# pollen\_pipeline.py

import torch

import torch.nn as nn

from torchvision import transforms, datasets, models

import numpy as np

from PIL import Image

import argparse

# ==== 1. Classification: RCANet-inspired (based on ResNet18) ====

class VDABlock(nn.Module):

def \_\_init\_\_(self, in\_channels):

super().\_\_init\_\_()

self.ventral = nn.Sequential(

nn.Conv2d(in\_channels, in\_channels, kernel\_size=1),

nn.Sigmoid()

)

self.dorsal = nn.Sequential(

nn.Conv2d(in\_channels, in\_channels, kernel\_size=1),

nn.Sigmoid()

)

def forward(self, x):

return x \* self.ventral(x) \* self.dorsal(x)

class RCANet(nn.Module):

def \_\_init\_\_(self, num\_classes=23):

super().\_\_init\_\_()

self.backbone = models.resnet18(pretrained=True)

self.backbone.fc = nn.Identity()

self.vdab = VDABlock(512)

self.classifier = nn.Linear(512, num\_classes)

def forward(self, x):

features = self.backbone(x)

attended = self.vdab(features.unsqueeze(-1).unsqueeze(-1)).squeeze(-1).squeeze(-1)

return self.classifier(attended)

def train\_classifier(data\_dir, epochs=10, lr=1e-4, batch\_size=32):

transform = transforms.Compose([

transforms.Resize((224,224)),

transforms.RandomHorizontalFlip(),

transforms.ToTensor()

])

dataset = datasets.ImageFolder(data\_dir, transform=transform)

loader = torch.utils.data.DataLoader(dataset, batch\_size=batch\_size, shuffle=True)

model = RCANet(num\_classes=len(dataset.classes)).to('cuda')

opt = torch.optim.Adam(model.parameters(), lr=lr)

criterion = nn.CrossEntropyLoss()

for epoch in range(epochs):

for imgs, labels in loader:

imgs, labels = imgs.to('cuda'), labels.to('cuda')

logits = model(imgs)

loss = criterion(logits, labels)

opt.zero\_grad(); loss.backward(); opt.step()

print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")

torch.save(model.state\_dict(), 'rcanet.pth')

# ==== 2. Synthetic Image Generation: StyleGAN2-ADA (PyTorch) ====

def generate\_synthetic(latent\_dim=512, n\_samples=16, ckpt\_path='stylegan2\_pollen.pt'):

from stylegan2\_ada\_pytorch import Generator

G = Generator(size=256, style\_dim=latent\_dim).cuda()

ckpt = torch.load(ckpt\_path)

G.load\_state\_dict(ckpt['g\_ema'], strict=False)

z = torch.randn(n\_samples, latent\_dim).cuda()

imgs = G(z, None)

imgs = (imgs.clamp(-1,1) + 1) \* 127.5

for i, img in enumerate(imgs):

pil = transforms.ToPILImage()(img.cpu().to(torch.uint8))

pil.save(f"synthetic\_pollen\_{i}.png")

print(f"Saved {n\_samples} synthetic pollen images.")

# ==== 3. Main CLI ====

if \_\_name\_\_ == "\_\_main\_\_":

parser = argparse.ArgumentParser(description="Pollen classifier & generator")

sub = parser.add\_subparsers(dest='cmd')

tr = sub.add\_parser('train')

tr.add\_argument('--data', required=True)

gn = sub.add\_parser('gen')

gn.add\_argument('--ckpt', required=True)

args = parser.parse\_args()

if args.cmd == 'train':

train\_classifier(args.data)

elif args.cmd == 'gen':

generate\_synthetic(ckpt\_path=args.ckpt)

else:

parser.print\_help()