1: The pseudocode of the boosting algorithm used in this project. T is the number of iterations, *h(x)* is the weak classifier, which will be defined in the next section.

2.2 Base Classifiers

In the previous section, the boosting algorithm has already set up a framework to "boost" a set of base classifiers into a strong one. The next step is to get these base classifiers. Since this is not the main purpose of this project, there will be only simple description of the approach in this section.

Kinh Tieu and Paul Viola proposed an approach to use 25 primitive filters to extract the texture features from the input image. The 25 simple linear features include "oriented edges", "center surround" and "bar" filters, which are shown in Figure 2.

The idea is to use these primitive filters as first order features to extract the feature map of the input images. Then, apply the same filters as second order features to get the arrangement of the feature map from the first order features. Finally, use the same way to get the arrangement of the second order feature map. The process starts by convoluting the 25 primitive filters with the images on the highest resolution. The

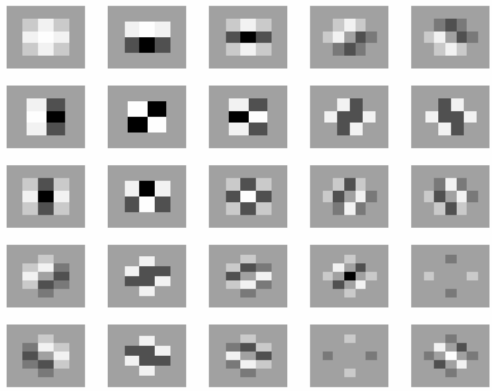

Figure 2: The 25 primitive filters used for extracting textures and texture of texture from the input images. The final feature value of the response will be used for classification.

results are the 25 feature maps. Rectify the feature maps and down sample by two. Use the 25 feature maps as the input of the second round and apply the same operations to get 625 feature maps. Repeat it again and produce 15,625 feature maps. All these operations are done on each color channel of the images. So, totally, there are 46,875 feature maps). Finally, sum up the feature map on each pixel to get one feature value which is later used for the base classifier. The mathematical representation of the process [1, 5] is as follows:

$$M_i = \downarrow 2(|f_i \otimes X|) \tag{3}$$

$$M_{i,j} = \downarrow 2(|f_j \otimes M_i|) \tag{4}$$

$$M_{i,j,k} = \downarrow 2(|f_k \otimes M_{i,j}|) \tag{5}$$

where X is the original image, f is a primitive filter, $\downarrow 2$ is the down sample by two operation and M is the feature map.

The final feature value is computed as: $g_{i,j,k,c} = \sum_{pixels} M_{i,j,k,c}$ (6)

where c is the color channel of the image.

3 Implementation

3.1 Base Classifier Construction

The 25 primitive filters introduced in the previous section are actually constructed by 5 original filters. These five original filters are just oriented edge, bar type features. Actually, any similar type of features can be used. Those used in this project are shown in Figure 3.

Figure 3: The five original filters used to construct the 25 primitive filters.

The matrix representation of the five filters is as follows:

$$m_1 = \begin{vmatrix} 1/4 & 1/2 & 1/4 \end{vmatrix} \tag{7}$$

$$m_2 = \begin{vmatrix} 1/2 & -1/2 \end{vmatrix} \tag{8}$$

$$m_3 = \begin{vmatrix} 1/4 & -1/2 & 1/4 \end{vmatrix} \tag{9}$$

$$m_4 = \begin{vmatrix} 0 & 1/4 & 0 \\ 1/4 & 0 & -1/4 \\ 0 & -1/4 & 0 \end{vmatrix} \tag{10}$$

$$m_5 = \begin{vmatrix} 0 & 1/4 & 0 \\ -1/4 & 0 & 1/4 \\ 0 & -1/4 & 0 \end{vmatrix} \tag{11}$$

The process to get the 25 primitive filters from the 5 original ones is to repeatedly convolute the 5 filters horizontally and vertically with each other to get 5×5 results:

$$f_k = m_i \otimes m_j' \tag{12}$$

where m is the matrix of each original filter, $i, j = 1,2,3,4,5$. f_k is the same as those in Eq. (3).

Now, the base classifier can be constructed with the 25 primitive filters and the detailed algorithm to get the single feature value. In order to make the base classifier as simple as possible, the base classifier only depend on one feature map, which means there will be 46,875 base classifiers in the whole set. The definition is as follows:

$$h_j(x) = \begin{cases} 1 & if\ p_i g_j(x) < p_i \theta_j \\ 0 & otherwise \end{cases} \tag{13}$$

where x is the input image, p_j is a parity to indicate the direction of inequality sign, θ_j is the threshold. $g_j(x)$ is the same as the one in Eq. (6), but with different subscript j, which index the corresponding feature map from the rearranged 46,875 ones. $h_j(x)$ equals 1,0 for positive images and negative images respectively.

Eq. (13) indicates that the two new parameters, p_j and θj , in the base classifier need to be decided through training. For p_j, it equals 1 if the positive set values are below the negative and -1 if positive set value are larger. In this project, the decision of choosing the threshold depends on the average of the positive set and the negative set.

$$\theta_j = \begin{cases} pr\left(\frac{1}{|C_0|}\sum_{x \in C_0} g_j(x) + \frac{1}{|C_1|}\sum_{x \in C_1} g_j(x)\right) & if \quad p_j = 1 \\ (1-pr)\left(\frac{1}{|C_0|}\sum_{x \in C_0} g_j(x) + \frac{1}{|C_1|}\sum_{x \in C_1} g_j(x)\right) & otherwise \end{cases} \tag{14}$$

where C_0 , C_1 represent the negative set and positive set respectively, pr is the parameter introduced to control the position of the threshold relatively to the centers. This is significant because in most situations the distribution of positive set and negative set are quite different. To place the threshold at the middle point of the two centers would probably produce many false classifications. The value of pr will be decided, in later section, by comparing the detection rate and false positive rate of the training and testing set under different values of pr.

According to the boosting algorithm theory, the base classifier should only be a little bit better than a random guess. But in this project, because the number of training round is limited to choose as less base classifiers as possible to increase the speed, the accuracy of each base classifier matters more than other situations. That's the reason to introduce this new parameter and trying to choose the optimal one.

3.2 Implementation Issues for Boosting

The detailed steps for the AdaBoost algorithm implementation are already presented in Figure 1. As is seen in the figure, for each round of training, all the images should be computed by all the base functions, which is very time-consuming. It is easy to notice that, in each round, only the weight of the images are changed while the classification results of each base classifier on each image remain the same. So, it is not necessary to compute the huge amount of feature maps every time but rather one time and store it for further use.

In this project, the whole task is divided into three stages. The first is initializing the primitive filters and learning the threshold for each base classifier. The second is to "boost" the strong classifier. The final stage is to construct the module to retrieve images from image set. As is shown in the previous section, all the feature maps are computed in the first stage for choosing the best threshold. So, this result is computed and stored for the boosting stage so that the total computational cost for training will drastically decrease.

In addition, at the end of the second stage, the threshold of the final strong classifier of AdaBoost is defined as $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$ in Figure 1. The form of the threshold is like a result of voting. Each of the base classifier votes with their weights. By changing the ratio of passing the voting, the criteria will be stronger or weaker and the detection rate and false positive rate of the strong classifier will also vary. So, we replace the coefficient 0.5 with a parameter λ, $\lambda \in [0,1]$.

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T}\alpha_t h_t(x) \geq \frac{1}{\lambda}\sum_{t=1}^{T}\alpha_t \\ 0 & otherwise \end{cases} \tag{15}$$

In later section, the value of λ will be changed to get the ROC curve of the testing result. The optimal value of λ will be decided according to that curve.

3.3 Data for Training and Testing

The training and testing data of this project are mainly from the Corel [6] image data set. The one used in this project has sets A and B. This project is designed to train a classifier to retrieve "lion", so, all the 29 "lion" images in set A are used as positive set (there is no "lion" images in set B). Because the size of the negative set should not be too much larger than the positive, there are only 72 images for negative set. Most of the negative set images are from set A, some are from set B. Because "Cross Validation" will be used to choose the parameter, the positive and negative sets are divided into two sets. The training set has 16 positive and 41 negative images. The testing set has 13 positive and 31 negative images.

Finally, the system is tested on a more general image set which has 15 "lion" images, randomly downloaded and resized from Google, and 204 "non-lion" images from set B. Because some of the images in set B has totally different object from those in set A. For example, there are images about trains in set B while there is no one in set A. In order to enrich the training set for better training, some of the representative images are chosen from set B and added to negative training set.

4 Results

4.1 Parameter Optimization

In order to show how the two parameters mentioned in section 3.1 and 3.2 would affect the performance of the strong classifier, the number of training round should be decided first. In this project, the two newly introduced parameters are set to the unbiased ones (pr=0.5, λ=2) so that it is easy to decide the proper training round. The training error of the strong classifier at different training rounds is shown in Figure 4.

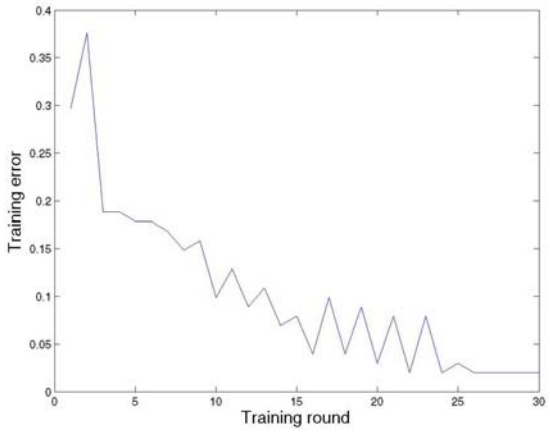

Figure 4: The training error (including false positive and false negative) of the strong classifier with different training round. This result is under the standard parameter pr=0.5, λ=2.

Kinh and Paul stated in their work [1] that the training round should reach the zero training error. From Figure 4, it is seen that there is drastic vibration from 16 to 24 while the minimum error does not differ much. In order to make the base classifier as less as possible to increase the computation speed, the training round is set to 16.

For the two new parameters—pr and λ, the ROC curve of detection rate and false positive rate on both the training set and testing set with different values of the two parameters are given in Figure 5. The range and interval of the two parameters could be at any scale.

"Cross validation" is used in this project to choose the best parameters. The smaller detection rate and larger false positive rate of both training test set with different parameter values are used to construct the new ROC curve, shown in Figure 6.

From Figure 6, there is no great difference when pr range from 0.45 to 0.55. So, in this project, pr is set to be 0.5. In order to get acceptable detection rate at low false positive rate, the λ chosen, according to the curve, is 1.9.

4.2 Testing Result

After choosing the optimal parameter, the retrieval system is applied to the testing set with 219 images. 9 "lion" images are retrieved out of 15 images. 20 false images are also returned out of 204 negative images. So, the detection rate is 60% and the false positive rate is 9.8%, which is satisfactory for the task of image retrieval. The retrieved images are presented in Figure 7. All the retrieved images are ranked from highest to lowest by the value computed by $f(x)$ of strong classifier $H(x) = sign(f(x))$.

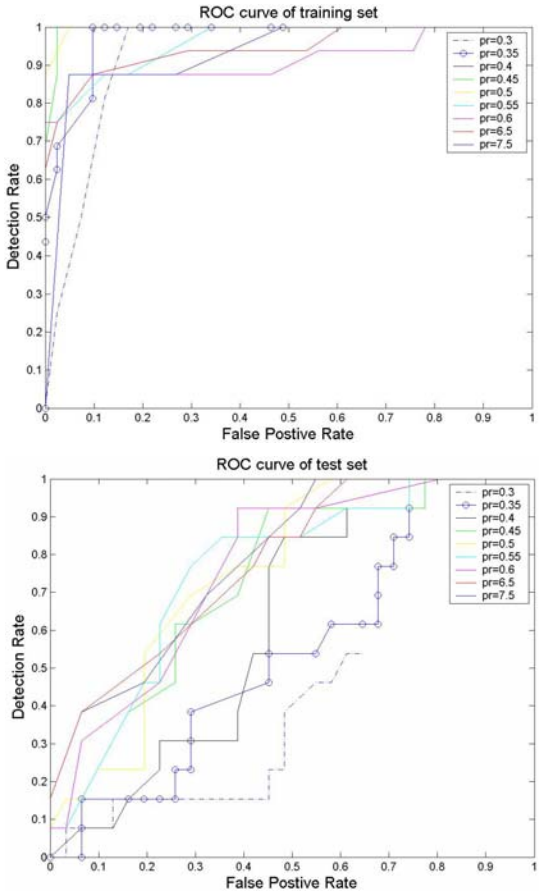

Figure 5. The upper one is the ROC curve on the training set and the lower one is on the testing set. The range of *pr* is [0.3, 0.7] with interval 0.05. The range of λ is [1, 10] with interval of 0.1. The λ increases while the detection rate decreases.

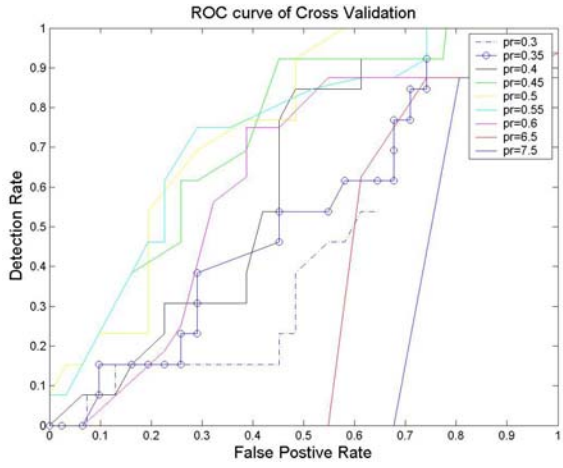

Figure 6: The ROC curve with cross validation on the training and testing set. The range and interval of the two parameters are the same as in Figure 5.

5 Discussion and Future work

The approach implemented in this project has shown satisfactory result for the task of image retrieval. This result demonstrates the efficiency of the boosting algorithm in practical applications of image retrieval on large databases. It is also shown that the control of the two parameters would significantly affect the result. What is presented in this project for choosing the threshold of each base classifier is just an intuitive algorithm which does not have theoretical support. More theoretical research can be done on the issue of how to choose the best threshold through learning. This work will be valuable if the computational cost of the strong classifier is more important because choosing a better threshold will significantly improve the detection rate and reduce the false positive rate while not increasing the classification time. Thus, the final strong classifier can be fast enough to be implemented in other real-time context or large scale data classification works.

After a simple analysis of the result, we can see that the size of training set and how the negative set are chosen would affect the result significantly. Figure 5 indicate that the training set and test set for cross validation might be very different. From Figure 7, we can notice that most negative feedback are visually similar, which probably indicate that the negative set for training does not have those similar images. Those engineering work need to be improved with more experiment and test. Due to the limit of time, it is not sufficiently done in this project.

Also, this project can be extended to an object recognition system. For example, there might be an image retrieved by both the “lion” classifier and “grass” classifier. This image can be segmented in different parts and resubmitted to the two classifiers. The segment with “lion” will probably have high rank in “lion” classifier and low rank in “grass” classifier. Thus, the segment is recognized as a lion and can be automatically labeled with the word “lion”, which complete the task of segmentation and recognition given an input image.

Acknowledgments

Thanks to Nando de Freitas for useful suggestions and inspirations. Thanks to Kenji Okuma for helpful discussions on the boosting algorithm and implementation details. Thanks to Kinh Tieu for providing the 5 primitive filters he used in his paper [1].

References

[1] Kinh Tieu and Paul Viola. Boosting Image Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2000

[2] Yoav Freund and Robert E. Schapire. A Decision-theoretic Generalization of On-line Learning and An Application to Boosting. In *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[3] Paul Viola and Michael J. Jones. Robust real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[4] Robert E. Schapire. The Boosting Approach to Machine Learning: An Overview. In *MSRI Workshop on Nonlinear Estimation and Classification,* 2002

[5] Jeremy. S. De Bonet and Paul Viola. Structure driven image database retrieval. In *Advanced Neural Information Processing Systems,* volume 10, 1998

[6] C. Corporation. Corel stock photo images. *http://www.corel.com*

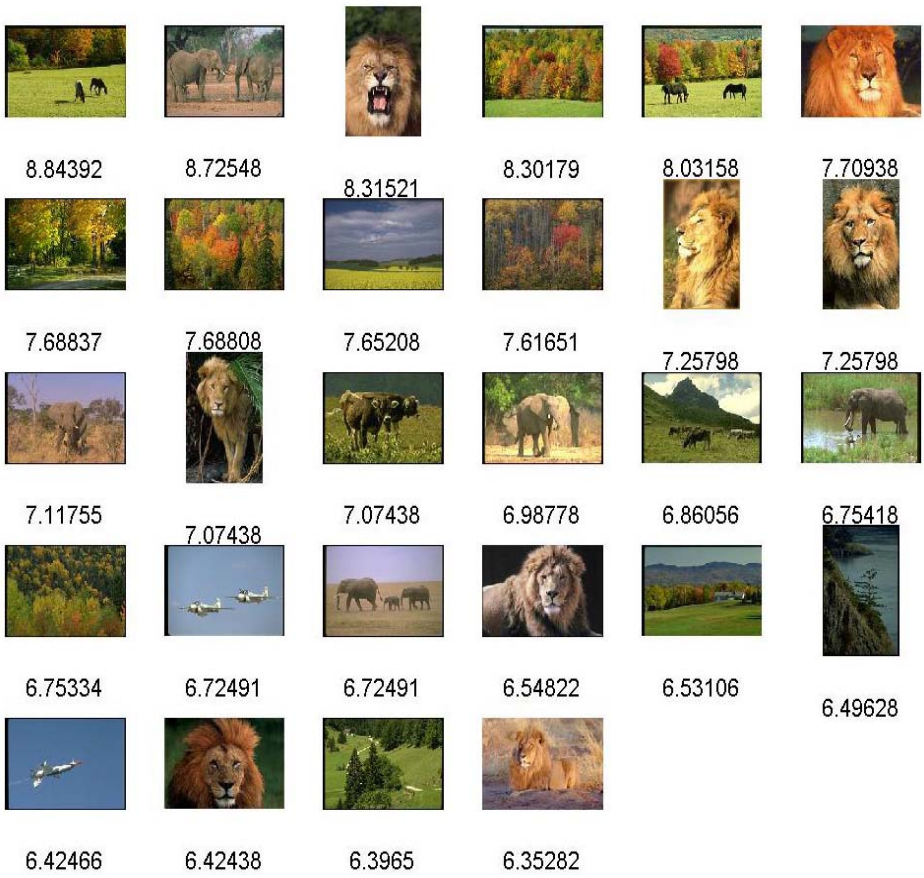

Figure 7: The retrieved images from the testing set. All the images are ranked from high to low according to the value of the final strong classifier *f(x)* on the image. These values are shown under each image in the output.