A Deep Learning tool for fast detector simulation

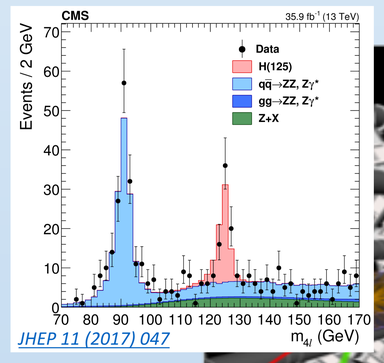
Federico Carminati[2], Gulrukh Khattak[2], **Sofia Vallecorsa[1]**, Jean-Roch Vlimant[3]

[1]Gangneung Wonju U. & CERN, [2]CERN, [3]Caltech
Sofia.Vallecorsa@cern.ch

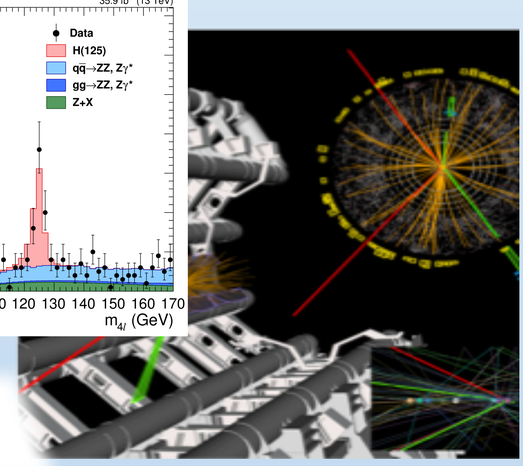
The project

Motivation

Simulation of particle transport through matter is fundamental for understanding the physics of **High Energy Physics** (HEP) experiments, as the ones at the Large Hadron Collider (**LHC**) at CERN. Currently, most of **LHC** worldwide distributed **CPU budget** – in the range of half a million CPU-years equivalent – is dedicated to **simulation.** A faster approach is to treat Monte Carlo **simulation** as a black-box and replace it by a **deep learning** algorithm trained on different particle quantities. Our project intends to test several DL approaches to achieve a speedup of at least x100 with respect to Monte Carlo techniques.

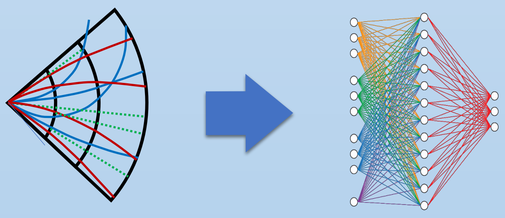


JHEP 11 (2017) 047



200 Computing centers in 20 countries: > 600k cores
@CERN (20% WLCG): 65k processor cores ; 30PB

Generative Adversarial Networks for fast simulation

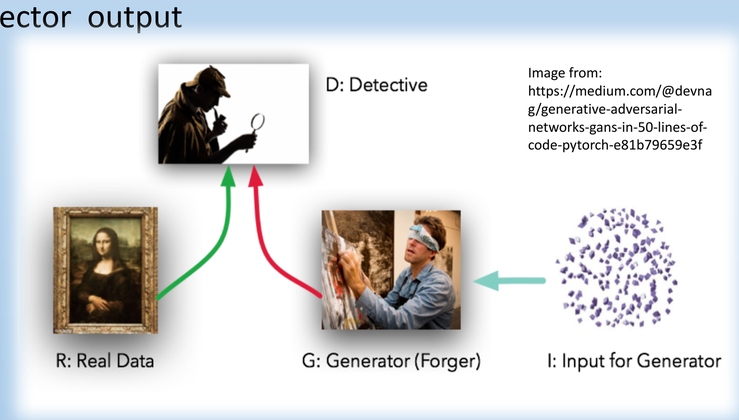


Generative models, such as **Generative Adversarial Networks (GAN**) are particularly suited to replace Monte Carlo: they generate **realistic** samples modelling complicated probability distributions. They allow **multi-modal output,** they can do **interpolation** and they are robust against **missing data**.

We can use Monte Carlo simulation to train GANs to reproduce realistic detector output

Generative Adversarial Networks simultaneously train two models: a **Generator G** and a **Discriminator D**

- **G reproduces the data** distribution starting from **random** noise
- **D estimates the probability** that a sample came from the training data rather than G
- Training procedure for G is to **maximize the probability of D making a mistake**



Project timeline

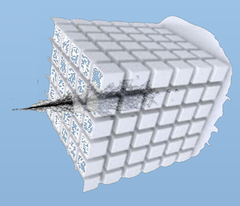
2017		2018 Q1-Q2	2018 Q3-Q4
Proof of concept	**Performance optimisation**	**Computing resources**	**Generalization**
Can we interpret detector output as images?	Can we achieve Monte Carlo precision? Can we sustain the increase in future detector complexity?	Distributed training on Cloud/HPC	Can we generalize architecture to a larger class of detectors? Meta-Optimization and hyper-parameter scans

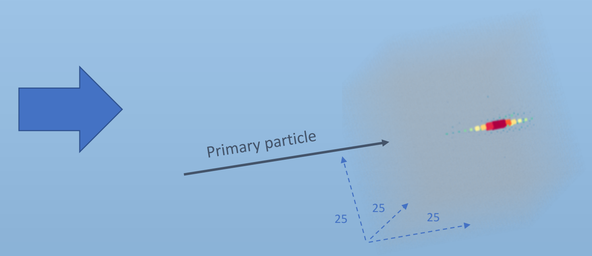
Achievements

One of the first 3D GAN implementations!

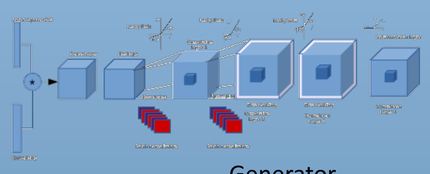
Start with the most time consuming simulations : high granularity calorimeters*. Single particles deposit energy in an array of calorimeter cells and generate a "**energy shower**", interpreted as a 3D image.

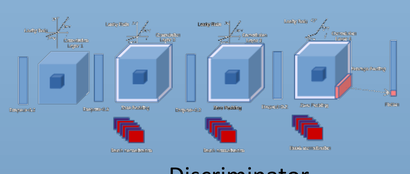
Data is essentially a 3D image





π shower in CLIC calorimeter simulated with Monte Carlo approach

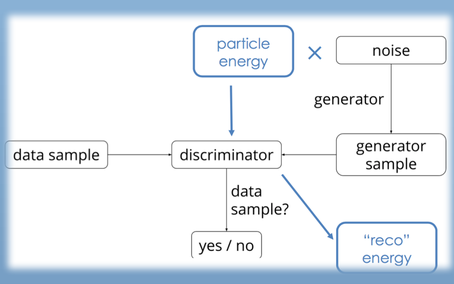

Generator


Discriminator

Generator and Discriminator are based on **3D convolutions**.

The **shower shape** depends on **particle type and energy** so we condition training on particle energy

Perform detailed validation against standard Monte Carlo: agreement is remarkable from the **physics** point of view**!** (More details in our research poster)



Computing resources

All tests run with Intel optimised Tensorflow 1.4.1. + keras 2.1.2

Inference: Using a trained model is very fast !

- **Orders of magnitude faster** than standard Monte Carlo
- Test inference on FPGA and integrated accelerators

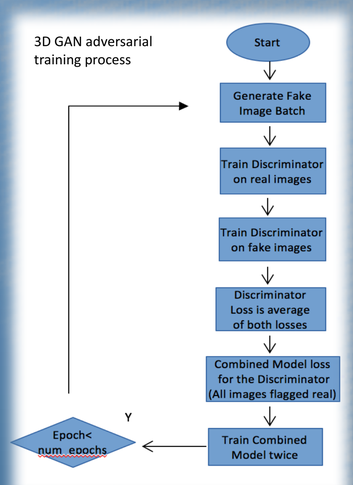
Time to create an electron shower		
Method	Machine	Time/Shower (msec)
Full Simulation (geant4)	Intel Xeon Platinum 8180	17000
3d GAN (batch size 128)	**Intel Xeon Platinum 8180**	**7**
3d GAN (batchsize 128)	GeForce GTX 1080	0.04
3d GAN (batchsize 128)	Intel i7 @2.8GHz (MacBookPro)	66

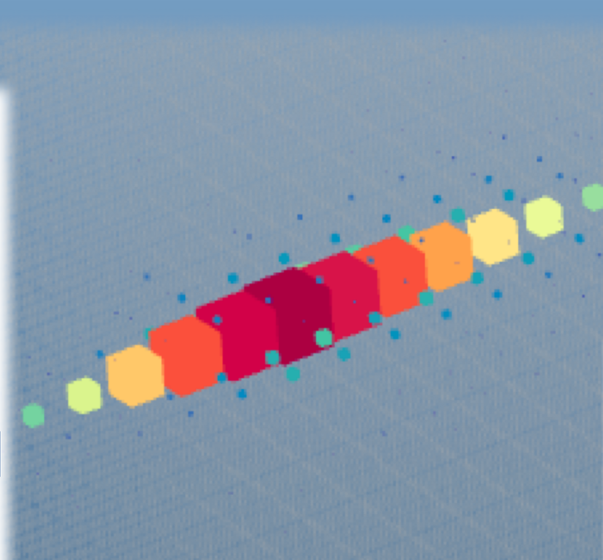
Training: 3D GAN are not "out-of-the-box" networks

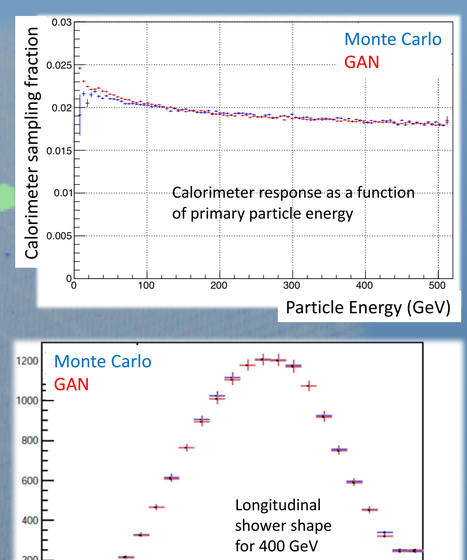
- **Complex training process**

Systems:
Intel Xeon Platinum 8180 @2.50 GHz (28 physical cores)
NVIDIA GeForce GTX 1080

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







Development

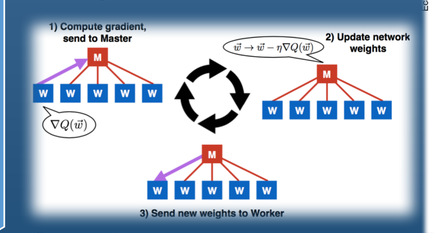
Distributed training for HPC

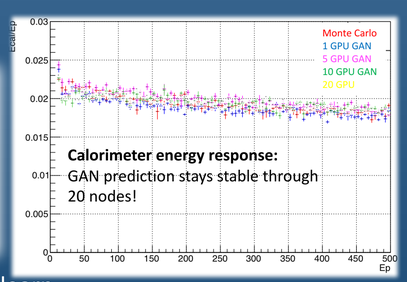
Implement **data parallelism** and study scaling on clusters

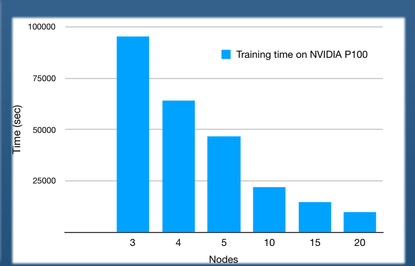
Modify MPI based library (mpi-learn[#]) to parallelize adversarial training process

Preliminary **scaling** measured at CSCS Swiss National Super Computing Center.

Asynchronous SGD

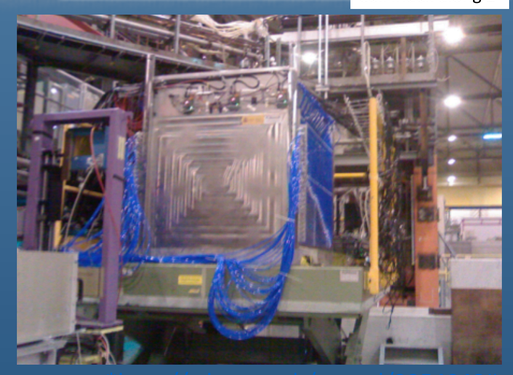






How generic our network can be?

- Our baseline is an example of **next generation calorimeter** detector
- **Extend** to other calorimeters
- Explore optimal network topology according to the problem to solve
  - **Hyper-parameters** scans and **meta-optimization**
  - **Fast training**


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

[#]https://github.com/duanders/mpi_learn

References

- Goodfellow et al. 2014
- Conditional GAN, arXiv: 1411.1744
- Deep Convolutional GAN, arXiv:1511.06434
- Auxiliary Classifier GAN, arXiv:1610.0958

Part of this work was conducted at "*iBanks*", the AI GPU cluster at Caltech.
We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of "*iBanks*"