LCD and LArIAT Datasets And CaloDNN and LArTPCDNN

Amir Farbin
(ATLAS/UTA)

LCD Calo Dataset made by M. Pierini (CMS/CERN) + JR Vlimant (CMS/Caltech)
LArIAT Dataset made by S. Shahsavarani (Neutrinos/UTA) + AF

Intro

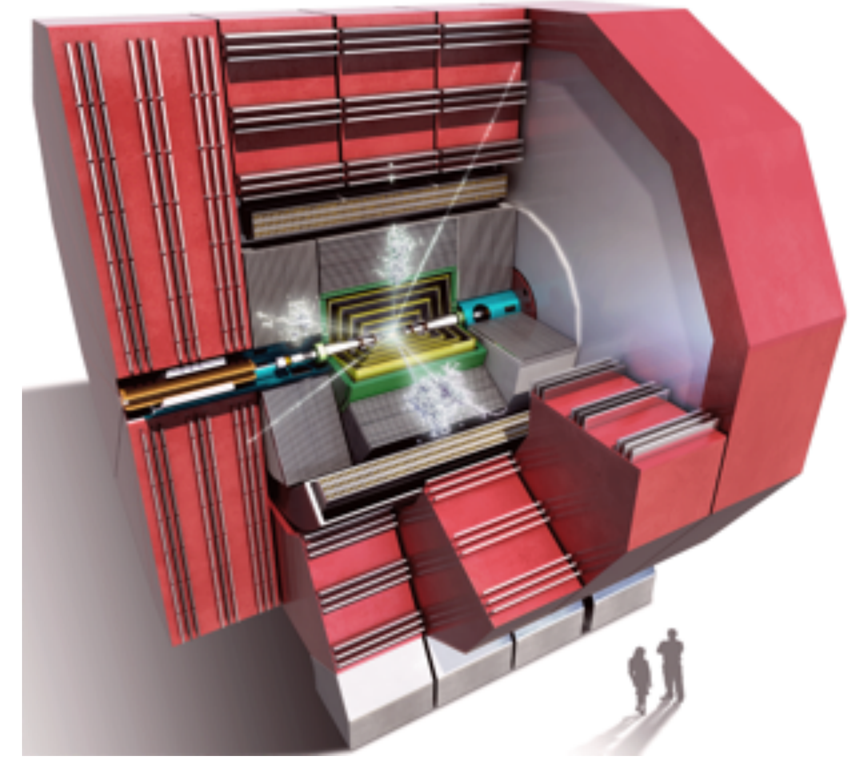
- Reconstruction level DL requires realistic detector simulation… not as easy as 4-vectors or parameterized detectors.
- Experiments are understandably strict about their data. Prohibits:
  - Cross experiment or HEP/ML collaboration
  - Rapid publication of DL R&D (no physics).
- Imaging detectors (Granular Calorimeters, TPCs, Cherenkov, …) ideally suited for Deep Learning.
- We generated the LCD and LArIAT Datasets to avoid these issues.
  - Dataset and code very similar, so I'll talk about both.
  - Weekly LCD meetings to organize work. Should do for LArIAT.
- Data Science @ LHC (Nov 2015 @ CERN) -> DS@HEP.
  - Experts workshop (July 2015): these datasets were introduced in prim. Goal was to make them public for NIPS… btut we didn't get a workshop and got busy.
  - Goal is to reveal datasets at next workshop. May 8-12 @ FNAL. https://indico.fnal.gov/conferenceDisplay.py?confId=13497

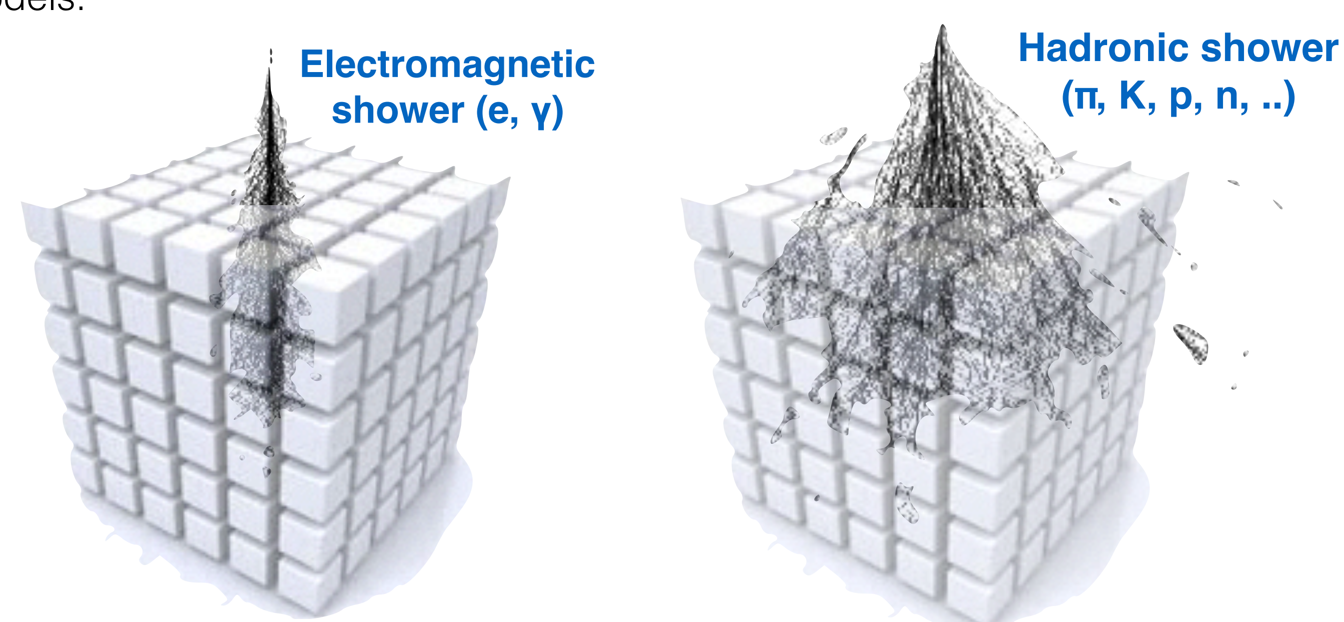
Message

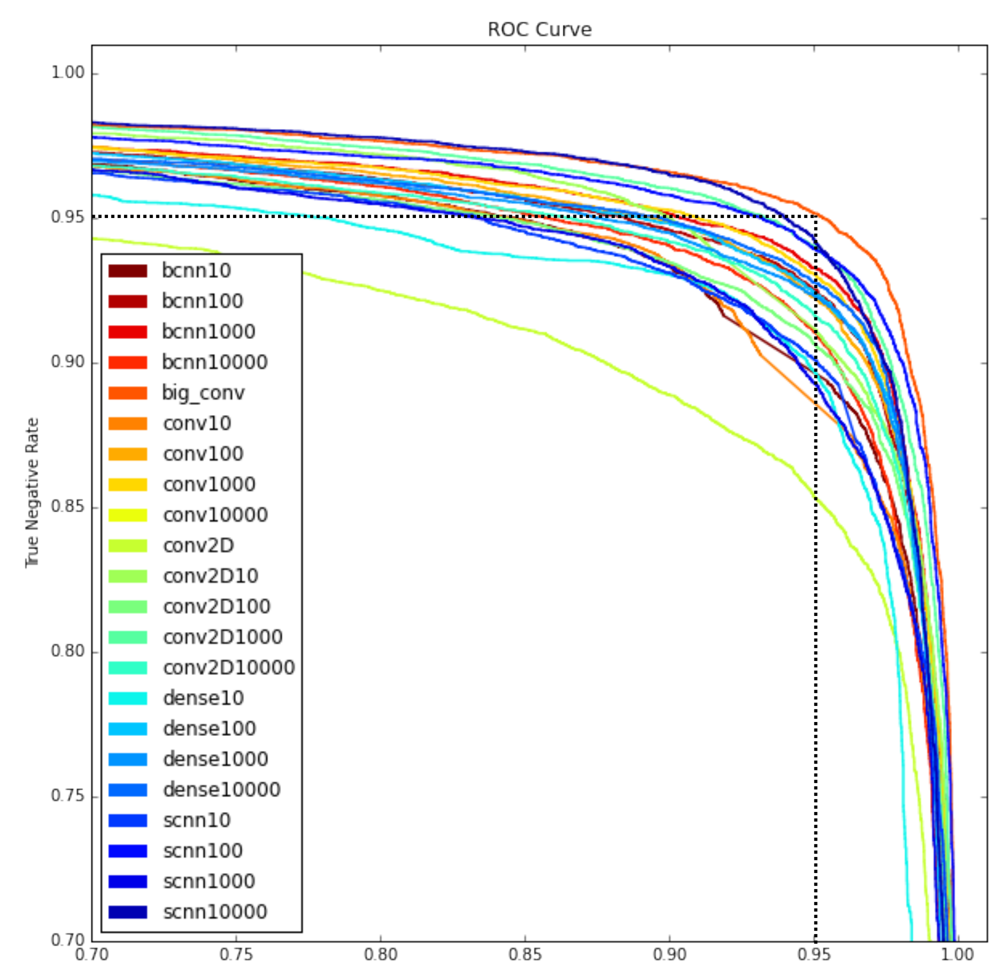
- Everyone is busy, so help is appreciated:
  - Contribute to finalizing data and Nature Scientific Data paper.
  - Collaborate on research.
  - We ask that Dataset paper would be the first, and all work done before DS@HEP WS be collaborative.
- These are large datasets (LCD = 20 GB so far, LArIAT = 20 TB)
  - Distribution and processing require extra thought
  - Code to efficiently read the data should be provided.
- Not clear if we should distribute full running examples… or even collaborative code used for papers.
  - I'll present my packages… open to input and suggestions.
- I feel like I'm often working in a corner may make mistakes.
  - I have lots of questions I have no one to ask.
  - I hope this forum could be a place to share experiences and give advice…

LCD Calorimeter

- CLIC is a proposed CERN project for a linear accelerator of electrons and positrons to TeV energies (~ LHC for protons)
  - Not a real experiment yet, so we) can simulate data and make it public.
  - Simpler geometry than ATLAS…
- The LCD calorimeter is an array of absorber material and silicon sensors comprising the most granular calorimeter design available
  - Data is essentially a 3D image
  - So far several million Pi0, Elec, ChPi, Gamma. 10 to 510 GeV. Low energy and Jet samples planned.
  - ECAL (25x25x25) / HCAL (5x5x60) "window". Aux info: Energy, …
- First studies, π^0 vs γ classification with various DNNs by summer students.
  - Code/results not collected… but should be easy to redo.
  - New version of dataset.
  - Some visualization code exists… Full running example in CaloDNN.
- Many interesting problems: PID Classification, Energy Regression, Shower generative models.







Join the fun....

Imaging calorimeter data for Machine Learning applications in HEP

Josh Bendavid[a] **Kaustuv Datta**[a,b] **Amir Farbin**[c] **Nikolaus Howe**[d,e] **Jayesh Mahapatra**[d] **Maurizio Pierini**[d] **Maria Spiropulu**[a] **Jean-Roch Vlimant**[a]

[a] *California Institute of Technology*
[b] *Reed College*
[c] *University of Texas at Arlington*
[d] *CERN*
[e] *Williams College*

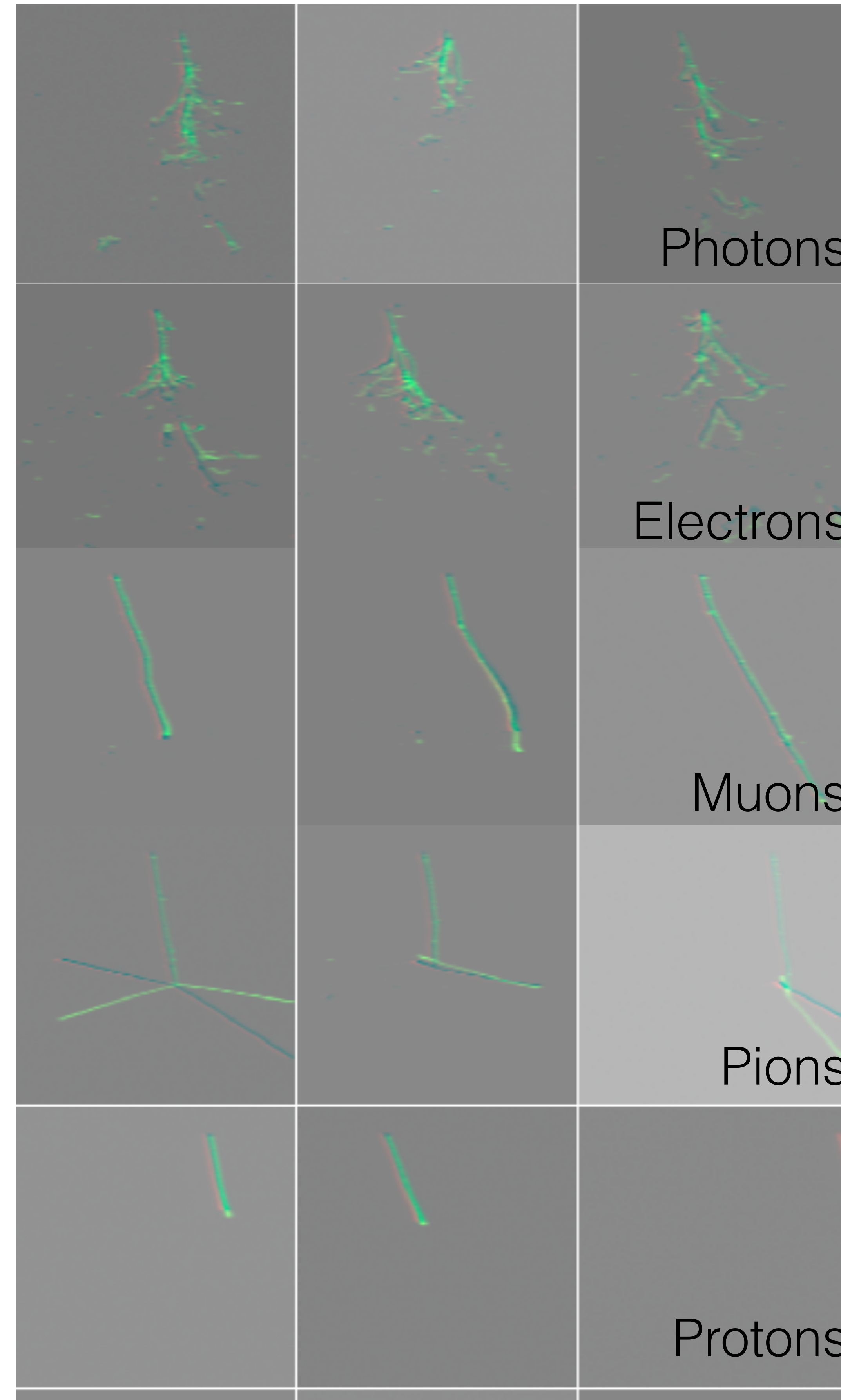
Photon identifiction and energy measurement with a highly granular calorimeter through Deep Learning

Josh Bendavid[a] **Kaustuv Datta**[a,b] **Amir Farbin**[c] **Nikolaus Howe**[d,e] **Jayesh Mahapatra**[d] **Maurizio Pierini**[d] **Maria Spiropulu**[a] **Jean-Roch Vlimant**[a]

[a] *California Institute of Technology*
[b] *Reed College*
[c] *University of Texas at Arlington*
[d] *CERN*
[e] *Williams College*

LArIAT Data

- LArIAT is a small LArTPC detector: 2 wire places with 240 wires each, 4096 samples.
- 1 M each of: antielectron, kaonPlus, nue_CC, nutaubar_CC pionMinus, antimuon, nue_NC, nutaubar_NC, pionPlus, antiproton, muon, numubar_CC, nutau_CC, electron, numubar_NC nutau_NC, proton, nuebar_CC, numu_CC, photon, kaonMinus, nuebar_NC, numu_NC, pion_0
- Data: Sim done.
    - Raw ADC readout: 2 x 4096 x 240 (essentially no noise)
    - Geant4 charge deposits. SparseTensor allows creating 3D images of any resolution. (Needs reprocessing of data-prep steps)
    - Aux info: type of interaction, energy, …
- Studies:
    - Preliminary studies very promising.
    - Subsequent work (P. Sadowski + C. Eng) showed impressive classification performance using siamese inception model trained for 1 week.
    - A bit of work on energy regression… not as straightforward.
    - Progress stalled…
- Interesting problems: PID classification, Energy Regression, Compression/ Noise suppression, 2x 2D -> 3D (DNN tomography)

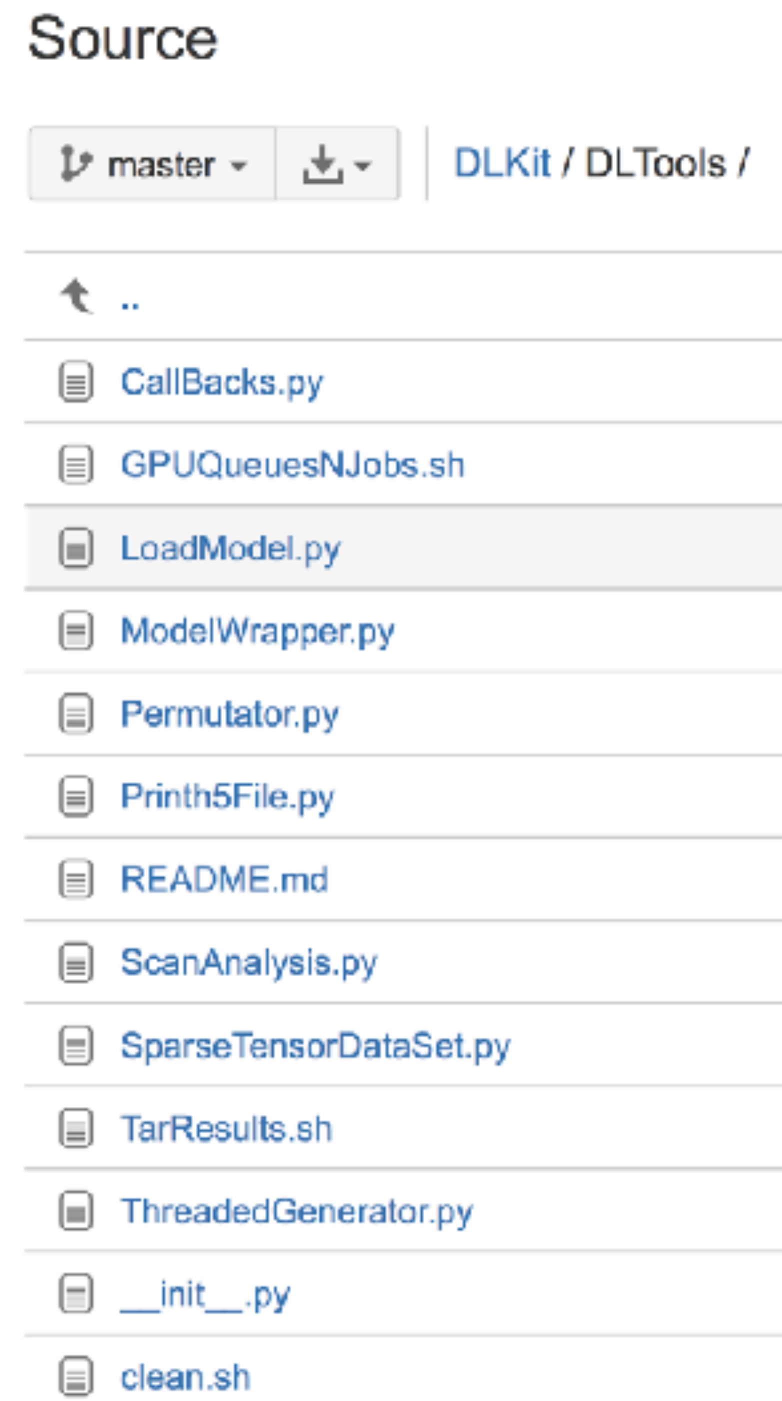


Technical Challenges

- Data comes as many h5 files, each containing O(1000) events, organized into directories by particle type.
- Needs to be read, mixed, “labeled”, and normalized…. can be time consuming.
- Doesn’t fit in memory…
- Very difficult to keep the GPU fed with data. GPU utilization often < 10%, rarely > 50%.
- Keras python generator mechanism:
    - Allows reading on the fly and parallel read
    - Found 2 problems: (Am I crazy?)
        - Multiprocessing requires the generators to be thread_safe, which means putting in a locking mechanism which only allows one process to read the data at a time. So > 2 processes not useful.
        - Easy to mess up and have parallel generator instances deliver overlapping data.
    - LCD data is ~ x10 slower with naive Keras generator vs preloading in memory.
- I wrote a standalone parallel generator: DLKit/ThreadedGenerator:
    - Python Global Interpreter Lock (GIL) allows only one thread to run at a time… so must use multiprocessing.
    - Current implementation: Filler process sends requests (file/block) via multiprocessing queues to workers processes that deliver data to corresponding threads via pipes that feed the generator via thread queues.
    - Bottle neck is the process to thread pipe… data needs to be serialized. Working on share memory solution…
    - Data can be premixed. Premix: ~2x slower than data in memory. Mix as you go: ~4x slower than data in memory.
    - System resources become problem when running many trainings in same system. Working on framework upgrade to simultaneously train several models with same data.

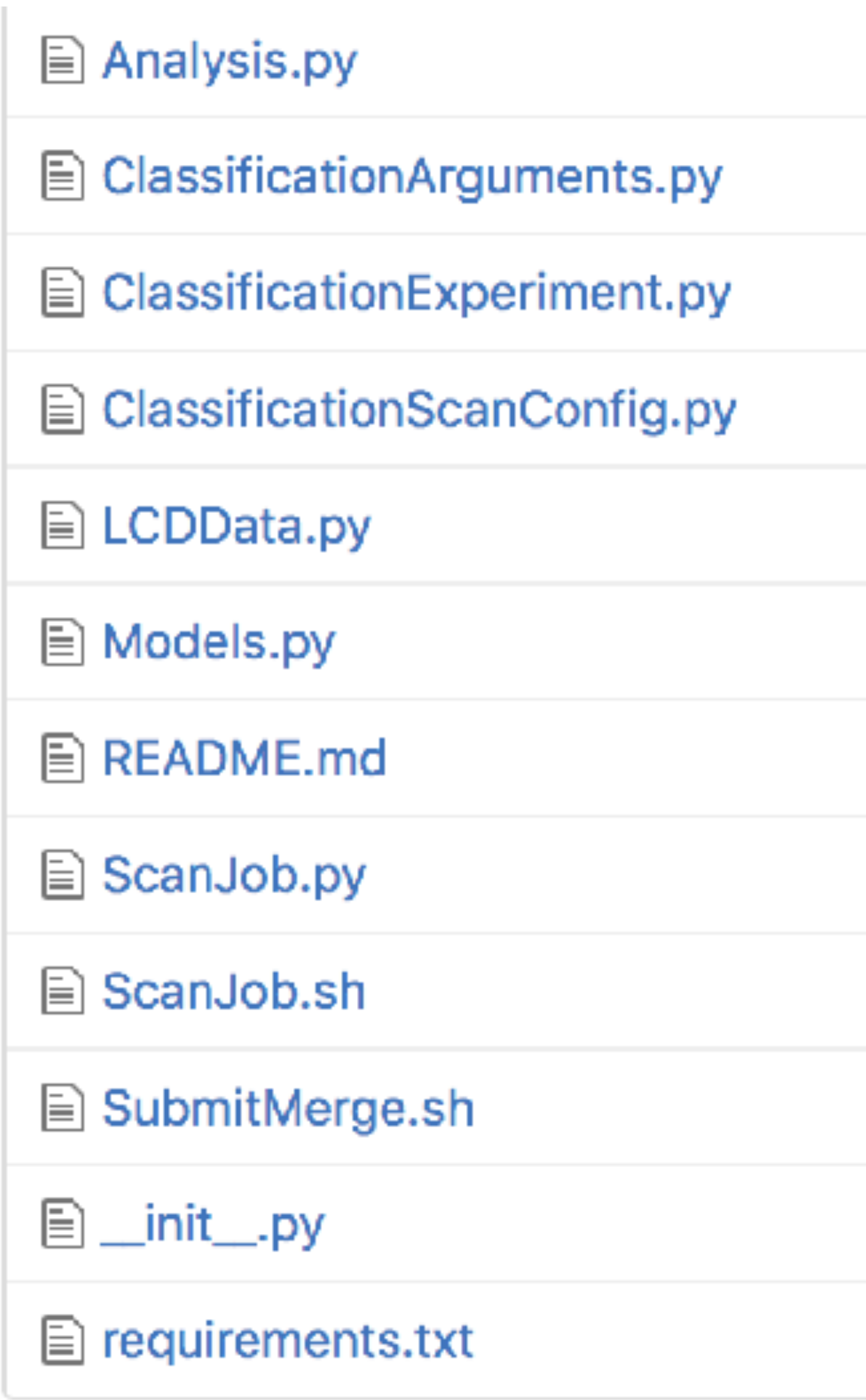
DLKit

- Thin layer on top of Keras.
- My personal DNN framework. I imagine many of you would write something similar…
- Handles book keeping for comparing large number of training sessions (e.g. for hyper parameter scan or optimization)
- Tools necessary to setup HEP problems.
- I have several HEP problems setup using this package:
  - EventClassificationDNN, MEDNN, CaloDNN, LArTPCDNN, …
- Hyperas or Spearmint integration demonstrated, but needs work.
- Keras / MPI Integration also in the works.
- Already ran on BlueWaters and Titan.
- https://bitbucket.org/anomalousai/dlkit/src



CaloDNN/LArTPCDNN

- Instantiates generators for efficiently reading or premixing data.
- Provides out-of-the-box running realistic (not toy) models.
- Orchestrates running large HP scans.
    - Makes tables…
    - Jupyter notebook analysis in works.
- Generates standard plots.
- https://github.com/UTA-HEP-Computing/CaloDNN
- Polishing up package for public…
- Gearing up for a big BlueWaters run…
    - Large HP Scan (not optimization)
    - “Regularization”: training time.



afarbin — ssh -YX orodruin.uta.edu — 111×29

```
Last login: Tue Feb 28 08:47:35 2017 from 192.168.1.13
afarbin@thecount:~$ cd LCD/DLKit/
afarbin@thecount:~/LCD/DLKit$ source setup.sh
(Keras) afarbin@thecount:~/LCD/DLKit$ python -m CaloDNN.ClassificationExperiment --help
usage: ClassificationExperiment.py [-h] [-C CONFIG] [-L LOADMODEL]
                                   [--gpu GPUID] [--cpu] [--NoTrain]
                                   [--NoAnalysis] [--Test] [-s HYPERPARAMSET]
                                   [--nopremix] [--preload] [-r RUNNINGTIME]

optional arguments:
  -h, --help            show this help message and exit
  -C CONFIG, --config CONFIG
                        Use specified configuration file.
  -L LOADMODEL, --LoadModel LOADMODEL
                        Loads a model from specified directory.
  --gpu GPUID           Use specified GPU.
  --cpu                 Use CPU.
  --NoTrain             Do not run training.
  --NoAnalysis          Do not run analysis.
  --Test                Run in test mode (reduced examples and epochs).
  -s HYPERPARAMSET, --hyperparamset HYPERPARAMSET
                        Use specificed (by index) hyperparameter set.
  --nopremix            Do not use the premixed inputfile. Mix on the fly.
  --preload             Preload the data into memory. Caution: requires lots
                        of memory.
  -r RUNNINGTIME, --runningtime RUNNINGTIME
                        End training after specified number of seconds.
(Keras) afarbin@thecount:~/LCD/DLKit$ 
```

ScanConfig.py

```

# Input for Premixed Generator
InputFile="/data/afarbin/LCD/LCD-Merged-All.h5"
# Input for Mixing Generator
FileSearch="/data/afarbin/LCD/*/*.h5"

# Generation Model
Config={
    "GenerationModel":"'Load'",
    "MaxEvents":int(3.e6),
    "NTestSamples":100000,
    "NClasses":4,

    "Epochs":1000,
    "BatchSize":1024,

    # Configures the parallel data generator that read the input.
    # These have been optimized by hand. Your system may have
    # more optimal configuration.
    "n_threads":4,  # Number of workers
    "multiplier":2, # Read N batches worth of data in each worker

    # How weights are initialized
    "WeightInitialization":"'normal'",

    # Normalization determined by hand.
    "ECAL":True,
    "ECALNorm":150.,

    # Normalization needs to be determined by hand.
    "HCAL":True,
    "HCALNorm":150.,
```

```
    # Set the ECAL/HCAL Width/Depth for the Dense model.
    # Note that ECAL/HCAL Width/Depth are changed to "Width" and "Depth",
    # if these parameters are set.
    "HCALWidth":32,
    "HCALDepth":2,
    "ECALWidth":32,
    "ECALDepth":2,

    # No specific reason to pick these. Needs study.
    # Note that the optimizer name should be the class name (https://keras.io/optimizers/)
    "loss":"'categorical_crossentropy'",

    # Specify the optimizer class name as True (see: https://keras.io/optimizers/)
    # and parameters (using constructor keywords as parameter name).
    # Note if parameter is not specified, default values are used.
    "optimizer":"'SGD'",
    #"lr":0.01,
    #"decay":0.001,

    # Parameter monitored by Callbacks
    "monitor":"'val_loss'",

    # Active Callbacks
    # Specify the CallBack class name as True (see: https://keras.io/callbacks/)
    # and parameters (using constructor keywords as parameter name,
    # with classname added).
    "ModelCheckpoint":True,
    "Model_Chekpoint_save_best_only":False,

    # Configure Running time callback
    # Set RunningTime to a value to stop training after N seconds.
    "RunningTime": 3600,
}
```

```

# Parameters to scan and their scan points.
Params={ "Width":[32,64,128,256,512],
         "Depth":range(1,5),
         "lr":[0.1,0.01,0.001],
         "decay":[0.1,0.01,0.001],
          }

```

```
(Keras) afarbin@thecount:~/LCD/DLKit$
(Keras) afarbin@thecount:~/LCD/DLKit$
(Keras) afarbin@thecount:~/LCD/DLKit$ python -m DLTools.ScanAnalysis TrainedModels.TestScan.1/
Using Theano backend.
                          Ele_AUC    Width    Depth    Pi0_AUC    ChPi_AUC    Gamma_AUC
----------------------  ---------  -------  -------  ---------  ----------  -----------
CaloDNN_32_1_Merged.23     0.9452       32        1     0.8608      0.9971       0.8802
CaloDNN_128_1_Merged.1     0.9639      128        1     0.9151      0.9964       0.9299
CaloDNN_64_1_Merged.1      0.9810       64        1     0.9453      0.9975       0.9508
CaloDNN_256_1_Merged.1     0.9870      256        1     0.9529      0.9987       0.9494
(Keras) afarbin@thecount:~/LCD/DLKit$
```