3D convolutional GAN for fast simulation

F. Carminati, G. Khattak, S. Vallecorsa

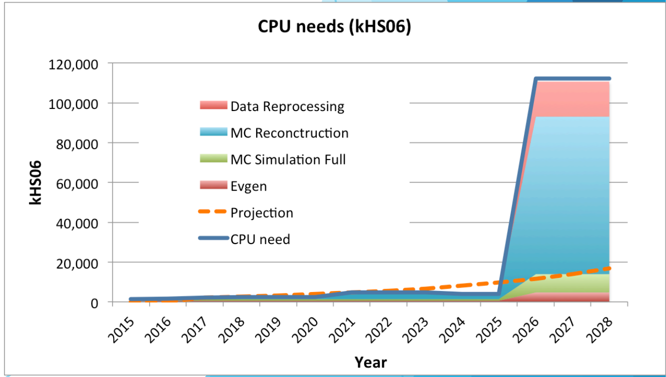
Outline

- Introduction
- Status
  - Generative Adversarial Networks for calorimeter simulation
  - Physics performance validation
- Plan for 2018
  - Generalisation
  - Optimisation of computing resources
- Summary

Introduction

- Detailed simulation has heavy computation requirements
- Activities on-going to speedup Monte Carlo techniques
  - Current code cannot cope with HL-LHC expected needs
- Improved, efficient and accurate fast simulation
  - Currently available solutions are detector dependent
- A general fast simulation tool based on Deep Learning
  - ML techniques are more and more performant in different HEP fields

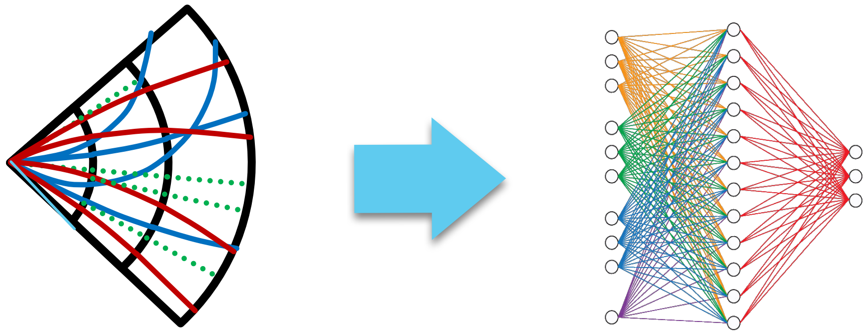
ATLAS experiment:



Campana, CHEP 2016

Deep Learning for fast simulation

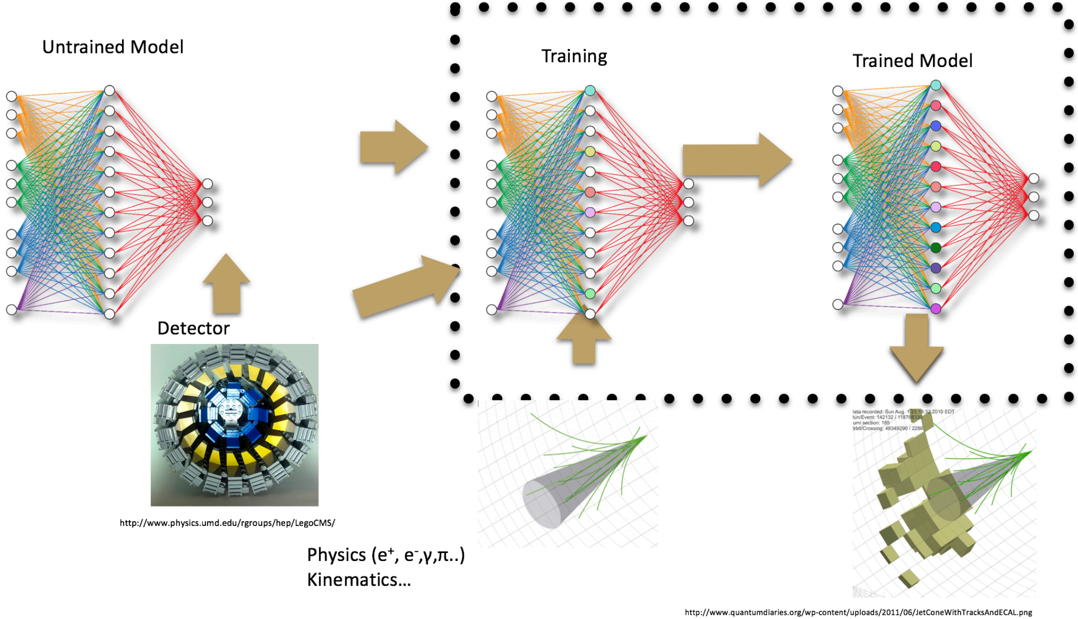
- Generic approach
- Can encapsulate expensive computations
- DNN inference step is faster than algorithmic approach
- Industry building highly optimized software, hardware, and cloud services.



- Use generative models to sample ralistic events from distributions
- Interpret detector output as images

A DL engine for fast simulation

- Start with time consuming detectors
- Next generation highly granular calorimeters
- Train on detailed simulation
  - Test training on real data
- Test different models
  - GAN, RNN, MPNN
- Embed training-inference cycle in simulation



Requirements

- A fast inference step:
  - It takes ~1 minute to simulate one electromagnetic shower with detailed simulation --> need at least a x100-1000 speedup
- Precise simulation results:
  - Need a detailed validation process
  - Probably cannot go below single precision floating points
- Generic customizable tool
  - Easy-to-use and easily extensible framework
- Large hyper parameters scans and meta-optimisation of the algorithm:
  - Training time under control
  - Scalability
  - Possibility to work across platforms

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

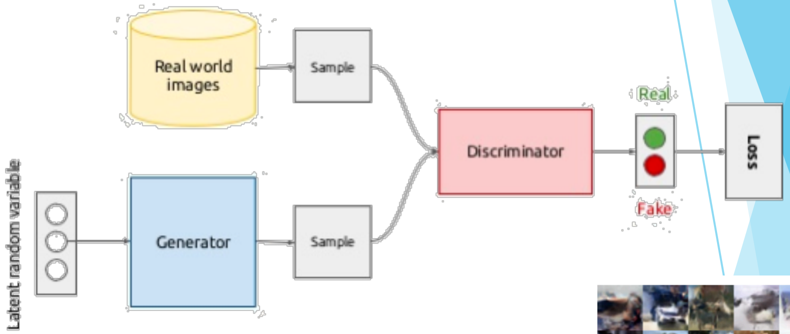
Proof of concept, benchmarking and validation

Generative Adversarial Networks

arXiv:1406.2661v1

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

- Generator learns to generate data starting from random noise
- Discriminator learns how to distinguish real data from generated data



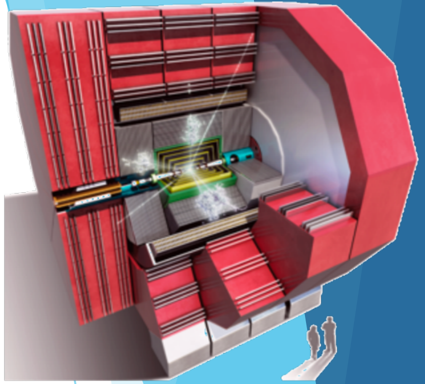
The counterfeiter/police case

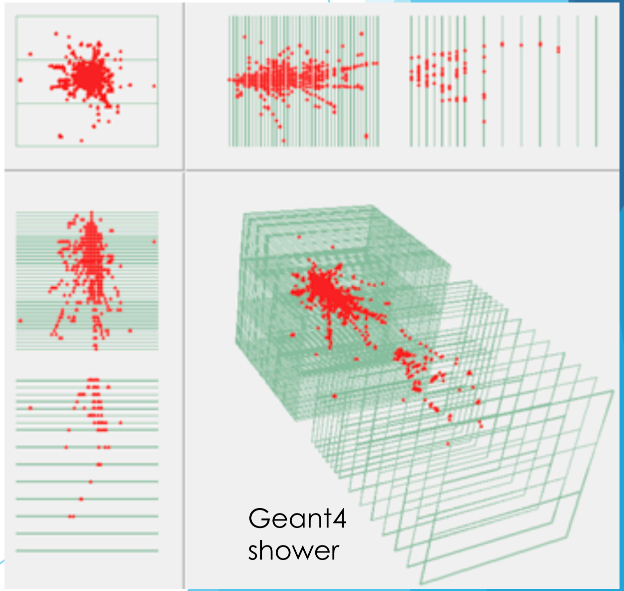
- Counterfeiter shows police the fake money
- Police says it is fake and gives feedback
- Counterfeiter makes new money based on feedback
- Iterate until police is fooled

GAN samples for CIFAR-10

CLIC calorimeter simulation

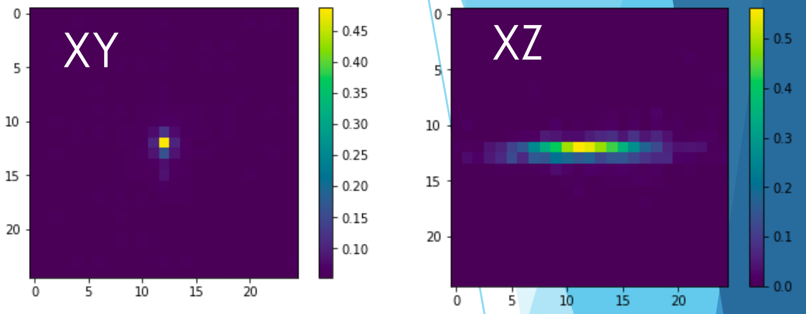
- Electromagnetic calorimeter detector design(*) within the Linear Collider Detector studies
- Highly segmented array of absorber material and silicon sensors (ECAL)
  - 1.5 m inner radius, 5 mm×5 mm segmentation: 25 tungsten absorber layers + silicon sensors
- 1M single particle samples (e,γ,π)
  - Flat spectrum (10-500) GeV
  - Orthogonal to detector surface
- +/- 10° random incident angle (NEW!)





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

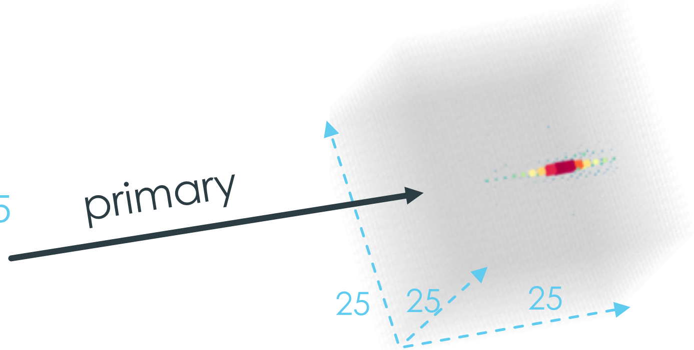
CLIC calorimeter simulation

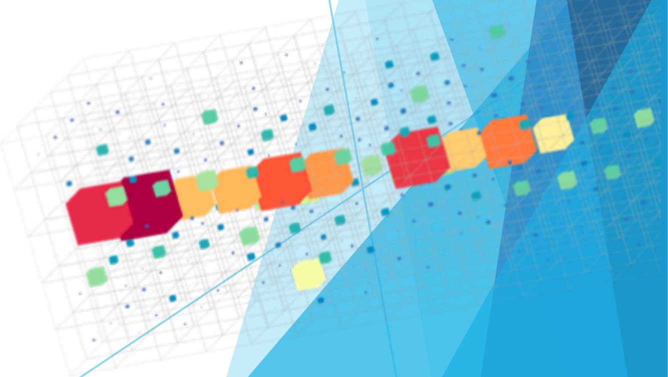


- Highly segmented
    - Segmentation is critical for particle identification and energy calibration.
- Sparse.
- Non-linear location-dependency

Data is essentially a 3D image

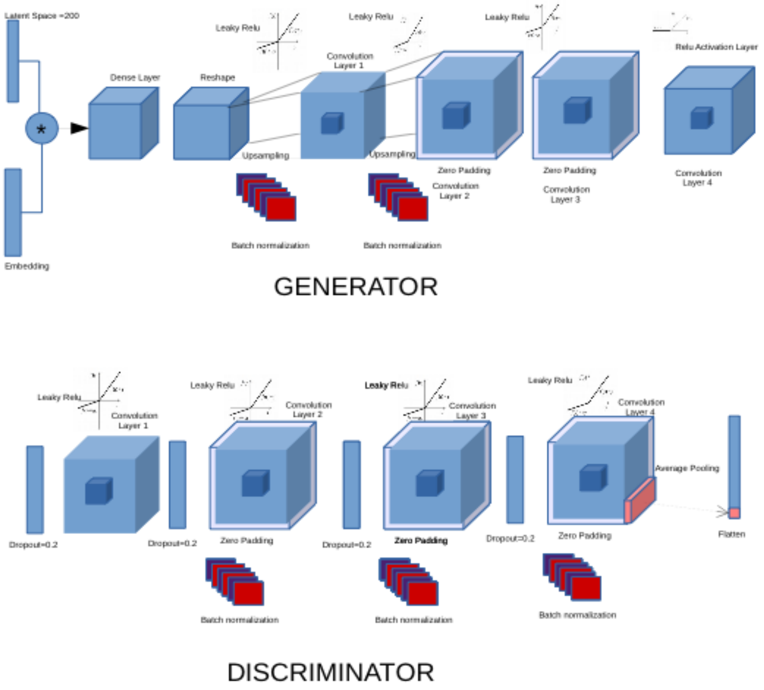
Stored as a 25x25x25 HDF5 dataset





3D convolutional GAN

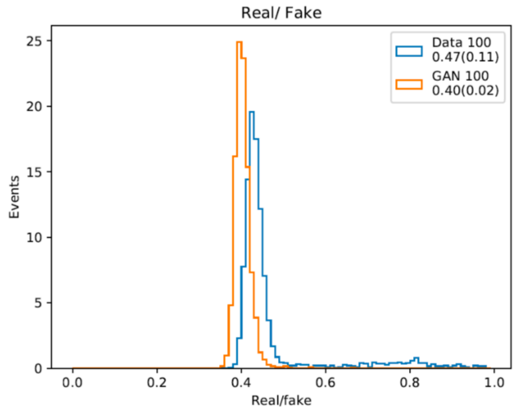
- Similar discriminator and generator models
  - 3d convolutions (keep X,Y symmetry) describe full shower development
- Tested several tips&tricks from literature*
  - Some helpful (no batch normalisation in the last step, LeakyRelu, no hidden dense layers, no pooling layers)
  - RMSProp optmiser for both networks
- Batch training
- Implementation in keras (TF backend)

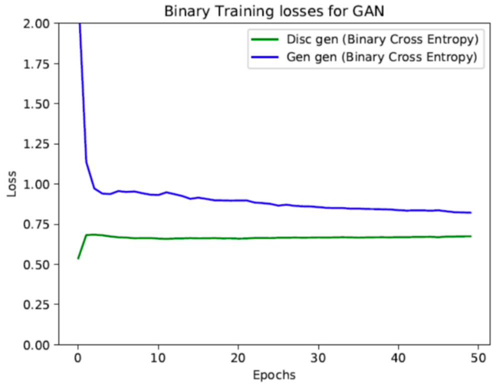


*https://github.com/soumith/ganhacks

Conditioning and auxiliary tasks

- Condition training on several input variables (particle type, energy)
- Auxiliary regression tasks assigned to the discriminator: primary particle energy and deposited energy
- Loss is linear combination of 3 terms:
  - Combined cross entropy (real/fake)
  - Mean absolute percentage error for regression tasks





Generalise to multi-class approach (or multi-discriminator approach): primary particle entry point, angle, etc..)

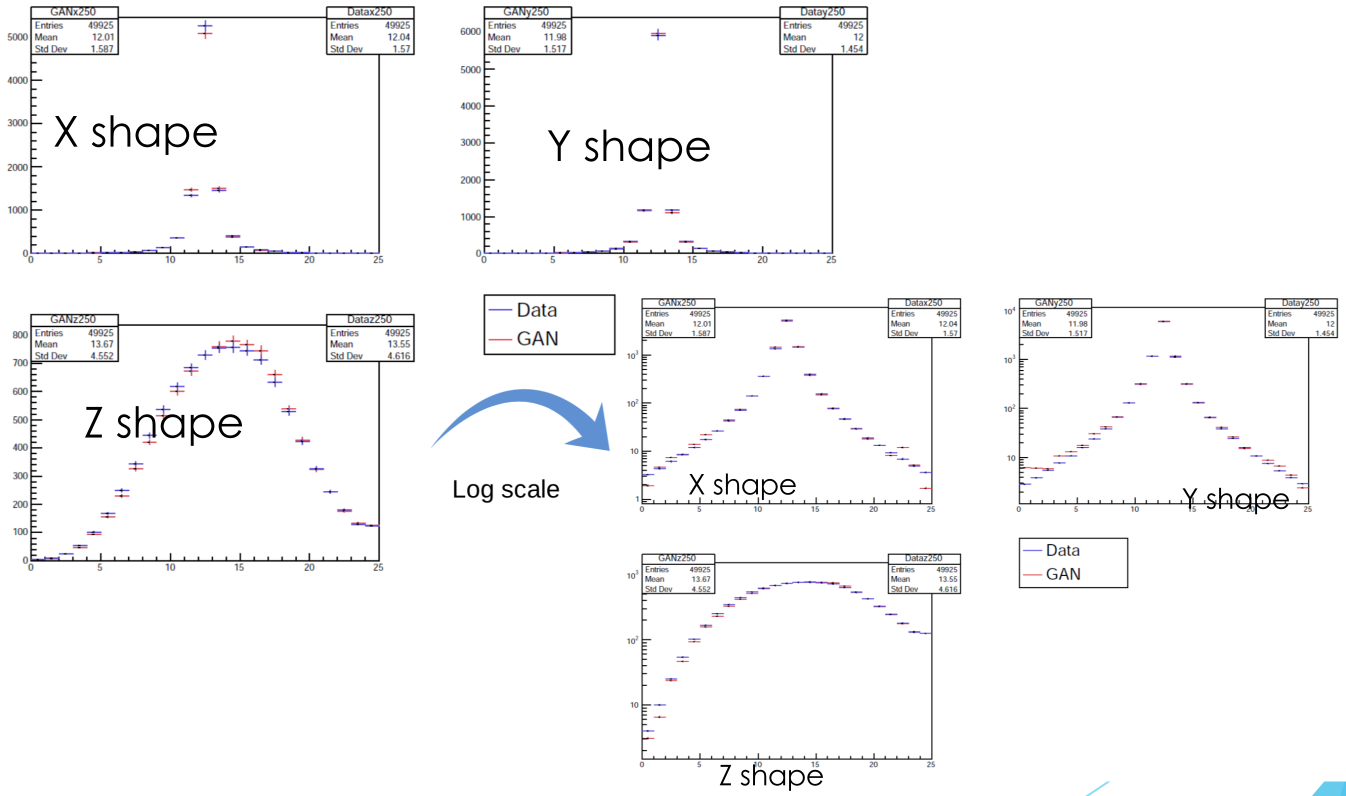
Validation and optimisation

- Detailed GAN vs GEANT4 comparison (More than 200 Plots!)
  - High level quantities (shower shapes)
  - Detailed calorimeter response (single cell response)
  - Particle properties (primary particle energy)
- Optimisation on
  - Network Architecture (Layers, filters, kernels, initialisation)
  - Losses definition
  - Data pre-processing
  - Rely on GAN losses only !! No physics variable explicitly constrained!
- Results agree within a few % to Geant4 (labelled "DATA" in next slides ☺)

We run on Caltech ibanks GPU cluster thanks to Prof M. Spiropulu

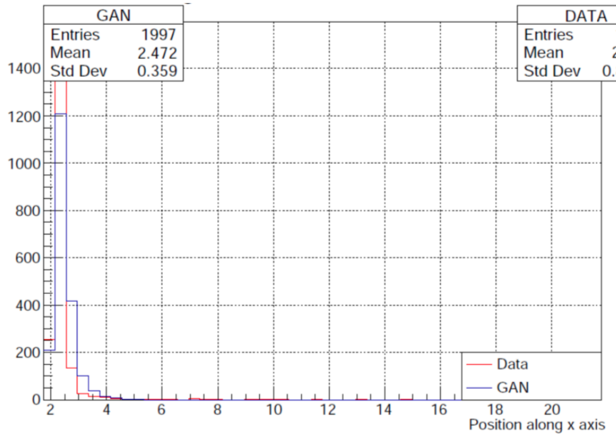
Shower shapes

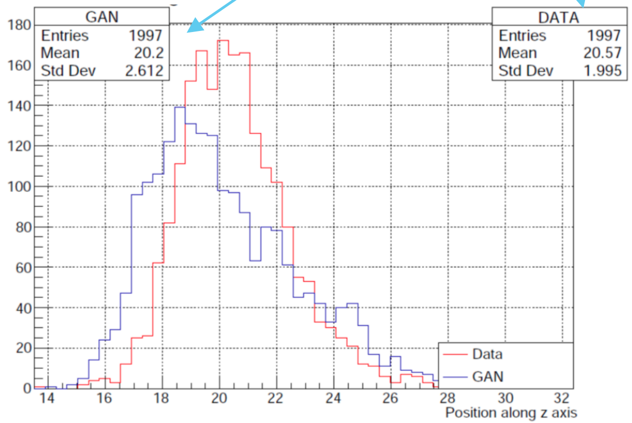
250 GeV electron



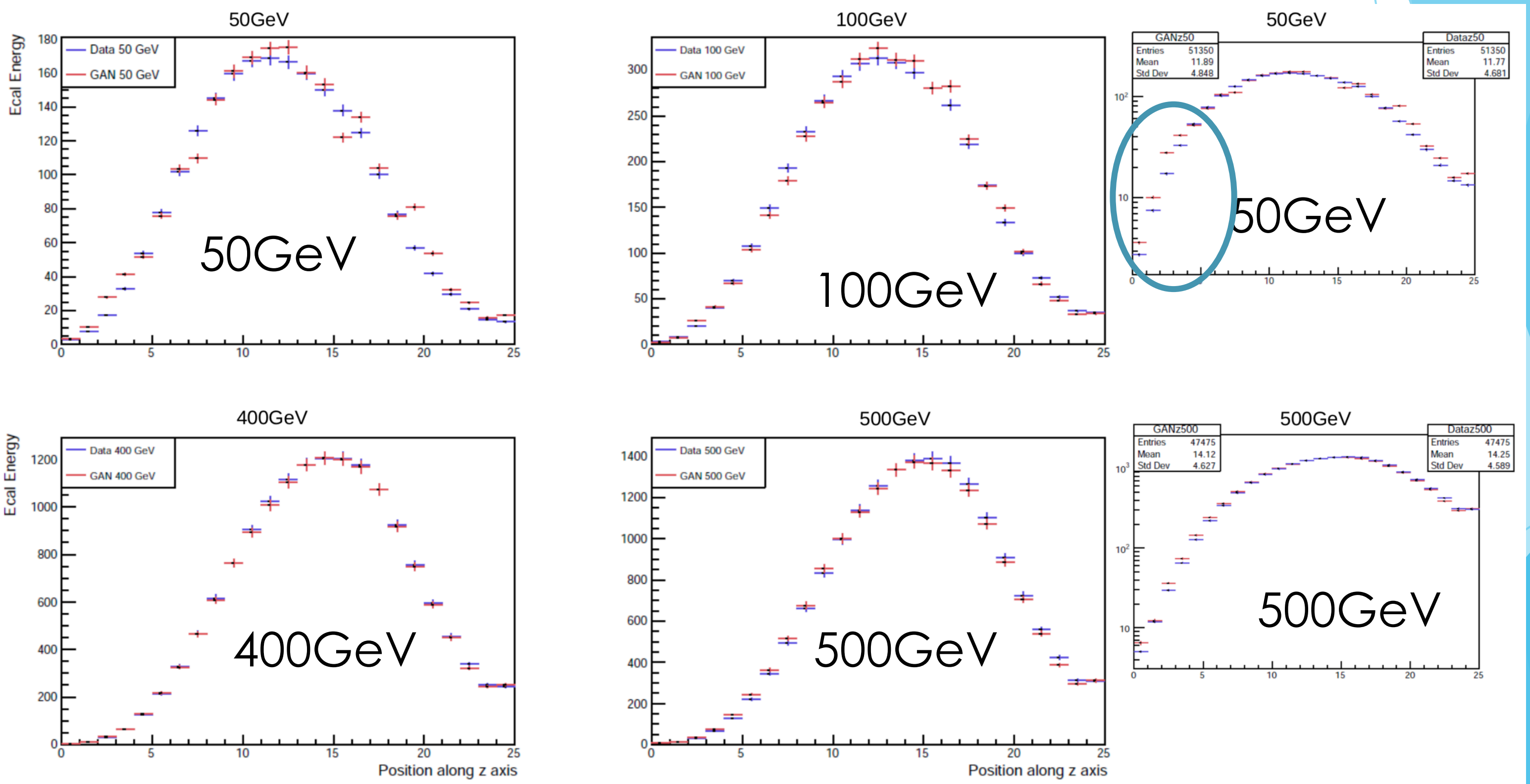
Shower shape moments: width

Central values are consistent
Stdev still slightly off

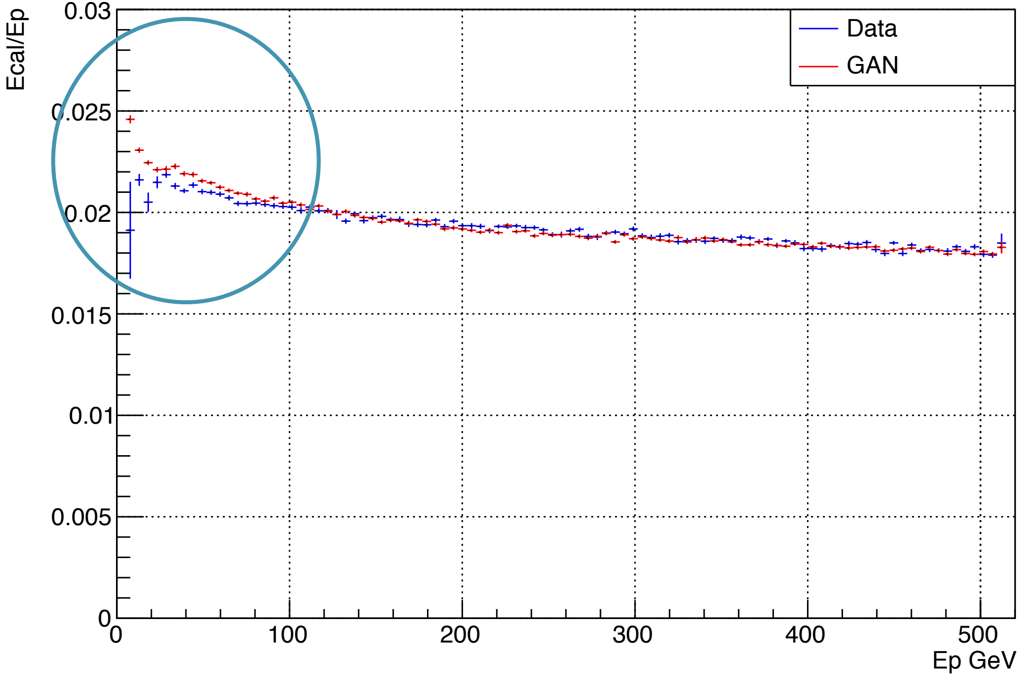




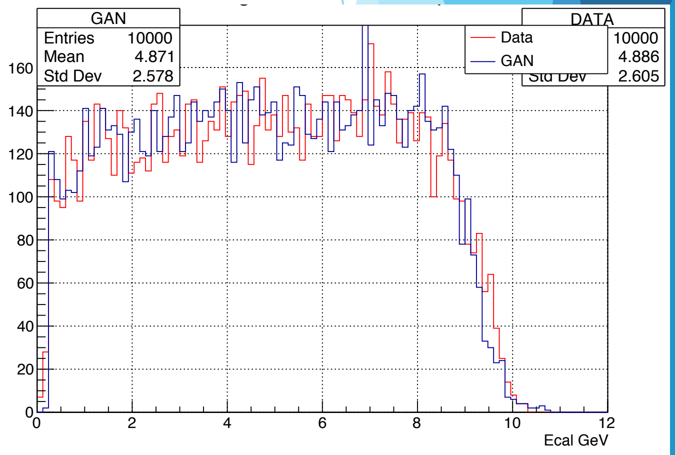
Shower shapes vs primary energy



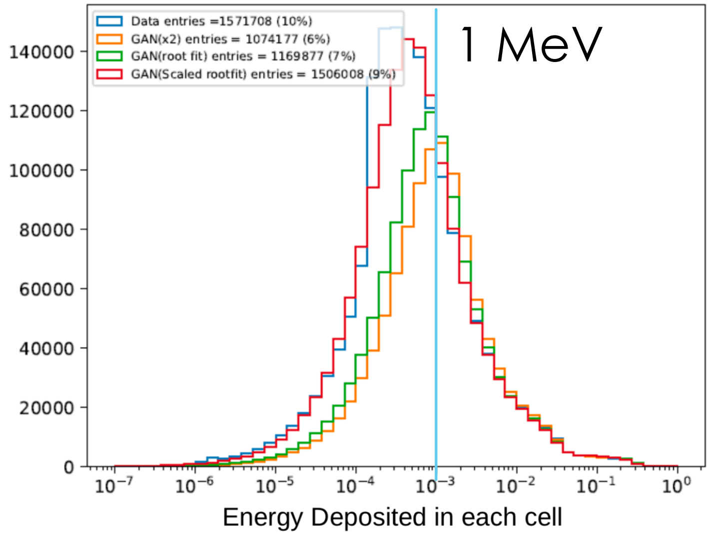
Calorimeter sampling fraction



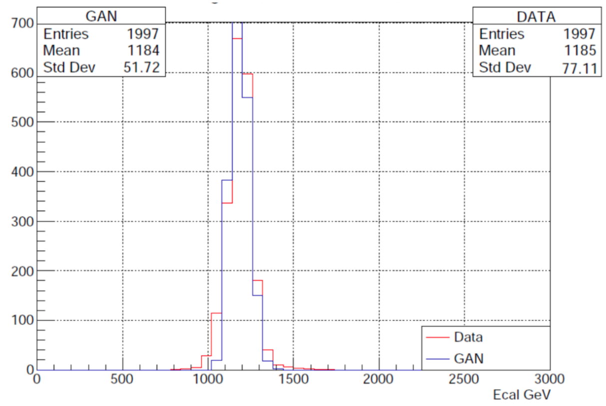
Total deposited energy



Low energy performance & single cells



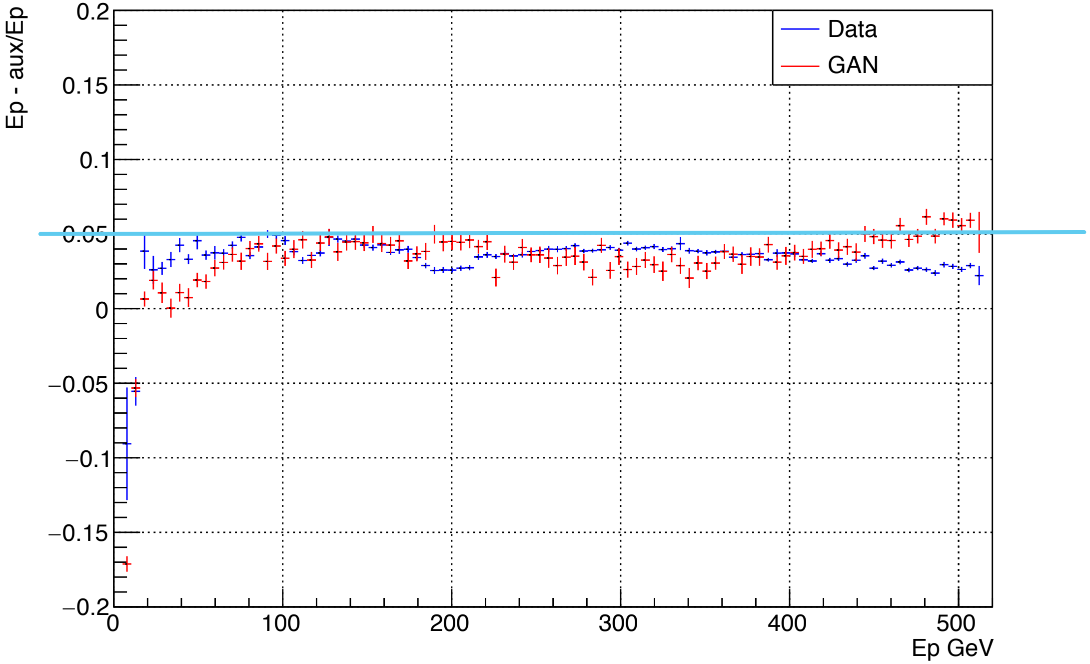
Number of hits (above 200 keV)



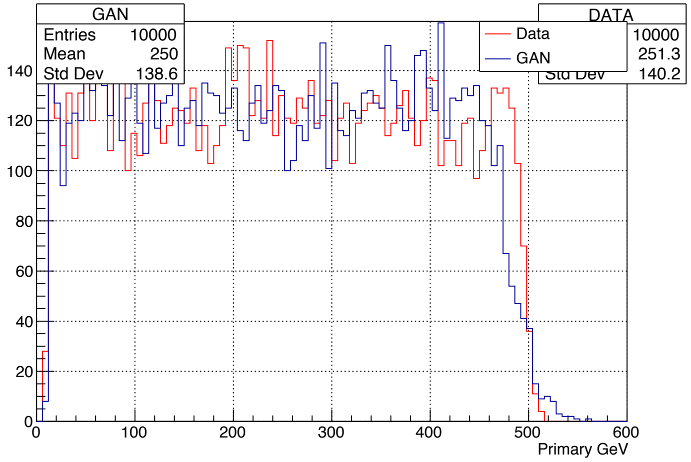
Several pre-processing optimisation steps improved performance at low energy

More details in [G. Khattak talk at IML workshop]

Discriminator regression on input energy

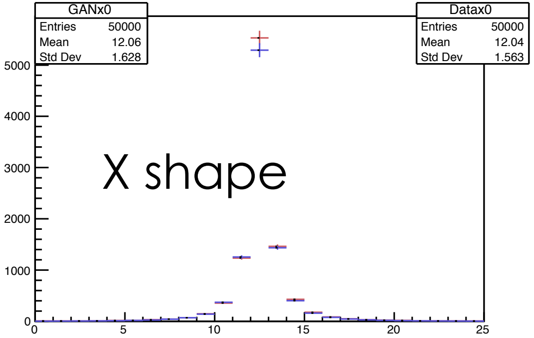


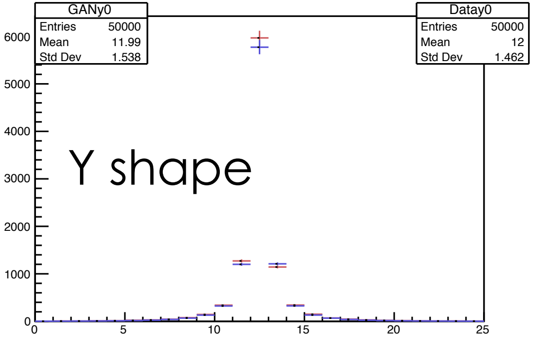
5% error on auxiliary energy regression

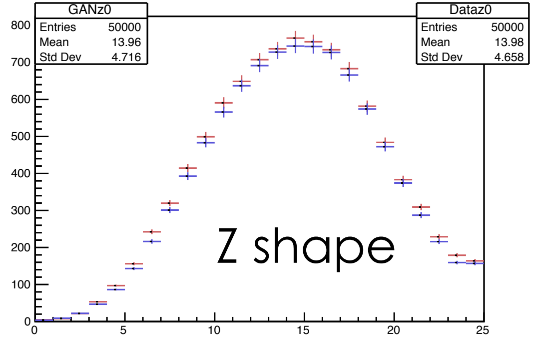


Pions

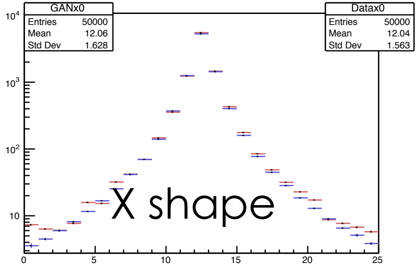
10-500 GeV

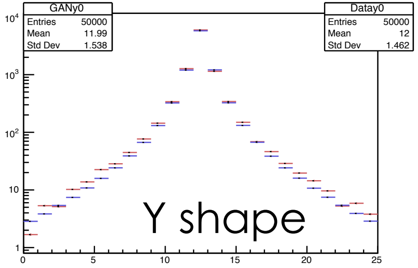


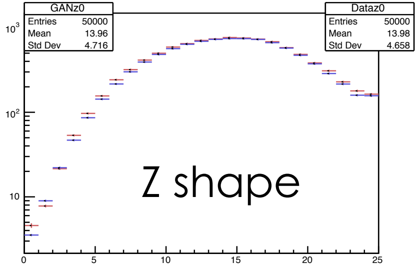


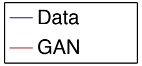


Log scale



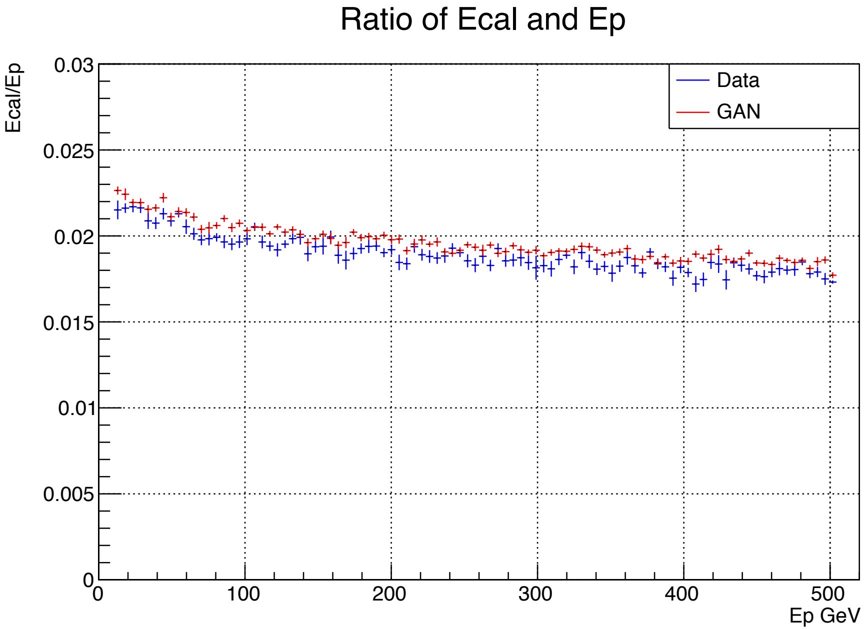


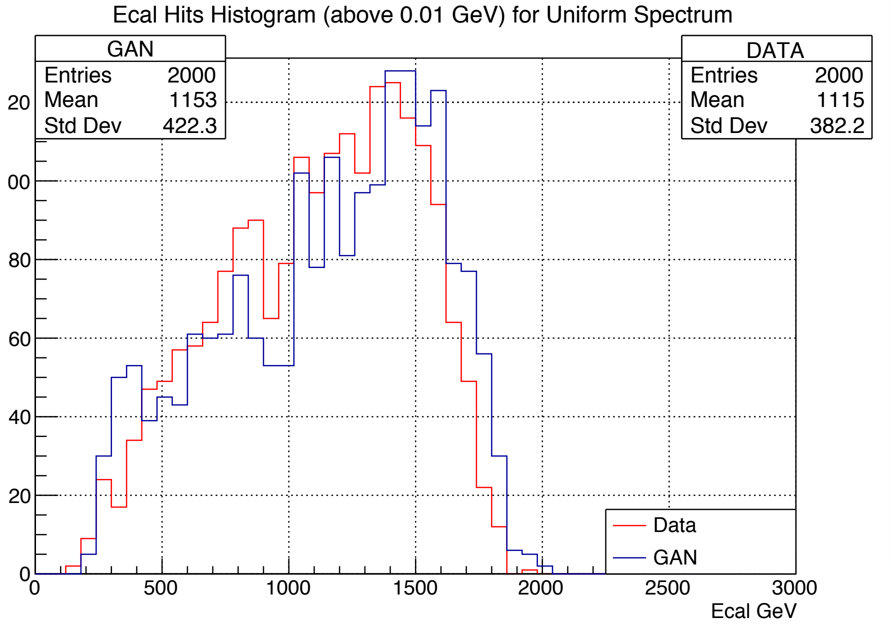




Deposited energy

10-500 GeV -Pions





GAN seems to overestimate slightly energy deposits

Computing resources

- Inference: using a trained model is very fast
  - Orders of magnitude faster than detailed simulation (👍)
  - Next step: test inference on FPGA and integrated accelerators

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**
3d GAN (batchsize 128)	GeForce GTX 1080	0.04
3d GAN (batchsize 128)	Intel i7 @2.8GHz (MacBookPro)	66

- Training time (30 epochs, 200k particles)
  - 1d on an NVIDIA GTX-1080
  - ~30 days on Intel Xeon 8180 *

Time to train for 30 epochs		
Method	Machine	Training time (days)
3d GAN (batchsize 128)	Intel Xeon Platinum 8180 (Intel optimised TF)	30*
3d GAN (batchsize 128)	GeForce GTX 1080	1

*TF1.4 (compiled for AVX2) + missing 3D convolution optimisation in Intel MKL-DNN

2018 Plan

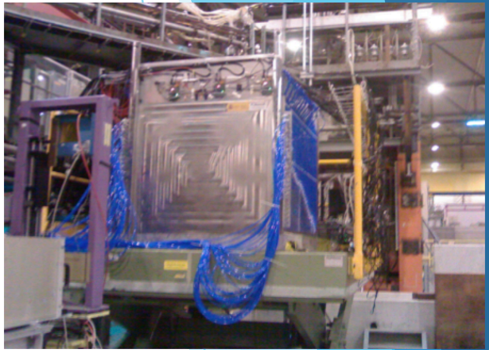
Some work on validation is still ongoing at very low energy

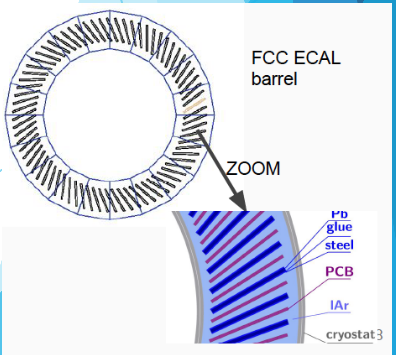
Focus on generalisation and computing resources optimisation

Generalisation

- Our baseline is an example of next generation highly granular detector
- Extend to other similar calorimeters
- FCC LAr calorimeter
- CALICE SDHCAL (testbeam data available!)
- Explore optimal network topology according to the problem to solve
  - Hyper-parameters tuning and meta-optimization
    - Sklearn/skopt, Spearmint, …
    - Test genetic approach

SDHCAL prototype during SPS test beam

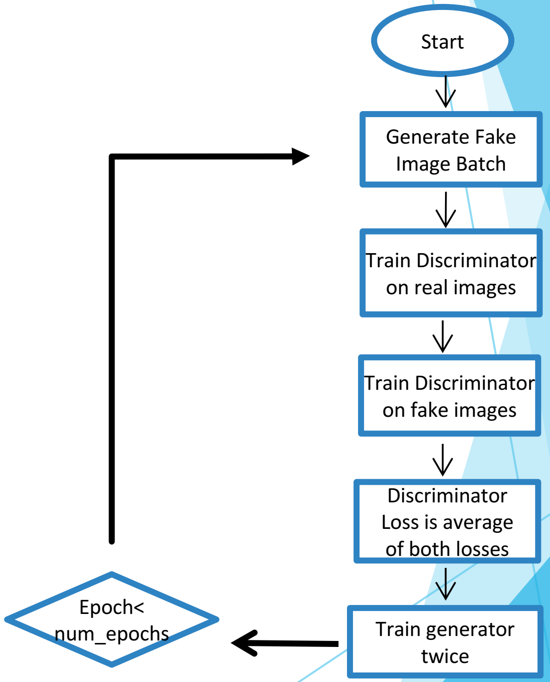




Parallel Training

- Implement data parallelism and study scaling on clusters
- Test data parallelism
  - multiple tasks train the same model on different mini-batches of data, updating shared parameters hosted in one or more nodes
- Tested both Synchronous & Asyncronous training
  - **Asynchronous training**: each replica has an independent training loop that executes without coordination.
  - **Synchronous training**: all of the replicas read the same values for the current parameters, compute gradients in parallel, then apply them together.

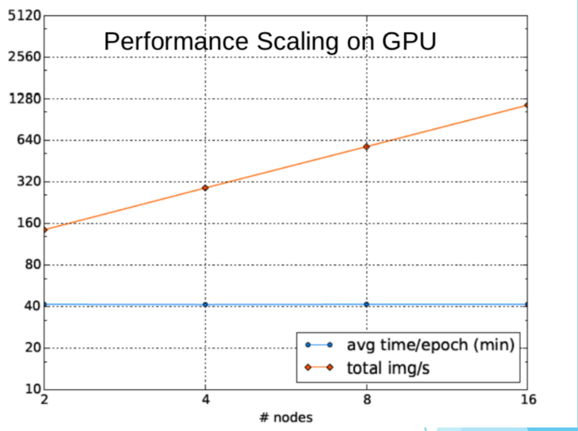
GAN training is a multi-step process

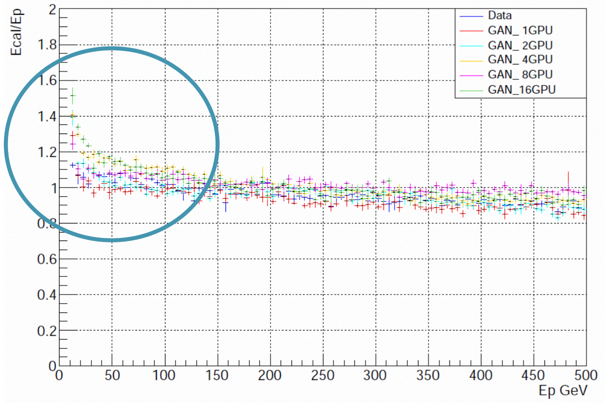


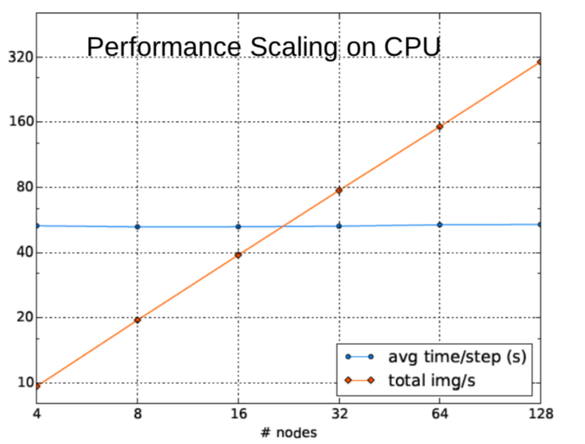
Collaboration with D. Moise , Cray inc.
Submitted to SuperComputing 2018

Synchronous approach

- Cray ML plugin to scale training across multiple GPU and CPU nodes
- Optimal scaling through a large number of nodes
- Observed performance degradation at low energy
- Increase in "effective" batch size?
- Possibly compensate by increasing learning rate..
- Work in progress…





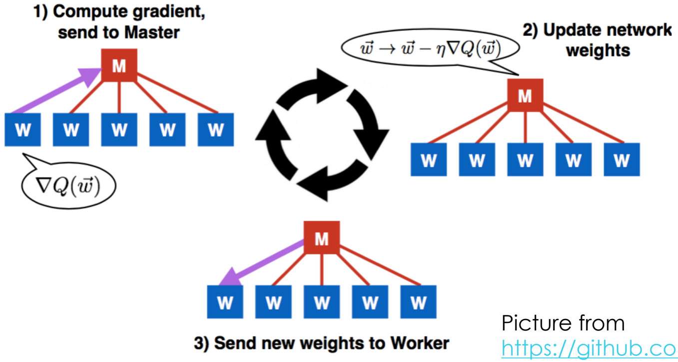


	GPU System	CPU System
Model	XC40/XC50	XC50
Computer nodes	Intel Xeon E5-2697 v4 @ 2.3GHz (18 cores, 64GB RAM) and NVIDIA Tesla P100 16GB	Two Intel Xeon Platinum 8160 @ 2.1GHz (2 x 24 cores, 192GB RAM)
Interconnect	Aries, Dragonfly network topology	Aries, Dragonfly network topology
Step	Epoch	Batch

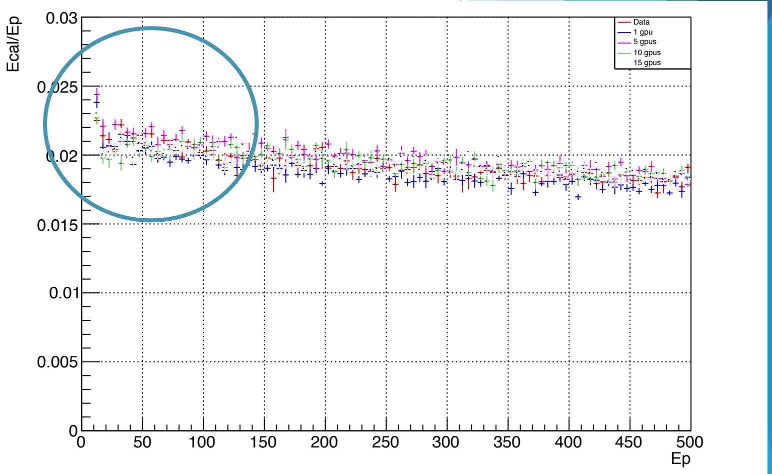
Collaboration with J.R, Vlimant, Caltech.
Submitted to International SuperComputing 2018

Asynchronous approach

- Modify mpi-learn library ([https://github.com/duanders/mpi_learn)](https://github.com/duanders/mpi_learn)
- Elastic SGD
- Test on 20 GPU (Nvidia P100) at CSCS
- Good scaling
- No performance degradation at low energy!
- Work in progress…



Picture from https://github.com/duanders/mpi_learn



Summary

- Generative models seem natural candidates for fast simulation
  - Rely on the possibility to interpret “events” as “images”
  - Many studies ongoing in the different experiments: very promising results!
- 3d GAN is the initial step of a wider plan for an integrated configurable tool
  - First prototype achieves remarkable agreement with G4 simulation

Plan

- Prove we can generalise this network to other calorimeters
- Integration in HEP frameworks
- Extend research to different NN architectures and go beyond detector response simulation
- Computing performance optimisation
  - Efficient training is a priority
  - Different environments: cloud, HPC
  - Big Data approach integration

Thanks !

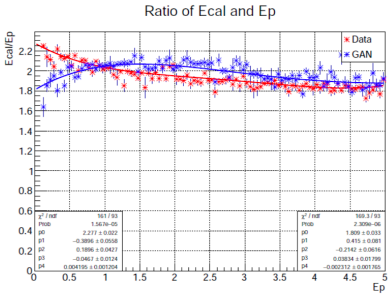
Questions?

Computing resources

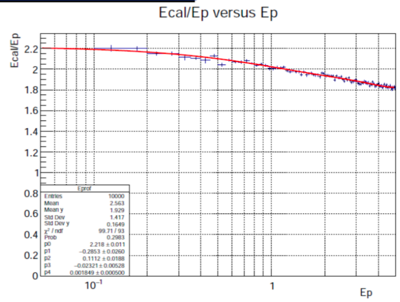
- All tests run with Intel optimised Tensorflow 1.4.1. + keras 2.1.2
  - Compiled TF sources (-O3 –march=broadwell –config=mkl) (AVX2)*
  - TF linked to MKL-DNN
- Use NCHW data format
- OpenMP setup (for Skylake)
  - KMP_BLOCKTIME = 1
  - KMP_HW_SUBSET=1T
  - OMP_NUM_THREADS=28 (physical cores)
  - KMP_AFFINITY=balanced
- Systems:
  - Intel Xeon Platinum 8180 @2.50 GHz (28 physical cores)
  - NVIDIA GeForce GTX 1080

* Currently AVX512 TF build is broken

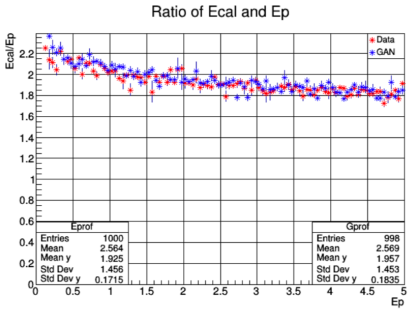
Sampling Fraction (Ep = GeV/100)



Constant Factor of 50

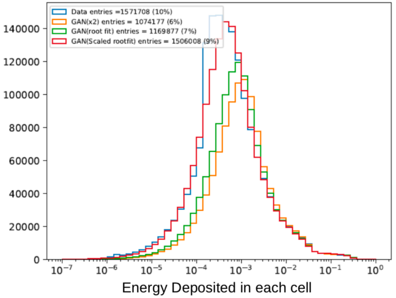


Fitting the Data to fourth order polynomial



Histogram of energies deposited in cells for 10 to 500 GeV

- Geant4 Data
- GAN
  - ECAL sum = Fixed Factor x Ep
  - 4th order polynomial fit for ECAL sum
  - Cell energies scaled by 100



More details in G. Khattak talk at IML workshop