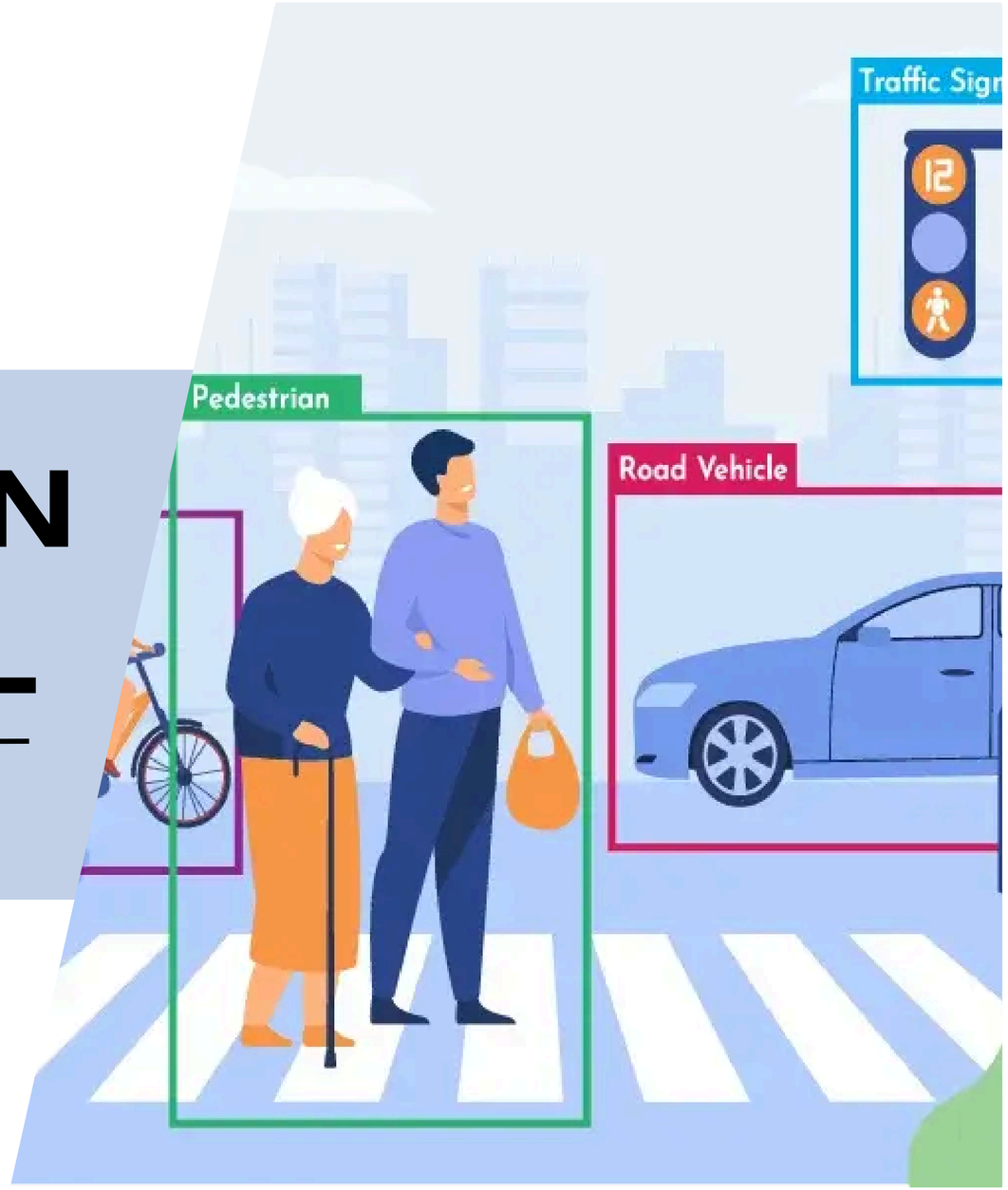
Institute of Technology of Cambodia

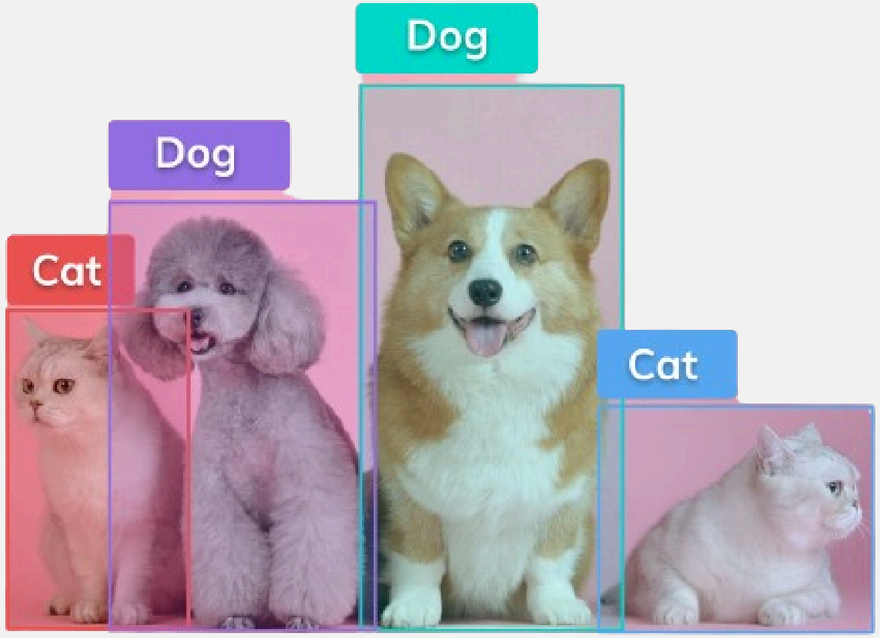
Electrical and Energy Engineering

OBJECT DETECTION WITH YOLO MODEL

PROJECT FOR ABU ROBOCON 2024 COMPETITION



CONTENT



1. INTRODUCTION

1.1. What is Object Detection?

Object detection is a computer vision method used to find and identify objects in images or videos. It draws a bounding boxes around the objects to show their location and track their movement.

Use cases for Real-World Application:

- Object Counting
- Self-Driving Car
- Security System ... etc

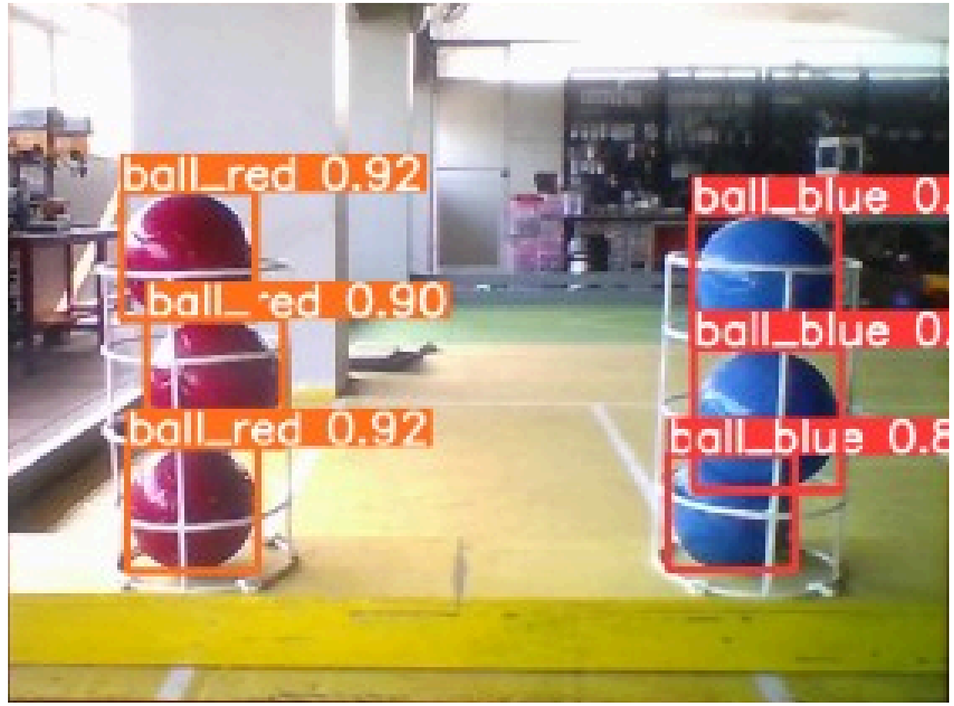


Figure 1: Ball Detection

Historical Periods of Object Detection

The Object detection is usually separated into two historical periods (Before and After Deep Learning):

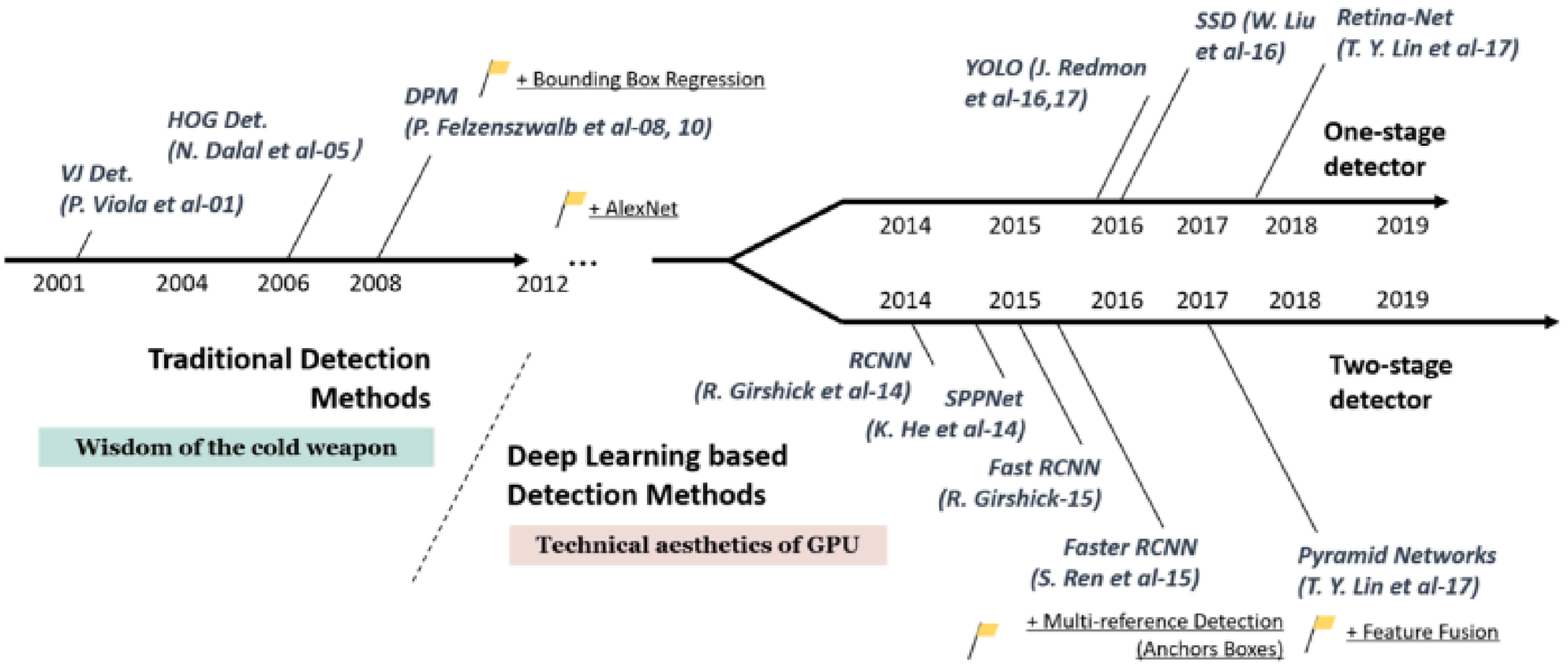


Figure 2: Object Detection Historical Periods (https://wikidocs.net/167508)

1. INTRODUCTION

1.2. Overview of yolo model

The YOLO model was first published (by Joseph Redmon et al.) in 2015. YOLO (You Only Look Once) is a state-of-the-art deep learning model for real-time object detection in computer vision applications

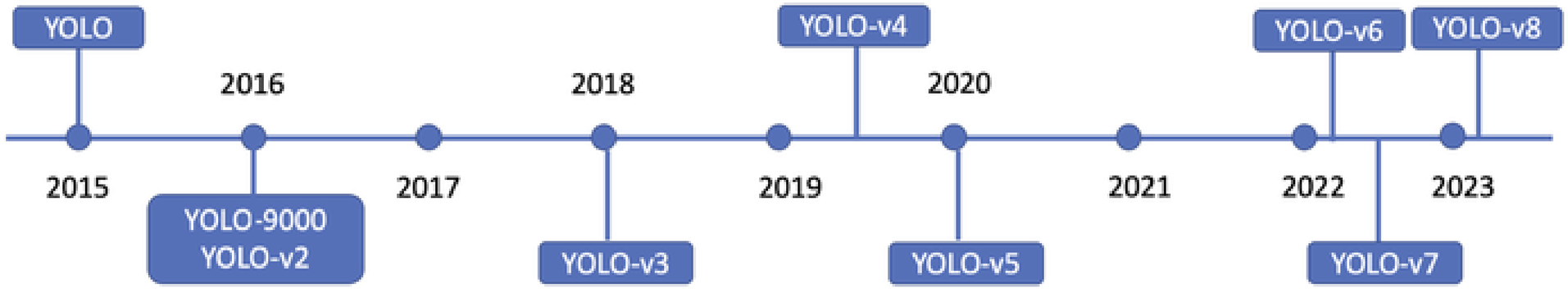


Figure 3: Time line of YOLO model

How does it work?

In the YOLO model is trained on labeled datasets, it divides the image into a grid and predicts bounding boxes and probabilities for each object in each grid cell in a single, pass through the network. The YOLO algorithm takes an image input and passed to deep Convolutional Neural Network (CNN) and neural network then, generates an output in the form of a vector.

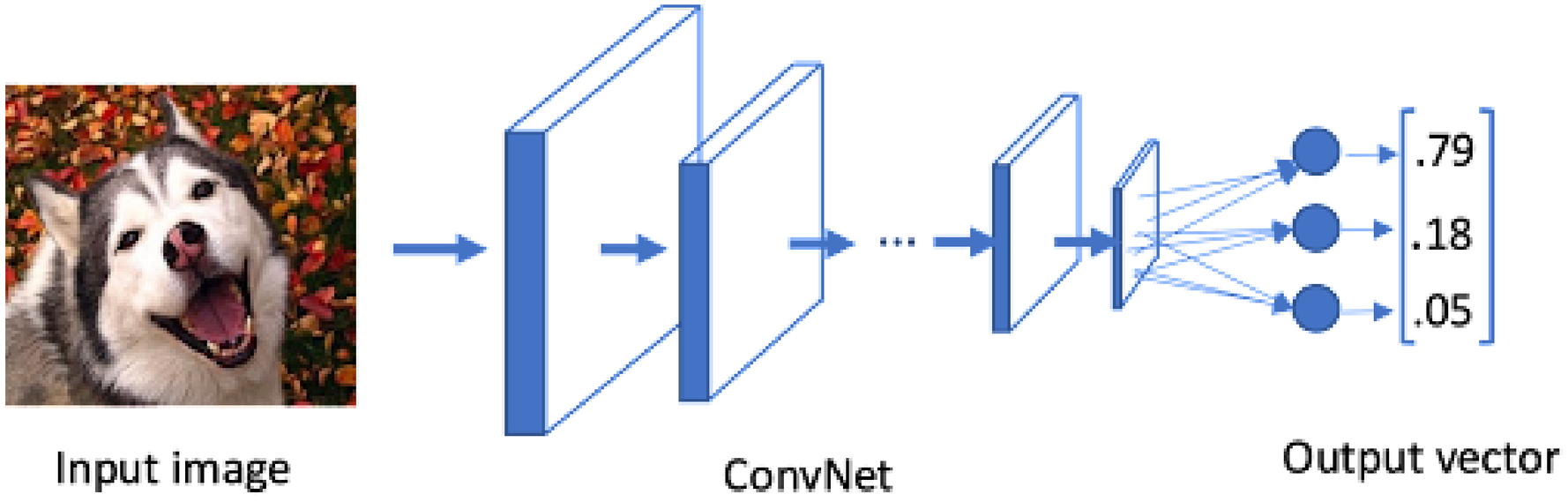


Figure 4: YOLO Model Algorithm

1. INTRODUCTION

1.3. Objective

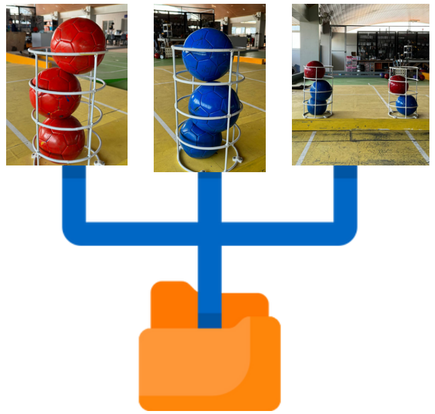
The primary objective of this project is to participate in the ABU Robocon Competition by performing object detection (system) to identify blue and red balls. It will allow a robot to complete challenges autonomously under the rules of the competition.

Figure 5: Roboncon Game Fields (Area3)

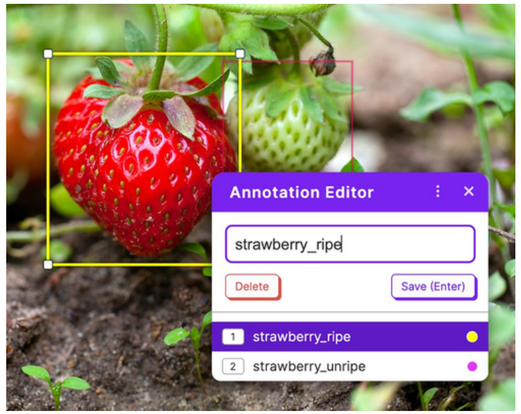
1. INTRODUCTION

1.4. Scope of works

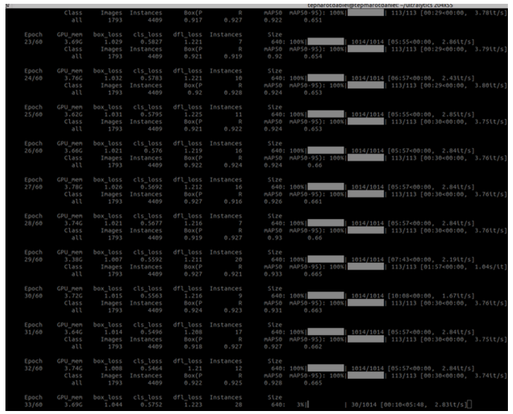
In this project we worked on:



Data collection



Annotate data on Roboflow (Link)

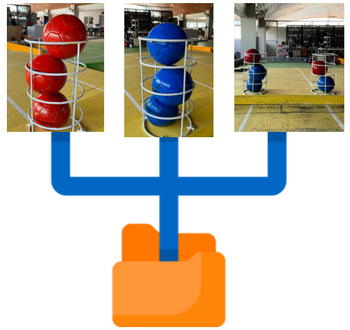


Model Training

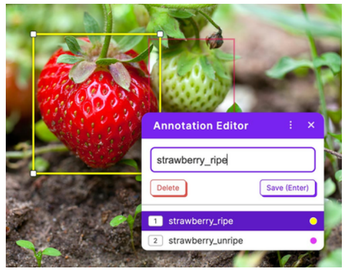
Hardware Implementation

2. SETUP YOLO MODEL

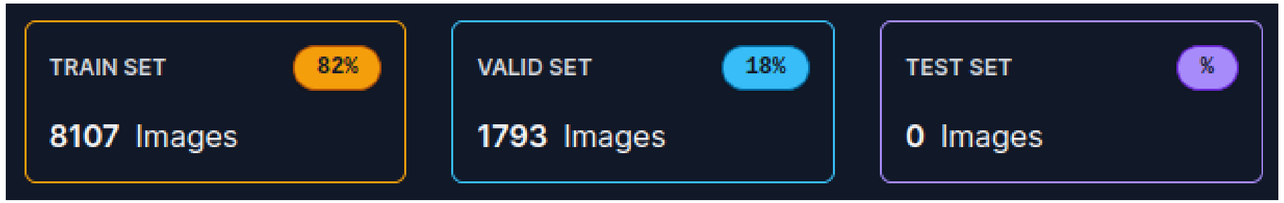
2.1. Dataset Preparation



Data Collection: Capture image from camera frame or discover on website (Roboflow)



Data Annotated: Draw Bounding box around the object and label the name for it



Size of Dataset: Train, Valid, and Test

2. SETUP YOLO MODEL

2.2. YOLO Model Selection

We choose yolov8 to train with the dataset. Yolov8 were from the Ultralytics team. A full range of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification.

Figure 6: Ultralytics YOLOv8

Why YOLOv8 ?

Once more, the Ultralytics team has conducted benchmarking of YOLOv8 using the COCO dataset, revealing notable advancements compared to prior YOLO iterations across all five model sizes.

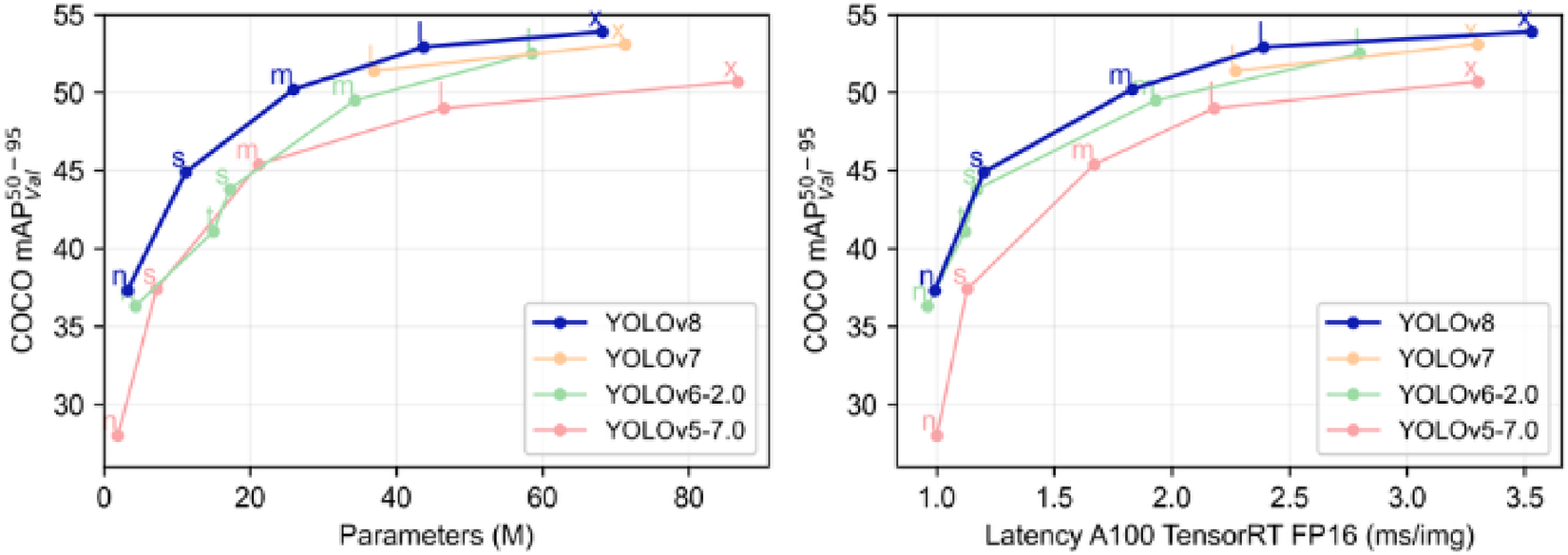


Figure 7: Benchmark Results Across YOLO (**mAP**:Performance, **In fps**: Speed of the inference, **FLOPs and params**: Compute or the model size)

Comparison Object detection model size

In the comparison, on the COCO dataset the YOLOv8m model obtained a **mean Average Precision (mAP)** of 50.2%. Meanwhile, the YOLOv8x, the largest model among the set, achieved 53.9% mAP.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 8: The comparison of object detection across five different model sizes (source)

2. SETUP YOLO MODEL

2.3. Training Procedure

We used Asus ROG Strix G15 to train the dataset on the **yolov8m.pt** model.

- Dataset size: Around 2 0000
- CLI: yolo train data=(.yaml) model=yolov8m.pt epochs=100 imgsz=640 device=0 batch=8
- Duration: 2 days

CPU	
CPU Series	AMD Ryzen 7
CPU Cores	8
CPU Threads	16
Display	
Display Resolution	Full-HD (1920 x 1080 Pixel)
Display Refresh Rate	144 Hz
Display Panel Type	IPS
Display Surface Type	non-glare
Graphics Card	
GPU Model	NVIDIA GeForce RTX 3050
GPU VRAM Capacity	4 GB

Figure 9: Asus ROG Specification

Results after Training the Dataset:

We used Ultralytics packages YOLOv8.2.16, python3.10.12 torch-2.2.2+cu121 CUDA:0 (Nvidia Geforce RTX 3050, VRAM = 3902MB)

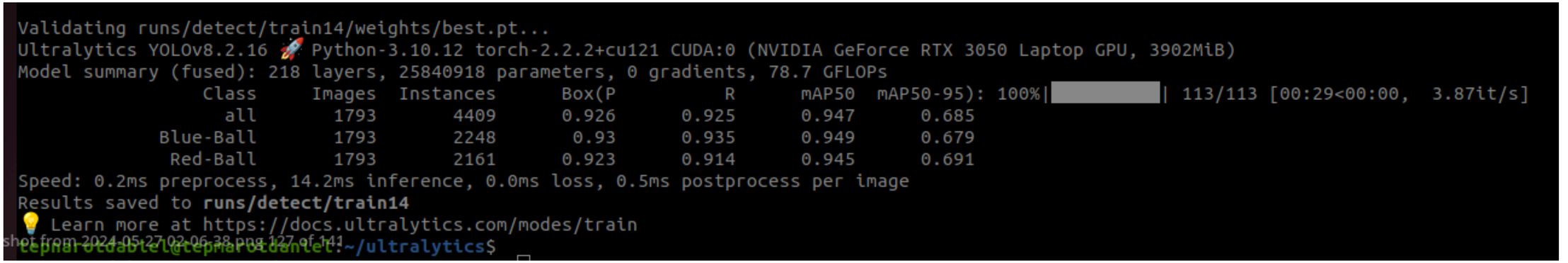


Table A: Training Results of Dataset

Class	Precesion	Recall	mAP	mAP50-95
All	0.926	0.925	0.947	0.685
Blue-Ball	0.93	0.935	0.949	0.679
Red-Ball	0.923	0.914	0.945	0.691

3. HARDWARE IMPLEMENTATION

In this project, we implemented on the Jetson AGX Xavier using the RealSense D435i.

Jetson Flashing (Ubuntu 20.04):

- Jetpack 5.1.2 buit-in Libraries:
  - CUDA: 11.4.315, TensorRT: 8.5.2.2
  - CuDNN: 8.6.0.166, OpenCV: 4.5.4

Setup YOLO Model on Jetson AGX Xavier:

- Setup Realsense for Jetson Packages
- Ultralytics: 8.2.16 (Remove torch and torchvision)
- Re-Install pytorch and torchvision (Available with JP 5.1.2)
- Setup onnx for tensor RT conversion

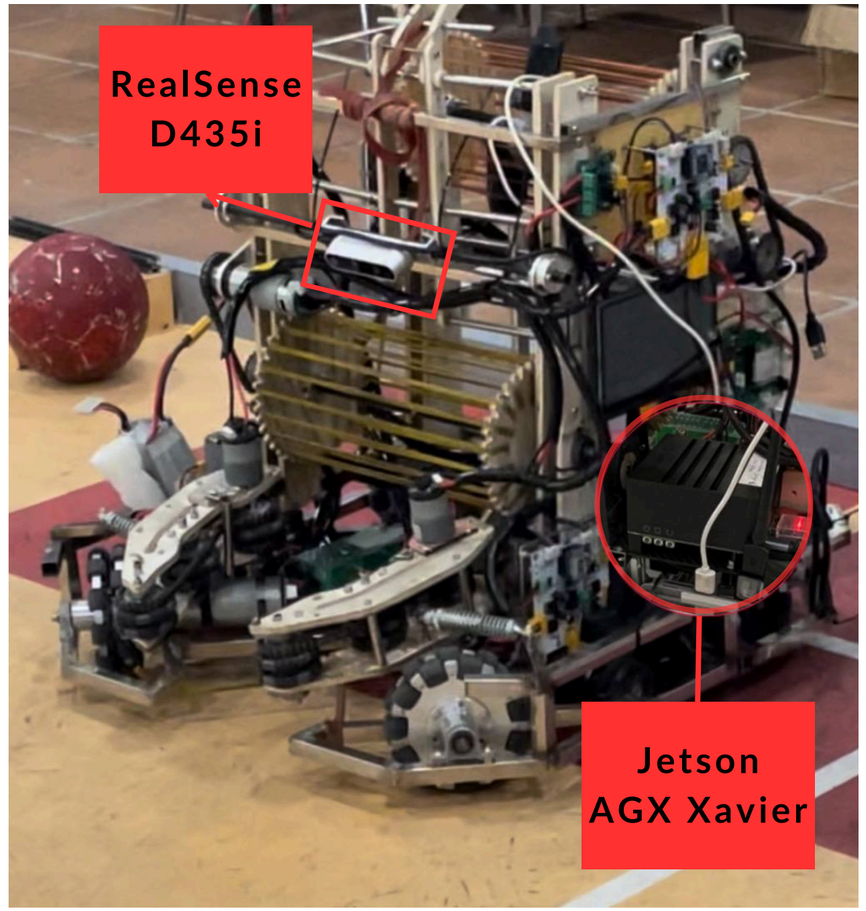


Figure 10: MR2 (Credit: ITC01 H101)

4. RESULT AND DISCUSSION

4.1. Results

After, setup the yolo model on the hardware we get the result from the both frames of Realsense D435i and Webcam on the Robot MR2:

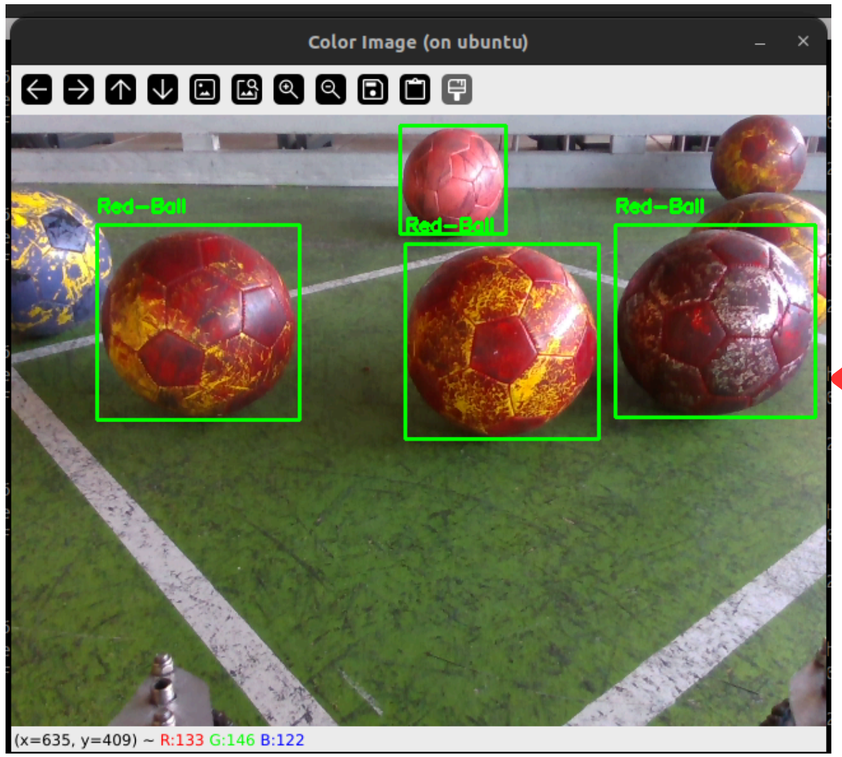


Figure 11.a: From Realsense Frame

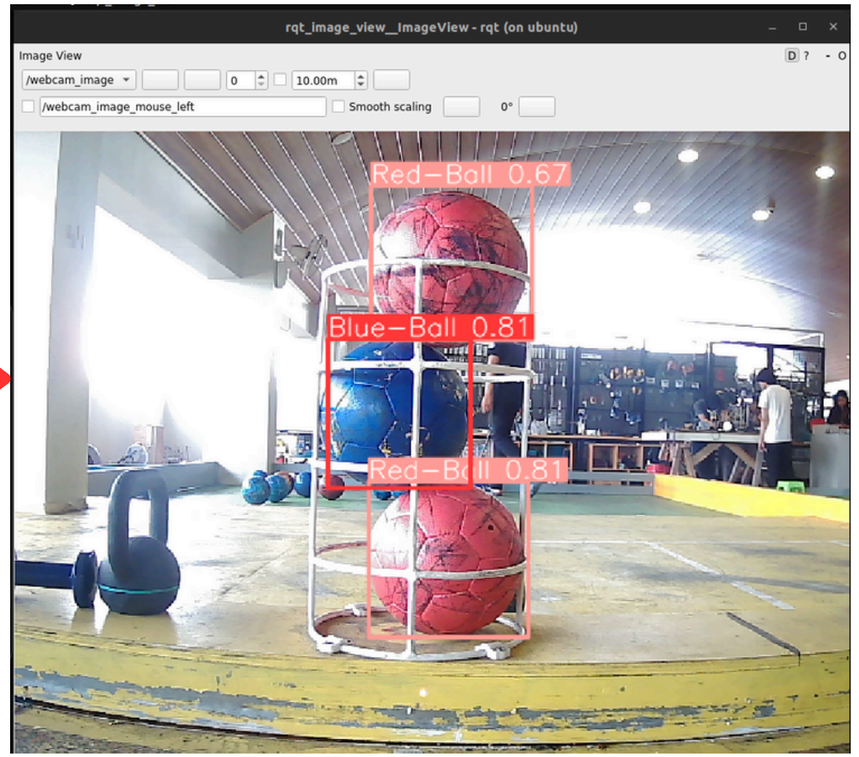
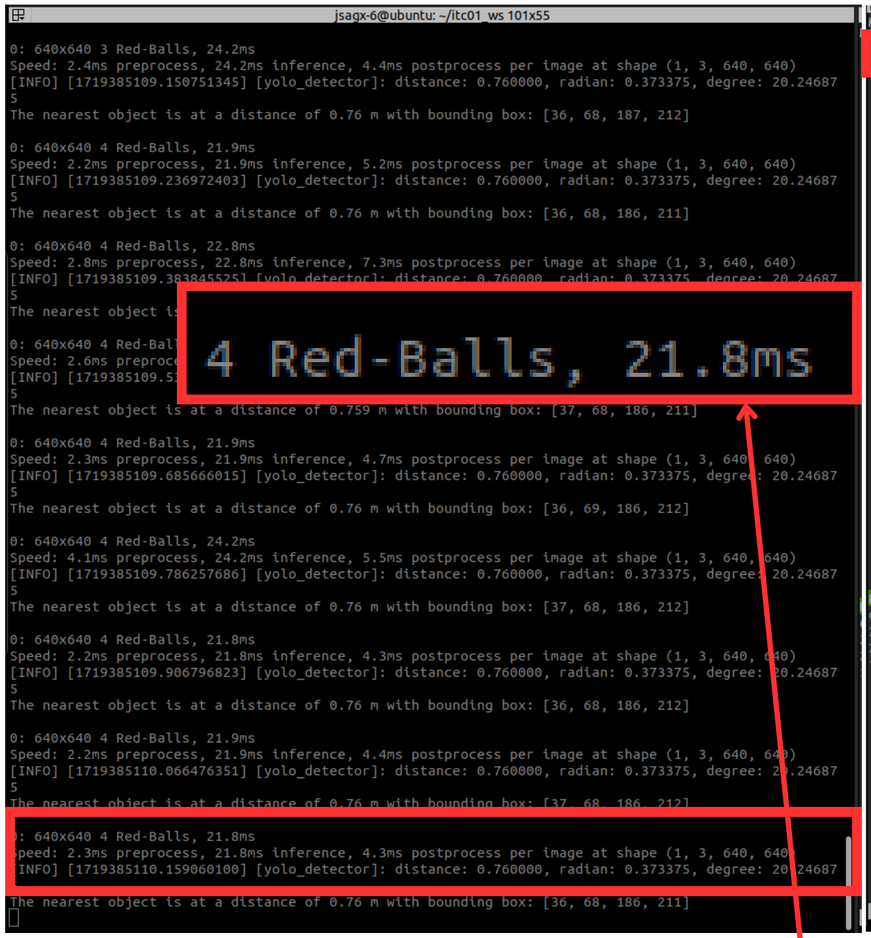
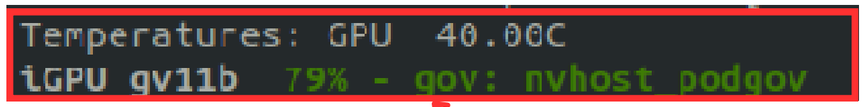
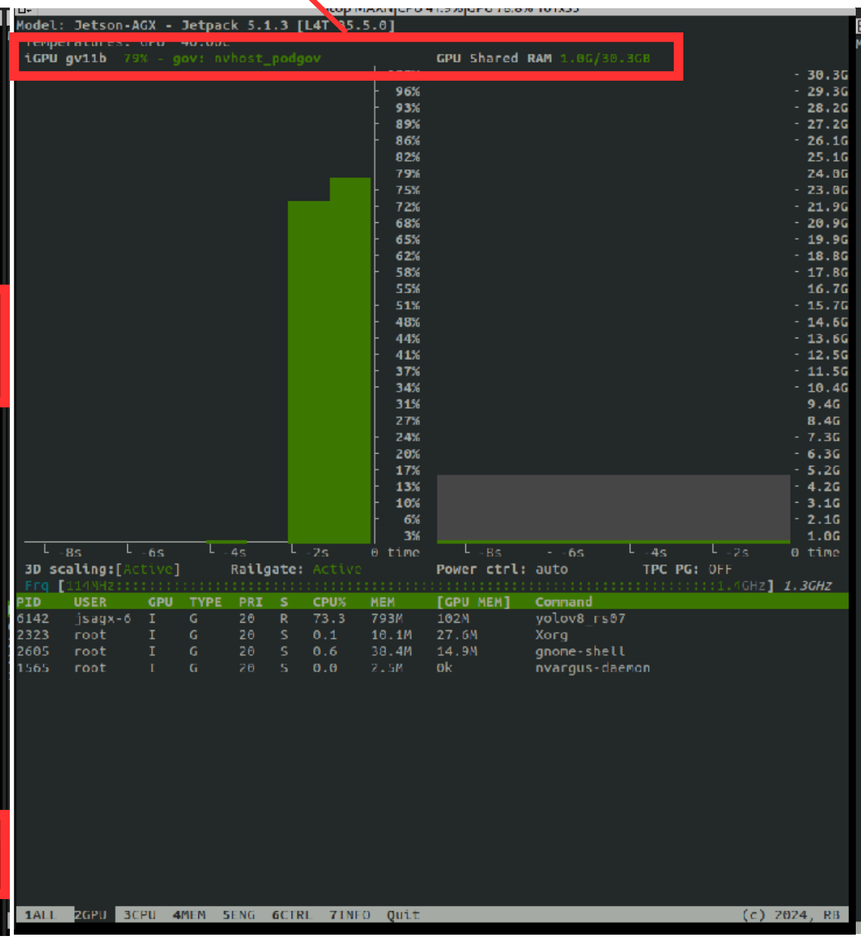


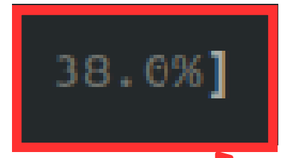
Figure 11.b: From Webcam Frame

4.2. Discussions









a). Inference Time: measures how quickly the model can process input data and detected -> 21.8ms

b). GPU Performance -> 79%

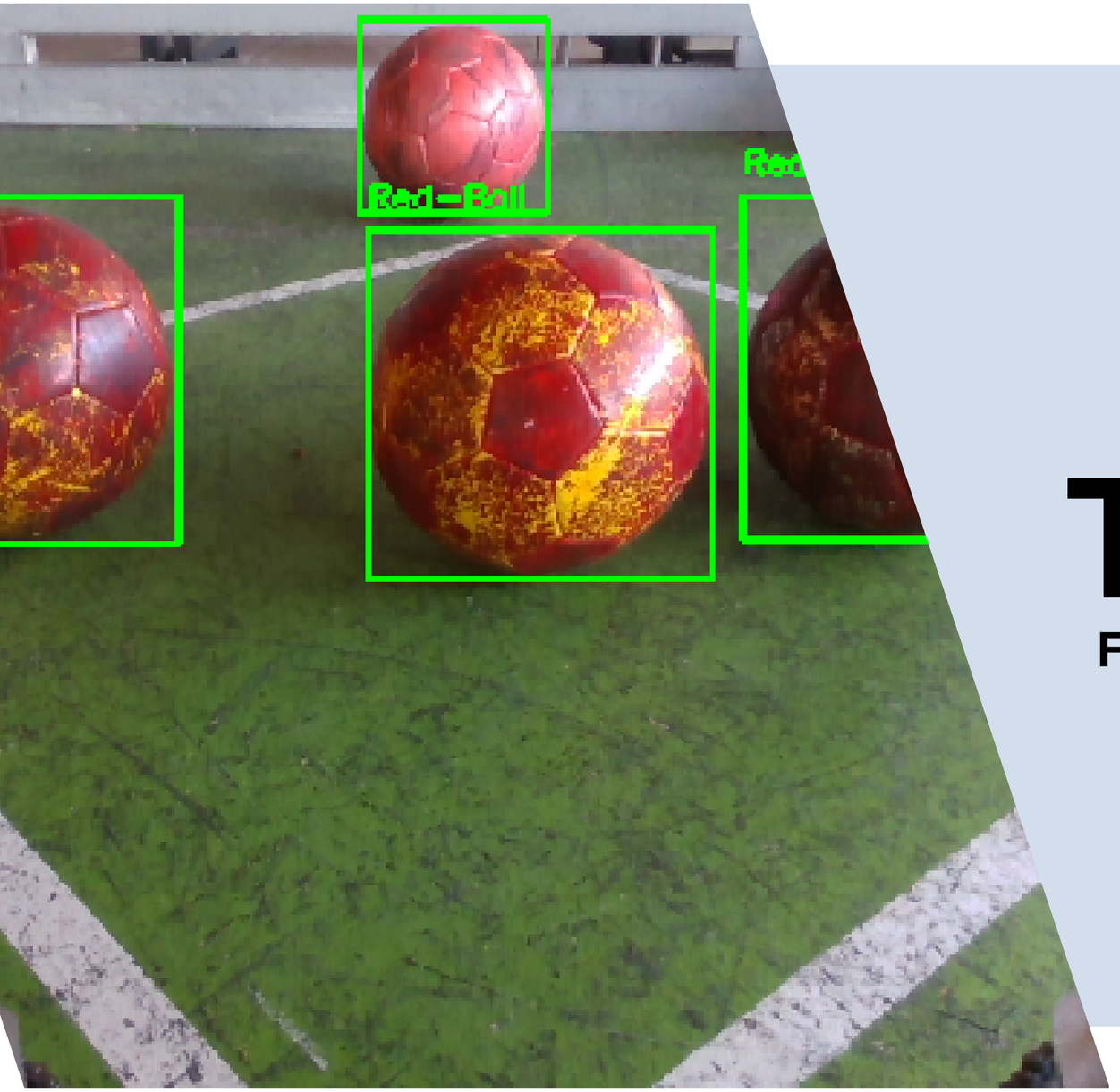
c). CPU Performance: Run all nodes in the same time: 38 %

5. CONCLUSION

The purpose of this project is to enable the robot to automatically catch and deliver balls to silos within a maximum of 3 minutes for a game competition. Therefore, base on 3 main points:

- The GPU utilization ranges between 79-90%, indicating that the model leverages the GPU efficiently, which is expected for a deep learning application like YOLOv8.
- The CPU usage at 38% indicates there is enough capacity for other tasks, ensuring the system can handle additional processes without overloading.
- The YOLOv8m.pt model demonstrates accuracy and efficiency with an inference time of 21.8ms.

As the result, our model operates efficiently on the robot and does not cause any slowdowns, making it suitable for the competition.



THANK YOU

FOR YOUR ATTENTION!!

REFERENCES

[1] Panja, E., Hendry, H., & Dewi, C. (2024). YOLOv8 Analysis for Vehicle Classification Under Various Image Conditions. Scientific Journal of Informatics, 11(1), 127–138.

[2] Xiao, X., & Feng, X. (2023). Multi-Object pedestrian tracking using improved YOLOV8 and OC-SORT. Sensors, 23(20), 8439.

[3] Terven, J., Córdova-Esparza, D., & Romero-González, J. (2023). A comprehensive review of YOLO architectures in computer vision: from YOLOV1 to YOLOV8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680–1716.

[4] Lou, H., Duan, X., Guo, J., Liu, H., Gu, J., Bi, L., & Chen, H. (2023). DC-YOLOV8: Small-Size Object Detection Algorithm based on Camera Sensor. Electronics, 12(10)

APPENDIX