# Step 1: Import Libraries

import os

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification\_report, confusion\_matrix

# Step 2: Define Paths

train\_dir = 'dataset/train'

val\_dir = 'dataset/val'

test\_dir = 'dataset/test'

# Step 3: Image Preprocessing

img\_height, img\_width = 150, 150

batch\_size = 32

train\_datagen = ImageDataGenerator(rescale=1./255, rotation\_range=20, zoom\_range=0.2,

horizontal\_flip=True, shear\_range=0.2)

val\_datagen = ImageDataGenerator(rescale=1./255)

train\_data = train\_datagen.flow\_from\_directory(train\_dir, target\_size=(img\_height, img\_width),

batch\_size=batch\_size, class\_mode='categorical')

val\_data = val\_datagen.flow\_from\_directory(val\_dir, target\_size=(img\_height, img\_width),

batch\_size=batch\_size, class\_mode='categorical')

# Step 4: CNN 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),

Flatten(),

Dropout(0.5),

Dense(256, activation='relu'),

Dense(train\_data.num\_classes, activation='softmax')

])

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

model.summary()

# Step 5: Train the Model

history = model.fit(train\_data, validation\_data=val\_data, epochs=10)

# Step 6: Evaluate on Test Data (optional)

test\_data = val\_datagen.flow\_from\_directory(test\_dir, target\_size=(img\_height, img\_width),

batch\_size=batch\_size, class\_mode='categorical', shuffle=False)

predictions = model.predict(test\_data)

y\_pred = np.argmax(predictions, axis=1)

print(classification\_report(test\_data.classes, y\_pred, target\_names=list(test\_data.class\_indices.keys())))

# Step 7: Save the Model

model.save('butterfly\_classifier\_model.h5')