Generating Calorimeter Showers as Point Clouds

Simon Schnake
Deutsches Elektronen-Synchrotron DESY
RWTH Aachen
simon.schnake@desy.de

Dirk Krücker
Deutsches Elektronen-Synchrotron DESY
dirk.kruecker@desy.de

Kerstin Borras
Deutsches Elektronen-Synchrotron DESY
RWTH Aachen
kerstin.borras@desy.de

Abstract

In particle physics, precise simulations are necessary to enable scientific progress. However, accurate simulations of the interaction processes in calorimeters are complex and computationally very expensive, demanding a large fraction of the available computing resources in particle physics at present. Various generative models have been proposed to reduce this computational cost. Usually, these models interpret calorimeter showers as 3D images in which each active cell of the detector is represented as a voxel. This approach becomes difficult for high-granularity calorimeters due to the larger sparsity of the data. In this study, we use this sparseness to our advantage and interpret the calorimeter showers as point clouds. More precisely, we consider each hit as part of a hit distribution depending on a global latent calorimeter shower distribution. A first model to learn calorimeter showers as point clouds is presented. The model is evaluated on a high granular calorimeter dataset.

1 Introduction

Particle physics requires precise simulations to achieve scientific progress. More than half of the computational resources of the Large Hadron Collider (LHC) computing grid are already used for simulation [1]. In the future High Luminosity Phase of the LHC, about 100 times as many simulated events will be needed [2]. The simulation of calorimeters is typically one of the most computationally intensive parts of a detector simulation, and there is an ongoing effort to reduce this computational cost using machine learning.

A high-energetic particle interacting with the material of a calorimeter typically creates a cascade of secondary particles, which in turn interact with the material of the calorimeter, creating an avalanche of lower-energy particles. Modern calorimeters often consist of a sandwich of passive absorber material and active high granular sensors which record the energy deposit (the so-called hits).

In previous attempts at generative modelling of calorimeter showers with Deep Learning, the energy deposits were considered as a 3D grid of voxels and the individual shower as a 3D image. Most models[3–11] used to study the artificial generation of calorimeter data, were *Generative Adversarial Networks* (GANs)[12]. GANs can have relatively flexible architectures and are fast generators, but they suffer from several shortcomings: incomplete distribution coverage (*mode collapse*) and convergence is often difficult to achieve. Recent developments extend the plain GAN approach and combinations of GANs and Autoencoders [13, 14] are used. A completely different approach are Normalizing Flows[15] which allow likelihood-based training also on calorimeter data [16, 17]. This

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

lead to high quality results. One drawback of the CaloFlow [16] approach and normalizing flows in general is that they do not scale well to large dimensions and therefore are not directly applicable for simulating high granular detectors with many active cells, such as the future CMS forward calorimeter HGCal [18] or the proposed CALICE calorimeter [19]. To scale likelihood-training to high dimensions, *Mikuni et. al.*[20] proposed to use score-based diffusion models instead of normalizing flows[15]. We take a different route of scaling likelihood-base training. All previous mentioned models are trained on voxel-based data. This has multiple disadvantages. First, a high granular calorimeter most cells have no entries, therefore the data is sparse. Second, an irregularly shaped calorimeter is often not directly transferable to voxels. The architecture of the model has to be developed directly for the calorimeter. So one model is not directly applicable for another calorimeter dataset. This leads to worse comparison possibilities of the different models. We interpret the hits of the showers as point clouds. This resolves the sparsity of the data, since empty cells are dropped. Also, the geometry of the calorimeter is transferable, and the model can be used for different datasets without altering the model structure.

2 Model

The model here described is based on *PointFlow* [21]. The target of the PointFlow paper was to model the surface of objects as point clouds. PointFlow used continuous normalizing flows. In contrast, *Klokov et al.*[22] updated the model with discrete normalizing flows. This leads to faster training and inference.

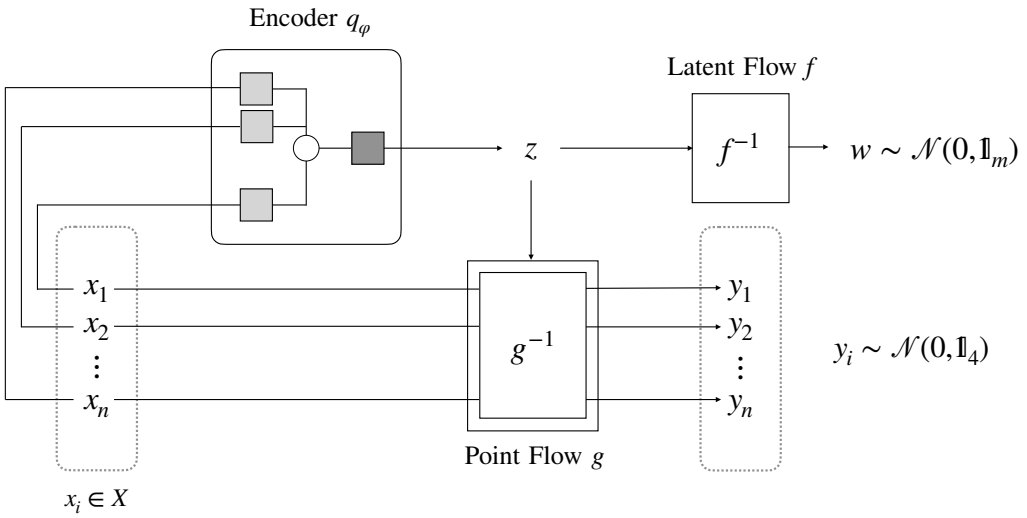

Figure 1: A schematic of the model.

Like both models, our model consists of three sub-models, as shown in Fig. 1. The permutation invariant encoder q_φ, which maps the entire point cloud X into the latent representation z. Our encoder design is based on Particle Flow Networks [23]. The z-representation is enriched with the number of all hits n_{hits} and the transformed total energy of the hits E_{sum}. The other two models are both conditional normalizing flows, based on rational quadratic spline (RQS) coupling layers[24] and ResNet[25] conditioners. To keep the latent space more flexible, z is transformed by the *Latent Flow*, similar to the *Variational Lossy Autoencoder*[26]. The flow is conditioned on the energy of the incoming electron E_{in}. The *Point Flow* transforms each point x_i separately, it is conditioned on the latent variable z, and therefore models the distribution of the points x_i conditioned on the distribution of z.

The model is trained by minimizing the function

$$\mathcal{L} = \underbrace{-\mathbb{E}_{q_\varphi(z|X)}\left[\sum_i^{n_{\text{hits}}} \ln \mathcal{N}(g^{-1}(x_i, z); 0, \mathbb{1}) + \ln\left|\det \frac{\mathrm{d}g^{-1}(x_i, z)}{\mathrm{d}z}\right|\right]}_{\mathcal{L}_{\text{recon}}}$$
$$\underbrace{-\mathbb{E}_{q_\varphi(z|X)}\left[\ln \mathcal{N}(f^{-1}(z, E_{\text{in}}); 0, \mathbb{1}) + \ln\left|\det \frac{\mathrm{d}f^{-1}(z, E_{\text{in}})}{\mathrm{d}z}\right|\right]}_{\mathcal{L}_{\text{prior}}}$$
$$\underbrace{+\left(-\frac{d}{2}(1 + \ln(2\pi)) - \sum_{i=1}^{d} \ln \sigma_i\right)}_{\mathcal{L}_{\text{entr}}} \tag{1}$$

with the ADAM[27] optimizer. The loss function at hand is based on the loss function of a variational autoencoder (ELBO). Since normalizing flows are invertible, $\mathcal{L}_{\text{recon}}$ is not trained on reconstructing the point clouds. Instead, the model is trained to maximize the likelihood of the transformed data to a Gaussian. With $\mathcal{L}_{\text{prior}}$ the model learns a transformation from a Gaussian to the latent space z. The last part is the entropy of the decoder and acts as a regularization term on the z space. An exact derivation of the loss function 1 can be found in the appendix A.

All shower hits are generated in parallel and independently of each other. The model cannot take into account whether a calorimeter cell has already been taken by another generated shower hit. Therefore we need a special sampling method to get consistent showers.

For shower generation, we want to sample from the probability density of the possible showers. Therefore, we need the probability that a calorimeter cell was hit and the probability density of the possible energies in the hit cell. Both quantities are not directly accessible. We approximate these probabilities with the Point Flow. We sample a relatively large number of points (10k) with their corresponding probabilities. The average probability of all points in one cell is taken as the probability of hitting that cell. We sample n_{hits} cells without replacement and with their probability. For each sampled cell, we then pick an energy value according to all energy values sampled for the hit cell and their probability. Therefore, we use the model as a Monte Carlo approximator of the showers.

3 Experimental Setup

The CLIC calorimeter data set[10, 28] is used to test the model. In the data, the showering of electrons entering a section of the CLIC detector were simulated in GEANT4[29]. The energy of the incoming electrons is between 10 – 510 GeV. The ECAL part of the simulation is used. The resulting calorimeter images have a dimensionality of $(51 \times 51 \times 25)$.

The voxel dataset is transformed into a point cloud dataset. All voxels with energy input are considered. The 3 position coordinates are uniformly distributed over the voxel space. So that the resulting distribution fills a unit box. This processing is referred to as *dequantization*[30]. Since the energy inputs are greater than zero and decrease exponentially, the logarithm of the energy is used. The resulting 4D point clouds were then normalized.

$$p_i = \text{norm}((x_i, y_i, z_i, \ln(E_i)) \tag{2}$$

The model is trained using four parallel NVIDIA V100 GPUs for a total time of 24 hours. The full source code for the study alongside the hyperparameter settings is available at `https://github.com/simonschnake/CaloPointFlow`.

4 Results and Discussion

For generative models to replace simulations, their outputs must be consistent. To validate this, we look at different distributions of the generated data.

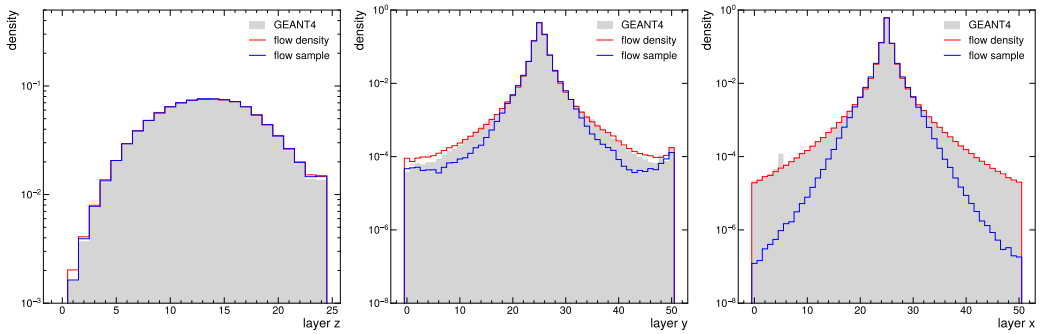

Figure 2: the average shower profiles in all three directions are shown. In gray the one of GEANT4. In red the direct density of the flow. In blue the result of the sampling.

Fig. 2 shows the average shower profile in all directions. The results of the direct density of the model and the results after sampling from the model are compared with the simulation data. It can be seen that the model produces matching results, but the tails of the distributions are not well represented by the sampling.

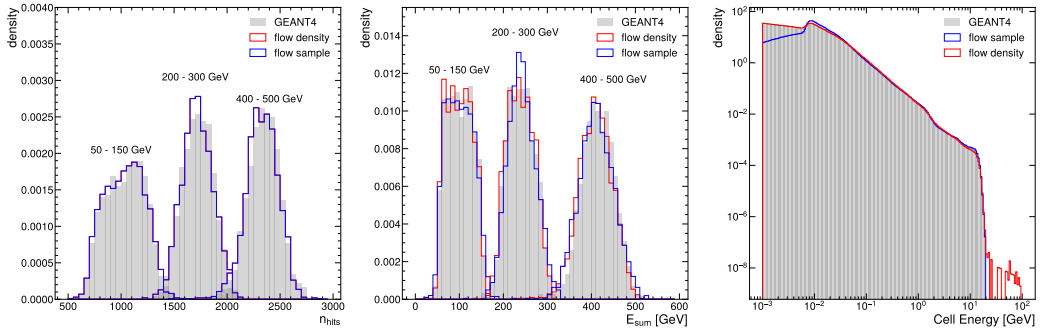

Figure 3: Three statistics of the data are shown. On the left, the number of cells with energy in three energy regimes. In the middle, the summed energy in the cells in the same regimes. On the right, the distribution of energy inputs in the cells. The color scheme is consistent with Fig. 2.

Fig. 3 compares three statistics of the showers. The first two show that both the number of cells hit and the sum of the energy in the cells agree well with the results of GEANT4 - both for the direct density of the model and for the sampled results. The right graph shows that there are significantly fewer low-energy hits after sampling. This is consistent with the decrease in the tails in Fig. 2.

5 Conclusion and Outlook

The results of the model appear promising. Except for the tails, the model generates showers of a high quality. A possible further development to get the problems of the model at the tails under control would be the use of a post-processing network, as shown in [13, 14]. We are currently investigating the model's performance on other datasets. This will be part of another publication.

Overall, the model shows good results and can overcome the problems of voxel based models.

6 Broader Impact

Generative models can have negative social impact, by creating *deepfakes*. However, our work is dedicated to the application in particle physics does not cause any capabilities beyond those already reported. Within particle physics, the use of generative models could cause experiments to give false results. This is a veritable danger and before employing it, the properties of the generated data should

be studied intensively. The application of the model could lead to better use of computer resources and thus could reduce CO_2 emissions. Likewise, it can lead to a decrease in research costs and allows for more hypotheses to be tested, thus supporting scientific progress. The positive consequences clearly outweigh by the negative ones.

7 Acknowledgement

This research was funded by Deutsches Elektronen-Synchrotron DESY, a member of the Helmholtz Association (HGF). This research was supported through the Maxwell computational resources operated at DESY.

References

1. Albrecht, J. *et al.* A Roadmap for HEP Software and Computing R&D for the 2020s. *Computing and Software for Big Science* **3,** 7. https://doi.org/10.1007/s41781-018-0018-8 (2019).
2. Aberle, O. *et al. High-Luminosity Large Hadron Collider (HL-LHC): Technical design report* https://cds.cern.ch/record/2749422 (CERN, Geneva, 2020).
3. De Oliveira, L., Paganini, M. & Nachman, B. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. *Computing and Software for Big Science* **1.** https://doi.org/10.1007%2Fs41781-017-0004-6 (Sept. 2017).
4. Paganini, M., de Oliveira, L. & Nachman, B. Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters. *Physical Review Letters* **120.** https://doi.org/10.1103%2Fphysrevlett.120.042003 (Jan. 2018).
5. Paganini, M., de Oliveira, L. & Nachman, B. CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Physical Review D* **97.** https://doi.org/10.1103%2Fphysrevd.97.014021 (Jan. 2018).
6. Vallecorsa, S. Generative models for fast simulation. *Journal of Physics: Conference Series* **1085,** 022005. https://doi.org/10.1088/1742-6596/1085/2/022005 (Sept. 2018).
7. Chekalina, V. *et al.* Generative Models for Fast Calorimeter Simulation: the LHCb case&gt. *EPJ Web of Conferences* **214** (eds Forti, A., Betev, L., Litmaath, M., Smirnova, O. & Hristov, P.) 02034. https://doi.org/10.1051%2Fepjconf%2F201921402034 (2019).
8. Erdmann, M., Glombitza, J. & Quast, T. Precise Simulation of Electromagnetic Calorimeter Showers Using a Wasserstein Generative Adversarial Network. *Computing and Software for Big Science* **3,** 4. https://doi.org/10.1007/s41781-018-0019-7 (2019).
9. Khattak, G. R., Vallecorsa, S., Carminati, F. & Khan, G. M. *Fast Simulation of a High Granularity Calorimeter by Generative Adversarial Networks* 2021. https://arxiv.org/abs/2109.07388.
10. Belayneh, D. *et al.* Calorimetry with deep learning: particle simulation and reconstruction for collider physics. *The European Physical Journal C* **80.** https://doi.org/10.1140%2Fepjc%2Fs10052-020-8251-9 (July 2020).
11. Rehm., F. *et al. Reduced Precision Strategies for Deep Learning: A High Energy Physics Generative Adversarial Network Use Case* in *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods - ICPRAM,* (SciTePress, 2021), 251–258. ISBN: 978-989-758-486-2.
12. Goodfellow, I. J. *et al. Generative Adversarial Networks* 2014. https://arxiv.org/abs/1406.2661.
13. Buhmann, E. *et al.* Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed. *Computing and Software for Big Science* **5.** https://doi.org/10.1007%2Fs41781-021-00056-0 (May 2021).
14. Buhmann, E. *et al. Hadrons, Better, Faster, Stronger* 2021. https://arxiv.org/abs/2112.09709.
15. Rezende, D. J. & Mohamed, S. *Variational Inference with Normalizing Flows* 2015. https://arxiv.org/abs/1505.05770.
16. Krause, C. & Shih, D. *CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows* 2021. https://arxiv.org/abs/2106.05285.

17. Krause, C. & Shih, D. *CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows* 2021. https://arxiv.org/abs/2110.11377.
18. *The Phase-2 Upgrade of the CMS Endcap Calorimeter* tech. rep. (CERN, Geneva, 2017). https://cds.cern.ch/record/2293646.
19. CALICE Collaboration *et al. Hadronic Energy Resolution of a Combined High Granularity Scintillator Calorimeter System* 2018. https://arxiv.org/abs/1809.03909.
20. Mikuni, V. & Nachman, B. *Score-based Generative Models for Calorimeter Shower Simulation* 2022. https://arxiv.org/abs/2206.11898.
21. Yang, G. *et al. PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows* 2019. https://arxiv.org/abs/1906.12320.
22. Klokov, R., Boyer, E. & Verbeek, J. *Discrete Point Flow Networks for Efficient Point Cloud Generation* 2020. https://arxiv.org/abs/2007.10170.
23. Komiske, P. T., Metodiev, E. M. & Thaler, J. Energy flow networks: deep sets for particle jets. *Journal of High Energy Physics* **2019.** https://doi.org/10.1007%2Fjhep01%282019%29121 (Jan. 2019).
24. Durkan, C., Bekasov, A., Murray, I. & Papamakarios, G. *Neural Spline Flows* 2019. https://arxiv.org/abs/1906.04032.
25. He, K., Zhang, X., Ren, S. & Sun, J. *Deep Residual Learning for Image Recognition* 2015. https://arxiv.org/abs/1512.03385.
26. Chen, X. *et al. Variational Lossy Autoencoder* 2016. https://arxiv.org/abs/1611.02731.
27. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* 2014. https://arxiv.org/abs/1412.6980.
28. Pierini, M. & Zhang, M. *CLIC Calorimeter 3D images: Electron showers at Fixed Angle* (Zenodo, Jan. 2020). https://doi.org/10.5281/zenodo.3603122.
29. Agostinelli, S. *et al.* GEANT4–a simulation toolkit. *Nucl. Instrum. Meth. A* **506,** 250–303 (2003).
30. Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S. & Lakshminarayanan, B. Normalizing Flows for Probabilistic Modeling and Inference. https://arxiv.org/abs/1912.02762 (2019).
31. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* 2013. https://arxiv.org/abs/1312.6114.

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] A working point cloud model was shown.
   (b) Did you describe the limitations of your work? [Yes] The limitation in modelling the tails of the distribution are shown in Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 6
   (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)? [Yes] a link will be included in the unblind version
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] can be found in the repository
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] This will be explored in the future.

(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] See Section 3

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] The reference to of the CLIC dataset is listed.
   (b) Did you mention the license of the assets? [No] Can be found on the website.
   (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? [N/A]
   (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]

A Derivation of the Loss function

The model is co-trained like a variational autoencoder[31] in which the Evidence Lower Bound (ELBO) is maximized.

The model should model the probability density $p(X)$ of the possible showers X. We construct the model in the framework of a variational autoencoder [31]. The encoder $q_\varphi(z|X)$ and decoder $p_\theta(X|z)$ over the probability distribution of the latent variable $p(z)$ can be trained simultaneously by maximizing the Evidence Lower Bound (ELBO) on the log-likelihood $\ln P(X)$. It is equivalent to minimize the negative ELBO

$$\mathcal{L} = -\mathbb{E}_{q_\varphi(z|X)}[\ln p_\theta(X|z)] + D_{KL}(q_\varphi(z|X)||p(z)) \tag{3}$$

$$= -\underbrace{\mathbb{E}_{q_\varphi(z|X)}\left[\ln p_\theta(X|z)\right]}_{\mathcal{L}_{\text{recon}}} - \underbrace{\mathbb{E}_{q_\varphi(z|X)}\left[\ln p_\theta(z)\right]}_{\mathcal{L}_{\text{prior}}} - \underbrace{\mathbb{E}_{q_\varphi(z|X)}\left[-\ln q_\varphi(z|X)\right]}_{\mathcal{L}_{\text{entr}}}. \tag{4}$$

The loss function consists of three parts. The first part $\mathcal{L}_{\text{recon}}$ is the reconstruction error of the shower X. Minimizing $\mathcal{L}_{\text{recon}}$ is equal to generating hits of the shower with a high likelihood. The second part $\mathcal{L}_{\text{prior}}$ is the expectation value of the prior distribution under the approximated distribution of the encoder. Minimizing $\mathcal{L}_{\text{prior}}$ is equal to generating points of z latent space with a high probability. The last term $\mathcal{L}_{\text{entr}}$ is the entropy of the encoded values. It is a regularization on the latent space z.

Here the encoder $q_\varphi(z|x)$ approximates z conditioned on X. To let a sampled value of the auto-encoder differentiable, we use the reparametrization trick $z = \mu + \epsilon \odot \sigma$, where μ and σ are the outputs of the encoder. ϵ is a normal distributed random value.

The expected value $\mathbb{E}_{q_\varphi(z|X)}[\ln q_\varphi(z|X)]$ is the entropy $\mathcal{H}(q_\varphi(z|X))$. Since we approximate the prior $p_\theta(X)$ as locally Gaussian, the entropy is given by

$$\mathcal{L}_{\text{entr}} = \mathcal{H}(q_\varphi(z|X)) = \frac{d}{2}(1 + \ln(2\pi)) + \sum_{i=1}^{d} \ln \sigma_i. \tag{5}$$

To make the latent distribution $p(z)$ more flexible, the latent z space is constructed as the transformation of a Gaussian by a normalizing flow f. Therefore, the expectation value of the probability $p(z)$ can be written as

$$\mathbb{E}_{q_\varphi(z|X)}[\ln p(z)] = \mathbb{E}_{q_\varphi(z|X)}\left[\ln \mathcal{N}(f^{-1}(z, E_{\text{in}}); 0, \mathbb{1}) + \ln \left|\det \frac{\mathrm{d}f^{-1}(z, E_{\text{in}})}{\mathrm{d}z}\right|\right]. \tag{6}$$

Here the flow f is also conditioned on the energy of the incoming electron E_{in}.

The conditional probability density of the entire shower can be separated into the ones of the individual shower hits

$$\ln p_\theta(X|z) = \ln \prod_{i=1}^{n_{\text{hits}}} p_\theta(x_i|z) = \sum_{i=1}^{n_{\text{hits}}} \ln p_\theta(x_i|z). \tag{7}$$

The conditional distribution of the individual hits $p_\theta(x_i|z)$ is approximated by another normalizing flow

$$\ln p_\theta(x_i|z) = \ln \mathcal{N}(g^{-1}(x_i, z); 0, \mathbb{1}) + \ln \left|\det \frac{\mathrm{d}g^{-1}(x_i, z)}{\mathrm{d}z}\right|. \tag{8}$$

Inserting all parts, the loss function is given by

$$\begin{aligned}
\mathcal{L} = &\underbrace{-\mathbb{E}_{q_\varphi(z|X)}\left[\sum_i^{n_{\text{hits}}} \ln \mathcal{N}(g^{-1}(x_i, z); 0, \mathbb{1}) + \ln \left|\det \frac{\mathrm{d}g^{-1}(x_i, z)}{\mathrm{d}z}\right|\right]}_{\mathcal{L}_{\text{recon}}} \\
&\underbrace{-\mathbb{E}_{q_\varphi(z|X)}\left[\ln \mathcal{N}(f^{-1}(z, E_{\text{in}}); 0, \mathbb{1}) + \ln \left|\det \frac{\mathrm{d}f^{-1}(z, E_{\text{in}})}{\mathrm{d}z}\right|\right]}_{\mathcal{L}_{\text{prior}}} \\
&\underbrace{+\left(-\frac{d}{2}(1 + \ln(2\pi)) - \sum_{i=1}^{d} \ln \sigma_i\right)}_{\mathcal{L}_{\text{entr}}}.
\end{aligned} \tag{9}$$