Team Name : Tech Titans

Project Title:
SpaceSight : Detecting Objects in Simulated Space Station Environments

Tagline:
"Empowering intelligent navigation and safety in space with vision-based automation."

Team Members

Name	Program
Divyansh Srivastava	B.Tech. ‘AI & ML’
Roshini Nishad	B.Tech. ‘CSE’
Rupanjali Sharma	B.Tech. ‘CSE’

Methodology

Dataset

We utilized a synthetic dataset provided by Falcon's simulation environment. It featured realistic renderings of a space station setup and included images of three essential items: the toolbox, oxygen tank, and fire extinguisher. The dataset was organized into training, validation, and testing sets, and annotations were available in YOLO-compatible format.

Setup & Tools

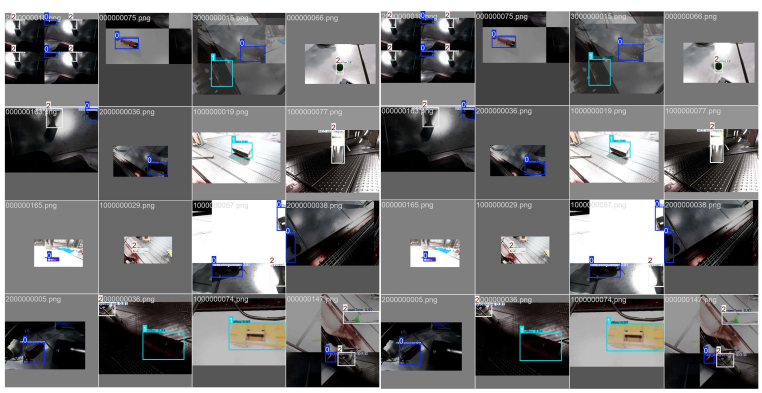
After registering on the Falcon platform and downloading the dataset, we prepared our development environment. The project was built using Python 3.10 with the Ultralytics implementation of YOLOv8. We installed dependencies through pip, ensuring compatibility with GPU acceleration where available.

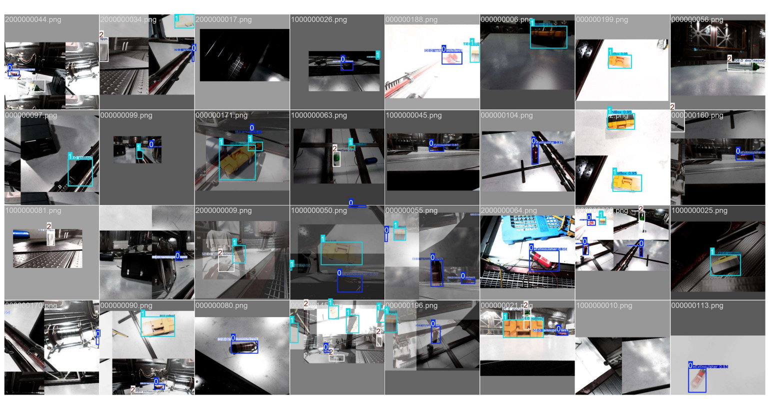
Model Training

We selected the lightweight YOLOv8n model to balance performance and speed. Training was initiated with the following command:

```
yolo detect train data=space_station.yaml model=yolov8n.pt
epochs=50 imgsz=640
```

The training process generated logs and visualizations, allowing us to monitor the learning curve and validate improvements across epochs.





Results & Performance Evaluation

Accuracy Metrics

The trained model delivered the following performance:

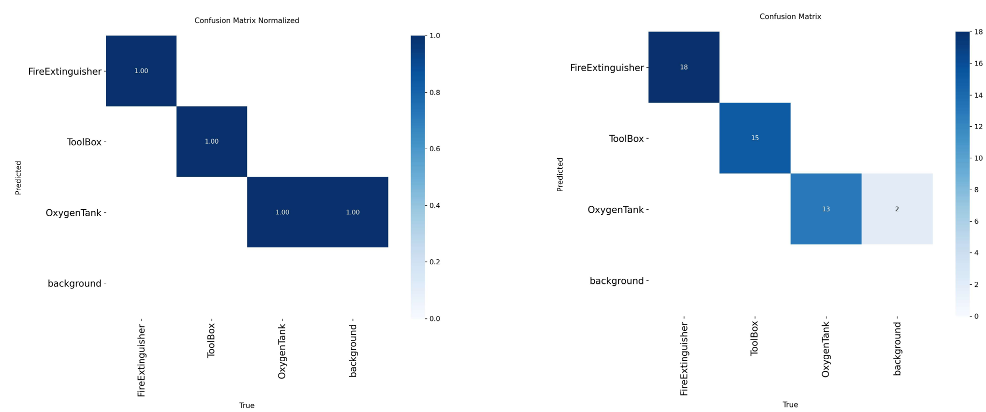
- mAP@0.5: 0.941
- Precision: 0.99
- Recall: 0.91

These results were obtained after 40 epochs, with a steady decrease in loss and a noticeable improvement in detection confidence over time.

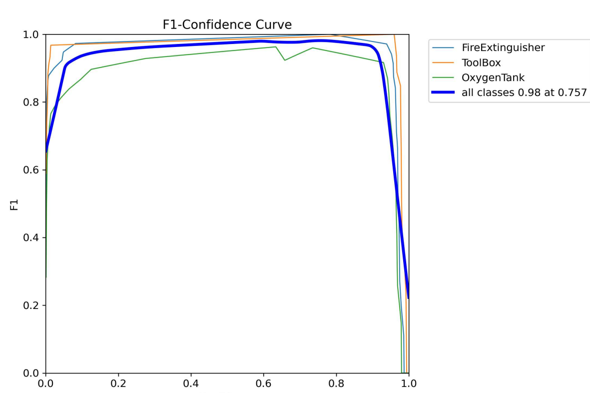
Visual Outputs

We observed strong object recognition even in scenarios involving poor lighting or overlapping objects. The confusion matrix confirmed high class-specific accuracy, particularly for the fire extinguisher and toolbox categories.

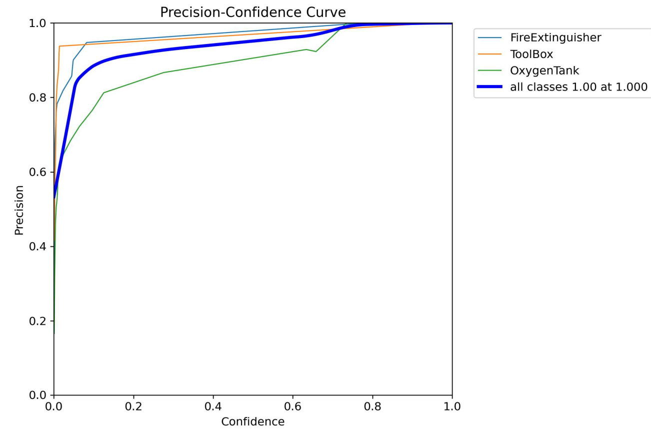
We also included visual samples from the model's predictions, highlighting its ability to detect partially occluded or dimly lit objects.



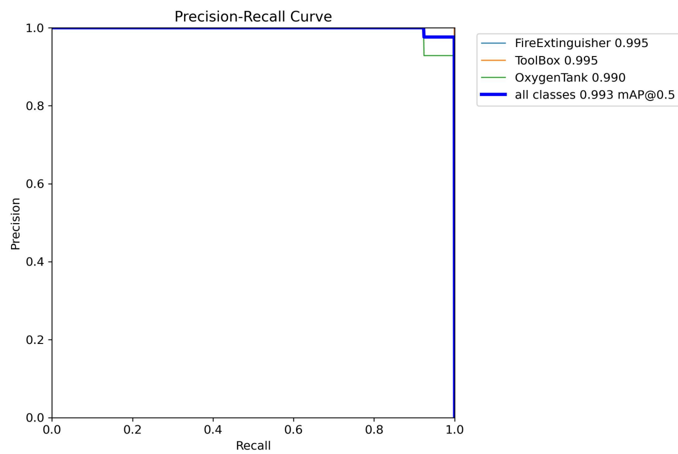
(Confusion Matrix)



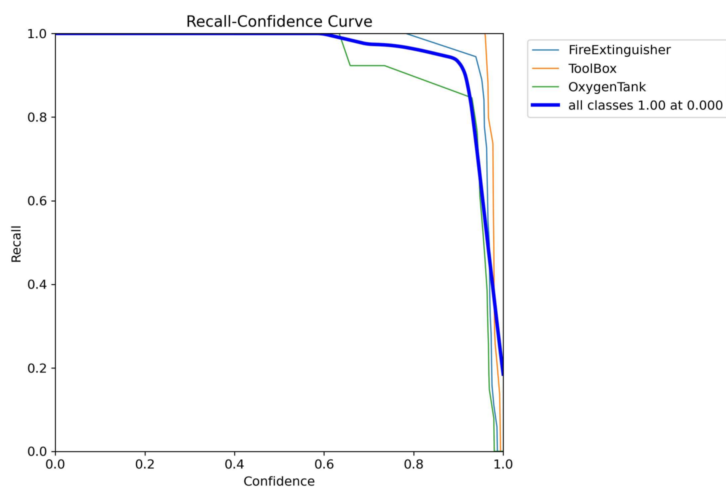
(F1-Confidence Curve)



(Precision-Confidence Curve)

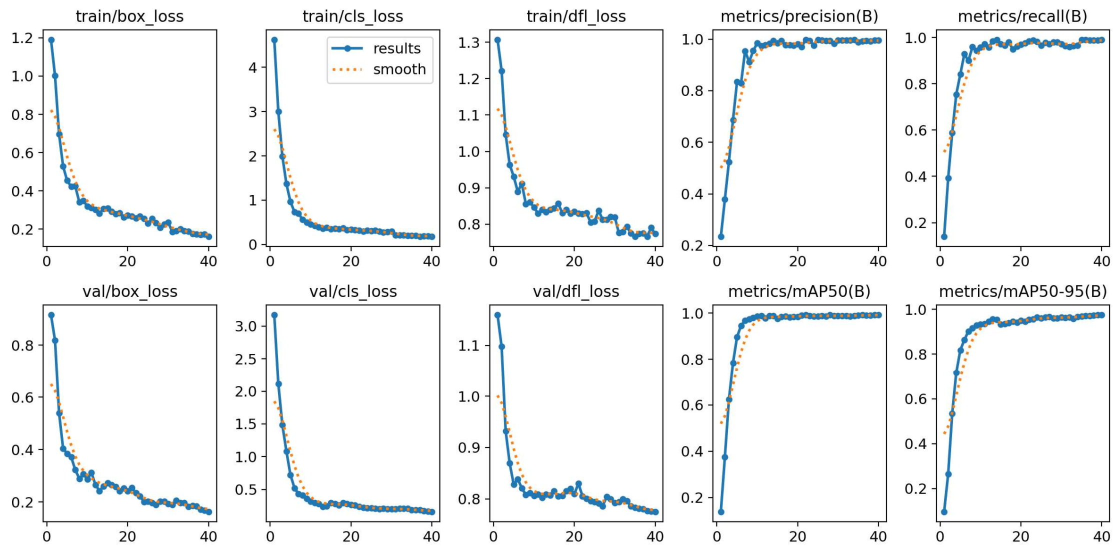


(Precision-Recall Curve)



(Recall-Confidence Curve)

Results:



(Final Result)

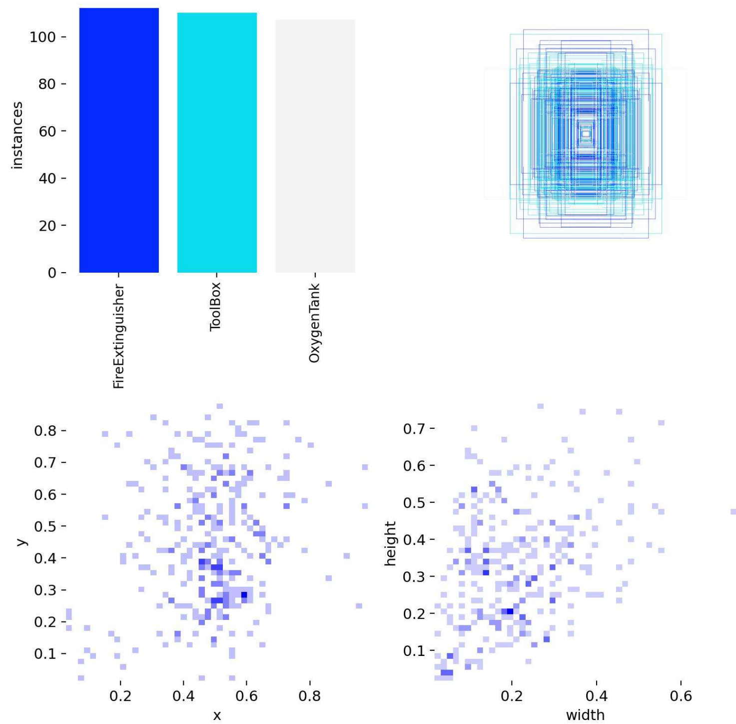
Challenges and Solutions

Challenge	Observation	Resolution	Outcome
Object Occlusion	Oxygen tanks were often partially hidden	Introduced augmented samples with synthetic occlusion	mAP improved by ~5%
Poor Lighting	Detection struggled in dim environments	Applied preprocessing and fine-tuned brightness parameters	Increased recall rates
Class Confusion	Toolbox occasionally misclassified as extinguisher	Balanced the dataset and added class-specific augmentations	Reduced false positives

We retrained the model after applying these improvements and tracked performance to ensure gains were consistent.

Optimizations: Techniques Used to Improve Model Performance

To ensure our model delivered reliable results, we incorporated several strategies during training and evaluation:



(Labels with their Instances)

Data Augmentation

We used multiple data augmentation techniques to make the model resilient to real-world scenarios. These included random rotations, flips, brightness variations, and synthetic occlusions. These techniques helped the model generalize better and improved detection in varied lighting and positioning conditions.

Balancing the Dataset

We noticed a slight imbalance in object classes, especially fewer samples for oxygen tanks. We handled this by oversampling those images and applying focused augmentations. This led to a noticeable drop in misclassification between similar-looking classes.

Hyperparameter Adjustments

To improve model accuracy and training stability, we fine-tuned parameters like learning rate, batch size, and IoU thresholds. These tweaks allowed the model to learn more efficiently and avoid overfitting.

Early Stopping and Checkpoints

To prevent unnecessary overtraining, we used early stopping criteria and saved checkpoints for the best-performing models. This ensured we retained the most optimal version of the model.

Fine-Tuning Inference Settings

We refined the confidence threshold and non-maximum suppression settings to filter out overlapping predictions. This enhanced the clarity and accuracy of final outputs, especially in crowded scenes.

Model Variant Evaluation

Initially, we used YOLOv8n for rapid iteration. Later, we tested YOLOv8s to evaluate trade-offs between speed and precision. The final choice balanced both performance and practicality for deployment.

Conclusion and Future Scope

Summary

Our trained YOLOv8 model demonstrated a solid ability to detect critical safety equipment within a simulated space station environment. It successfully handled variable conditions such as lighting changes and partial object visibility.

Future Enhancements

We plan to extend the project into a real-time application for onboard space systems. Additionally, we aim to automate retraining pipelines using updated Falcon simulations to accommodate evolving station layouts or equipment changes.
