**experience and learnings**

I started working on the object detection project from the morning of May 8th. I followed a structured approach using Command Prompt to create the folder structure. I created an `Object-detection-project` directory, inside which I placed a `data` folder and a `src` folder. In the `data` folder, I uploaded the `COCO128.v@-640x640.coco` dataset that I downloaded from Roboflow. In the `src` folder, I created Python files such as `dataset.py`, `eval.py`, `model.py`, `test.py`, and `train.py`. I also created a `requirements.txt` file to manage the necessary dependencies.  
  
I continued working in a structured manner on VS Code until the evening of May 12th, but I constantly faced import errors. I used ChatGPT for assistance, but it repeatedly suggested similar solutions that didn’t resolve the issue. Eventually, I discovered that my virtual environment had compatibility issues with `torch` and `torchvision`. I then created a new virtual environment called `venv10`, downgraded my Python version to 3.10.4 (to ensure `torch` compatibility), and switched to using Jupyter Notebook to avoid further issues.  
  
I showcased my code to ChatGPT and presented my approach, but it pointed out several critical corrections. Here's how the collaboration unfolded:

**Problem Statement**

The project was structured well, but the code ran into several runtime and compatibility issues. Some of the main problems involved:  
- Improper device transfer for tensors inside the targets dictionary.  
- Incorrect bounding box format (expected [x\_min, y\_min, x\_max, y\_max] but given [x, y, width, height]).  
- Wrong tensor manipulation within the training loop.

**My Initial Attempt (Failed Version)**

model = fasterrcnn\_resnet50\_fpn(pretrained=True)  
model.roi\_heads.box\_predictor = FastRCNNPredictor(model.roi\_heads.box\_predictor.cls\_score.in\_features, 91)  
  
for images, targets in train\_loader:  
 images = [img.to(device) for img in images]  
 targets = [t.to(device) for t in targets]  
 loss = model(images, targets)

Issue: `.to(device)` cannot be called on dictionaries or lists directly.

**ChatGPT's Correction**

targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

This correction fixed the device transfer issue.

**Dataset Class – Bounding Box Format Fix**

Initial (incorrect):  
boxes.append([bbox[0], bbox[1], bbox[2], bbox[3]])  
  
Corrected:  
boxes.append([  
 float(bbox[0]),  
 float(bbox[1]),  
 float(bbox[0] + bbox[2]),  
 float(bbox[1] + bbox[3])  
])

**Training Loop Fix**

Initial:  
images = images.to(device)  
targets = targets.to(device)  
  
Corrected:  
images = [img.to(device) for img in images]  
targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

**Final Working Code**

The final working code integrated all the fixes identified through this debugging journey. After approximately 4 hours of training, the model performed successfully.  
  
from torchvision.models.detection.faster\_rcnn import FastRCNNPredictor  
from torchvision.models.detection import fasterrcnn\_resnet50\_fpn  
  
num\_classes = 91  
model = fasterrcnn\_resnet50\_fpn(pretrained=True)  
in\_features = model.roi\_heads.box\_predictor.cls\_score.in\_features  
model.roi\_heads.box\_predictor = FastRCNNPredictor(in\_features, num\_classes)  
model.to(device)  
  
optimizer = torch.optim.SGD(model.parameters(), lr=0.005, momentum=0.9, weight\_decay=0.0005)  
  
model.train()  
for epoch in range(10):  
 for images, targets in train\_data\_loader:  
 images = [img.to(device) for img in images]  
 targets = [{k: v.to(device) for k, v in t.items()} for t in targets]  
 optimizer.zero\_grad()  
 loss\_dict = model(images, targets)  
 losses = sum(loss for loss in loss\_dict.values())  
 losses.backward()  
 optimizer.step()  
 print(f"Epoch [{epoch+1}/10], Loss: {losses.item():.4f}")

**Result**

After training for approximately 4 hours using the COCO128 dataset (640x640 resolution), the model was successfully evaluated and returned valid predictions.  
This experience taught me how to troubleshoot effectively and collaborate with AI tools like ChatGPT to iteratively reach a functional, optimized solution.

# **What Went Wrong and What Went Right**

Although the assignment was due before 12 AM last night, I chose to focus on getting everything right instead of rushing to submit a half-baked solution. One of the key things that went wrong initially was the time spent troubleshooting repeated import errors and resolving virtual environment issues. My first virtual environment kept throwing errors related to `torchvision` and `torch`, which was frustrating and significantly slowed down my progress.  
  
However, what went right was the structured way I approached the solution. From setting up the folder structure to writing modular Python files and switching to Jupyter Notebook with a stable Python 3.10.4 environment, each step brought me closer to a working prototype. Additionally, leveraging ChatGPT's assistance helped me debug the final issues and reach a fully functional object detection pipeline.  
  
Despite the delay, I believe taking a few extra hours to ensure quality and correctness was the right decision. Perfection and understanding matter more to me than just meeting a deadline.