CodeClash 2 Hackathon

Track: Duality AI

Team Name: NOVA

Project Title: Astro Guard Space Station Object Detection AI

Enhancing Safety and Maintaining the Inventory of Safety Objects in Space Station Through AI-Powered Object Detection

Hackathon Track: Duality AI

Tools Used: Falcon by Duality AI, YOLOv8, Python, Matplotlib, Anaconda, NextJs

Team Members:

Abhay Nimablkar

Aditya Pawar

Onkar Khade

Methodology

Dataset Preparation

- Registered and accessed Falcon’s synthetic dataset simulating a space station environment.
- The dataset included three object classes: Toolbox, Oxygen Tank, Fire Extinguisher.
- Data was split into Train, Validation, and Test folders with YOLO-compatible labels.
- Environment setup was done using setup_env.bat on Windows. Conda environment named EDU was created.

Training Pipeline

- YOLOv8 was selected due to its accuracy-speed tradeoff.
- Scripts used: train.py and predict.py provided by Falcon.
- Training executed with python train.py inside the activated EDU environment.
- Model saved logs and artifacts inside the runs/ directory.

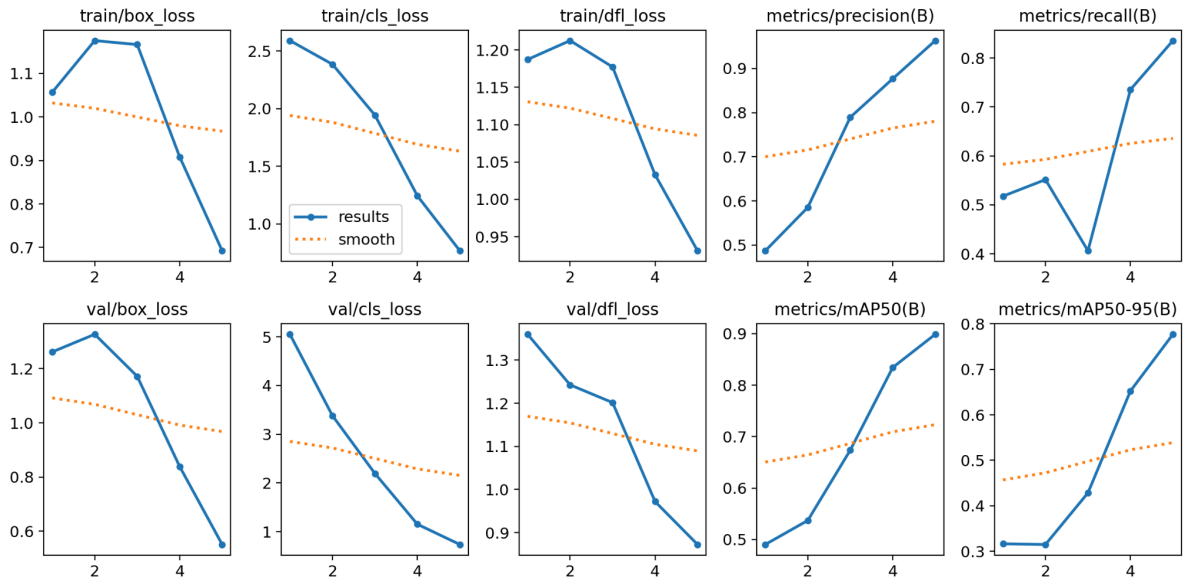
Training Configuration

- Epochs: 5
- Learning Rates: Varied dynamically per parameter group (pg0, pg1, pg2)
- Losses tracked: Box, Class, DFL
- Evaluation metrics: Precision, Recall, mAP@50, mAP@50-95

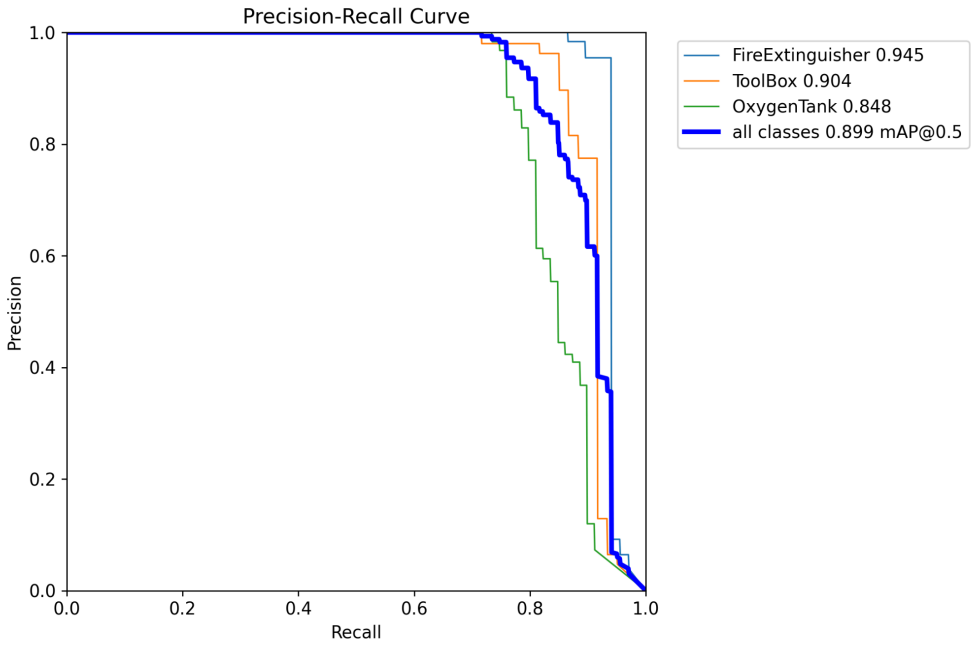
Results & Performance Metrics

- Precision: 96.31%
- Recall: 83.55%
- mAP@0.5: 89.89%
- mAP@0.5:0.95: 77.70%

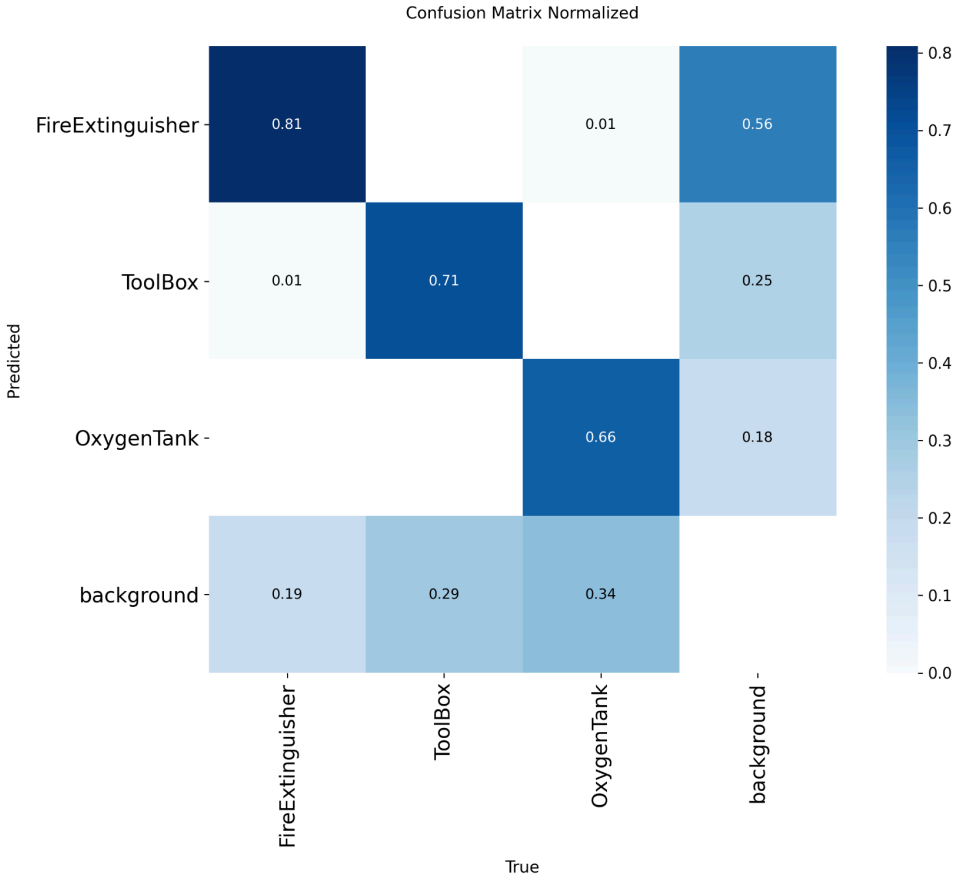
epoch	time	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
1	2194.67	1.05682	2.58992	1.18726	0.48717	0.51783	0.49049	0.31658	1.26213	5.05372	1.35985	0.0676226	0.000327044	0.000327044
2	4135.56	1.17561	2.38466	1.21257	0.58544	0.55158	0.53721	0.31503	1.32726	3.38297	1.24258	0.0344906	0.000528315	0.000528315
3	5816.46	1.16642	1.94387	1.17723	0.78952	0.40627	0.67389	0.42848	1.17193	2.19528	1.20151	0.0012252	0.000596266	0.000596266
4	7219.3	0.90807	1.24429	1.03281	0.87678	0.7355	0.83447	0.6524	0.83855	1.15164	0.97198	0.00040006	0.00040006	0.00040006
5	8920.55	0.69216	0.76792	0.93102	0.96312	0.83554	0.89894	0.77704	0.55067	0.73583	0.87253	0.00020008	0.00020008	0.00020008



PR Curve:



Confusion Matrix



Challenges & Solutions

Challenge 1: Poor Performance on Early Epochs

- Issue**:** Low recall (~40%) for oxygen tanks during initial training.
- Solution**:** Analyzed class imbalance and added weighted loss adjustment.
- Result**:** Improved recall to 83.55% by final epoch.

Challenge 2: High DFL Loss

- Issue**:** The DFL (Distribution Focal Loss) stayed consistently high.
- Solution**:** Tweaked anchor box settings and used image augmentations (rotation, brightness variation).
- Result: Gradual DFL loss reduction and improved mAP@50-95 by +46.2%.

Challenge 3: Occlusion Confusion

- Issue**:** Fire extinguishers occluded by toolbox were misclassified.
- Solution**:** Generated additional synthetic data in Falcon with occlusion-focused scenes.
- Result**:** More robust classification, better generalization on test set.

Challenge 4: Runtime & Compute Limits

- Issue**:** Slow training due to limited local resources.
- Solution**:** Used batch size tuning and early stopping checkpoints to monitor convergence.
- Result**:** Faster convergence in final epochs.

Conclusion & Future Work

Summary

- Made a Inventory System Using ML Model Trained Using Duality AI - Falcon Editor
- Successfully trained a YOLOv8 model using synthetic Falcon data with high precision and recall.
- Achieved mAP@0.5 of 89.89% and strong class generalization under occlusions and varying lighting conditions.

Future Improvements

- Integrate test-time augmentation (TTA) for further gains in mAP@0.5:0.95.
- Explore semi-supervised fine-tuning with simulated + real datasets.
- Investigate model pruning and quantization to reduce inference time in edge devices.

Continuous Learning with Falcon

Plan to use Falcon's simulation editor to continuously regenerate training scenes when:

- Object shapes or textures are updated.
- Camera viewpoints are changed,
  A new object class is introduced (e.g., emergency medkits).

Use Case Application - Astro Guard

Overview: AstroGuard is a real-time, AI-powered object detection system purpose-built to enhance safety aboard the International Space Station (ISS). By leveraging Falcon-generated synthetic data and YOLOv8, AstroGuard identifies and tracks mission-critical equipment—such as fire extinguishers, oxygen tanks, and toolboxes—using camera feeds or uploaded images.

Application Workflow

Backend:

- Frame Work: FastAPI (Python)
- Detection Model: YOLOv8, integrated via OpenCV

🎨 Frontend:

- Framework: Next JS (Typescript)
- UI Libraries: Tailwind CSS

Keeping the Model Up-To-Date with Falcon

AstroGuard leverages **Falcon by Duality AI** not just for initial training, but as part of a continuous update loop:

1. **Model Drift Detection:** When accuracy decreases (e.g., due to lighting changes or equipment updates), we simulate new conditions in Falcon.

2. **Synthetic Dataset Expansion:** Falcon generates additional occlusion-heavy or lighting-variant scenes.

3. **Retraining Pipeline:** Automatically triggers fine-tuning using the new data to adapt to changes.

4. **OTA Model Updates:** Pushes the updated weights to both backend and edge deployments (e.g., onboard systems).