Image classification using Textons

Julio Nicolás Reyes Torres
Universidad de los Andes
jn.reyes10@uniandes.edu.co

Juan David Triana
Universidad de los Andes
jd.triana@uniandes.edu.co

Abstract

This article presents 2 methods of supervised classification for images using textones. The textons are used as a strategy of representation for images. A classification model called "KNN" (Nearest neighbors classification) that implements uniform weights around a set of query points evaluated with a kind of metric. The other one is called "Random Forest", the main idea is to implement a robust system that evaluates different branches of a main tree. The results are evaluated into a "confusion matrix" which represents the quality of the output of a classifier.

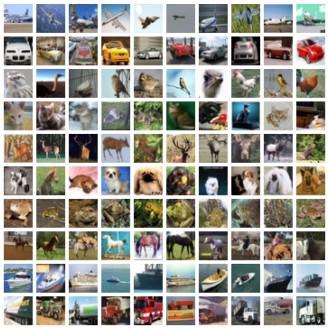


Figure 1: Example of the data set selected

1. Introduction

One of the most common computer vision problems lies upon image classification. There are various approximations of solutions towards this problem that include classification modules such as support vector machines, random forests, nearest neighbors and more. Another factor needed to be able to classify images is the basic descriptor which extract important features from images depending on its nature [6]. On this specific article and practice textons are going to be used to describe features from natural images. Textons are fundamental micro-structures that extract features that come from the image's textures [1]. Furthermore, two classification methods are going to be implemented: k cluster nearest neighbors and random forest. The purpose of this article is to investigate how good are textures and textons as descriptors for a classification problem.

2. Methods

2.1. Data set

The data set used for this specific article is available on the Computer Science University of Toronto web page [8] and was obtained by Krizhecsky and his collaborators. It consists of 60000 images in ten different classes. Among these classes are dogs, frogs, horses and more. Each image is sampled in a 32X32 format divided into train (50000) and test (10000).

2.2. Filtering and texton creation

In order to obtain a feature description based on textures, it is necessary to implement a bank of filters. This bank includes various types of filters of gaussian modules. In general, each filter has a default standard deviation and each one rotates to extract border features. Since the images have a small resolution, a standard deviation of 0.6 was used to limit the boundaries of each filter in an adequate manner.

To obtain a texton map related to an image i is necessary to convolve each filter with the image to obtain a response. Afterwards, k-means was implemented to classify each texture. The centroid of the output of the k-means algorithm allows the selection of the closest texture according to the training algorithm. Each texture contains important information regarding the small patterns from each image. This includes gradients and filter softening obtained from the filter banks. One of the most important characteristics of textures is the outlined patterns of each natural image.

2.3. Classification

As it was mentioned before, two classification methods were used to evaluate texture importance on image description: k-cluster nearest neighbors and random forests.

2.3.1 K-cluster nearest neighbors

The nearest neighbors classification algorithm is a supervised classification method which uses uniform weights around a set of query points. These query points are evaluated with a selected metric [5]. In most cases euclidean or chi-squared metrics are used. For this specific case, the python module sci-kitLearn was used, which implements the Minkowski metric space. It is characterized to be a tensor that combines euclidean distances in an extended dimensional space (3D-4D) [4].

The module algorithm includes various hyperparameters that can be modified to obtain different results. The basic parameters used in this case were:

1. Number of neighbors = 5
2. Uniform weights
3. KDTree algorithm
4. Leaf size = 30
5. Minkowski metric

2.3.2 Random Forest Algorithm

It is a supervised classification algorithm widely used in classification and regression problems, it is based on the construction of deciding trees. The main idea is to implement a robust system that evaluates different branches of a main tree, these branches represent different structures of impartial models that evaluate characteristics not necessarily correlated and then evaluate them.

This algorithm is one of the most accurate of its kind, preserving the idea that just like in a forest, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results. [7, 2]

The Random Forest function for python includes different hyperparameters that can improve its predictive effectiveness [3], the most important are:

1. Criterion : default="gini"
2. max depth
3. min samples leaf
4. max leaf nodes

Advantages

- It is a very accurate learning algorithm
- Run efficiently in large databases
- It offers an extrinsic model to analyze interactions between the variables
- Have an effective method to estimate lost data and maintain accuracy when data is lost.

Disadvantages

- Can overadjust in certain groups of data noisy classifications
- If there are correlated attributes in performance, smaller groups are favored over larger groups
- In general the classification is difficult to be interpret.

3. Results

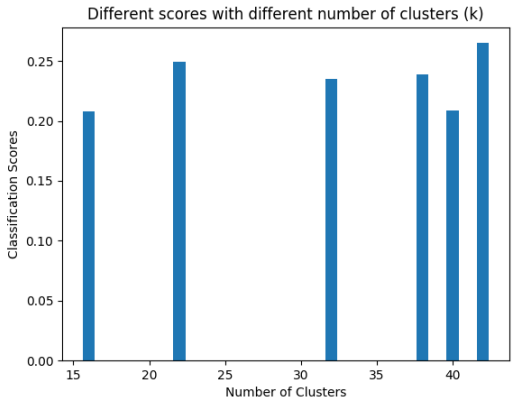
Note: In the article the matrix of confusion and the information of the precision and time are presented in more detail for the Random Forest classifier, because it presents a better classification adjustment than the KNN.

3.1. Nearest Neighbors (KNN)

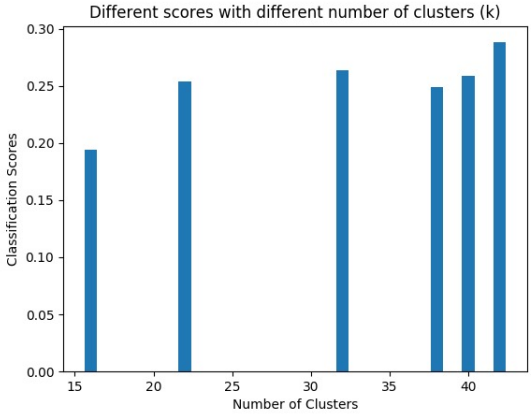
The initial base data used to train the classifier consisted on 1% of the images of mode 1 as well as the test base (500 images each). To evaluate the strength of the algorithm it was necessary to implement the confusion matrix. The mean score precision value was 0.23, meaning that the algorithm can identify around of 23% the test images correctly.

Additionally a change on the cluster number was iterably changed to obtain different valued depending on the amount of textons used, since cluster numbers are completely randomized and do not have a specific value. Following is presented the different scores obtained for the case:

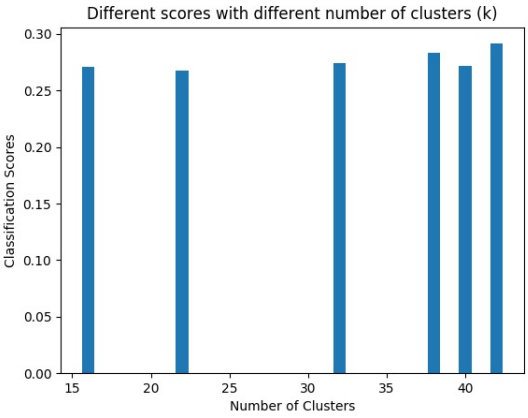
1. **Test Images = 1000:**



2. **Test Images = 3000:**



3. **Test Images = 5000:**



Regarding the confusion matrix, according to the baseline of the algorithm, the following figure shows the proper confusion matrix with 500 images.

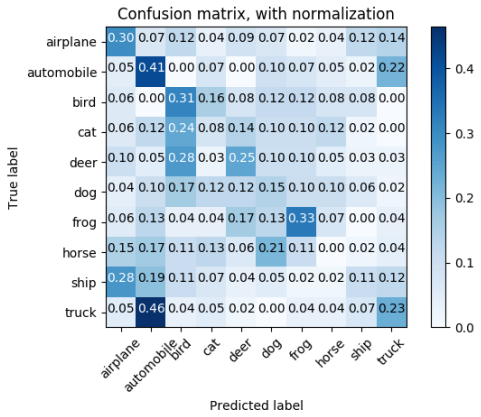


Figure 2: Confusion matrix for the KNN algorithm

3.2. Random Forest

The "Test set" kept constant with: 1000 *images*
To evaluate different responses of the Random Forest, the number of clusters (k) and the number of training images were varied. Following is presented the information about the accuracy and the spending time for different values of the "Training set", and the "confusion matrix" for the best results:

1. **Train Set = 1000**

# Clusters (K)	Accuracy (%)	Time (s)
16	24,9	419,915
22	29,0	198,46
32	30,7	528,013
28	29,4	575,9
42	30,7	310,96

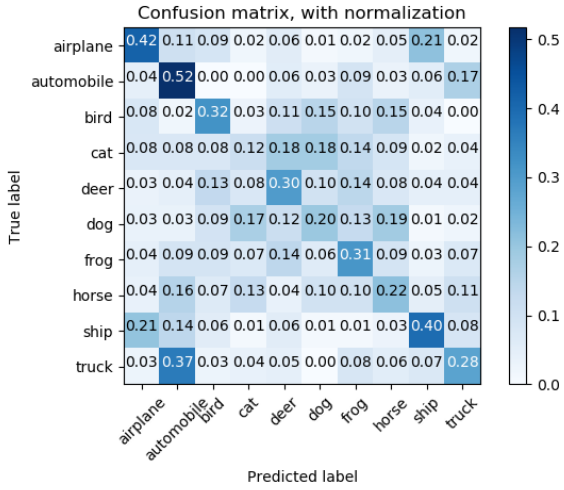


Figure 3: Confusion Matrix for K = 32. Accuracy = 30, 7%

2. **Train Set = 3000**

# Clusters (K)	Accuracy (%)	Time (s)
16	25,9%	410,85
22	25,5%	400,77
32	33,2%	513,28
28	34,3%	535,24
42	34,6%	1321,89

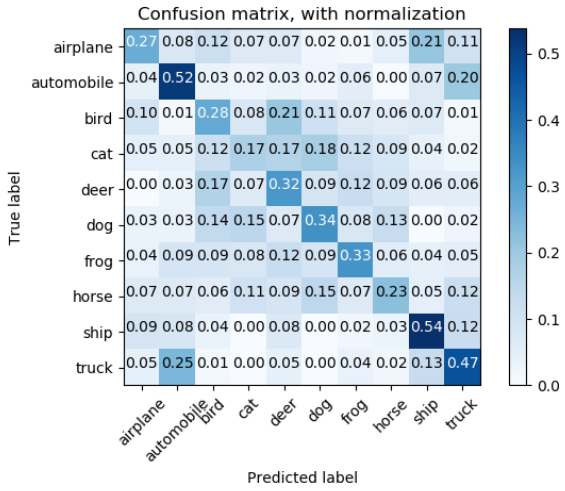


Figure 4: Confusion Matrix for K =42. Accuracy = 34, 6%

3. **Train Set = 5000**

# Clusters (K)	Accuracy (%)	Time (s)
16	24,8%	617,46
22	32,4%	807,13
32	32,8%	1098,62
28	34,1%	1132,32
42	35,2%	1842,24

4. Discussion

One of the most challenging problems about using textons is the computation time and storage use. A normal run (600 images) of the algorithm on a powerful server took around 2 minutes to complete. Moreover, using a sample of 10000 images took 60 minutes to complete. This is a very long time lapse regarding computation. The reason why is because to obtain a texton dictionary and assign this dictionary takes a vast amount of RAM memory. This comes in hand with the specific number of textons and filters used. To recall, cluster usage in the algorithm is applied on k-means, for this purpose. So, texton dictionary creation and assignment use vast amounts of storage due to the k-means algorithm that needs to be applied to classify the different textons.

On the other hand, the time to compute the classification methods used on these datasets was greatly smaller than computing textons. Although computation time increased upon the increase of number of clusters for classification, it never exceeded a five second margin. This means that the whole computational storage capacity is focused on the texton representation due to the great amount of operations that need to be made per image.

Now focusing on the proper results of the KNN classifier, it is interesting to see that the number of clusters (k) for this case do not change drastically the score of prediction. Moreover, it seems that the amount of textons needed to optimize classification and feature extraction is below or among the number of filters used (16) on the filter bank. It is possible that the small resolution of the dataset natural images don't need a high amount of filters. Since the operations are made on a finite domain, having a standard deviation of 0.6 can mean that there is no integral amount of pixels to cover.

5. Conlusions

- The textons can be used to describe features from natural images, they can extract fundamental microstructures that come from the image's textures. As we can saw, the textons are the "strategy" of image representation that are implemented to run a classification model of prediction.

- KNN it's a supervised method which uses uniform weights between the neighbors, the different images obtained in KNN section, represents the accuracy depending of the number of Training images. It can conclude that the more training images there are, the better the trend of grouping on the same average.

- The Random forest Algorithm got the best accuracy response, the predictive behavior improved as the number of clusters and the number of training images increased, the best result was around 35%. The Confusion Matrix gave a good visual representation of the algorithm response.

References

[1] e. a. Chu, S. What are textons? *International Journal of Computer Vision*, 62(1):121–143, 2005.

[2] Dataaspirant. How the random forest algorithm works in machine learning, 2017.

[3] S. learn. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.2 documentation.

[4] W. MathWorld. Minkowski Metric.
[5] Scikitlearn. 1.6Neares Neighbors.
[6] Stackoverflow. What is a feature descriptor in image processing?
[7] Towards Data Science. Random Forest in Python, 2017.
[8] T. University. The CICAR-10 dataset.