Experiment- 7

Student Name: Kush Bhasin **UID:** 20BCS7677
Branch: BE(CSE) **Section/Group:** 20BCS-1/B
Semester: 7th
Subject Name: Computer Vision Lab
Subject Code: 20CSP- 422

1. **Aim:** Write a program to analyze various object detection algorithms with machine learning.

2. **System Requirements:**
   - Python 3.9
   - Visual Studio Code

3. **Description:**
   Object detection is a fundamental task in computer vision and has numerous applications in fields like autonomous driving, surveillance, and image recognition. There are several object detection algorithms that leverage machine learning techniques to detect and localize objects in images.
   1. R-CNN (Region-based Convolutional Neural Networks): R-CNN is an early object detection algorithm that uses a two-stage approach. It first generates region proposals using techniques like Selective Search and then extracts features from these regions using a CNN. These features are fed into an SVM classifier to determine the presence of objects and their bounding box coordinates.
   2. Fast R-CNN: Fast R-CNN improves upon R-CNN by introducing a shared convolutional feature map for the entire image instead of processing each region proposal independently. These speeds up the computation and allows for end-to-end training. It also replaces the SVM classifier with a softmax layer for object classification.
   3. Faster R-CNN: Faster R-CNN takes the speed improvement further by introducing a Region Proposal Network (RPN) that generates region proposals directly from the convolutional feature map, eliminating the need for external region proposal methods. This results in a unified model that jointly learns region proposals and object classification.
   4. YOLO (You Only Look Once): YOLO is a single-stage object detection algorithm that divides the input image into a grid and predicts bounding boxes and class probabilities directly from each grid cell. YOLO can achieve real-time object detection due to its efficient architecture and simultaneous prediction of multiple objects in a single pass.
   5. SSD (Single Shot MultiBox Detector): SSD is another single-stage object detection algorithm that operates at multiple scales. It applies a set of default bounding boxes with different aspect ratios to the feature maps at various scales and predicts class probabilities and offsets for these boxes.SSD achieves a good balance between speed and accuracy.

6. RetinaNet: RetinaNet introduces a novel focal loss that addresses the class imbalance problem in object detection. It utilizes a feature pyramid network and a classification subnetwork to predict objectness scores and refine the bounding box coordinates. The focal loss assigns higher weights to hard examples, leading to improved performance, especially for detecting objects in the presence of many background regions.
7. Mask R-CNN: Mask R-CNN extends Faster R-CNN by adding an additional branch for pixel-wise segmentation masks. In addition to object detection and bounding box regression, Mask R-CNN can generate high-quality instance masks for each detected object. It is widely used in tasks that require precise object segmentation.
8. EfficientDet: EfficientDet is a family of efficient and accurate object detection models that achieve a good trade-off between model size and performance. These models employ efficient backbone architectures, compound scaling techniques, and advanced network design principles to achieve state-of-the-art accuracy with fewer computational resources.

4. Steps:

- Gather a dataset of labeled images for object detection (e.g., COCO dataset).
- Preprocess the dataset by resizing and normalizing the images.
- Split the dataset into training and testing sets.
- Choose and implement different object detection algorithms (e.g., Faster R-CNN, YOLO, SSD).
- Train each algorithm on the training set using the machine learning library.
- Evaluate the performance of each algorithm on the testing set using appropriate metrics such as precision, recall, and mean average precision (mAP).
- Analyze and compare the results obtained from different algorithms.
- Draw conclusions about the strengths, weaknesses, and trade-offs of each algorithm.

5. Code:

```
import cv2
import matplotlib.pyplot as plt
from PIL import Image
from torchvision import transforms
from torchvision.models.detection import fasterrcnn_resnet50_fpn,
FasterRCNN_ResNet50_FPN_Weights
model = fasterrcnn_resnet50_fpn(weights=FasterRCNN_ResNet50_FPN_Weights.DEFAULT)
model.eval()

CATEGORY = ['__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane',
'bus', 'train',
            'truck', 'boat', 'traffic light', 'fire hydrant', 'N/A', 'stop sign',
'parking meter', 'bench', 'bird', 'cat', 'dog',
            'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'N/A',
'backpack', 'umbrella', 'N/A', 'N/A',
            'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports
ball', 'kite', 'baseball bat', 'baseball glove',
```

```
            'skateboard', 'surfboard', 'tennis racket', 'bottle', 'N/A', 'wine
glass', 'cup', 'fork', 'knife',
            'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli',
'carrot', 'hot dog', 'pizza', 'donut',
            'cake', 'chair', 'couch', 'potted plant', 'bed', 'N/A', 'dining table',
'N/A', 'N/A', 'toilet', 'N/A', 'tv', 'laptop',
            'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven',
'toaster', 'sink', 'refrigerator', 'N/A',
            'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier',
'toothbrush']

def predict(img_path, threshold):
  img = Image.open(img_path)
  transform = transforms.Compose([transforms.ToTensor()])
  img = transform(img)
  pred = model([img])
  pred_class = [CATEGORY[i] for i in list(pred[0]['labels'].numpy())]
  pred_boxes = [[(i[0], i[1]), (i[2], i[3])] for i in
list(pred[0]['boxes'].detach().numpy())]
  pred_score = list(pred[0]['scores'].detach().numpy())
  pred_t = [pred_score.index(x) for x in pred_score if x>threshold][-1]
  pred_boxes = pred_boxes[:pred_t+1]
  pred_class = pred_class[:pred_t+1]
  return pred_boxes, pred_class

def detect_objects(img_path, threshold=0.5, rect_th=3, text_size=3, text_th=3):
  boxes, pred_cls = predict(img_path, threshold)
  img = cv2.imread(img_path)
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  for i in range(len(boxes)):
    cv2.rectangle(img, (int(boxes[i][0][0]), int(boxes[i][0][1])),
(int(boxes[i][1][0]), int(boxes[i][1][1])), color=(0, 255, 0), thickness=rect_th)
    cv2.putText(img, pred_cls[i], (int(boxes[i][0][0]), int(boxes[i][0][1])),
cv2.FONT_HERSHEY_SIMPLEX, text_size, (0,255,0),thickness=text_th)
  plt.figure(figsize=(20,30))
  plt.imshow(img)
  plt.xticks([])
  plt.yticks([])
  plt.show()

detect_objects('street.jpg')
```

6. Output: