DAYTIME WORKING RGB DATA I/P

1→ SETTING UP ENVIRONMENT AND ACTIVATING IT

```
(base) yash@Yashs-MacBook-Air-3 ~ % conda env list

# conda environments:
#
base                  * /opt/anaconda3
helper                  /opt/anaconda3/envs/helper

(base) yash@Yashs-MacBook-Air-3 ~ % conda create -n capstone python=3.10
Channels:
 - conda-forge
 - defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done
```

2→ CHECKING THE CHANNELS ALREADY PRESENT

```
(base) yash@Yashs-MacBook-Air-3 ~ % conda activate capstone
(capstone) yash@Yashs-MacBook-Air-3 ~ % conda config --show-sources
==> /opt/anaconda3/.condarc <==
channels:
  - https://repo.anaconda.com/pkgs/main
  - https://repo.anaconda.com/pkgs/r

==> /Users/yash/.condarc <==
default_python: None
channels:
  - conda-forge
  - defaults
```

3→ INSTALLING PACKAGES FROM CONDA AND PIP

```
(capstone) yash@Yashs-MacBook-Air-3 ~ % conda install ultralytics matplotlib opencv
Channels:
 - conda-forge
 - defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3/envs/capstone

  added / updated specs:
    - matplotlib
    - opencv
    - ultralytics
```

4→ INSTALLING JUPYTER LAB AND THEN LAUNCHING IT

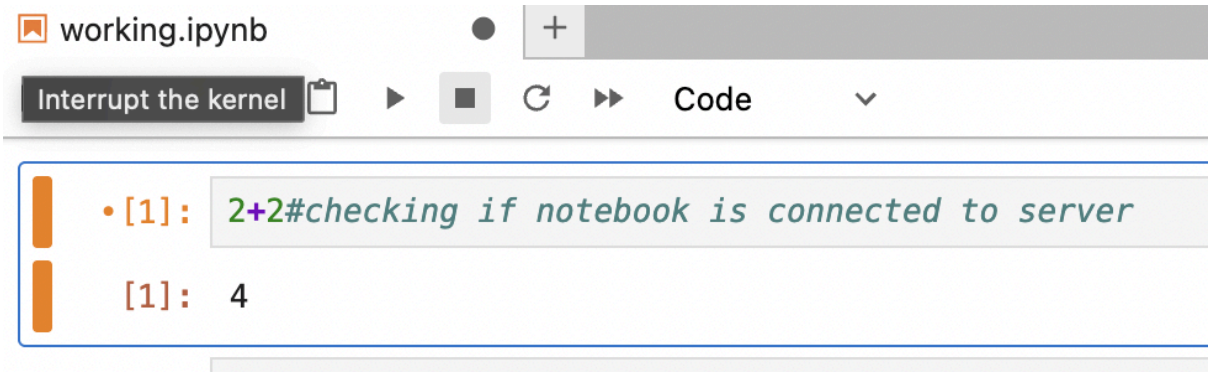
```
(capstone) yash@Yashs-MacBook-Air-3 ~ % conda install jupyterlab
Channels:
 - conda-forge
 - defaults
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: done
```

LAUNCH→ OPENS IN BROWSER(DEFAULT)

```
(capstone) yash@Yashs-MacBook-Air-3 ~ % jupyter-lab
[I 2025-03-19 23:00:37.588 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2025-03-19 23:00:37.590 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2025-03-19 23:00:37.592 ServerApp] jupyterlab | extension was successfully linked.
[I 2025-03-19 23:00:37.979 ServerApp] notebook_shim | extension was successfully linked.
[I 2025-03-19 23:00:38.010 ServerApp] notebook_shim | extension was successfully loaded.
[I 2025-03-19 23:00:38.012 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2025-03-19 23:00:38.013 ServerApp] jupyter_server_terminals | extension was successfully loaded.
```

NOTE→ Pip installs only python based packages whereas conda installs packages with larger dependencies.like(cpp,opencv). Conda is faster than anaconda navigator and feels nice lol.

5→ create a python notebook to run script and check if it is connected to server. If not then reactivate environment.



6→ importing and setting YOLO

[18]:
```python
from ultralytics import YOLO
import cv2
#setting device to cpu or gpu or anything else use torch for that
import torch

# Check for computing device
if torch.backends.mps.is_available():
    device = "mps"  # Use Metal Performance Shaders (MPS) for Macbook

elif torch.has_vulkan:  # Check if Vulkan is available (Raspberry Pi 5 may support it)
    device = "vulkan"  #so basically Vulkan is gpu for r
else:
    device = "cpu"   # Default to CPU

model = YOLO("yolov8n.pt").to(device)
```

NOTE→device is the computing element that is gonna be used. It can be cpu, gpu or mps in case of mac. In case of our Raspberry Pi5 it is most likely gonna be our general CPU but we can use external gpu to improve speed. A gpu WONT improve accuracy but it will improve inference speed.

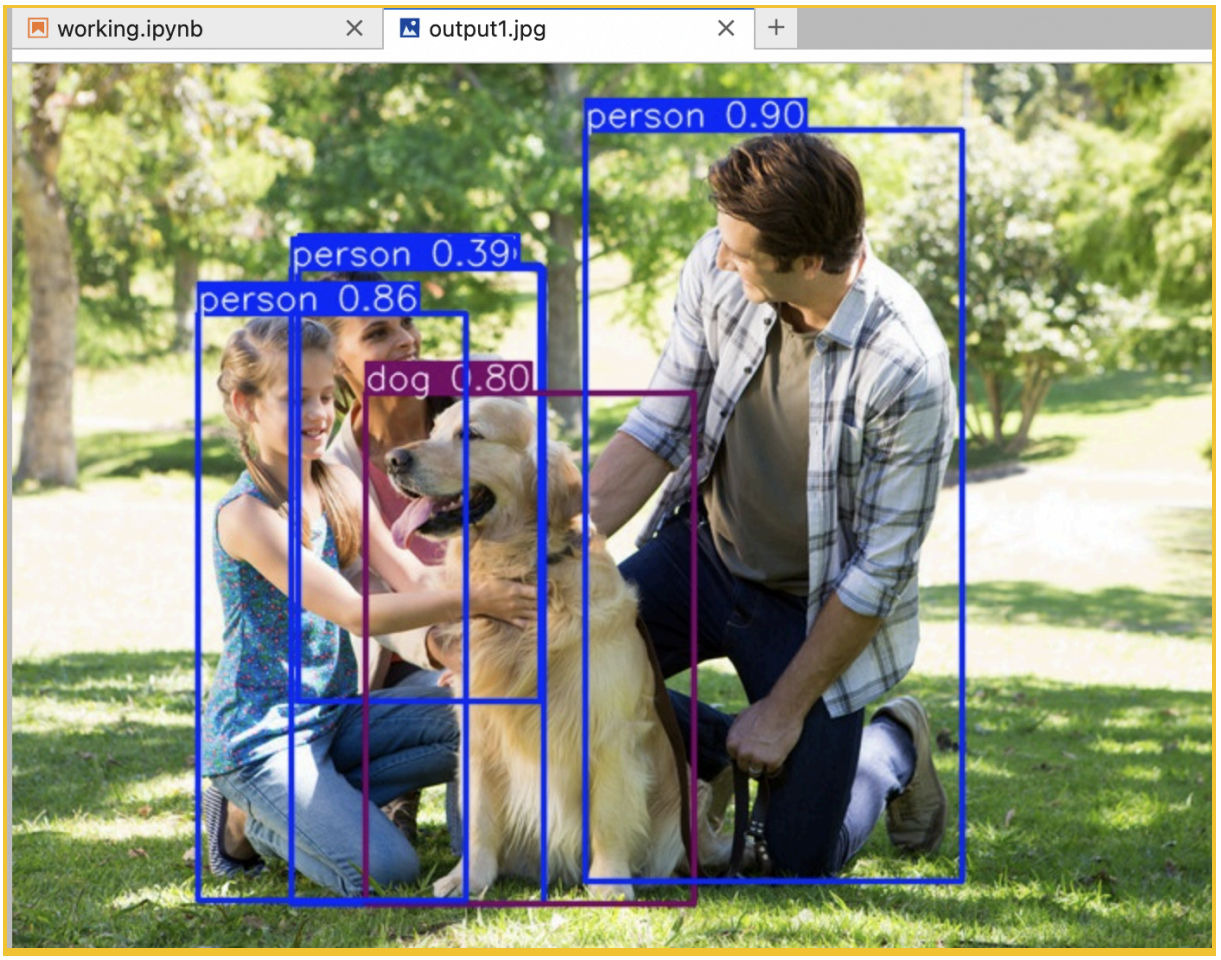
IMPROVING ACCURACY→

1. **Better dataset quality** (clean labels, diverse samples).
2. **Improved model architecture** (switch to a better YOLO variant, like YOLOv8/YOLOv11).
3. **More training epochs** (if fine-tuning your model).
4. **Hyperparameter tuning** (learning rate, batch size, etc.).

7→ running model on a single image to identify PEOPLE

```python
import matplotlib.pyplot as plt
image_path = "tester1.jpg"  # Change this to your image file
img = cv2.imread(image_path)

# Perform inference
results = model(image_path)

# Plot results
for r in results:
    r.show()  # Show results in a window

    # Optionally, save the results
    r.save(filename="output1.jpg")
```

OUTPUT IMAGE→

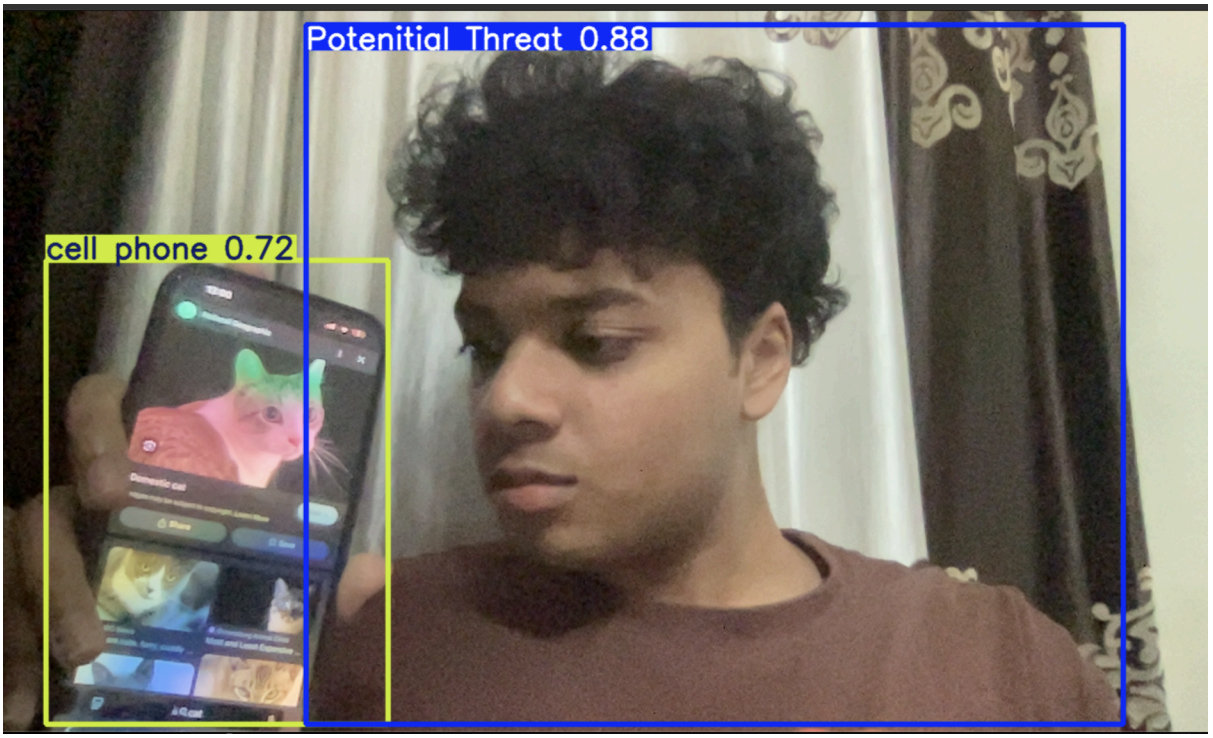


NOTE→ We will change the person class label to threat as people are a threat for our rover. But are all people a threat???

8→ Implementing Model on Live Feed and checking if threat is there?

```python
import cv2
import torch
from ultralytics import YOLO

# Load YOLO model (replace with your YOLOv11 model if available)
model = YOLO("yolov8n.pt")  # Use your trained model if different

# Open webcam
cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Run YOLO inference
    results = model(frame)

    # Get detected objects
    for r in results:
        for i in range(len(r.boxes.cls)):  # Iterate over detected objects
            class_id = int(r.boxes.cls[i])  # Get class ID
            if model.names[class_id] == "person":  # Check if detected class is "person"
                r.names[class_id] = "Potenitial Threat"  # Change label dynamically

    # Draw results on frame
    annotated_frame = results[0].plot()

    # Display frame
    cv2.imshow("Human Detection", annotated_frame)

    # Stop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT ON LIVE FEED→



Successfully Identified potential threat.

Challenges→

1) currently model says that every person is a threat. How to distinguish?

2)Need to prepare Dataset for this

3)How to Fine tune the Model on our dataset?