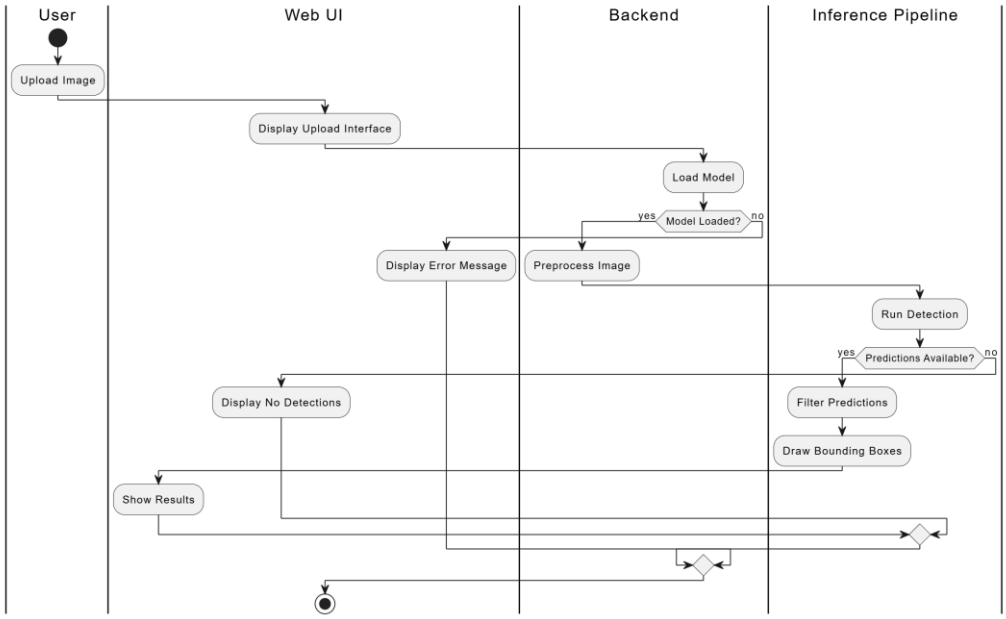
Report

Experience Report

Finishing up this object detection project with Faster R-CNN with a backbone of MobileNetV2 was a beneficial and learning experience. This is how the process unfolded, problems that I had faced, AI tool utilization, and things learned through it.

Work-Flow Diagram



Challenges Faced

One of the major issues I encountered was merging the custom MobileNetV2 backbone into the Faster R-CNN framework. While PyTorch's API is thoroughly documented, performing the changes while maintaining compatibility with all parts (such as anchor generators, ROI poolers, etc.) was problematic. Moreover, handling varying image input sizes and visualization with PIL (particularly with newer vs. older Pillow versions) presented small but frustrating obstacles.

Loading the model and handling pretrained weights also involved trial and error, particularly handling model state dictionaries mismatching due to layer or architecture version changes.

AI Tool Usage

- I relied extensively on ChatGPT throughout this assignment, especially for:
- Prompting AI to create starter code for developing a MobileNetV2-based Faster R-CNN architecture.
- Debugging mismatches between model weight files and architecture with the help of error messages.
- Enhancing visualization by requesting a clean method of drawing bounding boxes and backgrounded label text using PIL.
- Clarifying the internal modules of FasterRCNN and why the ROI Pooler and Anchor Generator are used.

Report

- I made sure to read and comprehend the AI-written code. In a few instances, I adapted it to meet my particular requirements (e.g., utilizing a fallback pre-trained model in case loading had failed).

What I Learned

I learned more about how to utilize custom backbones in object detection models and how detection pipelines are executed end to end. Previously, I had only utilized pre-trained models (such as YOLO or typical Faster R-CNN), but now I am more confident when making adjustments to architectures. I learned how to properly handle inference, post-processing, and visualization as well.

What Surprised Me

I was impressed with how modular and flexible the torchvision detection API is. Though it is complicated behind the scenes, it still provides a lot of room for customization. Also, I did not anticipate AI software like ChatGPT to be so useful in dissecting difficult-to-understand error messages and promptly providing working alternatives.

AI vs. Manual Coding Balance

I believe that using AI accelerated the boilerplate configuration and debugging, but I still had to reason and know what I was doing in order to get the code to function end-to-end. The equilibrium felt sound—I was not merely copying and pasting code. Indeed, I frequently asked ChatGPT the reason why we needed to make these changes and utilized it as an interactive learning aid instead of a substitute for my own coding.

Suggestions for Improving the Assignment

- It would be useful to offer a reference model checkpoint or data to test against, particularly since VOC has licensing and download complications.
- Adding in optional extensions (such as training, mAP metric evaluation on, or applying a different backbone) could serve to challenge stronger students.
- More definition on the error handling, output visualization, or model comparison requirement would impose more structure on the learning objectives.
- Overall, this task provided me with an actual world experience of creating and testing customized object detection pipelines, and taught me how to utilize AI tools properly—not simply depend on them passively.