7. Implement the standard VGG-16 & 19 CNN architecture model to classify multi category

image dataset and check the accuracy

```
import tensorflow as tf
from tensorflow.keras.applications import VGG16, VGG19
from tensorflow.keras import layers, models

# Path to your local dataset
data_dir = "C:/Users/Sahyadri/.keras/datasets/flower_photos"

# Load dataset directly from directory
ds_train = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(224, 224),
    batch_size=32
)

ds_test = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(224, 224),
    batch_size=32
)

# Normalize and one-hot encode labels
normalization_layer = tf.keras.layers.Rescaling(1./255)
ds_train = ds_train.map(lambda x, y: (normalization_layer(x), tf.one_hot(y,
depth=5)))
ds_test = ds_test.map(lambda x, y: (normalization_layer(x), tf.one_hot(y,
depth=5)))

ds_train = ds_train.prefetch(tf.data.AUTOTUNE)
ds_test = ds_test.prefetch(tf.data.AUTOTUNE)


# Function to create VGG model
def create_vgg_model(vgg_model_class):
    base_model = vgg_model_class(weights='imagenet',
                                 include_top=False,
                                 input_shape=(224, 224, 3))
    base_model.trainable = False

    return models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(5, activation='softmax')
    ])

# Function to train and evaluate
def train_and_evaluate(model_class, model_name):
    print(f"\nUsing {model_name}:")
    model = create_vgg_model(model_class)
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    model.fit(ds_train, validation_data=ds_test, epochs=10)
    loss, acc = model.evaluate(ds_test)
    print(f"{model_name} Accuracy: {acc*100:.2f}% | Loss: {loss:.4f}")
    return model

# Train both models
vgg16_model = train_and_evaluate(VGG16, "VGG16")
vgg19_model = train_and_evaluate(VGG19, "VGG19")
```