Fast Calorimeter Simulation Challenge 2022

Claudius Krause

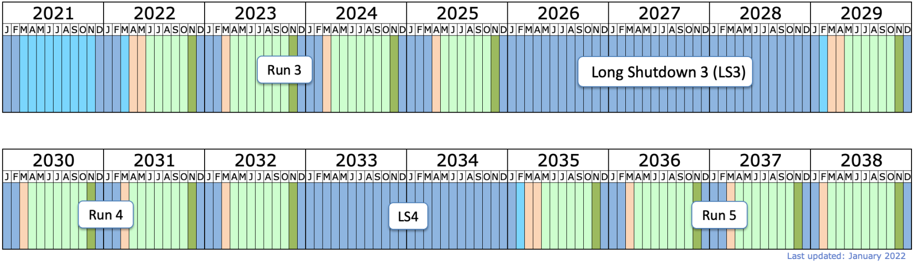
Rutgers, The State University of New Jersey

April 11, 2022

With
Michele Faucci Giannelli, Gregor Kasieczka, Ben Nachman, Dalila Salamani, David Shih and Anna Zaborowska

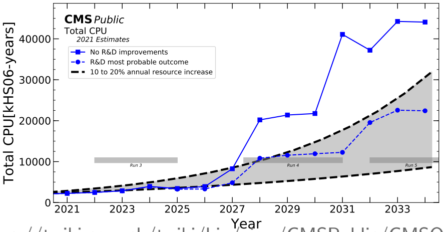
https://calochallenge.github.io/homepage/

The LHC will need a lot of computing ressources.

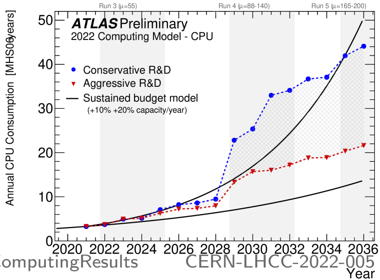


Shutdown/Technical stop
Protons physics
Ions
Commissioning with beam
Hardware commissioning/magnet training

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

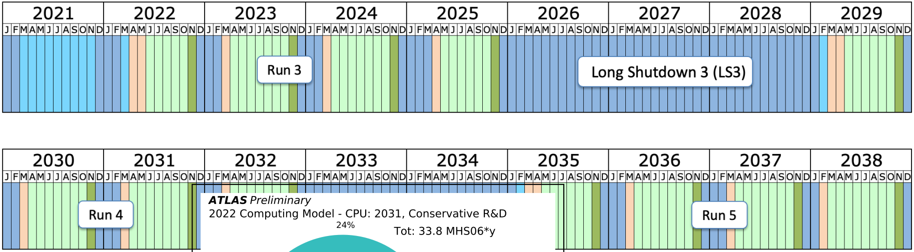


https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

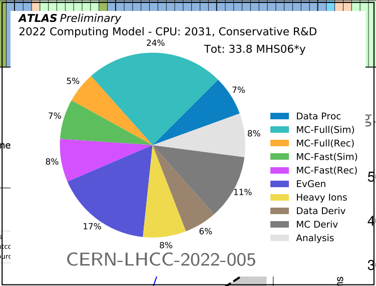


CERN-LHCC-2022-005

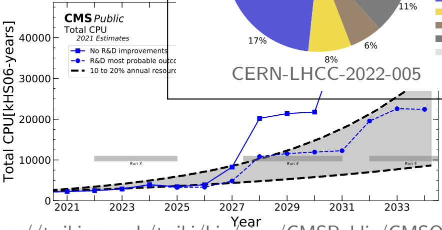
The LHC will need a lot of computing ressources.



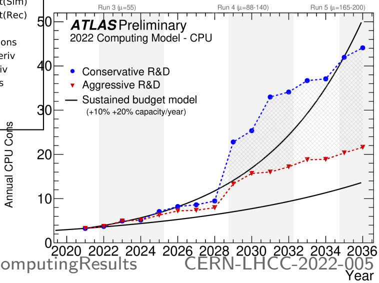
https://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm



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:

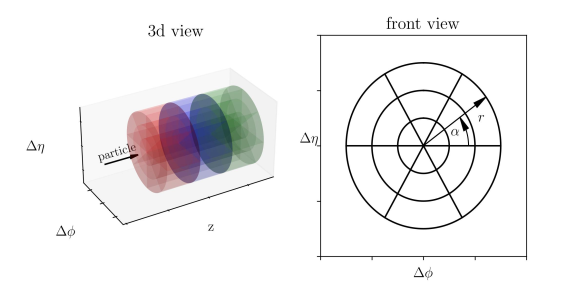
$\boxed{\text{Develop a model that samples from } p(\text{shower}|E_{\text{incident}})}$

(for the dataset(s) you like.)

- Data on Zenodo: Dataset 1 Dataset 2 Dataset 3
- Webpage: `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

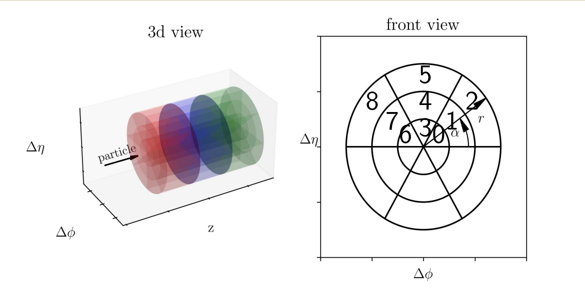
- The 3 datasets have the same format, but differ in size/complexity ("easy" $\rightarrow$ "medium" $\rightarrow$ "hard").
- The geometry is based on segmented, concentric cylinders.



- 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 $r \quad \alpha \quad z$.

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`:

```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.

Dataset 3 ("hard"):

- electron showers (40500-dim.)
- same detector as dataset 2, but voxelization to much higher 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]:
```python
# 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]:
```python
# 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]:
```python
# 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]:
```python
# 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]:
```python
# 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)
```

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:

⇒ 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

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:

- ⇒ 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.

Evaluating the Models

The surrogates should be fast and faithful!

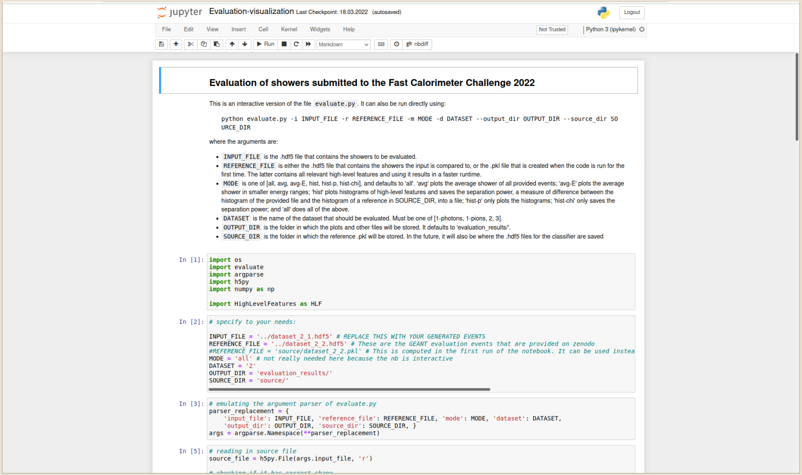
We will be looking at:

- ⇒ 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)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

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')
```

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

- Anna Zaborowska organized an in-person discussion group @ CERN. First meeting is tomorrow, 4/12, at 11 in 42/3-032. More details will be on EP-RD Software team's Mattermost channel "calochallengecern".
- 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.
- If you are at CERN, check out tomorrow's meeting.