Textons

Federico Ulloa
Universidad De Los Andes
Cra. 1 18a-12, Bogotá, Colombia
f.ulloa10@uniandes.edu.co

Esteban Vargas
Universidad De Los Andes
Cra. 1 18a-12, Bogotá, Colombia
e.vargas11@uniandes.edu.co

Abstract

In this laboratory, image classification was made using textons. Textons are texture descriptors, the basic units of the patterns that compose a texture. Database from ponce group was used for this work, being divided into training and test sets. Firstly, a texton dictionary was created by convoluting all the filters in the filter bank to crops of the training images, then the responses were clustered using k-means. These texton dictionary obtained, was then used to train a nearest neighbor classifier and a random forest classifier. Lastly, test images were classified using these trained classifiers.

1. Introduction

Texture is many times an important feature for image processing. It can be defined as repetitive patterns in a region of the image. The basic unit of one of these patterns is called a texton, and was defined by Julesz in 1981.[1]

A texton map of an image is a representation of the different textures of an image given a texton dictionary. A texton dictionary is the resultant categories of textures obtained from the following procedure. Basically, pixels of an image are filtered by the filters in the filter bank (a collection of filters in different sizes and orientations) and have a different response to each. With these responses, K-means is used to cluster pixels with similar responses into texture categories, obtaining the texton dictionary. The texton histograms of the images obtained are used to train a classifier such as nearest neighbor or random forests to classify other images.

The texton representation of an image tells us how is the image composed with respect to texture. In other words, a texton map of an image shows how the different regions of the image responded to the filter bank, giving us an idea of which textures appear (or are closer to) in each part of the image.

Clearly, some filters from the filter bank are more discriminative than others, as from each texture, a different response is obtained to each filter. As there are some elements that are more common among the patterns in textures, these will tend to have a higher chance of a high response in general.

2. Methodology

2.1. Database

The database used in this laboratory is from the ponce group. [2] This database consists of 25 different classes of images with 20 images each. Each class contains images with the same kind of texture, and each image is composed of a single texture. All the images are gray scale, with 640x480 pixels in size and in JPG format. The dataset was separated into train and test sets.

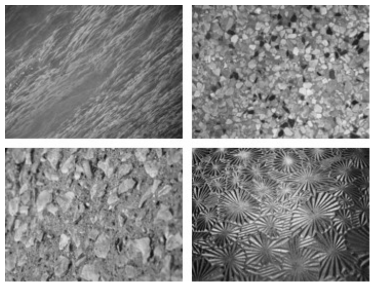
Figure 1: Sample images of the data-set

2.2. Training

Firstly, as this process is very expensive computationally as each training image has to be convoluted with each one of the filters in the filter bank, and there is a large amount of images relative to this, images were re-sized to 50x50, 100x100 and 200x200. This can be done because we know that each image is composed of a single texture. Then, to create the texton dictionary, each re-sized image from the training set was convoluted by every filter in the filer bank provided. After that, all the responses to every filter were clustered using kmeans. For this, k of 25, 40 and 55 were used, meaning that we built dictionaries composed of 25, 40 and 55 textons respectively. The number of textons were chosen in that range because they are greater than the number of classes found in the dataset, knowing that the images in each category are made up of a single texture each, and we wanted to determine which of these dictionaries works better for to classify images in the problem.

Then, after obtaining the texton dictionaries, we computed the texton histograms of the training images and with these, two classifiers were trained and evaluated: nearest neighbor and random forest. Naturally, distance between histograms were used to train both classifiers.

2.3. Nearest neighbors

For the nearest neighbors classifier, different experiments were done. First, samples of 50, 100 and 200 pixels of each images were selected with different number of textons 25, 40 and 55 in order to determine the best method hyperparameters combination. Then, as the KNN classifier only have one hyperparameter, the K amount of nearest neighbors was varied to determine the best one.

2.4. Random forests

For the random forests, the hyperparameters found were the number of estimators (trees), the maximum depth of the trees, the minimum samples required to split an internal node and the minimum number of samples required to be at a leaf node. In order to find the best hyperparameters of the classifier an exhaustive search were made using a Grid Search over a grid of the hyperparameters, 3-fold cross validation were used to train and obtain the score of each combination.

3. Results

3.1. Nearest neighbors

After running experiments with different amount of pixels, the best results were obtained with 100 pixels,as shown in the figures. Apart from the ACA, computation time was also a determining factor in choosing the optimal number of pixels. With 50 pixels and k=40, the algorithm took 315 seconds, 996 seconds for 100 pixels, and 3853 seconds for 200 pixels. Results were not significantly better for 200 pixels compared to 100 pixels and it took a lot more time, reason why we chose 100 pixels as the optimal number of pixels.

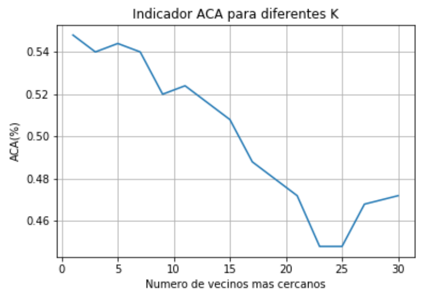

Figure 2: ACA for KNN with 50x50 pixels

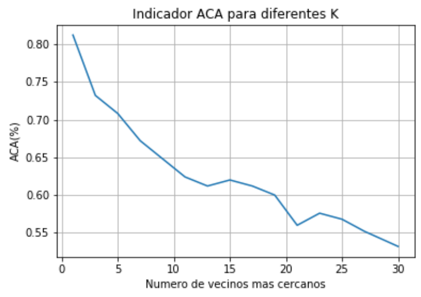

Figure 3: ACA for KNN with 200x200 pixels

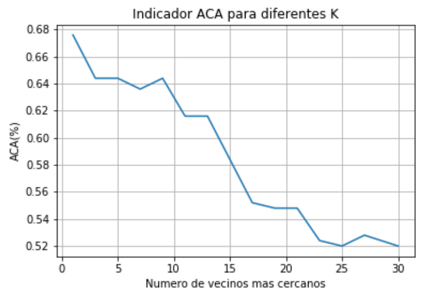

Figure 4: ACA for KNN with 100x100 pixels and k=25

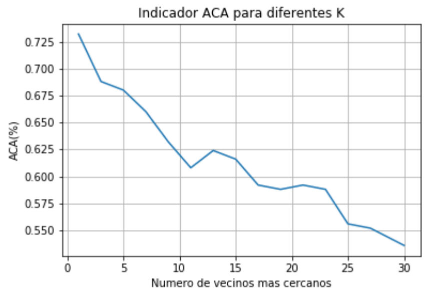

Figure 5: ACA for KNN with 100x100 pixels and k=40

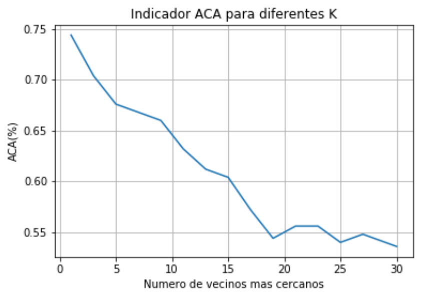

Figure 6: ACA for KNN with 100x100 pixels and k=55

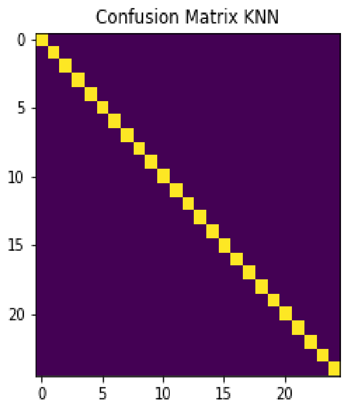

Figure 7: Train set confusion matrix, ACA=100 percent

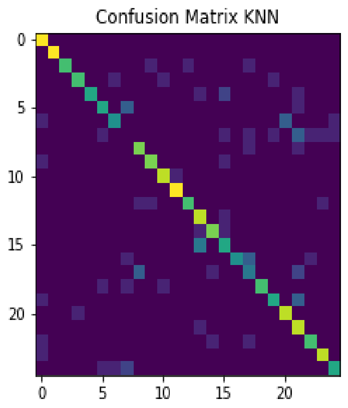

Figure 8: Test set confusion matrix, ACA=71.2 percent

For the variation of K (Number of nearest neighbors), as seen in the figures, the ACA decreases as the K increases, because with a big K, the classifier starts to overfit the data. As seen in figure 4, the best ACA obtained with the optimal number of pixels is near to 1 which. On the other hand, the hardest class for KNN to classify was class number 7 which corresponds to water see figure 9. The train time for KNN was 0.38s which was pretty fast taking into account that speed is one of the properties of KNN classifier.

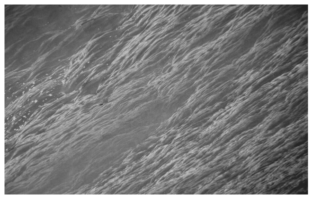

Figure 9: Example image of class 7 corresponding to water

One possible reason of why this is the hardest class is because is really similar to other classes like wood class, and the differences are in color but we are only taking into account texture.

3.2. Random forest

For the random forests, as it can be seen in the figures, the ACA increases with the depth of the trees, until it reaches and optimum and then the classifier also starts to overfit the data. As demonstrated in figure 9, an ACA of near 0.75 was reached for a depth between 10 and 15. After doing the grid search, the best hyperparametes found for the Random forest classifier were, maxDepth=20 and numberOfTrees=2000.

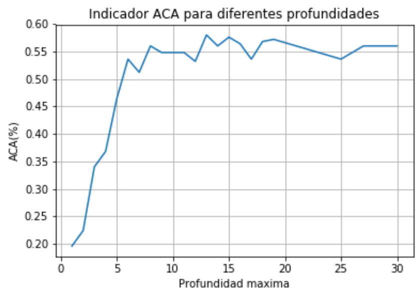

Figure 10: ACA for RF with 50x50 pixels and k=40

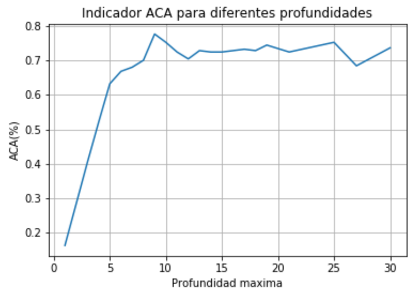

Figure 11: ACA for RF with 200x200 pixels and k=40

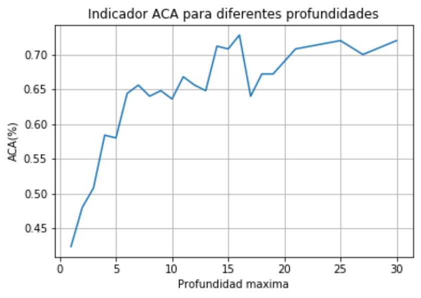

Figure 12: ACA for RF with 100x100 pixels and k=25

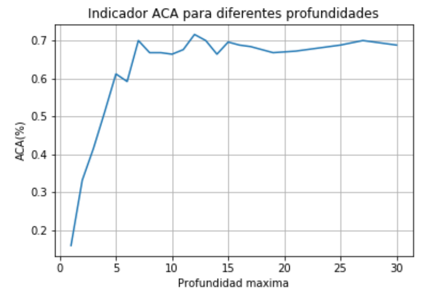

Figure 13: ACA for RF with 100x100 pixels and k=40

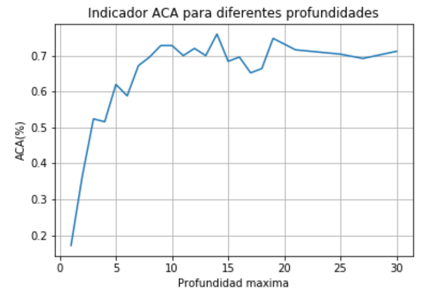

Figure 14: ACA for RF with 100x100 pixels and k=5

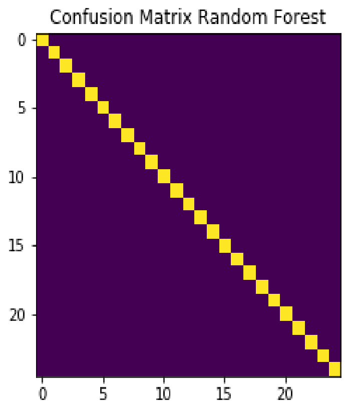

Figure 15: Train set confusion matrix, ACA=100 percent

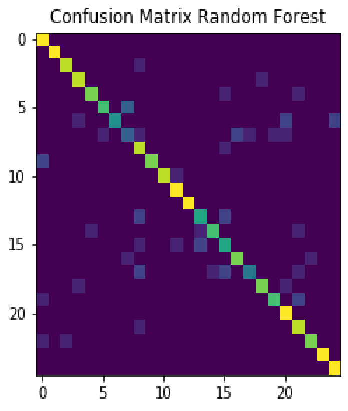

Figure 16: Test set confusion matrix, ACA=80 percent

Analyzing figure 16 the hardest class to classify for random forest is again class number 7, how ever unlike KNN that wasn't able to classify any of the 10 images of water class in the test set correctly, random forest was able to classify 4 of them. Also, same as KNN the ACA in the train set was 100 percent. The training time for random forest was 1.68s which is a little bit slower than KNN but gives better results. Finally, the total time to compute the textons and train the random forest in its final version was 998s, knowing that the required time to train the random forest is around 2s, the time expended to create the texton dictionary is about 996s.

4. Conclusions

Texton representation of images is a good way to do classification in various applications, but it also tends to be very expensive computationally. For this reason, it is very important to find a way to sub-sample the images. In this case, images were re-sized to reduce the computation time.

For the two classifiers used, it is also important to choose the parameters correctly to obtain the best results with an optimal time. Overall, random forests worked better than nearest neighbors. This is because random forests varies the amount of characteristics evaluated in each tree randomly, while with nearest neighbors, the number of characteristics is not random. Also, nearest neighbors overfit the data faster than random forests, as random forests reduce the overfit by its random nature, meaning that it will choose random features in each tree to build the forest and train the classifier.

Creating the texton dictionary takes a very long time because for this, each training image has to be convoluted by every filter in the filter bank, and all the responses are clustered. All this takes a big amount of computation.

The method could be improved by sub-sampling the images in another way, like taking a single patch of each image as they contain a single texture, and comparing other classifiers, like support vector machine, for instance.

References

[1] S. Zhu, C. Guo, Y. Wang and Z. Xu, "What are Textons?", International Journal of Computer Vision, vol. 62, no. 12, pp. 121-143, 2005.

[2] "Ponce Research Group: Datasets", 2018. [Online]. Available: http://www-cvr.ai.uiuc.edu/ponce$_grp/data/.[Accessed : 26 - Feb - 2018]$.