# Laboratory 5 - Features

Guillaume Jeanneret
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
g.jeanneret10@uniandes.edu.co

Jorge Madrid
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
ja.madrid2714@uniandes.edu.co

## Abstract

*In this laboratory, textons are used to identify the category of an test image, belonging to 25 different categories. A dataset of 1000 images, where 750 were training and 250 were test, was partially used to train the machine. TreeBagging method and Chi-Squared distance was used to achieve the categorization using the histograms.*

## 1. Introduction

The categorization of some textures isn't hard for the human eye. For computers, this isn't the case. because a texture depends on the information given by neighborhood and not solely by the pixel information, getting the computer to understand this information is hard. Then, to get a computer to understand the basic information about the texture it is necessary to comprehend how a texture *works*. A texture from a material is repetitive and depends on the scale were the picture was taken. For example, the picture of a rock at close range is different versus the photo of a road full of them. This is given by the resolution and the amount of details that the picture can detect. Given the basics of filtering, the textons were created. The basic idea consist on that a pixel is from a selected texture if the response to the filtering with a filter bank is similar. Thus, given a certain number of clusters, each representation of the pixel can be classified training a machine. The final result consist of each pixel been labeled by a category given by the trained machine.

## 2. Materials and Methods

### 2.1. Database

The database for this task was provided by The Ponce Group at the University of Illinois [1]. It consists of 1000 gray-scale image divided into the train and the test categories, with 750 and 250 images, respectively. These images are divided themselves into different exemplars of the same texture but with a different orientation. There are several photographs of similar tree barks, each with a different orientation.

### 2.2. Methods

#### Textons

The texton dictionary was created by first selecting 8 to 10 images from the 30 exemplars of each texture. As texture is a local feature, we extracted a 100x100 patch from each image. All these image patches were concatenated into a single image (to facilitate indexation during convolution). Then, a bank with 32 filters was created. The whole image collection was convoluted with each of the filters from the bank, producing a 32-layer response. Then, the equivalent pixels from all of the layers were put together into a vector in a 32-dimensional space. These vectors were clustered together using k-means. The centroids of these clusters correspond to the textons in the dictionary. We produce one texton for each image from the training set that we used. Those filters from the bank that are most similar to the texture patterns in the sample images will be the ones that contribute the most to discrimination.

#### Classifiers

We first use a nearest-neighbor classifier to discriminate the textures from the test set. To do this, the filter bank is applied to the test image, and a histogram of the 32-dimensional filter response is created for the image. This histogram is compared by means of a $\chi$ square metric with the textons in the dictionary. The label of the nearest neighbor is then set to the test image.

For the nearest-neighbor classifier we use hyperparameters such as the number of filters in the bank and the metric to evaluate the distance between histograms, for instance. The metric is of particular importance because the closest texton strongly depends on it. We use the chi-square metric because it is well suited for histogram comparison.

We mentioned that we only considered windows of the images to perform the convolution with the filter bank. The purpose of this was to reduce computing time (mainly in the clustering of the texture histograms and the production

| Method | Percentage |
|---|---|
| Chi-Square Distance | 4% |
| Random Forest | 52% |

Table 1. Results of the classification

of the texton dictionary, rather than in the convolution step). The size of such window is of special consideration because choosing too small windows may strongly limit the classification capability of the algorithm. Even if texture is a local feature, windows that are comparable in size to the filters or to the texture patterns will prove unfit.

In the bagged trees approach, window size and number of filters in the bank are also important hyperparameters. The number of trees in the forest is a very relevant parameter, both in terms of computation time and specificity of the training. Larger forests are not as specific to the training set but are more computationally expensive.

## 2.3. Algorithm

The first step was the creation of the filter bank and loading the images and linking them together. To optimize the velocity of the algorithm and given the repetitive patterns of the structures, a window of $100x100$ per picture was taken and not the whole photo. Also, not every image from a category was taken. We took a total 9 photos of 30 because the algorithm took too much time to execute. Continuing with the algorithm, applying each filter in the filter bank, a 32-dimensional *photo* was created. Thus, using this images, k-means was implemented to create the textons. The number k was calculated following the next identity: $k = 25 * \#[images - per - class]$ Here, with every original picture, each pixel was assigned with a texton label. Then, a histogram was created of each training image. This histograms are representation of each class.

To assign each class in the group of test images, two methods were implemented: The first one was using the Chi-Square distance between the histograms of the images. The distance between two histograms is calculated following the equation 1. And the last method was a Random Tree Algorithm using a total of 10 trees. This number was set because the percentage of true positive was not increasing at a reasonable rate in comparison to time of computer time.

The evaluation method was done using the confusion matrix.

$$D(\vec{x}, \vec{y}) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)} \qquad (1)$$

## 3. Results

The general results are shown in table 1.

| Number of trees | Percentage |
|---|---|
| 5 | 48% |
| 6 | 50% |
| 7 | 51 % |
| 8 | 52 % |
| 9 | 53 % |
| 10 | 54 % |

Table 2. Results of the classification changing the number of trees

| Photos per Category | Percentage |
|---|---|
| 4 | 9.4% |
| 8 | 7.4% |
| 9 | 4.4% |
| 15 | *Didn't finished* |

Table 3. Results of the classification using $\chi^2$ distance and changing the number of images, an example of over-fitting

Nonetheless, the results of the Random Forest changing the sizes (with 9 pictures per category in training) are shown in table 2.

In the other hand, The percentage of the classification using different amounts of images and the $\chi^2$ distance changed following the table 3.

## 4. Discussion

Without thinking twice, the Random Forest Algorithm achieved a highly percentage in the confusion matrix diagonal, as seen in figure 1, which achieved a confidence However, overfitting is an issue to be considered. Figure 2 shows the confusion matrix for an experiment in which the number of trees was set to 50. The overall confidence of the classification was 7%, which is unexpectedly low, given that the other parameters were not modified. When too many trees are incorporated, the algorithm learns the details of the train images, instead of their behavior.

Compare the results of figure 1 to those of figure 3. Note that the diagonal elements are more representative for the forest than the nearest-neighbor method. We report a confidence of 4.4% for the nearest-neighbor classifier (when using the same parameters such as image patch size and number of images fr training), although higher confidence values of 9.4% were achieved when the number of training images was lowered to 4.

Training both methods took a very long time. The main reason of this event is that the k-means incorporated in the algorithm is a NP-hard algorithm. The difference in time was set by the number of initial pictures. When the number of images per class was 4, the time required to execute the k-means algorithm was approximately 26 minutes, using a 6 GB memory ram (+ 2GB of video card). With 5, the time

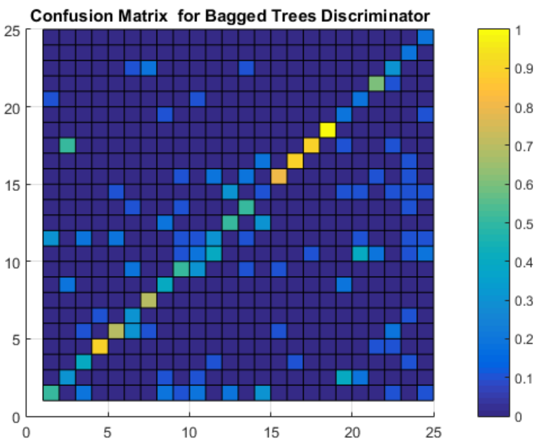

Figure 1. Confusion matrix for the classification with the TreeBagger algorithm. **A forest of 10 trees was implemented.** For each type of textures, 9 of the 30 available images were used for training. Images were windowed to 100x100 px.The overall confidence for this classification was **52%**. Note the stronger weight of the diagonal elements, corresponding to correct classifications, in this matrix.

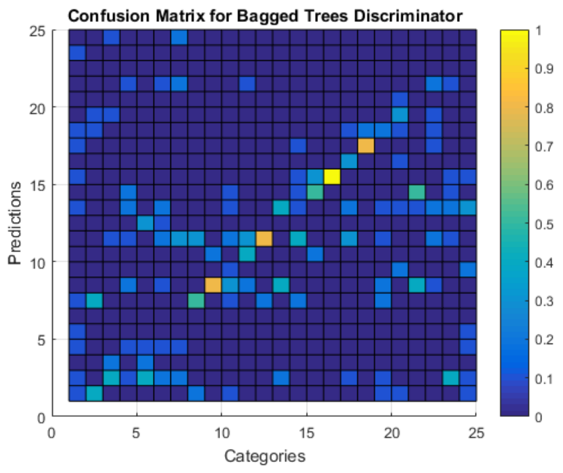

Figure 2. Confusion matrix for the classification with the TreeBagger algorithm. **A forest of 50 trees was implemented.** For each type of textures, 9 of the 30 available images were used for training. Images were windowed to 100x100 px.The overall confidence for this classification was **7%**, which is very low. This may be a result of overfitting.

was 1 hour using the same computer. With 8 pictures, using the course server was a requirement because the process took too long.

The time required to train and apply the Random Forest algorithm took between 7 and 8 minutes, without the k-means process. In the other hand, the $\chi^2$ distance took about the same time.

Taking in consideration the $\chi^2$ distance classification, the confusion was uniformly distributed. In the BaggedTrees algorithm with 10 trees, the confusion was done mainly in the last categories.

## 5. Conclusion

- The limitation of this methods are the high time consuming to achieve some results
- The training needs a big quantity of images to achieve a great model, but it is easy to over-fit this last one, generating a source of error in the evaluation time.
- Training a machine in a laptop can damage it if the computer isn't powerful enough.
- Optimization processes are necessary to achieve a less complex but sufficient algorithm to achieve a real-time result.

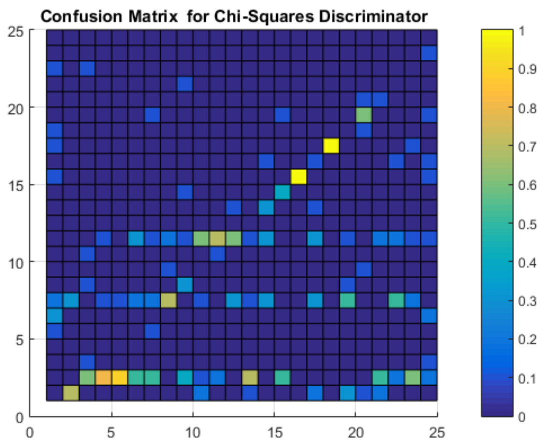

Figure 3. Confusion matrix for the nearest neighbor classification using the chi-square metric. The classificator was trained using 9 images per tye of texture. The overall confidence was of **4.4%** .

## References

[1] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265.

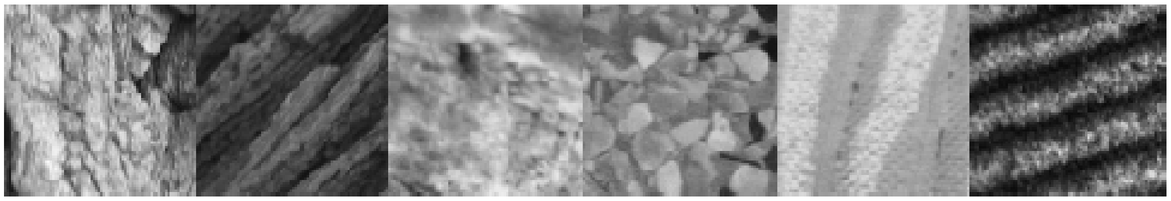

Figure 4. Example of the union of various images.