Experience Report

This report details the development process of a multi-label object detection model trained on the Pascal VOC 2007 dataset. The project initially began with a baseline model EfficientNetB0 as a backbone and an object detection approach of FPN. The development evolved through various stages of troubleshooting, architectural improvements, and performance optimizations. The integration of AI assistance, particularly ChatGPT and the Gemini code editor in Colab, played the main role in overcoming development challenges and accelerating progress.

**1. Initial Model and Challenges**

The initial approach utilized EfficientNetB0 with basic bounding box regression and classification heads trained using cross-entropy loss. Despite functional training code and data parsing routines, the model demonstrated limited accuracy (~33%), revealing shortcomings in the architecture and training setup.

Key challenges encountered included:

* Limited backbone capacity leading to underfitting.
* Single-label classification per anchor restricting multi-object detection.
* Mismatch between predicted output shapes and ground truth label formats causing training crashes.
* Insufficient evaluation metrics to assess model performance meaningfully.
* Lack of bounding box filtering resulting in overlapping detections.
* Complexities in handling multi-object annotations and generating multi-label ground truths.

**2. AI Assistance and Technical Resolutions**

Utilizing ChatGPT significantly contributed to overcoming the the challenges through:

* **Model Architecture Enhancements:**  
  Upgrading the backbone to EfficientNetB3, then to ResNet50 with Feature Pyramid Networks (FPN) to capture multi-scale features improved representation capacity.
* **Multi-Label Classification Transition:**  
  Shifting from softmax to sigmoid activation and replacing focal loss with binary crossentropy facilitated multi-class, multi-object detection per anchor.
* **Data Label Handling:**  
  Implementing utilities to reshape and expand ground truth labels aligned training inputs with model outputs, resolving dimensionality conflicts.
* **Advanced Evaluation Metrics:**  
  Introducing mean Average Precision (mAP) and Precision-Recall curves provided an understanding of detection quality beyond accuracy.
* **Bounding Box Post-Processing:**  
  Integrating Non-Maximum Suppression (NMS) reduced redundant bounding boxes, improving interpretability of visual predictions.
* **Visualization Tools:**  
  Adding functions to overlay bounding boxes with class labels on images enhanced qualitative model assessment.

**3. Usage of Gemini Code Editor in Colab**

The Gemini AI code editor within Colab notebooks proved invaluable for making precise, incremental code adjustments without disrupting the overall workflow. It facilitated:

* Quick refactoring and testing of loss functions, label formatting, and model layers.
* Efficient debugging of shape mismatches and runtime errors.
* Seamless integration of AI-suggested code snippets.

This interactive coding environment accelerated iterative development cycles.

**4. Learning Outcomes**

The project deepened understanding of several key concepts:

* The distinction between multi-class and multi-label object detection and corresponding architectural implications.
* The advantages of GIoU loss for bounding box regression and focal loss for class imbalance handling. (This was something very new for me)
* The complexities of data preprocessing for multi-object datasets.

**5. Balancing Manual Coding and AI Assistance**

The collaboration between manual programming and AI guidance yielded significant benefits:

* AI rapidly identified errors and suggested fixes, reducing debugging time.
* AI-provided architectural and methodological improvements enhanced model accuracy.
* Explanations and examples from AI provided deeper conceptual and practical knowledge.
* Manual validation and adaptation of AI suggestions ensured relevance and correctness.

**7. Conclusion**

Model 1 (EfficientNetB3 + FPN) provided an accuracy of 60%. Shows model is learning very good on imbalanced data as well as not overfitting.

Model 2 (ResNet50 + FPN) provided an accuracy of 75% providing better results that EfficientNetB3.

**Rough Thoughts:**

1. Started with EfficientNetB0 and FPN as I wanted to try a new model and learn how it works. It was fun experience there.
2. Was very upset with the initial accuracy, thought of using the already tried models as it is always easy to work with them.
3. Soon discarded that thought as learning is the key. Got to know about the other variants of EfficientNet. Tried applying those.
4. Learnt about new concepts (thanks to ChatGPT) to help my model work better. (Will definitely be using these further).
5. Was still not happy with the metrics so changed the base model to ResNet (Something which I have worked with before) and thought of using it with FPN only.
6. Gemini integrated in collab helped a lot with missed lines of code from ChatGPT which threw error that even ChatGPT couldn’t understand.
7. Main problem faced: GPU. Can never work them on free version of Collab but it worked fast enough on L4 GPU (T4 for some reason was not able to run it).