Contents

List of Tables

List of Figures

Maintenance Manual

1 Installation

The source code files for the software installation are available at our GitHub repository. Nevertheless, the datasets are not publicly available and can only be accessed upon approval of request since they are a property of the University of Aberdeen thus strictly confidential.

2 Temporary Files

Possible temporary files are the TensorBoard logs, i.e. individual metric plots, created at the end of each epoch. We do not utilise these logs when evaluating and presenting our results. This is because we develop our own utility functions so that we can freely tailor the result plots to our need and taste. Hence, the user should decide whether to keep or remove these non-essential logs.

2.1 Data Directory Structure

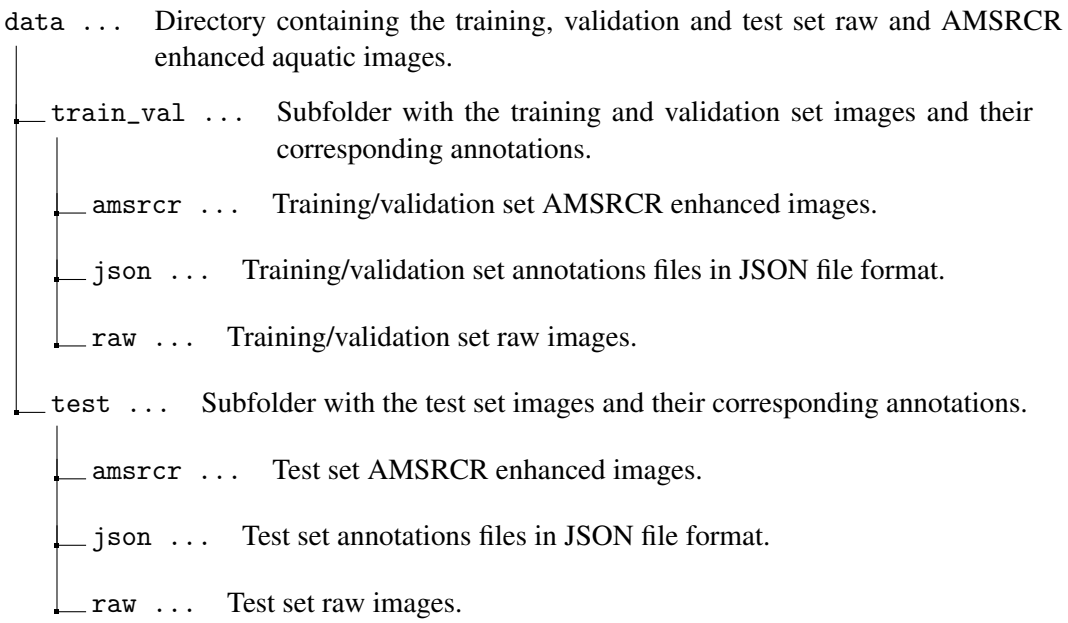

Figure 1: Project data directory tree. Training and validation sets adopt Sparkling Star deployment images while the test set utilises Shetland deployment image samples.

3 Software Dependencies

All required program script packages and dependencies are set forth in Table 1 hereunder:

Package	Version
cudatoolkit	11.3.1
opencv	4.5.5
pandas	1.4.1
pefile	2021.9.3
python	3.9.10
pip	latest
pytorch	1.11.0
scikit-image	0.19.2
scikit-learn	1.0.2
torchaudio	0.11.0
torchvision	0.12.0

(a) Conda

Package	Version
cloudpickle	2.1.0
fvcore	0.1.5.post20220512
matplotlib	3.5.2
omegaconf	2.2.1
pycocotools	2.0.4
PyYAML	6.0
tensorboard	2.8.0

(b) Pip

Table 1: Python main dependencies of the project source code scripts. These should automatically install all other secondary dependencies.

4 Requirements

4.1 Disk Space

The project program files are around 45 MB in size (without the pretrained models), which every contemporary computer system hardware can handle comfortably. On the other hand, the bulk of both the Sparkling Star and Shetland deployment sample datasets would require a minimum computer disk space of around 1 GB in total. This includes the raw together with the AMSRCR enhanced variants of the images of the datasets. Also, each of the pretrained model files can take up to 500 MB of additional storage space. Therefore, we do not include COCO pretrained weights of the 4 baseline models due to the large size of these files. They can be obtained separately at Detectron2, CenterMask2 and AdelaiDet. We provide only the finalised CAM-RCNN model weights file, which is available at our GitHub repository.

4.2 Memory (RAM)

Processing of large datasets and model architectures often involve holding a substantial amount of data in memory. Therefore, it is recommended that the program scripts are run on systems with at least 6 GB of available.

5 System Files and Folders

In this section we explore the project directories by describing their content and summarising the role of the present source code files. Firstly, in the following directory tree (Figure 2), we portray the contents of the source code directory and its sub-directories.

aquatic_is

- `AdelaiDet ...` Toolbox for multiple instance-level recognition tasks on top of Detectron2 [1].
- `centermask ...` CenterMask model implementation library on top of Detectron2 [2].
- `coco_pretrained ...` Directory that holds the COCO pretrained models for the transfer learning.
- `configs...` Folder that contains the baseline configuration files of the models.
- `data ...` Refer to Figure 1.
- `output ...` Directory comprising training output files, i.e. metrics, model weights, etc.
- `preprocessing ...` Directory containing the AMSRCR image enhancement algorithm.
- `pretrained ...` Directory containing the finalised CAM-RCNN weights (GitHub).
- `constants.py ...` Script containing important constants.
- `custom_trainer.py ...` Script that implements the CustomTrainer class.
- `demo.py ...` Script for creating demo inference videos.
- `fixNvPe.py ...` Script to disable Address Space Layout Randomization (ASLR).
- `inspect_results.py ...` Script used for inspecting loss results along with validation and test set output metrics of a model.
- `instance_segmentor.py ...` Script that implements the InstanceSegmentor class used for initialising and training/evaluating instance segmentation models.
- `loss_eval_hook.py ...` Script that implements the validation loss evaluation hook for Detectron2.
- `optional_helpers.py ...` Script with additional auxiliary non-essential functions.
- `plot_dist.py ...` Script plotting the data distributions across the training, validation and test sets.
- `predictor.py ...` Script that implements the VisualizationDemo class used in visualising and creating inference demo videos..
- `run_amsrcr.py ...` Script that runs the AMSRCR image enhancement technique on a given image dataset.
- `train_eval_model.py ...` Script used for training an instance segmentation model.
- `utils.py ...` Script that implements various utility functions.
- `visualize_json_results.py ...` Script used for visualising model JSON IS results.

Figure 2: Project source code files and folders.

Bibliography

[1] Adelaide Intelligent Machines (AIM) Group. Adelaidet. https://github.com/aim-uofa/AdelaiDet. Accessed: 02 May, 2022.

[2] Youngwan Lee. Centermask: Real-time anchor-free instance segmentation, in cvpr 2020. https://github.com/youngwanLEE/centermask2. Accessed: 02 May, 2022.