Graph Generative Adversarial Networks for Sparse Data Generation in High Energy Physics

Raghav Kansal, Javier Duarte
University of California San Diego
La Jolla, CA 92093, USA

Breno Orzari, Thiago Tomei
Universidade Estadual Paulista
São Paulo/SP - CEP 01049-010, Brazil

Maurizio Pierini, Mary Touranakou*
European Organization for Nuclear Research (CERN)
CH-1211 Geneva 23, Switzerland

Jean-Roch Vlimant
California Institute of Technology
Pasadena, CA 91125, USA

Dimitrios Gunopoulos
National and Kapodistrian University of Athens
Athens 15772, Greece

Abstract

We develop a graph generative adversarial network to generate sparse data sets like those produced at the CERN Large Hadron Collider (LHC). We demonstrate this approach by training on and generating sparse representations of MNIST handwritten digit images and jets of particles in proton-proton collisions like those at the LHC. We find the model successfully generates sparse MNIST digits and particle jet data. We quantify agreement between real and generated data with a graph-based Fréchet Inception distance, and the particle and jet feature-level 1-Wasserstein distance for the MNIST and jet datasets respectively.

1 Introduction

At the CERN Large Hadron Collider (LHC), large simulated data samples are generated using Monte Carlo (MC) methods in order to translate the predictions of the standard model (SM), or beyond the SM theories, into observable detector signatures. These samples, numbering in the billions of events, are needed in order to accurately assess the predicted yields and their associated uncertainties. In order to achieve the highest level of accuracy possible, GEANT4-based simulation [1] is used to model the interaction of particles traversing the detector material. However, this approach comes at a high computational cost. At the LHC, such simulation workflows account for a large fraction of the total computing resources of the experiments, and with the planned high-luminosity upgrade, the expanded need for MC simulation may become unsustainable [2].

To accelerate simulation workflows, alternative methods based on generative deep learning models have been studied, including generative adversarial networks (GANs) [3–5] and variational autoencoders (VAEs) [6]. Applications include generating particle shower patterns in calorimeters [7–12], particle jets [13–15], event-level kinematic quantities [16–19], pileup collisions [20], and cosmic ray showers [21].

While these studies have proven to be effective for specific high energy physics (HEP) simulation tasks, it can be both challenging and inefficient to generalize such linear and convolutional neural

*Also at National and Kapodistrian University of Athens, Athens, Greece.

Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020), Vancouver, Canada.

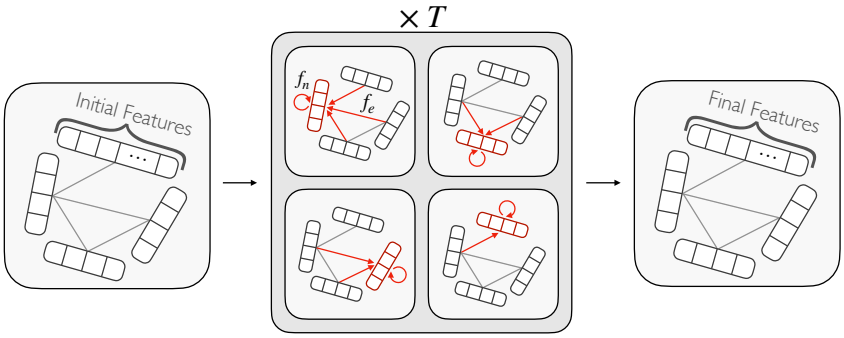

Figure 1: A message-passing neural network architecture.

network architectures to a full, low-level description of collision events due to the sparsity, complexity, and irregular underlying geometry (e.g. Ref. [22]) of HEP detector data. In this paper, we investigate a graph-based GAN to inherently account for data sparsity and any irregular geometry in the model architecture. As noted in Ref. [23], while graph networks have been successfully applied to classification and reconstruction tasks in HEP, they have yet to be explored for generative tasks, and this paper presents innovative work in this direction.

As a proxy for an LHC dataset, we first consider two sparse versions of the MNIST hand-written digit dataset [24]: one sparsified by hand and the other the so-called superpixels dataset [25]. Then, we apply the same strategy to a simulated dataset of jets produced in proton-proton collisions like those occurring at the LHC [26]. We note that while, for convenience, we train on simulated data, for real applications this model could be trained on experimental data.

2 Datasets

Our first dataset is a sparse graph representation of the MNIST dataset. From each image, we select the 100 highest intensity pixels as the nodes of a fully connected graph, with their feature vectors consisting of the x, y coordinates and intensities. This is directly analogous to selecting the coordinates and momenta of the highest momentum particles in a jet or highest energy hits in a detector. The second dataset, known as the MNIST superpixels dataset [25], was created by converting each MNIST image into 75 superpixels, corresponding to the nodes of a graph. The centers and intensities of the superpixels comprise the hidden features of the nodes. Edge features for both datasets are chosen to be the Euclidean distance between the connected nodes.

Finally, the third dataset [26–28] consists of simulated particle jets with transverse momenta $p_{\mathrm{T}}^{\mathrm{jet}} \approx 1$ TeV, originating from W and Z bosons, light quarks, top quarks, and gluons produced in $\sqrt{s} = 13$ TeV proton-proton collisions in an LHC-like detector. For our application, we only consider gluon jets and limit the number of constituents to the 30 highest p_{T} particles per jet (with zero-padding if there are fewer than 30). For each particle, the following three features resulted in the best performance: the relative transverse momentum $p_{\mathrm{T}}^{\mathrm{rel}} = p_{\mathrm{T}}^{\mathrm{particle}}/p_{\mathrm{T}}^{\mathrm{jet}}$ and the relative coordinates $\eta^{\mathrm{rel}} = \eta^{\mathrm{particle}} - \eta^{\mathrm{jet}}$ and $\phi^{\mathrm{rel}} = \phi^{\mathrm{particle}} - \phi^{\mathrm{jet}} \pmod{2\pi}$. We represent each jet as a fully-connected graph with the particles as the nodes. A single edge feature is taken to be the $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ between the connected particles. For evaluation we additionally consider the jet relative mass $m^{\mathrm{jet}}/p_{\mathrm{T}}^{\mathrm{jet}}$.

3 Graph Generator and Discriminator Architecture

For both the generator and discriminator we use a message-passing neural network (MPNN) architecture [29]. For a graph $G^t = (V^t, E^t)$ after t iterations of message passing ($t = 0$ corresponds to the original input graph), V^t a set of N nodes each with its own feature vector $\mathbf{h}_v^t$, and E^t a set of edges

each with its own feature vector $\mathbf{e}_{vw}^t$, we define one additional iteration of message passing as

$$\mathbf{m}_v^{t+1} = \sum_{w \in \mathcal{N}_v} f_e^{t+1}(\mathbf{h}_v^t, \mathbf{h}_w^t, \mathbf{e}_{vw}^t) \tag{1}$$

$$\mathbf{h}_v^{t+1} = f_n^{t+1}(\mathbf{h}_v^t, \mathbf{m}_v^{t+1}) \,, \tag{2}$$

where $\mathbf{m}_v^{t+1}$ is the aggregated message vector sent to node v, $\mathbf{h}_v^{t+1}$ is the updated hidden state of node v, f_e^{t+1} and f_n^{t+1} are arbitrary functions which are in general unique to each iteration t, and $\mathcal{N}_v$ is the set of nodes in the neighborhood of node v. The functions f_e^t and f_n^t are implemented in our case as independent multilayer perceptrons (MLPs).

The generator receives as input a graph G_{g}^0 containing a set of N nodes initialized with feature vectors $\mathbf{h}_v^0$ randomly sampled from a latent normal distribution, and then goes through T_{g} message passing iterations to output the final graph $G_{\mathrm{g}}^{T_{\mathrm{g}}}$ with new node features. The discriminator receives as input either a real or generated graph G_{d}^0 and goes through T_{d} message passing iterations to produce a final graph $G_{\mathrm{d}}^{T_{\mathrm{d}}}$, with a single binary feature for each node classifying it as real or fake. This feature then is averaged over all nodes with a 50% cutoff for the final discriminator output.

We note that a limitation of this architecture is that a particular model can only generate a fixed number of nodes and a constant graph topology. To overcome this we select a maximum number of nodes to produce per dataset and use zero-padding when necessary. We leave exploring generating variable-size dynamic graph topologies to future work.

A separate optimization is performed for every task to choose the hyperparameters T_{g}, T_{d}, hidden node feature size $|\mathbf{h}_v^t|$, and the number of layers and neurons in each layer of each f_e^t and f_n^t network. A different model is optimized for each MNIST digit, in analogy with the HEP use case, in which different generator settings are chosen for generating different physics processes. A variety of architectures experimented with, out of which an MPNN for both the generator and discriminator was most successful, are discussed in Appendix A.

4 Training

We use the least squares loss function [30] and the RMSProp optimizer with a learning rate of 10^{-5} for the discriminator and 3×10^{-5} for the generator [31], except for the superpixel digits '2', '4', and '9' where a learning rate of 10^{-5} for both the generator and discriminator had better performance. We use LeakyReLU activations (with negative slope coefficient 0.2) for all intermediate layers, and tanh and sigmoid activations for the final outputs of the generator and discriminator respectively. We attempted discriminator regularization to alleviate mode collapse via dropout [32], batch normalization [33], a gradient penalty [34], spectral normalization [35], adaptive competitive gradient descent [36] and data augmentation of real and generated graphs before the discriminator [37–39]. Apart from dropout (with fraction 0.5), none of these demonstrated significant improvement with respect to mode dropping or graph quality.

4.1 Evaluation

For model evaluation and optimization, as well as a quantitative benchmark on these datasets for comparison, we propose a graph-based Fréchet Inception distance (FID) [31] inspired metric for the MNIST datasets, and the 1-Wasserstein distance (W_1) for the jets dataset as in Ref. [13, 40]. The two metrics differ for reasons explained below.

Traditionally, the FID metric is used on image datasets, using the pre-trained Inception-v3 image classifier network. It compares the statistics of the outputs of a single layer of this network between generated and real samples, and has been shown to be a consistent measure of similarity between generated and real samples in terms of both quality and diversity. To adapt FID to graph datasets, we use the MoNet model of Ref. [25] as the pre-trained classifier, which can be found at Ref. [41], and calculate what we call the graph Fréchet distance (GFD):

$$\mathrm{GFD} = ||\mu_{\mathrm{r}} - \mu_{\mathrm{g}}||^2 + \mathrm{Tr}(\mathbf{\Sigma}_{\mathrm{r}} + \mathbf{\Sigma}_{\mathrm{g}} - 2(\mathbf{\Sigma}_{\mathrm{r}}\mathbf{\Sigma}_{\mathrm{g}})^{1/2}) \,, \tag{3}$$

where μ_{r} (μ_{g}) is the vector of means of the activation function outputs of the first fully connected layer in the pre-trained MoNet model for real (generated) images and $\mathbf{\Sigma}_{\mathrm{r}}$ ($\mathbf{\Sigma}_{\mathrm{g}}$) is the corresponding covariance matrix.

To evaluate the performance on the jet dataset, we calculate directly W_1 between the distributions of the three particle-level features and the jet m/p_{T} in the real and generated samples. Unlike for MNIST, these quantities correspond to meaningful physical observables hence measuring W_1 between their distributions is a more desirable metric than the GFD. We use bootstrapping to calculate a baseline W_1 between samples within the real dataset alone, taking P pairs of random sets of N jets and calculating W_1 between the distributions of each pair. For three combinations of P and N, the means and standard deviations are shown in Table 2. The W_1 values between the real and generated distributions are similarly calculated by generating P sets of N jets each and comparing them to P random sets of N real jets.

5 Results

For the MNIST-derived datasets, we optimized the hyperparameters of our model using our GFD metric. A sample of hyperparameter settings we tested with their corresponding GFD scores for all 10 digits can be seen in Appendix B. Based on this optimization, the final hyperparameters chosen for the three datasets as listed in Table 1. Fig. 2 (left) shows a comparison between real and generated digits for the sparse MNIST dataset. The generator is able to reproduce all 10 digits successfully with high accuracy and little evidence of mode dropping. Similarly, Fig. 2 (right) compares real and generated digits for the MNIST superpixel dataset. Again, we can see that the model successfully reproduces the real samples, though there is some evidence of mode dropping, particularly with the more complex digits and rarer modes. We leave exploring this issue further to future work. The average of our best GFD scores across all 10 digits is 0.52 and 0.30 for the Sparse MNIST and superpixels dataset respectively.

Table 1: Optimized hyperparameters for each dataset.

Dataset	Digits	T_{g}	T_{d}	f_e (Neurons per layer)				f_n (Neurons per layer)				$\lvert\mathbf{h}_v^t\rvert$
				In	1	2	Out	In	1	2	Out	
Sparse MNIST	2 3 4 5 7	1	1	65	64	—	128	160	256	256	32	32
	0 1 6 8 9	1	1	65	96	160	192	224	256	256	32	32
Superpixels	All	2	2	65	64	—	128	160	256	256	32	32
Jet	—	2	2	65	96	160	192	224	256	256	32	32

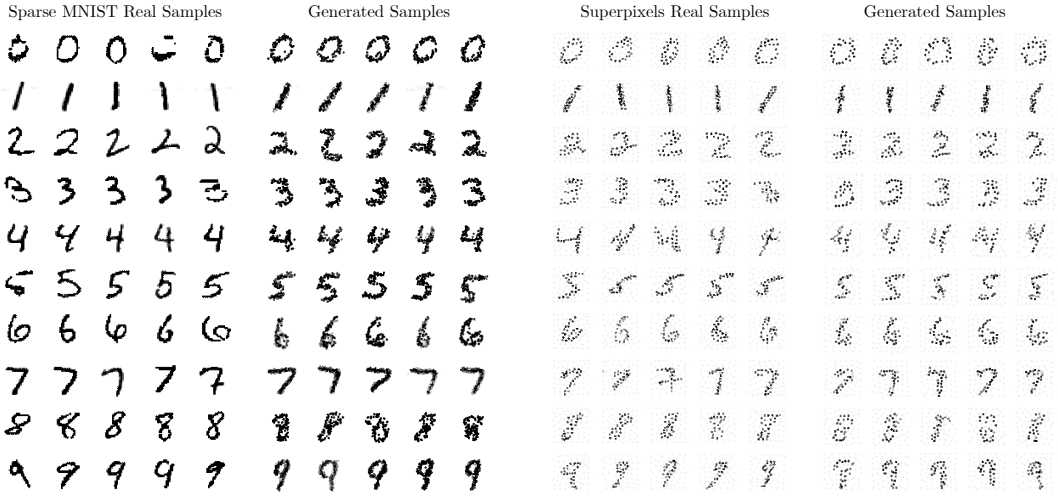

Figure 2: Samples from our sparse MNIST dataset (far left) compared to samples from our graph GAN (center left). Samples from the MNIST superpixels dataset (center right) compared to samples from our graph GAN (far right).

Our results on the jet dataset using our message-passing architecture show excellent agreement, both qualitatively and quantitatively using W_1. Example generated and real distributions of particle η^{rel}, ϕ^{rel}, and $p_{\mathrm{T}}^{\mathrm{rel}}$ and jet m/p_{T} are shown in Fig. 3 for 100,000 jets. The mean W_1 values between real

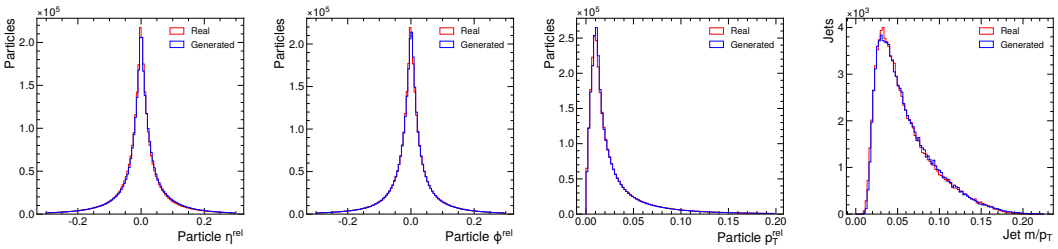

Figure 3: Distributions of the particle η^{rel}, ϕ^{rel}, and $p_{\mathrm{T}}^{\mathrm{rel}}$, and jet m/p_{T} for 100,000 real and generated jets.

Table 2: Mean W_1 values and standard deviations between particle-level distributions of η^{rel}, ϕ^{rel}, and $p_{\mathrm{T}}^{\mathrm{rel}}$, and the jet-level distribution of m/p_{T} derived from comparing randomly selected sets of N real jets and from comparing sets of random N real and N generated jets. This comparison is repeated P times to derive a mean and standard deviation.

		W_1 mean $\pm$ standard deviation ($\times 10^{-3}$)							
		Pairs of real distributions				Real and generated distributions			
N	P	η^{rel}	ϕ^{rel}	$p_{\mathrm{T}}^{\mathrm{rel}}$	Jet m/p_{T}	η^{rel}	ϕ^{rel}	$p_{\mathrm{T}}^{\mathrm{rel}}$	Jet m/p_{T}
100	1,000	6 ± 2	6 ± 2	1.4 ± 0.5	6 ± 2	5 ± 2	11 ± 4	2 ± 1	6 ± 2
1,000	100	1.8 ± 0.1	1.7 ± 0.1	0.47 ± 0.02	1.9 ± 0.7	2.4 ± 0.6	3 ± 1	0.7 ± 0.2	2.2 ± 0.9
10,000	10	0.5 ± 0.1	0.5 ± 0.1	0.11 ± 0.02	0.5 ± 0.1	2.2 ± 0.2	1.3 ± 0.4	0.51 ± 0.06	1.1 ± 0.5

and generated jet distributions are presented in Table 2. For samples of 100 jets, the mean W_1 values (between real and generated jet samples) agree with the expected ones (between real jet samples) within one standard deviation, but this is not the case when the jet sample size is increased to 1,000 or 10,000. Thus, while the generator has sufficient fidelity for smaller sample sizes, there is room for improvement for larger ones. We also note that there is little evidence of mode collapse with this dataset because we can see that the entire distributions, including rarer data samples in the tails, are reproduced with high accuracy.

6 Summary

We have presented a novel architecture for generating graphs using a generative adversarial network based on a message-passing neural network, which we successfully apply to two MNIST-derived graph datasets as well as an LHC jet dataset. This architecture works efficiently with sparse data and inherently adapts to any underlying geometry. We find the model generates realistic MNIST graph data albeit with some evidence of mode dropping, which we quantify with our graph Fréchet distance. For the jet dataset, we measure the quality of the generator using a metric based on the 1-Wasserstein distance and find high accuracy for smaller sample sizes. The application of our model to a high energy physics dataset demonstrates its flexibility, and indicates this approach may be readily used for fast simulation of a variety of scientific datasets, including sensor-level data in high granularity calorimeters.

Broader Impacts

Physics experiments needing to generate large simulated datasets may benefit from this work. If this type of algorithm is used by experiments to produce such datasets, to produce datasets, it may reduce the computational cost of running the experiment. At the same time, if the fidelity of the algorithm is not as high as desired, it may result in suboptimal or inaccurate scientific results. Other beneficiaries of this work may include any group with a need to generate graph-based datasets following some realistic patterns.

Acknowledgments and Disclosure of Funding

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 772369). R. K. was partially

supported by an IRIS-HEP fellowship through the U.S. National Science Foundation (NSF) under Cooperative Agreement OAC-1836650. J. D. is supported by the U.S. Department of Energy (DOE), Office of Science, Office of High Energy Physics Early Career Research program under Award No. DE-SC0021187. B. O. was partially supported by grants #2018/01398-1 and #2019/16401-0, São Paulo Research Foundation (FAPESP). J-R. V. is partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 772369) and by the U.S. DOE, Office of Science, Office of High Energy Physics under Award No. DE-SC0011925, DE-SC0019227, and DE-AC02-07CH11359. This work was performed using the Pacific Research Platform Nautilus HyperCluster supported by NSF awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute. Thanks to CENIC for the 100 Gpbs networks. Also, thanks to Kwonjoon Lee for valuable discussions about GAN training.

Appendices

A Architecture Experiments

We experimented with multiple GAN architectures for producing graphs. This included standard MLP and CNN generators and discriminators, which were predictably unsuccessful due to the architectures not being permutation invariant. However, despite this limitation, a CNN classifier achieved $> 90\%$ accuracy on our sparse MNIST dataset.

A better architecture we attempted was using a gated recurrent unit (GRU) based recurrent neural network (RNN) as our generator together with a CNN discriminator because of its success as a classifier. The RNN received as input a random sample from our latent space and iteratively output each nodes' features in sequence. While there was evidence of this model learning some graph structure in Fig. 4 (left), it was not able to reproduce digits. Nonetheless, this model has the desirable ability to produce graphs with arbitrary numbers of nodes and future research could explore this further.

The MPNN generator was a clear improvement and was able to successfully reproduce graphs from our dataset. We tested a CNN discriminator initially, and the GAN produced high-quality outputs, as seen in Fig. 4 (right). However, training was difficult and inconsistent, and there was clear evidence of mode collapse. Replacing the CNN with an MPNN improved both these aspects.

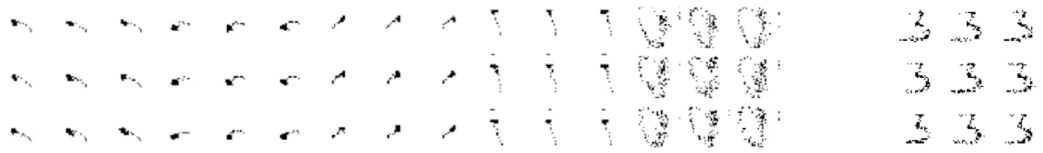

Figure 4: Samples from an RNN generator with a CNN discriminator (left), which exhibit some structure, but no coherent digits. Samples from an MPNN generator with a CNN discriminator (right), which shows a successful output but displays mode collapse.

With the MPNN, we experimented with an additional step in the discriminator after the message-passing iterations to produce the final classification output

$$f_{nd}\left(\sum_{v \in V_{\mathrm{d}}^{T_{\mathrm{d}}}} \mathbf{h}_v^{T_{\mathrm{d}}}\right), \tag{4}$$

where f_{nd} is implemented as an MLP. This allows the discriminator to take a holistic look at the graph instead of classifying on a per node basis. However, empirically we found that this addition marginally decreased performance so was not used in the final architecture.

For edge features, we tested the absolute Euclidean distance and the vector displacement in Cartesian and polar coordinates between node positions, as well as the difference in node intensities. The Euclidean distance alone was the most effective. We also investigated fully connected graphs versus edge connections only within a local neighborhood as in Ref. [25] and the performance was comparable.

B Hyperparameter Optimization

A characteristic sample of hyperparameter combinations we tested for the sparse MNIST dataset digit '3' can be seen in Table 3. A similar sample for the superpixels MNIST dataset digit '3' can be seen in Table 4. Based on this hyperparameter optimization, the final settings for the MNIST datasets were chosen as shown in Table 1. For the jet dataset, we chose one of the hyperparameter settings optimized for the MNIST datasets and found it to be effective.

Table 3: Sample of hyperparameter combinations for the sparse MNIST dataset with their respective GFD scores for digit '3.' The selected combination is in bold.

T_{g}	T_{d}	f_e (Neurons per layer)				f_n (Neurons per layer)					$\lvert\mathbf{h}_v^t\rvert$	GFD
		In	1	2	Out	In	1	2	3	Out		
1	**1**	**65**	**64**	**—**	**128**	**160**	**256**	**256**	**—**	**32**	**32**	**0.42**
1	1	65	64	—	128	160	256	256	256	32	32	0.50
1	1	65	92	160	192	224	256	256	256	32	32	0.92
2	1	65	64	—	128	160	256	256	—	32	32	0.80
1	2	65	64	—	128	160	256	256	—	32	32	1.48

Table 4: Sample of hyperparameter combinations for the MNIST superpixels dataset with their respective GFD scores for digit '3'.

T_{g}	T_{d}	f_e (Neurons per layer)				f_n (Neurons per layer)					$\lvert\mathbf{h}_v^t\rvert$	GFD
		In	1	2	Out	In	1	2	3	Out		
1	1	65	64	—	128	160	256	256	—	32	32	2.48
1	2	65	64	—	128	160	256	256	—	32	32	1.93
2	1	65	64	—	128	160	256	256	—	32	32	1.18
2	**2**	**65**	**64**	**—**	**128**	**160**	**256**	**256**	**—**	**32**	**32**	**0.22**
2	2	65	64	—	128	160	256	—	—	32	32	2.37
2	2	65	64	—	128	160	256	256	256	32	32	0.44
2	2	65	92	160	192	160	256	256	—	32	32	0.31
2	2	17	64	—	128	136	256	256	—	8	8	1.08
2	2	257	92	160	192	320	256	256	—	128	128	0.38
3	3	65	92	160	192	160	256	256	—	32	32	0.62

References

[1] GEANT4 Collaboration, "GEANT4: A Simulation toolkit", *Nucl. Instrum. Methods Phys. Res. A* **A506** (2003) 250, doi:10.1016/S0168-9002(03)01368-8.

[2] HEP Software Foundation, "A roadmap for HEP software and computing R&D for the 2020s", *Comput. Softw. Big Sci.* **3** (2019) 7, doi:10.1007/s41781-018-0018-8, arXiv:1712.06982.

[3] I. J. Goodfellowet al., "Generative adversarial nets", in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani et al., eds., p. 2672. Curran Associates, Inc., 2014. arXiv:1406.2661.

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN", (2017). arXiv:1701.07875.

[5] I. Gulrajaniet al., "Improved training of Wasserstein GANs", in *Advances in Neural Information Processing Systems 30*, I. Guyon et al., eds., p. 5767. Curran Associates, Inc., 2017. arXiv:1704.00028.

[6] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes", (2013). arXiv:1312.6114.

[7] M. Paganini, L. de Oliveira, and B. Nachman, "Accelerating science with generative adversarial networks: An application to 3D particle showers in multilayer calorimeters", *Phys. Rev. Lett.* **120** (2018) 042003, doi:10.1103/PhysRevLett.120.042003, arXiv:1705.02355.

[8] M. Paganini, L. de Oliveira, and B. Nachman, "CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks", *Phys. Rev. D* **97** (2018) 014021, doi:10.1103/PhysRevD.97.014021, arXiv:1712.10321.

[9] M. Erdmann, J. Glombitza, and T. Quast, "Precise simulation of electromagnetic calorimeter showers using a Wasserstein generative adversarial network", *Comput. Softw. Big Sci.* **3** (2019) 4, doi:10.1007/s41781-018-0019-7, arXiv:1807.01954.

[10] D. Salamani et al., "Deep generative models for fast shower simulation in atlas", in *2018 IEEE 14th International Conference on e-Science (e-Science)*, p. 348. 2018. doi:10.1109/eScience.2018.00091.

[11] D. Belayneh et al., "Calorimetry with deep learning: particle simulation and reconstruction for collider physics", *Eur. Phys. J. C* **80** (2020) 688, doi:10.1140/epjc/s10052-020-8251-9, arXiv:1912.06794.

[12] ATLAS Collaboration, "Fast simulation of the ATLAS calorimeter system with generative adversarial networks", Technical Report ATL-SOFT-PUB-2020-006, 2020.

[13] L. de Oliveira, M. Paganini, and B. Nachman, "Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis", *Comput. Softw. Big Sci.* **1** (2017) 4, doi:10.1007/s41781-017-0004-6, arXiv:1701.05927.

[14] P. Musella and F. Pandolfi, "Fast and accurate simulation of particle detectors using generative adversarial networks", *Comput. Softw. Big Sci.* **2** (2018) 8, doi:10.1007/s41781-018-0015-y, arXiv:1805.00850.

[15] S. Carrazza and F. A. Dreyer, "Lund jet images from generative and cycle-consistent adversarial networks", *Eur. Phys. J. C* **79** (2019) 979, doi:10.1140/epjc/s10052-019-7501-1, arXiv:1909.01359.

[16] S. Otten et al., "Event generation and statistical sampling for physics with deep generative models and a density information buffer", (2019). arXiv:1901.00875.

[17] B. Hashemi et al., "LHC analysis-specific datasets with generative adversarial networks", (2019). arXiv:1901.05282.

[18] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat, and S. Palazzo, "DijetGAN: A generative-adversarial network approach for the simulation of QCD dijet events at the LHC", *J. High Energy Phys.* **08** (2020) 110, doi:10.1007/JHEP08(2019)110, arXiv:1903.02433.

[19] A. Butter, T. Plehn, and R. Winterhalder, "How to GAN LHC events", *SciPost Phys.* **7** (2019) 075, doi:10.21468/SciPostPhys.7.6.075, arXiv:1907.03764.

[20] J. Arjona Martínez et al., "Particle generative adversarial networks for full-event simulation at the LHC and their application to pileup description", *J. Phys. Conf. Ser.* **1525** (2020) 012081, doi:10.1088/1742-6596/1525/1/012081, arXiv:1912.02748.

[21] M. Erdmann, L. Geiger, J. Glombitza, and D. Schmidt, "Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks", *Comput. Softw. Big Sci.* **2** (2018) 4, doi:10.1007/s41781-018-0008-x, arXiv:1802.03325.

[22] CMS Collaboration, A. Martelli, "The CMS HGCAL detector for HL-LHC upgrade", in *5th Large Hadron Collider Physics Conference*. 8, 2017. arXiv:1708.08234.

[23] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics", (2020). arXiv:2007.13681. Accepted by *Mach. Learn.: Sci. Technol.*

[24] Y. LeCun and C. Cortes, "MNIST handwritten digit database", 2010. http://yann.lecun.com/exdb/mnist/.

[25] F. Monti et al., "Geometric deep learning on graphs and manifolds using mixture model CNNs", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 5425. IEEE, New York, NY, 2017. arXiv:1611.08402. doi:10.1109/CVPR.2017.576.

[26] M. Pierini, J. M. Duarte, N. Tran, and M. Freytsis, "hls4ml LHC jet dataset (30 particles)", 01, 2020. doi:10.5281/zenodo.3601436.

[27] J. Duarte et al., "Fast inference of deep neural networks in FPGAs for particle physics", *J. Instrum.* **13** (2018) P07027, doi:10.1088/1748-0221/13/07/P07027, arXiv:1804.06913.

[28] E. Coleman et al., "The importance of calorimetry for highly-boosted jet substructure", *J. Instrum.* **13** (2018) T01003, doi:10.1088/1748-0221/13/01/T01003, arXiv:1709.08705.

[29] J. Gilmeret al., "Neural message passing for quantum chemistry", in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, eds., volume 70, p. 1263. PMLR, 2017. arXiv:1704.01212.

[30] X. Mao et al., "Multi-class generative adversarial networks with the L2 loss function", (2016). arXiv:1611.04076.

[31] M. Heuselet al., "GANs trained by a two time-scale update rule converge to a local nash equilibrium", in *Advances in Neural Information Processing Systems 30*, I. Guyon et al., eds., p. 6626. Curran Associates, Inc., 2017. arXiv:1706.08500.

[32] N. Srivastavaet al., "Dropout: A simple way to prevent neural networks from overfitting", *J. Mach. Learn. Res.* **15** (2014) 1929.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, eds., volume 37, p. 448. PMLR, 2015. arXiv:1502.03167.

[34] I. Gulrajaniet al., "Improved training of wasserstein gans", in *Advances in Neural Information Processing Systems 30*, I. Guyon et al., eds., p. 5767. Curran Associates, Inc., 2017. arXiv:1704.00028.

[35] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks", in *6th International Conference on Learning Representations*. 2018. arXiv:1802.05957.

[36] F. Schäfer, H. Zheng, and A. Anandkumar, "Implicit competitive regularization in GANs", (2019). arXiv:1910.05852.

[37] T. Karras et al., "Training generative adversarial networks with limited data", (2020). arXiv:2006.06676.

[38] N.-T. Tran et al., "On data augmentation for GAN training", (2020). arXiv:2006.05338.

[39] Z. Zhao et al., "Image augmentations for GAN training", (2020). arXiv:2006.02595.

[40] Y. Lu, J. Collado, D. Whiteson, and P. Baldi, "SARM: Sparse autoregressive model for scalable generation of sparse images in particle physics", (2020). arXiv:2009.14017.

[41] R. Kansal, "rkansal47/graph-gan: v0.1.0", 2020. doi:10.5281/zenodo.4299011.