Automatic Classification of Natural and Synthetic Images

MAURIZIO GARBARINO

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2008

Automatic Classification of Natural and Synthetic Images

M A U R I Z I O G A R B A R I N O

Master's Thesis in Computer Science (30 ECTS credits)
at the School of Computer Science and Engineering
Royal Institute of Technology year 2008
Supervisor at CSC was Stefan Carlsson
Examiner was Stefan Carlsson

TRITA-CSC-E 2008:112
ISRN-KTH/CSC/E--08/112--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

The classification of images based on semantic description is a challenging and important problem in automatic image retrieval. A binary classificator for natural and synthetic images has been developed.

Single simple features are extracted from the raw images data in order to exploit the difference of color pattern and spatial correlation of pixels in natural and synthetic images. These features have poor accuracy if used alone but when combined together forming a more complex and accurate global classifier, their precision is boosted. Three different global approaches have been tested implementing the AdaBoost algorithm, a decision tree and a neural network.

Using neural network it was possible to achieve an accuracy of 94.2% that is in par with the best result that is possible to find in literature. The classifiers have also been tested on a thumbnail dataset showing an increased speed of 490% and, for the best case, an accuracy of 92.5%. These results confirm that the developed classifier is suitable to be used in an image search engine context.

Contents

Chapter 1

Introduction

1.1 Image Classification

The purpose of an image classification system is to separate images into different classes. An ideal system should be able to discern different images with no hesitation just like an human being. Unfortunately, sometimes the classification task is hard and ambiguous even for an human. This makes the problem even more challenging.

In this project, a binary classifier is developed. The two classes involved in the classification are *Natural* and *Synthetic* images.

Given an image, the classifier extract and analyze some of the most relevant features and combine them in order to generate an opinion.

1.2 Previous Work

Image classification is not a new field. Many studies have been made lately and different approaches have been proposed. Martin Szummer and Rosalind W. Picard [28] built an Indoor-Outdoor image classification based on nearest neighbor classification. They split the images into n blocks and extract color and texture features from different sub blocks and combine them together in order to exploit spatial properties of the image. Their work showed an accuracy of 90 percent on the dataset. An additional approach using neural network has been tried but without any improved results.

Another binary classification system has been proposed in [33] and [31] where advanced features such as the DCT coefficients obtained with the JPEG compression and edge directions are evaluated. The accuracy obtained was 94 percent for the best case in their test.

The WebSeek search engine [27] performs the classification based on some

information extracted from the color histograms of the images. The system described in [23] not only uses some information from the image content, but also from the image context (the HTML document in which the image is embedded). The image content features used are the number of colors, the fraction of impure colors (not red, green or blue and not black, white or gray), the shape of the image (squareness), the neighbor variation (fraction of neighbor pixels having the same color) and the color dispersion (distance of the mean color in the sorted color histogram). Moreover, the filename portion of the image URL is evaluated. If it contains words that are usually associated with the specified image class, then it is considered for the classification. Unfortunately in [27] and [23] is not easy to evaluate the system accuracy. In [27] the authors claim an accuracy rate of 92.3 percent for Web graphics and 91.4 percent for Web photographs. However, the definition of what they consider photographs and graphics is not clear. The authors of [23] also don' t specify exactly what they consider photographs and graphics.

In articles [29] and [30] Vailaya et al. describe a method to classify vacation images into different classes such as city/landscape, indoor/outdoor, and sunset/mountain/forest scenes. A Bayesian framework has been used in order to separate the images in a classification hierarchy. The reported accuracy is 95.3 percent for city versus landscape classification, 90.5 percent for indoor versus outdoor classification, and 96.6 percent for sunset versus forest/mountain classification.

Gorkani et al. [15] propose an interesting method for distinguishing city and suburb from country and landscape scenes. The most dominant orientation in the image texture is extracted and is used as metric. City and landscape images have a different dominant orientation: prevalent horizontal for landscapes and vertical for city. Citing the authors, they say that it takes humans almost no time or 'brain power' to distinguish between those image classes, so there should exist an easy and fast to calculate feature. They report a classification accuracy of 92.8 percent on only 98 test images. The results are good, though, the size of the dataset has to be taken into account while looking at the accuracy.

Yiu et al. in [32] classify pictures into indoor/outdoor scenes using color histograms and texture orientation. The algorithm by Gorkani and Picard [15] has been used in order to exploit the orientation. The vertical orientation is the main discriminant feature: indoor images tend to depict artifacts, and artifacts usually have strong vertical or horizontal lines.

Bradshaw proposes a method for labeling image regions as man-made or natural. For example, houses, roads and buildings are man made, while hills, tree and mountains in the background are natural. For homogeneous images depicting either only natural objects or only man-made, an accuracy of 90

percent is claimed. Bradshaw also applied the same approach to the problem of distinguishing outdoor versus indoor images [8] and he reported a classification accuracy of 86.3 percent.

Schettini et al. [26] developed a system able to classify photographs, graphics, text, and compound documents using a number of different features: edge distribution, color distribution and statistics, texture features wavelet coefficients, and percentage of pixel having the skin color. Compound documents are just images consisting of mixed photographs, graphics, and text. Single decision trees trained by the CART (Classification And Regression Tree) algorithm are used as base classifier and multiple decision trees are combined together in order to form an aggregate accurate classifier. Different values of accuracy have been achieved between 88 percent and 95 percent for graphics versus photos versus text The same method has been applied by the author to address the problem of indoor, outdoor and close-up images classification reporting precision values between 87 percent and 91 percent.

The system described in [6] use several features to evaluate the differences between photographs and graphics. A dataset of GIF and JPEG images is used but they only search for 'simple' graphics such as navigation buttons icons or drawings. The features used include farthest neighbor, saturation, the number of colors, most prevalent color, farthest neighbor histogram, and a few more. An error rate of 9 percent is reported for distinguishing JPEG encoded images. They combine single classifiers using multiple decision tree. The best performances have been achieved with GIF graphics but since the dataset used is unbalanced (the GIF images are about 63 percent of the dataset) the results can be not really accurate.

In the project described by Lienhart and Hartmann [18], an algorithm to distinguish actual photographs from computer-generated but realistic-looking images has been developed. Because the computer-generated images (such as screen shots from photo-realistic computer games or ray tracing images) have a lower level of noise than real photos, they measured the amount of noise using histograms of the difference between the original image and its denoised version. This feature can distinguish between actual photos and computer-generated images.

Tian-Tsong Ng et al. addressed the same problem of Lienhart and Hartmann in [19]. They proposed a geometry-based image model, motivated by the physical image generation process, to address the problem. The suggested model reveals certain physical differences between the two image categories, such as the gamma correction in photographic images and the sharp structures in computer graphics. Although there are many project on this problem, the existing methods are not scalable and cannot be used on a large image collection. This is, for example, because the computational cost of some algorithms

is very high and the per-image feature extraction time of [19] is more than 50 seconds. Moreover some features used before are not robust enough to noise.

1.3 Limitations and Constraints

The main purpose of this Master's Thesis is to find and create a system for image classification. Since the system has been conceived to be integrated in a global image search engine, it has been developed under speed and scalability constraints.

The manually labeled images dataset has been created downloading random images from the internet. Even though the images have been chosen with the aim of having a rich and significant dataset and it has been tried to avoid redundant data, some kind of natural or synthetic images might be not represented. Thus, the behavior of the classifier could be not optimal in some particular cases for those images that are missing in the dataset.

Images compressed with a lossy method such as JPEG have some of their features altered. For example compression artifacts can add noise where an uniform area is expected (especially in synthetic images), a soft and fading edge can be replaced with a sharp one. Because of these modification of the pictures, performances of some of the classifiers differ from the ideal case and thus, the error rate is higher than using not compressed images.

For example, images are usually compressed and resized in the web environment. Due to the interpolation used in the resizing process, the number of unique colors could greatly increase. So the performance of the feature using the number of colors would degrade significantly.

1.4 Method

Thinking of what the main differences between photographs and graphics are, we can see that something very simple comes into our mind: graphics are typically generated using a limited number of colors and usually containing only a few areas of uniform colors. Moreover, highly saturated colors are more likely to be used. Sharp edges also are typical feature characterizing synthetic images that can be used by an image classifier. It is possible to easily spot these characteristics in maps figure 2.4f, charts figure 2.4e, logos figure 2.4 and cartoons figure 2.4d On the other hand, very often a photograph depicts real life objects and subjects. These have usually textures, smooth angles, larger variety of colors but less saturated. Because of the way a photograph

is acquired and the way a camera works, natural pictures also results in being more noisy.

For an human being, distinguishing between a photograph and a graphic is almost always an easy task. It is often just matter of a glance. Unfortunately it is not for a computer: there is not a simple and easy feature to just extract and process. Noisiness, sharpness of the edges saturation must be derived from the raw data available from the picture in different ways even using complex structures such as histograms or correlograms. Those features must be then combined together in order to build a solid classifier since if used individually, they can lead to poor or even wrong results.

In figure 1.1 are the summarized steps in order to classify an image:

- Extract different features that give an individual classification.
- Combine them together using a classification method in order to boost the performance of the single classifiers.
- Compare the output and assign the image to a category.

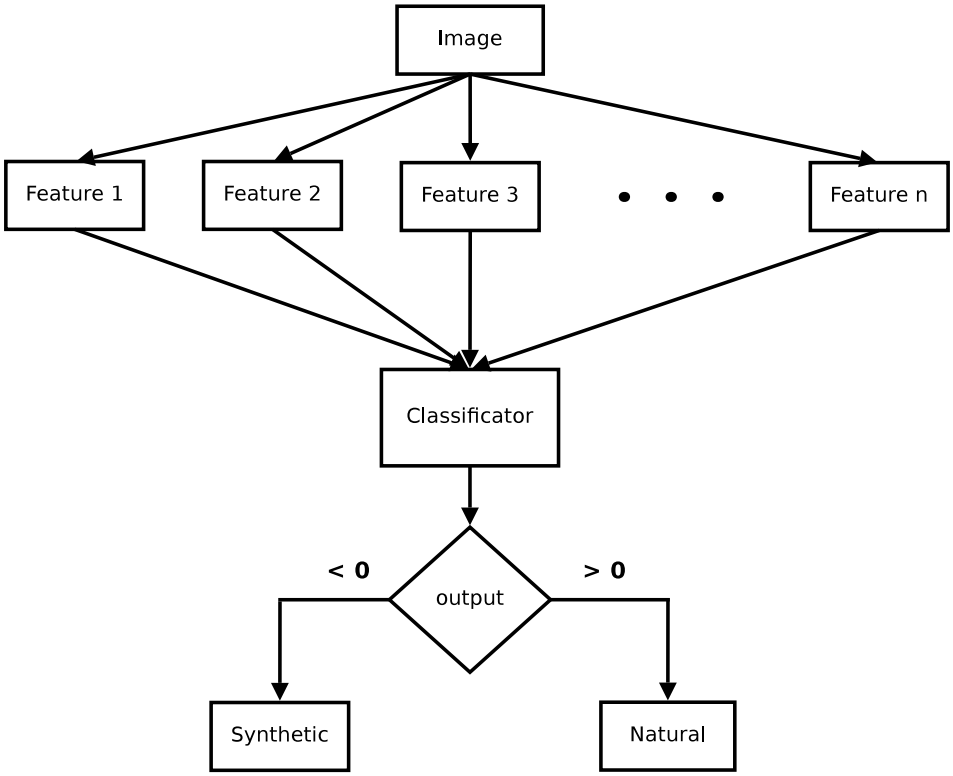


Figure 1.1: Outline of the main algorithm. After extracting a set of features, those are combined together in an aggregate classifier. The output is then compared to the threshold and the image is classified according to this result.

1.5 Thesis Outline

This section outlines the structure of the thesis.

- **Chapter 2 Images Dataset** introduces the dataset used and the criterion for building it. It also includes some examples of each class and of images that were not possible to classify because they had both natural and synthetic features.
- **Chapter 3 Image Features** treats in detail the features extracted from the raw image in order to build simple classifiers and discuss their pros and cons with respect to the accuracy and the different .
- **Chapter 4 Classification Method** introduces adaBoost, decision tree and neural network methods used in order to combine single classifiers described in the previous chapter into a more accurate one.

- **Chapter 5 Performance** presents and discuss different performances achieved with different classification methods. It also highlights the main reason of such results and how it might be possible to improve performances.
- **Chapter 6 Conclusion and Further Work** concludes the thesis with a discussion of the performances and proposes some further work that can be developed in the future such as video classification.

Chapter 2

Images Dataset

The images dataset used during the project consists of 6000 full color *JPEG* images. Some of them have been automatically downloaded from some photographs sites, some have been manually downloaded by searching with PicSearch [4] engine for different kind of images. The downloaded images were then hand-labeled as "Natural" or "Synthetic".

The images dataset consists of only *JPEG* images. This chose was made because of a main assumption: since images can be compressed in a different way, the classification algorithm has to be tuned accordingly to each format: multiple factors have to been taken into account such as the reduced/altered palette and the noise added while resizing/compressing. Hence a mixed dataset can not be used with a single classification method. *GIF* and *PNG* formats are very popular formats for synthetic images but less used for encoding photographs. For transmissions of true color images over the internet, especially photographic ones, *JPEG* is almost always a better choice [3]. For this reason, the *JPEG* format has been chosen since it is possible to find a large number of natural and synthetic images.

Since at the beginning it was easier to automatically retrieve natural-like images than synthetic ones, a first test has been done using an imbalanced dataset (70 percent natural, 30 percent synthetic). This setup showed a very good accuracy for natural images and a poor accuracy for synthetic images because the training methods were influenced by the greater number of natural images. For this reason a balanced dataset is preferred: the final dataset is formed by 3000 natural and 3000 synthetic images.

The size of images is neither too large nor too small, it vary from 100x100 pixels to 1000x1000 pixels and the mean size is (516 ± 139)x(593 ± 165). The scatter plot in figure 2.1 represents the size of the images in the dataset.

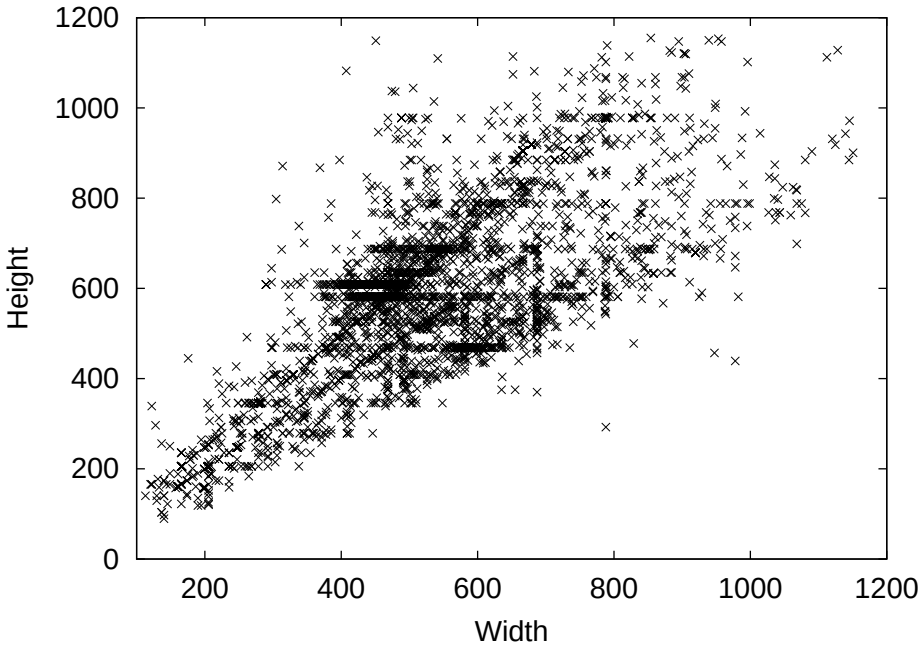


Figure 2.1: Image size distribution. The most frequent size in the dataset is s (516 ± 139)x(593 ± 165).

2.1 Natural Images

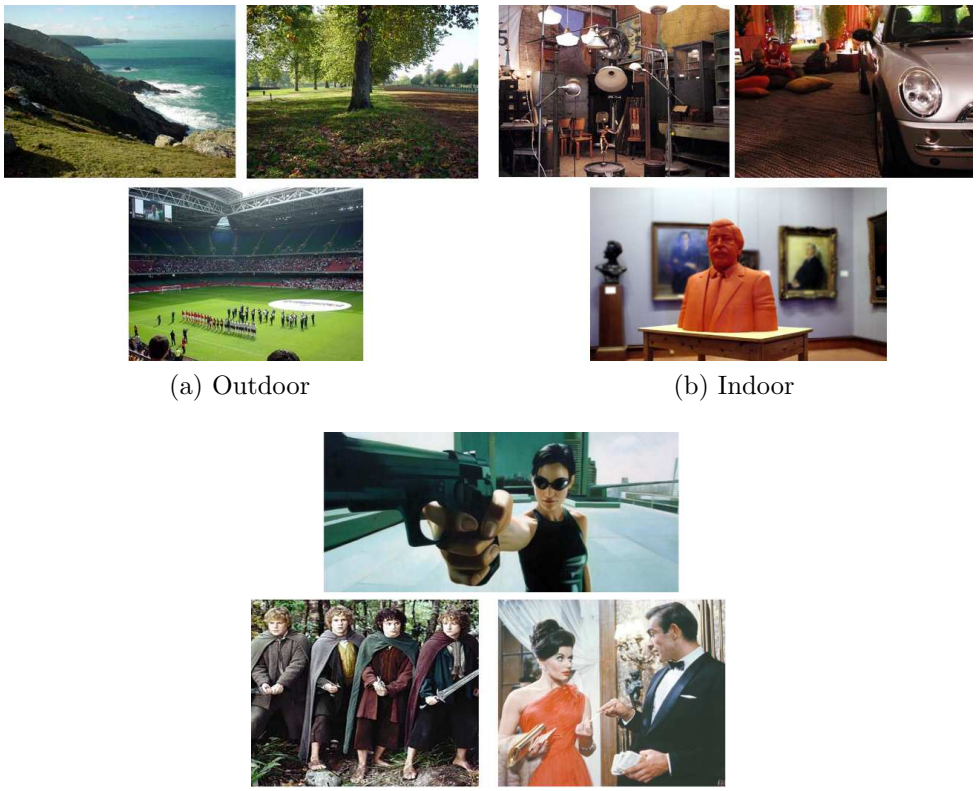
Most of the photographs were downloaded automatically from different photographic websites using wget and then manually labeled. The images were then labeled as *Natural* if belonging to one of the following categories:

- Outdoor
- Indoor
- Movies

Example of natural images is shown in figure 2.2

2.2 Synthetic Images

Since it was not possible to find any website where to get suitable synthetic images, a manually search has been done using the PicSearch [4] image search



(a) Outdoor

(b) Indoor

(c) Movies

Figure 2.2: Examples of Natural Images.

engine. Queries such as “Walt Disney cartoon”, “test chart”, “local map” have been used and then suitable images were chosen and downloaded. The images were then labeled as *Synthetic* if belonging to one of the following categories:

- Drawings
- Computer game screenshots
- Window application screenshot
- Cartoons
- Logos
- Charts
- Maps

Example of synthetic images is shown in figure 2.4

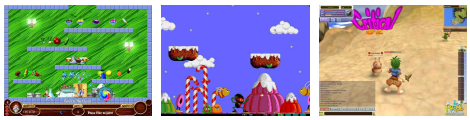
2.3 Ambiguous Images

Downloading and manually labeling natural and synthetic images took some time and moreover some images were difficult to classify because they had both natural and synthetic component such a screenshot of an editing software with a photograph figure 2.3a, or photograph edited with a digital frame figure 2.3c, or a natural object in a synthetic environment figure 2.3c. Images that could not be classified because of being ambiguous were not included in the dataset.

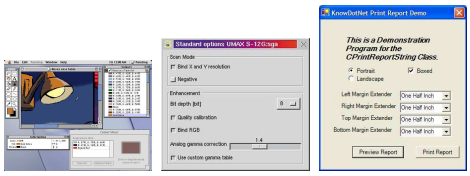
(a) (b) (c)

Figure 2.3: Examples of ambiguous images. It is hard to determine an unambiguous classification.

(a) Drawings



(b) Computer Game Screenshots



(c) Window Application Screenshot

(d) Cartoons

(e) Charts

(f) Maps

(g) Logos

Figure 2.4: Examples of Synthetic Images.

Chapter 3

Image Features

The main step in order to be able to classify an image is to extract numerical features from the raw data. These common features will be then combined in different ways according to different classification methods. ImageMagick [2] libraries for C++ have been used for images handling. The following features have been extracted:

- Different colors ratio
- Most used colors
- Saturation average
- Saturation pixels ratio
- Gray histogram smoothness
- Spatial gray level dependence
- Colors correlogram
- Farthest neighbor histogram

These single features exhibit promising performance with low computational cost. Since each classifier has its own weakness and strengths, better performances can be achieved using more then one feature together. In the following section, i will describe and analyze in detail characteristics and performance of different single classifiers used during the project.

3.1 Colors

Color transitions from pixel to pixel have different models in photographs and graphics. Photographs depict objects of the real world, and in such a context, it is not common to find regions of constant color because objects tend to be shaded and have texture. In addition, during the process of taking a photograph, some noise is added to the subject and that causes neighbor pixels to have different RGB values (even when they are supposed to have the same color).

It is possible to exploit these simple features related to colors by extracting and analyzing the following features.

3.1.1 Number of Different Colors

Photographs often have more color than graphics. This is because synthetic images tend to have large uniform regions with the same color. On the Web in particular, graphics with few color are more popular because they compress better.

The number of different colors of an image is extracted but it cannot be directly used as metric since the raw number is also dependent from the size of the image. Therefore a more accurate metric is used: the rate between the number of different colors and the number of total pixels. A scan of the pixels matrix is performed in order to count different colors. This value is then divided by the total number of pixels.

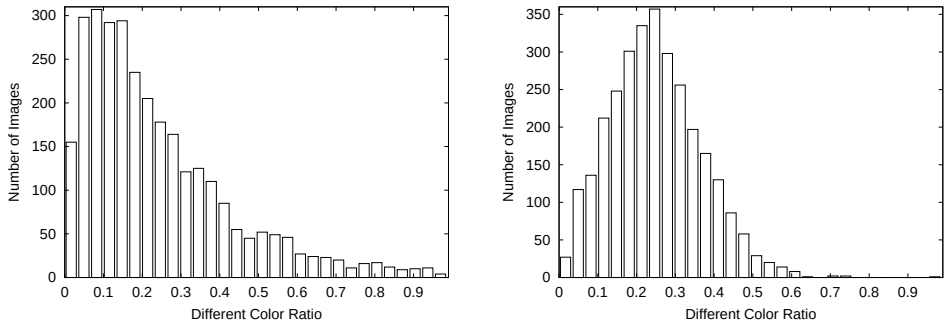
$$differentColorsRatio = \frac{nDifferentPixels}{nTotalPixels}$$

If the ratio is between 0.16 and 0.49, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 35 percent.

In figure 3.1 is shown the distribution of images by different color ratio.

3.1.2 Saturation Average

Photographs depict objects of the real world and highly saturated objects are not very common. Certain colors are much more likely to appear in graphics than in photographs. For example, cartoons, maps, charts and logos often have large regions covered with highly saturated colors. Those colors are much less frequent in photographs. Some natural images though can have a big saturated area and be misclassified by this metric. For example a picture



(a) Synthetic (b) Natural

Figure 3.1: Different Colors Pixels Ratio. Most of the natural images have a ratio between 0.16 and 0.49.

of people with bright yellow jackets (figure 3.2a) or a picture of a red sport car (figure 3.2b).

Given RGB values, the saturation level of a pixel is defined as the greatest absolute difference of values between red green and blue.

$$saturation = max(abs(red - green), abs(red - blue), abs(green - blue))$$

The average value of the saturation of all pixels in an image is calculated. If it results between 9.44 and 47.22, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 28 percent.

In figure 3.3 is shown the distribution of images by different saturation average.

3.1.3 Highly Saturated Pixels

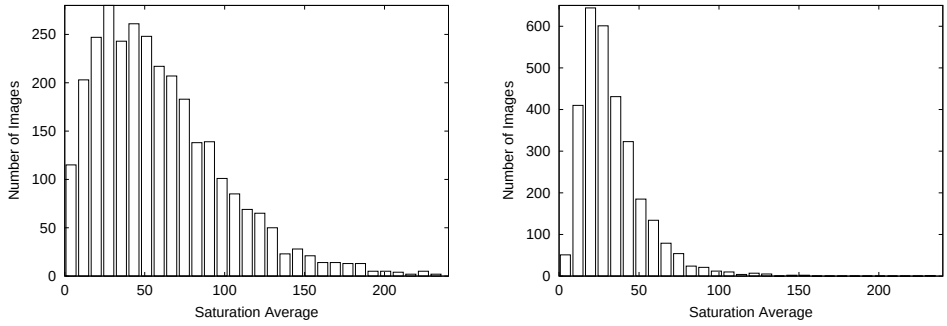
Another feature that can be derived from the saturation is the ratio of pixels with a saturation value greater than a given threshold. Values of threshold between 10 and 250 have been tested. figure 3.4 shows the results of the test: the lowest error rate has been obtained with a value of threshold of 80.

The number of pixels with a saturation level greater then 80 is divided by the number of total pixels for a given image. If it results lower than 0.1, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 27 percent.

In figure 3.5 is shown the distribution of images by different ratio of pixels having a saturation value greather than 80.

(a) (b)

Figure 3.2: Examples of Photographs With Highly Saturated Colors. These photographs are misclassified as synthetic because of their high saturation.



(a) Synthetic (b) Natural

Figure 3.3: Saturation Average. Most of the natural images have saturation values between 9.44 and 47.22.

3.1.4 Most Used Colors

By analyzing the most used color in an image it is possible to extract some interesting features:

- **Raw value (RGB).** The same assumption as before is made: It is expected that some colors being more frequent in synthetic images rather then in photographs. Table 3.1 shows the value of the threshold for each color in the RGB color space and the error rate achieved using that threshold. Images with color values lower than the threshold are classified as natural. In figure 3.7 is shown the distribution of images by RGB values of the most used color.

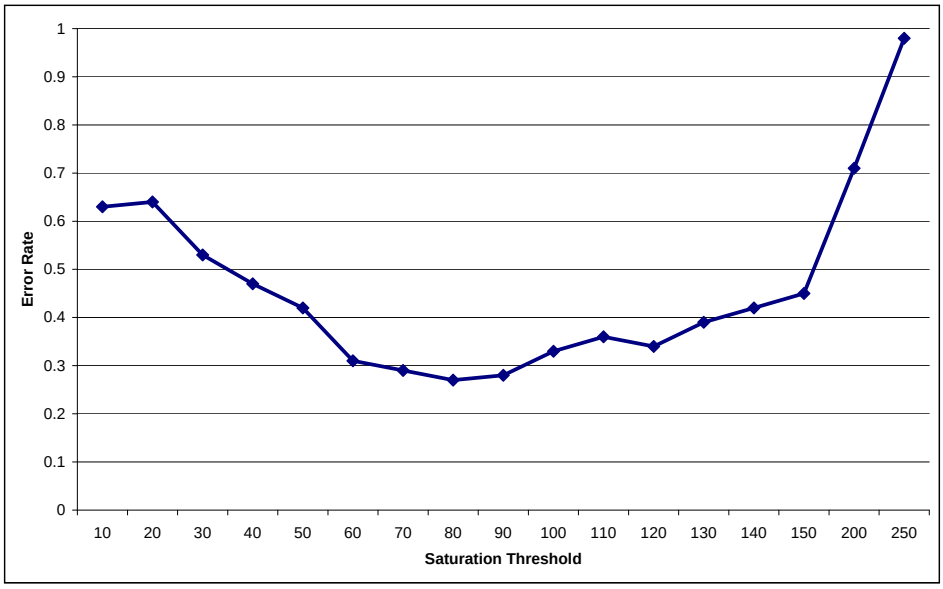
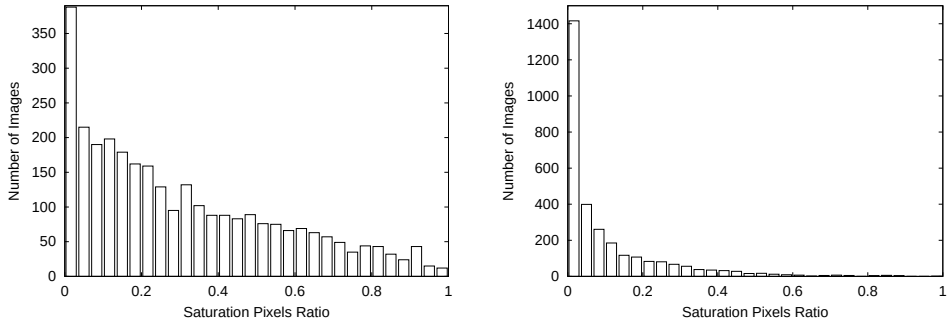


Figure 3.4: Error Rate vs Saturation Threshold. The lowest error rate of 0.27 is obtained using a threshold of 80.



(a) Synthetic (b) Natural

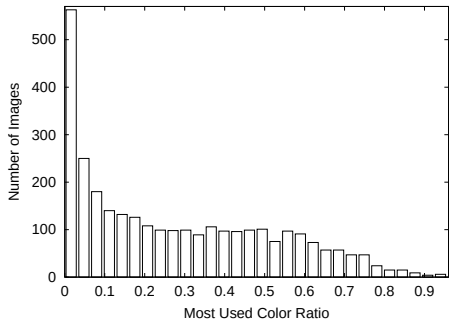
Figure 3.5: Highly Saturated Pixels Ratio. Most of the natural images have a ratio lower than 0.1.

Color	Threshold	Error Rate
Red	24.8	0.36
Green	230	0.42
Blue	148	0.43

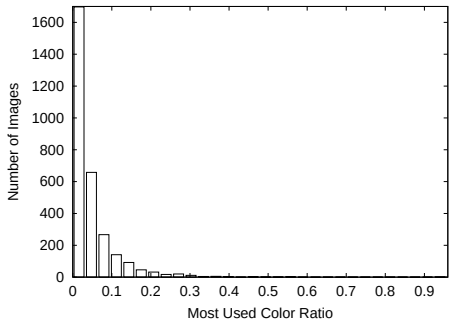
Table 3.1: Most Used Colors. The red component gives a more accurate classification than Green and Blue

- **Fraction of pixels having the most used color.** This is an alternative way of exploiting the presence of large uniformly colored area in synthetic images. The number of pixels having the most common color is divided by the total number of pixels. If it results lower than 0.11, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 21 percent. In figure 3.6 is shown the distribution of images by different ratio of pixels having the most used color.

- **Saturation.** The saturation level is calculated with the same method described in Chapter 3.1.2. If it results lower than 147, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 38 percent. In figure 3.8 is shown the distribution of images by different value of saturation of the most used color.

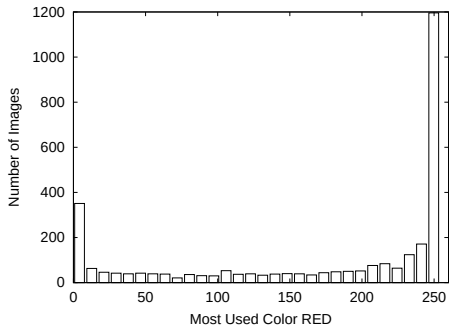


(a) Synthetic

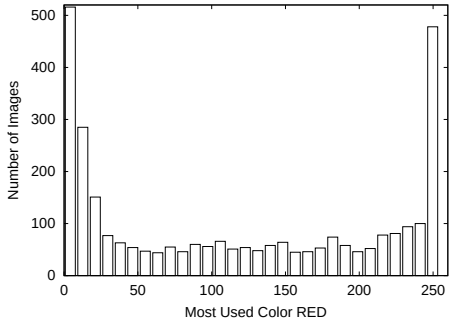


(b) Natural

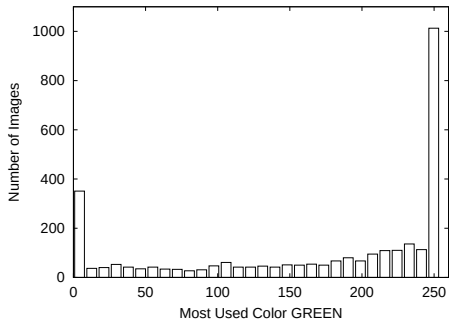
Figure 3.6: Most Used Color Pixels Ratio. Most of the natural images have a ratio lower than 0.11.



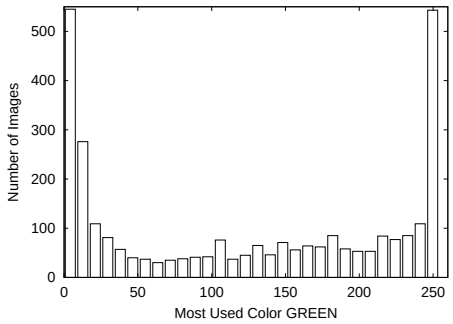
(a) Synthetic



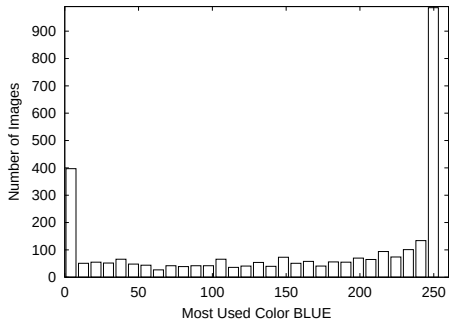
(b) Natural



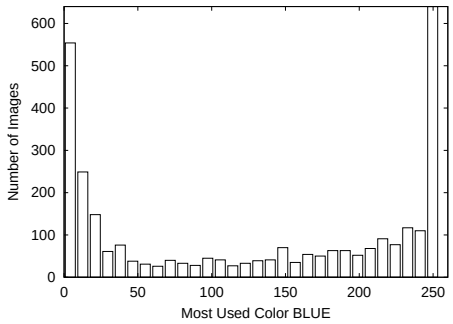
(c) Synthetic



(d) Natural

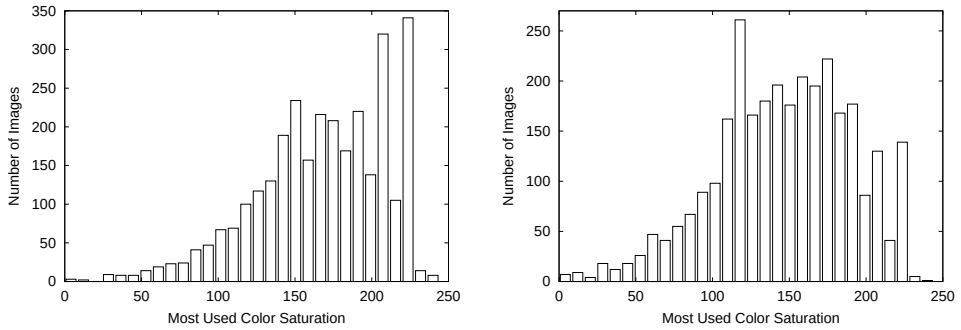


(e) Synthetic



(f) Natural

Figure 3.7: Most Used Colors (red, green, blue). The thresholds are shown in Table 3.1



(a) Synthetic (b) Natural

Figure 3.8: Most Used Color Saturation. Most of the natural images have a saturation value lower than 147.

3.2 Space correlation

The color analysis gives some good methods for the classification of images. Though, it does not take in account the spatial correlation of pixels. Different methods have been tested in this project in order to exploit the different sharpness of the edges and noisiness in pictures.

Sharp transactions in real life objects are not common. In photographs, edges are often faded and blend in with the background or with other objects. As opposed to this, objects in synthetic images, tend to have very sharp edges.

As noisy nature of photographs has been assumed, in a natural image even pixels that are in the same colored area may have a different RGB value.

3.2.1 Spatial Gray-level Dependence

Since a simple specific feature for sharpness does not exists, a Spatial Gray-Level Dependence (SGLD henceforth) Histogram [20] has been used. The structure used for analyzing SGLD is a two-dimensional histogram. If a pair of pixels is part of a flat colored area having the same brightness, an entry nearby the diagonal of the SGLD matrix is incremented. On the other hand, if the pair is on an edge, the 2 pixels will have a significant difference in brightness if the edge is sharp but they will have similar brightness if the edge is not sharp. The incremented entry is therefore far from the diagonal in the first case and somewhere in the area between the diagonal and the outer corner of the diagram in the second case.

In order to extract this feature, the preliminary step is to transform the

color space of the into gray scale. The value of brightness for every pixel is then available. The brightness is defined as the average of the red, green and blue color components of an image.

$$brightness = \frac{red + green + blue}{3}$$

The values of the SGLD matrix are incremented following the Algorithm 1.

Algorithm 1: SGLD Matrix Update.

Let: $SGLD$ be an integer matrix of 256x256 elements

Initialize $SGLD[][] = 0$

:

for *each pixel in image* **do**

  for *each neighbor of pixel* **do**

    $SGLD[pixel.getBrightness()][neighbor.getBrightness()] + +$

  end

end

Given a pixel, it is paired with all its 8 neighbors. This means that top-left, top, top-right, left, right, bottom-left, bottom and bottom-right neighbor are considered

For each pair (pixel, neighbor) the brightness β_p, β_n is extracted and the value in position (β_p, β_n) is incremented. At the end of this first step, the SGLD matrix is populated with the information about the brightness similarity of all contiguous pairs of pixels.

For example, if 2 pixels (p, n) have similar brightness $(\beta_n \cong \beta_p)$, the SGLD matrix will be incremented at the index $[b1, b2]$ that is very close to the diagonal (because $(\beta_n \cong \beta_p)$). These 2 pixels are likely to be either part of the same colored area, or part of a very smooth edge. If 2 pixels (p, n) have very different brightness $(\beta_n \gg \beta_p)$, the SGLD matrix will be incremented at the index $[\beta_n, \beta_p]$ that is far from the diagonal (since $(\beta_n \gg \beta_p)$). These 2 pixels are likely to be part of a strong edge. It has been supposed that synthetic images have stronger edges and more uniformed colored area so it is fair to expect having many pixels either around the diagonal or very far from it. In the case of natural image, it is expected to have more pixels in the noisy area between the diagonal and the border but not very far.

The second step is to analyze this matrix and extract a numeric value that gives an estimate of edge sharpness. Since the purpose of the SGLD analysis

is edge sharpness it is needed to filter all points that are not related to edges. Around the diagonal there are only values that have the same (or very similar) values of brightness, thus being more likely to be part of an homogeneous area. By filtering all values nearer to the diagonal than a certain threshold and by calculating the average distance from the diagonal of the remaining pairs, it is possible to extract Spatial Gray-Level Dependence average. Thresholds values between 1 and 30 have been tested. figure 3.9 shows the results of the test: the lowest error rate has been obtained with a value of threshold of 16.

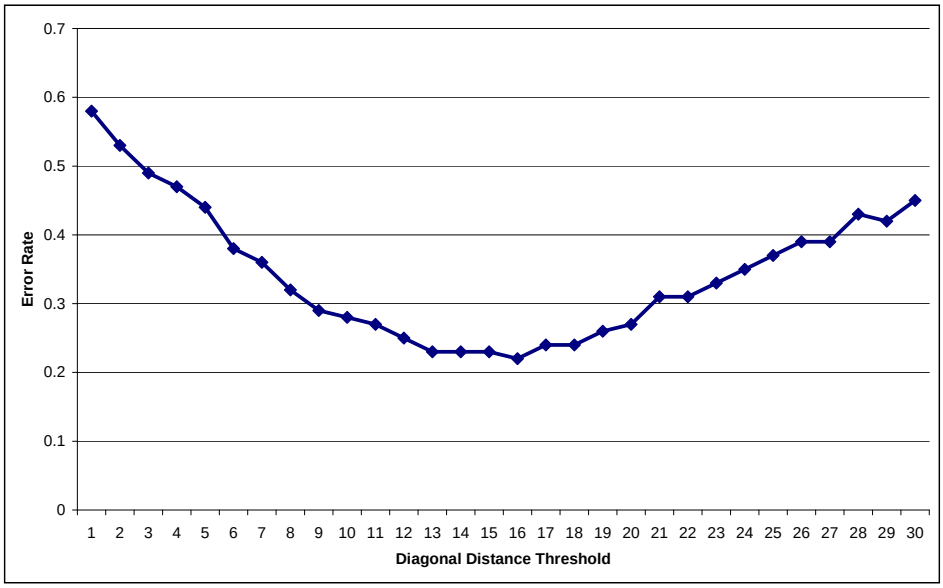


Figure 3.9: Error Rate vs Distance From The Diagonal Threshold. The lowest error rate of 22 percent is obtained using a threshold of 16.

The average obtained is then used as metric. If it results lower than 49, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 22 percent.

As illustrated in the examples given in figure 3.10, the SGLD histograms are different for graphics and pictures. Apart from the surrounding of the diagonal, where many pairs are located for both kind of images, natural and synthetic images have different distribution of points in the SGLD histogram.

Synthetic images have a many entries very far away from the diagonal (even the limit points (0, 255) and (255, 0))

3.2.2 Farthest neighbor

The matrix obtained is similar to the one built for the color correlogram. The feature measured with this method is related to the sharpness of the edges as well but gives a different point of view. It is possible to have an estimate of the color transitions in graphics and photographs.

The values of the matrix are calculated following the Algorithm 2.

Algorithm 2: Farthest Neighbor Array Creation

Let: $farthestNeighbor$ be an integer array

Initialize $farthestNeighbor[] = 0$

:

for *each pixel in image* **do**
  for *each neighbor of pixel* **do**
    if *abs(pixel.color()-neighbor.color()) > farthestNeighbor[pixel]* **then**
      $farthestNeighbor[pixel] = abs(pixel.color() - neighbor.color())$
    end
  end
end

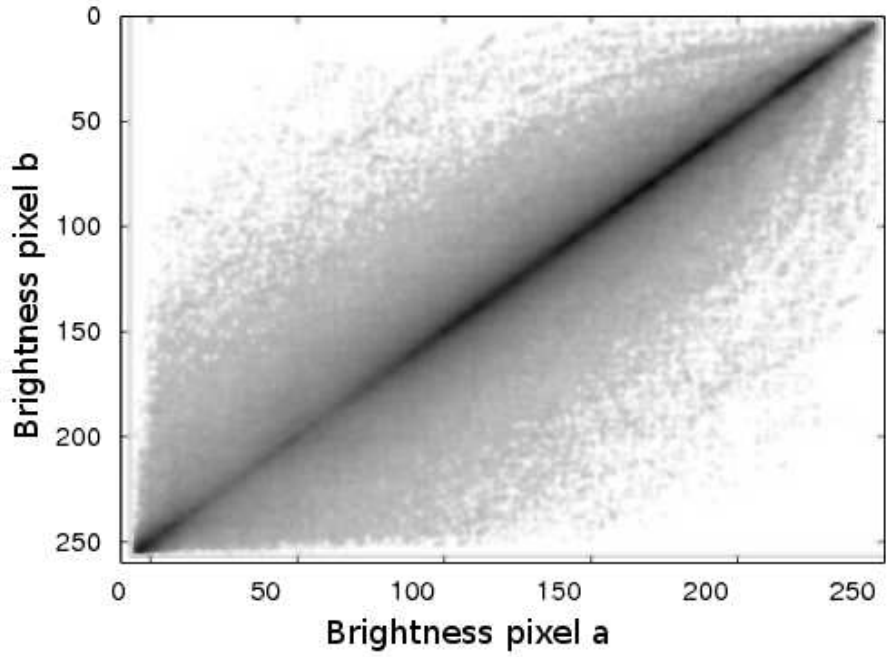
Given any pixel $p1$ and its neighbor $p2$, let r, g, b be the red, green and blue color components, the color distance is defined as follow:

$$distance = \sqrt{(r1 - r2)^2 + (g1 - g2)^2 + (b1 - b2)^2}$$

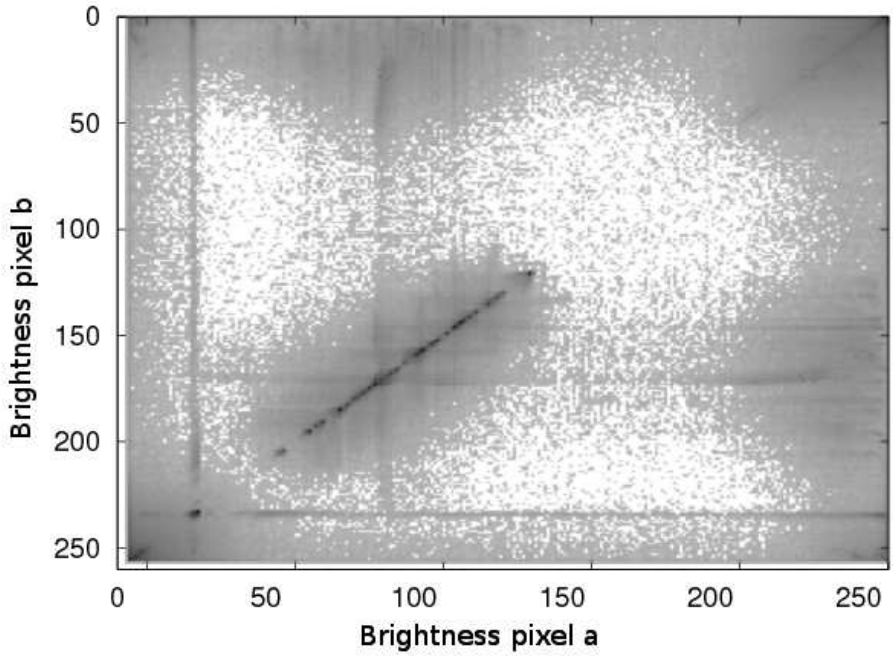
That results in a lower average value for this metric for natural images. If it is less than 46, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 31 percent.

The rationale behind this test is to see how abrupt the color changes are in an image. In photographs they tend to be smoother than in drawings. Therefore, drawings tend to score higher than photographs in this test

This metric is based on the assumptions made about color transitions in graphics and photographs.



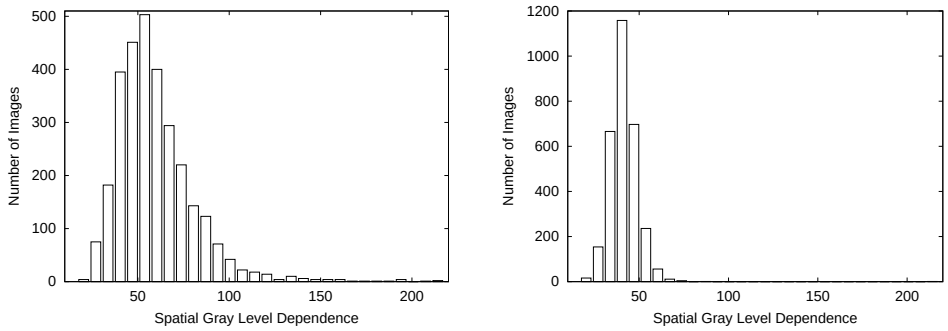
(a) Natural



(b) Synthetic

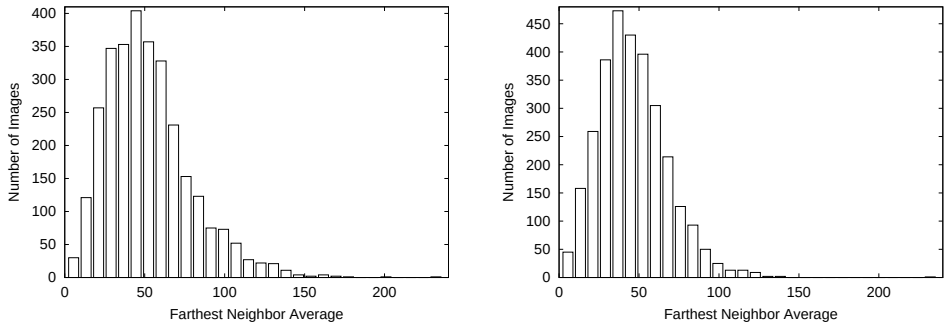
(a) Natural and Synthetic Pictures

Figure 3.10: Examples of SGLD histogram. The typical histogram derived from natural images has values only around the diagonal.



(a) Synthetic (b) Natural

Figure 3.11: Spatial Gray Level Dependence. Most of the natural images have a value lower than 49.



(a) Synthetic (b) Natural

Figure 3.12: Farthest Neighbor Average. Most of the natural images have an average value lower than 46.

3.2.3 Color Correlogram

As introduced before, because of the way a camera takes pictures and because of the nature of the real life objects depicted by the photographs, natural images results in being more noisy than synthetic ones. In order to exploit this feature, a color correlogram [17] is used.

A color correlogram of an image is a two-dimensional matrix where, for each row and column (r, c) is stored the probability that a given pixel (r, c) has the same color (or within a certain threshold) as his neighbors. A color correlogram describe the global distribution of the local spatial correlation of colors. This kind of feature is not possible to obtain using a simple color histogram since it only gives only information about the color distribution in an image but not about spatial correlation.

Since looking for exact pairs of colors was an option too restrictive and the results were not discriminating enough, a tolerance for the color similarity has been used. Values of tolerance between 0 and 10 have been tested. figure 3.13 shows the result of tests. The best performance are achieved using a tolerance of 2 for each R, G, B color component.

The algorithm used is summarized in the Algorithm 3.

Given any pixel, its color is compared with all its 8 neighbors. This means that top-left, top, top-right, left, right, bottom-left, bottom and bottom-right neighbor are considered. The probability that a neighbor pixel has the same color is then calculated as the number of neighbors having the same color divided by the number of neighbors (8). Then, the average value is calculated and used as metric. If it is between 6 percent and 41 percent, the image is

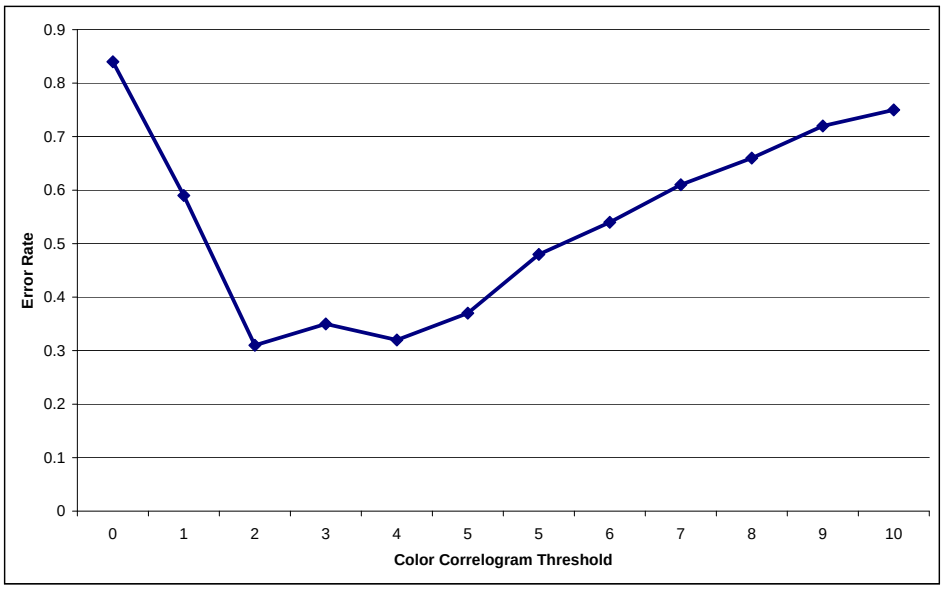


Figure 3.13: Error Rate vs Color Similarity Threshold. The lowest error rate of 32 percent is obtained using a tolerance of 2.

Algorithm 3: Correlogram Array Creation

Let: *correlogram*[] be an integer array

Initialize *correlogram*[] = 0

:

for *each pixel in image* **do**

  for *each neighbor of pixel* **do**

    if *abs(pixel.color() - neighbor.color()) < threshold* **then**

      correlogram[*pixel*] + +

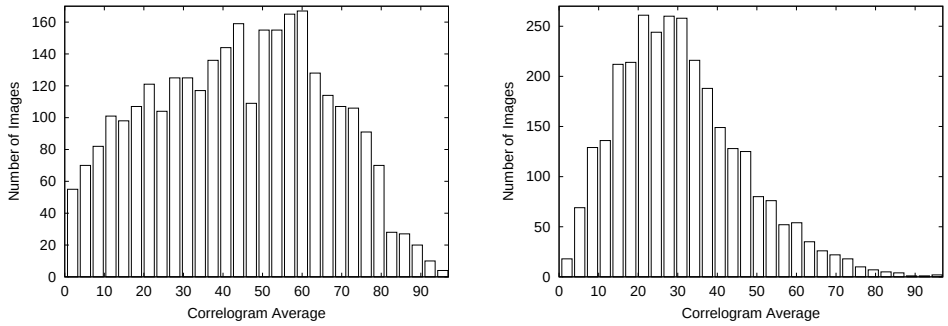
    end

  end

  correlogram[*pixel*]/ = 8 ∗ 100

end

classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 31 percent. This summarize the fact that generally, natural images are noisy and with fading borders and, on the other hand, synthetic images are characterized by stronger edges transaction and uniformly colored area.



(a) Synthetic (b) Natural

Figure 3.14: Correlogram Average. Most of the natural images have an average value between 6 percent and 41 percent.

3.2.4 Gray Histogram

The gray histogram is the gray scale version of the color histogram. It represents the distribution of colors in an image, derived by counting the number of pixels of each of all gray intensity. The analysis of this histogram can give an estimation of the distribution of area having the same color/brightness. In figure 3.15 it is possible to see the typical difference between a natural picture histogram and a synthetic one. The natural histogram example, figure 3.15b, shows an overall smooth trend. The peaks (if any) are not really sharp. This means that the picture has an uniform brightness. In the synthetic histogram example figure 3.15a, there are some high and narrow peaks due to the fact that in the picture there are lot of areas having the same brightness.

The method used to build the histogram array is simple. For each pixel in the image, the value of the array at the index corresponding to the pixel brightness (0 to 255) is incremented by 1. The histogram is then normalized by dividing the value of all bins by the number of pixels. Once the histogram is done, a value estimating the smoothness is calculated using the Algorithm 4.

This value is lower for natural images having a smooth gray histogram and higher for synthetic images having an histogram with more spikes. If the sum

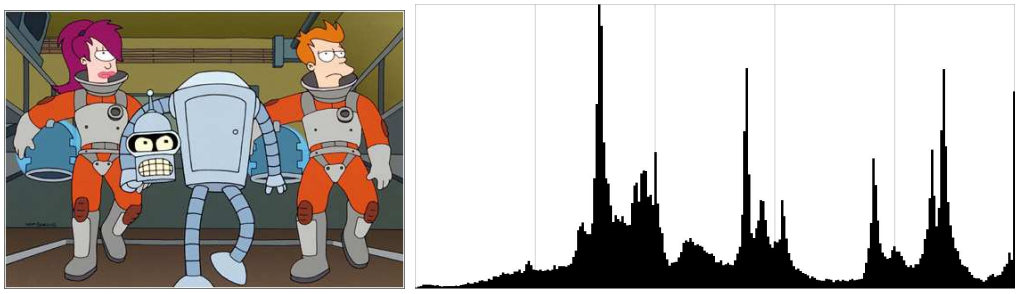
Algorithm 4: Correlogram Smoothness Evaluation

Let: grayHistogram be the normalized gray histogram array

for *i=0, i<256-1, i++* **do**
$smoothness+ = abs(grayHistogram[i] - grayHistogram[i+1])$
end

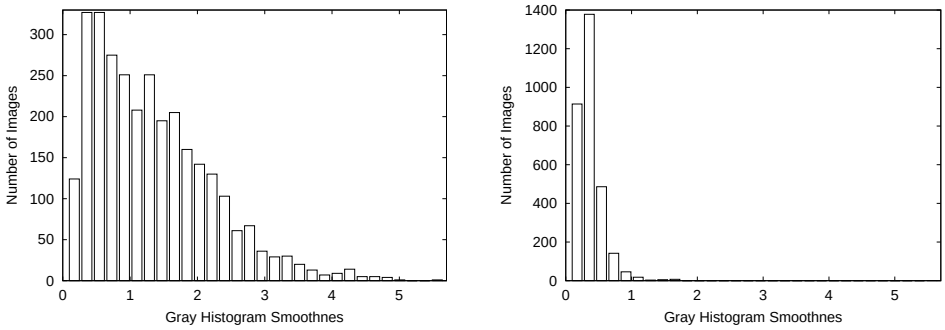
is less than 0.27, the image is classified as natural, otherwise it is classified as synthetic. The error rate of this classifier is 17 percent This is the single classifier that gives the best accuracy.

(a) Natural



(b) Synthetic

Figure 3.15: Example of Gray Histogram.



(a) Synthetic (b) Natural

Figure 3.16: Gray Histogram Smoothness

Chapter 4

Classification Methods

Given a set of features and a training set of positive and negative images, different machine learning approaches could be used to learn a classification rule. All the classifiers described in the previous chapter have their own weakness. Moreover, the individual error rate is high. In order to improve the accuracy, a method that combine all those single classifiers is needed. 3 different methods have been tested during the project. These technique using different approaches but have the same purpose to build an improved classifier based on several weaker classifiers combined together.

The actual classification is expressed as a value between -1 and 1. Extreme values indicates an higher confidence. Images which are not easily classified have a value around 0.

4.1 AdaBoost

Boosting is a general method used to improve the performance of a learning algorithm. It combines classifiers which can be not very accurate, even just a little bit better than random guessing. The first provable effective boosting algorithms were presented by Shapire [24] and Freund [13]. It construct an ensemble of classifiers and use a voting mechanism for the classification.

AdaBoost (Adaptive Boost) is considered adaptive because it builds a general classifier by tweaking the weight of single classifiers using samples of the data set that were previously misclassified. Though it is sensitive to noisy data and outliers, it is less susceptible to the overfitting problem than most other algorithms.

4.1.1 Training

The concept on which AdaBoost is based is simple. During each iteration, the classifier having the lowest error rate with respect to the current distribution of weights is chosen. Then, a coefficient α_t is calculated so that it will locally minimize the error function. The weights array is then updated in the following way: the weight is decreased for images correctly classified and increased for images misclassified, so that the new classifier focuses more on those examples. This allows to intuitively measure the relevance of each single classifier and avoid redundant classifiers. This steps are repeated an arbitrary number of times. The more it runs, the more accurate it is on the training set but, if trained too much, it can happen that it will overfit. This can be reduced by limiting the number of steps to a value that optimize the accuracy and minimize the overfitting rate. The algorithm used is a variation of the original proposed by Freund and Schapire [14].

Algorithm 5: AdaBoost

Given: a previously labeled dataset of m images.

Let: x_i be the image i and y_i be the class (1 for natural images and -1 for synthetic) of the image i, $D_t(i)$ be the weight distribution for the image i at the step t and T be the number of iteration.

Initialize $D_1(i) = \frac{1}{m}$ for each image i.

:

for $t = 1, ..., T$: **do**

1: Get the classifier h_t that minimize the error ϵ_t with respect to the distribution D_t.

if $error < 0.5$ **then**
stop

end

2: $\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$ according to the error rate ϵ_t

3: $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_y(x_i)}}{Z_t}$

end

Each round, $t = 1, ..., T$:, AdaBoost evaluates the accuracy of each weak classifier on the training data set based on the current distribution of weights over the samples $D_t(i)$. A parameter α_t is calculated and assigned to the classifier with the lowest error rate h_t. This parameter can be intuitively seen

as the measure of the importance that the chosen classifier has over the final classification. The weights array is updated so that the incorrectly classified examples are rated more than the correctly one. In this way, the observation that the previously classifier poorly predicts receive greater weight on the next iteration. After T iterations, the algorithm stops and the final classifier consists of a linear combination of the single classifiers multiplied by their importance defined as:

$$H(i) = \left(\sum_{t=1}^{T} \alpha_t h_y(x) \right)$$

4.1.2 Classification

Once the algorithm is trained, an array of coefficients (weights), one for each classifier, is created. The classification is then easy to calculate. All features are extracted from a given image and an aggregate classification is performed. The resulting value of the single classification (+1 or -1) is then multiplied by the weight calculated during the training phase. These values are then added together and if the result is greater than 0, the image is classified as natural, otherwise it is classified as synthetic. In figure 4.1 these steps are shown.

This method has the peculiarity of excluding redundant classifiers. In fact if two classifiers have the same behavior, one of them will not be taken into account. Therefore, after the training part, it is possible to exclude some of the single features that do not give additional information to the classifier maintaining the same grade of accuracy. This can dramatically decrease the time needed to classify an image.

The problem of over-fitting has not been addressed in the early work on boosting. However, experiments using AdaBoost [9, 11, 21] have shown that it is highly resistant to over-fitting. Schapire et. al. [25] give a theoretical explanation of this fortunate phenomenon. There is currently much debate on whether or not this explanation is exhaustive [10, 16]

4.2 Decision Tree

In machine learning, a decision tree is a predictive model that maps some features to an output trough a succession of conditional statements. In a tree structure, leaves represent classifications and branches represent conjunctions of features that lead to those classifications.

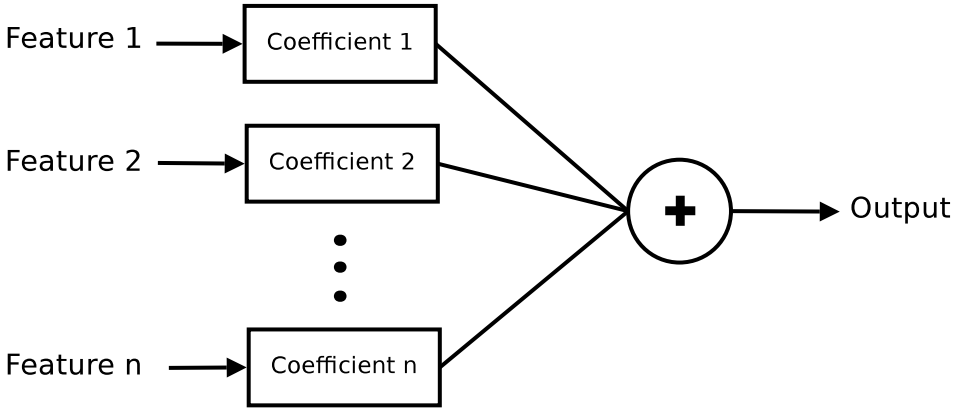


Figure 4.1: AdaBoost Classification Diagram. The output is the sum of all the single features multiplied by a coefficient.

4.2.1 Training

The Algorithm 6 explains the training steps for the tree. These steps are recursively performed starting from the root node.

A single classifier is taken into account only one time from the root node to a leaf node. The classifier c_i that performs best is chosen among the list c of classifiers that are still available. Then the dataset is split into 2 subsets composed by images classified as synthetic and natural by c_i. The list c updated and c_i is removed. The 2 subsets are then used as base learning set for the 2 children nodes. They are also used to calculate the local accuracy of the nodes. The accuracy coefficient gives an estimation of with respect to natural images, if trained with d_n, and synthetic images if trained with d_s. It is the very accuracy of the classifier c_i on the current subset. When all subsets and all classifiers are used, the algorithm ends and the decision tree is ready.

4.2.2 Classification

Each node has an individual coefficient based on the error rate that the chosen classifier has with respect to the current subset. 2 temporary global coefficient β_n and β_s are used respectively for the probability that, on a given point, the image is natural or synthetic. Given an image and all its features, for each node, starting from the root, the local classifier is used. If the image results natural, the next node will be the node that during the training step was assigned to the natural subset. Otherwise, the other node will be used. The global coefficients β_n and β_s are then multiplied by the coefficient related to

Algorithm 6: Decision Tree - Training

Given: a dataset $d(i)$, a list of classifiers c and a root node

repeat

Get: the classifier c_i that minimize the error ϵ_i among the list of classifiers c that are not already used

Split: the current dataset $d(i)$ into 2 subset d_n having only images classified as Natural and d_n having only images classified as Synthetic by the classifier c_i

Calculate: the accuracy α_n for the left child node and α_s for the right child node using the classifier c_i respectively with d_n and d_s

Remove: c_i from c

if *((d_s or d_n) not empty) and (c not empty)* **then**

1: Assign d_n to the left child and d_s to the right child

2: Recursively repeat the previous steps

end

until *((d_s or d_n) not empty) and (c not empty)* ;

the current node accordingly to the result of the classifier.

On a given node, the image is classified using the local classifier and β_n is multiplied by the probability that it is natural (α_n) and β_s will be multiplied by the probability that it is synthetic ($1-\alpha_n$) This is repeated until a leaf node is reached. The local accuracy of every node contribute to the final classification. When a leaf node is reached, the global coefficient are compared. If $\beta_n > \beta_s$ the image is classified as natural, otherwise it is classified as synthetic.

figure 4.2 shows the first nodes of a sample trained decision tree. For example, an image is taken and all features are extracted. Then the classifier matched to the root node (*classifier 1*) is used and it classifies the image as Synthetic. Now we have:

- $\beta_n = \alpha_n = 0.3$
- $\beta_s = (1 - \alpha_n) = 0.7$

Then the classified matched to the right node (*classifier 2*) is used and it classifies the images as Natural. Now we have:

- $\beta_n = \beta_n * (1 - \alpha_n) = 0.2 * 0.6 = 0.12$

Algorithm 7: Decision Tree - Classification

Given: an image i and a *root* node of a trained decision tree

repeat
  Classify: the image i with the classifier c associated with this node
  if *i is Natural* **then**
    $\beta_n* = \alpha_n$
    $\beta_s* = 1 - \alpha_n$
    Recursively repeat previous steps using the natural child node
  else
    $\beta_s* = \alpha_s$
    $\beta_n* = 1 - \alpha_s$
    Recursively repeat previous steps using the synthetic child node
  end
until *node is not a leaf*

if $\beta_n > \beta_s$ **then**
  i is Natural
else
  i is Synthetic
end

- $\beta_s = \beta_s * \alpha_n = 0.8 * 04 = 0.32$

Then the classified matched to the right node (*classifier 5*) is used and it classifies the images as Synthetic. At this point the algorithm stops because that node is a leaf (during the training phase no images were classified as synthetic by the associated single classifier). The final output is: $\beta_n = 0.12$ and $\beta_s = 0.32$. Since $\beta_s > \beta_n$ the image is classified as Synthetic.

4.3 Neural Network

An Artificial Neural Network [5] is a computational network inspired by the way biological nervous systems process information. It is composed of a number of interconnected processing elements (neurones) that interact in order to solve a specific problem. They, can be used to extract patterns and connections between features that are too complex to be noticed by either humans The main feature of a neural network is the fact that they learn by example. Once the network is feed with data, it will run a number of iterations converging to the best result. During the training process, the connection between

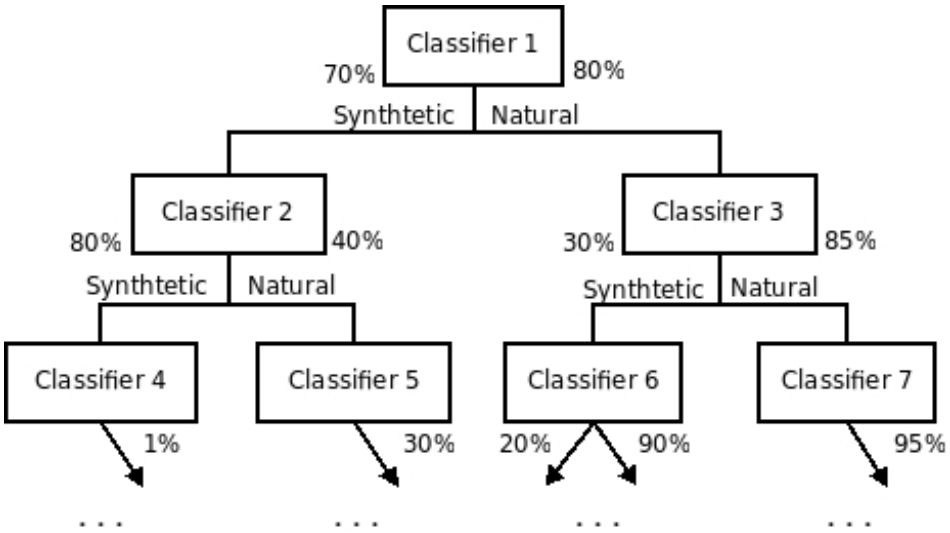


Figure 4.2: Example of Decision Tree Classification.

neurons are tweaked to make the network give the same outputs as seen in the training data. Neural networks can be used to solve classification problems and detect patterns that are too complex to be noticed by either humans or other computer techniques. The trained network can then be used on new data to classify it with an high accuracy.

As shown in figure 4.3, the Neural Network used is composed by 3 layers of neurons. The input layer, as its name suggests, is formed by the input data. The output layer is the final classification. The intermediate layer (hidden layer) is needed to be able to solve problems that are not linearly separable.

4.3.1 Training

Some values have to be customized in order to optimize the performance. This has to be achieved without building an overfitted network that gives precise results for the training data, but incorrect results for all other data.

The following parameters had to be chosen:

- **Number of hidden neurons**. It is not possible to know the optimum size of the hidden layer without actually running some tests. In general, the more hidden neurons are used, the more the network will be able to classify complex problems. But, if a too large hidden layer is used, the network will overfit the training data set. After some tests, i found

out that the best performances are achieved using 6 hidden neurons. Between 1 and 6, the accuracy on both training and testing set increase. After 6, the accuracy on the training set continue to increase but the accuracy on the testing set starts to decrease because of the network being overfitted.

- **Number of iteration**. The problem of overfitting has to be considered also while setting the number of iteration. Though for the testing set the more iterations mean the more accuracy, it is not true for the training set. After a number of iteration, the accuracy on the testing set decrease for the same reason as before: the network starts to overfit. Different values have been tested and 3000 iterations has been chosen as number of iterations.

- **Learning rate**. Changing the learning rate parameter, it is possible to tweak the speed at wich the neural network will converge to the optimum layout. It is involved in the weights update phase after every iteration. With an high value, the weights change rapidly thus, the algorithm needs less iteration in order to converge but, the final value are less accurate if a too high value is used because the weights fluctuate around the optimum value. Since an algorithm that does not take into account the learning rate (RPROP) has been used for the final classifier, the value of this parameter did not affect the final results but has been tweaked while evaluating other learning algorithms.

- **Activation function for**

  - *Hidden layer.*
  - *Output layer.*

  The activation function is the core of a neural network. It takes some input and gives an output between 0 and 1. It is possible to use different kind of functions such as linear, sigmoid or gaussian. Compared to all of those, the sigmoid symmetric gives the best results for both the hidden and the output layer.

- **Training Algorithm** As explained in the documentation page of the FANN library [1], there are 4 implemented algorithm:

  - *Incremental standard backpropagation algorithm*, where the weights are updated after each training pattern. This means that the

weights are updated many times during a single epoch. For this reason some problems, will train very fast with this algorithm, while other more advanced problems will not train very well.

- *Batch standard backpropagation algorithm*, where the weights are updated after calculating the mean square error for the whole training set. This means that the weights are only updated once during a epoch. For this reason some problems, will train slower with this algorithm. But since the mean square error is calculated more correctly than in incremental training, some problems will reach a better solutions with this algorithm.
- *RPROP* A more advanced batch training algorithm which achieves good results for many problems. The RPROP training algorithm [22] is adaptive, and does therefore not use the learning rate. The actual learning algorithm implemented is the iRPROP training algorithm [5] which is a variety of the standard RPROP training algorithm.
- *Quick prop* A more advanced batch training algorithm which achieves good results for many problems. The quickprop [12] training algorithm uses the learning rate parameter along with other more advanced parameters.

RPROP training algorithm has been used since it gives the best performances compared to the other methods tested for this project.

4.3.2 Classification

For the project, a network composed by 12 input neurons, an hidden layer and 1 output has been used. Several number of hidden layer configuration have been tested. The best results have been achieved using 6 hidden neurons. Another parameter to tweak was the number of iterations. After some test, the best accuracy/performance ratio was achieved by running 3000 iterations.

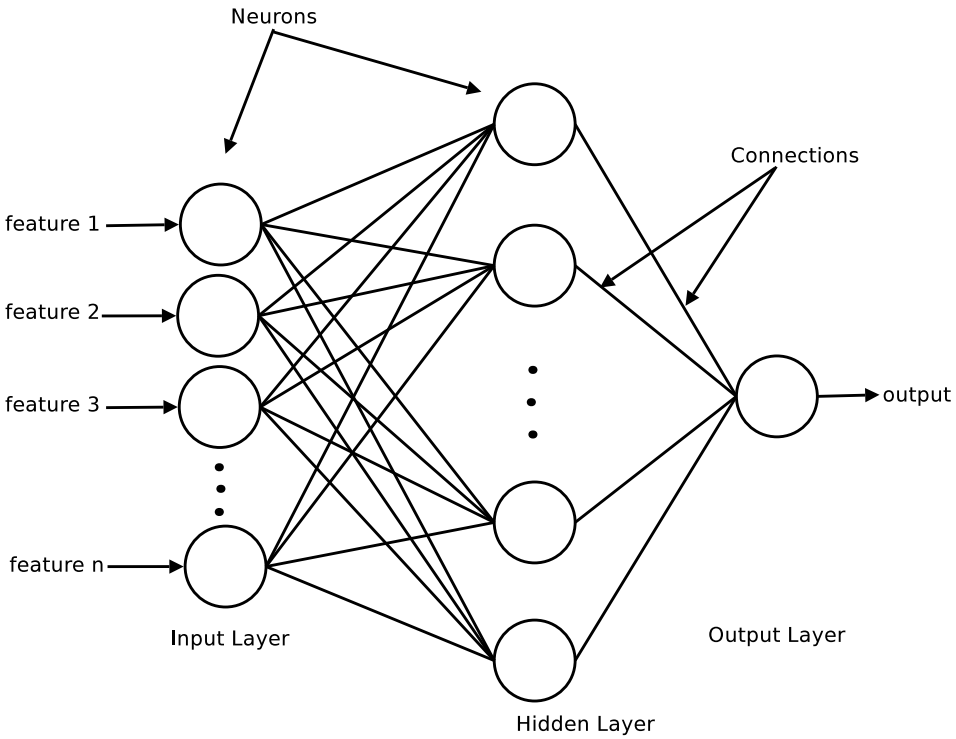


Figure 4.3: Neural Network Diagram. The input neurons are connected to the output neuron trough a network of coefficients.

Chapter 5

Testing

Each single classifier described in the previous chapters has been tested and tweaked changing thresholds and input parameters so that its accuracy results as high as possible. Now the aggregate classifier has to be tested in order to evaluate the global performance of the system on the data set. Different methods of tests are possible. The basic approaches is to use the entire dataset as training and testing to evaluate the classifiers and estimate the error rate but it has two fundamental problems:

1. The final model will probably **overfit the training data**. When a classification method is trained too much on a certain data, it can happen that it will have a poor performance when used on different data. It is important to not build a model which so completely reflects the weirdnesses of the training data that it misses the larger picture.

2. The error rate estimate will be **excessively optimistic**. Because the classifiers are trained too much on the same data, it is common to achieve a very high accuracy on the training set that is actually far from the real one.

Since one of the main concern during the testing phase is to avoid the overfitting, a more accurate approach has to be used. I decided to not use the entire data set and split the training images into disjoint subsets when training a classification method.

5.1 Method

K-fold cross validation [7] has been used to estimate how accurately a classification algorithm will predict data that it was not trained on. The training dataset is randomly partitioned into k subsets. Each iteration, one of the k

subsets is excluded from the first phase and the remaining k-1 subsets are used together as training set. Then, when training is done, the data that was excluded is used to test the performance on new data, different from the learning set. In my experiment, the dataset has been randomly split into 5 partitions. Each classification method has been tested 5 times using $\frac{4}{5}$ of the subsets for training and the remaining $\frac{1}{5}$ for testing. The average error is then used as estimator of the performance. This procedure, has been repeated 10 times and the average error is extracted from single results. the results reported are therefore the average of 10 5-fold cross validation tests.

5.2 Performance

In this section the performance of three different classification method are compared using the same image dataset described in chapter 2. Results are provided for natural and synthetic images. AdaBoost, Decision Tree and Neural Network have been used and tested. These different approaches reported to have different performances when used on the selected dataset.

5.2.1 AdaBoost

The algorithm has been tested finding the best number of iteration. For each iteration the classifier that perform best is chosen and its weight incremented. The algorithm converges to the best accuracy fast: even after only 20 rounds the changes in weights are small. A value of $T = 50$ iteration has been used for the experiment since the final variation of coefficients was less than 10^{-6}.

In Table 5.1 it is possible to see the different weights of single classifiers after the training. The table shows that the Farthest Neighbor metric and the Most Used Color Saturation metric are not considered in the final boosted classifier. According to the adaBoost algorithm, this means that these two classifiers were not adding any useful information to the classification because their decision were similar or identical to the one taken by previous classifiers.

For example both SGLD Variance and Farthest Neighbor metrics consider the spatial correlation of pixels and they probably give very similar classification. For the adaBoost algorithm, SGLD Variance is performing better (even slightly better is enough) thus, there is no need to have a "duplicate" classification,

The global error rate reported for AdaBoost is 11.9 percent (7.7 percent for natural images and 16.1 percent for synthetic images) This method shows a big difference of performance between the two categories. It performs better with natural images than with synthetic images. This is because some features

extracted from synthetic images are more similar to the threshold chosen for the natural class or, using other words, there are more synthetic images that looks like natural than vice versa.

Since some of the features have a very low weight for this classifier, a new test with only the 6 most relevant single classifier has been done. The test was performed using the same method as before but with the following selected features:

- Most used colors pixels ratio
- Saturated pixels ratio
- SGLD variance
- Gray histogram Smoothness
- Most used color(red)
- Saturation average

The results were in line with what was expected: 13.8 percent (9.8 percent for natural images and 17.8 percent for synthetic images) The accuracy is higher than the best single classifier (Gray Histogram Smoothness with 17 percent error rate) but less than using all available classifiers. However the accuracy loss is only 1.9 percent using half classifiers. This can be a positive trade-off if a speedup of the system is required with only a minor loss of accuracy.

5.2.2 Decision Tree

The decision tree algorithm gives almost the same performances of AdaBoost. The global error rate reported is 12.1 percent (10.8 percent for natural images and 13.4 percent for synthetic images) Its performances are more balanced between natural and synthetic images than adaBoost. After the tree is trained, there isn't a specific way to know what single classifier can be excluded from the tree because classifiers are used dynamically, in different order. It was not possible therefore to exclude any single classifier from the algorithm.

5.2.3 Neural Network

A trained neural network consists on a number of coefficients that bind input and output neurons through the neurons contained in the hidden layer. Such a intricate net, makes not possible to have an exact estimation of the correlation of single input with the output. Therefor,e unlike with AdaBoost, the

Classifier	Coefficient
Most Used Colors Pixels Ratio	0.1850
Saturated Pixels Ratio	0.1619
SGLD Variance	0.1499
Gray Histogram Smoothness	0.1286
Most Used Color(red)	0.1118
Saturation Average	0.0982
Different Colors Ratio	0.0910
Most Used Color(blue)	0.0427
Most Used Color(green)	0.0173
Color Correlogram	0.0135
Most Used Color Saturation	0.0000
Farthest Neighbor	0.0000

Table 5.1: AdaBoost Coefficients. Higher values of coefficients are associated to classifiers with high accuracy.

	Natural	Synthetic	Global
AdaBoost	0.077	0.161	0.119
Decision Tree	0.108	0.134	0.121
Neural Network	0.052	0.064	0.058

Table 5.2: Error Rates. The neural network classifier has the best overlall performance.

relevance of each classificator is unknown. It is not easy to discern between a very useful one and a a one that can be discarded.

The global error rate reported is 5.8 percent (5.2 percent for natural images and 6.4 percent for synthetic images). This is the best classification method tested. It gives also balanced error between natural and synthetic classification. The reason of such a success is because of the ability of an artificial neural network to handle problems that are not linearly separable. It not just assign a weight to a single classifier but combine them all together reducing the global error rate.

5.3 False Positive vs True Positive

In an image search engine context it is very important to minimize the number of false positive. Given a category of images, *True Positive* are defined as the images member of that category that are correctly classified and *False Positive* are defined as the images member of the other category that are misclassified as member of the given category.

If an user is searching for a drawing of an house, he will check the option "Synthetic images only". When the search returns an high number of results, often it does not matter how many of them are made available to the user, he will most likely consider only the first few pages. In order to maximize the quality of the service offered by the search engine a better strategy might be to have a lower number of more accurate results.

The 3 pairs of graphs in figure 5.1 show the correlation between number of images that are correctly classified and number of images that are incorrectly reported as Synthetic/Natural.

The dashed line represents the number of false positive and the straight line represents the number of true positive. The vertical lines represent the level of confidence for which the rate of false positive is 5, 2 and 1. Synthetic images results are on the left side and natural images on the right side.

For Synthetic images (left side):

- straight line = $\sum$ (*Synthetic images having confidence* $< x$) (True positive)
- dashed line = $\sum$ (*Natural images having confidence* $< x$) (False positive)

For Natural images (right side):

- straight line = $\sum$ (*Natural images having confidence* $> x$) (True positive)
- dashed line = $\sum$ (*Synthetic images having confidence* $> x$) (False positive)

Table 5.3 shows the number of displayed images when searching for Natural or Synthetic with a desired rate of False Positive (5, 2 and 1 percent). With the first 2 methods the results are good but not impressive: the number of images returned are on average down to about 50 percent with 2 percent of false positive and to about 30 percent with 1 percent. This means that the accuracy of a search with 10.000 matches can be boosted displaying "only" 3.000 items and having 1 false positive every 16 pages of results (assuming 18 items per page).

	Synthetic			Natural		
False Positive	0.05	0.02	0.01	0.05	0.02	0.01
AdaBoost	0.77	0.60	0.25	0.72	0.28	0.14
Decision Tree	0.73	0.53	0.40	0.71	0.57	0.32
Neural Network	0.92	0.85	0.78	0.91	0.81	0.54

Table 5.3: False Positive Comparison. The neural network method reported the highest number of true positive on all 3 cases.

Better performances can be achieved by using the Neural Network approach: with the constrain of 1 percent of false positive this method performed better than earlier approach especially with synthetic images. Up to 78 percent of synthetic images can be displayed having only 1 percent of false positive.

5.4 Thumbnails Test

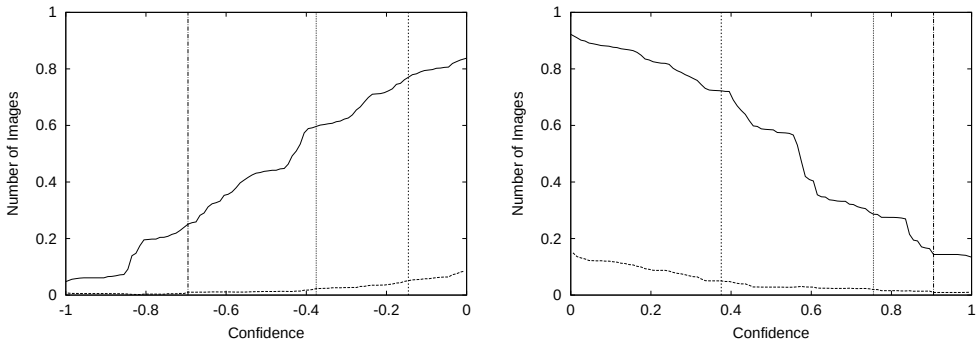
Thumbnails are reduced-size versions of pictures. They are used by image-organizing programs and image search engines often as preview of full-size images.

A new dataset has been created by resizing all the images in the old dataset to 128x128 pixels The same 5-fold cross validation test has been done using the same single classifier. Table 5.4 shows the error rate achieved using AdaBoost, Decision Tree and Neural Network on the resized dataset.

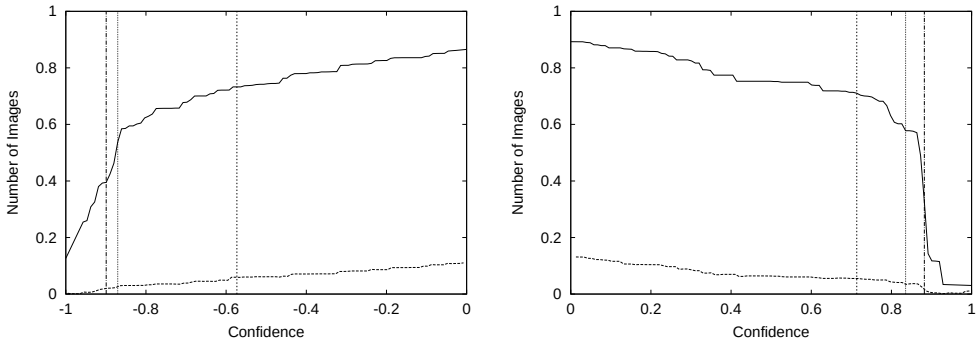
The accuracy is comparable to the performance achieved on the normal dataset.

The performance test has been done on a desktop machine with an AMD Athlon 64 x2 Dual-Core 3800+ processor, 1 Gb RAM under a 64 bit distribution of Ubuntu Linux. This is an average machine, therefore i expect the following reported performances to be a lower bound estimation. The normal dataset has been analyzed and classified in 654 seconds, that is 9.2 images per second. The thumbnails dataset has been analyzed and classified in 133 seconds that is 45.1 images per second: 4.9 times faster than the normal dataset. The increase in speed is very large, especially compared to the increase in the error rate.

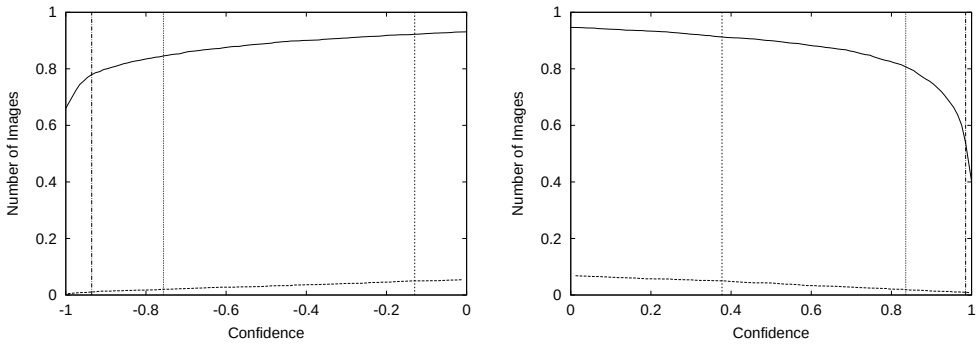
The ratio performance-speed result to be very convenient in this case and it allows large volume of images to be indexed and classified.



(a) AdaBoost



(b) Decision Tree



(c) Neural Network

Figure 5.1: Comparison of False Positive and True Positive. Synthetic and Natural images are represented on the left and right side respectively. The vertical lines represent the confidence for which the rate of false positive is 5, 2 and 1 percent.

	Natural	Synthetic	Global
AdaBoost	0.084	0.192	0.138
Decision Tree	0.158	0.146	0.152
Neural Network	0.062	0.087	0.075

Table 5.4: Error Rates on Thumbnails Dataset. The retes are comparable to the test performed on the full-size dataset.

Chapter 6

Conclusion and Further Work

In the course of this thesis three methods for the classification of synthetic and natural images have been developed and tested. Some base classifiers have been implemented and tweaked for the maximum individual accuracy. Single features are combined together using different algorithms with different performances.

A common dataset of manually labeled natural/synthetic images has been used as test set. This set has been lately resized to perform the same tests on resized images.

In the best case, the performances achieved are on par or slightly better than previous works.

6.1 Conclusions

In figure 6.1 are summarized the performances of the 3 tested methods using the dataset described in Chapter 2. AdaBoost and Decision Tree have a similar global accuracy but adaBoost performs better on natural images than on synthetic images. These 2 methods are outperformed when using a Neural Network. It showed the best accuracy on the classification of both natural and synthetic images. The overall error rate of 5.8 percent is half the error of the firs two method. In figure 6.2 is possible to compare the results of the test performed on the thumbnails dataset. Even if the error rates are higher, the Neural Network classifier has better performances.

figure 6.3 shows the behavior of different classifiers with a false positive constraint. Neural network performs again better than other 2 methods. The very good results showed by the Neural Network with respect to the problem of image classification is due to the fact that this method is able to handle very complex relations between the single features extracted from the raw image. It

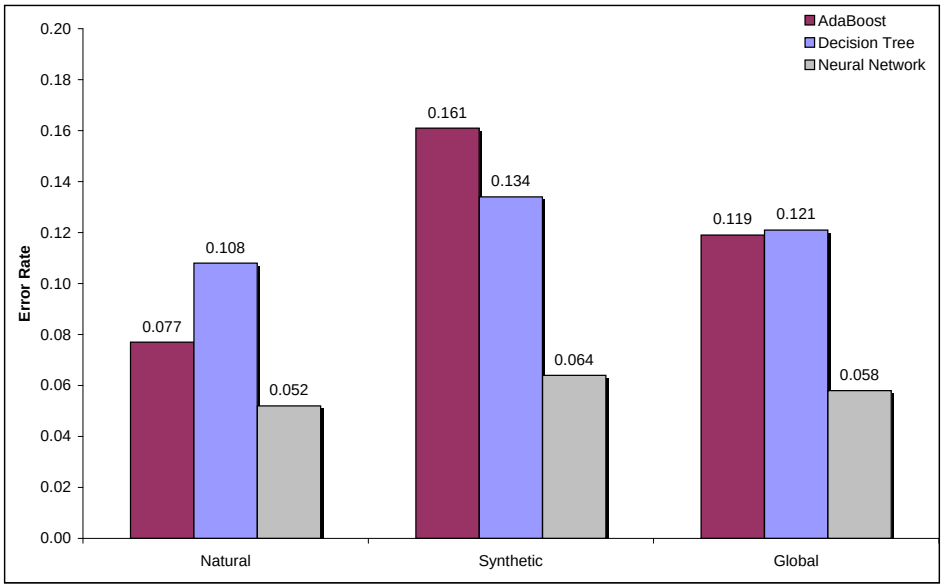


Figure 6.1: AdaBoost, Decision Tree and Neural Network classifiers performance comparison on full-size dataset.

is able to weigh and match the inputs in a way that is very efficient. AdaBoost and the Decision Tree algorithm are able to discern between classes of images but their performances are limited by the high complexity of the problem: it is not just a matter of finding a threshold on the number of different colors in an image. It is a complicate non linear problem and that is the reason of the success of Neural Network.

6.2 Further Work

6.2.1 Ambiguous Images

A method for handling ambiguous images can be developed by splitting them into n subimages and performing the classification on each part. Then it is possible to analyze the single areas and determine if the image contains some mixed Synthetic-Natural element. This would be useful when the initial classification has a low level of confidence in order to enhance the accuracy.

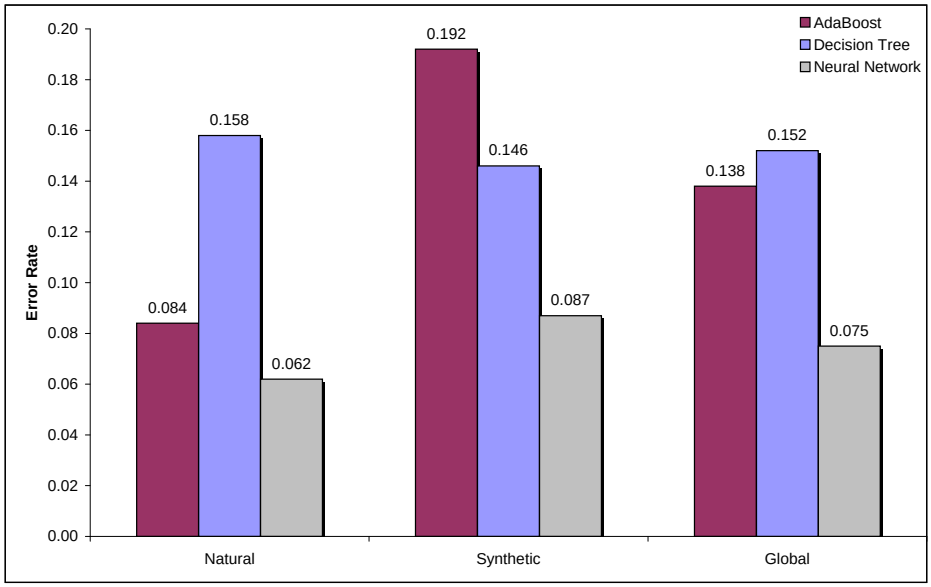
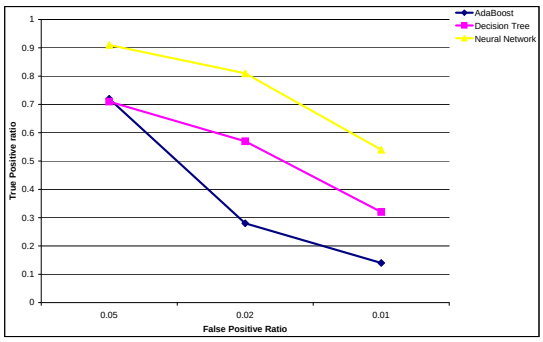
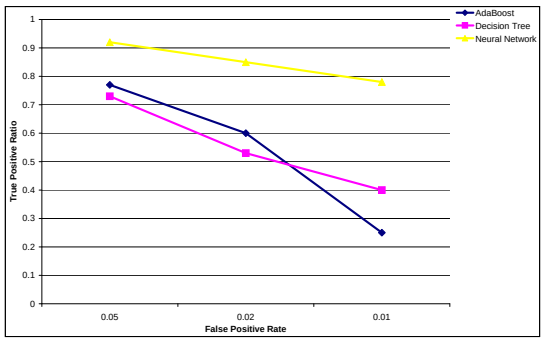


Figure 6.2: AdaBoost, Decision Tree and Neural Network classifiers performance comparison on thumbnails dataset.



(a) Natural



(b) Synthetic

Figure 6.3: False Positive vs True Positive. Neural Network performs much better than other methods.

6.2.2 Black and White Images

Black and white images have been excluded from the dataset because most of the classifiers rely on full colors features and that would have cause a large number of misclassified images. However it is possible to find some classifier able to perform well on black and white images.

6.2.3 Video Classification

The image classification can be extended to video classification. Given a video it is sufficient to extract some sample frame and gives them to the classifier. In this way it would be possible to differentiate between movies, cartoons or even flash animation/presentation. Even though i think that the majority of the classifiers will give good accuracy without any changes, it might be possible to tweak them for a better dedicate movie classification.

6.2.4 Improved Thumbnails Classifier

The tests with the resized dataset showed that it is possible to greatly speedup the classification using thumbnails. Those tests however have been done using the single classifiers conceived for full size images. Some features are not really much significant when dealing with a reduced size images such as 128x128 pixels. For example it is hard to have an estimation of the sharpness of the edges with so few pixels. Tweaking the old classifiers and testing some new ones dedicated to small-size images can increase the overall accuracy of the system and increase the performance-speed ratio even more.

6.2.5 Cascade Classifiers

It is possible to develop more binary classifiers and building a tree of classifier in order to classify images in different categories. For example, after a first classification between Natural and Synthetic it is possible to further classify the image into subcategories such as city/landscape and then as seaside/mountain or indoor/outdoor for natural images and as drawing/chart/map/cartoon for synthetic images.

Bibliography

[1] Fast artificial neural network library. http://leenissen.dk/fann/.

[2] Image magick. http://www.imagemagick.com.

[3] Intro to png features. http://www.libpng.org/pub/png/pngintro.html.

[4] Picsearch. http://www.picsearch.com.

[5] D. Anderson and G. McNeill. Artificial Neural Networks Technology. *A DACS (Data & Analysis Center for Software) State-of-the-Art Report, Contract*.

[6] V. Athitsos, M.J. Swain, and C. Frankel. Distinguishing photographs and graphics on the world wide web. *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 10–17, 1997.

[7] A. Blum, A. Kalai, and J. Langford. Beating the Hold-Out: Bounds for K-fold and Progressive Cross-Validation.

[8] B. Bradshaw. Semantic based image retrieval: a probabilistic approach. *Proceedings of the eighth ACM international conference on Multimedia*, pages 167–176, 2000.

[9] L. Breiman. Bias, variance, and arcing classifiers. *Unpublished manuscript. Available from ftp://ftp.stat.berkeley.edu/pub/users/breiman/arcall.ps.Z*, 1996.

[10] L. Breiman. Arcing the edge. Technical report, Technical Report 486, Statistics Department, University of California at Berkeley, 1997, 1997.

[11] H. Drucker and C. Cortes. Boosting decision trees. *NIPS*, 8:479–485, 1996.

[12] S.E. Fahlman. Faster-Learning Variations on Back-Propagation: An Empirical Study. *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51, 1988.

[13] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[14] Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[15] MM Gorkani and RW Picard. Texture orientation for sorting photos 'at a glance'. *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, 1, 1994.

[16] A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 692:699, 1998.

[17] J. Huang, S.R. Kumar, M. Mitra, and W.J. Zhu. Image indexing using color correlograms, June 12 2001. US Patent 6,246,790.

[18] R. Lienhart and A. Hartmann. Classifying images on the web automatically. *Journal of Electronic Imaging*, 11:445, 2002.

[19] T.T. Ng, S.F. Chang, J. Hsu, L. Xie, and M.P. Tsui. Physics-motivated features for distinguishing photographic images and computer graphics. *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 239–248, 2005.

[20] Salil Prabhakar, Hui Cheng, Zhigang Fan, John C. Handley, and Ying wei Lin. Picture/graphics classification system and method. United States Patent 6,947,597 B2, January 2006.

[21] J.R. Quinlan. Bagging, boosting, and C4. 5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 725:730, 1996.

[22] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: theRPROP algorithm. *Neural Networks, 1993., IEEE International Conference on*, pages 586–591, 1993.

[23] N.C. Rowe and B. Frew. Automatic caption localization for photographs on World Wide Web pages. *Information Processing and Management*, 34 (1):95–107, 1998.

[24] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5 (2):197–227, 1990.

[25] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[26] R. Schettini, C. Brambilla, G. Ciocca, A. Valsasna, and M. De Ponti. A hierarchical classification strategy for digital documents. *Pattern Recognition*, 35(8):1759–1769, 2002.

[27] J.R. Smith and S.F. Chang. Searching for Images and Videos on the World-Wide Web. *IEEE MultiMedia*, 1997.

[28] M. Szummer and R.W. Picard. Indoor-Outdoor Image Classification.

[29] A. Vailaya. *Semantic classification in image databases*. PhD thesis, Michigan State University. Dept. of Computer Science & Engineering, 2000.

[30] A. Vailaya, M. Figueiredo, A. Jain, and H.J. Zhang. Bayesian framework for hierarchical semantic classification of vacation images. *IEEE Trans. on Image Processing*, 10(1):117–130, 2001.

[31] A. Vailaya, A. Jain, and H.J. Zhang. On Image Classification: City vs. Landscape.

[32] E.C. Yiu. Image classification using color cues and texture orientation. 1996.

[33] H.J. Zhang. Scheme for visual feature-based image indexing. *Proceedings of SPIE*, 2420:36, 1995.

TRITA-CSC-E 2008:112
ISRN-KTH/CSC/E--08/112--SE
ISSN-1653-5715

www.kth.se