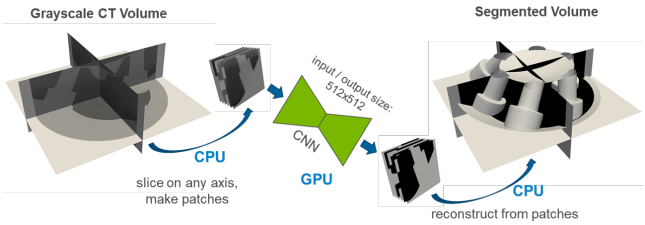
CTSegNet Documentation

Release 0.1

Argonne National Laboratory

Apr 13, 2020

Contents



CTSegNet is a package for end-to-end 3D segmentation workflow for large X-ray tomographic datasets using 2D fully convolutional neural networks (fCNN).

CHAPTER 1

Features

- List here
- the module features

CHAPTER 2

Contribute

- Documentation: https://github.com/aniketkt/CTSegNet/tree/master/doc
- Issue Tracker: https://github.com/aniketkt/CTSegNet/docs/issues
- Source Code: https://github.com/aniketkt/CTSegNet/

CHAPTER 3

Content

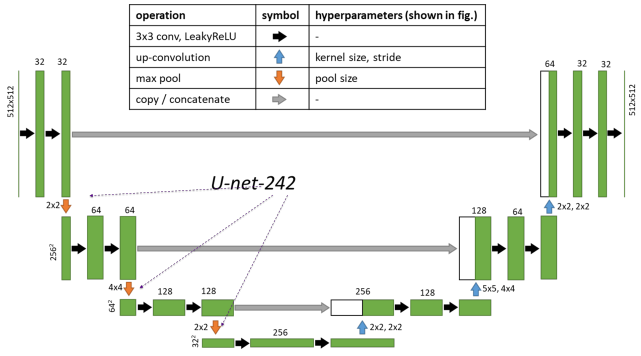
3.1 About

3.1.1 The Algorithm

fCNN architecture

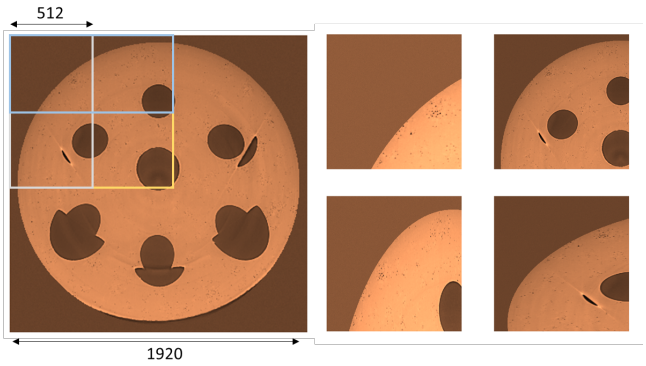
CTSegNet deploys unique Unet-like models trained with focal loss to provide accuracy with reduced number of convolutional layers. The methodology and performance metrics are discussed in [A1].

Here is a sample architecture that you can build using the model_utils sub-module in CTSegNet. We will refer to it as Unet-242 because of the 2-4-2 implementation of pooling layers.

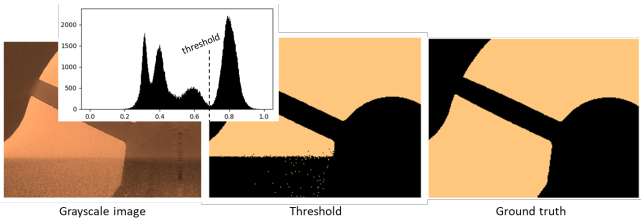


What is unique about CTSegNet?

While Unet-based segmentation is now commonplace, it is primarily limited to 2D data since 3D convolutional layers require prohibitively large GPU memory during training. Our approach efficiently exploits 2D fCNNs for 3D segmentation. You can generate multiple 3D masks by slicing along any axis, and choose a patching strategy based on the resolution-to-context trade-off in your CT data. For an fCNN with input/output images sized 512:sup:*2*, you can make patches in several ways. This a slice drawn from a scan of a gasoline injector along the transverse plane.



An ensemble vote from several 3D segmentations maps yields near voxel accuracy in many cases, where thresholding just won't work. Here's an example of a band-like artifact from restricted field-of-view in a CT scan (sagittal plane is shown).



The data_utils.data_io module contains the DataFile class, which enables fast and memory-efficient slicing using hdf5 format so you can visualize and segment 100GB+ datasets from your workstation. With this, you can segment only parts of your data or test models on slices of your data, with a few lines of code. Tiff format is also supported but with limited functionality.

Tell me more

Read our paper at [A1] or contact me at: atekawade [at] anl [dot] gov

3.2 Install

3.2.1 Installation from Source

To download the entire package with executables, sample model files and config files, clone the master branch and install locally. To download the .h5 model files when cloning, you will need Git LFS installed.

```
git clone https://github.com/aniketkt/CTSegNet.git
pip install CTSegNet/.
```

ct_segnet only

To install only ct_segnet modules into your python 3 environment, use pip. For compatibility with tensorflow 1.14, please install ct_segnet 1.16 from the tf-1 branch.

```
$ pip install git+https://github.com/aniketkt/CTSegNet.git@master#egg=ct_segnet
```

in a prepared virtualenv or as root for system-wide installation.

3.2.2 Command-line interface

CTSegNet also provides a command-line interface with config files. While executable scripts are provided, it's easy to write your own too. Data formats supported are .tiff sequence and hdf5. Example config files are provided in cfg_files/.
TRAIN/TEST: Extract training data from arbitrarily sized CT data and ground-truth pairs:

```
python bin/make_training_dataset.py -c cfg_files/setup_train.cfg
```

Build and train several Unet-like fCNN architectures for an input image size of your choice:

```
python bin/train_fCNN.py -t cfg_files/train.cfg -m cfg_files/models/Unet242.cfg
```

SEGMENT: An end-to-end 3D segmentation workflow that binarizes 2D images extracted from 3D CT data using the fCNN model, then rebuilds the corresponding 3D segmentation map. The hdf5 version is optimized for low RAM usage in very large (>50 GB) datasets.:

```
python bin/run_segmenter.py -c cfg_files/setup_seg.cfg
```

USE HDF5 FORMAT: Re-package your CT data into hdf5 format, with methods to determine optimal chunk size. Although optional, using hdf5 format accelerates read/write time while slicing through your datasets. Set -c as chunk size in MB or chunk shape z,y,x.:

```
python bin/rw_utils/convert_to_hdf5.py -f my_tiff_folder -o output_file.hdf5 -c 20.0
```

3.3 Usage

3.3.1 to be completed

add here

3.4 API reference

CTSegNet Modules:

3.4.1 ct_segnet.seg_utils

Created on Sat Nov 16 17:13:22 2019

@author: atekawade

CTSegNet is more than a 2D CNN model - it's a 3D Segmenter that uses 2D CNNs. The set_utils.py defines the Segmenter class that wraps over a keras U-net-like model (defined by models.py), integrating 3D slicing and 2D patching functions to enable the 3D-2D-3D conversations in the segmentation workflow. To slice a 3D volume, we manipulations such as 45 deg rotations, orthogonal slicing, patch extraction and stitching.

Functions:

read_tomo

class ct_segnet.seg_utils.**Segmenter**(*model_filename=None*, *model=None*, *model_name='unknown'*)

Bases: object

The Segmenter class wraps over a keras model, integrating 3D slicing and 2D patching functions to enable the 3D-2D-3D conversations in the segmentation workflow.

seg_chunk(*p*, *max_patches=None*, *overlap=None*, *nprocs=None*, *arr_split=1*)

Segment a volume of shape (nslices, ny, nx). The 2D keras model passes along nslices, segmenting images (ny, nx) with a patch size defined by input to the model max_patches : tuple, (my, mx) are # of patches along Y, X in image (ny, nx) overlap : tuple or int, number of overlapping pixels between patches nprocs : number of CPU processors for multiprocessing Pool arr_split : breakdown chunk into arr_split number of smaller chunks

seg_image(*p*, *max_patches=None*, *overlap=None*)

function to test the segmenter on arbitrary sized 2D image; extracts patches shape = input shape of 2D CNN max_patches : tuple, (my, mx) are # of patches along Y, X in image p : greyscale image of shape (ny, nx) overlap : tuple or int, number of overlapping pixels between patches

ct_segnet.seg_utils.**get_repadding**(*crops*, *d_shape*)

Returns padding values to restore 3D np array after it was cropped. crops : 3 tuples in a list [(nz1,nz2), (ny1,ny2), (nx1,nx2)] d_shape : original shape of 3D array

ct_segnet.seg_utils.**message**(*_str*)

ct_segnet.seg_utils.**process_data**(*p*, *segmenter*, *preprocess_func=None*, *max_patches=None*, *overlap=None*, *nprocs=None*, *rot_angle=0.0*, *slice_axis=0*, *crops=None*, *arr_split=1*)

Segment a volume of shape (nz, ny, nx). The 2D keras model passes along either axis (0,1,2), segmenting images with a patch size defined by input to the model in the segmenter class. max_patches : tuple, (?,?) number of patches along each axis of 2D image overlap : tuple or int, number of overlapping pixels between patches slice_axis : int (0,1,2); axis along which to draw slices crops : list of three tuples; each tuple (start, stop) will

define a python slice for the respective axis

rot_angle [rotate volume around Z axis before slicing along any given axis.] Note this is redundant if slice_axis = 0

nprocs : number of CPU processors for multiprocessing Pool arr_split : breakdown chunk into arr_split number of smaller chunks preprocess_func : pass a preprocessing function that applies a 2D filter on an image

3.5 Credits

3.5.1 Citations

We kindly request that you cite the following article [A1] if you use project.

3.5.2 References

Bibliography

[A1] Aniket Tekawade, Brandon A. Sforzo, Katarzyna E. Matusik, Alan L. Kastengren, and Christopher F. Powell. High-fidelity geometry generation from CT data using convolutional neural networks. In Bert Müller and Ge Wang, editors, *Developments in X-Ray Tomography XII*. SPIE, September 2019. URL: https://doi.org/10.1117/12.2540442, doi:10.1117/12.2540442.

[B1] Aniket Tekawade, Brandon A. Sforzo, Katarzyna E. Matusik, Alan L. Kastengren, and Christopher F. Powell. High-fidelity geometry generation from CT data using convolutional neural networks. In Bert Müller and Ge Wang, editors, *Developments in X-Ray Tomography XII*. SPIE, September 2019. URL: https://doi.org/10.1117/12.2540442, doi:10.1117/12.2540442.

Python Module Index

C

Index