# 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.