MACHINE LEARNING-BASED UNFOLDING FOR CROSS SECTION MEASUREMENTS IN THE PRESENCE OF NUISANCE PARAMETERS

By Huanbiao Zhu[1,a], Krish Desai[2,d], Mikael Kuusela[1,b], Vinicius Mikuni[3,e], Benjamin Nachman[4,f] and Larry Wasserman[1,c]

[1] *Department of Statistics and Data Science, Carnegie Mellon University,* [a]*huanbiaz@andrew.cmu.edu;* [b]*mkuusela@andrew.cmu.edu;* [c]*larry@stat.cmu.edu*

[2] *Department of Physics, University of California, Berkeley,* [d]*krish.desai@berkeley.edu*

[3] *Nagoya University, Kobayashi-Maskawa Institute, Japan,* [e]*vmikuni@hepl.phys.nagoya-u.ac.jp*

[4] *Department of Particle Physics and Astrophysics, Stanford University; Fundamental Physics Directorate, SLAC National Laboratory,* [f]*nachman@stanford.edu*

Statistically correcting measured cross sections for detector effects is an important step across many applications. In particle physics, this inverse problem is known as *unfolding*. In cases with complex instruments, the distortions they introduce are often known only implicitly through simulations of the detector. Modern machine learning has enabled efficient simulation-based approaches for unfolding high-dimensional data. Among these, one of the first methods successfully deployed on experimental data is the OmniFold algorithm, a classifier-based Expectation-Maximization procedure. In practice, however, the forward model is only approximately specified, and the corresponding uncertainty is encoded through nuisance parameters. Building on the well-studied OmniFold algorithm, we show how to extend machine learning-based unfolding to incorporate nuisance parameters. Our new algorithm, called Profile OmniFold, is demonstrated using a Gaussian example as well as a particle physics case study using simulated data from the CMS Experiment at the Large Hadron Collider.

1. Introduction.

Detector effects distort spectra from their true values. Statistically removing these distortions is essential for comparing results across experiments and for facilitating broad, detector-independent analysis of the data. In particle and nuclear physics, this problem is known as *unfolding*. While the problem is general, we will focus on this application area because particle detector responses are highly complex and are typically characterized only implicitly through detailed simulations, making simulation-based approaches particularly relevant. The objective is to recover the underlying distribution (called *differential cross section* in physics) of some physical quantity x, referred to as particle-level (or pre-detector level) truth, from observations of a smeared version y, known as detector-level (or reconstructed) data.

In practice, both x and y can be high-dimensional and their probability densities are related by a Fredholm integral equation of the first kind

$$p_Y(y) = \int_{\mathcal{X}} k(y,x) p_X(x) dx, \tag{1}$$

where $k(y,x)$ is the *response kernel* that models the detector response. Without considering efficiency effects, it can be interpreted as the conditional density of observing smeared y given true x, i.e., $k(y,x) = p(y|x)$. The goal of unfolding is to estimate the true density function p_X given an i.i.d. sample of smeared observations $Y_1, ..., Y_n \sim p_Y$.

Keywords and phrases: Ill-posed inverse problems, Fredholm integral equation of the first kind, Simulation-based Inference, Expectation-Maximization algorithm.

Recently, a number of machine learning-based approaches have been proposed to address this problem (Arratia et al., 2022; Huetsch et al., 2024), and among those, one of the earliest methods proposed is OmniFold (Andreassen et al., 2020, 2021). OmniFold (OF) is a classifier-based algorithm that iteratively reweights simulated data to match the experimental data. At the population level, OmniFold is an Expectation-Maximization (EM) algorithm (Dempster, Laird and Rubin, 1977), provided an infinite sample size and optimal classifier. OmniFold has been shown to be effective in high-dimensional settings and successfully applied to experimental data from the Large Hadron Collider (LHC) at CERN and other particle and nuclear physics experiments (Andreev et al., 2021; H1 Collaboration, 2022a; Collaboration, 2022b; Komiske, Kryhin and Thaler, 2022; Andreev et al., 2023; H1 Collaboration, 2023; Song, 2023; Pani, 2024; CMS Collaboration, 2024a; Aad et al., 2024; ATLAS Collaboration, 2024b; Badea et al., 2025; Canelli et al., 2025).

However, one limitation in OmniFold, and all other current machine learning-based methods, is the assumption that the detector response is accurately modeled in the simulation. In practice, this is only approximately true, with the simulation potentially depending on a number of nuisance parameters that can be constrained by the observed data. Specifically, this means we have the following forward model

$$p_Y(y) = \int_{\mathcal{X}} k_\theta(y,x) p_X(x) dx, \tag{2}$$

where the response kernel depends on some nuisance parameters θ. Without a correctly specified response kernel, the solution by OmniFold and other machine learning methods will be biased. Traditionally, this bias has been addressed by repeating the measurement using systematic variations of the detector response, which is an expensive and conservative step. In this paper, we approach this problem by proposing a new algorithm, called Profile OmniFold (POF), for unfolding in the presence of nuisance parameters. POF can be seen as an extension to the original OF algorithm, which iteratively reweights the simulated data, but at the same time simultaneously updates the nuisance parameters. This paper builds upon and expands the preliminary results presented in the NeurIPS Machine Learning and the Physical Sciences (ML4PS) workshop paper Zhu et al. (2024).

The rest of this paper is organized as follows. In Section 2, we provide an overview of EM algorithms applied to the unfolding problem, along with recent developments in simulation-based machine learning approaches for this class of problems. Building on this foundation, Section 3 introduces our new methodology, which addresses the challenge posed by nuisance parameters in the response kernel—a scenario where existing machine learning methods do not apply. In Section 4, we demonstrate the proposed method using a simulated Gaussian example. Section 5 presents an application to publicly available simulated data from the CMS Experiment at the LHC. Finally, Section 6 discusses the limitations of our approach and outlines directions for future research. The proofs of the main propositions as well as additional experimental results are provided in the supplementary material.

2. EM Algorithm for Unfolding. There is a rich body of literature on solving the Fredholm integral equation of the first kind in Equation (1). In particular, an EM algorithm has been widely used to solve this problem, which has the following form

$$f^{(k+1)}(x) = f^{(k)}(x) \int \frac{p_Y(y)}{p_Y^{(k)}(y)} p(y|x) dy, \tag{3}$$

where

$$p_Y^{(k)}(y) = \int p(y|x') f^{(k)}(x') dx', \tag{4}$$

and $f^{(0)}(x) \geq 0$ for any $x \in \mathcal{X}$ and $\int_{\mathcal{X}} f^{(0)}(x)dx = 1$. The solution after k iteration is given by $f^{(k)}$. To the best of our knowledge, the earliest description of this algorithm appears in the work of Kondor (1983), which referred to it as the method of convergent weights. They did not derive the algorithm from the EM perspective, but instead based on the intuition that

$$\left[1 - \frac{1}{r^{(k)}(y)}\right] p(y) = \int p(y|x)(p_X(x) - f^{(k)}(x))dx, \quad y \in \mathcal{Y}, \tag{5}$$

where $r^{(k)}(y) := \frac{p_Y(y)}{p_Y^{(k)}(y)}$ is the ratio of the smearing density and the updated density from the algorithm after k iteration. Therefore, by constructing a sequence $r^{(k)}(y)$ that converges to one, the hope is that the corresponding $f^{(k)}$ will converge to p_X. Kondor presented the description of the algorithm along with a few examples, but did not establish the convergence property of the algorithm. Subsequently, Mülthei and Schorr (1987, 1989); Mülthei (1992) connected the algorithm to the maximization of a concave functional, namely the population-level log-likelihood for a density function f on $\mathcal{X}$

$$\ell(f) = \int p_Y(y) \log\left(\int p(y|x)f(x)dx\right) dy, \tag{6}$$

or equivalently, minimization of the Kullback-Leibler divergence (Mülthei and Schorr, 1989)

$$\mathrm{KL}\left(p_Y, \int p(\cdot|x)f(x)dx\right) = \int p_Y(y) \log \frac{p_Y(y)}{\int p(y|x)f(x)dx} dy, \tag{7}$$

with respect to f. In particular, under the assumption that the kernel $p(y|x)$ is strictly positive on the compact support $[0,1]^2$, Mülthei and Schorr (1987, Theorem 8) showed that if $f^{(k)}$ converges to some $\tilde{f}$ with respect to the L^1 norm, then $\tilde{f}$ is a maximizer of (6). Subsequently, Mülthei (1992, Theorem 5) showed that $p_Y^{(k)}$ converges uniformly to p_Y if p_X is strictly positive. Without assuming compact support, Chae, Martin and Walker (2019, Theorem 1) proved that $\mathrm{KL}(p_Y, p_Y^{(k)}) \to \inf_f \mathrm{KL}(p_Y, \int p(y|x)f(x)dx)$, provided there exists a convergent sequence $(f_*^{(k)})_{k\geq 1}$ such that $\mathrm{KL}(p_Y, p_Y^{(k)}) \to \mathrm{KL}(p_Y, \int p(y|x)f_*^{(k)}(x)dx)$. Beyond these results, Eggermont and Lariccia (1995, 1997); Eggermont (1999) established similar convergence results for a smoothed version of the EM algorithm. More recently, Crucinio, Doucet and Johansen (2023) analyzed the theoretical properties of the expectation maximization smoothing (EMS) scheme and proposed a particle algorithm as a sequential Monte Carlo method to approximate the EMS iteration.

Before these developments, part of the motivation for understanding the algorithm (3) comes from the well-known results by Shepp and Vardi (1982); Vardi, Shepp and Kaufman (1985), where they studied the discretized version of (1) as a model for image reconstruction in Positron Emission Tomography (PET). In particle physics, similar models have also been applied in binned unfolding, where measurements are binned into a histogram (or are naturally represented as discrete, e.g., in images) and the particle-level spectrum is likewise represented as a histogram.

2.1. *Binned Unfolding.* In the binned setting, the task is to estimate the true unknown histogram mean $\boldsymbol{\lambda} = [\lambda_1, ..., \lambda_B]$, where $\lambda_j = \int_{\mathcal{X}_j} p_X(x)dx$ for bins $\mathcal{X}_1, ..., \mathcal{X}_B$ at the particle level. The observed data are the detector-level histogram $\mathbf{n}^* = [n_1^*, ..., n_D^*]^T$, where D is the number of bins at the detector level. Since events can be modeled as Poisson point processes, each bin count independently follows a Poisson distribution (Kuusela, 2012; Blobel, 2011).

Therefore, the likelihood function for $\boldsymbol{\lambda}$ given the the observed data $\mathbf{n}^*$ is

$$L(\boldsymbol{\lambda}|\mathbf{n}^*) = \prod_{i=1}^{D} \frac{\left(\sum_{j=1}^{B} K_{ij}\lambda_j\right)^{n_i^*}}{n_i^*!} e^{-\sum_{j=1}^{B} K_{ij}\lambda_j}, \tag{8}$$

where K is a $D \times B$ *response matrix* with entries representing the bin-to-bin smearing probabilities, i.e., $K_{ij} = P(\text{observation in bin } i \mid \text{true value in bin } j)$.

To obtain the maximum likelihood estimate, a classical approach has been the D'Agostini iteration (D'Agostini, 1995). D'Agostini iteration can be viewed as an EM algorithm with early stopping (Kuusela, 2012), which is equivalent to the procedure originally proposed in Shepp and Vardi (1982). The same algorithm has also been known as the Richardson-Lucy algorithm (Richardson, 1972; Lucy, 1974). Specifically, starting from an initial guess $\boldsymbol{\lambda}^{(0)} > \mathbf{0}$, each component of $\hat{\boldsymbol{\lambda}}^{(k+1)}$ is updated iteratively by

$$\hat{\lambda}_j^{(k+1)} = \frac{\hat{\lambda}_j^{(k)}}{\sum_i K_{ij}} \sum_i \frac{K_{ij} n_i^*}{\sum_l K_{il} \hat{\lambda}_l^{(k)}}, \quad j = 1, ..., B. \tag{9}$$

After k iterations, the solution is given by $\hat{\boldsymbol{\lambda}}^{(k)} = (\hat{\lambda}_1^{(k)}, ..., \hat{\lambda}_B^{(k)})$. As $k \to \infty$, it can be shown that $\hat{\boldsymbol{\lambda}}^{(k)}$ converges to the maximum likelihood estimate of $\boldsymbol{\lambda}$ (Vardi, Shepp and Kaufman, 1985). Also, since each step of the iteration increases the likelihood monotonically, stopping early in the iterations regularizes the solution. Moreover, comparing with the update rule in (9), the algorithm (3) can be viewed as a continuous analog of the discrete update in (9), assuming the full efficiency (i.e. $\sum_i K_{ij} = 1$ for all j).

2.2. *Unbinned Unfolding.* Binned unfolding has been the classical approach in particle and nuclear physics for decades. However, discretization requires pre-specifying the number of bins, which itself is a tuning parameter and can vary between different experiments. Additionally, binning limits the number of observables that can be simultaneously unfolded. This motivates the development of unbinned unfolding, which turns out to be closely related to algorithm (3).

As mentioned above, the iterative algorithm (3) can be derived as an EM algorithm that aims to maximize the population-level log-likelihood. The idea is that we treat the set of smeared observations Y as the observed variables and the target truth quantities X as the unobserved latent variables. The parameter (function) of interest f is the density function of X. The corresponding population-level Q-function is the expected complete-data log-likelihood conditioning on the observed variables Y and the current estimate $f^{(k)}$ integrating with respect to p_Y, i.e.,

$$\begin{aligned} Q(f, f^{(k)}) &= \int p_Y(y) \int p(x|y, f^{(k)}) \log[p(y|x) f(x)] dx dy \\ &= \int p_Y(y) \int \frac{p(y|x) f^{(k)}(x)}{\int p(y|x') f^{(k)}(x') dx'} \log[p(y|x) f(x)] dx dy. \end{aligned} \tag{10}$$

In the EM algorithm, the expectation (E) step computes the function $Q(f, f^{(k)})$ and the maximization (M) step updates the estimate by solving $f^{(k+1)} = \arg\max_f Q(f, f^{(k)})$ subject to the constraint $\int f(x) dx = 1$.

Proposition 1 establishes that solving the optimization problem above yields the algorithm (3). This result (and similar variants) has been presented in Mülthei and Schorr (1987); Andreassen et al. (2020); Falcão and Takacs (2025). We provide the proof in supplementary material for completeness.

PROPOSITION 1. Let $f^{(k+1)} = \arg\max_f Q(f, f^{(k)})$ subject to the constraint that $\int f(x) dx = 1$. Then

$$f^{(k+1)}(x) = f^{(k)}(x) \int \frac{p_Y(y)}{\int p(y|x') f^{(k)}(x') dx'} p(y|x) dy. \tag{11}$$

2.3. *Machine-learning based Method for Unfolding:* OMNIFOLD. Although the EM algorithm (3) provides a principled approach to solving the inverse problem in (1), this is challenging to implement in practice for two key reasons: (1) the analytic forms of p_Y and $p(y|x)$ are typically unknown in particle and nuclear physics experiments, and (2) both distributions may be high-dimensional, making them difficult to estimate. Recently, however, a line of machine learning-based unfolding methods has independently developed variants of algorithm (3) that circumvent these challenges and enable obtaining solutions to (1) even in high-dimensional settings. The first (and so far, only) one to be deployed to experimental data is OMNIFOLD (Andreassen et al., 2020, 2021). The core idea of OMNIFOLD is to use neural network classifiers to estimate the density ratios involved in the EM algorithm (3), thereby avoiding explicit estimation of p_Y or $p(y|x)$. This is made possible by access to a set of Monte Carlo (MC) simulated data $\{X_i', Y_i'\}_{i=1}^n \sim q_{X,Y}$, which in particle and nuclear physics is routinely available from high-fidelity detector simulations that mimic the data-generating process. The key assumption here is that $q(y|x) = p(y|x)$, meaning the response kernel (or the forward operator) stays the same between the MC and experimental data. However, it should be noted that the marginal distributions q_X and p_X (and hence q_Y and p_Y) are not assumed to be the same. For simplicity, we will omit the subscript X and Y in what follows if there is no confusion. Under this setting, OMNIFOLD approaches the unfolding task as follows:

Provided pairs of MC simulations $\{X_i', Y_i'\}_{i=1}^n \sim q_{X,Y}$ and a set of observed detector-level data $\{Y_i\}_{i=1}^m \sim p_Y$, let $\nu(x)$ be a reweighting function on the MC particle-level density $q(x)$. The goal is to estimate the true reweighting function $\nu^*(x) = \frac{p(x)}{q(x)}$. By this reparameterization, the population log-likelihood for a reweighting function ν is

$$\ell(\nu) = \int p(y) \log \left(\int p(y|x)\nu(x)q(x)dx \right) dy, \tag{12}$$

and the corresponding Q-function is

$$Q(\nu, \nu^{(k)}) = \int p(y) \int \frac{p(y|x)\nu^{(k)}(x)q(x)}{\int p(y|x')\nu^{(k)}(x')q(x')dx'} \log[p(y|x)\nu(x)q(x)]dxdy. \tag{13}$$

Subject to the normalization constraint $\int \nu(x)q(x)dx = 1$, the EM update takes the form

$$\nu^{(k+1)}(x) = \nu^{(k)}(x) \int \frac{p(y)}{\int \nu^{(k)}(x')q(x', y)dx'} p(y|x)dy. \tag{14}$$

OMNIFOLD implements this update via a two-step procedure:

1. Detector-level reweighting:
$r^{(k)}(y) = \frac{p(y)}{\tilde{q}^{(k)}(y)}$, where $\tilde{q}^{(k)}(y) = \int \nu^{(k)}(x')q(x', y)dx'$.
2. Particle-level reweighting:
$\nu^{(k+1)}(x) = \nu^{(k)}(x)\frac{\tilde{q}^{(k)}(x)}{q(x)}$, where $\tilde{q}^{(k)}(x) = \int r^{(k)}(y')q(x, y')dy'$.

Notice that combining these two steps yields the update (14). Moreover, since each step involves a density ratio, it can be estimated using a binary classifier without estimation of the marginal densities separately; see Section 2.4 for details. These two steps also have intuitive interpretations: the first step learns to reweight the detector-level MC density $\tilde{q}^{(k)}(y)$ in the current iteration to match the detector-level experimental density $p(y)$. The second step pulls back this density ratio to the particle level and updates the corresponding particle-level weights. In the next iteration, the updated weights are pushed forward to the detector level, and the process repeats. Here, pushing forward (or pulling back) refers to transferring the corresponding weights between paired particle-level and detector-level events. Further details are available in Andreassen et al. (2020).

2.4. *Estimating density ratio through binary classification.* As a key ingredient in the two steps of OMNIFOLD, we briefly describe how to estimate a density ratio using a classifier. Given i.i.d. samples $x_1, \ldots, x_n \sim p_1$ and $x_1', \ldots, x_m' \sim p_0$, assign class labels $c = 1$ to $\{x_i\}_{i=1}^n$ and $c = 0$ to $\{x_i'\}_{i=1}^m$. Additionally, let w_i denote the weight associated with x_i. The weighted density ratio we aim to estimate is

$$r(x) = \frac{w(x)p_1(x)}{p_0(x)}. \tag{15}$$

Using Bayes' rule, we can express

$$\frac{p(c=1|x)}{p(c=0|x)} = \frac{w(x)p_1(x)}{p_0(x)} \cdot \frac{p(c=1)}{p(c=0)}$$

and hence

$$r(x) = \frac{p(c=1|x)}{p(c=0|x)} \cdot \frac{p(c=0)}{p(c=1)}. \tag{16}$$

A probabilistic classifier $\hat{f} : \mathcal{X} \to [0,1]$ is trained on the weighted dataset $\{w_i, x_i, c = 1\}_{i=1}^n$ and $\{x_i', c = 0\}_{i=1}^m$. The output of the classifier approximates the conditional probability of class $c = 1$, i.e., $\hat{f}(x) = \hat{p}(c = 1|x)$. On the other hand, the prior odds can be estimated from the weighted sample sizes as

$$\frac{p(c=1)}{p(c=0)} \approx \frac{\sum_{i=1}^n w_i}{m}. \tag{17}$$

Thus, the density ratio can be estimated as

$$\hat{r}(x) = \frac{\hat{f}(x)}{1 - \hat{f}(x)} \cdot \frac{\sum_{i=1}^n w_i}{m}. \tag{18}$$

Under a Bayes optimal classifier and as the sample size tends to infinity, $\hat{r}(x) \to r(x)$. Similar results can also be shown if the weights are associated with x_i' instead of x_i. For more details about density ratio estimation using classifiers, see, for example, Andreassen and Nachman (2019); Cranmer, Pavez and Louppe (2015); Chapter 4 of Sugiyama, Suzuki and Kanamori (2012).

2.5. *Other machine learning approaches.* Although this work is primarily motivated by OMNIFOLD, which is a classifier-based EM algorithm, other machine learning approaches have also been proposed for unbinned unfolding. Notably, another line of machine learning methods for unfolding uses generative models to learn the conditional distribution of the unfolded events given the observed data (Datta, Kar and Roy, 2019; Bellagente et al., 2020; Shmakov et al., 2023; Backes et al., 2024; Diefenbacher et al., 2024; Butter et al., 2025a,b,c; Barman, Choudhury and Sarkar, 2025; Petitjean et al., 2025). In particular, it proceeds as follows: Initialize $p^{(0)}(x) = q(x), p^{(0)}(y) = q(y)$, then for iteration k,

1. Train a generative model for $p^{(k)}(x|y)$ using the generated data at $(k-1)^{th}$ iteration. Conditioning on experimental data $Y_i \sim p_Y$, generate $\tilde{X}_i^{(k)} \sim p^{(k)}(\cdot|Y_i)$. Denote the distribution of $\tilde{X}_1^{(k)}, \ldots, \tilde{X}_n^{(k)}$ as $p^{(k)}(x)$.
2. Estimate $r^{(k)}(x) = \frac{p^{(k)}(x)}{q(x)}$. Then reweight the detector-level MC density by $p^{(k)}(y) = r_{push}^{(k)}(y)q(y)$, where $r_{push}^{(k)}(y) = \int r^{(k)}(x)q(x|y)dx$.

At the population level, this approach is equivalent to OMNIFOLD. To see this, notice that

$$
\begin{aligned}
p^{(k+1)}(x) &= \int p^{(k)}(x|y)p(y)dy \\
&= \int \frac{p(y|x)p^{(k)}(x)}{p^{(k)}(y)}p(y)dy \\
&= p^{(k)}(x)\int \frac{p(y)}{p^{(k)}(y)}p(y|x)dy,
\end{aligned} \tag{19}
$$

where $p^{(k)}(y) = \int r^{(k)}(x)q(x|y)q(y)dx$. Since $p^{(0)}(x) = q(x)$ and $p^{(0)}(y) = q(y)$, by induction we have $p^{(k)}(y) = \tilde{q}^{(k)}(y)$ for all $k \geq 0$. This shows that the update for $p^{(k)}(x)$ matches that of $\nu^{(k)}(x)$ in (14) with $p^{(k)}(x) = \nu^{(k)}(x)q(x)$. While the generative unfolding is still an EM algorithm in population level, most paper have been focusing on one step of training a generative model to learn $p(x|y)$ without iterating the procedure necessarily. Implementation-wise, various generative models have been explored in this context, including generative adversarial networks (Datta, Kar and Roy, 2019; Bellagente et al., 2020), diffusion models (Shmakov et al., 2023), normalizing flows (Backes et al., 2024), Schrödinger Bridges (Diefenbacher et al., 2024), and conditional flow matching (Petitjean et al., 2025).

3. Unfolding in the Presence of Nuisance Parameters in the Forward Operator.

Although the response kernel is approximately the same between the simulated and experimental data, in practice it may still depend on one or more nuisance parameters. Consider the forward model (2), where $\theta \in \mathbb{R}^p$ denotes the nuisance parameter. For simplicity, we focus on the case $p = 1$. If the nuisance parameter is misspecified (i.e. $p(y|x) \neq q(y|x)$), the results obtained from OMNIFOLD and other machine learning–based unfolding methods will be biased. To address this, we introduce the Profile OMNIFOLD algorithm, which extends the original OMNIFOLD algorithm to simultaneously update the nuisance parameter θ while iteratively reweighting the simulation data.

Related work includes the approach of Chan and Nachman (2023), which also performs unbinned unfolding with nuisance parameter profiling. Their method uses neural networks to directly maximize the Poisson log-likelihood function. While a significant step forward, this approach is limited to the case where the particle-level data are unbinned but the detector-level data are binned so that one can write down the explicit Poisson likelihood for bin counts. On the other hand, POF is completely unbinned at both the detector and particle levels, and does not assume any parametric model.

3.1. *Algorithm.*

As in OMNIFOLD, let $\nu(x)$ be a reweighting function on the MC particle-level density $q(x)$. Also, assume $q(y|x)$ is specified by the nuisance parameter $\bar{\theta}$, i.e., $q(y|x) = p(y|x, \bar{\theta})$. Moreover, let $w(y, x, \theta)$ be a reweighting function on the MC response kernel $q(y|x)$, i.e., $w(y, x, \theta) = p(y|x, \theta)/q(y|x)$. Then the goal is to maximize the population log-likelihood

$$
\begin{aligned}
&\ell(\nu, \theta) = \int p(y) \log p(y|\nu, \theta)dy + \log p_0(\theta) \\
&\text{subject to } \int \nu(x)q(x)dx = 1,
\end{aligned} \tag{20}
$$

where $p_0(\theta)$ is a prior on θ to constrain the nuisance parameter, usually determined from auxiliary measurements. This is not strictly a Bayesian prior, but rather can be viewed as an optional likelihood term for the auxiliary measurements (Cranmer, 2015). One example is the Gaussian likelihood, $\log p_0(\theta) = -\frac{(\theta-\bar{\theta})^2}{2\sigma_0^2}$.

The POF algorithm, like the original OF algorithm, is an EM algorithm. It iteratively updates the reweighting function $\nu(x)$ and nuisance parameter θ toward the maximum likelihood estimate. For the log-likelihood specified in (20), the Q function is given by

(21)

$$\begin{aligned} Q(\nu,\theta|\nu^{(k)},\theta^{(k)}) &= \int p(y)\int p(x|y,\nu^{(k)},\theta^{(k)})\log p(x,y|\nu,\theta)dxdy + \log p_0(\theta) \\ &= \int p(y)\int \frac{w(y,x,\theta^{(k)})q(y|x)\nu^{(k)}(x)q(x)}{\int w(y,x',\theta^{(k)})q(y|x')\nu^{(k)}(x')q(x')dx'}\log p(x,y|\nu,\theta)dxdy + \log p_0(\theta) \\ \text{subject to } & \int \nu(x)q(x)dx = 1. \end{aligned}$$

The E-step in the EM algorithm is to compute the Q function and M-step is to maximize over ν and θ. The maximizer will then be used as the updated parameter values in the next iteration. Specifically, in the k^{th} iteration, we obtain the update $(\nu^{(k+1)},\theta^{(k+1)})$ by solving $(\nu^{(k+1)},\theta^{(k+1)}) = \arg\max_{\nu,\theta} Q(\nu,\theta|\nu^{(k)},\theta^{(k)})$. This optimization problem can be solved separately for ν and θ, which is described by Proposition 2 below.

PROPOSITION 2. Let

$$\nu^{(k+1)}(x) = \nu^{(k)}(x)\int \frac{p(y)}{\tilde{q}^{(k)}(y)} w(y,x,\theta^{(k)})q(y|x)dy, \tag{22}$$

(23)

$$\theta^{(k+1)} = \arg\max_{\theta}\left[\int\int q(x,y)\nu^{(k)}(x)w(y,x,\theta^{(k)})\frac{p(y)}{\tilde{q}^{(k)}(y)}\log[w(y,x,\theta)]dxdy + \log p_0(\theta)\right],$$

where $\tilde{q}^{(k)}(y) = \int w(y,x',\theta^{(k)})\nu^{(k)}(x')q(x',y)dx'$. Then $(\nu^{(k+1)},\theta^{(k+1)}) = \arg\max_{\nu,\theta} Q(\nu,\theta|\nu^{(k)},\theta^{(k)})$ subject to the constraint that $\int \nu(x)q(x)dx = 1$.

The proof of the proposition is provided in the supplementary material.

REMARK 1. The reason that the optimization problem can be solved separately for ν and θ is that the Q function is separable in terms of ν and θ. Specifically, we can write

$$\begin{aligned} Q(\nu,\theta|\nu^{(k)},\theta^{(k)}) &= \int p(y)\int p(x|y,\nu^{(k)},\theta^{(k)})\log[\nu(x)q(x)q(y|x)]dxdy \\ &\quad + \int p(y)\int p(x|y,\nu^{(k)},\theta^{(k)})\log[w(y,x,\theta)]dxdy + \log p_0(\theta) \\ &= Q_1(\nu|\nu^{(k)},\theta^{(k)}) + Q_2(\theta|\nu^{(k)},\theta^{(k)}). \end{aligned} \tag{24}$$

Therefore, Equations (22) and (23) correspond to maximizing Q_1 and Q_2 separately.

More concretely, the POF algorithm can be implemented via the following three steps in each iteration:

1. Detector-level reweighting:
   $r^{(k)}(y) = \frac{p(y)}{\tilde{q}^{(k)}(y)}$, where $\tilde{q}^{(k)}(y) = \int w(y,x',\theta^{(k)})\nu^{(k)}(x')q(x',y)dx'$.
2. Particle-level reweighting:
   $\nu^{(k+1)}(x) = \nu^{(k)}(x)\frac{\tilde{q}^{(k)}(x)}{q(x)}$, where $\tilde{q}^{(k)}(x) = \int w(y',x,\theta^{(k)})r^{(k)}(y')q(x,y')dy'$.
3. Nuisance parameter update:
   $\theta^{(k+1)} = \arg\max_{\theta} Q_2(\theta|\nu^{(k)},\theta^{(k)})$.

The first step is almost identical to the first step in the original OF algorithm, which involves computing the ratio of the detector-level experimental density and reweighted detector-level MC density using the push-forward weights of $w(y, x, \theta^{(k)})\nu^{(k)}(x)$. The difference from OF is the presence of the additional term $w(y, x, \theta^{(k)})$, which is the ratio of the response kernels parametrized by θ. As in the original OmniFold, the density ratio $r^{(k)}(y)$ can still be estimated by training a classifier to distinguish between the experimental data distribution $p(y)$ and the reweighted MC distribution $\tilde{q}^{(k)}(y)$.

The second step also closely mirrors the second step of the original OF algorithm, which involves computing the ratio of the reweighted particle-level MC density using the pull-back weights $w(y, x, \theta^{(k)})r^{(k)}(y)$ and the particle-level MC density.

The third step updates the nuisance parameter θ by numerically optimizing the Q_2 function. Several strategies are possible: one can directly optimize the Q_2 function or alternatively solve for its stationary condition. The key aspect is that the Q_2 function as well as its gradient with respect to θ are computable up to a constant for different values of θ, which makes the optimization feasible. More details are given in Section 3.2.

In summary, the POF algorithm extends the original OF iteration by introducing an additional step for updating the nuisance parameter. This extension offers several advantages. First, as in OF, the key quantities estimated throughout the procedure are density ratios, which can be efficiently learned via classifiers without estimating each density separately. Second, POF preserves the EM structure, which guarantees that the likelihood is non-decreasing at each iteration under infinite sample size and optimal classifier. Finally, the first two steps closely resemble the OF update, making the algorithm easy to implement as an extension of existing software. An overview of the POF algorithm is illustrated in Figure 1.

However, unlike OF, POF does not have a convergence guarantee since the likelihood function is generally not concave when there are nuisance parameters. Empirically, we observe that the algorithm can converge to different solutions depending on the initialization of the nuisance parameter $\theta^{(0)}$, raising the question of which solution should be selected.

To address this issue, we propose to use a goodness-of-fit statistic based on the weighted accuracy of the step-1 classifier. Recall that in Step 1, a classifier is trained to estimate the density ratio $r^{(k)}(y) = \frac{p(y)}{\tilde{q}^{(k)}(y)}$. If the iterative procedure converges to the correct solution, the reweighted distribution $\tilde{q}^{(k)}(y)$ should match the observed distribution $p(y)$. Consequently, the classifier will be trying to discriminate two distributions that are nearly identical. In that case, the classifier's accuracy should approach 0.5. Based on this observation, we propose the following goodness-of-fit statistic:

$$V = 1 - 2 \cdot \left| \frac{\sum_{i=1}^{n} w_i \mathbb{1}\{\hat{c}_i = c_i\}}{\sum_{i=1}^{n} w_i} - 0.5 \right| \tag{25}$$

where w_i is the detector-level weight assigned to i^{th} observation, c_i is the true class label, and $\hat{c}_i$ is the predicted label. Given b candidate solutions $(\hat{\nu}_1, \hat{\theta}_1), ..., (\hat{\nu}_b, \hat{\theta}_b)$ initialized from different starting points, we select the solution $(\hat{\nu}_{i^*}, \hat{\theta}_{i^*})$ with the highest statistic, i.e., $i^* = \arg\max_i V_i$. Currently, V is a heuristic statistic which depends on the classifier's validation accuracy. In practice, a good solution yields V close to 1, whereas values substantially below 1 indicate a potential mismatch at the detector level. However, we have not yet established a principled investigation of its statistical properties, which we leave for future work.

3.2. *More details on nuisance parameters update.*

During each iteration, the nuisance parameters are updated according to Eq. (23). Note that in the equation, $w(y, x, \theta^{(k)}), \nu^{(k)}(x)$ were already computed in the previous iteration, $\frac{p(y)}{\tilde{q}^{(k)}(y)}$ is the density ratio being estimated by the step-1 classifier, and $\log p_0(\theta)$ is known. The only unknown term is $\log[w(y, x, \theta)]$.

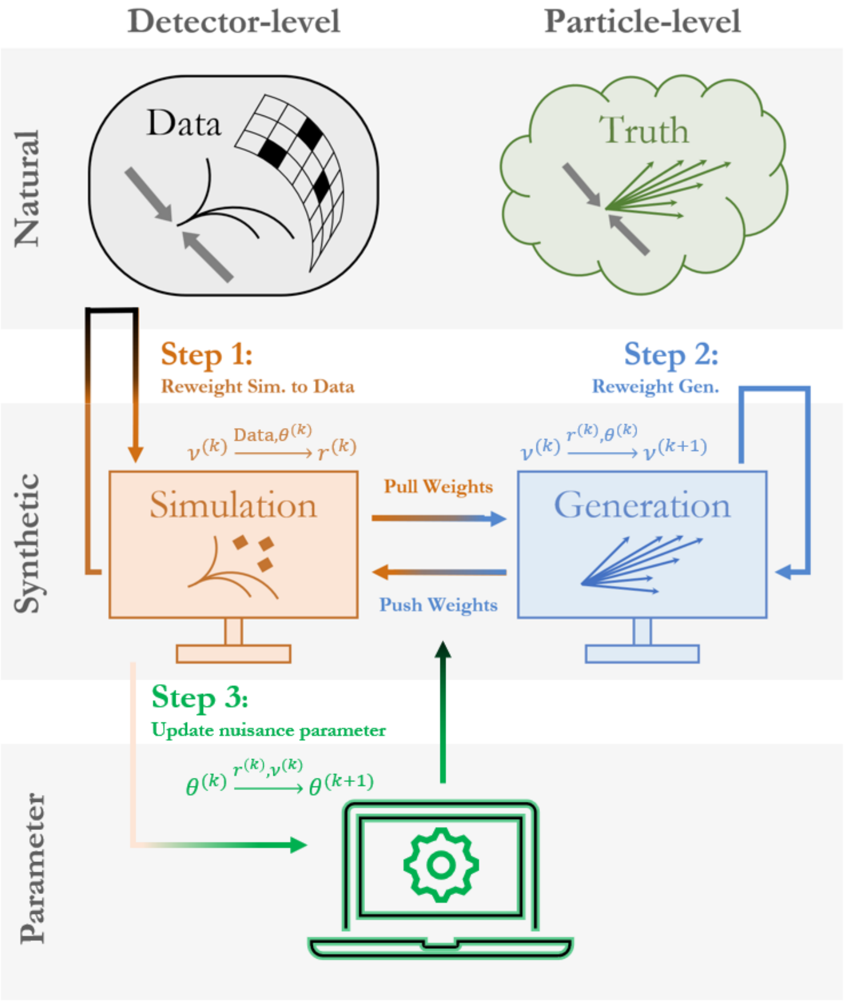
Detector-level
Particle-level
Natural
Data
Truth
Step 1:
Reweight Sim. to Data
Step 2:
Reweight Gen.
Synthetic
$\nu^{(k)} \xrightarrow{\text{Data},\theta^{(k)}} r^{(k)}$
$\nu^{(k)} \xrightarrow{r^{(k)},\theta^{(k)}} \nu^{(k+1)}$
Simulation
Pull Weights
Generation
Push Weights
Step 3:
Update nuisance parameter
$\theta^{(k)} \xrightarrow{r^{(k)},\nu^{(k)}} \theta^{(k+1)}$
Parameter

FIG 1. *An overview of the POF algorithm. Portions of the image have been adapted from Andreassen et al. (2020) for the original* OMNIFOLD *algorithm. In step 1, the current particle-level weights $\nu^{(k)}$ are pushed to the detector level with the current nuisance parameters $\theta^{(k)}$, which are used to compute the density ratio $r^{(k)}$. In step 2, the ratio $r^{(k)}$ is pulled pack to the particle level using the same nuisance parameters. In step 3, the nuisance parameters are updated based on the current weights $\nu^{(k)}$ and the density ratio $r^{(k)}$. The procedure is iterated for a fixed number of times.*

Combined with the other quantities, this term is integrated with respect to the joint density $q(x, y)$, which can be approximated by a sample average over the MC data. Therefore, the only challenge lies in the estimation of the function $w(y, x, \theta)$. Recall that $w(y, x, \theta)$ is defined as the ratio of two conditional densities $\frac{p(y|x,\theta)}{q(y|x)}$. Chan and Nachman (2023) proposed to learn $w(y, x, \theta)$ by factorizing it into the product of two density ratios

$$w(y, x, \theta) = \frac{p(y|x, \theta)}{q(y|x)} = \frac{p(x, y|\theta)}{q(x, y)} \cdot \frac{q(x)}{p(x|\theta)}, \tag{26}$$

where $\frac{p(x,y|\theta)}{q(x,y)}$ and $\frac{q(x)}{p(x|\theta)}$ can be estimated separately using the classifiers. To learn this function, an additional set of synthetic data $\{X_i, \theta_i, Y_i\}$[1] is required, where the density of Y_i is given by $p_{Y_i}(y) = \int p(y|x, \theta_i) p_{X_i}(x) dx$. In this setup, the choice of the distribution for X_i is flexible, as long as its support covers the data domain. One practical option is to use particle-level MC samples for X_i and generate the corresponding Y_i by applying forward models parametrized by different θ_i. Proposition 3 formalizes this procedure.

PROPOSITION 3. Let $X_i \sim \mathcal{P}_X, \theta_i \sim \mathcal{P}_\theta, Y_i \sim p(\cdot|X_i, \theta_i)$, and $X_i' \sim \mathcal{Q}_X, \theta_i' \sim \mathcal{Q}_\theta, Y_i' \sim q(\cdot|X_i)$. Let $f_1 : \mathcal{X} \times \mathcal{Y} \times \Theta \to [0,1]$ be the Bayes optimal classifier discriminating dataset $\mathcal{D}_1 = \{X_i, Y_i, \theta_i\}$ from $\mathcal{D}_2 = \{X_i', Y_i', \theta_i'\}$. Let $f_2 : \mathcal{X} \times \Theta \to [0,1]$ be the Bayes optimal classifier discriminating dataset $\tilde{\mathcal{D}}_2 = \{X_i', \theta_i'\}$ from $\tilde{\mathcal{D}}_1 = \{X_i, \theta_i\}$. Then the output of the classifiers satisfy:

$$\frac{f_1(x,y,\theta) f_2(x,\theta)}{(1 - f_1(x,y,\theta))(1 - f_2(x,\theta))} = \frac{p(y|x,\theta)}{q(y|x)}. \tag{27}$$

The proof of the proposition is provided in the supplementary material.

REMARK 2. The procedure outlined in Proposition 3 was also used in Chan and Nachman (2023), although a formal proof was not provided. In this setting, the auxiliary variable θ_i' does not influence the synthetic data X_i', Y_i'; rather, it functions solely as a supporting variable for classifier training. While the distributions of θ_i and θ_i' may differ, as allowed by the proposition, in practice, it might be preferable to set $q_\theta = p_\theta$ to avoid potential numerical instability. A convenient choice is to use a uniform distribution over the parameter space. Similarly, although the proposition specifies $X_i \sim \mathcal{P}_X, X_i' \sim \mathcal{Q}_X$, there is no restriction against using the same distribution for both. Therefore, a practical procedure can be summarized as follows:

1. Sample X_i', X_i from the particle-level MC distribution $q(x)$.
2. Sample θ_i' from a chosen distribution p_θ, e.g., uniform distribution over the parameter space. Sample θ_i from the same distribution p_θ.
3. Generate Y_i' from the forward model $q(y|X_i')$. Generate Y_i from the forward model $p(y|X_i, \theta_i)$.
4. Train classifier f_1 to distinguish $\mathcal{D}_1 = \{X_i, Y_i, \theta_i\}$ from $\mathcal{D}_2 = \{X_i', Y_i', \theta_i'\}$, and train classifier f_2 to distinguish $\tilde{\mathcal{D}}_2 = \{X_i', \theta_i'\}$ from $\tilde{\mathcal{D}}_1 = \{X_i, \theta_i\}$.

4. Simulation Study: Gaussian Example.

In this section, we illustrate the POF algorithm with a simple Gaussian example. Consider a one-dimensional Gaussian distribution at the particle level and two Gaussian distributions at the detector level. The data are generated as follows:

$$\begin{aligned} Y_{i1} &= X_i + Z_{i1}, \\ Y_{i2} &= X_i + Z_{i2}, \end{aligned} \tag{28}$$

where $X_i \sim \mathcal{N}(\mu, \sigma^2), Z_{i1} \sim \mathcal{N}(0,1), Z_{i2} \sim \mathcal{N}(0, \theta^2)$. Here, θ is the nuisance parameter, which only affects the second coordinate of the detector-level data. This is qualitatively similar to the physical case of being able to measure the same quantity twice. This is also an identifiable model since the characteristic function of (Y_1, Y_2) satisfies

$$\varphi_{Y_1,Y_2}(t_1, t_2) = \varphi_X(t_1 + t_2) e^{-\frac{1}{2}(t_1^2 + t_2^2 \theta^2)}, \tag{29}$$

[1] We slightly abuse the notation here and use X_i, Y_i to denote the synthetic data, but this should not be confused with the experimental data.

where φ_X is the characteristic function of X. Since $e^{-\frac{1}{2}(t_1^2+t_2^2\theta^2)} > 0$ for all t_1, t_2, this uniquely determines φ_X, and hence also the distribution of X. Because the response kernel in this case is a Gaussian density, the analytic form of $p(y|x,\theta)$ is known and, consequently, $w(y,x,\theta)$ as well. Thus, we can directly plug in the analytic form of $w(y,x,\theta)$ into the POF algorithm without the need to estimate it using classifiers. As a comparison, we present results both using the analytic form and the estimated w function as described in Section 3.2.

4.1. *Dataset.* Based on the above data-generating process, Monte Carlo data are generated with $\mu = 0, \sigma = 1, \theta = 1$ and experimental data are generated with $\mu = 0.8, \sigma = 1, \theta = 1.5$. We simulate 10^5 events each for the MC data and experimental data.

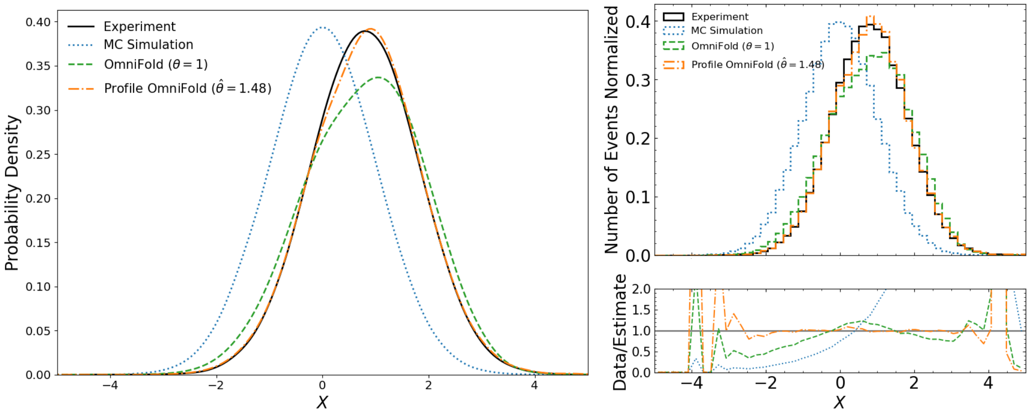

FIG 2. *Results of unfolding a 2D Gaussian example. Analytic w function is being used in the algorithm.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

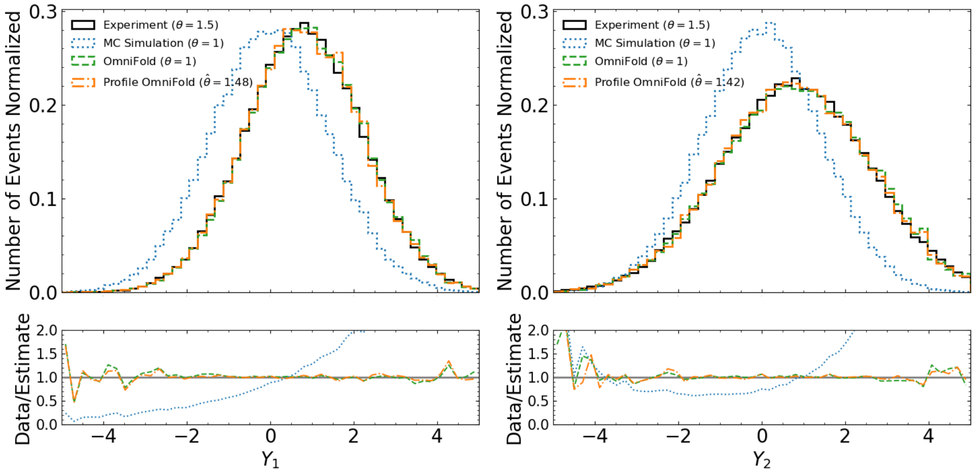

FIG 3. *Results corresponding to Figure 2 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

4.2. *Neural network architecture and training.* The neural network classifier for estimating density ratios during POF iteration is implemented in TensorFlow and Keras (Abadi et al., 2016; Chollet, 2017). The network contains three hidden layers with 50 nodes per layer and employs the ReLU activation function. The output layer consists of a single node with a sigmoid activation function. Training is performed with the Adam optimizer (Kingma and Ba, 2017) with learning rate $\eta = 0.001$ using a weighted binary cross-entropy loss. The model is trained for up to 20 epochs with a batch size of 10,000, and early stopping with a patience of 3 epochs is applied, that is, training stops if the validation loss does not improve for 3 consecutive epochs. None of the hyperparameters were optimized.

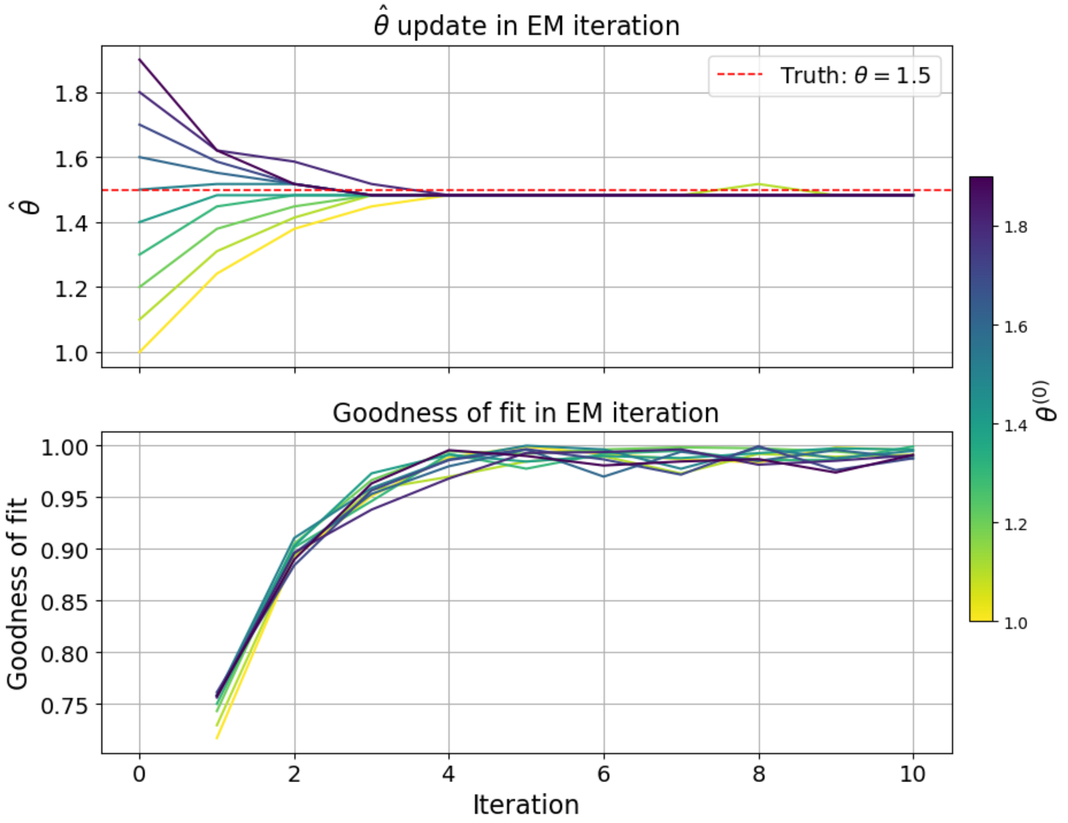

FIG 4. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 2.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

The neural network classifiers for estimating the w function ($w(y,x,\theta) = p(y|x,\theta)/q(y|x)$) are implemented in PyTorch (Paszke et al., 2019).[2] The same architecture is employed for both classifiers in Proposition 3. The network contains three hidden layers with 50 nodes per layer and employs the ReLU activation function. Batch normalization is applied after each hidden layer, and a dropout layer with a rate of 0.1 is added after the second hidden layer.

[2] The reason we used two different frameworks is that the original implementation of OMNIFOLD was in TensorFlow/Keras, while the codebase for learning w function from Chan and Nachman (2023) was in PyTorch. There is no technical reason preventing from using a single framework.

The output layer consists of a single node with a sigmoid activation function. Training uses the Adam optimizer with learning rate $\eta = 0.001$ and the weighted binary cross-entropy loss. This classifier is trained for up to 1000 epochs with a batch size of 10,000, and early stopping with patience 10 is used. In addition, we train an ensemble of 10 networks with bootstrap resampling to reduce the variance of the estimated w function. The range of nuisance parameter θ used in training is set to be $[0.5, 2.0]$.

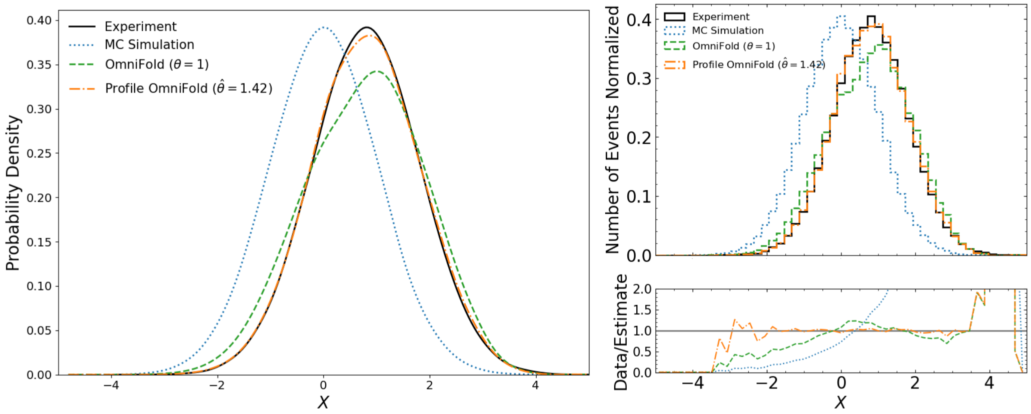

FIG 5. *Results of unfolding a 2D Gaussian example. Estimated w function is being used in the algorithm.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

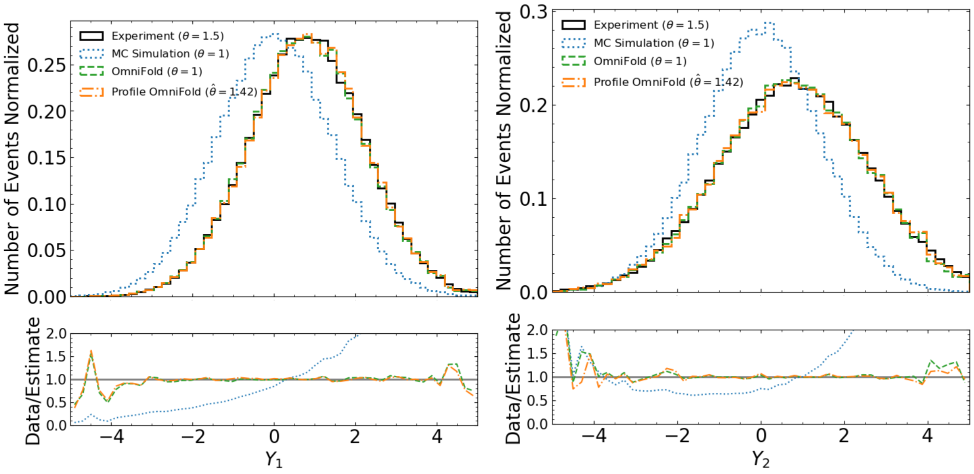

FIG 6. *Results corresponding to Figure 5 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

4.3. *Results.* Figure 2 illustrates the results of unfolding the 2D Gaussian data using both the proposed POF algorithm and the original OF algorithm. In the unbinned solution, kernel density estimates are used to represent the simulation, data, and reweighted distributions, while the binned solution employs histograms with 50 bins. The blue curve is the Monte Carlo distribution for which the reweighting function $\nu(x)$ will be applied. The results show that the original OF solution (green) deviates significantly from the true distribution (black). This discrepancy arises because OF assumes $p(y|x) = q(y|x)$, which is invalid in the present setting. In contrast, the POF algorithm simultaneously updates the nuisance parameter along with the reweighting function. The results show that the unfolded solution (orange) aligns closely with the truth and the estimated nuisance parameter is $\hat{\theta} = 1.48$, which is close to the true value $\theta = 1.50$.

Figure 3 shows the corresponding reweighted detector-level spectra of Y_1 and Y_2. The reweighted spectra are close to the experimental distribution (black) for both POF (orange) and OF (green) solutions. This is expected as both POF and OF are designed to reweight the detector-level MC distribution to match the experimental distribution as closely as possible. However, since the response kernel is not correctly specified in the original OF algorithm, the reweighted particle-level distribution does not match the true particle-level distribution, leading to a poor unfolding result.

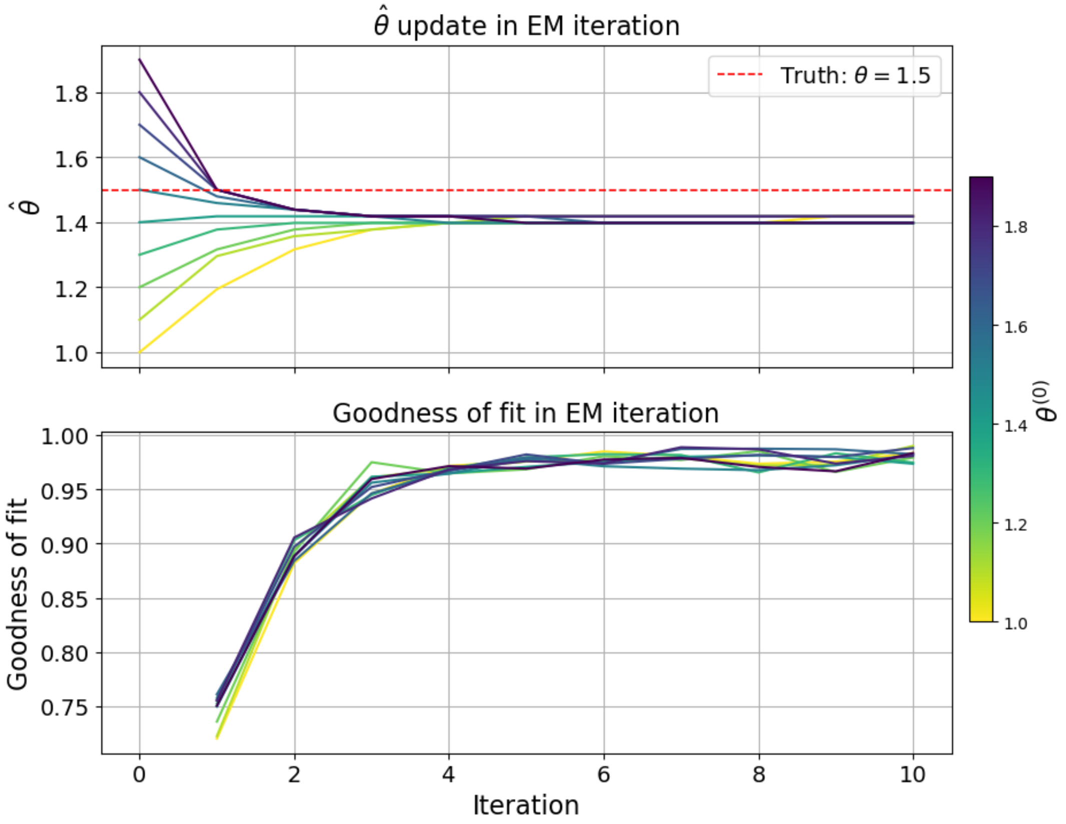

FIG 7. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 5.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

Moreover, Figure 4 shows the evolution of the nuisance parameter θ and the goodness-of-fit statistic for the step-1 classifier, defined in Eq. (25). The top plot shows that the nuisance parameter converges to the true value within a few iterations, regardless of the initial value.

The bottom plot shows the goodness-of-fit statistic converging to 1 for all initial values, indicating that the reweighted distribution $\tilde{q}(y)$ is close to the target distribution $p(y)$. As discussed in Section 3.1, the results shown in Figure 2-3 correspond to the solution with the highest goodness-of-fit statistic, although in this case study all solutions give equally good fits after a few iterations.

Figure 5 illustrates the results obtained using the estimated w function instead of the analytic form. The results show that the POF solution (orange) still aligns closely with the truth (black). However, the estimated nuisance parameter is $\hat{\theta} = 1.42$, which is slightly off from the true value $\theta = 1.50$. Figure 7 shows that the estimated nuisance parameter converges to around 1.42, regardless of the initial value. The goodness-of-fit statistic converges to close to 1 (although slightly worse than the case with analytic w) for all initial values, indicating that the reweighted distribution $\tilde{q}(y)$ is close to the target distribution $p(y)$. Figure 6 shows that the reweighted detector-level spectra of Y_1 and Y_2 are also close to the experimental distribution (black) for both POF (orange) and OF (green) solutions.

In practice, we have observed that the estimated nuisance parameter is sensitive to the quality of the fitted w function. Even small changes in classifier training, such as a different range for simulating θ_i, or simply training two neural networks with the same hyperparameters could lead to different estimates of the nuisance parameters. Such variability can be reduced by training an ensemble of neural networks, but it still remains a practical challenge when applying the algorithm. Nevertheless, the final unfolded density, which is the primary quantity of interest, appears to be relatively robust to variations in the fitted w function.

We have also experimented with different values of the true nuisance parameter θ in the data-generating process, and the results are qualitatively similar to those presented here. The details are included in the supplementary material.

5. Simulated Public Collider Data by the CMS Experiment. We now study the important process of generic quark and gluon scattering at high-energy particle colliders. The outgoing particles radiate and produce collimated streams of particles called jets. The inclusive jet energy spectrum is useful for studies of the strong force at small distance scales, for searches for new, fundamental interactions, and for developing physics models that enable other analyses with jets as a background process. The most likely type of event consists of two, high-energy jets that have nearly the same momentum transverse to the collision axis due to conservation of momentum. The detector acts locally in space, so the measured momenta of the two jets are independently smeared. The amount of smearing is approximately known from simulations. These simulations and their data-based calibrations contain a number of nuisance parameters. In our example, we are only sensitive to the effective jet energy resolution, which governs the overall amount of momentum smearing. We introduce a nuisance parameter that is a multiplicative factor determining how much the jet energy resolution in data differs from simulation. Due to the symmetries of the problem, the difference in the jet momenta is particularly sensitive to this nuisance parameter without being sensitive to the underlying momentum spectrum. Thus, we measure the joint distribution of the sum and difference in jet momenta per event while simultaneously constraining the nuisance parameter.

5.1. *Dataset.* To demonstrate this setup in practice, we use high-fidelity simulations from the CMS Experiment at the Large Hadron Collider. In particular, the data were generated with PYTHIA 6.426 (Sjöstrand, Mrenna and Skands, 2006) using the Z2 tune (Chatrchyan et al., 2011) and interfaced with a GEANT4-based (Agostinelli et al., 2003; Allison et al., 2006; Allison et al., 2016) detailed detector simulation of the CMS experiment (Chatrchyan et al., 2008). This dataset comes from the CMS Open Data Portal (CMS collaboration, 2016a; CMS Collaboration, 2016b,c) and is processed into an MIT Open Data format (Komiske et al., 2019a,b,c, 2020). We use the events as both "simulation" and "data" in order to have a known target for testing.

The datasets from this collection are sorted by the parton-level hard-scattering scale $\hat{p}_T$ from PYTHIA, which is in general different from the jet-level transverse momentum p_T we are interested in studying. For simplicity, we consider one slice of the collection with 600 GeV $< \hat{p}_T <$ 800 GeV. This slice corresponds to sufficiently high momentum jets that effects from the triggering system are not relevant. Particles (at truth level) or particle flow candidates (at reconstructed/detector level) are used as inputs to jet clustering, implemented using FASTJET 3.2.1 (Cacciari, Salam and Soyez, 2012; Cacciari and Salam, 2006) and the anti-k_t algorithm (Cacciari, Salam and Soyez, 2008) with radius parameter $R = 0.5$. The total number of events is 43,892. We consider the two highest p_T jets at both particle (truth) and detector (reconstructed) levels, denoted as $p_{T,1}^{\text{truth}}, p_{T,2}^{\text{truth}}, p_{T,1}^{\text{reco}}, p_{T,2}^{\text{reco}}$. Then our observables are

$$\begin{aligned} Y_1 &= p_{T,1}^{\text{truth}} + \theta(p_{T,1}^{\text{reco}} - p_{T,1}^{\text{truth}}) + p_{T,2}^{\text{truth}} + \theta(p_{T,2}^{\text{reco}} - p_{T,2}^{\text{truth}}), \\ Y_2 &= p_{T,1}^{\text{truth}} + \theta(p_{T,1}^{\text{reco}} - p_{T,1}^{\text{truth}}) - p_{T,2}^{\text{truth}} - \theta(p_{T,2}^{\text{reco}} - p_{T,2}^{\text{truth}}), \end{aligned} \tag{30}$$

and the target quantity is

$$X = p_{T,1}^{\text{truth}} + p_{T,2}^{\text{truth}}. \tag{31}$$

Here θ adjusts the jet energy resolution in data relative to simulation. The true value of θ in data is 1.7 while the nominal value in simulation is 1.0, indicating that the jet energy resolution in data is 70% larger than that in simulation. To induce a mismatch in the marginal distribution between the simulation and the data, we construct the MC sample by applying weighted sampling to the events.

5.2. *Neural network architecture and training.* The neural network architecture and training procedure used in the POF algorithm follow the same setup as in the Gaussian example. For training the w function, the same architecture is employed for both classifiers as in Section 4.2. The only difference is that the f_1 classifier is trained with an early-stopping patience of 30 epochs, as we found that it requires more iterations to converge in practice. The range of nuisance parameter θ used in training is set to be $[0.5, 2.0]$.

5.3. *Results.* Figure 8 presents the unfolded CMS Open Data results obtained using both the proposed POF algorithm and the original OF algorithm. In the unbinned solution, kernel density estimates are used to represent the simulation, data, and reweighted distributions, while the binned solution employs histograms with 50 bins. Consistent with the Gaussian example, the original OF solution (green) deviates substantially from the truth (black) because it assumes the nuisance parameter is fixed and correctly specified at $\theta = 1.0$. In contrast, the POF solution (orange) closely matches the truth, with the estimated nuisance parameter of $\hat{\theta} = 1.62$ (true value $\theta = 1.7$).

Figure 9 shows the corresponding reweighted detector-level spectra for Y_1 and Y_2. Although the reweighted distributions for both POF (orange) and OF (green) generally follow the experimental data (black), some noticeable discrepancies appear—for example, around the peak of the Y_1 distribution and near zero in the Y_2 distribution. These deviations are plausibly attributed to mismatches in support between the experimental and MC detector-level distributions, which can induce large or unstable weights. Notably, this behavior is not specific to the POF approach and is also observed in the original OF algorithm. Nevertheless, the unfolded distribution obtained via POF remains in close agreement with the truth, indicating that the impact of these weight instabilities on the final unfolded solution is limited.

Moreover, Figure 10 shows that the nuisance parameter does not necessarily converge to a good estimate. In particular, when initialized at $\theta^{(0)} = 1.0$ or 1.1, the estimated value stabilizes around $\hat{\theta} \approx 1.35$. In these cases, the corresponding goodness-of-fit statistic also

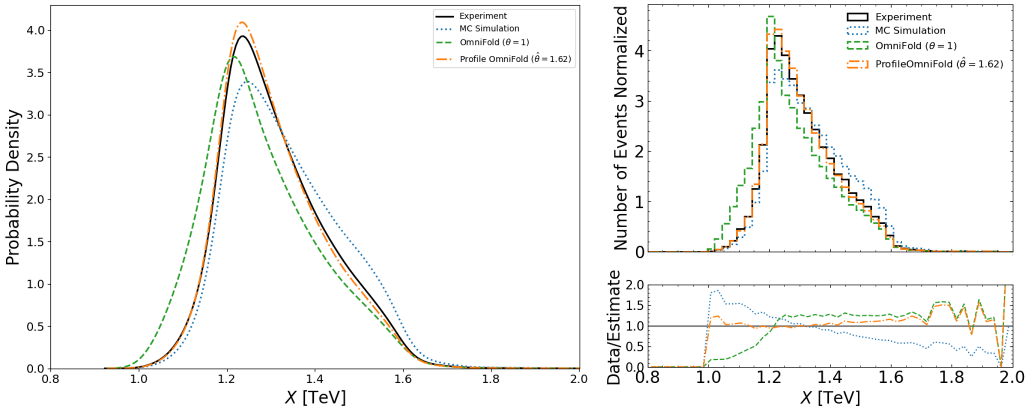

FIG 8. *Results of unfolding the CMS open data.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

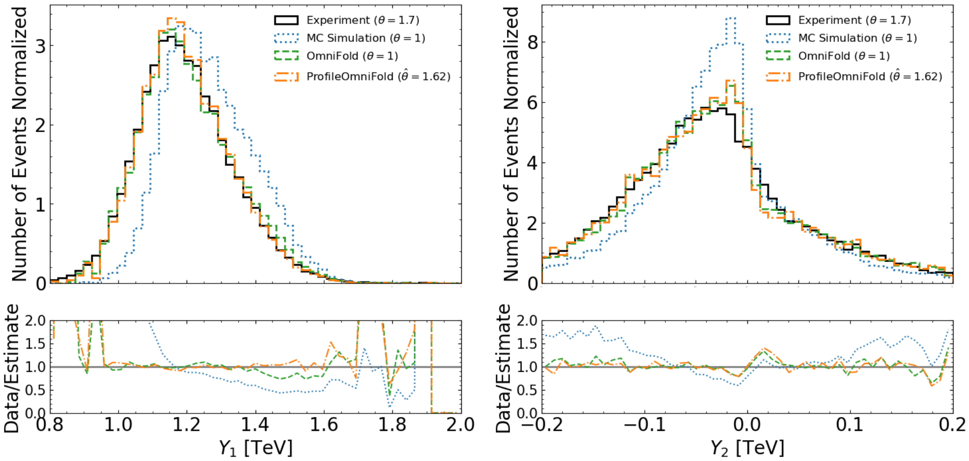

FIG 9. *Results corresponding to Figure 8 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

fails to approach 1, indicating that the reweighted distribution $\tilde{q}(y)$ does not adequately match the observed distribution $p(y)$. This behavior suggests that the algorithm may converge to a local, rather than global, maximum of the likelihood if the starting value $\theta^{(0)}$ is far from the optimal value. The results reported in Figures 8-9 are obtained by selecting, among all runs, the solution achieving the highest goodness-of-fit statistic, thereby avoiding this issue. Additional experiments with different jet energy resolution are provided in the supplementary material. These experiments yield consistent results: POF reliably recovers the true particle-level distribution, whereas OF fails in the presence of a misspecified nuisance parameter. However, the choice of the initial nuisance parameter remains important for POF, as starting too far from the true value can lead to suboptimal estimates. For both OF and POF, the reweighted detector-level distributions may also fail to perfectly match the experimental data, particularly when the discrepancy between the true θ and the nominal θ used in simulation is large. These findings highlight the importance of employing multiple initializations and selecting the best solution based on the goodness-of-fit statistic.

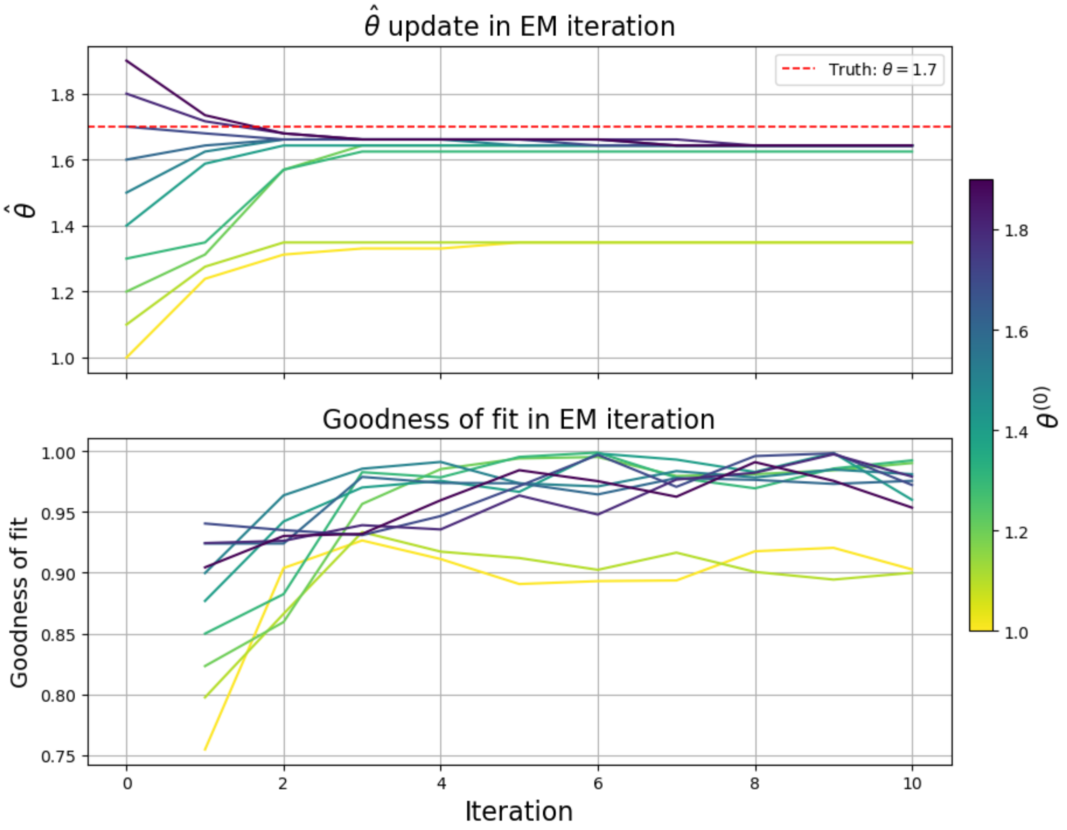

FIG 10. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 8.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

6. Discussion. In this paper, we have proposed a new simulation-based unfolding algorithm, Profile OMNIFOLD, which extends the original OMNIFOLD algorithm to the case where the forward model is not completely specified. POF iteratively updates the reweighting function and nuisance parameters to maximize the population log-likelihood. The algorithm is an EM algorithm that shares similar steps as in OMNIFOLD, which allows for easy implementation while preserving many of its benefits, such as being able to use machine-learning classifiers to estimate density ratios during the iteration.

The results from the simple Gaussian example and an open dataset from the CMS Experiment demonstrate the algorithm's promising performance. In the case of an incorrectly specified forward model, the POF algorithm is able to accurately estimate the true particle-level distribution, whereas the original OMNIFOLD algorithm fails.

One limitation for the current method is the requirement of training the w function, which is the conditional density ratio of the smearing kernel parametrized by the nuisance parameter and the Monte Carlo simulation, i.e., $w(y,x,\theta) = p(y|x,\theta)/q(y|x)$. We found empirically that the estimate nuisance parameters are rather sensitive to the estimated w function. Different training configurations, such as the number of epochs, early stopping, and range of the nuisance parameter for the training data, can affect the convergence of the nuisance parameter and the final unfolding results. One direction for future work is to explore potential ways to improve the robustness of the w function estimation, or even to avoid the need of estimating the w function altogether.

Another important direction for future work is uncertainty quantification of the unfolding results. The current POF algorithm does not provide uncertainty estimates for either the unfolded distribution or the nuisance parameters. Even in the original OMNIFOLD algorithm, it is unclear how to propagate the uncertainty in the classifier-based density ratio estimates to the final unfolded result. One possible (computationally expensive) solution is to use bootstrap resampling (Canelli et al., 2025) to estimate the uncertainty of the unfolded distribution and nuisance parameters. However, rigorous analysis of uncertainty quantification still remains an open problem in this setting.

Acknowledgments. We thank Wahid Bhimji for co-hosting RZ during his internship at LBNL. We acknowledge the computing resources provided by National Energy Research Scientific Computing Center (NERSC) for performing the experiments in this manuscript. We thank members of the Statistical Methods for the Physical Sciences (STAMPS) Research Center at CMU for insightful discussions and feedback on this work.

Funding. BN, KD, and RZ were supported by the U.S. Department of Energy (DOE), Office of Science under contracts DE-AC02-05CH11231 and DE-AC02-76SF00515. VM was supported by JST EXPERT-J, Japan Grant Number JPMJEX2509. MK, LW and RZ were supported by NSF grants DMS-2053804 and DMS-2310632.

SUPPLEMENTARY MATERIAL A: ADDITIONAL EXPERIMENTAL RESULTS

A.1. Gaussian Data. In this section, we include additional results for Gaussian data with different nuisance parameters. The nuisance parameter θ takes values in $\{0.6, 0.8, 1.2, 1.4\}$.

1. $\theta = 0.6$

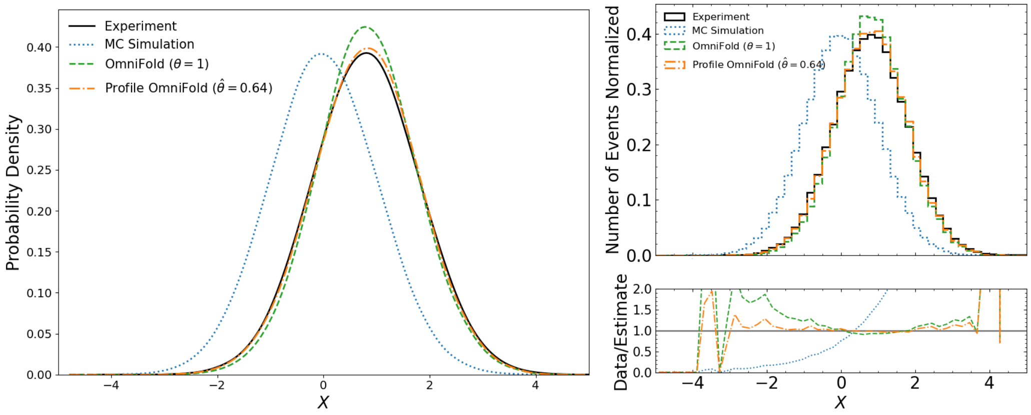

FIG 11. *Results of unfolding a 2D Gaussian example. Analytic w function is being used in the algorithm.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

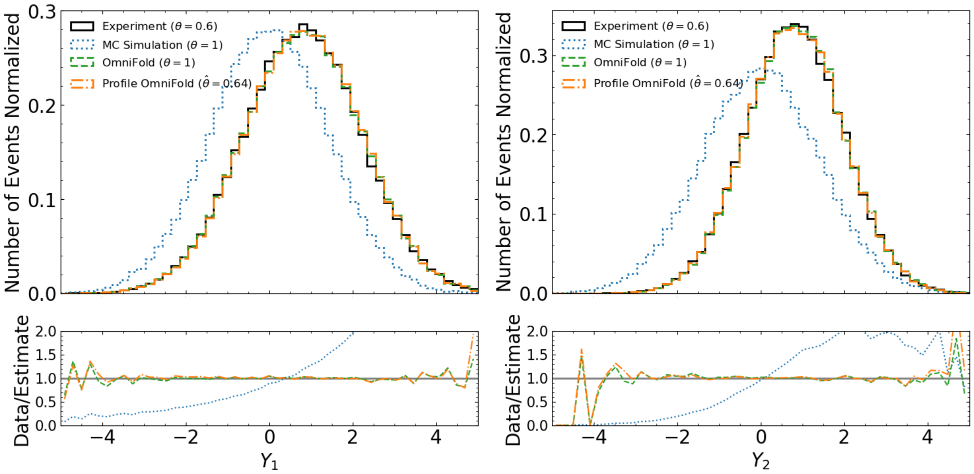

FIG 12. *Results corresponding to Figure 11 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

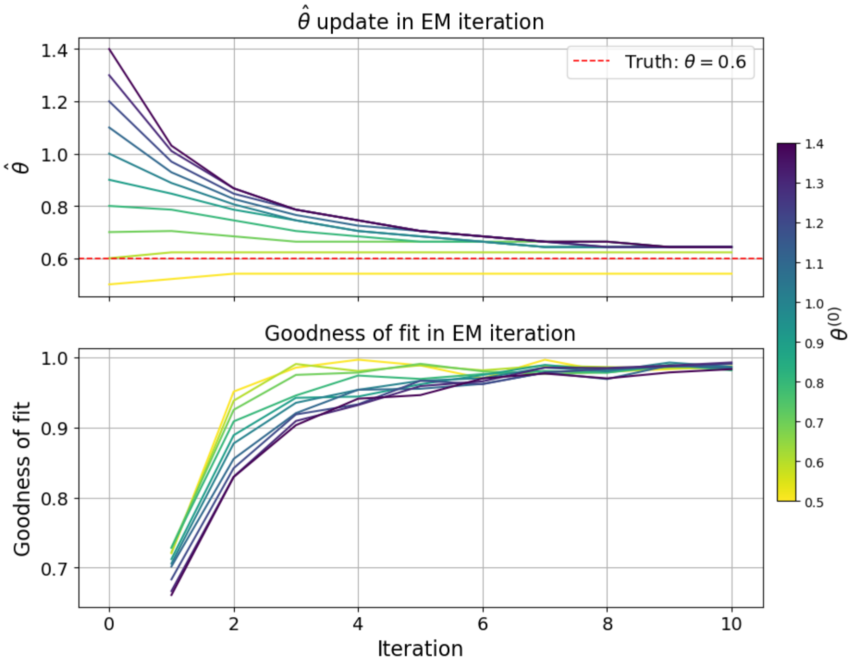

FIG 13. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 11.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

2. $\theta = 0.8$

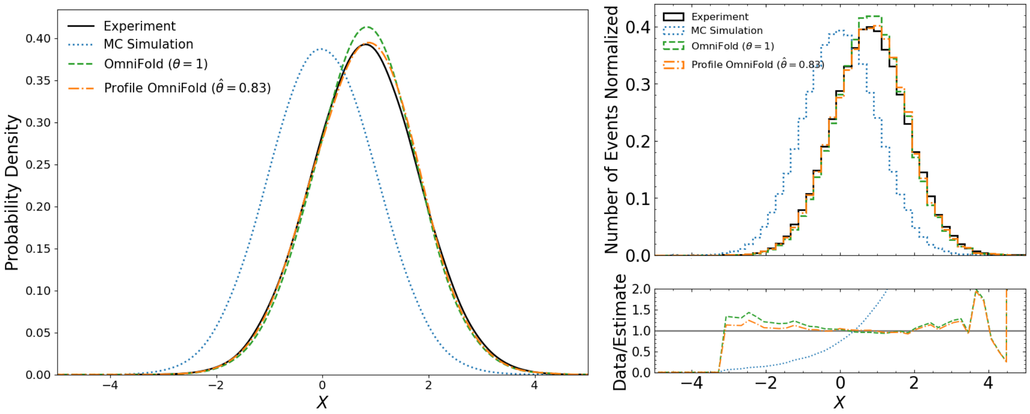

FIG 14. *Results of unfolding a 2D Gaussian example. Analytic w function is being used in the algorithm.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

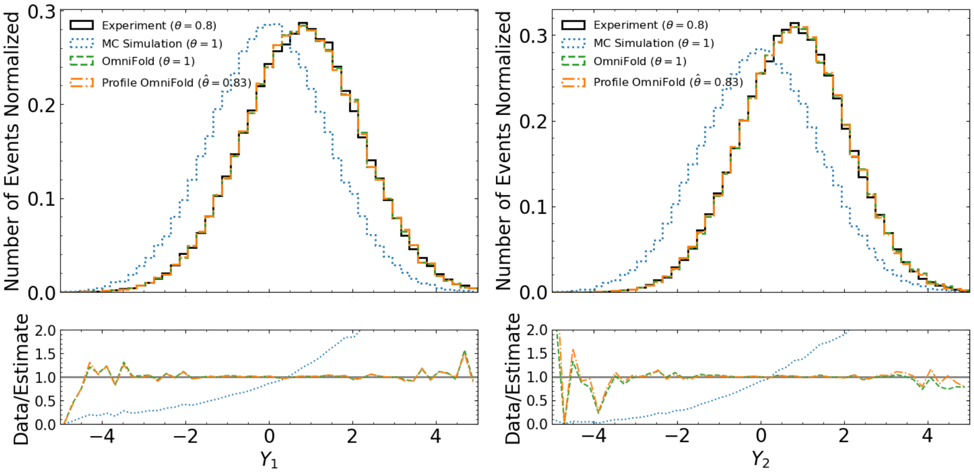

FIG 15. *Results corresponding to Figure 14 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

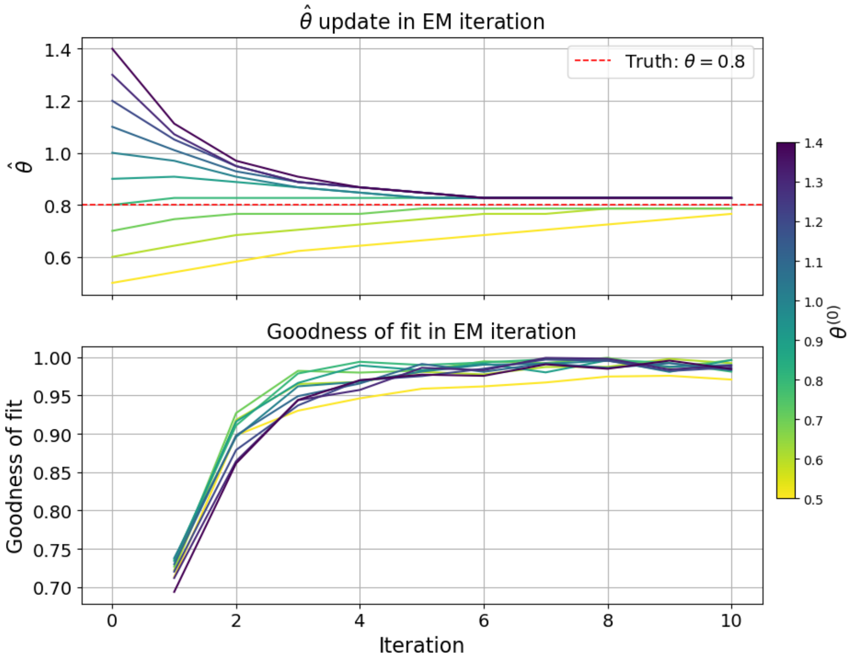

FIG 16. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 14. **Top**: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$. **Bottom**: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

3. $\theta = 1.2$

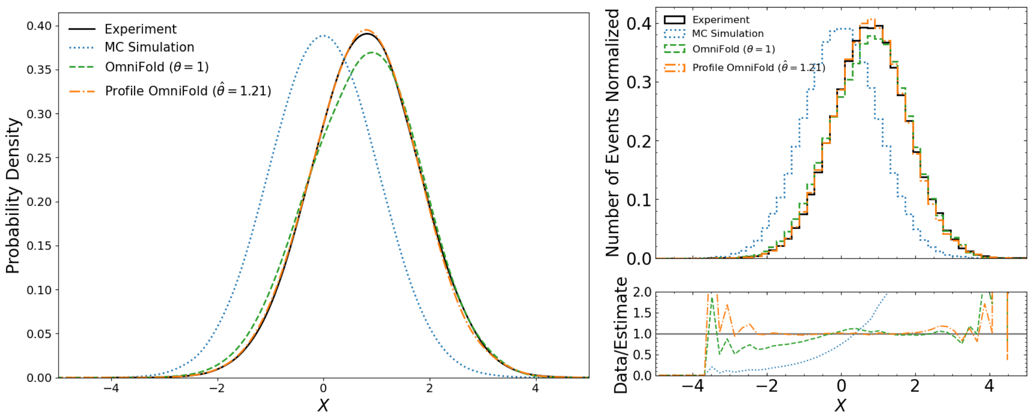

FIG 17. *Results of unfolding a 2D Gaussian example. Analytic w function is being used in the algorithm. **Left**: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations. **Top-right**: Histograms of the four corresponding spectra, aggregated into 50 bins. **Bottom-right**: The ratio of the truth spectrum to the unfolded spectra.*

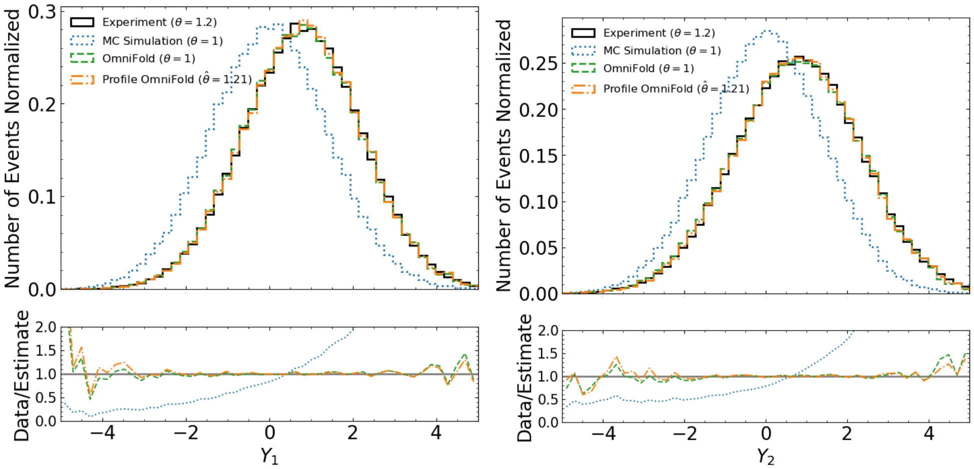

FIG 18. *Results corresponding to Figure 17 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

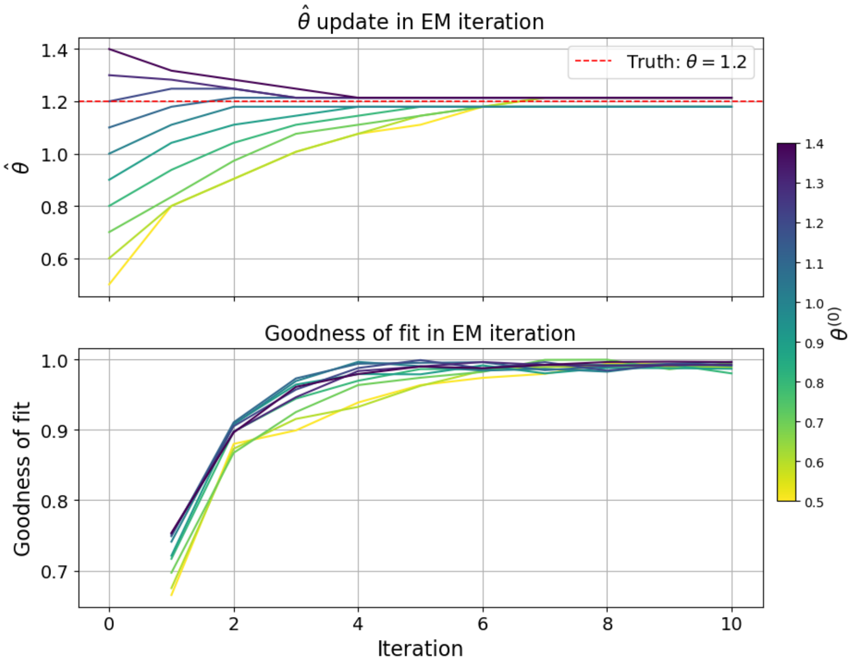

FIG 19. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 17.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

4. $\theta = 1.4$

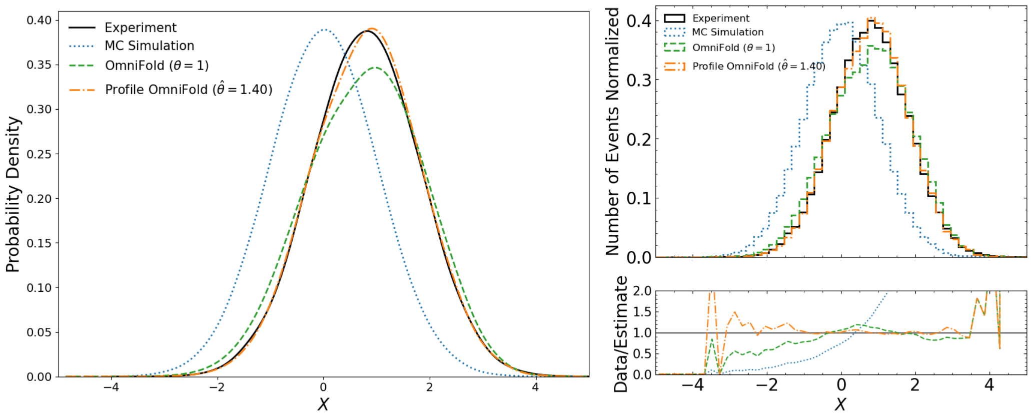

FIG 20. *Results of unfolding a 2D Gaussian example. Analytic w function is being used in the algorithm.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

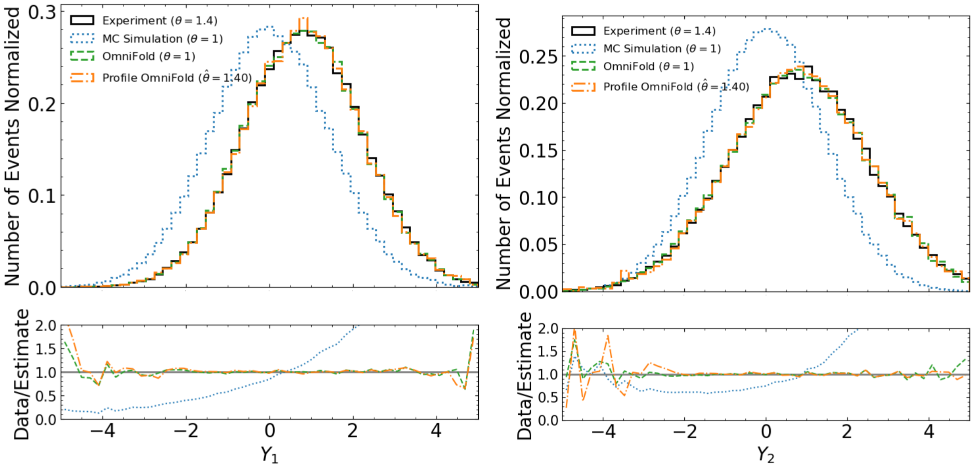

FIG 21. *Results corresponding to Figure 20 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

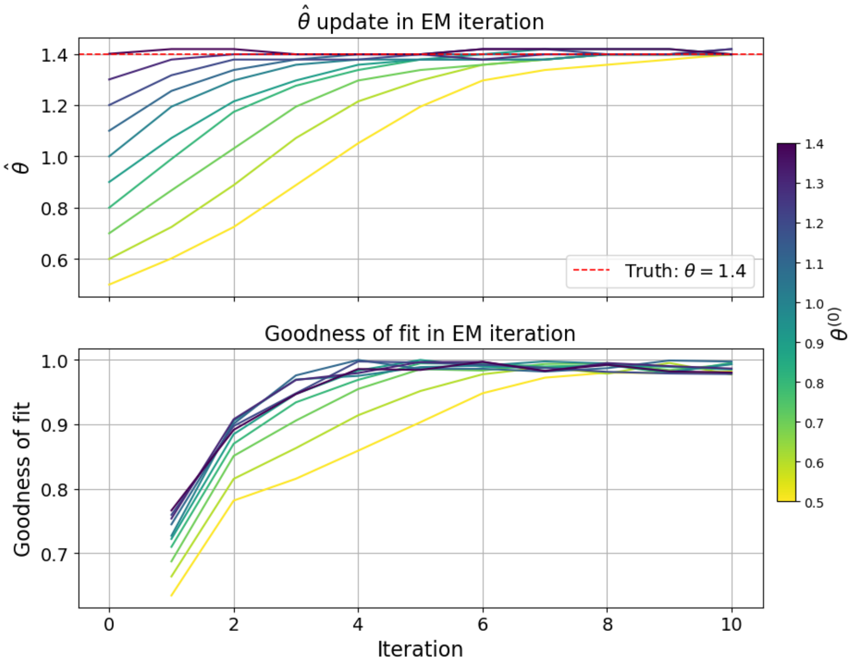

FIG 22. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 20.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

A.2. CMS Open Data. In this section, we include extended empirical results for the CMS open data. The nuisance parameter θ takes values in $\{0.6, 0.8, 1.2, 1.4\}$. The range of nuisance parameter θ used in training w function is set to be $[0.5, 1.5]$.

1. $\theta = 0.6$

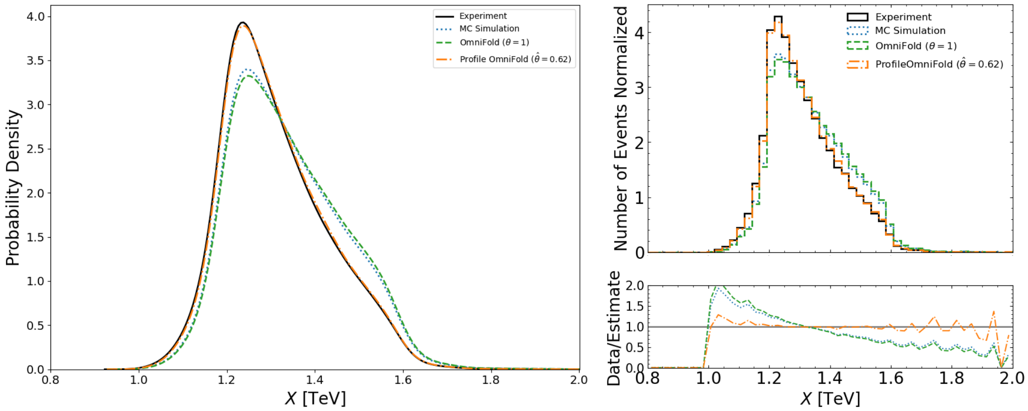

FIG 23. *Results of unfolding the CMS open data.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

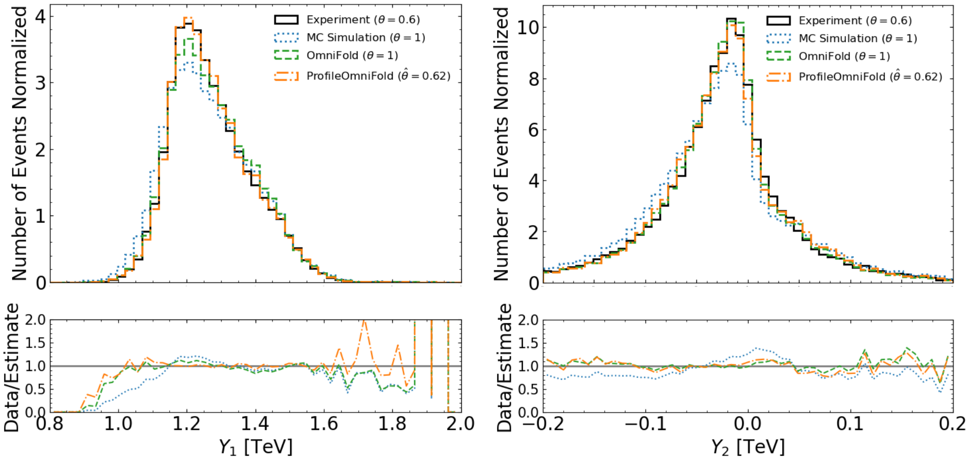

FIG 24. *Results corresponding to Figure 23 in detector-level space.* ***Left****: Histograms of the corresponding spectra of* Y_1. ***Right****: Histograms of the corresponding spectra of* Y_2.

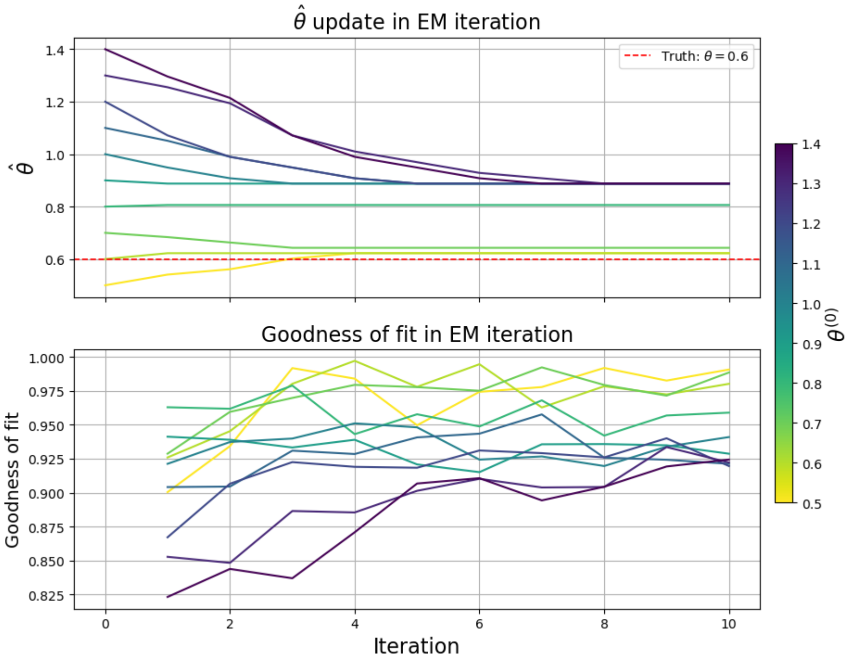

FIG 25. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 23.* ***Top****: Updated estimates* $\hat{\theta}$ *across iterations for different initializations* $\theta^{(0)}$. ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

2. $\theta = 0.8$

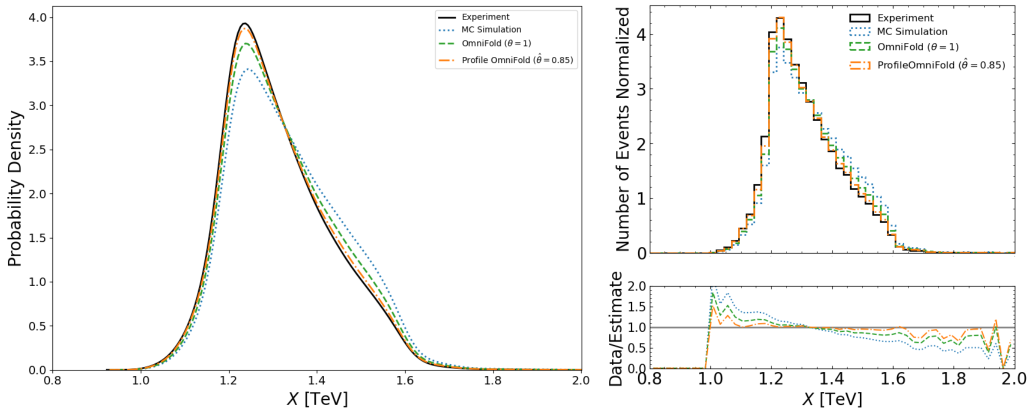

FIG 26. *Results of unfolding the CMS open data.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

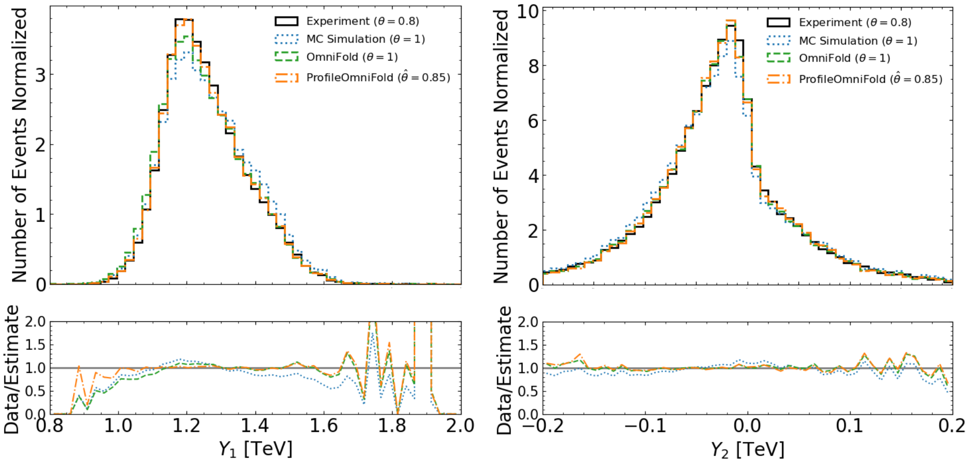

FIG 27. *Results corresponding to Figure 26 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

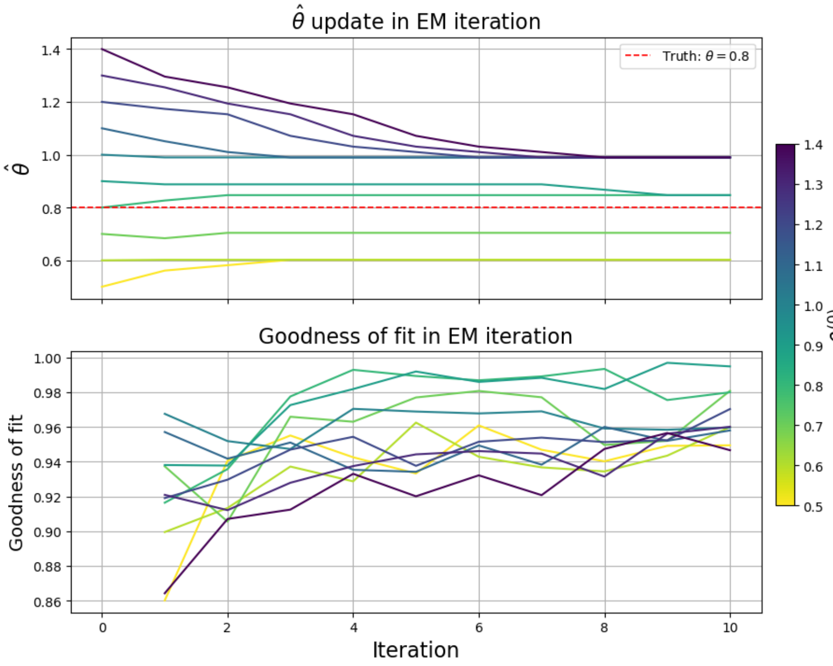

FIG 28. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 26.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

3. $\theta = 1.2$

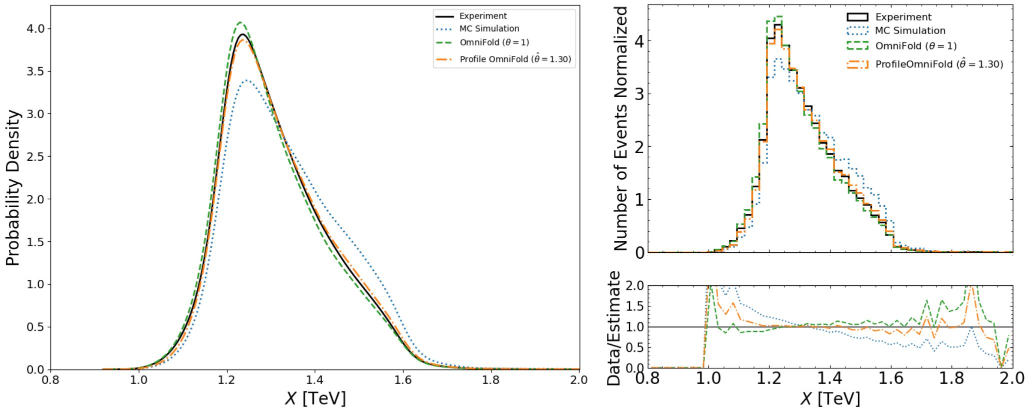

FIG 29. *Results of unfolding the CMS open data.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

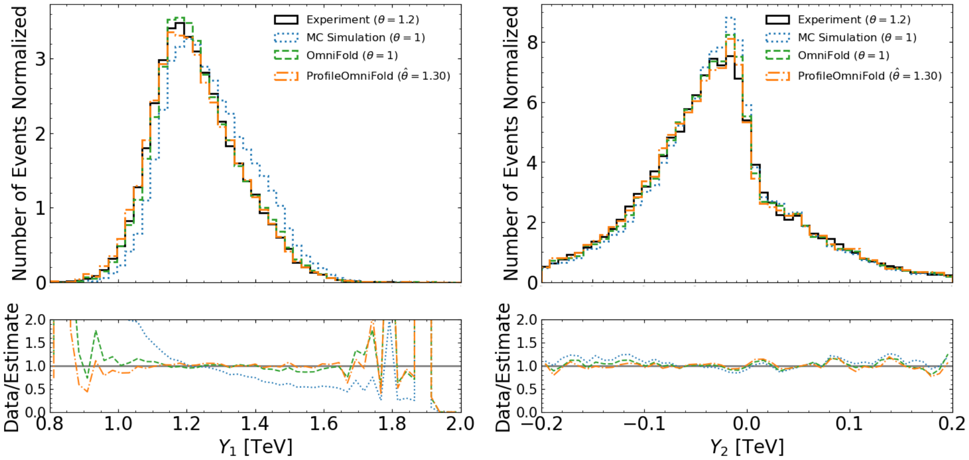

FIG 30. *Results corresponding to Figure 29 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

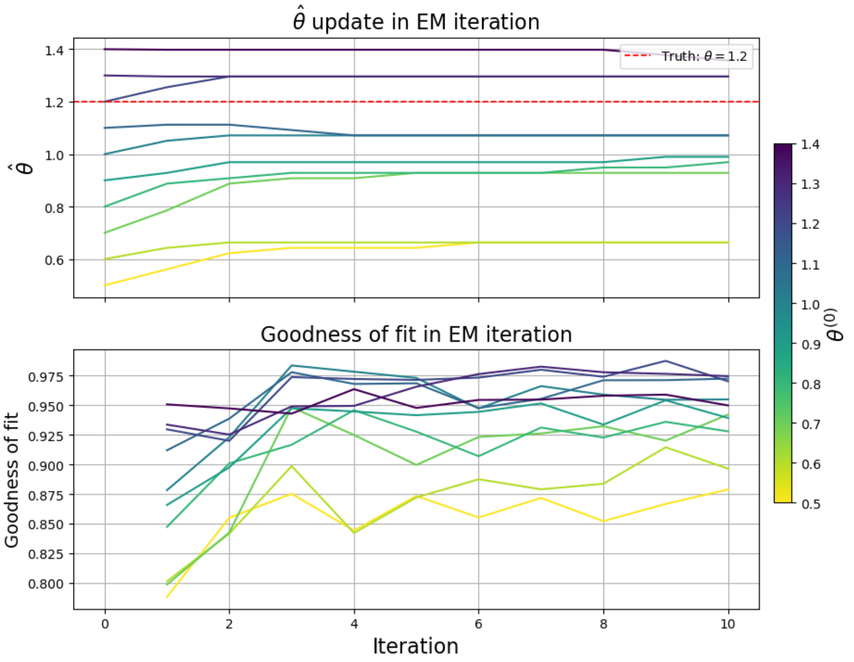

FIG 31. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 29.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

4. $\theta = 1.4$

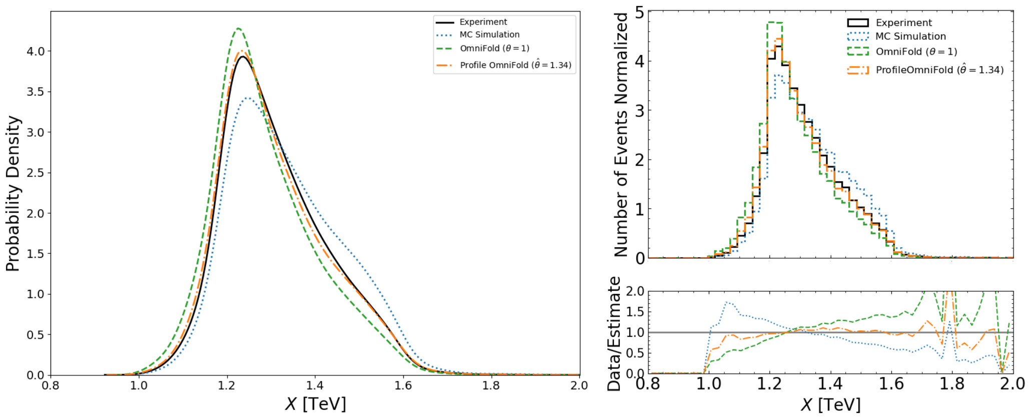

FIG 32. *Results of unfolding the CMS open data.* ***Left****: Particle-level kernel density estimates of the truth distribution (black), the MC distribution (blue), and the reweighted MC distributions obtained using the POF (orange) and OF (green) algorithms, each run for 10 iterations.* ***Top-right****: Histograms of the four corresponding spectra, aggregated into 50 bins.* ***Bottom-right****: The ratio of the truth spectrum to the unfolded spectra.*

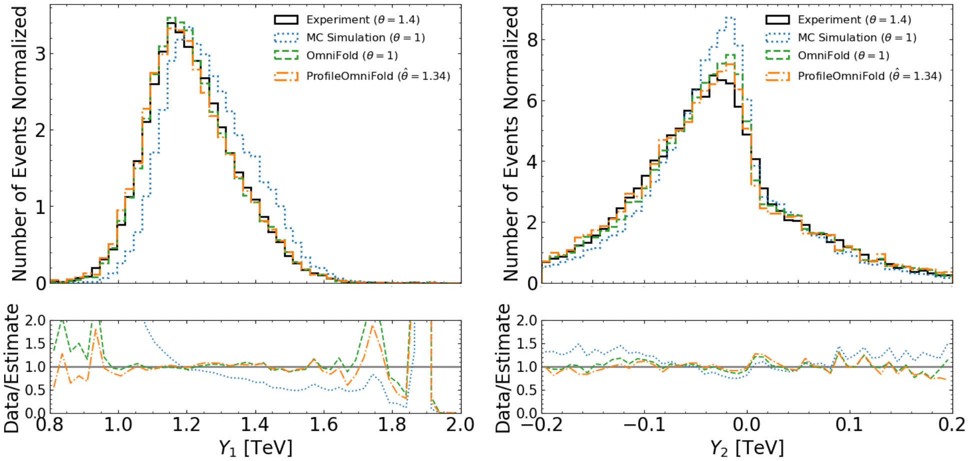

FIG 33. *Results corresponding to Figure 32 in detector-level space.* ***Left****: Histograms of the corresponding spectra of Y_1.* ***Right****: Histograms of the corresponding spectra of Y_2.*

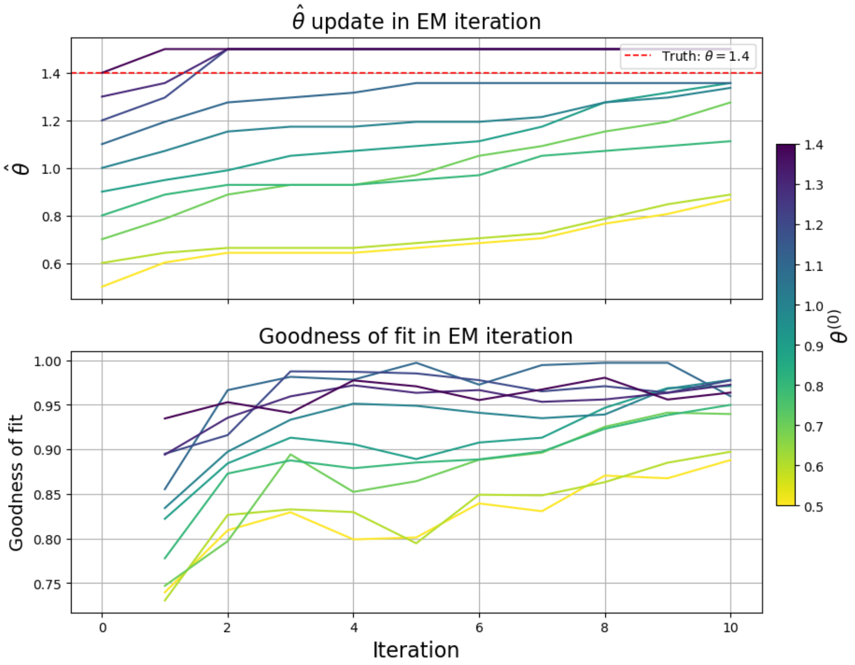

FIG 34. *Evolution of the nuisance parameter and the step-1 classifier's goodness-of-fit statistic for the results shown in Figure 32.* ***Top****: Updated estimates $\hat{\theta}$ across iterations for different initializations $\theta^{(0)}$.* ***Bottom****: Goodness-of-fit statistic of the step-1 classifier at each iteration.*

SUPPLEMENTARY MATERIAL B: TECHNICAL DERIVATIONS

In this section, we provide detailed proofs of the propositions that were omitted in the main text.

B.1. EM Derivation. In this section, we provide a detailed derivation of the EM algorithm presented in Section 2.2.

PROOF OF PROPOSITION 1. First, recall the population-level log-likelihood of the density function is

$$l(f) = \int p(y) \log \left(\int p(y|x) f(x) dx \right) dy,$$

$$\text{subject to} \int f(x) dx = 1.$$

The corresponding Q-function is

$$Q(f, f^{(k)}) = \int p(y) \int p(x|y, f^{(k)}) \log p(x, y|f) dx dy,$$

$$\text{subject to} \int f(x) dx = 1.$$

To solve for $f^{(k+1)} = \arg\max_f Q(f, f^{(k)})$, we solve the problem in the Lagrangian form

$$\tilde{Q}(f, f^{(k)}) = \int p(y) \int p(x|y, f^{(k)}) \log p(x, y|f) dx dy - \lambda \left(\int f(x) dx - 1 \right).$$

The Gâteaux derivative $\frac{\delta \tilde{Q}}{\delta f}$ satisfies

$$\begin{aligned}
\int \frac{\delta \tilde{Q}}{\delta f(x)}\phi(x)dx &= \left[\frac{d}{d\epsilon}\tilde{Q}(f+\epsilon\phi)\right]_{\epsilon=0} \\
&= \left[\frac{d}{d\epsilon}\int p(y)\int p(x|y,f^{(k)})\log[p(y|x)(f(x)+\epsilon\phi(x))]dxdy - \lambda\int(f(x)+\epsilon\phi(x))dx - \lambda\right]_{\epsilon=0} \\
&= \left[\int p(y)\int p(x|y,f^{(k)})\frac{p(y|x)\phi(x)}{p(y|x)(f(x)+\epsilon\phi(x))}dxdy - \lambda\int\phi(x)dx\right]_{\epsilon=0} \\
&= \int p(y)\int p(x|y,f^{(k)})\frac{p(y|x)\phi(x)}{p(y|x)f(x)}dxdy - \lambda\int\phi(x)dx \\
&= \int \phi(x)\left[\int p(y)p(x|y,f^{(k)})\frac{1}{f(x)}dy - \lambda\right]dx.
\end{aligned}$$

Therefore, this shows that

$$\frac{\delta \tilde{Q}}{\delta f(x)} = \int p(y)p(x|y,f^{(k)})\frac{1}{f(x)}dy - \lambda.$$

Setting the derivative to be 0,

$$\begin{aligned}
\frac{\delta \tilde{Q}}{\delta f(x)} &= \int \frac{p(x|y,f^{(k)})}{f(x)}p(y)dy - \lambda = 0 \\
\lambda &= \int \frac{1}{f(x)}p(x|y,f^{(k)})p(y)dy \\
&= \int \frac{1}{f(x)}\frac{p(y|x)f^{(k)}(x)}{\int p(y|x')f^{(k)}(x')dx'}p(y)dy.
\end{aligned}$$

Integrating both sides by $\int f(x)dx$ yields that $\lambda = 1$. Therefore, the stationary point satisfies

$$f(x) = \int \frac{p(y|x)f^{(k)}(x)}{\int p(y|x')f^{(k)}(x')dx'}p(y)dy.$$

Moreover, the second order derivative $\frac{\delta \tilde{Q}}{\delta f(x)\delta f(x')}$ satisfies

$$\begin{aligned}
\int\int \frac{\delta \tilde{Q}}{\delta f(x)\delta f(x')}\phi(x)\phi(x')dxdx' &= \left[\frac{d^2}{d\epsilon^2}\tilde{Q}(f+\epsilon\phi)\right]_{\epsilon=0} \\
&= \left[\frac{d}{d\epsilon}\int p(y)\int p(x|y,f^{(k)})\frac{\phi(x)}{(f(x)+\epsilon\phi(x))}dxdy - \lambda\int\phi(x)dx\right]_{\epsilon=0} \\
&= \left[-\int p(y)\int p(x|y,f^{(k)})\frac{\phi^2(x)}{[(f(x)+\epsilon\phi(x))]^2}dxdy\right]_{\epsilon=0} \\
&= \int \frac{\phi^2(x)}{f^2(x)}\left(-\int p(y)p(x|y,f^{(k)})dy\right)dx.
\end{aligned}$$

Therefore, this implies that

$$\frac{\delta \tilde{Q}}{\delta f(x)\delta f(x')} = -\frac{\delta(x-x')}{f^2(x)}\int p(y)p(x|y,f^{(k)})dy \leq 0$$

which shows that $\tilde{Q}$ is concave in f.

□

The proof works similarly for OmniFold update by reparameterizing $f(x) = \nu(x)q(x)$. See details in Andreassen et al. (2020).

B.2. Profile OMNIFOLD Derivation.

PROOF OF PROPOSITION 2. The Q-function in the POF algorithm is

$$\begin{aligned} Q(\nu,\theta|\nu^{(k)},\theta^{(k)}) &= \int p(y) \int p(x|y,\nu^{(k)},\theta^{(k)}) \log p(x,y|\nu,\theta) dxdy + \log p_0(\theta) \\ &= \int p(y) \int p(x|y,\nu^{(k)},\theta^{(k)}) \log[w(y,x,\theta)q(y|x)\nu(x)q(x)] dxdy + \log p_0(\theta) \\ &\text{subject to } \int \nu(x)q(x)dx = 1. \end{aligned}$$

The Q-function can be decomposed into two parts that depend on ν and θ separately, i.e.

$$\begin{aligned} Q(\nu,\theta|\nu^{(k)},\theta^{(k)}) = &\int p(y) \int p(x|y,\nu^{(k)},\theta^{(k)}) \log[\nu(x)q(x)q(y|x)] dxdy \\ &+ \int p(y) \int p(x|y,\nu^{(k)},\theta^{(k)}) \log[w(y,x,\theta)] dxdy + \log p_0(\theta) \\ &= Q_1(\nu|\nu^{(k)},\theta^{(k)}) + Q_2(\theta|\nu^{(k)},\theta^{(k)}). \end{aligned}$$

Therefore, we can maximize Q by maximizing Q_1 and Q_2 separately. Write Q_1 in its Lagragian form

$$\tilde{Q}_1(\nu,|\nu^{(k)},\theta^{(k)}) = Q_1(\nu|\nu^{(k)},\theta^{(k)}) - \lambda\left(\int \nu(x)q(x)dx - 1\right).$$

Take derivative of $\tilde{Q}_1$ with respect to $\nu(x)$ and set it to be 0,

$$\frac{\delta}{\delta\nu(x)}\tilde{Q}_1(\nu,|\nu^{(k)},\theta^{(k)}) = \frac{\int p(y)p(x|y,\nu^{(k)},\theta^{(k)})dy}{\nu(x)} - \lambda q(x) = 0.$$

Integrating both sides over $\int \nu(x)dx$ yields that $\lambda = 1$. Therefore, the stationary condition for $\nu(x)$ satisfies

$$\begin{aligned} \nu(x) &= \frac{\int p(y)p(x|y,\nu^{(k)},\theta^{(k)})dy}{q(x)} \\ &= \int \frac{p(y)w(y,x,\theta^{(k)})q(y|x)\nu^{(k)}(x)dy}{\int w(y,x',\theta^{(k)})q(y|x')\nu^{(k)}(x')q(x')dx'} \\ &= \nu^{(k)}(x) \int \frac{p(y)}{\tilde{q}^{(k)}(y)} w(y,x,\theta^{(k)})q(y|x)dy, \end{aligned}$$

where $\tilde{q}^{(k)}(y) = \int w(y,x',\theta^{(k)})q(y|x')\nu^{(k)}(x')q(x')dx'$. Moreover, since

$$\frac{\delta}{\nu(x)\nu(x')}\tilde{Q}(\nu,\theta^{(k)}|\nu^{(k)},\theta^{(k)}) = -\frac{\delta(x-x')}{\nu^2(x)} \int p(y)p(x|y,\nu^{(k)},\theta^{(k)})dy \leq 0,$$

the stationary point is a global maximum. Now for Q_2, we have

$$Q_2(\theta|\nu^{(k)},\theta^{(k)}) = \int p(y) \int p(x|y,\nu^{(k)},\theta^{(k)}) \log[w(y,x,\theta)] dxdy + \log p_0(\theta).$$

Hence,

$$
\begin{aligned}
\text{argmax}_\theta Q_2(\theta|\nu^{(k)}, \theta^{(k)}) &= \text{argmax}_\theta \int p(y) \int p(x|y, \nu^{(k)}, \theta^{(k)}) \log[w(y, x, \theta)] dxdy + \log p_0(\theta) \\
&= \text{argmax}_\theta \int \int p(y) \frac{w(y, x, \theta^{(k)}) q(y|x) \nu^{(k)}(x) q(x)}{\tilde{q}^{(k)}(y)} \log[w(y, x, \theta)] dxdy + \log p_0(\theta) \\
&= \text{argmax}_\theta \int \int q(x, y) \nu^{(k)} w(y, x, \theta^{(k)}) \frac{p(y)}{\tilde{q}^{(k)}(y)} \log[w(y, x, \theta)] dxdy + \log p_0(\theta).
\end{aligned}
$$

□

B.3. w Function Training Through Classifiers.

PROOF OF PROPOSITION 3. First, rewrite

$$
\begin{aligned}
w(y, x, \theta) &= \frac{p(y|x, \theta)}{q(y|x)} \\
&= \frac{p(x, y|\theta)}{p(x|\theta)} \cdot \frac{q(x, y)}{q(x)} \\
&= \frac{p(x, y|\theta)}{q(x, y)} \cdot \frac{q(x)}{p(x|\theta)}.
\end{aligned}
$$

For the first ratio, let $f_1 : \mathcal{X} \times \mathcal{Y} \times \Theta \to [0, 1]$ be the Bayes optimal classifier to distinguish dataset $\mathcal{D}_1 = \{X_i, Y_i, \theta_i\}$ from $\mathcal{D}_2 = \{X_i', Y_i', \theta_i'\}$. Then the learned ratio from the classifier is

$$
\begin{aligned}
\frac{f_1(x, y, \theta)}{1 - f_1(x, y, \theta)} = \frac{p(x, y, \theta)}{q(x, y, \theta)} &= \frac{p(x, y|\theta) p(\theta)}{q(x, y|\theta) q(\theta)} \\
&= \frac{p(x, y|\theta) p(\theta)}{q(x, y) q(\theta)}
\end{aligned}
$$

where the last line follows since $q(x, y|\theta) = q(x, y)$, i.e. the joint distribution of X_i', Y_i' does not depend on θ_i'. Similarly, for the second ratio, let $f_2 : \mathcal{X} \times \Theta \to [0, 1]$ be the Bayes optimal classifier to distinguish dataset $\tilde{\mathcal{D}}_2 = \{X_i', \theta_i'\}$ from $\tilde{\mathcal{D}}_1 = \{X_i, \theta_i\}$. Then the learned ratio from the classifier is

$$
\begin{aligned}
\frac{f_2(x, \theta)}{1 - f_2(x, \theta)} = \frac{q(x, \theta)}{p(x, \theta)} &= \frac{q(x|\theta) q(\theta)}{p(x|\theta) p(\theta)} \\
&= \frac{q(x) q(\theta)}{p(x|\theta) p(\theta)}
\end{aligned}
$$

where again the last line follows since the distribution of X_i' does not depend on θ_i'. Therefore, combining these, we have

$$
\begin{aligned}
\frac{f_1(x, y, \theta) f_2(x, y, \theta)}{(1 - f_1(x, y, \theta))(1 - f_2(x, y, \theta))} &= \frac{p(x, y|\theta) p(\theta)}{q(x, y) q(\theta)} \cdot \frac{q(x) q(\theta)}{p(x|\theta) p(\theta)} \\
&= \frac{p(x, y|\theta)}{q(x, y)} \cdot \frac{q(x)}{p(x|\theta)}.
\end{aligned}
$$

□

SUPPLEMENTARY MATERIAL C: DATA AND CODE ACCESS

The code and datasets used in this study are available at:

- GitHub repository: https://github.com/richardzhs/ProfileOmnifold

REFERENCES

AAD, G. et al. (2024). A simultaneous unbinned differential cross section measurement of twenty-four Z+jets kinematic observables with the ATLAS detector.

ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M. et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *OSDI* **16** 265–283.

AGOSTINELLI, S. et al. (2003). GEANT4–a simulation toolkit. *Nucl. Instrum. Meth. A* **506** 250–303. https://doi.org/10.1016/S0168-9002(03)01368-8

ALLISON, J. et al. (2006). Geant4 developments and applications. *IEEE Transactions on Nuclear Science* **53** 270-278. https://doi.org/10.1109/TNS.2006.869826

ALLISON, J. et al. (2016). Recent developments in Geant4. *Nucl. Instrum. Meth. A* **835** 186–225. https://doi.org/10.1016/j.nima.2016.06.125

ANDREASSEN, A. and NACHMAN, B. (2019). Neural Networks for Full Phase-space Reweighting and Parameter Tuning.

ANDREASSEN, A., KOMISKE, P. T., METODIEV, E. M., NACHMAN, B. and THALER, J. (2020). OmniFold: A Method to Simultaneously Unfold All Observables. *Physics Reivew Letter* **124**.

ANDREASSEN, A., KOMISKE, P. T., METODIEV, E. M., NACHMAN, B., SURESH, A. and THALER, J. (2021). Scaffolding Simulations with Deep Learning for High-dimensional Deconvolution. In *9th International Conference on Learning Representations*.

ANDREEV, V. et al. (2021). Measurement of lepton-jet correlation in deep-inelastic scattering with the H1 detector using machine learning for unfolding.

ANDREEV, V. et al. (2023). Unbinned Deep Learning Jet Substructure Measurement in High Q^2 ep collisions at HERA.

ARRATIA, M. et al. (2022). Publishing unbinned differential cross section results. *JINST* **17** P01024. https://doi.org/10.1088/1748-0221/17/01/P01024

BACKES, M., BUTTER, A., DUNFORD, M. and MALAESCU, B. (2024). An unfolding method based on conditional invertible neural networks (cINN) using iterative training. *SciPost Phys. Core* **7** 007. https://doi.org/10.21468/SciPostPhysCore.7.1.007

BADEA, T. E.-P. A. A., BATY, A., BOSSI, H., CHEN, Y.-C., CHEN, Y., ZHANG, J., INNOCENTI, G. M., MAGGI, M., MCGINN, C., PETERS, M., SHENG, T.-A., MIKUNI, V., AVAYLON, M., KOMISKE, P., METODIEV, E., THALER, J., NACHMAN, B. and LEE, Y.-J. (2025). Unbinned measurement of thrust in e^+e^- collisions at $\sqrt{s} = 91.2\,\text{GeV}$ with ALEPH archived data.

BARMAN, R. K., CHOUDHURY, A. and SARKAR, S. (2025). Reconstructing Sparticle masses at the LHC using Generative Machine Learning. *arXiv preprint*.

BELLAGENTE, M., BUTTER, A., KASIECZKA, G., PLEHN, T. and WINTERHALDER, R. (2020). How to GAN away detector effects. *SciPost Physics* **8** 070. https://doi.org/10.21468/SciPostPhys.8.4.070

BLOBEL, V. (2011). Unfolding Methods in Particle Physics. *PHYSTAT2011 Proceedings* 240. https://doi.org/10.5170/CERN-2011-006

BUTTER, A., DIEFENBACHER, S., HUETSCH, N., MIKUNI, V., NACHMAN, B., PALACIOS SCHWEITZER, S. and PLEHN, T. (2025a). Generative Unfolding with Distribution Mapping. *SciPost Physics* **18** 200. https://doi.org/10.21468/SciPostPhys.18.6.200

BUTTER, A., HEIMEL, T., HUETSCH, N., KAGAN, M. and PLEHN, T. (2025b). Simulation-Prior Independent Neural Unfolding Procedure. *arXiv preprint*.

BUTTER, A., HUETSCH, N., MIKUNI, V., NACHMAN, B. and PALACIOS SCHWEITZER, S. (2025c). Analysis-ready Generative Unfolding.

CACCIARI, M. and SALAM, G. P. (2006). Dispelling the N^3 myth for the k_t jet-finder. *Phys. Lett.* **B641** 57-61. https://doi.org/10.1016/j.physletb.2006.08.037

CACCIARI, M., SALAM, G. P. and SOYEZ, G. (2008). The Anti-k(t) jet clustering algorithm. *JHEP* **04** 063. https://doi.org/10.1088/1126-6708/2008/04/063

CACCIARI, M., SALAM, G. P. and SOYEZ, G. (2012). FastJet User Manual. *Eur. Phys. J.* **C72** 1896. https://doi.org/10.1140/epjc/s10052-012-1896-2

CANELLI, F., CORMIER, K., CUDD, A., GILLBERG, D., HUANG, R. G., JIN, W., LEE, S., MIKUNI, V., MILLER, L., NACHMAN, B., PAN, J., PANI, T., PETTEE, M., SONG, Y. and TORALES, F. (2025). A Practical Guide to Unbinned Unfolding.

Chae, M., Martin, R. and Walker, S. G. (2019). On an algorithm for solving Fredholm integrals of the first kind. *Statistics and Computing* **29** 645–654. https://doi.org/10.1007/s11222-018-9829-z

Chan, J. and Nachman, B. (2023). Unbinned Profiled Unfolding. *Physical Review D*. https://doi.org/10.1103/PhysRevD.108.016002

Chatrchyan, S. et al. (2008). The CMS Experiment at the CERN LHC. *JINST* **3** S08004. https://doi.org/10.1088/1748-0221/3/08/S08004

Chatrchyan, S. et al. (2011). 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. https://doi.org/10.1007/JHEP09(2011)109

Chollet, F. (2017). Keras. https://github.com/fchollet/keras.

CMS collaboration (2016a). Simulated dataset QCD_Pt-1000to1400_TuneZ2_7TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive). *CERN Open Data Portal*. https://doi.org/10.7483/OPENDATA.CMS.96U2.3YAH

CMS Collaboration (2016b). Simulated dataset QCD_Pt-1400to1800_TuneZ2_7TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive). *CERN Open Data Portal*. https://doi.org/10.7483/OPENDATA.CMS.RC9V.B5KX

CMS Collaboration (2016c). Simulated dataset QCD_Pt-1800_TuneZ2_7TeV_pythia6 in AODSIM format for 2011 collision data (SM Exclusive). *CERN Open Data Portal*. https://doi.org/10.7483/OPENDATA.CMS.CX2X.J3KW

H1 Collaboration (2022a). Machine learning-assisted measurement of multi-differential lepton-jet correlations in deep-inelastic scattering with the H1 detector. *H1prelim-22-031*.

Collaboration, L. (2022b). Multidifferential study of identified charged hadron distributions in Z-tagged jets in proton-proton collisions at $\sqrt{s}$ =13 TeV.

H1 Collaboration (2023). Machine learning-assisted measurement of azimuthal angular asymmetries in deep-inelastic scattering with the H1 detector. *H1prelim-23-031*.

CMS Collaboration (2024a). Measurement of event shapes in minimum bias events from pp collisions at 13 TeV Technical Report, CERN, Geneva.

ATLAS Collaboration (2024b). Measurement of Track Functions in ATLAS Run 2 Data.

Cranmer, K. (2015). Practical Statistics for the LHC. *arXiv:1503.07622*.

Cranmer, K., Pavez, J. and Louppe, G. (2015). Approximating Likelihood Ratios with Calibrated Discriminative Classifiers.

Crucinio, F. R., Doucet, A. and Johansen, A. M. (2023). A Particle Method for Solving Fredholm Equations of the First Kind. *Journal of the American Statistical Association* **118** 937–947. https://doi.org/10.1080/01621459.2021.1962328

D'Agostini, G. (1995). A Multidimensional unfolding method based on Bayes' theorem. *Nucl. Instrum. Meth.* **A362** 487-498. https://doi.org/10.1016/0168-9002(95)00274-X

Datta, K., Kar, D. and Roy, D. (2019). Unfolding with Generative Adversarial Networks. *arXiv preprint arXiv:1903.11485*.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39** 1–38.

Diefenbacher, S., Liu, G.-H., Mikuni, V., Nachman, B. and Nie, W. (2024). Improving generative model-based unfolding with Schrödinger bridges. *Physical Review D* **109**. https://doi.org/10.1103/PhysRevD.109.076011

Eggermont, P. P. B. (1999). Nonlinear Smoothing and the EM Algorithm for Positive Integral Equations of the First Kind. *Applied Mathematics and Optimization* **39** 75–91.

Eggermont, P. P. B. and Lariccia, V. N. (1995). Maximum Smoothed Likelihood Density Estimation for Inverse Problems. *The Annals of Statistics* **23** 199-220.

Eggermont, P. P. B. and Lariccia, V. N. (1997). Nonlinearly Smoothed EM Density Estimation with Automated Smoothing Parameter Selection for Nonparametric Deconvolution Problems. *Journal of the American Statistical Association* **92** 1451–1458.

Falcão, A. and Takacs, A. (2025). High-Dimensional Unfolding in Large Backgrounds.

Huetsch, N. et al. (2024). The Landscape of Unfolding with Machine Learning.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization.

Komiske, P. T., Kryhin, S. and Thaler, J. (2022). Disentangling Quarks and Gluons in CMS Open Data. *Phys. Rev. D* **106** 094021. https://doi.org/10.1103/PhysRevD.106.094021

Komiske, P., Mastandrea, R., Metodiev, E., Naik, P. and Thaler, J. (2019a). CMS 2011A Simulation | Pythia 6 QCD 1000-1400 | pT > 375 GeV | MOD HDF5 Format. https://doi.org/10.5281/zenodo.3341502

Komiske, P., Mastandrea, R., Metodiev, E., Naik, P. and Thaler, J. (2019b). CMS 2011A Simulation | Pythia 6 QCD 1400-1800 | pT > 375 GeV | MOD HDF5 Format. https://doi.org/10.5281/zenodo.3341770

Komiske, P., Mastandrea, R., Metodiev, E., Naik, P. and Thaler, J. (2019c). CMS 2011A Simulation | Pythia 6 QCD1800-inf | pT > 375 GeV | MOD HDF5 Format. https://doi.org/10.5281/zenodo.3341772

Komiske, P. T., Mastandrea, R., Metodiev, E. M., Naik, P. and Thaler, J. (2020). Exploring the Space of Jets with CMS Open Data. *Phys. Rev. D* **101** 034009. https://doi.org/10.1103/PhysRevD.101.034009

Kondor, A. (1983). Method of convergent weights: An iterative procedure for solving Fredholm's integral equations of the first kind. *Nuclear Instruments and Methods* **216** 177–181.

Kuusela, M. (2012). Statistical Issues in Unfolding Methods for High Energy Physics. *Aalto University Master's Thesis*.

Lucy, L. B. (1974). An iterative technique for the rectification of observed distributions. *Astronomical Journal* **79** 745. https://doi.org/10.1086/111605

Mülthei, H. N. (1992). Iterative continuous maximum-likelihood reconstruction method. *Mathematical Methods in the Applied Sciences* **15** 275–286.

Mülthei, H. N. and Schorr, B. (1987). On an iterative method for a class of integral equations of the first kind. *Mathematical Methods in the Applied Sciences* **9** 137–168.

Mülthei, H. N. and Schorr, B. (1989). On properties of the iterative maximum likelihood reconstruction method. *Mathematical Methods in the Applied Sciences* **11** 331–342.

Pani, T. (2024). Generalized angularities measurements from STAR at $\sqrt{s_{NN}}$ = 200 GeV. *EPJ Web Conf.* **296** 11003. https://doi.org/10.1051/epjconf/202429611003

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*.

Petitjean, A., Butter, A., Greif, K., Palacios Schweitzer, S., Plehn, T., Spinner, J. and Whiteson, D. (2025). Generative Unfolding of Jets and Their Substructure. *arXiv preprint*.

Richardson, W. H. (1972). Bayesian-Based Iterative Method of Image Restoration. *J. Opt. Soc. Am.* **62** 55–59. https://doi.org/10.1364/JOSA.62.000055

Shepp, L. A. and Vardi, Y. (1982). Maximum likelihood reconstruction for emission tomography. *IEEE transactions on medical imaging* **1** 113–122. https://doi.org/10.1109/TMI.1982.4307558

Shmakov, A., Greif, K., Fenton, M., Ghosh, A., Baldi, P. and Whiteson, D. (2023). End-To-End Latent Variational Diffusion Models for Inverse Problems in High Energy Physics. In *Advances in Neural Information Processing Systems*.

Sjöstrand, T., Mrenna, S. and Skands, P. Z. (2006). PYTHIA 6.4 Physics and Manual. *JHEP* **05** 026. https://doi.org/10.1088/1126-6708/2006/05/026

Song, Y. (2023). Measurement of CollinearDrop jet mass and its correlation with SoftDrop groomed jet substructure observables in $\sqrt{s} = 200$ GeV *pp* collisions by STAR.

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

Vardi, Y., Shepp, L. A. and Kaufman, L. (1985). A Statistical Model for Positron Emission Tomography. *Journal of the American Statistical Association* **80** 8-20. https://doi.org/10.1080/01621459.1985.10477119

Zhu, H., Desai, K., Kuusela, M., Mikuni, V., Nachman, B. and Wasserman, L. (2024). Multidimensional Deconvolution with Profiling. https://doi.org/10.48550/arXiv.2409.10421