Fast Simulation with Generative Adversarial Networks

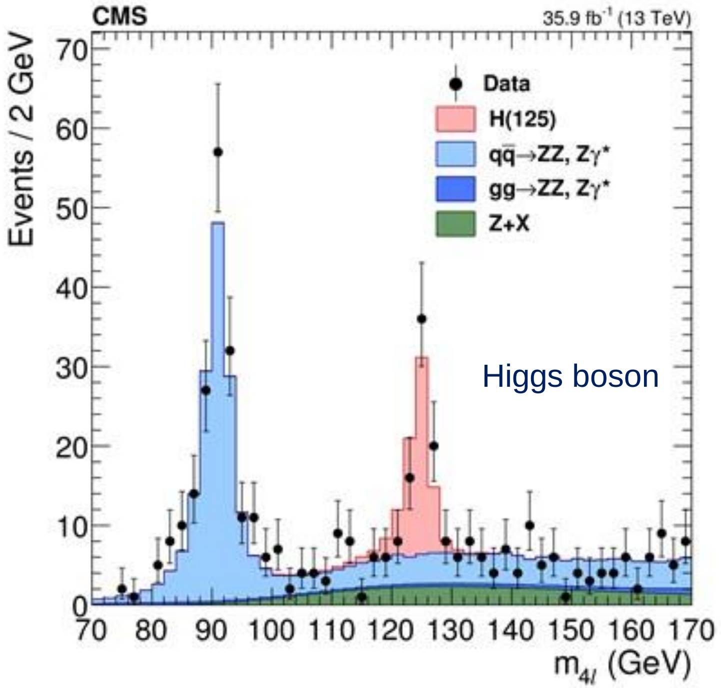
F. Carminati

S. Vallecorsa, G. Khattak

Outline

- Introduction
- Generative Models
  - Generative Adversarial Networks
- Our model
  - Physics Performance
  - Single-node Performance
  - Multi-node Performance
- Extending the application domain
- Summary

Monte Carlo Simulation



Essential for data analysis & detector design

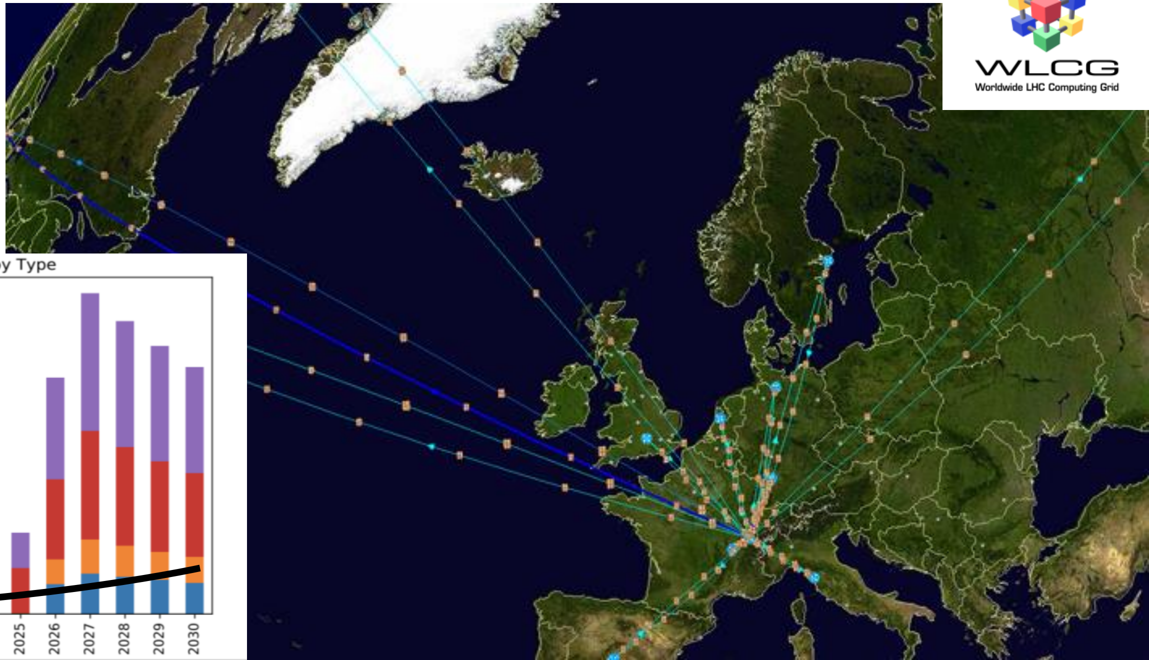
Understand how detector design affects measurements and physics

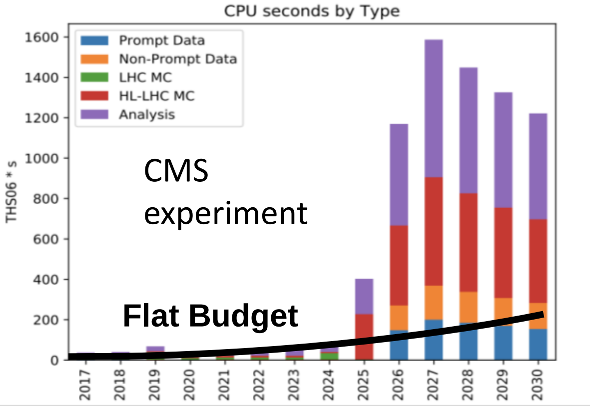
- Correct for inefficiencies, inaccuracies, unknowns
- Compare theory models to data

Complex physics and geometry modeling

- >50% of **Worldwide LHC Computing Grid (WLCG)** power today

Increase by 100x by 2025!



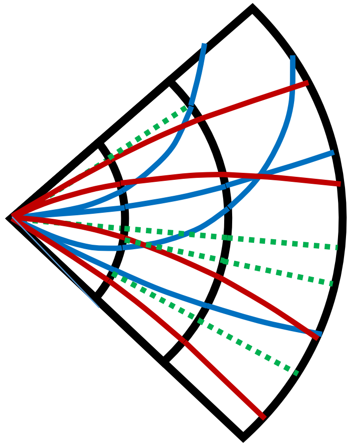


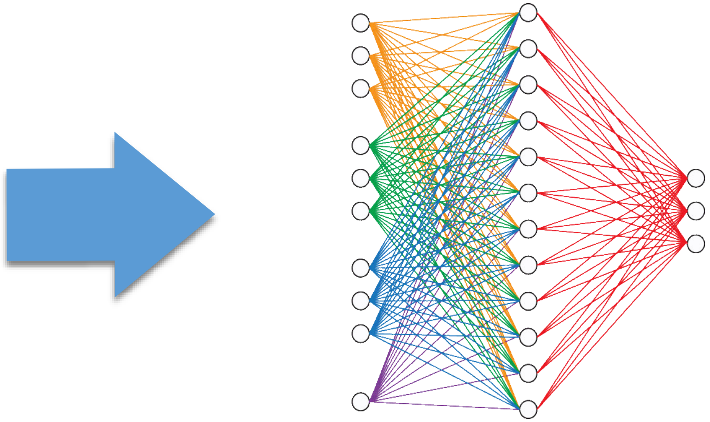
CERN openlab

Deep Learning for fast simulation

Replace Monte Carlo with a DNN that generates directly detectors output

- Accurate simulation results
- Fast inference step
- Generic customizable tool
  - Easily extensible framework to different detector use cases
- Complex architecture optimization
  - Training time under control
  - Scalability
  - Possibility to work across platforms





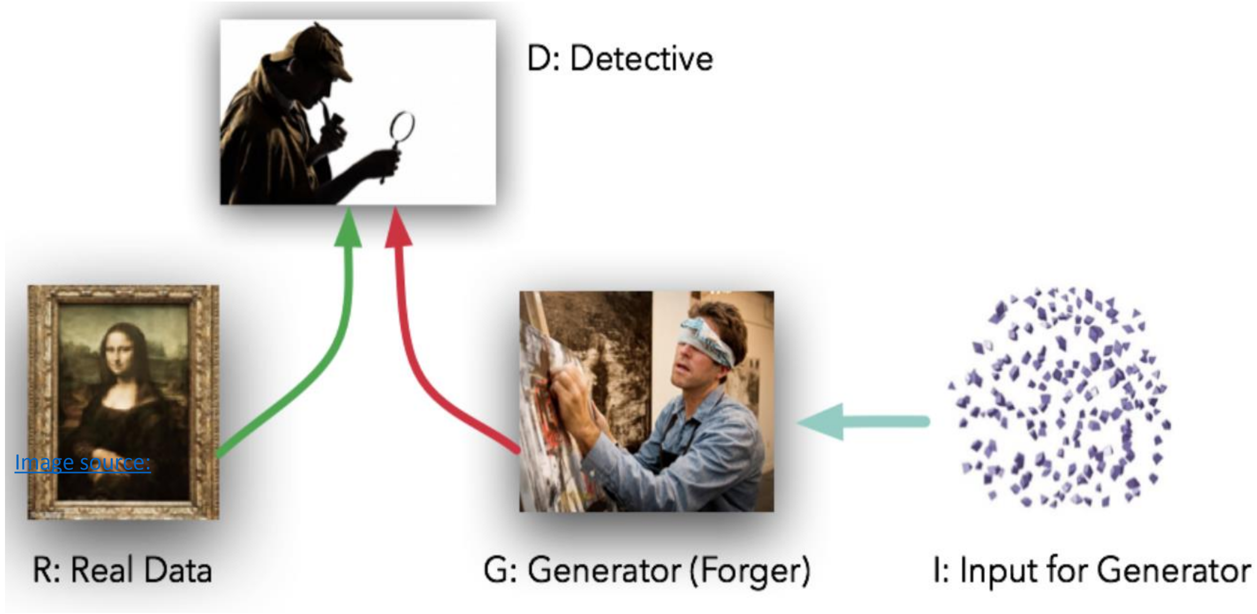
Goodfellow et al. : arXiv:1406.2661v1

Generative adversarial networks

Simultaneously train two networks that compete with each other

Generator G generates data from random noise

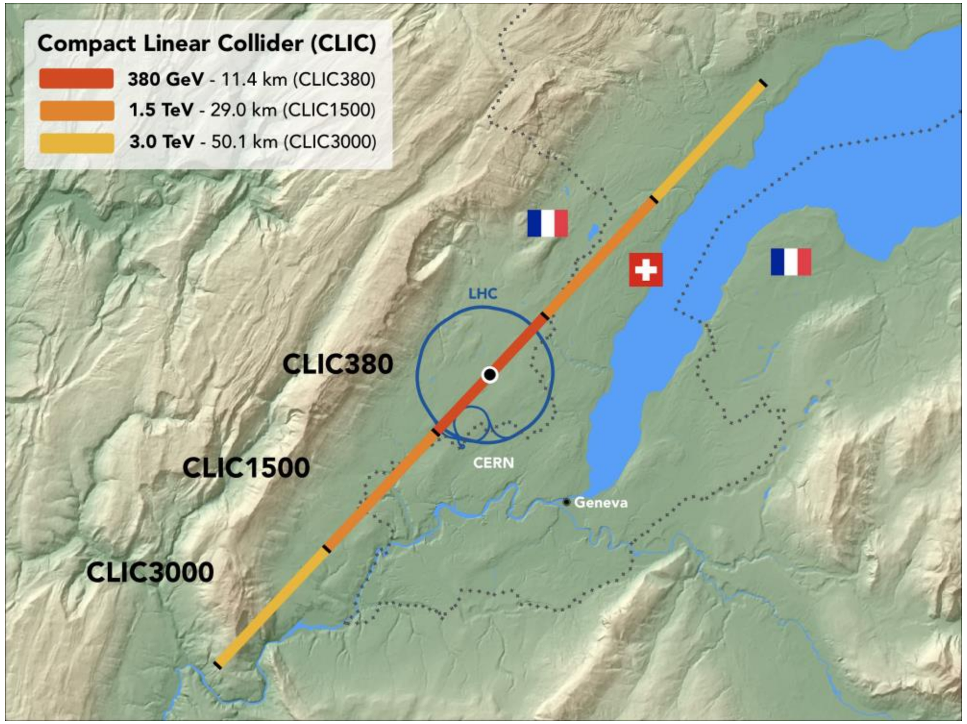
Discriminator D learns how to distinguish real data from generated data



Karras et al. ICLR2018

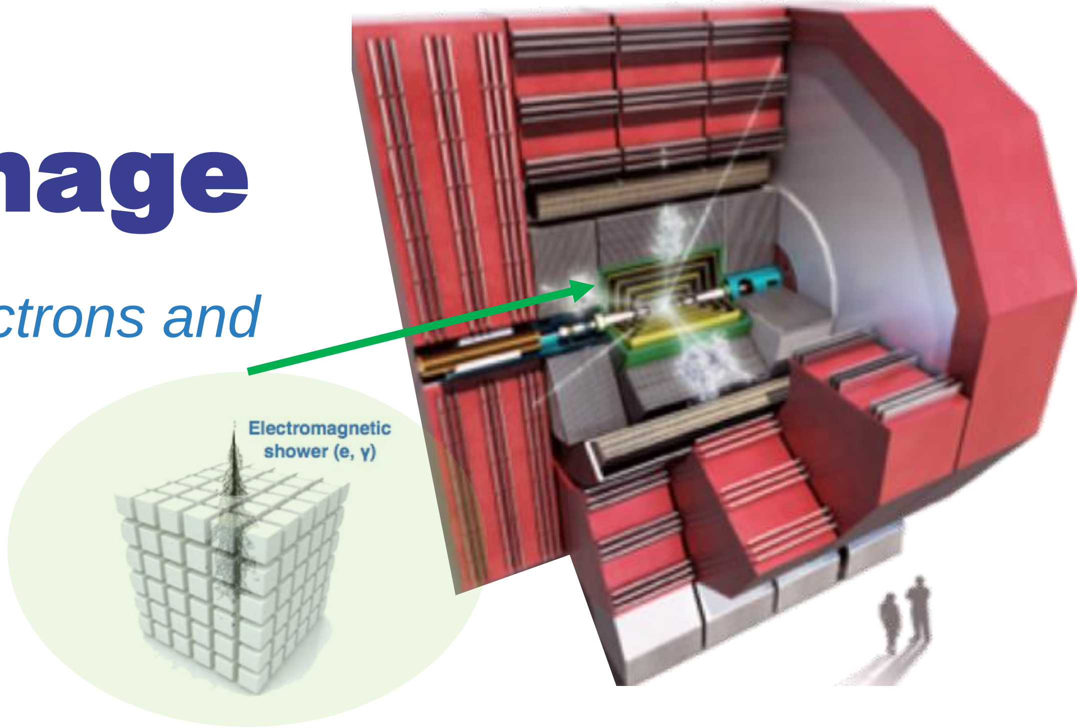
Compact LInear Collider

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



Detector output as 3D image

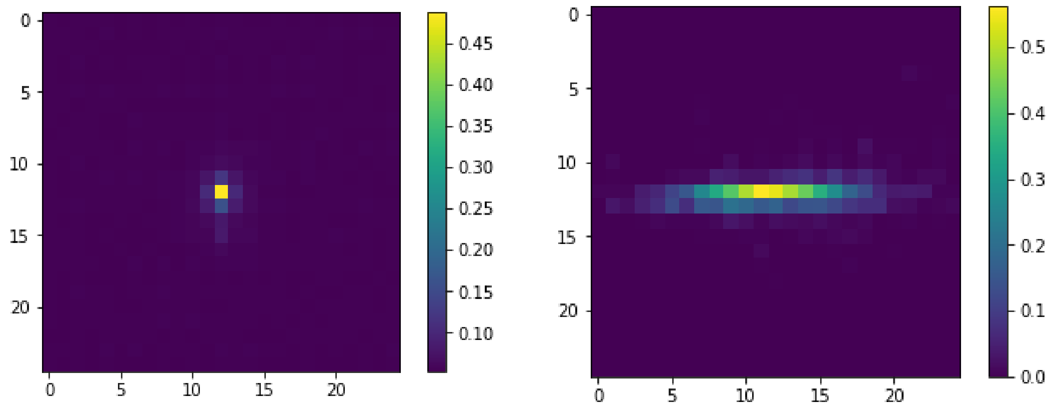
CLIC is a CERN project for a linear accelerator of electrons and positrons to TeV energies

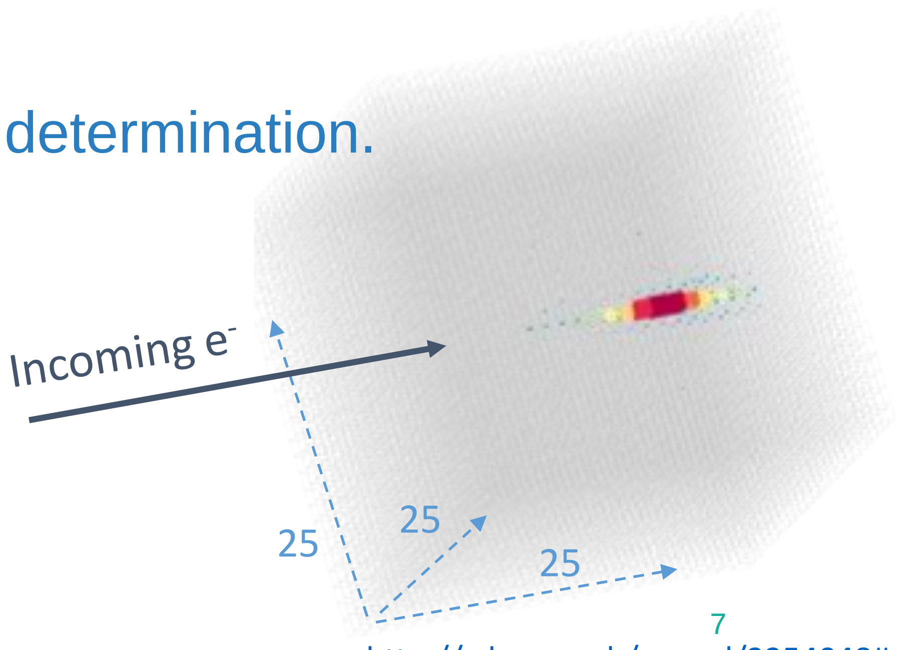


Electromagnetic calorimeter design

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

Segmentation is critical for particle identification and energy determination.

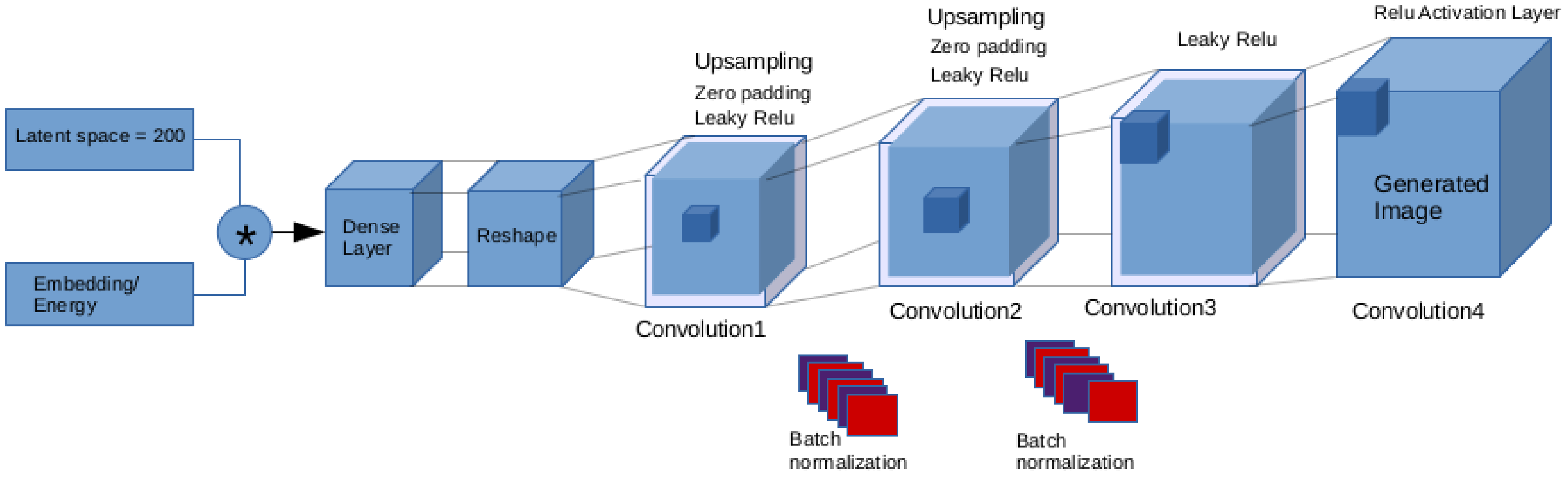




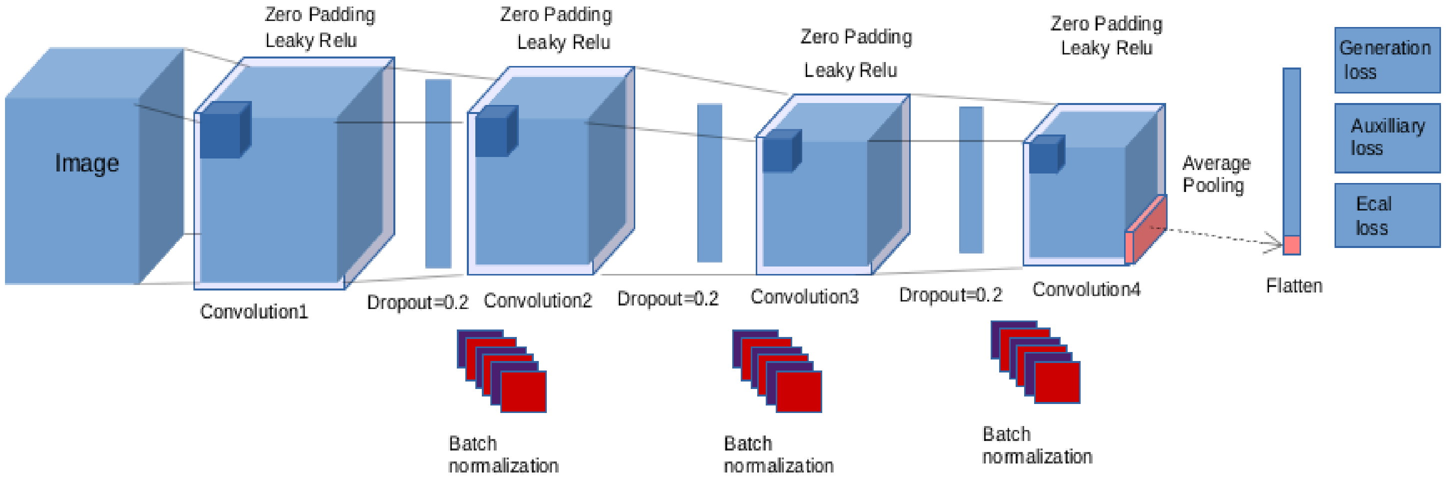
http://cds.cern.ch/record/2254048#

Our model: 3D convolutional GAN

Generator



Discriminator

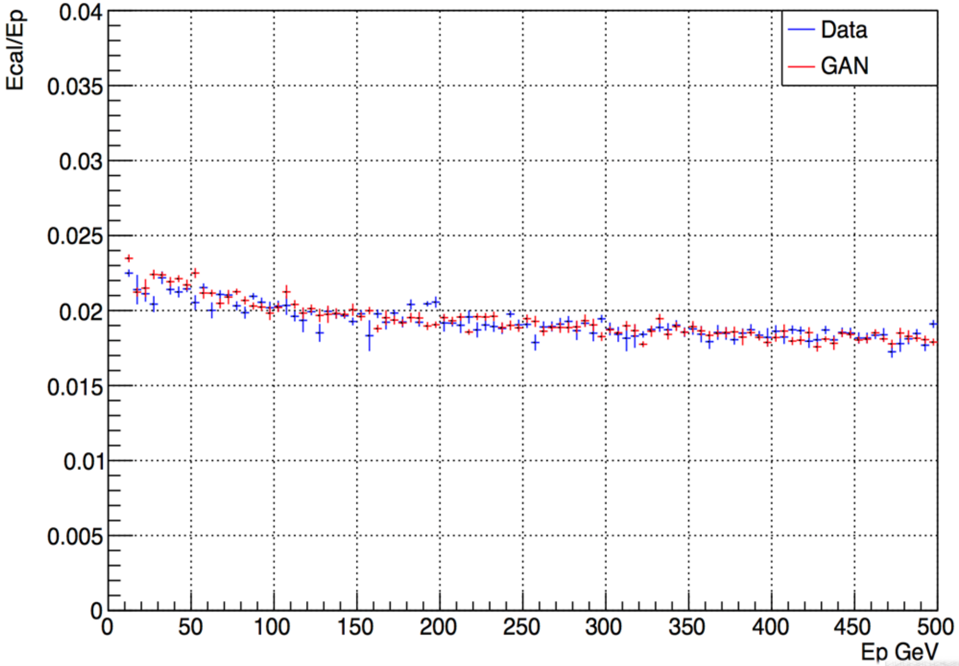


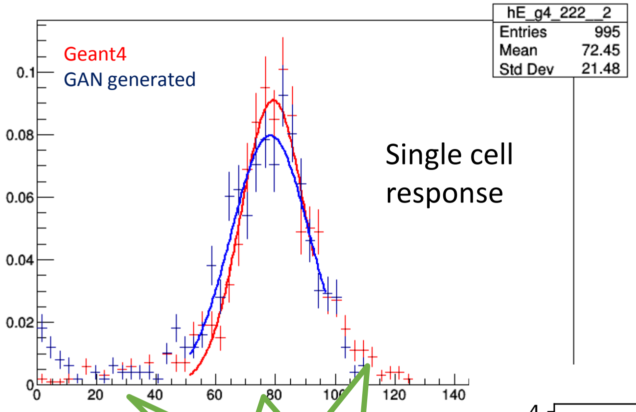
~1M parameters

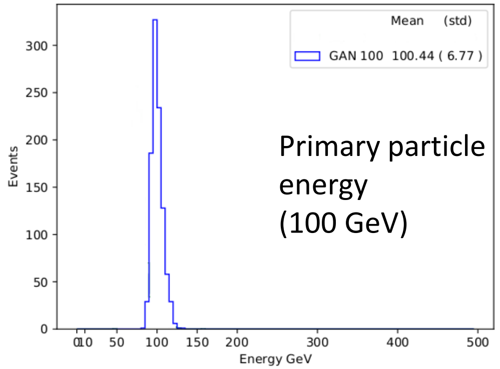
Total model Size: 3.8MB

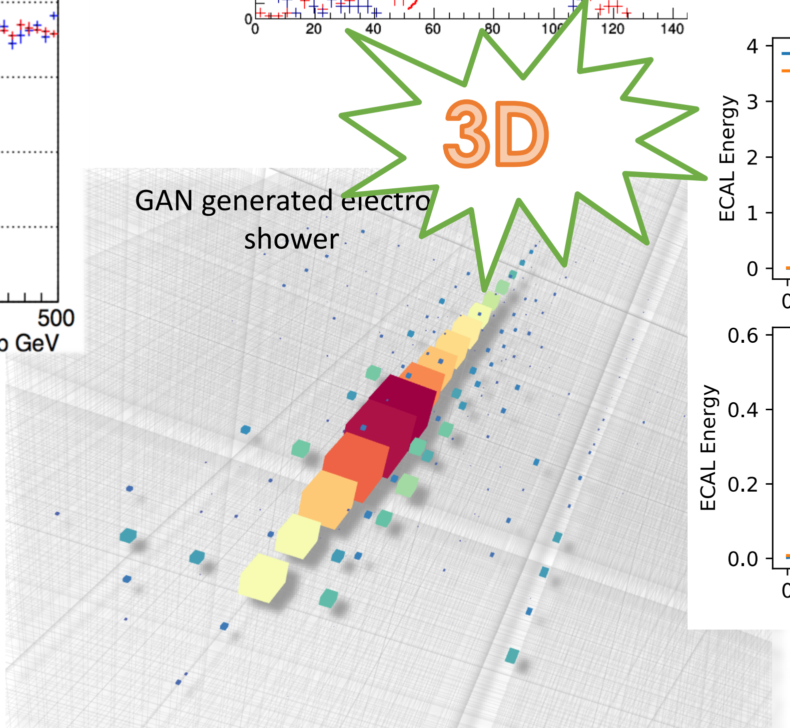
Physics simulation with GANs

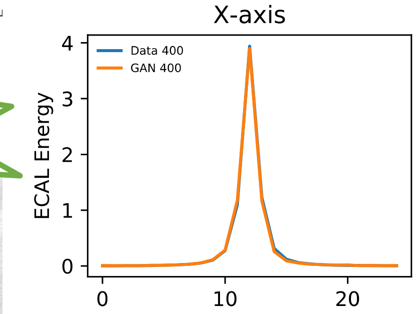
Comparison to Monte Carlo

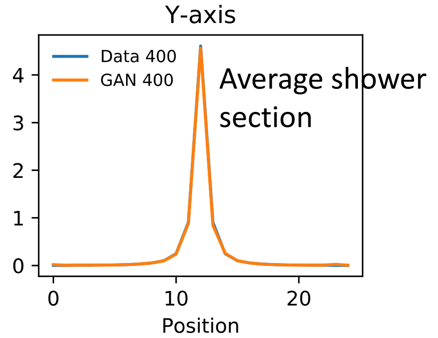


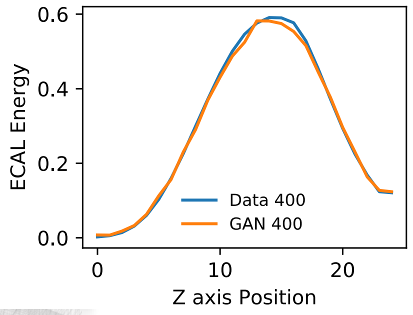


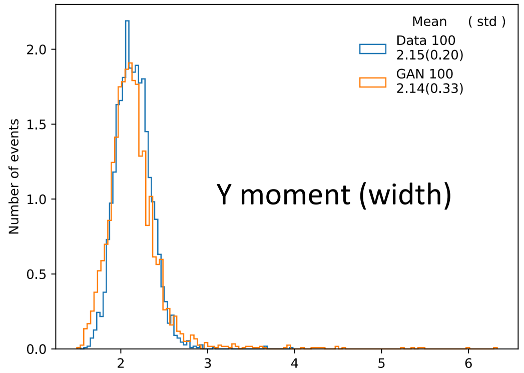












Generation speedup

Inference:

Classical Monte Carlo simulation requires 17 s/shower

3DGAN takes 7 ms/shower

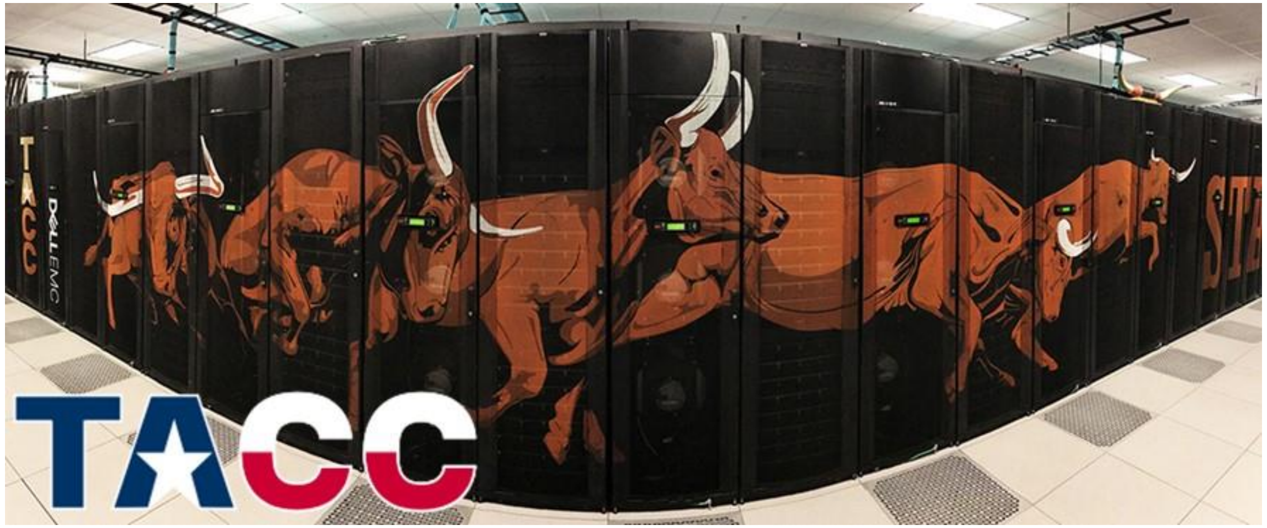
➔ speedup factor > 2500!!

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**

CERN openlab

Parallelizing Training

- Keras
    - Simplicity and high productivity
- TensorFlow + MKL-DNN w/ 3D Conv Support
- Distribute training via Horovod
    - Ensure data is loaded in parallel
- 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

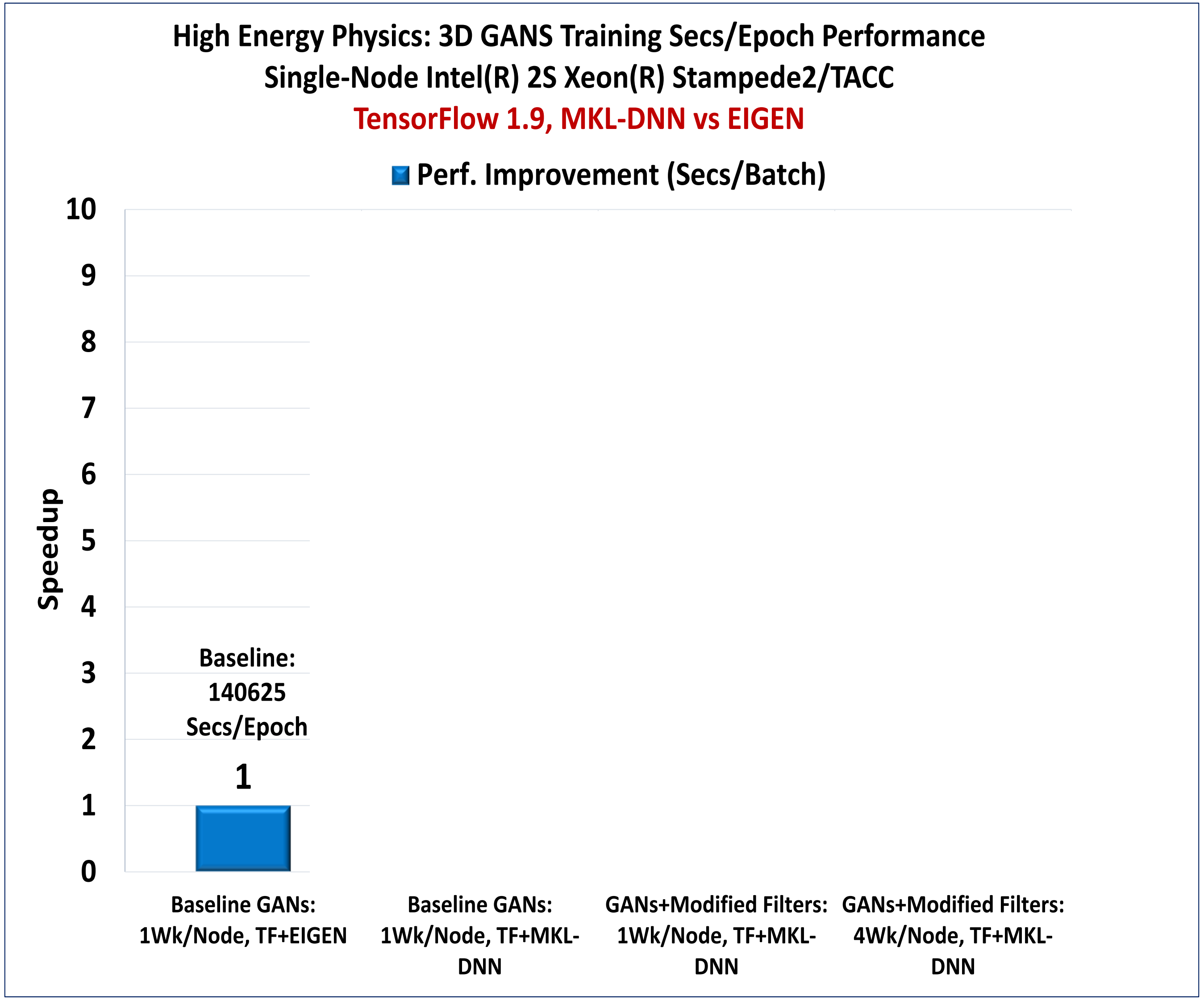
Our baseline:

1 worker/node TF + Eigen

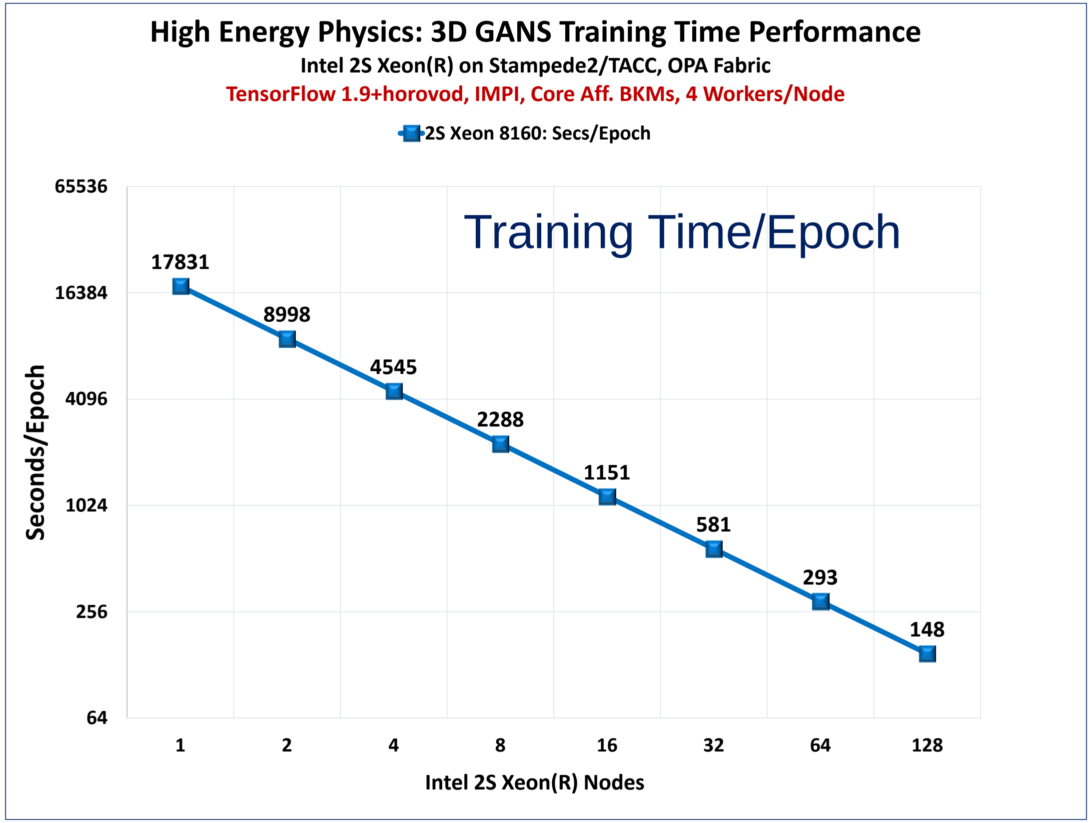
Replace Eigen with MKL-DNN

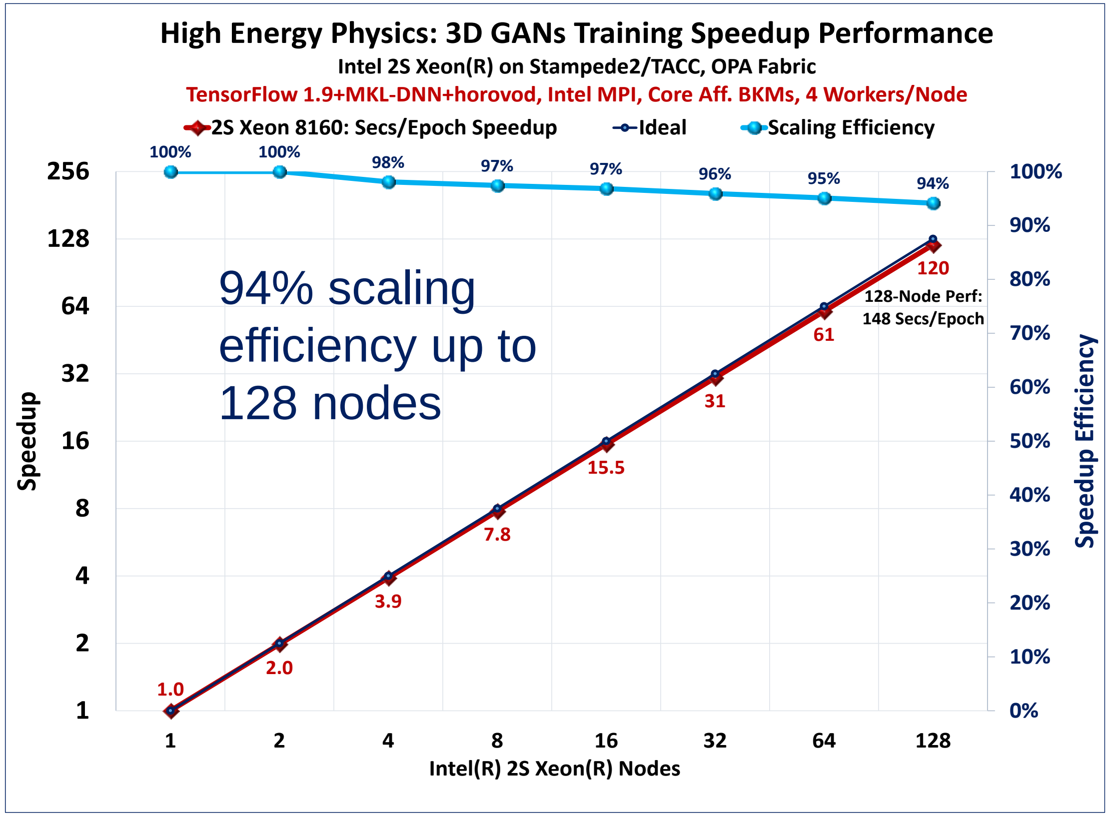
+ Optimize number of convolution filters

+ Parallelize to 4 workers/node



Multi-Node Scaling Performance

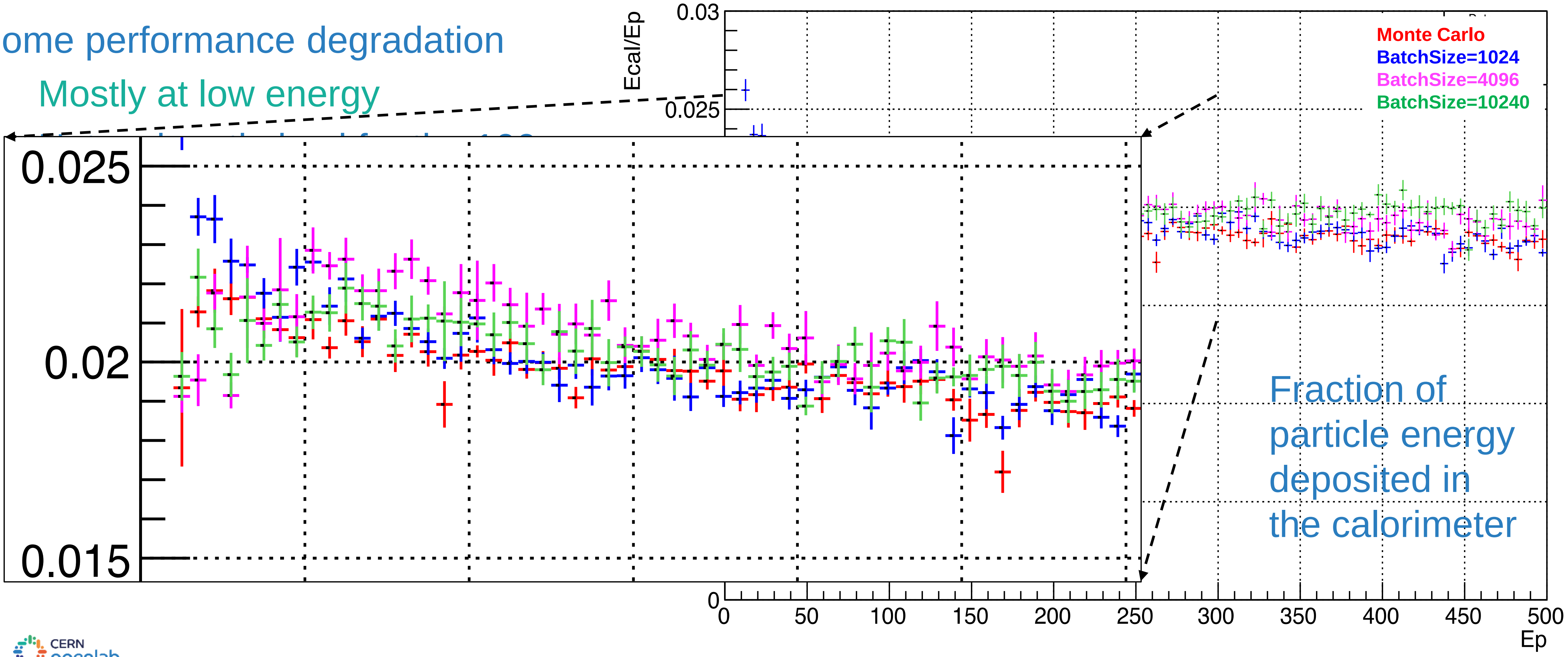




Physics performance

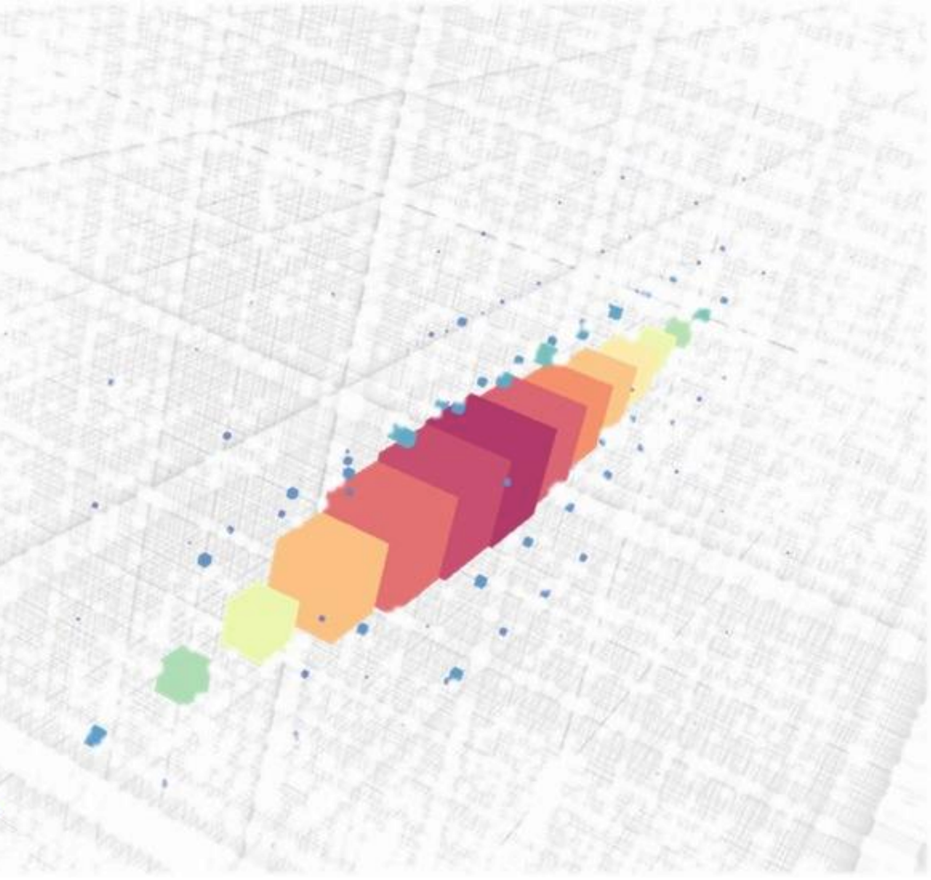
Some performance degradation

Mostly at low energy



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

Our initial results are very promising

- Reduced training time by 8x on single node
- Linear scaling brings down training time to ~2min/epoch on 128 nodes

2019 Plan

- Study generalisation to different detector use cases
  - Test available frameworks to perform hyper-parameter optimisation
    - Integration to distributed training approach to reduce training time. (mpi_learn/mpi_opt)
- Test integration with Big Data frameworks
  - BigDL and Spark
- Test dedicated hardware when available
  - Future Nervana platforms for training and inference
  - Optimise inference on integrated FPGA systems

Thank you

Questions?

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