import torch

from torch.utils.data import DataLoader, Dataset

from torchvision import transforms

from PIL import Image

import os

# Define transformations

transform = transforms.Compose([

transforms.Resize((64, 64)), # Resize images to 64x64

transforms.ToTensor(),

transforms.Normalize([0.5], [0.5]) # Normalize to [-1, 1] range for GANs

])

# Custom Dataset to load images from a directory

class SimpleImageDataset(Dataset):

def \_init\_(self, image\_folder, transform=None):

self.image\_folder = image\_folder

self.transform = transform

self.image\_paths = [os.path.join(image\_folder, img) for img in os.listdir(image\_folder) if img.endswith(('.png', '.jpg', '.jpeg'))]

def \_len\_(self):

return len(self.image\_paths)

def \_getitem\_(self, idx):

img\_path = self.image\_paths[idx]

image = Image.open(img\_path).convert('RGB')

if self.transform:

image = self.transform(image)

return image, 0 # We return 0 as a dummy label for compatibility

# Upload images to Colab

from google.colab import files

uploaded = files.upload()

# Create a directory for the images

os.makedirs('uploaded\_images', exist\_ok=True)

# Save uploaded files to directory

for filename in uploaded.keys():

with open(f'uploaded\_images/{filename}', 'wb') as f:

f.write(uploaded[filename])

# Initialize the dataset and dataloader

dataset = SimpleImageDataset(image\_folder='uploaded\_images', transform=transform)

dataloader = DataLoader(dataset, batch\_size=64, shuffle=True)

# Example: Display the number of images

print(f"Total images loaded: {len(dataset)}")

# Check one batch

for batch in dataloader:

images, \_ = batch

print(f"Batch size: {images.shape}")

break # Remove this break if you want to loop through all batches