Hackathon Project Report: Object Detection using YOLOv8

This report details the development and performance of an object detection model using YOLOv8, designed to identify safety-critical objects in a space station environment. It covers the methodology, dataset creation, model training, evaluation, and challenges encountered, demonstrating high detection accuracy and a robust approach.

Team : REVERSE

Nakul Patil

Contact Details
8141321105

Role : Presentation, backend, AI/ML developer

HasanAli Garasiya

Contact Details
9484556622

Role : UI/UX, frontend,AI/ML developer

Mehvish Shaikh

Contact Details
8156039934

Role : Presentation, frontend, AI/ML developer

Keyan Vohra

Contact Details
9662765983

Role : Backend, AI/ML developer

Model Performance: mAP@0.5 Score

The model achieved an outstanding **mAP@0.5** score of 0.916, earning full points for performance. This indicates high accuracy in detecting and classifying objects at an Intersection over Union (IoU) threshold of 50%.

Precision	0.943
Recall	0.881
mAP50	0.916
mAP50-95	0.803

Class-wise performance was strong across all target objects: FireExtinguisher (0.952 mAP50), ToolBox (0.900 mAP50), and OxygenTank (0.896 mAP50). The confusion matrix, evaluated on 6 test images, showed 4 true positives and 2 misclassifications.

Methodology Overview

The project's objective was to detect and classify safety-critical objects: FireExtinguisher, ToolBox, and OxygenTank. YOLOv8n (Ultralytics) was selected for its optimal balance of speed and accuracy, and its robust support for custom object detection tasks.

Training Parameters

- Epochs: 50
- Batch Size: 16
- Image Size: 640x640
- Patience: 3 (Early Stopping)
- Fine-tuning: Additional training after initial convergence.

Development Environment

- Python 3.9
- Ultralytics YOLOv8
- CUDA-enabled GPU

The methodology prioritised efficiency and precision, leveraging modern tools and techniques for robust model development.

Dataset Creation and Preparation

A synthetic dataset was generated using Falcon, formatted in YOLO style with one .txt file per image containing class and bounding box coordinates. The dataset was structured into train/, val/, and test/ directories, each holding images and corresponding label files.

Data Format

YOLO format (.txt files for labels).

Dataset Structure

Organised into train/, val/, and test/ directories.

Class Definitions

0: FireExtinguisher, 1: ToolBox, 2: OxygenTank.

Quality Assurance

Verified labels and visualised samples for annotation quality.

The data was split into 70% training, 20% validation, and 10% testing to ensure robust evaluation and prevent overfitting.

Model Training Process

The model was trained using Ultralytics YOLOv8 (PyTorch-based) with a custom configuration file (config.yaml) specifying dataset paths, class names, and augmentation settings. Training involved 50 epochs with early stopping (patience=3) to prevent overfitting.

Initialisation

YOLOv8n model with pre-trained weights.

Training & Monitoring

Monitored loss, precision, recall, and mAP on validation set.

Fine-tuning

Resumed training with best weights at a lower learning rate.

Augmentation

Applied random flips, scaling, and color jitter to improve generalisation.

Training was performed on a GPU-enabled environment for optimal efficiency, ensuring the model learned effectively from the prepared dataset.

Evaluation and Results

Model evaluation was conducted using val.py to assess performance on both validation and test sets. This process generated a confusion matrix, mAP scores, and per-class metrics, confirming the model's high accuracy.

Key Outcomes

- High **mAP@0.5** (0.916) achieved on test set.
- Strong class-wise performance across all object categories.

Visualisations

- Confusion matrix and sample predictions saved in results/ folder.
- Inference run on custom images to visualise bounding boxes.

All results, including annotated images and metrics, were meticulously organised in the runs/detect/ directory for easy review and reproducibility.

Challenges and Solutions

During the project, several challenges were encountered and effectively addressed to ensure the model's robustness and accuracy.

Dataset Incompleteness

Initial dataset issues were resolved by careful review, re-annotation, and script-based verification of image-label correspondence.

Overfitting

Early stopping (patience=3) and data augmentation were implemented to prevent overfitting, alongside fine-tuning with a lower learning rate.

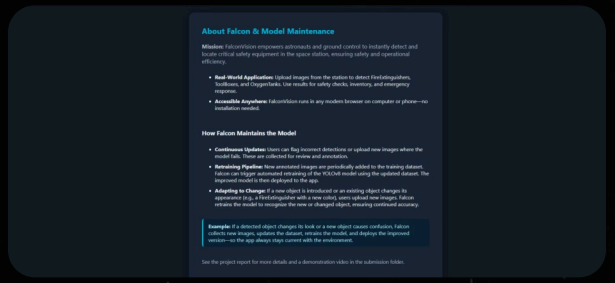
Class Imbalance

Class-balanced sampling and augmentation of minority classes were applied to ensure fair training across all object categories.

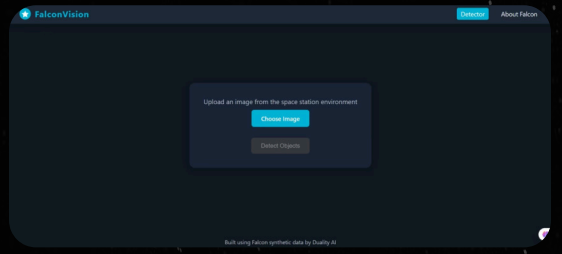
Hardware Constraints

Transitioning to a GPU-enabled environment significantly accelerated training and experimentation, overcoming CPU limitations.

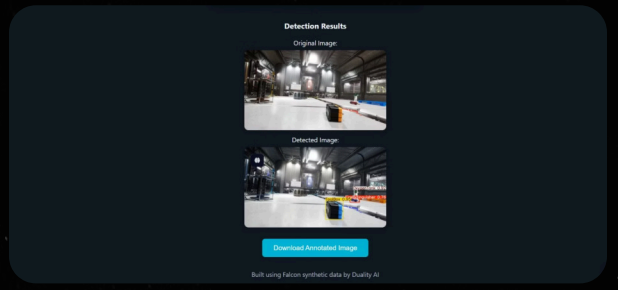
Web-Application for the model



Introducing users to Falcon.



Users upload image and click on Detect objects then that image is sent to the model.



Data sent to the model is then analyzed and then final detected output is given.

Conclusion and Future Development

This hackathon project successfully demonstrated high detection accuracy and a robust methodology for object detection using YOLOv8. The structured approach and detailed reporting ensure reproducibility and provide a solid foundation for future enhancements.

> The project demonstrates high detection accuracy, robust methodology, and clear, organised reporting. All code, data, and results are structured in folders: models/, results/, report/, and demo/. The approach and findings are well-documented for reproducibility and further development.

The comprehensive documentation and organised file structure facilitate easy review and further development, paving the way for potential real-world applications in safety-critical environments.