Kernel methods for Image Classification : Challenge

***Abstract*—In this paper, we study the problem of multi-class classification for images.**
Our work is divided into two main parts. First we focus on image processing as we implemented different type of features such as histograms and dense Sift descriptors.
Then, we focus on Support Vector Machine. Indeed, the SVM classifier has been tried with different error costs to manage imbalanced dataset with different kernels.

I. Kernel svm methods

A. SVM and CS-SVM

We start by considering the Kernel SVM problem seen in class (slide number 152).

$$\min \frac{1}{2} w.w^T + C \sum_i \xi_i \tag{1}$$

$$s.t. \quad y_i(w.\Phi(x_i) + b) \geq 1 - \xi_i \tag{2}$$

$$\xi_i \geq 0 \quad \forall i \tag{3}$$

The problem of SVM is sensitive to the class imbalanced (this problem occurs in one vs all configuration).
To rectify it, we implement a Cost-Sensitive SVM, as described in [3] and [2]. This method allows us to make different weights on the minority and majority class.
Let C^+ and C^- the misclassification cost of positive and negative class respectively.
We assume that the the positive class is the minority class and the negative class is the majority class.
The SVM primal problem becomes :

$$\min(\frac{1}{2} w.w^T + C^+ \sum_{i:y_i=+1} \xi_i + C^- \sum_{i:y_i=-1} \xi_i) \tag{4}$$

$$s.t. y_i(w.\Phi(x_i) + b) \geq 1 - \xi_i \tag{5}$$

$$\xi_i \geq 0 \; \forall i \tag{6}$$

And the dual problem becomes :

$$\max_{\alpha_i}(\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j)) \tag{7}$$

$$s.t. \sum_i y_i \alpha_i = 0, \; 0 \leq \alpha_i^+ \leq C^+, \; 0 \leq \alpha_i^- \leq C^- \; \forall i \tag{8}$$

where α_i^+ and α_i^- are the Lagrangian multipliers of positive and negative examples, respectively.

So, if $C^+ \geq C^-$, the impact of the class imbalanced can be reduced. Indeed, the minority class have the higher misclassification cost.

B. Choice of Kernel function

Based on the studies of [5] we mainly focused on two kernels :
- Laplacian Kernel : $K(x, y) = \exp(-\gamma ||x - y||_1)$.
- χ^2 Kernel : $K(x, y) = \sum_i (\frac{(x_i - y_i)^2}{x_i + y_i})$.

These kernels show a superior performance with histogram representations. Suppose a P-pixel bin in the histogram accounts for a single uniform color region in the image (with histogram x). A small change of color in this region can move the P pixels to a neighboring bin, resulting in a slightly different histogram x. If we assume that this neighbouring bin was empty in the original histogram x, the kernel values are :

$$\begin{aligned} K_{gaussian}(x, x') &= exp(-2\rho(P-0)^2) &&= exp(-2\rho P^2) \\ K_{Laplacian}(x, x') &= exp(-2\rho|P-0|) &&= exp(-2\rho P) \\ K_{\chi^2}(x, x') &= exp(-2\rho \frac{(P-0)^2}{P+0}) &&= exp(-2\rho P) \end{aligned}$$

And we find that the χ^2 and Laplacian kernels have a linear exponential decay while the Gaussian one has a quadratic exponential decay.

C. One vs All and One vs One

There are two main methods to handle multi-class classification, we used and experimented both.

1) One vs All: For each class, we train a SVM where the SVM must classify whether a sample is drawn from this class or from one of the "all" other classes.
Finally we compute the distance to the separating hyperplane for each sample and each SVM and retrieve the class whose this distance is maximised.

2) One vs One: In this method we train a SVM for each couple of class. Then the final label is decided with a vote decision.

II. Image features

We relied on [5], [4] and [6] to design the image features.

1) Color Histogram: A good technique to represent the images is the color histogram technique. A color histogram of an image represents the proportion of the different types of color and their number of pixels. We can represent a color in one vector corresponding to a position in the RGB (Red-Green-Blue) space.

2) Transition Histogram: This histogram counts the jumps in the signal. The transition histogram of the discrete signal $(x_1, , x_N)$ is an histogram H such that:

$$H_p = Card\{i \in [1, ., N-1], p \in [x_i, x_{i+1}[\}.$$

This histogram has two characteristics: translation invariant and scale invariant. For images, we need two transition histograms: an horizontal and a vertical.

3) Dense Sift: Let's introduce the Scale Invariant Feature Transform (SIFT). It detects keypoints corresponding to the relevant image regions and extracts descriptors which are discriminative (scale and direction corresponding to the gradient and its magnitude).
The dense Sift algorithm is computing SIFT features on a regular grid of the image. Then, each SIFT is described as a vector and mapped to a visual word. Affecting the feature to a visual word is made through a process of clustering. This clustering depends on two parameters which are the number of orientations (or angles), r, and the width n of the $n \times n$ array of orientation histograms. Then, the image is split in regions (3×3 regions in our case) and the frequency of the visual words in every region is recorded in an histogram. Every one of those histograms is normalized. Finally, the image is represented by a vector resulting from the concatenation of those histograms.
In our case, the best results are achieved with a 4×4 array of histograms with 8 orientation bins in each, for each of the 9 regions. Thus, we get an histogram of size 129 for every region region and the image is finally represented by a vector of size $128 \times 9 = 1152$.
The process is represented on the Figure 1.

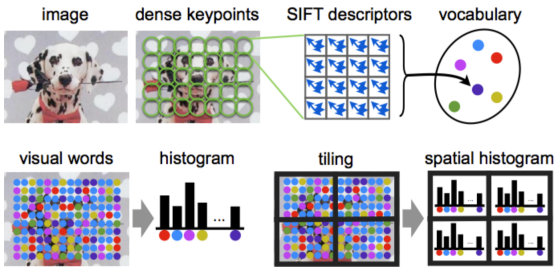

Fig. 1. Constructing dense SIFT descriptors

III. Implementation and Results

A. Implementation

We implemented the classifiers and the image processing in python, using a lot the jupyter notebook for the experiments. We used the package CVXOPT to solve the mathematical problems of SVM and CS-SVM.
For the implementation of image processing (in particular for the Dense Sift features) we relied on two image courses we followed in the MVA and on this article [6] and we used only the functions available in Numpy and scipy.spatial.distance and scipy.(and so we didn't use image processing libraries or functions).

B. Experiments and Results

We experiment several things in this challenge. The first one was to use directly the features we get with a SVM and a Gaussian kernel, but the results were very poor, what is not surprising.
Then, we decided to look for features more appropriate for image classification, scale and rotation invariant features and we experiment the histogram features (color and transition) It was pretty easy to implement and we use it to make our first submission. We found by cross-validation that the Laplacian Kernel with $\gamma = 1/d$ with d the dimension of the features, was the most accurate (between Gaussian, Chi-2 and Laplacian). We use the one vs all method with a CS-SVM with regularization costs.
Our other and highest submission was with the Dense Sift features, and we used again a Laplacian Kernel with $\gamma = 1/d$ but in one vs one method.

TABLE I
Final results

Data+Classifiers	Public Score	Private Score
Color+Transition histogram One-vs-all laplacian kernel	0.30700	0.33700
Dense sift One vs one laplacian kernel	0.38600	0.35200

Dense sift features allow to learn better to classify images but the histogram features are interesting in the fact that it seems to be less sensitive to overfitting, since the private score is higher than the public score.
However we join to this report the code for Dense sift which is our highest submission.

IV. Conclusion

In conclusion this project was pretty interesting because could not use Machine Learning libraries and so we really had to become familiar with how to construct an SVM and kernels. However, the disadvantage is that implmenting every algorithm requires a consequent time. Thus there were several things we have planned to do, like stacking the two models (features histograms and dense sift with SVM) and using a regression to learn how combine the pros of these two models. We also thought to use a Kernel histogram like [1] this article, but we ran out of time.

References

[1] Annalisa Barla, Francesca Odone, and Alessandro Verri. Histogram intersection kernel for image classification. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III–513. IEEE, 2003.
[2] Rukshan Batuwita and Vasile Palade. Class imbalance learning methods for support vector machines. *Imbalanced learning: Foundations, algorithms, and applications*, 83, 2013.
[3] Peng Cao, Dazhe Zhao, and Osmar Zaiane. An optimized cost-sensitive svm for imbalanced data learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 280–292. Springer, 2013.
[4] Olivier Chapelle. Support vector machines et classification d'images. 1998.
[5] Olivier Chapelle, Patrick Haffner, and Vladimir N Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.
[6] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.