TP : Classification d'images avec des descripteurs globaux

Univ. Lille – Master Informatique, parcours RVA, UE AIV2

1 Objectifs

On s'intéresse dans ce TP à la classification d'images. L'objectif est de mettre en place un pipeline de classification d'images basé sur des descripteurs globaux (histogrammes de couleurs et histogrammes de LBP) et un classifieur supervisé.

2 Données

Un sous-ensemble du jeu de données Caltech-101 sera utilisé. Il contient 1000 images réparties en 10 classes (100 images / classe) et les labels (classes) associés à ces images. Chaque image contient un objet principal correspondant à son label.

Le jeu de données est fourni dans l'archive `caltech101_subset.tar.gz` disponible sur Moodle. L'archive contient :

— les images du jeu de données, dans le répertoire `caltech101_subset`,

— la liste des 1000 images à utiliser et leur label associé, dans le fichier (texte) `caltech-subset.files`.

3 Logiciels

L'implémentation du TP se fera en Python. Les bibliothèques suivantes seront utilisées :

— OpenCV (package `cv2`), pour les fonctions de base sur les images (ouverture, conversion, etc.),

— Scikit-image (package `skimage`), pour des fonctions additionnelles sur les images (calcul des LBP),

— Scikit-learn (package `sklearn`), pour la partie classification supervisée,

— Numpy (package `numpy`), pour la manipulation des vecteurs / matrices,

— Matplotlib (package `matplotlib.pyplot`) pour la visualisation,

— le module `descriptors.py`, fourni sur Moodle, pour le calcul des histogrammes de couleurs.

Des fonctions utiles pour répondre aux questions du TP sont suggérées à la fin de chaque section de ce sujet ; il vous appartient de consulter leur documentation pour comprendre leur usage.

4 Classification par histogrammes de couleurs

On commence par mettre un place un système de reconnaissance d'images basé sur l'utilisation des histogrammes de couleurs joints. Pour rappel, le pipeline de reconnaissance est le suivant :

1. Calcul des descripteurs sur l'ensemble des images du jeu de données. On utilisera des histogrammes à $n_b = 8$ bins par canal (soit 512 bins au total).
2. Séparation des descripteurs et des labels en un jeu d'entraînement et un jeu de test.
3. Entraînement du classifieur sur le jeu d'entraînement.
4. Prédiction des labels des images du jeu de test grâce au classifieur.
5. Évaluation des prédictions du classifieur sur les données de test.

On utilisera ici un classifieur de type régression logistique (avec les paramètres par défaut de Scikit-learn). On prendra 80% du jeu de données pour l'entraînement et 20% pour le test. L'évaluation se basera sur le taux de reconnaissance (*accuracy*) et la matrice de confusion du classifieur. La graine aléatoire (*random state*) du tirage des ensembles d'entraînement et de test sera fixée (valeur arbitraire) pour permettre de reproduire les expérimentations et de conserver la même répartition des données pour les tests suivants.

Questions

1. Écrire une fonction qui permet de calculer les histogrammes de couleurs de toutes les images du jeu de données.
2. Écrire une fonction qui effectue le processus complet d'apprentissage et de test et retourne les métriques d'évaluation sur l'ensemble de test.
3. Quel est le taux de reconnaissance obtenu ? Vous paraît-il faible ou élevé ?
4. D'après la matrice de confusion :
   (a) Quelles sont les classes pour lesquelles la reconnaissance est la meilleure ?
   (b) Quelles sont les classes pour lesquelles la reconnaissance est la pire ?
   (c) Quelles sont les classes les plus confondues ?

   Interprétez ces résultats du point de vue de l'information visuelle portée par le descripteur.

Fonctions / classes / méthodes utiles

— `imread()`, fonction du module `cv2`, qui permet de lire un fichier image,
— `color_histogram()`, fonction du module `descriptors.py`, qui calcule l'histogramme de couleurs d'une image,
— `LogisticRegression`, classe du module `sklearn.linear_model`, qui implémente le classifieur régression logisitque,
— `fit()`, méthode des classifieurs Scikit-learn, qui effectue l'entraînement du classifieur,
— `predict()`, méthode des classifieurs Scikit-learn, qui retourne les prédictions du classifieur entraîné pour un ensemble de test,
— `accuracy_score()`, fonction du module `sklearn.metrics`, qui calcule le taux de reconnaissance,
— `confusion_matrix()`, fonction du module `sklearn.metrics`, qui calcule la matrice de confusion,
— `train_test_split()`, fonction du module `sklearn.model_selection`, qui sépare un jeu de données en ensembles d'entraînement et de test.

5 Implémentation des histogrammes de LBP

On va maintenant implémenter un autre descripteur : les histogrammes de LBP. Pour rappel, ce descripteur s'obtient pour une image via les étapes suivantes :

1. Conversion de l'image en niveaux de gris.
2. Calcul des signatures LBP en chaque pixel de l'image.
3. Calcul de l'histogramme des signatures de LBP.

Les signatures LBP des pixels seront calculées sur 8 points (soit 256 LBP possibles) avec un radius de 1.

Questions

1. Écrivez une fonction qui retourne l'histogramme de LBP d'une image.

2. Écrivez une fonction qui calcule les histogrammes de LBP pour l'ensemble des images du jeu de données.

Fonctions / méthodes utiles

— `cvtColor()`, fonction du module `cv2`, qui permet de convertir une image d'un espace de couleurs vers un autre,

— `local_binary_pattern()`, fonction du module `skimage.features`, qui calcule la signature LBP des pixels d'une image,

— `astype()`, méthode des tableaux Numpy, qui permet de convertir le type du tableau.

6 Classification par histogrammes de LBP

On va maintenant reproduire les expérimentations précédentes en utilisant les histogrammes de LBP comme descripteur.

Questions

1. Mettez en place un pipeline de reconnaissance d'images similaire au précédent, mais reposant maintenant sur le descripteur histogramme de LBP.
2. Calculez le taux de reconnaissance (*accuracy*) et la matrice de confusion de votre classifieur sur les données de test.
3. Comparez la matrice de confusion à celle obtenue avec les histogrammes de couleurs :
   — Quelles classes sont mieux reconnues qu'avec les histogrammes de couleurs ?
   — Quelles classes sont significativement moins bien reconnues ?

   Interprétez ces résultats d'un point de vue visuel, compte-tenu de l'information portée par les descripteurs.

7 Fusion des descripteurs

On peut combiner des descripteurs différents en un descripteur unique en concaténant les vecteurs de chaque descripteur en un unique vecteur [1] [2]. Ici, on va chercher à combiner les histogrammes de couleurs avec les histogrammes de LBP.

Questions

1. Quelle sera la taille du descripteur résultant pour une image ?
2. Écrivez une fonction qui permet d'obtenir un descripteur unique par concaténation des histogrammes de couleurs et des histogrammes de LBP calculés précédemment.
3. Évaluez ce nouveau descripteur (taux de classification) avec le même pipeline de classication que précédemment.
4. Comparez le taux de classification obtenu avec les taux obtenus indépendamment par chacun des descripteurs. Comment interprétez-vous ce résultat ?

Fonction utile : `hstack()`, fonction du module `numpy`, qui permet de concaténer des tableaux Numpy ligne à ligne.

1. Cette opération peut nécessiter une normalisation supplémentaire du descripteur final, mais ce n'est pas le cas ici.
2. On parle de fusion précoce (*early fusion*) de descripteurs.

8 Découpage géométrique des images

On peut inclure dans un descripteur de l'information sur la géométrie de l'image en découpant cette image en plusieurs régions et en calculant un descripteur pour chaque région. Le descripteur final est obtenu en concaténant dans un ordre fixe les descripteurs calculés sur chaque région de l'image. Ici, on effectuera un découpage des images selon une grille de 5×5 régions. Vous pourrez utiliser, au choix, les histogrammes de couleurs ou les histogrammes de LBP comme descripteurs des régions.

Questions

1. Quelle sera la taille du descripteur final obtenu pour une image ?
2. Écrivez une fonction qui permet de calculer le descripteur d'une image subdivisée en régions.
3. Écrivez une fonction qui permet de calculer les descripteurs de l'ensemble des images du jeu de données.
4. Mettez en place le pipeline de classification basé sur ces descripteurs.
5. Mesurez le taux de reconnaissance (*accuracy*) obtenu sur l'ensemble de test.
6. Comparez cette valeur à celle obtenue précédemment la version globale du descripteur et interprétez la différence.

Fonctions utiles

— le slicing sur les tableaux Numpy,

— `hstack()`, fonction du module `numpy`, qui permet de concaténer des tableaux Numpy ligne à ligne.