Q1

In [1]:
```
import os
import numpy as np
import random
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
import cv2
```

In [2]:
```
def load_images_from_directory(directory, limit=None):
    images = []
    labels = []
    count = 0
    for filename in os.listdir(directory):
        if count == limit:
            break
        if filename.endswith(".jpg") or filename.endswith(".png"):
            img_path = os.path.join(directory, filename)
            img = cv2.imread(img_path)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            images.append(img)
            labels.append(1 if "female" in filename.lower() else 0)
            count += 1
    return np.array(images), np.array(labels)
```

In [3]:
```
X_train_male, Y_train_male = load_images_from_directory(r'C:\Users\ABC\Desktop\
X_train_female, Y_train_female = load_images_from_directory(r'C:\Users\ABC\Desk

X_train = np.concatenate((X_train_male, X_train_female))
Y_train = np.concatenate((Y_train_male, Y_train_female))

X_test_male, Y_test_male = load_images_from_directory(r'C:\Users\ABC\Desktop\BA
X_test_female, Y_test_female = load_images_from_directory(r'C:\Users\ABC\Deskto

X_test = np.concatenate((X_test_male, X_test_female))
Y_test = np.concatenate((Y_test_male, Y_test_female))
```

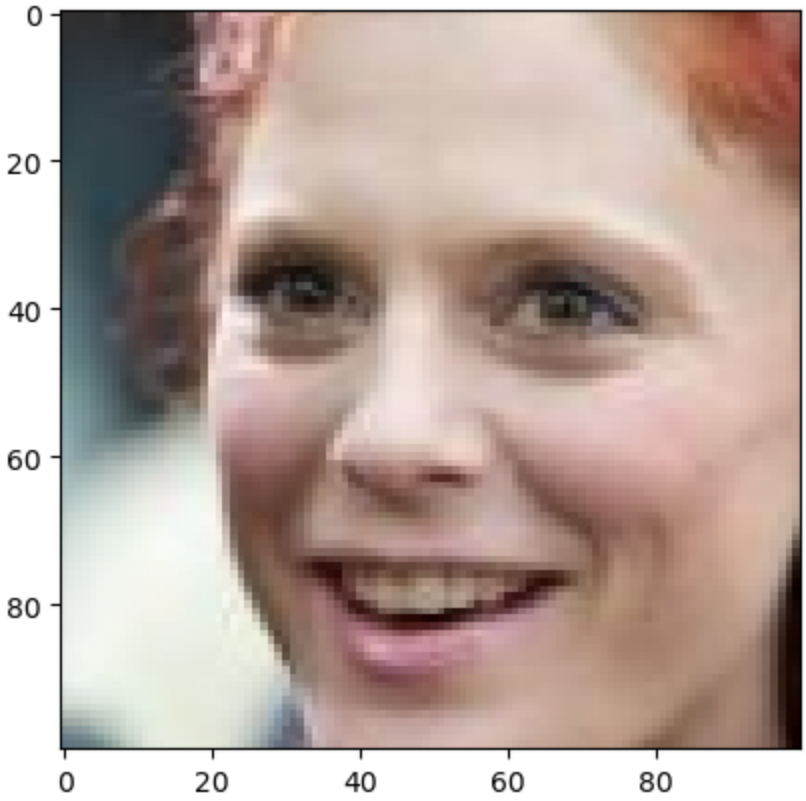
In [4]:
```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_front
```

In [5]:
```
def detect_face_and_align(image):
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5

    if len(faces) == 0:
        return None, None

    (x, y, w, h) = faces[0]
    face = image[y:y + h, x:x + w]
    aligned_face = cv2.resize(face, (100, 100))

    return aligned_face, (x, y, w, h)
```

In [6]:
```
X_train_aligned = []
for i in range(len(X_train)):
    aligned_face, _ = detect_face_and_align(X_train[i])
    if aligned_face is not None:
        X_train_aligned.append(aligned_face)

X_train_aligned = np.array(X_train_aligned)
Y_train = Y_train[:len(X_train_aligned)]
```

In [7]:
```
X_test_aligned = []
for i in range(len(X_test)):
    aligned_face, _ = detect_face_and_align(X_test[i])
    if aligned_face is not None:
        X_test_aligned.append(aligned_face)

X_test_aligned = np.array(X_test_aligned)
Y_test = Y_test[:len(X_test_aligned)]
```

In [8]:
```
idx = random.randint(0, len(X_train_aligned))
plt.imshow(X_train_aligned[idx, :])
plt.show()
```



In [9]:
```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

In [10]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy
```

In [11]:
```
model.fit(X_train_aligned, Y_train, epochs=10, batch_size=64)
```

```
Epoch 1/10
3/3 [==============================] - 3s 433ms/step - loss: 3.9231 - accurac
y: 0.7234
Epoch 2/10
3/3 [==============================] - 1s 445ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 3/10
3/3 [==============================] - 1s 457ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 4/10
3/3 [==============================] - 1s 461ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 5/10
3/3 [==============================] - 1s 457ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 6/10
3/3 [==============================] - 1s 449ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 7/10
3/3 [==============================] - 1s 464ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 8/10
3/3 [==============================] - 1s 466ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 9/10
3/3 [==============================] - 1s 450ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
Epoch 10/10
3/3 [==============================] - 1s 463ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
```

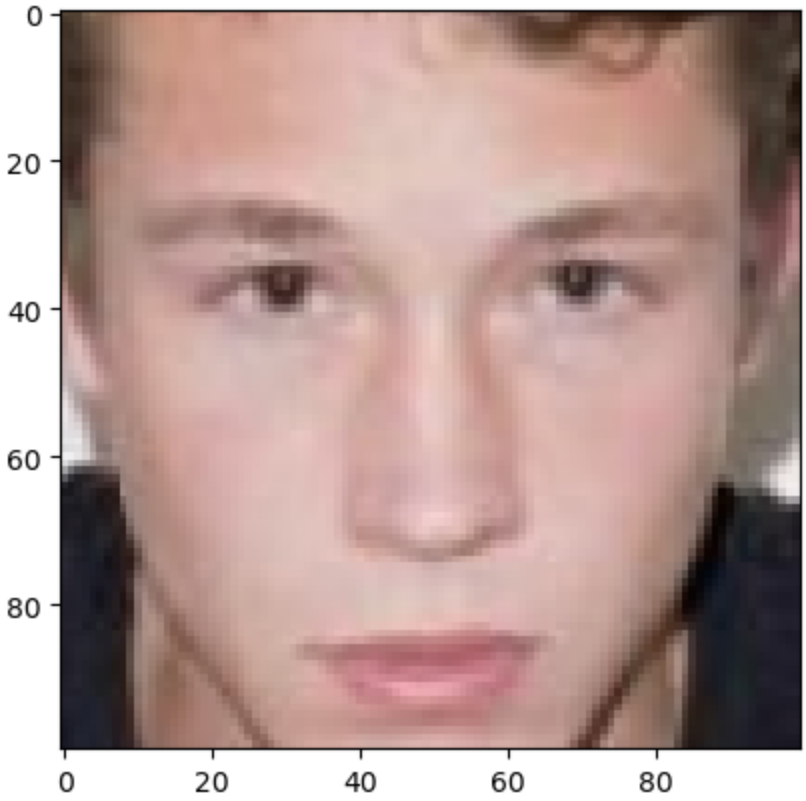
Out[11]:
```
<keras.src.callbacks.History at 0x292b7ab1b90>
```

In [12]:
```
model.evaluate(X_test_aligned, Y_test)
```

```
7/7 [==============================] - 1s 63ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
```

Out[12]:
```
[0.0, 1.0]
```

In [13]:
```
idx2 = random.randint(0, len(Y_test))
plt.imshow(X_test_aligned[idx2, :])
plt.show()
```



In [14]:
```
y_pred = model.predict(X_test_aligned[idx2, :].reshape(1, 100, 100, 3))
y_pred = y_pred > 0.5

if y_pred == 0:
    pred = 'male'
else:
    pred = 'female'

print("Our model says it is a:", pred)
```

```
1/1 [==============================] - 0s 141ms/step
Our model says it is a: male
```

Q2

In [75]:
```
import glob

def load_images_from_directory(directory, limit=None):
    images = []
    labels = []
    count = 0
    image_paths = glob.glob(os.path.join(directory, '*.jpg')) + glob.glob(os.pa

    for img_path in image_paths[:limit]:
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is None:
            print(f"Error loading image: {img_path}")
            continue
        img = cv2.resize(img, (100, 100))
        images.append(img)
        labels.append(os.path.basename(directory).lower())
        count += 1

    print("Loaded images:", len(images))
    print("Loaded labels:", len(labels))

    return np.array(images), np.array(labels)
```

In [76]:
```
X_train_angry, Y_train_angry = load_images_from_directory(r'C:\Users\ABC\Deskto
X_train_happy, Y_train_happy = load_images_from_directory(r'C:\Users\ABC\Deskto
X_train_other, Y_train_other = load_images_from_directory(r'C:\Users\ABC\Deskto
X_train_sad, Y_train_sad = load_images_from_directory(r'C:\Users\ABC\Desktop\BA

X_train = np.concatenate((X_train_angry, X_train_happy, X_train_other, X_train_
Y_train = np.concatenate((Y_train_angry, Y_train_happy, Y_train_other, Y_train_

X_test_angry, Y_test_angry = load_images_from_directory(r'C:\Users\ABC\Desktop\
X_test_happy, Y_test_happy = load_images_from_directory(r'C:\Users\ABC\Desktop\
X_test_other, Y_test_other = load_images_from_directory(r'C:\Users\ABC\Desktop\
X_test_sad, Y_test_sad = load_images_from_directory(r'C:\Users\ABC\Desktop\BAI\

X_test = np.concatenate((X_test_angry, X_test_happy, X_test_other, X_test_sad))
Y_test = np.concatenate((Y_test_angry, Y_test_happy, Y_test_other, Y_test_sad))
```

```
Loaded images: 50
Loaded labels: 50
Loaded images: 50
Loaded labels: 50
Loaded images: 47
Loaded labels: 47
Loaded images: 50
Loaded labels: 50
Loaded images: 75
Loaded labels: 75
Loaded images: 90
Loaded labels: 90
Loaded images: 47
Loaded labels: 47
Loaded images: 84
Loaded labels: 84
```

In [77]:
```
print("Number of images in X_train:", len(X_train))
print("Number of labels in Y_train:", len(Y_train))
```

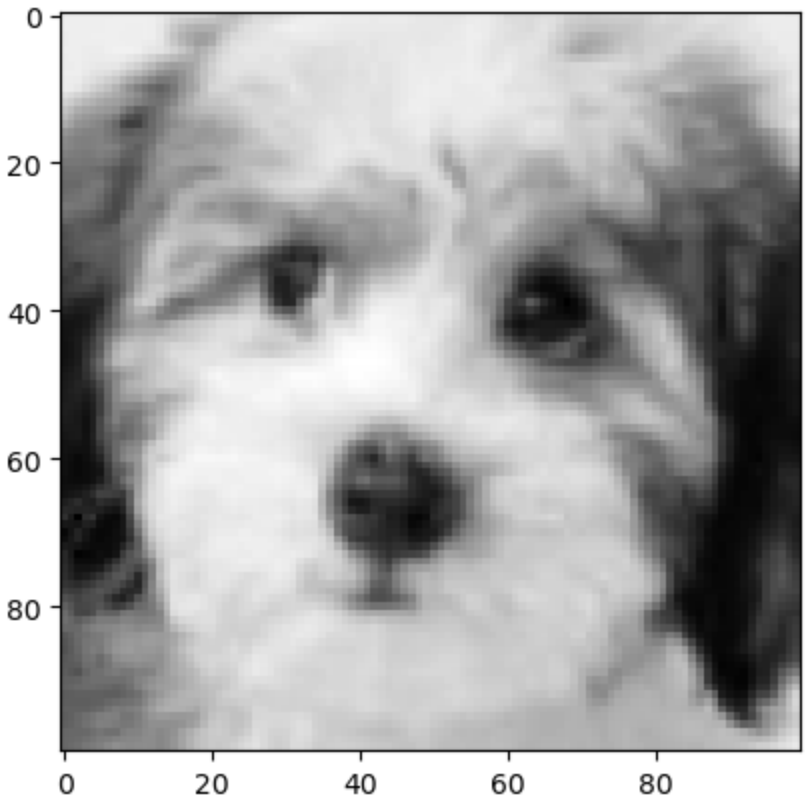
```
Number of images in X_train: 197
Number of labels in Y_train: 197
```

In [78]:
```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_front
```

In [81]:
```
def detect_face_and_align(image):
    gray = image
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5

    if len(faces) == 0:
        return None, None

    (x, y, w, h) = faces[0]
    face = image[y:y + h, x:x + w]
    aligned_face = cv2.resize(face, (100, 100))

    return aligned_face, (x, y, w, h)
```

In [82]:
```
X_train_aligned = []
for i in range(len(X_train)):
    aligned_face, _ = detect_face_and_align(X_train[i])
    if aligned_face is not None:
        X_train_aligned.append(aligned_face)

X_train_aligned = np.array(X_train_aligned)
Y_train = Y_train[:len(X_train_aligned)]
```

In [83]:
```
idx = random.randint(0, len(X_train_aligned) - 1)
plt.imshow(X_train_aligned[idx], cmap='gray')
plt.show()
```



In [84]:
```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(4, activation='softmax')
])
```

In [85]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics
```

In [90]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
Y_train_encoded = label_encoder.fit_transform(Y_train)

Y_test_filtered = [label for label in Y_test if label in label_encoder.classes_

Y_test_encoded = label_encoder.transform(Y_test_filtered)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(4, activation='softmax')
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics

model.fit(X_train_aligned, Y_train_encoded, epochs=10, batch_size=64)
```

```
Epoch 1/10
1/1 [==============================] - 1s 1s/step - loss: 0.0000e+00 - accura
cy: 1.0000
Epoch 2/10
1/1 [==============================] - 0s 50ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 3/10
1/1 [==============================] - 0s 48ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 4/10
1/1 [==============================] - 0s 46ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 5/10
1/1 [==============================] - 0s 46ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 6/10
1/1 [==============================] - 0s 44ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 7/10
1/1 [==============================] - 0s 50ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 8/10
1/1 [==============================] - 0s 47ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 9/10
1/1 [==============================] - 0s 40ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
Epoch 10/10
1/1 [==============================] - 0s 50ms/step - loss: 0.0000e+00 - accu
racy: 1.0000
```

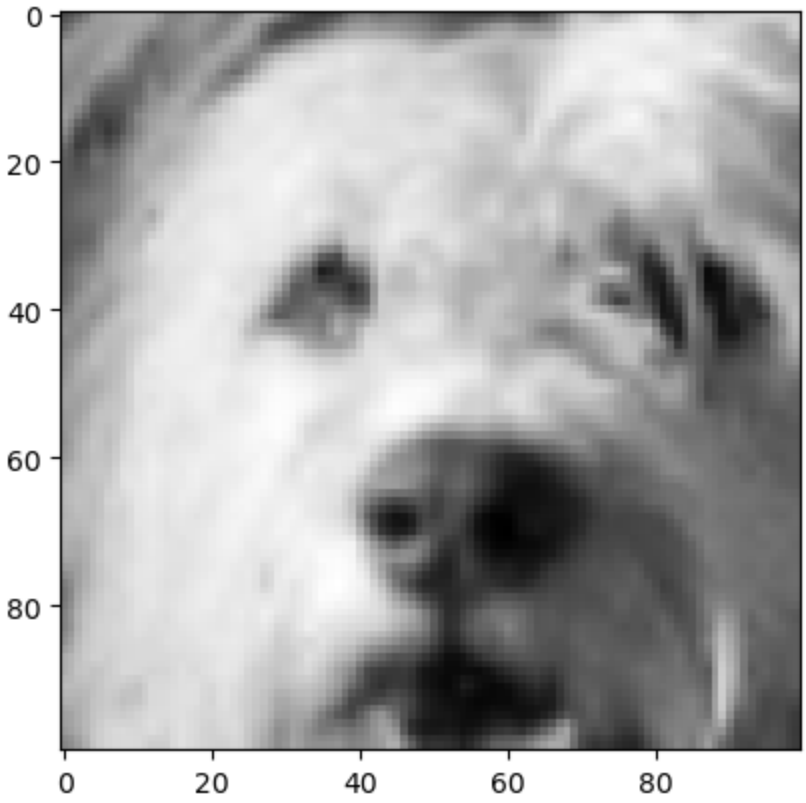
Out[90]: <keras.src.callbacks.History at 0x292cb515090>

In [93]:
```
X_test_aligned = []
for i in range(len(X_test)):
    aligned_face, _ = detect_face_and_align(X_test[i])
    if aligned_face is not None:
        X_test_aligned.append(aligned_face)

X_test_aligned = np.array(X_test_aligned)

Y_test_encoded = label_encoder.transform(Y_test)

model.evaluate(X_test_aligned, Y_test_encoded)
```

```
1/1 [==============================] - 0s 278ms/step - loss: 0.0000e+00 - acc
uracy: 1.0000
```

Out[93]:
```
[0.0, 1.0]
```

In [94]:
```
idx2 = random.randint(0, len(Y_test) - 1)
plt.imshow(X_test_aligned[idx2], cmap='gray')
plt.show()
```



In [95]:
```
y_pred_probs = model.predict(X_test_aligned[idx2].reshape(1, 100, 100, 1))
y_pred = np.argmax(y_pred_probs)

expression_mapping = {0: 'angry', 1: 'happy', 2: 'other', 3: 'sad'}
pred = expression_mapping[y_pred]

print("Our model predicts the expression as:", pred)
```

```
1/1 [==============================] - 0s 85ms/step
Our model predicts the expression as: angry
```

Q3

In [7]:
```
import pandas as pd
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

df = pd.read_csv(r"C:\Users\ABC\Desktop\BAI\BAI-S7\CV Lab\Lab 08\Task 03\train

def load_and_preprocess_images(file_paths):
    images = []
    for file_path in file_paths:
        img = cv2.imread(r"C:\Users\ABC\Desktop\BAI\BAI-S7\CV Lab\Lab 08\Task 0
        img = cv2.resize(img, (224, 224))
        images.append(img)
    return np.array(images)

X = load_and_preprocess_images(df['ID'].values)
y = df['Class'].values

label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

X_train, X_val, y_train, y_val = train_test_split(X, y_encoded, test_size=0.2,

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='linear'))

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

model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y

val_loss, val_mae = model.evaluate(X_val, y_val)
print(f"Validation Mean Absolute Error: {val_mae}")

model.save("age_estimation_model.h5")
```

```
Epoch 1/10
498/498 [==============================] - 1199s 2s/step - loss: 10342.1006 -
mae: 6.4624 - val_loss: 0.8018 - val_mae: 0.7515
Epoch 2/10
498/498 [==============================] - 1485s 3s/step - loss: 1.1094 - ma
e: 0.7983 - val_loss: 0.8762 - val_mae: 0.8198
Epoch 3/10
498/498 [==============================] - 1405s 3s/step - loss: 0.8141 - ma
e: 0.8279 - val_loss: 0.8206 - val_mae: 0.8414
Epoch 4/10
498/498 [==============================] - 1071s 2s/step - loss: 0.7899 - ma
e: 0.8250 - val_loss: 0.8079 - val_mae: 0.8379
Epoch 5/10
498/498 [==============================] - 1092s 2s/step - loss: 0.7743 - ma
e: 0.8131 - val_loss: 0.8161 - val_mae: 0.8352
Epoch 6/10
498/498 [==============================] - 1071s 2s/step - loss: 0.7655 - ma
e: 0.8050 - val_loss: 0.8127 - val_mae: 0.8338
Epoch 7/10
498/498 [==============================] - 1071s 2s/step - loss: 0.7532 - ma
e: 0.7959 - val_loss: 0.8220 - val_mae: 0.8364
Epoch 8/10
498/498 [==============================] - 1050s 2s/step - loss: 0.7467 - ma
e: 0.7914 - val_loss: 0.8214 - val_mae: 0.8358
Epoch 9/10
498/498 [==============================] - 1127s 2s/step - loss: 0.7328 - ma
e: 0.7809 - val_loss: 0.8354 - val_mae: 0.8341
Epoch 10/10
498/498 [==============================] - 1078s 2s/step - loss: 0.7215 - ma
e: 0.7714 - val_loss: 0.8334 - val_mae: 0.8371
125/125 [==============================] - 66s 525ms/step - loss: 0.8334 - ma
e: 0.8371
Validation Mean Absolute Error: 0.8370787501335144

C:\Users\ABC\anaconda3\Lib\site-packages\keras\src\engine\training.py:3079: U
serWarning: You are saving your model as an HDF5 file via `model.save()`. Thi
s file format is considered legacy. We recommend using instead the native Ker
as format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [40]:
```
import cv2
import numpy as np
from keras.models import load_model
from sklearn.preprocessing import LabelEncoder

face_cascade_path = cv2.data.haarcascades + 'haarcascade_frontalface_alt.xml'
face_cascade = cv2.CascadeClassifier(face_cascade_path)

def classify_age(video_path, model_path, label_encoder):
    model = load_model(model_path)

    cap = cv2.VideoCapture(video_path)

    if not hasattr(label_encoder, 'classes_') or len(label_encoder.classes_) ==
        raise ValueError("LabelEncoder is not fitted. Fit the LabelEncoder befo

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbo

        for (x, y, w, h) in faces:
            face = frame[y:y + h, x:x + w]
            face = cv2.resize(face, (224, 224))

            face = face / 255.0
            face = np.expand_dims(face, axis=0)

            predicted_age = model.predict(face).squeeze()

            # Need to define threshold since the classification was done accord
            young_age_threshold = 25
            old_age_threshold = 60

            if predicted_age < young_age_threshold:
                age_label = "YOUNG"
            elif predicted_age >= old_age_threshold:
                age_label = "OLD"
            else:
                age_label = "MIDDLE"

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, f"Age: {age_label}", (x, y - 10), cv2.FONT_HERSH

        cv2.imshow('Age Estimation', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

model_path = "age_estimation_model.h5"
```

```
label_encoder = LabelEncoder()
label_encoder.fit(y_train)

video_path = r"C:\Users\ABC\Videos\Captures\Baby's Day Out full movie - YouTube

classify_age(video_path, model_path, label_encoder)
```

```
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
```

```
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
```

```
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
```

```
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 81ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
```

```
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 42ms/step
```