Mini Project : 2

Title - Implement Gender and Age Detection: predict if a person is a male or female and also their age

Theory -

Collect and prepare the dataset: In this case, we can use the "UTKFace" dataset which contains images of faces with their corresponding gender and age labels. We need to preprocess the data, like resizing the images to a uniform size, shuffling the dataset, and limit the age to a certain value (like 100 years).
Split the dataset: Split the dataset into training, validation, and testing sets. The usual split ratio is 80%, 10%, and 10%, respectively.
Define data generators: Define data generators for training, validation, and testing sets using the "ImageDataGenerator" class in Keras. This class provides data augmentation techniques that can improve the model's performance, such as rotation, zoom, and horizontal flip.
Define the neural network model: Define a convolutional neural network (CNN) model that takes the face images as input and outputs two values - the probability of being male and the predicted age. The model can have multiple convolutional and pooling layers followed by some dense layers.
Compile the model: Compile the model with appropriate loss and metrics for each output (gender and age). In this case, we can use binary cross-entropy loss for gender and mean squared error (MSE) for age.
Train the model: Train the model using the fit method of the model object. We need to pass the data generators for the training and validation sets, as well as the number of epochs and batch size.
Evaluate the model: Evaluate the model's performance on the testing set using the evaluate method of the model object. This will give us the accuracy and mean absolute error (MAE) of the model.
Predict the gender and age of a sample image: Load a sample image and preprocess it. We can use the "cv2" library to read the image, resize it to the same size as the training images, and normalize it. Then, we can use the "predict" method of the model object to get the predicted gender and age.

Source Code -

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
# Define constants
img_height = 128
img_width = 128
batch_size = 32
epochs = 10
# Load the "UTKFace" dataset
df = pd.read_csv('UTKFace.csv')
df['age'] = df['age'].apply(lambda x: min(x, 100)) # limit age to 100
df = df.sample(frac=1).reset_index(drop=True) # shuffle the dataset
df['image_path'] = 'UTKFace/' + df['image_path']
df_train = df[:int(len(df)*0.8)] # 80% for training
df_val = df[int(len(df)*0.8):int(len(df)*0.9)] # 10% for validation
df_test = df[int(len(df)*0.9):] # 10% for testing
# Define data generators for training, validation, and testing sets
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_dataframe(
dataframe=df_train,
x_col='image_path',
y_col=['male', 'age'],
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='raw')
val_generator = val_datagen.flow_from_dataframe(
dataframe=df_val,
x_col='image_path',
y_col=['male', 'age'],
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='raw')
test_generator = test_datagen.flow_from_dataframe(
dataframe=df_test,
x_col='image_path',
y_col=['male', 'age'],
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='raw')
# Define the neural network model
model = Sequential([
Conv2D(32, (3,3), activation='relu', input_shape=(img_height, img_width, 3)),
MaxPooling2D((2,2)),
Conv2D(64, (3,3), activation='relu'),
MaxPooling2D((2,2)),
Conv2D(128, (3,3), activation='relu'),
MaxPooling2D((2,2)),
Conv2D(128, (3,3), activation='relu'),
MaxPooling2D((2,2)),
Flatten(),
Dropout(0.5),
Dense(512, activation='relu'),
Dense(2)
])
# Compile the model
model.compile(optimizer='adam',
loss={'dense_1': 'binary_crossentropy', 'dense_2': 'mse'},
metrics={'dense_1': 'accuracy', 'dense_2': 'mae'})
# Train the model
history = model.fit(train_generator,
epochs=epochs,
validation_data=val_generator)
# Evaluate the model on the test set
loss, accuracy, mae = model.evaluate(test_generator)
print("Test accuracy:", accuracy)
print("Test MAE:", mae)
# Predict the gender and age of a sample image
img = cv2.imread('sample_image.jpg')
img
```

Conclusion- In this way Gender Age Detection Implemented.