DL Data Validation Toolset Documentation

Release 0.1.0

Kevin Wierman

Jul 26, 2017

Contents:

The DL Data Validation Toolset is a framework built around a series of tools meant to help establish if a generated HDF5 file for the DL in HEP effort contains valid data. Each test is meant to be run as a unit test on a generated file and establish what the file contains along with problems that might arise due to using this file for deep learning.

CHAPTER 1

Goal

The goal of this project is to create comprehensive reports on data produced external to the deep learning framework. Reports created should contain information on the validity of using these files in Deep Learning Frameworks as well as detailed information on the contents of the files and diagnostically relevant calculations.

Part of the diagnostic tools should be some visualization of the contents of the files.

Installation

Prereqs

This framework depends heavily on HDF5. Users should have the HDF5 libraries with compression back installed.

For instance, on Ubuntu, this can be accomplished with

```
sudo apt-get install libhdf5-serial-dev
```

Similarly, on RHEL, this is done with `dev` being replaced by `devel`.

Installation with Pip

For non-development usage, one may install with:

```
pip install <options> git+https://github.com/HEP-DL/dl_data_validation_toolset
```

Where <options> typically contain `--user`, `--upgrade` or both.

Installation with dist-utils

For development usage, one may install with:

```
git clone https://github.com/HEP-DL/dl_data_validation_toolset
cd dl_data_validation_toolset
make install
```

Usage

Command Line Utility

The primary interface is the command line interface.

By default, the command line utility can be called with

```
generate_report
```

This will look for data in the `data` subdirectory of the current working directory and place output in the `results` subdirectory of the current working directory.

The library can be configured with `YAML` by specifying the `--config` flag.

An example configuration YAML looks like:

```
results_path: /home/my_user/results
scan_paths: [/home/my_user/dl_data/, /data/]
tar: False
```

Output

The command link util creates a tarball of the HTML output of the report generator. This is placed in the results path and is named by current time stamp.

If one untars the ball and opens the `index.html` file, the summary page contains links to file reports, etc...

Note that in order for the page to appear correctly, internet connectivity is required.

Python API

Refer to the *dl_data_validation_toolset* page.

Development

This package is very much a work in progress.

Test Driven Development

Unit tests are kept in the *tests* directory. These are unit tests for developing code. This is not to be confused with data tests, though the code pattern for both resemble each other.

These can be run with the command:

```
make test
```

Bear in mind that flake8 is used to lint the code.

Documentation

Documentation can be built with:

```
make docs
```

Inline documentation follows Sphinx guidelines. An example can be found here.

Data Tests

The data validation tests can be found in dl_data_validation_toolset/data_tests.

To create a new data validation test or series of tests, simply copy one of the examples (such as consistency). Once your test is ready, please modify *__test_names__* to match the filename of your new test, minus the file extension.

dl_data_validation_toolset

dl_data_validation_toolset package

Subpackages

dl_data_validation_toolset.data_tests package

Submodules

dl_data_validation_toolset.data_tests.labels module

class dl_data_validation_toolset.data_tests.labels.**LabelTests**(*filename*)
Bases: *dl_data_validation_toolset.framework.base_test.BaseTest*

logger = <logging.Logger object>

test_label_diversity()

test_label_exists()

test_nonzero_labels()

Module contents

Definitions of the unit tests to perform on data

dl_data_validation_toolset.data_tests.**initialize**()

dl_data_validation_toolset.framework package

Submodules

dl_data_validation_toolset.framework.base_test module

class dl_data_validation_toolset.framework.base_test.**BaseTest**(*filename*)
Bases: object

logger = <logging.Logger object>

validate(*report*)

dl_data_validation_toolset.framework.base_unittest module

class dl_data_validation_toolset.framework.base_unittest.**BaseTestCase**(*methodName='runTest'*)
Bases: unittest.case.TestCase

setUp()
Monkeypatches the file functionality out so that we don't run on real files during unit tests.

tearDown()

Module contents

dl_data_validation_toolset.templates package

Module contents

The templates module contains mako templates for the HTML report generation.

Submodules

dl_data_validation_toolset.cli module

Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index