OL_mrcnn

Release 1.0.0

Tom Soulaire

Jun 03, 2024

CONTENTS

This tool is a model based on a MRCNN architecture that enables to

- detect and crop cells in an image (grayscale, RGB, or more channels)
- classify cells in an image
- segment cells in an image

CHAPTER

ONE

INSTALLATION

To install the model on a local machine (requiring Python 3.9):

1. create conda environment:

```
conda env create -f environment.yml
```

2. activate conda environment:

```
conda activate OL_mrcnn
```

3. download the model weights here and place them in the folder `/logs`. These contain the original weights (trained on COCO dataset) and the weights trained on a custom dataset.

CHAPTER

TWO

USER GUIDE

2.1 Image cropping

- Add your dataset in the folder `/data`
- OPTIONAL: preprocess your data with the `preprocessing.ipynb` notebook
- Configure the `image_cropper.ipynb` notebook:
  - `DEVICE`: device to use for inference. Default value is 'cpu:0'.
  - `detection_min_confidence`: minimum confidence level for the detections. Default value is 0.7.
  - `detection_nms_threshold`: non-maximum suppression threshold. Eliminates the least confident detection when the IoU of 2 masks is above this value. Default value is 0.3.
  - `weights_subpath`: subpath in the */logs* folder to the weights file.
  - `results_name`: name of the folder where the results will be saved.
  - `test_dir`: name of the folder where the images are stored.
  - `num_gpu`: number of GPUs to use for inference. Default value is 1.
  - `num_img_per_gpu`: number of images to process in parallel on each GPU. Default value is 1.
  - `VISUALIZE`: if True, displays the images with the detections. Default value is False.
- Run the notebook. The results will be saved in the folder `/results/results_name`.

2.2 Full pipeline

- In this setup, we run a first model to crop and classify objects in the images. Then we run a second model on the cropped images to get a refined mask.
- In the `model_pipeline.ipynb` notebook, configure the following parameters:
  - `DEVICE`: device to use for inference. Default value is 'cpu:0'.
  - `gpu_count_macro`: number of GPUs to use for the first model. Default value is 1.
  - `num_img_per_gpu_macro`: number of images to process in parallel on each GPU for the first model. Default value is 1.
  - `min_confidence_macro`: minimum confidence level for the detections in the first model. Default value is 0.7.
  - `nms_threshold_macro`: non-maximum suppression threshold for the first model. Default value is 0.3.

  - `nms_multiclass_macro`: non-maximum suppression threshold between classes for the first model. Default value is 0.3.
  - `gpu_count_micro`: number of GPUs to use for the second model. Default value is 1.
  - `num_img_per_gpu_micro`: number of images to process in parallel on each GPU for the second model. Default value is 1.
  - `min_confidence_micro`: minimum confidence level for the detections in the second model. Default value is 0.7.
  - `nms_threshold_micro`: non-maximum suppression threshold for the second model. Default value is 0.3.
  - `MACRO_MODEL_SUBPATH`: subpath in the */logs* folder to the weights file of the first model.
  - `MICRO_MODEL_SUBPATH`: subpath in the */logs* folder to the weights file of the second model.
  - `RESULTS_NAME`: name of the folder where the results will be saved.
  - `TEST_DIR`: name of the folder where the images are stored.
  - `VISUALIZE`: if True, displays the images with the detections. Default value is False.

- Run the notebook. The results will be saved in the folder `/results/RESULTS_NAME`.

2.3 Retraining your own model

2.3.1 Data structure

- Create a `/data` folder in the root directory.
- Inside the `/data` directory, put your images in a folder named `/imgs` and your binary masks in a folder named `/masks`. The name, size and format of the masks must match the images.
- In the ``roi_labels_to_json.py``script, configure the ``dir_path``in the *main()* function. Run in a terminal:

```
python roi_labels_to_json.py
```

- Move the label files to a `jsons` folder in the ``/data``directory.
- In the `format_data.py` script, configure the `dir_path` in the *main()* function. Configure the size the of the training / validation / test datasets (usually 0.6, 0.2, 0.2) Run in a terminal:

```
python format_data.py
```

2.3.2 Retraining a single class model

- **In the `custom.py` script, configure the following:**
  - `GRAYSCALE`: if True, the model will be trained on grayscale images. Default value is False.
  - `DATA_PATH`: path to the dataset. Default value is '/data'.
  - `NAME`: name of the model.
  - `GPU_COUNT`: number of GPUs to use. Default value is 1.
  - `IMAGES_PER_GPU`: number of images to process in parallel on each GPU. Default value is 1.
  - `NUM_CLASSES`: number of classes. Default value is 2.

  - `EPOCHS`: number of epochs. Default value is 50.
  - `STEPS PER EPOCH`: number of steps per epoch. Default value is 50.
  - `LEARNING_RATE`: learning rate. Default value is 0.001.
  - `LAYERS`: layers to train. Default value is 'heads'.
  - `DETECTION_MIN_CONFIDENCE`: minimum confidence level for the detections. Default value is 0.7.
  - `DEVICE`: device to use for training. Default value is 'cpu:0'.
  - `MAX_GT_INSTANCES`: maximum number of instances in the ground truth. Default value is 100.
  - `DETECTION_MAX_INSTANCES`: maximum number of instances in the detections. Default value is 35.
  - in the ``CustomDataset``class, modify or add lines :

- Run the script in a terminal:

```
python custom.py
```

2.3.3 Retraining a multi-class model

- Same instructions as before but on the `custom_multi.py` script.
- Run the script in a terminal:

```
python custom_multi.py
```

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search