diluvian Documentation

Release 0.0.6

Andrew S. Champion

Mar 11, 2022

Contents

Contents:

CHAPTER 1

diluvian

Flood filling networks for segmenting electron microscopy of neural tissue.

PyPI Release	
Documentation	
License	
Build Status	

Diluvian is an implementation and extension of the flood-filling network (FFN) algorithm first described in *[Januszewski2016]*. Flood-filling works by starting at a seed location known to lie inside a region of interest, using a convolutional network to predict the extent of the region within a small field of view around that seed location, and queuing up new field of view locations along the boundary of the current field of view that are confidently inside the region. This process is repeated until the region has been fully explored.

As of December 2017 the original paper's authors have released their implementation.

1.1 Quick Start

This assumes you already have CUDA installed and have created a fresh virtualenv. See installation documentation for detailed instructions.

Install diluvian and its dependencies into your virtualenv:

```
pip install diluvian
```

For compatibility diluvian only requires TensorFlow CPU by default, but you will want to use TensorFlow GPU if you have installed CUDA:

```
pip install 'tensorflow-gpu==1.3.0'
```

To test that everything works train diluvian on three volumes from the CREMI challenge:

```
diluvian train
```

This will automatically download the CREMI datasets to your Keras cache. Only two epochs will run with a small sample set, so the trained model is not useful but will verify Tensorflow is working correctly.

To train for longer, generate a diluvian config file:

```
diluvian check-config > myconfig.toml
```

Now edit settings in the `[training]` section of `myconfig.toml` to your liking and begin the training again:

```
diluvian train -c myconfig.toml
```

For detailed command line instructions and usage from Python, see the usage documentation.

1.2 Limitations, Differences, and Caveats

Diluvian may differ from the original FFN algorithm or make implementation choices in ways pertinent to your use:

- By default diluvian uses a U-Net architecture rather than stacked convolution modules with skip links. The authors of the original FFN paper also now use both architectures (personal communication). To use a different architecture, change the `factory` setting in the `[network]` section of your config file.
- Rather than resampling training data based on the filling fraction f_a, sample loss is (optionally) weighted based on the filling fraction.
- A FOV center's priority in the move queue is determined by the checking plane mask probability of the first move to queue it, rather than the highest mask probability with which it is added to the queue.
- Currently only processing of each FOV is done on the GPU, with movement being processed on the CPU and requiring copying of FOV data to host and back for each move.

CHAPTER 2

Installation

Diluvian requires CUDA. For help installing CUDA, follow the TensorFlow installation instructions for GPU support. Note that diluvian will only install TensorFlow CPU during setup, so you will want to install the version of `tensorflow-gpu` diluvian requires:

```
pip install 'tensorflow-gpu==1.3.0'
```

You should install diluvian in a virtualenv or similar isolated environment. All other documentation here assumes a a virtualenv has been created and is active.

The neuroglancer PyPI package release is out-of-date, so to avoid spurious console output and other issues you may want to install from source.

To use skeletonization you must install the skeletopyze library into the virtualenv manually. See its documentation for requirements and instructions.

2.1 Stable release

To install diluvian, run this command in your terminal:

```
pip install diluvian
```

This is the preferred method to install diluvian, 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 diluvian can be downloaded from the Github repo.

You can either clone the public repository:

```
git clone git://github.com/aschampion/diluvian
```

Or download the tarball:

```
curl  -OL https://github.com/aschampion/diluvian/tarball/master
```

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

```
python setup.py install
```

CHAPTER 3

Usage

3.1 Basic Usage

Arguments for the `diluvian` command line interface are available via help:

```
diluvian -h
diluvian train -h
diluvian fill -h
diluvian sparse-fill -h
diluvian view -h
...
```

and also *in the section below*.

3.1.1 Configuration Files

Configuration files control most of the behavior of the model, network, and training. To create a configuration file:

```
diluvian check-config > myconfig.toml
```

This will output the current default configuration state into a new file. Settings for configuration files are documented in the `config module documentation`. Each section in the configuration file, like `[training]` (known in TOML as a *table*), corresponds with a different configuration class:

- `Volume`
- `Model`
- `Network`
- `Optimizer`
- `Training`
- `Postprocessing`

To run diluvian using a custom config, use the `-c` command line argument:

```
diluvian train -c myconfig.toml
```

If multiple config files are provided, each will be applied on top of the previous state in the order provided, only overriding the settings that are specified in each file:

```
diluvian train -c myconfig1.toml -c myconfig2.toml -c myconfig3.toml
```

This allows easy compositing of multiple configurations, for example when running a grid search.

3.1.2 Dataset Files

Volume datasets are expected to be in HDF5 files. Dataset configuration is provided by TOML files that give the paths to these files and the HDF5 group paths to the relevant data within them.

Each dataset is a TOML array entry in the datasets table:

```
[[dataset]]
name = "Sample A"
hdf5_file = "sample_A_20160501.hdf"
image_dataset = "volumes/raw"
label_dataset = "volumes/labels/neuron_ids"
```

`hdf5_file` should include the full path to the file.

Multiple datasets can be included by providing multiple `[[dataset]]` sections.

To run diluvian using a dataset configuration file, use the `-v` command line argument:

```
diluvian train -v mydataset.toml
```

3.2 As a Python Library

To use diluvian in a project:

```
import diluvian
```

If you are using diluvian via Python, it most likely is because you have data in a custom format you need to import. The easiest way to do so is by constructing or extending the `Volume class`. For out-of-memory datasets, construct a volume class backed by block-sparse data structures (`diluvian.octrees.OctreeVolume`). See `ImageStackVolume` for an example.

Once data is available as a volume, normal training and filling operations can be called. See `diluvian.training.train_network()` or `diluvian.diluvian.fill_region_with_model()`.

3.3 Command Line Interface

Train or run flood-filling networks on EM data.

```
usage: diluvian [-h]
                {train,fill,sparse-fill,validate,evaluate,view,check-config,gen-subv-
↪bounds}
                ...
```

Sub-commands:

train Train a network from labeled volumes.

```
usage: diluvian train [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                      [-v VOLUME_FILES] [--no-in-memory] [-rs RANDOM_SEED]
                      [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                      [-mo MODEL_OUTPUT_FILEBASE] [-mc MODEL_CHECKPOINT_FILE]
                      [--early-restart] [--tensorboard] [--viewer]
                      [--metric-plot]
```

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

-mo, --model-output-filebase Base filename for the best trained model and other output artifacts, such as metric plots and configuration state.

-mc, --model-checkpoint-file Filename for model checkpoints at every epoch. This is different than the model output file; if provided, this HDF5 model file is saved every epoch regardless of validation performance.Can use Keras format arguments: https://keras.io/callbacks/#modelcheckpoint

--early-restart=False If training is aborted early because an early abort metric criteria, restart training with a new random seed.

--tensorboard=False Output tensorboard log files while training (limited to network graph).

--viewer=False Create a neuroglancer viewer for a training sample at the end of training.

--metric-plot=False Plot metric history at the end of training. Will be saved as a PNG with the model output base filename.

fill Use a trained network to densely segment a volume.

```
usage: diluvian fill [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                     [-v VOLUME_FILES] [--no-in-memory] [-rs RANDOM_SEED]
```

```
                    [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                    [--partition-volumes] [--no-bias]
                    [--move-batch-size MOVE_BATCH_SIZE]
                    [--max-moves MAX_MOVES]
                    [--remask-interval REMASK_INTERVAL]
                    [--seed-generator [{grid,sobel}]] [--ordered-seeds]
                    [--ignore-mask IGNORE_MASK]
                    [--background-label-id BACKGROUND_LABEL_ID] [--viewer]
                    [--max-bodies MAX_BODIES] [--reject-early-termination]
                    [--resume-file RESUME_FILENAME]
                    segmentation_output_file
```

Positional arguments:

segmentation_output_file Filename for the HDF5 segmentation output, without extension. Should contain "{volume}", which will be substituted with the volume name for each respective volume's bounds.

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

--partition-volumes=False Partition volumes and only fill the validation partition.

--no-bias=True Overwrite prediction mask at the end of each field of view inference rather than using the anti-merge bias update.

--move-batch-size=1 Maximum number of fill moves to process in each prediction batch.

--max-moves Cancel filling after this many moves.

--remask-interval Interval in moves to reset filling region mask based on the seeded connected component.

--seed-generator="sobel" Method to generate seed locations for flood filling.

Possible choices: grid, sobel

--ordered-seeds=True Do not shuffle order in which seeds are processed.

--ignore-mask=False Ignore the mask channel when generating seeds.

--background-label-id=0 Label ID to output for voxels not belonging to any filled body.

--viewer=False Create a neuroglancer viewer for a each volume after filling.

--max-bodies Cancel filling after this many bodies (only useful for diagnostics).

--reject-early-termination=False Reject seeds that terminate early, e.g., due to maximum move limits.

--resume-file Filename for the TOML configuration file of a segmented label volume from which to resume filling. The configuration should only contain one dataset.

sparse-fill Use a trained network to fill random regions in a volume.

```
usage: diluvian sparse-fill [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                            [-v VOLUME_FILES] [--no-in-memory]
                            [-rs RANDOM_SEED]
                            [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                            [--partition-volumes] [--no-bias]
                            [--move-batch-size MOVE_BATCH_SIZE]
                            [--max-moves MAX_MOVES]
                            [--remask-interval REMASK_INTERVAL]
                            [--bounds-num-moves BOUNDS_NUM_MOVES BOUNDS_NUM_
↪MOVES BOUNDS_NUM_MOVES]
                            [--augment] [-bi BOUNDS_INPUT_FILE]
```

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

--partition-volumes=False Partition volumes and only fill the validation partition.

--no-bias=True Overwrite prediction mask at the end of each field of view inference rather than using the anti-merge bias update.

--move-batch-size=1 Maximum number of fill moves to process in each prediction batch.

--max-moves Cancel filling after this many moves.

--remask-interval Interval in moves to reset filling region mask based on the seeded connected component.

--bounds-num-moves Number of moves in direction to size the subvolume bounds.

--augment=False Apply training augmentations to subvolumes before filling.

-bi, --bounds-input-file Filename for bounds CSV input. Should contain "{volume}", which will be substituted with the volume name for each respective volume's bounds.

validate Run a model on validation data.

```
usage: diluvian validate [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                         [-v VOLUME_FILES] [--no-in-memory] [-rs RANDOM_SEED]
                         [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
```

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

evaluate Evaluate a filling result versus a ground truth.

```
usage: diluvian evaluate [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                         [-v VOLUME_FILES] [--no-in-memory] [-rs RANDOM_SEED]
                         [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                         [--border-threshold BORDER_THRESHOLD]
                         [--partition-volumes]
                         ground_truth_name prediction_name
```

Positional arguments:

ground_truth_name Name of the ground truth volume.

prediction_name Name of the prediction volume.

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

--border-threshold=25 Region border threshold (in nm) to ignore. Official CREMI default is 25nm.

--partition-volumes=False Partition volumes and only evaluate the validation partitions.

view View a set of co-registered volumes in neuroglancer.

```
usage: diluvian view [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                     [-v VOLUME_FILES] [--no-in-memory] [-rs RANDOM_SEED]
                     [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                     [--partition-volumes]
                     [volume_name_regex]
```

Positional arguments:

volume_name_regex Regex to filter which volumes of those defined in the volume configuration should be loaded.

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

--partition-volumes=False Partition volumes and view centered at the validation partitions.

check-config Check a configuration value.

```
usage: diluvian check-config [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                             [-v VOLUME_FILES] [--no-in-memory]
                             [-rs RANDOM_SEED]
                             [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                             [config_property]
```

Positional arguments:

config_property Name of the property to show, e.g., 'training.batch_size'.

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

gen-subv-bounds Generate subvolume bounds.

```
usage: diluvian gen-subv-bounds [-h] [-c CONFIG_FILES] [-cd] [-m MODEL_FILE]
                                [-v VOLUME_FILES] [--no-in-memory]
                                [-rs RANDOM_SEED]
                                [-l {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                                [--bounds-num-moves BOUNDS_NUM_MOVES BOUNDS_
↪NUM_MOVES BOUNDS_NUM_MOVES]
                               bounds_output_file num_bounds
```

Positional arguments:

bounds_output_file Filename for the CSV output. Should contain "{volume}", which will be substituted with the volume name for each respective volume's bounds.

num_bounds Number of bounds to generate.

optional arguments

-c=[], --config-file=[] Configuration files to use. For defaults, see 'diluvian/conf/default.toml'. Values are overwritten in the order provided.

-cd Add default configuration file to chain of configuration files.

-m, --model-file Existing network model file to use for prediction or continued training.

-v=[], --volume-file=[] Volume configuration files. For example, see 'diluvian/conf/cremi_datasets.toml'.Values are overwritten in the order provided.

--no-in-memory=True Do not preload entire volumes into memory.

-rs, --random-seed Seed for initializing the Python and NumPy random generators. Overrides any seed specified in configuration files.

-l, --log Set the logging level.

Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

--bounds-num-moves Number of moves in direction to size the subvolume bounds.

CHAPTER 4

Contributing

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

4.1 Development

Here's how to set up *diluvian* for local development.

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

```
$ git clone git@github.com:your_name_here/diluvian.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 diluvian
$ cd diluvian/
$ 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 diluvian tests
$ python setup.py test
$ 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.

CHAPTER 5

Contributors

- Andrew S. Champion <andrew.champion@gmail.com>

5.1 Acknowledgements

This library is an implementation and extension of the flood-filling network algorithm first described in *[Januszewski2016]* and network architectures in *[He2016]* and *[Ronneberger2015]*.

This library is built on the wonderful Keras library by François Chollet and TensorFlow.

Skeletonization uses the skeletopyze library by Jan Funke, which is an implementation of *[Sato2000]* and *[Bitter2002]*.

Diluvian uses a packaging and build harness cookiecutter template.

CHAPTER 6

History

6.1 0.0.6 (2018-02-13)

- Add CREMI evaluation command.
- Add 3D region filling animation.
- Fix region filling animations.
- F_0.5 validation metrics.
- Fix pip install.
- Many other fixes and tweaks (see git log).

6.2 0.0.5 (2017-10-03)

- Fix bug creating U-net with far too few channels.
- Fix bug causing revisit of seed position.
- Fix bug breaking sparse fill.

6.3 0.0.4 (2017-10-02)

- Much faster, more reliable training and validation.
- U-net supports valid padding mode and other features from original specification.
- Add artifact augmentation.
- More efficient subvolume sampling.
- Many other changes.

6.4 0.0.3 (2017-06-04)

- Training now works in Python 3.
- Multi-GPU filling: filling will now use the same number of processes and GPUs specified by `training.num_gpus`.

6.5 0.0.2 (2017-05-22)

- Attempt to fix PyPI configuration file packaging.

6.6 0.0.1 (2017-05-22)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Bibliography

[Januszewski2016] Michał Januszewski, Jeremy Maitin-Shepard, Peter Li, Jorgen Kornfeld, Winfried Denk, and Viren Jain. Flood-filling networks. *arXiv preprint arXiv:1611.00421*, 2016.

[Januszewski2016] Michał Januszewski, Jeremy Maitin-Shepard, Peter Li, Jorgen Kornfeld, Winfried Denk, and Viren Jain. Flood-filling networks. *arXiv preprint arXiv:1611.00421*, 2016.

[He2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[Ronneberger2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: convolutional networks for biomedical image segmentation. MICCAI 2015. 2015.

[Sato2000] Mie Sato, Ingmar Bitter, Michael A. Bender, Arie E. Kaufman, and Masayuki Nakajima. TEASAR: tree-structure extraction algorithm for accurate and robust skeletons. PCCGA 2000. 2000.

[Bitter2002] Ingmar Bitter, Arie E. Kaufman, and Mie Sato. Penalized-distance volumetric skeleton algorithm. IEEE Trans on Visualization and Computer Graphics. 2002.