Level up your performance calculation
of the fast shower simulation model

Anna Zaborowska

CERN

ML4Jets2023

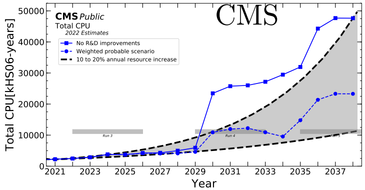
This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (CERN-OPEN-2018-006)

Why to use parameterisation / fast(er) simulation?

CERN-CMS-NOTE-2022-008

To speed-up simulation in order to generate more data within same CPU time:

- to fit within available computing resources;
- to provide sufficient amount of simulation data for comparison with the experimental data;

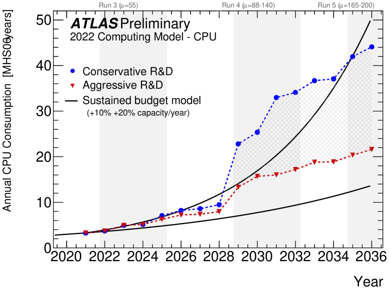


Why to use parameterisation / fast(er) simulation?

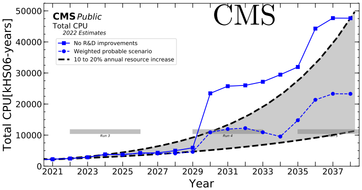
To speed-up simulation in order to generate more data within same CPU time:

- to fit within available computing resources;
- to provide sufficient amount of simulation data for comparison with the experimental data;

CERN-LHCC-2022-005



CERN-CMS-NOTE-2022-008

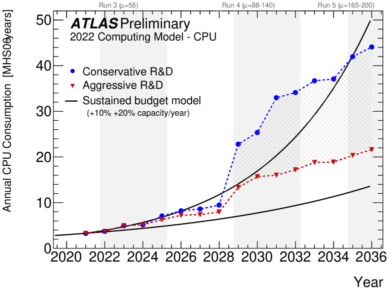


Why to use parameterisation / fast(er) simulation?

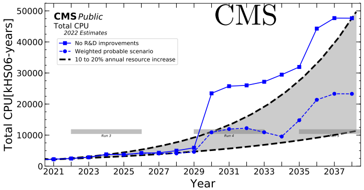
To speed-up simulation in order to generate more data within same CPU time:

- to fit within available computing resources;
- to provide sufficient amount of simulation data for comparison with the experimental data;

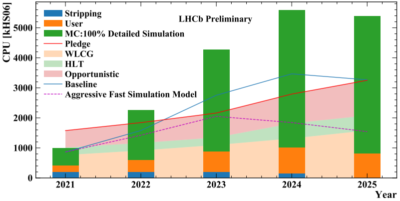
CERN-LHCC-2022-005



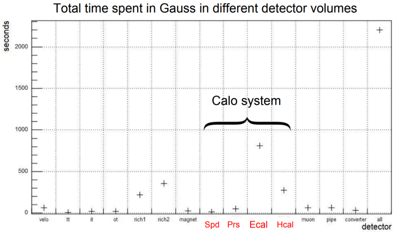
CERN-CMS-NOTE-2022-008



LHCb-FIGURE-2019-01



LHCb-TALK-2018-349



CPU time in calorimeter system: ~ 53%

Fast simulation in the experiments

While for full simulation GEANT4 is a standard toolkit,

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

Fast simulation in the experiments

While for full simulation GEANT4 is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment. ⟶ CMS and LHCb use fast simulation hooks offered by GEANT4

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

⟶ CMS and LHCb use fast simulation hooks offered by GEANT4

⟶ ATLAS is investigating a move from custom framework to Geant4

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

⟶ CMS and LHCb use fast simulation hooks offered by GEANT4
⟶ ATLAS is investigating a move from custom framework to Geant4
⟶ Easy to include in any existing G4 application

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

⟶ CMS and LHCb use fast simulation hooks offered by GEANT4
⟶ ATLAS is investigating a move from custom framework to Geant4
⟶ Easy to include in any existing G4 application

Use of cell-level data implies experiment-specific models, not every experiment can afford that, but it is probably the fastest (can they offer sufficient accuracy?).

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

⟶ CMS and LHCb use fast simulation hooks offered by GEANT4
⟶ ATLAS is investigating a move from custom framework to `Geant4`
⟶ Easy to include in any existing G4 application

Use of cell-level data implies experiment-specific models, not every experiment can afford that, but it is probably the fastest (can they offer sufficient accuracy?).

Use of granular voxelization and point clouds can be transferable, and is easier if fast simulation is performed fully inside GEANT4, from taking over particle transportation, via inference, and deposition of generated showers. ⟶ reuse of tools.

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

⟶ CMS and LHCb use fast simulation hooks offered by GEANT4
⟶ ATLAS is investigating a move from custom framework to Geant4
⟶ Easy to include in any existing G4 application

Use of cell-level data implies experiment-specific models, not every experiment can afford that, but it is probably the fastest (can they offer sufficient accuracy?).

Use of granular voxelization and point clouds can be transferable, and is easier if fast simulation is performed fully inside GEANT4, from taking over particle transportation, via inference, and deposition of generated showers. ⟶ reuse of tools.

⟶ LHCb already has an implementation of a Par04 (=CaloChallenge d2&3) setup

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.

⟶ CMS and LHCb use fast simulation hooks offered by GEANT4
⟶ ATLAS is investigating a move from custom framework to Geant4
⟶ Easy to include in any existing G4 application

Use of cell-level data implies experiment-specific models, not every experiment can afford that, but it is probably the fastest (can they offer sufficient accuracy?).

Use of granular voxelization and point clouds can be transferable, and is easier if fast simulation is performed fully inside GEANT4, from taking over particle transportation, via inference, and deposition of generated showers. ⟶ reuse of tools.

⟶ LHCb already has an implementation of a Par04 (=CaloChallenge d2&3) setup

True performance measurements possible once ML model is in production software. (see talk by M.Giannelli at CaloChallenge workshop)

Fast simulation in the experiments

While for full simulation GEANT4 A SIMULATION TOOLKIT is a standard toolkit, fast simulation is typically implemented outside, in the **custom** software frameworks of each experiment.
⟶ CMS and LHCb use fast simulation hooks offered by GEANT4
⟶ ATLAS is investigating a move from custom framework to Geant4
⟶ Easy to include in any existing G4 application

Use of cell-level data implies experiment-specific models, not every experiment can afford that, but it is probably the fastest (can they offer sufficient accuracy?).

Use of granular voxelization and point clouds can be transferable, and is easier if fast simulation is performed fully inside GEANT4, from taking over particle transportation, via inference, and deposition of generated showers. ⟶ reuse of tools.

⟶ LHCb already has an implementation of a Par04 (=CaloChallenge d2&3) setup

True performance measurements possible once ML model is in production software. (see talk by M.Giannelli at CaloChallenge workshop)

... in the meantime...

Performance of the model (beyond physics validation)

In common: all those exporiments use `C++` based frameworks, so inference must be implemented in `C++`, taking into account how events are processed.

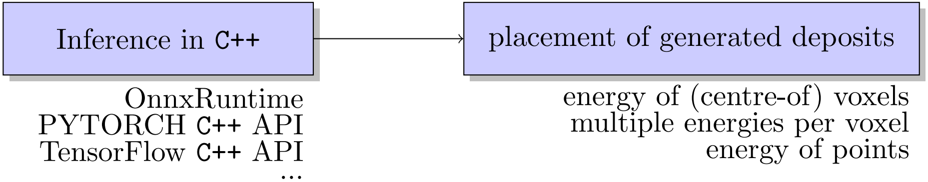
Performance of the model (beyond physics validation)

In common: all those experiments use `C++` based frameworks, so inference must be implemented in `C++`, taking into account how events are processed. ⟶ batching

Performance of the model (beyond physics validation)

In common: all those exporiments use `C++` based frameworks, so inference must be implemented in `C++`, taking into account how events are processed. ⟶ batching

Time-wise (CPU and/or GPU):



Performance of the model (beyond physics validation)

In common: all those experiments use `C++` based frameworks, so inference must be implemented in `C++`, taking into account how events are processed. ⟶ batching

Time-wise (CPU and/or GPU):

Inference in `C++` → placement of generated deposits

OnnxRuntime
PYTORCH `C++` API
TensorFlow `C++` API
...

energy of (centre-of) voxels
multiple energies per voxel
energy of points

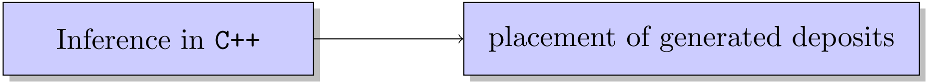
Memory-wise:

the larger the models
the larger the number of models ⇒ the larger memory usage
(per particle type, per region)

Performance of the model (beyond physics validation)

In common: all those exporiments use `C++` based frameworks, so inference must be implemented in `C++`, taking into account how events are processed. ⟶ batching

Time-wise (CPU and/or GPU):



OnnxRuntime
PYTORCH `C++` API
TensorFlow `C++` API
...

energy of (centre-of) voxels
multiple energies per voxel
energy of points

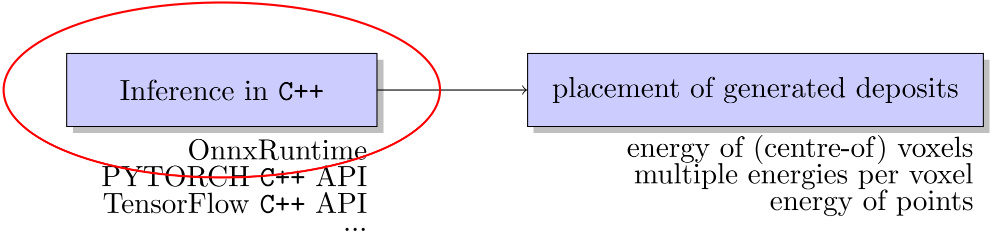
Memory-wise:

the larger the models
the larger the number of models ⇒ the larger memory usage
(per particle type, per region)

Performance of the model (beyond physics validation)

In common: all those exporiments use C++ based frameworks, so inference must be implemented in C++, taking into account how events are processed. ⟶ batching

Time-wise (CPU and/or GPU):



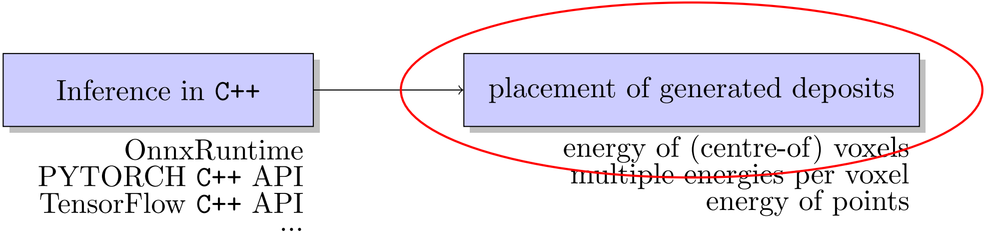
Memory-wise:

the larger the models
the larger the number of models (per particle type, per region) ⇒ the larger memory usage

Performance of the model (beyond physics validation)

In common: all those experiments use `C++` based frameworks, so inference must be implemented in `C++`, taking into account how events are processed. ⟶ batching

Time-wise (CPU and/or GPU):



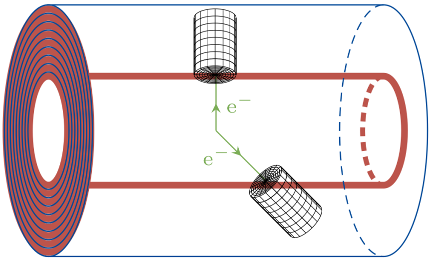
Memory-wise:

the larger the models
the larger the number of models ⇒ the larger memory usage
(per particle type, per region)

Detector setup

Presented data comes from the `Par04` example of Geant4.

Simple cylinders of active (Si) and passive (W) materials.



Detector setup

Presented data comes from the `Par04` example of GEANT4.

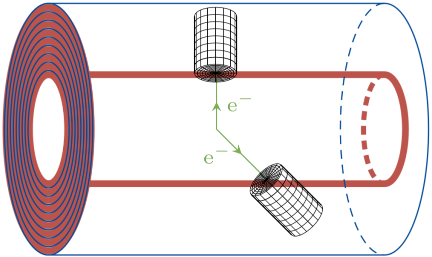
Simple cylinders of active (Si) and passive (W) materials.

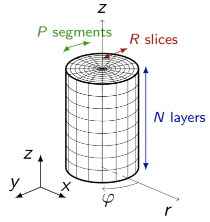
New: introduced (physical) cells, with the total number in the detector: either 300k or 3M.

The total voxelisation of showers is 6.5k (like CaloChallenge dataset2) or 40k (like dataset3) and created around the shower center.

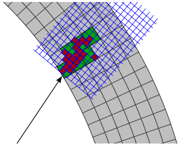
Voxelisation is used first to produce training data, and then generated showers at those (voxels') positions must be placed at/mapped to the cells.

$\longrightarrow$ No matter if voxels or point clouds are used: it's their number in generated shower that counts.





dataset2:
$R \times P \times N = 9 \times 16 \times 45$
dataset3:
$R \times P \times N = 18 \times 50 \times 45$



detector cells vs shower voxels

Disclaimer

Caution: Those results apply to this particular example (which is the origin of the CaloChallenge datasets 2&3).

Disclaimer

Caution: Those results apply to this particular example
(which is the origin of the CaloChallenge datasets 2&3).

Par04 example of Geant4:

300k readout granularity: gitlab.cern.ch/fastsim/par04/-/tree/v11.1_lowgran
3M readout granularity: gitlab.cern.ch/fastsim/par04/-/tree/v11.1_highgran

CaloChallenge dataset 3 is the default, dataset 2 obtained by changing the input macro parameters.

Disclaimer

Caution: Those results apply to this particular example (which is the origin of the CaloChallenge datasets 2&3).

Par04 example of Geant4:

- 300k readout granularity: gitlab.cern.ch/fastsim/par04/-/tree/v11.1_lowgran
- 3M readout granularity: gitlab.cern.ch/fastsim/par04/-/tree/v11.1_highgran

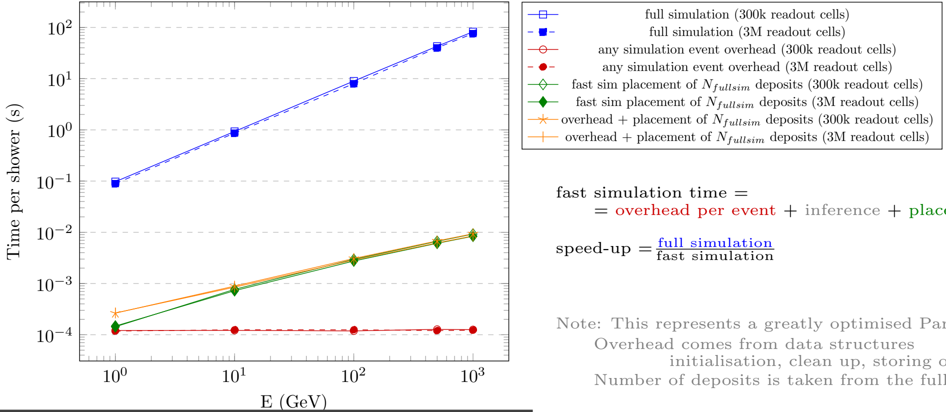
CaloChallenge dataset 3 is the default, dataset 2 obtained by changing the input macro parameters.

There is **no model**, no calculation (ML or parameterisation). Results represent the 'extra' cost of high granularity.

All data points are an average over 10 runs, from 1000 shower samples per particle energy.

Time performance

E (GeV)	1	10	100	500	1000
full sim (300k)	0.097	0.93	9.0	42.9	82.3
full sim (3M)	0.089	0.86	8.0	40.0	76.0
event overhead (300k)	0.00012	0.00012	0.00012	0.00013	0.00013
event overhead (3M)	0.00012	0.00012	0.00013	0.00012	0.00012
fast sim deposits placement (300k, dataset3)	0.00014	0.00077	0.00296	0.00666	0.00923
fast sim deposits placement (3M, dataset3)	0.00015	0.00073	0.00273	0.00609	0.00837
overhead + placement (300k)	0.00026	0.00089	0.00308	0.00679	0.00936
overhead + placement (3M)	0.00027	0.00085	0.00287	0.00621	0.00849



fast simulation time =
= overhead per event + inference + placement of N deposits

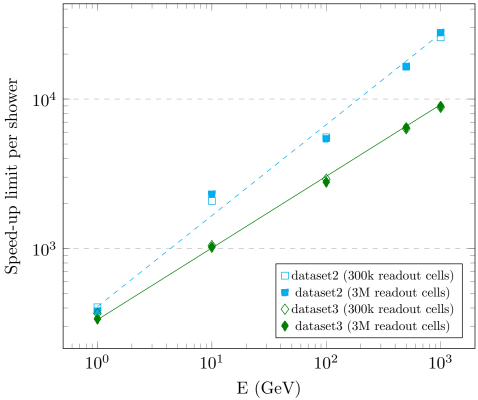
speed-up $= \frac{\text{full simulation}}{\text{fast simulation}}$

Note: This represents a greatly optimised Par04 simulation.
Overhead comes from data structures initialisation, clean up, storing output, ...
Number of deposits is taken from the full simulation.

Speed-up limit

E (GeV)	1	10	100	500	1000
speed-up limit for dataset 2 (fit)	358	2143	6061	17146	28108
speed-up limit for dataset 3 (fit)	321	1073	3029	6547	9274

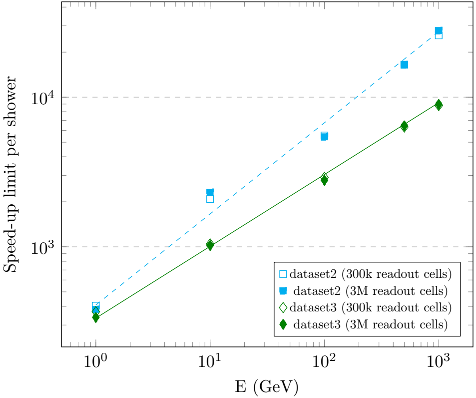
$$\text{speed-up limit} = \frac{\text{full simulation}}{\text{overhead} + \text{placement}}$$



Speed-up limit

E (GeV)	1	10	100	500	1000
speed-up limit for dataset 2 (fit)	358	2143	6061	17146	28108
speed-up limit for dataset 3 (fit)	321	1073	3029	6547	9274

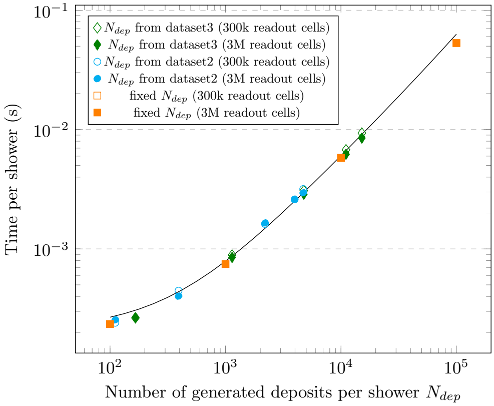
$$\text{speed-up limit} = \frac{\text{full simulation}}{\text{overhead} + \text{placement}}$$



Low energy particles are most populous,
so likely the average speed-up per shower is no larger than $\mathcal{O}(1000)$.

Which is still a huge gain!

Time performance estimation (as a function on number of deposits)



green diamonds correspond to dataset 3,

cyan circles correspond to the dataset 2,

orange circles is an artificial placement of N_{dep}.

Fit to data:

$t(s) = 5.015e^{-13} N_{dep}^2 + 5.79e^{-7} N_{dep} + 0.00021$

↑ can be applied to any dataset 2 or 3 calculations on top of the inference time.

Batching

Few notes:

- Batch size for inference has crucial impact on speed-up;

Batching

Few notes:

- Batch size for inference has crucial impact on speed-up;
- It is not impossible to batch showers across events, but it requires major work on experiments' frameworks, and we are not yet even at batch=1 models in production (exception: ATLAS).

Batching

Few notes:

- Batch size for inference has crucial impact on speed-up;
- It is not impossible to batch showers across events, but it requires major work on experiments' frameworks, and we are not yet even at batch=1 models in production (exception: ATLAS).
- Possible batching within a single event can be larger for heavy events ($t\bar{t}$) and smaller for lighter ones (minimum bias).

Batching

Few notes:

- Batch size for inference has crucial impact on speed-up;
- It is not impossible to batch showers across events, but it requires major work on experiments' frameworks, and we are not yet even at batch=1 models in production (exception: ATLAS).
- Possible batching within a single event can be larger for heavy events ($t\bar{t}$) and smaller for lighter ones (minimum bias).
- To remember: fast simulation may require more models (per particle, per detector region).

Batching

Few notes:

- Batch size for inference has crucial impact on speed-up;
- It is not impossible to batch showers across events, but it requires major work on experiments' frameworks, and we are not yet even at batch=1 models in production (exception: ATLAS).
- Possible batching within a single event can be larger for heavy events ($t\bar{t}$) and smaller for lighter ones (minimum bias).
- To remember: fast simulation may require more models (per particle, per detector region).
- This exercise give a rough estimate on calculated batch sizes. It is done using Pythia and looking at particles entering calorimeters.

Simplistic detector

Few layers of materials (average tracker budget) are placed in front of the calorimeter.

Particles are counted at the entrance to the calorimeters.

(mean and RMS given for minbias, Gaussian fit for $t\bar{t}$)

Simple application used for do this study:

gitlab.cern.ch/fastsim/particle_multiplicities

Simplistic detector

Few layers of materials (average tracker budget) are placed in front of the calorimeter.
Particles are counted at the entrance to the calorimeters.

(mean and RMS given for minbias, Gaussian fit for $t\bar{t}$)

Simple application used for do this study:
gitlab.cern.ch/fastsim/particle_multiplicities

		$\|\eta\| < 1.5$ E>100 MeV	E>1 GeV	E>10 GeV	$\|\eta\| < 3$ E>100 MeV	E>1 GeV	E>10 GeV
$t\bar{t}$ @ 14 TeV pp	e^+/e^-	58±22	**7.7** ± 6.0	0.6±0.9	311±184	**24.9**±10.7	1.2±1.5
	γ	107±40	**17.0** ± 8.9	1.2±1.6	536±432	**46.3**±16.8	2.2±2.2
	π^+/π^-	83±33	**29.6** ± 13.4	0.6±3.7	363±232	**104.7**±37.1	7.3±4.3
minimum bias @ 14 TeV	e^+/e^-	11.8±11.7	**0.3**±0.8	-	108±84	**2.7**±3.3	-
	γ	26.3±22.6	**1.2**±1.8	-	206±156	**7.8**±7.5	-
	π^+/π^-	24.5±20.9	**3.6**±4.5	-	160±118	**33.4**±27.5	0.3±0.6
$t\bar{t}$ @ 365 GeV ee	e^+/e^-	22.7±9.1	**4.0**±3.0	0.25±0.5	26.2±10.6	**4.5**±2.9	0.3±0.5
	γ	38.4±14.2	**8.8**±4.0	0.4±0.7	44.6±15.4	**9.7**±3.9	0.5±0.7
	π^+/π^-	29.9±11.4	**14.9**±6.0	0.4±1.4	32.4±12.1	**16.1**±6.1	0.5±1.5

See talk from yesteday for more details on ODD.
Particles are counted at the entrance to the calorimeters.

(mean and RMS given for minbias, Gaussian fit for $t\bar{t}$)

Analysis application used for do this study: gitlab.cern.ch/fastsim/ddODD

See talk from yesteday for more details on ODD.
Particles are counted at the entrance to the calorimeters.

(mean and RMS given for minbias, Gaussian fit for $t\bar{t}$)

Analysis application used for do this study: gitlab.cern.ch/fastsim/ddODD

		$\|\eta\| < 1.5$ *(barrel)*			$\|\eta\| < 3$ *(barrel + endcaps)*		
		$E > 100$ MeV	$E > 1$ GeV	$E > 10$ GeV	$E > 100$ MeV	$E > 1$ GeV	$E > 10$ GeV
$t\bar{t}$ **@ 14 TeV**	e^+/e^-	64.1±25.6	**9.1**±6.2	0.6±1.0	86.8±33.2	**15.5**±7.8	1.1±1.4
	γ	113.1±44	**19.3**±9.2	1.3±1.5	159±68	**35**±13.6	2.7±2.6
	π^+/π^-	83±31	**32.0**±13.0	1.6±3.5	119±45	**55.7**±20.4	4.4±5.9
minbias @ 14 TeV	e^+/e^-	12.0±11.6	**0.4**±0.8	**-**	19.7±18.4	**1.6**±2.2	-
	γ	24.3±21.8	**1.3**±2.0	-	41.2±35.0	**6.0**±6.4	-
	π^+/π^-	22.3±19.9	**4.0**±5.2	-	35.5±30.1	**11.8**±11.6	0.3±0.7

Summary

1. ML models are exciting to develop, but imagine the joy of having your model actually there inside the experiment, in the production software!

Summary

1. ML models are exciting to develop, but imagine the joy of having your model actually there inside the experiment, in the production software!

   It takes few more steps ...

Summary

1. ML models are exciting to develop, but imagine the joy of having your model actually there inside the experiment, in the production software!

   It takes few more steps ...

2. Any speed-up value, a comparison to Geant4 simulation time **should** take into account a small event overhead + hit placement time (0.2–10 ms/shower)

Summary

1. ML models are exciting to develop, but imagine the joy of having your model actually there inside the experiment, in the production software!

   It takes few more steps ...

2. Any speed-up value, a comparison to GEANT4 simulation time **should** take into account a small event overhead + hit placement time (0.2–10 ms/shower)→ otherwise only absolute measurements are fair.

Summary

1. ML models are exciting to develop, but imagine the joy of having your model actually there inside the experiment, in the production software!

   It takes few more steps ...

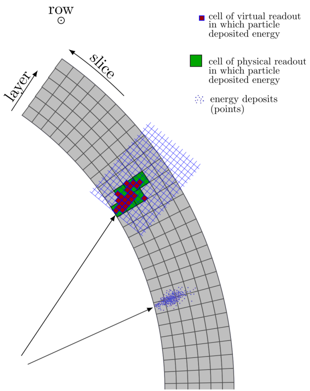
2. Any speed-up value, a comparison to GEANT4 simulation time **should** take into account a small event overhead + hit placement time (0.2–10 ms/shower)$\rightarrow$ otherwise only absolute measurements are fair.

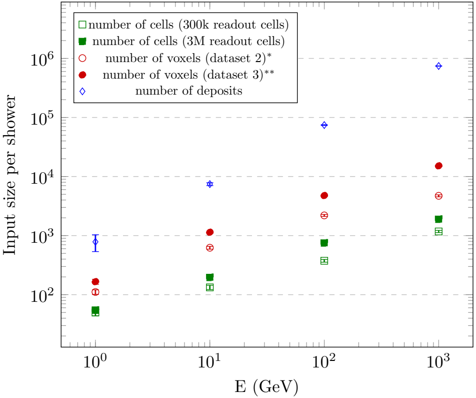
3. So far (per event) realistic batching is no larger than 20 – 40, depending if a single model can be used for barrel or barrel+endcap region (and it's for heavy events). Inference per batch=1 should probably always be measured (useful for lighter events).

BACKUP

Size of the input vs shower representation

E (GeV)	1	10	100	1000
N_{cells} (300k)	50±6	133±11	375±20	1175±51
N_{cells} (3M)	55±7	197±17	754±34	2796±55
N_{voxels} (dataset 2)	111 ±10	623 ± 34	2'198±84	4'712±186
N_{voxels} (dataset 3)	166 ± 16	1'140 ±53	4'769 ±162	15'164 ±470
$N_{deposits}$	785±249	7'444±423	74'318±1204	743'009±4267



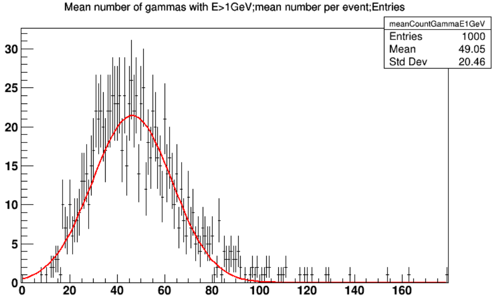


* voxelization with around 0.5 $X_0 \times 0.5\ R_M \times 0.4$ rad voxel size

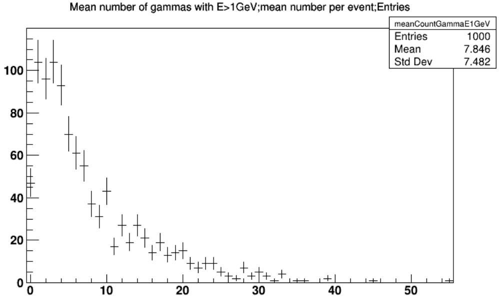
** voxelization with around 0.5 $X_0 \times 0.25\ R_M \times 0.125$ rad voxel size

Particle multipliciies

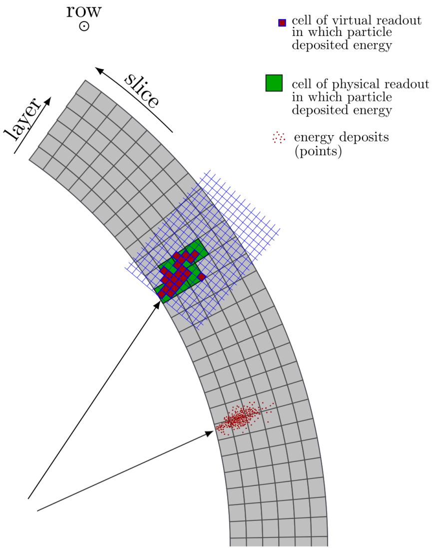
gg2ttbar @ 14 TeV



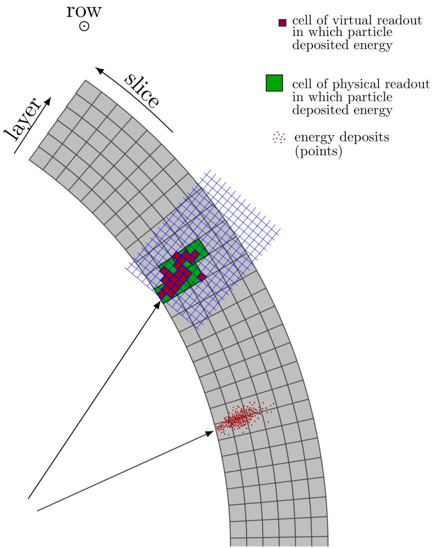
gg2minbias @ 14 TeV

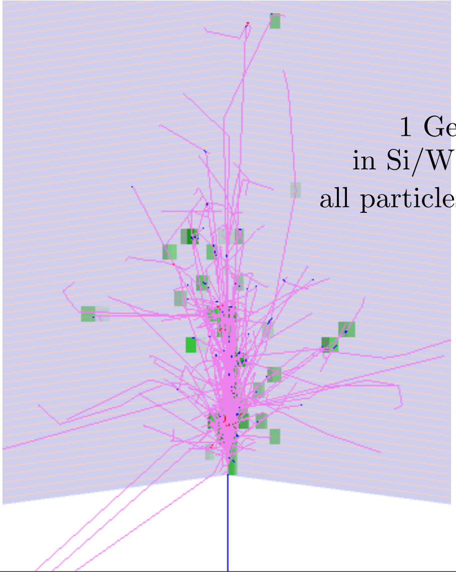


Shower data representation: cells, voxels, point clouds, ...



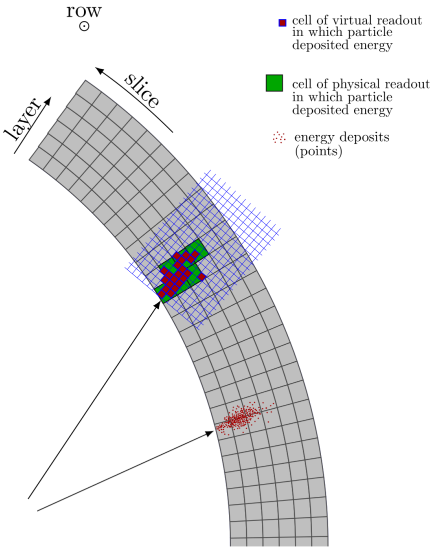
Shower data representation: cells, voxels, point clouds, ...

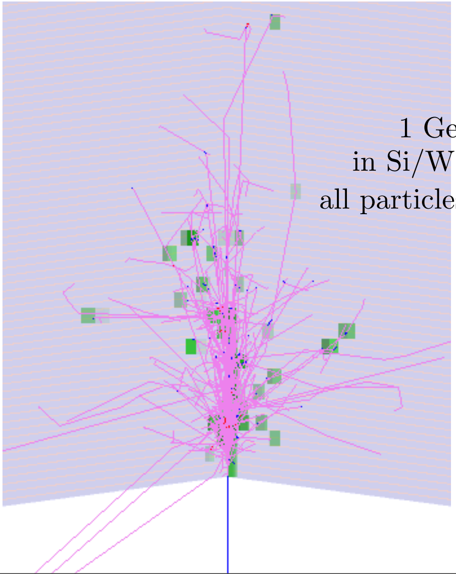




1 GeV e^-
in Si/W detector
all particles $> 1\,\mathrm{keV}$

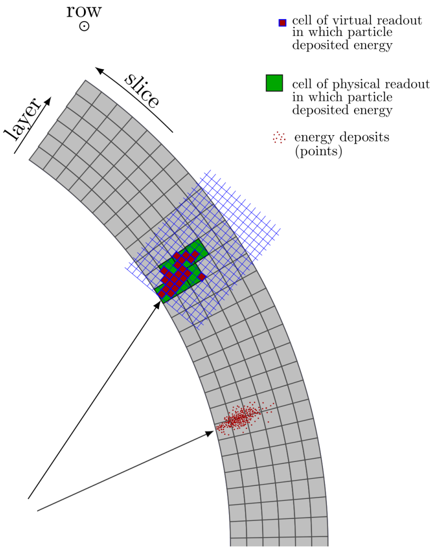
Shower data representation: cells, voxels, point clouds, ...

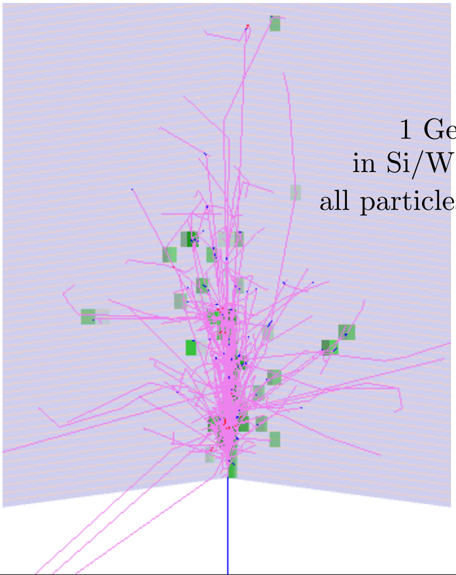




1 GeV e^-
in Si/W detector
all particles $> 10\,\mathrm{keV}$

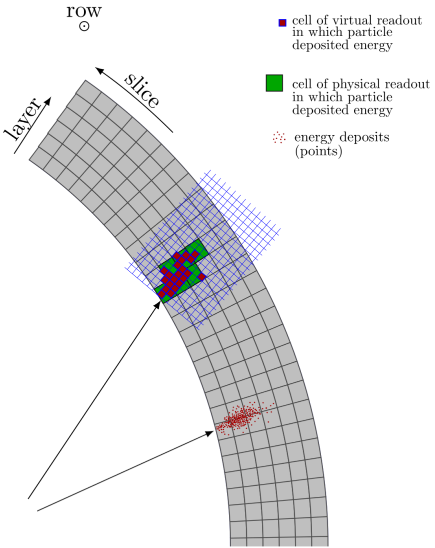
Shower data representation: cells, voxels, point clouds, ...

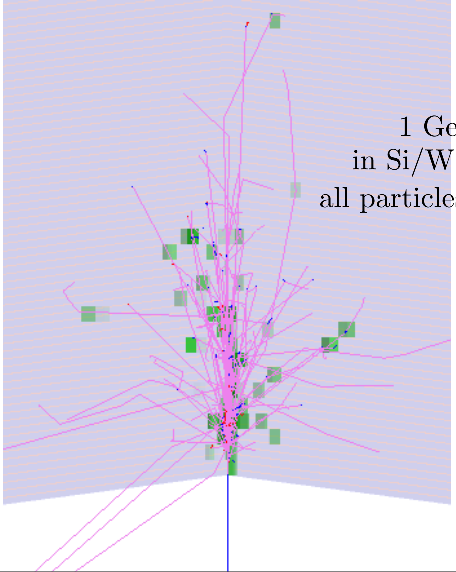




1 GeV e^-
in Si/W detector
all particles $> 100\,\mathrm{keV}$

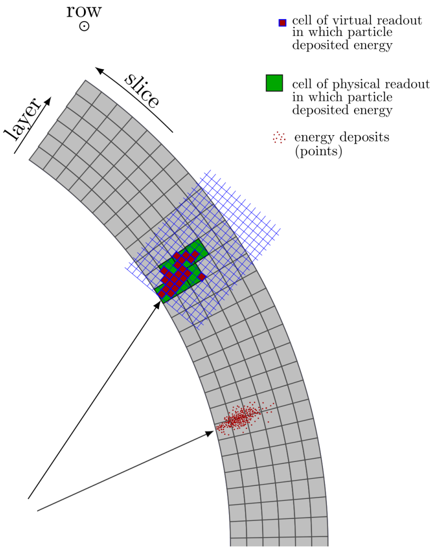
Shower data representation: cells, voxels, point clouds, ...

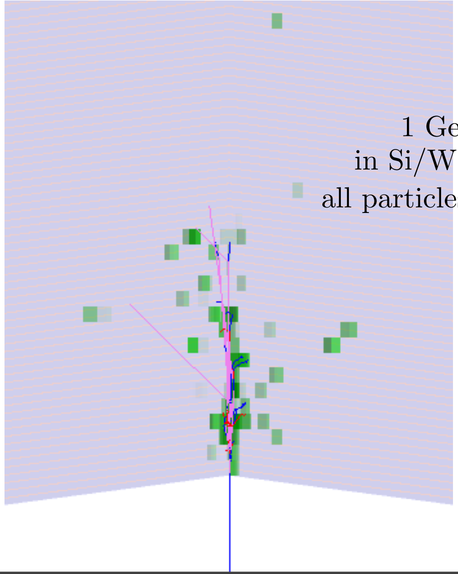




1 GeV e^-
in Si/W detector
all particles > 1 MeV

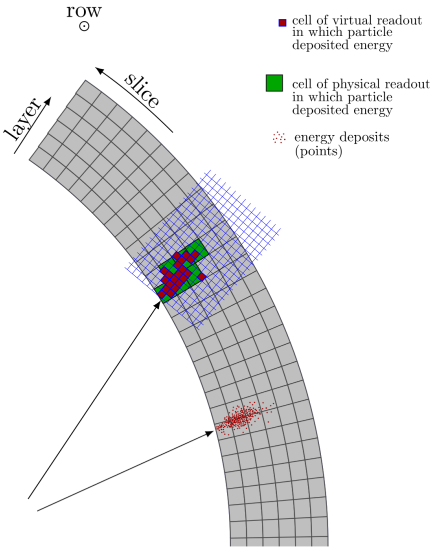
Shower data representation: cells, voxels, point clouds, ...

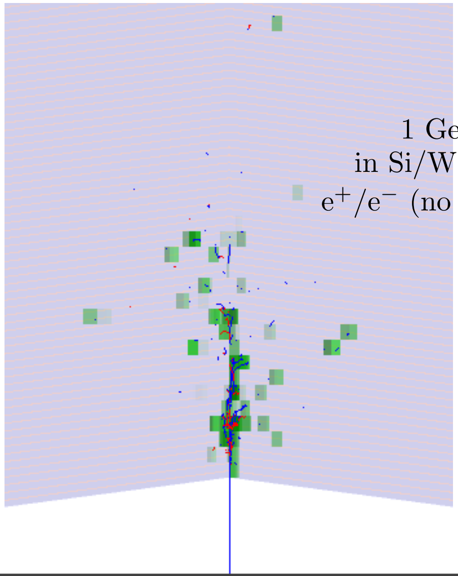




1 GeV e^-
in Si/W detector
all particles $> 10\,\text{MeV}$

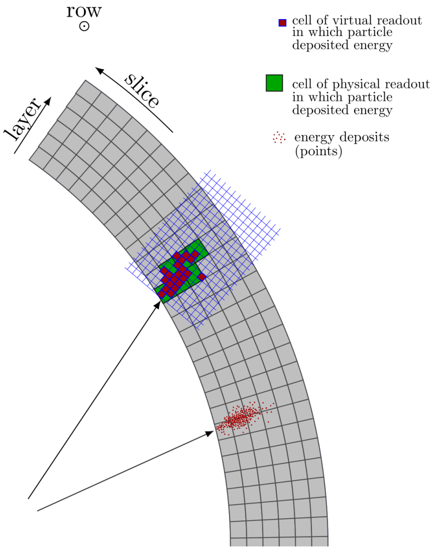
Shower data representation: cells, voxels, point clouds, ...

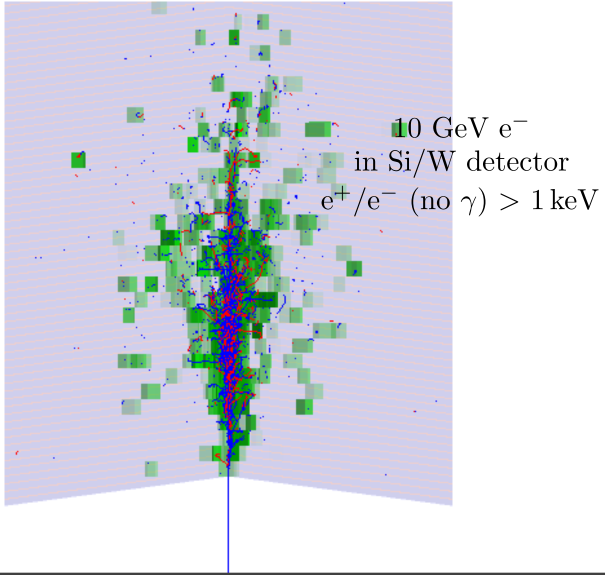




1 GeV e^-
in Si/W detector
e^+/e^- (no γ) $> 1\,\mathrm{keV}$

Shower data representation: cells, voxels, point clouds, ...





Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was som
unprocessed data that should have been added to the final page this extra page
been added to receive it.

If you rerun the document (without altering it) this surplus page will go away,
because LaTeX now knows how many pages to expect for this document.