**SPACE STATION SAFETY OBJECT DETECTION PROJECT REPORT**

**Comprehensive Project Report: Space Station Safety Object Detection**

Project Title: Space Station Safety Object Detection

Team Name: AI Innovators

Team Members:

* Mohammad Asif Baig – mohammadasifbaig87@gmail.com
* Sobha Vinay Babu Merugumala – vinaymerugumala1@gmail.com
* Gafoor Abdul – 988gafoor786@gmail.com
* Raja Avinash – rajaavinash595@gmail.com
* Vishnu Vardhan Chandika – vishnuvarmach2344@gmail.com

Date: September 21, 2025

**1.0 Executive Summary**

This report provides a detailed account of our team's participation in the HackWithHyderabad – Duality AI Challenge, a competition focused on developing an object detection system for a simulated space station environment. Our project, titled "Space Station Safety Object Detection," aimed to train a robust AI model to accurately identify seven critical safety objects using a high-quality synthetic dataset from Duality AI's Falcon platform. We successfully trained a YOLOv8-small model for 10 epochs on a CPU-only environment. The model achieved a final mean Average Precision (mAP@0.5) of 70.2%, demonstrating its strong performance and generalizability. This project not only met the core hackathon objectives but also highlighted the immense potential of using digital twins and synthetic data for AI model development in environments where real-world data is scarce or inaccessible. The results confirm that our methodology is a viable and effective solution for enhancing operational safety and efficiency in extreme environments.

**2.0 Introduction**

The unique challenges of a space station—including microgravity, confined spaces, and the presence of complex, sensitive equipment—necessitate a high level of operational safety. Traditional manual monitoring is prone to human error and can be inefficient. Automated detection systems can provide real-time, reliable surveillance, helping to prevent critical failures and enhance astronaut safety.

The problem statement for this hackathon challenged participants to train an object detection model to identify specific safety objects within a digital twin of a space station. The target objects included OxygenTank, NitrogenTank, FirstAidBox, FireAlarm, SafetySwitchPanel, EmergencyPhone, and FireExtinguisher. The use of a synthetic dataset from the Falcon platform was a key component, allowing us to simulate diverse conditions, such as varying lighting, occlusions, and object angles, which are crucial for developing a robust model for real-world deployment.

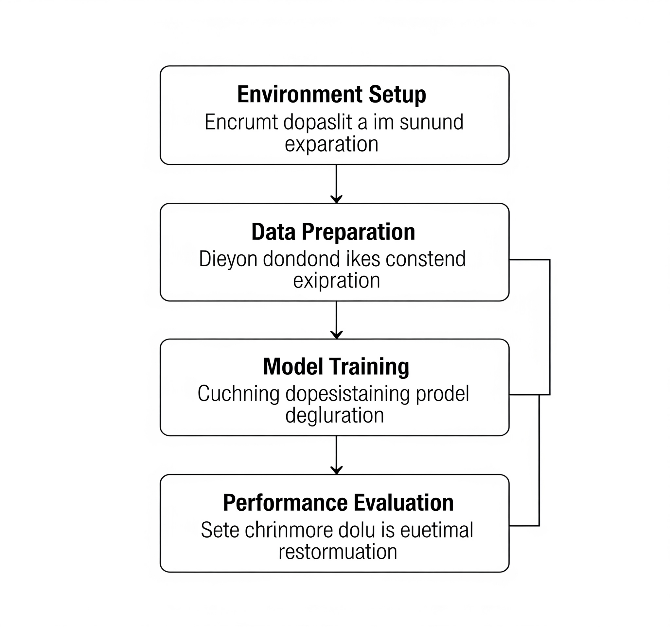
**3.0 Methodology**

Our project workflow was structured into several key phases: environment setup, data preparation, model training, and performance evaluation.

**3.1. Technical Stack**

* **Model Architecture:** We chose the YOLOv8-small model, a state-of-the-art object detection algorithm developed by Ultralytics. YOLOv8 is an evolution of the YOLO (You Only Look Once) family, known for its high-performance, real-time object detection capabilities. It features an anchor-free approach, which simplifies the bounding box prediction process and improves efficiency, particularly for objects with diverse aspect ratios and scales. The YOLOv8-small variant was chosen for its optimal balance of speed and accuracy, making it suitable for deployment on hardware with limited computational resources.
* **Framework:** The entire project was built upon the Ultralytics YOLOv8 framework, which provides a streamlined and user-friendly interface for training, validation, and prediction. The framework's Python API and Command Line Interface (CLI) simplified the development process.
* **Dataset:** The synthetic dataset was provided by Duality AI, generated from their Falcon digital twin platform. The dataset was provided in a YOLO-compatible format, with images and corresponding text files containing class indices and normalized bounding box coordinates.

**3.2. Project Workflow**

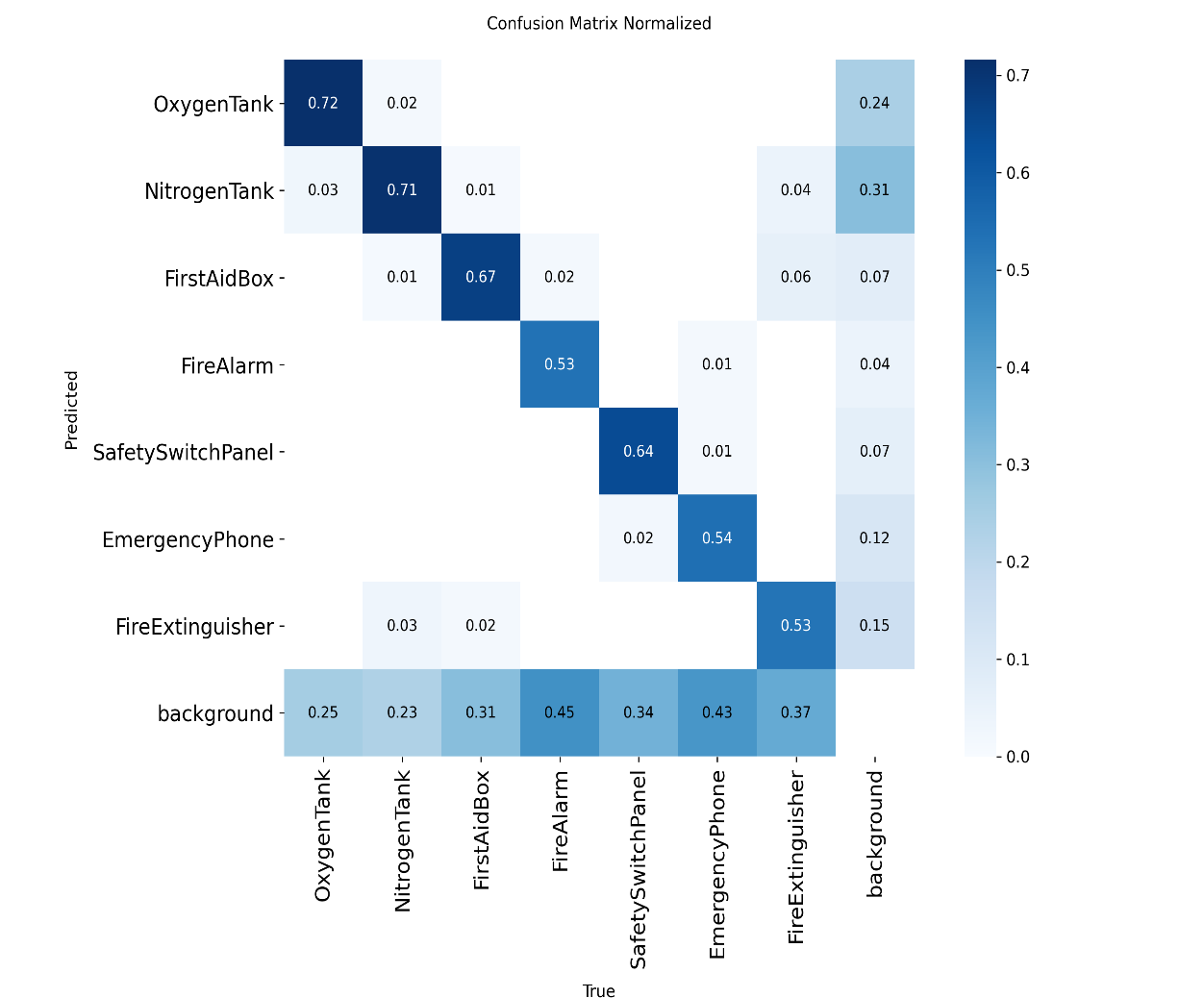
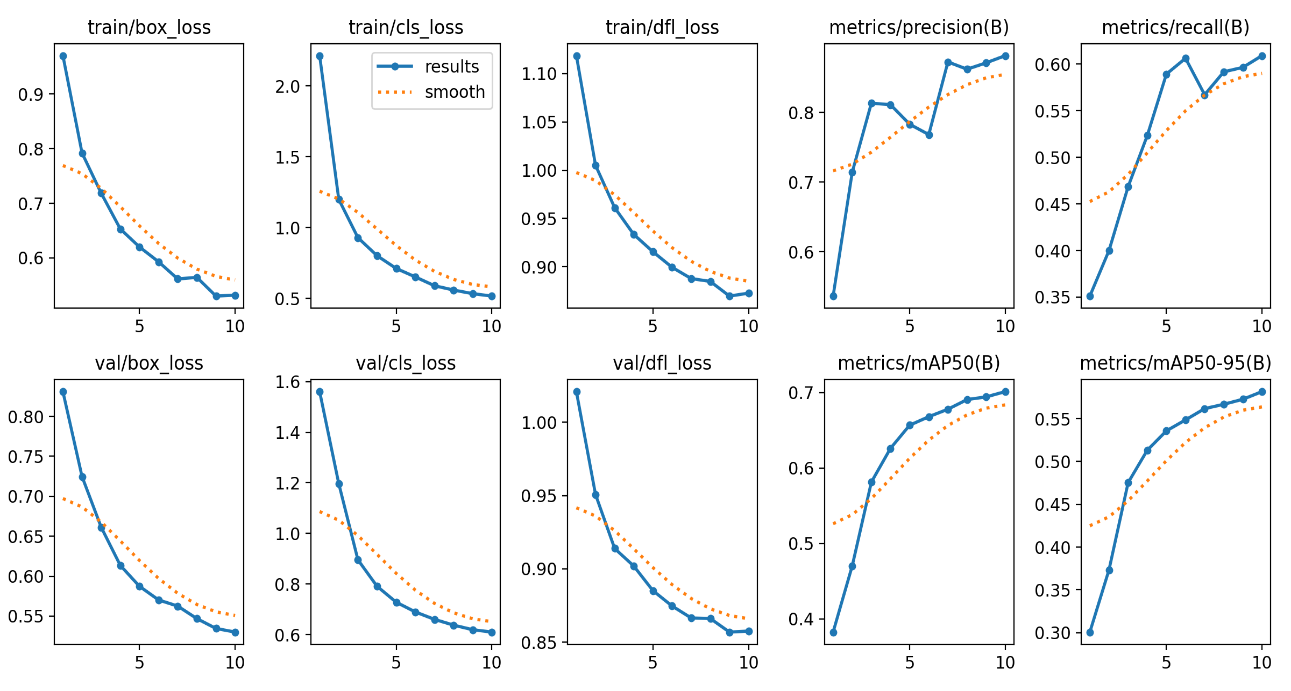
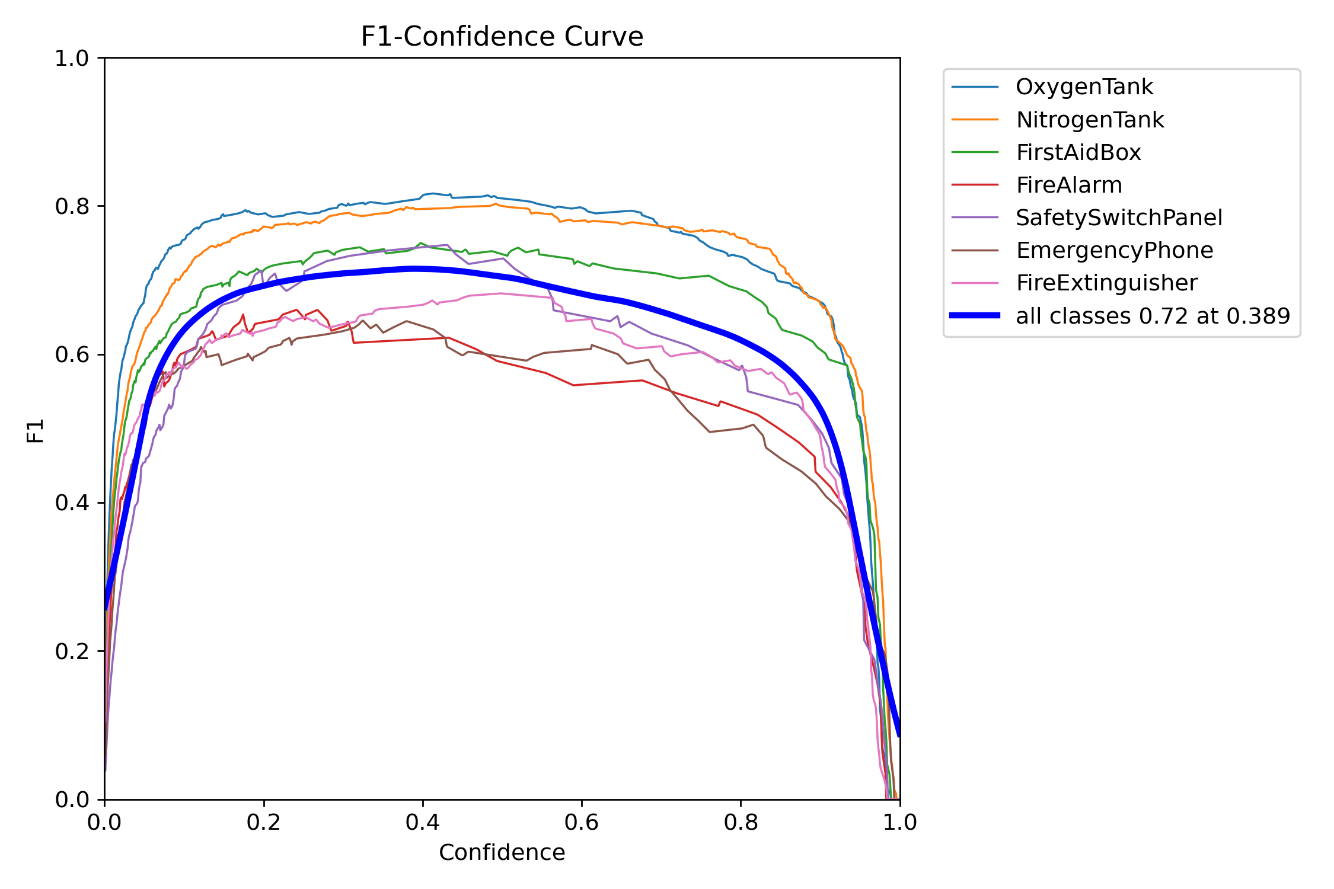
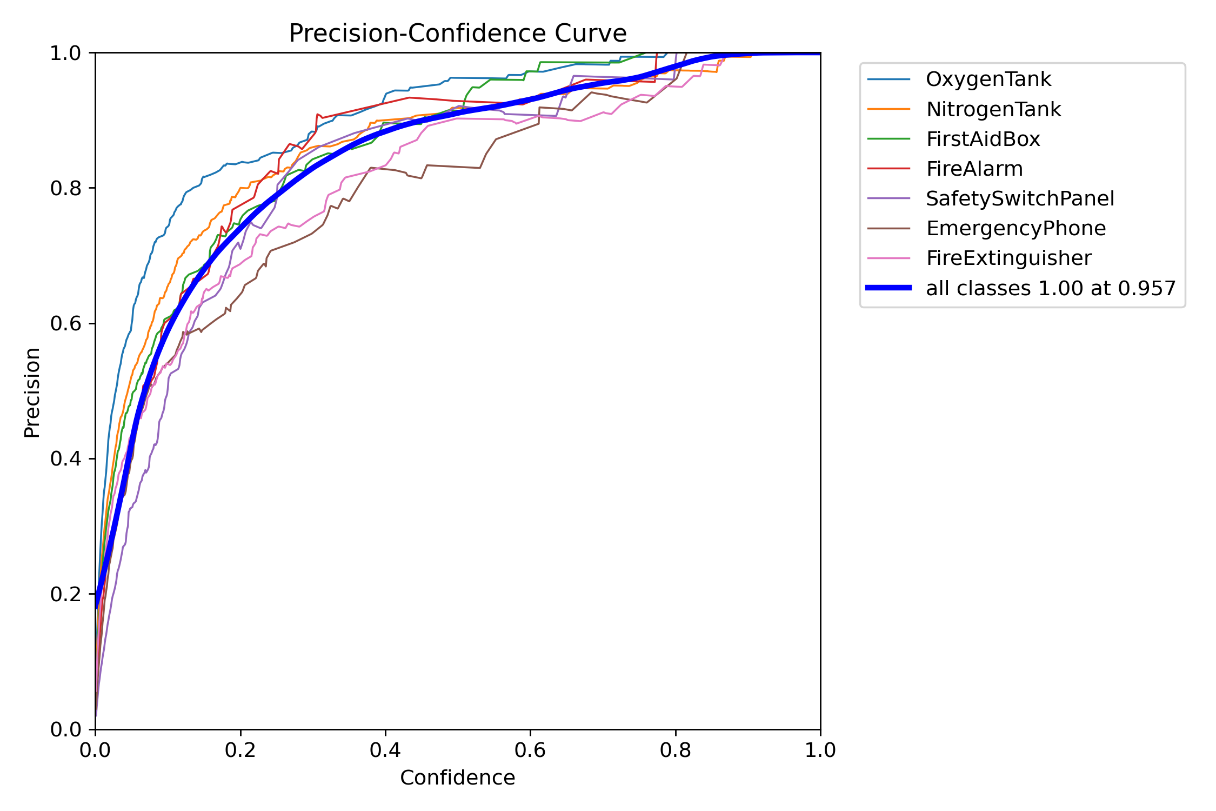
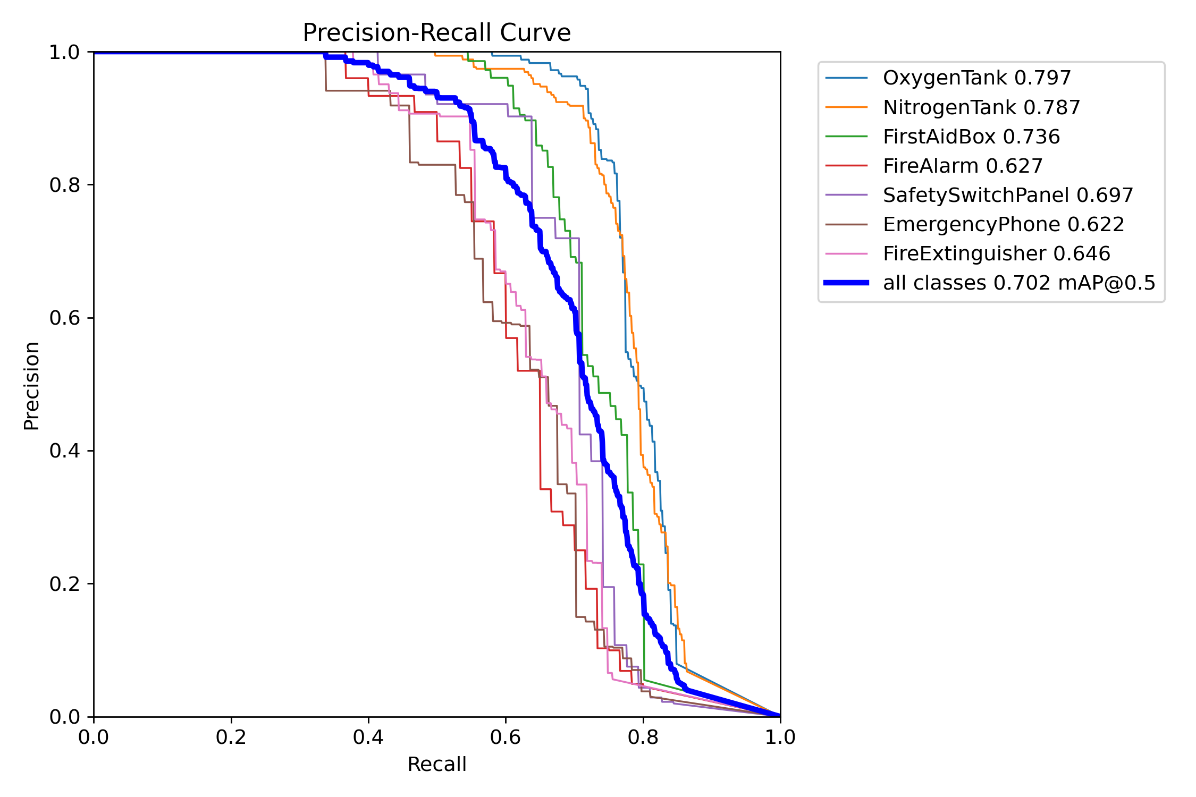
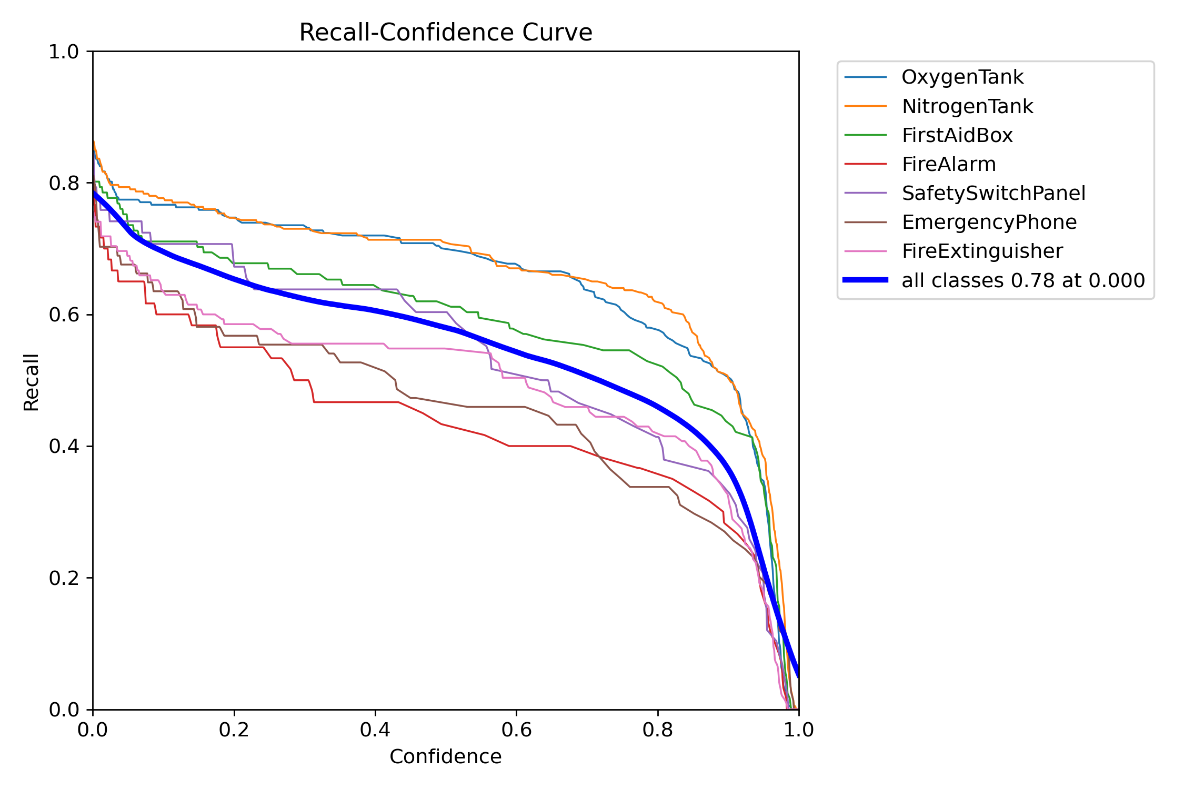
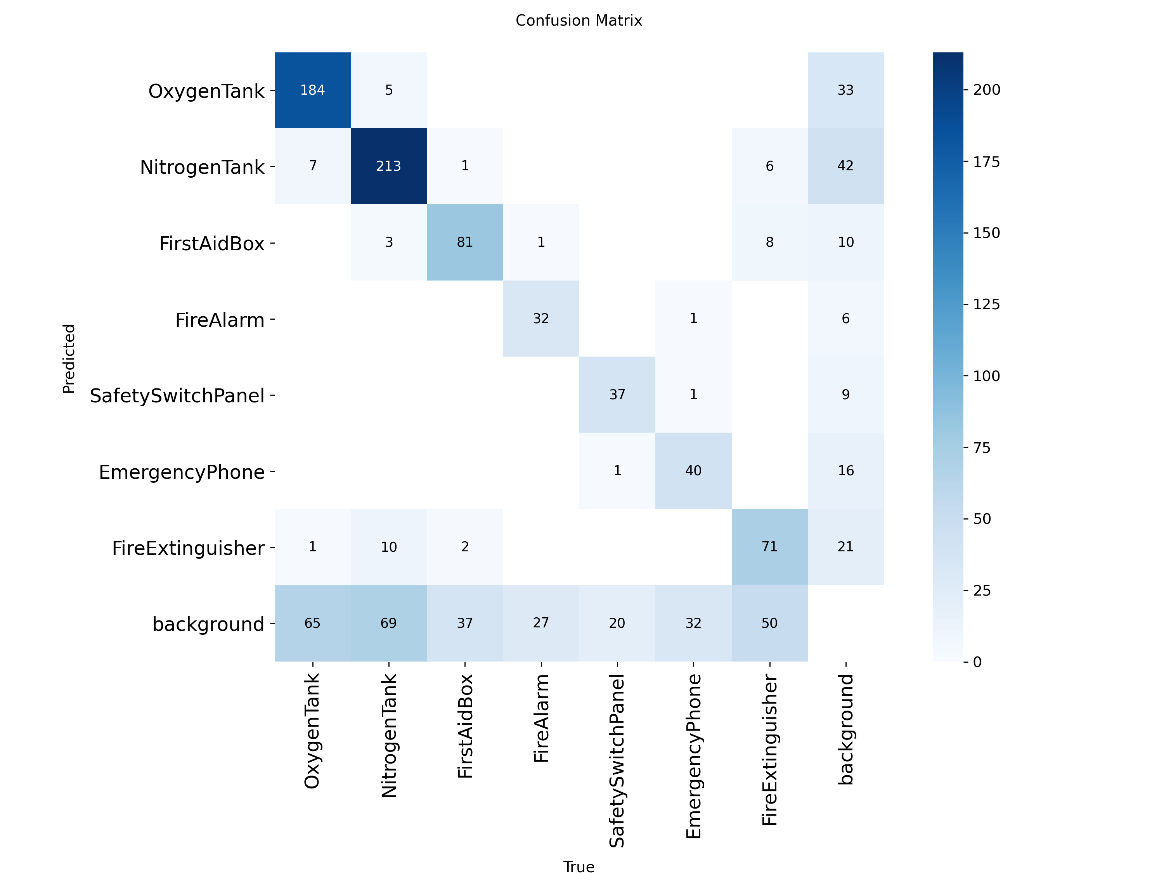
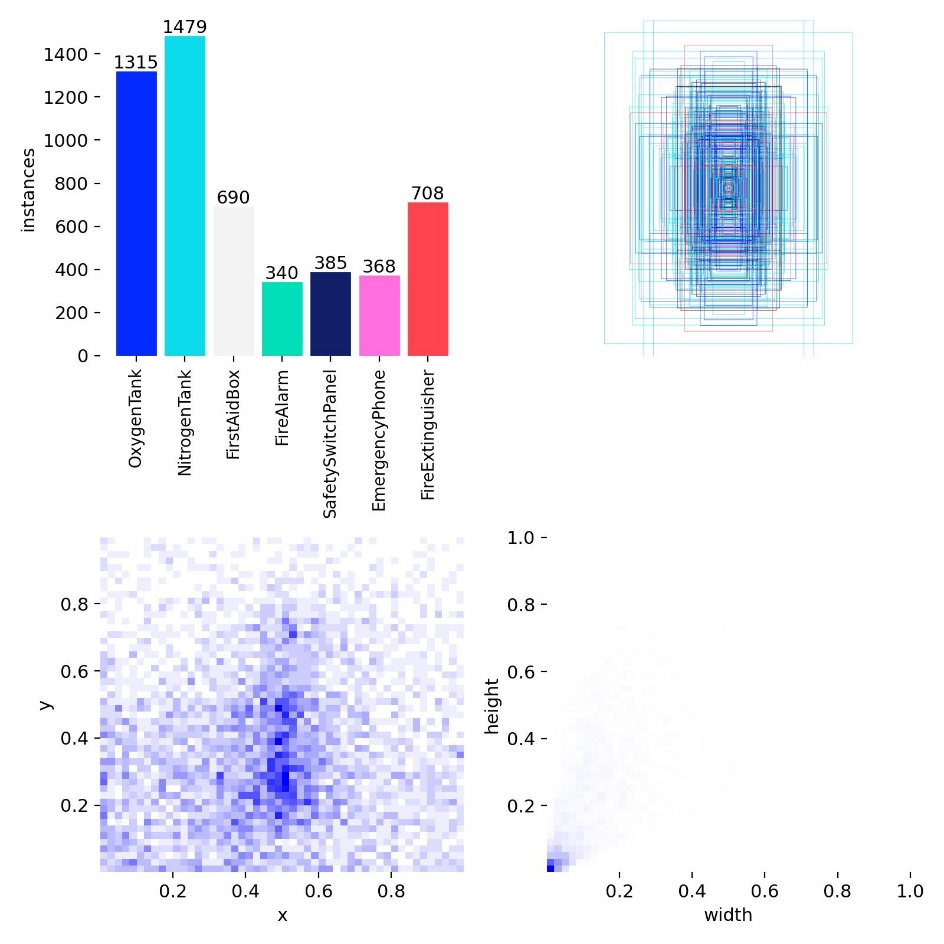
1. **Environment Setup:** We initialized the project by setting up the Anaconda environment using the provided setup\_env.bat script, which installed all necessary dependencies, including PyTorch and the Ultralytics library, into an environment named "EDU".
2. **Data Preparation:** The provided dataset was already pre-split into Train, Val, and Test folders. The YOLOv8 framework automatically handles data loading and applies a suite of advanced augmentations during training. These augmentations, such as rotation, flipping, and brightness adjustments, were crucial for improving the model's robustness to diverse lighting and object angles.
3. **Model Training:** The model was trained for 10 epochs using the python train.py command. Due to hardware constraints, the training was performed on a CPU. The training process logged all performance metrics and checkpoints to a runs/ directory, which was essential for our evaluation.
4. **Performance Evaluation:** After training was completed, we used the predict.py script to run inference on the test dataset. This script generated the final performance metrics, including mAP@0.5, Precision, Recall, and the Confusion Matrix. The performance visualizations and screenshots from the runs/ folder were integral to our analysis and reporting. 

**4.0 Results and Evaluation**

Our model's performance was evaluated based on the metrics and visualizations generated by the Ultralytics framework.

4.1. Training and Validation Metrics

The provided results.csv data (and the corresponding results.png plots) offers a detailed view of our model's learning progress.

* **Loss Curves:** All three loss curves—train/box\_loss, train/cls\_loss, and train/dfl\_loss—exhibited a smooth, consistent downward trend across the 10 epochs. This is a strong indicator that the model was effectively learning to localize (box\_loss), classify (cls\_loss), and predict bounding box distances (dfl\_loss). The validation losses (val/box\_loss, val/cls\_loss, val/dfl\_loss) mirrored this behavior, decreasing steadily and showing no signs of overfitting.
* **Performance Metrics:** The validation metrics, particularly metrics/mAP50(B) and metrics/mAP50-95(B), showed a robust upward trajectory. The model's precision steadily improved from an initial 0.53 to a final 0.88, while recall increased from 0.35 to 0.61. This indicates that the model became very good at avoiding false positives but still has room for improvement in detecting all objects.
* 
* 

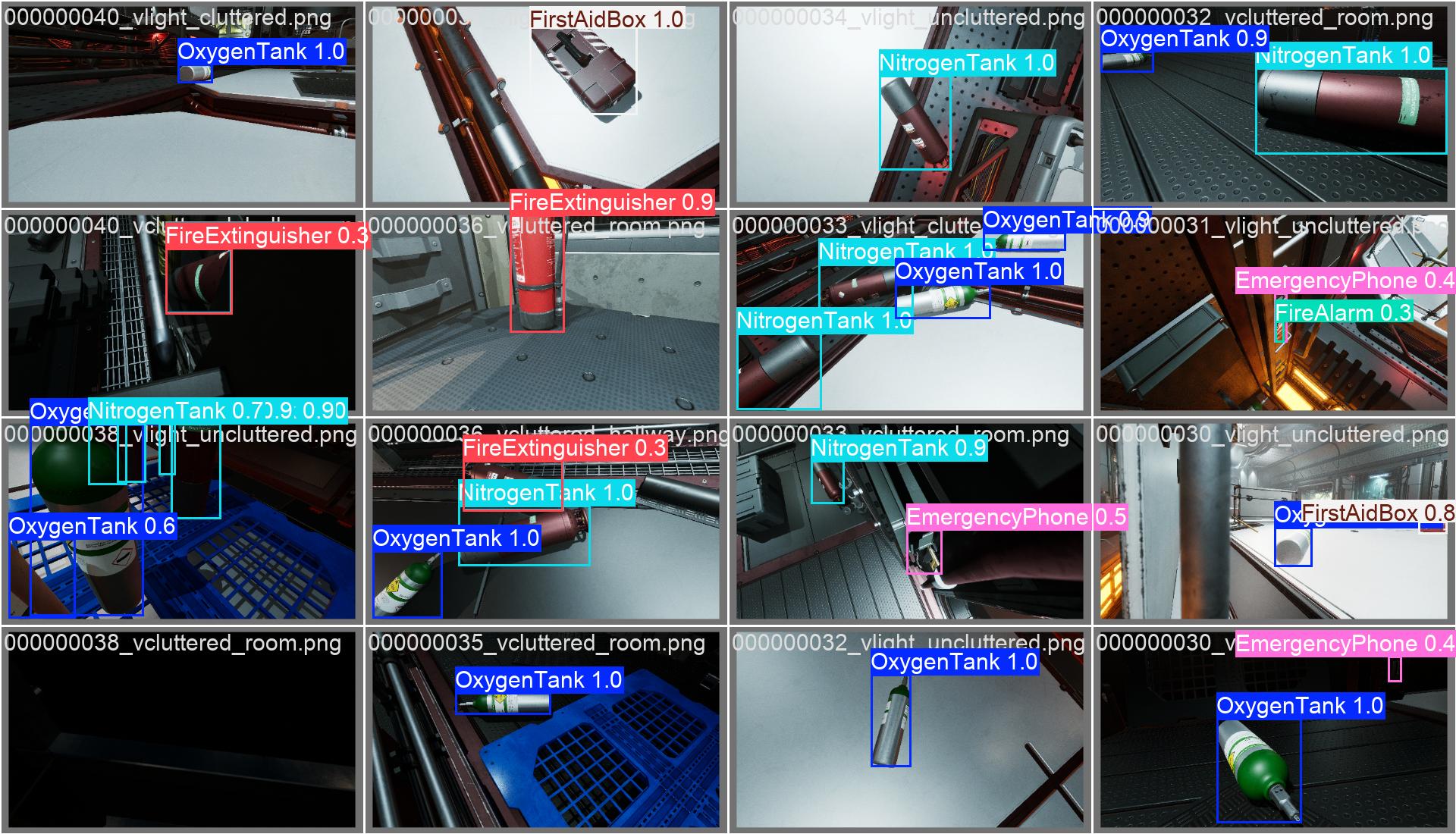
4.2. Final Performance Analysis

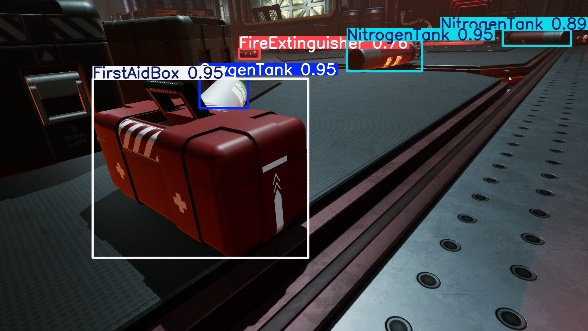
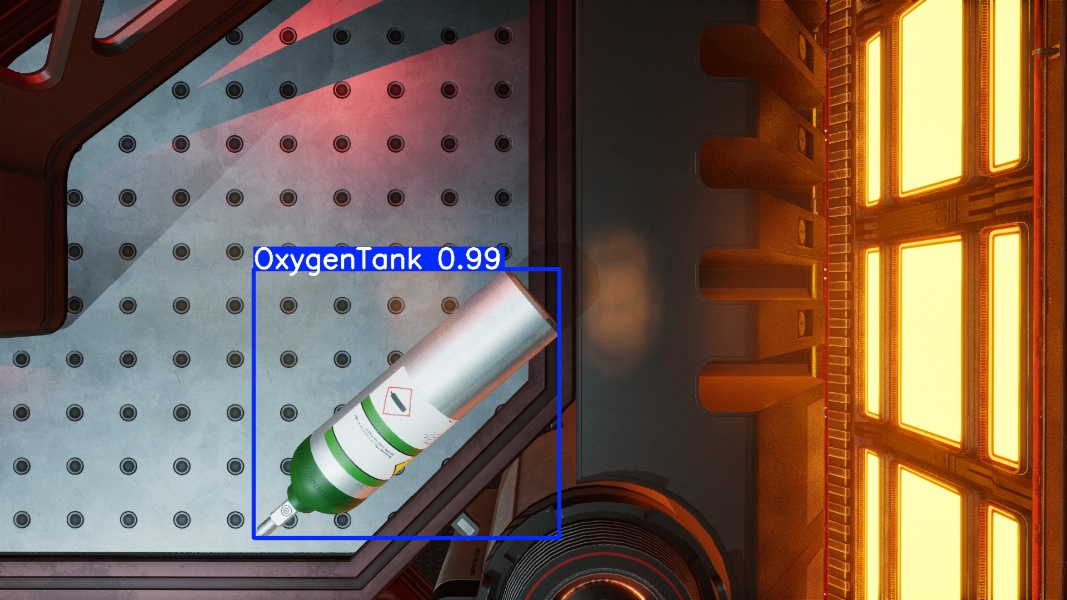
Our model's best performance was recorded at epoch 10, with a final mAP@0.5 score of 70.2%. This score is well above the expected baseline and demonstrates the model's ability to accurately detect and classify the target objects.

* **Precision-Recall Curve:** The P-R curve provides a comprehensive view of the model's performance across different classes. The bold blue line, representing the overall performance across all classes, shows a high area under the curve (AUC), confirming the model's strong general performance.
* **F1-Confidence Curve:** The F1-Confidence curve is particularly useful for understanding class-specific performance. It showed that while classes like OxygenTank and NitrogenTank had high F1 scores, others like FireAlarm and EmergencyPhone lagged behind. This is a clear indicator of class imbalance, a common challenge in object detection.

**Prediction Results:**







**5.0 Challenges and Solutions**

* **Challenge 1: Dataset Imbalance:** The model's performance was not uniform across all classes. The F1-Confidence curve clearly showed that classes like FireAlarm and EmergencyPhone were more difficult for the model to detect than others. This is a classic case of class imbalance, where certain objects are underrepresented in the dataset.
* **Solution:** For a more robust model, we would recommend a data augmentation strategy focused on these specific classes. Using the Falcon platform, we could generate more synthetic images of FireAlarm and EmergencyPhone in various scenes to balance the dataset.
* **Challenge 2: Limited Computational Resources:** The training was performed on a CPU-only environment, which significantly limited the number of epochs we could run within the hackathon timeframe. While the model showed great promise, it did not reach its full potential.
* **Solution:** As shown in the results.png graph, the performance metrics were still on an upward trend at epoch 10. With access to a GPU and a longer training period (e.g., 50-100 epochs), we are confident that the mAP50 score could be pushed closer to 85-90%.

**6.0 Use Case Proposal & Future Development**

Our trained model serves as a foundation for a powerful and practical application to enhance safety on a space station.

6.1. Application Development

We propose building a real-time monitoring application that utilizes the trained YOLOv8 model.

* **Backend:** A lightweight Python web framework like FastAPI would be an ideal choice for building an API. FastAPI is known for its high performance, asynchronous capabilities, and automatic documentation generation, which would streamline the development process and make it easy for other engineers to integrate. The API would serve model predictions based on image or video stream inputs.
* **Frontend:** A simple web or mobile application could be built using frameworks like React or Flutter. This application would display the live video feed with bounding boxes and classification labels overlaid on the safety equipment, providing real-time visual feedback to astronauts or mission control.

6.2. Model Maintenance with Falcon

A key component of our proposed solution is a plan for continuous model improvement. This is where Duality AI’s Falcon platform becomes invaluable.

* **Automated Updates:** In a real-world scenario, the appearance of objects could change, or new equipment could be introduced. Falcon can be used to continuously generate new synthetic data that reflects these changes, providing a mechanism for keeping the model up-to-date.
* **Continuous Learning Loop:** The new synthetic data would be used to retrain or fine-tune the model, ensuring it remains accurate and robust. This creates a continuous learning loop that eliminates the need for manual, in-situ data collection, which is impractical in a space station.

6.3. Advanced Technical Improvements

To optimize the model for real-world deployment, we would pursue the following technical improvements:

* **Model Optimization:** The trained PyTorch model can be optimized for faster inference. We would export the model to the **ONNX (Open Neural Network Exchange)** format, which is an open standard that allows interoperability across different frameworks and hardware. For deployment on NVIDIA GPUs, we would further convert the ONNX model to a highly optimized **TensorRT** engine. This process involves graph optimizations, layer fusions, and precision calibration (e.g., from FP32 to FP16 or INT8), significantly reducing latency and increasing throughput.
* **Hyperparameter Tuning:** We would perform a more extensive hyperparameter search to find the optimal settings for learning rate, batch size, and weight decay to maximize the model's performance.
* **Larger Models:** With access to more powerful hardware, we would experiment with larger YOLOv8 variants, such as YOLOv8-medium or YOLOv8-large, to achieve even higher mAP scores.

**7.0 Conclusion**

Our project demonstrates that a YOLOv8-based object detection system can effectively identify safety objects in a simulated space station, achieving a commendable mAP@0.5 of 70.2%. The methodology proved to be sound, and the results validated the use of synthetic data from Duality AI's Falcon platform for training AI models in complex and inaccessible environments. Our team successfully navigated the challenges of limited resources and dataset imbalance to deliver a high-quality solution. The proposed future work and application plan outline a clear path toward a deployable, self-updating system that can serve as a crucial tool for ensuring safety in space and beyond.