FACULTÉ DES SCIENCES ET INGÉNIERIE

SORBONNE UNIVERSITÉ

RECONNAISSANCE DES FORMES POUR L'ANALYSE ET L'INTERPRÉTATION D'IMAGES

Rapport de TMEs 1, 2 et 3

Auteur :
Ahmed Tidiane BALDÉ

Encadrants :
Arthur DOUILLARD
Yifu CHEN

14 octobre 2019

TABLE DES MATIÈRES

INTRODUCTION

PRÉAMBULE

La vision par ordinateur est à ce jour l'un des domaines les plus prisés de l'intelligence artificielle. Un domaine porteur et indispensable, dont les avancés sont une des composantes majeures sur laquelle repose plusieurs autres secteurs notamment l'automobile, la défense et la sécurité, ainsi que les différents appareils électroniques dont nous faisons usage au quotidien.

Ces applications ne cessent d'émerger et certaines pourraient ici avoir un retentissement si important sur nos différents modes de vie. Je pense en particulier à une éventuelle ville plus intelligente équipées de caméras capable de suivre et de reconnaître tout types d'objets. La chine, au vue de leur progression sur ce sujet, pourrait servir d'illustration.

Elle est cette science qui permet aux machines d'analyser, de traiter et de comprendre plusieurs images. Tâche qui paraîtrait simple dit ainsi aux yeux de l'humain mais qui en réalité s'avère complexe pour une machine.

Dans ces trois (3) premiers *TMEs*, nous abordons alors l'analyse et la reconnaissance des images. Nous nous voyons confier une tâche qui consiste à classifier de nouvelles images en différentes catégories. Nous commencerons par calculer nos SIFT sur toutes les images à notre disposition puis à représenter chacune par un vecteur *Bag of Words*. Les différents mots de ce dernier ayant été trouvés à l'issu d'un *clustering*. Et enfin, nous procéderons à l'apprentissage.

DESCRIPTION D'IMAGES PAR SIFT ET BAG OF WORDS

PARTIE 1 - SIFT

1.1 CALCUL DU GRADIENT D'UNE IMAGE

Q1-1.1

Soit $h_y = \begin{bmatrix} -1/2 \\ 0 \\ -1/2 \end{bmatrix}$ et $h_x = \begin{bmatrix} 1/2 \\ 1 \\ 1/2 \end{bmatrix}$.

Alors M_x et M_y valent bien respectivement :

$M_x = h_y \times h_x^T$

$M_y = h_x \times h_y^T$

Q1-1.2

L'intérêt de séparer le filtre de convolution est de simplifier les calculs, augmenter donc l'efficacité d'un point de vue computationnel.

1.2 CALCUL DE LA REPRÉSENTATION SIFT D'UN PATCH

Q1-2.3

Le masque *Gaussien* pondéré aux gradients de notre image sert ici à *lisser* cette dernière pour se séparer du bruit et des détails sans intérêts.

Q1-2.4

La discrétisation des orientations rend notre SIFT robuste à la rotation. En effet, il y a un compromis à faire entre *précision* et *robustesse*. Plus notre discrétisation est précise, autrement dit l'angle entre deux (2) orientations est très petit, de ce fait notre matrice d'orientation ainsi que notre SIFT sont également plus conséquents, plus nous perdons en robustesse. Une légère rotation de la même image donnerait ainsi deux (2) SIFT dont les valeurs censées correspondre seraient légèrement décalées. En revanche, une discrétisation moins précise, serait plus robuste à la rotation mais a priori moins performante.

Q1-2.5

Les différents *post-processings* correspondent en quelques sortes à des a priori tout comme que le masque *Gaussien*. Ce sont des informations qui représentent ce à quoi nous voudrions que nos SIFT ressemblent.

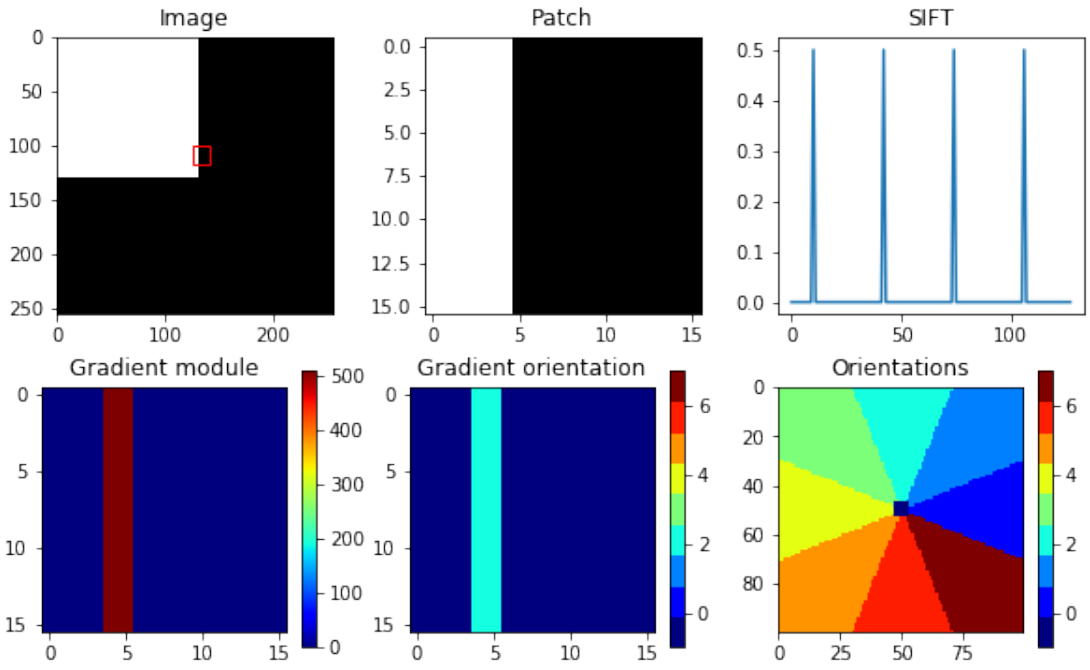
En l'occurrence, la norme L_2 met l'accent sur les grands changements d'intensité dans notre matrice de magnitude. Si les changements sont trop faibles (inférieur à un seuil de 0.5) dans ce cas nous renvoyons un vecteur nul. De ce fait, nous donnons moins d'importance aux régions correspondantes à des zones uniformes. Ensuite pour avoir un vecteur unitaire nous normalisons notre descripteur par la norme L_2, et pour finir, nous ramenons toutes les valeurs supérieures à 0.2 à cette même valeur pour éviter qu'une orientation prenne le dessus sur une autre.

Q1-2.6

Au vue de tous les avantages du SIFT énumérés ci-haut, nous pensons que c'est un principe qui est en effet une façon raisonnable de décrire numériquement un patch d'image. Sa robustesse face aux changements d'intensité, au zoom ainsi qu'à de faibles rotations, voir son invariance total dans le cadre de l'utilisation du descripteur de *Lowe*, constituent très certainement de bonnes raisons pour son usage.

Q1-2.7

En subdivisant le gradient du module en 16 *sous-régions*, et en la parcourant de haut en bas, puis de gauche à droite, nous notons d'emblée le changement d'intensité qui correspond aux différents pics dans le graphe du *SIFT*. Rappelons que l'axe des x de ce dernier correspond séquentiellement au gradient du module des différentes sous régions dans le même ordre de parcours énoncé ci-dessus.
Nous observons sur l'histogramme le changement d'amplitude qui témoigne de la variation d'intensité, l'orientation et le module du gradient y correspondent bien.



PARTIE 2 - DICTIONNAIRE VISUEL

Q2.8

Le dictionnaire est ici une nécessité dans la description de notre image car il nous permet d'avoir une représentation fini et pertinente de notre image. Le cas échéant, nous aurions représenté notre image par l'ensemble des SIFT qui la composent. Cependant cette dernière alternative n'est pas envisageable car non seulement elle impliquerait la prise en compte de certains SIFT qui ne représentent pas vraiment notre image, mais le nombre de régions varierait également d'une image à une autre, chose qui ne faciliterait pas l'apprentissage de modèles.

Q2.9

En considérant les points $\{x_i\}_{i=1..n}$ assignés à un *cluster* c, montrons que le centre du *cluster* qui minimise la dispersion est bien le barycentre des points x_i :

$$c = min_c \sum ||x_i - c||_2^2$$

Soit :

$$F = argmin_c \sum (x_i - c)^2$$

Sa dérivée première et seconde nous donne respectivement :

$$\frac{\partial F}{\partial c} = \sum_{i=1}^{n} -2(x_i - c) = -2\sum_{i=1}^{n}(x_i - c) \tag{1}$$

$$\frac{\partial^2 F}{\partial^2 c} = \sum_{i=1}^{n} 2 = 2n \tag{2}$$

La dérivée seconde de F étant positive alors sa dérivée première atteint bien un minimum global en 0. Nous avons donc :

$$\frac{\partial F}{\partial c} = -2\sum_{i=1}^{n}(x_i - c) = 0 \rightarrow \sum_{i=1}^{n}(x_i - c) = 0 \rightarrow \sum_{i=1}^{n}(x_i) - nc = 0 \rightarrow \sum_{i=1}^{n} x_i = nc \rightarrow c = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3}$$

c est donc en effet le barycentre qui minimise les points $\{x_i\}_{i=1..n}$.

Q2.10

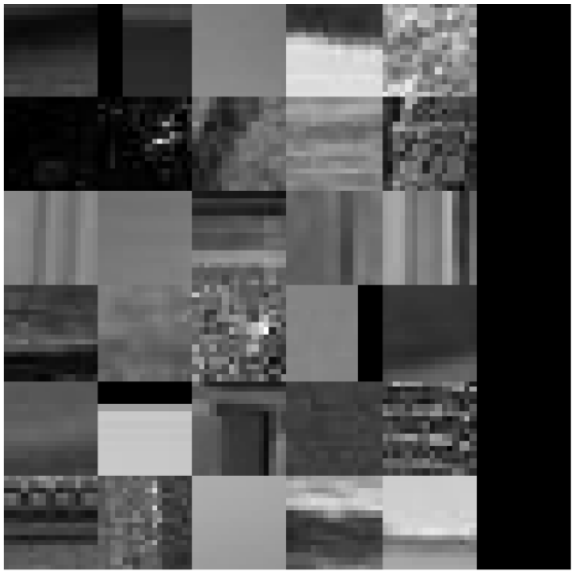
Pour choisir le nombre de *clusters* k idéal, il nous incombe d'effectuer plusieurs expériences en amont et choisir le k qui minimise l'inertie. Il convient tout autant de signaler qu'il y a un compromis à faire entre un k très élevé qui correspondrait notamment au nombre maximum de SIFT que nous avons et par conséquent à l'inertie minimale possible et un k un peu moins élevé qui augmenterait sur la même échelle l'inertie. Ces expériences peuvent être réalisées en faisant appel à certaines fonctions de *ScikitLearn* notamment *GridSearch* ou le *RandomSearch*. Cette méthode est communément appelée, *Elbow Method*.

Q2.11

Rappelons que les éléments du dictionnaire sont les *centroïdes* qui représentent au mieux les différentes catégories présentes dans notre base d'images. Ces derniers sont tout simplement des vecteurs de 128 dimensions dans l'espace des *SIFT*. Il est alors difficile d'effectuer une analyse ainsi car nous ne pouvons pas les visualiser. Dès lors, nous devons alors passer par les exemples de *patchs* pour parvenir à notre fin. C'est ainsi que nous procédons par calcul de distance euclidienne entre les *SIFT* et les différents *barycentres* de *clusters*, et attribuer à chacun de ces derniers, le *SIFT* le plus proche. De ce fait, la région qui le représente au mieux.

Q2.12

Nous appliquons le *KMeans* sur 5% de la base avec $k = 1001$, le dernier *cluster* étant un vecteur de zéros pour les SIFT nuls. Ensuite, nous assignons aux 30 premiers *clusters* les régions qui leurs représentent au mieux grâce aux calculs de la distance euclidienne en amont. Nous obtenons le résultat que voici :



Les différentes régions correspondent pour certaines à des toits, des bandes. Et pour d'autres, nous avons l'impression d'avoir des zones uniformes telles un aperçu du ciel ou un changement radical d'intensité (du gris au noir).

PARTIE 3 - BAG OF WORDS (BOW)

Q3.13

Le vecteur z représente la fréquence des mots visuels.

Q3.14

Après application du *BoW* sur une image, nous obtenons les mots visuels ci-dessous :
Les différents mots visuels qui représentent l'image se distingue de par leur nature. Certains *patchs* notamment les *oranges* ont l'air de correspondre à la partie grise du ciel et les *bleus* quand à eux, aux

changements horizontaux d'intensité. D'autres comme les *verts* correspondraient eux aussi aux changement d'intensité mais cette fois-ci verticaux, les *rouges* à l'île qui paraît au loin et les *marrons* à la texture de la forêt.
En particulier sur cet exemple, nous pouvons conclure que les mots visuels matérialisent plutôt bien l'image fournie.

Q3.15

Le codage au plus proche voisin est un codage qui permet d'avoir de bons résultats. L'affectation qui y est utilisée consiste à exiger qu'un exemple ne peut appartenir qu'à une seule classe. En revanche d'autres affectations *soft* peuvent être utilisées ce qui consisterait à donner un pourcentage d'appartenance d'un exemple à toutes les classes.

Q3.16

L'intérêt d'utiliser le *pooling* somme c'est parce qu'il est invariant à la position et il favorise les mots visuels les plus fréquents dans une image. Nous pourrons utiliser en alternative le *pooling max* ou bien *mean*.

Q3.17

La normalisation L_2 nous permet de surmonter un problème qui est la disparité des images, plus particulièrement de leurs tailles en l'occurrence. Nous ne voudrions pas qu'une image plus grande ait des valeurs beaucoup plus conséquentes qu'une image plus petite. La norme L_2 ramène ainsi les valeurs dans un espace de même dimension. La normalisation L_1 qui, quand à elle transformerait un vecteur en un

autre plus *sparse* pourrait aussi être utilisée. Tout autant que d'autres normes plus fortes.

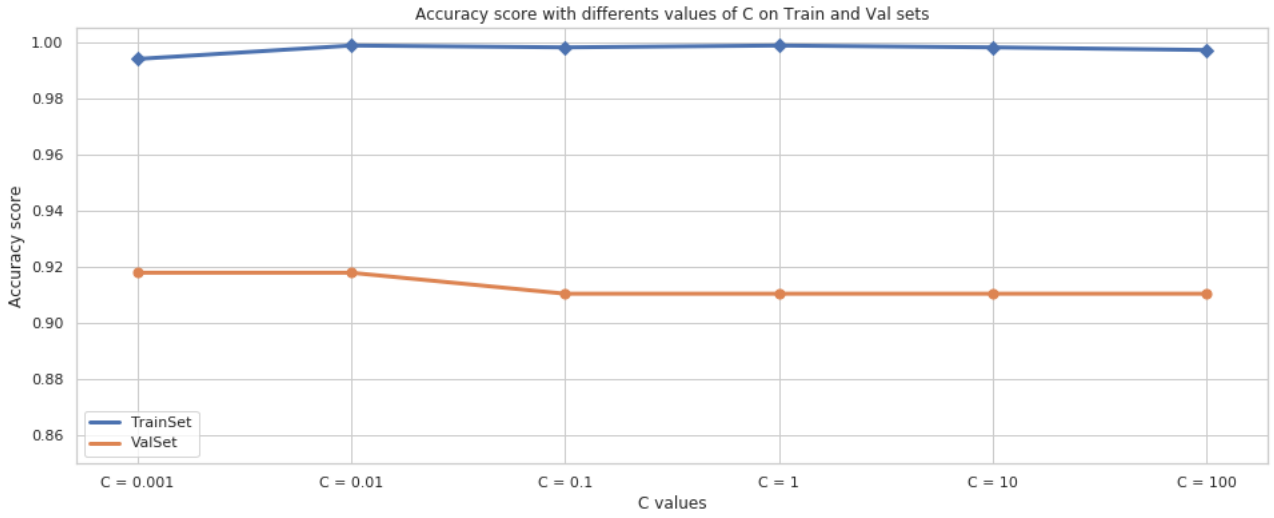
CLASSIFICATION D'IMAGES PAR SVM

PARTIE 2 - PRATIQUE

Après avoir transformé nos images en *BoW*, nous nous attelons alors à la tâche de classification. Comme suggérer dans le sujet, nous avons divisé nos données en trois (3) sous-ensembles qui sont : Train, validation et Test correspondant respectivement à 70, 10 et 20 pour cent de nos données. Rappelons que le modèle utilisé est un SVM linéaire pour de la classification *multiclass*.

Q2.1

Après un premier apprentissage, nous nous penchons sur l'*hyper-paramétrage* de notre modèle afin de trouver la valeur idéale de *C*, qui sert ici à réguler la marge. Plus C est petit, plus nous tolérons certaines exceptions dans la marge. Et inversement plus il est grand, plus nous revenons au problème de base du SVM pourlequel on n'était complètement intolérant et exigeait que les points soient linéairement séparables.



L'ensemble de *validation* ne comptant que pour 10% des données, nous ne voyons pas beaucoup de changement dans les résultats. Cependant nous pouvons noter que les résultats sont légèrement meilleurs quand *C* est petit même si nous observons l'effet inverse dans l'ensemble *Train*. Cela se justifie potentiellement par le fait que notre modèle généralise alors mieux ainsi.

Une fois l'*hyper-paramétrage* terminé, nous choisissons le meilleur modèle avec un le paramètre *C* idéal qui n'est autre que *C=0.001* et testons sur l'ensemble dédié. Nous obtenons alors un score de : 0.9051155115511551.

Q2.2

L'ensemble de *validation* nous permet principalement de pouvoir faire du *gridsearch* dans le but de choisir les meilleurs *hyper-paramètres*. Tâche que nous aurions été obligés de faire sur l'ensemble de *test*. Et naturellement nos modèles auraient été biaisés par rapport à ce même ensemble.
Il est donc nécessaire d'avoir un ensemble de *validation* pour pouvoir prétendre à une évaluation efficace et non biaisée de notre algorithme sur l'ensemble de *test*.
Rappelons que dans l'idéal nos données de *test* doivent faire office de nouvelles données complètement inconnues, rencontrées dans un cas réel. Elles ne doivent donc en aucun cas intervenir dans l'*apprentissage* de nos modèles ou dans leurs *hyper-paramétrages*.

Q2.3

Pour la classification de nouvelles images :
- Calculer les SIFT de l'image.
- Encoder l'image en BoW.
- Faire un *pooling* et récupérer le vecteur z de la nouvelle image.
- Prédire la classe de ce vecteur z grâce au classifieur SVM appris auparavant.

Q2.4

Nous pouvons améliorer notre chaîne de traitement :
- Lors d'extraction des features en utilisant des méthodes qui réduisent l'espace de dimension de nos données, comme la PCA.
- Lors de la classification, réduire le temps de calcule en utilisant des kernel permettant la séparation non-linéaire des données tel que kernel trick.

CONCLUSION

Dans ces trois (3) premiers *TMEs*, nous avons appris à calculer les descripteurs SIFT sur toutes les images à notre disposition, qui est une technique basée sur l'analyse du contenu local d'une image indépendemment de l'échelle et le comparer à celui d'autres images. Les résultats obtenus sont plutôt satisfaisants au vue de la technique utilisée ainsi que des différents avantages qu'elle regorge dans le cadre de la reconnaissance d'images tels que la robustesse à la rotation, aux changements d'intensité, etc...