

Fast Detector Simulations with ML

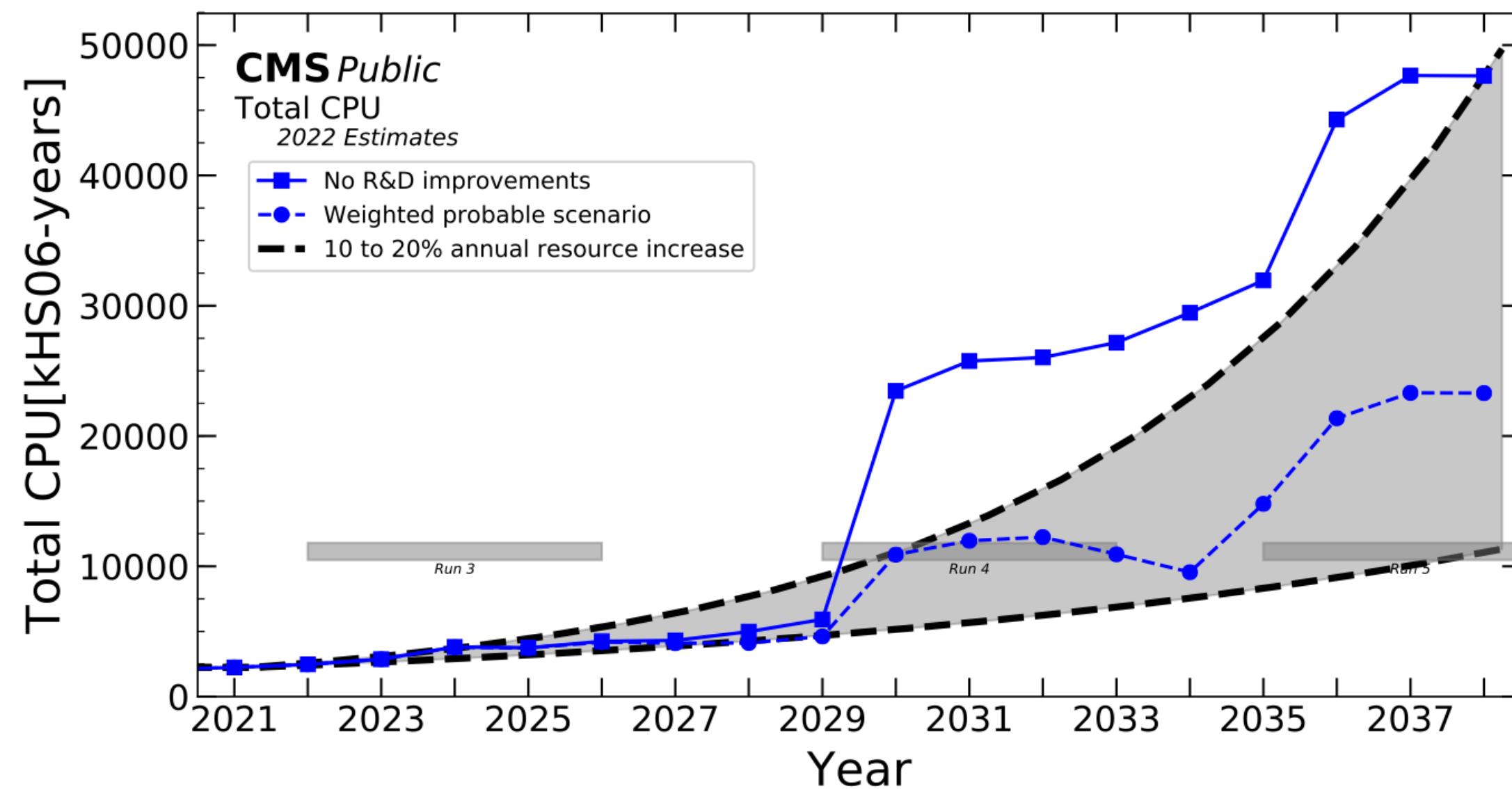
AI4EIC Workshop

David Shih
October 11, 2022



Fast ML for Surrogate Modeling

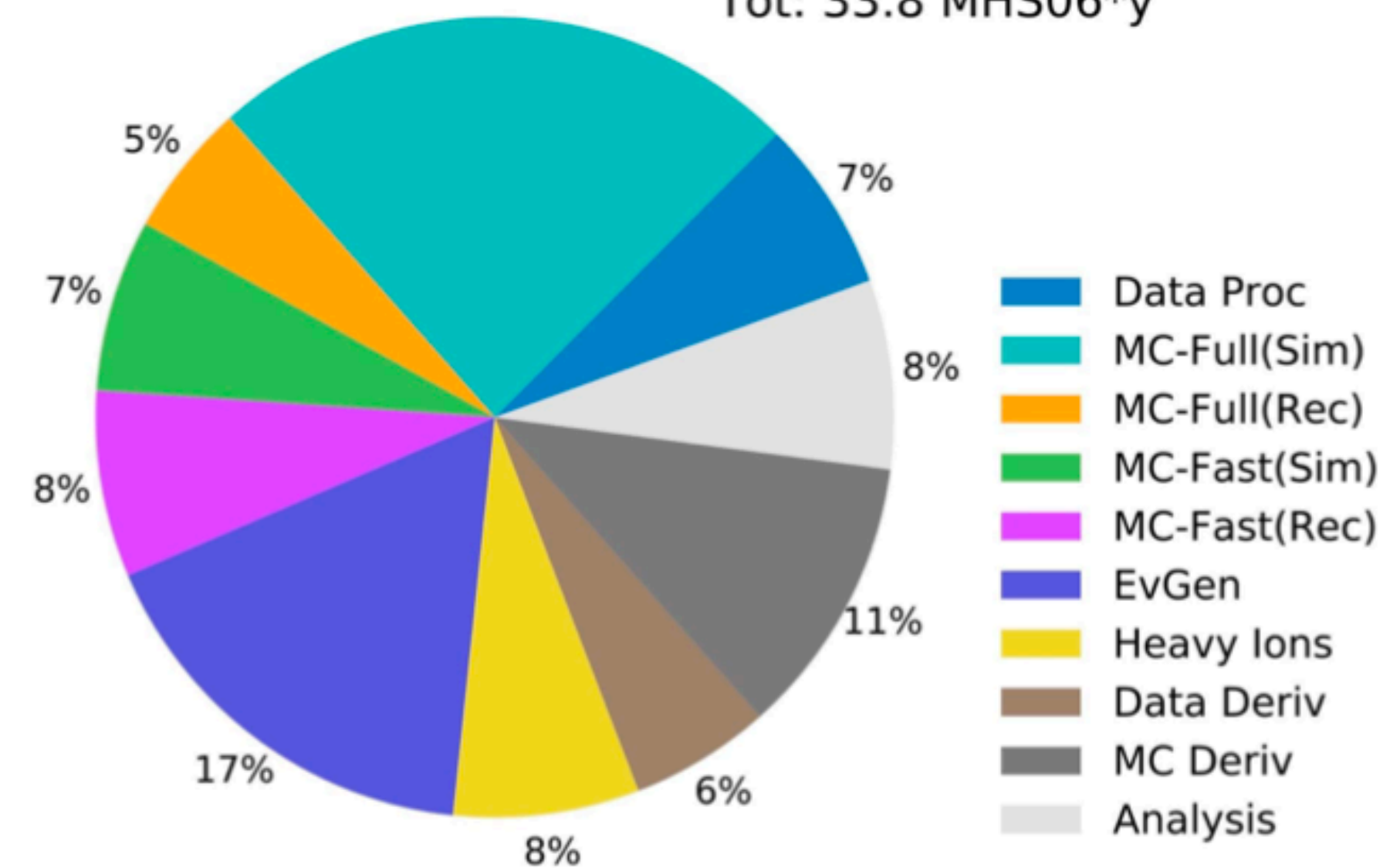
<https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults>



ATLAS Preliminary

2022 Computing Model - CPU: 2031, Conservative R&D

Tot: 33.8 MHS06*y



CERN-LHCC-2022-005

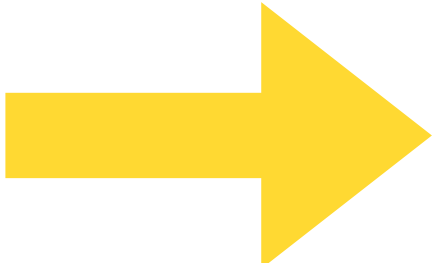
Detector simulation (GEANT4) and event generation (MG5, Pythia, Herwig, ...) are major — and growing — bottlenecks at LHC and other experiments

Fast ML for Surrogate Modeling



Fast ML for Surrogate Modeling

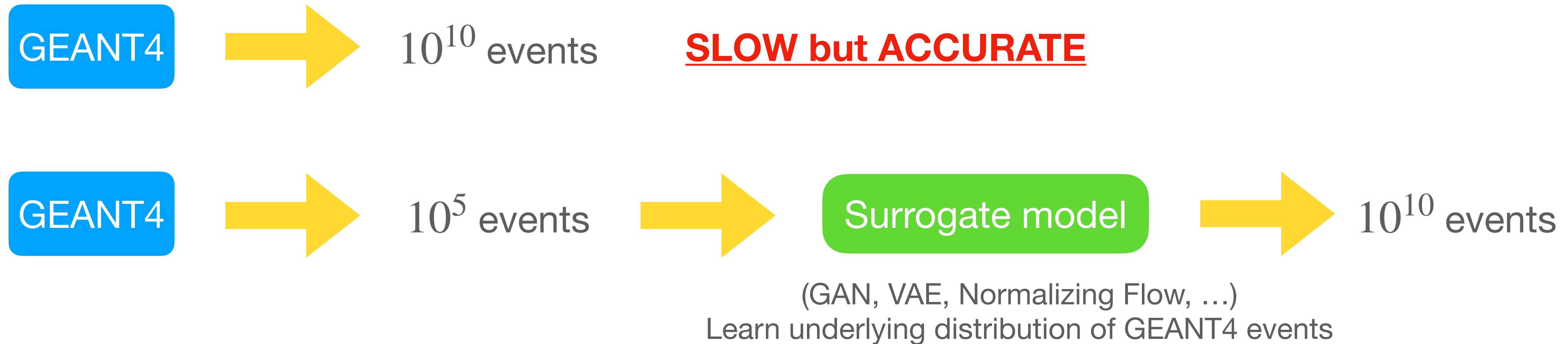
GEANT4  10^{10} events **SLOW but ACCURATE**

GEANT4  10^5 events

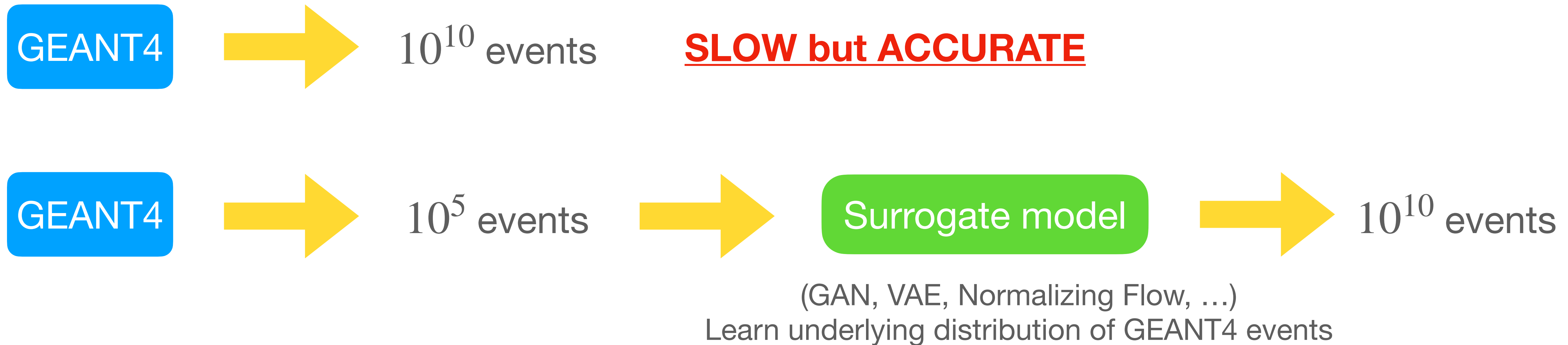
Fast ML for Surrogate Modeling



Fast ML for Surrogate Modeling



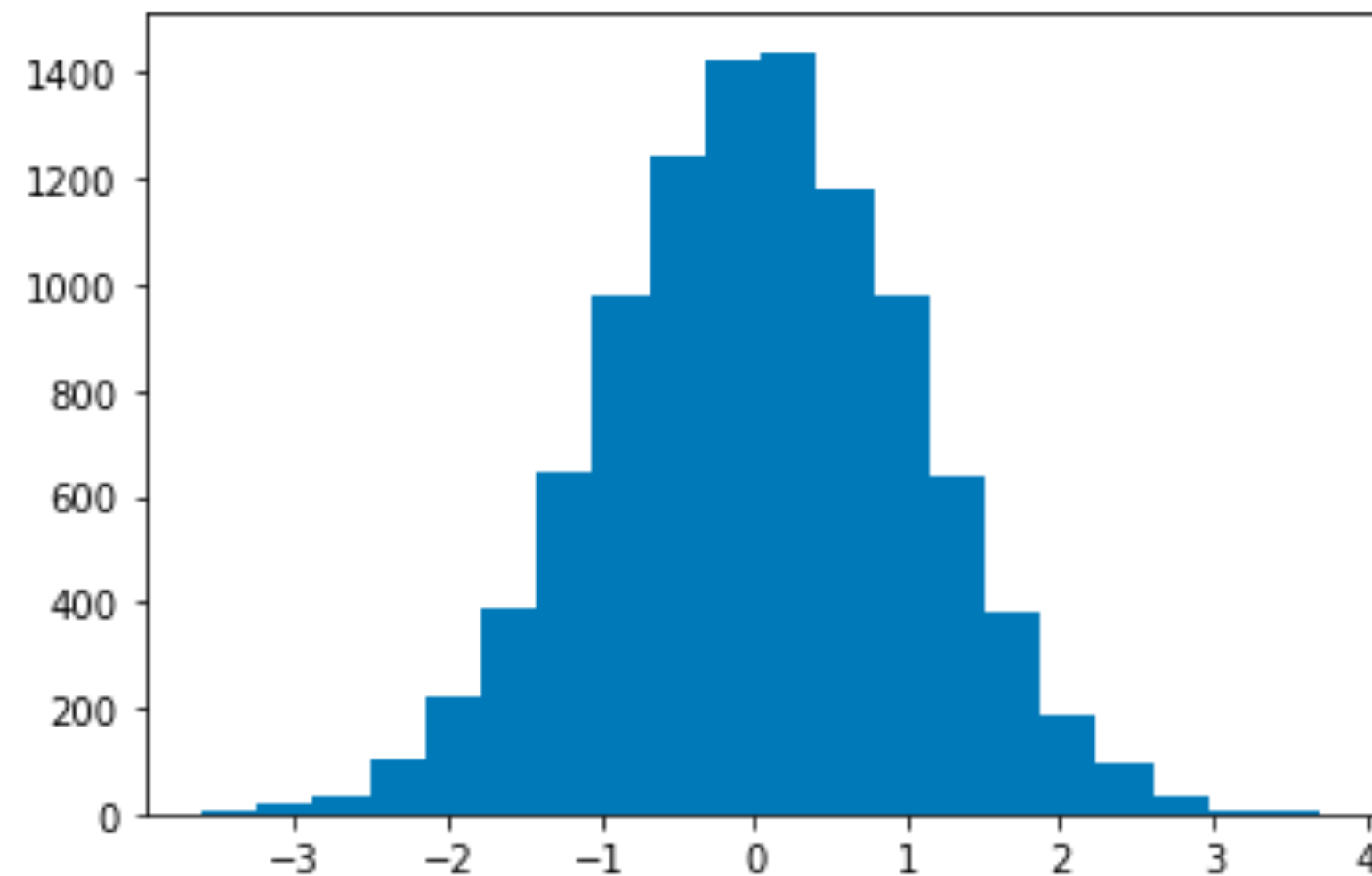
Fast ML for Surrogate Modeling



FAST and ACCURATE?

Surrogate Model = Generative Model

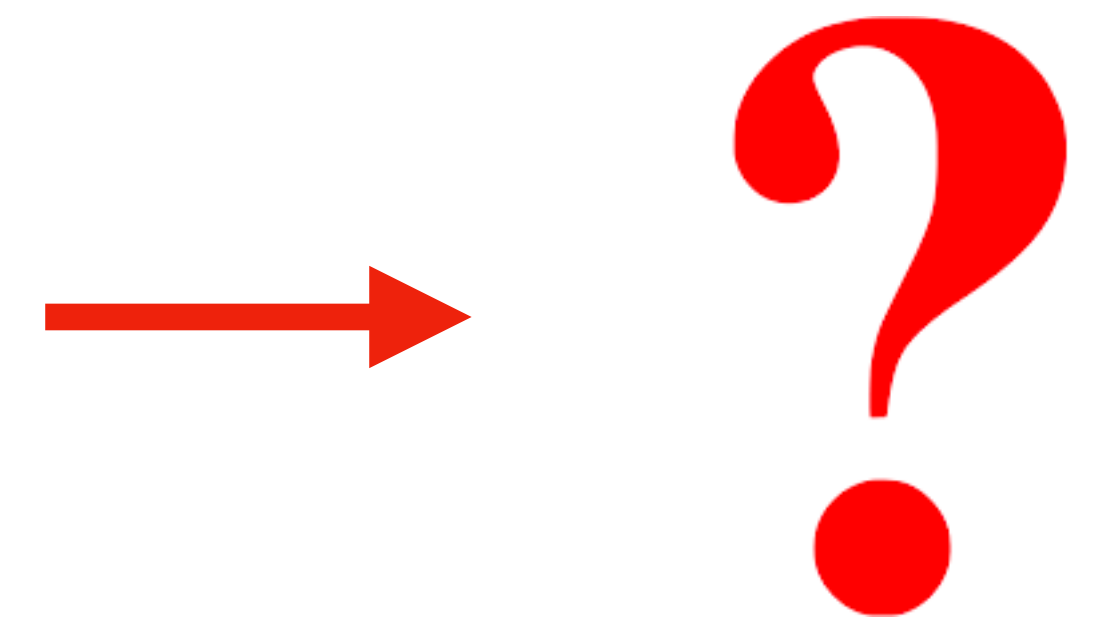
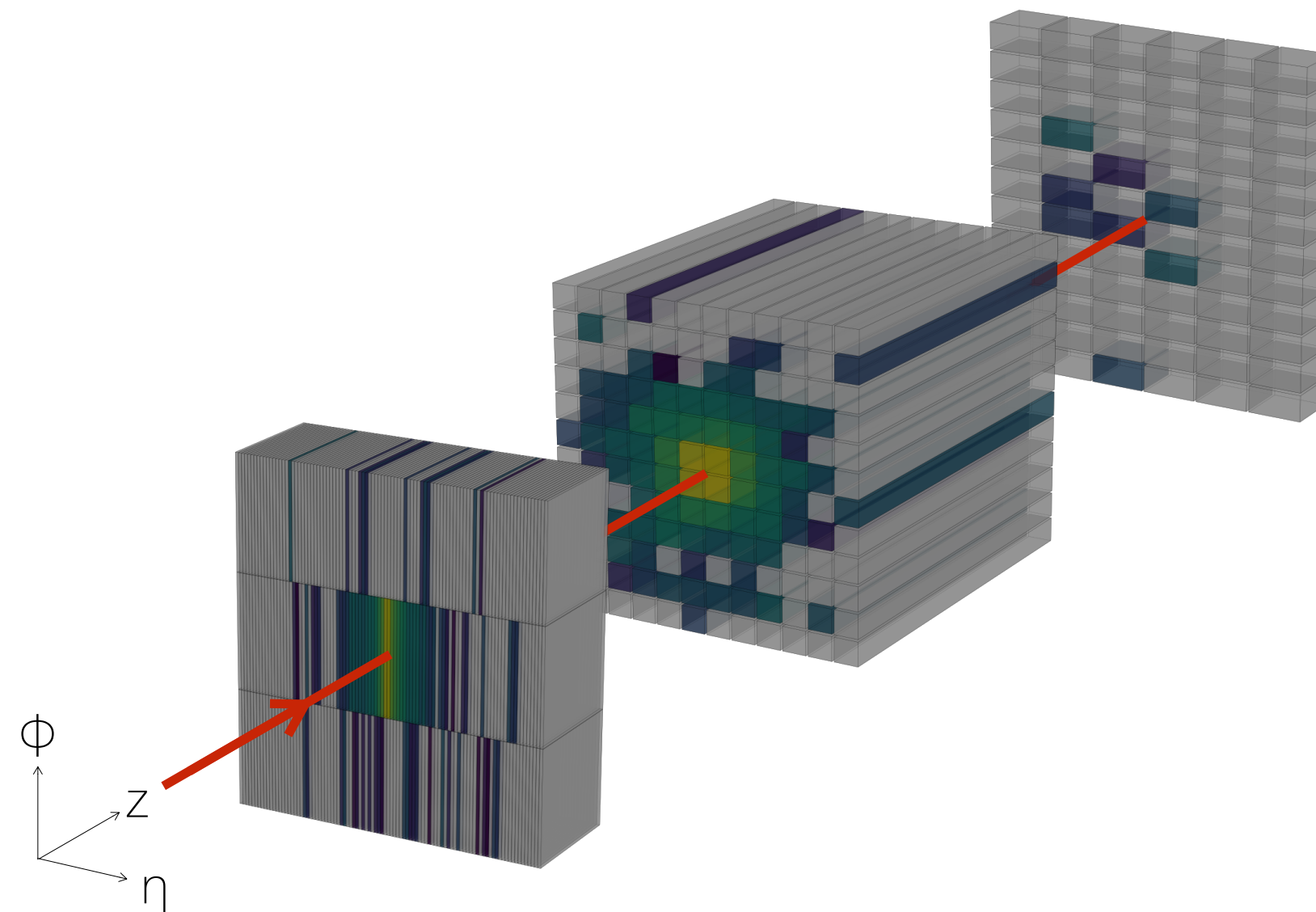
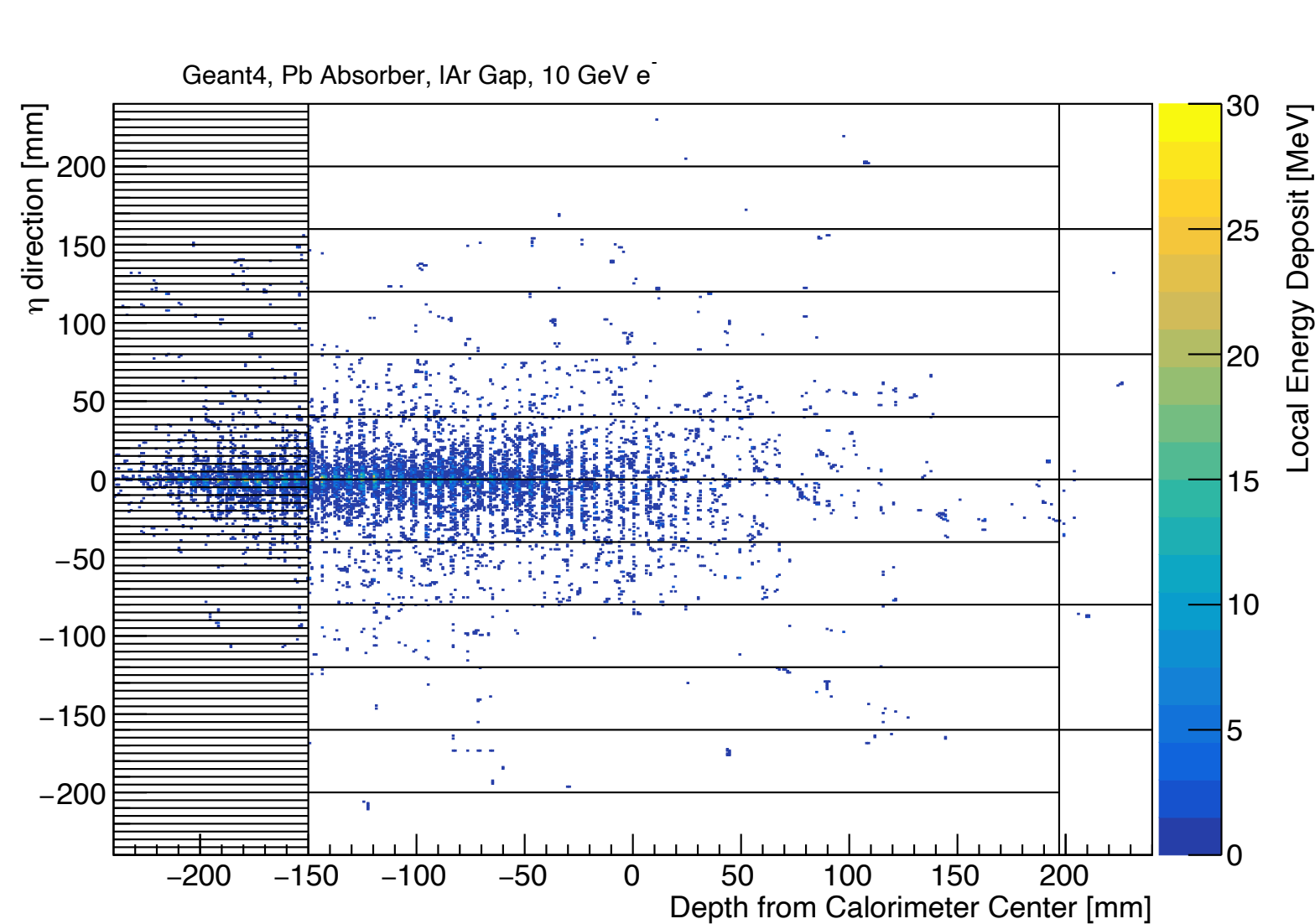
Generative modeling: general class of ML methods, aims to learn underlying $p(x)$ of data and then sample from it



$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Surrogate Model = Generative Model

How to build a generative model for very high-dimensional feature spaces?



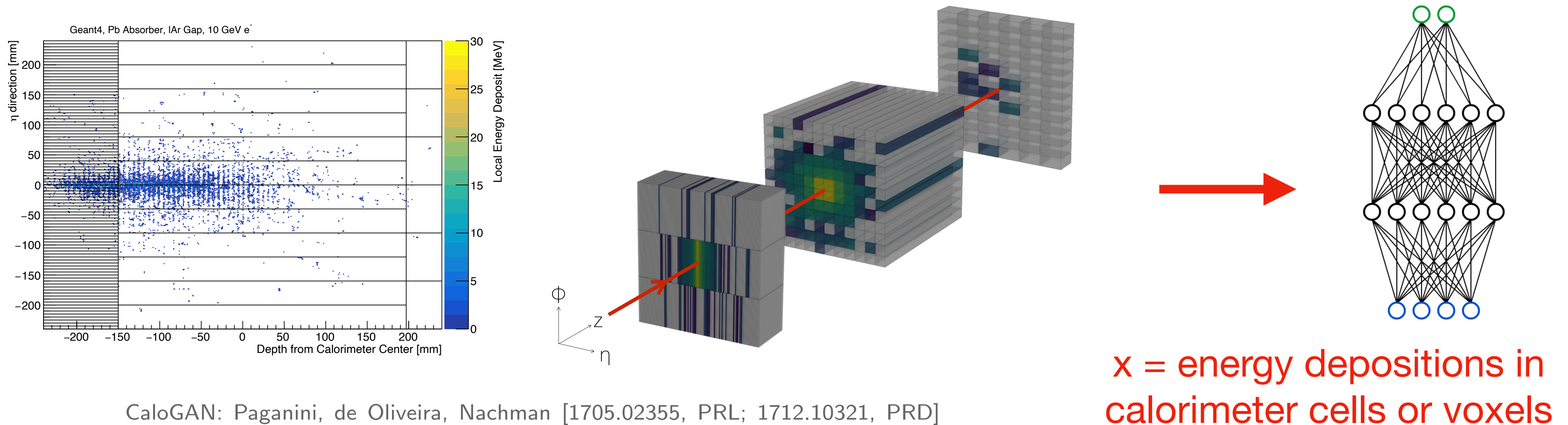
CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

x = energy depositions in
calorimeter cells or voxels

$\dim x = 500, 5000, 50000, \dots$

Surrogate Model = Generative Model

How to build a generative model for very high-dimensional feature spaces?

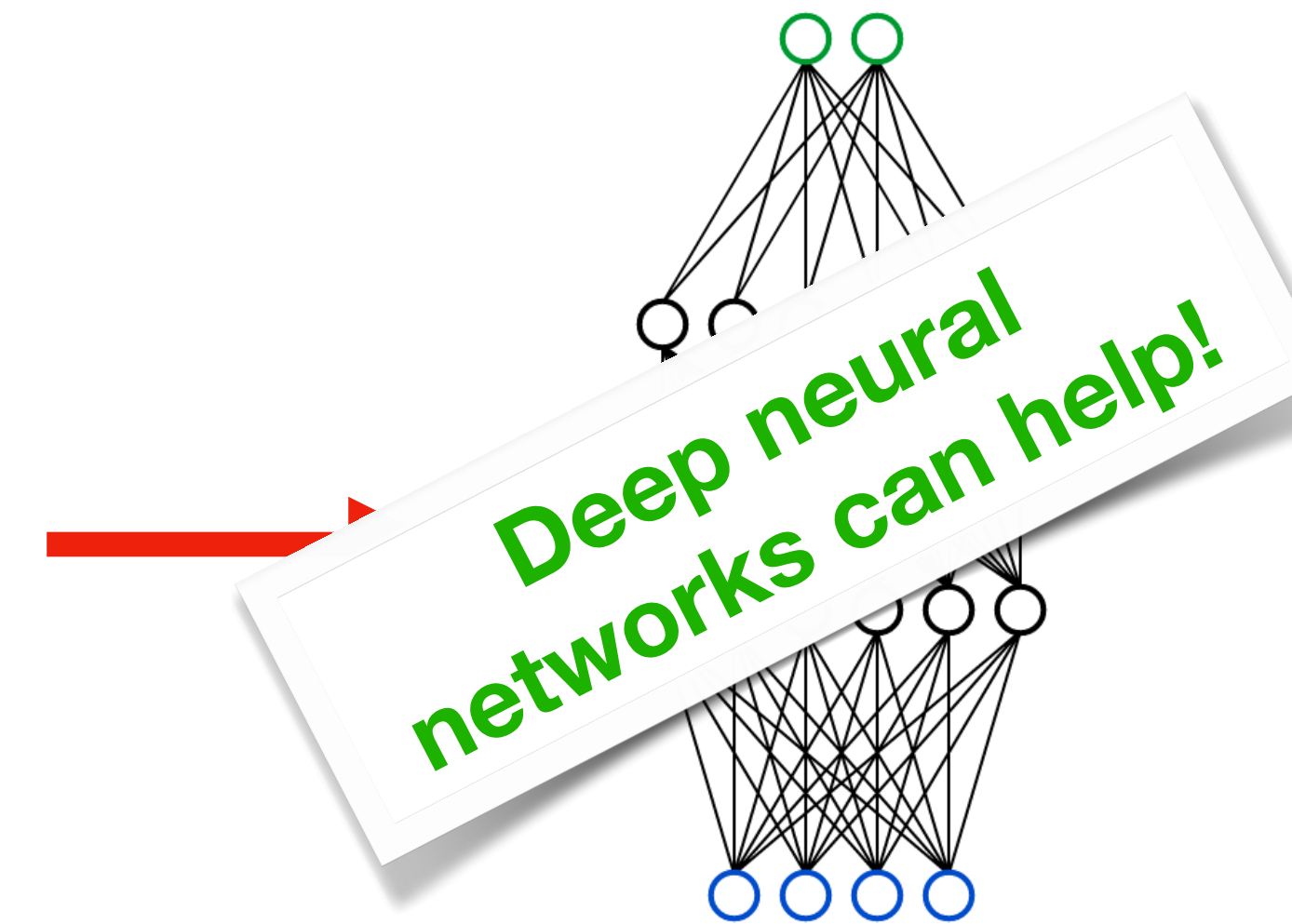
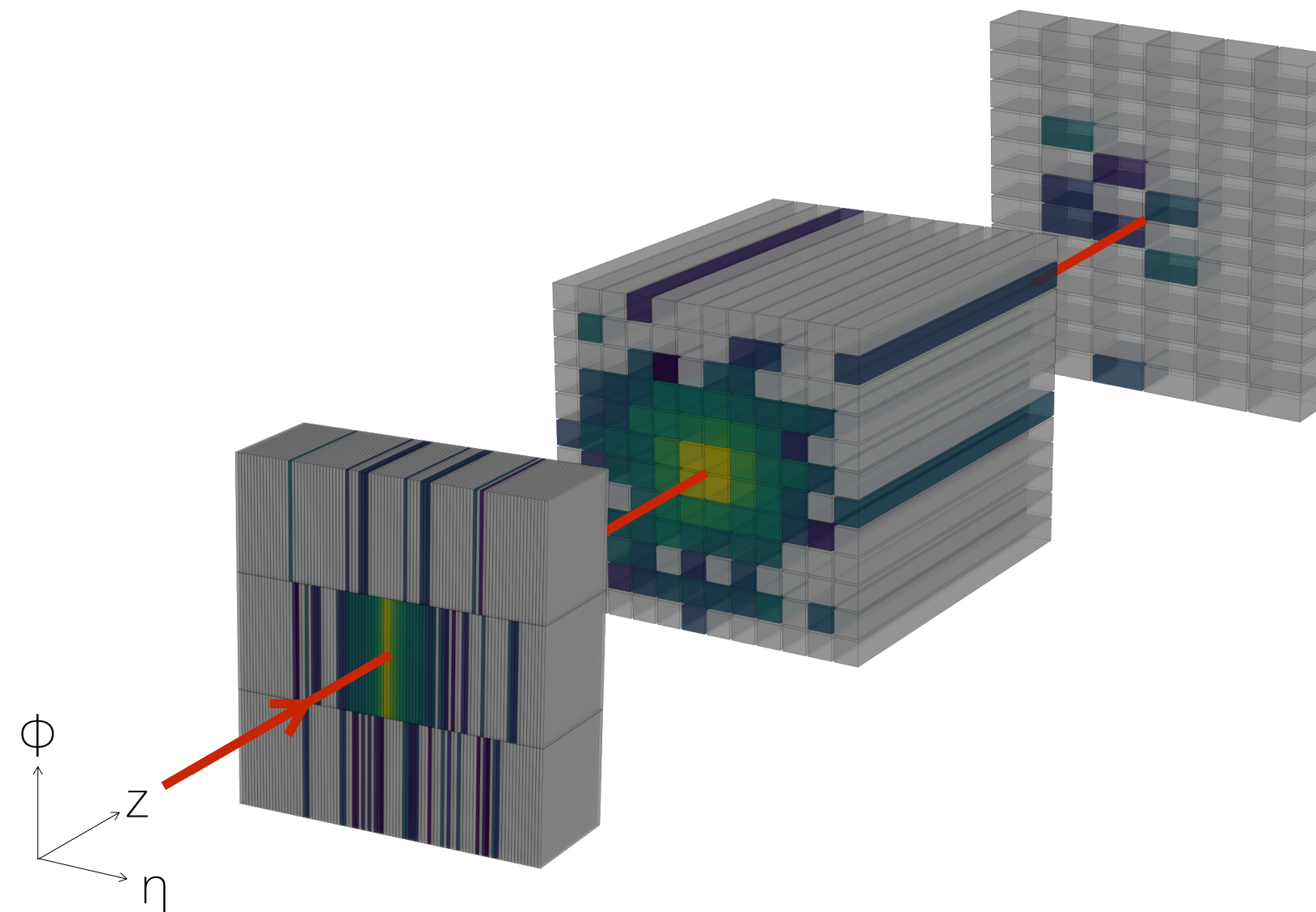
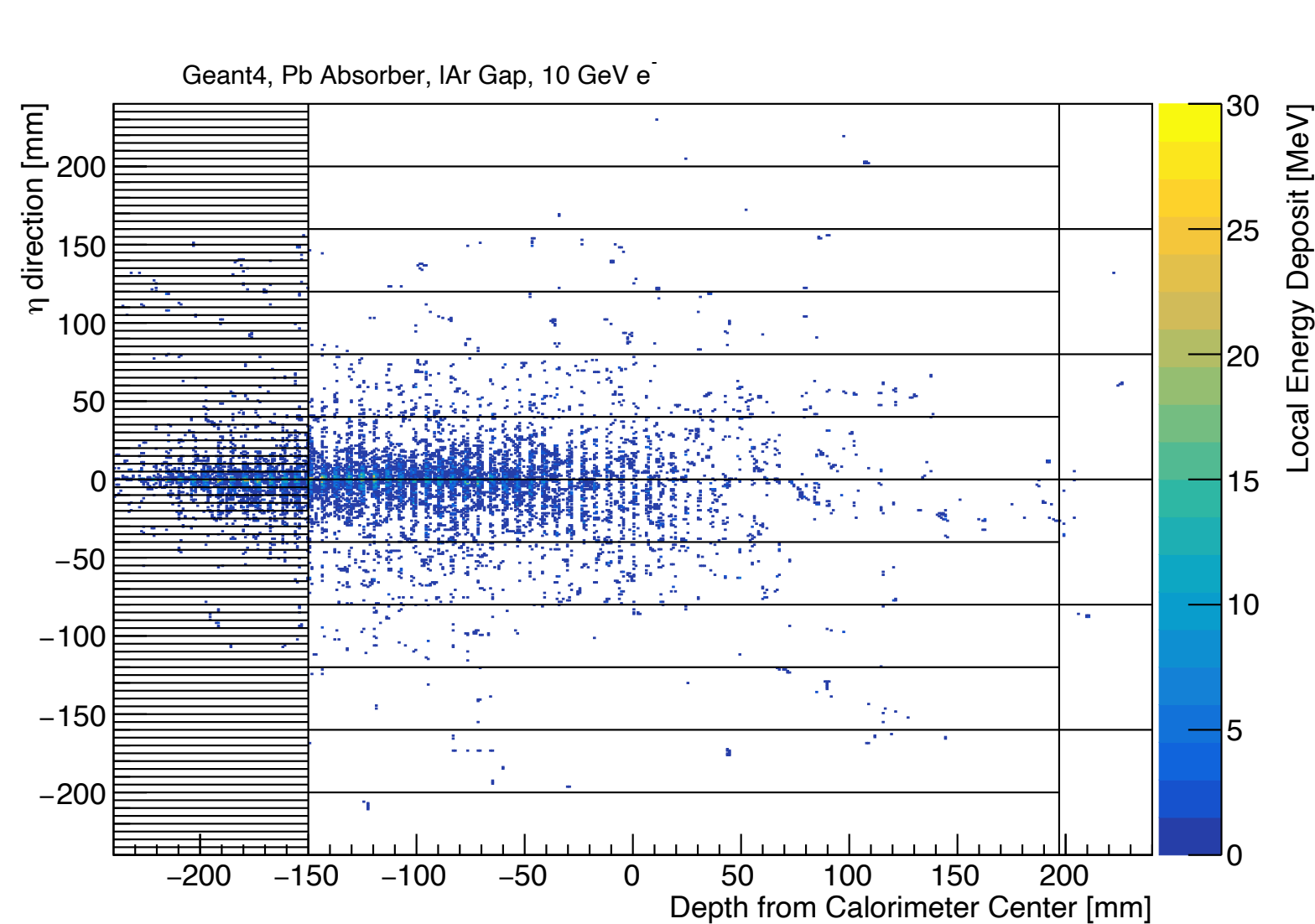


CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

dim x = 500, 5000, 50000, ... 5

Surrogate Model = Generative Model

How to build a generative model for very high-dimensional feature spaces?



CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

x = energy depositions in
calorimeter cells or voxels

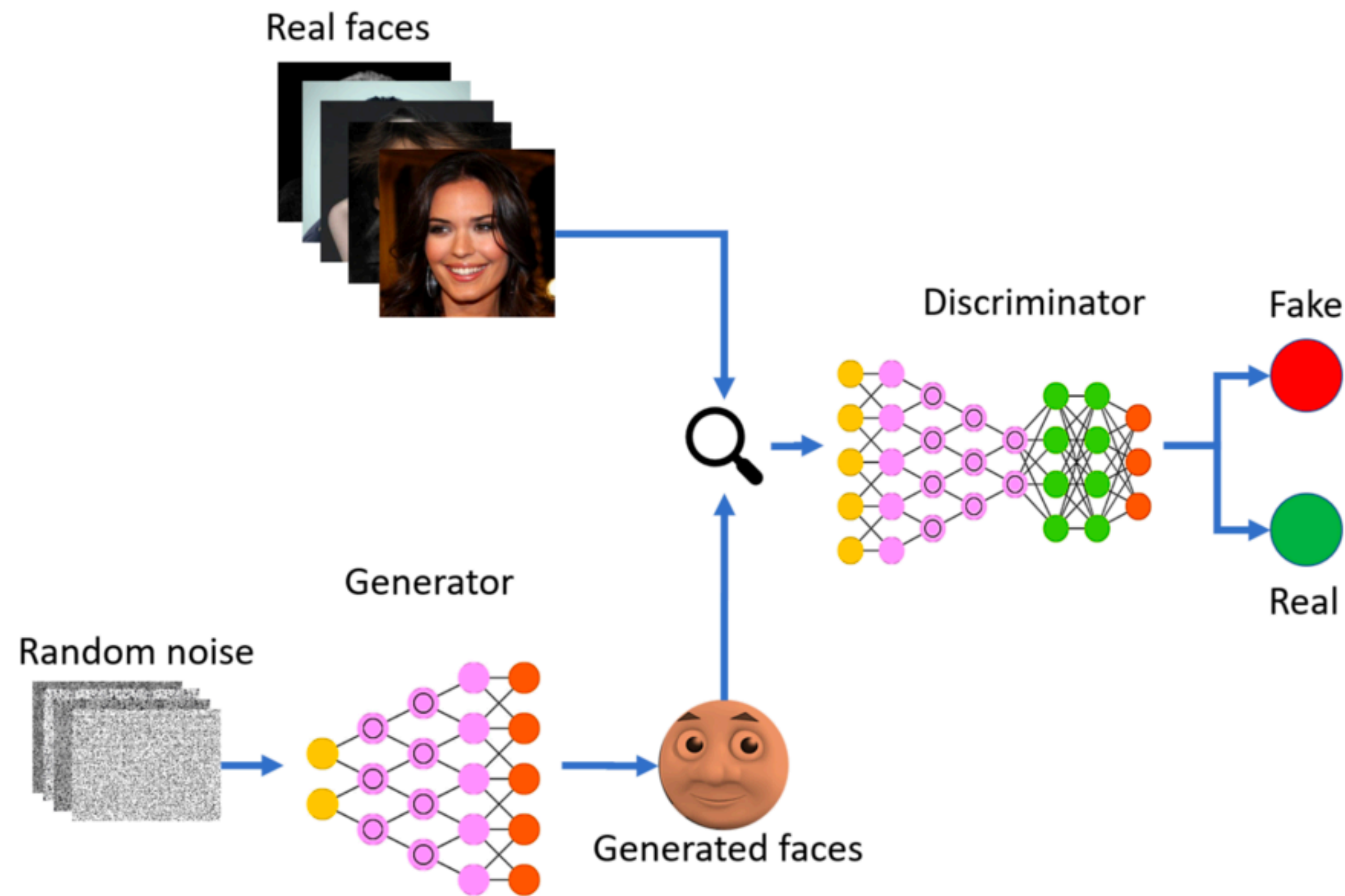
$\dim x = 500, 5000, 50000, \dots$

Deep generative models

- Learn $p(x)$ **implicitly**
 - Generative Adversarial Networks (GANs)
 - Variational Autoencoders (VAEs)
- Learn $p(x)$ **explicitly**
 - Normalizing Flows (NFs)

GANs

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



Idea: train generator and discriminator simultaneously with min-max objective (saddle point)

- Generator tries to fool discriminator
- Discriminator tries to tag generator

Pros:

- capable of generating very realistic images

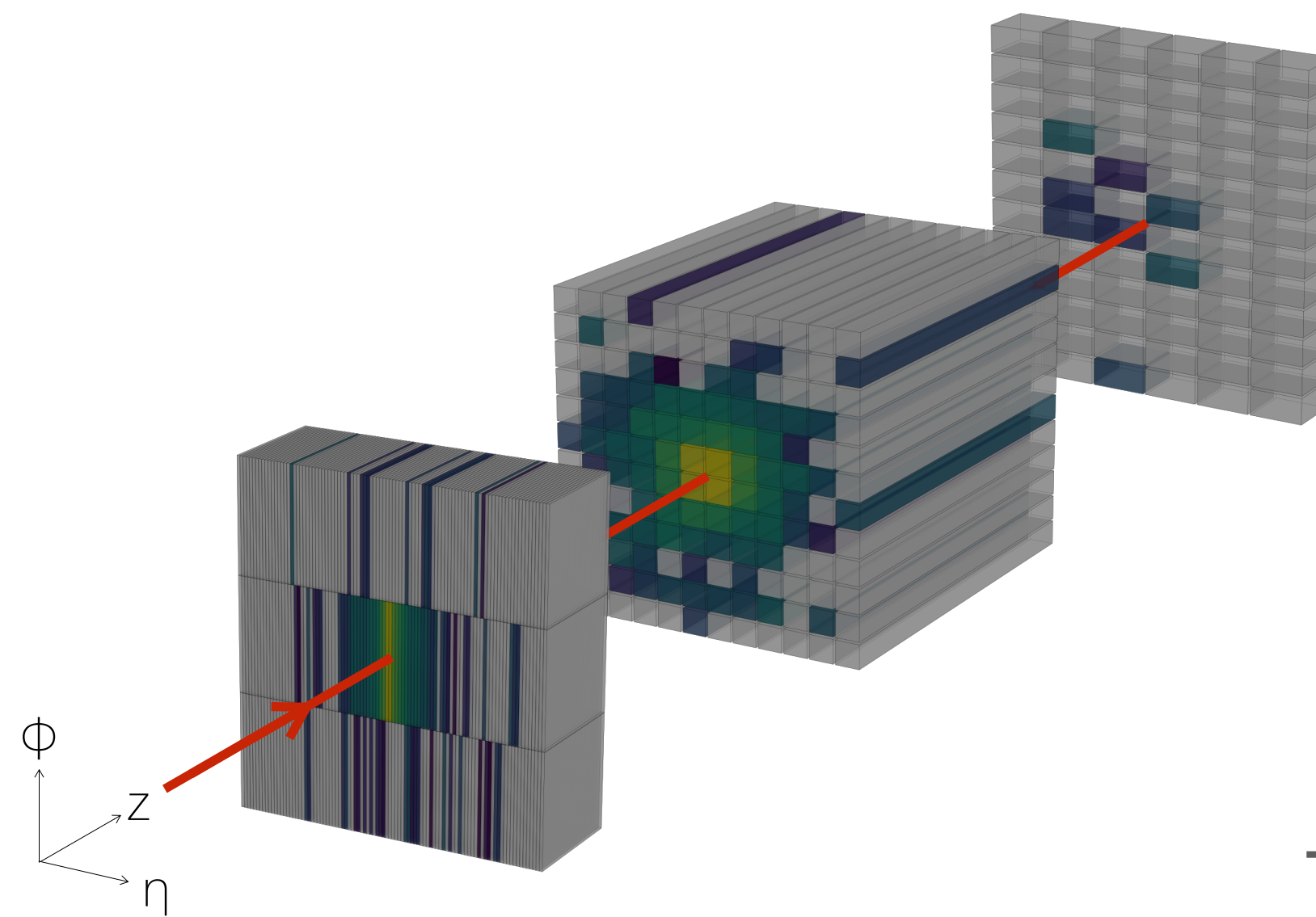
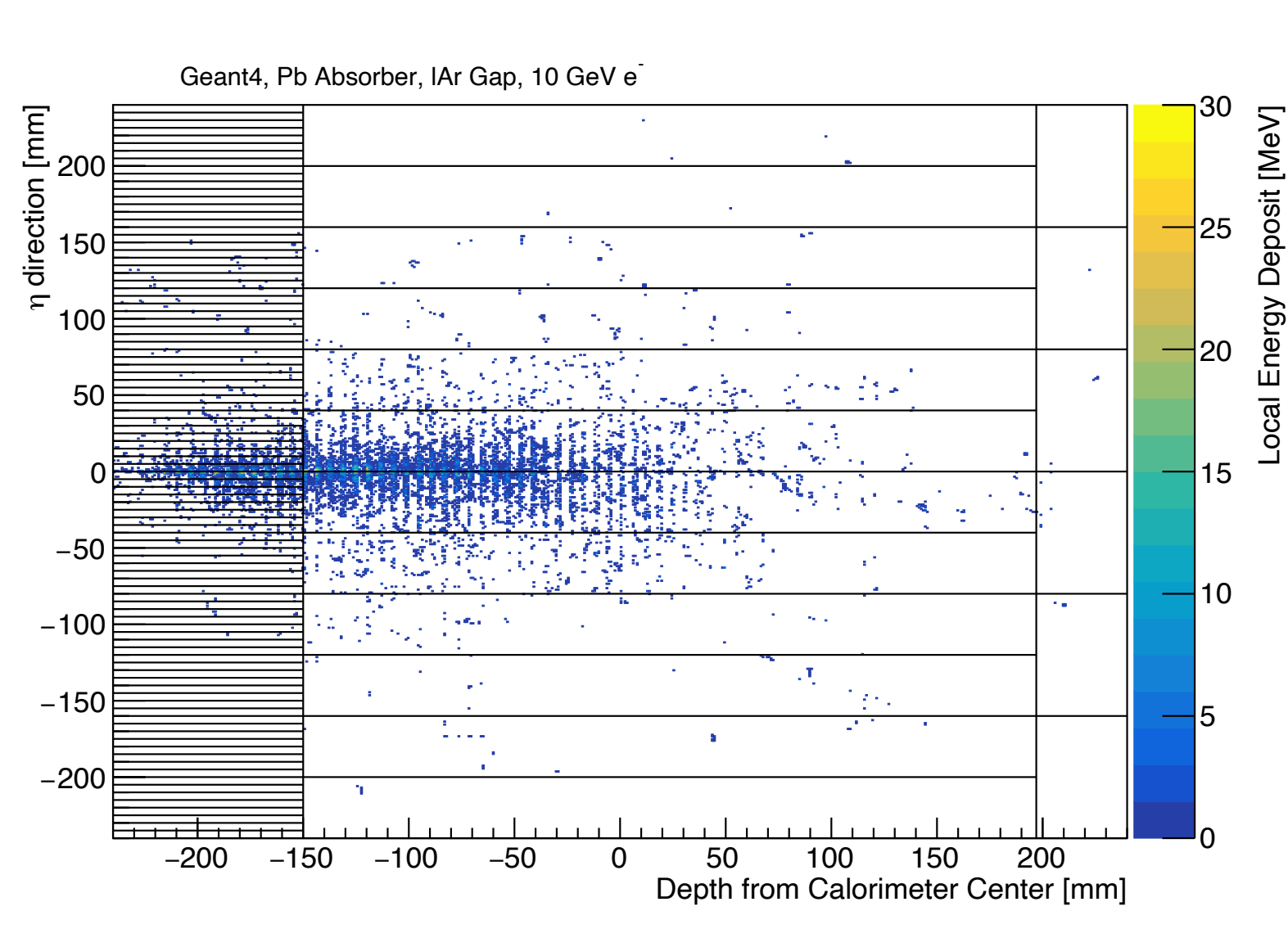
Cons:

- unstable training
- mode collapse
- difficult model selection

<https://medium.com/sigmoid/a-brief-introduction-to-gans-and-how-to-code-them-2620ee465c30>

CaloGAN

First ever application of deep generative modeling to fast calorimeter simulation



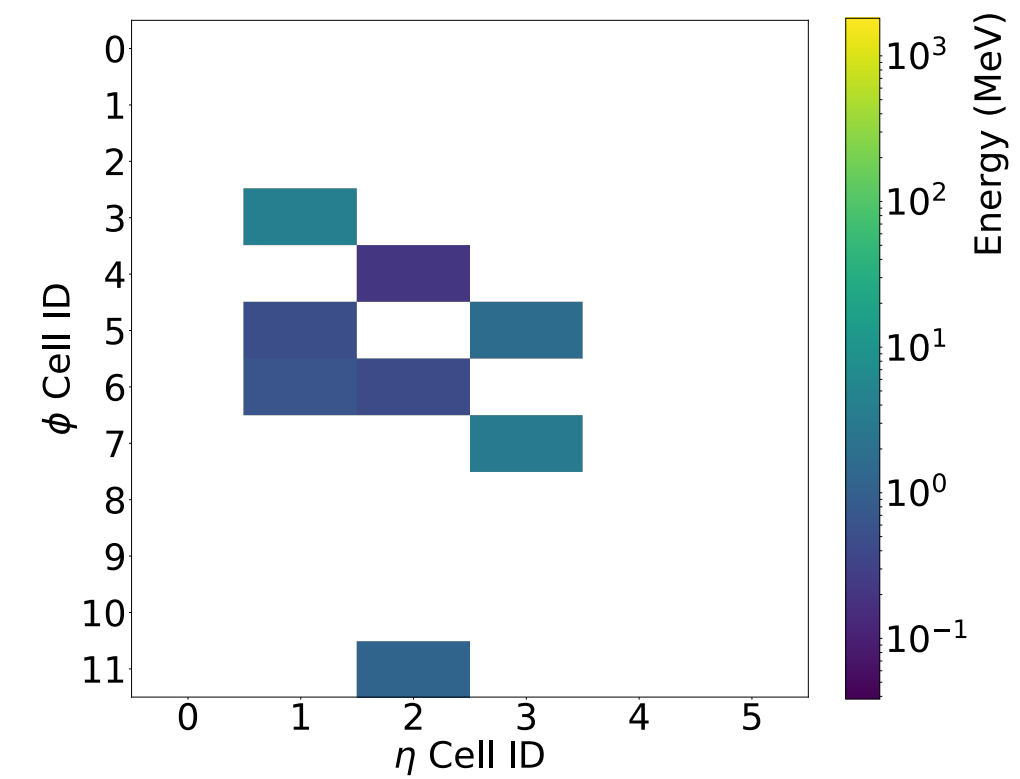
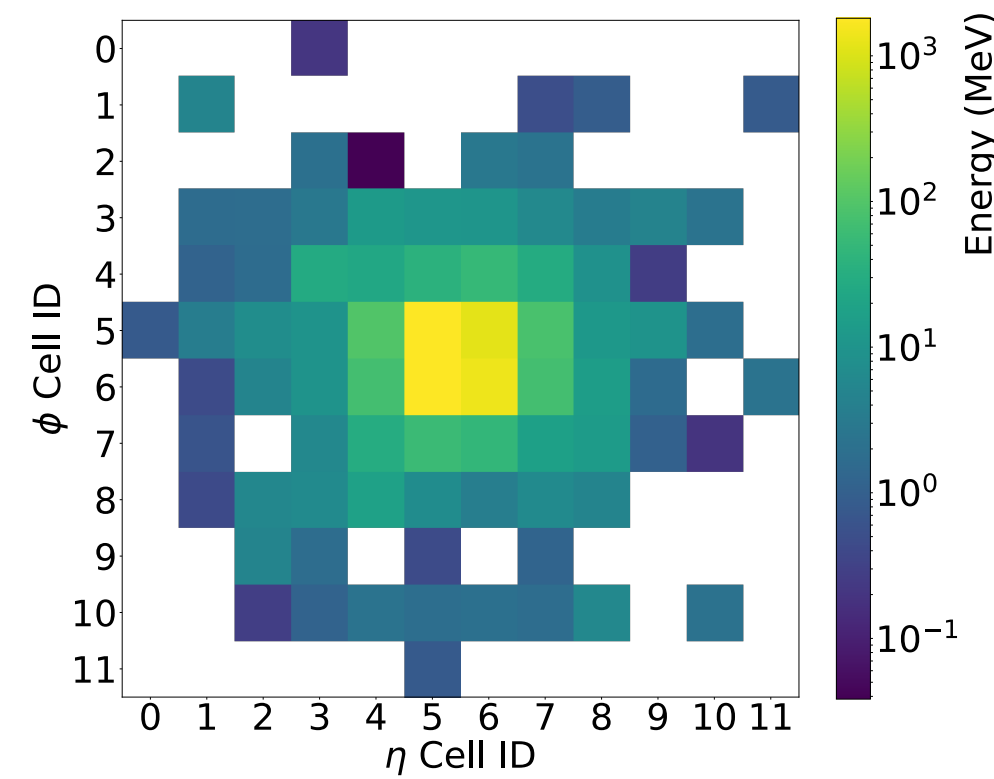
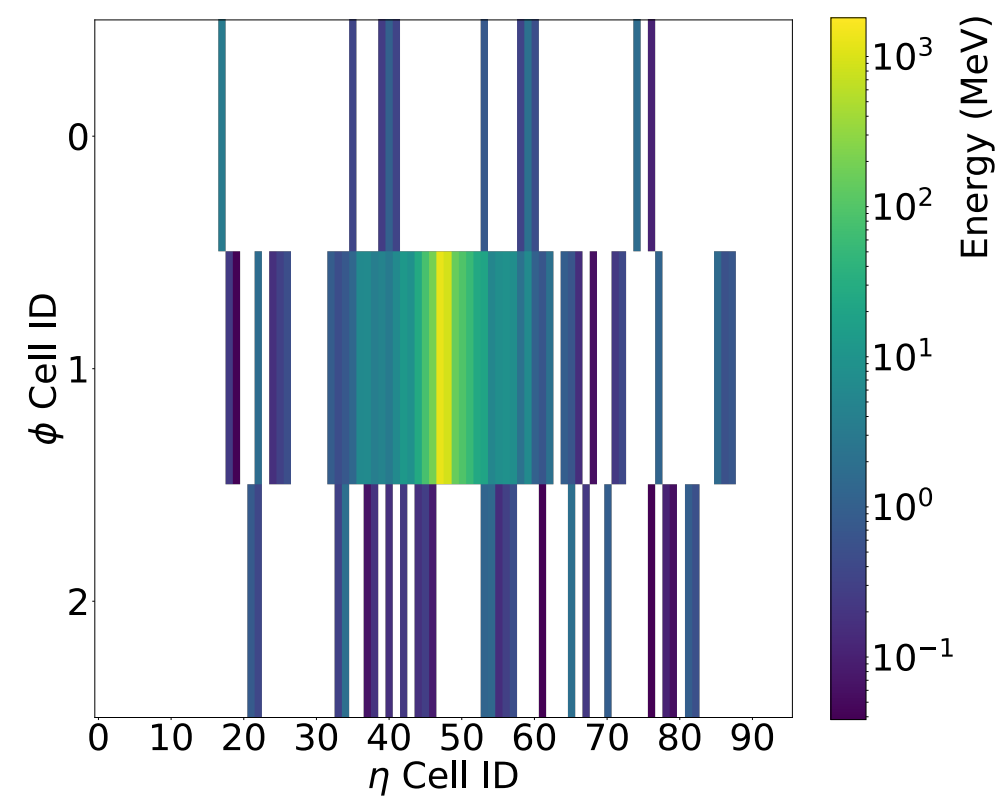
Toy calorimeter based on ATLAS
ECAL

CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

3 LAr layers: $3 \times 96 + 12 \times 12 + 12 \times 6 =$
504 voxels

CaloGAN

- e^+ , γ , π^+ GEANT4 showers (100k each)
- incident E uniformly sampled 1-100 GeV
- goal: train generative model on the showers and learn $p(\vec{E}_{\text{voxels}} | E_{\text{inc}})$



CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

CaloGAN

From [1712.10321](#)

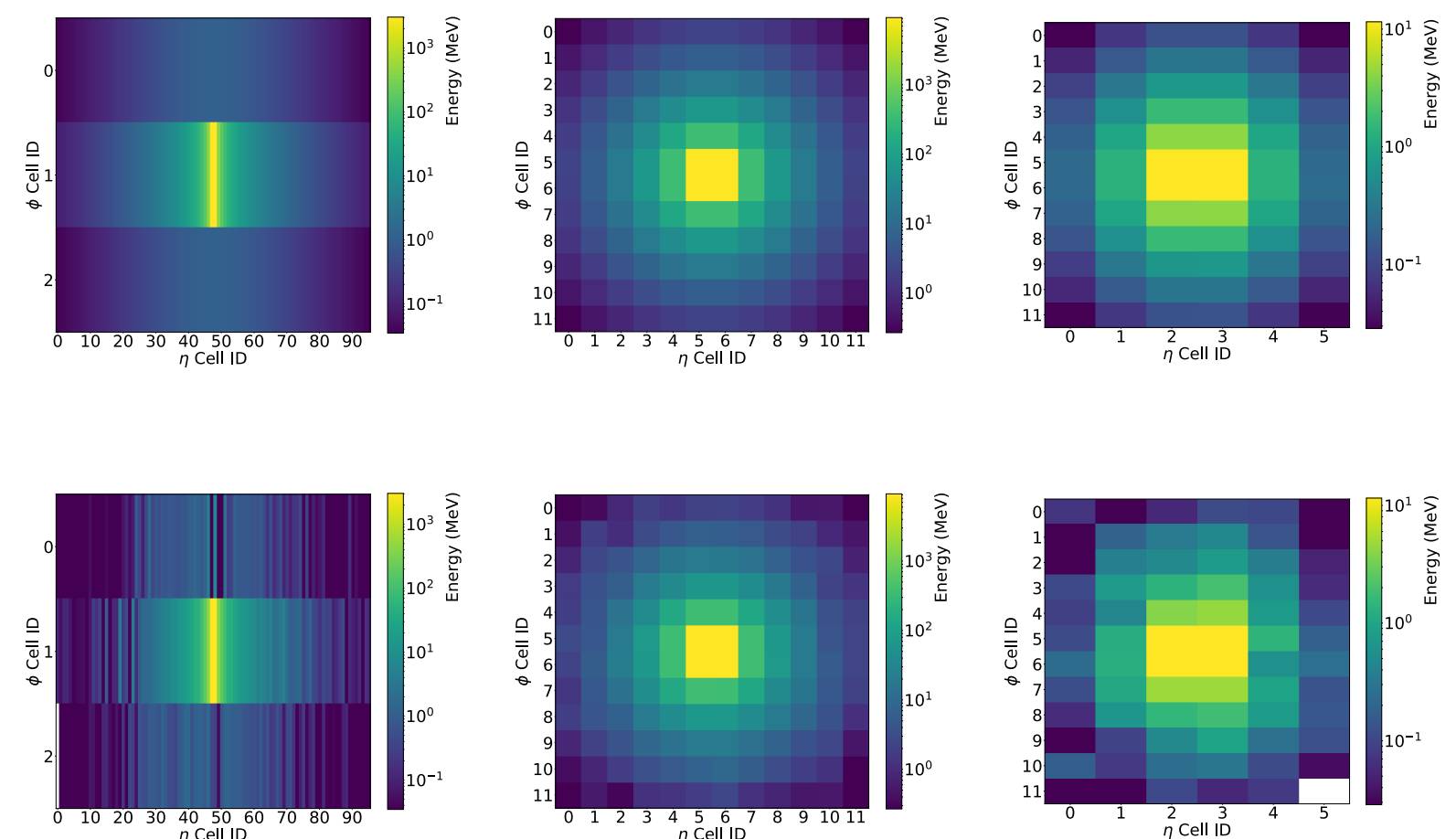


FIG. 6: Average e^+ GEANT4 shower (top), and average e^+ CALOGAN shower (bottom), with progressive calorimeter depth (left to right).

Impressive results!

But also some big discrepancies

Speedup:

GEANT4: 2000 ms/shower (CPU)

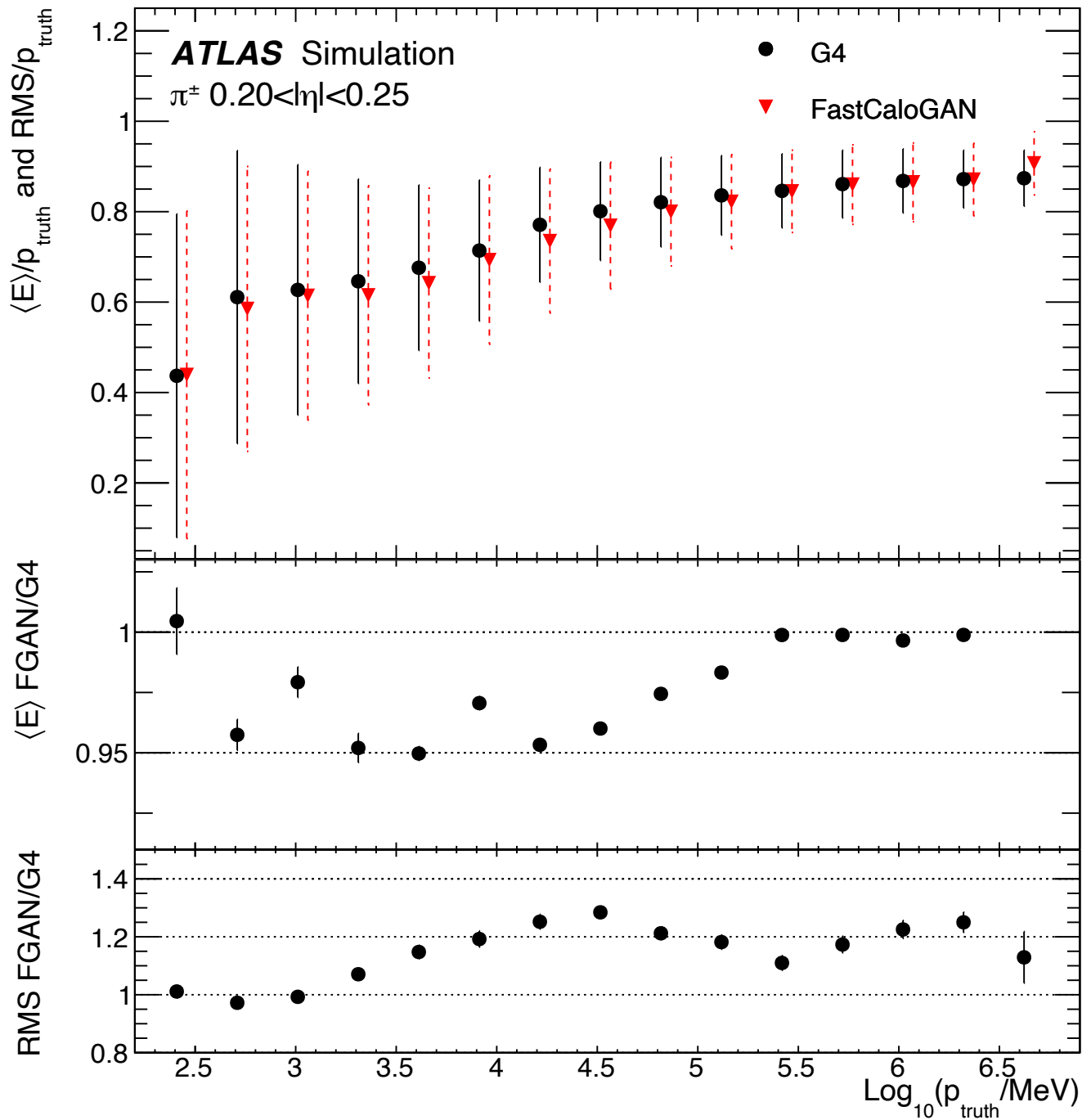
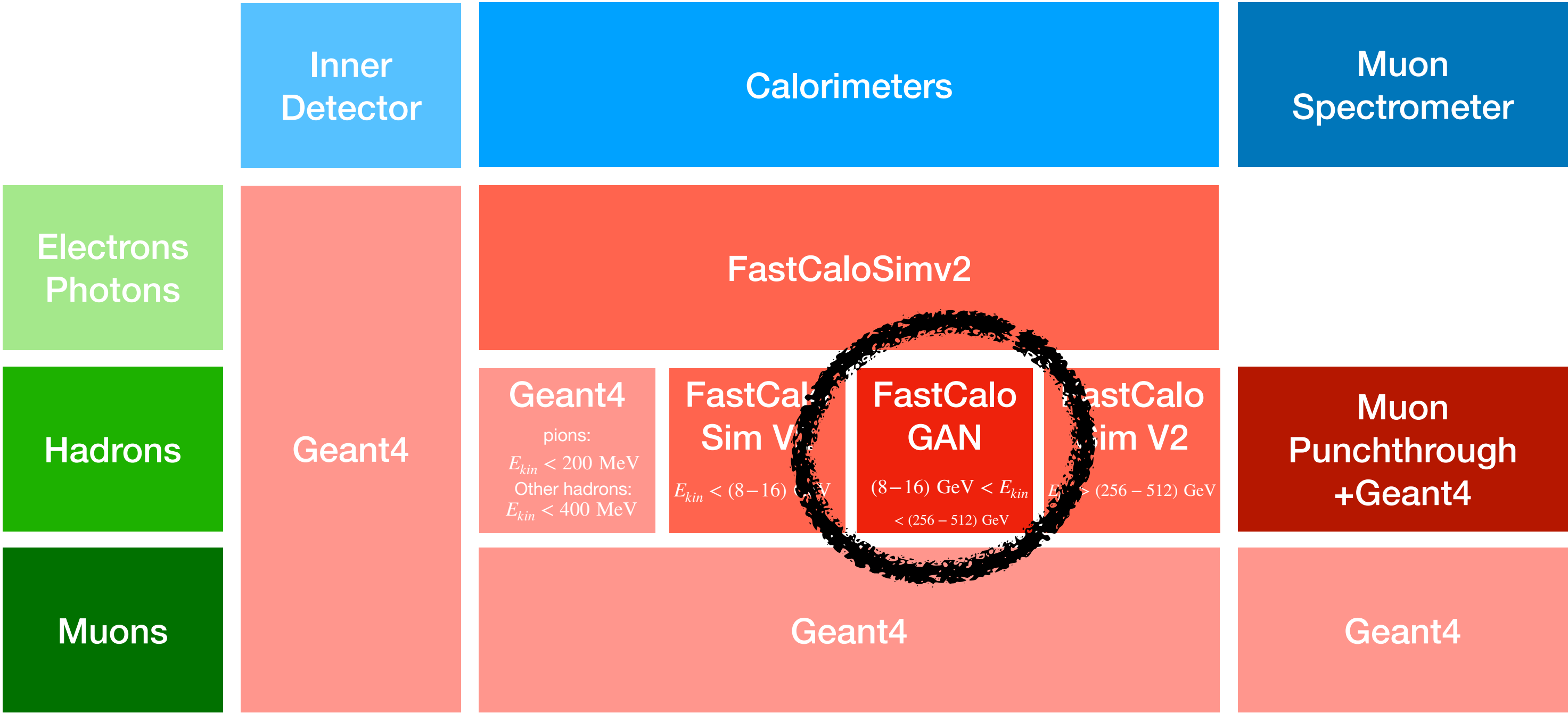
CaloGAN: 0.01 ms/shower (GPU)



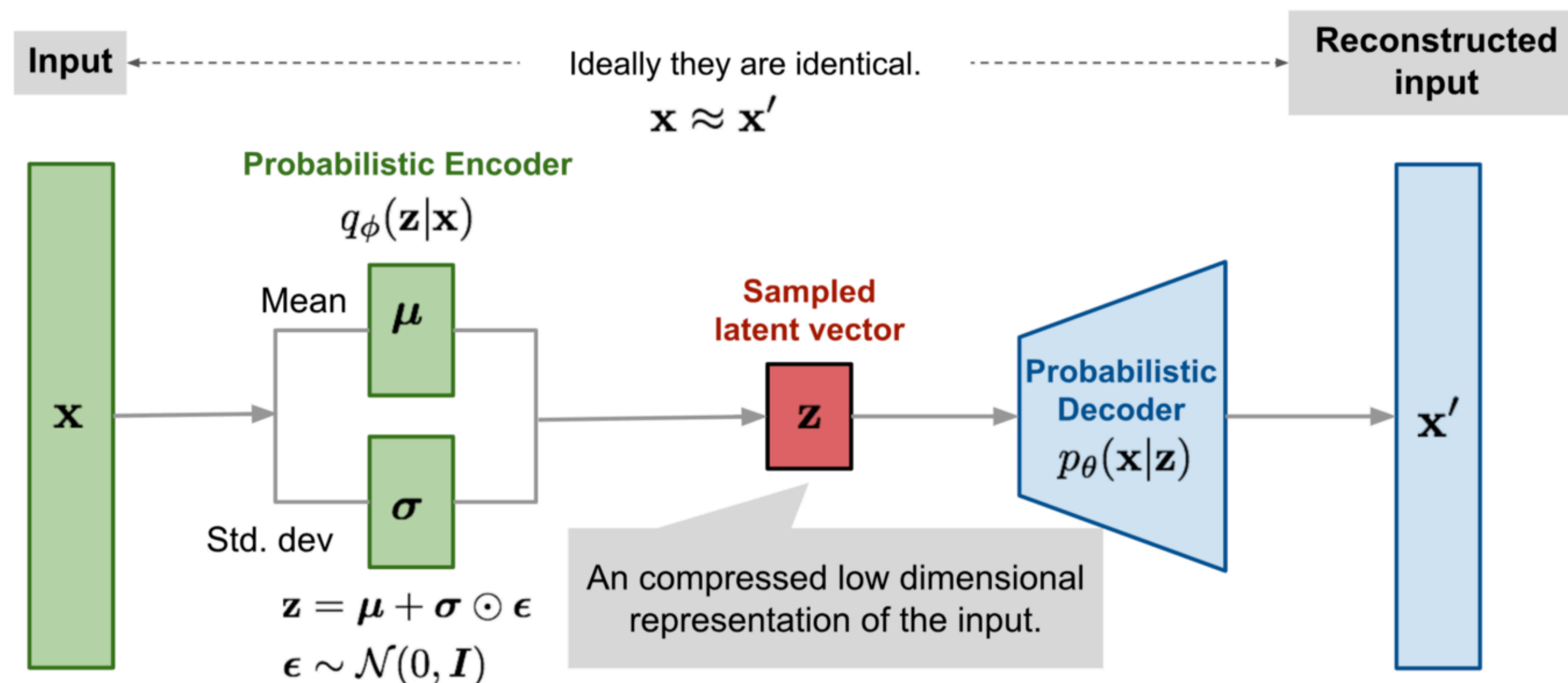
AtlFast3

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D \left(\mathbf{x}^{(i)} \right) + \log \left(1 - D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(D \left(G \left(\mathbf{z}^{(i)} \right) \right) \right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^m \left[f \left(\mathbf{x}^{(i)} \right) - f \left(G \left(\mathbf{z}^{(i)} \right) \right) \right]$	$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m f \left(G \left(\mathbf{z}^{(i)} \right) \right)$

- ATLAS now uses a (W)GAN for a tiny portion of its new FastSim [2109.02551]



Variational Autoencoders



<https://towardsdatascience.com/an-introduction-to-variational-auto-encoders-vaes-803ddfb623df>

Idea: train a probabilistic autoencoder to compress data into regularized latent space

Pros:

- Much easier to train than GANs

Cons:

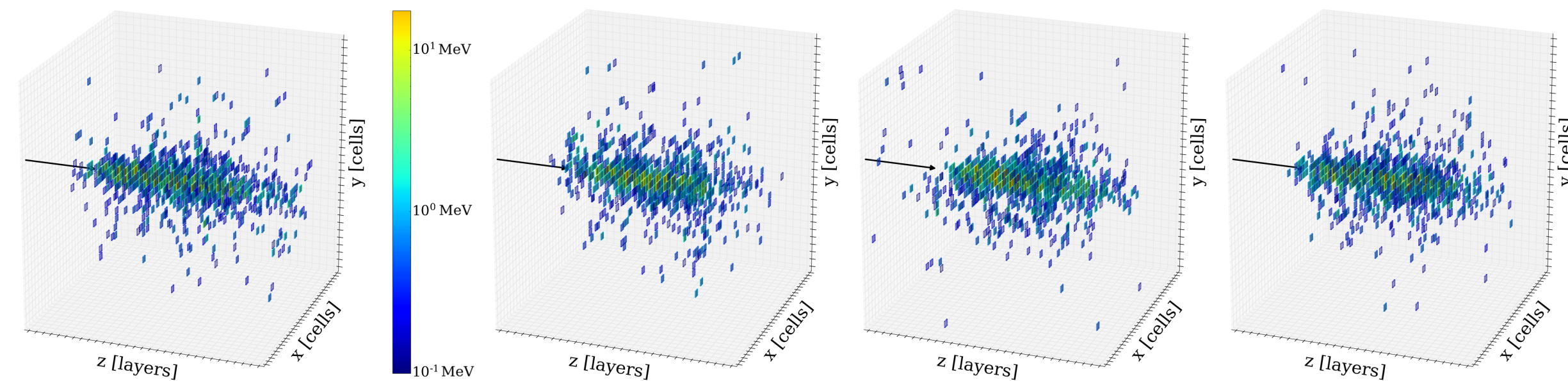
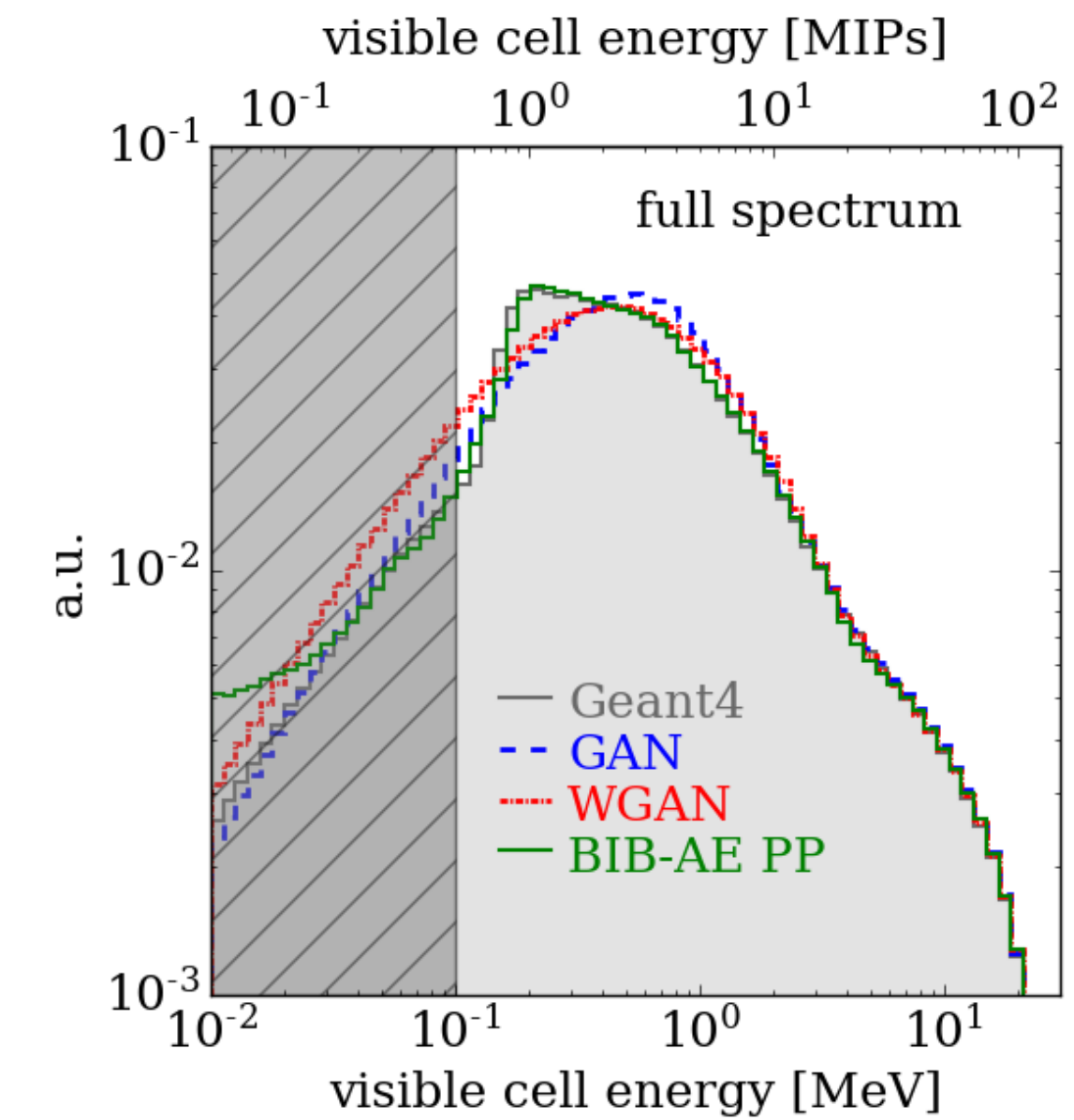
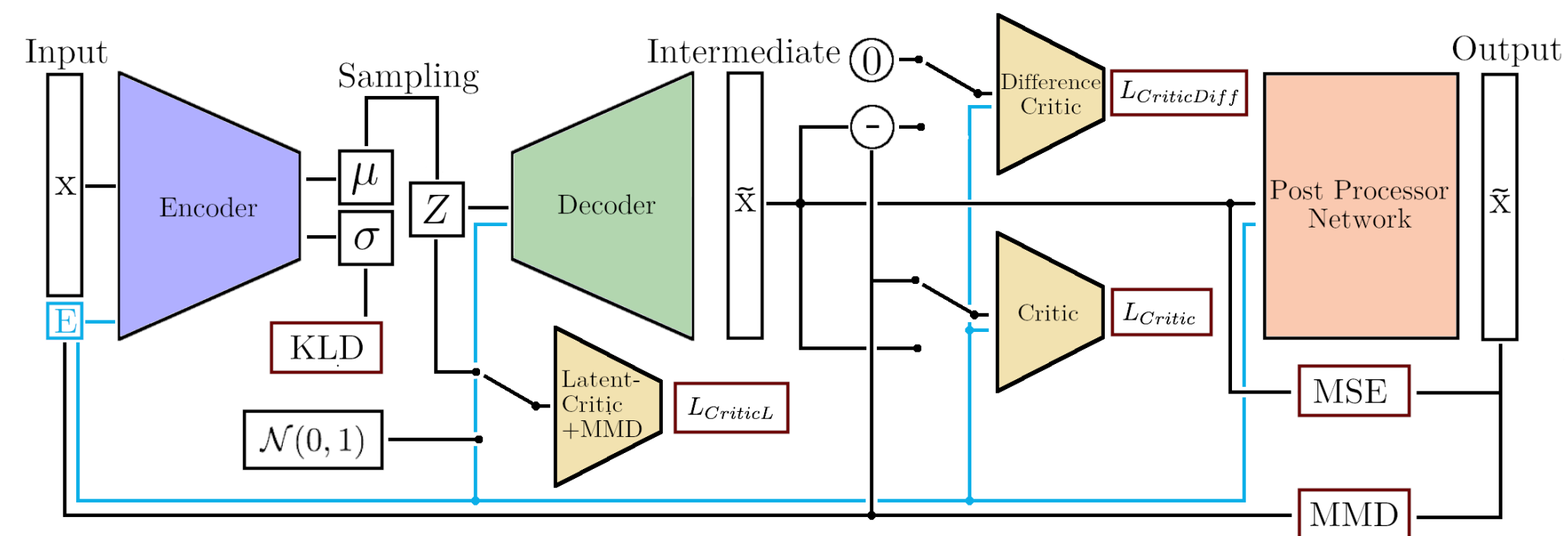
- Variational objective not optimal (only a lower bound)
- Not state-of-the-art on any real-world task

$$-\underbrace{\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})]}_{\text{reconstruction error}} + \underbrace{\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))}_{\text{regularization}}$$

BIB-AE

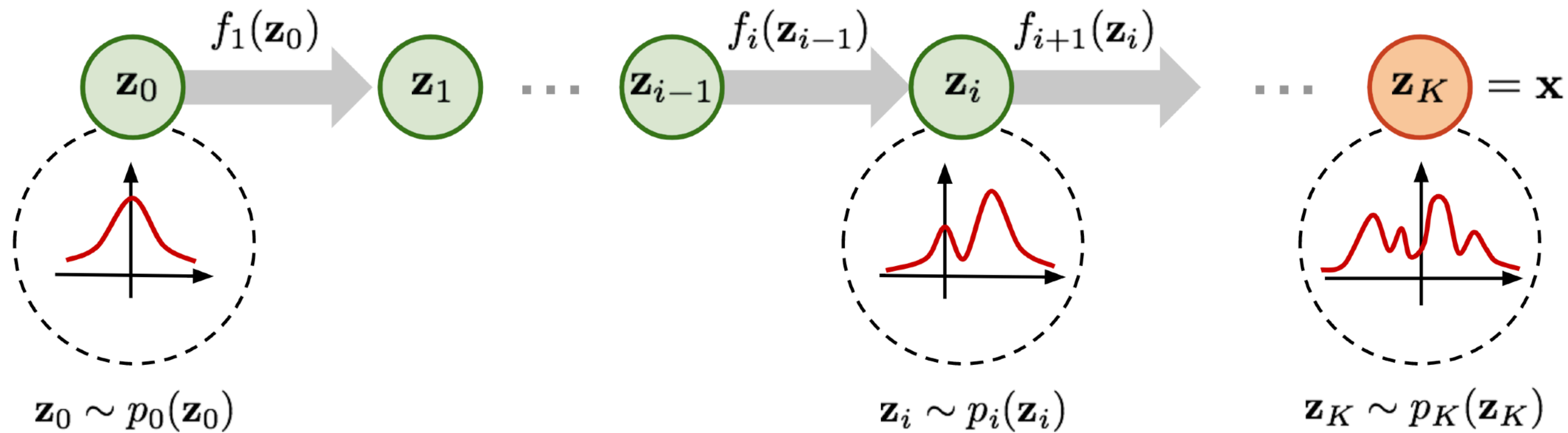
Buhmann et al [2005.05334, 2112.09709]

Combination of VAE and GAN



30x30x30 = 27,000 voxels ILD prototype — **current frontier in dimensionality**

Normalizing Flows



<https://lilianweng.github.io/posts/2018-10-13-flow-models/>

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

Idea: learn an invertible transformation between data space and latent space with simple base distribution (e.g. normal distribution)

Pros:

- Get density estimation and generative modeling
- Stable, optimal training (MLE)
- Principled model selection

Cons:

- Can be computationally expensive
- Fast vs slow direction

CaloFlow I

Krause & DS 2106.05285

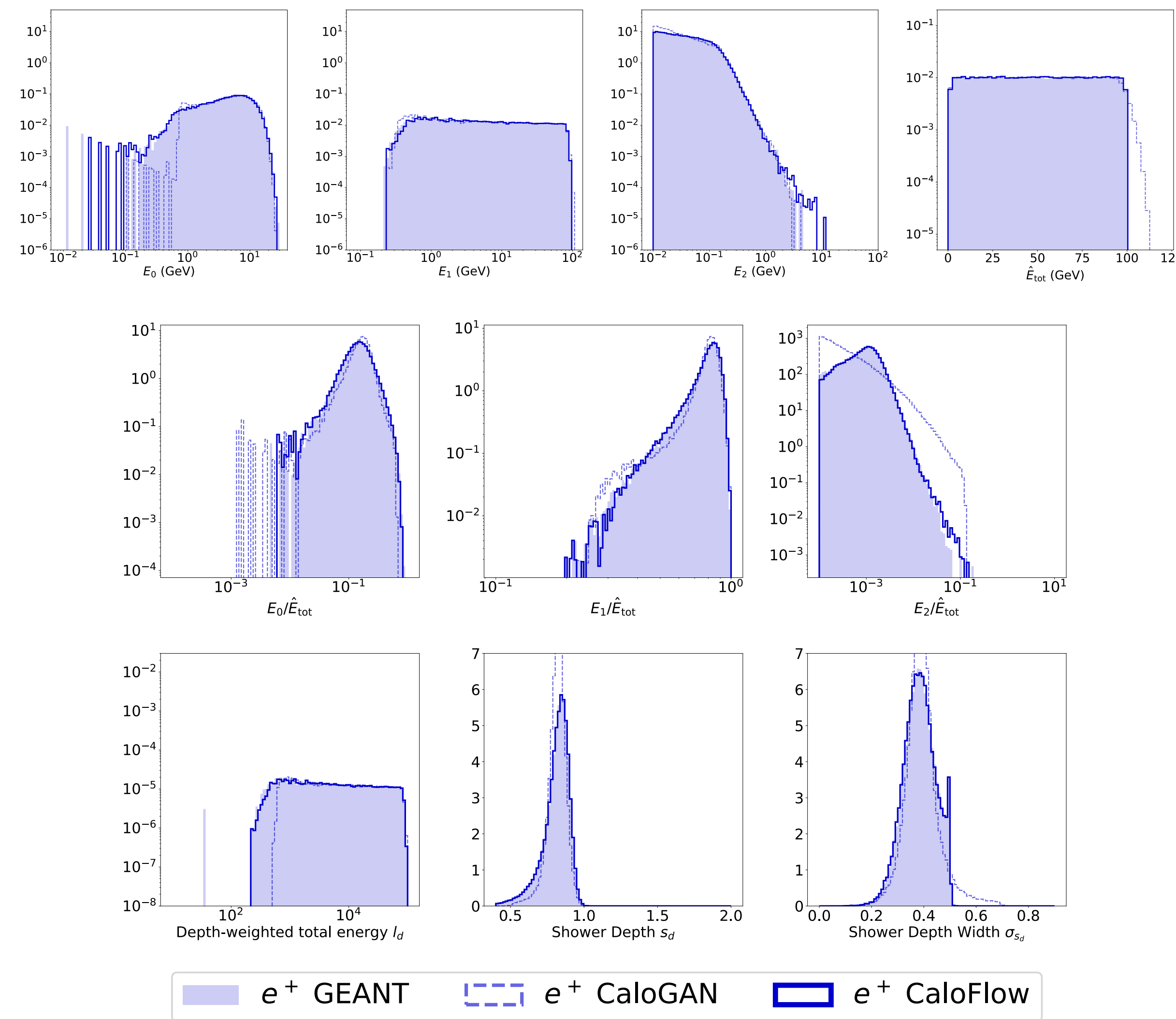
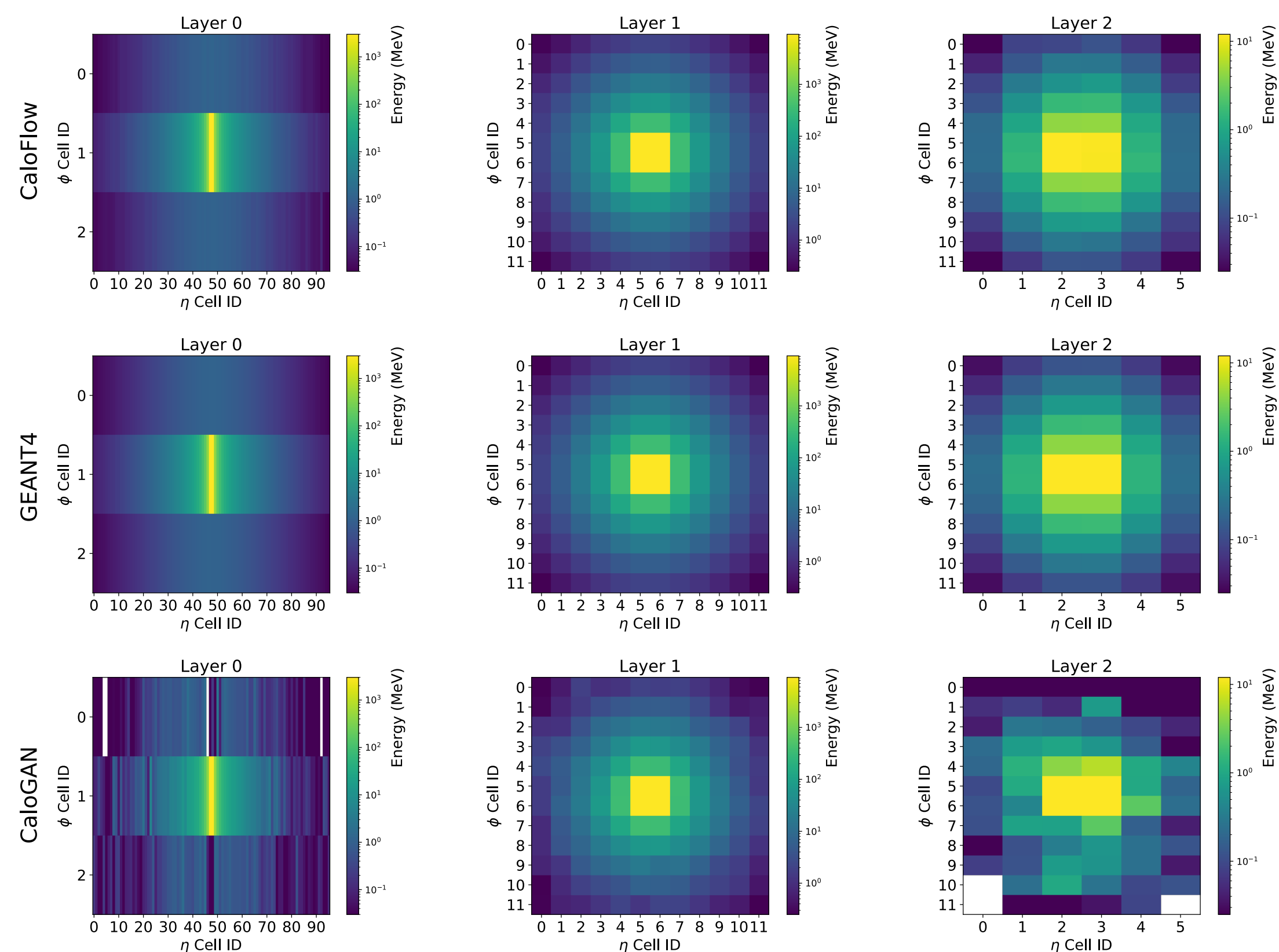


Figure 5. Average shower shapes for e^+ . Columns are calorimeter layers 0 to 2, top row shows CALOFlow, center row GEANT4, and bottom row CALOGAN

Performance vs GAN much improved!

CaloFlow I

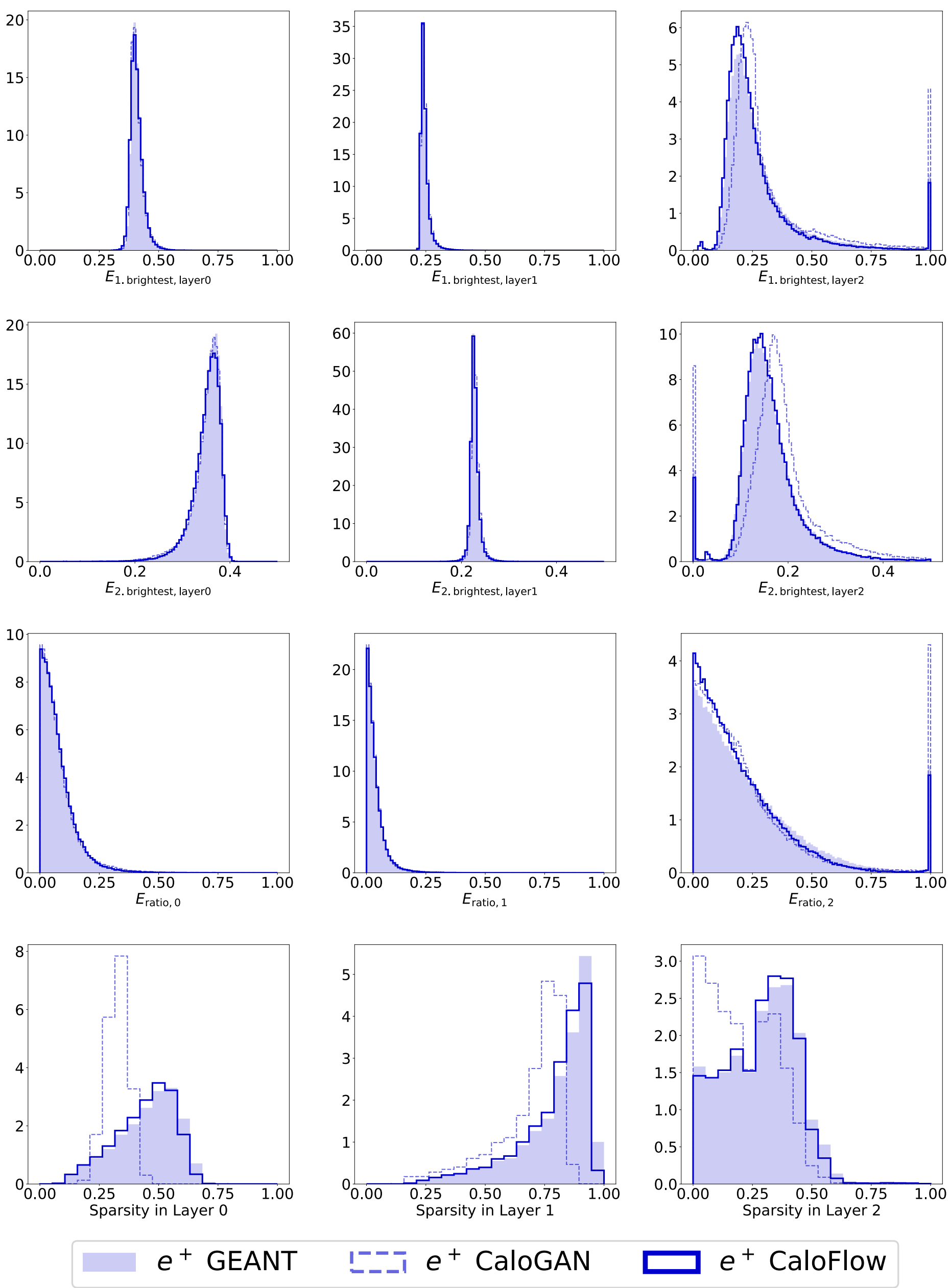
Krause & DS 2106.05285

AUC	GEANT4 vs. CaloGAN	GEANT4 vs. CaloFlow
e^+	1.000(0)	0.847(8)
γ	1.000(0)	0.660(6)
π^+	1.000(0)	0.632(2)

First to ever pass the “ultimate classifier metric” test:

DNN binary classifier, generated vs reference samples

Perfect generative model $p_{gen}(x) = p_{ref}(x) \Rightarrow$
classifier AUC=0.5



CaloFlow I

Krause & DS 2106.05285

Table 4. Generation time of a single calorimeter shower in ms. Times were obtained on a TITAN V GPU. GEANT4 needs 1772 ms per shower [8].

batch size	CALOGAN		CALOFlow
	batch size requested	100k requested	
10	455	2.2	835
100	45.5	0.3	96.1
1000	4.6	0.08	41.4
5000	1.0	0.07	36.2
10000	0.5	0.07	36.2

Main drawback of NF architecture we used (MAF): sampling is slow

Accurate, but not fast!

CaloFlow II

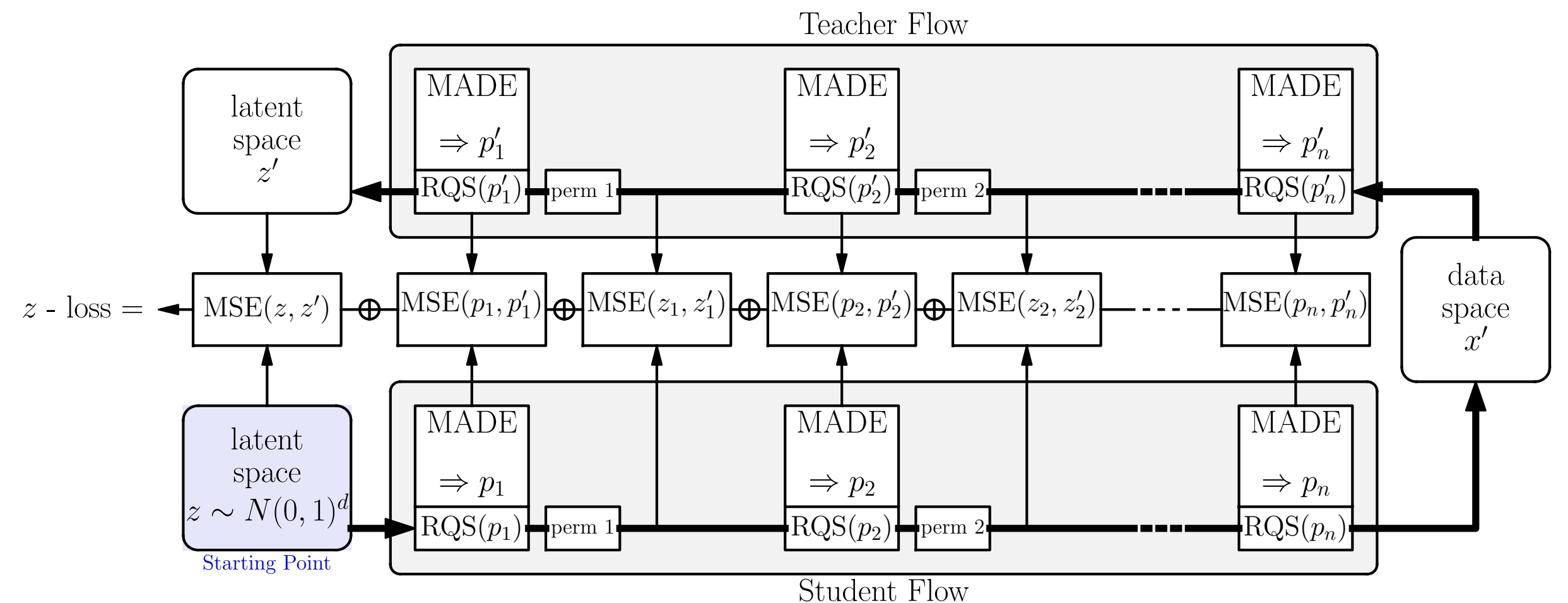
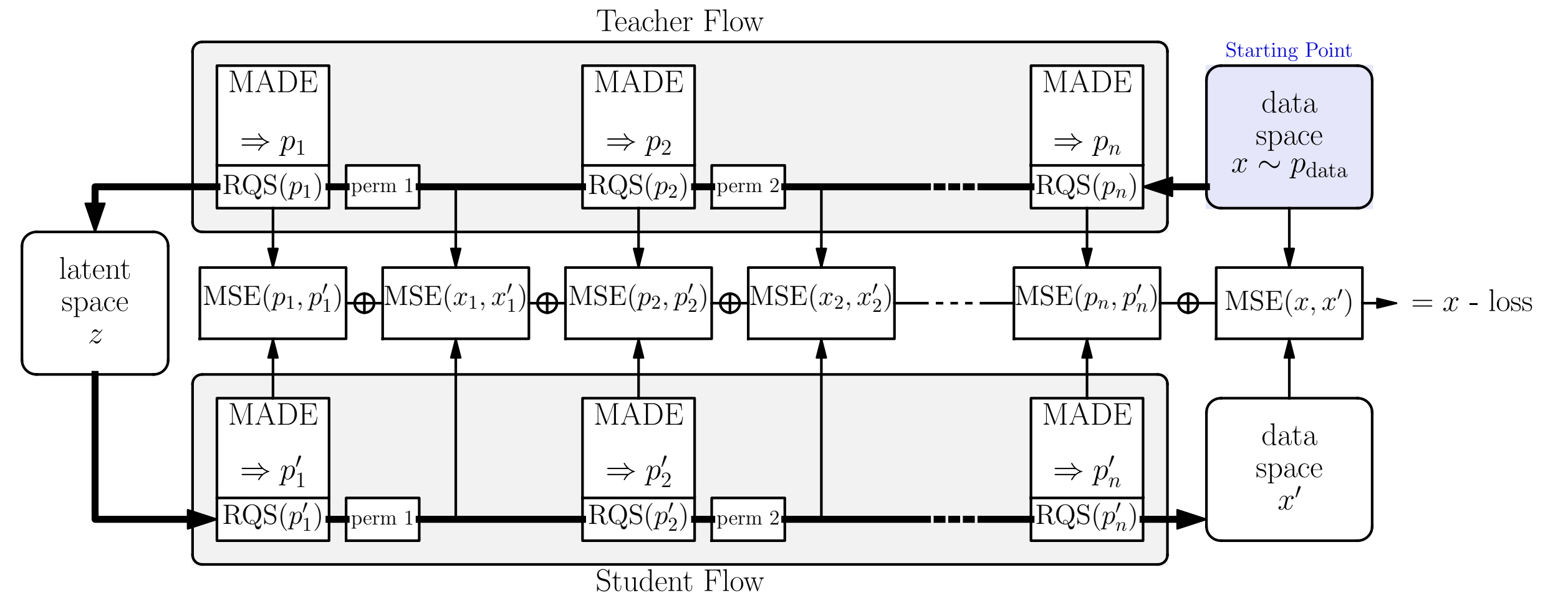
Krause & DS 2110.11377

IAF cannot be trained with MLE loss.

Instead we developed a new teacher/student method to train it. Based on “Probability Density Distillation”.
van den Oord et al [1711.10433](#)
Huang et al PMLR 2020

Idea: constrain IAF to be exact inverse of MAF, layer by layer and coupling by coupling

$$L = 0.5 \left(\underbrace{\text{MSE}(z, z') + \sum_i \text{MSE}(z^{(i)}, z'^{(i)}) + \sum_i \text{MSE}(p_z^{(i)}, p_z'^{(i)})}_{z\text{-loss}} \right) + 0.5 \left(\underbrace{\text{MSE}(x, x') + \sum_i \text{MSE}(x^{(i)}, x'^{(i)}) + \sum_i \text{MSE}(p_x^{(i)}, p_x'^{(i)})}_{x\text{-loss}} \right)$$



Training only involves fast directions of MAF and IAF ¹⁸

CaloFlow II

IAF with RQS transformations

Krause & DS 2110.11377

Similarly impressive accuracy

And a factor of 500 faster,
on par with GAN!

AUC / JSD		DNN	
		GEANT4 vs. CALOFlow v2 (student)	GEANT4 vs. CALOFlow v1 (teacher) [17]
e^+	unnormalized	0.785(7) / 0.200(10)	0.847(8) / 0.345(12)
	normalized	0.824(5) / 0.255(8)	0.869(2) / 0.376(4)
γ	unnormalized	0.761(14) / 0.167(18)	0.660(6) / 0.067(4)
	normalized	0.761(4) / 0.159(6)	0.794(4) / 0.213(7)
π^+	unnormalized	0.729(2) / 0.144(3)	0.632(2) / 0.048(1)
	normalized	0.807(2) / 0.231(4)	0.751(4) / 0.148(4)

Table 4. Training and evaluation times of CALOFlow and CALOGAN.

	CALOFlow		CALOGAN	GEANT4	
	v1 (teacher) [17]	v2 (student)			
training	22+82 min	+ 480 min	210 min		0 min
generation	time per shower				
batch size			batch size req.	100k req.	
10	835 ms	5.81 ms	455 ms	2.2 ms	1772 ms
100	96.1 ms	0.60 ms	45.5 ms	0.3 ms	1772 ms
1000	41.4 ms	0.12 ms	4.6 ms	0.08 ms	1772 ms
10000	36.2 ms	0.08 ms	0.5 ms	0.07 ms	1772 ms

CaloChallenge 2022

Fast Calorimeter Simulation Challenge 2022

[View on GitHub](#)

Welcome to the home of the first-ever Fast Calorimeter Simulation Challenge!

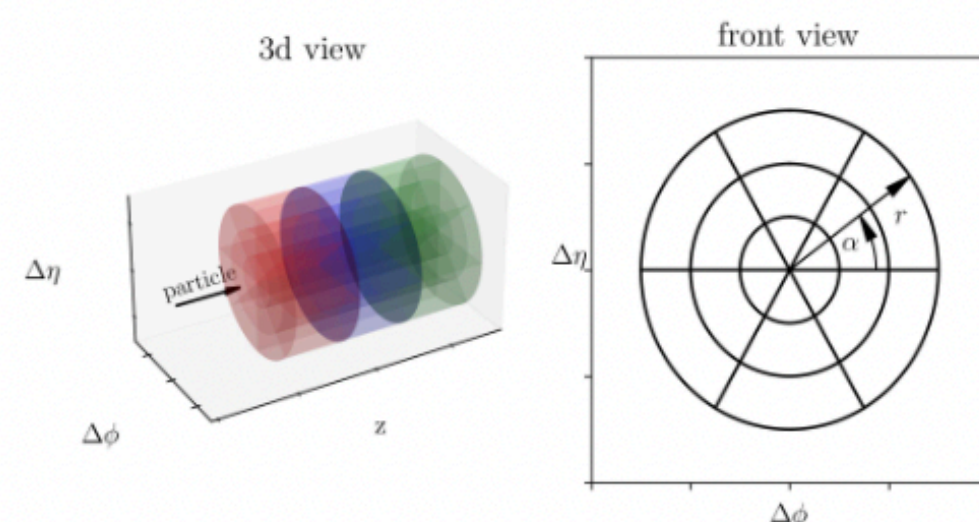
The purpose of this challenge is to spur the development and benchmarking of fast and high-fidelity calorimeter shower generation using deep learning methods. Currently, generating calorimeter showers of interacting particles (electrons, photons, pions, ...) using GEANT4 is a major computational bottleneck at the LHC, and it is forecast to overwhelm the computing budget of the LHC experiments in the near future. Therefore there is an urgent need to develop GEANT4 emulators that are both fast (computationally lightweight) and accurate. The LHC collaborations have been developing fast simulation methods for some time, and the hope of this challenge is to directly compare new deep learning approaches on common benchmarks. It is expected that participants will make use of cutting-edge techniques in generative modeling with deep learning, e.g. GANs, VAEs and normalizing flows.

This challenge is modeled after two previous, highly successful data challenges in HEP – the [top tagging community challenge](#) and the [LHC Olympics 2020 anomaly detection challenge](#).

Datasets

The challenge offers three datasets, ranging in difficulty from “easy” to “medium” to “hard”. The difficulty is set by the dimensionality of the calorimeter showers (the number layers and the number of voxels in each layer).

Each dataset has the same general format. The detector geometry consists of concentric cylinders with particles propagating along the z-axis. The detector is segmented along the z-axis into discrete layers. Each layer has bins along the radial direction and some of them have bins in the angle α . The number of layers and the number of bins in r and α is stored in the binning .xml files and will be read out by the HighLevelFeatures class of helper functions. The coordinates $\Delta\varphi$ and $\Delta\eta$ correspond to the x- and y axis of the cylindrical coordinates. The image below shows a 3d view of a geometry with 3 layers, with each layer having 3 bins in radial and 6 bins in angular direction. The right image shows the front view of the geometry, as seen along the z axis.



Ongoing data challenge for fast calorimeter simulation

Organizers: Giannelli, Kasieczka, Krause, Nachman, Salamani, **DS**, Zaborowska

3 datasets:

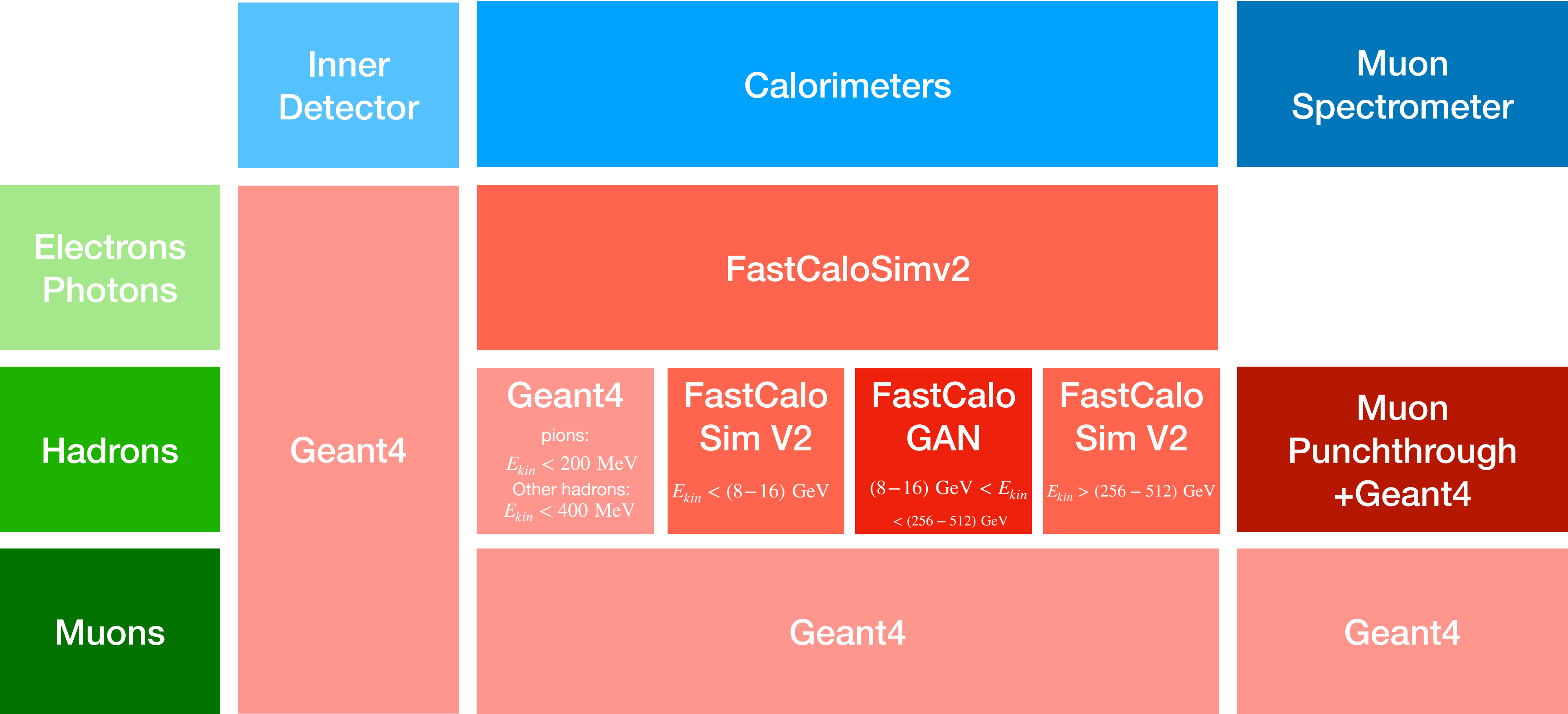
- “easy” — official ATLAS CaloSim ($\sim 10^2$ voxels)
- “medium” — GEANT4 example detector ($\sim 10^3$ voxels)
- “hard” — GEANT4 example detector ($\sim 10^4$ voxels)

First results to be presented at ML4Jets2022@Rutgers in November

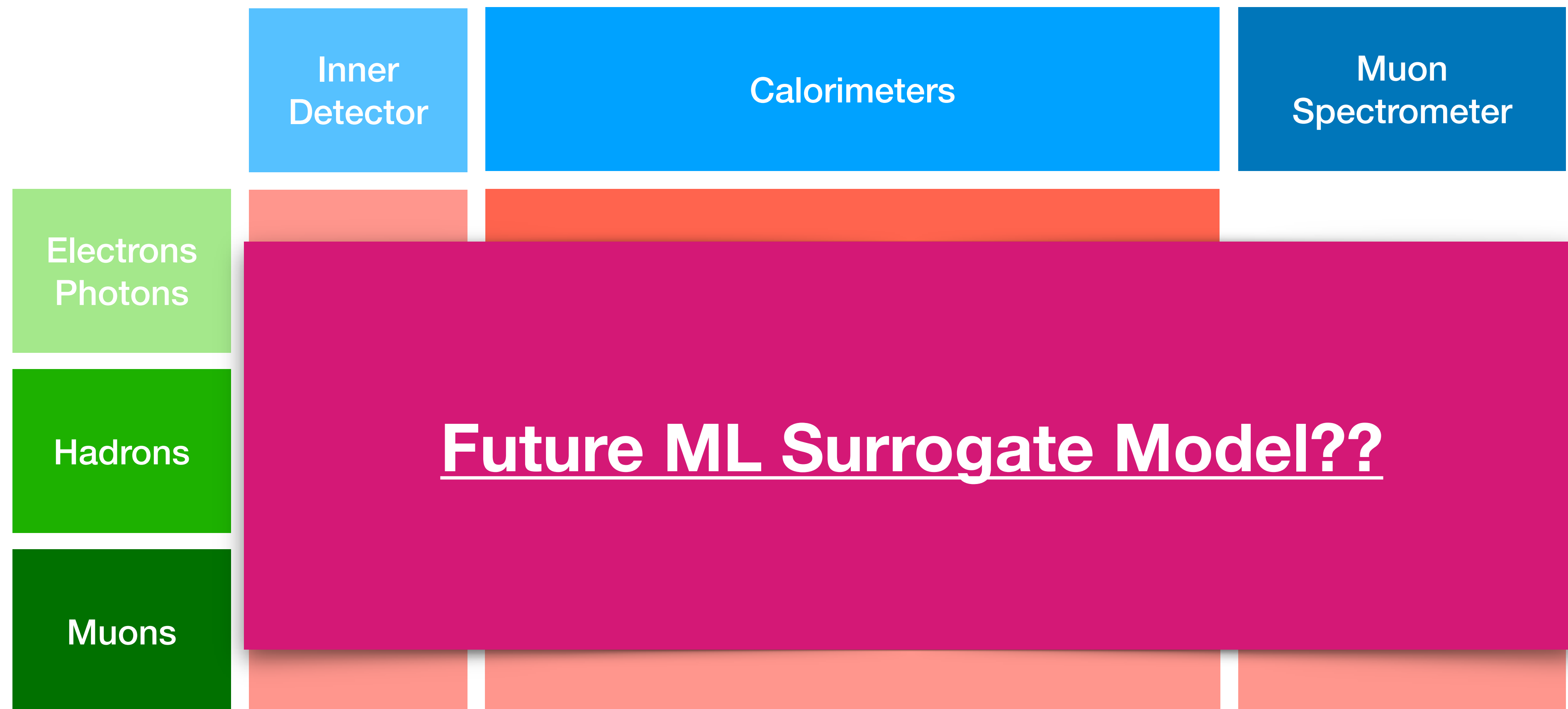
Conclusions

- These are exciting times for FastSim with deep learning!
- GANs, VAEs and Normalizing Flows have shown much initial promise for fast and accurate detector simulation
 - Normalizing Flow is currently state-of-the-art on “small” dimensionalities (500 voxels), first ever to pass a classifier test
 - GAN+VAE is currently state-of-the-art on “large” dimensionalities (10^4 voxels)
- Expect more exciting developments with the outcome of the CaloChallenge

Outlook



Outlook



Can we dream of “replacing” GEANT4 with a future fast and accurate ML surrogate model?