# CV PROJECT

BY- NISHKARSH

# Introduction and Problem statement

Built a deep learning model to detect red and white bullseyes using YOLOv8, aimed at real-time accuracy .The project addresses bullseye detection challenges by using a custom made dataset and combining YOLOv8 for training with OpenCV for testing. for drone applications.

# Project Overview

1. Dataset Preparation
2. Model training
3. Evaluation
4. Result Analysis

# Dataset Preparation

**Dataset-1**
Downloaded 50-60 images from google and created a dataset by applying data augmentation in roboflow.

**Dataset-2**
Used a dataset present in roboflow of 800 images of red and white bullseyes.

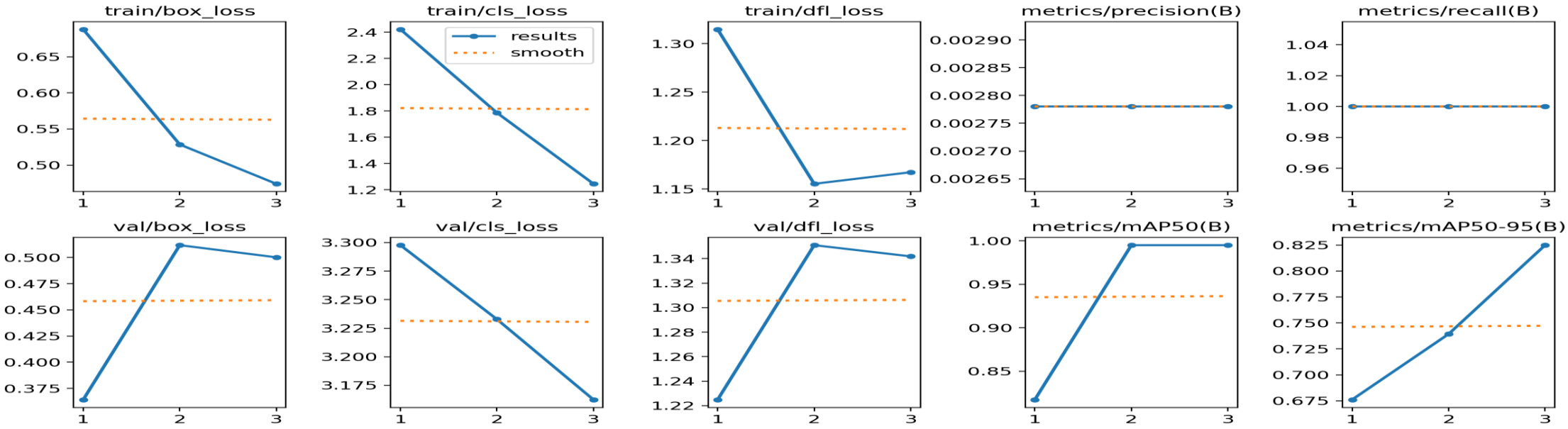
**Final Dataset**
Combined 1$^{st}$ and 2$^{nd}$ dataset and also added images of bullseyes other than red and white , applied data augmentation on that dataset in roboflow.

# Model Training And Evaluation

## On Dataset-1

Trained yolov8 on this dataset and was getting very poor accuracy.

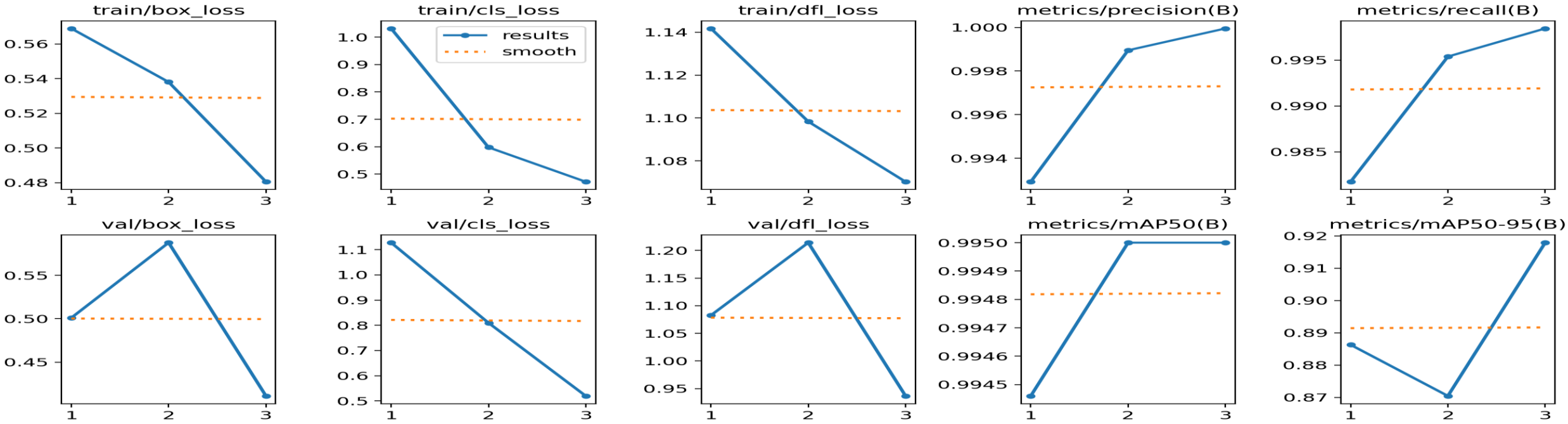
Reason: Dataset was very small and consisted images of almost same kind not very much augmented.



## Dataset 2

This was a large dataset trained model on this and was getting very high accuracy . The model overfits . Was nearly predicting every bullseye as red and white.

Reason: Consisted images of same kind( of only red and white bullseye).



## Final dataset

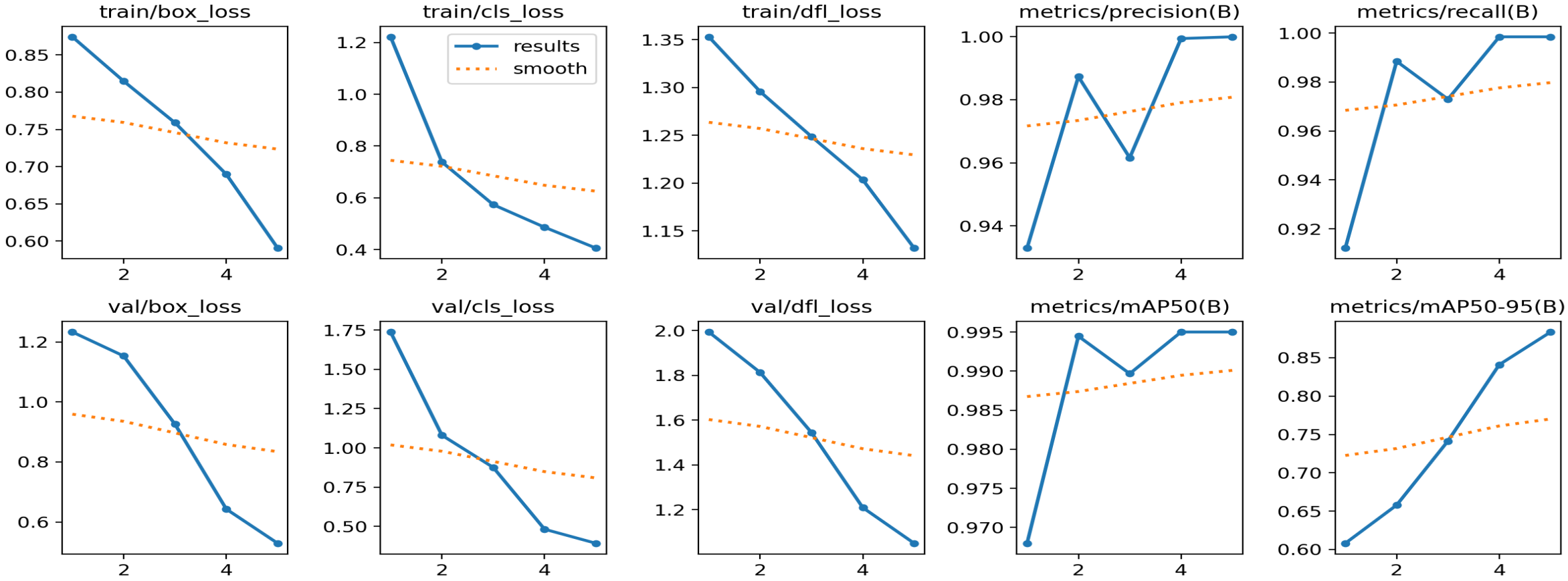
This dataset consisted of all red and white bullseyes and bullseyes other than red/white . Using roboflow marked the other bullseye as null . Important observation was that it was still detecting the other bullseyes but the confidence score on red and white bullseyes was pretty high as compared to the bullseyes other than red and white . So I added a code to consider only those bullseyes as red and white whose confidence score was more than 0.8 and the Model was working pretty fine on both videos and live webcam

# Code snippet

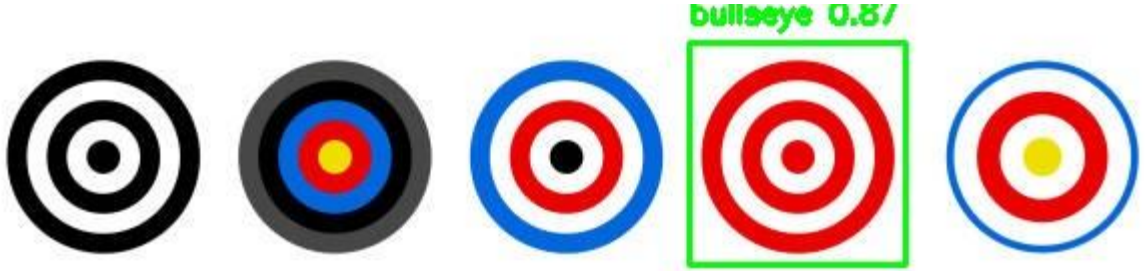
```
for result in results:
    boxes = result.boxes
    for box in boxes:
        conf = float(box.conf)
        if conf > 0.8:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            cls = int(box.cls[0])
            label = f"{model.names[cls]} {conf:.2f}"

            # Draw the bounding box and label on the image
            cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(image, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.5, (0, 255, 0), 2)
```

# Evaluation Metrices



# Some output images





# Test Video

# Conclusion and learning

Exported the model in onnx format. One of the most important takeaways from this project was realizing how crucial the dataset is to a model's performance. Initially, training with only red and white bullseyes caused the model to overfit, resulting in misclassification of other bullseyes. Expanding the dataset to include various colors and patterns significantly improved the accuracy and generalization of the model. The project provided a hands-on example of how overfitting happens when the model sees too few variations during training. By diversifying the training data, the model learned to generalize better to unseen inputs.