EX.NO.: 07
DATE: 24.07.2025

OBJECT DETECTION WITH YOLOv3 AND MASK R-CNN

AIM
To implement and compare object detection using YOLOv3 and Mask R-CNN models for accurately detecting and labeling objects in images, leveraging pre-trained deep learning architectures.

ALGORITHM
For YOLOv3 (You Only Look Once):

1. Load YOLOv3 configuration (`yolov3.cfg`) and weights (`yolov3.weights`).
2. Read the input image and prepare a blob for the model.
3. Perform a forward pass to obtain detection predictions.
4. For each detection:
    - Extract class probabilities and bounding box coordinates.
    - Filter out low-confidence predictions.
5. Apply Non-Maximum Suppression (NMS) to reduce duplicate boxes.
6. Draw final bounding boxes and class labels on the image.

For Mask R-CNN (Region-based CNN):

1. Load the pre-trained Mask R-CNN model (`frozen_inference_graph.pb` and `.pbtxt`).
2. Read the input image and create a blob.
3. Perform a forward pass to obtain:
    - Bounding box predictions
    - Corresponding segmentation masks
4. For each detection with high confidence:
    - Extract class label, box coordinates, and mask.
    - Resize the mask to fit the bounding box.
    - Apply the mask to the image region.
5. Overlay the results with labels, boxes, and masks.

CODE AND OUTPUT
```
import cv2
import numpy as np

# Load YOLO
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Load image
image = cv2.imread("image.jpg")
height, width, channels = image.shape

# Prepare image for YOLO
blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layers)
```

```
# Process detections
class_ids = []
confidences = []
boxes = []
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            # Object detected
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

# Non-max suppression
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

# Draw bounding boxes
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence = confidences[i]
        color = (0, 255, 0)
        cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
        cv2.putText(image, f"{label} {confidence:.2f}", (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

# Save or display result
cv2.imwrite("yolo_output.jpg", image)
# cv2.imshow("YOLO Detection", image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```