Pre-trained version of model

This is done in code with:

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
```

This line saves the model weights to a file called "model_final.pth".

Grading script

Use the following script to load the pre-trained model and perform testing:

```
# Load the pre-trained model

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5

predictor = DefaultPredictor(cfg)



# Perform testing on a given dataset

evaluator = COCOEvaluator("test_dataset", cfg, False, output_dir=cfg.OUTPUT_DIR)

test_loader = build_detection_test_loader(cfg, "test_dataset")

inference_on_dataset(predictor.model, test_loader, evaluator)



# Visualize the results

test_dicts = DatasetCatalog.get("test_dataset")

for d in random.sample(test_dicts, 3):
```

```
img = cv2.imread(d["file_name"])

visualizer = Visualizer(img[:, :, ::-1], metadata=test_metadata, scale=0.5)

outputs = predictor(img)

v = visualizer.draw_instance_predictions(outputs["instances"].to("cpu"))

plt.imshow(v.get_image()[:, :, ::-1])

plt.show()
```

Python Version

This code was developed and tested using Python 3.8. Please ensure this version or later is installed on system.

Installation

1. Ensure to have Jupyter Notebook installed on your system.
2. In Jupyter Notebook cell, install these packages:

```
!pip install -U torch torchvision

!pip install cython

!git clone https://github.com/facebookresearch/detectron2.git

!pip install -e detectron2

!pip install git+https://github.com/facebookresearch/fvcore.git

!pip install pycocotools
```

Description

There is no requirement for folder structure in the root directory. Ensure to change the dataset directory to the desired folder and run the scripts.

How to Run the Code

To run the code, first, make sure to follow step 2 in Installation to install and clone Detectron2 repository from GitHub to your local system. Next, run the code with the attached ipynb file.

Output

The output of the code includes the trained model file (model_final.pth) and a JSON file (results.json) containing the evaluation results.

Where the Code Saves the Output

The code saves the output in the output/ directory.

Scripts to train model

```
cfg = get_cfg()

cfg.merge_from_file("detectron2/configs/COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml")

cfg.DATASETS.TRAIN = ("train_dataset",)

cfg.DATASETS.TEST = ("val_dataset",)

cfg.DATALOADER.NUM_WORKERS = 2

cfg.MODEL.WEIGHTS = "detectron2://COCO-
Detection/faster_rcnn_R_50_FPN_3x/137849458/model_final_280758.pkl"

cfg.SOLVER.IMS_PER_BATCH = 2

cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 3000

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128


num_classes = get_num_classes_from_coco_json(train_coco_path)

cfg.MODEL.ROI_HEADS.NUM_CLASSES = num_classes

cfg.TEST.EVAL_PERIOD = 1000


cfg.MODEL.DEVICE = "cpu"

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)

trainer = DefaultTrainer(cfg)

trainer.resume_or_load(resume=False)

trainer.train()
```

This script is using the pre-trained Faster R-CNN model with a ResNet-50 backbone from the Detectron2 model zoo. The training data and validation data are specified by "train_dataset" and "val_dataset" respectively. The training process is configured with various hyperparameters such as learning rate, maximum iterations, batch size per image, etc.

Scripts to test model

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5

predictor = DefaultPredictor(cfg)


evaluator = COCOEvaluator("val_dataset", cfg, False, output_dir=cfg.OUTPUT_DIR)

val_loader = build_detection_test_loader(cfg, "val_dataset")

inference_on_dataset(predictor.model, val_loader, evaluator)
```

```
val_dicts = DatasetCatalog.get("val_dataset")

for d in random.sample(val_dicts, 3):

    img = cv2.imread(d["file_name"])

    visualizer = Visualizer(img[:, :, ::-1], metadata=val_metadata, scale=0.5)

    outputs = predictor(img)

    v = visualizer.draw_instance_predictions(outputs["instances"].to("cpu"))

    plt.imshow(v.get_image()[:, :, ::-1])

    plt.show()
```

This script loads the trained model weights from "model_final.pth" and uses it to create a predictor. It then performs inference on the validation set using the created predictor and evaluates the model's performance. It also visualizes predictions on a few randomly selected images from the validation set.

You would input a new test sample by replacing d["file_name"] in the cv2.imread() function with the path to your new test image. The image should be in the same format as the images in your training and validation datasets. The script will then load your image, make a prediction using the trained model, and visualize the prediction.

```
img_path = 'path/to/your/test/image.jpg'

img = cv2.imread(img_path)

visualizer = Visualizer(img[:, :, ::-1], metadata=Val metadata, scale=0.5)

outputs = predictor(img)

v = visualizer.draw_instance_predictions
```

Comment

For our code, we just need to install all the packages in the installation section and then run the code, the root folder will save the pretrain model along with output folder. One more thing, the scripts require large amounts of memory on computer so make sure to have enough space. If you have PC with an NVIDIA graphics card, change “cpu” to “gpu” in the code to save training time.