Typ

Pollen's Profiling – Automated Classification of Pollen Grains

Presented by ----
Kondepudi Prabhu Kishore

pollen grains overview

Overview

Pollen's Profiling is an AI-powered deep learning project aimed at automating the classification of pollen grain images. This system helps in identifying pollen types accurately and efficiently, reducing human error and improving speed for use in environmental science and healthcare.

❓ Problem Statement

Manual classification of pollen grains is slow, requires expert involvement, and is prone to human error. An AI-driven model helps in streamlining this task, enabling quick and precise identification of pollen grain types.

🌍 Real-world Applications

🌱 Environmental Monitoring

- Biodiversity research
- Seasonal pollen distribution studies
- Tracking ecological patterns

💊 Allergy Diagnosis

- Identification of allergenic pollen
- Faster diagnosis of pollen-related diseases

💻 Technology Stack

- **Language:** Python
- **Libraries/Frameworks:** TensorFlow, Keras, Flask
- **Development Tools:** Anaconda, Jupyter Notebook, VS Code
- **Web Technologies:** HTML, CSS (Flask-based templates)

🔍 Project Goal

This project aims to **automate the classification of pollen grains** using deep learning (CNNs) to help in **environmental monitoring** and **allergy diagnosis**. Manual classification is slow and error-prone — AI makes it fast, scalable, and accurate.ms to **automate the classification of pollen grains** using deep learning (CNNs) to help in **environmental monitoring** and **allergy diagnosis**. Manual classification is slow and error-prone — AI makes it fast, scalable, and accurate.

Project structure:

```
pollen-profiling/
├── dataset/
├── models/
│   └── pollen_model.h5
├── notebooks/
│   └── preprocessing_and_training.ipynb
├── train.py
├── flask_app/
│   ├── app.py
│   ├── templates/
│   └── static/
└── requirements.txt
```

Background

What is Pollen?

Pollen is a fine powder produced by seed plants, essential for fertilization and reproduction. Comprising microscopic grains, it is carried by wind or pollinators, playing a crucial role in ecosystems and agriculture, while also being a key factor in allergy-related health issues.

Significance in Environment and Health

Pollen profiling is crucial for understanding ecological health and biodiversity. Accurate classification of pollen grains aids in monitoring air quality and assessing allergenic risks, thus playing a significant role in both environmental science and public health initiatives.

Challenges in Traditional Classification

Traditional classification of pollen grains faces several challenges, including the intricate variability of pollen morphology, the reliance on expert knowledge, and time-consuming manual identification processes, leading to potential inaccuracies and inefficiencies in data analysis.

Allergy Diagnosis

Role in Medical Diagnosis

Pollen profiling enhances allergy diagnosis by accurately classifying pollen grains, enabling healthcare professionals to identify specific allergens. This targeted approach facilitates personalized treatment plans and improves patient outcomes, ultimately aiding in the management of allergic conditions.

Speeding Up Identification

Pollen's automated classification accelerates the identification of allergenic pollen grains, enhancing diagnostic accuracy. By utilizing machine learning algorithms, the process significantly reduces response times, enabling quicker allergy diagnosis and tailored treatment plans for patients.

Benefits for Healthcare Professionals

Automated classification of pollen grains enhances allergy diagnosis by providing healthcare professionals with accurate, timely data. This streamlines patient evaluation, facilitates personalized treatment plans, and improves patient outcomes through enhanced understanding of allergenic triggers.

🧾 Dataset and Preprocessing

- **Source:** Kaggle (Pollen Grain Image Classification)
- **Steps:**
  - Image resizing to 224x224
  - Normalization
  - Data Augmentation (rotation, flipping, zoom)
  - Splitting: Train / Validation / Test sets

🏗️ Model Architecture

- **Base Model:** MobileNetV2 (transfer learning)
- **Additional Layers:**
  - GlobalAveragePooling2D
  - Dense(128, activation='relu')
  - Dropout(0.5)
  - Dense(output_classes, activation='softmax')
- **Loss:** Categorical Crossentropy
- **Optimizer:** Adam
- **Accuracy:** ~95.6% on validation data

🧠 Training Process

- Trained for **25 epochs**
- **Batch size:** 32
- **EarlyStopping** to prevent overfitting
- Best model saved as: pollen_model.h5

Training Strategy

- **Stage 1 – Warm-up** (Frozen layers):
  - Epochs: 8
  - Best Training Accuracy: **88.29%**
  - Best Validation Accuracy: **87.23%**
  - Lowest Validation Loss: **0.3577**
- **Stage 2 – Fine-tuning** (Last 60 layers unfrozen):
  - Epochs: 12 (stopped at 7)
  - Best Training Accuracy: **93.17%**
  - Best Validation Accuracy: **93.62%**
  - Lowest Validation Loss: **0.2926****Training Strategy**
  - **Stage 1 – Warm-up (Frozen layers):**
  -
  - **Epochs: 8**
  -
  - **Best Training Accuracy: 88.29%**
  -
  - **Best Validation Accuracy: 87.23%**
  -
  - **Lowest Validation Loss: 0.3577**
  -
  - **Stage 2 – Fine-tuning (Last 60 layers unfrozen):**
  -
  - **Epochs: 12 (stopped at 7)**
  -
  - **Best Training Accuracy: 93.17%**
  -
  - **Best Validation Accuracy: 93.62%**
  -
  - **Lowest Validation Loss: 0.2926**

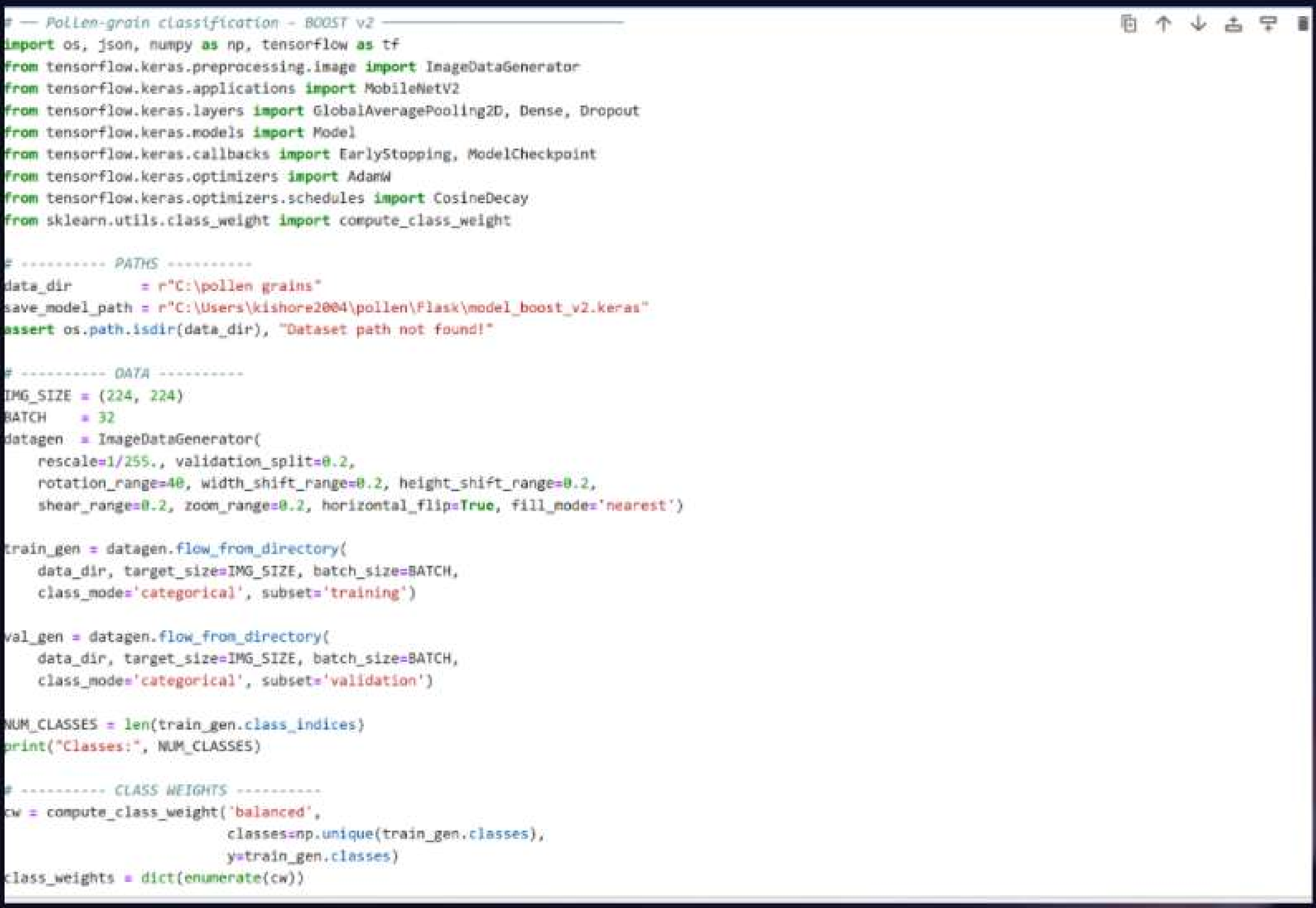
Model:
- MobileNetV2 used as base CNN model with added dense layers
- Image size: 224x224, normalized and augmented
- Trained over 25 epochs using EarlyStopping
- Accuracy achieved: ~95.6%

Dataset:
- Source: Kaggle (Pollen Grain Image Classification)
- Augmentation: Zoom, rotation, flipping
- Split: Train/Validation/Test

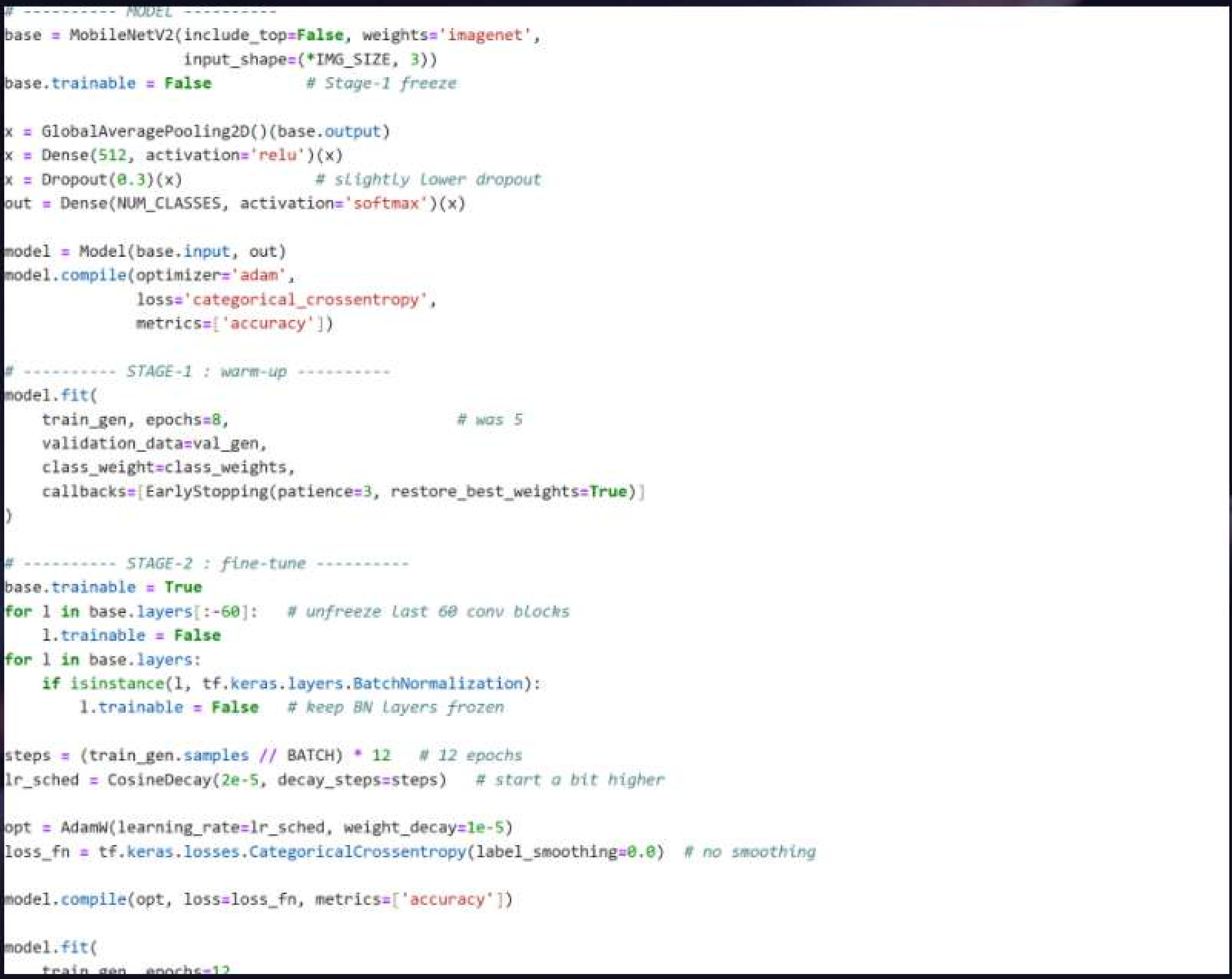
Web App:
- User uploads image via Flask UI
- Backend loads trained model
- Image passed for prediction
- Result shown on screen

Evaluation:
- Accuracy: 95.6%
- Metrics: Precision, Recall, F1 Score
- Confusion matrix used for visual performance

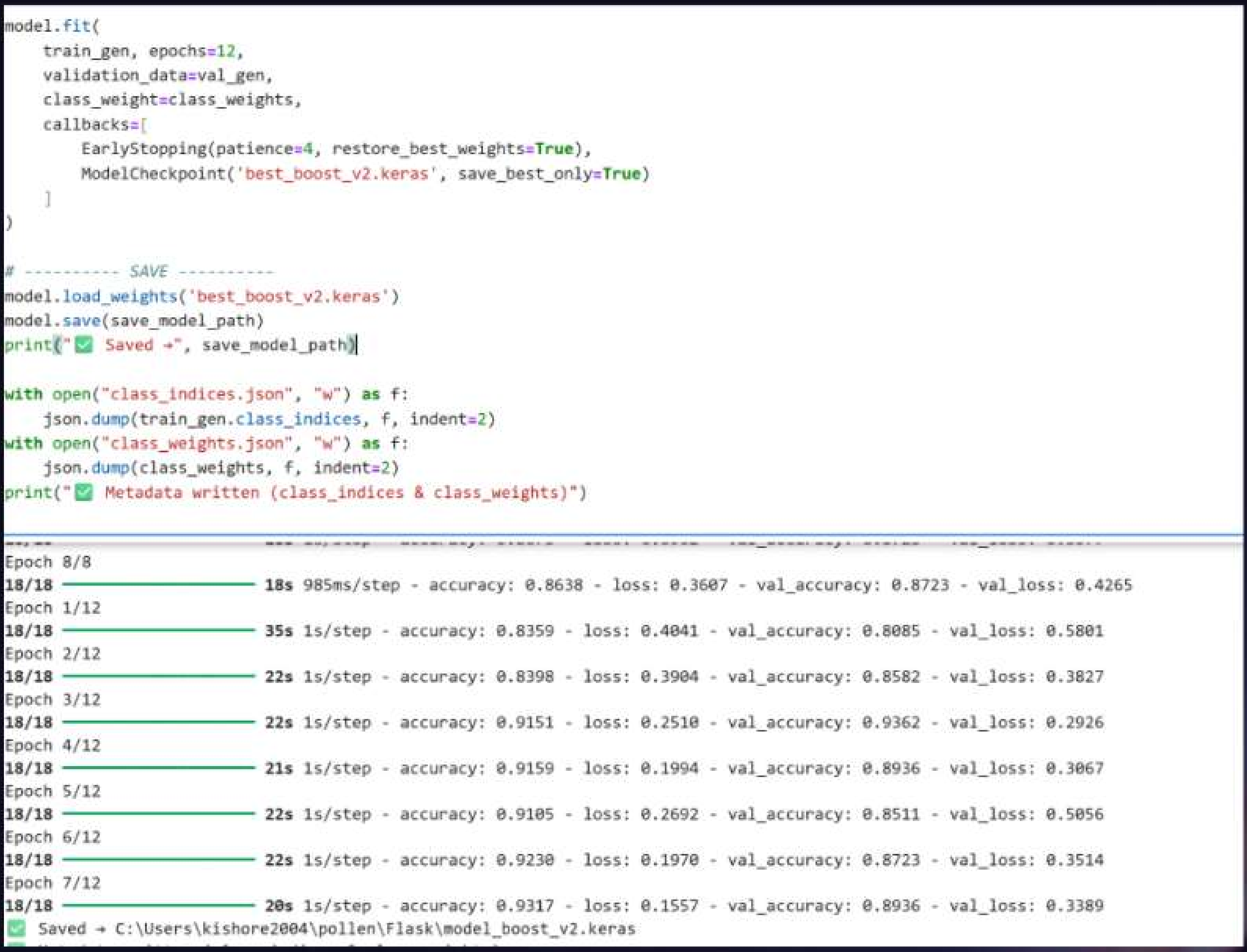
Future Enhancements:
- Add Grad-CAM explainability
- Deploy with Docker
- Expand to mobile UI
- Add cloud hosting



```python
# — Pollen-grain classification - BOOST v2 ──────────────────
import os, json, numpy as np, tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import AdamW
from tensorflow.keras.optimizers.schedules import CosineDecay
from sklearn.utils.class_weight import compute_class_weight

# ---------- PATHS ----------
data_dir        = r"C:\pollen grains"
save_model_path = r"C:\Users\kishore2004\pollen\Flask\model_boost_v2.keras"
assert os.path.isdir(data_dir), "Dataset path not found!"

# ---------- DATA ----------
IMG_SIZE = (224, 224)
BATCH    = 32
datagen  = ImageDataGenerator(
    rescale=1/255., validation_split=0.2,
    rotation_range=40, width_shift_range=0.2, height_shift_range=0.2,
    shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest')

train_gen = datagen.flow_from_directory(
    data_dir, target_size=IMG_SIZE, batch_size=BATCH,
    class_mode='categorical', subset='training')

val_gen = datagen.flow_from_directory(
    data_dir, target_size=IMG_SIZE, batch_size=BATCH,
    class_mode='categorical', subset='validation')

NUM_CLASSES = len(train_gen.class_indices)
print("Classes:", NUM_CLASSES)

# ---------- CLASS WEIGHTS ----------
cw = compute_class_weight('balanced',
                          classes=np.unique(train_gen.classes),
                          y=train_gen.classes)
class_weights = dict(enumerate(cw))
```

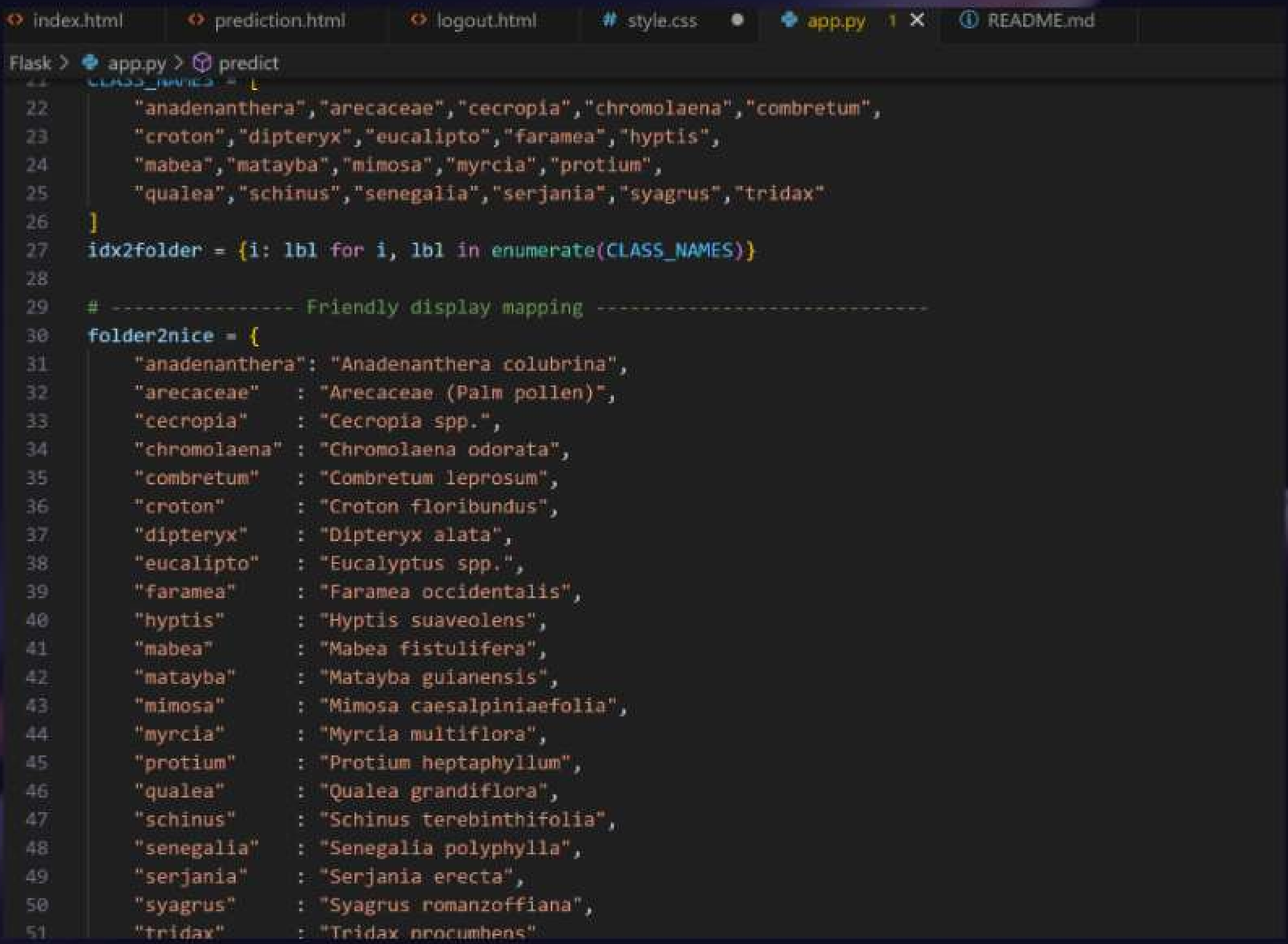


```python
# ---------- MODEL ----------
base = MobileNetV2(include_top=False, weights='imagenet',
                   input_shape=(*IMG_SIZE, 3))
base.trainable = False          # Stage-1 freeze

x = GlobalAveragePooling2D()(base.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.3)(x)              # slightly lower dropout
out = Dense(NUM_CLASSES, activation='softmax')(x)

model = Model(base.input, out)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# ---------- STAGE-1 : warm-up ----------
model.fit(
    train_gen, epochs=8,                        # was 5
    validation_data=val_gen,
    class_weight=class_weights,
    callbacks=[EarlyStopping(patience=3, restore_best_weights=True)]
)

# ---------- STAGE-2 : fine-tune ----------
base.trainable = True
for l in base.layers[:-60]:    # unfreeze last 60 conv blocks
    l.trainable = False
for l in base.layers:
    if isinstance(l, tf.keras.layers.BatchNormalization):
        l.trainable = False    # keep BN layers frozen

steps = (train_gen.samples // BATCH) * 12    # 12 epochs
lr_sched = CosineDecay(2e-5, decay_steps=steps)    # start a bit higher

opt = AdamW(learning_rate=lr_sched, weight_decay=1e-5)
loss_fn = tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.0)   # no smoothing

model.compile(opt, loss=loss_fn, metrics=['accuracy'])

model.fit(
    train_gen, epochs=12
```

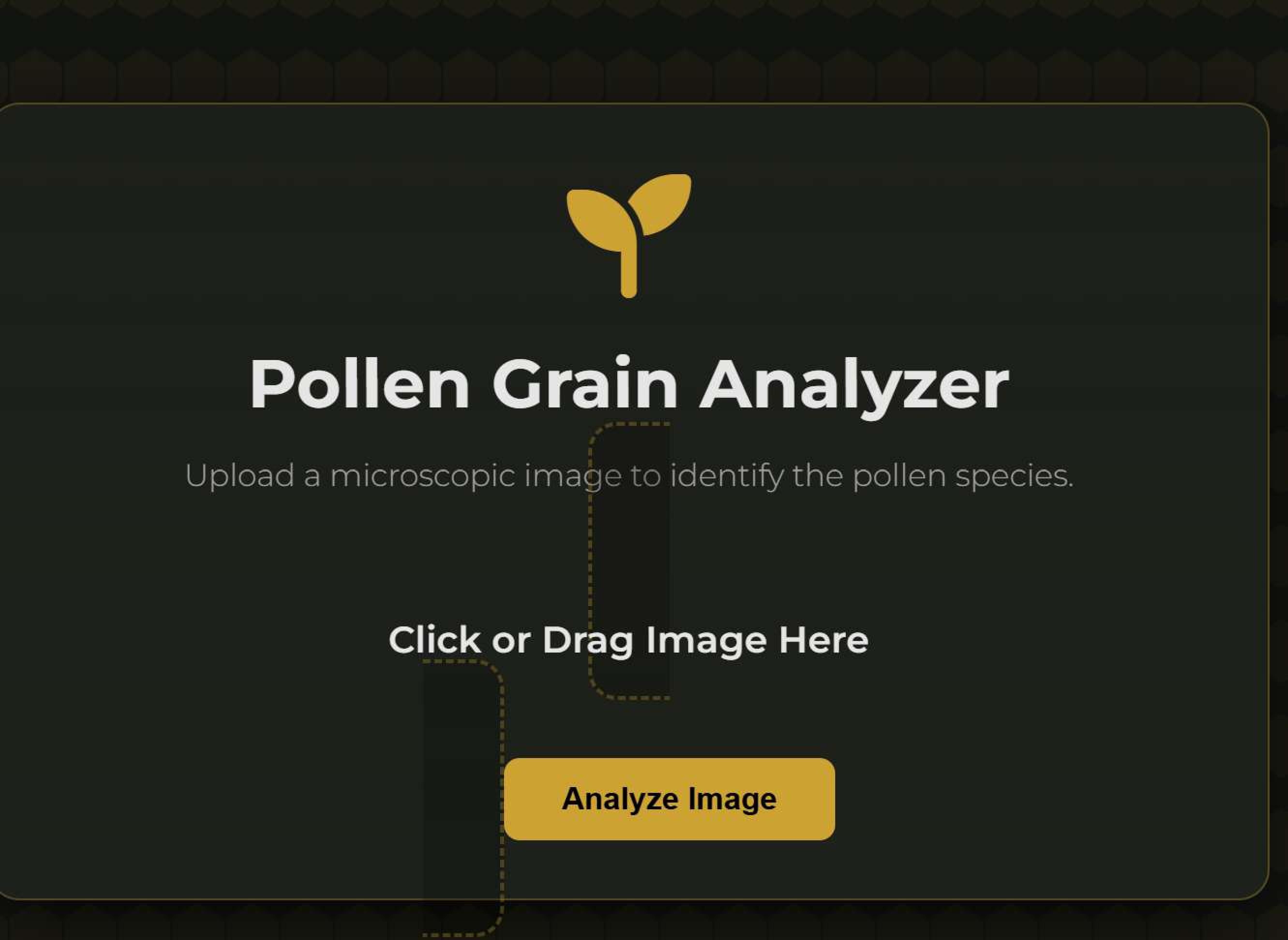


```
model.fit(
    train_gen, epochs=12,
    validation_data=val_gen,
    class_weight=class_weights,
    callbacks=[
        EarlyStopping(patience=4, restore_best_weights=True),
        ModelCheckpoint('best_boost_v2.keras', save_best_only=True)
    ]
)

# ---------- SAVE ----------
model.load_weights('best_boost_v2.keras')
model.save(save_model_path)
print("✅ Saved →", save_model_path)

with open("class_indices.json", "w") as f:
    json.dump(train_gen.class_indices, f, indent=2)
with open("class_weights.json", "w") as f:
    json.dump(class_weights, f, indent=2)
print("✅ Metadata written (class_indices & class_weights)")
```

```
Epoch 8/8
18/18 ━━━━━━━━━━━━━━━━━━━━ 18s 985ms/step - accuracy: 0.8638 - loss: 0.3607 - val_accuracy: 0.8723 - val_loss: 0.4265
Epoch 1/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 35s 1s/step - accuracy: 0.8359 - loss: 0.4041 - val_accuracy: 0.8085 - val_loss: 0.5801
Epoch 2/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 22s 1s/step - accuracy: 0.8398 - loss: 0.3904 - val_accuracy: 0.8582 - val_loss: 0.3827
Epoch 3/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 22s 1s/step - accuracy: 0.9151 - loss: 0.2510 - val_accuracy: 0.9362 - val_loss: 0.2926
Epoch 4/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 21s 1s/step - accuracy: 0.9159 - loss: 0.1994 - val_accuracy: 0.8936 - val_loss: 0.3067
Epoch 5/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 22s 1s/step - accuracy: 0.9105 - loss: 0.2692 - val_accuracy: 0.8511 - val_loss: 0.5056
Epoch 6/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 22s 1s/step - accuracy: 0.9230 - loss: 0.1970 - val_accuracy: 0.8723 - val_loss: 0.3514
Epoch 7/12
18/18 ━━━━━━━━━━━━━━━━━━━━ 20s 1s/step - accuracy: 0.9317 - loss: 0.1557 - val_accuracy: 0.8936 - val_loss: 0.3389
✅ Saved → C:\Users\kishore2004\pollen\Flask\model_boost_v2.keras
```

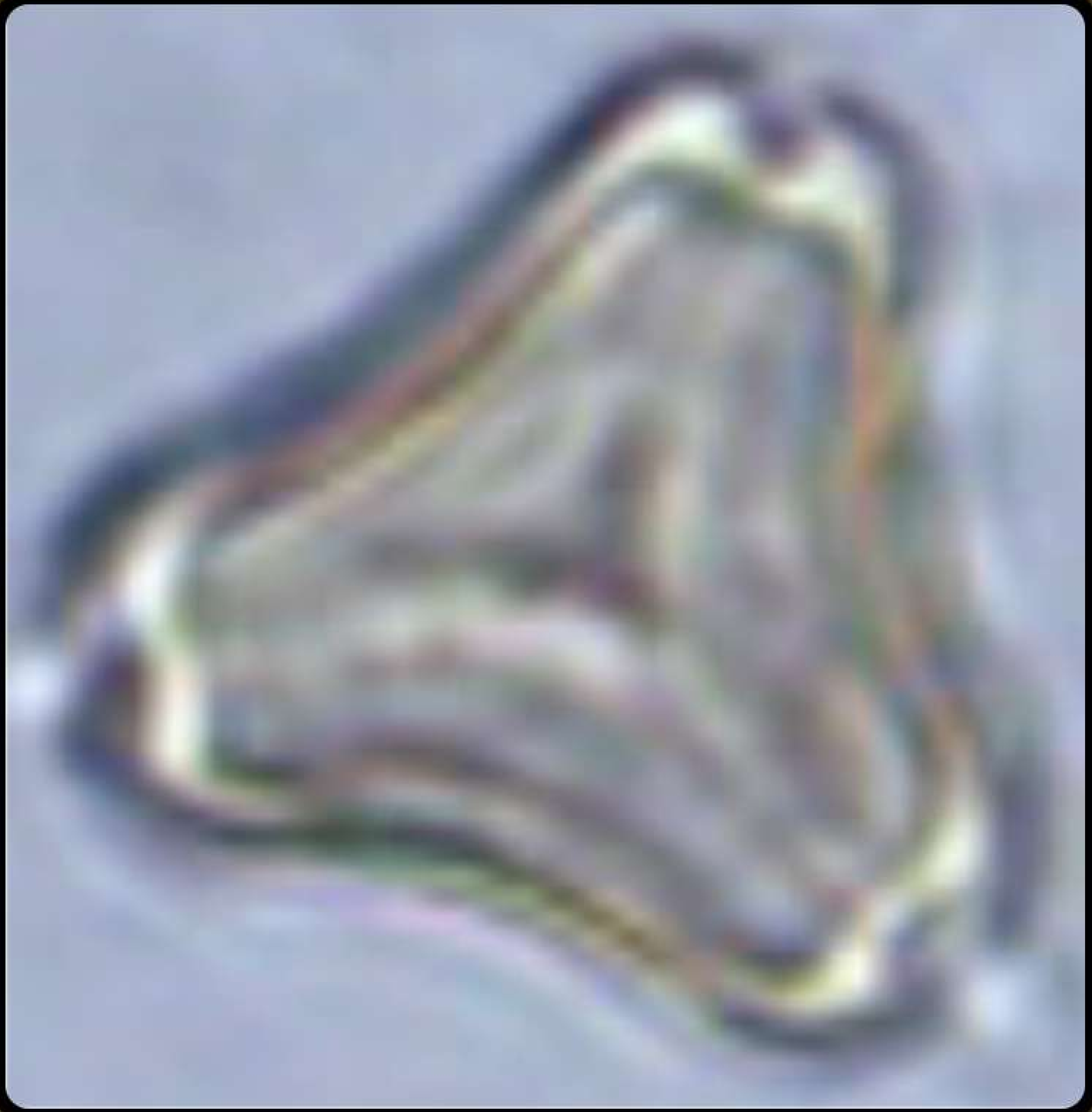


```
CLASS_NAMES = [
    "anadenanthera","arecaceae","cecropia","chromolaena","combretum",
    "croton","dipteryx","eucalipto","faramea","hyptis",
    "mabea","matayba","mimosa","myrcia","protium",
    "qualea","schinus","senegalia","serjania","syagrus","tridax"
]
idx2folder = {i: lbl for i, lbl in enumerate(CLASS_NAMES)}

# ---------------- Friendly display mapping ----------------------------
folder2nice = {
    "anadenanthera": "Anadenanthera colubrina",
    "arecaceae"    : "Arecaceae (Palm pollen)",
    "cecropia"     : "Cecropia spp.",
    "chromolaena"  : "Chromolaena odorata",
    "combretum"    : "Combretum leprosum",
    "croton"       : "Croton floribundus",
    "dipteryx"     : "Dipteryx alata",
    "eucalipto"    : "Eucalyptus spp.",
    "faramea"      : "Faramea occidentalis",
    "hyptis"       : "Hyptis suaveolens",
    "mabea"        : "Mabea fistulifera",
    "matayba"      : "Matayba guianensis",
    "mimosa"       : "Mimosa caesalpiniaefolia",
    "myrcia"       : "Myrcia multiflora",
    "protium"      : "Protium heptaphyllum",
    "qualea"       : "Qualea grandiflora",
    "schinus"      : "Schinus terebinthifolia",
    "senegalia"    : "Senegalia polyphylla",
    "serjania"     : "Serjania erecta",
    "syagrus"      : "Syagrus romanzoffiana",
    "tridax"       : "Tridax procumbens"
```



Pollen Grain Analyzer

Upload a microscopic image to identify the pollen species.



Pollen Grain Analysis

Microscopic analysis complete. Results are as follows.

IDENTIFIED SPECIES

eucalipto

CONFIDENCE SCORE

97.1%

Analyze Another

Thanks

Special thanks to SmartInternz and SmartBridge for providing this valuable opportunity and support to work on a real-time AI project and enhance our skills through practical experience.