HOMEWORK ASSIGNMENT #5

DUE: Tuesday, November 19, 2024, 3:00PM

CSCI677: Advanced Computer Vision, Prof. Nevatia

Fall Semester, 2024

This is a programming assignment on fine-tuning two object detection models. 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 ask you to experiment with fine tuning of two different object detection models. One is a Faster RCNN with FPN; the other is a DETR model.

Faster RCNN:

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 an objection detection model. The pretrained model can be found in Detectron2 model zoo

https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md

You are asked to experiment with the Faster RCNN “R50-FPN, 3x” model, ID# 137849458.

Detectron2 provides 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_2012_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 .

DETR Model:

We will use the official code for DETR at: https://github.com/facebookresearch/detr. To setup for DETR model in colab, run:

```
!git clone https://github.com/facebookresearch/detr.git
```

You can use the following command to finetune DETR on a COCO 2017 format dataset:

```
! python main.py --batch_size 2   --resume
https://dl.fbaipublicfiles.com/detr/detr-r50-e632da11.pth     --
coco_path <path to dataset>  --output_dir <path to save checkpoints> -
-lr <lr> --lr_backbone <lr_backbone> --epochs 2
```

You need to pick a good LR for finetuning. A good idea is to pick a LR smaller than the

default LR used for training from scratch. You will find the default LR in `main.py`. You can also finetune the model for more epochs.

To avoid evaluating on the entire dataset during training, please comment the following lines in `main.py`:

```
  # line 214-217
   test_stats, coco_evaluator = evaluate(
            model, criterion, postprocessors, data_loader_val, base_ds,
device, args.output_dir
        )
  # and line 223-236
if args.output_dir and utils.is_main_process():
            with (output_dir / "log.txt").open("a") as f:
                f.write(json.dumps(log_stats) + "\n")

            # for evaluation logs
            if coco_evaluator is not None:
                (output_dir / 'eval').mkdir(exist_ok=True)
                if "bbox" in coco_evaluator.coco_eval:
                    filenames = ['latest.pth']
                    if epoch % 50 == 0:
                        filenames.append(f'{epoch:03}.pth')
                    for name in filenames:

torch.save(coco_evaluator.coco_eval["bbox"].eval,
                                      output_dir / "eval" / name)
```

To visualize the detection results, you can check: https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_attention.ipynb

b) Dataset:

We will use the Pascal VOC dataset for object detection (pretrained models ares trained on MSCOCO dataset). Even though the dataset is the same for the two detectors, they use different formats so you have to process and load differently.

For the Detectron model, 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/voc2012/VOCtrainval_11-May-2012.tar
!tar -xvf VOCtrainval_11-May-2012.tar
```

Detectron2 uses "datasets" as the default directory for data. Change folder name to "datasets/VOC2012" by using `!mv VOCdevkit datasets`

For the DETR model, dataset setup is more complex. DETR only supports COCO 2017 dataset format, which follows the file structure of:

```
- path_to_coco/
    -annotations/
      -- instances_train2017.json
      -- instances_val2017.json
    -train2017/ images for training
    -val2017/
```

While the VOC dataset format is:

```
- VOCdevkit/
  - VOC2012
      -Annotations/ annotation xml files for each image
      -ImageSets/
      -JPEGImages/
      -SegmentationClass
      -SegmentationObject
```

To convert the VOC dataset to COCO format, you need to convert the xml annotation files in VOC into a single json file. There are several important details regarding the data format conversion as well. We provide a notebook for the data format conversion at https://colab.research.google.com/drive/10IPTFxi0eu41xI3xS_iZjjkWVRnuBgtA?usp=sharing which includes all the details.

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.

To evaluate DETR, you can simply run:

```
!python main.py --batch_size 2   --resume <path to your
checkpoints> --eval --no_aux_loss     --coco_path
../VOC_coco_format/
```

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 step 2 for DETR model. It's optional to repeat step 3 for DETR model. To enable Tensorboard for DETR, you need to change the `main.py` file from the official repo.

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. Also compare the performance of the models before and after fine tuning.
3. As the models may be slow to train, number of training epochs may be limited; however, you are still encouraged to try some variations if possible.