Fabric Pattern Classification using CNN (TensorFlow & Keras)

This Python code implements a deep learning model using a Convolutional Neural Network (CNN)

to classify different fabric patterns such as floral, striped, checked, dotted, and plain.

Directory structure:

fabric_dataset/

??? floral/

??? striped/

??? checked/

??? dotted/

??? plain/

```
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

# Paths
train_dir = 'fabric_dataset/train'
val_dir = 'fabric_dataset/val'

# Image preprocessing and augmentation
image_size = (150, 150)
batch_size = 32

train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,
                                   width_shift_range=0.2, height_shift_range=0.2,
                                                          shear_range=0.2,  zoom_range=0.2,
horizontal_flip=True)

val_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(train_dir,
                                               target_size=image_size,
                                               batch_size=batch_size,
                                               class_mode='categorical')

val_data = val_datagen.flow_from_directory(val_dir,
                                           target_size=image_size,
                                           batch_size=batch_size,
                                           class_mode='categorical')
```

```
# CNN Model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 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(128, activation='relu'),
    Dense(train_data.num_classes, activation='softmax')
])

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

# Train
history = model.fit(train_data,
                    epochs=15,
                    validation_data=val_data)

# Save the model
model.save("fabric_pattern_classifier.h5")

print("Model training complete and saved.")
```