CODE FOR THE POLLEN'S PROFILLING: AUTOMATED CLASSIFICATION OF POLLEN GRAINS

Technologies Used

- Python
- TensorFlow / Keras
- ImageDataGenerator
- CNN
- Classification

Project Structure (Example):

pollen_project/

│

├── dataset/

│ ├── daisy/

│ ├── sunflower/

│ ├── rose/

│ └── dandelion/

│

├── train_model.py

└── classify_pollen.py

Step 1: Train the Model (train_model.py)

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Paths
DATASET_DIR = "dataset/"
IMG_HEIGHT, IMG_WIDTH = 128, 128
BATCH_SIZE = 32
EPOCHS = 15

# Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
```

```
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

# 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(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(train_generator.num_classes, activation='softmax')
])
```

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

# Save Best Model
checkpoint = ModelCheckpoint("best_model.h5", monitor='val_accuracy', save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=3)

# Train
model.fit(
  train_generator,
  epochs=EPOCHS,
  validation_data=val_generator,
  callbacks=[checkpoint, early_stop]
)
```

Step 2: Predict New Images (classify_pollen.py)

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing import image
import os

# Load model
model = tf.keras.models.load_model("best_model.h5")

# Class Labels
labels = ['daisy', 'dandelion', 'rose', 'sunflower']  # Modify based on your folders

def classify_image(img_path):
    img = image.load_img(img_path, target_size=(128, 128))
    img_tensor = image.img_to_array(img)
    img_tensor = np.expand_dims(img_tensor, axis=0)
    img_tensor /= 255.

    prediction = model.predict(img_tensor)
    class_index = np.argmax(prediction)
    confidence = prediction[0][class_index]
    return labels[class_index], confidence

# Example usage
test_image = "test/pollen.jpg"
predicted_class, confidence = classify_image(test_image)
print(f"Predicted Class: {predicted_class} ({confidence:.2f})")
```