Image Classification using Random Forests and Ferns

The paper is implemented on caltech-101 dataset having 101 object categories and also in more challenging dataset Caltech-256 dataset having large number of object categories in total 256 object categories. The significance of this Caltech-256 is its large inter-class variability, as well as a larger intra-class variability than in Caltech101. The model works quite well in this complex dataset also.

Traditionally image classification was considered as scene matching but this paper focuses on the on the object instance in order to learn its visual description. Instead of using the entire image to learn the model here this paper focuses on object instances to do so we automatically learn the region of interest of each training image. The intuition is that between subsets of the training image there will be high visual similarity of the object instance. In practice, regions of interest lead to an improvement in classification performance when the model is learnt from the ROI in each training image. Using the ROI detection and sliding window is a significant benefit. It increases performance from 3% to 5% depending on the degree of object pose variation within the datasets. But, for Caltech-101 adding the ROI optimization does not increase performance as much as in the case of Caltech-256. This is because Caltech-101 has less pose variation within the training images for a class

One more pros of this paper is that using both shape and appearance for spatial pyramid representation. And here classifier is also enabled to be selective for shape and appearance. As some class is better classified by shape and some class by appearance, so, it may increase the performance of the classifier.

For classification tasks Random Forest and Random Fern classifiers are used. The advantage of using Random Forest and Fern is that it is much faster than the traditional classifier such as SVM and also comparable performance to SVM. Here, Random ferns has somewhat less accuracy in the task but it is faster than even Random Forest. When using the random forest, the standard deviation is small (around 1%) independent of the number of trees used. This is in contrast to random ferns where if few ferns are used the variance is very large, however as the number of ferns is increased the random selection method does not cause large variations on the classifier performance. The main advantage of using ferns is that the training time increases linearly with the number of tests, while for random forests it increases exponentially with the depth. For both, random forests, and ferns the test time increases linearly with the number of trees/ferns. Increasing the depth increases performance however it also increases the memory required to store trees and ferns.

Performance for the random forests is 80.0% and with the M-SVM is 81.3% which is somewhat comparable. However, using random forests/ferns instead of a M-SVM, is far less computationally expensive – the classification time is reduced by a factor of 40.

Taking the tests at random usually results in a small loss of reliability but considerably reduces the learning time. In this paper different methods are used to inject randomness. Randomness can be injected at two points during training: in subsampling the training data so that each tree is grown using a different subset; and in selecting the node tests. When choosing a binary test randomness is injected into the training set per tree: one third of the training images per category are randomly selected and used to determine the node tests by the entropy criterion, and the remaining training images are used to estimate the posterior probabilities in the terminal nodes. This heuristic involves randomizing over both tests and training data.

As the performance increases when using more training data meaning that with less data, the training set is not large enough to estimate the full posterior. Since we have a limited number of training images per category to overcome this the paper suggests a way to synthetically generate additional training images. We generate 10 new images for each original training example obtaining 300 additional images per category. When using 5 training images and the extra data the performance increases from 19.7% to 29.1%, and it increases from 43.5% to 45.3% when using 30 training images.

Some of cons of this paper are:

1. The paper does not discuss the overfitting problem. It does not discuss if there is any chance of overfitting if we take large numbers of trees or we increase the depth of each tree too much. If there are not enough trees there may be a chance of overfitting if we increase the depth of the tree too much.
2. The fern classifier used can be very memory hungry.

RAM = ((2)^(fern size)) * sizeof(float) * NumFerns * NumClasses

So, from the above formula it is clear that if we increase the fern memory requirement increases. Hence, fern is much memory consumable.

[reference:ForestsAndFernsTalk.pdf].

In future there may be few improvements in the paper like ROI can be of any more flexible region other than a rectangular it may increase the performance of model. Also, more complicated algorithm classifiers such as CNN can be used so it will increase the performance because nowadays CNNs are used for image classification and recognition because of its high accuracy.