Unbinned profiled unfolding

Jay Chan[1,2,*] and Benjamin Nachman[2,3,†]

[1]*Department of Physics, University of Wisconsin-Madison, Madison, Wisconsin 53706, USA*
[2]*Physics Division, Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA*
[3]*Berkeley Institute for Data Science, University of California, Berkeley, California 94720, USA*

(Received 17 February 2023; accepted 12 June 2023; published 7 July 2023)

Unfolding is an important procedure in particle physics experiments that corrects for detector effects and provides differential cross section measurements that can be used for a number of downstream tasks, such as extracting fundamental physics parameters. Traditionally, unfolding is done by discretizing the target phase space into a finite number of bins and is limited in the number of unfolded variables. Recently, there have been a number of proposals to perform unbinned unfolding with machine learning. However, none of these methods (like most unfolding methods) allow for simultaneously constraining (profiling) nuisance parameters. We propose a new machine learning-based unfolding method that results in an unbinned differential cross section and can profile nuisance parameters. The machine learning loss function is the full likelihood function, based on binned inputs at detector level. We first demonstrate the method with simple Gaussian examples and then show the impact on a simulated Higgs boson cross section measurement.

DOI: 10.1103/PhysRevD.108.016002

I. INTRODUCTION

One of the most common analysis goals in particle and nuclear physics is the measurement of differential cross sections. These quantities encode the rate at which a particular process occurs as a function of certain observables of interest. From measured cross sections, a number of downstream inference tasks can be performed, including the estimation of fundamental parameters, tuning simulations, and searching for physics beyond the Standard Model. The key challenge of cross section measurements is correcting the data for detector distortions, a process called deconvolution or unfolding. See Refs. [1–4] for recent reviews on unfolding and Refs. [5–7] for the most widely used unfolding algorithms.

Until recently, all cross section measurements were performed with histograms. In particular, the target spectra and experimental observations were binned and the unfolding problem is recast in the language of linear algebra. That is, one would like to determine the signal strength, defined as the ratio of the observed signal yield to the theoretical prediction, for each bin based on the measurements from experimental observations. This approach comes with the limitation that the binning must be determined beforehand. This makes it difficult to compare measurements with different binning. Furthermore, the optimal binning depends on the downstream inference task.

Modern machine learning (ML) has enabled the creation of unfolding methods that can process unbinned data [8]. Deep generative models such as generative adversarial networks [9–11] and variational autoencoders [12,13] produce implicit models that represents the probability density of the unfolded result and allow to sample from the probability density. Methods based on normalizing flows [14–17] allow for both sampling and density estimation. In contrast, the classifier-based method OmniFold [18,19] iteratively reweights a simulated dataset. A summary of machine learning-based unfolding methods can be found in Ref. [8] and recent applications of these techniques (in particular, of OmniFold) to experimental data are presented in Refs. [20–23]. While powerful, none of these approaches can simultaneously estimate cross sections and fit (nuisance) parameters. This can be a significant shortcoming when the phase space region being probed has nontrivial constraining power for systematic uncertainties.

Unfolding methods that can also profile have been proposed. One possibility is to treat the cross section in each region of particle-level phase space (i.e., in a histogram bin) as a free parameter and then perform a likelihood fit as for any set of parameters of interest and nuisance parameters. For example, this is the setup of the simplified template cross section (e.g., Refs. [24–27]) measurements for Higgs boson kinematic properties. Another possibility is fully Bayesian unfolding [28], which samples from the

[*]cchan62@wisc.edu
[†]bpnachman@lbl.gov

posterior probability over the cross section in each bin of the particle-level phase space and over the nuisance parameters. All of these methods require binning.

In this paper, we propose a new machine learning-based unfolding method that is both unbinned at particle level and can profile, referred to as unbinned profiled unfolding (UPU). UPU reuses all the standard techniques used in binned maximum likelihood unfolding and combines them with ML methods that allow for unbinned unfolding. Specifically, we use the binned maximum likelihood at detector level as the metric to optimize the unfolding, while the unfolding takes unbinned particle-level simulations as inputs.

The rest of this paper is organized as follows. In Sec. II, we describe the procedure and implementation details of UPU. We then present simple Gaussian examples to demonstrate the usage of UPU in Sec. III. In Sec. IV, we apply UPU to a simulated Higgs boson cross section measurement at the Large Hadron Collider. The conclusions and outlook are then given in Sec. V.

II. UNBINNED PROFILED UNFOLDING

A. Statistical setup

UPU generalizes binned maximum likelihood unfolding to the unbinned case. Binned maximum likelihood unfolding can be described by the following optimization setup:

$$(\hat{k}, \hat{\theta}) = \mathrm{argmax}_{(k,\theta)} \Pr(m|k,\theta) p_0(\theta), \tag{1}$$

where $m \in \mathbb{R}^{N_m}$ is a vector representing the counts in each of the N_m bins at detector level, $k \in \mathbb{R}^{N_k}$ is a vector representing the counts in each of the N_k bins at particle level (usually $N_m \geq N_k$), θ are the nuisance parameters, and p_0 is the prior on θ. Our idea is to keep the structure of Eq. (1) but replace k with an unbinned estimator of the particle-level spectrum. Suppose that the particle-level phase space is[1] $\mathbb{R}^N$ and let[2] $\tau[\omega] \in \mathbb{R}^{\mathbb{R}^N}$ parametrize the probability density over this space for parameters ω. The goal of UPU is then to optimize

$$(\hat{\omega}, \hat{\theta}) = \mathrm{argmax}_{(\omega,\theta)} \Pr(m|\tau[\omega],\theta) p_0(\theta), \tag{2}$$

where the final result would be given by $\tau[\hat{\omega}]$. The challenge with the construction in Eq. (2) is that for a given truth spectrum $\tau[\omega]$, we need to know the predicted detector-level distribution. In the binned case, this is readily computed by multiplying k by the response matrix $R_{ij} = \Pr(\text{measure in bin } i | \text{truth is bin } j)$. When the truth are unbinned, we need the full detector response. This is never known analytically and would be challenging to estimate numerically with a surrogate density estimator.[3] To address this challenge, we make use of the fact that simulated events come in pairs, with a matching between particle-level and detector-level events. Instead of estimating τ directly, we use a fixed simulation (with particle-level spectrum $\tau[\omega_0]$) and then learn a reweighting function $w_0[\lambda]$ to estimate the likelihood ratio between the unfolded result and the fixed simulation at particle level. Schematically:

$$(\hat{\lambda}, \hat{\theta}) = \mathrm{argmax}_{(\lambda,\theta)} \Pr(m|\tau[\omega_0] w_0[\lambda],\theta) p_0(\theta), \tag{3}$$

where in practice, we only have samples from $\tau[\omega_0]$ and w_0 is a surrogate model. The number of predicted events in a given bin i is then a sum over weights $w_0[\hat{\lambda}]$ (evaluated at particle level) for simulated events with a detector-level value in bin i. The probability over values m is then a product over Poisson probability mass functions, since the bins are statistically independent. The fact that the probability mass is known is crucial and means that UPU does not readily generalize the case where the detector-level phase space is also unbinned. This is the case for OmniFold, which also uses reweighting at particle level. In contrast to UPU, OmniFold uses an expectation-maximization-type algorithm to iteratively converge to the maximum likelihood estimate and does not currently allow for mixed implicit/explicit likelihood constraints (as is needed for nuisance parameters).

B. Machine learning approach

For particle-level features T and detector-level features R, the main goal is to train the likelihood ratio estimator $w_0(T)$, which reweights the simulated particle-level spectrum. In the absence of profiling, this corresponds to the following loss function:

$$L = \prod_{i=1}^{n_{\mathrm{bins}}} \Pr\left(n_i \middle| \sum_{j=1}^{n_{\mathrm{MC}}} w_0(T_j) \mathbb{I}_i(R_j)\right), \tag{4}$$

where n_i is the number of observed events in bin i, n_{MC} is the number of simulated events, and $\mathbb{I}_i(\cdot)$ is the indicator function that is one when $\cdot$ is in bin i and zero otherwise. When w_0 is parametrized as a neural network (see Sec. II C), then the logarithm of Eq. (4) is used for training:

$$\log L = \sum_{i=1}^{n_{\mathrm{bins}}} \left[n_i \log\left(\sum_{j=1}^{n_{\mathrm{MC}}} w_0(T_j) \mathbb{I}_i(R_j)\right) - \sum_{i=1}^{n_{\mathrm{MC}}} w_0(T_j) \mathbb{I}_i(R_j) \right], \tag{5}$$

[1]Assuming the space is suitably standardized to remove units.

[2]We will use $[\cdot]$ to denote the parameters of the function and $(\cdot)$ to denote the inputs of the function, e.g., $f[\theta](x)$ is a functional in θ and a function in x.

[3]Note that Eq. (2) is a probability distribution over probability distributions so building it from the per-event detector response is nontrivial.

TABLE I. Summary of Gaussian example datasets.

Data set	Parameters	Number of events	Purpose
$D_{\mathrm{sim}}^{1.0}$	$\mu = 0$, $\sigma = 1$ and $\epsilon = 1$	200,000	Nominal simulation
D_{obs}	$\mu = 0.8$, $\sigma = 1$ and $\epsilon = 1.2$	100,000	Observed data
D_{sim}^{*}	$\mu = 0$, $\sigma = 1$ and $\epsilon = (0.2, 1.8)$	200,000	Train w_1
$D_{\mathrm{sim}}^{1.2}$	$\mu = 0$, $\sigma = 1$ and $\epsilon = 1.2$	100,000	Validate w_1

where we have dropped constants that do not affect the optimization. Experimental nuisance parameters modify the predicted counts in a particular bin given the particle-level counts. We account for these effects with a second reweighting function:

$$w_1(R|T,\theta) = \frac{p_\theta(R|T)}{p_{\theta_0}(R|T)}, \tag{6}$$

where $p_\theta(R|T)$ is the conditional probability density of R given T with nuisance parameters θ. Importantly, w_1 does not modify the target particle level distribution. Incorporating w_1 into the log likelihood results in the full loss function:

$$\log L = \sum_{i=1}^{n_{\mathrm{bins}}} \left[n_i \log\left(\sum_{j=1}^{n_{\mathrm{MC}}} w_0(T_j) w_1(R_j|T_j,\theta) \mathbb{I}_i(R_j) \right) - \sum_{j=1}^{n_{\mathrm{MC}}} w_0(T_j) w_1(R_j|T_j,\theta) \mathbb{I}_i(R_j) \right] + \log p_0(\theta). \tag{7}$$

Since w_1 does not depend on the particle-level spectrum, it can be estimated prior to the final fit and only the parameters of w_0 and the value(s) of θ are allowed to float when optimizing Eq. (7).

C. Machine learning implementation

In our subsequent case studies, the reweighting functions w_0 and w_1 are parametrized with neural networks. The w_0 function is only constrained to be non-negative and so we choose it to be the exponential of a neural network.

The pretraining of w_1 requires neural conditional reweighting [29], as a likelihood ratio in R conditioned on T and parametrized in θ. While there are multiple ways of approximating conditional likelihood ratios, the one we found to be the most stable for the examples we have studied for UPU is the product approach:

$$w_1(R|T,\theta) = \left(\frac{p_\theta(R,T)}{p_{\theta_0}(R,T)}\right)\left(\frac{p_{\theta_0}(T)}{p_\theta(T)}\right), \tag{8}$$

where the two terms on the right-hand side are separately estimated and then their product is w_1. For a single feature T, a likelihood ratio between samples drawn from a probability density p and samples drawn from a

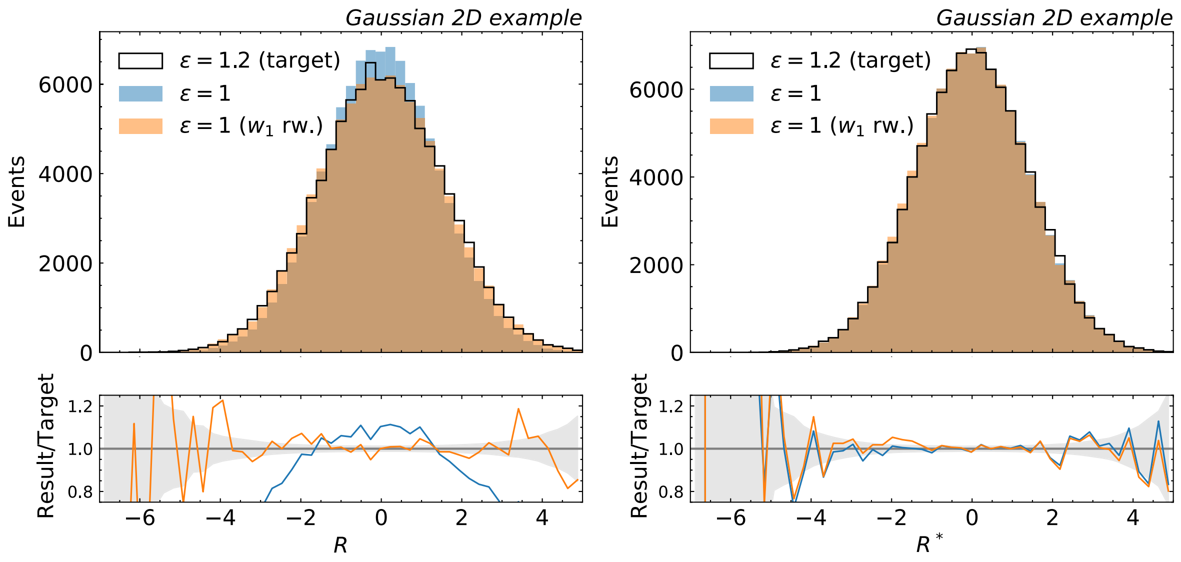

FIG. 1. Gaussian 2D example: the nominal detector-level spectra R (left) and R^* (right) with $\epsilon = 1$ reweighted by the trained w_1 conditioned at $\epsilon = 1.2$ and compared to the spectra with $\epsilon = 1.2$. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of $D_{\mathrm{sim}}^{1.2}$ events in a given bin.

probability density q is estimated using the fact that machine learning classifiers approximate monotonic transformations of likelihood ratios (see, e.g., Refs. [30,31]). In particular, we use the standard binary cross entropy loss function

$$L_{\mathrm{BCE}}[f] = -\sum_{Y\sim p}\log(f(Y)) - \sum_{Y\sim q}\log(1-f(Y)), \qquad (9)$$

and then the likelihood ratio is estimated as $f/(1-f)$. The last layer of the f networks are sigmoids in order to constrain their range to be between 0 and 1. The function f is additionally trained to be parametrized in θ by training with pairs (Y,Θ) instead of just Y, where Θ is a random variable corresponding to values θ sampled from a prior probability distribution. We will use a uniform prior when training the parametrized classifiers.

All neural networks are implemented using PyTorch [32] and optimized with Adam [33] with a learning rate of 0.001 and consist of three hidden layers with 50 nodes per layer. All intermediate layers use ReLU activation functions. Each network is trained for 10,000 epochs with early stopping using a patience of 10. The w_1 training uses a batch size of 100,000. The w_0 network is simultaneously

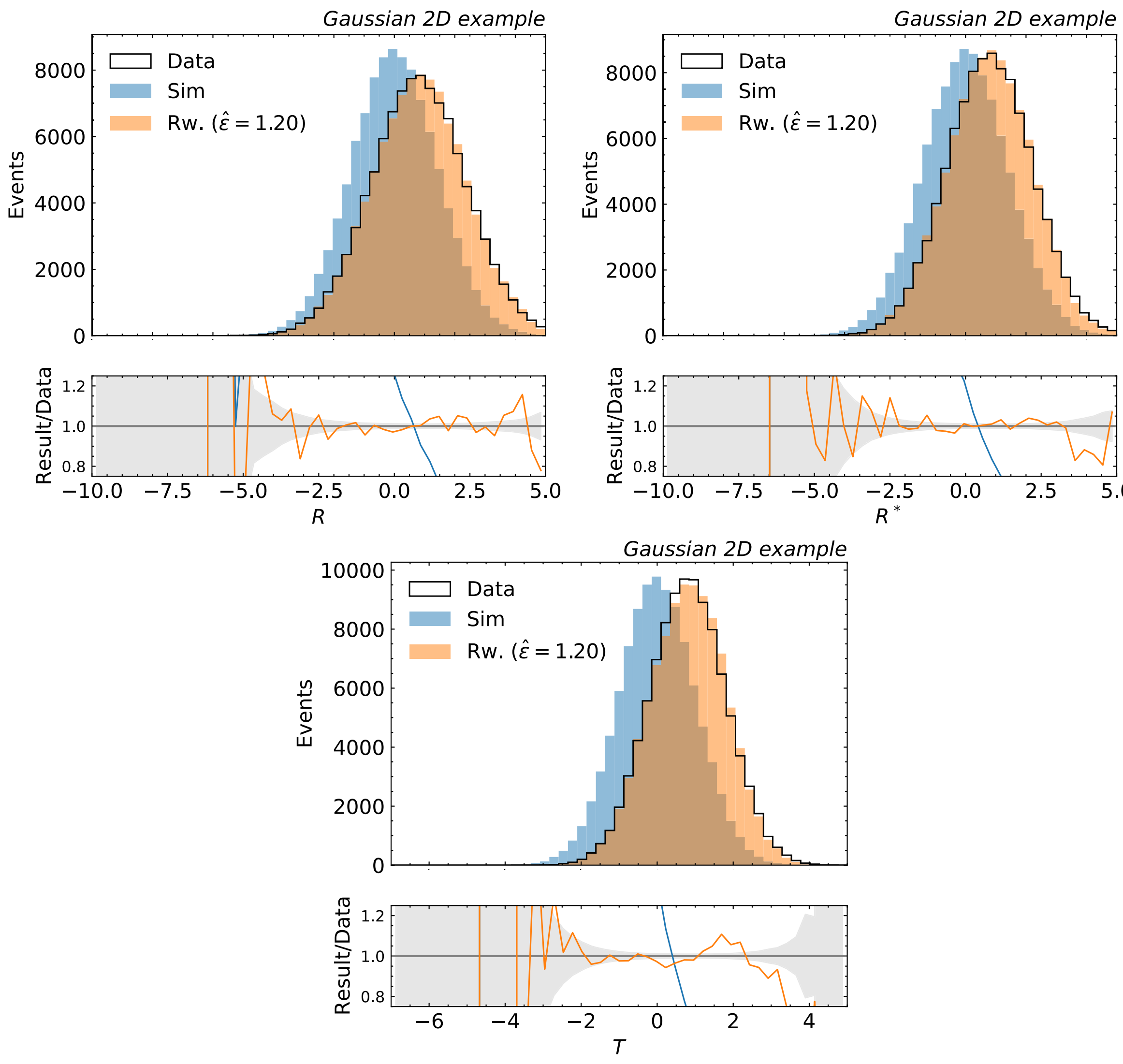

FIG. 2. Gaussian 2D example: results of the w_0 optimization. The nuisance parameter ϵ is optimized simultaneously with w_0 with the prior constraint set to 80%. The fitted ϵ is 1.20 ± 0.004. Top left: the detector-level spectrum R of the simulation template D_{sim} reweighted by the trained $w_0 \times w_1$, compared to the R spectrum of the observed data D_{obs}. Top right: the detector-level spectrum R' of the simulation template D_{sim} reweighted by the trained $w_0 \times w_1$, compared to the R^* spectrum of the observed data D_{obs}. Bottom: the particle-level spectrum T of the simulation template D_{sim} reweighted by the trained w_0, compared to the T spectrum of the observed data D_{obs}. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of observed events in a given bin.

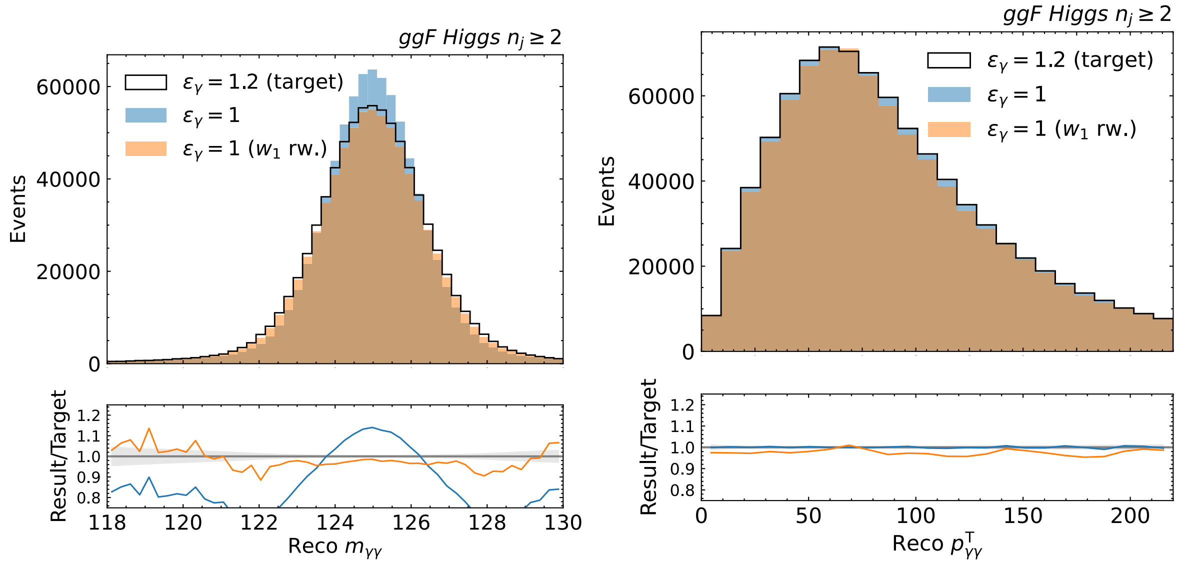

FIG. 3. Higgs boson cross section: the nominal detector-level spectra $m_{\gamma\gamma}$ (left) and $p_{\gamma\gamma}^T$ (right) with $\epsilon_\gamma = 1$ reweighted by the trained w_1 conditioned at $\epsilon_\gamma = 1.2$ and compared to the spectra with $\epsilon_\gamma = 1.2$. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of $D_{\text{sim}}^{1.2}$ events in a given bin.

optimized with θ and uses a batch size that is the full dataset, which corresponds to performing the fit in Eq. (7) over all the data. All neural networks are implemented using PyTorch [32] and optimized with Adam [33] with a learning rate of 0.001 and consist of three hidden layers with 50 nodes per layer. All intermediate layers use ReLU activation functions. Each network is trained for 10,000 epochs with early stopping using a patience of 10. The w_1 training uses a batch size of 100,000. The w_0 network is simultaneously optimized with θ and uses a batch size that is the full dataset, which corresponds to performing the fit in Eq. (7) over all the data.

III. GAUSSIAN EXAMPLE

We now demonstrate the proposed method with a simple numerical example. Here, each dataset represents a one-dimensional Gaussian distribution in the particle level and a two-dimensional distribution in the detector level. The particle-level Gaussian random variable T is described by mean μ and standard deviation σ, while the detector-level variables are given by

$$R = T + Z, \tag{10}$$

$$R^* = T + Z^*, \tag{11}$$

where Z (Z^*) is a Gaussian random variable with mean β (β^*) and standard deviation ϵ (ϵ^*). ϵ is considered to be the only nuisance parameter, and β, β^* are fixed to 0, and ϵ^* is fixed to 1. In this case, the nuisance parameter ϵ only has effect on the R spectrum and the R^* spectrum depends purely on the particle-level spectrum T. This setup is thus sensitive to both the effect of w_0 and that of w_1.[4]

Three datasets are prepared for the full training procedure. As summarized in Table I, the first dataset $D_{\text{sim}}^{1.0}$ is used as the nominal simulation sample, which contains 200,000 events with $\mu = 0$, $\sigma = 1$, and $\epsilon = 1$. The second data set D_{obs} is used as the observed data, which contains 100,000 events with $\mu = 0.8$, $\sigma = 1$, and $\epsilon = 1.2$. To train the w_1 reweighter, the third dataset D_{sim}^*, which contains 200,000 events with $\mu = 0$, $\sigma = 1$, and ϵ uniformly distributed from 0.2 to 1.8, is prepared and used as the simulation with systematic variations. In addition, another dataset $D_{\text{sim}}^{1.2}$ of 100,000 events with $\mu = 0$, $\sigma = 1$, and $\epsilon = 1.2$ is produced for validating the w_1 reweighter. All datasets used in the training procedure are split to 50% for training and 50% for validating.

A w_1 reweighter is trained to reweight $D_{\text{sim}}^{1.0}$ to D_{sim}^*. The trained w_1 is tested with the nominal R and R^* spectra ($D_{\text{sim}}^{1.0}$) reweighted to $\epsilon = 1.2$ and compared to the R and R^* spectra with $\epsilon = 1.2$. As shown in Fig. 1, the trained w_1 reweighter has learned to reweight the nominal R spectrum to match the R spectrum with ϵ at 1.2, and R^* is independent of the w_1 reweighter.

Based on the trained w_1 reweighter, a w_0 reweighter and the nuisance parameter ϵ are optimized simultaneously

[4]An ill-defined example is shown in Appendix A, where the considered detector-level observable, a one-dimensional Gaussian distribution, is not able to distinguish between effects from particle level and effects from detector level with θ. This limitation also exists in the standard binned maximum likelihood unfolding, as shown in Appendix B.

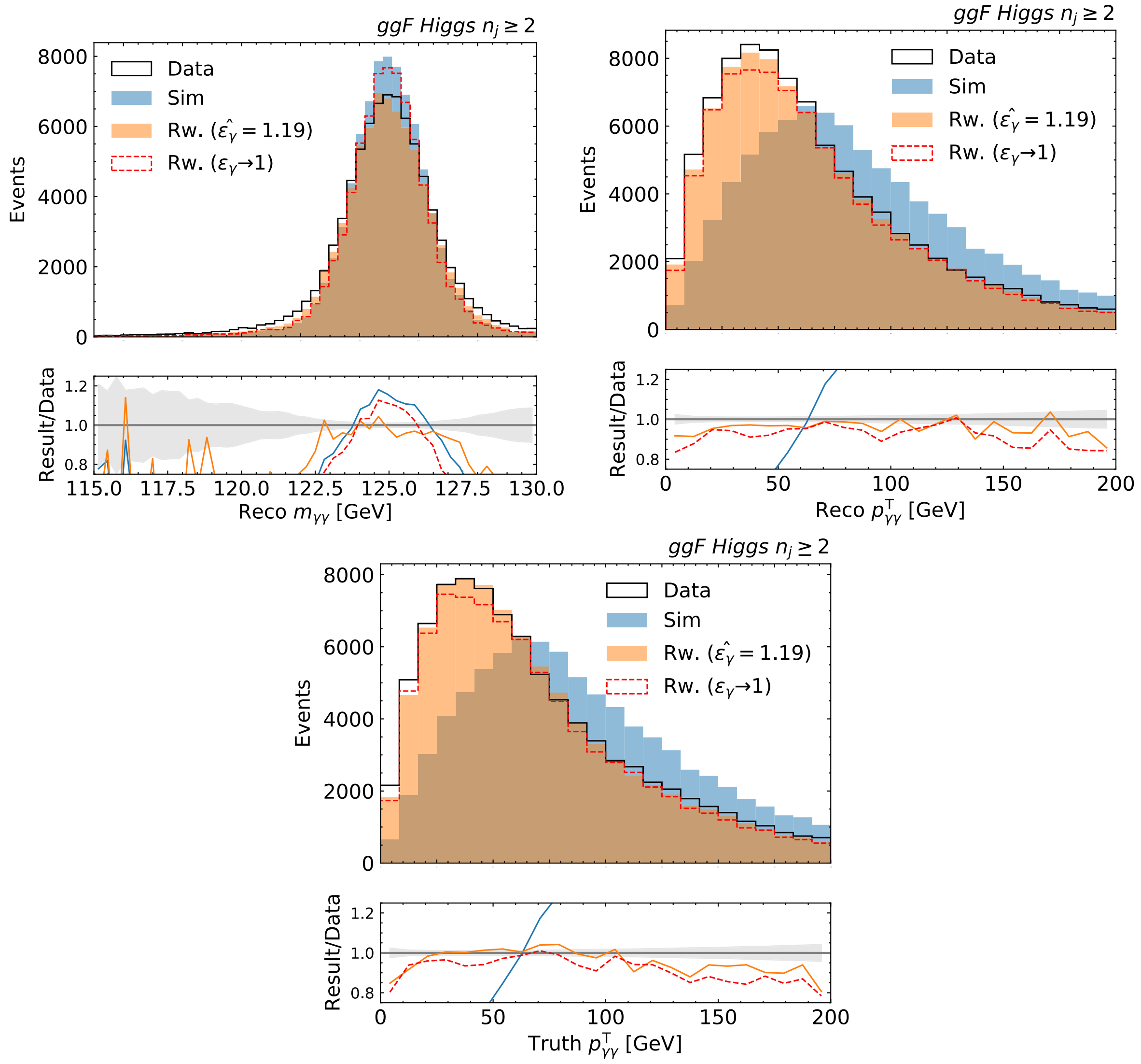

FIG. 4. Higgs boson cross section: results of the w_0 optimization. The nuisance parameter ϵ_γ is optimized simultaneously with w_0 with the prior constraint set to 50% (orange) or fixed to 1 for comparison (red). The fitted ϵ_γ is 1.19 ± 0.007. Top left: the detector-level spectrum $m_{\gamma\gamma}$ of the simulation template D_{sim} reweighted by the trained $w_0 \times w_1$, compared to the $m_{\gamma\gamma}$ spectrum of the observed data D_{obs}. Top right: the detector-level spectrum $p^T_{\gamma\gamma}$ of the simulation template D_{sim} reweighted by the trained $w_0 \times w_1$, compared to the $p^T_{\gamma\gamma}$ spectrum of the observed data D_{obs}. Bottom: the particle-level spectrum $p^T_{\gamma\gamma}$ of the simulation template D_{sim} reweighted by the trained w_0, compared to the $p^T_{\gamma\gamma}$ spectrum of the observed data D_{obs}. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of observed events in a given bin.

using D_{sim} as the simulation template with D_{obs} as the observed data used in Eq. (7). The prior in the penalty term in Eq. (7) is configured with an uncertainty of 80%. The fitted ϵ is 1.20 ± 0.004[5] (correct value is 1.2). As shown in Fig. 2, the reweighted spectra match well with observed data in both detector and particle level. For more realistic uncertainties (so long as the simulation is close to the right answer), the fidelity is even better.

[5]The fitted value is averaged over five different w_0 reweighters that are trained in the same way but with different random initializations. The standard deviation of the fitted values is taken as the error.

IV. HIGGS BOSON CROSS SECTION

We now demonstrate the unfolding method in a physics case—a Higgs boson cross section measurement. Here, we focus on the diphoton decay channel of the Higgs boson. The goal is then to measure the transverse momentum spectrum of the Higgs boson p^T_H using the transverse momentum of the diphoton system $p^T_{\gamma\gamma}$ at detector level. The photon resolution ϵ_γ is considered as a nuisance parameter. In this case, the $p_{\gamma\gamma}^{\mathrm{T}}$ spectrum is minimally affected by ϵ_γ. Therefore, we also consider the invariant mass spectrum of the diphoton system $m_{\gamma\gamma}$ at detector level, which is highly sensitive to ϵ_γ. In addition, In order to have a large spectrum difference between different datasets for demonstration purpose, we consider only events that contain at least two reconstructed jets, where the leading-order calculation would significantly differ from next-to-leading-order calculation.

Similar to the Gaussian examples, we prepare the following datasets:

(i) D_{obs}: used as the observed data.

(ii) $D_{\mathrm{sim}}^{1.0}$: used as the nominal simulation sample.

(iii) $D_{\mathrm{sim}}^{1.2}$: used as the simulation sample with a systematic variation.

(iv) D_{sim}^{*}: simulation sample with various ϵ_γ values for training the w_1 reweighter.

D_{obs} is generated at next-to-leading order using the POWHEGBOX program [34,35], while the rest are generated at leading order using MadGraph5_aMC@LO v2.6.5 [36]. For all samples, the parton-level events are processed by PYTHIA 8.235 [37,38] for the Higgs decay, the parton shower, hadronization, and the underlying event. The detector simulation is based on DELPHES 3.5.0 [39] with detector response modified from the default ATLAS detector card. For both D_{obs} and $D_{\mathrm{sim}}^{1.2}$, the photon resolution ϵ is multiplied by a factor of 1.2. For D_{sim}^{*}, the multiplier of ϵ is uniformly scanned between 0.5 and 1.5 with a step size of 0.01. $D_{\mathrm{sim}}^{1.0}$ uses the default ATLAS detector card.

Each of the spectra of particle-level $p_{\gamma\gamma}^{\mathrm{T}}$, detector-level $p_{\gamma\gamma}^{\mathrm{T}}$, and detector-level $m_{\gamma\gamma}$ is standardized to the spectrum with a mean of 0 and a standard deviation of 1 before being passed to the neural networks. A w_1 reweighter is then trained to reweight $D_{\mathrm{sim}}^{1.0}$ to D_{sim}^{*}. The trained w_1 is tested with the nominal detector level $p_{\gamma\gamma}^{\mathrm{T}}$ and $m_{\gamma\gamma}$ spectra ($D_{\mathrm{sim}}^{1.0}$) reweighted to $\epsilon_\gamma = 1.2$ and compared to the detector level $p_{\gamma\gamma}^{\mathrm{T}}$ and $m_{\gamma\gamma}$ spectra with $\epsilon_\gamma = 1.2$. As shown in Fig. 3, the trained w_1 reweighter has learned to reweight the nominal detector level $m_{\gamma\gamma}$ spectrum to match the detector level $m_{\gamma\gamma}$ spectrum with ϵ_γ at 1.2, and the detector level $p_{\gamma\gamma}^{\mathrm{T}}$ variable is independent of the w_1 reweighter.

The w_0 reweighter and ϵ are optimized simultaneously based on the pretrained w_1 reweighter. The prior of ϵ_γ is 50%. The fitted ϵ_γ is 1.19 ± 0.007. As shown in Fig. 4, the reweighted spectra match well with observed data in both detector and particle level. This means that the observed data p_H^{T} spectrum is successfully unfolded with the nuisance parameter ϵ_γ properly profiled. For comparison, we also perform UPU with ϵ_γ fixed at 1. As shown in Fig. 4, the unfolded p_H^{T} spectrum in this case has a larger nonclosure with the observed data due to the lack of profiling.

V. CONCLUSION AND OUTLOOK

In this paper, we proposed UPU, a new ML-based unfolding method that can process unbinned data and profile. The method uses the binned maximum likelihood as the figure of merit to optimize the unfolding reweighting function $w_0(t)$, which takes unbinned particle-level spectra as inputs. $w_0(t)$ and the nuisance parameters θ are optimized simultaneously, which also requires to learn a conditional likelihood ratio $w_1(t, r|\theta)$ that reweights the detector-level spectra based on the profiled values of nuisance parameters and is taken as an input for the optimization of $w_0(t)$ and θ.

In the Gaussian example, we demonstrated the optimization of w_1 and the optimization of w_0 and θ. The setup considers one dimension in the particle level and two dimensions in the detector level. The additional detector-level observable, which does not depend on θ, breaks the degeneracy between particle-level and detector-level effects and thus allows for optimization of w_1 and θ at the same time.

We also applied UPU to the Higgs boson cross section measurement. We considered one dimension at particle level and two dimensions at detector level. With one detector-level variable sensitive to the target particle-level observable and one sensitive to the effect of nuisance parameters, the data are successfully unfolded and profiled. The impact of profiling is also demonstrated by comparing with the result of nuisance parameter fixed to the nominal value. This can be readily extended to higher dimensions in either particle level or detector level, provided all particle-level and detector-level effects are distinguishable in the considered detector-level spectra. In the case of more than one nuisance parameters, one can either train multiple w_1 for each nuisance parameter separately or train a single w_1, which takes all nuisance parameters as inputs. As the effects of multiple nuisance parameters are usually assumed independent, one could take a product of individually trained reweighters.

As with any measurement, quantifying the uncertainty is critical to interpret UPU results. Just as in the binned case, one can calculate the uncertainty on the nuisance parameters which can be determined by fixing a given parameter to target values and then simultaneously reoptimizing w_0 and the rest of the nuisance parameters. A new feature of UPU is that the likelihood (ratio) itself is only an approximation, using neural networks as surrogate models. This is a challenge for all machine-learning-based unfolding, and uncertainties can be probed by comparing the results with different simulations. Future extensions of UPU may be able to also use machine learning to quantify these model uncertainties as well as process unbinned data also at detector level.

The code for this paper can be found at https://github.com/qwerasd903/UnbinnedProfiledUnfolding, which uses Jupyter Notebooks [40] and employs NumPy [41] for data manipulation and Matplotlib [42] for visualization. All of the machine learning was performed on an NVIDIA A100 Graphical Processing Unit (GPU) and reproducing the entire notebook takes about 13 hours. The physics data sets are hosted on Zenodo at Ref. [43].

ACKNOWLEDGMENTS

We thank A. Ghosh, V. Mikuni, J. Thaler, D. Whiteson, and S. L. Wu for helpful discussions and detailed feedback on the manuscript. B. N. is supported by the U.S. Department of Energy (DOE), Office of Science under Contract No. DE-AC02-05CH11231. J. C. is supported by the U.S. Department of Energy (DOE), Office of Science under Contracts No. DE-AC02-05CH11231 and No. DE-SC0017647.

APPENDIX A: GAUSSIAN EXAMPLE WITH ONE DIMENSION AT BOTH PARTICLE AND DETECTOR LEVELS

In this appendix, we apply UPU to the Gaussian example where each dataset represents one-dimensional Gaussian random variables at both the particle and detector levels. The particle-level random variable T is described by mean μ and standard deviation σ, while the detector-level variable is given by

$$R = T + Z, \tag{A1}$$

where Z is a Gaussian random variable with mean β and standard deviation ϵ.

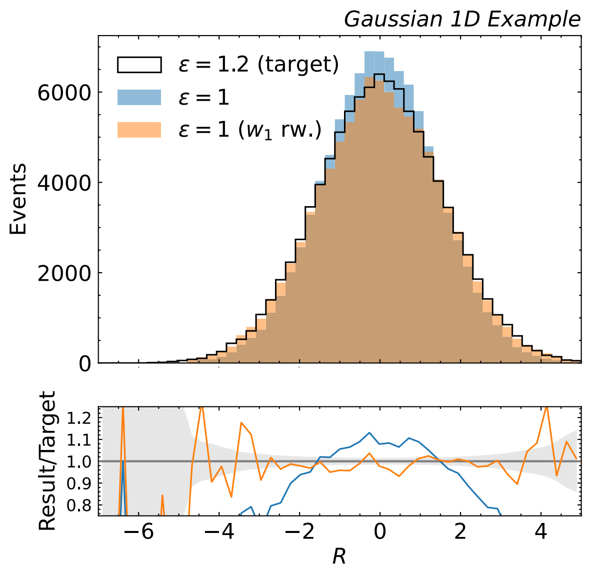

FIG. 5. Gaussian 1D example: the nominal R distribution ($\epsilon = 1$) reweighted by the trained w_1 conditioned at $\epsilon = 1.2$ and compared to R distribution with $\epsilon = 1.2$. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of $D_{\text{sim}}^{1.2}$ events in a given bin.

ϵ is considered to be the only nuisance parameter, and β is fixed to 0. Similar to the setup in Sec. III, four datasets are prepared for the full training and validation procedure:

(i) $D_{\text{sim}}^{1.0}$: 200,000 events with $\mu = 0$, $\sigma = 1$ and $\epsilon = 1$.
(ii) D_{obs}: 100,000 events with $\mu = 0.2$, $\sigma = 1$ and $\epsilon = 1.2$.
(iii) D_{sim}^{*}: 200,000 events with $\mu = 0$, $\sigma = 1$ and ϵ uniformly distributed from 0.2 to 1.8.
(iv) $D_{\text{sim}}^{1.2}$: 100,000 events with $\mu = 0$, $\sigma = 1$ and $\epsilon = 1.2$.

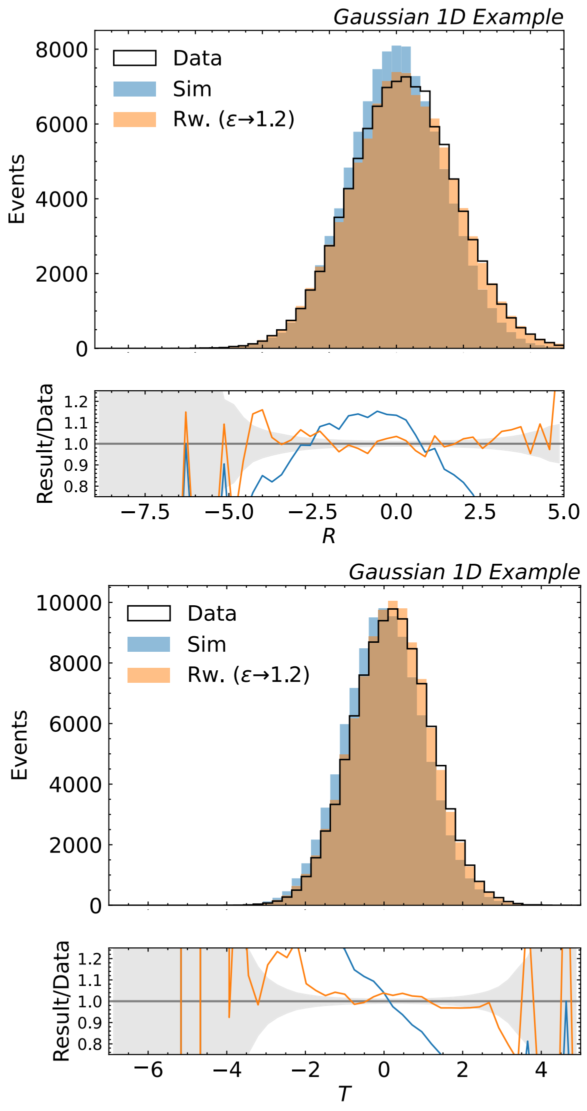

FIG. 6. Gaussian 1D example: results of the w_0 optimization. The nuisance parameter ϵ is fixed to 1.2, and the penalty term in Eq. (7) is set to 0. Top: the detector-level spectrum R of the simulation template D_{sim} reweighted by the trained $w_0 \times w_1$, compared to the R spectrum of the observed data D_{obs}. Bottom: the particle-level spectrum T of the simulation template D_{sim} reweighted by the trained w_0, compared to the T spectrum of the observed data D_{obs}. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of observed events in a given bin.

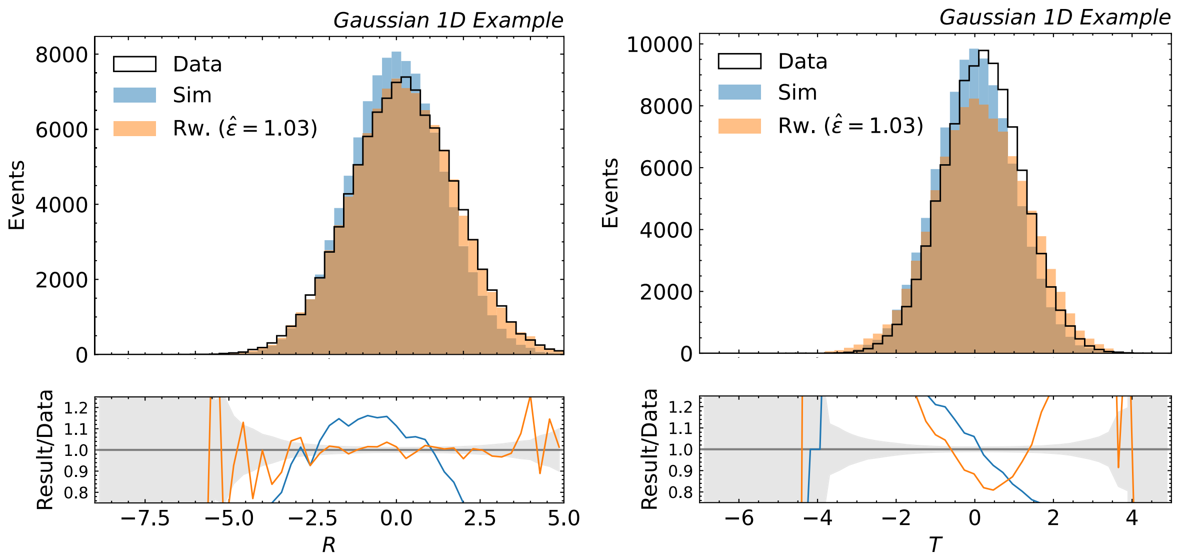

FIG. 7. Gaussian 1D example: results of the w_0 optimization. The nuisance parameter ϵ is optimized simultaneously with w_0 and the best-fit value is $\hat{\epsilon} = 1.03 \pm 0.016$. Left: the detector-level spectrum R of the simulation template D_{sim} reweighted by the trained $w_0 \times w_1$, compared to the R spectrum of the observed data D_{obs}. Right: the particle-level spectrum T of the simulation template D_{sim} reweighted by the trained w_0, compared to the T spectrum of the observed data D_{obs}. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of observed events in a given bin.

A w_1 reweighter is trained to reweight $D^{1.0}_{\mathrm{sim}}$ to D^{*}_{sim}. The trained w_1 is then tested with the nominal R distribution ($D^{1.0}_{\mathrm{sim}}$) reweighted to $\epsilon = 1.2$ ($w_1(R|T, \epsilon = 1.2)$) and compared to the R spectrum with $\epsilon = 1.2$ ($D^{1.2}_{\mathrm{sim}}$). As shown in Fig. 5, the trained w_1 reweighter has learned to reweight the nominal R spectrum to match the R spectrum with ϵ at 1.2.

With this trained w_1 reweighter, a w_0 reweighter is trained using $D^{1.0}_{\mathrm{sim}}$ as the simulation template with D_{obs} as the observed data used in Eq. (7). In the first scenario, the nuisance parameter ϵ for the w_1 reweighter is fixed to 1.2, and the penalty term in Eq. (7) $\log(\theta)$ is set to 0 (no constraint). As shown in Fig. 6, the w_0 reweighter is able to learn to reweight the particle-level spectrum T by matching the detector-level spectrum R to the observed spectrum. In the second scenario, the nuisance parameter ϵ is trained together with the w_0 reweighter. The prior in the penalty term in Eq. (7) is set to be a Gaussian probability density with a 80% uncertainty. As shown in Fig. 7, the trained w_0 and optimized ϵ are tested. The fitted ϵ is 1.03 ± 0.016[6] (true value is 1.2). The reweighted distribution matches well with observed data in the detector-level spectrum but the particle-level spectrum has a large nonclosure. This is because of the degeneracy between the w_0 and w_1 reweighters in the effect on the detector-level spectrum. In other words, detector effects can mimic changes in the particle-level cross section, so the data cannot distinguish between these two scenarios. This is a common issue which also exists in the standard binned maximum likelihood unfolding. For comparison, we also perform the standard binned maximum likelihood unfolding. As shown in Appendix B, the unfolded T spectrum in this case also fails to represent the true T spectrum. An 80% uncertainty is highly exaggerated from typical scenarios, but it clearly illustrates the challenge of profiling and unfolding at the same time.

[6]The fitted value is averaged over five different w_0 reweighters, which are trained in the same way, but with different random initializations. The standard deviation of the fitted values is taken as the error.

APPENDIX B: BINNED MAXIMUM LIKELIHOOD UNFOLDING WITH GAUSSIAN EXAMPLES

In this appendix, we present results of the standard binned maximum likelihood unfolding with Gaussian examples. The scenarios are

(i) One-dimension in both particle and detector level: This is the same example as described in Appendix A. The prior constraint for ϵ is set to 80%. The result is shown in Fig. 8 with ϵ fitted to 1.08 ± 0.02, which also indicates a degeneracy problem between particle and detector levels.

(ii) One dimension in particle level and two dimensions in detector level: This is the same example as described in Sec. III. The prior constraint for ϵ is set to 80%. The result is shown in Fig. 9 with ϵ fitted to 1.19 ± 0.003. The degeneracy problem is resolved after considering an additional spectrum in the detector level.

All the maximum likelihood fittings are performed using PYHF [44,45].

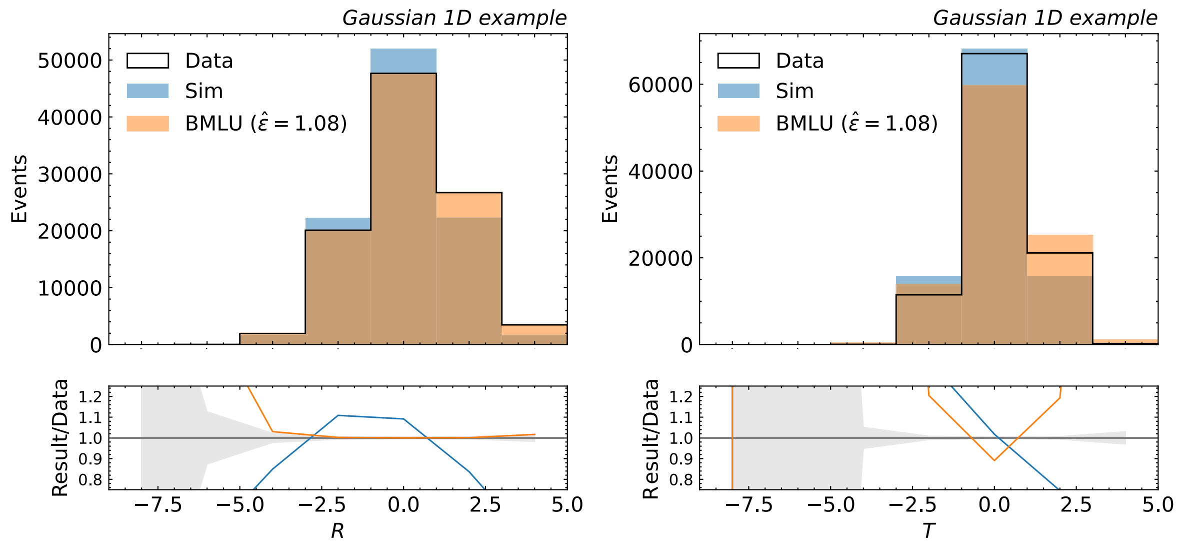

FIG. 8. Gaussian 1D example: results of the binned maximum likelihood unfolding. The prior constraint for ϵ is set to 80% and the fitted ϵ is 1.08 ± 0.02. Left: the fitted detector-level spectrum R of the simulation template D_{sim}, compared to the R spectrum of the observed data D_{obs}. Right: the unfolded particle-level spectrum T of the simulation template D_{sim}, compared to the T spectrum of the observed data D_{obs}. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of observed events in a given bin.

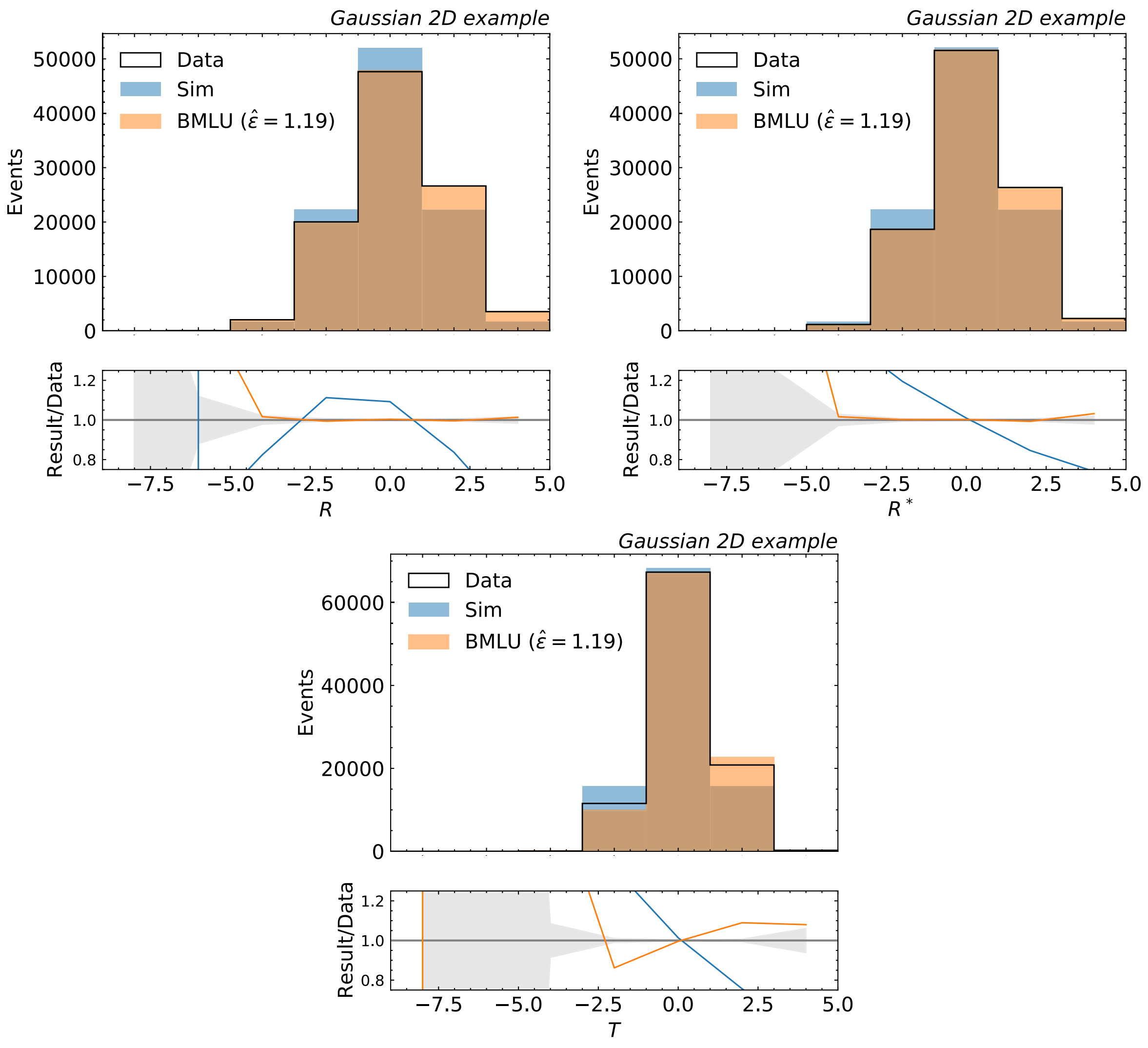

FIG. 9. Gaussian 2D example: results of the binned maximum likelihood unfolding. The prior constraint for ϵ is set to 80% and the fitted ϵ is 1.19 ± 0.003. Top left: the fitted detector-level spectrum R of the simulation template D_{sim}, compared to the R spectrum of the observed data D_{obs}. Top right: the fitted detector-level spectrum R^* of the simulation template D_{sim}, compared to the R' spectrum of the observed data D_{obs}. Bottom: the unfolded particle-level spectrum T of the simulation template D_{sim}, compared to the T spectrum of the observed data D_{obs}. The shaded band in the bottom panel represents the data statistical uncertainty, which is estimated as $1/\sqrt{n}$, where n is the number of observed events in a given bin.

[1] G. Cowan, A survey of unfolding methods for particle physics, Conf. Proc. C **0203181**, 248 (2002).
[2] V. Blobel, Unfolding methods in particle physics, in *PHYSTAT2011 Proceedings* (CERN, Geneva, 2011), p. 240.
[3] V. Blobel, Unfolding, in *Data Analysis in High Energy Physics: A Practical Guide to Statistical Methods* (2013), p. 187, 10.1002/9783527653416.ch6.
[4] R. Balasubramanian, L. Brenner, C. Burgard, G. Cowan, V. Croft, W. Verkerke, and P. Verschuuren, Statistical method and comparison of different unfolding techniques using RooFit, Int. J. Mod. Phys. A **35**, 2050145 (2020).
[5] G. D'Agostini, A multidimensional unfolding method based on Bayes' theorem, Nucl. Instrum. Methods Phys. Res., Sect. A **362**, 487 (1995).
[6] A. Hocker and V. Kartvelishvili, SVD approach to data unfolding, Nucl. Instrum. Methods Phys. Res., Sect. A **372**, 469 (1996).
[7] S. Schmitt, TUnfold: An algorithm for correcting migration effects in high energy physics, J. Instrum. **7**, T10003 (2012).
[8] M. Arratia *et al.*, Publishing unbinned differential cross section results, J. Instrum. **17**, P01024 (2021).

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2, NIPS'14* (MIT Press, Cambridge, MA, USA, 2014), pp. 2672–2680.

[10] K. Datta, D. Kar, and D. Roy, Unfolding with generative adversarial networks, arXiv:1806.00433.

[11] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, How to GAN away detector effects, SciPost Phys. **8**, 070 (2020).

[12] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, arXiv:1312.6114.

[13] J. N. Howard, S. Mandt, D. Whiteson, and Y. Yang, Foundations of a fast, data-driven, machine-learned simulator, Sci. Rep. **12**, 7567 (2022).

[14] D. J. Rezende and S. Mohamed, Variational inference with normalizing flows, Int. Conf. Mach. Learn. **37**, 1530 (2015).

[15] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone, and U. Köthe, Invertible networks or partons to detector and back again, SciPost Phys. **9**, 074 (2020).

[16] M. Vandegar, M. Kagan, A. Wehenkel, and G. Louppe, Neural empirical Bayes: Source distribution estimation and its applications to simulation-based inference, in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research Vol. 130, edited by A. Banerjee and K. Fukumizu (PMLR, 2021), pp. 2107–2115, arXiv:2011.05836.

[17] M. Backes, A. Butter, M. Dunford, and B. Malaescu, An unfolding method based on conditional invertible neural networks (cINN) using iterative training, arXiv:2212.08674.

[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).

[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* (ICLR, United States, 2021), arXiv:2105.04448.

[20] A. Andreassen and B. Nachman, Neural networks for full phase-space reweighting and parameter tuning, Phys. Rev. D **101**, 091901 (2020).

[21] V. Andreev *et al.*, Measurement of Lepton-Jet Correlation in Deep-Inelastic Scattering with the H1 Detector Using Machine Learning for Unfolding, Phys. Rev. Lett. **128**, 132002 (2022).

[22] H1 Collaboration, Multi-differential jet substructure measurement in high Q^2 DIS events with HERA-II data, H1prelim-22-034 (2022).

[23] LHCb Collaboration, Multidifferential study of identified charged hadron distributions in Z-tagged jets in proton-proton collisions at $\sqrt{s} = 13$ TeV, arXiv:2208.11691.

[24] D. de Florian *et al.* (LHC Higgs Cross Section Working Group), *Handbook of LHC Higgs Cross Sections: 4. Deciphering the Nature of the Higgs Sector* (CERN, Geneva, 2017), Vol. 2.

[25] J. R. Andersen *et al.*, Les Houches 2015: Physics at TeV colliders standard model working group report, in *9th Les Houches Workshop on Physics at TeV Colliders* (PhysTeV, Les Houches, France, 2016), arXiv:1605.04692.

[26] N. Berger *et al.*, Simplified template cross sections—stage 1.1, arXiv:1906.02754.

[27] S. Amoroso *et al.*, Les Houches 2019: Physics at TeV colliders: Standard model working group report, in *11th Les Houches Workshop on Physics at TeV Colliders: PhysTeV Les Houches* (PhysTeV, Les Houches, France, 2020), arXiv:2003.01700.

[28] G. Choudalakis, Fully Bayesian unfolding, arXiv:1201.4612.

[29] B. Nachman and J. Thaler, Neural conditional reweighting, Phys. Rev. D **105**, 076015 (2022).

[30] 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).

[31] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density Ratio Estimation in Machine Learning* (Cambridge University Press, Cambridge, England, 2012).

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., Vancouver, 2019), pp. 8024–8035.

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

[34] C. Oleari, The POWHEG-BOX, Nucl. Phys. B, Proc. Suppl. **205–206**, 36 (2010).

[35] S. Alioli, P. Nason, C. Oleari, and E. Re, NLO Higgs boson production via gluon fusion matched with shower in POWHEG, J. High Energy Phys. 04 (2009) 002.

[36] 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, J. High Energy Phys. 07 (2014) 079.

[37] T. Sjöstrand, S. Mrenna, and P. Z. Skands, PYTHIA 6.4 physics and manual, J. High Energy Phys. 05 (2006) 026.

[38] 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).

[39] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (DELPHES 3 Collaboration), DELPHES 3, A modular framework for fast simulation of a generic collider experiment, J. High Energy Phys. 02 (2013) 057.

[40] 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, Amsterdam, 2016), pp. 87–90.

[41] C. R. Harris *et al.*, Array programming with NumPy, Nature (London) **585**, 357 (2020).

[42] J. D. Hunter, Matplotlib: A 2d graphics environment, Comput. Sci. Eng. **9**, 90 (2007).

[43] J. Chan and B. Nachman, Higgs to diphoton channel at least 2 jet datasets, 10.5281/zenodo.7553271 (2023).

[44] L. Heinrich, M. Feickert, and G. Stark, PYHF: v0.7.0, https://github.com/scikit-hep/pyhf/releases/tag/v0.7.0.

[45] L. Heinrich, M. Feickert, G. Stark, and K. Cranmer, PYHF: Pure-Python implementation of HistFactory statistical models, J. Open Source Software **6**, 2823 (2021).