INM705 Deep Learning For Image Analysis
Lab 4 Instructions

Datasets:

Pascal VOC 2012 images are the same (Images dir), now you also need to load bboxes from xml files in directory **annotations**. In the **dataset.py** and **utils.py** files, **extract_bboxes_pascal**, returns a dictionary of ground truth (gt) classes and a dictionary of gt bounding boxes of dimensions (N) and (N,4) respectively, where N is the number of gt objects.

Training your model with Faster R-CNN:

The current implementation of the model uses a ResNet50 as a feature extractor with FPN (feature proposal network), it was pretrained on MS COCO 2017 dataset and has 91 output classes. You can evaluate it on out-of-the-box on Pascal VOC as Pascal's list of classes is a subset of MS COCO. To finetune it (it's an exercise), it's better to re-initialize the class_logits layer in region of interest (RoI) to account for the change in the number of classes from 90+1 to 20+1. Follow the instructions in the previous labs on how to do it.

Without creating your own anchor generator and RoIAlign, you can tweak other parameters of Faster R-CNN, this is done by setting the relevant values of keywords in the model kwargs dictionary **frcnn_args.** For example, to set the number of anchors sampled during training to 256, you need

```
frcnn_args['rpn_batch_size_per_image'] = 256
model = models.detection.faster_rcnn.fasterrcnn_resnet50_fpn(pretrained=True, **frcnn_args)
```

EDIT: don't forget that you can't download on Hyperion. If you use the pretrained=False argument, the model will still try to download some weights. If you'd rather started training from scratch, you have to use another argument. For that you need to investigate the source code for Faster R-CNN

If you investigate faster_rcnn.py interface, you can see all arguments that the model uses. Some of them will be discussed during the lab. During training the inputs in Faster R-CNN are a list of Tensors (images) and a list of dict of Tensors (classes an bboxes).

Exercises (suggested):

1. Complete the code for the evaluation of the model's average precision on Pascal VOC 2012 dataset and evaluate the provided model **pascal_voc_detection_model.pth** on the test data using the **compute_ap** method provided.

2. Try different feature extractors, such as ResNet18, ResNet34. Do you get a drop in performance? (First, you will have to create a different backbone using **resnet_fpn_backbone** method, then, instantiate **FasterRCNN** model with this backbone and the same number of classes).

3. Create your own **AnchorGenerator** and **RoIAlign** modules and pass them as keyword arguments in the fasterrcnn_resnet50_fpn.

4. Create a local copy of the detection package from torchvision. You will need to fix some imports.

5a (advanced). Amend the code in **backbone_utils.py (resnet_fpn_backbone)** to change the connections between the backbone and change the architecture of FPN (e.g. the number of output channels of FPN, it is 256 currently). For example, to connect FPN only to the last layer of ResNet50 and output the last feature layer:

returned_layers=[4]

This will create a dictionary **return_layers**:

return_layers = {'layer4': '0'}

Here 'layer4' must match the name of the backbone (ResNet50 in this case) layer, otherwise you get an error:

ValueError: return_layers are not present in model

Also string **'0'** will be the name of the FPN's output. You also must know the number of feature maps output by this layer (layer4, 2048) to adjust the number of inputs in FPN (**in_channels_list**), as otherwise the code will not be correct. You will have to re-initialize some Faster R-CNN layers, such as **box_roi_pool** (RoI alignment layer) to accept a different number of layers, e.g. for one FPN output.

```
if box_roi_pool is None:
    box_roi_pool = MultiScaleRoIAlign(
                        # single feature map
                        featmap_names=['0'],
                        output_size=7,
                        sampling_ratio=2)
```

Here string **'0'** refers to the name of the feature map output by FPN, so it must match the one defined in **resnet_fpn_backbone** method's **return_layers** dictionary value.

Also, the actual layers in the feature extractor selected for training are

```
layers_to_train = ["layer4", "layer3", "layer2", "layer1", "conv1"][:trainable_layers]
```

'trainable layers' is the argument you need to use in order to select the layers to train. Obviously they come in a specific order. Write some hacking trick at the model instantiation stage to chage this order, that is, to replace this list. Obviously, this has to be reflected in the input-output choices for FPN.

5b (advanced). Using the approach above, you can truncate the output of the feature extractors, by deleting some of the layers. Do you always get a drop in performance?

6. Save the trained model and evaluate on Pascal VOC validation set, compare to the pretrained model.