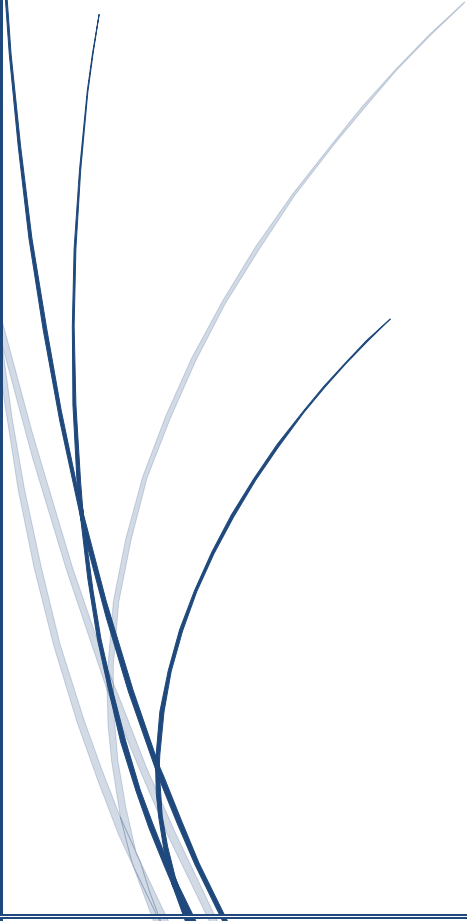




2/15/2021

# AGE-GENDER PREDICTION



Youssra Saadeddine et Ez-zahraouy  
Imane  
ECOLE DES SCIENCES DE L'INFORMATION

## Présentation du Sujet " Gender and Age Classification using CNNs":

L'âge et le sexe, deux des attributs faciaux clés, jouent un rôle très fondamental dans les interactions sociales, faisant de l'estimation de l'âge et du sexe à partir d'une image de visage unique une tâche importante dans les applications intelligentes, telles que le contrôle d'accès, l'interaction homme-ordinateur, l'application de la loi, intelligence marketing et surveillance visuelle, etc.

La prédiction de l'âge à partir d'une seule image n'est pas un problème très facile à résoudre car l'âge perçu dépend de nombreux facteurs et les personnes du même âge peuvent sembler assez différentes dans diverses parties du monde. De plus, les gens essaient très fort de cacher leur âge réel.

### Présentation du DataSet :



Le jeu de données UTKFace est un jeu de données de visage à grande échelle avec une longue période d'âge (de 0 à 116 ans). L'ensemble de données comprend plus de 20 000 images de visage avec des annotations d'âge, de sexe et d'origine ethnique. Les images couvrent de grandes variations de pose, d'expression faciale, d'illumination, d'occlusion, de résolution, etc.

Cet ensemble de données pourrait être utilisé pour une variété de tâches, par exemple, détection de visage, estimation de l'âge, progression / régression d'âge, localisation de points de repère, etc.

#### Points forts

- Se compose de 20k + images de visage dans la nature (un seul visage dans une image)
- Fournit les faces alignées et découpées en conséquence
- Fournit les repères correspondants (68 points)

- Les images sont étiquetées par âge, sexe et origine ethnique

### Samples :



### Labels

The labels of each face image is embedded in the file name, formatted like  
[age]\_[gender]\_[race]\_[date&time].jpg

[age] is an integer from 0 to 116, indicating the age

[gender] is either 0 (male) or 1 (female)

[race] is an integer from 0 to 4, denoting White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern).

[date&time] is in the format of yyyyymmddHHMMSSFFF, showing the date and time an image was collected to UTKFace

**Voici le lien duDataSet:** <https://www.kaggle.com/jangedoo/utkface-new>

Exemples :



## Data Explorer

162.91 MB

### UTKFace

- 100\_0\_0\_2017011221...
- 100\_0\_0\_2017011221...
- 100\_1\_0\_2017011018...
- 100\_1\_0\_2017011221...
- 100\_1\_0\_2017011221...
- 100\_1\_0\_2017011221...
- 100\_1\_0\_2017011719...
- 100\_1\_0\_2017011921...
- 100\_1\_2\_2017010517...
- 100\_1\_2\_2017011221...
- 100\_1\_2\_2017011222...
- 101\_0\_0\_2017011221...
- 101\_1\_2\_2017010517...
- 103\_0\_2\_2017011221...
- 105\_0\_0\_2017011221...
- 105\_1\_0\_2017011221...
- 105\_1\_0\_2017011221...

Hide tree

## < UTKFace (23.7k files)



100\_0\_0\_201701122135...  
7.73 KB



100\_0\_0\_201701122152...  
7.04 KB



100\_1\_0\_201701101837...  
8.81 KB



100\_1\_0\_201701122130...  
7.02 KB



100\_1\_0\_201701122133...  
7.45 KB



100\_1\_0\_201701122150...  
5.01 KB



100\_1\_0\_201701171954...  
5.72 KB



100\_1\_0\_201701192120...  
6.58 KB



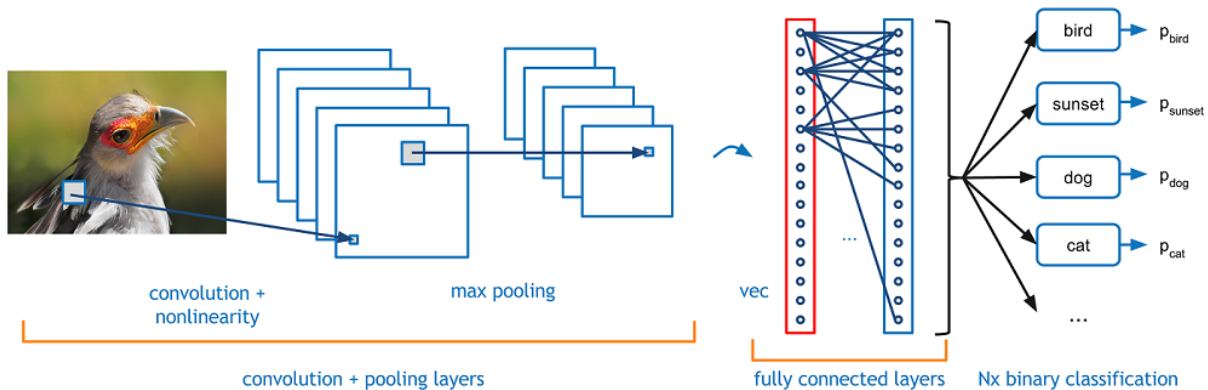
Activate Windows  
Go to Settings to activate Windows.



## CNN :

Les réseaux convolutifs sont une forme particulière de réseau neuronal multicouches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères.

Une convolution est la façon dont l'entrée est modifiée par un filtre. Dans les réseaux convolutifs, plusieurs filtres sont utilisés pour découper l'image et les mapper une par une et apprendre différentes parties d'une image d'entrée.



2012 a été la première année où les réseaux neuronaux ont pris de l'importance alors qu'Alex Krizhevsky les a utilisés pour gagner la compétition ImageNet de cette année (essentiellement, les Jeux olympiques annuels de vision par ordinateur), faisant passer le record d'erreur de classification de 26% à 15%, une amélioration étonnante. Depuis lors, de nombreuses entreprises utilisent le deep learning au cœur de leurs services.

Facebook utilise des réseaux neuronaux pour leurs algorithmes de marquage automatique, Google pour leur recherche de photos, Amazon pour leurs recommandations de produits, Pinterest pour la personnalisation de leur flux d'accueil et Instagram pour leur infrastructure de recherche.



## La classification des images :

La classification des images consiste à prendre une image d'entrée et à produire une classe ou une probabilité de classes qui décrit le mieux l'image. Pour les humains, cette tâche de reconnaissance est l'une des premières compétences que nous apprenons dès notre naissance et en est une qui vient naturellement et sans effort à l'âge adulte. Sans même réfléchir à deux fois, nous sommes en mesure d'identifier rapidement et de manière transparente l'environnement dans lequel nous nous trouvons ainsi que les objets qui nous entourent. Lorsque nous voyons une image ou simplement lorsque nous regardons le monde qui nous entoure, la plupart du temps, nous sommes en mesure de caractériser immédiatement la scène et de donner à chaque objet une étiquette, le tout sans même le remarquer consciemment. Ces compétences pour être capable de reconnaître rapidement des modèles, de généraliser à partir de connaissances antérieures et de s'adapter à différents environnements d'image sont celles que nous ne partageons pas avec nos machines. Lorsqu'un ordinateur voit une image (prend une image en entrée), il voit un tableau de valeurs de pixels. En fonction de la résolution et de la taille de l'image, elle verra un tableau de nombres  $150 \times 150 \times 3$  (le 3 se réfère aux valeurs RVB). Chacun de ces nombres reçoit une valeur de 0 à 255 qui décrit l'intensité du pixel à ce point. Ces chiffres, bien que sans signification pour nous lorsque nous effectuons la classification d'images, sont les seules entrées disponibles pour l'ordinateur. L'idée est que nous donnons à l'ordinateur ce tableau de nombres et il produira des nombres qui décrivent la probabilité que l'image soit une certaine classe. Ce que nous voulons que l'ordinateur fasse, c'est être capable de faire la différence entre toutes les images qu'il donne et de comprendre les caractéristiques. C'est le processus qui se déroule également dans notre esprit inconsciemment. Lorsque nous regardons une photo, nous pouvons la classer comme telle si l'image a des caractéristiques identifiables. De la même manière, l'ordinateur peut effectuer une classification d'images en recherchant des caractéristiques de bas niveau telles que des arêtes et des courbes, puis en développant des concepts plus abstraits grâce à une série de couches convolutives. Ceci est un aperçu général de ce que fait un CNN. Un aperçu plus détaillé de ce que font les CNN serait que vous preniez l'image, la passiez à travers une série de couches convolutives, non linéaires, regroupées (sous-échantillonnage) et entièrement connectées, et obteniez une sortie. Comme nous l'avons dit précédemment, la sortie peut être une classe unique ou une probabilité de classes qui décrit le mieux l'image. Maintenant, le plus difficile est de comprendre ce que font chacune de ces couches. Alors, passons au plus important.

## Tableau récapitulatif de la Revue de littérature :

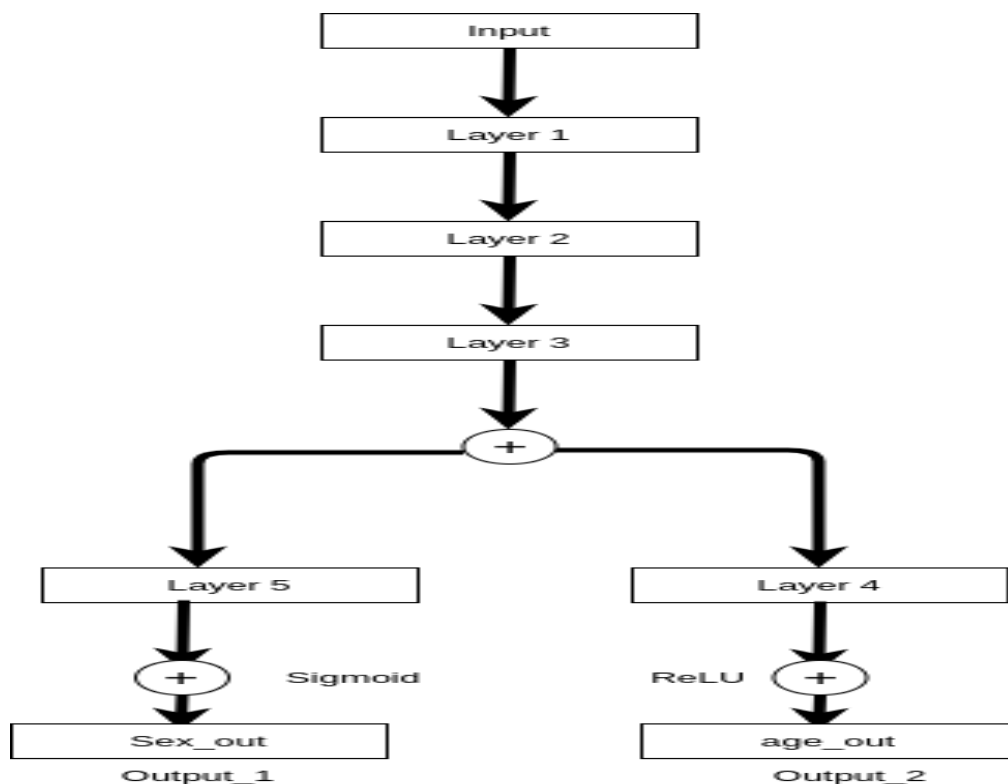
	Ttitre papier	Entraînement ou transfer learning	Main goal	Main results	Modèles utilisés	Perfrmance	
1	Facial Data-Based Deep Learning: Emotion, Age and Gender Prediction	Entrainement learning	predict both age and gender using the same model	classifies female gender better than males	Own architecture CNN model	82%	
2	Deep Convolutional Neural Network for Age Estimation based on VGG-Face Model	Transfer learning	Use a model that was trained initially for face recognition task to predict age	prove that a CNN model, which was trained for face recognition task, can be utilized for age estimation to improve performance	VGG-Face consists of 11 layers, eight convolutional layers and 3 fully connected layers	90.57%	
3	Deeply Learned Classifiers for Age and Gender Predictions of Unfiltered Faces	Entrainement learning	formulate the age and gender classifications task as a classification problem in which the CNN model learns to predict the age and gender from a face image	model obtains the state-of-the-art performance in both age group and gender classification	Own architecture CNN model	96.2%	
4	Understanding and Comparing Deep Neural Networks for Age and Gender Classification	Transfer learning	compare four popular neural network architectures to predict the age and gender from a face image	achieve state of the art performance for gender classification on the Adience benchmark data set	AdienceNet, CaffeNet, Googlenet VGG-16	89% 90.6% 91,7% 92.2%	
5	Age and Gender Classification using Convolutional Neural Networks	Entrainement learning	propose a simple convolutional net architecture that can be used even when the amount of learning data is limited	CNN can be used to provide improved age and gender classification results, even considering the much smaller size of contemporary unconstrained image sets labeled for age and gender	Own architecture CNN model	79.3% for gender & 45.1% for Age	
6	Deep Convolutional Neural Networks and Support Vector Machines for Gender Recognition	Entrainement learning	Explore the applicability of deep convolutional neural networks on face gender recognition and achieve state-ofthe-art	state-ofthe-art classification rates can be achieved using relatively short training times	CNN + Fine tuning + dropout-SVM + oversampling	96.6%	



7	Joint Estimation of Age and Gender from Unconstrained Face Images using Lightweight Multi-task CNN for Mobile Applications	Entrainement learning	An efficient convolutional neural network (CNN) called lightweight multi-task CNN for simultaneous age and gender classification. It uses depthwise separable convolution to reduce the model size and save the inference time.	the LMTCNN can decrease the size of model and speed up the inference on mobile devices while maintaining the accuracy of age and gender classification.	Levi_Hassner CNN [8] LMTCNN-1-1 LMTCNN-2-1	Gender: 82.52% 82.04% 85.16 Age : 44.14 40.84 44.26
8	Multimodal Age and Gender Classification Using Ear and Profile Face Images	Transfer learning(multimodal)	The main objective is to enhance the accuracy of soft biometric trait extraction from profile face images by additionally utilizing a promising biometric modality: ear appearance.	Although age classification performance is better with profile face images, ear images are found to be more useful than profile face images in gender classification. All these results indicate that ear and profile face images contain useful cues for age and gender classification	the pretrained CNN models, namely VGG-16 and ResNet-50	Ear: Age Acc: 60.97% 60.97% Gender Acc: 97.56% 98.00% Profile: Age Acc: 65.73% 62.37% Gender Acc: 95.81% 94.05%
9	A Fusion-based Gender Recognition Method Using Facial Images	Entrainement learning	This article proposes a fusion-based gender recognition method which uses facial images as input. Firstly, it's utilizes pre-processing and a landmark detection method in order to find the important landmarks of faces. Thereafter, four different frameworks are proposed which are inspired by state-of-the-art gender recognition systems	Experimental results show the power and effectiveness of the proposed method. This method obtains recognition rate of 94% for neutral faces of FEI face dataset, which is equal to state-of-the-art rate for this dataset	Local Binary Pattern (LBP) Principal Component Analysis (PCA) Support Vector Machine (SVM) Linear Discriminant Analysis (LDA) F1:LBP+PCA F2:PCA+SVM F3:SVM F4:LDA	F1:78% F2:86% F3:90% F4:86%

10	Age and Gender Classification using Convolutional Neural Networks	Entrainement learning	proposer une architecture réseau convolutive simple qui peut être utilisé même lorsque la quantité de données d'apprentissage est limité.	La Réalisation d' une architecture convolutive simple qui peut être utilisé même lorsque la quantité de données d'apprentissage est limité.	Own Architecture similaire to CaffeNet et AlexNet	86,8 %for gender and 84,7 % for age
----	---	-----------------------	--	--	---	-------------------------------------

### CNN architecture : article 1



### -CNN architecture article 3

L'architecture CNN est un nouveau réseau à six couches, comprenant quatre couches convolutives et deux couches entièrement connectées. La conception CNN est une architecture d'apprentissage profond séquentiel de bout en bout, comprenant des phases d'extraction de caractéristiques et de classification. La phase d'extraction de caractéristiques comporte quatre couches convolutives, avec les paramètres correspondants, notamment le nombre de filtres, la taille du noyau de chaque filtre et la foulée. Il contient la couche convolutive, la couche d'activation (unité linéaire rectifiée

(ReLU)), la normalisation par lots (au lieu de la normalisation de réponse locale conventionnelle), la couche de pooling maximal et une perte (toutes les couches convolutionnelles ont une perte fixe de 25%). L'étape de classification, en revanche, contient deux couches entièrement connectées, qui gèrent la phase de classification du modèle. La première couche entièrement connectée contient 512 neurones, suivis d'un ReLU, puis de la normalisation par lots et, enfin, d'une couche d'abandon à un taux d'abandon de 0,5. La deuxième et la dernière couche entièrement connectée produisent

512 caractéristiques qui sont densément mappées à 8 ou 2 neurones pour des tâches de classification.

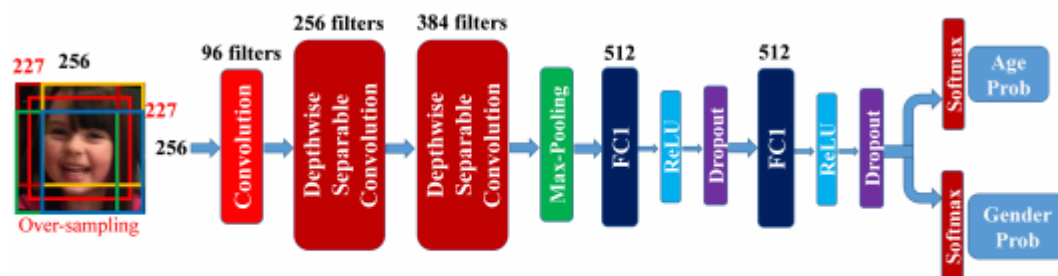
### **-CNN architecture article 5**

1. 96 filtres de taille  $3 \times 7 \times 7$  pixels sont appliqués à l'entrée dans la première couche convolutive, suivie d'un rectifié opérateur linéaire (ReLU), une couche de max pooling prenant la valeur maximale des régions  $3 \times 3$  avec des foulées de deux pixels et une couche de normalisation de réponse locale [28].
  2. La sortie  $96 \times 28 \times 28$  de la couche précédente est alors traité par la deuxième couche convolutive, contenant 256 filtres de taille  $96 \times 5 \times 5$  pixels. Encore une fois, ce est suivi par ReLU, une couche de pooling max et une couche de normalisation de réponse locale avec le même hyper paramètres comme avant.
  3. Enfin, la troisième et dernière couche convolutive fonctionne sur le blob  $256 \times 14 \times 14$  en appliquant un ensemble de 384 filtres de taille  $256 \times 3 \times 3$  pixels, suivis de ReLU et une couche de max pooling.
- Les couches entièrement connectées suivantes sont ensuite définies par:
4. Une première couche entièrement connectée qui reçoit la sortie de la troisième couche convolutive et contient 512 neurones, suivi d'un ReLU et d'une couche de suppression.
  5. Une deuxième couche entièrement connectée qui reçoit le 512-sortie dimensionnelle de la première couche entièrement connectée et contient à nouveau 512 neurones, suivis d'un ReLU et une couche d'abandon.
  6. Une troisième couche entièrement connectée qui correspond à la finale classes d'âge ou de sexe.
- Enfin, la sortie de la dernière couche entièrement connectée est alimentée à une couche soft-max qui attribue une probabilité à chaque classe. La prédiction elle-même est faite en prenant la classe avec le probabilité maximale pour l'image de test donnée

### **-CNN architecture article 6**

Name	Type	Output dimensions	Description
data	Input	$3 \times 227 \times 227$	Mirroring, random crops
conv1	Convolution	$96 \times 55 \times 55$	96 kernels of size 11, stride 4, ReLU
pool1	Max pooling	$96 \times 27 \times 27$	96 kernels of size 3, stride 2, LRN
conv2	Convolution	$256 \times 27 \times 27$	256 kernels of size 5, pad 5, group 2, ReLU
pool2	Max pooling	$256 \times 13 \times 13$	Pool size 3, stride 2, LRN
conv3	Convolution	$384 \times 13 \times 13$	384 kernels of size 3, stride 1, pad 1, group 2, ReLU
conv4	Convolution	$384 \times 13 \times 13$	384 kernels of size 3, stride 1, pad 1, group 2, ReLU
conv5	Convolution	$256 \times 13 \times 13$	256 kernels of size 3, stride 1, pad 1, group 2, ReLU
pool5	Max pooling	$256 \times 6 \times 6$	Pool size 3, stride 2
fc6	Fully connected	$4096 \times 1 \times 1$	ReLU, Dropout
fc7	Fully connected	$4096 \times 1 \times 1$	ReLU, Dropout
fc8	Fully connected	$2 \times 1 \times 1$	Classification with hinge loss

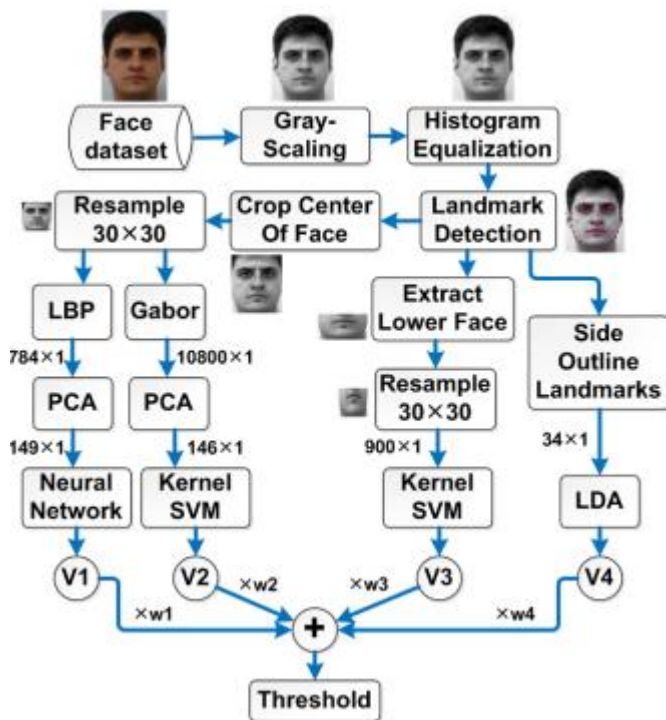
CNN architecture article 7:



**Figure 1. The architecture of the LMTCNN.**

CNN architecture article 9:





**Let's Code:**

## 1. Importing libraries and modules.

Lien vers le notebook : <https://www.kaggle.com/youssrased/ag-ysaadez>

- Importons d'abord tous les paquets keras requis avec lesquels nous allons construire notre CNN
- Il y a deux manières d'utilisation de keras :

1. en utilisant le backend Tensorflow et
2. en utilisant le backend Theano ici, nous utiliserons le backend tensorflow

```
# data visualisation and manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

#model selection
from sklearn.model_selection import train_test_split
```

```

from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, roc_curve, roc_auc_score
from sklearn.preprocessing import LabelEncoder

#preprocess.
from keras.preprocessing.image import ImageDataGenerator

#deep learning librairaies
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam, SGD, Adagrad, Adadelta, RMSprop
from keras.utils import to_categorical

# specifically for cnn
from keras.layers import Dropout, Flatten, Activation
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
import tensorflow as tf
import random as rn

# specifically for manipulating images and getting numpy arrays of pixel values of images.
import cv2
import numpy as np
from tqdm import tqdm
import os
from PIL import Image

```

Voyons maintenant pourquoi des packages ci-dessus est importé :

- nous avons importé Sequential from keras.models, pour initialiser notre modèle de réseau neuronal en tant que réseau séquentiel. Il existe deux méthodes de base pour initialiser un réseau de neurones, soit par une séquence de layers, soit sous forme de graphique.
- nous avons importé Conv2D from keras.layers, c'est pour effectuer l'opération de convolution, c'est-à-dire la première étape d'un CNN, sur les images d'entraînement. Puisque nous travaillons ici sur des images , qui sont essentiellement des tableaux à 2 dimensions.
- nous avons importé MaxPooling2D from keras.layers, qui est utilisé pour l'opération de mise en commun, c'est-à-dire l'étape 2 du processus de création d'un CNN. Pour

construire ce réseau de neurones particulier, nous utilisons une fonction Max pooling, il existe différents types d'opérations de pooling comme Min Pooling, Avg Pooling, etc.

Ici, dans MaxPooling, nous avons besoin du pixel de valeur maximale de la région d'intérêt respective.

- nous avons importé Flatten from keras.layers, qui est utilisé pour Flattening. c'est le processus de conversion de tous les tableaux bidimensionnels résultants en un seul vecteur linéaire long et continu.
- nous avons importé Dense from keras.layers, qui est utilisé pour effectuer la connexion complète du réseau neuronal, qui est l'étape 4 dans le processus de création d'un CNN.

## 2. Load Data

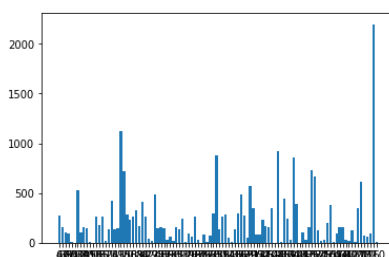
```
[2]: path = "/kaggle/input/utkface-new/UTKFace/"
      files = os.listdir(path)
      size = len(files)
      print("Total samples:", size)
      print(files[0])
```

```
Total samples: 23708
26_0_2_20170104023102422.jpg.chip.jpg
```

Image resize :

```
[3]: import cv2
      images = []
      ages = []
      genders = []
      for file in files:
          image = cv2.imread(path+file,0)
          image = cv2.resize(image, dsize=(64,64))
          image = image.reshape((image.shape[0], image.shape[1], 1))
          images.append(image)
          split_var = file.split('_')
          ages.append(split_var[0])
          genders.append(int(split_var[1]))
```

Visualisation des ages :

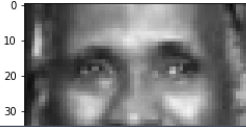


```
Max value: 99
```

Voir une image :

```
[5]: def display(img):
      plt.imshow(img[:, :, 0])
      plt.set_cmap('gray')
      plt.show()
      idx = 500
      sample = images[idx]
      print("Gender:", genders[idx], "Age:", ages[idx])
      display(sample)
```

Gender: 0 Age: 54



Activate Windows  
Go to Settings to activate Windows

Console

Créer une fonction pour grouper les ages :

```
[6]: def age_group(age):
      if age >=0 and age < 18:
          return 1
      elif age < 30:
          return 2
      elif age < 80:
          return 3
      else:
          return 4
```

**Splitting Data :**

**20% pour le test et 80% pour le training**

```
[8]: from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2, shuffle = True)
      print("Samples in Training:", x_train.shape[0])
      print("Samples in Testing:", x_test.shape[0])
```

Samples in Training: 18966  
Samples in Testing: 4742

**Créer CNN model :**

```
[11]: inputs = Input(shape=(64,64,1))
      conv1 = Conv2D(32, kernel_size=(3, 3), activation='relu')(inputs)
      conv2 = Conv2D(64, kernel_size=(3, 3), activation='relu')(conv1)
      pool1 = MaxPooling2D(pool_size=(2, 2))(conv2)
      conv3 = Conv2D(128, kernel_size=(3, 3), activation='relu')(pool1)
      pool2 = MaxPooling2D(pool_size=(2, 2))(conv3)
      conv4 = Conv2D(256, kernel_size=(3,3), activation = 'relu')(pool2)
      conv5 = Conv2D(256, kernel_size=(3,3), activation = 'relu')(conv4)
      flat = Flatten()(conv5)

      age_conv1 = Dense(256, activation='relu')(flat)
      age_conv2 = Dense(128, activation='relu')(age_conv1)
      age_conv3 = Dense(64, activation='relu')(age_conv2)
      age_conv4 = Dense(32, activation='relu')(age_conv3)
      age_model = Dense(1, activation='softmax')(age_conv4)

      gender_conv1 = Dense(256, activation='relu')(flat)
      gender_conv2 = Dense(128, activation='relu')(gender_conv1)
```

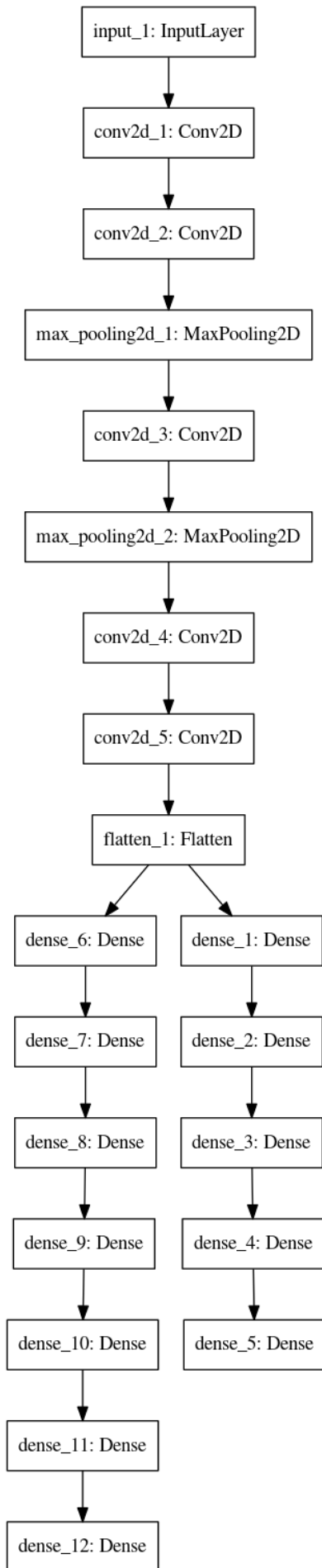
Console



## 5. Compile model

Maintenant que nous avons terminé la création de notre modèle CNN, il est temps de le compiler.

```
[12]: model = Model(inputs=inputs, outputs=[age_model,gender_model])  
      model.compile(optimizer = 'adam', loss = ['mse','binary_crossentropy'],metrics=['accuracy'])
```



## 6. Fit model on training data

- steps\_per\_epoch contient le nombre d'images d'entraînement, c'est-à-dire le nombre d'images que contient le training\_set.
- Epoch, Une Epoch est quand un ensemble de données ENTIER est passé en forward et en backward à travers le réseau neuronal seulement UNE FOIS.
- size batch c'est le nombre total du training examples present dans un single batch.

```
h = model.fit(x_train,[y_train[:,0],y_train[:,1]],  
             validation_data=(x_test,[y_test[:,0],y_test[:,1]]),  
             epochs = 25, batch_size=128,shuffle = True)
```

Train on 18966 samples, validate on 4742 samples

Epoch 1/25

18966/18966 [=====] - 436s 23ms/step - loss: 0.8386 - dense\_5\_loss: 0.2112 - dense\_12\_loss: 0.6273 - dense\_5\_accuracy: 0.0292 - dense\_12\_accuracy: 0.6272 - val\_loss: 0.7047 - val\_dense\_5\_loss: 0.2097 - val\_dense\_12\_loss: 0.4881 - val\_dense\_5\_accuracy: 0.0253 - val\_dense\_12\_accuracy: 0.7796

Epoch 2/25

18966/18966 [=====] - 430s 23ms/step - loss: 0.6075 - dense\_5\_loss: 0.2108 - dense\_12\_loss: 0.3965 - dense\_5\_accuracy: 0.0292 - dense\_12\_accuracy: 0.8141 - val\_loss: 0.5767 - val\_dense\_5\_loss: 0.2097 - val\_dense\_12\_loss: 0.3581 - val\_dense\_5\_accuracy: 0.0253 - val\_dense\_12\_accuracy: 0.8000

le

Activate Windows

La fonction get age et get gender :

```
def get_age(distr):
    distr = distr*4
    if distr >= 0.65 and distr <= 1.4: return "0-18"
    if distr >= 1.65 and distr <= 2.4: return "19-30"
    if distr >= 2.65 and distr <= 3.4: return "31-80"
    if distr >= 3.65 and distr <= 4.4: return "80 +"
    return "Unknown"

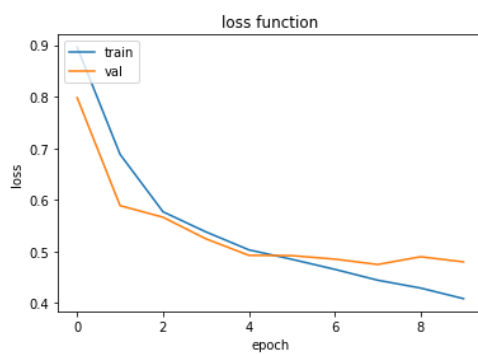
def get_gender(prob):
    if prob < 0.5: return "Male"
    else: return "Female"

def get_result(sample):
    sample = sample/255
    val = model.predict( np.array([sample]) )
    age = get_age(val[0][0])
    gender = get_gender(val[1][0])
    print("Values:", val, "\nPredicted Gender:", gender, "Predicted Age:", age)

indexes = [500,59,80,2,4546,7,9,256,45]
for idx in indexes:
    sample = images[idx]
```

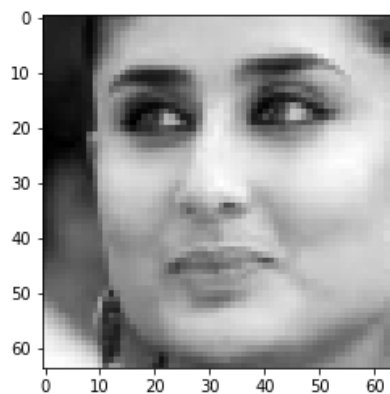
Activate Windows  
Go to Settings to activate Windows.

Visualisation :



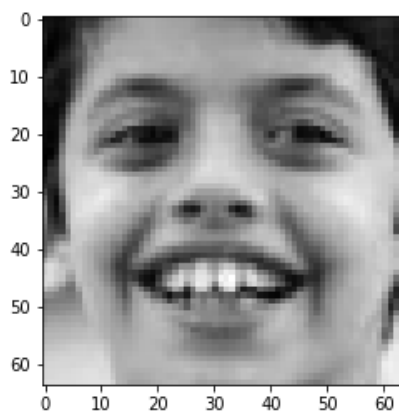
Activa

**Prédiction :**



```
Actual Gender: Female Age: 25  
Values: [array([[1.]], dtype=float32), array([[0.9785446]], dtype=float32)]  
Predicted Gender: Female Predicted Age: 80 +
```

A  
G



```
Actual Gender: Male Age: 7  
Values: [array([[1.]], dtype=float32), array([[0.18846251]], dtype=float32)]  
Predicted Gender: Male Predicted Age: 80 +
```

**On constate qu'il y'a un problème au niveau de prédiction de l'age**



## Model 2 :

Voyons maintenant un autre code pour gender prediction :

Lien vers le code : <https://www.kaggle.com/yousrsasaad/ysaadzah>

Import libraries :



```
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
```

Splitting le nom : le genre se trouve dans la deuxième position dans le nom d'image  
Créer une liste classe qui contient les différents classes

```
[6]: shuffle(onlyfiles)
gender = [i.split('.')[1] for i in onlyfiles]
```

```
[7]: classes = []
for i in gender:
    i = int(i)
    classes.append(i)
```

Lire les images et resize les images et les ajoute dans la liste X\_data

```
[8]: X_data = []
for file in onlyfiles:
    face = misc.imread(file)
    face = cv2.resize(face, (32, 32) )
    X_data.append(face)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
This is separate from the ipykernel package so we can avoid doing imports until
```

Normalize data=> avoir seulement des valeurs entre 0 et 1 pour les pixels

```
[11]: # normalize data
X = X.astype('float32')
X /= 255
```

## Définie 2 categorie et diviser les données entre train set et test set

```
[13]: categorical_labels = to_categorical(classes, num_classes=2)

▶ (x_train, y_train), (x_test, y_test) = (X[:15000], categorical_labels[:15000]), (X[15000:], categorical_labels[15000:])
(x_valid, y_valid) = (x_test[:7000], y_test[:7000])
(x_test, y_test) = (x_test[7000:], y_test[7000:])
```

## Définir maintenant un model CNN :

```
[17]: model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(32,32,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a look at the model summary
model.summary()
```

```
[18]: model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
```

```
[19]: model.fit(x_train,
              y_train,
              batch_size=64,
              epochs=15,
              validation_data=(x_valid, y_valid),)
```

## Créer des labels et évaluer le model

```
[20]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test, verbose=0)

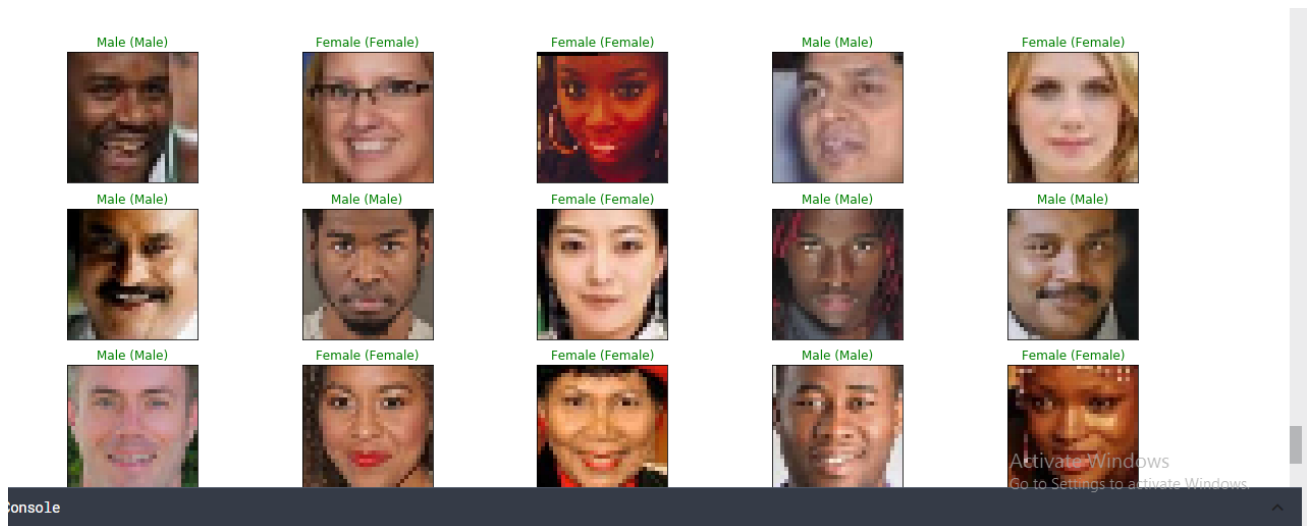
# Print test accuracy
print('\n', 'Test accuracy:', score[1])
```

Test accuracy: 0.88411766

```
[21]: labels = ["Male", # index 0
             "Female", # index 1
             ]
```

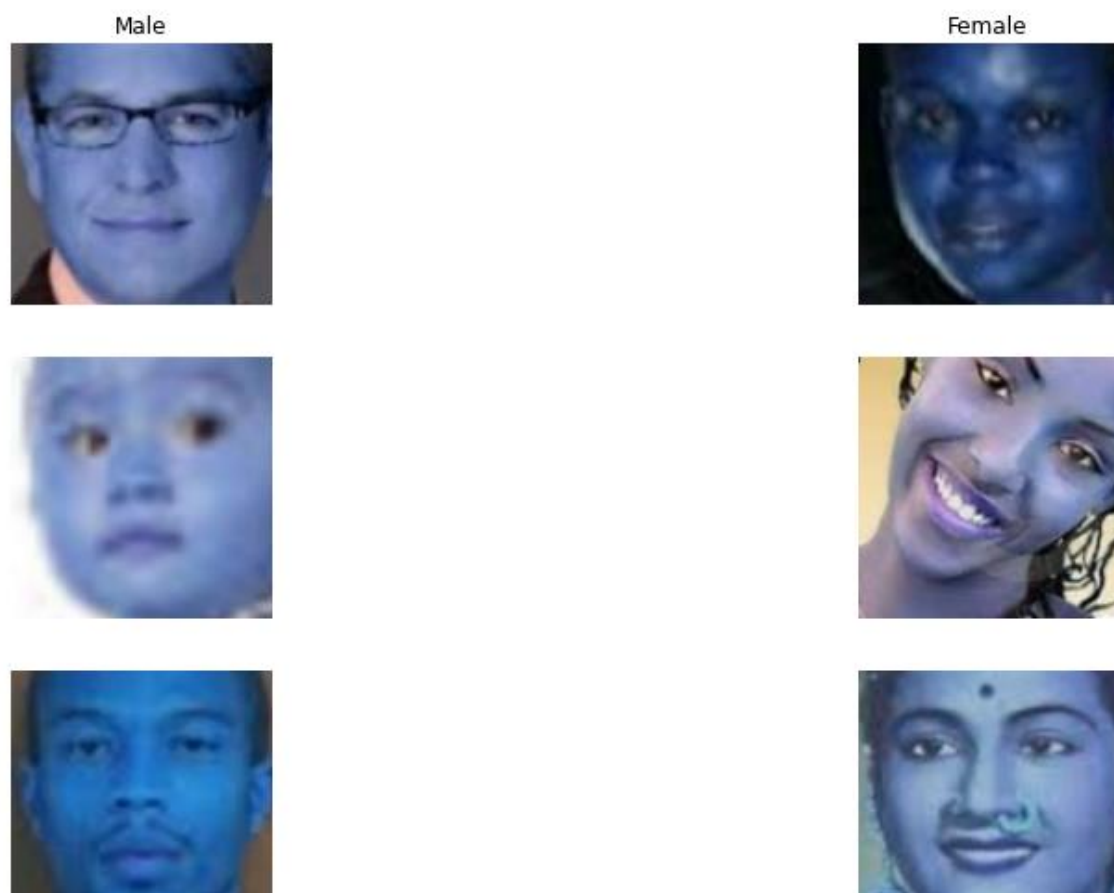
Activate Windows  
Go to Settings to activate Windows

**Result : une performance de 88%**



### Model 3

**Gender Prediction using CNN :** <https://www.kaggle.com/yousrsaad/g-saadez>



```

model = Sequential()

model.add(Conv2D(16, kernel_size = (3, 3), activation = 'relu', input_shape
= (ROW, COL, 3)))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

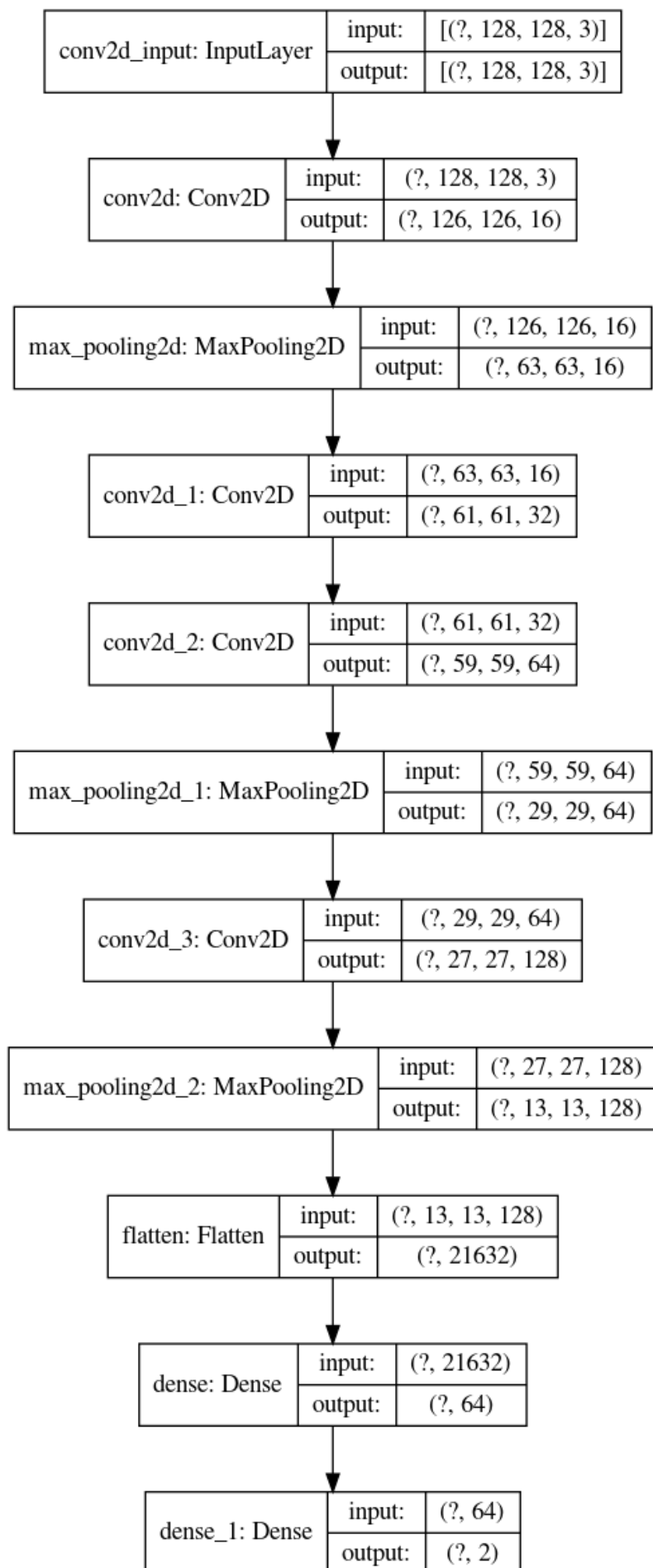
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(2, activation = 'softmax'))
model.summary()

```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 16)	448

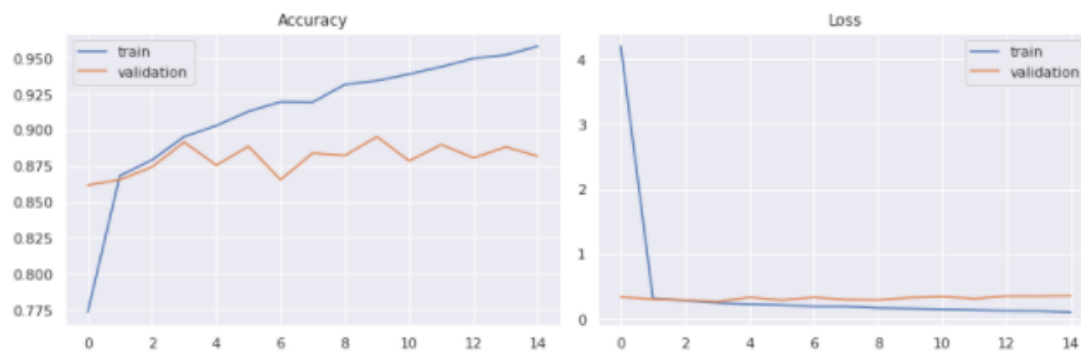
Activate  
Go to Settings





```
sns.lineplot(history.epoch, history.history['loss'], label = 'train')
sns.lineplot(history.epoch, history.history['val_loss'], label = 'validation')
plt.title('Loss')
plt.tight_layout()

#plt.savefig('epoch_history.png')
plt.show()
```



Activ  
Go to !

## Model 4:

**Age Prediction using CNN: 64%** <https://www.kaggle.com/youssrasaad/age-yousim>

```
In [3]: os.chdir('UTKFace')
```

```
In [4]: im = Image.open('1_0_0_20161219140623097.jpg.chip.jpg').resize((128,128))
im
```



```
In [5]: onlyfiles = os.listdir()
```

**Classer les ages => 4 classes à savoir**

```
In [8]: classes = []
        for i in age:
            i = int(i)
            if i <= 14:
                classes.append(0)
            if (i>14) and (i<=25):
                classes.append(1)
            if (i>25) and (i<40):
                classes.append(2)
            if (i>=40) and (i<60):
                classes.append(3)
            if i>=60:
                classes.append(4)
```

## Splitting data:

```
In [16]: (x_train, y_train), (x_test, y_test) = (X[:15008], categorical_labels[:15008]), (X[15008:], categorical_labels[15008:])
        (x_valid, y_valid) = (x_test[:7000], y_test[:7000])
        (x_test, y_test) = (x_test[7000:], y_test[7000:])
```

```
In [17]: len(x_test)
```

```
Out[17]: 1700
```

```
In [18]: len(x_train)+len(x_test) + len(x_valid) == len(X)
```

```
Out[18]: True
```

Activ  
Go to

## Création de model:

```
In [19]: model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=2, padding='same', activation='relu', input_shape=(32,32,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(32,32,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))

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

# Take a look at the model summary
model.summary()
```

A  
Gc

## Compile et fit le model :

```
In [20]: model.compile(loss='categorical_crossentropy',
                        optimizer='adam',
                        metrics=['accuracy'])
```

```
In [21]: model.fit(x_train,
                    y_train,
                    batch_size=100,
                    epochs=25,
                    validation_data=(x_valid, y_valid),)
```

```
Epoch 1/25
151/151 [=====] - 34s 218ms/step - loss: 1.5098
- accuracy: 0.3646 - val_loss: 1.1634 - val_accuracy: 0.5086
Epoch 2/25
151/151 [=====] - 33s 216ms/step - loss: 1.1773
- accuracy: 0.4997 - val_loss: 1.0556 - val_accuracy: 0.5509
Epoch 3/25
```

Act  
Go t

## L'accuracy

```

- accuracy: 0.6739 - val_loss: 0.8267 - val_accuracy: 0.6411
Epoch 25/25
151/151 [=====] - 33s 217ms/step - loss: 0.7287
- accuracy: 0.6853 - val_loss: 0.8251 - val_accuracy: 0.6490

Out[21]:
<tensorflow.python.keras.callbacks.History at 0x7fc0a8e33550>

```

```

In [22]:
# Evaluate the model on test set
score = model.evaluate(x_test, y_test, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score[1])

```

```

Test accuracy: 0.6388235092163086

```

```

In [23]:
labels = ["CHILD", # index 0
          "YOUTH", # index 1
          "ADULT", # index 2

```

Activ.  
Go to 5



Ar  
Go

### Model 5 :

[https://colab.research.google.com/drive/1t\\_AwzIZ\\_j2ajkZEcXZIKCcrOBUSxKqCq?usp=sharing](https://colab.research.google.com/drive/1t_AwzIZ_j2ajkZEcXZIKCcrOBUSxKqCq?usp=sharing)

Age / Gender.net sont des modèles de classification par âge et sexe formés sur l'ensemble de données Audience-OUI.

Published in IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, 2015

Nous aborderons une application intéressante du Deep Learning appliquée aux visages. Nous évaluerons l'âge et déterminerons le sexe de la personne à partir d'une seule image. Le modèle est formé par Gil Levi et Tal Hassner. Nous discuterons

brèvement des principales idées de l'article et fournissons des instructions étape par étape sur la façon d'utiliser le modèle dans OpenCV.

**Les auteurs ont utilisé une architecture de réseau neuronal convolutif très simple, similaire à CaffeNet et AlexNet. Le réseau utilise 3 couches convolutives, 2 couches entièrement connectées et une couche de sortie finale. Les détails des couches sont donnés ci-dessous.**

- ❖ Conv1: La première couche convolutive a 96 nœuds de taille de noyau 7.
- ❖ Conv2: La deuxième couche de conv a 256 nœuds avec une taille de noyau 5.
- ❖ Conv3: La troisième couche de conv a 384 nœuds avec une taille de noyau 3.
- ❖ Les deux couches entièrement connectées ont 512 nœuds chacune.
- ❖ Ils ont utilisé l'ensemble de données Adience pour entraîner le modèle.

```
import cv2
```

```
MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
age_list = ['(0, 2)', '(4, 6)', '(8, 12)', '(15, 20)', '(25, 32)', '(38, 43)', '(48, 53)', '(60, 100)']
gender_list = ['Male', 'Female']
```

```
path_age_pro=r'/gdrive/My Drive/DL/age_deploy.prototxt'
path_age_coffemodel=r'/gdrive/My Drive/DL/age_net.caffemodel'
path_gender_pro=r'/gdrive/My Drive/DL/gender_deploy.prototxt'
path_gender_coffemodel=r'/gdrive/My Drive/DL/gender_net.caffemodel'
path_cascade=r'/gdrive/My Drive/DL/haarcascade_frontalface_alt.xml'
```

```
def initialize_caffe_models():

    age_net = cv2.dnn.readNetFromCaffe(
        path_age_pro,
        path_age_coffemodel)

    gender_net = cv2.dnn.readNetFromCaffe(
        path_gender_pro,
        path_gender_coffemodel)

    return(age_net, gender_net)
```

```

face=r'/gdrive/My Drive/DL/girl2.jpg'
def read_from_image(age_net,gender_net):
    font=cv2.FONT_HERSHEY_SIMPLEX
    image=cv2.imread(face)
    face_cascade=cv2.CascadeClassifier(path_cascade)
    gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    faces=face_cascade.detectMultiscale(gray,1.1,5)
    if(len(faces)>0):
        print("found {} faces".format(str(len(faces))))
        for (x,y,w,h) in faces:
            cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),2)
        #Get Face
        face_img=image[y:y+h,h:h+w].copy()
        blob=cv2.dnn.blobFromImage(face_img,1,(227,227),MODEL_MEAN_VALUES,swapRB=False)
        #Predict Gender
        gender_net.setInput(blob)
        gender_preds=gender_net.forward()
        gender=gender_list[gender_preds[0].argmax]
        print("Gender:"+ gender)
        #Predict Age
        age_net.setInput(blob)
        age_preds=age_net.forward()
        age=age_list[age_preds[0].argmax]
        print("Age Range:"+ age)
        overlay_text="%S %S"%(gender,age)
        cv2.putText(image,overlay_text,(x,y),font,0.5,(100,100,255),2,cv2.LINE_AA)
        cv2.imshow("",image)

```

Activate  
Go to Setting

```

#Predict Age
age_net.setInput(blob)
age_preds=age_net.forward()
age=age_list[age_preds[0].argmax]
print("Age Range:"+ age)
overlay_text="%S %S"%(gender,age)
cv2.putText(image,overlay_text,(x,y),font,0.5,(100,100,255),2,cv2.LINE_AA)
cv2.imshow("",image)
cv2.waitKey(0)

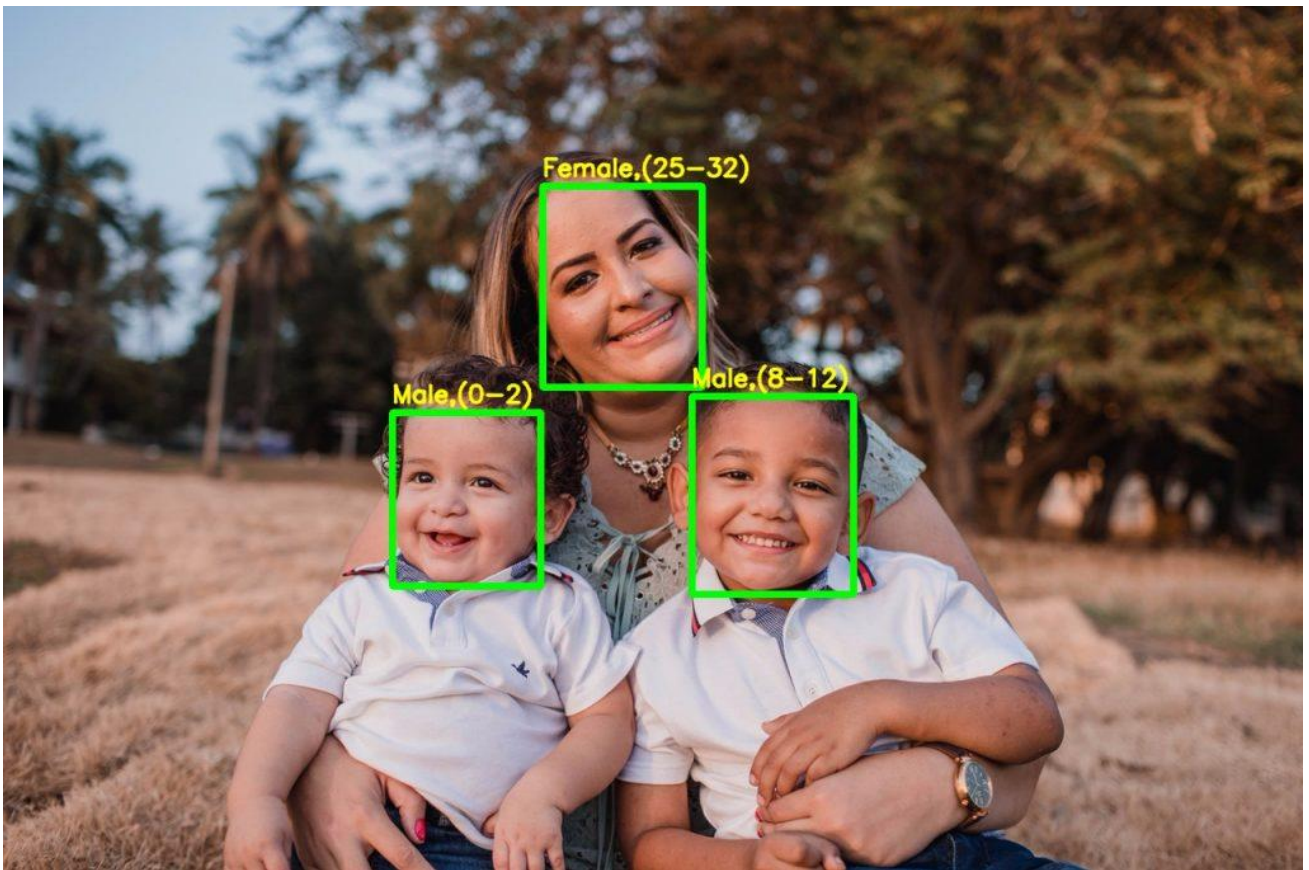
```

```
age_net, gender_net = initialize_caffe_models()
```

```
read_from_image(age_net, gender_net)
```

Activ  
Go to







Apparent age: 16.15  
Gender: Man



Apparent age: 19.07  
Gender: Woman



Apparent age: 28.29  
Gender: Woman



Apparent age: 41.93  
Gender: Man



Apparent age: 15.37  
Gender: Woman



Apparent age: 16.29  
Gender: Woman



Apparent age: 18.21  
Gender: Woman



Apparent age: 33.08  
Gender: Man



Apparent age: 26.22  
Gender: Man



Apparent age: 14.68  
Gender: Man



Apparent age: 49.14  
Gender: Man



Apparent age: 44.67  
Gender: Woman



Activate Windows  
Go to Settings to activate W

## Conclusion

La prédiction du sexe et de l'âge des personnes à partir de leurs images faciales est un problème de recherche continu et actif.

Les chercheurs ont suggéré un certain nombre de méthodes pour résoudre ce problème, mais les critères et les performances réelles sont encore insuffisants. Une approche statistique de reconnaissance de formes pour résoudre ce problème est proposée dans ce projet.

Dans ce projet, des images de visage d'individus ont été entraînées avec des réseaux de neurones convolutifs, et l'âge et le sexe avec un taux de succès élevé ont été prédits nous avons pu atteindre une performance de 99% pour la prédiction du sexe et 64% pour la prédiction de l'âge.