**AI Internship – Object Detection Assignment**

**Experience Report by Saurav Ghoshal**

#### 1. How the system works.(An Overview)

In this project, I implemented a **YOLO-style object detection model** using a **ResNet50 backbone** trained on the **Pascal VOC 2007 dataset**. The system works in three primary stages:

1. **Feature Extraction with CNN Backbone:**

I used a pre-trained ResNet50 (excluding its classification layers) to extract high-level spatial features from input images.

1. **YOLO-style Detection Head:**

I added a custom convolutional head that predicts bounding box coordinates, objectness scores, and class probabilities. Each grid cell in the final feature map is responsible for detecting objects within that region.

1. **Training and Evaluation:**

The model was trained on VOC images, and evaluation metrics like mAP, precision, and recall were used to assess performance. A custom collate\_fn was added to handle variable-size annotations from VOC.

#### 2. Challenges Faced During Implementation

One of the biggest challenges was understanding how to plug a detection head (YOLO-style) into a classification backbone like ResNet50. The architecture wasn’t built for object detection, so modifying it without breaking the model flow was tricky. Another major hurdle was dealing with the Pascal VOC dataset structure. The XML annotations weren’t straightforward to parse and convert into a YOLO-like format, especially when dealing with batched data.

Also, the Runtime Error: each element in list of batch should be of equal size error was frustrating because it stemmed from inconsistent image sizes and annotation structures, which aren’t things you notice right away. It taught me how finicky Data Loaders and collate functions can be in PyTorch.

#### 3. How I Used AI Tools to Help With Coding

AI tools like ChatGPT were invaluable. I used it for:

* Figuring out how to correctly modify the ResNet50 backbone.
* Writing the custom collate function to handle VOC’s XML labels.
* Understanding how YOLO heads work and how to structure multi-anchor output.
* Placeholder loss function design (and understanding why it’s not trivial).
* Quick code debugging — especially PyTorch dimension mismatches.

However, I didn’t copy-paste blindly. I always asked why certain things were structured a certain way and made sure I understood the core logic before proceeding.

#### 4. What I Learned From the Project

* The backbone architecture isn’t just about feature extraction — how you *connect* the detection head really matters.
* Object detection requires careful label handling — the pre-processing is as important as the model itself.
* Training object detection models from scratch (even with pre-trained backbones) needs thoughtful loss functions, normalization, and anchor logic.
* Building a minimal viable detection model isn’t as easy as using something prebuilt like YOLOv8 — and that’s okay.

#### 5. What Surprised Me About the Process

Honestly, I thought integrating a YOLO head would be quick. But the moment you go beyond classification, the complexity of bounding box handling, label parsing, and output shaping increases exponentially. The Pascal VOC dataset was also messier than expected. I had to manually deal with XML trees and label inconsistencies. The surprise wasn’t that things were difficult — it was how *many* different moving parts there were just to get a basic pipeline working.

#### 6. Balance Between Writing Code Myself vs. Using AI Assistance

This was a huge learning point. I found that writing every line manually wasn’t realistic under time pressure — and honestly, not the most effective way to learn. Using AI to guide my architecture, correct bugs, and suggest best practices let me focus more on understanding rather than blindly struggling. That said, I didn’t treat the AI like a “do it for me” tool — I treated it like a tutor I could ask 100 dumb questions without being judged.

The balance, for me, was:

* Write and structure core logic myself
* Use AI to generate building blocks or debug
* Never use AI suggestions without *understanding them first*

#### 7. Suggestions for Improving the Assignment

* Instead of asking us to choose both a backbone and a detection framework, I’d suggest narrowing it to one specific combo (like ResNet50 + YOLO head) and letting us focus more on implementation than decision paralysis.
* More starter code for label pre-processing or data collation would help — not to spoon-feed us, but to let us focus on learning the *model logic* rather than spending 60% of the time cleaning up VOC XML files.
* Including a very simple pre-defined loss function that works would’ve been helpful — object detection losses can be hard to implement from scratch if we haven’t done it before.

**Final Thoughts**

This assignment was **a rich, practical deep-dive into object detection**. It taught me how to move from theory to implementation, while also showing the evolving role AI can play in development. I feel more confident now in using CNNs for real-world computer vision tasks and even more confident in co-developing with AI.