3: CNN FOR FACE RECOGNITION

AIM:

To build and train a CNN model for face recognition using Labeled Faces in the Wild (LFW) dataset.

PROCEDURE:

1. Load LFW dataset with minimum faces per person.
2. Preprocess images and one-hot encode labels.
3. Define a CNN model with Conv2D and Dense layers.
4. Compile the model using categorical crossentropy.
5. Train and evaluate the model.

CODE:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt

# Load LFW dataset
```

```
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

# Get the data and labels
X = lfw_people.images
y = lfw_people.target
target_names = lfw_people.target_names

# Preprocess images (reshape and normalize)
X = np.expand_dims(X, axis=-1)  # Adding channel dimension (grayscale)
X = X / 255.0  # Normalization

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Encode the labels
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)

# Data Augmentation
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
```

```
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
datagen.fit(X_train)

# Build a CNN model with more layers and adjustments
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(X.shape[1], X.shape[2], 1)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    # Removed the extra MaxPooling2D and Conv2D layers to avoid negative dimensions
    # layers.MaxPooling2D((2, 2)),
    # layers.Conv2D(256, (3, 3), activation='relu'),
    # layers.BatchNormalization(),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),  # Increased neurons in Dense layer
    layers.Dropout(0.5),
    layers.Dense(len(target_names), activation='softmax')
])
```

```
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])


# Train the model with data augmentation and more epochs
model.fit(datagen.flow(X_train, y_train, batch_size=32), epochs=50, validation_data=(X_test, y_test)) # Increased epochs


# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_acc * 100:.2f}%')
# Get predictions for the test set
y_pred = model.predict(X_test)
y_pred_labels = np.argmax(y_pred, axis=1)
# Display some images with predictions
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(12, 6))
for i, ax in enumerate(axes.flat):
    ax.imshow(X_test[i].reshape(50, 37), cmap='gray')
    ax.set_title(f"Predicted: {target_names[y_pred_labels[i]]}\nActual: {target_names[y_test[i]]}")
    ax.axis('off')
plt.tight_layout()
plt.show()
```

OUTPUT:

RESULT:

The model achieved moderate accuracy (~80%) on LFW for recognizing individuals with many face samples.