DeepTreeGANv2: Iterative Pooling of Point Clouds

Moritz A.W. Scham ⓘ*
Deutsches Elektronen-Synchrotron DESY,
Germany
`moritz.scham@desy.de`

Dirk Krücker ⓘ
Deutsches Elektronen-Synchrotron DESY,
Germany
`dirk.kruecker@desy.de`

Kerstin Borras ⓘ†
Deutsches Elektronen-Synchrotron DESY,
Germany
`kerstin.borras@desy.de`

Abstract

In High Energy Physics, detailed and time-consuming simulations are used for particle interactions with detectors. To bypass these simulations with a generative model, the generation of large point clouds in a short time is required, while the complex dependencies between the particles must be correctly modelled. Particle showers are inherently tree-based processes, as each particle is produced by the decay or detector interaction of a particle of the previous generation. In this work, we present a significant extension to DeepTreeGAN [1], featuring a critic, that is able to aggregate such point clouds iteratively in a tree-based manner. We show that this model can reproduce complex distributions, and we evaluate its performance on the public JetNet 150 dataset.

1 Introduction

In particle physics, detailed, near-perfect simulations of the underlying physical processes, the measurements, and the details of the experimental apparatus are state-of-the-art. However, accurate simulations of large and complex detectors, especially calorimeters, using current Monte Carlo-based tools such as those implemented in the Geant4 [2] toolkit are computationally intensive. Currently, more than half of the worldwide Large Hadron Collider (LHC) grid resources are used for the generation and processing of simulated data [3]. For the future High-Luminosity upgrade of the LHC (HL-LHC [4]), the computational requirements will exceed the computational resources without significant speedups, e.g., for the CMS experiment by 2028 [5] projecting the current technologies. This will become even more demanding for future high-granularity calorimeters, e.g., the CMS HGCAL [6] with its complex geometry and extremely large number of channels. To address these challenges, fast simulation approaches based on Deep Learning, including Generative Adversarial Networks (GAN) [7] and Variational Autoencoders (VAE) [8], have been explored early [9–14] and deployed recently [15].

---

*Also at Jülich Supercomputing Centre, Institute for Advanced Simulation, Germany and RWTH Aachen University, III. Physikalisches Institut A, Germany

†Also at RWTH Aachen University, III. Physikalisches Institut A, Germany

The majority of these approaches consider the data as voxels living on three-dimensional grids. Especially for high-granularity calorimeters like the HGCAL, the calorimeter cells are highly irregular, and the data is often sparse. In such a situation, it is beneficial to consider the data as Point Clouds (PC), e.g., tuples of cell energies and three-dimensional coordinates, i.e., four-dimensional point clouds [16–18]. For modelling calorimeter showers, which are cascades of particles, it seems natural to model the creation of such point clouds as a tree-like process. Such an approach can be implemented by Graph Neural Networks (GNN) [19] and Graph Convolutions [20]. For modelling PC data representing three-dimensional objects, a similar approach [21] called tree-GAN [21] exists. For our use case, especially for modelling the large dynamic range of the energy dimension, this approach is not sufficient. Here we report on the development of a largely extended Graph GAN approach, which we named *DeepTreeGAN* that can be applied to multiple areas of PC generation. To demonstrate the abilities of our model, we present a first application to the JetNet dataset introduced in the next section.

1.1 JetNet

In this study, the JetNet [22, 23] datasets are used. Each dataset contains PYTHIA [24] jets with an energy of about 1 TeV, with each jet containing up to 30 or 150 constituents (here: 150). The datasets differ in the jet-initiating parton. Here, the dataset for top quark jets is studied. Each dataset contains about 170k individual jets split as 110k/10k/50k for training/test/validation, where the validation dataset is used for our results. The jet constituents, the particles, are clustered with a cone radius of $R = 0.8$. These particles are considered to be massless and can therefore be described by their 3-momenta or equivalently by transverse momentum p_T, pseudorapidity η, and azimuthal angle ϕ. In the JetNet datasets, these variables are given relative to the jet momentum: $\eta_i^{\mathrm{rel}} := \eta_i - \eta^{\mathrm{jet}}$, $\phi_i^{\mathrm{rel}} := \phi_i - (\phi^{\mathrm{jet}} \mod 2\pi)$, and $p_{T,i}^{\mathrm{rel}} := p_{T,i}/p_{\mathrm{T,jet}}$, where i runs over the particles in a jet. Calculating the invariant mass from these relative quantities, for example, for the jet mass, implies $m^{\mathrm{rel}} = m_{jet}/p_{\mathrm{T,jet}}$. This dataset has gained popularity in the particle physics community as a benchmark for PC-based generative models [16–18, 25–34].

2 Architecture

PC-based GANs [17, 27, 30, 33] frequently choose a refinement approach: Starting with a PC of the desired size with features sampled from noise, the PC is updated multiple times in the generator and the critic. In the final step of the critic, the points are aggregated and mapped to a single value, usually with a feedforward neural network (FFN). In this way, the dimensionality is greatly reduced in a single step. Commonly, the PC is updated by first transforming the points individually. Then they are aggregated into a global vector, which is used in-turn to transform the points again. In a image-based GAN approach like DCGAN [35], the generator would apply convolutions in sequence to iteratively upscale a random vector to an image and the critic would apply convolutions in sequence to iteratively downscale an image to a scalar. These convolutions cannot be directly translated to PCs, which is why the mentioned refinement approach is prevalent for PC-based GANs. Our approach instead is to translate this principle and to construct a critic that reduces the dimensionality iteratively. In this way, we complement the DeepTree generator [1], that iteratively upscales PCs.

2.1 Generator

The generator part of this model largely matches the one published in [1], with a few key differences: Instead of splitting the leaf vector into one part per branch and mapping the latter separately to the desired number of branches, the full leaf vector is now mapped to the number of branches times number of features. The generator uses 2, 3, 5, and 5 branches (with a product of 150) and goes from 64 to 33 to 20 to 10 to 3 features. For each event, the number of constituents is sampled from the dataset and the output of the generator is cut to this size.

2.2 Critic

The critic is shown in Figure 1, its *components* are described in the following section. It features three *subcritics*, that are applied to the different stages of aggregation: The first is applied directly on the input PC (up to 150 points), the other two are applied after each *bipartite pool* (pooling to 30 points
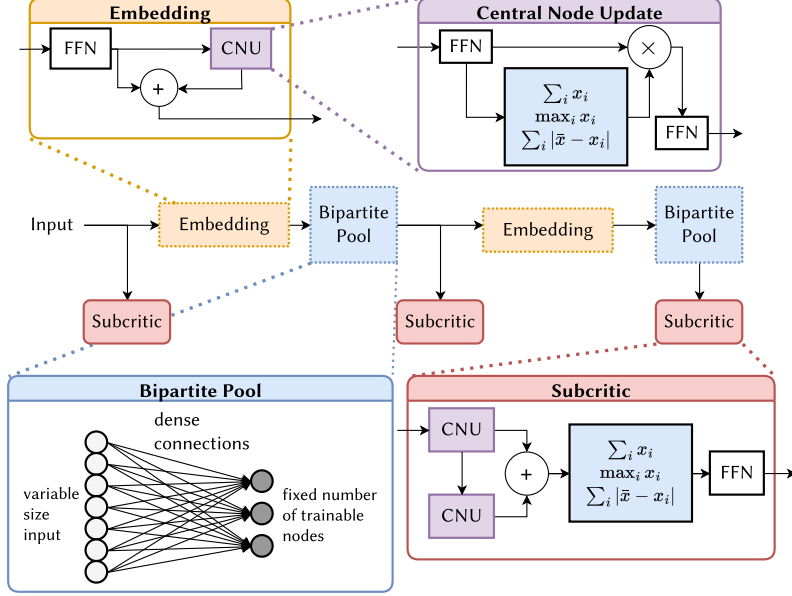
Figure 1: The critic, as described in 2.2

and 6 points). The expectation is that the first *subcritic* provides feedback on the more low-level features, while the two later *subcritics* operate on aggregated points and thus provide feedback on more high-level features. Before the *bipartite pool*, the *embedding* layer is applied, which maps the number of features to 10 and transforms the input.
The critic is conditioned on p_T, η and the mass of the jet.

Bipartite Pool For the iterative reduction of the number of points a pooling operation should fulfill the following conditions: First, it must be differentiable to enable backpropagation. Then it should be invariant under permutation because the input is permutation invariant. As the number of input points varies, it needs to accept an arbitrary number of points, down to a single one. The run-time pooling operation should scale well to large PCs, ideally linear with the number of points. Finally, it should map to a fixed number of points to construct a regular shaped output tensor, that allows faster and more convenient implementations. To achieve these requirements, we propose constructing a bipartite graph, densely connecting the input PC to a fixed number of trainable nodes and applying a Message Passing Layer (MPL) to this graph. This will return a PC with a fixed number of points. The dense connection yields a number of edges that equals the number of input points times the number of trainable nodes. Because each edge yields one message, the run-time of the MPL should be proportional to the size of the input PC. As an MPL, Gatv2Conv [36] is used with 16 attention heads, as implemented in PyTorch Geometric [37].
A similar approach has been presented in [38], but implemented as an attention mechanism instead of an MPL. An advantage of our choice is that the bipartite pool can handle variable sized PCs without masking.

FFNs The *FFNs* used in this critic consist of 3 layers with 100 hidden nodes without bias. The first two hidden layers are followed by a 50% dropout layer and LeakyReLU activation with a negative slope of 0.1. Spectral normalization [39] is applied to the *FFNs*, except for the *FFN* in the *embedding* layer, where batch normalization [40] is applied before the activation.

Multi-Aggregation At different stages, the points need to be aggregated in a single vector. This is commonly done by summing over the points, e.g., [33]. To provide additional information about the distribution, we compute this vector by concatenating the sum, the maximum, the number of points and the width. The width of the distribution is estimated by computing the mean absolute deviation: $\frac{1}{n} \sum_i |x_i - \bar{x}|$.

Central Node Update For the following description of the *embedding* layer and the *subcritics*, we define a *Central Node Update* Layer (*CNU*): First, the input is transformed with an *FFN*. Then the transformed PC is aggregated with the *multi-aggregation*. Lastly, each of the transformed nodes is concatenated with the aggregated vector and passed through a second *FFN* that maps the nodes back to their original dimension.

Embedding The *embedding* layer first maps the points independently to 10 features with an *FFN* and passes the PC though a *CNU* layer with a residual connection.

Subcritics Each subcritic is constructed using two *CNUs* with a residual connection. The points are then aggregated in the *multi-aggregation* scheme. This aggregated vector is then concatenated with the condition (p_T, η and mass of the jet) and passed to an *FFN*, mapping the vector to a single output. This yields three output values for each event which are summed such that the application of the loss and the backpropagation takes place for all the outputs in parallel.

2.3 Training

Generator and critic are trained using the Hinge [41] loss, with a ratio of 1 to 2 gradients steps. The optimizer for the generator (critic) is Adam [42] with $\beta_1 = 0.9, \beta_2 = 0.999$, a weight decay of 10^{-4} and a learning rate of 10^{-5} ($3 * 10^{-5}$). Additional to the Hinge loss, the generator is also trained to minimize the feature matching loss [43] with a factor of 0.1. As an "intermediate layer" the concatenation of the *multi-aggregation* of the input PC, together with the output of the *embedding* and *bipartite pool* layers are used.

2.4 Preprocessing & Postprocessing

For the training, the η^rel and ϕ^rel distributions are scaled separately to a normal distribution. The $p_\mathrm{T}^\mathrm{rel}$ distribution is scaled using a Box-Cox transformation with standardizing as implemented in [44]. To produce samples, the inverse scaling is applied to the output of the generator. The original dataset is scaled so that $\sum_i p_\mathrm{T}^i = 1$. This feature is difficult to model for the generator, while it is easy to recognize for the critic. To avoid limiting the performance of the model, the output of the generator is scaled in the same way.
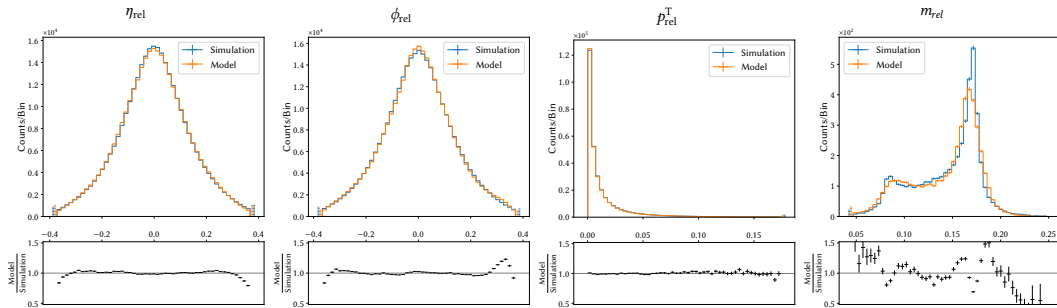
3 Results

Figure 2: The distributions of $\eta^\mathrm{rel}, \phi^\mathrm{rel}, p_\mathrm{T}^\mathrm{rel}$ for constituents of the jets and the mass of the jets for the top-quarks `JetNet` 150 dataset (Simulation), as described in Section 1.1 and DeepTreeGAN (Model).

The variables of the constituents and the mass of the top quark dataset are shown in Figure 2. The generated distributions of the constituents closely match the distribution of the data. While the double peak structure of the mass distribution is clearly visible, the model is not able to produce the peak quite as sharp as in the dataset. In Table 1, DeepTreeGAN is compared to other state-of-the-art GANs and found to be competitive in this context. Diffusion and flow-matching based models [29, 34] have

Table 1: Comparison of the proposed DeepTreeGAN to EPiC-GAN [33] and MDMA [27]. The metrics were computed on a 25k event hold-out sample of the `JetNet` 150 top quark dataset. The 'Limit' row gives the measured in-sample distance by bootstrapping the hold-out dataset. Lower is better for all metrics. Values taken from [27, Table 1]. The metrics are provided by the `JetNet` library [45]. The best performance is printed in bold.

Model	$W_1^M(\times 10^3)$	$W_1^P(\times 10^3)$	$W_1^{EFP}(\times 10^5)$	$FPD(\times 10^4)$
Limit	0.42 ± 0.09	0.12 ± 0.04	1.22 ± 0.32	1.2 ± 0.6
EPiC-GAN	0.69 ± 0.08	0.65 ± 0.03	2.67 ± 0.39	22 ± 1
MDMA	$\mathbf{0.57 \pm 0.09}$	0.10 ± 0.02	$\mathbf{2.12 \pm 0.64}$	5.3 ± 0.9
DeepTreeGAN	1.49 ± 0.04	0.13 ± 0.02	5.01 ± 0.08	$\mathbf{3.4 \pm 0.7}$

recently provided leading results for the JetNet dataset, but sample production is generally slower compared to GAN approaches. The DeepTreeGAN code and weights are available on GitHub.[3]

4 Conclusion

In this work, the DeepTreeGAN model, first introduced in [1], was significantly extended to model up to 150 jet constituents of the `JetNet` dataset. To this end, a PC-based critic has been developed, featuring a pooling operation (Section 2.2), that allows the iterative downscaling of PCs. Through its significant advantages, this novel pooling implementation may prove useful not only for generative tasks, but also any PC regression or classification task. This extension represents a significant milestone in scaling this model to handle even larger point clouds, such as particle showers in high-granularity calorimeters.

Acknowledgement

References

[1] Moritz A.W. Scham et al. "DeepTreeGAN: Fast Generation of High Dimensional Point Clouds". Submitted to EPJ Web of Conferences CHEP, `https://arxiv.org/abs/2311.12616`. May 2023.

[2] S. Agostinelli et al. "GEANT4–a simulation toolkit". In: *Nucl. Instrum. Meth. A* 506 (2003), p. 250. DOI: `10.1016/S0168-9002(03)01368-8`.

[3] Johannes Albrecht et al. "A Roadmap for HEP Software and Computing R&D for the 2020s". In: *Computing and Software for Big Science* 3.1 (Mar. 2019), p. 7. DOI: `10.1007/s41781-018-0018-8`. arXiv: `1712.06982 [physics.comp-ph]`. URL: `https://doi.org/10.1007%2Fs41781-018-0018-8`.

[4] G. Apollinari et al. "High Luminosity Large Hadron Collider HL-LHC. High Luminosity Large Hadron Collider HL-LHC". In: *CERN Yellow Report* (2015). Chapter 1 in High-Luminosity Large Hadron Collider (HL-LHC) : Preliminary Design Report, pp. 1–19. DOI: `10.5170/CERN-2015-005.1`. arXiv: `1705.08830`. URL: `https://cds.cern.ch/record/2120673`.

[5] CMS Offline Software and Computing. *CMS Phase-2 Computing Model: Update Document*. Tech. rep. Geneva: CERN, 2022. URL: `https://cds.cern.ch/record/2815292`.

[6] The CMS collaboration. "The CMS HGCAL detector for HL-LHC upgrade". In: *5th Large Hadron Collider Physics Conference*. Aug. 2017. DOI: `10.48550/arXiv.1708.08234`. arXiv: `1708.08234 [physics.ins-det]`.

---

[3]`https://github.com/DeGeSim/nips23DeepTreeGANv2`

[7] Ian Goodfellow et al. "Generative Adversarial Networks". In: *Commun. ACM* 63.11 (Oct. 2020). Ed. by Z. Ghahramani et al., p. 139. ISSN: 0001-0782. DOI: 10.1145/3422622. arXiv: 1406.2661 [stat.ML]. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[8] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: (2013). DOI: 10.48550/arxiv.1312.6114. arXiv: 1312.6114 [stat.ML].

[9] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. In: *Comput. Software Big Sci.* 4.1 (Sept. 2017). DOI: 10.1007/s41781-017-0004-6. eprint: 1701.05927.

[10] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. "Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters". In: *Phys. Rev. Lett.* 120.4 (2018), p. 042003. DOI: 10.1103/PhysRevLett.120.042003. arXiv: 1705.02355 [hep-ex].

[11] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. "CaloGAN: Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks". In: *Phys. Rev. D* 97.1 (2018), p. 014021. DOI: 10.1103/PhysRevD.97.014021. arXiv: 1712.10321 [hep-ex].

[12] Martin Erdmann et al. "Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks". In: *Comput. Softw. Big Sci.* 2.1 (2018), p. 4. DOI: 10.1007/s41781-018-0008-x. arXiv: 1802.03325 [astro-ph.IM].

[13] Martin Erdmann, Jonas Glombitza, and Thorben Quast. "Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network". In: *Comput. Softw. Big Sci.* 3.1 (2019), p. 4. DOI: 10.1007/s41781-018-0019-7. arXiv: 1807.01954 [physics.ins-det].

[14] F. Carminati et al. "Three dimensional Generative Adversarial Networks for fast simulation". In: *J. Phys. Conf. Ser.* 1085.3 (2018), p. 032016. DOI: 10.1088/1742-6596/1085/3/032016.

[15] Georges Aad et al. "AtlFast3: The Next Generation of Fast Simulation in ATLAS". In: *Comput Softw Big Sci* 6 (2022), p. 7. DOI: https://doi.org/10.1007/s41781-021-00079-7.

[16] Benno Käch et al. "JetFlow: Generating Jets with Conditioned and Mass Constrained Normalising Flows". In: (2022). arXiv: 2211.13630 [hep-ex].

[17] Benno Käch, Dirk Krücker, and Isabell Melzer-Pellmann. *Point Cloud Generation using Transformer Encoders and Normalising Flows*. 2022. arXiv: 2211.13623 [hep-ex].

[18] Simon Schnake, Dirk Krücker, and Kerstin Borras. *Generating Calorimeter Showers as Point Clouds*. Dec. 2022. URL: https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_77.pdf.

[19] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20 (2009), pp. 61–80. URL: https://api.semanticscholar.org/CorpusID:206756462.

[20] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *IEEE Transactions on Neural Networks.* 5.1 (2016), pp. 61–80. DOI: 10.1109/TNN.2008.2005605. arXiv: 1609.02907.

[21] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. *3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions*. 2019. arXiv: 1905.06292 [cs.CV].

[22] Raghav Kansal et al. *JetNet*. Version 2. Aug. 2022. DOI: 10.5281/zenodo.6975118. URL: https://doi.org/10.5281/zenodo.6975118.

[23] Raghav Kansal et al. *JetNet150*. Version 2.0.0. Aug. 2022. DOI: 10.5281/zenodo.6975117. URL: https://doi.org/10.5281/zenodo.6975117.

[24] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. "A Brief Introduction to PYTHIA 8.1". In: *Comput. Phys. Commun.* 178 (2008), p. 852. DOI: 10.1016/j.cpc.2008.01.036. arXiv: 0710.3820 [hep-ph].

[25] Erik Buhmann. "Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed". In: May 2020.

[26] Erik Buhmann et al. "Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network". In: *EPJ Web Conf.* 251 (2021), p. 03003. DOI: 10.1051/epjconf/202125103003. eprint: 2102.12491.

[27] Benno Käch and Isabell Melzer-Pellmann. *Attention to Mean-Fields for Particle Cloud Generation*. 2023. arXiv: 2305.15254 [hep-ex].

[28]   Vinicius Mikuni and Benjamin Nachman. "Score-based generative models for calorimeter shower simulation". In: *Phys. Rev. D* 106.9 (2022), p. 092009. DOI: 10.1103/PhysRevD. 106.092009. arXiv: 2206.11898 [hep-ph].

[29]   Vinicius Mikuni, Benjamin Nachman, and Mariel Pettee. "Fast Point Cloud Generation with Diffusion Models in High Energy Physics". In: (Apr. 2023). arXiv: 2304.01266 [hep-ph].

[30]   Raghav Kansal et al. "Particle Cloud Generation with Message Passing Generative Adversarial Networks". In: (2022). arXiv: 2106.11535 [cs.LG].

[31]   Raghav Kansal et al. "Evaluating generative models in high energy physics". In: *Physical Review D* 107.7 (Apr. 2023). DOI: 10.1103/physrevd.107.076017. URL: https://doi. org/10.11032Fphysrevd.107.076017.

[32]   Matthew Leigh et al. *PC-JeDi: Diffusion for Particle Cloud Generation in High Energy Physics*. 2023. arXiv: 2303.05376 [hep-ph].

[33]   Erik Buhmann, Gregor Kasieczka, and Jesse Thaler. "EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets". In: (Jan. 2023). arXiv: 2301.08128 [hep-ph].

[34]   Erik Buhmann et al. *EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion*. 2023. arXiv: 2310.00049 [hep-ph].

[35]   Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[36]   Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive are Graph Attention Networks?" In: *International Conference on Learning Representations*. 2022. URL: https://openreview. net/forum?id=F72ximsx7C1.

[37]   Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[38]   Juho Lee et al. *Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks*. Tech. rep. arXiv:1810.00825 [cs, stat] type: article. arXiv, May 2019. DOI: 10.48550/arXiv.1810.00825. arXiv: 1810.00825 [cs.LG]. URL: http://arxiv.org/ abs/1810.00825 (visited on 12/07/2023).

[39]   Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *6th International Conference on Learning Representations*. 2018. arXiv: 1802.05957 [cs.LG]. URL: https://openreview.net/forum?id=B1QRgziT-.

[40]   Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[41]   Jae Hyun Lim and Jong Chul Ye. *Geometric GAN*. 2017. arXiv: 1705.02894 [stat.ML].

[42]   Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. DOI: 10.48550/arXiv.1412.6980. eprint: 1412.6980.

[43]   Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[44]   F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[45]   Raghav Kansal et al. "JetNet: A Python package for accessing open datasets and benchmarking machine learning methods in high energy physics". In: *Journal of Open Source Software* 8.90 (Oct. 2023), p. 5789. DOI: 10.21105/joss.05789. URL: https://joss.theoj.org/ papers/10.21105/joss.05789.