Model weights are

1. Setup and Import Libraries

```python

import numpy as np

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

import matplotlib.pyplot as plt

import cv2

import os

```

2. Data Loading and Preprocessing

#Assuming you have a dataset where the images are stored in directories named after the ages.

```python

# Define paths

dataset\_path = '/path/to/your/dataset'

image\_size = (224, 224)

batch\_size = 32

#Data augmentation and normalization

datagen = ImageDataGenerator(

rescale=1./255,

validation\_split=0.2,

horizontal\_flip=True,

zoom\_range=0.2

)

# Load data with ImageDataGenerator

train\_generator = datagen.flow\_from\_directory(

dataset\_path,

target\_size=image\_size,

batch\_size=batch\_size,

class\_mode='sparse',

subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

dataset\_path,

target\_size=image\_size,

batch\_size=batch\_size,

class\_mode='sparse',

subset='validation'

)

```

3. Model Building

Using Transfer Learning with ResNet50 as the base model.

```python

# Load the ResNet50 model with pre-trained weights

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Freeze the base model layers

for layer in base\_model.layers:

layer.trainable = False

# Add custom layers on top of ResNet50

model = models.Sequential([

base\_model,

layers.GlobalAveragePooling2D(),

layers.Dense(1024, activation='relu'),

layers.Dropout(0.5),

layers.Dense(1, activation='linear') # Regression for age prediction

])

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.0001),

loss='mean\_squared\_error',

metrics=['mean\_absolute\_error'])

model.summary()

```

4. Model Training

```python

# Train the model

history = model.fit(

train\_generator,

validation\_data=validation\_generator,

epochs=10

)

```

5. Saving Model Weights

After training, save the model weights to a file.

```python

# Save the model weights

model.save\_weights('age\_detection\_model\_weights.h5')

```

6. Loading Model Weights

When you need to use the model later, load the saved weights into the same model architecture.

```python

# Rebuild the model architecture

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

for layer in base\_model.layers:

layer.trainable = False

model = models.Sequential([

base\_model,

layers.GlobalAveragePooling2D(),

layers.Dense(1024, activation='relu'),

layers.Dropout(0.5),

layers.Dense(1, activation='linear') # Regression for age prediction

])

# Compile the model

model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.0001),

loss='mean\_squared\_error',

metrics=['mean\_absolute\_error'])

# Load the model weights

model.load\_weights('age\_detection\_model\_weights.h5')

```

7. Evaluation and Visualization

Evaluate the model and visualize the training process.

```python

# Evaluate on the validation set

val\_loss, val\_mae = model.evaluate(validation\_generator)

print(f'Validation MAE: {val\_mae:.2f}')

# Plot training & validation loss and MAE

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.legend()

plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)

plt.plot(history.history['mean\_absolute\_error'], label='Training MAE')

plt.plot(history.history['val\_mean\_absolute\_error'], label='Validation MAE')

plt.legend()

plt.title('Training and Validation MAE')

plt.show()

```

8. Real-Time Age Detection (Optional)

You can add a real-time detection system using OpenCV:

```python

# Real-time age prediction

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if not ret:

break

# Preprocess the image

img = cv2.resize(frame, image\_size)

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

img = img / 255.0

# Predict age

predicted\_age = model.predict(img)

predicted\_age = int(predicted\_age[0][0])

# Display the result

cv2.putText(frame, f'Predicted Age: {predicted\_age}', (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

cv2.imshow('Age Detection', frame)

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

break

cap.release()

cv2.destroyAllWindows()

```

9. Saving and Loading Full Model (Alternative)

Alternatively, you can save and load the entire model, including architecture and weights, using `model.save()` and `tf.keras.models.load\_model()`.

```python

# Save the full model

model.save('age\_detection\_full\_model.h5')

# Load the full model

model = tf.keras.models.load\_model('age\_detection\_full\_model.h5')

```