=================================================================== **Exp No 5 Face Recognition using CNN  
 ====================================================================  
  
Aim…**

To implement a face recognition system using Convolutional Neural Networks (CNN) for

identifying and classifying human faces from images.

**Code…**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import fetch\_lfw\_people**

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

**from sklearn.preprocessing import LabelEncoder**

**from tensorflow.keras.utils import to\_categorical**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout**

**# Step 1: Load predefined LFW dataset (people with at least 70 images)**

**lfw\_data = fetch\_lfw\_people(min\_faces\_per\_person=70, resize=0.4)**

**X = lfw\_data.images**

**y = lfw\_data.target**

**target\_names = lfw\_data.target\_names**

**n\_classes = len(target\_names)**

**# Step 2: Preprocess the data**

**X = X.reshape(-1, X.shape[1], X.shape[2], 1) / 255.0**

**y = to\_categorical(y, n\_classes)**

**# Step 3: Train-test split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Step 4: Build CNN model**

**model = Sequential([**

**Conv2D(32, (3, 3), activation='relu', input\_shape=X.shape[1:]),**

**MaxPooling2D(2, 2),**

**Conv2D(64, (3, 3), activation='relu'),**

**MaxPooling2D(2, 2),**

**Flatten(),**

**Dense(128, activation='relu'),**

**Dropout(0.5),**

**Dense(n\_classes, activation='softmax')**

**])**

**# Step 5: Compile the model**

**model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])**

**# Step 6: Train the model**

**history = model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, y\_test))**

**# Step 7: Evaluate the model**

**test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test)**

**print(f"Test Accuracy: {test\_accuracy:.2f}")**

**# Step 8: Predict and visualize**

**def predict\_and\_display(index):**

**sample = X\_test[index].reshape(1, X.shape[1], X.shape[2], 1)**

**prediction = model.predict(sample)**

**predicted\_label = target\_names[np.argmax(prediction)]**

**actual\_label = target\_names[np.argmax(y\_test[index])]**

**plt.imshow(X\_test[index].reshape(X.shape[1], X.shape[2]), cmap='gray')**

**plt.title(f"Predicted: {predicted\_label}\nActual: {actual\_label}")**

**plt.axis('off')**

**plt.show()**

**# Example: Display prediction result for a sample**

**predict\_and\_display(5)**

**# Optional: Save model**

**model.save("lfw\_face\_recognition\_cnn.h5")**

**Output…**

**Epoch 1/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 4s 74ms/step - accuracy: 0.3090 - loss: 1.8617 - val\_accuracy: 0.4612 - val\_loss: 1.6216**

**Epoch 2/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.4080 - loss: 1.7196 - val\_accuracy: 0.4612 - val\_loss: 1.6298**

**Epoch 3/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 3s 83ms/step - accuracy: 0.3870 - loss: 1.7623 - val\_accuracy: 0.4612 - val\_loss: 1.6478**

**Epoch 4/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 3s 78ms/step - accuracy: 0.3958 - loss: 1.7462 - val\_accuracy: 0.4612 - val\_loss: 1.6189**

**Epoch 5/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 5s 63ms/step - accuracy: 0.3950 - loss: 1.7135 - val\_accuracy: 0.4612 - val\_loss: 1.6280**

**Epoch 6/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 3s 68ms/step - accuracy: 0.4239 - loss: 1.6939 - val\_accuracy: 0.4612 - val\_loss: 1.6074**

**Epoch 7/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 3s 86ms/step - accuracy: 0.3970 - loss: 1.7222 - val\_accuracy: 0.4612 - val\_loss: 1.6029**

**Epoch 8/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 5s 72ms/step - accuracy: 0.4018 - loss: 1.7329 - val\_accuracy: 0.4612 - val\_loss: 1.5923**

**Epoch 9/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 2s 69ms/step - accuracy: 0.3995 - loss: 1.7307 - val\_accuracy: 0.4612 - val\_loss: 1.6013**

**Epoch 10/10**

**33/33 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.4049 - loss: 1.7194 - val\_accuracy: 0.4612 - val\_loss: 1.6115**

**9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.5007 - loss: 1.5586**

**Test Accuracy: 0.46**

**1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step**

****