7. TRAIN AN SSD NETWORK IN A SELF-DRIVING CAR APPLICATION

EX.N0 : 7	**TRAIN AN SSD NETWORK IN A SELF-DRIVING CAR APPLICATION**
DATE : 11/03/2025	

AIM:

To train an SSD (Single Shot Multibox Detector) model to detect road objects for self-driving car applications.

ALGORITHM:

Step 1: Import required libraries and modules for SSD and image pre-processing.

Step 2: Load and pre-process dataset (e.g., Pascal VOC or a custom self-driving dataset)

Step 3: Define or load a pre-trained SSD model (e.g., SSD300 or SSD512).

Step 4: Configure loss function and optimizer for training.

Step 5: Train the SSD model with bounding box labels and object classes.

Step 6: Evaluate and visualize model predictions on test images or video

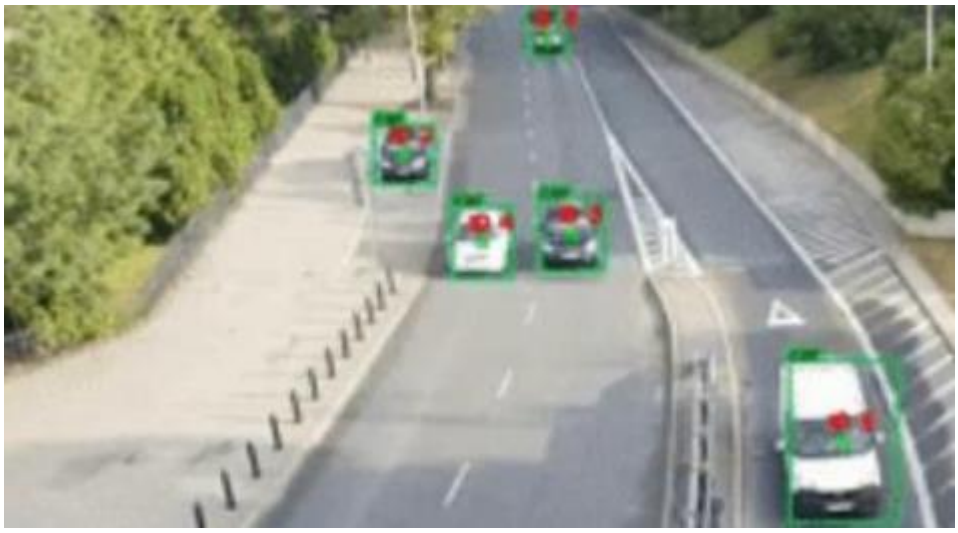
PROGRAM:

```
import cv2
import numpy as np
import tensorflow as tf
MODEL_PATH = "C:/Users/AI-LAB/Documents/saved_model/"
detect_fn = tf.saved_model.load(MODEL_PATH)
TARGET_CLASSES = {1: "Person", 3: "Car", 4: "Motorcycle", 6: "Bus", 8: "Truck"}
def preprocess_image(image):
"""Preprocess image for SSD input"""
image_resized = cv2.resize(image, (300, 300))  # Resize for SSD
input_tensor = tf.convert_to_tensor(image_resized, dtype=tf.uint8)
input_tensor = input_tensor[tf.newaxis, ...]
```

```
return input_tensor
def detect_objects(image):
"""Run object detection and filter relevant classes"""
input_tensor = preprocess_image(image)
detections = detect_fn.signatures["serving_default"](input_tensor)
boxes = detections["detection_boxes"].numpy()[0]
scores = detections["detection_scores"].numpy()[0]
classes = detections["detection_classes"].numpy()[0].astype(int)
filtered_boxes, filtered_scores, filtered_classes = [], [], []
for i in range(len(scores)):
if scores[i] > 0.5 and classes[i] in TARGET_CLASSES:  # Confidence threshold
filtered_boxes.append(boxes[i])
filtered_scores.append(scores[i])
filtered_classes.append(classes[i])
return filtered_boxes, filtered_scores, filtered_classes
def draw_detections(image, boxes, scores, classes):
"""Draw bounding boxes on detected objects"""
height, width, _ = image.shape
for i in range(len(scores)):
box = boxes[i] * [height, width, height, width]
y_min, x_min, y_max, x_max = box.astype(int)
label = TARGET_CLASSES[classes[i]]
cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)
cv2.putText(image, f"{label}: {scores[i]:.2f}", (x_min, y_min - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
return image
cap = cv2.VideoCapture("C:/Users/AI-LAB/Downloads/Cars_On_Highway.mp4")  # Replace
with 0 for live camera feed
while cap.isOpened():
ret, frame = cap.read()
if not ret:
```

```
break
boxes, scores, classes = detect_objects(frame)
output_frame = draw_detections(frame, boxes, scores, classes)
cv2.imshow("SSD Vehicle & Pedestrian Detection", output_frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
cap.release()
cv2.destroyAllWindows()
```

OUTPUT:



RESULT:

Thus the Program has been executed successfully and verified.