A Project Report on

Gender and Age Detection : Predict if a person is a male and female and also their age.

SUBMITTED

BY

RAJ NANASAHEB PATIL
NIRAJ VIKRAM MORE

in partial fulfillment of the requirements for the award of the degree of

Bachelor in
COMPUTER ENGINEERING
For Academic Year 2024-2025

DEPARTMENT OF COMPUTER ENGINEERING
MET's Institute of Engineering Bhujbal Knowledge City
Adgaon, Nashik-422003

THE MET LEAGUE OF COLLEGES
MET Bhujbal Knowledge City
AS SHARP AS YOU CAN GET

CERTIFICATE

This is to Certify that

RAJ NANASAHEB PATIL

NIRAJ VIKRAM MORE

has completed the necessary Mini Project work and prepared the report on

"Gender and Age Detection: Predict if a person is a male or female and also their age"

in satisfactory manner as a fulfillment of the requirement of the award of degree of Bachelor of Computer Engineering in the Academic year 2024-2025

Prof. V. A. Hiray
Subject Teacher

Dr. P. M. Yawalkar
H.O.D

Problem Statement:
Build a deep learning model that detects Gender and age from their facial features using images.

Objective:
We have two main objectives; firstly, we will classify a person into gender by studying the different features of their face. It is a fundamental and binary classification, male and female. Finally, we would also predict the age of that human being by concentrating on the various features of the face and hair color and other such distinctions.

Pre-Requisites:
- Knowledge of Python programming.
- Knowledge of Deep learning algorithms.

Theory:

In recent years, facial analysis has become a prominent area of research in computer vision and artificial intelligence. Among various facial attributes, gender classification and age estimation are two vital tasks with numerous real-world applications, including smart surveillance systems, human-computer interaction, targeted advertising, demographic data collection, and social media filters.

Traditional methods relied heavily on handcrafted features and classical machine learning models, which often lacked accuracy and generalization. With the rise of deep learning, particularly Convolutional Neural Networks (CNNs), we can now automatically learn relevant facial features directly from raw image data, making predictions more robust and reliable.

In this project, we aim to develop a deep learning model using CNN to automatically detect the gender (Male/Female) and estimate the age of individuals based on facial images.

Dataset Used: UTKFace Dataset :

To train and evaluate our model, we used the UTKFace dataset, a large-scale face dataset suitable for age, gender, and ethnicity prediction tasks. This dataset is widely used in facial analysis research due to its diversity and well-annotated labels.

◆ **Key features of the UTKFace dataset:**

- Contains over **20,000 facial images**.
- Images are of varying resolutions and collected in the wild.
- Each image is labeled with:
    - **Age** (ranging from 0 to 116 years),
    - **Gender** (0 = Male, 1 = Female),
    - **Ethnicity** (not used in this project).
- The file naming format is:
  age_gender_race_date.jpg,
  for example: 24_1_2_20170116174525125.jpg

The dataset provides a diverse and balanced set of facial images across different age groups and genders, making it ideal for building and training a CNN-based multi-task model.

Process:

The process of detecting gender and predicting age from facial images using deep learning is a multi-step pipeline that begins with data preparation and ends with prediction and evaluation. At the core of this system lies a Convolutional Neural Network (CNN), which excels at learning spatial hierarchies from visual data.

The first step in this process involves data collection and preprocessing. For this project, the UTKFace dataset is used, which contains over 20,000 images labeled with age, gender, and ethnicity. These images are of varying resolutions and were collected under real-world conditions. Since raw images come in different sizes, they are first resized to a fixed shape (commonly 128x128 pixels with three color channels). The pixel values are normalized — typically scaled between 0 and 1 — to ensure consistent input for the model. Along with this, the age and gender labels are extracted from the filenames of the images. To ensure a fair and unbiased evaluation, the dataset is split into training, validation, and test sets.

Once the data is ready, the model architecture is constructed. A CNN is employed due to its strength in extracting meaningful features from images through the use of filters. The architecture starts with an input layer followed by several convolutional layers, which detect various features like edges, shapes, and textures. These are followed by pooling layers that reduce the dimensionality of the feature maps while retaining essential information. After a series of such layers, the output is flattened into a one-dimensional vector, which is then passed through fully connected dense layers to learn high-level representations. The final layer of the network is split into two branches: one for gender classification and another for age prediction. The gender branch uses a sigmoid activation function to output a probability of the face being male or female, while the age branch either uses a softmax layer (for classification into age groups) or a linear layer (for predicting exact age as a regression problem).

Following the architecture setup, the model is compiled and trained. Appropriate loss functions are chosen for each task — typically binary cross-entropy for gender classification and mean squared error or categorical cross-entropy for age prediction. An optimizer like Adam is used to adjust the model's weights and minimize the loss during training. The model is trained for several epochs, with validation data helping to monitor the model’s performance and prevent overfitting. To make the model more robust, techniques like data augmentation (flipping, rotation, zooming) are applied during training.

After training, the model's performance is evaluated using the test data. Gender prediction accuracy and age prediction error (using metrics like Mean Absolute Error) are calculated. Additionally, visualizations such as loss curves and confusion matrices are used to interpret model behavior and performance. Once the model is finalized, it can take a new facial image as input and accurately predict the individual's gender and age. These predictions can be integrated into a user interface or visualized using image processing libraries like OpenCV or matplotlib.

This end-to-end process highlights the power of deep learning in performing real-time facial analysis and provides a scalable solution for applications requiring demographic inference from images.

Code:

Install Modules

```
%pip install opencv-python
%pip install pandas
%pip install numpy
%pip install matplotlib
%pip install seaborn
%pip install pydot
%pip install graphviz
%pip install tensorflow
%pip install keras
```

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from tqdm.notebook import tqdm
warnings.filterwarnings('ignore')
%matplotlib inline

import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img  # Use tf.keras.
 ↪preprocessing.image
from keras.models import Sequential, Model
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D, Input
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model
```

Load the Dataset

```
!unzip UTKFace.zip
```

```
BASE_DIR = '/content/UTKFace'
```

```
image_paths = []
age_labels = []
gender_labels = []

for filename in tqdm(os.listdir(BASE_DIR)):
```

```
    if filename.startswith('.'):  # Check if filename starts with a dot (hidden files)
        continue  # Skip hidden files like .DS_Store
    image_path = os.path.join(BASE_DIR, filename)
    temp = filename.split('_')
    age = int(temp[0])
    gender = int(temp[1])
    image_paths.append(image_path)
    age_labels.append(age)
    gender_labels.append(gender)
```

```
df = pd.DataFrame()
df['image'], df['age'], df['gender'] = image_paths, age_labels, gender_labels
df.head()
```

```
gender_dict = {0:'Male', 1:'Female'}
```

```
from PIL import Image
img = Image.open(df['image'][10])
plt.axis('off')
plt.imshow(img);
```

```
sns.distplot(df['age'])
```

```
sns.countplot(x='gender', data=df, hue='gender')
plt.xlabel("Gender")
plt.xticks([0, 1], labels=["Male", "Female"])
plt.title("Distribution of Gender")
```

```
plt.figure(figsize=(15, 15))
files = df.iloc[0:25]

for index, file, age, gender in files.itertuples():
    plt.subplot(5, 5, index+1)
    img = load_img(file)
    img = np.array(img)
    plt.imshow(img)
    plt.title(f"Age: {age} Gender: {gender_dict[gender]}")
    plt.axis('off')
```

```
def extract_features(images):
    features = []
    for image in tqdm(images):
        # Load image using PIL and convert to grayscale
        img = Image.open(image).convert('L')
        # Use Image.Resampling.LANCZOS instead of Image.ANTIALIAS
        img = img.resize((128, 128), Image.Resampling.LANCZOS)
        img = np.array(img)
        features.append(img)

    features = np.array(features)
    features = features.reshape(len(features), 128, 128, 1)
```

```
    return features
```

```
X = extract_features(df['image'])
```

```
X.shape
```

```
X = X/255.0
```

```
y_gender = np.array(df['gender'])
y_age = np.array(df['age'])
```

```
input_shape = (128, 128, 1)
```

Model Creation

```
inputs = Input((input_shape))
# convolutional layers
conv_1 = Conv2D(32, kernel_size=(3, 3), activation='relu') (inputs)
maxp_1 = MaxPooling2D(pool_size=(2, 2)) (conv_1)
conv_2 = Conv2D(64, kernel_size=(3, 3), activation='relu') (maxp_1)
maxp_2 = MaxPooling2D(pool_size=(2, 2)) (conv_2)
conv_3 = Conv2D(128, kernel_size=(3, 3), activation='relu') (maxp_2)
maxp_3 = MaxPooling2D(pool_size=(2, 2)) (conv_3)
conv_4 = Conv2D(256, kernel_size=(3, 3), activation='relu') (maxp_3)
maxp_4 = MaxPooling2D(pool_size=(2, 2)) (conv_4)

flatten = Flatten() (maxp_4)
```

```
# fully connected layers
dense_1 = Dense(256, activation='relu') (flatten)
dense_2 = Dense(256, activation='relu') (flatten)

dropout_1 = Dropout(0.3) (dense_1)
dropout_2 = Dropout(0.3) (dense_2)

output_1 = Dense(1, activation='sigmoid', name='gender_out') (dropout_1)
output_2 = Dense(117, activation='softmax', name='age_out') (dropout_2)

model = Model(inputs=[inputs], outputs=[output_1, output_2])

model.compile(
    loss={
        'gender_out': 'binary_crossentropy',
        'age_out': 'sparse_categorical_crossentropy'
```

```
    },
    optimizer='adam',
    metrics={
        'gender_out': 'accuracy',
        'age_out': 'accuracy'
    }
)
```

```
from tensorflow.keras.utils import plot_model
plot_model(model)
```

```
history = model.fit(x=X, y=[y_gender, y_age], batch_size=32, epochs=30,␣
↪validation_split=0.2)
```

Plot the Results

```
acc = history.history['gender_out_accuracy']
val_acc = history.history['val_gender_out_accuracy']
epochs = range(len(acc))
```

```

plt.plot(epochs, acc, 'b', label='Training Accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation Accuracy')
plt.title('Accuracy Graph')
plt.legend()
plt.figure()

loss = history.history['gender_out_loss']
val_loss = history.history['val_gender_out_loss']
```

```
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.plot(epochs, val_loss, 'r', label='Validation Loss')
plt.title('Loss Graph')
plt.legend()
plt.show()
```

```
# plot results for age
loss = history.history['age_out_loss']
val_loss = history.history['val_age_out_loss']
epochs = range(len(loss))

plt.plot(epochs, loss, 'b', label='Training Loss')
plt.plot(epochs, val_loss, 'r', label='Validation Loss')
plt.title('Loss Graph')
plt.legend()
plt.show()
```

```
image_index = 10
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:",␣
 ↪y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = np.argmax(pred[1][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

```
image_index = 300
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:",␣
 ↪y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = np.argmax(pred[1][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

```
image_index = 2500
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:",␣
 ↪y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = np.argmax(pred[1][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

```
image_index = 2567
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:",␣
 ↪y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = np.argmax(pred[1][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

```
image_index = 1577
print("Original Gender:", gender_dict[y_gender[image_index]], "Original Age:",␣
 ↪y_age[image_index])
# predict from model
pred = model.predict(X[image_index].reshape(1, 128, 128, 1))
pred_gender = gender_dict[round(pred[0][0][0])]
pred_age = np.argmax(pred[1][0])
print("Predicted Gender:", pred_gender, "Predicted Age:", pred_age)
plt.axis('off')
plt.imshow(X[image_index].reshape(128, 128), cmap='gray');
```

Output:

```
Original Gender: Male Original Age: 62
1/1             0s 178ms/step
Predicted Gender: Male Predicted Age: 62
```


```
Original Gender: Male Original Age: 45
1/1             0s 55ms/step
Predicted Gender: Male Predicted Age: 45
```

```
Original Gender: Female Original Age: 61
1/1             0s 88ms/step
Predicted Gender: Female Predicted Age: 61
```

```
Original Gender: Male Original Age: 47
1/1             0s 82ms/step
Predicted Gender: Male Predicted Age: 47
```

Conclusion:

In this project, we developed a deep learning model using Convolutional Neural Networks (CNNs) to predict gender and age from facial images. Leveraging the UTKFace dataset, the model successfully classified gender and estimated age with good accuracy. The multi-output architecture effectively handled both tasks simultaneously, showcasing the power of CNNs in facial attribute recognition. While the model performed well, future improvements could focus on refining age predictions and enhancing generalization across diverse face conditions. Overall, this project demonstrates the potential of deep learning in applications like personalized advertising, security, and demographic analysis.