Fast Calorimeter Simulation Challenge 2022

calochallenge.github.io/homepage

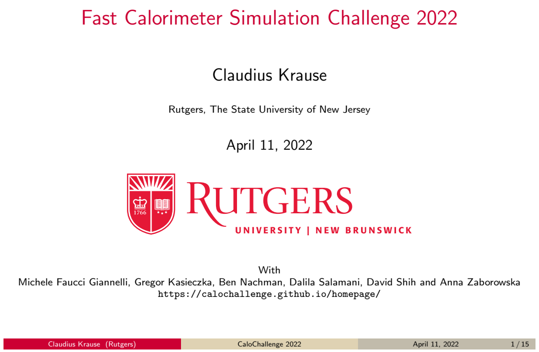
Michele Faucci Giannelli, Gregor Kasieczka, Claudius Krause, Ben Nachman, Dalila Salamani, David Shih, and **Anna Zaborowska**

Learning to Discover, Workshop on Generative Models,
April 26th 2022

Acknowledgements

Almost all of the slides were prepared by Claudius Krause and presented at the joint HSF and ML4Sim meeting:

https://indico.cern.ch/event/1140563/



Fast Calorimeter Simulation Challenge 2022

Claudius Krause

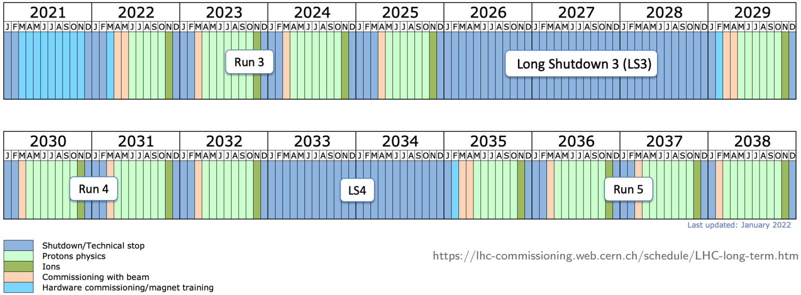
Rutgers, The State University of New Jersey

April 11, 2022

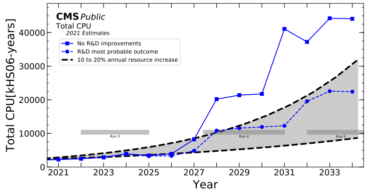
RUTGERS
UNIVERSITY | NEW BRUNSWICK
1766

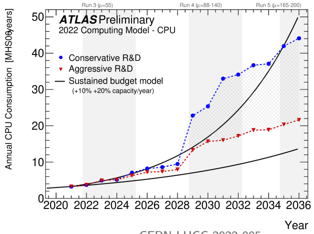
With
Michele Faucci Giannelli, Gregor Kasieczka, Ben Nachman, Dalila Salamani, David Shih and Anna Zaborowska
https://calochallenge.github.io/homepage/

The LHC will need a lot of computing ressources.



https://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm

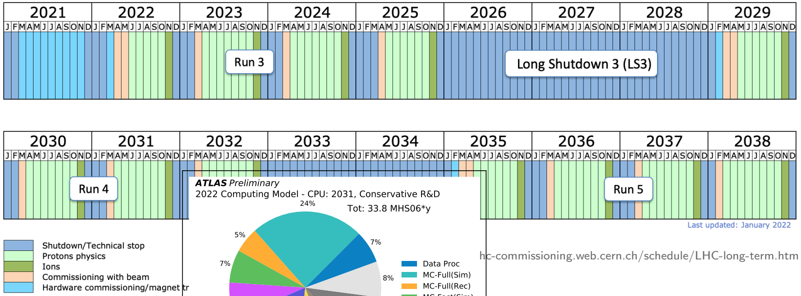




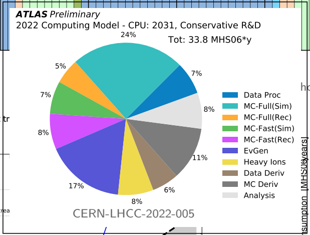
https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

CERN-LHCC-2022-005

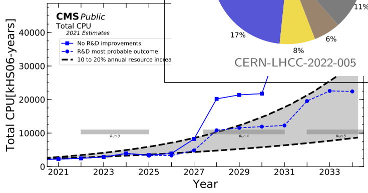
The LHC will need a lot of computing ressources.



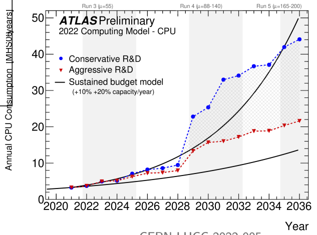
lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm



CERN-LHCC-2022-005



https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults



CERN-LHCC-2022-005

There was a lot of progress in the last years.

- The immense progress of ML in the past decade led to awesome results for calorimeter simulation surrogates!

⇒ We have seen the use of GANs, VAEs, Normalizing Flows, and their derivates on a variety of datasets.

- Examples (biased towards us organizers and non-exhaustive):

  CaloGAN: 1712.10321 PRD; 1705.02355 PRL
  Erdmann et al.: 1802.03325 CSBS; 1807.01954 CSBS
  Belayneh et al.: 1912.06794 EPJC
  BIB-AE: 2005.05334 CSBS; 2112.09709
  AtlFast3: 2109.02551; FastCaloGAN: ATL-SOFT-PUB-2020-006
  CaloFlow: 2106.05285; 2110.11377

⇒ No systematic comparison of methods available!

Why a Challenge?

- A challenge compares a variety of models on the same dataset.
- The datasets will also be benchmarks in the future, once new models become available.
- Winners are strong candidates for the new generation of FastSim.
- A challenge creates a survey of existing models with pros and cons.
- A challenge also collects ideas and approaches for preprocessing etc.
- Previous challenges on top tagging and anomaly detection were very successful.

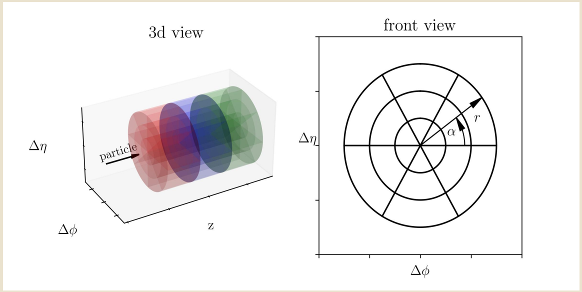
Introducing: Fast Calorimeter Simulation Challenge 2022

⇒ The main task:
$\boxed{\text{Develop a model that samples from } p(\text{shower}|E_{\text{incident}})}$
(for the dataset(s) you like.)

- Data on Zenodo: Dataset 1 Dataset 2 Dataset 3
- WebpStructureage: `https://calochallenge.github.io/homepage/`
- Code: `https://github.com/CaloChallenge/homepage/tree/main`
- Join the ML4Jets Slack workspace, and then the #calochallenge channel.
- Join the Google Mail Group.

The Structure of the Data in General

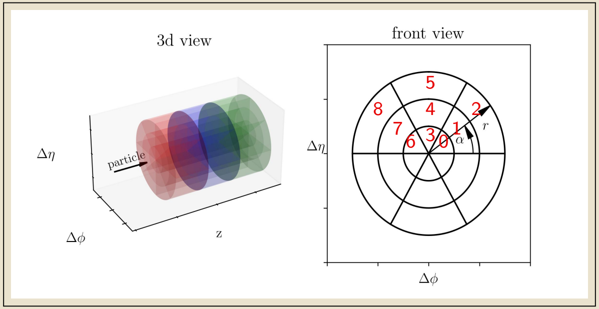
- The 3 datasets have the same format, but differ in size/complexity (“easy” $\rightarrow$“medium” $\rightarrow$“hard”).
- The geometry is based on segmented, concentric cylinders.



- The number of bins in z, r, and α is different for each dataset (see `.xml` files).

The Structure of the Data in General

- The 3 datasets have the same format, but differ in size/complexity ("easy" $\rightarrow$ "medium" $\rightarrow$ "hard").
- The geometry is based on segmented, concentric cylinders.

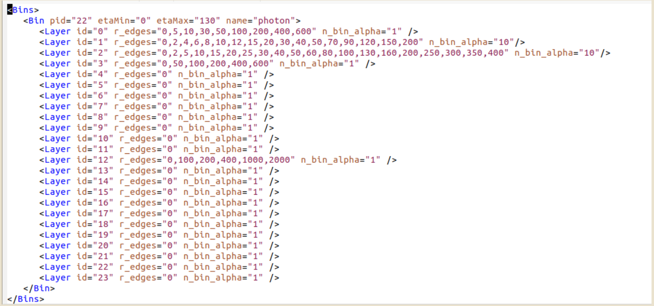


- The number of bins in z, r, and α is different for each dataset (see `.xml` files).
- In the files, all voxels are flattened, with counting order r α z.

The Structure of the Data in General

- The datasets come in `.hdf5` format. (can be read with `h5py`)
- Each file has 2 “hdf5-datasets” in it:
  “incident_energies” of shape (num_events, 1) contains E_{inc} in MeV
  “showers” of shape (num_events, num_voxels) contains the flattened energy depositions of each voxel in MeV

The dataset-specific geometry is stored in `binning_dataset_*.xml`:



```
<Bins>
   <Bin pid="22" etaMin="0" etaMax="130" name="photon">
      <Layer id="0" r_edges="0,5,10,30,50,100,200,400,600" n_bin_alpha="1" />
      <Layer id="1" r_edges="0,2,4,6,8,10,12,15,20,30,40,50,70,90,120,150,200" n_bin_alpha="10"/>
      <Layer id="2" r_edges="0,2,5,10,15,20,25,30,40,50,60,80,100,130,160,200,250,300,350,400" n_bin_alpha="10"/>
      <Layer id="3" r_edges="0,50,100,200,400,600" n_bin_alpha="1" />
      <Layer id="4" r_edges="0" n_bin_alpha="1" />
      <Layer id="5" r_edges="0" n_bin_alpha="1" />
      <Layer id="6" r_edges="0" n_bin_alpha="1" />
      <Layer id="7" r_edges="0" n_bin_alpha="1" />
      <Layer id="8" r_edges="0" n_bin_alpha="1" />
      <Layer id="9" r_edges="0" n_bin_alpha="1" />
      <Layer id="10" r_edges="0" n_bin_alpha="1" />
      <Layer id="11" r_edges="0" n_bin_alpha="1" />
      <Layer id="12" r_edges="0,100,200,400,1000,2000" n_bin_alpha="1" />
      <Layer id="13" r_edges="0" n_bin_alpha="1" />
      <Layer id="14" r_edges="0" n_bin_alpha="1" />
      <Layer id="15" r_edges="0" n_bin_alpha="1" />
      <Layer id="16" r_edges="0" n_bin_alpha="1" />
      <Layer id="17" r_edges="0" n_bin_alpha="1" />
      <Layer id="18" r_edges="0" n_bin_alpha="1" />
      <Layer id="19" r_edges="0" n_bin_alpha="1" />
      <Layer id="20" r_edges="0" n_bin_alpha="1" />
      <Layer id="21" r_edges="0" n_bin_alpha="1" />
      <Layer id="22" r_edges="0" n_bin_alpha="1" />
      <Layer id="23" r_edges="0" n_bin_alpha="1" />
   </Bin>
</Bins>
```

The Structure of the Data in General

Dataset 1 ("easy"):

- comes in 2 "flavors": photons (368-dim.) and pions (533-dim.)
- uses the ATLAS detector and is based on the dataset of AtlFast3: 2109.02551; FastCaloGAN: ATL-SOFT-PUB-2020-006

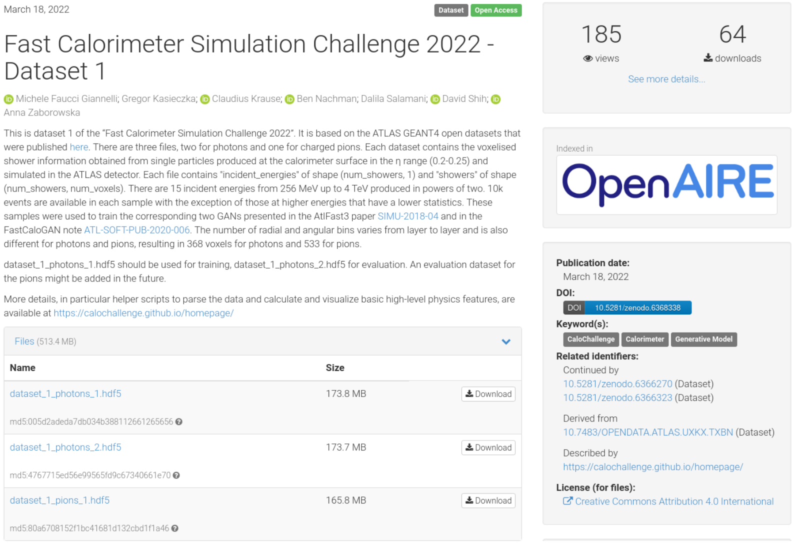
Dataset 2 ("medium"):

- electron showers (6480-dim.)
- uses detector made of alternating active (silicon) and passive (tungsten) layers, based on the Par04 GEANT4 example (with lower granularity).

Dataset 3 ("hard"):

- electron showers (40500-dim.)
- same detector as dataset 2, but voxelization to much higher granularity, based on the Par04 GEANT4 example

Dataset 1



March 18, 2022

Dataset Open Access

Fast Calorimeter Simulation Challenge 2022 - Dataset 1

Michele Faucci Giannelli; Gregor Kasieczka; Claudius Krause; Ben Nachman; Dalila Salamani; David Shih; Anna Zaborowska

This is dataset 1 of the "Fast Calorimeter Simulation Challenge 2022". It is based on the ATLAS GEANT4 open datasets that were published here. There are three files, two for photons and one for charged pions. Each dataset contains the voxelised shower information obtained from single particles produced at the calorimeter surface in the η range (0.2-0.25) and simulated in the ATLAS detector. Each file contains "incident_energies" of shape (num_showers, 1) and "showers" of shape (num_showers, num_voxels). There are 15 incident energies from 256 MeV up to 4 TeV produced in powers of two. 10k events are available in each sample with the exception of those at higher energies that have a lower statistics. These samples were used to train the corresponding two GANs presented in the AtlFast3 paper SIMU-2018-04 and in the FastCaloGAN note ATL-SOFT-PUB-2020-006. The number of radial and angular bins varies from layer to layer and is also different for photons and pions, resulting in 368 voxels for photons and 533 for pions.

dataset_1_photons_1.hdf5 should be used for training, dataset_1_photons_2.hdf5 for evaluation. An evaluation dataset for the pions might be added in the future.

More details, in particular helper scripts to parse the data and calculate and visualize basic high-level physics features, are available at https://calochallenge.github.io/homepage/

Files (513.4 MB)

Name	Size	
dataset_1_photons_1.hdf5 md5:005d2adeda7db034b388112661265656	173.8 MB	Download
dataset_1_photons_2.hdf5 md5:4767715ed56e99565fd9c67340661e70	173.7 MB	Download
dataset_1_pions_1.hdf5 md5:80a6708152f1bc41681d132cbd1f1a46	165.8 MB	Download

185 views

64 downloads

See more details...

Indexed in

OpenAIRE

Publication date:
March 18, 2022

DOI:
DOI 10.5281/zenodo.6368338

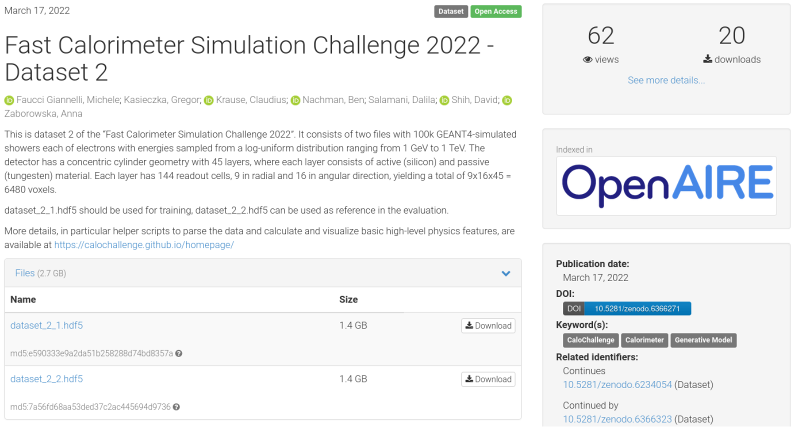
Keyword(s):
CaloChallenge Calorimeter Generative Model

Related identifiers:
Continued by
10.5281/zenodo.6366270 (Dataset)
10.5281/zenodo.6366323 (Dataset)

Derived from
10.7483/OPENDATA.ATLAS.UXKX.TXBN (Dataset)

Described by
https://calochallenge.github.io/homepage/

Dataset 2



March 17, 2022

Dataset Open Access

Fast Calorimeter Simulation Challenge 2022 - Dataset 2

Faucci Giannelli, Michele; Kasieczka, Gregor; Krause, Claudius; Nachman, Ben; Salamani, Dalila; Shih, David; Zaborowska, Anna

This is dataset 2 of the "Fast Calorimeter Simulation Challenge 2022". It consists of two files with 100k GEANT4-simulated showers each of electrons with energies sampled from a log-uniform distribution ranging from 1 GeV to 1 TeV. The detector has a concentric cylinder geometry with 45 layers, where each layer consists of active (silicon) and passive (tungsten) material. Each layer has 144 readout cells, 9 in radial and 16 in angular direction, yielding a total of 9x16x45 = 6480 voxels.

dataset_2_1.hdf5 should be used for training, dataset_2_2.hdf5 can be used as reference in the evaluation.

More details, in particular helper scripts to parse the data and calculate and visualize basic high-level physics features, are available at https://calochallenge.github.io/homepage/

Files (2.7 GB)

Name	Size	
dataset_2_1.hdf5 md5:e590333e9a2da51b258288d74bd8357a	1.4 GB	Download
dataset_2_2.hdf5 md5:7a56fd68aa53ded37c2ac445694d9736	1.4 GB	Download

62 views   20 downloads

See more details...

Indexed in OpenAIRE

Publication date:
March 17, 2022

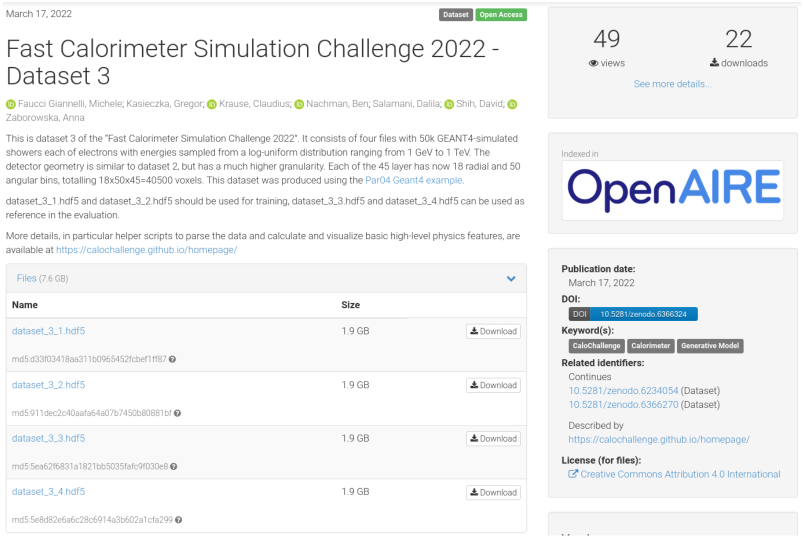
DOI:
DOI 10.5281/zenodo.6366271

Keyword(s):
CaloChallenge Calorimeter Generative Model

Related identifiers:
Continues
10.5281/zenodo.6234054 (Dataset)

Continued by
10.5281/zenodo.6366323 (Dataset)

Dataset 3



March 17, 2022

Dataset | Open Access

Fast Calorimeter Simulation Challenge 2022 - Dataset 3

Faucci Giannelli, Michele; Kasieczka, Gregor; Krause, Claudius; Nachman, Ben; Salamani, Dalila; Shih, David; Zaborowska, Anna

This is dataset 3 of the "Fast Calorimeter Simulation Challenge 2022". It consists of four files with 50k GEANT4-simulated showers each of electrons with energies sampled from a log-uniform distribution ranging from 1 GeV to 1 TeV. The detector geometry is similar to dataset 2, but has a much higher granularity. Each of the 45 layer has now 18 radial and 50 angular bins, totalling 18x50x45=40500 voxels. This dataset was produced using the Par04 Geant4 example.

dataset_3_1.hdf5 and dataset_3_2.hdf5 should be used for training, dataset_3_3.hdf5 and dataset_3_4.hdf5 can be used as reference in the evaluation.

More details, in particular helper scripts to parse the data and calculate and visualize basic high-level physics features, are available at https://calochallenge.github.io/homepage/

Files (7.6 GB)

Name	Size	
dataset_3_1.hdf5 md5:d33f03418aa311b0965452fcbef1ff87	1.9 GB	Download
dataset_3_2.hdf5 md5:911dec2c40aafa64a07b7450b80881bf	1.9 GB	Download
dataset_3_3.hdf5 md5:5ea62f6831a1821bb5035fafc9f030e8	1.9 GB	Download
dataset_3_4.hdf5 md5:5e8d82e6a6c28c6914a3b602a1cfa299	1.9 GB	Download

49 views | 22 downloads

See more details...

Indexed in

OpenAIRE

Publication date:
March 17, 2022

DOI:
DOI 10.5281/zenodo.6366324

Keyword(s):
CaloChallenge | Calorimeter | Generative Model

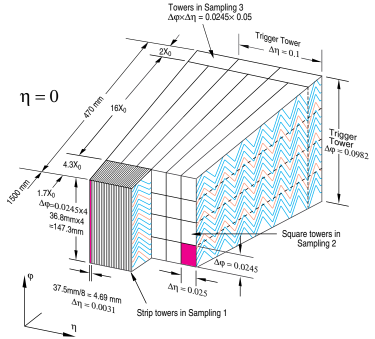
Related identifiers:
Continues
10.5281/zenodo.6234054 (Dataset)
10.5281/zenodo.6366270 (Dataset)

Described by
https://calochallenge.github.io/homepage/

License (for files):
Creative Commons Attribution 4.0 International

Dataset 1

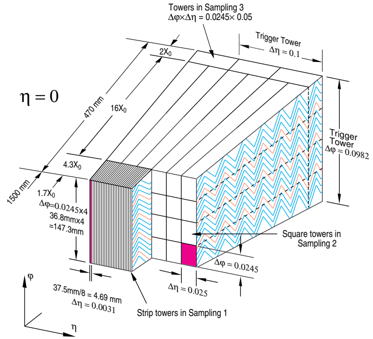
- Based on ATLAS dataset form CERN Opendata portal



Technical Design Report ATLAS, CERN, 1996

Dataset 1

- Based on ATLAS dataset form CERN Opendata portal
- photons and pions item single η region (0.2–0.25)
- 15 discrete incident energies from 256 MeV up to 4 TeV
- 10k in each sample



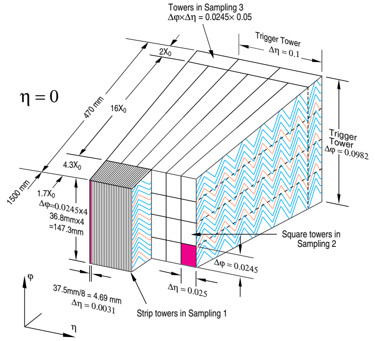
Technical Design Report ATLAS, CERN, 1996

Dataset 1

- Based on ATLAS dataset form CERN Opendata portal
- photons and pions item single η region (0.2–0.25)
- 15 discrete incident energies from 256 MeV up to 4 TeV
- 10k in each sample
- Number of radial and angular bins varies from layer to layer and is different for photons and pions, resulting in 368 voxels for photons and 533 for pions.

Layer	r edges [mm]	N bins α
PreSamplerB	0,5,10,30,50,100,200,400,600	1
EMB1	0,2,4,6,8,10,12,15,20,30,40,50,70,90,120,150,200	10
EMB2	0,2,5,10,15,20,25,30,40,50,60,80,100,130,160,200,250,300,350,400	10
EMB3	0,50,100,200,400,600	1
TileBar0	0,100,200,400,1000,2000	1

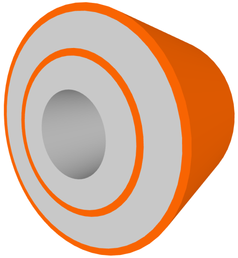
Nominal binning for photons and electrons with $0 < \eta < 1.3$ range. [ATL-SOFT-PUB-2020-006]



Technical Design Report ATLAS, CERN, 1996

Datasets 2 & 3

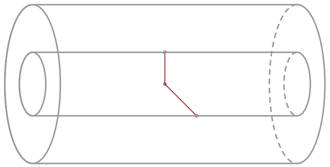
- Data simulated with `Par04 Geant4 example` released in Geant4 11.0
- Simplistic geometry, collider-style concentric cylinders with up to two interchanging materials



par04.web.cern.ch

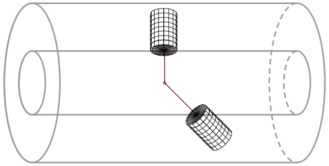
Datasets 2 & 3

- Data simulated with `Par04 GEANT4 example` released in Geant4 11.0
- Simplistic geometry, collider-style concentric cylinders with up to two interchanging materials
- Particle direction and position is measured at the entrance to calorimeter, so scoring of energy deposits is done relative to the particle direction



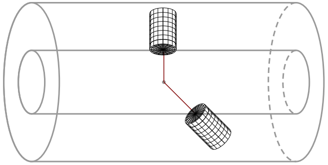
Datasets 2 & 3

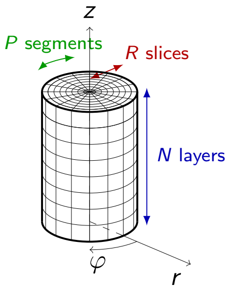
- Data simulated with `Par04` Geant4 example released in Geant4 11.0
- Simplistic geometry, collider-style concentric cylinders with up to two interchanging materials
- Particle direction and position is measured at the entrance to calorimeter, so scoring of energy deposits is done relative to the particle direction
- Similar 'pictures' are obtained independently on angle of the incident particle



Datasets 2 & 3

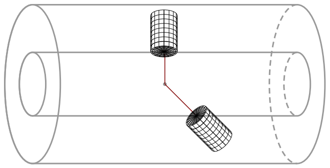
- Data simulated with `Par04 Geant4 example` released in Geant4 11.0
- Simplistic geometry, collider-style concentric cylinders with up to two interchanging materials
- Particle direction and position is measured at the entrance to calorimeter, so scoring of energy deposits is done relative to the particle direction
- Similar 'pictures' are obtained independently on angle of the incident particle
- Granularity of shower deposition is configurable (dataset 2 is less granular than dataset 3)

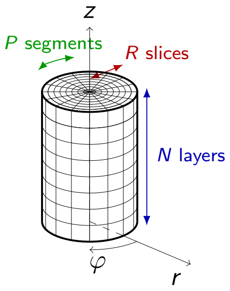




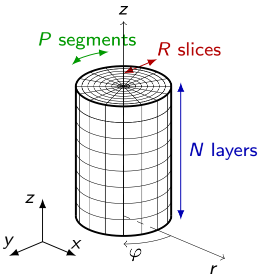
Datasets 2 & 3

- Data simulated with `Par04 GEANT4 example` released in Geant4 11.0
- Simplistic geometry, collider-style concentric cylinders with up to two interchanging materials
- Particle direction and position is measured at the entrance to calorimeter, so scoring of energy deposits is done relative to the particle direction
- Similar 'pictures' are obtained independently on angle of the incident particle
- Granularity of shower deposition is configurable (dataset 2 is less granular than dataset 3)
- Electons with log-uniform energy spectrum, from 1 GeV to 1 TeV, perpendicular to detector (for this challenge)

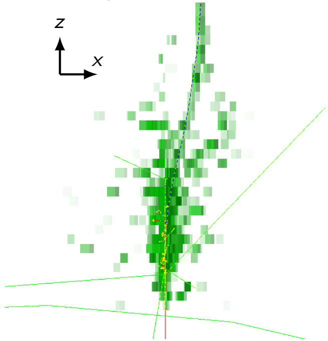




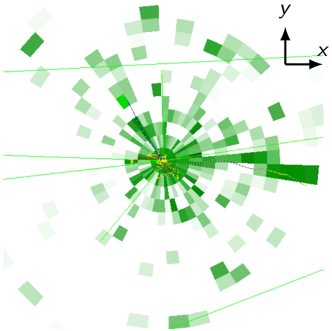
Datasets 2 & 3



10 GeV e$^-$, dataset 3

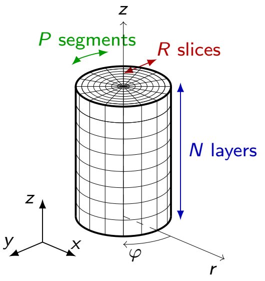


10 GeV e$^-$, dataset 3

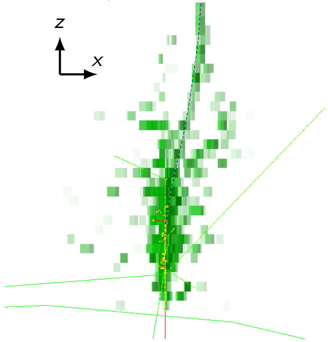


- Par04 example defines 0.3 mm Si and 1.4 mm W layers

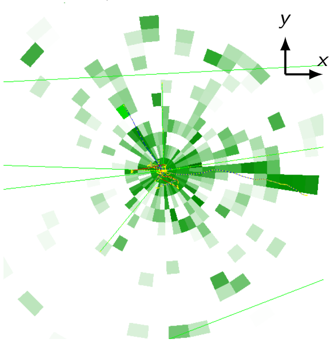
Datasets 2 & 3



10 GeV e$^-$, dataset 3

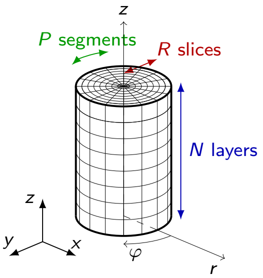


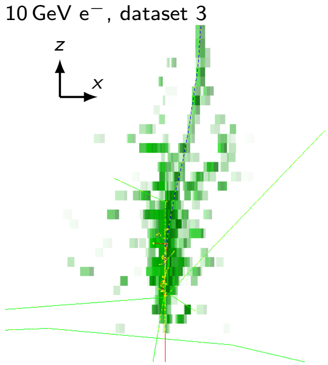
10 GeV e$^-$, dataset 3

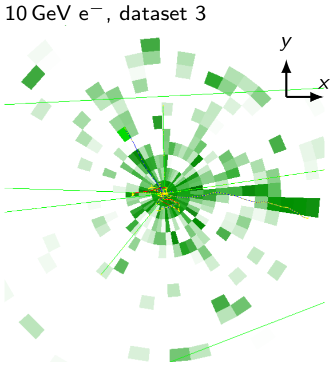


- Par04 example defines 0.3 mm Si and 1.4 mm W layers
- Dataset 3:
  - Readout granularity is $\Delta r \times \Delta\varphi \times \Delta z = 2.3\,\text{mm} \times \frac{2\pi}{50} \times 3.4\,\text{mm}$ (with $\Delta r \approx 0.25\,R_M$ and $\Delta z \approx 0.6\,X_0$)
  - Number of readout cells is $R \times P \times N = 18\text{x}50\text{x}45$ aiming for 95% containment of 1 TeV particles
  - Open access dataset for different incident angles, for SiW (and scintillator-Pb) available at 10.5281/zenodo.6082201

Datasets 2 & 3

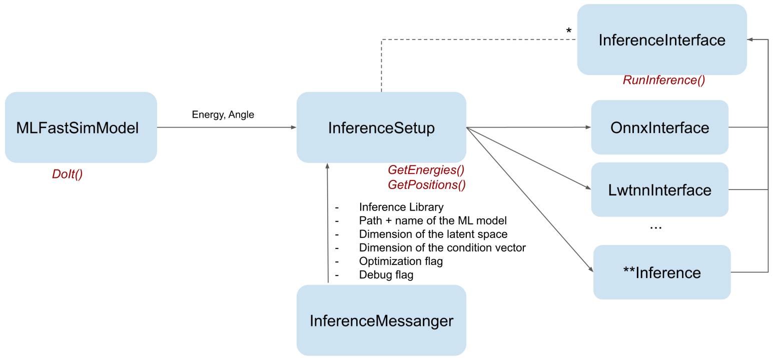






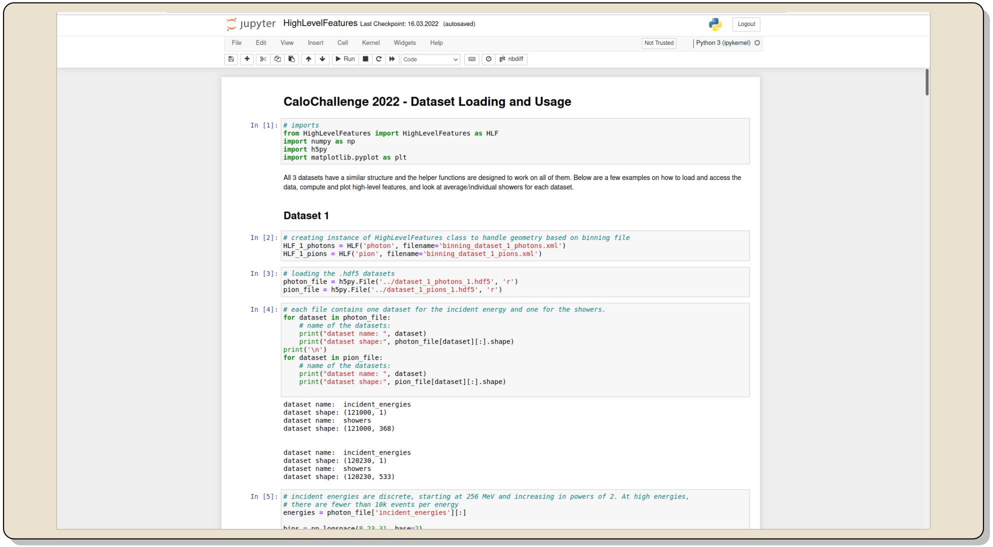
- Par04 example defines 0.3 mm Si and 1.4 mm W layers
- Dataset 3:
  - Readout granularity is $\Delta r \times \Delta\varphi \times \Delta z = 2.3\,\mathrm{mm} \times \frac{2\pi}{50} \times 3.4\,\mathrm{mm}$ (with $\Delta r \approx 0.25\,R_M$ and $\Delta z \approx 0.6\,X_0$)
  - Number of readout cells is $R \times P \times N = 18\mathrm{x}50\mathrm{x}45$ aiming for 95% containment of 1 TeV particles
  - Open access dataset for different incident angles, for SiW (and scintillator-Pb) available at 10.5281/zenodo.6082201
- Dataset 2:
  - Readout granularity is $\Delta r \times \Delta\varphi \times \Delta z = 4.7\,\mathrm{mm} \times \frac{2\pi}{16} \times 3.4\,\mathrm{mm}$ (with $\Delta r \approx 0.5\,R_M$ and $\Delta z \approx 0.6\,X_0$)
  - Number of readout cells is $R \times P \times N = 9\mathrm{x}16\mathrm{x}45$

Beyond CaloChallenge: Geant4 Par04 example: inference within C++ framework



- Fast simulation with ML within Geant4
- Example demonstrates how to incorporate inference libraries (ONNX Runtime, LWTNN, soon also pyTorch)
- Allows to calculate fairly speed-up of simulation (which will depend on granularity)

The Structure of the Data and High–Level Features — Code



jupyter HighLevelFeatures Last Checkpoint: 16.03.2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
nbdiff

CaloChallenge 2022 - Dataset Loading and Usage

In [1]:
```
# imports
from HighLevelFeatures import HighLevelFeatures as HLF
import numpy as np
import h5py
import matplotlib.pyplot as plt
```

All 3 datasets have a similar structure and the helper functions are designed to work on all of them. Below are a few examples on how to load and access the data, compute and plot high-level features, and look at average/individual showers for each dataset.

Dataset 1

In [2]:
```
# creating instance of HighLevelFeatures class to handle geometry based on binning file
HLF_1_photons = HLF('photon', filename='binning_dataset_1_photons.xml')
HLF_1_pions = HLF('pion', filename='binning_dataset_1_pions.xml')
```

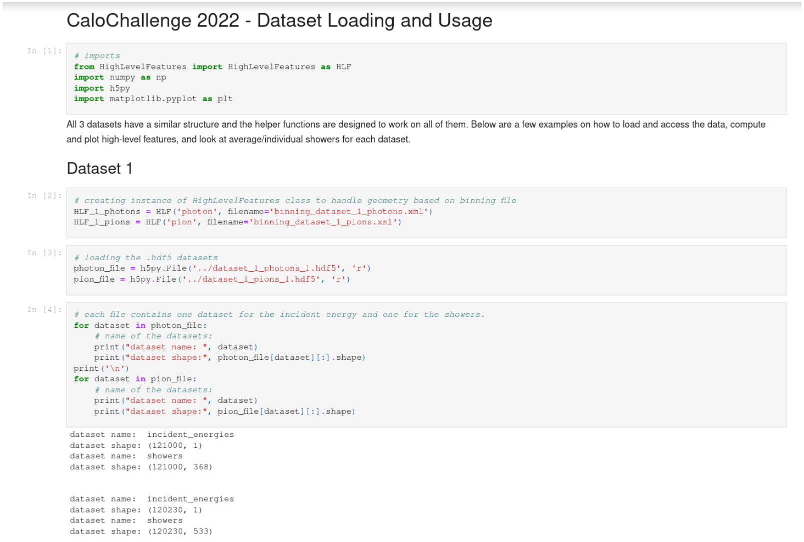
In [3]:
```
# loading the .hdf5 datasets
photon_file = h5py.File('../dataset_1_photons_1.hdf5', 'r')
pion_file = h5py.File('../dataset_1_pions_1.hdf5', 'r')
```

In [4]:
```
# each file contains one dataset for the incident energy and one for the showers.
for dataset in photon_file:
    # name of the datasets:
    print("dataset name: ", dataset)
    print("dataset shape:", photon_file[dataset][:].shape)
print('\n')
for dataset in pion_file:
    # name of the datasets:
    print("dataset name: ", dataset)
    print("dataset shape:", pion_file[dataset][:].shape)
```

```
dataset name:  incident_energies
dataset shape: (121000, 1)
dataset name:  showers
dataset shape: (121000, 368)


dataset name:  incident_energies
dataset shape: (120230, 1)
dataset name:  showers
dataset shape: (120230, 533)
```

In [5]:
```
# incident energies are discrete, starting at 256 MeV and increasing in powers of 2. At high energies,
# there are fewer than 10k events per energy
energies = photon_file['incident_energies'][:]
```

The Structure of the Data



CaloChallenge 2022 - Dataset Loading and Usage

In [1]:
```
# imports
from HighLevelFeatures import HighLevelFeatures as HLF
import numpy as np
import h5py
import matplotlib.pyplot as plt
```

All 3 datasets have a similar structure and the helper functions are designed to work on all of them. Below are a few examples on how to load and access the data, compute and plot high-level features, and look at average/individual showers for each dataset.

Dataset 1

In [2]:
```
# creating instance of HighLevelFeatures class to handle geometry based on binning file
HLF_1_photons = HLF('photon', filename='binning_dataset_1_photons.xml')
HLF_1_pions = HLF('pion', filename='binning_dataset_1_pions.xml')
```

In [3]:
```
# loading the .hdf5 datasets
photon_file = h5py.File('../dataset_1_photons_1.hdf5', 'r')
pion_file = h5py.File('../dataset_1_pions_1.hdf5', 'r')
```

In [4]:
```
# each file contains one dataset for the incident energy and one for the showers.
for dataset in photon_file:
    # name of the datasets:
    print("dataset name: ", dataset)
    print("dataset shape:", photon_file[dataset][:].shape)
print('\n')
for dataset in pion_file:
    # name of the datasets:
    print("dataset name: ", dataset)
    print("dataset shape:", pion_file[dataset][:].shape)
```

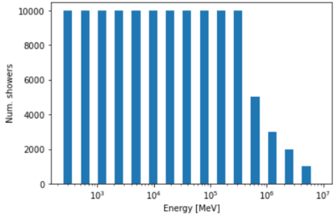
```
dataset name:  incident_energies
dataset shape: (121000, 1)
dataset name:  showers
dataset shape: (121000, 368)


dataset name:  incident_energies
dataset shape: (120230, 1)
dataset name:  showers
dataset shape: (120230, 533)
```

HighLevelFeatures.ipynb

The Structure of the Data

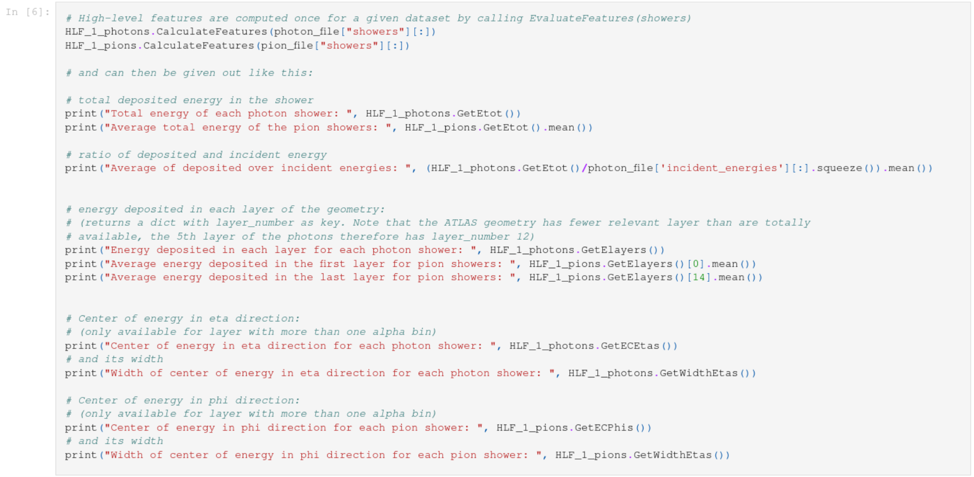
In [5]:

```
# incident energies are discrete, starting at 256 MeV and increasing in powers of 2. At high energies,
# there are fewer than 10k events per energy
energies = photon_file['incident_energies'][:]

bins = np.logspace(8,23,31, base=2)
plt.hist(energies, bins=bins)
plt.xscale('log')
plt.xlabel('Energy [MeV]')
plt.ylabel('Num. showers')
plt.show()
```



HighLevelFeatures.ipynb

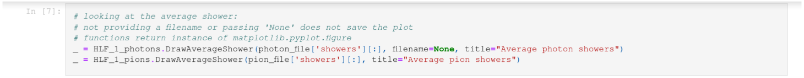
High–Level Features



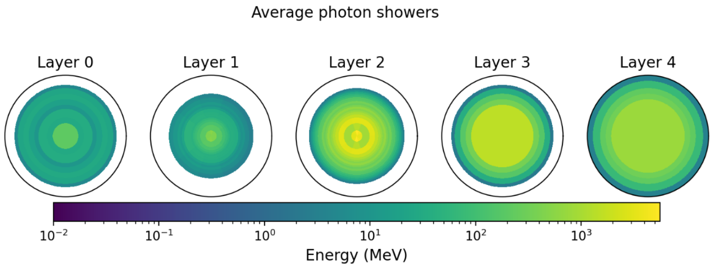
```
In [6]:
# High-level features are computed once for a given dataset by calling EvaluateFeatures(showers)
HLF_1_photons.CalculateFeatures(photon_file["showers"][:])
HLF_1_pions.CalculateFeatures(pion_file["showers"][:])

# and can then be given out like this:

# total deposited energy in the shower
print("Total energy of each photon shower: ", HLF_1_photons.GetEtot())
print("Average total energy of the pion showers: ", HLF_1_pions.GetEtot().mean())

# ratio of deposited and incident energy
print("Average of deposited over incident energies: ", (HLF_1_photons.GetEtot()/photon_file['incident_energies'][:].squeeze()).mean())


# energy deposited in each layer of the geometry:
# (returns a dict with layer_number as key. Note that the ATLAS geometry has fewer relevant layer than are totally
# available, the 5th layer of the photons therefore has layer_number 12)
print("Energy deposited in each layer for each photon shower: ", HLF_1_photons.GetElayers())
print("Average energy deposited in the first layer for pion showers: ", HLF_1_pions.GetElayers()[0].mean())
print("Average energy deposited in the last layer for pion showers: ", HLF_1_pions.GetElayers()[14].mean())


# Center of energy in eta direction:
# (only available for layer with more than one alpha bin)
print("Center of energy in eta direction for each photon shower: ", HLF_1_photons.GetECEtas())
# and its width
print("Width of center of energy in eta direction for each photon shower: ", HLF_1_photons.GetWidthEtas())

# Center of energy in phi direction:
# (only available for layer with more than one alpha bin)
print("Center of energy in phi direction for each pion shower: ", HLF_1_pions.GetECPhis())
# and its width
print("Width of center of energy in phi direction for each pion shower: ", HLF_1_pions.GetWidthEtas())
```

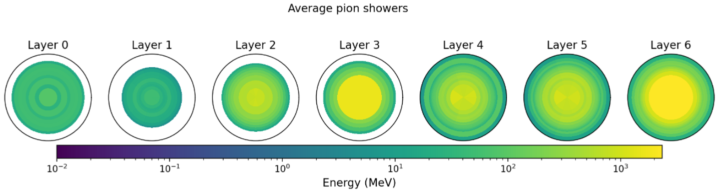
HighLevelFeatures.ipynb

High–Level Features



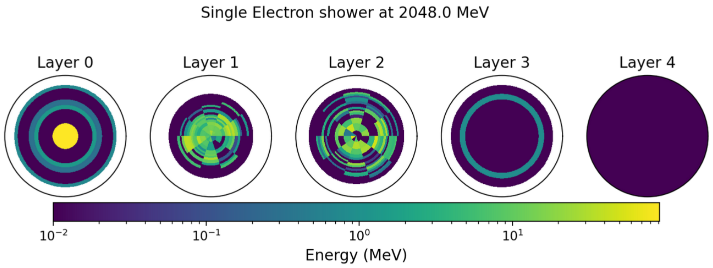
```
In [7]:
# looking at the average shower:
# not providing a filename or passing 'None' does not save the plot
# functions return instance of matplotlib.pyplot.figure
_ = HLF_1_photons.DrawAverageShower(photon_file['showers'][:], filename=None, title="Average photon showers")
_ = HLF_1_pions.DrawAverageShower(pion_file['showers'][:], title="Average pion showers")
```

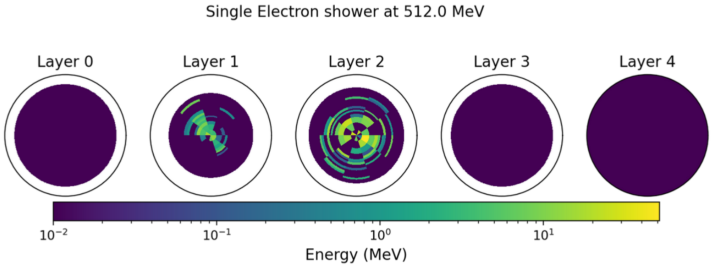




HighLevelFeatures.ipynb

High–Level Features



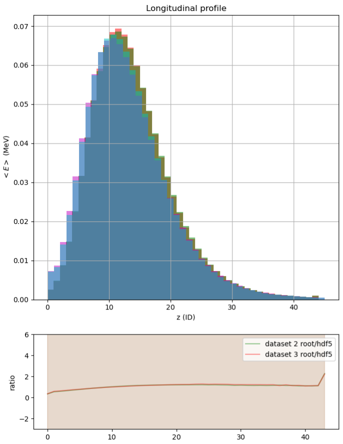


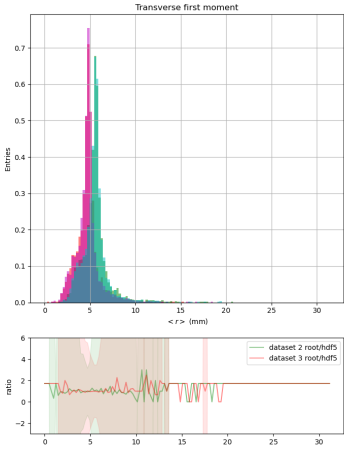
HighLevelFeatures.ipynb

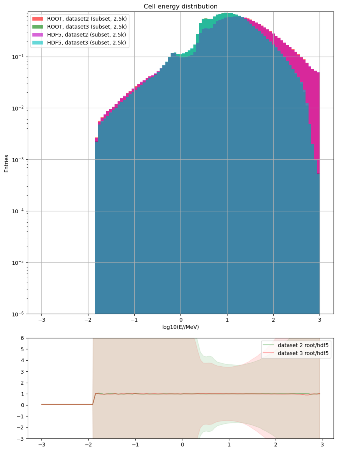
High–Level Features

Work in progress to add:

- shower shape features (profiles, first, second moments),
- cell energy distribution.







Demonstrator plots, to be implemented for CaloChallenge

The Structure of the Data and High–Level Features

Points for discussion:

- Any other high-level features needed / relevant?
- Any other histograms needed?
- Any other plots / visualizations wished for?

$\Rightarrow$ Send a pull-request if you have some!

Evaluating the Models

The surrogates should be fast and faithful!

We will be looking at:
$\Rightarrow$ Sampling time, (training time, memory usage)

Evaluating the Models

The surrogates should be fast and faithful!

We will be looking at:

$\Rightarrow$ Sampling time, (training time, memory usage)

$\Rightarrow$ Histograms of high-level features, and their separation power:

$\langle S^2 \rangle = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_{1,i}-h_{2,i})^2}{h_{1,i}+h_{2,i}}$ Diefenbacher et al. 2009.03796

Evaluating the Models

The surrogates should be fast and faithful!

We will be looking at:

$\Rightarrow$ Sampling time, (training time, memory usage)

$\Rightarrow$ Histograms of high-level features, and their separation power:

$$\langle S^2 \rangle = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_{1,i}-h_{2,i})^2}{h_{1,i}+h_{2,i}}$$

Diefenbacher et al. 2009.03796

$\Rightarrow$ Interpolation capabilities of datasets 1: $\langle S^2 \rangle$ at an E_{inc} left out in training.

Evaluating the Models

The surrogates should be fast and faithful!

We will be looking at:

$\Rightarrow$ Sampling time, (training time, memory usage)

$\Rightarrow$ Histograms of high-level features, and their separation power:

$\langle S^2 \rangle = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_{1,i} - h_{2,i})^2}{h_{1,i} + h_{2,i}}$ Diefenbacher et al. 2009.03796

$\Rightarrow$ Interpolation capabilities of datasets 1: $\langle S^2 \rangle$ at an E_{inc} left out in training.

$\Rightarrow$ A binary classifier to distinguish samples from GEANT4, based on high-level features.

Evaluating the Models

The surrogates should be fast and faithful!

We will be looking at:

$\Rightarrow$ Sampling time, (training time, memory usage)

$\Rightarrow$ Histograms of high-level features, and their separation power:

$$\langle S^2 \rangle = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_{1,i}-h_{2,i})^2}{h_{1,i}+h_{2,i}}$$

Diefenbacher et al. 2009.03796

$\Rightarrow$ Interpolation capabilities of datasets 1: $\langle S^2 \rangle$ at an E_{inc} left out in training.

$\Rightarrow$ A binary classifier to distinguish samples from GEANT4, based on high-level features.

$\Rightarrow$ A binary classifier to distinguish samples from GEANT4, based on voxel information.

Evaluating the Models

The surrogates should be fast and faithful!

We will be looking at:

⇒ Sampling time, (training time, memory usage)

⇒ Histograms of high-level features, and their separation power:

$$\langle S^2 \rangle = \frac{1}{2} \sum_{i=1}^{n_{\text{bins}}} \frac{(h_{1,i} - h_{2,i})^2}{h_{1,i} + h_{2,i}}$$

Diefenbacher et al. 2009.03796

⇒ Interpolation capabilities of datasets 1: $\langle S^2 \rangle$ at an E_{inc} left out in training.

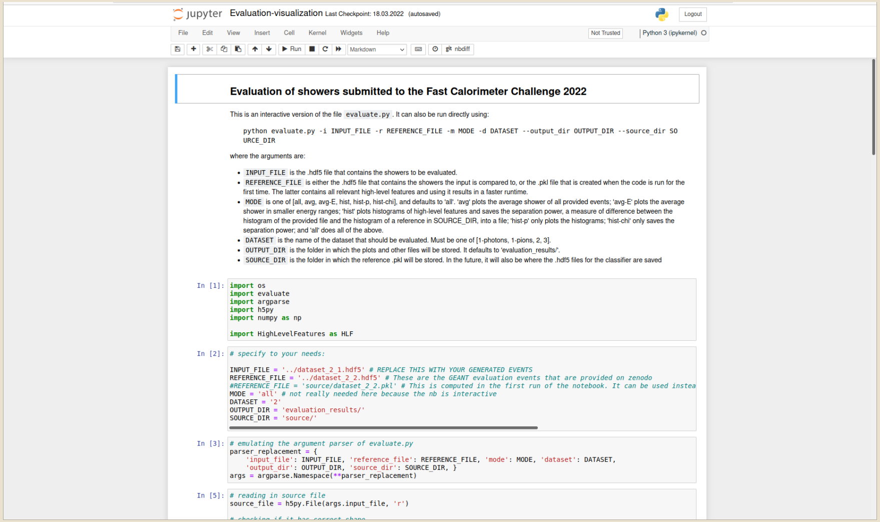
⇒ A binary classifier to distinguish samples from GEANT4, based on high-level features.

⇒ A binary classifier to distinguish samples from GEANT4, based on voxel information.

Note:

- Almost all of these require the distributions of E_{inc} to agree.
- Data needs to be written in the same `.hdf5` format as the training data.

Evaluating the Models — Code


jupyter Evaluation-visualization Last Checkpoint: 18.03.2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
nbdiff

Evaluation of showers submitted to the Fast Calorimeter Challenge 2022

This is an interactive version of the file `evaluate.py`. It can also be run directly using:

```
python evaluate.py -i INPUT_FILE -r REFERENCE_FILE -m MODE -d DATASET --output_dir OUTPUT_DIR --source_dir SO
URCE_DIR
```

where the arguments are:

- `INPUT_FILE` is the .hdf5 file that contains the showers to be evaluated.
- `REFERENCE_FILE` is either the .hdf5 file that contains the showers the input is compared to, or the .pkl file that is created when the code is run for the first time. The latter contains all relevant high-level features and using it results in a faster runtime.
- `MODE` is one of [all, avg, avg-E, hist, hist-p, hist-chi], and defaults to 'all'. 'avg' plots the average shower of all provided events; 'avg-E' plots the average shower in smaller energy ranges; 'hist' plots histograms of high-level features and saves the separation power, a measure of difference between the histogram of the provided file and the histogram of a reference in SOURCE_DIR, into a file; 'hist-p' only plots the histograms; 'hist-chi' only saves the separation power; and 'all' does all of the above.
- `DATASET` is the name of the dataset that should be evaluated. Must be one of [1-photons, 1-pions, 2, 3].
- `OUTPUT_DIR` is the folder in which the plots and other files will be stored. It defaults to 'evaluation_results/'.
- `SOURCE_DIR` is the folder in which the reference .pkl will be stored. In the future, it will also be where the .hdf5 files for the classifier are saved

In [1]:
```python
import os
import evaluate
import argparse
import h5py
import numpy as np

import HighLevelFeatures as HLF
```

In [2]:
```python
# specify to your needs:

INPUT_FILE = '../dataset_2_1.hdf5' # REPLACE THIS WITH YOUR GENERATED EVENTS
REFERENCE_FILE = '../dataset_2_2.hdf5' # These are the GEANT evaluation events that are provided on zenodo
#REFERENCE_FILE = 'source/dataset_2_2.pkl' # This is computed in the first run of the notebook. It can be used instea
MODE = 'all' # not really needed here because the nb is interactive
DATASET = '2'
OUTPUT_DIR = 'evaluation_results/'
SOURCE_DIR = 'source/'
```

In [3]:
```python
# emulating the argument parser of evaluate.py
parser_replacement = {
    'input_file': INPUT_FILE, 'reference_file': REFERENCE_FILE, 'mode': MODE, 'dataset': DATASET,
    'output_dir': OUTPUT_DIR, 'source_dir': SOURCE_DIR, }
args = argparse.Namespace(**parser_replacement)
```

In [5]:
```python
# reading in source file
source_file = h5py.File(args.input_file, 'r')

# checking if it has correct shape
```

Evaluating the Models

We don't expect to have a single clear winner!

Instead, we are looking forward to a diversity of approaches with a plethora of new ideas.

Points for discussion:

- Any other high-level features for histograms / classifier?
- What kind of preprocessing should be used for low-level classifier?
- Any other metrics?

Looking Ahead

- We will add more plot features.
- We will add the code for the classifiers and other missing metrics.
- There will be a dedicated session at ML4Jets @ Rutgers in October/November this year. Please send us your samples ahead of time (tbd when), so we can run them through our common pipeline.
- There will be a summary paper at the very end.

Fast Calorimeter Simulation Challenge 2022

- Webpage: `https://calochallenge.github.io/homepage/`
- Code: `https://github.com/CaloChallenge/homepage/tree/main`

- Data on Zenodo: Dataset 1   Dataset 2   Dataset 3

- Join the ML4Jets Slack workspace, and then the #calochallenge channel.

- Join the Google Mail Group.

- CERN EP-RD fast sim group organizes (probably monthly) in-person discussion group @ CERN. Join mattermost channel for announcements: EP-RD Software team's Mattermost channel "calochallengecern".

Calorimeter in the Open Data Detector

Erica Brondolin, Dalila Salamani, and
Anna Zaborowska

Learning to Discover, Workshop on Generative Models,
April 26th 2022

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (CERN-OPEN-2018-006)

Open Data Detector

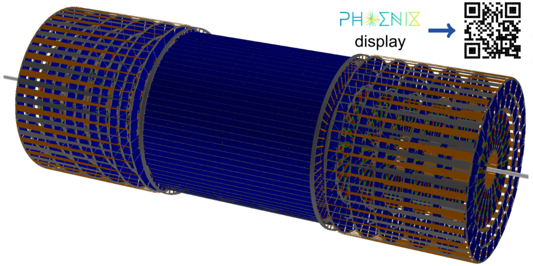
DOI 10.5281/zenodo.6445359

GitLab

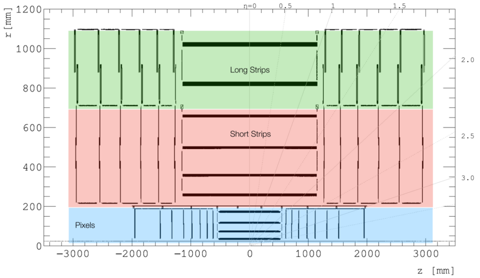
OpenDataDetector (ODD)

is attempted to provide a template (HL-)LHC style particle detector for algorithm research and development.

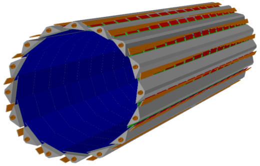
cern.ch/go/Q897



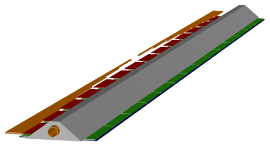
Implementation of tracking system:



Innermost Pixel barrel layer

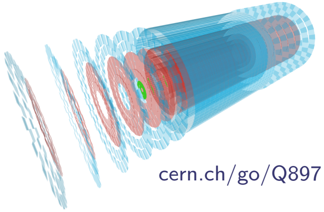


Stave with sensors pointing inwards



Figures from ACAT 2021 poster (Paul Gessinger, Andreas Salzburger, Joana Niermann)

Open Data Detector - why?



cern.ch/go/Q897

- Experiment independent
- Free (and easy!) to use, open access, public data
- Software benchmark detector
- Can be used to set up challenges (fast simulation, reconstruction, . . .)
- Implemented tracking system is an evolution of detector used for Tracking Machine Learning Challenges
- Kick-off meeting to discuss calorimeter system is **tomorrow** (please register via indico to get informed of updates)

How do we want to use calorimeter in ODD?

Fast simulation:

- Study and benchmarking of fast simulation techniques (classical and machine-learning based)
- Follow-up of Calo Challenge with ODD calorimeter data

Reconstruction:

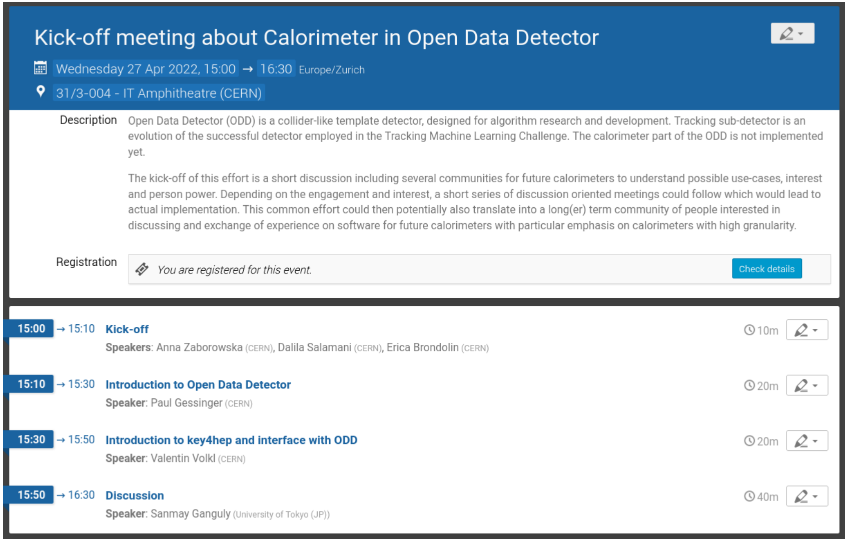
- Study and benchmarking of reconstruction algorithms

We would like to get to know your ideas!

- Tracker $+$ calorimeter $\Rightarrow$ particle flow techniques
- ...

Step-by-step plan

1. Collect ideas how to use it, design ideas and requirements
2. Gather information on available manpower
3. Design the calorimeter (EM and H part)
4. Implement it in DD4hep
5. Run simulation
6. Once satisfying - freeze and document versions, produce high stat data
7. Publish and use!



Kick-off meeting about Calorimeter in Open Data Detector

Wednesday 27 Apr 2022, 15:00 → 16:30 Europe/Zurich

31/3-004 - IT Amphitheatre (CERN)

Description: Open Data Detector (ODD) is a collider-like template detector, designed for algorithm research and development. Tracking sub-detector is an evolution of the successful detector employed in the Tracking Machine Learning Challenge. The calorimeter part of the ODD is not implemented yet.

The kick-off of this effort is a short discussion including several communities for future calorimeters to understand possible use-cases, interest and person power. Depending on the engagement and interest, a short series of discussion oriented meetings could follow which would lead to actual implementation. This common effort could then potentially also translate into a long(er) term community of people interested in discussing and exchange of experience on software for future calorimeters with particular emphasis on calorimeters with high granularity.

Registration: *You are registered for this event.*

15:00 → 15:10 **Kick-off** 10m
Speakers: Anna Zaborowska (CERN), Dalila Salamani (CERN), Erica Brondolin (CERN)

15:10 → 15:30 **Introduction to Open Data Detector** 20m
Speaker: Paul Gessinger (CERN)

15:30 → 15:50 **Introduction to key4hep and interface with ODD** 20m
Speaker: Valentin Volkl (CERN)

15:50 → 16:30 **Discussion** 40m
Speaker: Sanmay Ganguly (University of Tokyo (JP))

indico.cern.ch/event/1147195/