RAMNIRANJAN JHUNJHUNWALA COLLEGE

Department of DSAI

Ghatkopar (West), Mumbai - 400086

2021-2022

Mini-Project Report
On
Age & Gender Detection

In partial fulfillment of M.Sc. (DSAI Sem II)

By

Mr. Manoj H. Yadav

Project Guide

Prof. Bharati Bhole

RAMNIRANJAN JHUNJHUNWALA COLLEGE (AUTONOMOUS)

(Affiliated to University of Mumbai)

GHATKOPAR(WEST), 400086.

Certificate

This is to certify that the Project entitled, "**Age & Gender Detection** "

is bonafide work of **Mr. Manoj H. Yadav** bearing **Seat No: - 43** submitted in partial fulfilment of the requirements for the award of Degree Master of Science in DSAI,

Signature of Internal Guide   Sign of Co-Ordinator

Examiner

Date: College Seal

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher **Prof. BHARATI BHOLE** as well as our principal **Dr. Himanshu Dawda** & R.J college. who gave me the golden opportunity to do this wonderful project on the topic **'Age & Gender Detection'** , which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Abstract

Automatic prediction of age and gender from face images has drawn a lot of attention recently, due it is wide applications in various facial analysis problems. However, due to the large intra-class variation of face images (such as variation in lighting, pose, scale, occlusion), the existing models are still behind the desired accuracy level, which is necessary for the use of these models in real-world applications. In this work, we propose a deep learning framework, based on the ensemble of attentional and residual convolutional networks, to predict gender and age group of facial images with high accuracy rate.

Chapter 1

Introduction

Age and gender information are very important for various real world applications, such as social understanding, biometrics, identity verification, video surveillance, human-computer interaction, electronic customer, crowd behaviour analysis, online advertisement, item recommendation, and many more. Despite their huge applications, being able to automatically predicting age and gender from face images is a very hard problem, mainly due to the various sources of intra-class variations on the facial images of people, which makes the use of these models in real world applications limited.

Dataset :

UTKFace dataset is a large-scale face dataset with long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc. This dataset could be used on a variety of tasks, e.g., face detection, age estimation, age progression/regression, landmark localization, etc.

Chapter II

Problem Define

Here I have used the dataset having 1176 files. It has 1176 images of faces belonging to both males and females with ages ranging from 0 to 116. Each image has labels that show the corresponding age and gender. Male is given by 0 and Female is given by 1.

Tools

Colaboratory, or "**Colab**" for short, is a product from Google Research. f**Colab** allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

What Colab Offers You?

As a programmer, you can perform the following using Google Colab.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

Solution

Code :-

Github path :-

Mounting Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Data Preprocessing

```
fldr="/content/drive/MyDrive/UTKFace"
```

```
import os
files=os.listdir(fldr)
```

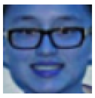
get the data and prepare the training sets. The 'images' list contains all the 1176 images

```
import cv2
ages=[]
genders=[]
images=[]

for fle in files:
  age=int(fle.split('_')[0])
  gender=int(fle.split('_')[1])
  total=fldr+'/'+fle
  print(total)
  image=cv2.imread(total)

  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
  image= cv2.resize(image,(48,48))
  images.append(image)
```

```
/content/drive/MyDrive/UTKFace/15_1_0_20170109214352795.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214319385.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_1_20170104005130400.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214409051.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214307598.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_1_20170112191212510.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214024612.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214723528.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214626752.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170116232438243.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214302271.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_0_20170109214328421.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104221722328.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_2_20170116175234078.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_2_20170104013425867.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_4_20170103200935782.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104222011950.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104222618503.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_2_20161219190855506.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104221641789.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_1_20170112230538604.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_2_20170104012024121.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104221725742.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_2_20170104015856031.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_2_20170104012441969.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_1_20170112230550725.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_1_20170112210325253.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104221933959.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20161220145451968.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_3_20170104222007428.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_4_20170103201247846.jpg.chip.jpg
/content/drive/MyDrive/UTKFace/15_1_1_20170116000638538.jpg.chip.jpg
```

```
for fle in files:
  age=int(fle.split('_')[0])
  gender=int(fle.split('_')[1])
  ages.append(age)
  genders.append(gender)
```

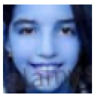
```
[6] from google.colab.patches import cv2_imshow
    cv2_imshow(images[24])
```



```
[7] print(ages[24])
    print(genders[24])
```

```
15
1
```

```
[8] cv2_imshow(images[53])
```



```
[9] print(ages[53])
    print(genders[53])
```

```
15
1
```

```
[10] import numpy as np
     images_f=np.array(images)
     genders_f=np.array(genders)
     ages_f=np.array(ages)
```

```
[11] np.save(fldr+'image.npy',images_f)
     np.save(fldr+'gender.npy',genders_f)
     np.save(fldr+'age.npy',ages_f)
```

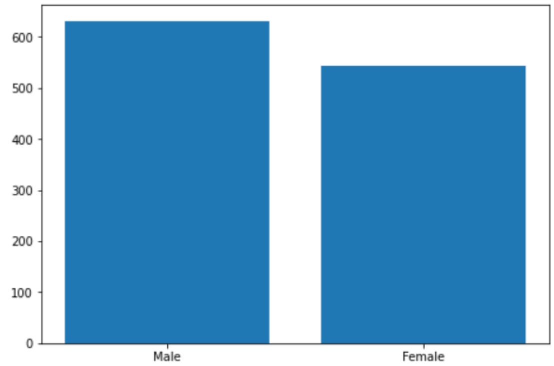
Male = 0 Female= 1

```
[12]
     values, counts = np.unique(genders_f, return_counts=True)
     print(counts)

     [632 544]
```

Now, need to check the distribution of our sets.

The first bar graph shows the distribution of gender. It seems well balanced. The second line graph shows the variation of samples of different ages.

```
[13] import matplotlib.pyplot as plt
     fig = plt.figure()
     ax = fig.add_axes([0,0,1,1])
     gender = ['Male', 'Female']
     values=[632,544]
     ax.bar(gender,values)
     plt.show()
```

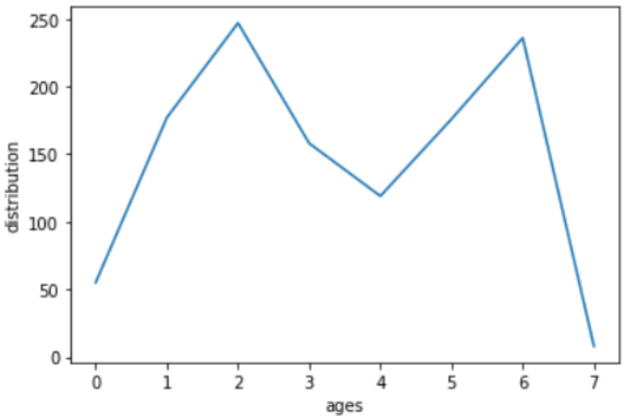


```
[14]
values, counts = np.unique(ages_f, return_counts=True)
print(counts)
```

```
[ 55 177 247 158 119 176 236   8]
```

```
[15] val=values.tolist()
cnt=counts.tolist()
```

```
[16] plt.plot(counts)
plt.xlabel('ages')
plt.ylabel('distribution')
plt.show()
```



The below snippet takes the age and gender for each image sample index wise and converts each one into a list and appends them to the labels list. This is done to create the one-dimensional label vectors. *So, the shape of the 'labels' list will be: *

[[[age(1)],[gender(1)]],

[[age(2)],[gender(2)]], ………………[[age(n)],[gender(n)]]]

```
[17] labels=[]

i=0
while i<len(ages):
  label=[]
  label.append([ages[i]])
  label.append([genders[i]])
  labels.append(label)
  i+=1
```

Next, convert the labels and images list into NumPy arrays, normalize the images, and create the training and test data splits. using a 25% test split.

```
[18] images_f_2=images_f/255
```

```
[19] labels_f=np.array(labels)
```

```
[20] images_f_2.shape
```

```
(1176, 48, 48, 3)
```

```
[21] import tensorflow as tf
     from sklearn.model_selection import train_test_split
```

```
[22] X_train, X_test, Y_train, Y_test= train_test_split(images_f_2, labels_f,test_size=0.25)
```

```
[23] Y_train[0:5]
```

```
array([[[16],
        [ 0]],

       [[55],
        [ 0]],

       [[56],
        [ 0]],

       [[16],
        [ 0]],

       [[18],
        [ 1]]])
```

Y_train[0] denotes the gender labels vector, and Y_train[1] denotes the age labels vector

```
[24] Y_train_2=[Y_train[:,1],Y_train[:,0]]
     Y_test_2=[Y_test[:,1],Y_test[:,0]]
```

```
Y_train_2[0][0:5]
```

```
array([[0],
       [0],
       [0],
       [0],
       [1]])
```

```
[26] Y_train_2[1][0:5]
```

```
array([[16],
       [55],
       [56],
       [16],
       [18]])
```

Model

```python
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten,BatchNormalization
from tensorflow.keras.layers import Dense, MaxPooling2D,Conv2D
from tensorflow.keras.layers import Input,Activation,Add
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
import tensorflow as tf

def Convolution(input_tensor,filters):

    x = Conv2D(filters=filters,kernel_size=(3, 3),padding = 'same',strides=(1, 1),kernel_regularizer=l2(0.001))(input_tensor)
    x = Dropout(0.1)(x)
    x= Activation('relu')(x)

    return x
def model(input_shape):
  inputs = Input((input_shape))

  conv_1= Convolution(inputs,32)
  maxp_1 = MaxPooling2D(pool_size = (2,2)) (conv_1)
  conv_2 = Convolution(maxp_1,64)
  maxp_2 = MaxPooling2D(pool_size = (2, 2)) (conv_2)
  conv_3 = Convolution(maxp_2,128)
  maxp_3 = MaxPooling2D(pool_size = (2, 2)) (conv_3)
  conv_4 = Convolution(maxp_3,256)
  maxp_4 = MaxPooling2D(pool_size = (2, 2)) (conv_4)
  flatten= Flatten() (maxp_4)
  dense_1= Dense(64,activation='relu')(flatten)
  dense_2= Dense(64,activation='relu')(flatten)
  drop_1=Dropout(0.2)(dense_1)
```

```python
  drop_1=Dropout(0.2)(dense_1)
  drop_2=Dropout(0.2)(dense_2)
  output_1= Dense(1,activation="sigmoid",name='sex_out')(drop_1)
  output_2= Dense(1,activation="relu",name='age_out')(drop_2)
  model = Model(inputs=[inputs], outputs=[output_1,output_2])
  model.compile(loss=["binary_crossentropy","mae"], optimizer="Adam",
  metrics=["accuracy"])

  return model
```

```
[28] Model=model((48,48,3))
```

```
[29] Model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 48, 48, 3)]  0           
__________________________________________________________________________________________________
 conv2d (Conv2D)                (None, 48, 48, 32)   896         input_1[0][0]
__________________________________________________________________________________________________
 dropout (Dropout)              (None, 48, 48, 32)   0           conv2d[0][0]
__________________________________________________________________________________________________
 activation (Activation)        (None, 48, 48, 32)   0           dropout[0][0]
__________________________________________________________________________________________________
 max_pooling2d (MaxPooling2D)   (None, 24, 24, 32)   0           activation[0][0]
__________________________________________________________________________________________________
 conv2d_1 (Conv2D)              (None, 24, 24, 64)   18496       max_pooling2d[0][0]
__________________________________________________________________________________________________
 dropout_1 (Dropout)            (None, 24, 24, 64)   0           conv2d_1[0][0]
__________________________________________________________________________________________________
 activation_1 (Activation)      (None, 24, 24, 64)   0           dropout_1[0][0]
__________________________________________________________________________________________________
 max_pooling2d_1 (MaxPooling2D) (None, 12, 12, 64)   0           activation_1[0][0]
__________________________________________________________________________________________________
 conv2d_2 (Conv2D)              (None, 12, 12, 128)  73856       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
 dropout_2 (Dropout)            (None, 12, 12, 128)  0           conv2d_2[0][0]
__________________________________________________________________________________________________
 activation_2 (Activation)      (None, 12, 12, 128)  0           dropout_2[0][0]
__________________________________________________________________________________________________
 max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 128)    0           activation_2[0][0]
__________________________________________________________________________________________________
 conv2d_3 (Conv2D)              (None, 6, 6, 256)    295168      max_pooling2d_2[0][0]
__________________________________________________________________________________________________
 dropout_3 (Dropout)            (None, 6, 6, 256)    0           conv2d_3[0][0]
__________________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
 dropout_3 (Dropout)            (None, 6, 6, 256)    0           conv2d_3[0][0]
__________________________________________________________________________________________________
 activation_3 (Activation)      (None, 6, 6, 256)    0           dropout_3[0][0]
__________________________________________________________________________________________________
 max_pooling2d_3 (MaxPooling2D) (None, 3, 3, 256)    0           activation_3[0][0]
__________________________________________________________________________________________________
 flatten (Flatten)              (None, 2304)         0           max_pooling2d_3[0][0]
__________________________________________________________________________________________________
 dense (Dense)                  (None, 64)           147520      flatten[0][0]
__________________________________________________________________________________________________
 dense_1 (Dense)                (None, 64)           147520      flatten[0][0]
__________________________________________________________________________________________________
 dropout_4 (Dropout)            (None, 64)           0           dense[0][0]
__________________________________________________________________________________________________
 dropout_5 (Dropout)            (None, 64)           0           dense_1[0][0]
__________________________________________________________________________________________________
 sex_out (Dense)                (None, 1)            65          dropout_4[0][0]
__________________________________________________________________________________________________
 age_out (Dense)                (None, 1)            65          dropout_5[0][0]
==================================================================================================
Total params: 683,586
Trainable params: 683,586
Non-trainable params: 0
__________________________________________________________________________________________________
```

Training

```
[30] from tensorflow.keras.callbacks import ModelCheckpoint
     import tensorflow as tf
```

```
[31] fle_s='Age_sex_detection.h5'
     checkpointer = ModelCheckpoint(fle_s, monitor='val_loss',verbose=1,save_best_only=True,save_weights_only=False, mode='auto',save_freq='epoch')
     Early_stop=tf.keras.callbacks.EarlyStopping(patience=75, monitor='val_loss',restore_best_weights=True),
     callback_list=[checkpointer,Early_stop]
```

```
[32] History=Model.fit(X_train,Y_train_2,batch_size=64,validation_data=(X_test,Y_test_2),epochs=500,callbacks=[callback_list])
     14/14 [==============================] - 0s 17ms/step - loss: 4.0458 - sex_out_loss: 0.0282 - age_out_loss: 3.5443 - sex_out_accuracy: 0.9898 - age_out_accuracy: 0.0000e+00 - val_los

     Epoch 00194: val_loss did not improve from 10.51776
     Epoch 195/500
     14/14 [==============================] - 0s 16ms/step - loss: 3.9087 - sex_out_loss: 0.0290 - age_out_loss: 3.4065 - sex_out_accuracy: 0.9921 - age_out_accuracy: 0.0000e+00 - val_los

     Epoch 00195: val_loss did not improve from 10.51776
     Epoch 196/500
     14/14 [==============================] - 0s 16ms/step - loss: 4.0233 - sex_out_loss: 0.0245 - age_out_loss: 3.5257 - sex_out_accuracy: 0.9932 - age_out_accuracy: 0.0000e+00 - val_los

     Epoch 00196: val_loss did not improve from 10.51776
     Epoch 197/500
     14/14 [==============================] - 0s 17ms/step - loss: 3.9205 - sex_out_loss: 0.0209 - age_out_loss: 3.4267 - sex_out_accuracy: 0.9955 - age_out_accuracy: 0.0000e+00 - val_los

     Epoch 00197: val_loss did not improve from 10.51776
     Epoch 198/500
     14/14 [==============================] - 0s 14ms/step - loss: 4.2125 - sex_out_loss: 0.0322 - age_out_loss: 3.7079 - sex_out_accuracy: 0.9921 - age_out_accuracy: 0.0000e+00 - val_los

     Epoch 00198: val_loss did not improve from 10.51776
     Epoch 199/500
     14/14 [==============================] - 0s 15ms/step - loss: 4.0694 - sex_out_loss: 0.0349 - age_out_loss: 3.5621 - sex_out_accuracy: 0.9921 - age_out_accuracy: 0.0000e+00 - val_los

     Epoch 00199: val_loss did not improve from 10.51776
```

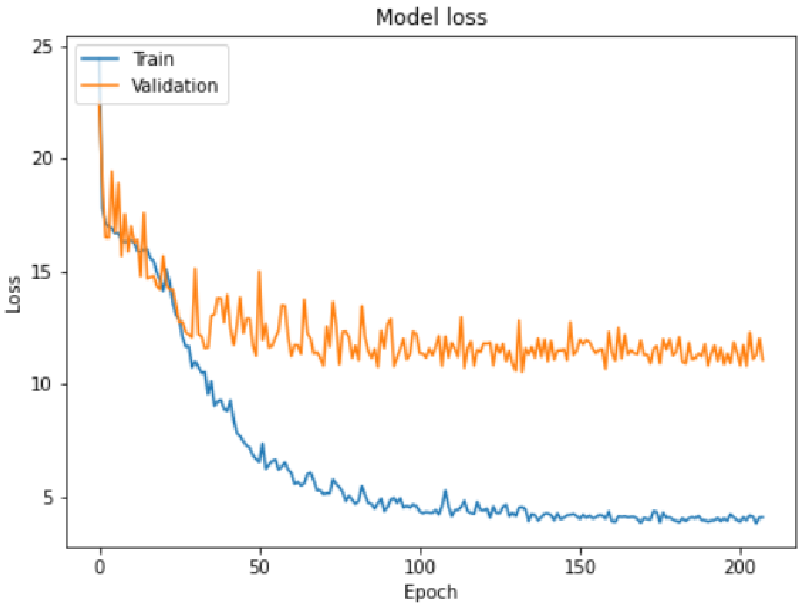
Evaluation

```
[33] Model.evaluate(X_test,Y_test_2)

     10/10 [==============================] - 0s 13ms/step - loss: 10.5178 - sex_out_loss: 0.5334 - age_out_loss: 9.5204 - sex_out_accuracy: 0.8333 - age_out_accuracy: 0.0000e+00
     [10.517757415771484,
      0.5333877801895142,
      9.520434379577637,
      0.8333333134651184,
      0.0]
```

```
[34] pred=Model.predict(X_test)
```

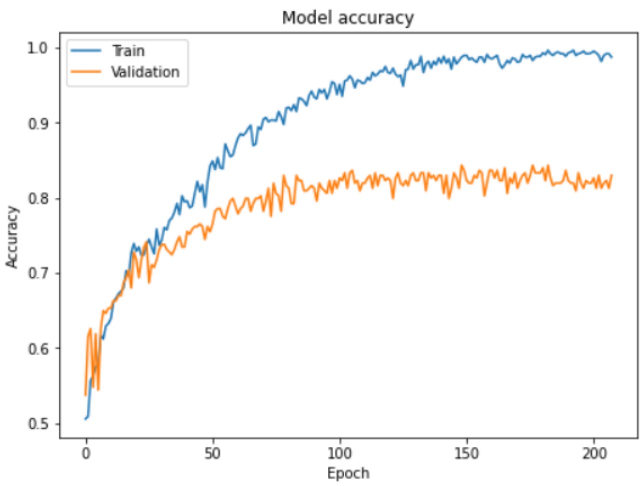
```
[35] pred[1]
            [14.348306 ],
            [16.573887 ],
            [21.406288 ],
            [23.79018  ],
            [14.944648 ],
            [15.425083 ],
            [46.339478 ],
            [13.215786 ],
            [15.617091 ],
            [19.16021  ],
            [16.418028 ],
            [18.371405 ],
            [15.27672  ],
            [27.671713 ],
            [28.537458 ],
            [14.751473 ],
            [13.902635 ],
            [27.589676 ],
            [41.763275 ],
            [11.847379 ],
            [16.185276 ],
```

```
[36] plt.plot(History.history['loss'])
     plt.plot(History.history['val_loss'])
     plt.title('Model loss')
     plt.ylabel('Loss')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Validation'], loc='upper left')
     plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                         wspace=0.35)
```



For Gender

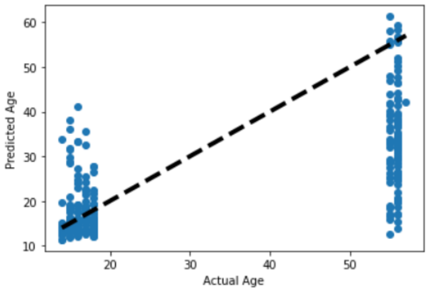
```
[37] plt.plot(History.history['sex_out_accuracy'])
     plt.plot(History.history['val_sex_out_accuracy'])
     plt.title('Model accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel('Epoch')
     plt.legend(['Train', 'Validation'], loc='upper left')
     plt.subplots_adjust(top=1.00, bottom=0.0, left=0.0, right=0.95, hspace=0.25,
                         wspace=0.35)
```



For age

The below curve shows the model traced linear regression line in black and the blue dots show the distribution of test samples.

```
[38] fig, ax = plt.subplots()
     ax.scatter(Y_test_2[1], pred[1])
     ax.plot([Y_test_2[1].min(),Y_test_2[1].max()], [Y_test_2[1].min(), Y_test_2[1].max()], 'k--', lw=4)
     ax.set_xlabel('Actual Age')
     ax.set_ylabel('Predicted Age')
     plt.show()
```



For Gender

```
[39] i=0
     Pred_l=[]
     while(i<len(pred[0])):

       Pred_l.append(int(np.round(pred[0][i])))
       i+=1
```

```
[40] from sklearn.metrics import confusion_matrix

     from sklearn.metrics import classification_report
```

model obtained an F1 score of 0.82 for the female gender and 0.85 for Male gender. So, it classifies male gender better than females.

```
[43]
     report=classification_report(Y_test_2[0], Pred_l)
```

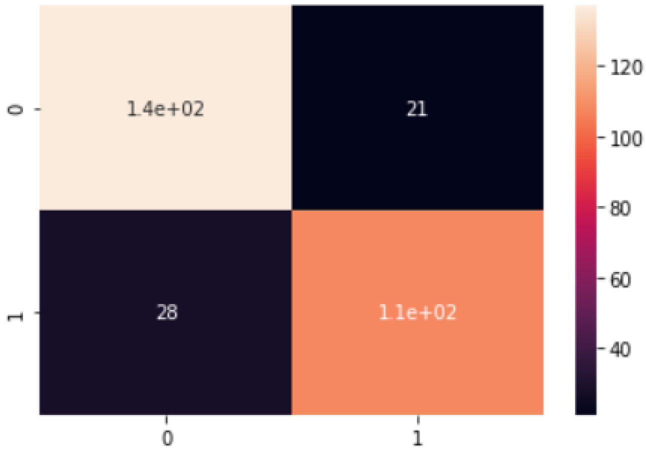
```
[44] print(report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       158
           1       0.84      0.79      0.82       136

    accuracy                           0.83       294
   macro avg       0.83      0.83      0.83       294
weighted avg       0.83      0.83      0.83       294
```

```
[45]
results = confusion_matrix(Y_test_2[0], Pred_l)
```

```
[46] import seaborn as sns

sns.heatmap(results, annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f3c30303c50>

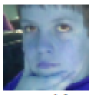


```
[47] def test_image(ind,images_f,images_f_2,Model):
       cv2_imshow(images_f[ind])
       image_test=images_f_2[ind]
       pred_1=Model.predict(np.array([image_test]))
       #print(pred_1)
       sex_f=['Male','Female']
       age=int(np.round(pred_1[1][0]))
       sex=int(np.round(pred_1[0][0]))
       print("Predicted Age: "+ str(age))
       print("Predicted Sex: "+ sex_f[sex])
```

```
[48] test_image(57,images_f,images_f_2,Model)
```

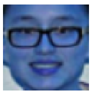
```
Predicted Age: 17
Predicted Sex: Male
```

```
[49] test_image(137,images_f,images_f_2,Model)
```



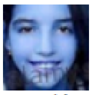
```
Predicted Age: 13
Predicted Sex: Male
```

```
[51] test_image(24,images_f,images_f_2,Model)
```



```
Predicted Age: 19
Predicted Sex: Male
```

```
[52] test_image(53,images_f,images_f_2,Model)
```

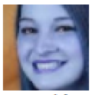


```
Predicted Age: 12
Predicted Sex: Female
```

```
[53] test_image(969,images_f,images_f_2,Model)
```

```
Predicted Age: 31
Predicted Sex: Female
```

```
[54] test_image(551,images_f,images_f_2,Model)
```



```
Predicted Age: 22
Predicted Sex: Female
```

Conclusion

Here , I have came to the end of the project on ‘Age & Gender Detection’ included all the necessary points that are required in the project .

I have completed successfully the project

REFRENCE

https://towardsdatascience.com/facial-data-based-deep-learning-emotion-age-and-gender-prediction-47f2cc1edda7

https://www.kaggle.com/jangedoo/utkface-new

https://learnopencv.com/age-gender-classification-using-opencv-deep-learning-c-python/

PROJECT CODE LINK

https://github.com/Manoj123-github/DSAI/blob/main/Project%20-%20Age_%26_Gender_Prediction.ipynb