Medical Images Dataloader Documentation

Release 0.1.12

Matteo Rossi

Nov 22, 2021

CONTENTS:

1 Medical Images Dataloader **1**
- 1.1 Features . . . 1
- 1.2 Credits . . . 1

2 Installation **3**
- 2.1 Stable release . . . 3
- 2.2 From sources . . . 3

3 Usage **5**

4 med_dataloader **7**
- 4.1 med_dataloader package . . . 7

5 Contributing **11**
- 5.1 Types of Contributions . . . 11
- 5.2 Get Started! . . . 12
- 5.3 Pull Request Guidelines . . . 13
- 5.4 Tips . . . 13
- 5.5 Deploying . . . 13

6 Credits **15**
- 6.1 Development Lead . . . 15
- 6.2 Contributors . . . 15

7 History **17**
- 7.1 0.1.12 (2021-10-29) . . . 17
- 7.2 0.1.11 (2021-09-10) . . . 17
- 7.3 0.1.10 (2021-05-11) . . . 17
- 7.4 0.1.9 (2021-05-11) . . . 17
- 7.5 0.1.8 (2021-05-09) . . . 17
- 7.6 0.1.6 (2021-05-06) . . . 18
- 7.7 0.1.5 (2021-04-30) . . . 18
- 7.8 0.1.4 (2021-04-29) . . . 18
- 7.9 0.1.1 (2021-04-20) . . . 18
- 7.10 0.1.0 (2021-04-16) . . . 18

8 Indices and tables **19**

Python Module Index **21**

Index **23**

CHAPTER

ONE

MEDICAL IMAGES DATALOADER

A general-purpose Dataloader for Tensorflow 2.x. It supports many medical image formats.

- Free software: MIT license
- Documentation: https://med-dataloader.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install Medical Images Dataloader, run this command in your terminal:

```
$ pip install med_dataloader
```

This is the preferred method to install Medical Images Dataloader, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Medical Images Dataloader can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/mrossi93/med_dataloader
```

Or download the tarball:

```
$ curl -OJL https://github.com/mrossi93/med_dataloader/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use Medical Images Dataloader in a project:

```
import med_dataloader
```

CHAPTER

FOUR

MED_DATALOADER

4.1 med_dataloader package

4.1.1 Submodules

4.1.2 med_dataloader.cli module

Console script for med_dataloader.

4.1.3 med_dataloader.med_dataloader module

Main module.

class med_dataloader.med_dataloader.**DataLoader**(*mode*, *imgA_label=None*, *imgB_label=None*, *input_size=None*, *data_dir='./Data'*, *output_dir=None*, *is_B_categorical=False*, *num_classes=None*, *norm_boundsA=None*, *norm_boundsB=None*, *extract_only=None*, *use_3D=False*)

Bases: object

[summary]

__init__(*mode*, *imgA_label=None*, *imgB_label=None*, *input_size=None*, *data_dir='./Data'*, *output_dir=None*, *is_B_categorical=False*, *num_classes=None*, *norm_boundsA=None*, *norm_boundsB=None*, *extract_only=None*, *use_3D=False*)

[summary]

Parameters

- **mode** (*[type]*) – [description]
- **imgA_label** (*str*) – Identifier for class A. It's the name of the folder inside data_dir that contains images labeled as class A.
- **imgB_label** (*str*) – Identifier for class B. It's the name of the folder inside data_dir that contains images labeled as class B.
- **input_size** (*int*) – Dimension of a single image, defined as input_size x input_size. Currently, it supports only squared images.
- **data_dir** (*str, optional*) – Path to directory that contains the Dataset. This folder **must** contain two subfolders named like imgA_label and imgB_label. Defaults to './Data'.
- **output_dir** (*[type], optional*) – [description]. Defaults to None.

- **is_B_categorical** (*bool, optional*) – [description]. Defaults to False.
- **num_classes** (*[type], optional*) – [description]. Defaults to None.
- **norm_boundsA** (*[type], optional*) – [description]. Defaults to None.
- **norm_boundsB** (*[type], optional*) – [description]. Defaults to None.
- **extract_only** (*int, optional*) – Indicate wheter to partially cache a certain amount of elements in the dataset. Please remember that if `output_dir` folder is already populated, you need to clean this folder content to recreate a partial cache file. When it is set to None, the entire Dataset is cached. Defaults to None.
- **use_3D** – Indicate whether to use three-dimensional data in the cache (if True) or to extract two-dimensional slices from the 3D volumes (if False). Defaults to False.

Raises

- **ValueError** – [description]
- **FileNotFoundError** – [description]
- **ValueError** – [description]
- **FileNotFoundError** – [description]
- **FileNotFoundError** – [description]
- **ValueError** – [description]
- **ValueError** – [description]
- **ValueError** – [description]
- **FileNotFoundError** – [description]

static check_type(*path*)

fix_image_dims(*img*, *size*)

Fix tensor dimensions so that they are of the proper size to carry out Tensorflow operations.

This function performs three steps:

1. Squeeze to remove axis with dimension of 1
2. Expand the dimensions of the tensor by adding one axis
3. Resize and pad the tensor to a target width and height

If *use_3D* was enabled, volume is not resized and padded.

Parameters

- **img** – image or volume to be processed
- **size** – desired size of image or volume in the two/three axis.

get_dataset(*batch_size=32*, *augmentation=False*, *random_crop_size=None*, *random_rotate=False*, *random_flip=False*)

get_imgs(*img_paths*, *img_label*, *img_type*, *is_RGB*, *is_categorical=False*, *num_classes=None*, *norm_bounds=None*)

Open image files for one class and store it inside cache.

This function performs all the (usually) slow reading operations that is necessary to execute at least the first time. After the first execution information are saved inside some cache file inside Cache folder (typically created in your Dataset folder, at the same level of Images folder). This function detects if cache files are already present, and in that case it skips the definition of these files. Please take into account that cache

files will be as big as your Dataset overall size. First execution may result in a considerably bigger amount of time.

Parameters **img_paths** (*str*) – Path to single class images.

Returns

Tensorflow dataset object containing images of one classes converted in Tensor format, without any other computations.

Return type tf.Data.Dataset

get_imgs_paths()

Get paths of every single image divided by classes.

Returns

two list containing the paths of every images for both classes. The list is sorted alphabetically, this can be usefull when images are named with a progressive number inside a folder (e.g.: 001.xxx, 002.xxx, . . . , 999.xxx)

Return type list, list

static is_3D_data(*path*)

static is_RGB_data(*path*)

static norm_with_bounds(*image*, *bounds*)

Image normalisation. Normalises image in the range defined by lb and ub to fit[0, 1] range.

open_img(*path*)

Open an image file and convert it to a tensor.

Parameters **path** (*tf.Tensor*) – Tensor containing the path to the file to be opened.

Returns Tensor containing the actual image content.

Return type tf.Tensor

static random_crop(*imgA*, *imgB*, *crop_size=256*)

static random_flip(*imgA*, *imgB*)

static random_rotate(*imgA*, *imgB*)

med_dataloader.med_dataloader.**generate_dataset**(*data_dir*, *imgA_label*, *imgB_label*, *input_size*, *output_dir=None*, *extract_only=None*, *norm_boundsA=None*, *norm_boundsB=None*, *is_B_categorical=False*, *num_classes=None*, *use_3D=False*)

med_dataloader.med_dataloader.**get_dataset**(*data_dir*, *percentages*, *batch_size*, *train_augmentation=True*, *random_crop_size=None*, *random_rotate=True*, *random_flip=True*)

4.1.4 Module contents

Top-level package for Medical Images Dataloader.

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/mrossi93/med_dataloader/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Medical Images Dataloader could always use more documentation, whether as part of the official Medical Images Dataloader docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/mrossi93/med_dataloader/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *med_dataloader* for local development.

1. Fork the *med_dataloader* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/med_dataloader.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv med_dataloader
$ cd med_dataloader/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 med_dataloader tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.7, 3.8, and for PyPy. Check https://travis-ci.com/mrossi93/med_dataloader/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_med_dataloader
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Matteo Rossi <rossimatteo1993@gmail.com>

6.2 Contributors

- Davide Marzorati <davide.marzorati.93@gmail.com>

CHAPTER

SEVEN

HISTORY

7.1 0.1.12 (2021-10-29)

- Fixed minor bug in function norm_with_bounds

7.2 0.1.11 (2021-09-10)

- Fixed support for 3D Images
- Fixed minor bugs

7.3 0.1.10 (2021-05-11)

- Added support one-hot encoding in case of multi-class label

7.4 0.1.9 (2021-05-11)

- Added support for RGB Images
- Fixed some bugs related to norm_bounds types

7.5 0.1.8 (2021-05-09)

- Main Changes in the package structure. Now there are two main functions: generate_dataset and get_dataset, both leveraging on DataLoader class.
- The generation of the dataset can be handled also by CLI, to simplify usage.
- Processed data can live by themself. No more need to transfer also original file (e.g. to Drive to make use of them on Colab)

7.6 0.1.6 (2021-05-06)

- Improved flexibility for image data types. Now cache dimension reflects the actual dataset dimension.

7.7 0.1.5 (2021-04-30)

- Added support for 3D files: now Dataloader automatically detects whether a file is 2D or 3D and returns the properly sized dataset. Please remember that med_dataloader returns tf.data.Dataset object for 2D tasks, 3D is not yet supported.
- Added new notebook in examples folder.

7.8 0.1.4 (2021-04-29)

- **Improved code flexibility:**
  - It is possibile to choose which type of data augmentation is performed
  - Boundaries for data normalization can be set by the user
  - Images can be resized automatically by the user
- Added basic_usage example also as a notebook

7.9 0.1.1 (2021-04-20)

- Added code for package
- Basic example of usage inside folder “examples”
- Partial documentation

7.10 0.1.0 (2021-04-16)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

med_dataloader, 10
med_dataloader.cli, 7
med_dataloader.med_dataloader, 7

INDEX

Symbols

__init__() (*med_dataloader.med_dataloader.DataLoader method*), 7

C

check_type() (*med_dataloader.med_dataloader.DataLoader static method*), 8

D

DataLoader (*class in med_dataloader.med_dataloader*), 7

F

fix_image_dims() (*med_dataloader.med_dataloader.DataLoader method*), 8

G

generate_dataset() (*in module med_dataloader.med_dataloader*), 9
get_dataset() (*in module med_dataloader.med_dataloader*), 9
get_dataset() (*med_dataloader.med_dataloader.DataLoader method*), 8
get_imgs() (*med_dataloader.med_dataloader.DataLoader method*), 8
get_imgs_paths() (*med_dataloader.med_dataloader.DataLoader method*), 9

I

is_3D_data() (*med_dataloader.med_dataloader.DataLoader static method*), 9
is_RGB_data() (*med_dataloader.med_dataloader.DataLoader static method*), 9

M

med_dataloader
    module, 10
med_dataloader.cli
    module, 7
med_dataloader.med_dataloader
    module, 7
module
    med_dataloader, 10
    med_dataloader.cli, 7
    med_dataloader.med_dataloader, 7

N

norm_with_bounds() (*med_dataloader.med_dataloader.DataLoader static method*), 9

O

open_img() (*med_dataloader.med_dataloader.DataLoader method*), 9

R

random_crop() (*med_dataloader.med_dataloader.DataLoader static method*), 9
random_flip() (*med_dataloader.med_dataloader.DataLoader static method*), 9
random_rotate() (*med_dataloader.med_dataloader.DataLoader static method*), 9