Fast simulation of the LHCb electromagnetic calorimeter response using VAEs and GANs

Fedor Sergeev\textsuperscript{1,4}, Nikita Jain, Ivan Knunyants, George Kostenkov\textsuperscript{2} and Ekaterina Trofimova\textsuperscript{2,3}

\textsuperscript{1}Department of Control and Applied Mathematics, Moscow Institute of Physics and Technology, 9 Institutskiy per., Dolgoprudny 141701, Russian Federation
\textsuperscript{2}Department of Computer Science, National Research University Higher School of Economics, 11 Pokrovsky Boulevard, Russian Federation
\textsuperscript{3}Skolkovo Institute of Science and Technology, 30, bld. 1 Bolshoy Boulevard, Russian Federation

E-mail: sergeev.fi@phystech.edu

Abstract. Modern experiments in high-energy physics require an increasing amount of simulated data. Monte-Carlo simulation of calorimeter responses is by far the most computationally expensive part of such simulations. Recent works have shown that the application of generative neural networks to this task can significantly speed up the simulations while maintaining an appropriate degree of accuracy. This paper explores different approaches to designing and training generative neural networks for simulation of the electromagnetic calorimeter response in the LHCb experiment.

1. Introduction

Large Hadron Collider (LHC), built by the European Center for Nuclear Research (CERN), is arguably the most giant scientific experiment in the world. LHC is a proton and heavy-ion collider. Its operation has led to the discovery of the Higgs boson, quark-gluon plasma, tetra-, and pentaquarks, as well as providing other valuable results for particle physics.

LHC hosts seven detectors, which register the results of particles collisions — beams of secondary particles. A detector consists of multiple sub-detectors, which specialize in measuring specific parameters of passing particles. Sub-detectors identify particle type, measure particle trajectory and energy. The resulting data allows for the reconstruction of the initial collision.

To perform such reconstruction, simulations of detector response for a given input are required. In the case of LHC, the amount of necessary simulation data is extremely large: it exceeds 15 petabytes per year [1]. Massive amounts of computations call for the development of new fast simulation methods.

One of the most computationally expensive part of an LHC detector simulation is the Monte Carlo simulation of the calorimeter responses [2]. In this work, we develop a generative models-based method for simulating response of the electromagnetic calorimeter of the LHCb detector.

\textsuperscript{4} Present address: Section of Mathematics, École polytechnique fédérale de Lausanne, EPFL FSB SMA, Lausanne 1015, Switzerland.
2. LHCb detector

The Large Hadron Collider beauty (LHCb) experiment is one of the four main experiments of LHC. It investigates CP violation and studies the interaction of b-hadrons. The LHCb detector consists of the following parts [3]:

- a vertex locator, which is used for fine measurements of particle trajectories for separation of primary and secondary particles,
- two ring-imaging Cherenkov detectors (RICH1-2), which identify types of low- and high-momentum particles,
- a tracking system (TT, TT1-3), which records particles’ trajectories.
- a magnet, which bends and directs the track of charged particles,
- an electromagnetic (ECAL) and a hadronic (HCAL) calorimeters, which measure energy of photons and electrons, and hadrons respectively,
- a set of Muon detectors (M1-5), which identify muons.

The LHCb detector has two calorimeters: ECAL and HCAL. The HCAL response depends on the ECAL response. Therefore, for simplicity, we only consider the simulation of the ECAL response of LHCb.

The electromagnetic calorimeter is made of alternating layers of lead and scintillation plates. When a photon or an electron enters the calorimeter, it collides with a metal plate, producing a few secondary photons and electrons. The produced particles hit metal plates too, and, in turn, create even more particles. The following chain reaction produces a stream of electrons and photons, which is called an electromagnetic shower.

The secondary particles from the electromagnetic shower interact with plastic molecules in scintillation plates when traversing through the calorimeter. The excited scintillator molecules emit ultraviolet light, which is then collected and measured. The amount of light emitted by the scintillation plates is proportional to the particle’s energy that entered the calorimeter. Therefore the detector can measure the energy of the primary particle.

It is important to note that ECAL has a cellular structure. It is used for measuring the energy of multiple particles entering the calorimeter at the same time.

The amount of particles entering the ECAL close to the center of the initial collision is much more considerable than the number of particles towards the detector’s edge. Therefore,
smaller cells are installed the center of the ECAL and larger cells on the edges of ECAL. This configuration allows ECAL to measure the energy of more independent particles and ultimately increase the measurement accuracy.

3. Generative neural networks
A generative model is a type of neural network that is used for reproducing a given distribution. They have shown excellent results for generating complex high-dimensional data, such as text [4] and images [5].

The two most popular and effective generative neural networks are variational autoencoder (VAE) and a generative adversarial network (GAN). In this section, we discuss the working principles of VAEs and GANs.

3.1. Conditional Variational autoencoder
CVAE [9, 13] is a directed model which consists of two main parts: an encoder (E) and a decoder (D). E forms a hidden representation, or latent space z, of the input data. D restores the original form of the input data from the encoded one by sampling from a distribution \( p_{VAE}(x|z) \) (fig. 2). Encodings distribution is regularised during the training to ensure that its latent space has good properties, allowing new data generation. More formally, the approximate inference \( q(z|x) \) during training is used as an E network to receive \( z \), and \( p_{VAE}(x|z) \) is used as a D to generate data. The trade-off between E and D can be achieved by variational lower bound \( L(q) \) maximization:

\[
L(q) = \exp_z q(z|x) \log p_{VAE}(z, x) + H(q(z|x)) = \exp_z q(z|x) \log p_{VAE}(x|z) - D_{KL}(q(z|x)||p_{VAE}(z)) \leq \log p_{VAE}(x),
\]

where the first term is a joint log-likelihood of the input data and the latent space, second term is the approximate posterior entropy, \( q \) follows the Gaussian distribution.

3.2. Conditional Generative adversarial network
GAN [11] is another generative method based on differentiable generator networks. Unlike CVAE, CGAN does not provide a precise density estimation. Instead, it is based on a game theory approach: the two networks, Generator (G) and Discriminator (D), compete against each other to find Nash equilibrium. G directly produces samples from a simple distribution, e.g. Gaussian, \( x = g(z; \theta^g) \). D emits the probability \( x = d(x; \theta^d) \) that the sample was obtained...
from the distribution of the data and not from the distribution of the model (fig. 3). Thus, the generated data should follow a distribution that is as close as possible to the true distribution of the original data. More formally, the minimax game of two networks D and G can be represented as follows:

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

(2)

where $p_{\text{data}}$ is a true data distribution.

One of the most regular GAN variations is the conditional GAN (CGAN), which is built by adding the conditional vector along with the noise vector [12]. CGAN allows to provide an additional information to the model in order to generate the data with some desired features.

3.3. Application to High Energy Physics

Application of deep generative models for simulations in high energy physics was studied in papers [6, 7]. The authors aimed to replace the traditional Monte Carlo simulator with a much quicker CGAN-based simulator. The algorithm proposed in [6] is based on DCGAN architecture [17], and uses only particles’ energy parameter as an input. In [7], a conditional Wasserstein CGAN [16] is considered on particle coordinates and momentum projections as an input, and particle energies as an output. The latter approach shows better performance in terms of generation accuracy. Both methods demonstrate significant simulation-time speedups compared to the standard Monte Carlo simulation performed with the GEANT4 package [8].

In this paper, we develop upon results presented in [7]. New CVAE and combined CVAE+CGAN architectures are proposed, as well as a more straightforward design of a CGAN architecture. The proposed CGAN architecture reaches a comparable generation quality and is easier and quicker to train since its discriminator does not use a pretrained regressor. We also test our models on energy distributions, produced by both electrons and photons, while in [7] only electrons were used.

4. Data

We consider an event when one electron or photon with known momentum $p = (p_x, p_y, p_z)$ enters the calorimeter at a known location $r = (x, y)$. The calorimeter response is presented as a $30 \times 30$ energy distribution matrix. We consider that an element of this matrix is equal to the amount energy that is located in a $2 \times 2$ cm$^2$ physical area.

Note that the calorimeter cell modules are 4, 6, and 12 cm in length. Thus, by averaging “2 cm-long” matrix elements, we will be able to obtain values for all three cell module sizes. In the
The total number of cells of the energy distribution matrix (30) is chosen so that the matrix could capture the information about one full electromagnetic shower, which width can reach approximately 60 cm.

Energy distributions for the given $r, p$ are computed using the GEANT4. In our work for each particle type, we use 100 thousand samples for training and 100 thousand for testing. The initial particles’ coordinates lie in the range $[-1, 1]$ cm. Energy is given in GeV and lies in the range from 1 to 100 GeV. Momentum is calculated such that the angle at which the particle enters the calorimeter would be normally distributed.

(a) Coordinates and momentum histograms. (b) Mean energy distributions. 

Figure 4: Visualisation of the training data.

5. Metric

To assess the quality of the model output we use precision-recall distribution area under the curve metric (PRD-AUC) [14]:

$$\text{PRD}(Q, P) = \{ (\theta \alpha(\lambda), \theta \beta(\lambda)) | \lambda \in (0, \infty), \theta \in [0, 1] \},$$

where $P$ and $Q$ are distributions, that are defined on a finite state space,

$$\alpha(\lambda) = \sum_{\omega \in \Omega} \min(\lambda P(\omega), Q(\omega)).$$
\[ \beta(\lambda) = \sum_{\omega \in \Omega} \min(P(\omega), \frac{Q(\omega)}{\lambda}). \]

The PRD-AUC score is a metric that disentangles precision (quality of generated samples) from recall (proportion of target distribution that is covered by the generated distribution, richness).

In our work, we use the minimum of two PRD-AUC scores:

(i) over raw images: catches overall proximity of generated and real calorimeter responses distributions

(ii) over a set of physical metrics: catches proximity of distributions for the following physical statistics:

- shower asymmetry along and across the direction of inclination: cluster asymmetry is calculated relatively to the line of projection of inclined track;
- shower width along and across direction;
- sparsity level of the data: the number of cells with energies above a certain threshold, basically shows how well the medium and low energy tails of energy distributions are modeled overall.

Thus, two energy distributions are considered similar if, first, distributions themselves, and, second, distributions of secondary characteristics are identical. The use of the "physical" PRD-AUC part allows us to measure the sample quality more appropriately for our application.

6. Preprocessing

To speed up and stabilize the training process of a generative model, one can use data preprocessing. In our case, the center pick of energy distribution is much higher than the rest of the energy distribution. This presents a problem as generative models might not capture the energy distributions towards the edge of the matrix. It is, therefore, necessary to consider using the normalization of the energy distribution vector.

We consider the following energy distribution processing types:

(i) logarithmic transformation

\[ f(E) = \log (1 + E), \quad (4) \]

(ii) Box-Cox transformation

\[ f(E, \lambda) = \begin{cases} 
\frac{E^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\
\log E & \text{if } \lambda = 0.
\end{cases} \quad (5) \]

Box Cox transformation is usually viewed as an extension of the logarithmic transformation. Both of these functions flatten the central peak of the energy distribution. This, in theory, should allow the generative model to better reproduce energies towards the edges of the matrix.

7. Models

We use CVAE and CGAN models for generating an energy distribution matrix with the concatenated \( r \) and \( p \) vectors as a condition.
7.1. CVAE

The model consists of two neural networks: encoder and decoder. Since the model is symmetrical with respect to latent space, we want the encoder and decoder to be similar. A simple architecture also does training and tuning the network simpler.

The encoder consists of one stride-1 and four sequential stride-2 convolution layers. The output of the convolutions is then flattened and concatenated with the condition vector of the given energy image. Next, three fully connected layers reduce the dimension of the vector. Finally, two independent, fully connected layers are applied to the product of previous layers to generate vectors of means and variances.

The decoder takes a generated noise from means and variances and passes it through a fully connected layer. The output is concatenated with the condition vector. The following three layers are reshaped to be passed through the four following stride-2 convolutions and one stride-1 convolution.

7.2. CGAN

As it was previously mentioned, CGAN consists of two neural networks: discriminator and generator. It was decided to use almost the same architecture for CGAN’s generator as for the CVAE’s decoder so that the comparison between these two approaches is more accurate. The discriminator is also quite similar to CVAE’s encoder (fig. 8).
Both previously mentioned architectures have their drawbacks. In the case of a CVAE, the training process is usually easy to establish, but the final quality of samples is far from ideal. In the case of a CGAN, it is the other way around: training a CGAN is notoriously difficult \[11\], but the ending quality is higher than that of CVAE. To improve upon both approaches, let us combine both architectures.

One way to improve CGAN architecture is to find a useful technique for combining the condition vector with the noise vector. In the original CGAN architecture \[12\], as well as in our CGAN architectures, a simple concatenation is used. We can, however, use a pre-trained CVAE instead. This allows us to, firstly, improve upon the CVAE results by adding additional CGAN-trained layers, and, secondly, simplify the CGAN training by using a shallow architecture (the shallow architecture is sufficient since CVAE already produces energy distributions of acceptable quality).

### Results and discussion

CVAE, CGAN, and a combined CVAE+CGAN models were trained for reproducing energy distributions, logarithmic energy distributions, and Box-Cox-transformed energy distributions\[5\]. The PRD-AUC scores of these nine models on the test dataset were then compared. Note that in the case of training on transformed energy distributions, the score is computed on the generated samples with the inverse transformation.

For all models, a batch size of 512 was used. Adam optimizers \[15\] with leaning rates $5 \cdot 10^{-4}$, $4 \cdot 10^{-4}$, $1 \cdot 10^{-4}$ were used for CVAE, CGAN discriminator, and CGAN generator respectively.

Wasserstein loss with local gradient penalty was used for training GANs. CGAN was trained with three discriminator iterations per 1 generator iteration for the reproduction of energy distributions, iterations ratio 2/1 for the reproduction of transformed energy distribution, and 4/1 ratio for the combined CVAE+CGAN model, respectively.

Results obtained for both electrons and photons are almost identical, so, for brevity, we present training curves and generation examples only for electrons.

---

\[5\] Source code is available at gitlab.com/lambda-hse/calogan_project.
First, let us compare the efficiency of preprocessing techniques. We see that the logarithm transformation of energy distributions enables CVAE and CGAN architectures to train faster and more stably. The same cannot be said for the CVAE+CGAN architecture. Our implementation of Box-Cox transformation does not seem to enable fast training of the model. However, we see that the CVAE+CGAN architecture rapidly achieves high results. Therefore we consider the Box-Cox transformation to be the least effective one.

Second, the CVAE+CGAN architecture is flexible. Adding layers to a pure CVAE or a CGAN architecture is often difficult since it can drastically change the behavior of a model, and a completely different set of hyperparameters must be used in order for it to work. The CVAE+CGAN architecture avoids such problems. Thus, one can say that this model is able to make the training more stable and predictable.
Table 1: PRD-AUC on test dataset for various energy preprocessing techniques $f(E)$.

Next, we can compare the generation results visually for the best models: CVAE trained on log energies, CGAN trained on log energies, and CVAE+CGAN trained on untransformed energies.

The CVAE-generated samples (fig. 10) look plausible, but smoothed-out. The samples don’t replicate the sudden jumps and drops in the energy distribution.

The CGAN-generated samples (fig. 11) to present such jumps and drops, but the absolute value of the energy distribution is not always replicated well, and, in some cases, the distribution itself is not precise enough.

The CVAE+CGAN-generated samples (fig. 12) avoid the issues of previous models. However, in this case, we see that the tails of the energy distribution are often exaggerated. This might suggest a need for applying some regularisation for the CGAN part of the model to avoid producing overestimated values for edges of the energy distribution.
9. Conclusion
Results obtained in this paper suggest that the development a production-ready generative model for reproducing energy distribution in the electromagnetic calorimeter of the LHCb experiment relies on a combination of an efficient architecture with an appropriate preprocessing of the input data. Three models were developed: a variational autoencoder, a generative adversarial network, and a combination of the two. They were trained on three variants of input energy distributions: unprocessed, transformed with a logarithm, and processed with a Box-Cox transformation. Best results are achieved by CGAN model implementation on logarithmic energy distributions. However, a combined CVAE+CGAN model has shown promising results and is more flexible than other two models. For both CVAE and CGAN, the best results were achieved on the logarithmic dataset.
Acknowledgments
The authors are thankful to Fedor Ratnikov for providing invaluable guidance and expertise. We thank Vladislav Belavin for providing data and code. The work leading to these results has received funding from Russian Science Foundation under grant agreement no 19-71-30020. The research was supported in part through computational resources of HPC facilities at NRU HSE.

References
[1] Boccali T 2019 Computing models in high energy physics Reviews in Physics 4 100034
[2] Bozzi C 2014 Tech. rep. CERN-LHCb-PUB-2015-004
[3] The LHCb Collaboration 2008 The LHCb Detector at the LHC, Journal of Instrumentation 3
[4] Yang Z, Hu Z, Salakhutdinov R and Berg-Kirkpatrick T 2017 Improved Variational Autoencoders for Text Modeling using Dilated Convolutions Int. Conf. on Machine Learning
[5] Brock A, Donahue J and Simonyan K 2019 Large Scale GAN Training for High Fidelity Natural Image Synthesis Int. Conf. on Learning Representations
[6] Paganini M, Oliveira L and Nachman B 2018 CaloGAN: simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks Phys. Rev. D 97, 014021
[7] Chekalina V, Orlova E, Ratnikov F, Ulyanov D, Ustyuzhanin A and Zakharov E 2019 Generative models for fast calorimeter simulation: the LHCb case EPJ Web Conf. 214 02034
[8] Agostinelli S et al 2003 GEANT4 — a simulation toolkit Nucl. Instr. Meth. A vol 506 no 3 pp 250–303
[9] Kingma D and Welling M 2014 Auto-encoding variational bayes Int. Conf. on Learning Representations
[10] Sohn K, Lee H and Y 2015 Learning structured output representation using deep conditional generative models Advances in Neural Information Processing Systems 28 pp 3483–3491
[11] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets Advances in neural information processing systems 3 pp 2672–2680
[12] Mirza M and Osindero S 2014 Conditional Generative Adversarial Nets Preprint arXiv:1411.1784
[13] Rezende D J, Mohamed S and Wierstra D 2014 Stochastic backpropagation and approximate inference in deep generative models Int. Conf. on Machine Learning
[14] Sajjadi M, Bachem O, Lucic M, Bousquet O and Gelly S 2018 Assessing generative models via precision and recall Conf. on Neural Information Processing Systems
[15] Kingma D and Ba J 2014 Adam: a method for stochastic optimization Int. Conf. on Learning Representations
[16] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V and Courville A C 2017 Improved training of Wasserstein GANs Advances in Neural Information Processing Systems vol 30 pp 5769—5779
[17] Radford A, Metz L and Chintala S 2016 Unsupervised representation learning with deep convolutional generative adversarial networks CoRR