Assignment 4

Object Detection

Team Members

Aya Khames Khairy	18010442
Basel Ayman Mohamed	18010458
Pancee Wahid Mohamed	18010467

Models

1. Faster R-CNN:

It consists of two modules: the first module is a deep CNN that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions.

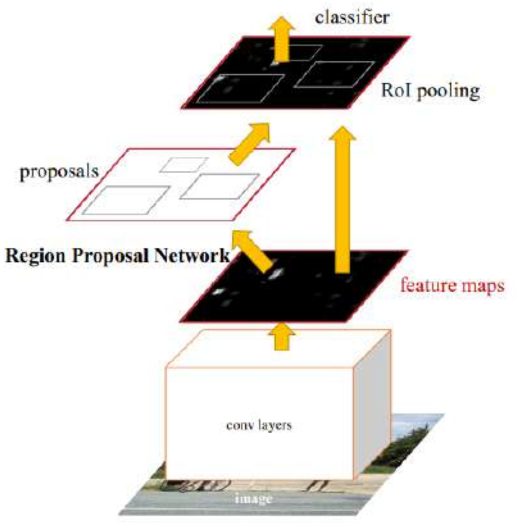


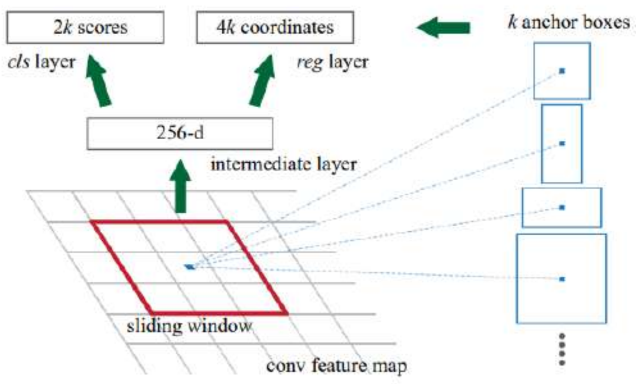
Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Region Proposal Network (RPN):

It takes an image of any size as input and outputs a set of rectangular object proposals, each with an objectness score.

To generate region proposals, they slide a small network over the convolutional feature map output by the last shared convolutional layer.

At each sliding-window location, it predicts multiple region proposals (the maximum number of possible proposals for each location is k). So, the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate probability of object or not object for each proposal. The k proposals are parameterized relative to k reference boxes (anchors).



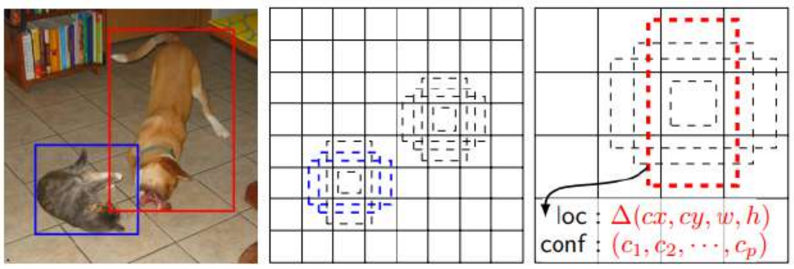
Sharing Features for RPN and Fast R-CNN:

Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways. A technique is developed to allow sharing of convolutional layers between the two networks, rather than learning two separate networks.

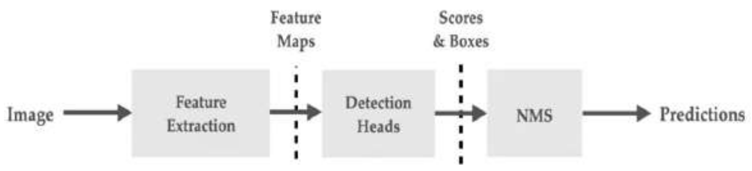
Some RPN proposals highly overlap with each other. To reduce redundancy, non-maximum suppression (NMS) on the proposal regions based on their cls scores is adopted.

2. Single Shot MultiBox Detector (SSD):

SSD completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computations in a single network, and is as accurate as the approaches that do them in separate stages.



loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$



Base Network(Feature Extraction): the VGG-16 network is used as a base, but other networks should also produce good results like ResNet, these earlier layers are based on a standard architecture used for high quality image classification.

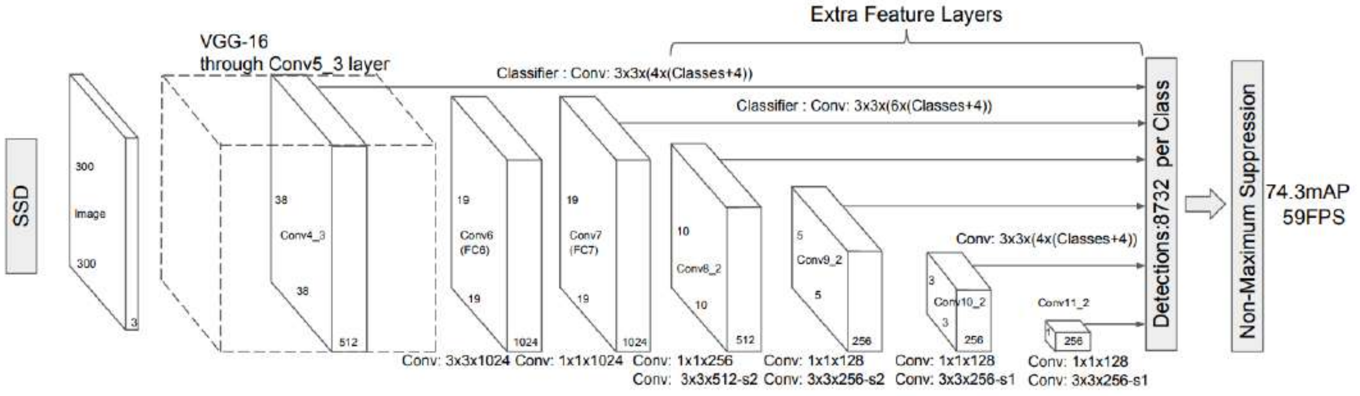
Multi-Scale Feature Maps for Detection(Feature Extraction): convolutional feature layers are added to the end of the truncated base network, they decrease in size progressively and allow predictions of detections at multiple scales, the convolutional model for predicting detections is different for each feature layer.

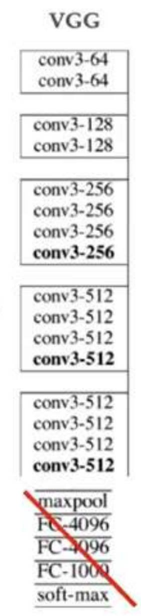
Convolutional Predictors for Detection(Detection Heads): Each added feature layer (or optionally an existing feature layer from the base network) of size (m × n × p) can produce a fixed set of detection predictions using a set of convolutional filters of size (3 × 3 × p) these small kernels produce either a score for a category, or a shape offset relative to the default box coordinates.

Default Boxes and Aspect Ratios: a set of default bounding boxes(k) is associated with each feature map cell, for multiple feature maps at the top of the network, at each feature map cell, predict the offsets relative to the default box shapes(4) in the cell, as well as the per-class scores(c) that indicate the presence of a class instance in each of those boxes → k(4 + c) for each cell in the different feature maps scales.

Combining predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes, and allowing different default box shapes in several feature maps lets us efficiently discretize the space of possible output box shapes.

Non-Maximum Suppression(NMS): in this stage the multiple detections for the same object is eliminated to one detection.





The VGG is used without the final stage of preparing for classification to get the resulting output as a good starting point for the feature maps.

3. RetinaNet Detector:

RetinaNet is a single, unified network composed of a *backbone* network and two task-specific *subnetworks*. The backbone is responsible for computing a convolutional feature map over an entire input image and is an off-the-self convolutional network. The first subnet performs convolutional object classification on the backbone's output; the second subnet performs convolutional bounding box regression.

Feature Pyramid Network Backbone: In brief, FPN augments a standard convolutional network with a top-down pathway and lateral connections so the network efficiently constructs a rich, multi-scale feature pyramid from a single resolution input image. Each level of the pyramid can be used for detecting objects at a different scale. While many design choices are not crucial, they emphasize the use of the FPN backbone is; preliminary experiments using features from only the final ResNet layer yielded low AP.

Anchors: In total there are 9 anchors per level. Each anchor is assigned a length K one-hot vector of classification targets, where K is the number of object classes, and a 4-vector of box regression targets. Specifically, anchors are assigned to ground-truth object boxes using an intersection-over-union (IoU) threshold of 0.5; and to background if their IoU is in [0, 0.4]. As each anchor is assigned to at most one object box, the corresponding entry in its length K label vector is set to 1 and all other entries to 0.

Classification Subnet: The classification subnet predicts the probability of object presence at each spatial position for each of the A anchors and K object classes. This subnet is a small FCN attached to each FPN level; parameters of this subnet are shared across all pyramid levels. Its design is simple. Taking an input feature map with C channels from a given pyramid level, the subnet applies four 3×3 conv layers, each with C filters and each followed by ReLU activations, followed by a 3×3 conv layer with KA filters. Finally sigmoid activations are attached to output the KA binary predictions per spatial location.

Box Regression Subnet: In parallel with the object classification subnet, another small FCN is attached to each pyramid level for the purpose of regressing the offset from each anchor box to a nearby ground-truth object, if one exists. The design of the box regression subnet is identical to the classification subnet except that it terminates in 4A linear outputs per spatial location. The object classification subnet and the box regression subnet, though sharing a common structure, use separate parameters.

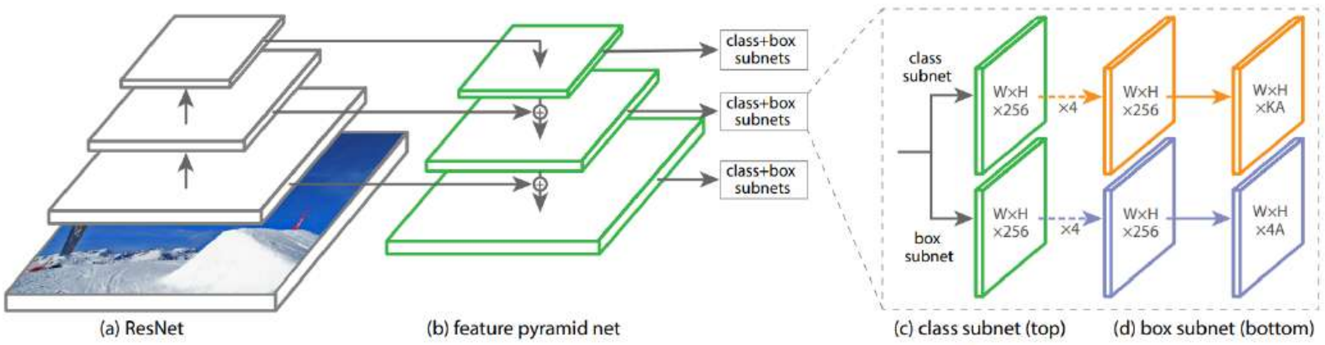


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

Datasets

We used:

COCO 2017 → Validation set
PASCAL VOC 2012 → Validation set

Used Datasets	COCO 2017	PASCAL VOC 2012
Description	Large-scale variations, segmentation, and captioning dataset	Generic scenes, large-pose variations, and cleaned up performance metric
# of Categories	91	20
# of Objects	Many per image	Few per image
Occlusions	Heavy	Heavy

Samples in PASCAL VOC 2012 are easier and have fewer objects than COCO 2017 which make the models perform better on its samples as will be shown in results.

Fifty-one

We used fifty-one which is an open-source tool for building high-quality datasets and computer vision models. We used it to load and show datasets and navigate them easily through fifty-one GUI which also helps in showing bounding boxes of ground truths and predictions of the models.

Code

Installations and Imports

```
[2]  !pip uninstall flask
     !pip install fiftyone
     !pip install torch torchvision
     !pip uninstall urllib3
     !pip install urllib3
     !pip install ipywidgets>=7.5
```

```
[3]  import cv2
     from google.colab.patches import cv2_imshow
     import numpy as np
     from matplotlib import pyplot as plt
     import math
     import torch
     import torchvision
     import fiftyone as fo
     import fiftyone.zoo as foz
     from fiftyone import ViewField as F
     from PIL import Image
     from torchvision.transforms import functional as func
```

```
[ ]  # Run the model on GPU if it is available
     device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

Loading COCO 2017 validation dataset

```
[ ]  # Load COCO 2017 validation set
     coco = foz.load_zoo_dataset(
         "coco-2017",
         split="validation"
     )
     coco.persistent = True

     # Get class list
     coco_classes = coco.default_classes
```

```
[ ]  coco_ds = coco.take(1000, seed=51)
```

```
[ ]  print(coco_ds)
```

```
Dataset:     coco-2017-validation
Media type:  image
Num samples: 1000
Sample fields:
    id:                  fiftyone.core.fields.ObjectIdField
    filepath:            fiftyone.core.fields.StringField
    tags:                fiftyone.core.fields.ListField(fiftyone.core.fields.StringField)
    metadata:            fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.metadata.ImageMetadata)
    ground_truth:        fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    faster_rcnn:         fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    ssd:                 fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    retinanet:           fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    eval_faster_RCNN_tp: fiftyone.core.fields.IntField
    eval_faster_RCNN_fp: fiftyone.core.fields.IntField
    eval_faster_RCNN_fn: fiftyone.core.fields.IntField
    eval_ssd_tp:         fiftyone.core.fields.IntField
    eval_ssd_fp:         fiftyone.core.fields.IntField
    eval_ssd_fn:         fiftyone.core.fields.IntField
    eval_retinanet_tp:   fiftyone.core.fields.IntField
    eval_retinanet_fp:   fiftyone.core.fields.IntField
    eval_retinanet_fn:   fiftyone.core.fields.IntField
View stages:
    1. Take(size=1000, seed=51)
```

Loading PASCAL VOC 2012 validation dataset

```
[ ] voc = foz.load_zoo_dataset(
        "voc-2012",
        split="validation"
    )
    voc.persistent = True

    # Get class list
    voc_classes = voc.default_classes
```

```
Downloading split 'validation' to '/root/fiftyone/voc-2012/validation'
INFO:fiftyone.zoo.datasets:Downloading split 'validation' to '/root/fiftyone/voc-2012/validation'
Downloading http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar to /root/fiftyone/voc-2012/tmp-downlo
100% ████████████████████ 1999639040/1999639040 [02:04<00:00, 20940751.19it/s]
Extracting /root/fiftyone/voc-2012/tmp-download/VOCtrainval_11-May-2012.tar to /root/fiftyone/voc-2012/tmp-download
 100% |███████████████| 5823/5823 [1.3m elapsed, 0s remaining, 76.8 samples/s]
INFO:eta.core.utils: 100% |███████████████| 5823/5823 [1.3m elapsed, 0s remaining, 76.8 samples/s]
Dataset info written to '/root/fiftyone/voc-2012/info.json'
INFO:fiftyone.zoo.datasets:Dataset info written to '/root/fiftyone/voc-2012/info.json'
Loading 'voc-2012' split 'validation'
INFO:fiftyone.zoo.datasets:Loading 'voc-2012' split 'validation'
 100% |███████████████| 5823/5823 [17.4s elapsed, 0s remaining, 269.5 samples/s]
INFO:eta.core.utils: 100% |███████████████| 5823/5823 [17.4s elapsed, 0s remaining, 269.5 samples/s]
Dataset 'voc-2012-validation' created
INFO:fiftyone.zoo.datasets:Dataset 'voc-2012-validation' created
```

```
[ ] voc_ds = voc.take(1000, seed=51)
```

```
[ ] print(voc_ds)
```

```
Dataset:     voc-2012-validation
Media type:  image
Num samples: 1000
Sample fields:
    id:                  fiftyone.core.fields.ObjectIdField
    filepath:            fiftyone.core.fields.StringField
    tags:                fiftyone.core.fields.ListField(fiftyone.core.fields.StringField)
    metadata:            fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.metadata.ImageMetadata)
    ground_truth:        fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    faster_rcnn:         fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    ssd:                 fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    retinanet:           fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
    eval_faster_RCNN_tp: fiftyone.core.fields.IntField
    eval_faster_RCNN_fp: fiftyone.core.fields.IntField
    eval_faster_RCNN_fn: fiftyone.core.fields.IntField
    eval_ssd_tp:         fiftyone.core.fields.IntField
    eval_ssd_fp:         fiftyone.core.fields.IntField
    eval_ssd_fn:         fiftyone.core.fields.IntField
    eval_retinanet_tp:   fiftyone.core.fields.IntField
    eval_retinanet_fp:   fiftyone.core.fields.IntField
    eval_retinanet_fn:   fiftyone.core.fields.IntField
View stages:
    1. Take(size=1000, seed=51)
```

Loading pre-trained models

We are using Faster R-CNN, SSD, and RetinaNet from PyTorch.

```
[ ] # Load a pre-trained Faster R-CNN model
    faster_RCNN = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
    faster_RCNN.to(device)
    faster_RCNN.eval()

    # Load a pre-trained SSD model
    ssd = torchvision.models.detection.ssd300_vgg16(pretrained=True)
    ssd.to(device)
    ssd.eval()

    # Load a pre-trained RetinaNet model
    retinaNet = torchvision.models.detection.retinanet_resnet50_fpn_v2(pretrained=True)
    retinaNet.to(device)
    retinaNet.eval()
```

Get detections

```
[ ]  def get_detections(model, image):
       c, h, w = image.shape
       preds = model([image])[0]
       labels = preds["labels"].cpu().detach().numpy()
       scores = preds["scores"].cpu().detach().numpy()
       boxes = preds["boxes"].cpu().detach().numpy()

       detections = []
       for label, score, box in zip(labels, scores, boxes):
         # Convert to [top-left-x, top-left-y, width, height] in relative coordinates in [0, 1] x [0, 1]
         x1, y1, x2, y2 = box
         rel_box = [x1 / w, y1 / h, (x2 - x1) / w, (y2 - y1) / h]

         detections.append(
             fo.Detection(
                 label=coco_classes[label],
                 bounding_box=rel_box,
                 confidence=score
             )
         )

       return detections
```

```
[ ]  def predict(ds):
       # Add predictions to samples
       with fo.ProgressBar() as pb:
           for sample in pb(ds):
               # Load image
               image = Image.open(sample.filepath)
               image = func.to_tensor(image).to(device)

               # Perform inference
               detections_faster_RCNN = get_detections(faster_RCNN, image)
               detections_ssd = get_detections(ssd, image)
               detections_retinaNet = get_detections(retinaNet, image)

               # Save predictions to dataset
               sample["faster_rcnn"] = fo.Detections(detections=detections_faster_RCNN)
               sample["ssd"] = fo.Detections(detections=detections_ssd)
               sample["retinanet"] = fo.Detections(detections=detections_retinaNet)
               sample.save()

       print("Finished adding predictions")
```

```
[ ]  predict(coco_ds)
```

```
 100% |███████████████| 1000/1000 [7.5m elapsed, 0s remaining, 2.4 samples/s]
INFO:eta.core.utils: 100% |███████████████| 1000/1000 [7.5m elapsed, 0s remaining, 2.4 samples/s]
Finished adding predictions
```

```
[ ]  predict(voc_ds)
```

```
 100% |███████████████| 1000/1000 [7.0m elapsed, 0s remaining, 2.0 samples/s]
INFO:eta.core.utils: 100% |███████████████| 1000/1000 [7.0m elapsed, 0s remaining, 2.0 samples/s]
Finished adding predictions
```

Filter detections with low confidence

```
[ ]  common_classes=['person', 'car', 'dog', 'sheep', 'bottle', 'cat', 'cow',
                     'horse', 'bicycle', 'boat', 'bus', 'train', 'chair']
```

```
[ ]  filtered_coco_faster_RCNN = coco_ds.filter_labels("faster_rcnn", F("confidence") > 0.50, only_matches=False)
     filtered_coco_ssd = coco_ds.filter_labels("ssd", F("confidence") > 0.30, only_matches=False)
     filtered_coco_retinaNet = coco_ds.filter_labels("retinanet", F("confidence") > 0.40, only_matches=False)

     filtered_voc_faster_RCNN = voc_ds.filter_labels("faster_rcnn", F("confidence") > 0.50, only_matches=False)
     filtered_voc_ssd = voc_ds.filter_labels("ssd", F("confidence") > 0.30, only_matches=False)
     filtered_voc_retinaNet = voc_ds.filter_labels("retinanet", F("confidence") > 0.40, only_matches=False)
```

```
[ ] faster_RCNN_coco_results = filtered_coco_faster_RCNN.evaluate_detections(
        "faster_rcnn", gt_field="ground_truth", eval_key="eval_faster_RCNN", classes=coco_classes, compute_mAP=True)

    ssd_coco_results = filtered_coco_ssd.evaluate_detections(
        "ssd", gt_field="ground_truth", eval_key="eval_ssd", classes=coco_classes, compute_mAP=True)

    retinaNet_coco_results = filtered_coco_retinaNet.evaluate_detections(
        "retinanet", gt_field="ground_truth", eval_key="eval_retinanet", classes=coco_classes, compute_mAP=True)

    faster_RCNN_voc_results = filtered_voc_faster_RCNN.evaluate_detections(
        "faster_rcnn", gt_field="ground_truth", eval_key="eval_faster_RCNN", classes=common_classes, compute_mAP=True)

    ssd_voc_results = filtered_voc_ssd.evaluate_detections(
        "ssd", gt_field="ground_truth", eval_key="eval_ssd", classes=common_classes, compute_mAP=True)

    retinaNet_voc_results = filtered_voc_retinaNet.evaluate_detections(
        "retinanet", gt_field="ground_truth", eval_key="eval_retinanet", classes=common_classes, compute_mAP=True)
```

```
Evaluating detections...
INFO:fiftyone.utils.eval.detection:Evaluating detections...
 100% |██████████████| 1000/1000 [56.4s elapsed, 0s remaining, 19.4 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [56.4s elapsed, 0s remaining, 19.4 samples/s]
Performing IoU sweep...
INFO:fiftyone.utils.eval.coco:Performing IoU sweep...
 100% |██████████████| 1000/1000 [25.2s elapsed, 0s remaining, 39.9 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [25.2s elapsed, 0s remaining, 39.9 samples/s]
Evaluating detections...
INFO:fiftyone.utils.eval.detection:Evaluating detections...
 100% |██████████████| 1000/1000 [33.1s elapsed, 0s remaining, 25.1 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [33.1s elapsed, 0s remaining, 25.1 samples/s]
Performing IoU sweep...
INFO:fiftyone.utils.eval.coco:Performing IoU sweep...
 100% |██████████████| 1000/1000 [16.3s elapsed, 0s remaining, 58.3 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [16.3s elapsed, 0s remaining, 58.3 samples/s]
Evaluating detections...
INFO:fiftyone.utils.eval.detection:Evaluating detections...
 100% |██████████████| 1000/1000 [42.4s elapsed, 0s remaining, 23.3 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [42.4s elapsed, 0s remaining, 23.3 samples/s]
Performing IoU sweep...
INFO:fiftyone.utils.eval.coco:Performing IoU sweep...
 100% |██████████████| 1000/1000 [22.1s elapsed, 0s remaining, 45.5 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [22.1s elapsed, 0s remaining, 45.5 samples/s]
Evaluating detections...
INFO:fiftyone.utils.eval.detection:Evaluating detections...
 100% |██████████████| 1000/1000 [28.2s elapsed, 0s remaining, 43.0 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [28.2s elapsed, 0s remaining, 43.0 samples/s]
Performing IoU sweep...
INFO:fiftyone.utils.eval.coco:Performing IoU sweep...
 100% |██████████████| 1000/1000 [12.1s elapsed, 0s remaining, 89.9 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [12.1s elapsed, 0s remaining, 89.9 samples/s]

Evaluating detections...
INFO:fiftyone.utils.eval.detection:Evaluating detections...
 100% |██████████████| 1000/1000 [14.4s elapsed, 0s remaining, 71.5 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [14.4s elapsed, 0s remaining, 71.5 samples/s]
Performing IoU sweep...
INFO:fiftyone.utils.eval.coco:Performing IoU sweep...
 100% |██████████████| 1000/1000 [8.8s elapsed, 0s remaining, 119.2 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [8.8s elapsed, 0s remaining, 119.2 samples/s]
Evaluating detections...
INFO:fiftyone.utils.eval.detection:Evaluating detections...
 100% |██████████████| 1000/1000 [20.4s elapsed, 0s remaining, 59.8 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [20.4s elapsed, 0s remaining, 59.8 samples/s]
Performing IoU sweep...
INFO:fiftyone.utils.eval.coco:Performing IoU sweep...
 100% |██████████████| 1000/1000 [10.6s elapsed, 0s remaining, 103.1 samples/s]
INFO:eta.core.utils: 100% |██████████████| 1000/1000 [10.6s elapsed, 0s remaining, 103.1 samples/s]
```

Faster R-CNN on COCO 2017 dataset

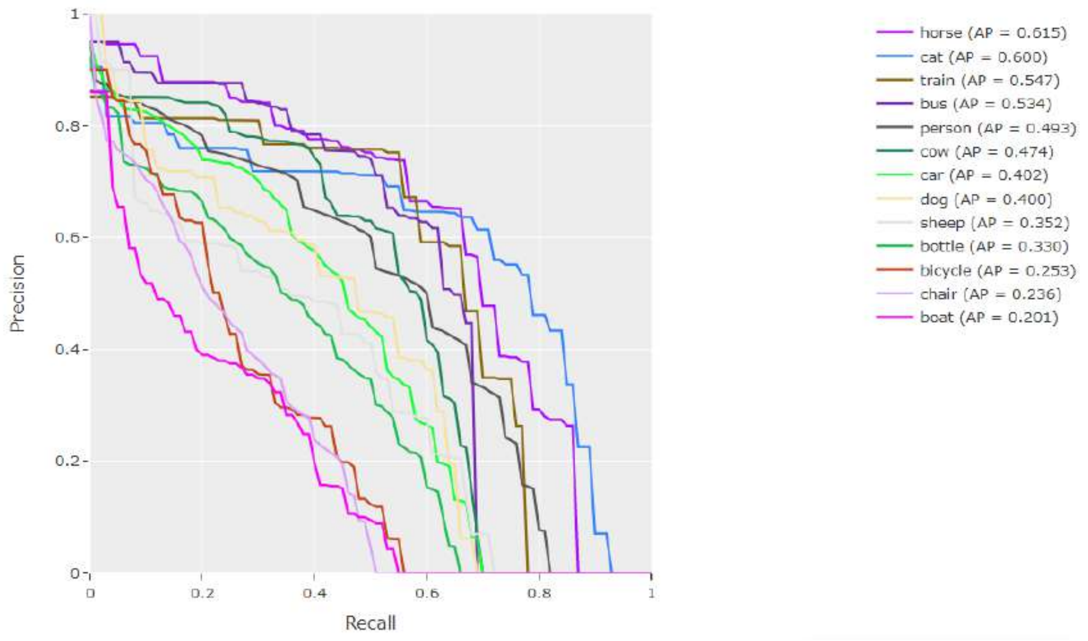
```
[ ] # Print a classification report -- Faster R-CNN -- COCO dataset
    faster_RCNN_coco_results.print_report(classes=common_classes)
    print("mAP = ", faster_RCNN_coco_results.mAP())
    plot = faster_RCNN_coco_results.plot_pr_curves(classes=common_classes)
    plot.show()
```

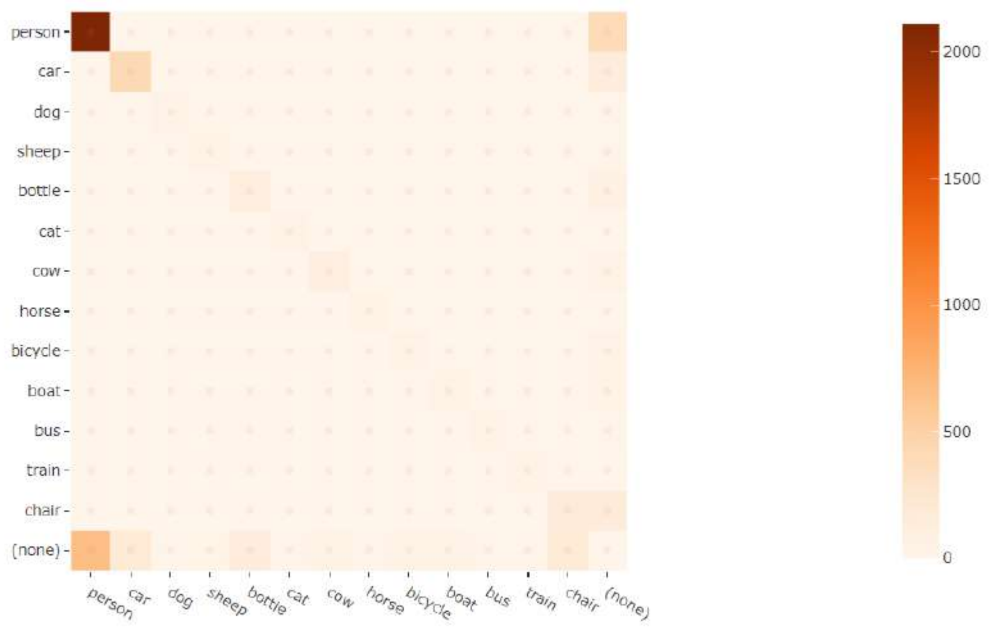
```
              precision    recall  f1-score   support

      person       0.76      0.84      0.80      2499
         car       0.67      0.74      0.71       545
         dog       0.61      0.68      0.65        44
       sheep       0.63      0.72      0.67        57
      bottle       0.47      0.65      0.54       187
         cat       0.70      0.92      0.80        38
         cow       0.76      0.74      0.75       134
       horse       0.78      0.86      0.82        50
     bicycle       0.54      0.55      0.55        65
        boat       0.48      0.60      0.54        70
         bus       0.67      0.69      0.68        51
       train       0.62      0.78      0.69        36
       chair       0.46      0.51      0.48       305

   micro avg       0.69      0.78      0.73      4081
   macro avg       0.63      0.71      0.67      4081
weighted avg       0.70      0.78      0.74      4081

mAP =  0.35487406849841835
```





SSD on COCO 2017 dataset

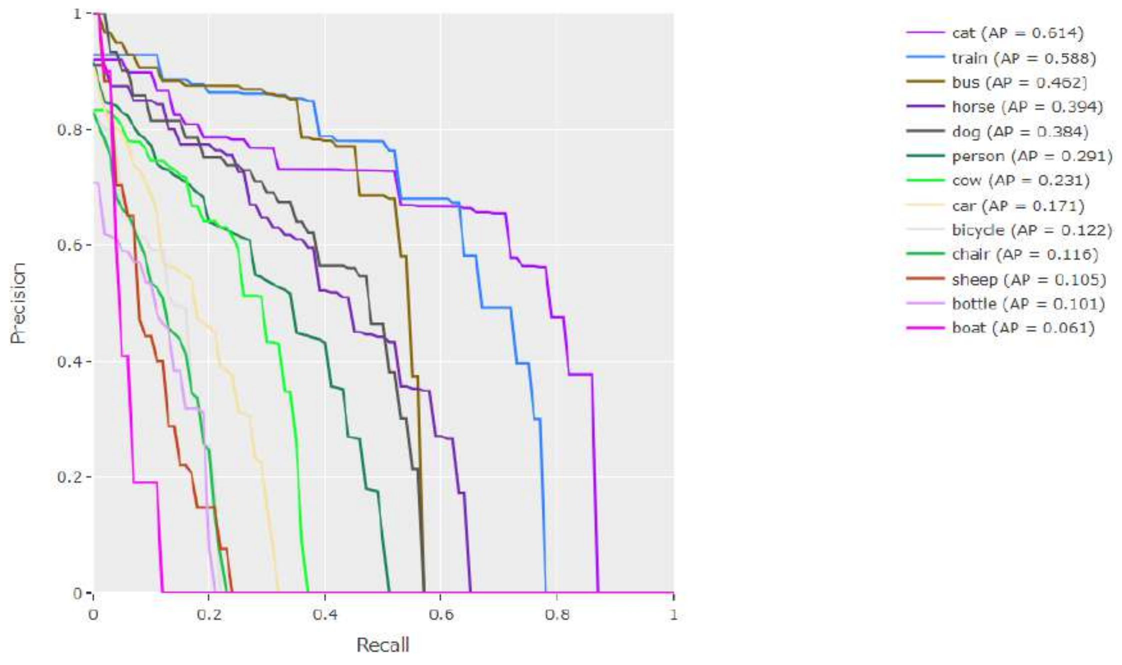
```
[ ]  # Print a classification report -- SSD -- COCO dataset
     ssd_coco_results.print_report(classes=common_classes)
     print("mAP = ", ssd_coco_results.mAP())
     plot = ssd_coco_results.plot_pr_curves(classes=common_classes)
     plot.show()
```

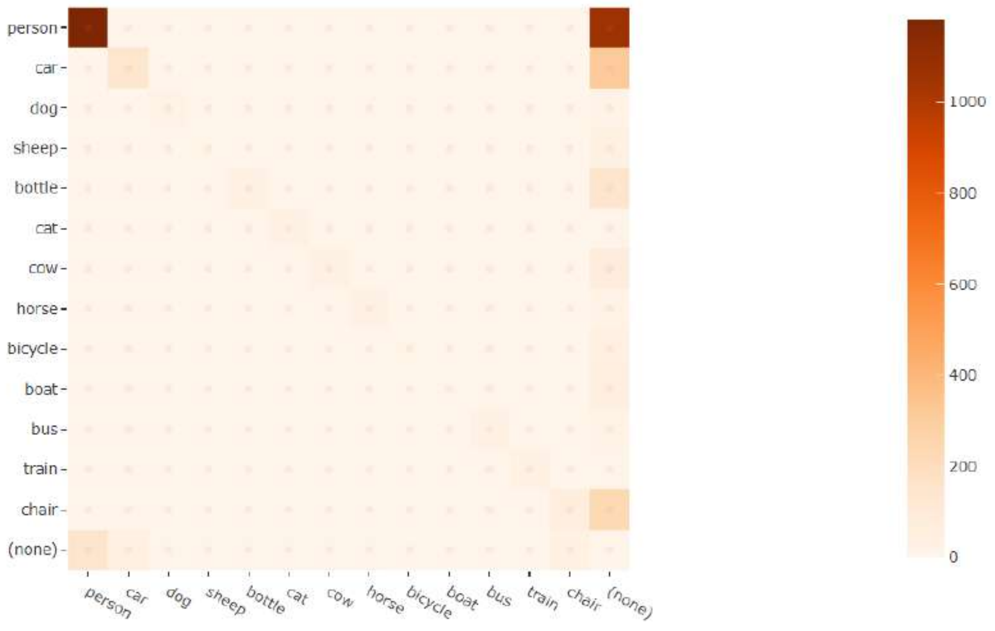
```
               precision    recall  f1-score   support

      person       0.89      0.53      0.66      2247
         car       0.76      0.31      0.44       457
         dog       0.69      0.57      0.62        44
       sheep       0.72      0.23      0.35        57
      bottle       0.76      0.20      0.32       187
         cat       0.87      0.87      0.87        38
         cow       0.84      0.35      0.49       120
       horse       0.82      0.64      0.72        50
     bicycle       0.82      0.22      0.34        65
        boat       0.64      0.11      0.19        62
         bus       0.94      0.57      0.71        51
       train       0.93      0.78      0.85        36
       chair       0.64      0.23      0.33       304

   micro avg       0.85      0.44      0.58      3718
   macro avg       0.79      0.43      0.53      3718
weighted avg       0.83      0.44      0.57      3718

mAP =  0.21310037432771503
```





RetinaNet on COCO 2017 dataset

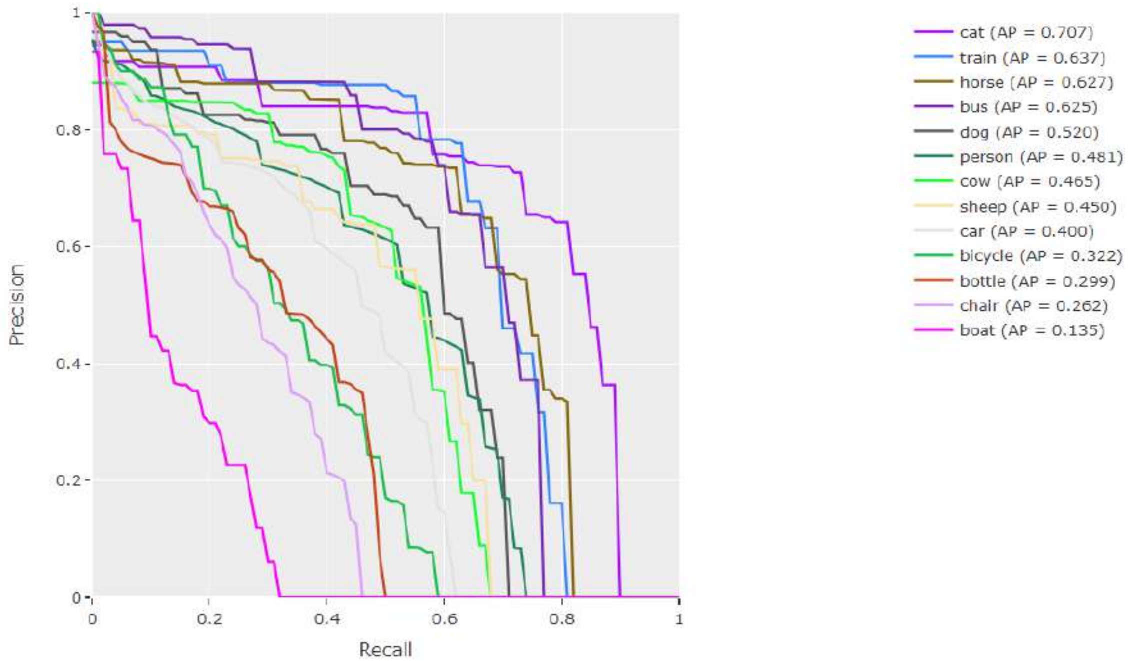
```
[ ]  # Print a classification report -- RetinaNet -- COCO dataset
     retinaNet_coco_results.print_report(classes=common_classes)
     print("mAP = ", retinaNet_coco_results.mAP())
     plot = retinaNet_coco_results.plot_pr_curves(classes=common_classes)
     plot.show()
```

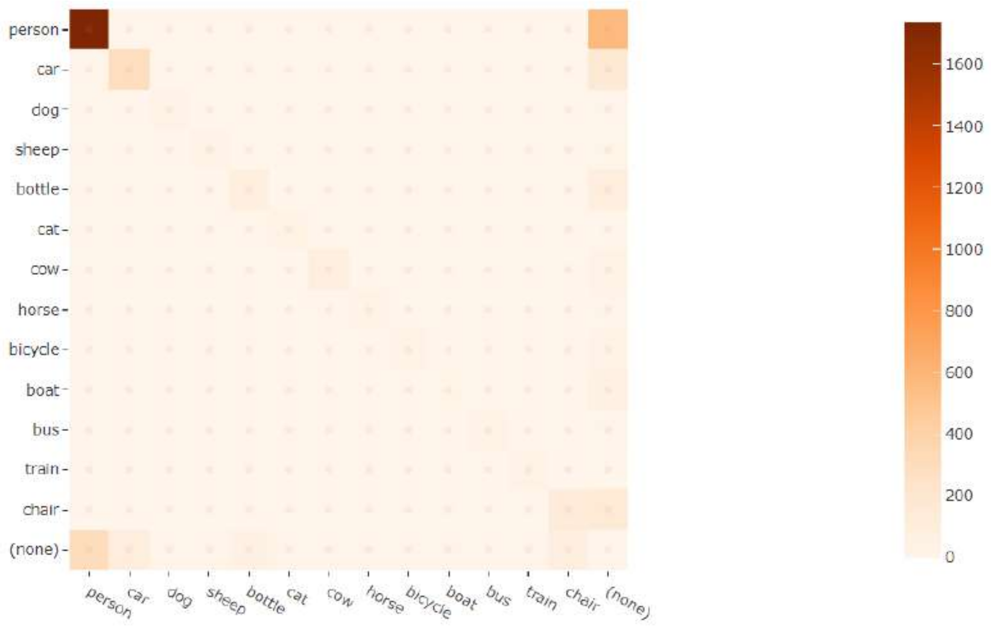
```
              precision    recall  f1-score   support

      person       0.85      0.75      0.80      2314
         car       0.74      0.63      0.68       471
         dog       0.70      0.70      0.70        44
       sheep       0.81      0.67      0.73        57
      bottle       0.69      0.49      0.57       187
         cat       0.72      0.89      0.80        38
         cow       0.90      0.70      0.79       128
       horse       0.84      0.82      0.83        50
     bicycle       0.78      0.58      0.67        65
        boat       0.61      0.31      0.41        62
         bus       0.89      0.76      0.82        51
       train       0.74      0.81      0.77        36
       chair       0.62      0.45      0.52       304

   micro avg       0.81      0.69      0.74      3807
   macro avg       0.76      0.66      0.70      3807
weighted avg       0.80      0.69      0.74      3807

mAP =  0.3735687967508535
```





Faster R-CNN on PASCAL VOC 2012 dataset

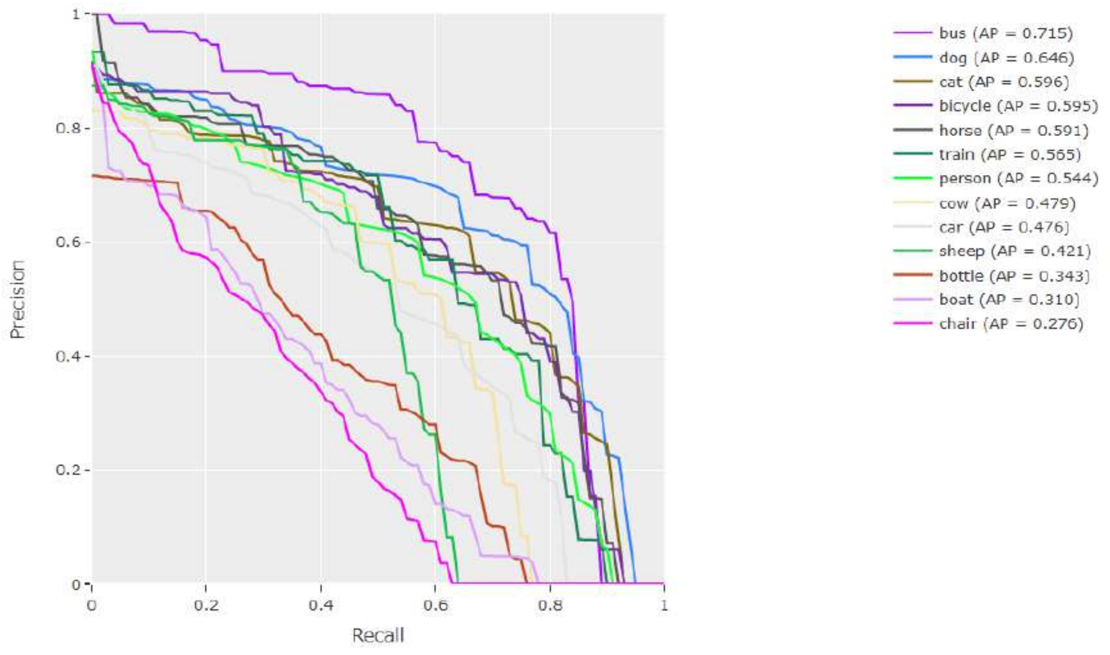
```
[ ] # Print a classification report -- Faster R-CNN -- PASCAL VOC 2012 dataset
    faster_RCNN_voc_results.print_report(classes=common_classes)
    print("mAP = ", faster_RCNN_voc_results.mAP())
    plot = faster_RCNN_voc_results.plot_pr_curves(classes=common_classes)
    plot.show()
```

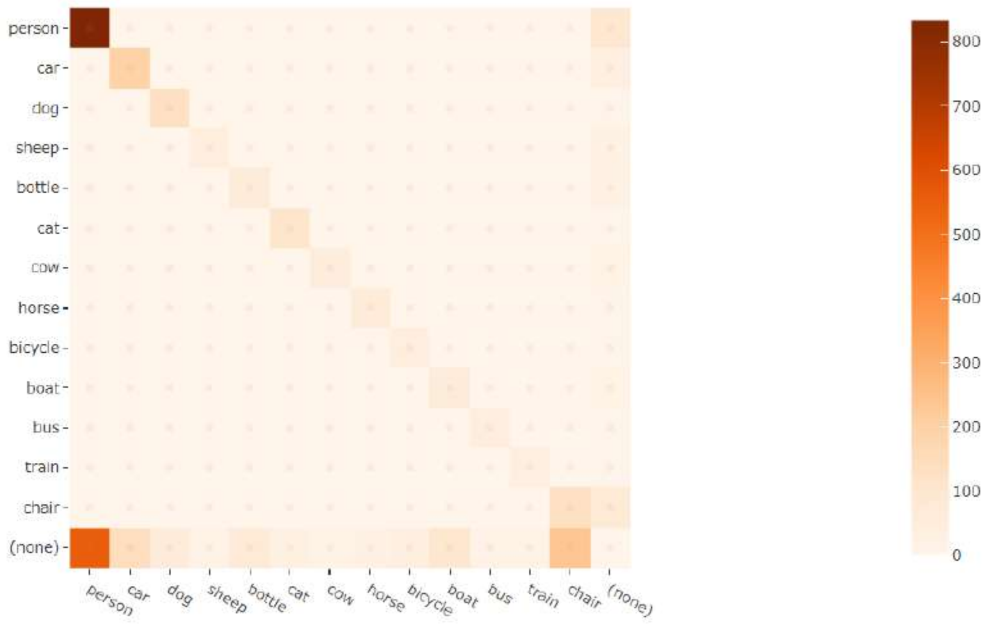
```
              precision    recall  f1-score   support

      person       0.60      0.90      0.72       921
         car       0.58      0.83      0.68       237
         dog       0.68      0.95      0.79       139
       sheep       0.71      0.63      0.67        73
      bottle       0.44      0.75      0.55        80
         cat       0.77      0.93      0.84       114
         cow       0.79      0.76      0.78        71
       horse       0.72      0.91      0.80        69
     bicycle       0.56      0.93      0.70        54
        boat       0.38      0.78      0.51        77
         bus       0.72      0.89      0.80        53
       train       0.75      0.89      0.81        46
       chair       0.35      0.63      0.45       209

   micro avg       0.58      0.85      0.69      2143
   macro avg       0.62      0.83      0.70      2143
weighted avg       0.59      0.85      0.69      2143

mAP =  0.5044338340682839
```





SSD on PASCAL VOC 2012 dataset

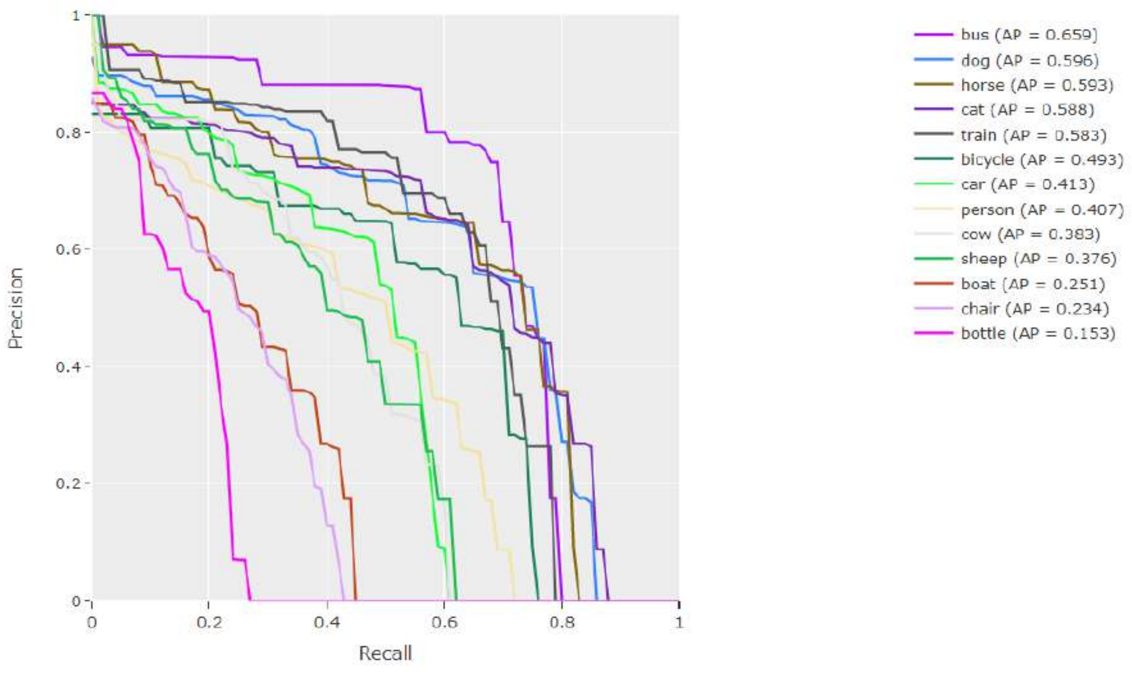
```
[ ] # Print a classification report -- SSD -- PASCAL VOC 2012 dataset
    ssd_voc_results.print_report(classes=common_classes)
    print("mAP = ", ssd_voc_results.mAP())
    plot = ssd_voc_results.plot_pr_curves(classes=common_classes)
    plot.show()
```

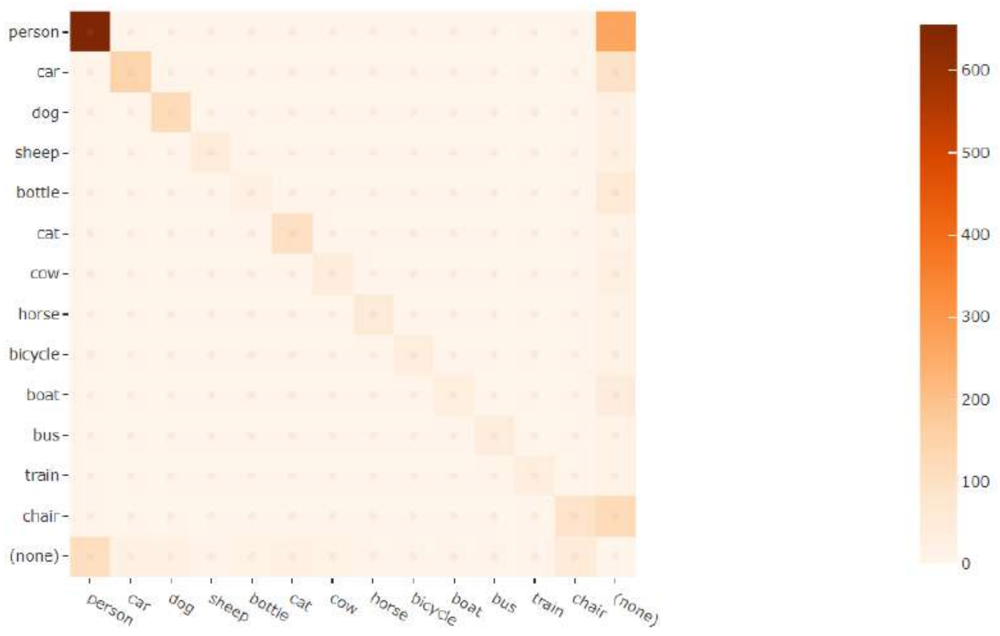
```
              precision    recall  f1-score   support

      person       0.86      0.71      0.78       921
         car       0.89      0.61      0.72       237
         dog       0.84      0.86      0.85       139
       sheep       0.87      0.62      0.72        73
      bottle       0.68      0.26      0.38        80
         cat       0.85      0.88      0.87       114
         cow       0.75      0.61      0.67        71
       horse       0.90      0.83      0.86        69
     bicycle       0.87      0.76      0.81        54
        boat       0.83      0.44      0.58        77
         bus       0.88      0.79      0.83        53
       train       0.88      0.78      0.83        46
       chair       0.63      0.43      0.51       209

   micro avg       0.84      0.67      0.74      2143
   macro avg       0.83      0.66      0.72      2143
weighted avg       0.83      0.67      0.73      2143

mAP =  0.4406238848034472
```





RetinaNet on PASCAL VOC 2012 dataset

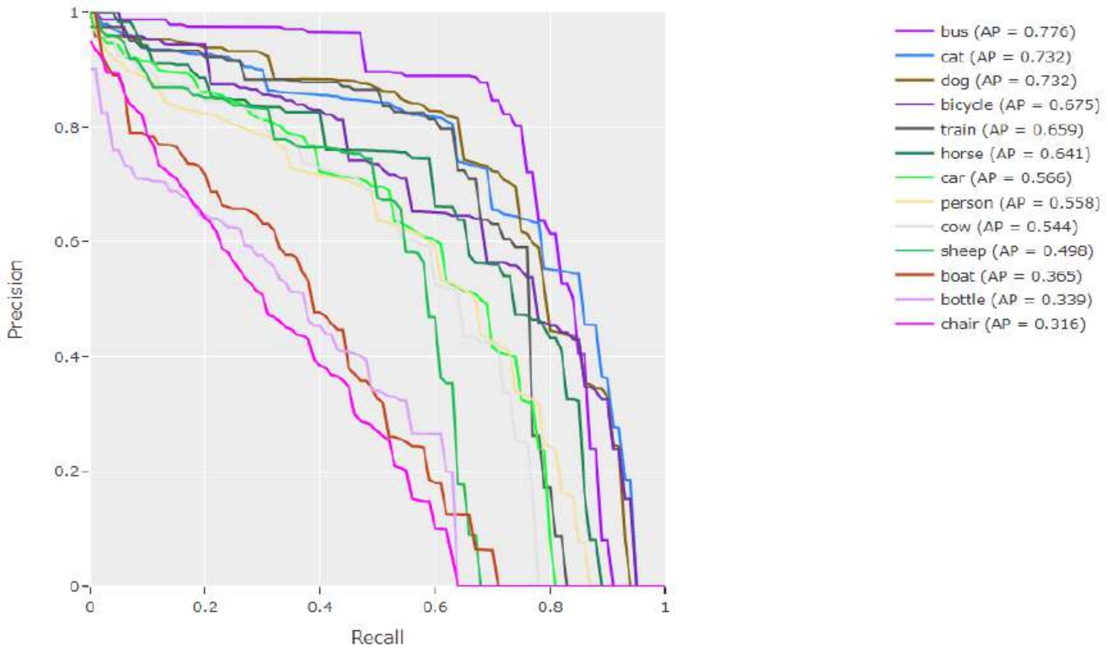
```
[ ]  # Print a classification report -- RetinaNet -- PASCAL VOC 2012 dataset
     retinaNet_voc_results.print_report(classes=common_classes)
     print("mAP = ", retinaNet_voc_results.mAP())
     plot = retinaNet_voc_results.plot_pr_curves(classes=common_classes)
     plot.show()
```

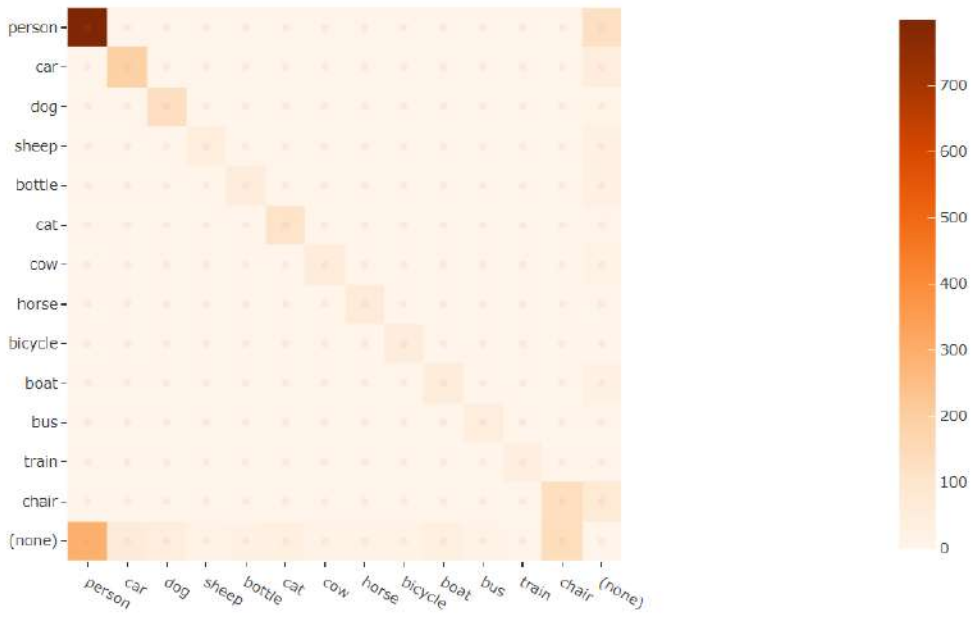
```
               precision    recall  f1-score   support

      person        0.73      0.87      0.79       921
         car        0.76      0.81      0.78       237
         dog        0.73      0.94      0.82       139
       sheep        0.83      0.67      0.74        73
      bottle        0.64      0.64      0.64        80
         cat        0.77      0.95      0.85       114
         cow        0.79      0.77      0.78        71
       horse        0.80      0.88      0.84        69
     bicycle        0.76      0.94      0.84        54
        boat        0.60      0.70      0.65        77
         bus        0.80      0.91      0.85        53
       train        0.86      0.83      0.84        46
       chair        0.49      0.64      0.55       209

   micro avg        0.71      0.83      0.77      2143
   macro avg        0.74      0.81      0.77      2143
weighted avg        0.72      0.83      0.77      2143

mAP =  0.5691832933285179
```





Results

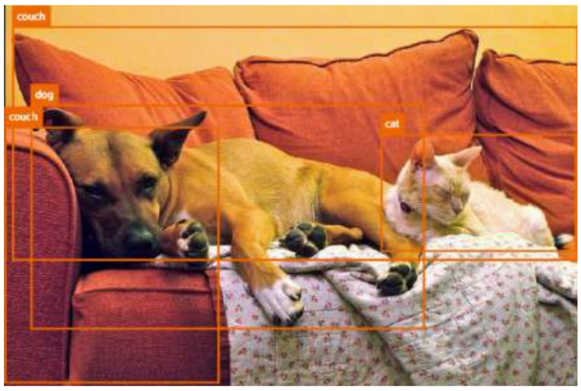
Comparing the three selected models on the two selected datasets:

Used Models		Faster R-CNN	SSD	RetinaNet
# of stages		2	1	1
Base Network		ResNet-50	VGG-16	ResNet-50
Speed		slow	fastest	faster
mAP	COCO	0.35	0.21	0.37
	PASCAL VOC	0.50	0.44	0.56
COCO	Good at classes	horse, cat	cat, train	cat, train
	Bad at classes	chair, boat	bottle, boat	chair, boat
PASCAL VOC	Good at classes	bus, dog	bus, dog	bus, cat
	Bad at classes	boat, chair	chair, bottle	bottle, chair

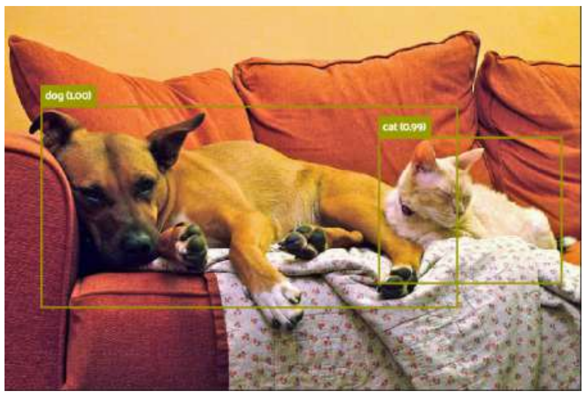
Samples of good and bad results:

COCO 2017:

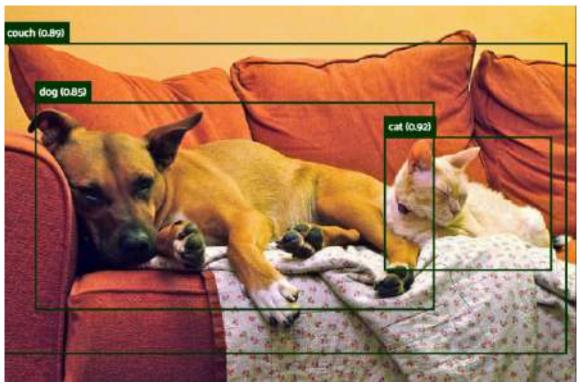
- Good
  - Ground Truth



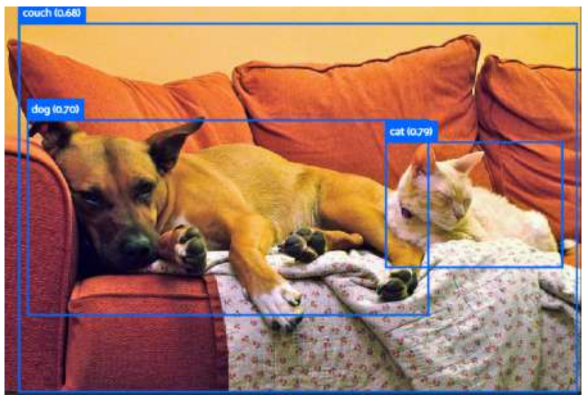
  - Faster_RCNN



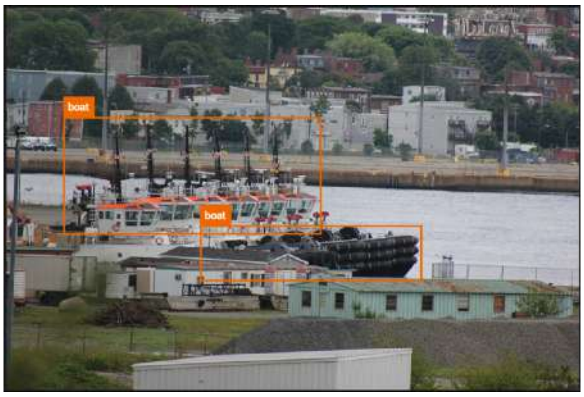
  - SSD



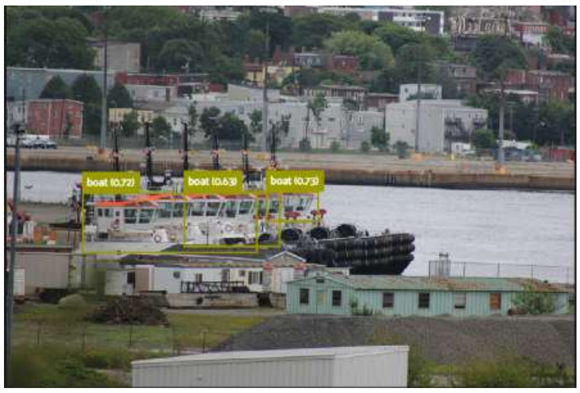
  - RetinaNet



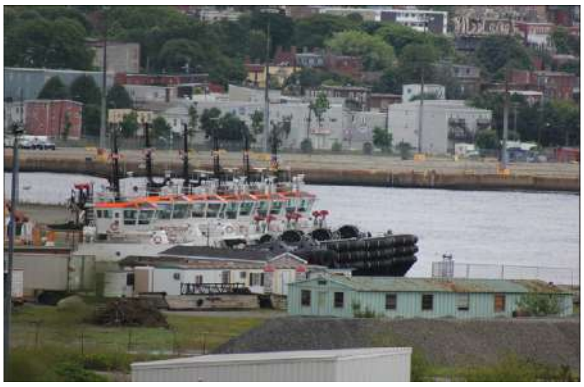
- Bad
    - Ground Truth



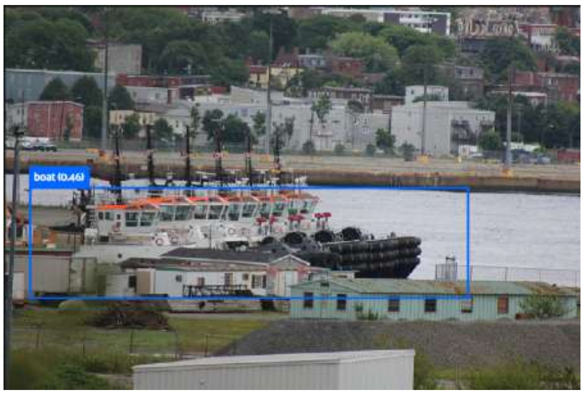
    - Faster_RCNN



    - SSD

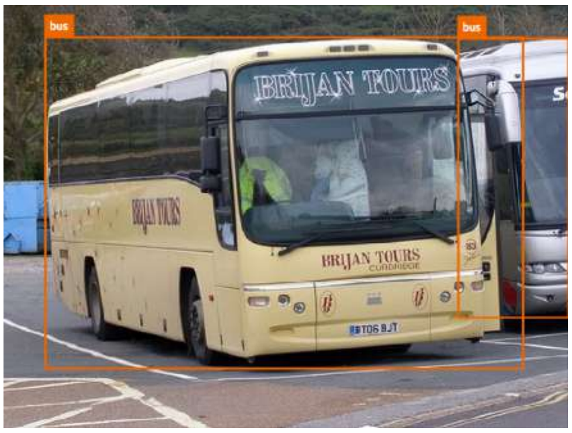


    - RetinaNet

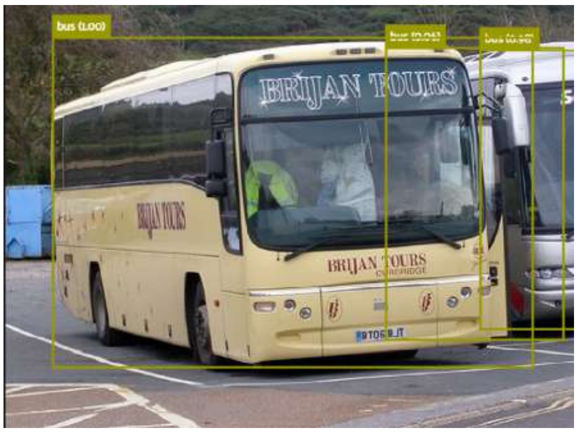


PASCAL VOC 2012:

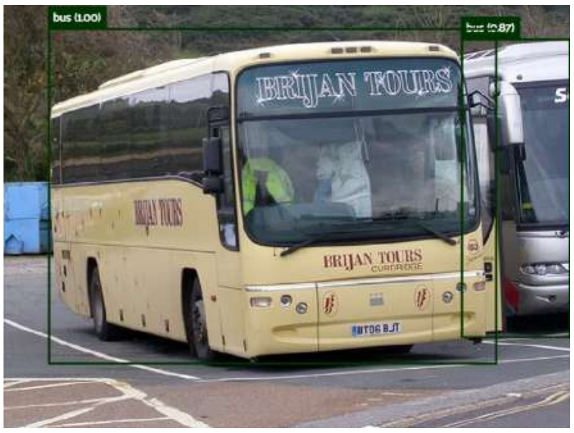
- Good
    - Ground Truth



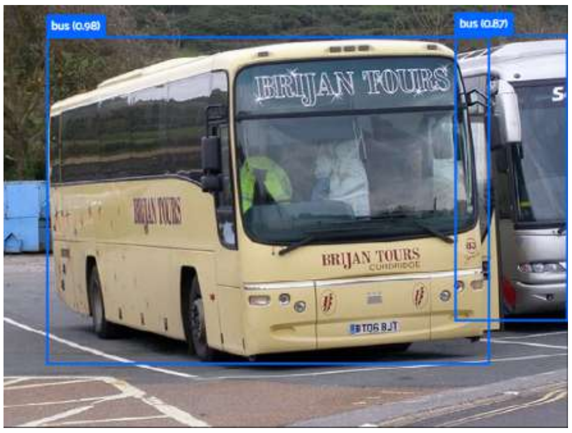
    - Faster_RCNN



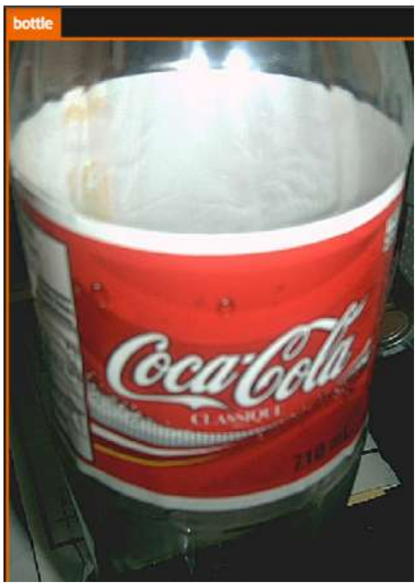
    - SSD



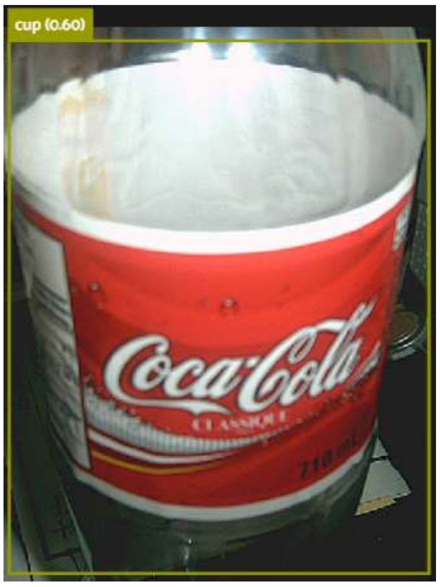
    - RetinaNet



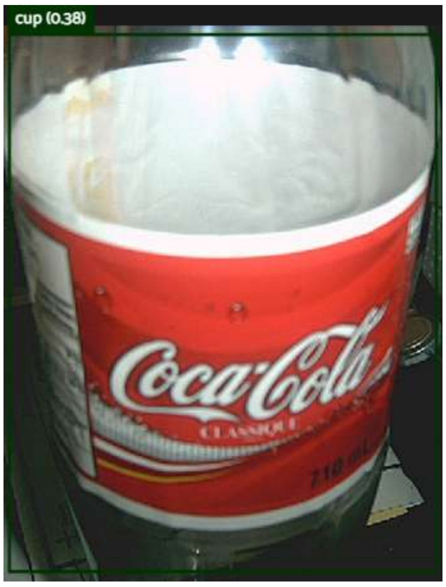
- Bad
  - Ground Truth



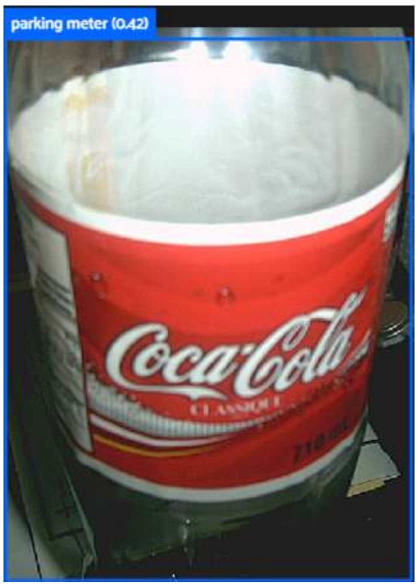
  - Faster_RCNN



  - SSD



- RetinaNet



References:

[1] Ren, Shaoqing, et al. "**Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**." *Advances in neural information processing systems* 28 (2015).
[2] Liu, Wei, et al. "**SSD: Single Shot Multibox Detector.**" *European conference on computer vision*. Springer, Cham, 2016.
[3] Lin, Tsung-Yi, et al. "**Focal Loss for Dense Object Detection.**" *Proceedings of the IEEE international conference on computer vision*. 2017.