Normalizing Flows for High-Dimensional Detector Simulations

Florian Ernst[1,2], Luigi Favaro[1], Claudius Krause[1,3], Tilman Plehn[1], and David Shih[4]

1 Institut für Theoretische Physik, Universität Heidelberg, Germany
2 Experimental Physics Department, CERN, Geneva, Switzerland
3 Institut für Hochenergiephysik (HEPHY), Österreichische Akademie der Wissenschaften (ÖAW), Vienna, Austria
4 NHETC, Department of Physics & Astronomy, Rutgers University, Piscataway, NJ USA

December 18, 2023

Abstract

Whenever invertible generative networks are needed for LHC physics, normalizing flows show excellent performance. A challenge is their scaling to high-dimensional phase spaces. We investigate their performance for fast calorimeter shower simulations with increasing phase space dimension. In addition to the standard architecture we also employ a VAE to compress the dimensionality. Our study provides benchmarks for invertible networks applied to the CaloChallenge.

Contents

1 Introduction

Simulations are a defining aspect of LHC physics, bridging experiment and fundamental theory and allowing for a proper interpretation of LHC measurements [1, 2]. The simulation chain starts from the hard interaction at the scattering vertex, and progresses through the radiation of soft particles, the decay of heavy, unstable particles, hadronization of colored states, and subsequent interaction of all particles in the event with the detector. In the development of LHC as a precision-hadron collider, the last step has become a major bottleneck in speed and precision, in particular the reproduction of the detailed interactions of incident and secondary particles within the calorimeters. Generating these calorimeter showers with GEANT4 [3–5], based on first principles, takes a substantial amount of the LHC computing budget. Without significant progress, simulations will be the limiting factor for all analyses at the high-luminosity upgrade of the LHC.

One development driving faster LHC simulations is the advent of deep generative networks. Such networks have shown great promise for LHC physics in the past few years, providing fast and accurate surrogates for simulations in high-dimensional phase spaces [6]. They learn the underlying probability distribution of events or calorimeter showers from a reference dataset and then generate new samples based on this learned distribution [7,8]. We have seen successful applications to all steps in the simulation chain [2], phase space integration [9–19]; parton showers [20–27]; hadronization [28–31]; detector simulations [32–61]; and end-to-end event generation [62–68].

For LHC physics, it is crucial that these networks are not used as black boxes, but their performance can be investigated, understood, and improved systematically [67, 69–73]. This is especially important when their conditional counterparts are used for inference [74–76], probabilistic unfolding [77–86], or anomaly detection [87–92].

In this paper, we will focus on the problem of building fast and accurate surrogate models for calorimeter shower simulation, using the technology of normalizing flows. We have seen in a number of contexts that normalizing flows are a promising technique for fast calorimeter simulation [41, 43, 49, 73], but there are also major challenges with scaling them up to more granular (higher-dimensional) calorimeters [50, 51, 56, 61]. These challenges are especially interesting because recently, continuous-time generative models (diffusion models and continuous normalizing flows trained with flow-matching) have entered LHC physics [25–27, 47, 55, 57, 58, 68, 72, 76, 93–95] and show impressive performance which is not as limited by the dimensionality of the data. However, their gain in expressivity comes at the expense of slower generation, leading to an interesting trade-off between speed and quality of generated events or showers.

Here, we will build on previous works [50, 51, 56, 61] attempting to scale up normalizing flows to higher-granularity calorimeters. Focusing on the datasets [96–99] of the Fast Calorimeter Simulation Challenge [100], we will tackle this problem in two ways.

- First, we will show how impressive gains in both speed and accuracy can be achieved by switching from the fully-autoregressive flows of [41, 43, 49, 51, 56] to flows based on coupling layers [101–103] which are equally fast in the sampling and density estimation directions. Following the terminology of [104, 105], we will refer to coupling-layer based flows as *invertible neural networks* (INNs) throughout this work. Using the INN framework, we are able to obtain state-of-the-art results on dataset 1 (pions and photons) and dataset 2 of the CaloChallenge.

- Second, to reach the dimensionality of dataset 3, we will combine the INN framework with a VAE. Similar to the approaches of [50] and [106], we will train the INN on the (much lower-dimensional) latent space of a VAE fit to the showers of dataset 3. Then sampling from the INN and passing this through the decoder of the VAE, we will obtain a surrogate

model for dataset 3. We will see that the results here, while not state-of-the-art in terms of quality, are very fast to generate, so could fill out another point in the Pareto frontier of fast calorimeter shower simulation.

The paper starts by introducing the CaloChallenge datasets in Sec. 2. In Sec. 3 we introduce our fast INN version [104, 105] of a normalizing flow, as well as a VAE+INN combination. In Sec. 4 we discuss their performance on the different dataset, with increasing phase space dimensionality and including learned classifier weights. We conclude and provide timing information in Sec. 5. In the Appendices we provide details on the different network architectures and hyperparameters and compare the INN performance to CaloFlow.

2 Datasets

We use the public datasets [96–99] of the Fast Calorimeter Simulation Challenge [100]. They consist of showers simulated with GEANT4 for different incident particles. The general geometry is the same across all datasets: the detector volume is segmented into layers in the direction of the incoming particle. Each layer is segmented along polar coordinates in radial (r) and angular (α) bins. A shower is given as the incident energy of the incoming particle and the energy depositions in each voxel.

Dataset 1 (DS1) provides calorimeter showers for central photons and charged pions. They have been used in ATLFAST3 [42]. The voxelizations of the 5 photon layers and 7 pion layers in radial and angular bins ($n_r \times n_\alpha$) are

$$
\begin{aligned}
\text{photons} \quad & 8 \times 1,\ 16 \times 10,\ 19 \times 10,\ 5 \times 1,\ 5 \times 1 \\
\text{pions} \quad & 8 \times 1,\ 10 \times 10,\ 10 \times 10,\ 5 \times 1,\ 15 \times 10,\ 16 \times 10,\ 10 \times 1
\end{aligned}
\tag{1}
$$

This gives 368 voxels for photons and 533 voxels for pions. The incoming particles are simulated for 15 different incident energies $E_{\text{inc}} = 256$ MeV ... 4.2 TeV, increasing by factors of two, with the sample sizes given in Tab. 1. The original ATLAS dataset does not require an energy threshold. The effect of a threshold on the shower distributions at the detector cell level requires further studies. We require $E_{\text{min}} = 1$ MeV to all generated voxels, motivated by the readout threshold of the calorimeter cells and the fact that photon showers require a minimum cell energy of 10 MeV to cluster and pion showers start clustering at 300 MeV [107].

Datasets 2 and 3 (DS2/3) are not modeled after existing detectors. They assume 45 layers of active silicon detector (thickness 0.3 mm), alternating with inactive tungsten absorber layers (thickness 1.4 mm) at $\eta = 0$. Each dataset contains 100,000 GEANT4 positron showers with log-uniform $E_{\text{inc}} = 1$ ... 1000 GeV. The only difference between the two datasets is the voxelization. In dataset 2, each layer is divided into 16×9 angular and radial voxels, defining 6480 voxels in total. Dataset 3 uses 50×18 voxels per layer or 40,500 voxels in total. The minimal recorded energy per voxel for these two datasets is 15.15 keV.

E_{inc}	256 MeV ... 131 GeV	256 GeV	0.512 GeV	1.04 TeV	2.1 TeV	4.2 TeV
photons	10000 per energy	10000	5000	3000	2000	1000
pions	10000 per energy	9800	5000	3000	2000	1000

Table 1: Sample sizes for different incident energies in dataset 1.

3 CaloINN

We study two different network architectures. First, we benchmark a standard INN and demonstrate its precision and generation speed especially for low-dimensional phase space. Second, we embed this INN in a VAE, with the goal of describing datasets 2 and 3 with the same physics content, but a much larger phase space dimensionality.

3.1 INN

Normalizing flows describe bijective mappings between a (Gaussian) latent space r and the physical phase space x,

$$p_{\text{latent}}(r) \quad \underset{\leftarrow \overline{G}_\theta(x)}{\overset{G_\theta(r)\rightarrow}{\longleftrightarrow}} \quad p_{\text{model}}(x) \sim p_{\text{data}}(x)\,. \tag{2}$$

$\overline{G}_\theta(x)$ denotes the inverse transformation to $G_\theta(r)$. The INN variant [104,105] of normalizing flows is completely symmetric in the two directions. After training the network, $p_{\text{data}}(x) \sim p_{\text{model}}(x)$, we use the INN to sample $p_{\text{model}}(x)$ from $p_{\text{latent}}(r)$ [6].

The building block of our INN architecture is the coupling layer [101–103]. We replace the standard affine layer by a more expressive spline transformation. We use different spline transformations, depending on speed and expressivity. For datasets 1, we employ a rational quadratic spline [108], while for dataset 2 we use a cubic spline [109]. All INN hyperparameters are given in Tab 3.

The INN is implemented using the FREIA[§] package [110], and is trained with a likelihood loss,

$$\mathcal{L}_{\text{INN}} = -\Big\langle \log p_{\text{model}}(x) \Big\rangle_{p_{\text{data}}} = -\Bigg\langle \log p_{\text{latent}}\big(\overline{G}_\theta(x)\big) + \log \bigg| \frac{\partial \overline{G}_\theta(x)}{\partial x} \bigg| \Bigg\rangle_{p_{\text{data}}}\,. \tag{3}$$

The first term ensures that the latent representation remains Gaussian, while the second term constructs the correct transformation to the phase space distribution. Given the structure of $\overline{G}_\theta(x)$ and the latent distribution p_{latent}, both terms can be computed efficiently.

As a noteworthy preprocessing we normalize each shower to the layer energy. The energy information is encoded as

$$u_0 = \frac{\sum_i E_i}{E_{\text{inc}}} \qquad \text{and} \qquad u_i = \frac{E_i}{\sum_{j\geq i} E_j}\,, \tag{4}$$

in terms of the energy depositions per layer E_i. The u_i are appended to the list of voxels for each shower. We do not explore a separate training for the energy and the voxel dimensions which would simplify the learning process of the energy dimensions. We train the INN on the full data, conditioned on the logarithm of the incident energies. Unlike, for instance, CaloFlow [43,49,56] we train a single network without any distillation. We provide the details of the preprocessing in App. A.

3.2 VAE+INN

The problem with the INN is the scaling towards dataset 3 with its high-dimensional phase space of 40k voxels. To solve this scaling problem we introduce an additional VAE to reduce the dimensionality of the INN mapping. Differently from [50], we do not estimate the dimensionality of the manifold but rather optimize the reconstruction of the VAE while keeping a

---

[§]We provide the code in a Github repository at https://github.com/heidelberg-hepml/CaloINN

low-dimensional latent space. The VAE consists of a preprocessing block, an encoder-decoder combination, and a postprocessing block. Both, the decoder and the encoder are conditioned on the incident energies and additional energy variables. Therefore, we compress normalized showers in the latent space and jointly learn the energy and the latent variables with the INN. During generation, the INN samples into the latent space of the VAE, and the VAE decoder translates this information to the shower phase space. We set the latent space to 50 for dataset 1 and dataset 2, and to 300 for dataset 3. Other specifics of the network are different in the three datasets and are provided in App. A.

The goal of our standard βVAE is to learn to reconstruct the input data. We assume a Gaussian distribution for the encoder network $E(z|x)$. The VAE loss for the compression is

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \beta D_{\text{KL}}[E(z|x), p_{\text{latent}}(z)] \,, \tag{5}$$

with the usual binary cross entropy loss and the Gaussian prior. For a Gaussian encoder the KL-divergence can be computed analytically, and the coupling strength is $\beta = 10^{-9}$.

For the decoder we use a Bernoulli likelihood, because it outperforms for example its Gaussian counterpart. The Gaussian decoder does not model the shower geometry well, and it under-populates the low-energy regions. The continuous Bernoulli distribution [111] leads to instabilities, as the average energy deposition in the normalized space is close to zero. We use a Bernoulli decoder,

$$D(x|\lambda(z)) = \lambda(z)^x (1 - \lambda(z))^{1-x} \,, \tag{6}$$

defining the combined VAE loss

$$\mathcal{L}_{\text{VAE}} = \left\langle \left\langle x \log \lambda + (1-x) \log(1-\lambda) \right\rangle_{z \sim E(z|x)} + \beta \left[1 + \log \sigma_E^2 - \mu_E^2 - \sigma_E^2 \right] \right\rangle_{x \sim p_{\text{data}}} \,. \tag{7}$$

Because the Bernoulli distribution gives a binary probability we use its continuous mean λ as the prediction for the individual voxels.

The remaining differences between the unit-Gauss prior in the latent space and the encoder are mapped by the INN. Applying a 2-step training we first train the VAE and then train the INN given the learned latent space. This means we pass the encoder means and the standard deviations, as well as the energy variables to the INN. The INN is trained as described above, mapping the latent representation of the VAE to a standard Gaussian. As for the full INN, the energy information is encoded following Eq (4) and learned by the latent flow. Both encoder and decoder of the VAE are conditioned to these variables.

For the larger datasets 2 and 3, we employ a mixture of a convolutional and a fully connected VAE. Our assumption is that the calorimeter layers do not require a full correlation, so we can simplify the structure by compressing consecutive layers jointly in a first-step compression. We use an architecture with fully connected sub-blocks, resembling a kernel architecture with a kernel size k (number of jointly encoded calorimeter layers) and a stride s (distance between two neighboring kernel blocks). After this first compression we concatenate these latent sub-spaces and compress them a second time into our final latent space. For the decoding we reverse this two-step structure. The overlapping regions of the fully connected kernel blocks are summed over.

4 Results

The main physics reason for specific shower features is the incident energy. Low-energy showers will interact with only a few layers of the calorimeter and quickly widen, leading to a broad

center-of-energy distribution in earlier calorimeter layers and a high sparsity in the given voxelization. High-energy showers penetrate the calorimeter more deeply. They will be collimated in the initial layers and have low sparsity since each shower is likely to deposit energy in each voxel.

To see if the ML-learned showers reflect these physics properties, we look at physics-motivated and high-level features. Given a shower with energy depositions $\mathcal{I}$, we look at the center of energy and its width for each layer,

$$\langle \zeta \rangle = \frac{\zeta \cdot \mathcal{I}}{\sum_i \mathcal{I}_i} \qquad \text{and} \qquad \sigma_{\langle \zeta \rangle} = \sqrt{\frac{\zeta^2 \cdot \mathcal{I}}{\sum_i \mathcal{I}_i} - \langle \zeta \rangle^2} \qquad \text{for} \qquad \zeta = \eta, \phi \; ; \qquad (8)$$

where $\sum_i$ runs over the voxels in one layer. We also look at the energy deposition in each layer; the layer sparsity; and for dataset 1, the ratio $E_{\text{tot}}/E_{\text{inc}}$ for each discrete incident energy.

To analyze the quality of our generative networks in more detail and to identify failure modes, we train a classifier $D(x)$ on the voxels, to distinguish GEANT4 showers from generated showers [41, 43, 73]. By the Neyman-Pearson lemma the trained classifier approximates the likelihood-ratio. This means we can compute the correction weight [73] and use the weight distributions as an evaluation metric

$$w(x) = \frac{D(x)}{1 - D(x)} \approx \frac{p_{\text{data}}}{p_{\text{model}}}(x) \, . \qquad (9)$$

For these weights it is crucial that we evaluate them on the training and on the generated datasets combined, because typical failure modes correspond to tails for one of the two datasets [73]. In addition, we always check if showers with especially small or large weights cluster in phase space, allowing us to identify failure modes of the respective generative network.

<div style="text-align:right">understanding of failure mode in phase space</div>

4.1 Dataset 1 photons

We start with the photons in dataset 1, the simplest case in terms of dimensionality and of complexity, since photons only undergo a handful interactions in the calorimeter. In this established benchmark normalizing flows are known to excel. We summarize the most interesting high-level features for the GEANT4 training data, the INN generator, and the VAE+INN generator in Fig. 1.

For instance for the calorimeter layer 2, we first look at the shower shape in rapidity. Dataset 1 is not a symmetric in η and ϕ, because the shower were not generated around $\eta = \phi = 0$. All showers have the same mean width, regardless of the incident energy. This is captured by both networks at the level of 5% to 20%. A failure mode of the INN is the region $\sigma_{\langle \eta \rangle} < 20$ mm. where the network undershoots the training data by up to 30%. A peculiar feature of these distributions is a small peak at zero, which occurs when at most one voxel per layer receives a hit. These cases are better reproduced by the VAE, whereas the INN tends to produce slightly more collimated showers.

The sparsity λ_2 in the same layer is determined by the energy threshold of 1 MeV. The INN matches the truth over the entire λ-range to 10%, while the VAE struggles. In particular, its showers have too many active voxels, leading to the mis-modeled peak close to zero.

Next, we show the energy depositions in layers 0 and 2. Both networks show comparable performance over the entire energy range and are challenged by the sharp increase in energy around 100 MeV. The energy in layer-2 highlights the effect of discrete incident energies. High-energy showers at fixed energy deposit a similar amount of energy creating steps in the histogram.

Finally, the ratio $E_{\text{tot}}/E_{\text{inc}}$ exhibits a small bias in the energy generation for the VAE+INN towards low energies, artifact of the final threshold in the architecture. For smaller incident

energies, more voxels are zero [51], which causes a problem for the VAE+INN because we already know that the sparsity is its weakness.

To illustrate the discrete structure of the incident energies in dataset 1, we collect E_{tot}/E_{inc} for each incident energy in Fig. 2. The incident energy, provided during training and generation, carries energy-dependent information about the shower. For instance, low-energy showers have a much broader energy ratio distribution, in contrast to high-energy showers. Both generative networks learn the conditional distribution on E_{inc} with deviations up to 30% in the tails.

4.2 Dataset 1 pions

The physics of hadronic showers is significantly more complex than photon showers, so it is interesting to see how our INNs perform for a low-dimensional calorimeter simulation of pions. As before, we show shower shapes, sparsity, energy depositions, and the fraction of deposited energy in Fig. 3.

For the shower shapes, both networks show small, percent-level deviations in the bulk of the distributions. In addition, the VAE+INN is smearing out secondary peaks of the distributions. Both networks generate slightly too wide showers, predominantly where the energy deposition ends up in a narrow band of the calorimeter.

The slightly reduced quality of the generated pion showers can also be seen in the sparsity, especially for the VAE+INN. The energy depositions indicate similar failure modes. The sharp cut at low energy is smeared to a different extent by the networks, and a second deviation appears in the low-density region before the sharp cut. From the ratio E_{tot}/E_{inc} we see that at all energies the fraction of deposited energy can be very different from shower to shower, leading to the wide energy distribution far from one.
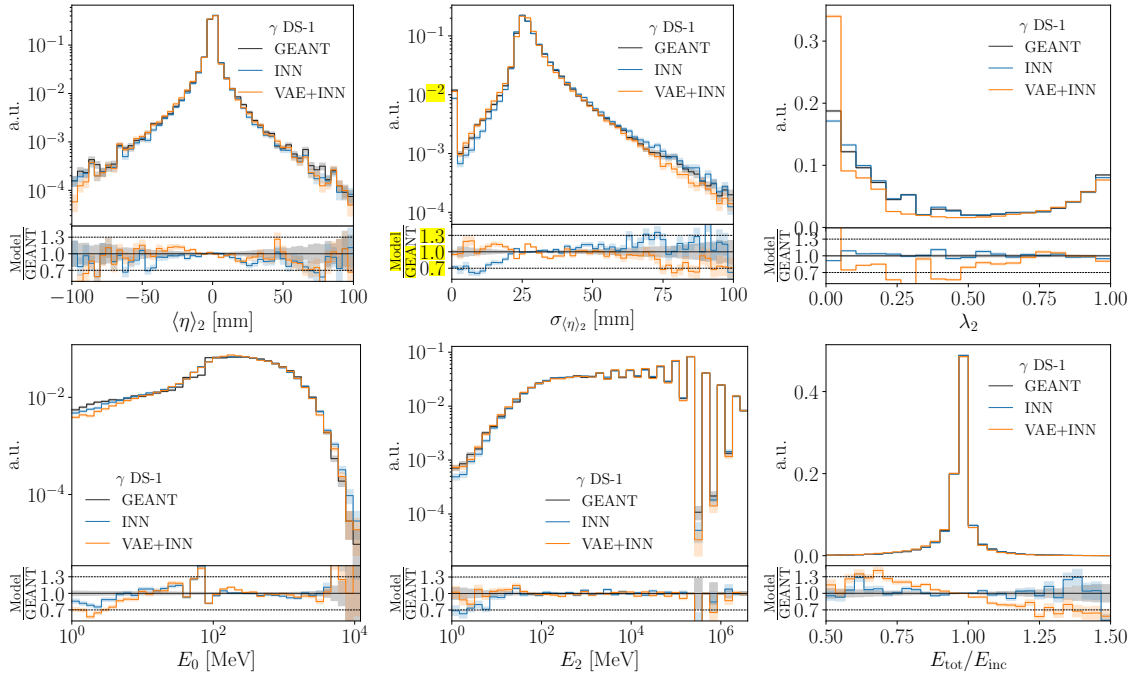
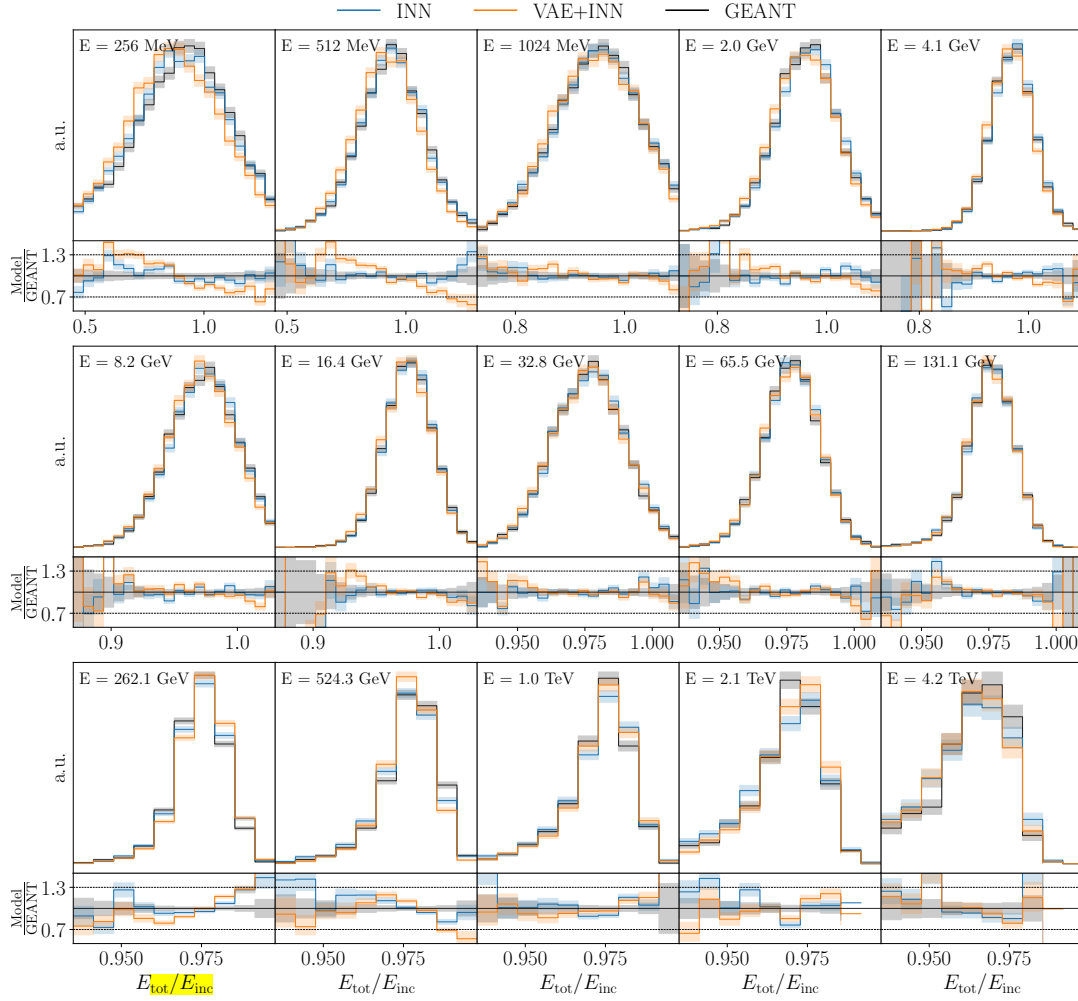Figure 1: Set of high-level features for γ showers in dataset 1, compared between GEANT4, INN, and VAE+INN.

Figure 2: Energy ratio E_{tot}/E_{inc} for each discrete incident energy, compared between GEANT4, INN, and VAE+INN for γ showers.

Low-level classifier

To evaluate the performance of our generative networks on dataset 1 systematically, we train a network to learn the classifier weights defined in Eq.(9) over the voxel space. In the left panel of Fig. 4 we show the weights for the γ-shower. We clearly see that the INN outperforms the VAE+INN. Its weight distribution peaks much closer to 1 and the corresponding AUC of 0.601 is substantially better than the corresponding AUC=0.936 of the VAE+INN.

More importantly, the INN does not show significant tails at large or small weights, which would indicate distinct failure modes. The peak of the VAE+INN, on the other hand, has moved away from 1. The tail at small weights indicates regions that are overpopulated by the network. We already know that this is the case for the sparsity. Large weights appear in phase space regions which the VAE+INN fails to populate, for instance the widths of the centers of energy.

In the right panel of Fig. 4 we see that the two generators perform more similar for π-showers. Both networks now show tails at small and large weights, two orders of magnitude away from one. This means there are regions that are over- and underpopulated by the generative networks. The fact that small weights appear for generated showers and large weights appear for the training data is generally expected. The AUCs for the INN and the VAE+INN are

8

0.805 and 0.864, respectively. The INN weight distribution is sharper around one, resulting in the smaller AUC compared to the combined VAE+INN approach. Altogether, we find that for dataset 1 with its limited dimensionality of 368 voxels for photons and 533 voxels for pions the INN works well, and that adding a VAE to compress the information does not help with the network performance.

4.3 Dataset 2 positrons

Dataset 2 is given in terms of 6480 voxels, the kind of dimensionality which will probe the limitations of the regular INN. The number of parameters for this network approaches 200M. The question will be, if the VAE+INN condensation helps the performance of the network. As before, we show a representative set of high-level features in Fig. 5. We choose layer 20, approximately in the middle of the calorimeter. It combines features from low-energy showers, which are absorbed in this region, and high-energy showers, which continue until the end of the calorimeter.

From the shower shapes we see that the INN-based architectures generate realistic showers at all energies. The training is stable and consistent across different runs of the same architecture. We only see sizeable deviations in the center of energy distributions in the first and last layers of the calorimeter, where there are less energy depositions. The agreement in phase space density between GEANT4 and the INN ranges from a few percent in the bulk of the distributions to 50% in the tails. Similar numbers apply to the width of the center of energy. The failure mode of the INN, regardless of the dataset, is an under-sampling of showers with width between the peak at zero and the secondary peak, for which the location depends on the layer but not on the incident energy. The VAE+INN generates showers of slightly worse quality. This is true for the shower shapes, for instance the peak position in $\sigma_{\langle \eta \rangle}$, but most obvious for the poorly reproduced sparsity.

The two networks learn the energy depositions in the layers in two very different spaces. The INN extracts them with a large number of voxels, while the VAE+INN compresses them
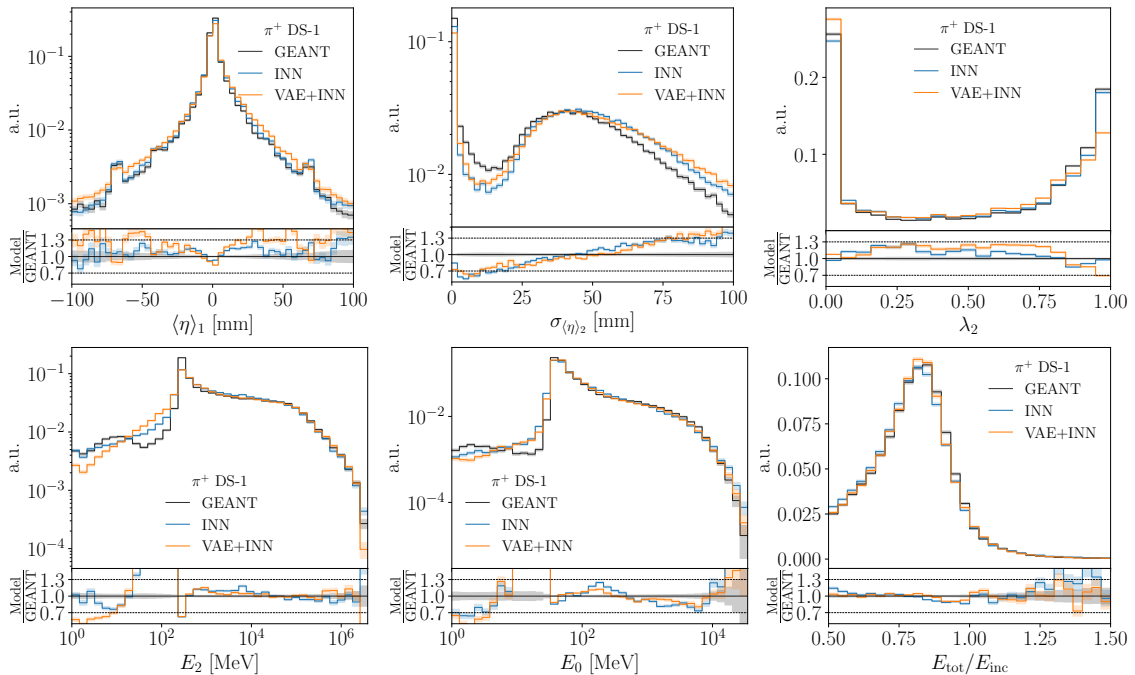

Figure 3: Set of high-level features for pion showers in dataset 1, compared between GEANT4, INN, and VAE+INN.
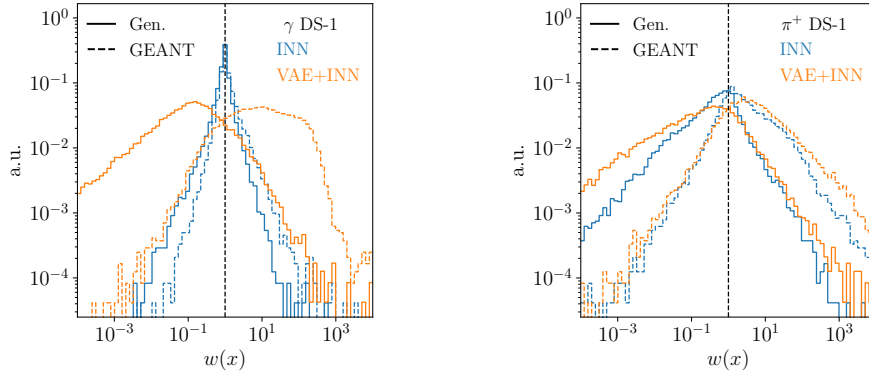
Figure 4: Classifier weight distributions in dataset 1. Classifier trained on γ showers (left) and π showers (right).

into a reduced space of around 50 additional features. This different expressivity is reflected in all energy distributions in Fig. 5. While in poorly populated tails the INN does better, for instance at low energies, the VAE+INN performs better for the main features in the central and high-energy regime. This is true for the layer-wise energies, but also for the ratio $E_{\text{tot}}/E_{\text{inc}}$.

Low-level classifier

Again, we show a systematic comparison for dataset 2 in terms of the classifier weights in the left panel Fig. 6. Compared to dataset 1, there is a clear deterioration of the INN performance for the higher-dimensional phase space. At small weights, the tail remains narrow, indicating that there are still no phase space regions where the network over-samples the true phase space distribution. For large weights the weight tail now extends to values larger than $w \sim 10^3$. This tail can be related to a recurrent under-sampling of showers with a small width of the center of energy in each layer, as seen in Fig. 5.

The classifier evaluating the VAE+INN generator highlights a few important structures as well. First, we have a clear over-sampled region in phase space with weights $w \sim 10^{-2}$, which we can relate to the center of energy distribution as well. As mentioned before, the VAE+INN over-samples showers with width close to the mean shower width. The classifier confirms this major failure mode. For the large-weight tail we checked that the under-sampled showers do not cluster in the same way, but are distributed over phase space, including tails of distributions.

The AUC values of the classifiers for dataset 2, 0.703 for the INN and 0.916 for the VAE+INN, confirm the challenge of the INN, especially relative to the well-modelled γ-showers in dataset 1. However, adding a VAE does not significantly improve the situation as long as it is technically possible to train an INN.

4.4 Dataset 3 positrons

Finally, we tackle dataset 3, which includes the same physics as dataset 2, but over a much higher-dimensional and extremely sparsely populated phase space. For this dataset we cannot train an INN without dimensionality reduction, so we only show VAE+INN results in Fig. 7. As expected, the performance is worse than for dataset 2, but the training is stable across different training runs. One problem is a worsening reconstructions of the centroids and widths for the later layers, which is likely related to the small average energy deposition per voxel. The maximum in the width distributions is overpopulated by the VAE+INN. For the energy distributions the VAE+INN is doing reasonably well, with serious deviations only in the low-energy tails.
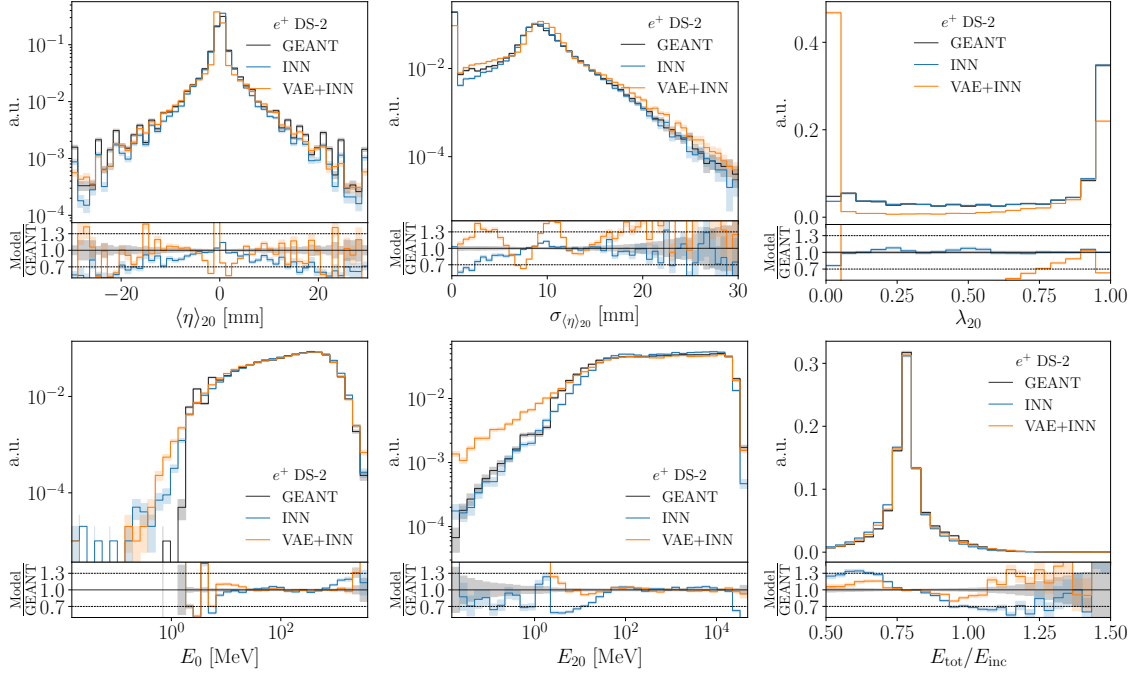
Figure 5: Set of high-level features for positron showers in dataset 2, compared between GEANT4, INN, and VAE+INN.

The classifier weights for the VAE+INN generating dataset 3 are shown in the right panel of Fig. 6. Even though the generative task is considerably harder, the learned weight distribution broadens centrally, but shows smaller tails than for dataset 2. The reason is that not only the generative network, but also our simple classifier are reaching their limits. However, the bulk

	Batch size	INN			
		1-photon	1-pion	2-positron	3-positron
GPU	1	24.79 ± 0.49	24.76 ± 0.35	50.90 ± 0.37	
	100	0.385 ± 0.002	0.406 ± 0.003	1.900 ± 0.026	
	10000	0.162 ± 0.002	0.191 ± 0.006	exceeding memory	
CPU	1	17.48 ± 0.09	18.88 ± 0.33	117.5 ± 1.8	
	100	0.827 ± 0.028	1.004 ± 0.047	14.26 ± 0.18	
	10000	0.510 ± 0.008	0.719 ± 0.016	15.24 ± 1.36	
	Batch size	VAE+INN			
		1-photon	1-pion	2	3
GPU	1	33.64 ± 0.32	33.54 ± 0.23	40.55 ± 0.40	$43.13 \pm 1.4^*$
	100	0.507 ± 0.005	0.544 ± 0.007	1.05 ± 0.02	3.44 ± 0.04
	10000	0.180 ± 0.002	0.228 ± 0.003	0.748 ± 0.018	—
CPU	1	20.83 ± 0.72	20.05 ± 0.13	28.11 ± 0.15	$39.46 \pm 1.1^*$
	100	0.582 ± 0.005	0.886 ± 0.015	1.94 ± 0.01	4.91 ± 0.01
	10000	0.328 ± 0.004	0.426 ± 0.014	1.25 ± 0.01	4.97 ± 0.08

Table 2: Per-shower generation times in ms. We show mean and standard deviation of 10 independent runs. The star indicates that only 10k samples were generated. The CPU timings were done with an Intel(R) Core(TM) i9-7900X at 3.30 GHz, the GPU timings with an NVIDIA TITAN V with 12GB RAM.
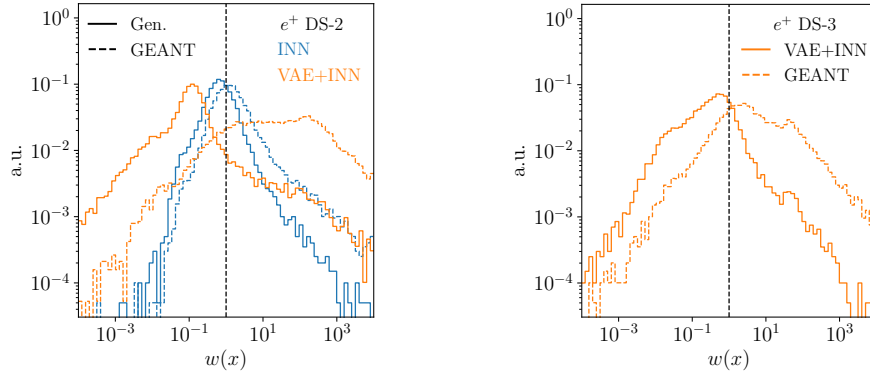
Figure 6: Classifier weight distributions. Classifier trained on e^+ showers on dataset 2 (left) and dataset 3 (right). The tails of dataset 3 should be taken with a grain of salt, giving the limitations of the simple classifier architecture.

of the classifier weight distribution clearly indicates that for dataset 3 the phase space density is mis-modelled by factors as large as 100 or 0.01 over large phase space regions. While the INN description of these position showers fails altogether, the VAE+INN results do not guarantee the level of precision we would expect for generative networks at the LHC.

4.5 Timing comparison

Finally, we time our networks using the CaloChallenge procedure. The INN architecture with modern coupling layers is ideally suited for fast and precise generation. We create a singularity container [112] of the software environment and take the time it takes to load the container, load the network, move it on the GPU, generate the samples, and save them to disk. In Tab. 2 we show the averaged results from ten runs. We observe a speed–up for increased batch size and when running on the GPU. The INN has a small advantage for dataset 1, but is unable to generate dataset 2 with the highest batch size and dataset 3 altogether. The VAE shows generation times at or below the millisecond mark.

5 Conclusions

Simulations are at the heart of the LHC program. Modern generative networks are showing great promise to improve their quality and speed, allowing them to meet the requirements of the high-luminosity LHC. In this paper, we have studied fast and precise normalizing flows, specifically an INN and a VAE+INN combination to generate calorimeter showers in high-dimensional phase spaces. As reference datasets we use the CaloChallenge datasets 1 to 3, with an increasing number of 368, 533, 6480, and 40,500 voxels.

For the simplest case, the dataset 1 photons, we have found that the INN generated high-fidelity showers and learns the phase space density of high-level features at the 10% level, except for failure modes which we can identify using high-level features and classifier weights over the low-level phase space. For this dataset, the VAE+INN shows no advantage, but less expressivity for example affecting the sparsity. For the pions in dataset 1 the INN faces more serious challenges, including mis-modeled features, and a wider range of learned classifier weights. The performance of the INN and the VAE+INN becomes much more similar.

For the positron showers in dataset 2 we have observed distinct advantages of the two networks, indicating that the compression by the VAE+INN helps learning the main features, but causes problems with low-energy depositions and the sparsity. Finally, the positrons in
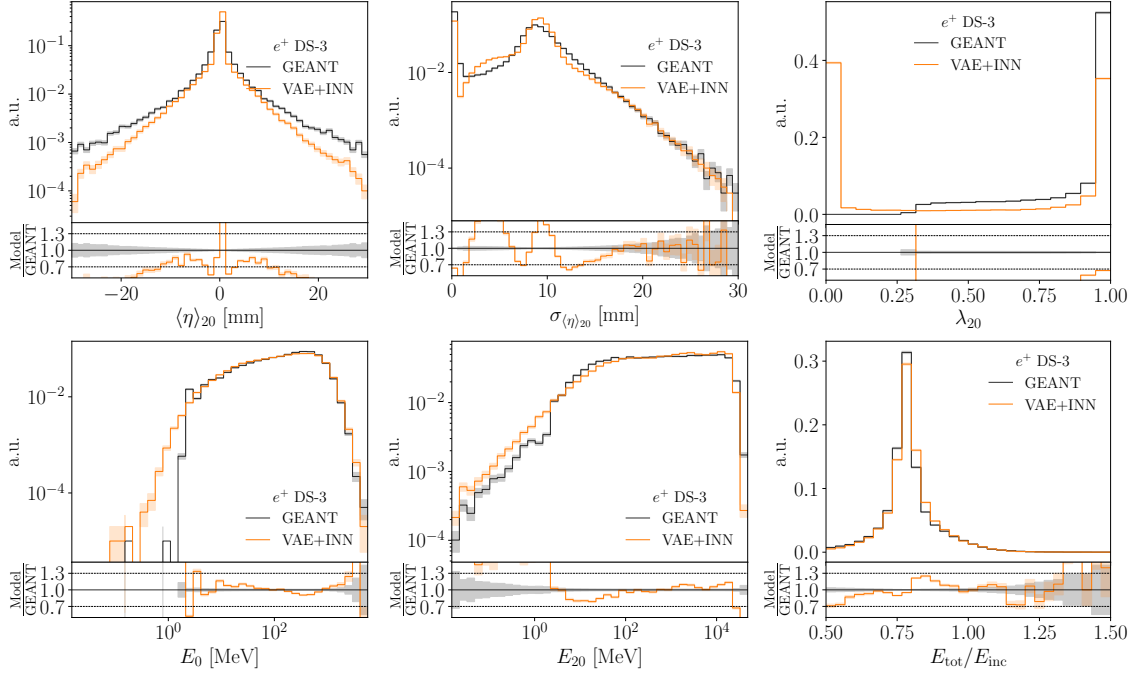
Figure 7: Set of high-level features for positron showers in dataset 3, compared between GEANT4, INN, and VAE+INN.

dataset 3 exceed the power of the plain INN, leaving us with the VAE+INN as the remaining option. The problem is that for this kind of phase space dimensionality the notion of precision-density estimation over phase space might not hold anymore for our network architectures.

Acknowledgements

A Network details

In this appendix we give some details on the network architectures and the preprocessing. The INN and the VAE+INN take layers normalized by the layer energy as input. The extra energy dimensions, calculated as in Eq. 4, are appended to the feature vector.

In the INN, we apply uniform noise and and a regularized logarithmic transformation with strength α. The transformation applied to the features is a rational quadratic spline [108] for dataset 1 and a cubic spline [109] for dataset 2. The prediction of the spline parameters is obtained with an MLP sub-network with 256 nodes for each hidden layer. To equally learn each dimension, we permute the order of the features after a transformation and normalize the output to mean zero and unit standard deviation with an ActNorm [103] layer. In the large-scale architecture, we stack twelve blocks to construct the INN with the additional preprocessing block.

The VAE preprocessing has a similar structure. After normalization, we apply an α-regularized logit transformation and a normalization to zero mean and unit standard deviation to each feature. We do not add noise during training and we set the latent dimension to 50 for dataset 1 and 2, and to 300 for dataset 3. We provide the full list of parameters in Tabs. 3 and 4.

Parameter	INN DS1/DS2	INN (with VAE)
coupling blocks	RQS / Cubic	RQS
# layers	4 / 3	3
hidden dimension	256	32
# of bins	10	10
# of blocks	12/14	18
# of epochs	450 / 200	200
batch size	512 / 256	256
lr scheduler	one cycle	one cycle
max. lr	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
$\beta_{1,2}$ (ADAM)	$(0.9, 0.999)$	$(0.9, 0.999)$
b	$5 \cdot 10^{-6}$	/
α	$1 \cdot 10^{-8}$	$1 \cdot 10^{-6}$

Table 3: Network and training parameters for the pure INN.

Parameter	VAE	
lr scheduler	Constant LR	
lr	$1 \cdot 10^{-4}$	
hidden dimension	5000, 1000, 500 (Set 1)	
	1500, 1000, 500 (Set 2)	
	2000, 1000, 500 (Set 3)	Inner VAE
latent dimension	50 (Set 1,2) / 300 (Set 3)	
# of epochs	1000	
batch size	256	
β	$1 \cdot 10^{-9}$	
threshold t [keV]	2 (Set 1) / 15.15 (Set 2,3)	
hidden dimension	1500, 800, 300	
kernel size	7	Kernel
kernel stride	3 (Set 2), 5 (Set 3)	

Table 4: Network and training parameters for the VAE-INN.

The classifiers trained for the evaluation of the generative networks are simple MLP net-

works with leaky ReLU. We use three layers with 512 nodes each and a batch size of 1000. The network is trained for 200 epochs with a learning rate of $2 \cdot 10^{-4}$ and the Adam optimizer with standard parameters. To prevent overfitting, especially for the larger datasets, we apply 30% dropout to each layer, and we reduce the learning rate on plateau with a decay factor of 0.1 and decay patience of 10. The splitting between training, validation, and testing is 60/20/20%. The selection of the best network is based on the best validation loss.

B CaloGAN dataset

In this section we discuss the INN performance on the even simpler CaloGAN dataset [32, 34]. The INN architecture is described in Sec. 3. To extract uncertainties from the generative network, we promote the deterministic INN to its Bayesian counterpart [67, 113]. The implementation follows the variational approximation substituting the linear layer with a mixture of uncorrelated Gaussians with learnable means and a diagonal covariance matrix. In practice, we only upgrade the last layer of each sub-network to a Bayesian layer [114].

Figure 8 showcases two high–level features as examples of the performance of the CaloINN as compared to the training data distribution generated by GEANT4. We show the brightest voxel distribution in layer 0, the average ϕ location of the showers in layer 2, and the width of the shower depth width defined as the standard deviation of s_d [41], with

$$s_d = \frac{\sum_{k=0}^{2} k E_k}{\sum_{k=0}^{2} E_k}. \tag{10}$$

The error bars in the GEANT4 distribution are the statistical errors while for the INN we estimate the uncertainties by sampling $N = 50$ times from the network and resampling the network parameters each time.

To evaluate our model on low-level observables, we resort again to classifier-based metrics. As already studied in a previous work [73], the INN samples are indistinguishable from the GEANT4 counterpart besides a few specific phase-space regions. We train a classifier on the
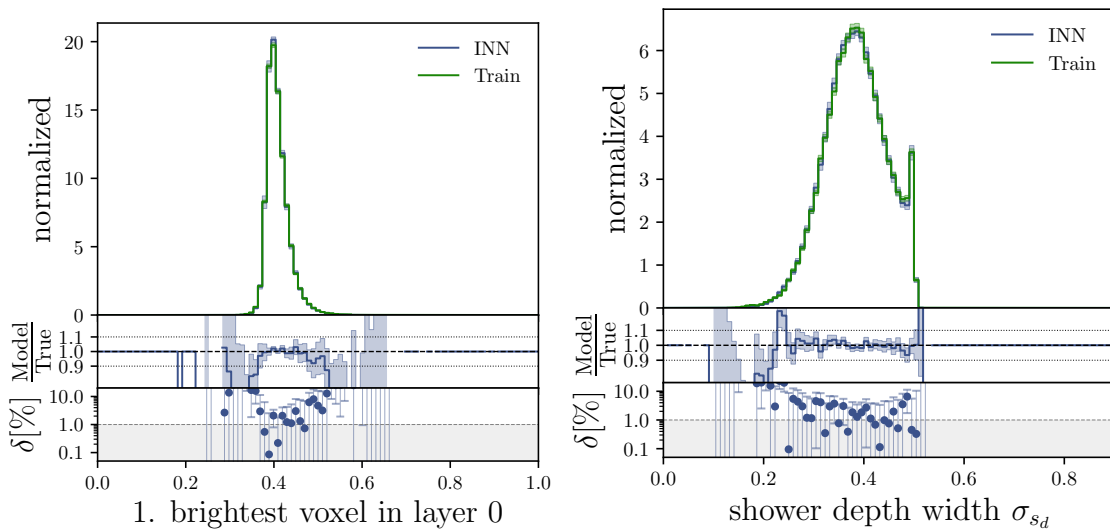
Figure 8: Comparison between CaloINN and GEANT4 on three high level features. Brightest voxel distribution in layer-0 (left), ϕ coordinate of the center of the shower in layer-2 (right), and width of the shower depth (bottom). Error bars on the INN are calculated after sampling from the Bayesian network $N = 50$ times.
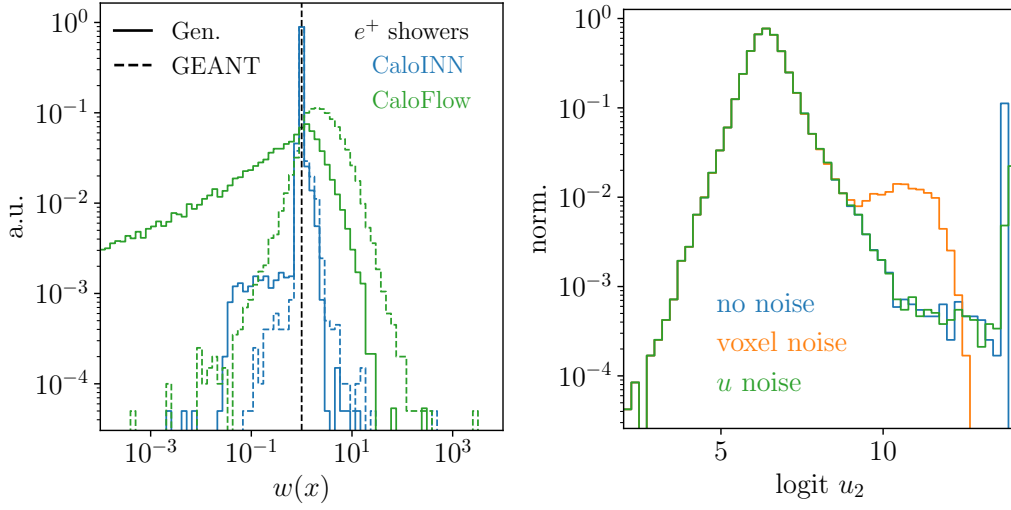
Figure 9: (left) Weight distribution of CaloFlow and CaloINN for e^+ showers. (right) u_2 distribution with different noise injections.

CaloFlow samples and find a large tail towards small weights. From clustering of the tail, we observe a clear dependence on the energy deposition total energy deposition. We link this effect to the learned energy variable $u_2 = E_1/(E_1 + E_2)$ and the noise injection procedure. If the noise is added at voxel-level, before calculating the additional energy variables, the flow learns distorted energy ratio distributions. Especially in the last layer, where the average energy deposition is smaller, this effect is larger. We summarize this effect in Fig. 9. We also provide the AUCs and the generation timings in Tab. 5.

AUC		CaloFlow [41]	CaloINN
	unnorm.	0.859(10)	0.525(2)
e^+	norm.	0.870(2)	0.598(3)
	hlf	0.795(1)	0.656(2)
	unnorm.	0.756(50)	0.530(2)
γ	norm.	0.796(2)	0.584(2)
	hlf	0.727(2)	0.671(2)
	unnorm.	0.649(3)	0.662(2)
π^+	norm.	0.755(3)	0.735(4)
	hlf	0.888(1)	0.786(4)

	Batch size	CaloFlow [43]	CaloINN
	1	$55.12 \pm 0.19^*$	$23.79 \pm 0.10^*$
GPU	100	0.744 ± 0.04	0.425 ± 0.005
	10000	0.249 ± 0.003	0.211 ± 0.003
	1	$119.9 \pm 0.9^*$	$46.39 \pm 3.18^*$
CPU	100	3.13 ± 0.11	1.14 ± 0.03
	10000	1.681 ± 0.004	0.72 ± 0.01

Table 5: (left) AUC of the two classifiers trained on the CaloFlow teacher and CaloINN samples. (right) Per shower generation timings in ms. We show mean and standard deviation of 10 independent runs of generating 100k showers. The star indicates that only 10k samples were generated in total.

References

[1] J. M. Campbell *et al.*, *Event Generators for High-Energy Physics Experiments*, In *Snowmass 2021* (2022), arXiv:2203.11110.

[2] S. Badger *et al.*, *Machine learning and LHC event generation*, SciPost Phys. **14**(4), 079 (2023), doi:10.21468/SciPostPhys.14.4.079, arXiv:2203.07460.

[3] S. Agostinelli *et al.*, *GEANT4–a simulation toolkit*, Nucl. Instrum. Meth. A **506**, 250 (2003), doi:10.1016/S0168-9002(03)01368-8.

[4] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce Dubois, M. Asai, G. Barrand, R. Capra, S. Chauvie, R. Chytracek, G. Cirrone, G. Cooperman *et al.*, *Geant4 developments and applications*, IEEE Transactions on Nuclear Science **53**(1), 270 (2006), doi:10.1109/TNS.2006.869826.

[5] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso, E. Bagli, A. Bagulya, S. Banerjee, G. Barrand, B. Beck, A. Bogdanov *et al.*, *Recent developments in geant4*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **835**, 186 (2016), doi:https://doi.org/10.1016/j.nima.2016.06.125, https://www.sciencedirect.com/science/article/pii/S0168900216306957.

[6] T. Plehn, A. Butter, B. Dillon and C. Krause, *Modern Machine Learning for LHC Physicists* (2022), arXiv:2211.01421.

[7] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying event samples*, SciPost Phys. **10**(6), 139 (2021), doi:10.21468/SciPostPhys.10.6.139, arXiv:2008.06545.

[8] S. Bieringer, A. Butter, S. Diefenbacher, E. Eren, F. Gaede, D. Hundhausen, G. Kasieczka, B. Nachman, T. Plehn and M. Trabs, *Calomplification — the power of generative calorimeter models*, JINST **17**(09), P09028 (2022), doi:10.1088/1748-0221/17/09/P09028, arXiv:2202.07352.

[9] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks* (2017), arXiv:1707.00028.

[10] M. D. Klimek and M. Perelstein, *Neural Network-Based Approach to Phase Space Integration*, SciPost Phys. **9**, 053 (2020), doi:10.21468/SciPostPhys.9.4.053, arXiv:1810.11509.

[11] I.-K. Chen, M. D. Klimek and M. Perelstein, *Improved Neural Network Monte Carlo Simulation*, SciPost Phys. **10**, 023 (2021), doi:10.21468/SciPostPhys.10.1.023, arXiv:2009.07819.

[12] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*, Mach. Learn. Sci. Tech. **1**(4), 045023 (2020), doi:10.1088/2632-2153/abab62, arXiv:2001.05486.

[13] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, *Exploring phase space with Neural Importance Sampling*, SciPost Phys. **8**(4), 069 (2020), doi:10.21468/SciPostPhys.8.4.069, arXiv:2001.05478.

[14] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event Generation with Normalizing Flows*, Phys. Rev. D **101**(7), 076002 (2020), doi:10.1103/PhysRevD.101.076002, arXiv:2001.10028.

[15] K. Danziger, T. Janßen, S. Schumann and F. Siegert, *Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates*, SciPost Phys. **12**(5), 164 (2022), doi:10.21468/SciPostPhys.12.5.164, arXiv:2109.11964.

[16] T. Heimel, R. Winterhalder, A. Butter, J. Isaacson, C. Krause, F. Maltoni, O. Mattelaer and T. Plehn, *MadNIS – Neural Multi-Channel Importance Sampling* (2022), arXiv:2212.06172.

[17] T. Janßen, D. Maître, S. Schumann, F. Siegert and H. Truong, *Unweighting multijet event generation using factorisation-aware neural networks*, SciPost Phys. **15**, 107 (2023), doi:10.21468/SciPostPhys.15.3.107, arXiv:2301.13562.

[18] E. Bothmann, T. Childers, W. Giele, F. Herren, S. Hoeche, J. Isaacson, M. Knobbe and R. Wang, *Efficient phase-space generation for hadron collider event simulation*, SciPost Phys. **15**, 169 (2023), doi:10.21468/SciPostPhys.15.4.169, arXiv:2302.10449.

[19] T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn and R. Winterhalder, *The MadNIS Reloaded* (2023), arXiv:2311.01548.

[20] L. de Oliveira, M. Paganini and B. Nachman, *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis*, Comput. Softw. Big Sci. **1**(1), 4 (2017), doi:10.1007/s41781-017-0004-6, arXiv:1701.05927.

[21] A. Andreassen, I. Feige, C. Frye and M. D. Schwartz, *JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics*, Eur. Phys. J. C **79**(2), 102 (2019), doi:10.1140/epjc/s10052-019-6607-9, arXiv:1804.09720.

[22] E. Bothmann and L. Debbio, *Reweighting a parton shower using a neural network: the final-state case*, JHEP **01**, 033 (2019), doi:10.1007/JHEP01(2019)033, arXiv:1808.07802.

[23] K. Dohi, *Variational Autoencoders for Jet Simulation* (2020), arXiv:2009.04842.

[24] E. Buhmann, G. Kasieczka and J. Thaler, *EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets* (2023), arXiv:2301.08128.

[25] M. Leigh, D. Sengupta, G. Quétant, J. A. Raine, K. Zoch and T. Golling, *PC-JeDi: Diffusion for Particle Cloud Generation in High Energy Physics* (2023), arXiv:2303.05376.

[26] V. Mikuni, B. Nachman and M. Pettee, *Fast Point Cloud Generation with Diffusion Models in High Energy Physics* (2023), arXiv:2304.01266.

[27] E. Buhmann, C. Ewen, D. A. Faroughy, T. Golling, G. Kasieczka, M. Leigh, G. Quétant, J. A. Raine, D. Sengupta and D. Shih, *EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion* (2023), arXiv:2310.00049.

[28] P. Ilten, T. Menzo, A. Youssef and J. Zupan, *Modeling hadronization using machine learning*, SciPost Phys. **14**, 027 (2023), doi:10.21468/SciPostPhys.14.3.027, arXiv:2203.04983.

[29] A. Ghosh, X. Ju, B. Nachman and A. Siodmok, *Towards a deep learning model for hadronization*, Phys. Rev. D **106**(9), 096020 (2022), doi:10.1103/PhysRevD.106.096020, arXiv:2203.12660.

[30] J. Chan, X. Ju, A. Kania, B. Nachman, V. Sangli and A. Siodmok, *Fitting a deep generative hadronization model*, JHEP **09**, 084 (2023), doi:10.1007/JHEP09(2023)084, arXiv:2305.17169.

[31] C. Bierlich, P. Ilten, T. Menzo, S. Mrenna, M. Szewc, M. K. Wilkinson, A. Youssef and J. Zupan, *Towards a data-driven model of hadronization using normalizing flows* (2023), arXiv:2311.09296.

[32] 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**(4), 042003 (2018), doi:10.1103/PhysRevLett.120.042003, arXiv:1705.02355.

[33] L. de Oliveira, M. Paganini and B. Nachman, *Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters*, J. Phys. Conf. Ser. **1085**(4), 042017 (2018), doi:10.1088/1742-6596/1085/4/042017, arXiv:1711.08813.

[34] 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**(1), 014021 (2018), doi:10.1103/PhysRevD.97.014021, arXiv:1712.10321.

[35] 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**(1), 4 (2018), doi:10.1007/s41781-018-0008-x, arXiv:1802.03325.

[36] M. Erdmann, J. Glombitza and T. Quast, *Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network*, Comput. Softw. Big Sci. **3**(1), 4 (2019), doi:10.1007/s41781-018-0019-7, arXiv:1807.01954.

[37] D. Belayneh *et al.*, *Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics*, Eur. Phys. J. C **80**(7), 688 (2020), doi:10.1140/epjc/s10052-020-8251-9, arXiv:1912.06794.

[38] 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**(1), 13 (2021), doi:10.1007/s41781-021-00056-0, arXiv:2005.05334.

[39] *Fast simulation of the ATLAS calorimeter system with Generative Adversarial Networks*, Tech. Rep. ATL-SOFT-PUB-2020-006, CERN, Geneva, All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-SOFT-PUB-2020-006 (2020).

[40] 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* (2021), arXiv:2102.12491.

[41] C. Krause and D. Shih, *CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows*, Phys. Rev. D **107**(11), 113003 (2023), doi:10.1103/PhysRevD.107.113003, arXiv:2106.05285.

[42] 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, arXiv:2109.02551.

[43] C. Krause and D. Shih, *CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows*, Phys. Rev. D **107**(11), 113004 (2023), doi:10.1103/PhysRevD.107.113004, arXiv:2110.11377.

[44] E. Buhmann, S. Diefenbacher, D. Hundhausen, G. Kasieczka, W. Korcari, E. Eren, F. Gaede, K. Krüger, P. McKeown and L. Rustige, *Hadrons, better, faster, stronger*, Mach. Learn. Sci. Tech. **3**(2), 025014 (2022), doi:10.1088/2632-2153/ac7848, arXiv:2112.09709.

[45] C. Chen, O. Cerri, T. Q. Nguyen, J. R. Vlimant and M. Pierini, *Analysis-Specific Fast Simulation at the LHC with Deep Learning*, Comput. Softw. Big Sci. **5**(1), 15 (2021), doi:10.1007/s41781-021-00060-4.

[46] A. Adelmann *et al.*, *New directions for surrogate models and differentiable programming for High Energy Physics detector simulation*, In *Snowmass 2021* (2022), arXiv:2203.08806.

[47] V. Mikuni and B. Nachman, *Score-based generative models for calorimeter shower simulation*, Phys. Rev. D **106**(9), 092009 (2022), doi:10.1103/PhysRevD.106.092009, arXiv:2206.11898.

[48] *Deep generative models for fast photon shower simulation in ATLAS* (2022), arXiv:2210.06204.

[49] C. Krause, I. Pang and D. Shih, *CaloFlow for CaloChallenge Dataset 1* (2022), arXiv:2210.14245.

[50] J. C. Cresswell, B. L. Ross, G. Loaiza-Ganem, H. Reyes-Gonzalez, M. Letizia and A. L. Caterini, *CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds*, In *36th Conference on Neural Information Processing Systems* (2022), arXiv:2211.15380.

[51] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, C. Krause, I. Shekhzadeh and D. Shih, *L2LFlows: generating high-fidelity 3D calorimeter images*, JINST **18**(10), P10017 (2023), doi:10.1088/1748-0221/18/10/P10017, arXiv:2302.11594.

[52] H. Hashemi, N. Hartmann, S. Sharifzadeh, J. Kahn and T. Kuhr, *Ultra-High-Resolution Detector Simulation with Intra-Event Aware GAN and Self-Supervised Relational Reasoning* (2023), arXiv:2303.08046.

[53] A. Xu, S. Han, X. Ju and H. Wang, *Generative Machine Learning for Detector Response Modeling with a Conditional Normalizing Flow* (2023), arXiv:2303.10148.

[54] S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, K. Krüger, P. McKeown and L. Rustige, *New Angles on Fast Calorimeter Shower Simulation* (2023), arXiv:2303.18150.

[55] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *CaloClouds: Fast Geometry-Independent Highly-Granular Calorimeter Simulation* (2023), arXiv:2305.04847.

[56] M. R. Buckley, C. Krause, I. Pang and D. Shih, *Inductive CaloFlow* (2023), arXiv:2305.11934.

[57] V. Mikuni and B. Nachman, *CaloScore v2: Single-shot Calorimeter Shower Simulation with Diffusion Models* (2023), arXiv:2308.03847.

[58] O. Amram and K. Pedro, *Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation*, Phys. Rev. D **108**(7), 072014 (2023), doi:10.1103/PhysRevD.108.072014, arXiv:2308.03876.

[59] S. Diefenbacher, V. Mikuni and B. Nachman, *Refining Fast Calorimeter Simulations with a Schrödinger Bridge* (2023), arXiv:2308.12339.

[60] M. Faucci Giannelli and R. Zhang, *CaloShowerGAN, a Generative Adversarial Networks model for fast calorimeter shower simulation* (2023), arXiv:2309.06515.

[61] I. Pang, J. A. Raine and D. Shih, *SuperCalo: Calorimeter shower super-resolution* (2023), arXiv:2308.11700.

[62] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri and R. Verheyen, *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer*, Nature Commun. **12**(1), 2985 (2021), doi:10.1038/s41467-021-22616-z, arXiv:1901.00875.

[63] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, *LHC analysis-specific datasets with Generative Adversarial Networks* (2019), arXiv:1901.05282.

[64] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, *DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC*, JHEP **08**, 110 (2019), doi:10.1007/JHEP08(2019)110, arXiv:1903.02433.

[65] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC Events*, SciPost Phys. **7**(6), 075 (2019), doi:10.21468/SciPostPhys.7.6.075, arXiv:1907.03764.

[66] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco and Y. Li, *Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)* (2020), arXiv:2001.11103.

[67] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent, *Generative Networks for Precision Enthusiasts* (2021), arXiv:2110.13632.

[68] A. Butter, N. Huetsch, S. P. Schweitzer, T. Plehn, P. Sorrenson and J. Spinner, *Jet Diffusion versus JetGPT – Modern Networks for the LHC* (2023), arXiv:2305.10475.

[69] S. Diefenbacher, E. Eren, G. Kasieczka, A. Korol, B. Nachman and D. Shih, *DCTRGAN: Improving the Precision of Generative Models with Reweighting*, JINST **15**(11), P11004 (2020), doi:10.1088/1748-0221/15/11/P11004, arXiv:2009.03796.

[70] R. Winterhalder, M. Bellagente and B. Nachman, *Latent Space Refinement for Deep Generative Models* (2021), arXiv:2106.00792.

[71] B. Nachman and R. Winterhalder, *ELSA – Enhanced latent spaces for improved collider simulations* (2023), arXiv:2305.07696.

[72] M. Leigh, D. Sengupta, J. A. Raine, G. Quétant and T. Golling, *PC-Droid: Faster diffusion and improved quality for particle cloud generation* (2023), arXiv:2307.06836.

[73] R. Das, L. Favaro, T. Heimel, C. Krause, T. Plehn and D. Shih, *How to Understand Limitations of Generative Networks* (2023), arXiv:2305.16774.

[74] 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**(6), 126 (2021), doi:10.21468/SciPostPhys.10.6.126, arXiv:2012.09873.

[75] 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, arXiv:2210.00019.

[76] T. Heimel, N. Huetsch, R. Winterhalder, T. Plehn and A. Butter, *Precision-Machine Learning for the Matrix Element Method* (2023), arXiv:2310.07752.

[77] K. Datta, D. Kar and D. Roy, *Unfolding with Generative Adversarial Networks* (2018), arXiv:1806.00433.

[78] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn and R. Winterhalder, *How to GAN away Detector Effects*, SciPost Phys. **8**(4), 070 (2020), doi:10.21468/SciPostPhys.8.4.070, arXiv:1912.00477.

[79] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman and J. Thaler, *OmniFold: A Method to Simultaneously Unfold All Observables*, Phys. Rev. Lett. **124**(18), 182001 (2020), doi:10.1103/PhysRevLett.124.182001, arXiv:1911.09107.

[80] 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, arXiv:2006.06685.

[81] M. Backes, A. Butter, M. Dunford and B. Malaescu, *An unfolding method based on conditional Invertible Neural Networks (cINN) using iterative training* (2022), arXiv:2212.08674.

[82] M. Leigh, J. A. Raine, K. Zoch and T. Golling, *ν-flows: Conditional neutrino regression*, SciPost Phys. **14**(6), 159 (2023), doi:10.21468/SciPostPhys.14.6.159, arXiv:2207.00664.

[83] J. A. Raine, M. Leigh, K. Zoch and T. Golling, *$ν^2$-Flows: Fast and improved neutrino reconstruction in multi-neutrino final states with conditional normalizing flows* (2023), arXiv:2307.02405.

[84] A. Shmakov, K. Greif, M. Fenton, A. Ghosh, P. Baldi and D. Whiteson, *End-To-End Latent Variational Diffusion Models for Inverse Problems in High Energy Physics* (2023), arXiv:2305.10399.

[85] J. Ackerschott, R. K. Barman, D. Gonçalves, T. Heimel and T. Plehn, *Returning CP-Observables to The Frames They Belong* (2023), arXiv:2308.00027.

[86] S. Diefenbacher, G.-H. Liu, V. Mikuni, B. Nachman and W. Nie, *Improving Generative Model-based Unfolding with Schrödinger Bridges* (2023), arXiv:2308.12351.

[87] B. Nachman and D. Shih, *Anomaly Detection with Density Estimation*, Phys. Rev. D **101**, 075042 (2020), doi:10.1103/PhysRevD.101.075042, arXiv:2001.04990.

[88] A. Hallin, J. Isaacson, G. Kasieczka, C. Krause, B. Nachman, T. Quadfasel, M. Schlaffer, D. Shih and M. Sommerhalder, *Classifying anomalies through outer density estimation*, Phys. Rev. D **106**(5), 055006 (2022), doi:10.1103/PhysRevD.106.055006, arXiv:2109.00546.

[89] J. A. Raine, S. Klein, D. Sengupta and T. Golling, *CURTAINs for your sliding window: Constructing unobserved regions by transforming adjacent intervals*, Front. Big Data **6**, 899345 (2023), doi:10.3389/fdata.2023.899345, arXiv:2203.09470.

[90] A. Hallin, G. Kasieczka, T. Quadfasel, D. Shih and M. Sommerhalder, *Resonant anomaly detection without background sculpting*, Phys. Rev. D **107**(11), 114012 (2023), doi:10.1103/PhysRevD.107.114012, arXiv:2210.14924.

[91] T. Golling, S. Klein, R. Mastandrea and B. Nachman, *FETA: Flow-Enhanced Transportation for Anomaly Detection* (2022), arXiv:2212.11285.

[92] D. Sengupta, S. Klein, J. A. Raine and T. Golling, *CURTAINs Flows For Flows: Constructing Unobserved Regions with Maximum Likelihood Estimation* (2023), arXiv:2305.04646.

[93] A. Butter, T. Jezo, M. Klasen, M. Kuschick, S. Palacios Schweitzer and T. Plehn, *Kicking it Off(-shell) with Direct Diffusion* (2023), arXiv:2311.17175.

[94] E. Buhmann, F. Gaede, G. Kasieczka, A. Korol, W. Korcari, K. Krüger and P. McKeown, *CaloClouds II: Ultra-Fast Geometry-Independent Highly-Granular Calorimeter Simulation* (2023), arXiv:2309.05704.

[95] J. Birk, E. Buhmann, C. Ewen, G. Kasieczka and D. Shih, *Flow Matching Beyond Kinematics: Generating Jets with Particle-ID and Trajectory Displacement Information* (2023), arXiv:2312.00123.

[96] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 1*, https://doi.org/10.5281/zenodo.6234054 (2022).

[97] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 1 version 3*, https://doi.org/10.5281/zenodo.8099322 (2023).

[98] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 2*, https://doi.org/10.5281/zenodo.6366271 (2022).

[99] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 - dataset 3*, https://doi.org/10.5281/zenodo.6366324 (2022).

[100] M. Faucci Giannelli, G. Kasieczka, B. Nachman, D. Salamani, D. Shih and A. Zaborowska, *Fast calorimeter simulation challenge 2022 github page*, https://github.com/CaloChallenge/homepage (2022).

[101] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear Independent Components Estimation*, arXiv e-prints arXiv:1410.8516 (2014), arXiv:1410.8516.

[102] L. Dinh, J. Sohl-Dickstein and S. Bengio, *Density estimation using Real NVP*, arXiv e-prints arXiv:1605.08803 (2016), arXiv:1605.08803.

[103] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible 1x1 convolutions* (2018), arXiv:1807.03039.

[104] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother and U. Köthe, *Analyzing inverse problems with invertible neural networks* (2018), arXiv:1808.04730.

[105] L. Ardizzone, C. Lüth, J. Kruse, C. Rother and U. Köthe, *Guided image generation with conditional invertible neural networks* (2019), arXiv:1907.02392.

[106] P. Esser, R. Rombach and B. Ommer, *A disentangling invertible interpretation network for explaining latent representations* (2020), arXiv:2004.13166.

[107] M. F. Giannelli, *private communication* (2023).

[108] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Neural spline flows*, Advances in Neural Information Processing Systems **32**, 7511 (2019), 1906.04032.

[109] C. Durkan, A. Bekasov, I. Murray and G. Papamakarios, *Cubic-spline flows*, arXiv preprint arXiv:1906.02145 (2019), 1906.02145.

[110] L. Ardizzone, T. Bungert, F. Draxler, U. Köthe, J. Kruse, R. Schmier and P. Sorrenson, *Framework for Easily Invertible Architectures (FrEIA)* (2018-2022).

[111] G. Loaiza-Ganem and J. P. Cunningham, *The continuous bernoulli: fixing a pervasive error in variational autoencoders* (2019), arXiv:1907.06845.

[112] G. M. Kurtzer, V. Sochat and M. W. Bauer, *Singularity: Scientific containers for mobility of compute*, PLOS ONE **12**(5), 1 (2017), doi:10.1371/journal.pone.0177459, https://doi.org/10.1371/journal.pone.0177459.

[113] M. Bellagente, M. Haussmann, M. Luchmann and T. Plehn, *Understanding Event-Generation Networks via Uncertainties*, SciPost Phys. **13**(1), 003 (2022), doi:10.21468/SciPostPhys.13.1.003, arXiv:2104.04543.

[114] M. Sharma, S. Farquhar, E. Nalisnick and T. Rainforth, *Do bayesian neural networks need to be fully stochastic?* (2023), arXiv:2211.06291.