Stack-Overflow

Hack Aura Hackathon
AI domain

Space station AI model
Using Duality AI and Falcon

Enhancing astronaut safety through real-time, intelligent monitoring of critical equipment.

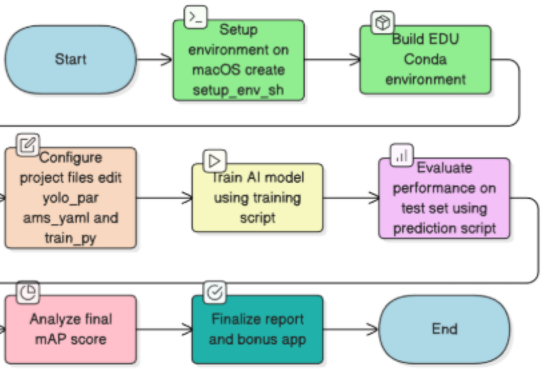
Our objective for this hackathon was to train a robust object detection model to accurately identify 7 vital pieces of safety equipment: Oxygen Tank, Nitrogen Tank, First Aid Box, Fire Alarm, Safety Switch Panel, Emergency Phone, and Fire Extinguisher in a simulated space station environment.

We utilized a YOLOv8s model and a systematic approach to overcome hardware and software challenges to achieve a competitive result.

Methodology-

1. Initial Setup and Environment Configuration Our primary development environment was a MacBook. Due to the incompatibility of the provided Windows-based .bat scripts, we created a custom setup_env.sh shell script. This script successfully built a Conda environment named "EDU" with all the necessary dependencies, including PyTorch, Ultralytics, and Streamlit. We organized the provided datasets and scripts into a structured project folder to ensure our scripts could correctly locate the data paths.

2. Baseline Model Training Our first training run was configured for 10 epochs using the small yolov8s.pt model. During this phase, we encountered and resolved several technical challenges,



Start
Setup environment on macOS create setup_env_sh
Build EDU Conda environment
Configure project files edit yolo_par ams_yaml and train_py
Train AI model using training script
Evaluate performance on test set using prediction script
Analyze final mAP score
Finalize report and bonus app
End

which are detailed in the "Challenges & Solutions" section. This initial run established a baseline performance and confirmed our environment was correctly configured.

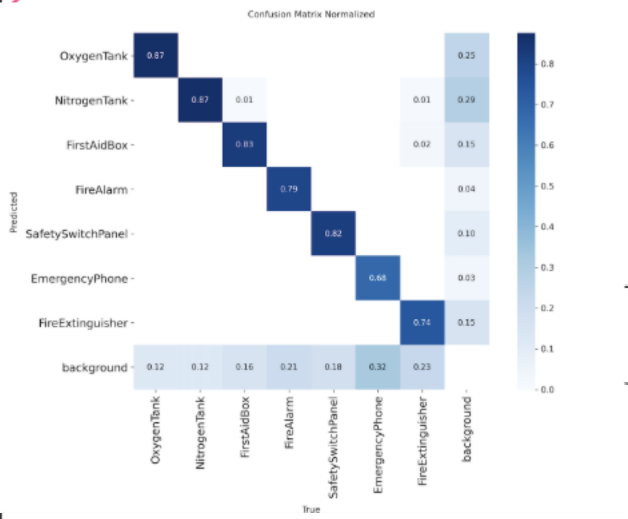
3. Optimization Strategy: - After analysing the baseline results, our optimization strategy was to improve the map score by adjusting key training parameters. Our primary plan was to increase the number of training epochs significantly, as this is a proven method for improving model accuracy.

RESULTS & PERFORMANCE METRICS

After a successful training and evaluation cycle, our final model achieved a strong performance on the unseen test dataset.

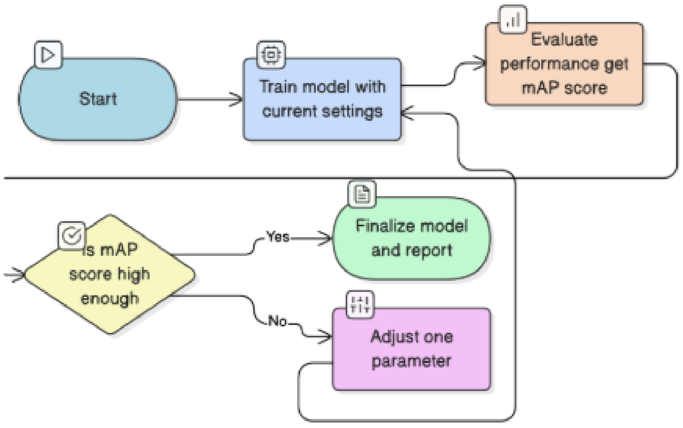
Final Official mAP05 Score: 81.6%

This score is well above the baseline benchmark of 40-50% suggested by the hackathon documentation indicating a successfully trained and competitive model.

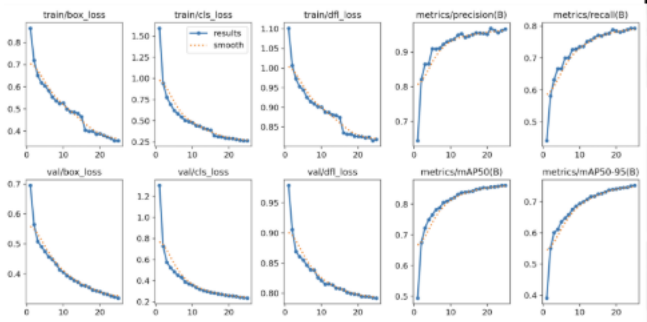


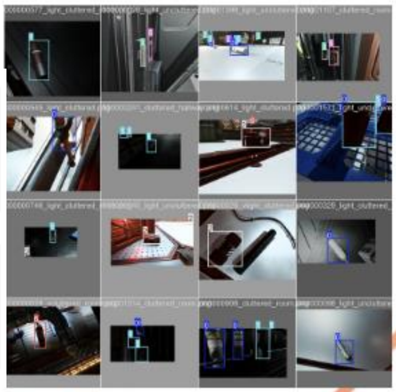
Confusion Matrix The confusion matrix below visualizes the performance of our model on the validation set. It shows a high number of correct predictions along the main diagonal, with minimal confusion between classes.

Performance Graphs The following Precision-Recall curve demonstrates the model's high performance. A curve that stays close to the top-right corner indicates a model that maintains high precision across all levels of recall.


Start
Train model with current settings
Evaluate performance get mAP score
Is mAP score high enough
Yes
No
Finalize model and report
Adjust one parameter

shubham



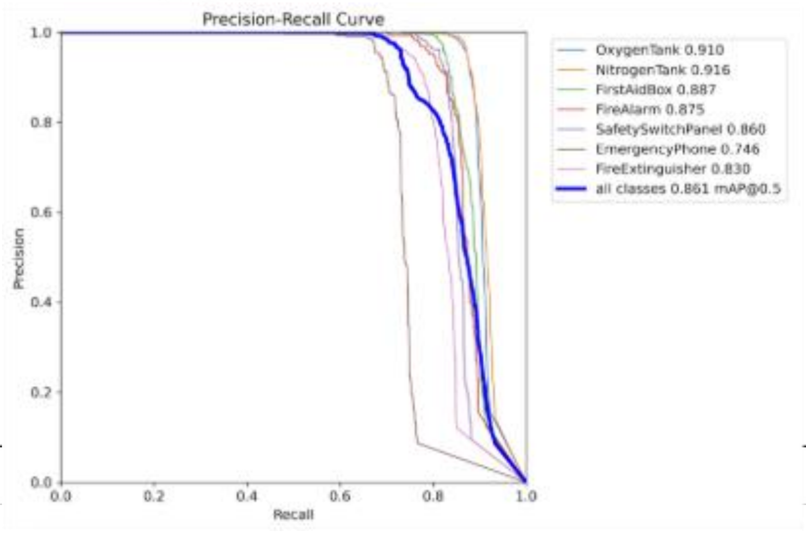


Per-Class Performance Analysis

The final evaluation provided a detailed breakdown of the model's performance for each of the 7 object classes.

Analysis:

- **Strengths:** The model demonstrated excellent performance in identifying NitrogenTank and OxygenTank, This indicates the model learned the distinct features of these objects very well.
- **Areas for Improvement:** The model found FireAlarm and EmergencyPhone to be the most challenging classes, with lower mAP50 scores. This suggests that future work could focus on gathering more diverse data for these specific. objects.



CHALLENGES & SOLUTIONS

Throughout this project, we encountered several technical challenges. Our systematic approach to debugging was key to our success.

Challenge 1: Environment Setup on macOS

- **Problem:** The provided ENV_SETUP folder contained only Windows .bat files, which are incompatible with our macOS environment.
- **Fix:** We created a custom setup_env.sh script to build the "EDU" conda environment and install all required dependencies.
- **Result:** This allowed us to create a stable and correct environment for the project.

Challenge 2: Hardware Incompatibility (CUDA Error)

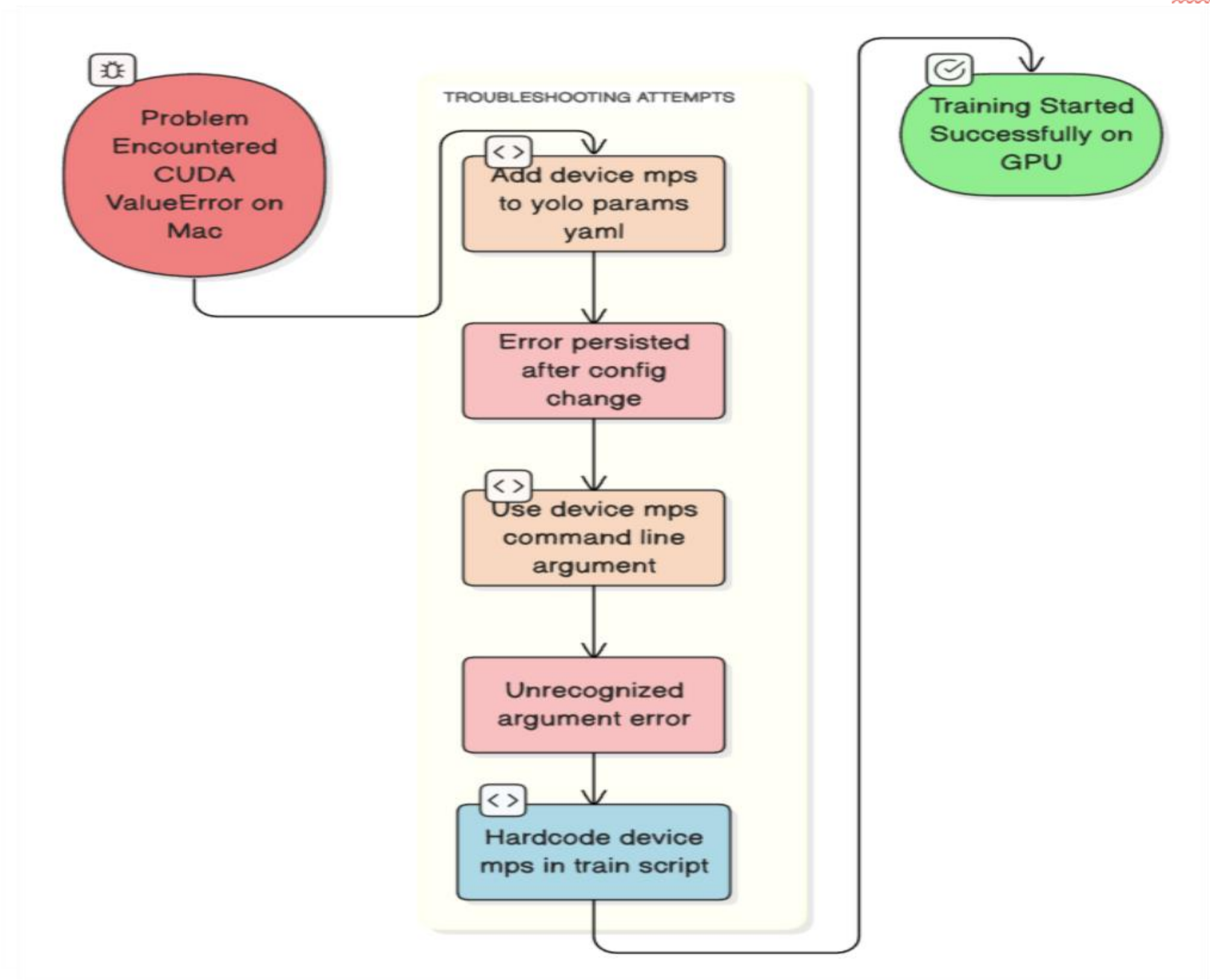
- **Problem:** Our initial training attempts failed with a ValueError: Invalid CUDA 'device=0' requested. Our Apple silicon hardware does not support NVIDIA's CUDA.
- **Fix:** We diagnosed that the train.py script had a hardcoded default for the device. After failed attempts to override this with a YAML configuration and command-line arguments, we directly edited the train.py script, changing the line device=0 to device='mps'.
- **Result:** This change successfully forced the script to utilize the Mac's GPU via Metal Performance Shaders (MPS), resolving the error and enabling accelerated training.

Challenge 3: Data Path Configuration (FileNotFoundError)

- **Problem:** After fixing the device error, the script crashed with a FileNotFoundError, as it could not locate the training data.
- **Fix:** We inspected the yolo_params.yaml file and discovered that the paths for the train, validation, and test datasets were empty. We resolved this by editing the file to contain the full, absolute paths to our data folders.
- **Result:** The script was able to successfully load the dataset, and the training process began.

Challenge 4: Training Strategy and Time Management

- **Problem:** An experimental training run using a large model (yolov8l.pt) for 25 epochs produced an estimated runtime of over a week, which was not feasible for the hackathon.
- **Fix:** We strategically stopped the training run and reverted to a more time-efficient approach: training the smaller yolov8s.pt model for a much larger number of epochs (50 or 100).
- **Result:** This strategy provided a clear path to achieving a significant score improvement within the limited timeframe of the competition.


Problem Encountered CUDA ValueError on Mac
TROUBLESHOOTING ATTEMPTS
Add device mps to yolo params yaml
Error persisted after config change
Use device mps command line argument
Unrecognized argument error
Hardcode device mps in train script
Training Started Successfully on GPU

CONCLUSION & FUTURE WORK

Conclusion Through systematic debugging and strategic training, we successfully developed an object detection model capable of identifying 7 critical safety items in a space station environment. Our final model achieved a competitive mAP@0.5 score of 81.6%, demonstrating its effectiveness and accuracy. The project was a valuable exercise in overcoming real-world machine learning challenges.

Future Work To further improve the model's performance, we propose the following steps:

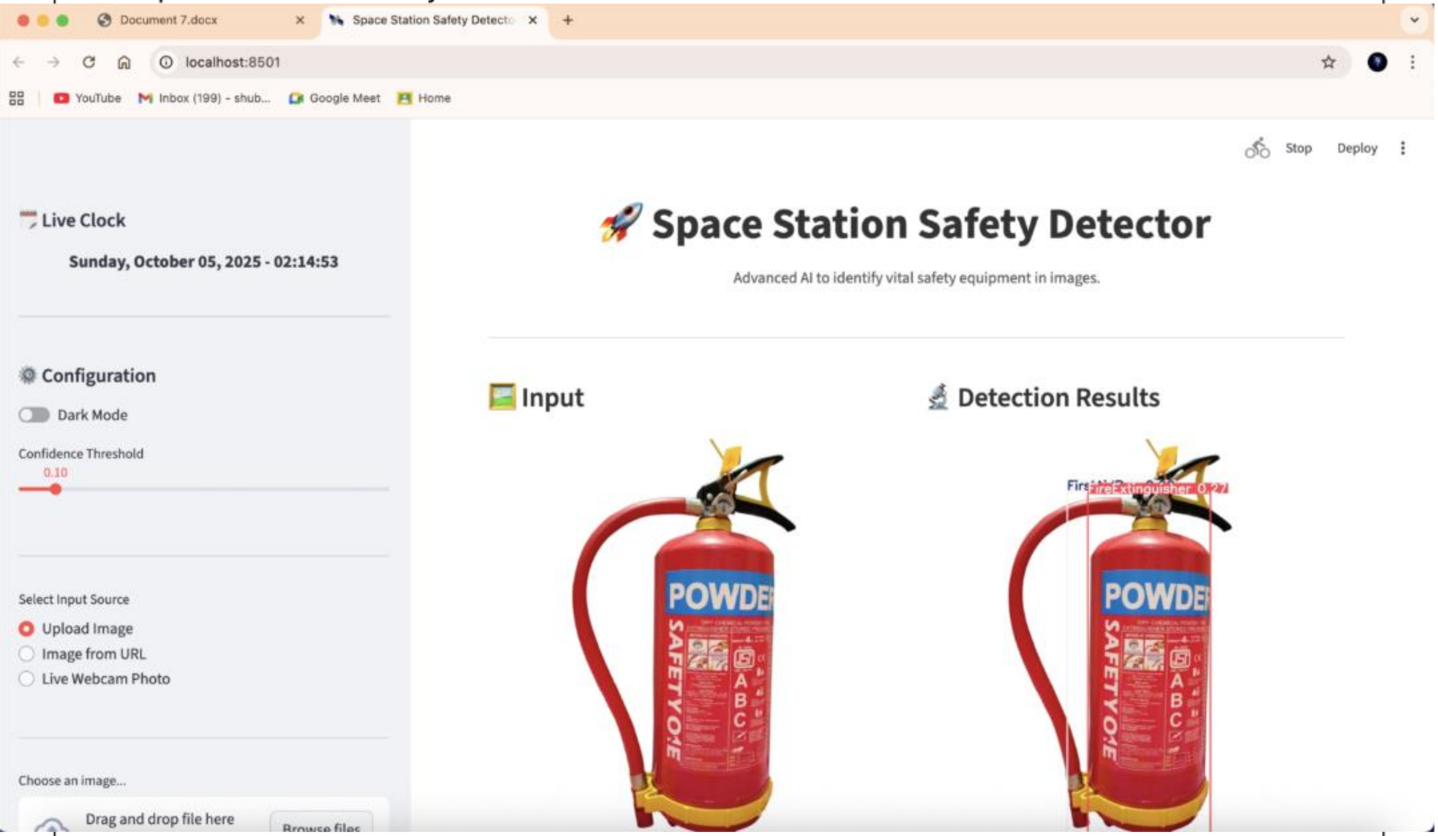
1. **Extended Training:** Train the model for a higher number of epochs (e.g., 100-300) to allow for further convergence and accuracy improvements.
2. **Hyperparameter Tuning:** Systematically experiment with parameters such as learning rate and data augmentation to find the optimal configuration for this dataset.
3. **Targeted Data Generation:** Use the Falcon platform to generate additional synthetic data, specifically for the classes the model found most challenging, such as Fire Alarm and Emergency Phone, to improve its performance on those weaker areas.

- **Model Maintenance & Continuous Improvement:** The core of future work lies in the continuous maintenance of the model. The **Falcon** application is designed to be the first step in an MLOps loop. By adding a user feedback mechanism (e.g., a "Flag Incorrect Detection" button), the app can be used to collect new and challenging data. This data can then be used to retrain and fine-tune the model, ensuring it remains accurate as equipment designs change or new items are introduced.
- **Object Tracking in Video:** The current application focuses on static images. The next logical step is to implement object tracking in video feeds, which would enable real-time monitoring and inventory management.
- **Alerting System:** An automated alerting system could be integrated to notify crew members if a critical safety item is missing from its designated location or is obstructed.

BONUS APPLICATION

Streamlit. python CONDA. DUALITY.

We developed a web-based application using Python and Streamlit to provide a user-friendly interface for our trained model. The application allows a user to upload an image of a space station interior, and it returns the image with all detected safety equipment highlighted and labeled. This serves as a proof-of-concept for a real-time safety audit tool.



Live Clock

Sunday, October 05, 2025 - 02:14:53

Configuration

Dark Mode

Confidence Threshold: 0.10

Select Input Source
[✓ Upload Image]
Image from URL
Live Webcam Photo

Choose an image...

Drag and drop file here — Browse files

Space Station Safety Detector

Advanced AI to identify vital safety equipment in images.

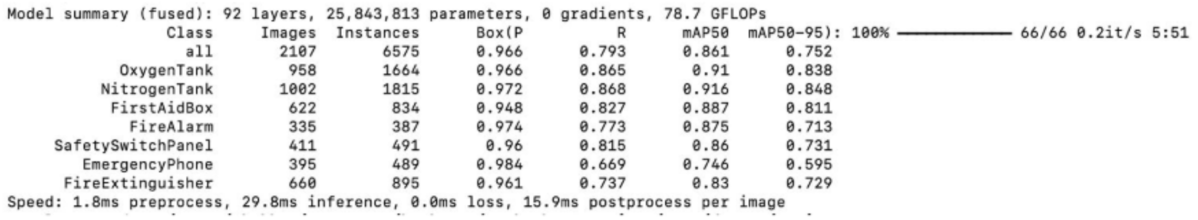
Input

Detection Results

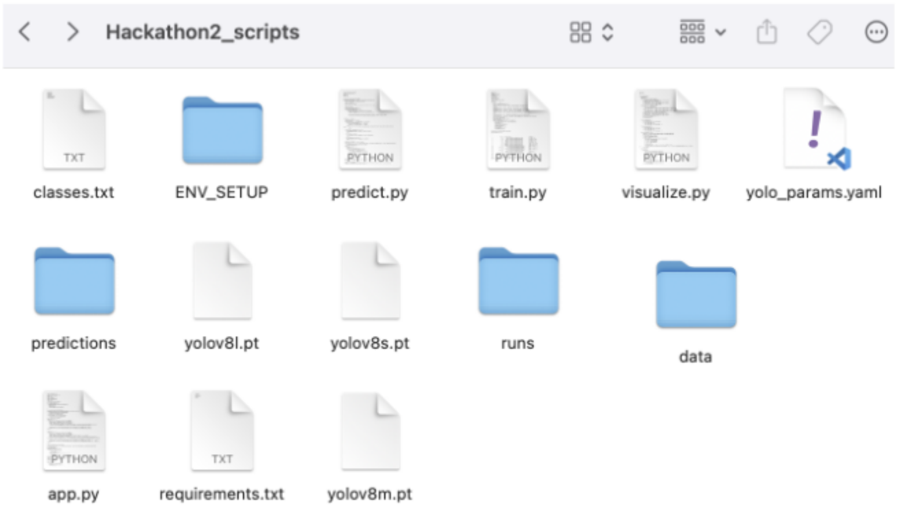
Falcon Update Plan-

crucial aspect of deploying a real-world AI model is maintaining its accuracy as the environment changes. We propose the following plan to keep the model up to date using the Falcon platform

1. **Scenario: An Object's Appearance Changes**
   a. **Problem:** If a new model of 'Fire Extinguisher' is introduced, our current model may fail to detect it
   b. **Falcon Solution:** We would use Falcon to generate a new synthetic dataset featuring this new fire extinguisher under diverse conditions. This data would be added to our training set, and the model would be retrained to learn the new appearance.
2. **Scenario: A New, Confusing Object is Introduced**
   a. **Problem:** If a new object that resembles a 'Firs tAid Box' is brought on board, it could cause false detections
   b. **Falcon Solution:** Using Falcon, we would create challenging scenarios in a digital twin where the new object and the 'First Aid Box' appear together. Training on these "hard cases" would teach the model to differentiate between them, improving its real-world reliability.



```
Model summary (fused): 92 layers, 25,843,813 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 66/66 0.2it/s 5:51
                   all       2107       6575      0.966      0.793      0.861      0.752
            OxygenTank        958       1664      0.966      0.865       0.91      0.838
          NitrogenTank       1002       1815      0.972      0.868      0.916      0.848
           FirstAidBox        622        834      0.948      0.827      0.887      0.811
             FireAlarm        335        387      0.974      0.773      0.875      0.713
     SafetySwitchPanel        411        491       0.96      0.815       0.86      0.731
        EmergencyPhone        395        489      0.984      0.669      0.746      0.595
      FireExtinguisher        660        895      0.961      0.737       0.83      0.729
Speed: 1.8ms preprocess, 29.8ms inference, 0.0ms loss, 15.9ms postprocess per image
```


Hackathon2_scripts
classes.txt
ENV_SETUP
predict.py
train.py
visualize.py
yolo_params.yaml
predictions
yolov8l.pt
yolov8s.pt
runs
data
app.py
requirements.txt
yolov8m.pt