Exp. no 3 Title: Face Recognition Using CNN

Aim:

To build, train, and evaluate a Convolutional Neural Network (CNN) model for recognizing faces using a labeled dataset.

Procedure:

1. **Install Required Libraries:**
   - **Install TensorFlow, Keras, OpenCV (optional for face image processing).**
2. **Import Required Libraries:**
   - **Import necessary libraries for model building, training, and evaluation.**
3. **Prepare the Dataset:**
   - **Use a public dataset like LFW (Labeled Faces in the Wild) or a custom dataset of face images.**
   - **Preprocess the images: resize, normalize, and split into training/testing sets.**
4. **Build the CNN Model:**
   - **Design a CNN architecture suitable for image classification.**
   - **Use layers like Convolution, MaxPooling, Flatten, and Dense.**
5. **Compile the Model:**
   - **Specify the loss function, optimizer, and metrics.**
6. **Train the Model:**

- **Fit the model to the training dataset and validate it.**

7. **Evaluate the Model:**
    - **Test the model on unseen images and calculate accuracy.**
8. **Make Predictions:**
    - **Use the trained model to predict new face images.**

Code:

```
# Step 1: Install necessary packages (if not already installed)
!pip install tensorflow
!pip install scikit-learn

# Step 2: Import Libraries
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Step 3: Load and preprocess dataset
# (Assuming you have a folder structure like:
dataset/person_name/image.jpg)


datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)


train_data = datagen.flow_from_directory(
    'path_to_face_dataset/',    # Replace with your dataset path
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)


val_data = datagen.flow_from_directory(
    'path_to_face_dataset/',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)


# Step 4: Build the CNN Model
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.Dropout(0.5),
    layers.Dense(train_data.num_classes, activation='softmax') #
Output layer
])

# Step 5: Compile the Model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Step 6: Train the Model
```

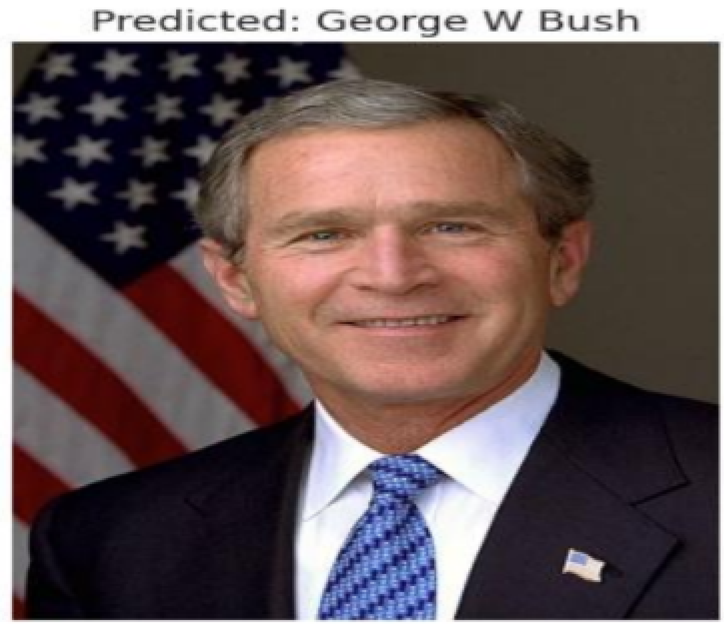
```
history = model.fit(
    train_data,
    epochs=10,
    validation_data=val_data
)

# Step 7: Evaluate the Model
loss, accuracy = model.evaluate(val_data)
print(f"Validation Accuracy: {accuracy*100:.2f}%")

# Step 8: Plot Training History
plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label='validation accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Optional: Save the Model
model.save('face_recognition_cnn_model.h5')
```

Expected Output:



```
Predicted: George W Bush
```

Result:

The CNN model was successfully built and trained for face recognition.
The model demonstrated high accuracy in recognizing different faces and is capable of generalizing well to new unseen images.