Classification with texture features

Catalina Gómez
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
c.gomez10@uniandes.edu.co

Diana Herrera
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
ds.herrera10@uniandes.edu.co

Abstract

Classification is one of the main tasks in vision computer field. It can be achieved by analyzing textures from images, as most of images belonging to the same class may have similar textures. These are understood as local information concerning not only the intensity value of one pixel but the set of intensities from each pixel and the ones in its vicinity. In this experiment, a set of 32 filters with different orientation, geometry and scale was used to build 40, 60 and 100 textons through k-means, based on a dataset composed with 25 different categories, to train Bagged Trees and fine KNN classifiers. Bagged Trees system performed better than KNN but both methods exceeded 50% ACA when using 100 textons.

1. Introduction

Classification is one of the main tasks in vision computer field. It can be achieved by analyzing textures from images, as most of images belonging to the same class may have similar textures. These are understood as local information concerning not only the intensity value of one pixel but the set of intensities from each pixel and the ones in its vicinity. In order to gather information from this vicinity, filtering operation can be applied to reveal information about local structures within the kernel or mask, according to the magnitude of the response. Different types of filters reveal different patterns, and if two pixels have similar responses to one filter, then they may have a similar texture. After applying a collection of filters to all the pixels within an image, it can be represented by the answer to all the filters.

The main elements that characterize a texture are known as textons, which are required for texture perception in human vision, and represent the relationships between pixels within a region [1]. The distribution of textons over an image gives an idea of the different texture patterns that make up the image. This distribution, or histogram of texton map, can be used to represent a particular image, and hence classify images based on their distribution of the basic elements that make up its texture, when compared to training data, that was to train a model in which each distribution correspond to a category. In addition, textons depend on the collection of filters applied to highlight the texture patterns, which may include different orientations and scales to characterize as complete as possible these patterns.

2. Materials and Methods

2.1. Dataset

The dataset used was provided by Ponce group [2]. For training and testing, the algorithm contained similar instances for 25 different types of textures. In total there were 750 images within the training set and 250 for testing. Each type of texture was represented in the training set by 30 images that have similar repeated patterns, but have some differences at the scale (or size) of the structures that make up the images. In addition, some of these textures included stripe-like patterns that could have different orientations, such as vertical, horizontal or with some inclination. The 25 types textures included a diversity of geometries like squares, circles, stripes (as mentioned before), or asymmetric spots. Each type of texture used to train the algorithm cloud be evaluated with 10 instances within the testing set, which had similar patterns thus allowing for an accurate classification with the trained models.

2.2. Methodology overview

The overall description of the method used is described, although different trials were done and specifications will be described in following sections.

1. Crop all the training images to a quarter of their size, as their textures were repetitive throughout the image and computational power required decreases significatively.

2. Create a filterbank composed with 32 filters based on different geometries, orientations, variance and scales: half of them with size 13*13 and the other one 19*19. The family of filters created attempted to match the

features that characterize each type of texture, considering their diversity of orientations in lines and shapes in patterns, specially round and elongated ones. All of them are based on Gaussian filters and their derivatives and were created by using David Martin *oeFilter* function.

3. Run the filterbank created over all training images concatenated side by side.

4. Create a texton dictionary by computing textons from the response of applying the filterbank to all training images, and by using k-means with the response of each pixel from each image to the whole family of filters. K textons were computed according to the specified k value, corresponding to the centroids of the clusters made by k-means algorithm.

5. The computed textons were assigned to each training image, to obtain a map of textons from each one.

6. Texton histogram was computed for each training image and a matrix was built with each row corresponding to the histogram of each image. Labels were added to the last column of the matrix ranging from 1-25 according to the texture represented.

7. Using the application of Matlab, Classification Learner, a Bagged Tree Classifier and a fine KNN Classifier were trained with the resulting texton histogram matrix from training images. The first one had (parameters) and the second one used 1 distance as results with greater amounts were worse.

8. All test images were cropped to quarter their size, filterbank was applied to them, textons assigned according to the answer of each pixel to filterbank and a matrix with each image texton histogram was built.

9. Prediction of the trained Bagged Tree Classifier and fine KNN Classifier was applied over the matrix of test images.

10. Confusion matrix was built, normalized and Average Classification Accuracy was calculated.

2.3. Specific trials performed

Three trials were done, specifications from each are shown in Table 1.

Table 1. Trials performed

Trial number	Cropped size (vs original)	Number of training images per texture	K (number of textons)
1	Quarter (121*161)	10	40
2	Quarter (121*161)	20	60
3	Quarter (121*161)	30	100

2.4. Deeper explanation of methodolody steps

- *Data preprocessing*: Since texture is characterized by repetitive patterns, the information of the texture is distributed along all the image. To represent the texture, it would be enough with a cropped version of the original image. Particularly, the images were cropped to one fourth of its dimensions and a unique trial with halve their size was made. The reduction in the number of pixels would reduce the number of instances to classify with the k-means algorithm, and hence, the computational complexity of this process. Another option was to reduce down the number of instances for each type of texture within the training set because the images of the same texture were very similar. This alternative was also applied, and the number of instances taken to train the algorithm varied in each trial, selected randomly. The number of images within the testing set was not changed.

- *Creating the filterbank*: When creating the filterbank, filters created mush be carefully chosen as some of them could be more discriminative than others depending on the textures contained on the images from the dataset. For example, if all the textures contained in the images are made of lines, certainly line filters will be more discriminative than point filters, as a line is a set of points. Geometry is an important parameter to take into account when creating the filterbank. Additionally, filter orientation is of great importance as not every texture has the same orientation and if there are some common orientations on dataset images, filters with those orientations must be used. Similarly, scale is a parameter of great importance because similar textures might have different scales and therefore, filters with same shape and orientation but different scales will give a better discrimination.

- *Creating the texton dictionary*: To create the texton dictionary, each pixel must be represented in a high-dimensional space based on its answer after applying a family of filters. The answer for each filter at each pixel is concatenated within a vector and the total of vectors of all the pixels for all the training images is the input for the clustering technique with k-means, in which we must specify the number of clusters desired (K). The output of the function implemented in Matlab includes the label for each pixel and the coordinates of the centroids at the representation space. These centroids are known as textons, and make up the texton dictionary.

- *Selecting number of textons or K parameter*: The number of textons depends on the k parameter used for k-means, which determines the number of clusters to

group all the data, according to the similarity of the responses for the filters. There is not an exact method to choose K, in most literature different K are tried and results are compared. Usually, the larger the number of textons, the greater the ACA from classification obtained as each texture can be represented in a different and more precise way. Nevertheless, increasing this parameter increases the computational power required to perform k-means, hence a balance must be done between a large enough number of textons to discriminate different textures and an amount that computational power available can process.

- *Description of the classifiers, their hyperparameters and distance metrics*: Two classifiers were used, Random Forests and K-Nearest Neighbor(KNN). The first one is considered a computationally efficient technique that can operate quickly over large datasets [7]. It works basically as a sequence of binary decisions that lead an input to a single leaf, belonging to one class of the classification. The idea is very straightforward although hyperparameters such as the amount of splits or the amount of trees in the forest can be specified to increase accuracy. On the other hand, KNN compares the distance of the point being tested to the points with whom the algorithm was trained. The amount of training points with whom the distance is evaluated is determined by the k hyperparameter, and it means the k closest neighbors to the tested point. Depending on the meaning of the points used in the classifier, the distance used can be chosen as another hyperparameter. In our case, the points represented histograms, hence Chi square distance can be used, in which minimum value means similar histogram, or else kernel distance can be used, in which a value closer to one means similarity [5].

- *Hyperparameters of the classifiers and how to choose them*: First of all, it is paramount to note that for both classifiers used, Bagged Trees and fine KNN, the texton histograms, calculated for training images after assignment of textons was made, were the features used to train them. The process to do so corresponds as follows: once the textons were extracted, the vector of representation (with Number of filters dimension) for each pixel of each image within the training set was compared with the textons using the euclidean distance. The texton from the directory that minimized the distance was assigned to each pixel, as a different label for each texton, giving as a result the texton map for each image. Then, the histogram of the texton map for each of the training images was computed and used to train two classification models, K-Nearest Neighbor (kNN) and Random forest. The first one categorizes new points based on their distance to the points in the training set, in our algorithm the points are the histograms, and this metric can be changed to different distances, such as city block metric, euclidean, Mahalanobis, cosine, correlation, Chebychev distances, among others [5]. kNN searches for the k closest points in the data used to train to a new set of points. The parameters k and distance metric can be adjusted in order to change the result of the classification. The second type of classifier, was implemented with the TreeBagger function, which combines the result of many binary decision trees to reduce the effect of overfitting and improve generalization [6]. One of the parameters that can be adjusted is the number of trees, the larger the amount of trees, the more specific classification will be made, because every image will be evaluated in every tree and in the end, the resulting leaves from each tree will be averaged or will vote to classify the image evaluated, avoiding the problem of misleading classification caused by a one node mistake. Similarly, number of nodes or splits within one tree can be stated and the larger the amount, the more precise the classification will be, as the probability for different images to fall into the same leaf is smaller, because they must have fallen through the same binary answer in loads of nodes. Nevertheless increasing any of these parameters increases computational power required and can lead to overfitting, therefore a balance must be made because not always increasing these hyperparameters increases significantly the accuracy of the classification [8].

2.5. Machine used

The computer used contained an Intel core i7 5930k processor with 6 cores and worked at 3.7GHz, with 32Gb DDR4 ram at 3GHz. Storage of 512Gb sata SSD and 2Tb HDD at 7200rpm.

3. Results

3.1. Confusion matrices

3.1.1 Training images, Random Forest

Figures 1 to 3 correspond to the confusion matrix for the training data used for the Random Forest model.

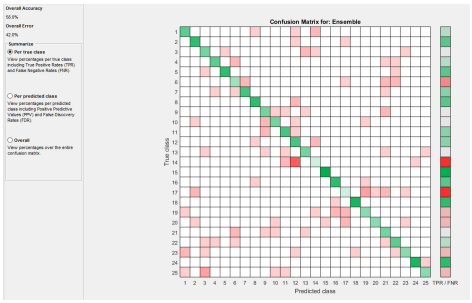

Figure 1. Confusion matrix for training images, first trial

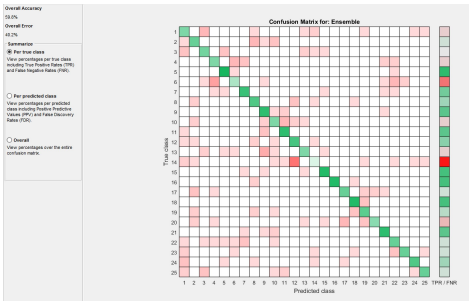

Figure 2. Confusion matrix for training images, second trial

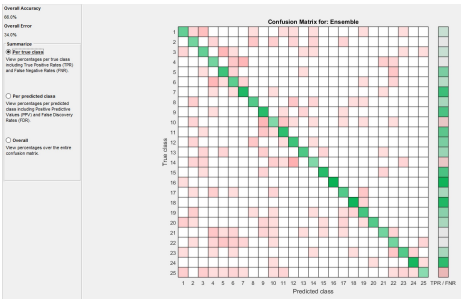

Figure 3. Confusion matrix for training images, third trial

3.1.2 Training images, KNN

Figures 4 to 6 correspond to the confusion matrix for the training data used for the KNN model.

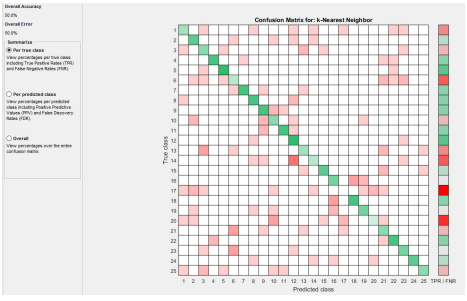

Figure 4. Confusion matrix for training images, first trial

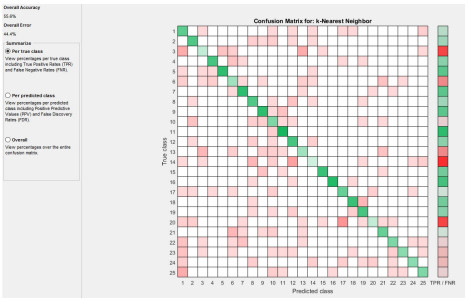

Figure 5. Confusion matrix for training images, second trial

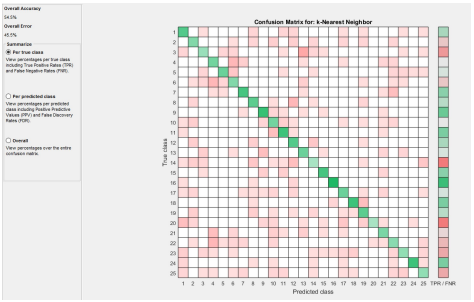

Figure 6. Confusion matrix for training images, third trial

3.2. Testing images, Random Forest

Figures 7 to 9 correspond to the confusion matrix for the testing data used to validate the Random forest model.

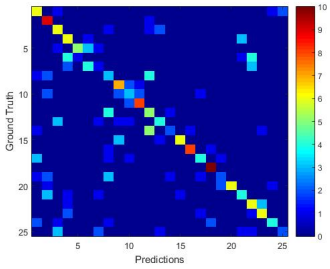

Figure 7. Confusion matrix for testing images, first trial

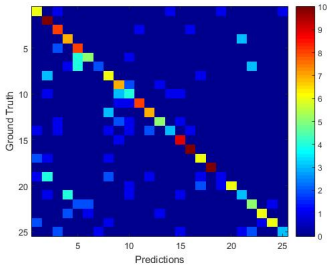

Figure 8. Confusion matrix for testing images, second trial

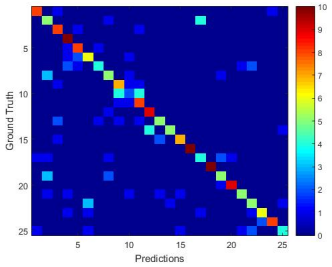

Figure 9. Confusion matrix for testing images, third trial

3.3. Testing images, KNN

Figures 10 to 12 correspond to the confusion matrix for the testing data used to validate the KNN model.

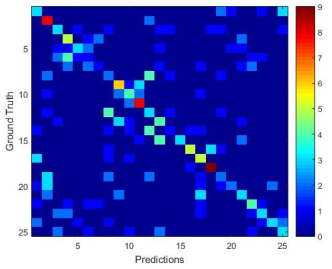

Figure 10. Confusion matrix for testing images, first trial

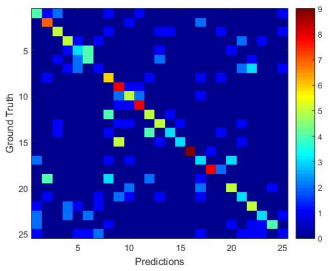

Figure 11. Confusion matrix for testing images, second trial

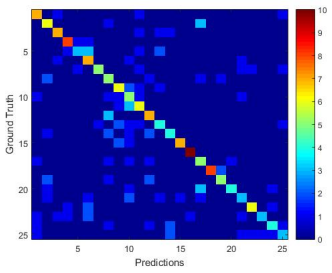

Figure 12. Confusion matrix for testing images, third trial

Based on confusion matrices, Average Classification Accuracy (ACA) was calculated. Other metrics such as Error Rate, Precision, Recall and F-score would have been useful to give further information about whether major problems were related with the precision or recall of the algorithm, but were not calculated this time.

3.4. ACA results

Table 2. Trials performed

Trial number	ACA train	ACA test
1 RF	58%	49.2%
1 KNN	50%	37.6%
2 RF	61.6%	60.4%
2 KNN	55.4%	45.2%
3 RF	66%	65.2%
3 KNN	54.5%	55.2%

4. Discussion

When analyzing the results obtained from the classifiers, it is noticeable that Random Forest performed better than KNN in the three trials done in percentages between 5 and 10%. This might be because Random forests do not assume features behave or interact linearly, unlike KNN, and they can handle properly high dimensional spaces, such as the one used here with 32 dimensions, as well as a large number of training instances [9]. In general, all results from both classifiers and the three trials were satisfactory, considering that images had to be classified among 25 classes, which is a 4% random chance to classify them properly. ACA results for training are higher than testing because the classifiers analyzes their features to be trained, while it only assigns labels to test according to the respective algorithm of the classifier. It is paramount to note that as would be expected, the higher number of training images and textons, the better results were obtained, although doing so increases computational complexity exponentially. This must be carefully balanced as there is a threshold in which increasing further these parameters does not improves significantly the result but does increase computational complexity.

The most time consuming step of the algorithm is creating the texton dictionary, because each pixel of each training image is being represented by a 32*1 vector containing the response to every filter. K-means must first place each one of these points in a 32 dimensional space. Afterwards it initializes k-random centroids, and then proceed to calculate the distance of each point in the space to each centroid, to assign the tag of the closest centroid. This process requires a huge amount of calculations but that is not it. Afterwards, the algorithm recalculates the mean

of the points belonging to the same cluster and use these points as centroids. The distance from each point in the space to each centroid is calculated again and the process repeats the number of iterations determined as a parameter or in our case, repeats 100 times until a sub optimal solution is achieved, as probably it is not possible to find an exact solution. The amount of calculations per cycle would be equal to the multiplication of the number of pixels in each training image by the number of training images used by the number of cyclces by k in a 32 dimensional space, Which is certainly a huge amount and increases quadratically when increasing the amount of training images or the k number.
As for the training and predicting process of the classifiers, Random Forest requires slightly more time than fine KNN but both of them are ready in less than 30 seconds. Prediction in both classifiers is extremely fast.
Furthermore, when confusion matrices are analyzed we can notice that texture categories which contain very similar patterns, or better called textures, are the ones that cause most confusion because textons assigned are probably very similar due to their very close similarity in textures. This may happen between classes 1, 4 and 21.

Additionally, besides the obvious limitation in terms of ram and computational power to run trials with bigger cropped images or higher k, probably the most important limitation of this method is that it uses k-means, which is a unsupervised technique used to group points based on the notion of similarity and the number of clusters initially defined. For texture classification, the ideal number of clusters (k) would be all the texture patterns within the images of the training set; however, we knew there were 25 categories to train the algorithm, but not all the texture patterns present in all the images. This implies, that the initialization of the problem, including k, determines the final partition output, which corresponds to the textons. Some representations will result in better discriminative and characteristic patterns, depending also on the collections of filters used to represent the pixels. On the other hand, despite the fact that the filter response at each pixel depends on its vicinity, when k-means algorithm groups pixels with consistent texture (or filter responses), we do not know which texture corresponds to which tag, and there is no correspondence or consecutiveness between the tags, i.e., consecutive tags do not correspond with adjacent textures, hence some spatial information is lost.

The method used could be improved by taking into account information from intensity histograms together with texton histogram, as luminosity is also a powerful differential factor to classify images. Additionally, creating the filter bank in a more detailed way, comparing scales contained in training dataset, all the shapes and most common orientations would led to a more discriminative filter bank that differentiates better even similar textures. On the other hand, another option is to consider shape descriptors since they take into account information from its vicinity, like texture. One of these methods is called Scale Invariant Feature Transform (SIFT), which computes the histogram of oriented gradients (HOG) for every patch centered at each pixel. These representation in a high-dimensional space could be used as an input for k-means algorithm, to find the centroids and then assign labels to each pixel according to its closest centroid [10].

5. Conclusions

- Texture characterization through texton dictionary is a useful method to classify images, since it takes into account the local information to group pixels with consistent textures.
- The definition of the filterbank must be made according to the texture patterns that one is willing to find within the images to classify. The idea is that each filter highlights some distinguishable features, in order to provide a better representation in the high-dimensional space.
- Increasing the amount of textons used to characterize different textures and the amount of training images does improve the result but it also increases computational complexity exponentially.
- K-means is a useful method to perform non-supervised classification although it is computationally expensive.
- Different kinds of classifiers can be trained to be used in classification tasks. Random Forest performed better than KNN this time because it does not assume linearity between data and handles better big dimensional spaces, although both performed well.
- As it can be expected, objects belonging to different classes but with similar textures are the ones that generate the most confusion to the trained classifier.
- Using a small amount of textons would under fit the problem, but using a very huge amount of them could cause over fitting.
- Computational power available must be carefully considered before choosing the number of textons and training images.
- To reduce computational complexity, preprocessing stage can be made taking into account that textures are patterns repeated over an image, hence cutting just enough to leave a recognizable pattern can reduce significantly computational power required.

References

[1] Y. Javed and M. Khan, "Image texture classification using textons", 2011 7th International Conference on Emerging Technologies, 2011.

[2] Ponce Group. 2005. Datasets for computer vision research. [online] Available at http://www-cvr.ai.uiuc.edu/ponce_grp/data/

[3] Digital image processing, R. González, P. Woods. 3rd Ed., Prentice Hall, 2008.

[4] A. Oliva, A. Torralba & P. Schyns. Hybrid images. Association for Computer Machinery, 2006.

[5] MathWorks. Classification Using Nearest Neighbors. [Online]. Available at: https://www.mathworks.com/help/stats/classification-using-nearest-neighbors.html#btap7k2

[6] MathWorks. TreeBagger. [Online]. Available at: http://www.mathworks.com/help/stats/treebagger.html

[7] Oshiro, T; Perez, P; Baranauskas, J. s.f. How many trees in a Random Forest? [online] Available at http://link.springer.com/chapter/10.1007/978-3-642-31537-4_13

[8] Leek, J. s.f. Random Forests. [online video] Available at https://www.coursera.org/learn/practical-machine-learning/lecture/XKsl6/random-forests

[9] Amatriain,X. 2015. What are the advantages of different classification algorithms? [Online]. Available at: https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

[10] Computer Vision: Algorithms and Applications. R. Szeliski. Springer, 2011.