Experience Report: Object Detection Model Building Assignment

AI Intern Assignment

Submitted by: Meka Durga Sai Vardhan Reddy

Submitted on May 12, 2025

1 Introduction

As part of the AI Intern assignment, I built an object detection model using a pre-trained CNN backbone, added detection layers, trained it on the COCO dataset, and evaluated its performance. This project was both challenging and rewarding, offering hands-on experience with computer vision and the opportunity to collaborate with AI tools like Grok (built by xAI). Below, Ill share my genuine thoughts on the challenges I faced, how I used AI assistance, what I learned, surprises I encountered, my feelings about AI collaboration, and suggestions for improving this assignment.

2 Challenges Faced During Implementation

The biggest challenge I faced was the training time. I used an NVIDIA GeForce RTX 3050 Laptop GPU with 4GB VRAM, which limited the batch size I could use initially. The COCO dataset, with 118,287 training images, resulted in an overwhelming number of batches per epoch59,144 with a batch size of 2. Early attempts took 10-20 hours for just 2 epochs, which was frustrating because I couldnt iterate quickly to test different configurations. I also ran into issues with corrupted images in the dataset (e.g., 000000461506.jpg), which caused the training to crash until I modified the code to skip such files.

Another challenge was memory management. My GPUs limited VRAM meant I had to carefully balance batch size, image resolution, and model complexity. Initially, I used EfficientNet-B3 as the backbone, but it consumed too much memory, forcing me to reduce the batch size and slow down training. Debugging these memory issues and optimizing the pipeline felt overwhelming at times, especially since I wasnt familiar with techniques like gradient accumulation or mixed precision training at the start.

3 How I Used AI Tools to Help with Coding

I heavily relied on Grok to assist with coding, debugging, and optimization. Heres how I used it:

- **Initial Setup**: I asked Grok to help me select a backbone and detection framework. I chose EfficientNet with a Feature Pyramid Network (FPN) because Grok explained that FPN is effective for multi-scale object detection, which is crucial for a dataset like COCO with objects of varying sizes.
- **Code Generation**: I prompted Grok to generate the initial train.py, coco_dataset.py, and fpn_efficientnet.py files. For example, I asked, "Generate a training script for object detection using EfficientNet and FPN on the COCO dataset." Grok provided a solid starting point, including data loading, model setup, and a training loop with mixed precision training.
- **Debugging**: When I encountered the corrupted image error, I shared the error message with Grok and asked, "How do I handle corrupted files in my dataset?" Grok suggested modifying coco_dataset.py to skip corrupted images, which worked perfectly after a few iterations.
- **Optimization**: The training was too slow, so I repeatedly asked Grok to optimize the code. For instance, I said, "The process of epochs is getting slow, make it run fast," and later, "Make the train file use the GPU to full capacity and run fast." Grok introduced techniques like mixed precision training, gradient accumulation, torch.compile, and image resizing (from 300x500 to 224x224). It also suggested switching to a lighter backbone (EfficientNet-B0) and reducing the dataset size for testing.
- **Understanding Code**: For each piece of code Grok generated, I asked follow-up questions like, "What does torch.compile do?" or "How does gradient accumulation work?" This helped me understand the concepts behind the code rather than just copying it blindly.

Overall, Grok was a lifesaver for generating code snippets, explaining concepts, and suggesting optimizations. I documented that most of the code in train.py, coco_dataset.py, and fpn_efficientnet.py was AI-assisted, but I made sure to understand each component by asking Grok for explanations.

4 What I Learned from the Project

This project taught me a lot about object detection and deep learning workflows. I learned how to:

- **Build an Object Detection Model**: I gained hands-on experience with integrating a pre-trained backbone (EfficientNet) with an FPN and Faster R-CNN framework. I now understand how FPN helps detect objects at different scales by creating a feature pyramid.
- **Optimize Training**: I learned techniques like mixed precision training (using torch.amp), gradient accumulation, and torch.compile to speed up training and manage GPU memory. I also saw the impact of image resizing and using a lighter backbone on training speed.
- **Handle Real-World Issues**: Dealing with corrupted images in the dataset taught me the importance of robust data preprocessing. I also learned how to profile training performance (using Data Time and Compute Time) to identify bottlenecks.
- **Collaborate with AI**: I developed a workflow for working with AI toolsstarting with a high-level prompt, refining the code through iterative requests, and asking for explanations to deepen my understanding.

5 What Surprised Me About the Process

I was surprised by how much GPU memory management affected the training process. I initially thought my 4GB VRAM would be sufficient, but I quickly ran into out-of-memory errors with larger batch sizes. It was eye-opening to see how techniques like gradient accumulation and image resizing could make such a big difference.

I was also surprised by how effective AI assistance was for debugging. When I encountered the corrupted image error, I expected it to take hours to fix, but Grok suggested a solution within minutes. This made me realize how powerful AI tools can be for accelerating development, especially for someone like me whos still learning.

6 How I Feel About the Balance Between Writing Code Myself vs. Using AI Assistance

I found the balance between writing code myself and using AI assistance to be a bit tricky. On one hand, Grok saved me a ton of time by generating complex code (like the training loop and dataset handling) that wouldve taken me days to write from scratch.

It also introduced me to advanced techniques I wouldnt have known about otherwise, like mixed precision training and torch.compile. On the other hand, I sometimes felt like I was relying too heavily on Grok, especially early on when I didnt fully understand the generated code. I made a conscious effort to ask for explanations and modify the code myself (e.g., adjusting batch sizes or image resolutions), which helped me feel more ownership over the project.

Overall, I think AI assistance is incredibly valuable for learning and prototyping, but I need to balance it with more manual coding to build my confidence. For this project, Id say 80% of the code was AI-generated or AI-assisted, but I spent a lot of time understanding and tweaking it, which made me feel like I was still contributing meaningfully.

7 Suggestions for Improving This Assignment

Here are a few suggestions to improve the assignment:

1. **Provide More Guidance on Hardware Constraints**: It wouldve been helpful to have guidance on how to handle limited GPU memory, like a recommended batch size or image resolution for different VRAM capacities. I spent a lot of time figuring this out through trial and error.

2. **Suggest a Smaller Dataset for Beginners**: The COCO dataset was overwhelming due to its size. A smaller dataset (or a subset of COCO) could be recommended for beginners to make initial training faster and less intimidating.

3. **Include a Section on Evaluation Metrics**: While the assignment asked for evaluation metrics like mAP, I wasnt sure how to compute them properly. Providing a simple evaluation script or pointing to a library (e.g., pycocotools for COCO evaluation) wouldve been helpful.

4. **Encourage More Manual Coding Early On**: While AI assistance was great, I think the assignment could encourage more manual coding for certain components (e.g., writing the data loader from scratch) to help interns build foundational skills before relying on AI.

8 Conclusion

This assignment was a challenging but rewarding experience. I learned a lot about object detection, training optimization, and collaborating with AI tools. While the long training times and GPU memory issues were frustrating, using Grok to debug and optimize the process taught me how to tackle real-world deep learning problems. Im excited to apply

these skills to future projects and continue finding the right balance between AI assistance and manual coding.