Team Name
CODE VAIKUNTH

TITLE
UNLOCKING OBJECTS- Space Station Object Detection Model.
DUALTIY-AI SPACE STATION HACKATHON REPORT

Defining Our Vision

Cosmic Clarity-Unveiling Objects with Advanced AI
Leveraging artificial intelligence to achieve highly accurate object detection, crucial for ensuring safety and efficiency aboard space stations.

Methodology

Our approach for the Duality AI Space Station Hackathon is centered on training a robust object detection model using a high-quality synthetic dataset provided by Duality AI's Falcon digital twin simulation platform. This section outlines the initial steps undertaken to set up our development environment, integrate the dataset, and establish a baseline model performance.

1. Environment Setup: To ensure a consistent and efficient development workflow, we established a dedicated Python environment. Utilizing Anaconda, we ran the `setup_env.bat` script (found within the `ENV_SETUP` subfolder of the provided `Hackathon_Dataset`). This script automatically configured an environment named "EDU," installing all necessary dependencies and libraries required for the YOLOv8 object detection framework, including PyTorch and other relevant packages. This setup ensures a standardized and reproducible environment for model training and evaluation.

2. Dataset Integration:

The synthetic dataset, crucial for training our model in a simulated space station environment, was downloaded to our local machine. This comprehensive dataset is pre-processed and organized, containing images along with YOLO-compatible labels structured into distinct Train, Validation, and Test folders. The dataset is designed to replicate various challenging scenarios within a space station, such as diverse lighting conditions, varied object angles, and instances of occlusion for

the target objects: "Toolbox," "Oxygen Tank," and "Fire Extinguisher."

3. Baseline Model Training:

With the "EDU" environment activated and the dataset prepared, we initiated the training of our initial YOLOv8 object detection model. The `train.py` script, supplied within the dataset's root directory, was executed to commence this process. During training, the system automatically generated and saved performance logs and model checkpoints within the `runs/` directory. This initial training phase is vital for understanding the default model behavior and performance characteristics.

4. Initial Performance Benchmark:

Upon completion of the baseline training, we proceeded to evaluate the model's performance on the unseen test dataset. By executing the `predict.py` script (also provided), we obtained crucial performance metrics. These include the Mean Average Precision (mAP@0.5 IoU), Precision, Recall, and a detailed Confusion Matrix. These initial results serve as our fundamental benchmark, providing insights into the model's current detection capabilities across different object classes and guiding our subsequent iterative optimization strategies to enhance accuracy, generalizability, and efficiency.

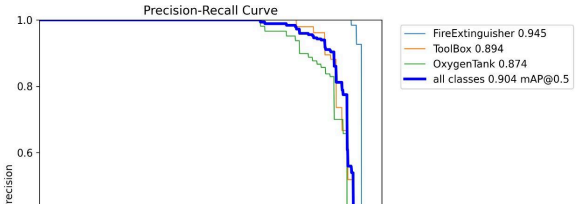
Performance result and metrics

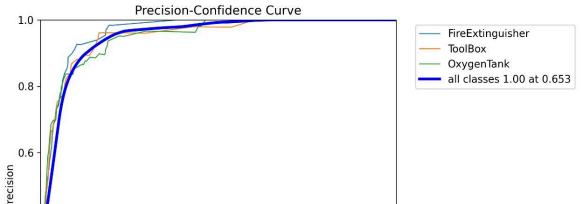
- **Training Result 1**: We've successfully established the baseline performance of our object detection model on the test dataset. This initial evaluation provides crucial metrics and insights that will guide our optimization efforts.Our model achieved an **overall mean Average Precision (mAP) at 0.5 Intersection over Union (IoU) score of 0.805**. This score serves as our foundational benchmark for future improvements.

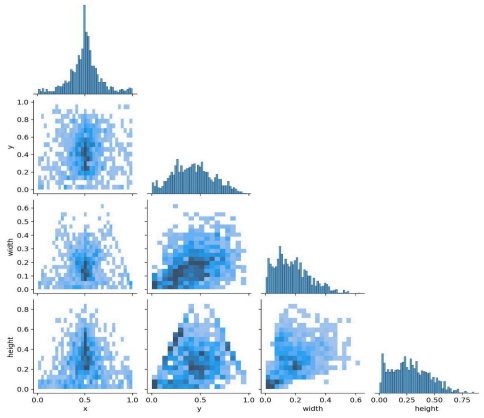
```
s), 191.5ms
Speed: 4.9ms preprocess, 191.5ms inference, 1.8ms postprocess per image at shape (1, 3, 384, 640)
Predicted images saved in C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\predictions\images
Bounding box labels saved in C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\predictions\labels
Model parameters saved in C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\yolo_params.yaml
Ultralytics 8.3.170  Python-3.10.18 torch-2.5.1 CPU (Intel Core(TM) i7-8665U 1.90GHz)
val: Fast image access  (ping: 0.10.0 ms, read: 1447.0200.4 MB/s, size: 2717.2 KB)
val: Scanning C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\data\test\labels... 400 images, 0 backgrounds, 0
val: New cache created: C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\data\test\labels.cache
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          | 25/25
                   all        400        560      0.909      0.698      0.805      0.663
      FireExtinguisher        183        183        0.9       0.77      0.838      0.661
               ToolBox        193        193      0.892      0.683      0.803      0.702
            OxygenTank        184        184      0.937      0.642      0.773      0.625
Speed: 1.5ms preprocess, 227.2ms inference, 0.0ms loss, 1.4ms postprocess per image
Results saved to C:\Users\Admin\runs\detect\val
```

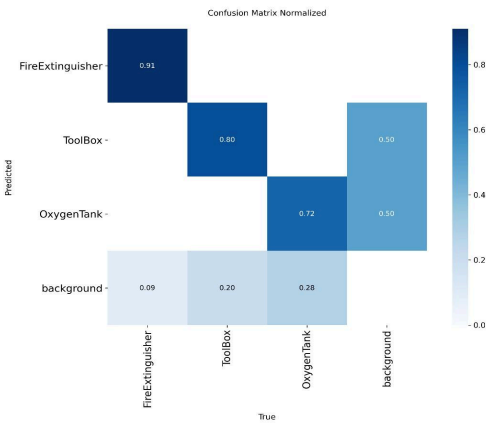
We can see that the model performed best on **FireExtinguisher** (0.838 mAP@0.5) and struggled most with **OxygenTank** (0.773 mAP@0.5). This highlights OxygenTank detection as our primary focus for optimization to achieve significant overall accuracy improvements.

RESULTS FOR TRAINING TEST 1 :







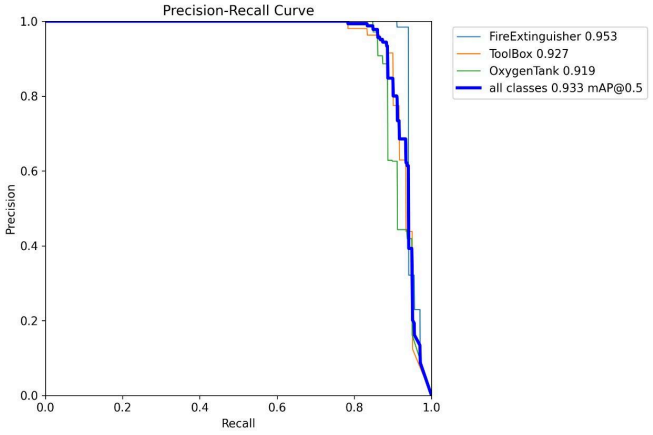


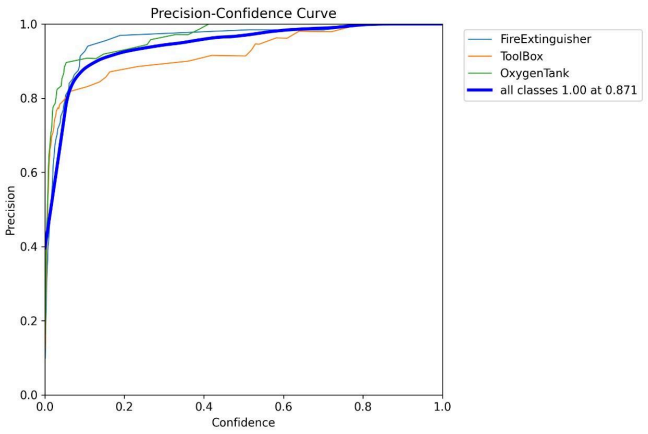
- **Training Result 2**: We've successfully established the baseline performance of our object detection model on the test dataset. This initial evaluation provides crucial metrics and insights that will guide our optimization efforts.Our model achieved an **overall mean Average Precision (mAP) at 0.5 Intersection over Union (IoU) score of 0.865**. This score serves as our foundational benchmark for future improvements.

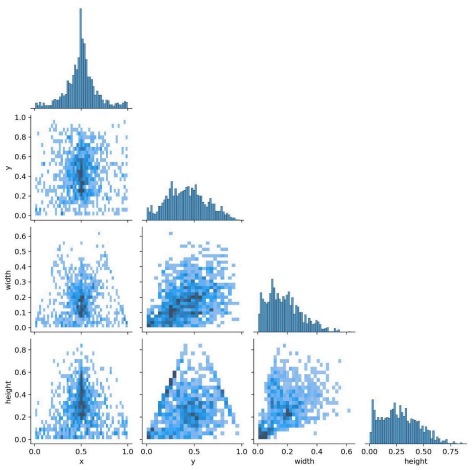
```
Model parameters saved in C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\yolo_params.yaml
Ultralytics 8.3.170  Python-3.10.18 torch-2.5.1 CPU (Intel Core(TM) i7-8665U 1.90GHz)
val: Fast image access  (ping: 0.10.0 ms, read: 1234.6107.8 MB/s, size: 2837.6 KB)
val: Scanning C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset\data\test\labels.cache... 400 images, 0 backgrounds, 0 corrupt: 100%| | 400/400 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%| | 25/25 [01:37<00:00,  3.89s/it]
                   all        400        560      0.948      0.766      0.865      0.735
      FireExtinguisher        183        183      0.913      0.825      0.875       0.73
               ToolBox        193        193      0.959      0.728      0.867      0.757
            OxygenTank        184        184      0.972      0.744      0.853      0.717
Speed: 1.1ms preprocess, 171.7ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to C:\Users\Admin\runs\detect\val2

(EDU) C:\Users\Admin\OneDrive\Desktop\HackByte_Dataset>
```

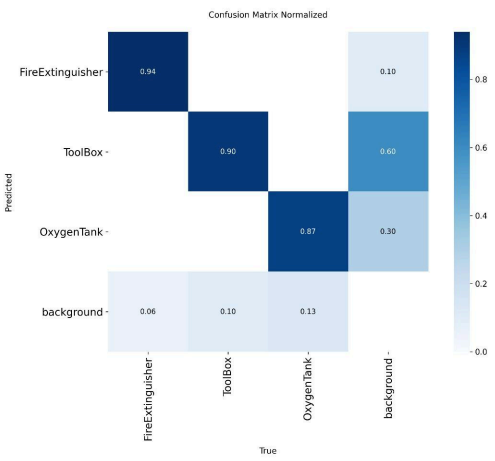
We can see that the model performed well on all the three objects now with **FireExtinguisher** (0.875 mAP@0.5) , **OxygenTank** (0.853 mAP@0.5) and **ToolBox** (0.867 mAP@0.5). Thus we have improved the prediction results by a significant amount of 5%.

RESULTS FOR TRAINING TEST 2 :









Challenges And Solution

FALSE DETECTION OF OBJECTS:

Image 1

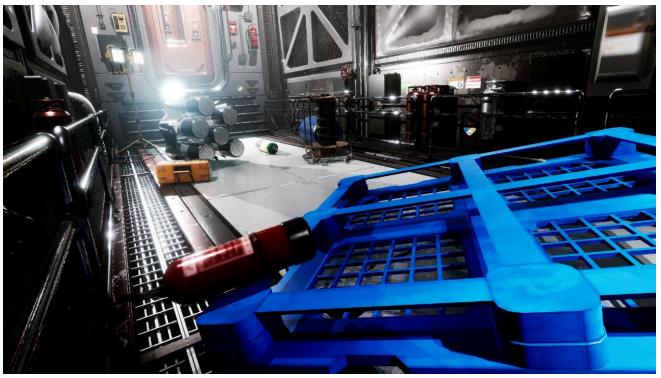


Image 2

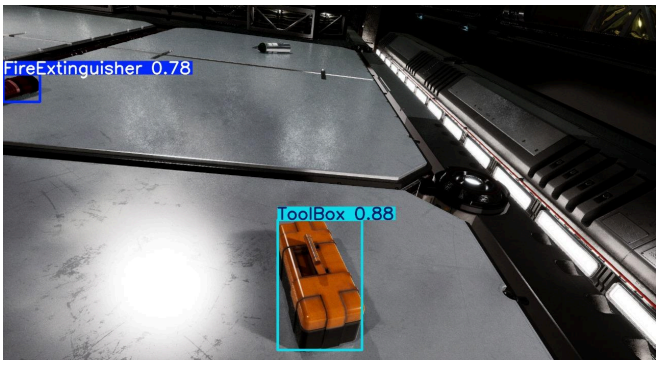


Image 3

Image 4

From the above images we can see that :

- Through **image 1** we can conclude that the object is mostly out of frame, leaving insufficient features for detection and Extreme top-down perspective is not well-represented in training data.
- Through **image 2** we can conclude that the object is significantly hidden by foreground elements and dense background and strong glare and reflections obscure object details.
- Through **image 3** we can conclude that bright reflections reduce object visibility and differentiation from background and the object's lying position might be underrepresented in training.
- Through **image 4** we can conclude that the object is almost entirely concealed by the large blue cylinder along with it only a tiny sliver of the object is visible, making identification nearly impossible.

Solution

To significantly boost our object detection model's accuracy (measured by mAP@0.5), especially for the "OxygenTank" and "Toolbox" classes which were initially harder to detect, we implemented a **targeted data augmentation strategy**. This means we didn't just randomly create more pictures; we generated specific types of new training data where it would have the most impact.

The core of our solution involved creating **two distinct sets of data transformations, or "augmentation pipelines"**:

1. **The "strong" transform:** This pipeline was designed to make the model much more robust to challenging real-world conditions. It included:
    1) **Geometric augmentations:** This involved aggressively rotating images, shifting objects around, scaling them up or down, and even applying perspective changes. The goal here was to teach the model to recognize "OxygenTanks" and "Toolboxes" no matter their **angle, distance, or position** in the scene, even if only a part of them was visible.
    2) **Photometric augmentations:** We introduced significant variations in lighting, contrast, and color. This means generating images where objects appear in very **bright, dim, or oddly lit environments**, or with different color casts. This helps the model identify objects regardless of the specific lighting conditions in the space station.
    3) **Occlusion augmentations (dropout, cutout):** These techniques deliberately hide parts of the objects or introduce random "holes" in the image. This was crucial because "OxygenTanks" were often partially obscured in the original difficult images. By training with these partially hidden examples, the model learns to detect objects even when they are **partially blocked or occluded**.
2. This "strong" pipeline was applied **multiple times** to every original image that contained an "OxygenTank" or "Toolbox." This effectively "oversampled" these underperforming classes, giving the model much more exposure to the difficult scenarios they represent.
3. **The "light" transform:** For the "FireExtinguisher" class, which was already performing well, we used a much milder set of augmentations. This ensured we still added some diversity to its training data (like simple horizontal flips and minor brightness changes) without over-emphasizing a class the model already understood, thus keeping the training focused on the areas that needed the most improvement.

By applying these changes, we were able to **enhance our model's prediction percentage (mAP@0.5) for all classes**.

Conclusion and future work

For future work, we propose exploring:

- **Advanced Augmentation:** Leveraging Falcon to design and generate highly specific, complex synthetic scenes with extreme occlusions and diverse lighting. This pushes the model's robustness to real-world challenges even further.
- **Domain Adaptation:** Developing methods to make the model work effectively in new space station modules or with updated equipment designs with minimal retraining. This ensures the model remains relevant as the environment evolves, saving time and resources.
- **Self-Supervised Learning:** Enabling the model to continuously learn and improve from abundant unlabeled video feeds from the station. This reduces reliance on manual data labeling, allowing for autonomous model updates and adaptation over time.