GANs for simulation: development for distributed computing systems

Elena Orlova, Higher School of Economics, Moscow, Russia

Advised by Sofia Vallecorsa, Andrei Gheata

14 August, 2017

Outline

- Introduction
  - Generative Adversarial Networks
- Implementation
- Summary

Introduction

Experiments need tools for fast simulation. Currently available solutions depend on the experiments.

We want to:

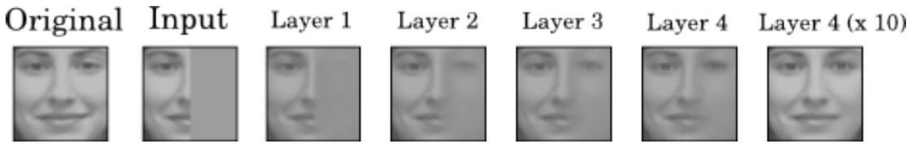
- have a generic interface in GeantV capable of using different fastsim options including ML based
- reproduce particle interactions with matter in a detector and provide related quantities such as energy or particle type

Generative Adversarial Networks!

DL for simulation

Generative models (Generative Stochastic Networks, Variational Auto-Enconders, Generative Adversarial Networks, ..) can be used for simulation:

- Realistic generation of samples
- Use complicated probability distributions
- Optimize multiple output for a single input
- Work well with missing data



Ranzato, Susskind, Mnih, Hinton, IEEE CVPR 2011

https://arxiv.org/pdf/1605.05396.pdf

GAN results

Samples of images of bedrooms generated by a DCGAN trained on the LSUN dataset.

Samples drawn trained on the CIFAR-10 dataset

https://arxiv.org/pdf/1701.00160v1.pdf

Generative Adversarial Networks

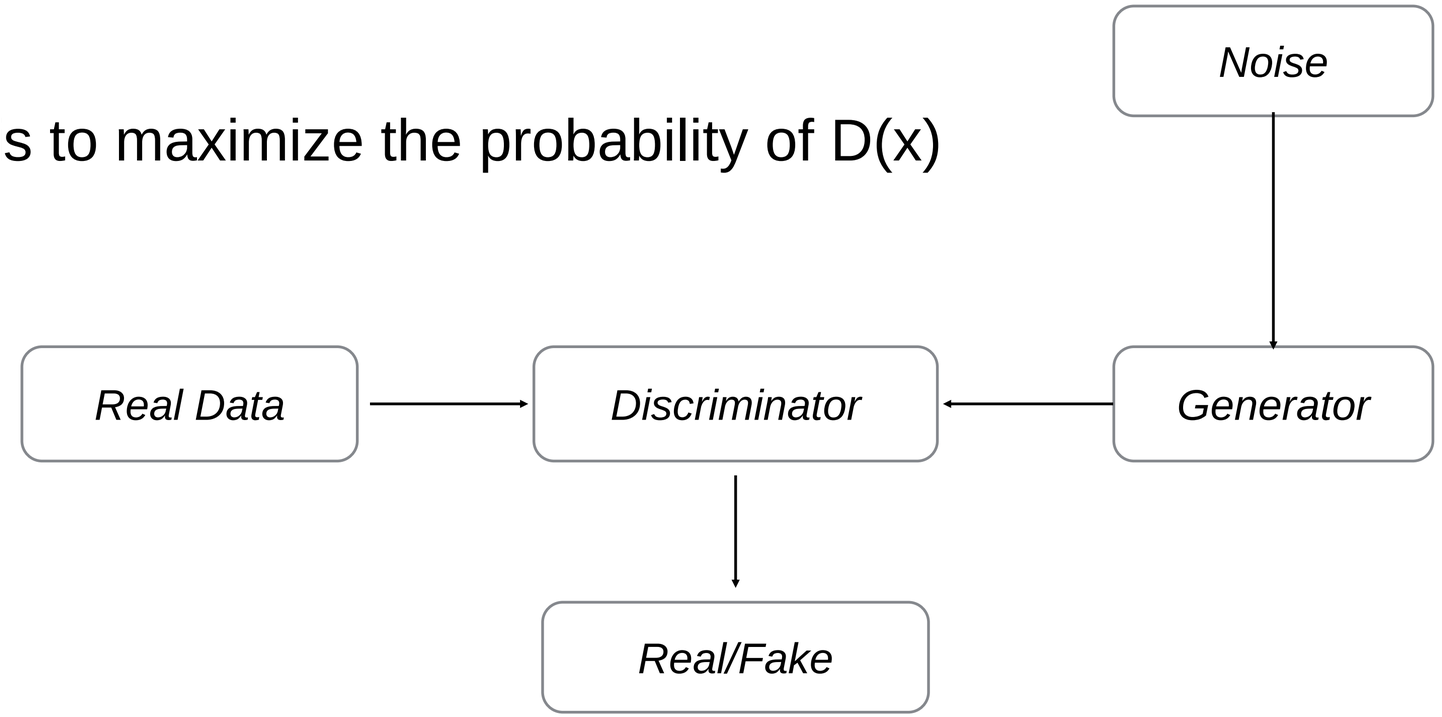
Generator vs. **Discriminator**

Generative Adversarial Networks

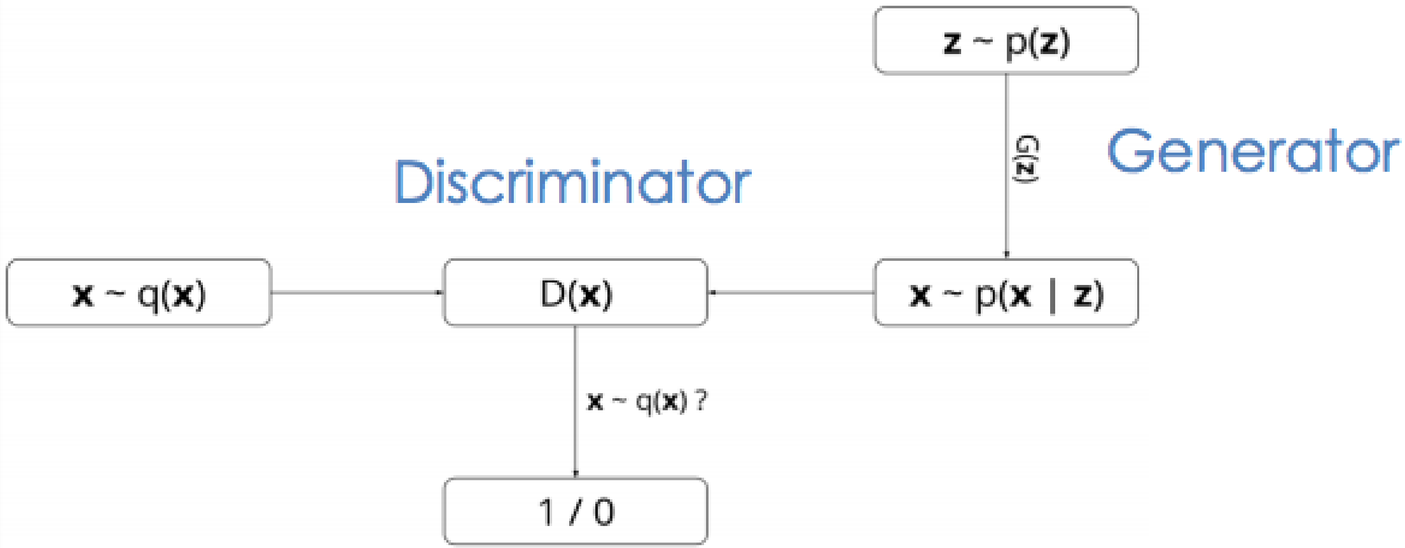
Two models:

- D(x) classifies images: fake or real
- G(x) produces images taking as input some random noise

Training procedure for G(z) is to maximize the probability of D(x) making a mistake



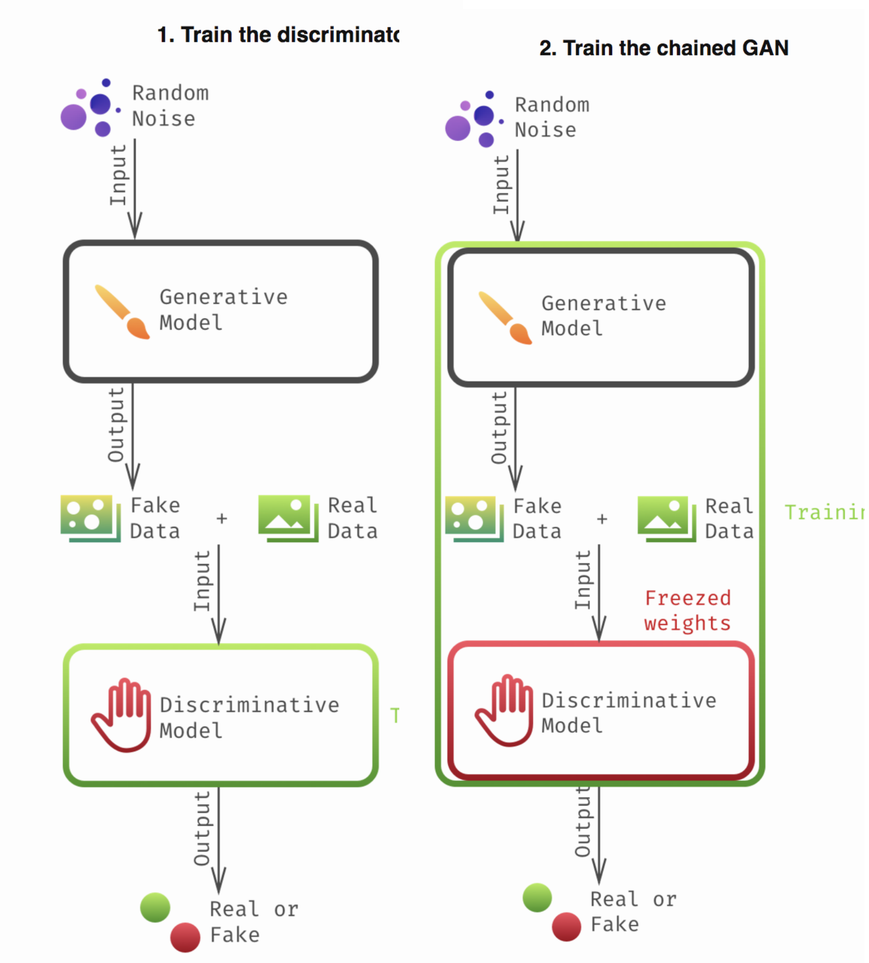
Generative Adversarial Networks



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

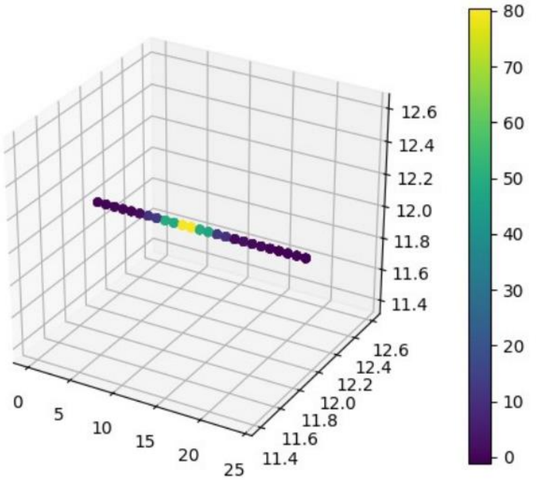
Train procedure

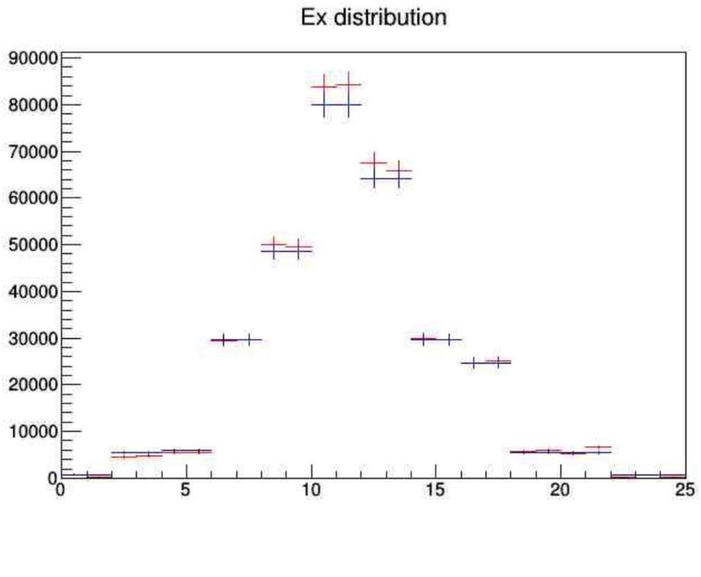
- Sample noise and generate images with G
- Sample images from training dataset and train the D to recognize G data from real data
- Train combined G + D to tell you that G data it is real
- At this stage D weights are frozen.
- Back feed info to discriminator and repeat for as many epochs as needed

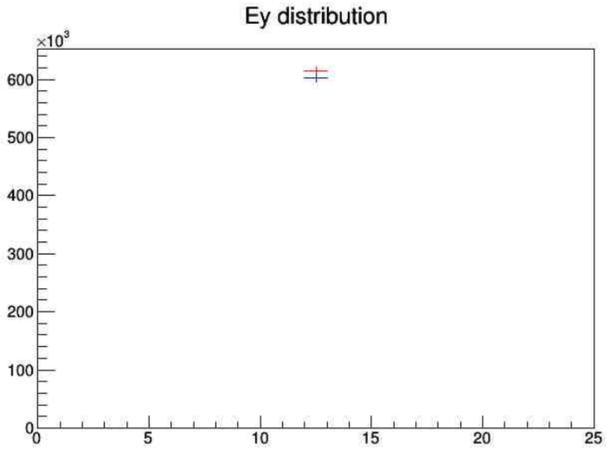


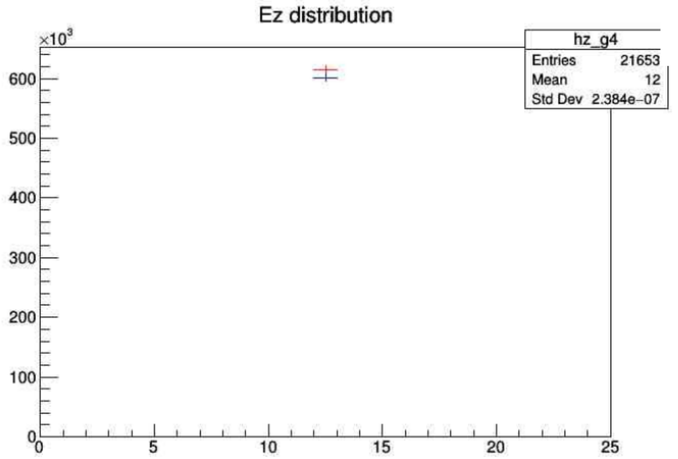
Toy gaussian test

- Generate simplified 1D distribution to test GAN behavior
- Two gaussians centered at x=0.
- Fake a two classes (particles) signal
- Reproduce distribution with GAN



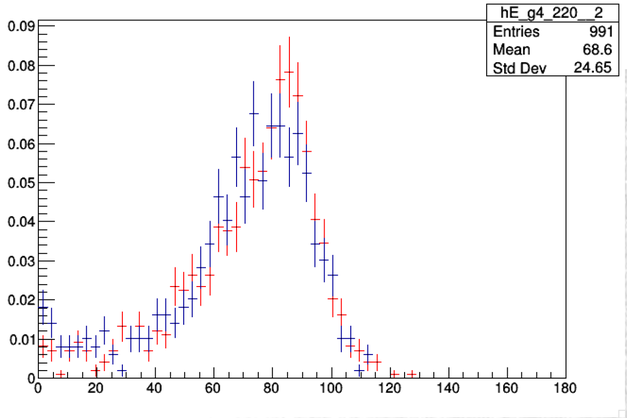


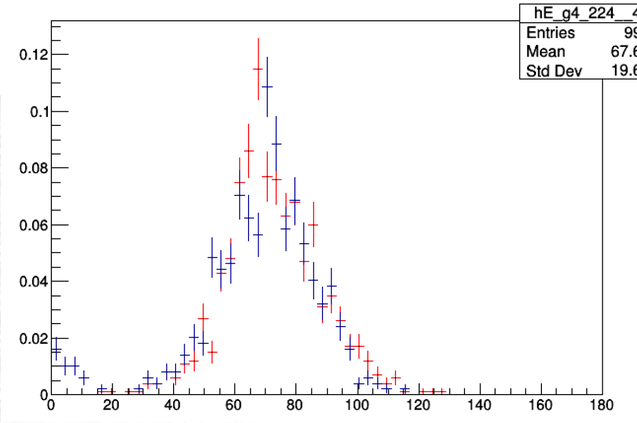


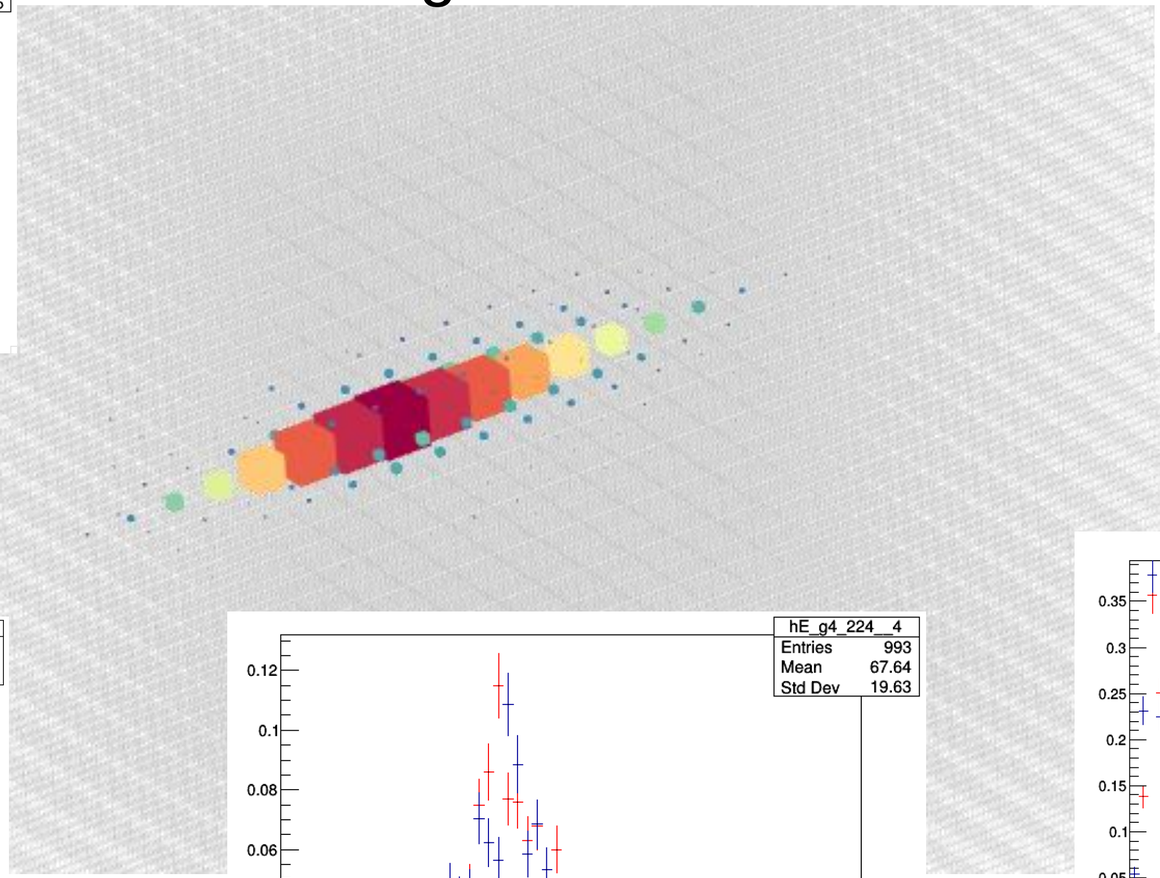


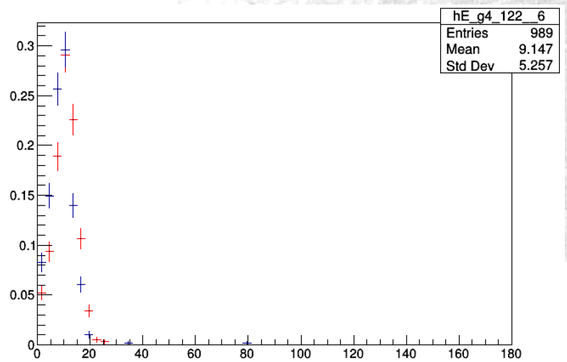
LCD calorimeter showers

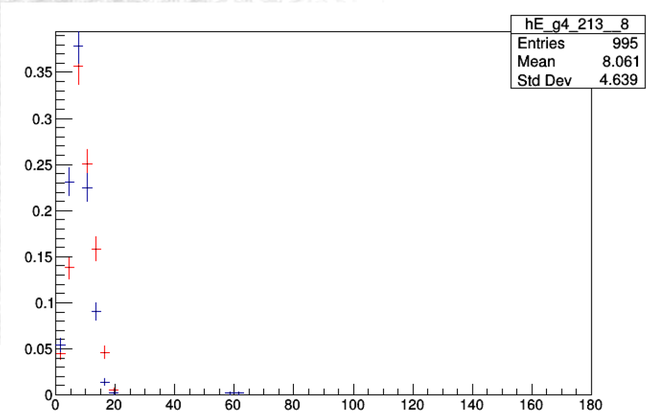
Single electrons



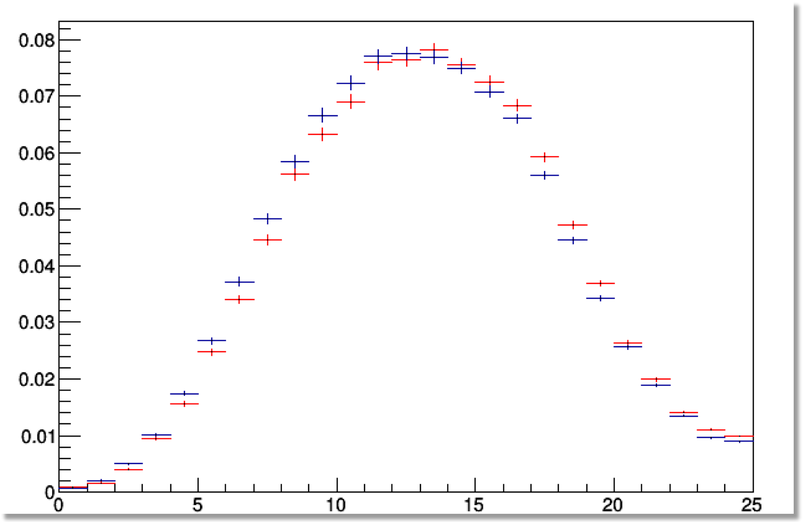




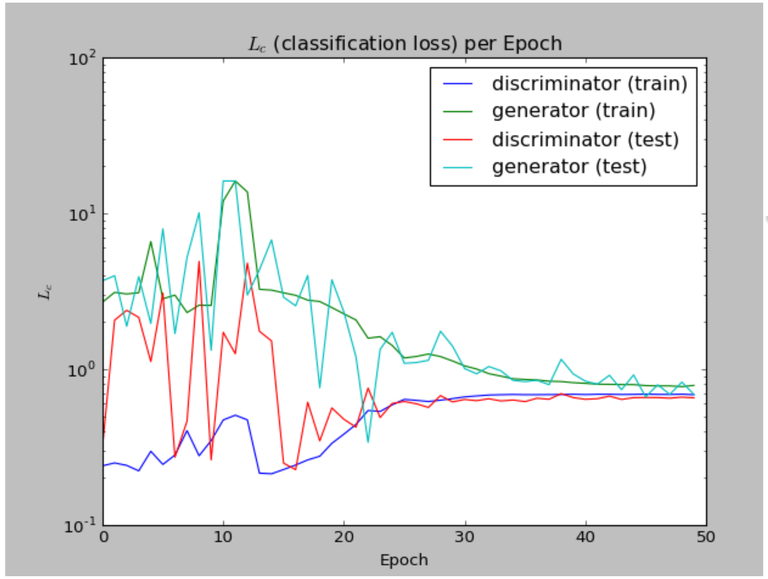


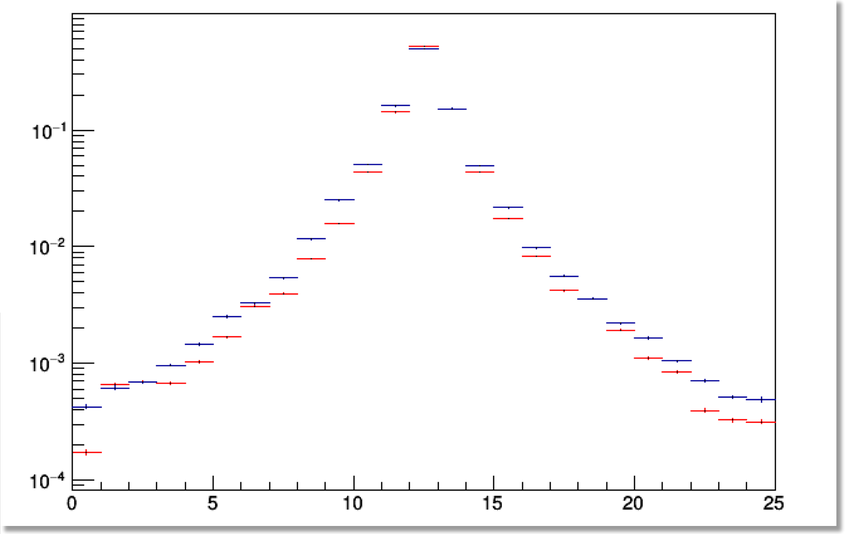


LCD calorimeter showers



Shower longitudinal section





Shower transverse section

Improving training

1 day on GTX1080

Improving performance:

- hyperparameters
- train on CPU clusters not only GPU clusters

neon

neon

- new deep learning framework committed to best performance on all hardware

- designed for ease-of-use, extensibility and « out-of-the-box » scaling through multiple nodes.

http://neon.nervanasys.com/index.html/

Keras/neon

Keras

- No 3D transpose convolutional layer:
  - Conv + Upsampling

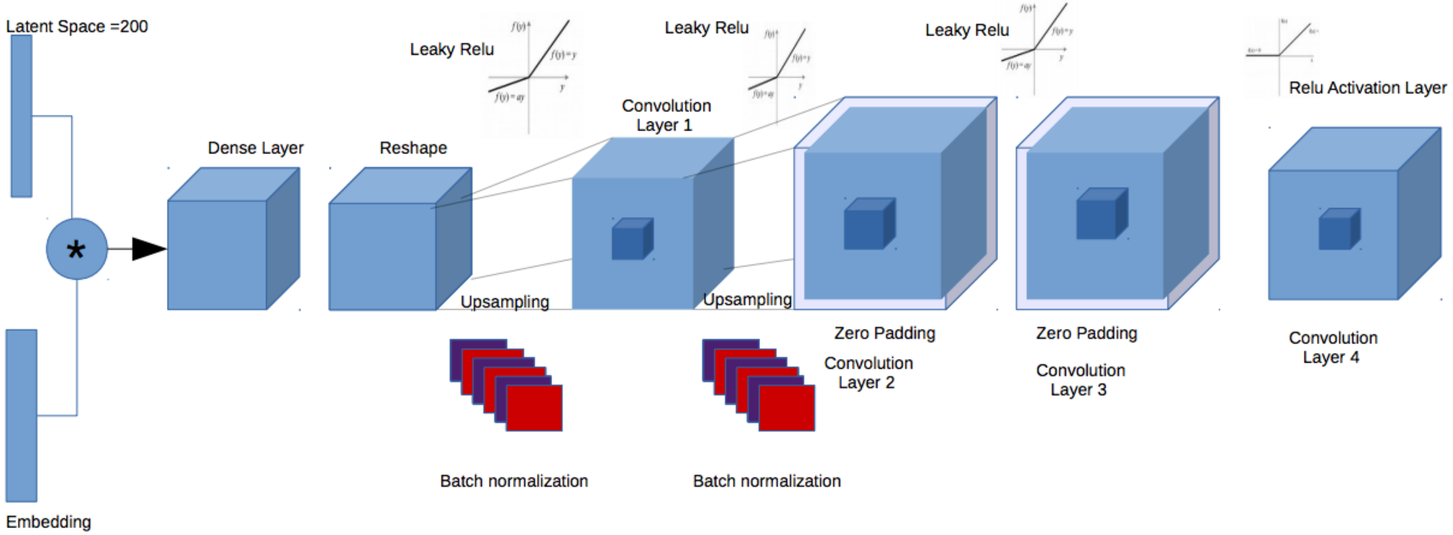
Neon

- Transpose convolutional layer in 3D
- GAN model is implemented

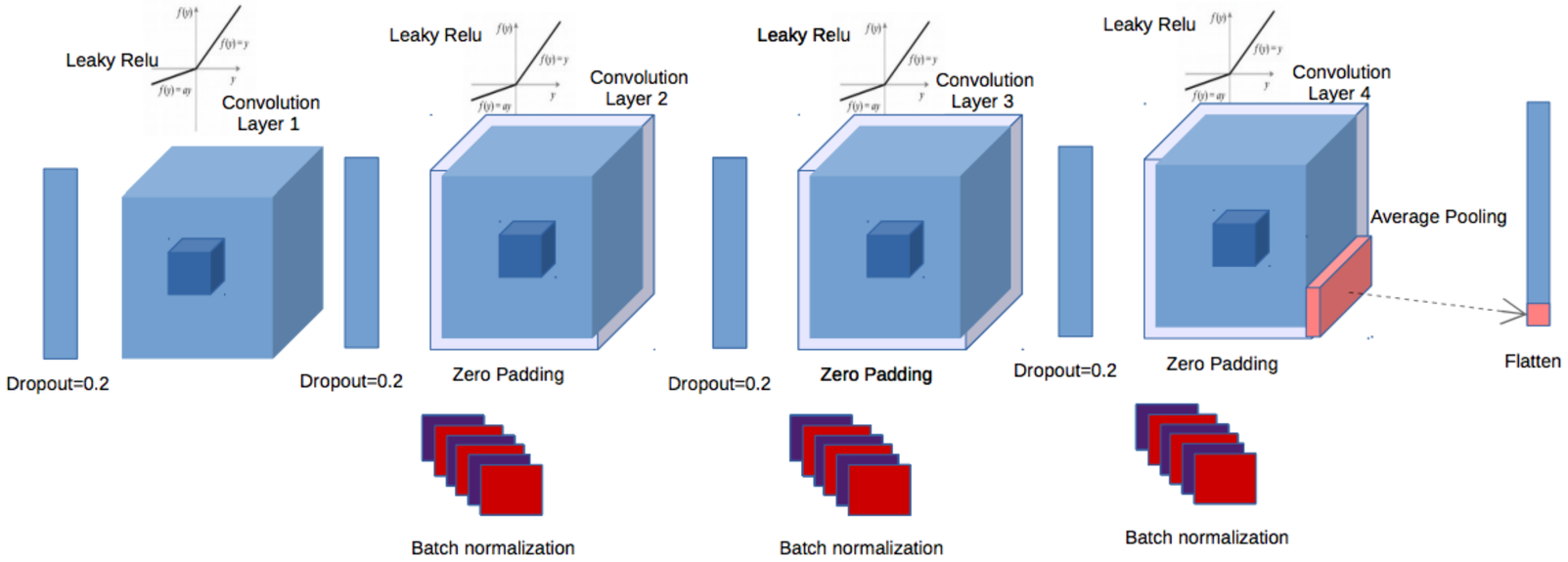
Output: $\boldsymbol{1 + (W-F+2P)/S}$,
where W - input volume, F - filter size, P - padding, S - stride

GAN in 3D

- Generator and Discriminator are based on 3D convolutions

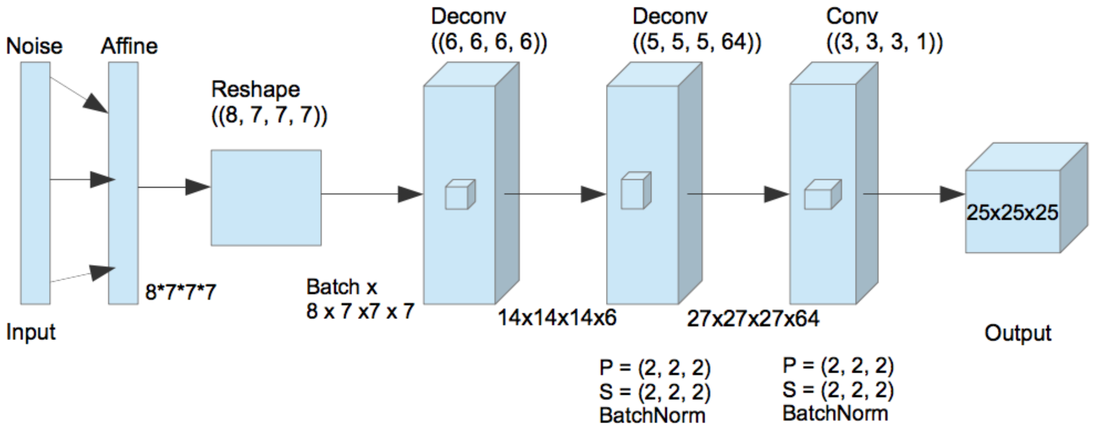


Generator

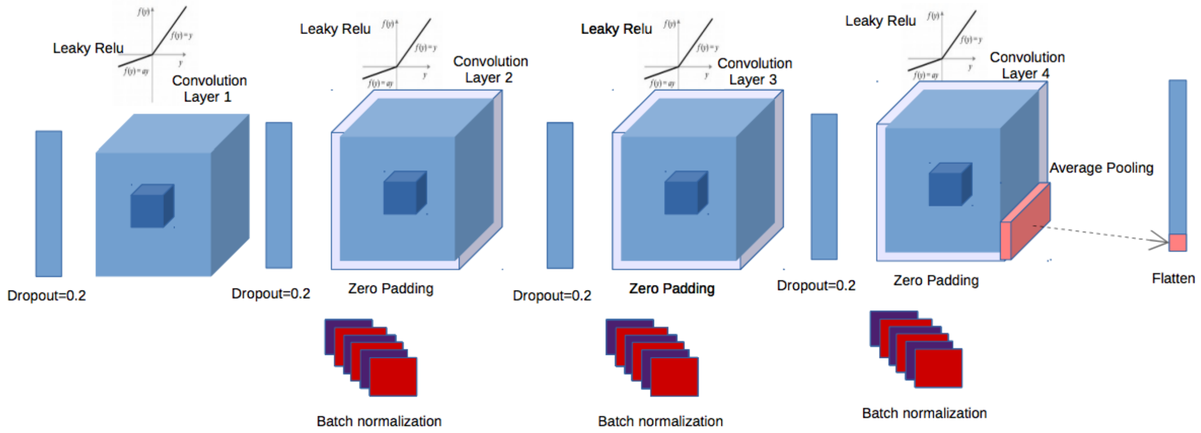


Discriminator

neon model



Generator



Discriminator

Current state

Our case is non standard and common framework might not be optimised for it: we have to work in collaboration with developers to fix issues:

- 3D deconv
- “mkl” backend
- no embedding layer

Conclusion and plans

- keep working on implementation
- add parametrisation
- testing with different parameters on different architectures

Motivation

Historically simulation in particle physics has relied on Monte Carlo methods. Deep learning is a completely new approach.

It will allow physicist to produce a huge amount of simulated data , needed by the next high luminosity LHC experiments, using limited computing resources .

Thank you!

Intel Competiton

The Modern Code Developer Challenge

- The goal for Intel is to give budding developers the opportunity to use modern programming methods to improve code that helps move science forward.

https://software.intel.com/en-us/articles/the-modern-code-developer-challenge