**Duality AI Problem statement Documentation**

**Title**

**Team Name: Team Dracarys**

**Team Members: C G KISHORE(**[**cgkishore03@gmail.com**](mailto:cgkishore03@gmail.com)**)**

**M. Muni Chandra Goud (**[**chandum7824@gmail.com**](mailto:chandum7824@gmail.com)**)**

**Project name: fody**

**Tagline: fody – A fine tuned object detection model using Duality AI**

**Complete Methodology from training to Evaluation**

**Dataset**

**Access & download**

* **Go to Falcon Duality AI and log in / sign up.** [**(falcon\_signup\_link)**](https://falcon.duality.ai/auth/sign-up?utm_source=hackathon&utm_medium=instructions&utm_campaign=hyderabad)
* **Open the dataset page and click the provided download links.** [**(download\_dataset\_link)**](https://falcon.duality.ai/secure/documentation/7-class-hackathon&utm_source=hackathon&utm_medium=instructions&utm_campaign=hyderabad)
* **Download the three main packages: train, test, and scripts (each is a separate zip or archive).**

**Dataset Overview**

**Source & Generation**

* Dataset generated using **FalconEditor** on Falcon Duality AI.
* Scene simulated in a **space station environment** to replicate real-world safety-critical conditions.

**Key Characteristics**

* Includes **varying lighting conditions** to improve robustness in detection.
* Contains **occlusions and overlapping objects**, simulating realistic cluttered scenarios.
* Captured from **diverse camera angles and distances** for generalization.

**Target Object Categories (7 classes)**

1. Oxygen Tank
2. Nitrogen Tank
3. First Aid Box
4. Fire Alarm
5. Safety Switch Panel
6. Emergency Phone
7. Fire Extinguisher

**Image Counts per Split**

**Training dataset folder contains:**

* **Training Set**: 1,767 images + corresponding labels
* **Validation Set**: 336 images + corresponding labels

**Test dataset folder contains**:

* **Test Set**: 1,408 images + corresponding labels

**Scripts Provided**

* Dataset also included a **scripts folder** containing:
* Training and testing scripts
* Example YAML configuration files

**Preprocessing**

* No additional preprocessing was performed, as the dataset was already:
  + Properly annotated in YOLO format (image–label pairs).
  + Balanced with **diverse angles, lighting conditions, and distances**.
  + Split into train/validation/test sets in suitable structure.

**YOLO Model Selection**

**Chosen Model**

* Selected **YOLOv11s** (small variant of YOLOv11).
* Reason:
  + Lightweight architecture with fewer parameters.
  + Optimized for **real-time detection** on limited hardware.
  + Faster inference speed while maintaining good accuracy.
  + Suitable for deployment in **resource-constrained environments** (e.g., mobile or edge devices).

**Model Training Setup**

* **Environment:** Training was performed on **Google Colab** to leverage GPU support, which is required for fine-tuning YOLO models efficiently.
* **Dataset Preparation:** The training dataset was uploaded as a **ZIP folder to Google Drive** and extracted directly in Colab.
* **How to Download YOLOv11s Model**
  + Step 1: Install the Ultralytics YOLO package: pip install ultralytics
  + Step 2: Import YOLO in Python: from ultralytics import YOLO
  + Load the YOLOv11s pre-trained model (this will automatically download the weights): model = YOLO('yolov11s.pt')
* **Configuration:** A **YAML file** was created specifying the paths for the training, validation, and test folders.

**Example : train: #path to training folder(s)**

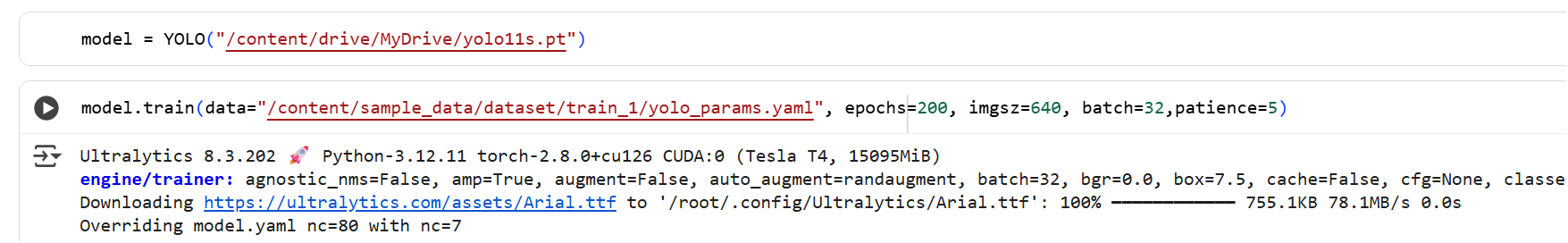
**val: #path to validation folder(s)**

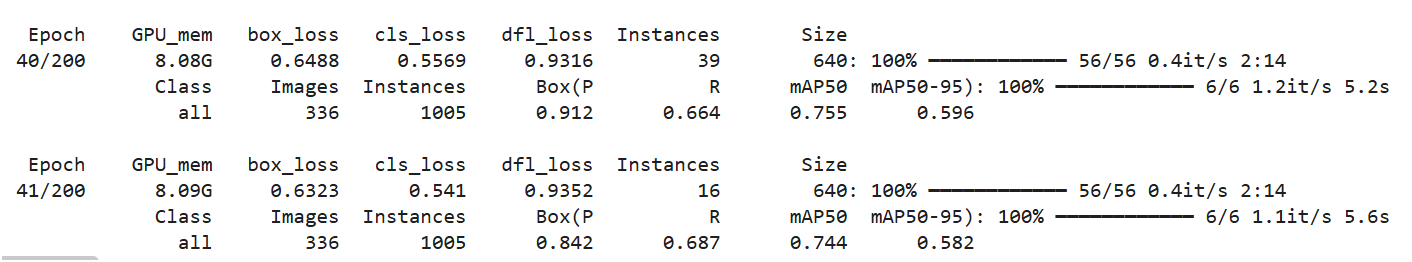
**test: #path to testing images**

**nc: 7**

**names: ['OxygenTank', 'NitrogenTank', 'FirstAidBox', 'FireAlarm', 'SafetySwitchPanel', 'EmergencyPhone', 'FireExtinguisher']**

* **Training Parameters:**
  + **Epochs:** 200 (training was stopped at the 41st epoch due to Google Colab usage limits)
  + **Batch Size:** 32
  + **Patience :** 20
  + **Learning Rate & Optimizer:** Default YOLOv11s settings were used, as they are well-optimized for YOLO training.





* **Model Saving:** After training, the **runs and the trained model weights** were saved back to Google Drive for later evaluation and deployment.

**Performance During Training :** After training the YOLOv11s model for 41 epochs, the model achieved a **mAP@0.5 score of 0.744** on the validation dataset. This indicates that the model is performing well in detecting and classifying the target objects with good accuracy.

**Check out this note book for complete training process(click on below link):**

[Model\_Training\_notebook\_Link](https://github.com/CGKishore/Foyo/blob/main/Model%20Training.ipynb)

**Performance Evaluation of Trained Model:**

**Evaluation Dataset**

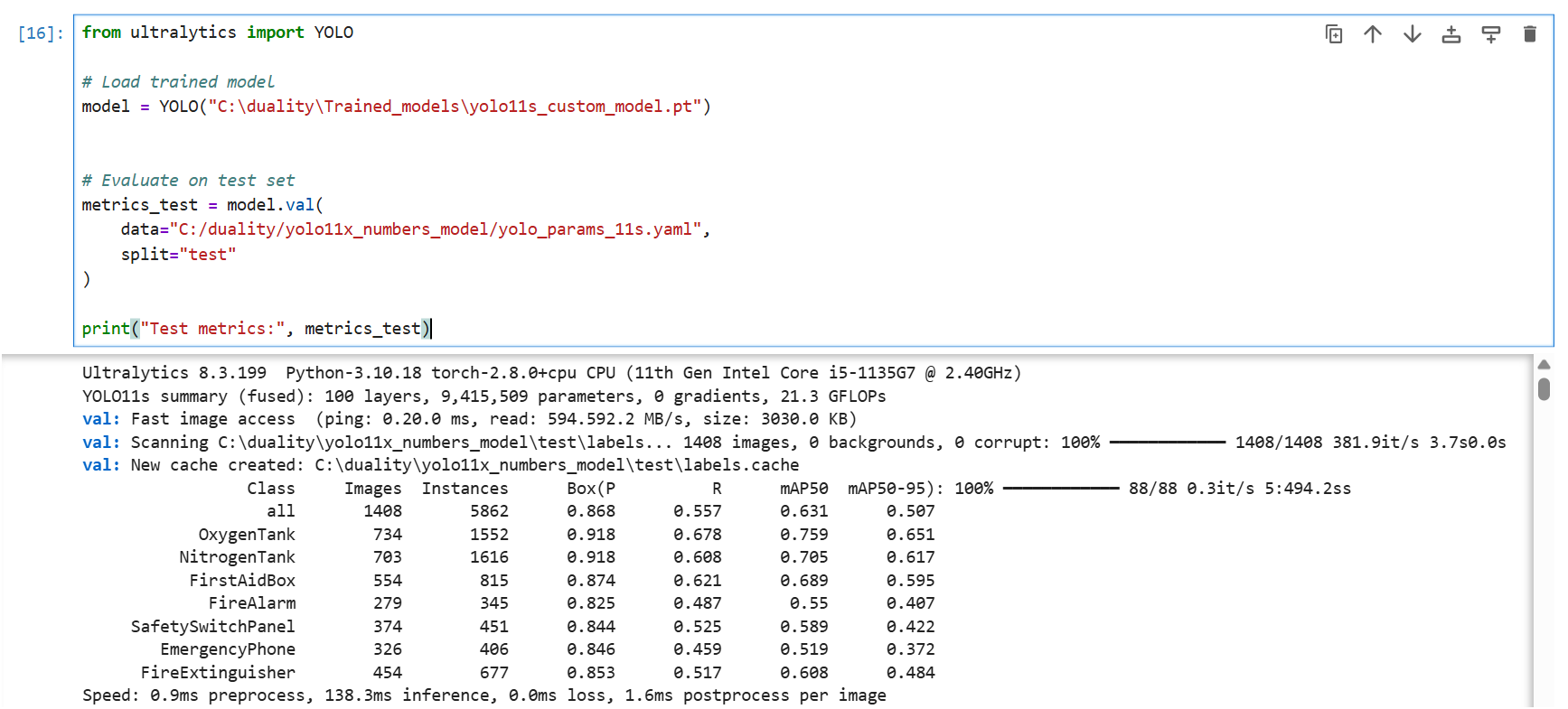
* The **test dataset** was used to evaluate the model.
* This dataset contains images that the model has not seen during training and validation.

**Evaluation Environment**

* Evaluation was performed **locally in Jupyter Notebook using CPU only**.
* GPU resources were **not required** since only inference and metric calculations were done.

**Evaluation Model Training**

* **Install and Import Required Package :** pip install ultralytics
* **Import YOLO Module**
* **Evaluate on Test Dataset**
* **View Evaluation result**

****

**Complete Evaluation of model script :** [**Model\_Evaluation\_Notebook\_link**](https://github.com/CGKishore/Foyo/blob/main/Model%20Evaluation.ipynb)

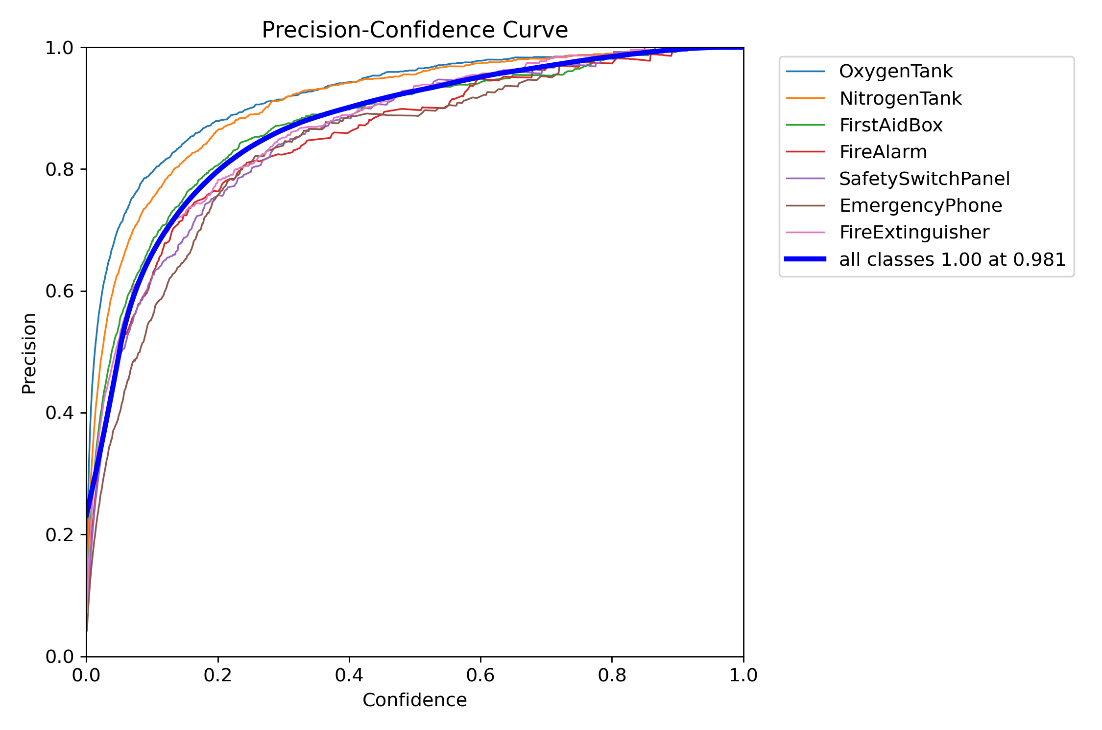
**Results & Performance Metrics**

**mAP Scores**

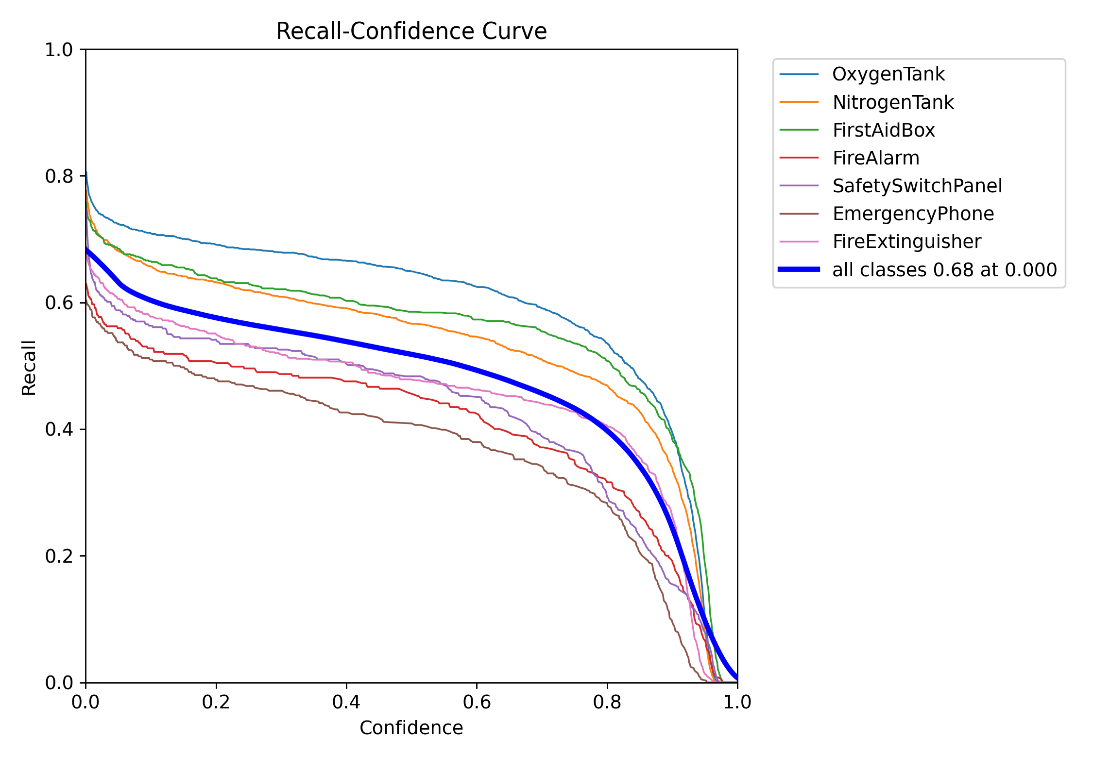
* **mAP@0.5:** 0.6312
* **mAP@0.5:0.95:** 0.5066
* **Class-wise mAP@0.5:**
  + OxygenTank: 0.6505
  + NitrogenTank: 0.6168
  + FirstAidBox: 0.5951
  + FireAlarm: 0.4065
  + SafetySwitchPanel: 0.4220
  + EmergencyPhone: 0.3719
  + FireExtinguisher: 0.4836

**Precision, Recall**

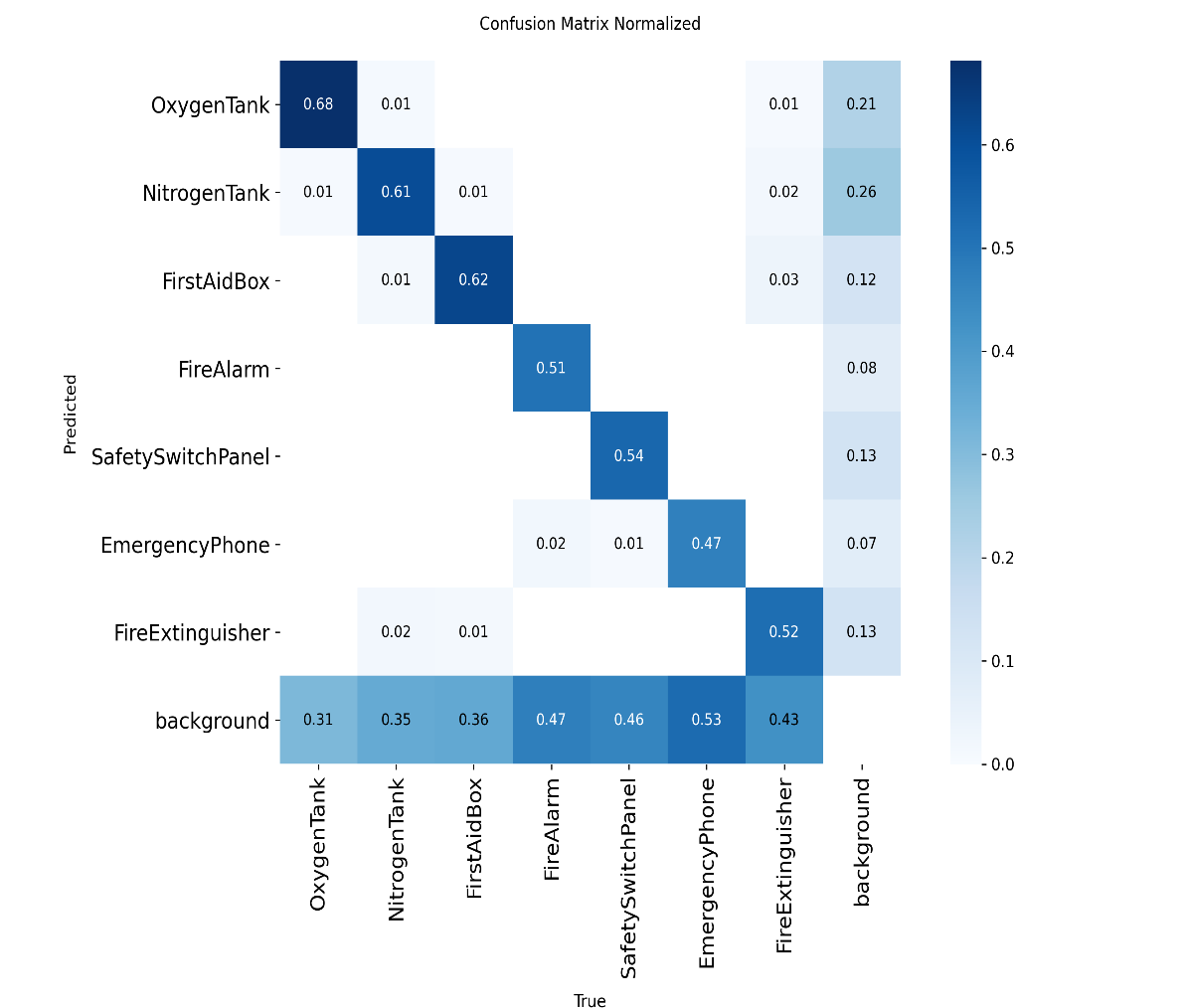
* **Overall Precision:** 0.8684 (high, meaning most detected objects are correct)



* **Overall Recall:** 0.5565(moderate, meaning some objects were missed)



**Confusion Matrix**

****

**True Positives (Diagonal Values):**

* The main diagonal of the confusion matrix represents the **correctly detected objects**.
* For example:
  + **OxygenTank:** 68% of actual objects were correctly identified (Recall = 0.68)
  + **FirstAidBox:** 62% correctly identified
  + **SafetySwitchPanel:** 54% correctly identified
* A high value on the diagonal indicates strong detection performance for that class.

**False Negatives (Missed Detections):**

* The 'background' row represents **objects missed entirely** by the model.
* For instance:
  + 31% of OxygenTank objects were missed
  + 35% of NitrogenTank objects were missed
  + 43% of FireExtinguisher objects were missed
* These false negatives highlight the main limitation of the model, often caused by **small object size, low resolution, or occlusion**.

**False Positives (Misclassifications):**

* Off-diagonal cells show **misclassifications**, where objects are assigned the wrong class.
* Examples include:
  + 1% of OxygenTank objects predicted as NitrogenTank
  + 2% of FireAlarm objects predicted as EmergencyPhone
* In this model, the misclassification rate between classes is relatively low, which is positive.

**Inference Speed**

* Preprocessing: 0.85 ms/image, Inference: 138.25 ms/image (CPU), Postprocessing: 1.63 ms/image

**Conclusion & Observations**

**Success Cases**

* The model performs well on dominant objects like OxygenTank and NitrogenTank.
* **Graph/Image:** Include a few sample images showing **correct detections with bounding boxes and labels**.

**Failure Cases**

* Some smaller or occluded objects are missed (EmergencyPhone, FireAlarm).
* False positives occur occasionally due to overlapping objects.
* **Graph/Image:** Include sample images highlighting **missed detections or misclassifications**.

**Summary:**

* Overall, the model demonstrates **strong true positive detection**, especially for larger or more distinct objects.
* The **high false negative rate** is the main area for improvement, we can improve by train the model with more data with that labels.