An Empirical Study of Machine Learning Techniques Applied to image Watermarking.

José Ramón Bronet, David Masip.

Abstract.

A digital watermark embeds an imperceptible signal into data such as audio, video and images, for a variety of purposes, including captioning and copyright control. The different watermarking mechanisms are sensible to different types of attacks such as rotation, escalation, Gaussian noise addition or compression. This empirical study shows if applying machine learning techniques can improve the detection rate of watermarks after being attacked.

In this study we use 4 different watermarking methods (discrete wavelet transform, discrete cosine transform, histogram shifting and least significant bit) with the gentleBoost machine learning algorithm.

We use 4 different attacks (rotation, escalation, JPEG compression and Gaussian noise addition) with different attack intensities to show the success rates and the false positives rates of the trained systems, comparing, then, with those 4 watermarking methods not trained.

1. Introduction

Digital watermarks are pieces of information added to digital data (audio, video, or still images) that can be detected or extracted later to make an assertion about the data. This information can be textual data about the author, its copyright, etc...; or it can be an image itself. The information to be hidden is embedded by manipulating the contents of the digital data, allowing someone to identify the original owner, or in the case of illicit duplication of purchased material, the buyer involved. These digital watermarks remain intact under transmission / transformation, allowing us to protect our ownership rights in digital form.

With the current increase of audiovisual media fraud, watermarking is becoming a relevant technique for protecting exploit and author rights. A watermark is a form, image or text that is impressed onto paper, which provides evidence of its authenticity. Digital watermarking is an extension of this concept in the digital world. In recent years the phenomenal growth of the Internet has highlighted the need for mechanisms to protect ownership of digital media. Exactly identical copies of digital information, be it images, text or audio, can be produced and distributed easily. In such a scenario, who is the artist and who the plagiarist? It’s impossible to tell—or was, until now. Digital watermarking is a technique that provides a solution to the long standing problems faced with copyrighting digital data.

Digital watermarks are pieces of information added to digital data (audio, video, or still images) that can be detected or extracted later to make an assertion about the data. This information can be textual data about the author, its copyright, etc...; or it can be an image itself. The information to be hidden is embedded by manipulating the contents of the digital data, allowing someone to identify the original owner, or in the case of illicit duplication of purchased material, the buyer involved. These digital watermarks, in theory, remain intact under transmission / transformation, allowing us to protect our ownership rights in digital form.

But, as usually, the reality is very different than the theory. Different simple attacks weakens the effectiveness of watermark recovering mechanism, those simple attacks could be rotation, escalation or simple compression; which are methods commonly and currently used by regular computers users for ripping movies, songs and pictures.

A 'raw' recovering of a digital watermark presents a low flexible and tolerant way to work, with bad recovering capabilities to small errors. We propose to use machine learning mechanisms that could anticipate the most common attacks, giving a more flexible and fault tolerant decoding.

2. Previous works

A good starting point is the Ingemar J. Cox, Matt L. Miller [3] historical study of the watermarking principles and practices, that piece of information gives a good global idea of the concepts used here.

Some researchers have put sizeable effort to develop new decoder structures for increasing decoding performance. For example, Ching-Yung et al. [4] [5]. Those works were all aligned on developing better algorithms and models capable of getting a better results on the watermark detection probability.

Some others like Asifullah Khan et al. [1] used machine learning techniques. Along the study we see how a neuronal network were used to create an expert model capable of getting better results. However, boosting techniques were never tried.

There is a very sophisticated paper [4] that shows the damage that is done by some different attacks like rotation or escalation. This article leaves clear the importance of researching different mechanisms to be applied in digital watermarking.

3. A machine learning approach

The idea is to use different encoding/decoding mechanisms, which allows to encode a bit stream as a binary mark. Then the decoding mechanisms give back another bit streams with the recovered marks. The approach is to attack some images, for all the different encoding/decoding methods with different attacks and intensities, to then extract the attacked marks and provide those ones to the learning system [6], in order to learn from them.

Once trained, we attack our marked images with different attacks intensities to then know how the system classifies them.

We use random generated marks and keys, and in each case the key act as a different element, its role will be described on each method later on.

The marks are random streams of bits, in fact we used vectors of 11 bits and 11 key positions, e.g.:

mark = [0 1 1 1 1 0 1 1 1]

Keys are generated in the same way, as random streams of numbers; but in this case decimal numbers. The meaning of the keys is dependent on the watermarking technique, so as with the marks, we will explain its meaning later on.

key = [12 34 35 41 49 50 51 5032 5037]

Note: To simplify the LSB algorithm the random numbers generated for the keys are sorted as in the example above.

3.1 Digital watermarking used methods

This study uses four different digital watermarking encoding/decoding methods, which are:

1. **Less Significant Bit.** The most widely used technique to hide data is the usage of the LSB [1]. Although there are several disadvantages to this approach, the relative easiness to implement it, makes it a popular method. To hide a secret message inside a image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. When using a 24-bit colour image, a bit of each of the red, green and blue colour components can be used, so a total of 3 bits can be stored in each pixel. For example, the following grid can be considered as 3 pixels of a 24-bit colour image, using 9 bytes of memory:

   (00100111 11101001 11001000)
   (00100111 11001000 11101001)
   (11001000 00100111 11101001)

When the character A, whose binary value equals 01000001, is inserted, the following grid results:

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001001 00100110 11101001)

In this case, only three bits need to be changed to insert the character successfully. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximal cover size. The resulting changes that are made to the least significant bits are too small to be recognized by the human eye, so the message is effectively hidden. While using a 24 bits image gives a relatively large amount of space to hide messages, it is also possible to use a 8 bits image as a cover source. Because of the smaller space and different properties, 8 bits images require a more careful approach. Where 24 bits images use three bytes to represent a pixel, an 8 bits image uses only one. Changing the LSB of that byte will result in a visible change of colour, as another colour in the available palette will be displayed. Therefore, the cover image needs to be selected more carefully and preferably be in grayscale, as the human eye will not detect the difference between different gray values as easy as with different colours.

We treat coloured images as 8 bits greyscaled ones.

Given a two dimensional image Im2D of *Height x Width* bytes, we convert into a 1 dimensional vector of bytes Im1D, then, within the positions indicated by the random key, the less significant bit its changed for the bit of our mark. e.g.

$$\text{Im2D} = \begin{bmatrix} 23 & 255 \\ 128 & 0 \end{bmatrix}$$

Im1D = [23 255 128 0]

Key in the LSB function: In this method the key is used to know where are the bytes that we are going to change its less significant bit value with the value of our mark. e.g.

mark = [1 0]
key = [12 45]

After reshaping the image into one dimension (vector V), we set the less significant bit of the positions 12 and 45 of V with the values 1 and 0 respectively.

For more information see P.D.Khandait and S.P.Khandait [2].

2. **Discrete Wavelet Transform.** Transforms discrete signal from time domain into time-frequency domain. The transformation product is a set of coefficients organized in the way that enables not only spectrum analyses of the signal, but also spectral behaviour of the signal in time. This is achieved by decomposing signal, breaking it

into two components, each caring information about source signal. Filters from the filter bank used for decomposition come in pairs: low pass and high pass.

Key in the DWT function: In this particular method we use the key to set Matlab's random key generator, to then create the PN sequences in both encoding and decoding.

For more information review section 6 Conference. D. Kundur and D. Hatzinakos [8].

3. **Discrete Cosine Transform.** The discrete cosine transform (DCT) is a real transform that transforms a sequence of real data points into its real spectrum and therefore avoids the problem of redundancy.

   The most common variant of the discrete cosine transform is the type-II DCT, which is often called simply "the DCT" and is the one that we have used in our study.

   Key in the DCT2 function: In this particular method we use the key to set Matlab's random key generator, to then create the PN sequences in both encoding and decoding.

   A nice study about the discrete cosine transform applied to digital watermarking is found in section 6 Matt L. Miller et al. [5].

4. **Histogram Shifting.** Histogram shifting scheme [7] utilizes the redundancy of the host image statistical information to hide secret data. For a given host image, the brief operating process is described as follows:

   Generate the histogram of the host image and judge whether a zero point whose frequency in the histogram is zero exists. If no zero point exists, known as non-zero situation, select a lowest point whose frequency is lowest and use a bitmap to record the locations. After that, set the lowest point value to zero for creating a zero point in the histogram, which is known as zero situation. For a zero situation, choose a peak point whose frequency is highest and a zero point, then shift the histogram bits between the peak and zero points one bit width towards the zero point direction. Finally, scan the host image and hide 1 secret bit of data when the pixel with value of peak point is met. If secret data is ‘0’, the peak point value keeps unchanged, otherwise, it changes to the adjacent zero point value. After hiding secret data, the peak point and zero point (or lowest point) values are sent to the receiver who then performs the reverse operation to extract the secret data and restore the original host image without any distortion.

   Key in the Histogram Shifting function: The key is not used in this method.

   See the article [10] for more information about histogram shifting techniques applied to digital watermarking.

3.2 Binary Watermarking

We focus on the detection of the watermark, so that, our only interest is to know if an image is marked or not.

3.3 Gentleboost.

A good comprehension of gentleboost and previous boosting methods was given by Mark Culp et al. [9]. The implementation used in this and our study, is described below:

Gentle AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weights w_i.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$
   (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

Algorithm: *A modified version of the Real AdaBoost algorithm, using Newton stepping rather than exact optimization at each step*

4. Experiments

In our study, we have used 10 different images (lena, f-16-1, baboon, peppers, earth, moon, pentagon, boats, starship and f-16-2) and 10 different random marks with random keys as well. The idea is to mark each image with the different marks in sequence, one image marked with one mark per experiment, to have a set of 100 items for each of the digital marking algorithms selected (having a total of 400 marked images). Each of the items is attacked with 4 different mechanisms and different intensities for each mechanism. Those 4 attack mechanisms are:

- Rotation
- Escalation
- Gaussian noise addition
- JPEG compression

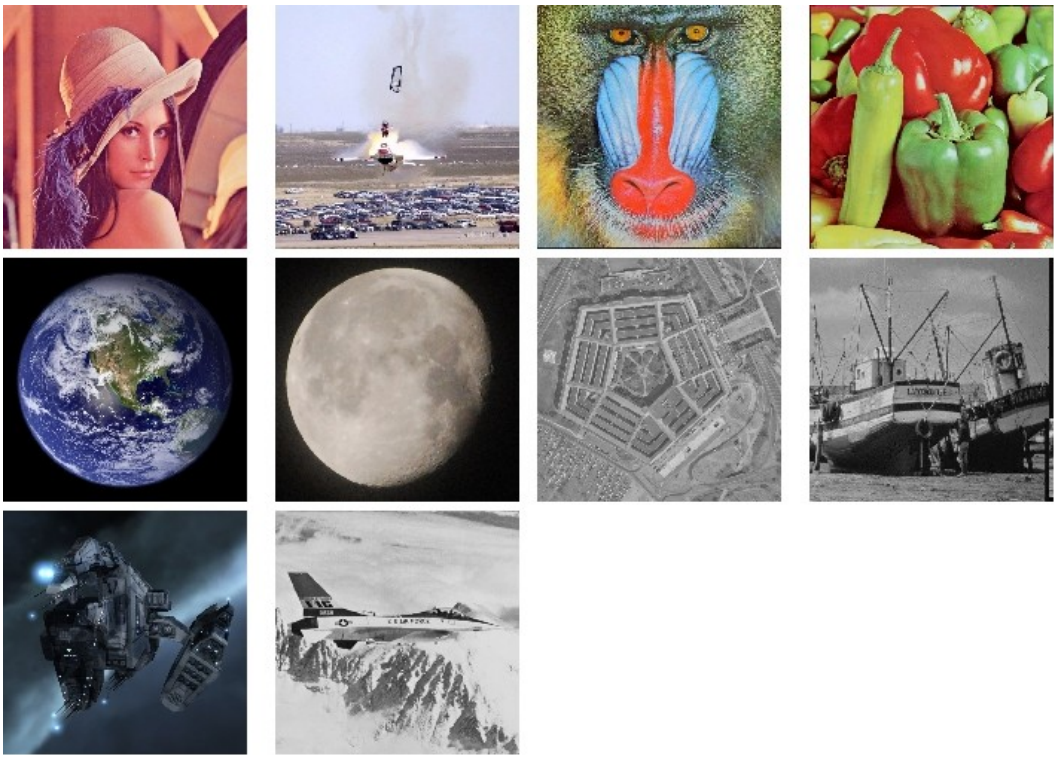
The 4 different digital watermarking algorithms selected for the study are:

- Less Significant Bit
- Discrete Cosine Transform
- Discrete Wavelet Transform
- Histogram Shifting

The trained systems are implemented with the gentleboost algorithm, using a set of 200 rounds to train each of the different watermarking methods. So the idea is to get 4 different trained classifies, one for LSB, another one for DCT2, another one for DWT and finally one for HS.

The mechanism to train the boost is to mark and attack the resultant marked images to then extract the attacked mark. Now with a set of attacked marks and the original marks we instruct the boosts to learn from them.

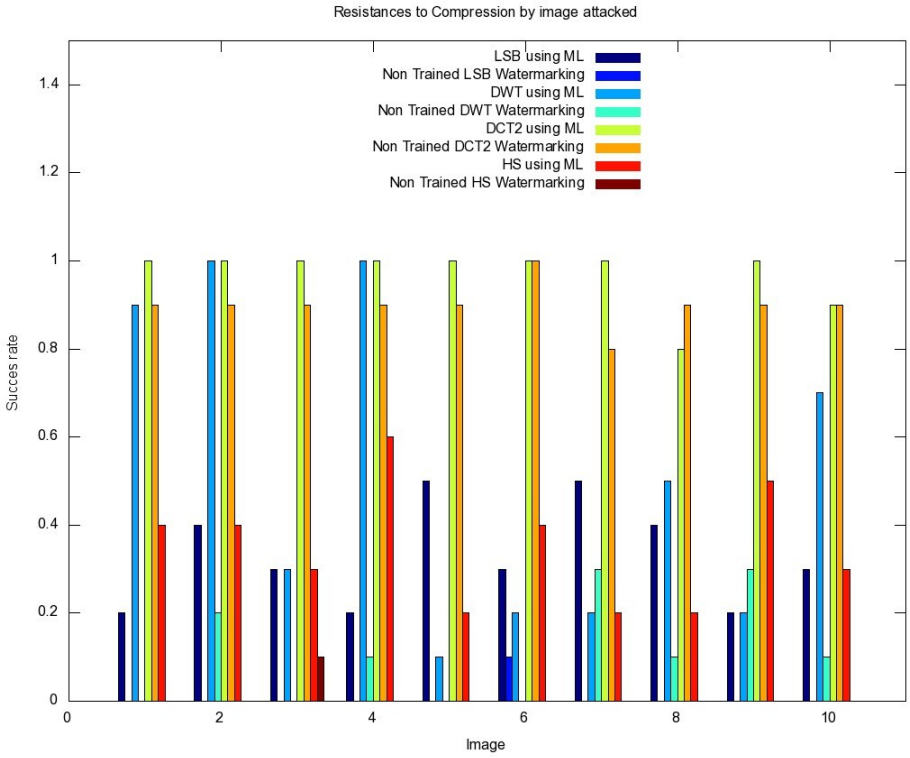
4.1 The images. We have used 10 different images, in the following section we will see them in the results per image along the x axis (represented with numbers from 1 to 10); those images (in order from left to right and up down) are the following:

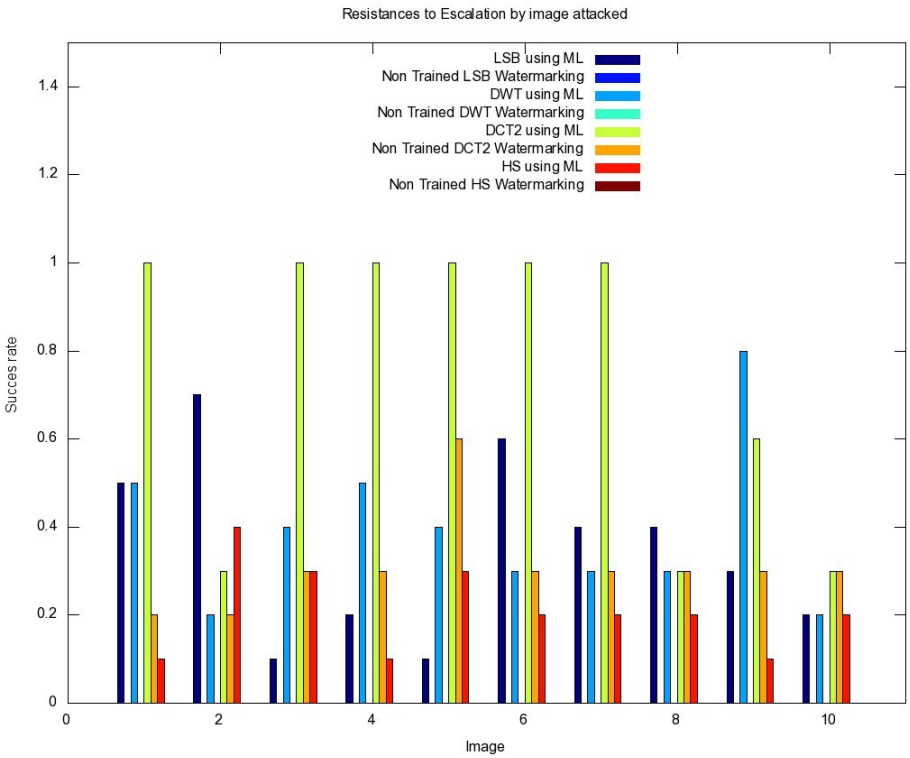


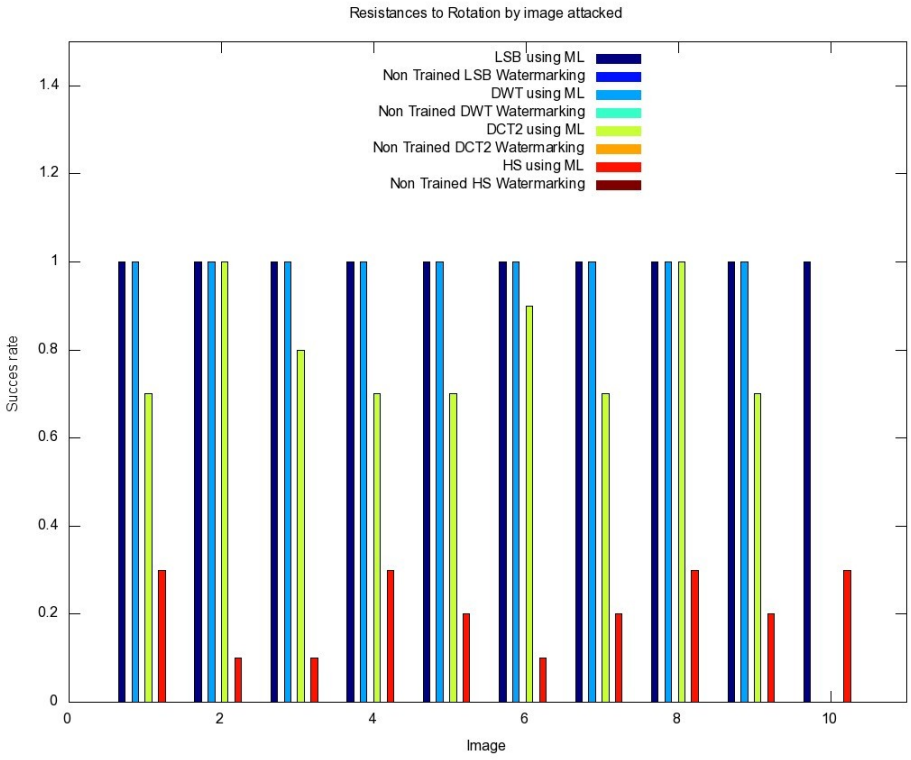
Some of them are in colour and some of them are greyscaled, however this does not make any difference to us, as we have pointed before, we treat them all as greyscaled images.

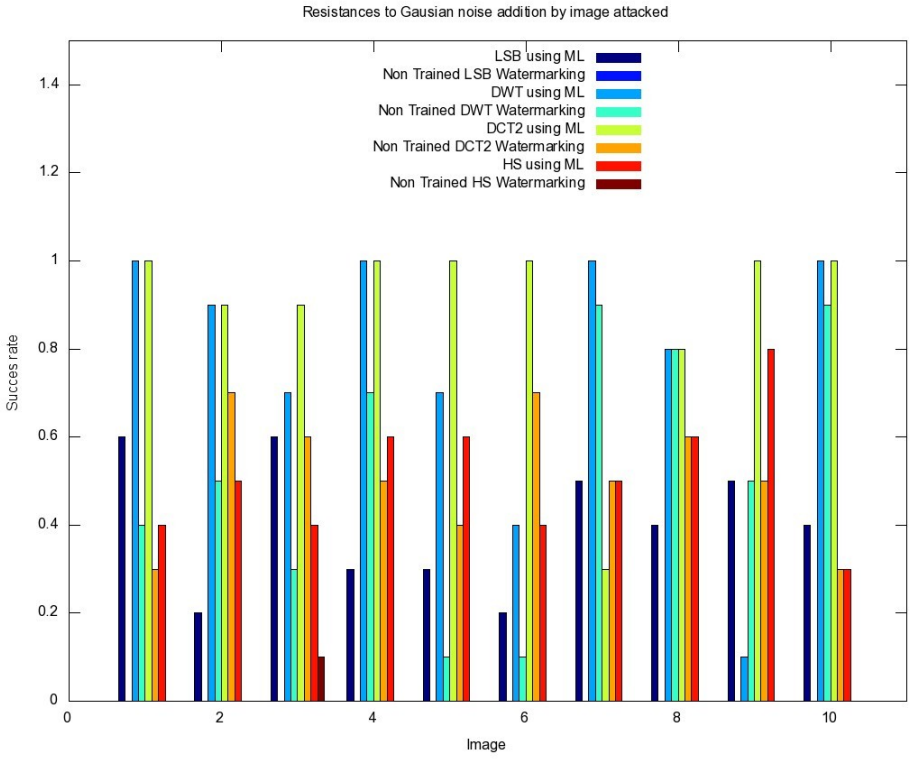
4.2 Results per image. The following graphs show the mark detection success ratio (y axis) per one of the 10 marked images (x axis). Each of the graphs corresponds to a different watermark algorithm.

Take into account that each image is attacked with different intensities.


Resistances to Compression by image attacked
LSB using ML
Non Trained LSB Watermarking
DWT using ML
Non Trained DWT Watermarking
DCT2 using ML
Non Trained DCT2 Watermarking
HS using ML
Non Trained HS Watermarking
Succes rate
Image
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10


Resistances to Escalation by image attacked
LSB using ML
Non Trained LSB Watermarking
DWT using ML
Non Trained DWT Watermarking
DCT2 using ML
Non Trained DCT2 Watermarking
HS using ML
Non Trained HS Watermarking
Succes rate
Image
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10


Resistances to Rotation by image attacked
LSB using ML
Non Trained LSB Watermarking
DWT using ML
Non Trained DWT Watermarking
DCT2 using ML
Non Trained DCT2 Watermarking
HS using ML
Non Trained HS Watermarking
Succes rate
Image
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10


Resistances to Gausian noise addition by image attacked
LSB using ML
Non Trained LSB Watermarking
DWT using ML
Non Trained DWT Watermarking
DCT2 using ML
Non Trained DCT2 Watermarking
HS using ML
Non Trained HS Watermarking
Succes rate
Image
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10

The different intensity ranges used per experiment are:

- Compression: Intensity from 44 to 80
- Escalation: Intensity from 0.16 to 1.6
- Gaussian noise addition: Sigma from 4 to 40
- Rotation: Right rotation in degrees from 29 to 290.

4.2.1 General graphs comments:

There is a high resistance to compression by DWT and DCT2 using ML; we have detected a high non trained DCT2 resistance as well. The rest of the mechanisms shows a low (<50%) resistance to compression.

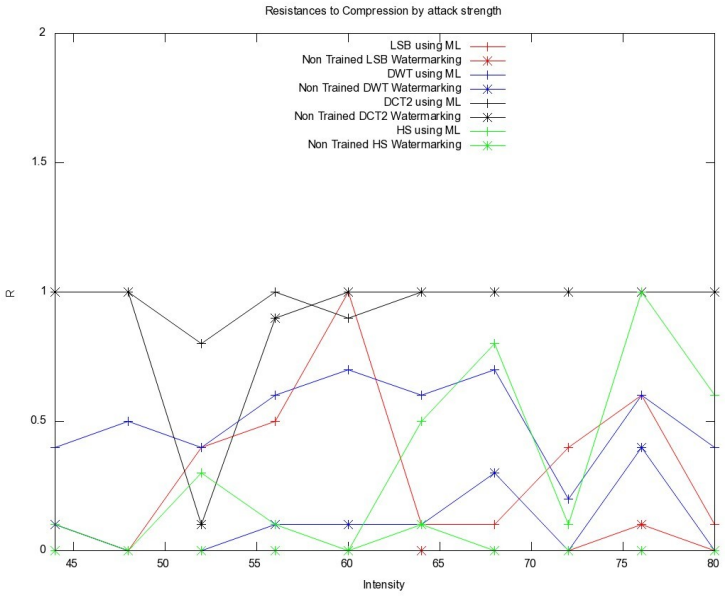
The escalation shows that the strongest method is the trained DCT2, while the rest of them appears not much efficient.

Rotation resistances quite high for LSB and DWT trained systems, DCT2 shows a capable to detect a mark with a high ratio as well. However and due to the nature of the LSB and rotation, we can say that the trained LSB results are not valid; so that we have to rely (in this particular case) in the others methods, so trained DCT2 and DWT are the leaders here.

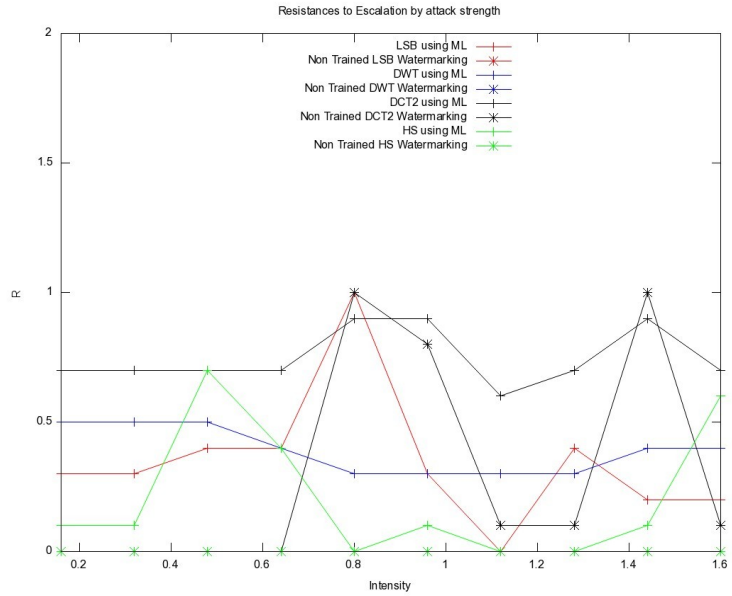
Gaussian noise addition resistance is lead by both trained DWT and DCT2. Non trained DT2 takes the next position while the rest does not perform bad.

4.3 Results per attack intensities. In this section we want to show the different success mark recovering ratio, for trained and non trained watermarking algorithms, depending on the intensity of the attacks. So the following graphs show the mark detection success ratio (y axis) by the different intensity of the attacks (y axis).

The intensity value of the attacks is explained below each figure.

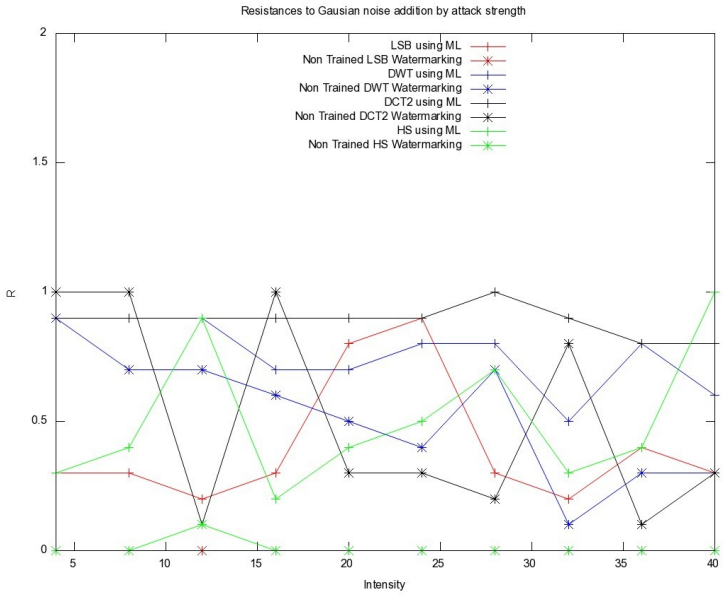


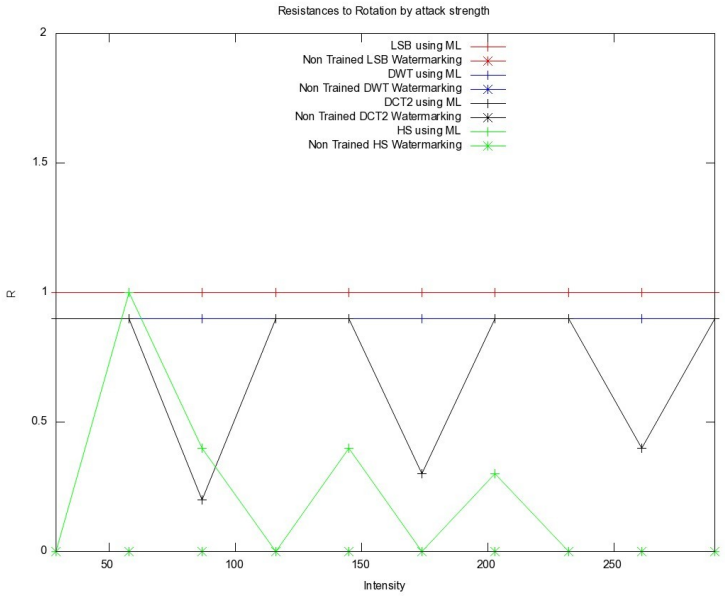
The intensity above reflects the compression rate, the higher the number the compressed is the image.



The intensity above reflects the escalation factor, if equals to 1 there is no escalation, below 1 the image is equally resized to less size while if greater than 1 the image is equally amplified.

The intensity above reflects the Gaussian noise addition sigma Gaussian factor.





The intensity above reflects the image rotation to the right in degrees. e.g. 45 means image rotated 45 degrees to the right.

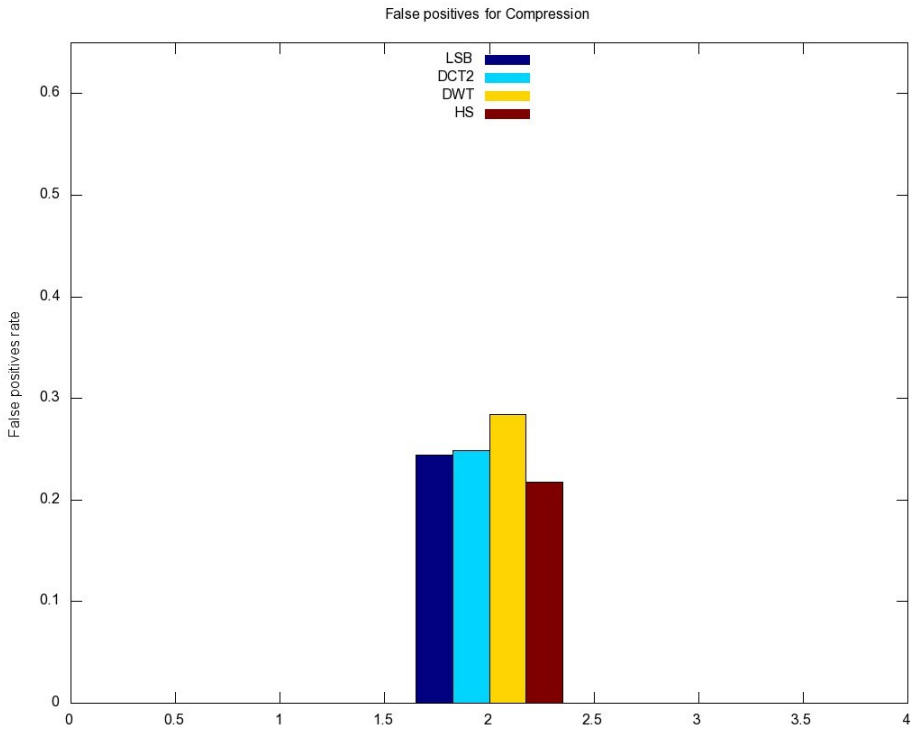
4.3.1 General graphs comments:

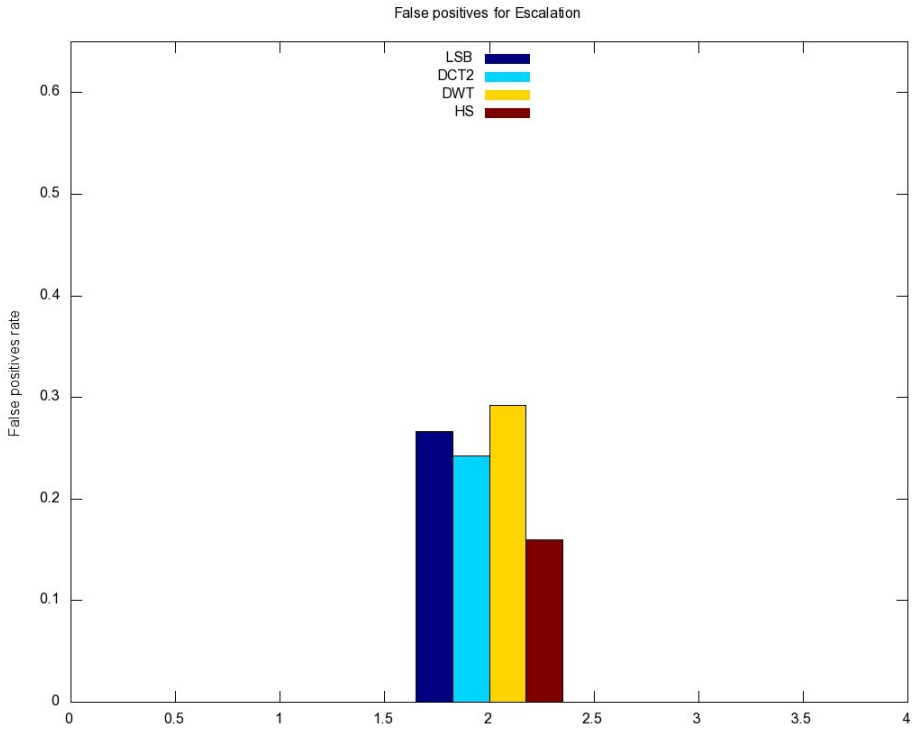
As in the section above, we can see how well performs the trained systems in comparison with non trained ones, the results here are aligned with the results of the previous section, trained DCT2 and DWT2 shows better resistances as the intensity of the attack increases. While the rest of them perform much worst with a higher attack intensity.

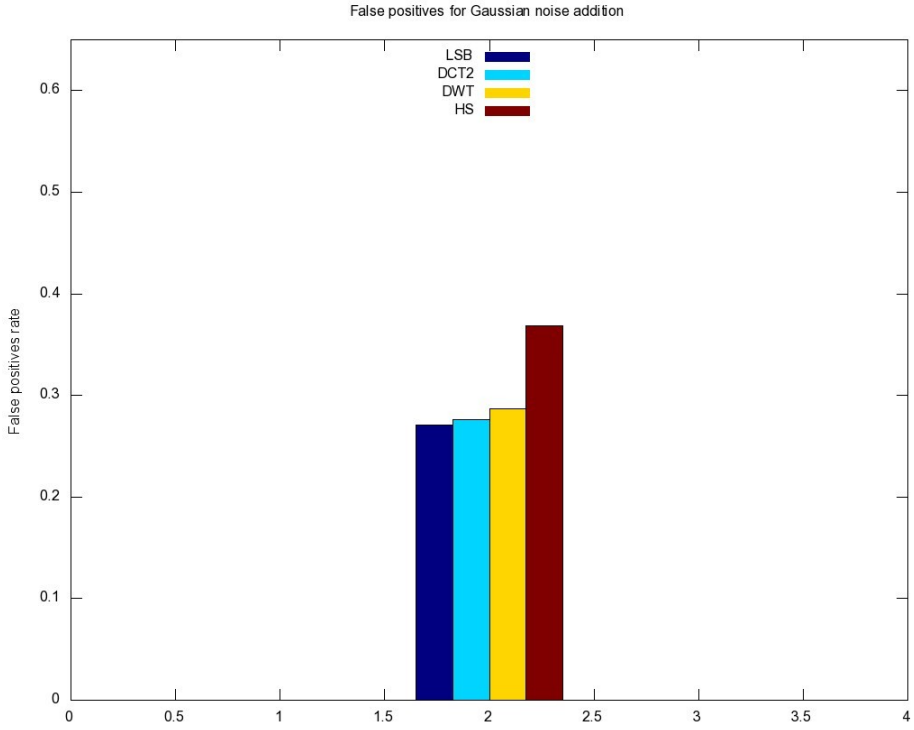
The previous rotation LSB comment applies in the same way here: Due to the nature of the LSB and rotation, we can say that the trained LSB results are not valid.

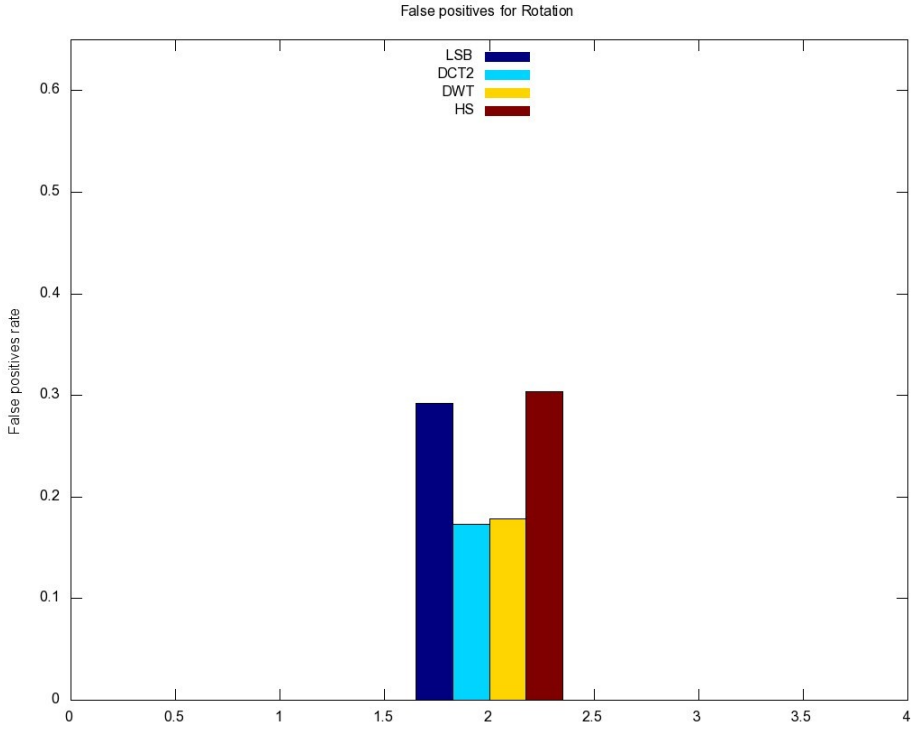
4.4 False positives rates. In this section we cover the false positives rates for each of the machine learned (gentleboost) mechanisms. For each graph we represent the ratio of the false positives for LSB, DCT2,DWT and HS trained methods. The value is obtained by generating random marks, excluding from those random marks the ones that we have used to train positively and negatively the classifiers, so the number of elements classified as “1” by the gentleboost classifier are added and then divided by the number of random marks generated.

The following figures shows the false positives ratio for each of the attack types within the y axis; take a look into the values, that go around 20%-30%.




False positives for Escalation
LSB
DCT2
DWT
HS
False positives rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0
0.5
1
1.5
2
2.5
3
3.5
4


False positives for Gaussian noise addition
LSB
DCT2
DWT
HS
False positives rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0
0.5
1
1.5
2
2.5
3
3.5
4


False positives for Rotation
LSB
DCT2
DWT
HS
False positives rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0
0.5
1
1.5
2
2.5
3
3.5
4

5. Conclusions

Escalation, rotation, Gaussian noise addition and compression affect heavily in the capacity of the algorithms to recover the marks, while the capability of recovering those marks for the trained systems is incredible high, in some cases we are comparing less than 5% success ratio for the non trained systems with 100% of success for the trained ones.

As expected we have a mark recovering ratio highly degradated while the intensity of the attack increases for the non trained systems, this behaviour is occurring for the trained systems as well, but with less impact.

One of the most interesting things coming from the results is the reliability of the trained systems, for mostly all the experiments, and for all the 4 different watermarking methods used, the trained systems did get equal or better detection ratio than its non trained homologous (e.g. LSB trained is equal or better than the LSB non trained).

Unfortunately the false positives rates are high for all of the trained methods, an average of 20%-30% is common for all of the methods and for all of the attacks, which makes this the point to focus on in order to enhance machine learning by boosting applied to digital watermarking.

The high percentage of the false positives explains the high success rate for detecting marks, a better trained classifier could reduce significantly the false positives rates while decreasing the success mark detection rate as well. However, paying that price, machine learning watermarking by boosting enhances significantly non trained methods.

Another point is that the stronger the method is (e.g. DCT2 in our case) the best results for the trained system in terms of false positives and mark recovering rates. So by using a strong mechanism, joint with boosting techniques, will give a high reliable watermarking technique.

We have detected that the LSB technique is not a valid approach to face a rotation attack. After a rotation, the most probable is that the key positions to look for mark were moved to somewhere else, and because we are trying to learn from the mark and not from the marked image, the LSB method will not be able to be trained to detect the mark with this type of attack.

6. Future works

There are things that, even they seem logical, we can not probe it, for example the idea of training better the classifiers to get a better performance in terms of false positives, at the cost of loosing some good mark recovery ratio.

Another improvement will be to transform the coloured images into the YcbCr space, instead of the RGB space that we have used due to lack of time.

7. References

The following works were used in this research. The list below contains the title, type, authors, publish place and date of the work.

- [1] *Machine learning based adaptive watermark decoding in view of anticipated attack*. Article. Asifullah Khan , Syed Fahad Tahir , Abdul Majid, Tae-Sun Choi. Pattern Recognition - Elsevier, volume 41, chapter 8, pages 2594 – 2610. June 2008.

- [2] *LSB Technique for Secure Data Communication.* Article. P.D.Khandait and S.P.Khandait. Journal of Engineering, Volume 4, Number 1, ISSN : 0973 3663. page 27. December 2007.

- [3] *The first 50 years of electronic watermarkin*g. Article. Ingemar J. Cox, Matt L. Miller. EURASIP Journal on Applied Signal Processing , volume 2002 chapter 1, pages 126 – 132. October 2001.

- [4] Rotation, Scale, *and Translation Resilient Watermarking for Images*. Article. Ching-Yung Lin, Min Wu , Jeffrey A. Bloom, Ingemar J. Cox , Matt L. Miller, and Yui Man Lui. Image Processing, IEEE Transactions on , volume 10, chapter 5 , pages 767 – 782. May 2001.

- [5] *Applying Informed Coding and Embedding to Design a Robust, High Capacity Watermark*. Article. Matt L. Miller, Gwena l J. Dorr, Ingemar J.Cox. Image Processing, IEEE Transactions on , volume 13, chapter , pages 792 – 807. June 2004.

- [6] *Damageless Digital Watermarking by Machine Learning.* Conference. Kensuke Naoe, Hideyasu Sasaki, Yoshiyasu Takefuji. International Conference of Soft Computing and Pattern Recognition. December 2009.

- [7] *High Capacity, Reversible Data Hiding In Medical Images*. Conference. Fallahpour, D.Megias and M. Ghanbari. Image Processing (ICIP), 2009 16th IEEE International Conference on , pages 4241 – 4244. November 2009.

- [8] *Robust digital image watermarking method using wavelet-based fusion*. Conference. D. Kundur. D. Hatzinakos. Image Processing,. Proceedings., International Conference on. Volume 1, pages 544 – 547. 1997

- [9] *ada: an R Package for Stochastic Boosting*. Mark Culp, Kjell Johnson,George Michailidis. Article. Journal of Statistical Software. Volume VV, issue II. 2006.

- [10] *A fast performance estimation scheme for histogram shifting based multi-layer embedding.* Conference. Junxiang Wang, and Jiangqun Ni, 17th International Conference on Image Processing IEEE 2010.