

Exp No: 15:- Implement a YOLO Model

to Detect Objects

Aim:-

To implement and understand the YOLO (You only look once) model for real time object detection in images.

Objectives:-

1. To understand the working of YOLO architecture for object detection
2. To detect & localize multiple objects in an image simultaneously
3. To apply pre-trained YOLO weights on a sample dataset or image
4. To visualize bounding boxes & confidence scores.

Algorithm:-

1. Import YOLO pre-trained model (YOLOv3)
2. Load test images for detection
3. preprocess image (resize, normalize)
4. perform object detection using YOLO model
5. Draw bounding boxes & labels on detected objects.
6. Display and analyze output.

Pseudo Code:

Load YOLO pre-trained model

Load class labels

For each image:

 preprocess (resize, normalize)

 pass through YOLO model

 Get bounding boxes, confidence, class ids

 Draw boxes & labels.

Display detected image

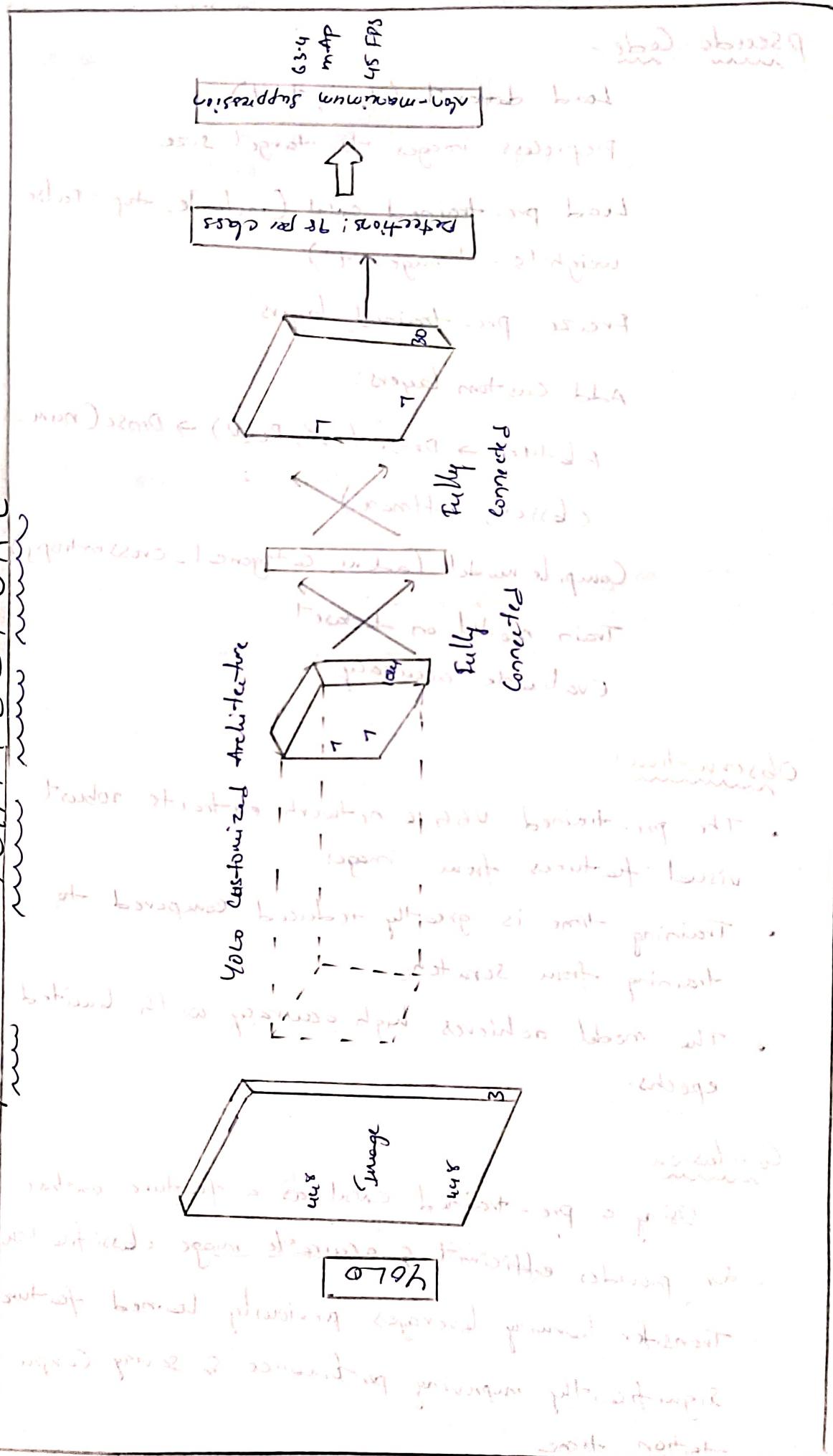
Observations:

- YOLO detects multiple objects with bounding boxes and confidence scores.
- Inference is fast and efficient, suitable for real-time applications.
- Model accurately detects objects even in cluttered scenes.

Conclusion:

The YOLO model demonstrates powerful real-time objects detection capabilities by simultaneously predicting class labels & bounding boxes. Its speed and accuracy make it widely used in autonomous systems, surveillance, & robotics.

YOLO ARCHITECTURE



object detection

TensorRT 7.0.1 + CUDA 10.2 + PyTorch 1.7.0

Output:-

1. 1000 objects bounding boxes from TensorRT at
448x640 (no detections), 68.1ms (N/A)
2. 1000 objects bounding boxes from TensorRT at
Speed : 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

1. 1000 objects bounding boxes from TensorRT at
448x640 (no detections), 68.1ms (N/A)

2. 1000 objects bounding boxes from TensorRT at
Speed : 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

1. 1000 objects bounding boxes from TensorRT at
Speed : 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

1. 1000 objects bounding boxes from TensorRT at
Speed : 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

(Evaluation) TensorRT bounding box proposal at
1000 objects at 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

(Evaluation) TensorRT bounding box proposal at
1000 objects at 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

(Evaluation) TensorRT bounding box proposal at
1000 objects at 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

TensorRT bounding box proposal at
1000 objects at 10.7ms preprocess, 68.1ms inference, 0.8ms
0.8ms postprocess per image at shape (1,3,448,640)

CO dltlab15.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text | ▶ Run all ▾

```
[4] ✓ 3m from google.colab import files
uploaded = files.upload()

Choose Files IMG-20250...WA0017.jpg
IMG-20250523-WA0017.jpg(image/jpeg) - 58650 bytes, last modified: 29/10/2025 - 100% done
Saving IMG-20250523-WA0017.jpg to IMG-20250523-WA0017.jpg

[5] ✓ 1s from ultralytics import YOLO
import cv2

# Step 1: Load a pre-trained YOLO model (v8 nano - fast and small)
model = YOLO('yolov8n.pt')

# Step 2: Load your image (you must have an image in the same folder)
img_path = '/content/IMG-20250523-WA0017.jpg' # <-- replace this with your image filename

# Step 3: Perform object detection
results = model(img_path)

# Step 4: Display detected objects
# Iterate through the results list and show each result
for r in results:
    r.show() # Opens a window with bounding boxes

# Step 5: Print detected class names in console
print("\nDetected Objects:")
for r in results:
    for c in r.boxes.cls:
        print(model.names[int(c)])
```



image 1/1 /content/IMG-20250523-WA0017.jpg: 448x640 (no detections), 68.1ms
Speed: 10.7ms preprocess, 68.1ms inference, 0.8ms postprocess per image at shape (1, 3, 448, 640)



Detected Objects: