**EX.NO.:** 06
**DATE:** 17.07.2025

## DEVELOPMENT OF OBJECT DETECTION MODEL

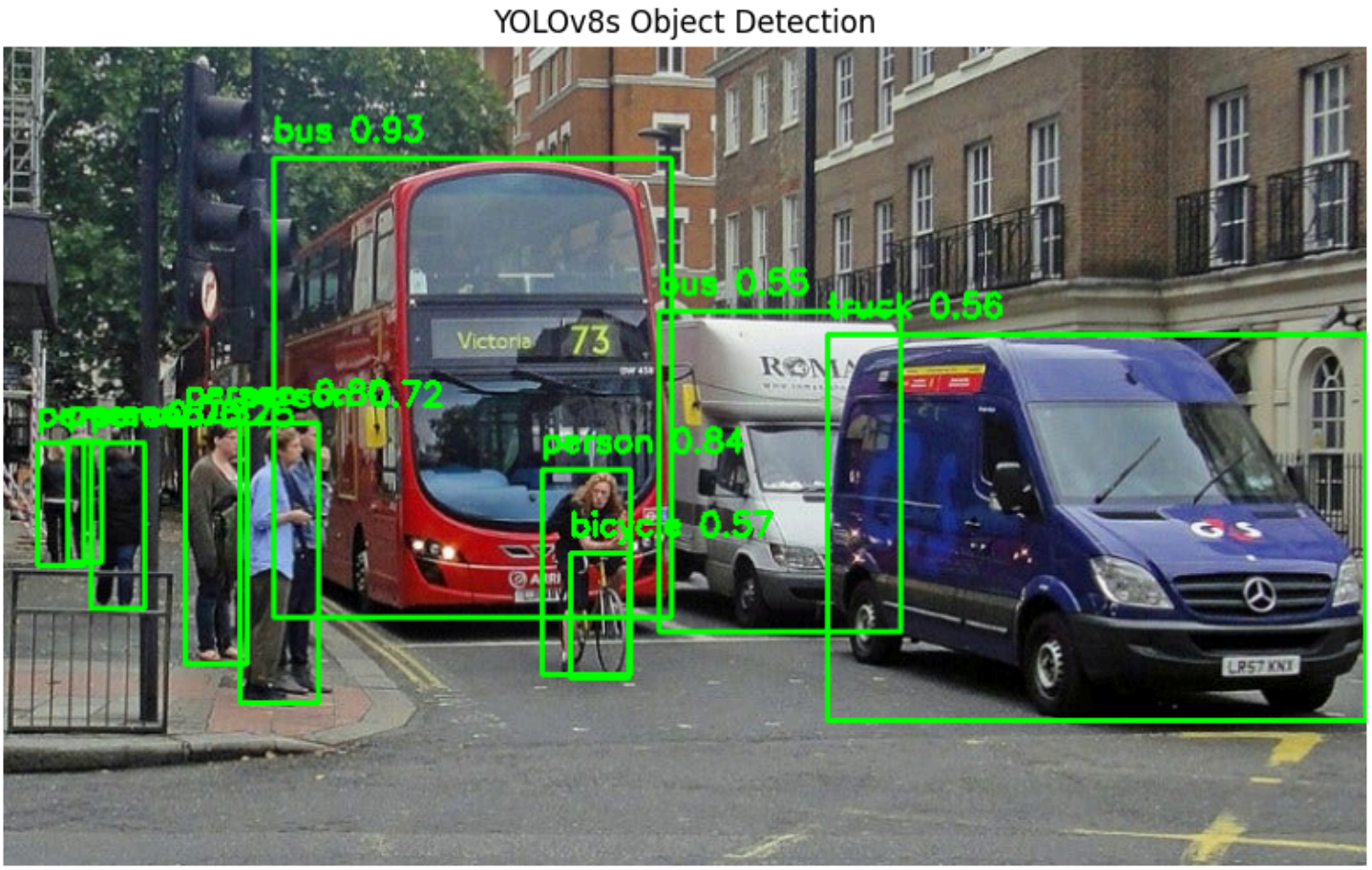
### AIM
To develop and train a custom object detection model using Convolutional Neural Networks (CNN) that can identify and localize objects in an image by predicting bounding boxes and class labels.

### ALGORITHM
1. **Input Preprocessing**
   - Resize input images to a fixed size (e.g., 224×224).
   - Normalize pixel values to [0, 1] range.
2. **CNN-Based Feature Extraction**
   - Apply a sequence of convolutional and pooling layers to extract deep features from the input image.
3. **Flatten and Fully Connected Layers**
   - Flatten the feature map and pass it through dense layers to interpret the extracted features.
4. **Output Layer Prediction**
   - Predict a 5-element vector:
     `[x_center, y_center, width, height, class]`
     - All values are normalized (between 0 and 1).
5. **Loss Calculation**
   - Use **Mean Squared Error (MSE)** loss to compare predicted bounding box and class label with the ground truth.
6. **Model Training**
   - Optimize weights using **Adam optimizer** and backpropagation for multiple epochs.
7. **Prediction and Visualization**
   - Feed a test image to the model.
   - Get predicted bounding box and class.
   - Draw the bounding box on the image for visualization.

### CODE AND OUTPUT
```
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
import os
from collections import Counter

# Load YOLOv8s model (better accuracy than 'n')
model = YOLO("yolov8s.pt")

# Set image path
image_path = "image.jpg"  # <-- Replace with your actual image filename

# Check file exists
if not os.path.exists(image_path):
    raise FileNotFoundError(f"Image not found at: {image_path}")

# Read image
image = cv2.imread(image_path)
if image is None:
```

```
    raise ValueError("Failed to read image. Check file format or path.")

# Run detection
results = model(image)[0]

# For object count
detected_labels = []

# Loop through detected boxes
for i, box in enumerate(results.boxes):
    conf = float(box.conf[0])
    if conf < 0.5:
        continue  # skip low confidence predictions

    x1, y1, x2, y2 = map(int, box.xyxy[0])
    cls_id = int(box.cls[0])
    label = model.names[cls_id]

    # Save label for counting
    detected_labels.append(label)

    # Draw bounding box and label
    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(image, f"{label} {conf:.2f}", (x1, y1 - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

    # Save cropped object
    cropped_object = image[y1:y2, x1:x2]
    cv2.imwrite(f"cropped_{label}_{i}.jpg", cropped_object)

    # Optional: Sound alert if a person is detected (Windows only)
    if label == "person":
        try:
            import winsound
            winsound.Beep(1000, 400)  # Beep sound
        except:
            pass  # Ignore if not on Windows

# Print object counts
object_counts = Counter(detected_labels)
print("Detected objects:", dict(object_counts))

# Save the output image
output_path = "output_detected.jpg"
cv2.imwrite(output_path, image)
print(f"Output image saved to: {output_path}")

# Show image using matplotlib
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
plt.figure(figsize=(10, 8))
plt.imshow(image_rgb)
plt.axis('off')
plt.title("YOLOv8s Object Detection")
plt.show()
```




**INFERENCE**

The model learns to detect and localize objects using only image input and bounding box annotations. It outputs bounding box coordinates and class probabilities, even for new, unseen images. With sufficient training data and proper annotations, this model can be scaled to multiple classes and real-world object detection tasks.