



Soft Margin Spectral Normalization for GANs

Alexander Rogachev¹ · Fedor Ratnikov¹

Received: 6 December 2023 / Accepted: 4 June 2024
© The Author(s) 2024

Abstract

In this paper, we explore the use of Generative Adversarial Networks (GANs) to speed up the simulation process while ensuring that the generated results are consistent in terms of physics metrics. Our main focus is the application of spectral normalization for GANs to generate electromagnetic calorimeter (ECAL) response data, which is a crucial component of the LHCb. We propose an approach that allows to balance between model's capacity and stability during training procedure, compare it with previously published ones and study the relationship between proposed method's hyperparameters and quality of generated objects. We show that the tuning of normalization method's hyperparameters boosts the quality of generative model.

Keywords Generative models · High energy physics · Deep learning · Regularization

Introduction

The Large Hadron Collider (LHC), built by the European Organization for Nuclear Research (CERN), is the world's largest collider. The LHCb experiment [1] at LHC focuses on studies of the heavy flavor physics, precise measurements of CP violation, and other effects in and beyond the Standard Model. The LHCb detector consists of several components, including an electromagnetic calorimeter (ECAL). The ability to simulate the expected detector response is a vital requirement for the physics analysis of the collected data and extracting physics results. The use of the Geant4 package [2] for simulating detector responses is computationally expensive and resource-intensive, requiring to speed this process up.

Previously, it was shown [3] that GAN-based frameworks have the potential to serve as fast generative models to speed up the simulation. The auxiliary regression extension [4], that introduces physics metrics to a Discriminator part of the model, allowing it to detect objects with poorly reproduced properties, demonstrated improvements of the physics

quality for generated objects. We proposed to train the model in a multitask manner using two objectives: adversarial and regressive ones. However, we used spectral normalization to achieve training stability, and resulted in a reduction in model's capacity, thus it was complicated for the model to solve multiple tasks at the same moment.

In this paper, we study the relationship between model's Lipschitz constant and quality of generated objects. We propose a regularization technique that allows to balance between model's capacity and stability. We compare the quality achieved using different regularization techniques, evaluating the generated objects in terms of both general and physics metrics, including shower asymmetry, shower width, and sparsity level, and study the relationship between the learning rate, method's hyperparameters and quality of generated objects.

GANs in High Energy Physics

Generative adversarial networks (GANs) approach is a prominent technique for developing generative models. GAN framework comprises two components, the generator and discriminator, which compete against each other to recreate objects from a given distribution [5]. The generator (G) learns to map an easy-to-sample distribution, such as a standard normal distribution, to the target distribution. The discriminator (D) provides feedback to improve the

✉ Alexander Rogachev
airogachev@hse.ru

Fedor Ratnikov
fedor.ratnikov@cern.ch

¹ Laboratory of Methods for Big Data Analysis, HSE University, 11 Pokrovsky Bulvar, Moscow, Russia

generator by detecting discrepancies between real reference objects and ones generated by the model. Mathematically, this can be represented as a mini-max game:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim N(0,1)} [\log(1 - D(G(z)))], \quad (1)$$

where p_{data} is a true data distribution, $D(x)$ is the output of the discriminator.

GAN can be extended to a conditional model (CGAN) if both the generator and discriminator are conditioned on additional information y such as class labels or some property of objects we want to generate:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} [\log D(x|y)] + \mathbb{E}_{z \sim N(0,1)} [\log(1 - D(G(z|y)))]. \quad (2)$$

The idea of utilizing GANs for simulation in high energy physics was introduced by Paganini et al. [6] and subsequently developed in [7–10] with the implementation of Wasserstein conditional GAN, [11] investigated the optimization of GANs with hyperparameter scans, [12] applied three-dimensional convolutional layers. Comparisons were made between GAN performance and models based on conditional variational autoencoders, and their combination, with CGAN yielding the best results [13].

Through the Fast Calorimeter Simulation Challenge 2022 [14], the research community organized competition, novel GAN-based [15] methods were introduced, as well as flows [16–18] and diffusion-based ones [19–21].

Currently, diffusion models show state-of-the-art results in terms of quality of generated objects. In this work, we focus on GANs as they accelerate the simulation the most, providing appropriate quality, and we focus on the stability of the training process.

In our prior work [3], we proposed enhancing the generative model's quality by adding self-attention layers to the previously best-performing architecture. This approach allowed convolutional neural networks to capture and employ long-range relationships between image regions throughout the training process. Spectral normalization [22] was also added into both models to stabilize the training process.

Lately in [4] we trained GAN with additional regressors that evaluated object's physics properties simultaneously with the adversarial objective. This setting led to better reproduced metrics that were introduced to the regressor.

Spectral Normalization for GANs

One of the key challenges in the training of GANs is the performance control of the discriminator. This procedure is highly unstable due to step-by-step two-models training

setting, therefore, the density ratio estimation done by the discriminator may be inaccurate, generator networks fail to learn the multimodal structure of the target distribution, gradients may vanish and explode [23]. These facts motivated researches to introduce different forms of restrictions and regularization to the choice of the discriminator.

Spectral Normalization introduced in [22] outperforms other techniques, such as weight normalization [23], weight clipping [24], and gradient penalty [25] as the Lipschitz constant is the only hyperparameter to be tuned, and the power iteration method [26], used under the hood of this approach, is relatively fast.

A neural network f_θ with parameter θ is called *Lipschitz continuous* if there exist a constant $c \geq 0$ such that:

$$\|f_\theta(t_0) - f_\theta(t_1)\|_p \leq c \|t_0 - t_1\|_p, \quad (3)$$

for all possible inputs t_0, t_1 under a p -norm of choice. The parameter c is called the *Lipschitz constant*. Intuitively, this constant c bounds the rate of change of f_θ .

Spectral normalization controls the Lipschitz constant of the discriminator function D by constraining the spectral norm of each layer $g(\mathbf{h}) = W\mathbf{h}$. By definition, Lipschitz norm $\|g\|_{\text{Lip}}$ is equal to $\sup_{\mathbf{h}} \sigma(\nabla_{\mathbf{h}} g(\mathbf{h}))$, where $\sigma(A)$ is the spectral norm of the matrix A (L_2 matrix norm of A)

$$\sigma(A) := \max_{\mathbf{h}: \|\mathbf{h}\|_2=1} \|\mathbf{A}\mathbf{h}\|_2 = \max_{\|\mathbf{h}\|_2 \leq 1} \|\mathbf{A}\mathbf{h}\|_2, \quad (4)$$

which is equivalent to the largest singular value of A . Thus, for a given layer $g(\mathbf{h}) = W\mathbf{h}$, the norm is given by $\|g\|_{\text{Lip}} = \sup_{\mathbf{h}} \sigma(\nabla_{\mathbf{h}} g(\mathbf{h})) = \sup_{\mathbf{h}} \sigma(W) = \sigma(W)$.

It was shown in [22] that if we normalize each W^l using spectral normalization, we can achieve the fact that $\|f\|_{\text{Lip}}$ is bounded from above by 1. Thus, the vanilla spectral normalization requires no hyperparameters to be tuned, however we have no ability to control the degree of regularization.

Lipschitz Networks

Reduction of the Capacity Background

The expressivity reduction issue introduced by the normalization was faced on our own during research, presented in [4]. We suggested training GAN in a multitask manner, adding regressors that evaluate some predefined object's properties. These regressors share layers and weights with the Discriminator, and we use mean squared error as a loss function for this part of the network, therefore the new objective function has the following form:

$$\mathcal{L}(\theta) = \mathcal{L}_{adv}(\theta) + \sum_{k=1}^K \frac{\alpha_k}{N} \sum_{i=1}^N (o_i - \tilde{o}_i)^2, \tag{5}$$

where $\mathcal{L}_{adv}(\theta)$ is the adversarial part of the Discriminator’s loss, K is the number of the object’s properties, that we evaluate via regressors, α_k is the weight of k -th regression loss, N is the number of objects, o_i is the real property value and \tilde{o}_i is the predicted value.

We expected the discriminator to focus on some specific features of the objects, that pass through, forcing the generator to reproduce them better. As the model now has to optimize multiple loss functions and solve additional tasks, our model is required to be more expressive. As traverse cluster asymmetry was reproduced the worst, we set its values as targets for the regressive part of the model.

Without spectral normalization, regression part performed better: this model achieved a lower mean squared error, the second term in 5, nevertheless generative part was highly unstable. However, if we add regression part to the model with layers that were normalized, we do not improve the quality of generated samples as we expected, as the regression task is not solved with an appropriate quality.

These results led us to the need of controlling the L-constant of the discriminator, balancing between training stability and model’s capacity.

Regularization Methods

Through the application of spectral normalization it is possible to stabilize the training procedure, however it directly affects the expressivity of the network. Intuitively, it would be useful to have an ability to control the degree of regularization, balancing between stability and network’s expressive power.

The general idea here is to design a regularization based on the architecture of the k -Lipschitz networks as it was described in [27], constraining all the layers to be 1-Lipschitz and multiplying the final layer with a constant k to make it k -Lipschitz. Making it trainable allows us to treat k as a regularization term (Alpha- k), thus the augmented loss function \mathcal{J} with an original loss function $\mathcal{L}(\theta)$ and its weight α looks as follows:

$$\mathcal{J}(\theta, k) = \mathcal{L}(\theta) + \alpha k. \tag{6}$$

A Lipschitz-like regularization may be redefined as the summation of squared Lipschitz bounds of each layer (L_2), so generalized formulation from [26] looks as follows:

$$\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \sum_{i=1}^l \|W_i\|_p^2. \tag{7}$$

It was reported, that such formulation fails to capture the exponential growth of the Lipschitz constant with respect to the network depth [28], so changing the architecture may require a new α .

Another approach, mentioned in [28], is to define the Lipschitz regularization directly on the weight matrices (L_{inf}):

$$\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \prod_{i=1}^l \|W_i\|_\infty. \tag{8}$$

Mentioned approaches (6), (7) and (8) do not allow to explicitly set any value as a desired Lipschitz constant of the network. Even though otherwise it requires tuning the hyperparameters, some settings require foreknown constants, e.g., [24] requires 1-Lipschitz critic. Using [27] may lead to constant, that is less than 1, affecting the network’s capacity and gradients used to train the generator, as the only way of controlling the regularization is the weight α . It would be intuitive to include this constant into the training objective in such a way that the network remains unaffected as long as the constant is below the target value, so we suggest defining the augmented loss function using *margin value* m and hinge-loss-style form:

$$\mathcal{J}(\theta) = \mathcal{L}(\theta) + \alpha \max \left(0, \prod_{i=1}^l \|W_i\|_2 - m \right). \tag{9}$$

Learning Rate and L-constant

As our approach requires us to set two hyperparameters, we study the relationship between them. In [29] it is shown that in order for the model with Lipschitz constant L to converge learning rate γ should meet the following requirements:

$$0 \leq \gamma \leq \frac{2}{L}. \tag{10}$$

In (6), (7) and (8) we can not set the L value explicitly, but we can play around with the margin value m in (9) and compare the quality of generated objects. For every particular constant we train the model 5 times, using the same pool of learning rates, that remain constant during the fitting procedure. The main objective here is to find a value that allows our model to start to converge, however, we also tried to find a learning rate providing the fastest quality boosting, as it is possible to just set a relatively low learning rate that requires more iterations.

We define quality gain QG as the mean difference of quality of generated objects between adjacent epochs:

$$QG = \frac{1}{N_e} \sum_{i=2}^{N_e} (PRD_i - PRD_{i-1}), \quad (11)$$

where PRD_i is Physical PRD evaluated after the i -th epoch, N_e is the number of epochs, used to evaluate the gain. As we want to find the starting point for the model to converge and evaluate the speed of quality gain, we use only 30 first epochs to calculate the gain. We plot quality gains versus learning rate (Figs. 4, 5) to find the best γ .

ECAL Response Generation

Dataset

The dataset utilized in our experiments comprises information concerning electron interactions within the electromagnetic calorimeter (ECAL). The ECAL employs the

"shashlik" technology, which consists of alternating layers of lead and scintillation plates. The readout cells within different modules are of varying sizes, with dimensions of 4×4 , 6×6 , and $12 \times 12 \text{ cm}^2$, enabling the aggregation of responses of $2 \times 2 \text{ cm}^2$ logical cells to obtain a response for all granularities. All events in the dataset correspond to electrons with a specific momentum and direction entering the calorimeter at a given location, resulting in the generation of an electromagnetic shower in the ECAL. The sum of all energies deposited in the scintillator layers of a single cell produces a matrix of energies corresponding to the ECAL response for the impacting electron, as depicted in Fig. 1. The dataset of such ECAL responses comprises a 30×30 cell matrix of $2 \times 2 \text{ cm}^2$ cells approximately centered on the energy cluster's location, produced using the GEANT4 v10.4 package. Figure 1 shows a sample of these matrices.

We use 60,000 events for training and other 60,000 events as a test dataset. We apply logarithmic transformation

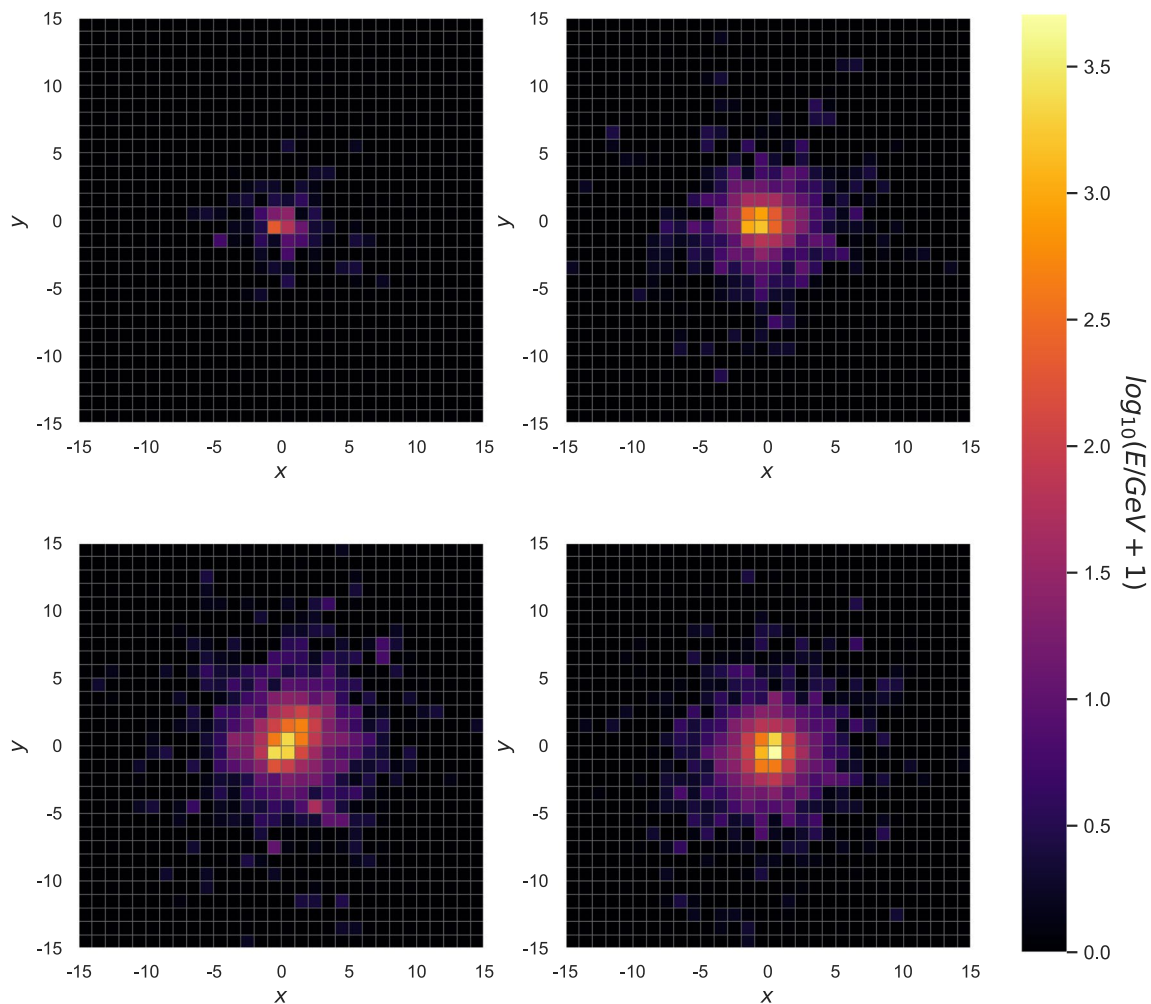


Fig. 1 Sample of energy deposition matrices from the ECAL response dataset

$\log(x + 1)$ to the energy deposit matrix during training, like we did in [3, 4], as it provides a better generation quality.

Quality Evaluation

In our case generated objects do not represent an image in general terms, so we can not use perception-based approaches, that apply some pretrained models to compare real and generated image. Thus, we use precision-recall-based method.

Given a reference distribution P and a learned distribution Q , precision intuitively measures the quality of samples from Q , while recall measures the proportion of P that is covered by Q . As the model generates samples that are close to some real ones, precision increases. Meanwhile, to avoid generating same objects and evaluate variety, the model needs to generate *different* samples that are close to *different* real samples, increasing the recall.

Through our experiments we use precision and recall for distributions (PRD) [30], an efficient algorithm to compute these quantities, to evaluate the performance of different models, comparing the quality of generated samples.

As we want to evaluate generated responses not only in terms of general quality but in terms of physics metrics as well, we use the minimum of two PRD-AUC scores, evaluated over raw images (RAW PRD) and a set of the physics statistics (Physical PRD):

- shower asymmetry along and across the direction of inclination;
- shower width;
- the number of cells with energies above a certain threshold, the sparsity level.

PRD requires discrete distributions as its input, so the evaluation pipeline looks as follows:

- unite the objects from real and generated distributions.
- use energy deposit matrix as features to represent object in case of RAW PRD.
- use physical properties of the object as features in case of Physical PRD.
- cluster all objects using MiniBatchKMeans with 400 clusters based on image or physical features.
- evaluate the PRD over the pair of histograms built after the clustering procedure.

Results and Discussion

In order to compare the behavior of different approaches, we train the same model multiple times using different regularization terms and weights α to achieve different

L-constants, as it is the only hyperparameter that can be chosen on our own. Our approach requires setting the margin value, thus we divide runs into 6 groups. Every single group represents runs with close values of the achieved L-constant. Then we use the average of constants in every group, achieved using other methods, and set it as a margin for our loss. We choose the best model based on PRD evaluated on the set of physical metrics, as described in "Quality Evaluation".

Methods that were mentioned previously, e.g., (6), were highly sensitive to their only hyperparameter α , the weight of the regularization term. Most of the runs provided us with highly unstable training procedure (low α , high L-constant) or both generative and regressive losses did not improve (high α , low expressivity). In our approach we can explicitly pass the desired constant as a hyperparameter, and even it is not guaranteed for model to have exactly this values as its constant, we achieve the desired behavior. Model's constant becomes closer the margin m , allowing us to directly search for the L value, providing us with the best performance.

Another advantage comes from the fact that our loss do not penalize the model if its constant is less than margin, thus we can pick up a relatively high α and it would not lead to a constant's degradation. Basically, if we set margin $m = 0$, formulation our loss becomes closer to 8, however it may be clearly seen from Fig. 3 that $L_i n f$ without margin provides us with worse quality of reproduced physics metrics, as initially it penalizes the model even with a low L-constant.

During the hyperparameters' tuning of other approaches, we often faced the problem of L-constant with values below one. This turns into a poor quality, leading us to omit these results on the plot, as they were way worse than the 1-constant baseline. This quality degradation be explained by the fact of model's decreasing capacity.

According to Figs. 2 and 3 all methods have sweet-spots providing the best possible quality. As the constant is close to 1 the model has lower capacity. As we increase the constant, the training procedure becomes unstable. However, in our approach we don't penalize the model if it's constant is less than the margin, and it leads to better results when we allow the model to achieve higher constant. Even in case when we set the margin equal to 1, we boost the quality of the baseline as we allow layers to vary their constants as we only want the product to be lower than one but not the every single constant.

According to Figs. 4 and 5, the higher margin and constant we have, the lower learning rate we should use. This fact should be taken into account for our approach, otherwise we can face a situation when $\gamma > \frac{2}{L} \geq \frac{2}{m}$, model does not converge and our loss does not push the model to decrease its constant.

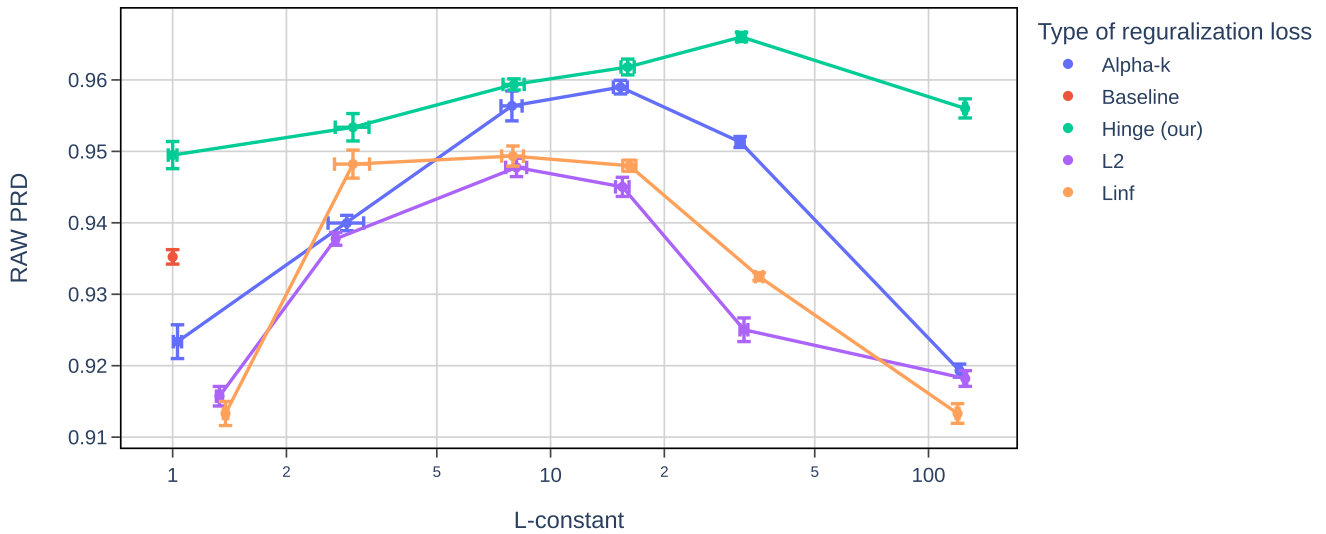


Fig. 2 RAW PRD-score of different regularization techniques, evaluated over raw images. Baselines represents model without SN; Alpha-k, L2, Linf represent approaches (6), (7), (8) respectively; Hinge represents our approach (9)

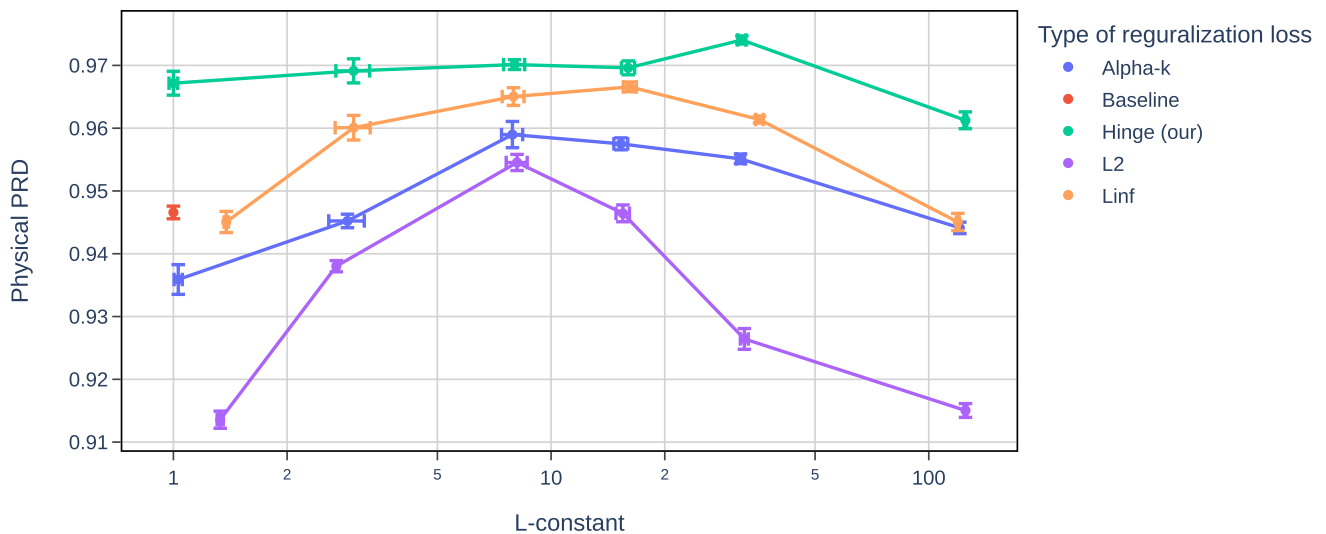


Fig. 3 Physical PRD-score of different regularization techniques, evaluated over physical properties ("Quality Evaluation"). Baselines represents model without SN; Alpha-k, L2, Linf represent approaches (6), (7), (8) respectively; Hinge represents our approach (9)

Figure 4 shows the overall situation, meanwhile Fig. 5 focuses on the part that is mostly important for our needs—the learning rate values when models start to converge. These values slightly differ from the theoretical estimations. However, we may notice that models that have relatively low margins (1 and 8) start to converge only when they have learning rates less than theoretical estimations. Meanwhile, models with relatively high margins (64 and 125) start to do it with even a bit higher learning rates than we expect. This behavior may be explained as we do

not set the L-constant of the model explicitly, but we set the margin to add penalty for the cases above it. Thus, the model may achieve even lower constant itself and start to converge with a slightly higher learning rate.

Even though quality gain values of all the trained models with different margins trained using learning rates $\eta \in [10^{-4} : 10^{-2}]$ are close to each other, and these rates may be a reliable choice to pick from, Figs. 2 and 3 show that margins provide different best achieved qualities through the whole training.

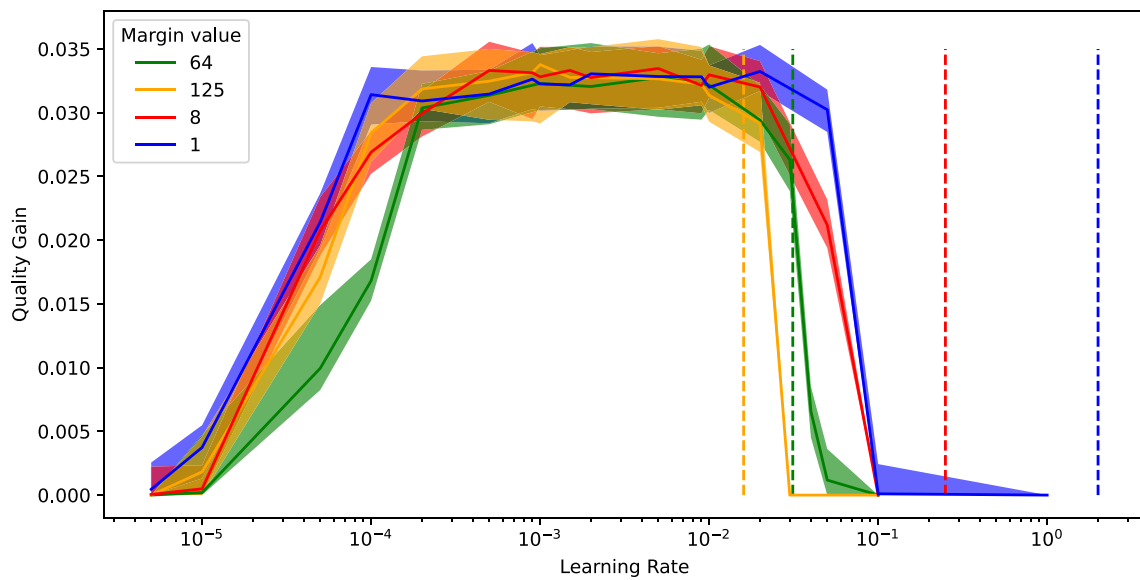


Fig. 4 The relationship between the quality gain and learning rate. $L/2$ lines are plotted as dotted ones

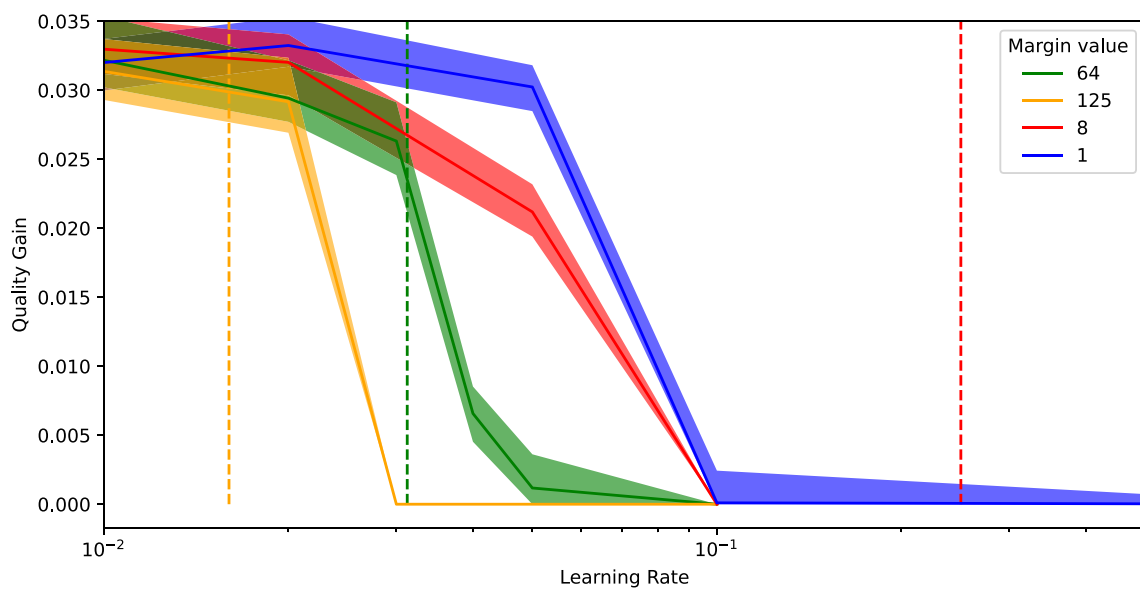


Fig. 5 The relationship between the quality gain and learning rate zoomed

Conclusion

In this paper we propose a novel adaptive normalization technique that allows us to control the balance between GAN’s capacity and training stability. We successfully apply this technique to the LHCb ECAL dataset, boosting previously published quality metrics.

To deal with the model’s capacity reduction we introduce an additional loss with a margin hyperparameter,

allowing it to avoid penalizing the model with a constant that is lower than the predefined value. We demonstrate that it improves the quality of generated objects, comparing with other techniques.

We also consider the relationship between margin and learning rate and empirically show that increasing the margin we should decrease the learning rate for a better model convergence. This relationship should be taken into account during training procedure hyperparameters’ optimization. The

problem of searching the margin value, providing the best quality and stability is going to be a subject for the future work.

Fast simulation applications would benefit from the proposed method, as it allows stabilizing GAN training, providing quality boosting. Our Hinge regularization is a generic technique that is not initially restricted to LHCb ECAL case. We would be interested in applying our regularization to

other architectures, even not GAN-based ones, tasks, and datasets.

Appendix A Dataset Details

See Figs. 6 and 7.

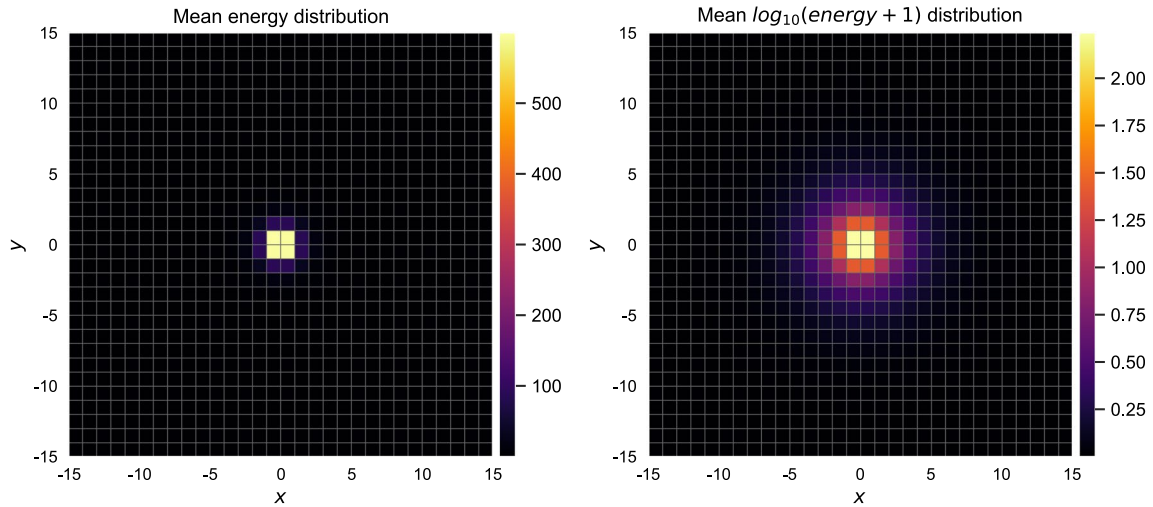


Fig. 6 Visualization of the averaged deposited energy in ECAL dataset in original and logarithmic domain

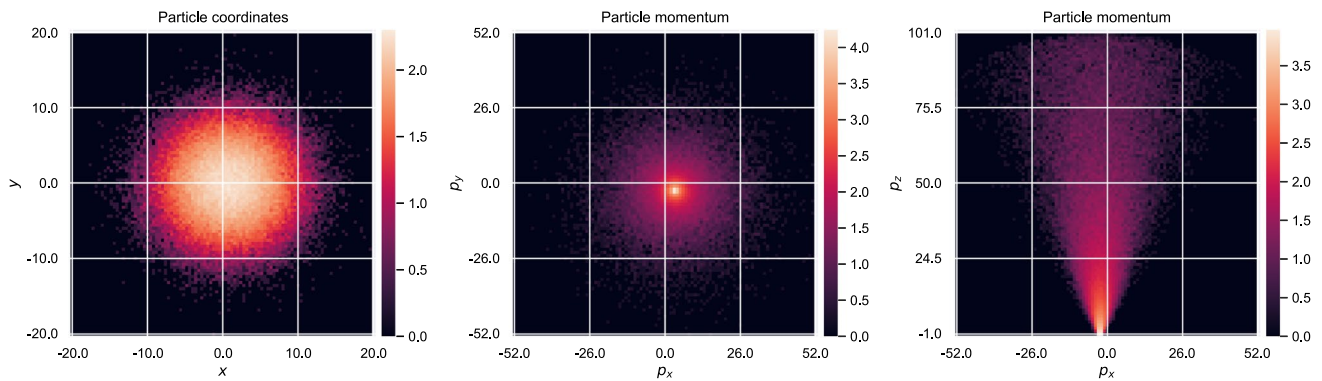


Fig. 7 Distributions of particle point and momentum, used as condition for generation, in the ECAL dataset

Appendix B Training Details

The general idea of the discriminator’s architecture that was used through all the experiments is shown at Fig. 8.

The discriminator consists of multiple convolutional layers one after another. We used 5 layers with LeakyReLU activation function. The output of the 3rd convolutional layer is fed both into the 4th convolutional layer and into the regressor [4], thus we use very first layers for both generative and regression tasks. The regressor itself is quite straightforward and consists only of a convolutional layer and a fully connected one. We use the output of the FC-layer to calculate regressive loss and feed it back to the Discriminator, concatenating it with other inner features. The flattened and concatenated features are processed by multiple fully-connected layers, providing the final output of the Discriminator.

We apply spectral normalization to all the layers to set the model’s Lipschitz constant to 1. Then we add a trainable constant after every single FC-layer to perform our experiments, using these values to calculate all the regularization terms.

Baseline model was trained using no regularization technique, its Lipschitz constant is set to 1. Alpha-k (6), L_2 (7), L_{inf} (8) and our Hinge (9) models were trained using same architecture, varying type and hyperparameters of regularization term. All models were trained using the following parameters:

- Generator’s input latent dimension size: 256.
- Batch size: 256.
- Number of Discriminator/generator learning steps ratio: 3.
- Optimizer: Adam(betas=[0,0.99]).
- Number of epochs: 150.
- GPU: NVIDIA RTX 3090.

One training epoch takes 3.57 ± 0.13 min.

Fitted generator requires $1.18 \text{ ms} \pm 69.9 \mu\text{s}$ to generate one object. However, it is complicated to compare it with GEANT4 performance (250 ms) side-by-side, as we can utilize batch inference, e.g., it takes $17.8 \text{ ms} \pm 467 \mu\text{s}$ to generate a batch of 4096 objects.

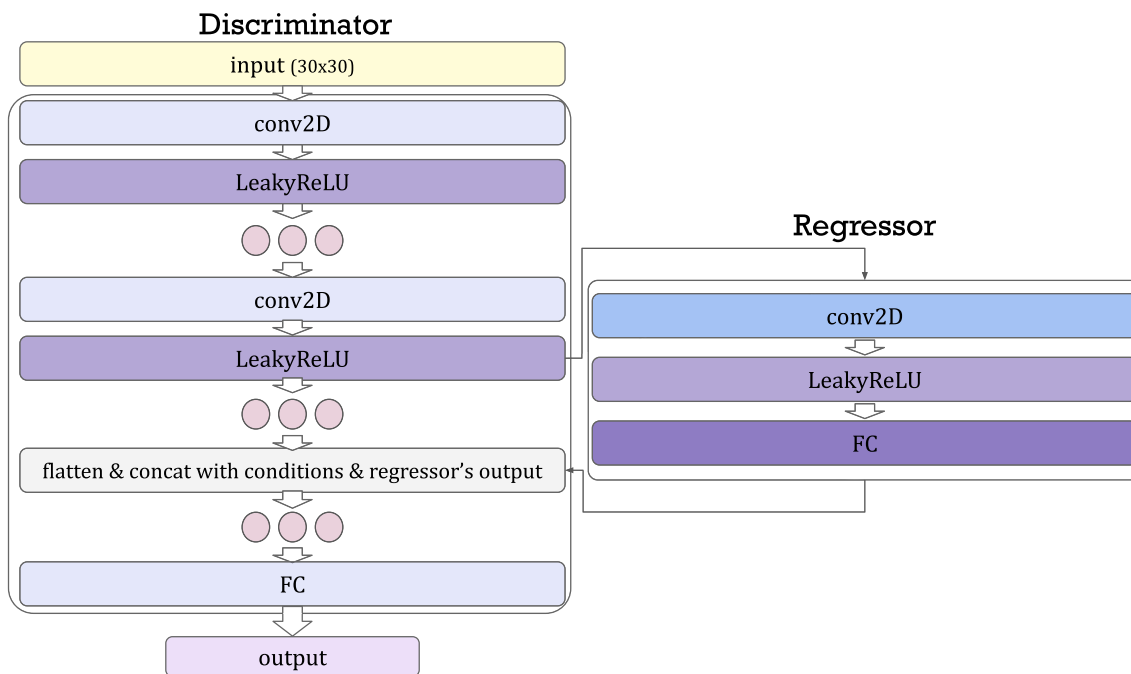


Fig. 8 Visualization of the architecture

Appendix C Physical metrics

Here we plot distributions of physical properties of generated objects achieved using baseline without any

regularization and best models trained with regularization methods Alpha-k (6), L_2 (7), L_{inf} (8) and our Hinge (9) (Figs. 9, 10, 11, 12, 13).

Fig. 9 Physical metrics distributions of dataset, generated using baseline

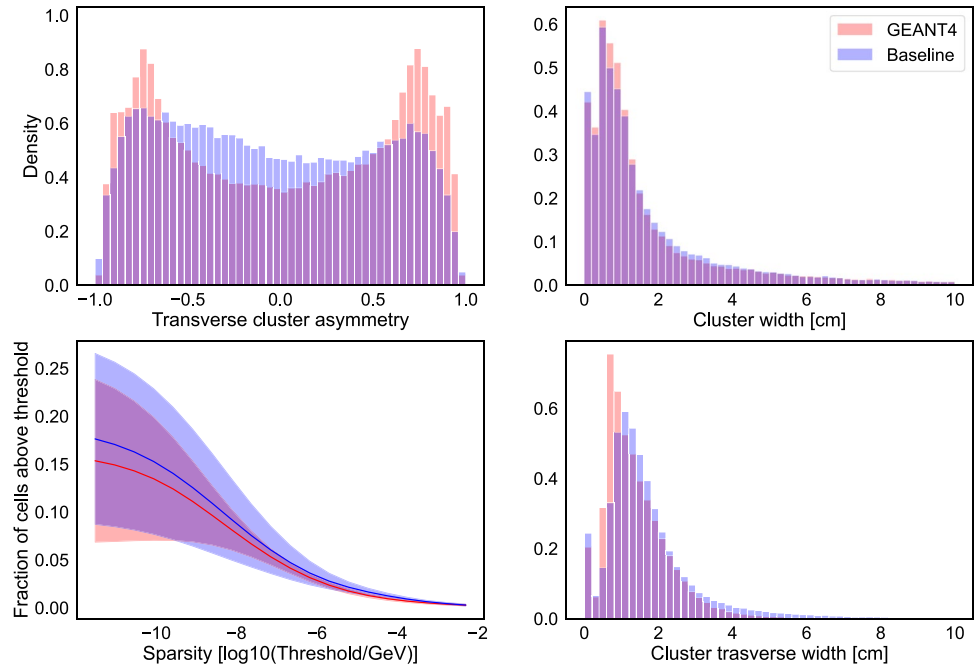


Fig. 10 Physical metrics distributions of dataset, generated using L_2 (7)

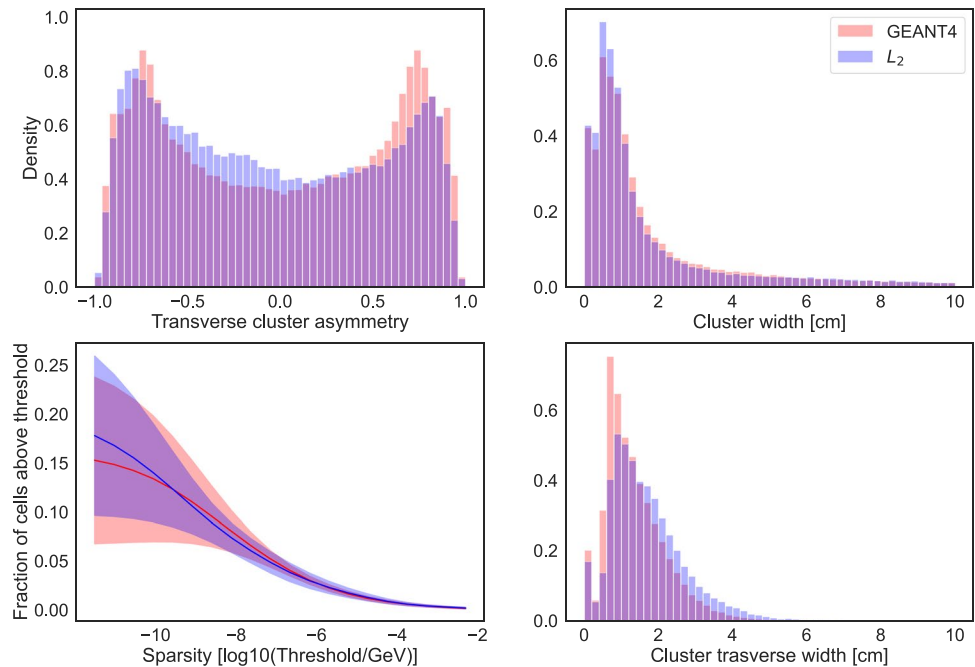


Fig. 11 Physical metrics distributions of dataset, generated using Alpha-k (6)

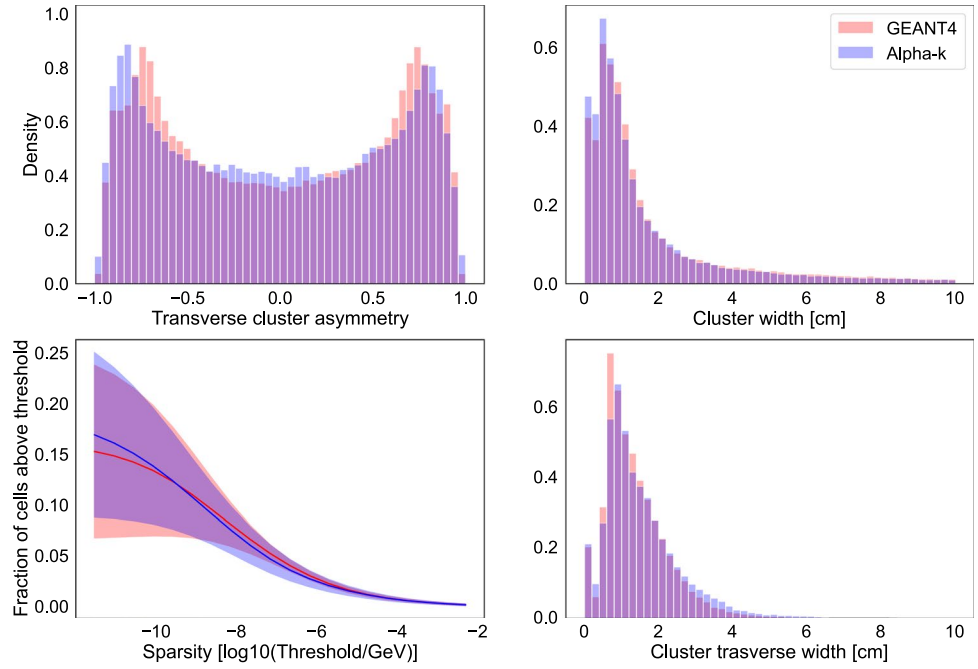


Fig. 12 Physical metrics distributions of dataset, generated using L_{inf} (8)

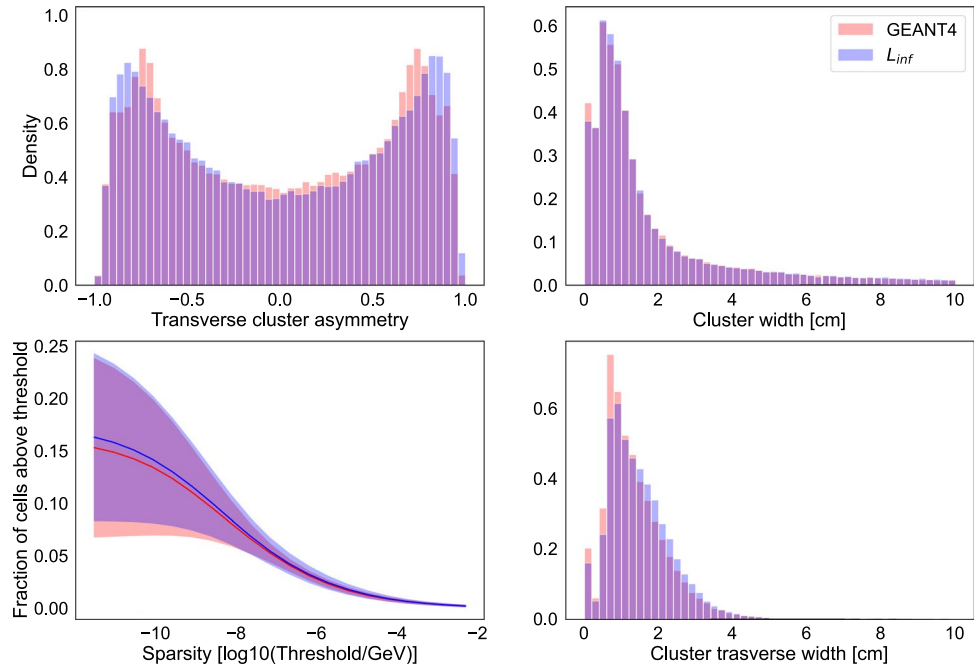
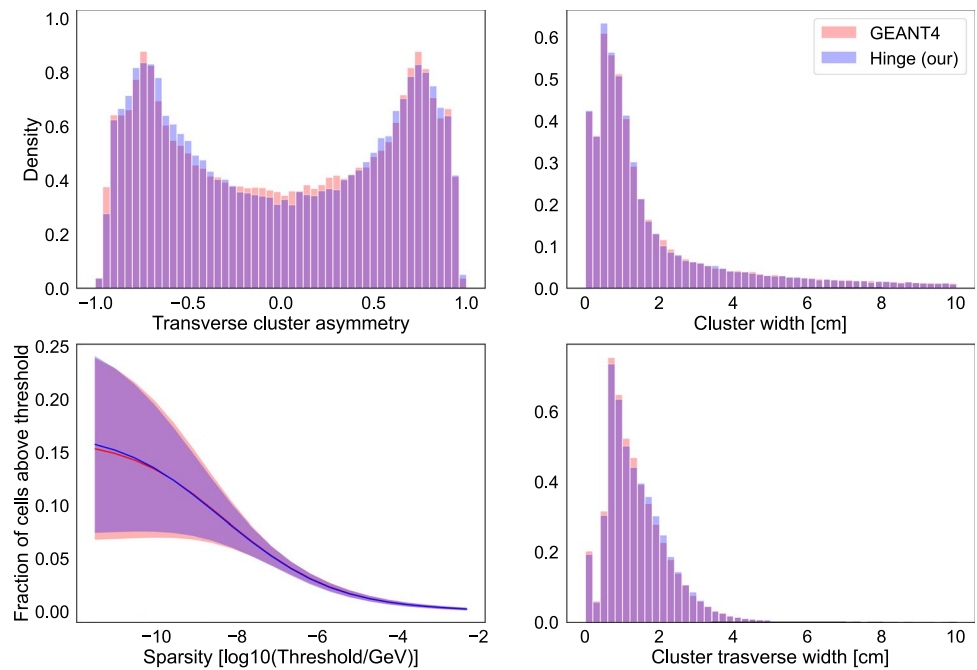


Fig. 13 Physical metrics distributions of dataset, generated using Hinge (9)



Acknowledgements The research leading to these results has received funding from the Basic Research Program at the National Research University Higher School of Economics. This research was supported in part through computational resources of HPC facilities at NRU HSE.

Author contributions A.R. wrote the main manuscript text, prepared figures, performed all the model experiments. FR wrote the introduction, discussed experiments and reviewed the manuscript.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. The LHCb Collaboration (2008) The LHCb detector at the LHC. *J Instrum* 3(08):08005–08005. <https://doi.org/10.1088/1748-0221/3/08/s08005>
2. Agostinelli S et al (2003) Geant4—a simulation toolkit. *Nucl Instrum Methods Phys Res Sect A* 506:250. [https://doi.org/10.1016/S0168-9002\(03\)01368-8](https://doi.org/10.1016/S0168-9002(03)01368-8)
3. Rogachev A, Ratnikov F (2021) Fast simulation of the electromagnetic calorimeter response using self-attention generative adversarial networks. *EPJ Web Conf* 251:03043. <https://doi.org/10.1051/epjconf/202125103043>
4. Rogachev A, Ratnikov F (2022) GAN with an auxiliary regressor for the fast simulation of the electromagnetic calorimeter response. *arXiv*. <https://doi.org/10.48550/ARXIV.2207.06329>. <https://arxiv.org/abs/2207.06329>
5. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. *arXiv*. <https://doi.org/10.48550/arXiv.1406.2661>
6. Paganini M, Oliveira L, Nachman B (2018) Calogan: simulating 3d high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. *Phys Rev D* 97(1):041021. <https://doi.org/10.1103/physrevd.97.014021>
7. Chekalina V, Orlova E, Ratnikov F, Ulyanov D, Ustyuzhanin A, Zakharov E (2019) Generative models for fast calorimeter simulation: the lhcb case. *EPJ Web Conf* 214:02034. <https://doi.org/10.1051/epjconf/201921402034>
8. Erdmann M, Geiger L, Glombitza J, Schmidt D (2018) Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks. *Comput Softw Big Sci* 2:4. <https://doi.org/10.1007/s41781-018-0008-x>
9. Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasieczka G, Korol A, Krüger K (2021) Getting high: high fidelity simulation of high granularity calorimeters with high speed. *Comput Softw Big Sci* 5(1):13. <https://doi.org/10.1007/s41781-021-00056-0>
10. Erdmann M, Glombitza J, Quast T (2019) Precise simulation of electromagnetic calorimeter showers using a Wasserstein generative adversarial network. *Comput Softw Big Sci* 3(1):1–13. <https://doi.org/10.1007/s41781-018-0019-7>
11. Belayneh D, Carminati F, Farbin A, Hooberman B, Khattak G, Liu M, Liu J, Olivito D, Pacela VB, Pierini M, Schwing

- A, Spiropulu M, Vallecorsa S, Vlimant J-R, Wei W, Zhang M (2020) Calorimetry with deep learning: particle simulation and reconstruction for collider physics. *Eur Phys J C* 80(7):1–31. <https://doi.org/10.1140/epjc/s10052-020-8251-9>
12. Vallecorsa Sofia (2019) Carminati, Federico, Khattak, Gulrukh: 3d convolutional gan for fast simulation. *EPJ Web Conf* 214:02010. <https://doi.org/10.1051/epjconf/201921402010>
 13. Sergeev F, Jain N, Knunyants I, Kostenkov G, Trofimova E (2021) Fast simulation of the LHCb electromagnetic calorimeter response using VAEs and GANs. *J Phys Conf Ser* 1740(1):012028. <https://doi.org/10.1088/1742-6596/1740/1/012028>
 14. Giannelli MF, Kasieczka G, Krause C, Nachman B, Salamani D, Shih D, Zaborowska A (2022) Fast calorimeter simulation challenge. <https://calochallenge.github.io/homepage/>. <https://calochallenge.github.io/homepage/>
 15. Giannelli MF, Zhang R (2023) CaloShowerGAN, a Generative Adversarial Networks model for fast calorimeter shower simulation. arXiv. <https://doi.org/10.48550/arXiv.2309.06515>
 16. Pang I, Raine JA, Shih D (2023) Supercalo: calorimeter shower super-resolution. arXiv preprint [arXiv:2308.11700](https://arxiv.org/abs/2308.11700)
 17. Buckley MR, Krause C, Pang I, Shih D (2023) Inductive caloflow. arXiv preprint [arXiv:2305.11934](https://arxiv.org/abs/2305.11934)
 18. Krause C, Pang I, Shih D (2022) Caloflow for calochallenge dataset I. arXiv preprint [arXiv:2210.14245](https://arxiv.org/abs/2210.14245)
 19. Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasieczka G, Korol A, Korcari W, Krüger K, McKeown P (2023) Caloclouds: fast geometry-independent highly-granular calorimeter simulation. *J Instrum* 18(11):11025
 20. Amram O, Pedro K (2023) Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation. *Phys Rev D* 108(7):072014
 21. Mikuni V, Nachman B (2022) Score-based generative models for calorimeter shower simulation. *Phys Rev D* 106(9):092009. <https://doi.org/10.1103/physrevd.106.092009>
 22. Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. arXiv. <https://doi.org/10.48550/arXiv.1802.05957>
 23. Arjovsky M, Bottou L (2017) Towards principled methods for training generative adversarial networks. arXiv. <https://doi.org/10.48550/ARXIV.1701.04862>. arXiv:1701.04862
 24. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein GAN. arXiv. <https://doi.org/10.48550/ARXIV.1701.07875>. arXiv:1701.07875
 25. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of Wasserstein GANs. arXiv. <https://doi.org/10.48550/ARXIV.1704.00028>. arXiv:1704.00028
 26. Yoshida Y, Miyato T (2017) Spectral norm regularization for improving the generalizability of deep learning. arXiv. <https://doi.org/10.48550/ARXIV.1705.10941>. arXiv:1705.10941
 27. Anil C, Lucas J, Grosse R (2019) Sorting out lipschitz function approximation. In: *International Conference on Machine Learning*, pp. 291–301
 28. Liu H-TD, Williams F, Jacobson A, Fidler S, Litany O (2022) Learning smooth neural functions via lipschitz regularization. arXiv. <https://doi.org/10.48550/ARXIV.2202.08345>. arXiv:2202.08345
 29. Taylor AB, Hendrickx JM, Glineur F (2017) Exact worst-case convergence rates of the proximal gradient method for composite convex minimization. arXiv. <https://doi.org/10.48550/ARXIV.1705.04398>. arXiv:1705.04398
 30. Sajjadi MSM, Bachem O, Lucic M, Bousquet O, Gelly S (2018) Assessing Generative Models via Precision and Recall

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.