Contents

List of Figures

User Manual

The instructions described in this part are related to the Windows OS.

1 Setting up the Project Environment

Here, we present the steps necessary for setting up a conda environment for the project. They are as follows:

Step 1: Install the latest conda (Anaconda);

Step 2: Download the source code and extract to the desired location;

Step 3: Enter an Anaconda Prompt terminal window in administrator mode;

Step 4: Navigate to the source code directory;

Step 5: Create a conda environment for the project by running:

```
conda create -n <env_name> -f environment.yml
```

where `<env_name>` (defaults to "aquatic") should be replaced by the name of the environment as desired by the user. Basically, this command creates a new conda environment with all necessary project dependencies.

Step 6: Activate the conda environment via:

```
conda activate <env_name>
```

where `<env_name>` is as above.

Step 7: Add the "data" and "coco_pretrained" directories to the current directory. The "data" folder contains the training, validation and test aquatic dataset images whereas the coco_pretrained folder comprise the baseline model weights. These both are only available on request.

2 Image Enhancement

We train our models with both raw and AMSRCR enhanced aquatic images. This process is carried out automatically by utilising the "run_amsrcr.py" project script. We present the actions required to run the script below:

Step 1: Enter an Anaconda Prompt terminal window in administrator mode;

Step 2: Navigate to the directory of the source code files;

Step 3: Activate the project conda environment created earlier (see Section 1);

Step 4: Run the image enhancement script by entering the following command into the terminal:

```
python run_amsrcr.py <dataset_name>
```

where `<dataset_name>` should be set to either "train_val" or "test". On the other hand, the optional arguments are shown and described in Figure 1 below. By default, these optional values are the ones presented in "constants.py" script given by the `<dataset_name>` argument. For details, please refer to the "run_amsrcr.py" script implementation.

```
usage: run_amsrcr.py [-h] [--orig-dir ORIG_DIR] [--target-dir TARGET_DIR] DATASET_NAME

positional arguments:
  DATASET_NAME          dataset for the AMSRCR image enhancement: [train_val, test]

optional arguments:
  -h, --help            show this help message and exit
  --orig-dir ORIG_DIR   original image directory (default: False)
  --target-dir TARGET_DIR
                        target directory to store the AMSRCR enhanced images (default: False)

    Examples:

    Perform AMSRCR image enhancement on the aquatic test set:
        $ python run_amsrcr.py test
```

Figure 1: "run_amsrcr.py" script command prompt help message.

3 Training and Evaluating Models

In the present section, we instruct the user about the instance segmentation training, evaluation and prediction procedures. We analyse the required steps to run the necessary script hereunder.

Step 1: Enter an Anaconda Prompt terminal window in administrator mode;

Step 2: Navigate to the directory of the source code files;

Step 3: Activate the project conda environment created earlier (see Section 1);

Step 4: To train the reportedly best performing final CAM-RCNN model for 3 runs of 12 epochs, as presented in our project thesis, run:

```
python train_eval_model.py cam-rcnn
```

For evaluating a pretrained model and obtaining predictions on images, please see Figure 2 below. To display the message in the command prompt run:

```
python train_eval_model.py -h
```

```
usage: train_eval_model.py [-h] [--unseeded] [--epochs EPOCHS] [--loss LOSS_NAME] [--no-gn] [--no-aug] [--no-amsrcr]
                           [--lrd] [--mbbnms] [--run-name RUN_NAME] [--run-count COUNT] [--eval-only MODEL_PATHS]
                           [--eval-args TEST_SET TTA INF_FOLDER_NAME] [--predict IMAGE_PATHS MODEL_PATH]
                           [--pred-set PRED_SET]
                           MODEL_NAME

positional arguments:
  MODEL_NAME            model name from: [mrcnn, cam-rcnn, cmask, cinst, solov2]

optional arguments:
  -h, --help            show this help message and exit
  --unseeded            disable script seeding (default: False)
  --epochs EPOCHS       number of training epochs (default: 12)
  --loss LOSS_NAME      instance segmentation loss function to use (applies only for the CAM-RCNN model): [bce, dice,
                        dicebce, tversky, explog, logcosh] (default: dicebce)
  --no-gn               do not use Group Normalization layers (applies only for the CAM-RCNN model) (default: False)
  --no-aug              disable image resizing and flipping (augmentation) during training (default: False)
  --no-amsrcr           use original images during training and inference (default: False)
  --lrd                 use learning rate decay (default: False)
  --mbbnms              use matrix bounding box non-maximum suppression (default: False)
  --run-name RUN_NAME   name of the current run folder (where the pretrained model will be saved) (default: baseline)
  --run-count COUNT     how many times to train the model (default: 3)
  --eval-only MODEL_PATHS
                        evaluate pretrained models from the given paths (use regex for multiple folders) (default:
                        False)
  --eval-args TEST_SET TTA INF_FOLDER_NAME
                        booleans to enable/disable test set and TTA usages, and a name for the inference folder to
                        store the evaluation results (default: [True, False, 'test_inference'])
  --predict IMAGE_PATHS MODEL_PATH
                        make prediction on given test/validation set images using the provided pretrained weights
                        (default: False)
  --pred-set PRED_SET   test/validation set to use for predictions (default: test)

    Examples:

    Train best performing CAM-RCNN model (with DBL, AUG, AMSRCR and TTA)
    for 3 runs of 12 epochs (by default), as presented in our paper:
        $ python train_eval_model.py cam-rcnn

    Run evaluation on the test set without TTA using all baseline pretrained CAM-RCNN models.
    Results are saved to a folder titled "test_inference" by default:
        $ python train_eval_model.py cam-rcnn --eval-only output/cam-rcnn/*_baseline

    Run prediction on AMSRCR enhanced test set images with the best performing CAM-RCNN pretrained model:
        $ python train_eval_model.py cam-rcnn --predict data/test/amsrcr/*.jpg model_final.pth
```

Figure 2: "train_eval_model.py" script command prompt help message.

4 Inspecting Model Results

Here, we outline the steps necessary to plot model loss curves as well as to save the obtained metric results to a CSV file. They are summarised as follows:

Step 1: Enter an Anaconda Prompt terminal window in administrator mode;

Step 2: Navigate to the directory of the source code files;

Step 3: Activate the project conda environment created earlier (see Section 1);

Step 4: To plot training and validation loss curves of a model run:

```
python inspect_results.py --plot-losses <run_folders>
 ↪  <model_name>
```

where `<run_folders>` are the run output folders of the model. The user should use regex to define multiple model run folders, as shown in Figure 3. The

`<model_name>` is the name of the model used in the plot title. For saving the obtained model metric results, please see Figure 3 below. To display a command prompt help message regarding this script run:

```
python inspect_results.py -h
```

```
usage: inspect_results.py [-h] [--plot-losses RUN_FOLDERS MODEL_NAME]
                          [--save-txt RUN_FOLDERS METRICS_FOLDERS SAME_FW CSV_NAME]
                          [--save-json RUN_FOLDERS MODEL_NAME]

optional arguments:
  -h, --help            show this help message and exit
  --plot-losses RUN_FOLDERS MODEL_NAME
                        plot losses of the instance segmentation model (default: False)
  --save-txt RUN_FOLDERS METRICS_FOLDERS SAME_FW CSV_NAME
                        save model metric results from the outputted text file (default: False)
  --save-json RUN_FOLDERS MODEL_NAME
                        plot model metric results from the outputted JSON file (default: False)

    Examples:

    Plot baseline (default) CAM-RCNN model averaged training and validation losses.
    Please define the metrics folders using the regex "*" symbol to get all folders
    automatically. The command line command below utilises all folders ending with "_baseline":
        $ python inspect_results.py --plot-losses ./output/cam-rcnn/*_baseline CAM-RCNN

    Save baseline (default) CAM-RCNN model metrics results using the output text files.
    Please define the metrics folders using the regex "*" symbol to get all folders
    automatically. The command line command below utilises all output folders ending with "_baseline"
    and all metrics folders starting with "test_". Alternatively, one can specify only a single metric
    folder such as "test_inference":
        $ python inspect_results.py --save-txt ./output/cam-rcnn/*_baseline test_* True "test_folder_results"
```

Figure 3: "inspect_results.py" script command prompt help message.

5 Visualising Instance Mask Predictions

To visualise the predicted instance mask by a given model we go through the following steps:

Step 1: Enter an Anaconda Prompt terminal window in administrator mode;

Step 2: Navigate to the directory of the source code files;

Step 3: Activate the project conda environment created earlier (see Section 1);

Step 4: To visualise the JSON file predictions obtained during model evaluation run:

```
python visualize_json_results.py --input <json_file>
 ↪  --output <output_dir> --dataset <dataset_name>
 ↪  --conf-threshold <conf_threshold> --no-amsrcr
 ↪  <disable_amsrcr>
```

where: `<json_file>` is the path to the produce JSON file; `<output_dir>` is the output directory to save the mask prediction visualisation images; `<dataset_name>` is the name of the dataset; `<json_file>` is the confidence threshold to use for the predictions (defaults to 0.5); and `<disable_amsrcr>` is a boolean flag to decide if AMSRCR enhancement images are to be used for the predictions (defaults to

"False"). Please see Figure 4 below for more details. To display a command prompt help message regarding this script run:

```
python visualize_json_results.py -h
```

```
usage: visualize_json_results.py [-h] --input INPUT --output OUTPUT [--dataset DATASET] [--no-amsrcr]
                                 [--conf-threshold CONF_THRESHOLD]

A script that visualizes the json predictions from COCO or LVIS dataset.

optional arguments:
  -h, --help            show this help message and exit
  --input INPUT         JSON file produced by the model (default: None)
  --output OUTPUT       output directory (default: None)
  --dataset DATASET     name of the dataset (default: coco_2017_val)
  --no-amsrcr           disable AMSRCR image enhancement (default: False)
  --conf-threshold CONF_THRESHOLD
                        confidence threshold (default: 0.5)
```

Figure 4: "visualize_json_results.py" script command prompt help message.

6 Plotting Data Distribution Across Sets

In this section, we present the steps needed to successfully plot the data distributions of both aquatic images and animal instances regarding the training, validation and test sets. They are as follows:

Step 1: Enter an Anaconda Prompt terminal window in administrator mode;

Step 2: Navigate to the directory of the source code files;

Step 3: Activate the project conda environment created earlier (see Section 1);

Step 4: Plot the training, validation and test set instances per category and image distributions by running:

```
python plot_dist.py <train_val_paths> <test_paths>
```

where `<train_val_paths>` is the training and validation set JSON annotation file paths and `<test_paths>` is the test set ones. Please see Figure 5 below for more details. To display a command prompt help message regarding this script run:

```
python plot_dist.py -h
```

```
usage: plot_dist.py [-h] train_val_paths test_paths

positional arguments:
  train_val_paths  training and validation set JSON file paths
  test_paths       test set JSON file paths

optional arguments:
  -h, --help       show this help message and exit

    Examples:

    Plot the training, validation and test set instances per category and image distributions.
    Please use regex (i.e. "*" as below) to specify that all JSON files are to be used:
        $ python plot_dist.py data/train_val/json/*.json data/test/json/*.json

    Note that the plots will be displayed one by one!
```

Figure 5: "plot_dist.py" script command prompt help message.

7 Creating a Demo Video

In this part we go through the steps for creating a demo video (only the CAM-RCNN model is supported):

Step 1: Enter an Anaconda Prompt terminal window in administrator mode;

Step 2: Navigate to the directory of the source code files;

Step 3: Activate the project conda environment created earlier (see Section 1);

Step 4: Create a directory for the output files (optional);

Step 5: Create a demo inference video file by running:

```
python demo.py --config-file <config_file> --video-input
↪ <input_file> --output <output_dir>
↪ --confidence-threshold <threshold> --opts MODEL.WEIGHTS
↪ <weights_path>
```

where: `<config_file>` is the configuration file of the model (use "demo/cam-rcnn.yaml" since currently CAM-RCNN is the only supported model); `<input_file>` is the selected input video recording (ours are located inside "demo/input_videos/"); `<output_dir>` is the directory to hold the demo inference videos (has to be created beforehand or simply use the current one); `<threshold>` is the minimum score for the instance predictions to be shown (defaults to 0.5); and `<weights_path>` is the path to the model weights file. Please see Figure 6 below for more details. To display a command prompt help message regarding this script run:

```
python demo.py -h
```

```
usage: demo.py [-h] [--config-file FILE] [--webcam] [--video-input VIDEO_INPUT] [--input INPUT [INPUT ...]]
               [--output OUTPUT] [--confidence-threshold CONFIDENCE_THRESHOLD] [--opts ...]

Detectron2 demo for builtin configs

options:
  -h, --help            show this help message and exit
  --config-file FILE    path to config file (default:
                        configs/quick_schedules/mask_rcnn_R_50_FPN_inference_acc_test.yaml)
  --webcam              Take inputs from webcam. (default: False)
  --video-input VIDEO_INPUT
                        Path to video file. (default: None)
  --input INPUT [INPUT ...]
                        A list of space separated input images; or a single glob pattern such as 'directory/*.jpg'
                        (default: None)
  --output OUTPUT       A file or directory to save output visualizations. If not given, will show output in an OpenCV
                        window. (default: None)
  --confidence-threshold CONFIDENCE_THRESHOLD
                        Minimum score for instance predictions to be shown (default: 0.5)
  --opts ...            Modify config options using the command-line 'KEY VALUE' pairs (default: [])
```

Figure 6: "demo.py" script command prompt help message.