Assigment 4 CV: Object Detection

Mayar Ayman: 19016744 Salma Saeed: 19015779 Sally Kamel: 19015762

December 30, 2023

1 Datasets used

1.1 COCO

COCO (Common Objects in Context) dataset is a widely used benchmark dataset in computer vision. It consists of images depicting everyday scenes with rich annotations for tasks such as object detection, segmentation, and captioning. The dataset includes bounding box annotations for object instances, pixel-wise segmentation masks for certain categories, and captions for images. With 80 diverse object categories and a large number of images, COCO serves as a valuable resource for training and evaluating computer vision models. It has been a key dataset for challenges and competitions in object detection and image captioning, driving advancements in these areas.

1.2 PASCAL VOC

PASCAL VOC dataset is not as large as COCO, the dataset consists of images covering 20 object categories, ranging from people and animals to vehicles and everyday objects. Each image is annotated with ground truth information, including bounding box annotations for object detection and, in some versions, pixel-wise segmentation masks for more detailed analysis.

2 Code Explain

- "handle-predictions" function: - It checks if CUDA (GPU) is available and sets the device accordingly (GPU if available, CPU otherwise). - The provided model is moved to the specified device (GPU or CPU). - The model is set to evaluation mode which disables specific layers like dropout and batch normalization that are used during training. - Using a progress bar, the function iterates through each sample in the dataset. - For each sample, it loads an image and converts it to a PyTorch tensor while moving it to the specified device. - A PyTorch tensor is a fundamental data structure used in PyTorch, a popular open-source machine learning library primarily utilized for deep learning tasks. It is a multi-dimensional array similar to NumPy arrays but specifically designed for GPU-accelerated computations. - The input image is passed through the model to obtain predictions (labels, scores, and bounding boxes). - The predictions are converted to FiftyOne format, which includes normalized bounding boxes. - These formatted predictions are saved to the dataset. - Finally, a message is printed to indicate the completion of the predictions for the given model.

- "high-conf-dataset" function: - This function Filters the dataset based on the confidence scores. It keeps only samples where the confidence scores are greater than 0.5 and samples with no matches after filtering are retained. - Filters the dataset based on the predicted labels ("label"). It keeps only samples where the predicted labels are among the specified classes provided in the classes list and samples with no matches after filtering are retained. - returns a subset of the original dataset that satisfies the filtering conditions

- "evaluate" function: - It performs evaluations on the provided dataset: - f"dataset-name-predictions-of-model-name": Specifies the field in the dataset containing the predictions of the specified model for evaluation. - gt-field="ground-truth": Specifies the ground truth field against which the predictions are evaluated. - eval-key="eval": Assigns a key to the evaluation results. -

classwise=False: Indicates that the evaluation is performed for the dataset as a whole rather than separately for each class. - compute-mAP=True: Specifies to compute the mean Average Precision (mAP) metric during evaluation. A higher value is generally considered better. mAP measures the average precision across different classes. - Returns the evaluation results obtained after assessing the high-confidence predictions against the ground truth annotations in the dataset.

- "print-classes-classification-report-and-curves" function: - Generates a classification report showing metrics such as precision, recall, F1-score, and support for each class specified in the classes list. It's based on the evaluation results. - Plots Precision-Recall (PR) curves for each class specified in the classes list. The PR curve illustrates the trade-off between precision and recall at various threshold levels

- "print-classes-confusionMatrix" function: - Creates a confusion matrix visualization for the specified classes. A confusion matrix displays the true positive, false positive, true negative, and false negative counts for each class, helping to assess the model's performance

- "model-processing" function: - This function handles predictions for a given model on the provided dataset using "handle-predictions". - Generates a dataset containing only high-confidence predictions. It filters the original dataset to retain only samples where the model's predictions exceed a confidence threshold and belong to specified non-numeric classes. - Evaluates the high-confidence dataset using the specified model. - Prints a classification report and precision-recall curves. - Prints the Mean Average Precision (mAP) value obtained from the evaluation results. - Finally, the function returns the high-confidence dataset and the evaluation results

3 Architectute Models we used

	SSD-VGG	Retina-ResNet	Faster R-CNN
Number of stages	Single	Single	Multi(2)
Number of layers	16	50	28

Table 1: architecture comparison

3.1 RerinaNet ResNet

3.1.1 Overview

- RetinaNet is a one-stage object detection model.
- It utilizes a Feature Pyramid networks function to detect small objects in the image.
- 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. The two subnetworks feature a simple design that the authors propose specifically for one-stage, dense detection.

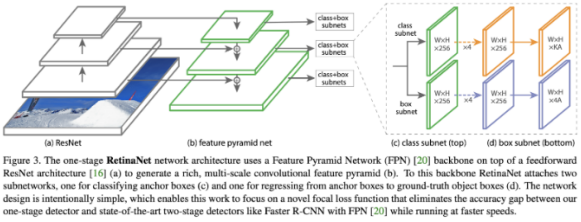


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying 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.

Figure 1: RetinaNet

- Backbone: Feature Pyramid network built on top of ResNet50 or ResNet101.
- Classification subnet: It predicts the probability of object presence at each spatial position for each of the A anchors and K object classes. Takes a input feature map with C channels from a pyramid level, the subnet applies four 3x3 conv layers, each with C filters amd each followed by ReLU activations. Finally sigmoid activations are attached to the outputs. Focal loss is applied as the loss function.
- Box Regression Subnet: Similar to classification net used but the parameters are not shared. Outputs the object location with respect to anchor box if an object exists. smooth-L1loss with sigma equal to 3 is applied as the loss function to this part of the sub-network.

3.1.2 Focal Loss Function

- It utilizes a focal loss function to address class imbalance during training.
- RetinaNet uses a focal loss function, a dynamically scaled cross entropy loss, where the scaling factor decays to zero as confidence in the correct class increases. Intuitively, this scaling factor can automatically down-weight the contribution of easy examples during training and rapidly focus the model on hard examples.

3.1.3 RetinaNet Detector

- ResNet is used for deep feature extraction.
- Feature Pyramid Network (FPN) is used on top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution input image. (Originally, FPN is a two-stage detector which has state-of-the-art results.
- FPN is multiscale, semantically strong at all scales, and fast to compute.
- There are some modest changes for the FPN here. A pyramid is generated from P3 to P7. Some major changes are: P2 is not used now due to computational reasons. (ii) P6 is computed by strided convolution instead of downsampling. (iii) P7 is included additionally to improve the accuracy of large object detection.

3.1.4 Weaknesses and Strengths

3.1.5 Output

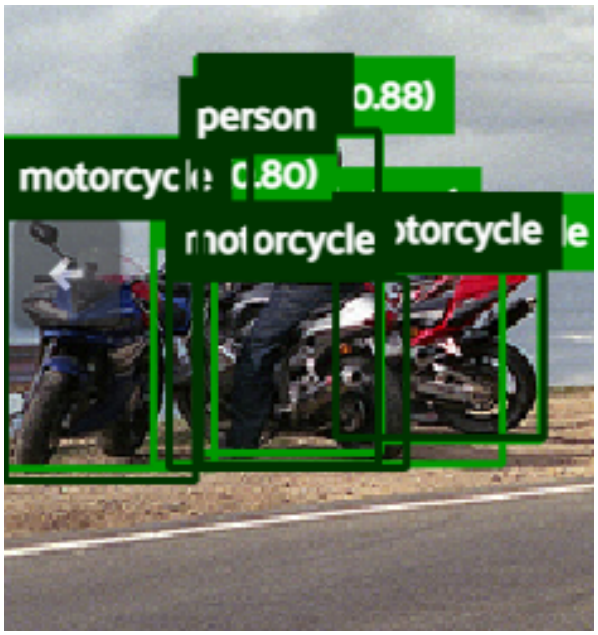


Figure 2: RetinaNet: success coco

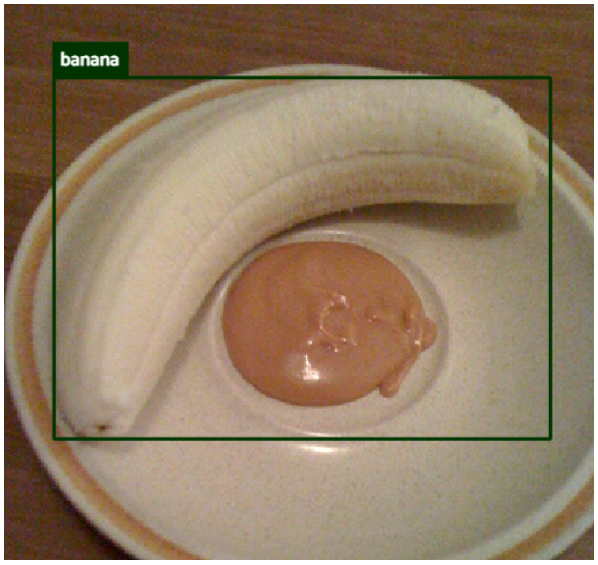


Figure 3: RetinaNet: failure coco

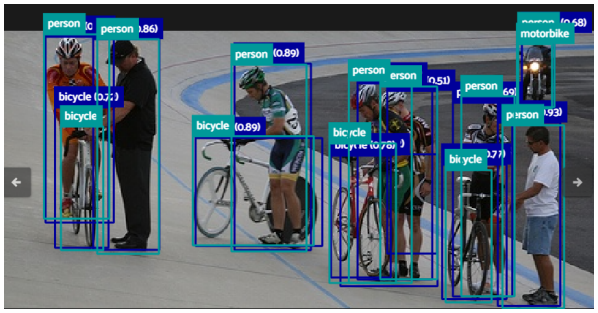


Figure 4: VOC: success voc

Figure 5: VOC: failure voc

3.2 SSD 300 VGG16

The Single-Shot Detector (SSD) is an object detection model that performs detection in a single forward pass of the network. Unlike earlier methods like R-CNN, SSD does not require a separate region proposal step or sliding window techniques to generate detection regions. Instead, it combines the tasks of object localization and classification within the same network architecture. The SSD network typically comprises two main components: Backbone and SSD Head.

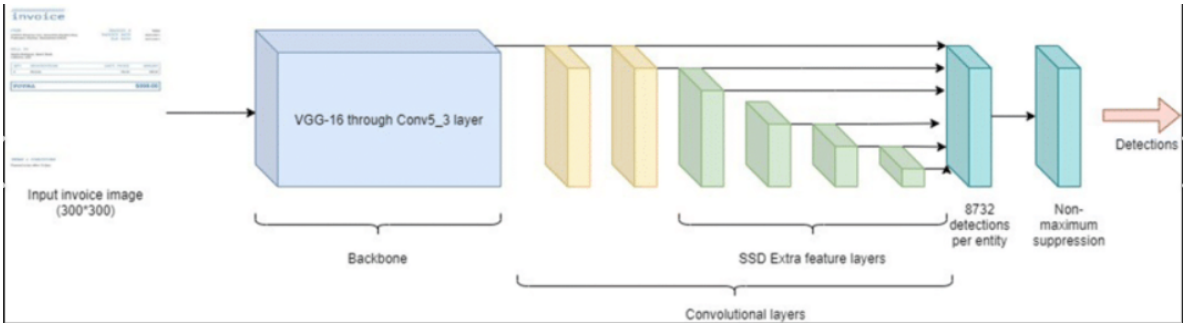


Figure 6: SSD Model Architecture

3.2.1 Backbone:

This component is a pre-trained Convolutional Neural Network (CNN). It extracts meaningful features from input images. The choices for the backbone network here is VGG16.

- Input Layer: The input to the VGG16 backbone is typically an RGB image of size 300*300 pixels.
- Convolutional Layers: VGG16 consists of 13 convolutional layers stacked one after another, each followed by a rectified linear unit (ReLU) activation function. These convolutional layers use small receptive fields (3x3), and they're designed to learn hierarchical features from the input image.
- We can notice that SSD uses the VGG16 architecture until the $\text{conv}5_3 layer and it also uses the output of the conv4_3 layer$
$After every two convolutional layers, VGG16 incorporates max-pooling layers with a 2x2 window and a stride of 2. Max$

- Adaptation for Object Detection: In SSD 300 VGG16, the original VGG16 architecture is typically modified at the end. The fully connected layers present in the original VGG16 architecture are often removed or replaced with convolutional layers. This modification allows the network to generate multi-scale feature maps instead of fully connected layers.

3.2.2 SSD Head:

This part of the network operates on the feature maps produced by the backbone. It is responsible for predicting bounding box positions and class labels for objects in the image. This component follows the feature extraction phase provided by the backbone network (VGG16) and operates on the generated feature maps to detect and classify objects within an image.

- Feature Maps: The SSD head receives feature maps extracted from the backbone network. These feature maps contain hierarchical representations of the input image at multiple scales and levels of abstraction.
- Multiscale Predictions: The SSD head applies a series of convolutional filters, typically referred to as auxiliary convolutional layers, to the feature maps. These layers have varying receptive fields, enabling predictions to be made at multiple spatial resolutions or feature map scales.
- Default Boxes (Anchor Boxes): SSD generates default bounding boxes, also known as anchor boxes, at different aspect ratios and positions across the feature maps. These default boxes serve as reference boxes used for predicting bounding box coordinates and object classes.
- Predictions: The SSD head simultaneously predicts two main things for each default box:
  - Object Class: It estimates the probability of the presence of each class within the default box. This is done using softmax activation for multiclass classification.
  - Bounding Box Offsets: It predicts the offsets required to adjust the dimensions and position of the default boxes to more accurately encapsulate the ground truth object bounding boxes.

- Non-Maximum Suppression (NMS): After obtaining these predictions, a post-processing step called non-maximum suppression is applied to filter out redundant or overlapping detections, ensuring that only the most confident and non-overlapping predictions are retained.

3.2.3 Outputs using COCO dataset:

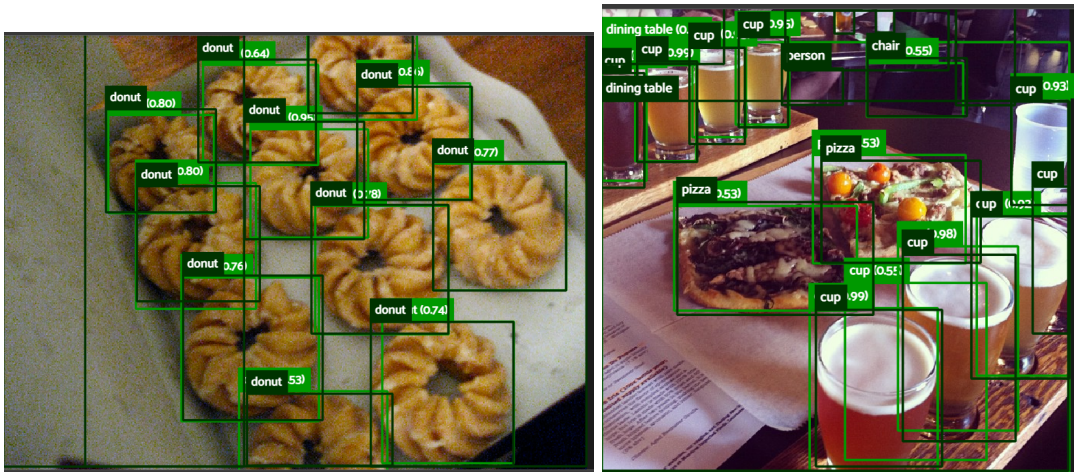


Figure 7: SSD:success coco

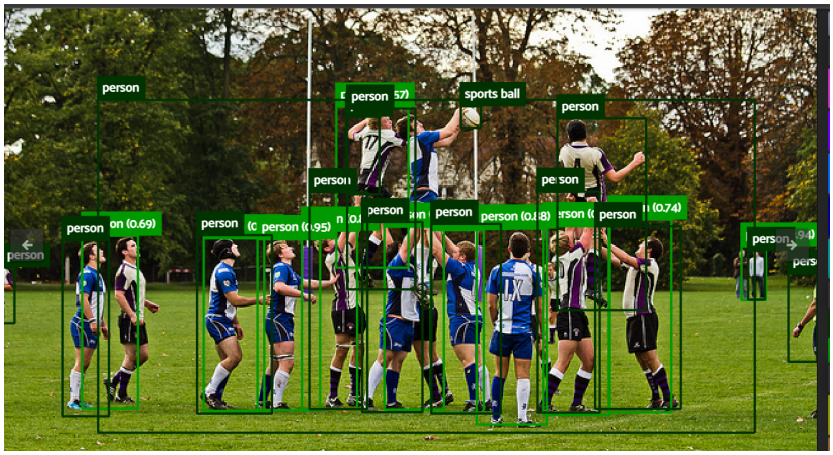


Figure 8: SSD:success coco

Figure 9: SSD:failure coco

Figure 10: SSD:failure coco

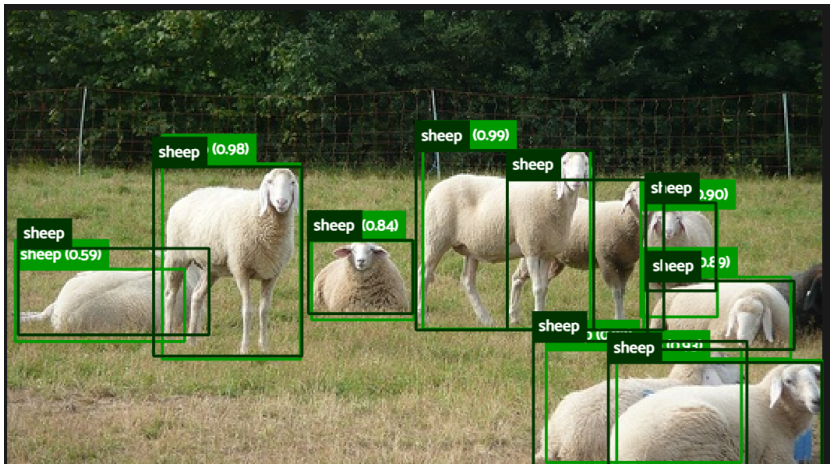


Figure 11: SSD:success voc

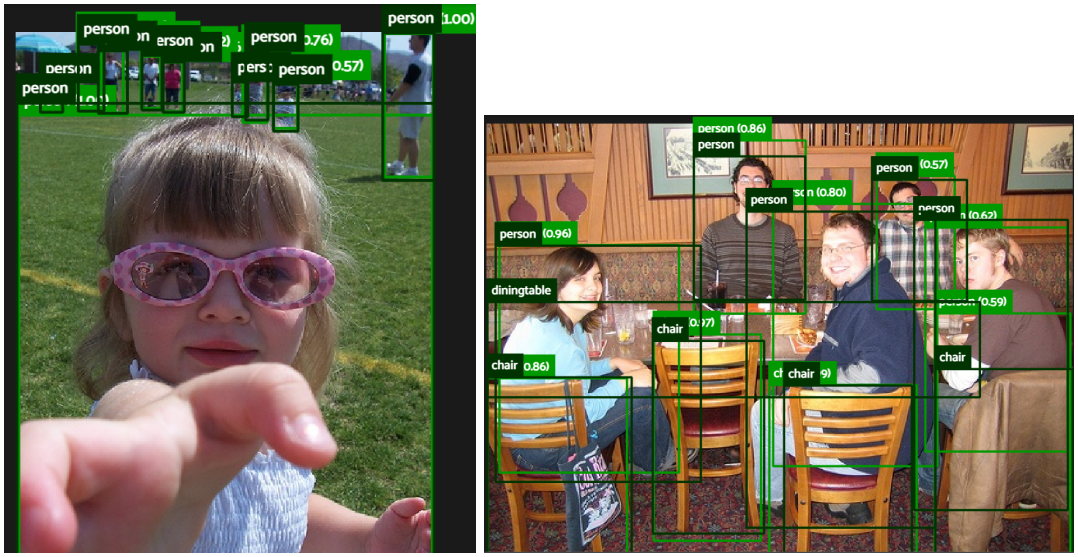


Figure 12: SSD:success voc

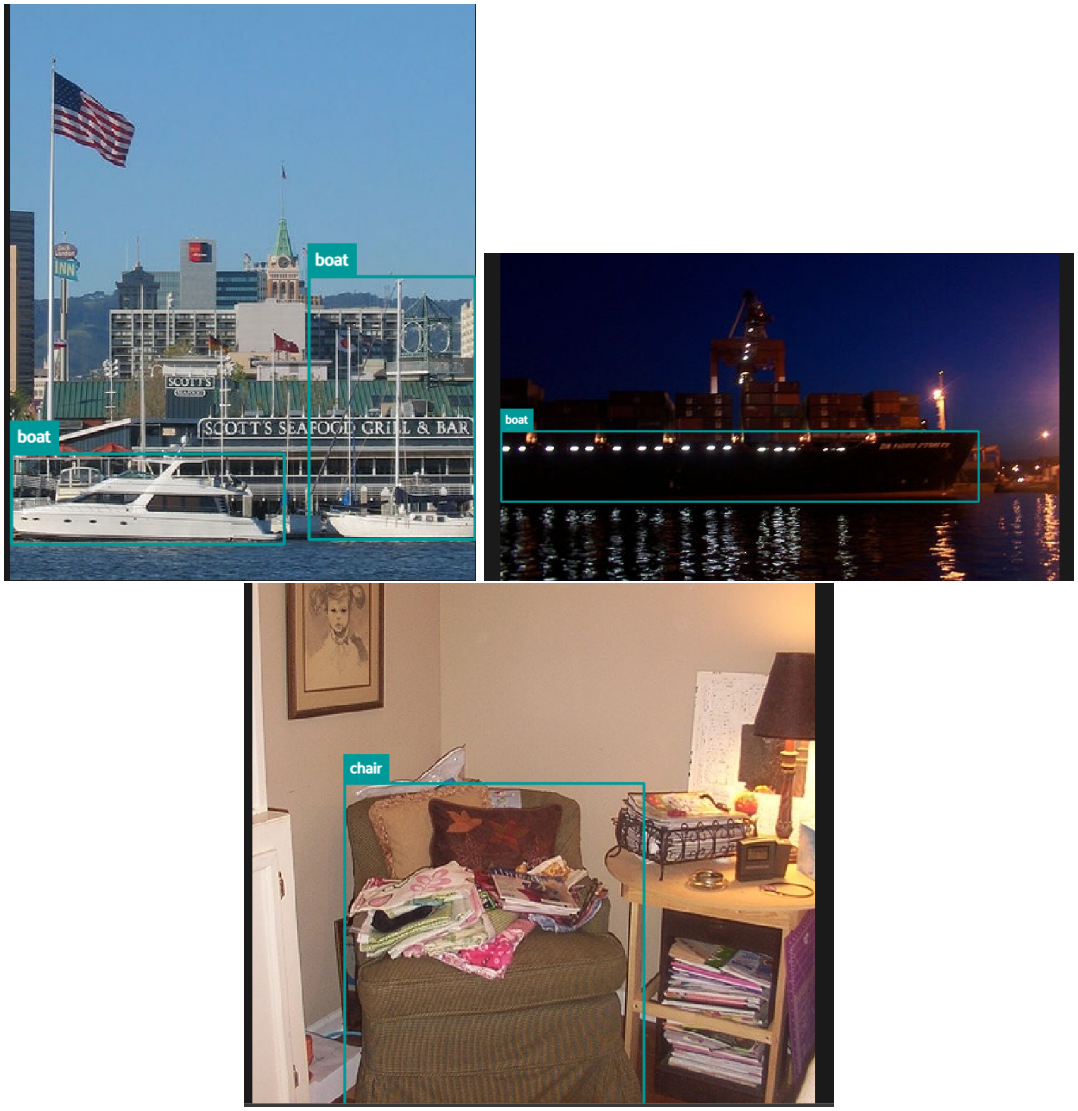


Figure 13: SSD:failure voc

3.2.4 Weaknesses and Strengths

- Strength:
  - SSD is a single-stage object detection method that directly predicts object bounding boxes and class probabilities, eliminating the need for a separate proposal generation stage. This makes it faster.
  - Multi-scale Feature Maps: SSD uses multiple layers in the network to generate feature maps at different scales, allowing it to detect objects of various sizes within an image.
- Weaknesses:
  - Lower Accuracy for Small Objects: Although SSD performs well in detecting larger objects, it may struggle with smaller objects due to the fixed-size default bounding boxes at multiple scales.
  - Localization Accuracy: The accuracy of object localization might not be as high as some two-stage detectors, especially for objects with irregular shapes or extreme poses.

3.3 Faster R-CNN

The Faster R-CNN is an improvement on the Fast R-CNN and the R-CNN. The R-CNN extracts region proposals from the image instead of trying every sliding window and inputs the warped region proposals into a CNN which outputs a classification. Faster R-CNN achieves high accuracy and efficiency in object detection tasks, and it has become a benchmark for comparison with other object detection algorithms. The integration of the RPN allows the model to learn region proposals jointly with object detection, making it more end-to-end trainable and computationally efficient compared to previous methods.

3.3.1 Region Proposal Network (RPN)

This is a neural network that operates on the convolutional feature maps of the input image. The RPN generates region proposals, which are potential bounding boxes around objects.

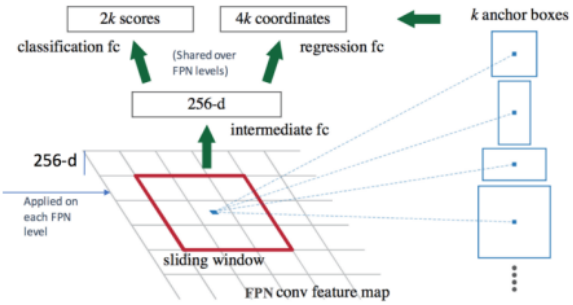


Figure 14: Region Proposal Network

ø

3.3.2 Outputs

Figure 15: RCNN:success coco

Figure 16: RCNN: bad coco

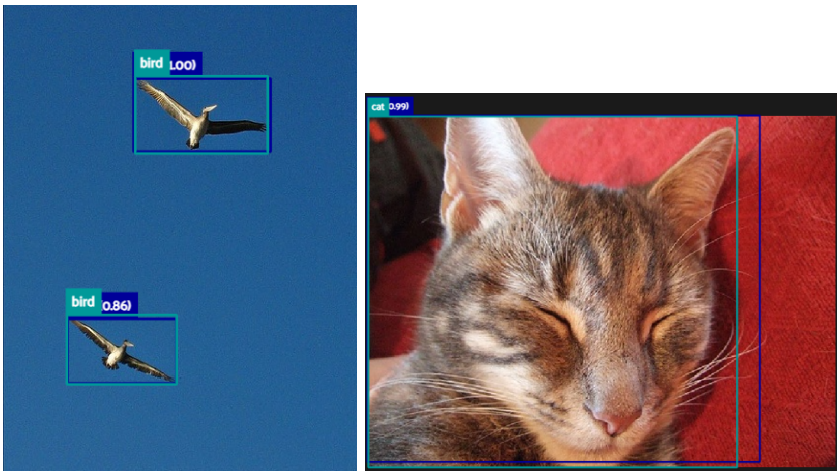


Figure 17: RCNN:success voc

Figure 18: RCNN: bad voc

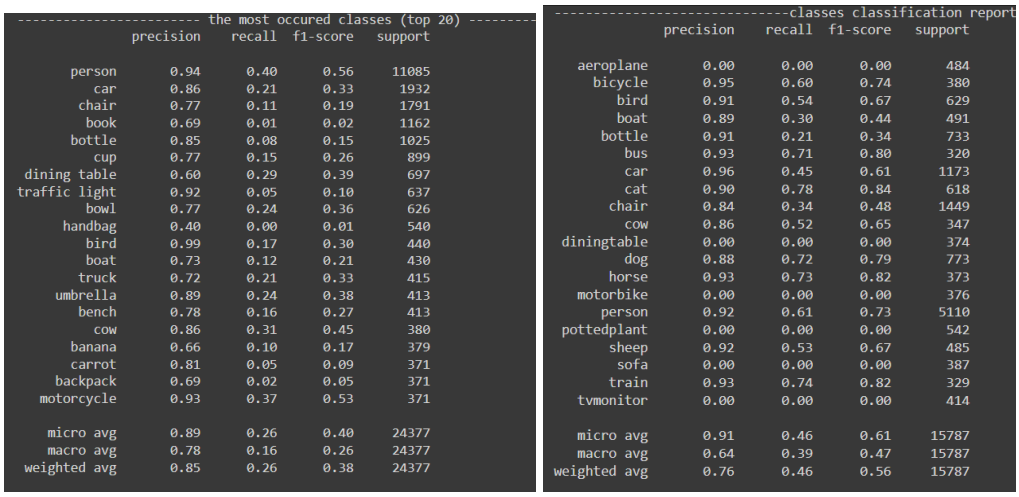
3.3.3 Weaknesses and Strengths

- Suitable Cases: Various Object Classes and Diverse Environments:
- UnSuitable Cases: Real-time Applications with Stringent Latency Requirements because Faster R-CNN may not be the most suitable choice due to its computational complexity.

4 comparison

	SSD-VGG	Retina-ResNet	Faster R-CNN
COCO	0.179	0.292	0.289
VOC	0.272	0.3497	0.351

Table 2: mAP comparison



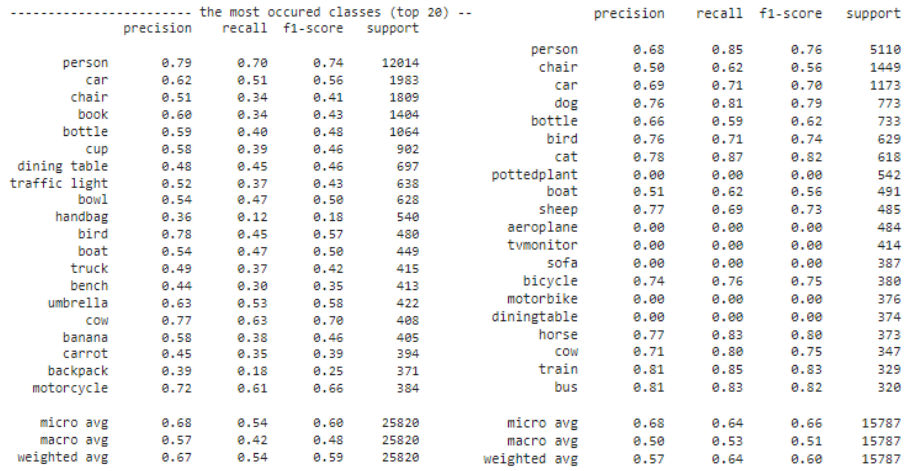
---------- the most occured classes (top 20) ---------

	precision	recall	f1-score	support
person	0.94	0.40	0.56	11085
car	0.86	0.21	0.33	1932
chair	0.77	0.11	0.19	1791
book	0.69	0.01	0.02	1162
bottle	0.85	0.08	0.15	1025
cup	0.77	0.15	0.26	899
dining table	0.60	0.29	0.39	697
traffic light	0.92	0.05	0.10	637
bowl	0.77	0.24	0.36	626
handbag	0.40	0.00	0.01	540
bird	0.99	0.17	0.30	440
boat	0.73	0.12	0.21	430
truck	0.72	0.21	0.33	415
umbrella	0.89	0.24	0.38	413
bench	0.78	0.16	0.27	413
cow	0.86	0.31	0.45	380
banana	0.66	0.10	0.17	379
carrot	0.81	0.05	0.09	371
backpack	0.69	0.02	0.05	371
motorcycle	0.93	0.37	0.53	371
micro avg	0.89	0.26	0.40	24377
macro avg	0.78	0.16	0.26	24377
weighted avg	0.85	0.26	0.38	24377

----------classes classification report

	precision	recall	f1-score	support
aeroplane	0.00	0.00	0.00	484
bicycle	0.95	0.60	0.74	380
bird	0.91	0.54	0.67	629
boat	0.89	0.30	0.44	491
bottle	0.91	0.21	0.34	733
bus	0.93	0.71	0.80	320
car	0.96	0.45	0.61	1173
cat	0.90	0.78	0.84	618
chair	0.84	0.34	0.48	1449
cow	0.86	0.52	0.65	347
diningtable	0.00	0.00	0.00	374
dog	0.88	0.72	0.79	773
horse	0.93	0.73	0.82	373
motorbike	0.00	0.00	0.00	376
person	0.92	0.61	0.73	5110
pottedplant	0.00	0.00	0.00	542
sheep	0.92	0.53	0.67	485
sofa	0.00	0.00	0.00	387
train	0.93	0.74	0.82	329
tvmonitor	0.00	0.00	0.00	414
micro avg	0.91	0.46	0.61	15787
macro avg	0.64	0.39	0.47	15787
weighted avg	0.76	0.46	0.56	15787

Figure 19: SSD results on COCO — VOC



---------- the most occured classes (top 20) --

	precision	recall	f1-score	support
person	0.79	0.70	0.74	12014
car	0.62	0.51	0.56	1983
chair	0.51	0.34	0.41	1809
book	0.60	0.34	0.43	1404
bottle	0.59	0.40	0.48	1064
cup	0.58	0.39	0.46	902
dining table	0.48	0.45	0.46	697
traffic light	0.52	0.37	0.43	638
bowl	0.54	0.47	0.50	628
handbag	0.36	0.12	0.18	540
bird	0.78	0.45	0.57	480
boat	0.54	0.47	0.50	449
truck	0.49	0.37	0.42	415
bench	0.44	0.30	0.35	413
umbrella	0.63	0.53	0.58	422
cow	0.77	0.63	0.70	408
banana	0.58	0.38	0.46	405
carrot	0.45	0.35	0.39	394
backpack	0.39	0.18	0.25	371
motorcycle	0.72	0.61	0.66	384
micro avg	0.68	0.54	0.60	25820
macro avg	0.57	0.42	0.48	25820
weighted avg	0.67	0.54	0.59	25820

	precision	recall	f1-score	support
person	0.68	0.85	0.76	5110
chair	0.50	0.62	0.56	1449
car	0.69	0.71	0.70	1173
dog	0.76	0.81	0.79	773
bottle	0.66	0.59	0.62	733
bird	0.76	0.71	0.74	629
cat	0.78	0.87	0.82	618
pottedplant	0.00	0.00	0.00	542
boat	0.51	0.62	0.56	491
sheep	0.77	0.69	0.73	485
aeroplane	0.00	0.00	0.00	484
tvmonitor	0.00	0.00	0.00	414
sofa	0.00	0.00	0.00	387
bicycle	0.74	0.76	0.75	380
motorbike	0.00	0.00	0.00	376
diningtable	0.00	0.00	0.00	374
horse	0.77	0.83	0.80	373
cow	0.71	0.80	0.75	347
train	0.81	0.85	0.83	329
bus	0.81	0.83	0.82	320
micro avg	0.68	0.64	0.66	15787
macro avg	0.50	0.53	0.51	15787
weighted avg	0.57	0.64	0.60	15787

Figure 20: RCNN results on COCO — VOC

	precision	recall	f1-score	support
person	0.89	0.64	0.75	11384
car	0.81	0.51	0.63	1967
chair	0.71	0.23	0.35	1798
book	0.65	0.06	0.12	1165
bottle	0.80	0.36	0.49	1044
cup	0.78	0.39	0.52	903
dining table	0.70	0.25	0.37	697
traffic light	0.71	0.39	0.51	638
bowl	0.73	0.35	0.48	628
handbag	0.57	0.11	0.18	540
bird	0.89	0.46	0.60	462
boat	0.70	0.33	0.45	431
truck	0.59	0.35	0.44	415
bench	0.64	0.24	0.35	413
umbrella	0.81	0.45	0.58	421
cow	0.82	0.61	0.70	391
banana	0.70	0.22	0.34	383
carrot	0.69	0.20	0.32	376
backpack	0.60	0.18	0.28	371
motorcycle	0.82	0.55	0.66	377
micro avg	0.84	0.47	0.60	24804
macro avg	0.73	0.35	0.46	24804
weighted avg	0.81	0.47	0.58	24804

	precision	recall	f1-score	support
aeroplane	0.00	0.00	0.00	484
bicycle	0.80	0.75	0.77	380
bird	0.87	0.71	0.78	629
boat	0.75	0.52	0.61	491
bottle	0.80	0.49	0.61	733
bus	0.89	0.80	0.84	320
car	0.81	0.69	0.75	1173
cat	0.84	0.89	0.86	618
chair	0.70	0.48	0.57	1449
cow	0.77	0.75	0.76	347
diningtable	0.00	0.00	0.00	374
dog	0.80	0.80	0.80	773
horse	0.82	0.83	0.82	373
motorbike	0.00	0.00	0.00	376
person	0.81	0.81	0.81	5110
pottedplant	0.00	0.00	0.00	542
sheep	0.87	0.70	0.77	485
sofa	0.00	0.00	0.00	387
train	0.83	0.81	0.82	329
tvmonitor	0.00	0.00	0.00	414
micro avg	0.80	0.61	0.69	15787
macro avg	0.57	0.50	0.53	15787
weighted avg	0.67	0.61	0.63	15787

Figure 21: RetinaNet results on COCO - VOC