New developments in fast simulation with machine learning

— LHCP 2022, virtually in Taipeh —

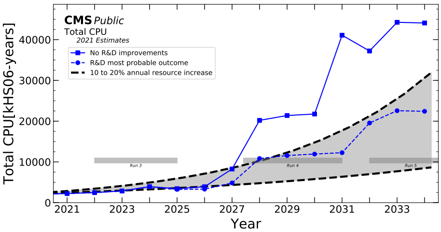
Claudius Krause

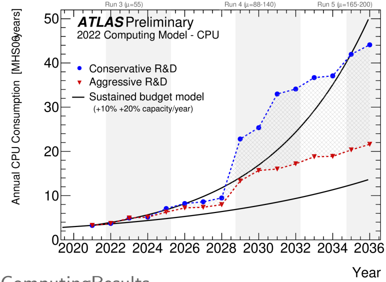
Rutgers, The State University of New Jersey

May 18, 2022

In collaboration with David Shih
arXiv: 2106.05285 and 2110.11377

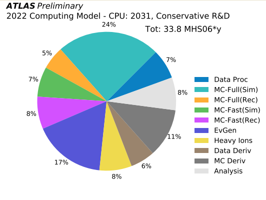
Deep Generative Models will be crucial for the LHC.





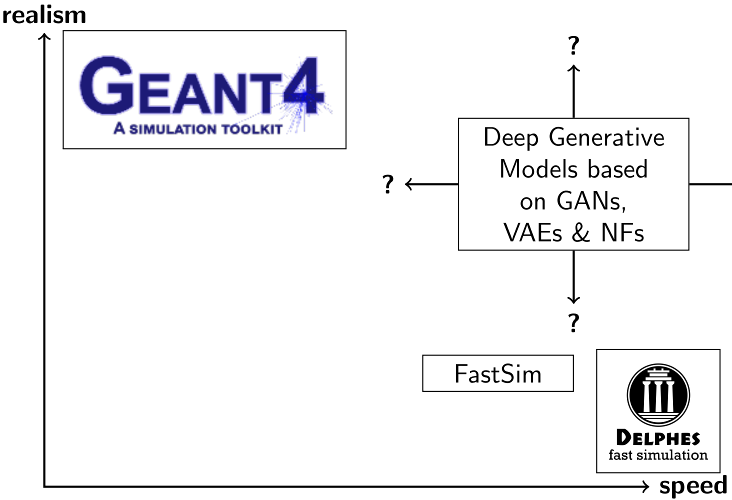
https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults
https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/UPGRADE/CERN-LHCC-2022-005/

- At the start of LHC Run 4, the computational needs will likely exceed the available budget.
- A large fraction goes into simulation.

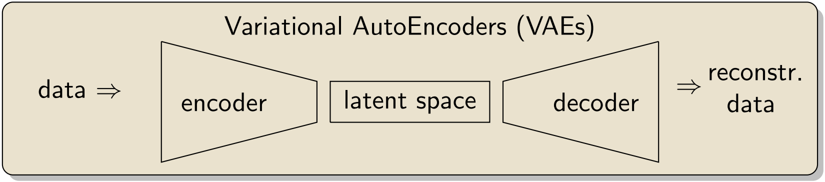


CERN-LHCC-2022-005

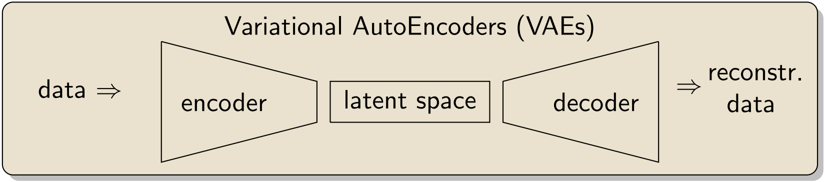
Detector Simulation needs to be fast and faithful

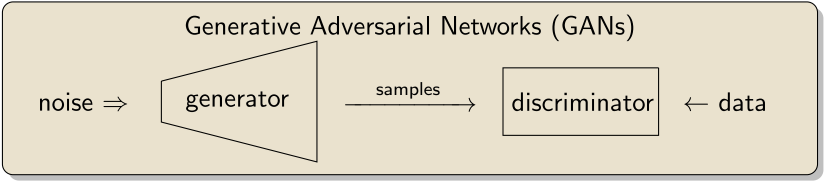


There are 3 main deep generative models.



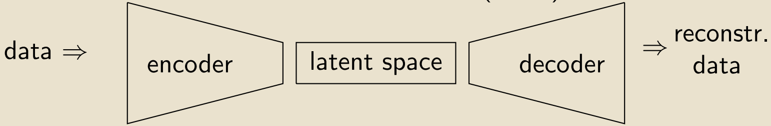
There are 3 main deep generative models.





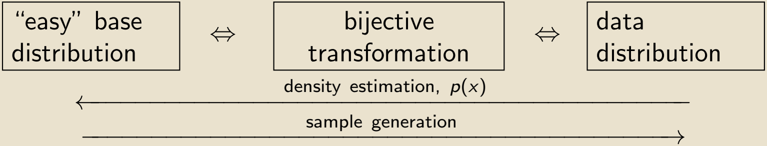
There are 3 main deep generative models.

Variational AutoEncoders (VAEs)



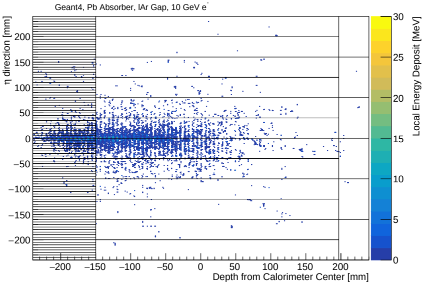
Generative Adversarial Networks (GANs)

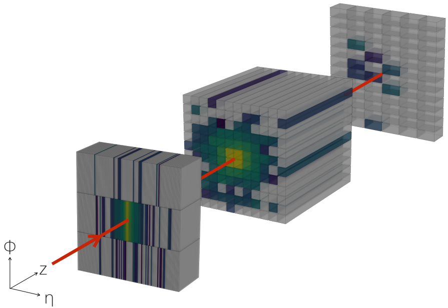
Normalizing Flows (NFs)



We use the same calorimeter geometry as CaloGAN

- We consider a simplified version of the ATLAS ECal: flat alternating layers of lead and LAr
- They form three instrumented layers of dimension 3×96, 12×12, and 12×6

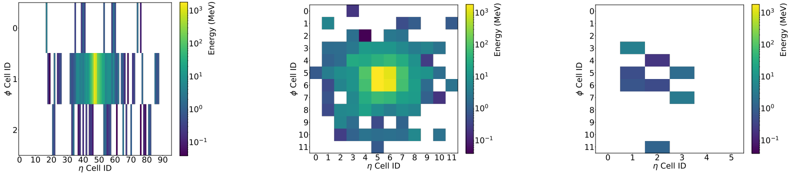




CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

We use the same calorimeter geometry as CaloGAN

- The Geant4 configuration of CaloGAN is available at https://github.com/hep-lbdl/CaloGAN
- We produce our own dataset: available at [DOI: 10.5281/zenodo.5904188]
- Showers of e^+, γ, and π^+ (100k each)
- All are centered and perpendicular
- E_{tot} is uniform in $[1, 100]$ GeV and given in addition to the energy deposits per voxel:



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

Normalizing Flows learn a change-of-coordinates efficiently.

"easy" base distribution ⇔ bijective transformation ⇔ "target" distribution, data

density estimation, $p(x)$ ←

sample generation →

- Our architecture is autoregressive:
  - Masked Autoregressive Flow (MAF), introduced in Papamakarios et al. [arXiv:1705.07057], are slow in sampling and fast in inference.
  - Inverse Autoregressive Flow (IAF), introduced in Kingma et al. [arXiv:1606.04934], are fast in sampling and slow in inference.
- Our transformation is given by Rational Quadratic Splines.
  - A NN predicts the bin widths, heights, and derivatives at each knot.
  - Based on Durkan et al. [arXiv:1906.04032]
    Gregory/Delbourgo [IMA Journal of Numerical Analysis, '82]

CaloFlow uses a 2-step approach.

Flow I

- learns $p_1(E_0, E_1, E_2 | E_{\text{tot}})$
- is a MAF that is optimized using the LL.

Flow II

- learns $p_2(\vec{\mathcal{I}} | E_0, E_1, E_2, E_{\text{tot}})$ of normalized showers
- in CaloFlow v1 (2106.05285 — called "teacher"):
  - MAF trained with LL
  - Slow in sampling ($\approx 500\times$ slower than CaloGAN)
- in CaloFlow v2 (2110.11377 — called "student"):
  - IAF trained with Probability Density Distillation from teacher (LL prohibitive)
    van den Oord et al. [1711.10433]
    i.e. matching IAF parameters to frozen MAF
  - Fast in sampling ($\approx 500\times$ faster than CaloFlow v1)

A Classifier provides the “ultimate metric”.

According to the Neyman-Pearson Lemma we have:

- The likelihood ratio is the most powerful test statistic to distinguish the two samples.
- A powerful classifier trained to distinguish the samples should therefore learn (something monotonically related to) this.
- If this classifier is confused, we conclude $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$

$\Rightarrow$ This captures the full 504-dim. space.

? But why wasn't this used before?

$\Rightarrow$ Previous deep generative models were separable to almost 100%!

DCTRGAN: Diefenbacher et al. [2009.03796, JINST]

CaloFlow passes the "ultimate metric" test.

According to the Neyman-Pearson Lemma we have:
$p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$ if a classifier cannot distinguish data from generated samples.

AUC		DNN based classifier		
		GEANT4 vs. CaloGAN	GEANT4 vs. (teacher) CaloFlow v1	GEANT4 vs. (student) CaloFlow v2
e^+	unnorm.	1.000(0)	0.859(10)	0.786(7)
	norm.	1.000(0)	0.870(2)	0.824(4)
γ	unnorm.	1.000(0)	0.756(48)	0.758(14)
	norm.	1.000(0)	0.796(2)	0.760(3)
π^+	unnorm.	1.000(0)	0.649(3)	0.729(2)
	norm.	1.000(0)	0.755(3)	0.807(1)

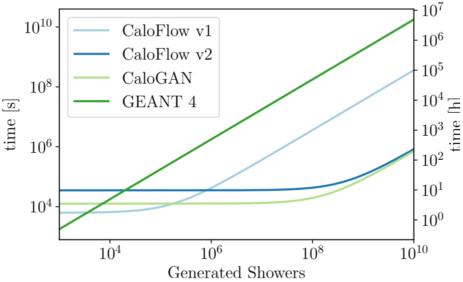
AUC ($\in [0.5, 1]$): Area Under the ROC Curve

Sampling Speed: The Student beats the Teacher!

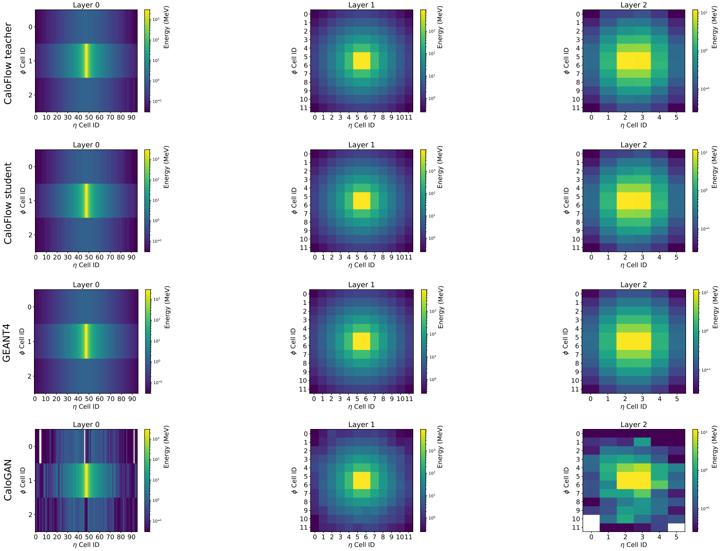
	CaloFlow* teacher	CaloFlow* student	CaloGAN* batch size req.	CaloGAN* 100k req.	Geant4†
training	22+82 min	+ 480 min	210 min		0 min
generation batch size	time per shower				
10	835 ms	5.81 ms	455 ms	2.2 ms	1772 ms
100	96.1 ms	0.60 ms	45.5 ms	0.3 ms	1772 ms
1000	41.4 ms	0.12 ms	4.6 ms	0.08 ms	1772 ms
10000	36.2 ms	**0.08 ms**	0.5 ms	**0.07 ms**	1772 ms

*: on our Titan V GPU

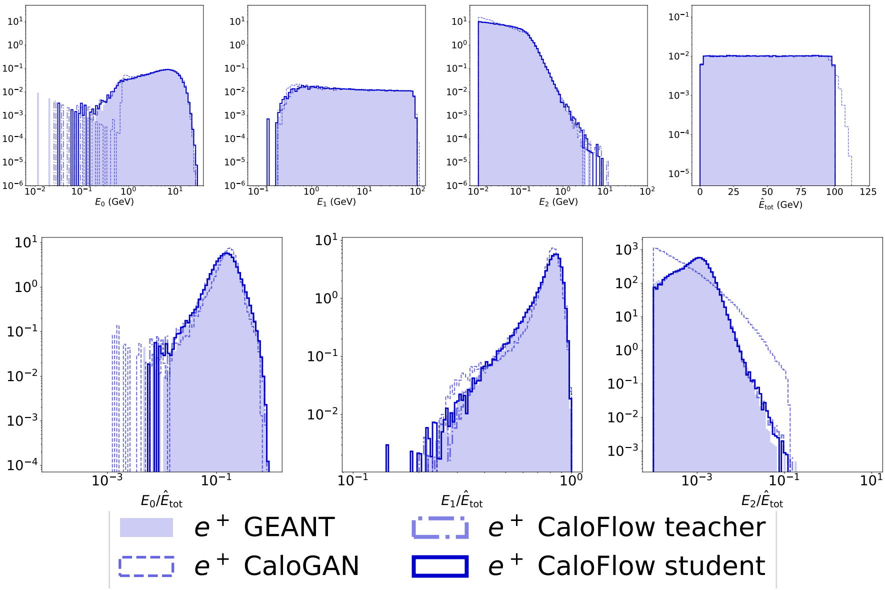
†: on the CPU of CaloGAN: Paganini, de Oliveira, Nachman [1712.10321, PRD]



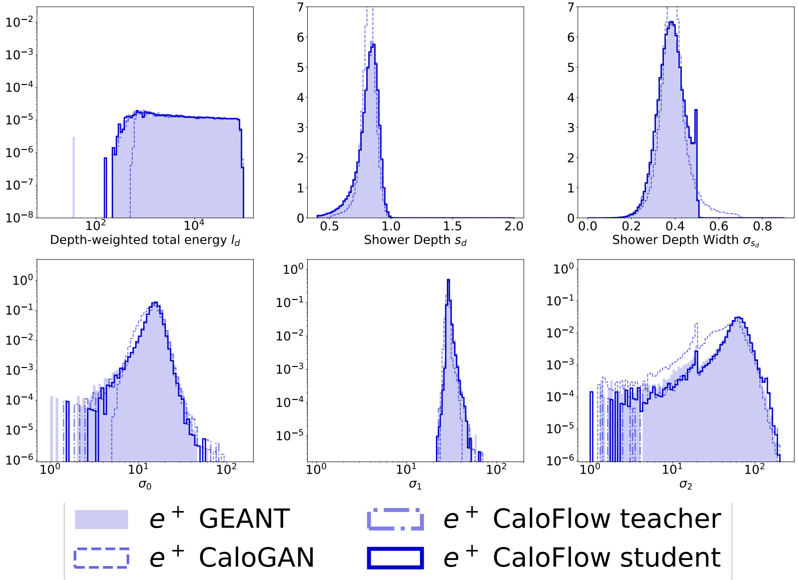
CaloFlow: Comparing Shower Averages: e^+



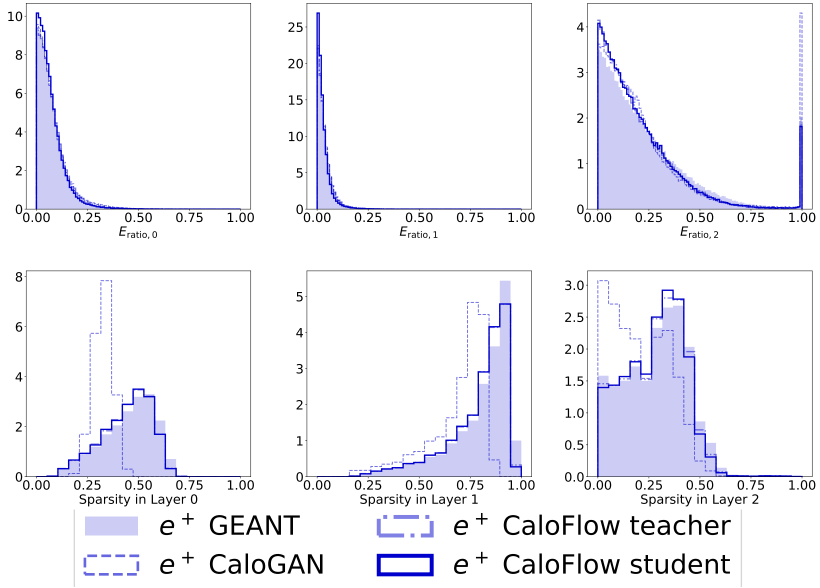
CaloFlow: Flow I histograms: e^+



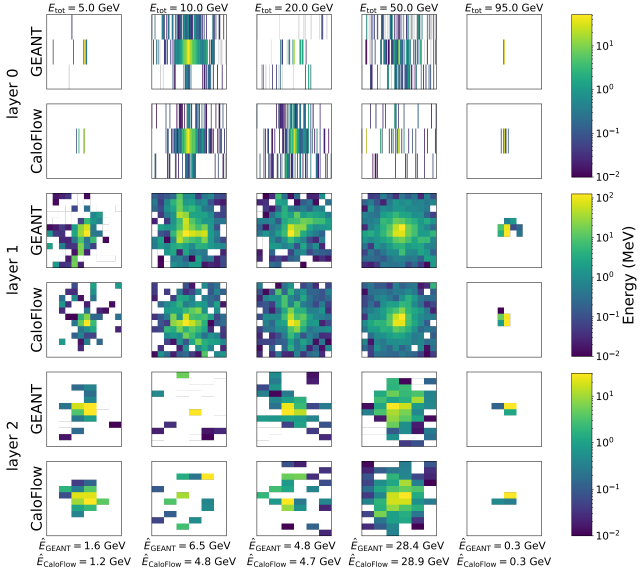
CaloFlow: Flow I+II histograms: e^+



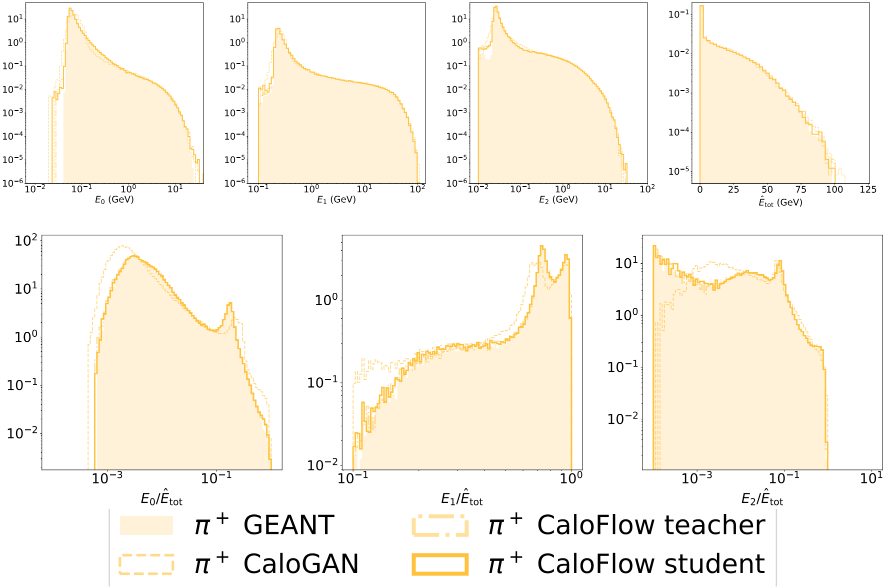
CaloFlow: Flow II histograms: e^+



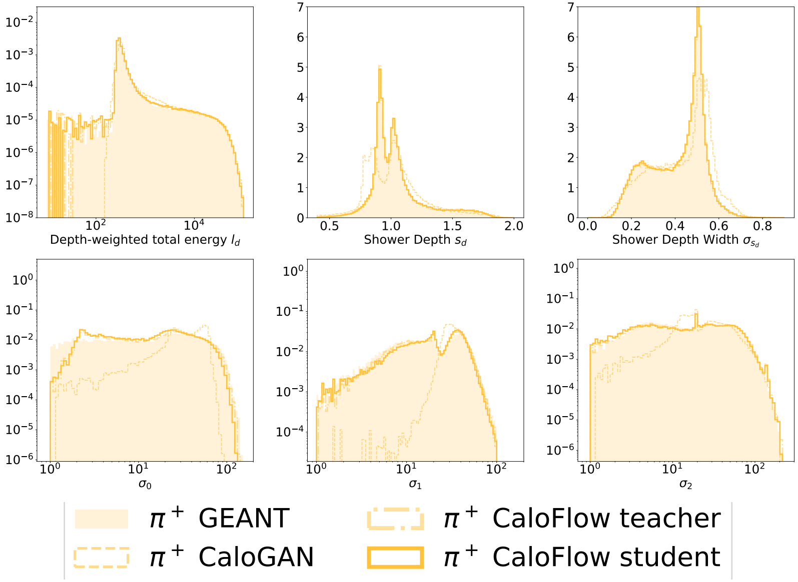
CaloFlow: Nearest Neighbors: π^+ (student)



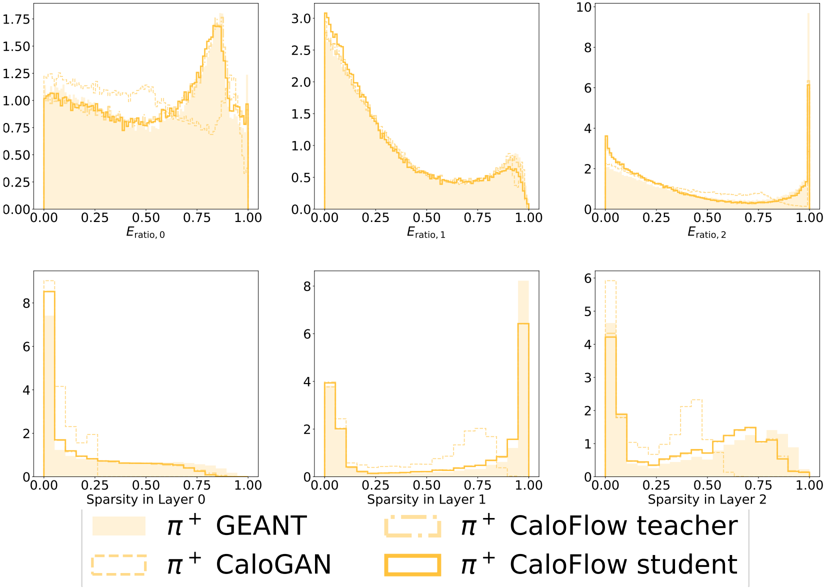
CaloFlow: Flow I histograms: π^+



CaloFlow: Flow I+II histograms: π^+



CaloFlow: Flow II histograms: π^+

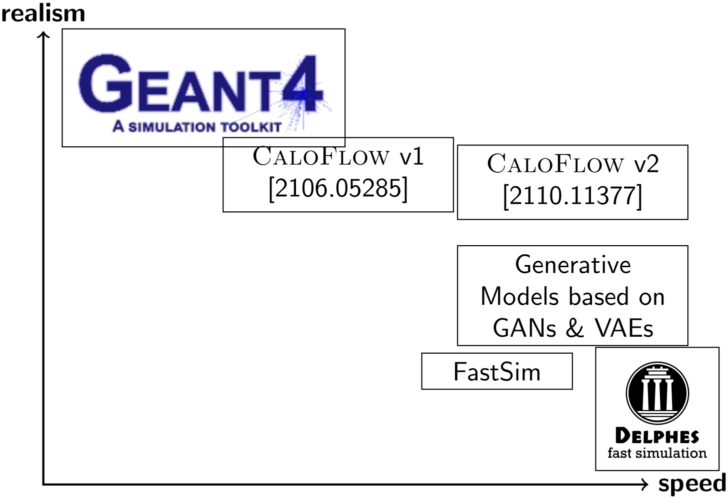


A little Advertisement — CaloChallenge 2022

Michele Faucci Giannelli, Gregor Kasieczka, Claudius Krause, Ben Nachman, Dalila Salamani, David Shih, and Anna Zaborowska

$\Rightarrow$ `https://calochallenge.github.io/homepage/`

New developments in fast simulation with machine learning: CaloFlow — Summary —



Backup

The Bijector is a chain of "easy" transformations.

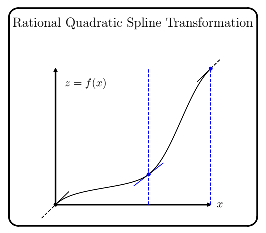
Each transformation

- must be invertible and have analytical Jacobian
- is chosen to factorize:
  $\vec{C}(\vec{x};\vec{p}) = (C_1(x_1;p_1), C_2(x_2;p_2), \dots, C_n(x_n;p_n))^T$,
  where $\vec{x}$ are the coordinates to be transformed and $\vec{p}$ the parameters of the transformation.

Rational Quadratic Splines:

Durkan et al. [arXiv:1906.04032]

Gregory/Delbourgo [IMA Journal of Numerical Analysis, '82]

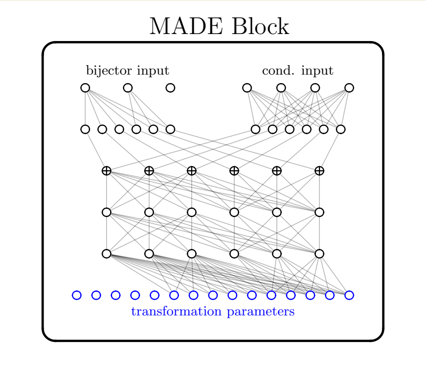


$$C = \frac{a_2\alpha^2 + a_1\alpha + a_0}{b_2\alpha^2 + b_1\alpha + b_0}$$

- numerically easy
- expressive

The NN predicts the bin widths, heights, and derivatives that go in a_i&b_i.

Masking Ensures the Autoregressive Property.



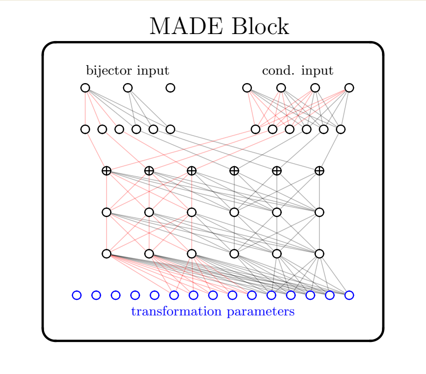
Implementation via masking:

- a single "forward" pass gives the full output of all $p(x_i|x_{i-1}\dots x_1)$.
  $\Rightarrow$ very fast
- the "inverse" needs to loop through all dimensions and gets a single $p(x_i|x_{i-1}\dots x_1)$ each time.
  $\Rightarrow$ very slow

Germain/Gregor/Murray/Larochelle [arXiv:1502.03509]

- Masked Autoregressive Flow (MAF), introduced in Papamakarios et al. [arXiv:1705.07057], are slow in sampling and fast in inference.
- Inverse Autoregressive Flow (IAF), introduced in Kingma et al. [arXiv:1606.04934], are fast in sampling and slow in inference.

Masking Ensures the Autoregressive Property.



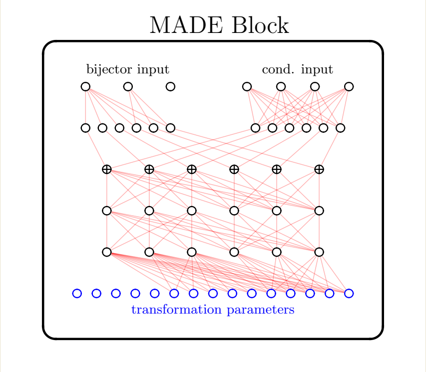
Implementation via masking:

- a single "forward" pass gives the full output of all $p(x_i|x_{i-1} \dots x_1)$.
  $\Rightarrow$ very fast
- the "inverse" needs to loop through all dimensions and gets a single $p(x_i|x_{i-1} \dots x_1)$ each time.
  $\Rightarrow$ very slow

Germain/Gregor/Murray/Larochelle [arXiv:1502.03509]

- Masked Autoregressive Flow (MAF), introduced in Papamakarios et al. [arXiv:1705.07057], are slow in sampling and fast in inference.
- Inverse Autoregressive Flow (IAF), introduced in Kingma et al. [arXiv:1606.04934], are fast in sampling and slow in inference.

Masking Ensures the Autoregressive Property.



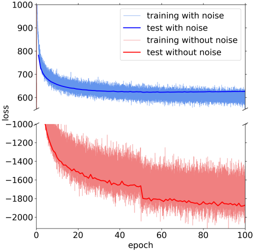
Implementation via masking:

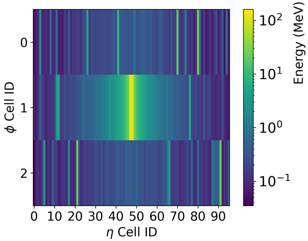
- a single "forward" pass gives the full output of all $p(x_i|x_{i-1}\dots x_1)$.
  $\Rightarrow$ very fast
- the "inverse" needs to loop through all dimensions and gets a single $p(x_i|x_{i-1}\dots x_1)$ each time.
  $\Rightarrow$ very slow

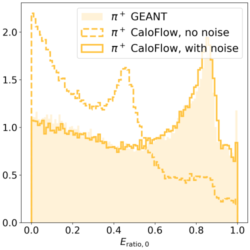
Germain/Gregor/Murray/Larochelle [arXiv:1502.03509]

- Masked Autoregressive Flow (MAF), introduced in Papamakarios et al. [arXiv:1705.07057], are slow in sampling and fast in inference.
- Inverse Autoregressive Flow (IAF), introduced in Kingma et al. [arXiv:1606.04934], are fast in sampling and slow in inference.

Adding Noise is important for the sampling quality.

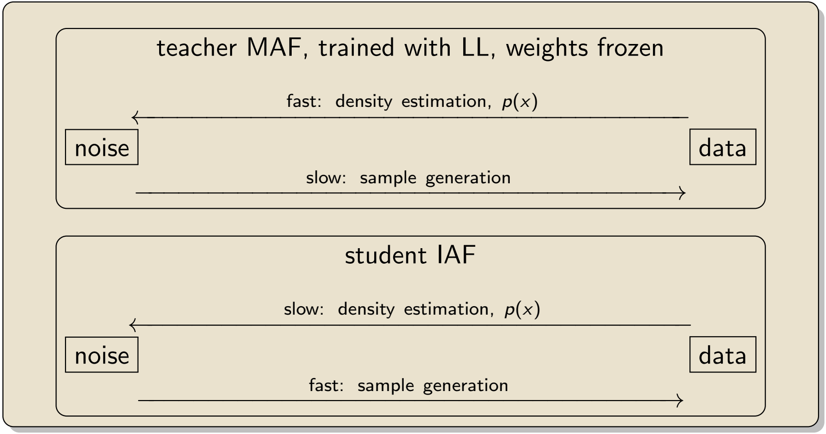






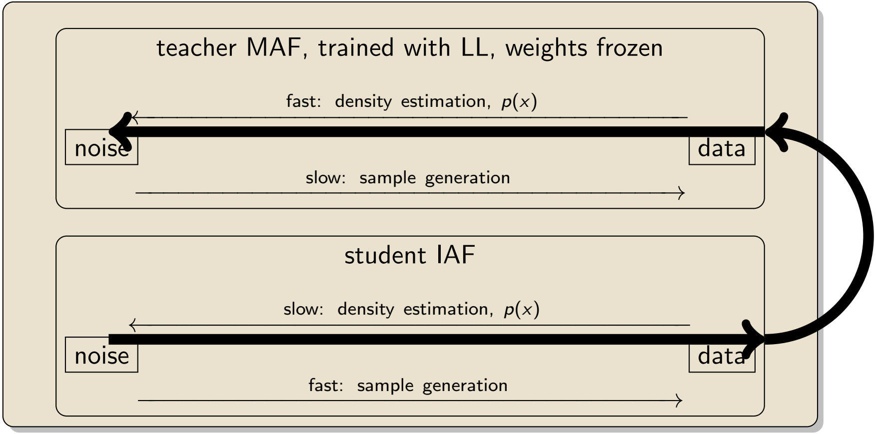
- The log-likelihood is less noisy, but smaller. Yet, the quality of the samples is much better!
- This is due to a "wider" mapping of space and less overfitting.

Probability Density Distillation passes the information from the teacher to the student



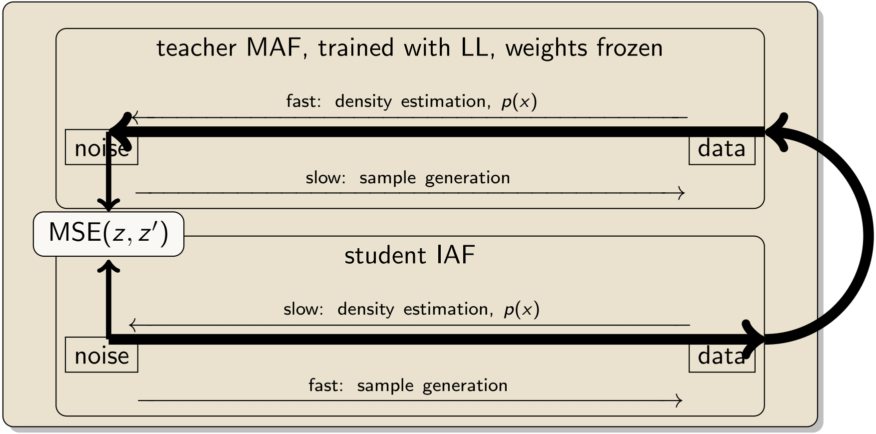
$$\text{Loss} = \text{MSE}(z, z') + \text{MSE}(x, x') + \text{MSE}(z_i, z_i') + \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')$$

Probability Density Distillation passes the information from the teacher to the student



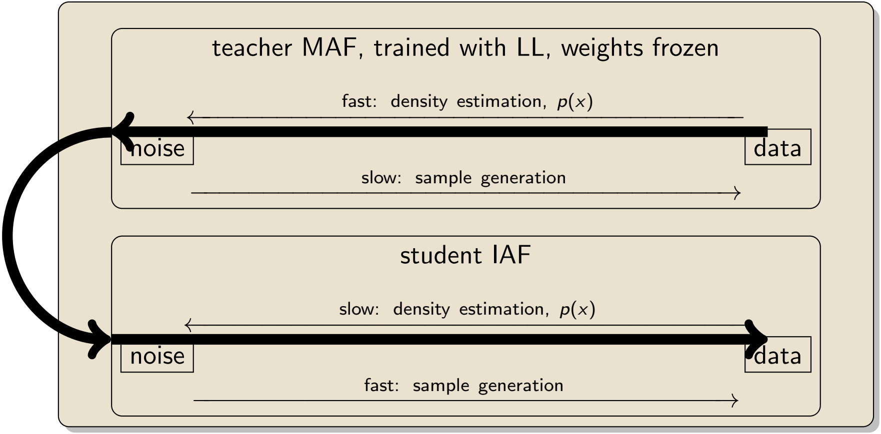
$$\text{Loss} = \text{MSE}(z, z') + \text{MSE}(x, x') + \text{MSE}(z_i, z_i') + \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')$$

Probability Density Distillation passes the information from the teacher to the student



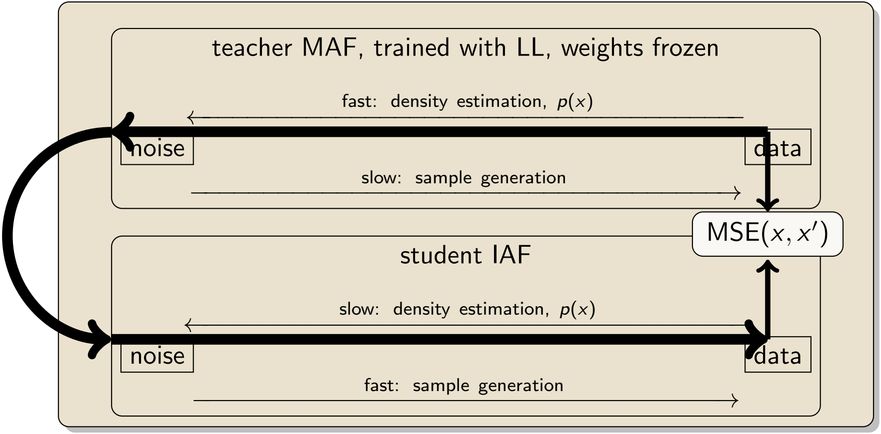
$$\text{Loss} = \text{MSE}(z, z') + \text{MSE}(x, x') + \text{MSE}(z_i, z_i') + \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')$$

Probability Density Distillation passes the information from the teacher to the student



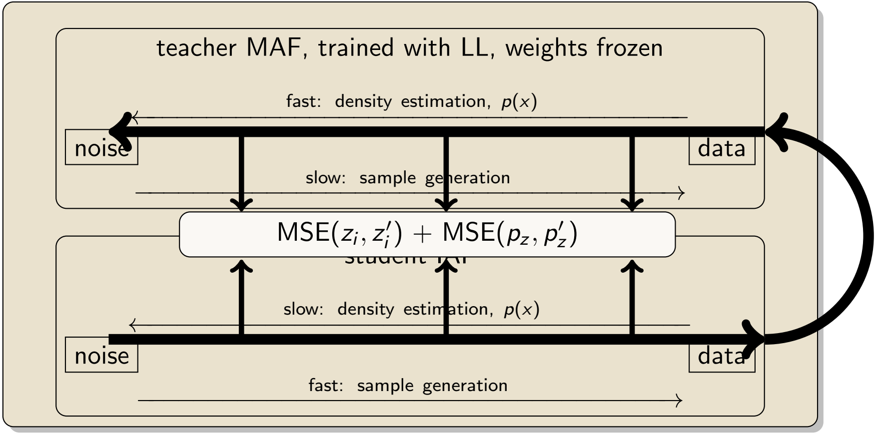
$$\text{Loss} = \text{MSE}(z, z') + \text{MSE}(x, x') + \text{MSE}(z_i, z_i') + \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')$$

Probability Density Distillation passes the information from the teacher to the student



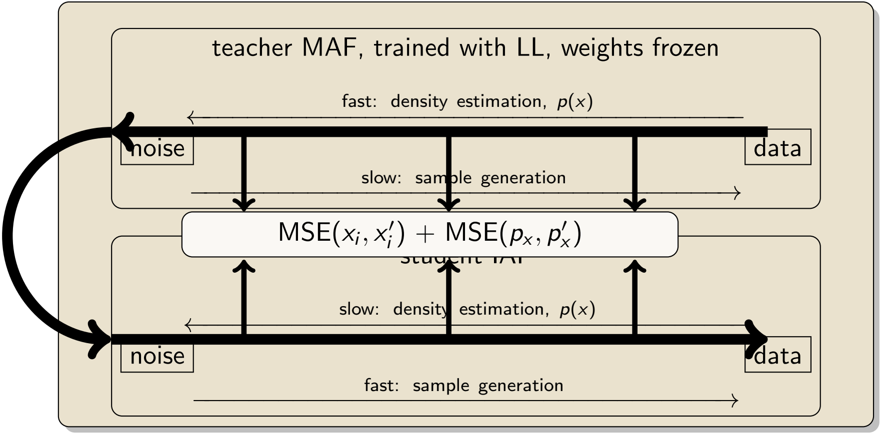
$$\text{Loss} = \text{MSE}(z, z') + \text{MSE}(x, x') + \text{MSE}(z_i, z_i') + \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')$$

Probability Density Distillation passes the information from the teacher to the student



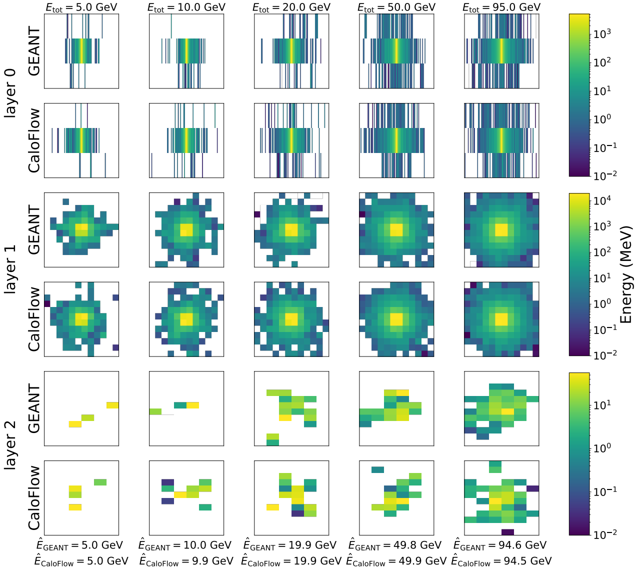
$$\begin{aligned}\text{Loss} = \text{MSE}(z, z') &+ \text{MSE}(x, x') + \text{MSE}(z_i, z_i') \\ &+ \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')\end{aligned}$$

Probability Density Distillation passes the information from the teacher to the student

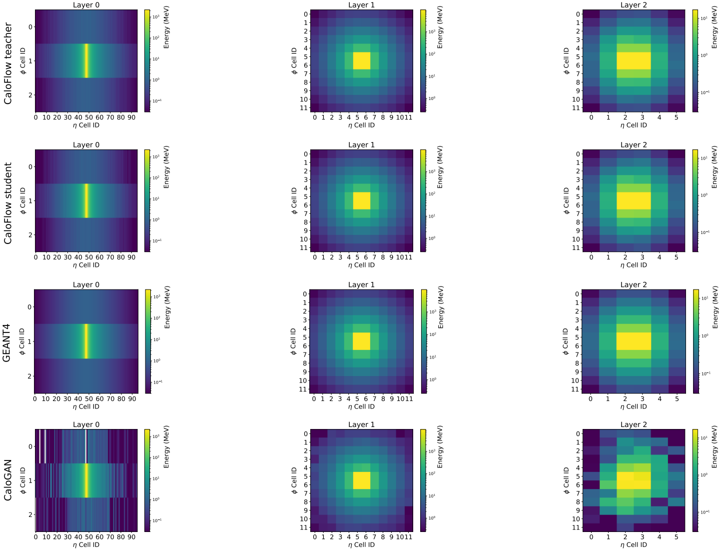


$$\text{Loss} = \text{MSE}(z, z') + \text{MSE}(x, x') + \text{MSE}(z_i, z_i') + \text{MSE}(x_i, x_i') + \text{MSE}(p_z, p_z') + \text{MSE}(p_x, p_x')$$

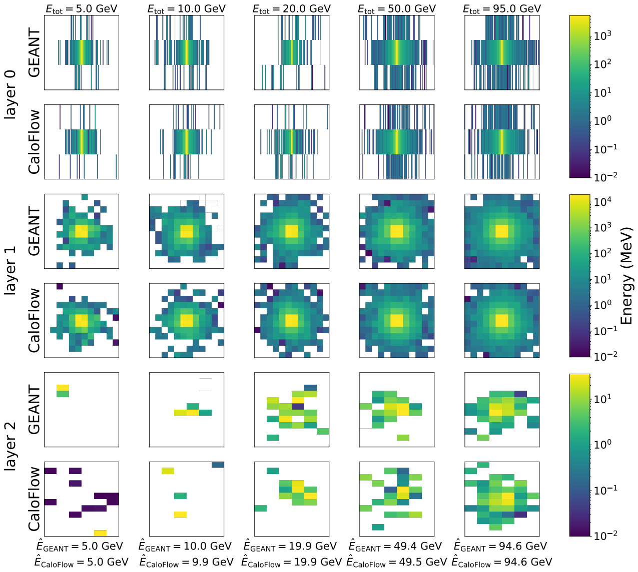
Nearest Neighbors: e^+ (student)



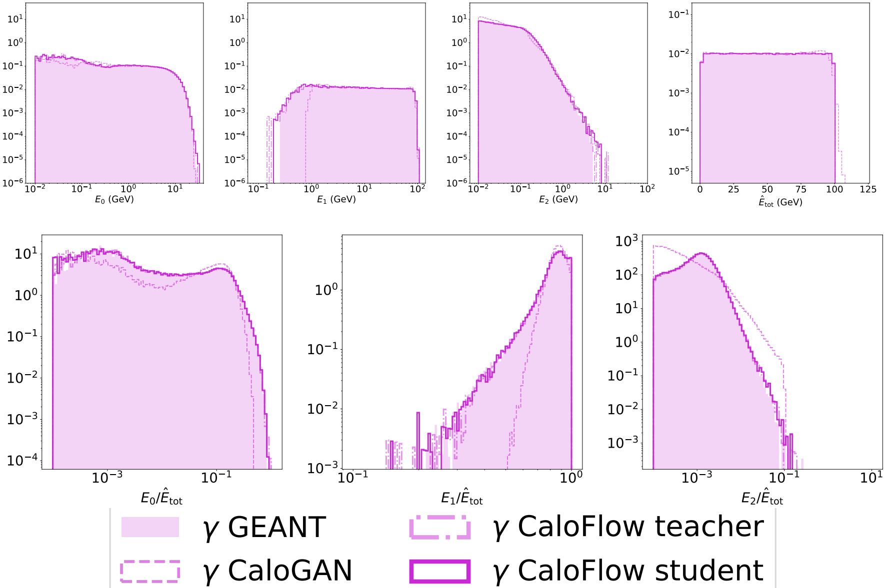
Comparing Shower Averages: γ



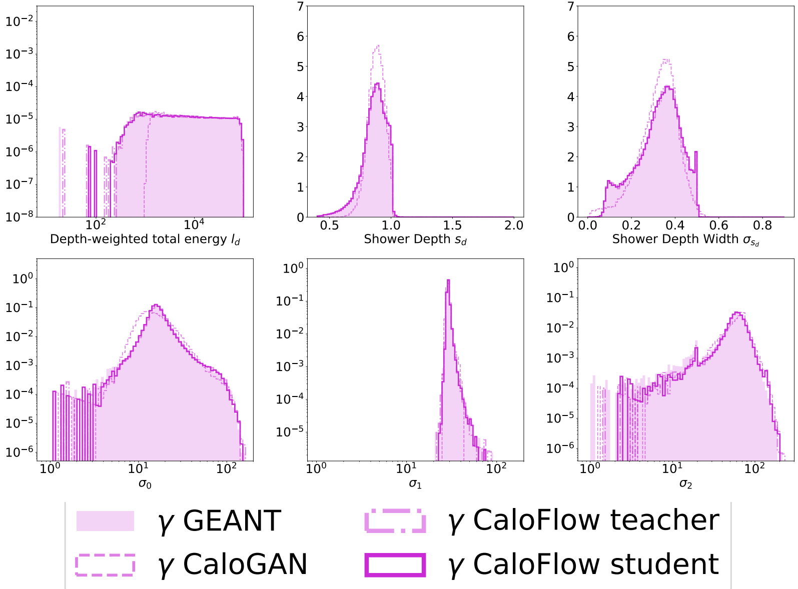
Nearest Neighbors: γ (student)



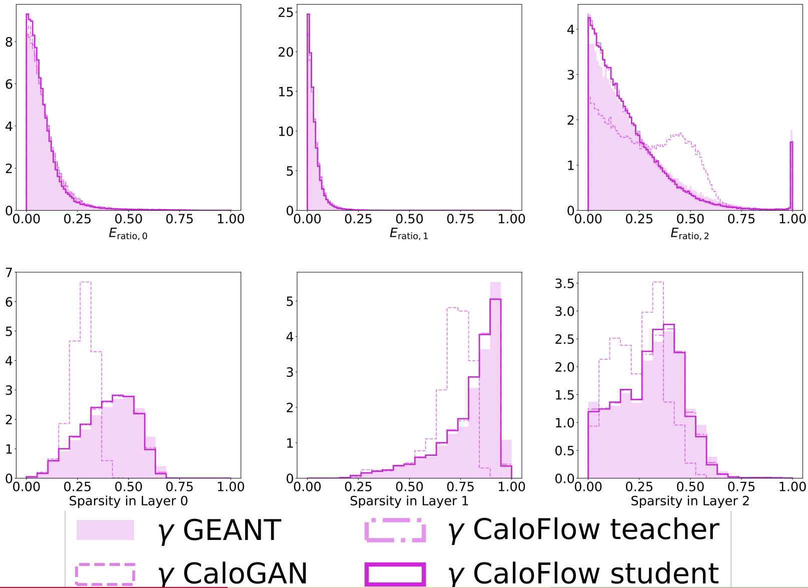
Flow I histograms: γ



Flow I+II histograms: γ



Flow II histograms: γ



Comparing Shower Averages: π^+