Performance optimisation of 3DGAN training and inference on Intel architectures

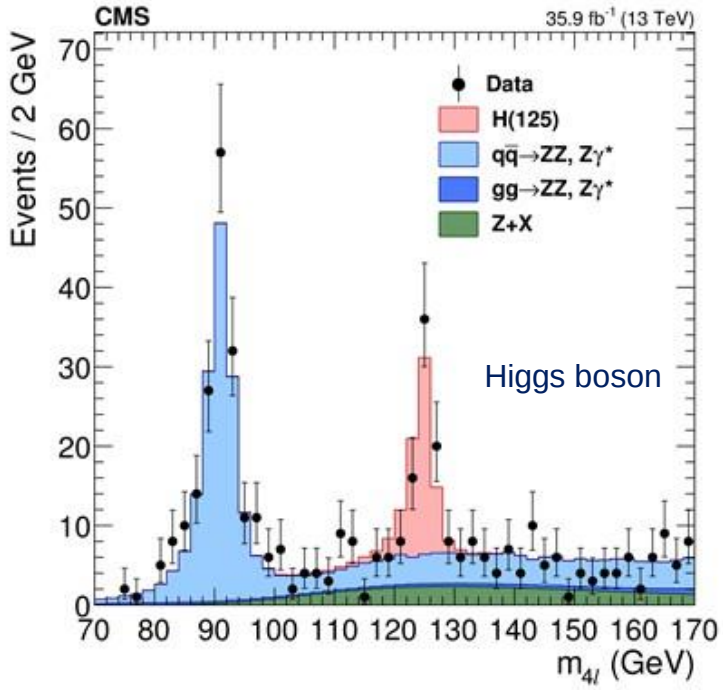
Sofia Vallecorsa

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

Outline

Monte Carlo Simulation

Essential for data analysis & detector design



Understand how detector design affects measurements and physics

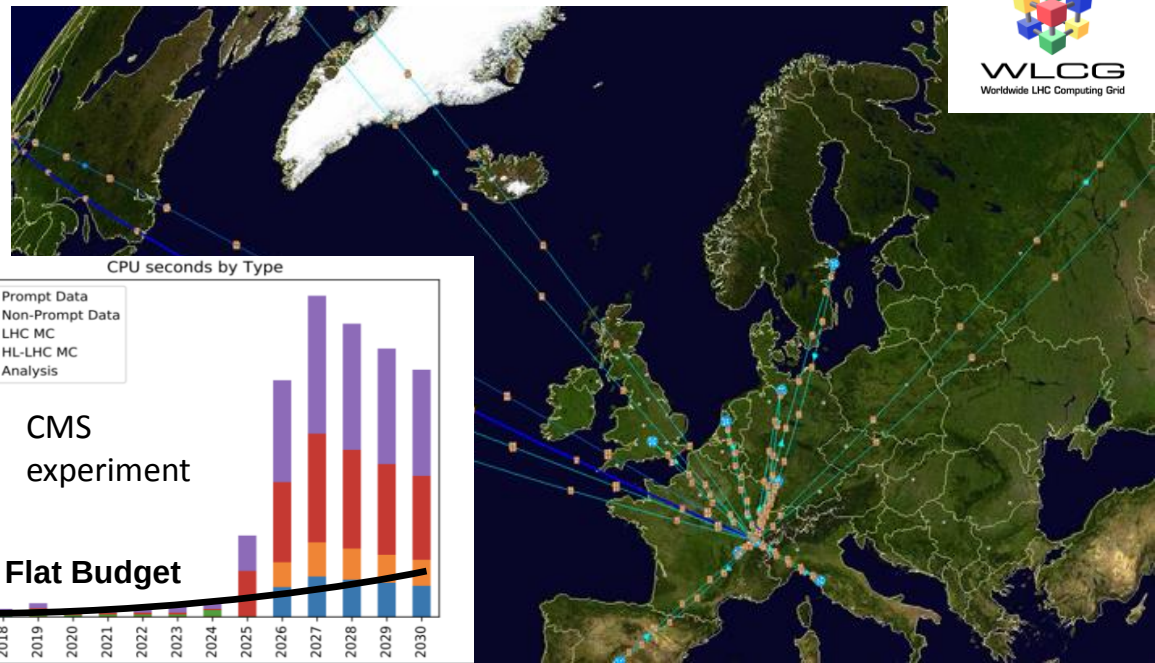
- Correct for inefficiencies, inaccuracies, …
- Compare theory models to data

It requires complex physics and geometry modeling

- >50% of **Worldwide LHC Computing Grid (WLCG)** power today
- HL-LHC brings x100 increase in simulation need

Generative Models a **generic approach**

- Inference is much faster than Monte Carlo



CERN openlab

Deep Generative Models

Extract meaningful representations from data

Internal representations learned by shallow systems are simple (Bengio & LeCun 2007, Bengio 2009)

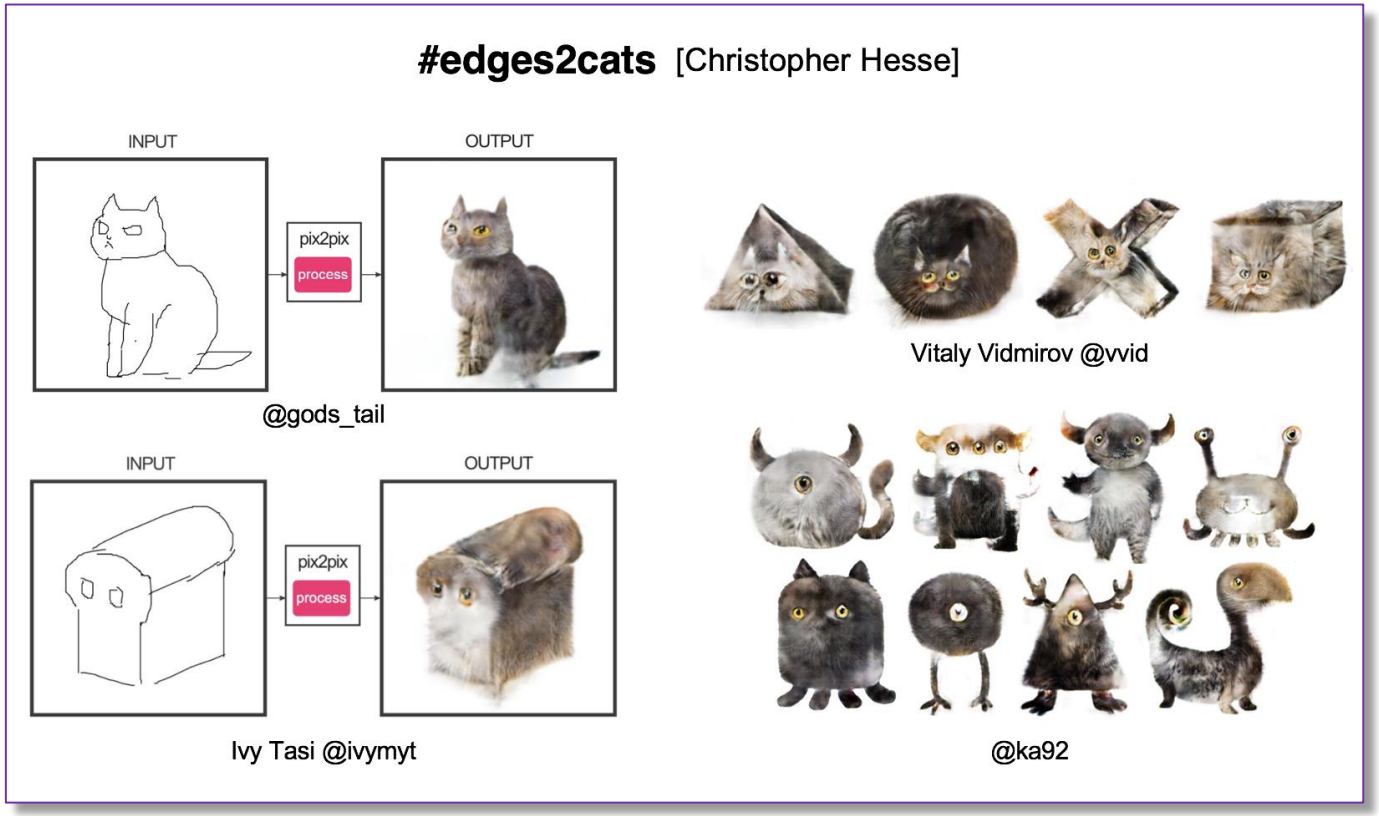
- Incapable of learning complex hidden structures
- Require large amounts of labeled data

→ Deep Generative Models

- → Allow higher levels of abstractions
- → Improve generalisation and transfer

→ Multiple applications

- Find Underlying Factors (**Discovery**)
- Detect Rare events (**Anomaly Detection**)
- Predict future events (**Planning**)
- Find Analogies (**Transfer Learning**)



CERN openlab

arXiv:1406.2661v1

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

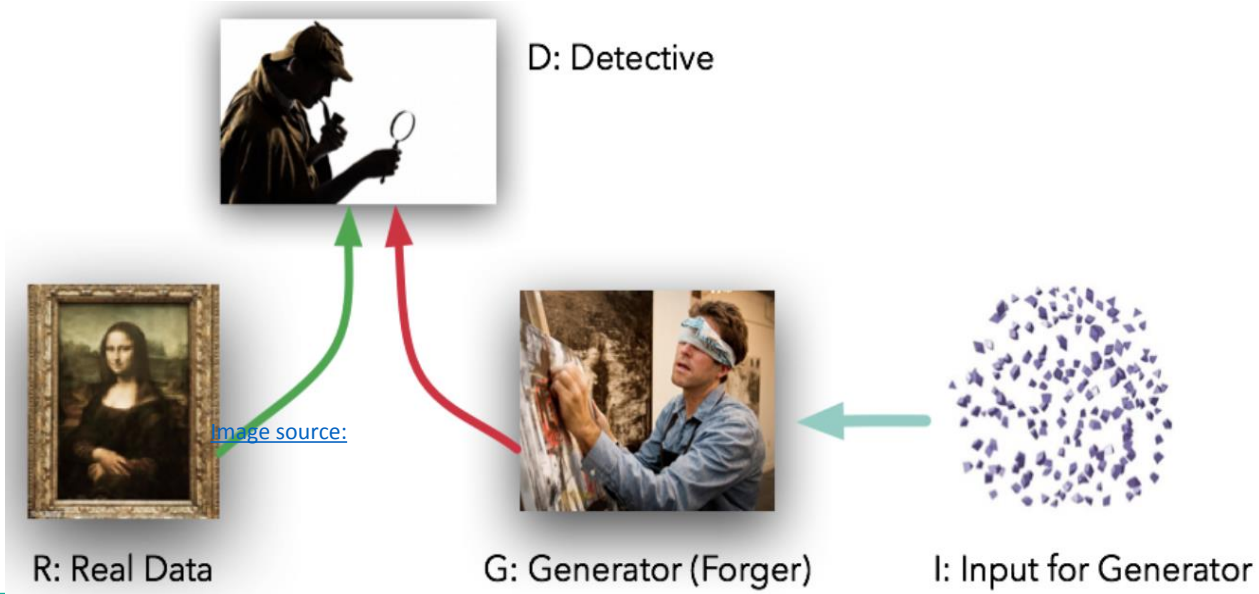


Image source:

The counterfeiter/detective case

Counterfeiter shows its Monalisa to the detective

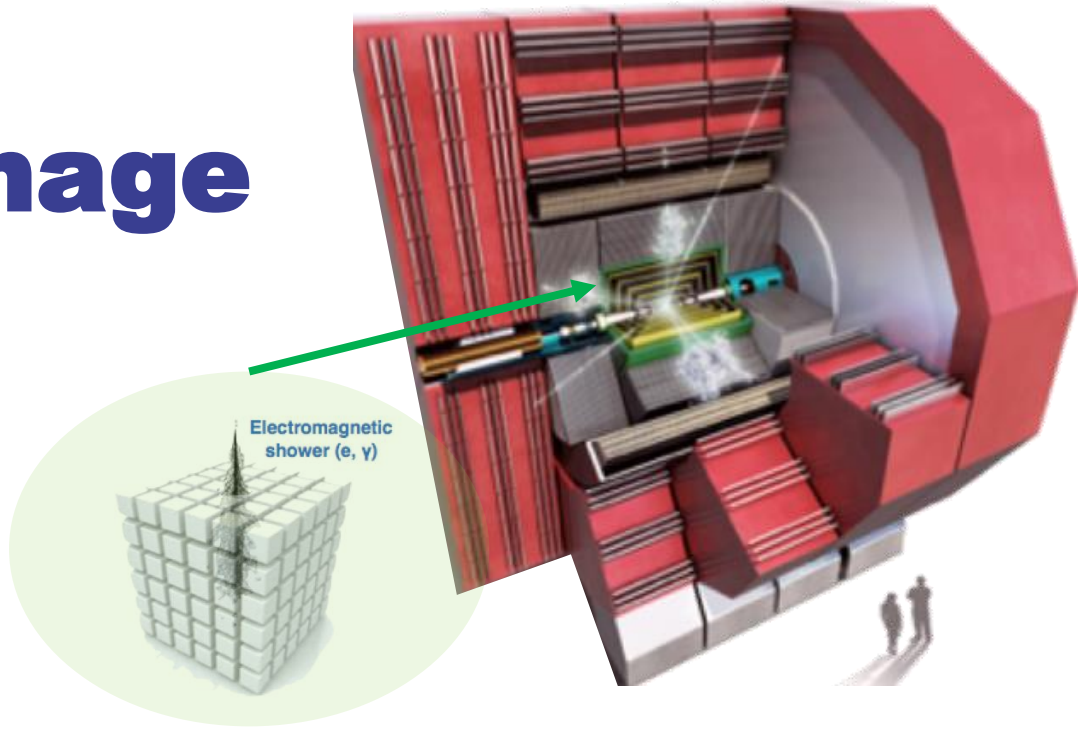
Detective says it is fake

Counterfeiter makes new Monalisa-style based on feedback

Iterate until detective is fooled

Detector output as 3D image

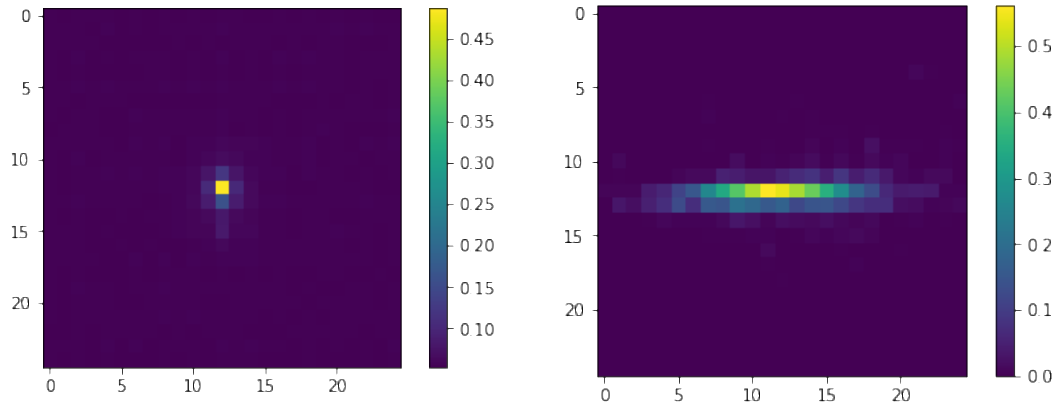
Array of absorber material and silicon sensors

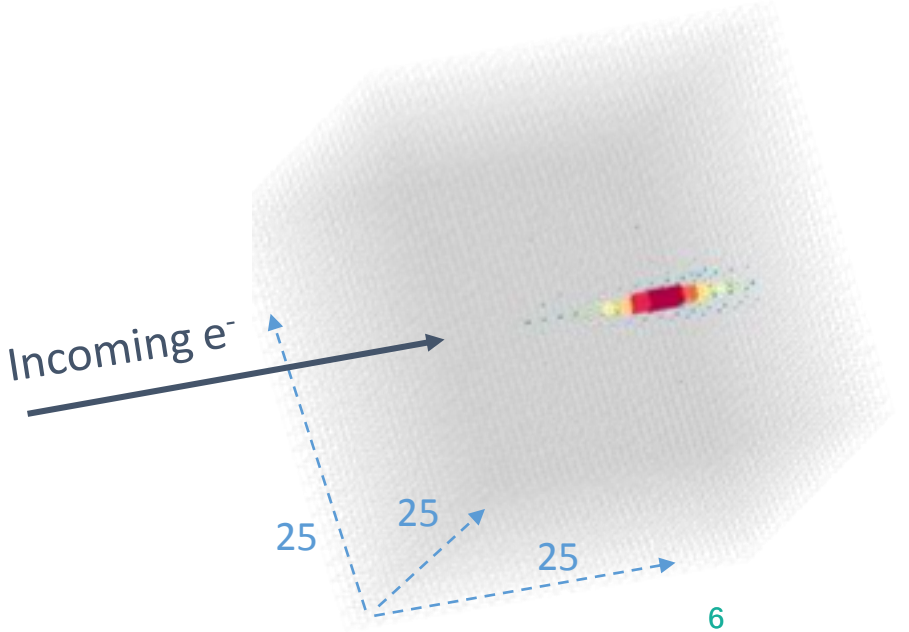


CLIC is a CERN project for a linear accelerator of electrons and positrons to TeV energies

Electromagnetic calorimeter design

- Sparse images
- Highly segmented (pixelized)
- Large dynamic range

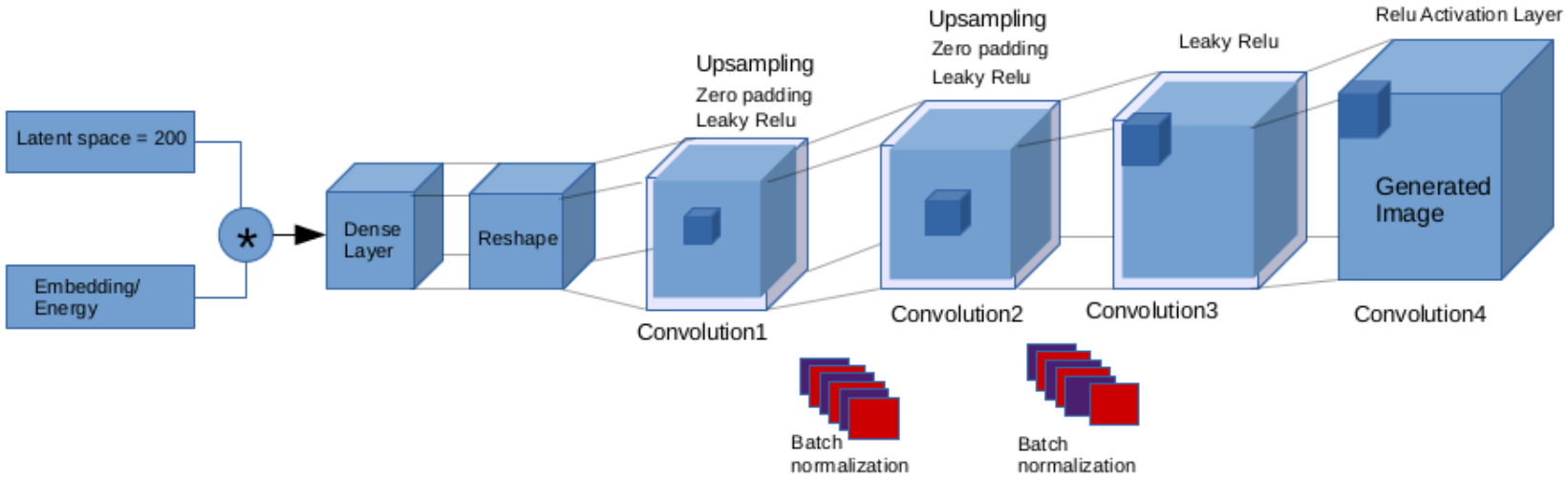




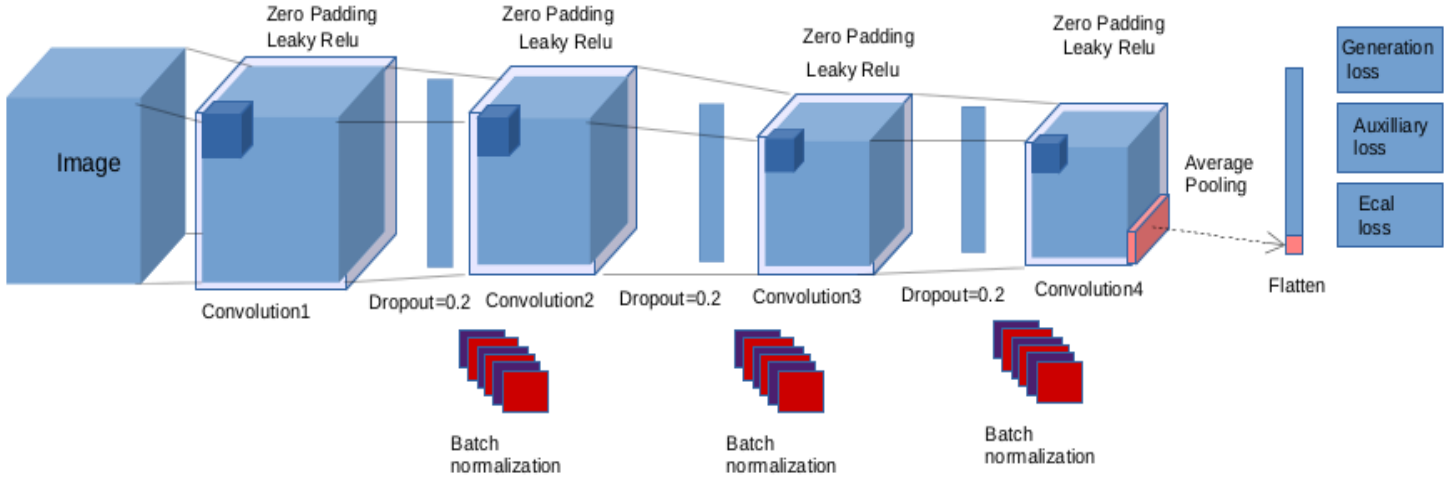
http://cds.cern.ch/record/2254048#

Our model: 3D convolutional GAN

Generator



Discriminator



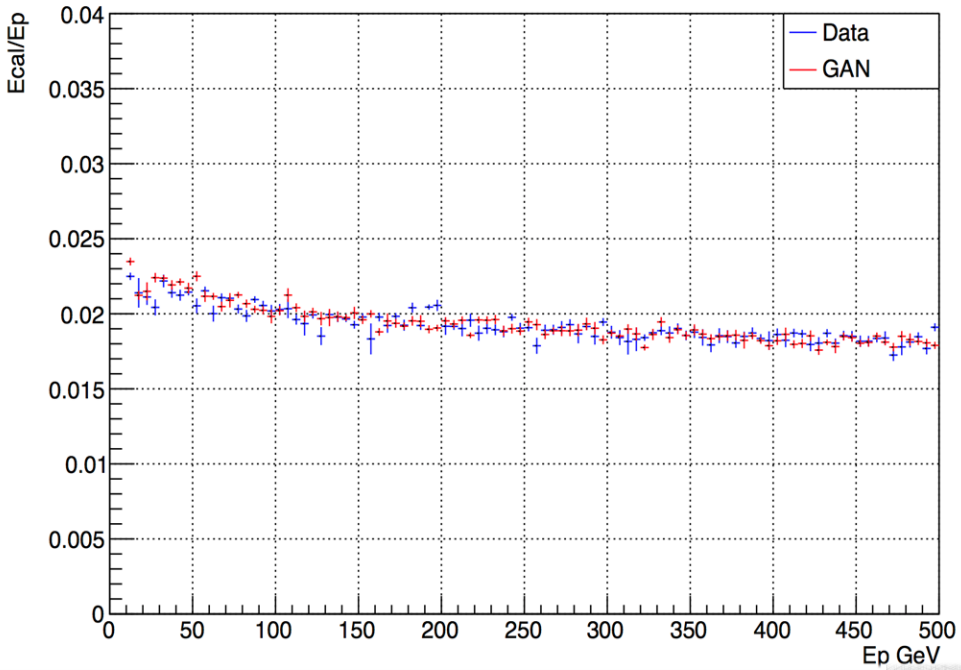
~1M parameters

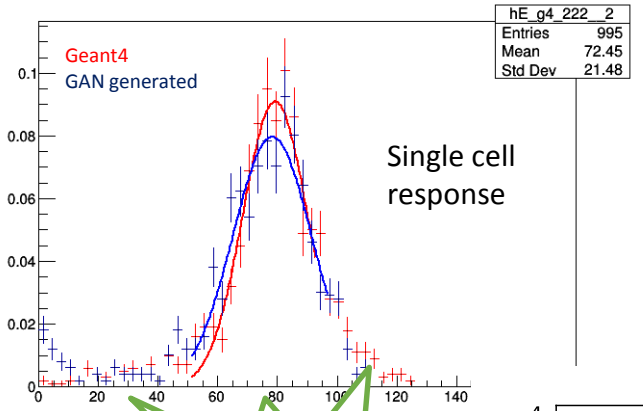
Total model Size: 3.8MB

openlab

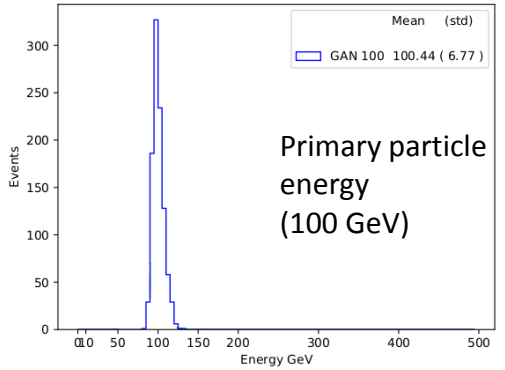
Physics simulation with GANs

Comparison to Monte Carlo





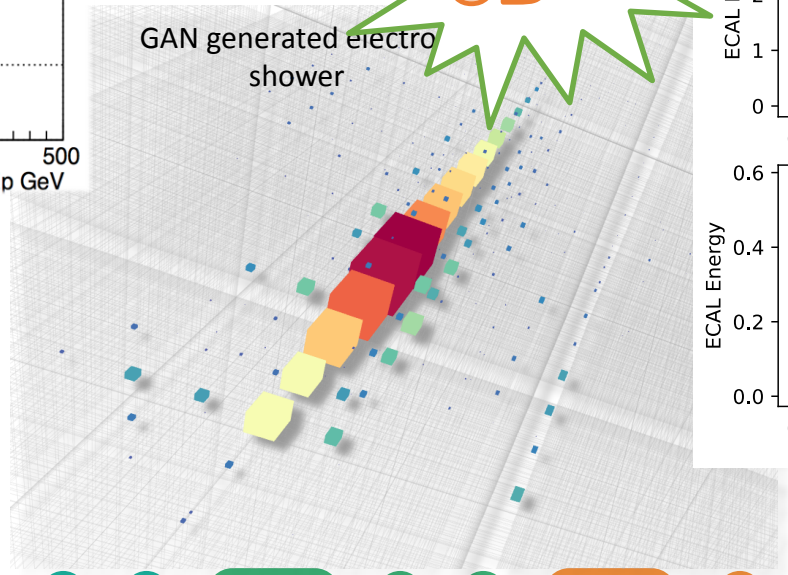
Single cell response

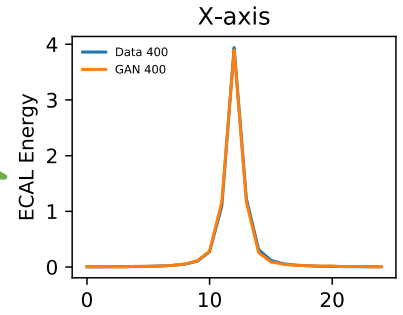


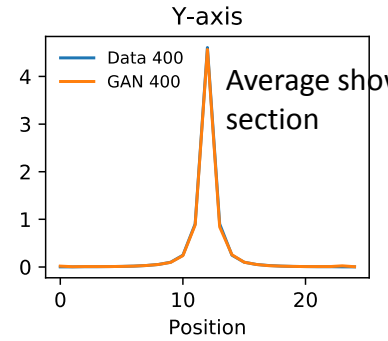
Primary particle energy (100 GeV)

3D

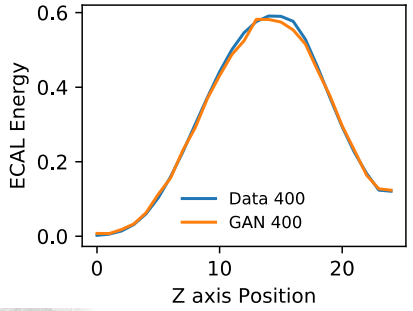
GAN generated electro shower

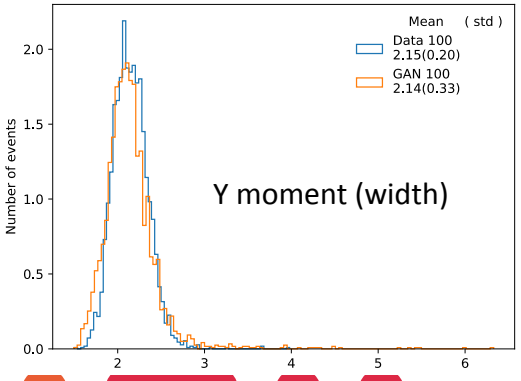






Average shower section





Y moment (width)

CERN openlab

Parallelizing Training

A fast training process requires a distributed approach

Distribute training via Horovod

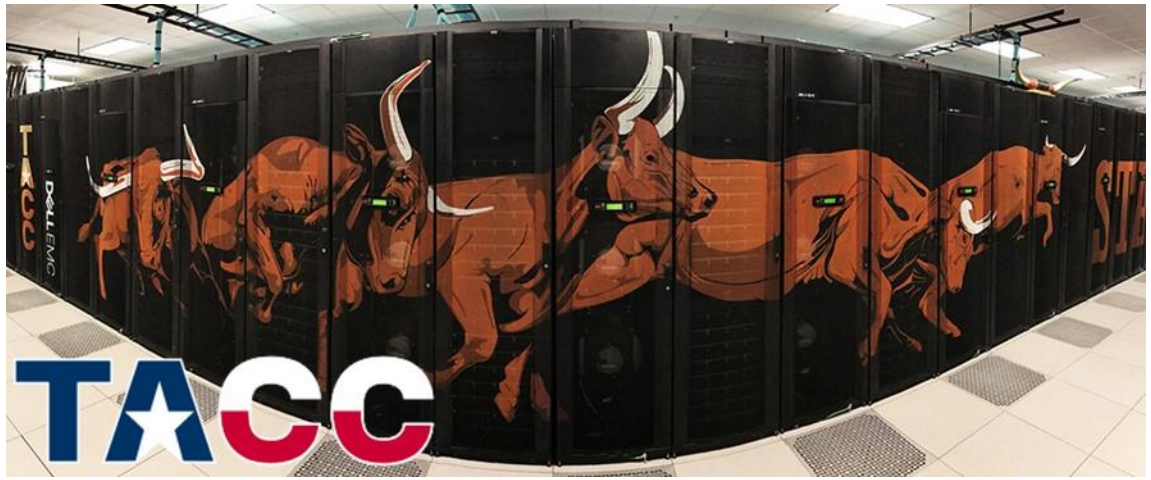
- Ensure data is loaded in parallel

Use Keras, TensorFlow + MKL-DNN

- MKL-DNN includes 3D Conv Support

Run on Stampede2 cluster

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



Single-Node optimisation

Training performance

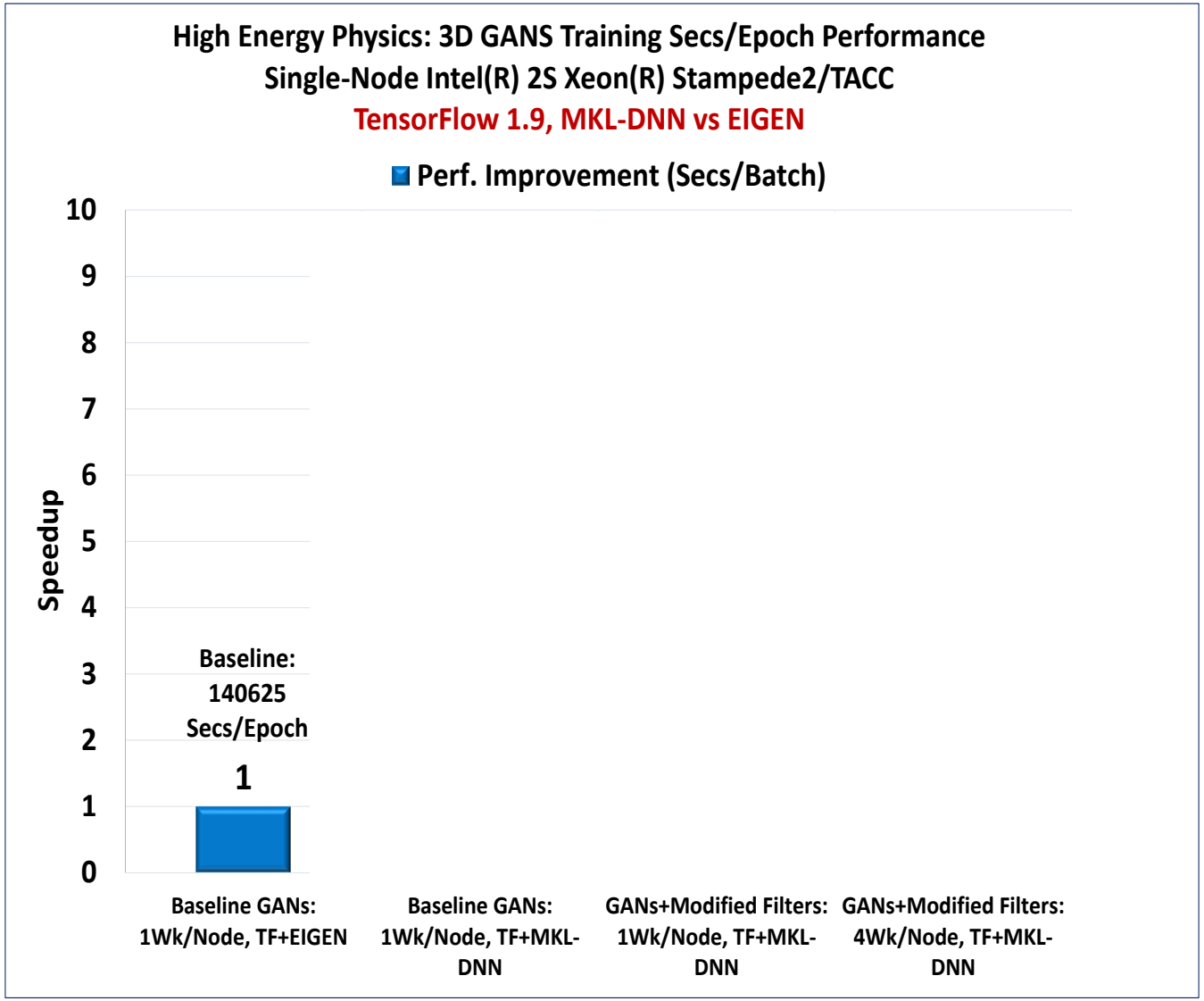
Our baseline:

1 worker/node TF + Eigen

Replace Eigen with MKL-DNN

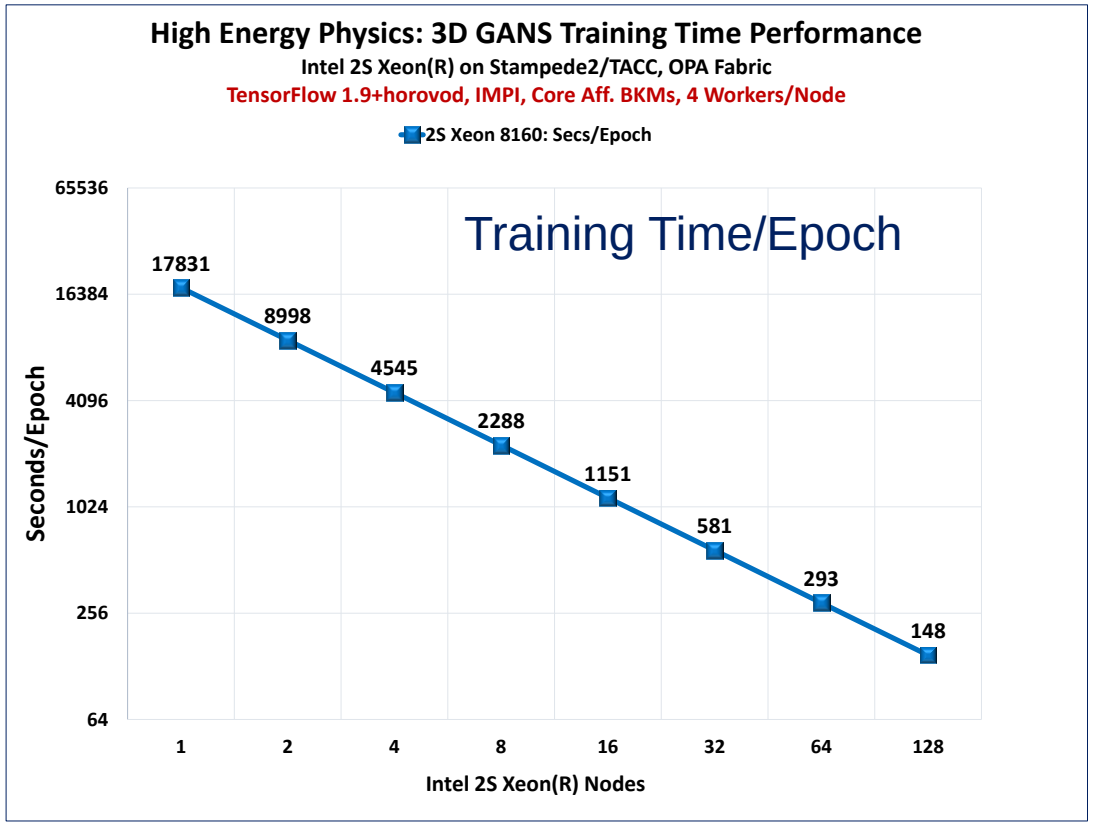
+ Optimize number of convolution filters

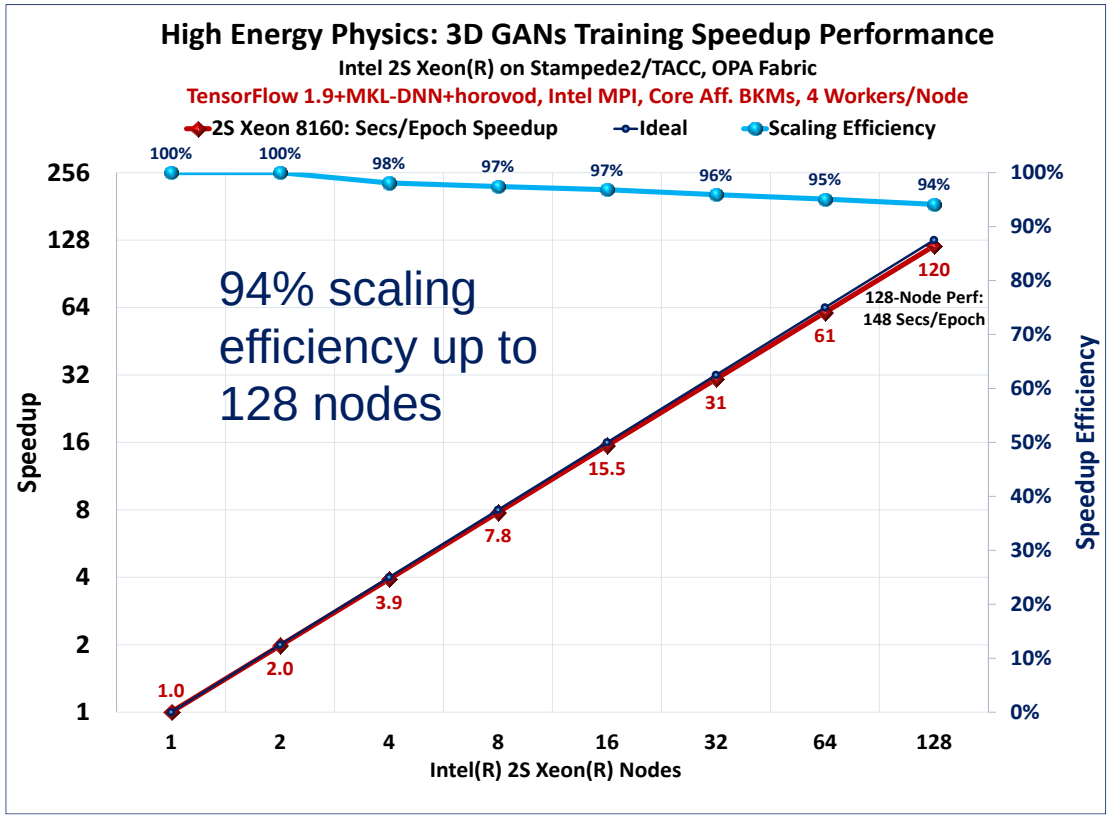
+ Parallelize to 4 workers/node



CERN openlab

Multi-Node Scaling Performance

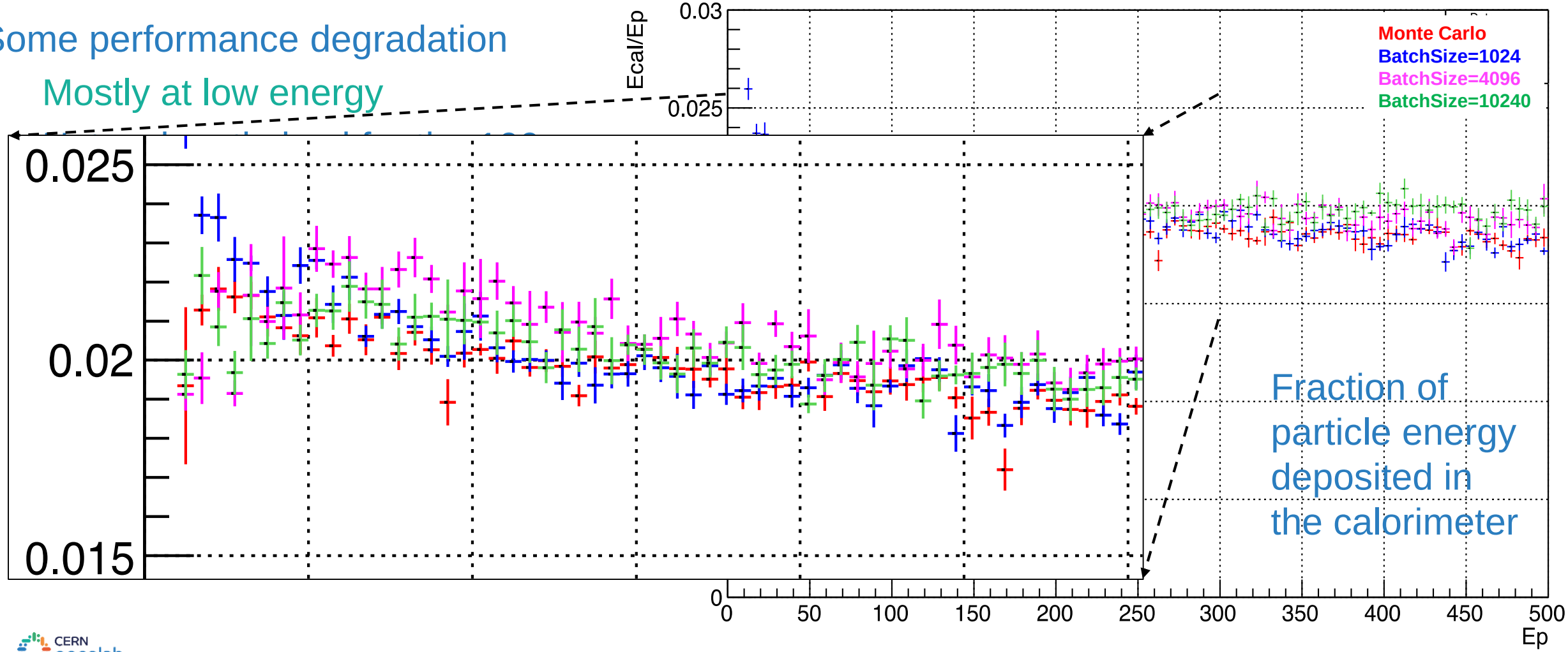




Physics performance

Some performance degradation

Mostly at low energy



Inference time

Inference:

Classical Monte Carlo simulation requires 17 s/shower

3DGAN (TF 1.4) takes 7 ms/shower

➔ speedup factor > 2500

Time to create an electron shower		
Method	Machine	Time/Shower (msec)
Classical Monte Carlo	2S Intel® Xeon® Platinum 8180	17000
3D GAN (batch size 128)	**2S Intel® Xeon® Platinum 8180**	**7**

Further optimsation (TF 1.9 + MKL-DNN with 3D/conv)

+ adjust TF (interop/intraop) parameters ~ **1 - 4 ms/shower * on a Intel Xeon Platinum 8160:**

+ use of multiple data streams:

1 stream 1.25 ms/shower

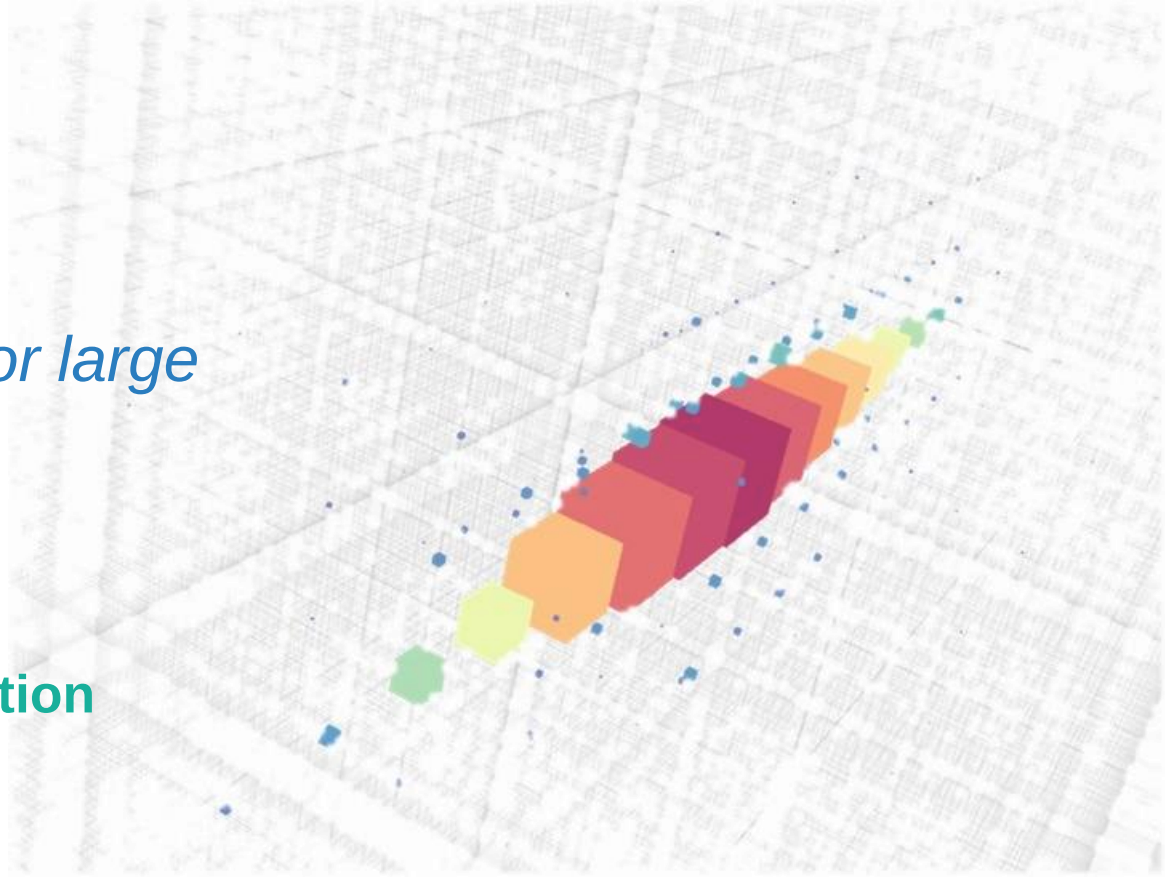
2 streams 0.93 ms/shower

4 streams 0.85 ms/shower

https://ark.intel.com	Intel Xeon Platinum 8180	Intel Xeon Platinum 8160
# cores	28	24
# threads	56	48
Base Frequency	2.5 GHz	2.1 GHz

*measured on 10000 showers

Summary



Extensive NN training will be a new workflow for large HEP experiments

Distributed training and HPC optimization is critical

- Enables **architecture optimization** and **generalization**
- **Increase the size of the problems** we can solve

Initial results on 3DGAN optimisation are very promising

- **Reduced training time by 8x on single node**
- Linear scaling brings down training time to ~2min/epoch on 128 nodes
- **Inference time is reduced by a factor 7x**

Preliminary results hints at better performance removing keras layer: we are investigating

Thank you

Questions?

sofia.vallecorsa@cern.ch,

damian.podareanu@surfsara.nl,

valeriu.codreanu@surfsara.nl

vikram.a.saletore@intel.com,

federico.carminati@cern.ch

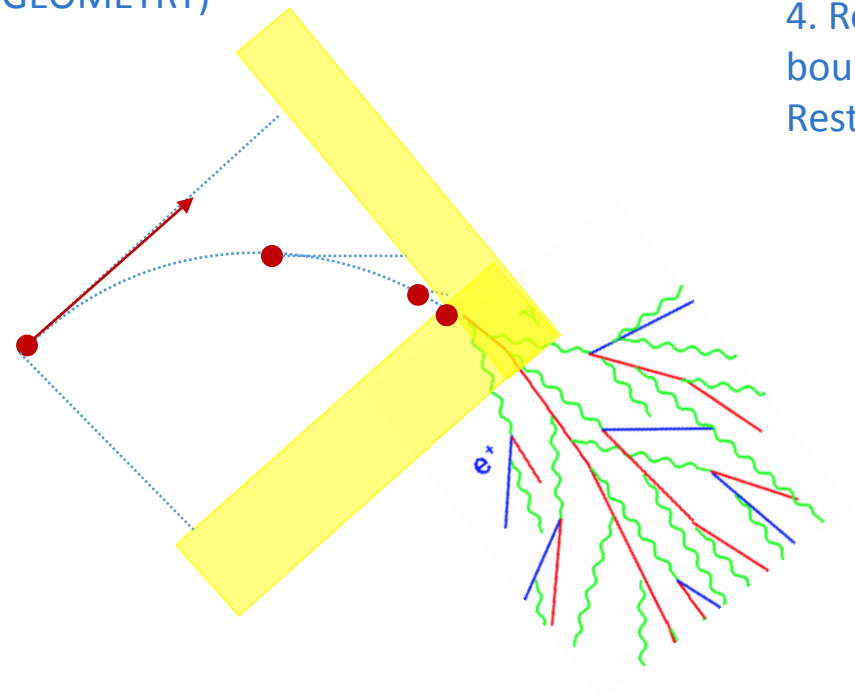
Classical Monte Carlo simulation

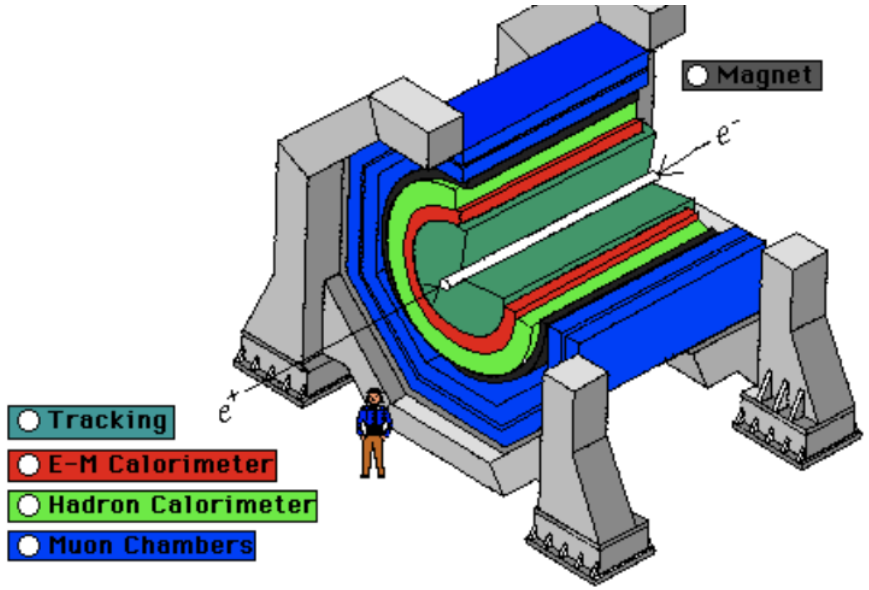
1. Calculate step particle could travel before doing a PHYSICS interaction

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

3. Propagate with selected step

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





Stampede2/TACC Configuration Details

Compute Nodes: 2 sockets Intel® Xeon® Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of 48 cores per node, 2 Threads per core, L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K, 96 GB of DDR4, Intel® Omni-Path Host Fabric Interface, dual-rail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP, 10 Gbit Ethernet, 200 GB local SSD, Red Hat* Enterprise Linux 6.7.

TensorFlow 1.6: Built & Installed from source: https://www.tensorflow.org/install/install_sources

Model: CERN 3D GANS from https://github.com/sara-nl/3Dgan/tree/tf

Dataset: CERN 3D GANS from https://github.com/sara-nl/3Dgan/tree/tf

Performance measured on 256 Nodes with:
OMP_NUM_THREADS=24 HOROVOD_FUSION_THRESHOLD=134217728 export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2 \
mpirun -np 512 -ppn 2 python resnet_main.py --train_batch_size 8 \
--num_intra_threads 24 --num_inter_threads 2 --mkl=True \
--data_dir=/path/to/gans_script.py --kmp_blocktime 1

https://portal.tacc.utexas.edu/user-guides/stampede2

Architecture, Dataset & Runtime Options

Optimise filter sizes

- **Conv Filters**: Multiple of 16 (MKL-DNN optimizations)

Dataset: 200000 electrons

- **Training Samples**: 180000 & **Validation**: 20000

Batch Size: 8/Worker, # Workers/Node=4/Node (Mapped to NUMA domains)

TF tuning: inter_op: 2 & Intra_op: 11 (Xeon® 8160 is 24C/CPU); **AVX512 –FMA support**

Learning Rate: 0.001, **Optimizer**: RMSprop

Warmup Epochs: 5 (Facebook Methodology), **Training Epochs:** 25