Deep Learning Libraries

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```
pip install numpy opencv-python tensorflow matplotlib seaborn scikit-learn
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-
packages (1.26.4)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-
packages (2.17.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.8.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-
packages (0.13.2)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (24.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.2
0.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.25.5)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (75.1.0)
```

```
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.68.0)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.1)
Requirement already satisfied: keras>=3.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.5.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.55.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
packages (from seaborn) (2.2.2)
Requirement already satisfied: scipy>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.45.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->tensorflow) (0.13.1)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
```

```
/usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17-
>tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.18,>=2.17->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow)
(2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow) (0.1.2)
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import zipfile
import cv2

# Deep Learning Libraries
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers, regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

1. Loadtheimages.

2. Discard images that correspond to ages outside the range [10, 90].

```
def load_and_preprocess_data(zip_file_path):
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall('dataset')

    images = []
    ages = []

    dataset_folder ="/content/drive/MyDrive/FCAI /utkcropped"

    # Walk through extracted directory
    for filename in os.listdir(dataset_folder):
        if filename.endswith('.jpg'):
            parts = filename.split('_')
            try:
                age = int(parts[0])

                if 10 <= age <= 90:
                    img_path = os.path.join(dataset_folder, filename)
                    img = cv2.imread(img_path)
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    img = cv2.resize(img, (64, 64))

                    images.append(img)
                    ages.append(age)
            except ValueError:
                continue

    X = np.array(images) / 255.0
    y = np.array(ages)

    return X, y
```

Pre Model

```
# Split the data
def split_data(X, y, test_size=0.2, val_size=0.2):
    from sklearn.model_selection import train_test_split

    X_train_val, X_test, y_train_val, y_test = train_test_split(
        X, y, test_size=test_size, random_state=42
    )

    X_train, X_val, y_train, y_val = train_test_split(
        X_train_val, y_train_val, test_size=val_size, random_state=42
    )
```

```
    return X_train, X_val, X_test, y_train, y_val, y_test

def train_model(model, X_train, y_train, X_val, y_val,
data_augmentation=None):
    # Callbacks
    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True
    )

    reduce_lr = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=3,
        min_lr=0.00001
    )

    # Training
    if data_augmentation:
        history = model.fit(
            data_augmentation.flow(X_train, y_train, batch_size=32),
            validation_data=(X_val, y_val),
            epochs=10,
            callbacks=[early_stopping, reduce_lr]
        )
    else:
        history = model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=20,
            batch_size=32,
            callbacks=[early_stopping, reduce_lr]
        )

    return history

def plot_training_history(histories, model_names):
    plt.figure(figsize=(15, 5))

    # Training Loss
    plt.subplot(1, 2, 1)
    for history, name in zip(histories, model_names):
        plt.plot(history.history['loss'], label=f'{name} Training Loss')
        plt.plot(history.history['val_loss'], label=f'{name} Validation
Loss')

    plt.title('Model Training and Validation Loss')
    plt.xlabel('Epoch')
```

```
    plt.ylabel('Loss')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

Two large convolutional neural network (CNN) models with custom architectures.

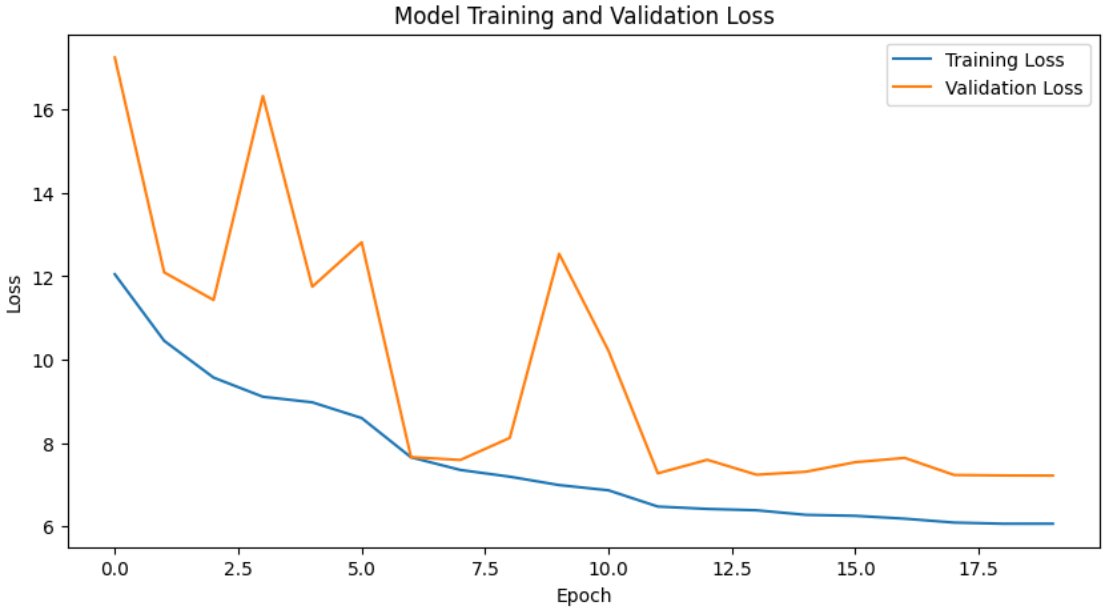
Modle 1

```
# Custom CNN Model 1: Deep CNN
def create_deep_cnn_model1(input_shape=(64, 64, 3)):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape,
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(64, (3, 3), activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(128, (3, 3), activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Flatten(),
        layers.Dense(256, activation='relu',
                     kernel_regularizer=regularizers.l2(0.001)),
        layers.Dropout(0.5),
        layers.Dense(1, activation='linear')
    ])

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

def main():
    zip_file_path = "/content/drive/MyDrive/FCAI /DL_Assignment 1.zip"
    X, y = load_and_preprocess_data(zip_file_path)

    X_train, X_val, X_test, y_train, y_val, y_test = split_data(X, y)

    model = create_deep_cnn_model1()
    history = train_model(model, X_train, y_train, X_val, y_val)
```

```
    plt.figure(figsize=(10, 5))
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # Evaluate Model
    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Mean Absolute Error: {test_mae}")

main()
```

```
Epoch 1/20
412/412 ━━━━━━━━━━━━━━━━━━━━15s 19ms/step - loss: 13.5787 - mae: 12.9609 -
val_loss: 17.2450 - val_mae: 16.6422 - learning_rate: 0.0010
Epoch 2/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 10.5029 - mae: 9.8816 -
val_loss: 12.0934 - val_mae: 11.4556 - learning_rate: 0.0010
Epoch 3/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 9.6632 - mae: 9.0219 -
val_loss: 11.4303 - val_mae: 10.7726 - learning_rate: 0.0010
Epoch 4/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 9.1362 - mae: 8.4724 -
val_loss: 16.3225 - val_mae: 15.6367 - learning_rate: 0.0010
Epoch 5/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 8.9382 - mae: 8.2471 -
val_loss: 11.7494 - val_mae: 11.0350 - learning_rate: 0.0010
Epoch 6/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 8.5419 - mae: 7.8207 -
val_loss: 12.8146 - val_mae: 12.0667 - learning_rate: 0.0010
Epoch 7/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 7.6754 - mae: 6.9324 -
val_loss: 7.6619 - val_mae: 6.9365 - learning_rate: 2.0000e-04
Epoch 8/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 7.3332 - mae: 6.6127 -
val_loss: 7.5944 - val_mae: 6.8894 - learning_rate: 2.0000e-04
Epoch 9/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 7.2841 - mae: 6.5835 -
val_loss: 8.1233 - val_mae: 7.4364 - learning_rate: 2.0000e-04
Epoch 10/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 7.0181 - mae: 6.3352 -
val_loss: 12.5380 - val_mae: 11.8660 - learning_rate: 2.0000e-04
Epoch 11/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.8614 - mae: 6.1915 -
val_loss: 10.2125 - val_mae: 9.5521 - learning_rate: 2.0000e-04
Epoch 12/20
```

```
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.5656 - mae: 5.9057 -
val_loss: 7.2734 - val_mae: 6.6163 - learning_rate: 4.0000e-05
Epoch 13/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.4378 - mae: 5.7816 -
val_loss: 7.5986 - val_mae: 6.9452 - learning_rate: 4.0000e-05
Epoch 14/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.4425 - mae: 5.7899 -
val_loss: 7.2395 - val_mae: 6.5902 - learning_rate: 4.0000e-05
Epoch 15/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.2136 - mae: 5.5649 -
val_loss: 7.3120 - val_mae: 6.6665 - learning_rate: 4.0000e-05
Epoch 16/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.2416 - mae: 5.5967 -
val_loss: 7.5407 - val_mae: 6.8985 - learning_rate: 4.0000e-05
Epoch 17/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.1372 - mae: 5.4957 -
val_loss: 7.6429 - val_mae: 7.0043 - learning_rate: 4.0000e-05
Epoch 18/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.0581 - mae: 5.4195 -
val_loss: 7.2341 - val_mae: 6.5961 - learning_rate: 1.0000e-05
Epoch 19/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.0364 - mae: 5.3987 -
val_loss: 7.2224 - val_mae: 6.5854 - learning_rate: 1.0000e-05
Epoch 20/20
412/412 ━━━━━━━━━━━━━━━━━━━━3s 6ms/step - loss: 6.0754 - mae: 5.4386 -
val_loss: 7.2193 - val_mae: 6.5832 - learning_rate: 1.0000e-05
```




```
129/129 ━━━━━━━━━━━━━━━━━━━━1s 7ms/step - loss: 7.0537 - mae: 6.4176
Test Mean Absolute Error: 6.466558933258057
```

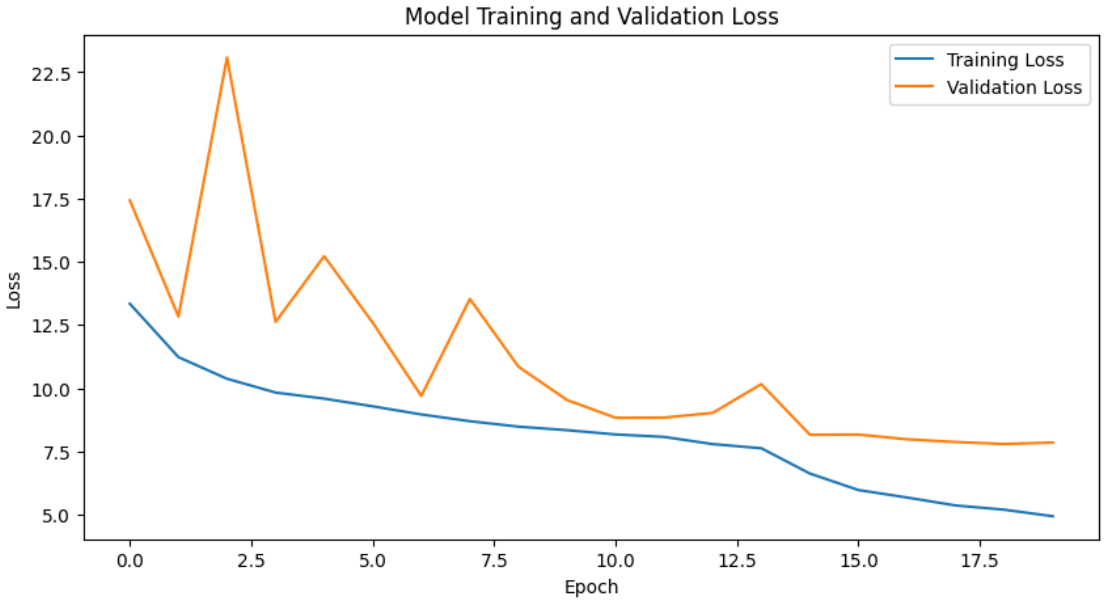
Modle 2

```
def create_deep_cnn_model2(input_shape=(64, 64, 3)):
    model = models.Sequential([
        layers.Conv2D(64, (3, 3), activation='relu', input_shape=input_shape,
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(128, (3, 3), activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(256, (3, 3), activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(512, (3, 3), activation='relu',
                      kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),

        layers.Flatten(),
        layers.Dense(1024, activation='relu',
                     kernel_regularizer=regularizers.l2(0.001)),
        layers.Dropout(0.5),
        layers.Dense(1, activation='linear')
    ])

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

def main1():

    zip_file_path = "/content/drive/MyDrive/FCAI /DL_Assignment 1.zip"
    X, y = load_and_preprocess_data(zip_file_path)

    X_train, X_val, X_test, y_train, y_val, y_test = split_data(X, y)

    model = create_deep_cnn_model2()
    history = train_model(model, X_train, y_train, X_val, y_val)

    plt.figure(figsize=(10, 5))
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Training and Validation Loss')
    plt.xlabel('Epoch')
```

```
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Mean Absolute Error: {test_mae}")

main1()
```

```
Epoch 1/20
412/412 ━━━━━━━━━━━━━━━━━━━━20s 31ms/step - loss: 14.8094 - mae: 12.9078 -
val_loss: 17.4319 - val_mae: 15.8284 - learning_rate: 0.0010
Epoch 2/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 11.5735 - mae: 10.0127 -
val_loss: 12.8345 - val_mae: 11.3713 - learning_rate: 0.0010
Epoch 3/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 10.4723 - mae: 9.0078 -
val_loss: 23.0845 - val_mae: 21.6226 - learning_rate: 0.0010
Epoch 4/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 9.8285 - mae: 8.3586 -
val_loss: 12.6209 - val_mae: 11.1703 - learning_rate: 0.0010
Epoch 5/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 9.6331 - mae: 8.1867 -
val_loss: 15.2241 - val_mae: 13.8023 - learning_rate: 0.0010
Epoch 6/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 9.1898 - mae: 7.7628 -
val_loss: 12.5908 - val_mae: 11.1641 - learning_rate: 0.0010
Epoch 7/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 8.9211 - mae: 7.4893 -
val_loss: 9.6938 - val_mae: 8.2452 - learning_rate: 0.0010
Epoch 8/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 8.7241 - mae: 7.2663 -
val_loss: 13.5302 - val_mae: 12.0495 - learning_rate: 0.0010
Epoch 9/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 8.3225 - mae: 6.8301 -
val_loss: 10.8508 - val_mae: 9.3377 - learning_rate: 0.0010
Epoch 10/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 8.3284 - mae: 6.8008 -
val_loss: 9.5319 - val_mae: 7.9922 - learning_rate: 0.0010
Epoch 11/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 8.2354 - mae: 6.6958 -
val_loss: 8.8288 - val_mae: 7.2620 - learning_rate: 0.0010
Epoch 12/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 8.0376 - mae: 6.4525 -
val_loss: 8.8364 - val_mae: 7.2250 - learning_rate: 0.0010
Epoch 13/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 7.7545 - mae: 6.1349 -
val_loss: 9.0223 - val_mae: 7.3993 - learning_rate: 0.0010
Epoch 14/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 7.6382 - mae: 6.0053 -
```

```
val_loss: 10.1613 - val_mae: 8.5197 - learning_rate: 0.0010
Epoch 15/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 6.8295 - mae: 5.2073 -
val_loss: 8.1589 - val_mae: 6.5971 - learning_rate: 2.0000e-04
Epoch 16/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 5.9676 - mae: 4.4239 -
val_loss: 8.1650 - val_mae: 6.6723 - learning_rate: 2.0000e-04
Epoch 17/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 5.6444 - mae: 4.1656 -
val_loss: 7.9789 - val_mae: 6.5380 - learning_rate: 2.0000e-04
Epoch 18/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 5.3248 - mae: 3.8948 -
val_loss: 7.8710 - val_mae: 6.4712 - learning_rate: 2.0000e-04
Epoch 19/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 5.1489 - mae: 3.7571 -
val_loss: 7.7924 - val_mae: 6.4248 - learning_rate: 2.0000e-04
Epoch 20/20
412/412 ━━━━━━━━━━━━━━━━━━━━6s 14ms/step - loss: 4.9074 - mae: 3.5470 -
val_loss: 7.8498 - val_mae: 6.5105 - learning_rate: 2.0000e-04
```




```
129/129 ━━━━━━━━━━━━━━━━━━━━1s 11ms/step - loss: 7.6636 - mae: 6.2959
Test Mean Absolute Error: 6.37539005279541
```

#Two smaller models that apply transfer learning by taking the features extracted by a base CNN model of a well-known architecture as their input, and learning to leverage these features for the age estimation task.

ResNet50

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers, regularizers
```

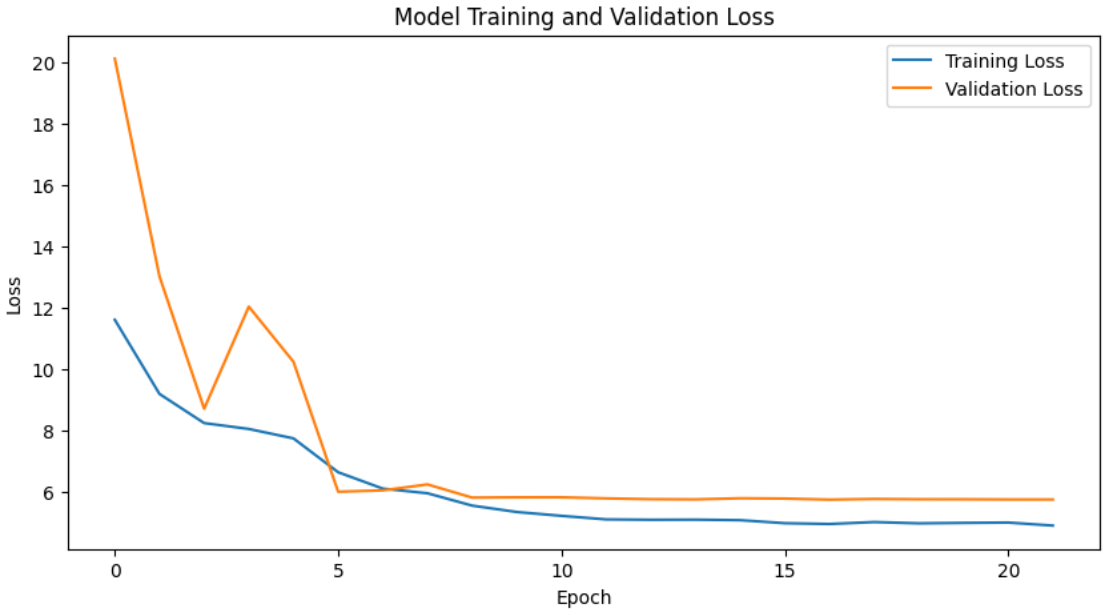
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

def load_and_preprocess_data(zip_file_path):

    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall('dataset')

    images = []
    ages = []

    dataset_folder =r"/content/drive/MyDrive/FCAI /utkcropped"

    for filename in os.listdir(dataset_folder):
        if filename.endswith('.jpg'):
            parts = filename.split('_')
            try:
                age = int(parts[0])
                if 10 <= age <= 90:
                    img_path = os.path.join(dataset_folder, filename)
                    img = cv2.imread(img_path)
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    img = cv2.resize(img, (64, 64))

                    images.append(img)
                    ages.append(age)
            except ValueError:
                continue

    X = np.array(images) / 255.0
    y = np.array(ages)

    return X, y



def create_transfer_learning_model2(input_shape=(64, 64, 3)):
    base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=input_shape)

    model = models.Sequential([
        base_model,
        layers.Flatten(),
        layers.Dense(512, activation='relu',
                     kernel_regularizer=regularizers.l2(0.001)),
        layers.Dropout(0.6),
        layers.Dense(1, activation='linear')
    ])
```

```
    model.compile(optimizer='adam', loss='mean_absolute_error',
metrics=['mae'])
    return model

def split_data(X, y, test_size=0.2, val_size=0.2):
    from sklearn.model_selection import train_test_split

    X_train_val, X_test, y_train_val, y_test = train_test_split(
        X, y, test_size=test_size, random_state=42
    )

    X_train, X_val, y_train, y_val = train_test_split(
        X_train_val, y_train_val, test_size=val_size, random_state=42
    )

    return X_train, X_val, X_test, y_train, y_val, y_test


def train_model(model, X_train, y_train, X_val, y_val,
data_augmentation=None):
    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True
    )

    reduce_lr = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=2,
        min_lr=1e-6
    )

    if data_augmentation:
        history = model.fit(
            data_augmentation.flow(X_train, y_train, batch_size=32),
            validation_data=(X_val, y_val),
            epochs=10,
            callbacks=[early_stopping, reduce_lr]
        )
    else:
        history = model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=50,
            batch_size=32,
            callbacks=[early_stopping, reduce_lr]
        )
```

```
    return history

def main3():
    zip_file_path = r"/content/drive/MyDrive/FCAI /DL_Assignment 1.zip"
    X, y = load_and_preprocess_data(zip_file_path)

    X_train, X_val, X_test, y_train, y_val, y_test = split_data(X, y)

    model = create_transfer_learning_model2()
    history = train_model(model, X_train, y_train, X_val, y_val)

    plt.figure(figsize=(10, 5))
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Mean Absolute Error: {test_mae}")

    model.save("age_prediction_model.h5")
    print("Model saved as 'age_prediction_model.h5'.")

main3()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━1s 0us/step
Epoch 1/50
412/412 ━━━━━━━━━━━━━━━━━━━━114s 126ms/step - loss: 13.6702 - mae: 12.9076 -
val_loss: 20.1047 - val_mae: 19.6905 - learning_rate: 0.0010
Epoch 2/50
412/412 ━━━━━━━━━━━━━━━━━━━━79s 52ms/step - loss: 9.4550 - mae: 9.1017 -
val_loss: 13.0077 - val_mae: 12.7903 - learning_rate: 0.0010
Epoch 3/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 54ms/step - loss: 8.2578 - mae: 8.0673 -
val_loss: 8.6964 - val_mae: 8.5677 - learning_rate: 0.0010
Epoch 4/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 53ms/step - loss: 7.9392 - mae: 7.8233 -
val_loss: 12.0236 - val_mae: 11.9364 - learning_rate: 0.0010
Epoch 5/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 54ms/step - loss: 7.8534 - mae: 7.7737 -
val_loss: 10.2247 - val_mae: 10.1627 - learning_rate: 0.0010
Epoch 6/50
412/412 ━━━━━━━━━━━━━━━━━━━━22s 54ms/step - loss: 6.8093 - mae: 6.7487 -
val_loss: 5.9902 - val_mae: 5.9335 - learning_rate: 2.0000e-04
```

```
Epoch 7/50
412/412 ━━━━━━━━━━━━━━━━━━━━40s 53ms/step - loss: 6.0977 - mae: 6.0420 -
val_loss: 6.0348 - val_mae: 5.9826 - learning_rate: 2.0000e-04
Epoch 8/50
412/412 ━━━━━━━━━━━━━━━━━━━━21s 52ms/step - loss: 5.9846 - mae: 5.9336 -
val_loss: 6.2288 - val_mae: 6.1813 - learning_rate: 2.0000e-04
Epoch 9/50
412/412 ━━━━━━━━━━━━━━━━━━━━22s 54ms/step - loss: 5.5749 - mae: 5.5276 -
val_loss: 5.7994 - val_mae: 5.7528 - learning_rate: 4.0000e-05
Epoch 10/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 53ms/step - loss: 5.3743 - mae: 5.3280 -
val_loss: 5.8085 - val_mae: 5.7630 - learning_rate: 4.0000e-05
Epoch 11/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 54ms/step - loss: 5.1918 - mae: 5.1466 -
val_loss: 5.8096 - val_mae: 5.7654 - learning_rate: 4.0000e-05
Epoch 12/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 56ms/step - loss: 5.1055 - mae: 5.0614 -
val_loss: 5.7732 - val_mae: 5.7293 - learning_rate: 8.0000e-06
Epoch 13/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 56ms/step - loss: 5.0229 - mae: 4.9791 -
val_loss: 5.7469 - val_mae: 5.7034 - learning_rate: 8.0000e-06
Epoch 14/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 56ms/step - loss: 5.0904 - mae: 5.0470 -
val_loss: 5.7407 - val_mae: 5.6977 - learning_rate: 8.0000e-06
Epoch 15/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 56ms/step - loss: 5.1569 - mae: 5.1140 -
val_loss: 5.7768 - val_mae: 5.7343 - learning_rate: 8.0000e-06
Epoch 16/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 56ms/step - loss: 4.9661 - mae: 4.9237 -
val_loss: 5.7677 - val_mae: 5.7258 - learning_rate: 8.0000e-06
Epoch 17/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 56ms/step - loss: 4.9136 - mae: 4.8717 -
val_loss: 5.7326 - val_mae: 5.6908 - learning_rate: 1.6000e-06
Epoch 18/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 56ms/step - loss: 4.9830 - mae: 4.9413 -
val_loss: 5.7548 - val_mae: 5.7132 - learning_rate: 1.6000e-06
Epoch 19/50
412/412 ━━━━━━━━━━━━━━━━━━━━21s 51ms/step - loss: 5.0009 - mae: 4.9594 -
val_loss: 5.7449 - val_mae: 5.7035 - learning_rate: 1.6000e-06
Epoch 20/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 52ms/step - loss: 4.9787 - mae: 4.9374 -
val_loss: 5.7442 - val_mae: 5.7030 - learning_rate: 1.0000e-06
Epoch 21/50
412/412 ━━━━━━━━━━━━━━━━━━━━43s 56ms/step - loss: 4.9214 - mae: 4.8802 -
val_loss: 5.7380 - val_mae: 5.6969 - learning_rate: 1.0000e-06
Epoch 22/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 59ms/step - loss: 4.8712 - mae: 4.8302 -
val_loss: 5.7373 - val_mae: 5.6965 - learning_rate: 1.0000e-06
```




```
129/129 ━━━━━━━━━━━━━━━━━━━━4s 30ms/step - loss: 5.5172 - mae: 5.4754

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.

Test Mean Absolute Error: 5.554823398590088
Model saved as 'age_prediction_model.h5'.
```

VGG16

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers, regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

def load_and_preprocess_data(zip_file_path):

    images = []
    ages = []

    dataset_folder ='/content/drive/MyDrive/faces/utkcropped'

    for filename in os.listdir(dataset_folder):
        if filename.endswith('.jpg'):

            parts = filename.split('_')
```

```
            try:
                age = int(parts[0])

                if 10 <= age <= 90:
                    img_path = os.path.join(dataset_folder, filename)
                    img = cv2.imread(img_path)
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    img = cv2.resize(img, (64, 64))

                    images.append(img)
                    ages.append(age)
            except ValueError:
                continue

    X = np.array(images) / 255.0
    y = np.array(ages)

    return X, y

def create_transfer_learning_model1(input_shape=(64, 64, 3)):
    base_model = VGG16(weights='imagenet', include_top=False,
input_shape=input_shape)

    model = models.Sequential([
        base_model,
        layers.Flatten(),
        layers.Dense(256, activation='relu',
                     kernel_regularizer=regularizers.l2(0.001)),
        layers.Dropout(0.5),
        layers.Dense(1, activation='linear')
    ])

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


def split_data(X, y, test_size=0.2, val_size=0.2):
    from sklearn.model_selection import train_test_split

    X_train_val, X_test, y_train_val, y_test = train_test_split(
        X, y, test_size=test_size, random_state=42
    )

    X_train, X_val, y_train, y_val = train_test_split(
        X_train_val, y_train_val, test_size=val_size, random_state=42
    )
```

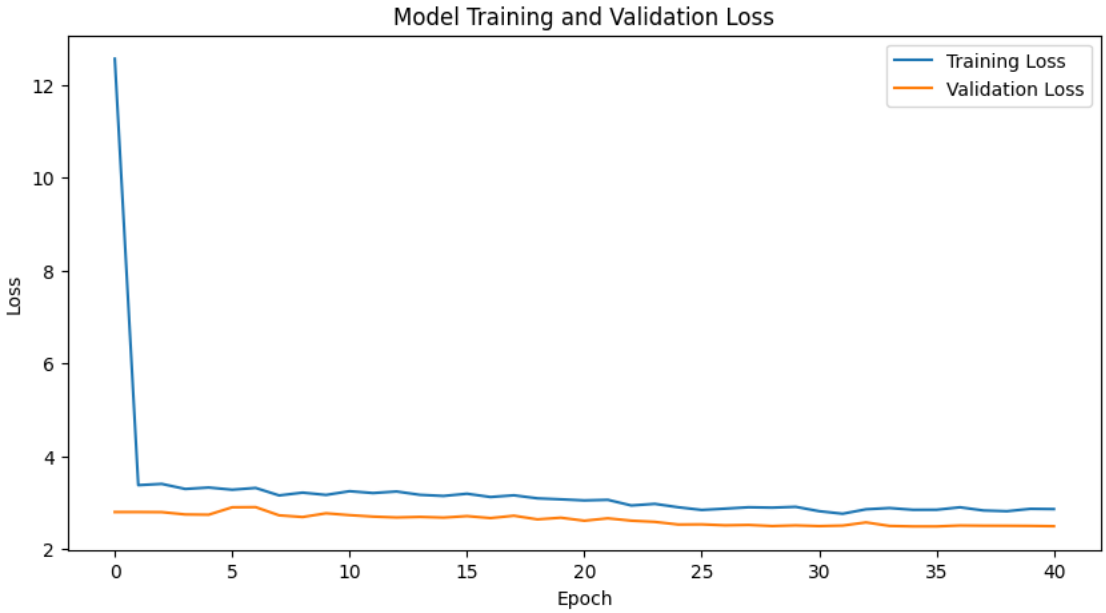
```
    return X_train, X_val, X_test, y_train, y_val, y_test



def train_model(model, X_train, y_train, X_val, y_val,
data_augmentation=None):

    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True
    )

    reduce_lr = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=2,
        min_lr=1e-6
    )

    if data_augmentation:
        history = model.fit(
            data_augmentation.flow(X_train, y_train, batch_size=32),
            validation_data=(X_val, y_val),
            epochs=10,
            callbacks=[early_stopping, reduce_lr]
        )
    else:
        history = model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=50,
            batch_size=32,
            callbacks=[early_stopping, reduce_lr]
        )

    return history

def main2():
    # Load Data
    zip_file_path = '/content/drive/MyDrive/faces/utkcropped'
    X, y = load_and_preprocess_data(zip_file_path)

    # Split Data
    X_train, X_val, X_test, y_train, y_val, y_test = split_data(X, y)

    # Create and Train Model
    model = create_transfer_learning_model1()
    history = train_model(model, X_train, y_train, X_val, y_val)
```

```
    # Plot Training History
    plt.figure(figsize=(10, 5))
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # Evaluate Model
    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Mean Absolute Error: {test_mae}")
    model.save("age_prediction_model.h5")
    print("Model saved as 'age_prediction_model.h5'")

main2()
```

```
Epoch 1/50
39/39 ━━━━━━━━━━━━━━━━━━━━13s 186ms/step - loss: 26.2351 - mae: 25.8795 -
val_loss: 2.7915 - val_mae: 2.5894 - learning_rate: 0.0010
Epoch 2/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 65ms/step - loss: 3.4029 - mae: 3.2133 -
val_loss: 2.7918 - val_mae: 2.6265 - learning_rate: 0.0010
Epoch 3/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 67ms/step - loss: 3.4092 - mae: 3.2490 -
val_loss: 2.7893 - val_mae: 2.6427 - learning_rate: 0.0010
Epoch 4/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 66ms/step - loss: 3.2943 - mae: 3.1513 -
val_loss: 2.7393 - val_mae: 2.6064 - learning_rate: 0.0010
Epoch 5/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 66ms/step - loss: 3.3194 - mae: 3.1894 -
val_loss: 2.7340 - val_mae: 2.6119 - learning_rate: 0.0010
Epoch 6/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 66ms/step - loss: 3.3952 - mae: 3.2754 -
val_loss: 2.8936 - val_mae: 2.7807 - learning_rate: 0.0010
Epoch 7/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 64ms/step - loss: 3.2434 - mae: 3.1317 -
val_loss: 2.8970 - val_mae: 2.7908 - learning_rate: 0.0010
Epoch 8/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 70ms/step - loss: 3.1580 - mae: 3.0519 -
val_loss: 2.7199 - val_mae: 2.6149 - learning_rate: 2.0000e-04
Epoch 9/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 64ms/step - loss: 3.2489 - mae: 3.1442 -
val_loss: 2.6838 - val_mae: 2.5803 - learning_rate: 2.0000e-04
Epoch 10/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 64ms/step - loss: 3.2023 - mae: 3.0991 -
val_loss: 2.7641 - val_mae: 2.6622 - learning_rate: 2.0000e-04
Epoch 11/50
```

```
39/39 ━━━━━━━━━━━━━━━━━━━━2s 63ms/step - loss: 3.2589 - mae: 3.1572 -
val_loss: 2.7245 - val_mae: 2.6240 - learning_rate: 2.0000e-04
Epoch 12/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 68ms/step - loss: 3.2312 - mae: 3.1307 -
val_loss: 2.6927 - val_mae: 2.5923 - learning_rate: 4.0000e-05
Epoch 13/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 70ms/step - loss: 3.2412 - mae: 3.1408 -
val_loss: 2.6741 - val_mae: 2.5741 - learning_rate: 4.0000e-05
Epoch 14/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 64ms/step - loss: 3.1609 - mae: 3.0610 -
val_loss: 2.6864 - val_mae: 2.5867 - learning_rate: 4.0000e-05
Epoch 15/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 64ms/step - loss: 3.1312 - mae: 3.0316 -
val_loss: 2.6714 - val_mae: 2.5720 - learning_rate: 4.0000e-05
Epoch 16/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 63ms/step - loss: 3.2141 - mae: 3.1149 -
val_loss: 2.7035 - val_mae: 2.6045 - learning_rate: 4.0000e-05
Epoch 17/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 70ms/step - loss: 3.2147 - mae: 3.1158 -
val_loss: 2.6607 - val_mae: 2.5621 - learning_rate: 4.0000e-05
Epoch 18/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 66ms/step - loss: 3.2288 - mae: 3.1303 -
val_loss: 2.7116 - val_mae: 2.6134 - learning_rate: 4.0000e-05
Epoch 19/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 68ms/step - loss: 3.1037 - mae: 3.0055 -
val_loss: 2.6335 - val_mae: 2.5356 - learning_rate: 4.0000e-05
Epoch 20/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 63ms/step - loss: 3.0945 - mae: 2.9967 -
val_loss: 2.6687 - val_mae: 2.5712 - learning_rate: 4.0000e-05
Epoch 21/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 64ms/step - loss: 2.9893 - mae: 2.8920 -
val_loss: 2.6039 - val_mae: 2.5069 - learning_rate: 4.0000e-05
Epoch 22/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 68ms/step - loss: 3.0786 - mae: 2.9817 -
val_loss: 2.6558 - val_mae: 2.5592 - learning_rate: 4.0000e-05
Epoch 23/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 65ms/step - loss: 2.9056 - mae: 2.8091 -
val_loss: 2.6050 - val_mae: 2.5088 - learning_rate: 4.0000e-05
Epoch 24/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 64ms/step - loss: 2.9494 - mae: 2.8533 -
val_loss: 2.5791 - val_mae: 2.4831 - learning_rate: 8.0000e-06
Epoch 25/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 63ms/step - loss: 2.8896 - mae: 2.7935 -
val_loss: 2.5222 - val_mae: 2.4262 - learning_rate: 8.0000e-06
Epoch 26/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 62ms/step - loss: 2.8007 - mae: 2.7047 -
val_loss: 2.5254 - val_mae: 2.4295 - learning_rate: 8.0000e-06
Epoch 27/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 64ms/step - loss: 2.8189 - mae: 2.7230 -
val_loss: 2.5047 - val_mae: 2.4089 - learning_rate: 8.0000e-06
```

```
Epoch 28/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 68ms/step - loss: 2.9107 - mae: 2.8149 -
val_loss: 2.5131 - val_mae: 2.4174 - learning_rate: 8.0000e-06
Epoch 29/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 66ms/step - loss: 2.8819 - mae: 2.7861 -
val_loss: 2.4885 - val_mae: 2.3929 - learning_rate: 8.0000e-06
Epoch 30/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 64ms/step - loss: 2.8542 - mae: 2.7586 -
val_loss: 2.5031 - val_mae: 2.4076 - learning_rate: 8.0000e-06
Epoch 31/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 64ms/step - loss: 2.8179 - mae: 2.7224 -
val_loss: 2.4876 - val_mae: 2.3921 - learning_rate: 8.0000e-06
Epoch 32/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 62ms/step - loss: 2.7453 - mae: 2.6499 -
val_loss: 2.4991 - val_mae: 2.4038 - learning_rate: 8.0000e-06
Epoch 33/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 68ms/step - loss: 2.8924 - mae: 2.7971 -
val_loss: 2.5676 - val_mae: 2.4724 - learning_rate: 8.0000e-06
Epoch 34/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 64ms/step - loss: 2.9100 - mae: 2.8148 -
val_loss: 2.4914 - val_mae: 2.3962 - learning_rate: 1.6000e-06
Epoch 35/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 65ms/step - loss: 2.8239 - mae: 2.7287 -
val_loss: 2.4810 - val_mae: 2.3858 - learning_rate: 1.6000e-06
Epoch 36/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 69ms/step - loss: 2.8243 - mae: 2.7291 -
val_loss: 2.4809 - val_mae: 2.3857 - learning_rate: 1.6000e-06
Epoch 37/50
39/39 ━━━━━━━━━━━━━━━━━━━━3s 66ms/step - loss: 3.0190 - mae: 2.9238 -
val_loss: 2.5007 - val_mae: 2.4055 - learning_rate: 1.6000e-06
Epoch 38/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 64ms/step - loss: 2.8173 - mae: 2.7222 -
val_loss: 2.4961 - val_mae: 2.4010 - learning_rate: 1.6000e-06
Epoch 39/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 64ms/step - loss: 2.6913 - mae: 2.5962 -
val_loss: 2.4949 - val_mae: 2.3998 - learning_rate: 1.0000e-06
Epoch 40/50
39/39 ━━━━━━━━━━━━━━━━━━━━5s 64ms/step - loss: 2.9207 - mae: 2.8256 -
val_loss: 2.4920 - val_mae: 2.3969 - learning_rate: 1.0000e-06
Epoch 41/50
39/39 ━━━━━━━━━━━━━━━━━━━━2s 63ms/step - loss: 2.9778 - mae: 2.8827 -
val_loss: 2.4854 - val_mae: 2.3903 - learning_rate: 1.0000e-06
```



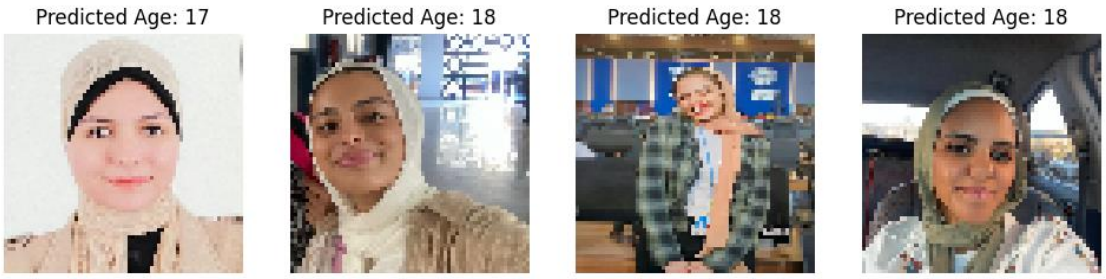

```
13/13 ━━━━━━━━━━━━━━━━━━━━0s 38ms/step - loss: 2.4259 - mae: 2.3308

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.

Test Mean Absolute Error: 2.340360403060913
Model saved as 'age_prediction_model.h5'
```

8. Select the best model, use it to make age predictions on face images of your team, and plot these images with the estimated ages.

```
import os
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

def predict_and_visualize(model_path, image_paths):

    model = tf.keras.models.load_model(model_path)

    images = []
    valid_image_paths = []

    for img_path in image_paths:
        if not os.path.exists(img_path):
            print(f"Image not found: {img_path}")
```

```
            continue
        img = cv2.imread(img_path)
        if img is None:
            print(f"Failed to load image: {img_path}")
            continue
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (64, 64))
        images.append(img)
        valid_image_paths.append(img_path)

    if not images:
        print("No valid images to process.")
        return

    images = np.array(images) / 255.0

    predictions = model.predict(images)
    predicted_ages = predictions.flatten()

    plt.figure(figsize=(12, 6))
    for i, img in enumerate(images):
        plt.subplot(1, len(images), i + 1)
        plt.imshow(img)
        plt.title(f"Predicted Age: {int(predicted_ages[i])}")
        plt.axis('off')
    plt.show()

image_paths = [
"/content/Menna.jpg",
    "/content/Mariam.jpg",
    "/content/jana.jpg",
    "/content/Zaynab.jpg"
]

predict_and_visualize("age_prediction_model.h5",image_paths)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to
be built. `model.compile_metrics` will be empty until you train or evaluate
the model.

1/1 ━━━━━━━━━━━━━━━━━━━━0s 370ms/step
```



VGG16

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers, regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
import os  # Import the os module
from tensorflow.keras.applications import VGG16, ResNet50
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import zipfile
import cv2

# Deep Learning Libraries
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers, regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
def load_and_preprocess_data(zip_file_path):

    images = []
    ages = []

    dataset_folder ='/content/drive/MyDrive/FCAI /utkcropped'

    for filename in os.listdir(dataset_folder):
        if filename.endswith('.jpg'):

            parts = filename.split('_')
            try:
                age = int(parts[0])

                if 10 <= age <= 90:
                    img_path = os.path.join(dataset_folder, filename)
                    img = cv2.imread(img_path)
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    img = cv2.resize(img, (64, 64))

                    images.append(img)
                    ages.append(age)
            except ValueError:
                continue

    X = np.array(images) / 255.0
```

```
    y = np.array(ages)

    return X, y

def create_transfer_learning_model1(input_shape=(64, 64, 3)):
    base_model = VGG16(weights='imagenet', include_top=False,
input_shape=input_shape)

    model = models.Sequential([
        base_model,
        layers.Flatten(),
        layers.Dense(256, activation='relu',
                     kernel_regularizer=regularizers.l2(0.001)),
        layers.Dropout(0.5),
        layers.Dense(1, activation='linear')
    ])

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


def split_data(X, y, test_size=0.2, val_size=0.2):
    from sklearn.model_selection import train_test_split

    X_train_val, X_test, y_train_val, y_test = train_test_split(
        X, y, test_size=test_size, random_state=42
    )

    X_train, X_val, y_train, y_val = train_test_split(
        X_train_val, y_train_val, test_size=val_size, random_state=42
    )

    return X_train, X_val, X_test, y_train, y_val, y_test



def train_model(model, X_train, y_train, X_val, y_val,
data_augmentation=None):

    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True
    )

    reduce_lr = ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
```

```
        patience=2,
        min_lr=1e-6
    )

    if data_augmentation:
        history = model.fit(
            data_augmentation.flow(X_train, y_train, batch_size=32),
            validation_data=(X_val, y_val),
            epochs=10,
            callbacks=[early_stopping, reduce_lr]
        )
    else:
        history = model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=50,
            batch_size=32,
            callbacks=[early_stopping, reduce_lr]
        )

    return history

import cv2 # Import the OpenCV library
import numpy as np #Import the numpy Library

def main2():
    # Load Data
    zip_file_path = '/content/drive/MyDrive/FCAI /DL_Assignment 1.zip'
    X, y = load_and_preprocess_data(zip_file_path)

    # Split Data
    X_train, X_val, X_test, y_train, y_val, y_test = split_data(X, y)

    # Create and Train Model
    model = create_transfer_learning_model1()
    history = train_model(model, X_train, y_train, X_val, y_val)

    # Plot Training History
    plt.figure(figsize=(10, 5))
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # Evaluate Model
    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Mean Absolute Error: {test_mae}")
```

```
    model.save("age_prediction_model.h5")
    print("Model saved as 'age_prediction_model.h5'")

main2()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━0s 0us/step
Epoch 1/50
412/412 ━━━━━━━━━━━━━━━━━━━━54s 98ms/step - loss: 24.1014 - mae: 23.8853 -
val_loss: 13.4696 - val_mae: 13.3534 - learning_rate: 0.0010
Epoch 2/50
412/412 ━━━━━━━━━━━━━━━━━━━━59s 64ms/step - loss: 13.3339 - mae: 13.2258 -
val_loss: 13.4993 - val_mae: 13.4077 - learning_rate: 0.0010
Epoch 3/50
412/412 ━━━━━━━━━━━━━━━━━━━━28s 67ms/step - loss: 13.3378 - mae: 13.2460 -
val_loss: 12.7734 - val_mae: 12.6873 - learning_rate: 0.0010
Epoch 4/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 68ms/step - loss: 12.3991 - mae: 12.3128 -
val_loss: 10.1688 - val_mae: 10.0839 - learning_rate: 0.0010
Epoch 5/50
412/412 ━━━━━━━━━━━━━━━━━━━━40s 65ms/step - loss: 10.5264 - mae: 10.4438 -
val_loss: 9.8306 - val_mae: 9.7496 - learning_rate: 0.0010
Epoch 6/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 68ms/step - loss: 9.4403 - mae: 9.3630 -
val_loss: 8.5121 - val_mae: 8.4421 - learning_rate: 0.0010
Epoch 7/50
412/412 ━━━━━━━━━━━━━━━━━━━━27s 65ms/step - loss: 8.6056 - mae: 8.5376 -
val_loss: 8.2044 - val_mae: 8.1414 - learning_rate: 0.0010
Epoch 8/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 65ms/step - loss: 8.2628 - mae: 8.2011 -
val_loss: 7.6528 - val_mae: 7.5935 - learning_rate: 0.0010
Epoch 9/50
412/412 ━━━━━━━━━━━━━━━━━━━━27s 65ms/step - loss: 7.9129 - mae: 7.8542 -
val_loss: 7.7065 - val_mae: 7.6521 - learning_rate: 0.0010
Epoch 10/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 64ms/step - loss: 7.7057 - mae: 7.6531 -
val_loss: 7.7257 - val_mae: 7.6759 - learning_rate: 0.0010
Epoch 11/50
412/412 ━━━━━━━━━━━━━━━━━━━━27s 65ms/step - loss: 7.1314 - mae: 7.0817 -
val_loss: 7.1162 - val_mae: 7.0680 - learning_rate: 2.0000e-04
Epoch 12/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 67ms/step - loss: 6.5581 - mae: 6.5101 -
val_loss: 6.8062 - val_mae: 6.7596 - learning_rate: 2.0000e-04
Epoch 13/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 67ms/step - loss: 6.2543 - mae: 6.2077 -
val_loss: 6.7856 - val_mae: 6.7403 - learning_rate: 2.0000e-04
Epoch 14/50
412/412 ━━━━━━━━━━━━━━━━━━━━28s 67ms/step - loss: 6.0338 - mae: 5.9887 -
val_loss: 6.6632 - val_mae: 6.6192 - learning_rate: 2.0000e-04
```

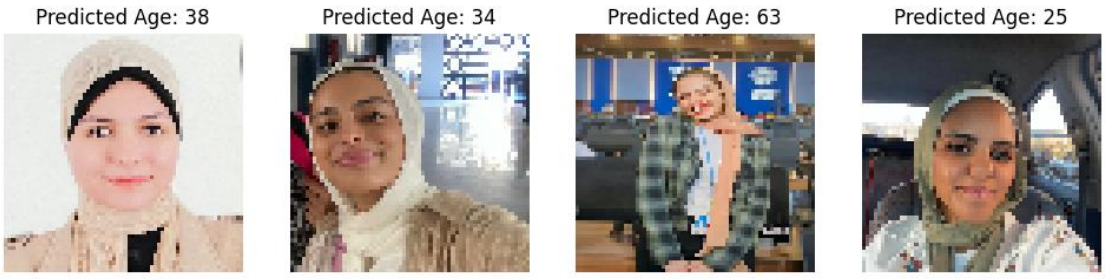
```
Epoch 15/50
412/412 ━━━━━━━━━━━━━━━━━━━━40s 65ms/step - loss: 5.9056 - mae: 5.8620 -
val_loss: 6.6917 - val_mae: 6.6493 - learning_rate: 2.0000e-04
Epoch 16/50
412/412 ━━━━━━━━━━━━━━━━━━━━27s 65ms/step - loss: 5.6445 - mae: 5.6022 -
val_loss: 6.6551 - val_mae: 6.6138 - learning_rate: 2.0000e-04
Epoch 17/50
412/412 ━━━━━━━━━━━━━━━━━━━━27s 65ms/step - loss: 5.4421 - mae: 5.4012 -
val_loss: 6.5996 - val_mae: 6.5601 - learning_rate: 2.0000e-04
Epoch 18/50
412/412 ━━━━━━━━━━━━━━━━━━━━28s 68ms/step - loss: 5.3080 - mae: 5.2690 -
val_loss: 6.7268 - val_mae: 6.6894 - learning_rate: 2.0000e-04
Epoch 19/50
412/412 ━━━━━━━━━━━━━━━━━━━━28s 68ms/step - loss: 5.0909 - mae: 5.0538 -
val_loss: 6.6902 - val_mae: 6.6546 - learning_rate: 2.0000e-04
Epoch 20/50
412/412 ━━━━━━━━━━━━━━━━━━━━28s 68ms/step - loss: 4.8938 - mae: 4.8581 -
val_loss: 6.5635 - val_mae: 6.5281 - learning_rate: 4.0000e-05
Epoch 21/50
412/412 ━━━━━━━━━━━━━━━━━━━━28s 69ms/step - loss: 4.6128 - mae: 4.5774 -
val_loss: 6.5574 - val_mae: 6.5222 - learning_rate: 4.0000e-05
Epoch 22/50
412/412 ━━━━━━━━━━━━━━━━━━━━41s 70ms/step - loss: 4.4972 - mae: 4.4621 -
val_loss: 6.5963 - val_mae: 6.5617 - learning_rate: 4.0000e-05
Epoch 23/50
412/412 ━━━━━━━━━━━━━━━━━━━━40s 68ms/step - loss: 4.4202 - mae: 4.3856 -
val_loss: 6.5977 - val_mae: 6.5637 - learning_rate: 4.0000e-05
Epoch 24/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 70ms/step - loss: 4.3193 - mae: 4.2853 -
val_loss: 6.5717 - val_mae: 6.5378 - learning_rate: 8.0000e-06
Epoch 25/50
412/412 ━━━━━━━━━━━━━━━━━━━━42s 72ms/step - loss: 4.3513 - mae: 4.3174 -
val_loss: 6.5693 - val_mae: 6.5355 - learning_rate: 8.0000e-06
Epoch 26/50
412/412 ━━━━━━━━━━━━━━━━━━━━30s 72ms/step - loss: 4.3029 - mae: 4.2692 -
val_loss: 6.5666 - val_mae: 6.5328 - learning_rate: 1.6000e-06
```


```
129/129 ━━━━━━━━━━━━━━━━━━━━5s 37ms/step - loss: 6.3550 - mae: 6.3198

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.

Test Mean Absolute Error: 6.368239879608154
Model saved as 'age_prediction_model.h5'
```

```python
import os
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

def predict_and_visualize(model_path, image_paths):

    model = tf.keras.models.load_model(model_path)

    images = []
    valid_image_paths = []

    for img_path in image_paths:
        if not os.path.exists(img_path):
            print(f"Image not found: {img_path}")
            continue
        img = cv2.imread(img_path)
        if img is None:
            print(f"Failed to load image: {img_path}")
```

```
            continue
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (64, 64))
        images.append(img)
        valid_image_paths.append(img_path)

    if not images:
        print("No valid images to process.")
        return

    images = np.array(images) / 255.0

    predictions = model.predict(images)
    predicted_ages = predictions.flatten()

    plt.figure(figsize=(12, 6))
    for i, img in enumerate(images):
        plt.subplot(1, len(images), i + 1)
        plt.imshow(img)
        plt.title(f"Predicted Age: {int(predicted_ages[i])}")
        plt.axis('off')
    plt.show()

image_paths = [
"/content/Menna.jpg",
    "/content/Mariam.jpg",
    "/content/Jana.jpg",
    "/content/Zaynab.jpg"
]

predict_and_visualize("age_prediction_model.h5",image_paths)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to
be built. `model.compile_metrics` will be empty until you train or evaluate
the model.

1/1 ━━━━━━━━━━━━━━━━━━━━2s 2s/step
```



Model	Mean Absolute Error
Custom CNN model 1	6.466558933258057
customCNNmodel2	6.37539005279541
ResNet50	5.554823398590088
VGG16 1	2.340360403060913
VGG16 2	6.3198

Conclusion:

As seen in the previous table, VGG16 version 1 got the lowest MAE.