YOLO: You Only Look Once Real-Time Object Detection

Klodjan Hidri
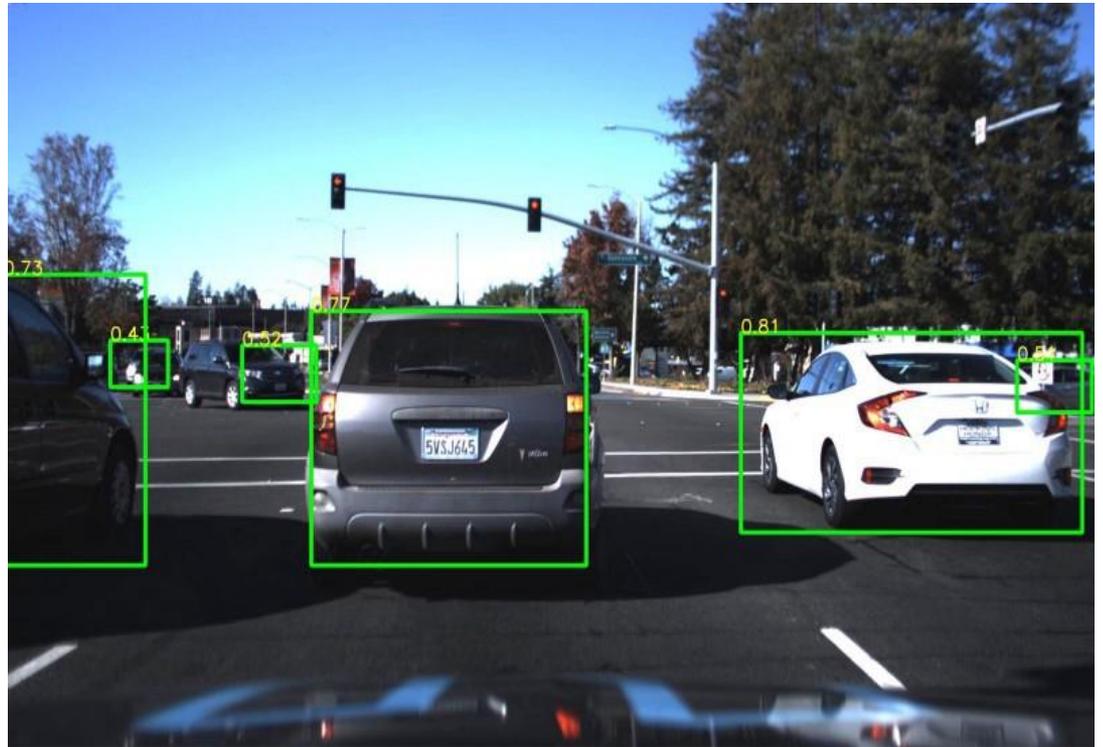
Introduction

YOLO You Only Look Once CNN-based

Single-stage Detection

Object Detection Identifies Objects in images/videos
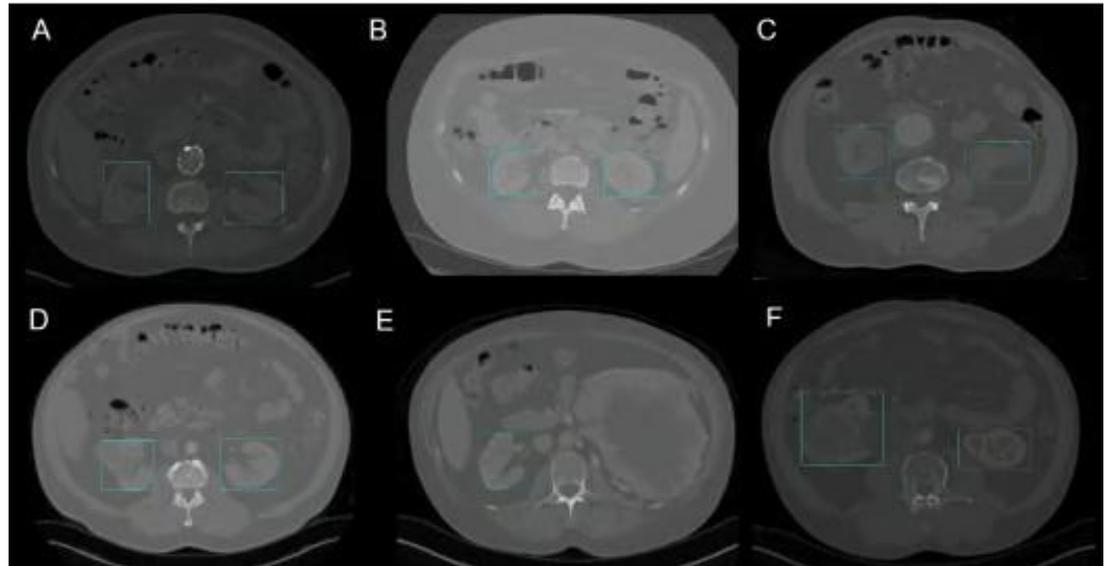
A real-time Object Detection System
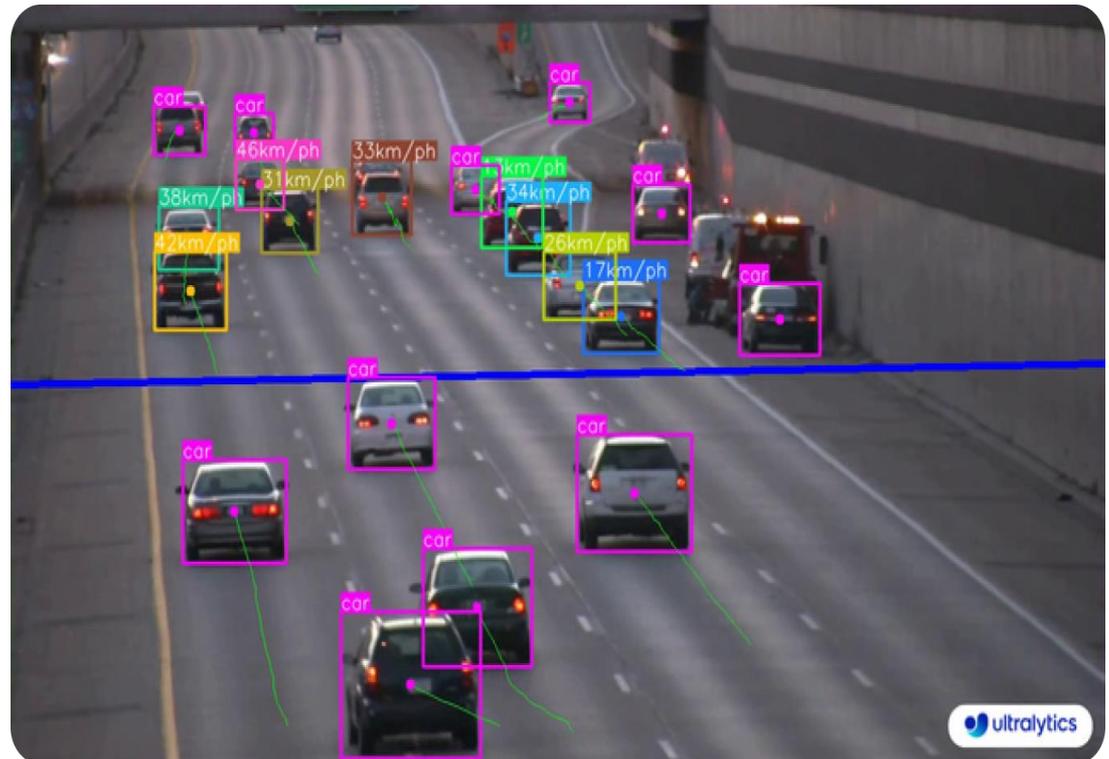
Applications

- Autonomous driving

Applications

- Medical imaging

Applications

- Video security surveillance

Applications

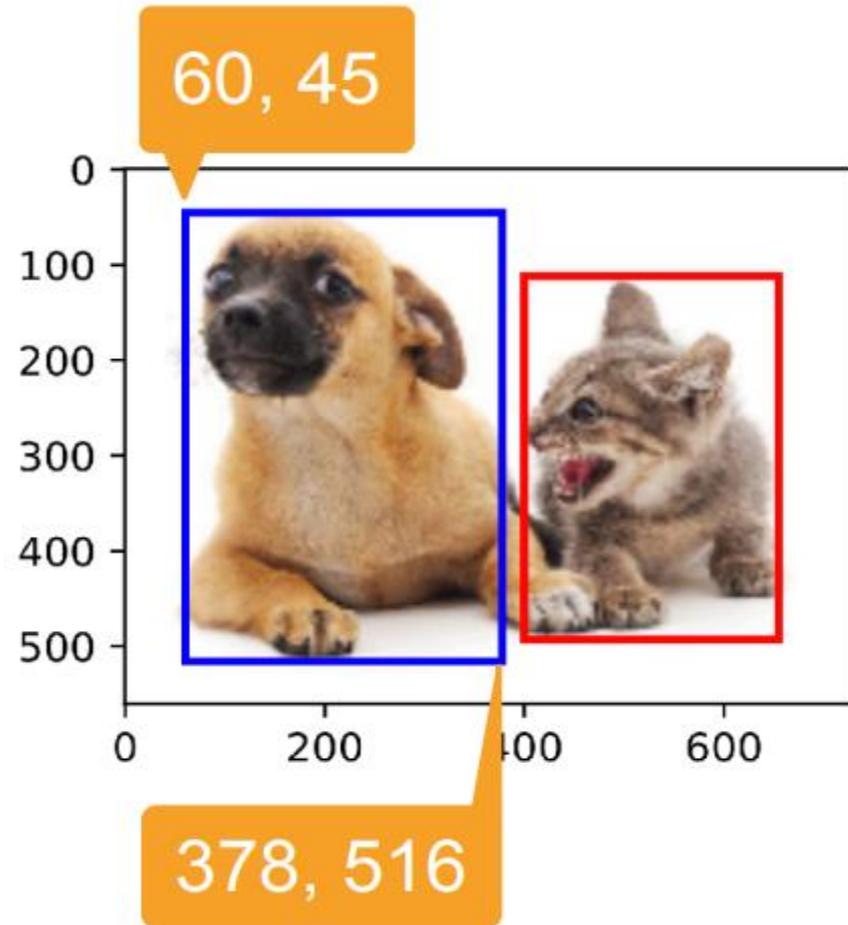- Robotics

Older Object Detections Models

- R-CNN   CNN-based

- Fast R-CNN   CNN-based
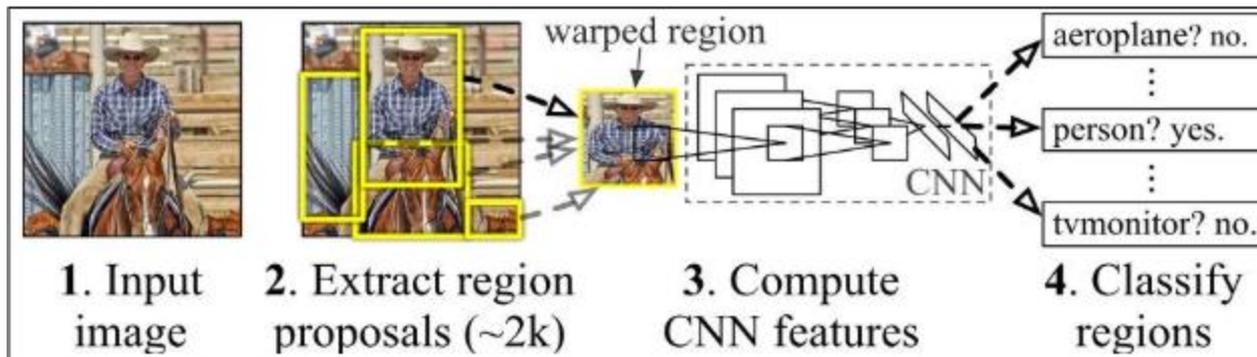
- Faster R-CNN CNN-based

Bounding Box

- A bounding box can be defined by 4 numbers, – (top-left x, top-left y, bottom-right x, bottom-right y)
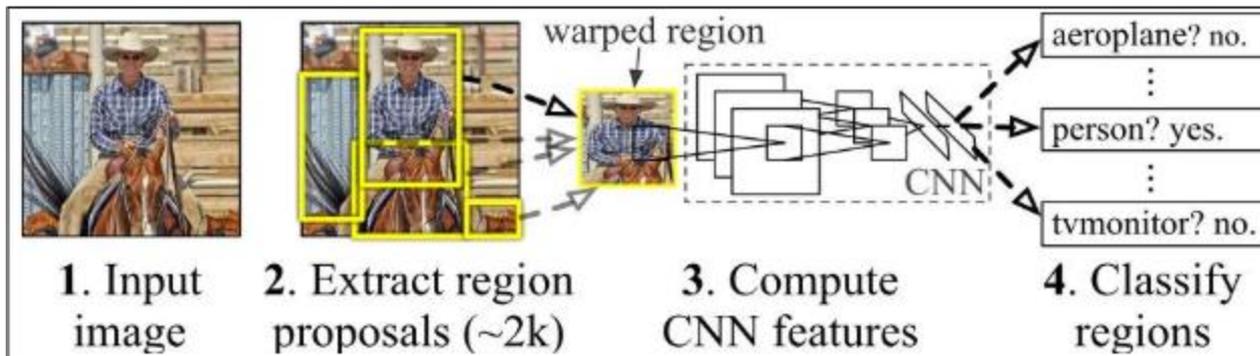
Older O.D. Overview

- First stage of detector
- **Input image** →
  - Take the raw input image
- **Region Proposals** → (Selective Search)
  - Generate ~2000 **region proposals** (candidate bounding boxes)

1. Input image 2. Extract region proposals (~2k) 3. Compute CNN features 4. Classify regions

warped region — CNN — aeroplane? no. … person? yes. … tvmonitor? no.

Older O.D. Overview

- **Second Stage of detector**
- **Warp & Extract Features**
  - Each proposed region is cropped and **warped to a fixed size** (e.g., 227×227 pixels)
- **Classification**
- -For each region's features, use the model to decide what object it contains (or background)
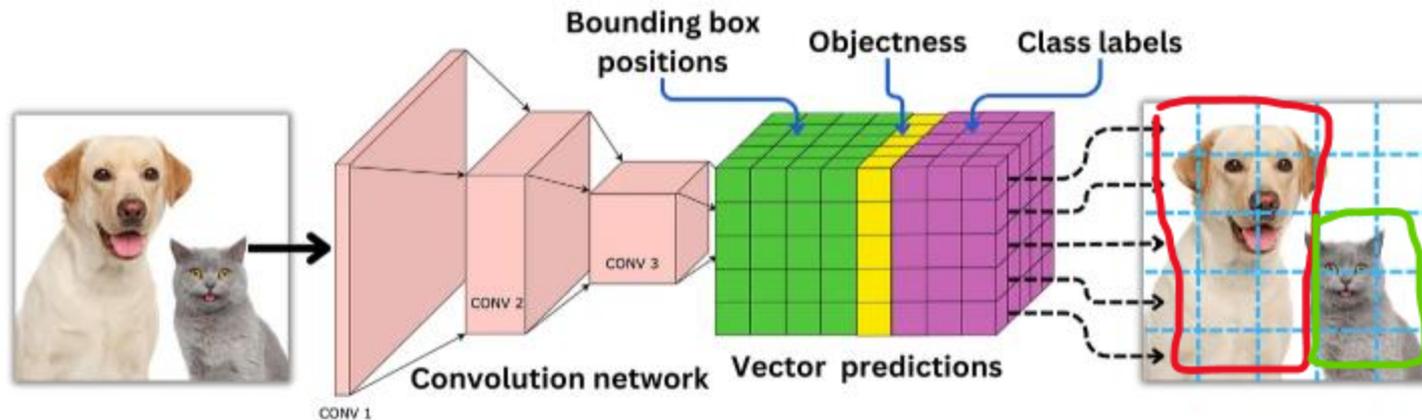
Older O.D. Overview

- 👉 Strength: Very accurate, especially on small or overlapping objects
- 👉 Weakness: Slower due to region proposal stage.
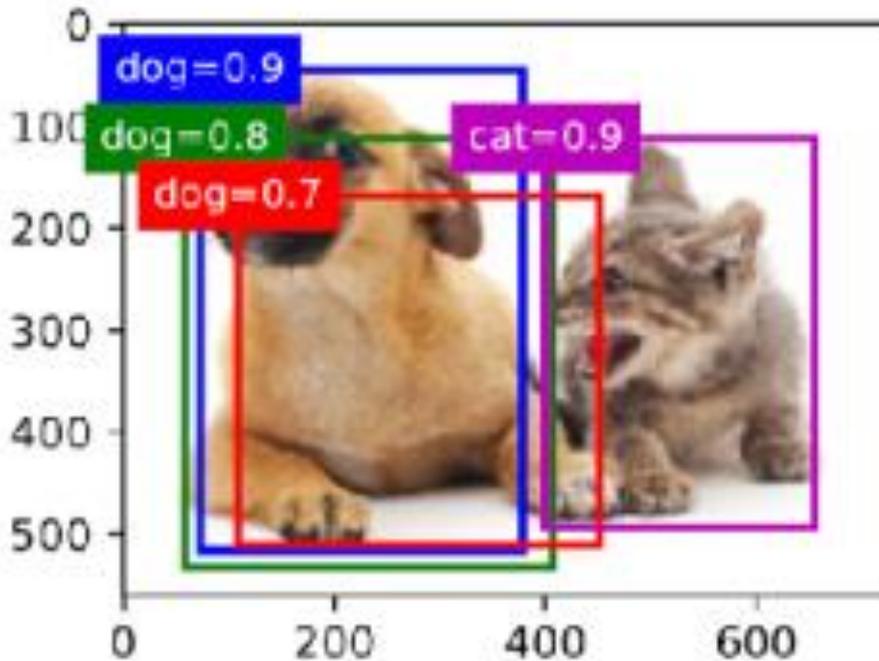- 👉 Complexity: complex pipeline very hard to train

How YOLO Works
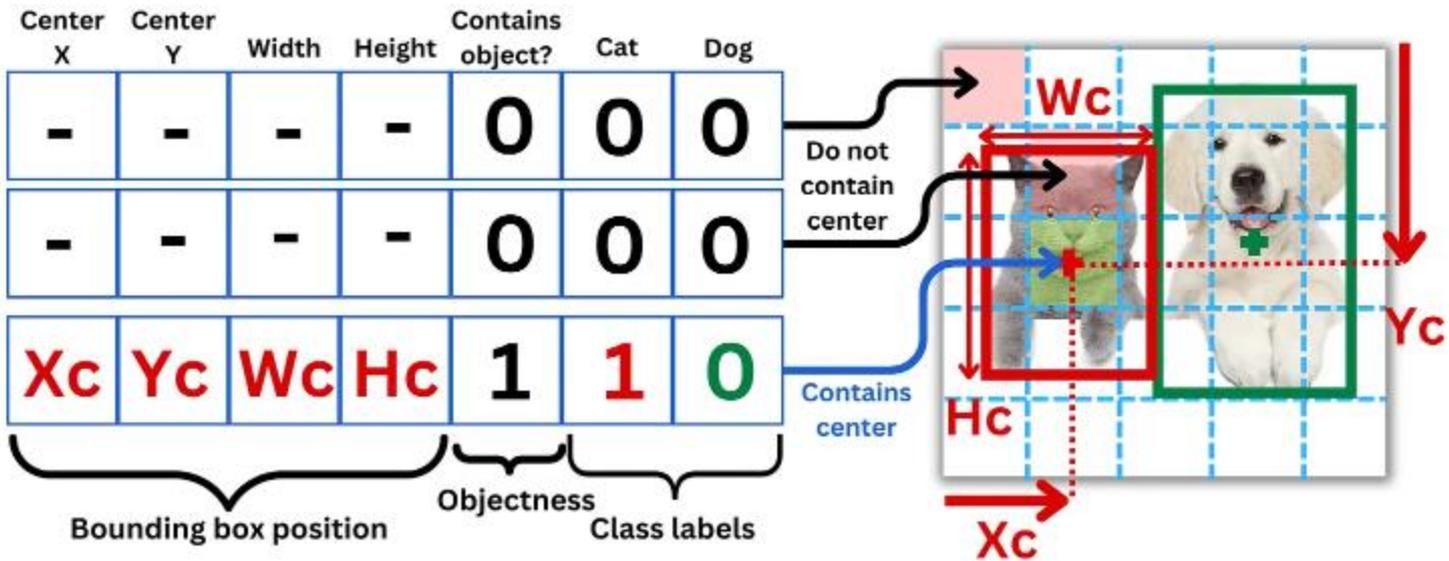
How YOLO Works

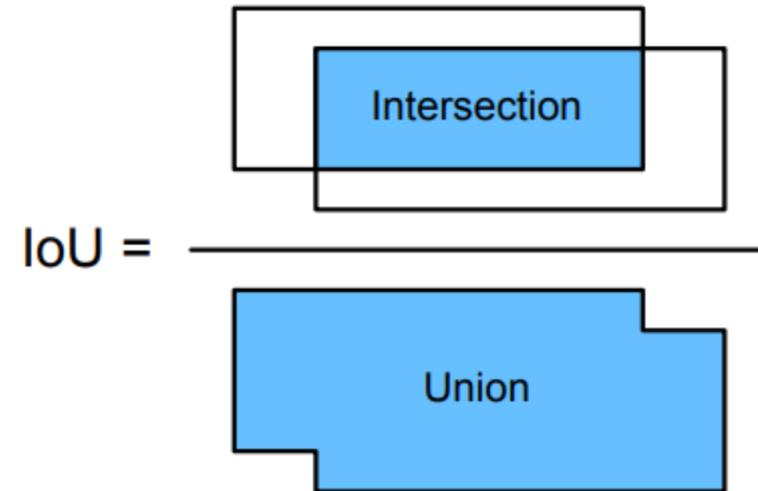- Prediction of Bounding Boxes on each cell
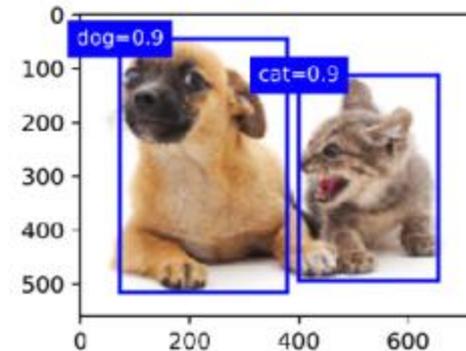
How YOLO Works

- Detection Ambiguities

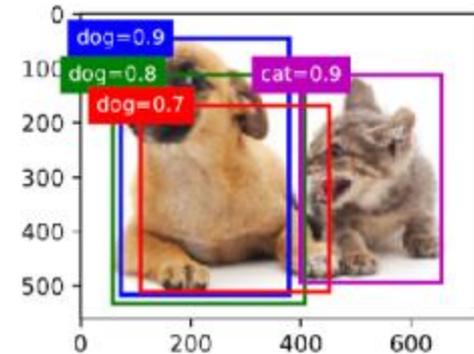# How YOLO Works

- IoU Intersection over union
- IoU measures the similarity between two bounding boxes
  - 0 means no-overlapping
  - 1 means identical

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$$

How Yolo Works

- User defines It's IoU selection threshold

- YOLO computes IoU

- Ignore predictions IoU< threshold

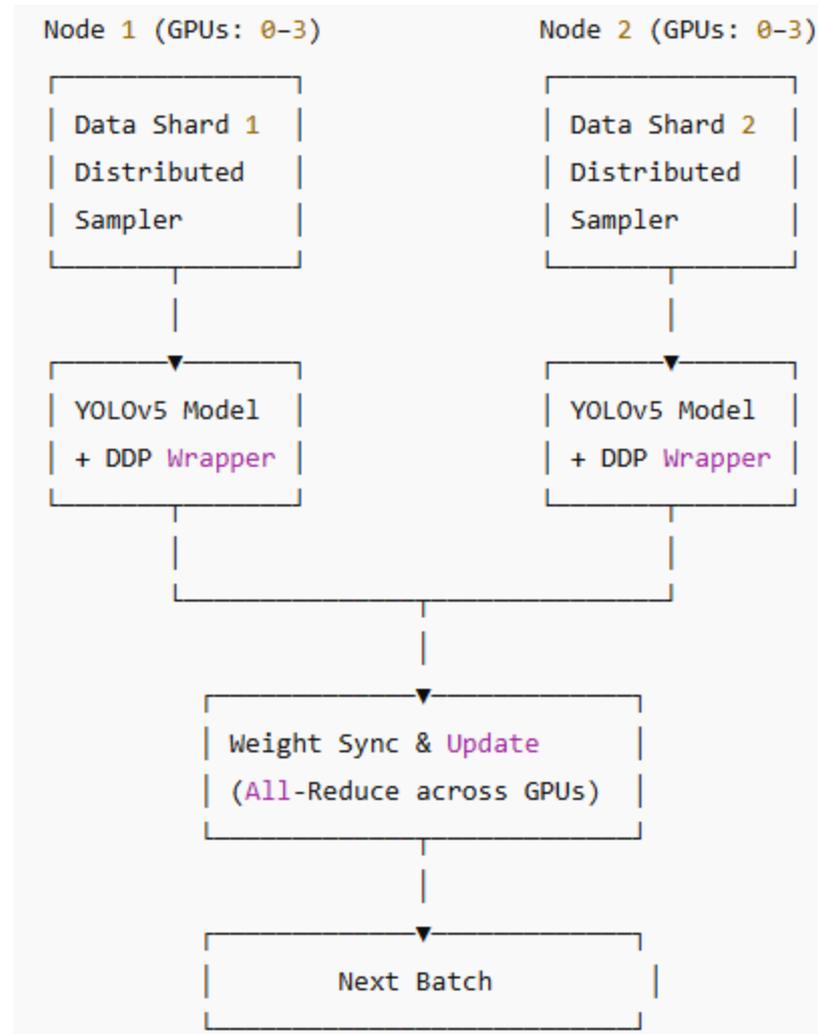- Consider predictions IoU > threshold

How YOLO Works

- Non-Max Suppression or NMS
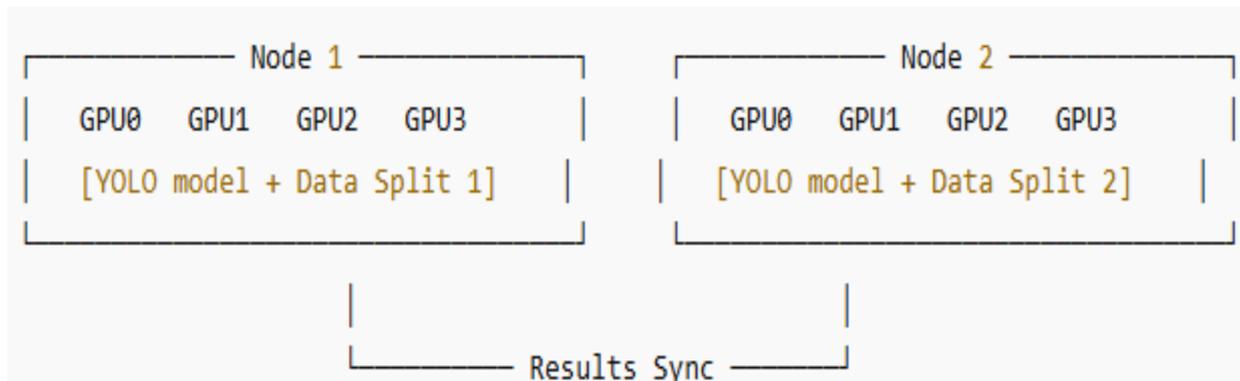- Learning non-maximum suppression Paper

YOLO Overview

- 👉 Strength: Extremely fast, real-time systems

- 👉 Weakness: Lower accuracy on small objects, on overlapping (compared to two-stage methods)

- 👉 Complexity: Simpler pipeline on training

YOLO Retrain In Distributed Mode
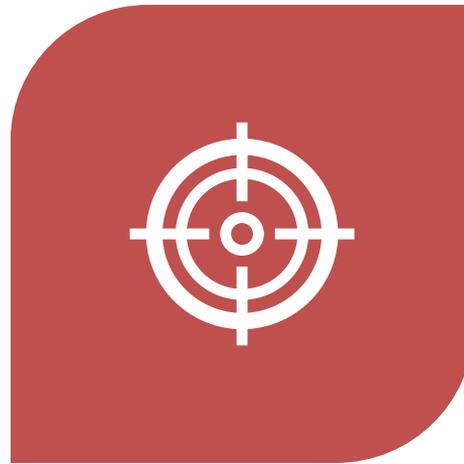
YOLO Inference In Distributed Mode

- No **weights synchronization** is needed
- Each node/GPU just loads the trained YOLO weights and processes its assigned data

```
┌──────────── Node 1 ────────────┐   ┌──────────── Node 2 ────────────┐
│  GPU0   GPU1   GPU2   GPU3      │   │  GPU0   GPU1   GPU2   GPU3      │
│  [YOLO model + Data Split 1]    │   │  [YOLO model + Data Split 2]    │
└────────────────┬───────────────┘   └───────────────┬────────────────┘
                 │                                    │
                 └──────────── Results Sync ──────────┘
```

YOLO

- https://docs.ultralytics.com/models
- https://pytorch.org/
- https://cocodataset.org/#home

Conclusion

THANK YOU