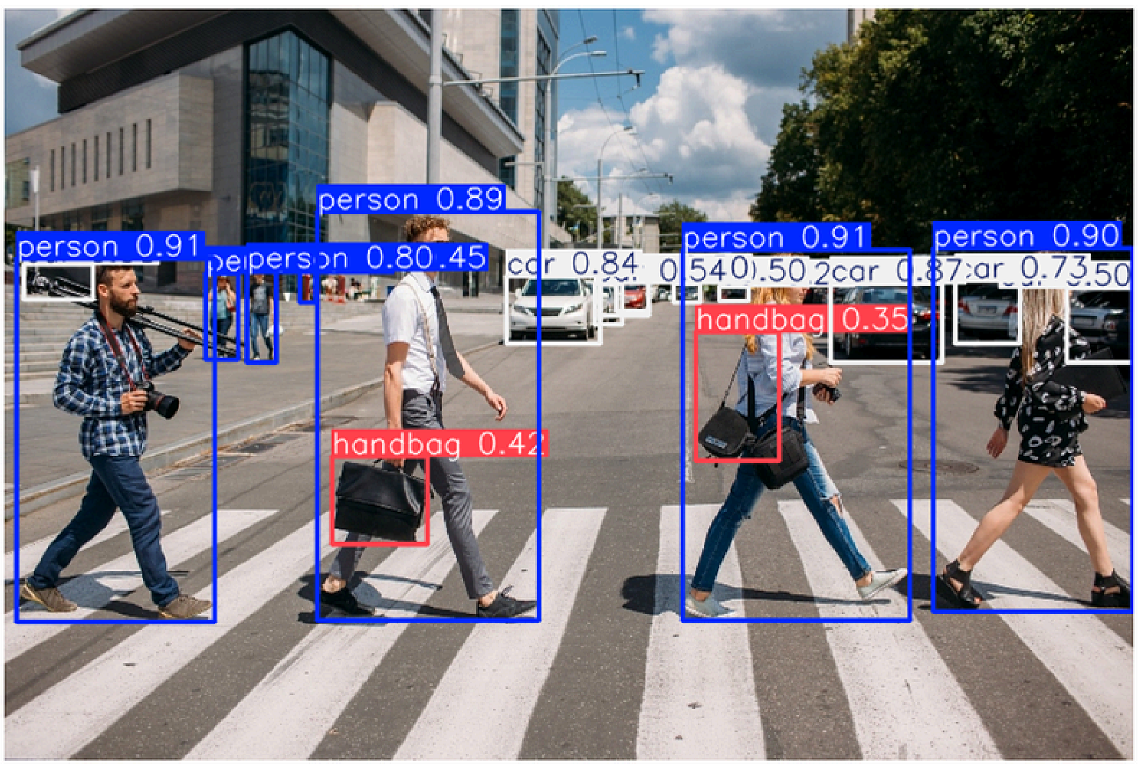
YOLO

📌 Summary of Key Points

Topic	Explanation
What is YOLO?	A fast and real-time object detection algorithm.
What does it do?	Detects **what** objects are in an image and **where** they are.
Why is it important?	It’s extremely **fast** (real-time capable) and **accurate**.
Where is it used?	Self-driving cars, surveillance, robotics, etc.
Main idea?	Treats detection as a **single regression problem**.



🧠What is Object Detection?

Imagine you have a photo, and you want to:

- **Classify** the objects in it (e.g., “dog”, “car”, “person”)
- **Locate** them (i.e., draw a box around each one)

That's object detection. You want:

- **Class label**
- **Bounding box coordinates** (x, y, width, height)

🦁 What is YOLO?

YOLO = You Only Look Once

That means:

> The algorithm looks at the image one time and simultaneously finds all objects and their locations.

❗ Traditional methods (like R-CNN) work in two steps:

1. Propose possible object regions
2. Classify each region

This is **slow**.

✅ YOLO does it in one step, making it:

- **Very fast**
- **Suitable for real-time tasks**

> 💡 **Real-Time Performance**: It's extremely fast — models like **YOLOv4, v5, v8, and YOLO-NAS** can run at real-time FPS on modern GPUs.

🧱 How YOLO Works (Step-by-Step)

🖼️ Step 1: Divide Image into Grid

YOLO divides the input image into an **S × S grid** (say 7×7 for YOLOv1).

Each grid cell:

- Is responsible for detecting objects **whose center falls inside it**
- Predicts:
    - **Bounding boxes**
    - **Confidence score** for each box
    - **Class probabilities**

📦 Step 2: Predict Bounding Boxes

Each grid cell predicts **B bounding boxes**:

- Each box = 5 values:

```
x, y, w, h, confidence
```

- `x`, `y`: center of box (relative to grid cell)
- `w`, `h`: width and height (relative to the whole image)
- `confidence`: how sure the model is that the box contains an object

🏷️ Step 3: Predict Class Probabilities

Each grid cell also predicts:

- **C class probabilities** (e.g., dog, person, car...)

Then combine class probabilities × confidence score to find:

```
final score = class probability × confidence
```

🎯 Step 4: Output Final Detections

- YOLO takes the most confident predictions
- Uses **Non-Maximum Suppression (NMS)** to remove duplicate boxes

🔢 Loss Function (How YOLO learns)

The model minimizes a **custom loss** that includes:

- **Localization loss**: For bounding box position
- **Confidence loss**: How certain the model is about object presence
- **Classification loss**: Accuracy of predicted object class

YOLO uses **Mean Squared Error** + tweaks for better box prediction.

⚙️ Versions of YOLO (Very Important)

Version	What Changed?
YOLOv1	Original paper (2015), fast but less accurate
YOLOv2	Improved accuracy, used anchors, batch norm
YOLOv3	Introduced multi-scale detection, better architecture (Darknet-53)
YOLOv4	Used bag-of-freebies (training tricks) and bag-of-specials (modules)
YOLOv5*	Not official by original authors, but **very popular**, fast, and flexible
YOLOv6/7/8	Further speed and accuracy improvements; YOLOv8 adds segmentation too

> 💡 YOLOv5 is the **most popular and beginner-friendly** version to start with.

💻 Real-World Applications

Use Case	Why YOLO?
Self-driving cars	Real-time object detection
Surveillance	Detect people, threats instantly
Retail analytics	Track customer movement, products
Medical imaging	Detect tumors in scans
Robotics	Detect and interact with objects

YOLO Python Code

```
pip install ultralytics
```

```
from ultralytics import YOLO

# Load pretrained model
model = YOLO('yolov5s.pt')  # 's' means small, also try yolov5m/l/x

# Run on image
results = model('image.jpg')

results
```

```
'carrot', 52: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant', 59: 'bed', 60: 'dining table', 61: 'toilet', 62: 'tv', 63: 'laptop', 64: 'mouse',
65: 'remote', 66: 'keyboard', 67: 'cell phone', 68: 'microwave', 69: 'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator', 73: 'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy
bear', 78: 'hair drier', 79: 'toothbrush'}
 obb: None
 orig_img: array([[[252, 252, 252],
        [252, 252, 252],
        [252, 252, 252],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[252, 252, 252],
        [252, 252, 252],
        [252, 252, 252],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[252, 252, 252],
        [252, 252, 252],
        [252, 252, 252],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,
```

```
results[0].show()
```

Another Model

```
from ultralytics import YOLO

# Load pre-trained model (YOLOv8s = small version)
model2 = YOLO("yolov8s.pt")

# Detect objects
results = model("image.jpg")

# Show results
results[0].show()
```

🧠 Trivia

Model	Speed	Accuracy
YOLO	Fastest	High
SSD	Medium	High
Faster-RCNN	Slow	Highest

Train Custom YOLO Model

```
from ultralytics import YOLO

# Load a base model
model = YOLO("yolov8n.pt")  # nano version
```

```
# Train on your dataset (replace with your data.yaml)
model.train(data="coco128.yaml", epochs=50)
```