Deep Learning for fast simulation

IXPUG workshop – ISC 2018

F. Carminati, V. Codreanu, G. Khattak, H. Pabst, D. Podareanu,V. Saletore, **S. Vallecorsa**

Outline

Introduction: Deep Learning for fast simulation

Generative Adversarial Networks

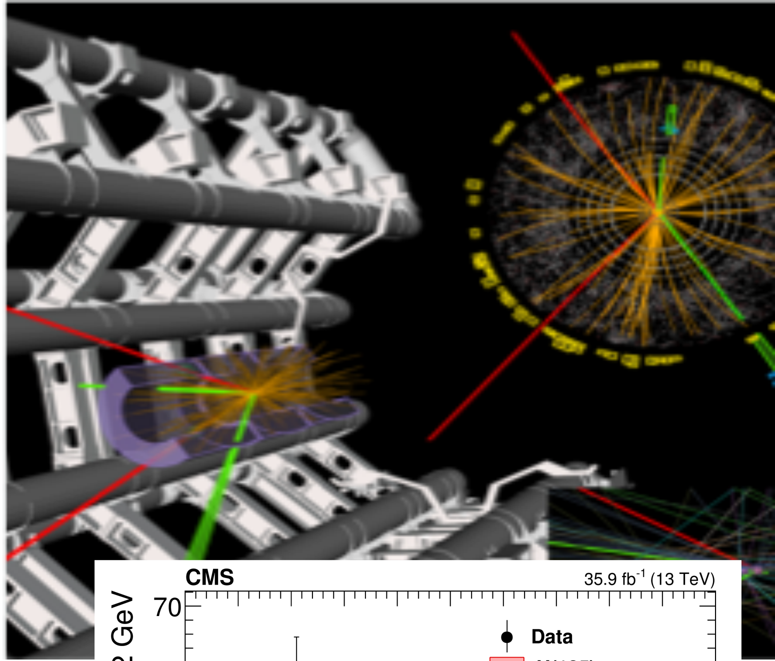
- Model architecture
- The training sample
- Physics Performance

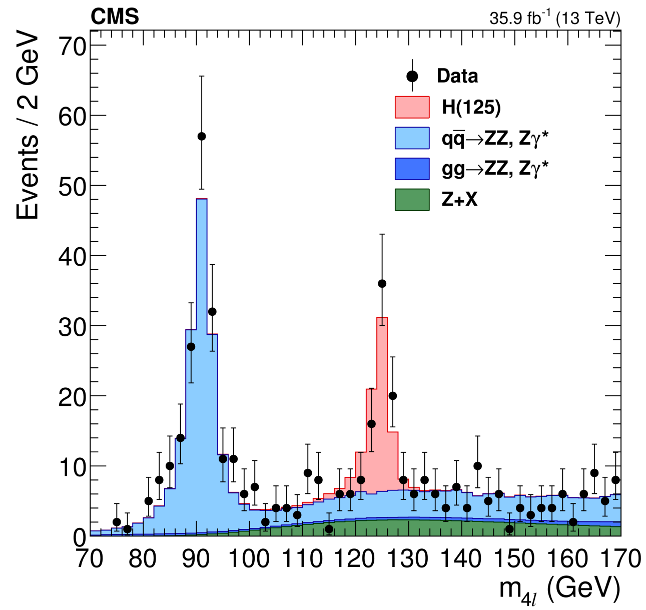
Computing Performance

Outlook and plans

Monte Carlo Simulation: Why

Detailed simulation of subatomic particles is essential for data analysis, detector design





Understand how detector design affect measurements and physics

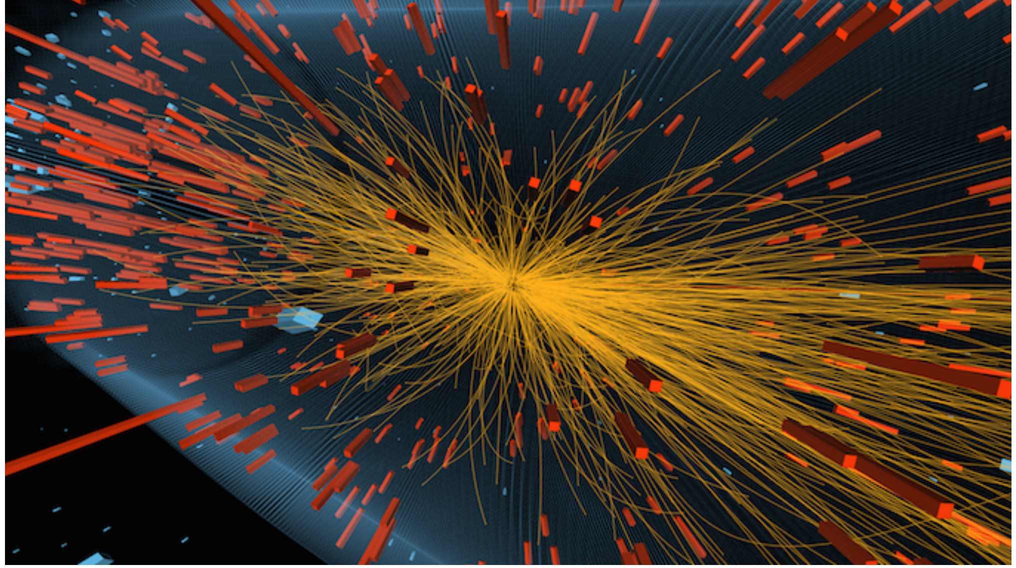
Correct for inefficiencies, inaccuracies, unknowns.

Theory models to compare data against.

A good simulation demonstrates that we understand the detectors and the physics we are studying

CERN openlab

The problem



Complex physics and geometry modeling

Heavy computation requirements

>50% of WLCG power for simulations

Current code cannot cope (HL-LHC in 2025)

Currently available solutions **detector dependent**

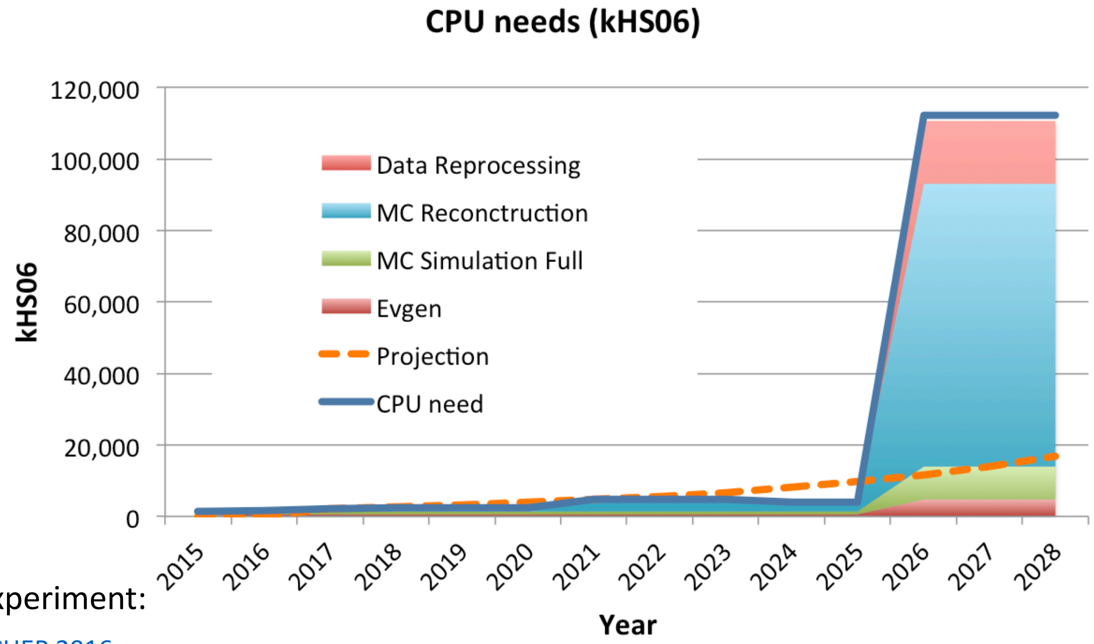
Focus on EM Calorimeter

WLCG
Worldwide LHC Computing Grid

200 Computing centers in 20 countries: > 600k cores

@CERN (20% WLCG): 65k cores; 30PB disk + >35PB tape storage

ATLAS experiment:



Campana, CHEP 2016

Classical Monte Carlo simulation

simplified from
[A. Gheata]

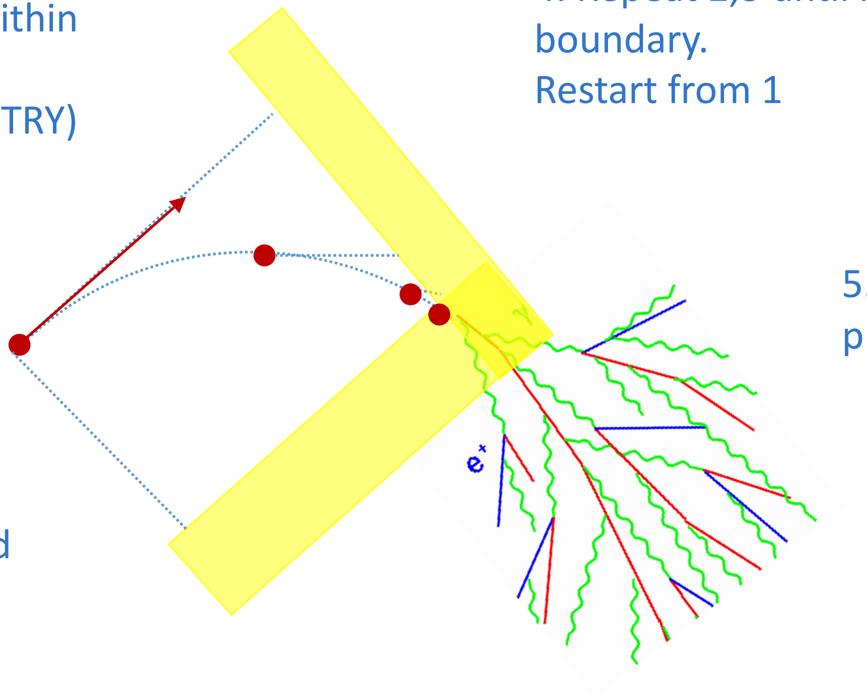
1. Calculate step particle could travel before doing a PHYSICS interaction

2a. Chek if step is within volume boundaries(GEOMETRY)

3. Propagate with selected step

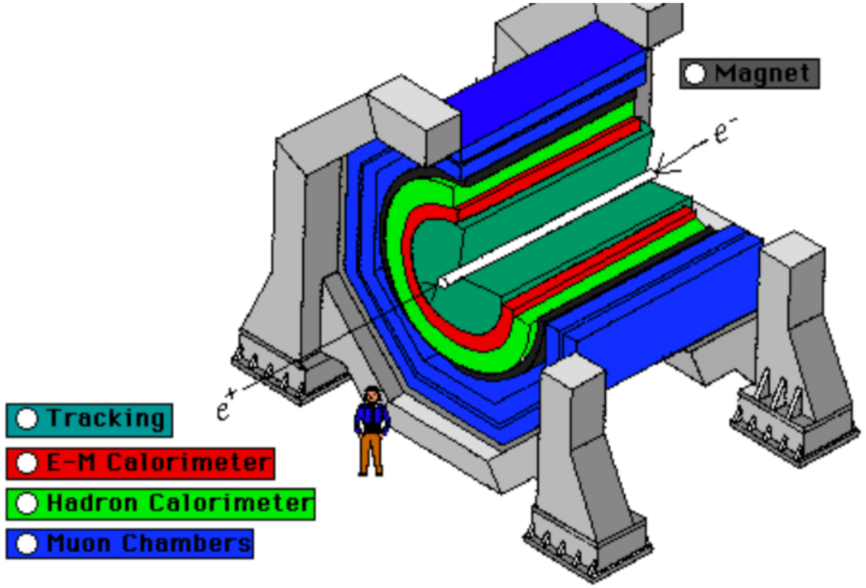
4. Repeat 2,3 until reaching volume boundary.
Restart from 1

5. PHYSICS process



Repeat **stages** 1 to 5 :

- For every particle trajectory step
- For every primary particle
- For every secondary particle



Deep Learning for fast simulation

Improved, efficient and accurate fast simulation

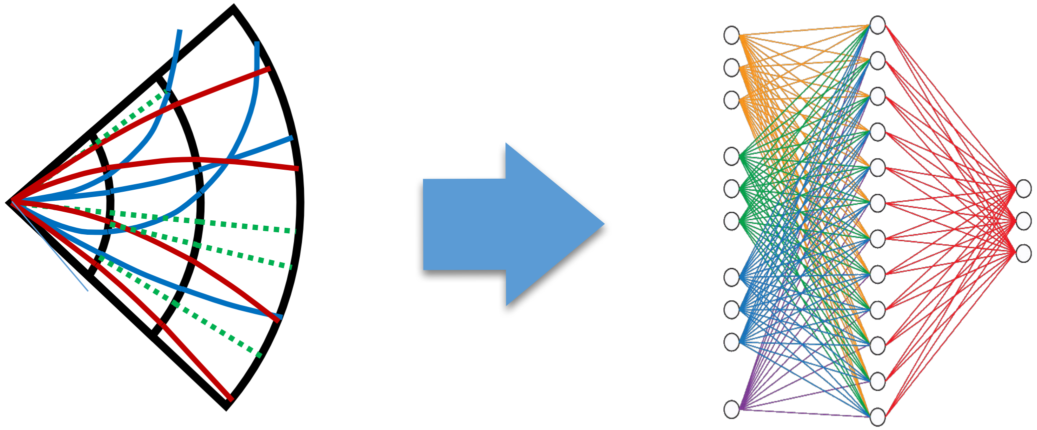
Generic approach

Can encapsulate expensive computations

Inference step is faster than algorithmic approach

Already parallelized and optimized for GPUs/HPCs.

Industry building highly optimized software, hardware, and cloud services.



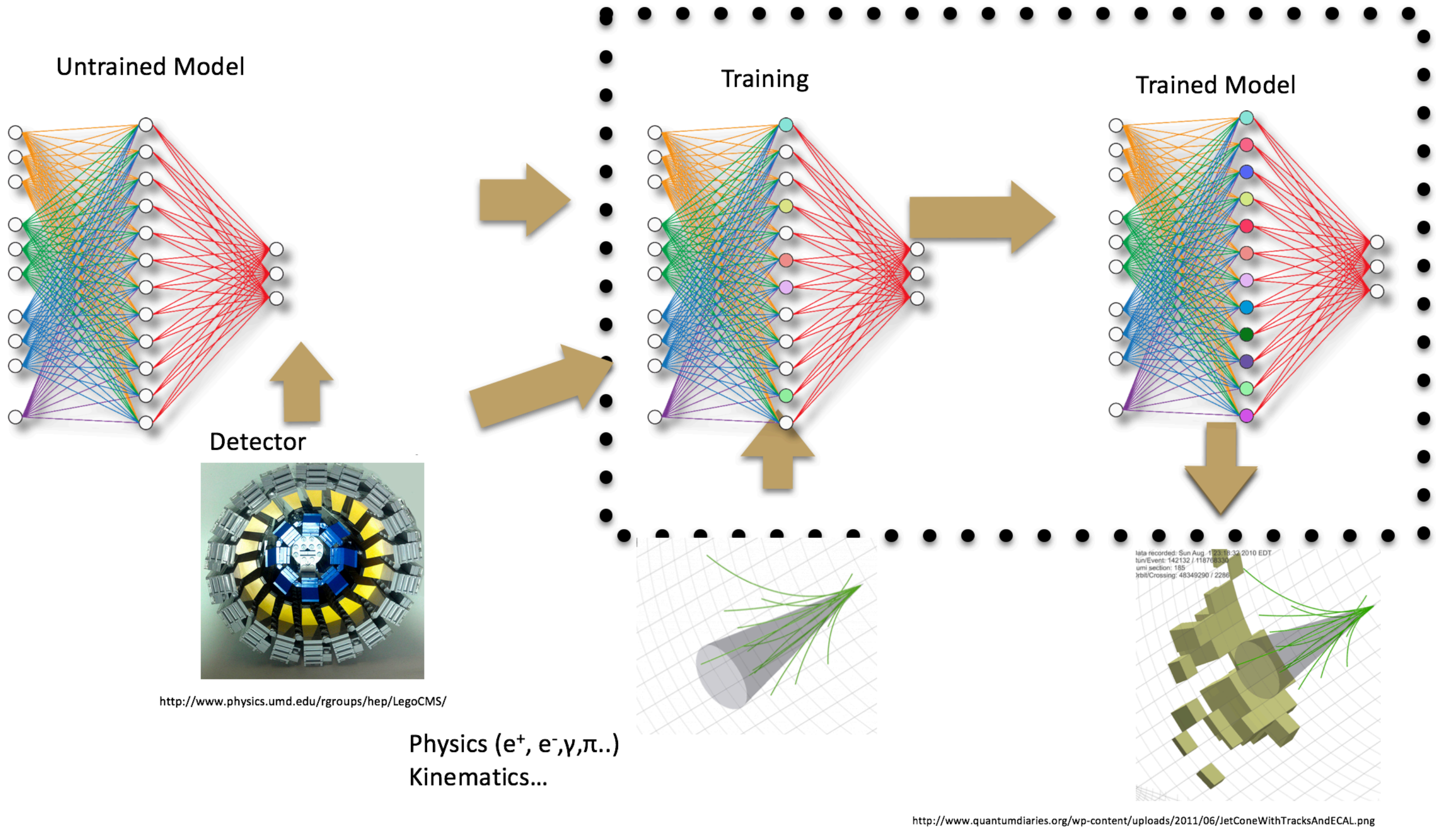
Can we keep accuracy while doing things faster?

Requirements

- Precise simulation results:
  - Detailed validation process
- A fast inference step
- Generic customizable tool
  - Easy-to-use and easily extensible framework
- Large hyper parameters scans and meta-optimisation:
  - Training time under control
  - Scalability
  - Possibility to work across platforms

A DL engine for fast simulation

- Start with time consuming detectors
    - Reproduce particle showers in calorimeters
- Train on detailed simulation
    - Test training on real data
- Test different models
    - Generative Adversarial Networks
- Embed training-inference cycle in simulation



A plan in two steps

Can image-processing approaches be useful?

Can we preserve accuracy while increasing speed?

Can we sustain the increase in detector complexity (future highly-granular calorimeters)?

- A first proof of concept
- Understand performance and validate accuracy

How generic is this approach?

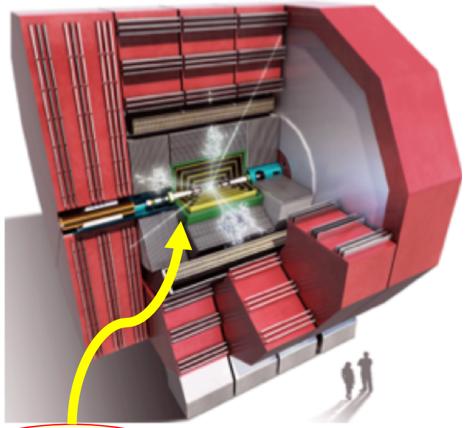
Can we “adjust” architecture to fit a large class of detectors?

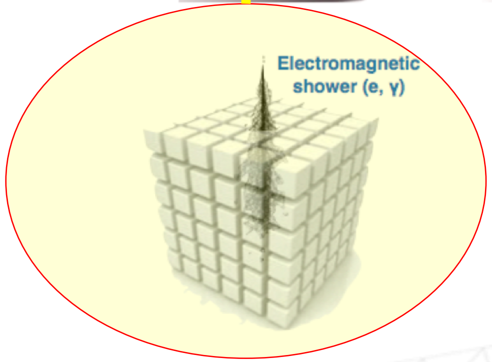
What resources are needed?

- Prove generalisation is possible
- Understand and optimise computing resources
- Reduce training time
- HPC friendly

CLIC Calorimeter

Array of absorber material and silicon sensors



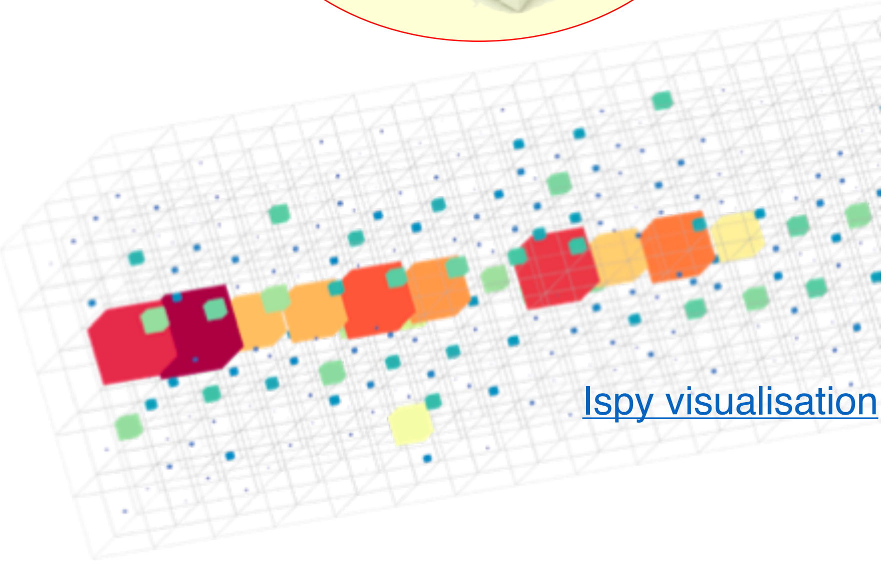


CLIC is a CERN project for a linear accelerator of electrons and positrons to TeV energies

Associated electromagnetic calorimeter detector design(*)

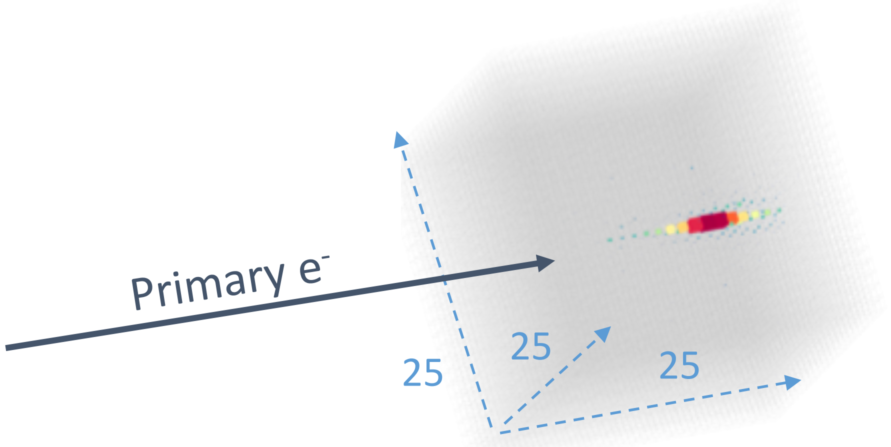
Highly segmented (pixelized)

Segmentation is critical for particle identification and energy calibration.



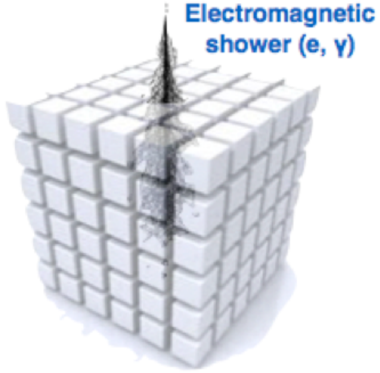
Ispy visualisation

Detector output is essentially a 3D image



(*) http://cds.cern.ch/record/2254048#

CLIC calorimeter data

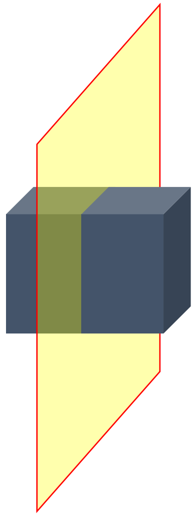


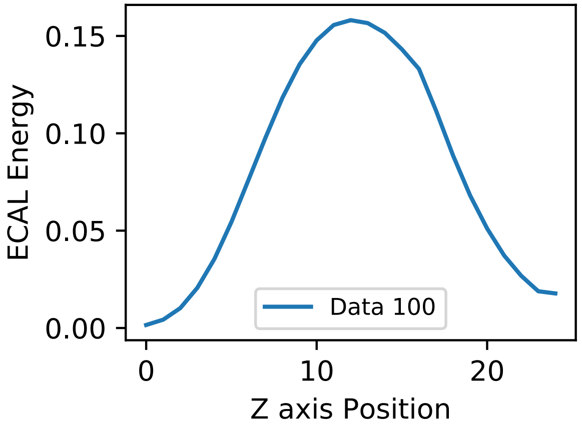
Sparse.

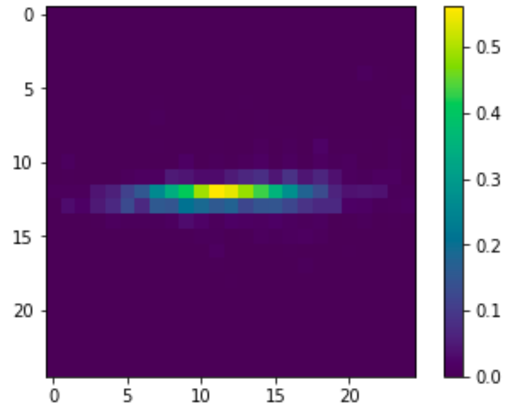
Non-linear location-dependency

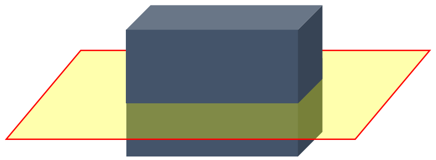
Dataset: 200.000 electrons depositing energy in the calorimeter

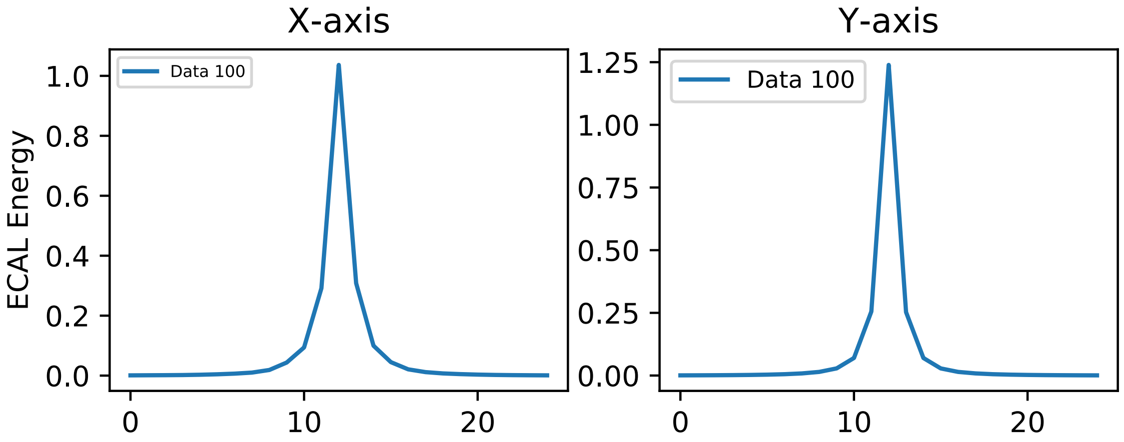
Energy range: 10-510GeV

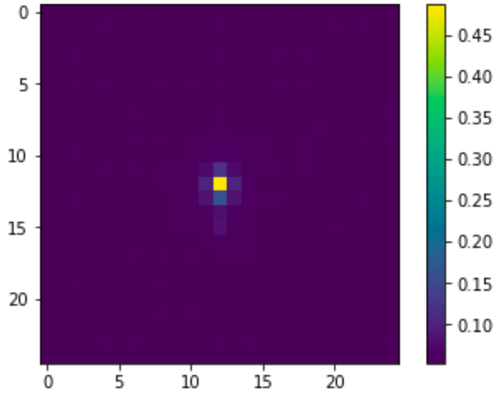












Generative adversarial networks

Simultaneously train two networks that compete and cooperate with each other:

Generator G generates data from random noise

Discriminator D learns how to distinguish real data from generated data

https://arxiv.org/pdf/1701.00160v1.pdf

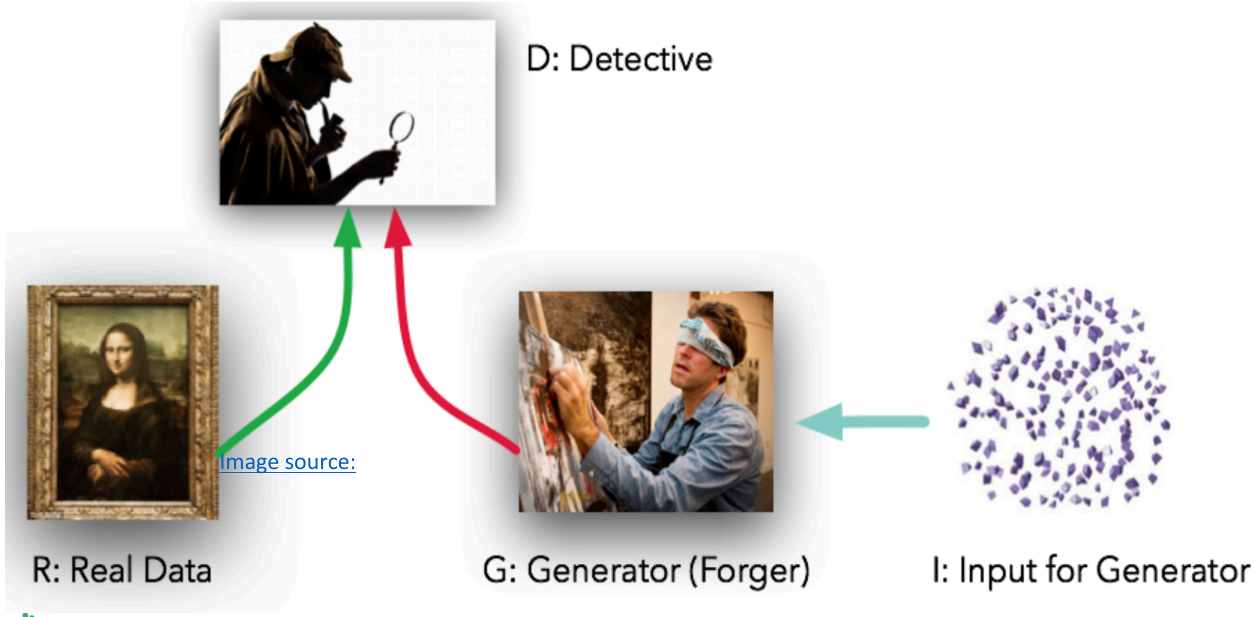


Image source:

The counterfeiter/detective case

Counterfeiter shows the Monalisa

Detective says it is fake and gives feedback

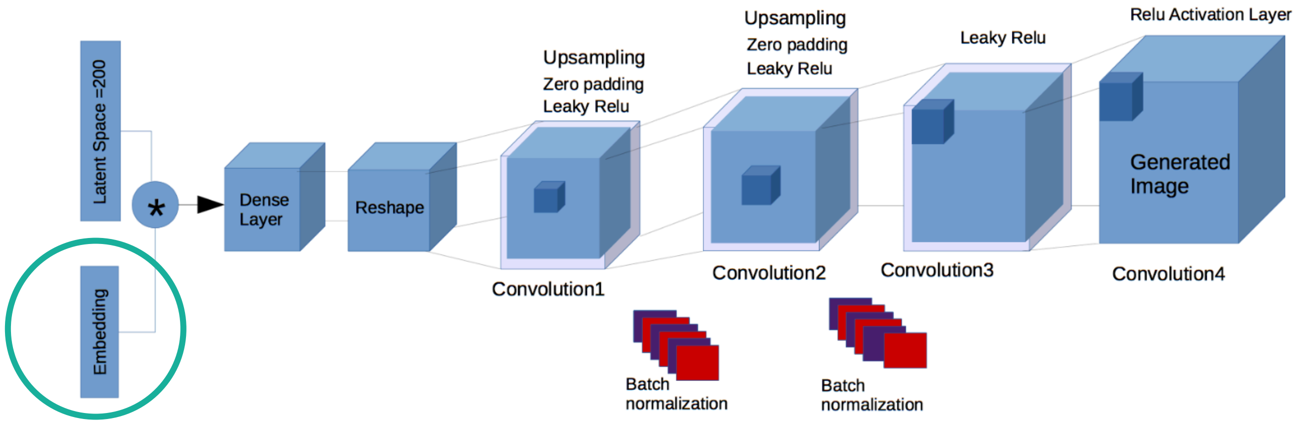
Counterfeiter makes new Monalisa based on feedback

Iterate until detective is fooled

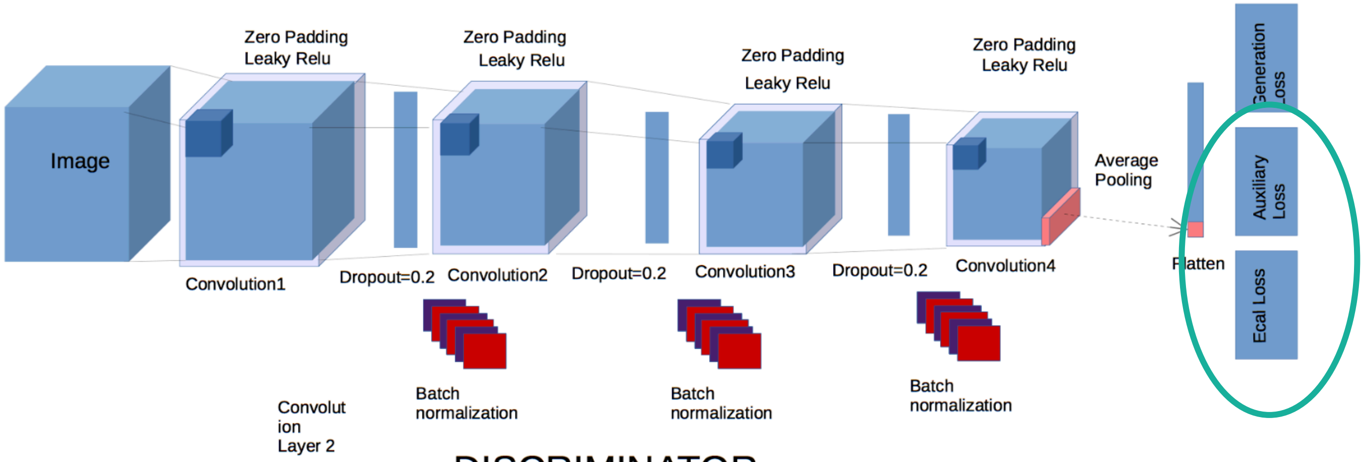
Network architectures

3D conditional GAN with two auxiliary regression tasks

Based on 3D convolution/deconvolutions to describe whole volume



GENERATOR



DISCRIMINATOR

Conditioning and auxiliary tasks

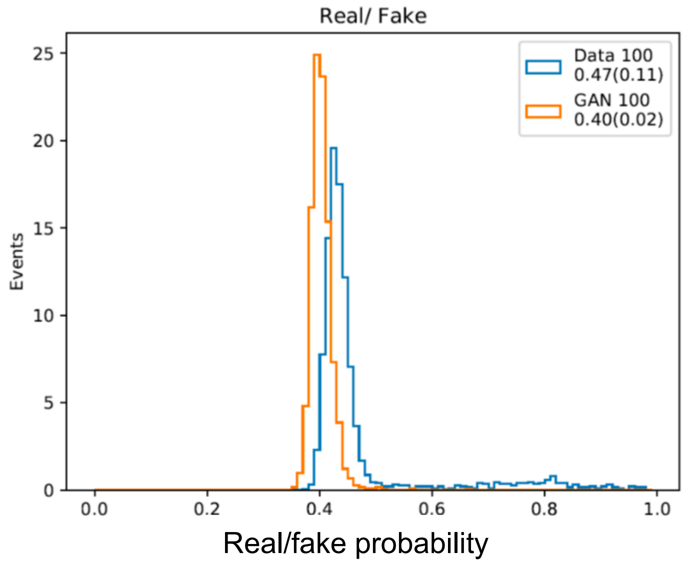
Condition training on several input variables (particle type, energy, incidence angle)

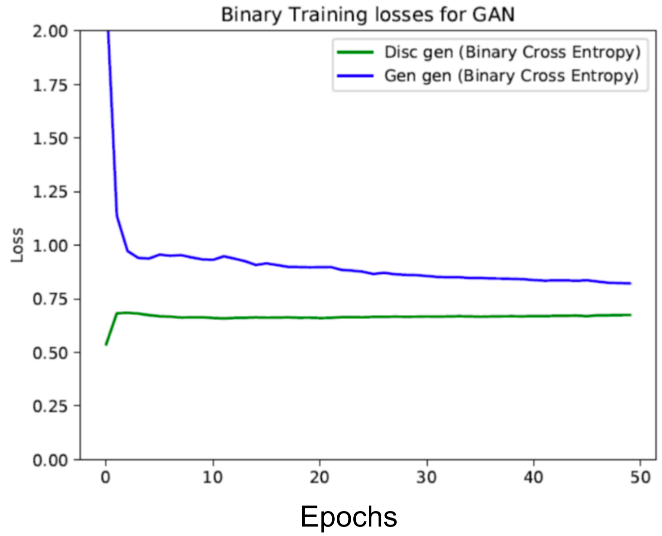
Auxiliary regression tasks assigned to the discriminator: primary particle energy, deposited energy, incidence angle

Loss is linear combination of 3 terms:

- Combined cross entropy (real/fake)
- Mean absolute percentage error for regression tasks

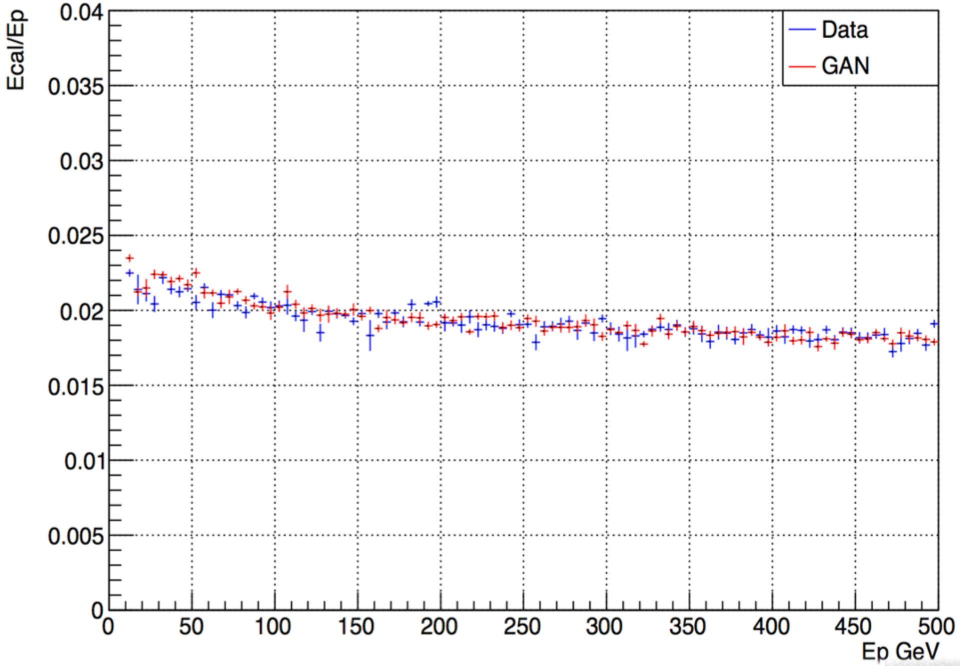
Easily generalisable to multi-class approach (or multi-discriminator approach): angle..

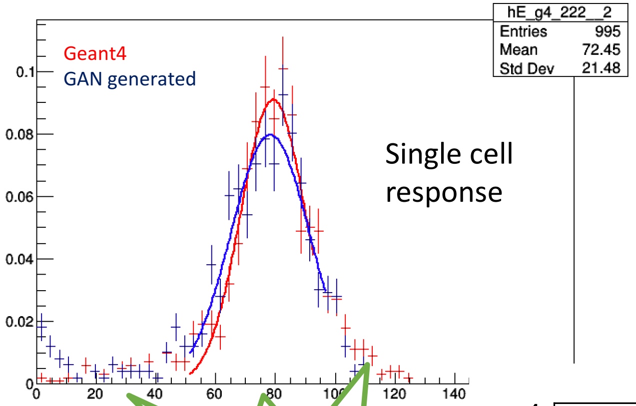


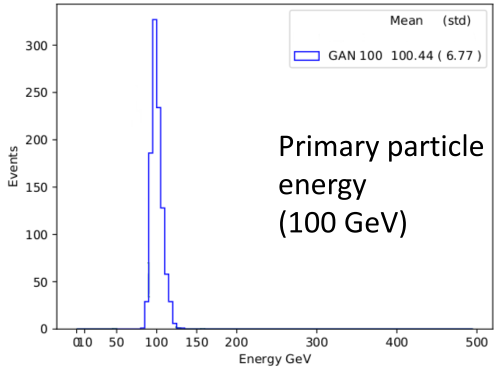


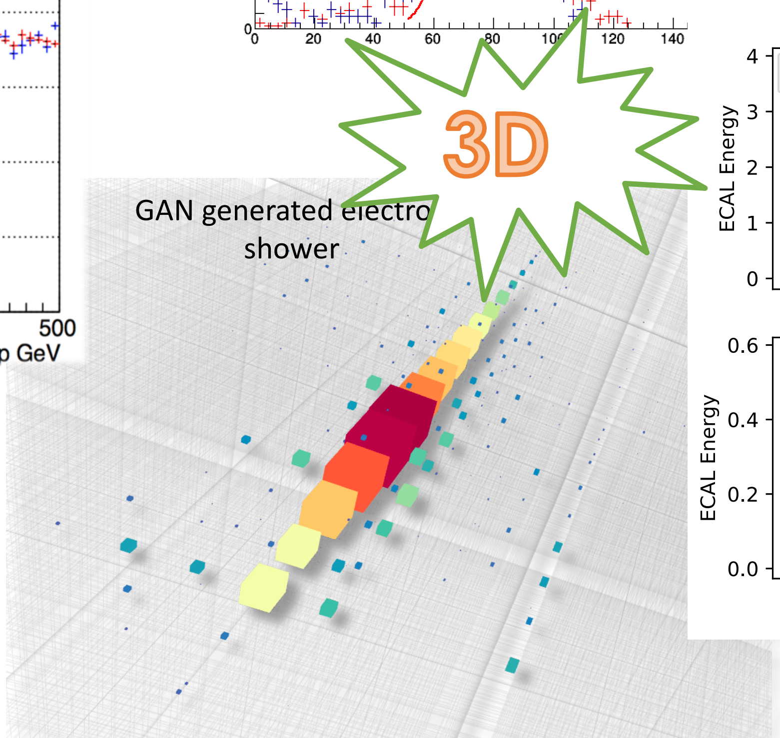
RESULTS validation

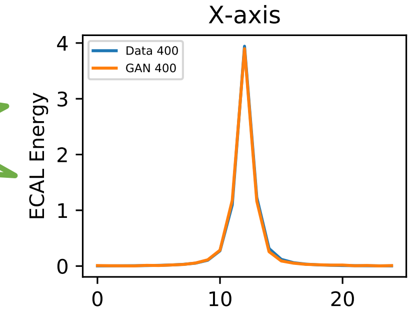
Comparison to Monte Carlo data

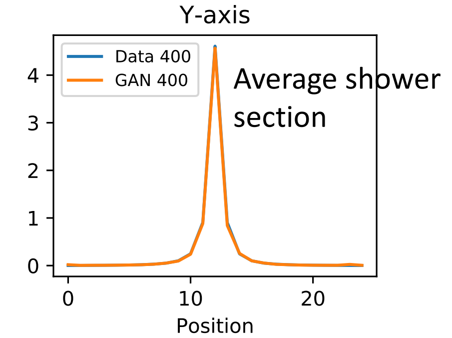


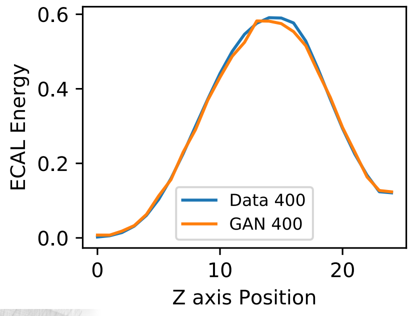


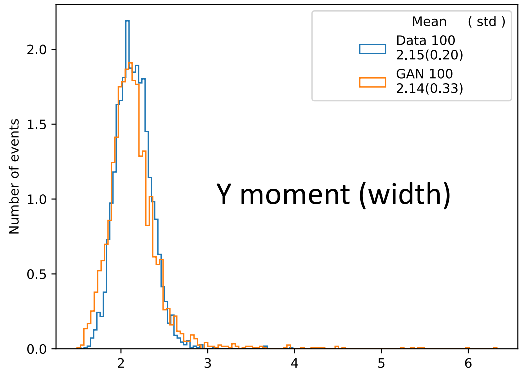












CERN openlab

Generation speedup

Using a trained model is very fast

Inference:

Classical Monte Carlo requires 17 s/shower

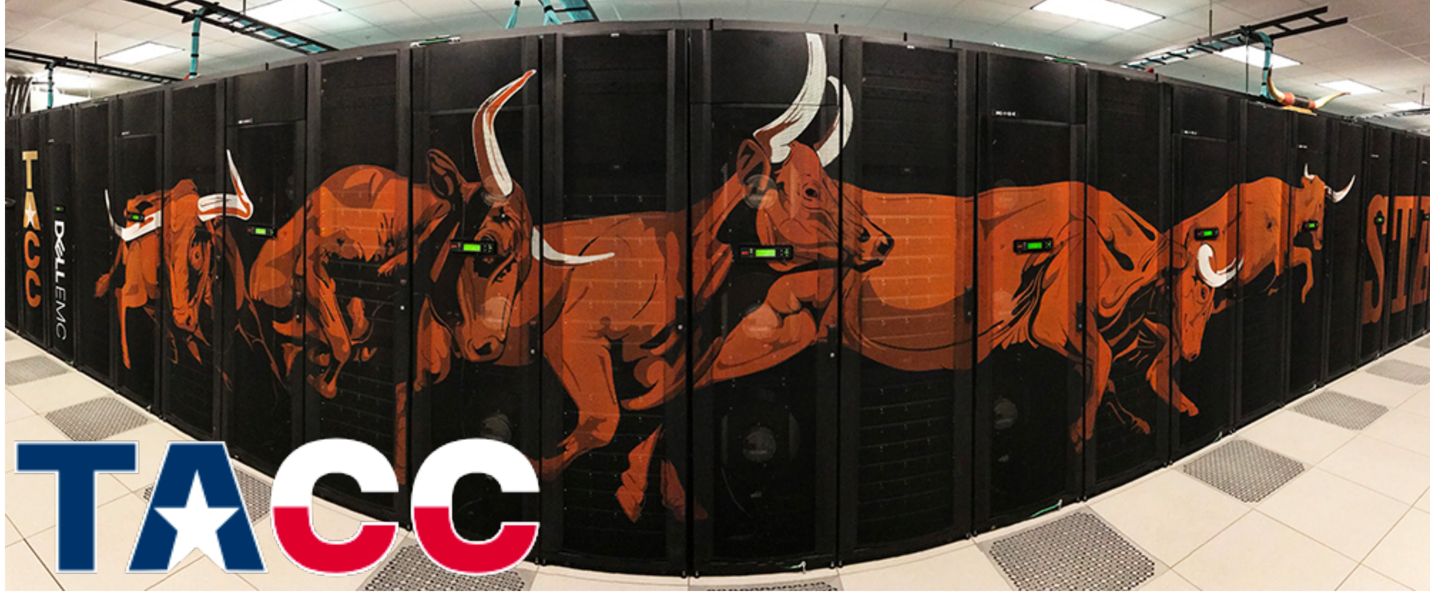
3DGAN takes 7 ms/shower

➔ speedup factor > 2500!!

Time to create an electron shower		
Method	Machine	Time/Shower (msec)
Full Simulation (geant4)	Intel Xeon Platinum 8180	17000
3D GAN (batch size 128)	**Intel Xeon Platinum 8180**	**7**

CERN openlab

Distributed training



Use keras 2.13 /Tensorflow 1.9 (Intel optimised)

- AVX512 –FMA-XLA support
- Intel® MKL-DNN (with 3D convolution support)

Optimised multicore utilisation

- inter_op_paralellism_threads/intra_op_paralellism threads

Horovod 0.13.4

- Synchronous SGD approach
- MPI_AllReduce

Run on TACC Stampede2 cluster:

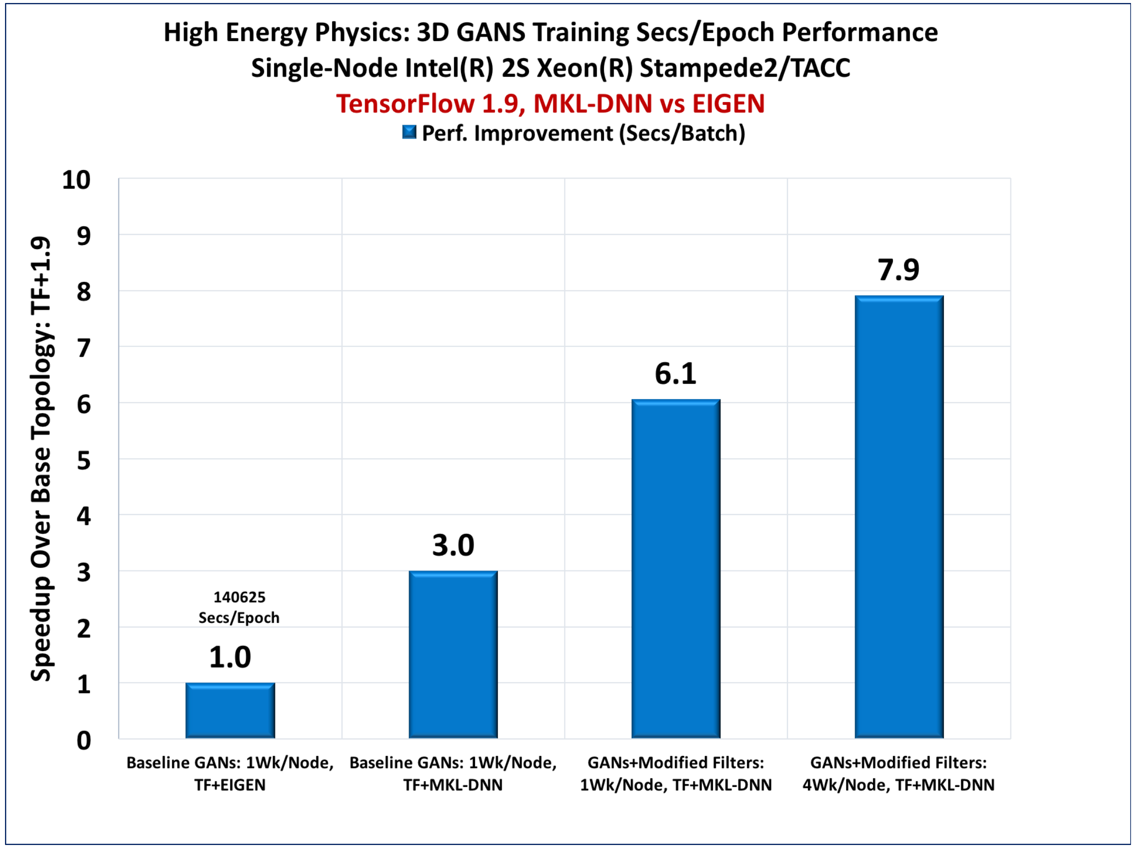
- Dual socket Intel Xeon 8160
- 2x 24 cores per node, 192 GB RAM
- Intel® Omni-Path Architecture

Test several MPI scheduling configurations

- 2,4, 8 processes per nodes.
- Best machine efficiency with 4 processes/node

Training time optimisation

- **1 worker/node TF + Eigen (baseline)**
- 1 worker/node TF + MKL-DNN
- 1 worker/node, TF+ MKL-DNN, optimised number of convolution filters
- 4 workers/node, TF+ MKL-DNN, optimised number of convolution filters

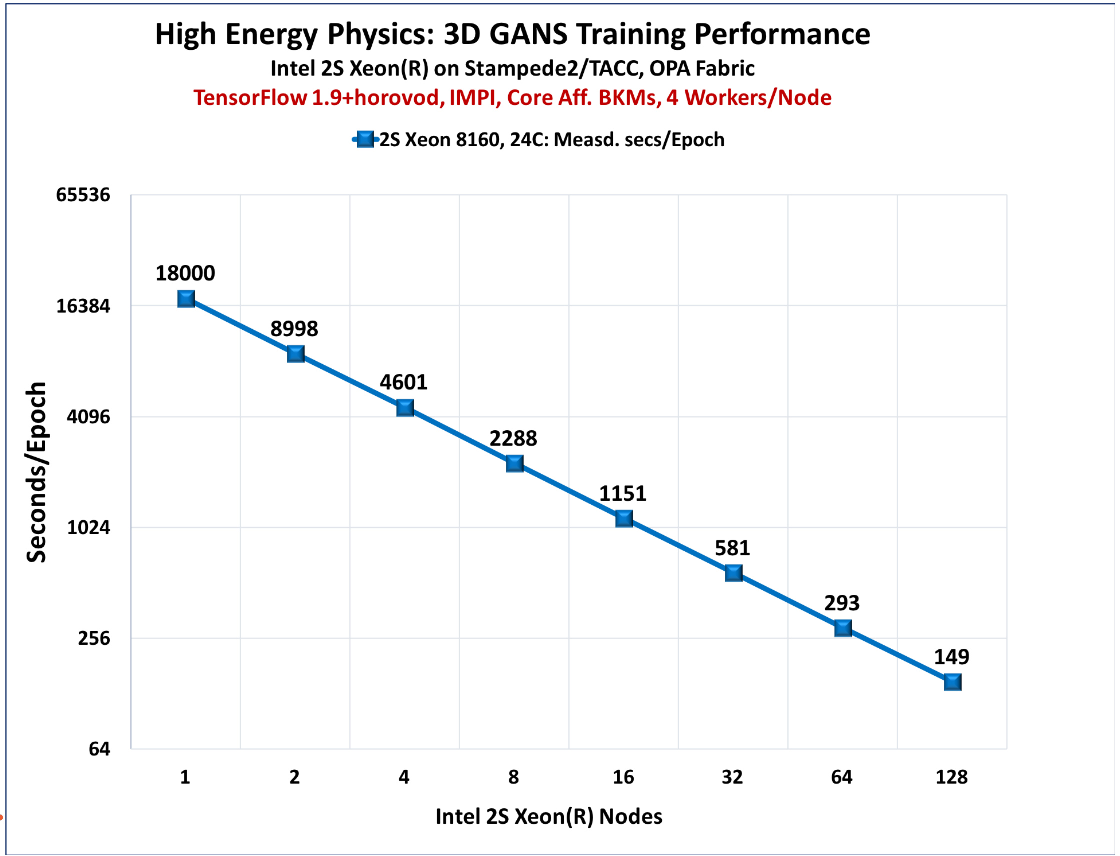


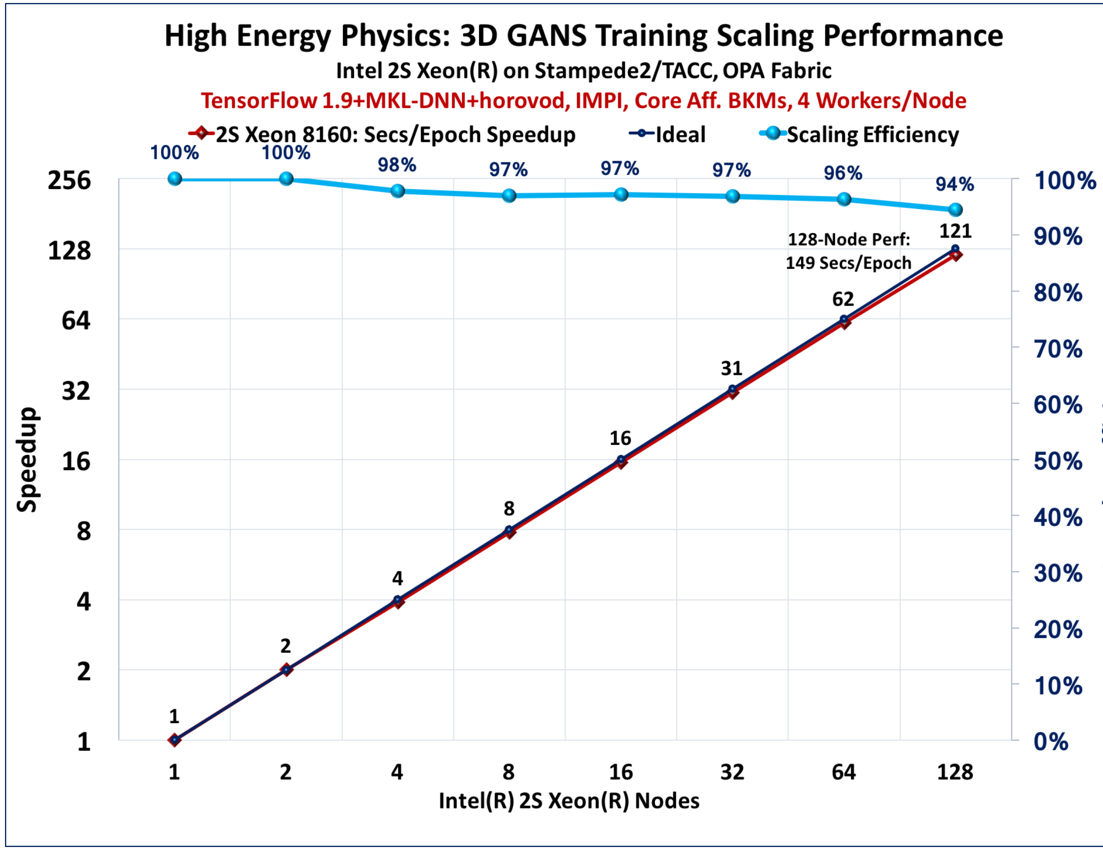
CERN openlab

Scaling results

Distributed training using data parallelism

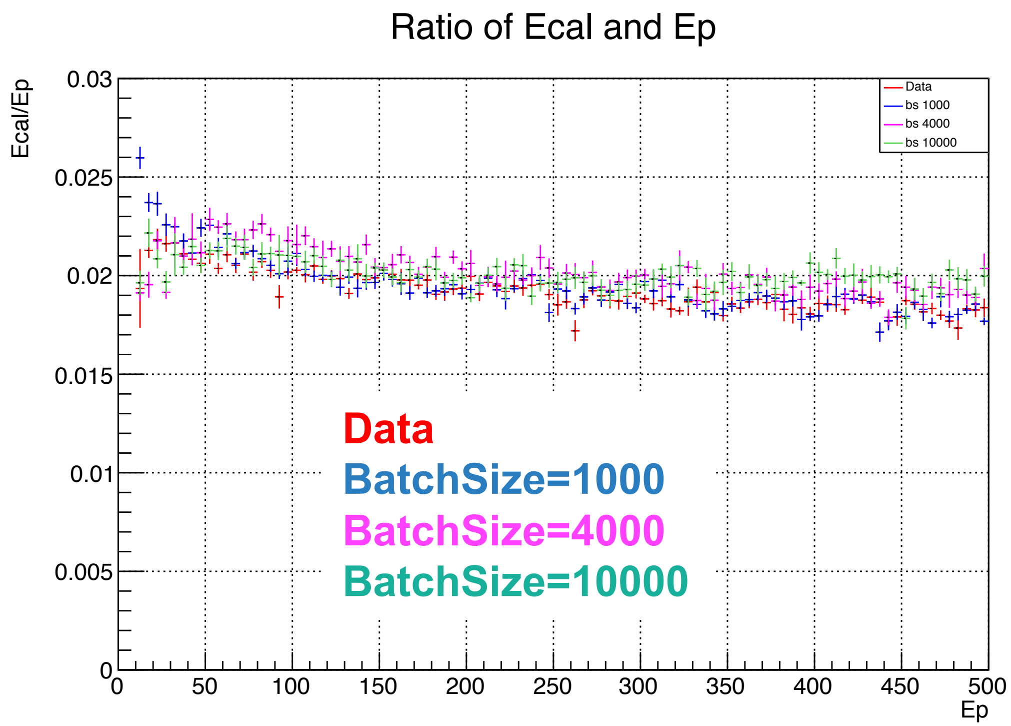
94% scaling efficiency up to 128 nodes





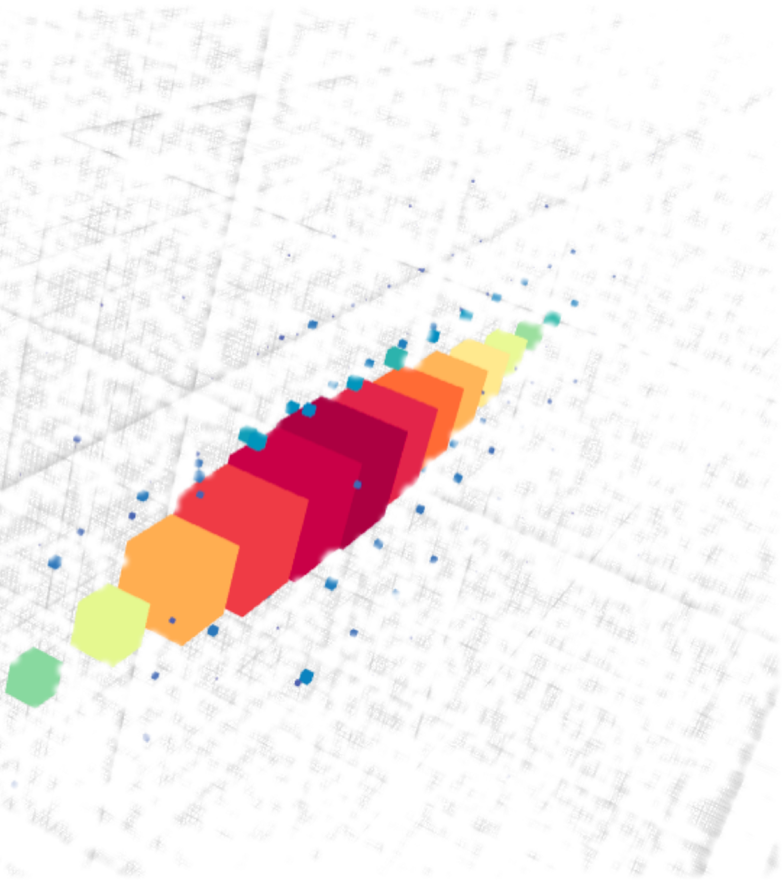
Physics performance at scale

- Some performance degradation
  - Mostly in the low energy regions for large batchsize (4096)
  - Network optimised for the 100-200 GeV central region
  - Applied warmup and scaling of initial learning rate
  - Further investigation ongoing



Conclusion & Plans

First results are very promising from physics perspective



- Distributed training process and optimisation to scale on clusters is critical
  - Allows meta-optimisation and hyperparameter scans in order to generalize to different detectors
  - Parallelizing training process and optimize scaling on clusters
- Initial results are very promising
  - Reduced training time by x8 on single node
  - Linear scaling brings down training time to ~2min on 128 nodes
- Keep working on the understanding / optmisation of physics performance at scale

Questions?

Sofia.Vallecorsa@cern.ch