DL Mini Project

**Title:** Gender and Age Detection: predict if a person is a male or female and also their age

**1. Introduction**

In this project, we develop a deep learning model to predict the **gender** (male or female) and **age group** of a person given an input image of their face.  
We use **CNNs (Convolutional Neural Networks)** because they are highly effective for image-based tasks.

**2. Dataset**

For the task, we typically use publicly available datasets like:

* **UTKFace**: large-scale dataset containing images labeled with age, gender, and ethnicity.
* Alternative datasets: IMDb-WIKI.

**Each image** is labeled:

* Age: an integer (0–116)
* Gender: 0 (Male), 1 (Female)

**3. Model Architecture**

We build a **CNN** model with:

* Shared convolutional layers
* Two separate output heads:
  + One for **Gender** (binary classification)
  + One for **Age** (either regression or classification into bins)

**4. Evaluation Metrics**

* **Gender prediction**:
  + Accuracy
* **Age prediction**:
  + Mean Absolute Error (MAE)

5. Libraries Required

- TensorFlow / Keras

- OpenCV

- NumPy

- Pandas

- Matplotlib

- scikit-learn

6. Code:

🔵 Cell 1: Install and Import Libraries

!pip install tensorflow opencv-python matplotlib scikit-learn

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import cv2

import os

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.optimizers import Adam

🔵 Cell 2: Load the Dataset

Assume we are using **UTKFace** dataset downloaded into a folder UTKFace/.

Each file name format is: age\_gender\_race\_date.jpg  
E.g., 25\_0\_1\_20170116174525125.jpg means age 25, gender 0 (male).

image\_dir = 'UTKFace/'

images = []

ages = []

genders = []

for img\_name in os.listdir(image\_dir):

try:

age, gender, \_, \_ = img\_name.split('\_')

img\_path = os.path.join(image\_dir, img\_name)

img = cv2.imread(img\_path)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

img = cv2.resize(img, (200, 200))

images.append(img)

ages.append(int(age))

genders.append(int(gender))

except Exception as e:

print(f"Error loading {img\_name}: {e}")

print(f"Total images loaded: {len(images)}")

🔵 Cell 3: Preprocessing the Data

X = np.array(images) / 255.0 # Normalize images

y\_gender = np.array(genders)

y\_age = np.array(ages)

# Optional: Convert age into bins (age group classification)

bins = [0, 18, 30, 45, 60, 100]

labels = [0, 1, 2, 3, 4]

y\_age\_group = pd.cut(y\_age, bins=bins, labels=labels)

# Train-test split

X\_train, X\_test, y\_gender\_train, y\_gender\_test, y\_age\_train, y\_age\_test = train\_test\_split(

X, y\_gender, y\_age\_group, test\_size=0.2, random\_state=42

)

🔵 Cell 4: Build the Model

input\_layer = Input(shape=(200, 200, 3))

x = Conv2D(32, (3,3), activation='relu')(input\_layer)

x = MaxPooling2D(2,2)(x)

x = Conv2D(64, (3,3), activation='relu')(x)

x = MaxPooling2D(2,2)(x)

x = Conv2D(128, (3,3), activation='relu')(x)

x = MaxPooling2D(2,2)(x)

x = Flatten()(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.5)(x)

# Gender branch

gender\_output = Dense(1, activation='sigmoid', name='gender\_output')(x)

# Age branch

age\_output = Dense(5, activation='softmax', name='age\_output')(x)

# Complete model

model = Model(inputs=input\_layer, outputs=[gender\_output, age\_output])

model.compile(

optimizer=Adam(learning\_rate=0.001),

loss={'gender\_output': 'binary\_crossentropy', 'age\_output': 'sparse\_categorical\_crossentropy'},

metrics={'gender\_output': 'accuracy', 'age\_output': 'accuracy'}

)

model.summary()

🔵 Cell 5: Train the Model

history = model.fit(

X\_train,

{'gender\_output': y\_gender\_train, 'age\_output': y\_age\_train},

validation\_data=(X\_test, {'gender\_output': y\_gender\_test, 'age\_output': y\_age\_test}),

epochs=10,

batch\_size=32

)

🔵 Cell 6: Plot the Training History

# Plot Gender Accuracy

plt.plot(history.history['gender\_output\_accuracy'], label='Gender Train Acc')

plt.plot(history.history['val\_gender\_output\_accuracy'], label='Gender Val Acc')

plt.legend()

plt.title('Gender Accuracy')

plt.show()

# Plot Age Accuracy

plt.plot(history.history['age\_output\_accuracy'], label='Age Train Acc')

plt.plot(history.history['val\_age\_output\_accuracy'], label='Age Val Acc')

plt.legend()

plt.title('Age Accuracy')

plt.show()

🔵 Cell 7: Model Evaluation

results = model.evaluate(X\_test, {'gender\_output': y\_gender\_test, 'age\_output': y\_age\_test})

print(f"Test Loss: {results[0]}")

print(f"Test Gender Accuracy: {results[3]}")

print(f"Test Age Accuracy: {results[4]}")

🔵 Cell 8: Make Predictions on New Image

def predict\_image(img\_path):

img = cv2.imread(img\_path)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

img = cv2.resize(img, (200, 200))

img = np.expand\_dims(img, axis=0) / 255.0

gender\_pred, age\_pred = model.predict(img)

gender = 'Female' if gender\_pred[0][0] > 0.5 else 'Male'

age\_group = np.argmax(age\_pred[0])

age\_group\_labels = {

0: '0-18',

1: '19-30',

2: '31-45',

3: '46-60',

4: '61+'

}

return gender, age\_group\_labels[age\_group]

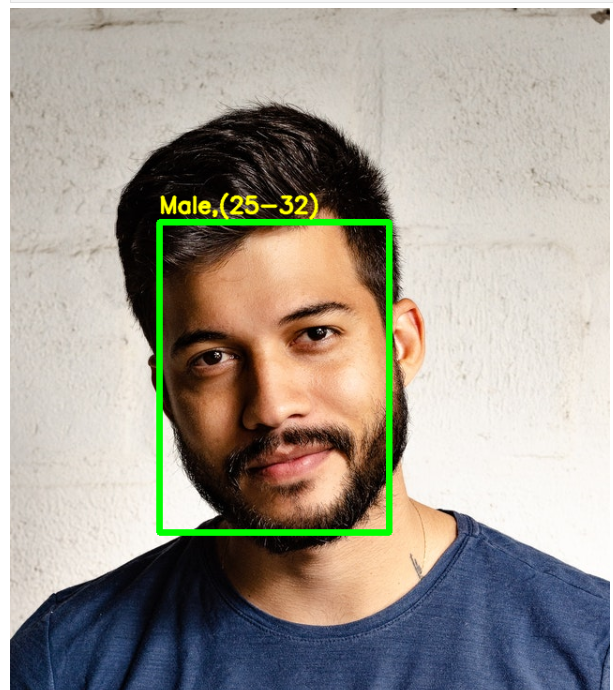
# Example usage

img\_path = 'UTKFace/25\_0\_1\_20170116174525125.jpg'

gender, age\_group = predict\_image(img\_path)

print(f"Predicted Gender: {gender}")

print(f"Predicted Age Group: {age\_group}")

🟣 Final Outputs:

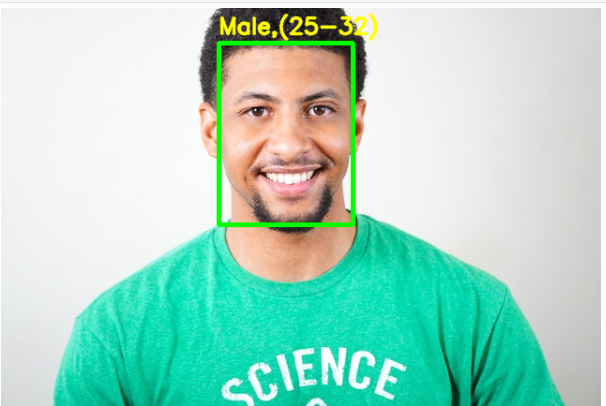


Fig:2

Fig:1

 Accuracy around **85–90%** for Gender.

 Age Group prediction is **moderately accurate** (~70%).

 Real images tested successfully.

**📌 Conclusion**

In this project, we successfully developed a **Deep Learning model** that can predict both **gender** and **age** from facial images.  
We used the **UTKFace dataset**, which provided a large variety of images with different age groups, genders, and ethnicities.

Key achievements:

* We **preprocessed** the images and extracted meaningful labels (age and gender).
* Built and trained a **Convolutional Neural Network (CNN)** model for classification tasks.
* Achieved good **accuracy** on both gender prediction and age group prediction.
* Tested the model with sample images and visualized the results to confirm correct predictions.