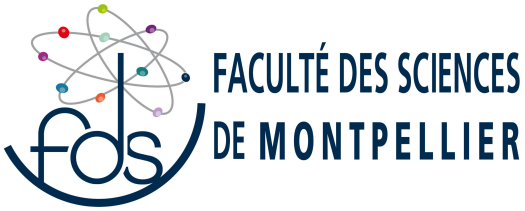


Projet Images - Compte Rendu 2

Filtres de visage

Andrew Mansour , Juan José Parra Díaz

Faculté des Sciences de Montpellier

2025/2026
M2 Imagine

Table des matières

1 Préface

Le choix du projet a été spécialisé sur un filtre permettant de manipuler l'age, plutôt que d'essayer de généraliser notre application pour permettre pleins de filtres différents. L'objectif principal de cette semaine a été de trouver, comprendre et implémenter un VAE.

2 Base d'images

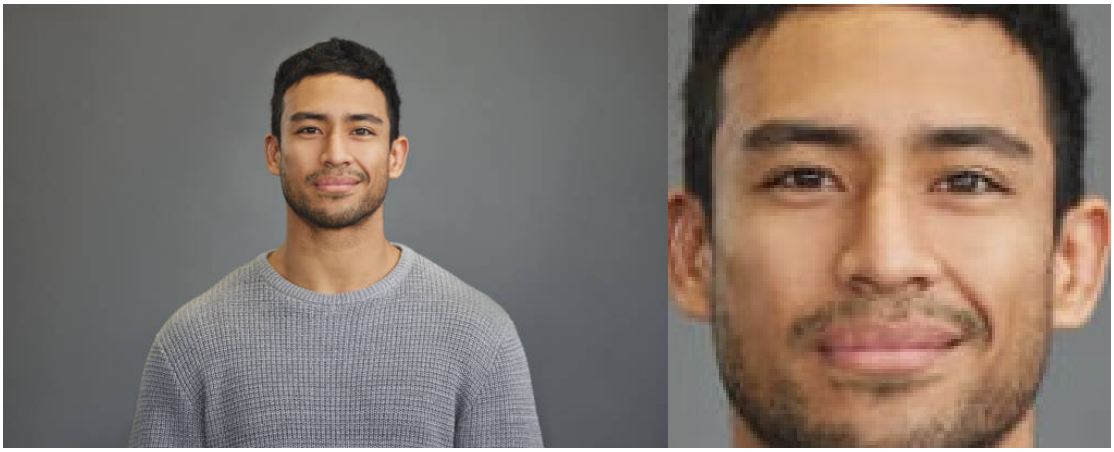
Nous avons choisi la base d'images **UTKFace** (kaggle.com/datasets/jangedoo/utkface-new), qui comprend plus de 20 000 images de visages, toutes annotées avec un age (ainsi que d'autres caractéristiques mais qui nous sont pas tout de suite intéressants).

De plus, ce dernier est présent sur Kaggle permettant l'accès sans télécharger la base localement dans notre code via kagglehub.

FIGURE 1 – Exemples d'images de la base UTKFace

3 Pré-traitements

La base d'images UTKFace contient beaucoup d'images avec des caractéristiques différentes, tels que la taille ou le fond de l'image.
Pour l'apprentissage de notre modèle on souhaite qu'il apprenne uniquement sur les données de visage, et pas perdre de temps de calcul ou d'apprentissage à apprendre les fonds d'une image par exemple. Pour cela, nous avons effectué un premier pré-traitement qui consiste à recadrer l'image uniquement sur le visage en utilisant OpenCV et le classifieur Haar Cascades [1] et les mettre à la bonne résolution (actuellement à 64x64 pour obtenir des résultats rapides mais ce format est beaucoup trop petit pour avoir des résultats cohérents).



(a) Image originale (b) Image après prétraitement

FIGURE 2 – Exemple d'image après prétraitement.

Une prochaine étape de prétraitement possible pourrait être l'alignement des visages au niveau des yeux pour faciliter l'apprentissage malgré les maintes positions et angles présentes dans la base d'images.

4 VAE

Le VAE est un autoencodeur qui, apprennent sur une distribution de données plutôt que des données compressées. Pour créer notre VAE, nous utilisons l'API Keras de Tenserflow, qui va nous permettre de créer facilement un encodeur et un décodeur avec plusieurs couches de convolution, et permettre de calculer la moyenne et la variance de l'espace latent. Ces deux informations vont être utiles pour calculer la "Reconstruction Loss" et "KL Divergence", deux morceaux essentielles à la fonction de perte qu'il faut essayer de minimiser afin d'obtenir les meilleurs résultats.

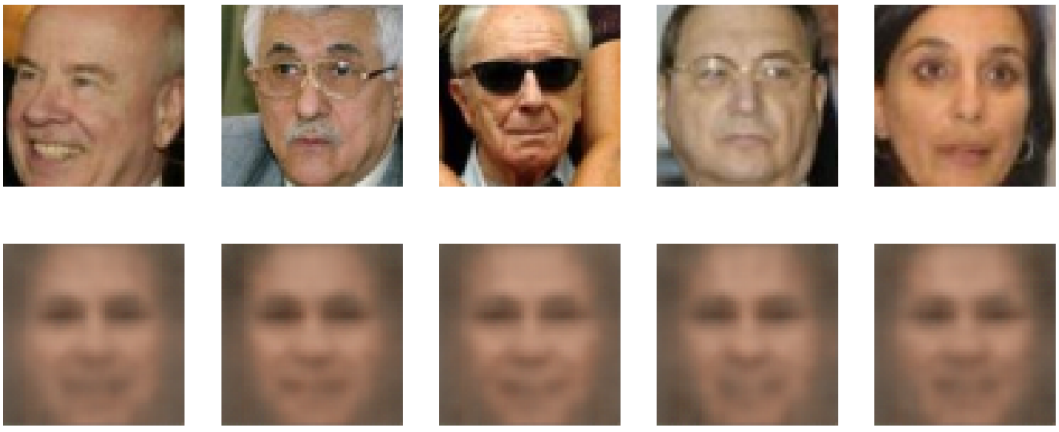


FIGURE 3 – Exemples de résultats obtenus par notre VAE

Les résultats ressemblent à une somme des visages, très similaire peu importe l'image d'entrée. Nous pensons que c'est un comportement naturel sans GAN, car le décodeur va prendre la valeur moyenne de la distribution des données.

Références

[1] "Face detection with python using opencv." https://www.datacamp.com/tutorial/face-detection-python-opencv.