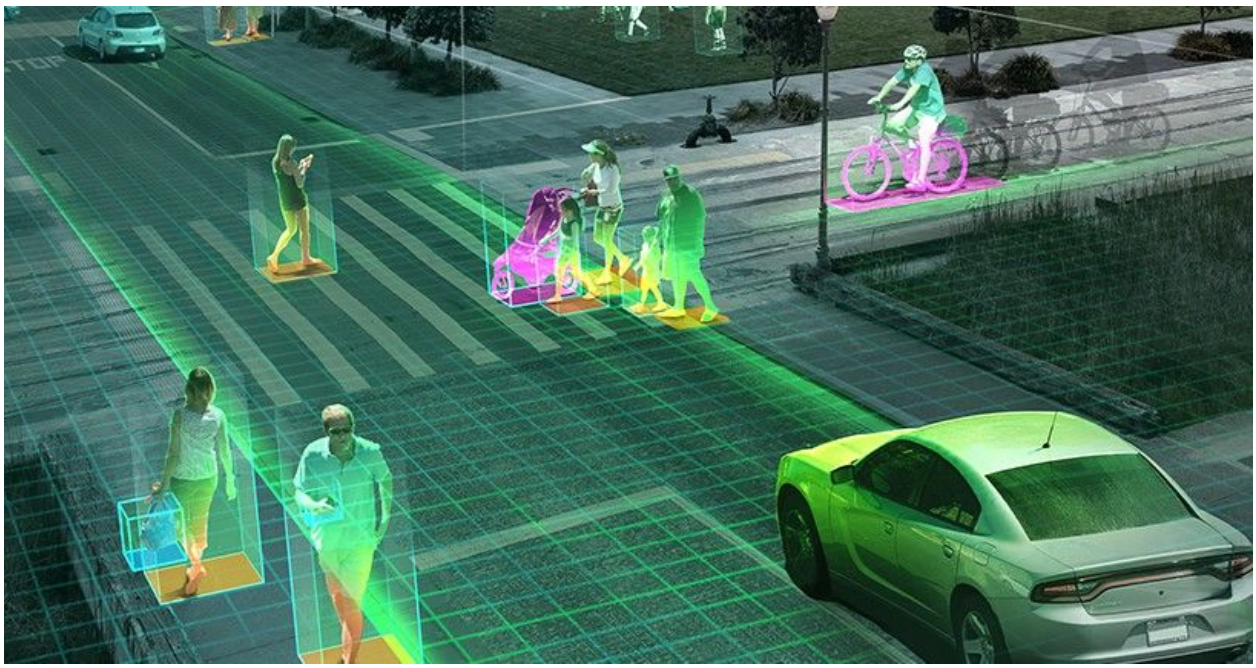
LOG635 - Introduction aux systèmes intelligents

Laboratoire 2 (v.3)

Apprendre à reconnaître des objet



Introduction

Maintenant que notre agent peut parcourir le monde, il est désormais prêt à apprendre. Vous avez déjà expérimenté avec ses capacités perceptuelles car Cozmo peut déjà reconnaître certaines choses. Dans le cadre de ce laboratoire, nous lui apprendrons à reconnaître des formes simples. Pour ce faire, nous utiliserons la banque d'images qui a été générée par l'ensemble des équipes dans tous les groupes lors du laboratoire 1.

Ensemble de données

Les images prises par toutes les équipes dans tous les groupes forment un ensemble de données que nous appellerons **EnsembleA** pour la suite de l'énoncé de ce laboratoire.

Objectifs

L'objectif de ce laboratoire consiste à construire et comparer des modèles d'apprentissage machine pour la reconnaissance d'objets simples. Pour permettre une identification simple et rapide, vous placerez des marqueurs (voir fig. 1) sur les objets (ou les personnes!) que vous désirez identifier. Ces objets seront liés à des énoncés que nous utiliserons au labo 3.

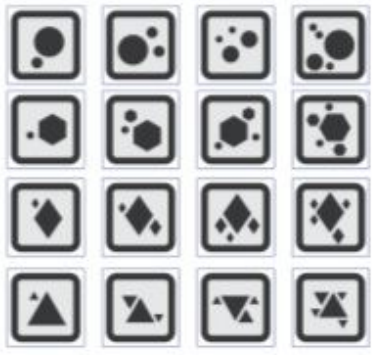


Figure 1. Un ensemble de marqueurs.

Objectifs spécifiques (OS)

1. À partir de *EnsembleA*, préparer un sous-ensemble comportant plusieurs exemples couvrant **minimalement** huit (8) symboles; cet ensemble sera appelé **EnsembleB**. Justifiez le choix de votre EnsembleB.
2. (optionnel-mais peut-être nécessaire - cf point 3) Réduire l'image (320x140) à une dimension plus petite (ex. 160 x 70);
3. (optionnel) Si vous le désirez, extraire la région d'intérêt (là où apparaît un marqueur) dans une image; l'extraction peut se faire manuellement ou de façon automatique à l'aide de la bibliothèque OpenCV (algorithme de détection de formes).
4. (optionnel) Détailler les altérations effectuées sur l'image.

   Parmi les altérations possibles:

   - L'application d'un flou pour faire disparaître les détails sur les images ce qui rend la détection de formes beaucoup plus efficace.

- L'application d'un filtre gris ce qui permet de réduire le nombre de primitives dans les images.
- L'augmentation de données: les méthodes les plus populaires pour faire l'augmentation de données sont la rotation, la translation et la distorsion des images (ajout d'un bruit);

5. Utilisez les pixels de l'image (de la région d'intérêt) pour représenter vos images; (optionnel) utilisez d'autres types de caractéristiques tels que le nombre de formes dans l'image, excentricité de la plus grande forme, etc.
6. Implémenter un réseaux de neurones complet (N'UTILISEZ PAS les implémentations disponibles en librairie) comportant minimalement 1 couche cachée; Faites une implémentation vectorisée (pas de boucles for).
7. Évaluer les performances de votre modèle en utilisant judicieusement votre *EnsembleB*. Vous pouvez choisir de faire une validation croisée à k échantillons (k=3 ou 5) ou une méthode hold-out (entraînement + validation + test). Justifiez le choix de vos hyperparamètres.
8. Évaluer trois autres modèles (SVM, Knn et RN) disponibles dans la librairie Scikit-learn;
9. (bonus) Tester d'autres modèles de la librairie Scikit-learn (ou provenant d'une autre librairie) tel que par exemple CNN.
10. Effectuer des tests de scalabilité des modèles en réduisant la taille des données d'entraînement: 80%, 60%, 40%, 20%.
11. Comparer tous les modèles en affichant les résultats et en indiquant les hyperparamètres que vous avez choisis:
    a. tableau précision, rappel, F-mesure
    b. matrice de confusion
    c. temps nécessaire pour l'entraînement (graphe ou tableau comparatif)
    d. temps nécessaire pour les prédictions (graphe ou tableau comparatif)
    e. résultats des tests de scalabilité

NB: Il est recommandé de normaliser les primitives pour les deux implémentation des réseaux de neurones (vous pouvez utiliser le MinMaxScaler ou le StandardScaler de scikit-learn).

Notes importantes

- **Marqueurs** (OS1): Vous pouvez choisir le sous-ensemble de marqueurs qui vous convient (minimum 8 types distincts).
- **Caractéristiques**: En guise de caractéristiques, vous devez utiliser les pixels des images réduites ou non (OS2). Vous pouvez choisir d'extraire d'autres caractéristiques, mais l'utilisation des pixels lors des expérimentations est nécessaires.
- **Zone d'intérêt**: Vous pouvez utiliser l'image en entier (réduite ou non) ou choisir d'extraire des zones d'intérêt (OS3). Cet aspect est totalement optionnel et s'adresse à ceux ou celles qui auraient déjà suivi un cours de vision par ordinateurs. Il n'est donc pas tenu en compte dans l'évaluation.
- **Altération**: Aucune altération de l'image n'est exigée (OS4). Par contre, si vous choisissez d'altérer les images, vous devrez détailler ces altérations.
- **Modèles** (OS7, OS8, OS9): Vous devez comparer 4 modèles -- votre implémentation d'un RN, et les modèles disponibles dans Scikit-learn: SVM, Knn et RN. Si vous testez d'autres modèles, ceux-ci doivent être soumis aux mêmes tests de comparaison.
- **Évaluation** (OS7, OS8, OS9): Chaque évaluation comprend les éléments mentionnés en OS11.
- **Temps** (OS11d, OS11e): Indiquez les caractéristiques du PC utilisé.
- **Scalabilité** (OS10 et OS11f): pour effectuer ce test vous devez réduire la taille des données d'entraînement afin de constater les effets sur la qualité des résultats des modèles.

Rapport

Votre rapport sera composé de deux (2) parties.

1. La première partie consiste en un court rapport écrit qui présente:
    a. (optionnel) les descriptions des points 2, 3 et 4.
    b. les résultats des modèle d'apprentissage (point 11);
2. La deuxième partie comprend votre *EnsembleB* afin que nous puissions reproduire vos résultats .
3. La troisième partie est constituée du code Python associé.

Lexique

EnsembleA	Ensemble contenant les images prises par toutes les équipes dans tous les groupes. Chaque image est de taille 320 x 240 pixels.
EnsembleB	Ensemble contenant les images que vous avez sélectionnées dans EnsembleA. Ces images concernent spécifiquement les classes de symboles ArUco que vous avez choisies de reconnaître.
RN	Réseau neuronal simple comportant minimalement 1 couche caché.
SVM	Support Vector Machine
Knn	K-nearest neighbour
CNN	Convolutional Neural Network