cellseg

Release 0.0.0

Nelson Gonzabato

Mar 01, 2021

CONTENTS

1 cellseg **1**
- 1.1 data module . . . 1
- 1.2 utils module . . . 1

2 cellseg: Multiclass Cell Segmentation **3**

3 cellseg: Multiclass Cell Segmentation **5**

4 Contributing to cellseg **7**

5 Indices and tables **9**

Python Module Index **11**

Index **13**

CHAPTER

ONE

CELLSEG

1.1 data module

class data.**DataProcessor**(*image_dir*, *dir_type='image'*, *target_size=(512, 512)*, *image_suffix='tif'*)

Bases: Generic[torch.utils.data.dataset.T_co]

to_uint8(*image*)

transform(*image*)

Parameters

- **image** – Stacked image eg (71, 1200, 1200)
- **mask** – Stacked mask eg (71, 1200, 1200)

Returns Transformed images that have been randomly cropped, flipped, and resized to the target size.

data.**convertScaleAbs**(*src*[, *dst*[, *alpha*[, *beta*]]]) → dst

. @brief Scales, calculates absolute values, and converts the result to 8-bit. . . On each element of the input array, the function convertScaleAbs . performs three operations sequentially: scaling, taking an absolute . value, conversion to an unsigned 8-bit type: . f[texttt{dst} (I)= texttt{saturate_cast<uchar>} (| texttt{src} (I)* texttt{alpha} + texttt{beta} |)f] . In case of multi-channel arrays, the function processes each channel . independently. When the output is not 8-bit, the operation can be . emulated by calling the Mat::convertTo method (or by using matrix . expressions) and then by calculating an absolute value of the result. . For example: . @code{.cpp} . Mat_<float> A(30,30); . randu(A, Scalar(-100), Scalar(100)); . Mat_<float> B = A*5 + 3; . B = abs(B); . // Mat_<float> B = abs(A*5+3) will also do the job, . // but it will allocate a temporary matrix . @endcode . @param src input array. . @param dst output array. . @param alpha optional scale factor. . @param beta optional delta added to the scaled values. . @sa Mat::convertTo, cv::abs(const Mat&)

1.2 utils module

utils.**get_thresholds**(*image*, *method='li'*)

Parameters

- **image** – A stacked image of class DataProcessor
- **method** – One of li, watershed or multiotsu

Returns Thresholded methods based on the method.

utils.**show_images**(*dataset_object*, *stack_number=0*, *number=None*, *fig_size=(20, 20)*, *target='image'*)

Parameters

- **dataset_object** – An object of class DataLoader
- **stack_number** – Frame number for tiff images. Defaults to zero.
- **number** – Number of images to plot from the frame
- **target** – Type of images to show. One of "image" or "mask"
- **fig_size** – Figure size, defaults to (20, 20)

Returns A plot showing images from the stack number chosen.

utils.**show_thresholded**(*original*, *otsu*, *li*, *number=28*)

Parameters

- **original** – Original Image of class DataProcessor
- **otsu** – Otsu/watershed/li thresholded image
- **li** – Otsu/watershed/li thresholded image
- **number** – Frame number to view

Returns Three images side by side

CHAPTER

TWO

CELLSEG: MULTICLASS CELL SEGMENTATION

Introduction

`cellseg` is a PyTorch (`torch`) based deep learning package aimed at multiclass cell segmentation.

Installation

```
pip install cellseg
```

Or if you want to build from source

```
git clone git@github.com:Nelson-Gon/cellseg.git
cd cellseg
python setup.py install
```

Development stage

- [x] Read Tiff Images
- [x] Read Non Tiff Images
- [x] Write Data Transformers and Loaders
- [ ] Write functional model
- [ ] Modify model weights/layers

Usage

```
# imports data, utils, model
from cellseg import *
```

CHAPTER

THREE

CELLSEG: MULTICLASS CELL SEGMENTATION

version 0.0.0

- Preserve name on PyPI
- Fixed issues with `show_images` showing blank images for masks (labels).
- Fixed issues with `uint16` not working with `PIL`.
- `DataProcessor` can now transform images to a given target size.
- Renamed `DataLoader` class to `DataProcessor` to avoid conflicts with `torch.utils.data.DataLoader`
- Added initial simple CNN model with a single layer
- Added `show_images` in `utils.py` to allow quick visualization of a given number of images from a given stack of images.
- Implemented data loaders.
- Conceptualized project

CHAPTER

FOUR

CONTRIBUTING TO CELLSEG

This document provides guidelines for contributions to `cellseg`.

Kinds of contribution

- Typo fixes
- Documentation enhancements
- Pull requests

Fixing typos and enhancing documentation

To fix typos and/or grammatical errors, please edit the corresponding `.py` or `.md` file that generates the documentation.

Please also update the docs using `sphinx`

Pull Requests

- Please raise an issue for discussion and reproducibility checks at issues
- Once the bug/enhancement is approved, please create a Git branch for the pull request.
- Make changes and ensure that builds are passing the necessary checks on Travis.
- Update `changelog.md` to reflect the changes made.
- Do the following:

```
bash scripts/mkdocs.sh #projectnamehere
```

- Releasing

```
bash scripts/release.sh
```

The above does the following:

- Makes `dist` with `python setup.py sdist` at the very minimum. Ensure everything necessary is included in `Manifest.in`.
- Uploads `dist` to test.pypi.org with `twine upload --repository-url https://test.pypi.org/legacy/ dist/*`
- If everything looks good, asks you to upload to pypi.org with `twine upload dist/*`

Please note that the 'cellseg' project is released with a Contributor Code of Conduct. By contributing to this project, you agree to abide by its terms.

See also for a guide on Sphinx documentation.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

data, 1

u

utils, 1

INDEX

C

convertScaleAbs() (*in module data*), 1

D

data
    module, 1
DataProcessor (*class in data*), 1

G

get_thresholds() (*in module utils*), 1

M

module
    data, 1
    utils, 1

S

show_images() (*in module utils*), 1
show_thresholded() (*in module utils*), 2

T

to_uint8() (*data.DataProcessor method*), 1
transform() (*data.DataProcessor method*), 1

U

utils
    module, 1