Deep generative models for fast shower simulation in ATLAS

Dalila Salamani

University of Geneva

dalila.salamani@cern.ch

On behalf of the ATLAS collaboration

14th e-science IEEE International Conference

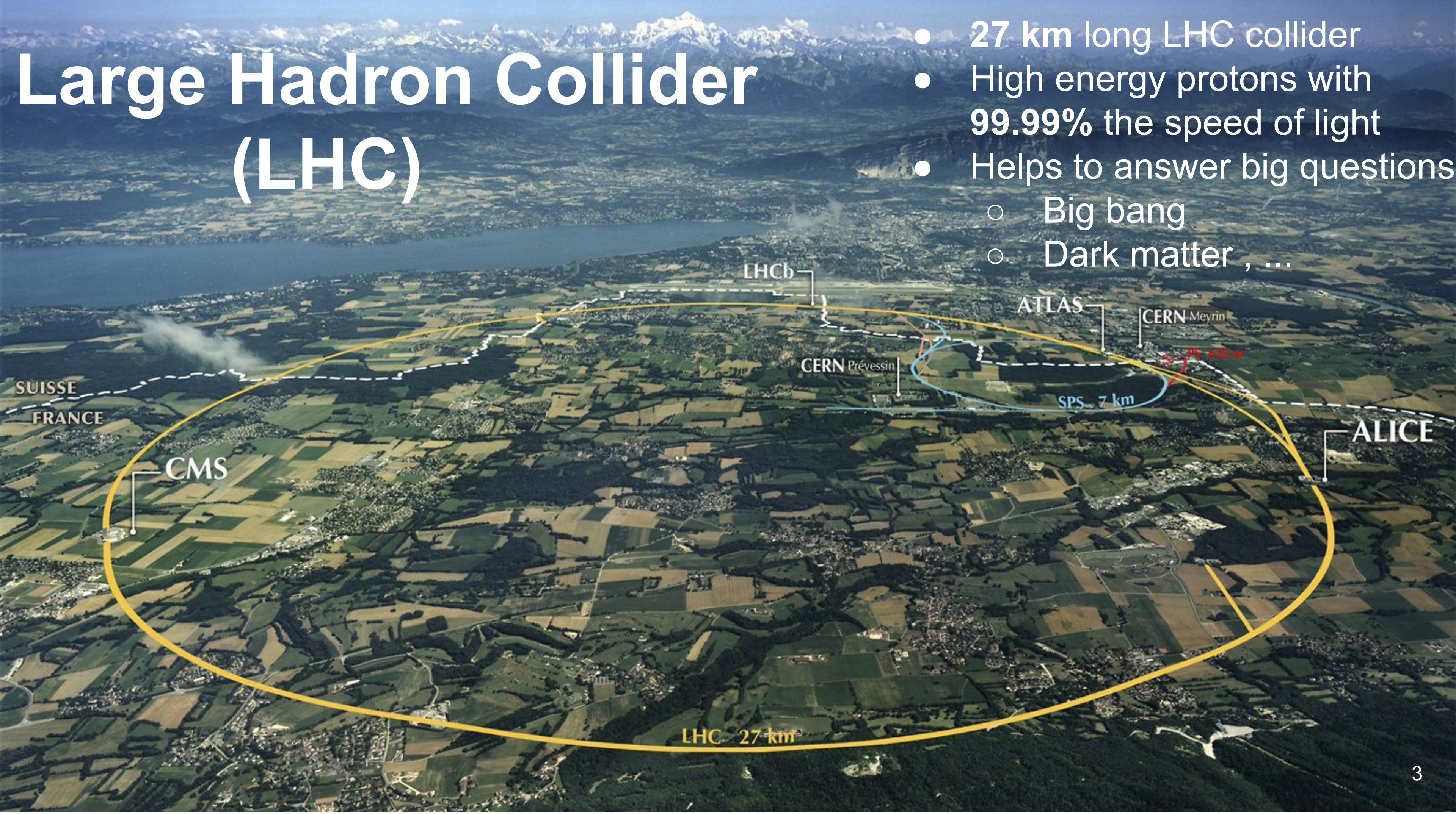
2018

Outline

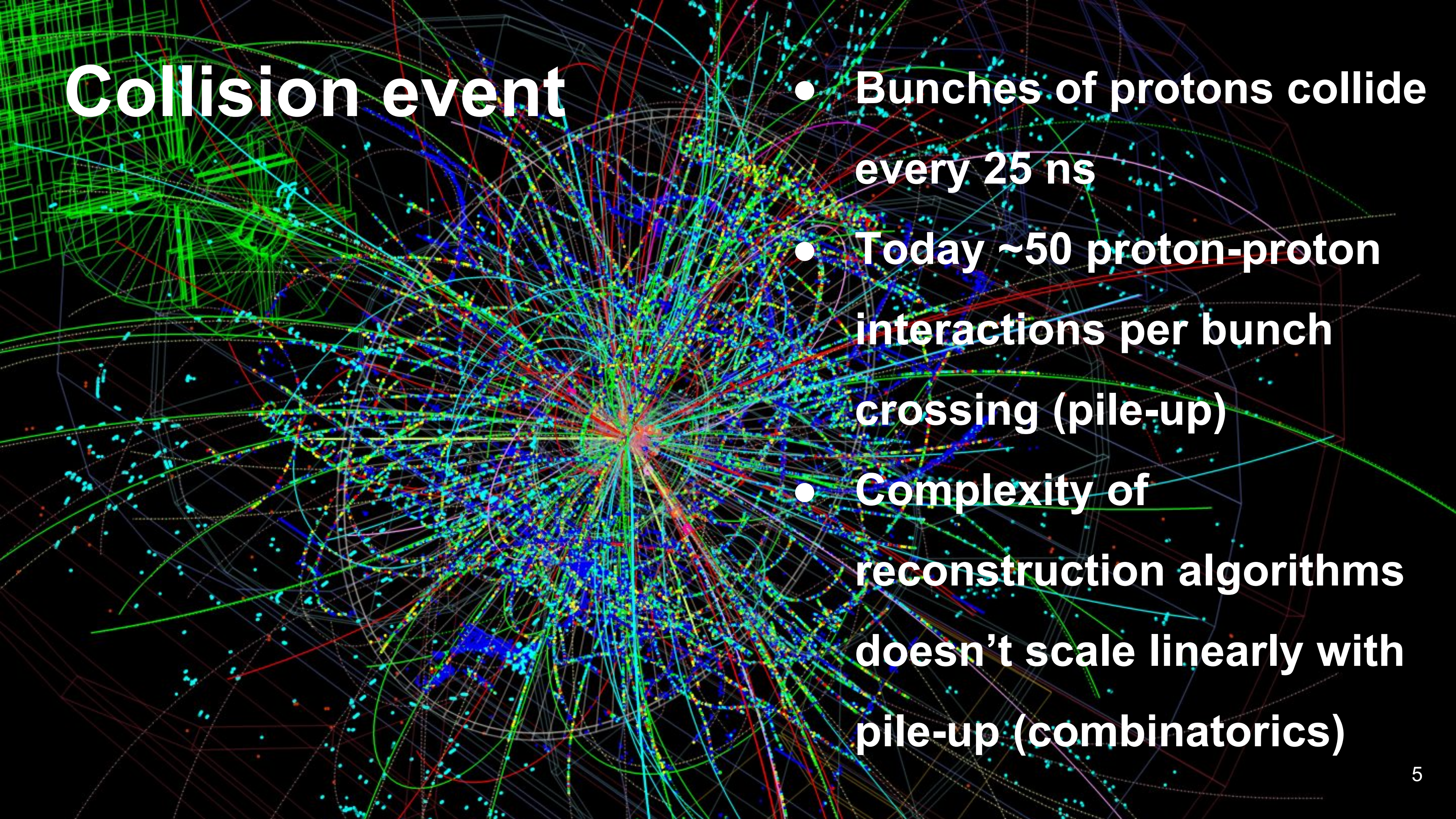
- Context & Motivation
- Rise of generative models
  - Generative models for HEP
  - Model architectures : VAE & GAN
- Validation of generation performance
- Conclusion

Large Hadron Collider (LHC)

- **27 km** long LHC collider
- High energy protons with **99.99%** the speed of light
- Helps to answer big questions
  - Big bang
  - Dark matter , ...



Collision event

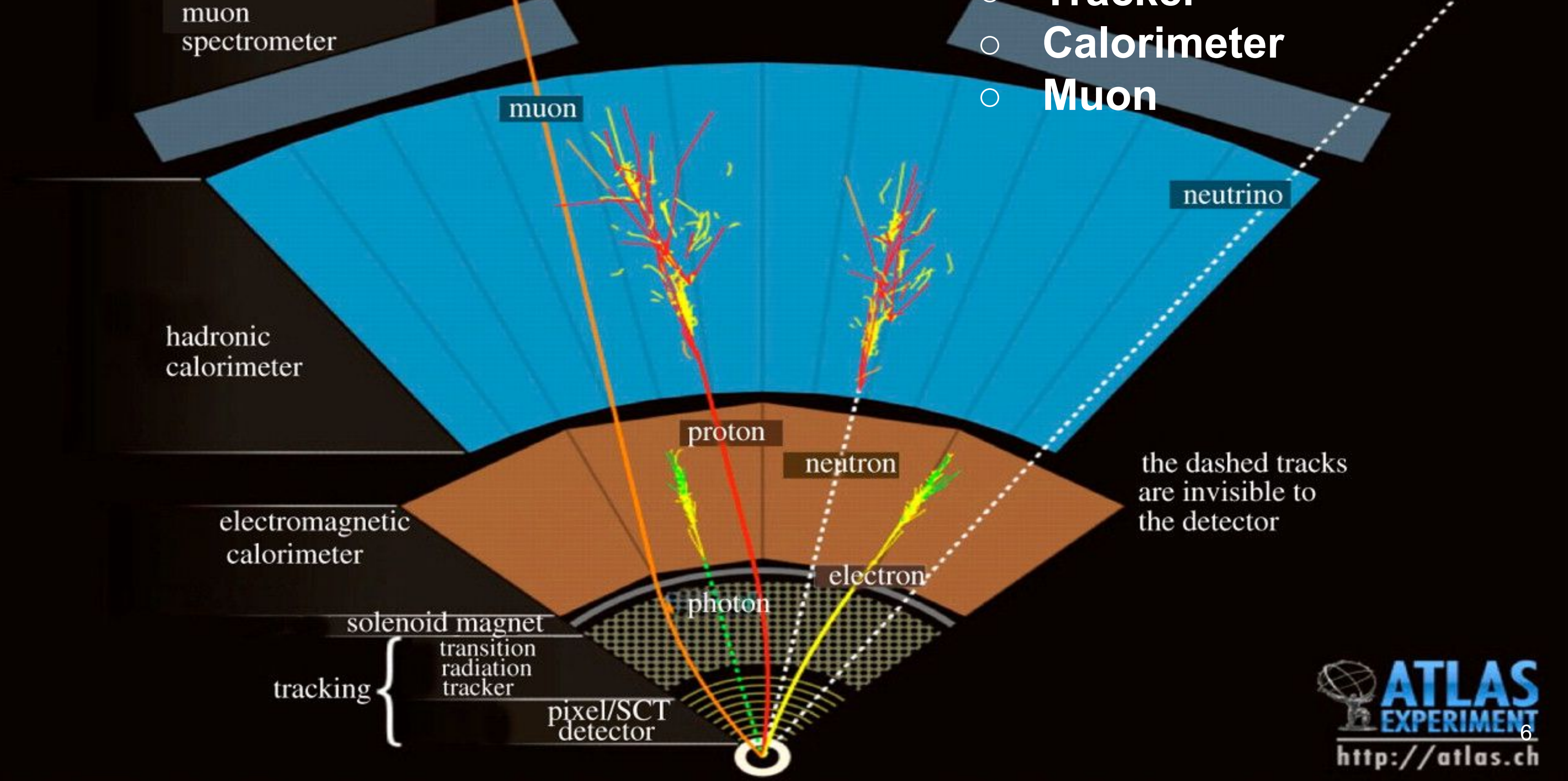


Collision event

- Bunches of protons collide every 25 ns
- Today ~50 proton-proton interactions per bunch crossing (pile-up)
- Complexity of reconstruction algorithms doesn't scale linearly with pile-up (combinatorics)

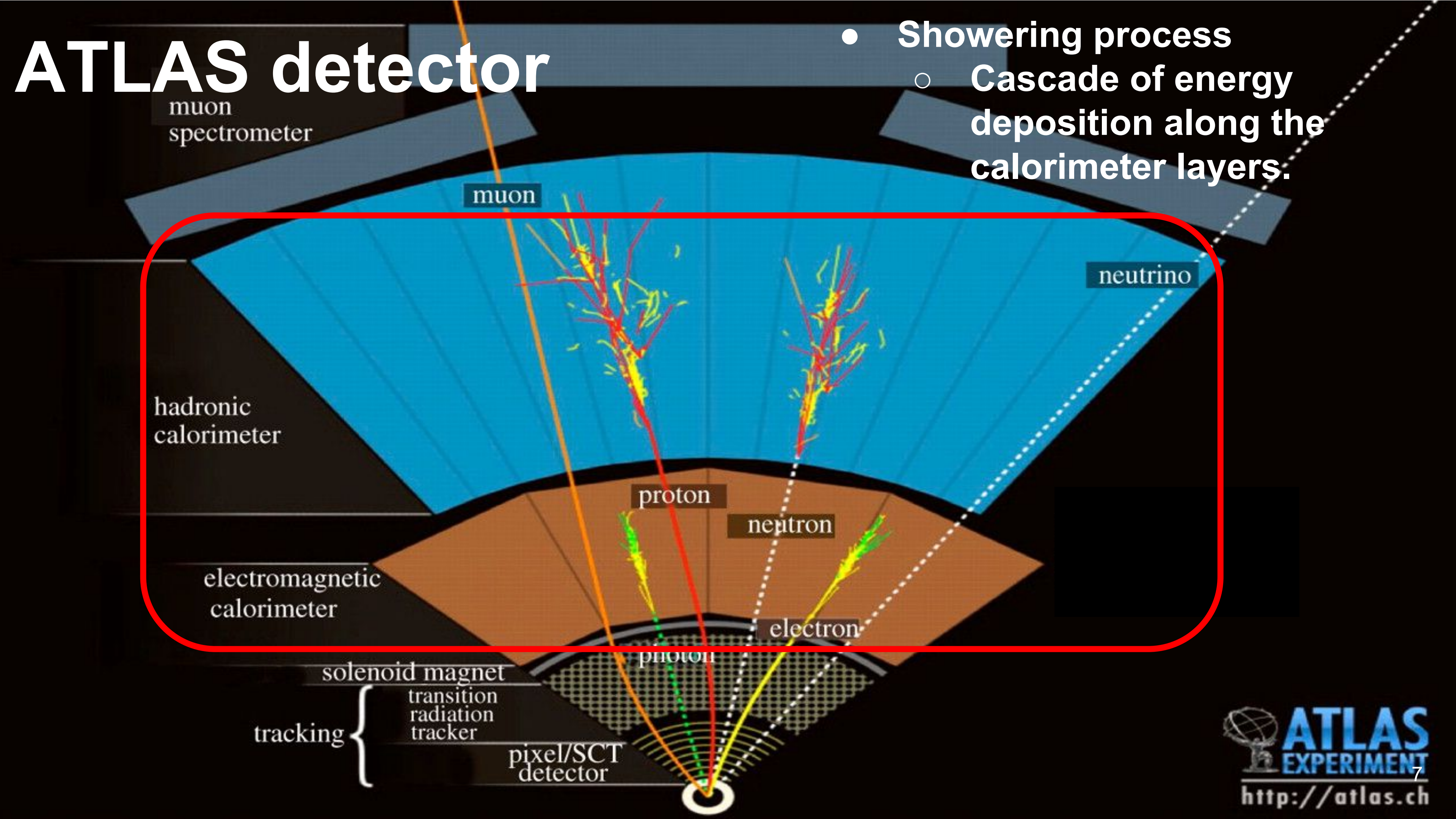
ATLAS detector

- ATLAS detector
  - Tracker
  - Calorimeter
  - Muon



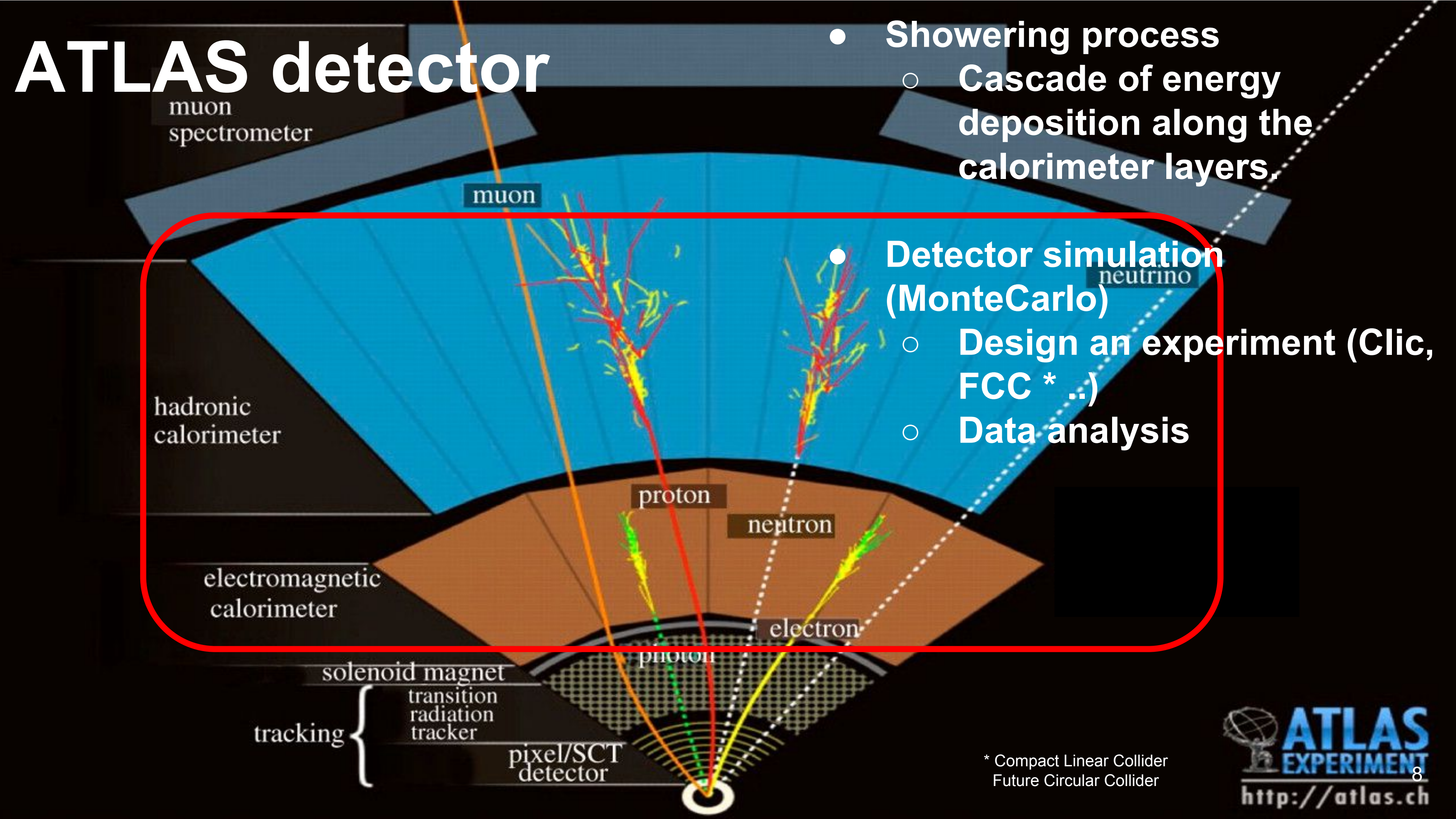
ATLAS detector

- **Showering process**
  - **Cascade of energy deposition along the calorimeter layers.**



ATLAS detector

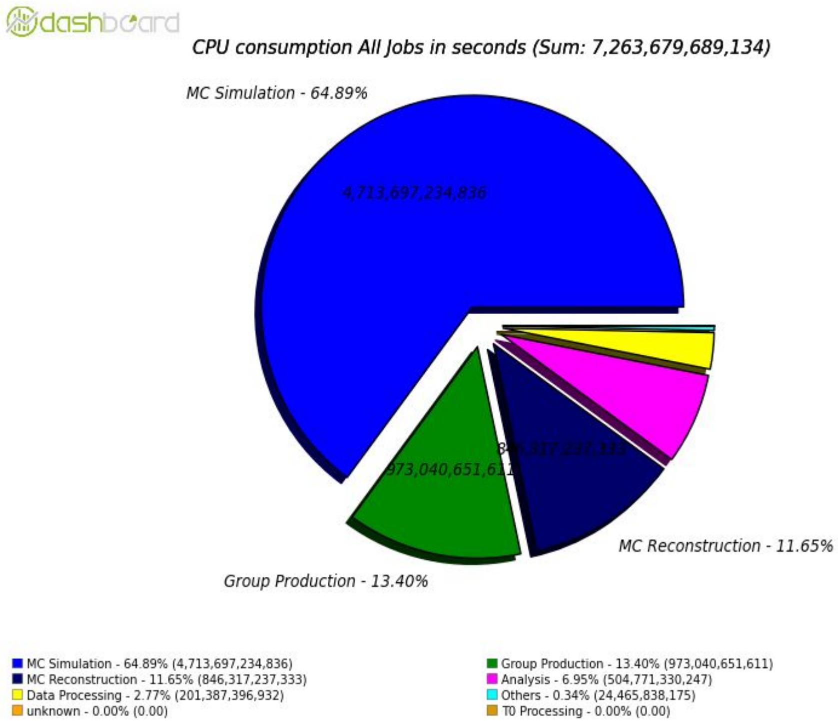
- Showering process
  - Cascade of energy deposition along the calorimeter layers.
- Detector simulation (MonteCarlo)
  - Design an experiment (Clic, FCC * ..)
  - Data analysis



* Compact Linear Collider
Future Circular Collider

Motivation

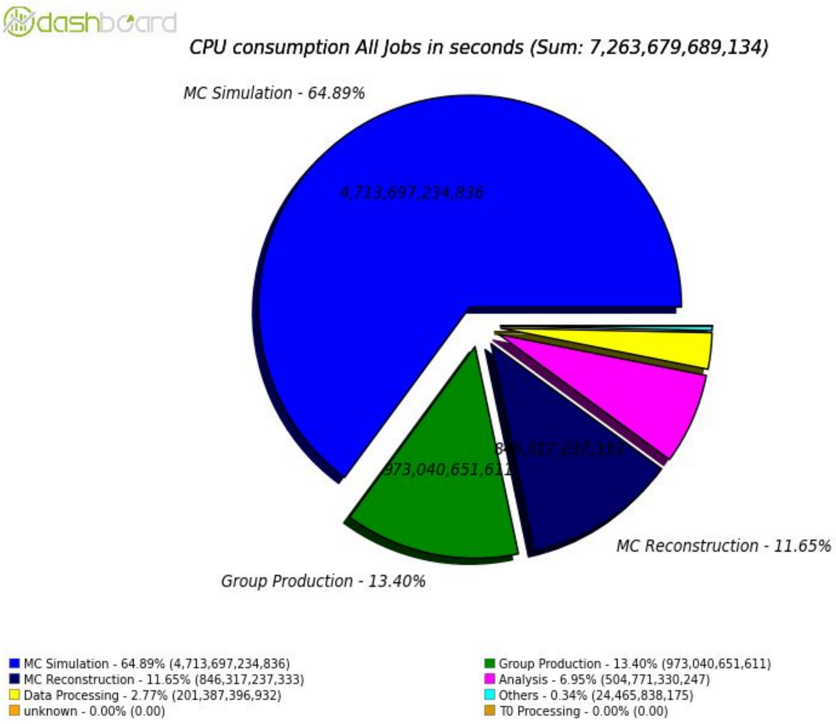
- Successful physics program in ATLAS depends on the availability of high statistics Monte Carlo simulated events.
- Currently **>50 % of ATLAS computing time** is spent on shower simulation.
- LHC is collecting more and more events (High Luminosity Upgrade) → more CPU consumption.



Reference

Motivation

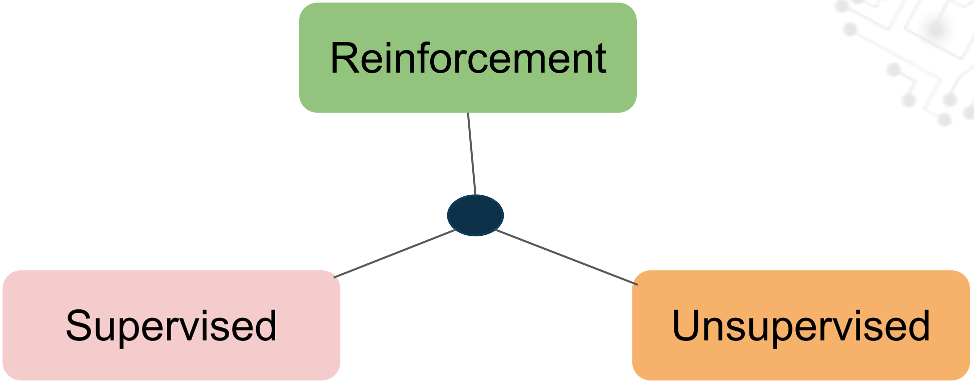
- Successful physics program in ATLAS depends on the availability of high statistics Monte Carlo simulated events.
- Currently **>50 % of ATLAS computing time** is spent on shower simulation.
- LHC is collecting more and more events (High Luminosity Upgrade) $\rightarrow$ more CPU consumption.
- Challenge: Develop fast shower simulation framework.



Reference

Machine learning

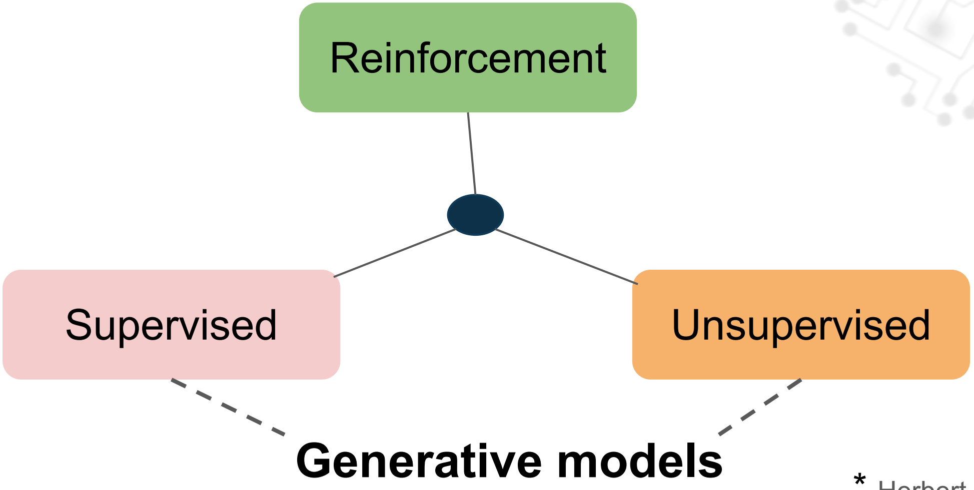
- Learning process
  - Learn to improve performance by experience *
  - Automatic & Adapted model to domain application
  - Discover knowledge from dataset (engineering bottleneck)



* Herbert Simon, Turing Award 1975, Nobel prize 1978

Machine learning

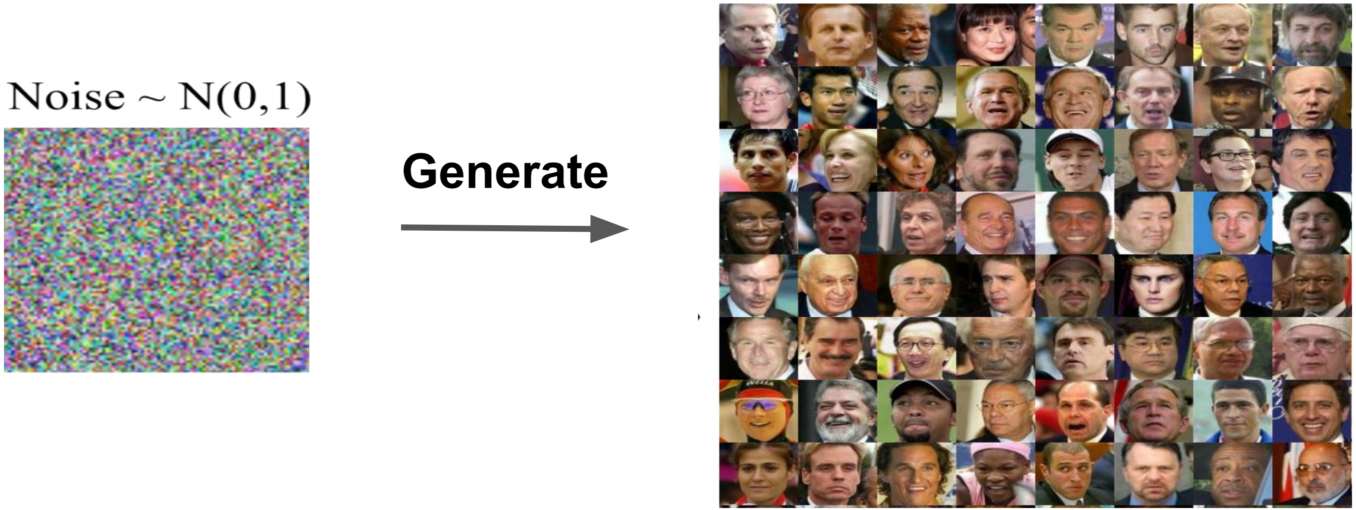
- Learning process
  - Learn to improve performance by experience *
  - Automatic & Adapted model to domain application
  - Discover knowledge from dataset (engineering bottleneck)



* Herbert Simon, Turing Award 1975, Nobel prize 1978

Generative models

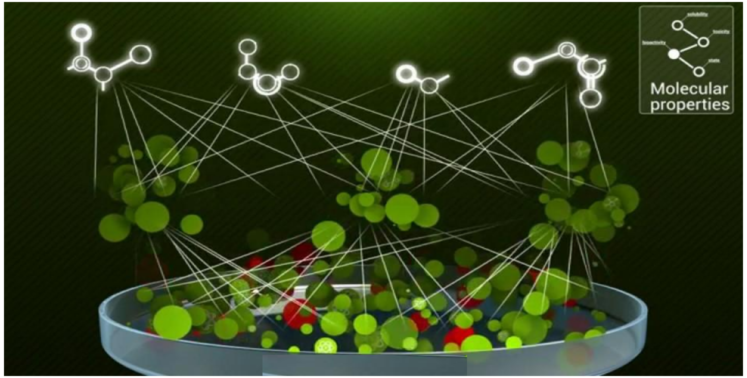
- Learn the true **data distribution** of the training set **<u>to reproduce it</u>**.
- **<u>Adopted approaches</u>**: Use of deep neural networks to learn the approximation function of the true (& sparse) distribution, Variational Autoencoders (**VAEs**) & Generative Adversarial Networks(**GANs**)



Generative models : domain application

Learning to generate speech : [Den Oord et al, 2016]

Drug Discovery : [Chen et al, 2018]



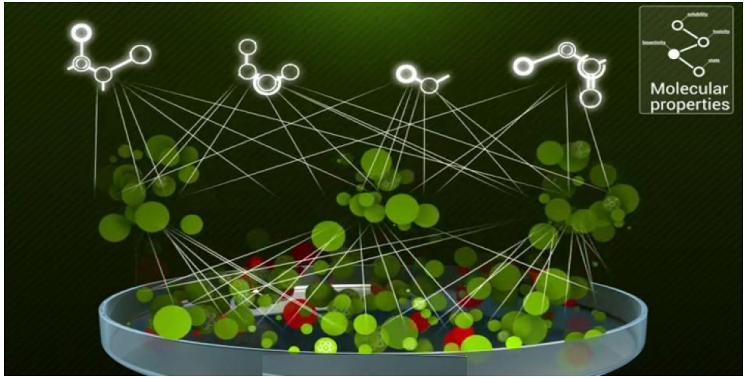
Learning to generate images : [Brock et al, 2018]

Generative models : domain application

Learning to generate speech : Den Oord et al, 2016

And now...HEP

Drug Discovery : Chen et al, 2018



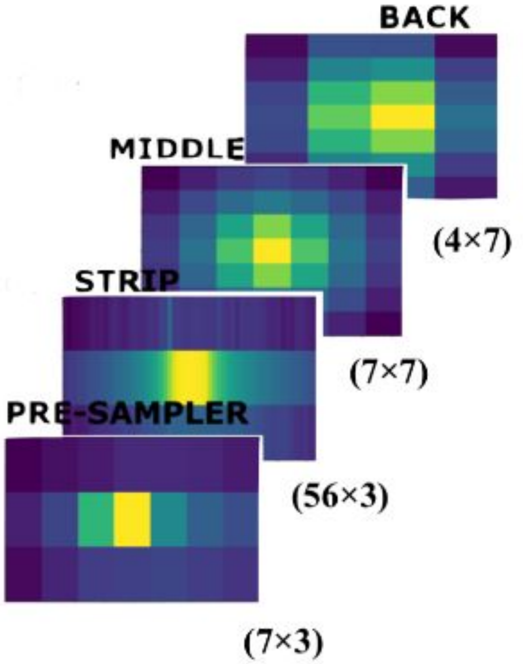
Learning to generate images : Brock et al, 2018

Generative models for **HEP (Showering)**

- Model the shower process.
- Take into account the ATLAS calorimeter geometry.
- Validation : shower shape variables distribution comparison.
- Fast & accurate modeling.
- First application of deep generative models for fast shower simulation in ATLAS: Public Note ATL-SOFT-PUB-2018-001.

Dataset & preprocessing

- Single photon samples in the electromagnetic. calorimeter (4 layers with different granularities).
- Pseudorapidity **0.20 < |η| < 0.25**.
- Energies in **[1, 260] GeV** logarithmically spaced.
- A total of **266 cells** (7 x 3, 56 x 3, 7 x and 4 x 7) are considered for energy deposits.

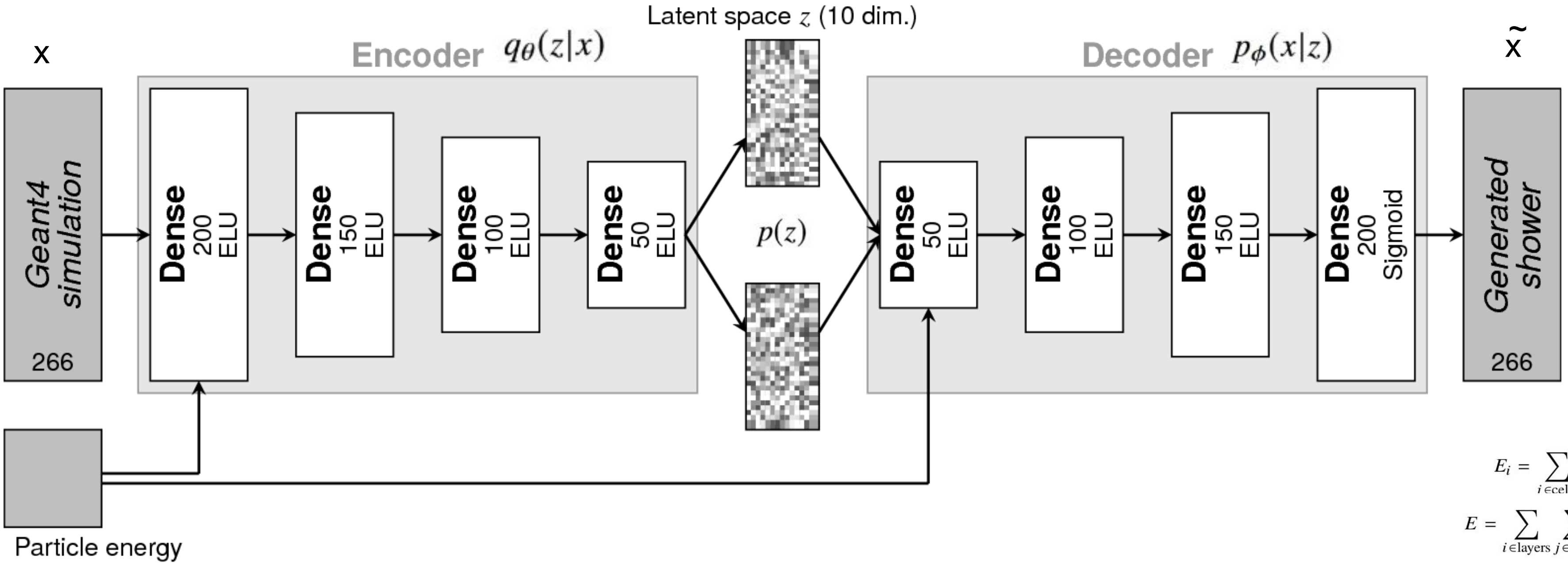


TensorFlow Keras

Using **HDF5** format

Reference

VAE model architecture



$$E_i = \sum_{i \in \text{cells}} E_{ij}$$

$$E = \sum_{i \in \text{layers}} \sum_{j \in \text{cells}} E_{ij}$$

$$L_{\text{VAE}}(x, \tilde{x}) = w_{\text{reco}} E_{z \sim q_\theta(z|x)}[\log p_\phi(x|z)] - w_{\text{KL}} \text{KL}(q_\theta(z|x)||p(z)) + w_{E_{\text{tot}}} L_{E_{\text{tot}}}(x, \tilde{x}) + \sum_i^M w_i L_{E_i}(x, \tilde{x})$$

Reconstruction Loss

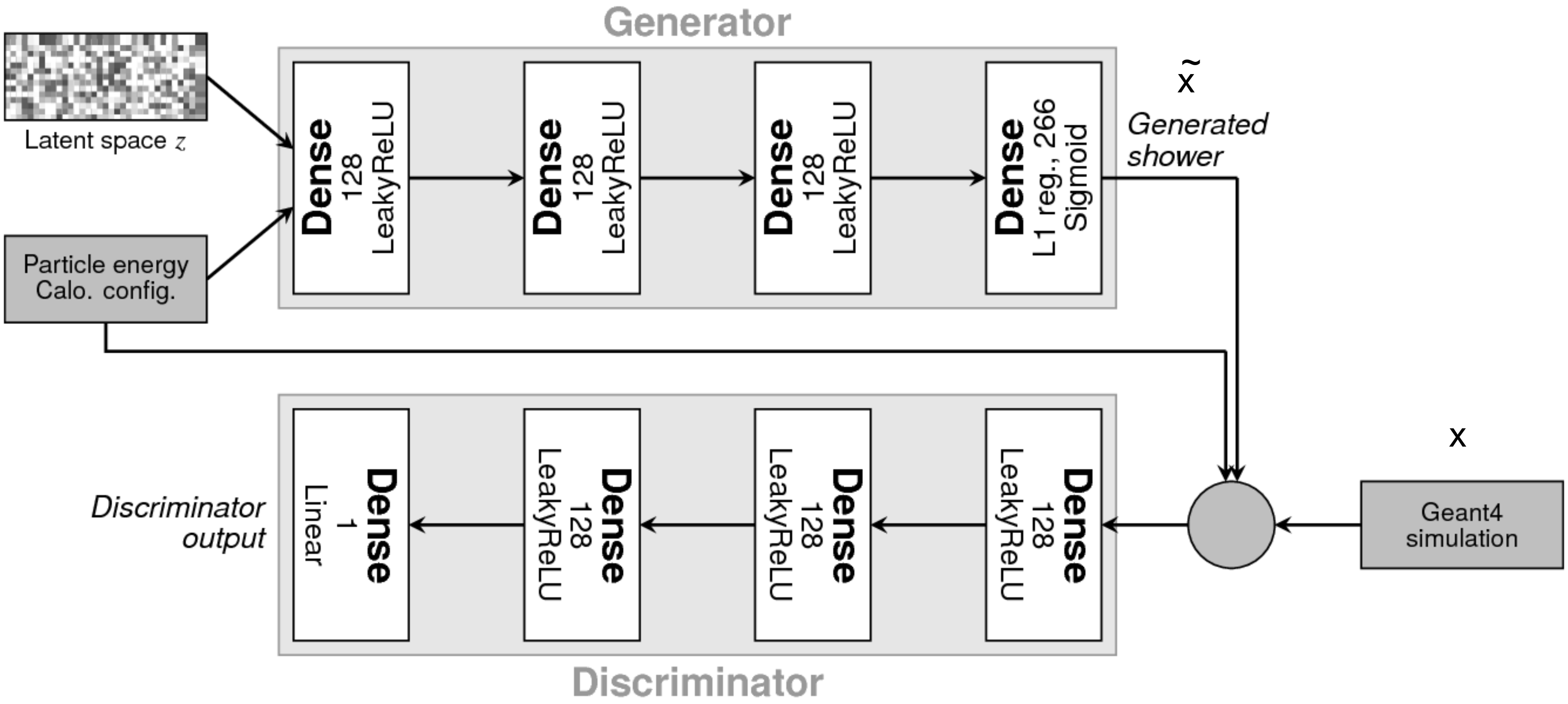
KL Loss

Total Energy Loss

Energy fraction per layer Loss

Reference

GAN model architecture



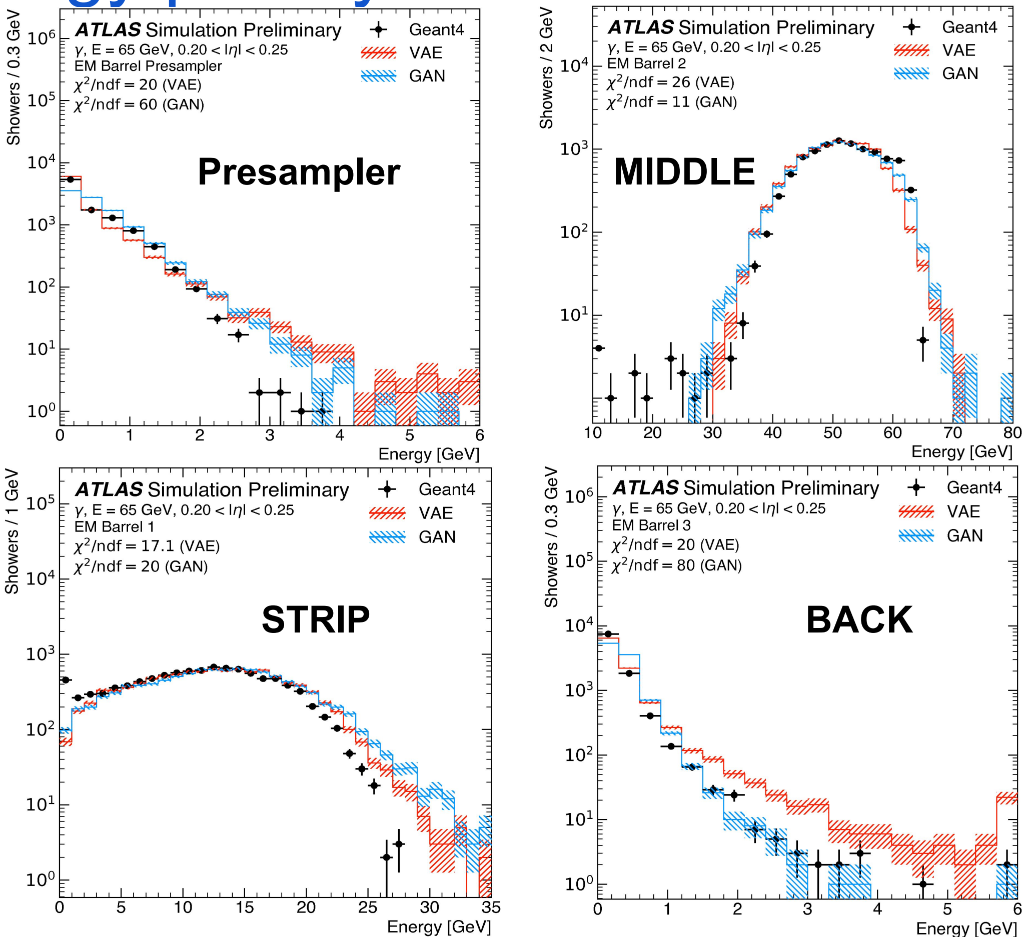
$$L_{\mathrm{GAN}} = \underset{\tilde{x}\sim p_{\mathrm{gen}}}{E}[D(\tilde{x})] - \underset{x\sim p_{\mathrm{Geant4}}}{E}[D(x)] + \lambda \underset{\hat{x}\sim p_{\hat{x}}}{E}[(||\Delta_{\hat{x}} D(\hat{x})||_2 - 1)^2]$$

Wasserstein loss

[Reference](#)

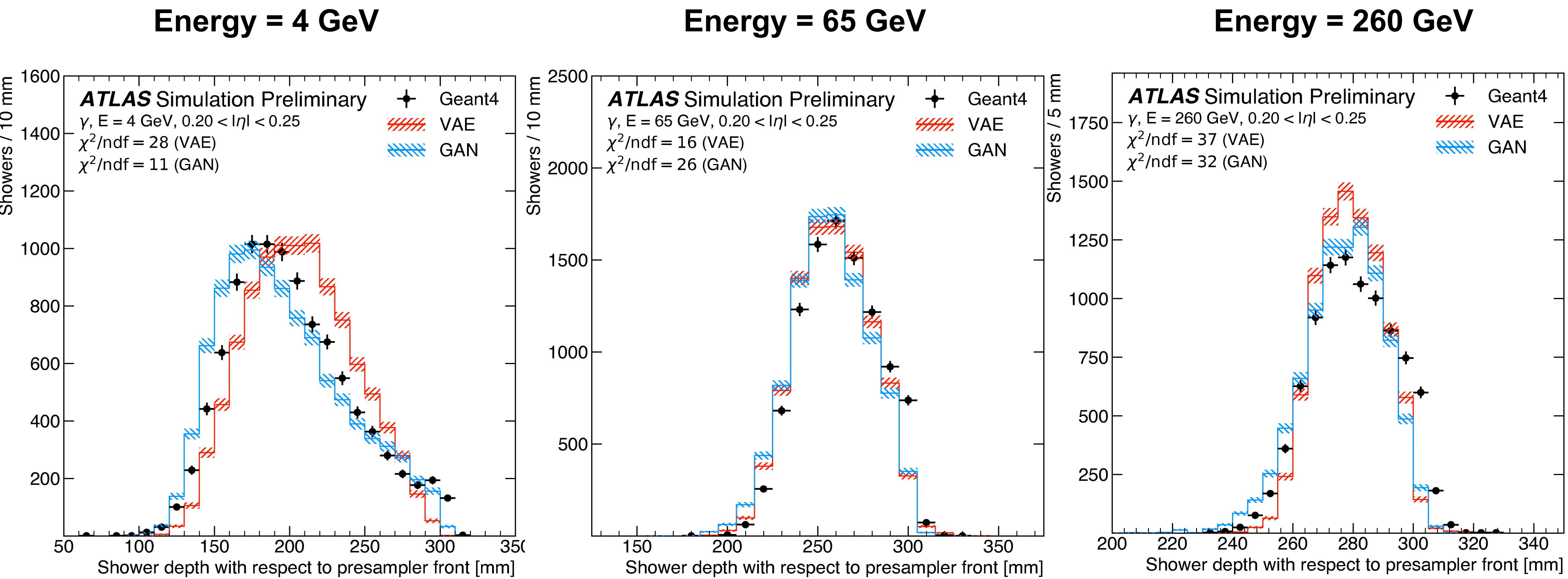
Generation results: energy per layer

- Energy deposited in the individual electromagnetic calorimeter layers for photons 65 GeV.
- Challenges posed by layers with low (and sparse) energy deposits, i.e. late showers.



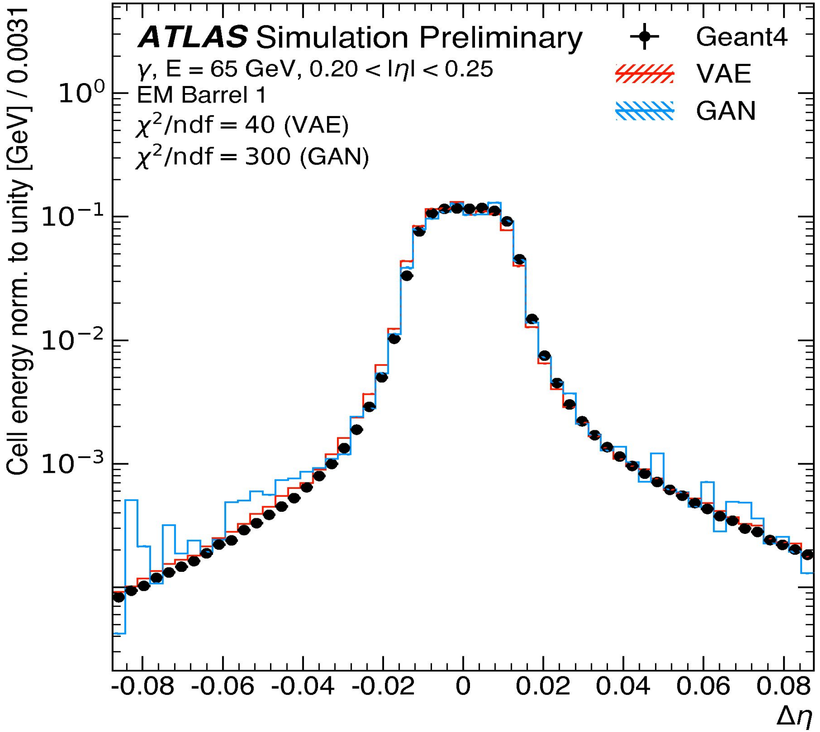
[Reference](#)

Generation results: reconstructed longitudinal shower center

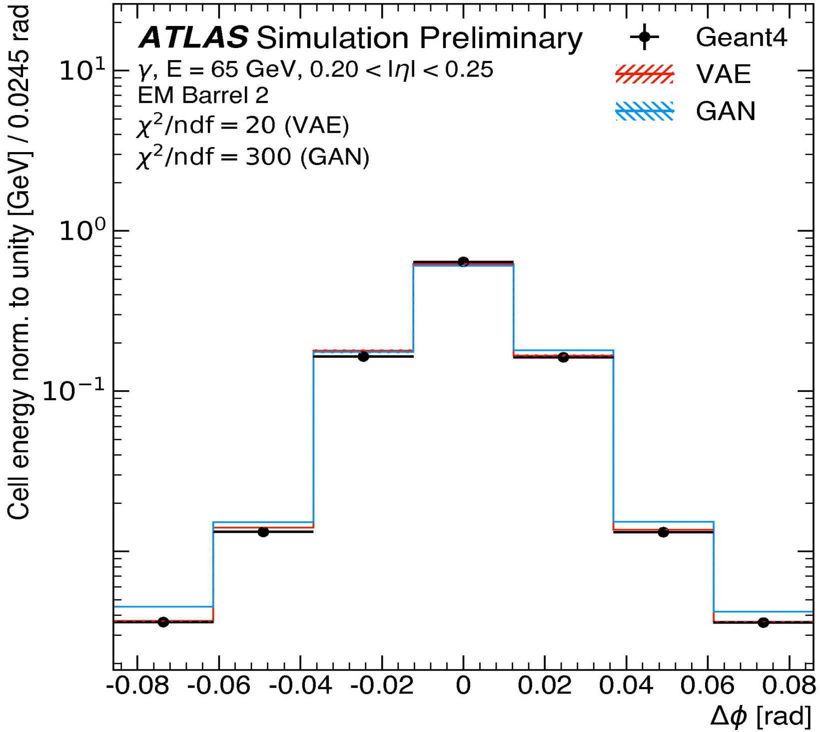


Generation results: Average energy vs Δη , Δϕ

Average energy vs Δη Layer : STRIP



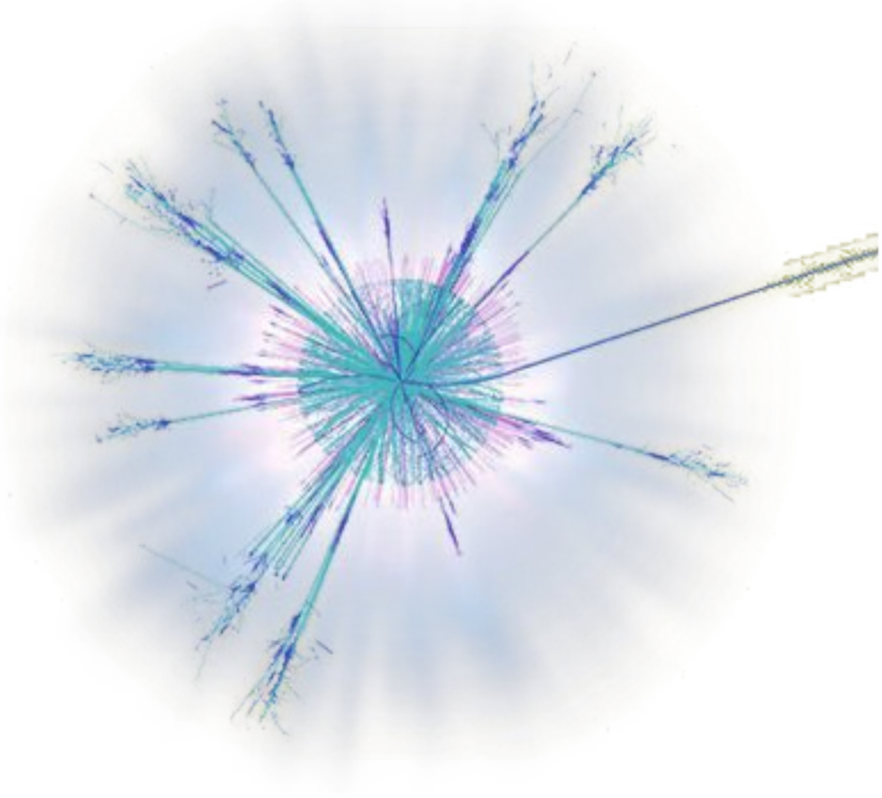
Average energy vs Δϕ Layer : MIDDLE



Conclusion & Outlook

- Fast shower simulation is essential for LHC experiments physics program.
- Proof of concept for generative Deep Learning models for simulating particle showers.
- Promising results and active development towards achieving required accuracy.
- **Outlook**: improve the model to fit a larger class of particle types & pseudorapidity regions.

Thank you



Backup slides

Hyperparameters optimization for VAE

Hyperparameter	Values
Latent space dim.	[1, ... , **10**, ... , 100]
Reco. weight	(0, ... , **1**, ... , 3]
KL weight	(0, ... , $\mathbf{10^{-4}}$, ... , 1]
E_{tot} weight	[0, ... , $\mathbf{10^{-2}}$, ... , 1]
E_i weights	[0, ... , $\mathbf{8 \times 10^{-2}}$, ... , 1] [0, ... , $\mathbf{6 \times 10^{-1}}$, ... , 1] [0, ... , $\mathbf{2 \times 10^{-1}}$, ... , 1] [0, ... , $\mathbf{10^{-1}}$, ... , 1]
Hidden layers (encoder)	1, 2, 3, **4**, 5
Hidden layers (decoder)	1, 2, 3, **4**, 5
Units per layer	[180, ... , **200**, ... , 266] [120, ... , **150**, ... , 180] [80, ... , **100**, ... , 120] [10, ... , **50**, ... , 80]
Activation func.	**ELU**, ReLU, SELU, LeakyReLU, PReLU
Kernel init.	zeros, ones, random normal, random uniform, truncated normal, **variance scaling**, `glorot_normal`
Bias init.	zeros, **ones**, random normal, random uniform, truncated normal, variance scaling, `glorot_normal`
Optimizer	**RMSprop**, Adam, Adagrad, Adadelta, Nadam
Learning rate	[10^{-2}, ... ,$\mathbf{10^{-4}}$, ... , 10^{-6}]
Mini-batch size	50, **100**, 150 , 1000

Hyperparameters optimization for GAN

Hyperparameter	Values
Hidden layers	1, **3**, 5, 10
Units per layer	64, **128**, 512, 1024
Activation func.	SELU + Sigmoid, **LeakyReLU** + {**Sigmoid**, ReLU, Gauss, Sigmoid + ReLU, clipped ReLU, softmax, softmax + ReLU}
Activity `L1_REG_WEIGHT` (Gen.)	0, $\mathbf{10^{-5}}$, 10^{-2}
Kernel init.	`glorot_uniform`, `lecun_normal`
Gradient penalty	one-sided, **two-sided**
Gradient penalty weight	0, **10**, 20
Training ratio	20, 10, **5**, 3, 1
Learning rate	$\mathbf{5 \times 10^{-5}}$, 5×10^{-6}, 1×10^{-6} (training ratio 5) 5×10^{-5}, 5×10^{-6}, 1×10^{-5}, 1×10^{-7} (training ratio 3) 1×10^{-6} (training ratio 1)
Mini-batch size	**64**, 1024
Preprocessing (all norm. to E_γ)	$\log_{10} E_{\text{cell}}$, $\log_{10}(E_{\text{cell}} \times 10^{10})$, $E_{\mathbf{cell}}$
Conditioning	$\{E_\gamma, \mathbf{log_{10}} E_\gamma\}$ + **multi-hot encoding of cell alignments**