université
PARIS-SACLAY

Manuel technique de l'application:

Reconnaissance faciale et détection de sentiments

UE17

Membres du groupe:

- Takfarinas NAIT-LARBI
- Laetitia BENALI
- To Dung NGUYEN
- Lydia AMRANE

Responsables:

- Fatiha ZAIDI
- Lila BOUKHATEM

1. Introduction:

a. Contexte:

Dans le cadre de notre formation Master Data Science de l'université paris saclay, et de l'UE17 Projets**,** nous sommes chargés de réaliser un projet dans le thème de l'apprentissage automatique sur les notions que nous avons abordées au cours de cette année et d'appliquer nos connaissances à un cas d'usage réel.

b. Choix de la thématique:

Nous avions le libre arbitre quant au choix de la thématique que nous allions aborder. C'est donc d'un commun accord que nous avons décidé de nous aventurer dans le terrain de la reconnaissance faciale.

Pourquoi?
La reconnaissance faciale est aujourd'hui un domaine connu, et nous avons porté notre intérêt là-dessus car il est en grande expansion, c'est un sujet qui peut être enrichi de pleins d'idées et de cas d'usages différents.

c. Etapes du projet:

3 grandes étapes sont à distinguer :

- ❏ Élaboration du modèle de reconnaissance faciale.
- ❏ Élaboration du modèle de détection de sentiments.
- ❏ Élaboration de l'application.

En effet, nous avons décidé d'agrémenter notre reconnaissance faciale d'une détection de sentiments, 2 notions différentes qui forment les grands axes de notre projet.

d. Répartition des tâches:

Au cours de ce projet nous avons décidé en premier lieu de nous retrouver lors d'ateliers de réflexion et de travail collectif afin d'avancer plus rapidement et plus efficacement.
Puis, une fois que notre ligne directrice a été définie, nous nous sommes partagés les tâches concernant le codage, la mise en place de l'application et la rédaction des manuels.

2. Première étape : Élaboration du modèle de reconnaissance faciale:

a. Ressources et documentation:

Afin de construire notre modèle de reconnaissance faciale, nos recherches nous ont menées à plusieurs bibliothèques et méthodes python dans ce thème là. Les librairies ***OpenCv*** et ***DLib*** nous semblaient être le meilleur choix afin de réaliser ce que nous voulions faire.

b. Dataset:

Parmis les avantages de Dlib que nous avions choisis est que le modèle se base sur du *one shot learning*, c'est à dire que nous avons besoin uniquement que d'une ou 2 images afin d'entraîner notre modèle.

c. Modèles d'entraînement:

La bibliothèque Dlib nous fournit un certain nombre de modèles pré-entraînés que nous devions télécharger afin d'y appliquer notre dataset.

d. Procédé et explication des fonctions les plus importantes:

Le principe de cette reconnaissance faciale est de récupérer les images du dataset, les traduire en Array , puis de les encoder grâce à la fonction d'encodage.
Pour encoder une image, nous retrouvons dans la librairie Dlib la fonction ***face_detector*** qui nous fournit une liste des coordonnées des visages sur l'image.

Grâce à ces résultats nous pouvons appliquer la fonction ***predictor_shape_68*** qui nous retourne 68 points de coordonnées d'un visage ce qui nous assure une bonne qualité de reconnaissance.

Puis il faudra appliquer la fonction ***compute_face_descriptor*** qui nous fournit un vecteur de dimension 128 qui décrit le visage.
À l'issue de ces étapes on a donc une liste de coordonnées du visage ainsi qu'une liste de landmarks (points du visage) et le vecteur 128.

Suite à ça nous allons appliquer la fonction de reconnaissance ***linalg.norm*** qui est la fonction *la plus importante*. Cette fonction calcule la norme de tous les

visages connus **-** la norme du visage qui vient d'être détecté (score de similarité):
linalg.norme (visages connus - visage détecté)
Grâce à ça on va récupérer le visage le plus proche qui est connu du visage qui vient d'être détecté.

Puis avec un seuil de tolérance qu'on a fixé à **0.6** on considère que le résultat de notre soustraction est acceptable pour reconnaître un visage ou non (plus il est proche de 0, plus le niveau d'exigence est haut).

Au final, il faudra ajouter certaines fonctionnalités afin d'afficher le cadre sur le visage, d'id client, les points des landmarks pour aboutir à un modèle de reconnaissance utilisable et fonctionnel.

3. Deuxième étape : Élaboration du modèle de détection de sentiments:

a. Ressources et documentation:

Afin de construire notre modèle de détection d'émotions, nos recherches nous ont menées à plusieurs bibliothèques et méthodes python dans ce thème là. Les librairies ***Keras et tenserflow*** nous semblaient être le meilleur choix afin de réaliser ce que nous voulions faire

b. Dataset :

Afin d'entraîner le modèle de détection d'émotions on a utiliser un dataset qu'on a trouvé sur Kaggle, qui est constitué d'un ensemble d'images qui sont réparties sur 5 classes qui sont **HAPPY, SURPRISE, SAD , NEUTRAL, ANGRY,** vous pouvez trouver les donnée avec ce lien :
https://drive.google.com/drive/folders/1Anba3tHukZN1S4hjzfvUGdLJKojzZxRH?usp=sharing .

c. Etapes de construction Modèles d'entraînement:

Afin d'expliquer la construction du modèle, on peut divisé ce processus en trois étapes qui sont les suivantes :

- Uploader les données, et appliquer la technique d' Image Augmentation, qui une technique de génération de données artificielles à partir de données originales, en apportant de légères modifications sur les images brutes. On appliquer cette technique uniquement sur le train set et pas sur Le test set car on voulait fausser les scores du modèle et rentrer dans l'overfitting .

- Création et entraînement du modèle, notre modèle est constitué de 7 couches, une couche d'entrée, 5 couches cachées, et une couche de sortie, et pour cela on a utilisé la bibliothèque Keras
- Enregistrement du modèle pour l'utiliser dans le flux de caméra .

4. Troisième étape : Élaboration de l'application:

Notre application finale est le résultat de 2 applications intermédiaires différentes.

La première est l'interface dans laquelle nous observons le flux vidéo des clients. La seconde est l'interface dans laquelle nous obtenons les informations concernant les clients reconnus à l'intérieur du magasin.

a. Interface vidéo:

Cette interface a été créée en parallèle de la conception de nos 2 modèles. En effet, il s'agit de l'affichage résultant de l'exécution de notre algorithme.
On y retrouve en temps réel l'affichage de l'ID de chaque client sur son visage ainsi que l'humeur détectée.

Nous avons également décidé de réaliser 3 programmes distincts afin de gérer plusieurs points du magasin.
En effet, notre idée de conception est de mettre en place 3 systèmes de caméras: à l'entrée du magasin, aux caisses et à la sortie du magasin.

- **À l'entrée du magasin:**
  L'idée est qu'à l'entrée du magasin, il faudrait poster des caméras qui récupéreraient les informations du client, de le créer s'il ne fait pas partie de nos bases, ou bien de le mettre à jour sinon.

- **Aux caisses du magasin:**
  Nous avons pensé à également poster des caméras aux caisses dans le but d'avoir des informations quant aux clients ayant acheté ou pas.

- **A la sortie du magasin:**
  Les caméras se trouvant à la sortie pourront nous apporter les informations concernant l'humeur des personnes à leur sortie, dans le but d'une étude de satisfaction, mais aussi de pouvoir mettre à jour les clients étant dans le magasin ou pas.

b. Interface Informations clients:

Tout d'abord, il est important de savoir que dans l'interface vidéo, il s'agit d'une bande passante de vidéo, il ne s'agit pas d'enregistrer et de stocker les rushs.

Notre première idée a donc été de réaliser des photos des visages des clients lorsque l'interface vidéo les détectent, afin de pouvoir les stocker.
Pour ce faire, il a fallu premièrement rechercher le moyen de pouvoir réaliser ces photos et nous avons procédé de la manière suivante:

Lorsqu'on détecte un visage, on prend une photo, on récupère les coordonnées le délimitant, puis on rogne la photo en fonction de ces coordonnées, ainsi, il est possible de récupérer plusieurs visages en même temps.

Deuxièmement, il était indispensable de créer une base de données afin de pouvoir stocker ces images en temps réel pour obtenir les informations de nos clients.

a. Base de donnée:

La solution afin de pouvoir stocker nos informations a donc été d'utiliser une base ***MongoDb.*** Cela nous paraissait intéressant d'en plus, avoir une base de données que nous pourrions interroger en
MongoDb met à disposition des BD se trouvant sur le cloud, que nous pouvons manipuler à l'aide PyMongo, qui est une distribution fournissant des outils et reliant Python et MongoDb.
De plus, nous avons appris cette année que MongoDb n'est pas une base RDBMS, on pourra donc y stocker des informations non formatées telles que des images, ce qui a appuyé notre choix.
Une fois notre Base de Données connectée et opérationnelle, nous y avons introduit les photos des clients identifiées avec leur ID client, et nous avons décidé de l'enrichir avec certaines informations.

b. Informations clients:

Le choix des informations à fournir a été réfléchi de manière industrielle et métier.
En effet, il nous paraissait important d'avoir des informations utiles afin de pouvoir prouver l'utilité de notre application.
Nous avons donc choisis:

- ★ ***Identifiant client (clé):*** Le même identifiant que celui affiché et celui de la photo stockée
- ★ ***L'image:*** Collectée sur l'interface vidéo
- ★ ***Nb_visit:*** Le nombre de visites du client dans le magasin
- ★ ***Nb_achat:*** Le nombre d'achats du client dans le magasin
- ★ ***Mean_shop:*** Le panier moyen par client
- ★ ***Emotion_in:*** L'humeur du client à l'entrée du magasin
- ★ ***Emotion_out :*** L'humeur du client à la sortie du magasin
- ★ ***Last_visit:*** La date de sa dernière visite
- ★ ***Proba_Achat:*** La probabilité d'acheter
- ★ ***Voleur:*** Booléen nous informant si le client a déjà volé dans le magasin ou pas
- ★ ***Inshop:*** Booléen nous informant si le client est à l'intérieur du magasin ou pas

Ces informations ne sont pas toutes mises à jour en même temps.
En effet, comme il y a 3 systèmes de caméras dans le magasin, c'est donc là qu'on en verra l'utilité.

- ➔ ***À l'entrée :***

  Lorsqu'un client rentre dans le magasin et qu'il n'est pas reconnu dans notre base de donnée, on lui créera à ce moment-là son profil avec toutes les informations initialisées.
  On a donc le vecteur:
  (id_client, image, nb_visit = 0, nb_achat= 0, mean_shop = 0, emotion_in, emotion_out ='nothing', last_visit, proba_achat = 0.5, voleur=0, inshop = 1)
  Sinon, si le client est reconnu, on y fera les mises à jour des variables: emotion_in, inshop=1.

➔ ***À la caisse :***

Un client arrivant à la caisse est un client qui achète, on mettra donc à jour : nb_achat+=1, mean_shop, proba_achat

Sachant que les informations concernant le mean_shop nous seront transmises de la part du/de la caissier/e au moment où elle validera l'achat du client sur son poste.

De plus, si le client n'est pas reconnu par la base il sera créé à ce moment-là. (Dans le cas où il y a eu une mauvaise détection à l'entrée)

➔ ***À la sortie :***

Lors de la sortie du client les informations mises à jour seront donc: nb_visit+=1 , emotion_out, last_visit, inshop=0.

Également, tout client dont le *inshop=0* (qui n'est pas dans le magasin) ne sera pas affiché sur l'interface d'informations clients.

Finalement, en rassemblant ces 2 interfaces, on crée une application utilisateur en utilisant une des bibliothèques d'interface graphique de python : **Tkinter**.
C'est donc sur cette interface qu'on retrouvera nos fonctionnalités introduites précédemment.
Afin de savoir comment utiliser cette application, il faudra se reporter au manuel d'utilisation.