Computing and Software for Big Science (2021) 5:15
https://doi.org/10.1007/s41781-021-00060-4

ORIGINAL ARTICLE

Analysis-Specific Fast Simulation at the LHC with Deep Learning

C. Chen[1] · O. Cerri[2] · T. Q. Nguyen[2] · J. R. Vlimant[2] · M. Pierini[3]

Received: 12 October 2020 / Accepted: 12 May 2021 / Published online: 9 June 2021

Abstract
We present a fast-simulation application based on a deep neural network, designed to create large analysis-specific datasets. Taking as an example the generation of W + jet events produced in $\sqrt{s} = 13$ TeV proton–proton collisions, we train a neural network to model detector resolution effects as a transfer function acting on an analysis-specific set of relevant features, computed at generation level, i.e., in absence of detector effects. Based on this model, we propose a novel fast-simulation workflow that starts from a large amount of generator-level events to deliver large analysis-specific samples. The adoption of this approach would result in about an order-of-magnitude reduction in computing and storage requirements for the collision simulation workflow. This strategy could help the high energy physics community to face the computing challenges of the future High-Luminosity LHC.

Keywords Hadron Collider Physics · Fast Simulation · Deep Learning · High Energy Physics computing

Introduction

At the CERN Large Hadron Collider (LHC), high-energy proton–proton (*pp*) collisions are studied to consolidate our understanding of physics at the energy frontier and possibly to search for new phenomena. While these studies are typically conducted according to a data driven methodology, synthetic data from simulated *pp* collisions are a key ingredient to a robust analysis development. Particle physicists rely extensively on an accurate simulation of the physics processes under study, including a detailed description of the response of their detector to a given set of incoming particles. These large sets of synthetic data are typically generated with experiment-specific simulation software, based on the `GEANT4` [1] library. Through Monte Carlo techniques, `GEANT4` provides the state of the art in terms of simulation accuracy. The first two runs of the LHC highlighted the remarkable agreement between data and simulation, with discrepancies observed at the level of a few percent. On the other hand, running `GEANT4` is demanding in terms of resources. As a consequence of this, delivering synthetic data at the pace at which the LHC delivers real data is one of the most challenging tasks for the computing infrastructures of the LHC experiments. It is then more and more common for LHC physics analyses to be affected by large systematic uncertainties due to the limited amount of simulated data. This is particularly true for precise measurements of Standard Model processes for which large datasets are already available today. In the future, with the high-luminosity LHC upgrade, this will become a serious problem for most of the LHC data analyses [2]. Our community is called to reduce the computing resources needed for central simulation workflows by at least one order of magnitude, not to jeopardize the accuracy gain expected when operating the LHC at a high luminosity.

✉ M. Pierini
maurizio.pierini@cern.ch

C. Chen
c.chen@cern.ch

O. Cerri
olmo@caltech.edu

T. Q. Nguyen
thong@caltech.edu

J. R. Vlimant
jvlimant@caltech.edu

[1] State Key Laboratory of Nuclear Physics and Technology, School of Physics, Peking University Haidan, Beijing 100871, China

[2] California Institute of Technology, Pasadena, CA 91125, USA

[3] European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

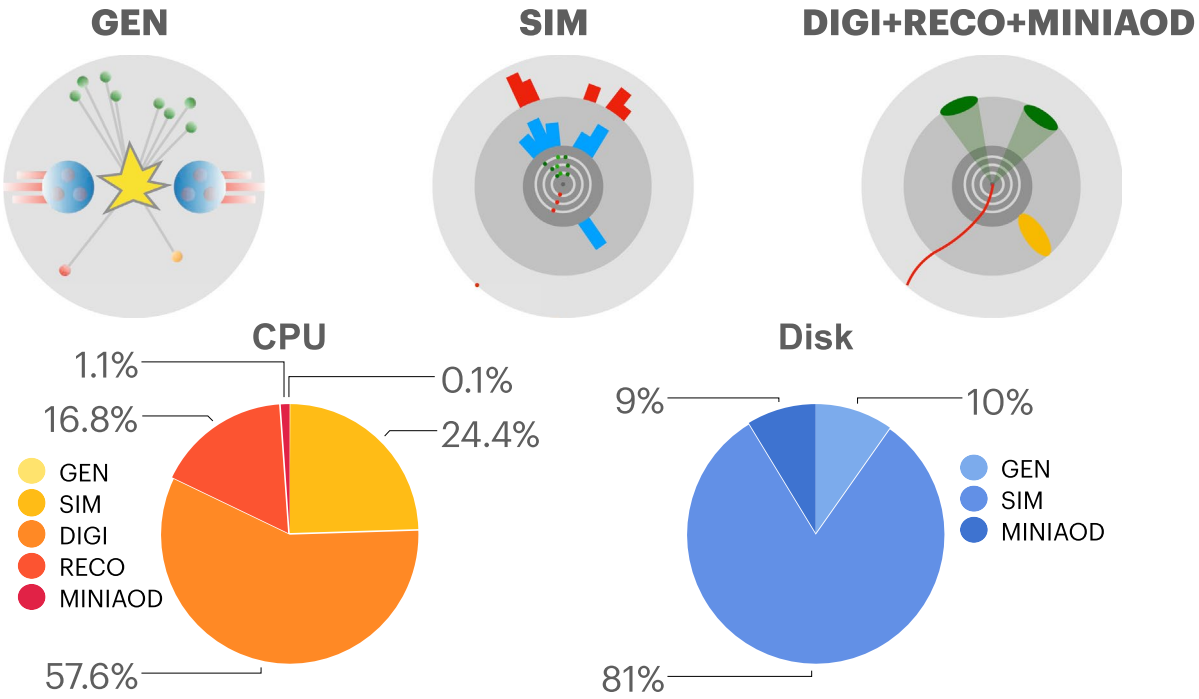

Fig. 1 TOP: The event generation workflow of the CMS experiment. The *pp* collision process is simulated up to the production of stable (hence observable) particles (GEN). The simulation of the detector response is modelled by the GEANT4 library (SIM). The resulting energy deposits are turned into digital signals (DIGI) that are then reconstructed by the same software used to process real collision events (RECO). At this stage, high-level objects such as jets are reconstructed. Starting from the RECO data format, a reduced analysis data format (MINIAOD) is derived. BOTTOM: computing resource breakdown for the generation workflow of the CMS experiment, in terms of CPU (left) and storage disk (right). See Appendix 8 for details

To give a concrete example, we consider the event simulation workflow of the CMS experiment, schematically represented in Fig. 1. The first step (GEN) consists in running an event generator library, simulating a *pp* collision, the production of high-mass particles from it, and the decay of these particles to those stable particles which are then seen by the detector. This step creates the so-called generator-level view of a collision event, corresponding to what a perfect detector would see. The simulation of the detector response (SIM) translates this flow of particles into a set of detector hits, taking into account detector imperfections and the limited experimental resolution. These hits are converted to the same digital format (DIGI) produced by the detector electronics and then reconstructed by the same software used to process real collision events (RECO). At this stage, high-level objects such as jets are created. Starting from the RECO data format, a reduced analysis data format (MINIAOD) is derived [3]. Figure 1 also provides a breakdown of CPU and disk resources for each of these steps. Details on the procedure followed to measure these values are given in Appendix 8.

Recently, generative algorithms based on Deep Learning (DL) techniques have been proposed as a possible solution to speed up GEANT4. When following this approach, one typically focuses on an image representation of LHC collisions (e.g., energy deposits in a calorimeter) and develops some kind of generative model [4–8] to by-pass GEANT4 when simulating the detector response to individual particles [9–13] or to groups of particles, such as jets [14–16] or cosmic rays [17]. Generative models were considered also for similar applications in HEP, such as amplitude [18] and full event topology [19–21] generation. While these studies demonstrate the potential of generative models for HEP, more work is needed to fully integrate this new methodology in the centralized computing system of a typical LHC experiment. In particular, one needs to work beyond the collision-as-image paradigm so that the DL-based simulation accounts for the irregular geometry of a typical detector while delivering a dataset in a format compatible with downstream reconstruction software.

Other studies [22–24] investigated a more extreme approach: rather than training models to perform generic generation tasks in a broader software framework (e.g., a DL-based shower generator in GEANT), one could design analysis-specific generators, with the limited scope of delivering arrays of values for physics quantities which are relevant to a specific analysis. Reducing the event representation to a vector of meaningful quantities, one could

obtain a large amount of events in short time and with small storage requirements by skipping all the intermediate steps of the data processing. The considered features could be the fundamental quantities used by a given analysis (e.g., the four-momenta of the final-state reconstructed objects in a search for new particles). In this context, both generative adversarial networks (GANs) [22, 23] and variational autoencoders (VAEs) [22] were considered. In this case, one learns the N-dimensional probability density function (N-dim pdf) of the event, in a space defined by the quantities of interest for a given analysis. Sampling from this function, one can then generate new data. The open question with this approach stands with the trade-off between statistical precision (which decreases with the increase amount of generated events) and the systematic uncertainty that could be induced by a non accurate description of the N-dim pdf. When training both VAEs and GANs, one learns how to interpolate between the samples provided in the training dataset. The limited amount of data in the training dataset is the ultimate precision-limiting factor, as discussed in Ref. [25], but generative models retain amplification capability similarly to what a fitting function does, as shown in Ref. [26] for GANs. Ultimately, one needs to balance the statistical uncertainty (i.e., the amplification factor when augmenting the dataset) and systematic uncertainties associated to the accuracy with which the generative model interpolates between the training data points. The balance will be reached tuning, among other things, the training dataset size. The optimal configuration, intrinsically application specific, determines whether a generative model is computationally convenient.[1]

In this paper, we propose to rephrase the problem of analysis specific dataset generation. Rather than morphing a distribution in a latent space into a target distribution, we want to start from the ideal-detector distribution and morph it into the actual-detector distribution, learning a fast-and-accurate detector response model. We do so combining the strength of multi dimensional deep neural regressors to the adaptive power of kernel density estimation, which has a long and successful tradition in particle physics [27]. A similar goal is presented in Ref. [24] in which invertible neural networks are utilized with a focus on being able to perform unfolding (morphing from reconstructed level information to generator level distributions). For a given physics study, we assume that the interesting features can be represented by a limited set of high-level quantities (the feature vector $\mathbf{x}$). We assume that a training dataset is provided. For each collision event in the dataset, the feature vector is computed at three stages: (i) at generator level $\mathbf{x}_{\mathrm{G}}$, i.e., before applying any detector simulation. This view of the collision event corresponds to the perfect-resolution ideal detector case; (ii) at reconstruction level $\mathbf{x}_{\mathrm{R}}$, i.e. after the simulation of the detector response, modelled with GEANT4; (iii) at the output of the DL model $\mathbf{x}_{\mathrm{DL}}$.[2] We model the detector response as a function of the generator-level feature vector:

$$x^i_{\mathrm{DL}} = \mathcal{N}(\mu^i_{\mathrm{R}}(\mathbf{x}_{\mathrm{G}}), \sigma^i_{\mathrm{R}}(\mathbf{x}_{\mathrm{G}})), \tag{1}$$

where $\mathcal{N}(\mu, \sigma)$ is a one-dimension Normal function centered at μ with variance σ^2 and the index i runs over the components of the feature vector $\mathbf{x}$. We train a DL model to simultaneously learn the functions $\boldsymbol{\mu}_{\mathrm{R}}(\mathbf{x}_{\mathrm{G}})$ and $\boldsymbol{\sigma}_{\mathrm{R}}(\mathbf{x}_{\mathrm{G}})$ and then use the Normal model of Eq. (1) to generate $\mathbf{x}_{\mathrm{DL}}$ from $\mathbf{x}_{\mathrm{G}}$. Under the assumption that large sets of $\mathbf{x}_{\mathrm{G}}$ values can be obtained in relatively short time (which is typically the case for High Energy Physics applications), this strategy would result in a sizable save of computing resources. On one hand, one would reduce computing time bypassing the more intense steps of the generation workflow. In addition, one would reduce the need for large storage elements: rather than storing individual collision data, which demands an event storage allocation between $\mathcal{O}$(1MB) (for raw data) and $\mathcal{O}$(10kB) (for analysis-ready object collections), one would directly handle a few relevant quantities for a given analysis. One could save resources by utilizing analysis-specific fast simulation models for data augmentation, e.g., generating 10% of the required data with the traditional GEANT4 workflow and the remaining 90% only up to the GEN step. These data, shared among the $\mathcal{O}$(100) analyses, would be used to create analysis-specific training and inference datasets. Even considering that $\mathcal{O}$(100) analysis teams would have to train $\mathcal{O}$(100) specific generative models, the strategy we propose would result in an important resource gain, provided a large enough training facility.[3]

We demonstrate this strategy at work on a concrete example, namely the generation of $W + 1$ jet events produced in $\sqrt{s} = 13$ TeV pp collisions, similar to those recorded at the LHC. We discuss the model design and training, its performance and its accuracy for factor-ten data augmentation.

This paper is structured as follows: "Benchmark Dataset" provides a full description of the input dataset and its feature-vector representation. "Model Description and

[1] Here, we are assuming that GEANT4 will be used to generate the training dataset and the generative model will then be used to scale up the simulated dataset size. If the desired accuracy can be reached only at the price of more training data to be generated, the net gain of this approach would be reduced.

[2] Given the limited computing resources at hand, it was not possible to carry on this study on a GEANT4-based dataset. Instead, we used the DELPHES [28], which provides a realistic setup to demonstrate the proposed strategy.

[3] The model presented in this work was trained on a RTX2080 GPU by NVIDIA in 30 minutes. Even a small-size GPU cluster with O(10) GPUs dedicated to this use case could then serve the needs of a large collaboration. Its cost is negligible on the scale of the large computing infrastructures built for the LHC experiments.

Training" describes the model architecture and the training setup. "Results" and "Computing Resources" discuss the model performance in terms of accuracy and resource utilization, respectively. Conclusions and outlook are given in "Conclusions".

Benchmark Dataset

As a benchmark problem, we consider the generation of $W + 1$ jet events produced in $\sqrt{s} = 13$ TeV pp collisions. The starting point is the inclusive production of $W \to \mu\nu$ events using PYTHIA8 [29]. At this stage, we require each event to have at least one muon with a transverse momentum $p_T > 22$ GeV.[4] Detector effects are modelled using DELPHES v3.4.2 [28]. We consider the CMS detector model for the HL-LHC upgrade, distributed with DELPHES. At this stage, the event is overlaid to minimum-bias events to model the effect of pileup, i.e., those parasitic pp collisions happening at the same beam crossing as the interesting event. For each collision, the number of pileup collisions is sampled from a Poisson distribution with expectation value set at 200, to match the expected conditions for HL-LHC.

At generator level (GEN), jets are clustered using the anti-kt algorithm [30] with jet-size parameter $R = 0.5$, taking the four-momenta of all the stable particles in the event as input. We consider events with one clustered jet, with $p_T > 30$ GeV and $|\eta| < 2.4$. To avoid the double counting of muons as jets, we require $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} > 0.5$ between the muon and the jet in each event.

At reconstruction level, jets are clustered from the list of particles returned by the DELPHES particle-flow algorithm. As for the GEN jets, we consider anti-kt jets with $R = 0.5$. Both the muon and jet are matched to the corresponding generator-level object, selecting the reconstructed object (e.g., a muon) with the smallest ΔR from the corresponding generator-level object. Since our final state is composed of one jet and one muon, this simple algorithm does not generate ambiguity in the association. When generalizing this approach to more complex event topologies, one might modify the matching algorithm to prevent that the same gen-level object is associated to multiple reconstructed objects. In addition, we discard events with mismatched muons by requiring that the relative residual of the muon p_T to be $|p_T^G - p_T^R|/p_T^G < 10\%$. This requirement allows us to remove a small fraction of events ($\sim 0.5\%$ of the total) in which the muon from the W boson is not reconstructed but another muon is found. In DELPHES, inefficiency in muon reconstruction happens through an uncorrelated hit-or-miss procedure based on pseudo-random numbers. Working in an experimental environment, one would retain the whole dataset from a more accurate simulation, based on specific physic requirements that would induce learnable correlations.

The feature vector $\mathbf{x}$ is built considering the following nine quantities:

- The muon momentum in Cartesian coordinates: p_x^μ, p_y^μ, and p_z^μ.
- The jet momentum in Cartesian coordinates: p_x^j, p_y^j, and p_z^j.
- The logarithm of the jet mass $\log(M_j)$.
- The missing transverse energy in Cartesian coordinates: E_x^{miss} and E_y^{miss}.

In addition, we consider a set of 12 auxiliary features, computed from the input feature vector $\mathbf{x}$:

- The muon momentum in longitudinal-boost-invariant coordinates: p_T^μ, η^μ, and ϕ^μ.
- The jet momentum in longitudinal-boost-invariant coordinates: p_T^j, η^j, and ϕ^j.
- The missing transverse energy in polar coordinates: E_T^{miss} and ϕ_{miss}.
- The transverse mass M_T, i.e., the mass of the four momentum obtained summing the the muon transverse momentum $(E_T^\mu, p_x^\mu, p_y^\mu, 0)$ to the missing transverse energy $(E_T^{\text{miss}}, E_x^{\text{miss}}, E_y^{\text{miss}}, 0)$.
- S_T, i.e., the scalar sum of E_T^{miss}, p_T^μ, and p_T^j.
- The jet mass: M_j.

These quantities are computed at generator and reconstruction level and are used to assess how well the correlation between the generated quantities is modeled. Unlike the feature-vector quantities, they do not enter the definition of the loss function.

The model training and performance assessment is done on a dataset of 2M events, which we separate in a test and a learning datasets, containing 20% and 80% of the events, respectively. The learning dataset is further split into a training (70%) and a validation (30%) dataset. To test the data augmentation properties of the proposed strategy, we also consider a larger test dataset, containing 10 M events.

Both the training and large-size testing datasets are published on Zenodo [31, 32].

[4] 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 $c = \hbar = 1$.

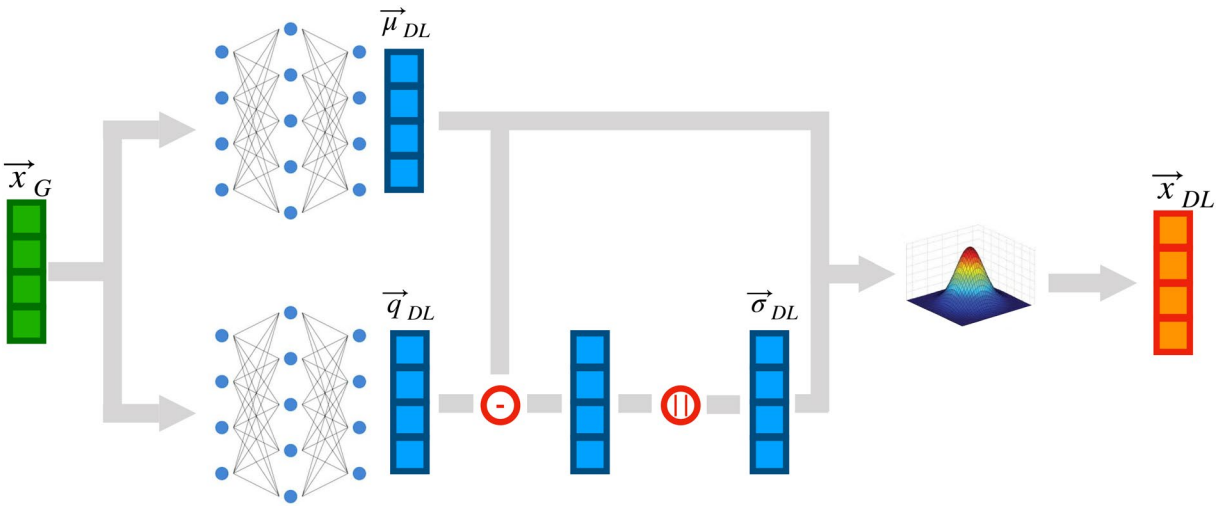

Fig. 2 Model architecture: a feature vector at generator level $\mathbf{x}_G$ is given as input to two regression models, returning vectors of central values ($\boldsymbol{\mu}_{DL}$) and RMS ($\boldsymbol{\sigma}_{DL}$), from which the reconstructed feature vector predicted by the DL model $\mathbf{x}_{DL}$ is generated

Model Description and Training

Our model architecture is represented in Fig. 2. The input vector $\mathbf{x}_G$ of generator-level features is passed to two regressive models, each returning a vector with the same dimensionality of $\mathbf{x}_G$. One is interpreted as a vector of mean values $\boldsymbol{\mu}_{DL}$. The other one is interpreted as the "$\pm 1\sigma$" quantile $\mathbf{q}_{DL}$. By taking the absolute difference between each mean value and its corresponding quantile, we compute the RMS values $\boldsymbol{\sigma}_{DL}$.

Each regressive model consists of a six-layer dense neural network. The first and last layers have nine nodes each, while the intermediate layers have 100 nodes. All layers except the last one are activated by LeakyReLU [33] functions, with $\alpha = 0.05$. Linear activation functions are used for the last layer. The model output is then computed as $\mathbf{x}_{DL} = \boldsymbol{\mu}_{DL} + \boldsymbol{\sigma}_{DL} \cdot \boldsymbol{\epsilon}$, where the vector $\boldsymbol{\epsilon}$ contains random numbers sampled from a Normal function centered at 0 with unit variance. In addition to the main features $\mathbf{x}$, we compute a set of auxiliary features (see "Benchmark Dataset") used for a further post-training validation.

The loss function is defined as the sum of a mean absolute error on $\boldsymbol{\mu}_{DL}$ and a quantile regression on $\mathbf{q}_{DL}$:

$$\mathcal{L}_{RECO} = \left\langle \left\| \boldsymbol{\mu}_{DL} - \mathbf{x}_R \right\|_1 + QR(\mathbf{q}_{DL}, \mathbf{x}_R) \right\rangle, \tag{2}$$

where the average is done over a training subset, and the quantile regression loss QR is defined as:

$$QR(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{k} \Theta(x_i, y_i)|x_i - y_i|, \tag{3}$$

where

$$\Theta(x, y) = (1 - \gamma)\theta(x - y) + \gamma\theta(y - x). \tag{4}$$

The step function $\theta(t)$ is set to one (zero) for positive (negative) values of t and $\gamma = 0.841$. This choice of γ guarantees that the loss is minimized to learn the quantile corresponding to one standard deviation.

We implement the model in `KERAS` [34] and train it with the Adam [35] optimizer, with batches of 128 and an epoch-dependent learning rate $\mathrm{lr} = 0.001/(1 + n_{\mathrm{epoch}})$. The model is trained for 100 epochs, but convergence is typically reached between 30 epochs. The network parameter values corresponding to the smallest validation loss are taken as the optimal configuration.

Results

The trained model is used to generate samples of reconstructed events from generator-level events. We evaluate the training performance by comparing the output distributions with those obtained by `DELPHES` for the same generator-level events.

A comparison is shown in Fig. 3 for the feature-vector quantities. The sample derived from the DL model is similar to the the one obtained running a classic generation workflow. We train the model ten times and produce ten distributions. The bin-by-bin spread of these distributions is considered as a systematic uncertainty associated to the DL model, which is summed in quadrature to the statistical uncertainty in the same bin to compute the total uncertainty,

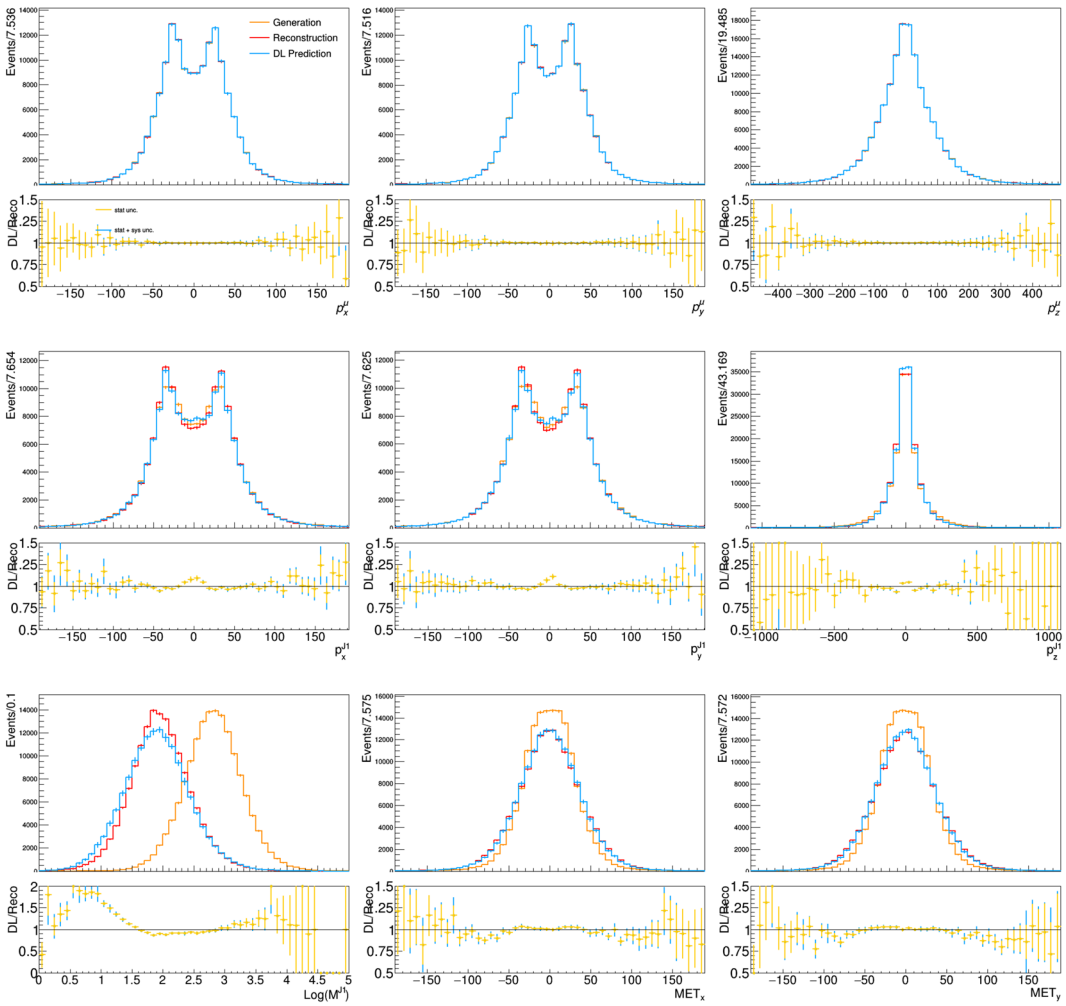

Fig. 3 Distribution of reconstructed and model-predicted quantities for the feature-vector quantities, compared to the corresponding quantities from generator-level quantities provided as input to the model. The bottom panel below each plot shows the bin-by-bin ratio of the model-predicted over reconstructed distribution for each quantity, labelled DL/Reco. The error bars on the model-predicted quantities is composed of the statistical uncertainty and systematic uncertainty associated with model training, represented by the different colors

shown by the error bars of the DL model in the figure. These systematic uncertainties are included to all the DL distributions shown in this paper. Only the statistical uncertainty is shown for the corresponding distributions of reconstructed quantities.

The model can account for small perturbations and major distortions of the GEN distribution, as well as the default detector simulation workflow. The agreement is not perfect, and certainly the model can be improved. Nevertheless, the reached accuracy is comparable to that of a typical data-to-simulation comparison and certainly sufficient to support the novel procedure that we want to put forward in this study. The observed agreement goes beyond one-dimensional projections of the input features. The distributions of auxiliary quantities, computed as a function of the feature-vector quantities, are also modelled to a good precision (see Fig. 4). This demonstrates that the DL-based generator accounts for correlations between quantities, as much as the traditional `DELPHES` workflow does.

While a comparison of dataset distribution gives a confidence of the quality achieved by the DL model, one can further test the achieved precision by looking at relative residual distributions. Our DL model does not sample events from a latent space (like a GAN or a plain VAE). Instead, it works as a fast simulation of a given generator-level event, preserving the correspondence between the reconstructed and the generated event, which allows us to compare event-by-event relative residual distributions. These distributions, which quantify the detector effects on the analysis-specific interesting quantities, are shown in Fig. 5. There, we compare the relative residuals between reconstructed and generated quantities, for the DL-based and the traditional simulation workflow. An overall agreement is observed, despite a bias on the muon and jet momentum coordinates. While the

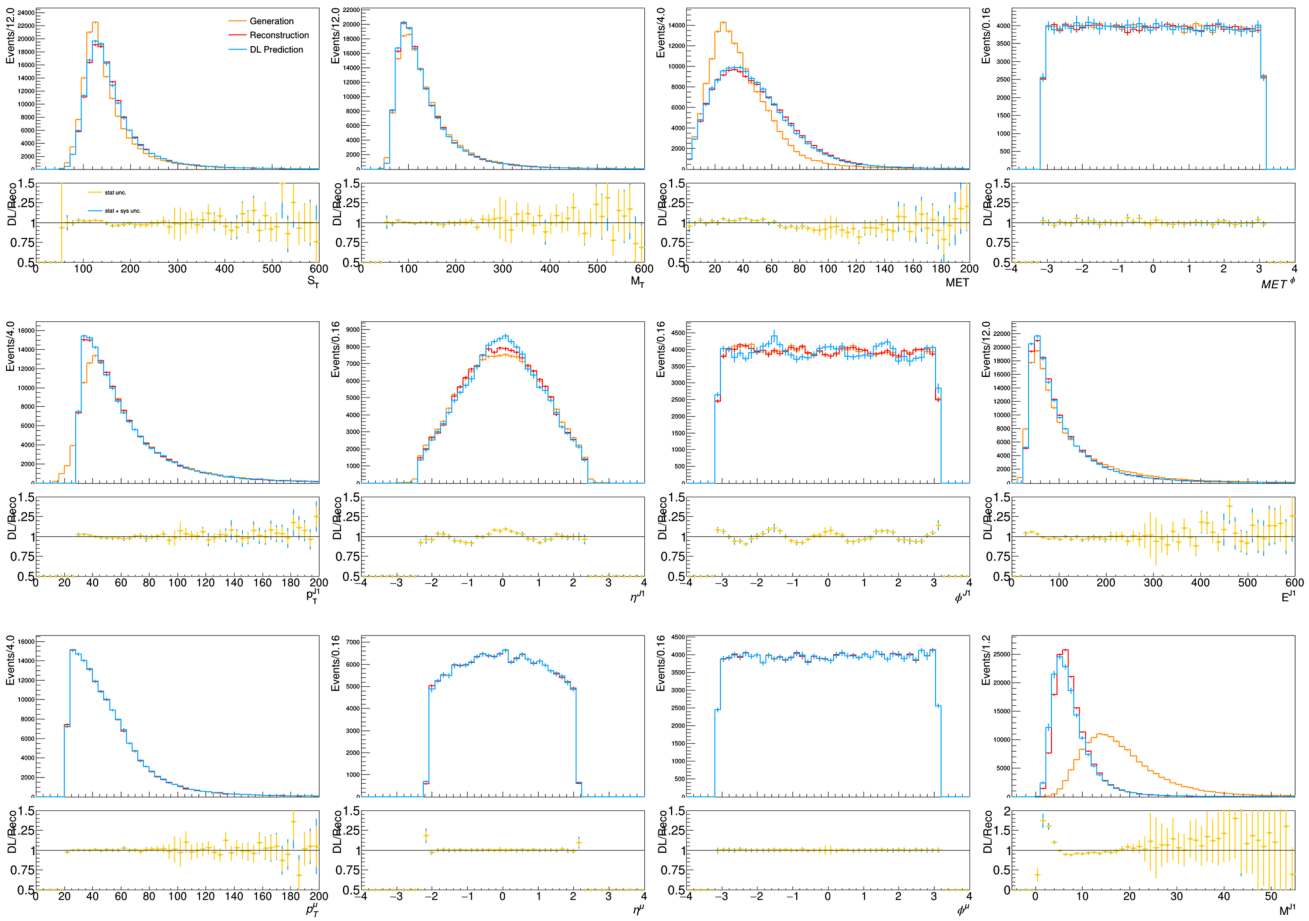

Fig. 4 Distribution of reconstructed and model-predicted auxiliary quantities, compared to the corresponding generator-level quantities. The bottom panel below each plot shows the bin-by-bin ratio of the model-predicted over reconstructed distribution for each quantity, labelled DL/Reco. The error bars on the model-predicted quantities is composed of the statistical uncertainty and systematic uncertainty associated with model training, represented by the different colors

distribution ratio shown in the bottom panel tends to magnify the effect that the distribution shift has on the tails, this residual difference between the target and learned resolution model has little impact on the simulation quality downstream, as one could judge by looking at the corresponding distributions in Fig. 3.

Figure 6 shows the same comparison for the auxiliary quantities. As the plot shows, a correct modeling of the residuals is obtained for energies, masses, and momenta. On the other hand, the model struggles to account for the high-resolution detector response on the η and ϕ coordinates. While this has little impact on the modeling of the ϕ and η distributions (see Fig. 3), this is certainly an aspect to improve in real-life applications. Deeper models on larger training data could learn the function better. In addition, one could modify the loss function to force the network to learn specific auxiliary quantities (e.g., the jet mass) with critic networks (as done in the context of GAN training) and explore non-Gaussian response functions. To this extent, working in Cartesian coordinates might be a better choice, to facilitate the calculation of the auxiliary quantities in the loss function. We did not expand our study in these directions, for which a target dataset based on a full detector simulation would be more appropriate.

Appendix 9 provides further assessments of the generation quality, showing 2D distributions of quantities derived from the DL-based generator vs the traditional one.

While our method relies on a Gaussian smearing function, it could be generalized to more complex functions if needed. In that case, one would have to learn more quantiles to model response functions with more than two parameters and then express these parameters as a function of the learned quantiles. On the other hand, it should be stressed that the response functions learned by our method are the result a convolution of the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ distribution (approximated by the Neural Network) and the Gaussian sampling function. Since the former is typically described by a non-Gaussian distribution, our model can learn non-Gaussian

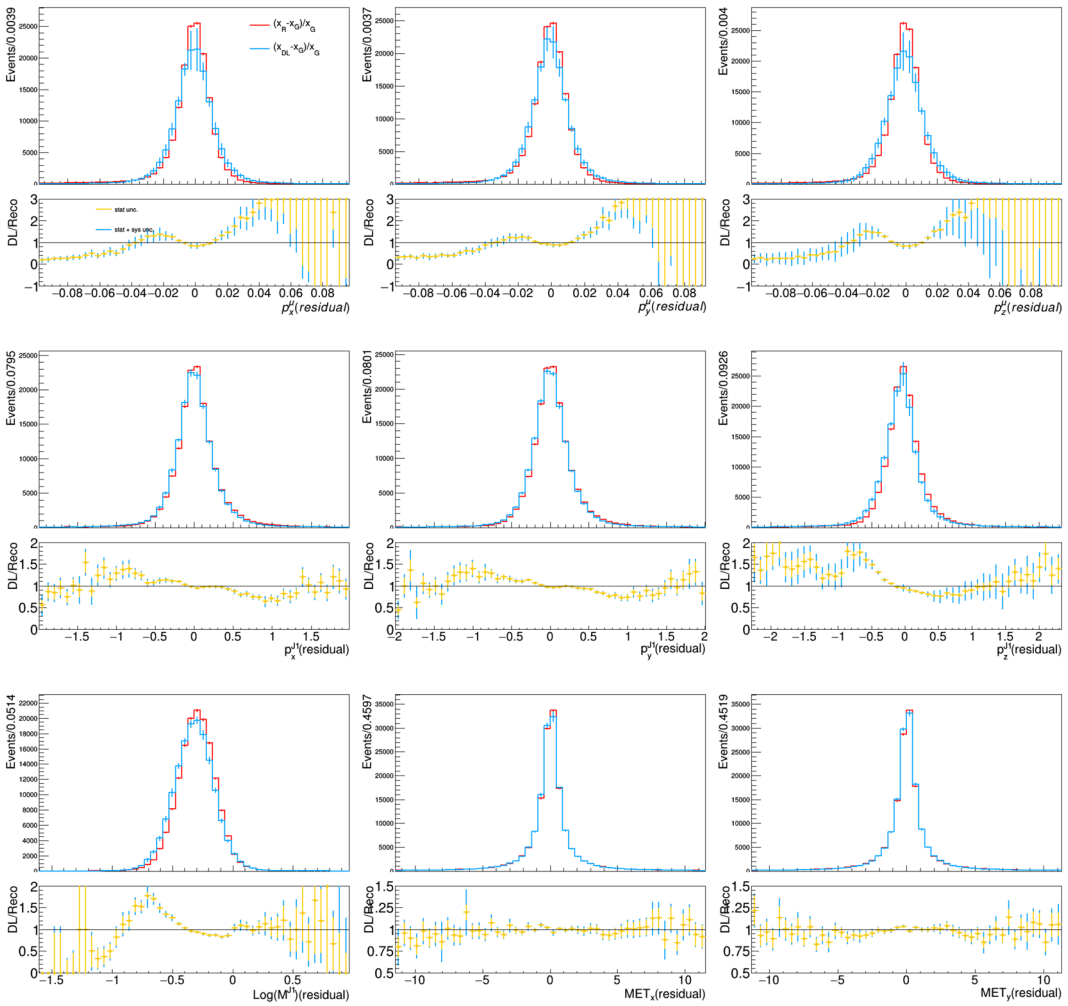

Fig. 5 Relative residual distribution for reconstructed and model-predicted quantities in the feature vector, computing with respect to the reference input. The bottom panel of each plot shows the ratio between the two relative residuals, expected to be consistent with 1 for a DL model which correctly models the detector response of the traditional workflow. The error bars on the model-predicted quantities is composed of the statistical uncertainty and systematic uncertainty associated with model training, represented by the different colors

detector response even when relying on a simple Gaussian sampling. This is the case, for instance, of the asymmetric tail of the MET residual distribution or the ϕ_{miss} double-peak structure shown in Fig. 6. On a practical side, a Gaussian sampling was adequate for this study, based on DELPHES data, but one might have to consider more complex sampling functions when trying to emulate with GEANT-based simulation.

To test the scaling of model accuracy with the inference dataset size, we apply our DL-based fast simulation strategy to a dataset five times bigger than what used for training. Figures 7 and 8 show the comparison of the distributions obtained in this case, compared to what is obtained with DELPHES, respectively for the input vector and the auxiliary features. The corresponding relative residual distributions are shown in Appendix 10. Figure 9 shows the differential double ratio distribution (high-statistics over low-statistics) for the reco-to-DL ratios. In presence of a systematic effect masked at low statistics, the reduction of the uncertainty in the high-statistics sample would unveil the problem. Instead we do observe flat double ratios, i.e. a similar behavior of the DL model for the small and the large sample. In view of this empirical observation, we are confident that the DL model accuracy would scale at much larger dataset size than what is used for training.

These distributions agree with those obtained when the training and inference dataset size agree, i.e., no accuracy deterioration is observed due to the scaling of the dataset size. This fact suggests that the proposed methodology scales adequately with the inference dataset size.

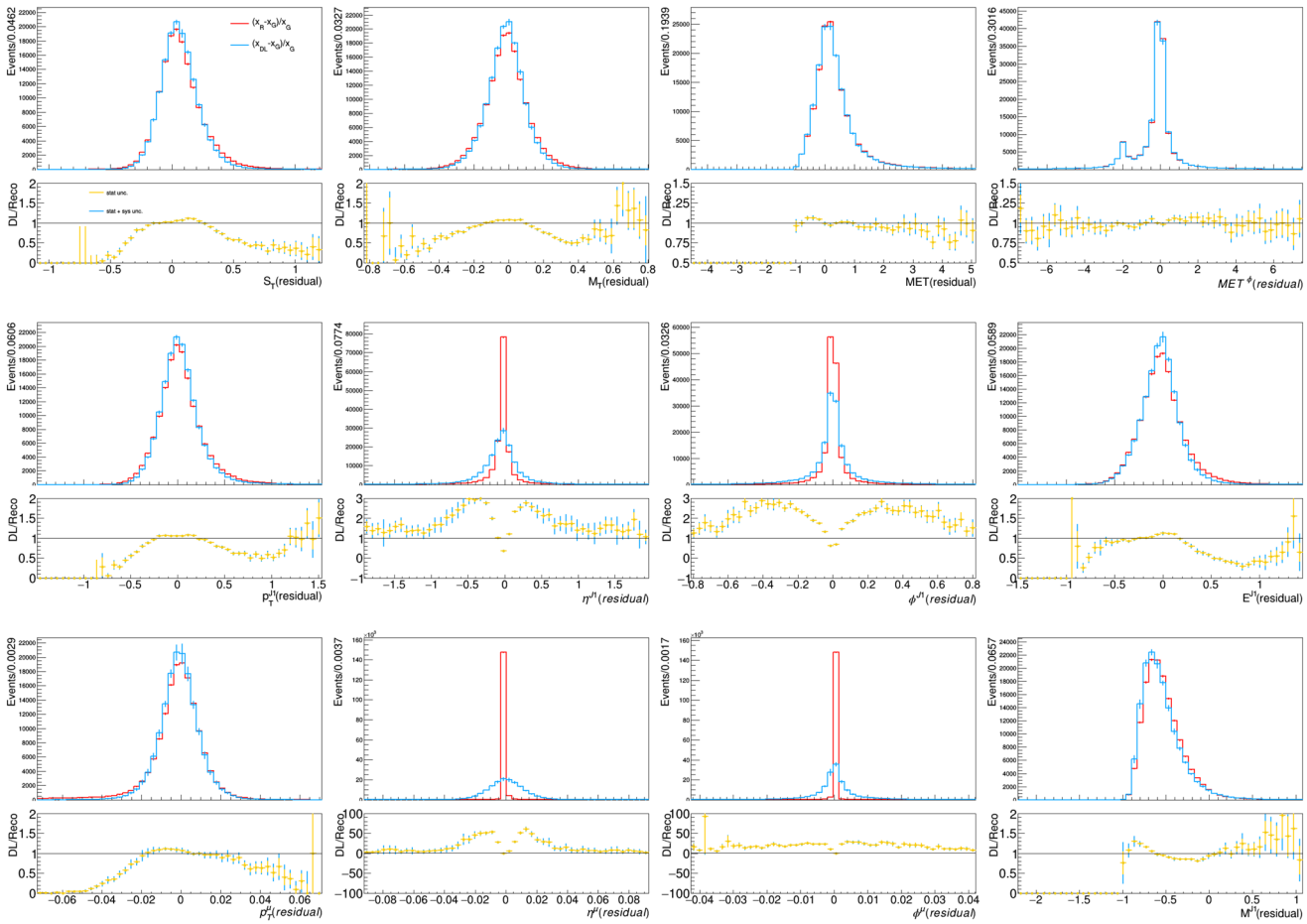

Fig. 6 Relative residual distribution for reconstructed and model-predicted auxiliary quantities, computing with respect to the reference input. The bottom panel of each plot shows the ratio between the two relative residuals, expected to be consistent with 1 for a DL model which correctly models the detector response of the traditional workflow. The error bars on the model-predicted quantities is composed of the statistical uncertainty and systematic uncertainty associated with model training, represented by the different colors

Computing Resources

To fully assess the advantage of the proposed generation workflow, we consider the following use case: an analysis team requests N events to be centrally produced by the central computing infrastructure of their experimental collaboration. Instead, the central system would deliver N events at generator level (GEN step of Fig. 1), while processing only $n < N$ of them through the full chain. The analysis team would then (i) run their data analysis software on the n events, and (ii) train on these data a DL-based fast-simulation like the one presented in "Model Description and Training". With this model, they would then (iii) process the other $(N - n)$ generator-level events and produce the dataset required for their analysis.

To assess the resource savings, we point out that step (iii) comes with negligible computational costs. Model inference on a CPU requires 100 s to run on 100,000 events (i.e., $\mathcal{O}(1)$ ms/event), which results in a 8 MB file (saved as a compressed HDF5 file) for the example use case we discussed. While these details would change depending on the analysis-specific event representation, the quoted values give a reasonable order-of-magnitude estimate of the expected resource needs. Step (ii) can run at a minimal cost: our model could train within 30 minutes when running on a commercial GPU. The residual cost is then entirely driven by step (i). While a traditional workflow requires O(100) s/event of CPU time and occupies O(1) MB/event of storage, producing the same statistics (N events) of GEN-only events would require 10% disk allocation with a negligible CPU cost, as shown in Fig. 1.

As a consequence, by adopting the strategy outlined above, one would save a factor N/n in CPU (i.e., only spend sizable CPU resources to produce the training dataset, which would remain generic and could serve more than one analysis). The storage allocation would result from the sum of n events in full format and $(N - n)$ GEN-only events, for a total saving of $N/(n + 10\%(N - n))$. For instance, considering

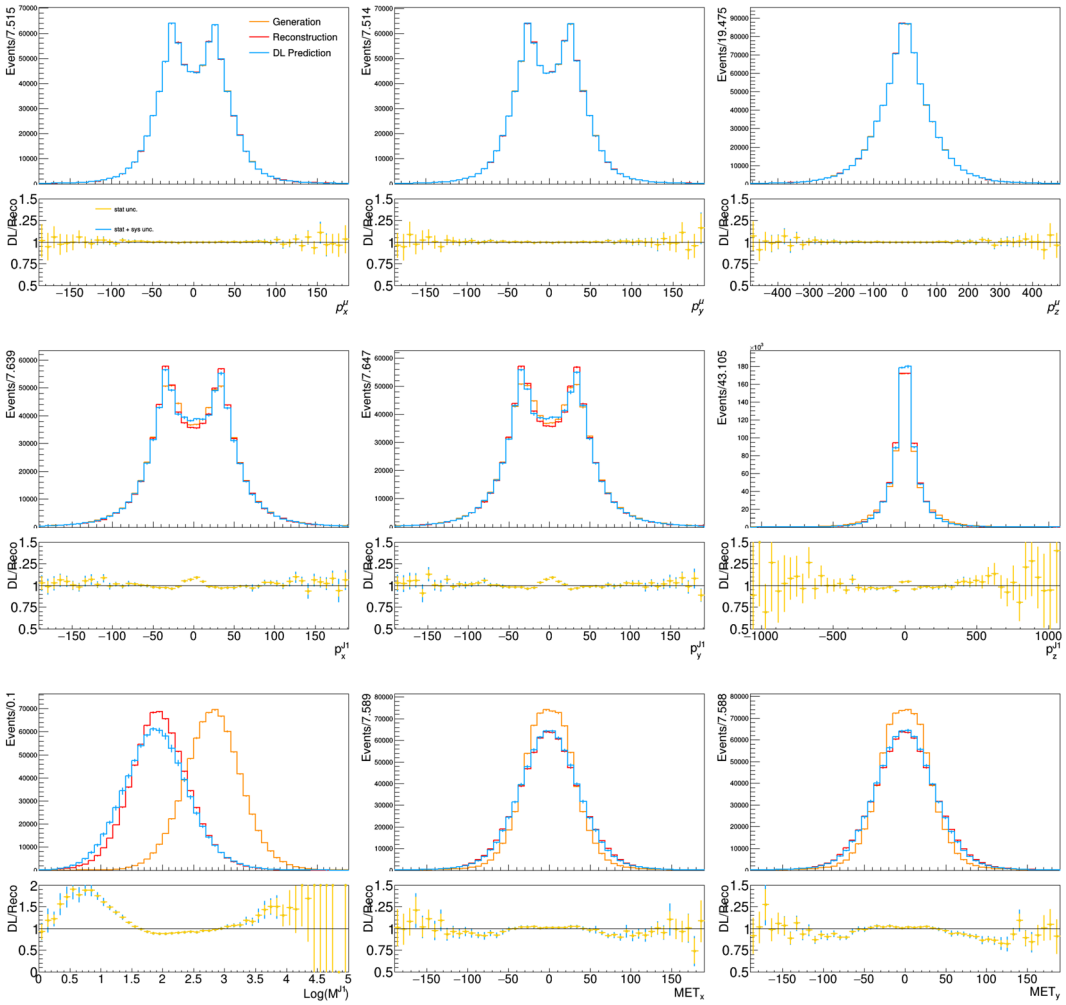

Fig. 7 Distribution of reconstructed and model-predicted quantities for the feature-vector quantities, compared to the corresponding quantities from generator-level input. In this case, the model is applied to a dataset five times larger than the training dataset. The error bars on the model-predicted quantities is composed of the statistical uncertainty and systematic uncertainty associated with model training, represented by the different colors

$N = 1M$ events and $n = 10\%N$, one would save 90% of the CPU resources and 79% of the disk storage, almost equally shared among the full-format training data and the $(N - n)$ GEN-only data.

In principle, the adoption of Next-to-Leading order precision as a default for event generators could make the cost of the GEN step more relevant in the future. On the other hand, the upgrade of the detectors towards more granularity will also substantially increase the SIM. We then expect that the SIM step would still be the dominant consumer of CPU time, unless acceleration strategies like those proposed here will introduce beyond-GEANT alternatives. In addition, we do expect progresses to speed up the GEN step as well, e.,g., moving the computation to GPUs or similar accelerators [36], or using deep learning in phase-space integration [37–40].

Conclusions

We presented a proposal for a new data augmentation strategy for fast simulation workflows at LHC experiments, which exploits a generative Deep Learning model to convert an analysis-specific representation of collision events at generator level to the corresponding representation at reconstruction level. Following this procedure, one could

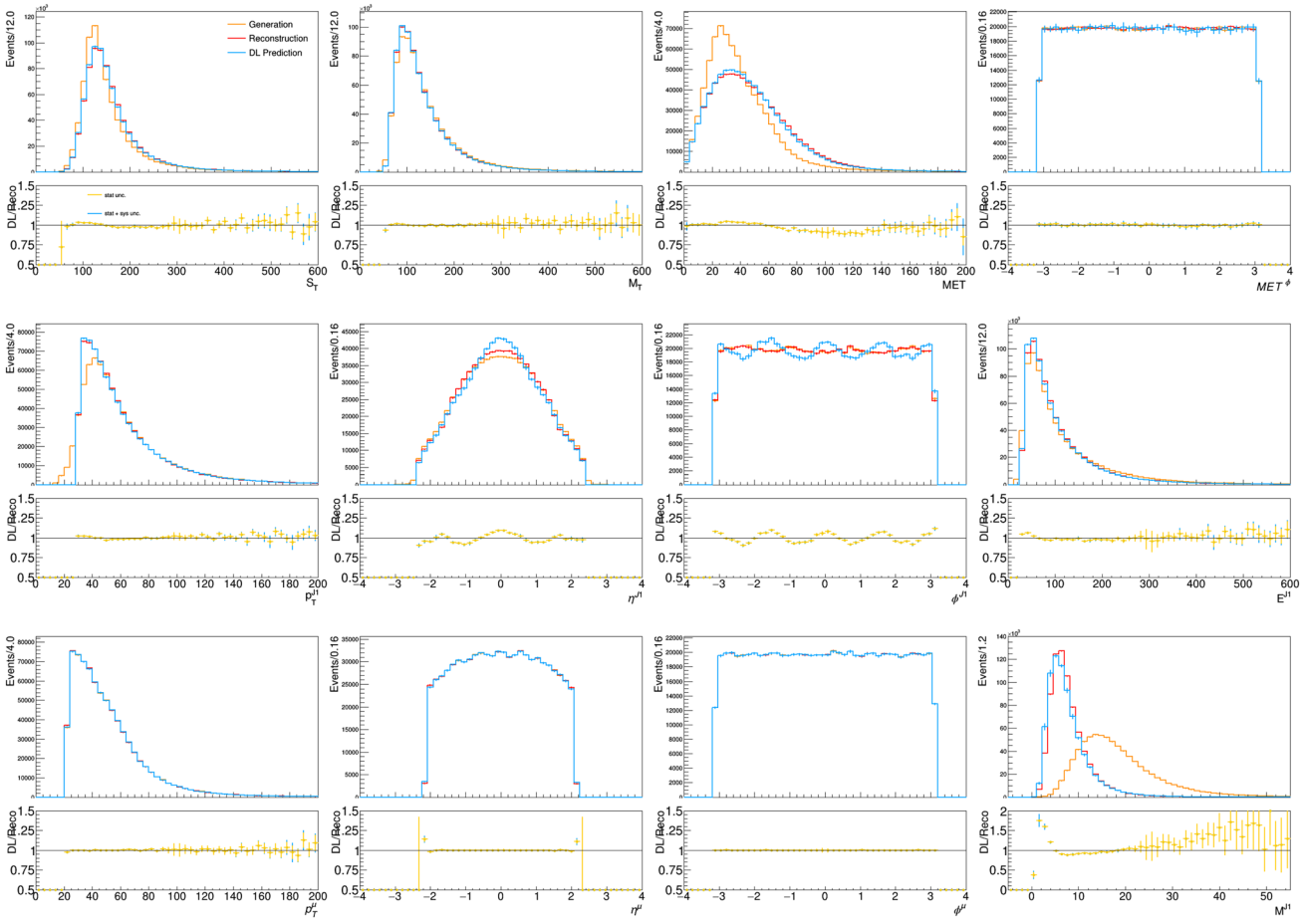

Fig. 8 Distribution of reconstructed and model-predicted quantities in the auxiliary quantities, compared to the corresponding quantities from generator-level input. In this case, the model is applied to a dataset five times larger than the training dataset. The error bars on the model-predicted quantities is composed of the statistical uncertainty and systematic uncertainty associated with model training, represented by the different colors

replace any request of N simulated events with an $n < N$ request, providing the residual $(N - n)$ events at generator level. Bypassing the detector simulation and reconstruction process for the $(N - n)$ events, one would benefit of a substantial reduction in terms of required resources.

We demonstrated that a simple mean-and-variance regression model with a Gaussian sampling function allows to reach a good performance, producing a dataset which resembles that from a traditional workflow. We showed that the accuracy is preserved when applying our strategy to a test dataset much larger than the training dataset.

The proposed model is much simpler than a generative model, e.g., a GAN. The architecture is easier to train and the task it learns to solve is simpler than generative realistic events from random points in a latent space. The generator-level input carries much of the domain knowledge and the statistical fluctuations of the target dataset size. In addition, thanks to the light computational weight of the training and inference steps, one could consider to train several models and apply them to the same test dataset, using the spread of predictions to evaluate a simulation systematic uncertainty.

We believe that the LHC experiments could benefit from adopting the proposed procedure, particularly for the high-precision measurement era during the High-Luminosity LHC phase.

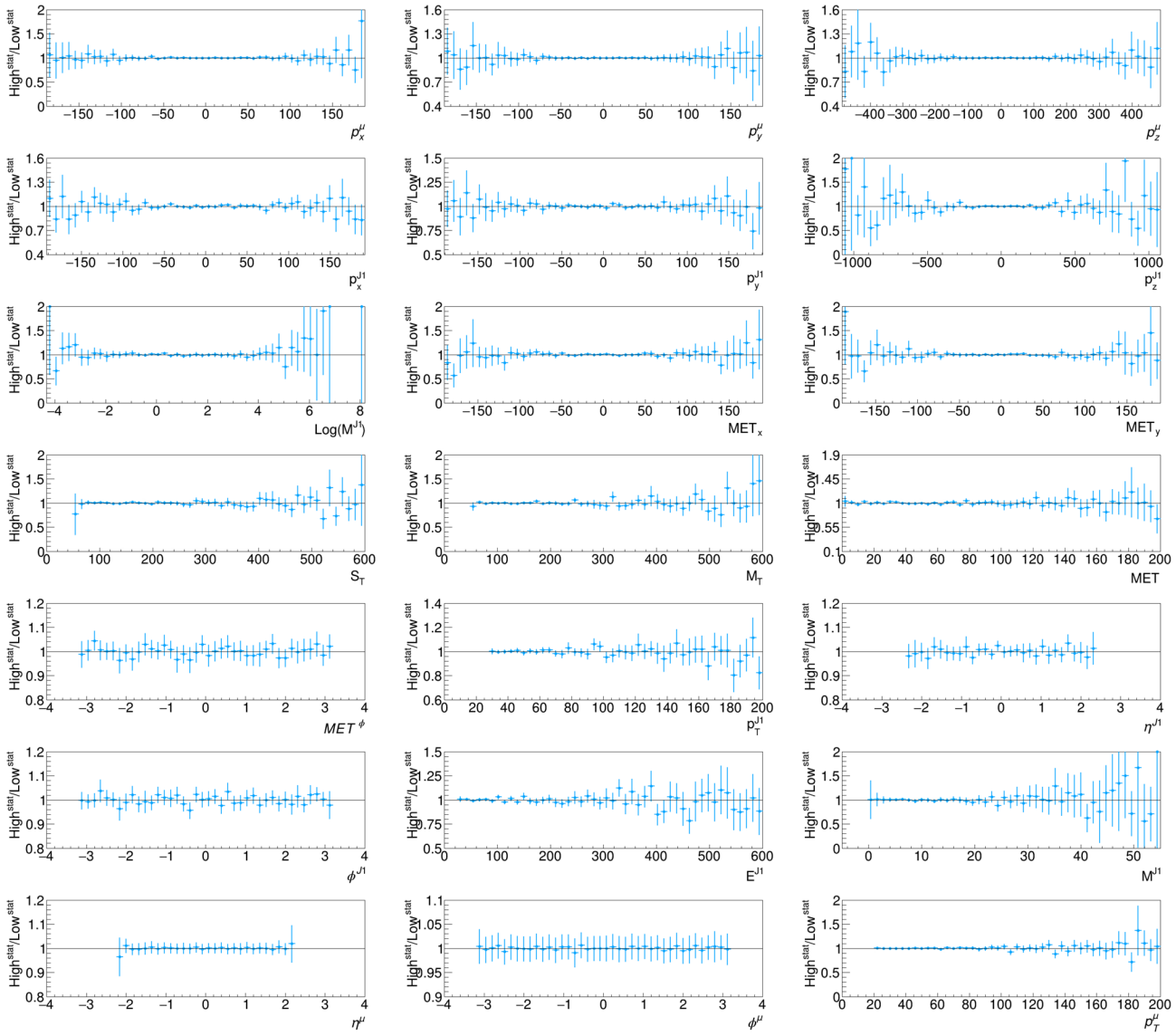

Fig. 9 Differential double ratio distribution (high-statistics over low-statistics) for the reco-to-DL ratios shown in Figs. 3 and 7 and in Figs. 4 and 8

Appendix

Resource Utilization for a Standard GEANT4-Based Generation Workflow

In this appendix, we describe how we derived the values quoted in Fig. 1. We take as a reference the CMS experiment. In absence of a published reference with a breakdown of CPU and disk resources for GEN, SIM, and DIGI+RECO steps, we derived the quantities quoted in Fig. 1 by generating QCD events on CPU, through the CERN batch system. To do so, we relied on the open-source CMSSW software [41] and followed the instructions provided by the CMS collaboration on the CERN Open Data portal [42].

We consider the same setup used to generate one of the QCD Run II samples published on the CERN Open Data portal [43] and the software installation available on CERN `cvmfs` distributed file system.

For each step, we ran jobs with 100 and 10 events. For each job, we recorded CPU time and output file size. Each

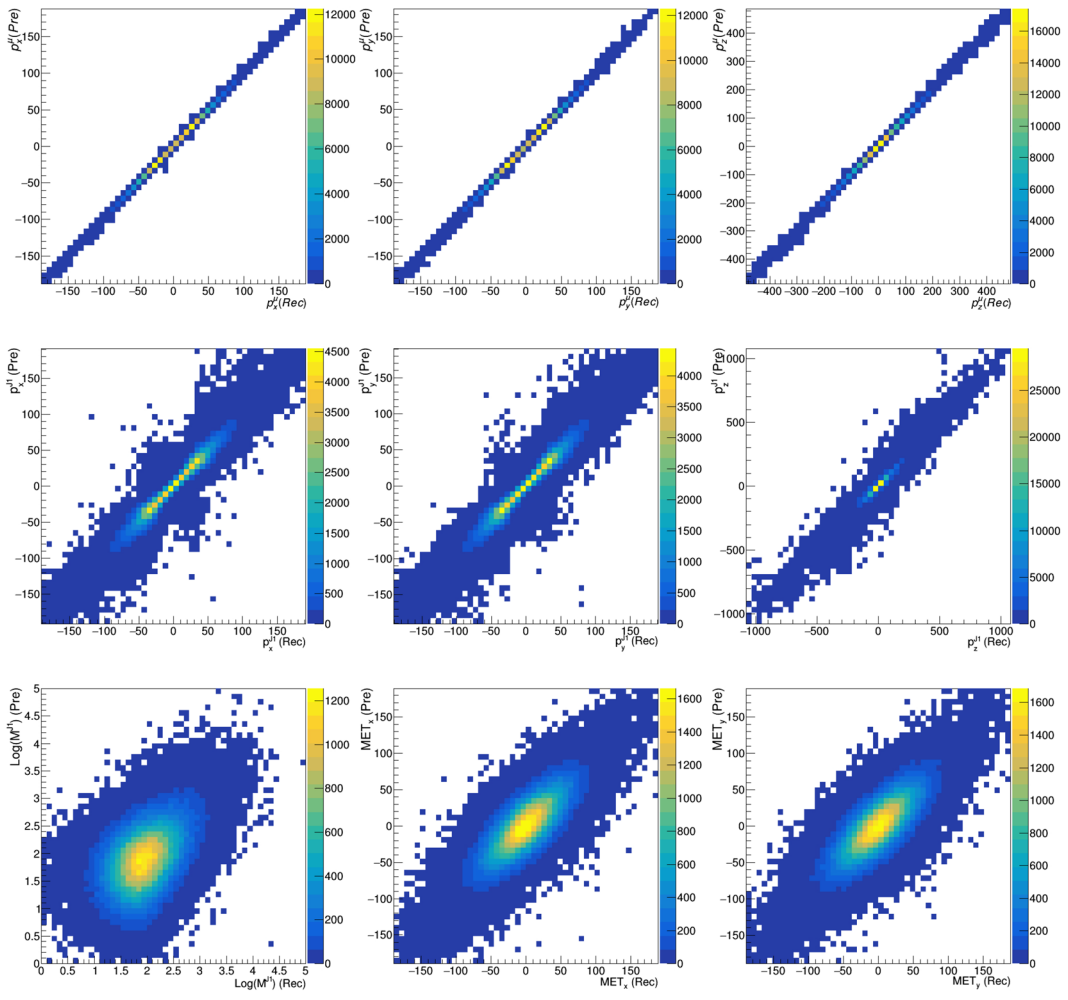

Fig. 10 Distribution of input features predicted by the model as a function of the corresponding quantities from detector simulation

step is repeated 10 times and the average of each quantity is considered. The typical uncertainty on these mean values, measured by the standard deviation of the 10 values, is found to be at most of a few percent and hence considered negligible. After computing the average for each set of jobs, we take the difference between the 100-event and 10-event job of each kind, to remove the overhead CPU time and file size that doesn't originate from per-event tasks. By dividing these differences by 90, we derive the per-event quantities quoted in Fig. 1.

Further Validation of the DL Generation Workflow

Figures 10 and 11 show the distribution predicted by the model as a function of the corresponding quantities from detector simulation, respectively for input and auxiliary features. Both the reconstruction techniques start from the generator-level information and model the detector response through a set of random degrees of freedom. The strong correlation and the symmetric distribution around the diagonal

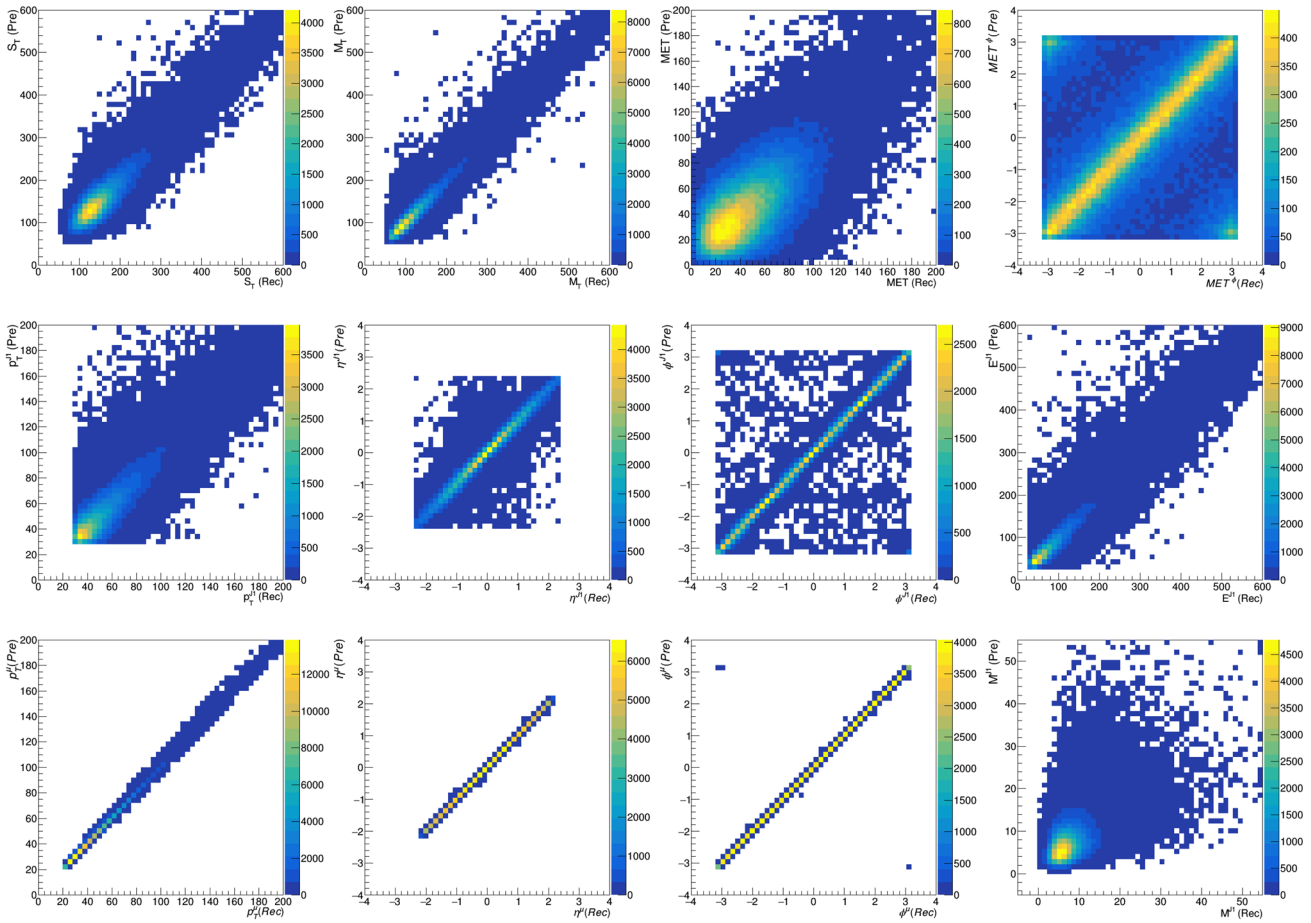

Fig. 11 Distribution of auxiliary features predicted by the model as a function of the corresponding quantities from detector simulation

demonstrate that, to a large extent, the two event representations are equivalent.

Scaling with Dataset Size

Figures 12 and 13 show the comparison between reconstructed and generated quantities with five times more data, computed from detector simulation and processing the generator-level event with our model. Qualitatively, these distributions agree with those of Figs. 5 and 6, i.e., no accuracy deterioration is observed due to the scaling of the dataset size. This fact proves the robustness of the proposed methodology and its effectiveness for data augmentation.

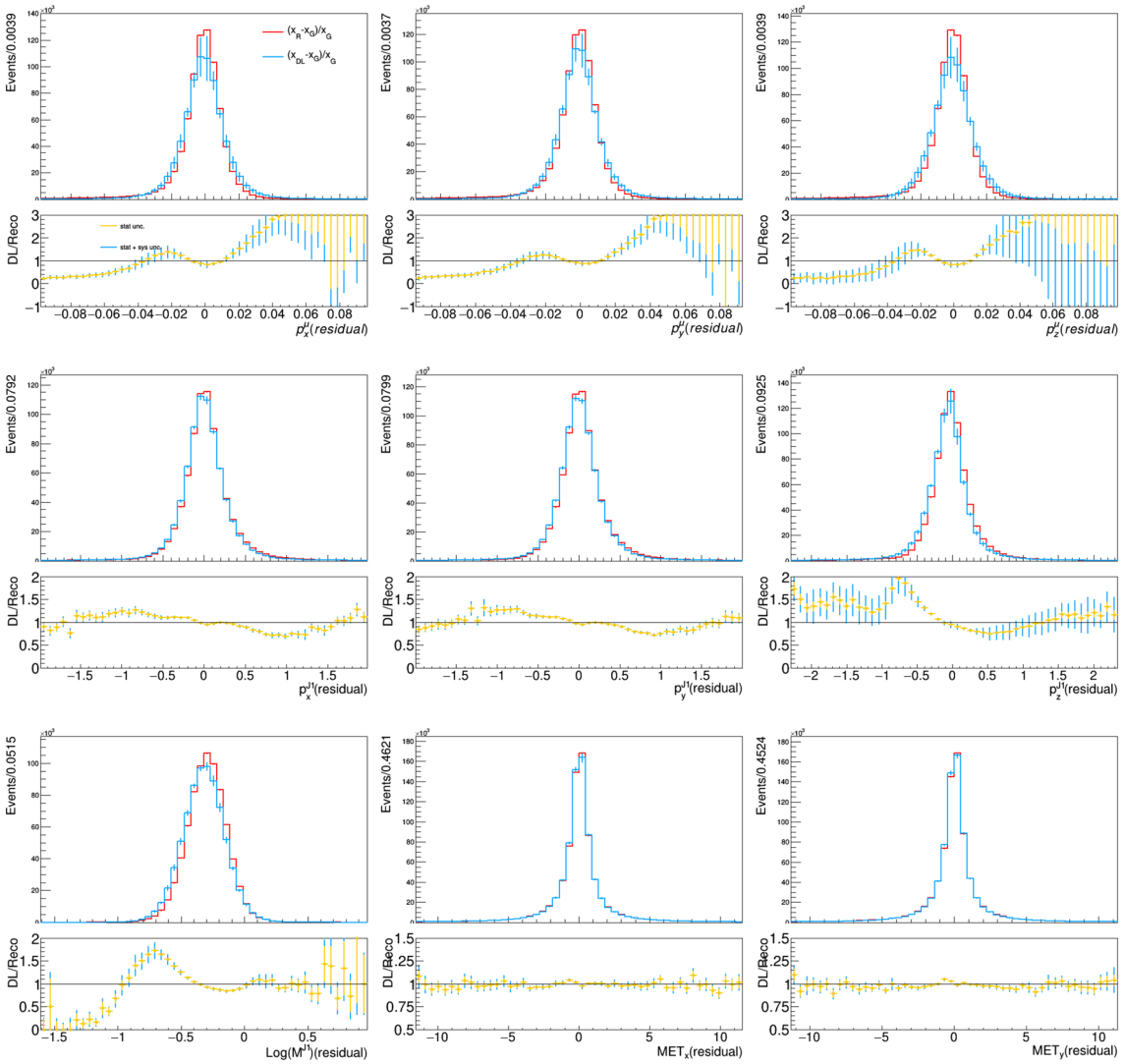

Fig. 12 Predict on an inference dataset five times larger than the training dataset. Relative residual distribution for reconstructed and model-predicted quantities in the feature vector, computing with respect to the reference input

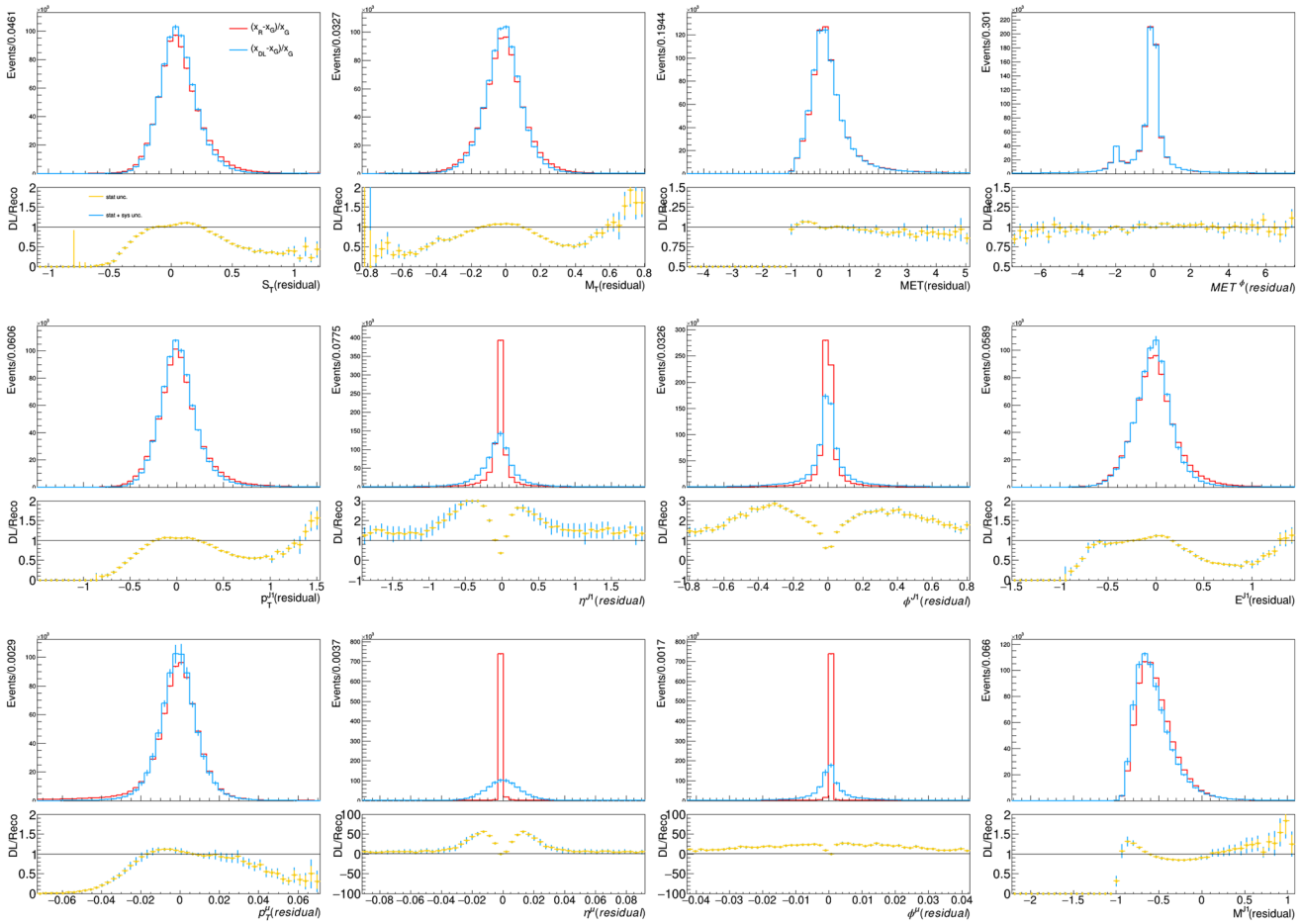

Fig. 13 Predict on an inference dataset five times larger than the training dataset. Relative residual distribution for reconstructed and model-predicted auxiliary quantities, computing with respect to the reference input

Acknowledgements This project is partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement n^o 772369) and by the United States Department of Energy, Office of High Energy Physics Research under Caltech Contract No. DE-SC0011925. This work was conducted at "*iBanks*," the AI GPU cluster at Caltech. We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of "*iBanks*."

Funding Open Access funding provided by CERN.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Agostinelli S et al (2003) GEANT4: a simulation toolkit. Nucl Instrum Method A 506:250. https://doi.org/10.1016/S0168-9002(03)01368-8
2. Albrecht J et al (2019) A Roadmap for HEP software and computing R&D for the 2020s. Comput. Softw. Big Sci. 3(1):7. https://doi.org/10.1007/s41781-018-0018-8
3. Petrucciani G, Rizzi A, Vuosalo C (2015) Mini-AOD: a new analysis data format for CMS. J Phys Conf Ser 664(7):7. https://doi.org/10.1088/1742-6596/664/7/072052
4. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks
5. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein GAN

6. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of Wasserstein GANs
7. Rezende D.J, Mohamed S, Wierstra D (2014) Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st international conference on machine learning, proceedings of machine learning research, vol 32. http://proceedings.mlr.press/v32/rezende14.html
8. Kingma DP, Welling M (2013) Auto-encoding variational bayes. ArXiv e-prints
9. Paganini M, de Oliveira L, Nachman B (2018) CaloGAN: simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks. Phys Rev D 97(1):014021. https://doi.org/10.1103/PhysRevD.97.014021
10. Erdmann M, Glombitza J, Quast T (2019) Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network. Comput Softw Big Sci 3(1):4. https://doi.org/10.1007/s41781-018-0019-7
11. Salamani D, Gadatsch S, Golling T, Stewart GA, Ghosh A, Rousseau D, Hasib A, Schaarschmidt J (2018) Deep generative models for fast shower simulation in ATLAS. In: 14th international conference on e-science, p 348. https://doi.org/10.1109/eScience.2018.00091
12. Belayneh D et al (2020) Calorimetry with deep learning: particle simulation and reconstruction for collider physics. Eur Phys J C 80(7):688. https://doi.org/10.1140/epjc/s10052-020-8251-9
13. Buhmann E, Diefenbacher S, Eren E, Gaede F, Kasieczka G, Korol A, Krüger K (2020) Getting high: high fidelity simulation of high granularity calorimeters with high speed
14. de Oliveira L, Paganini M, Nachman B (2017) Learning Particle physics by example: location-aware generative adversarial networks for physics synthesis. Comput Softw Big Sci 1(1):4. https://doi.org/10.1007/s41781-017-0004-6
15. Musella P, Pandolfi F (2018) Fast and accurate simulation of particle detectors using generative adversarial networks. Comput Softw Big Sci 2(1):8. https://doi.org/10.1007/s41781-018-0015-y
16. Carrazza S, Dreyer FA (2019) Lund jet images from generative and cycle-consistent adversarial networks. Eur Phys J C 79(11):979. https://doi.org/10.1140/epjc/s10052-019-7501-1
17. Erdmann M, Geiger L, Glombitza J, Schmidt D (2018) Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks. Comput Softw Big Sci 2(1):4. https://doi.org/10.1007/s41781-018-0008-x
18. Bishara F, Montull M (2019) (Machine) Learning amplitudes for faster event generation
19. Di Sipio R, Faucci Giannelli M, Ketabchi Haghighat S, Palazzo S (2020) DijetGAN: a generative-adversarial network approach for the simulation of QCD Dijet events at the LHC. JHEP 08:110. https://doi.org/10.1007/JHEP08(2019)110
20. Butter A, Plehn T, Winterhalder R (2019) How to GAN LHC events. Sci Post Phys 7:075. https://doi.org/10.21468/SciPostPhys.7.6.075
21. Arjona Martínez J, Nguyen T.Q, Pierini M, Spiropulu M, Vlimant JR (2019) Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description. In: 19th International workshop on advanced computing and analysis techniques in physics research: empowering the revolution: bringing machine learning to high performance computing (ACAT 2019) Saas-Fee, Switzerland, March 11–15, 2019
22. Otten S, Caron S, de Swart W, van Beekveld M, Hendriks L, van Leeuwen C, Podareanu D, Ruiz de Austri R, Verheyen R (2019) Event generation and statistical sampling for physics with deep generative models and a density information buffer
23. Hashemi B, Amin N, Datta K, Olivito D, Pierini M (2019) LHC analysis-specific datasets with Generative Adversarial Networks
24. Bellagente M, Butter A, Kasieczka G, Plehn T, Rousselot A, Winterhalder R, Ardizzone L, Köthe U (2020) Invertible networks or partons to detector and back again. Sci Post Phys 9:074. https://doi.org/10.21468/SciPostPhys.9.5.074
25. Matchev KT, Shyamsundar P (2020) Uncertainties associated with GAN-generated datasets in high energy physics
26. Butter A, Diefenbacher S, Kasieczka G, Nachman B, Plehn T (2020) GANplifying event samples
27. Cranmer KS (2001) Kernel estimation in high-energy physics. Comput Phys Commun 136:198. https://doi.org/10.1016/S0010-4655(00)00243-5
28. de Favereau J, Delaere C, Demin P, Giammanco A, Lemaître V, Mertens A, Selvaggi M (2014) DELPHES 3, a modular framework for fast simulation of a generic collider experiment. JHEP 02:057. https://doi.org/10.1007/JHEP02(2014)057
29. Sjöstrand T et al (2015) An introduction to PYTHIA 8.2. Comput Phys Commun 191:159. https://doi.org/10.1016/j.cpc.2015.01.024
30. Cacciari M, Salam GP, Soyez G (2008) JHEP 04:063. https://doi.org/10.1088/1126-6708/2008/04/063
31. Pierini M, Chen C (2020) Data augmentation at the LHC through analysis-specific fast simulation with deep learning: W + jet training/test dataset, data augmentation at the LHC through analysis-specific fast simulation with deep learning: W + jet training/test dataset . https://doi.org/10.5281/zenodo.4080943
32. Pierini M, Chen C (2020) Data augmentation at the LHC through analysis-specific fast simulation with deep learning: W + jet large test dataset, data augmentation at the LHC through analysis-specific fast simulation with deep learning: W+jet large test dataset. https://doi.org/10.5281/zenodo.4080968
33. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on deep learning for audio, speech and language processing https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf
34. Chollet F (2015) keras. https://github.com/fchollet/keras
35. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. CoRR abs/1412.6980. http://arxiv.org/abs/1412.6980
36. Hagiwara K, Kanzaki J, Li Q, Okamura N, Stelzer T (2013) Fast computation of MadGraph amplitudes on graphics processing unit (GPU). Eur Phys J C 73:2608. https://doi.org/10.1140/epjc/s10052-013-2608-2
37. Klimek MD, Perelstein M (2020) Neural network-based approach to phase space integration. Sci. Post Phys. 9:053. https://doi.org/10.21468/SciPostPhys.9.4.053
38. Gao C, Isaacson J, Krause C (2020) i-flow: high-dimensional integration and sampling with normalizing flows. Mach Learn Sci Technol 1(4):045023. https://doi.org/10.1088/2632-2153/abab62
39. Gao C, Höche S, Isaacson J, Krause C, Schulz H (2020) Event generation with normalizing flows. Phys Rev D 101(7):076002. https://doi.org/10.1103/PhysRevD.101.076002
40. Carrazza S, Cruz-Martinez JM (2020) VegasFlow: accelerating Monte Carlo simulation across multiple hardware platforms. Comput Phys Commun 254:107376. https://doi.org/10.1016/j.cpc.2020.107376
41. Cmssw framework. https://github.com/cms-sw/cmssw
42. Cms open data (2015) http://opendata.cern.ch/search?experiment=CMS
43. Simulated dataset QCD_Pt_470to600_TuneCUETP8M1_13TeV_pythia8. In: MINIAODSIM format for 2016 collision data. https://doi.org/10.7483/OPENDATA.CMS.HBBW.LTT4. CERN Open Data Portal http://opendata.cern.ch/record/12013,