Deep Learning: Classification

Object Detection and Instance Semantic Segmentation

Pascal VOC 2012 evaluated with COCO Metrics

INM 705 Coursework

Azucena Ascencio-Cabral

Implementation

- *Dataset* : Pascal VOC 2012
  - Images
  - Ground Truth
  - XML Annotations
- *Environment*
  - Python
- *Framework*
  - PyTorch
- *Hardware*
  - Google Colab GPUs
  - 4 workers

Implementation - Architectures

- Models implemented using *PyTorch*
  - Faster RCNN-50-FPN - off the shelf pretrained model
  - Mask RCNN-50-FPN - off the shelf pretrained model
  - Mask RCNN-101-FPN - Built on ResNet-101-FPN backbone network and Anchor Generator 16, 32, 64, 128, 256, 512, aspect ratios: (0.5, 1.0, 2.0)
    - ✓ ***Pretrained parameter on ImageNet, sourced from Facebook Benchmark for Transfer Learning [1]***
    - ✓ ***Pretrained parameters on COCO sourced from J. Yang github repository [2]***

Pascal VOC 2012 -Benchmark

Dataset	Benchmark	mAP at IoU = 0.5	Pretrained
Pascal VOC 2012	Faster R-CNN-VGG-16 [1]	67.0%	ImageNet
Pascal VOC 2007 + 2012	Faster RCN ResNet-101[2]	74.9%	ImageNet
Pascal VOC 2012	Attention based Faster-RCNN-50 [3]	87.2% *Animal classes	COCO
Pascal VOC 2012	Semi- convolutional Mask RCNN- 101-FPN [4]	69.9%	ImageNet
Pascal VOC 2012	Mask RCNN- 101-FPN	69% as cited in [4]	ImageNet

Implementation

- **Dataset Class PascalVOC:**

  voc_classes: '__background__', 'aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus','car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor'

  - *Inputs :* Images, Masks and annotations
  - *Outputs:* PIL Images, targets
  - *Targets coco style* : "area" , "boxes", "iscrowd", "labels", "image_id", "masks'

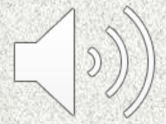
- **Transformations:**
  - Images and masks *resized* to (800 x 800). Boxes *re-scaled to 800/x and 800/y*.
  - Images and targets *transformed to tensors*
  - No padding.
  - *Horizontal flip* for each image of *probability 0.5* during training

Building the Models

```
def maskrcnn_resnet101_fpn(pretrained=False, progress=True, num_classes=91, pretrained_backbone=True,
                           trainable_backbone_layers=3, **kwargs):

    assert trainable_backbone_layers <= 5 and trainable_backbone_layers >= 0
    # dont freeze any layers if pretrained model or backbone is not used
    if not (pretrained or pretrained_backbone):
        trainable_backbone_layers = 5
    if pretrained:
        # no need to download the backbone if pretrained is set
        pretrained_backbone = False
    backbone = resnet_fpn_backbone('resnet101', pretrained_backbone,
                                   trainable_layers = trainable_backbone_layers)
    model = MaskRCNN(backbone, num_classes, **kwargs)
    return model
```

```
def get_transform(train):
    transforms = []
    # converts the image, a PIL image, into a PyTorch Tensor
    transforms.append(T.ToTensor())
    if train:
        # randomly flip the trainin and ground-truth for augmentation
        transforms.append(T.RandomHorizontalFlip(0.5))
    return T.Compose(transforms)
```

```
if backbone == "resnet_101_fpna":

    print("Maskrcnn_101_fpn - pretrained on ImageNet")

    model = maskrcnn_resnet101_fpn(pretrained=False, num_classes= num_classes)
    #From facebook benchmark, train ed with imagenet"
    pretrained_weights= torch.load(pretrained_path +
                                   'modified_model_maskrcnn_101_fpn.pth')
    model_dict = model.state_dict()
    #Replace weights from the backbone
    pretrained_weights = {k: v for k, v in pretrained_weights.items() if k in model_dict}

    model_dict.update(pretrained_weights)
    model.load_state_dict(model_dict)

    #create an anchor_generator for the FPN which by default has 5 outputs
    if anchor == True:
        anchor_generator = AnchorGenerator(
        sizes=tuple([(16, 32, 64, 128, 256, 512) for _ in range(5)]),
        aspect_ratios = tuple([(0.5, 1.0, 2.0) for _ in range(5)]))
        model.rpn.anchor_generator = anchor_generator
        model.rpn.head = RPNHead(256, anchor_generator.num_anchors_per_location()[0])
        print("Anchors", anchor_generator)

    # get number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    # replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    # get the number of input features for the mask classifier
    in_features_mask = model.roi_heads.mask_predictor.conv5_mask.in_channels
    hidden_layer = 256
    # replace the mask predictor with a new one
    model.roi_heads.mask_predictor = MaskRCNNPredictor(in_features_mask, hidden_layer,
                                                       num_classes)
```



Training hyperparameters

- *Batch size* : 2 and 4
- *Learning rate*: 0.001, 0.005, 0.0001
- *Learning scheduler* : step 5, after 5 or 10 epochs
- *Weight decay*: 0.0001, 0.0005
- *Momentum* : 0.9
- *Optimizer* : SGD, Adam

- **VOC metrics**

- *A prediction with IoU > 0.5 is considered as True Positive*
- *Two predictions of IoU 0.5 and 0.9 would be given the same weight*

Evaluation metrics
12 metrics – COCO Style

```
Average Precision (AP):
   AP                 % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50         % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75         % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small           % AP for small objects: area < 32^2
   AP^medium          % AP for medium objects: 32^2 < area < 96^2
   AP^large           % AP for large objects: area > 96^2
Average Recall (AR):
   AR^max=1           % AR given 1 detection per image
   AR^max=10          % AR given 10 detections per image
   AR^max=100         % AR given 100 detections per image
AR Across Scales:
   AR^small           % AR for small objects: area < 32^2
   AR^medium          % AR for medium objects: 32^2 < area < 96^2
   AR^large           % AR for large objects: area > 96^2
```

Image taken from COCO common Objects in Context [5]

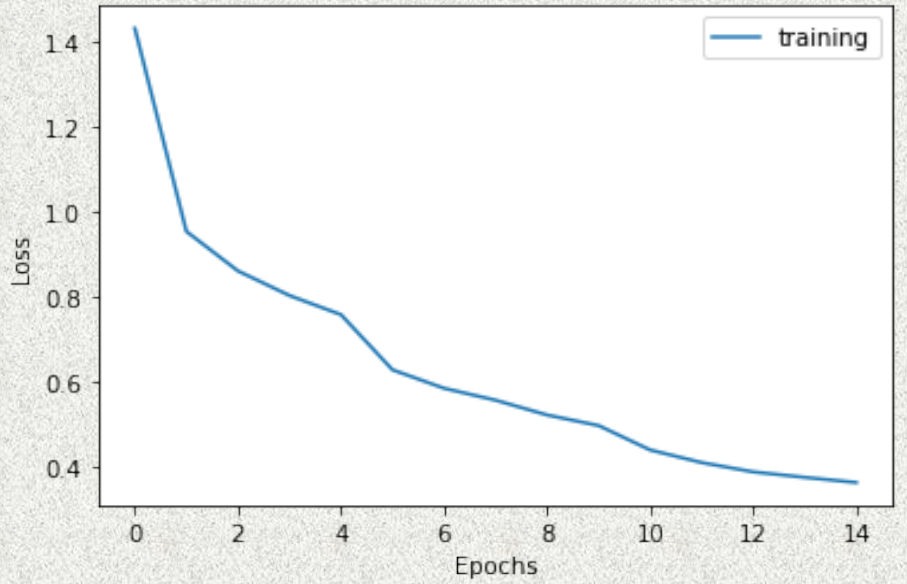
Advantages

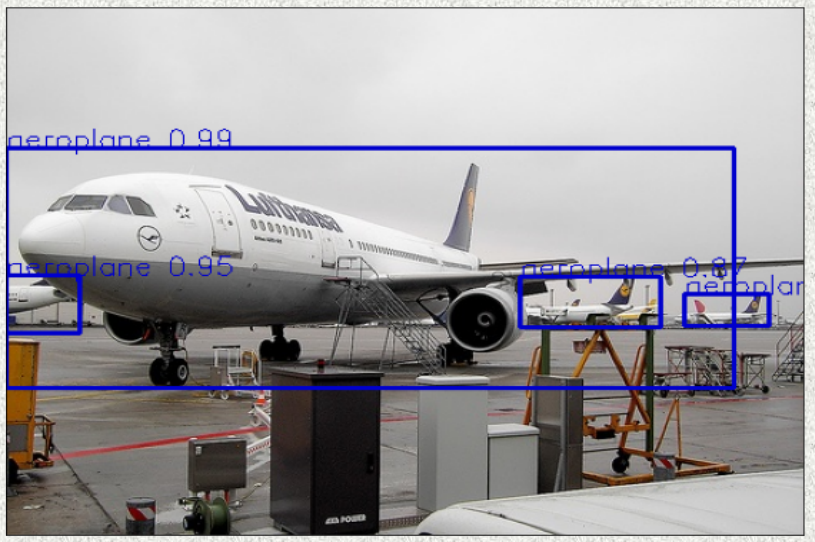
- **Use a range of IoU threshold values, and calculate mAP for each IoU, computes their average to get the final mAP.**
- ***[0:.01:1] R=101 recall thresholds for evaluation***
- $$mAPCOCO = \frac{mAP0.50 + mAP.55 + \cdots + mAP0.95}{10}$$
- **AP average over all categories**

Experiment Results
Highest and lower performance

Performance Methods									
Method	Epochs	L rate	Optimizer	AP @ IoU=0.50:0.95 box	mAP @ IoU 0.50 box	mAP @ IoU 0.75 box	IoU=0.50:0.95 segment	mAP @ IoU 0.50 segment	mAP @ IoU 0.75 segment
Faster - 50	15	0.005	SGD	89.2	99.00	97.5	N/A	N/A	N/A
Faster - 50	20	0.005	SGD	85.9	98.90	97.1	N/A	N/A	N/A
Faster - 50	5	0.005	SGD	49.6	80.00	55.2	N/A	N/A	N/A
Mask - 50	15	0.005	SGD	54.0	82.30	60.8	43.6	71.0	45.7
Mask - 50	20	0.005	SGD	53.2	82.10	60.2	43.4	71.7	45.2
Mask - 50	12	0.001	SGD	54.4	81.70	60.6	43.8	71.4	47.5
Mask – 101a with anchor generator	20	0.0001	Adam	35.8	60.0	38.8	31.6	54.1	31.9
Mask – 101a	12	0.001	SGD	25.9	56.4	19.8	23.6	48.9	20.0
Mask – 101a with anchor generator	20	0.001	Adam	29.3	55.3	28.3	27.3	48.3	27.0
Mask -101b -with anchor generator	20	0.0001	Adam	38.3	67.4	40.4	31.2	57.0	31.5
Mask – 101b	15	0.005	Adam	33.7	62.30	34.5	29.9	53.9	28.5
Mask – 101b	15	0.001	SGD	29.4	60.1	24.9	26.5	52.5	22.7
Lower Performance Methods									
Faster -50	5	0.001	SGD	80.0	49.6	55.2	-	-	-
Mask - 50	5	0.005	SGD	28.2	54.10	25.2	25.2	47.8	23.2
Mask - 101a	25	0.001	SGD	15.70	41.50	8	7.9	19.6	4.6
Mask - 101a	30	0.001	SGD	16.00	43.20	8.1	11.40	26.70	8.30
Mask - 101b	20	0.001	SGD	25.0	53.90	18.80	22.9	46.4	19.4
Mask - 101b	15	0.001	SGD	29.4	58.60	27.1	27.80	50.70	26.70

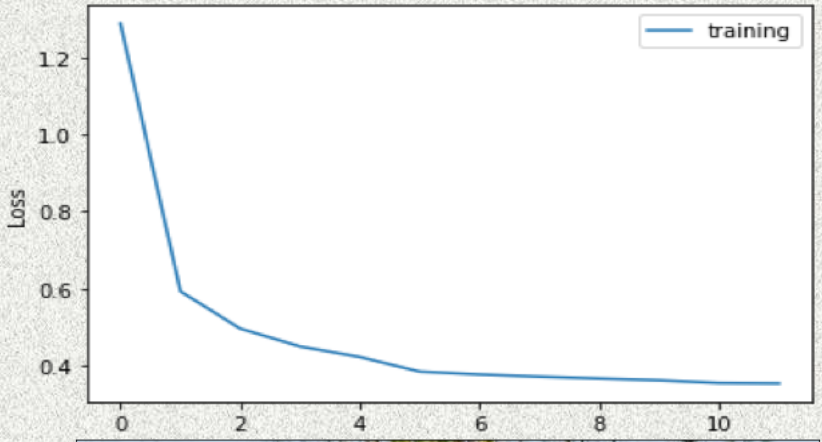
Faster-RCNN ResNet-50

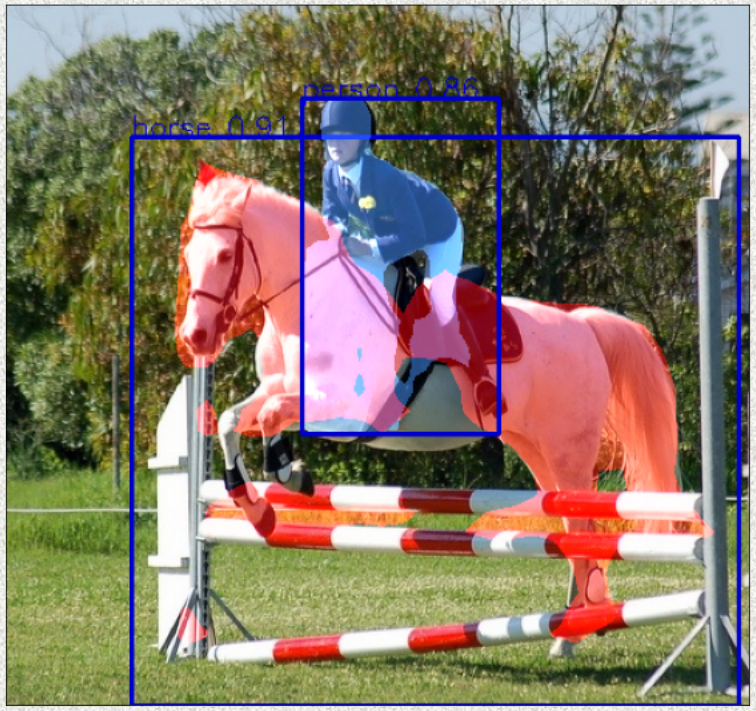





```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.892
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.990
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.975
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.942
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.923
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.889
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.651
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.918
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.919
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.951
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.939
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.915
```

Mask RCNN ResNet-50 FPN

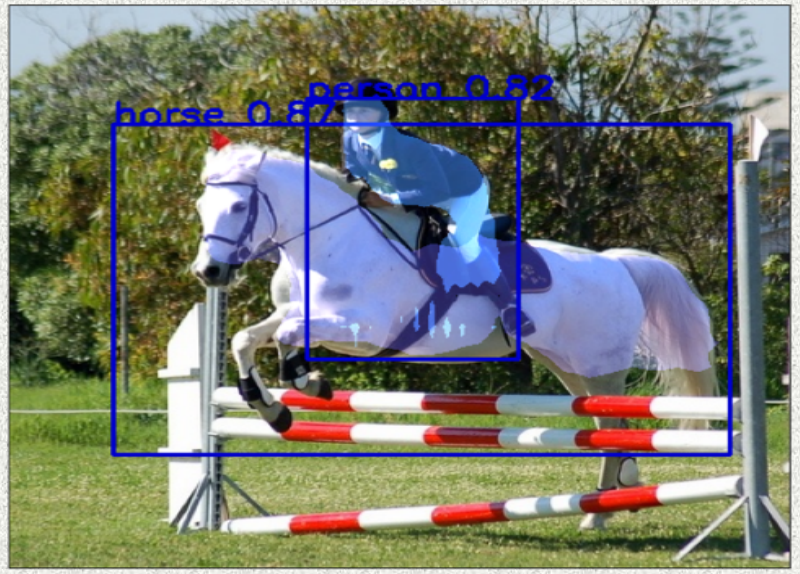


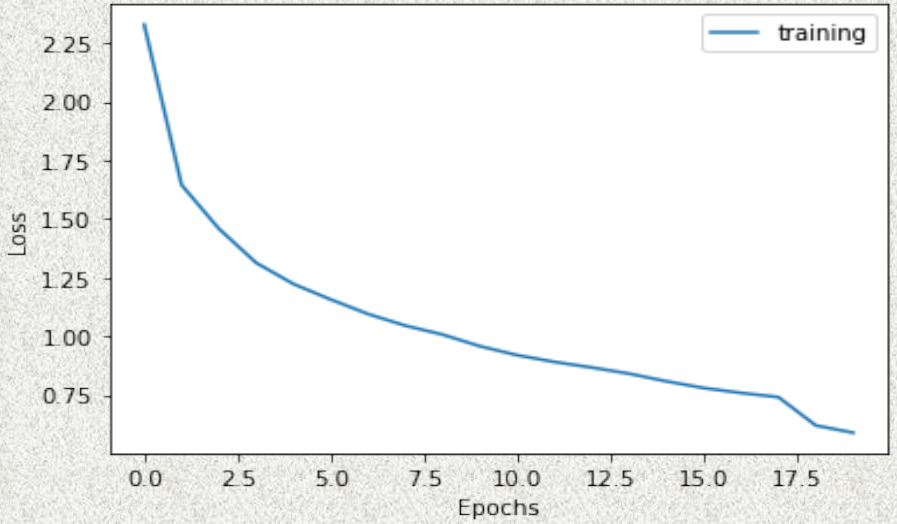



```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.540
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.823
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.608
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.269
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.454
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.595
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.459
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.632
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.637
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.354
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.536
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.685
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.436
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.710
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.457
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.038
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.316
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.529
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.405
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.524
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.528
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.132
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.429
```

Note: The plot or this run was note available, the displayed plot is for the third best Mask-RCNN-50-FPN
AP @ IoU 0.5= 81.70

Mask RCNN-101 FPN – Anchor Generator



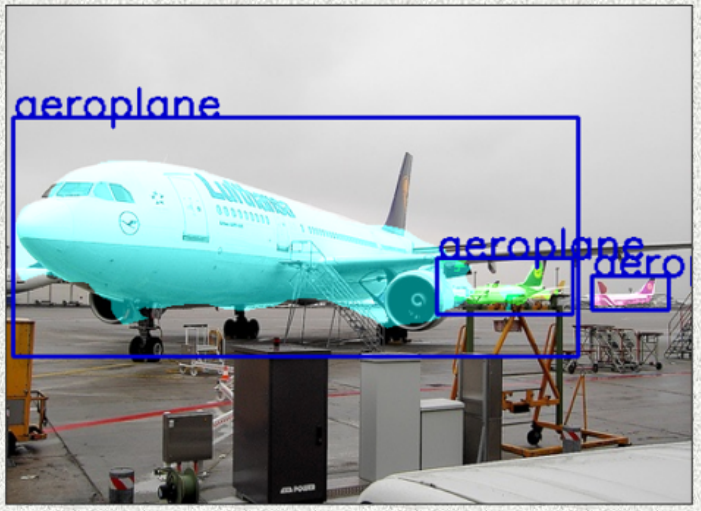


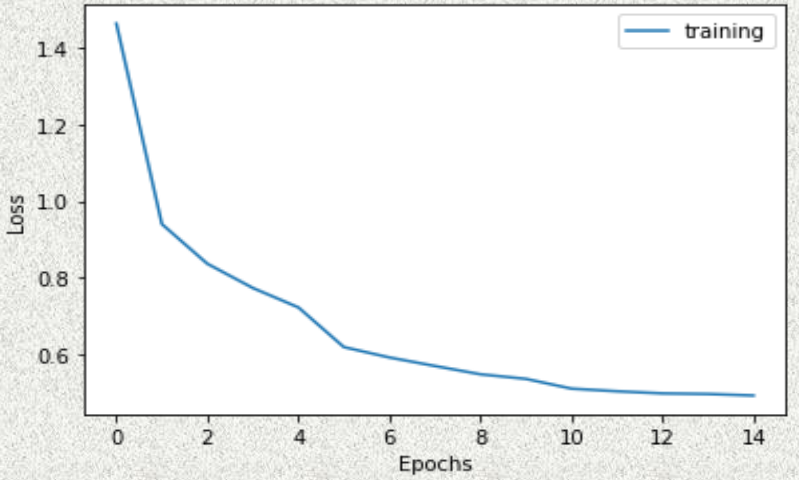
Pretrained Parameters from Facebook - ImageNet

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.358
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.600
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.380
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.177
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.391
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.458
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.459
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.002
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.224
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.494
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.316
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.540
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.319
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.117
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.352
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.331
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.405
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.406
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.156
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.439
```

Mask RCNN ResNet-101 FPN

Pretrained Parameters from Facebook - ImageNet

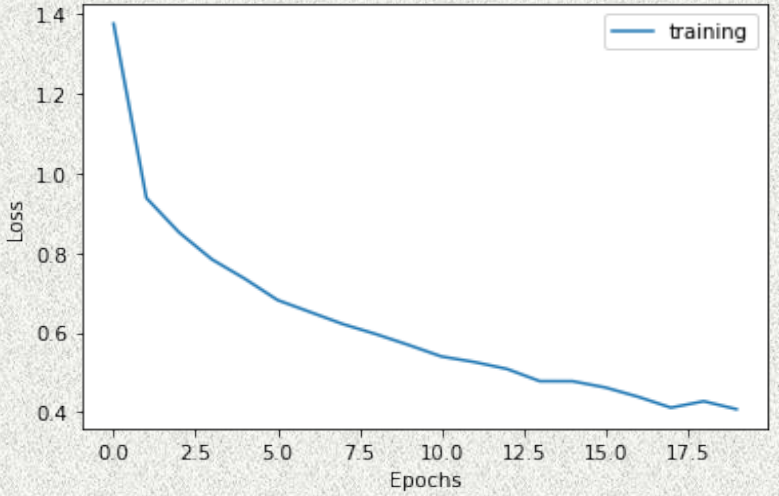


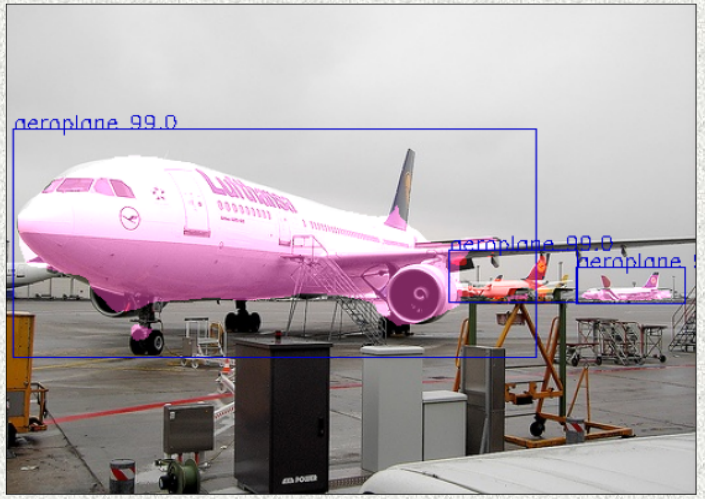



```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.250
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.539
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.188
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.005
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.150
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.273
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.277
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.403
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.416
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.019
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.248
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.446
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.229
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.464
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.194
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.042
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.258
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.267
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.352
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.359
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.150
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.388
```

Mask RCNN 101 FPN – Parameters pretrained on COCO

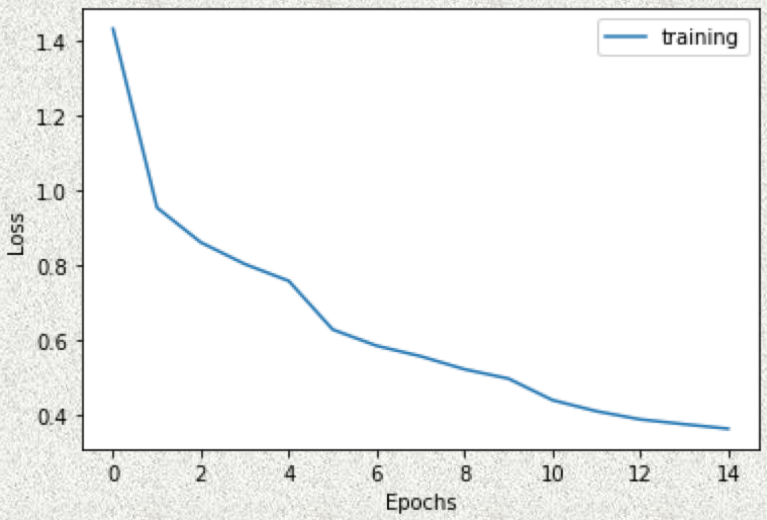
six anchor generators and 3 aspects ratios (16, 32, 64, 128, 256, 512), (0.5, 1.0, 2.0)

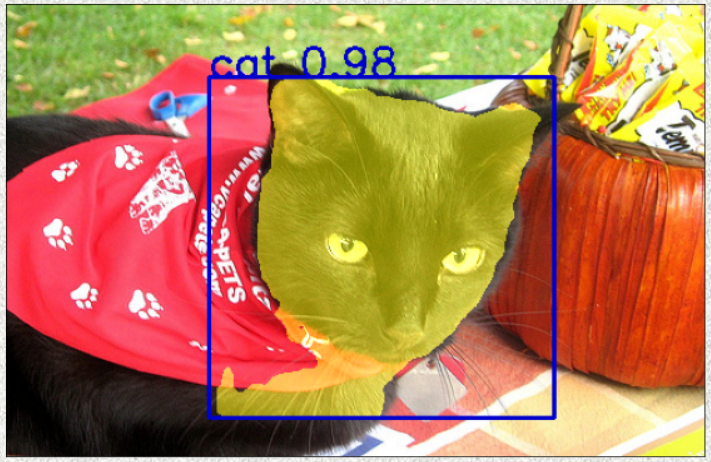





```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.383
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.674
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.404
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.170
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.418
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.367
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.480
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.485
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.272
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.523
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.312
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.570
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.315
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.052
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.347
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.318
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.389
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.391
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.138
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.423
```

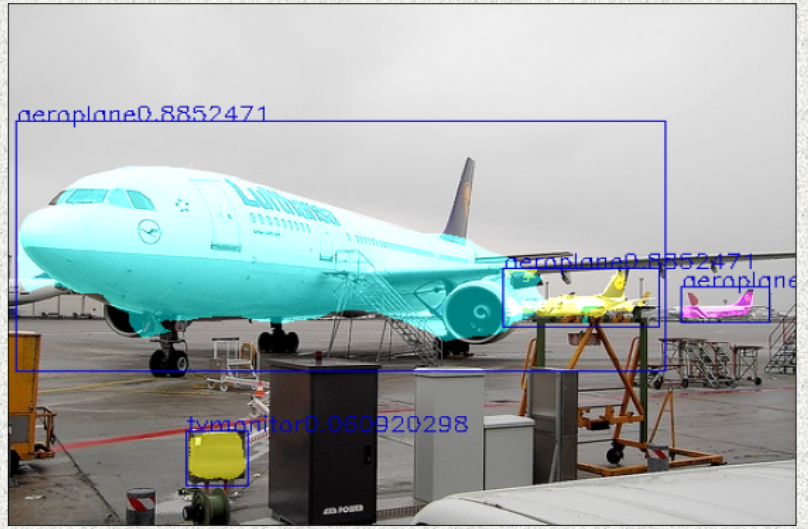
Mask RCNN 101 FPN – Parameters pretrained on COCO

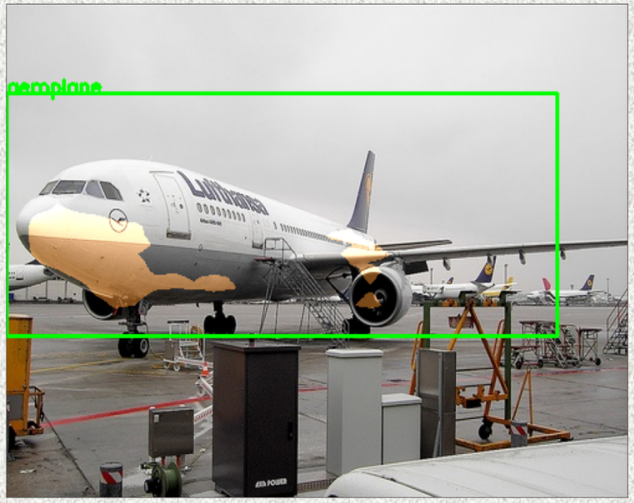


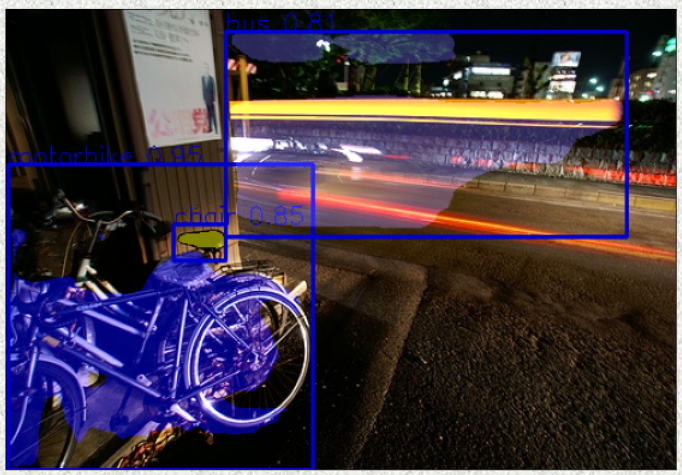



```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.337
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.623
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.345
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.067
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.199
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.365
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.349
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.479
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.481
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.096
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.267
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.509
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.299
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.525
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.285
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.115
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.335
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.319
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.412
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.413
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.033
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.189
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.439
```

Mask RCNN ResNet-101 FPN – Underperformers







Conclusions

- Faster RCNN-50-FPN and Mask RCNN-50-FPN outperformed the benchmark for the Pascal VOC 2012 dataset.
- Mask RCNN-101-FPN reached AP 38.3% , 31.2% and AP 67.4% and 57.0% at IoU=0.5.
- For the long backbone models performance improved when using anchor generators.
- Backbone models require longer training.
- For long backbone architectures with anchor generator are sensitive to images size.
- Better results and faster convergence when fine-tuning on pretrained COCO parameters.
- COCO metrics help to track the performance of the models and identify the sensitivity of the Mask-RCNN-101 model to anchors, backbone and hypermeters.

References

- S. Ren, K. He, R. Girshick, and J. Sun, ‘Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks’, in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.
- J. Yang, *jwyang/faster-rcnn.pytorch*. 2020.
- K. Morabia, J. Arora, and T. Vijaykumar, ‘Attention-based Joint Detection of Object and Semantic Part’, *arXiv:2007.02419 [cs]*, Jul. 2020, Accessed: Jul. 02, 2020. [Online]. Available: http://arxiv.org/abs/2007.02419.
- D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi, ‘Semi-convolutional Operators for Instance Segmentation’, *arXiv:1807.10712 [cs]*, Jul. 2018, Accessed: Jul. 04, 2020. [Online]. Available: http://arxiv.org/abs/1807.10712.