CaloFlow for *CaloChallenge*

Ian Pang

Rutgers, The State University of New Jersey

November 2, 2022

Based on work in collaboration with
M. Buckley, C. Krause and D. Shih

Motivation

- Generating calorimeter showers with GEANT4 is major computational bottleneck at LHC
- Urgent need for fast and accurate calorimeter simulation
- Developed methods based on normalizing flows and applied them to *CaloChallenge* datasets

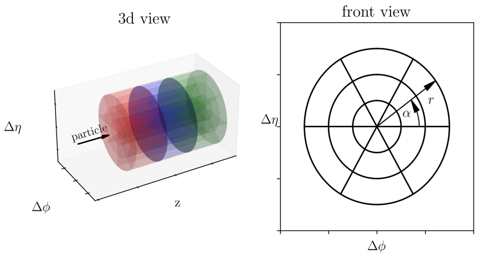
Outline

1. CaloChallenge
2. CaloFlow on DS 1
   - Method
   - Results
3. iCaloFlow on DS 2 & 3
   - Method
   - Results (preliminary)

CaloChallenge 2022

- 3 datasets with increasing dimensionality

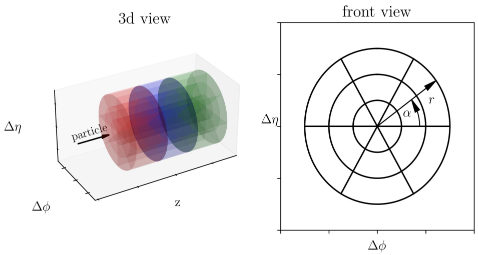
M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska
https://calochallenge.github.io/homepage/



CaloChallenge 2022

- 3 datasets with increasing dimensionality
- CaloFlow training time scales badly with N_{voxels}

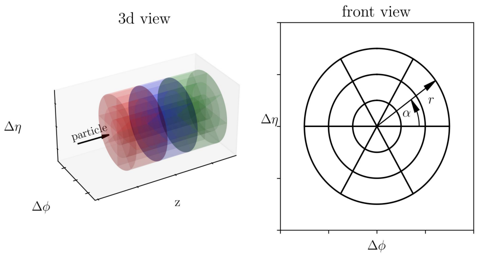
M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska
https://calochallenge.github.io/homepage/



CaloChallenge 2022

- 3 datasets with increasing dimensionality
- CaloFlow training time scales badly with N_{voxels}
- CaloFlow applied to Dataset 1 (low dim)
  - 5(7) layers for photons(pions)
  - 368(533) voxels for photons(pions)

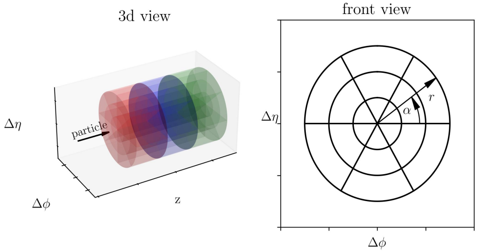
M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska
https://calochallenge.github.io/homepage/



CaloChallenge 2022

- 3 datasets with increasing dimensionality
- CaloFlow training time scales badly with N_{voxels}
- CaloFlow applied to Dataset 1 (low dim)
  - 5(7) layers for photons(pions)
  - 368(533) voxels for photons(pions)
- iCaloFlow applied to Datasets 2 and 3 (high dim)
  - 45 layers each for Datasets 2 and 3
  - 6480(40500) voxels for Dataset 2(3)

M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska
https://calochallenge.github.io/homepage/



CaloFlow (Low dimensional dataset)

Goal: Learn $p(\vec{\mathcal{I}}|E_{\text{inc}})$

2-flow process

CaloFlow (Low dimensional dataset)

Goal: Learn $p(\vec{\mathcal{I}}|E_{\text{inc}})$

2-flow process

Flow-I

- Learns $p_1(E_i|E_{\text{inc}})$
- is MAF trained with LL

CaloFlow (Low dimensional dataset)

Goal: Learn $p(\vec{\mathcal{I}}|E_{\text{inc}})$

2-flow process

Flow-I

- Learns $p_1(E_i|E_{\text{inc}})$
- is MAF trained with LL

Flow-II

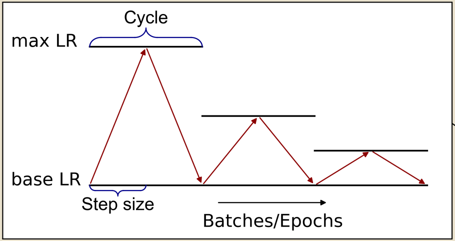
- Learns $p_2(\vec{\mathcal{I}}|E_i, E_{\text{inc}})$
- Teacher (MAF) - Slow in sampling; Fast in density estimation
- Student (IAF) - Fast in sampling; Slow in density estimation

Main updates to CaloFlow

Cyclic LR

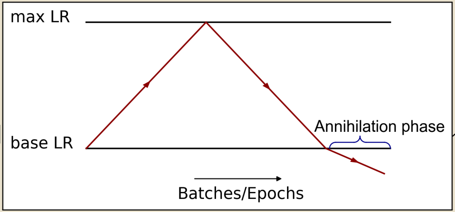
Regular:

Smith [IEEE, 2017]



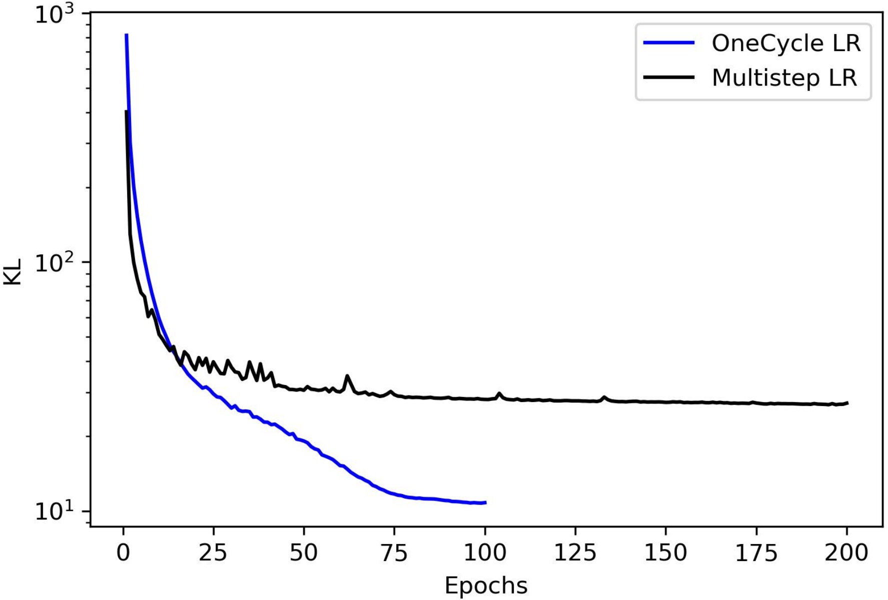
OneCycle:

Smith et al. [SPIE, 2019]



Fewer epochs required for lower loss

Main updates to CaloFlow



Classifier scores: "ultimate metric" test

- According to Neyman-Pearson lemma, we have $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$ if an optimal classifier cannot distinguish between the two datasets.
- Trained binary classifier directly on low-level and high-level features of CaloFlow and Geant4 samples.

Classifier scores: "ultimate metric" test

- According to Neyman-Pearson lemma, we have $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$ if an optimal classifier cannot distinguish between the two datasets.
- Trained binary classifier directly on low-level and high-level features of CALOFLOW and GEANT4 samples.

- Low-level features:
  1. Voxel energies $\vec{\mathcal{I}}$
  2. Incident energies E_{inc}
- High-level features:
  1. Incident energies E_{inc}
  2. Layer energies E_i
  3. Centers of energy in η and ϕ directions + their widths

Classifier scores: "ultimate metric" test

AUC / JSD		DNN based classifier	
		GEANT4 vs. CALOFLOW (teacher)	GEANT4 vs. CALOFLOW (student)
γ	low-level	0.701(3) / 0.092(3)	0.739(3) / 0.131(4)
	high-level	0.551(3) / 0.013(2)	0.556(3) / 0.015(2)
π^+	low-level	0.779(1) / 0.185(2)	0.854(3) / 0.313(6)
	high-level	0.698(2) / 0.104(3)	0.726(3) / 0.128(3)

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

JSD ($\in [0, 1]$): Jensen-Shannon divergence based on binary cross entropy

Classifier scores: “ultimate metric” test

AUC / JSD		DNN based classifier	
		GEANT4 vs. CALOFLOW (teacher)	GEANT4 vs. CALOFLOW (student)
γ	low-level	0.701(3) / 0.092(3)	0.739(3) / 0.131(4)
	high-level	0.551(3) / 0.013(2)	0.556(3) / 0.015(2)
π^+	low-level	0.779(1) / 0.185(2)	0.854(3) / 0.313(6)
	high-level	0.698(2) / 0.104(3)	0.726(3) / 0.128(3)

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

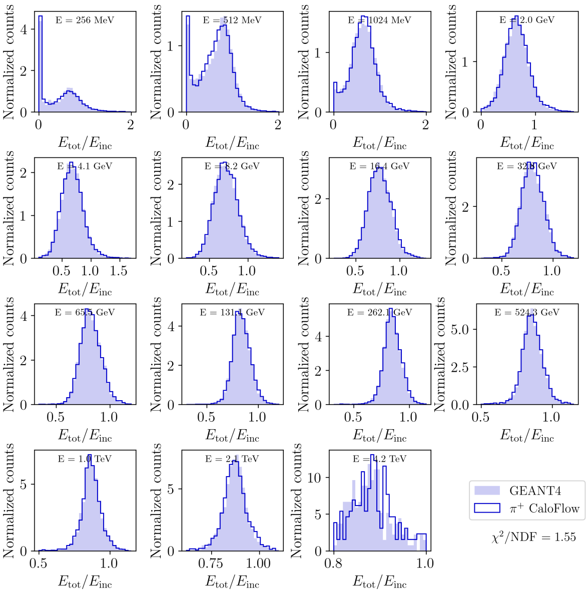
JSD ($\in [0, 1]$): Jensen-Shannon divergence based on binary cross entropy

All AUC and JSD much below 1 (less is better) $\implies$ High-fidelity!

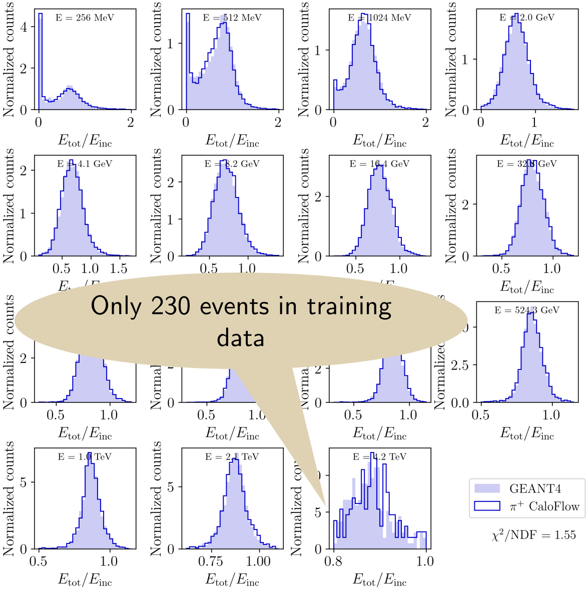
In comparison, CALOSCORE has AUC $= 0.98$ for low-level features

Mikuni et al. [arXiv:2206.11898]

π^+ $E_{\text{tot}}/E_{\text{inc}}$

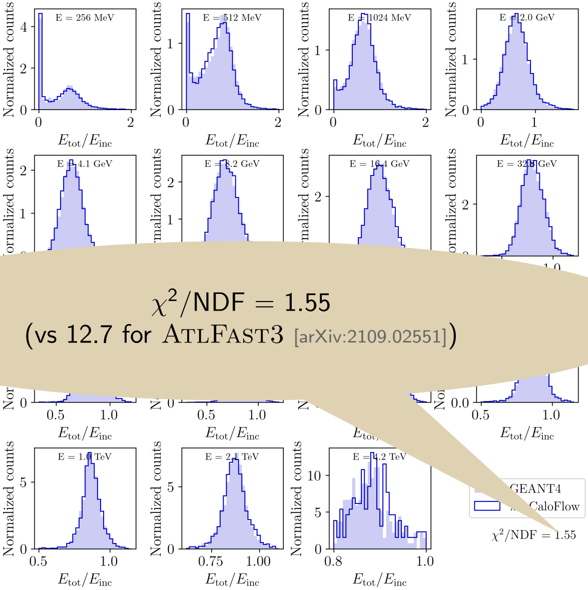


π^+ $E_{\text{tot}}/E_{\text{inc}}$



Only 230 events in training data

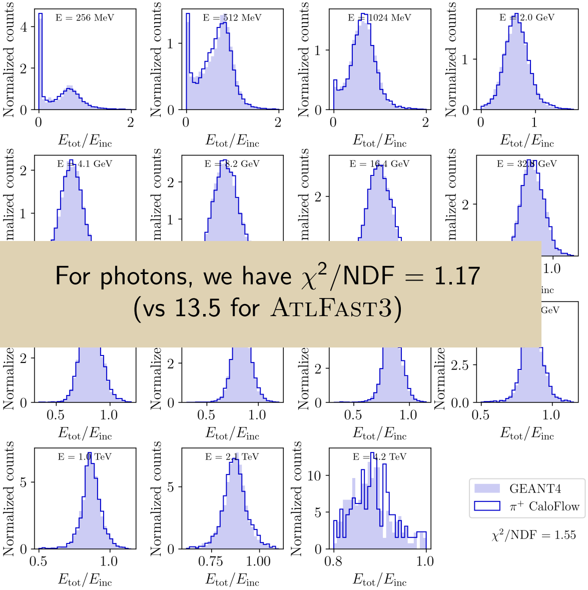
π^+ $E_{\text{tot}}/E_{\text{inc}}$



χ^2/NDF $= 1.55$

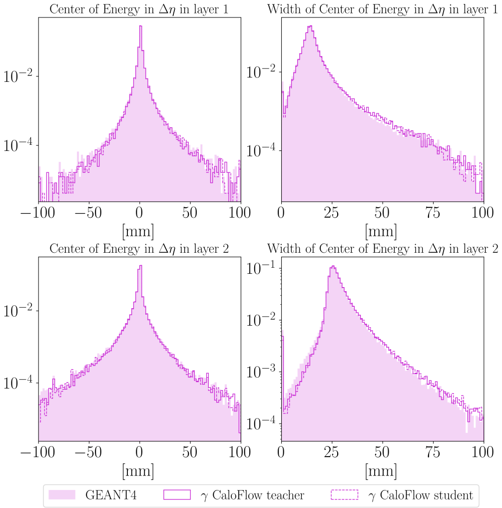
(vs 12.7 for AtlFast3 [arXiv:2109.02551])

π^+ $E_{\text{tot}}/E_{\text{inc}}$

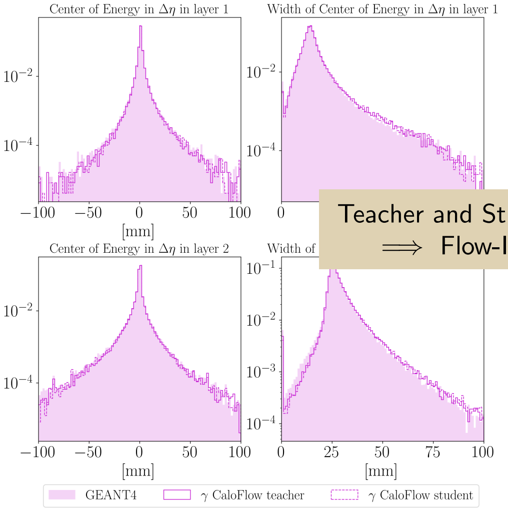


For photons, we have $\chi^2/\text{NDF} = 1.17$
(vs 13.5 for AtlFast3)

γ shower shape

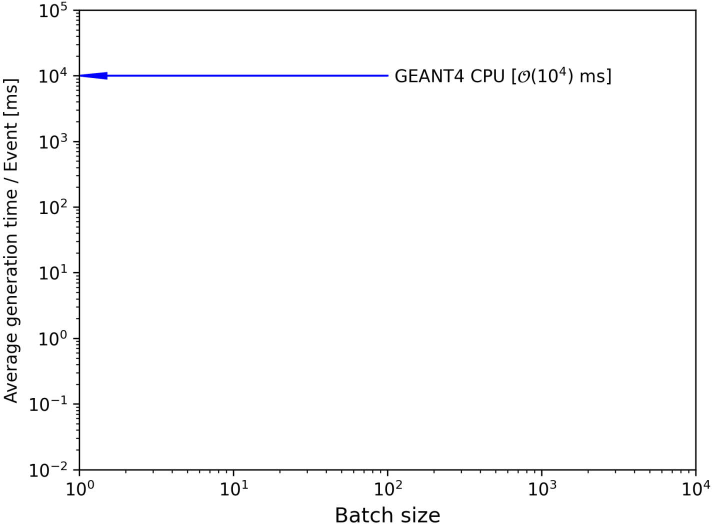


γ shower shape

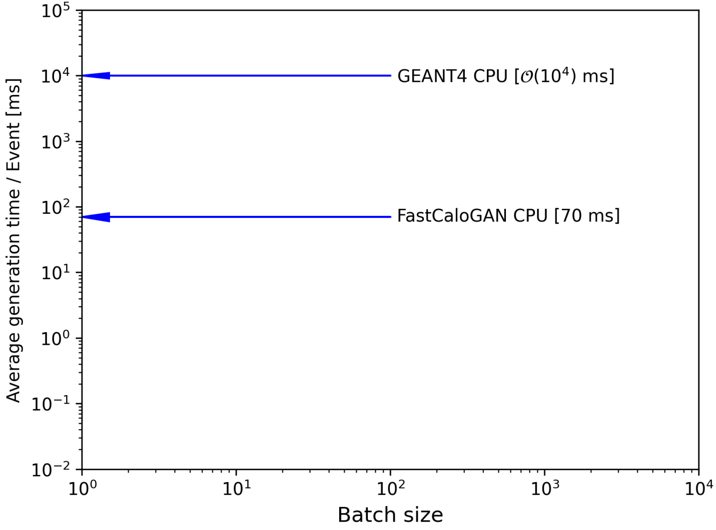


Teacher and Student overlapped
$\implies$ Flow-II well-trained

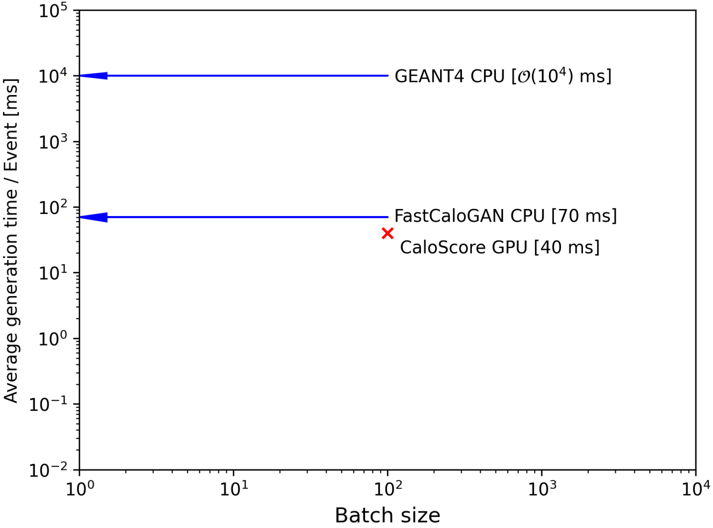
Generation Timing



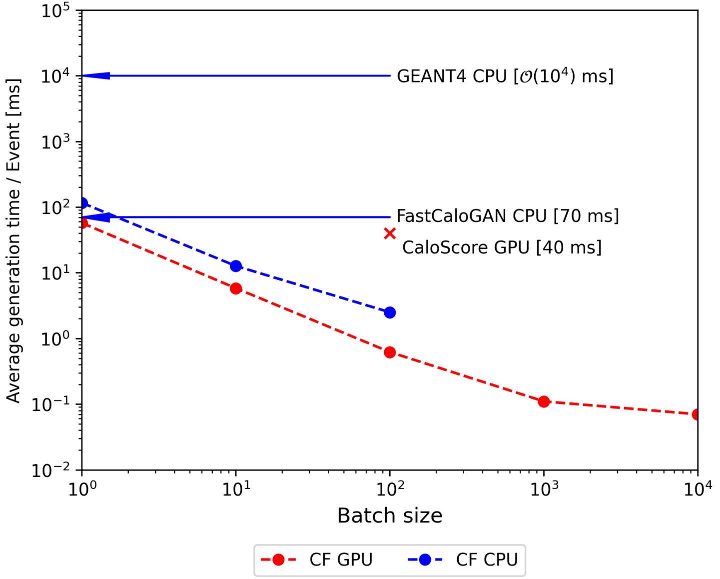
Generation Timing



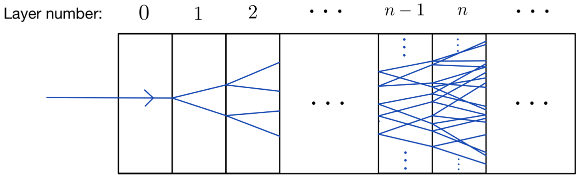
Generation Timing



Generation Timing

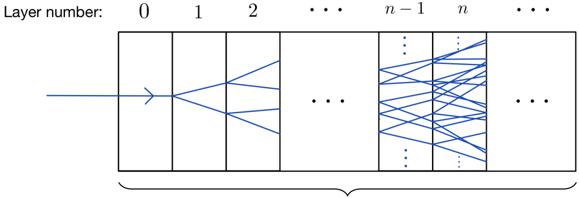


ICaloFlow (High dimensional datasets)



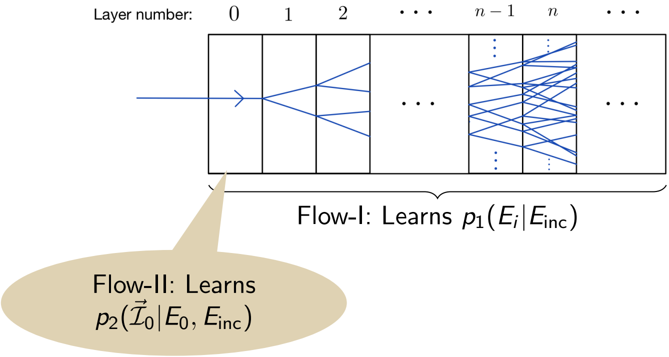
- DS 2 has $\mathcal{O}(10)$ more voxels than DS 1
- DS 3 has $\mathcal{O}(100)$ more voxels than DS 1
- Training time $\propto N_{\text{voxels}} \implies$ Inefficient to directly apply CaloFlow!
- Here we need ICaloFlow!

ICaloFlow (High dimensional datasets)

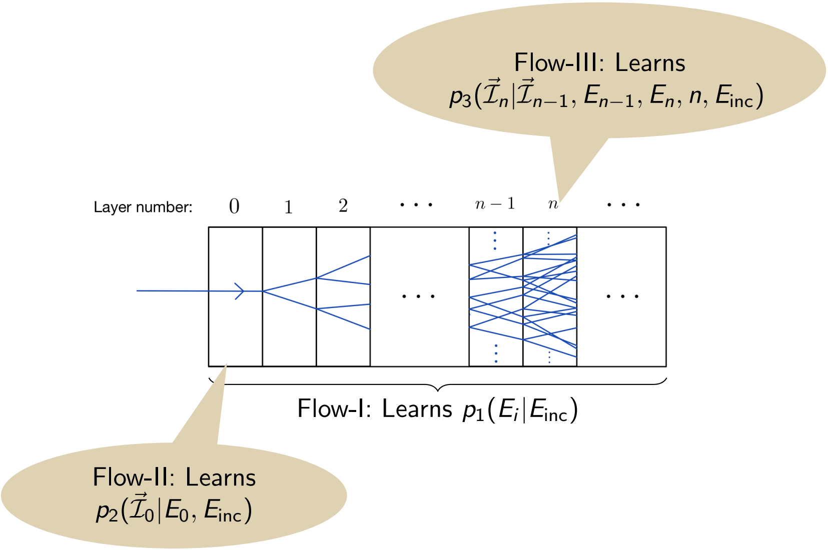


Flow-I: Learns $p_1(E_i|E_{\mathrm{inc}})$

iCaloFlow (High dimensional datasets)



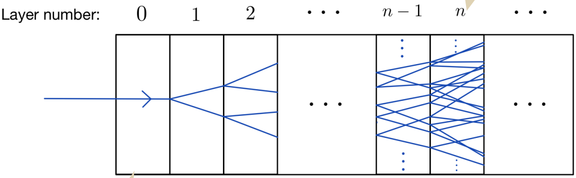
iCaloFlow (High dimensional datasets)



iCaloFlow (High dimensional datasets)

Flow-III is trained simultaneously over $\vec{\mathcal{I}}_n$ for $n > 0$. Saves on memory footprint.

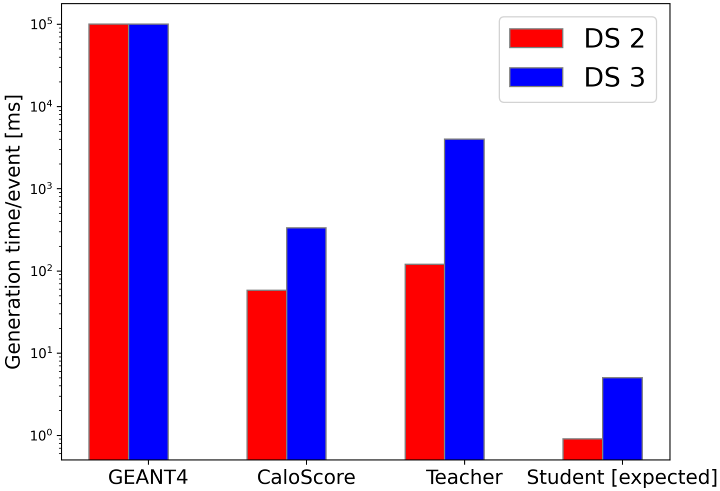
Flow-III: Learns $p_3(\vec{\mathcal{I}}_n | \vec{\mathcal{I}}_{n-1}, E_{n-1}, E_n, n, E_{\text{inc}})$



Flow-I: Learns $p_1(E_i | E_{\text{inc}})$

Flow-II: Learns $p_2(\vec{\mathcal{I}}_0 | E_0, E_{\text{inc}})$

Generation Timing

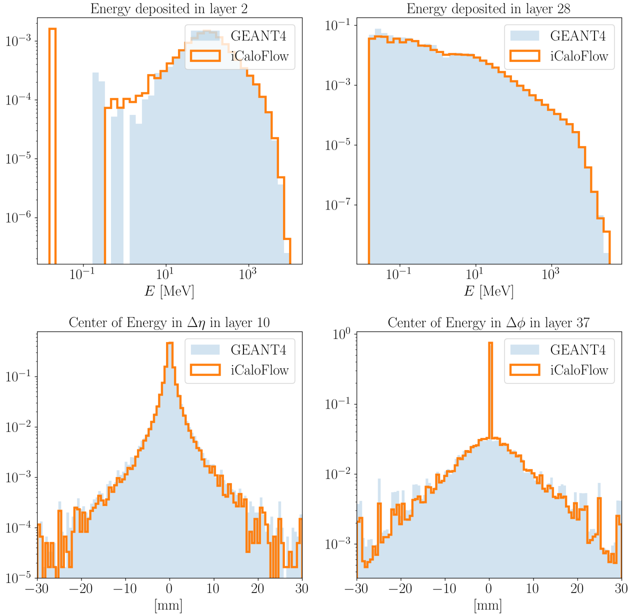


Classifier scores (preliminary)

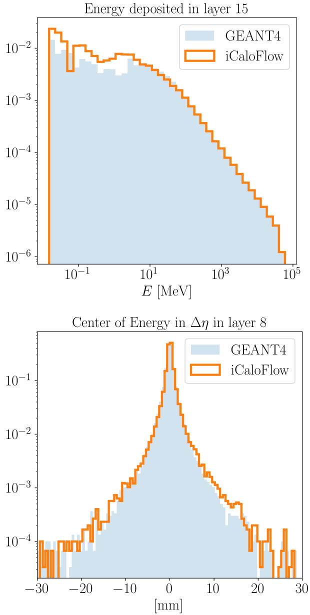
AUC / JSD		DNN based classifier GEANT4 vs. iCALOFLOW (teacher)
DS 2	low-level	0.823(3)/ 0.263(6)
	high-level	0.860(2)/ 0.329(5)
DS 3	low-level	0.8892 / 0.4112
	high-level	0.9306 / 0.5239

In comparison, CALOSCORE has AUC $= 0.98$ for low-level features in both DS.

Datasets 2 histograms (preliminary)



Datasets 3 histograms (preliminary)



Summary

(1) CaloFlow on DS 1 [arXiv:2210.14245]

- Updates to CaloFlow
- Good performance on DS 1 (histograms, χ^2/NDF, classifier scores)
- $\mathcal{O}(10^2) - \mathcal{O}(10^5)$ speed up compared to Geant4

Summary

(1) CaloFlow on DS 1 [arXiv:2210.14245]

- Updates to CaloFlow
- Good performance on DS 1 (histograms, χ^2/NDF, classifier scores)
- $\mathcal{O}(10^2) - \mathcal{O}(10^5)$ speed up compared to Geant4

(2) iCaloFlow on DS 2 & 3

- Inductively learn each layer (Good for high dim)
- Promising results for teacher (MAF)
- Future work: Train student model to speed up sample generation

Backup

Dataset 1

1. Voxelized version of ATLAS detector config with $\eta \in [0.2, 0.25]$
2. Used to train FastCaloGAN of AtlFast3 [2109.02551, Comput.Softw.Big Sci.]
3. 121000 photon showers & 120230 charged pion showers

10.5281/zenodo.6368338

	Number of voxels	Number of layers
γ	368	5
π^+	533	7

CaloFlow algorithm works well here!

Datasets 2 and 3

100k electron showers
in simulated detector

Dataset 2

1. 45 calorimeter layers
2. 144 voxels per layer
3. $144 \times 45 =$ 6480 voxels!
4. $\mathcal{O}(10)$ more voxels than DS 1

10.5281/zenodo.6366271

Dataset 3

1. 45 calorimeter layers
2. 900 voxels per layer
3. $900 \times 45 =$ 40500 voxels!
4. $\mathcal{O}(100)$ more voxels than DS 1

10.5281/zenodo.6366324

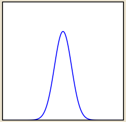
Here we need inductive CaloFlow (iCaloFlow)!

Normalizing flows in generative modelling

Basic idea

Efficient coordinate transformations (Think tractable Jacobian!)

"easy" base distribution

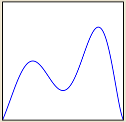


$\Leftrightarrow$

bijective transformation

$\Leftrightarrow$

"target" distribution



$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

$\xleftarrow{\text{density estimation, } p(x)}$

$\xrightarrow{\text{sample generation}}$

Normalizing flows in generative modelling

- Rational Quadratic Splines (RQS) chosen as transformations
  Durkan et al. [arXiv:1906.04032], Gregory/Delbourgo [IMA J. of Num. An., '82]
- NFs learn parameters θ of a composition of RQS
  Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]
- Autoregressive architecture (MAF/IAF) ensures triangular Jacobian for fast evaluation

MAF (*Teacher*)

Papamakarios et al. [arXiv:1705.07057]

- slow in sampling
- fast in density estimation ✓
- trained with LL

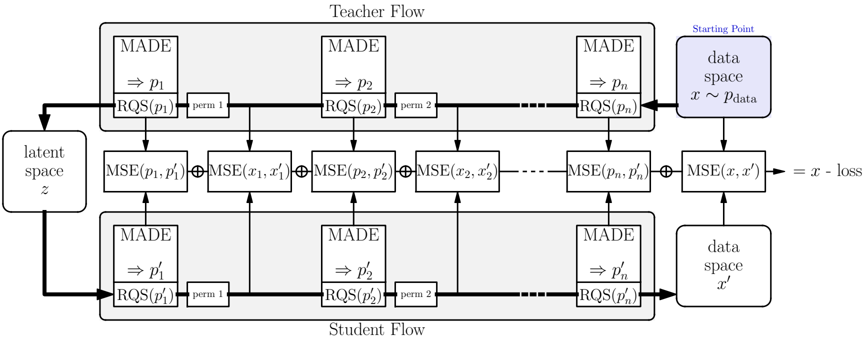
IAF (*Student*)

Kingma et al. [arXiv:1606.04934]

- fast in sampling ✓
- slow in density estimation
- trained with PDD

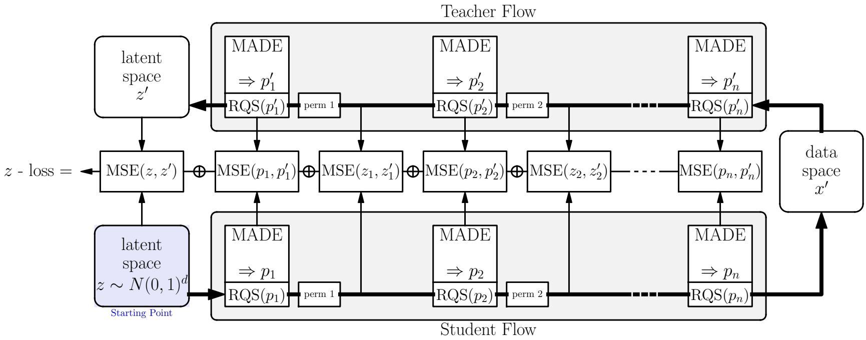
van den Oord et al. [arXiv:1711.10433]

PDD (x-loss)



Teacher Flow
Starting Point
MADE
$\Rightarrow p_1$
MADE
$\Rightarrow p_2$
MADE
$\Rightarrow p_n$
data space $x \sim p_{\text{data}}$
$\text{RQS}(p_1)$
perm 1
$\text{RQS}(p_2)$
perm 2
$\text{RQS}(p_n)$
latent space z
$\text{MSE}(p_1, p_1')$
$\text{MSE}(x_1, x_1')$
$\text{MSE}(p_2, p_2')$
$\text{MSE}(x_2, x_2')$
$\text{MSE}(p_n, p_n')$
$\text{MSE}(x, x')$
$= x$ - loss
MADE
$\Rightarrow p_1'$
MADE
$\Rightarrow p_2'$
MADE
$\Rightarrow p_n'$
data space x'
$\text{RQS}(p_1')$
perm 1
$\text{RQS}(p_2')$
perm 2
$\text{RQS}(p_n')$
Student Flow

PDD (z-loss)



Teacher Flow
latent space z'
MADE $\Rightarrow p'_1$
RQS(p'_1)
perm 1
MADE $\Rightarrow p'_2$
RQS(p'_2)
perm 2
MADE $\Rightarrow p'_n$
RQS(p'_n)
z - loss =
MSE(z, z')
MSE(p_1, p'_1)
MSE(z_1, z'_1)
MSE(p_2, p'_2)
MSE(z_2, z'_2)
MSE(p_n, p'_n)
data space x'
latent space $z \sim N(0,1)^d$
Starting Point
MADE $\Rightarrow p_1$
RQS(p_1)
perm 1
MADE $\Rightarrow p_2$
RQS(p_2)
perm 2
MADE $\Rightarrow p_n$
RQS(p_n)
Student Flow

Training time

		Number of training epochs (Training time)	
		Teacher	Student
γ	flow-I	100 (46 min)	-
	flow-II	100 (77 min)	100 (360 min)
π^+	flow-I	100 (52 min)	-
	flow-II	100 (119 min)	150 (658 min)

Main updates to CaloFlow

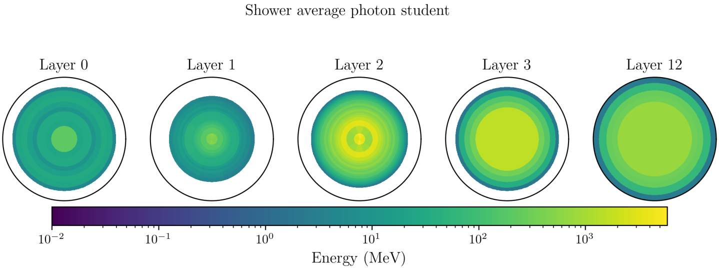
(1) Unit-space definition

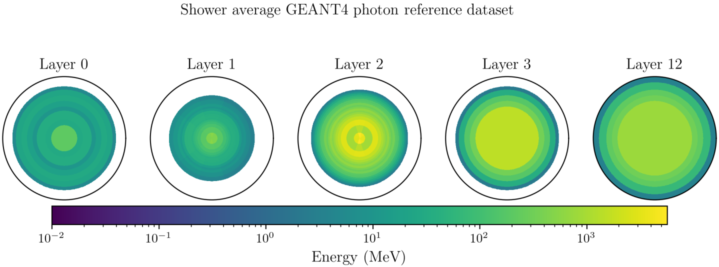
- Layer energies transformed to $\vec{u} \in [0,1]^{N_{\text{layers}}}$
- $u_0 = \frac{\sum_{i=0}^{N_{\text{layers}}} E_i}{E_{\text{inc}}}$ is sometimes > 1
- Solution: Rescale by $\max(u_0)$

(2) π^+ flow-I noise level

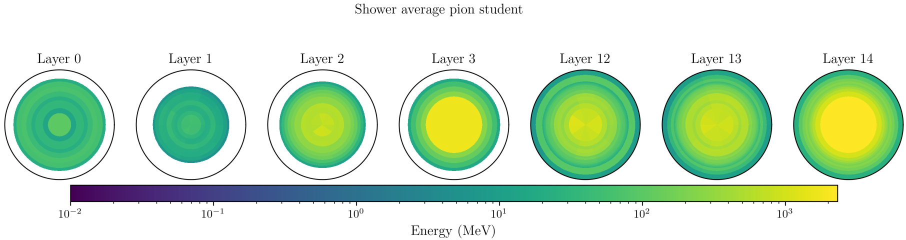
- Higher noise level $\rightarrow$ Poor performance learning π^+ layer energies
- Solution: Use lower noise level for π^+ flow-I

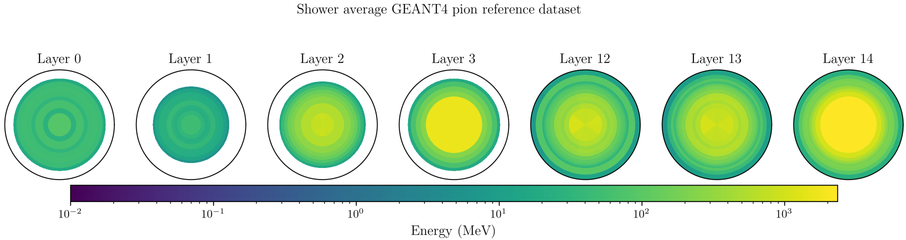
γ Shower images



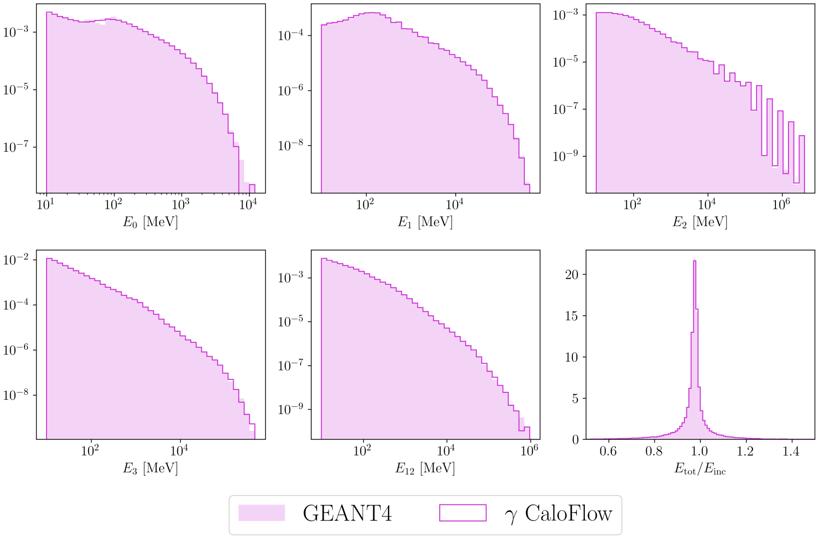


π^+ Shower images

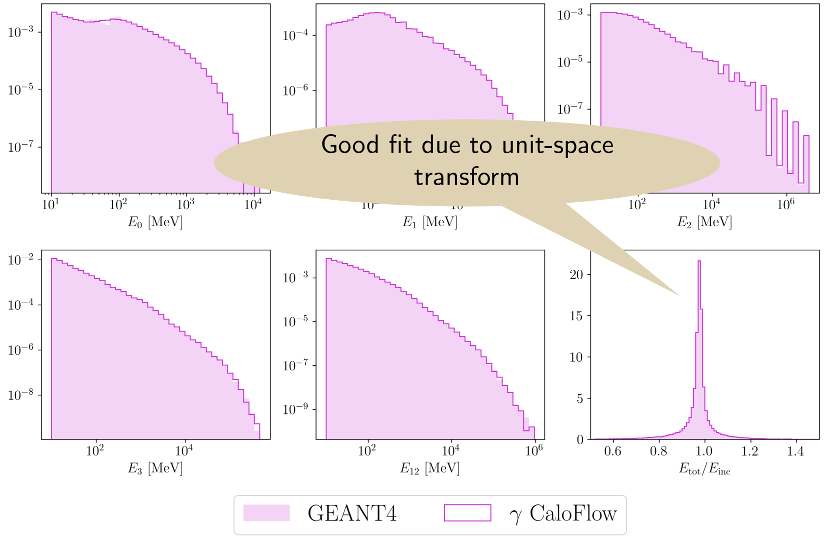




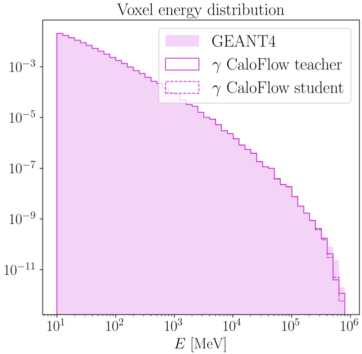
γ layer energies

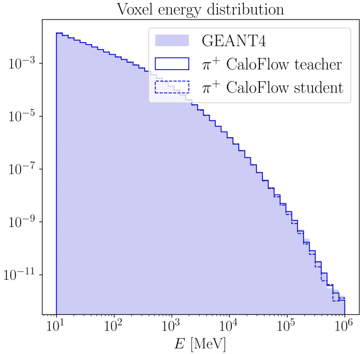


γ layer energies

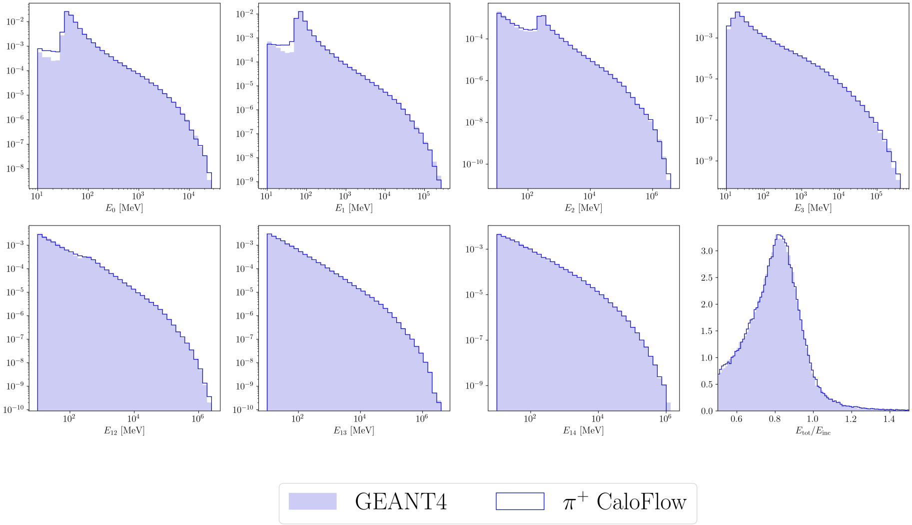


Voxel energy distribution

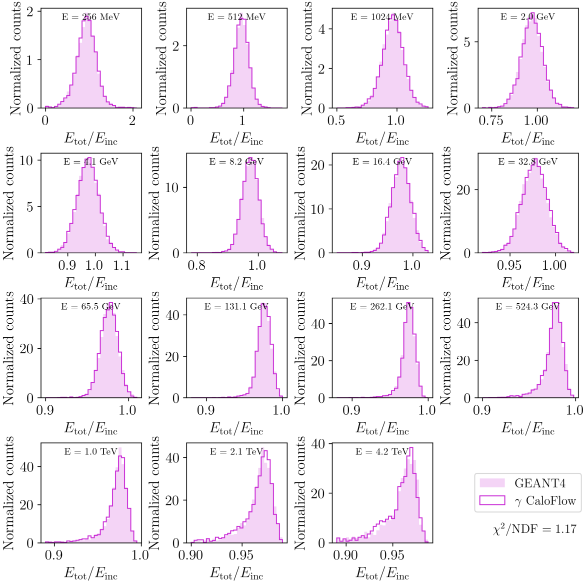




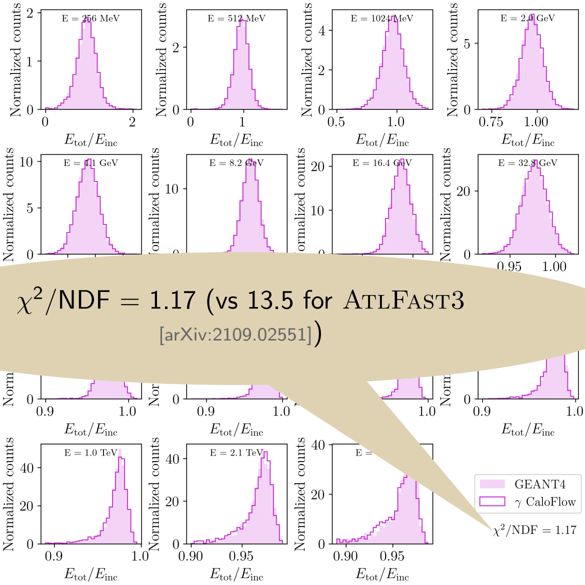
π^+ layer energies



γ $E_{\mathrm{tot}}/E_{\mathrm{inc}}$

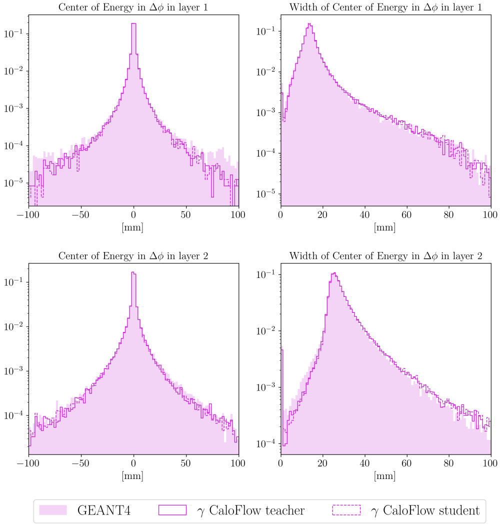


γ $E_{\text{tot}}/E_{\text{inc}}$

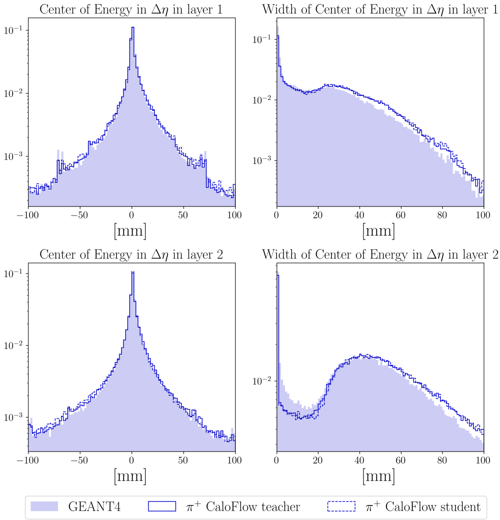


χ^2/NDF = 1.17 (vs 13.5 for ATLFAST3 [arXiv:2109.02551])

γ shower shape



π^+ shower shape



π^+ shower shape