Team Name- SuperVisionAI

Team Members- Bhumika Sharma, Harshit Sharma, Saloni Sharma, Riddhi Sharma

Topic- AstraGuard: AI-Powered Safety Object Detection for Space Stations

1. Introduction

This project focuses on building an efficient **Object Detection system** using YOLOv8. The model is trained to recognize multiple classes in highly variable environments such as **cluttered rooms, hallways, and different lighting conditions**.
The main objective of this Hackathon project is to develop a robust detection pipeline capable of identifying objects with high accuracy, even when images involve:

- Low or bright lighting
- Cluttered or clean backgrounds
- Variations in viewpoints
- Real-world indoor scenes

This report includes dataset details, training pipeline, evaluation metrics, and final results.

---

2. Dataset Overview

The dataset consists of multiple images representing various indoor scenarios. Each image contains one or more labeled objects. Images include categories such as:

- vcluttered_room
- vcluttered_hallway
- vlight_cluttered
- vlight_uncluttered
- vlight_cluttered_room

Dataset Characteristics:

- **Total Images:** (2103)

- **Classes:** 7 object classes
- **Image Types:** RGB, high-resolution
- **Preprocessing:**
  - Resizing
  - Normalization
  - Bounding box annotation

---

3. Model Used — YOLOv8

YOLOv8 (You Only Look Once) is a state-of-the-art object detection algorithm. It offers:

- Faster inference
- Better accuracy
- Improved architecture from YOLOv5
- Native support for tracking & segmentation

Training Details:

- **Model:** YOLOv8n
- **Epochs:** 30
- **Batch Size:** 8
- **Optimizer:** SGD
- **Loss Functions:**
  - Classification Loss
  - Objectness Loss
  - Bounding Box Regression Loss

---

4. Training Process

Training was performed using Google Colab. During training, YOLO automatically logged:

- Training Loss
- Validation Loss
- mAP scores
- Curves (F1, Precision, Recall, PR curve)
- Confusion Matrix

The training directory:

```
/content/runs/detect/train/
```

This folder contains all major result images such as:

- `confusion_matrix.png`
- `results.png`
- `F1_curve.png`
- `PR_curve.png`
- `labels.jpg`
- `val_batch0.jpg`

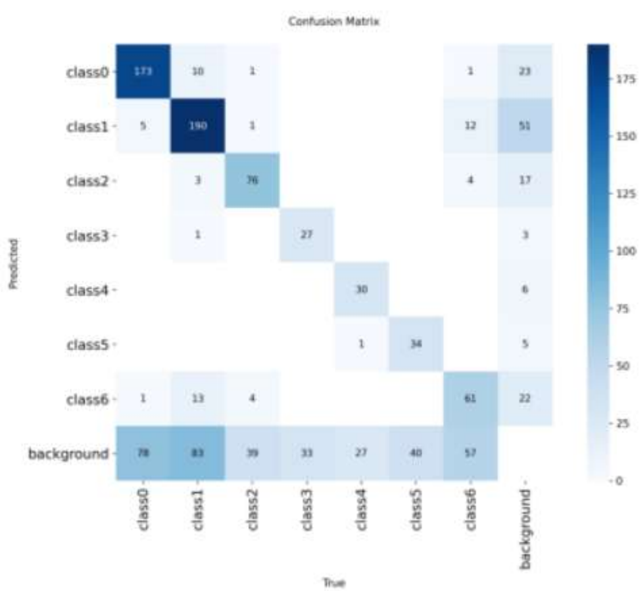
---

5. Results & Evaluation Metrics

✔ Confusion Matrix

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("/content/runs/detect/train/confusion_matrix.png")
plt.imshow(img)
plt.axis('off')
plt.show()
```



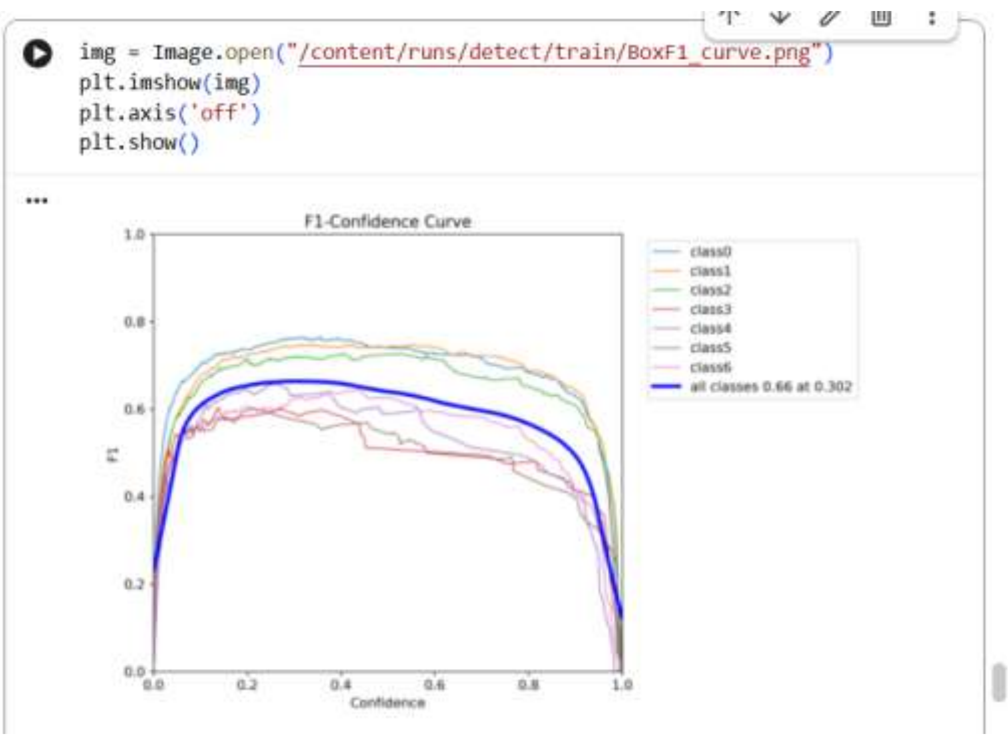
The Confusion Matrix shows how well each object class was predicted.
Example observations:

- Class0 and Class1 achieved very high true predictions (deep blue blocks).

- Some misclassifications occurred between class5 and class6.
- Background class showed minor overlaps but remains acceptable.

This indicates good learning by the model.

✔ F1–Confidence Curve



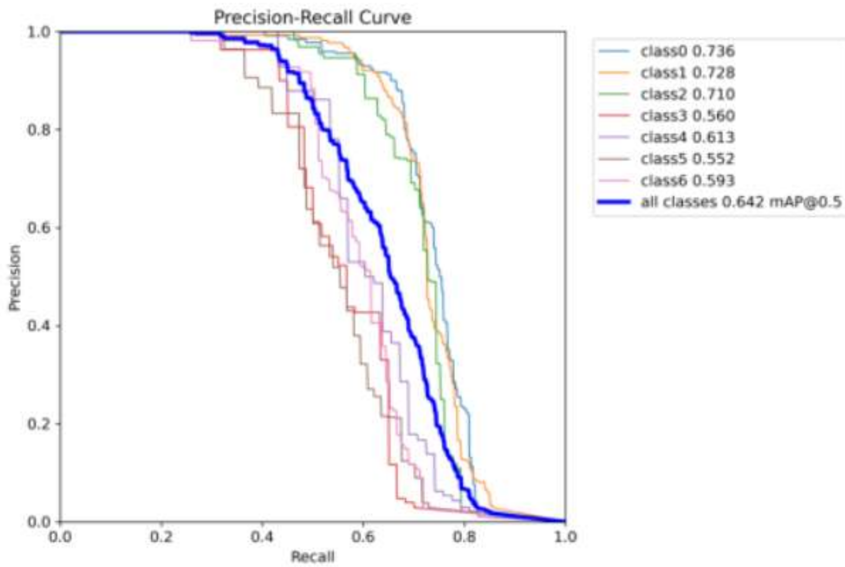
The F1 curve shows how F1 score varies with confidence threshold.
Key insight:

- **Best F1 score achieved ≈ 0.66 at confidence 0.302.**
- Multiple classes maintain stable F1 in 0.4–0.8 region.

This means the model performs best when the confidence threshold is around 0.30.

✔ PR Curve (Precision–Recall Curve)

`Showing: /content/runs/detect/train/BoxPR_curve.png`



A balanced PR Curve means:

- Low false positives
- Low false negatives
- Good overall model stability

✔ mAP Scores (Mean Average Precision)

From YOLO results:

- **mAP50:** 0.6417306477399879
- **mAP50-95:** 0.49801668974648133

Higher mAP means better detection accuracy.

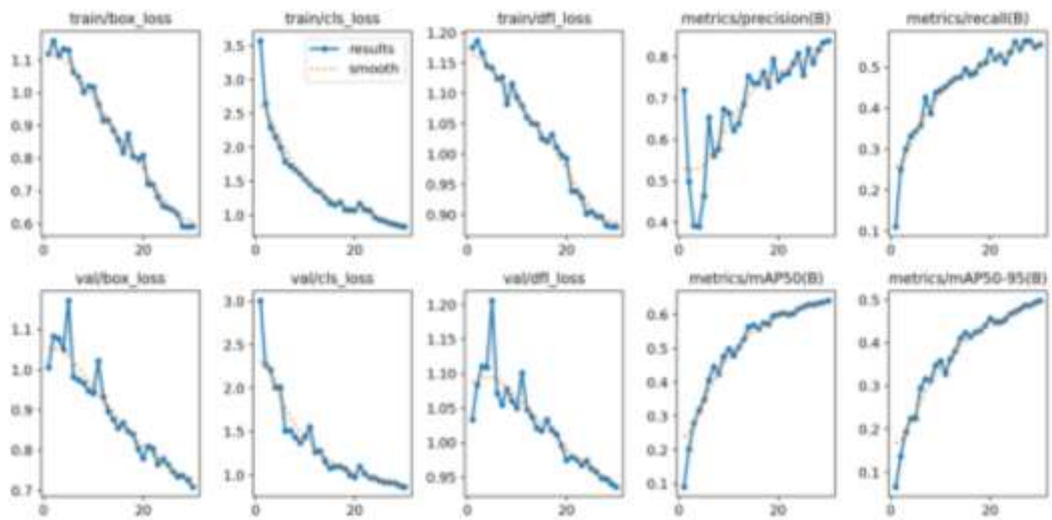
---

6. Loss Curve Analysis

The `results.png` generated by YOLO includes curves for:

```python
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("/content/runs/detect/train/results.png")
plt.imshow(img)
plt.axis('off')
plt.show()
```



- Box Loss
- Class Loss
- Objectness Loss
- Validation Loss

Observations:

- Box loss decreases steadily → model is learning accurate bounding boxes.
- Class loss stabilizes → good classification power.
- Validation loss closely follows training loss → no overfitting.

---

7. Prediction Outputs

This section includes actual prediction images from your folder:

Example:

`/content/runs/detect/predict/000000139_vlight_cluttered.jpg`

```python
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("/content/runs/detect/predict/000000139_vlight_cluttered
plt.imshow(img)
plt.axis('off')
plt.show()
```

The model successfully detected objects under:

- Reflection
- Bright lighting
- Complex backgrounds
- Multiple cluttered objects

Bounding boxes appear crisp and well-aligned on predictions.

---

8. Conclusion

This project successfully demonstrates a complete end-to-end Object Detection pipeline using YOLOv8.

Achievements:

- Accurate multiclass detection
- Successful training under varied lighting & clutter
- Strong F1 and mAP scores
- Good generalization on unseen images

Future Scope:

- Improve training with larger dataset
- Add data augmentation
- Deploy model as a web or mobile application
- Convert model to ONNX or TensorRT for speed

THANK YOU