import os

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing import image, ImageDataGenerator

from tensorflow.keras.models import Sequential

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

# SETTINGS

IMG\_SIZE = (150, 150)

BATCH\_SIZE = 32

EPOCHS = 10

DATASET\_PATH = "dataset" # Folder with subfolders like /floral, /striped, etc.

# LOAD AND PREPROCESS DATA

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2,

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

train\_data = datagen.flow\_from\_directory(DATASET\_PATH, target\_size=IMG\_SIZE,

batch\_size=BATCH\_SIZE, class\_mode='categorical',

subset='training')

val\_data = datagen.flow\_from\_directory(DATASET\_PATH, target\_size=IMG\_SIZE,

batch\_size=BATCH\_SIZE, class\_mode='categorical',

subset='validation')

# MODEL DEFINITION

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(\*IMG\_SIZE, 3)),

MaxPooling2D(2, 2),

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

MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

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

])

# COMPILE & TRAIN

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

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

# PLOT RESULTS

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label="Train")

plt.plot(history.history['val\_accuracy'], label="Val")

plt.title("Accuracy")

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label="Train")

plt.plot(history.history['val\_loss'], label="Val")

plt.title("Loss")

plt.legend()

plt.show()

# SAVE MODEL

model.save("fabric\_pattern\_model.h5")

# PREDICT SAMPLE IMAGE

test\_img\_path = "sample.jpg" # Replace with actual test image path

if os.path.exists(test\_img\_path):

img = image.load\_img(test\_img\_path, target\_size=IMG\_SIZE)

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

prediction = model.predict(img\_array)

class\_names = list(train\_data.class\_indices.keys())

print("Predicted Pattern:", class\_names[np.argmax(prediction)])

else:

print("Test image not found. Skipping prediction.")