Experience Report – Object Detection Model (Car Detection using Faster R-CNN with ResNet50 + FPN)

This assignment was a very interesting experience for me. Building an end-to-end object (car) detection model, deploying it with FastAPI, and integrating functionalities like training, prediction, and metric evaluation taught me far more than I expected when I began.

In my previous object detection projects, I had always worked with YOLO. This was the first time I worked with Faster R-CNN + ResNet50 + FPN.

Challenges faced During Implementation

The biggest challenge I faced was understanding and integrating the components of Faster R-CNN + ResNet50 + FPN into a single pipeline. Also, in my VS Code setup, I tried to make the code as modular as possible, which introduced additional complexity.

I read a lot about the model architecture and its documentation. It was time-consuming but also an enjoyable learning process. Integrating the model with FastAPI was another interesting and rewarding task.

How I used AI tools to help with Coding

I used ChatGPT extensively throughout this project. Whenever I got stuck while understanding components of PyTorch's object detection APIs, FastAPI routing, evaluation metrics like mAP, or even formatting data correctly, ChatGPT was my go-to support.

Instead of blindly copying code, I always tried to understand the logic first. ChatGPT helped me break down complex concepts into simpler explanations that made things easier to grasp.

Since I was new to this type of object detection technique, ChatGPT helped me learn everything and saved a lot of time that would have gone into reading lengthy documentation.

Additionally, I am working on deploying my model as a Docker container on an Amazon EC2 instance.

What Surprised Me:

I was surprised by how smooth the development process became once I started using AI tools effectively. I was also amazed to discover how many different object detection models exist. Learning object detection in such depth was truly exciting and eye-opening.

Balance Between Coding Myself vs. Using AI:

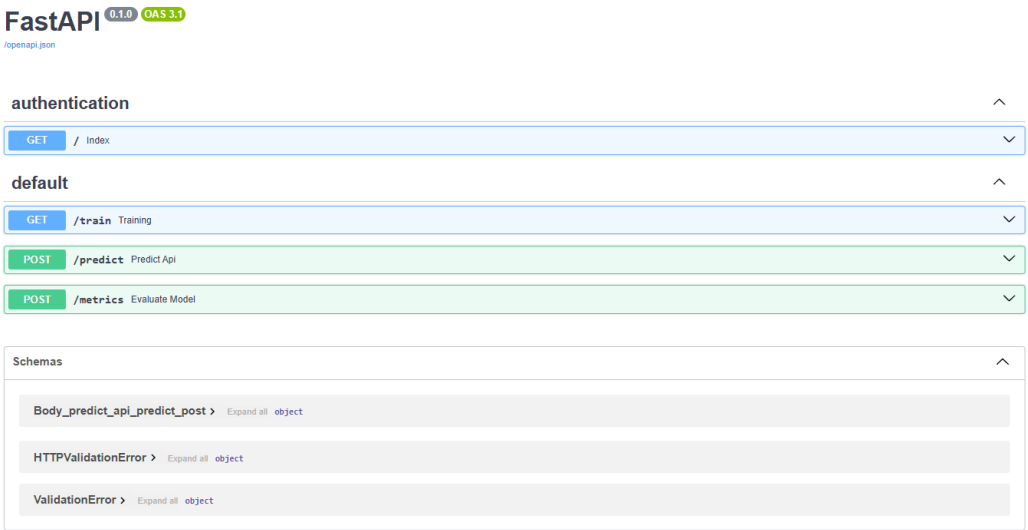
This was a critical insight for me. While AI helped reduce the time I spent searching documentation or browsing Stack Overflow, I made sure not to become completely dependent on it. I always reviewed the suggestions, tried to understand the logic, and made modifications as needed.

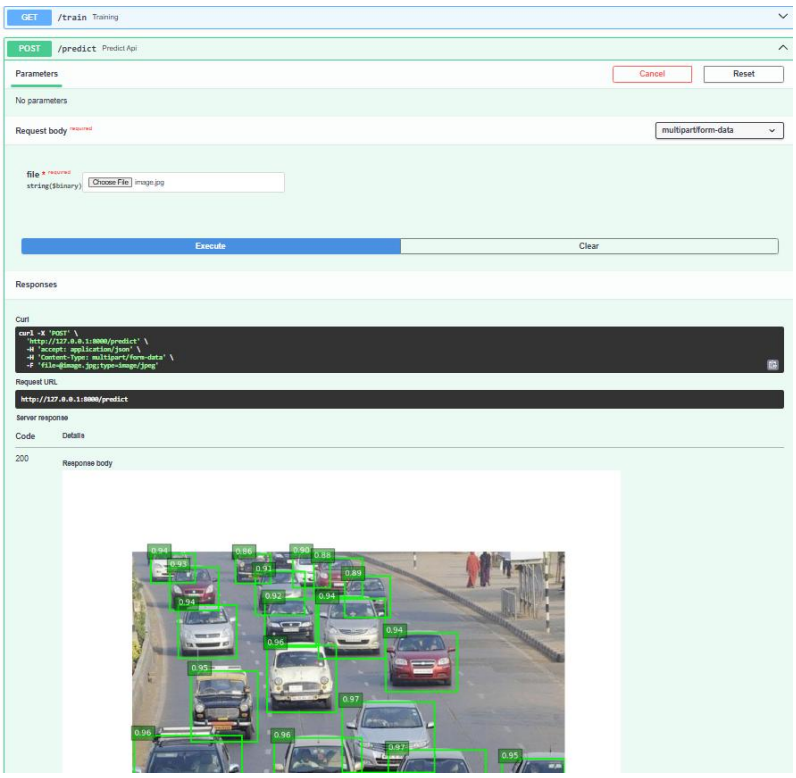
I feel that the right balance is to use AI as a mentor, not as a crutch. It boosts productivity but shouldn't replace real learning.

Suggestions for Improving the Assignment:

- It would be great if the assignment explicitly encouraged modular coding practices and deployment.
- Encouraging the use of CI/CD or containerization could elevate the learning experience further.
- Also, we can also encourage the participant to Dockerize it and used it in the in the aws ec2 instance

Snapshots of my project





POST /metrics Evaluate Model

Parameters

Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/metrics' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://127.0.0.1:8000/metrics
```

Server response

Code	Details
200	Response body


```
{
  "Metric": {
    "0": "map",
    "1": "map_50",
    "2": "map_75",
    "3": "map_small",
    "4": "map_medium",
    "5": "map_large",
    "6": "mar_1",
    "7": "mar_10",
    "8": "mar_100",
    "9": "mar_small",
    "10": "mar_medium",
    "11": "mar_large",
    "12": "map_per_class",
    "13": "mar_100_per_class",
    "14": "classes"
  },
  "Value": {
    "0": "tensor(0.3911)",
    "1": "tensor(0.5902)",
    "2": "tensor(0.4331)",
    "3": "tensor(0.2247)",
    "4": "tensor(0.3817)",
    "5": "tensor(0.5213)",
    "6": "tensor(0.2776)",
    "7": "tensor(0.5372)",
```


Download

Response headers

```
content-length: 586
content-type: application/json
date: Thu,15 May 2025 11:04:17 GMT
server: uvicorn
```