proton$_{d}ecay_{s}tudy Documentation$

Release 0.1.0

Kevin Wierman

Feb 25, 2018

Contents

Liquid argon time projection chambers (LArTPCs) are an innovative technology used in neutrino physics measurements that can also be utilized in establishing lifetimes on several partial lifetimes for proton and neutron decay. Current analyses suffer from low efficiencies and purities that arise from the misidentification of nucleon decay final states as background processes and vice-versa. One solution is to utilize convolutional neural networks (CNNs) to identify decay topologies in LArTPC data. In this study, CNNs are trained on Monte Carlo simulated data, labeled by truth, and then assessed by out-of-sample simulation. Currently running LArTPCs play an instrumental role in establishing the capabilities of this technology. Simultaneousy, the next generation tens-of-kilotons flagship LArTPC experiment – one of whose main charges is to search for nucleon decay – is planning on using this technology in the future. We discuss analysis possibilities and further, a potential application of proton decay-sensitive CNN-enabled data acquisition.

Contents:

CHAPTER 1

Installation

As far as pre-requistes go, the big one is Tensorflow. We require *Tensorflow>=1.0.0*, but this is not strictly enforced due to some issues on certain machines.

The easiest way to install this package is by using pip:

```
pip install git+https://github.com/HEP-DL/proton_decay_study
```

If there is an existing installation, you can provide an upgrade flag:

```
pip install git+https://github.com/HEP-DL/proton_decay_study
```

CHAPTER 2

Usage

Nominally, this should be used via the Python API.

There are a few convenience endpoint defitions in *cli.py*.

For instance, the Kevnet training can be called with

```
train_kevnet --steps=100 --epochs=1000 --history=stage1.json --output=stage1.h5 dl_
↪data/v04_00_00/*.h5
```

The other endpoints can be called similarly

CHAPTER 3

proton_decay_study

3.1 proton_decay_study package

3.1.1 Subpackages

proton_decay_study.callbacks package

Submodules

proton_decay_study.callbacks.default module

Defines the default callbacks for usage in the mod:*proton_decay_study*

class proton_decay_study.callbacks.default.**HistoryRecord**
Bases: keras.callbacks.Callback

This is a stub in place for working on recording the training history

on_batch_end(*batch*, *logs={}*)

on_train_begin(*logs={}*)

write(*filename*)

Module contents

proton_decay_study.config package

Module contents

class proton_decay_study.config.**Config**
Bases: object

Represents configuration of network training

proton_decay_study.generators package

Submodules

proton_decay_study.generators.base module

class proton_decay_study.generators.base.**BaseDataGenerator**
Bases: object

Base data generator which hooks into the networks to provide an interface to the incoming data.

logger = <logging.Logger object>

next()

proton_decay_study.generators.gen3d module

class proton_decay_study.generators.gen3d.**Gen3D**(*datapaths*, *datasetname*, *labelsetname*, *batch_size=1*)
Bases: *proton_decay_study.generators.base.BaseDataGenerator*

Creates a generator for a list of files

input
Input shape property

Returns: A tuple representing

logger = <logging.Logger object>

next()

output
Output shape property

Returns: A tuple representing the shape of the first data this picks out of the file

class proton_decay_study.generators.gen3d.**Gen3DRandom**(*datapaths*, *datasetname*, *labelsetname*, *batch_size=1*)
Bases: *proton_decay_study.generators.gen3d.Gen3D*

proton_decay_study.generators.multi_file module

class proton_decay_study.generators.multi_file.**MultiFileDataGenerator**(*datapaths*, *datasetname*, *labelsetname*, *batch_size=10*)
Bases: *proton_decay_study.generators.base.BaseDataGenerator*

Creates a generator for a list of files

input

logger = <logging.Logger object>

next()
This should iterate over both files and datasets within a file.

output

proton_decay_study.generators.single_file module

class proton_decay_study.generators.single_file.**SingleFileDataGenerator**(*datapath*, *dataset*, *labelset*, *batch_size=10*)

Bases: *proton_decay_study.generators.base.BaseDataGenerator*

Creates a generator for a single file

logger = <logging.Logger object>

next()

proton_decay_study.generators.threaded_gen3d module

class proton_decay_study.generators.threaded_gen3d.**SingleFileThread**(*datasetname*, *labelsetname*, *batch_size*)

Bases: threading.Thread

Wrapper thread for buffering data from a single file

activeThreads = []

static killRunThreads(*frame*)
Sets the thread kill flag to each of the ongoing analysis threads

logger = <logging.Logger object>

queue = <Queue.Queue instance>

queueLock = <thread.lock object>

run()
Loops over queue to accept new configurations

single_status()

static startThreads(*datasetname*, *labelsetname*, *batch_size*)

static status()

threadLock = <thread.lock object>

visit(*parent*)

static waitTillComplete()

class proton_decay_study.generators.threaded_gen3d.**ThreadedMultiFileDataGenerator**(*datapaths*, *datasetname*, *labelsetname*, *batch_size=*, *nThreads=8*

Bases: *proton_decay_study.generators.base.BaseDataGenerator*

Uses threads to pull asynchronously from files

check_and_refill()

input

logger = <logging.Logger object>

next()

output

status()

proton_decay_study.generators.threaded_multi_file module

class proton_decay_study.generators.threaded_multi_file.**SingleFileThread**(*datasetname*, *labelsetname*, *batch_size*)

Bases: threading.Thread

Represents a single file that is asynchronously

activeThreads = []

get

static killRunThreads(*frame*)
Sets the thread kill flag to each of the ongoing analysis threads

logger = <logging.Logger object>

queue = <Queue.Queue instance>

queueLock = <thread.lock object>

run()
Loops over queue to accept new configurations

static startThreads(*datasetname*, *labelsetname*, *batch_size*)

threadLock = <thread.lock object>

static waitTillComplete()

class proton_decay_study.generators.threaded_multi_file.**ThreadedMultiFileDataGenerator**(*datap*
datas
name
la-
belse
name
batch
nThr

Bases: *proton_decay_study.generators.base.BaseDataGenerator*

Uses threads to pull asynchronously from files

input

logger = <logging.Logger object>

next()

output

Module contents

proton_decay_study.models package

Submodules

proton_decay_study.models.kevnet module

class proton_decay_study.models.kevnet.**Kevnet**(*generator*)

Bases: keras.engine.training.Model

assemble(*generator*)

logger = <logging.Logger object>

proton_decay_study.models.kevnet_fcn module

class proton_decay_study.models.kevnet_fcn.**capetian_modifier**(*generator*)

Bases: *proton_decay_study.models.kevnet.Kevnet*

logger = <logging.Logger object>

class proton_decay_study.models.kevnet_fcn.**percussive_treasurership**(*generator*)

Bases: *proton_decay_study.models.kevnet.Kevnet*

logger = <logging.Logger object>

Module contents

proton_decay_study.visualization package

Submodules

proton_decay_study.visualization.intermediate module

class proton_decay_study.visualization.intermediate.**IntermediateVisualizer**(*model*, *layer_name*, *data*)

Bases: keras.engine.training.Model

infer()

proton_decay_study.visualization.kevnet module

class proton_decay_study.visualization.kevnet.**KevNetVisualizer**(*model*, *data*)

initialize()

layers = ['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', 'block5_conv

logger = <logging.Logger object>

mkdir()

run()

Module contents

3.1.2 Submodules

3.1.3 proton_decay_study.cli module

3.1.4 Module contents

Base package for Proton Decay Study :author: Kevin Wierman

CHAPTER 4

Contributions

Your contributions to this are more than welcome!

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

V

W