Generative Adversarial Networks for fast simulation

Federico Carminati[1], Gulrukh Khattak[1], Vladimir Loncar[2], Thong Q Nguyen[3], Maurizio Pierini[1], Ricardo Brito Da Rocha [1], Konstantinos Samaras-Tsakiris[1], Sofia Vallecorsa[1] and Jean-Roch Vlimant[3]

[1] CERN, 1 Esplanade des Particules, Geneva, Switzerland
[3] California Institute of Technology,1200 E. California Blvd., Pasadena, CA 91125, USA
[2] Institute of Physics Belgrade, University of Belgrade, Pregrevica 118, 11080, Belgrade, Serbia

Abstract. Deep Learning techniques are being studied for different applications by the HEP community: in this talk, we discuss the case of detector simulation. The need for simulated events, expected in the future for LHC experiments and their High Luminosity upgrades, is increasing dramatically and requires new fast simulation solutions. Here we present updated results on the development of 3DGAN, one of the first examples using three-dimensional convolutional Generative Adversarial Networks to simulate high granularity electromagnetic calorimeters. In particular, we report on two main aspects: results on the simulation of a more general, realistic physics use case and on data parallel strategies to distribute the training process across multiple nodes on public cloud resources.

1. Introduction and Related Work

High Energy Physics (HEP) relies heavily on Monte Carlo simulation in order to model complex processes. However the detailed Monte Carlo approach is both time and resource intensive: currently more than 50% of the Worldwide LHC Grid [1] resources are devoted to simulation [2]. The need in terms of simulated data is expected to increase by a factor $100\times$ for the High Luminosity LHC [3] and fast simulation alternatives are being investigated. Existing techniques, based, for example, on parametrization [4, 5, 6] can reach different speed-ups while retaining various levels of accuracy.

Detailed simulation of high granularity calorimeters is particularly time-consuming, however, their output, a pattern of energy depositions in the different cells, can be interpreted as pixel intensities in a three-dimensional image. Particle characteristics, such as its type, energy and incident angle are input to the simulation process and can be used to condition the training of Deep Neural Networks and obtain models capable of generating the desired output according to a specific set of input parameters. Image generation is an important aspect of machine learning: a number of approaches exists including Generative Adversarial Networks (GAN). GANs implement the idea of adversarial training for generating sharp and realistic images [7]; their application to HEP simulation was introduced by the LAGAN [8] and the CaloGAN [9] models. In this case particle showers for simplified calorimeters were simulated as sets of two-dimensional images. Since then, several studies have tested the application of GANs to HEP simulations (for example [10, 11]).

Our previous work [12, 13] introduced the simulation of an example of high granularity calorimeter using true 3D convolutions to further exploit the correlations in the volumetric space. We demonstrated the benefits of such an approach by training a network to generate calorimeter energy showers according to the primary particle energy entering the calorimeter at a fixed (orthogonal angle). Here we generalise these results to a more realistic use case considering a particle entering the detector with a variable incident angle. Thus, our network learns a joint distribution of both the primary energy and the incident angle, it reproduces a detector volume that is 4$\times$ larger and it introduces domain knowledge in order to reach a high level of accuracy. We also present updated results on the data parallel approach we used to train our network on distributed systems. Our final goal is to prove that, by using meta-optimization and hyper-parameters scans, it is possible to tune the network architectures to simulate different detectors. In this perspective, an efficient training process becomes essential and the accent should therefore be on optimizing the computing resources needed to train the networks, studying parallelization and cross-platform development. This report is organised as follows: firstly we introduce our 3D convolutional GAN model, the data set and the training strategy. Section 3 summarises some example results obtained from the validation of physics performance. Section 4 discusses the implementation of a data parallel training approach together with preliminary scaling benchmarks on public cloud resources. We then conclude with a brief outlook on our plans for future development.

2. The 3D convolutional GAN

The 3DGAN model, described in [12], represents a first proof of concept of the possibility to use 3D convolutional GANs to simulate high granularity calorimeters. It addresses, however, an extremely simplified use case: the 3D image is limited in size and the simulated particles enter the detector with a fixed 90° angle. The primary particle energy is used to condition the training process, loosely following the ACGAN [14] approach. The Hadamard product between the energy and the latent space vector is calculated and used as an input to the generator network. At the same time, the loss function includes a constraint on the total deposited energy [15, 16].

This work extends the scope of [12] and simulates more realistic particles entering the detector with a variable incident angle and generating images that are 4$\times$ larger in size. Here, the GAN learns an angle-energy multivariate distribution and therefore the training is conditioned using both the incident angle and energy. As explained below, the 3DGAN architecture and the corresponding loss functions are modified to take into account physics based constraints.

2.1. Data set

The training data are generated in an effort to provide a common realistic data set that can be used to foster development of different Deep Learning and Machine Learning applications, from classification and regression networks to improve physics analysis to generative models for simulation. This data simulate a high granularity electromagnetic calorimeter (ECAL), designed in the context of the detector studies for the CLIC accellerator project [17], and they consist of a regular grid of $5.1\,mm^3$ cells and an inner calorimeter radius of $1.5\,m$. Further details can be found in [18, 16].

Here, we show results obtained using $140,000$ showers produced by single electrons with a primary energy (Ep) range of $100 - 200$ GeV and incident angle (θ) uniformly distributed between 60° and 120°. The final result is a three dimensional $51 \times 51 \times 25$ pixelized image centered around the barycenter of the shower: Figure 1 (a) presents 2D projections on the xy, xz and yz planes for Geant4 events. [1] Typically only a small fraction of cells (below 20%) receives some energy depositions and the size of the deposit can vary over a very large range (spanning more than 10 orders of magnitude). This high sparsity and large dynamic range

[1] The z axis lies along the detector depth and x, y are the transverse axes.

represent huge challenges compared to the typical RGB image generation problem where the pixel dynamic range is limited.

2.2. Architecture
A general GAN is made of two components, a discriminator and a generator: as explained in [12], the 3DGAN generator and discriminator networks consist of four 3D convolution layers. The discriminator has 16 filters with $5 \times 6 \times 6$ kernels and leaky ReLU activation functions in each layer. A batch normalization layer is added after the activation in all except the first layer. The output of the final convolution layer is flattened and connected to a sigmoid neuron corresponding to the GAN real/fake output as well as a linear unit performing energy regression. The generator has a latent vector of size 256. The first convolution layer has 64 filters with $6 \times 6 \times 8$ kernels. The next two layers have 6 filters of $5 \times 8 \times 8$ and $3 \times 5 \times 8$ kernels respectively. The last layer has a single filter of $2 \times 2 \times 2$ kernel. Leaky ReLU activation functions are used in all but the last layer that uses a ReLU. Batch normalization layers were added after the first and the second layer. The RMSprop [19] optimiser is used to train the network. The model is implemented in Keras [20] and Tensorflow v1.2 [21].

2.3. The loss function
As shown by the equation below, the loss function is build as a sum of several terms pertaining to the discriminator real/fake probability (L_G), the primary particle energy regression task (L_P) and a constraints on the total deposited energy (L_E), the pixel intensity spectrum (L_B) and the incident angles measurement (L_A). W are the corresponding weights, balancing the individual contributions.

$$L_{Tot} = W_G L_G + W_P L_P + W_A L_A + W_E L_E + W_B L_B$$

Extending the approach in [12], the last two quantities introduce domain-related terms in the loss function and they are essential in order to achieve the required level of accuracy over a very large pixel intensity dynamic range. L_G is implemented as a binary cross entropy, while a mean absolute percentage error is used for the primary energy,the deposited energy and the pixel intensity spectrum loss terms. The mean absolute error is used for the incident angle loss term.

2.4. The data preparation step and training process
Given the large pixel dynamic range, shown in figure 2 , we do not normalize the training data, instead we slightly reduce the dynamic range by applying a power function using an exponent smaller than one. The exponent is treated as a hyper-parameter and adjusted, to a value of 0.85, through a trial-and-error procedure. This procedure is essential to improve the description of the lower end of the pixel intensity spectrum.

The adversarial approach designs the training as a competition between the discriminator and the generator [7]. We adopt a balanced approach, by training the discriminator and the generator alternatively using the same number of steps. Initially, the discriminator is trained on a batch of real images and a batch of generated images (and label switching is applied [22]). Then, the generator is trained twice while keeping the discriminator weights fixed. Training on a single NVIDIA GeForce GTX 1080 card for one epoch requires about two hours and the training runs for 60 epochs.

3. Validation and Optimization
As a preliminary check we visually verify that 3DGAN can reproduce typical energy deposition patterns for different energy and incident angle values. Figure 1 shows some examples: it

compares 2D energy shower projections on the xy, xz and yz planes, as predicted by Geant4 and 3DGAN.

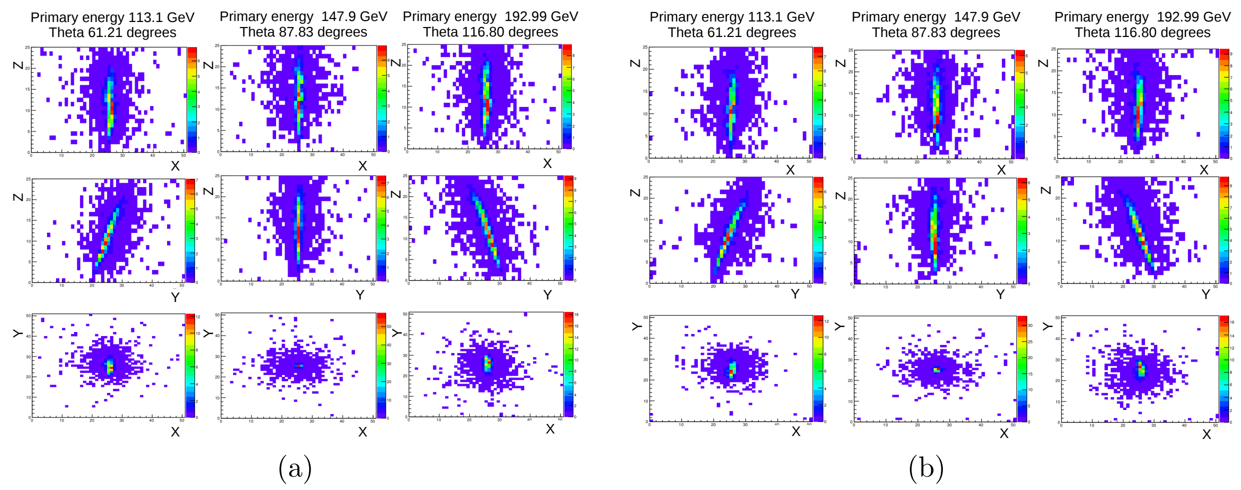


Figure 1. Geant4 vs. GAN generated events with similar primary energies and angles; a) Geant4 events ; b) generated events.

Figure 2 shows the Geant4/GAN ratio of the total energy deposited in the calorimeter as a function of the primary particle energy (a) and the single cell energy spectrum in linear (b) and logarithmic scale (c). It can be seen that, overall, GAN reproduces correctly the total energy deposited in the calorimeter and that single cell energies agree down to MeV values. As expected, the lower end of the spectrum is harder to simulate. Figure 3 (left) shows the Geant4 - GAN correlation difference calculated for different quantities (such as shower shapes distributions, deposited energy, incident angle, primary particle energy, etc..) as predicted by GAN and Geant4. It can be seen that the 3DGAN can correctly reproduce most internal correlations.

4. Distributed Training on public cloud

As Deep Learning models increase in complexity, model size and training time, the role played by distributed training becomes of primary importance: 3DGAN, for example, sums up to slightly more than a million parameters and it reaches convergence in about 5 days of training.

Several different algorithms for distributed training have been developed in recent years. Generally these algorithms work by splitting the training load across multiple concurrent

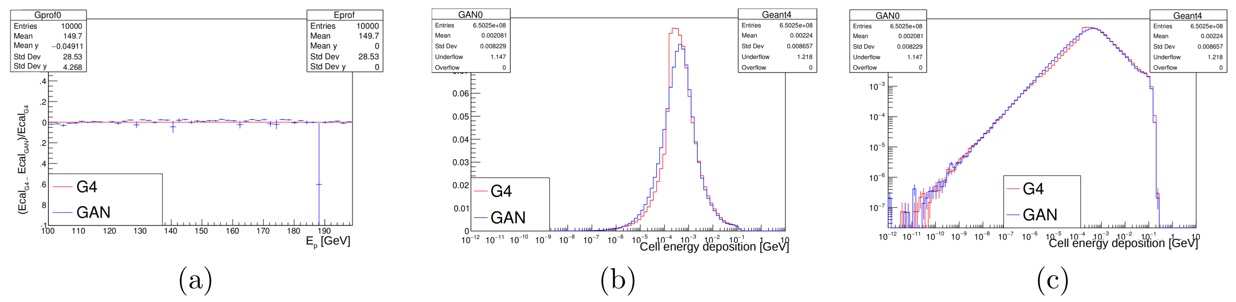
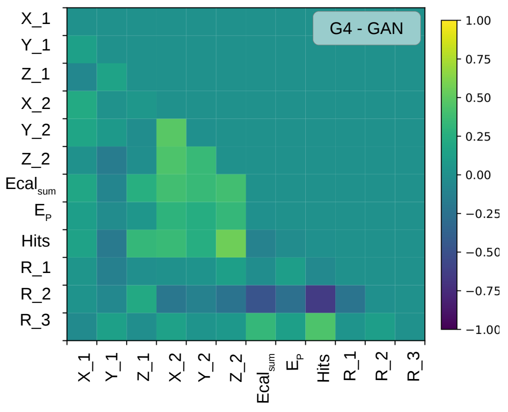
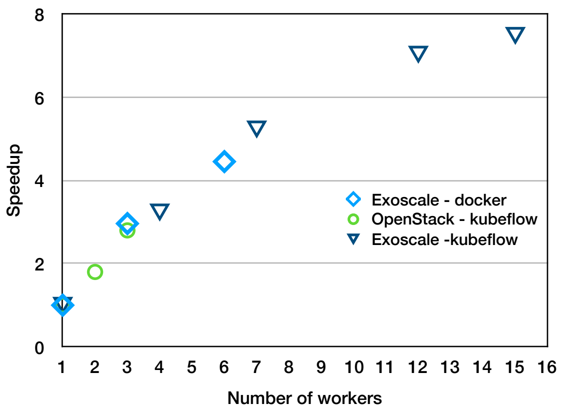


Figure 2. Geant4 vs. GAN comparison for 100-200 GeV primary particle energies; a) total energy deposited in calorimeter; b) single cell energy; c) single cell energy in log scale.



(a)



(b)

Figure 3. a) Geant4 - GAN difference between internal correlations calculated for several shower features; b) Speed up of training 3DGAN on the orthogonal incident angle sample, as a function of the number of MPI workers and with respect to training with one MPI worker.

processes, either threads on a single machine or jobs spread across separate nodes [23]. In order to reduce the training time we have interfaced 3DGAN to several frameworks, including Horovod [24] and mpi-learn [25], and benchmarked the parallel training process on different HPC systems [26, 23, 12].

Here we report on updated results on 3DGAN scaling performance on public clouds. Several initiatives exist that aim at understanding how the scientific community can integrate public clouds in their computing models. The European Commission funded project Helix Nebula Science Cloud (HNSciCloud) [27], for example, explored an hybrid cloud model linking together commercial cloud service providers and research organisations' in-house resources in order to provide an innovative vision for supporting the growing computing needs of the research community.

We have created a mpi-learn based docker [28] image and integrated it to kubernetes [29] and kubeflow [30] in order to smoothly deploy our workload on commercial cloud providers (for example Exoscale [2], equipped with NVIDIA P100 GPUs), via the HNSciCloud project. Results are shown in figure 3 (right): we have tested different deployment configurations and no overhead, due to the docker, kubernetes or kubeflow additional layer, has been observed. We have also compared the speedup performance obtained running on external cloud (blue) and on a small local set of GPUs, available in CERN Openstack (green) and we observed no difference in timing. Training time is significatly reduced, but the current speed-up is not linear. A possible explanation is that the workload for the workers is too small with respect to communication time and weights updates processing by the master. Analysis and optimisation of resource usage is part of our on-going work.

5. Summary and Plans

We presented updated results on the development and optimisation of our 3DGAN model: a three-dimensional convolutional Generative Adversarial Network for electromagnetic shower simulations in high granularity calorimeters. In particular, we have obtained results within 10% of Geant4 by introducing specific physics-related terms in the loss function and proved that our network can learn complex joint distributions. We have developed a fast deployment framework, based on commercially available platforms, such as kubernetes and kubeflow, to parallelise the

[2] https://www.exoscale.com

3DGAN training process across distributed resources available on public cloud. We obtained promising results that suggest that alternatives to in-house resources could be used to efficiently perform this kind of tasks.

In order to further optimise 3DGAN performance and extend it to the simulation of different calorimeter geometries we are currently developing a hyper-parameter optimisation framework based on genetic algorithms.

References

[1] Bird I 2011 *Annual Review of Nuclear and Particle Science* **61** 99–118 (*Preprint* https://doi.org/10.1146/annurev-nucl-102010-130059) URL https://doi.org/10.1146/annurev-nucl-102010-130059

[2] The Hep Software Foundation 2019 *Computing and Software for Big Science* **3** 7 ISSN 2510-2044 URL https://doi.org/10.1007/s41781-018-0018-8

[3] Apollinari G, Bjar Alonso I, Brning O, Fessia P, Lamont M, Rossi L and Tavian L 2017 *CERN Yellow Rep. Monogr.* **4** 1–516

[4] Lukas W 2012 *International Conference on Computing in High Energy and Nuclear Physics* vol 396

[5] Orbaker D 2010 *International Conference on Computing in High Energy and Nuclear Physics* vol 219

[6] Autiero D *et al.* (NOMAD Collaboration) 1998 *Nucl. Instrum. Methods Phys. Res., A* **425** 188. 28 p URL https://cds.cern.ch/record/364720

[7] Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 *ArXiv e-prints* (*Preprint* 1406.2661)

[8] de Oliveira L, Paganini M and Nachman B 2017 *Comput. Softw. Big Sci.* **1** 4 (*Preprint* 1701.05927)

[9] Paganini M, de Oliveira L and Nachman B 2018 *Phys. Rev. Lett.* **120** 042003 (*Preprint* 1705.02355)

[10] Erdmann M *et al. ArXiv High Energy Physics - Experiment e-prints* (*Preprint* https://arxiv.org/abs/1807.01954)

[11] Chekalina V *et al. ArXiv High Energy Physics - Experiment e-prints* (*Preprint* https://arxiv.org/abs/1812.01319)

[12] Khattak G, Vallecorsa S and Carminati F 2018 *2018 25th IEEE International Conference on Image Processing (ICIP)* pp 3913–3917 ISSN 2381-8549

[13] Carminati F, Khattak G and Vallecorsa S 2018 *2018 Computing in High Energy and Nuclear Physics(CHEP)*

[14] Odena A, Olah C and Shlens J 2016 *ArXiv e-prints* (*Preprint* 1610.09585)

[15] Carminati F, Gheata A, Khattak G, Lorenzo P M and Vallecorsa S 2017 *ACAT* (Seattle) URL https://indico.cern.ch/event/567550/contributions/2627179/

[16] Carminati F, Khattak G, Pierini M, Vallecorsafa S, Farbin A, Hooberman B, Wei W, Z M, P Vitoria B, Spiropulu M and Vlimant J 2017 *NIPS* URL https://dl4physicalsciences.github.io/files/nips_dlps_2017_15.pdf

[17] Boland M J *et al.* (CLIC, CLICdp) 2016 Updated baseline for a staged Compact Linear Collider (*Preprint* 1608.07537)

[18] Pierini M *et al.* 2016 The lcd dataset URL https://indico.hep.caltech.edu/event/102/contributions/41/

[19] Hinton G, Srivastava N and Swersky K 2012 Lecture 6a overview of minibatch gradi-ent descent.

[20] Chollet F *et al.* 2015 Keras https://github.com/fchollet/keras

[21] Abadi M *et al.* 2015 TensorFlow: Large-scale machine learning on heterogeneous systems software available from tensorflow.org URL https://www.tensorflow.org/

[22] Chintala S, Denton E, Arjovsky M and Mathieu M 2016 How to train a gan? tips and tricks to make gans work URL https://github.com/soumith/ganhacks

[23] Vlimant J R *et al.* 2018 *CHEP 2018 conference, in publication*

[24] Sergeev A and Balso M D 2018 *CoRR* **abs/1802.05799** (*Preprint* 1802.05799) URL http://arxiv.org/abs/1802.05799

[25] Anderson D, Vlimant J and Spiropulu M 2017 *CoRR* **abs/1712.05878** (*Preprint* 1712.05878) URL http://arxiv.org/abs/1712.05878

[26] Vallecorsa S *et al.* 2018 *High Performance Computing* vol 11203 URL https://doi.org/10.1007/978-3-030-02465-9_35

[27] Fernandes J *et al.* 2018 Activity report HNSciCloud pilot phase

[28] Merkel D 2014 *Linux J.* **2014** ISSN 1075-3583 URL http://dl.acm.org/citation.cfm?id=2600239.2600241

[29] The Kubernetes authors *Kubernetes Manual* [Online; accessed 31-May-2019] URL https://kubernetes.io/

[30] The Kubeflow authors *Kubeflow* [Online; accessed 31-May-2019] URL https://www.kubeflow.org/