CaloFlow for CaloChallenge dataset 1

Claudius Krause[1,2], Ian Pang[1] and David Shih[1]

1 NHETC, Dept. of Physics and Astronomy, Rutgers University, Piscataway, NJ, USA
2 Institut für Theoretische Physik, Universität Heidelberg, Germany

Abstract

CALOFLOW is a new and promising approach to fast calorimeter simulation based on normalizing flows. Applying CALOFLOW to the photon and charged pion GEANT4 showers of Dataset 1 of the Fast Calorimeter Simulation Challenge 2022, we show how it can produce high-fidelity samples with a sampling time that is several orders of magnitude faster than GEANT4. We demonstrate the fidelity of the samples using calorimeter shower images, histograms of high level features, and aggregate metrics such as a classifier trained to distinguish CALOFLOW from GEANT4 samples.

doi:[10.21468/SciPostPhys.16.5.126](#)

1   Introduction

The LHC has just restarted for its third run in the spring of 2022, and the need for accurate fast simulation at the LHC is ever more urgent. Simulation of calorimeter showers with GEANT4 [1–3] is already a major computational bottleneck at the LHC, and this is expected to further intensify as the detectors are upgraded and the luminosity is increased [4].

Recently, there have been significant advancements in fast calorimeter simulation through the application of deep generative models such as Generative Adversarial Networks (GANs), normalizing flows, Variational Autoencoders (VAEs) and other approaches [5–20] In CALOFLOW-I [9] and CALOFLOW-II [10], two of us have demonstrated the effectiveness of normalizing flows in emulating GEANT4 events with high-fidelity at generation speeds that are 10^4 times faster than GEANT4. Also, we found that CALOFLOW is robust against mode collapse that is a common problem in GAN-based generative models. In fact, CALOFLOW proved to be the first ever generative model in HEP that could pass the following stringent test: a binary classifier trained on the raw voxels of (CALOFLOW) generated vs. (GEANT4) reference showers could not distinguish the two with 100% accuracy. Normalizing Flows showed similar good performance in other generative tasks in high-energy physics [21–34].

In this paper, we adapt CALOFLOW to Dataset 1 from the *Fast Calorimeter Simulation Challenge 2022* [35] (hereafter referred to as the *CaloChallenge*). The *CaloChallenge* is aimed at encouraging the development of fast and high-fidelity calorimeter shower generation through the application of deep generative models. It includes three datasets with increasing dimensionality. Dataset 1 [36] has the lowest dimensionality and consists of photon γ (368 voxels) and charged pion π^+ (533 voxels) showers. This dataset is actually the official ATLAS simulation dataset used to develop the FASTCALOGAN model [8] used in a portion of ATLFAST3 [7], which is currently the official fast calorimeter simulation framework of the ATLAS collaboration. It is more realistic than the GEANT4 dataset [37] used in [5,6,9,10] because it includes a realistic sampling fraction and calorimeter geometry.

We will demonstrate the robustness of CALOFLOW even when applied to this new dataset. We will show that CALOFLOW can achieve comparably fast generation times as FASTCALOGAN, with a significant improvement in generation quality. We will also show that CALOFLOW compares favorably (on Dataset 1), in both generation time and quality, to CALOSCORE [12]. This is a recent approach that uses score-based diffusion models and is currently the only approach that has been trained successfully on all three datasets of the *CaloChallenge*.

The outline of the paper is as follows. In Section 2, we provide a brief description of *CaloChallenge* Dataset 1 (for more details, see [35]) and outline the differences compared to the CALOGAN dataset. In Section 3, we elaborate on the method we used to learn the distribution of showers from the GEANT4 reference datasets and the modifications we have made to adapt the previous version of CALOFLOW to *CaloChallenge* Dataset 1. We include the main results of the study in Section 4. Here we show detailed comparison between the CALOFLOW generated samples and the reference GEANT4 samples. Finally, we summarize our findings in Section 5 and include possibilities for future research.

2   Dataset

As stated in Section 1, there are three datasets in the *CaloChallenge* and our study focuses only on Dataset 1 [36], as the original version of CALOFLOW does not scale well to the dimensionalities of datasets 2 and 3. This dataset comprises calorimeter shower events for γ and π^+ particle types and is a subset of the ATLAS GEANT4 datasets that are published at [38]. The datasets described here have been used to train the ATLAS GAN-based model FASTCALOGAN [8] used in ATLFAST3 [7].

Table 1: Details of γ and π^+ shower datasets: N_z is the total number of calorimeter layers, N_α is the number of α bins in a given layer, and N_r is the number of radial bins in a given layer. N_α and N_r are listed according to increasing layer numbers (i.e. the number of α/r bins in layer 0 are leftmost on the list).

	Number of voxels	N_z	N_α	N_r
γ	368	5	[1, 10, 10, 1, 1]	[8, 16, 19, 5, 5]
π^+	533	7	[1, 10, 10, 1, 10, 10, 1]	[8, 10, 10, 5, 15, 16, 10]

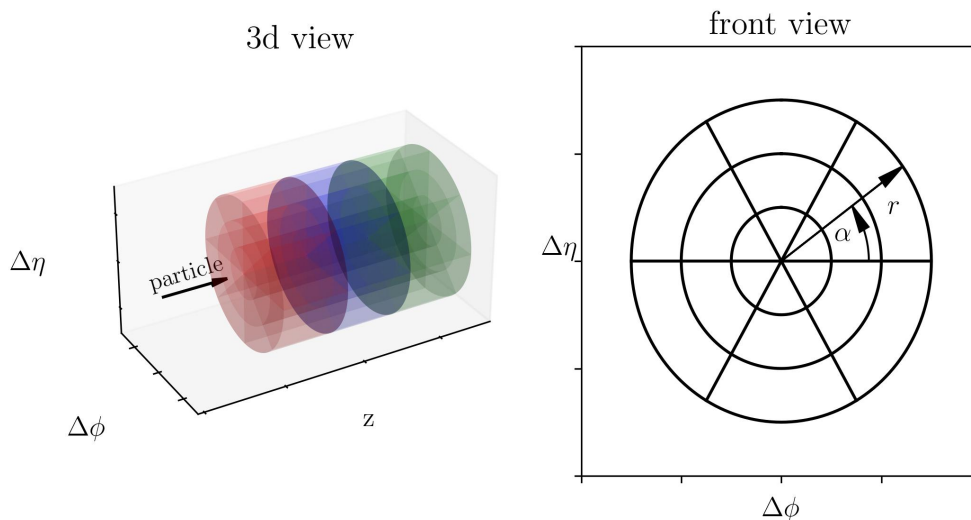



Figure 1: Diagram of coordinate system used in the *CaloChallenge* dataset [35].

The calorimeter setup is based on a voxelized version [7] of the current ATLAS detector configuration in the η range $[0.2, 0.25]$. As shown in left diagram in Figure 1, the calorimeter layers are simulated as concentric cylinders aligned along the z-axis, defined as the direction of travel of the particle. Working in cylindrical coordinates, $\Delta\phi$ and $\Delta\eta$ are defined as the x and y-axes respectively. Based on the voxelization, each calorimeter is further divided into radial and angular bins as shown in the right diagram in Figure 1. Here $r \equiv \sqrt{(\Delta\phi)^2 + (\Delta\eta)^2}$ and $\alpha \equiv \arctan\left(\frac{\Delta\eta}{\Delta\phi}\right)$. The total number of voxels, and the arrangement of voxels in z, α and r for the γ/π^+ datasets are shown in Table 1. For comparison, the CALOGAN calorimeter setup consists of 3 calorimeter layers and a total of 504 voxels regardless of the particle type.

The incident energies are discrete and range from 2^8 MeV to 2^{22} MeV, increasing in powers of 2. Hence, there are 15 different incident energies in total. (In contrast, the incident energies in the CALOGAN dataset were uniformly sampled.) The photon and pion datasets each contain 10000 events for every low incident energy. Fewer events were generated for each high incident energy due to the longer generation time with GEANT4. Figure 2 shows the distribution of incident energies in the photon and pion datasets.

In Dataset 1, there are two independent photon datasets with 121000 events each, and two independent pion datasets with 120800 events each. For both particle types, the first dataset is used for training the generative models (with a 70/30 train/val split); the second is used for training the classifier metric (with a 60/20/20 train/val/test split) and for producing all the evaluation plots. This is ideal, since then any overfitting of the density estimator to the training set could in principle be diagnosed by the independent dataset used for the classifier metric and evaluation plots.

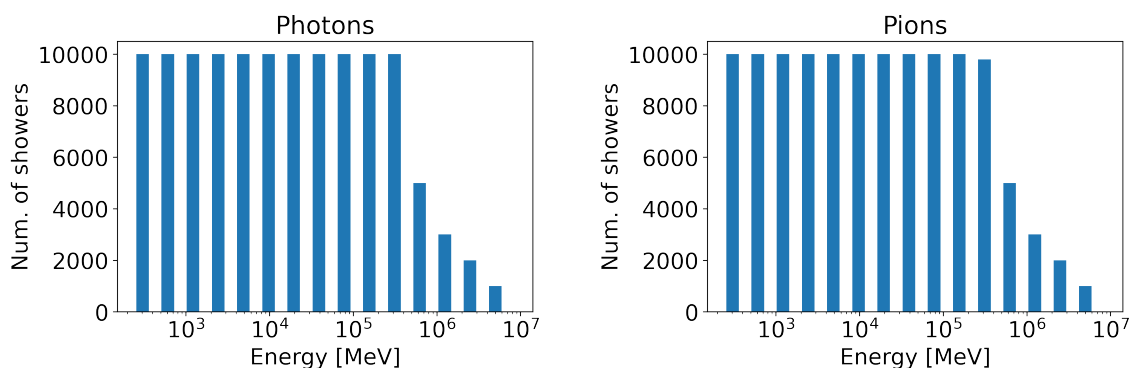



Figure 2: Breakdown of incident energies in *CaloChallenge* Dataset 1.

3 Method

Our approach follows the algorithm of CALOFLOW [9,10] to a large extent. Here we briefly describe the approach, focusing primarily on the differences with [9,10].

3.1 Normalizing flows

Normalizing Flows (NFs) (see [39–41] for reviews on NFs) are a class of machine learning models for density estimation and generative modeling that aim to learn a bijective transformation (with tractable Jacobian) between data and a latent space following a simple base distribution (e.g. the normal distribution). Since the probability of a point of the base distribution, as well as the Jacobian of the transformation is known, NFs can give the probability density $p(x)$ of each data point x in data space, i.e. NFs are density estimators. When run in the other direction, starting from samples z in the latent space generated from the base distribution, NFs can be used as generative models. Since by construction, the log-likelihood (LL) of datapoints under the flow are known, a NF can be trained by minimizing the negative LL.

Following [9,10], the NFs used here are based on the Masked Autoregressive Flow (MAF) [42] and Inverse Autoregressive Flow (IAF) [43] architectures with rational quadratic spline (RQS) transformations [44,45]. The autoregressive parameters of the RQS transformations are determined using neural networks known as MADE [46] blocks. This maximizes the expressivity of the NF, but comes at the expense of unbalanced evaluation times in the forward and inverse direction. MAFs are fast for density estimation but slow for sampling, while IAFs are fast for sampling but slow for density estimation. As in [9,10], we will refer to the MAF as the "teacher" model and the IAF as the "student" model, as the MAF was used as an intermediate step to obtain the trained IAF model, which is final product of the CALOFLOW method.

MAFs can be trained efficiently with the LL objective described above, but for IAFs this is usually not possible due to time and memory constraints. Training the IAF efficiently requires a different approach known as *Probability Density Distillation* (PDD) [47] or *teacher-student training*. Instead of fitting the IAF directly to the data, the approach involves fitting the IAF to the MAF. In practice, the fitting is implemented based on two training loss terms that we refer to as z and x-losses. To compute the z-loss, we begin with a sample z which is then passed through the student IAF to obtain a sample x' in data space and the corresponding likelihood $s(x')$. The data sample x' is then mapped via the teacher MAF back to the latent space which obtains the likelihood $t(x')$. Similarly, to compute the x-loss, one can start with a data sample x which maps to latent space z' via the teacher, and then map back to data space via the student. In the original PDD study [47] study, the KL divergence of $s(x')$ and $t(x')$ was initially used as the training loss. However, the authors noted that it does not converge well.

Hence, as in [10], we used a training loss function that is based on a mean square error that compares relevant values[1] at each equivalent stage of the teacher and student passes. The total loss function is then the sum of the x and z-losses. Such a loss function has proven effective in matching the student model to the teacher model. For more details we refer to [10].

3.2 CALOFLOW

CALOFLOW [9, 10] learns the distribution of calorimeter showers in voxel space conditioned on the incident energy, $p(\vec{\mathcal{I}}|E_{inc})$, in a two-step setup. In the first step, a normalizing flow called flow-I learns the distribution of energy depositions in the layers of the calorimeter E_i conditioned on the incident energy, $p_1(E_i|E_{inc})$. A second normalizing flow, called flow-II, learns the normalized showers conditioned on these energies, $p_2(\vec{\mathcal{I}}|E_i, E_{inc})$. Normalized in this context means that the sum of energy depositions in all voxels per calorimeter layer is 1.

When adapting CALOFLOW to *CaloChallenge* Dataset 1, we retained most of its key features, while making several important modifications that are elaborated on below:

1. **Preprocessing:**

   (a) **Unit-space definition:**

   In [9, 10], the energy deposition in the calorimeter layers were recursively transformed to $\vec{u} \in [0, 1]^{N_z}$, where u_0 (the first element in $\vec{u}$) was defined by:

   $$u_0 = \frac{\sum_{i=0}^{N_z-1} E_i}{E_{inc}}, \tag{1}$$

   In this study, we had to modify u_0 to account for cases where $\sum_{i=0}^{N_z-1} E_i > E_{inc}$:[2] We rescaled u_0 by its maximum value taken over all showers in the training set.[3] Specifically, we rescaled by $\max(u_0) = 2.42$ for γ and $\max(u_0) = 6.93$ for π^+. This rescaling of u_0 ensures that $u_0 \in [0, 1]$ for the showers of Dataset 1.

   (b) **Flow-I noise level for π^+:**

   For the flow-I of π^+ dataset, we used a smaller range for the uniform random noise $[0, 0.1]$ keV that is added to the voxel energies prior to summing them to get the layer energies. We observe that a lower noise level improved the training of π^+ flow-I. We kept the original noise range $[0, 1]$ keV when training flow-I on the γ dataset and flow-II for both particle types as lowering the noise level worsened the classifier score.

   (c) **Preprocessing of incident energies:**

   The incident energy E_{inc} is fed to flow-I and flow-II as a conditional label in the form:

   $$\log_{10}(E_{inc}/33.3 \text{ GeV}) \in [-2.5, 2.5] \tag{2}$$

   (In [9, 10], 10 GeV was used as the normalization point).

---

[1]For γ student, these consist of coordinates before and after passing them through the flows and RQS parameters from individual MADE blocks within the bijectors. For π^+ student, we did not enforce agreement with the teacher at the level of individual MADE blocks, but only at the endpoints of the flows.

[2]Naively, we might think that $E_{inc} \geq \sum_{i=0}^{N_z-1} E_i$ due to the conservation of energy. However, this is not always true due to rescaling that is done on the true deposited energy to account for the sampling fraction (< 1) of the calorimeter when creating the training/evaluation datasets.

[3]There was an exception when training on the γ dataset. We found that the event with the largest ratio of total deposited energy to incident energy was an outlier ($u_0 = 3.04$) and hence decided to ignore that single event when training our model.

(d) **Preprocessing of layer energies (Flow-II):** The preprocessing of layer energies used as conditional labels in flow-II is:

$$\log_{10}\left(\left(E_i + 1\,\text{keV}\right)/100\,\text{GeV}\right) - 1 \in [-2, 4], \tag{3}$$

which is slightly different than that of [9, 10].

2. **Number of training epochs:** See Table 2 for changes in the number of training epochs.

3. **Hyperparameters:**

(a) **Hidden layer/batch sizes:**

We experimented with different hidden layer sizes when adapting CALOFLOW to *CaloChallenge* Dataset 1. At times, larger hidden layer size leads to lower losses. However, at other times, it was surprising that the performance went down and the loss went up during training; this could be a sign of overfitting. Memory size was also a limitation when deciding on hidden layer size. Finally, we settled with the following hidden layer sizes: 378 for γ teacher; 736 for γ student; 533 for π^+ teacher; 500 for π^+ student. A summary of the MADE [46] block architecture features for the various models are shown in Table 3.

When training flow-I, we used a batch size of 200 for both particle types. When training flow-II, we used batch sizes of 500 for π^+ teacher and 200 for γ teacher. For the student models, we used the batch size of 175 as stated in [9, 10]. This choice of hyperparameters enabled us to achieve better overall classifier scores.

(b) E_{inc} **distribution in z-loss:**

Previously in [9,10], we used the same uniform E_{inc} distribution as the training data when feeding the conditional label through the student model to compute the z-loss. In general, it does not have to be the case. In this study, we have a discrete E_{inc} distribution in the training data. We found that using this same distribution works well when computing the z-loss for the γ student model. In contrast, we observed that using a uniform E_{inc} distribution that has the same range as the training data works better when computing the z-loss for the π^+ student model.

Table 2: Number of training epochs used in CALOFLOW when training on *CaloChallenge* Dataset 1. The corresponding training times, obtained on our TITAN V GPU, are included in parentheses. Note that flow-I is only trained once per dataset (γ or π^+), and then used for both the teacher and the student models.

		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 (98 min)	150 (520 min)

Table 3: Summary of MADE block architecture of the various teacher and student models. The number nodes in the input layer, hidden layer and output layer are shown for each of the 4 models.

		input	hidden	output
γ	Teacher	378	1×378	8464
	Student	736	1×736	8464
π^+	Teacher	533	1×533	12259
	Student	500	1×500	12259

4. **Learning rate (LR) schedule:**

   Instead of the simple multi-step decreasing LR schedule used in [9,10], here we adopted a cyclic LR schedule [48] (see Appendix A for details) when training flow-I and flow-II, as this was found to result in significantly improved performance. The effectiveness of cyclic LR schedule may be due the use of higher LRs at certain points in the training which helps the model move out of saddle points or local minima. One key difference in the implementation of cyclic LR is that the LR is updated after each batch in contrast to updating after each epoch milestone in [9,10]. As in [49], we observe that the OneCycle LR policy — a special case of cyclic LR which only relies on a single cycle — generally allows us to shorten the training time by obtaining a lower loss with a smaller number of training epochs. This is especially useful when training the student model (IAF) which generally has a longer training time per epoch compared to training the teacher model (MAF).

   When experimenting with LR schedules, we found that training with OneCycle LR often achieved better results compared to regular cyclic LR, so we adopted this throughout. The only exception was for γ teacher where training with regular cyclic LR had a better outcome.

4 Results

4.1 Average shower images

In Figures 3 and 4, we compare the average shower images between the samples generated by CALOFLOW and GEANT4 for the γ and π^+ datasets respectively. Looking at the average shower images for both samples, we see that they are virtually indistinguishable by eye. A more detailed comparison between the CALOFLOW and GEANT4 samples can be seen by looking at the histograms in Section 4.2.

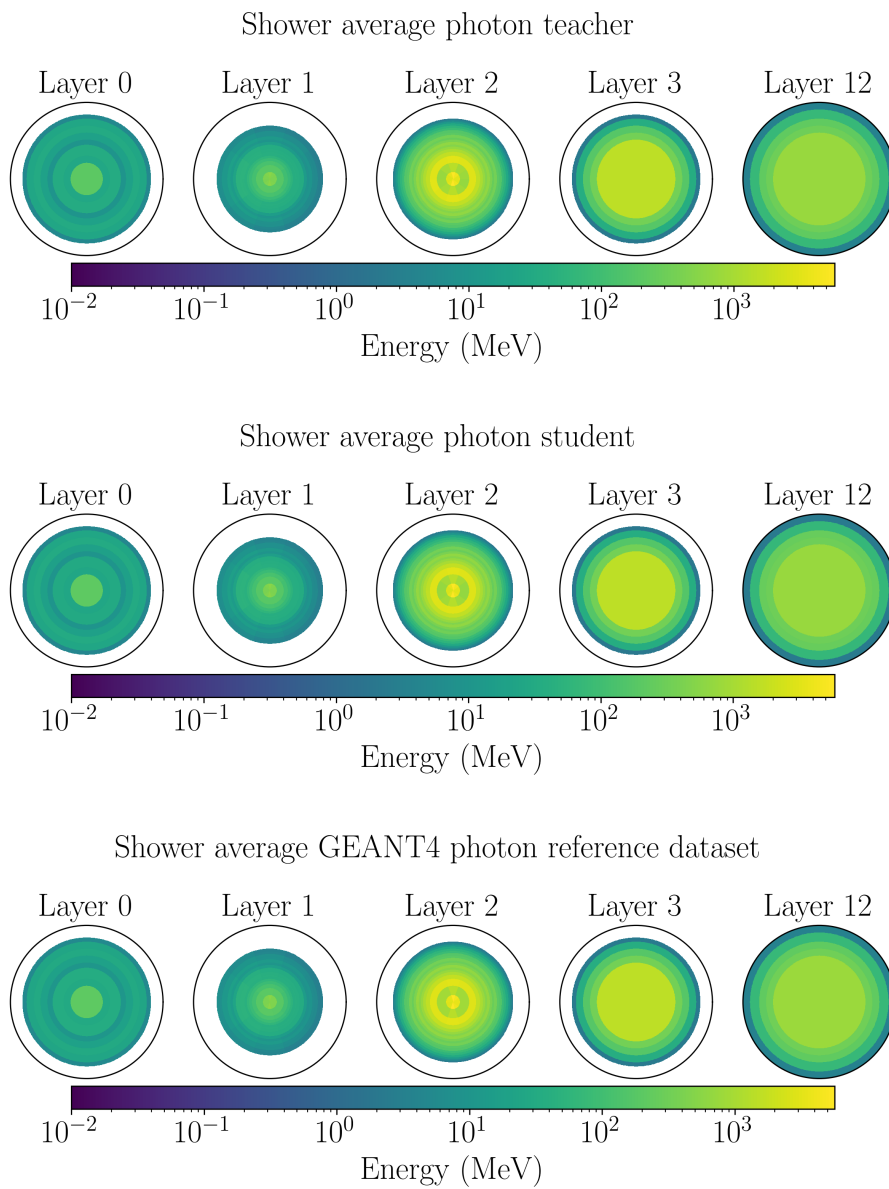



Figure 3: Shower averages for γ teacher (top), γ student (middle) and GEANT4 γ reference dataset (bottom) respectively.

Shower average pion teacher

Shower average pion student

Shower average GEANT4 pion reference dataset

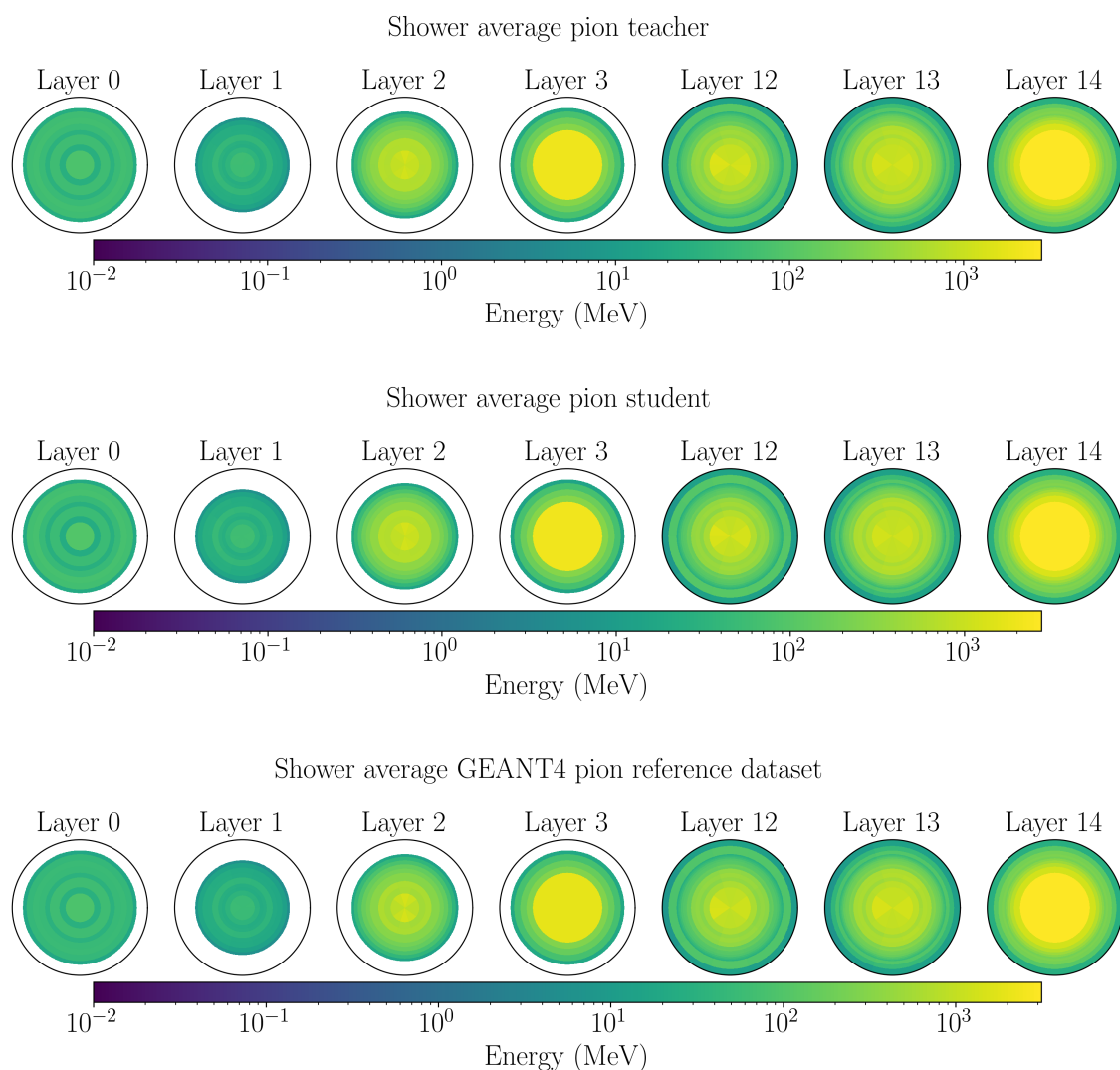



Figure 4: Shower averages for π^+ teacher (top), π^+ student (middle) and GEANT4 π^+ reference dataset (bottom) respectively.

4.2 Histograms

4.2.1 Energy histograms

Figures 5–8 show histograms corresponding to energy distributions produced by CALOFLOW compared to those from the GEANT4 reference sample. In Figure 5, we see that, despite having more calorimeter layers than before, flow-I is still able to precisely learn the γ layer energies. Also, the transformation to unit-space in the preprocessing ensured that $E_{\text{tot}}/E_{\text{inc}}$ is learned well and this is clearly reflected in the $E_{\text{tot}}/E_{\text{inc}}$ histogram at the bottom right of Figure 5. We are pleasantly surprised at how well flow-I was able to learn the spikes found at higher energies in the E_2 plot. There are some minor discrepancies at higher energies which can be attributed to there being fewer events with high E_{inc} as shown in Figure 2.

Similarly, we see from Figure 6 that flow-I was able to learn the π^+ layer energies very accurately. Initially, we found that flow-I struggled with learning the bimodal distribution found in the first 3 layer energy histograms (top row in Figure 6). However, training flow-I with a lower noise level helped it to better learn complicated distributions in the π^+ dataset.

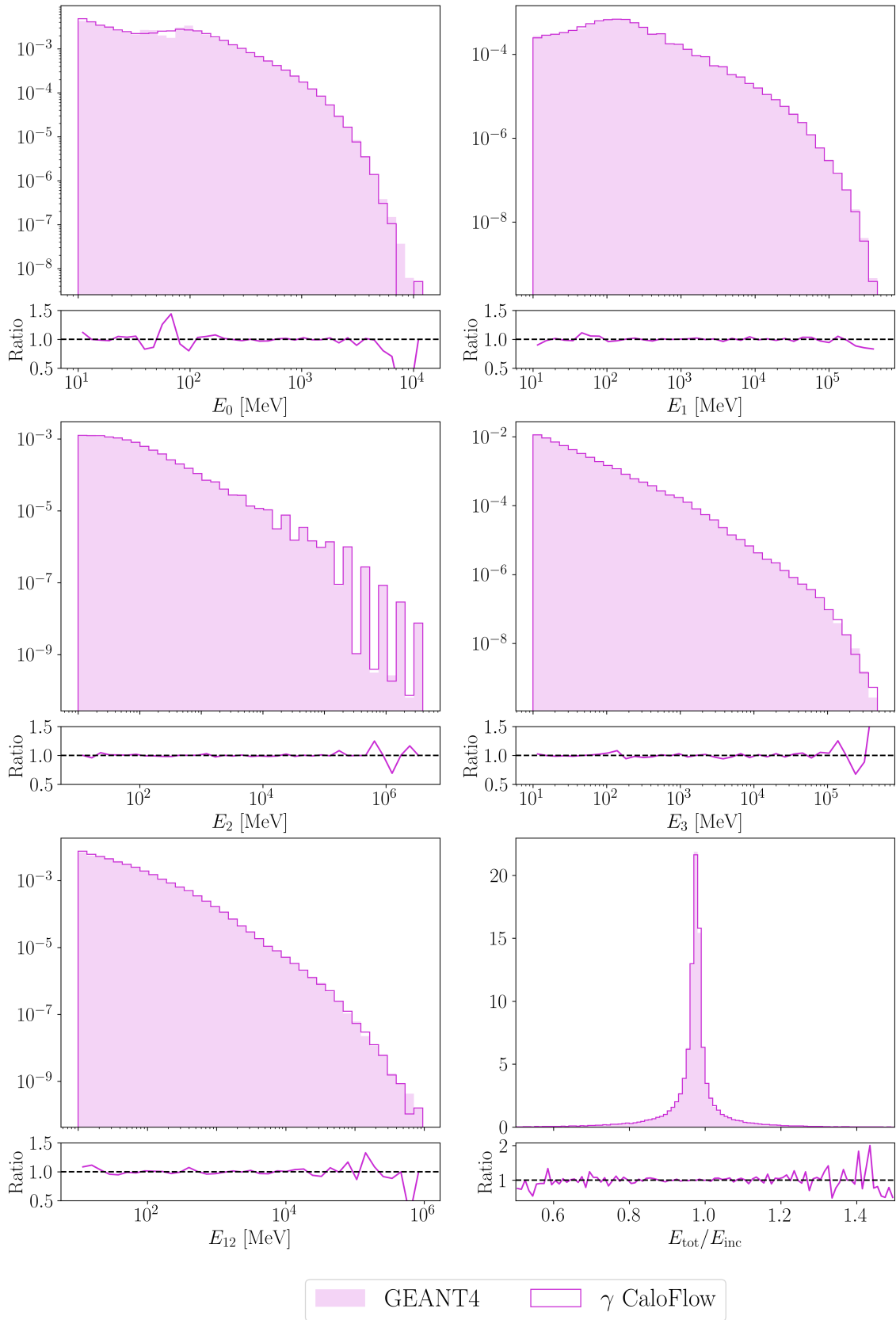



Figure 5: Energy distributions for γ dataset.

Table 4: Comparison of χ^2/NDF values between CALOFLOW and ATLFAST3 for histograms of E_{tot}/E_{inc} at discrete values of E_{inc} (see Figures 7 and 8).

	χ^2/NDF	
	CALOFLOW (this work)	ATLFAST3 [7]
γ	526/450 = **1.17**	5657/419 = 13.5
π^+	629/480 = **1.31**	5503/435 = 12.7

Like for the γ dataset, E_{tot}/E_{inc} is well-learned by flow-I. The slight discrepancy at high energies can be attributed to there being fewer events with high E_{inc} as shown in Figure 2.

Next, we compare the distributions of E_{tot}/E_{inc} from the CALOFLOW and GEANT4 samples for various discrete incident energies E_{inc}. The corresponding histograms are shown in Figures 7 and 8. An equivalent comparison can be found for the ATLFAST3 study in Figures 10 and 11 of [7]. As additional comparison, we included a digitized [50] version of the ATLFAST3 [7] result. Overall, we find that CALOFLOW is able to properly learn the mean and widths of the E_{tot}/E_{inc} distributions. CALOFLOW is even able to reproduce most of the distributions corresponding to higher E_{inc} which have lower statistics. By using lower noise level when training π^+ flow-I, we were able to properly describe the bimodal distribution found in the π^+ histograms corresponding to 256 MeV, 512 MeV, and 1024 MeV. This is a significant improvement from the results found in the ATLFAST3 study.

In Figures 9 and 10, we include plots that show a detailed comparison between the histograms generated based on CALOFLOW and GEANT4 shown in Figures 7 and 8. Following FASTCALOGAN [51], we calculated the mean and RMS from the histograms which have excluded outliers. In Figure 9, we find that there is less than 0.3% discrepancy in the mean value of each γ histogram in Figure 7. There is a larger discrepancy in the RMS values. Nevertheless, the maximum discrepancy in RMS is still only about 5% and it occurs at high $E_{inc} = 2.1$ TeV which has lower statistics. Most of the RMS discrepancies for lower E_{inc} are less than 2%. In Figure 10, we find that all the π^+ histogram means in Figure 8 have less than 2% discrepancy. We also see excellent agreement in the RMS with less than 3% discrepancy for most of the histograms. Comparing with the equivalent results from ATLFAST3 (see Figures 12(a) and 12(c) in [7]), we see that CALOFLOW managed to achieve a greater degree of agreement with GEANT4.

As an additional comparison with the ATLFAST3 results, we computed the χ^2 per degree of freedom (χ^2/NDF) for the histograms in Figures 7 and 8. Although we used a slightly different binning compared to ATLFAST3, dividing χ^2 by NDF should make the calculated values robust against different binning choices. The results can be found in Table 4. The ATLFAST3 result was taken from [7] and not from our digitization, as the line extraction is not very accurate. We find that the χ^2/NDF values of the CALOFLOW results in Figures 7 and 8 are $\mathcal{O}(10)$ better than the corresponding results found in ATLFAST3, indicating again that normalizing flows generate higher quality showers compared to GANs [9, 10].

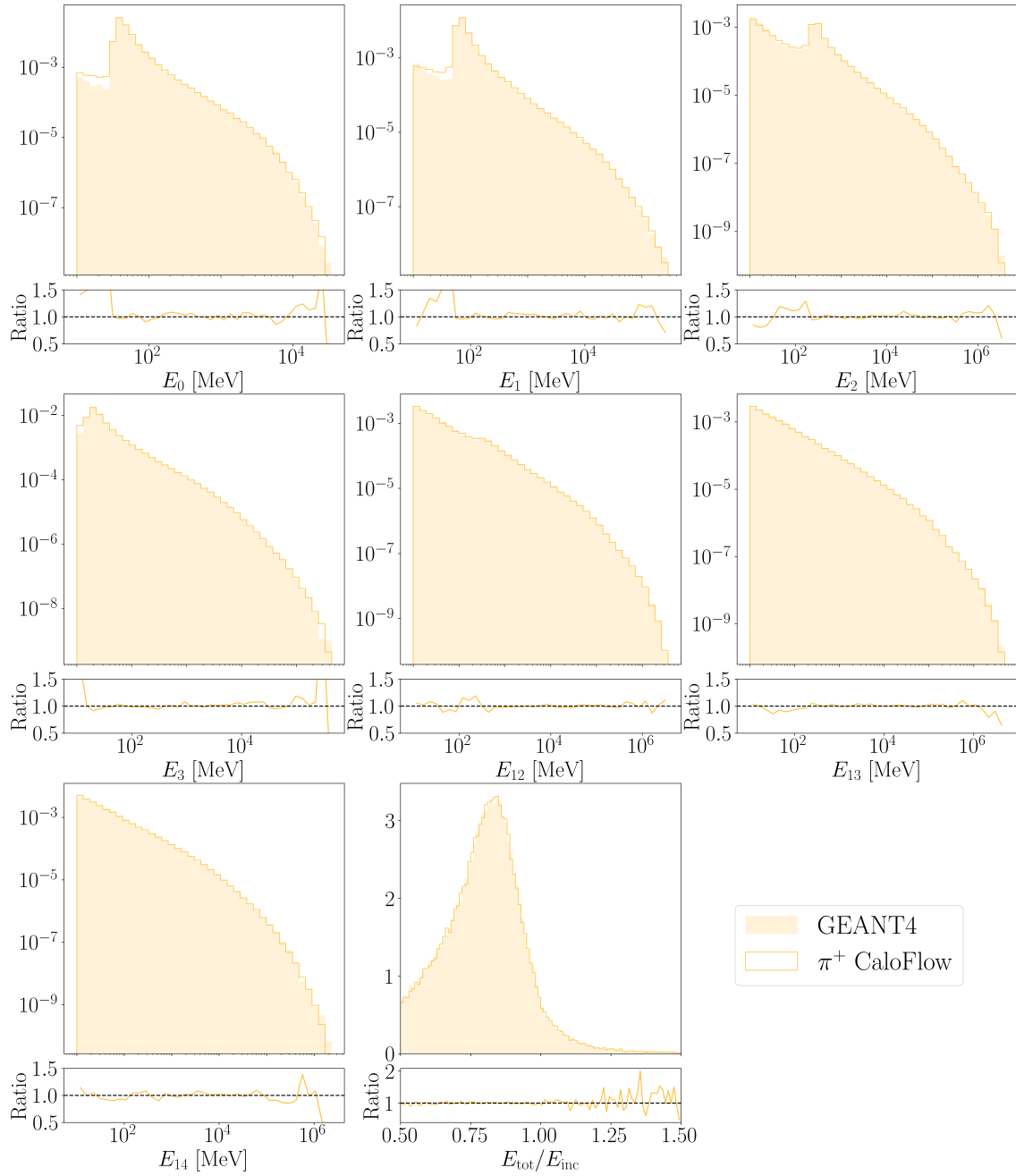



Figure 6: Energy distributions for π^+ dataset.

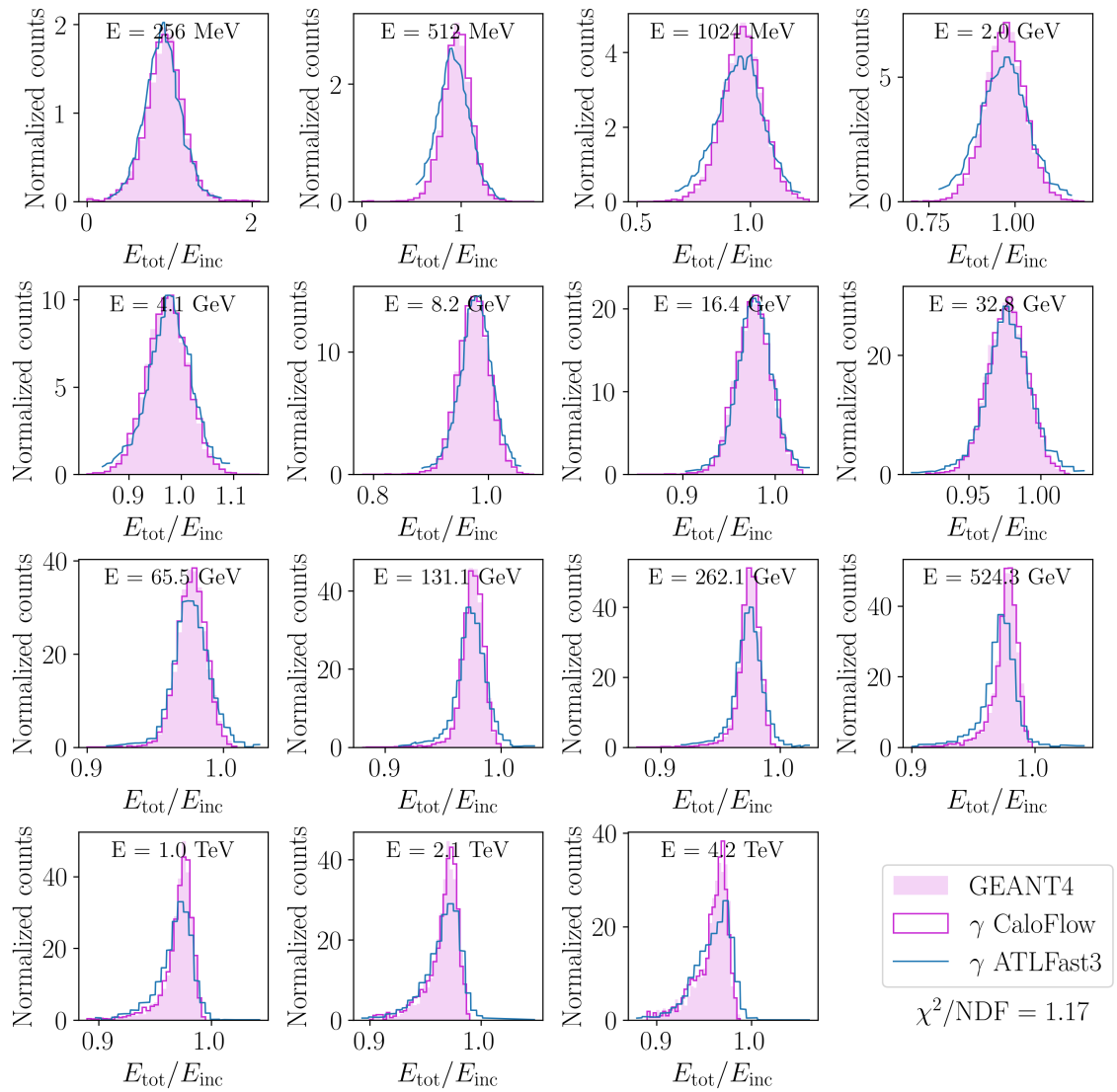



Figure 7: Histograms of $E_{\text{tot}}/E_{\text{inc}}$ for various discrete values of E_{inc} in the γ dataset. To guide the eye, we digitized [50] Figure 10 of ATLFAST3 [7].

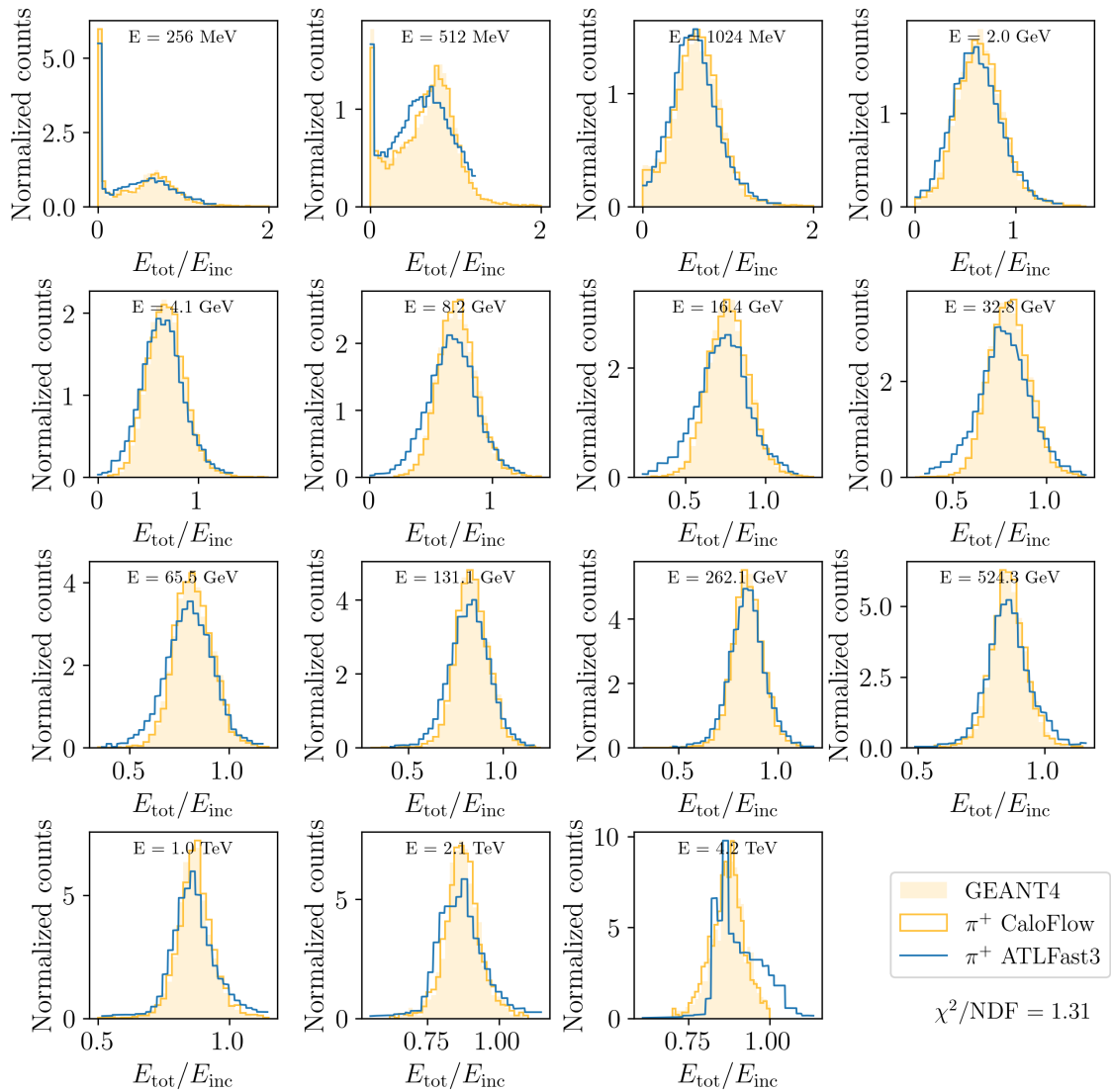



Figure 8: Histograms of $E_{\text{tot}}/E_{\text{inc}}$ for various discrete values of E_{inc} in the π^+ dataset. To guide the eye, we digitized [50] Figure 11 of ATLFAST3 [7].

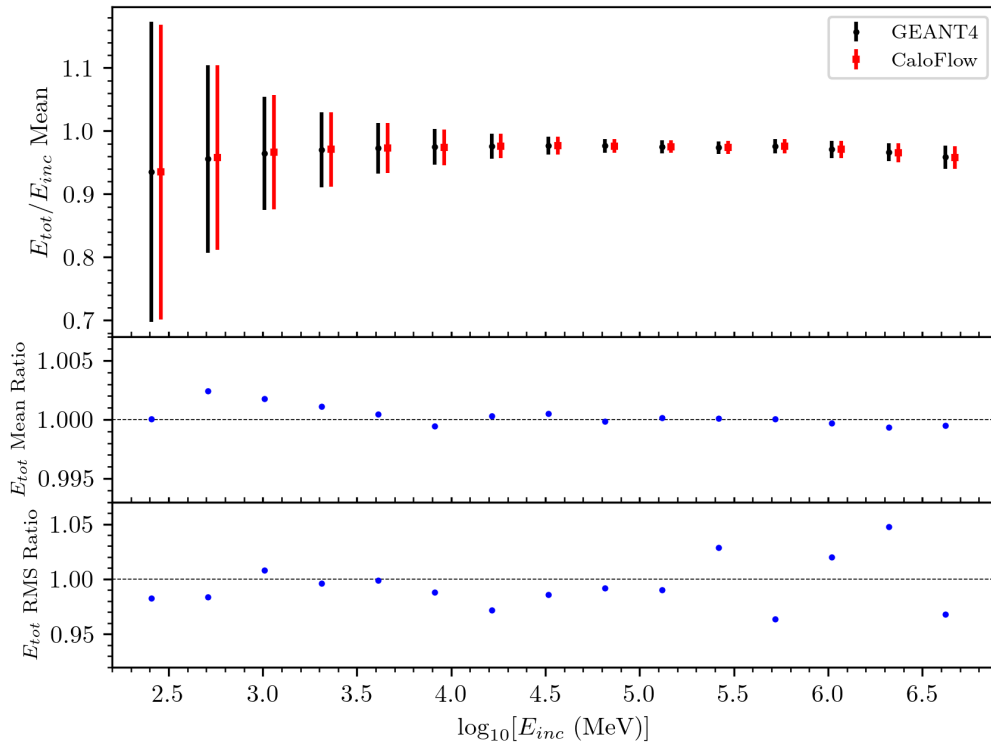



Figure 9: Comparison of mean and RMS of γ histograms between CALOFLOW and GEANT4 in Figure 7. Following FASTCALOGAN [51], we calculated the mean and RMS from the histograms which have excluded outliers. Top: Mean of E_{tot}/E_{inc} vs E_{inc}. Middle: Ratio of E_{tot} mean between CALOFLOW and GEANT4. Bottom: Ratio of E_{tot} RMS between CALOFLOW and GEANT4. The low statistical uncertainty in the data points of the lower two panels resulted in the error bars being smaller than the size of the markers. See Figure 12(a) in [7] for comparison with ATLFAST3 study.

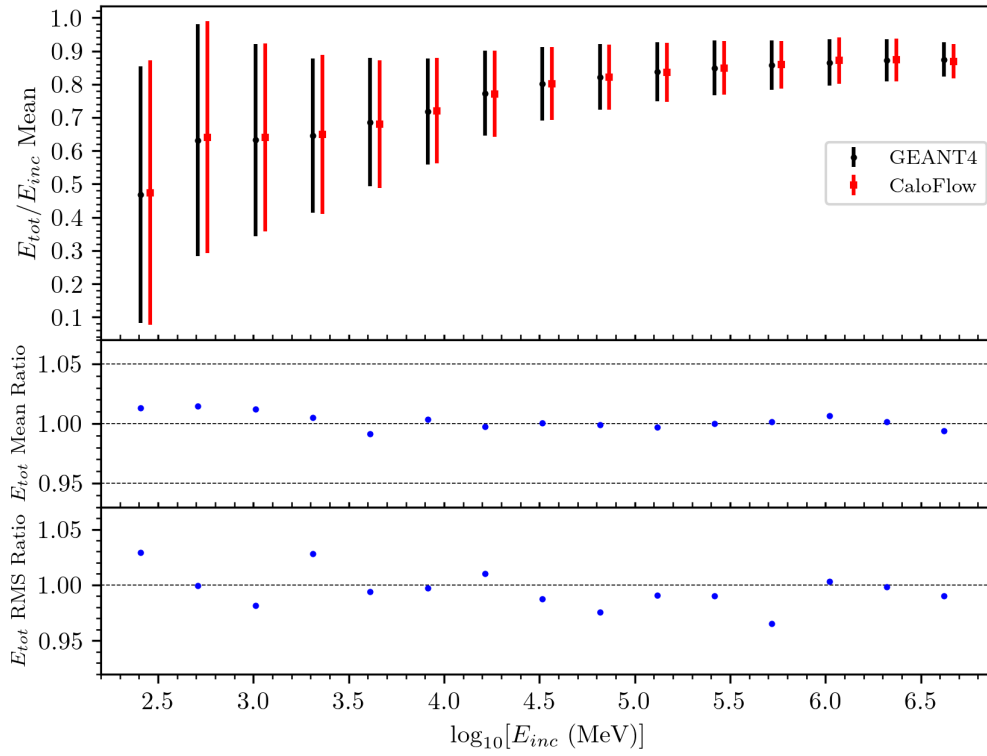



Figure 10: Same as Fig. 9 but for π^+ showers. See Figure 12(c) in [7] for comparison with ATLFAST3 study.

4.2.2 Shower shape histograms

Figures 11–16 show histograms corresponding to shower shape distributions produced by CALOFLOW compared to those from the GEANT4 reference sample. The center of energy in the η and ϕ directions are defined based on the location of the voxel centers. The center of energy in the $\eta(\phi)$ direction is defined via the locations of the voxel centers in mm, $H(F)$, as shown in eq. (4)

$$\langle \eta_k \rangle = \frac{\sum \left(\vec{\mathcal{I}}_k \odot H \right)}{E_k}, \quad \text{and} \quad \langle \phi_k \rangle = \frac{\sum \left(\vec{\mathcal{I}}_k \odot F \right)}{E_k}, \tag{4}$$

where $\vec{\mathcal{I}}_k$ contains the voxel energies in the kth layer, $\odot$ denotes the Hadamard product, and $\sum$ denotes sum over all elements.

The corresponding width of the center of energy is defined as shown in eq. (5).

$$\sigma_k^\eta = \sqrt{\frac{\sum \left(\vec{\mathcal{I}}_k \odot H^2 \right)}{E_k} - \left(\frac{\sum \left(\vec{\mathcal{I}}_k \odot H \right)}{E_k} \right)^2}, \quad \text{and} \quad \sigma_k^\phi = \sqrt{\frac{\sum \left(\vec{\mathcal{I}}_k \odot F^2 \right)}{E_k} - \left(\frac{\sum \left(\vec{\mathcal{I}}_k \odot F \right)}{E_k} \right)^2}, \tag{5}$$

We see in Figures 11–16 that the center of energy in both the η and ϕ directions modelled by CALOFLOW closely match the ones found in the GEANT4 reference sample. Furthermore, teacher and student histograms are mostly overlapped. This implies that the CALOFLOW student was very well trained on the CALOFLOW teacher. Due to the close similarity between the teacher and student histogram results, it will not be necessary for us to distinguish between the teacher and student results in the rest of Section 4.2. This result is remarkable considering the fact that there are spikes in the some of the center of energy histograms (e.g. see Fig 14) that might make them difficult to learn. As for the widths of the center of energy, we find

two peaks in the distribution for both γ and π^+. For γ, the global maximum is found away from zero, while the global maximum is found at zero for π^+. There is some disagreement at larger widths found in Figures 13-16. Still, we find excellent overall agreement in the bimodal distribution of the center of energy widths. Even at larger widths, CALOFLOW is able to model the general shape of the distribution.

Finally, looking at the distribution of voxel energies for all layers in Fig. 17, we see that they are closely modelled by CALOFLOW over 6 orders of magnitude for both, γ and π^+ showers. There is some discrepancy in the low voxel energy region between $[10^{-2}, 10^{-1}]$ MeV especially for the pion showers. There is a peak around 10^{-1} MeV corresponding to minimum ionizing particles (MIPs) in the pion voxel energy distribution. However, CALOFLOW is unable to properly model this feature.

Fig. 18 includes 3 voxel energy distributions with each corresponding to a different E_{inc}. Showers with $E_{\text{inc}} = 512$ MeV are taken to be representative of low energy events, while showers with $E_{\text{inc}} = 32768$ MeV are representative of mid energy events, and showers with $E_{\text{inc}} = 2097152$ MeV are representative of high energy events. We found that the MIP peak is present for low and mid E_{inc} events as shown in Fig. 18. The inability to model the MIP peak might be the reason for the poor fit at low voxel energies for the low and mid E_{inc} plots. In Section 4.3, we show that a trained binary classifier can be sensitive to the poor fit at low voxel energies.

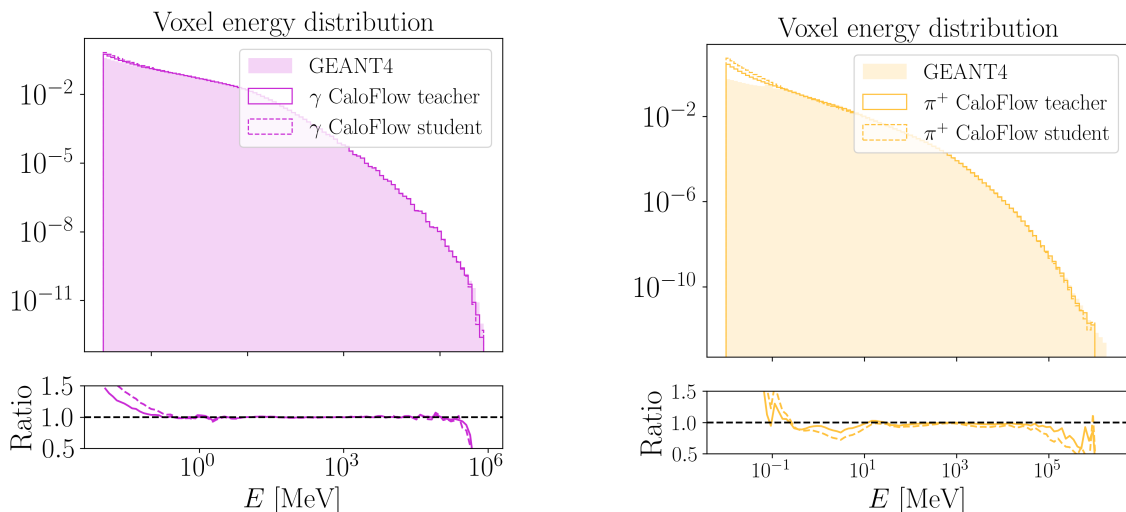



Figure 17: Distribution of voxel energies across all layers for γ and π^+ datasets.

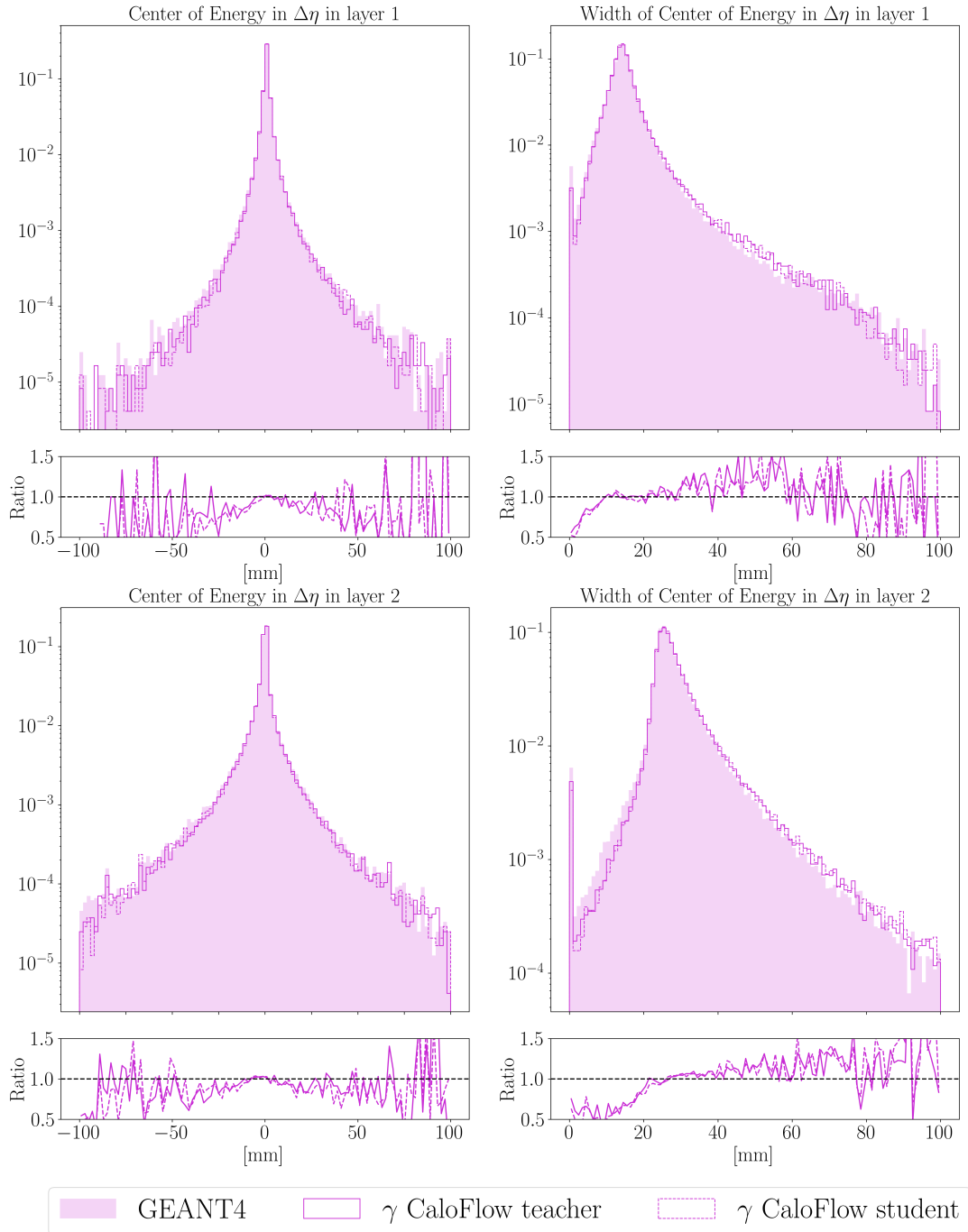



Figure 11: Distributions of the center of energy in the η and direction for the γ dataset. Note that layers 0, 3 and 12 only have a single α bin each. Hence, there is no positional information to extract for these layers.

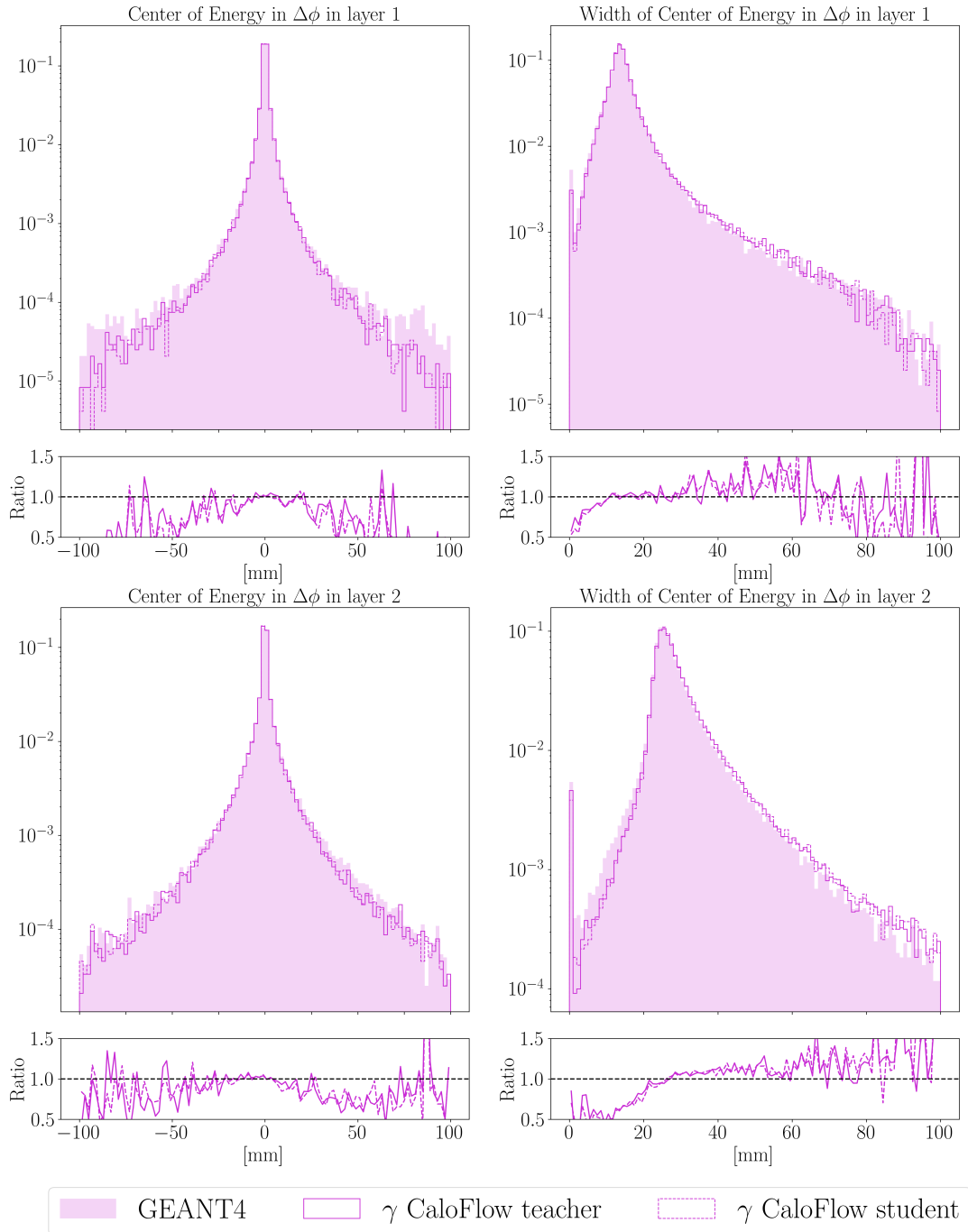



Figure 12: Distributions of the center of energy in the ϕ and direction for the γ dataset. Note that layers 0, 3 and 12 only have a single α bin each. Hence, there is no positional information to extract for these layers.

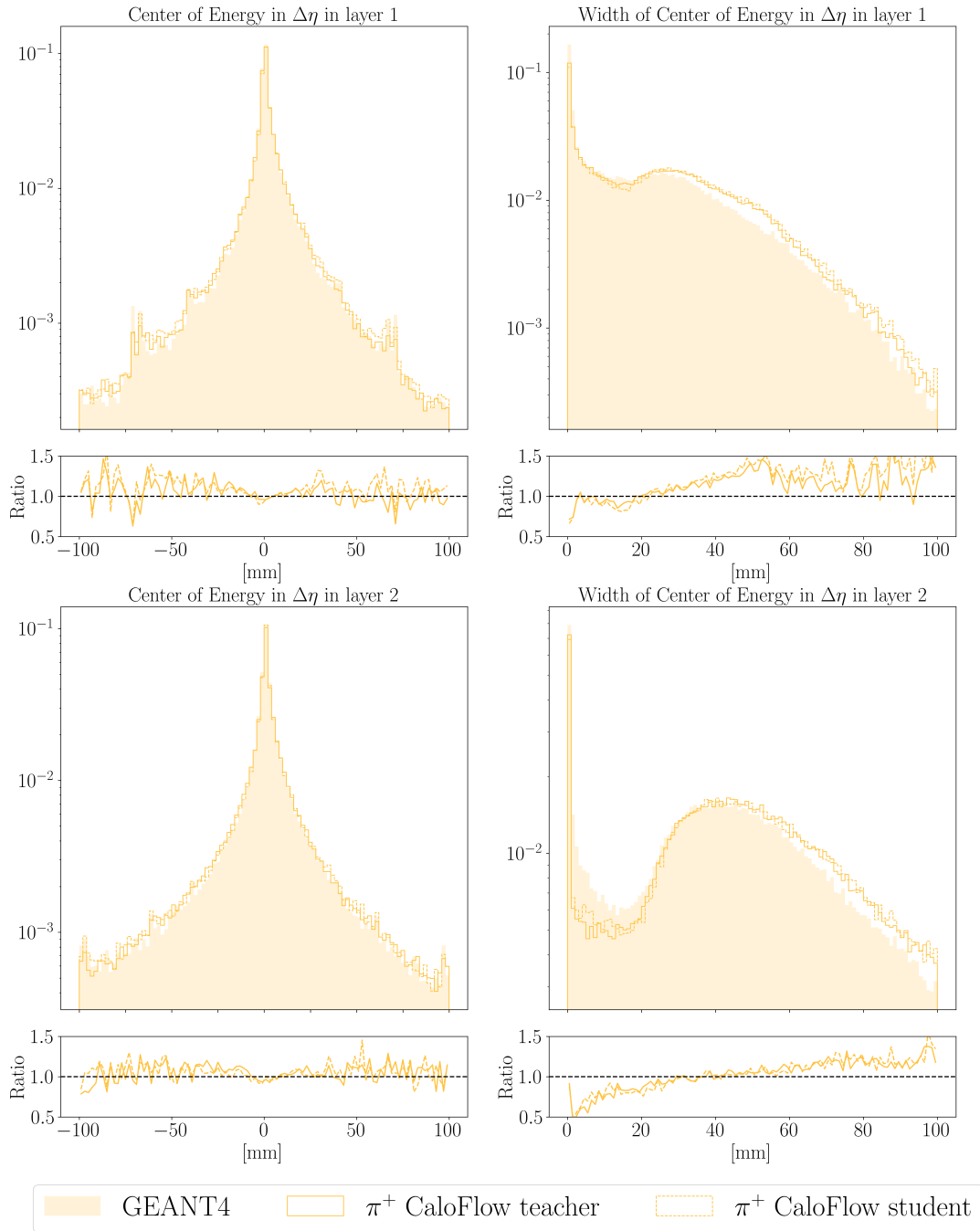



Figure 13: Distributions of the center of energy for layers 1 and 2 in the η direction for the π^+ dataset. Note that layers 0, 3 and 14 only have a single α bin each. Hence, there is no positional information to extract for these layers.

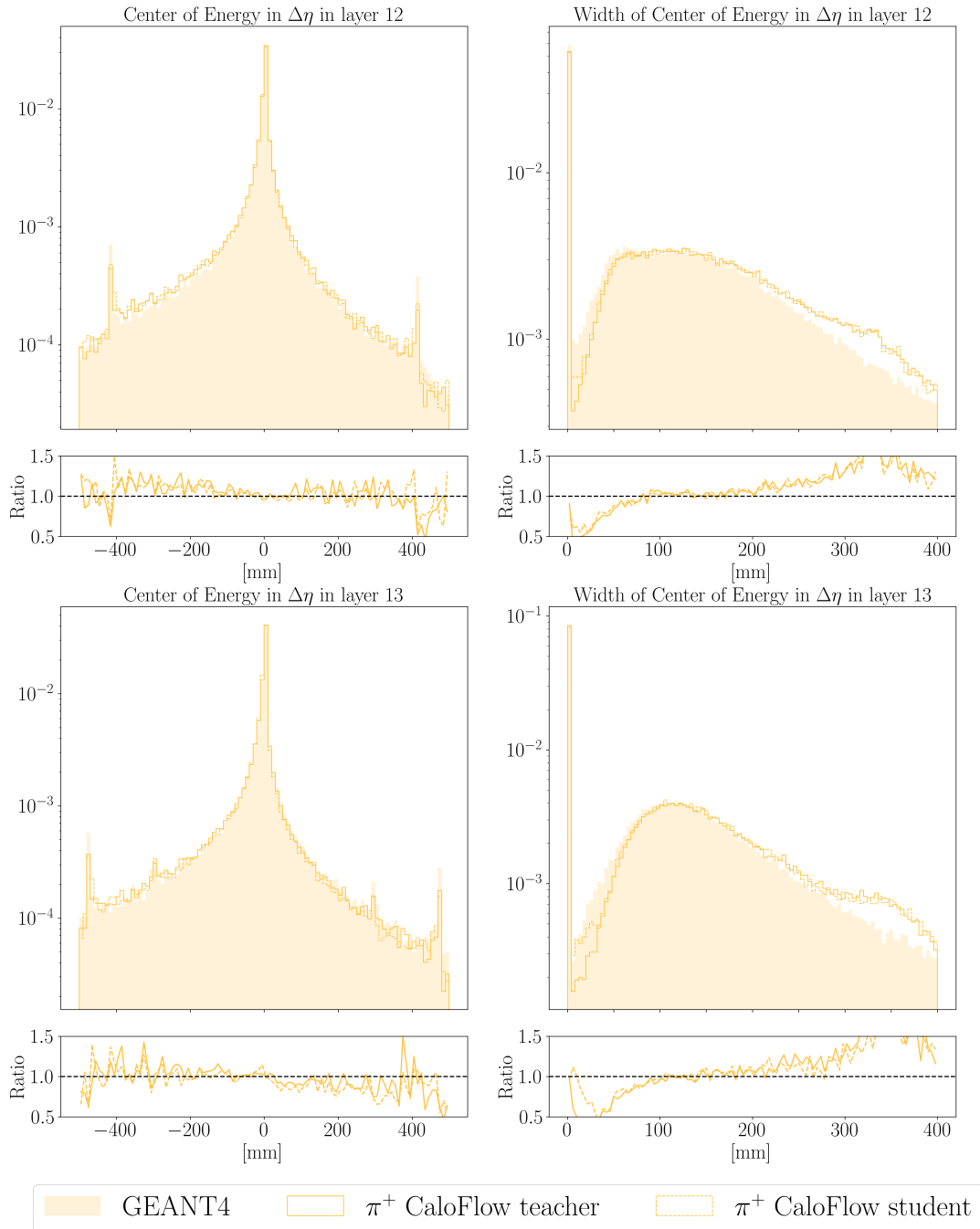



Figure 14: Distributions of the center of energy for layers 12 and 13 in the η direction for the π^+ dataset. Note that layers 0, 3 and 14 only have a single α bin each. Hence, there is no positional information to extract for these layers.

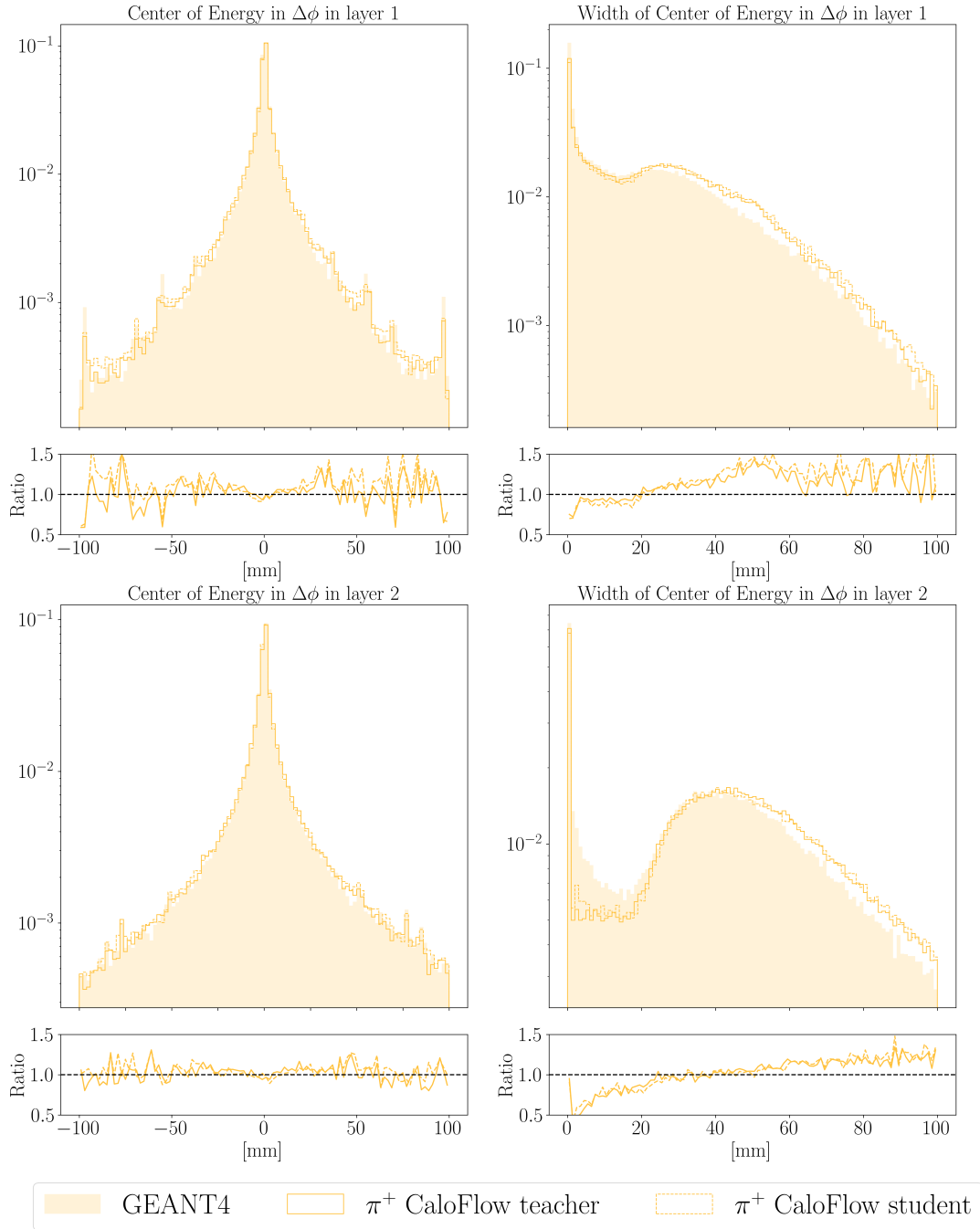



Figure 15: Distributions of the center of energy for layers 1 and 2 in the ϕ direction for the π^+ dataset. Note that layers 0, 3 and 14 only have a single α bin each. Hence, there is no positional information to extract for these layers.

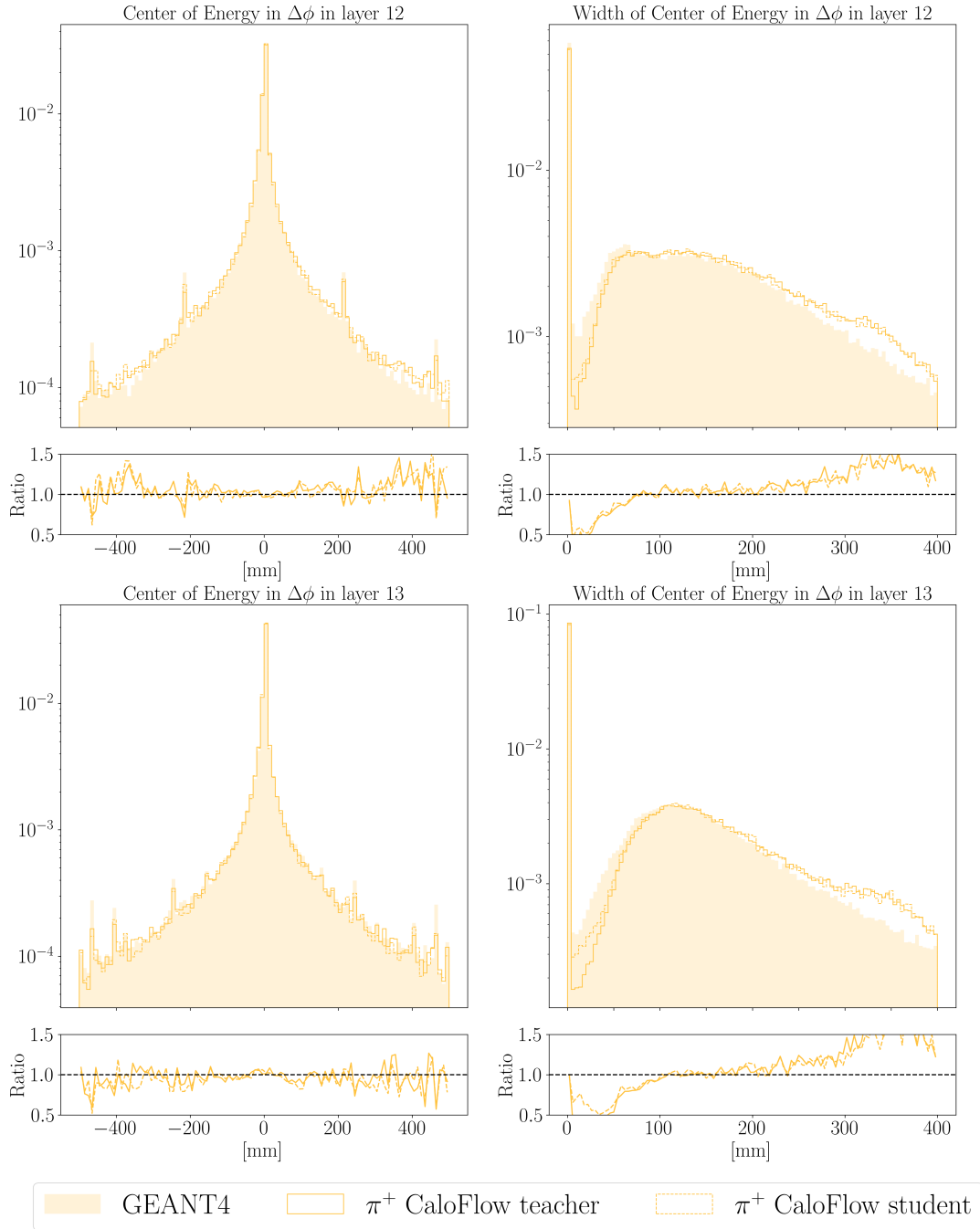



Figure 16: Distributions of the center of energy for layers 12 and 13 in the ϕ direction for the π^+ dataset. Note that layers 0, 3 and 14 only have a single α bin each. Hence, there is no positional information to extract for these layers.

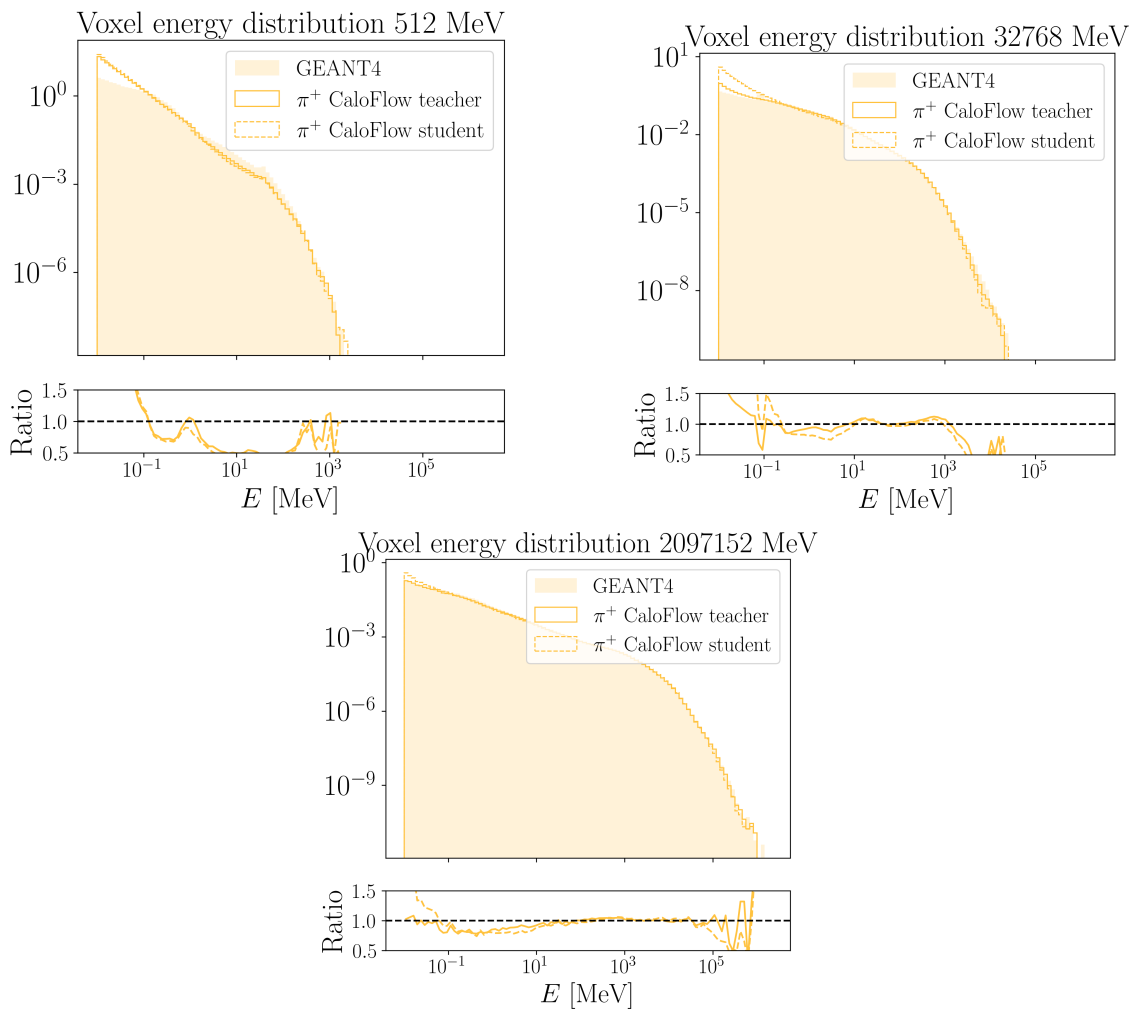



Figure 18: Distribution of voxel energies across all layers at $E_{\text{inc}} = 512$ MeV (low energy), $E_{\text{inc}} = 32768$ MeV (mid energy), and $E_{\text{inc}} = 2097152$ MeV (high energy) for π^+ dataset.

4.3 Classifier scores

After looking at the similarities in 1D histograms between the CALOFLOW and GEANT4 samples in Section 4.2, it is important to perform a full phase space analysis (including correlations) by comparing the probability density that CALOFLOW learned to the one induced by GEANT4. As in [9,10], we train a binary, neural classifier based on a fully-connected, deep neural network (DNN) to distinguish between generated CALOFLOW samples and GEANT4 samples. This serves as an approximation to the Neyman-Pearson classifier, which is the ultimate test of whether $p_{\text{generated}}(x) = p_{\text{data}}(x)$. According to the Neyman-Pearson lemma, we expect the AUC to be 0.5 if the true and generated probability densities are equal. The AUC is 1 if the classifier is able to perfectly distinguish between generated and true samples. The second metric, JSD $\in [0, 1]$, is the Jensen-Shannon divergence which also measures the similarity between the two probability distributions. The JSD is 0 if the two distributions are identical and 1 if they are disjoint.

We detail the classifier architecture and training procedure[4] in appendix B. The resulting

---

[4]We also trained a classifier with the architecture and preprocessing of [12] and found quantitatively similar results to tab. 5.

Table 5: Classifier results, based on 10 independent runs. The first (second) numbers in every entry are AUCs (JSDs). See text for more detailed explanations.

AUC / JSD		DNN based classifier	
		GEANT4 vs. CALOFLOW (teacher)	GEANT4 vs. CALOFLOW (student)
γ	low-level (regular)	0.701(3) / 0.092(3)	0.739(3) / 0.131(4)
	low-level (logit)	0.678(4)/ 0.075(4)	0.706(3)/ 0.100(3)
	high-level	0.551(3) / 0.013(2)	0.556(3) / 0.015(2)
π^+	low-level (regular)	0.827(3)/ 0.260(5)	0.866(2)/ 0.341(3)
	low-level (logit)	0.911(3)/ 0.457(7)	0.914(2)/0.464(6)
	high-level	0.692(2)/ 0.098(2)	0.706(4)/ 0.108(4)

scores are summarized in Table 5. Low-level input refers to incident energies E_{inc} and the calorimeter samples in the format provided in *CaloChallenge* datasets. Training the classifier on low-level input allows us to measure the difference between the generated and reference datasets based on the voxel energies and their physical location.

Meanwhile, the "high-level" scores in Table 5 correspond to a DNN classifier trained on a set of physically-relevant high-level features, here chosen to be: incident energy E_{inc}, layer energies E_i, the center of energies and their corresponding widths in the η and ϕ directions. The definitions of the center of energies and corresponding widths were stated in eqs. (4) and (5).

From the scores in Table 5, we see that CALOFLOW is able to produce generated samples of sufficiently high fidelity to fool the DNN-based classifier. This is evident from most of the classifier AUC and JSD scores in Table 5 being significantly lower than unity. Overall, the teacher models performed better than the student models. Nevertheless, the student models were still able to achieve impressive classifier scores which quantitatively demonstrates the effectiveness of normalizing flows in generative modeling tasks. We note that the high-level classifier scores are even better than the low-level classifier scores for a given model. This is good considering that these high-level features are probably more directly relevant for subsequent analysis steps (such as reconstruction).

Recently, the authors of CALOSCORE [12] have produced results for the γ portion of Dataset 1 (as well as Datasets 2 and 3), using a score-based diffusion model. For the classifier test between CALOSCORE-generated and GEANT4 showers (i.e. low-level features), they found an AUC of 0.98. Even accounting for the possibility of different classifier architectures and training hyperparameters, this likely indicates considerably more separability than our CALOFLOW-generated showers (whose AUC is 0.739 for the low-level classifier trained on γ showers from the student flow).

The low-level classifiers, with regular preprocessing as in [35], obtained from the 10 independent runs (see Table 5) were evaluated on π^+ showers with specific E_{inc}. The corresponding plot of AUC vs E_{inc} is shown in Fig. 19 (left plot). The plot seems to imply that CALOFLOW is fitting the voxel energy distribution better at low E_{inc}. However, this does not agree with our observations from the plots in Fig. 18. We found that preprocessing voxel energy inputs in the following way allows the classifier to become sensitive to deviations at low voxel energies: First we normalize the voxel energies by layer energies E_i (such that normalized voxel energies in each layer sum to one). Then, we transform the normalized voxel energies to logit space

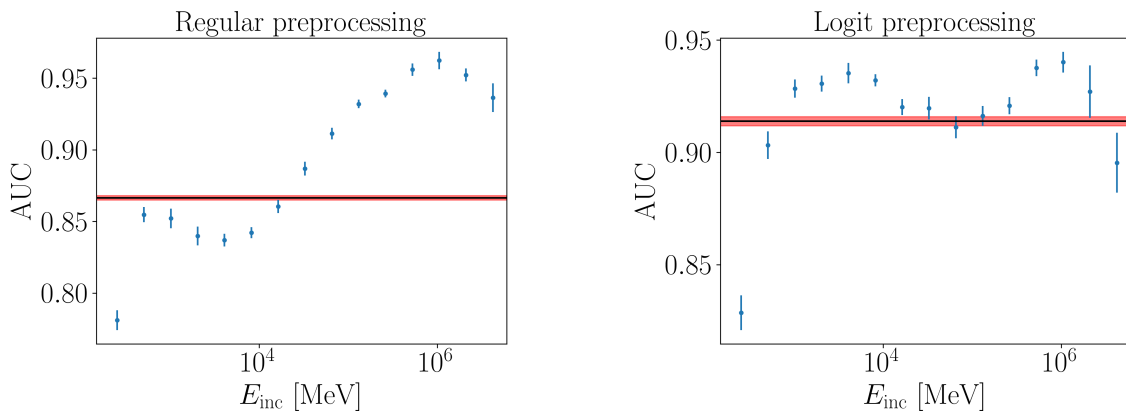



Figure 19: Plots of AUC vs E_{inc}. The AUCs are obtained using a classifier trained (10 independent runs) on low-level input from all showers from the CALOFLOW π^+ student and GEANT4 samples, and then evaluated on showers of specific E_{inc}. The left plot is based on voxel energy inputs with the regular proprecessing decribed in appendix B. The right plot is based on voxel energy inputs with logit preprocessing. The reference AUC based on showers of all E_{inc} is shown by the horizontal line.

as done during training. The overall classifier scores for this modified logit preprocessing for both particle types are also shown in Table 5.

The corresponding plot of AUC vs E_{inc} for this modified preprocessing is shown in Fig. 19 (right plot). Here we see that the low E_{inc} showers generally have higher AUC scores compared to the scores at the same E_{inc} in the left plot. This is more consistent with our observations from Fig. 18.

4.4 Generation timing

We found that the average time taken to generate a single shower events by CALOFLOW teacher for a generation batch size of 10000 on our TITAN V GPU is 18.91 ms for γ and 43.91 ms for π^+. For the same batch size, we were able to significantly reduce the generation time for CALOFLOW student.

In Table 6, we include the average time taken to generate a single shower events by CALOFLOW student for different generation batch sizes. The timings were separately evaluated on our Intel i9-7900X CPU at 3.30GHz and our TITAN V GPU. We observe that increasing the batch size reduces the generation time per shower event. However, we could not generate with large (1000 and 10000) batch sizes on the CPU due to memory constraints. We observed that the generation time is largely dependent on the sizes of layers in the MADE block (see Table 3 for MADE block layer sizes). This accounts for the difference (similarity) in generation timing between γ teacher (student) and π^+ teacher (student). With CALOFLOW student, we are able to achieve impressively low GPU generation times of 0.07 ms per event for γ and 0.09 ms per event for π^+.

Figure 20 compares the average CPU generation time per event found using CALOFLOW, FASTCALOGAN[5] and GEANT4. The generation times for 65 GeV and 2 TeV are shown for FASTCALOGAN and GEANT4 based on timings provided in Section 6.4 of [8]. For CALOFLOW, we included generation times for these two energies and also two other intermediate energies (262 GeV and 1 TeV). As we do not know the batch size used when timing FASTCALOGAN,

---

[5]Note that the timing of FASTCALOGAN also includes the time it takes to map the voxel energies back to calorimeter cells and we do not know how the total time splits between voxel energy generation by the GAN and cell assignment of the subsequent step.

Table 6: Average time taken to generate a single shower event by CALOFLOW student for the two particle types. The timing was computed for different generation batch sizes on our Intel i9-7900X CPU at 3.30GHz and our TITAN V GPU. We were not able to generate the shower events on the CPU for batch sizes of 1000 and 10000 due to memory constraints.

Particle type	Batch size	Student generation time per event	
		GPU	CPU
γ	1	57.23 ms	115.88 ms
	10	5.81 ms	12.68 ms
	100	0.62 ms	2.50 ms
	1000	0.11 ms	-
	10000	**0.07 ms**	-
π^+	1	74.09 ms	126.05 ms
	10	7.46 ms	14.03 ms
	100	0.80 ms	2.91 ms
	1000	0.15 ms	-
	10000	**0.09 ms**	-

we timed CALOFLOW separately using batch sizes of 1 (abbreviated as CF(1)) and 100 (abbreviated as CF(100)). We see from Figure 20 that CALOFLOW generation times are constant regardless of E_{inc}. In contrast, GEANT4 generation time increases with E_{inc}. At $E_{inc} = 65$ GeV, GEANT4 has a generation time that is $\mathcal{O}(10^2)$ slower compared to CF(1) and $\mathcal{O}(10^3)$ slower compared to CF(100). At $E_{inc} = 2$ TeV, GEANT4 has a generation time that is $\mathcal{O}(10^3)$ slower compared to CF(1) and $\mathcal{O}(10^4)$ slower compared to CF(100). The timings for FASTCALOGAN are on the same order as those found using CF(1).

We can also compare directly against the results of CALOSCORE for the γ portion of Dataset 1. Table II of [12] reports a time of 4s to generate 100 γ showers using the score-based diffusion model approach, on a GPU with batch size of 100. Meanwhile the CALOFLOW student accomplishes the same task on similar hardware in 0.062s, which is nearly 2 orders of magnitude faster.

5 Conclusions/outlook

We have shown that the CALOFLOW algorithm can be applied successfully to realistic detector simulations, namely Dataset 1 of the *Fast Calorimeter Simulation Challenge 2022*. Our results significantly outperform FASTCALOGAN, a deep generative model that was trained on the same dataset and is used in a (very limited) portion of ATLFAST3, the current fast-simulation framework of the ATLAS experiment. Based on the results shown here, we are confident that state-of-the-art deep generative models (and particularly those based on normalizing flows) will be able to play a much larger role in future versions of the fast-simulation framework.

From a machine-learning perspective, we have seen some drawbacks in the choice to use discrete incident energies, in particular concerning the need to interpolate between energies.

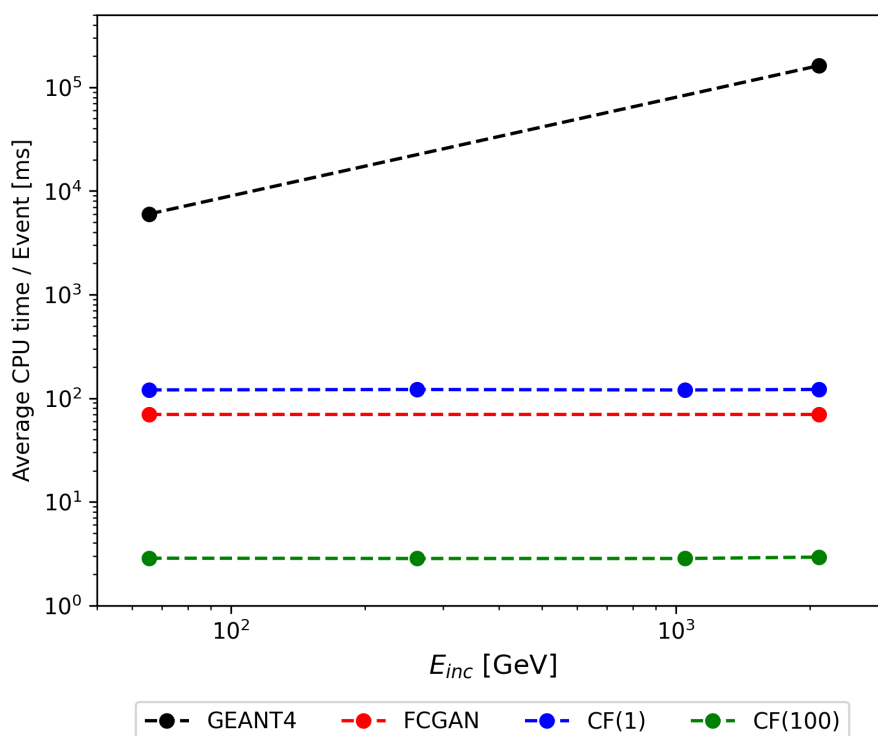



Figure 20: Comparison of CPU generation time per event for CALOFLOW (abbreviated as CF in legend), FASTCALOGAN (abbreviated as FCGAN in legend) and GEANT4 at several different E_{inc}. Numbers in parentheses refer to the generation batch sizes used in CALOFLOW. The CALOFLOW timing was based on our Intel i9-7900X CPU at 3.30GHz. The other timings are based on CPU used by FASTCALOGAN.

We expect CALOFLOW to interpolate well, but with the datasets provided by the ATLAS collaboration, there is no way of checking this in detail. We think training a conditional generative model using continuous, uniformly (or log-uniformly) sampled energies, instead of discrete energies, could be advantageous and should be considered in the future.

One of the main future directions stemming from this work is figuring out how to extend the CALOFLOW framework to larger numbers of voxels, e.g. those of Datasets 2 [52] and 3 [53] of the *CaloChallenge*. Indeed, the current CALOFLOW algorithm scales badly with the number of simulated voxels and cannot be applied as-is to Datasets 2 and 3. However, we expect straightforward modifications of the basic CALOFLOW approach to be strong contenders for fast-and-accurate simulation of these higher-dimensional datasets. In the future, it will be interesting to compare flow-based approaches to Datasets 2 and 3 to other approaches, such as the score-based diffusion models of [12] (which are currently the only generative models successfully trained on Datasets 2 and 3).

It would also be interesting to consider generalizing the CALOFLOW approach beyond the incident particles considered here, and beyond the narrow η slice. For example, it should be very straightforward to train a single conditional flow or pair of flows, conditioned on η as well as the incident energy. This could potentially simplify the ATLFAST3 framework, where 300 GANs were trained on individual narrow η slices.

Table 7: Base and max LRs used when training γ and π^+ teacher/student models. For γ teacher, the max LR here refers to the max LR in the first cycle.

		base LR	max LR
γ	Teacher	5×10^{-5}	2×10^{-3}
	Student	4×10^{-5}	1×10^{-3}
π^+	Teacher	4×10^{-5}	1×10^{-3}
	Student	2×10^{-5}	1×10^{-3}

Acknowledgments

We are grateful to Michele Faucci Gianelli and Ben Nachman for helpful discussions and comments on the draft.

In this work, we used the NumPy 1.16.4 [54], Matplotlib 3.1.0 [55], sklearn 0.21.2 [56], h5py 2.9.0 [57], pytorch 1.11.0 [58], and nflows 0.14 [59] software packages.

Funding information This work was supported by DOE grant DOE-SC0010008. CK would like to thank the Baden-Württemberg-Stiftung for support through the program *Internationale Spitzenforschung*, project *Uncertainties — Teaching AI its Limits* (BWST_IF2020-010). IP would like to thank the Rutgers Dept. of Physics and Astronomy for support through the Boyd scholarship.

A   Cyclic LR

With a cyclic LR schedule (see fig. 21a), the LR begins at a chosen base LR and then increases up to a maximum LR. The number of batches taken to increase from the base LR to the maximum LR is defined as the step size. After which, the LR decreases from the maximum LR to the base LR. This increase and subsequent decrease in LR is defined as a cycle, and this repeats for a chosen number of cycles. In certain cases, the maximum LR can be made to decrease with increasing number of batches.

OneCycle LR [49] (see fig. 21b) is a special case of cyclic LR which only relies on a single cycle followed by an annihilation phase where the base LR is gradually decreased up to a factor of 10,000.

For most of the models using OneCycle LR, the length of the annihilation phase was chosen to be 20% of the total number of training epochs shown in Table 2. The only exception was for π^+ teacher where the annihilation phase was chosen to be 10% of the total number of training epochs. The step size was taken to be half of the remaining number of epochs.

For γ teacher, which used regular cyclic LR, the length of one cycle was chosen to be 10 epochs. After each cycle, the max LR is decreased such that the difference between the max LR and the base LR is half that of the previous cycle. The step size in our study was fixed to be 5 epochs (i.e. half of a cycle).

The base and max LRs used when training the various models are shown in Table 7.

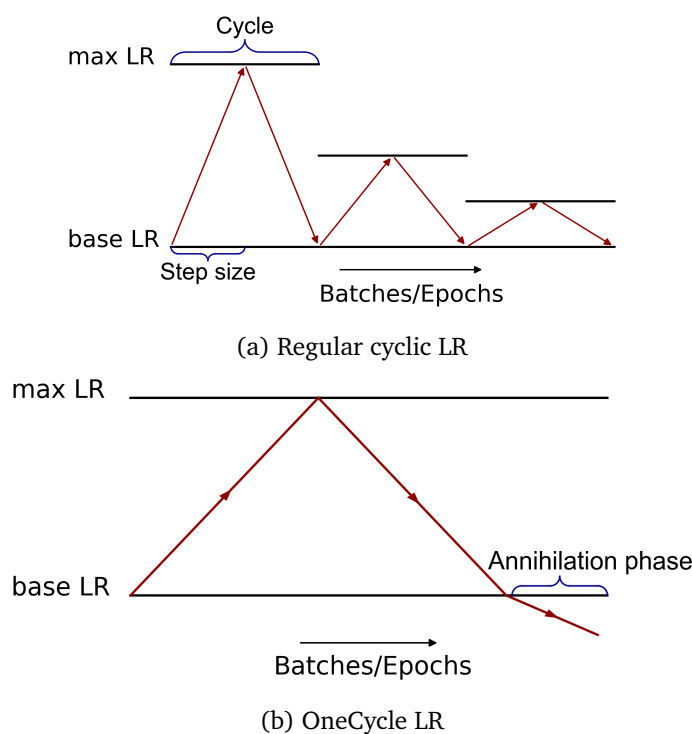
(a) Regular cyclic LR

(b) OneCycle LR

Figure 21: (a) Illustration of regular cyclic LR schedule with decreasing max LR. The length of one cycle was chosen to be 10 epochs. The step size in our study was fixed to be half the number of batches in one cycle. After each cycle, the max LR is decreased such that the difference between the max LR and the base LR is half that of the previous cycle. (b) Illustration of OneCycle LR schedule with annihilation phase.

B Classifier architecture

The classifier-based performance metric uses the classifier architecture that was provided in the evaluation script of the *CaloChallenge* [35]. It is based on the classifiers that were used in [9, 10]. In detail, the classifier is a deep, fully-connected neural network with an input and two hidden layers with 512 nodes each. The output layer returns a single number which is passed through a sigmoid activation function. All other activation functions are leaky ReLUs, with default negative slope of 0.01. We do not use any dropout or batch normalization regulators.

In the classifier of the low-level features, we use as input the incident energy (preprocessed as $\log_{10} E_{\text{inc}}$) and the energy deposition in each voxel (preprocessed as $\mathcal{I}/E_{\text{inc}}$). High-level features are the incident energy (preprocessed as $\log_{10} E_{\text{inc}}$), the energy deposited in each layer (preprocessed as $\log_{10}(E_i + 10^{-8})$), the center of energy in η (normalized with a factor 100), the center of energy in ϕ (normalized with a factor 100), the width of the η distribution (normalized with a factor 100), and the width of the ϕ distribution (normalized with a factor 100).

We split all the data in train/test/validation sets of the ratio (60:20:20). For the both the γ and π^+ showers, we used the second file that was provided at [36]. From our flow models, we used a generated sample of the same size and E_{inc} distribution as the GEANT4 data.

The networks are then optimized by training 50 epochs with an ADAM [60] optimizer with initial learning rate of $2 \cdot 10^{-4}$ and a batch size of 1000, minimizing the binary cross entropy. We use the model state with the highest accuracy on the validation set for the final evaluation and we subsequently calibrate the classifier using isotonic regression [61] of `sklearn` [56] based on the validation dataset before evaluating the test set.

References

[1] S. Agostinelli et al., *Geant4 — A simulation toolkit*, Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip. **506**, 250 (2003), doi:10.1016/S0168-9002(03)01368-8.

[2] J. Allison et al., *Geant4 developments and applications*, IEEE Trans. Nucl. Sci. **53**, 270 (2006), doi:10.1109/TNS.2006.869826.

[3] J. Allison et al., *Recent developments in Geant4*, Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip. **835**, 186 (2016), doi:10.1016/j.nima.2016.06.125.

[4] P. Calafiura, J. Catmore, D. Costanzo and A. Di Girolamo, *ATLAS HL-LHC computing conceptual design report*, Tech. Rep. CERN-LHCC-2020-015, CERN, Geneva, Switzerland (2020).

[5] M. Paganini, L. de Oliveira and B. Nachman, *CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks*, Phys. Rev. D **97**, 014021 (2018), doi:10.1103/physrevd.97.014021.

[6] M. Paganini, L. de Oliveira and B. Nachman, *Accelerating science with generative adversarial networks: An application to 3D particle showers in multilayer calorimeters*, Phys. Rev. Lett. **120**, 042003 (2018), doi:10.1103/PhysRevLett.120.042003.

[7] G. Aad et al., *AtlFast3: The next generation of fast simulation in ATLAS*, Comput. Softw. Big Sci. **6**, 7 (2022), doi:10.1007/s41781-021-00079-7.

[8] ATLAS collaboration, *Fast simulation of the ATLAS calorimeter system with generative adversarial networks*, Tech. Rep. ATL-SOFT-PUB-2020-006, CERN, Geneva, Switzerland (2020), https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-SOFT-PUB-2020-006.

[9] C. Krause and D. Shih, *Fast and accurate simulations of calorimeter showers with normalizing flows*, Phys. Rev. D **107**, 113003 (2023), doi:10.1103/PhysRevD.107.113003.

[10] C. Krause and D. Shih, *Accelerating accurate simulations of calorimeter showers with normalizing flows and probability density distillation*, Phys. Rev. D **107**, 113004 (2023), doi:10.1103/PhysRevD.107.113004.

[11] E. Buhmann et al., *Fast and accurate electromagnetic and hadronic showers from generative models*, Europhys. J. Web Conf. **251**, 03049 (2021), doi:10.1051/epjconf/202125103049.

[12] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Phys. Rev. D **106**, 092009 (2022), doi:10.1103/PhysRevD.106.092009.

[13] L. de Oliveira, M. Paganini and B. Nachman, *Controlling physical attributes in GAN-accelerated simulation of electromagnetic calorimeters*, J. Phys.: Conf. Ser. **1085**, 042017 (2018), doi:10.1088/1742-6596/1085/4/042017.

[14] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, *Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks*, Comput. Softw. Big Sci. **2**, 4 (2018), doi:10.1007/s41781-018-0008-x.

[15] M. Erdmann, J. Glombitza and T. Quast, *Precise simulation of electromagnetic calorimeter showers using a Wasserstein generative adversarial network*, Comput. Softw. Big Sci. **3**, 4 (2019), doi:10.1007/s41781-018-0019-7.

[16] ATLAS collaboration, *Deep generative models for fast photon shower simulation in ATLAS*, Comput. Softw. Big Sci. **8**, 7 (2024), doi:10.1007/s41781-023-00106-9.

[17] D. Belayneh et al., *Calorimetry with deep learning: Particle simulation and reconstruction for collider physics*, Eur. Phys. J. C **80**, 688 (2020), doi:10.1140/epjc/s10052-020-8251-9.

[18] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger, *Getting high: High fidelity simulation of high granularity calorimeters with high speed*, Comput. Softw. Big Sci. **5**, 13 (2021), doi:10.1007/s41781-021-00056-0.

[19] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol and K. Krüger, *Decoding photons: Physics in the latent space of a BIB-AE generative network*, Europhys. J. Web Conf. **251**, 03003 (2021), doi:10.1051/epjconf/202125103003.

[20] E. Buhmann et al., *Hadrons, better, faster, stronger*, Mach. Learn.: Sci. Technol. **3**, 025014 (2022), doi:10.1088/2632-2153/ac7848.

[21] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional integration and sampling with normalizing flows*, Mach. Learn.: Sci. Technol. **1**, 045023 (2020), doi:10.1088/2632-2153/abab62.

[22] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, *Exploring phase space with neural importance sampling*, SciPost Phys. **8**, 069 (2020), doi:10.21468/SciPostPhys.8.4.069.

[23] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event generation with normalizing flows*, Phys. Rev. D **101**, 076002 (2020), doi:10.1103/PhysRevD.101.076002.

[24] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone and U. Köthe, *Invertible networks or partons to detector and back again*, SciPost Phys. **9**, 074 (2020), doi:10.21468/SciPostPhys.9.5.074.

[25] B. Stienen and R. Verheyen, *Phase space sampling and inference from weighted events with autoregressive flows*, SciPost Phys. **10**, 038 (2021), doi:10.21468/SciPostPhys.10.2.038.

[26] S. Bieringer, A. Butter, T. Heimel, S. Höche, U. Köthe, T. Plehn and S. T. Radev, *Measuring QCD splittings with invertible networks*, SciPost Phys. **10**, 126 (2021), doi:10.21468/SciPostPhys.10.6.126.

[27] M. Bellagente, M. Haussmann, M. Luchmann and T. Plehn, *Understanding event-generation networks via uncertainties*, SciPost Phys. **13**, 003 (2022), doi:10.21468/SciPostPhys.13.1.003.

[28] T. Bister, M. Erdmann, U. Köthe and J. Schulte, *Inference of cosmic-ray source properties by conditional invertible neural networks*, Eur. Phys. J. C **82**, 171 (2022), doi:10.1140/epjc/s10052-022-10138-x.

[29] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent, *Generative networks for precision enthusiasts*, SciPost Phys. **14**, 078 (2023), doi:10.21468/SciPostPhys.14.4.078.

[30] R. Winterhalder, V. Magerya, E. Villa, S. Jones, M. Kerner, A. Butter, G. Heinrich and T. Plehn, *Targeting multi-loop integrals with neural networks*, SciPost Phys. **12**, 129 (2022), doi:10.21468/SciPostPhys.12.4.129.

[31] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman, T. Plehn, D. Shih and R. Winterhalder, *Ephemeral learning-augmenting triggers with online-trained normalizing flows*, SciPost Phys. **13**, 087 (2022), doi:10.21468/SciPostPhys.13.4.087.

[32] R. Verheyen, *Event generation and density estimation with surjective normalizing flows*, SciPost Phys. **13**, 047 (2022), doi:10.21468/SciPostPhys.13.3.047.

[33] M. Leigh, J. A. Raine, K. Zoch and T. Golling, *ν-flows: Conditional neutrino regression*, SciPost Phys. **14**, 159 (2023), doi:10.21468/SciPostPhys.14.6.159.

[34] A. Butter, T. Heimel, T. Martini, S. Peitzsch and T. Plehn, *Two invertible networks for the matrix element method*, SciPost Phys. **15**, 094 (2023), doi:10.21468/SciPostPhys.15.3.094.

[35] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge* (2022), https://calochallenge.github.io/homepage.

[36] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 – Dataset 1*, Zenodo (2022), doi:10.5281/zenodo.6234054.

[37] B. Nachman, L. de Oliveira and M. Paganini, *Electromagnetic calorimeter shower images*, Mendeley Data (2017), doi:10.17632/pvn3xc3wy5.1.

[38] ATLAS collaboration, *Datasets used to train the generative adversarial networks used in ATLFast3*, CERN Open Data Portal (2021), doi:10.7483/OPENDATA.ATLAS.UXKX.TXBN.

[39] I. Kobyzev, S. J. D. Prince and M. A. Brubaker, *Normalizing flows: An introduction and review of current methods*, IEEE Trans. Pattern Anal. Mach. Intell. **43**, 3964 (2021), doi:10.1109/TPAMI.2020.2992934.

[40] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, J. Mach. Learn. Res. **22**, 1 (2021).

[41] D. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in *Proceedings of the 32nd international conference on machine learning*, Lille, France (2015).

[42] G. Papamakarios, T. Pavlakou and I. Murray, *Masked autoregressive flow for density estimation*, in *Advances in neural information processing systems 30*, Curran Associates, New York, USA, ISBN 9781510860964 (2017).

[43] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever and M. Welling, *Improved variational inference with inverse autoregressive flow*, in *Advances in neural information processing systems 29*, Curran Associates, New York, USA, ISBN 9781510838819 (2016).

[44] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Neural spline flows*, in *Advances in neural information processing systems 32*, Curran Associates, New York, USA, ISBN 9781713807933 (2016).

[45] J. A. Gregory and R. Delbourgo, *Piecewise rational quadratic interpolation to monotonic data*, IMA J. Numer. Anal. **2**, 123 (1982), doi:10.1093/imanum/2.2.123.

[46] M. Germain, K. Gregor, I. Murray and H. Larochelle, *Made: Masked autoencoder for distribution estimation*, in *Proceedings of the 32nd international conference on machine learning*, Lille, France (2015).

[47] A. van den Oord, *Parallel WaveNet: Fast high-fidelity speech synthesis*, in *Proceedings of the 35th international conference on machine learning*, Stockholm, Sweden (2018).

[48] L. N. Smith, *Cyclical learning rates for training neural networks*, in *IEEE winter conference on applications of computer vision (WACV)*, Santa Rosa, USA (2017), doi:10.1109/WACV.2017.58.

[49] L. N. Smith and N. Topin, *Super-convergence: Very fast training of neural networks using large learning rates*, in *Artificial intelligence and machine learning for multi-domain operations applications*, Baltimore, USA (2019), doi:10.1117/12.2520589.

[50] *WebPlotDigitizer* (2024), https://apps.automeris.io/wpd.

[51] ATLAS collaboration, *Fastcalogan training project*, Zenodo (2021), doi:10.5281/zenodo.5589623.

[52] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 – Dataset 2*, Zenodo (2022), doi:10.5281/zenodo.6366271.

[53] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 – Dataset 3*, Zenodo (2022), doi:10.5281/zenodo.6366324.

[54] C. R. Harris et al., *Array programming with NumPy*, Nature **585**, 357 (2020), doi:10.1038/s41586-020-2649-2.

[55] J. D. Hunter, *Matplotlib: A 2D graphics environment*, Comput. Sci. Eng. **9**, 90 (2007), doi:10.1109/MCSE.2007.55.

[56] F. Pedregosa et al., *Scikit-learn: Machine learning in Python*, J. Mach. Learn. Res. **12**, 2825 (2011).

[57] A. Collette, *Python and HDF5*, O'Reilly Media, Sebastopol, USA, ISBN 9781449367831 (2013).

[58] A. Paszke et al., *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in neural information processing systems 32*, Curran Associates, New York, USA, ISBN 9781713807933 (2016).

[59] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *nflows: Normalizing flows in PyTorch*, Zenodo (2020), doi:10.5281/zenodo.4296287.

[60] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, (arXiv preprint) doi:10.48550/arXiv.1412.6980.

[61] C. Guo, G. Pleiss, Y. Sun and K. Q. Weinberger, *On calibration of modern neural networks*, (arXiv preprint) doi:10.48550/arXiv.1706.04599.