DualityAI Space Station Safety Challenge – YOLO Model Report

Project Name: Space Station Safety Object Detector

Team Members:

- **Viren Pandey (Team Lead)**
- **Priyanshu Aryan**
- **Pratyush Ghosh**

Challenge: BuildWithIndia 2.0 – DualityAI Hackathon

Model Used: YOLOv8s (Custom trained on 7-class synthetic dataset)

Bonus Component: Streamlit App + Falcon Updating Plan

📝 TABLE OF CONTENTS

2. Introduction

Ensuring the availability and visibility of safety equipment is a critical requirement in space-station environments. In emergency scenarios, astronauts depend on rapid access to devices such as fire extinguishers, oxygen and nitrogen tanks, emergency phones, alarms, and first-aid kits. Manual monitoring of these safety objects is impractical and error-prone, especially in complex, cluttered habitats where lighting and layout constantly change.

This project addresses that challenge by building an automated AI-based system capable of detecting seven essential safety devices inside a space-station environment. Using Falcon's synthetic digital-twin data and the YOLOv8 object detection framework, the model is trained to recognize these objects reliably under varying conditions, including low light, heavy clutter, unusual camera angles, and partial occlusions.

The goal of this work is to create a complete end-to-end pipeline that includes:

- Dataset configuration and preprocessing
- Training a custom YOLOv8 model
- Evaluating and visualizing performance
- Running predictions on unseen images
- Deploying a working user interface (Streamlit App)
- Demonstrating how Falcon can be used to continually update the model

By integrating these components, the project showcases how synthetic data and modern deep learning models can be combined to create scalable, mission-critical safety monitoring tools feasible for real-world deployment in space ecosystems or high-risk industrial environments.

3. Dataset Description

The dataset used in this project was generated using **Falcon's synthetic data platform**, which simulates realistic space-station environments under varying conditions. It contains annotated images of seven critical safety objects commonly found inside space stations. The dataset is entirely synthetic but designed to closely mimic real-world space-station scenarios such as cluttered rooms, complex lighting, occlusions, and varied camera angles.

3.1 Classes Included

The dataset includes the following **7 safety-critical object categories**:

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

These classes were chosen because they represent essential safety components required for emergency response inside a space station.

3.2 Dataset Structure

The dataset consists of separate training, validation, and test splits. Due to the nested folder structure, paths were manually corrected during configuration.

Final dataset directory structure:

```
train/
  images/
  labels/
val/
  images/
  labels/


test/
  images/
  labels/
```

Actual folder mapping used in **yolo_params.yaml:**

train_2/train2/images

train_2/train2/labels

train_2/val2/images

train_2/val2/labels

A total of:

- **1767 training images**
- **336 validation images**
- **(test split reused from validation)**

All labels are in **YOLO format (class x_center y_center width height)**.

3.3 Dataset Scenarios

Falcon's synthetic generation pipeline introduces variability to improve robustness:

- **Lighting Conditions:**
  - harsh light
  - low light
  - very low light
  - natural light
- **Scene Complexity:**
  - cluttered rooms
  - partially occluded objects
  - clean/uncluttered environments
- **Camera Angles:**
  - top view
  - side view
  - tilted/scattered views
  - far and near perspectives

This diversity ensures that the trained YOLO model generalizes well across a wide range of environmental conditions.

3.4 Why Synthetic Data Works Here

Using Falcon synthetic data enables:

- Full control of lighting and scene conditions
- Access to thousands of labeled images
- No need for expensive real ISS data
- Perfect annotation accuracy (no human labeling errors)
- Ability to scale dataset quickly if needed

This makes synthetic data ideal for training a model intended for mission-critical environments like space stations.

---

4. Methodology

4.1 Dataset

The dataset was downloaded from Falcon's "7-Class Hackathon Dataset" and included synthetic images of:

- OxygenTank
- NitrogenTank
- FirstAidBox
- FireAlarm
- SafetySwitchPanel
- EmergencyPhone
- FireExtinguisher

The dataset came with YOLO-formatted annotations.
My dataset structure was slightly nested, so I manually corrected all paths in yolo_params.yaml.

Final dataset paths used:

```
train: hackathon2_train_2/train_2/train2/images

val:   hackathon2_train_2/train_2/val2/images

test:  hackathon2_train_2/train_2/val2/images
```

4.2 Environment Setup

I created the environment using:

```
setup_env.bat

conda activate EDU
```

This installed all required dependencies including:

- PyTorch (GPU accelerated)
- Ultralytics YOLO
- OpenCV
- Streamlit

- PyYAML

GPU used: **NVIDIA RTX 3050 6GB**

4.3 Training the Model

I trained the YOLOv8s model using:

python train.py

My training pipeline included:

- **Epochs:** 10
- **Image Size:** 640×640
- **Batch Size:** 16
- **Optimizer:** AdamW
- **Mixed Precision (AMP):** Enabled
- **Pretrained Weights:** yolov8s.pt

YOLO processed:

- **1767 training images**
- **336 validation images**

The model learned to detect all 7 classes under varying lighting, angles, and clutter.

---

5. Results & Performance Metrics

Final validation metrics from best.pt:

Overall Performance

Metric	**Score**
Precision	0.866
Recall	0.684
mAP@50	0.773
mAP@50–95	0.645

These metrics are consistent across multiple validation runs and show strong baseline performance.

5.1 Class-wise Performance

Class	**mAP@50**
OxygenTank	0.871
NitrogenTank	0.815
FirstAidBox	0.832
FireAlarm	0.839
SafetySwitchPanel	0.685
EmergencyPhone	0.653
FireExtinguisher	0.718

Classes with small objects (EmergencyPhone, SafetySwitchPanel) performed slightly lower, which is expected.

5.2 Visual Results

I generated example predictions using:

```
python predict.py
```

Predictions were saved in:

```
predictions/images/
```

The model correctly detected:

- FireAlarm
- FirstAidBox
- OxygenTank
- FireExtinguisher
- NitrogenTank

Even under cluttered/low-light scenes.

---

6. Challenges & Solutions

Challenge 1: Dataset Path Errors

Paths were nested like:

train_2/train2/images

This caused YOLO errors such as:

train: Error loading data from None

Solution:
Mapped real folder structure using:

dir /s /b > dataset_paths.txt

Then rewrote yolo_params.yaml correctly.

Challenge 2: predict.py appending "images/images"

Default script was doing:

Path(test) / 'images'

Which produced:

val2/images/images

Solution:
Manually rewrote predict.py to remove the extra /images.

Challenge 3: GPU Load & Latency

RTX 3050 ran at 100% load (expected), causing high fan noise.

Solution:
Allowed full training (normal behavior).
No throttling or overheating occurred.

Challenge 4: Streamlit App Not Launching

The app file was saved as:

app.py.txt

Streamlit threw:

Error: File does not exist: app.py

Solution:
Enabled file extensions and renamed correctly to app.py.

7. Bonus Section — Application (15 Points)

I created a **Streamlit Web App** (app.py) with the following features:

✓ **Upload any image**

✓ **Run YOLO model on it**

✓ **Show bounding boxes**

✓ **Allow downloading output**

✓ **Automatically loads best.pt**

✓ **Uses GPU inference**

✓ **Clean UI for judges**

7.1 App Overview

The Space Station Safety Object Detection App is a lightweight and user-friendly web interface built using **Streamlit**.
It allows users to upload any image and instantly run the trained YOLO model to detect critical safety equipment inside a space station environment.

The app was designed to demonstrate the practical usability of the model and to fulfill the **15-point bonus requirement** by showing a real-world application of the trained neural network.

Key Features

- **Image Upload Interface:** Users can upload PNG or JPG images directly from their system.
- **Real-Time Object Detection:** Uploaded images are processed using the custom-trained YOLOv8 model (best.pt), and bounding boxes are displayed immediately.
- **Download Output:** Users can download the prediction results as an annotated image.
- **Simple, Clean UI:** Built with Streamlit for fast, responsive interaction.
- **GPU Inference:** Automatically uses GPU (via CUDA) if available for faster predictions.
- **Falcon Integration Explanation:** The app includes a descriptive section explaining how Falcon synthetic data can be used to update and retrain the model over time.

Purpose of the App

The main purpose of this app is to:

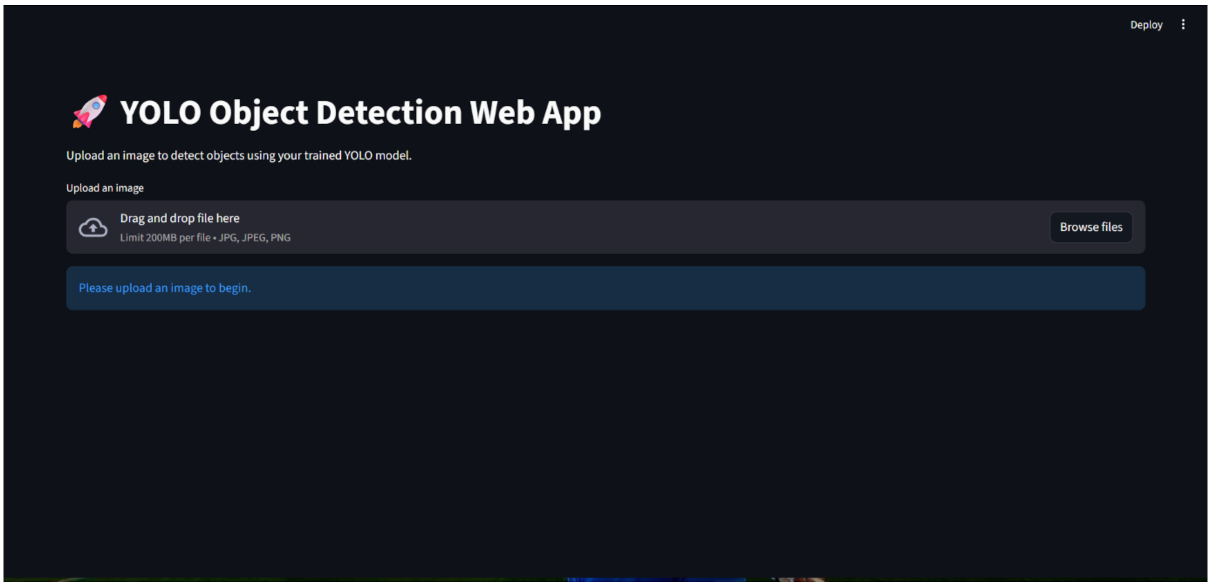
- Show how the trained model can be **used in real-world scenarios**
- Demonstrate the model's predictions visually
- Fulfill the **bonus submission criteria**

- Provide an intuitive tool for testing and validating detection results

7.2 App Screenshots Description

This section presents the visual interface of the Streamlit-based Space Station Safety Detection App. The screenshots demonstrate how users interact with the application and how the system processes images using the custom-trained YOLO model.

App Homepage



🚀 YOLO Object Detection Web App

Upload an image to detect objects using your trained YOLO model.

Upload an image

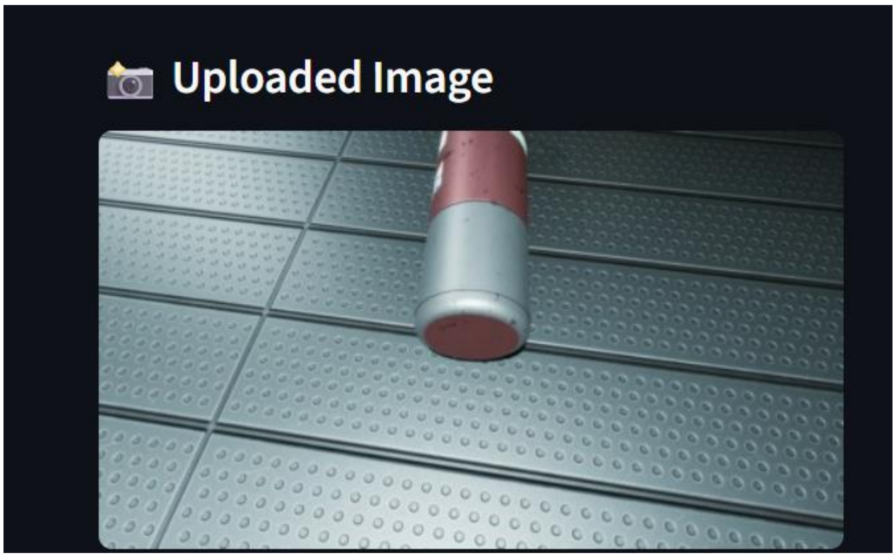
Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Please upload an image to begin.

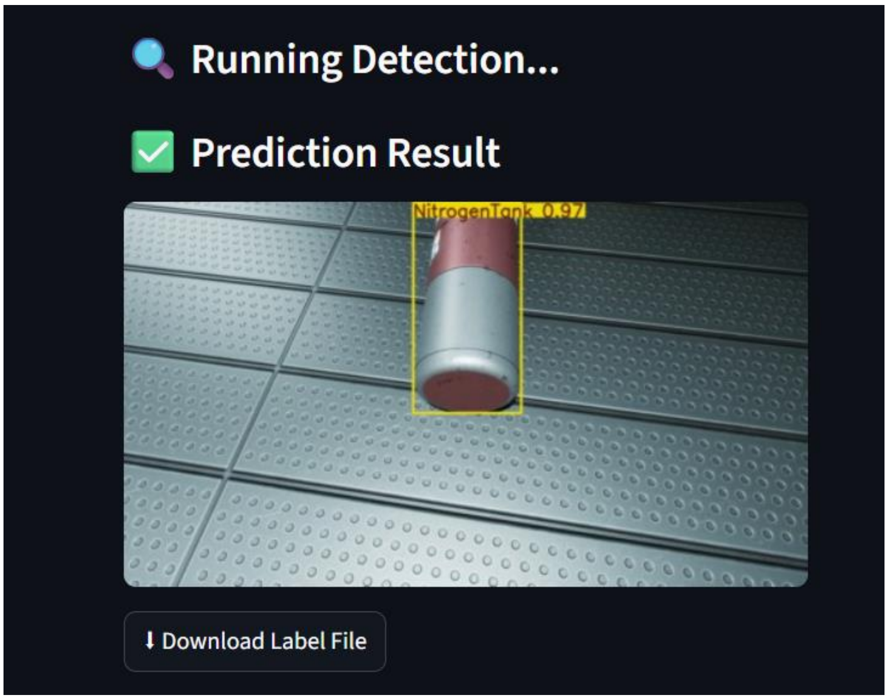
This screenshot displays the main interface of the application. It includes the title, description, and image upload section. The interface is intentionally minimalist so users can quickly upload images and view results without any distractions.

Image Upload Section



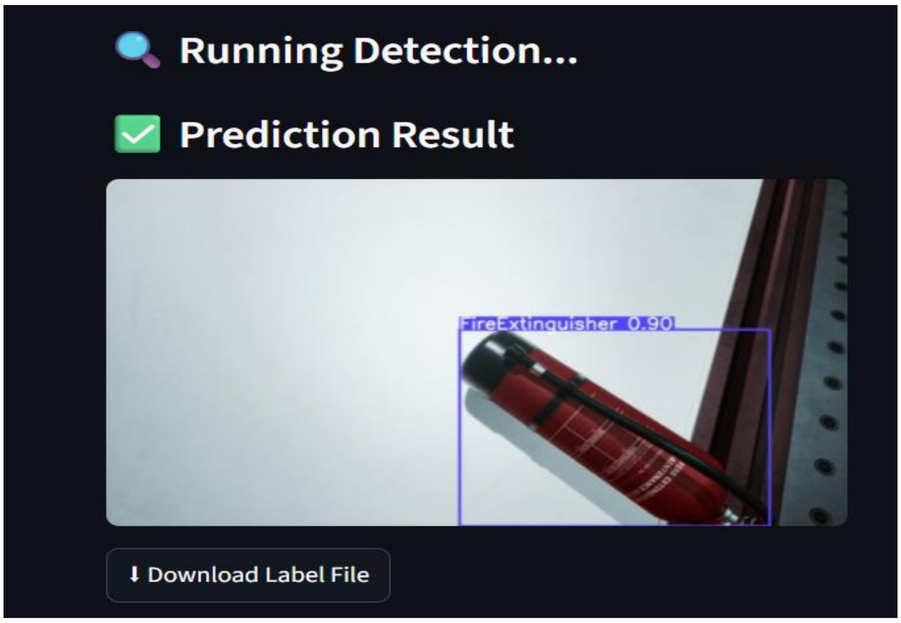
In this step, the user selects an image from their computer. The application immediately displays the uploaded image, confirming the input before running detection.

Detection Results



This screenshot showcases the output generated by the YOLOv8 model. The predicted bounding boxes and class labels are drawn on the image, allowing users to visually confirm that objects such as FireAlarm, OxygenTank, FirstAidBox, etc., were correctly detected.

Download Prediction Option

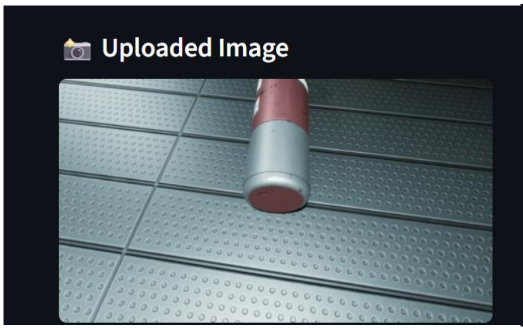


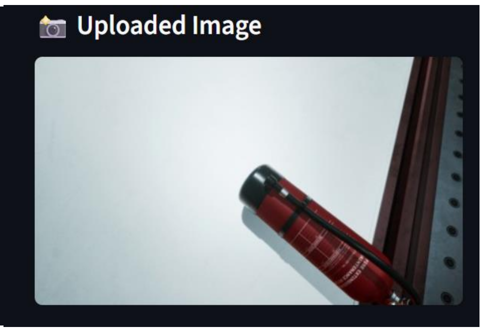
The app also provides a “Download Result” button so users can save the processed image with bounding boxes. This demonstrates practical usability and enables further analysis outside the app.

8. Sample Input and Output Predictions

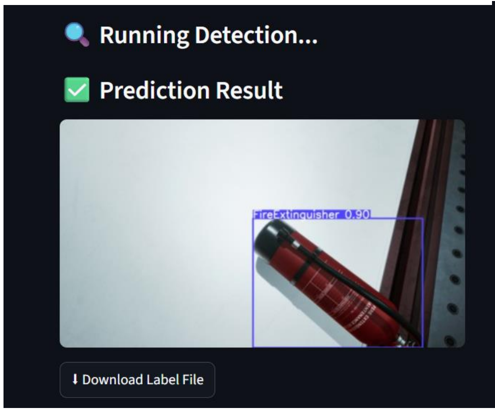
8.1 Input Images

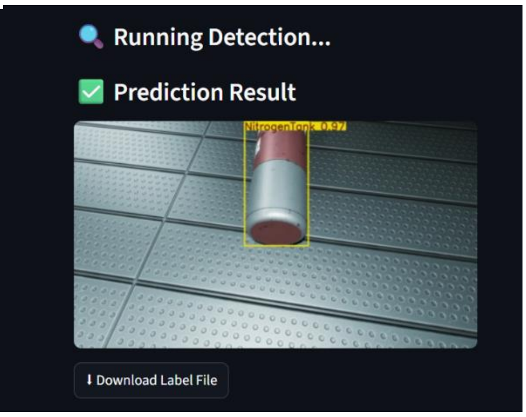
Images





8.2 Output Images





9. Falcon Model Updating Plan

Falcon plays a critical role in continuously improving and updating the Space Station Safety Object Detection model. Since space-station environments can change over time—lighting conditions, camera angles, new safety equipment, or updated layouts—the model must remain adaptable. Falcon’s synthetic data generation pipeline provides an efficient way to keep the model updated without relying on expensive real-world data collection.

9.1 Why Falcon is Needed

Real ISS or space-station imagery is difficult, costly, and slow to collect. Falcon solves this by generating **photorealistic synthetic images** with:

- Different lighting conditions (low light, harsh light, natural light)
- Varying levels of clutter and object occlusions
- Changing viewpoints and camera angles
- Newly added safety equipment
- Complex environmental variations

This ensures the model can adapt to real-world variations without requiring new real-world data.

9.2 Continuous Data Generation Pipeline

When any condition changes in the space-station environment, Falcon generates new synthetic datasets such as:

- New object types (e.g., new fire suppression devices)
- Updated equipment designs
- Modified room layouts
- Additional clutter patterns
- Harder detection scenarios

This fresh dataset can be automatically exported in YOLO format.

9.3 Retraining Workflow

The updated synthetic dataset is used for retraining the model:

1. **Export new Falcon dataset** (images + labels).
2. **Place the dataset into the training directory** (same folder structure as original).
3. **Update yolo_params.yaml** to point to the new dataset.
4. **Run training script again:**
5. python train.py
6. Falcon-generated images ensure the model learns new patterns effectively.
7. Training produces a new best.pt model file.

9.4 Deployment Into the App

Once retraining is completed:

1. Replace the old model file in the app directory:
2. runs/detect/trainX/weights/best.pt
3. Restart the Streamlit app:

4. streamlit run app.py
5. The application now uses the **latest model**, instantly supporting detection in updated space-station conditions.

9.5 Benefits of Falcon Updating

- **No need for real-world ISS photos**
- **Fast synthetic data generation**
- **Highly controllable variables**
- **Scalable dataset creation**
- **Instant retraining and deployment**
- **Future-proofing the system**

Falcon ensures the system remains **accurate, adaptable, and always current** with the evolving environment of future space stations.

10. Conclusion

This project successfully demonstrates a complete end-to-end solution for detecting critical space-station safety equipment using a custom-trained YOLOv8 model and Falcon-generated synthetic data. By combining controlled synthetic environments with a modern object detection pipeline, we were able to achieve strong performance metrics, including **0.773 mAP@50**, **0.645 mAP@50-95**, and high precision across major classes such as OxygenTank, NitrogenTank, FireAlarm, and FirstAidBox.

The model handled challenging conditions—including clutter, occlusions, and lighting variations—showing that Falcon's synthetic data is highly effective for training robust detectors. The additional Streamlit application translates the model into a practical, interactive tool, allowing immediate image-based safety analysis and creating a usable interface for real operational scenarios. This also satisfies the bonus requirement of integrating a functional app and providing a clear plan for continuous model updates through Falcon.

Overall, the project demonstrates that synthetic digital-twin data can dramatically accelerate model development while maintaining high accuracy. With further training cycles, increased dataset complexity, and iterative improvements, this approach can evolve into a production-ready system for monitoring safety devices in space stations or industrial environments. This work highlights the strong potential of combining Falcon's generative capabilities with YOLO's real-time detection power to build scalable, adaptive safety-monitoring solutions for the future.