PROJET TRAITEMENT ET ANALYSE D'IMAGES

IMR3 – 2021-2022

RECONNAISSANCE DE VISAGES

Contexte

La reconnaissance de visages est le problème de l'identification et de la vérification de personnes dans une image ou une vidéo par leur visage. C'est une tâche qui est trivialement exécutée par l'œil humain, même sous une lumière variable et lorsque les visages sont éventuellement modifiés par le maquillage ou l'âge, ou obstrués par la coiffure des cheveux ou des accessoires (chapeaux, lunettes). Néanmoins, elle est restée un problème de vision par ordinateur difficile pendant des décennies, jusqu'à très récemment où elle a connu un développement impressionnant grâce à l'essor de l'apprentissage profond. En effet, les méthodes modernes d'apprentissage profond sont capables d'apprendre des représentations riches et compactes de visages, ce qui a permis aux méthodes ainsi construites d'être successivement aussi performantes, puis de dépasser les capacités humaines de reconnaissance des visages.

Cette problématique est très largement répandue et est présente dans un large éventail de domaines d'application telles que la surveillance, la biométrie et, plus récemment, les réseaux sociaux. Elle peut se décliner de différentes façons et couvrir différents besoins tels que :

- L'appariement de visages : trouver la meilleure correspondance pour un visage donné,
- La recherche de similitude de visages : trouver les visages qui sont les plus similaires à un visage donné,
- La transformation de visage : générer de nouveaux visages qui sont similaires à un visage donné,
- La vérification de visage : une correspondance un-à-un d'un visage donné par rapport à une identité connue (par exemple, est-ce la bonne personne ?),
- L'identification du visage : une correspondance un-à-plusieurs d'un visage donné contre une base de données de visages connus (par exemple, qui est cette personne ?).

En pratique, la reconnaissance automatique de visages présente plusieurs défis critiques à relever dont les variations d'apparence, l'expression, le vieillissement, les variations d'échelle et les occlusions. Les visages doivent pouvoir être détectés puis reconnus avec toutes sortes d'orientations, d'angles et cela indépendamment de l'éclairage (niveaux de lumière), de l'orientation et de la distance de la caméra.

Objectif

Votre objectif dans ce projet est de mettre en œuvre les fonctionnalités nécessaires à la résolution de ces défis en suivant dans un premier temps différentes solutions appartenant à la reconnaissance d'objets dans une image puis en explorant certaines approches d'apprentissage profond.

Le projet, d'une durée totale de 24 heures, sera découpé en deux phases. Dans les deux phases, l'objectif visé est d'obtenir des résultats exploitables de reconnaissance de visages (voire

d’expressions), sinon le meilleur taux de reconnaissance dans le groupe. Le challenge est donc ouvert à tous les binômes.

Pour simplifier la position du problème, nous considérerons que la reconnaissance de visages peut être vue comme un processus qui comporte au premier abord quatre étapes successives : la détection des visages, l'alignement des visages, l'extraction des caractéristiques et enfin la reconnaissance des visages. Ces étapes sont succinctement décrites ci-dessous. Un système donné peut comporter de façon basique un module (ou un programme) distinct pour chaque étape ou combiner tout ou partie des étapes en un seul processus.

1. Détection des visages, i.e. localiser un ou plusieurs visages dans l'image et les marquer avec une boîte de délimitation (bounding box).
2. Alignement des visages , i.e. normaliser le(s) visage(s) pour qu'il soi(en)t cohérent(s) avec la base de données, comme la géométrie et la photométrie.
3. Extraction de caractéristiques , i.e. extraire les caractéristiques du visage qui peuvent être ensuite utilisées pour la reconnaissance.
4. Reconnaissance des visages, i.e. effectuer la comparaison du visage avec un ou plusieurs visages connus dans une base de données préparée.

La tâche de détection et/ou reconnaissance est considérée ici comme une tâche de prédiction supervisée, entraînée sur des échantillons de visages avec des entrées et des sorties connues. L’entrée est nécessairement une image qui contient au moins un visage, très probablement un visage préalablement détecté qui peut également avoir été recadré. La sortie peut varier en fonction du type de prédiction souhaité. Il peut s’agir par exemple d’une étiquette de classe binaire ou d’une probabilité de classe binaire dans le cas d'une tâche de vérification de visage, ou alors d’une étiquette de classe catégorielle ou d’un ensemble de probabilités dans le cas d'une tâche d'identification de visage, ou encore d’une métrique de similarité dans le cas d'une tâche de type similarité.

Indépendamment de l’approche suivie pour la phase de reconnaissance dans l’une ou l’autre des deux phases, il s’agira de dérouler une démarche en quatre étapes distinctes :

- Constitution du jeu de données d’apprentissage et de test
- Extraction des attributs
- Entraînement du classifieur sur les données d’apprentissage, et validation en cours d’apprentissage
- Test du classifieur sur les données de test

WORK PLAN

1. Préparation des outils et données

i. Outils : Python + Jupyter / Spyder / PyCharm / VSCode

La programmation sera réalisée en langage Python. Si ce dernier n'est pas installé sur votre machine, il est conseillé d'installer l'application Anaconda (disponible pour Windows et Linux) qui vous donnera un accès simplifié (via Anaconda Navigator) aux paquets et librairies scientifiques les plus récents de Python. Avec Anaconda, vous pourrez aussi facilement installer et lancer les deux IDE de base que nous utiliserons : Jupyter et Spyder.

Jupyter est un environnement web de développement en Python qui permet, depuis votre navigateur préféré, de créer, manipuler et exécuter des blocs de code Python, tout en permettant l'insertion d'annotations et de rich media de nature variée (image, vidéo, html, LaTeX, etc.). Les fichiers créés, appelés *notebooks*, ont l'extension .ipynb (IPython notebook)

Spyder est quant à lui un IDE standard prenant en charge une multitude de langages de programmation, et permettant entre autres le déboguage. Il est assez simple de transférer du code de Jupyter vers Spyder ou inversement.

Une alternative intéressante à ces deux IDE est la plateforme Google Colab qui permet l'exécution de code Python à la Jupyter, dans le cloud. Elle ne nécessite aucune installation, tout au plus l'existence d'un compte Google.

ii. L'environnement et les paquets nécessaires

Sous Anaconda, vous pouvez créer un ou plusieurs environnements de travail, chacun contenant les librairies et les paquets (*packages*) nécessaires à l'application visée. Durant ce projet, il ne sera pas nécessaire de créer un environnement particulier et vous pourrez ajouter les paquets dans l'environnement 'base (root)'.

Les principaux paquets à installer sont :

- Numpy
- Scipy
- Scikit-learn
- Scikit-image
- Opencv
- Matplotlib
- Tensorflow
- Keras

A chaque installation de paquet, Anaconda vérifie les dépendances et leur compatibilité pour une installation optimale. Il se peut que certains paquets manquent pour l'exécution de commandes Python ; dans ce cas il suffira de les installer dans votre environnement de travail, toujours via Anaconda (plus simple).

iii. Les données

La base d'images qui sera utilisée est nommée CelebA. Elle est accessible sur le lien suivant : https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

iv. Organisation standard des données

Lorsque l'on veut s'attaquer à un problème de classification de données de type *supervisé*, il faut prendre garde à bien spécifier les différents jeux de données à utiliser pour que la classification soit consistante et suffisamment robuste dans son exploitation.

Ainsi, un ensemble de données étiquetées (labellisées) sera généralement séparé en trois sous-ensembles disjoints (sans doublon) :

- un ensemble d'apprentissage : il va servir de référence, soit pour déterminer directement la classe d'appartenance d'un nouvel objet dans le cas des méthodes basées sur les distances entre plus proches voisins (*nearest neighbors* - NN) ; soit pour construire un *modèle* décisionnel qui sera utilisé ultérieurement pour prédire la classe d'un nouvel objet ;

- un ensemble de validation : celui-ci va servir à optimiser et valider les règles de décision, qui dépendent en général d'un ensemble de paramètres (p.ex. le nombre *k* de NNs pour les plus proches voisins, ou bien les multiples paramètres nécessaires au calcul d'un modèle SVM ou réseau de neurones) ;

- un ensemble de test : ce dernier va servir à évaluer la qualité de la classification obtenue sur des exemples nouveaux, c'est-à-dire n'ayant jamais été présentés au système une fois validé.

2. Phase 1- Détection et reconnaissance de visage par détection / classification supervisée d'attributs spécifiques

- Durée suggérée : 6 séances dont 4 en S50 et 2 en S1 (12 heures).

La démarche est structurée suivant le schéma classique (cf. support cours) :

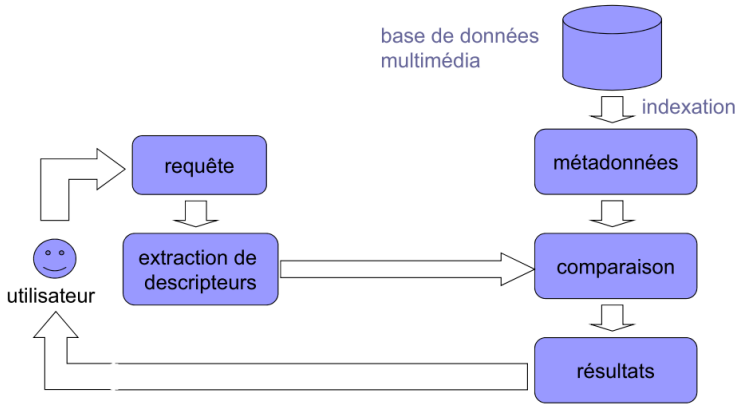

Figure 1 : Principe de la recherche d'images ou de données multimédia.

Chaque instance de la base d'images fait l'objet d'un calcul d'attributs (*features*) suivant différentes méthodes et le but est de renvoyer à l'utilisateur les images répondant le mieux à la requête.

☞**Consigne** :

- Attributs possibles
  - Cascades de Haar sous OpenCV
  - Eigenfaces

- *API*

- OpenCV Homepage.
  https://opencv.org/
- OpenCV GitHub Project.
  https://github.com/opencv/opencv
- matplotlib.patches.Rectangle
  https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Rectangle.html
- matplotlib.patches.Circle
  https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Circle.html
- Face Detection using Haar Cascades, OpenCV.
  https://docs.opencv.org/4.1.0/d7/d8b/tutorial_py_face_detection.html
- Cascade Classifier Training, OpenCV.
  https://docs.opencv.org/4.1.0/dc/d88/tutorial_traincascade.html
- Cascade Classifier, OpenCV.
  https://docs.opencv.org/4.1.0/db/d28/tutorial_cascade_classifier.html
- Eigenfaces, OpenCV.
  https://docs.opencv.org/4.1.0/da/d60/tutorial_face_main.html
  https://learnopencv.com/eigenface-using-opencv-c-python/

3. Phase 2- Détection et reconnaissance de visages par apprentissage profond

- Durée suggérée : 6 séances dont 2 en S1, 2 en S2 et 2 en S4 (12 heures).

Dans cette partie, il s'agira de mettre en œuvre un réseau de neurones (peu profond), qui prendra en entrée directement des images, sans passer par une étape d'extraction d'attributs.

i. Une première approche empirique d'un réseau de neurones

Afin de vous familiariser avec ce qu'est et comment fonctionne un réseau de neurones, nous vous invitons à consulter (jusqu'au bout) la page web interactive suivante :

http://playground.tensorflow.org

qui résume efficacement une grande partie des concepts qui seront utilisés plus loin pour la classification d'images. Le but est de partitionner au mieux des points en dimension 2 issus d'un ensemble d'apprentissage, en deux classes labellisées, tout en veillant à prédire au mieux la classe d'appartenance des points appartenant à un ensemble de test.

Une bonne règle de classification doit permettre d'éviter :

- le sur-apprentissage : en contraignant la méthode pour que la classification des données d'apprentissage suive au plus près les labels qui leur sont affectés, on la rend peu généralisable : appliquée à la base de test, la classification risque d'être peu performante.

- Le sous-apprentissage : en diminuant la complexité de l'algorithme ou du modèle de décision, celui-ci sera moins adapté aux données d'apprentissage, et conduira également à de faibles performances en test.

Il existe donc en principe une stratégie optimum de classification.

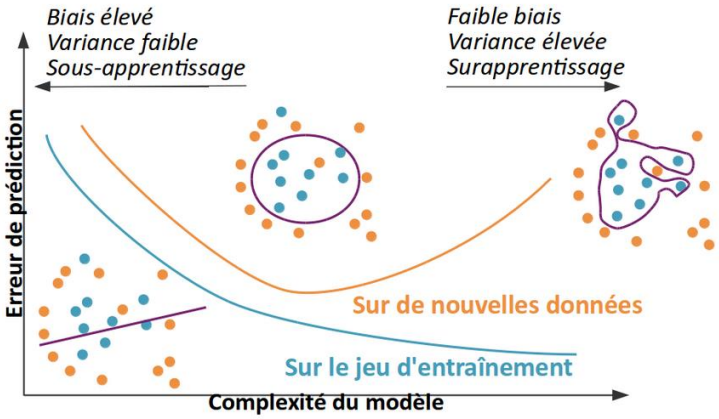

Figure 2 : Performances d'un modèle de classification en fonction de la complexité. Source : https://openclassrooms.com/fr/courses/4297211-evaluez-et-ameliorez-les-performances-dun-modele-de-machine-learning/4297218-comprenez-ce-qui-fait-un-bon-modele-d-apprentissage

ii. Application à la reconnaissance faciale

Vous trouverez sur :

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Une mise en œuvre simple d'un réseau de neurones convolutif peu profond. Une fois spécifié le réseau, ce code doit vous permettre de produire un modèle, qui contiendra la structure globale du réseau à la fin de l'apprentissage (paramètres compris) et sera enregistré au format HDF5 (.h5). Ce dernier sera ensuite utilisé pour la partie test dans un code Python distinct, qui permettra de prédire les classes d'appartenance des images de l'ensemble de test.

☞Consigne :

- Structures neuronales originales à développer sous TensorFlow en s'inspirant de la méthode précédente afin de la mettre en œuvre et de l'adapter pour une application de reconnaissance faciale.

- *API*

- TBA.

4. Rendu

Le compte-rendu du projet, avec ses deux phases (détection/reconnaissance par attributs puis par réseaux de neurones) devra être dématérialisé. Le plus simple est de procéder à un dépôt sous moodle de l'ensemble sous la forme d'une archive (de type zip, rar, 7z), ce qui vous permettra de décrire les solutions apportées, d'inclure vos codes (.py ou .ipynb) et d'illustrer le rendu avec vos résultats obtenus. En pratique deux fichiers sont attendus par binôme:

- un fichier pdf du CR de projet (max 20 pages)
- un dossier (arborescence adaptée) contenant vos codes .py et/ou .ipynb et tutoriels d'illustration.

Les bases d'images ne doivent pas être incluses, mais le code doit comprendre les indications nécessaires pour permette un accès facile aux répertoires et sous-répertoires dédiés (train, validation, test, results).

Vous veillerez également à la qualité de la rédaction (style et orthographe).

Référence bibliographique

Pour une découverte plus approfondie de la littérature scientifique correspondante, l'article intitulé « Deep Face Recognition : A Survey», publié en 2018 fournit un résumé utile de l'état de la recherche sur la reconnaissance de visages au cours des 30 dernières années, soulignant l'évolution de la démarche suivie en partant des méthodes reposant sur un apprentissage « holistique » (telles que les Eigenfaces), pour passer ensuite par celles basées sur la détection locale de caractéristiques spécifiques appropriées (hand-crafted), et arriver aux méthodes d'apprentissage peu profond (shallow learning), et enfin aux méthodes d'apprentissage profond qui sont actuellement à la pointe de la technologie.

[1] Deep Face Recognition: A Survey, 2018, https://arxiv.org/abs/1804.06655