HOMEWORK ASSIGNMENT #6

DUE: Tuesday, November 23, 2021, 5:00PM

CSCI677: Advanced Computer Vision, Prof. Nevatia

Fall Semester, 2021

This is a programming assignment on fine-tuning an object detection model. Due to limited time available to complete, we have essentially provided the needed code and pointers to helpful tutorial material. The following describes the basic library to use, the dataset to adapt to, the evaluation code and the experiments you should conduct.

a) Library:

We will use Detectron2 software (https://github.com/facebookresearch/detectron2). It is high-level wrapped code for object detection powered by PyTorch. Though Detectron2 supports multiple computer vision tasks, including instance segmentation, we will only use its object detection functionality in this assignment.

To install Detectron2, use the following code in Colab:

```
!pip install pyyaml==5.1
!pip install detectron2 -f
pip install detectron2 -f \
https://dl.fbaipublicfiles.com/detectron2/wheels/cu102/torch1.9/index.html
```

NOTE: You will need to restart your Colab runtime once to finish installation.

The task is to fine-tune a Faster RCNN objection detection model. The pretrained model can be found in Detectron2 model zoo (https://detectron2.readthedocs.io/_modules/detectron2/model_zoo/model_zoo.html).

You are asked to experiment with and compare two models "COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml" and "COCO-Detection/retinanet_R_50_FPN_3x.yaml".

Detectron2 provide a config dictionary, so that you can easily setup your experiment. Following is an example, you may need to substitute filenames and parameters as needed.

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"))
cfg.OUTPUT_DIR = 'MyVOCTraining'
cfg.DATASETS.TRAIN = ("voc_2007_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 1
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml")  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 1
cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
cfg.SOLVER.MAX_ITER = 3000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 20
```

You can choose a model with `cfg.merge_from_file()` and set initial weights with `cfg.MODEL.WEIGHTS`. You can choose training/testing dataset with `cfg.DATASETS.TRAIN/TEST`.

You can also find another example in Detectron2's official tutorial at https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5#scrollTo=7unkuuiqLdqd .

b) Dataset:

We will use the Pascal VOC dataset for object detection (pretrained model is trained on MSCOCO dataset). You will need to download the dataset to Colab but you don't need to prepare the dataset by yourself.

To download the dataset, you can use following code in Colab

```
!wget
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-
Nov-2007.tar
!tar -xvf VOCtrainval_06-Nov-2007.tar
```

Detectron2 uses “datasets” as the default directory for data. Change folder name to “datasets/VOC2007” by using `!mv VOCdevkit datasets`

c) Evaluation:

Detectron2 provides a high-level evaluator for Pascal VOC, you can use it simply as:

```
from detectron2.evaluation import PascalVOCDetectionEvaluator.
```

Evaluator returns Average Precision based on IoU (intersection-of-union); you only need to report AP50 which is the primary metric for Pascal VOC.

d) Your work:

1. Build a pipeline for fine-tuning object detection on Pascal VOC. You can use the code given above and refer to available tutorials if helpful.
2. Fine tune the network and show qualitative and quantitative results of the model you have trained. For qualitative results, show the object detection results for some images; you can use Detectron2 high level API in `detectron2.utils.visualizer.`
3. Show the evolution of loss functions. Detectron2 saves Tensorboard record by default. To use Tensorboard file, you can do either 1) Download record and run Tensorboard on your local machine or 2) use Colab(jupyter) integrated Tensorboard to read curves.

   ```
   %load_ext tensorboard
   %tensorboard --logdir PATH_TO_YOUR_RECORD_FOLDER
   ```
4. Repeat steps 2 and 3 above for the two models specified earlier (on page 1).

SUBMISSION:

Your submission file should contain the following:

1. The code you have written (in report PDF, .py file or .ipynb file, first is preferred for grading), including brief descriptions/comments of each function/training block.
2. Show qualitative and quantitative results of the two models and provide a comparison between the two. Include training curves for the two methods.