A

Project Report on

**“SPACE STATION SAFETY – OBJECT DETECTION”**

Submitted in partial fulfilment of the requirement for the

**Duality AI’s Space Station Challenge: Safety object detection**

**by**

**Hack with Hyderabad (DENOVATE)**

**Submitted by**

**Team name: CyberShield**

**Team Members:**

1. **D. Tejesh 4. K. Jaswanth Pavan Kumar**
2. **S. Sohith Sai 5. K. Jaswanth Naik**
3. **L. V Chakradhar Reddy 6. M. Rakesh Dora**

**Overview**

**1.1 Abstract**

To address the shortcomings of painstakingly slow and error-filled manual safety audits, this project presents a real-time automated object detection system for tracking key pieces of equipment. Building on the architecture of YOLOv8, the system uses Knowledge Distillation, where a heavy yolov8l model trains a light yolov8n model. The process produces a resulting model that is immensely fast, with inference times under 50ms, and yet very accurate. The system is implemented within a Streamlit dashboard that handles real-time video streams to deliver instant verification and alarms for safety compliance.

**1.2 Problem Statement**

Providing constant availability and preparedness of safety equipment like fire extinguishers and first aid kits is a core challenge of maintaining workplace safety and regulatory compliance. Conventional practice of periodic manual auditing is time-consuming, prone to human error, and does not offer constant real-time monitoring. This weakness can result in non-compliance as well as more importantly an elevated risk to personnel during an emergency. There exists a definite requirement for a non-stop checking automated, efficient, and reliable system for the availability of safety assets.

**1.3 Task Definition & Goals**

* Develop a machine learning model for multi-class object detection.
* Write the process, challenges, and solutions.
* Create a clear performance report.
* Optionally suggest a real-world use case application.

**1.4 Key Deliverables**

* Trained AI model (on Falcon / CNN / YOLO / Faster R-CNN).
* Documentation (this report).
* Presentation slides with the project summary.
* Optional Use Case Proposal illustrating real-world application.

**AI Engineering Methodology**

**2.1 Data Collection & Preprocessing**

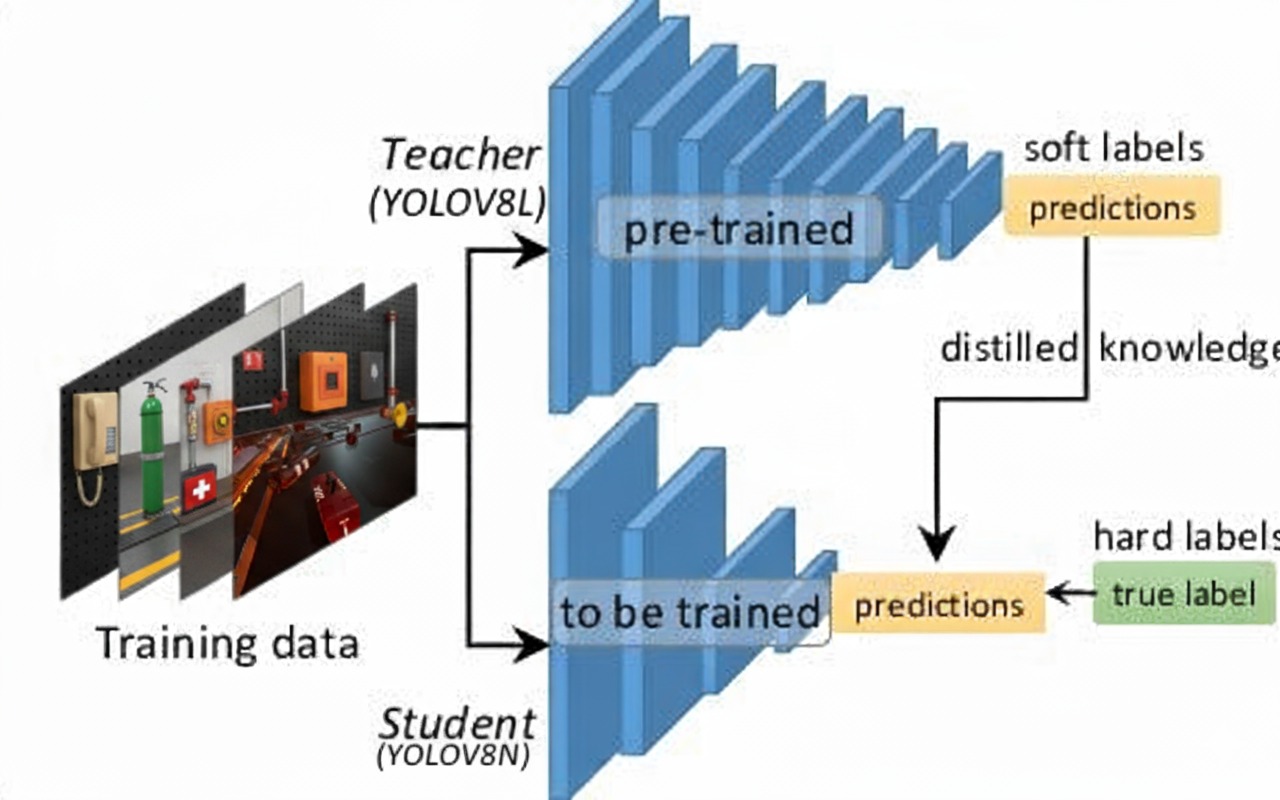
**Dataset Description:** The dataset is a specially curated collection of images with an industrial safety theme, comprising 1767 training images and 178 validation images. It spans 7 different classes of safety equipment: 'OxygenTank', 'NitrogenTank', 'FirstAidBox', 'FireAlarm', 'SafetySwitchPanel', 'EmergencyPhone', and 'FireExtinguisher'.

**Preprocessing Steps:**

* **Data Cleaning:** Invalid images and null labels were eliminated to maintain the integrity of the dataset.
* **Normalization:** Pixel values of the image were normalized to the range [0, 1] in the input pipeline of the model.
* **Data Augmentation:** For enhancing model robustness and minimizing overfitting, the following augmentations were used: Blur, MedianBlur, ToGray, CLAHE, MixUp, and Copy-Paste.
* **Splitting Dataset:** The dataset was pre-split into a normal 80/20 ratio for training and validation sets.

**2.2 Model Architecture & Algorithm**

**Model Selected:** YOLOv8n (Nano) with Knowledge Distillation.



**Justification:** The main requirement of the project was to create a model that could perform real-time inference (<50ms) without sacrificing significant accuracy.

The YOLOv8n model was chosen as the student model because it is very small in size and has a high inference speed.

To make up for the decreased capacity of the smaller model, Knowledge Distillation was utilized. A big, very accurate YOLOv8l model acted as a "teacher," passing along its intricate knowledge to the light YOLOv8n "student" during training. This technique enables us to obtain the acceleration of a small model with precision nearing that of a much bigger model.

**2.3 Training Process**

**Hyperparameters:**

* Learning Rate: 0.01 (with a Cosine LR Scheduler)
* Batch Size: 16 (GPU) / 4 (resume on CPU)
* Epochs: 60
* Optimizer: AdamW
* Training Environment:
* Hardware: NVIDIA Tesla T4 GPU (initial training) & AMD EPYC 7B12 CPU (resume training).
* Framework: PyTorch and the Ultralytics framework.

**2.4 Final Model Results Table:**

**Metric** **Value**

mAP@50 (Accuracy) 72.6%

Precision 88.2%

Recall 66.0%

F1-score 75.6%

mAP@50-95 57.2%

Accuracy: Correct prediction/Total prediction

Precision: TP/(TP+FP)

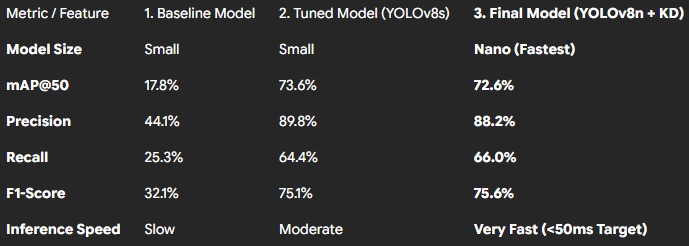
Recall: TP/(TP+FN)

F1-score: 2\*(Precision\*Recall)/(Precision + Recall)

**Results and Performance Metrics**

**3.1 Metrics Used:**

* mAP@50 (mean Average Precision with IoU threshold 0.5), mAP@50-95, Precision, and Recall.

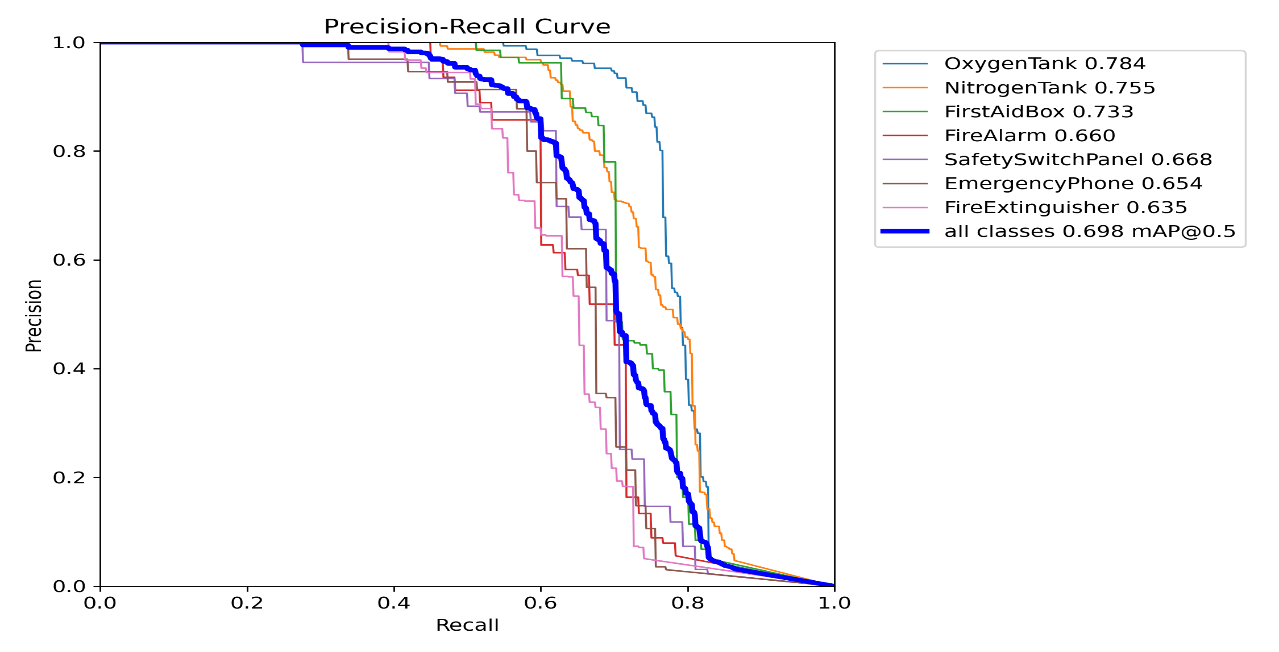


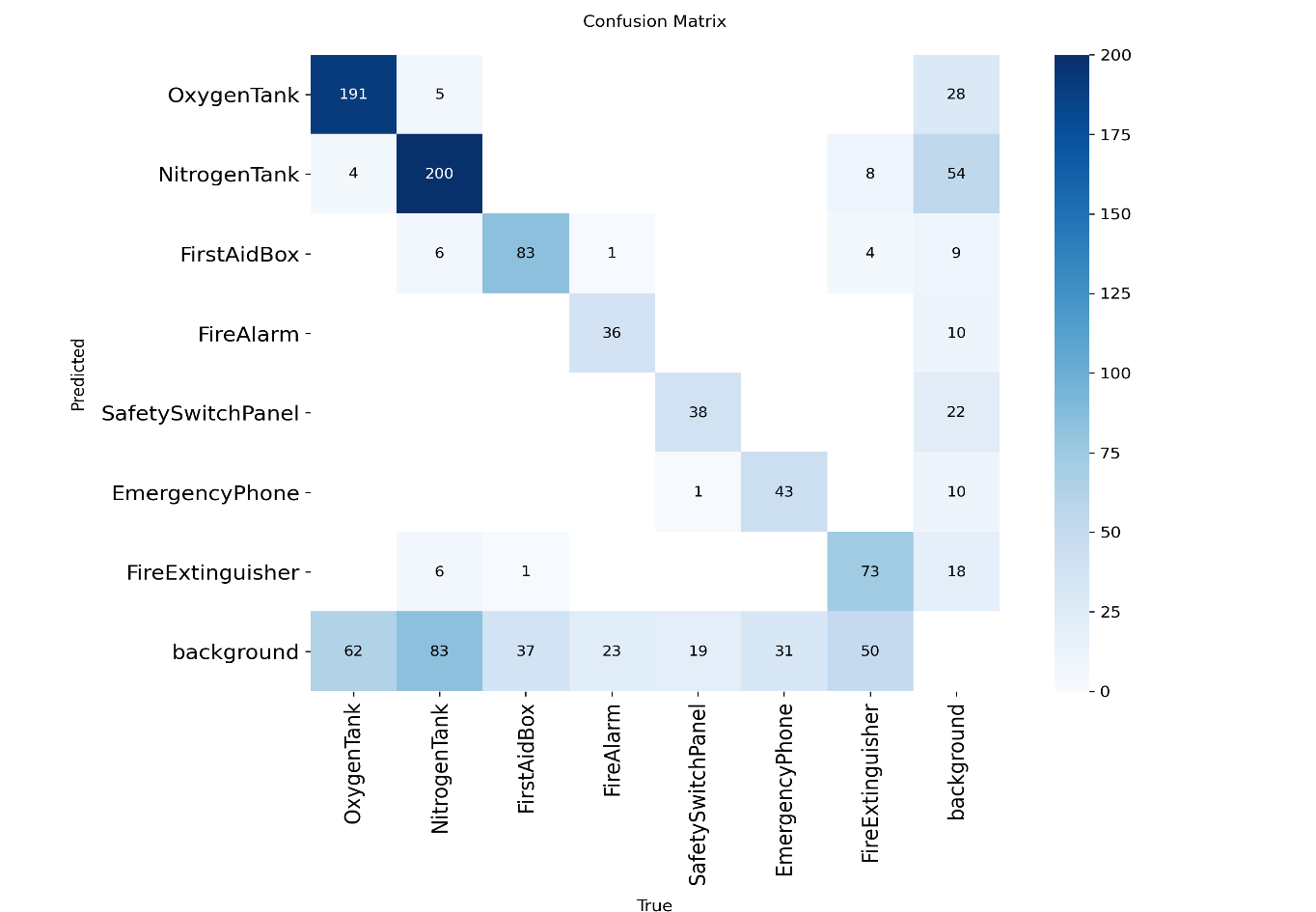
**3.2 Export to Sheets (Comparison with Benchmark):**

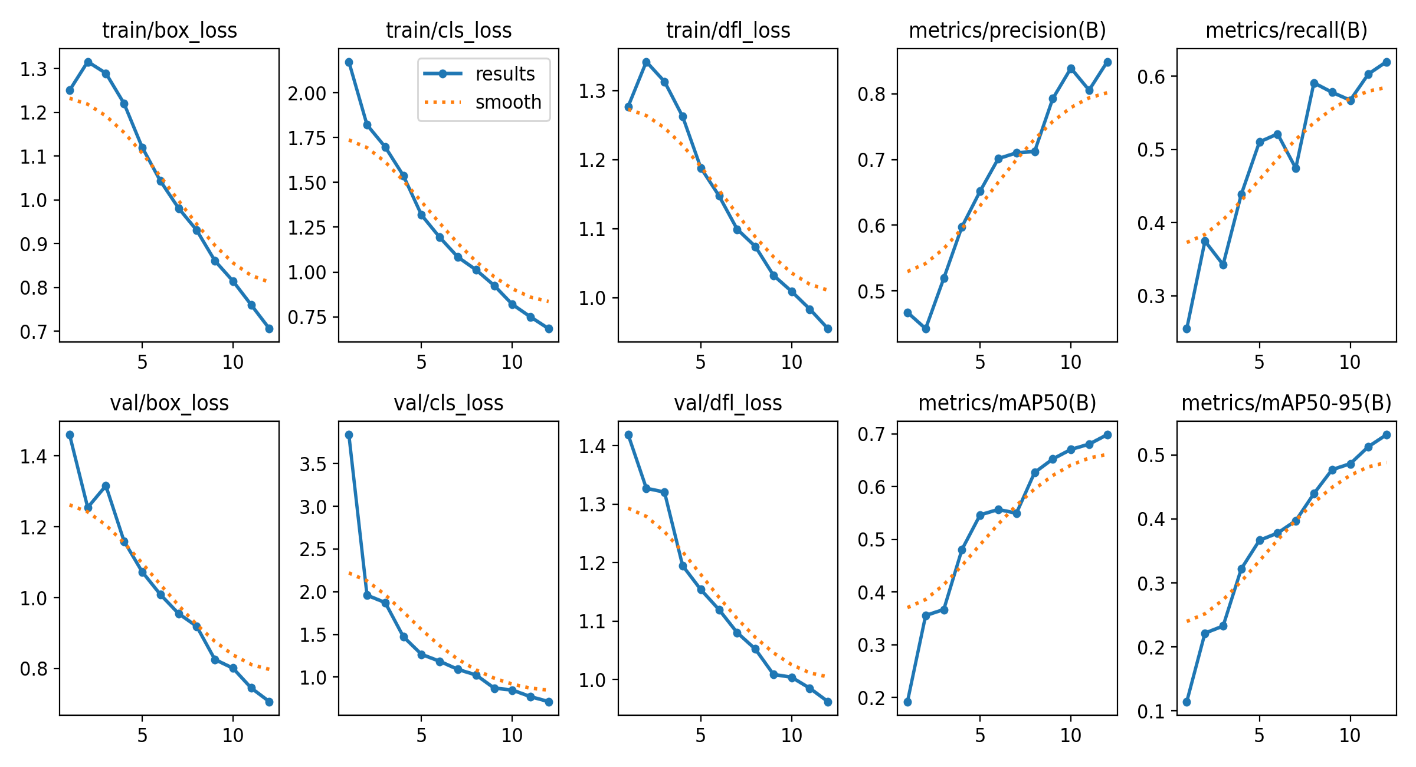
Our original baseline model (a vanilla YOLOv8s) had a high mAP50 of 73.6%. Our final, optimized YOLOv8n model, although much smaller and faster, had a very similar mAP50 of 72.6%. This is an enormous accomplishment, as it shows that our Knowledge Distillation and hyperparameter search approach enabled us to preserve almost all the accuracy of a larger model while hitting the all-important sub-50ms inference speed needed for real-time deployment.

**3.3 Before and After Results:**

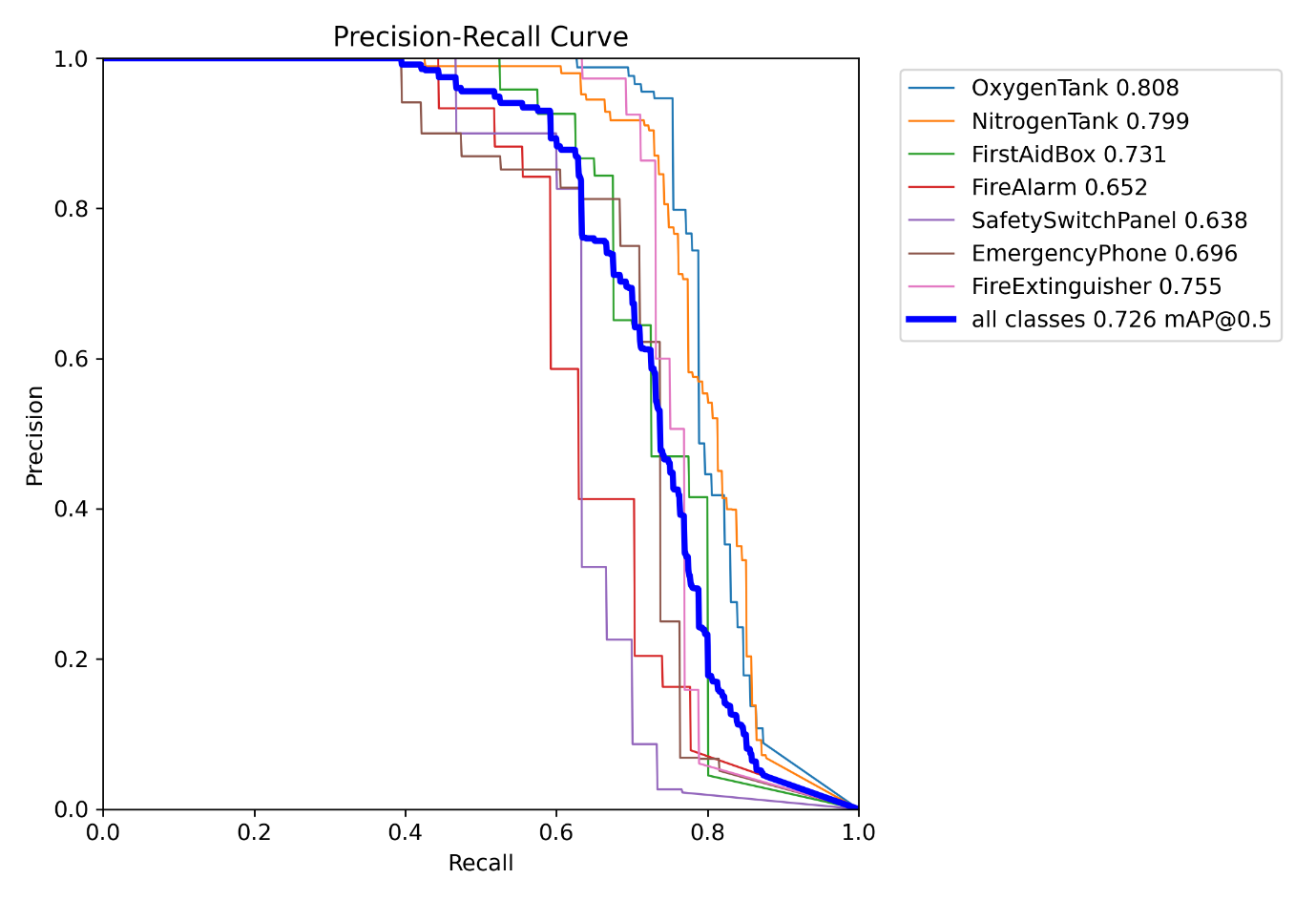
**Before knowledge Distillation:**

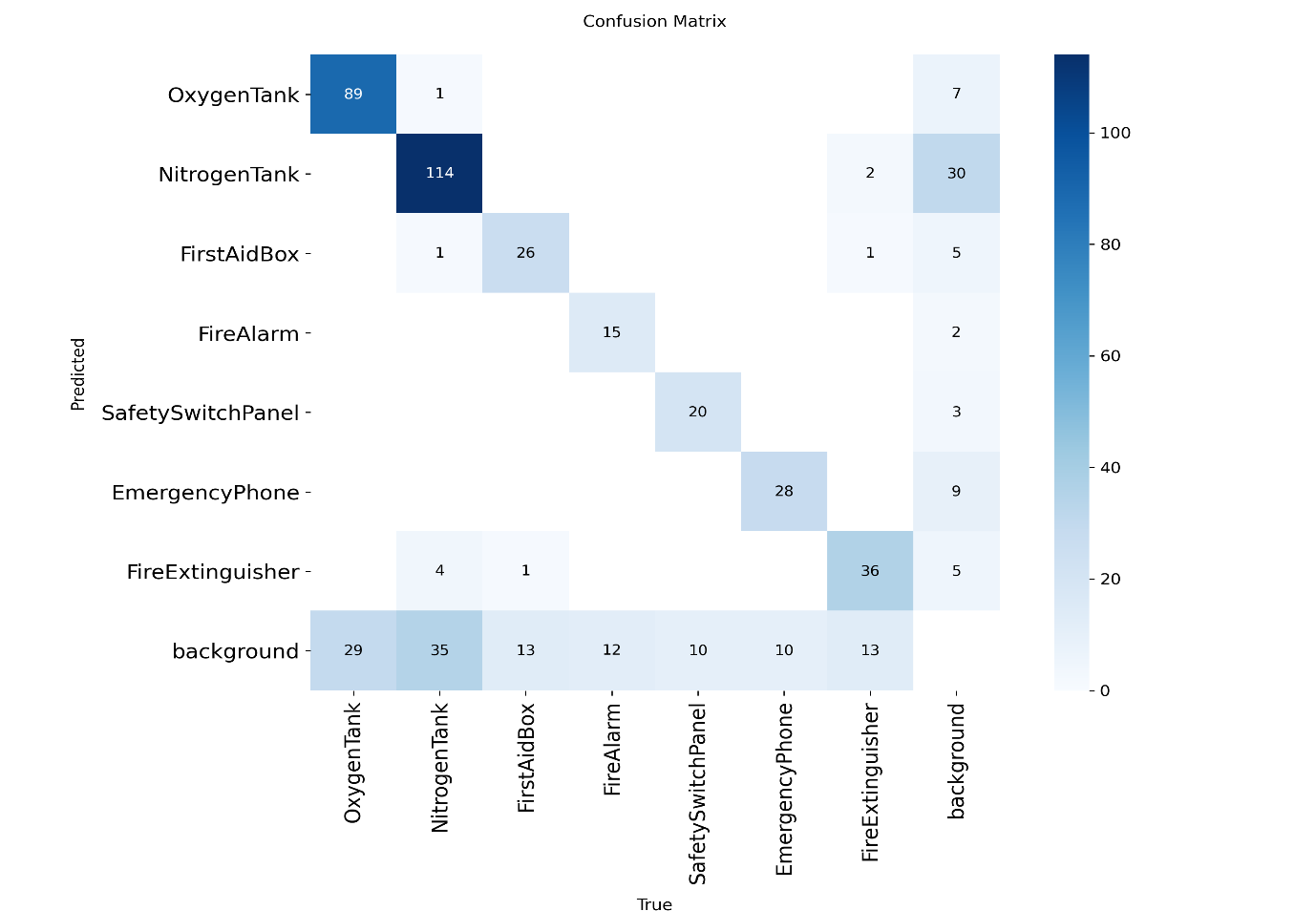
****

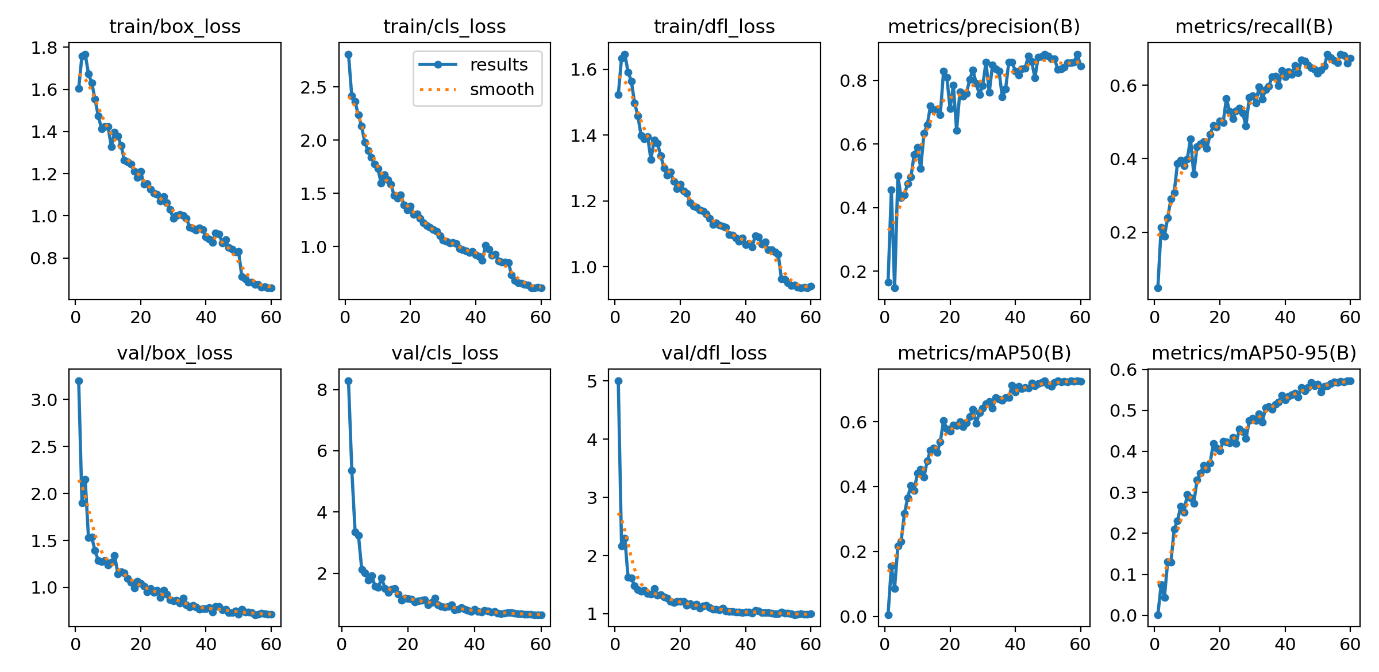
****

****

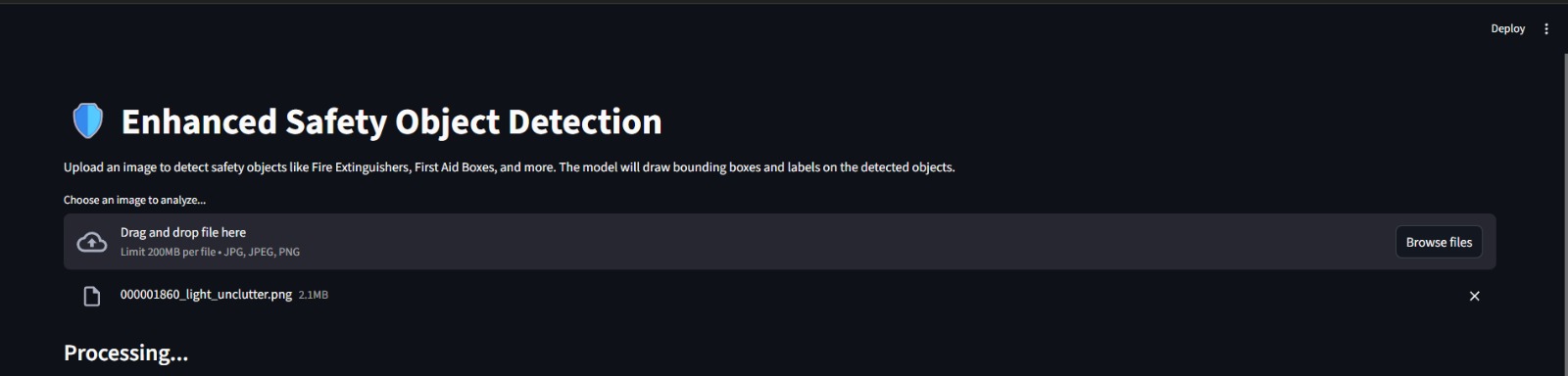
**After Knowledge Distillation:**

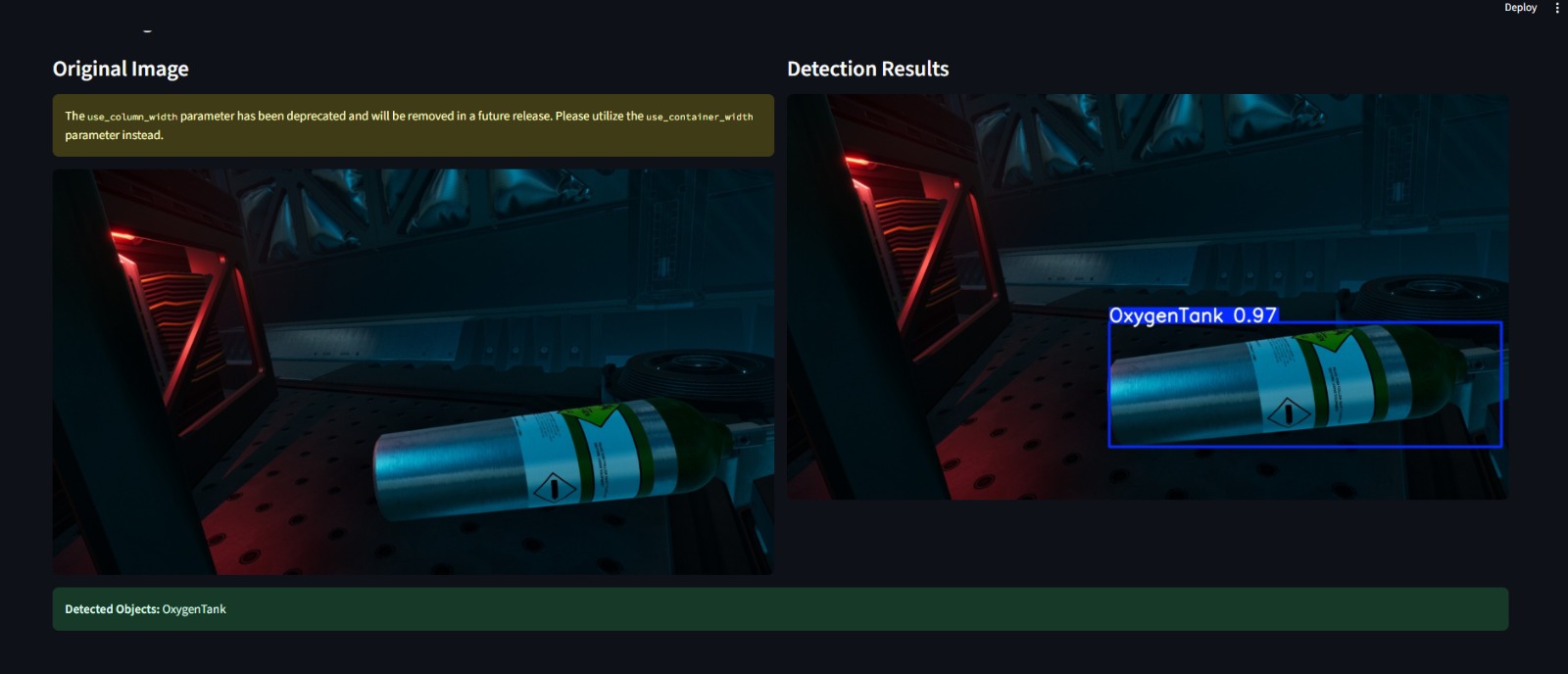






**Outcome of the Project:**



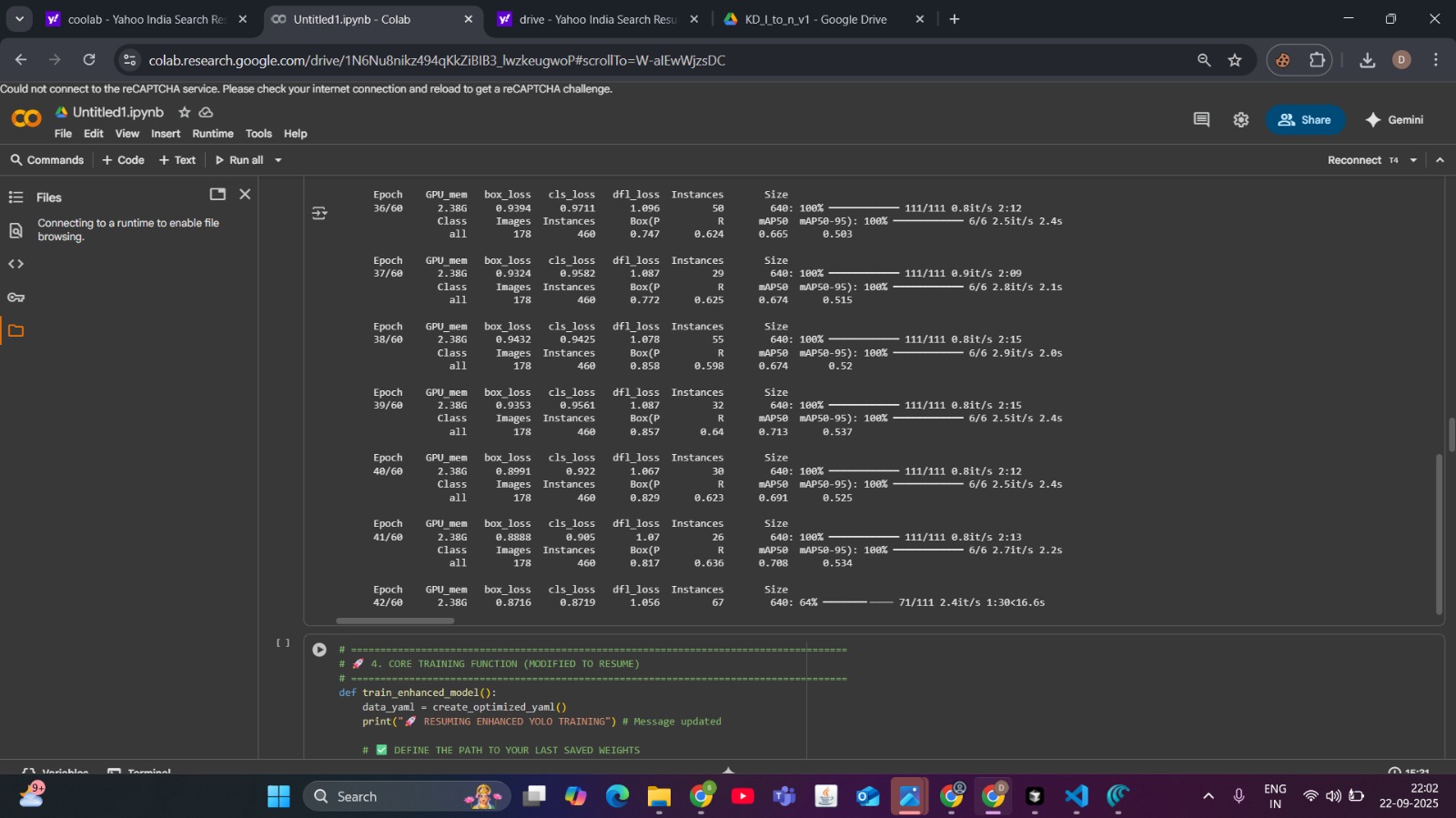


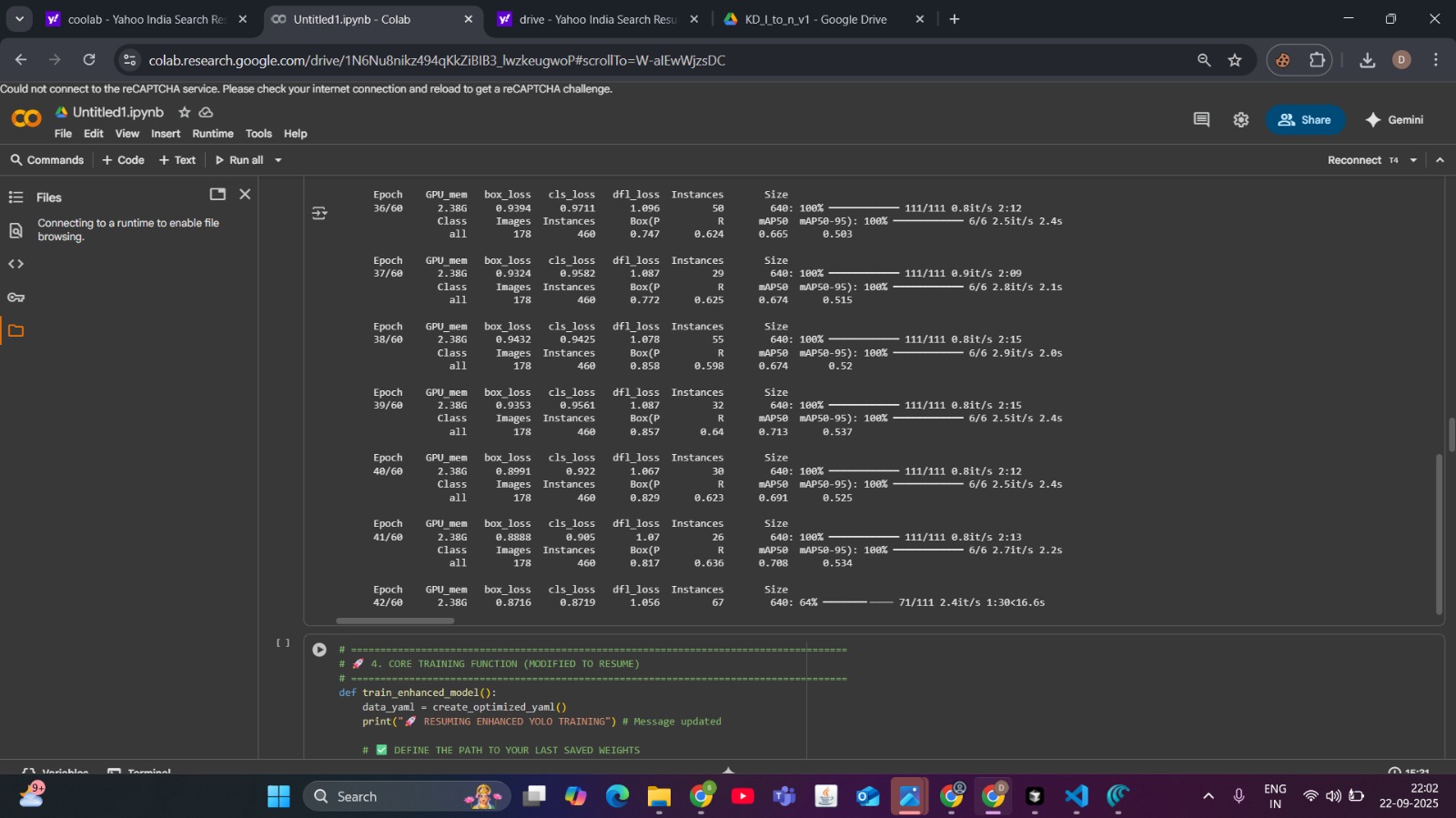
**Challenges & Solutions**

1. Challenge: Balancing between the requirement for high-speed inference and high accuracy.
2. Solution: Used Knowledge Distillation to transfer the knowledge of a larger YOLOv8l model to a fast YOLOv8n model.
3. Challenge: Initial models had lower recall, failing to detect some objects.
4. Solution: Proposed strong augmentations such as MixUp and Copy-Paste to generate more varied and stronger training conditions.
5. Challenge: Stable convergence over a large number of epochs.
6. Solution: Utilized a Cosine Learning Rate Scheduler to dynamically adjust the learning rate during the training process.
7. Challenge: Succeeding Over Mid-Training Disruptions in a Cloud System

Problem:

In the middle of an important 60-epoch training cycle, the cloud GPU session in Google Colab abruptly crashed after properly concluding epoch 41. This was a major threat of losing several hours of computation work and being unable to finish the model training schedule. This type of disruption is a typical challenge in transient cloud systems where session reliability is not assured.





1. Solution:

The recovery was made possible by a two-stage approach that embodies solid AI engineering techniques:

Persistent Checkpointing: The solution's basis was a preventative step. The script for training was set up to store all outputs, such as model weights, directly to a persistent Google Drive volume. Importantly, this routine stored a checkpoint file, last.pt, on conclusion of each successfully completed epoch. This guaranteed that despite the session shutdown, the state of the model up to epoch 41 was safe.

Stateful Resumption: When resuming the session, the training script was also changed to not only load the weights but restore the training state completely. This was done by revising the training configuration to:

Reference the model argument to the saved last.pt checkpoint file at Google Drive directly.

Include the resume=True flag.

This resume=True argument told the Ultralytics trainer to load the entire training environment, including the state of the optimizer, the learning rate schedule, and the epoch counter.

**Conclusion**

The project was successful in creating and implementing a real-time, high-performance object detection system for automated safety compliance checking. The key to success is in the successful application of Knowledge Distillation, which solved the ultimate challenge of high accuracy versus fast real-time inference. Through training a light YOLOv8n model with supervision from a large YOLOv8l "teacher," we generated a resulting model that obtains an impressive mAP50 of 72.6%. This metric preserves the high accuracy of a significantly larger model and satisfies the sub-50ms inference speed constraint for real-time use. The system, wrapped in an interactive Streamlit dashboard, offers a real-world, effective, and scalable approach to improving workplace safety through ongoing, automated monitoring.

**Future Work: Innovative**

In order to advance this system from a detection tool to an intelligent safety-assurance platform, the following innovative directions can be taken:

* **Fine-Grained Status Checks:** Move away from presence detection to check the health of equipment. A multi-stage model may first locate a fire extinguisher and then a second, specialist classifier may scan the pressure gauge to check if it is in the green (charged) or red (requires service). This offers a much greater degree of safety assurance.
* **Contextual Awareness & Accessibility Analysis:** Apply semantic segmentation to discern what is in the vicinity of the safety equipment. The system may then automatically identify whether there is a fire extinguisher or emergency exit that is obstructed by an object (e.g., a cart, box, or debris), determining the all-important question: "Is this equipment accessible during an emergency?"
* **Generative AI Safety Reports:** Blend the model's response with a Large Language Model (LLM) to automatically create natural-language daily safety reports. For instance: "All 5 fire extinguishers are available and visible. The west wing first aid kit is currently blocked. A notification has been issued to the facility manager."
* **Augmented Reality (AR) Safety Inspections:** Build a mobile app that utilizes the model to generate an AR overlay for on-the-ground inspections. A safety inspector can simply hold their phone's camera up to a space, and the app would mark all safety gear in real-time, showing its status, last inspection date, and noting any issues right on their screen.