



Détection de l'âge et du genre par intelligence artificielle

**“Le visage est l'image
de l'âme.”**

Cicéron

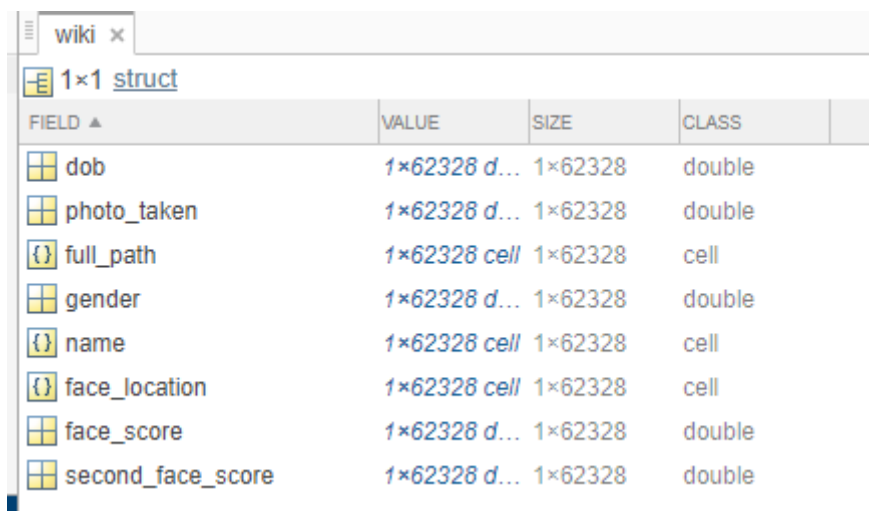
Détection du genre

Récupération et préparation des données

Avant de construire notre modèle il faut tout d'abord récupérer une base de données qui contient des différentes images d'homme et femme qui vont représenter nos données d'entraînement et de test.

Pour commencer, on a dû récupérer une base donnée qui contiennent plus de 60 000 images depuis IMDB-WIKI dataset, qui offre une énorme base donnée des différentes images de visages publiquement disponibles avec des étiquettes du genre et d'âge.

Dans cette base données, on a récupéré un fichier .mat (matlab) contenant toutes les métadonnées des images (chemin de l'image, genre ...), et grâce à ce fichier on a pu convertir ces métadonnées sous forme des fichiers Excel afin de récupérer le genre de chaque individu qui apparaît dans ces images dans un fichier .csv.



FIELD	VALUE	SIZE	CLASS
dob	1×62328 d...	1×62328	double
photo_taken	1×62328 d...	1×62328	double
full_path	1×62328 cell	1×62328	cell
gender	1×62328 d...	1×62328	double
name	1×62328 cell	1×62328	cell
face_location	1×62328 cell	1×62328	cell
face_score	1×62328 d...	1×62328	double
second_face_score	1×62328 d...	1×62328	double

Par la suite, on a divisé cette base de données en deux : données d'entraînement et données de test (80% - 20%) , et à l'aide des fichiers Excel générés, on a divisé les deux bases données en deux sous fichiers qui représentent le genre de la personne qui apparaît dans l'image (male / female).

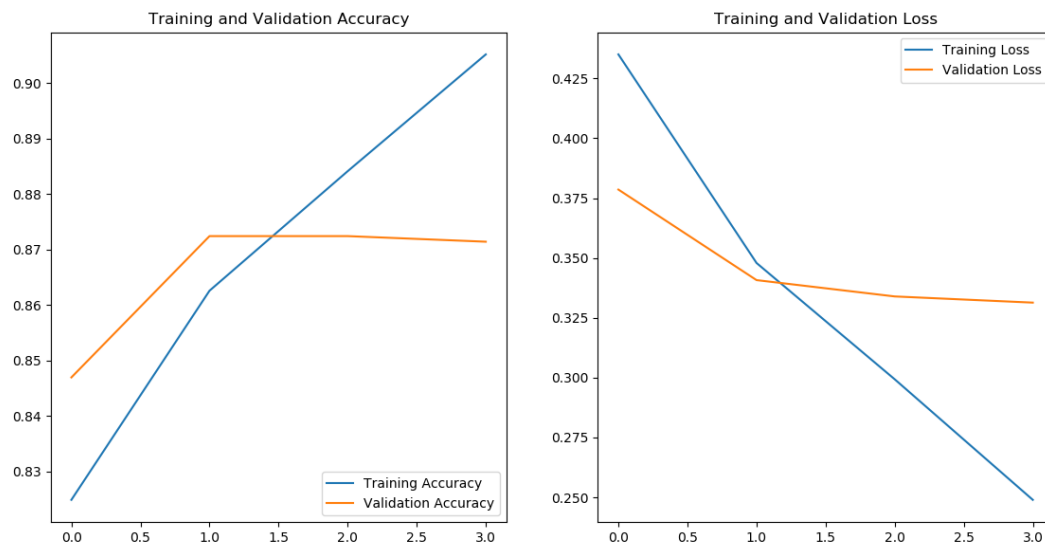
```
def setTrainingData():
    pathImg = getPathImg()
    genders = getGenderImg()
    i = 0
    for index,img in enumerate(pathImg):
        if genders[index] != "NaN":
            path = os.path.join("wiki_crop",img)
            if i < 47749:
                os.rename(path,"Dataset/train-data/"+class_names[int(genders[index])+"/"+str(i)+".jpg")
            else :
                os.rename(path,"Dataset/test-data/"+class_names[int(genders[index])+"/"+str(i)+".jpg")
            i=i+1
```

Création du modèle

Cette étape consiste à créer un modèle à l'aide de la bibliothèque Keras. Tout d'abord, on a construit notre réseau de neurones avec quatre couches "Conv2D" afin d'extraire les caractéristiques des images traitées, et deux couches "Dense", la première de type "relu" avec 64 neurones et le deuxième avec 1 neurones de type "sigmoid" qui va contenir le résultat final de notre prédiction (0 pour femme et 1 pour homme).

```
model = Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

Par suite, on a testé plusieurs modèles avec des différents nombre d'epochs et de batch size, mais le meilleur résultat obtenu était avec 4 epochs et un batch size de 128. On a réussi d'obtenir un résultat de 90% de réussite avec les données d'entraînement et 87% avec les données du test.



On a enregistré ce modèle sous forme de fichier .h5 sous le nom de 'genderDetection-model.h5', qu'on a converti en un fichier .tflite afin qui soit lisible et interprétable par TensorFlow Lite sous Android.

```
def convertToTflite():
    converter = tf.compat.v1.lite.TFLiteConverter.from_keras_model_file( 'genderDetection-model.h5' )
    converter.post_training_quantize = True
    tflite_buffer = converter.convert()
    open( 'model.tflite', 'wb' ).write( tflite_buffer )
```

Détection de l'âge

La reconnaissance de l'âge est d'une grande complexité, même avec les technologies d'intelligence artificielle actuelles. Dans un premier temps, nous avons dû trouver une base de données correcte pour ce type d'application. En effet, si la reconnaissance de genre est moins exigeante à ce niveau, la reconnaissance d'âge demande des données d'entraînement de qualité certaine.

La base de données UTK (Université de Knoxville, Tennessee) a semblé être un bon compromis avec des images de personnes de différentes origines et des âges allant de 1 à plus de 100 ans.

L'étape suivante fut de créer un modèle Keras. Tout d'abord, nous avons testé un modèle très simple composé d'une couche Dense avec 128 nœuds et d'une couche Softmax à 5 sorties pour 5 classes d'âge (0-19, 20-39, 40-59, 60-79 et 80-99).

```
model.add(keras.layers.Dense(256, activation='relu'))
model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(5, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=100, validation_split=0.2, batch_size=32, shuffle=True)
```

Les résultats de ce modèle étaient moyens, donc nous avons testé différents modèles plus complexes en introduisant notamment des couches Dropout afin de minimiser le phénomène d'overfitting. Il s'est avéré que les résultats de ce modèle n'étaient pas réellement meilleurs.

```
model.add(keras.layers.Conv2D(32, (3, 3), kernel_constraint=max_norm(3), bias_constraint=max_norm(3),
                             input_shape=(200, 200, 3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(keras.layers.Dropout(0.4))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(64, kernel_constraint=max_norm(3), bias_constraint=max_norm(3), activation='relu'))
model.add(keras.layers.Dropout(0.4))

model.add(keras.layers.Dense(5, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=100, validation_split=0.2, batch_size=32, shuffle=True)
```

Cependant, nous avons remarqué que le dataset d'entraînement comprenait beaucoup de photos appartenant à la classe 2 pour laquelle le modèle avait une très large tendance. Environ 50% des images d'entraînement appartenaient à celle-ci. Une fois le nombre de photos d'entraînement égalisées en fonction des classes (250 dans chaque classe), le taux de bonnes réponses s'est amélioré.

Avec 100 epochs sur un ordinateur plus puissant, nous avons réussi à obtenir des résultats de l'ordre de 85% de réussite, ce qui est satisfaisant étant donné la complexité de l'exercice et l'état de l'art dans ce domaine.

Une autre approche aurait pu être d'utiliser un modèle pré-entraîné à la place. Cependant, le manque de temps et de connaissances de notre côté nous a valu de préférer la solution énoncée ci-dessus.

Application Android

Détection faciale

Au début, on a utilisé **OpenCV** (Open Computer Vision), une bibliothèque graphique qui est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo afin de détecter les visages dans les images. Mais on a rencontré pas mal de problèmes au niveau de la configuration. Donc on a pensé à un autre outil plus facile à savoir "**Cloud Vision**" qui est une API (interface de programmation) lancée par Google, qui vient compléter les services et outils fournis par sa plateforme Google Cloud, qui va venir ajouter à notre application la capacité de reconnaissance d'entités au sein d'une image.

En général, l'API Cloud Vision classifie rapidement les images dans des milliers de catégories, détecte les visages avec les émotions associées, et reconnaît les mots imprimés dans de nombreuses langues.

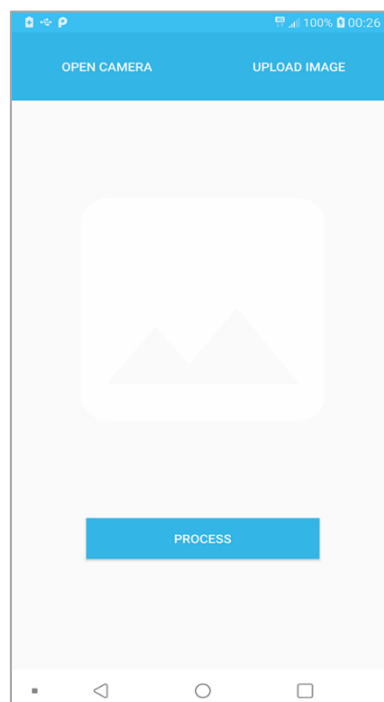
Notamment, elle analyse les images que nous lui fournissons en ayant recours à de l'IA (intelligence artificielle), et plus précisément au machine learning (capacité pour un programme "d'apprendre" via des algorithmes et de faire de la prédiction de données).

Mais dans notre projet, on va se focaliser sur la détection faciale : cette fonctionnalité permet de détecter l'apparition d'un visage sur des photos, avec les traits du visage associés tels que le placement des yeux, du nez et de la bouche afin de deviner par la suite le genre et l'âge de la personne à partir de l'image détectée.

Interface

On a créé notre interface avec **Android Studio** qui est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle.

Voilà un aperçu de l'interface de notre application mobile :



L'interface contient trois boutons :

- **Open camera** : permet d'ouvrir la caméra du téléphone afin de prendre une photo.
- **Upload image** : cette option permet de charger une image existante dans la galerie du téléphone.
- **Process** : permet de détecter le visage sur notre image et deviner le genre et l'âge de la personne.