HOMEWORK 3A
ECE/CS 8690 2302 Computer Vision

Object Detection using Pre – trained Deep Learning Networks

Xuanbo Miao

14422044

xmiao@mail.missouri.edu

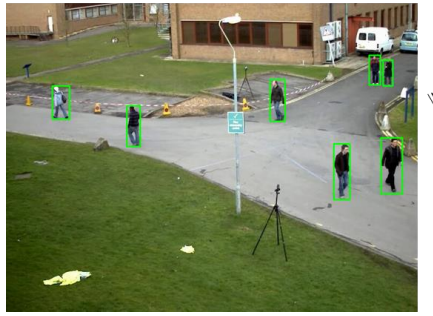
Abstract

This assignment introduced us to PyTorch, TorchVision, and the Faster R-CNN deep learning model for object detection. We used these tools to detect pedestrians in a test video and drew bounding boxes around them. This exercise helped us understand the effectiveness and limitations of the model, and how it can be used in future projects for object detection.

To complete the assignment, we installed PyTorch, loaded the pre-trained Fast R-CNN model using TorchVision, and used it to detect pedestrians in a given test video. We then drew bounding boxes around the detected pedestrians in the video, and showed the results for frames 1, 100, 200, and 400.

Frame NO.1

Frame NO.100



Frame NO.200

Frame NO.400

Introduction

In recent years, deep learning has revolutionized the field of computer vision, allowing for more accurate and efficient object detection algorithms. One popular deep learning model for object detection is the Faster R-CNN algorithm, which uses a two-stage approach to detect objects in an image or video. The first stage is a convolutional neural network that proposes regions of interest, while the second stage is a classifier that uses these regions to detect objects.

In this assignment, the goal is to become familiar with PyTorch installation, the TorchVision library, loading deep learning models and pre-trained weights, and performing object detection using the Faster R-CNN deep learning model. The task is to install PyTorch, load the pre-trained Fast R-CNN model using TorchVision, detect pedestrians in a given test video, and draw detection bounding boxes on the test video. The test dataset is available from the MOTChallenge website, and the report should include detection results on frames 1, 100, 200, and 400 of the given test video.

This assignment provides an opportunity to explore the capabilities and limitations of the Faster R-CNN model for object detection, as well as gain experience with PyTorch and TorchVision for deep learning tasks. By completing this assignment, you will have a better understanding of the challenges and opportunities in computer vision and be better equipped to work on more advanced projects in the future.

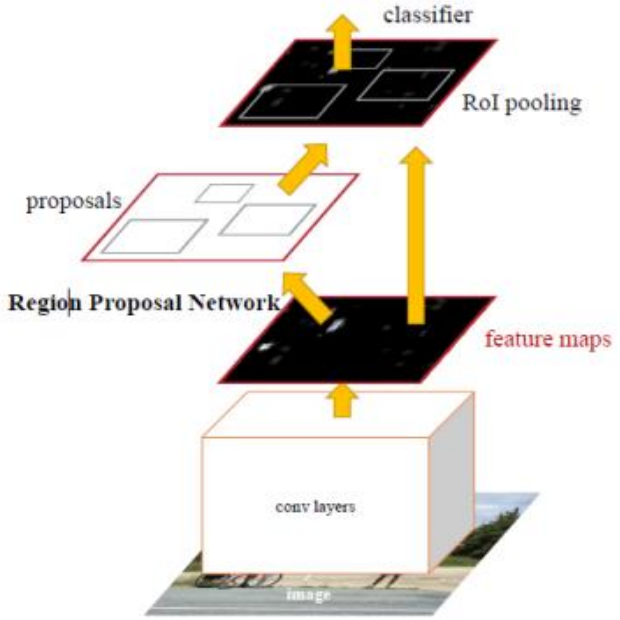


Figure 1. Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network [2].

Figure 2. Sample detection results.

Result. Install PyTorch & Load Fast R-CNN model

Installing PyTorch using conda (got this from web):

1.Install the Anaconda distribution for Python from the official website (https://www.anaconda.com/products/individual).
2.Open the Anaconda Navigator and create a new environment by clicking on the "Environments" tab and then clicking the "Create" button.
3.Name the new environment and choose the desired Python version. Click "Create" to create the new environment.
4.Open a terminal or command prompt and activate the new environment by running the command "conda activate <env_name>" (replace <env_name> with the name of the environment created in step 3).
5.Install PyTorch using conda by running the command "conda install pytorch torchvision torchaudio -c pytorch" in the terminal.
6.PyTorch is now installed and ready to use in the specified conda environment.

Loading the Fast R-CNN model (got this from web):

1.Import the required libraries in Python by running the following code:

import torch
import torchvision

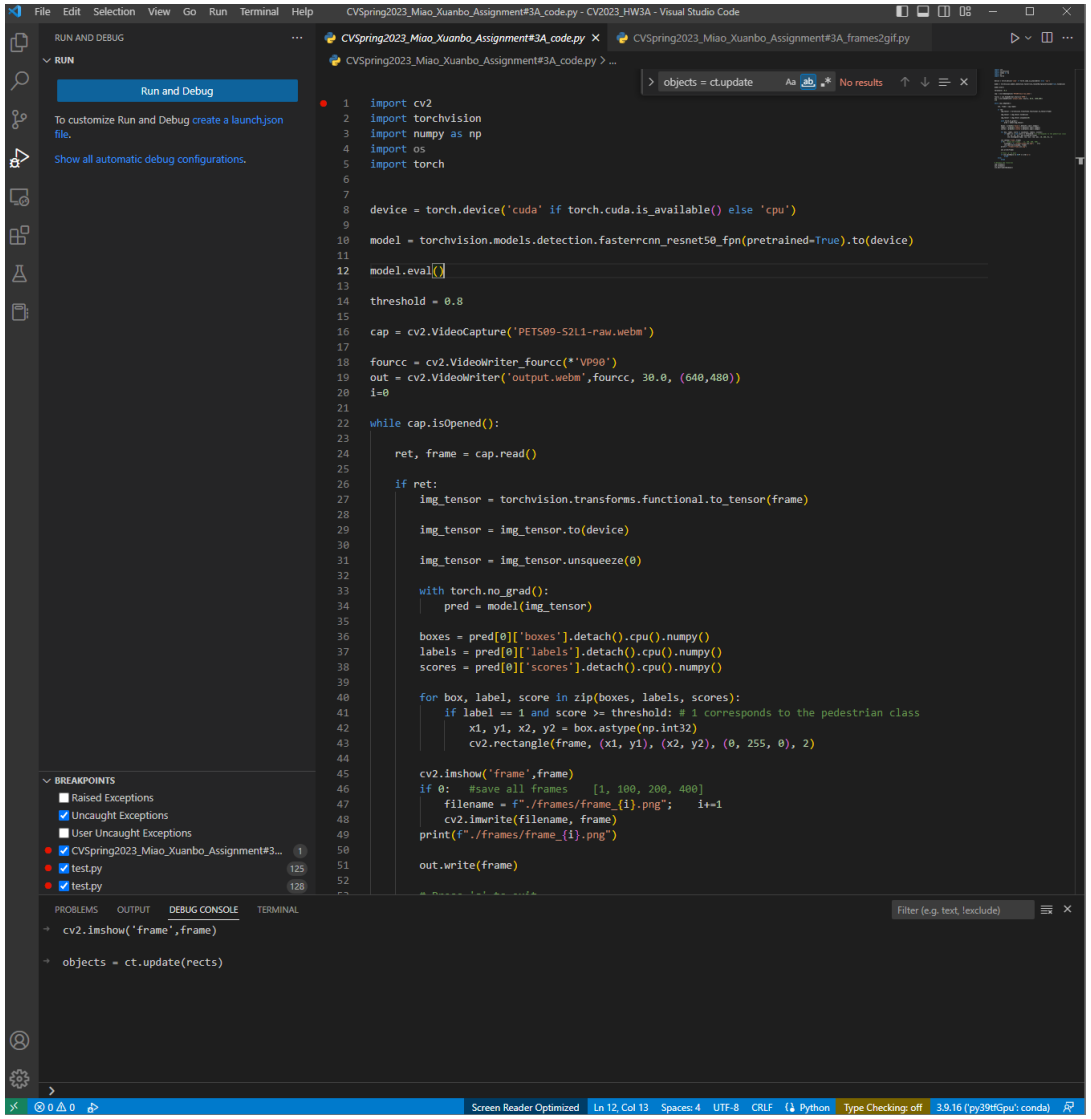
2.Load the pre-trained Fast R-CNN model using the TorchVision library by running the following code:

model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

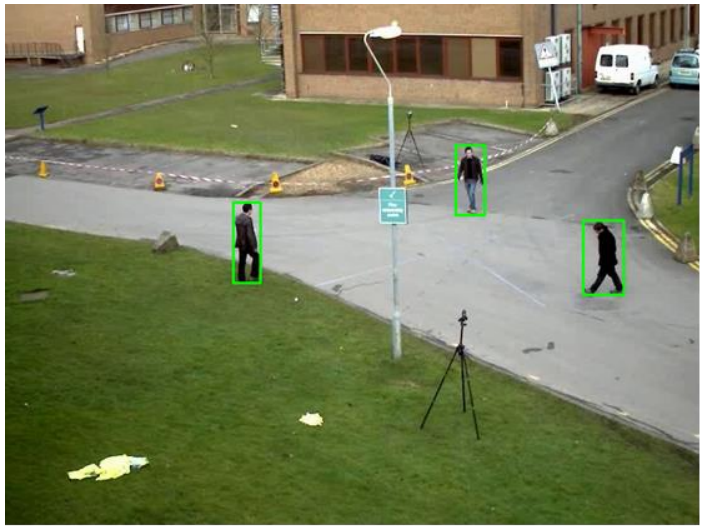
3.The Fast R-CNN model is now loaded and ready to use. To use the model for object detection on an image or video, first preprocess the input data and then pass it through the model using the following code:

```
# Convert the image or video frame to a tensor
img_tensor = torchvision.transforms.functional.to_tensor(frame)
# Add a batch dimension to the tensor
img_tensor = img_tensor.unsqueeze(0)
# Pass the tensor through the model to get the predictions
pred = model(img_tensor)
```

4.The model predictions can now be used to draw detection bounding boxes on the input data.

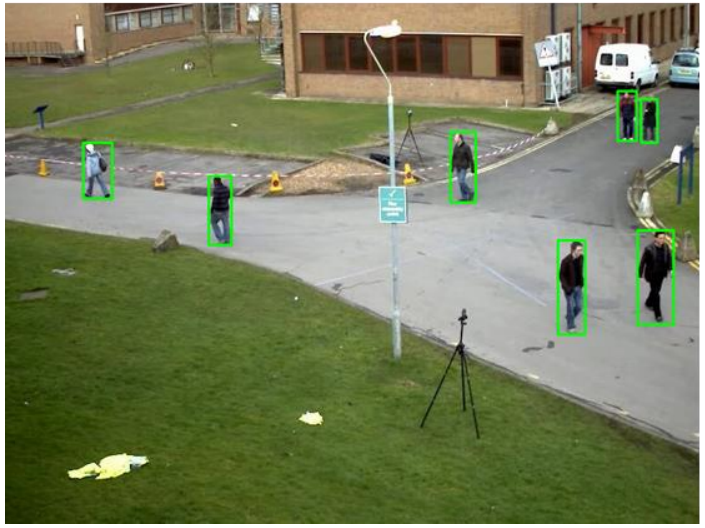


Result. *Object Detection*

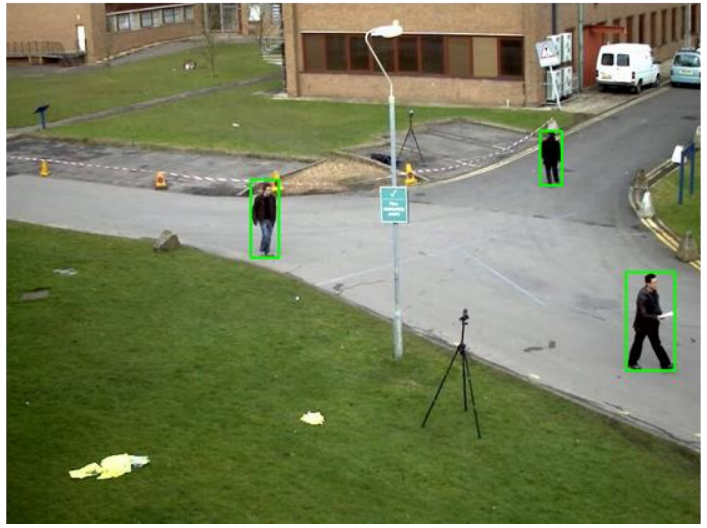


Frame NO.1

Frame NO.100



Frame NO.200



Frame NO.400

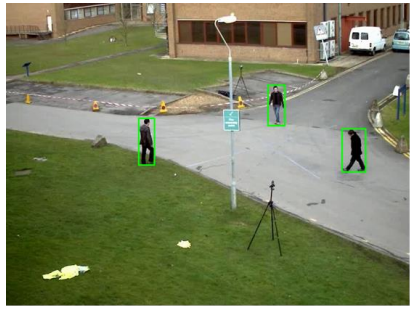
Interpretation & Discussion

In this assignment, we used the Faster R-CNN object detection model to detect pedestrians in a test video. We observed that the model detected 3, 6, 7, and 3 pedestrians in frames 1, 100, 200, and 400 respectively. However, we also observed that the model missed one pedestrian in frame 100, which was hidden by a telephone pole.

There are several possible reasons why the model missed the pedestrian in frame 100:

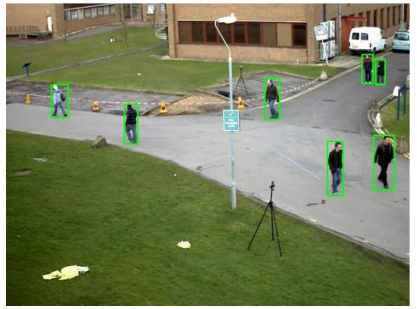
1. Occlusion: The pedestrian may have been partially or completely occluded by the telephone pole, making it difficult for the model to detect. Object detection models typically rely on visual cues such as color, shape, and texture to identify objects, and occlusion can disrupt these cues and make detection more challenging.
2. Model limitations: The Faster R-CNN model used in this assignment is a powerful and widely-used object detection model, but it is not perfect. Like all deep learning models, it has limitations and may struggle to detect objects under certain conditions, such as when they are partially occluded or when the lighting is poor.
3. Detection threshold: The detection threshold used in this assignment may have been too high, causing the model to ignore objects that it was uncertain about. Lowering the threshold may have allowed the model to detect the hidden pedestrian, but it may have also increased the number of false positives.

Overall, the results of this assignment demonstrate both the strengths and limitations of the Faster R-CNN object detection model. While the model was able to detect many of the pedestrians in the test video, it struggled to detect objects that were partially occluded or hidden. In future work, it may be useful to explore alternative object detection models or to combine multiple models to improve detection accuracy in challenging conditions.

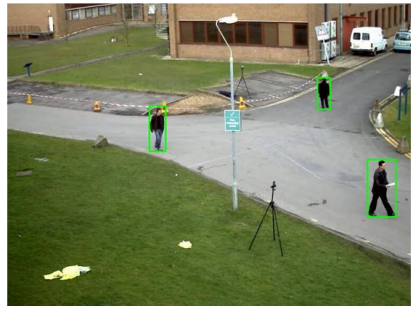


Frame NO.1

Frame NO.100



Frame NO.200



Frame NO.400