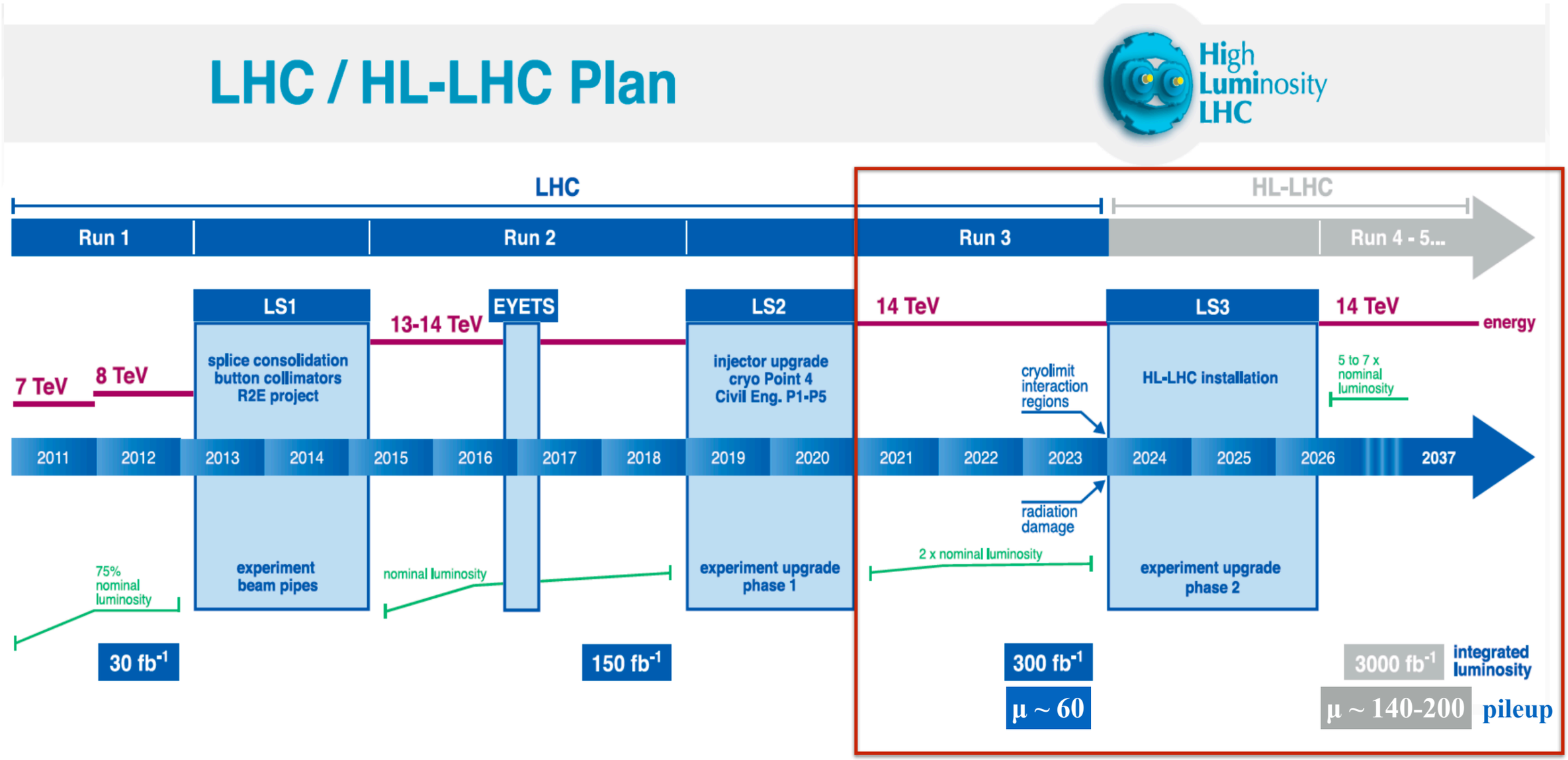
Fast Simulation in ATLAS

Hasib Ahmed

February 17, 2020

Towards Run 3 and beyond

Significant increase in integrated luminosity and number of collisions per bunch crossing in Run3 and Run4



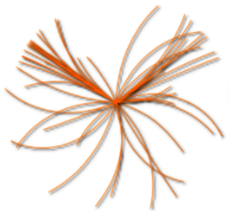
Increasing luminosity imposes greater requirement on Monte Carlo statistics

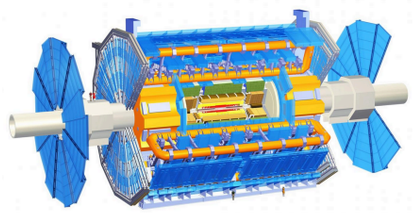
Monte Carlo Production Steps

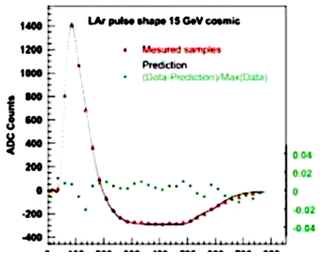
Interaction of particles with matter

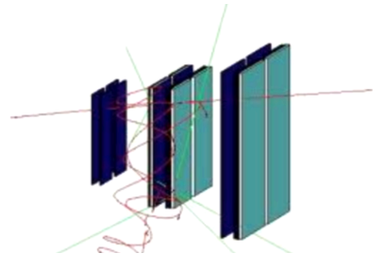
Hit/energy deposits converted to digital signal

Build physics objects



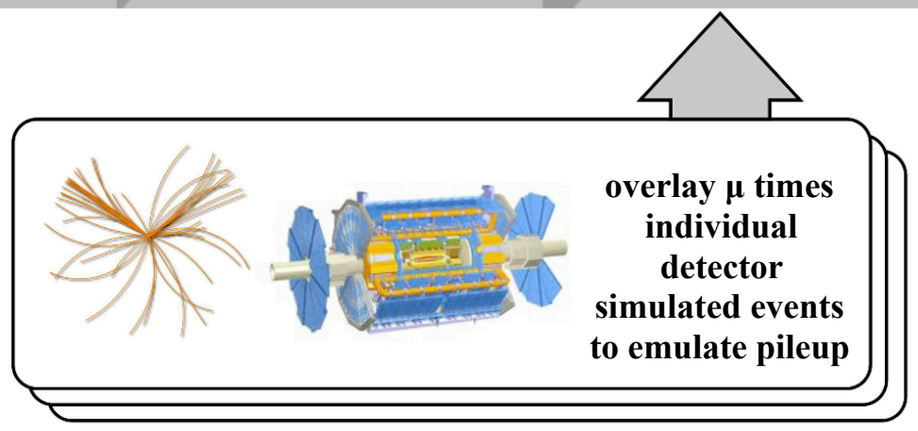






ROOT
Data Analysis Framework

Event Generation → Detector Simulation → Digitization → Reconstruction → Physics Analysis



Detector Simulation:

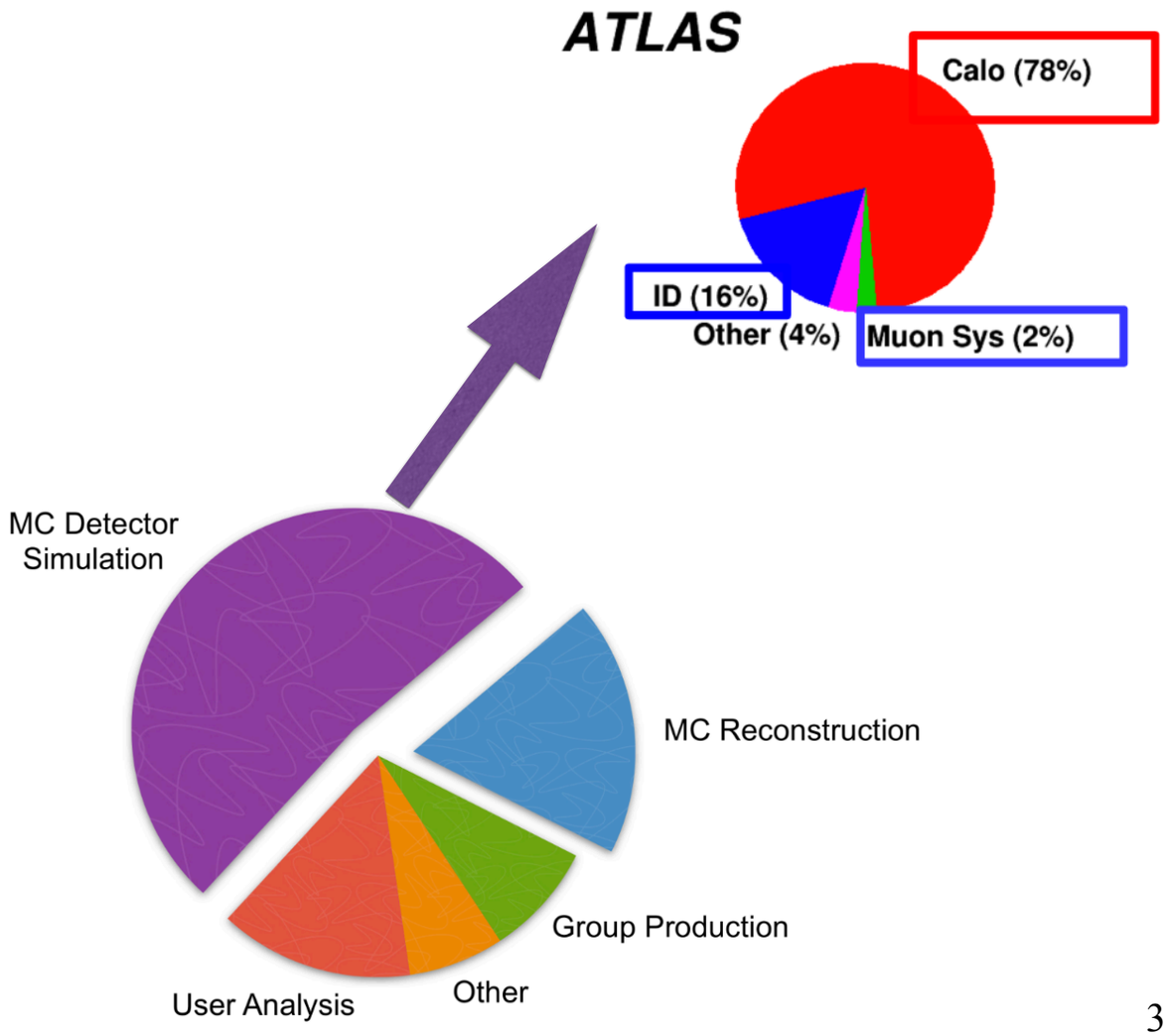
- ~ 80% of CPU spent in calorimeters
- Dense hit content in inner trackers

Digitization:

- Large number of inner tracker readout channels
- Complex modeling of readout emulation

Reconstruction:

- Pattern recognition (combinatorics) function of average pileup

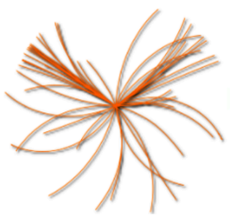


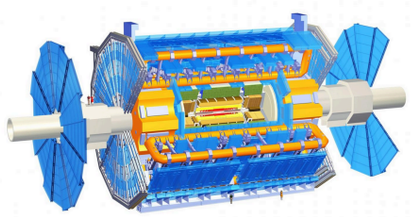
Fast Monte Carlo Production Steps

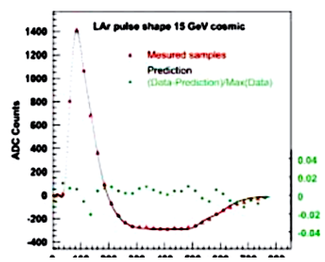
Interaction of particles with matter

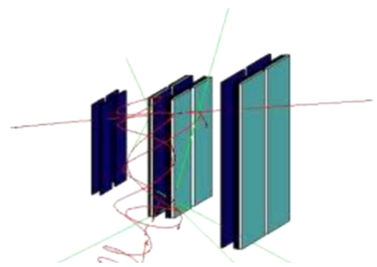
Hit/energy deposits converted to digital signal

Build physics objects









ROOT Data Analysis Framework

Event Generation → Detector Simulation → Digitization → Reconstruction → Physics Analysis

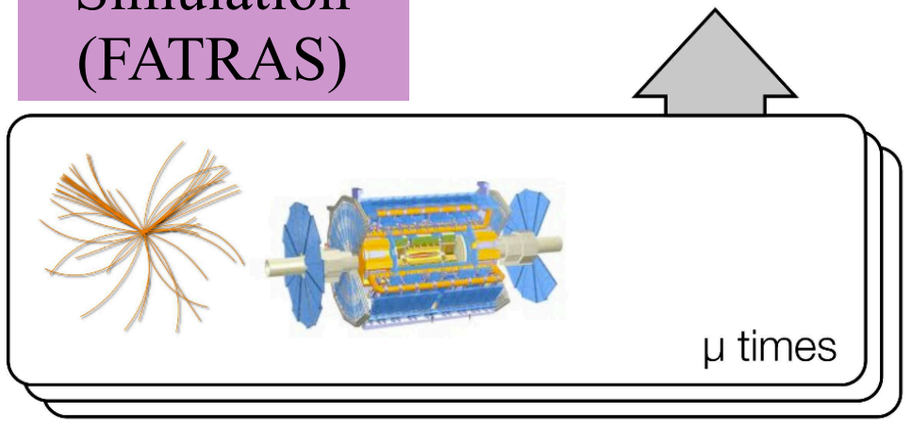
Fast Calorimeter Simulation (FastCaloSim)

Fast Digitization

Fast Reconstruction

Fast Tracker Simulation (FATRAS)

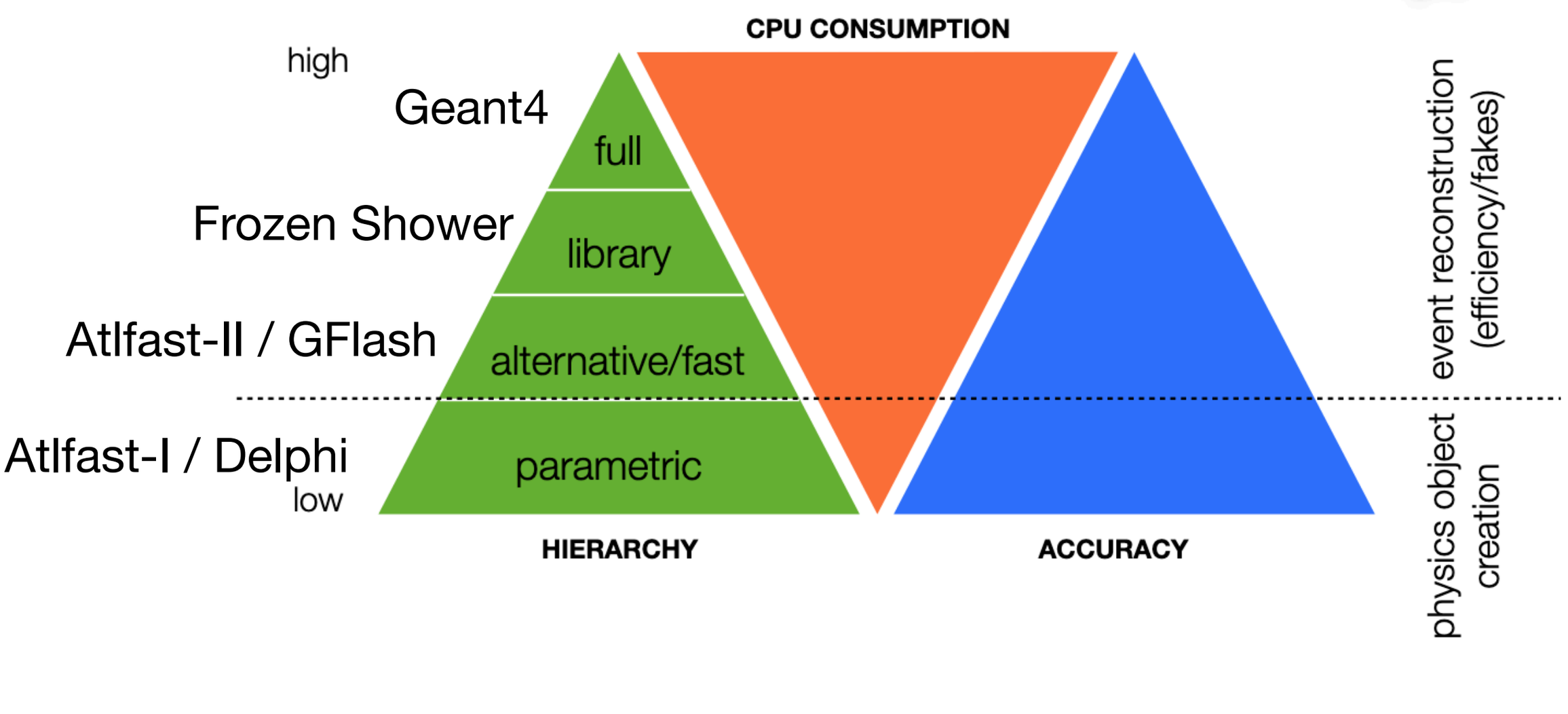
Pileup Overlay



How to speed up simulation

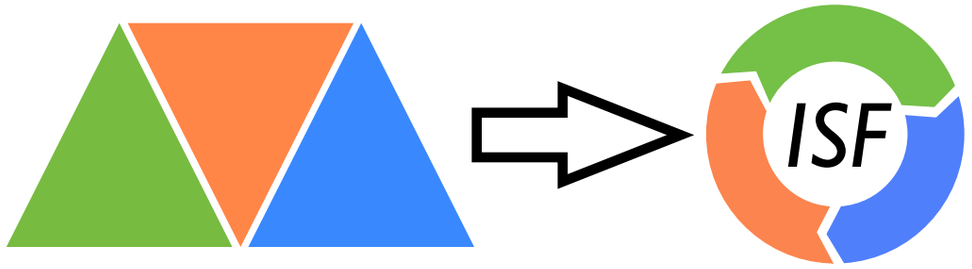
Sacrifice accuracy and precision for speed

- Assume simplified geometry
- Optimize particle transport and navigations
- Parameterize detector responses



Integrated Software Framework (ISF)

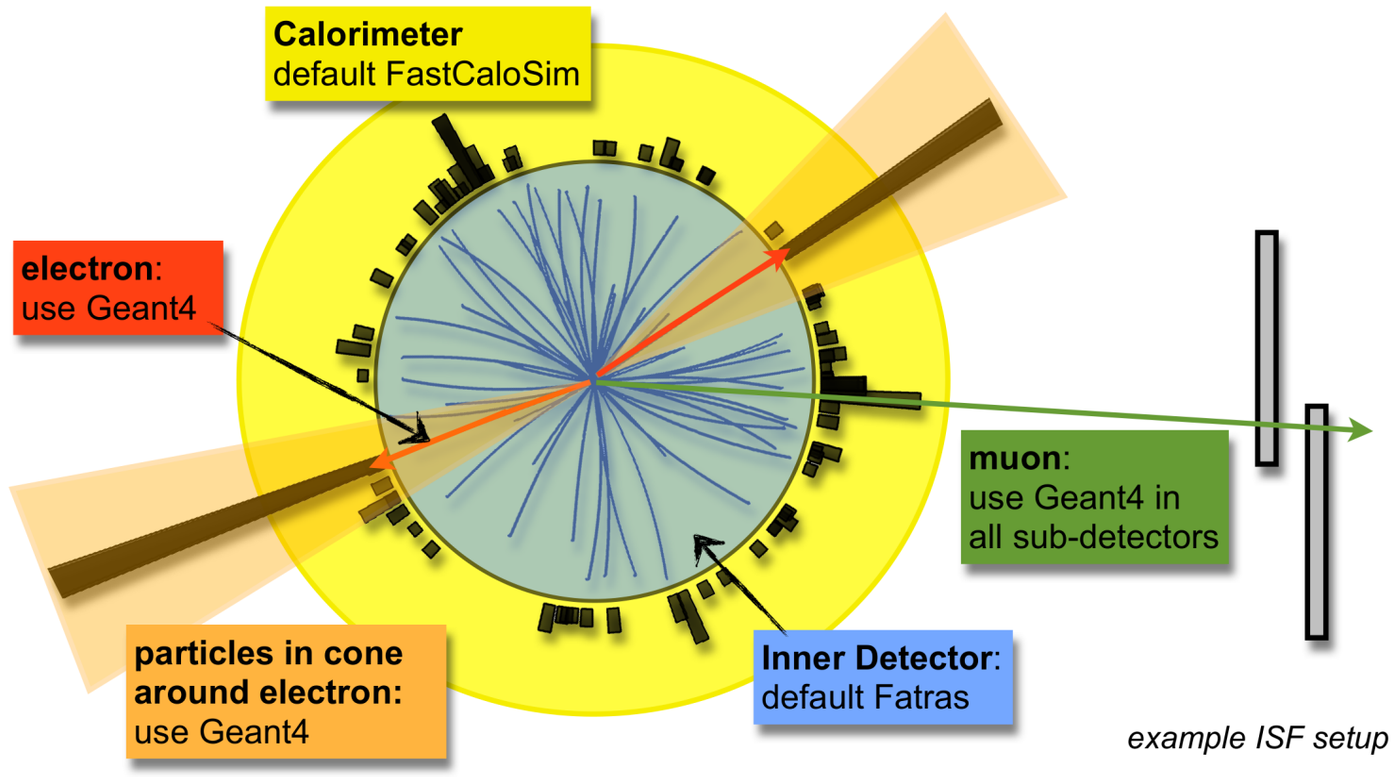
Break the simulation hierarchy by combining Full & Fast simulation in one event



- Choose different simulator for :
  - ✓ particle
  - ✓ sub-detector
  - ✓ interesting cone and region of interest (RoI)

- Save CPU time and keep high accuracy for interesting data

- Compatible with multi-processing (MP) and multi-threading (MT)

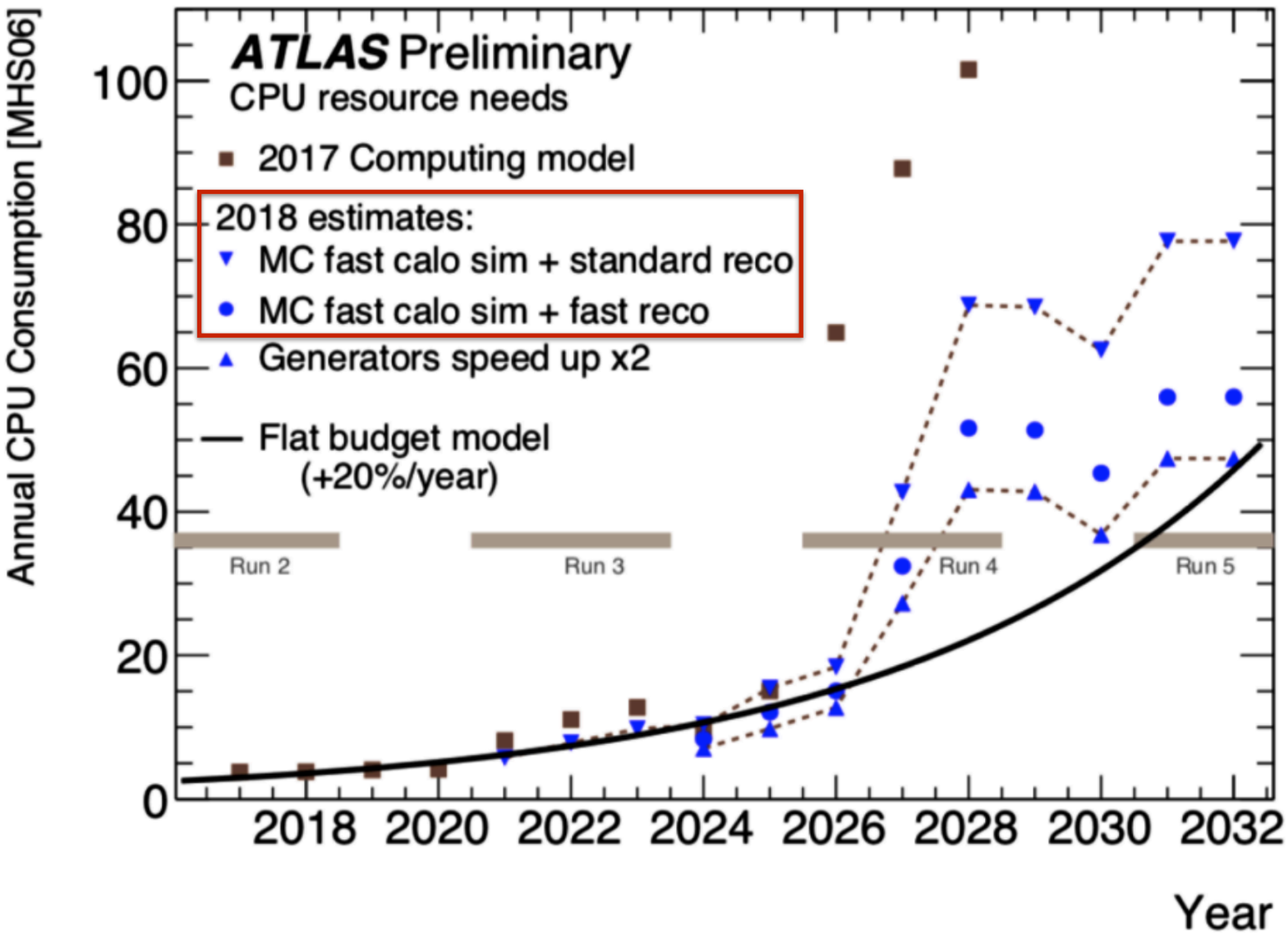


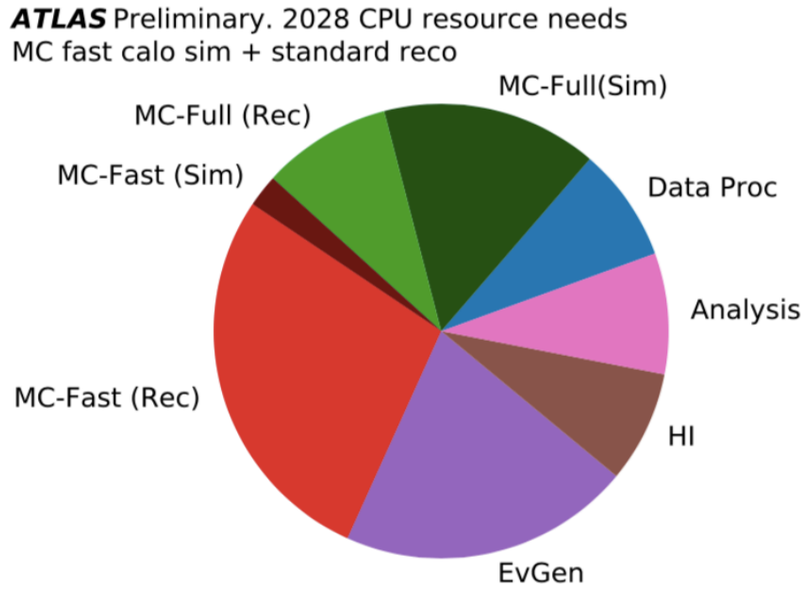
example ISF setup

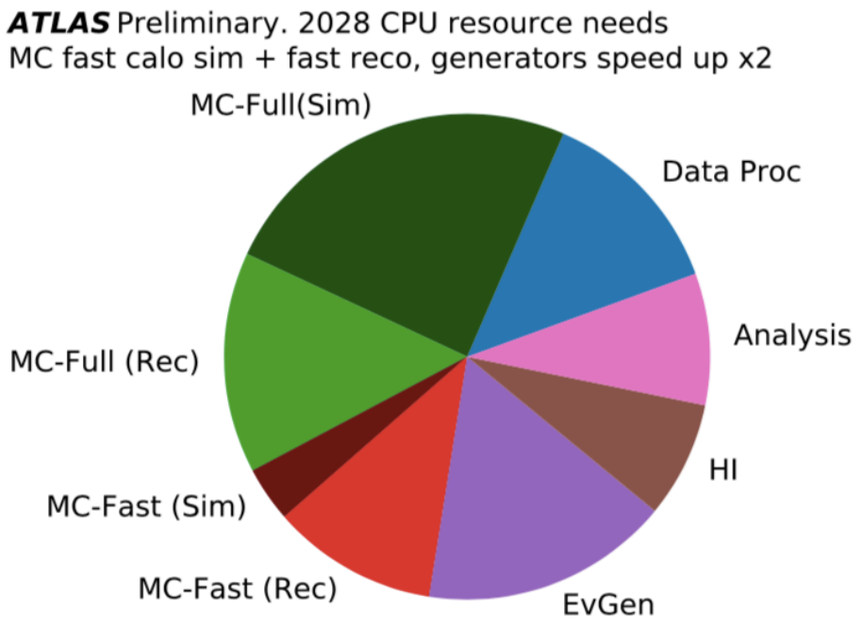
Run 3 and HL-HLC CPU Projections

Run 3: 50% of the simulation with FastCalo sim

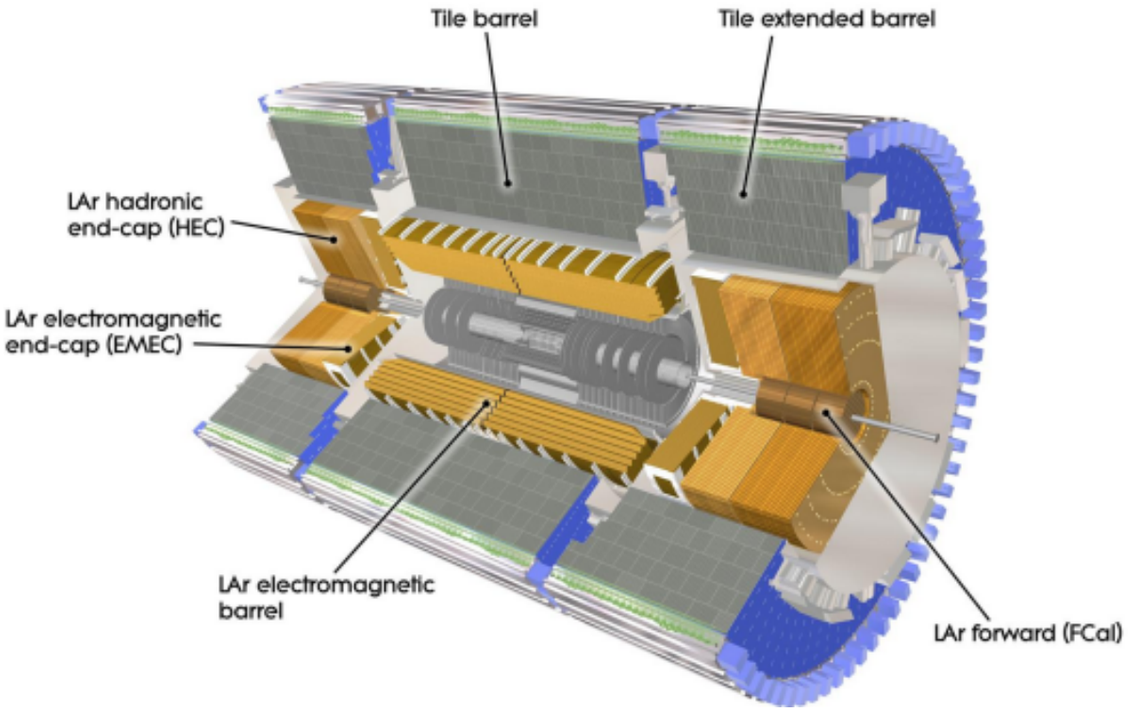
Run 4: 75% of the simulation with FastCalo sim





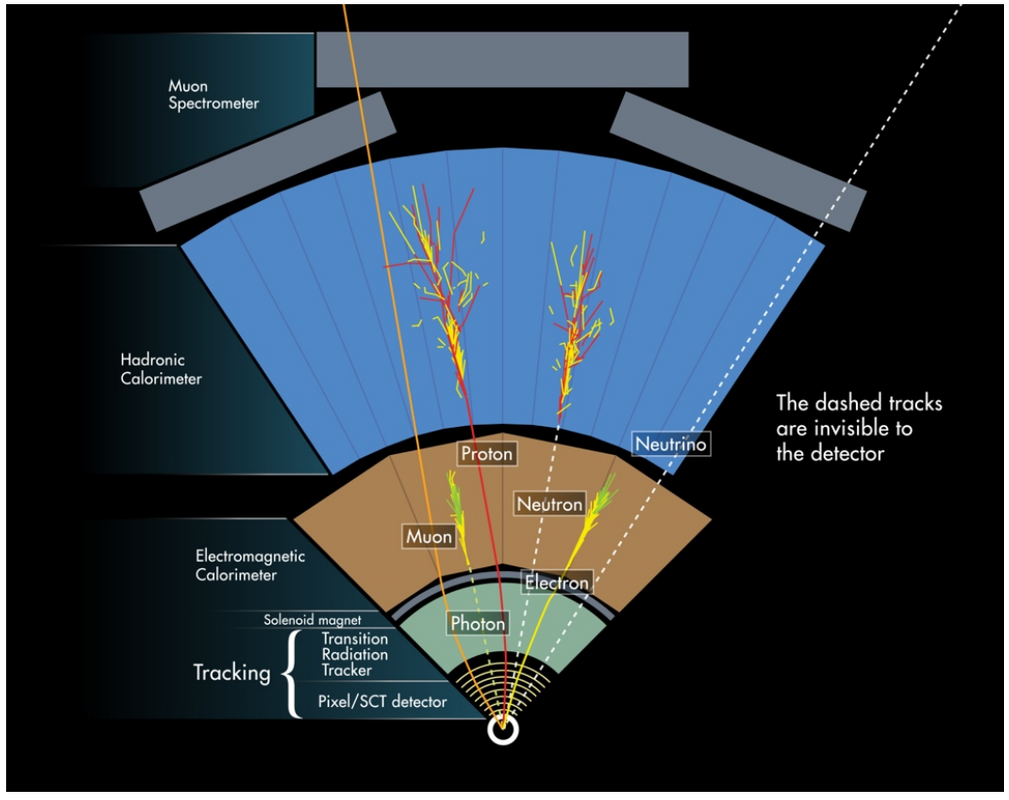


ATLAS Calorimeter



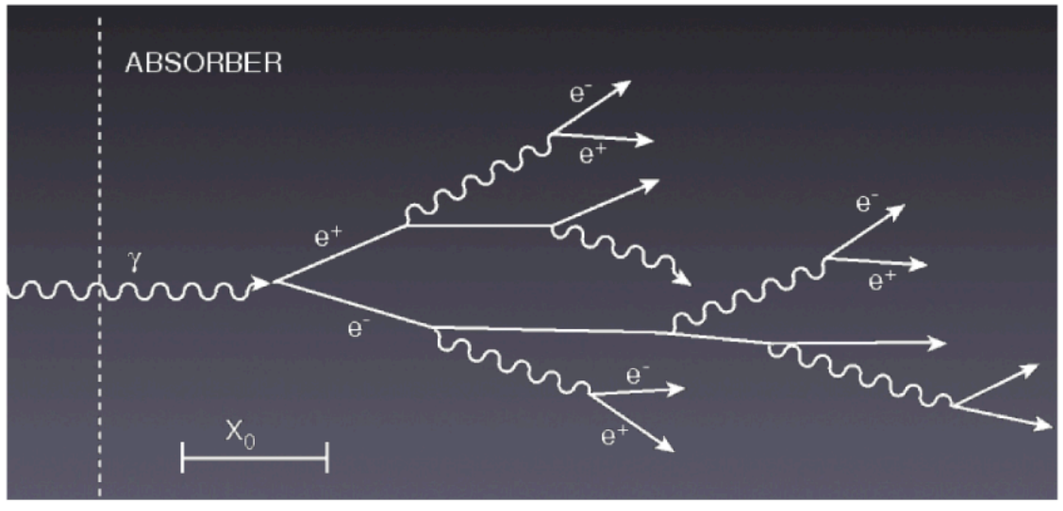
Covering $|\eta|<4.9$

Materials:
Liquid Argon + Lead, or copper or tungsten
Tile Cal: Steel + plastic

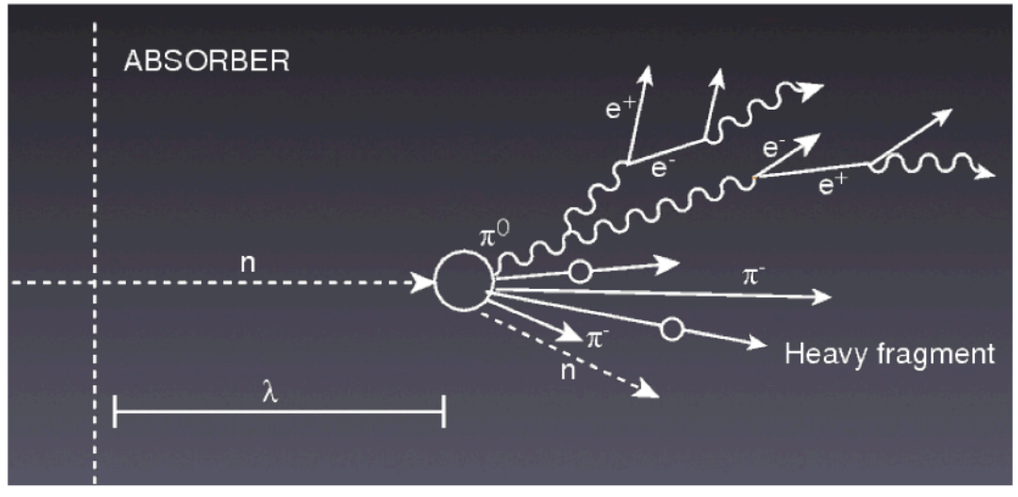


Readout channels: ~190 k in total
Samplings ("cell layers"): 24

EM shower:



Hadronic shower:



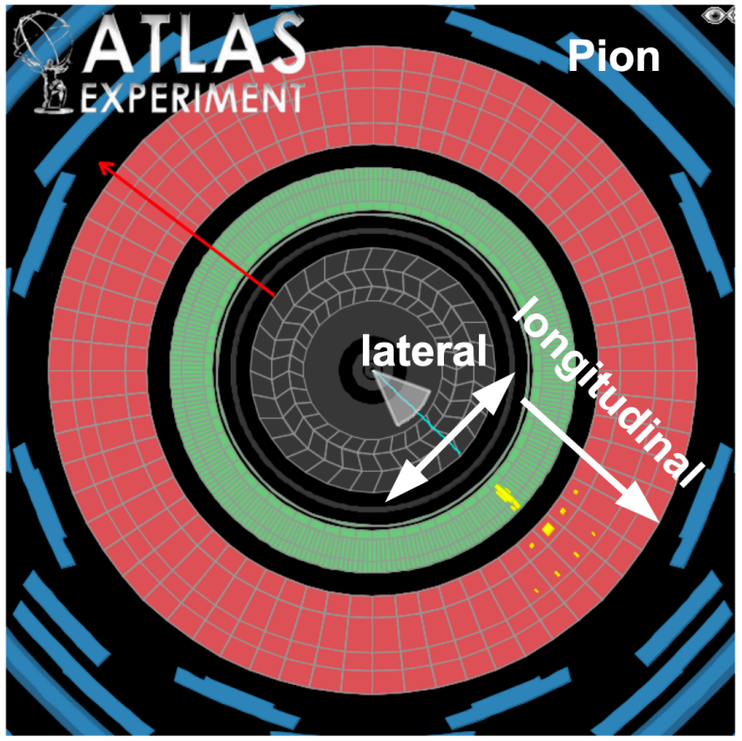
Wider, slower, larger fluctuations

Fast Calorimeter Simulation Principles

Instead of simulating particle interactions, parametrize the detector response of single particles

Use e/γ to represent Electro Magnetic shower

Use charged pions to represent Hadronic shower

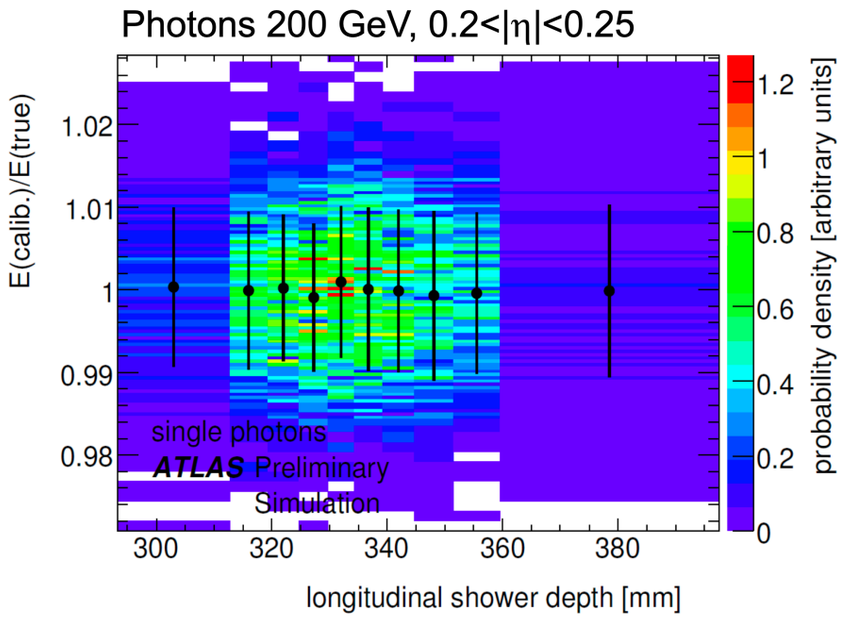


Parametrize the longitudinal and lateral shower development

Use the parametrization at simulation step to deposit energy in calorimeter cells using simplified geometry

Needs to be memory efficient and provide good CPU and Physics performance

FastCaloSimV1 as used in Run 1 & 2

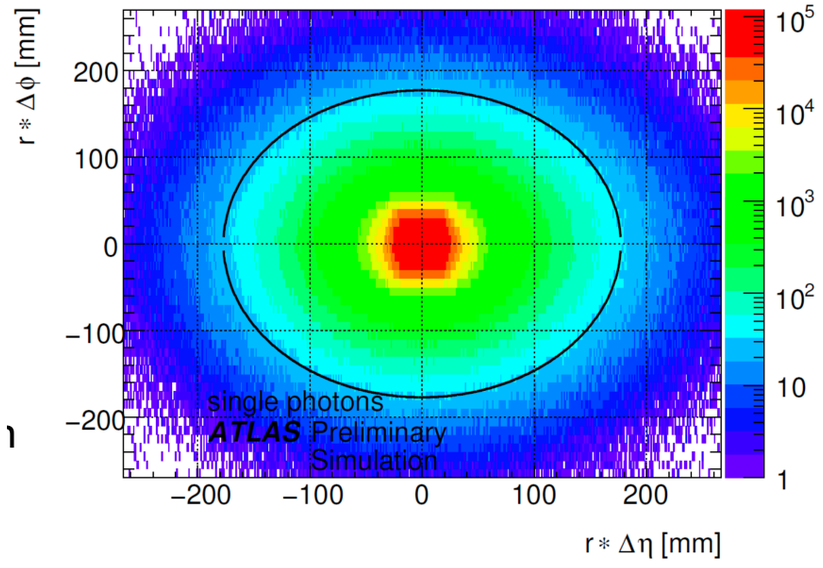


Longitudinal energy parametrization:

- For each particle, energy and |η| parametrize total energy, energy fraction in each layer as a function of shower shower depth
- Correlations between the energy deposits in all layers stored in correlation matrices
- During simulation randomly draw an energy value and energy fractions from the stored 2D histograms

Lateral shower shape parametrization:

- For each bin of shower depth, the average lateral shower shape is parametrized using a symmetric radial function modified centered around the impact point
- A asymmetric term is used to modify the symmetric function for particles entering at large incident angle
- During simulation the energy of a cell is determined by the integral of the shape function over the cell surface area



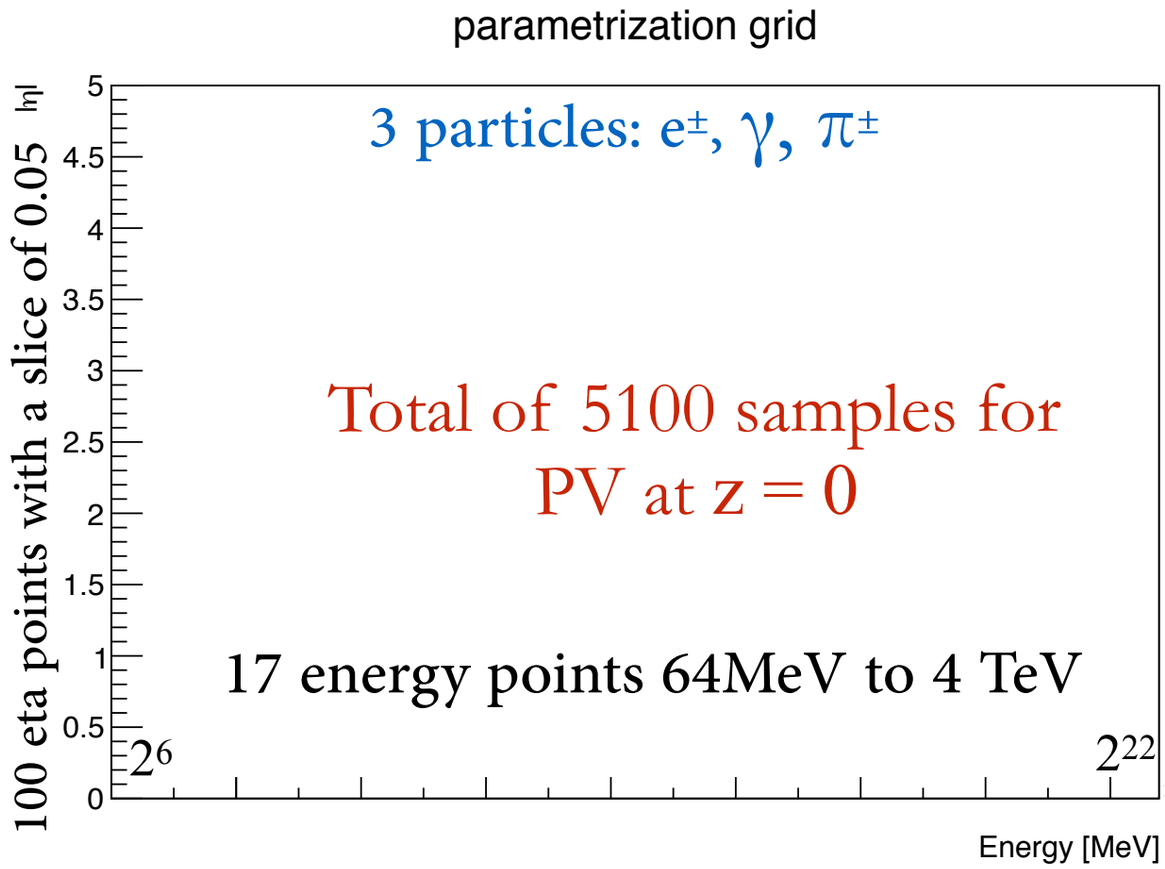
Good average shower description, poor modeling of substructure variables

Provides 10x speed gain compared to Full Geant4

FastCaloSimV2

Successor of FastCaloSim, to be used in bulk production in Run 3

- Geant4 simulated single particles: $e, \gamma, \pi^{\pm}$ without noise and cross-talk
- Single particles starts at the calorimeter surface (not at the detector center)
- Eta grid: 100 bins in size of 0.05 covering $0 < |\eta| < 5$
- Energy grid: 17 bins from 60 MeV - 4 TeV

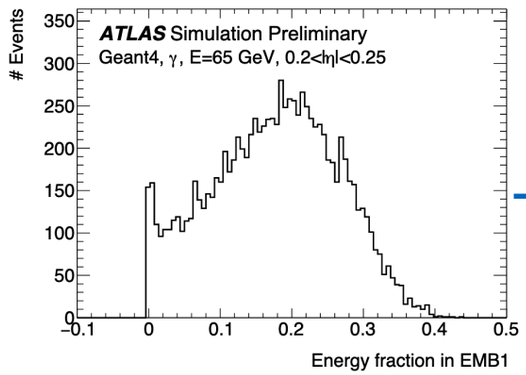


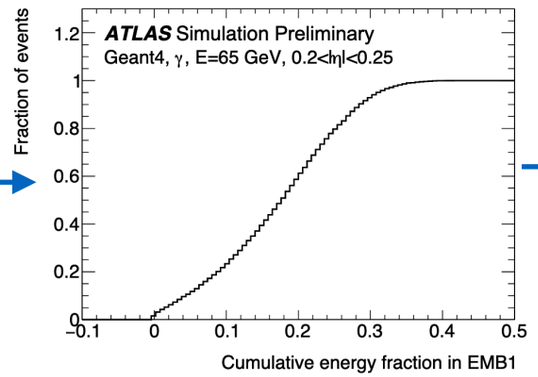
Challenges:

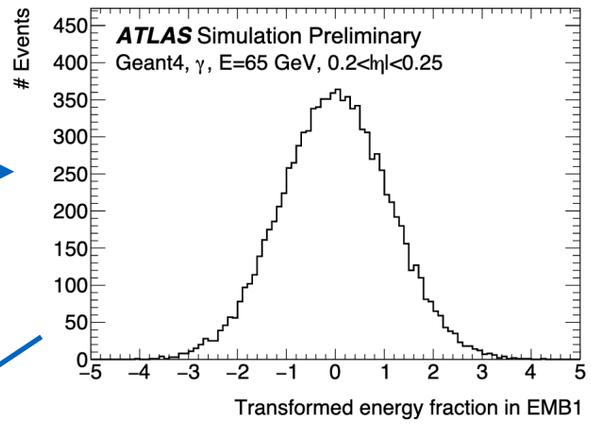
- Improve physics performance significantly
- Stringent memory requirement on the parametrization
- Maintain / improve on the speed gained by FastCaloSimV1

FastCaloSimV2: Dimensionality Reduction

Dimensionality reduction and energy decorrelation using Principal Component Analysis (PCA)







G4 simulated particles in E-η grid

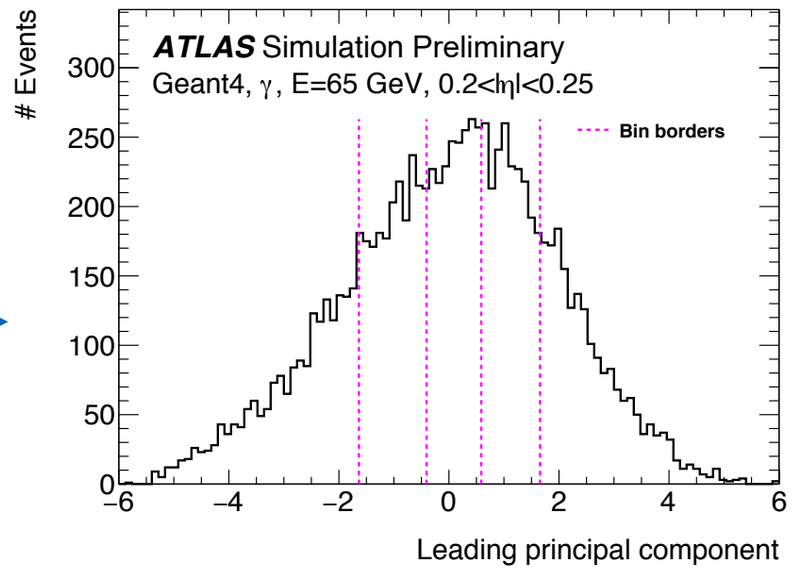
Total energy, Energy fractions in each layer

Principle Component Analysis (PCA) N components

1st PCA to divide Geant4 dataset

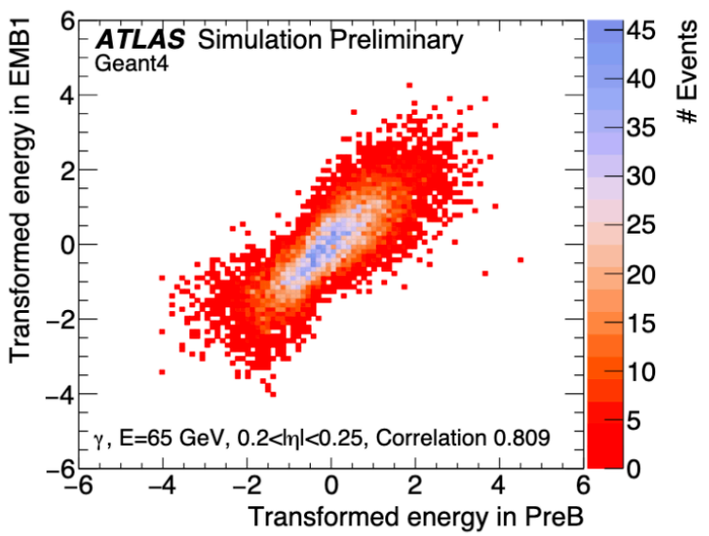
Longitudinal parametrization

Lateral parametrization

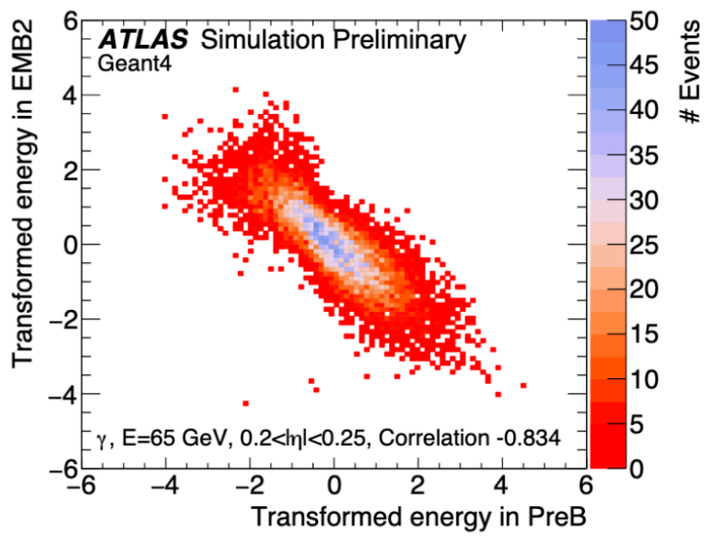


FastCaloSimV2: Dimensionality Reduction

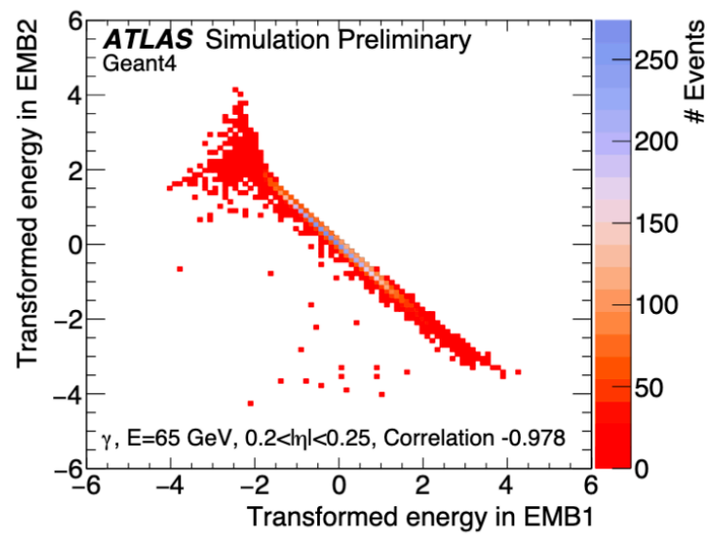
Before PCA:



(a) Presampler vs EM Barrel 1

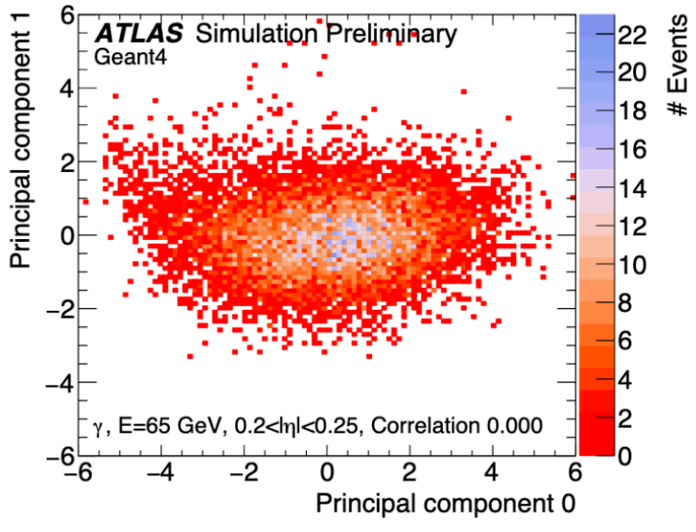


(b) Presampler vs EM Barrel 2

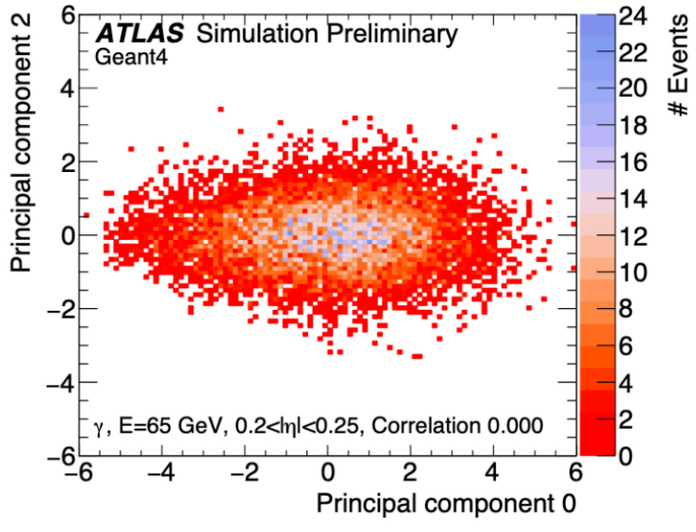


(c) EM Barrel 1 vs EM Barrel 2

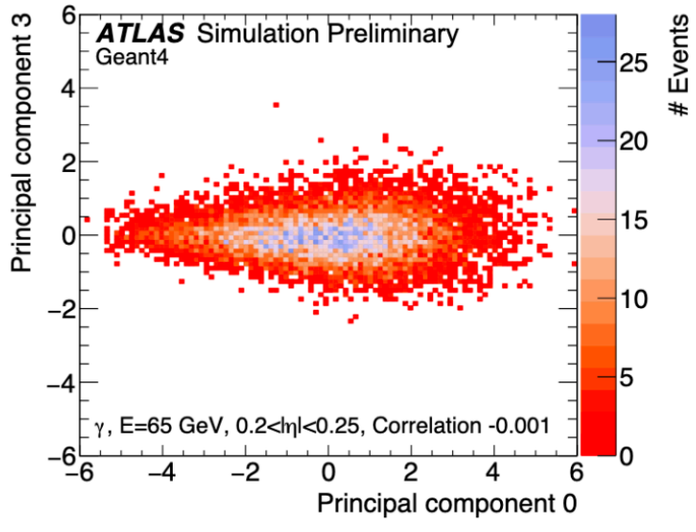
After PCA:



(a) First vs second PC



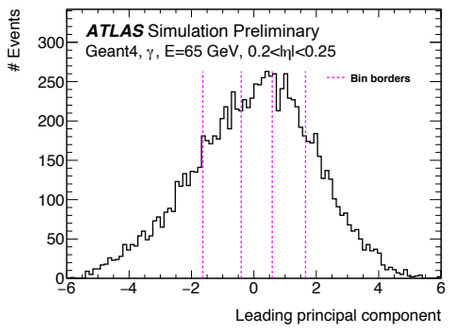
(b) First vs third PC



(c) First vs fourth PC

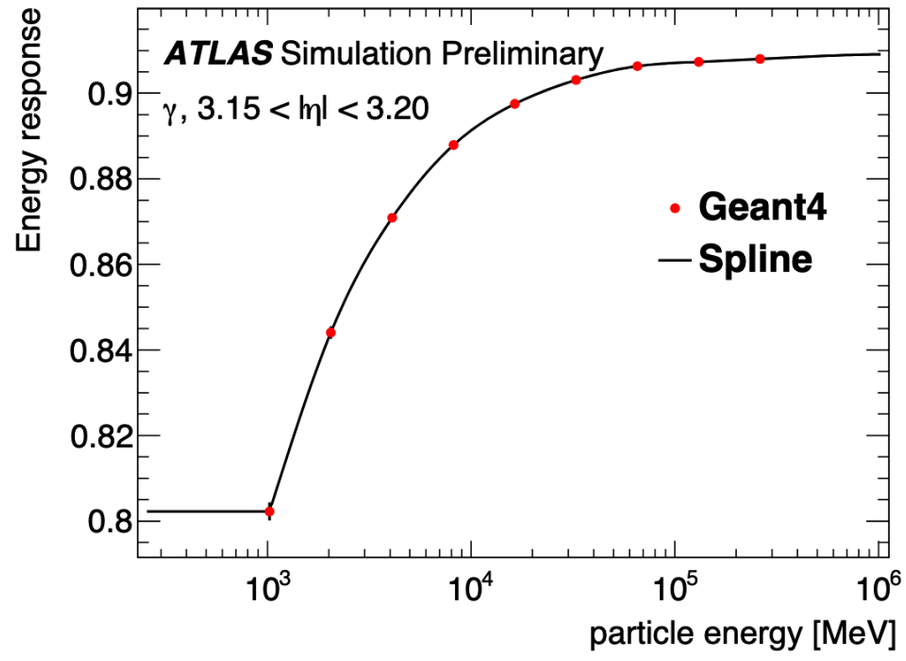
Energy Parametrization and Interpolation

Perform additional PCA on each bins of 1st Principal Component



additional PCA for further decorrelated dataset

store cumulative distributions, mean & rms of Gaussians and PCA matrix



Total energy response is interpolated using a spline

Simulation:

Uniform random Number select PCA bin

Uncorrelated Gaussian Random Numbers

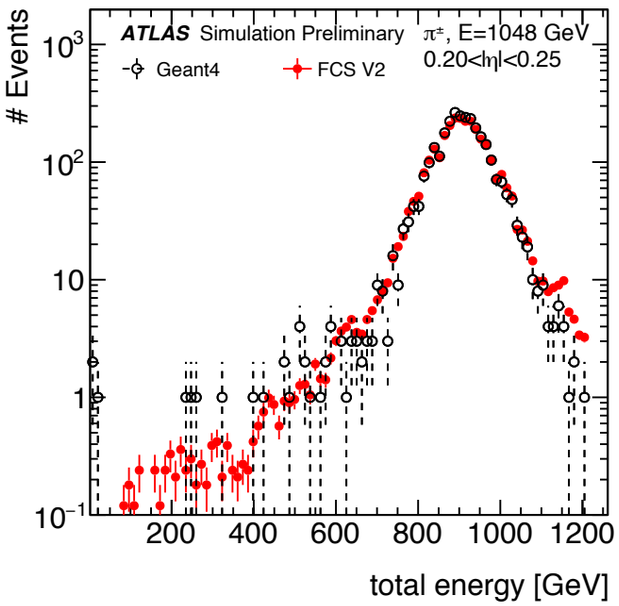
Inverse PCA matrix

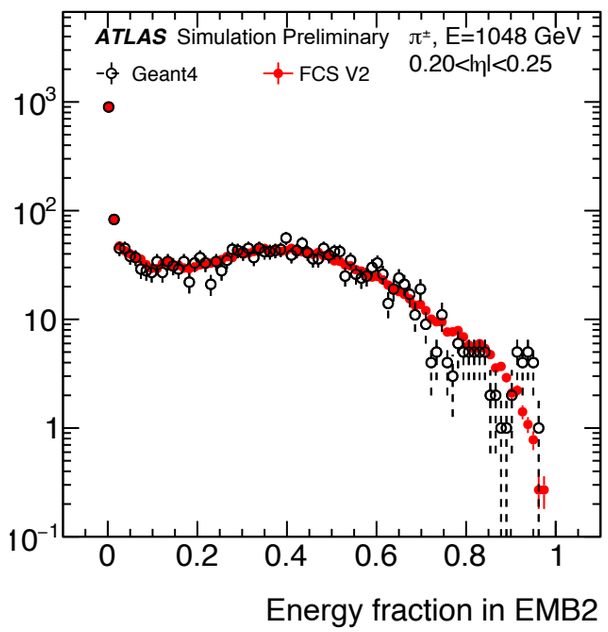
Correlated Gaussian Random Numbers

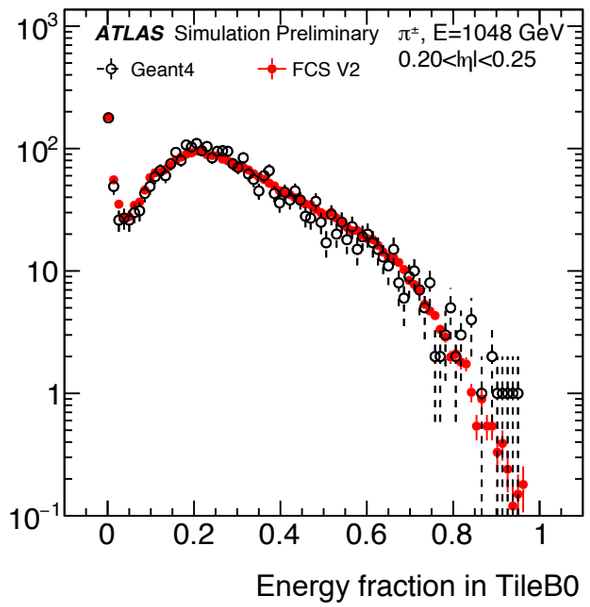
Cumulative distributions

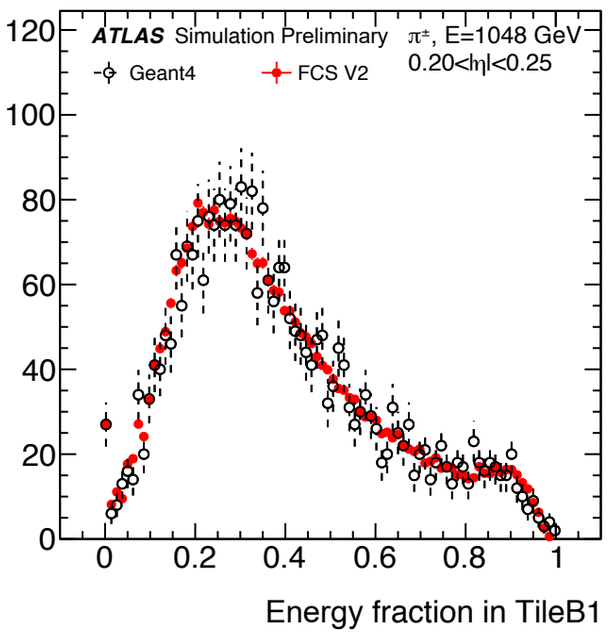
Energy Distributions

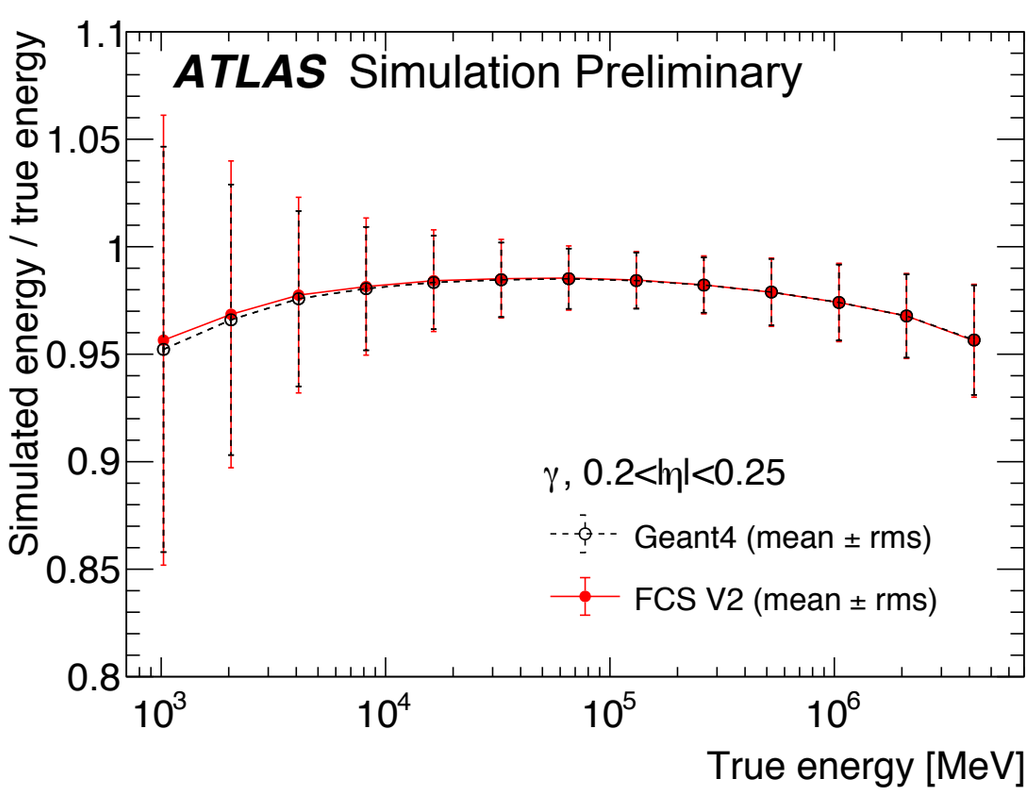
Energy Parametrization: Toy Validation

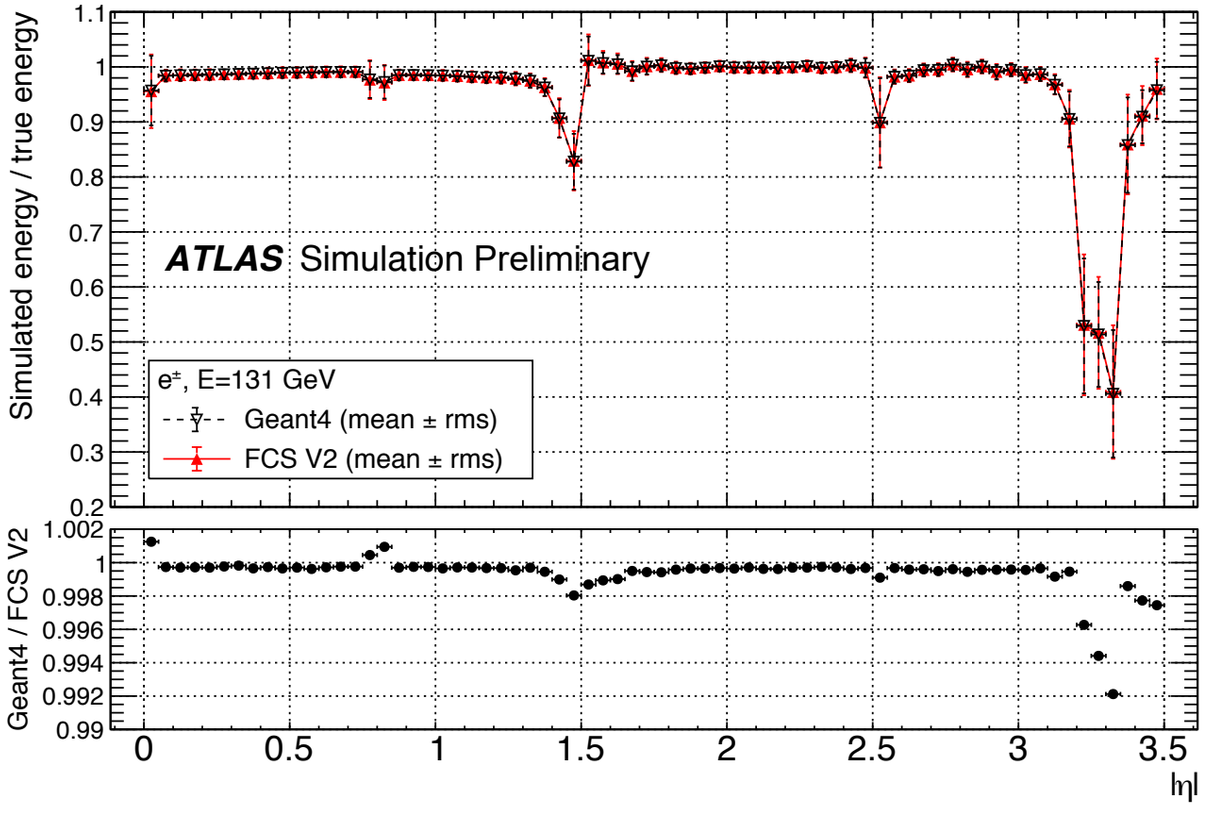






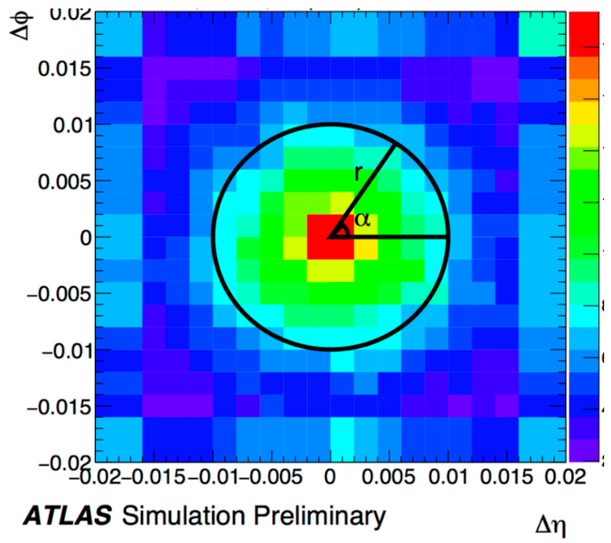






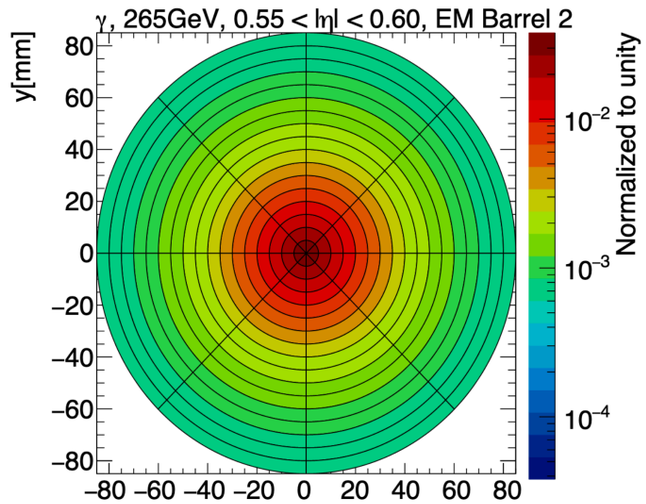
Lateral Shape Parametrization

Construct <u>average</u> lateral shower shape for each: particle, energy, eta, calorimeter layers and bins of 1st PCA



utilize symmetry

co-ordinate transform



During simulation draw N_{hits} randomly

Deposit: $E_{hit} = (E_{layer}/N_{hits}) \times weight$

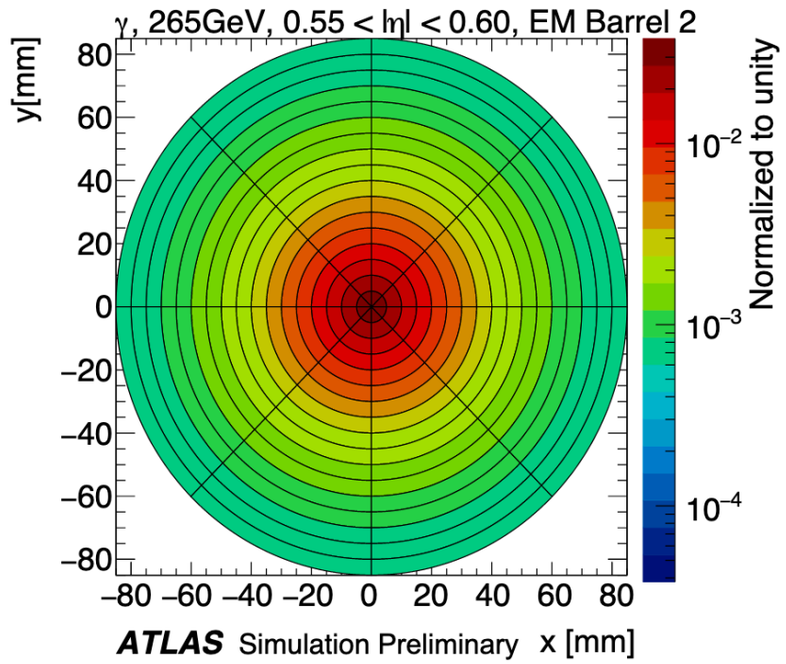
weight depends on the radial position of the hits

Random Fluctuation

Individual showers are different compared to average

Extremely important to model such fluctuation esp. for pions

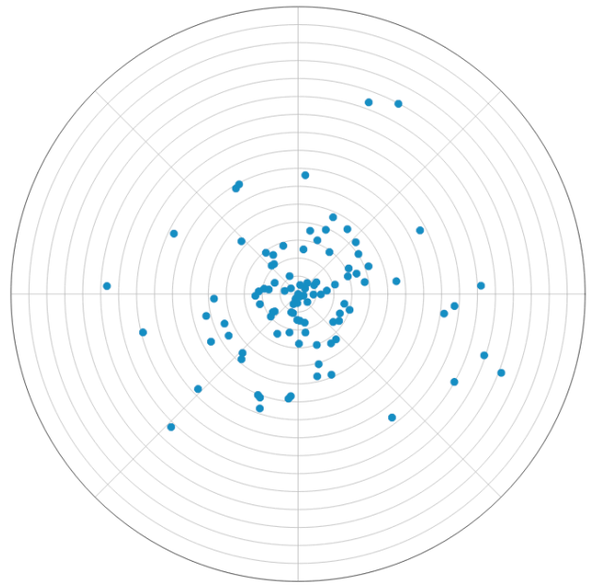
Use intrinsic resolution of the calorimeter to draw N_{hits}



Sampling Uncertainty (e.g. for EMB2): **stochastic term**

$$\sigma_E \approx \frac{10\,\%}{\sqrt{E}}$$

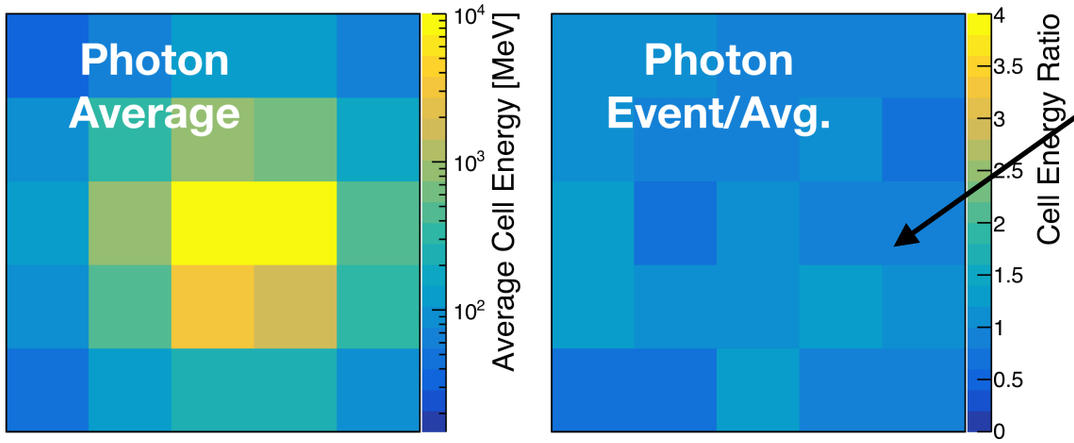
$$N_{hits}^{layer} \sim Poisson\left(\frac{1}{\sigma_E^2}\right)$$



The stochastic terms are dependent on: particles, calorimeter layers and η - calculated from special samples with energy deposits both in active and inactive materials

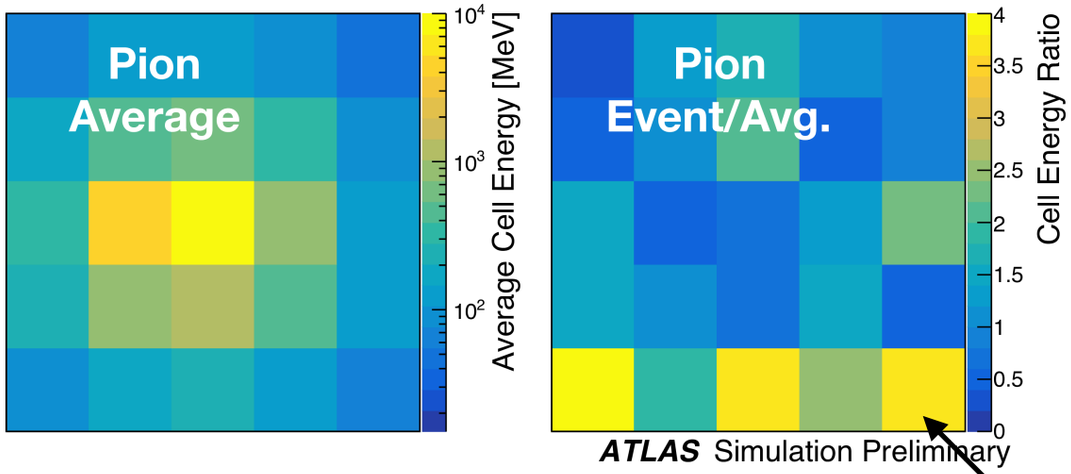
Correlated Fluctuation

Random fluctuation doesn't include longe-range correlations of energy fluctuations across cells cells



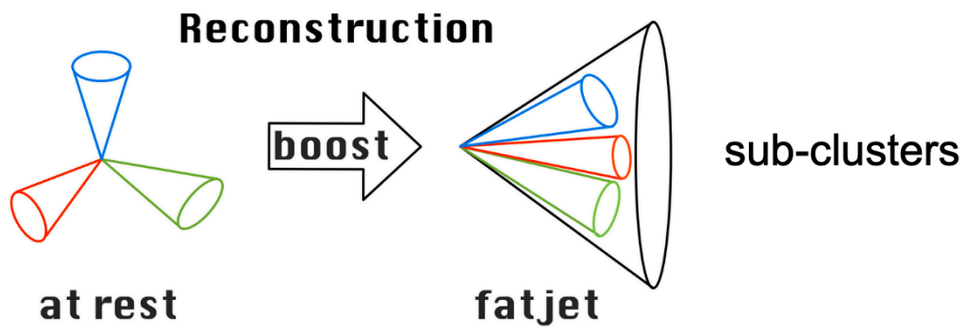
Photon event is very similar to average

Modeling these correlations well is important for modeling jet substructure



Energy ratios of single Geant4 events to their average shapes in a grid of 5x5 calorimeter cells for EMB2

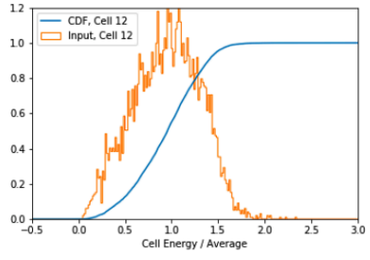
Pion event is very different!



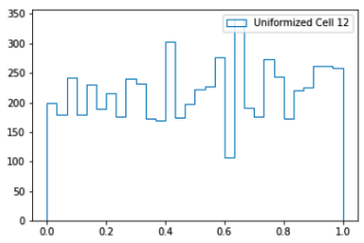
Correlated Fluctuation with Multivariate Gaussian

Use voxalized single pion shower / avg. shower cell as input

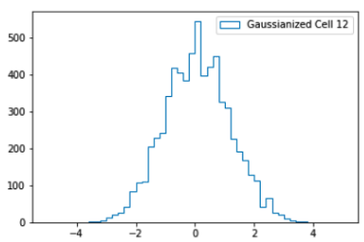
1. **Construct CDF from input distribution**



2. **Uniformize by sampling from CDF (CDF(x) for each x value in input)**



3. **erfinv(uniform) = Gaussian**



Inputs

Cumulative Dists.

Gaussianized Inputs

n-dim Gaussian (n means, n x n covariance matrix)

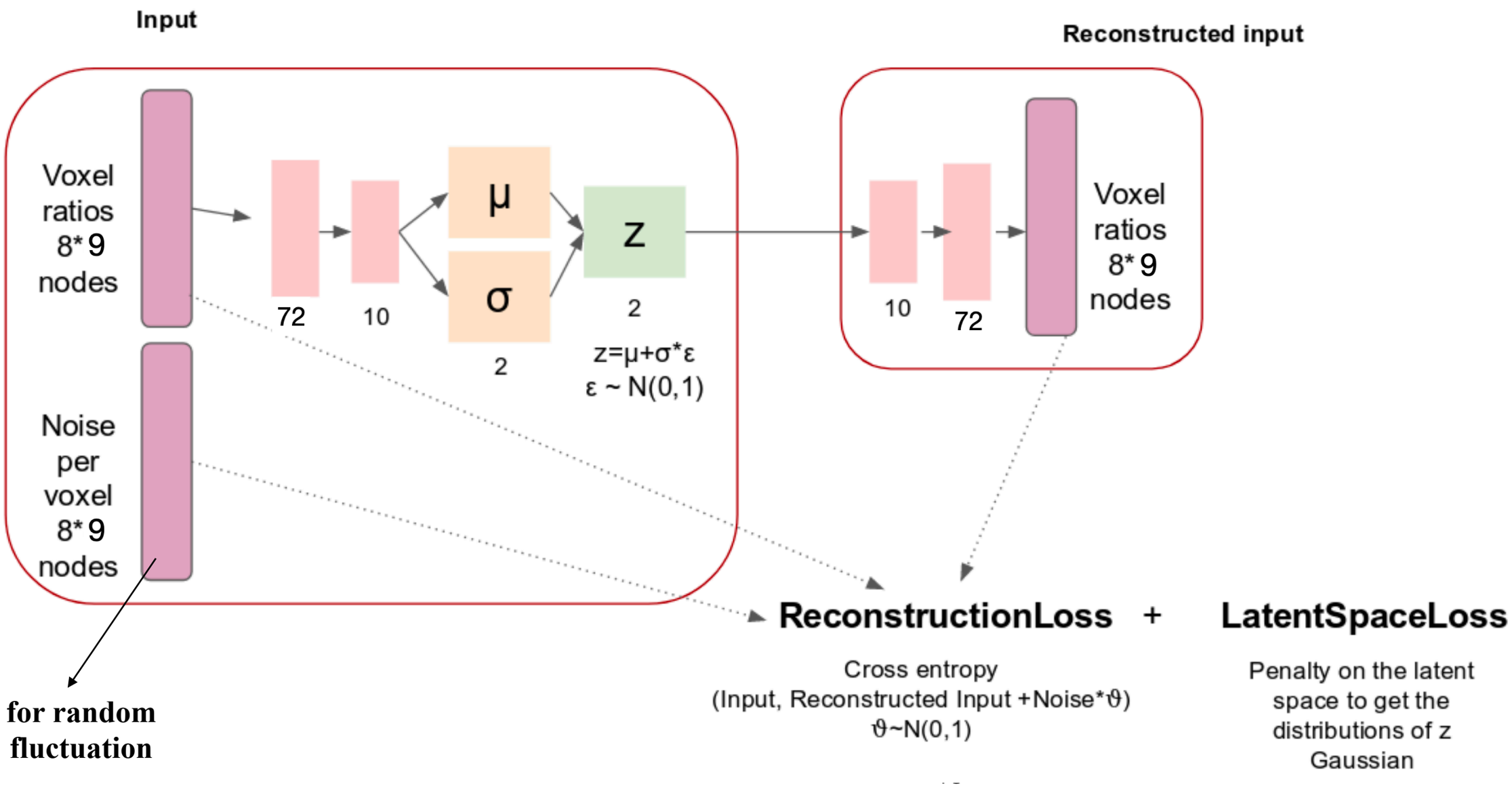
Cumulative Dists.

Outputs

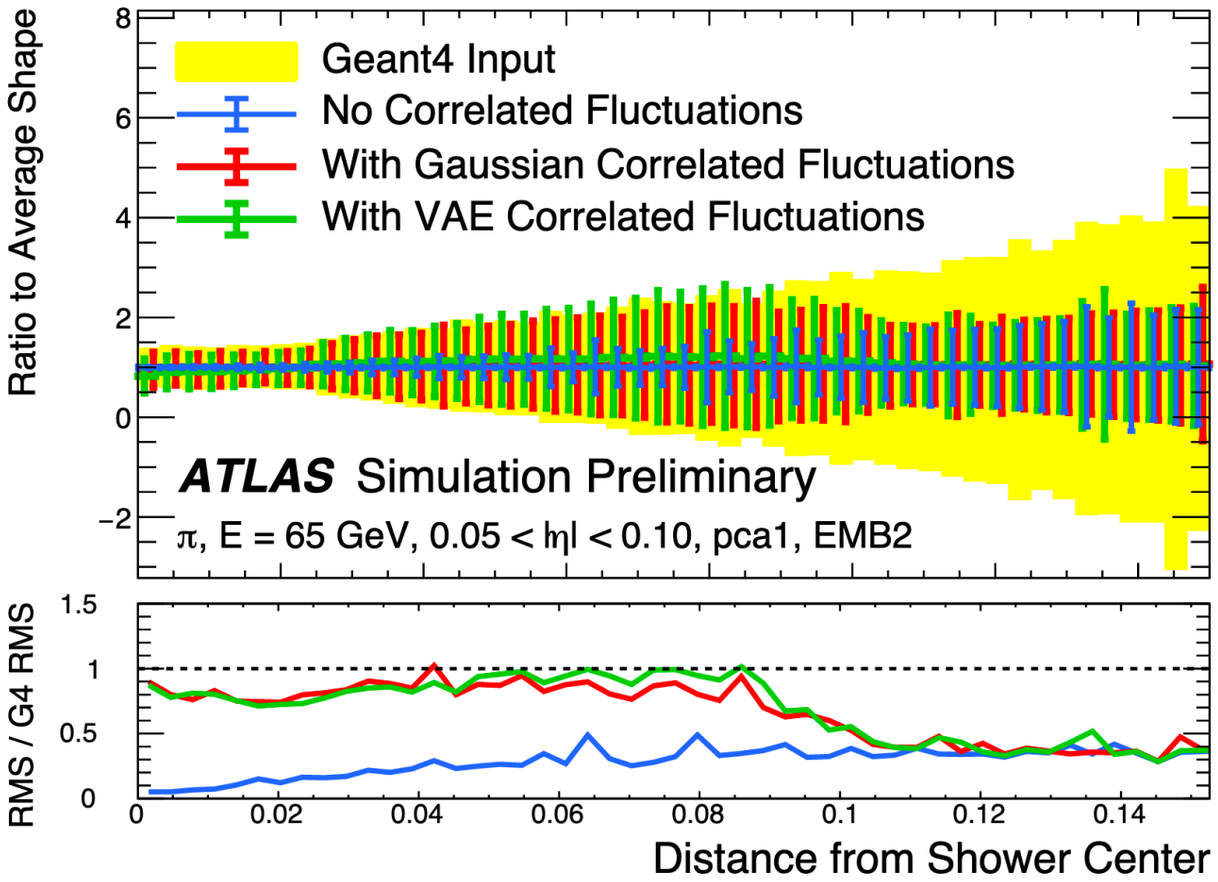
Generative Model

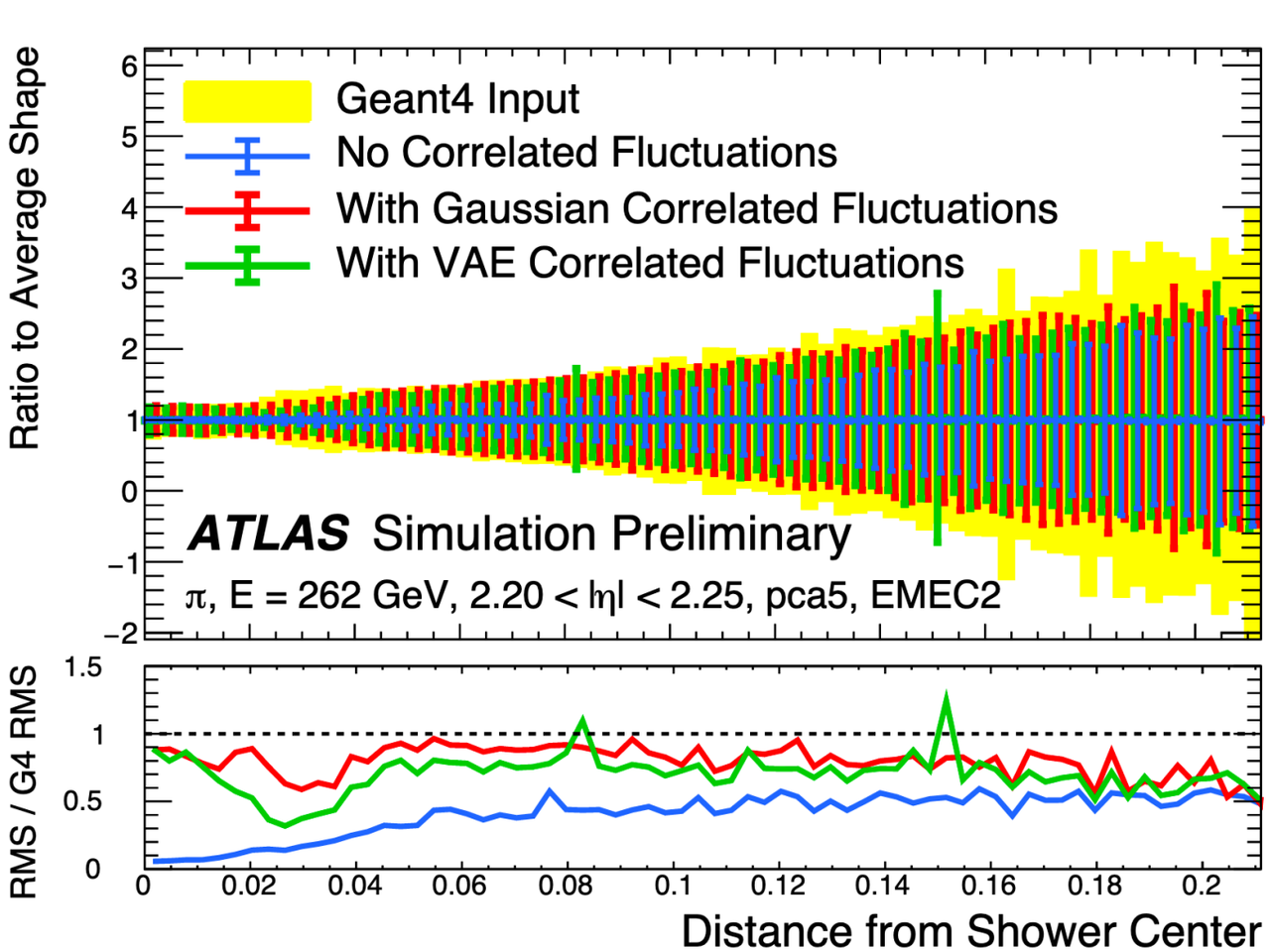
Correlated Fluctuation with Deep Learning

Use Variational Auto Encoders (VAE) with dense layers



Correlated Fluctuation: Toy Validation

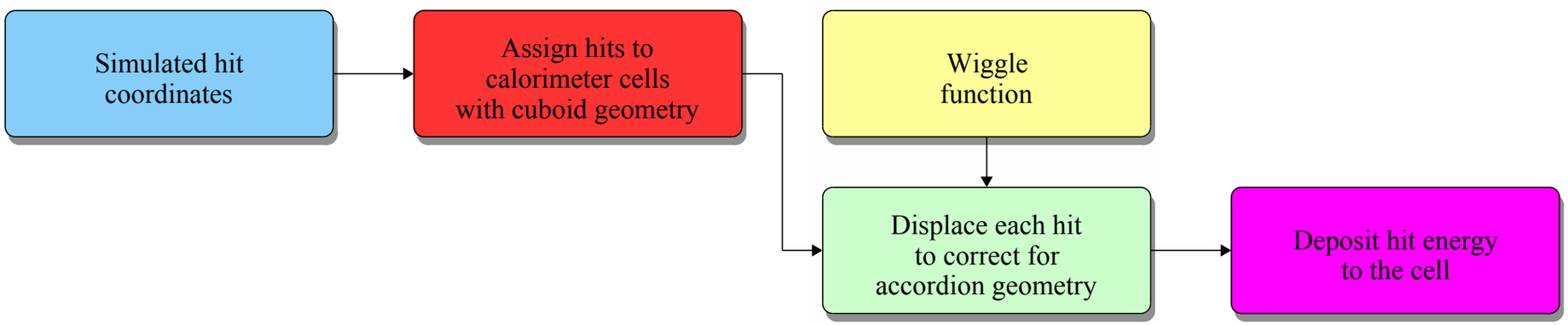


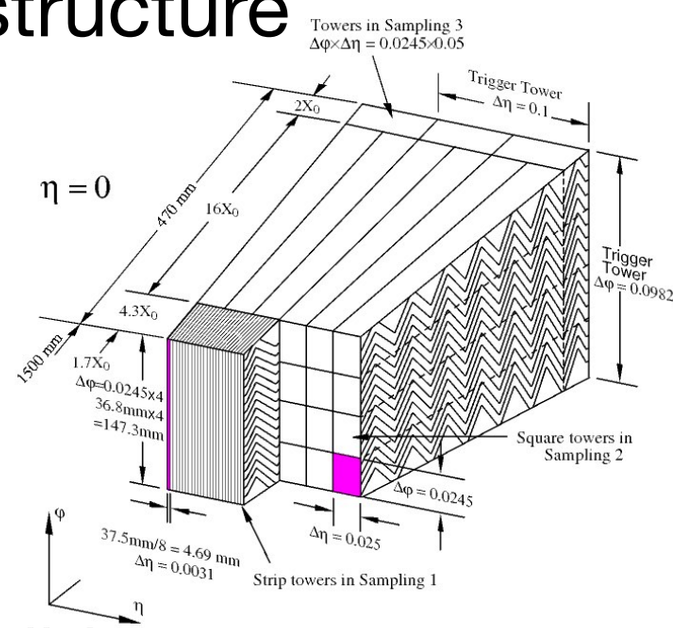


Hit to Cell assignment

Simulated hits assigned to cells assuming simplified cuboid geometry

Electromagnetic calorimeter have accordion structure

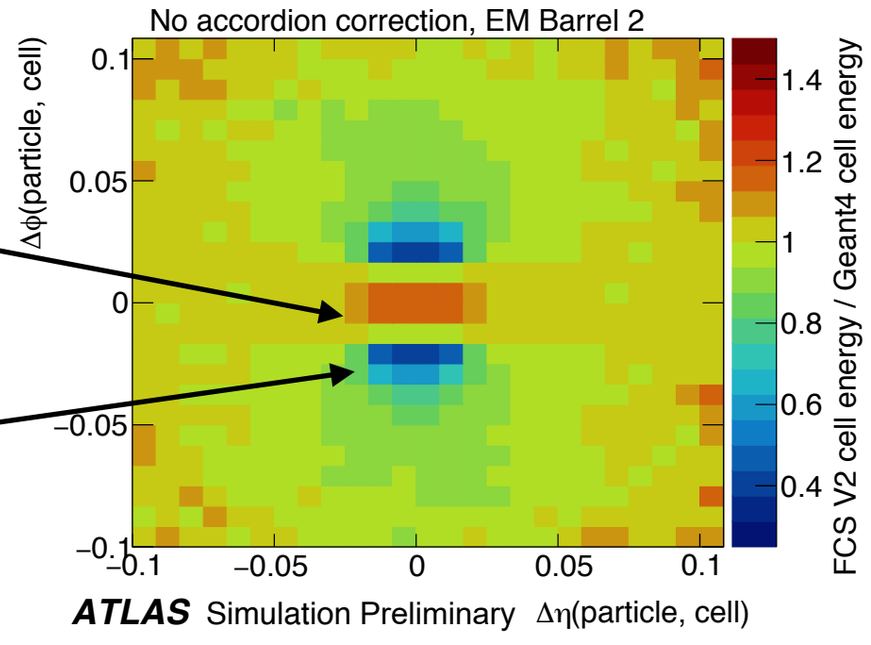


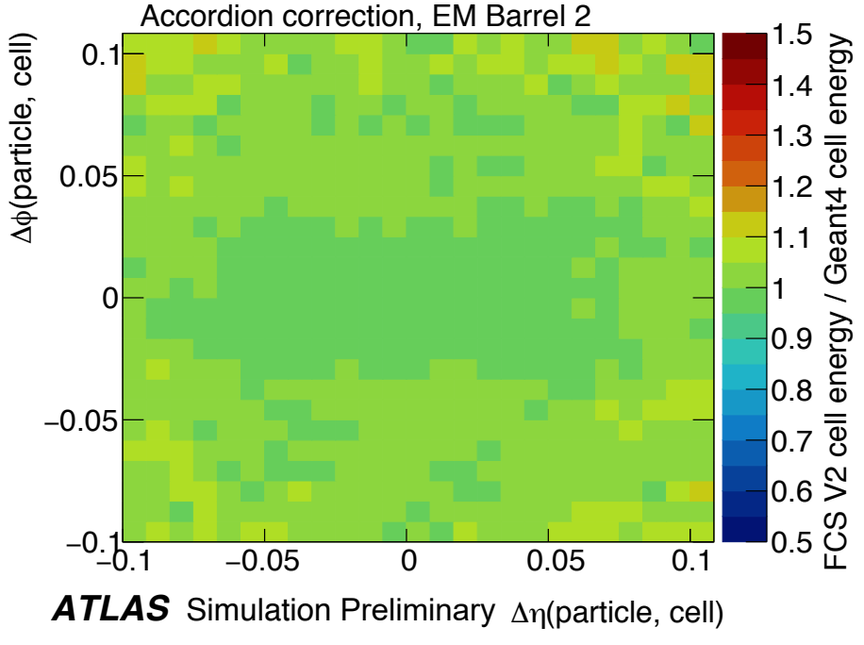


Wiggle the hit position based on a function describing the probability that a hit belongs to a neighboring cells

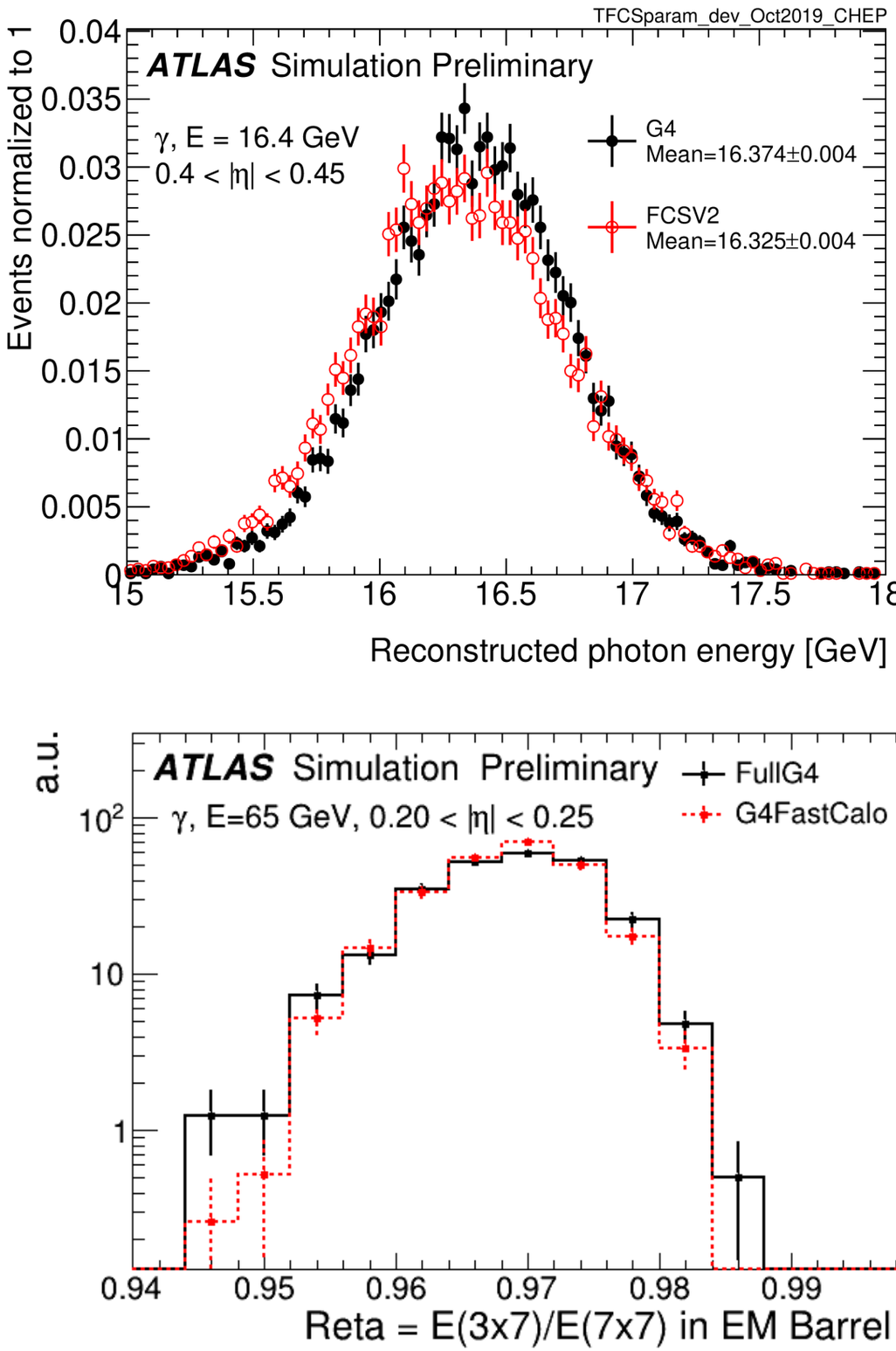
too much energy

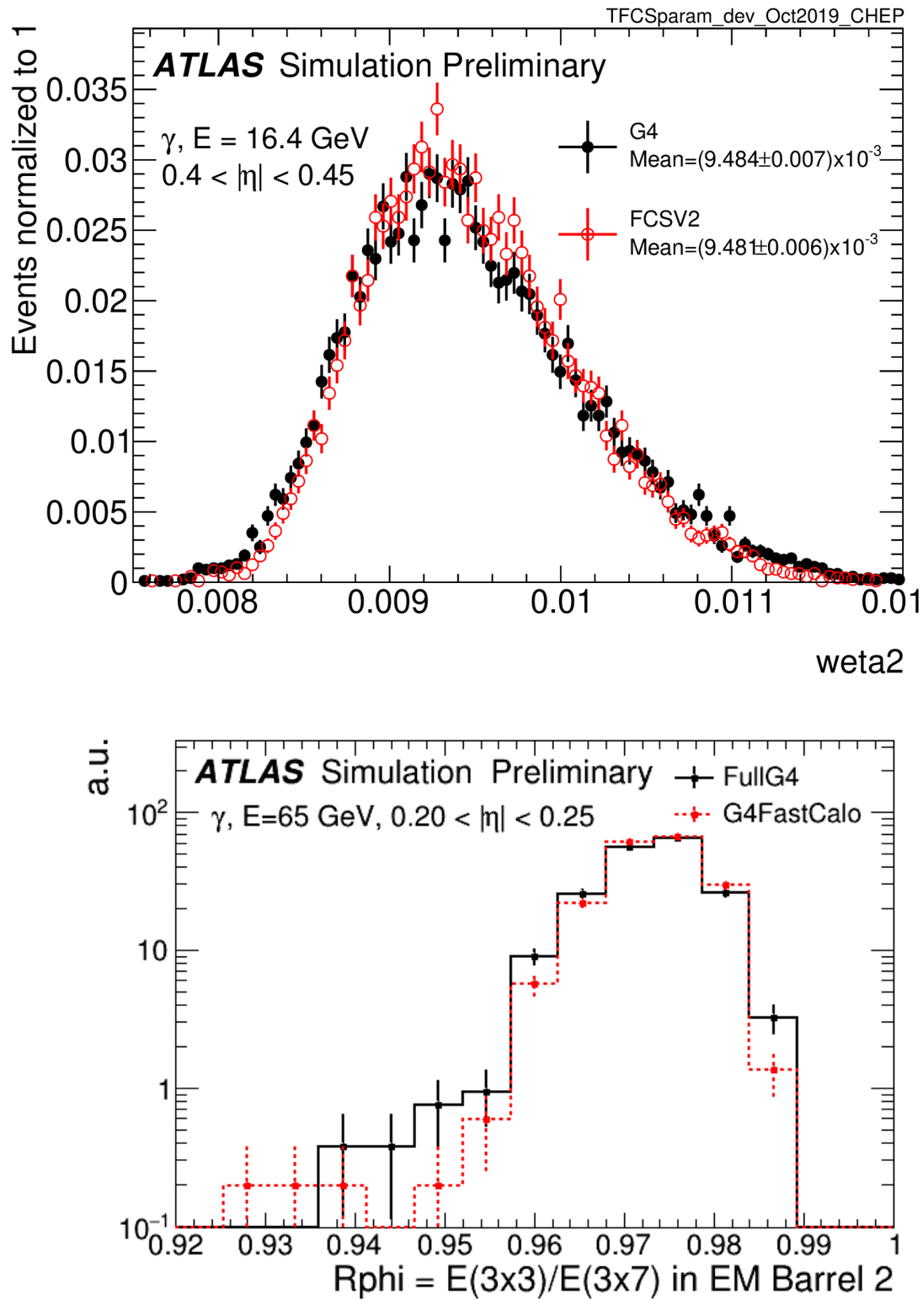
not enough energy



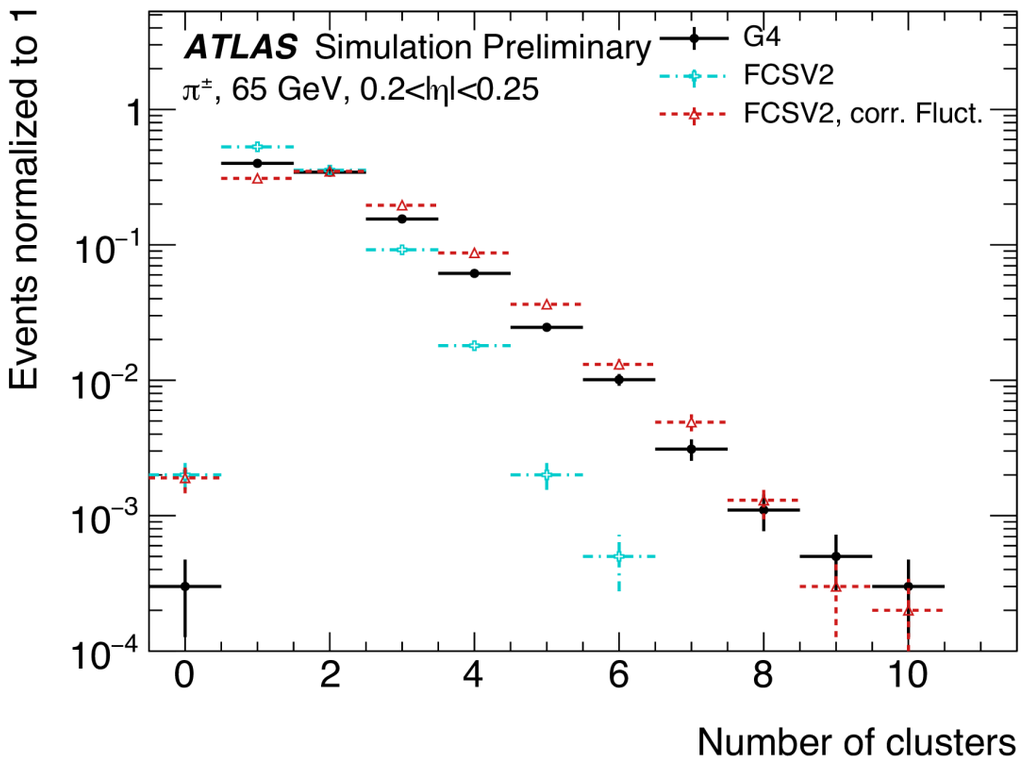


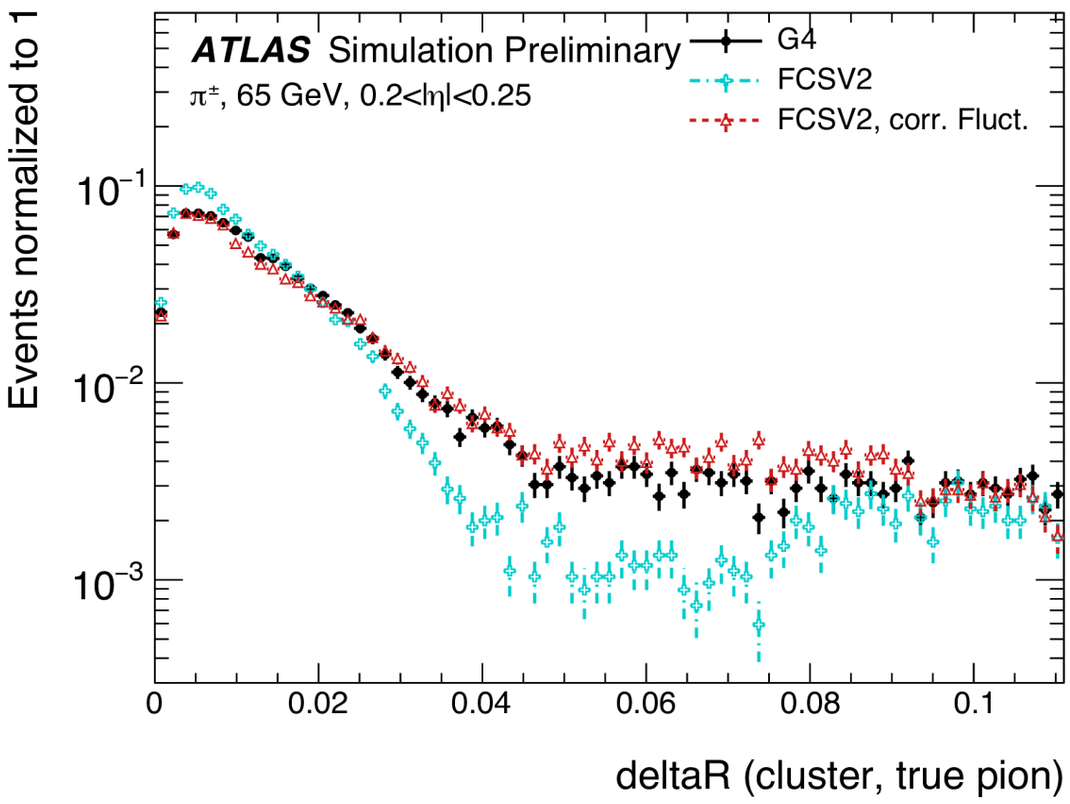
Photon Validation

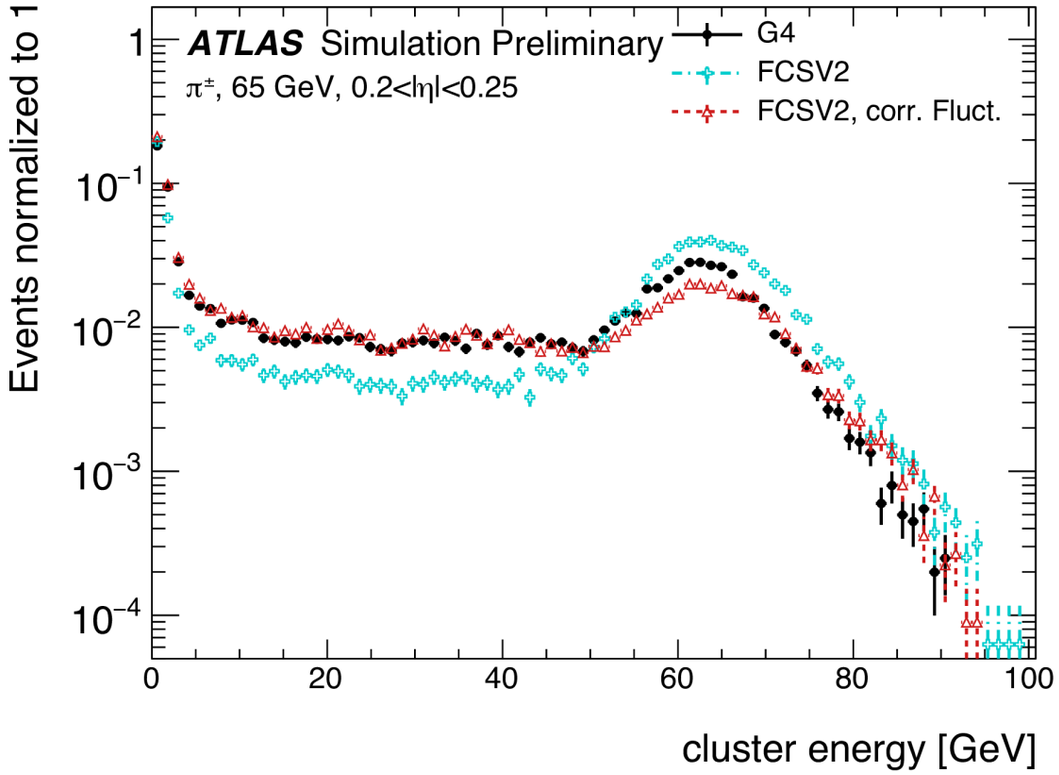


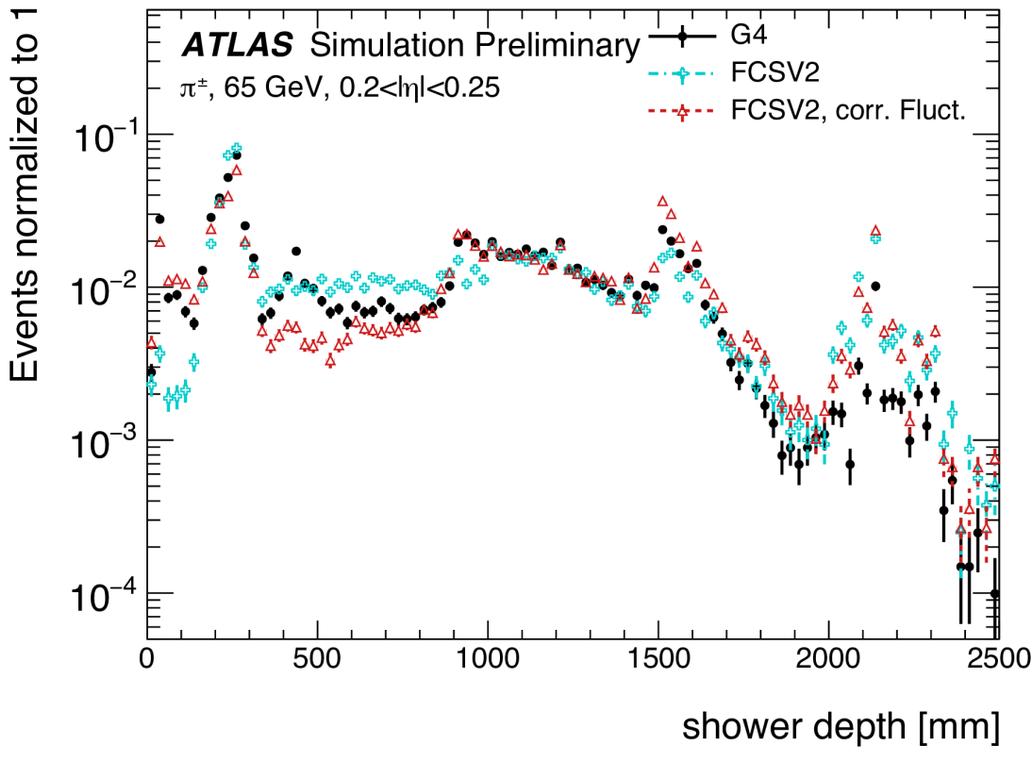


Pion Validation





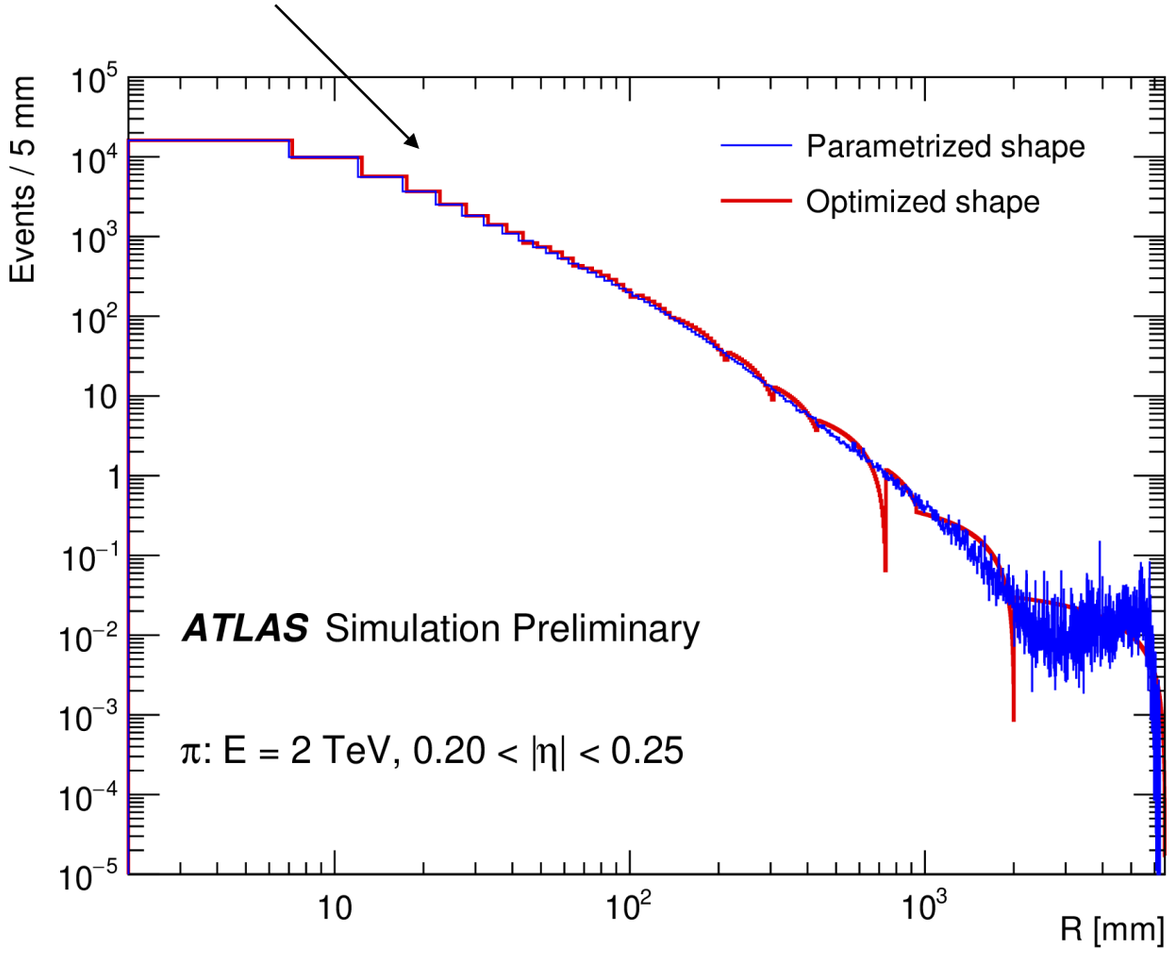




Memory Optimization

Replace high memory consuming histogram(ROOT::TH2) classes with efficient custom classes

Algorithm developed to perform optimized rebin, interpolation between bins and representing bin contents and bin edges in reduced (e.g. 8-, 16-bit) formats

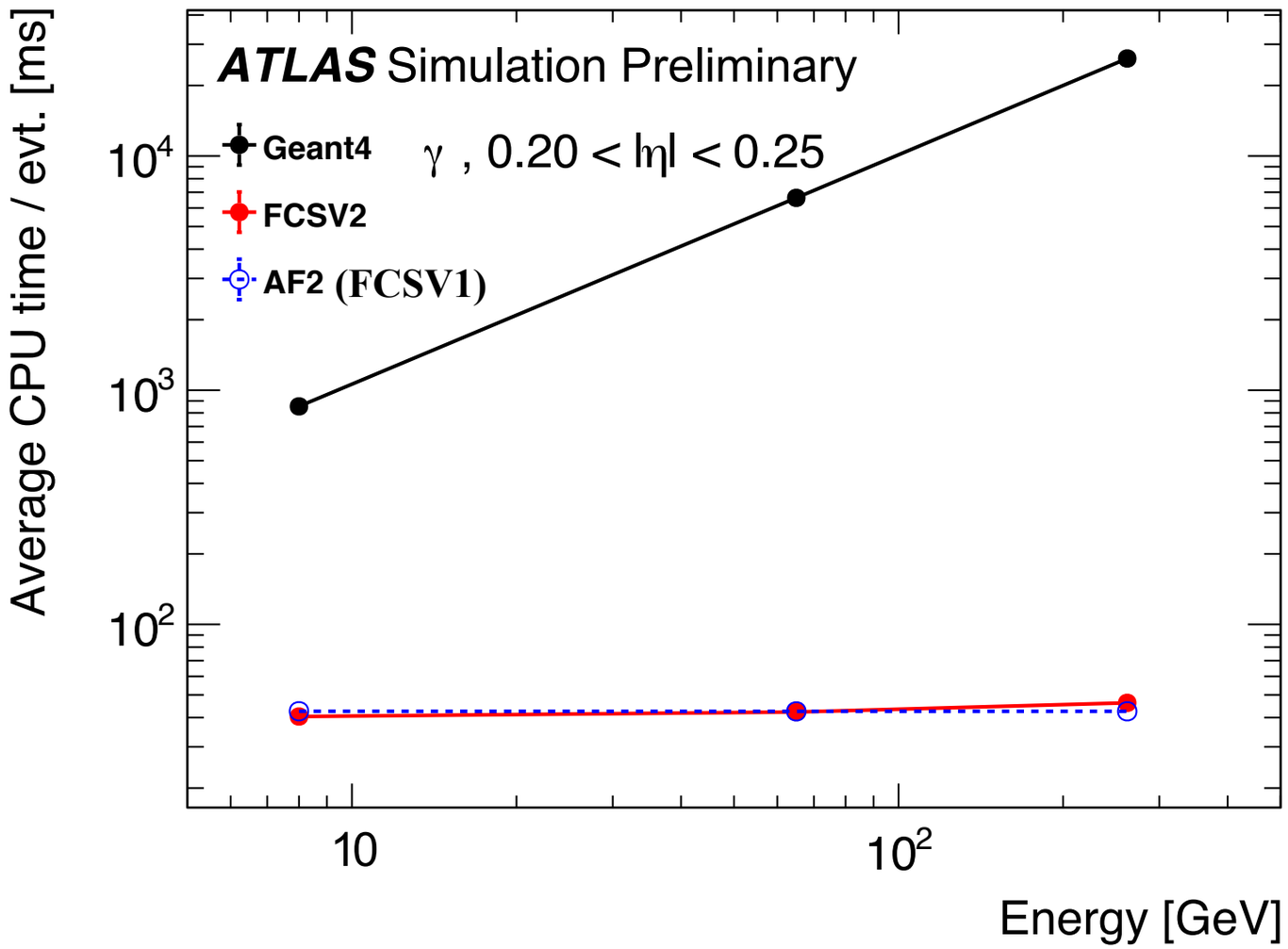


Parametrized shape:
- 1200+ bins of 5mm
- 43 Kbytes in memory

Optimized shape:
- 15 non-uniform bins in the same range with interpolation inside each bin
- 16-bit for bin edges
- 32-bit for bin content
- 353 bytes in memory!

CPU Performance

Single particle simulation compared to Geant4 and FastCaloSimV1

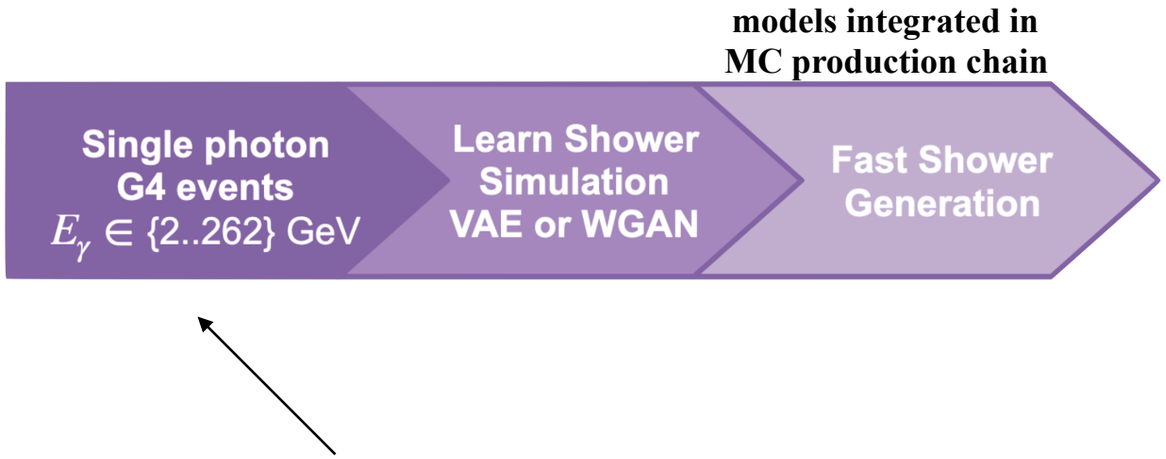


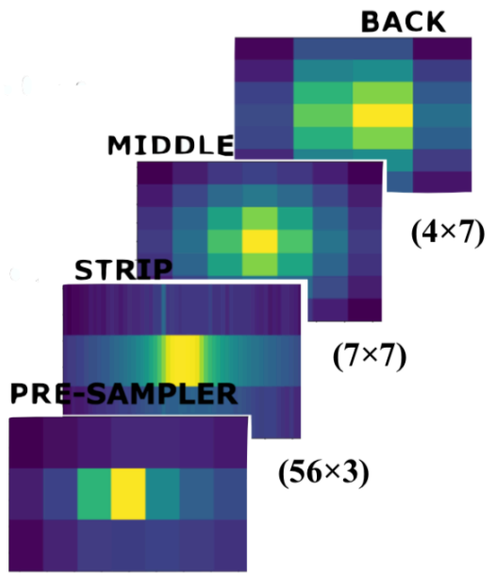
Particles are generated on calorimeter surface

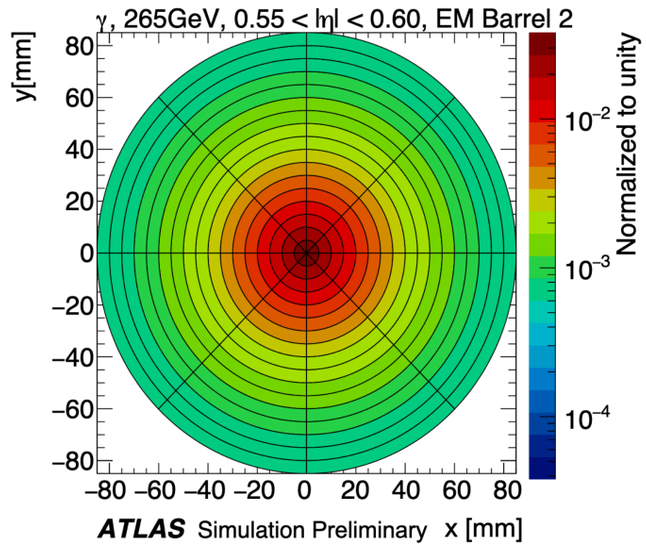
A factor of ~ 10 -25 times faster than Geant4

FastCaloSim with Deep learning : DNNCaloSim

Can we train deep networks to approximate showering in Geant4 ?







Use cell-level informations or voxalized hits (as in FastCaloSimV2)

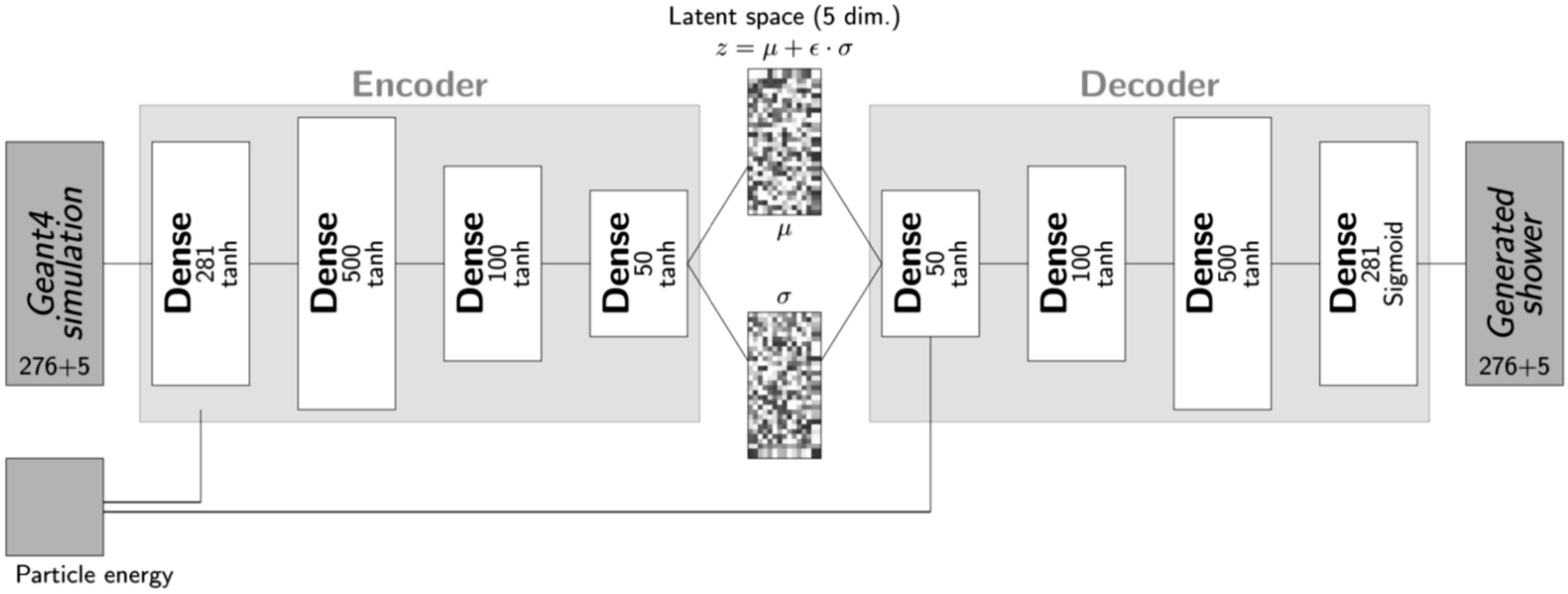
A deep network could learn:

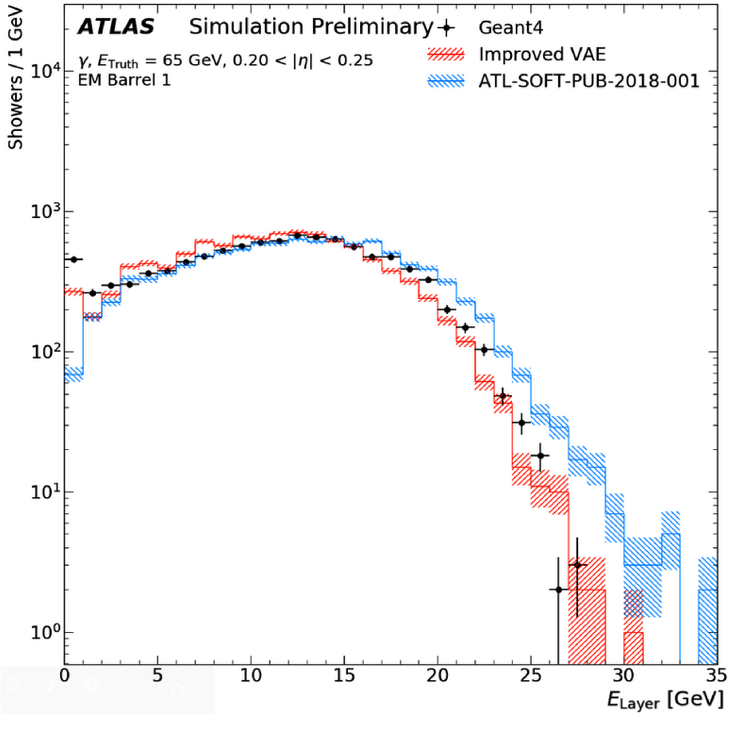
- energy correlations
- correlated fluctuations
- interpolate between energies
- interpolate between η

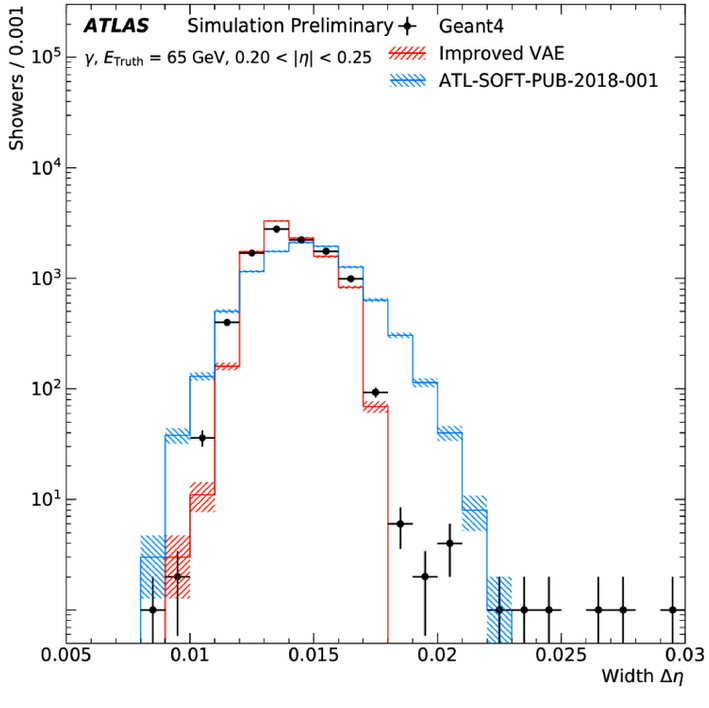
How big would the network be ?
CPU performance ?
Physics performance ?

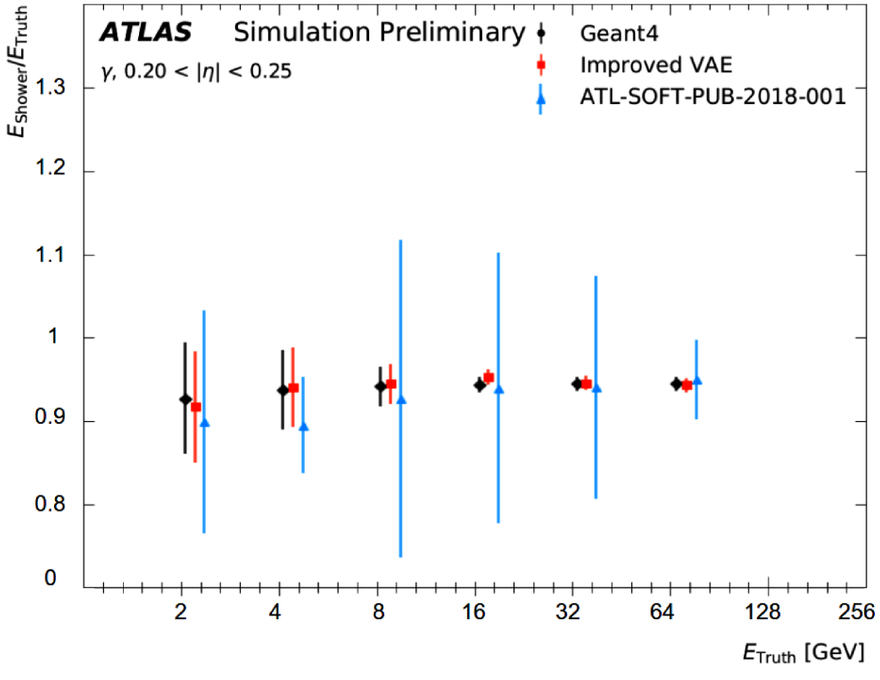
Variational Auto Encoders

train with cell level information





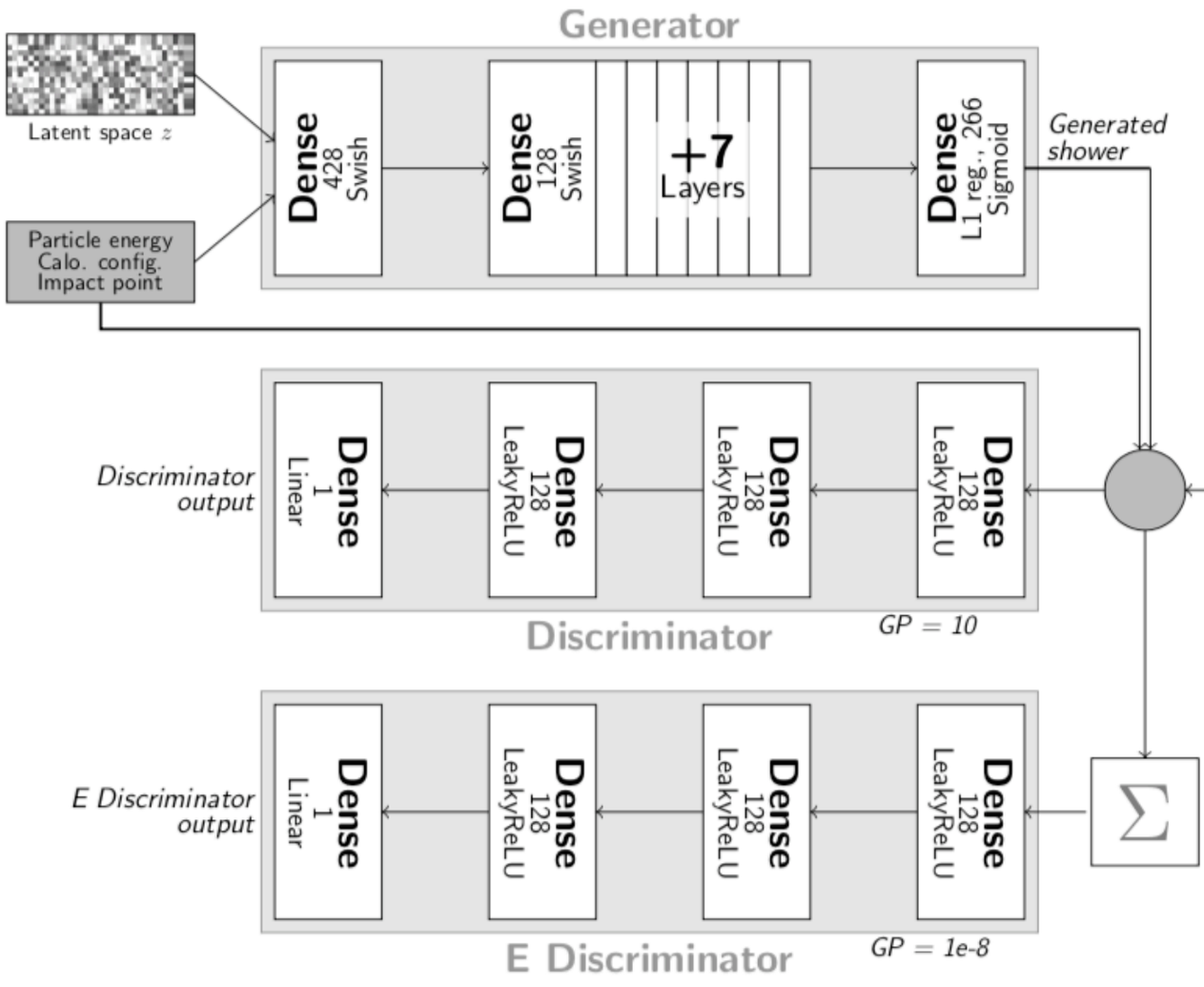


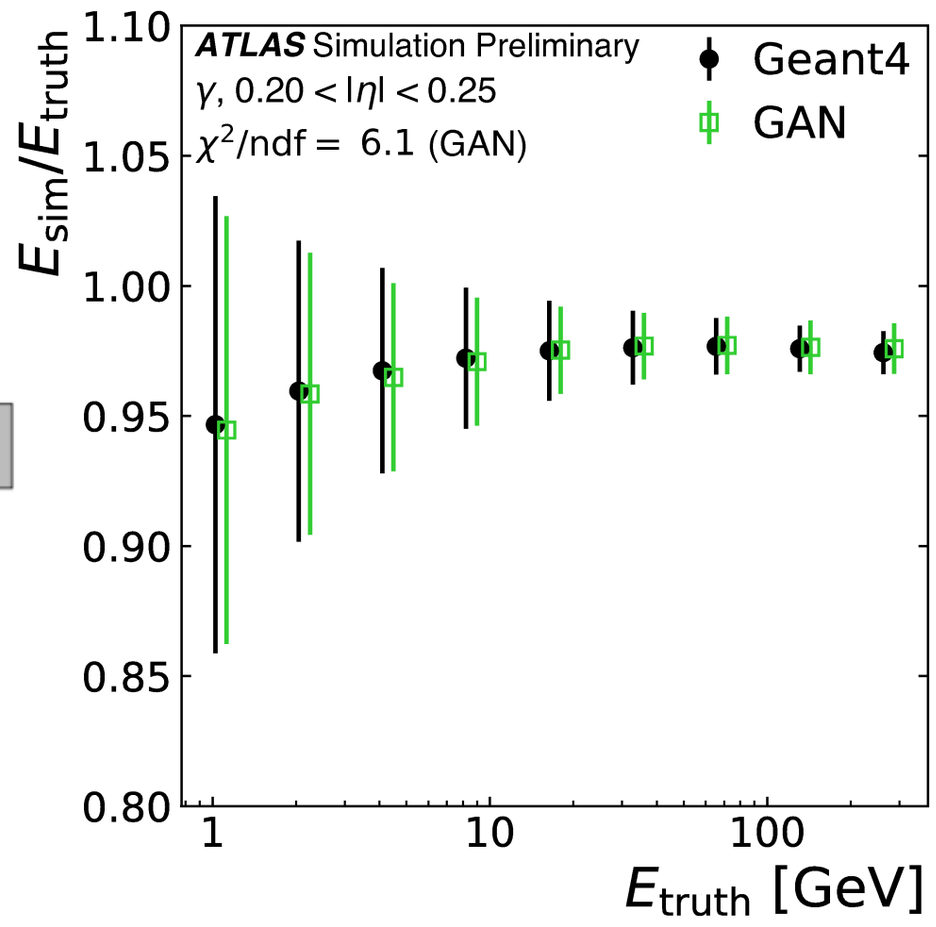


Generative Adversarial Network (GAN)

Use a WGAN with gradient penalty

train with cell level information

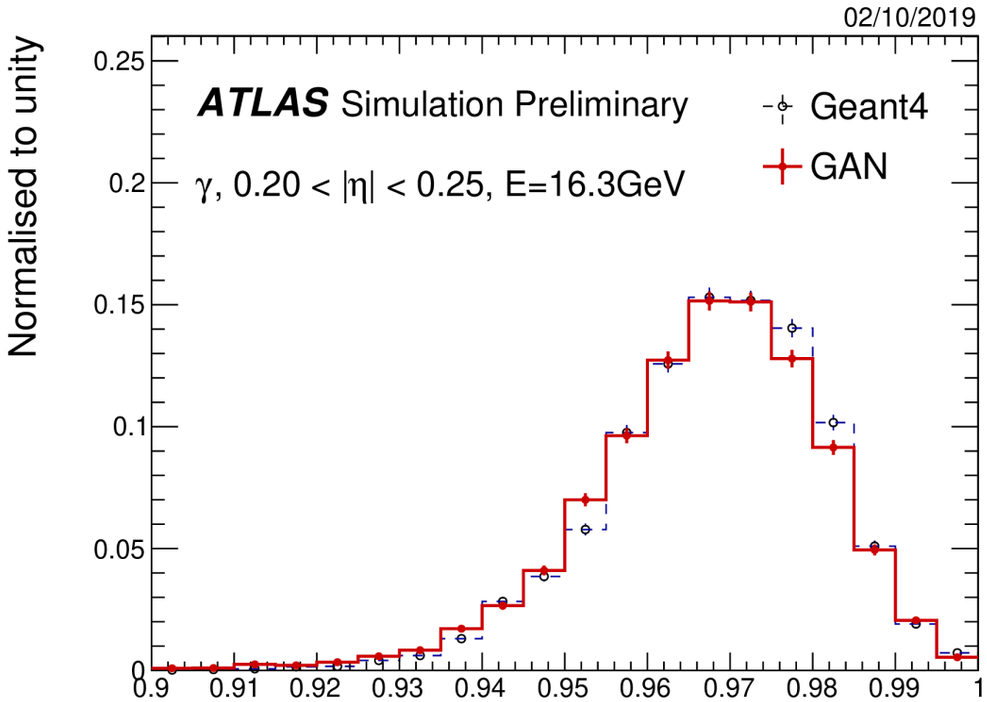


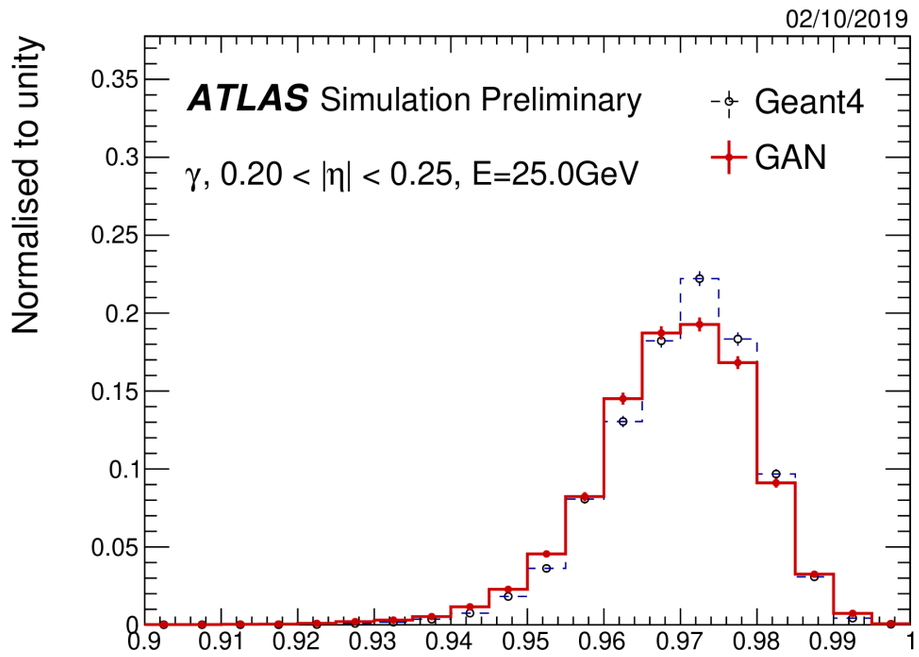


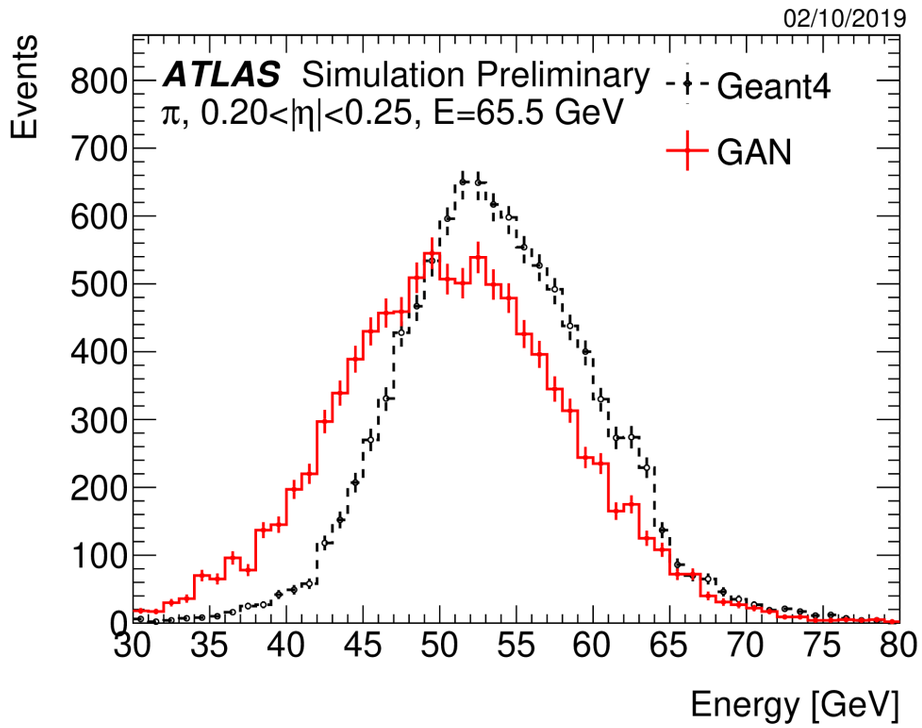
GAN trained with hit level information shows very good agreement even in pions - results coming soon!

DNNCaloSim Validations

Models integrated in ATLAS MC Production Chain







FastCaloSimV2 in GPU

Identify feasibility & challenges with GPU porting

Data structures modified from CPU to GPU

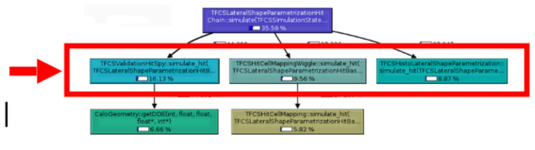
TFCSLateralShapeParametrizationHitChain::simulate(){

…

Loop on **nhit**

…

Loop on hit_simulation_chain



Call

End Loop

…

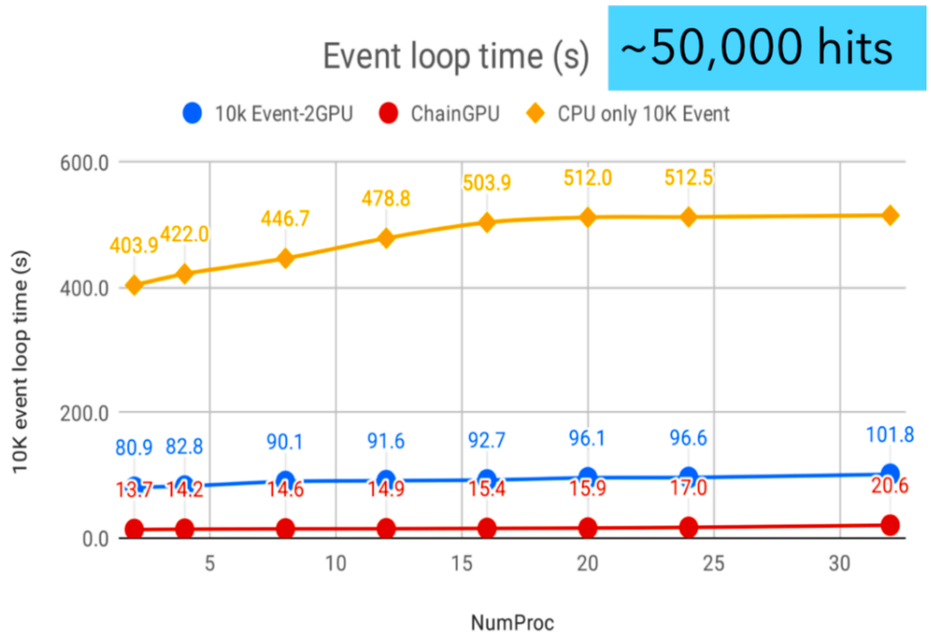
End Loop

…

}

Possible Parallelization

Impossible Parallelization because of data dependency



Actual simulation have fewer hits ~ 2-3X gain vs. CPU

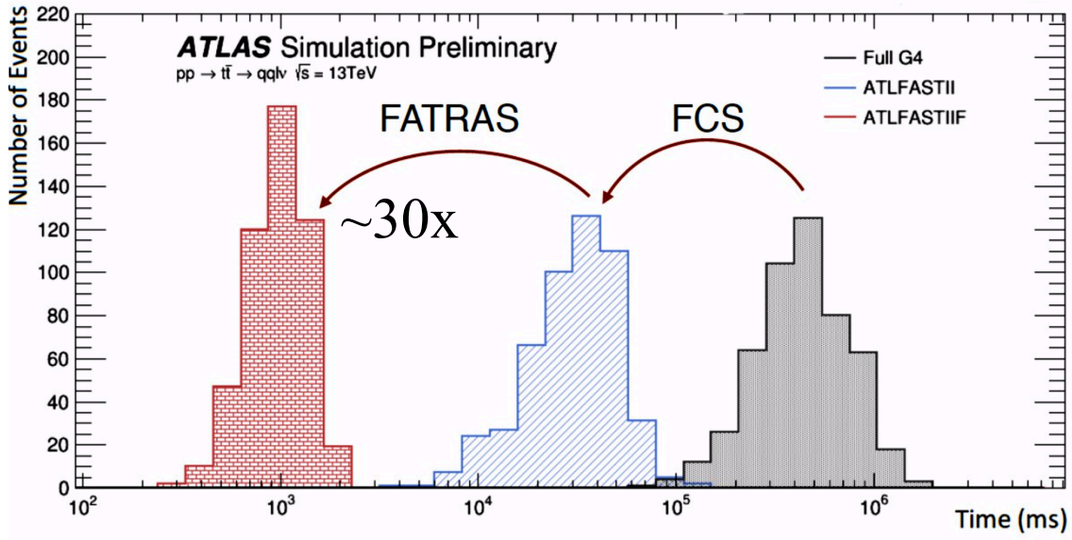
Simulation				~5,000 hits	
#MPI Processes	Particle	Energy	Min Eta	CPU (s) /10K event / process	GPU (s) / 10K event / process
1	Electron	65536	2.2	**18.8**	**6.0**
32	Electron	65536	2.2	**24.0**	**7.1**

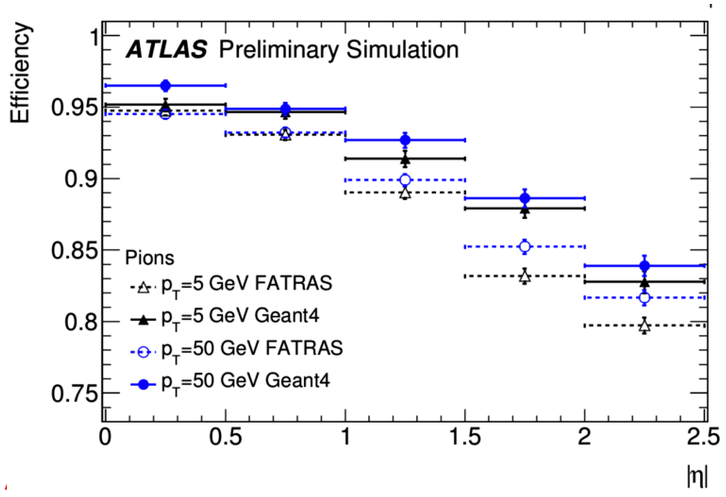
* CPU: Intel Xeon "Broadwell" 32 cores per node
* GPU: 2 NVIDIA P100 per node

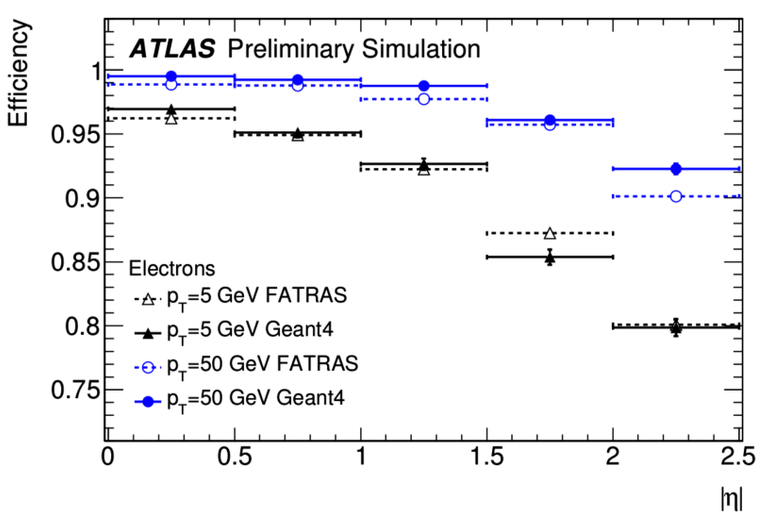
Fast Tracking Simulation (FATRAS)

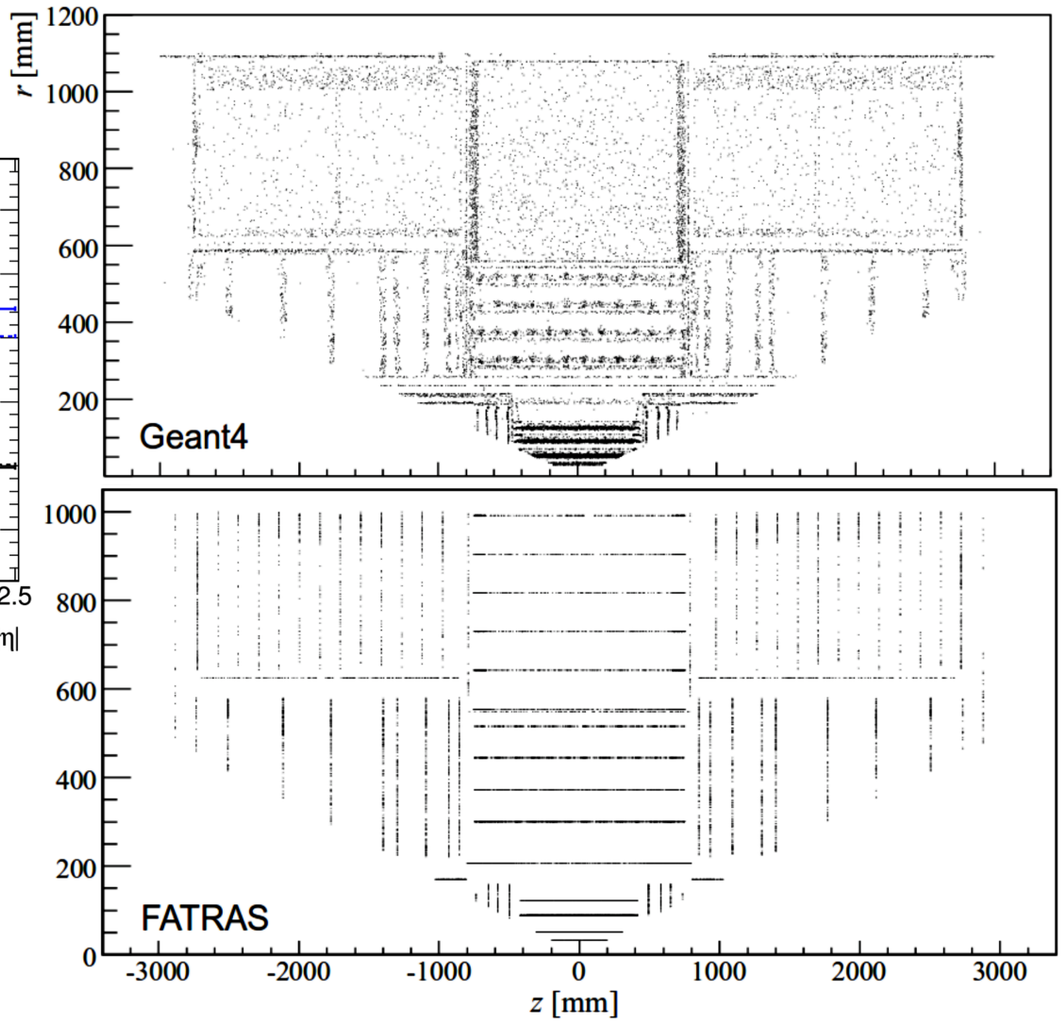
Can provide ~30x improvement in addition to FastCaloSim

- Use simplified reconstruction geometry model
- Use extrapolation engine for track parameter transport
- Material effects are applied according to the amount of traversed materials and physics processes









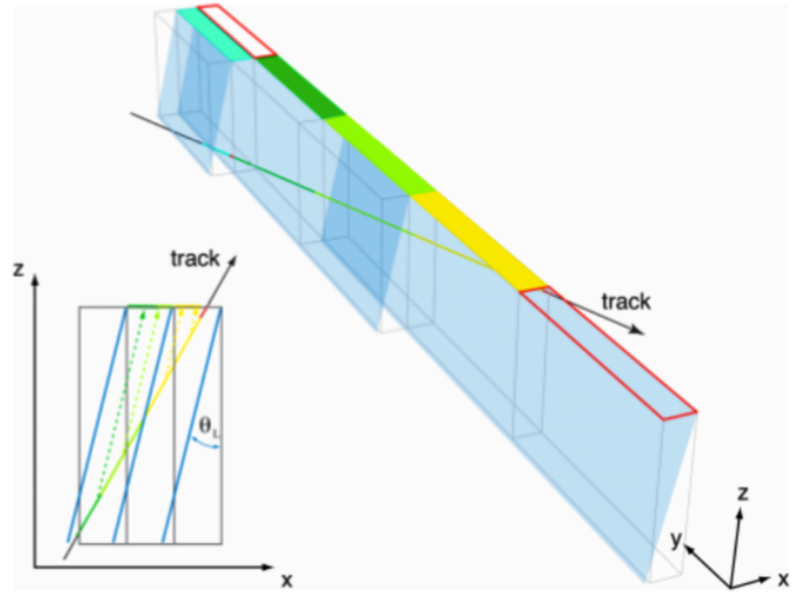
Will be used only in Inner detector

Fast Digitization

Charge collection mechanism is different for sub-detectors

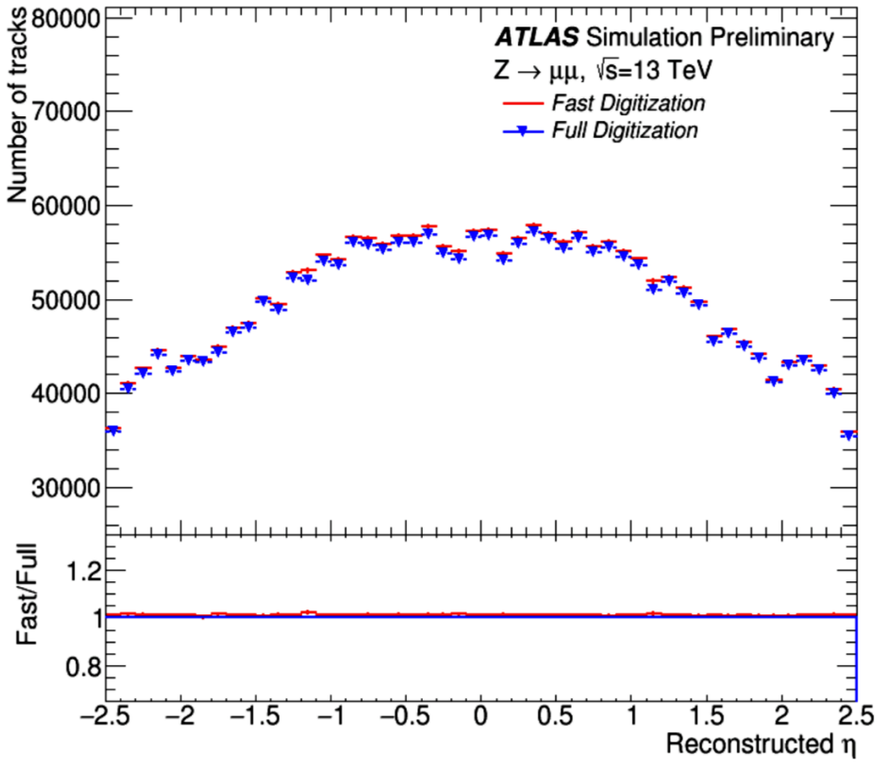
Silicon Trackers:

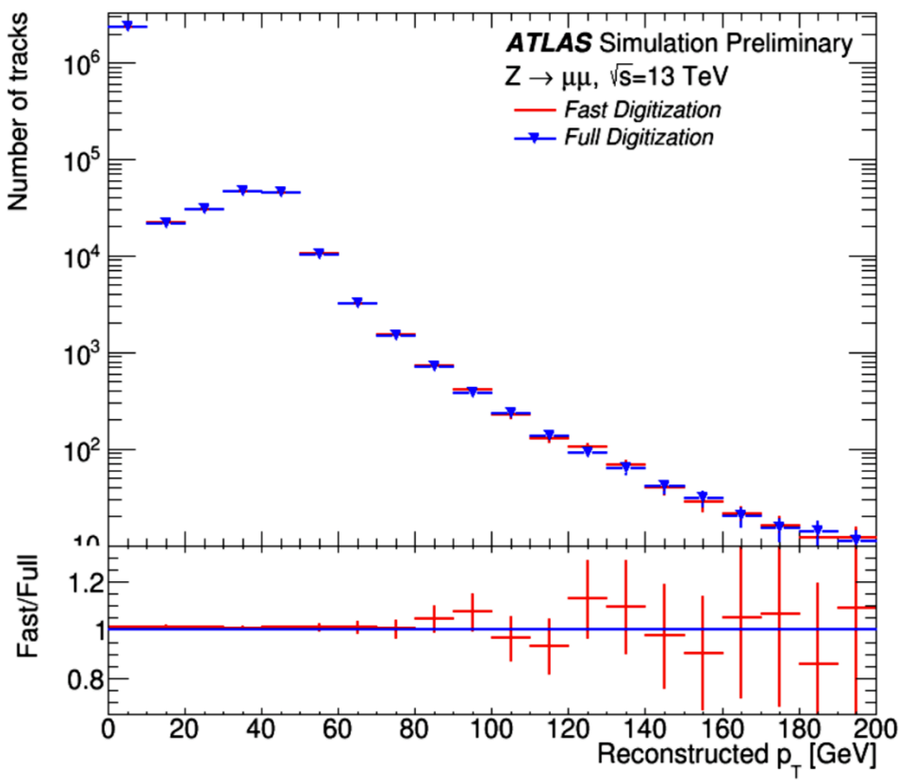
- Local entry and exit point in the detector module from detector simulation
- Evaluate the steps in each sensor, charge deposited in each pixel is proportional to the step
- Project the charge on the surface taking Lorentz shift into account
- Form clusters directly from track information (no cluster finding algorithm)

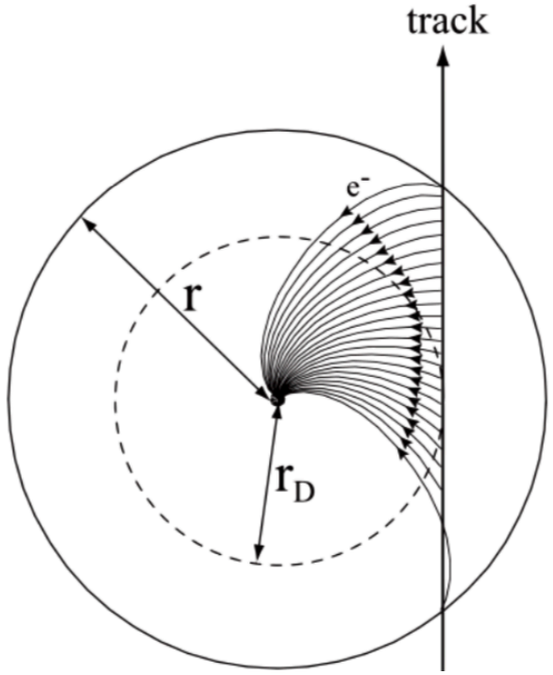


Transition Radiation Tracker:

- Emulate response from the radius of closest approach
- Take uncertainty into account by smearing the track to wire distance
- Parametrize response for particle ID



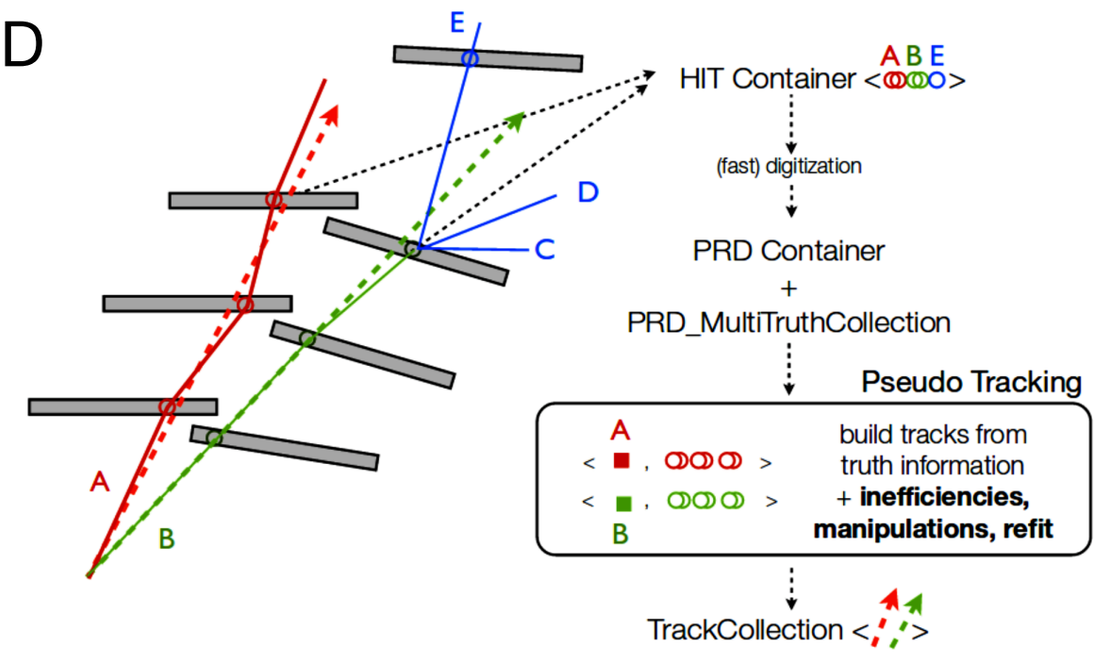


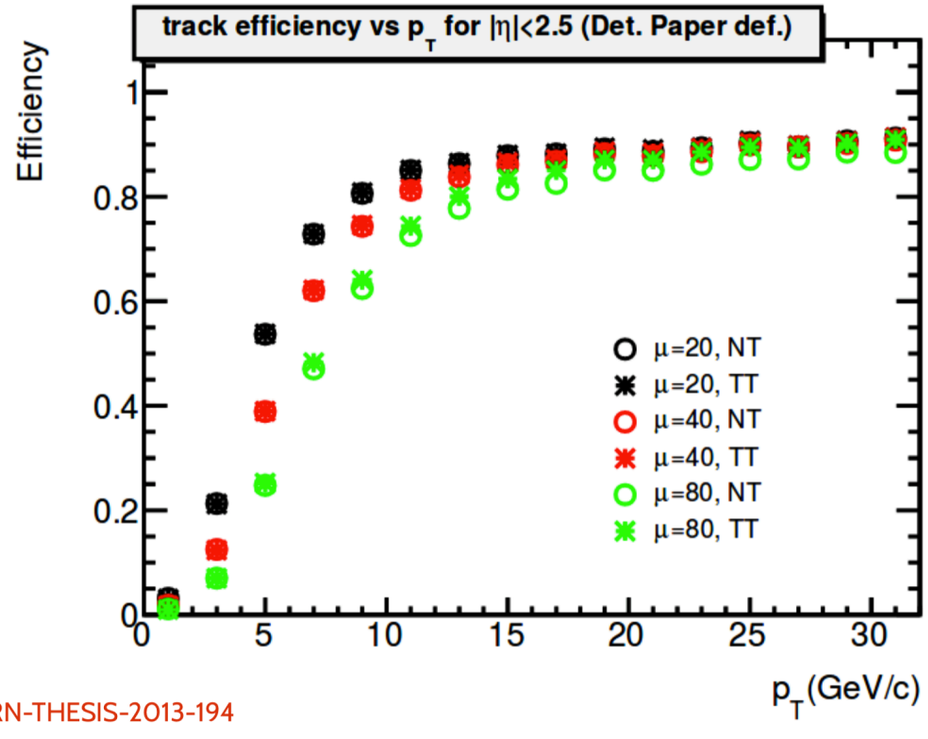


Fast Reconstruction

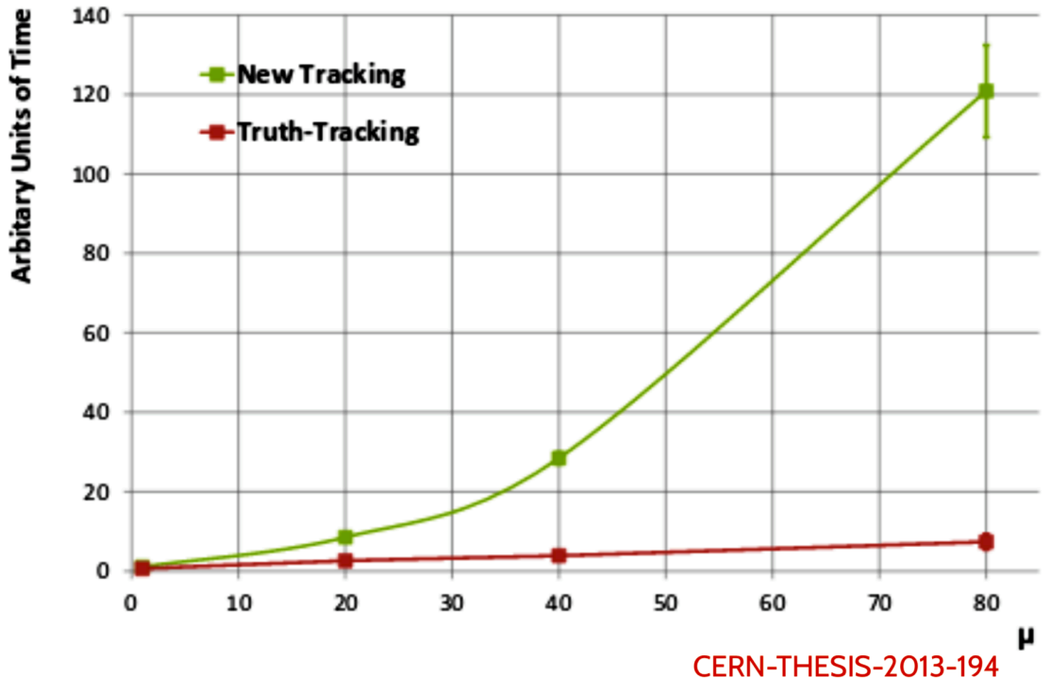
Exploit MC truth information

- Most of the time in reconstruction is spent in ID and increases with pile-up:
  - pattern recognition
  - track seeding
  - ambiguity treatment
- Use truth information to construct tracks
- Use smearing for efficiency and hit content





CERN-THESIS-2013-194



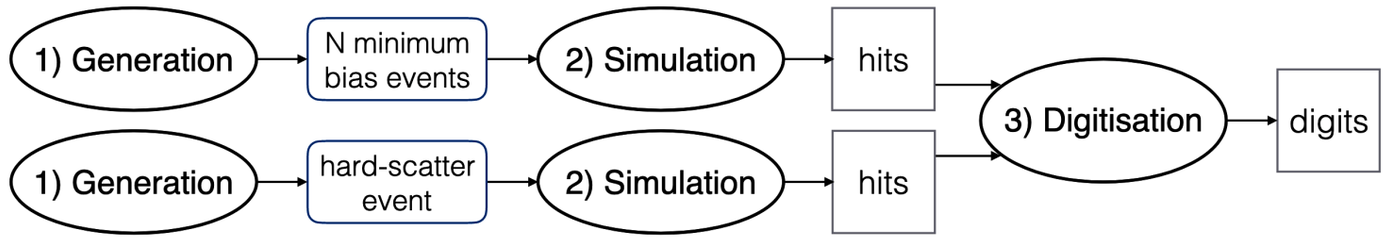
CERN-THESIS-2013-194

Pileup Overlay

CPU time directly proportional to pile-up

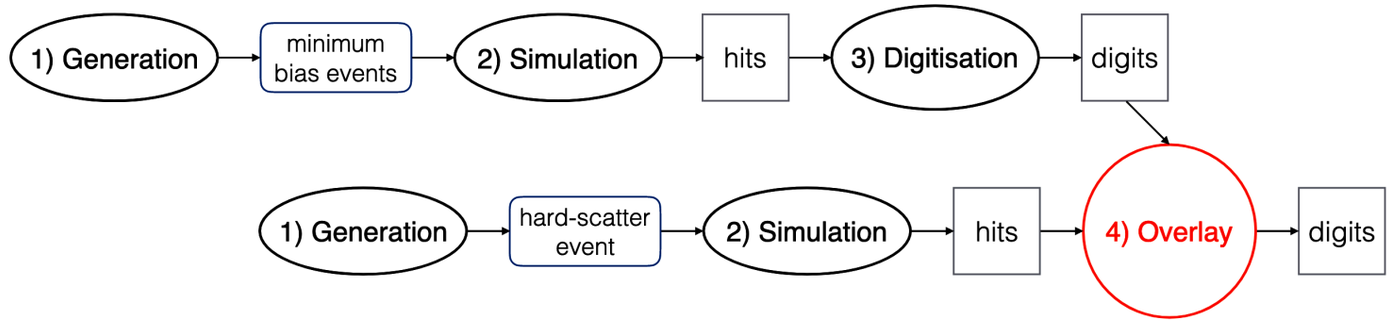
Current default:

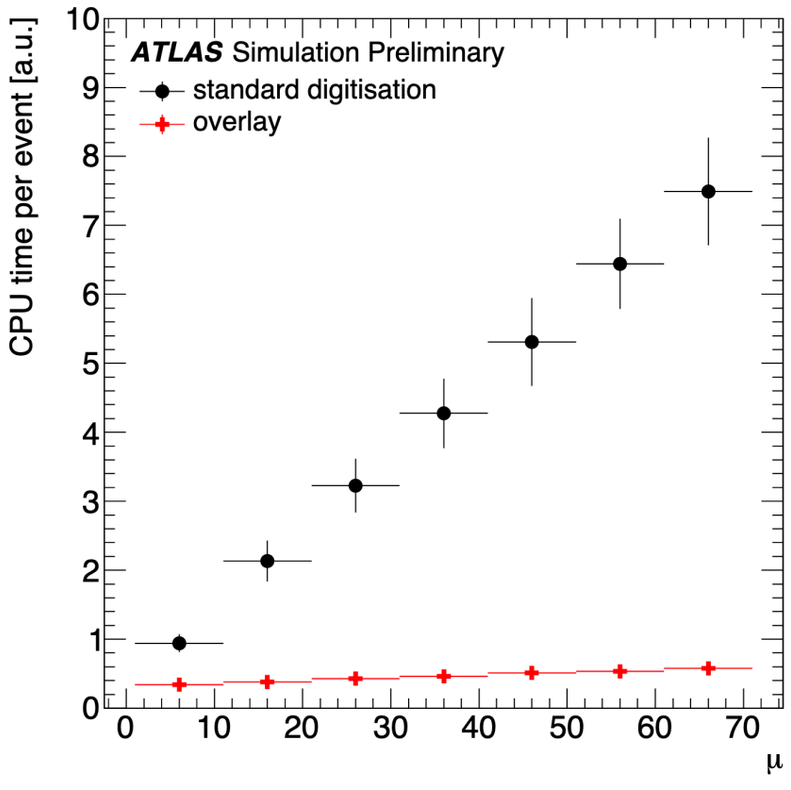
- Default approach require more than 2000 minimum bias events to produce a single MC event
- Large I/O and CPU consumption



Premix and Overlay:

- Simulate and digitize only pile-up events per campaign (using luminosity profile)
- Overly pile-up with hard scatter at digitization





CPU and I/O requirements has almost no pile-up dependance

Different dataset can have the identical set of merged pile-up events

Effect of two sub-threshold signals from different events can cause a signal above threshold to be lost

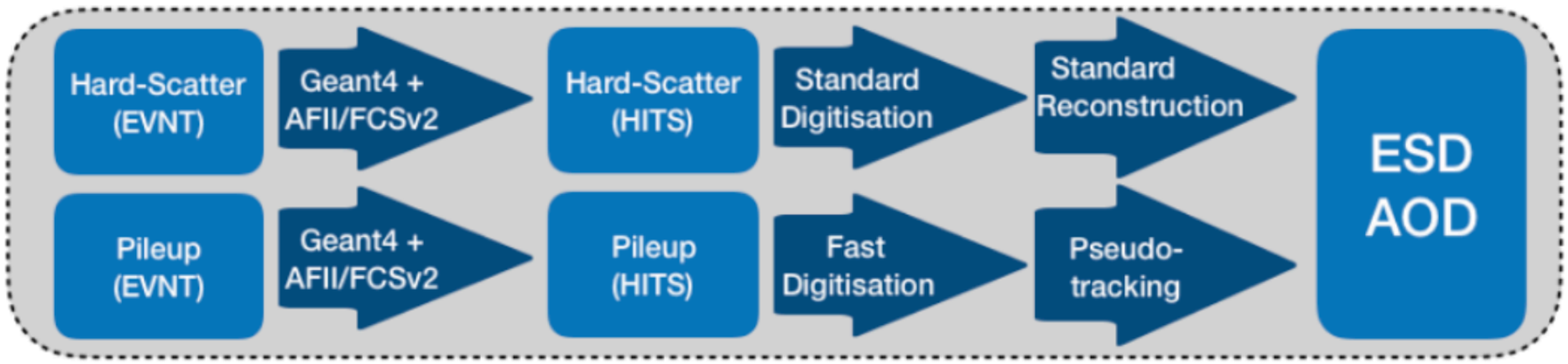
Summary

Fast Simulation is the future in ATLAS

FastCaloSimV2 will be used extensively in Run 3 and beyond

Fast reconstruction would follow

With ISF, there will be several option of running a combination of full and fast MC production chain

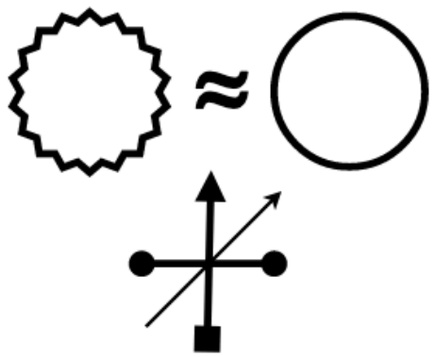


BACKUP

VAE Modeling

	VAE PubNote 2018	VAE CHEP 2019	Motivation
Total number of cells (EMCAL)	266	276	Defining an impact cell per layer (aligned to be the centre cell) resulting in centred showers per layer
Number of input nodes	266 + 1 (Energy condition)	276+5 +1 (Energy condition) (details next slide)	276 for cell energy ratios 5 fractions : Etot/Etruth, + 4 (Elayer/Etot))
Training strategy	Cell energies	Cell energy ratios (CellEnergy/EventEnergyLayer)	-Simplify learning process for the network : learn the energy per cell with respect to the total energy (per layer & event). -Renormalize to energies using the fraction of energy per layer and total energy/Etruth
Weighting strategy	-	Weights = 1/std	-Quantifying the width distributions : characterise the spread (uncertainty) of the data -A weight for each node (of the input and output layers) incorporated in the computation of the loss function
Objective function	Reconstruction + KL + total Energy + Energy fraction	Reconstruction + KL	-Physics knowledge injected in the cell ratios + 5 fractions -Limitations of the default linear combination of loss functions (Multiobjective optimisation)
Latent space dimension	10	5	-The network is able to learn a shower representation in 5D. -Less dimensions in the generation (sample from 5D Gaussian in the latent space)

How to speed up simulation



approximate geometry

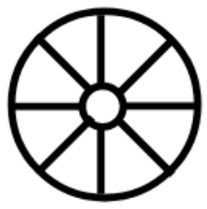
optimise transport and navigation

$\pi \approx 3$

approximate models

parameterisations

take shortcuts



use new technologies