MACHINE LEARNING Science and Technology

OPEN ACCESS

RECEIVED
1 April 2022

REVISED
17 June 2022

ACCEPTED FOR PUBLICATION
27 June 2022

PUBLISHED
13 July 2022

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Particle-based fast jet simulation at the LHC with variational autoencoders

Mary Touranakou[1,2,*], Nadezda Chernyavskaya[1], Javier Duarte[3], Dimitrios Gunopulos[2], Raghav Kansal[3], Breno Orzari[4], Maurizio Pierini[1], Thiago Tomei[4] and Jean-Roch Vlimant[5]

[1] European Organization for Nuclear Research (CERN), CH-1211 Geneva 23, Switzerland
[2] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, 157 72, Greece
[3] University of California San Diego, La Jolla, CA 92093, United States of America
[4] Universidade Estadual Paulista, São Paulo/SP—CEP 01049-010, Brazil
[5] California Institute of Technology, Pasadena, CA 91125, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: mtouranakou@di.uoa.gr

Keywords: generative models, sparse data simulation, particle physics

Abstract

We study how to use deep variational autoencoders (VAEs) for a fast simulation of jets of particles at the Large Hadron Collider. We represent jets as a list of constituents, characterized by their momenta. Starting from a simulation of the jet before detector effects, we train a deep VAE to return the corresponding list of constituents after detection. Doing so, we bypass both the time-consuming detector simulation and the collision reconstruction steps of a traditional processing chain, speeding up significantly the events generation workflow. Through model optimization and hyperparameter tuning, we achieve state-of-the-art precision on the jet four-momentum, while providing an accurate description of the constituents momenta, and an inference time comparable to that of a rule-based fast simulation.

1. Introduction

At particle colliders, collimated sprays of particles are produced as a consequence of the parton shower and hadronization processes typical of quantum chromo dynamics. These sprays of particles, called *jets*, are reconstructed applying a recombination clustering algorithm, exploiting physics-inspired metrics such as the anti-k_t distance [1]. Often, jets are clustered from energy deposits recorded in the electromagnetic and hadronic calorimeters of a particle detector. At the Large Hadron Collider (LHC), jets can be clustered from a list of reconstructed particles, the so-called particle-flow (PF) candidates [2, 3]. In this case, jets would be sparse sets of objects (the constituents), each represented by its momentum[6] and possibly a set of auxiliary features, such as the nature of the particle (electron, muon, etc), its electric charge, etc.

A time- and resource-effective strategy to simulate jet production is a fundamental asset for physics studies at the CERN LHC. Whether testing predictions of the standard model (SM), searching for evidence of physics beyond the SM, or assessing systematic uncertainties associated to a given measurement, physicists rely on an accurate simulation of the full collision process and of the detector response. This implies the need for a detailed simulation of a chain of very different steps, from the proton collision to the generation of the collected signal in the detector sensors. Typically, physicists simulate datasets at least 10 times larger than the amount of collected data, so that the precision on the final measurement is not limited by the amount of simulated data at hand.

[6] As common for collider physics, we use a Cartesian coordinate system with the z axis oriented along the beam axis, the x axis on the horizontal plane, and the y axis oriented upward. The x and y axes define the transverse plane, while the z axis identifies the longitudinal direction. The azimuth angle ϕ is computed with respect to the x axis. The polar angle θ is used to compute the pseudorapidity $\eta = -\log(\tan(\theta/2))$. The transverse momentum (p_T) is the projection of the particle momentum on the (x, y) plane. We fix units such that $c = \hbar = 1$.

© 2022 The Author(s). Published by IOP Publishing Ltd

A typical high-energy physics (HEP) simulation software relies on the GEANT4 [4] library to model the interaction of particles traversing the detector material. This approach, based on Monte Carlo (MC) techniques, provides a typical accuracy at the percentage level, but it comes at a high cost in terms of computing resource utilization. At the LHC, the simulation workflow consumes up to ∼50% of the total computing resources of an experiment. With the amount of collected data increasing, the need for MC simulation is going beyond what the available computing infrastructure could sustain. Projected to the planned high-luminosity LHC upgrade, this trend will eventually become unsustainable [5]. Jet simulation is one of the most expensive tasks, since jets are very abundant in LHC collisions and are made of many particles. Since the simulation of each of these particles is a demanding operation, the possibility of simulating all particles in a jet at once would be a major improvement. The main difficulty of this task is to match state-of-the-art accuracy, which also depends on the specific use case, e.g. which quantity a specific analysis uses. For instance, an algorithm reproducing the jet kinematics but not describing the angular distribution of the particles in the jet might be suitable for an analysis needing a good description of the jet momentum (e.g. a dijet resonance search), but not for an analysis exploiting jet substructure techniques (e.g. an all-jet diboson resonance search). Certainly, an algorithm describing every aspect of jet physics would be an ideal solution.

As a first step, a typical LHC simulation makes use of an *event generator*, modeling a proton-proton collision, the consequent production of quarks and gluons (among other particles), and their hadronization into jets of particles. Since no detector response is involved, this step typically requires a relatively modest amount of computing resources[7]. In addition, its content is independent of experimental aspects (reconstruction software, detector configuration, etc), so that a dataset of these *generator-level* events can be stored for long term and used many times (as is the case for the CMS experiment). Computing requirements significantly increase when detector effects are to be taken into account. At first, one typically uses GEANT4 to simulate the detector response. Then, the reconstruction software runs on the event and produces the objects (e.g. the PF candidates for CMS), eventually clustered into jets. These two steps could be bypassed by a *jet response function*, taking as input the list of jet constituents at generator level and returning the list of constituents at reconstruction level. In this paper, we aim at approximating this jet response function with a variational autoencoder (VAE), trained using generator-level jets as input and the corresponding reconstruction-level jets as a target. We represent a jet as a list of particles' momenta. Doing so, the VAE returns a reconstructed jet in a format that is already compatible with a typical PF-based analysis software. A different approach to the problem of data sparsity consists of representing the jet as a point cloud, as proposed for many HEP-specific problems [7]. We investigated that approach when training a generative adversarial network (GAN) [8] to generate the list jet constituent momenta from random numbers. A similar approach was presented in [9], where a graph VAE was used to generate detector *hits* in a jet, from which jet constituents could be reconstructed using standard rule-based algorithms, e.g. PF reconstruction [2, 3]. This work has many common points with [9], with two main differences: we do not use graph architectures, and we aim at learning the detector response and bypassing the standard rule-based reconstruction algorithms (to offer further speed up of the simulation process). We do so by taking the reconstructed jet as a target. In this respect, our algorithm could be used to replace detector parametrization approaches now used in Fast Simulation tools [10–12], while the algorithm of [9] aims at speeding up a GEANT-based full simulation. In the future, both approaches will be useful to HEP experiments and, most likely, the ultimate generative model will emerge from a combination of the two.

This paper is organized as follows: section 2 discusses related work. Sections 3 and 4 describe the benchmark dataset and the model architecture, respectively. A strategy to apply the model to a realistic use case is discussed in section 5. Training results are discussed in section 6. Conclusions are given in section 7.

2. Related works

In the recent past, several studies explored the possibility of speeding up the data simulation process using generative models based on deep neural networks (NNs). In particular, convolutional neural networks (CNNs) have been proposed to generate single-particle showers in a calorimeter [13–19], full jets at the LHC [20–22], multi-dimensional functions of kinematic quantities [23, 24], event kinematics at colliders [25, 26], and cosmic ray showers [27]. Both GANs [28–30] and VAEs [31] were considered.

These studies clearly demonstrate that integrating deep generative models in the data simulation workflows of HEP experiments could lead to an important saving in terms of computing resources. But there is an objective difficulty when scaling up these proof-of-concept solutions to production-ready

[7] This picture could in principle change if next-to-leading order precision would be adopted as a default. On the other hand, ongoing work on parallelizing event generation libraries on GPUs [6] may compensate for this precision increase.

simulation tools. The main problem lies in the complexity of a typical HEP detector, characterized by detector elements with different technology and geometry, partially overlapping with each other and with passive material (e.g. absorbers in calorimeters) in between. As a consequence of this, a typical HEP dataset consists of a sparse set of energy deposits, which often cannot be represented as a regular grid of pixels. Future detectors will be characterized by higher granularity, with small-size sensors designed to resolve hits from individual particles in dense environments, such as jet cores. This will make the sparsity of the event even more complicated. This is the main reason why most of the great ideas based on CNNs had so far a little impact on HEP experiments. Instead, other approaches were explored as alternatives to CNNs, e.g. a recurrent neural network (RNN) trained adversarially [32], graph NNs [8, 9, 33], or normalizing flows [34]. Similar issues are present in other domains, e.g. galaxy simulation in cosmology [35].

In this paper, we investigate an alternative strategy to overcome difficulties with the peculiar nature of HEP data. In previous studies, we discussed how to sample jets as sparse data from a probability density function, modeled using deep generative models. To this purpose, we considered both GANs [8] and VAEs [36]. Here we take a different approach, in which the input is a generator-level jet (as opposed to a vector of random coordinates in some latent space) and the aim of the training is to learn a morphing function from generator to reconstruction level.

The strategy is similar to what is discussed in [37], where a similar approach is followed to morph a set of analysis-specific features from generator- to reconstruction-level precision.

3. Dataset

The reference dataset consists of jets generated in $pp \rightarrow WW$ collisions at a center-of-mass energy $\sqrt{s} = 13$ TeV. The W bosons are forced to decay to quarks, that then shower to jets. The event generation is performed using PYTHIA8 [38]. The generated list of particles is passed to DELPHES [10], which applies detector effects using the CMS DELPHES description. At this stage, additional collision events are superimposed to the generated collision, to mimic the effect of so-called *pileup*. The number of collisions is randomly sampled from a Poisson distribution with expectation value set to 50, in agreement with the expected LHC running conditions for Run 3. The DELPHES PF reconstruction algorithm is applied to the event, returning the list of reconstructed particles. Reconstructed particles are required to have $p_T > 250$ MeV and be within $|\eta| < 3.2$. These particles are then clustered into jets using the anti-k_t [1] algorithm with jet-size parameter $R = 0.5$. Jets with $p_\mathrm{T} > 200$ GeV and within $|\eta| < 2.5$ are retained. These jets represent the target dataset. With the same setting, generator-level jets are clustered from the stable and detectable particles produced in the collision, before detector effects are taken into account. These jets represent the input dataset.

Target jets within p_T and η acceptance are matched to input jets minimizing the angular distance $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$. An input-target pair is formed, taking the closest input jet to each target jet. For both input and target jets, constituents are ordered by decreasing p_T and the first 50 particles are retained. When fewer particles are present, the list is zero-padded. The list contains the momentum of each constituent in Cartesian coordinates (p_x, p_y, p_z). The constituent mass is implicitly assumed to be zero. The main advantages of this specific choice are: it retains information of the position of the jet in the detector (as opposed to local coordinate choice); the distribution of these quantities is unbounded and symmetric around 0, which makes the learning process easier. In particular, we avoid issues related to the periodicity of ϕ and hard-threshold at boundaries (e.g. on p_T).

We apply feature-dependent standardization by subtracting the mean and scaling the features to unit variance. During early stages of this work, we verified that these choices help the model training to converge to more accurate configurations of the network weights. After this pre-processing, each jet is represented as a 2D array of 3×50 numbers. The whole dataset includes $\sim$1.7M jets. We split these data in three parts: 60% for training, 20% for validation, and 20% for testing. The dataset is published on Zenodo [39].

4. VAE architecture

The VAE architecture is schematically shown in figure 1. The encoder receives a single-channel 3×50 table, which is processed by three 2D convolutional layers, with 32 3×5 kernels, 64 1×5 kernels, and 128 1×5 kernels, respectively. The stride is set to 1 and zero padding is used when the kernel arrives at the edge of the table. The output tensor is flattened and passed to two dense layers, with 640 and 150 nodes, respectively. From the second layer, two 20-dimensional vectors are derived, corresponding to the mean μ and log-variance values of the latent space variables z. These values are used to define the Gaussian prior function from which a set of z values is sampled and passed to the decoder. The decoder architecture mirrors the encoder, with the Conv2D layers being replaced by ConvTrans2D layers. Leaky ReLU activation functions

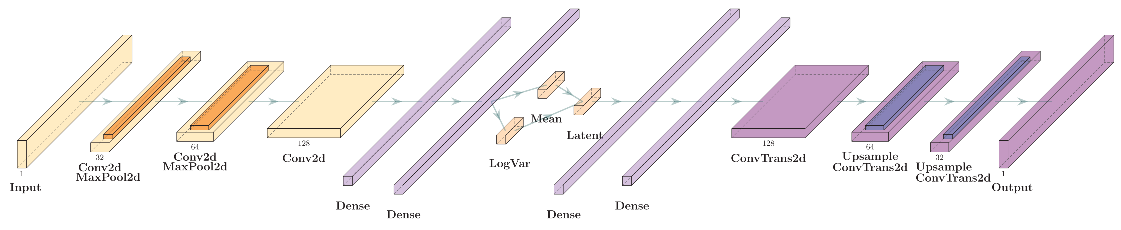

Figure 1. Graphical representation of the VAE architecture.

are used across the whole architecture with the coefficient of the negative slope set to 0.1, except for the encoder and decoder output layers, for which a linear activation function is used. Max pooling with the kernel size 1×2 and stride of 2 is applied after each of the first two convolution operations in the encoder. Respectively, two upsampling operations are used in the decoder, each placed before the last two ConvTrans2D layers with a bilinear interpolation scheme. Dropout is added to the output of the first dense layer in the encoder and to the output of the two dense layers in the decoder with a dropout rate of 0.2. The final architecture of the model was optimized starting from a much larger convolutional autoencoder with about 10 million trainable parameters based on the network developed in [36]. While experimenting with the number and size of convolutional kernels and dense layers, the dimension of the latent space, and the activation functions, we then reduced the size of the network down to about 300 000 trainable parameters while matching the performance. The deep learning (DL) model is implemented in PyTorch [40].

The model is trained using an input dataset, containing the list of particles at generator level, and a target dataset, containing the corresponding list after detector effects and event reconstruction. In this way, the model is trained to regress the detector response function starting from a generator-level jet, i.e. it corresponds to a Fast Simulation software in HEP computing literature. For this kind of application, in the similar phase space as the one used in this study, a typical state-of-the-art simulation has a 10% accuracy on jet kinematic distributions such as the momentum and pseudorapidity [10]. At the same time, the Fast Simulation software has difficulty faithfully reproducing detailed jet particle composition, and larger discrepancies are observed when comparing variables that depend on it, such as jet mass [41].

The model training is performed minimizing a domain-specific loss function:

$$L^{\mathrm{VAE}} = \frac{1}{N}\sum_{i=0}^{N}\left[\beta D^{i}_{\mathrm{KL}} + (1-\beta)\left(L^{i}_{\mathrm{R}} + \alpha_m\left(m^{i}_{\mathrm{jet}} - \hat{m}^{i}_{\mathrm{jet}}\right)^2 + \alpha_{p_{\mathrm{T}}}\left({p_{\mathrm{T}}}^{\mathrm{jet},i} - \hat{p}_{\mathrm{T}}^{\mathrm{jet},i}\right)^2\right)\right], \tag{1}$$

where N is the dataset size, L_{R} is the reconstruction loss (i.e. a distance between the target and the output), D_{KL} is the Kullback-Leibler (KL) divergence regularizer usually employed to force the data distribution in the latent space to a multi-dimensional Gaussian with unitary covariance matrix [42], and β is a parameter that controls the relative importance of the two terms [43]. The reconstruction loss L_{R} is computed using the permutation-invariant Chamfer loss [44]:

$$L_{\mathrm{R}} = \sum_i \min_j (p_i - \hat{p}_j)^2 + \sum_j \min_i (p_i - \hat{p}_j)^2, \tag{2}$$

where p_i is the feature for the ith target particle, and $\hat{p}_j$ is the corresponding quantity for the jth output particle. By construction, this quantity is invariant under the permutation of the input or output particle lists. We also experimented with a mean squared error (MSE) loss, observing typically worse results, as using MSE implies breaking permutation invariance of the data, since the reconstructed particles in the jets have no intrinsic ordering.

In equation (1), ${p_{\mathrm{T}}}^{\mathrm{jet}}$ and m^{jet} are the transverse momentum and mass of a target jet, respectively. Jet features are computed from the momenta of the target-jet constituents, while $\hat{p}_{\mathrm{T}}^{\mathrm{jet}}$ and $\hat{m}^{\mathrm{jet}}$ are the corresponding quantities computed from the model output. The coefficients α_m and $\alpha_{p_{\mathrm{T}}}$were chosen based on a grid search in the range between 0.1 and 10. The best learning configuration corresponds to the values $\alpha_m = 1.0$ and $\alpha_{p_{\mathrm{T}}} = 0.1$, such that the reconstruction loss L_{R} and the jet-p_T and jet-mass MSE constraints in equation (1) have similar magnitudes. The expression in equation (1) is only one of the possible ways one could enforce kinematic constraints on the jet generator. Similar approaches have been followed in previous works, e.g. for particle energy in GAN-based single-particle generators [14, 45]. The main difference here is that the quantity on which the constraint is applied on is analytically computed from the output list, as

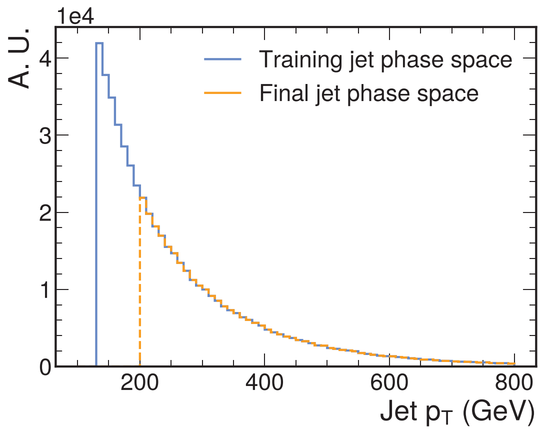

Figure 2. Jet phase space in the training sample extended to $p_T > 130$ GeV is shown in blue, final jet phase space with the selection of $p_T > 200$ GeV is shown in orange.

opposed of being regressed from an image. We also tried other combinations of kinematic constraints, e.g. the three momentum components in Cartesian coordinates, observing similar or worse results after training.

5. Target application

The aim of this work is to create a fast-simulation workflow for an analysis demanding a large sample of multijet events. As a reference, we consider the case of dijet resonance searches, where a search for a localized excess on top of a smooth background is performed. Traditionally, these searches are carried on through maximum-likelihood fit, in which the background is analytically modeled [46, 47]. Large samples of simulated multijet events are used to find an adequate model. In addition, a novel simulation-assisted strategy uses ratios of simulated distributions to avoid the need of a specific background analytical model [48]. Also in this case, having at hand a large simulated sample is crucial. Finally, the proposed strategy could be crucial to move the default event-generation precision to next-to-leading order, trading simulation computing time for generation computing time. This would be also relevant for analyses exploiting angular information about the dijet system [49]. Similar considerations hold for multijet searches.

The reference analysis requires an accurate model of jet kinematic properties for jets momenta larger than 200 GeV. As we will see, this can be achieved. But some care is required to model the sharp threshold at 200 GeV. As we experienced in the early stages of this study, an ML-based simulation struggles to model such a sharp threshold. As a solution, we extend the jet phase space in the training sample down to $p_T > 130$ GeV and we apply the selection of $p_T > 200$ GeV on the jets obtained as output of the VAE, as shown in figure 2. A similar problem exists for the p_T threshold of the jet constituents. In this case, we also extend the p_T range of the predicted model down to $p_T > 0$ MeV and apply the selection $p_T > 250$ MeV afterwards. Similar considerations hold for η, where the acceptance requirements are imposed on the predicted jets after inference.

As discussed in section 6, this setup provides an adequate description of the jet kinematics, but it fails in providing an accurate description of jet substructure. In this respect, the proposed model cannot be extended to other dijet resonance analyses, e.g. diboson resonance searches, where the accurate modelling of jet substructure is crucial. One could have then enforced a limited scope from the beginning and avoided generating jet constituents, working directly at the level of jet four momenta. However, we see two added values in working with jet constituents: on the one hand, we obtain a faithful description of the jet mass, the most crucial jet-substructure high-level feature; on the other hand, we establish a baseline model which could further improve to also model jet substructure. This will be the subject of future studies exploiting a permutation-equivariant graph architecture.

6. Results

We train all models using the Adam [50] optimizer with a learning rate of 0.0001 for 300 epochs. The training was repeated for several values of β, and the value corresponding to the best agreement between the output and target ($\beta = 1/9$) was chosen. During the training, we monitor the values of the total loss and its individual components evaluated on the training and validation datasets to check for overtraining.

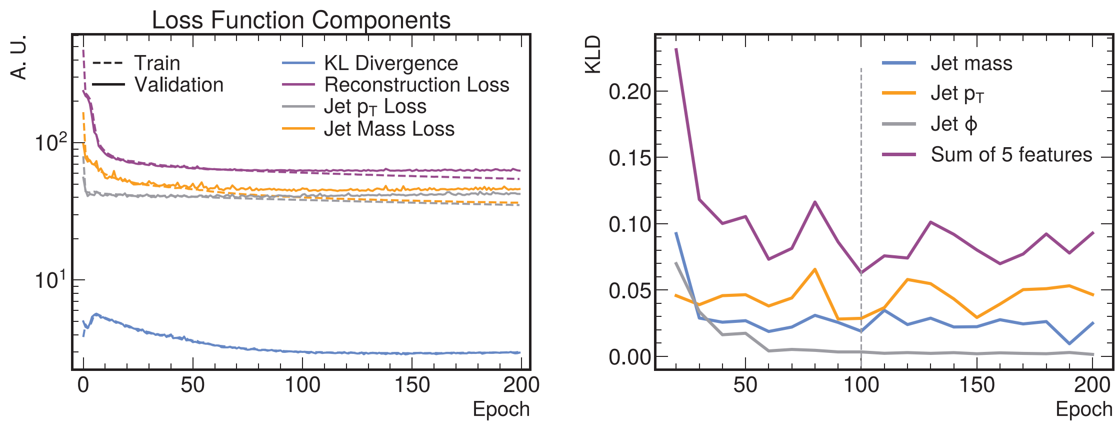

Figure 3. (Left) Evolution of various contributions to the loss, evaluated on the training (dashed lines) and validation (solid lines) datasets, as a function of the training epoch. (Right) Evolution of the KLD evaluated on the testing dateset as a function of the epoch. The total KLD, a sum of the KLD for the 5 jet features (mass, p_T, energy, η, ϕ), and three individual components are shown. The dashed vertical line indicates the selected epoch for the final model, where both the validation loss and the KLD are at the minimum.

To quantitatively evaluate the performance of different training settings, we compare the values of the loss function evaluated on the validation dataset. However, while the loss function is inherently multi-objective, the multiple objectives are combined in an ensemble in equation (1), thus reducing the problem to a scalar optimization. This might lead to overfitting to some individual objectives despite overall good performance, and in turn make it more difficult to compare different learners and select the best one. Therefore, in addition to comparing the loss values between different models, we also use the symmetrized version of the KL divergence (KLD) [51] between probabilities of the predicted and the target jet-kinematic distributions (mass, p_T, energy, η, ϕ) to evaluate reconstruction capabilities of the models. The smaller the values of the KLD, the smaller is the error of using predicted jet-kinematic distributions instead of the true ones, thus, the model with smaller KLD is preferential. The KLD is computed every 10 epochs on the testing dataset, after rescaling the DL-predicted and target distributions so that the reconstructed distribution is contained in the $[0,1]$ range.

Figure 3(left) shows the evolution of various contributions to the loss function (see equation (1)), evaluated on the training and validation datasets, as a function of the epoch. After the epoch number 120, the values of the reconstruction loss and the jet-p_T and jet-mass MSE constraints in the loss are all starting to increase, indicating an overtraining. The monitored evolution of the KLD computed on the testing dataset for the three jet features (mass, p_T, ϕ) and the total KLD sum of 5 features (mass, p_T, energy, η, ϕ) is shown in figure 3(right). The values of the KLD components and the total sum reach a plateau after the epoch number 100. We select the best model based on the minimum values of both, the total and individual components of the validation loss, as well as the KLD, while ensuring no overtraining. Therefore, the model from epoch number 100 is chosen as the best.

We compare the distributions of the DL predicted and the target p_x, p_y, and p_z of the jet constituents in figure 4. These distributions are obtained applying the constituents acceptance thresholds (see section 3) to the list of particles which is output by the VAE, as discussed in section 5. We observe a good agreement between the model prediction and the target reconstruction.

The output list of particles is then used to analytically compute the jet kinematic properties. Figure 5 (figure 6) shows the distribution of the jet kinematic properties explicitly used (not used) in the likelihood. The jet acceptance thresholds (see section 5) are imposed on the jet p_T and η for both the target and output jets. In general, a good agreement is observed. The residual discrepancies between the model prediction and the target reconstruction are smaller than the modelling differences typically observed between the jets data and MC reconstruction [52–54]. Remarkably, once forced to learn the jet mass and transverse momentum components, the model learns to model the entire jet kinematics, including non-linear functions of the three quantities above. This aspect proves that the training process converges to a solution that preserves the main physics of the jet shower. At this stage, such a generator would be useful to generate events for most of the physics studies performed at the LHC.

In appendix, we show the distribution of the jet features in the entire generation phase space, i.e. without enforcing the jet $p_T > 200$ GeV requirement on the target and output jets. There, the problem of modeling

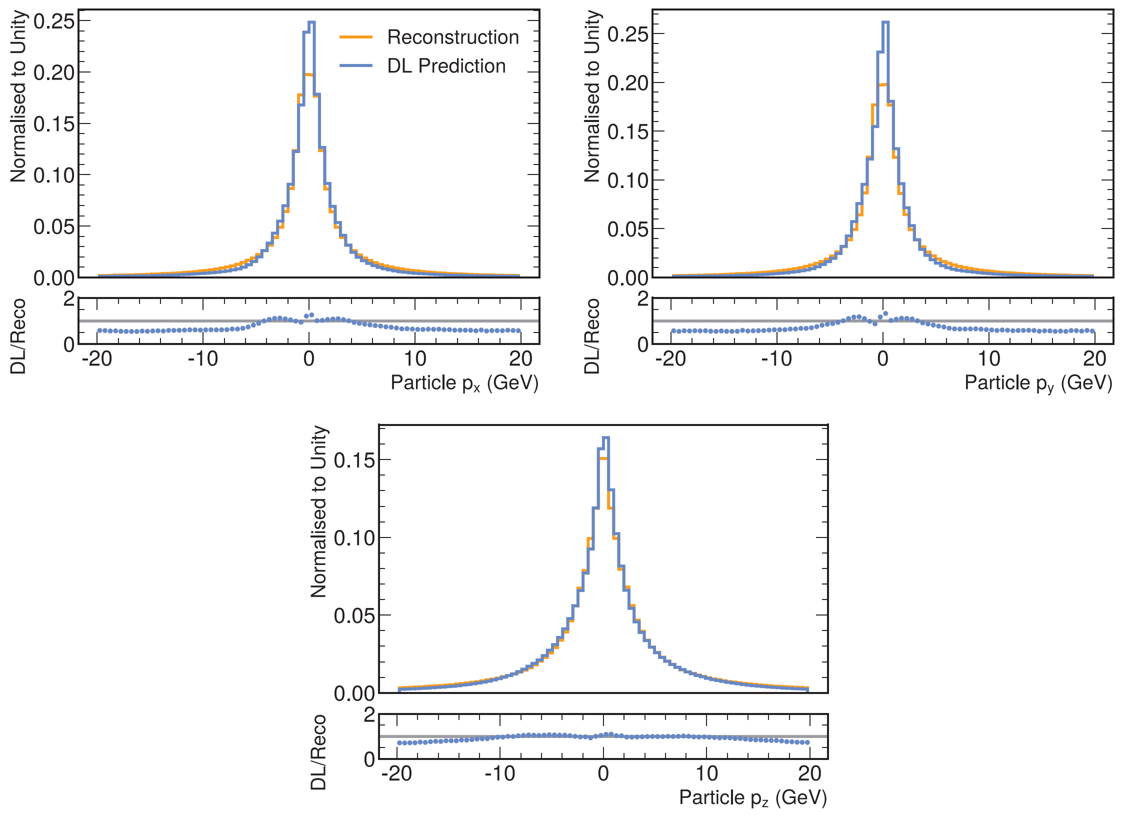

Figure 4. Comparison of the jet-constituent p_x (top-left), p_y (top-right), and p_z (bottom-center) distributions, for the output (DL prediction) and target (reconstruction) datasets. In the bottom panels, the ratio between the two distributions is shown. These distributions are obtained removing the zero-padding particles from the target list, and enforcing on the output of the DL model the same acceptance requirements that define the jet constituents (see section 3).

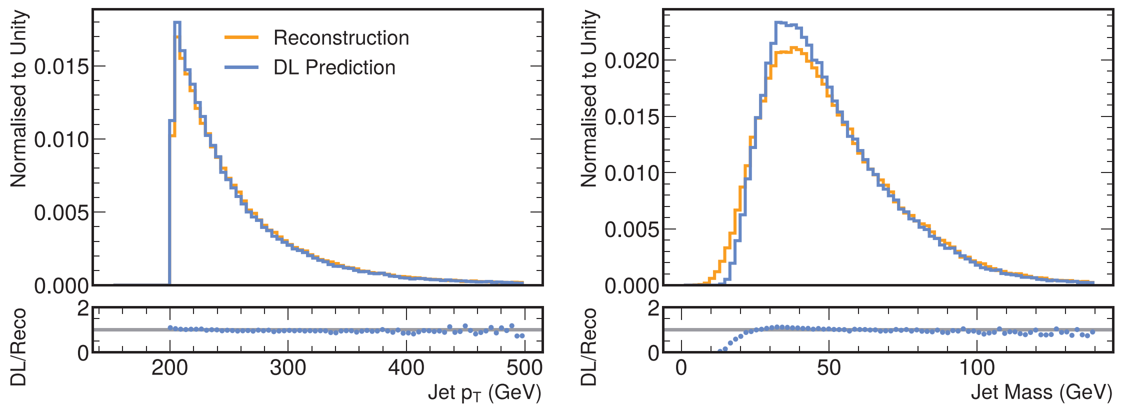

Figure 5. Comparison of the jet p_T (left) and mass (right) distributions, for the output (DL prediction) and target (reconstruction) datasets. In the bottom panels, the ratio between the two distributions is shown. These distributions are obtained removing the zero-padding particles from the target list, and enforcing on the output of the DL model the same acceptance requirements that define the jet constituents (see section 3).

the sharp p_T threshold is visible. Remarkably, this issue has little impact on the agreement observed in the jet mass distribution.

While our algorithm could serve the bulk of data analyses at the LHC, it still fails in faithfully describing the jet dynamics at constituents level. In fact, we verified that jet substructure quantities are not well reproduced. This is shown in figure 7, where the distribution of four momentum flows [55] are shown. The momentum flows are computed as $Flow_n = \sum_p \frac{p_T^p}{p_T^{Jet}}$, where the sum runs over all particles with distance $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$ from the jet axis falling within $(n-1)/4 \times R$ and $n/4 \times R$, where R is the jet size

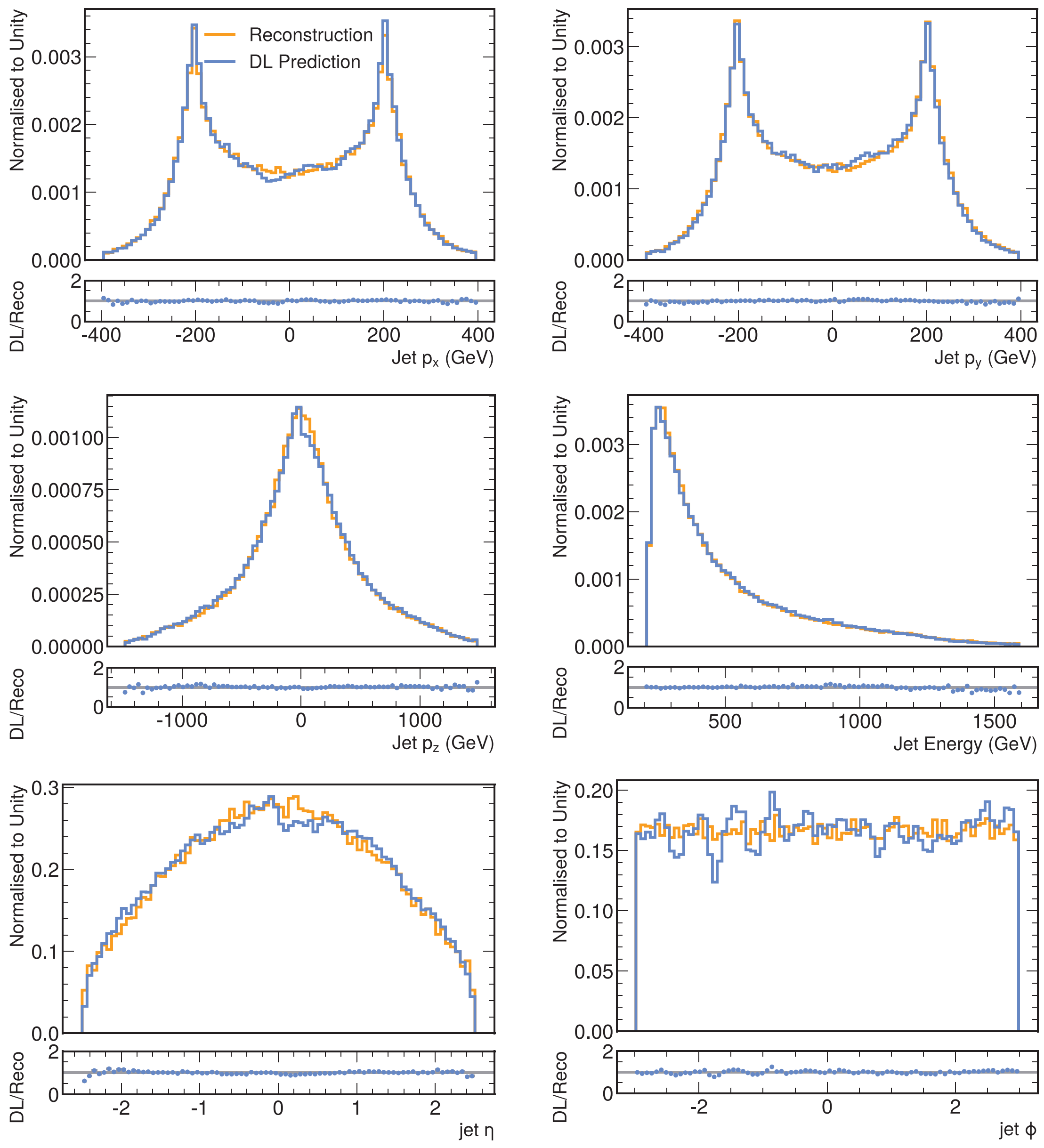

Figure 6. Comparison of the jet p_x (top-left), p_y (top-right), p_z (middle-left), energy (middle-right), η (bottom-left), and ϕ (bottom-right) distributions, for the output (DL prediction) and target (reconstruction) datasets. In the bottom panels, the ratio between the two distributions is shown. These distributions are obtained removing the zero-padding particles from the target list, and enforcing on the output of the DL model the same acceptance requirements that define the jet constituents (see section 3).

parameter. We tracked the cause of the mismodeling to the noise induced by zero-momentum fake particles (both in input and target), resulting from zero-padding the jet representation to a fixed dimension. This problem could be solved moving to a graph-based VAE architecture, as in [9]. Through a PyTorch Geometric [56] implementation, for example, one could avoid the need of zero-padding the datasets, possibly leading to a better representation of the jet substructure. This approach will be investigated in future studies.

The inference speed of the algorithm was measured running it on 1000 generator-level jets, and measuring the execution time. The test was performed calling the PyTorch library from a python script and running the algorithm on different hardware platforms. We obtain an inference time of 0.007 (0.004) seconds per event when running on a Intel Xeon Silver 4216 CPU (NVIDIA T4), while the traditional approaches typically require O(100) seconds per event of CPU time. This demonstrates how the proposed strategy represents a major speedup with respect to currently employed simulation algorithms. One should also keep in mind that this is an overestimate of the actual inference time in real world C++ computing environment, where tools such as ONNX run time [57] typically offer a further speed up with respect to a python environment. When running on a GPU, one could further increase the throughput by running the difference on a batch of all jets in an event.

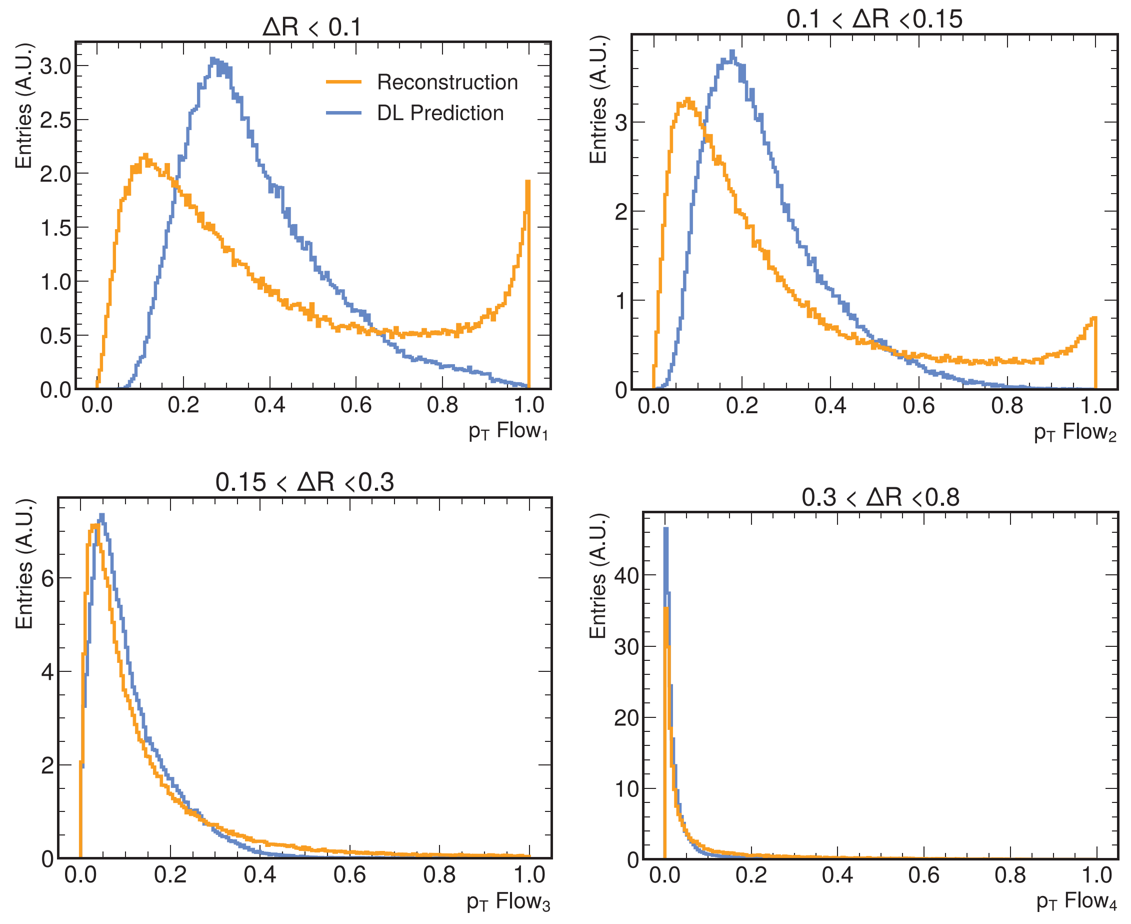

Figure 7. Distribution of the four *Flow$_n$* quantities (see text) for the output and target jets.

7. Conclusions

We present a jet fast-simulation algorithm based on a VAE, trained to learn the detector response function to a generator-level jet, represented as a list of particle momenta, and returning a list of reconstructed particle momenta. This algorithm correctly captures the reconstructed jet kinematics with high accuracy.

By bypassing the detector simulation and particle reconstruction step, an algorithm of this kind could be important to make simulation on demand a concrete possibility at the High-Luminosity LHC.

The main strength of the current algorithm is in its speed and high accuracy when modeling jet kinematic quantities, which makes it applicable to the majority of LHC physics studies. Its main limitation stands with the poor description of the jet substructure, a consequence of the noise induced by zero-momentum ghost particles introduced to equalize the length of the input particle list. A possible solution to this problem could be the use of a graph architecture with variable-length input, which we aim at investigating in the future.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.5281/zenodo.6047873 [39].

Acknowledgments

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 under Cooperative Agreement OAC-1836650, and by the LHC Physics Center at Fermi National Accelerator Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S.

Department of Energy (DOE). 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. D G is partially supported by the EU ICT-48 2020 project TAILOR (No. 952215). 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 Nos. DE-SC0011925, DE-SC0019227, and DE-AC02-07CH11359. B O and T T are supported by Grant #2018/25225-9, SãoPaulo Research Foundation (FAPESP). B O is also supported by Grant #2020/06600-3, São Paulo Research Foundation (FAPESP). This work was supported in part 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.

Appendix

In this appendix, we show the distribution of the jet features in the entire generation phase space before applying the jet p_T selection $p_T > 200$ GeV. Figure A.1 (figure A.2) shows the distribution of the jet kinematic properties explicitly used (not used) in the likelihood within the extended jet phase space before any selections.

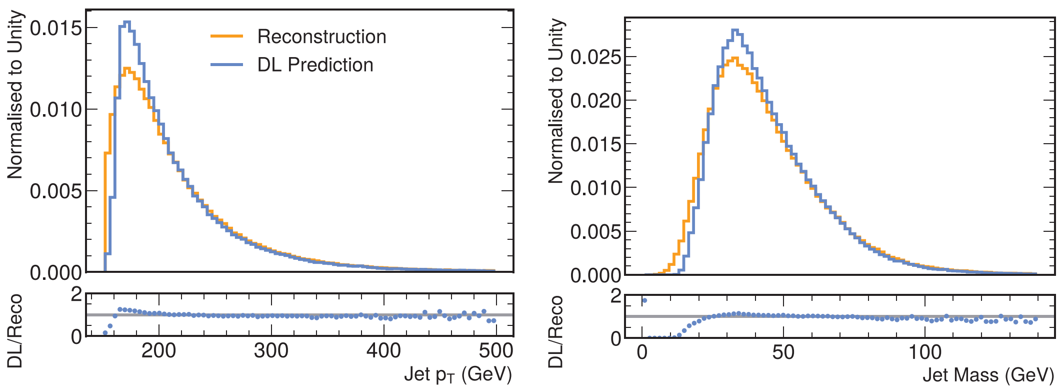

Figure A.1. Comparison of the jet p_T (left) and mass (right) distributions, for the output (DL prediction) and target (reconstruction) datasets before applying the jet p_T selection $p_T > 200$ GeV. In the bottom panels, the ratio between the two distributions is shown. These distributions are obtained removing the zero-padding particles from the target list, and enforcing on the output of the DL model the same acceptance requirements that define the jet constituents (see section 3).

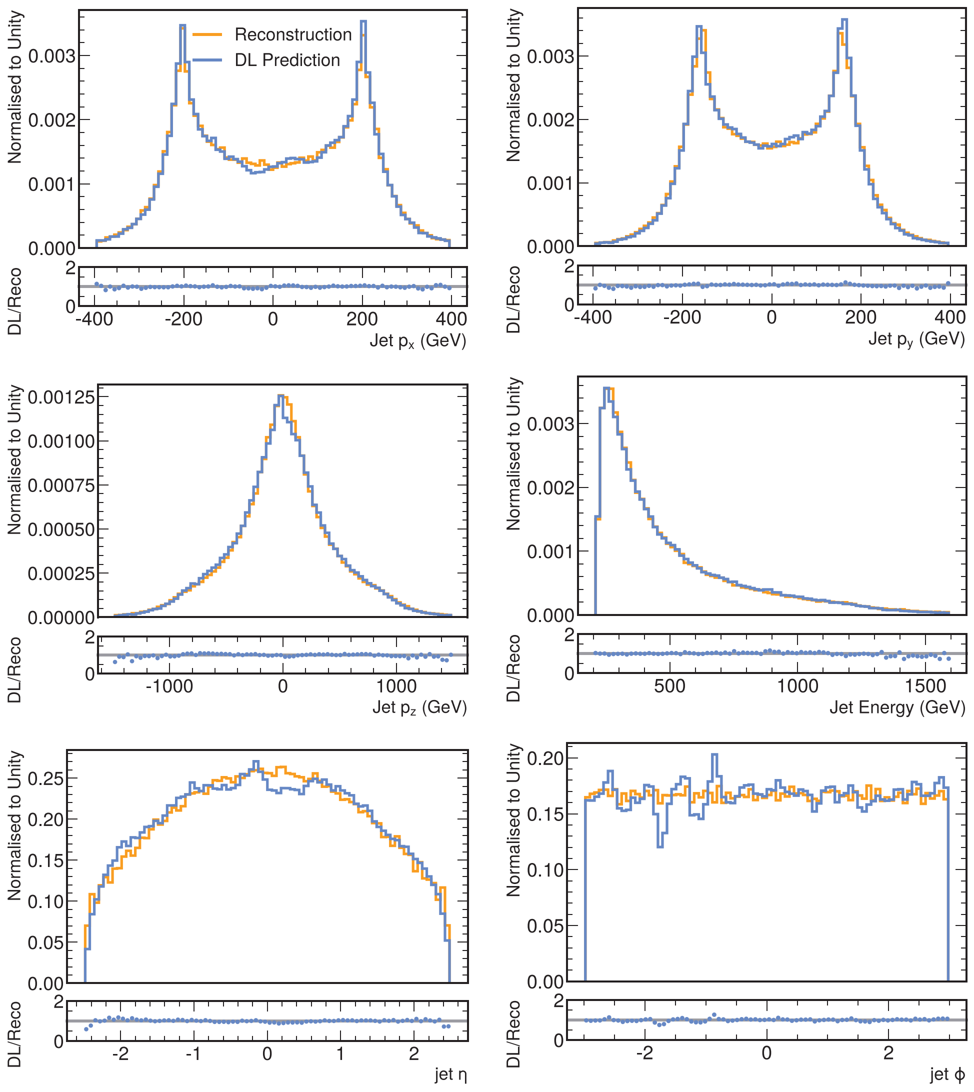

Figure A.2. Comparison of the jet p_x (top-left), p_y (top-right), p_z (middle-left), energy (middle-right), η (bottom-left), and ϕ (bottom-right) distributions, for the output (DL prediction) and target (reconstruction) datasets before applying the jet p_T selection $p_T > 200$ GeV. In the bottom panels, the ratio between the two distributions is shown. These distributions are obtained removing the zero-padding particles from the target list, and enforcing on the output of the DL model the same acceptance requirements that define the jet constituents (see section 3).

ORCID iDs

Mary Touranakou https://orcid.org/0000-0002-3682-3258
Nadezda Chernyavskaya https://orcid.org/0000-0002-2264-2229
Javier Duarte https://orcid.org/0000-0002-5076-7096
Dimitrios Gunopulos https://orcid.org/0000-0001-6339-1879
Raghav Kansal https://orcid.org/0000-0003-2445-1060
Breno Orzari https://orcid.org/0000-0003-4232-4743
Maurizio Pierini https://orcid.org/0000-0003-1939-4268
Thiago Tomei https://orcid.org/0000-0002-1809-5226
Jean-Roch Vlimant https://orcid.org/0000-0002-9705-101X

References

[1] Cacciari M, Salam G P and Soyez G 2008 The anti-k_t jet clustering algorithm *J. High Energy Phys.* **04** 063
[2] Sirunyan A M *et al* 2017 Particle-flow reconstruction and global event description with the CMS detector *J. Instrum.* **12** P10003
[3] Aaboud M *et al* 2017 Jet reconstruction and performance using particle flow with the ATLAS Detector *Eur. Phys. J.* C **77** 466
[4] Agostinelli S *et al* 2003 GEANT4: a simulation toolkit *Nucl. Instrum. Methods Phys. Res.* A **506** 250–303
[5] Albrecht J *et al* 2019 A roadmap for hep software and computing R & D for the 2020s *Comput. Softw. Big Sci.* **3** 7
[6] Hagiwara K, Kanzaki J, Li Q, Okamura N and Stelzer T 2013 Fast computation of MadGraph amplitudes on graphics processing unit (GPU) *Eur. Phys. J.* C **73** 2608
[7] Shlomi J, Battaglia P and Vlimant J R 2021 Graph neural networks in particle physics *Mach. Learn. Sci. Technol.* **2** 021001
[8] Kansal R *et al* 2020 Graph generative adversarial networks for sparse data generation in high energy physics *34th Conf. on Neural Information Processing Systems*
[9] Hariri A, Dyachkova D and Gleyzer S 2021 Graph generative models for fast detector simulations in high energy physics (arXiv:2104.01725)
[10] de Favereau J *et al* 2014 DELPHES 3: a modular framework for fast simulation of a generic collider experiment *J. High Energ. Phys.* **02** 057
[11] Sekmen S 2017 Recent developments in CMS fast simulation (arXiv:1701.03850)
[12] Aad G *et al* 2021 AtlFast3: the next generation of fast simulation in ATLAS vol 9 (arXiv:2109.02551)
[13] Paganini M, de Oliveira L and Nachman B 2018 Accelerating science with generative adversarial networks: an application to 3D particle showers in multilayer calorimeters *Phys. Rev. Lett.* **120** 042003
[14] Paganini M, de Oliveira L and Nachman B 2018 CaloGAN : simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks *Phys. Rev.* D **97** 014021
[15] Erdmann M, Glombitza J and Quast T 2019 Precise simulation of electromagnetic calorimeter showers using a wasserstein generative adversarial network *Comput. Softw. Big Sci.* **3** 4
[16] Salamani D *et al* 2018 Deep generative models for fast shower simulation in atlas *2018 IEEE 14th Int. Conf. on e-Science* (e-Science) p 348
[17] Belayneh D *et al* 2020 Calorimetry with deep learning: particle simulation and reconstruction for collider physics *Eur. Phys. J.* C **80** 688
[18] Buhmann E *et al* 2021 Getting high: high fidelity simulation of high granularity calorimeters with high speed *Comput. Softw. Big Sci.* **5** 13
[19] Buhmann E *et al* 2021 Fast and accurate electromagnetic and hadronic showers from generative models *EPJ Web Conf.* vol 251 p 03049
[20] de Oliveira L, Paganini M and Nachman B 2017 Learning particle physics by example: location-aware generative adversarial networks for physics synthesis *Comput. Softw. Big Sci.* **1** 4
[21] Musella P and Pandolfi F 2018 Fast and accurate simulation of particle detectors using generative adversarial networks *Comput. Softw. Big Sci.* **2** 8
[22] Carrazza S and Dreyer F A 2019 Lund jet images from generative and cycle-consistent adversarial networks *Eur. Phys. J.* C **79** 979
[23] Otten S *et al* 2021 Event generation and statistical sampling for physics with deep generative models and a density information buffer *Nat. Commun.* **12** 2985
[24] Hashemi B *et al* 2019 LHC analysis-specific datasets with generative adversarial networks (arXiv:1901.05282)
[25] Di Sipio R *et al* 2020 DijetGAN: a generative-adversarial network approach for the simulation of QCD dijet events at the LHC *J. High Energy Phys.* **08** 110
[26] Butter A, Plehn T and Winterhalder R 2019 How to GAN LHC events *SciPost Phys.* **7** 075
[27] Erdmann M *et al* 2018 Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks *Comput. Softw. Big Sci.* **2** 4
[28] Goodfellow I J *et al* 2014 Generative adversarial networks (arXiv:1406.2661)
[29] Arjovsky M, Chintala S and Bottou L 2017 Wasserstein GAN (arXiv:1701.07875)
[30] Gulrajani I *et al* 2017 Improved training of Wasserstein GANs (arXiv:1704.00028)
[31] Kingma D P and Welling M 2013 Auto-encoding variational Bayes (arXiv:1312.6114)
[32] Arjona Martńez J *et al* 2020 Particle generative adversarial networks for full-event simulation at the LHC and their application to pileup description *J. Phys.: Conf. Ser.* **1525** 012081
[33] Belavin V and Ustyuzhanin A 2020 Electromagnetic shower generation with graph neural networks *J. Phys.: Conf. Ser.* **1525** 012105
[34] Krause C and Shih D 2021 CaloFlow: fast and accurate generation of calorimeter showers with normalizing flows (arXiv:2106.05285)
[35] Lanusse F 2019 Machine learning in cosmology *ACAT 2019, Saas Fee (CH)* (available at: https://indico.cern.ch/event/708041/contributions/3308836/)
[36] Orzari B *et al* 2021 Sparse Data Generation for Particle-Based Simulation of Hadronic Jets in the LHC *38th Int. Conf. on Machine Learning Conf.*

[37] Chen C *et al* 2021 Data augmentation at the LHC through analysis-specific fast simulation with deep learning *Comput. Softw. Big Sci.* **5** 15
[38] Sjöstrand T *et al* 2015 An introduction to pythia 8.2 *Comput. Phys. Commun.* **191** 159–77
[39] Touranakou M *et al* 2022 Particle-based fast jet simulation at the LHC with variational autoencoders: generator-level and reconstruction-level jets dataset (available at: https://doi.org/10.5281/zenodo.6047873)
[40] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library (arXiv:1912.01703)
[41] Bellis M *et al* 2018 HEP software foundation community white paper working group–visualization (arXiv:1811.10309)
[42] Rezende D J and Mohamed S 2016 Variational inference with normalizing flows (arXiv:1505.05770)
[43] Higgins I *et al* 2017 β-VAE: Learning basic visual concepts with a constrained variational framework *5th Int. Conf. on Learning Representations*
[44] Fan H, Su H and Guibas L 2016 A point set generation network for 3D object reconstruction from a single image (arXiv:1612.00603)
[45] Belayneh D *et al* 2020 Calorimetry with deep learning: particle simulation and reconstruction for collider physics *Eur. Phys. J.* C **80** 688
[46] Khachatryan V *et al* 2016 Search for narrow resonances decaying to dijets in proton-proton collisions at $\sqrt{s} = 13$ TeV *Phys. Rev. Lett.* **116** 071801
[47] Aaboud M *et al* 2017 Search for new phenomena in dijet events using 37 fb^{-1} of *pp* collision data collected at $\sqrt{s}$ =13 TeV with the ATLAS detector *Phys. Rev.* D **96** 052004
[48] Sirunyan A M *et al* 2020 Search for high mass dijet resonances with a new background prediction method in proton-proton collisions at $\sqrt{s} = 13$ TeV *J. High Energy Phys.* **05** 033
[49] Sirunyan A M *et al* 2018 Search for new physics in dijet angular distributions using proton–proton collisions at $\sqrt{s} = 13$ TeV and constraints on dark matter and other models *Eur. Phys. J.* C **78** 789
[50] Kingma D K and Ba J 2015 Adam: a method for stochastic optimization (arXiv:1412.6980)
[51] Kullback S and Leibler R A 1951 On information and sufficiency *Ann. Math. Stat.* **22** 79–86
[52] Tumasyan A *et al* 2022 Study of quark and gluon jet substructure in Z+jet and dijet events from pp collisions *J. High Energy Phys.* **01** 188
[53] Aaboud M *et al* 2019 Measurement of jet-substructure observables in top quark, *W* boson and light jet production in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector *J. High Energy Phys.* **08** 033
[54] Sirunyan A M *et al* 2018 Measurement of jet substructure observables in $t\bar{t}$ events from proton-proton collisions at $\sqrt{s} = 13$TeV *Phys. Rev.* D **98** 092014
[55] Mangano M L *et al* 2016 Physics at a 100 TeV pp Collider: standard model processes (arXiv:1607.01831)
[56] Fey M and Lenssen J E 2019 Fast graph representation learning with PyTorch geometric (arXiv:1903.02428)
[57] ONNX Runtime developers ONNX Runtime 2021 (available at: www.onnxruntime.ai)