WGANs for simulation of e.m showers in ATLAS calorimeter

Paul Klein (paul.klein@supelec.fr)
Supervisors: Gilles Louppe, David Rousseau, Kyle Cranmer

Simulations of particles collisions at Large Hadron Collider (LHC) rely on the Geant4 package. However, these computations are CPU intensive and time consuming. We explore a way to drastically reduce event generation time by working on one of the most consuming part of Geant4 simulations: electromagnetic showers in the ATLAS calorimeter. Our goal is to give a proof of concept that we can use a Deep learning technical such as Generative Adversarial Networks (GANs) to accurately generate this complex phenomenon. More precisely, we implement the Wasserstein GANs (WGAN) approach - with a gradient penalty - to tackle this problem. This seemingly stable algorithm free us from most of classical tricks to train vanilla GANs.

I. INTRODUCTION

Simulations of particles collisions at the LHC experiment are handled by the Geant4 package [2]. These full simulations are time and CPU intensive (it needs several minutes to generate a single event), and may represent a bottleneck for many analysis of high energy physics experiments. Various approximations can be done to make faster simulations. However the generation time is still non-negligible and they are not suited to all physics applications. Simulations are nonetheless essentials for physicists to have strong statistics, in order to test hypothesis or build new models.

Thus, our motivation is to drastically reduce event production time, by working on one of the most time consuming part of the simulation: the generation of electromagnetic shower in the calorimeter. Our approach is to generate electromagnetic showers conditioned by the incident particle energy thanks to a GAN. Using a deep learning technical such as GAN would have a major benefit compared to the Geant4 package: once trained, a GAN could produce thousands of events per second.

A. Related works

GANs proposed by [5] have been a breakthrough approach to estimate generative models via an adversarial process. We will dwell upon principles of this framework later, but this method has been tested on several scientific contexts. For instance, GANs have been applied in astronomy [10] and in cosmology [8].

Recently there has been a lot of activity in using deep learning technicals applied to particle physics too, with exciting results.
[3] deal with the generation of jet images, thus providing a bridge between generative modeling in the Machine Learning community and simulated physical processes in high energy particle physics. They applied the novel GAN architecture to the production of jet images – 2D representations of energy depositions from particles interacting with a calorimeter.
This work provided them a work basis to go further, and thus try to simulate 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with GANs [9].

B. Our contribution

Unlike the caloGAN paper [9], we used the recently proposed Wasserstein GAN (WGAN)[1] approach, that makes significant progress toward stable training of vanilla GANs. In addition, we follow the alternative method suggested in [6], by removing weight clipping and adding gradient penalty in WGAN. We thus rely on a principled solution with strong convergence properties.

Furthermore, in order to take into account the complexity of the problem (geometry of the calorimeter, electronic noise,...), we used an ATLAS internal dataset to train our WGAN.

Comparisons of Monte-Carlo samples and generated ones are still an open problem, and is a key step we had to think of for a possible implementation in full simulations. We try to build a reliable pipeline to efficiently compare both datasets, by computing classic physics variates that we want to reproduce with our generative model. In addition, we feed an external discriminator to see if it was able to distinguish between both classes.

II. SIMULATING E.M SHOWER DATA

A. Data

ATLAS calorimeters measure the energy a particle loses as it passes through the detector. It is designed to stop or absorb most of the particles coming from a collision, forcing them to deposit all of their energy within the detector. Calorimeters consist of layers of passive or absorbing high-density material (inter-lead) with layers of an active medium: liquid argon. We are specifi-

cally interested in the ATLAS electromagnetic calorimeter (ECAL), which measures the energy of electrons and photons as they interact with matter. The calorimeter is organized in four layers (Pre-Sampler, Strip, Middle, Back), each one containing cells to measure energy deposits. As shown by the diagram below (Fig. 1), cells geometry depends on layers, each one having its own specific structure.

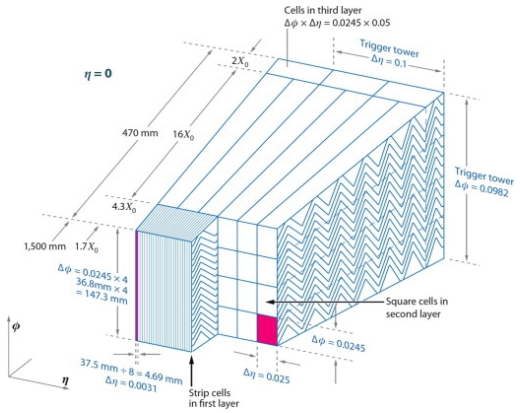

FIG. 1. Schematic of the ECAL.

We used an internal ATLAS dataset[1] to train our WGAN, and deal with a single particle type : photons. There is only one particle per event that is emitted at the center of the detector, and hits the calorimeter. We have access to several features such as the incident particle energy or location of the hit.

B. Preprocessing steps

Here is an outline of several specifications and simplifications we used on the former dataset to reduce the complexity of the problem, thus making the training part easier. We feed the WGAN with this preprocessed samples.

- We select events inside a specific region of the calorimeter to avoid edge effects ($0.1 < |\eta| < 1.1$)
- We focus on a 3*7 cluster. In total, we had 89 cells, spread over the four layers of the calorimeter (Fig. 2).
- We select events with a total energy below 300 GeV, which gives the following distribution of total energy per event (Fig. 3).

[1] mc15_13TeV:mc15_13TeV.423001.ParticleGun_single_photon_egammaET. merge.AOD.e3566_s2726_r7728_r7676

- We remove electronic noise from Monte-Carlo simulations. All cells energy deposit under a given threshold have been put to zero:

$$e < 120\,MeV \quad \text{(pre-Sampler)} \tag{1}$$

$$e < 60\,MeV \quad \text{(Strip)} \tag{2}$$

$$e < 200\,MeV \quad \text{(Middle)} \tag{3}$$

$$e < 200\,MeV \quad \text{(Back)} \tag{4}$$

By doing this, we ensured that we only have positive cells energy deposits.

- We divide energy cells deposits by the total energy of the event: thus, we had values in energy between 0 and 1 in the cluster.

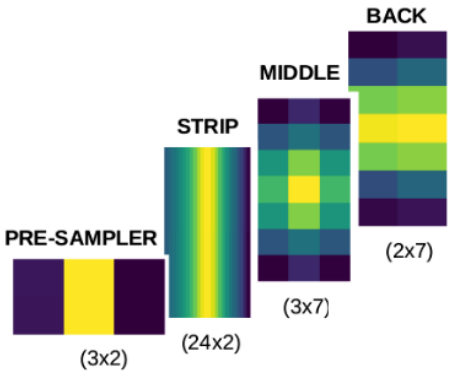

FIG. 2. Average energy deposit in the cluster (Monte-Carlo dataset).

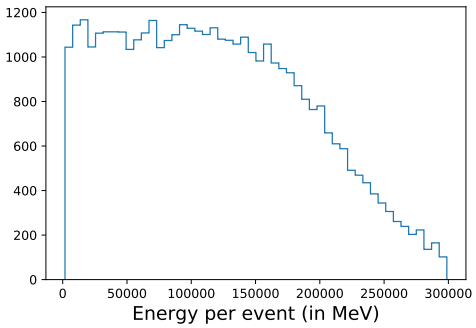

FIG. 3. Energy distribution of the Monte-Carlo dataset, after cutting below 300 GeV.

III. GENERATIVE ADVERSARIAL NETWORKS

A. Link with our problem

Our goal is to simulate electromagnetic showers in the ATLAS calorimeter. As explained previously, we want to drastically reduce event generation time because Geant4 simulations are CPU intensive. That's why neural networks suit well to our problem: once they have been trained, generate a new sample will take only few milliseconds.
We have followed several ideas during the project:

- conditioning electromagnetic showers to the energy of the incident particle. It is an extent of the former conditional GAN paper [7]. We feed the WGAN with normalized Monte-Carlo samples (as highlighted in the previous section), and the total energy of the event. After being trained, the generator produces samples of 89 values between 0 and 1, that we have to multiply by the conditional energy to recover values in MeV.

- in order to avoid as much as possible heuristic tricks to train our GAN, we rely on a seemingly strong algorithm for the training part: improved Wasserstein GAN [1].

- we want to check if our WGAN is able to learn more than just energy deposit in the cluster, and is capable of reproducing typical physics variates tied to electromagnetic showers.

B. Vanilla GANs

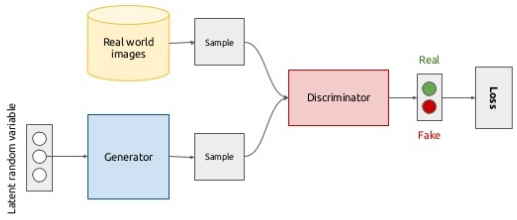

FIG. 4. Schematic of a GAN. (McGuinness, 2016)

GANs [5] are a framework for estimating generative models via an adversarial process, in which two models are simultaneously trained. A generative model G maps a source of noise to the input space, and a discriminative model D estimates the probability that a sample comes from the training data rather than G (Fig. 4). The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game, that we can formally express with the minimax objective:

$$\min_G \max_D \quad \mathop{\mathbb{E}}_{x \in Pr}[\log(D(x))] + \mathop{\mathbb{E}}_{\tilde{x} \in Pg}[\log(1 - D(\tilde{x}))] \quad (5)$$

However, GANs are well-known to be unstable: one can easily be confronted with vanishing gradients or mode collapse. While research continues to improve the fundamental stability of these models, there are bunch of tricks and hacks [4] you can use to train your GAN. But for most of them, you don't have any guarantee that it will fit with your specific problem and work. To sum up, vanilla GANs require long (and sometimes lucky) engineering to make them converge to the optimum.

C. Wasserstein GAN

At a more fundamental level, [1] argues that the Jensen-Shannon divergence, along with other common distances, are potentially not continuous and thus do not provide a usable gradient for the generator. Alternatively, they propose to use *Earth-Mover* (also called Wasserstein-1) distance, which is shown to have the desirable property of being continuous everywhere and differentiable almost everywhere. The WGAN value function results in a critic (new name to denote the discriminator in the paper) function whose gradient with respect to its input is much better behaved than its GAN counterpart; consequently optimization of the generator is made easier. The optimization problem for a WGAN is the following one:

$$\min_G \max_{D \in \Delta} \quad \mathop{\mathbb{E}}_{x \in Pr}[(D(x))] - \mathop{\mathbb{E}}_{\tilde{x} \in Pg}[D(\tilde{x})] \quad (6)$$

This new optimization problem gives rise to an additional requirement: the critic must lie within the space of 1-Lipschitz functions - referenced as Δ in formula 6. The authors choose to enforce this constraint through weight clipping. However, this solution drastically reduced network capacity as shown in [6]. Indeed, the optimal critic may not belong to 1-Lipschitz functions class. By clipping weights, we take the risk to converge to a non optimal solution.

D. Impoved training of WGANs

In this paper [6], authors removed weight-clipping, and add instead a regularization term in the loss function. This added term penalizes the network if its gradient norm moves away from 1. In other words, it has more or less the same effect than weight clipping, but it doesn't

reduce the network capacity as before.
The new objective is:

$$\min_G \max_{D\in\Delta} \mathop{\mathbb{E}}_{x\in P_r}[(D(x))] - \mathop{\mathbb{E}}_{\tilde{x}\in P_g}[D(\tilde{x})] + \lambda \mathop{\mathbb{E}}_{\hat{x}\in P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{7}$$

where λ is the gradient penalty coefficient.

The gradient term $\|\nabla_{\hat{x}} D(\hat{x})\|_2$ is with respect to the points $\hat{x}$, not the parameters of D. In geometric terms, $\hat{x}$ is a random point on the line between each pair of input points (a sample coming from the real distribution and an other one that has been generated).

$$\epsilon \sim U[0,1],\ x \sim P_r,\ \tilde{x} \sim P_g \tag{8}$$

$$\hat{x} = \epsilon x + (1-\epsilon)\tilde{x} \tag{9}$$

According to the authors, ***"WGAN with gradient penalty is the only method which successfully trains every architecture with a single set of default hyperparameters"***, for CIFAR-10 and LSUN datasets, compared to DCGAN, LSGAN, and former WGAN.

E. Conditional GANs

This is the conditional version of GANs. Starting with a classic GAN architecture, they can simply be built by feeding data we wish to condition on to both the generator and discriminator [7].For instance, it is possible to generate MNIST digits conditioned on class labels.

IV. METHODS

A. Our WGAN architecture

Working with the seemingly stable algorithm of WGAN allowed us to work with a very simple architecture for our neural networks.

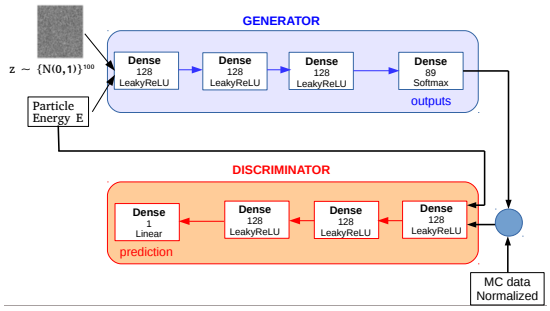

FIG. 5. Architecture of our neural networks.

This illustration (Fig. 5) is our default architecture. We tuned several parameters (number of layers, number of neurons, latent space, ...). We will dwell upon that part later in the document. Nonetheless, general comments can be done.

Generator is fed with a random vector coming from a standard normal distribution, and by the incident particle energy. The latent space size is 100, so the input vector has a shape of 101.

As outputs, the generator will produce samples of 89 normalized values (between 0 and 1). To recover real energy values in MeV and compare them to Monte-Carlo samples, we have to multiply those cells by the conditional total energy. We have chosen a *softmax* activation to enforce generated cells to lay between 0 and 1, and ensure the energy conservation.

Besides, the critic takes as inputs Monte-Carlo or generated samples, with their attached energy. An important precision is that events are considered as flat vectors: we didn't try to imply any specific geometry to the neural network. The output activation of the critic is linear since we are working with WGAN: the distance between its output for real and generated samples will tend to be as large as possible.

V. RESULTS

To compare the similarity between Monte-Carlo and generated samples, we used increasingly complex methods:

- average energy deposit in the cluster.
- mode collapse concerns are investigated by considering the nearest neighbors, and variances per cell.
- the correlation matrix.
- comparison of physics variates histograms.

A. Average energy deposit in the cluster

Firstly, we visually check that energy deposit in the cluster are matching in average (Fig. 6). Calorimeter cells are seen as pixels.

Here, we use the same energy scale in energy for the four layers of the calorimeter. The energy span goes from 10^{-2}, to 10^4. Our WGAN is able to reproduce the average shape pretty accurately, especially in the Middle which is the most energetic layer. On the other hand, we notice that energy cells with lower energy deposits (especially edge ones in the Strip en the Back) are not perfectly reproduced: for the WGAN, those cells tend to zero. This can be explained by the high span of values we have in energy, which makes it hard to learn for the WGAN. It is not able to reproduce accurately high energies and be as effective to depict variety for lower

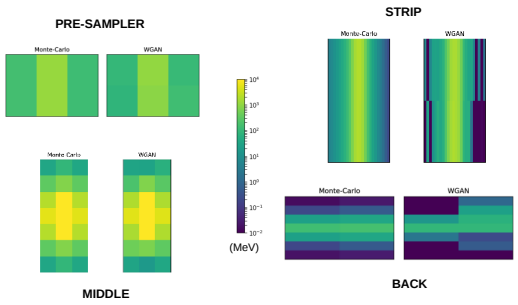

FIG. 6. Average energy deposit in the cluster.

values.

B. Nearest Neighbors

Now that we know what the average images look like in each layer, for both WGAN and GEANT generated showers, we can do another qualitative test of performance. Mode collapse is a commonly encountered failure case for vanilla GANs when the generator learns to produce samples with extremely low variety. In such a scenario the generator will exhibit very poor diversity among generated samples, which limits the usefulness of the learnt GAN. Mode collapse concerns can be investigated by considering the nearest neighbors among the training and generated datasets. By doing this, we visually check that WGAN images are not memorized samples from the training dataset (see an example Fig. 7).

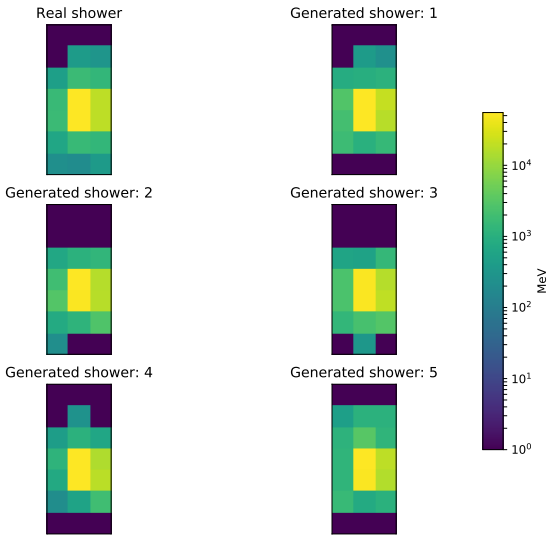

FIG. 7. One MC sample (170 GeV) and the five nearest neighbors (Euclidean distance) generated by the WGAN

C. Variance per cell

Nearest Neighbors plots show that we don't fully memorize the dataset. An other important point to check is that variance of WGAN cells are not drastically lower than Monte-Carlo samples, which would be a clue for partial mode collapse. Indeed, we can imagine that the WGAN only memorize several samples from the real distribution, which would be intractable with a simple nearest neighbors investigation. Thus, we compare variance per cell of WGAN and Monte-Carlo dataset. We see that the second order moment are of the same order of magnitude, which is reassuring towards risks of mode collapse.

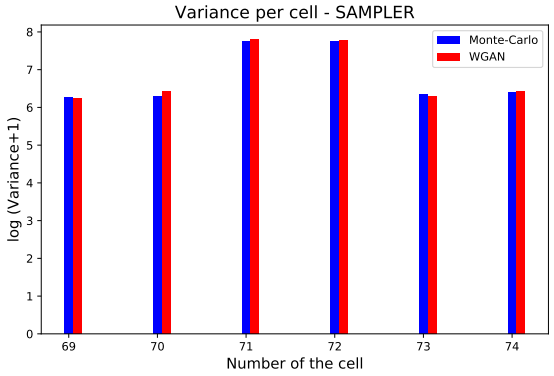

FIG. 8. Variance per cell in Pre-Sampler layer

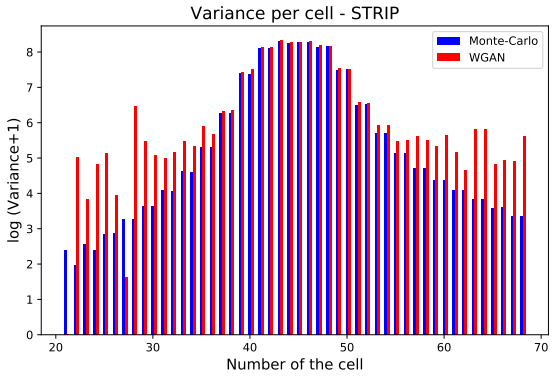

FIG. 9. Variance per cell in Strip layer

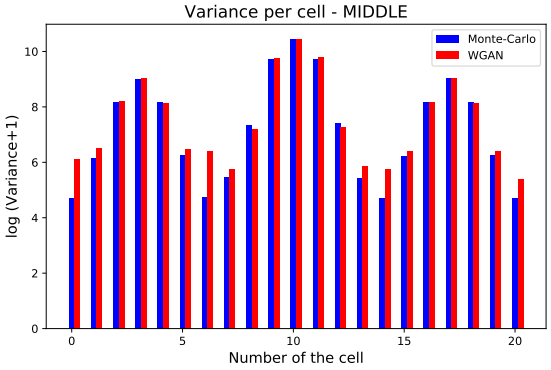

FIG. 10. Variance per cell in Middle layer

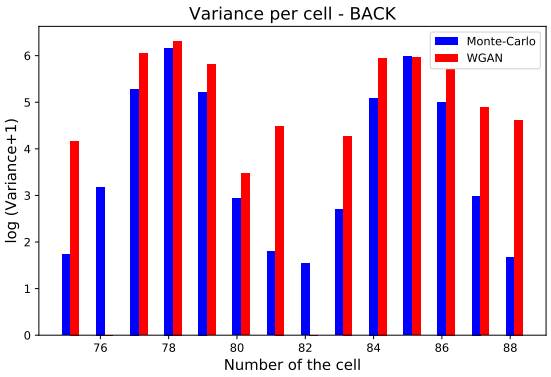

FIG. 11. Variance per cell in Back layer

D. Matrix of correlations

Finally, we compute the Pearson correlation coefficients (lies between -1 and 1) across cells of the calorimeter. Cells [0:20] corresponds to Middle, [21:69] to Strip, [70:75] to Pre-Sampler and [76:88] to Back. To give an absolute measure, we compute the mean squared error between generated correlation coefficients and MC ones 12.

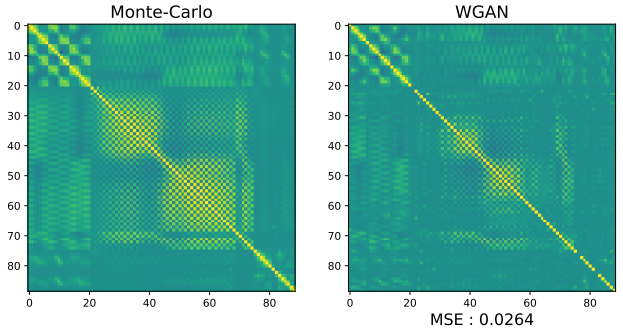

FIG. 12. Matrix of correlations between calorimeter cells

E. Additional comment: L1 regularization

To encourage sparsity, and improve our results for edge cells, a solution is to add a L1 regularization (with a weight regularization penalty of 1e-5) term in the Generator loss. This regularization term improves our results for edge cells without impacting on higher energies (Fig. 13).

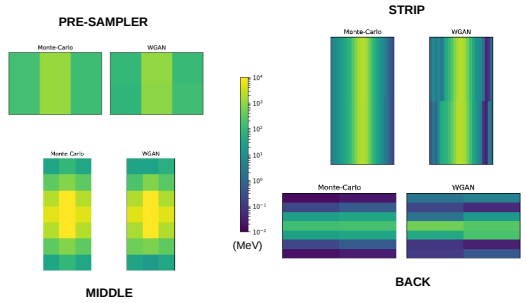

FIG. 13. With an additional L1 regularization.

However, we finally choose to apply the same energy cuts to generated events as those detailed in Section II B. We select this option because it was impossible to reproduce discontinuity of distributions without it (see Back layer of Fig. 14 for example). In this precise case, L1 regularization lose its interest for lower energy values (they are cut by our thresholds). But it can be a good alternative if one want to use directly generated events of the WGAN, without any energy thresholds.

F. Physics variates

To go further in the comparison of real and generated samples, we compute several classic physics variates. We generate exactly the same number of samples that we have in the MC dataset, and conditioned on the same energy. Those histograms are plotted across the entire dataset (40k events).
WGAN events are conditioned on the total energies of the Monte-Carlo dataset. As a result, WGAN histograms (in red) have to match as much as possible MC histograms (in blue) to be relevant.

Firstly, we compute the total energy per event in each layer Fig. 14. The non-linearity in the Back is due to our energy cuts. To be more precise, we calculate the ratio of total energy of the event that is absorbed by each layer Fig.15. We look in detail to the Middle layer: we check that the center of the cluster concentrate most of the energy per event. Thus, we compute the ratio between energy absorbed by the center (3*3 cells) over total energy absorbed by Middle layer Fig.21.
Then, we calculate physics variates that depend on coordinates (Eta and Phi): we take the same ones for both MC and generated events conditioned on equal total en-

ergy. There are averaged eta for each layer (weighted by the layer energy) Fig.16, and averaged phi Fig.17. In addition, shower width plots Fig.18 Fig.19, especially the shower width in the Middle layer w.r.t energy of the event Fig.20.

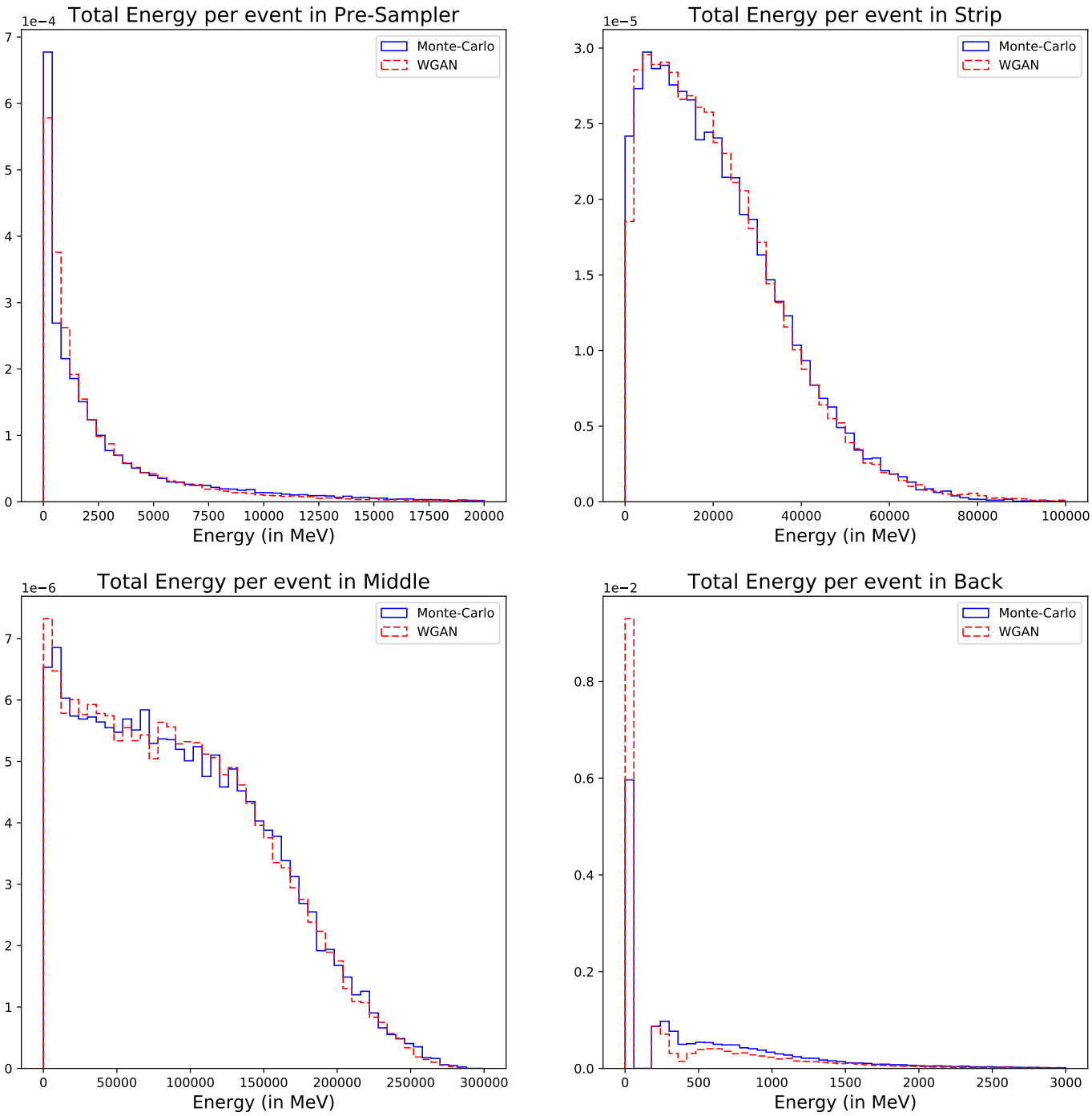

FIG. 14. Energy per layer of the calorimeter

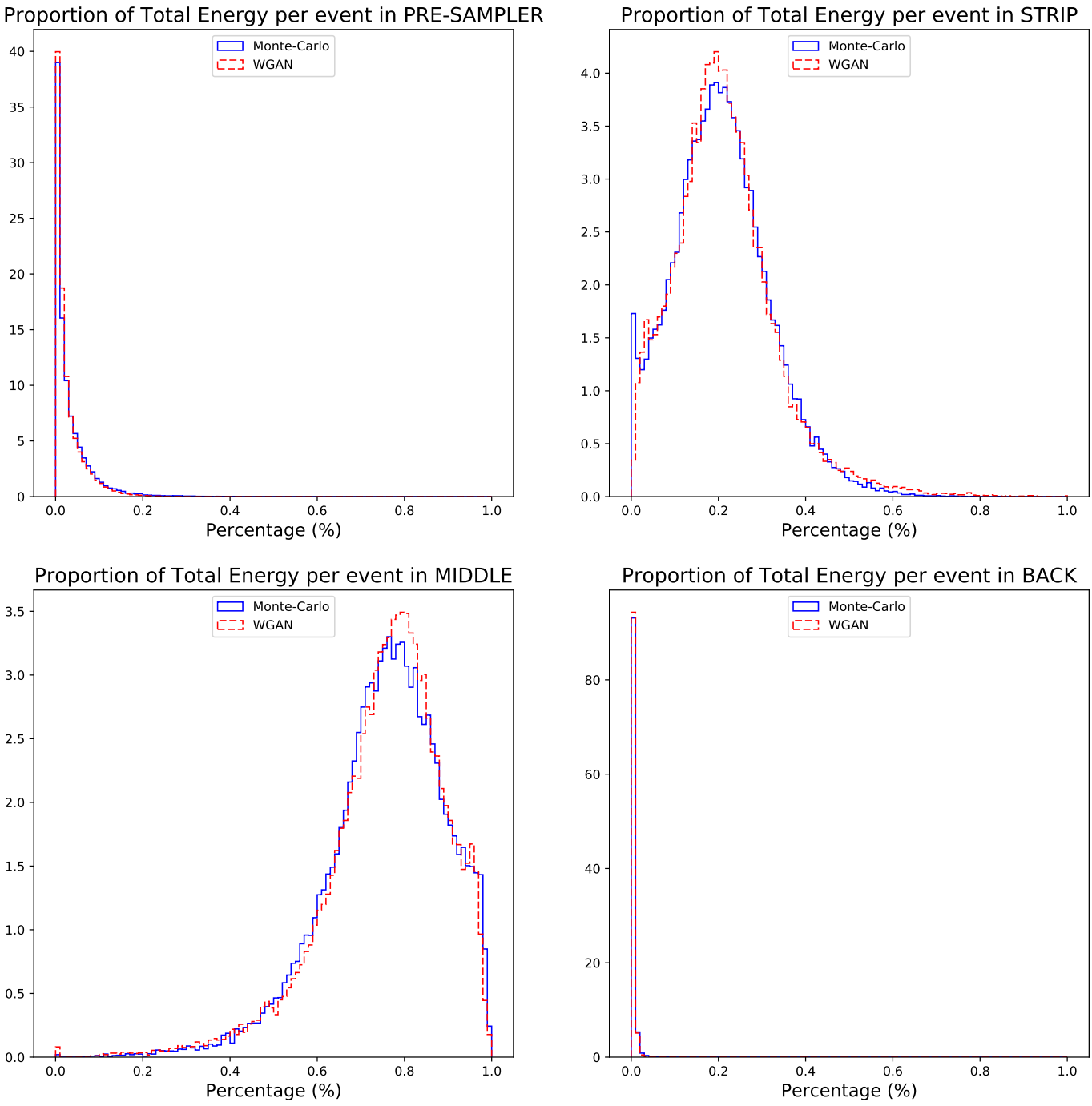

FIG. 15. Normalized energy per layer

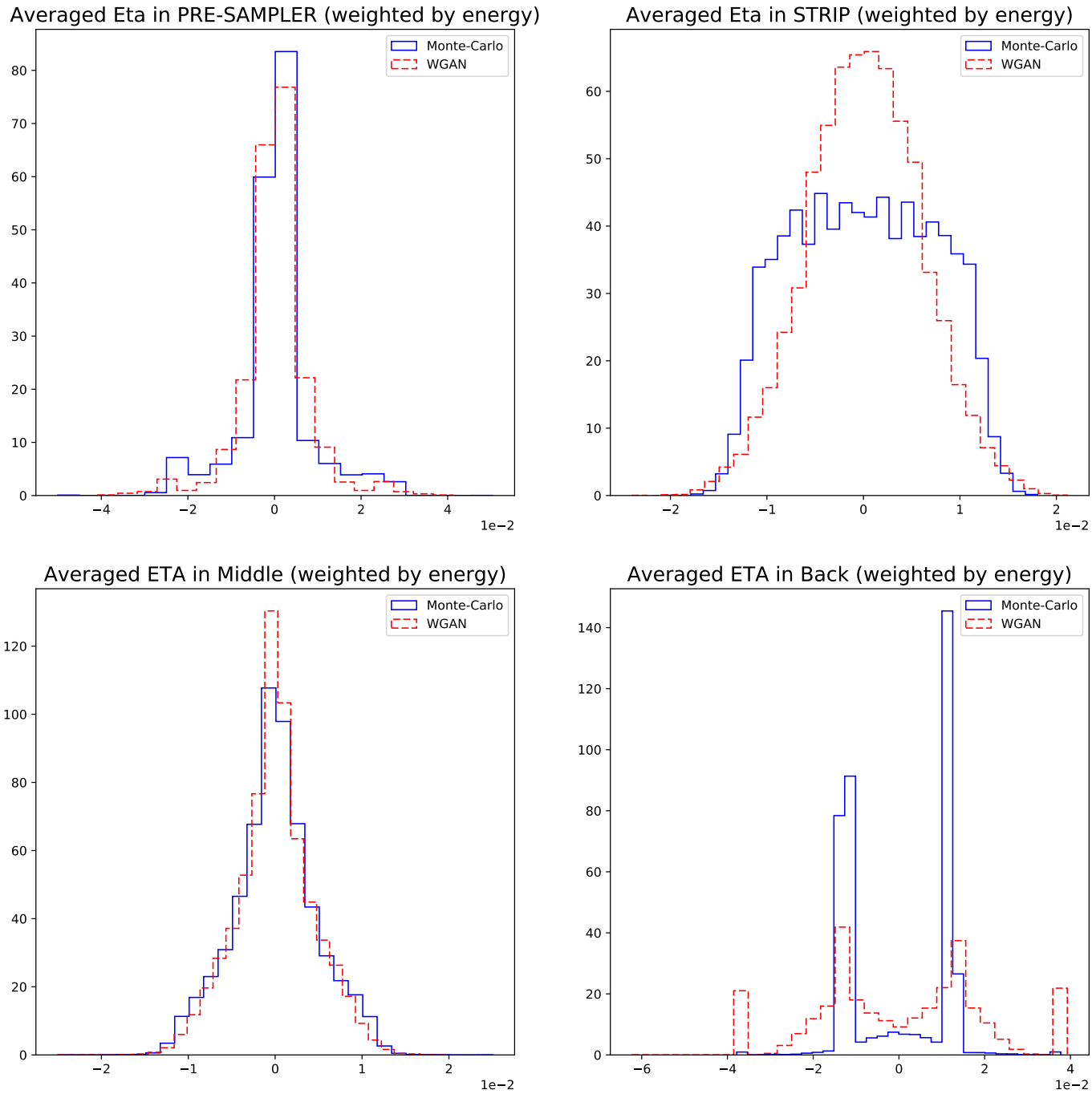

FIG. 16. Averaged Eta per layer

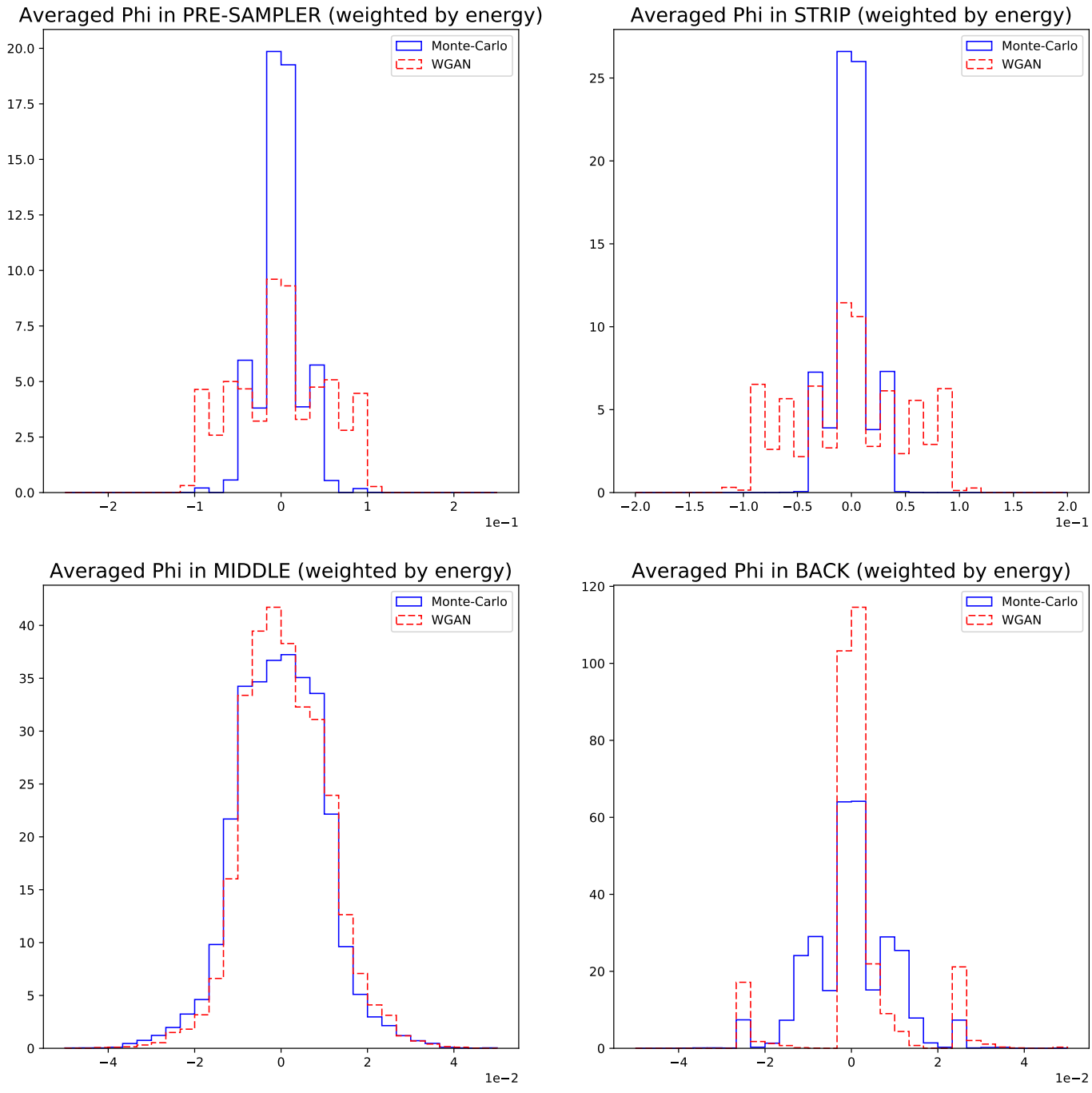

FIG. 17. Averaged Phi per layer

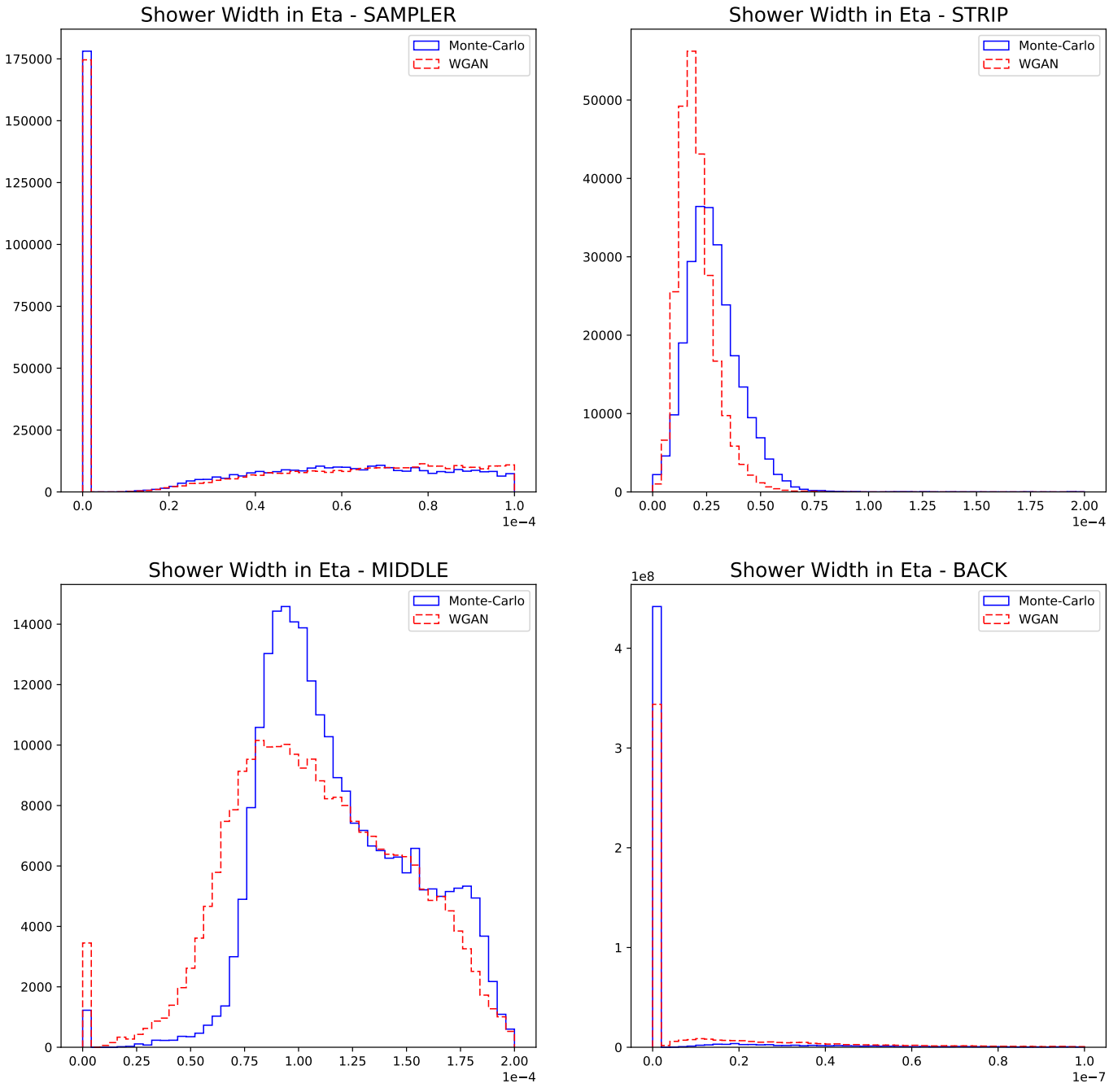

FIG. 18. Shower Width in Eta

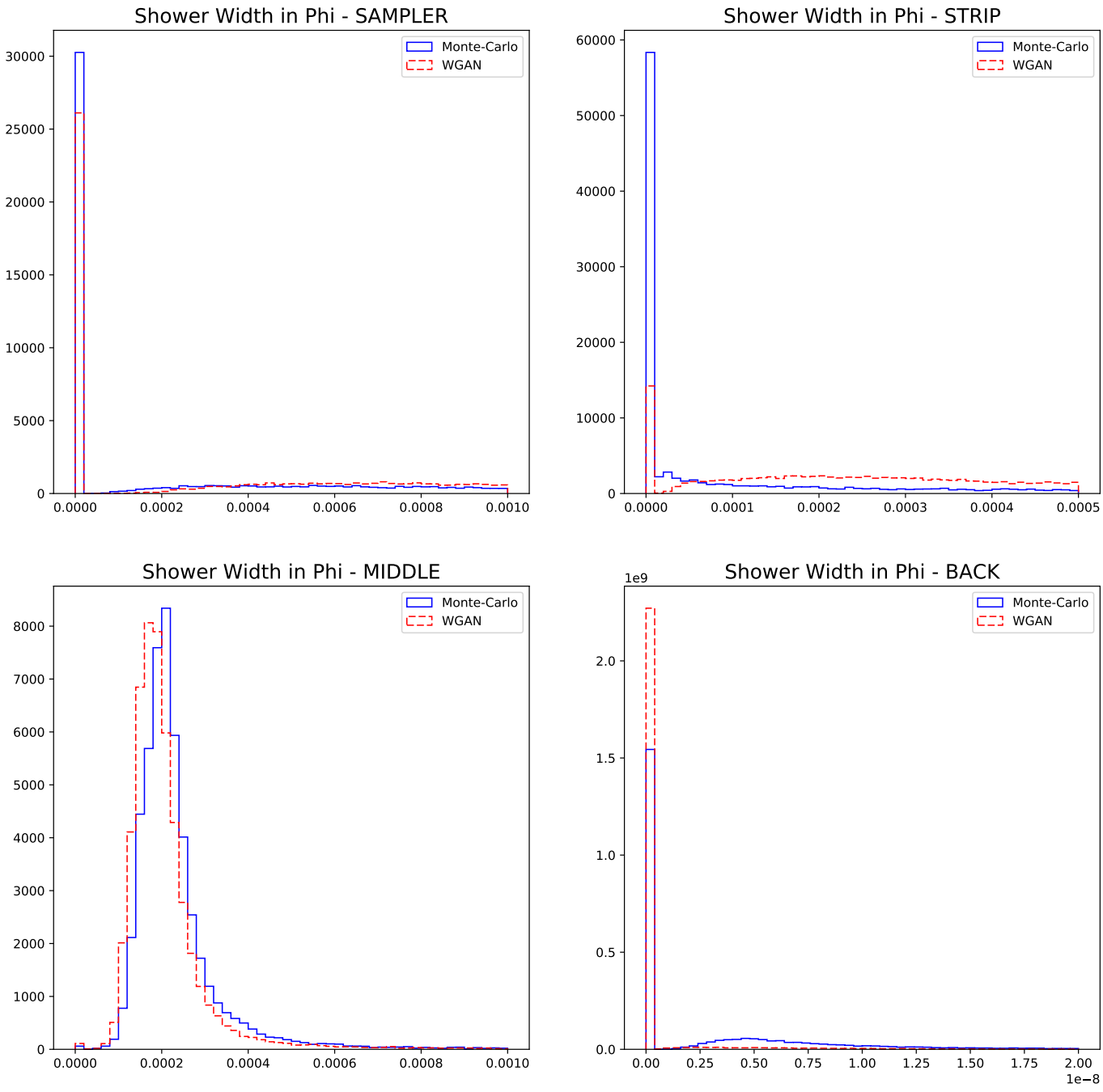

FIG. 19. Shower Width in Phi

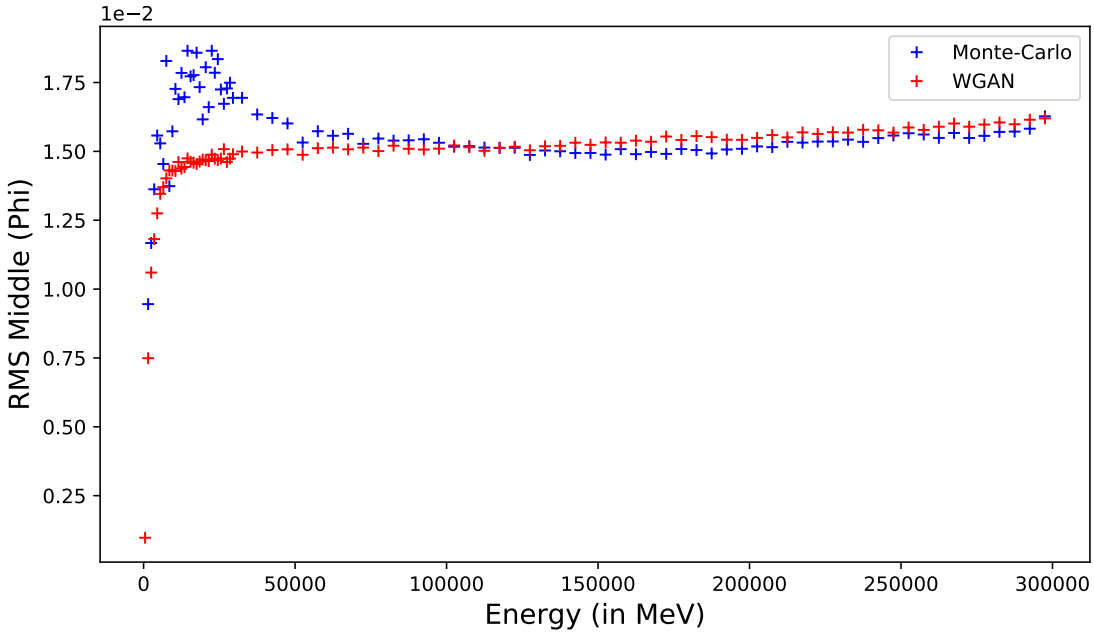

FIG. 20. Shower Width in Phi (Middle) w.r.t event total energy

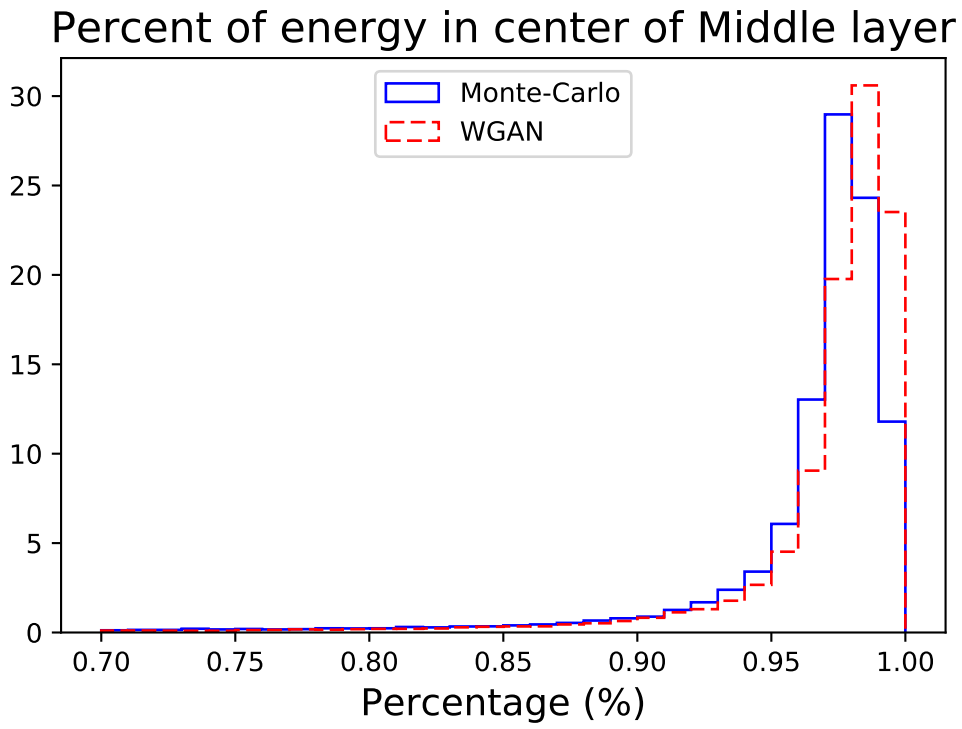

FIG. 21. Percent of energy in center (3*3) of Middle layer (3*7)

VI. INFLUENCE OF PARAMETERS

In this section, we try to tune parameters of our WGAN, and discuss influence of them over generated samples quality.

A. Reproducibility of our results

Firstly, we evaluate several times the same model to check seed influence over the variability of the trained WGAN. After the same number of epochs, we see that shapes and supports are globally the same. However we can notice an important variability over several physics variates, with the plot of Shower With across eta for instance (Fig. 22).

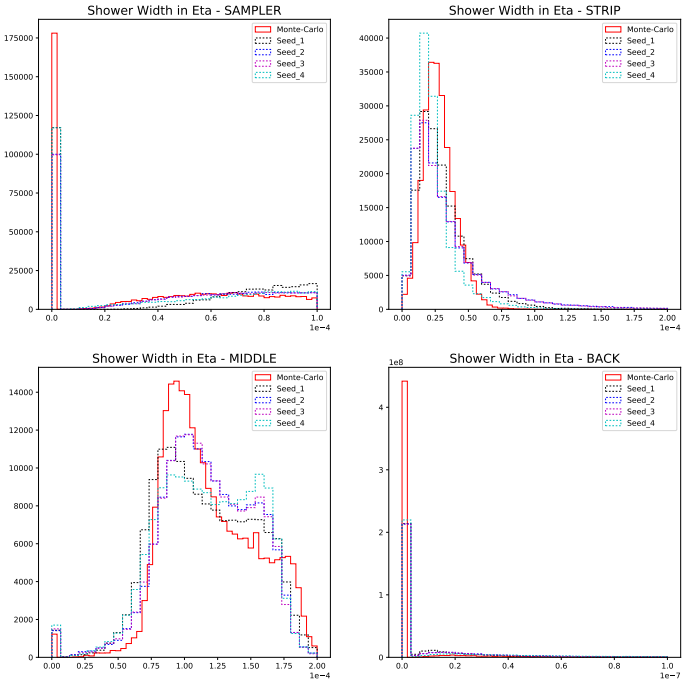

FIG. 22. Variability of Shower Width (Eta) depending on the seed

This variability can be explained by the intrinsic random aspect of a GAN. Indeed, there are random variables as inputs of the Generator network; weight values depend on initialization; even the training algorithm is partly random since it arbitrarily pick batches. Models should converge to the same solution but in practice we can glimpse that this variability has some influence on the result. Thus, we produce 5 different seeds for each model in this section, and consider the union of generated samples for following plots. It is a way to bring an average trend out, and make reliable comparisons over parameters influence.

B. Parameters tuning

We start with our default architecture and vary only one parameter at a time. By default, our WGAN has the following features:

- Numbers of layers: 3
- Number of neurons per layers: 128
- Latent space size: 100 (standard normal distribution)
- Activations : LeakyReLU, except for the last layer of the generator (softmax) and the last layer of the discriminator (linear)
- Gradient penalty weight: $\lambda = 10$ (default value of the paper [6])
- Number of epochs: 1000
- Training Ratio: 5 (number of discriminator training loops for one iteration of the generator weights)

1. *Number of layers - 1, 2, 3*

Firstly, we vary the number of layers for both generator and discriminator networks. Our goal is to see in what extent complexity of the network has a positive influence on our results.

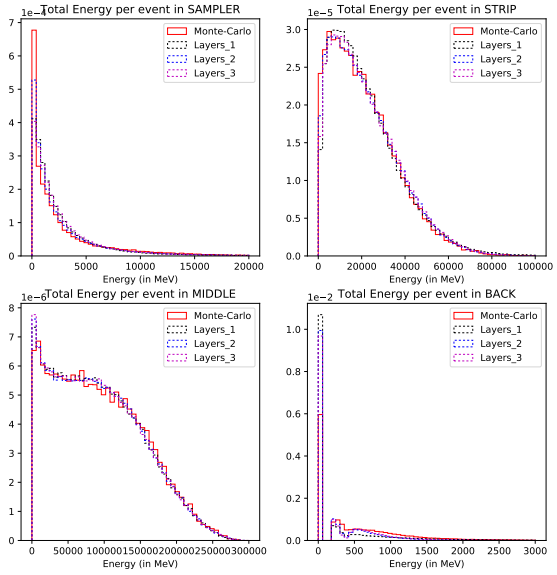

FIG. 23. Energy per layer of the calorimeter

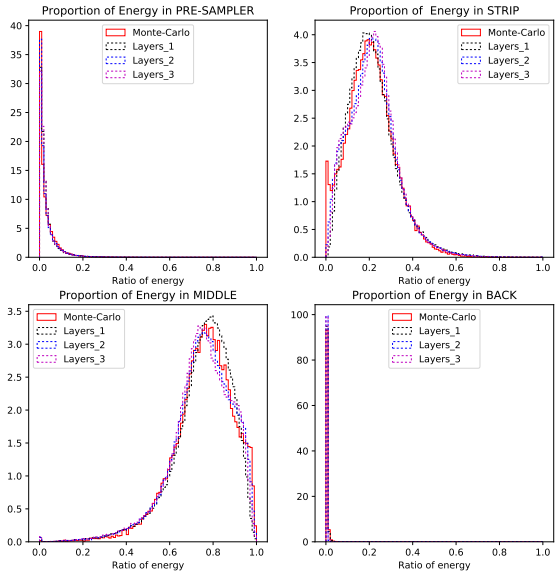

FIG. 24. Normalized energy per layer

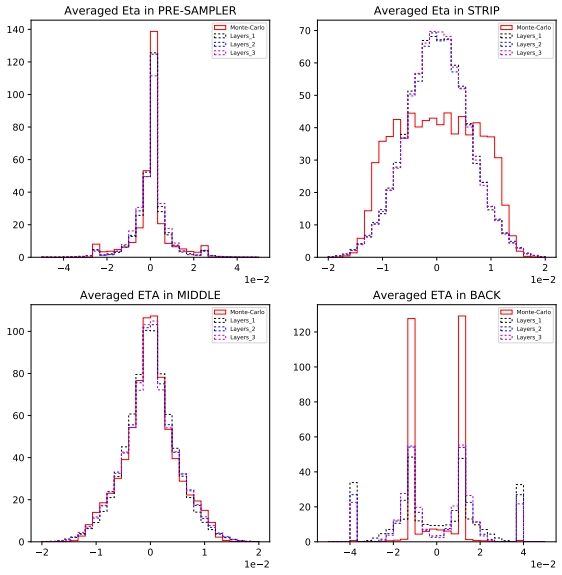

FIG. 26. Averaged Eta per layer

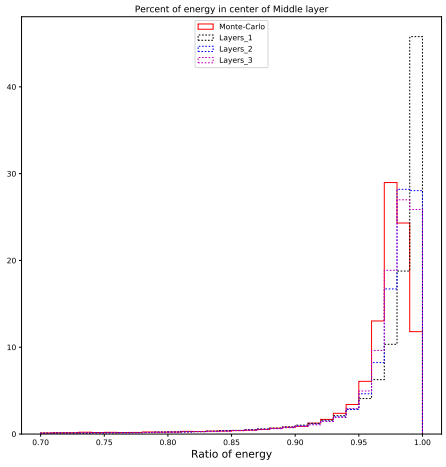

FIG. 25. Percent of energy in center (3*3) of Middle layer (3*7)

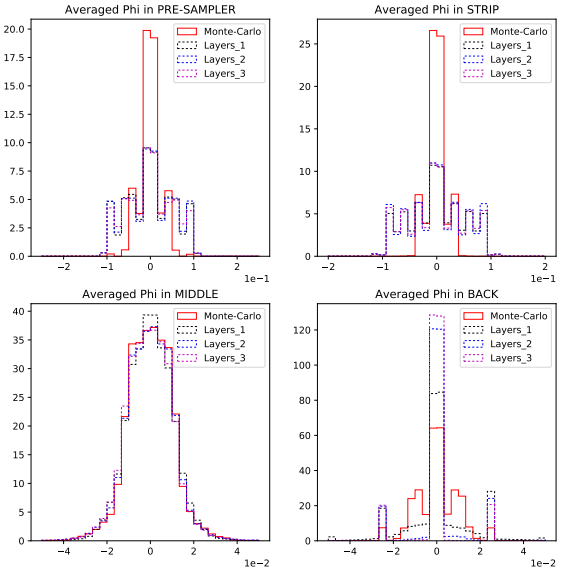

FIG. 27. Averaged Phi per layer

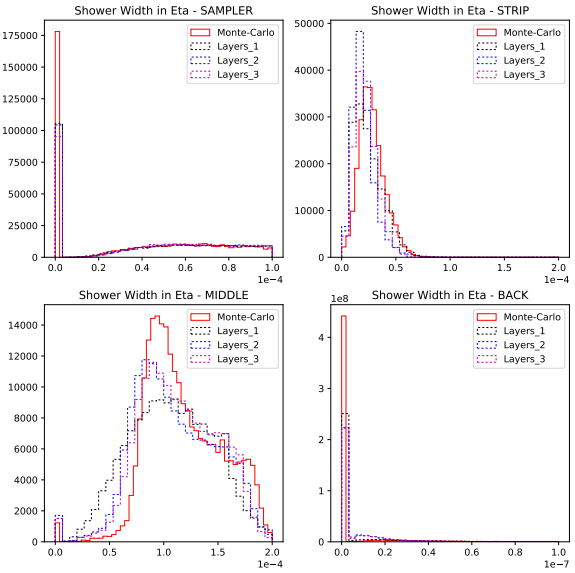

FIG. 28. Shower Width in Eta

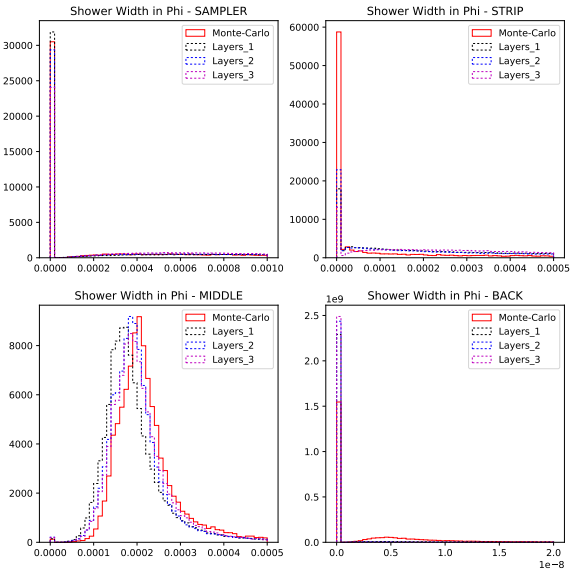

FIG. 29. Shower Width in Phi

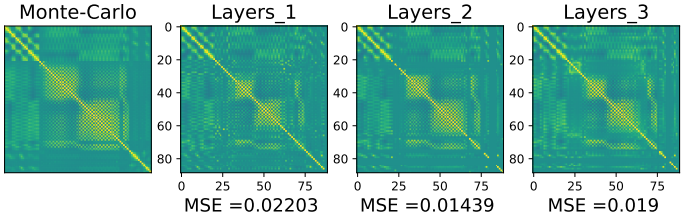

FIG. 30. Correlation Matrix

Comments: The simplest architecture - with only one layer - gives results that are satisfying, with only a few shifts compared to 2 and 3-layers architectures (ratio of energy in the center of Middle layer, Shower width in Eta and Phi in the Middle).
2 and 3-layers architectures do slightly better, but we don't glimpse significant differences between the two.
I only highlight those architectures in this report, because 4 and 5 layers perform similarly or worse than simpler frameworks.

2. Latent space size - 1, 10, 100

There is no clear understanding on how we should dimension latent space. A widespread idea is that it should have the same order of magnitude than degrees of freedom of the problem. That's why we have chosen 100 as the default value of the latent space size, since we have to generate 89 cells of the calorimeter. But this prior assumption has to be confronted with empirical results: that's why we also try with 1 and 10 latent space random variables.

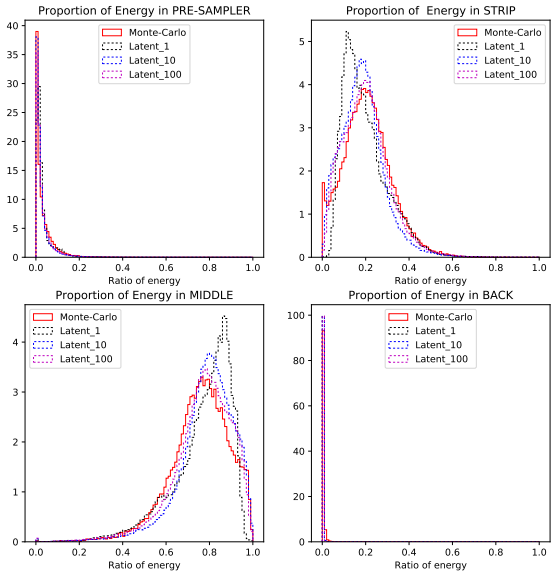

FIG. 32. Normalized energy per layer

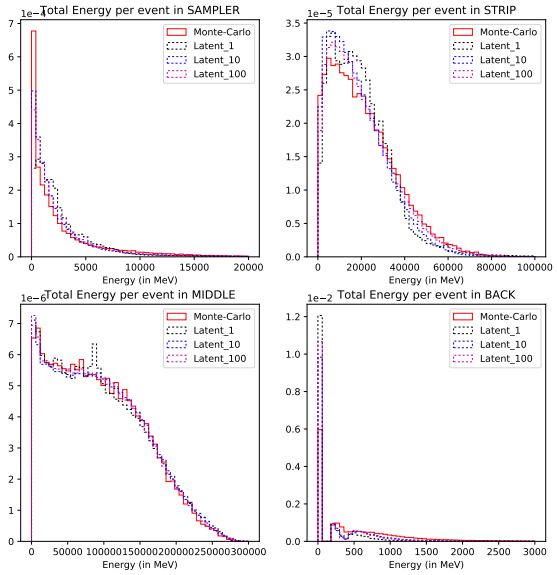

FIG. 31. Energy per layer of the calorimeter

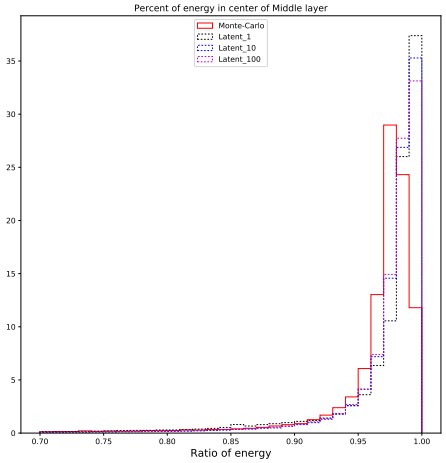

FIG. 33. Percent of energy in center (3*3) of Middle layer (3*7)

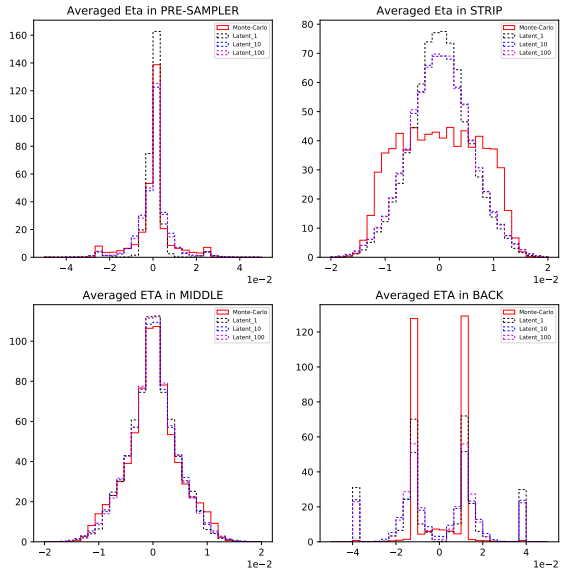

FIG. 34. Averaged Eta per layer

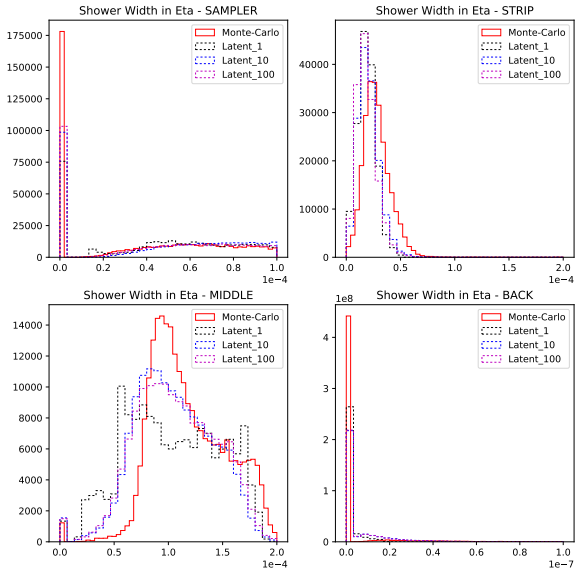

FIG. 36. Shower Width in Eta

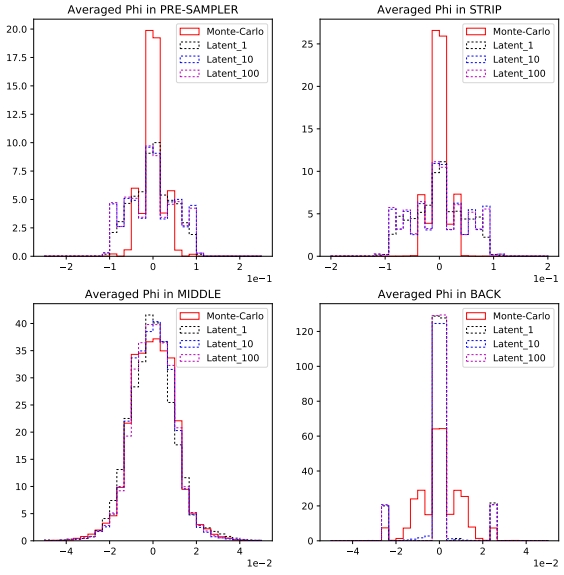

FIG. 35. Averaged Phi per layer

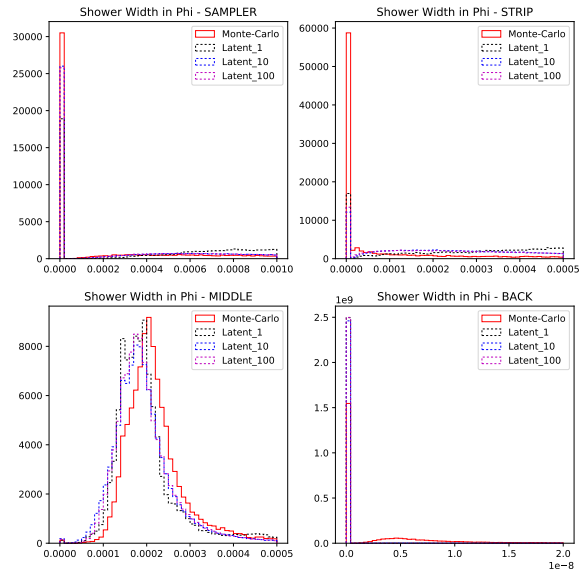

FIG. 37. Shower Width in Phi

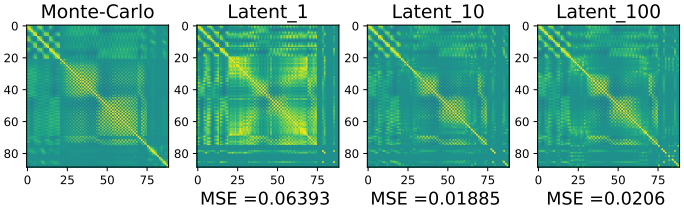

FIG. 38. Correlation Matrix

Comments: Influence of latent space size is not so pronounced: a latent space size of 1 is able to catch globally the good support and shapes. That can be explained by the fact that for a single random variable (and in a lesser extent for 10), the first layer of the neural network plays the role of a new latent space, of shape 128. The single random variable is mapped on the 128 neurons of the first layer, and the outputs go through the two remaining layers. We saw in the previous subsection that we had good results with only two layers of neurons, so the current result is not surprising.
We tried with a higher dimensioned vector of shape 250 in the latent space, but it gave poorer results (maybe it would have required a longer training).

3. Training Ratio - 1, 5, 10

In the former article of Goodfellow et al. ([5]), the discriminator should theoretically be optimal for each iteration of the generator weights. In practice, with a stable algorithm such as WGAN, a few iterations for the Discriminator can approximate pretty closely the optimal one.

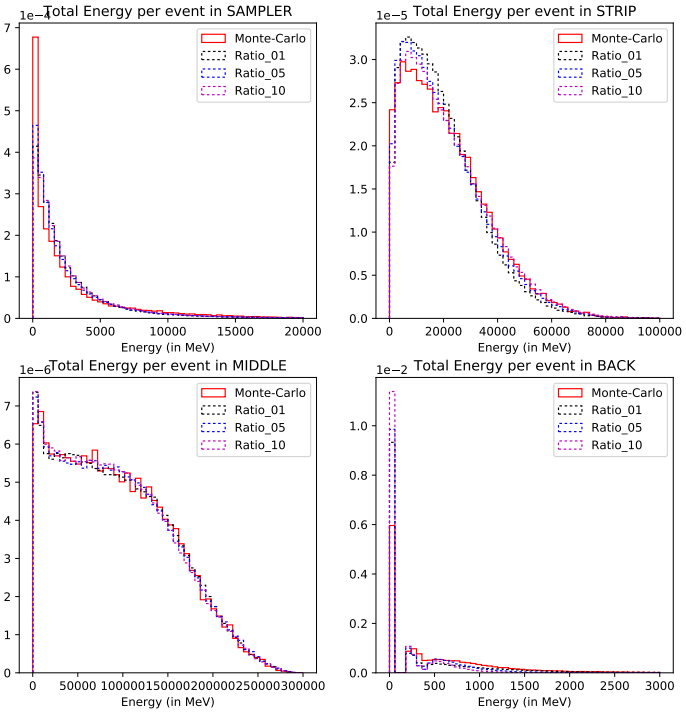

FIG. 39. Energy per layer of the calorimeter

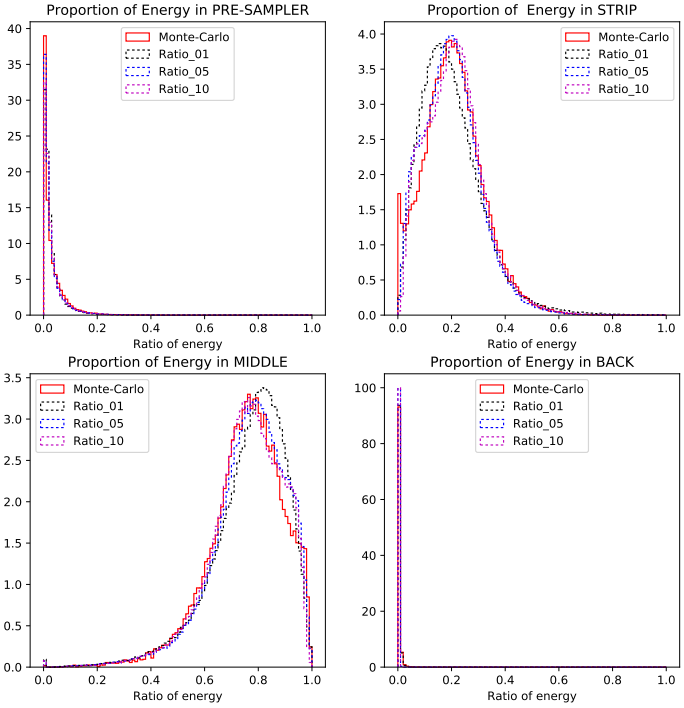

FIG. 40. Normalized energy per layer

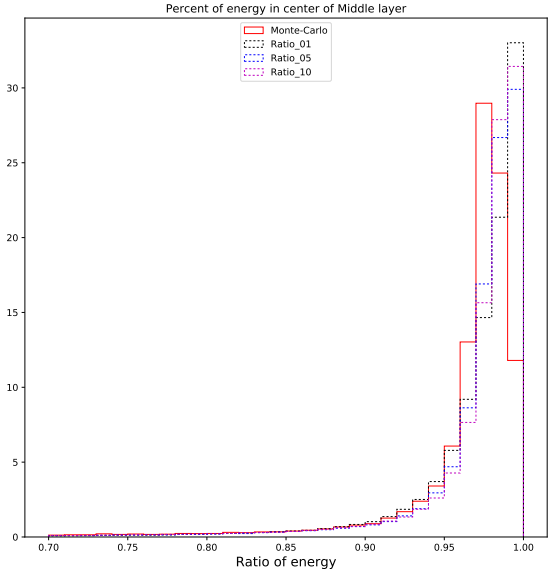

FIG. 41. Percent of energy in center (3*3) of Middle layer (3*7)

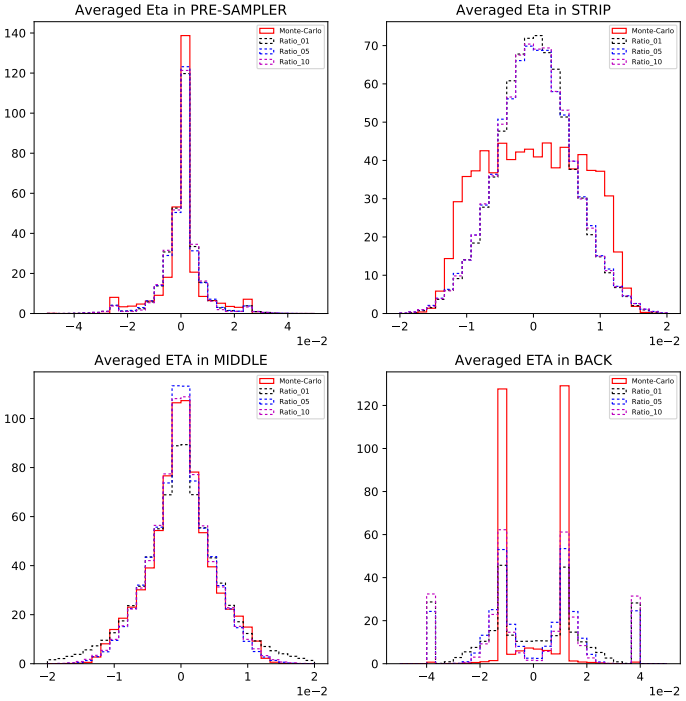

FIG. 42. Averaged Eta per layer

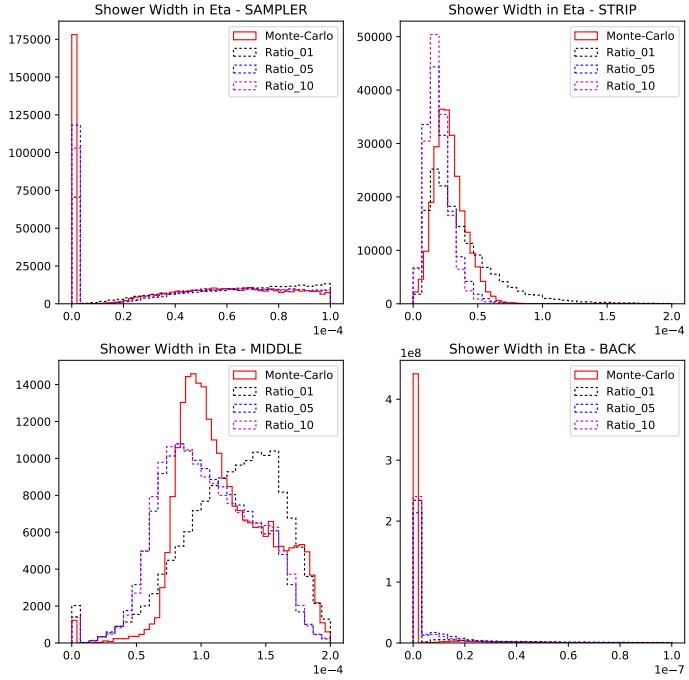

FIG. 44. Shower Width in Eta

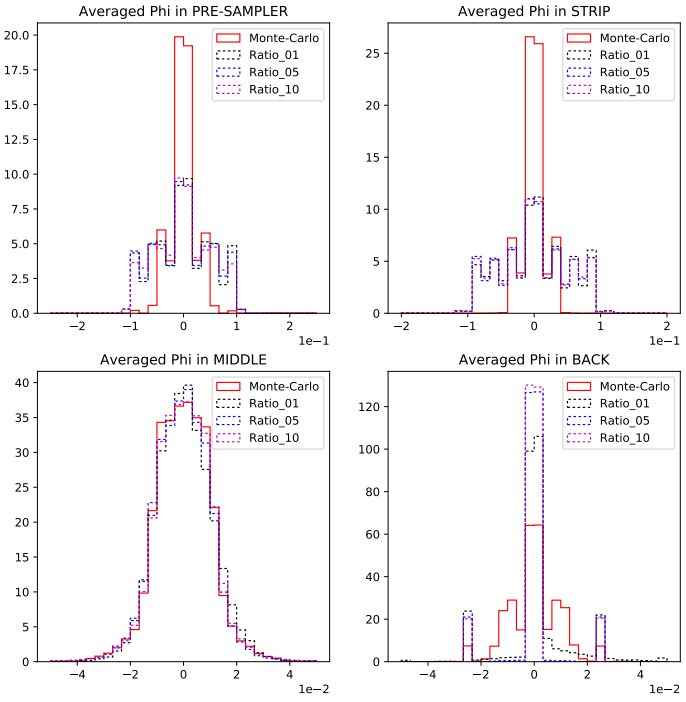

FIG. 43. Averaged Phi per layer

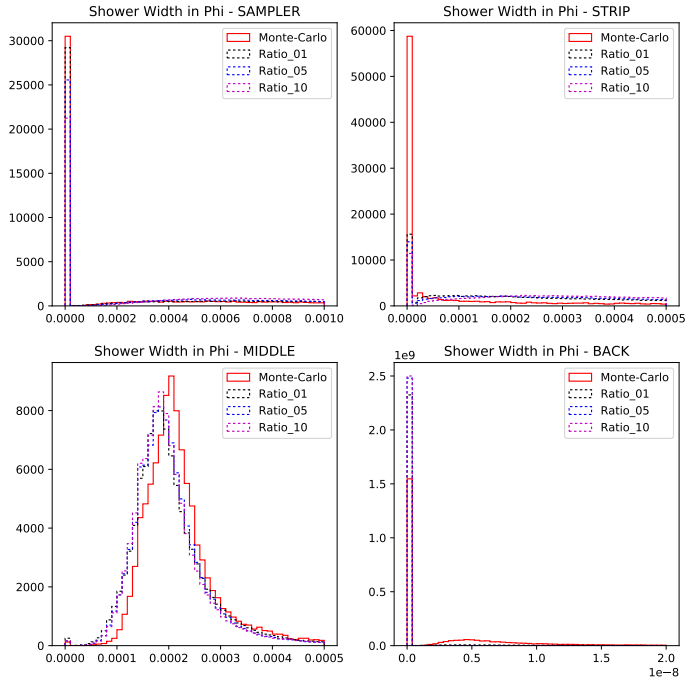

FIG. 45. Shower Width in Phi

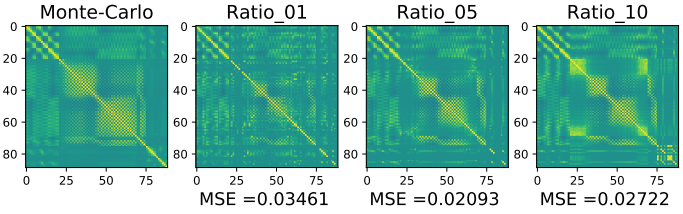

FIG. 46. Correlation Matrix

Comments: If a training ratio of 5 (default) and 10 give very similar results, a training ratio of 1 is less efficient (normalized energy per layer, Shower Width in Middle). The default parameter seems relevant.

4. Gradient penalty - 0, 10

As mentioned before, we use the algorithm of Improved Wasserstein GANs described in [6] to train our networks. We compare plots generated with $\lambda = 0$ (without gradient penalty), and the framework with *lambda* $= 10$ (default value in the paper).

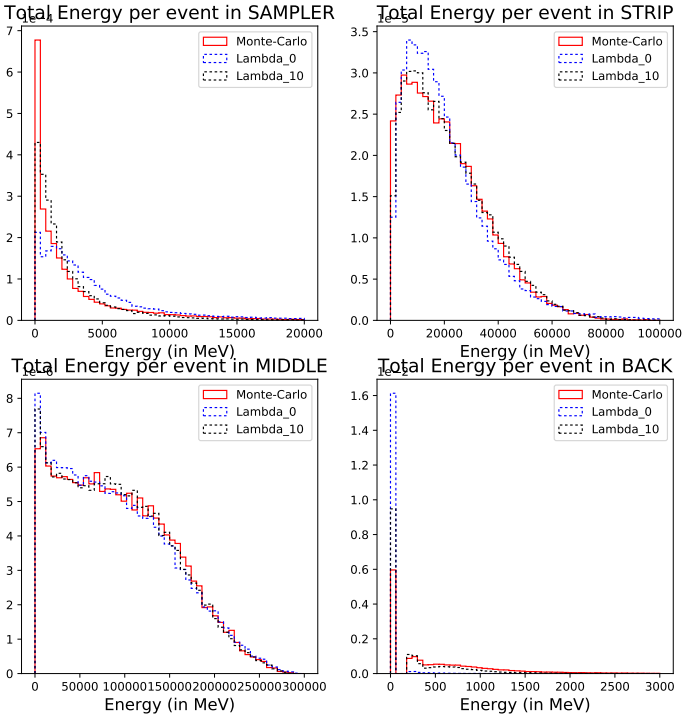

FIG. 47. Energy per layer of the calorimeter

Comments: We see that the gradient penalty is compulsory to make WGAN converge: for $\lambda = 0$, generated distribution is far from the real one compare to *lambda* $= 10$ distribution. An alternative would be to test with weight clipping instead of gradient penalty.

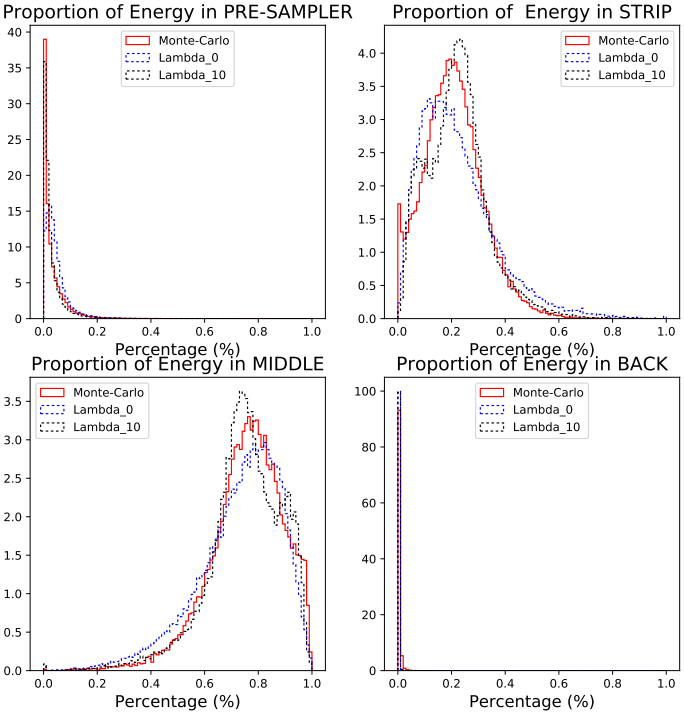

FIG. 48. Normalized energy per layer

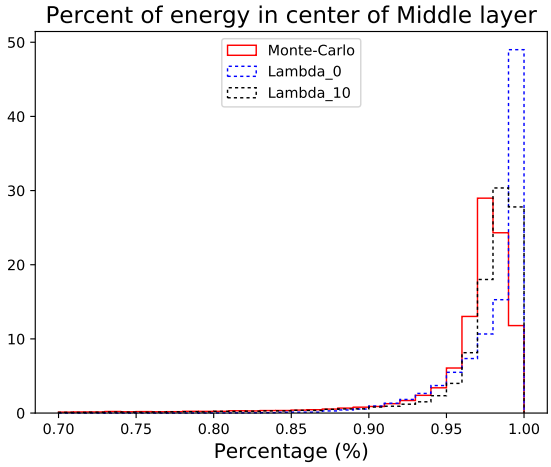

FIG. 49. Percent of energy in center (3*3) of Middle layer (3*7)

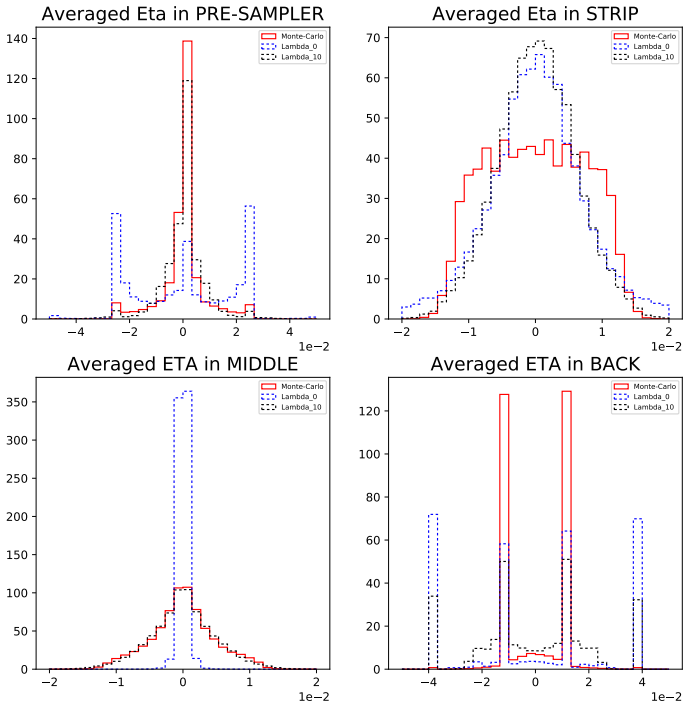

FIG. 50. Averaged Eta per layer

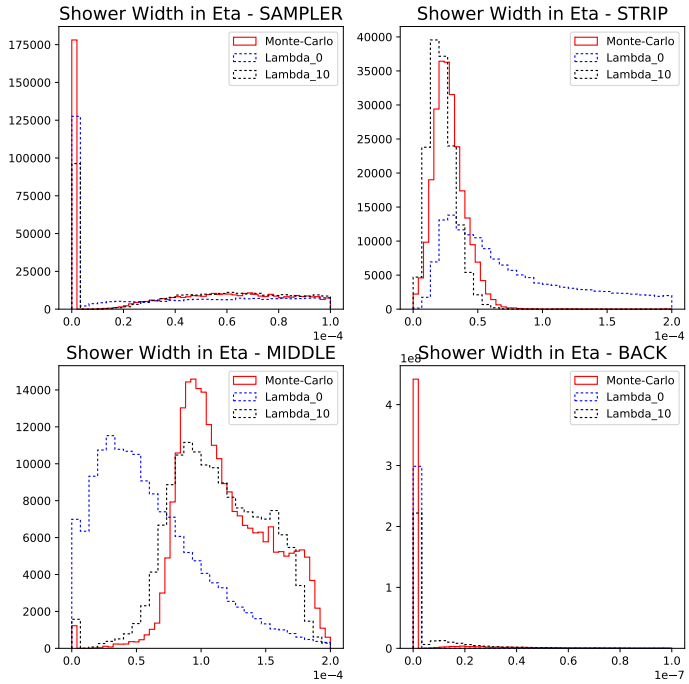

FIG. 52. Shower Width in Eta

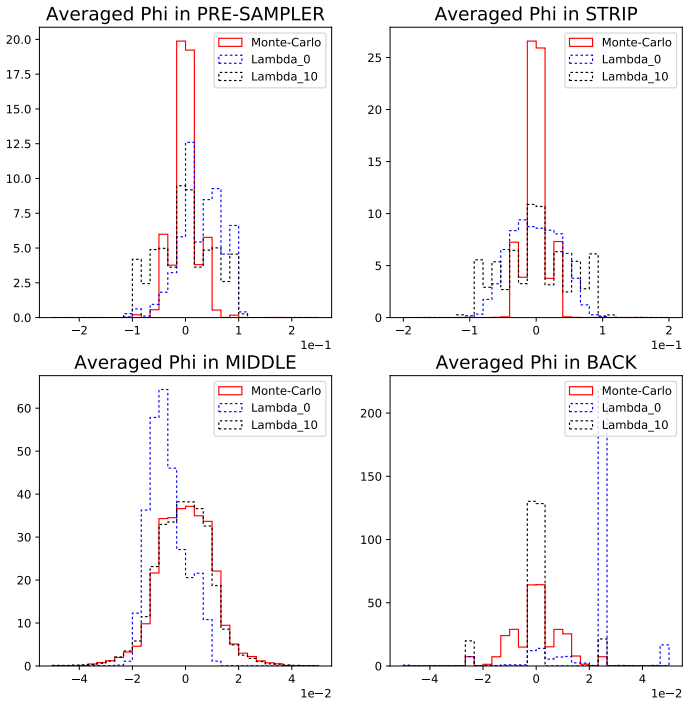

FIG. 51. Averaged Phi per layer

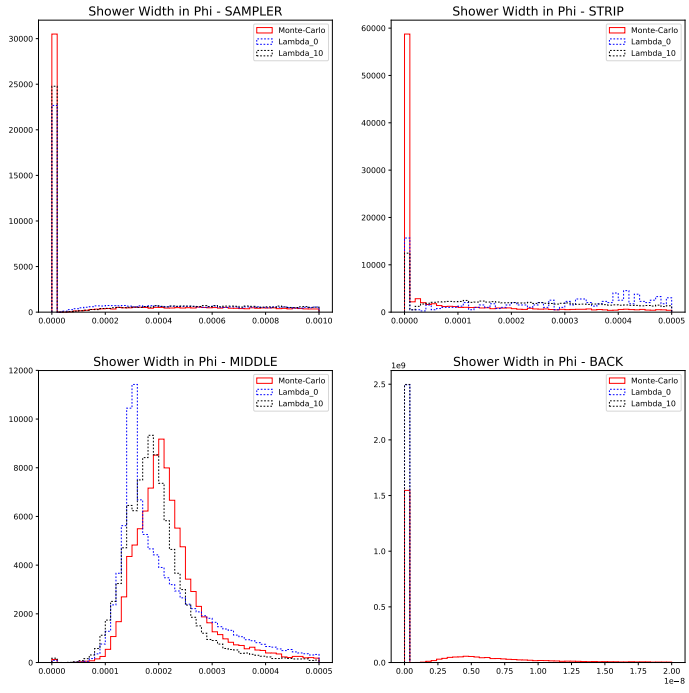

FIG. 53. Shower Width in Phi

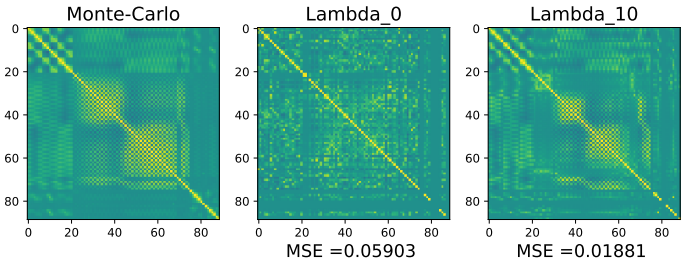

FIG. 1. Correlation Matrix

5. Results on a larger cluster

Considering our encouraging results on a small cluster of 89 cells, we test this approach on a larger one: 247 cells in total (14 in Pre-Sampler, 112 in Strip, 77 in Middle and 44 in Back. We keep exactly the same architecture to see if it was reliable on a slightly different problem.

A. First approach with our default WGAN

We use the same Dense architecture of WGAN on this larger cluster. We observe a degradation of performance over energy distributions, but networks parameters have not be tuned for this new input space.

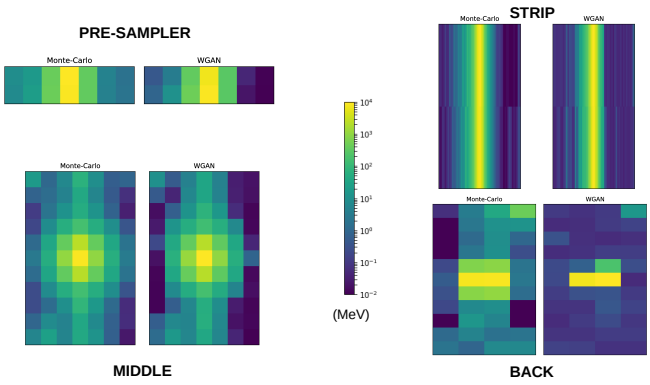

FIG. 2. Energy per layer of the calorimeter

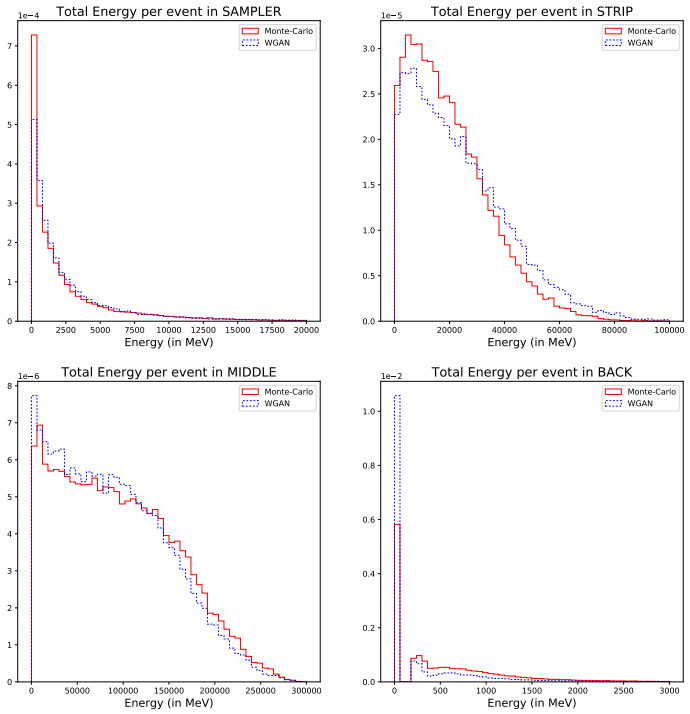

FIG. 3. Energy per layer of the calorimeter

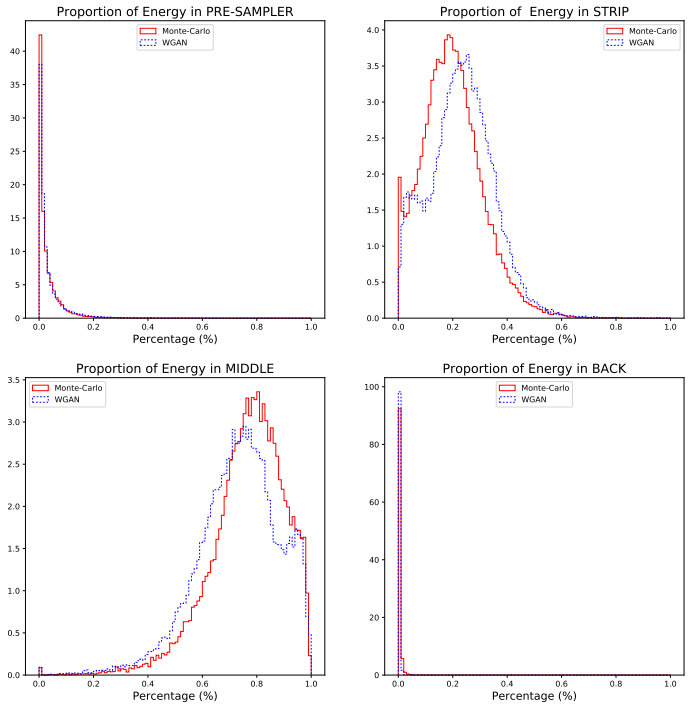

FIG. 4. Normalized energy per layer

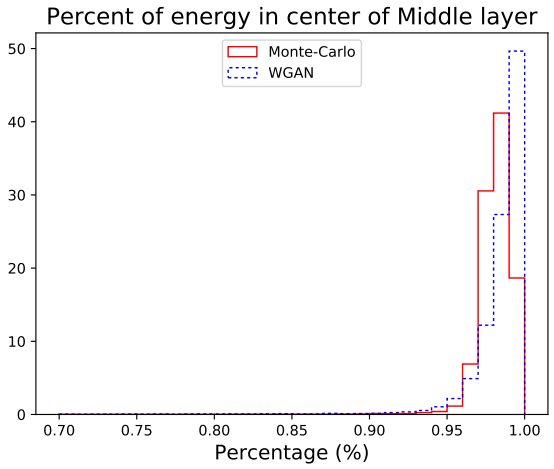

FIG. 5. Percent of energy in center (3*3) of Middle layer (7*11)

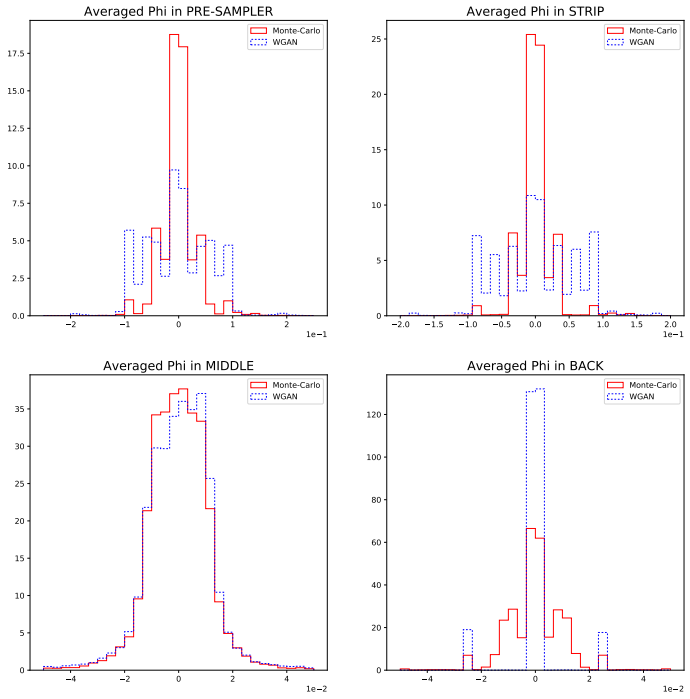

FIG. 7. Averaged Phi per layer

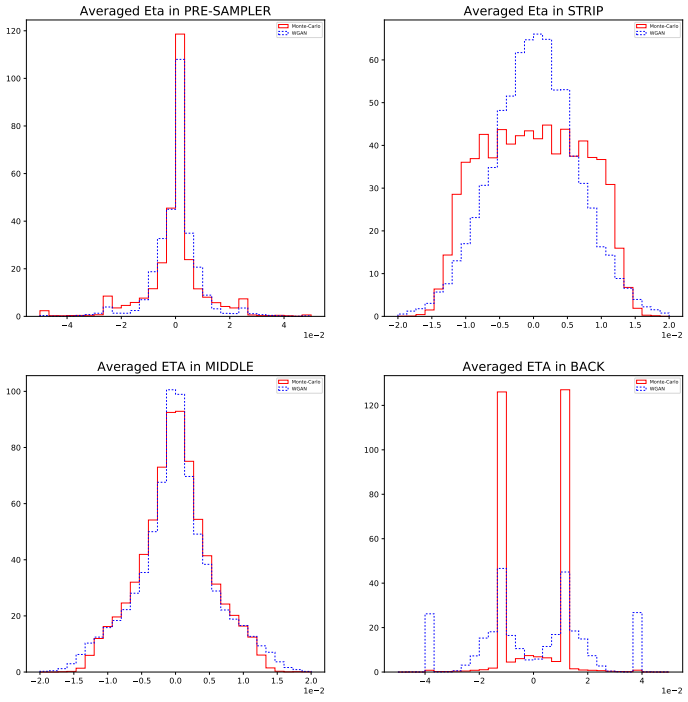

FIG. 6. Averaged Eta per layer

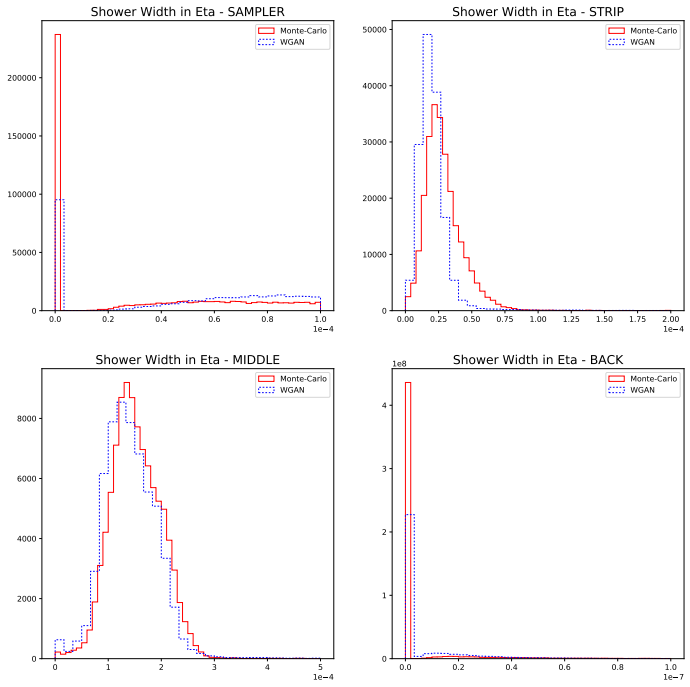

FIG. 8. Shower Width in Eta

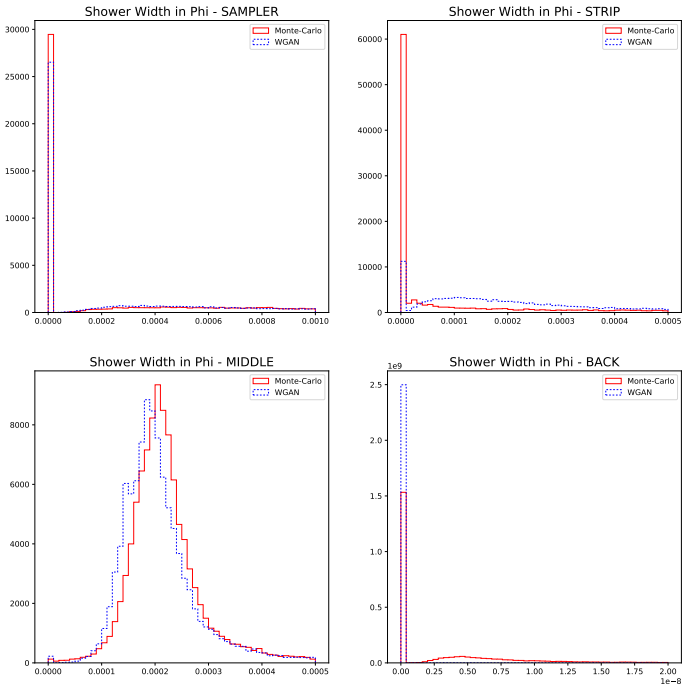

FIG. 9. Shower Width in Phi

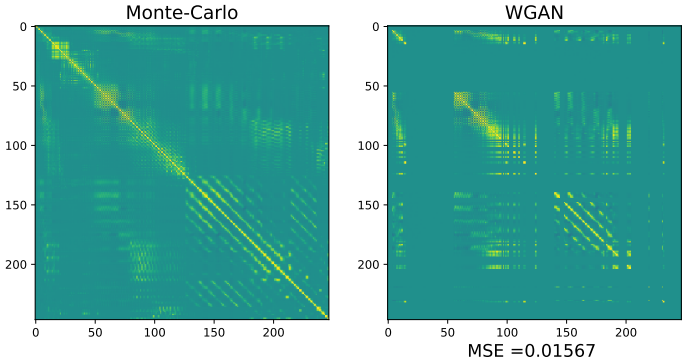

FIG. 10. Correlation Matrix

Comments: Our default framework manages to learn pretty accurately this new dimensioned dataset: it is an encouraging proof of its ability to generalize its learning strategy.
A consequence of using this larger cluster is that we remove the clustering effect we had for the Shower Width in Phi (Middle layer). Now Monte-Carlo distribution has a Gaussian shape for this variate, which is easier to learn with simple dense layers.
However, there is still work to do, for instance on the normalized energy per layer, that is shifted in both Strip and Middle layers. Moreover, the global aspect of generated pixels is not totally satisfying.

B. Condition on true pt

To go further, one should try to condition on $true\_energy = true\_pt * cosh(true\_eta)$, instead of the total energy of the event. By doing this, we want to have access to higher level physics variates and key plots like $(true\_energy - reco\_energy)$ with respect to $true\_energy$, where reco_energy is the total energy absorbed by calorimeter cells.

We lacked of time to make it run properly, but it should succeed without to many tunings. One could "normalize" energy cells by dividing their real values by true_pt. However, the softmax activation for the generator may not be ingenious since the sum of the normalized cells will not longer be equal to 1.

VII. CONCLUSION AND FUTURE OUTLOOK

This project gives an additional proof of concept that GANs could accurately describe a complex phenomenon such as electromagnetic showers in the ATLAS calorimeter, and offers a complementary approach to the CaloGAN paper [9]. With a really simple architecture (only dense layers), we obtain significant results, especially on variates only tied on energies (Fig. 14, and Fig. 15). We reproduce accurately first order moment (average energy deposits, Fig. 6), and also a reasonable learning of second order moment (variance per cell: Fig. 8, Fig.9, Fig.10, Fig. 11, and matrix of correlations: Fig.12). We checked that there were no mode collapse. The WGAN algorithm allowed us to have a reasonable complexity (a model can be trained in an hour on CPU), and freed us from most heuristics.

There is still progress to make for higher levels variates (depending on coordinates), especially the shower width in eta (Fig. 19), and averaged eta and phi weighted by the energy (Fig. 16 and Fig. 17). Using convolutions could be a possible solution, but I had some troubles with CUDA and did not have access to a grid, which would have make the training tedious.

Future work will probably have to discuss methods to quantify best generative models methods (several groups work independently of a very similar project, some using others frameworks such as Variational AutoEncoder). Solutions have been mentioned like use a classifier to distinguish real and generated samples, and choose the method with the best score (inputs could be cells energies and incident particle energy, physics variates). There are also some proposals in the literature to quantify the agreement between the original and the generated distributions, for example with a statistical test [11]. An other possibility would be to use FastCaloSim tests on a trained GAN and see how it performs compared to the fast simulation.

VIII. AVAILABLE CODE

Models have been written with Keras. An example of Python code is available on Github (`https://github.com/polklin/CERN_project`), with a Jupyter Notebook.

[1] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. *ArXiv e-prints* (Jan. 2017).

[2] Collaboration, G. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 506*, 3 (2003), 250 – 303.

[3] de Oliveira, L., Paganini, M., and Nachman, B. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. *ArXiv e-prints* (Jan. 2017).

[4] Goodfellow, I. NIPS 2016 Tutorial: Generative Adversarial Networks. *ArXiv e-prints* (Dec. 2017).

[5] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. *ArXiv e-prints* (June 2014).

[6] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved Training of Wasserstein GANs. *ArXiv e-prints* (Mar. 2017).

[7] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *CoRR abs/1411.1784* (2014).

[8] Mustafa, M., Bard, D., Bhimji, W., Al-Rfou, R., and Lukić, Z. Creating Virtual Universes Using Generative Adversarial Networks. *ArXiv e-prints* (June 2017).

[9] Paganini, M., de Oliveira, L., and Nachman, B. CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks. *ArXiv e-prints* (May 2017).

[10] Ravanbakhsh, S., Lanusse, F., Mandelbaum, R., Schneider, J., and Poczos, B. Enabling Dark Energy Science with Deep Generative Models of Galaxy Images. *ArXiv e-prints* (Sept. 2016).

[11] Sutherland, D. J., Tung, H.-Y., Strathmann, H., De, S., Ramdas, A., Smola, A., and Gretton, A. Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy. *ArXiv e-prints* (Nov. 2016).