EX NO: 8	**Object Detection with Single Shot Detector**

Aim:

To implement and train an **SSD (Single Shot MultiBox Detector)** for object detection using PyTorch. This will involve detecting objects in images, such as cars, pedestrians, and other relevant classes.

Algorithm:

1. **Install Dependencies**: Install required libraries such as PyTorch, torchvision, and other helper libraries.

2. **Load Pre-trained Model**: Use a pre-trained SSD model from PyTorch's model zoo (e.g., SSD with MobileNet backbone).

3. **Dataset Preparation**: Use an object detection dataset like Pascal VOC or COCO with annotations in the correct format.

4. **Preprocess Data**: Resize and normalize images, as well as convert bounding box coordinates into SSD format.

5. **Define SSD Model**: Load the pre-trained model with a MobileNet or VGG backbone.

6. **Training**: Train the model on the dataset, defining loss functions such as **Multibox loss**.

7. **Evaluate**: Validate the model's performance on test images or a test set.

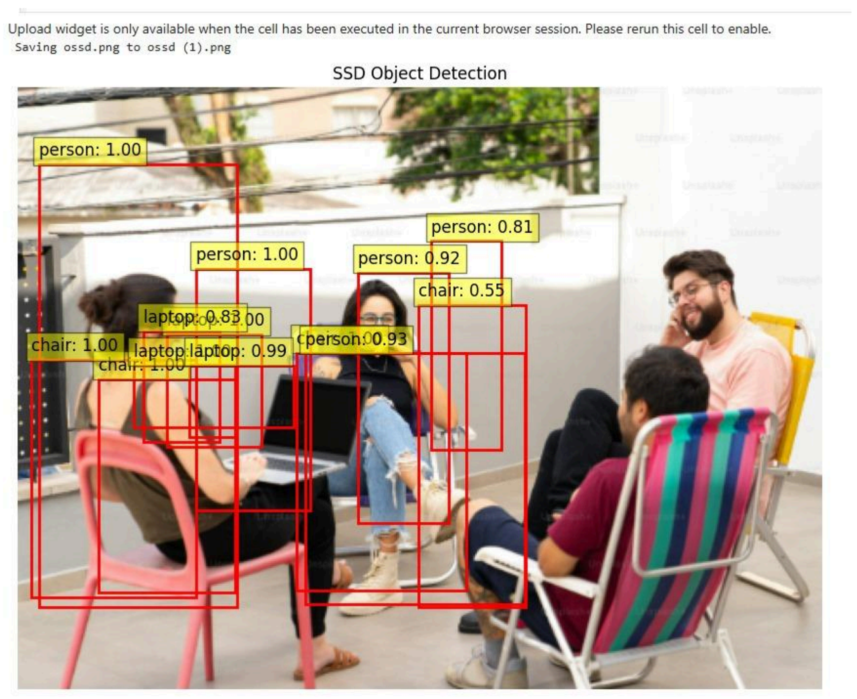
8. **Visualize the Results**: Show the predicted bounding boxes and labels on the test images.

Code:

```
import torch

import torchvision

from torchvision import transforms

import cv2

import numpy as np

import matplotlib.pyplot as plt

model = torchvision.models.detection.ssdlite320_mobilenet_v3_large(pretrained=True)

model.eval()

transform = transforms.Compose([

    transforms.ToTensor(),

])

image_path = 'image.jpg'  # Replace with your image path

image = cv2.imread(image_path)

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB

image_resized = cv2.resize(image_rgb, (320, 320))  # Resize to match the SSD input size

image_tensor = transform(image_resized)

image_tensor = image_tensor.unsqueeze(0)

with torch.no_grad():

    prediction = model(image_tensor)

boxes = prediction[0]['boxes'].cpu().numpy()

labels = prediction[0]['labels'].cpu().numpy()

scores = prediction[0]['scores'].cpu().numpy()
```

```
threshold = 0.5

boxes = boxes[scores > threshold]

for box, label in zip(boxes, labels):

   x1, y1, x2, y2 = box.astype(int)

   cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

     cv2.putText(image, f'Class {label}', (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
(255, 0, 0), 2)

plt.figure(figsize=(12, 8))

plt.imshow(image_rgb)

plt.axis('off')

plt.title("SSD Object Detection")

plt.show()
```

Output:



```
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
 Saving ossd.png to ossd (1).png
```

Result:

The SSD model detects objects in the image, drawing bounding boxes around detected objects, such as cars, pedestrians, or other classes in the image. Each box is labeled with the object class and confidence score.