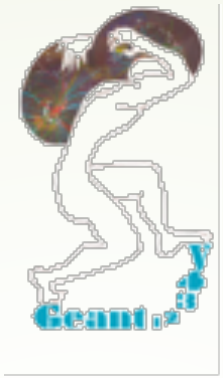


Machine Learning-based fast simulation in GeantV

Sofia Vallecorsa

for the GeantV project

ACAT 2017
21-25 August 2017
University of Washington, Seattle

Outline

- Introduction
- GEANTV framework for fast simulation
- Machine learning approach
- Generative Adversarial Networks for calorimeters
- Summary and plans

Introduction

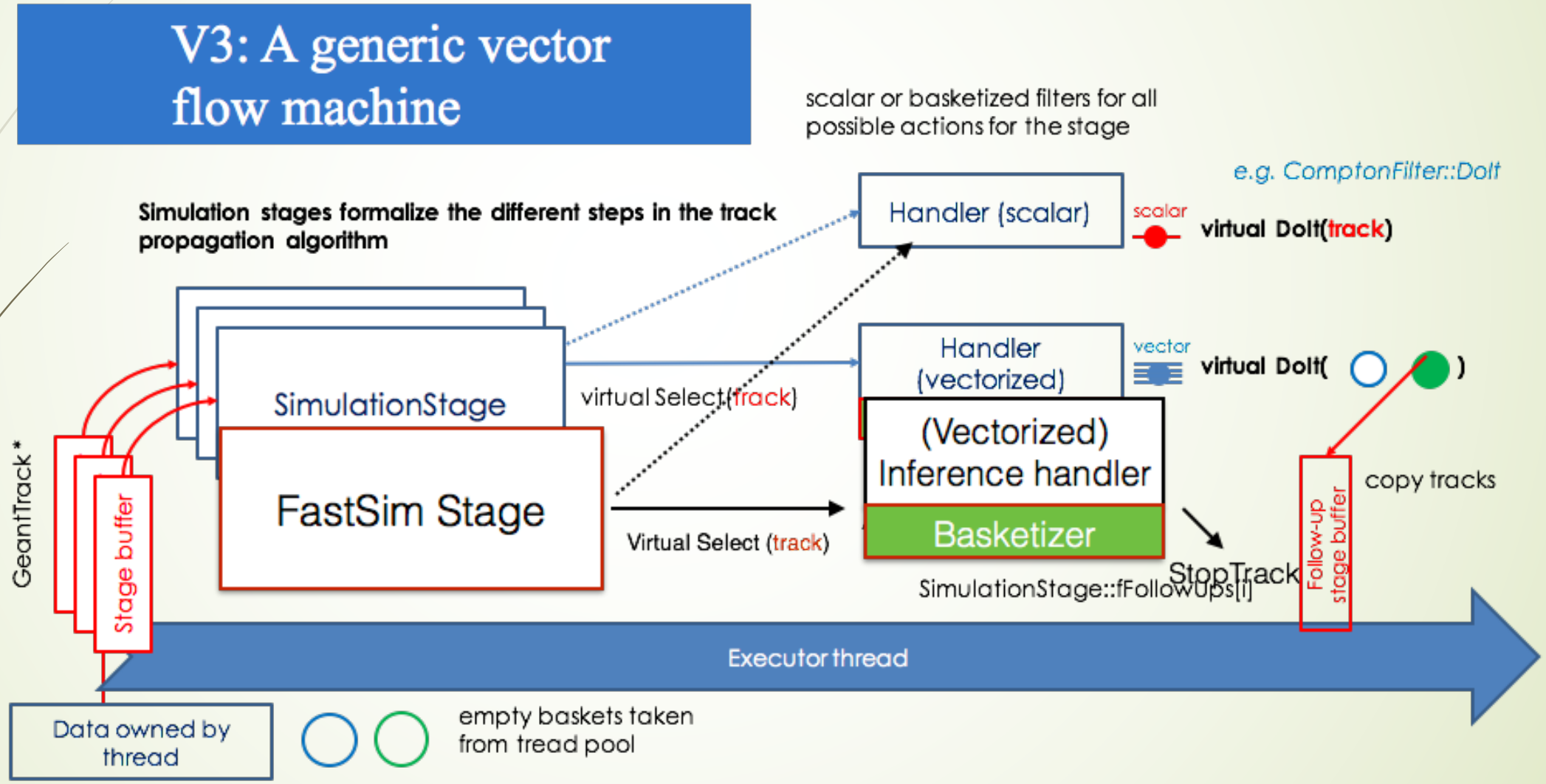
- Detailed simulation has heavy computation requirements
- A large fraction of current computing resources are devoted to Monte Carlo production
- Fast simulation is heavily used by experiments
  - Mostly when accuracy requirements are more "relaxed" (searches, upgrade studies,…)
  - A necessity in future HL-LHC runs
- Currently available solutions are detector dependent

GeantV fast simulation framework

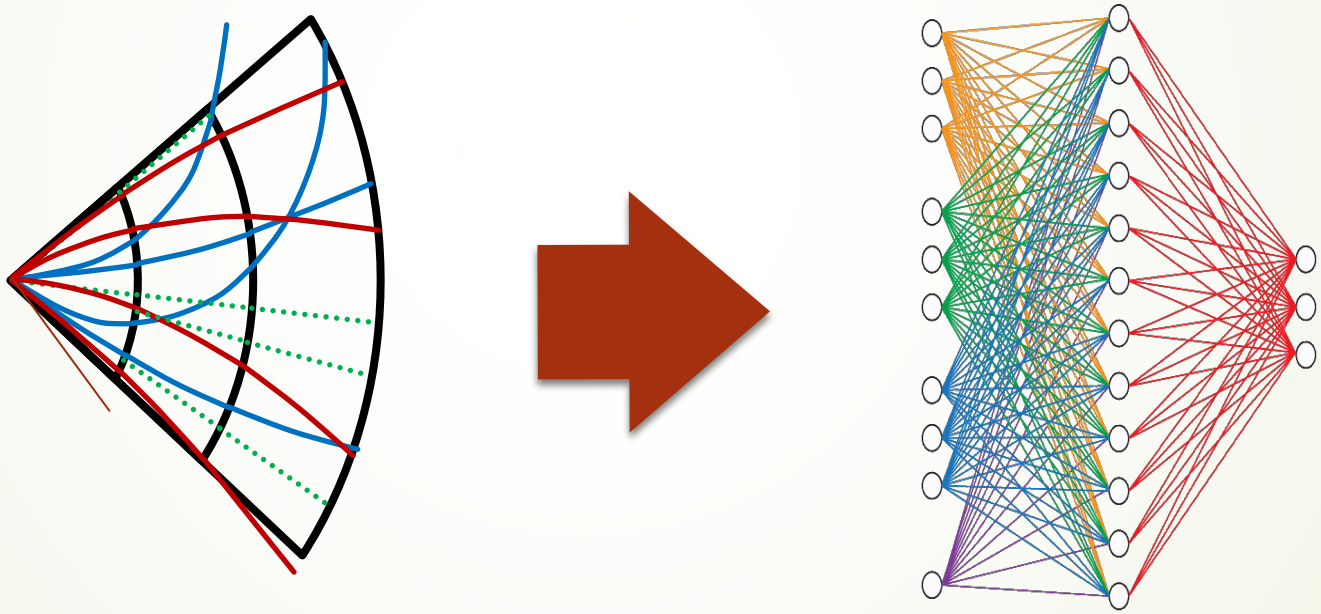
Fast, modular and fully configurable

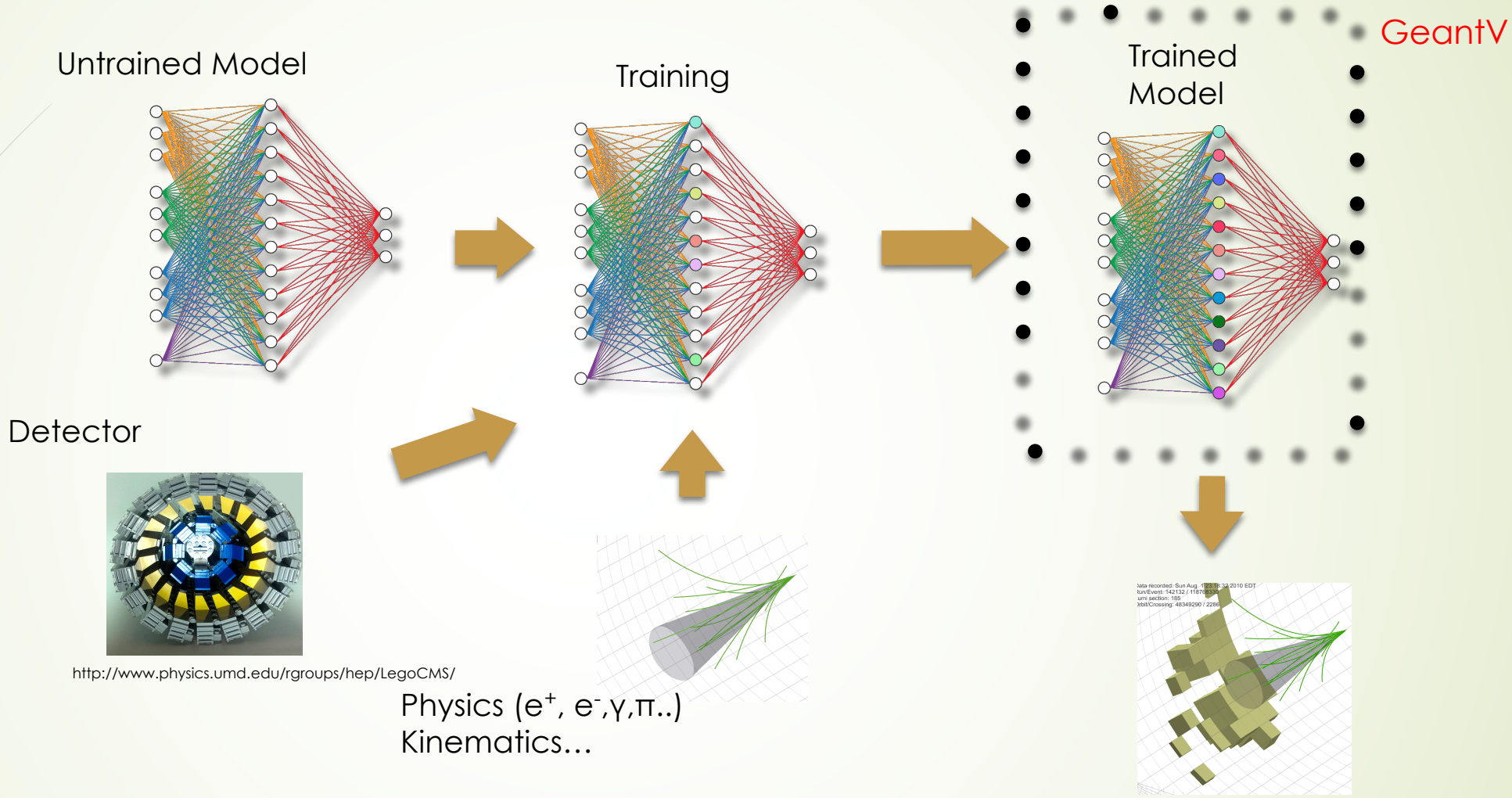
- Introduce fastsim in GeantV framework
  - Provide configurable interface (probably G4 userActions style)
  - Composite solutions (mix/match fast and full simulation within the same event)
  - A choice of basic tools: parametrization, libraries, machine learning tool

Integrating fast simulation in GeantV

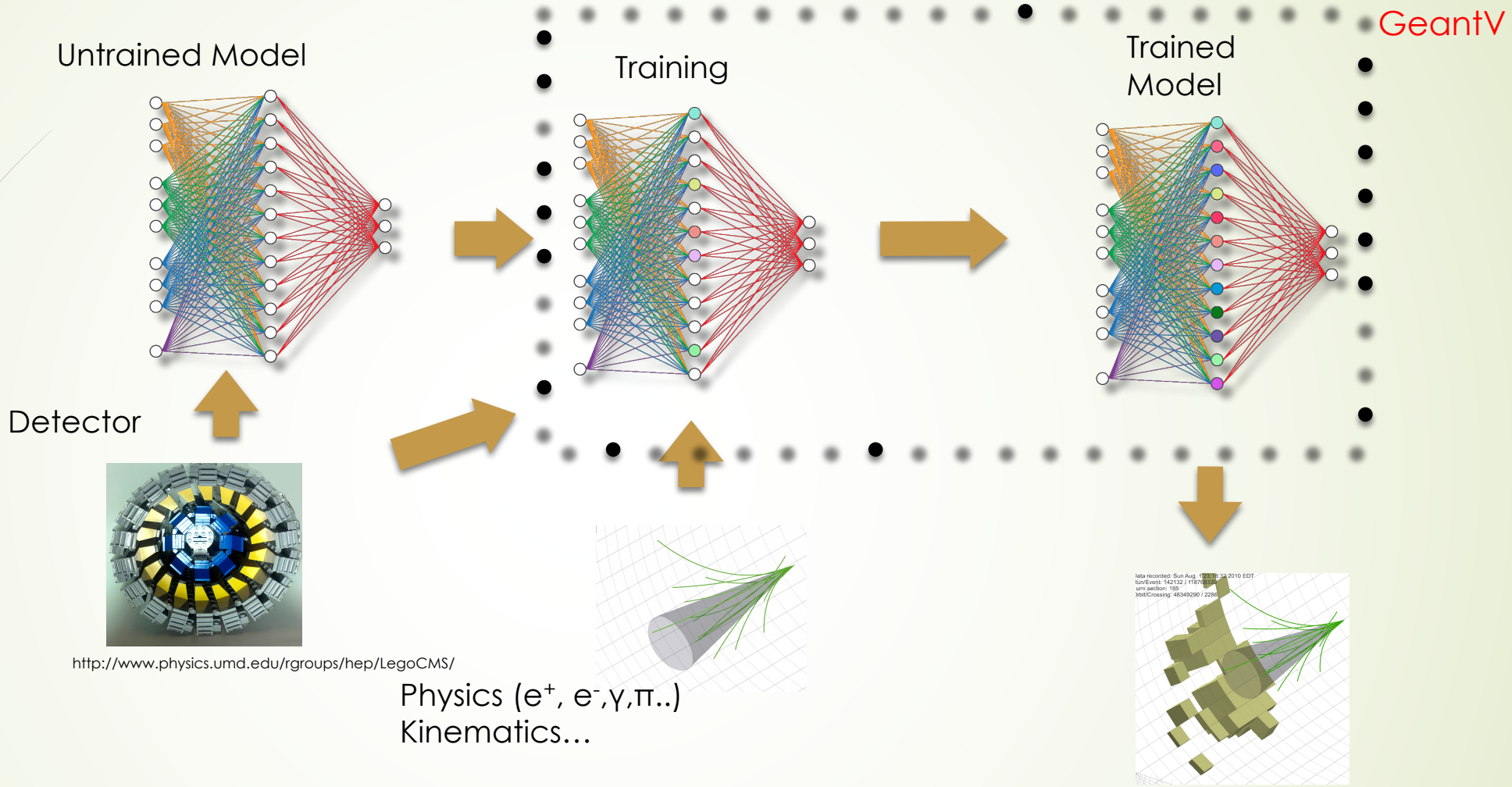


Machine Learning for fast sim





- Convert saved NN for application in C++ environment through libraries as lwtnn



- "Detector" info to drive the design/choice of model
- A tool capable of optimizing the model configuration and train
- Embed in GeantV

Design the ML model(s)

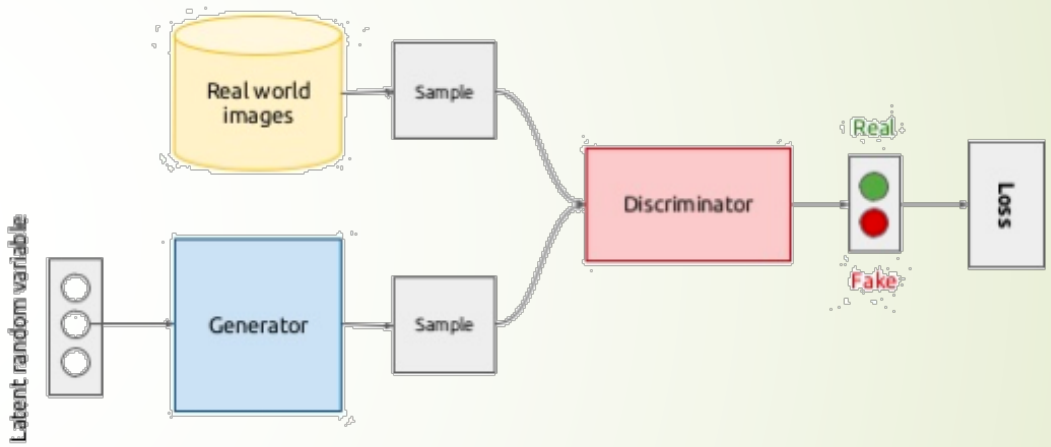
- Initially focus on time consuming detectors
  - Reproduce particle showers in electromagnetic and hadronic calorimeters
- Train networks on full simulation
  - Eventually test possibility of training on real data
- Test different techniques
  - Multi Objective regression, Feature extraction
- Test different models
  - Generative adversarial networks, Recurrent networks

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

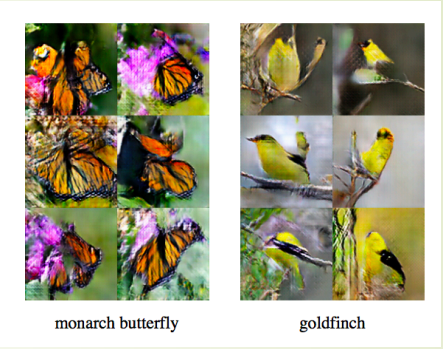


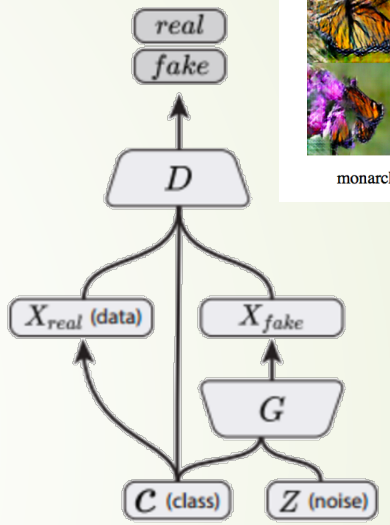
The counterfeiter/police case

- 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

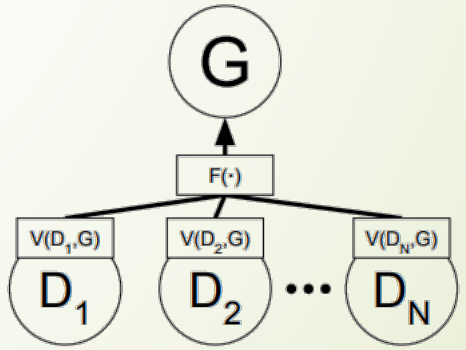
Many GAN flavors

- Original GAN based on MLP in 2014
- Deep Convolutional GAN in 2015
- Conditional GAN
  - Learn a parameterized generator $p_{model}(x \mid \theta)$;
  - Useful to obtain a single generator object for all θ configurations
- Auxiliary Classifer GAN
  - D can assign a class to the image
- Multi-adversarial
  - G is trained on feedback aggregated by multiple D



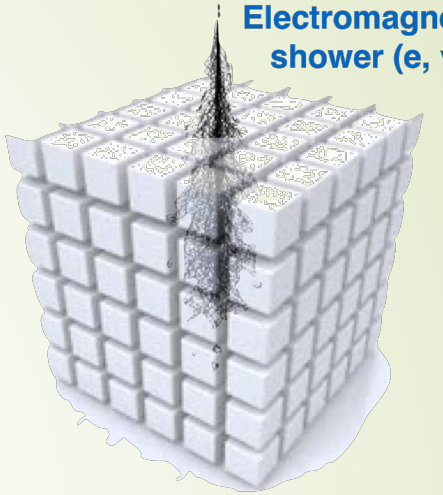


Conditional GAN
(Mirza & Osindero, 2014)

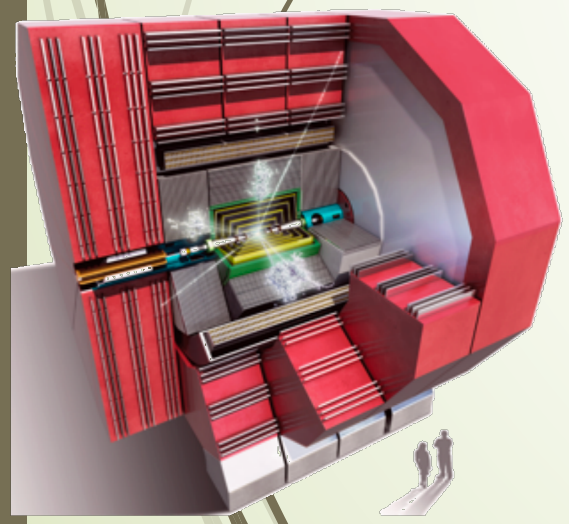


Durugkar et al., ICLR 2017

CLIC calorimeter



- CLIC is a CERN project for a linear accelerator of electrons and positrons to TeV energies
- Associated calorimeter detector design(*)
- An array of absorber material and silicon sensors
  - ECAL (1.5 m inner radius, 5 mm×5 mm segmentation): 25 tungsten absorber layers + silicon sensors
  - HCAL (3.0 cm×3.0 cm segmentation): 60 steel absorber layers + polystyrene scintillators

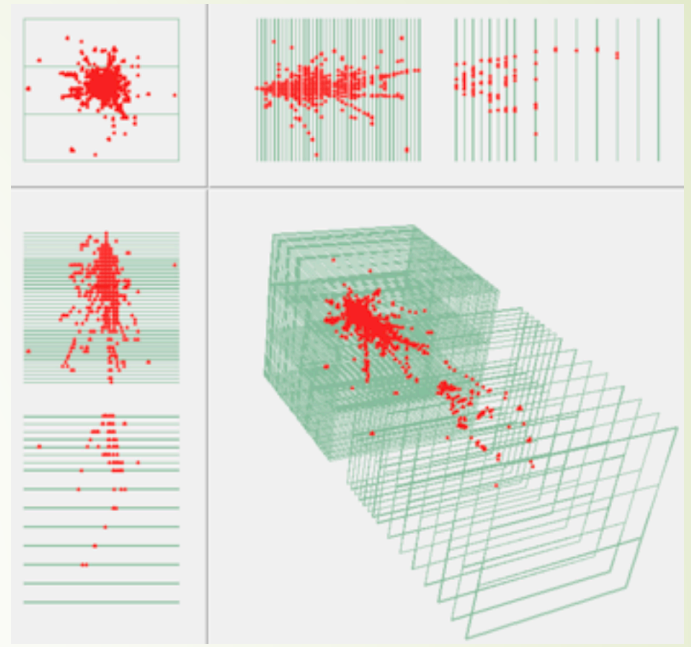


Data released within CERN OpenData initiative

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

CLIC calorimeter data

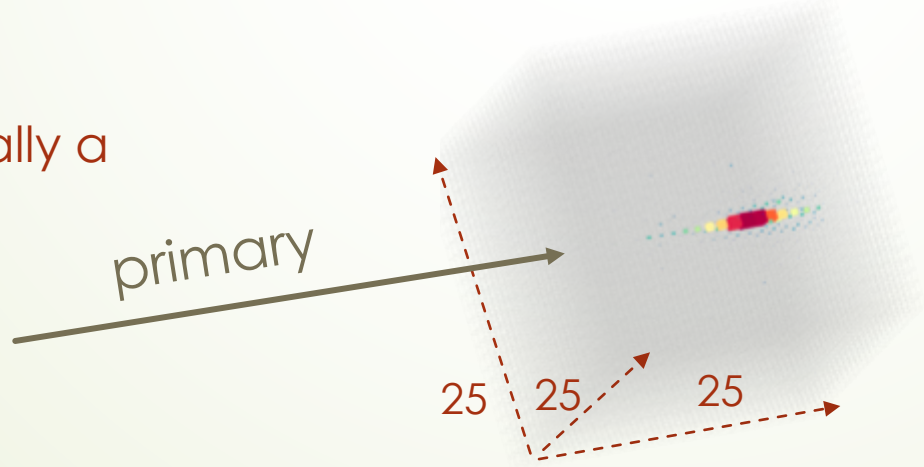
Geant4 shower

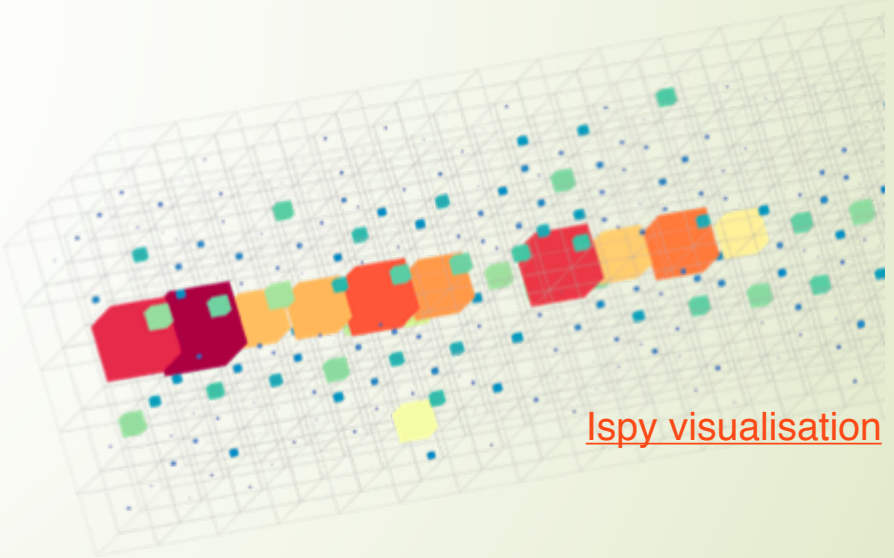


Geant4 single-particle benchmark datasets (e^+, e^-, γ, π) (DD4hep/DDG4/ddsim)

- Uniform energy distribution
- Fixed energy points (10,50,100,200 GeV)
- Simplified: no clustering/cluster id algorithms applied

Data is essentially a 3D image

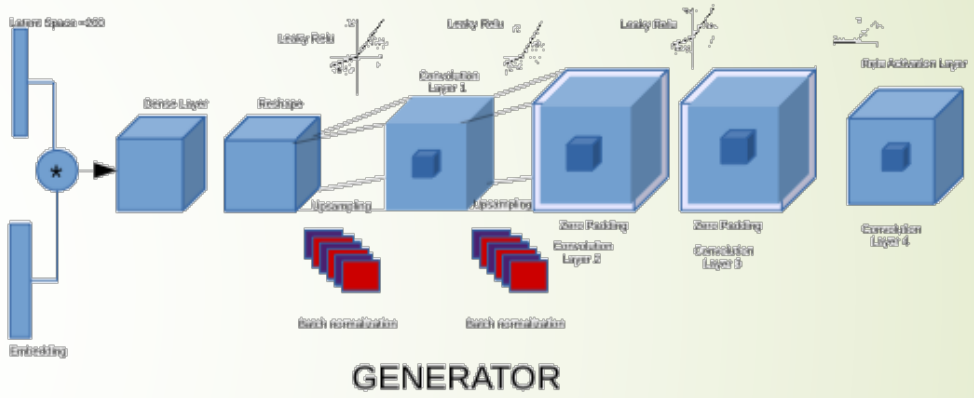


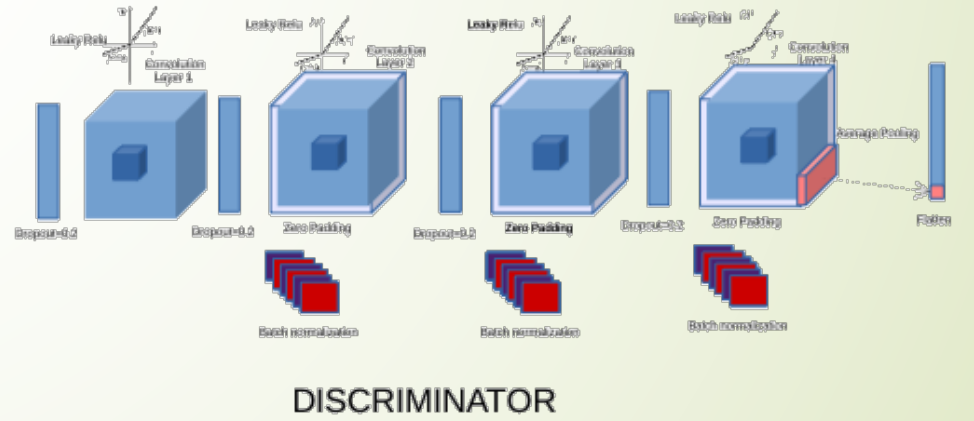


Ispy visualisation

3d GAN for calorimeter images

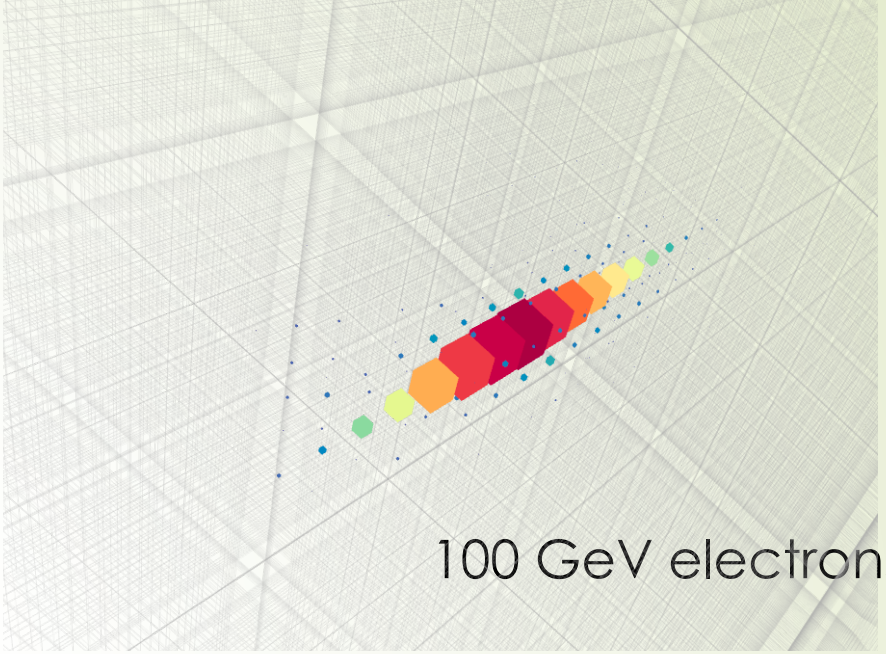
- Conditional GAN (particle type)
- Based on convolution/deconvolutions
  - 3D (de)convolutions to describe full shower development
- Implemented tips&tricks found in literature
  - Some helpful (no batch normalisation in the last step, LeakyRelu, no hidden dense layers, no pooling layers)
  - Some not (Adam optimiser)
- Batch training
- Loss is combined cross entropy



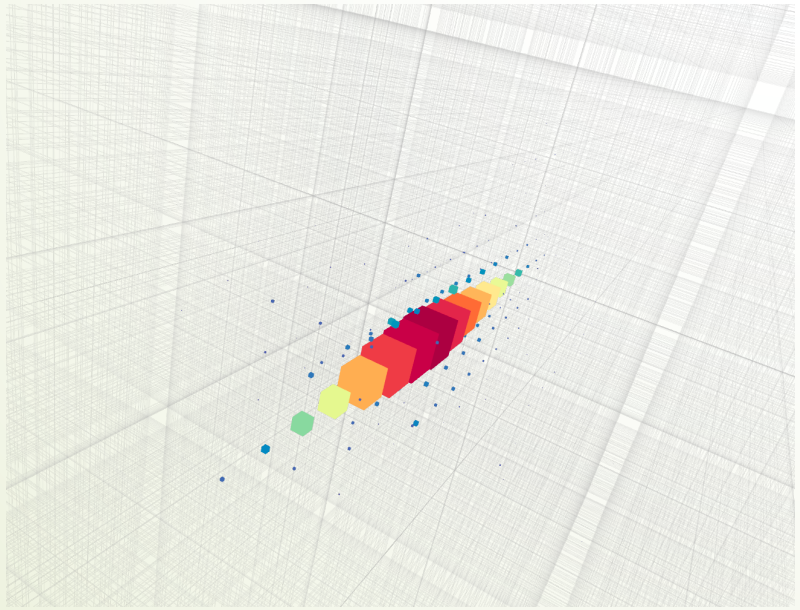


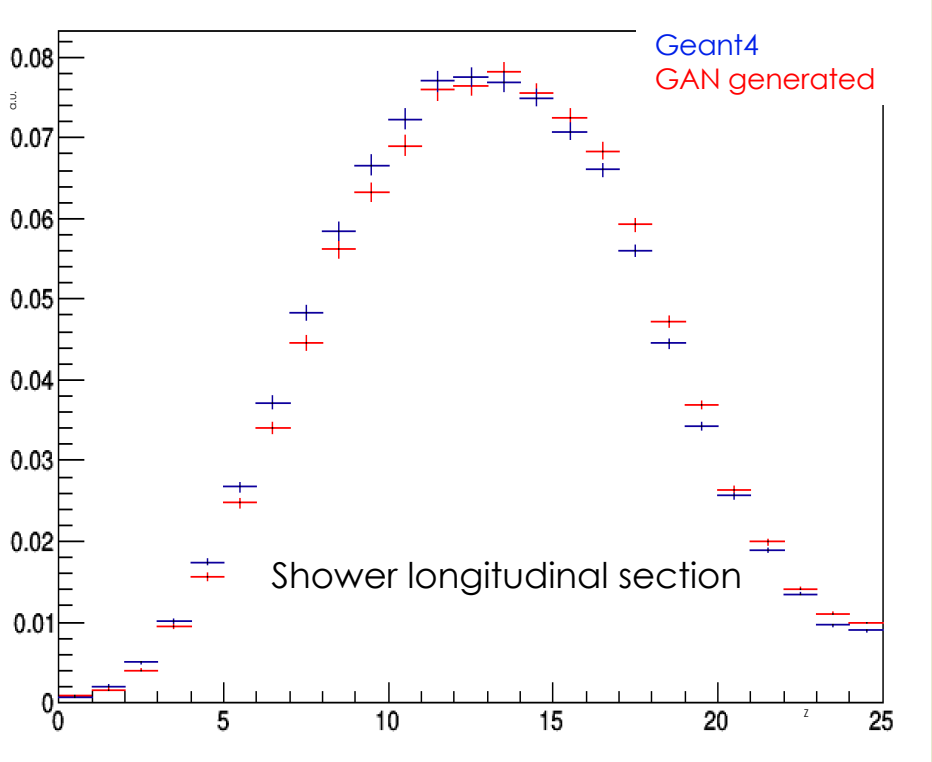
First 3d shower images

- Train networks on 150k electrons/photons
- Keep energy fixed at 100 GeV
- Qualitative analysis show no collapse problem
- Longitudinal shower shape slightly off



100 GeV electrons

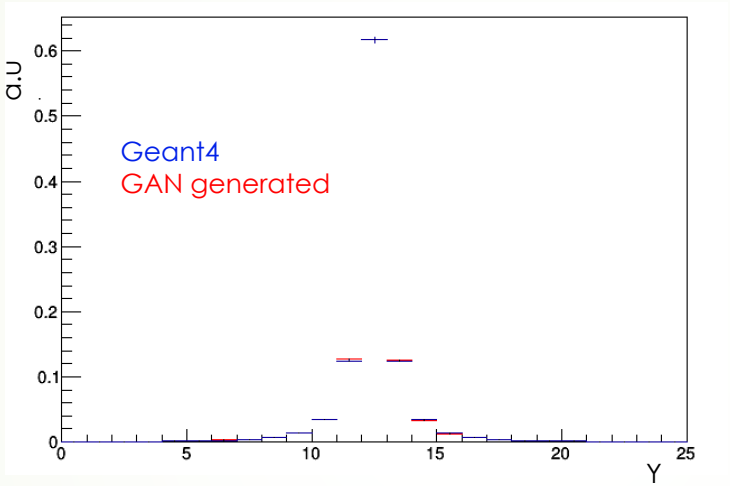


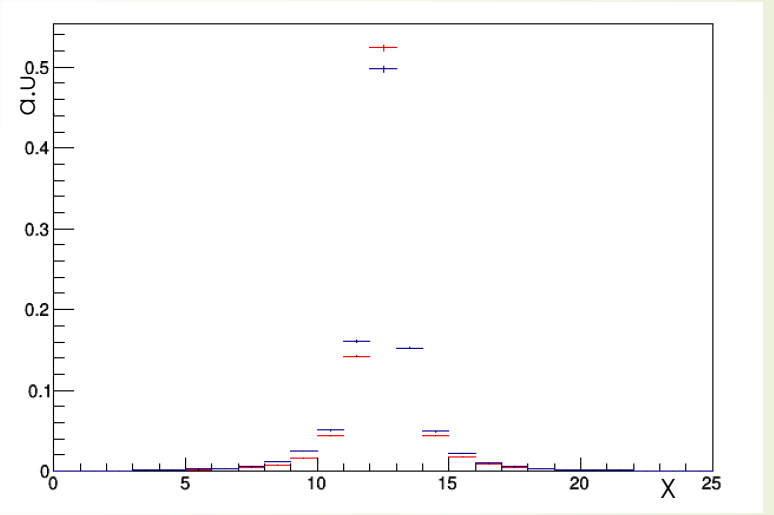


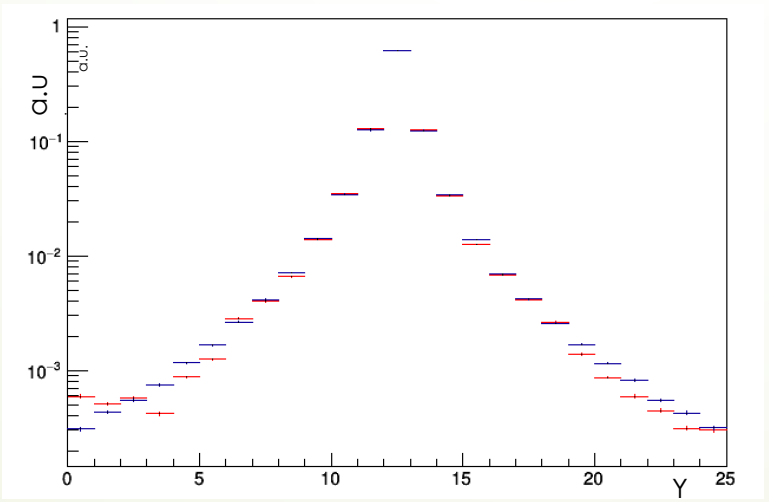
GAN generated showers

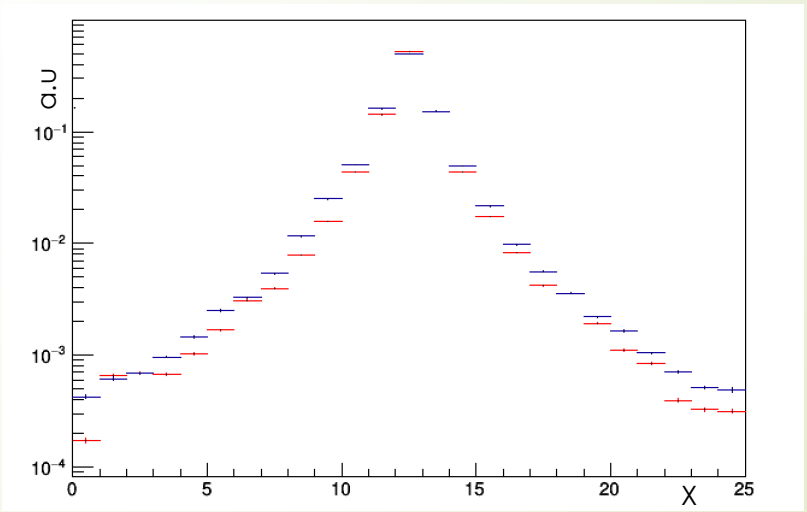
Shower transverse sections

100 GeV electrons



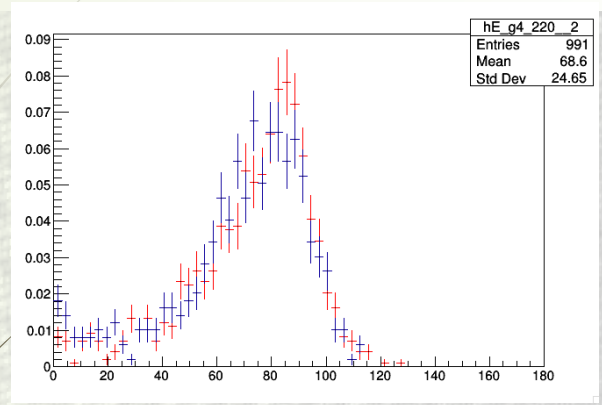


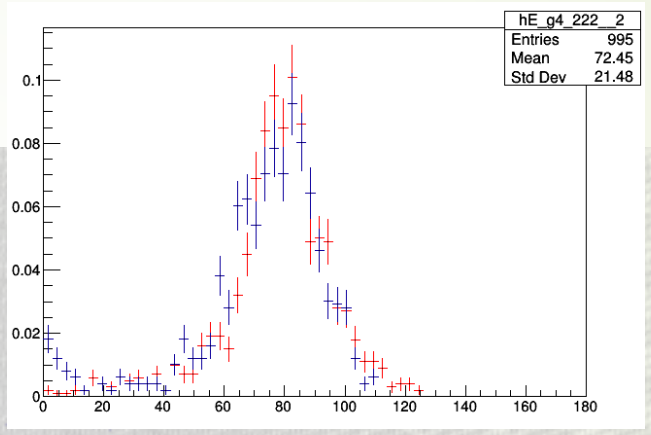


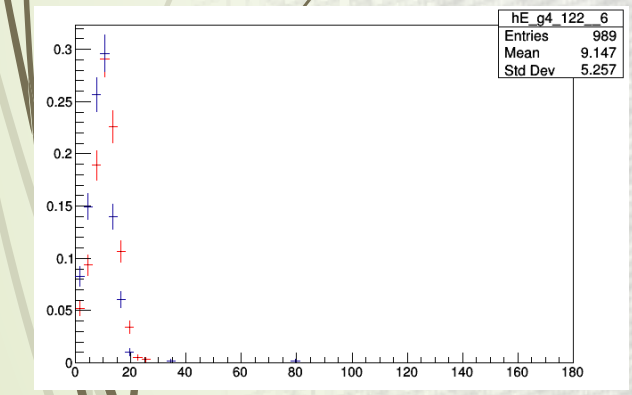


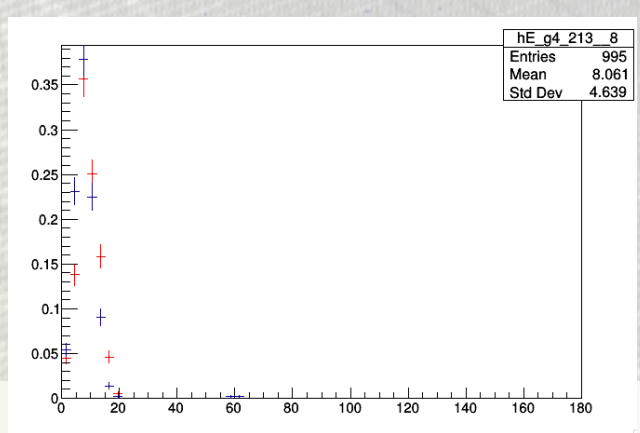
Single cell response

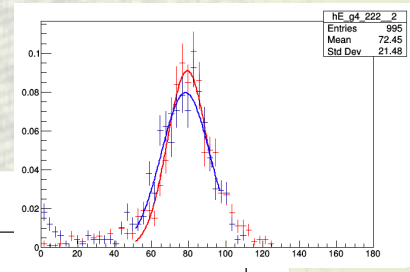
Preliminary

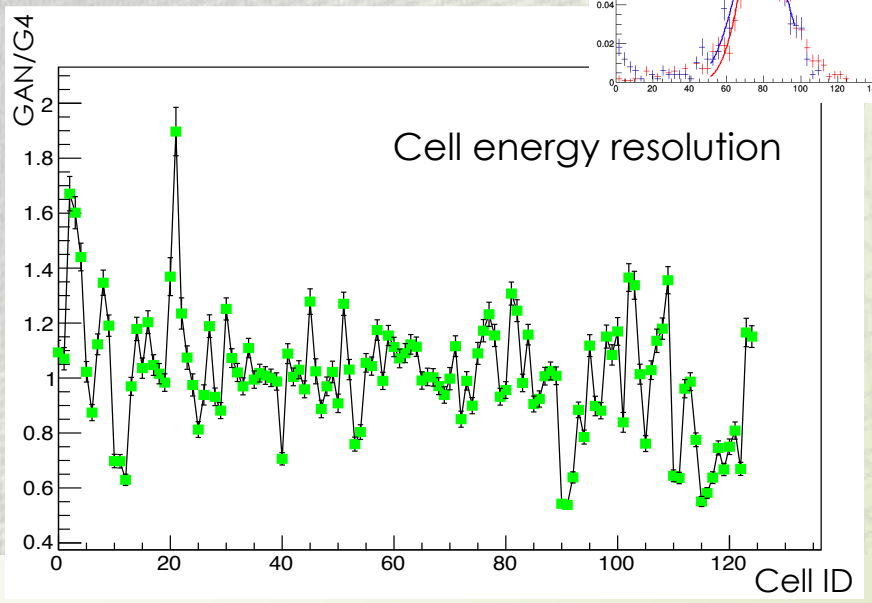








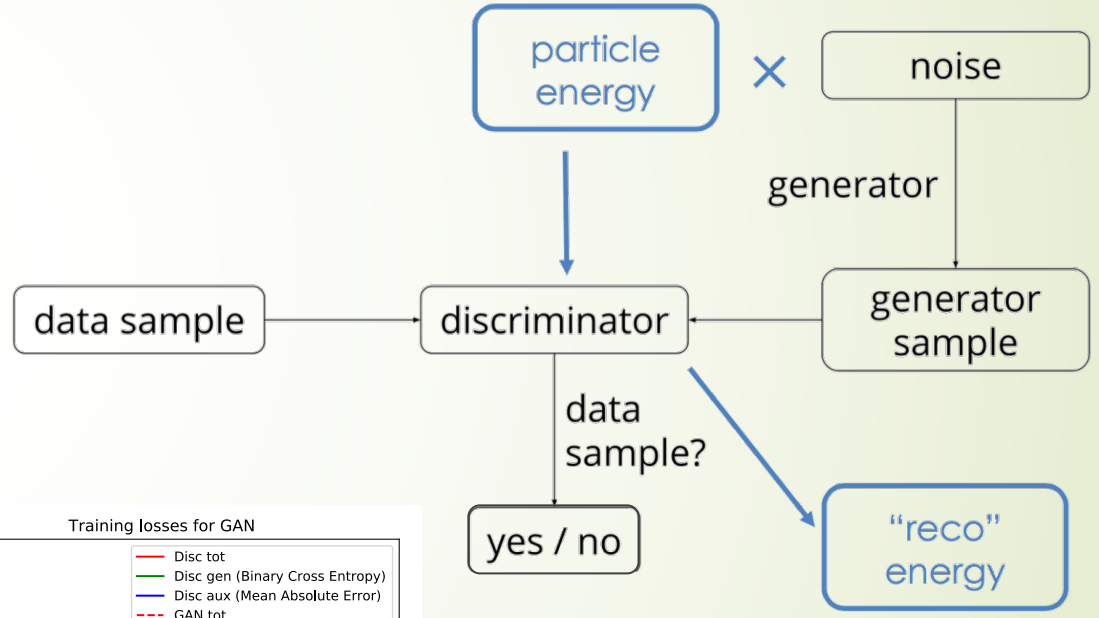


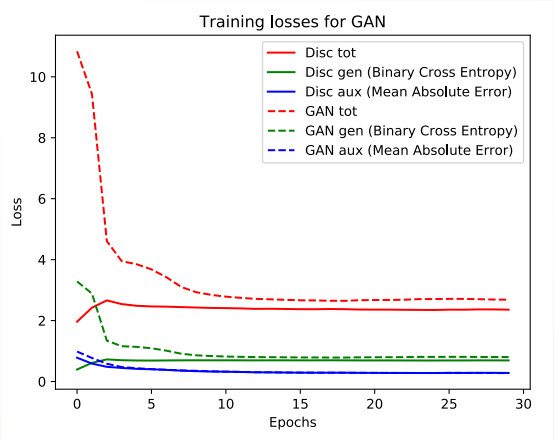


Energy as auxiliary classifier

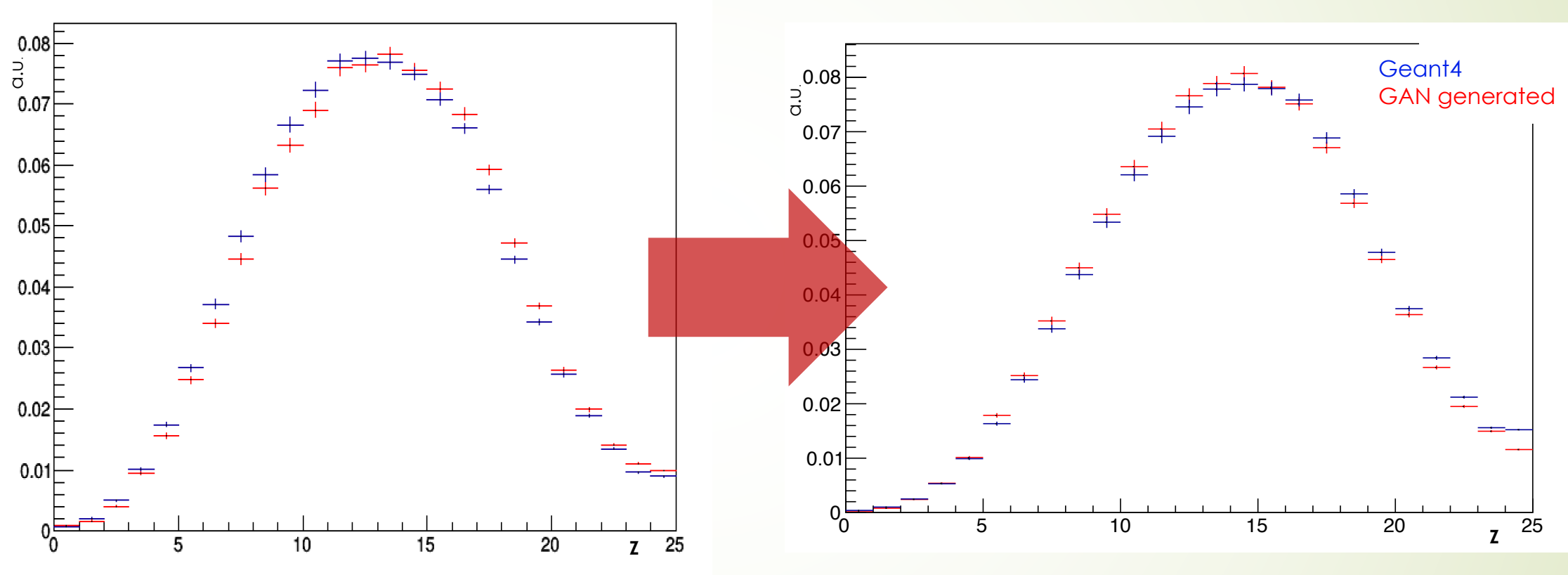
Training the generator and the discriminator using primary particle energy

- Add primary energy regression task to discriminator
- Auxiliary loss weight is scaled down wrt shower's
- Train on large continuous spectrum: [0,500] GeV
- Single particle type: electrons



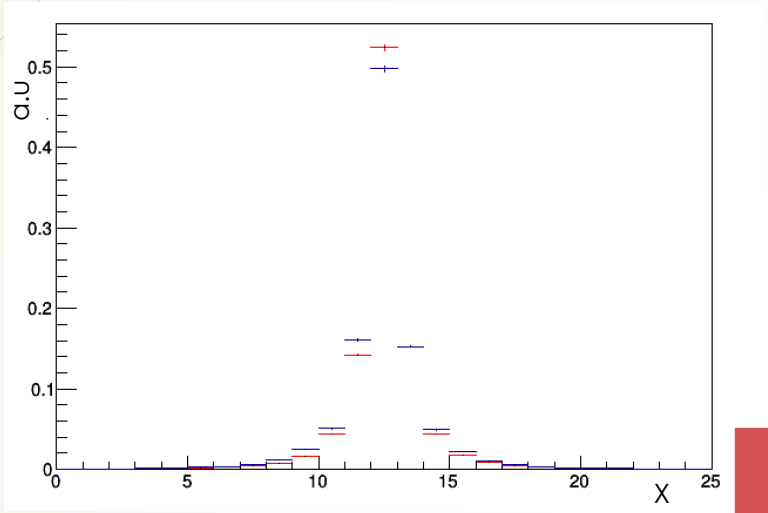


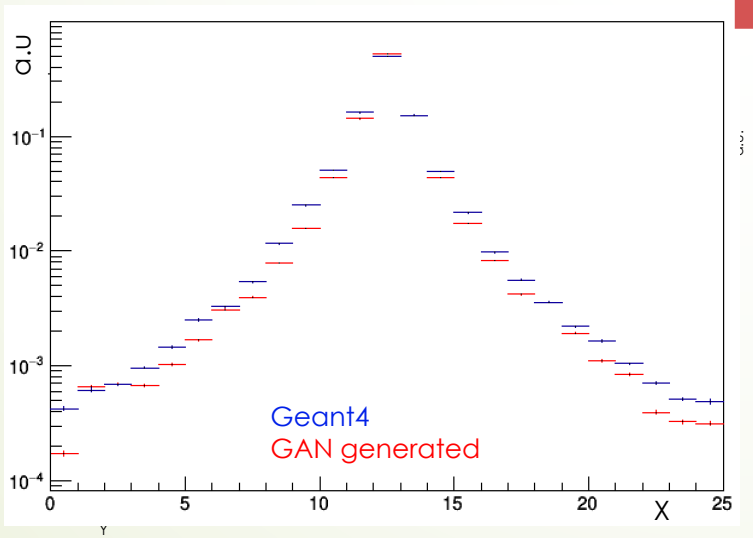
Generated showers: longitudinal profile

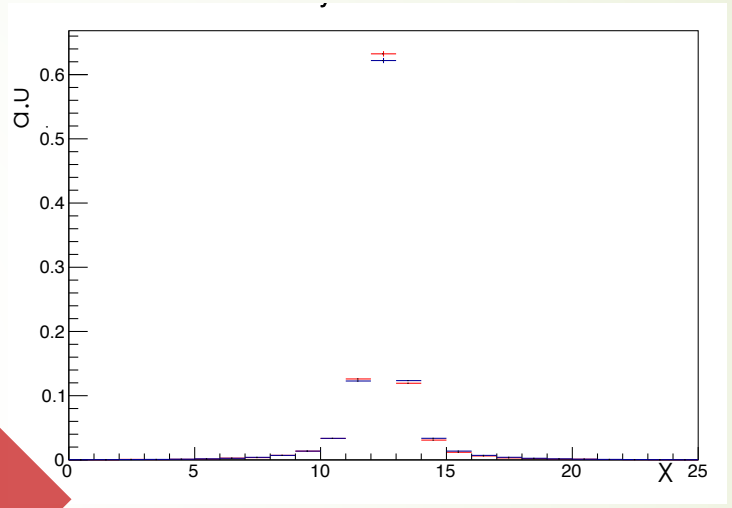


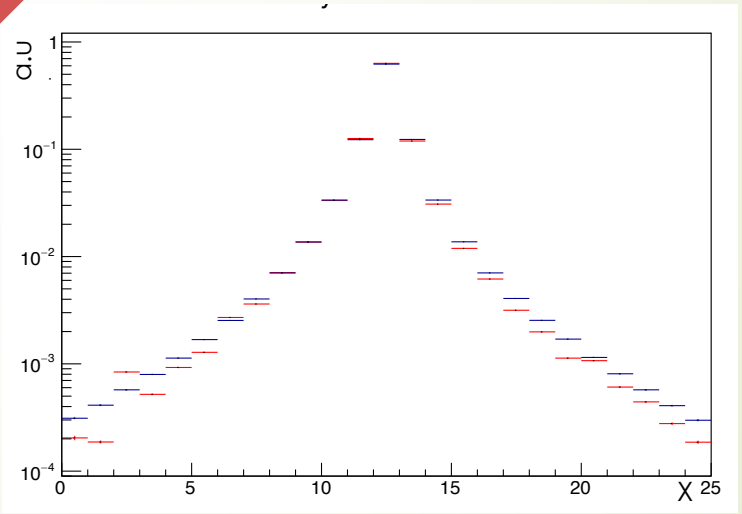
- Shower longitudinal profile shifts towards G4
- Agreement improves

Generated showers: transverse profile



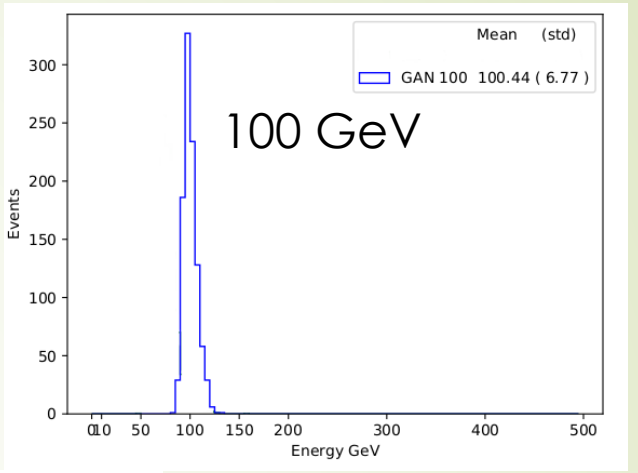


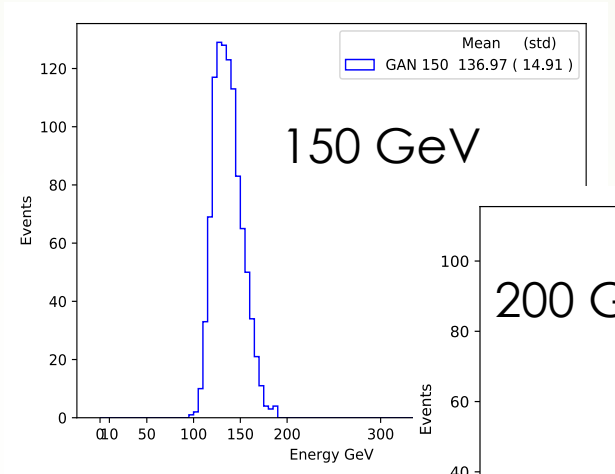


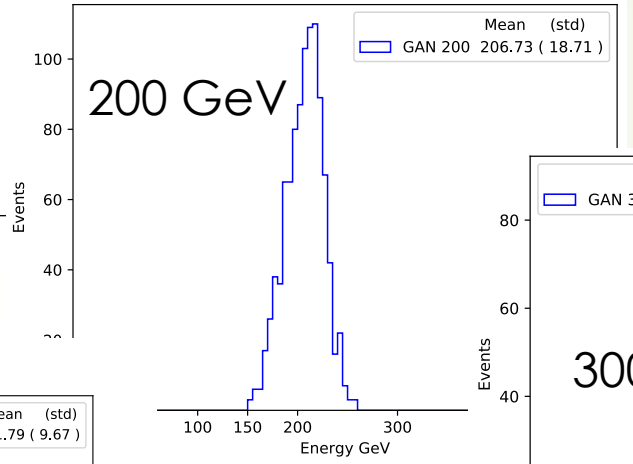


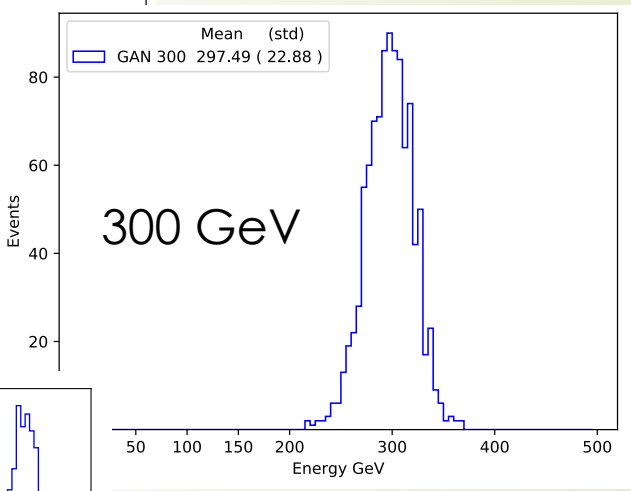
Energy regression

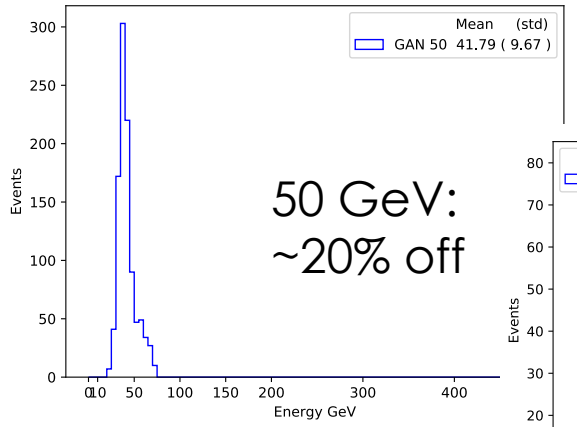
- Consistency check
  - Generate single energy samples
  - Compare to discriminator reconstructed energy
- Agreement is good.
  - Performance degrades at the edges

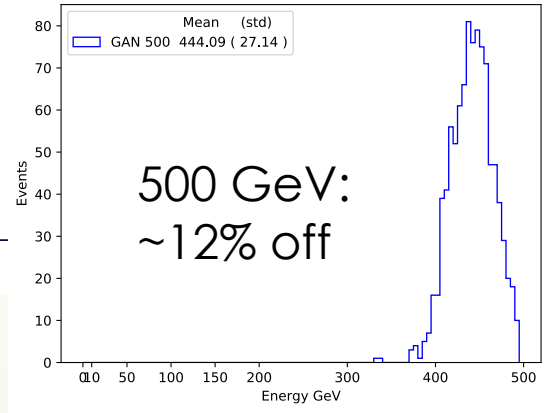












Next steps

Have a first example ready by the end of the year

- Ongoing optimization
  - Test interpolation / extrapolation
  - Effect of training sample statistics
- Comparison to other fast simulation approaches
- Generalize to multi-class approach (or multi-discriminator): primary particle entry point, angle, etc..
- Start optimization via hyper-parameter scan

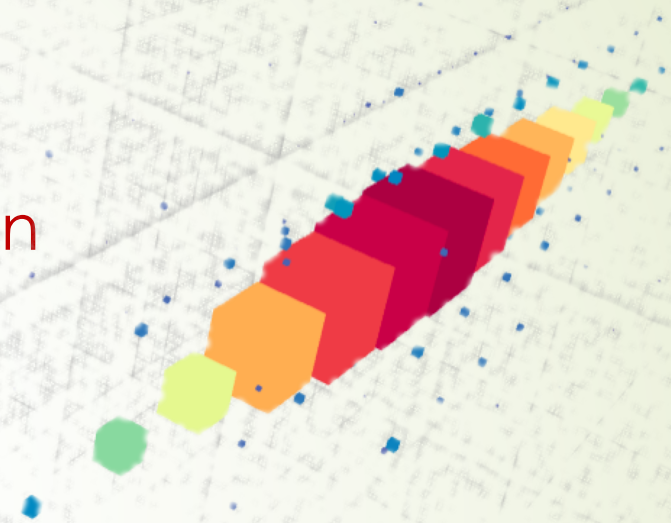
Timing

Keras + Tensorflow Implemetation	Time/Shower (msec)
My laptop (2.6 GHz Intel i7)	66
NVIDIA GeForce GTX 1080	0.04

- Generating a shower from trained model is extremely fast!
- Training takes about 1 day on the GTX1080
- Training time cannot become a bottleneck
  - Depending on the final use case retraining the networks might be necessary
  - Embedded algorithms to perform hyper-parameters tuning and meta-optimization
  - Scan large hyper-parameter space

Collaboration with Intel and CINECA (Italy) to test scaling on Marconi Intel Xeon Phi cluster

Summary



- A configurable framework for fast simulation
  - Easily embed user code
  - A reasonable library of algorithms
- A ML based tool
  - Evaluation and optimization of several models
- Integration in GEANTV: inference step first, then training tool
- Prototype interface and ML proof of concept in GEANTV alpha
  - Generalization and meta-optimization later on
  - It could be back-ported to Geant4

Thank you