**Task-1: Ball Detection**

**Summary:** This project involved developing a robust ball detection model using YOLOv8 for RoboSoccer. I started by understanding the necessary tools, creating a dataset with labeled images, and training the model with augmentations. Through iterative improvements, I enhanced its accuracy, tackled issues like missed detections, and optimized performance using Google Colab’s GPU. The final model successfully detects soccer balls in various conditions, making it a strong candidate for RoboSoccer applications.

Link to Google Collab = <https://colab.research.google.com/drive/18ZvOtlAqEZHvlI7QWYX5K6uimb5GyW1i?authuser=3#scrollTo=iWHRTzKBnfLI>

### **Ball Detection Project**

#### **Initial Approach**

As this was my first AI/ML project, I was eager to explore the necessary tools and techniques. I researched available resources and found **Roboflow** to be particularly useful for dataset preparation. To understand its functionalities, I watched multiple tutorials and familiarized myself with its features.

#### **Dataset Creation**

My first task was assembling a dataset. Initially, I considered using real football images and started by taking screenshots from YouTube videos. Instead of using full-frame images, I cropped them to focus only on the ball. The dataset initially had **20 images**, manually labeled using **Roboflow**. Afterward, I created a dataset version with no augmentation and later added **brightness augmentation** after learning its impact.

#### **Training the Model**

I first attempted training on **Kaggle**, but due to slow response times, I switched to **Google Colab**, where I had prior experience. I set up dependencies such as **Ultralytics, Roboflow, and PyTorch**, ensuring compatibility with **YOLOv8**. Guided by ChatGPT, I learned about training parameters and key metrics like **mAP50 (mean Average Precision at 50% IoU) and mAP50-90**.

I initially trained for **50 epochs** but extended it to **100 epochs** upon noticing a steady decline in loss graphs. I analyzed various loss metrics, including **train\_box\_loss** and **val\_box\_loss**, to evaluate model performance. The trained model was then uploaded to **Roboflow** for further testing.

#### **Challenges and Iterative Improvements**

After testing the model with new images, I observed a limitation: it failed to detect the ball when people were present in the image. To address this, I revised my dataset to include **soccer scenes with both players and balls**. This required retraining, and I leveraged **Google Colab’s free GPU**, which significantly reduced training time compared to CPU processing.

Later, I realized that our application was for **RoboSoccer**, so I created a new dataset with images from RoboSoccer matches. I introduced **brightness and blur augmentations** for better generalization. However, the model struggled to detect larger balls. To fix this, I used **anchor tuning**, which improved detection.

#### **Final Model and Performance**

The final model was trained on RoboSoccer images with **100 epochs** and optimized anchors. It achieved reliable detection with a confidence level of **50% or higher**. I validated it with real-world RoboSoccer videos, where it performed well. Throughout this process, I gained insights into **underfitting, overfitting, augmentation strategies, and confidence tuning**.

#### **Key Learnings and Takeaways**

* **Dataset quality is crucial**: Including diverse conditions improves detection.
* **Training on GPU significantly speeds up the process** compared to CPU.
* **Fine-tuning augmentations and anchors enhances accuracy.**
* **Understanding loss graphs helps in adjusting training strategies.**
* **Iterative improvements are essential** for real-world performance.

This project provided hands-on experience in **model training, dataset handling, and iterative optimization**, equipping me with valuable skills for future AI/ML applications.