Bullseye Detection Using Deep Learning and Computer Vision

Piyush Prajapati : 24B0930

April 2025

1 Dataset Preparation

1.1 Image Collection

The dataset was built in collaboration with my friend Saksham, with help of whom I was able to capture a lot of images. A total of over **200 photos** were collected. The first **40–50 images** were sourced from Google as screenshots.

Later, we shifted to capturing real-world images by creating a target using hand-made bullseye patterns. These were clicked in varied environments to ensure diversity in background, angle, and lighting conditions.

1.2 Data Annotation

All images were annotated using **CVAT (Computer Vision Annotation Tool)**. I manually marked each image to ensure precise localization of the bullseye targets. This step was critical to the model's eventual detection accuracy.

1.3 More Data Collection

Despite collecting 200+ images, I felt the dataset needed further diversity. Additional bullseye images were incorporated from online sources . I found an online dataset available form the following site : *https://universe.roboflow.com/bullseye-ijshx/my-first-project-5htcz*

2 Model Training

2.1 Learning Approach

I relied on YouTube tutorials to understand the implementation of YOLOv8. These resources helped me set up the environment, structure the dataset, and initiate model training. I watched the following tutorial mainly which was a huge help:
https://youtu.be/m9fH9OWn8YM?si=B7Kgd9cdNO3c2vTD

2.2 Training Strategy

Initially, I attempted to train the model for **100 epochs**, but frequent crashes in my laptop made me think of taking another way. As a workaround, I trained the model in batches of **5 epochs** using the same weights repeatedly. This allowed me to simulate longer training without overloading the system.

2.3 Model Configuration

I used **YOLOv8** for its balance between speed and accuracy. This is the code snippet for notebook in which i compiled:

```
from ultralytics import YOLO
model = YOLO("runs/detect/train/weights/best.pt")
results = model.train(data="config.yaml", epochs=5, batch=64)
```

3 Model Evaluation

3.1 Performance Metrics

After training, the model was evaluated by selecting some of the images from total dataset. YOLOv8 automatically generated performance graphs and statistics helped in analysis .These were : Mean Average Precision (mAP) Intersection over Union (IoU) Loss curves for classification, localization, and objections and many other graph helpful in analyzing progress

3.2 Observations

The results indicated that the model was successfully learning to identify bullseyes, though there is still room for improvement with a larger dataset and extended training epochs.

4 Model Conversion and Deployment

4.1 ONNX Export

The model was reformatted into ONXX format by running the following command in terminal

```
yolo export model=best.pt format=onnx
```

5 Conclusion

This project covered everything from collecting and annotating data to training and evaluating a YOLOv8 model for bullseye detection. Along the way, I faced issues like limited dataset size and system crashes during training, but working around them helped me understand the process better.

The model was trained in small chunks and still gave decent results. I also learned a lot about interpreting training graph.

Finally, the model was converted to ONNX format, which sets it up nicely for future deployment and real-time inference. Overall, this project taught me not just technical skills, but also how to stay patient and keep moving step by step until things start working.