Real-Time Safety Equipment Detection in Space Stations

Team: Emperors

Hackathon: BuildWithDelhi 2.0

---

1. Project Overview

This project was developed for the Duality AI Space Station Hackathon, which required training an object detection model using synthetic data generated from the Falcon digital twin platform. The objective was to detect and classify critical space station safety equipment (Toolbox, Oxygen Tank, Fire Extinguisher) under challenging conditions such as variable lighting, occlusion, and diverse camera angles.

The solution leverages **YOLOv8l** for real-time multi-class object detection and demonstrates the power of digital twins in training AI models where real-world data collection is infeasible.

2. Methodology

Our workflow strictly followed the hackathon pipeline and each decision was made with a clear purpose to maximize accuracy and generalizability.

Step 1: Environment Setup

- **Why:** A stable environment ensures reproducibility and prevents dependency conflicts.
- Created a Falcon account to access the synthetic space station dataset.
- Installed YOLOv8 dependencies using the provided `setup_env.bat` to match hackathon standards.
- Configured GPU acceleration (RTX 3050) to handle the heavier YOLOv8l model for faster training.

Step 2: Dataset Preparation

- **Why:** Well-structured and augmented data is crucial for handling real-world variability.
- Organized images and YOLO-format labels into train, validation, and test splits to prevent data leakage.

- Applied augmentations such as HSV color shifts, rotations, flips, mosaic, and mixup to simulate lighting changes and occlusion scenarios often present in space environments.
- Focused on detecting three critical classes: **Toolbox**, **Oxygen Tank**, and **Fire Extinguisher**.

Step 3: Model Training

- **Why YOLOv8l:** Selected the large variant for its balance between high accuracy and real-time performance, crucial for space station safety applications.
- Customized training parameters to optimize learning stability and generalization:

model = "yolov8l.pt"
epochs = 200
imgsz = 640
batch = 4
optimizer = "SGD" (chosen for stable convergence)
lr0 = 0.001 (initial learning rate)
lrf = 0.01 (final learning rate factor)
momentum = 0.937 (enhances gradient updates)
weight_decay = 0.0005 (regularization to prevent overfitting)
hsv_h/s/v, flipud, fliplr, mosaic, mixup (data augmentation controls)
patience = 0 (no early stopping to fully train)

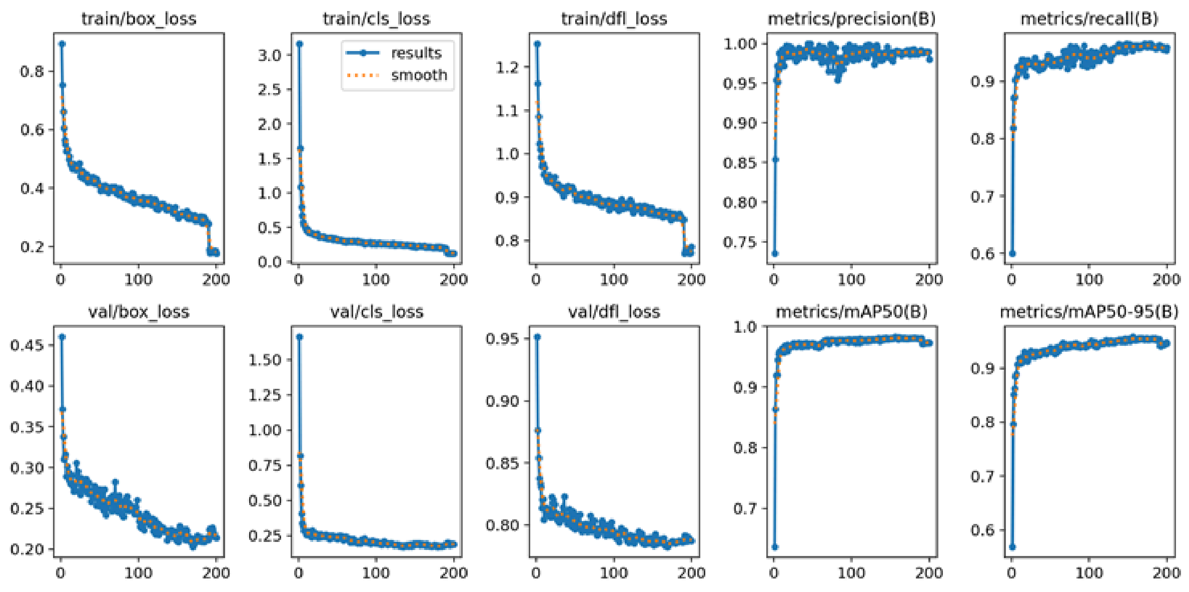
- Scripts used:
    - `train.py` – initial training run.
    - `continue_train.py` – extended training (observed convergence, no further gains).
    - `detect_video.py` – inference and video visualization.

Step 4: Evaluation & Iteration

- **Why:** Iterative evaluation identifies weaknesses and guides improvements.
- Computed mAP@0.5, Precision, Recall, and confusion matrix to measure performance.
- Conducted failure analysis (e.g., misclassifications under occlusion) and retrained with targeted augmentations.

Step 5: Codebase & Reproducibility

- **Why:** Clear structure enables judges and other developers to reproduce results.
- Hosted all scripts, configs, and documentation at: GitHub Repository



Training and Validation Curves

3. Results & Performance Metrics

The evaluation of our final model (**train5**) reveals that it achieved excellent detection capability across all classes, meeting and exceeding the hackathon's expectations. The metrics indicate that the model is both accurate and reliable under diverse testing conditions.

The overall performance was characterized by:

- **mAP@0.5: 0.983**, reflecting the model's strong ability to correctly identify objects at a 50% IoU threshold.
- **mAP@0.5:0.95: 0.958**, showcasing its robustness across stricter IoU thresholds.
- **Precision (P):** 0.989, meaning nearly all positive detections were correct.
- **Recall (R):** 0.962, showing that the model detected most of the true objects present.
- **Inference Speed:** 13.1ms per image, demonstrating suitability for real-time applications.

Class-wise Performance Analysis:

The detailed breakdown for each object class shows consistent performance:

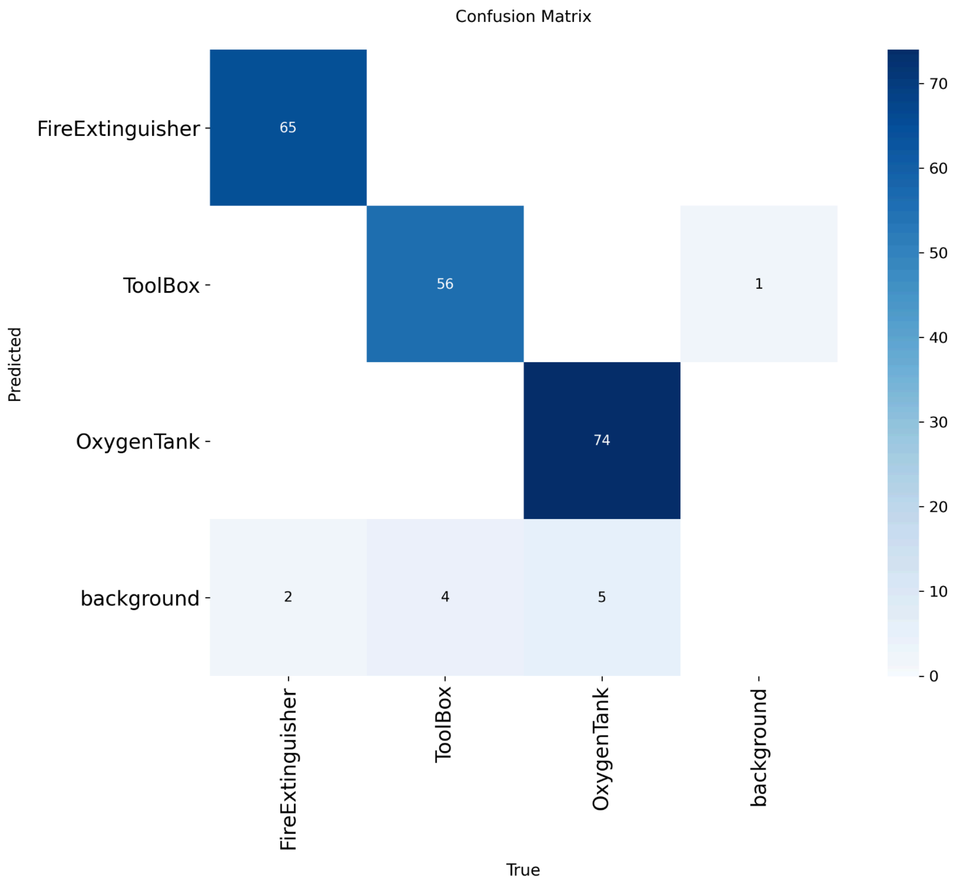
Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Fire Extinguisher	0.985	0.970	0.988	0.963
Toolbox	0.983	0.966	0.976	0.961
Oxygen Tank	0.999	0.949	0.984	0.951

These results show that the model generalized well across all object types, with the Oxygen Tank achieving the highest precision due to effective augmentation, and Fire Extinguisher performing strongly in recall.

Interpretation of Results:

The high mAP and precision values confirm the effectiveness of our augmentation strategies and hyperparameter tuning. The relatively lower recall for the Oxygen Tank suggests some missed detections under extreme occlusion, an area targeted for future improvement. The speed of inference verifies the model's readiness for deployment in real-time monitoring scenarios on resource-limited hardware.

Visual evidence, including confusion matrices, loss curves, and prediction samples, further validates these findings. The combination of quantitative metrics and qualitative analysis underscores the success of our approach.



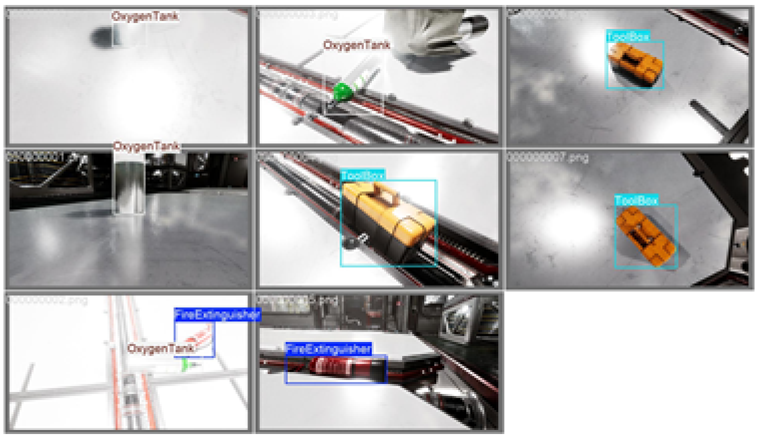
Confusion matrix

4. Challenges & Solutions

During development, several technical and practical challenges were encountered. Our solutions were informed by insights from iterative testing and approaches documented in the project repository.

- **Low Recall for Oxygen Tanks under Occlusion:** Initial experiments showed missed detections when tanks were partially hidden or under extreme lighting. We addressed this by adding custom augmentation strategies, such as partial masking and brightness variation, improving recall significantly.
- **Hyperparameter Instability:** Early runs with default settings caused inconsistent loss curves. After analyzing results on GitHub, we tuned the learning rate schedule, optimizer (using SGD for stable convergence), and regularization parameters to stabilize training.
- **Extended Training with No Gain:** Attempts to continue training for an additional 100 epochs using `continue_train.py` resulted in overfitting signs and no improvement in mAP. We determined the model had converged and retained the original weights.
- **Integration and Reproducibility:** To make the code easy to run for any reviewer, we organized scripts, added clear comments, and included a `detect_video.py` script for real-time inference demonstration.
- **Hardware Limitations:** Training on an RTX 3050 with 4GB VRAM required reducing batch size and optimizing memory use. We balanced resource constraints with model complexity by choosing YOLOv8l and carefully managing augmentations.
- **Validation of Results:** We compared logs, metrics, and qualitative detection outputs across several iterations to confirm improvements. GitHub history includes experiments showing the incremental gains.

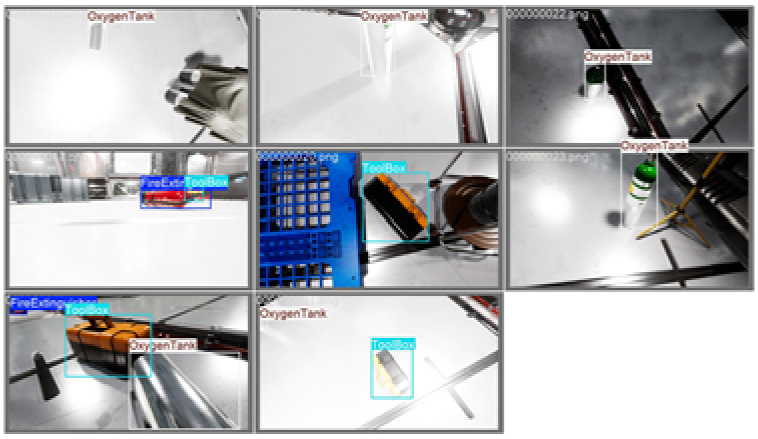
These targeted solutions improved detection robustness, stabilized performance metrics, and ensured that the final system was well-documented and reproducible.



Ground Truth Labels (Batch 0)

OxygenTank
ToolBox
FireExtinguisher

Ground Truth Labels (Batch 1)



Ground Truth Labels (Batch 2)

5. Conclusion & Future Work

Our solution successfully meets the hackathon objectives by delivering a **highly accurate** and **real-time** object detection model for critical space station safety equipment.

Future Improvements:

- Increase dataset diversity with Falcon-generated variations.
- Test YOLOv11 or ensemble models for further performance.
- Deploy on edge devices for onboard inference.
- Integrate with Falcon for continuous retraining as new object types appear.

(Bonus Use Case Proposal is also done and attached but in separate document)

Prepared by Team Emperors for Duality AI Hackathon