L3 - « *Computer Vision* » *L. Mascarilla,*	**TP de synthèse - Haar/CNN**	Université La Rochelle

L'objectif de ce TP, **qui donnera lieu à une note après remise d'un rapport (5-10 pages) et du code associé**, est de créer une application qui utilise des outils construits autour du DeepLearning (ainsi que ceux que nous avons déjà vu) .

- détecter (avec ou sans masque) les visages présents dans la scène.
- Afficher le nom des personnes détectées
- reconnaître l'émotion présente sur les visages.

Remarques :

- Chaque point de ce TP est une suggestion, vous êtes libres d'aller plus loin en explorant le web…

- L'objectif est de pouvoir porter votre code sur la JeVois .

Détection du visage :

A) Version 1

Vous utiliserez comme point de départ le TP sur les classifieurs par cascades de **Haar**.

1) Vous détecterez un visage dans le flux vidéo en utilisant le filtre préentrainé disponible dans `'haarcascade_frontalface_default.xml'`.

2) Affichez à chaque trame le ROI délimitant le visage.

B) Version 2

1) Vous pouvez améliorer votre méthode de détection en utilisant le réseau de détection de visages fourni par OpenCV (et présent sur la JeVois).

Le code nécessaire à l'utilisation de cette méthode sur la JeVois est donnée ici :

http://jevois.org/basedoc/PyDetectionDNN_8py_source.html

Une version plus simple est disponible sur Moodle en version OpenCV « pur », c'est-à-dire hors JeVois : **`test_face_detect_opencvdnn.py`**.

Elle peut vous servir de base pour votre détecteur !

Remarque : d'autres détecteurs potentiellement plus performants, mais plus consommateurs de ressources, sont fournis sur la page déjà donnée en référence :
http://jevois.org/basedoc/PyDetectionDNN_8py_source.html

- Que pensez-vous des résultats des méthodes de détection classiques/CNN ?

2) Vous trouverez sur le web des réseaux préentrainés qui détectent les visages masqués ou non.

Par exemple, celui-ci basé sur Yolo3 (version Darknet) est utilisable « facilement » sur la JeVois (ou en OpenCv pur) :

https://github.com/ibaiGorordo/Social-Distance-Feedback-For-The-Blind/tree/master/Part%202%20-%20Mask%20Detection/Face%20Mask%20Detection%20Inference%20Comparison

Testez !

Reconnaissance de visages :

La reconnaissance pourrait être réalisée par des cascades de Haar, mais il faudrait pour cela réaliser un entraînement une base de visages ce qui peut-être long… Une fois encore nous allons utiliser un réseau préentrainé.

Le réseau que vous allez utiliser s'appelle « **OpenFace** » :

https://cmusatyalab.github.io/openface/#overview

Vous trouverez sur Moodle le fichier contenant ce réseau :
openface.nn4.small2.v1.t7

Ce réseau a été créé sous PyTorch (une des api de deep learning les plus connues) et est utilisable dans OpenCV par :
net = dnn.readNetFromTorch(fichier)

Pour utiliser le réseau sur une image, vous :
- détecterez le ROI contenant le visage par la méthode précédente (Haar)
- normaliserez (i.e. vous transformerez l'image pour quelle soit au format attendu par OpenFace) ce ROI par la fonction dnn.blobFromImage.
L'image sera en couleur, au format 96x96, le facteur d'échelle 1/255, la moyenne (0,0,0) et `swapRB=True`.

Ensuite, le blob est fourni à l'entrée du réseau par la méthode `setInput`.
Le résultat sera obtenu par la méthode `forward`.

OpenFace renvoie alors un vecteur de 128 valeurs qui caractérise le visage donné en entrée du réseau.

Reconnaissance d'une personne :

Pour comparer deux visages entre eux, il faut calculer la (dis)**similarité** entre ldeux vecteurs a et b fournis par OpenFace. Techniquement, les vecteurs fournis par OpenFace sont des points de l'hypersphère en dimension 128, leur dissimilarité peut être calculée par une distance (classiquement on utilise le **carré de la distance Euclidienne**). Par exemple pour calculer la distance entre deux descriteurs desc1 et desc2 vous pourrez utiliser :

```
d = desc1 - desc2
distance = np.dot(d, d)
```

Pour savoir qui est présent dans le flux vidéo, il vous faudra suivre les étapes suivantes.
Avant de commencer la lecture du flux de la caméra :

- créer une base de données des images des visages des étudiants de la classe (une ou plusieurs images par étudiant).
- calculer les vecteurs donnés par OpenFace sur chacune des images.

Dans le traitement du flux de la caméra :

- détecter un visage (ROI)
- calculer le vecteur donné par OpenFace sur ce visage
- calculer la distance la distance entre le vecteur issu du ROI et chacun des vecteurs calculés sur la base de données. La distance la plus faible vous donne le nom de l'étudiant filmé.
- Afficher le nom de l'étudiant

Détection d'émotion :

Vous allez à présent détecter l'émotion affichée par le visage grâce à un autre réseau, qui est stocké dans le fichier emotion_ferplus.onnx, le principe général d'utilisation reste le même que pour le réseau OpenFace avec une différence notable : cette fois, le réseau fourni directement la probabilité de chaque émotion. Vous trouverez un exemple détaillé ici :
http://jevois.org/basedoc/PyEmotion_8py_source.html

La encore, vous obtiendrez de meilleurs résultats en utilisation comme entrée du réseau le ROI contenant le visage et non toute l'image.

Améliorations possibles: toujours plus de deep !

- Utiliser un meilleur détecteur de visages en utilisant dnn :
vous pouvez vous inspirer de https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/

- Améliorer la reconnaissance des visages en utilisant un meilleur classifieur, un k-NN ou même les Support Vector Machines (SVM) à la place de la plus petite distance :
`cv2.SVM()`, pour un exemple d'utilisation, voir ici : https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_svm/py_svm_opencv/py_svm_opencv.html#svm-opencv
ou
https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/
Remarque : ce dernier exemple utilise les SVM de scikit-learn (non présent sur la JeVois) mais vous pouvez utiliser le SVM inclus dans OpenCV

- Pour les passionnés, d'autres types de CNN sont disponibles avec OpenCV (et donc sur la JeVois) , et ils peuvent être entraînés pour reconnaître spécifiquement certains visages :
http://jevois.org/tutorials/UserTensorFlowTraining.html