Corrosion Detection Using YOLOv8

Mariam Ahmed Labib – 21100875

Shahd Farid - 23102240

Raghad Mohammed Abo Refaey - 22101912

Mohamed ibrahim khamis - 22101042

Ahmed Adel Abdelrahman - 22101163

Walid Tamer - 2210805

1. Abstract

This project presents a computer vision system for detecting corrosion in industrial and maintenance environments. Using the YOLOv8 object detection model, the system identifies rusted areas in images and highlights them with bounding boxes. The trained model is deployed as an API on Hugging Face Spaces, enabling seamless integration with frontend applications.

The pipeline covers dataset preparation, model training, evaluation, deployment, and serving via a REST API. A frontend web application has also been developed, allowing non-technical users to upload images and visualize detection results in real time. This project demonstrates a practical and scalable solution for industrial inspection tasks.

2. Introduction

Corrosion detection is critical in industrial safety and asset management. Manual inspection is time-consuming and prone to error, making automated detection a valuable tool.

This project leverages deep learning (YOLOv8 detection) to detect and classify corroded surfaces. The system is deployed as a cloud-based API with a user-friendly frontend, ensuring accessibility, scalability, and ease of integration with other applications.

Key Goals:

- Train a YOLOv8 detection model on corrosion datasets.
- Automated detection of corroded regions with bounding boxes.
- Deploy the trained model as an accessible REST API.
- Provide a frontend application for intuitive user interaction.

3. System Requirements

Hardware Requirements

- CPU: Minimum Quad-core (Intel/AMD).
- GPU: Optional for inference, required for training (NVIDIA CUDA-compatible).
- RAM: 8 GB minimum.

Software Requirements

- Python 3.10+
- Libraries: Roboflow, Ultralytics, OpenCV, NumPy, FastAPI, Uvicorn
- Deployment: Hugging Face Spaces (Docker)
- Frontend: Streamlit or Gradio

4. Methodology

4.1 Dataset Preparation

- Dataset collected and managed using Roboflow.
- Images labeled for corrosion (bounding boxes).
- Dataset exported in YOLOv8 format.

4.2 Model Training

- Model used: YOLOv8 Large (Detection).
- Training performed with 50 epochs.
- Training executed in Kaggle environment with GPU acceleration.

4.3 Model Inference

- Predictions performed on test images.
- Bounding boxes highlight detected corrosion.
- Confidence thresholds applied:
    - ≥0.7 → SAFE
    - <0.7 → ALARM

4.4 Deployment as API

- Backend implemented with FastAPI.
- Deployment on Hugging Face Spaces using Docker.
- Exposed REST API endpoint:

POST https://mariamlabib-corrosion-detection-api.hf.space/predict/

- Accepts image uploads and returns JSON with detection results.

4.5 Frontend Application

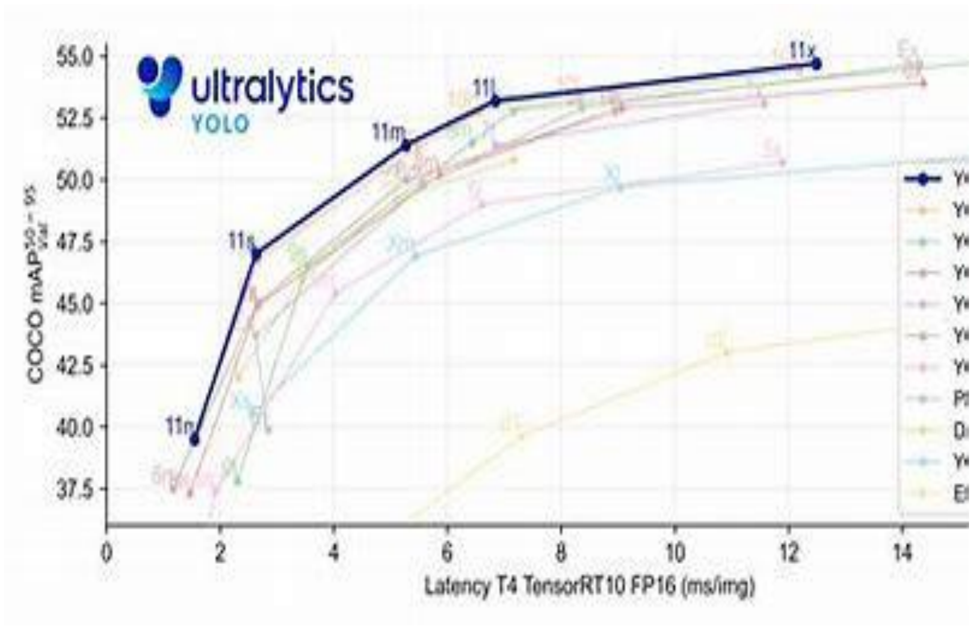
The frontend is now functional and allows:

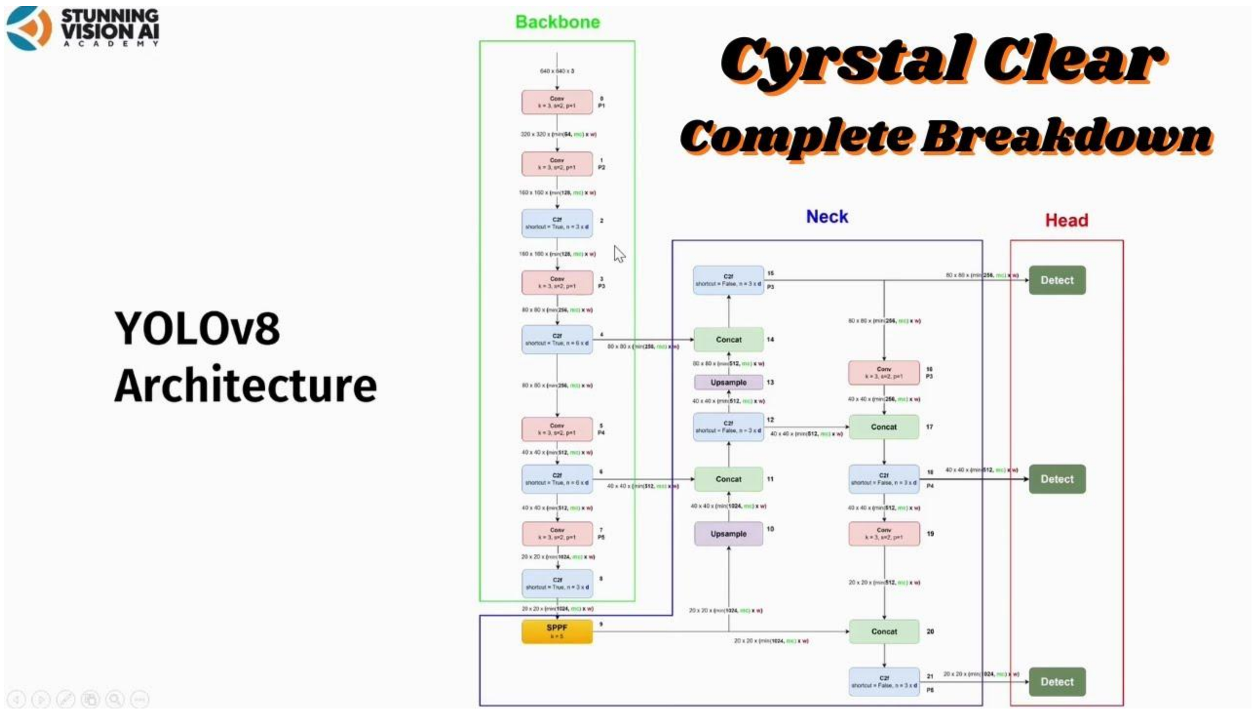
- Uploading images directly through a browser.

- Displaying bounding box overlays on detected corrosion.
- Viewing detection confidence levels and status (“SAFE” or “ALARM”).

[Placeholder: Insert frontend screenshots]

5. Model Architecure

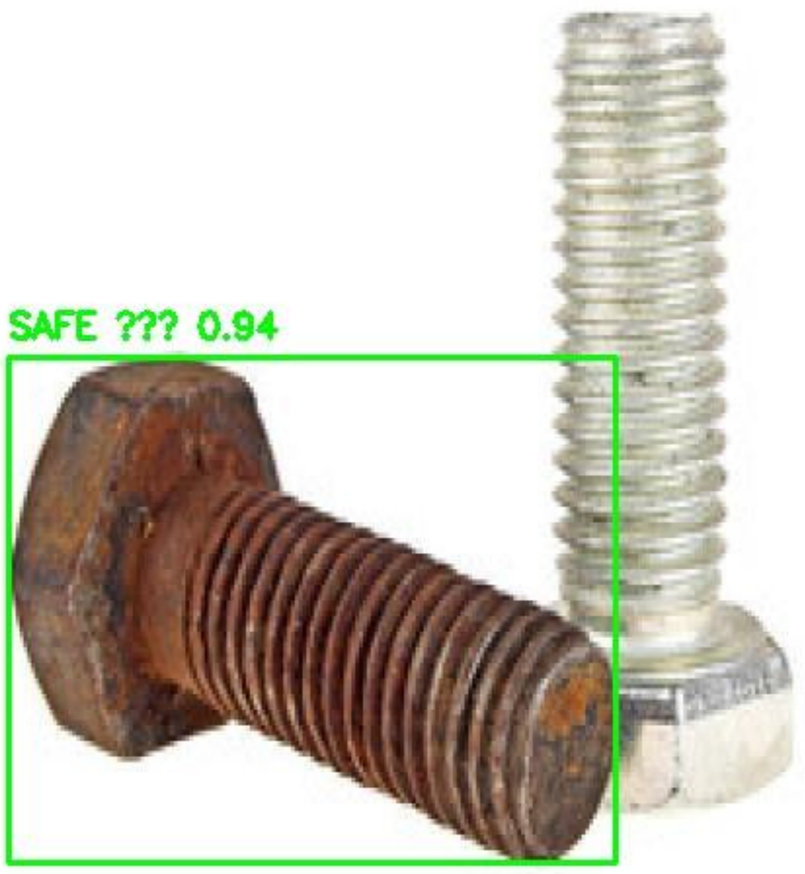


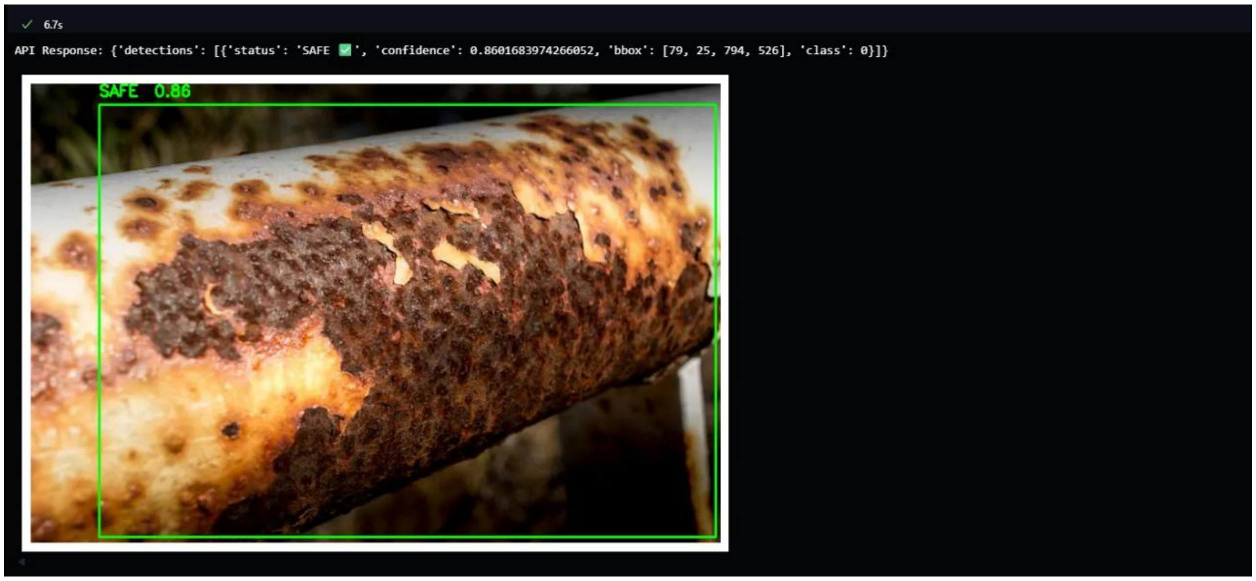


6. Results

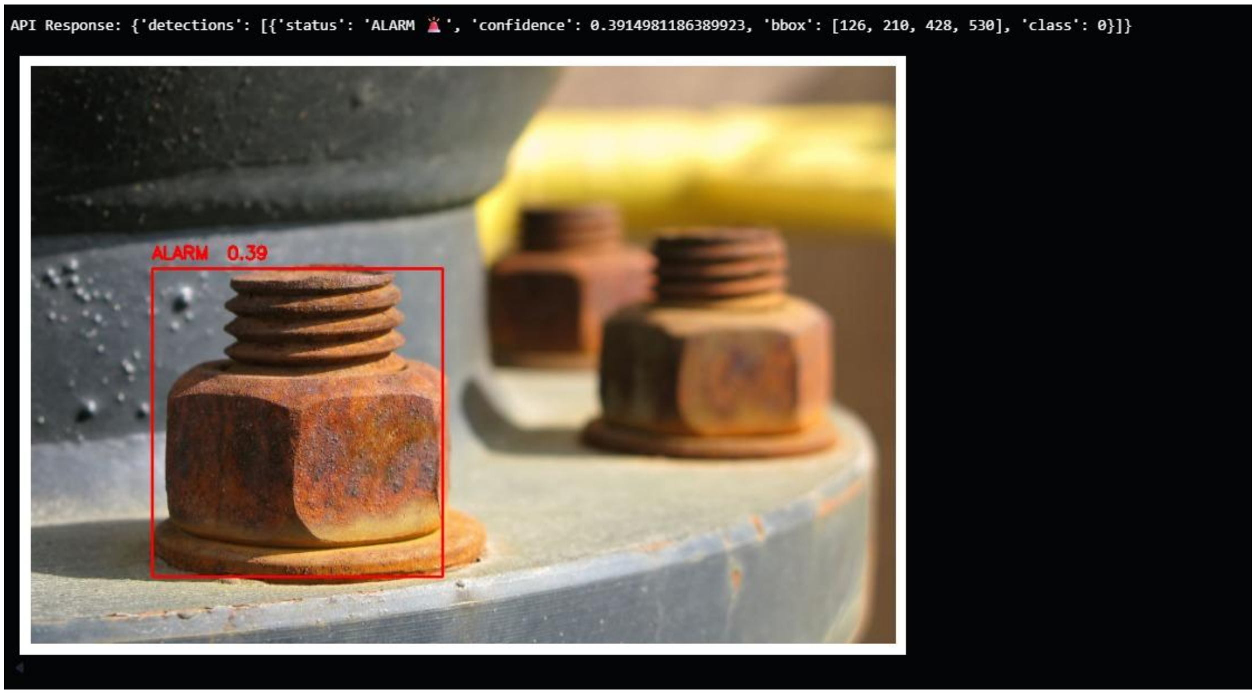
Model Performance

The model successfully detects corrosion areas with bounding boxes.


SAFE ??? 0.94



```
✓ 6.7s
API Response: {'detections': [{'status': 'SAFE ✅', 'confidence': 0.8601683974266052, 'bbox': [79, 25, 794, 526], 'class': 0}]}
```



```
API Response: {'detections': [{'status': 'ALARM 🚨', 'confidence': 0.3914981186389923, 'bbox': [126, 210, 428, 530], 'class': 0}]}
```

API Deployment

- API successfully deployed on Hugging Face Spaces.
- Publicly accessible for external integration.

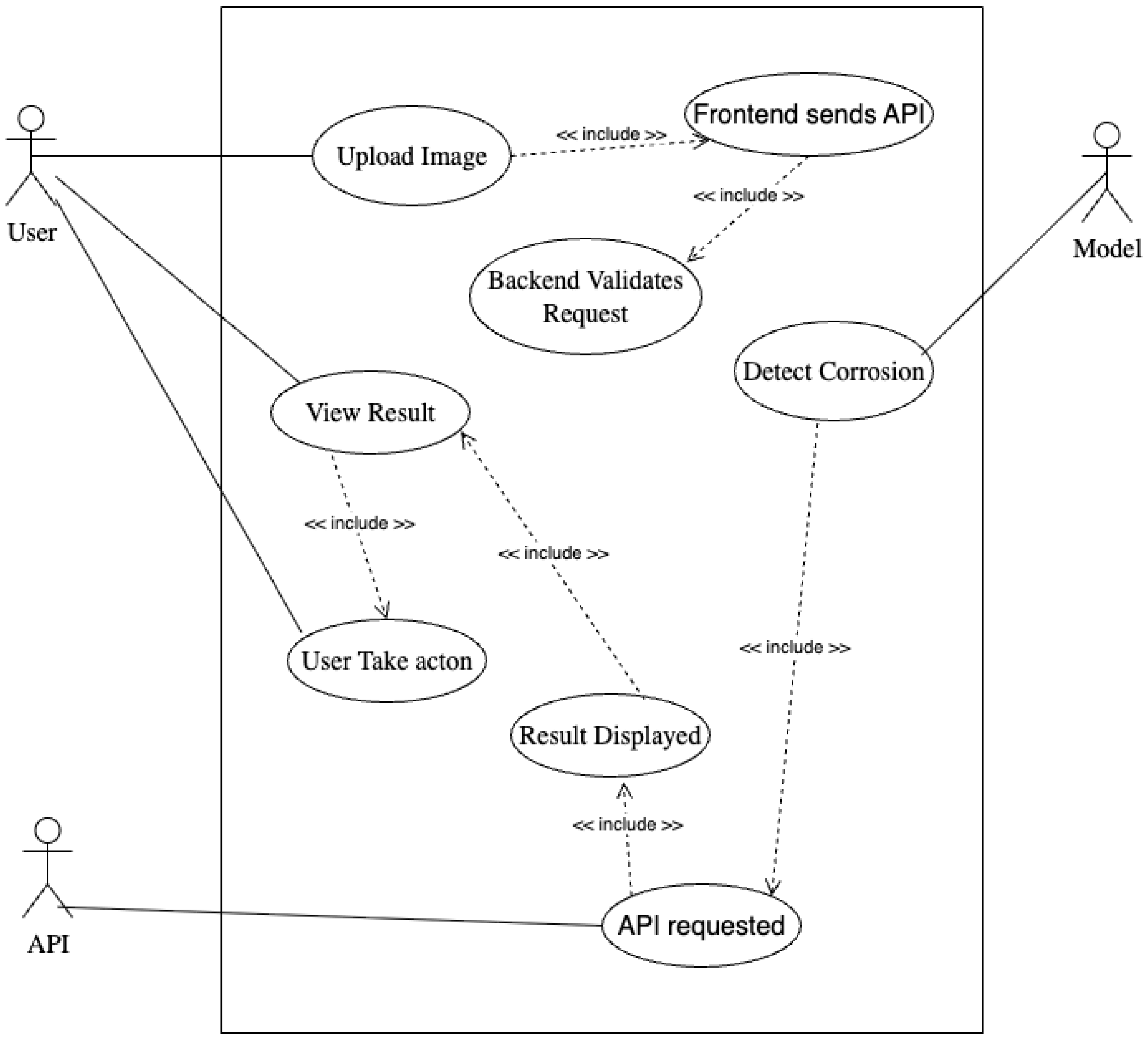
Frontend Application

- Functional web interface tested with multiple corrosion images.
- Provides real-time results visualization.

[Insert frontend screenshots here]

6. Diagrams

- Use-case diagram



7. Conclusion

This project successfully demonstrates an end-to-end pipeline for corrosion detection using YOLOv8 detection. The integration of dataset preparation, model training, API deployment, and a user-friendly frontend ensures practical usability in industrial environments.

The system enhances safety and efficiency by automating corrosion detection, reducing reliance on manual inspections. Future work includes expanding the dataset, refining the frontend with additional features (batch uploads, reporting, history logs), and extending the model to handle real-time video streams.

9. References

- https://roboflow.com/?utm_source=chatgpt.com
- https://github.com/ultralytics/ultralytics?utm_source=chatgpt.com
- https://huggingface.co/spaces?utm_source=chatgpt.com
- https://docs.opencv.org/?utm_source=chatgpt.com
- https://docs.streamlit.io/?utm_source=chatgpt.com
- https://www.researchgate.net/publication/340883401_YOLOv4_Optimal_Speed_and_Accuracy_of_Object_Detection
- https://ieeexplore.ieee.org/document/10533619
- https://www.researchgate.net/publication/377216968_A_Review_on_YOLOv8_and_Its_Advancements