Reconnaissance des formes pour l'analyse et l'interprétation d'images

Description d'images par SIFT, Bag of Word et classification d'images par SVM.

Loüet Joseph & Karmim Yannis
Spécialité **DAC**

Table des matières

1 Descripteur SIFT.

Dans la première partie de ces travaux pratiques en reconnaissance des formes pour l'analyse et l'interprétation d'images on s'intéresse à un algorithme élémentaire pour le domaine de la **Computer Vision**. L'algorithme **Scale-invariant feature transform (SIFT)** construit des descripteurs locaux d'une image sous forme de vecteur numérique, ces descripteurs vont nous permettre d'en extraire des features qui vont être très utile pour des tâches tels que la classification d'images, la reconnaissance de forme ou encore la recherche d'image.

1.1 Calcul du gradient d'une image.

Dans une image on part du principe que toutes les zones ne sont pas intéressante pour en extraire des informations, par exemple des fonds de couleur uniforme n'ont pas grand intérêt pour extraire la sémantique et les caractéristiques d'une image. Il est donc essentiel pour des algorithmes en vision de détecter automatiquement ces zones d'intérêts.
Le calcul du gradient est une manière efficace pour trouver des fortes variations dans une image. Ce calcul va être essentiel pour la construction de nos descripteurs.
Soit $G(x, y)$ le gradient d'une image au pixel (x, y) :

$$G(x,y) = \left[\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right]^\top = [I_x I_y]^\top$$

On fait une approximation des dérivées partielles par des différences finies afin de faciliter les calculs. On utilise les masques de Sobel $3x3$ suivant :

$$M_x = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; M_y = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

À partir de I_x et I_y on calcul la norme G_n et l'orientation G_o du gradient.

1. Montrons que les masques M_x et M_y sont séparables.
$(y_1, y_2, y_3).(x_1, x_2, x_3)^\top = M_x$
$(x_1, x_2, x_3).(y_1, y_2, y_3)^\top = M_y$

En résolvant ce système on trouve :
$M_x = \frac{1}{4}((1, 2, 1).(-1, 0, 1))$
$M_y = \frac{1}{4}((-1, 0, 1).(1, 2, 1))$
$h_x = \frac{1}{4}(-1, 0, 1)$, $h_y = \frac{1}{4}(1, 2, 1)$

2. L'intérêt de séparer le filtre de convolution est qu'il est plus rapide de multiplier par des vecteurs que par une matrice puisque les multiplications sont associatives.

1.2 Calcul de la représentation SIFT d'un patch.

3. Avant de calculer nos SIFT, on peut appliquer un filtre gaussien à nos patchs. Cela a pour but de lisser notre image afin que des détails trop petits qui sont susceptible de bruiter inutilement nos données soient enlever. On rendra nos descripteurs SIFT plus robuste.

4. La discrétisation des orientations permet à nos déscripteurs d'être robuste à la rotation. Pour une même image mais à laquelle on applique une rotation qui reste dans le même pas de discrétisation, notre descripteur SIFT sera invariant. Le choix du nombre de pas de discrétisation est un choix arbitraire et dépend du contexte dans le quel on veut appliquer notre descripteur SIFT.

5. La première étape du post-processing qui consiste à renvoyer un descripteur nul si sa norme 2 est inférieur à 0.5 permet de ne pas considérer les descripteurs qui ont un signal trop faible afin d'éviter d'en calculer pour des zones inintéressantes. La normalisation et le seuil à 0.2 permet ensuite de rendre le descripteur robuste à la luminosité en imposant une valeur maximale.

6. Le principe du SIFT est une façon raisonnable de faire de l'analyse d'image, car on construit un vecteur caractéristique d'une image qui est robuste aux rotations et à la luminosité. Le SIFT nous permet de repérer les régions pertinentes de l'image en détectant les variations à l'aide des Gradients I_x, I_y et on travail avec des données beaucoup plus petite, en l'occurrence des vecteurs deux fois moins volumineux que la taille du patch.

7. Pour interpréter notre sift on doit imaginer notre patch (image centrale)

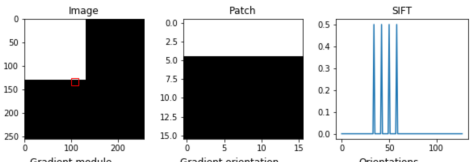


FIGURE 1 – Exemple de descripteur sift calculé sur un patch d'une image.

découpé en 16 sous régions, les 4 premières régions sont entièrement blanche, donc notre gradient est nul puisque l'image ne varie pas. Ensuite on peut supposer que nos 4 sous régions suivantes se trouvent dans la zone de bordure entre la partie blanche et noire, on reçoit 1 signal par sous régions. Puis nos dernières régions sont noires uniforme donc on ne reçoit pas de signal.

2 K-means : Du "Bag of Features" au "Bag of Word"

8. Chaque clé du dictionnaire qui est une centroide de notre kmeans qui représente un descripteur sift type pour ce cluster. Dans le processus de reconnaissance d'image, pour une image qu'on veut reconnaitre : on va lui calculer plusieurs descripteurs sift que l'on va ensuite assigner à nos sifts type. Ainsi en trouvant les sifts les plus proches, on peut déterminer les features et formes qui composent cette image.
On peut voir des sifts comme des mots et une image comme un document. Plusieurs de ces mots (sifts) donnent un document (image).

9. Le barycentre c du cluster est calculé en moyennant l'ensemble des vecteurs $x_i$de notre cluster. Si on considère l'écart entre x_i et c comme une erreur, c'est bien la moyenne de tout les x_i qui va minimiser la somme des erreurs.

10. Le nombre de clusters idéal est un hyper-paramètre à déterminer avec une cross validation par exemple en évaluant sur une tâche de classification par exemple. En revanche le temps d'exécution du kmeans est assez long, en pratique cela demande une grande puissance de calcul que de faire une recherche exhaustive du cluster idéal. De plus comme il s'agit d'apprentissage non supervisé avec kmeans c'est difficile d'évaluer la performance de notre modèle.

11. Les centroïdes de notre modèle représente un sift moyen pour l'ensemble des sifts de notre clusters, c'est donc difficilement interprétable, car afficher ce sift n'est pas visuellement compréhensible puisqu'on ne peut pas reconstruire fidèlement un patch à partir de ces descripteurs. Il est donc plus intéressant d'observer quel centroide on associe à des régions d'un ensemble d'exemples d'images. On pourra plus facilement interpréter quel cluster représente quels forme/région d'une image.

12. Après avoir calculer nos k-moyennes intialiser à 1000 clusters auquel on a jouté un vecteur nul pour les descripteurs nuls, on a prit aléatoirement 30 images des quelles on a récuperé les régions et calculé les descripteurs sifts associés. On a ensuite calculé des prototypes de ces clusters en y associant les régions dont le sift était le plus proche de chaque clusters. On peut distinguer une roue, une personne...

FIGURE 2 – 100 Prototypes de nos clusters.

3 Bag of Words.

13. Le calcul du descripteur BoW permet de définir un vecteur z qui caractérise notre image. Ce vecteur est donc une représentation globale de notre image, contrairement au sift qui nous permettait seulement d'extraire des features locaux dans des sous régions de notre image. Grâce à cette représentation, il est désormais plus facile d'effectuer certaines tâches comme la classification par exemple.

14.

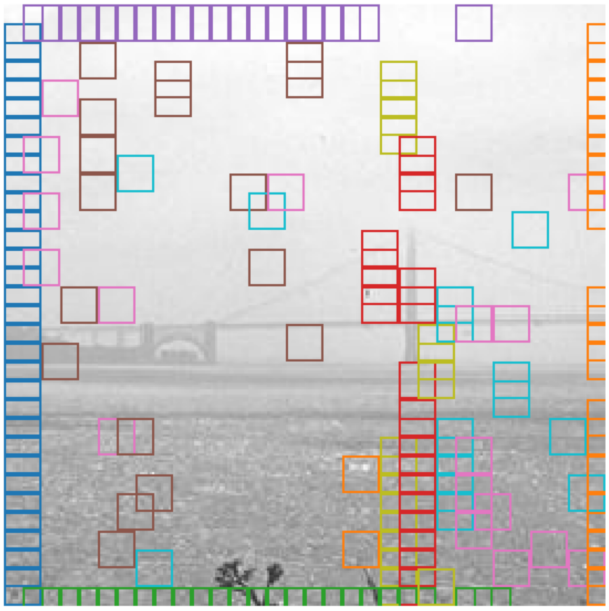


FIGURE 3 – Le SIFT et Bag Of Word appliqué sur la photo 557.

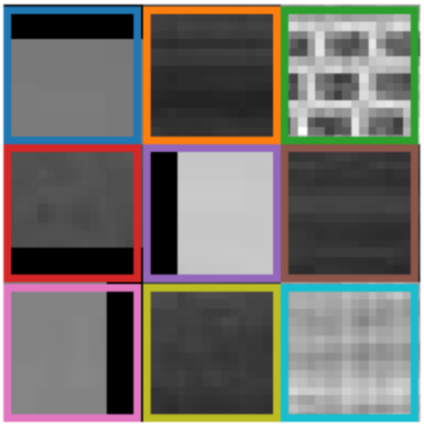


FIGURE 4 – Les régions similaires que l'on trouve sur la photo 557.

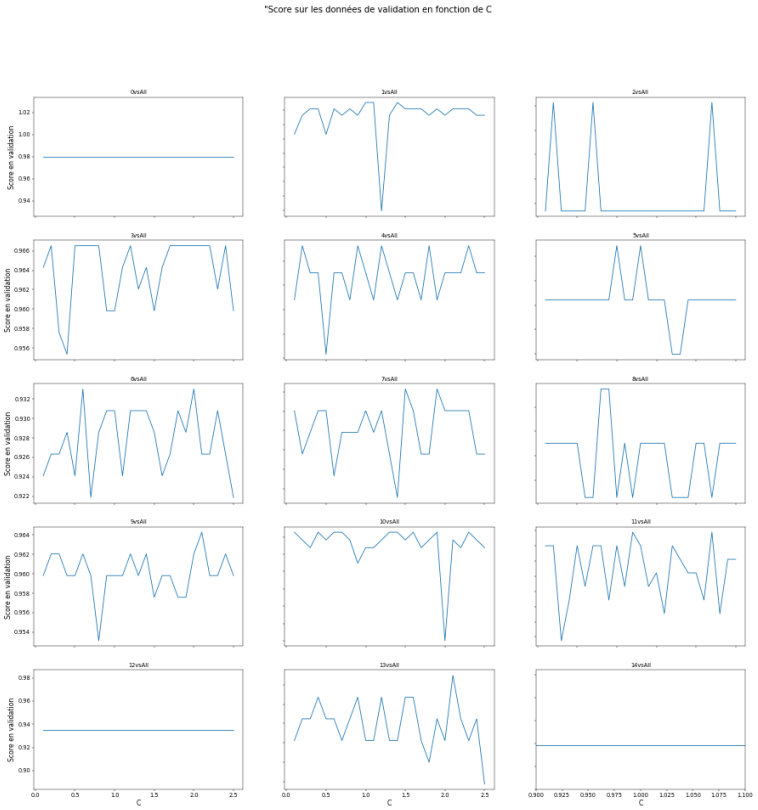
15. Le principe du codage plus proche voisins pour la phase de *coding* est d'assigner un patch à un et un seul cluster en utilisant un encodage *one-hot*. L'intérêt est de trouver un unique feature associé à ce patch en utilisant du *hard assignement*. On peut également penser à faire du *soft assignement* en donnant des probabilités d'appartenance de patch à chaque clusters.

16. L'intérêt d'un pooling somme est de compter combien de fois un cluster est apparu comme étant le plus proche dans cette image, on peut ainsi facilement donner plus d'importances à certains clusters (donc features) que d'autres. On peut également penser au max-pooling qui consiste uniquement à sélectionner le cluster (features) qui occurre le plus de fois dans notre image.

17. L'intérêt de la normalisation L2 est que la somme de notre vecteur z soit égale à 1, ce qui est très utile puisque si l'on ne normalise pas nos vecteurs, des images avec un grand nombre de features et d'autres avec un nombre de features assez faible ne serait pas traîté de la même manière. Normalisé permet donc de considérer ces vecteurs de la même façon peu importe le nombre de features.

4 Classification d'images par SVM.

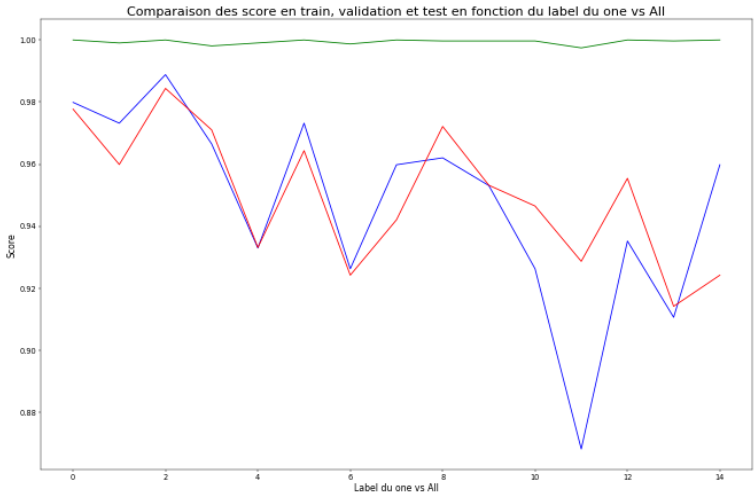
1. Puisque nous appliquons une stratégie *one versus all*, nous devons tracer une courbe de performance en fonction de C pour chaque classe contre toutes les autres. Dans le jeu de données, nous avons 15 classes différentes. Ainsi, nous obtenons ces courbes de performance sur les données de validation :



Nous pouvons observer que, pour certaines, la valeur de C ne change que très peu le score en validation. Les C à conserver sont les suivants :

Label	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C	0.1	0.2	0.1	0.3	0.2	0.8	1.6	0.3	1.1	1.0	0.1	0.4	0.1	0.5	0.1

Ainsi, si nous conservons les valeurs de C donnant le meilleur score pour chacune des classes, nous observons les résultats suivants :



La courbe verte représente le score en train avec le meilleur SVM en fonction de classe du *one versus all*. La courbe bleue représente le score en validation et la courbe rouge représente le score en test.

De ces faits, nous avons à notre disposition l'ensemble des SVMs nécessaires à la classification de nos images. Pour cela, nous prédisons la classe la plus probable grâce à tous les SVMs de la stratégie *one versus all*. Avec cette stratégie, nous obtenons un score en test de **0.6726**.

2. Théoriquement on ne peut pas exploiter et parcourir les données de test pendant la phase d'apprentissage de notre modèle. Néanmoins parfois on veut pouvoir évaluer notre modèle sur un sous ensemble, notamment pour la sélection d'hyper-paramètre. On va donc utiliser un ensemble de validation (qui fait parti de notre ensemble d'apprentissage) pour évaluer au fur et à mesure notre modèle.

3. Pour traiter une image en noir et blanc, on doit d'abord calculer les sifts sur toutes les sous régions de l'image afin d'extraire nos features. Puis à partir de notre dictionnaire visuel on peut calculer la représentation BoW de l'image, qui nous donnera un descripteur global. Pour finir on va utiliser notre classifieur SVM entraîné sur une base d'exemples dans le but de prédire un label à partir de notre descripteur global de l'image.

4. En plus d'optimiser les hyper-paramètres par *cross-validation*, on peut utiliser d'autres algorithmes pour l'extraction des features, comme un algorithme des k plus proches voisins *k-nn*, et d'autres méthodes de clustering.
Pour la classification, on peut également chercher à optimiser nos hyper-paramètres du SVM ou bien utiliser d'autres algorithme de classification comme une régression logistique multi-classes. On peut également faire de l'extraction de features et de la classification dans un même modèle à l'aide de réseaux de neurones.