Title
Neural conditional reweighting

Journal
Physical Review D, 105(7)

ISSN
2470-0010

Authors
Nachman, Benjamin
Thaler, Jesse

Publication Date
2022-04-01

DOI
10.1103/physrevd.105.076015

Peer reviewed

Neural Conditional Reweighting

Benjamin Nachman[1, 2, *] and Jesse Thaler[3, 4, †]

[1] *Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[2] *Berkeley Institute for Data Science, University of California, Berkeley, CA 94720, USA*
[3] *Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[4] *The NSF AI Institute for Artificial Intelligence and Fundamental Interactions*

There is a growing use of neural network classifiers as unbinned, high-dimensional (and variable-dimensional) reweighting functions. To date, the focus has been on marginal reweighting, where a subset of features are used for reweighting while all other features are integrated over. There are some situations, though, where it is preferable to condition on auxiliary features instead of marginalizing over them. In this paper, we introduce neural conditional reweighting, which extends neural marginal reweighting to the conditional case. This approach is particularly relevant in high-energy physics experiments for reweighting detector effects conditioned on particle-level truth information. We leverage a custom loss function that not only allows us to achieve neural conditional reweighting through a single training procedure, but also yields sensible interpolation even in the presence of phase space holes. As a specific example, we apply neural conditional reweighting to the energy response of high-energy jets, which could be used to improve the modeling of physics objects in parametrized fast simulation packages.

CONTENTS

I. INTRODUCTION

A common task in particle physics is to reweight one set of events P to match the statistical properties of another set of events Q. Here, $P = \{x_i\}$, $x_i \in \mathbb{R}^N$, are drawn independently and identically distributed from probability density $p(x)$, and Q is similarly drawn from $q(x)$. The reweighting function,

$$w(x) \approx \frac{q(x)}{p(x)}\,, \tag{1}$$

ensures that the expectation value of any weighted observable computed from P will match the same value computed from Q on average.[1] For example, P could be events from a control region while Q are events from a signal region, or P could be from simulation while Q could be from data, or P and Q could be from two different simulations with different parameter choices.

In nearly every case of interest in particle physics, p and q are not known analytically. When x is low-dimensional, it is common to create histograms to estimate p and q from the events in P and Q. One can then construct a binned reweighting function by taking ratios of the bin contents. This works well when $w(x)$ is slowly varying and x is low- (and fixed-) dimensional. When these conditions are not met, the traditional binned approach is not effective.

Neural network classifiers can be used to form unbinned, high- (and variable-) dimensional reweighting functions, which can be viewed as an application of simulation-based inference (see Ref. [2] for a review). In particular, the optimal classifier for distinguishing events drawn from P and Q is (any monotonic function of) the likelihood ratio $q(x)/p(x)$. Therefore, one can approximate $w(x)$ directly by interpreting the output of a classifier trained to distinguish the two event samples. This feature of classifiers is well known [3, 4] and has been widely used in particle physics for parameter estimation [5–13], domain adaptation [14], detector parameterizations [15], and unfolding [16–19].

* bpnachman@lbl.gov
† jthaler@mit.edu

[1] In certain cases, it is possible to resample the events to match the statistical uncertainties as well [1].

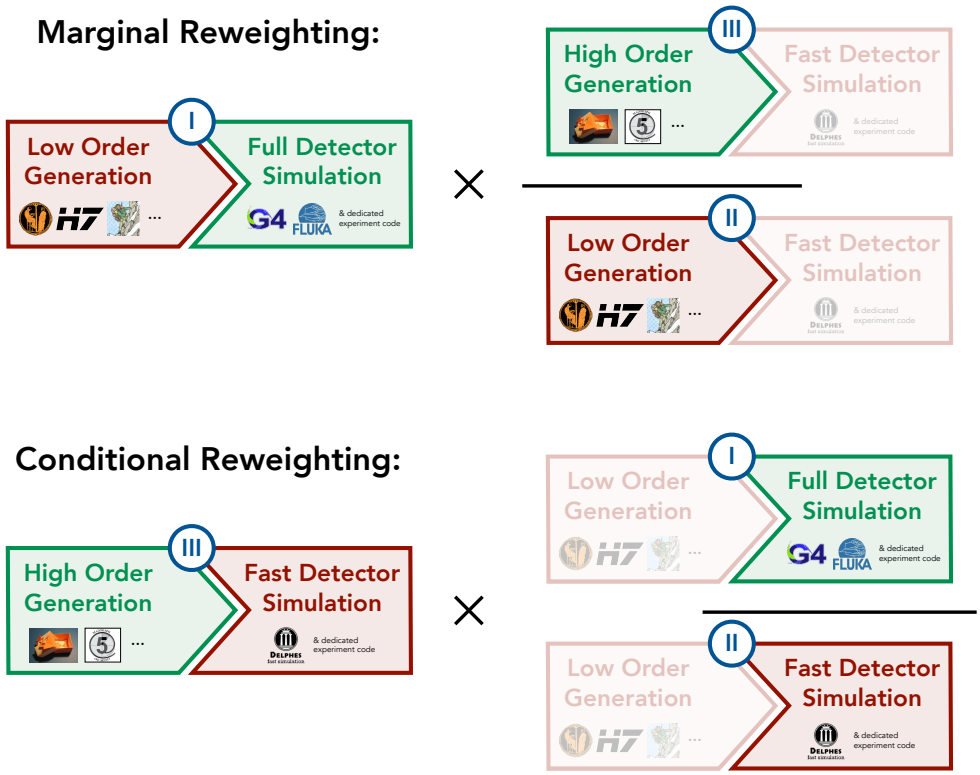

FIG. 1. Schematic diagrams contrasting marginal reweighting (top) with conditional reweighting (bottom), in the context of generation and simulation for collider physics. The goal is to create an event sample that has the particle-level kinematics of a high-order generator (e.g. POWHEG-BOX [20–22] or MG5_aMC [23]) with the detector-level reconstruction of a full detector simulation (based on e.g. GEANT 4 [24–26] or FLUKA [27, 28]). In marginal reweighting, one reweights events from a low-order generator (e.g. PYTHIA [29, 30], HERWIG [31, 32], or SHERPA [33, 34]) to match the kinematics of a high-order generator, marginalizing over the simulator. In conditional reweighting, one reweights events from a fast simulation (e.g. based on DELPHES [35–37]) to match the reconstruction of a full detector simulation, conditioning on the generator.

To our knowledge, in all applications to date of classifier-based reweighting, other event features x' are integrated over, such that:

$$w(x) \approx \frac{\int dx'\, q(x,x')}{\int dx'\, p(x,x')}\,. \tag{2}$$

This marginalization is often necessary when x' is not observable, as is the case when x represents detector-level quantities and x' represents particle-level quantities. This can be an issue, however, if $w(x)$ is applied to another data set where the probability density of x' is not the same as $q(x')$. For example, suppose that x' represents the particle-level jet energy, x is the detector-level jet energy, $p(x)$ represents the probability density of a fast simulation, and $q(x)$ is the probability density for a full simulation. One can train a model to reweight $p(x)$ to $q(x)$ to match the detector resolution, but if $p(x') \neq q(x')$, then there is a degeneracy between physics and detector effects. Even if $p(x') = q(x')$ (or if one reweights x and x' simultaneously), the reweighting function cannot be applied to another data set with a different energy distribution. It would therefore be ideal to reweight the conditional probabilities instead, such that:

$$w(x) \approx \frac{q(x|x')}{p(x|x')}. \tag{3}$$

In this paper, we introduce neural conditional reweighting, which is a strategy to extract the conditional probability ratio in Eq. (3). We first show how to achieve conditional reweighting by training two independent classifiers, one for joint reweighting and one for marginal reweighting. We then develop a custom loss function specifically for conditional reweighting, which is better suited to situations with phase space holes. Through a single training procedure, the resulting neural network can sensibly interpolate across minimally populated regions of phase space. We demonstrate the efficacy of our approach using simple Gaussian examples and a more realistic application in collider physics.

The primary motivating application of neural condi-

tional reweighting is shown in Fig. 1, where the goal is to improve generation and simulation for collider physics.[2] Here, we have three synthetic data sets:

- (I): Coarse Generator ⇒ Precise Simulator;
- (II): Coarse Generator ⇒ Coarse Simulator;
- (III): Precise Generator ⇒ Coarse Simulator.

Data sets (I) and (II) use a coarse particle-level generator while data set (III) uses a precise particle-level generator. By contrast, data sets (II) and (III) use a coarse detector-level simulator while data set (I) uses a precise detector-level simulator. The goal is to create a data set that has the most precise particle-level generation and the most precise detector-level simulation, which requires merging the best features of data sets (III) and (I), respectively. One way to construct this merged data set is to perform a marginal reweighting from the coarse particle-level truth to the precise particle-level truth, shown in the top line of Fig. 1. Here, we advocate for conditional reweighting, where we reweight only the detector response from (II) to (I) and then apply this to data set (III), shown in the bottom line of Fig. 1.

In the limit of infinite statistics and no phase space holes, both marginal reweighting and conditional reweighting yield the same final distributions. The aim of this paper is to highlight situations where conditional reweighting could outperform marginal reweighting in practical situations. In principle, one could bypass data set (II) entirely and directly conditional reweight (III) to (I), but we will argue that this is likely never better than marginal reweighting. Beyond reweighting, once can train surrogate models for generation and simulation (see Refs. [38, 39] for reviews), which we do not consider here.

The remainder of this paper is organized as follows. In Sec. II, we review neural reweighting and generalize the marginal version to the conditional case. We present a simple Gaussian example to illustrate the complementary of conditional and marginal reweighting in Sec. III. In Sec. IV, we present an application of neural conditional reweighting in the context of jet energy measurements at the Large Hadron Collider (LHC). The paper ends with our conclusions and outlook in Sec. V.

II. THE STATISTICS OF CONDITIONAL REWEIGHTING

A. Review of Marginal Reweighting

Let $f : \mathbb{R}^N \to [0,1]$ be a classifier with the goal of distinguishing events generated by probability densities p and q. This function can be obtained by minimizing an appropriate loss functional, such as the binary cross entropy (BCE):

$$L_{\text{BCE}}[f] = -\int dx \left(p(x) \log f(x) + q(x) \log(1 - f(x)) \right). \tag{4}$$

In practice, with finite training data, we would replace

$$\int dx\, p(x) \Rightarrow \sum_{x_i \in P}, \tag{5}$$

but for the remainder of this discussion, we consider the infinite statistics limit such that we can replace sums over events by integrals and then use functional optimization to determine the optimal classifier f.

The function f_{BCE} that optimizes the functional in Eq. (4) has the following well-known property [3, 4]:

$$\frac{1 - f_{\text{BCE}}(x)}{f_{\text{BCE}}(x)} = \frac{q(x)}{p(x)}, \tag{6}$$

such that one learns the per-instance likelihood ratio in the asymptotic limit. Note that this analysis assumes the same number of events sampled from q and p; if these are not the same, then Eq. (6) is multiplied by the relative frequency of the two random variables (prior ratio). Similar formulae apply to other loss functionals, and certain loss functionals such as the maximum likelihood classifier (MLC) loss [40, 41] result in classifiers that directly approximate the likelihood ratio without the transformation in Eq. (6).

In the case that the feature space consists of observed features $x \in \mathbb{R}^N$ and hidden (or latent) features $x' \in \mathbb{R}^M$, but the classifier f is only a function of x, then the learned function is related to the marginalized likelihood ratio:

$$\frac{1 - f_{\text{BCE}}(x)}{f_{\text{BCE}}(x)} = \frac{q(x)}{p(x)} \equiv \frac{\int dx'\, q(x,x')}{\int dx'\, p(x,x')}. \tag{7}$$

We call this procedure *marginal reweighting*. Note that the same symbols p and q are used to denote the marginal (e.g. $p(x), p(x')$) and joint (e.g. $p(x,x')$) probability densities, and primes are used to separate observed and latent quantities.

If, instead, we consider a classifier $f : \mathbb{R}^{N+M} \to [0,1]$ that depends on the full $(N+M)$-dimensional feature space, then the optimal learned function is related to the joint likelihood ratio:

$$\frac{1 - f_{\text{BCE}}(x,x')}{f_{\text{BCE}}(x,x')} = \frac{q(x,x')}{p(x,x')}, \tag{8}$$

and we call this *joint reweighting*.

A challenge faced by marginal (and to a lesser extent joint) reweighting is that the weights can become large and unphysical if $q(x)$ and $p(x)$ do not have overlapping support. In particular, if there is a region of phase space where $p(x) \simeq 0$, then Eq. (7) becomes singular. When this happens, conditional reweighting offers an alternative reweighting strategy.

[2] To avoid overlap in word usage, we use the word "generator" to refer to particle-level simulation tools, and "simulator" to refer to detector-level simulation tools.

B. Conditional Reweighting with Two Classifiers

We can easily extend the above formalism to conditional reweighting by noting the following:

$$\frac{q(x|x')}{p(x|x')} \equiv \frac{\frac{q(x,x')}{q(x')}}{\frac{p(x,x')}{p(x')}} = \frac{q(x,x')}{p(x,x')}\,\frac{p(x')}{q(x')}\,. \tag{9}$$

The first term is the joint reweighting in Eq. (8). The second term is the inverse of the marginal reweighting in Eq. (7), with the roles of x and x' reversed. Therefore, one can achieve conditional reweighting with two functions, each trained as a standard classifier.

A potential challenge with applying Eq. (9) in practice is that $q(x')$ might have inadequate support relative to $p(x')$ in some regions of phase space, leading to ill-behaved weights. Note that this is effectively the opposite problem as faced by marginal reweighting, so it is typically less of an issue in practice. That said, we can partially mitigate this issue by leveraging the ability of neural networks to interpolate.

C. Conditional Reweighting with a Single Classifier

A natural question is whether conditional reweighting could be learned in one learning step, instead of in two steps as in the above construction. A somewhat trivial way to accomplish this is to note that

$$\frac{q(x|x')}{p(x|x')} = \lim_{y'\to x'} \frac{q(x,x')\,p(y')}{p(x,x')\,q(y')}, \tag{10}$$

where $y' \in \mathbb{R}^M$. Therefore, to learn this ratio, we could train a classifier $f(x,x',y')$ to distinguish *pairs* of events drawn from $p(x,x')\,q(y')$ versus $q(x,x')\,p(y')$, and then set $x' = y'$. The reason this is somewhat trivial is that, assuming the BCE loss, the optimal classifier factorizes into two separate classifiers,

$$\frac{1-f_{\text{BCE}}(x,x',y')}{f_{\text{BCE}}(x,x',y')} = \frac{1-g_{\text{BCE}}(x,x')}{g_{\text{BCE}}(x,x')}\,\frac{h_{\text{BCE}}(y')}{1-h_{\text{BCE}}(y')}\,, \tag{11}$$

which might as well be optimized separately as in Eq. (9).

A more interesting construction follows from the relation:

$$\frac{q(x|x')}{p(x|x')} = \lim_{y'\to x'} \frac{q(x,y')\,p(x')}{p(x,x')\,q(y')}\,, \tag{12}$$

where the primed arguments in the numerator have been flipped relative to Eq. (10). Before taking $x' = y'$, we can learn this ratio with a new neural conditional reweighting (NCR) loss functional:

$$L_{\text{NCR}}[f] = -\int dx\,dx'\,dy\,dy'\,p(x,x')\,q(y,y') \times \Big(\log f(x,x',y') + \log(1-f(y,x',y'))\Big)\,. \tag{13}$$

Swapping the x and y integral labels in the second term, it is straightforward to show that the optimal classifier is:

$$\frac{1-f_{\text{NCR}}(x,x',y')}{f_{\text{NCR}}(x,x',y')} = \frac{q(x,y')\int dy\,p(y,x')}{p(x,x')\int dy\,q(y,y')}\,. \tag{14}$$

Inserting this into Eq. (12), we find

$$\frac{1-f_{\text{NCR}}(x,x',x')}{f_{\text{NCR}}(x,x',x')} = \frac{q(x|x')}{p(x|x')}\,, \tag{15}$$

which is our default approach to conditional reweighting.[3]

The NCR loss in Eq. (13) is similar to the BCE loss in Eq. (4), but instead of the events sampled from p and q contributing separately to the two terms, the events contribute to both terms.[4] For $x,y \in \mathbb{R}^N$ and $x',y' \in \mathbb{R}^M$, the density $p(x,x')\,q(y,y')$ means that a $(2N+2M)$-dimensional data set is sampled from p and q independently. In practice, one can approximate Eq. (13) by using the standard BCE loss with one $(N+2M)$-dimensional data set sampled from $p(x,x')\,q(y')$ with a label of 1 and a second data set sampled from $p(x')\,q(y,y')$ with a label of 0.

Because Eq. (15) is obtained from the $y' \to x'$ limit, we can partially mitigate the issue from Sec. II B of ill-behaved weights when there are dead regions of phase space. This works because neural networks typically yield sensible and smooth interpolations across the training domain, so we can use $y' \neq x'$ information to predict the behavior at $y' = x'$. As shown in App. A, we find only modest differences between using one classifer trained using the NCR loss (Eq. (15)) and two classifers each trained with the BCE loss (Eq. (11)). We find larger differences when comparing conditional reweighting against marginal reweighting, where the issue of dead phase space regions is more pronounced.

In the example application shown in Fig. 1, conditional reweighting is learned from two data sets (I) and (II) that have the same low-order generator, which means that $p(x') = q(x')$. From the above derivation, though, we see that this restriction is not necessary, and conditional reweighting can be learned for any $p(x')$ and $q(x')$ with overlapping support, a fact we leverage in Sec. IV. In practice, though, it is helpful for $p(x')$ and $q(x')$ to be similar, not only to ensure overlapping support but also to avoid unnecessarily large weights This is the reason why we recommend using data set (II) to derive the conditional reweighting factor in Fig. 1, instead of directly conditional reweighting (III) to (I).

---

[3] The approach in Eq. (10) corresponds to replacing $f(y,x',y')$ with $f(y,y',x')$ in the second term of Eq. (13).

[4] This form is similar to the setup in Ref. [41–43] where events are combined to use deep learning as way to estimate mutual information. See Ref. [44] for a related construction.

D. Technical Implementation

In our subsequent case studies, the functions f trained with the NCR loss will be parameterized with neural networks. While it is possible to train a generic function $f(x, x', y')$, we can take advantage of the known form of the optimal solution. Rewriting Eq. (14), the optimal f takes the form

$$f_{\text{NCR}}(x, x', y') = \frac{p(x|x')}{p(x|x') + q(x|y')}. \tag{16}$$

Interestingly, although f is naively a function of three variables, the optimal function can be expressed in terms of two functions of two variables each.

Armed with this insight, we construct our classifiers as

$$f(x, x', y') = \frac{e^{f_0(x,x')}}{e^{f_0(x,x')} + e^{f_1(x,y')}}, \tag{17}$$

where f_0, f_1 are each neural networks. The exponential is used because each term must be non-negative (as a conditional probability density). In fact, since f_0 and f_1 are expected to be similar log likelihoods, we can further simplify the problem by building f_0 and f_1 from the same components:

$$f_0(x, x') = W_0' \max(0, W_0\, g(x, x') + b_0) + b_0', \tag{18}$$
$$f_1(x, y') = W_1' \max(0, W_1\, g(x, y') + b_1) + b_1', \tag{19}$$

where $g : \mathbb{R}^{N+M} \to \mathbb{R}^{L_0}$ is a neural network, $W_i \in \mathbb{R}^{L_0 \times L_1}, W_i' \in \mathbb{R}^{L_1 \times 1}$ are weight matrices, and $b_i \in \mathbb{R}^{L_1}, b_i' \in \mathbb{R}$ are biases. In other words, f_0 and f_1 are shallow neural networks with a single hidden layer of size L_1 with the Rectified Linear Unit (ReLU) activation function that take as input a common deep neural network that outputs size L_0.

All neural networks are implemented using Keras [45] with the Tensorflow backend [46] and optimized with Adam [47]. Because we use BCE-like loss functions, Eq. (6) is needed to convert the classifier output to a likelihood ratio. The marginal reweighting networks consist of three hidden layers with 50 nodes per layer. The ReLU activation function is used for the intermediate layers while a sigmoid activation is used for the last layer. Each network is trained for 50 epochs with early stopping using a patience of 10 and deploys a batch size of 1000. Conditional reweighting uses the same training schedule and a similar network architecture: the g function in Eqs. (18) and (19) has two hidden layers with 50 nodes per layer and the ReLU activation. The shallow f_i networks have $L_1 = 50$.

III. GAUSSIAN EXAMPLES

We now present simple numerical examples that explore when conditional reweighting may be as good as or superior to marginal reweighting. Here, each data set in Fig. 1 is a one-dimensional Gaussian random variable. The "particle-level truth" random variables T_i are described by means μ_i and standard deviations σ_i with:

$$\mu_0 \equiv \mu_{(\text{I})} = \mu_{(\text{II})}, \qquad \sigma_0 \equiv \sigma_{(\text{I})} = \sigma_{(\text{II})}, \tag{20}$$
$$\mu_1 \equiv \mu_{(\text{III})}, \qquad \sigma_1 \equiv \sigma_{(\text{III})}. \tag{21}$$

The corresponding "detector-level reconstructed" random variables R_i are given by

$$R_i = T_i + Z_i, \tag{22}$$

where Z_i is a Gaussian random variable with mean b_i and standard deviation ϵ_i, with:

$$b_0 \equiv b_{(\text{I})}, \qquad \epsilon_0 \equiv \epsilon_{(\text{I})}, \tag{23}$$
$$b_1 \equiv b_{(\text{II})} = b_{(\text{III})}, \qquad \epsilon_1 \equiv \epsilon_{(\text{II})} = \epsilon_{(\text{III})}. \tag{24}$$

The desired target distribution combines the generation parameters of data set (III) with the simulation parameters of (I):

$$\mu_{\text{target}} = \mu_1, \qquad \sigma_{\text{target}} = \sigma_1, \tag{25}$$
$$b_{\text{target}} = b_0, \qquad \epsilon_{\text{target}} = \epsilon_0. \tag{26}$$

In each of the examples below, one million events are used for each data set with a 50% test-train split. In the collider physics context, one could use more events from data sets (II) and (III), as they are computationally cheaper to produce than (I), which involves the full simulation. None of these parameters were optimized, but we find that the results are stable to small changes in the setup. For conditional reweighting, the reweighter in Eqs. (17), (18), and (19) is trained using data sets (I) and (II) and then applied to data set (III); alternative implementations of conditional reweighting are shown in App. A.

A. Overlapping Support

For our first example, we consider a situation with no phase space gaps, such that marginal reweighting is expected to already perform well. The particle-level generation parameters are

$$\mu_0 = 0.0, \quad \mu_1 = 0.1, \quad \sigma_0 = 1.0, \quad \sigma_1 = 1.5, \tag{27}$$

such that the distributions have overlapping support. The detector-level simulation parameters are

$$b_0 = 0.0, \quad b_1 = -0.2, \quad \epsilon_0 = 0.5, \quad \epsilon_1 = 0.3, \tag{28}$$

so that the distortions are relatively small.

The results of marginal and conditional reweighting are shown in the top row and bottom row of Fig. 2, respectively. The left column shows the truth distribution for T_i while the right column shows the reconstructed distribution for R_i. Both marginal and conditional reweighting

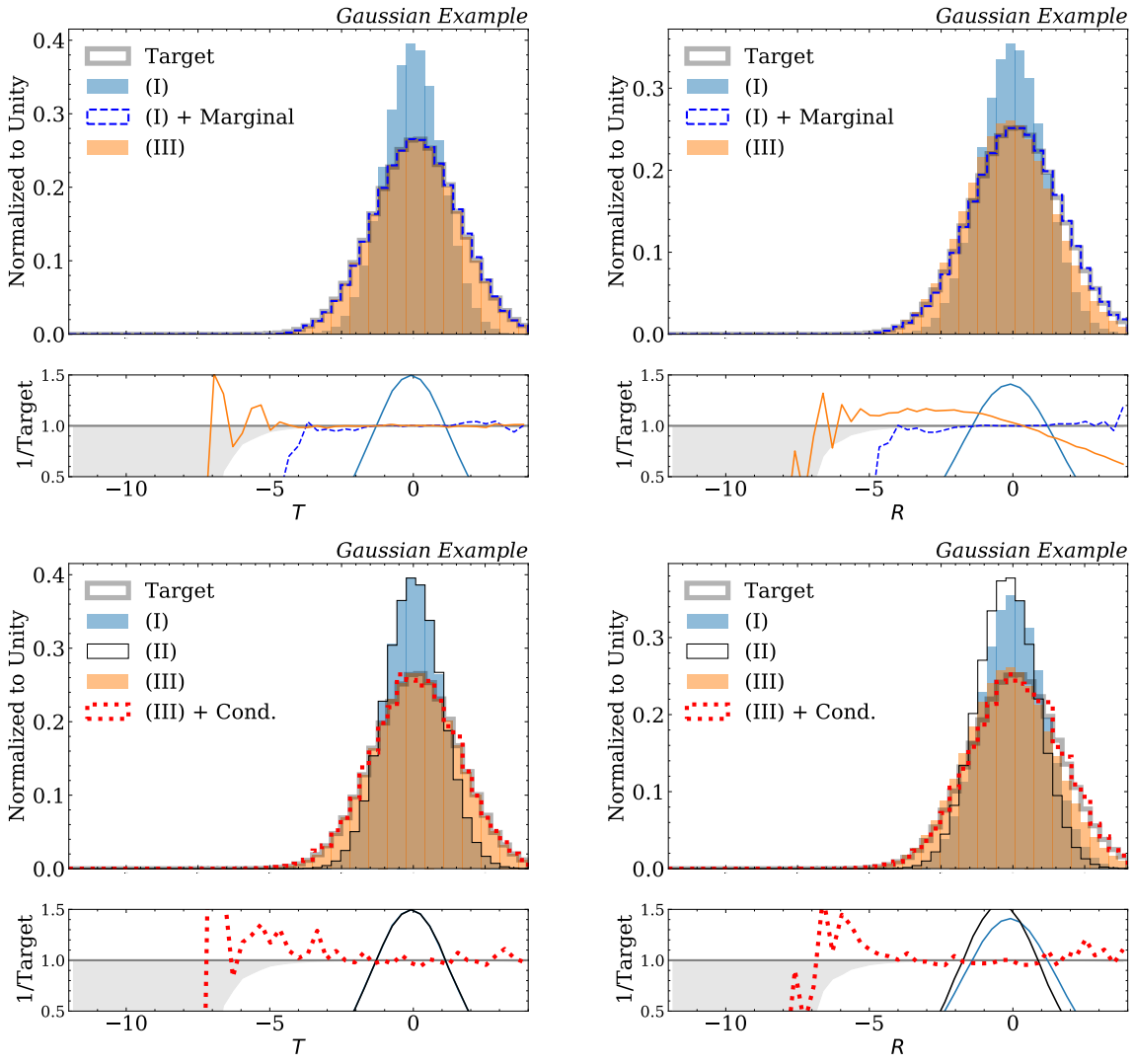

FIG. 2. Comparison of marginal reweighting (top row) and conditional reweighting (bottom row) in a plain Gaussian example. Histograms of the random variables are show at "particle level" (left column) and "detector level" (right column). Distribution (I) involves a coarse generator interfaced with a precise simulator. Distribution (II) involves a coarse generator interfaced with a course simulator. Distribution (III) involves a precise generator interfaced with a coarse simulator. To match the target distribution (precise generator interfaced with a precise simulator), one can either marginally reweight distribution (I) or conditionally reweight distribution (III). In this case, marginal reweighting yields better performance than conditional reweighting.

are able to achieve the target distribution. In particular, the conditionally reweighted (III) and the marginally reweighted (I) have the same particle-level distributions as data set (III), but the detector response of data set (I).

Upon careful inspection, one can see that marginal reweighting is able to match the target distribution more precisely than conditional reweighting, especially in the tails of the Gaussians. The exact agreement with the target varies with different pseudoexperiments and with different random initializations of the networks. However, this trend is robust: marginal reweighting is more precise than conditional reweighting in this context. This is to be expected because the conditional path in Fig. 1 requires a more complicated setup and involves a higher-dimensional learning problem compared to marginal reweighting. Without any phase space gaps, the marginal reweighting strategy in Eq. (7) is sufficient.

B. Extrapolation

To understand a context where conditional reweighting might be able to outperform marginal reweighting, consider the situation where there is a large hierarchy in the particle-level truth distributions:

$$p_{(\mathrm{I})}(T) \ll p_{(\mathrm{III})}(T). \tag{29}$$

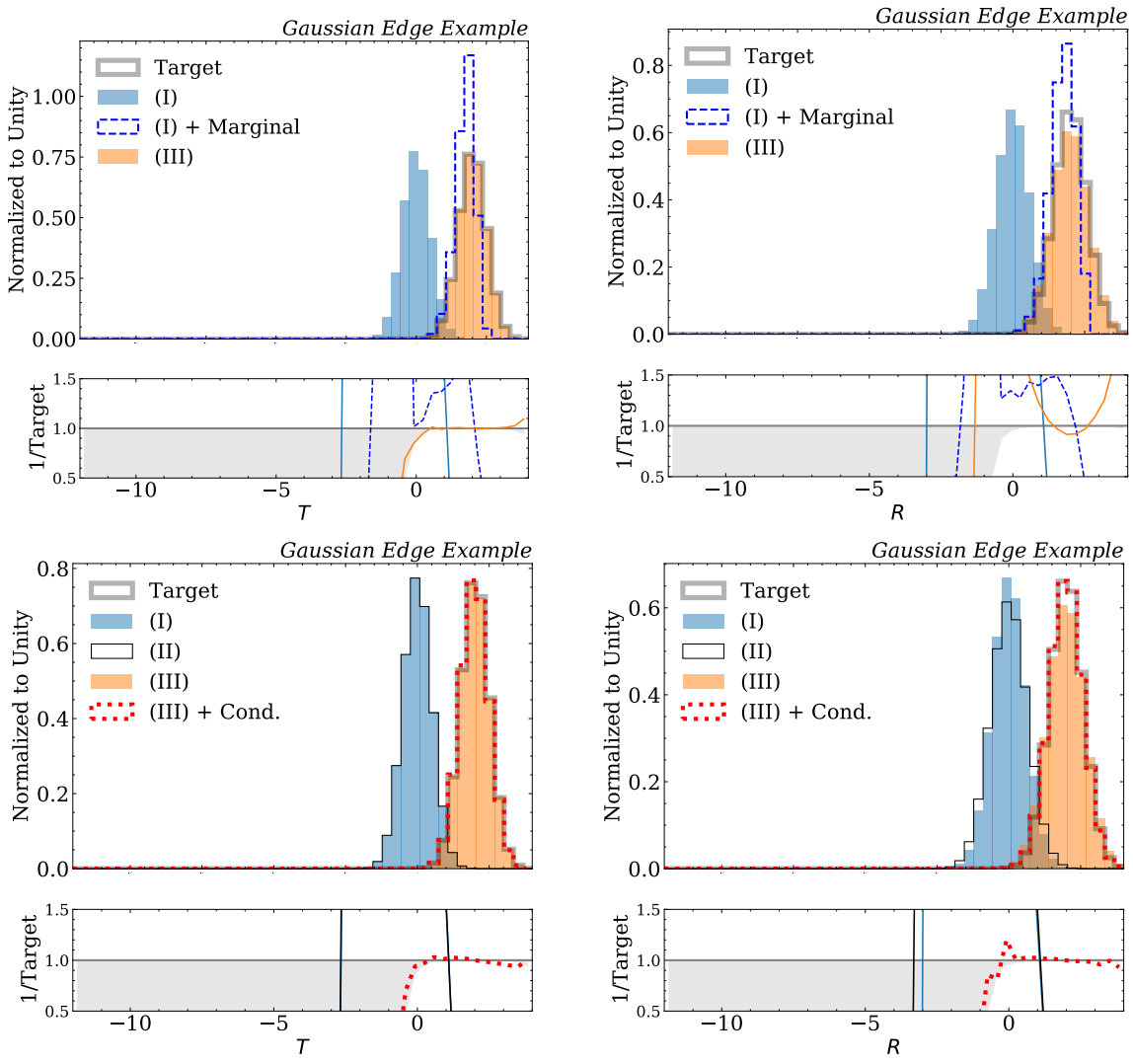

FIG. 3. Same as Fig. 2, but for a Gaussian edge example involving extrapolation. Marginal reweighting only yields sensible results where distributions (I) and (III) have significant phase space overlap. Conditional reweighting, by contrast, is able to extrapolate outside the naive training domain.

In this case, the marginal weights in Eq. (7) can become very large. Conditional reweighting may also fail in this context, but if the truth distribution in data set (II) is chosen to be close to the truth in data set (I), then at least one does not encounter large weights. To the extent that neural networks can sensibly extrapolate outside of the training domain, one can then conditionally reweight the reconstructed data set (III) to the target distribution. Of course, one has to be do a careful validation in any situation that involves extrapolation.

To explore the performance of conditional reweighting for extrapolation, the particle-level generation parameters for this example are:

$$\mu_0 = 0.0, \quad \mu_1 = 2.0, \quad \sigma_0 = \sigma_1 = 0.5, \tag{30}$$

such that there is very little overlap in their support, with the mean of the (III) truth being four standard deviations away from the mean of the (I) truth. On the other hand, the detector-level simulation parameters are:

$$b_0 = b_1 = 0.0, \quad \epsilon_0 = 0.3, \quad \epsilon_1 = 0.4, \tag{31}$$

such that the difference in the smearing behavior is relatively small.

The performance of marginal and conditional reweighting for extrapolation is presented in Fig. 3, with the same layout as Fig. 2. For $T \lesssim 1$, where $p_{(\mathrm{I})}(T) \gtrsim p_{(\mathrm{III})}(T)$, marginal reweighting is effective for the particle-level truth. For $T \gtrsim 1$, though, marginal reweighting fails to reproduce the target distribution because there are either few or no events in data set (I) to upweight. These same trends are present at detector-level in the upper right plot of Fig. 3.

By contrast, conditional reweighting is able to match both the truth and reconstructed distributions. Because

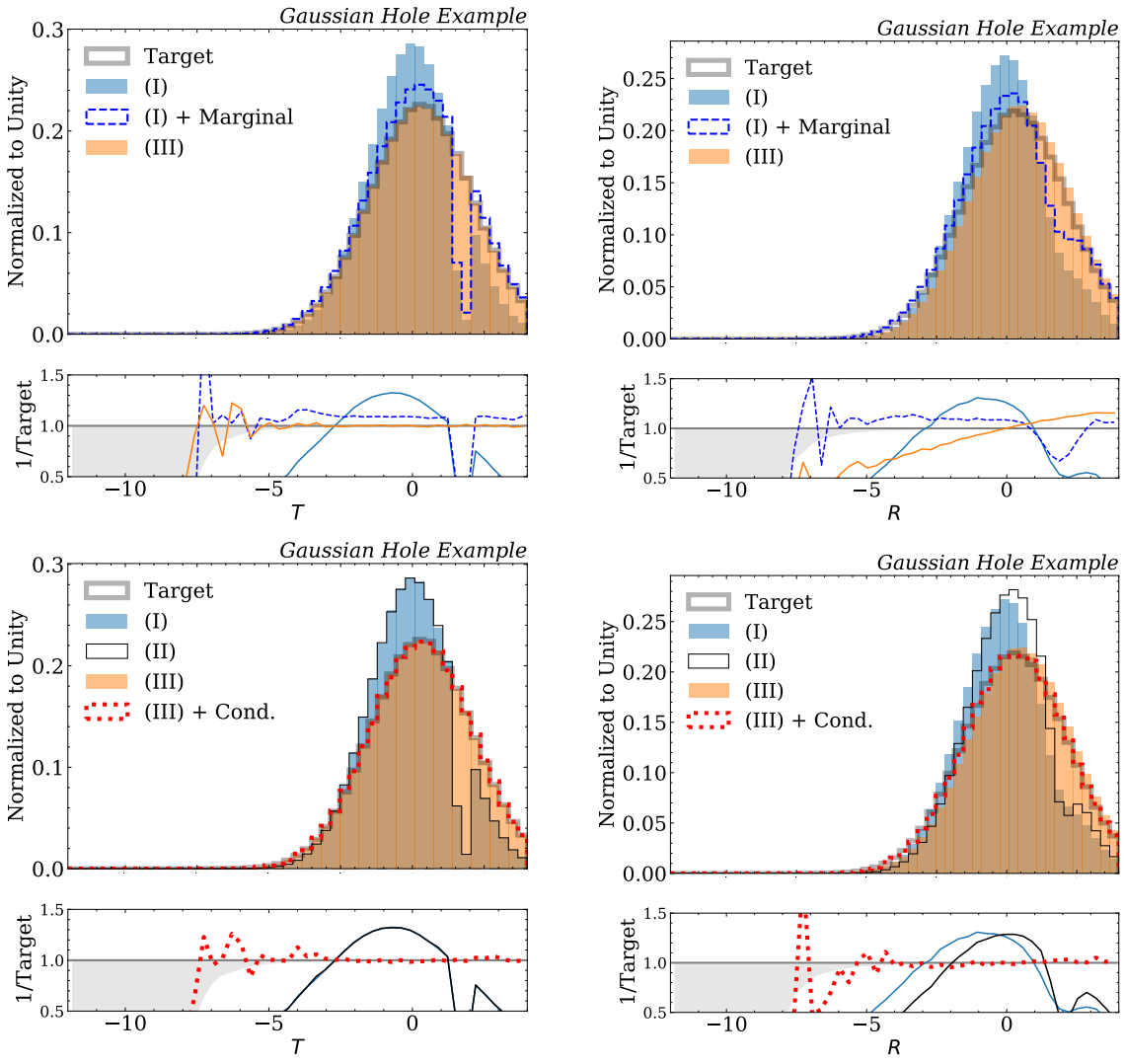

FIG. 4. Same as Fig. 2, but for a Gaussian hole example involving interpolation. Whereas marginal reweighting cannot accurately model the phase space gap, conditional reweighting is able to sensibly interpolate.

detector effects in this case are so similar between data sets (I) and (II), the conditional reweighting is nearly constant. For the particle-level truth, the good agreement is more or less guaranteed by construction, since data set (III) already has the desired truth distribution. For the detector-level reconstruction, there is no information to constrain the conditional reweighting for $R \gtrsim 1$, but the neural network is nevertheless able to smoothly extrapolate from the region of phase space with plenty of events.

We therefore expect that when the reweighting function is constant or smoothly continues from its behavior in regions with high density overlap, conditional reweighting may be as precise as it is in this example. This example also motivates explicitly prioritizing smooth extrapolation as part of the training loss.

C. Interpolation

Neural networks are known to be effective at interpolating, which is in general less fraught than extrapolation. There are known cases where regions of phase space may be undercovered by certain generators (e.g. dead zones in Herwig 7 [48]) or where a reweighting derived from one process needs to be applied to another with significantly different phase space distributions. This will be the context that we study for the jet energy response example in Sec. IV.

To study the interpolation case for the Gaussian example, we start with generation parameters

$$\mu_0 = 0.0, \quad \mu_1 = 0.3, \quad \sigma_0 = 1.5, \quad \sigma_1 = 1.8, \tag{32}$$

and simulation parameters

$$b_0 = 0.0, \quad b_1 = 0.2, \quad \epsilon_0 = 0.5, \quad \epsilon_1 = 0.3, \tag{33}$$

such that there is good phase space overlap. Then, we introduce a modification of the model, where

$$\Pr(|T_0 - c| < \delta) = 0 \quad \text{for} \quad c = 1.75, \quad \delta = 0.25. \quad (34)$$

Apart from this modification, the probability density of T_0 is proportional to a Gaussian distribution with the stated parameters in Eq. (32). This creates a gap in phase space that necessitates interpolation.

The performance of marginal and conditional reweighting for interpolation is shown in Fig. 4, again with the same layout as Fig. 2. Similarly to extrapolation, marginal reweighting at truth level is very effective away from the gap in phase space. Since $p_{(\mathrm{I})}(T) = 0$ in the gap, however, it is impossible for marginal reweighting to match the target distribution, for which the probability density is non-zero. This carries over to detector-level, where marginal reweighting suffers near $R \sim 1$. By contrast, conditional reweighting is effective across the entire domain, albeit with worse precision than marginal reweighting far from the phase space gap.

IV. JET ENERGY RESPONSE

We now present a physics case study where we expect conditional reweighting to be effective: simulation of the jet energy response at the LHC. To highlight the performance of conditional reweighting for interpolation, we will artificially construct a large phase space gap. Since we do not have a full target distribution to compare with the reweighted distributions, we use marginal reweighting on a sample without the phase space gap as a proxy. Despite these limitations, we hope this example highlights the complementarity of marginal and conditional reweighting.

A. Simulated Dijet Data Sets

This study is based on generic dijet production in quantum chromodynamics. As the "coarse" particle-level generator for data set (I), we use Pythia 6.426 [29] with the Z2 tune [49]. The "precise" particle-level generator for data set (III) is Pythia 8.219 [50]. Different from the study in Sec. III, we also use the "precise" Pythia 8.219 for data set (II), though as described below, we impose a phase space restriction such that data sets (I) and (II) have similar phase space coverage. Note that conditional reweighting does not require data sets (I) and (II) to have identical generators, though they should be as similar as possible to avoid unnecessarily large weights.

The fast detector simulation for data sets (II) and (III) is Delphes 3.4.1 [35–37] using the default CMS detector card. The full detector response for data set (I) uses a Geant4-based [24–26] full simulation of the CMS experiment [51]. More specifically, data set (I) comes from the CMS Open Data Portal [52–54] and processed into an MIT Open Data format [55–58]. Data sets (II) and (III) were generated for this study and are available at Ref [59].

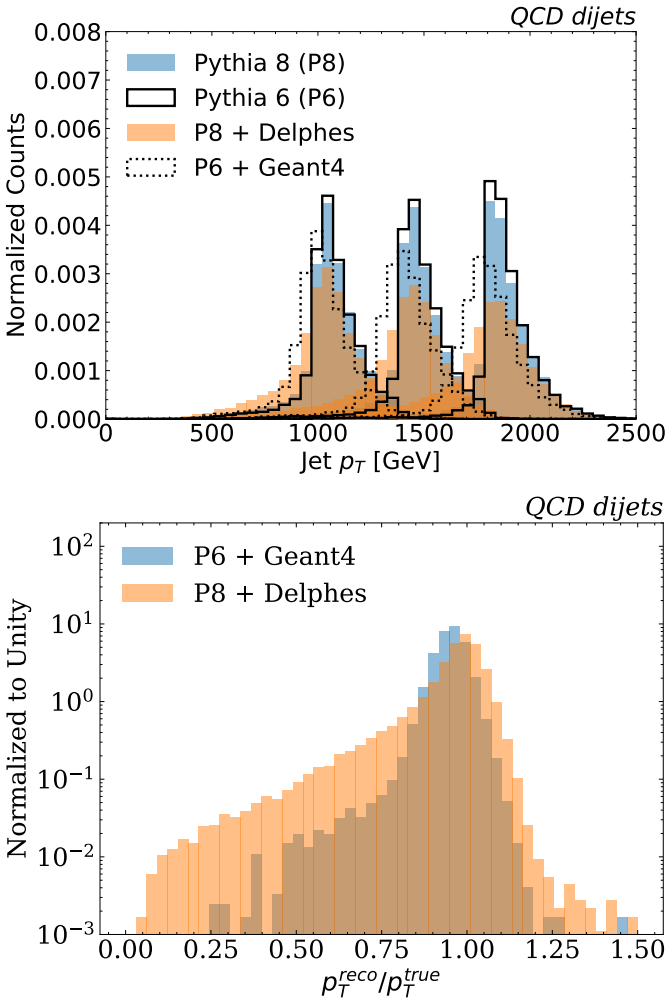

FIG. 5. Jet kinematics and reconstruction for the QCD example. Top: Histograms of particle- and detector-level jet transverse momenta (p_T). Bottom: Comparing the detector response for the Geant4 and Delphes event samples. The generation sample for data set (III) is represented by the Pythia 8 (P8) histogram while the corresponding simulation sample is represented by the P8+Delphes histogram. The samples for data sets (I) and (II) are missing the middle peak in the top plot.

For each data set, we have access to the parton-level hard-scattering scale $\hat{p}_T$ in Pythia, which is in general different from the jet-level transverse momentum p_T we are interested in studying. As is typical for the generation of steeply falling spectra, the full dijet data sets are constructed as a series of separate samples, each with a different range of $\hat{p}_T$. To avoid any issues related to the trigger, we focus on data sets where $\hat{p}_T > 1$ TeV. For this study, we consider three $\hat{p}_T$ ranges:

$$\hat{p}_T \in [1, 1.4] \text{ TeV}, \quad \hat{p}_T \in [1.4, 1.8] \text{ TeV}, \quad \hat{p}_T > 1.8 \text{ TeV}. \quad (35)$$

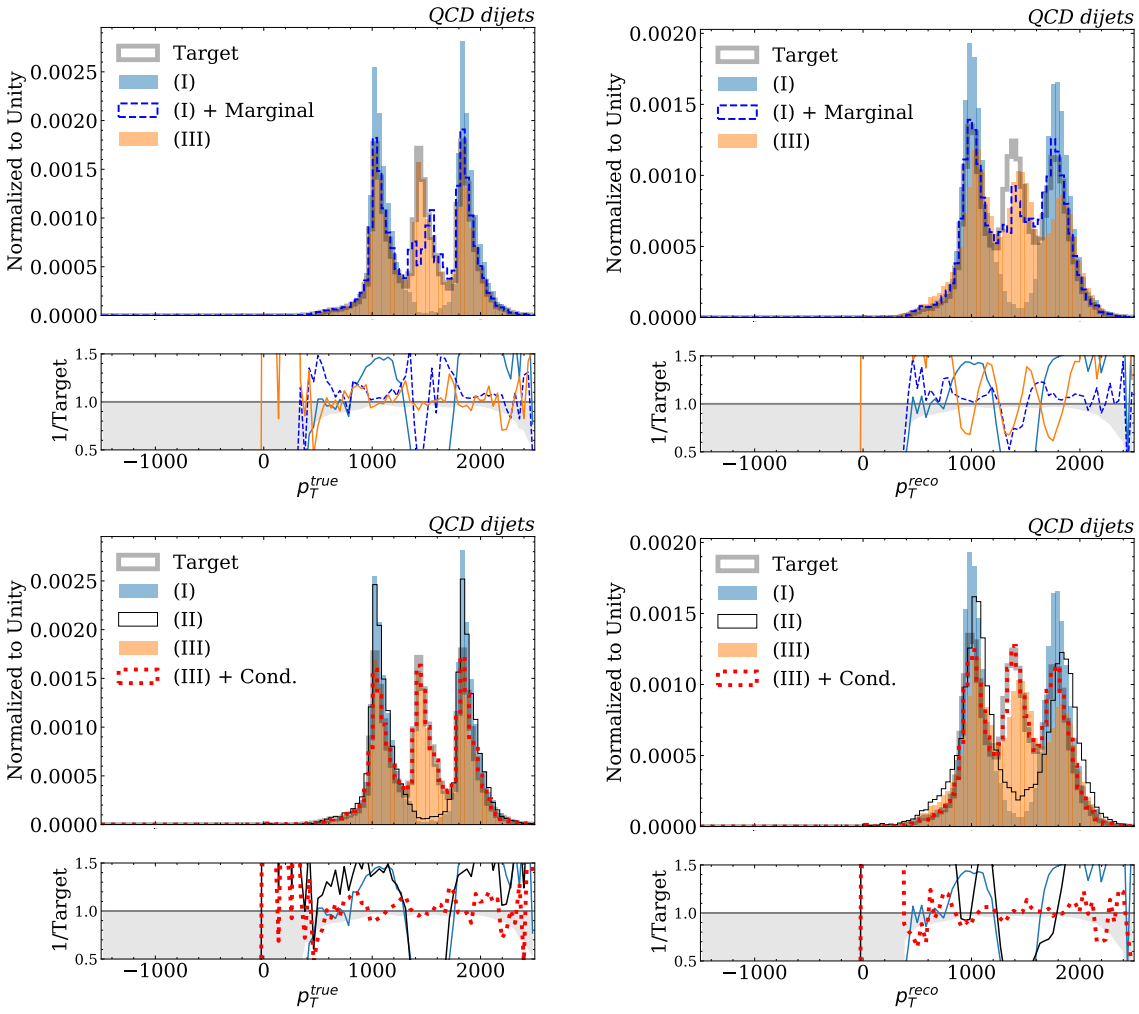

FIG. 6. Comparison of marginal reweighting (top row) and conditional reweighting (bottom row) in a QCD jet example. Shown are histograms of the true particle-level p_T (left column) and reconstructed detector-level p_T (right column). Distribution (I) involves Pythia 6 (with an artificial phase space gap at $\hat{p}_T \in [1.4, 1.8]$ TeV) interfaced with Geant4. Distribution (II) involves Pythia 8 (with the same phase space gap) interfaced with Delphes. Distribution (III) involves Pythia 8 interfaced with Delphes with no phase space gap. To match the target distribution (Pythia 8 with no phase space gap interfaced with Geant4), one can either marginally reweight distribution (I) or conditionally reweight distribution (III). Like the example in Fig. 4, marginal reweighting cannot bridge the phase space gap, whereas conditional reweighting yields a sensible interpolation.

Particles (at truth level) or particle flow candidates (at reconstructed level) are used as inputs to jet clustering, implemented using FastJet 3.2.1 [60, 61] and the anti-k_t algorithm [62] with radius parameter $R = 0.5$. The corresponding jet p_T spectra are shown in Fig. 5. When comparing to experimental data, a relative normalization would be applied to scale down the higher $\hat{p}_T$ slices, but we have elided those factors in this study to highlight the behavior of reweighting.

B. Results with Interpolation

To create a phase space gap and demonstrate the ability of conditional reweighting to interpolate, we remove the $\hat{p}_T \in [1.4, 1.8]$ TeV phase space slice from data sets (I) and (II). This effectively makes them both "coarse" generators, relative to the "precise" generator for data set (III) that covers the full phase space. Specifically, our three event samples are

- (I): Pythia 6 ⇒ Geant4 for $\hat{p}_T \in [1, 1.4]$ TeV and $\hat{p}_T > 1.8$ TeV;
- (II): Pythia 8 ⇒ Delphes for $\hat{p}_T \in [1, 1.4]$ TeV and $\hat{p}_T > 1.8$ TeV;
- (III): Pythia 8 ⇒ Delphes for $\hat{p}_T > 1$ TeV.

Each p_T slice within each sample has 10^4 jets.[5] We do not have a simulation of Pythia 8 with Geant4, so unlike the Gaussian case, we cannot display the exact target distribution. Instead, we use marginal reweighting on a bigger data set (up to 10^5 events per p_T slice) without the phase space gap to construct a synthetic target at detector level.

While we have artificially removed a $\hat{p}_T$ slice for this interpolation study, there are realistic contexts where this could happen. For example, legacy data corresponding to that $\hat{p}_T$ slice could be missing or corrupted, or that slice may have never been simulated (or simulated with reduced statistics) to save computing power. In the case of new physics searches, full simulation data sets may only be available at benchmark parameter values.

The results of marginal and conditional reweighting for the jet energy response are shown in Fig. 6. We use the identical neural network setup from Sec. III, and we see the same qualitative features as for the Gaussian interpolation example in Sec. III C. Conditional reweighting correctly has no effect at particle level and yields a smooth distribution at detector level. By contrast, marginal reweighting suffers near the phase space gap at 1.5 TeV, similarly to the interpolating Gaussian case. In addition to better matching the target distribution, conditional reweighting yields weights that are closer to unity.

V. CONCLUSIONS

In this paper, we extended the technique of neural network-based reweighting to the conditional case, where some features x are reweighted conditioned on other features x'. In regions of phase space that are well covered by the input and target probability densities, conditional reweighting is unlikely to outperform marginal reweighting. In phase space regions where the input probability density is small compared to the target probability density, though, conditional reweighting can yield improved behavior by leveraging the ability of neural networks to interpolate and extrapolate. This is relevant for constructing simulated data sets for the LHC, where full simulation may be too computationally costly to cover the full phase space, while fast simulation can be used to fill in the gaps.

An interesting feature of our approach to neural conditional reweighting is that we can derive the reweighting function in Eq. (12) through a single training procedure, instead of the naive two-step procedure suggested by Eq. (9). The key is to train on a higher dimensional phase space and then take an appropriate limit, which may be relevant for other machine learning applications. In practice, different approaches to neural conditional reweighting yield similar performance, as shown in App. A, but we prefer the single training procedure for its computational simplicity and conceptual elegance. Though we only showed one-dimensional examples in this paper, there are no conceptual barriers to handling multi-dimensional or variable-dimensional situations, which we plan to explore in future work.

An implicit assumption of our approach is that the neural network is well trained. This is required for all reweighting methods to work, since the relationship in Eq. (6) is only guaranteed in the asymptotic limit. A full quantitative comparison of different neural reweighting methods will need to assess systematic uncertainties, for example by analyzing the results for multiple trainings, performing closure tests on known targets, or comparing the results to low-dimensional binned methods. Eventually, one might want to use these reweighting uncertainties to guide the process of full simulation, where one prioritizes simulating regions of phase space that cannot be well modeled by (conditional) reweighting alone.

The main advantage of conditional reweighting is in cases where the input probability density is too small relative to the target. This is a generic challenge, not only for reweighting methods but for any generative modeling task where there is insufficient training data. Carefully constructed combinations of reweightings may be able to provide a partial solution to this problem, as could imposing smoothness requirements in the loss function to regularize how the neural network interpolates and extrapolates. Further hybrid methods that involve moving features instead of simply reweighting them (as in optimal transport problems [63–67]) may further extend the utility of these methods across high-energy physics and beyond.

CODE AND DATA

The code for this paper can be found at https://github.com/hep-lbdl/neuralconditional, which makes use of Jupyter notebooks [68] employing NumPy [69] for data manipulation and Matplotlib [70] to produce figures. All of the machine learning was performed on an Nvidia RTX6000 Graphical Processing Unit (GPU) and reproducing the entire notebook takes less than five minutes. The physics data sets are hosted on Zenodo at Ref. [56–59].

ACKNOWLEDGMENTS

BN is supported by the U.S. Department of Energy (DOE), Office of Science under contract DE-AC02-05CH11231. JT is supported by the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://iaifi.org/), and by the U.S. DOE Office of High Energy Physics under grant number DE-SC0012567.

[5] The original data sets have many more events, but a relatively small fraction is used here to ensure that the target is more accurate than the reweighted test cases and to amplify the impact of the phase space gap.

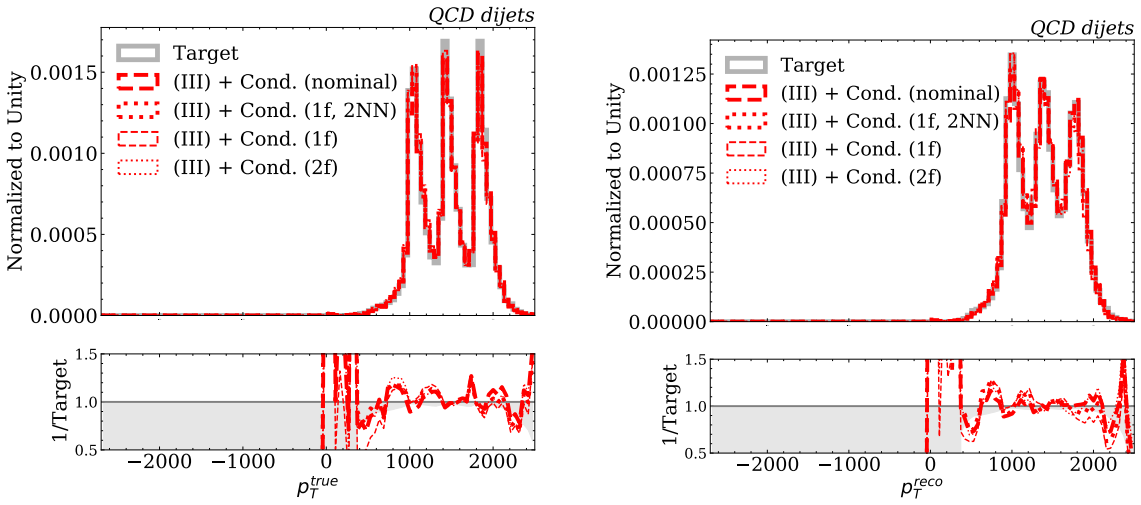

FIG. 7. Alternative neural conditional reweighting schemes for the dijet example in Fig. 6.

Appendix A: Alternative Neural Conditional Reweighting Schemes

In this appendix, we present results for three alternative neural conditional reweighting schemes to explore potential variations. The methods we compare are:

- (nominal): The nominal scheme, shown in the body of this paper, uses a single learned function built from Eqs. (17), (18), and (19).
- (1f, 2NN): This is a slightly more flexible version of the nominal setup, which still uses a single function built from Eq. (17), but f_0 and f_1 are now two independent neural networks with the same architecture as the marginal reweighting network.
- (1f): This is an even more flexible setup using the loss in Eq. (13), where we train a single neural network with three inputs without any constraints on its functional form.
- (2f): This is the two function setup from Eq. (9) that uses one joint reweighting and one marginal reweighting.

The results are shown in Figs. 7 and 8. All approaches work well, though the nominal approach does a somewhat better job tracking the target distribution.

[1] B. Nachman and J. Thaler, Neural resampler for Monte Carlo reweighting with preserved uncertainties, Phys. Rev. D **102**, 076004 (2020), arXiv:2007.11586 [hep-ph].
[2] K. Cranmer, J. Brehmer, and G. Louppe, The frontier of simulation-based inference, Proc. Nat. Acad. Sci. **117**, arXiv:1911.01429 (2019), arXiv:1911.01429 [stat.ML].
[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics (Springer New York Inc., New York, NY, USA, 2001).
[4] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density Ratio Estimation in Machine Learning* (Cambridge University Press, 2012).
[5] K. Cranmer, J. Pavez, and G. Louppe, Approximating Likelihood Ratios with Calibrated Discriminative Classifiers, (2015), arXiv:1506.02169 [stat.AP].
[6] J. Brehmer, K. Cranmer, G. Louppe, and J. Pavez, Constraining Effective Field Theories with Machine Learning, Phys. Rev. Lett. **121**, 111801 (2018), arXiv:1805.00013 [hep-ph].
[7] J. Brehmer, K. Cranmer, G. Louppe, and J. Pavez, A Guide to Constraining Effective Field Theories with Machine Learning, Phys. Rev. D **98**, 052004 (2018), arXiv:1805.00020 [hep-ph].
[8] J. Brehmer, F. Kling, I. Espejo, and K. Cranmer, MadMiner: Machine learning-based inference for particle physics, Comput. Softw. Big Sci. **4**, 3 (2020), arXiv:1907.10621 [hep-ph].
[9] J. Brehmer, G. Louppe, J. Pavez, and K. Cranmer, Mining gold from implicit models to improve likelihood-free inference, Proc. Nat. Acad. Sci. , 201915980 (2020), arXiv:1805.12244 [stat.ML].
[10] M. Stoye, J. Brehmer, G. Louppe, J. Pavez, and K. Cranmer, Likelihood-free inference with an improved cross-entropy estimator, (2018), arXiv:1808.00973 [stat.ML].
[11] A. Andreassen and B. Nachman, Neural Networks for Full Phase-space Reweighting and Parameter Tuning, Phys. Rev. D **101**, 091901(R) (2020), arXiv:1907.08209 [hep-ph].
[12] J. Hollingsworth and D. Whiteson, Resonance Searches with Machine Learned Likelihood Ratios, (2020), arXiv:2002.04699 [hep-ph].
[13] A. Andreassen, S. Hsu, B. Nachman, N. Suaysom, A. Suresh, Parameter Estimation using Neural Networks in the Presence of Detector Effects, Phys. Rev. D **103**, 036001 (2021), arXiv:2010.03569 [hep-ph].

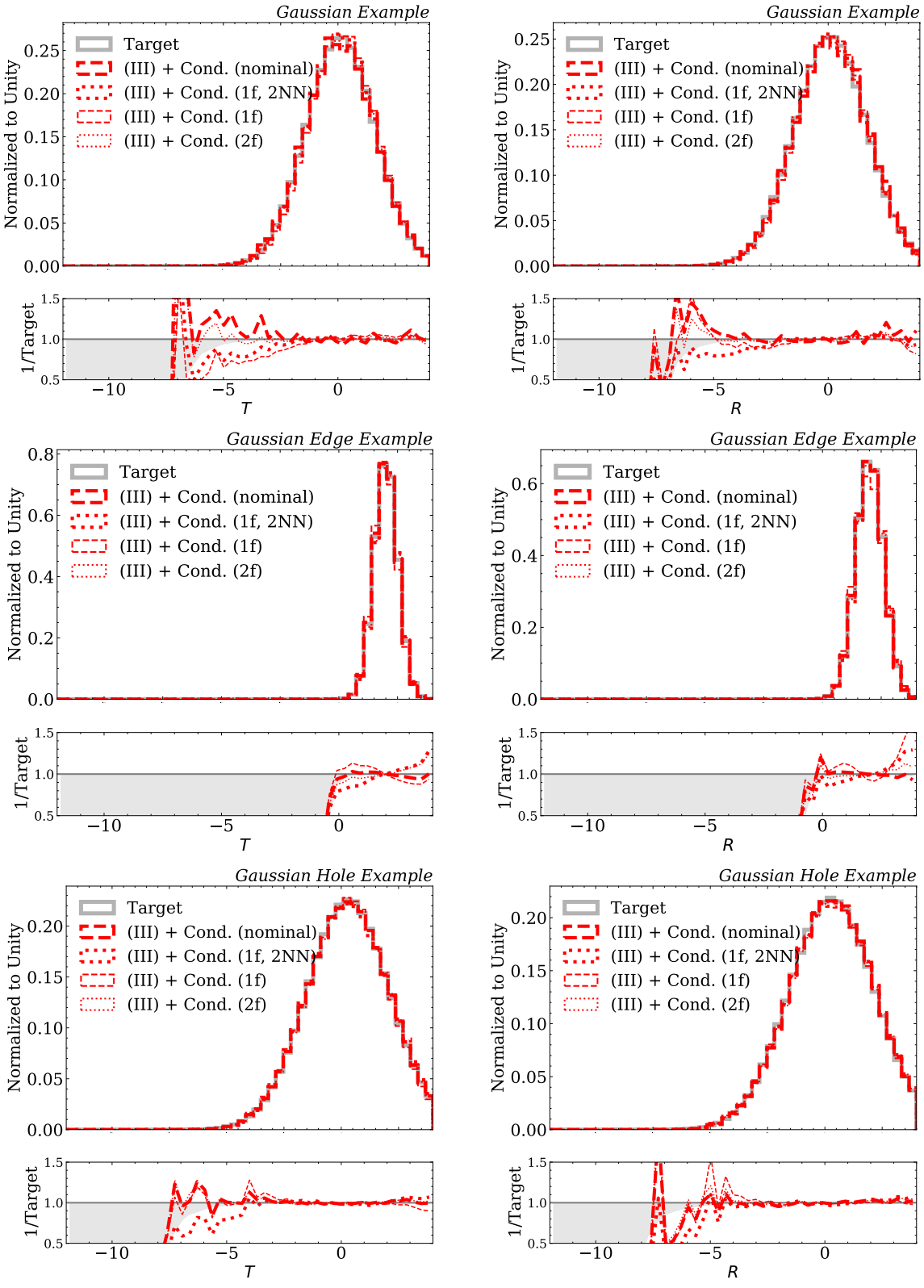

FIG. 8. Alternative neural conditional reweighting methods for the plain Gaussian example in Fig. 2 (top row), for the extrapolation Gaussian example in Fig. 3 (middle row), and for the interpolation Gaussian example in Fig. 4 (bottom row).

[14] A. Andreassen, B. Nachman, and D. Shih, Simulation Assisted Likelihood-free Anomaly Detection, Phys. Rev. D **101**, 095004 (2020), arXiv:2001.05001 [hep-ph].

[15] C. Badiali, F. Di Bello, G. Frattari, E. Gross, V. Ippolito, M. Kado, and J. Shlomi, Efficiency Parameterization with Neural Networks, Comput. Softw. Big Sci. **5**, 14 (2021), arXiv:2004.02665 [hep-ex].

[16] M. Bunse, N. Piatkowski, T. Ruhe, W. Rhode, and K. Morik, Unification of deconvolution algorithms for Cherenkov astronomy, in *5th International Conference on Data Science and Advanced Analytics (DSAA)* (IEEE, 2018) pp. 21–30.

[17] T. Ruhe, T. Voigt, M. Wornowizki, M. Börner, W. Rhode, and K. Morik, Mining for spectra - the dortmund spectrum estimation algorithm (2019).

[18] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, and J. Thaler, OmniFold: A Method to Simultaneously Unfold All Observables, Phys. Rev. Lett. **124**, 182001 (2020), arXiv:1911.09107 [hep-ph].

[19] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, A. Suresh, and J. Thaler, Scaffolding Simulations with Deep Learning for High-dimensional Deconvolution, in *9th International Conference on Learning Representations* (2021) arXiv:2105.04448 [stat.ML].

[20] S. Alioli, P. Nason, C. Oleari, and E. Re, A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX, JHEP **06**, 043, arXiv:1002.2581 [hep-ph].

[21] P. Nason, A New method for combining NLO QCD with shower Monte Carlo algorithms, JHEP **11**, 040, arXiv:hep-ph/0409146.

[22] S. Frixione, P. Nason, and C. Oleari, Matching NLO QCD computations with Parton Shower simulations: the POWHEG method, JHEP **11**, 070, arXiv:0709.2092 [hep-ph].

[23] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, JHEP **07**, 079, arXiv:1405.0301 [hep-ph].

[24] S. Agostinelli *et al.* (GEANT4), GEANT4–a simulation toolkit, Nucl. Instrum. Meth. A **506**, 250 (2003).

[25] J. Allison *et al.*, Geant4 developments and applications, IEEE Transactions on Nuclear Science **53**, 270 (2006).

[26] J. Allison *et al.*, Recent developments in Geant4, Nucl. Instrum. Meth. A **835**, 186 (2016).

[27] G. Battistoni *et al.*, Overview of the FLUKA code, Annals Nucl. Energy **82**, 10 (2015).

[28] T. Böhlen, F. Cerutti, M. Chin, A. Fassò, A. Ferrari, P. Ortega, A. Mairani, P. Sala, G. Smirnov, and V. Vlachoudis, The fluka code: Developments and challenges for high energy and medical applications, Nuclear Data Sheets **120**, 211 (2014).

[29] T. Sjöstrand, S. Mrenna, and P. Z. Skands, PYTHIA 6.4 Physics and Manual, JHEP **05**, 026, arXiv:hep-ph/0603175 [hep-ph].

[30] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, An Introduction to PYTHIA 8.2, Comput. Phys. Commun. **191**, 159 (2015), arXiv:1410.3012 [hep-ph].

[31] M. Bahr *et al.*, Herwig++ Physics and Manual, Eur. Phys. J. C **58**, 639 (2008), arXiv:0803.0883 [hep-ph].

[32] J. Bellm *et al.*, Herwig 7.0/Herwig++ 3.0 release note, Eur. Phys. J. C **76**, 196 (2016), arXiv:1512.01178 [hep-ph].

[33] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, and J. Winter, Event generation with SHERPA 1.1, JHEP **02**, 007, arXiv:0811.4622 [hep-ph].

[34] E. Bothmann *et al.* (Sherpa), Event Generation with Sherpa 2.2, SciPost Phys. **7**, 034 (2019), arXiv:1905.09127 [hep-ph].

[35] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (DELPHES 3), DELPHES 3, A modular framework for fast simulation of a generic collider experiment, JHEP **02**, 057, arXiv:1307.6346 [hep-ex].

[36] A. Mertens, New features in Delphes 3, *Proceedings, 16th International workshop on Advanced Computing and Analysis Techniques in physics (ACAT 14): Prague, Czech Republic, September 1-5, 2014*, J. Phys. Conf. Ser. **608**, 012045 (2015).

[37] M. Selvaggi, DELPHES 3: A modular framework for fast-simulation of generic collider experiments, *Proceedings, 15th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2013): Beijing, China, May 16-21, 2013*, J. Phys. Conf. Ser. **523**, 012033 (2014).

[38] A. Butter and T. Plehn, Generative Networks for LHC events, (2020), arXiv:2008.08558 [hep-ph].

[39] Y. Alanazi, N. Sato, P. Ambrozewicz, A. N. H. Blin, W. Melnitchouk, M. Battaglieri, T. Liu, and Y. Li, A survey of machine learning-based physics event generation, (2021), arXiv:2106.00643 [hep-ph].

[40] R. T. D'Agnolo and A. Wulzer, Learning New Physics from a Machine, Phys. Rev. D **99**, 015014 (2019), arXiv:1806.02350 [hep-ph].

[41] B. Nachman and J. Thaler, E Pluribus Unum Ex Machina: Learning from Many Collider Events at Once, Phys. Rev. D **103**, 116013 (2020), arXiv:2101.07263 [physics.data-an].

[42] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, Mutual information neural estimation, in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause (PMLR, 2018) pp. 531–540.

[43] D. Kim, K. Kong, K. T. Matchev, M. Park, and P. Shyamsundar, Deep-Learned Event Variables for Collider Phenomenology, (2021), arXiv:2105.10126 [hep-ph].

[44] B. K. Miller, A. Cole, P. Forré, G. Louppe, and C. Weniger, Truncated Marginal Neural Ratio Estimation 10.5281/zenodo.5043707 (2021), arXiv:2107.01214 [stat.ML].

[45] F. Chollet, Keras, `https://github.com/fchollet/keras` (2017).

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, Tensorflow: A system for large-scale machine learning., in *OSDI*, Vol. 16 (2016) pp. 265–283.

[47] D. Kingma and J. Ba, Adam: A method for stochastic optimization, (2014), arXiv:1412.6980 [cs].

[48] D. Reichelt, P. Richardson, and A. Siodmok, Improving the Simulation of Quark and Gluon Jets with Herwig 7, Eur. Phys. J. C **77**, 876 (2017), arXiv:1708.01491 [hep-ph].

[49] S. Chatrchyan *et al.* (CMS), Measurement of the Underlying Event Activity at the LHC with $\sqrt{s} = 7$ TeV and Comparison with $\sqrt{s} = 0.9$ TeV, JHEP **09**, 109, arXiv:1107.0330 [hep-ex].

[50] T. Sjöstrand, S. Mrenna, and P. Z. Skands, A Brief Intro-

duction to PYTHIA 8.1, Comput. Phys. Commun. **178**, 852 (2008), arXiv:0710.3820 [hep-ph].
[51] S. Chatrchyan *et al.* (CMS), The CMS Experiment at the CERN LHC, JINST **3**, S08004.
[52] CMS Collaboration, Simulated dataset QCD_Pt-1000to1400_TuneZ2_7TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive), CERN Open Data Portal 10.7483/OPENDATA.CMS.96U2.3YAH (2016).
[53] CMS Collaboration, Simulated dataset QCD_Pt-1400to1800_TuneZ2_7TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive), CERN Open Data Portal 10.7483/OPENDATA.CMS.RC9V.B5KX (2016).
[54] CMS Collaboration, Simulated dataset QCD_Pt-1800_TuneZ2_7TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive), CERN Open Data Portal 10.7483/OPENDATA.CMS.CX2X.J3KW (2016).
[55] P. T. Komiske, R. Mastandrea, E. M. Metodiev, P. Naik, and J. Thaler, Exploring the Space of Jets with CMS Open Data, Phys. Rev. D **101**, 034009 (2020), arXiv:1908.08542 [hep-ph].
[56] P. Komiske, R. Mastandrea, E. Metodiev, P. Naik, and J. Thaler, CMS 2011A Simulation | Pythia 6 QCD 1000-1400 | pT > 375 GeV | MOD HDF5 Format, 10.5281/zenodo.3341502 (2019).
[57] P. Komiske, R. Mastandrea, E. Metodiev, P. Naik, and J. Thaler, CMS 2011A Simulation | Pythia 6 QCD 1400-1800 | pT > 375 GeV | MOD HDF5 Format, 10.5281/zenodo.3341770 (2019).
[58] P. Komiske, R. Mastandrea, E. Metodiev, P. Naik, and J. Thaler, CMS 2011A Simulation | Pythia 6 QCD1800-inf | pT > 375 GeV | MOD HDF5 Format, 10.5281/zenodo.3341772 (2019).
[59] B. Nachman and J. Thaler, Delphes dijet dataset, 10.5281/zenodo.5108967 (2021).
[60] M. Cacciari, G. P. Salam, and G. Soyez, FastJet User Manual, Eur. Phys. J. **C72**, 1896 (2012), arXiv:1111.6097 [hep-ph].
[61] M. Cacciari and G. P. Salam, Dispelling the N^3 myth for the k_t jet-finder, Phys. Lett. **B641**, 57 (2006), arXiv:hep-ph/0512210 [hep-ph].
[62] M. Cacciari, G. P. Salam, and G. Soyez, The anti-k_t jet clustering algorithm, JHEP **04**, 063, arXiv:0802.1189 [hep-ph].
[63] P. T. Komiske, E. M. Metodiev, and J. Thaler, Metric Space of Collider Events, Phys. Rev. Lett. **123**, 041801 (2019), arXiv:1902.02346 [hep-ph].
[64] T. Cai, J. Cheng, N. Craig, and K. Craig, Linearized optimal transport for collider events, Phys. Rev. D **102**, 116019 (2020), arXiv:2008.08604 [hep-ph].
[65] M. Crispim Romão, N. F. Castro, J. G. Milhano, R. Pedro, and T. Vale, Use of a generalized energy Mover's distance in the search for rare phenomena at colliders, Eur. Phys. J. C **81**, 192 (2021), arXiv:2004.09360 [hep-ph].
[66] C. Cesarotti and J. Thaler, A Robust Measure of Event Isotropy at Colliders, JHEP **08**, 084, arXiv:2004.06125 [hep-ph].
[67] C. Cesarotti, M. Reece, and M. J. Strassler, The Efficacy of Event Isotropy as an Event Shape Observable, (2020), arXiv:2011.06599 [hep-ph].
[68] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, Jupyter notebooks – a publishing format for reproducible computational workflows, in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt (IOS Press, 2016) pp. 87 – 90.
[69] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Array programming with NumPy, Nature **585**, 357 (2020).
[70] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering **9**, 90 (2007).