**Plexor Junior AI Engineer**
**YOLO Object Detection Documentation**

**1. Labeling:**

- **a)** Tools: Used Makesense.ai to label frames. It is an easy to use and efficient web-based labeling tool
- **b)** Classes: Chose 2 classes for this
  - **i.** Person
  - **ii.** Item
- **c)** Frame: Extracted frames with an interval of 10 for the longer video (total 2455 frames, extracted 246) and an interval of 2 for the shorter video (total 204 frames, extracted 102)

**2. Training:**

- **a)** Data split: 80 train, 20 val
- **b)** Model: YOLOv8s
- **c)** Computation: Used Google Colab with T4 GPU (limited resource one of the main challenges in this project)
- **d)** Hyperparameters
  - **i. Test 1:**
    Epochs: 50 | Imgsz: 640 | Batch: 16 | Device: cuda (T4)
  - **ii. Test 2:**
    Epochs: 100 | Imgsz: 640 | Batch: 8 | Device: cuda (T4)

**3. Results:**

| | **Avg Inference** | **mAP 50** | **Recall** | **Precision** |
|---|---|---|---|---|
| **Test 1** | 11.75 | 51.8% | 51.8% | 52.18% |
| **Test 2** | 10.55 | 92.13% | 86.36% | 95.01% |

**4. Challenges/Solutions:**

- **a)** The first challenge was to *accurately* train the model using the T4 GPU, it only runs for a limited time before timing out on long videos. YOLO comes with a neat "stream" parameter that stores results one frame a time, which sped up training and prevented out of memory error.
- **b)** The image quality was poor, especially for the shelf_theft video, which made identifying items difficult.
- **c)** I tried multiple test runs with different conditions (frames, hyperparameters, etc) The current version (test 2) gave very good results because the frames were adequate and labeled well.
- **d)** Class imbalance was also a real problem, you can see the person in almost every single frame, whereas the item was seen only a fraction of the time. Changing batch size from 16 to 8 helped improve the model significantly (15 batches per epoch instead of 7 helped the model detect items better)