import torch

from torchvision import models, transforms

from torchvision.ops import box\_iou

from PIL import Image

import matplotlib.pyplot as plt

from matplotlib.patches import Rectangle

from google.colab import files

# Load a pre-trained Faster R-CNN model

model = models.detection.fasterrcnn\_resnet50\_fpn(weights="DEFAULT")

model.eval()

# Function to preprocess the image

def preprocess\_image(image\_path):

image = Image.open(image\_path).convert("RGB")

transform = transforms.Compose([

transforms.ToTensor() # Convert the image to a PyTorch tensor

])

return transform(image).unsqueeze(0), image

# Function to display image with bounding boxes

def display\_image\_with\_boxes(image, predictions, threshold=0.5):

plt.figure(figsize=(10, 10))

plt.imshow(image)

ax = plt.gca()

for i, box in enumerate(predictions['boxes']):

score = predictions['scores'][i].item()

if score > threshold:

x1, y1, x2, y2 = box.tolist()

width, height = x2 - x1, y2 - y1

ax.add\_patch(Rectangle((x1, y1), width, height, edgecolor='red', facecolor='none', linewidth=2))

ax.text(x1, y1, f"{score:.2f}", color='red', fontsize=12, backgroundcolor='white')

plt.axis('off')

plt.show()

# Function to calculate Precision

def calculate\_precision(pred\_boxes, gt\_boxes, iou\_threshold=0.5):

"""

Calculate Precision for a single image.

Args:

pred\_boxes (torch.Tensor): Predicted bounding boxes (N x 4).

gt\_boxes (torch.Tensor): Ground truth bounding boxes (M x 4).

iou\_threshold (float): IoU threshold to consider a match (default: 0.5).

Returns:

precision (float): Precision percentage.

"""

if len(pred\_boxes) == 0:

return 0.0

if len(gt\_boxes) == 0:

return 100.0 # No ground truth means perfect precision

# Compute IoU matrix

ious = box\_iou(pred\_boxes, gt\_boxes)

# Match predictions to ground truth based on IoU

matched\_gt = set()

true\_positives = 0

for pred\_idx in range(ious.size(0)):

max\_iou, gt\_idx = torch.max(ious[pred\_idx], dim=0)

if max\_iou >= iou\_threshold and gt\_idx.item() not in matched\_gt:

true\_positives += 1

matched\_gt.add(gt\_idx.item())

# Calculate Precision

precision = true\_positives / len(pred\_boxes)

return precision \* 100

# Upload the image to Colab

uploaded = files.upload()

image\_path = list(uploaded.keys())[0]

# Ground truth boxes for the input image (x1, y1, x2, y2)

# Replace these with actual ground truth values for your dataset

ground\_truth\_boxes = torch.tensor([[50, 50, 200, 200], [300, 300, 400, 400]])

# Preprocess the image

input\_tensor, original\_image = preprocess\_image(image\_path)

# Perform object detection

with torch.no\_grad():

outputs = model(input\_tensor)

# Filter predictions with a confidence threshold

confidence\_threshold = 0.5

predictions = {

"boxes": outputs[0]['boxes'][outputs[0]['scores'] > confidence\_threshold],

"scores": outputs[0]['scores'][outputs[0]['scores'] > confidence\_threshold]

}

# Calculate Precision

precision = calculate\_precision(

pred\_boxes=predictions['boxes'],

gt\_boxes=ground\_truth\_boxes,

iou\_threshold=0.5

)

# Display the image with bounding boxes

display\_image\_with\_boxes(original\_image, predictions, threshold=confidence\_threshold)

# Print the Precision percentage

print(f"Precision: {precision:.2f}%")