Assignment 2: Image classification

Rémi Lespinet

remi.lespinet@ens-paris-saclay.fr

I Training and testing an Image Classifier

I.A Data preparation and feature extraction

QA1: Why is the spatial tiling used in the histogram image representation?

Using spatial tiling adds spatial information about the objects on the scene, doing so implies to compute K-means 4 times (one for each quadrant), hence the computed words are 4 times larger. As a consequence, we will only compare features descriptors that are in the same quadrant (when we detect a feature, we will compute its SIFT descriptor, find the bin associated to that descriptor in the quadrant it has been detected and account for that bin in the part of the histogram corresponding to that quadrant).

If the number of training images is large enough, this is a good way to add spatial information without impacting the time complexity of the classifier.

I.B Train a classifier for images containing aeroplanes

QB1: Show the ranked training images in your report

The ranked images are represented in figure 1

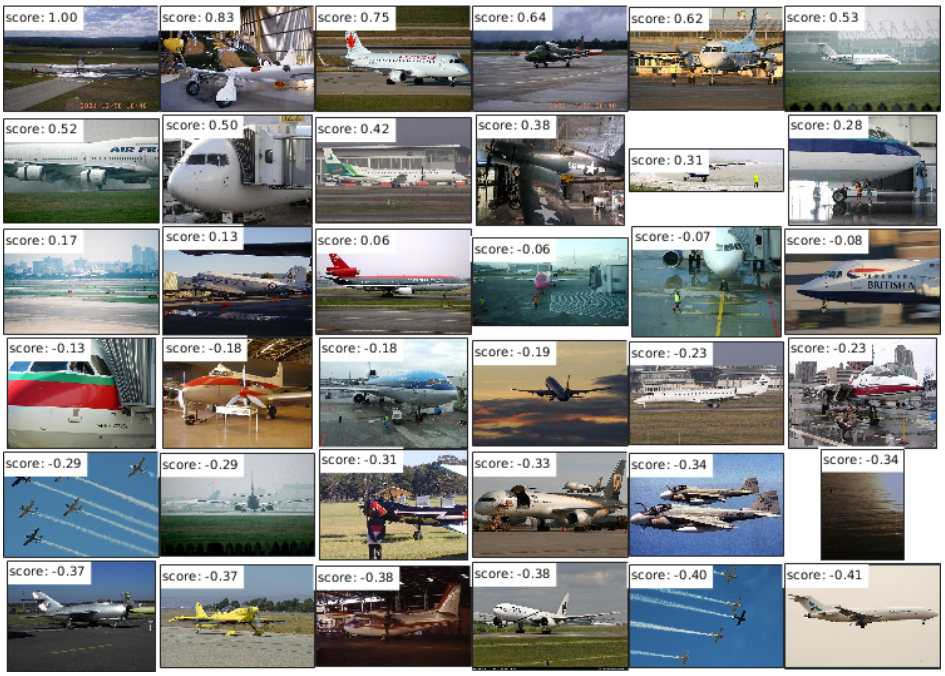
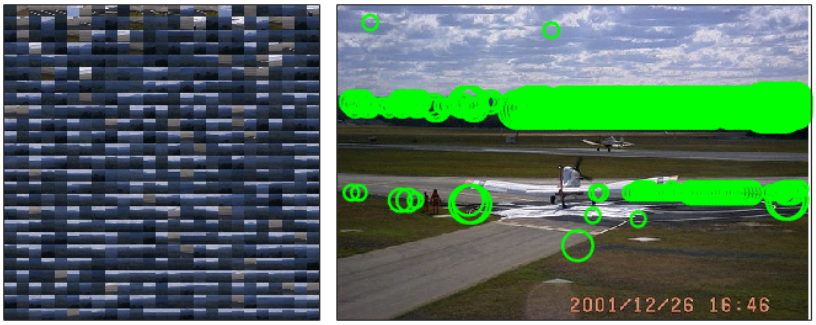


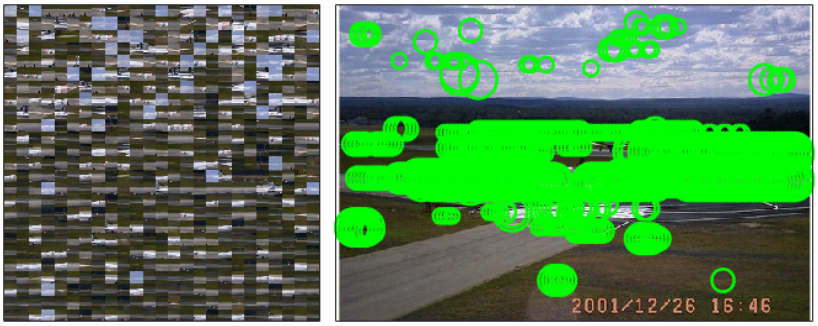
Figure 1: Representation of the top 36 ranked training images classified by a linear SVM classifier on a training database composed of aeroplane images (positive) and background images (negative)

QB2: In your report, show relevant patches for the three most relevant visual words (in three separate figures) for the top ranked training image. Are the most relevant visual words on the airplane or also appear on background?

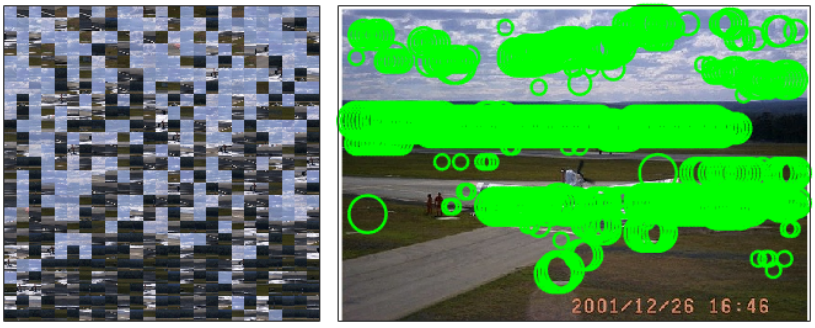
The three most relevant visual words for the top ranked training image are represented in figure 2. We can see in this figure that these visual words also appears on the background, in particular, the rank 1 visual word appears almost exclusively on the background of the image. An explanation for this is that images containing airplanes often contains sky, airstrip, etc which are used by the algorithm to recognize airplanes.



(a) Rank 1 visual word (index 201) corresponding to 797 feature descriptors, $word\_score = weight \cdot count = 0.21$



(b) Rank 2 visual word (index 455) corresponding to 787 feature descriptors, $word\_score = weight \cdot count = 0.15$



(c) Rank 3 visual word (index 336) corresponding to 871 feature descriptors, $word\_score = weight \cdot count = 0.14$

Figure 2: Representation of the patches for the 3 most relevant visual words for the top ranked image trained on the airplane dataset. count number of descriptors associated

I.C Classify the test images and assess the performance

QC1: Why is the bias term not needed for the image ranking?

The bias term is not needed because we do not need the exact score, we only need to rank the images based on their score, and the ranking goes unchanged if we add an arbitrary constant.

I.D Learn a classifier for the other classes and assess its performance

QD1: In your report, show the top ranked images, precision-recall curves and APs for the test data of all the three classes (aeroplanes, motorbikes, and persons). Does the AP performance for the different classes match your expectations based on the variation of the class images?

The figure 3, (resp. 4 and 5) represents the 36 best ranked test images according to the classifier trained on the train data using a dataset composed of airplanes (resp. motorbikes and persons) images and background images.

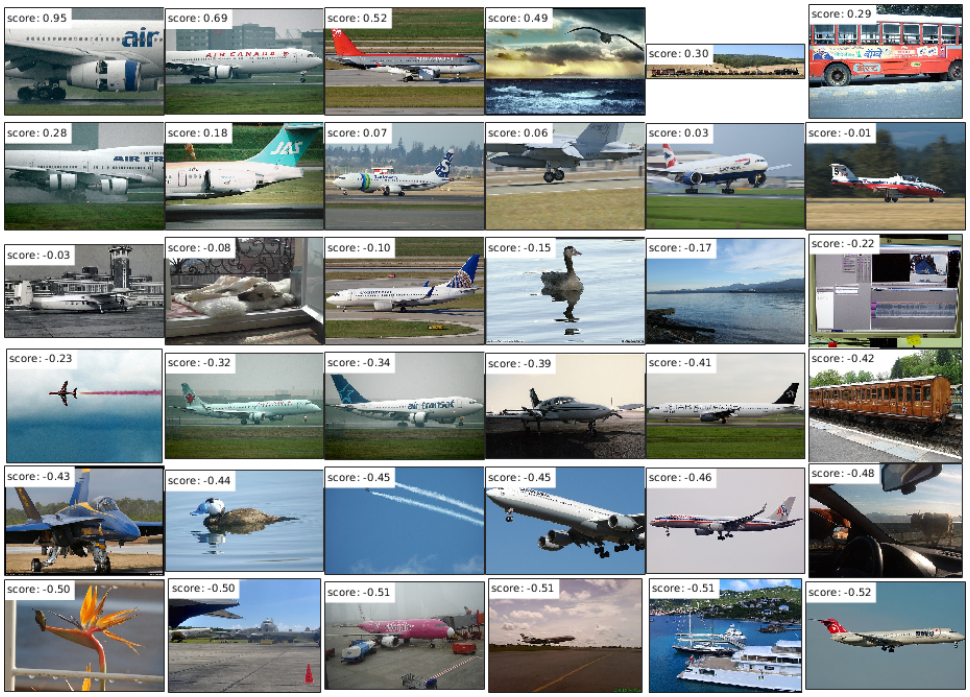


Figure 3: top 36 test images classified as airplane by the classifier using a database composed of airplane and background images

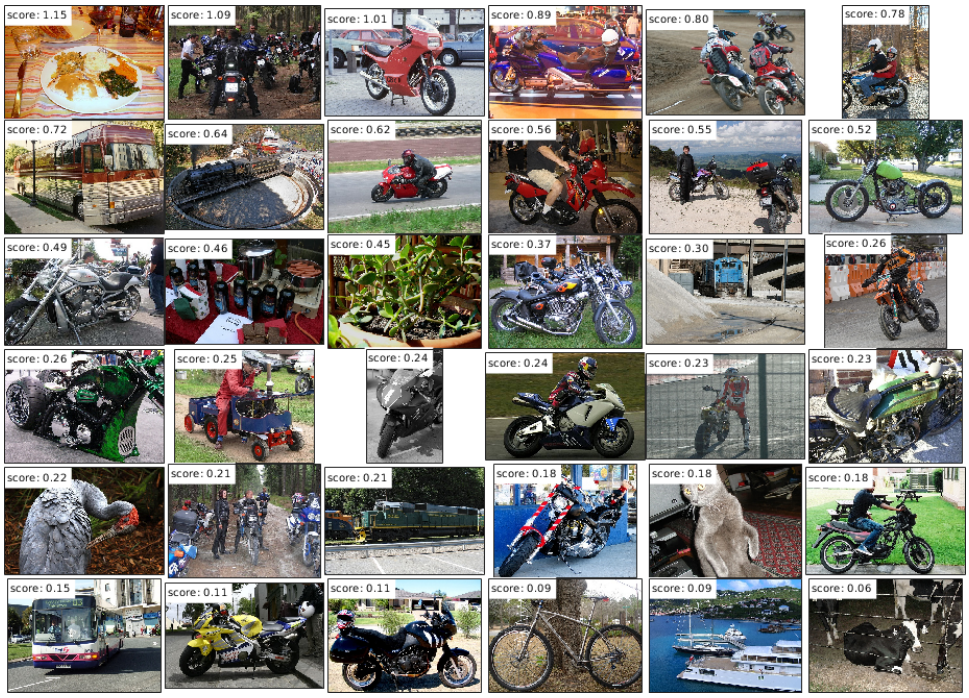


Figure 4: top 36 test images classified as motorbikes by the classifier using a database composed of motorbikes and background images

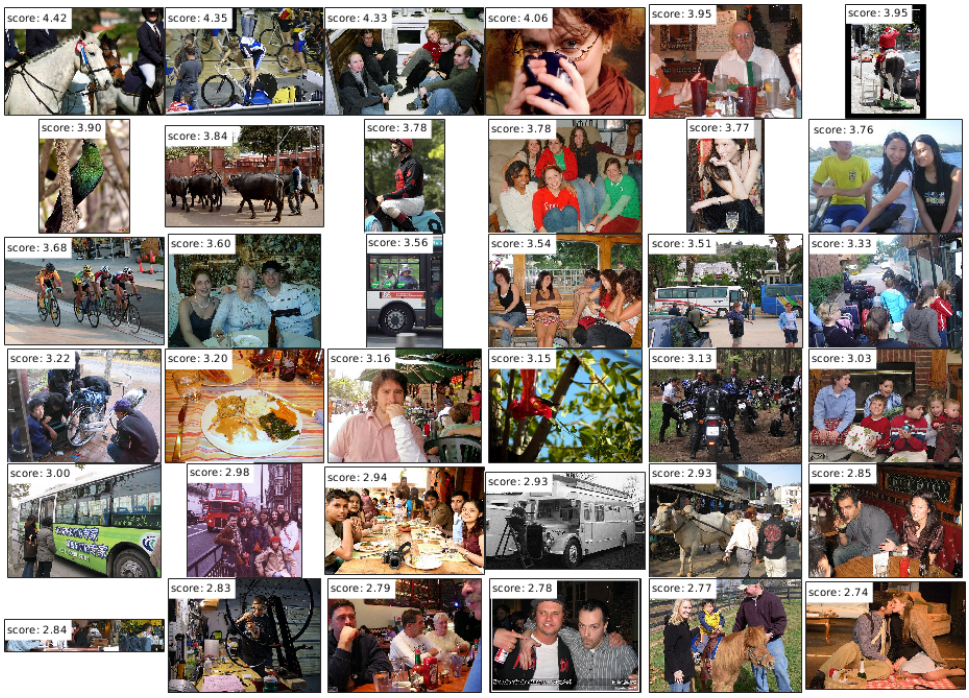
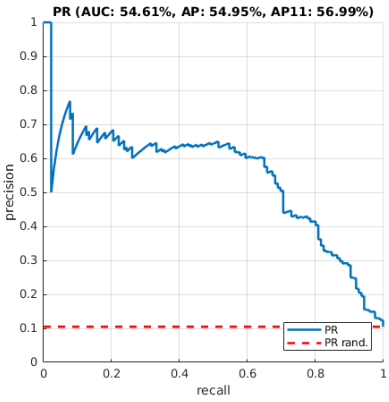
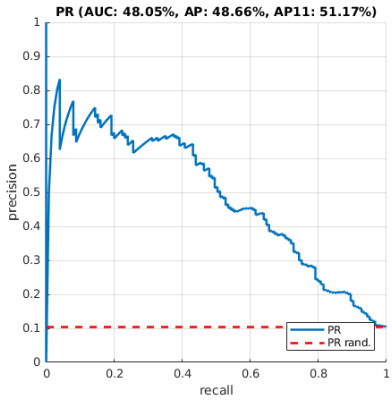


Figure 5: top 36 test images classified as persons by the classifier using a database composed of persons and background images

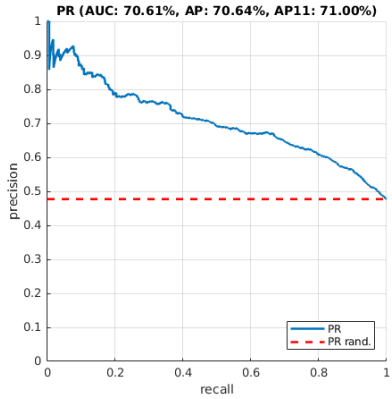
The precision-recall curves for each of the three classes (airplane, motorbike and person) is represented in the figures 6a, 6b and 6c respectively.



(a) precision-recall curve for the airplane classe



(b) precision-recall curve for the motorbike classe



(c) precision-recall curve for the person classe

Figure 6: Representation of the precision-recall curve for each of the three classes

The AP performance is presented in the table 1

Classe	AP
Airplane	54.95%
Motorbike	48.66%
Person	70.64%

Table 1: AP for each of the 3 classes

For the three classes, the number of train images is the number of background train images (1019) plus the number of images in the train data for this class, similarly, the number of test images is the number of background test images (1077) plus the number of images in the test data for this class. Table 2 shows the composition of the datasets used for this question on each of the three classes. (The test ratio corresponds to the random line present in the precition-recall curves)

The performances are correct given the fact that the model is not very elaborate yet, for the aeroplane dataset, we reach more than 50% average precision, which is decent in view of the low fraction number of images in the test dataset that actually represent an aeroplane (10%).

classes	positive train images	negative train images	train ratio	positive test images	negative test images	test ratio
aeroplane	112	1019	9.90 %	126	1077	10.47 %
motorbike	120	1019	10.54 %	125	1077	10.40 %
person	1025	1019	50.15 %	983	1077	47.72 %

Table 2: Composition of the train dataset and the test dataset for each of the 3 classes

QD2: For the motorbike class, give the rank of the first false positive image. What point on the precision-recall curve corresponds to this first false positive image? Give in your report the value of precision and recall for that point on the precision-recall curve.

For the motorbike, the best ranked image is a false positive (it's represented in figure 7)

Figure 7: Best ranked image for the motorbike class (false positive)

It corresponds to the situation where we only select the top ranked image. This image is wrong, so the recall is 0 and the precision is also 0. In matlab, if we extract the first 10 points in the precision-recall curve we obtain

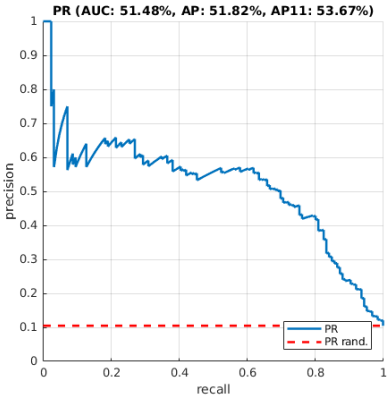
recall	0	0	0.008	0.016	0.024	0.032	0.04	0.040	0.040	0.048	...
precision	1	1.0e-10	0.500	0.667	0.750	0.800	0.833	0.714	0.625	0.667	...

The corresponding point is the second entry $(0, 1.0e-10)$. The first point is a convention when no sample are predicted (the number of positive images returned as well as the number of return images is 0, so the precision is 0 over 0 which is 1 by convention, and the number of positive images is 120 for the motorbike, so the recall is 0 over 120 which is 0). We can see that point in the figure 6b.

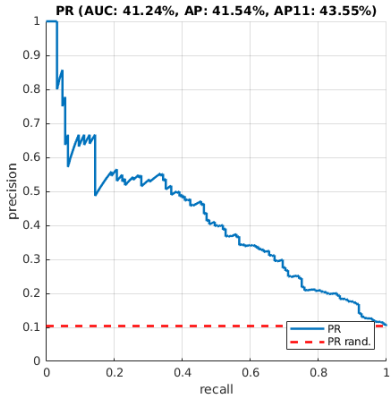
I.E Vary the image representation

QE1: Include in your report precision recall-curves and APs, and compare the test performance to the spatially tiled representation in stage D. How is the performance changing? Why?

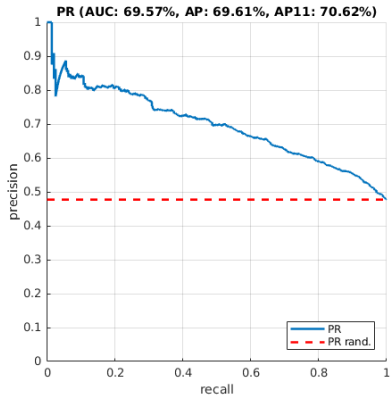
The precision-recall curves for all datasets are presented in figure 8. The comparison of the average precision with and without spatial tiling is presented in the table 3. We see that the performance is decreasing, this stems from the fact that we have lost the spatial information provided by the tiling. (For the next questions, I have reactivated the spatial tiling)



(a) precision-recall curve for the airplane classe



(b) precision-recall curve for the motorbike classe



(c) precision-recall curve for the person classe

Figure 8: Precision-recall curve for each of the three classes when there's no spatial tiling

Classe	AP (tiling)	AP (no tiling)
Motorbike	**48.66%**	41.54%
Airplane	**54.95%**	51.82%
Person	**70.64%**	69.61%

Table 3: Effect of tiling on average precision for each of the 3 classes

QE2: Modify exercise1.m to use L1 normalization and no normalization and measure the performance change

The table 4 presents the different average precision obtained for different type of normalizations ($\mathcal{L}^2$, $\mathcal{L}^1$ and no normalization) for each of the three classes (Airplane, Motorbike and Person).

Classe	$\mathcal{L}^1$ normalization	$\mathcal{L}^2$ normalization	no normalization
Motorbike	26.00%	48.66%	**48.73%**
Aeroplane	51.68%	54.95%	**62.45%**
Person	56.54%	**70.64%**	67.20%

Table 4: Effect of the histogram normalization on average precision for each of the 3 classes

QE3: What can you say about the self-similarity, K(h,h), of a BoVW histogram h that is L2 normalized?

We have

$$K(h,h) = \sum_{i=1}^{d} h_i^2 = \|h\|_2^2$$

Hence if h is $\mathcal{L}^2$ normalized, $K(h,h) = 1$

If h and h' are $\mathcal{L}^2$ normalized, Cauchy-Schwarz inequality gives us

$$K(h,h') \leq K(h,h)K(h',h') = 1$$

This means that $K(h,h') \leq 1$ with equality when $h = h'$.

This is not true if we normalize with $\mathcal{L}^1$, it is possible to have $K(h,h') > K(h,h)$ with h and h' being different. For example in dimension 2, if we take $h = (0.75, 0.25)$ and $h' = (1,0)$, we have $\|h\|_1 = 1$, $\|h'\|_1 = 1$, $K(h,h') = 0.75$, but $K(h,h) = 0.625$

QE4: Do you see a relation between the classification performance and L2 normalization?

We can see in the table that the $\mathcal{L}^2$ gives better results than the $\mathcal{L}^1$ norm for all the 3 classes, the results on this training set if we don't normalize are slighly better in the *Motorbike* and *Aeroplane* cases.

I.F Vary the classifier

QF1: Based on the rule of thumb introduced above, how should the BoVW histograms h and h' be normalized? Should you apply this normalization before or after taking the square root?

Explicitely computing the mapping means taking the square root of the histogram values $\tilde{h}_i = \sqrt{h_i}$. If we feed the linear SVM classifier with $\tilde{h}$, from the proposed rule of thumb we want $\tilde{h}$ to be $\mathcal{L}^2$ normalized.

$$h_i \xrightarrow{\sqrt{\cdot}} \sqrt{h_i} \xrightarrow{\mathcal{L}^2 normalize} \frac{\sqrt{h_i}}{\sqrt{\sum_{i=1}^{d} \sqrt{h_i}^2}} = \frac{\sqrt{h_i}}{\sqrt{\sum_{i=1}^{d} |h_i|}}$$

This is equivalent to

$$h_i \xrightarrow{\mathcal{L}^1 normalize} \frac{h_i}{\sum_{i=1}^{d} |h_i|} \xrightarrow{\sqrt{\cdot}} \frac{\sqrt{h_i}}{\sqrt{\sum_{i=1}^{d} |h_i|}}$$

So we can either normalize h in $\mathcal{L}^1$ before applying the square root or take the square root and then normalize in $\mathcal{L}^2$

QF2: Why is this procedure equivalent to using the Hellinger kernel in the SVM classifier?

Given two histograms h and h', the given procedure will feed the linear SVM with the modified input histograms $\hat{h}$ and $\hat{h'}$, and we have

$$K_{Linear}(\hat{h}, \hat{h'}) = \sum_{i=1}^{d} \hat{h}_i \hat{h'}_i = \sum_{i=1}^{d} \sqrt{h_i}\sqrt{h'_i} = K_{Hellinger}(h, h')$$

Hence this procedure is equivalent to using the Hellinger kernel in the SVM classifier.

QF3: Why is it an advantage to keep the classifier linear, rather than using a non-linear kernel?

Using a linear method on the modified histograms allows to compute the square roots only once, rather than computing it each time the kernel is evaluated, which increases performance.

QF4: Try the other histogram normalization options and check that your choice yields optimal performance. Summarize your finding in the report (include only mAP results, no need to include the full precision-recall curves).

The comparison of average precisions for different normalizations ($\mathcal{L}^1$, $\mathcal{L}^2$ and no normalization) of the histogram, using a Hellinger kernel classifier are presented in the table 5. We see that, as expected, the overall best results are obtained using $\mathcal{L}^1$ normalization.

class	$\mathcal{L}^1$	$\mathcal{L}^2$	none
Motorbike	**63.25**	52.81	55.64
Aeroplane	**70.72**	63.94	65.87
Person	**77.39**	71.45	69.71

Table 5: Effect of the histogram normalization on average precision for each of the 3 classes with the Hellinger kernel

We also see that the Hellinger using the $\mathcal{L}^1$ norm yields better results than the Linear kernel using the $\mathcal{L}^2$ norm for each of the 3 classes. We obtain about 15% increase in average precision for the motorbike and the aeroplane, and about 7% for the person class, which is a significative improvement. (The table 8 present these results as well as the results for different encoding, namely VLAD and FV that we try later on)

I.G Vary the number of training images

QG1: Report and compare performance you get with the linear kernel and with the Hellinger kernel for the different classes and proportions of training images (10%, 50% and 100%). You don't have to report the precision-recall curves, just APs are sufficient. Plot three curves (one curve for each class) into one figure. Produce two figures, one for the linear kernel and one for the Hellinger kernel. Make sure to properly label axis (use functions xlabel and ylabel), show each curve in a different color, and have a legend (function legend) in each figure. Show the two figures in your report.

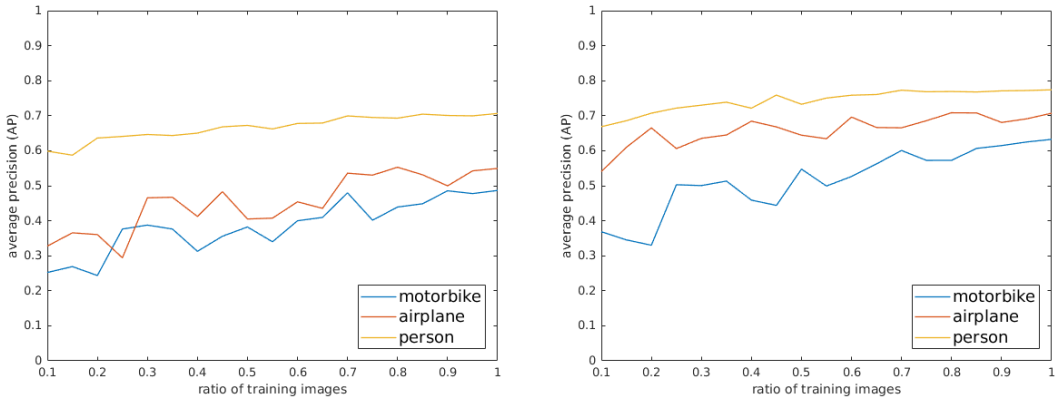
The table 6 present a comparison of the average precision obtained for different values of the ratio of training images taken (10%, 50%, 100%) using a linear and a Hellinger kernel.

The average precision increases with the number of training images as we can expect, we see that the Hellinger kernel gives overall better results, and seems to work better when there the number of training images is low.

	Linear kernel			Hellinger kernel		
class	$r = 0.1$	$r = 0.5$	$r = 1$	$r = 0.1$	$r = 0.5$	$r = 1$
motorbike	25.23%	38.23%	**48.66%**	36.85%	54.74%	**63.25%**
aeroplane	32.77%	40.50%	**54.95%**	54.05%	64.44%	**70.72%**
person	59.89%	67.23%	**70.64%**	66.86%	73.26%	**77.39%**

Table 6: Comparison of the average precision for different values of proportions of training images (r) for each of the 3 classes for the linear and Hellinger kernel

The figure 9 presents the variation of the average precision as a function of the percentage of training images taken for the training part for each of the three classes for both the linear (figure 9a) and the Hellinger kernel (figure 9b).



(a) Linear kernel

(b) Hellinger kernel

Figure 9: Average precision as a function of the percentage of the training images taken for training for each of the three classes (20 points are represented)

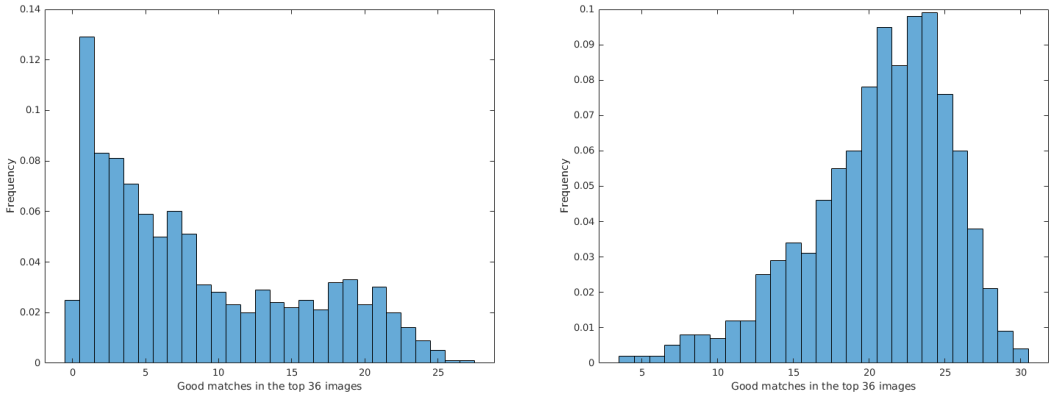
QG2: By analyzing the two figures, do you think the performance has 'saturated' if all the training images are used, or would adding more training images give an improvement?

We see in the curves that the average precision increases with the number of training, especially when the number of sample is low (between 10% and 70% of the training images taken). When approaching a ratio of 1, the curve still seems to increase even if it's very slight, hence, I think that adding images could make our average precision gain fractions of percents, but it will not make a huge improvement (especially for the *person* class which already has a large amount of training images).

II Training an Image Classifier for Retrieval using Internet image search

QP2.1: For the horse class, report the precision at rank-36 for 5 and 10 training images. Show the training images you used. Did the performance of the classifier improve when 10 images were used?

I downloaded 28 images and trained the classifier 1000 times over 5 random images from this set of 28 images, I obtained the histogram presented on figure 10a. I did the same for 10 over 28 images, the histogram is presented on figure 10b. The table 7 contains statistics for both the selection of 5 training images and 10 training images.



(a) 5 images selected

(b) 10 images selected

Figure 10: Histograms representing the number of good classified images in the top 36 scoring images for sets of training images containing 5 and 10 images (on 28 images downloaded)

selected images (over 28)	AP_{mean}	AP_{max}	AP_{std}	Top_36_{mean}	Top_36_{max}	Top_36_{std}
5	17.04%	45.68%	8.03%	8.54	27	6.95
10	36.19%	54.30%	7.50%	20.68	30	4.68

Table 7: Statistics obtained by the procedure described

The 5 training images that produced the highest number of good classified images in the top 36 are represented in the figure 11 and the top 36 images returned by the classifier are shown in figure 12 . Similarly, the set of 10 training images that produced the best result is represented in the figure 13, and the top 36 images in the figure 14,

The performance of the classifier is clearly improved when 10 images are used instead of 5, the histogram is shifted right, the mean precision in the top36 increases from about 8 (22%) to 21 over 36 (58%) which is very significiant.

Figure 11: Set of 5 training images of horses downloaded on the internet used to train the classifier

Figure 12: top 36 images classified as horses by the classifier trained on a database composed of 5 images downloaded from the internet (represented in fig 11) 27/36 are correctly classified

Figure 13: Set of 10 training images of horses downloaded on the internet used to train the classifier

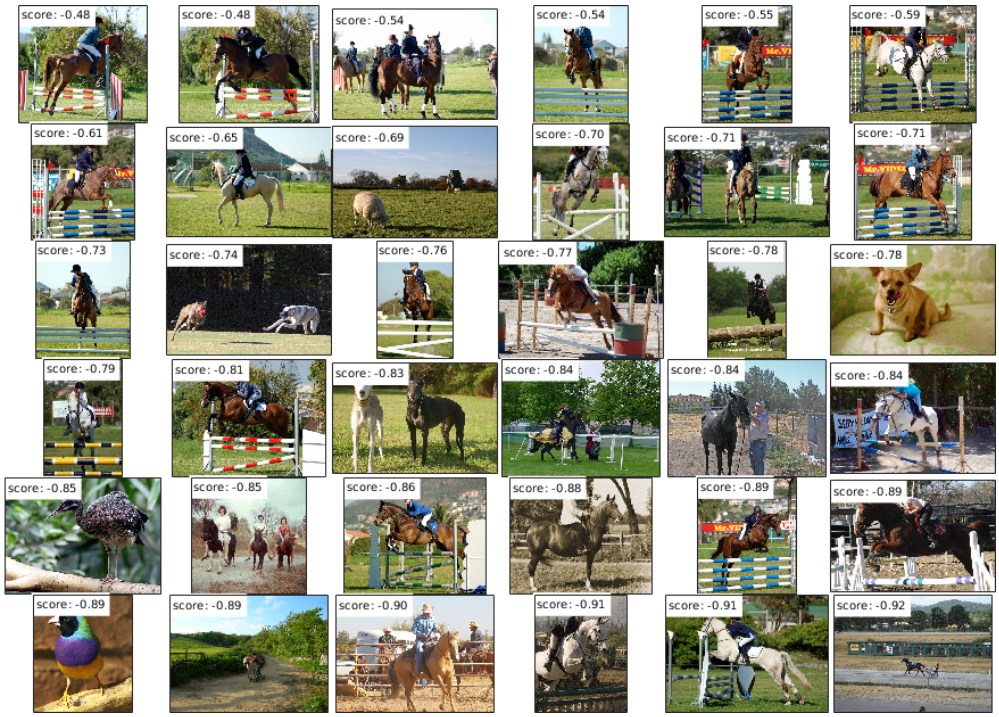


Figure 14: top 36 images classified as horses by the classifier trained on a database composed of 10 images downloaded from the internet (represented in fig 13). 30/36 are correctly classified

QP2.2: What is the best performance (measured by precision at rank-36) you were able to achieve for the horse and the car class? How many training images did you use? For each of the two classes, show examples of your training images, show the top ranked 36 images, and report the precision at rank-36. Compare the difficulty of of retrieving horses and cars

The best performance I've reached for the horse class is 32 achieved with a set of 13 images, e.g 88.89% precision at rank 36 (see figure 15 for the training images and figure 16 for the top 36 ranked images)

For the car class, I reached 36 with 7 images (100% precision at rank 36). The figure 17 shows the training images and the figure 18 shows the top 36 ranked images from the test dataset.

Figure 15: Set of 13 training images of horses used to reach 32 good classified images in the top 36 ranked images (best I could achieve)

Figure 16: top 36 images classified as horses by the classifier trained on a database composed of 7 images downloaded from the internet and background images (false positive)

Figure 17: Set of 7 training images of cars used to reach 36 good classified images in the top rank-36

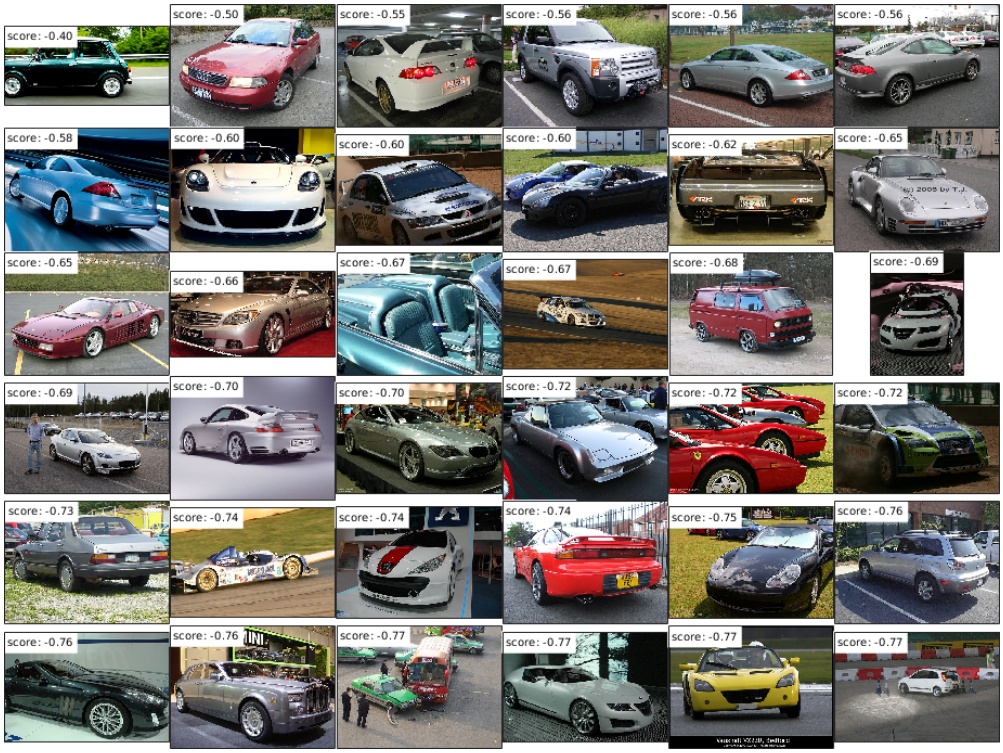


Figure 18: top 36 images classified as cars by the classifier trained on a database composed of 7 images downloaded from the internet and background images (false positive)

We can notice that recognizing horses is a lot more difficult than recognizing cars, for the car, reaching 100% precision at rank 36 was actually fairly easy. This can be explained by the fact that there is a lot more recognizable detection points on a car than on a horse (due to the sharp geometry of the object, the different material used, ...)

III Advanced Encoding Methods

III.A First order methods

QH1: Compare the dimension of VLAD and BoVW vectors for a given value of K. What should be the relation of the K in VLAD to the K in BoVW in order to obtain descriptors of the same dimension? You can ignore tiling.

The dimension of the VLAD vector is KN, K being the size of the vocabulary and N the size of a SIFT vector. The dimension of a BoVW vector is K, with K the size of the vocabulary.

For a BoVW vector to have the same dimension as a VLAD vector which has a vocabulary of size K, we need a vocabulary that is N times bigger than the one used for the VLAD vector e.g a vocabulary of size K' with $K' = KN$.

QH2: Replace the encoding used in exercise1 with the VLAD encoding, and repeat the classification experiments for the three classes of Part I (Both linear and Hellinger kernel). How do the results compare to the BoVW encoding? Report AP results in a table. No need to report all precision-recall curves.

By looking at the dimensions in MATLAB, we see that the vectors are of size 10240, versus $2048 = 4 \cdot 512$ (512 per tile) for the BoVW vectors, assuming there's also tiling, this means that $K = 20$, $N = 128$.

We need to be extra-careful, because a VLAD vector can have negative components. To use Hellinger's kernel, we use signed square-rooting (for each component we take the square root the absolute value and multiply by its sign) this is appropriate because in the formula below K is a measurement of similarity, and doing this operation ensures that if two multiplied components have the same sign (which means that they are similar) K is increased, whereas if they have opposite sign (which means they are not similar), K is decreased.

$$K(h, h') = \sum_{i=1}^{d} \sqrt{h_i h'_i}$$

The results of the average precision for each of the three classes using the VLAD encoding are represented in the table 8. We can see that the classifier using VLAD encoding outperforms the one using BoVW encoding for the two kernel used. The increase in average precision is particularly good for the motorbike and aeroplane cases (about 20% increase for the linear kernel).

	Linear kernel			Hellinger kernel		
classes	BoVW	VLAD	FV	BoVW	VLAD	FV
motorbike	48.66%	68.82%	**72.67%**	63.25%	75.42%	**81.14%**
aeroplane	54.95%	**74.62%**	70.64%	70.72%	75.56%	**78.13%**
person	70.64%	75.54%	**77.44%**	77.39%	78.86%	**82.11%**

Table 8: Comparison of BoVW and VLAD encoding method for each of the 3 classes with the linear and Hellinger kernel

III.B Second order methods

QI1: Replace the encoding used in exercise1 with the FV encoding, and repeat the classification experiments for the three classes of Part I. Report the results in the same table as QH2 so that you can see the performance of the three encoding methods side by side.

The results are presented in the previous table (table 8) for the linear and the Hellinger kernel classifier)

QI2: What are the advantages or disadvantages of FV compared to VLAD in terms of computation time and storage/memory footprint - especially for a large number (hundreds of millions) of images.

We have seen that a VLAD vector has KD components and that A Fisher Vector has $2KD$ component in the case where only the variance of each component is computed. This is actually how it's done in our case (only the variance of each component is computed), if we look at the amount of spaces taken on disk to store a fisher vector and a vlad vector, we see that the fisher vector takes about twice the size of the vlad vector (for example *background_train_vlad.mat* takes 38132 Ko and *background_train_fv.mat* takes 74384 Ko).

If we would want to compute the covariance matrix entirely, this would be $KD + KD^2$. In our case, $D = 128$, if we take $K = 20$, this number is already 330240 components per image, multiplied by 100000000 images this is about $132TB$ if we count 4 bytes per components.

If we only store the diagonal, we only have 5120 components per image, which gives $2TB$ of data. VLAD vectors would have given half this amount, which is $1TB$ of data.

In the processing part, the VLAD method need to compute the SIFT descriptors for each image and run K-means algorithm. For each image and for each descriptor, it then needs to compute the sum of residual. We then learn a classifier (SVM).

At run time, we need to find the centroid associated to each descriptor, using a kd-tree, compute the residual with respect to this centroid and account for it to the VLAD vector, once we have done that for all descriptors, we can predict using the classifier.

the FV method need to compute the SIFT descriptor for each image, and run GMM algorithm. For each image and for each descriptor, it then computes the mean of the gaussian kernel with the maximum probability (hopefully we can do that efficiently using an accelerating structure (some kind of modified kd-tree)). We then need to compute the residual and the variance (which we can do in one pass). once we have done that for each descriptors, and for each image, we learn a classifier.

At run time, for each descriptor, we need to find the mean of the gaussian kernel associated to the maximum probability, compute the residual, and the variance (we can do that online), we can then predict using the classifier.

We can notice that the GMM clustering is slower than K-means, assuming that there is an efficient structure for finding the mean of the gaussian kernel associated to the maximum probability, this part is slower by only a small amount, and for the FV, we only need to compute the variance, which should increase the time by a small amount at run time (but this cost can be important in preprocessing as it is multiplied by the number of training images).