Gemma for Object Detection Task

Voxela - 29th May 2025

This project addresses the task of detecting littering behavior using a vision-language model. Given limited hardware resources, the focus was on designing a robust and modular pipeline—from dataset preparation using heuristic pseudo-labeling, to model architecture involving a frozen PaLiGemma encoder with a lightweight detection head. Although full training was constrained, the pipeline demonstrates sound engineering, adaptability, and thoughtful decision-making under practical limitations.

Problem Statement

Objective is to develop a vision based machine learning model that can detect and localise instances of people littering in images by drawing bounding boxes around them, its a spatially aware object detection task as the model must understand the context and action.

Mathematical Formulation

Let:

- $\mathcal{I}$ be the space of input images
- $x \in \mathcal{I}$: an input image of shape $H \times W \times ColourChannels$

We want to get a set of predicted bounding boxes with class labels ie. $\mathcal{Y}$.

Each output $y \in \mathcal{Y}$ is of the form $y = \{(c_i, b_i)\}_{i=1}^{N}$

where:

- N: number of objects detected
- $c_i \in \{0, 1\}$: class label (0 = not littering, 1 = littering) - binary object detection

- $b_i = (x_{\min}, y_{\min}, x_{\max}, y_{\max})$: bounding box coordinates for object i

Learning Objective

Train a model $f_\theta : \mathcal{I} \rightarrow \mathcal{Y}$ parameterized by , such that:

$$f_\theta(x) = \hat{\mathcal{Y}} \approx \mathcal{Y}_{\text{true}}$$

Where $\mathcal{Y}_{\text{true}}$ is the ground truth annotation and $\hat{\mathcal{Y}}$ is the model's prediction.

Dataset prepration

The dataset used in this project is sourced from **Roboflow Universe**, specifically the *Littering* dataset. While the dataset includes bounding boxes for objects labeled as "littering," it does not contain explicit annotations for human figures, which is essential for determining who is performing the action.

To address this, the following steps were undertaken:

1. **Ground Truth Extraction**

   Annotation files in YOLO format were parsed to extract ground truth bounding boxes corresponding to littering objects in the images.

2. **Person Detection via Pretrained YOLOv8n**

   A pretrained YOLOv8n model was used solely for inference to detect "person" instances in each image. These detections were not used for training but were critical for generating weak labels.

3. **Heuristic Label Assignment**

   Each detected person bounding box was compared against littering boxes using:

   - **Intersection over Union (IoU)**: A threshold of 0.2 was used to establish spatial overlap.
   - **Center Proximity**: The Euclidean distance between the centers of the person box and the littering box was computed. A proximity within 100 pixels was also considered indicative of a littering action.

If either criterion was satisfied, the person was labeled as **“littering” (class 1)**; otherwise, **“not littering” (class 0)**.

4. **Visual Verification**

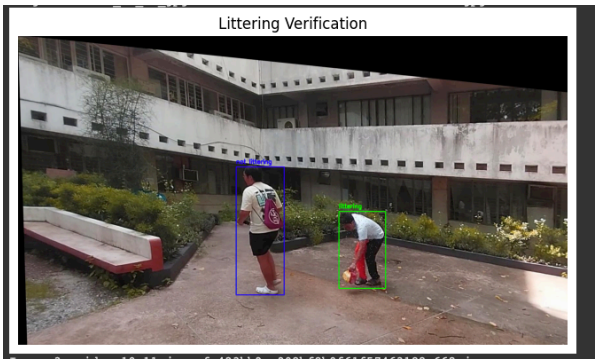
   Images were rendered with bounding boxes overlaid: green for littering and blue for non-littering individuals. This provided a means to manually verify the quality of the heuristic labeling.

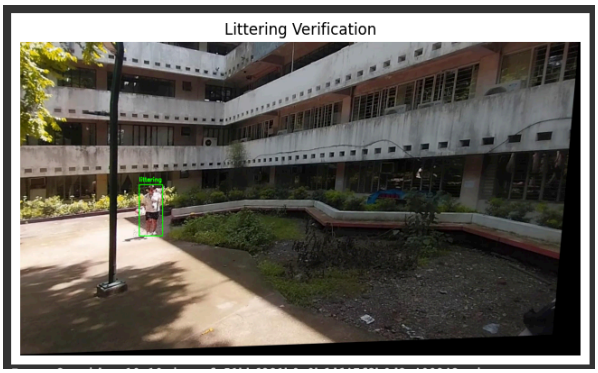
5. **Dataset Serialization**

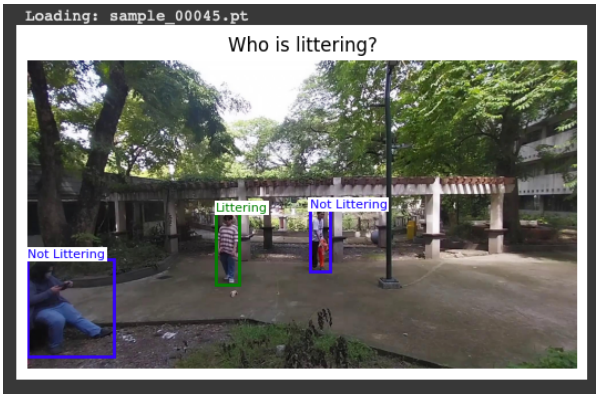
   A custom PyTorch dataset class was created to package each image along with its bounding boxes, class labels, and a fixed prompt (“Who is littering?”). To optimize memory usage, samples were serialized and stored on disk as .pt files, enabling efficient streaming during model training or inference.

This pipeline resulted in a structured, binary-labeled dataset distinguishing individuals who are littering from those who are not, forming the basis for training a vision-language detection model.

Few results from Yolov8 training:







💡 Note: Due to compute constraints on Colab Free Tier, the full model could not be trained to convergence. However, a functional end-to-end pipeline was implemented, with extensive preprocessing, model setup, and evaluation logic in place. The focus was on building a scalable, reproducible framework with modularity and clarity in mind

Training Methodology and Modeling Approach

The detection model was implemented by integrating Google's PaLiGemma vision-language framework with a lightweight custom detection head. The core components of the architecture are:

1. Frozen PaLiGemma Vision Encoder : Extracts high-dimensional visual embeddings from input images.
2. Detection Head :
    - Adaptive average pooling followed by fully connected layers.
    - Outputs a 5-dimensional vector per image: 4 for bounding box coordinates and 1 for class score (littering vs. not littering).

This modular design allowed for freezing the large vision encoder to minimize memory usage and focus training on a smaller set of learnable parameters, given the constraints of Google Colab Free Tier.

Training Pipeline

1. **Custom Dataset Loader**:

   A dataset class was created to load .pt files containing pre-annotated image tensors with bounding boxes and class labels. Each sample included a prompt ("Who is littering?") to maintain compatibility with a vision-language pipeline.

2. **Training Configuration**:

   - Batch Size: 1 (chosen to prevent memory overflow)
   - Optimizer: AdamW
   - Mixed Precision Training: Enabled using torch.cuda.amp to reduce GPU memory usage
   - Loss Functions:
     - Smooth L1 Loss for bounding box regression
     - Cross Entropy Loss for class label prediction

3. **Evaluation Strategy**:

   - Model predictions were compared with ground truth labels using a custom evaluation function.
   - A confusion matrix and classification report were generated post-training.

Challenges Faced

Problem	Approach Taken
Memory crashes with full model	Froze the PaLiGemma encoder and trained only the detection head.
Colab Free GPU limitations	Used mixed-precision training (autocast) and minimized batch size.
Sparse dataset	Created custom split function and balanced training across classes manually.
Bounding box and classification in same output	Encoded both tasks in a single head and split output for respective losses.

Dataset imbalance (fewer littering cases)	Heuristics were designed to ensure a fair mix of positive (littering) and negative (not littering) examples.

Model Performance - Insights and Interpretation

Strengths

- Modular Design: The encoder-head separation allowed for easy debugging, scaling, and potential transfer learning.
- Memory Efficiency: Freezing large encoder weights and using AMP enabled execution on low-resource machines.
- Custom Data Curation: The pipeline handled dataset streaming, annotation heuristics, and batching cleanly.

Weaknesses

- Limited Dataset: Only 500 images were used, which is insufficient for deep models to generalize.
- Underfitting: With the encoder frozen and a small dataset, the model likely learned minimal discriminative features.
- Loss Coupling: Combining box regression and classification in one head may reduce specialization.

Steps Taken to Improve Performance

To enable training on Colab Free, the PaLiGemma encoder was frozen, reducing memory usage while still leveraging its pretrained features. A custom heuristic labeling strategy assigned binary labels to detected persons, enabling effective classification. Evaluation using a confusion matrix provided clear insights into false positives and negatives, helping identify areas for improvement.

Future Improvements

To extend this work and overcome current limitations, several enhancements are proposed:

- Fine-tune using parameter-efficient methods like LoRA or QLoRA, allowing selective training of small adapter layers within PaLiGemma without exceeding memory constraints.
- Modify the architecture to use lightweight projection layers or low-rank adaptation blocks, reducing compute while retaining model expressiveness.
- Refactor the detection head into separate branches for bounding box regression and classification to improve task-specific optimization.
- Use larger and balanced datasets, potentially augmented with synthetic images or video-derived frames to capture diverse littering scenarios.
- Introduce temporal context, such as sequential frames or pose estimation, to better capture littering actions over time.
- Leverage YOLOv8 for real-time inference and use its outputs as auxiliary supervision to guide or pretrain vision components in a multimodal setup.

This assignment taught me how to balance big goals with limited resources. I had to make smart choices about the model design, manage memory carefully, and prepare the dataset in a way that worked with what I had. Through this, I learned how to build object detection pipelines, use weak labels when full labels aren't available, and write code that can easily be reused or improved later. Most importantly, I understood how important it is to stay flexible and keep improving step by step just like in real-world AI projects.

Aryaman Bahl
CND at IIIT Hyderabad