Informatique

Intelligence artificielle

Réseau de neurones et knn
Détection de balles

Présentation du sujet

Dans de nombreux systèmes automatisés (robots de tri, bras manipulateurs, véhicules autonomes etc.), la chaîne d'information repose sur une brique fondamentale : la détection d'objets par vision artificielle. Identifier la position et la taille d'un objet cible à partir d'une image est un problème de traitement du signal visuel que l'on retrouve, par exemple, dans les systèmes de contrôle qualité en production ou les robots de manipulation (comme illustré en fin de TP avec la balle saisie par une caméra embarquée).

Ce sujet suit la chaîne complète d'un tel système de perception :

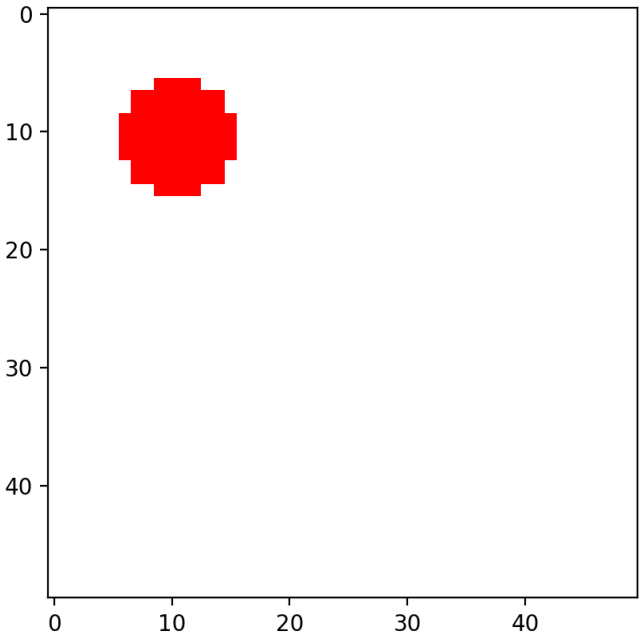
- Acquisition et représentation des données (Q1-Q2) : Transformer une image en un vecteur numérique exploitable, et constituer une base d'apprentissage (étape analogue à l'étalonnage d'un capteur).
- Apprentissage du modèle (Q3-Q4) : Entraîner un réseau de neurones à prédire la position et le diamètre d'un disque, ce qui correspond au bloc traitement de la chaîne d'information.
- Application en conditions réelles (Q5-Q9) : Tester le modèle sur des images réelles, avec redimensionnement et pré-traitement colorimétrique (illustration des contraintes d'intégration d'un algorithme dans un système physique réel).

Dans l'étude qui suit, l'objectif est de créer un réseau de neurones entrainé pour reconnaitre automatiquement la position d'une balle rouge sur une image blanche. Nous travaillerons avec des images de 50x50 pixels (nombre de lignes nl et nombre de colonnes nc).
Il sera alors possible d'étendre les possibilités de ce réseau de neurones, nous nous limiterons volontairement à une étude simple pour qu'elle ne prenne pas trop de temps.

Code élèves

Afin d'assurer un fonctionnement rapide sur tous les ordinateurs, je vous mets à disposition un dossier élèves.

Exécutez le code fourni, il créera une image blanche avec un disque rouge et l'affichera.



Création des datasets

On souhaite transformer une image (array) en une liste de flottants contenant, ligne après ligne, les valeurs R, G et B des pixels normalisées (/255).

Question 1: Créer une fonction Im2L(im) renvoyant la liste attendue

Vérifier :

```
>>> Im2L(Image)[0:10]
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

>>> len(Im2L(Image))
7500
```

Pour entrainer notre réseau de neurones, nous allons mettre en place 4 listes :

- trn_D : Liste des données, contenant pour chaque image, sa liste RGB issue de Im2L
- trn_l : Liste contenant, pour chaque image représentée par LD[i], la ligne l du centre du disque rouge s'il est présent, -1 sinon
- trn_c : Liste contenant, pour chaque image représentée par LD[i], la colonne c du centre du disque rouge s'il est présent, -1 sinon
- trn_d : Liste contenant, pour chaque image représentée par LD[i], le diamètre d du disque rouge s'il est présent, 0 sinon

On définit les deux listes suivantes :

```
Ld = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
Lrgb = [[255,0,0]]
```

A partir de ceux deux listes, nous allons proposer une fonction data() renvoyant les 4 listes trn_D, trn_l, trn_c et trn_d introduites précédemment, suivant la procédure suivante :

- Apprentissage d'une image blanche
- Pour chaque ligne l de l'image, chaque colonne c de l'image, chaque diamètre d dans Ld et chaque couleur rgb dans Lrgb, apprentissage de données de l'image concernée

On pourra afficher un pourcentage d'évolution dans la console afin de suivre ce qu'il se passe, mais ne pas en afficher trop au risque de beaucoup ralentir le code.

Question 2: Créer la fonction data attendue et créer les listes globales trn_D, trn_l, trn_c et trn_d

Apprentissage

Si besoin, pour installer le module scikit-learn, tenter dans la console « `pip install scikit-learn` » ou « `python -m pip install scikit-learn` ».

Nous allons utiliser un réseau de neurones « Classifier » de classification renvoyant un nombre fini de sorties (valeurs entières des lignes, colonnes et diamètres).

Quelques rappels :

Import module et réseau de neurones multicouches	`from sklearn.neural_network import MLPClassifier`
Création du modèle	`rn = MLPClassifier(solver='sgd',max_iter=10000)`
Entrainement du modèle	`rn.fit(trn_D,trn_l)`
Prédiction `RGB_Test` : Liste de RGB	`l = rn_l.predict([RGB_Test])[0]`
Sauvegarde de modèle	`from joblib import dump` `dump(rn,'rn.joblib')`
Chargement de modèle	`from joblib import load` `rn = load('rn.joblib')`

Question 3: Entrainer trois réseaux de neurones rn_l, rn_c et rn_d permettant de déterminer la position du centre du disque rouge et son diamètre à partir des données (Lrgb) d'une image quelconque de dimensions nl*nc

Remarque : il est normal que cela prenne beaucoup de temps… Si besoin, je vous partage mes modèles entrainés rnc, rnd et rnl (colonne, diamètre, ligne) à charger avec joblib.

On souhaite mettre en place une fonction test() créant une image aléatoire avec ou sans disque rouge, l'affichant, déterminant les données l,c,d par utilisation des réseaux de neurones entrainés, et affichant le disque trouvé en noir sur la même image.

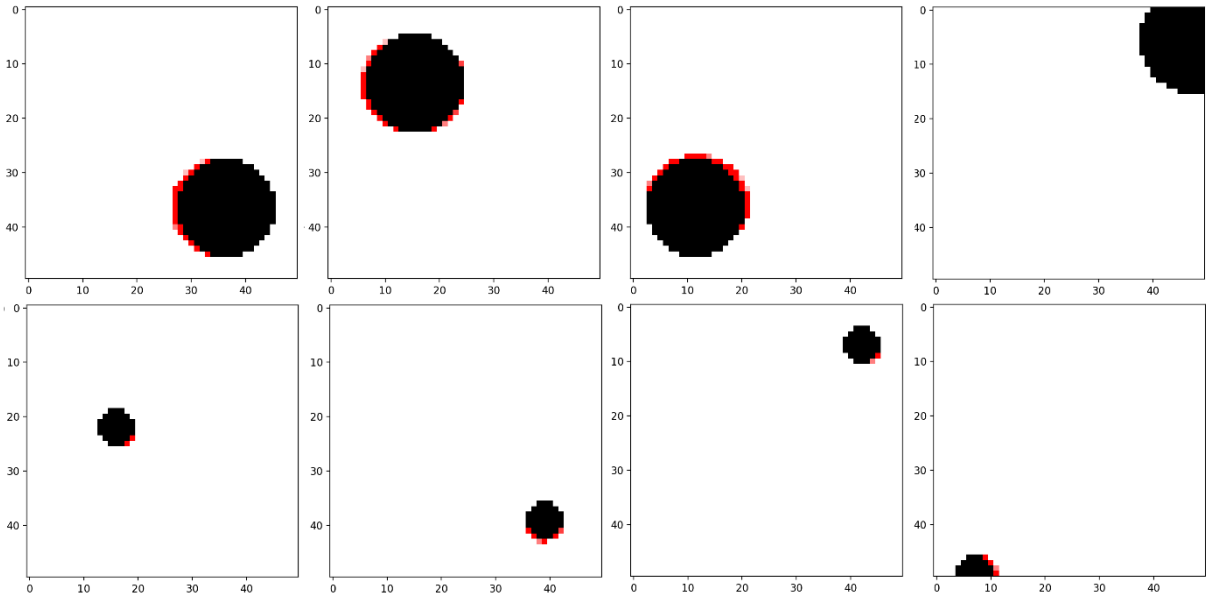
Question 4: Proposer la fonction test attendue, et réaliser plusieurs tests

Application

Vous avez à disposition dans le dossier élèves des images de disques rouges sur une image blanche carrée à une dimension différente des dimensions nl,nc.
Pour redimensionner ces images, on utilisera le module skimage et « resize » :

```
from skimage.transform import resize
Image = resize(Image,(nl,nc),anti_aliasing=False,preserve_range=True)
Image = Image.astype(np.uint8)
```

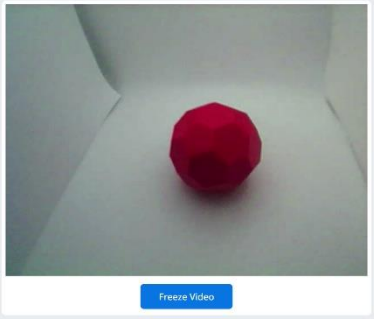
Question 5: Proposer une fonction Lecture(ch) ouvrant, redimensionnant, et renvoyant l'array associé à l'image de chemin ch
Question 6: Mettre en place le code ouvrant les 8 images fournies et affichant le disque trouvé à l'aide du réseau de neurones, en noir par-dessus



Pour aller plus loin

Amusons-nous maintenant ☺

Vous avez à disposition dans le dossier « Images » une photo nommée « Balle.bmp » d'une balle visualisée par la caméra d'un robot qui doit la manipuler, et donc, la trouver :



Il faut sélectionner le rouge dans l'image étudiée. Un simple critère sur les valeurs seules R, G et B des pixels n'est pas suffisant. Dans l'ouvrage collectif coordonné par Philippe FOUCHER de titre « Détection et reconnaissance de la signalisation verticale par analyse d'images », on trouve une référence [18] page 24 au travail de G. DUTILLEUX et P. CHARBONNIER intitulé « Détection de signalisation routière par ajustement de formes prototypes » dont voici un extrait :

2.1.1 Prédétection

Pour identifier les pixels présentant une dominantes rouge, on travaille dans l'espace RGB. Le prédétecteur stipule qu'un pixel est rouge si ses composantes vérifient

$$\begin{aligned} R &> \alpha(G+B) \\ R - \max(G,B) &> 2\alpha[\max(G,B) - \min(G,B)] \end{aligned} \quad (1)$$

Le premier critère revient à poser une contrainte sur la composante rouge normalisée $R/(R+G+B)$ qui doit être dominante. La normalisation donne sa robustesse au prédétecteur face aux variations d'illumination. Le deuxième critère vérifie que le pixel ne tend pas vers le jaune ou le magenta, c'est à dire que les composantes G et B ne sont pas trop proches l'une de l'autre. Le paramètre α peut être ajusté au niveau d'une séquence d'images. Ses valeurs habituelles varient de 0.5 à 0.75. On privilégie des valeurs de α relativement faibles pour ne pas risquer de voir un panneau disparaître lors de la prédétection.

Lorsqu'un pixel est considéré rouge, on le mettra en rouge sur l'image, blanc sinon.

Question 7: Créer une fonction Rouge(im,alpha) qui renvoie une nouvelle image (format array de numpy) qui sera la transformation de l'image im en rouge et blanc en respectant les critères proposés ci-dessus (im non modifiée)

On choisit $alpha = 0{,}75$.

Question 8: Ecrire les lignes de codes créant l'image rouge et blanc « Image_Rouge » et l'affichant

Question 9: Utiliser le réseau de neurones créé précédemment pour en déduire la position de la balle rouge sur l'image en l'ajoutant en noir sur l'image précédente

Il ne reste plus qu'à, et c'est peu dire, entrainer notre réseau de neurones sur de nouvelles données plus pertinentes, et en quantité. Par exemple, un fond généré aléatoirement avec une balle rouge, puis des balles de couleurs différentes. Mais la quantité de donnée d'entrainement sera colossale et les temps associés, démesurés sur nos petites machines.