

AI: Deep Learning for Computer Vision



A Data Science Approach

Clément Tamines & Florent Delgrange

UMONS

Faculty of Science

Mab2 Computer Science

Contents

1. Introduction

1.1 Machine learning

1.2 Signal

2. Étude de cas : âge et genre

2.1 Problème

2.2 Résistance aux rotations

2.3 Classification de genre

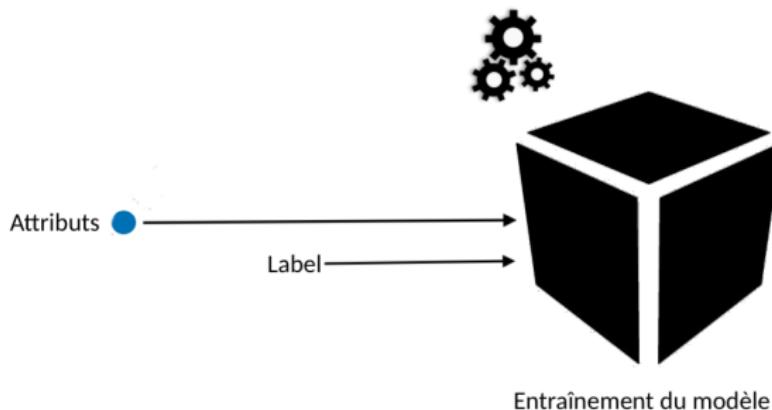
2.4 Régresseur d'âge

2.5 Résultats

Machine learning

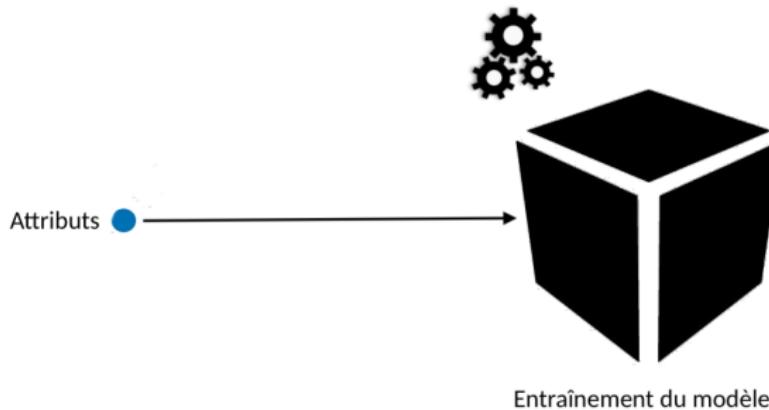
- Capacité d'apprendre à l'aide d'une base de données
 - Big Data
 - Nettoyage des données
 - Préparation des données
 - Entraînement de modèles grâce à ces données
 - Modèles prédictifs
 - classification
exemple : homme/femme ?
 - régression
exemple : âge ?

Entraînement



- **Entraînement supervisé** : le modèle s'entraîne sur base des attributs et des labels **labels** des instances du jeu de données et apprend de cette façon à reconnaître les labels des instances

Entraînement



- **Entraînement supervisé** : le modèle s'entraîne sur base des attributs et des labels **labels** des instances du jeu de données et apprend de cette façon à reconnaître les labels des instances
 - **Entraînement non-supervisé** : le modèle apprend à séparer les données et à les classer uniquement sur base des attributs (**clustering**)

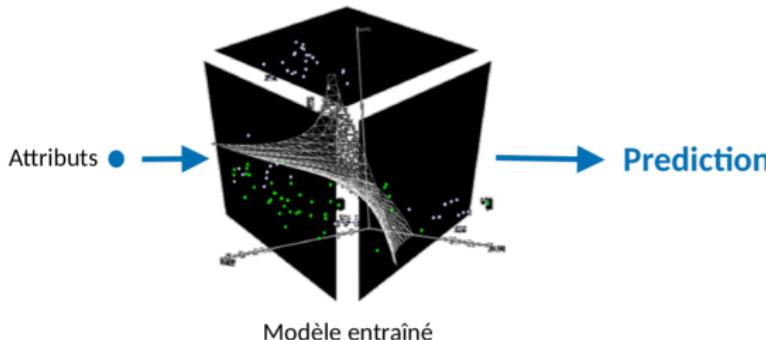
Entraînement

Exemple : Iris database



- Attributs: sepal length, sepal width, petal length, petal width
 - Labels: Iris Virginica, Iris Versicolors, Iris Setosa

Prédictions

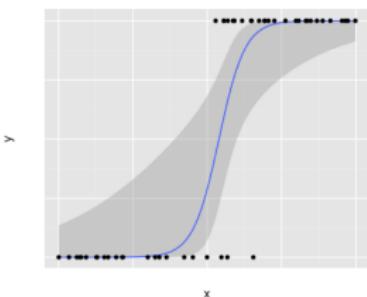


- **Exemple** : selon les longueurs/largeurs des sépales et pétales, prédire l'espèce d'Iris (classifieur)
- **Remarque** : Un régresseur peut être vu comme une fonction

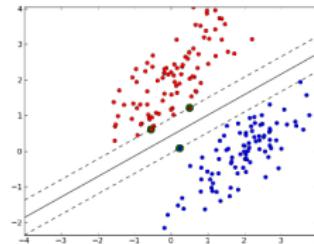
$$f : \mathbb{R}^n \rightarrow \mathbb{R}, x_1, x_2, \dots, x_n \mapsto f(x_1, x_2, \dots, x_n)$$

où $(x_i)_{1 \leq i \leq n}$ est un vecteur de n attributs

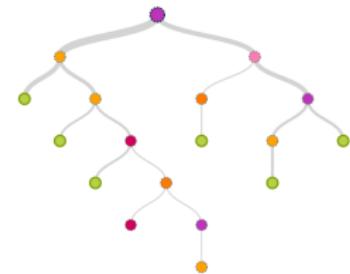
Types de modèles



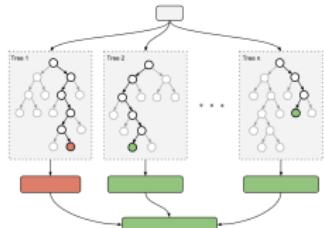
Régression logistique



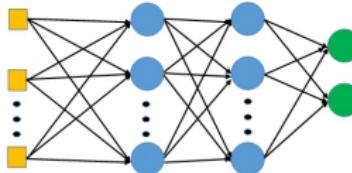
Machine à vecteurs de support



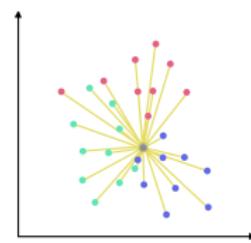
Arbre de décisions (CART)



Forêt aléatoire



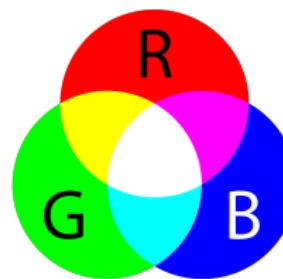
Perceptron à couches multiples (RNN)



Autres (kNN, Naive Bayes,
etc.)

Traitements du signal pour le machine learning

- Une image numérique est un signal discret en 2 dimensions
 - 3 matrices de valeurs de 0 à 255 (RGB)

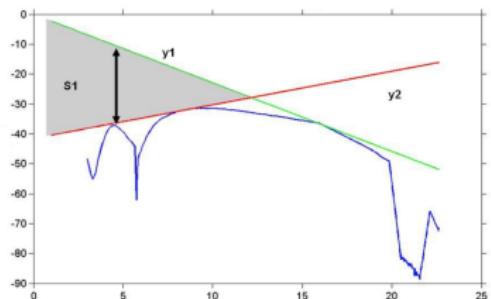
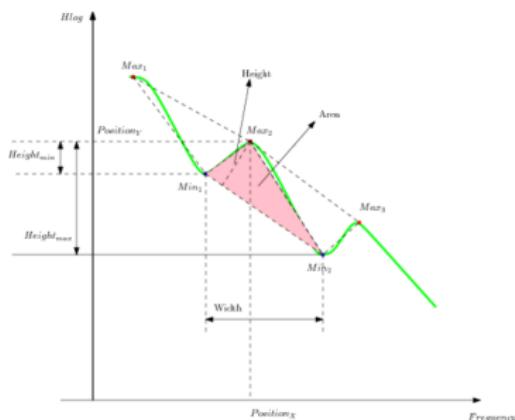


- Comment extraire des attributs du signal ?

Signal

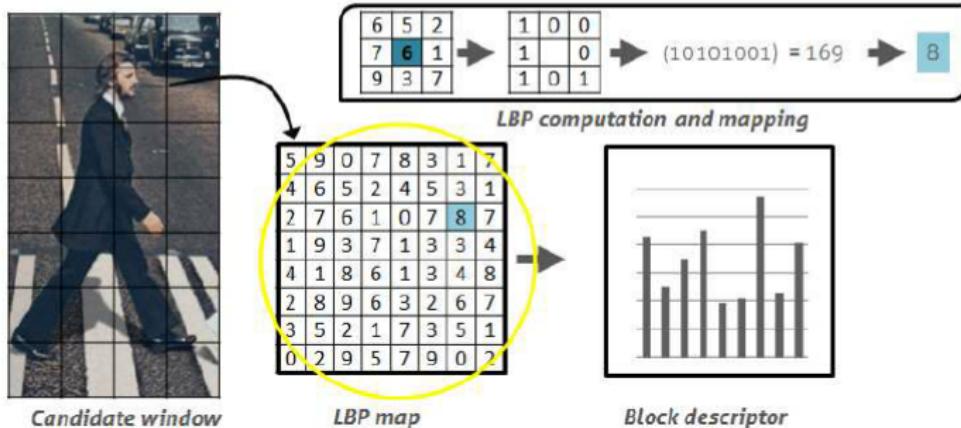
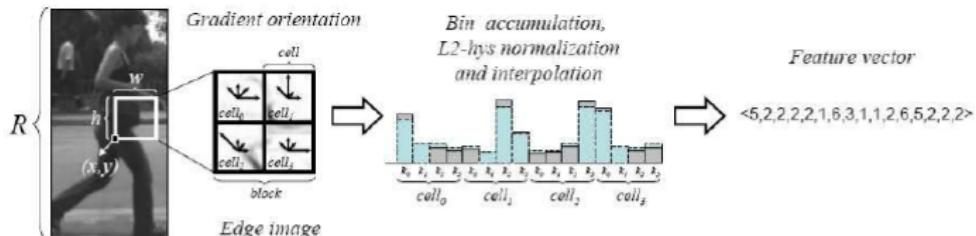
Extraction d'attributs

Méthodes pré-neuronales (1D signal)



Extraction d'attributs

Méthodes pré-neuronales (2D signal)



Contents

1. Introduction

1.1 Machine learning

1.2 Signal

2. Étude de cas : âge et genre

2.1 Problème

2.2 Résistance aux rotations

2.3 Classification de genre

2.4 Régresseur d'âge

2.5 Résultats

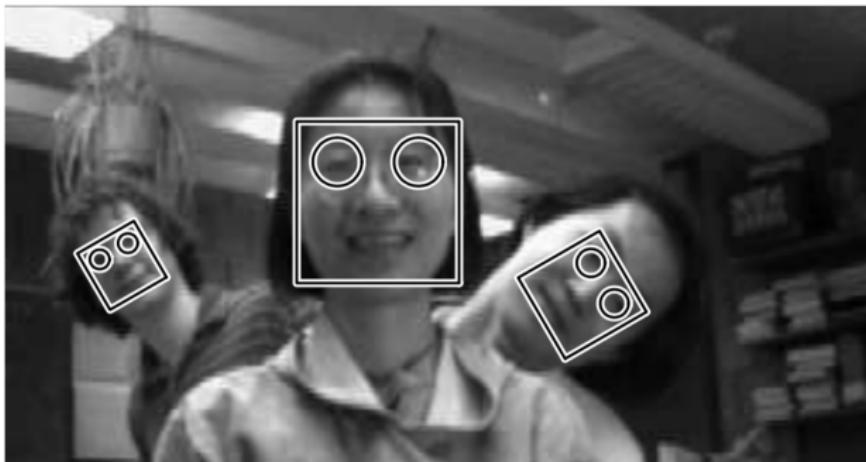
Étude de cas : âge et genre

- **Projet** : créer un réseau de neurones convolutionnel permettant de détecter l'**âge** et le **genre** d'une personne sur une photographie.
- Projet réalisé avec python et le framework Keras (fournissant une API pour Tensorflow)
- **Base de données**: Wikipedia, openu (43390 images en tout)
- **Étapes :**
 1. Régresseur de rotation
 2. Classifieur binaire homme/femme
 3. Régresseur d'âge

Résistance aux rotations

Problématique : le réseau doit reconnaître une personne en position penchée et quelle que soit la rotation du visage.

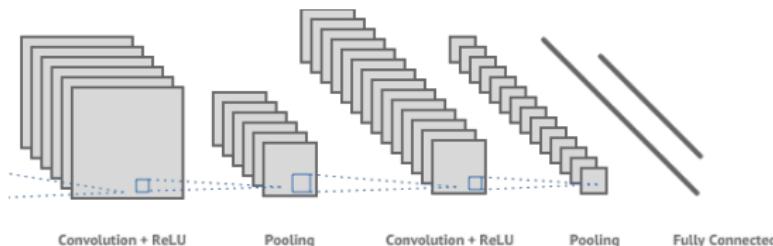
→ l'orientation du visage ne doit pas influencer le résultat !



Régresseur de rotation

- On commence par créer un réseau de neurones convolutionnel jouant le rôle de **régresseur** de rotation

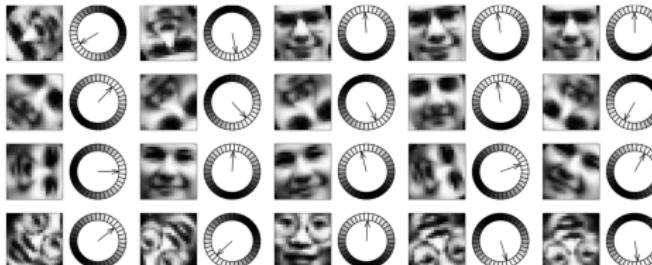
Entrée :
image



Sortie : angle
→ 45.94

Entraînement du régresseur de rotation

- On utilise la base de données Openf: toute les images de la base de données ont été pré-traitées au préalables et sont toutes parfaitement redressées
- On entraîne le réseau en générant une rotation aléatoire pour chaque image, et chaque rotation correspond au label de cette image.



Régresseur de rotation

Résultats

```
img = load_img('test1.png', target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(np.array(x, dtype='uint8'))

angle = model.predict(preprocess_input(np.array([x])))
print("angle={:.2f}".format(angle[0][0]))
x = rotate(x, -1 * angle, mode='nearest', reshape=False)
show(np.array(x, dtype='uint8'))
```



angle=45.948211669921875



```
img = load_img('pdrddy.png', target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(x)

angle = model.predict(preprocess_input(np.array([x])))
print("angle={:.2f}".format(angle[0][0]))
x = rotate(x, -1 * angle, mode='nearest', reshape=False)
show(np.array(x, dtype='uint8'))
```



angle=19.717863082885742



```
img = load_img('test2.png', target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(np.array(x, dtype='uint8'))

angle = model.predict(preprocess_input(np.array([x])))
print("angle={:.2f}".format(angle[0][0]))
x = rotate(x, -1 * angle, mode='nearest', reshape=False)
show(np.array(x, dtype='uint8'))
```

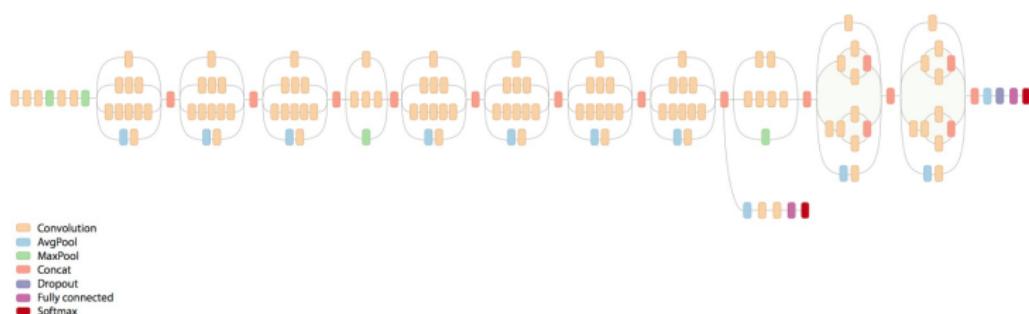


angle=-21.287107467651367



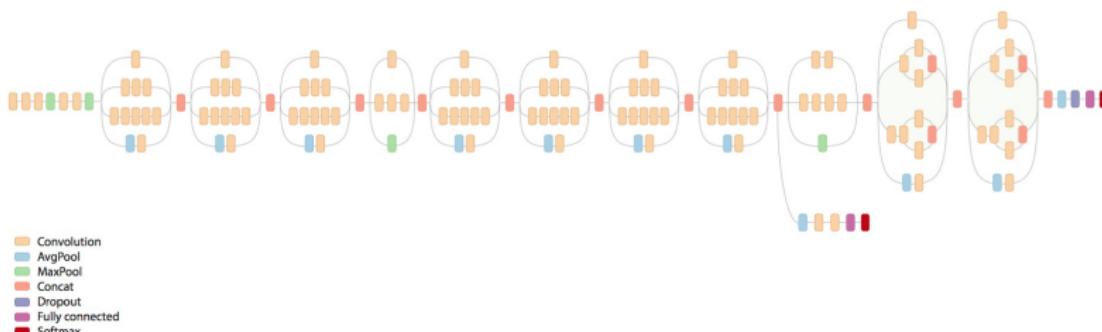
Classifieur homme/femme

- Basé sur le réseau **Xception** de Google
 - profondeur : 126
 - paramètres : 22,910,480
 - originalement entraîné pour reconnaître les images de la base de données **image net** (pas de lien avec notre problème)



Classifieur homme/femme

Apprentissage



- Apprentissage par **transfert**: on utilise le réseau Xception de Google déjà entraîné en “gelant” les 115 premières couches et en entraînant uniquement les couches restantes (top layers) pour notre problème.
- **Affinement du résultat** : on répète l’opération en gelant cette fois la moitié du réseau, mais avec un taux d’apprentissage plus faible

Classifieur homme/femme

Apprentissage

- Le réseau doit être le plus **robuste** possible face aux translations, aux zooms, etc.
 - Le sexe d'une personne ne doit pas changer en fonction de la position de son visage sur l'image
- Entraînement par **augmentation de données**



Classifieur homme/femme

Résultats

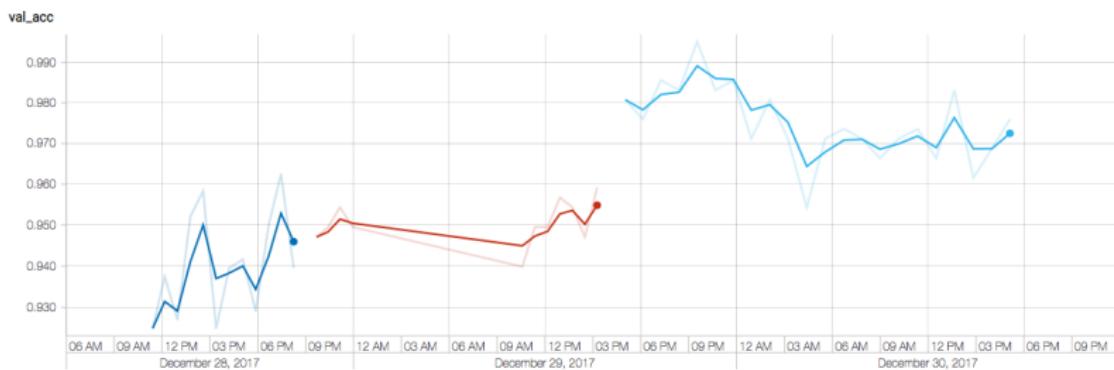
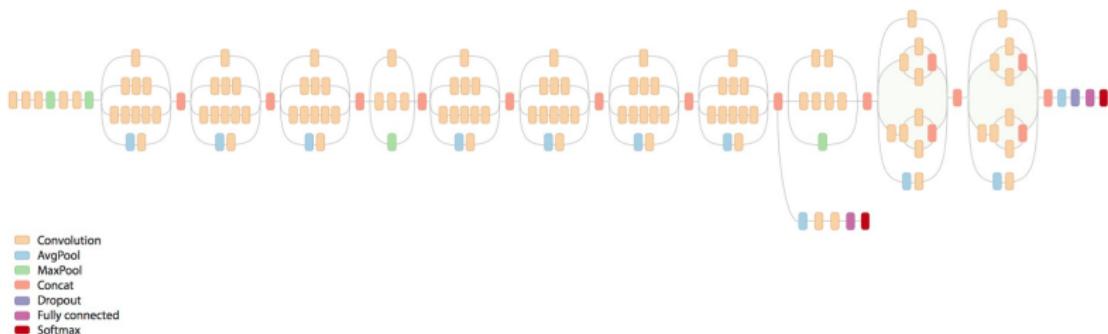


Figure: Entraînement du modèle - en bleu clair : phase d'affinement

- Accuracy : 99.52% (bornée) sur l'ensemble de test

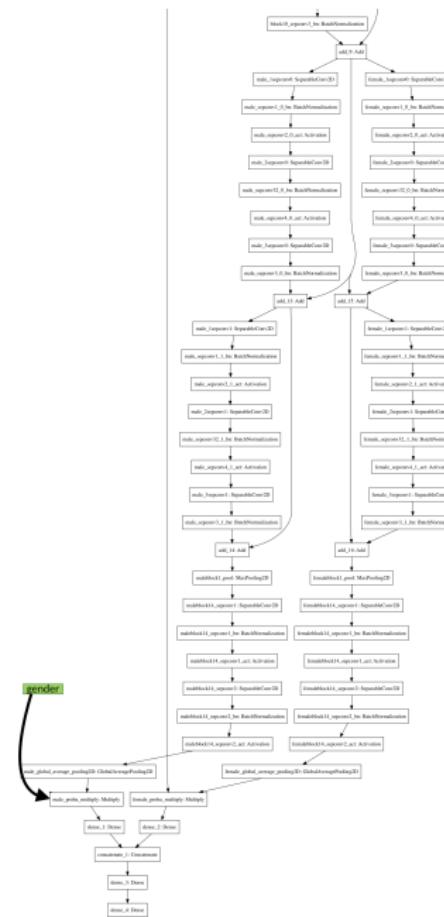
Régresseur d'âge

- Apprentissage par transfert (depuis Xception)
- Entraînement par augmentation de données
- Les 20 dernières couches de convolution sont modifiées
 - les dernières couches de convolution se divisent en 2 en fonction du sexe prédit par le réseau classifieur de genre



Régresseur d'âge

Régresseur d'âge



Régresseur d'âge

Régresseur d'âge

Résultats

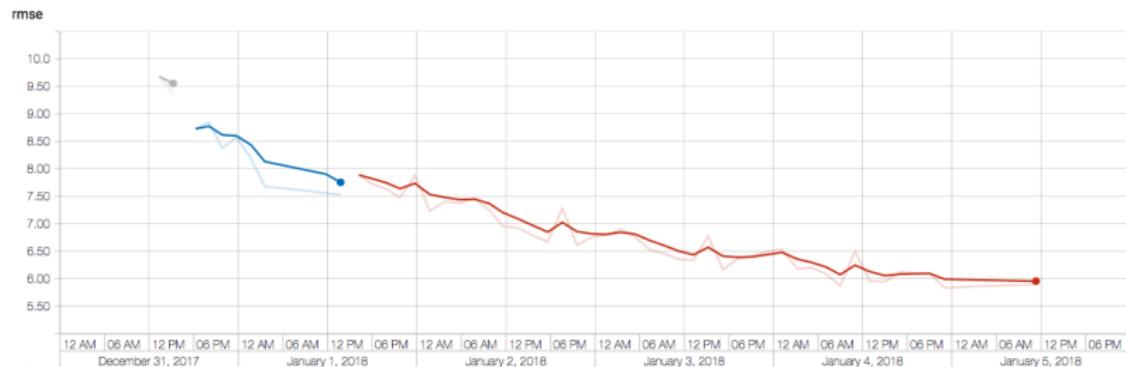


Figure: Entraînement du modèle

- rmse (root mean square error) : 5.88
 - l'âge d'une personne est dans une intervalle ± 5.88 de la prédiction
- Entraînement limité par le hardware

Résultats

Résultats

```
img = load_img('Flo.png',
    target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(x)
print("age : {} "
    .format(model.predict
        (preprocess_input(np.array([x])))))
→ age : [[ 24.6701107]]
```



Résultats

```
img = load_img('clem.png',
                target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(np.array(x, dtype='uint8'))
print("age : {} "
      .format(model.predict(
          preprocess_input(np.array([x])))))
```

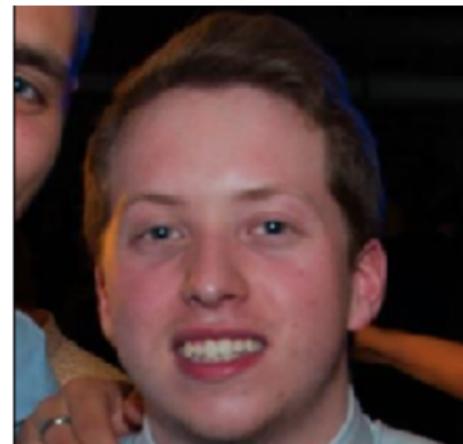
→ age : [[25.11619949]]



Résultats

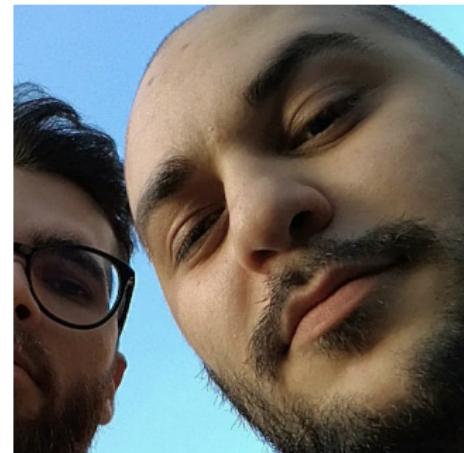
Résultats

```
img = load_img('jerem.png',
    target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(x)
print("age : {} "
    .format(model.predict
        (preprocess_input(np.array([x])))))
→ age : [[24.46559525]]
```



Résultats

```
img = load_img('giorgio-armani.png',
    target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(x)
print("age : {} "
    .format(model.predict
        (preprocess_input(np.array([x])))))
→ age : [[29.9426403]]
```



Résultats

```
img = load_img('gandalf.png',
    target_size=(299, 299))
x = img_to_array(img)
x = np.array(x, dtype='uint8')
show(x)
print("age : {} "
    .format(model.predict
        (preprocess_input(np.array([x])))))
```

→ age : [[90.07471466]]

Résultat réaliste



Résultats

```
img = load_img('JKRowling.png',  
    target_size=(299, 299))  
x = img_to_array(img)  
x = np.array(x, dtype='uint8')  
show(x)  
print("age : {} "  
    .format(model.predict  
        (preprocess_input(np.array([x])))))
```

→ age : [[52.0241394]]

Âge réel : 52 ans !

