**Object Detection Internship Report**

**Name: Shreevats Dhyani  
Project Title: Object Detection with Faster R-CNN on Pascal VOC  
Platform: Google Colab, PyTorch  
Model Architecture: Faster R-CNN with ResNet-50 FPN  
Dataset: Pascal VOC 2007 (trainval split)**

**🔧 Project Overview**

In this assignment, I developed a complete object detection pipeline using Faster R-CNN with a ResNet-50 FPN backbone, trained on the Pascal VOC 2007 dataset. The core objective was to understand the workings of a region-based object detector and implement training, evaluation, and inference workflows effectively. I submitted the Colab notebook after 2 training epochs due to time constraints but later experimented with 13 epochs, which yielded improved visual and detection accuracy, though I could not save that version in time.

**🛠️ Technical Stack & Methodology**

* Backbone Network: ResNet-50 with Feature Pyramid Networks (FPN), pretrained on ImageNet
* Detection Framework: Faster R-CNN via torchvision.models.detection
* Dataset: Pascal VOC 2007, using torchvision.datasets.VOCDetection
* Training Parameters:
  + Epochs: 2 (submitted), later extended to 13 for better accuracy
  + Optimizer: Stochastic Gradient Descent (SGD) with momentum
  + Loss Functions: RPN objectness loss, classification loss, and bounding box regression
* Evaluation: Qualitative evaluation through bounding box visualization and label overlays

**🧊 AI & Development Tools**

**ChatGPT**

* Helped with resolving shape mismatches and CUDA errors
* Explained how RoI heads and anchor generation work in Faster R-CNN
* Assisted in generating visualization utilities and fixing tensor plotting issues

**GitHub Copilot**

* Provided inline suggestions for loops, optimizer configs, and function signatures
* Helped automate repetitive operations (e.g., model eval mode, plotting loops)

**🧩 Challenges Faced**

| **Challenge** | **Resolution** |
| --- | --- |
| Tensor conversion issues | Used .detach().cpu().numpy() before plotting model predictions |
| False positives (e.g., “chair”) | Observed this was due to low training epochs and class imbalance |
| Plotting both GT & predictions | Added red/green colored boxes with labels for easier comparison |
| Colab memory and runtime limits | Submitted with 2 epochs but reran with 13 epochs to observe better results |

**📘 Learning Outcomes**

* How region proposal-based object detectors function end to end
* The role of FPN in multi-scale object localization
* The significance of training duration and class imbalance on detection accuracy
* How to visualize results for model debugging and performance analysis
* Practical insight into using pretrained networks with fine-tuning

**💡 Insights & Reflections**

* Model Accuracy Improves with Training: The difference between 2 and 13 epochs was immediately noticeable — detections became more confident and precise.
* Visualization is Key: Debugging through plots often revealed errors not apparent from logs or metrics.
* Backbones Matter: ResNet-50 FPN provided rich features even with limited epochs, indicating how much pretrained networks accelerate convergence.
* AI Tools Amplify Productivity: Tools like ChatGPT are most useful when paired with hands-on debugging and experimentation.

**⚖️ Human vs AI Contribution**

| **Component** | **Written by Me** | **AI Assisted** |
| --- | --- | --- |
| Dataset & DataLoader Setup | ✅ | ✖️ |
| Model Definition & Training Loop | ✅ | ✅ |
| Error Debugging (CUDA, shape) | ✅ | ✅ |
| Visualization Scripts | ✅ | ✅ |
| Architectural Insights | ✅ | ✅ (clarifications) |

**🔁 Suggestions for Future Versions of This Assignment**

* Include a pre-validation metric or mAP calculator to benchmark progress
* Offer anchor tuning presets to help beginners avoid empty prediction errors
* Add a bonus track for exporting the model to ONNX or applying to a custom dataset
* Encourage trying YOLO, SSD, or RetinaNet to compare head styles

**✅ Conclusion**

This assignment provided me with foundational experience in object detection. While my initial submission was trained for only 2 epochs, extending it to 13 epochs highlighted the value of iterative training and hyperparameter tuning. I now feel confident building on this work, adapting to new architectures, and even handling custom datasets or real-world applications.

Thank you for the opportunity!