CaloMan: Fast generation of calorimeter showers with density estimation on learned manifolds

Jesse C. Cresswell
Layer 6 AI
jesse@layer6.ai

Brendan Leigh Ross
Layer 6 AI
brendan@layer6.ai

Gabriel Loaiza-Ganem
Layer 6 AI
gabriel@layer6.ai

Humberto Reyes-González
University of Genoa & INFN
hreyes@ge.infn.it

Marco Letizia
University Of Genoa & INFN
marco.letizia@edu.unige.it

Anthony L. Caterini
Layer 6 AI
anthony@layer6.ai

Abstract

Precision measurements and new physics searches at the Large Hadron Collider require efficient simulations of particle propagation and interactions within the detectors. The most computationally expensive simulations involve calorimeter showers. Advances in deep generative modelling – particularly in the realm of high-dimensional data – have opened the possibility of generating realistic calorimeter showers orders of magnitude more quickly than physics-based simulation. However, the high-dimensional representation of showers belies the relative simplicity and structure of the underlying physical laws. This phenomenon is yet another example of the *manifold hypothesis* from machine learning, which states that high-dimensional data is supported on low-dimensional manifolds. We thus propose modelling calorimeter showers first by learning their manifold structure, and then estimating the density of data across this manifold. Learning manifold structure reduces the dimensionality of the data, which enables fast training and generation when compared with competing methods.

1 Introduction

The accurate simulation of particle detectors is a primary component of the high energy physics program. It allows us to map the data collected by an experiment to a theoretical description of the fundamental interactions of nature. Experimental collaborations at the Large Hadron Collider (LHC) rely on Monte Carlo simulations of physics from first-principles to emulate the passage of billions of particles through the several stages of the LHC detectors. A particularly involved step of the simulation pipeline is the generation of calorimeter showers. An incident particle interacts with an active material in the calorimeter, producing a shower of secondary particles. In turn, the shower deposits its energy into an array of scintillators which measure the energy distribution.

Currently, LHC calorimeter shower generation is done with Geant4 [2–4], the state-of-the-art simulator of the passage of particles through matter. However, these simulations are very time consuming, taking up to tens of minutes per event. As the LHC enters its High Luminosity era, Monte Carlo generation will only become more complex and computationally expensive, emphasizing the need for alternative approaches enabling fast simulation.

Recent advances in the field of deep generative modelling present a solution: train a surrogate model designed to emulate calorimeter showers, using data on incident particles and their resulting showers collected from experiment or expensive simulation. Deep generative models (DGMs) aim to learn the distribution of their training data, and can generate new data by sampling from the learned distribution.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

In the context of calorimeter showers, various types of generative models have been used including generative adversarial networks (GANs) [20] by CaloGan [32], normalizing flows (NFs) [34, 11] by CaloFlow [25, 26], and score-based generative models (SGMs) [39] by CaloScore [31]. Each approach has aimed to improve on various aspects of the problem. GANs allow fast simulation, but not inference (the ability to evaluate the likelihood of a datapoint according to the learned distribution). SGMs provide excellent quality generated samples, but are slow to train and sample from. NFs allow inference and are trained by the seemingly principled approach of maximum-likelihood estimation.

The main drawback of NFs is that they model a density that has the same dimensionality as the input data. For calorimeter showers, the data is presented as the energy deposited per voxel of the calorimeter which can be thought of as a vector in $\mathbb{R}^n$, where n is the number of voxels. Typically, n is in the hundreds or thousands, so the representation of the data is high-dimensional. This is in contrast to the relatively simple and highly structured underlying physical processes. For example, in an electromagnetic calorimeter with an incident photon, most of the interactions can be described by quantum electrodynamics [17]. The true distribution of electromagnetic showers is unknown, but its dimensionality is likely to be much smaller than n. In the machine learning context, this is an example of the *manifold hypothesis*, which states that high-dimensional data clusters around a low-dimensional embedded submanifold in the ambient space [5, 33].

Maximum-likelihood models like NFs are built on the assumption that the underlying distribution possesses a full-dimensional probability density $p(x)$ in the data space. When the data is confined to a low-dimensional manifold, this assumption is false: the data manifold is a subset of measure zero, over which no continuous density can be integrated to obtain non-zero probabilities. This mismatch between the model and the data leads to a phenomenon known as *manifold overfitting* [28], in which the model can maximize likelihood by concentrating probability density around the manifold while incorrectly distributing the density along the manifold. The resulting phenomenon is illustrated in Fig. 1. As a result, maximum-likelihood is an ill-posed objective for manifold-supported data.

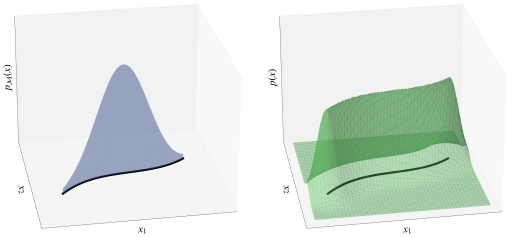

Figure 1: A low-dimensional density on a manifold (left), and a full-dimensional density model undergoing manifold overfitting (right). Although the full-dimensional model concentrates around the manifold, it distributes the density incorrectly along the manifold.

A common method when training NFs and other DGMs is to add full-dimensional Gaussian noise to the training data [45, 44, 25]. Adding noise can avoid the problem of manifold overfitting, but at the cost of no longer modelling the true distribution of the data, and not recovering the manifold structure [23] which could contain interesting physics.

In this work we propose to better model calorimeter showers by first learning their manifold structure and then estimating the density within [28]. This technique allows us to avoid manifold overfitting, while the dimensionality reduction step increases the efficiency of training and shower generation.

2 Method

2.1 Learning the Manifold of Calorimeter Showers

While it would be interesting to understand the exact distribution of showers from first-principles, for practical applications it suffices to learn the manifold $\mathcal{M}$ and probability density $p_{\mathcal{M}}(x)$ from data. To do so, we follow the general two-step procedure outlined by Loaiza-Ganem et al. [28]. The first step is to learn the manifold using a *generalized autoencoder*: any model that can construct low-dimensional latent encodings $z = \phi(x)$ of high-dimensional data x, and reconstruct the original data via an inverse transformation $x = \psi(z)$. The learned low-dimensional encodings z act as coordinates on the data manifold. The class of generalized autoencoders includes the autoencoder [38], variational autoencoder [24], Wasserstein autoencoder [41], bidirectional GAN [12, 14], and adversarial variational Bayes [30], among others.

The second step is to perform *density estimation* in the z coordinates, obtaining low-dimensional densities $p(z)$. Any DGM that explicitly constructs $p(z)$ can be used, including NFs, energy based

models [13], auto-regressive models [43], score-based models [40], and diffusion models [22]. Variational autoencoders and adversarial variational Bayes are also suitable as density estimators.

By capturing a probability density *within* the manifold, the two-step procedure evades the dimensionality mismatch in maximum-likelihood estimation. If required, probability densities in the data space can be computed using a change of metric from the low-dimensional coordinates into the high-dimensional space [19, 8, 36]:

$$p_{\mathcal{M}}(x) = p(z) \det \left(J_\psi(z)^T J_\psi(z) \right)^{-\frac{1}{2}}, \tag{1}$$

where $J_\psi(z)$ is the Jacobian of ψ evaluated at $z = \phi(x)$. Since the computation of Jacobians can be expensive, we emphasize that maximizing $p(z)$ directly is sufficient for training. In addition to being more mathematically principled, two-step models are highly performant: stable diffusion [35] is one example in which a diffusion model is trained on the data manifold learned by an autoencoder, and is capable of surprisingly photorealistic image generation.

Calorimeter shower simulation requires a model that can simulate showers conditional on incident energies. We assume the manifold contains showers x corresponding to all possible incident energies E_{inc} and aim to model the conditional density $p_{\mathcal{M}}(x \mid E_{\text{inc}})$. The generalized autoencoder only needs to map showers between the data space and latent space, which does not require knowledge of E_{inc}, hence we only add conditioning to the density estimator. When E_{inc} information is provided to the generalized autoencoder it can more easily cluster the shower data, leading to a segmented latent representation that may not generalize to unseen E_{inc}.

2.2 Intrinsic Dimension Estimation

In most examples of generalized autoencoders the dimensionality d of the latent representation is not learned, but is specified as a hyperparameter of the method. If d is not known *a priori*, it could be set empirically by trying many values and selecting the one with optimal validation performance, but training many models is computationally expensive. Instead, we employ a statistical estimator of the intrinsic dimension, several of which were reviewed in [33] and used to provide evidence for the manifold hypothesis. Here we select the Levina-Bickel estimator [27] with the MacKay-Ghahramani correction [29] because it is efficient to compute, and correlates well with synthetic datasets of known dimensionality [33]. The estimator is

$$\hat{d_k} = \left(\frac{1}{n(k-1)} \sum_{i=1}^{n} \sum_{j=1}^{k-1} \log \frac{T_k(x_i)}{T_j(x_i)} \right)^{-1}, \tag{2}$$

where $T_k(x_i)$ is the Euclidean distance between datapoint x_i and its kth nearest neighbour in the dataset $\{x_i\}_{i=1}^n$. The hyperparameter k sets the scale at which the manifold is probed, with smaller values giving a more close up view. In the following we estimate $\hat{d_k}$ on the training data, then use it to specify the generalized autoencoder architecture.

3 Experiments

3.1 Dataset

As a proof of concept, we model the photon dataset provided by the Fast Calorimeter Simulation Challenge 2022 [18]. The training and test datasets, each containing 121,000 electromagnetic calorimeter showers simulated by Geant4, display a cylindrical geometry divided into five layers of bins in polar coordinates, with 368 voxels of deposited energies. The incident photons are parallel to the z-axis of the calorimeter, and have energies ranging from 256 MeV to over 4 TeV, increasing in powers of two, with each level represented by 10,000 samples, except at higher energies where there are fewer. We perform additional preprocessing to the dataset to facilitate training, similar to [25] and [31]. Due to noise, the total deposited energy E_{dep} can be greater than the incident energy E_{inc}, with the largest ratio $E_{\text{dep}}/E_{\text{inc}}$ in the training set as $r_{\text{max}} = 3.1$. Each voxel of each shower is rescaled by $r_{\text{max}} E_{\text{inc}}$ to ensure voxels are in $[0, 1]$. We append E_{dep} as an additional feature, and linearly scale incident energies into the range $[0, 1]$. The training dataset is split 80/20 for validation.

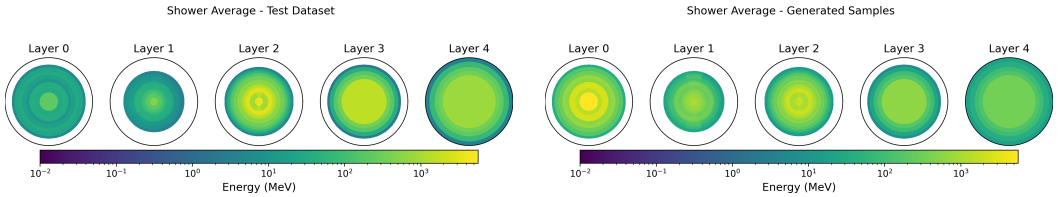

Figure 3: Average deposited energy per voxel for the test dataset (left), and generated samples (right). The samples were conditionally generated using the distribution of energies from the test set.

3.2 Intrinsic Dimension Estimates

We apply the dimension estimator Eq. (2) to the photon training dataset before and after preprocessing, with results shown in Fig. 2. Other works [25, 31] transform voxels to logit space which has a large effect on the estimated dimension since distances between datapoints are stretched. The logit transformation may have had empirical benefits in these prior works in part because it increased the "effective" intrinsic dimension and thereby reduced manifold overfitting in models that had mismatched dimensionality. Our model does not have this issue so we do not use the logit transformation. At very small k the estimator is biased to overestimate [27, 29]. Larger k means more distant neighbours are considered for each point, effectively looking at longer length scales which can smooth out noise but also miss small (compact) dimensions. This intuition explains the decreasing nature of $\hat{d}_k$ with k. Based on the value at $k = 10$ which has provided an accurate estimate in prior work [33, 6], we use the dimension $\hat{d}_{10} = 20$ for our experiments.

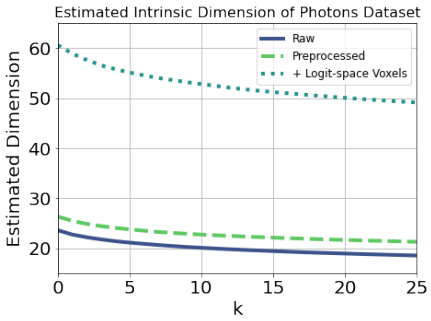

Figure 2: The estimated intrinsic dimension of the photon training dataset as a function of the hyperparameter k.

3.3 Manifold Learning and Density Estimation

We modelled showers with a VAE as our generalized autoencoder and a conditional NF for density estimation on the learned manifold, using the code of Loaiza-Ganem et al. [28]. The VAE's encoder and decoder architectures were multi-layer perceptrons with 3 hidden layers of 512 units. The NF, implemented with the code of Durkan et al. [16], consisted of a 4-layer rational-quadratic neural spline flow [15] and a 3-block residual network [21] at each layer. The output of each residual block was combined with the conditioning input using a gated linear unit [10]. The VAE and NF were each trained for 200 epochs for a total runtime of 110 minutes on a Titan V GPU, requiring only 1 GB of memory. In comparison, CaloScore [31] used 16 A100 GPUs and trained for 5 times as many epochs.

In Fig. 3 we show the average deposited energy per voxel for the test dataset, and for conditional samples from our model using the same incident energy distribution as the test set. The rotational symmetry of layers 1 and 2 was learned well, but our model allocated too much energy to layer 0. Conditioning the generalized autoencoder on incident energies could improve this.

Table 1: Histogram χ^2 separation powers in high-level features. Lower is better. L denotes layer. CE is the center of energy.

FEATURE	χ^2 POWER
$E_{\text{dep}}/E_{\text{inc}}$	0.0535
E_{dep}, L0	0.0540
E_{dep}, L1	0.0304
E_{dep}, L2	0.0243
E_{dep}, L3	0.0045
E_{dep}, L4	0.0009
CE in η, L1	0.0376
CE in η, L2	0.0512
CE in ϕ, L1	0.0145
CE in ϕ, L2	0.0391
Width in η, L1	0.1548
Width in η, L2	0.0538
Width in ϕ, L1	0.1080
Width in ϕ, L2	0.0489

In Table 1 we compare our model's samples to the test set using histograms of high-level features as described in the Challenge [18]. For each feature the χ^2 separation power between histograms is computed, with lower numbers indicating the model matching the true distribution. Deposited energies and the centers of energy are well-learned, with some improvement required for widths. Four corresponding histograms are shown in Fig. 4. Using the architecture and training details provided in [25], we trained a binary classifier to distinguish real and generated showers which achieved only

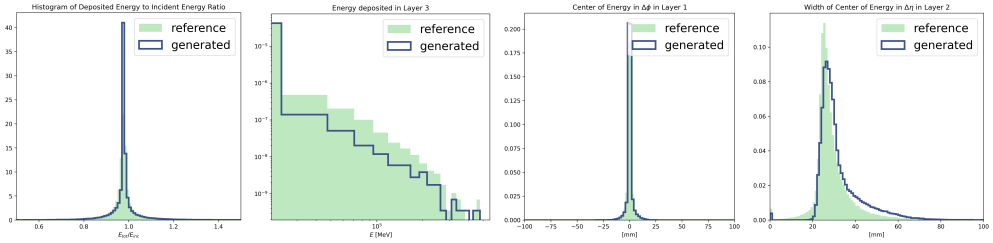

Figure 4: Histograms of high-level features comparing generated samples to the test set.

0.78 AUC. An AUC lower than 1.0 shows that many generated showers are indistinguishable from real ones. By comparison, CaloScore [31] reported 0.98 AUC.

Table 2: Sample generation times

Batch Size	Number of showers	Time per shower (ms)
1,000	1,000	0.0598
1,000	100,000	0.0844
10,000	10,000	0.0265
10,000	100,000	0.0246
50,000	50,000	0.0216
50,000	100,000	0.0201

Our method also demonstrates a significant speedup in generation time compared to others. In Table 2 we show the amortized time to generate each sample for various batch sizes and numbers of samples generated. We include both the runtime of the model, and overhead to undo preprocessing on the samples which is required to produce showers in their original format. For large batch sizes, which still comfortably fit on a single Titan V GPU, our generation time is as low as 0.02 ms per shower. In comparison, CaloScore [31] reported a sampling time of 40 ms per shower on the same dataset, while CaloFlow II's best settings (on a similar dataset) required 0.08 ms per sample [26]. The Geant4 simulation times increase with incident energy, and range from 100 ms to 3 s [1].

4 Discussion

In this work we investigated deep generative models for generating calorimeter showers that learn a low-dimensional manifold structure and estimate densities on it. While the calorimeter data was represented in 368-dimensional space, we found the intrinsic dimension of the shower manifold to be approximately 20, and showed that it is possible to accurately learn the data's density in so few dimensions. Compared to previous approaches, our use of dimension reduction led to lightweight models, significant speedups in training and sample generation, and good performance due to the avoidance of *manifold overfitting* [28]. While this work is preliminary, we expect that further improvements are possible as the methods are scaled up.

For calorimeter shower simulation, past work [32, 25, 1, 31] has focused solely on creating fast and accurate density models, and has achieved these goals to varying extents. However, past models do not inform us about the nature of the manifold supporting the data distribution, which could reveal interesting physics. Furthermore, modelling mismatches could impose fundamental barriers to accurate simulation. Consider the topology of the learned distribution - normalizing flows learn a diffeomorphism that acts on a simple prior distribution and hence will be unable to learn topologically non-trivial data manifolds [9]. In contrast, learning the manifold implicitly [37] can enrich the class of topologies that can be modelled without prior knowledge or additional assumptions. We expect that models which can better incorporate aspects like dimensionality, topology, geometry, compactness, and connectedness will not only perform better, but will provide more insight into the physical processes at play, especially when trained on experimental rather than simulated data.

Broader Impact While generative models certainly have far-reaching societal implications, for instance when used maliciously to generate images of humans [7], or new chemical structures which could be weaponized [42], we do not foresee negative societal outcomes from applying them to calorimeter showers. It should be acknowledged that the utility of surrogate models is limited by the quality of data they are trained on. In this setting, our method relies on accurate simulations of calorimeter showers from Geant4. Our method is not meant to replace physics-based simulation, but complement it by addressing the growing concern of computational cost.

Acknowledgments and Disclosure of Funding

M.L. acknowledges the financial support of the European Research Council (grant SLING 819789).

References

[1] G. Aad, A. Kupco, J. Chan, M. A. Principe Martin, P. A. Delsart, T. Dreyer, Y. Wang, K. Jakobs, A. M. Rodriguez Vera, S. Shaw, et al. AtlFast3: the next generation of fast simulation in ATLAS. Technical report, ATLAS-SIMU-2018-04-003, 2021.

[2] S. Agostinelli et al. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003. ISSN 0168-9002. doi: https://doi.org/10.1016/S0168-9002(03)01368-8.

[3] J. Allison et al. Geant4 developments and applications. *IEEE Transactions on Nuclear Science*, 53(1): 270–278, 2006. doi: 10.1109/TNS.2006.869826.

[4] J. Allison et al. Recent developments in Geant4. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 835:186–225, 2016. ISSN 0168-9002. doi: https://doi.org/10.1016/j.nima.2016.06.125.

[5] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[6] B. C. A. Brown, A. L. Caterini, B. L. Ross, J. C. Cresswell, and G. Loaiza-Ganem. The union of manifolds hypothesis and its implications for deep generative modelling. *arXiv:2207.02862*, 2022.

[7] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, H. Anderson, H. Roff, G. C. Allen, J. Steinhardt, C. Flynn, S. O. hÉigeartaigh, S. Beard, H. Belfield, S. Farquhar, C. Lyle, R. Crootof, O. Evans, M. Page, J. Bryson, R. Yampolskiy, and D. Amodei. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv:1802.07228*, 2018.

[8] A. L. Caterini, G. Loaiza-Ganem, G. Pleiss, and J. P. Cunningham. Rectangular flows for manifold learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[9] R. Cornish, A. Caterini, G. Deligiannidis, and A. Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2133–2143, 2020.

[10] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941. PMLR, 2017.

[11] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *ICLR*, 2017.

[12] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *ICLR*, 2017.

[13] Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32:3608–3618, 2019.

[14] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. *ICLR*, 2017.

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

[16] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. nflows: normalizing flows in PyTorch, Nov. 2020. URL `https://doi.org/10.5281/zenodo.4296287`.

[17] C. W. Fabjan and F. Gianotti. Calorimetry for particle physics. *Rev. Mod. Phys.*, 75:1243–1286, Oct 2003. doi: 10.1103/RevModPhys.75.1243.

[18] M. Faucci Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska. Fast Calorimeter Simulation Challenge 2022, 2022. URL `https://calochallenge.github.io/homepage/`.

[19] M. C. Gemici, D. Rezende, and S. Mohamed. Normalizing Flows on Riemannian Manifolds. *arXiv:1611.02304*, 2016.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[22] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[23] C. Horvat and J.-P. Pfister. Density estimation on low-dimensional manifolds: an inflation-deflation approach. *arXiv preprint arXiv:2105.12152*, 2021.

[24] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. *ICLR*, 2014.

[25] C. Krause and D. Shih. CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows. *arXiv:2106.05285*, 2021.

[26] C. Krause and D. Shih. CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows. *arXiv preprint arXiv:2110.11377*, 2021.

[27] E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[28] G. Loaiza-Ganem, B. L. Ross, J. C. Cresswell, and A. L. Caterini. Diagnosing and Fixing Manifold Overfitting in Deep Generative Models. *Transactions on Machine Learning Research*, 2022.

[29] D. J. MacKay and Z. Ghahramani. Comments on 'Maximum Likelihood Estimation of Intrinsic Dimension' by E. Levina and P. Bickel (2004). *The Inference Group Website, Cavendish Laboratory, Cambridge University*, 2005.

[30] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial Variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *International Conference on Machine Learning*, pages 2391–2400. PMLR, 2017.

[31] V. Mikuni and B. Nachman. Score-based Generative Models for Calorimeter Shower Simulation. *arXiv preprint arXiv:2206.11898*, 2022.

[32] 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, Jan 2018. doi: 10.1103/PhysRevD.97.014021. URL `https://link.aps.org/doi/10.1103/PhysRevD.97.014021`.

[33] P. Pope, C. Zhu, A. Abdelkader, M. Goldblum, and T. Goldstein. The Intrinsic Dimension of Images and Its Impact on Learning. In *International Conference on Learning Representations*, 2021.

[34] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[36] B. L. Ross and J. C. Cresswell. Tractable Density Estimation on Learned Manifolds with Conformal Embedding Flows. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[37] B. L. Ross, G. Loaiza-Ganem, A. L. Caterini, and J. C. Cresswell. Neural implicit manifold learning for topology-aware generative modelling. *ICML 2022 Workshop on Topology, Algebra, and Geometry in Machine Learning*, 2022.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[39] Y. Song and S. Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[40] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

[41] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. *ICLR*, 2018.

[42] F. Urbina, F. Lentzos, C. Invernizzi, and S. Ekins. Dual use of artificial-intelligence-powered drug discovery. *Nature Machine Intelligence*, 4(3):189–191, 2022.

[43] B. Uria, I. Murray, and H. Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *In Advances in Neural Information Processing Systems 26 (NIPS 26)*, 2013.

[44] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7): 1661–1674, 2011.

[45] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] Data is available under the Creative Commons CC0 waiver and we discussed and cited.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] Data is from a public challenge.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]