Face_gender_detection

August 23, 2024

[1]:
```
import cv2
import os
import numpy as np
```

[3]:
```
# Directory where images are stored
image_dir = "C:\\Own\\Notes\\facedetec"
# Parameters
target_size = (64, 64)
```

[4]:
```
# Function to extract labels (age and gender) from the filename
def extract_label(filename):
    parts = filename.split('_')
    if len(parts) >= 2:  # Ensure the filename has at least two parts
        age = int(parts[0])  # Age is the first part
        gender = int(parts[1])  # 0 for male, 1 for female
        return age, gender
    else:
        # Handle unexpected filename formats
        raise ValueError(f"Filename {filename} does not match the expected␣
 ↪format.")

# Function to preprocess images
def preprocess_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.resize(image, target_size)
    image = image / 255.0  # Normalize to [0, 1]
    return image
```

[5]:
```
# Lists to store images and labels
images = []
ages = []
genders = []

# Load and preprocess images
for filename in os.listdir(image_dir):
    if filename.endswith(".jpg"):
        try:
            image_path = os.path.join(image_dir, filename)
```

```
            image = preprocess_image(image_path)
            age, gender = extract_label(filename)
            images.append(image)
            ages.append(age)
            genders.append(gender)
        except ValueError as e:
            print(f"Skipping file {filename}: {e}")

# Convert lists to numpy arrays
images = np.array(images)
ages = np.array(ages)
genders = np.array(genders)

print(f"Loaded {len(images)} images.")
```

```
Loaded 24106 images.
```

[6]:
```
from sklearn.model_selection import train_test_split

# Split for gender and age at the same time
X_train, X_test, y_gender_train, y_gender_test, y_age_train, y_age_test =␣
 ↪train_test_split(
    images, genders, ages, test_size=0.15, random_state=42)

# Further split training data into training and validation sets
X_train, X_val, y_gender_train, y_gender_val, y_age_train, y_age_val =␣
 ↪train_test_split(
    X_train, y_gender_train, y_age_train, test_size=0.15, random_state=42)

print(f"Training set (Gender): {len(X_train)} images")
print(f"Validation set (Gender): {len(X_val)} images")
print(f"Test set (Gender): {len(X_test)} images")

print(f"Training set (Age): {len(X_train)} images")
print(f"Validation set (Age): {len(X_val)} images")
print(f"Test set (Age): {len(X_test)} images")
```

```
Training set (Gender): 17416 images
Validation set (Gender): 3074 images
Test set (Gender): 3616 images
Training set (Age): 17416 images
Validation set (Age): 3074 images
Test set (Age): 3616 images
```

[7]:
```
import tensorflow as tf
from tensorflow.keras import layers, models

# Gender Classification Model
```

```python
gender_model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')  # Sigmoid for binary classification
])

gender_model.compile(optimizer='adam',
                     loss='binary_crossentropy',
                     metrics=['accuracy'])

gender_model.summary()

# Train the Gender Model
gender_history = gender_model.fit(X_train, y_gender_train, epochs=10,
                                  validation_data=(X_val, y_gender_val),
                                  batch_size=32)

# Evaluate the Gender Model
gender_test_loss, gender_test_acc = gender_model.evaluate(X_test, y_gender_test)
print(f"Gender Test Accuracy: {gender_test_acc}")
```

```
C:\Users\nabar\AppData\Roaming\Python\Python312\site-
packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73,856

```
 max_pooling2d_2 (MaxPooling2D)    (None, 6, 6, 128)                   0

 flatten (Flatten)                 (None, 4608)                        0

 dense (Dense)                     (None, 128)                   589,952

 dense_1 (Dense)                   (None, 1)                         129


 Total params: 683,329 (2.61 MB)

 Trainable params: 683,329 (2.61 MB)

 Non-trainable params: 0 (0.00 B)

Epoch 1/10
545/545             29s 50ms/step -
accuracy: 0.6208 - loss: 0.6382 - val_accuracy: 0.7245 - val_loss: 0.5353
Epoch 2/10
545/545             27s 49ms/step -
accuracy: 0.7354 - loss: 0.5123 - val_accuracy: 0.7455 - val_loss: 0.5012
Epoch 3/10
545/545             26s 48ms/step -
accuracy: 0.7685 - loss: 0.4688 - val_accuracy: 0.7767 - val_loss: 0.4598
Epoch 4/10
545/545             26s 47ms/step -
accuracy: 0.8015 - loss: 0.4106 - val_accuracy: 0.7851 - val_loss: 0.4467
Epoch 5/10
545/545             26s 48ms/step -
accuracy: 0.8332 - loss: 0.3663 - val_accuracy: 0.7832 - val_loss: 0.4583
Epoch 6/10
545/545             27s 50ms/step -
accuracy: 0.8567 - loss: 0.3177 - val_accuracy: 0.7832 - val_loss: 0.4487
Epoch 7/10
545/545             29s 53ms/step -
accuracy: 0.8846 - loss: 0.2644 - val_accuracy: 0.7909 - val_loss: 0.4558
Epoch 8/10
545/545             31s 57ms/step -
accuracy: 0.9168 - loss: 0.1995 - val_accuracy: 0.7771 - val_loss: 0.5069
Epoch 9/10
545/545             30s 55ms/step -
accuracy: 0.9398 - loss: 0.1484 - val_accuracy: 0.7854 - val_loss: 0.6054
Epoch 10/10
545/545             27s 49ms/step -
accuracy: 0.9576 - loss: 0.1087 - val_accuracy: 0.7800 - val_loss: 0.7649
```

```
113/113             2s 16ms/step -
accuracy: 0.7887 - loss: 0.7861
Gender Test Accuracy: 0.7873340845108032
```

[8]:
```python
# Age Prediction Model
age_model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='linear')  # Linear activation for regression
])

age_model.compile(optimizer='adam',
                  loss='mean_squared_error',
                  metrics=['mae'])

age_model.summary()

# Train the Age Model
age_history = age_model.fit(X_train, y_age_train, epochs=10,
                            validation_data=(X_val, y_age_val),
                            batch_size=32)

# Evaluate the Age Model
age_test_loss, age_test_mae = age_model.evaluate(X_test, y_age_test)
print(f"Age Test MAE: {age_test_mae}")
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_4 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_5 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 128)	0

```
 flatten_1 (Flatten)              (None, 4608)                        0

 dense_2 (Dense)                  (None, 128)                   589,952

 dense_3 (Dense)                  (None, 1)                         129


 Total params: 683,329 (2.61 MB)

 Trainable params: 683,329 (2.61 MB)

 Non-trainable params: 0 (0.00 B)

Epoch 1/10
545/545             28s 49ms/step -
loss: 489.3333 - mae: 17.0137 - val_loss: 393.7560 - val_mae: 14.4393
Epoch 2/10
545/545             29s 52ms/step -
loss: 346.6157 - mae: 14.3457 - val_loss: 335.3572 - val_mae: 13.6060
Epoch 3/10
545/545             26s 48ms/step -
loss: 319.4973 - mae: 13.6960 - val_loss: 318.3306 - val_mae: 14.0178
Epoch 4/10
545/545             27s 49ms/step -
loss: 299.0772 - mae: 13.1789 - val_loss: 327.7289 - val_mae: 12.9445
Epoch 5/10
545/545             28s 51ms/step -
loss: 264.7646 - mae: 12.3388 - val_loss: 273.5220 - val_mae: 12.4717
Epoch 6/10
545/545             26s 48ms/step -
loss: 253.9722 - mae: 12.0075 - val_loss: 275.0884 - val_mae: 12.3510
Epoch 7/10
545/545             28s 51ms/step -
loss: 236.4400 - mae: 11.6336 - val_loss: 306.6012 - val_mae: 12.4704
Epoch 8/10
545/545             29s 53ms/step -
loss: 225.1664 - mae: 11.2472 - val_loss: 261.2545 - val_mae: 12.3088
Epoch 9/10
545/545             29s 54ms/step -
loss: 204.6790 - mae: 10.7438 - val_loss: 249.8689 - val_mae: 11.6112
Epoch 10/10
545/545             27s 49ms/step -
loss: 186.0629 - mae: 10.2460 - val_loss: 240.6633 - val_mae: 11.3291
113/113             2s 16ms/step -
loss: 232.1297 - mae: 11.0115
```

```
Age Test MAE: 11.030682563781738
```

[15]:
```
!pip uninstall opencv-python
!pip install opencv-python-headless
```

```
WARNING: Skipping opencv-python as it is not installed.

Requirement already satisfied: opencv-python-headless in
c:\users\nabar\anaconda3\envs\tensorflow\lib\site-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in
c:\users\nabar\appdata\roaming\python\python312\site-packages (from opencv-
python-headless) (1.26.3)
```

[17]:
```
import tensorflow as tf

# Assuming gender_model and age_model are already defined and trained

# Specify the directory where you want to save the models
save_directory = './saved_models/'

# Save the gender model
gender_model_path = save_directory + 'gender_model'
gender_model.save(gender_model_path)

# Save the age model
age_model_path = save_directory + 'age_model'
age_model.save(age_model_path)

print(f"Gender model saved to {gender_model_path}")
print(f"Age model saved to {age_model_path}")
```

```
Error while closing windows: OpenCV(4.10.0) D:\a\opencv-python\opencv-
python\opencv\modules\highgui\src\window.cpp:1295: error: (-2:Unspecified error)
The function is not implemented. Rebuild the library with Windows, GTK+ 2.x or
Cocoa support. If you are on Ubuntu or Debian, install libgtk2.0-dev and pkg-
config, then re-run cmake or configure script in function 'cvDestroyAllWindows'
```

[2]:
```
from keras.models import load_model
gender_model = load_model("C://Own//Notes//facedetec//facedetecgender_model.
 ↪keras")
age_model = load_model("C://Own//Notes//facedetec//facedetecage_model.keras")
```

[3]:
```
import cv2
import numpy as np

# Load the pre-trained face detector from OpenCV
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +␣
 ↪'haarcascade_frontalface_default.xml')
```

```
# Load your trained gender model
gender_model

# Placeholder for loading the age estimation model (ensure this is done before␣
 ↪running the code)
# Replace with the actual loading of your pre-trained age model
# age_model = ...

# Function to detect faces and predict gender and age
def detect_and_predict(frame, gender_model, age_model=None):
    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the frame
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,␣
 ↪minNeighbors=5, minSize=(60, 60))

    # Loop over the detected faces
    for (x, y, w, h) in faces:
        # Extract the face ROI and resize it to the target size
        face = frame[y:y+h, x:x+w]
        face_resized = cv2.resize(face, (64, 64))
        face_normalized = face_resized / 255.0
        face_reshaped = np.reshape(face_normalized, (1, 64, 64, 3))

        # Predict gender
        gender_prediction = gender_model.predict(face_reshaped)
        gender_label = "Male" if gender_prediction[0] < 0.5 else "Female"

        # Predict age if an age model is available
        if age_model:
            age_prediction = age_model.predict(face_reshaped)
            age_label = int(age_prediction[0])  # Assuming the age model␣
 ↪outputs an age value
        else:
            age_label = "Unknown"

        # Display the label and bounding box on the output frame
        label = f"{gender_label}, Age: {age_label}"
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame, label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,␣
 ↪255, 0), 2)

    return frame

# Start the webcam feed
```

```
cap = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Detect faces and predict gender and age
    frame = detect_and_predict(frame, gender_model, age_model)

    # Display the resulting frame
    cv2.imshow('Gender and Age Detection', frame)

    # Break the loop on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture and close windows
cap.release()
cv2.destroyAllWindows()
```

```
1/1             0s 227ms/step
1/1             0s 93ms/step
1/1             0s 25ms/step
1/1             0s 26ms/step
```

```
C:\Users\nabar\AppData\Local\Temp\ipykernel_8684\2565854014.py:37:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  age_label = int(age_prediction[0])  # Assuming the age model outputs an age
value
```

```
1/1             0s 26ms/step
1/1             0s 25ms/step
1/1             0s 25ms/step
1/1             0s 24ms/step
1/1             0s 29ms/step
1/1             0s 29ms/step
1/1             0s 28ms/step
1/1             0s 27ms/step
1/1             0s 28ms/step
1/1             0s 22ms/step
1/1             0s 26ms/step
1/1             0s 22ms/step
1/1             0s 26ms/step
1/1             0s 19ms/step
1/1             0s 21ms/step
1/1             0s 21ms/step
1/1             0s 26ms/step
```

```
1/1             0s 35ms/step
1/1             0s 37ms/step
1/1             0s 27ms/step
1/1             0s 28ms/step
1/1             0s 26ms/step
1/1             0s 27ms/step
1/1             0s 25ms/step
1/1             0s 31ms/step
1/1             0s 41ms/step
1/1             0s 30ms/step
1/1             0s 32ms/step
1/1             0s 31ms/step
1/1             0s 28ms/step
1/1             0s 18ms/step
1/1             0s 28ms/step
1/1             0s 31ms/step
1/1             0s 24ms/step
1/1             0s 34ms/step
1/1             0s 34ms/step
1/1             0s 30ms/step
1/1             0s 28ms/step
1/1             0s 24ms/step
1/1             0s 28ms/step
1/1             0s 26ms/step
1/1             0s 28ms/step
1/1             0s 29ms/step
1/1             0s 32ms/step
1/1             0s 36ms/step
1/1             0s 37ms/step
1/1             0s 38ms/step
1/1             0s 37ms/step
1/1             0s 48ms/step
1/1             0s 29ms/step
1/1             0s 33ms/step
1/1             0s 31ms/step
1/1             0s 38ms/step
1/1             0s 38ms/step
1/1             0s 32ms/step
1/1             0s 31ms/step
1/1             0s 34ms/step
1/1             0s 30ms/step
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[3], line 62
     59     cv2.imshow('Gender and Age Detection', frame)
     61     # Break the loop on 'q' key press
```

```
---> 62     if cv2.waitKey(1) & 0xFF == ord('q'):
     63         break
     65 # Release the capture and close windows

KeyboardInterrupt:
```

[ ]:

[ ]: