Bullseye Detection using YOLOv8 and ONNX

Saksham Khandelwal

May 2025

Outline

1. Introduction
2. Dataset Preparation
3. Annotation
4. Model Training
5. Results
6. Model Export
7. Inference Demo
8. Challenges and Solutions
9. Project Timeline
10. Conclusion and Future Work

Introduction & Aim

- Real-time bullseye detection
- Lack of specialized detectors for bullseye patterns
- Goal: build a robust, accurate, and fast detection system

Objectives

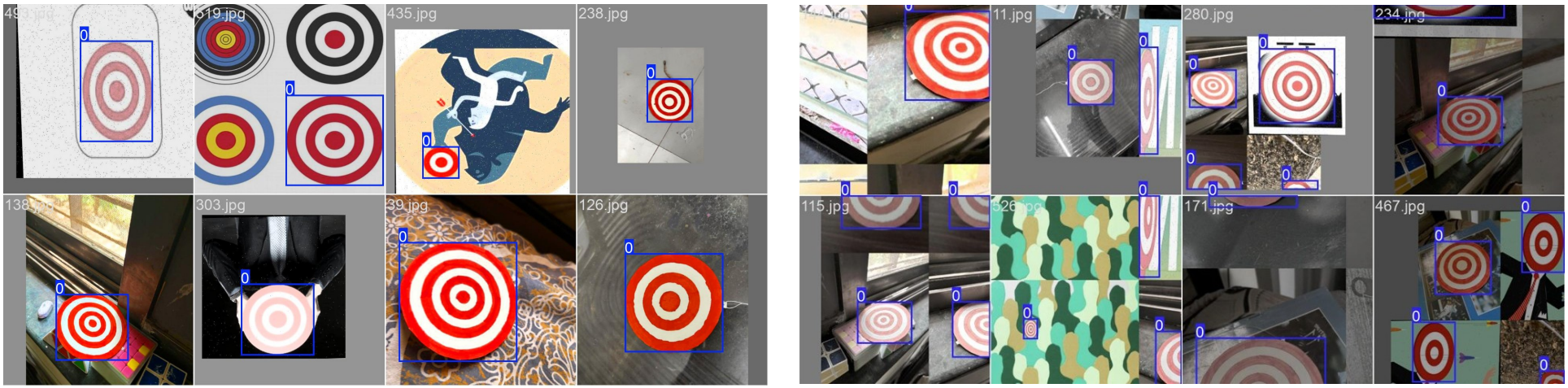
1. Create a diverse custom bullseye dataset
2. Fine-tune YOLOv8 and MobileNetSSD models
3. Evaluate performance: mAP, IoU, precision, recall
4. Optimize and export model to ONNX/TensorRT

Data Collection Overview

- **Hand-drawn captures**: varied lighting, backgrounds, angles
- **Online sources**: Google Images for red/white bullseyes
- **Roboflow Universe**: additional labeled samples

Total images: 550

Sample Dataset Images



figureExamples of bullseye images used for training

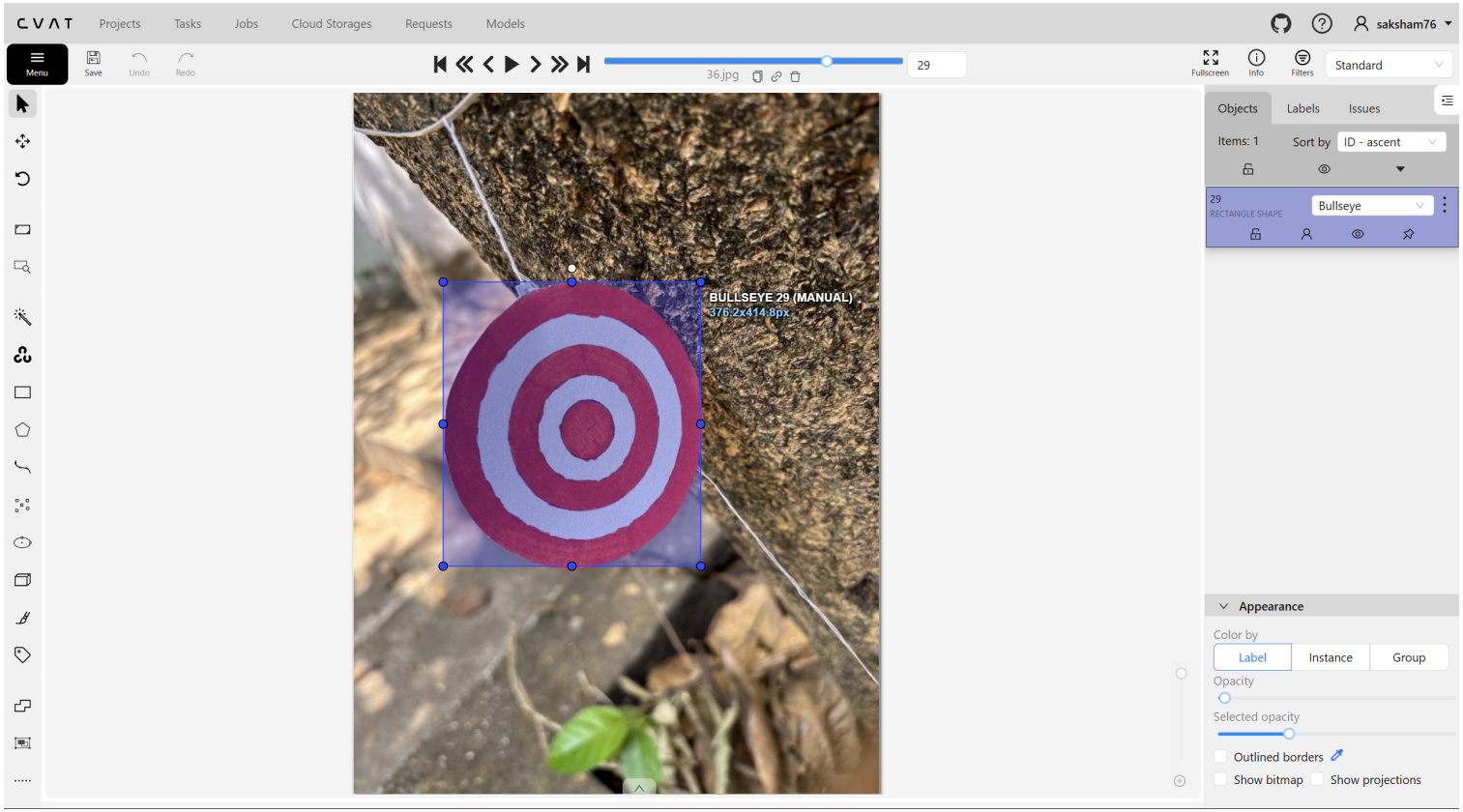
Preprocessing Pipeline

1. HEIC conversion via `pillow-heif` + PIL
2. Resize to 768x1024 with padding using OpenCV
3. Sequential renaming for YOLO compatibility
4. Train/Validation split

Key script: aspect ratio preservation ensures no distortion

Annotation with CVAT

- Tool: CVAT for precise bounding boxes
- Export: YOLO TXT format (class, x_center, y_center, width, height) normalized
- Folder structure: `images/train`, `labels/train`, etc.



CVAT annotation interface example

Model Selection

- **YOLOv8m**: 25.9M params, balanced speed/accuracy
- **Other variants**: nano (3.2M), small (11.2M), large (43.7M)
- Chosen for real-time capability on RTX 4070

Hyperparameters

Parameter	**Value**
Model	YOLOv8m
Batch Size	32
Epochs	100
Image Size	640x640
Workers	4
Learning Rate	0.01 (default)
Momentum	0.937 (default)
Weight Decay	0.0005 (default)
Device	NVIDIA RTX 4070

Table: Training Configuration Summary

Training Command

```
import torch

from ultralytics import YOLO

model = YOLO("yolov8m.yaml")

results = model.train(
    data="config.yaml",
    epochs=100,
    batch=32,
    workers=4,
    device=0
)
```

Data Augmentation

Default YOLOv8 augmentations:

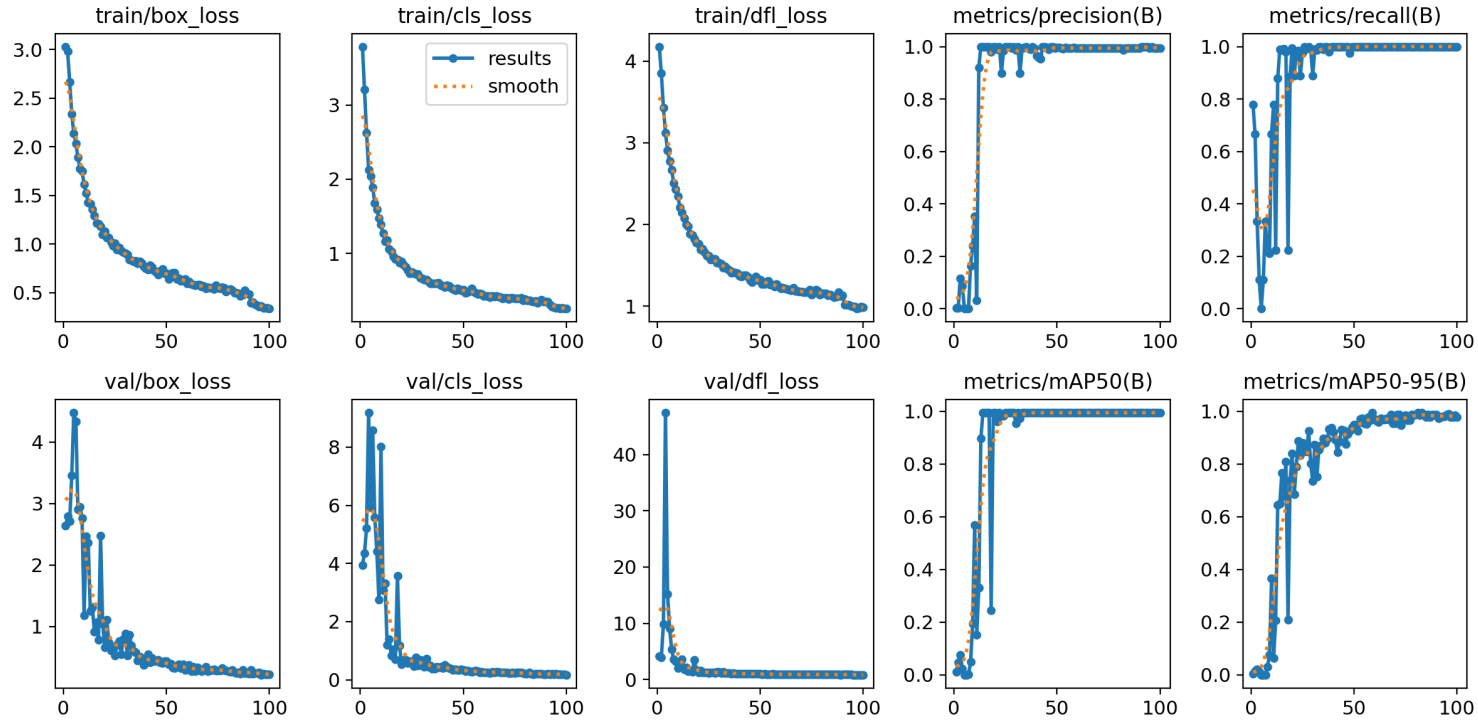
- Mosaic (enabled)
- HSV color jitter (h=0.015, s=0.7, v=0.4)
- Horizontal flip (0.5)
- Scaling, translation, rotation
- Perspective/stretch

Improves model robustness without manual tuning.

Quantitative Results

- Precision: 0.995
- Recall: 1.000
- mAP@0.5: 0.995
- mAP@0.5:0.95: 0.9785

Loss Curves



Training & Validation Loss vs. Epochs

Per-Class Performance

- Class "bullseye": AP@0.5 = 0.995

Exporting to ONNX

Conversion Command

```
yolo export model=runs/detect/train/weights/best.pt
format=onnx
```

- ONNX size: 98 MB

Side-by-Side Video

Play Original

Play Output

Key Challenges

- HEIC conversion aspect ratio preservation
- Annotation
- Ensuring GPU utilization in training

Solutions Applied

- Used `pillow-heif` + padding script
- Validated YOLO format via CVAT checks
- Configured torch CUDA and conda environment

Timeline of Work

[5 April] Assignment reviewed and setup
[6 April] Dataset collection and script development
[7-8 April] Annotation and preprocessing
[9 April] Model training and evaluation
[10 April] Model export and report writing

Conclusion

- High accuracy detection achieved
- Real-time inference demonstrated
- ONNX export enables flexible deployment

Future Work

- Deploy on edge devices with TensorRT
- Expand dataset with varied bullseye designs
- Implement tracking and multi-object scenarios

Questions?

Thank you!