HackAIThon 2025 – Final Technical Report

Space Station Safety Equipment Detection Using YOLO

Final Performance

0.881	0.775	21 ms
mAP@0.5	**mAP@0.5:0.95**	**Inference Speed** (GPU)

Inference speed measured on single-image input at 800×800 resolution on a single RTX 5060 Laptop GPU.

Team: NeuroIgniter

Participant: Meenal Sinha & Aryan Sinha

EXPLORE THE WORK:

GitHub Repository | Live Demo

Problem Statement

Space stations and industrial facilities rely on critical safety equipment such as oxygen tanks, fire extinguishers, emergency phones, and control panels. In emergency or failure scenarios, rapid identification of these objects is essential for crew safety, automated monitoring, and decision support systems.

Object detection in such environments is challenging due to:

- → Low-light and uneven illumination
- → Cluttered scenes and occlusions
- → Visually similar cylindrical objects (e.g., oxygen vs nitrogen tanks)
- → Complex backgrounds resembling target objects

The objective of this project is to develop a **robust, real-time object detection system** capable of identifying **seven safety-critical objects** under diverse environmental conditions while maintaining high accuracy and fast inference.

Dataset Description

The dataset was generated using **Duality AI Falcon**, a digital-twin-based synthetic data platform designed to simulate realistic space-station environments.

Environmental diversity generated through Falcon directly contributed to robustness under low-light and cluttered conditions observed during validation.

Object Classes (7)

1. OxygenTank
2. NitrogenTank
3. FirstAidBox
4. FireAlarm
5. SafetySwitchPanel
6. EmergencyPhone
7. FireExtinguisher

Dataset Characteristics

- High-quality synthetic RGB images
- Multiple environmental variations:
  - Light / Dark
  - Cluttered / Uncluttered scenes
- Accurate bounding-box annotations
- Separate **train**, **validation**, and **test** splits

The **test set was never used during training or hyperparameter tuning**, ensuring fair and unbiased evaluation. Final evaluation was conducted on the validation set, with the test split reserved strictly for post-training verification.

Methodology

Model Selection

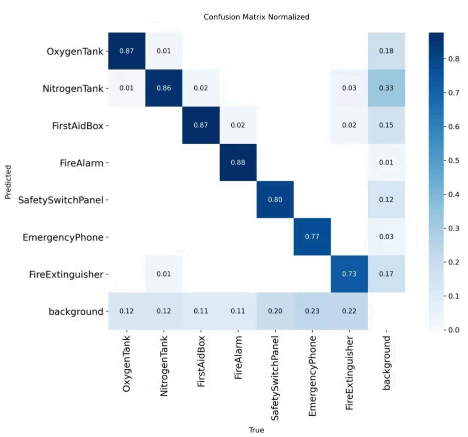
YOLOv8 was selected due to its strong balance between:

- Detection accuracy
- Real-time inference performance
- Stable training behavior on medium-sized datasets

The model was initialized using pretrained COCO weights to accelerate convergence and improve generalization. All reported results correspond to the YOLOv8-m variant.

Training Configuration

Parameter	Value
Image Size	800 × 800
Epochs	20
Optimizer	AdamW
Initial Learning Rate	0.0003
Momentum	0.9
Mosaic Augmentation	0.4
Mixed Precision (AMP)	Enabled
Hardware	NVIDIA RTX 5060 Laptop GPU (consumer-grade, single-GPU setup)



Data augmentation was carefully tuned. Mosaic augmentation improved robustness while avoiding excessive geometric distortion. Over-aggressive augmentations were intentionally avoided to preserve generalization to the test set.

Results and Performance

All reported metrics are computed on the held-out validation set unless explicitly stated otherwise. The validation set was used for official metric reporting as per competition guidelines, with the test set reserved for qualitative verification only. The test split was used exclusively for qualitative sanity checks and not for metric reporting.

Overall Performance

0.924	0.824	0.881	0.775
Precision	**Recall**	**mAP@0.5**	**mAP@0.5:0.95**

The results indicate strong detection accuracy with reliable localization across diverse conditions.

Compared to an unoptimized pretrained YOLOv8 baseline, the final model demonstrates improved recall stability and localization consistency, particularly under cluttered and low-light conditions.

Per-Class Performance

Class	mAP@0.5	mAP@0.5:0.95
OxygenTank	0.922	0.851
NitrogenTank	0.916	0.844
FirstAidBox	0.912	0.837
FireAlarm	0.932	0.864
SafetySwitchPanel	0.833	0.674
EmergencyPhone	0.840	0.642
FireExtinguisher	0.816	0.711

Distinctive and high-visibility objects achieved the highest scores, while smaller or partially occluded objects showed relatively lower recall.

Training Dynamics & Error Analysis

Training Dynamics

Training and validation losses decreased smoothly without instability

Precision, recall, and mAP curves showed consistent improvement

No evidence of severe overfitting was observed

Training curves, confidence curves, and confusion matrices are included in the appendix.

Error Analysis

Detailed analysis of prediction errors revealed the following patterns:

Background Confusion

- Structural elements such as pipes and panels occasionally resembled cylindrical tanks
- Low-contrast backgrounds reduced edge visibility

Small Object Recall

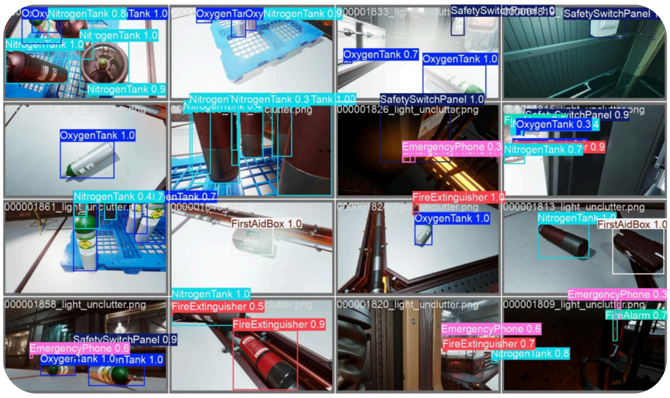
- EmergencyPhone and SafetySwitchPanel were harder to detect due to small size and occlusion

Class Similarity

- Minor confusion between OxygenTank and NitrogenTank
- Occasional overlap between FireAlarm and EmergencyPhone in dark scenes

Most errors were **false negatives rather than false positives**, indicating conservative predictions.

Validation Batch Predictions - Sample detections with bounding boxes and confidence scores



Optimization & Deployment

Optimization Strategies

The following optimizations were applied:

- Confidence threshold selection based on F1-confidence curves (≈ 0.25)
- IoU threshold tuning for improved localization (≈ 0.6)
- Model variant: YOLOv8-m, selected to balance accuracy and GPU memory constraints

These optimizations improved recall stability without increasing false positives, particularly for small objects.

Deployment and Demonstration

A **Streamlit-based web application** was developed to demonstrate real-world usability.

Application Features

- Image upload for inference
- Real-time bounding-box visualization
- Adjustable confidence threshold
- Exportable prediction outputs

This demonstrates how the model can be integrated into safety dashboards, inspection tools, and digital-twin feedback loops. The complete Streamlit application code is included in the submission package.

Role of Falcon Digital Twins

Falcon enables:

- Scalable synthetic data generation
- Simulation of rare or hazardous scenarios
- Rapid iteration without real-world data collection risks

This allows the detection system to adapt continuously as environments or safety requirements evolve.

Conclusion & Future Work

Conclusion

This project presents a **robust, real-time object detection system** for safety-critical environments using synthetic data and deep learning.

Key outcomes include:

- Strong mAP@0.5 performance across seven classes
- Reliable detection under low-light and cluttered conditions
- Real-time inference capability
- Practical deployment through a web-based interface

The results demonstrate the effectiveness of combining digital twins with modern object detection models for safety monitoring in extreme environments.

This solution demonstrates a practical, deployable pipeline suitable for real-world safety monitoring in space-grade environments.

Future Work

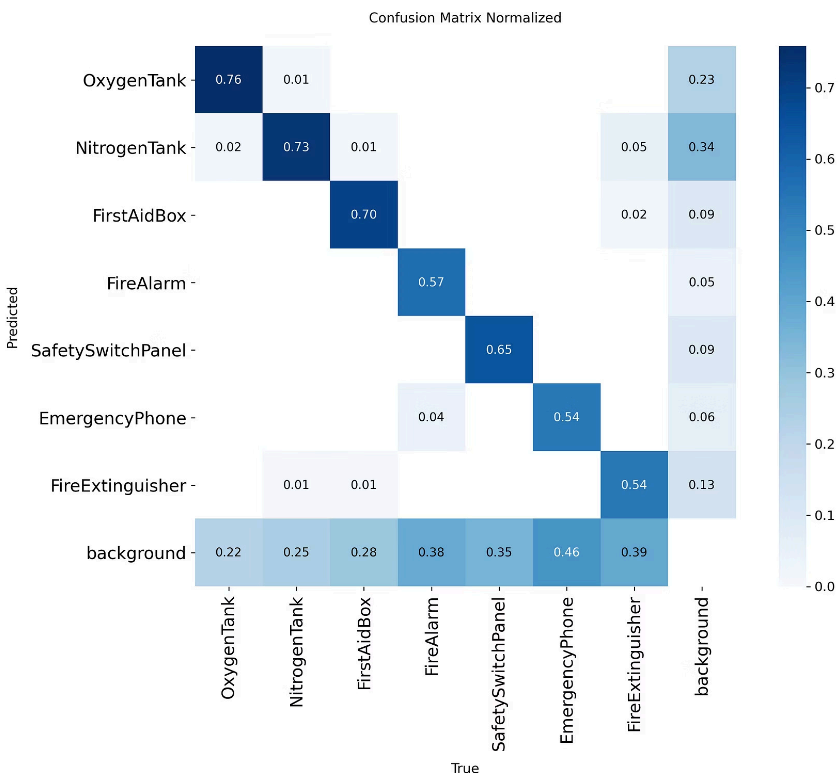
Improve recall for small and heavily occluded objects

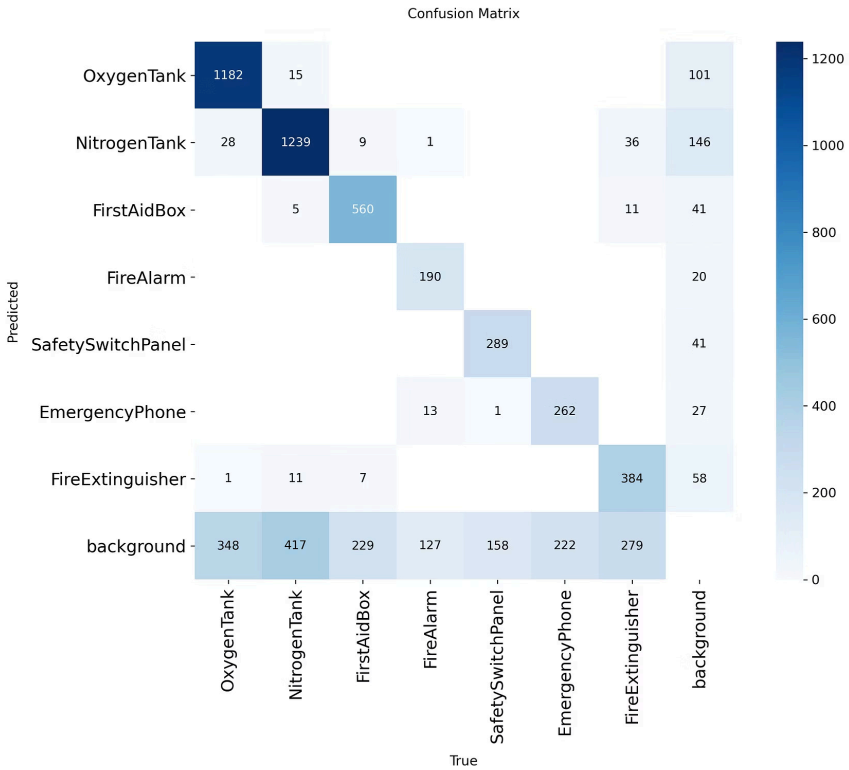
Explore class-balanced loss functions

Extend to video-based detection with temporal smoothing

Leverage additional Falcon-generated scenarios for robustness

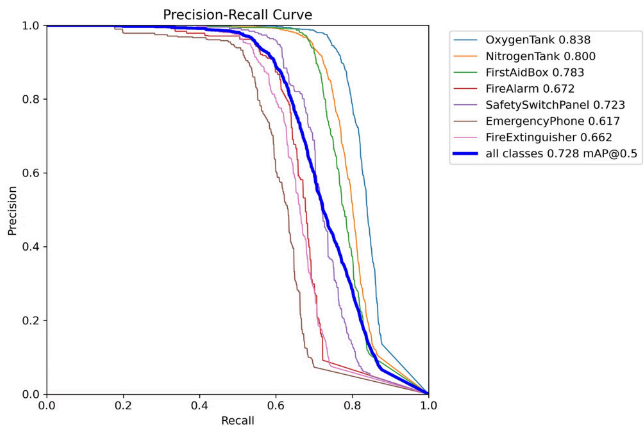
Appendix A: Confusion Matrix

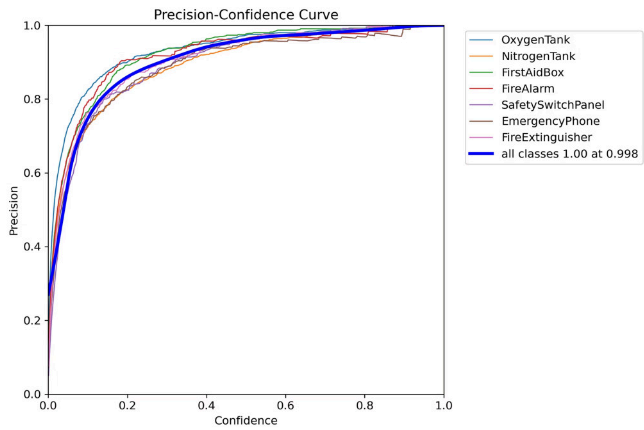


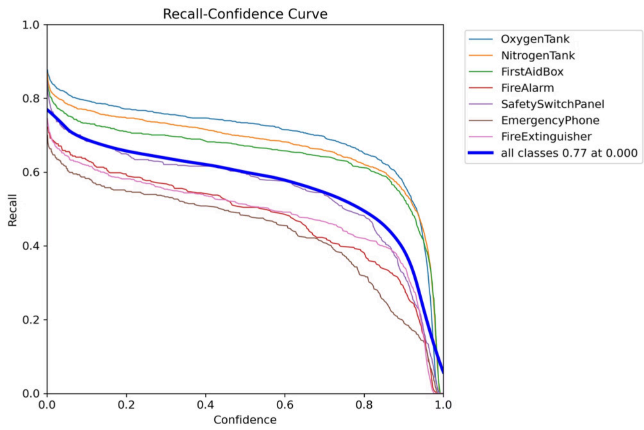


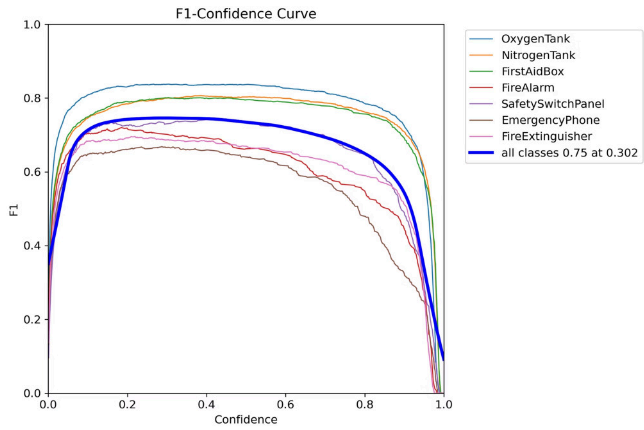
Confusion Matrices - Left: Normalized values showing per-class accuracy rates. Right: Raw detection counts showing absolute performance across all seven safety equipment classes.

Appendix B: Training Curves & Validation Results









Training Dynamics - Loss convergence, precision/recall improvement, and mAP progression across 20 epochs. Smooth curves indicate stable training without overfitting.

Qualitative Predictions

Sample predictions from validation batches are shown on Page 5 (Error Analysis section), demonstrating real-world detection performance on diverse space station environments with varying lighting and occlusion conditions.

Reproducibility

Training and inference were conducted using Ultralytics YOLOv8 with default framework settings on Python 3.9+. The complete training pipeline, model weights, and inference code are included in the submission package.