Tools for Unbinned Unfolding

Ryan Milton
University of California, Riverside
Riverside, CA 92521, USA
rmilt003@ucr.edu

Vinicius Mikuni
Nagoya University
Kobayashi-Maskawa Institute
Aichi 464-8602, Japan
vinicius.mikuni@gmail.com

Trevin Lee
University of California, San Diego
La Jolla, CA 92093, USA
trevin@ucsd.edu

Miguel Arratia
University of California, Riverside
Riverside, CA 92521, USA
miguel.arratia@ucr.edu

Tanvi Wamorkar
Stanford University
Stanford, CA 94305, USA
twamorka@stanford.edu

Benjamin Nachman
Stanford University
Stanford, CA 94305, USA
nachman@stanford.edu

Abstract

Machine learning enables measurements without discretization, moving beyond histogram-based approaches. To support adoption and standardization of unbinned unfolding, we develop new methods and software. We show that boosted decision trees provide a competitive and computationally efficient alternative to deep neural networks for problems with a limited number of high-level features, complementing models that scale to high-dimensional phase space. We extend existing datasets to include acceptance effects and perform controlled comparisons between binned and unbinned methods. Finally, we present two implementations of OmniFold: one integrated with RooUnfold and a standalone Python package.

1 Introduction

In many scientific applications, measurements are distorted by detectors, making it difficult to recover the underlying data distribution. In particle physics, this problem is known as unfolding: correcting observed data to estimate the true distribution before detector effects. Traditionally, unfolding is performed using histograms, where data are discretized into bins. In this approach, one inverts the equation $m = Rt$ to estimate t, where m represents the measured data, t the underlying true distribution (the data without detector effects), and R the response matrix encoding the probability that an event originating in truth bin j is measured in bin i:

$$R_{ij} = \Pr(\text{measure in bin } i \mid \text{truth in bin } j). \tag{1}$$

This matrix is estimated using realistic detector simulations. However, the naive solution $\hat{t} = R^{-1}m$ is unstable: it amplifies statistical fluctuations and may not exist when R is non-square. To address these issues, a variety of regularized inversion techniques have been developed (see Refs. [1–4] for reviews and Refs. [5–7] for commonly used algorithms).

Histogram-based unfolding imposes fixed binning choices and reduces high-dimensional data to low-dimensional representations, limiting the ability to preserve correlations and reinterpret results for new observables.

Conference on Physics and AI at Stanford University (PAI 2026).

Conversely, ML-based unfolding methods avoid binning and enable high-dimensional measurements, incorporating many features of the data and improving precision by accounting for correlations with detector effects. These approaches have been successfully applied in several experiments [8–17].

Despite progress in applying ML-based methods, standardized tools for performing such measurements remain lacking. In contrast, classical unfolding is supported by well-established software, most notably RooUnfold [18, 19], which is widely used across experiments. In this work, we present user-friendly software for unbinned unfolding. We introduce two complementary toolsets: a RooUnfold-inspired implementation of the OmniFold [20, 21] algorithm, and a standalone Python package available via the Python Package Index (pip). The RooUnfold-based implementation is designed to integrate seamlessly with existing workflows, allowing users to adopt ML-based methods without modifying their analysis inputs or outputs. The pip-based package provides a lightweight alternative that is entirely Python based.

2 Methodology

We first compare unbinned unfolding with a binned unfolding method — Lucy–Richardson deconvolution [22, 23], also known as Iterative Bayesian Unfolding (IBU) [5]. We first review IBU and then extend it to unbinned, high-dimensional data using machine learning (OmniFold).

2.1 Iterative Unfolding

We consider the simplified case of no background and identical particle-level and detector-level selections. Here, "detector-level" refers to observed data after measurement effects, while "particle-level" refers to the underlying true distribution. Starting from an initial particle-level estimate $t_j^{(0)}$, IBU updates the distribution as

$$t_j^{(n+1)} = \sum_i \frac{R_{ij} t_j^{(n)}}{\sum_k R_{ik} t_k^{(n)}} m_i, \tag{2}$$

where n is the iteration number. The final result $\hat{t}_{\text{IBU}} = t^{(N)}$ approaches the maximum likelihood solution as $N \to \infty$, though in practice N is limited for regularization.

This update can be broken down into two steps. First,

$$\omega_i^{(n)} = \frac{m_i}{\tilde{m}_i^{(n)}} = \frac{m_i}{\sum_k R_{ik} t_k^{(n)}}, \tag{3}$$

where $\tilde{m}_i^{(n)}$ is the detector-level spectrum after the nth iteration. The initial $\tilde{m}_i^{(0)}$ is the starting detector-level simulation. Applying the weights $\omega_i^{(n)}$ to $\tilde{m}_i^{(n)}$ reweights detector-level simulation to match the data m_i. Second,

$$\nu_j^{(n+1)} = \frac{\tilde{t}_j^{(n)}}{t_j^{(0)}} = \frac{\sum_i R_{ij} t_j^{(n)} \omega_i^{(n)}}{t_j^{(0)}}, \tag{4}$$

where $\tilde{t}_j^{(n)}$ is the particle-level spectrum after the nth iteration. After N iterations, the unfolded result is $t^{(0)}$ reweighted by $\nu^{(N)}$. This formulation makes explicit that IBU estimates two density ratios: detector-level weights ω (matching simulation to observed data) and particle-level weights ν (mapping back to the true distribution).

Backgrounds are typically subtracted bin-by-bin corrections before unfolding. $t^{(N)}$ is modified bin-by-bin after unfolding to handle events that do not pass detector-level selection.

2.2 Unbinned Unfolding

OmniFold extends IBU to continuous observables by replacing histograms with probability densities: $m \mapsto p_{\text{data}}(x)$, $\tilde{m}^{(0)} \mapsto p_{\text{det.}}(x)$, and $t^{(0)} \mapsto p_{\text{part.}}(z)$, where x and z are detector- and particle-level observables. The corresponding updates are

$$\omega^{(n)}(x) = \frac{p_{\text{data}}(x)}{\tilde{p}_{\text{det.}}^{(n)}(x)} = \frac{p_{\text{data}}(x)}{\int \mathrm{d}z\, p_{\text{det.}}(x) \nu^{(n)}(z)}, \tag{5}$$

and

$$\nu^{(n+1)}(z) = \frac{\tilde{p}^{(n)}_{\text{part.}}(z)}{p_{\text{part.}}(z)} = \frac{\int \mathrm{d}x\, p_{\text{part.}}(z)\omega^{(n)}(x)}{p_{\text{part.}}(z)}. \tag{6}$$

These density ratios are estimated using ML classifiers trained to distinguish data from simulation at detector and particle levels. The ratio ω is obtained by distinguishing data from detector-level simulation weighted by ν from the previous iteration, while ν is obtained by distinguishing particle-level simulation from the same events reweighted by ω. After N iterations, the final result is the initial set of particle-level events weighted by $\nu^{(N)}(z)$.

Depending on dimensionality, the method is referred to as UNIFOLD (one observable), MULTIFOLD (multiple observables), or OMNIFOLD (full phase space). We use the term OMNIFOLD to refer to the general algorithm unless stated explicitly otherwise.

We provide two implementations. The first is a RooUnfold-inspired version based on Boosted Decision Trees (BDTs), which integrates with existing workflows. BDTs are fast, stable, and require minimal tuning. BDTs also perform well on tabular data [24–26], which physics measurements often use. The second is a standalone Python package based on neural networks, which scales well to high-dimensional and variable-length inputs.

2.2.1 RooUnfold-inspired BDTs

We use GradientBoostingClassifiers from SCIKIT-LEARN [27] with the default parameters (depth 3, 100 estimators, learning rate 0.1). Corrections for events passing particle-level but not detector-level selections are included following Ref. [21], assigning weights via the conditional expectation $\omega^{(n)} \equiv \langle \omega^{(n)}(x)|z\rangle$, estimated using a GradientBoostingRegressor. In other words, we predict the weights for events that fail detector-level selections with a regression model trained to predict $\omega^{(n)}$ for events that pass the selections. Background subtraction is not implemented in this approach.

2.2.2 Deep Neural Networks

The neural network implementation uses KERAS [28] and TENSORFLOW [29], with built-in data handling and training. Two default models are provided: a fully connected network (DNN) for fixed-length inputs and a Point-Edge Transformer (PET) [30] for variable-length data. We use these default models with their set parameters in this work.

3 Application to jets at particle colliders

We study simulated jets produced in high-energy proton–proton collisions in the Z+jets dataset [31]. Jets are collimated sprays of particles and are a central object of study in collider physics. Simulated events are generated using PYTHIA [32], while pseudodata are generated using HERWIG [33]. Since we are working with simulated data, we know the particle-level distributions and can test our unfolding procedures. Detector effects are modeled using the DELPHES [34] fast simulation of the CMS detector. Particles are clustered into jets using the anti-k_T algorithm with radius parameter $R = 0.4$ [35, 36].

We consider six commonly used jet observables: jet mass m, constituent multiplicity M, jet width w, groomed jet mass $\ln \rho$, the substructure variable τ_{21}, and the momentum fraction z_g. These observables probe different aspects of the internal structure of jets.

We apply several unfolding strategies: OMNIFOLD using a transformer-based neural network (PET) with all of the particles in the events as input, MULTIFOLD using DNNs and BDTs to unfold all observables simultaneously, and UNIFOLD using BDTs to unfold each observable independently. We compare these results to binned unfolding using IBU. For training, 1.2×10^6 events from each dataset are used, with a validation split for neural networks. The remaining events are used for testing. IBU uses the full datasets.

Acceptance effects are simulated with a $p_T^{\text{det.}} > 150$ GeV selection, causing 15% of events to fail detector-level cuts; models are trained for $n = 5$ iterations. Neural networks are trained using the Adam optimizer for 50 epochs with a learning rate of 10^{-4}.

The unfolded results are shown in Fig. 1. All methods achieve good agreement with the truth distributions, typically within $\pm 15\%$. Methods that simultaneously unfold multiple observables (OMNIFOLD, MULTIFOLD) generally outperform single-observable approaches (UNIFOLD, IBU).

Performance differences are quantified using the triangular discriminator metric, defined as $\frac{1}{2}\sum_i (p_i - q_i)^2/(p_i + q_i)$, where p_i and q_i are the truth and unfolded distributions in bin i. The results in Table 1 show that ML-based approaches consistently improve agreement with the truth compared to IBU, with the best-performing method depending on the observable.

We did not explicitly measure computational costs, but BDTs train on CPUs while DNNs and PET use GPUs, making BDTs more accessible and efficient. The unbinned unfolding methods took a few hours to train with the BDTs training faster, while IBU took less than a minute to finish.

Table 1: Unfolding performance using the triangular discriminator metric. Lower values indicate better agreement with the truth distribution.

	Observable					
Method	m	M	w	$\ln\rho$	τ_{21}	z_g
OMNIFOLD PET	**0.79**	**0.42**	0.81	0.66	1.97	0.74
MULTIFOLD DNN	2.38	0.46	0.35	**0.20**	**0.63**	0.47
MULTIFOLD BDT	2.64	0.53	**0.31**	0.22	0.92	0.74
UNIFOLD BDT	9.51	1.44	0.46	0.73	1.11	**0.44**
IBU	9.74	1.46	0.48	0.89	1.15	0.45
Data	24.59	129.92	15.87	14.70	11.12	3.86
Generation	3.46	14.94	22.39	18.97	20.70	3.99

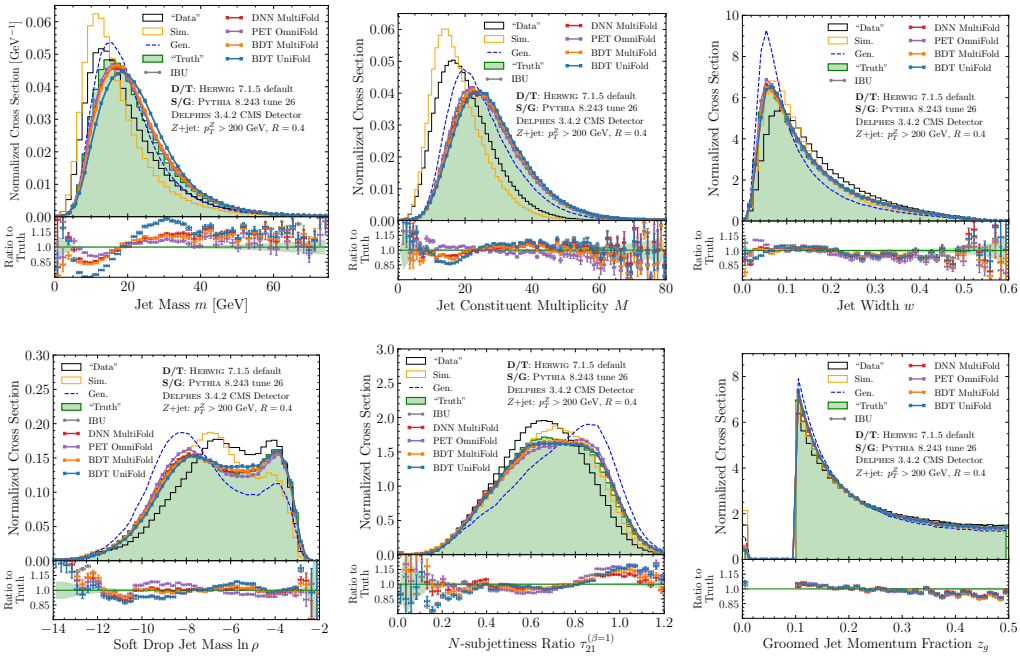

Figure 1: Comparison of unfolded jet observables. HERWIG is used as the "Data"/"Truth", while PYTHIA is the simulation and generation (Sim./Gen.). The events shown here are from the test datasets, which contain about 400k events. The bottom panels show the ratios of the unfolded distributions to the truth data in each bin. The uncertainties in each bin are the quadrature sum of the weights for events in that bin (the weights are 1 for non-unfolded results), divided by the truth bin entries. The uncertainty band shows $1\pm$ the truth uncertainty.

4 Conclusions

We present two software implementations of OmniFold: a RooUnfold-integrated framework and a standalone Python package. We show that boosted decision trees achieve performance comparable to neural networks for low-dimensional problems at lower computational cost. While our results are limited to jets, we expect our tools to be effective with many datasets. By incorporating acceptance effects, we demonstrate that these methods robustly handle realistic experimental conditions. These developments aim to broaden adoption of unbinned unfolding techniques and to support future methodological advances in data analysis across physics and related fields.

5 Code and paper availability

The code for the BDT implementation of OmniFold and for the studies in this paper are available at `https://github.com/rymilton/unbinned_unfolding/`. The deep neural network and Point-Edge Transformer implementations of OmniFold are available in the omnifold PyPI package. The published paper associated with this work is Reference [37].

References

[1] G. Cowan. A survey of unfolding methods for particle physics. *Conf. Proc.*, C0203181:248, 2002.

[2] Volker Blobel. Unfolding Methods in Particle Physics. *PHYSTAT2011 Proceedings*, page 240, 2011. doi: 10.5170/CERN-2011-006.

[3] Volker Blobel. Unfolding. *Data Analysis in High Energy Physics*, page 187, 2013. doi: 10.1002/9783527653416.ch6. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9783527653416.ch6`.

[4] Rahul Balasubramanian, Lydia Brenner, Carsten Burgard, Glen Cowan, Vincent Croft, Wouter Verkerke, and Pim Verschuuren. Statistical method and comparison of different unfolding techniques using RooFit. 2019.

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

[6] Andreas Hocker and Vakhtang Kartvelishvili. SVD approach to data unfolding. *Nucl. Instrum. Meth.*, A372:469–481, 1996. doi: 10.1016/0168-9002(95)01478-0.

[7] Stefan Schmitt. TUnfold: an algorithm for correcting migration effects in high energy physics. *JINST*, 7:T10003, 2012. doi: 10.1088/1748-0221/7/10/T10003.

[8] V. Andreev et al. Measurement of Lepton-Jet Correlation in Deep-Inelastic Scattering with the H1 Detector Using Machine Learning for Unfolding. *Phys. Rev. Lett.*, 128:132002, Mar 2022. doi: 10.1103/PhysRevLett.128.132002. URL `https://link.aps.org/doi/10.1103/PhysRevLett.128.132002`.

[9] H1 Collaboration. Machine learning-assisted measurement of multi-differential lepton-jet correlations in deep-inelastic scattering with the H1 detector. *H1prelim-22-031*, 2022. URL `https://www-h1.desy.de/h1/www/publications/htmlsplit/H1prelim-22-031.long.html`.

[10] V. Andreev et al. Unbinned deep learning jet substructure measurement in high Q2 ep collisions at HERA. *Physics Letters B*, 844:138101, 2023. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2023.138101. URL `https://www.sciencedirect.com/science/article/pii/S0370269323004355`.

[11] H1 Collaboration. Machine Learning-Assisted Measurement of Lepton-Jet Azimuthal Angular Asymmetries in Deep-Inelastic Scattering at HERA. 12 2024.

[12] R. Aaij et al. Multidifferential study of identified charged hadron distributions in Z-tagged jets in proton-proton collisions at $\sqrt{s} = 13$ TeV. *Phys. Rev. D*, 108:L031103, Aug 2023. doi: 10.1103/PhysRevD.108.L031103. URL `https://link.aps.org/doi/10.1103/PhysRevD.108.L031103`.

[13] G. Aad et al. Simultaneous Unbinned Differential Cross-Section Measurement of Twenty-Four $Z +$ jets Kinematic Observables with the ATLAS Detector. *Phys. Rev. Lett.*, 133:261803, Dec 2024. doi: 10.1103/PhysRevLett.133.261803. URL `https://link.aps.org/doi/10.1103/PhysRevLett.133.261803`.

[14] Georges Aad et al. Measurement of jet track functions in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. 2 2025.

[15] Measurement of event shapes in minimum bias events from pp collisions at 13 TeV. Technical report, CERN, Geneva, 2024. URL `https://cds.cern.ch/record/2899591`.

[16] Youqi Song. Measurement of CollinearDrop jet mass and its correlation with SoftDrop groomed jet substructure observables in $\sqrt{s} = 200$ GeV pp collisions by STAR. 7 2023.

[17] Tanmay Pani. Generalized angularities measurements from STAR at $\sqrt{}$SNN = 200 GeV. *EPJ Web Conf.*, 296:11003, 2024. doi: 10.1051/epjconf/202429611003.

[18] Tim Adye. Unfolding algorithms and tests using RooUnfold. In *PHYSTAT 2011*, pages 313–318, Geneva, 2011. CERN.

[19] Lydia Brenner, Rahul Balasubramanian, Carsten Burgard, Wouter Verkerke, Glen Cowan, Pim Verschuuren, and Vincent Croft. Comparison of unfolding methods using RooFitUnfold. *Int. J. Mod. Phys. A*, 35(24):2050145, 2020. doi: 10.1142/S0217751X20501456.

[20] Anders Andreassen, Patrick T. Komiske, Eric M. Metodiev, Benjamin Nachman, and Jesse Thaler. OmniFold: A Method to Simultaneously Unfold All Observables. *Phys. Rev. Lett.*, 124: 182001, 2020. doi: 10.1103/PhysRevLett.124.182001.

[21] Anders Andreassen, Patrick T. Komiske, Eric M. Metodiev, Benjamin Nachman, Adi Suresh, and Jesse Thaler. Scaffolding Simulations with Deep Learning for High-dimensional Deconvolution. In *9th International Conference on Learning Representations*, 5 2021.

[22] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79:745, June 1974. doi: 10.1086/111605.

[23] William Hadley Richardson. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am.*, 62(1):55–59, Jan 1972. doi: 10.1364/JOSA.62.000055. URL `http://www.osapublishing.org/abstract.cfm?URI=josa-62-1-55`.

[24] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, 2022. URL `https://arxiv.org/abs/2207.08815`.

[25] Thorben Finke, Marie Hein, Gregor Kasieczka, Michael Krämer, Alexander Mück, Parada Prangchaikul, Tobias Quadfasel, David Shih, and Manuel Sommerhalder. Tree-based algorithms for weakly supervised anomaly detection. *Phys. Rev. D*, 109(3):034033, 2024. doi: 10.1103/PhysRevD.109.034033.

[26] Marat Freytsis, Maxim Perelstein, and Yik Chuen San. Anomaly detection in the presence of irrelevant features. *JHEP*, 02:220, 2024. doi: 10.1007/JHEP02(2024)220.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] François Chollet et al. Keras. `https://keras.io`, 2015.

[29] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`.

[30] Vinicius Mikuni and Benjamin Nachman. OmniLearn: A Method to Simultaneously Facilitate All Jet Physics Tasks. 4 2024.

[31] Anders Andreassen, Patrick Komiske, Eric Metodiev, Benjamin Nachman, and Jesse Thaler. Pythia/Herwig + Delphes Jet Datasets for OmniFold Unfolding. *Zenodo*, 2019. doi: 10.5281/zenodo.3548091.

[32] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. An introduction to PYTHIA 8.2. *Comput. Phys. Commun.*, 191:159–177, 2015. doi: 10.1016/j.cpc.2015.01.024.

[33] Johannes Bellm et al. Herwig 7.1 Release Note. 5 2017.

[34] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. DELPHES 3, A modular framework for fast simulation of a generic collider experiment. *JHEP*, 02:057, 2014. doi: 10.1007/JHEP02(2014)057.

[35] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The anti-k_t jet clustering algorithm. *JHEP*, 04:063, 2008. doi: 10.1088/1126-6708/2008/04/063.

[36] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet User Manual. *Eur. Phys. J. C*, 72: 1896, 2012. doi: 10.1140/epjc/s10052-012-1896-2.

[37] Ryan Milton, Vinicius Mikuni, Trevin Lee, Miguel Arratia, Tanvi Wamorkar, and Benjamin Nachman. Tools for unbinned unfolding. *JINST*, 20(05):P05034, 2025. doi: 10.1088/1748-0221/20/05/P05034.