A High-Dimensional and Unbinned SM Measurement with the ATLAS Detector

Mariel Pettee, Ben Nachman, & Dag Gillberg

February 26th, 2025

ATLAS EXPERIMENT

BERKELEY LAB
Bringing Science Solutions to the World

"**Unfolding**" means removing unwanted detector effects from our experimental data.

This has the advantage of correcting a whole dataset on a statistical level, **making the data more flexible & useful for future analyses.**

In contemporary data analysis for fundamental physics, we want more from our unfolding strategies.

Traditional unfolding approaches (binned & few-dimensional) have significant room for improvement.

Specifically:

- Any future user of unfolded data would ideally want to **choose their own bins** for their measurement, but this is not possible with binned strategies in which bins are pre-determined.
- A future user of unfolded data might want to **modify the phase space,** but this is basically not possible with binned measurements.
- Measurements of **properties that are a function of many observables** are not possible with a few-dimensional unfolding.

Our recent result using the full ATLAS Run 2 dataset

Phys.Rev.Lett. 133 (2024) 26, 261803
DOI: 10.1103/PhysRevLett.133.261803

CERN-EP-2024-132
February 10, 2025

A simultaneous unbinned differential cross section measurement of twenty-four Z+jets kinematic observables with the ATLAS detector

The ATLAS Collaboration

The 24 observables describe Z+jets kinematics & properties of jet substructure.

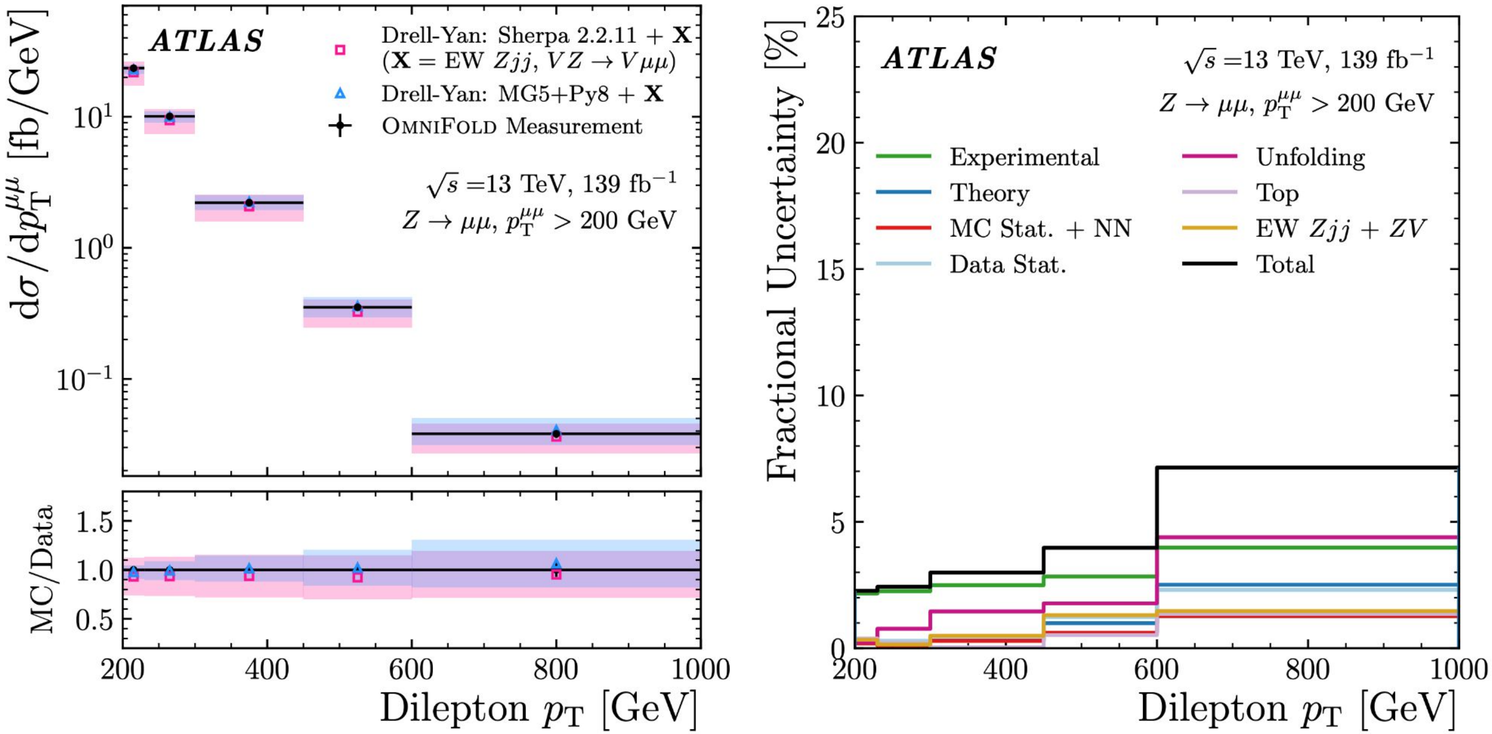
(Ultimately, our goal is to enable high-dimensional precision QCD studies.)

- Leading & sub-leading jet: p_T, y, ϕ, τ_1, τ_2, τ_3, m, $n_{\text{charged tracks}}$
- Leading & sub-leading muon: p_T, η, ϕ
- Di-muon system: p_T, y

	pT_ll	pT_l1	pT_l2	eta_l1	...	weights_trackPtScale	weights_theoryPSjet	weights_theoryPSsoft
0	479.442780	288.466919	198.183929	-0.117443	...	0.003174	0.002844	0.003195
1	274.524994	166.120789	125.378044	0.313321	...	0.008168	0.008563	0.008236
2	462.713226	335.697479	133.157684	0.766387	...	0.001638	0.001724	0.001890
3	215.157608	189.518021	25.711994	1.083798	...	0.004669	0.004622	0.004648
4	222.458313	128.850159	108.589226	-0.635713	...	0.002102	0.002417	0.002129
...	...	...	...	...	...	...	...	...
418009	934.971924	738.464722	196.525192	0.102944	...	0.000069	0.000070	0.000061
418010	245.813461	166.847061	93.757919	1.308837	...	0.000193	0.000189	0.000203
418011	478.670349	378.737518	108.016479	-0.328871	...	0.001969	0.001813	0.001825
418012	278.586029	249.255356	43.581135	0.632484	...	0.003238	0.003101	0.003090
418013	244.505249	219.796280	40.357105	1.833223	...	0.000947	0.000968	0.000957

The measurement is a 24-dimensional object.
Here is one of the unbinned differential cross-sections:

Dilepton p_T



But we can go beyond just measuring the 24 input variables...
we can also imagine **brand new observables** that we want to measure,
and even probe **different bins or regions of phase space**.

Let's construct some new observables...

$$\tau_{21} = \tau_2/\tau_1$$

This ratio of two parameters measuring jet substructure is useful for e.g. W vs. QCD jet classification:

$$\tau_N = \frac{1}{d_0}\sum_k p_{T,k} \min_A \{\Delta R_{A,k}\}$$

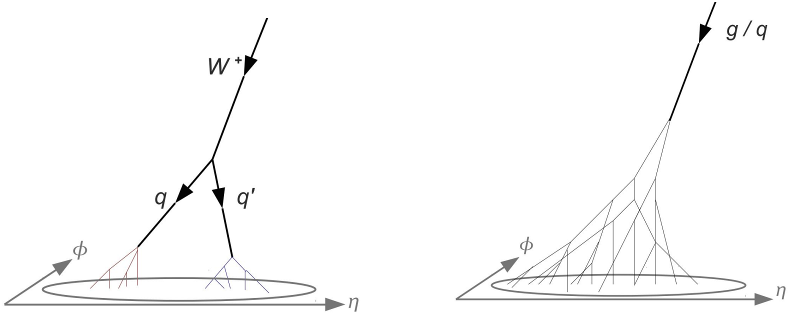
Sum over constituents

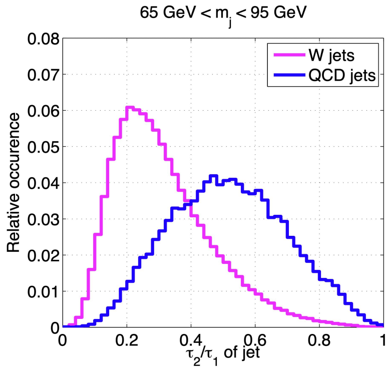
Minimize distance to candidate subjet axes

Jet shape that "counts" number of subjets!

# subjects:	$\leq$ N	> N
τ_N:	0	1

Adapted from "N-jettiness" [Stewart, Tackmann, Waalewijn: 1004.2489]



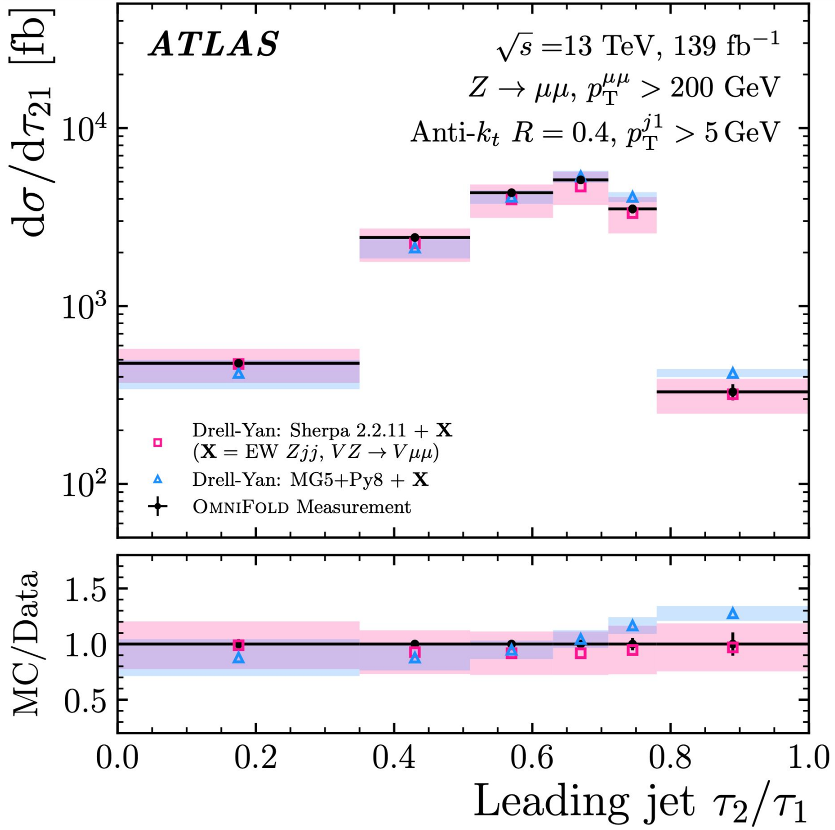


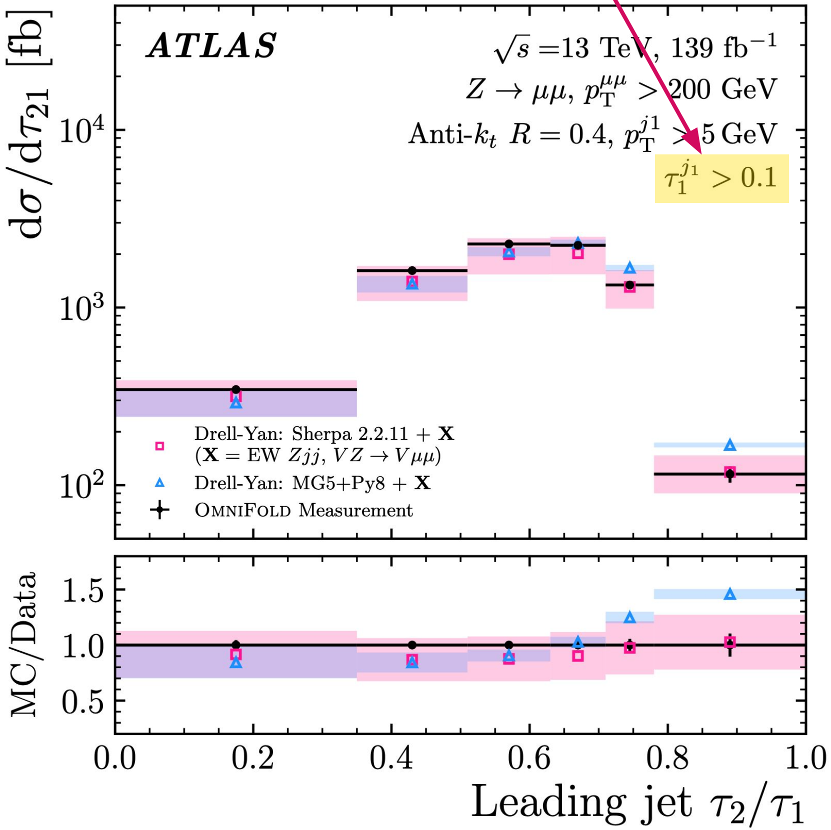
Images from J. Thaler's talk and paper from 2011

Let's construct some new observables...

IRC-safe phase space!

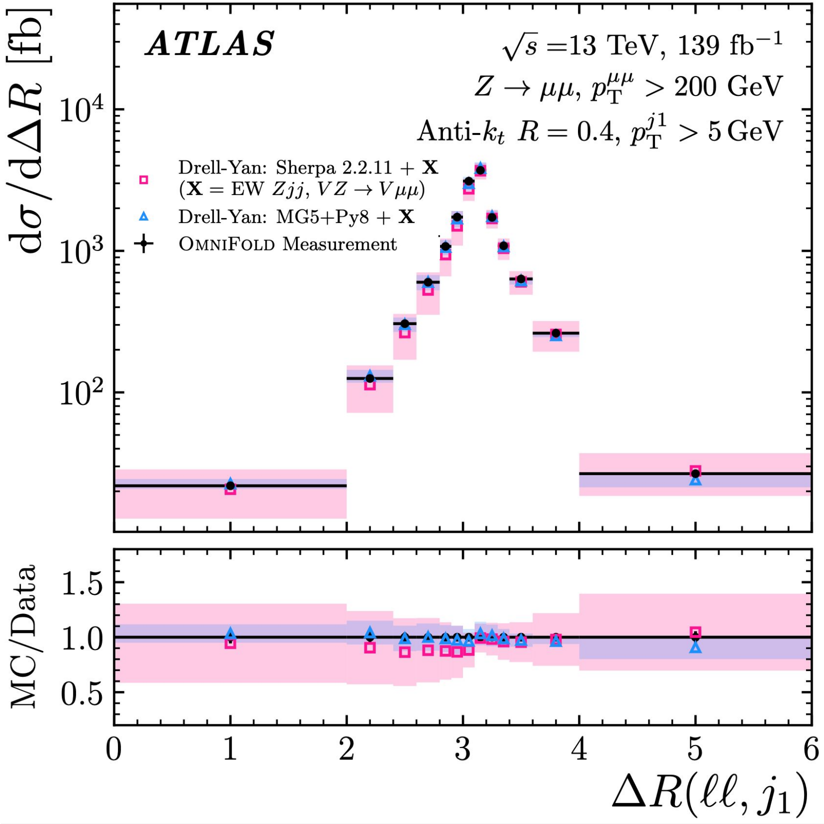
$$\tau_{21} = \tau_2/\tau_1$$



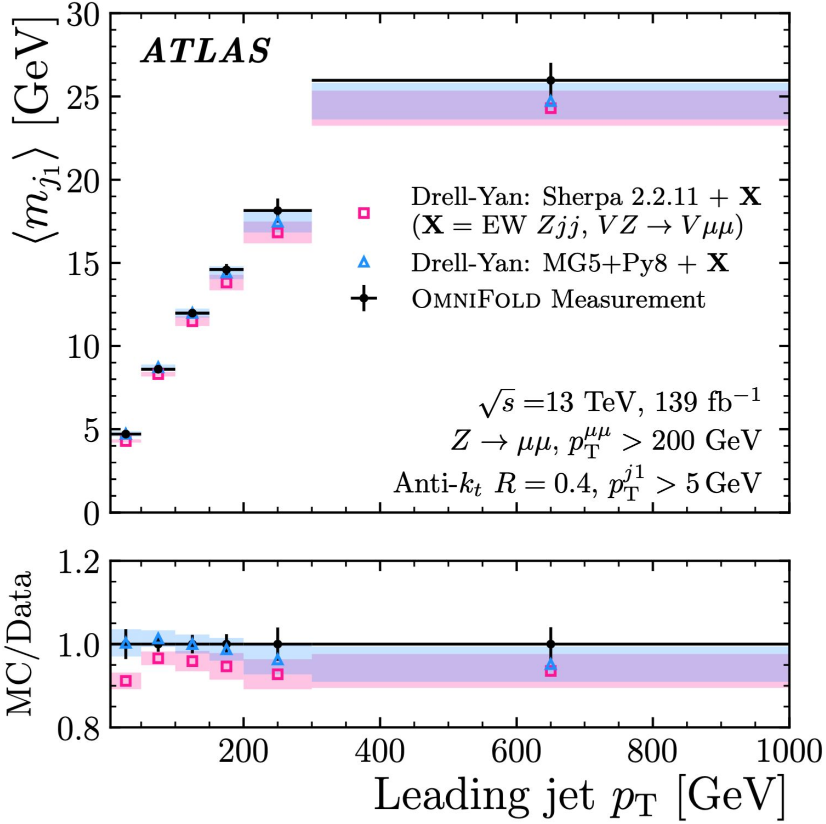


Let's construct some new observables...

$\Delta R(\ell\ell, j_1)$



Average m_{j1} in bins of leading jet p_T



Datasets & Jupyter notebooks

Our measurements are published here on Zenodo: https://zenodo.org/records/11507450

zenodo

Search records...

Communities My dashboard

mpettee@l...

The ATLAS Experiment at CERN

Published June 6, 2024 | Version v1

Dataset Open

Edit

New version

Share

ATLAS OmniFold 24-Dimensional Z+jets Open Data

ATLAS Collaboration

These datasets contain the unbinned, twenty-four-dimensional ATLAS Z+jets differential cross-section measurement presented in CERN-EP-2024-132. The measurements are presented as Pandas DataFrames in HDF5 format, and they are accompanied by MC predictions formatted as Numpy arrays. Measurements are provided both for "pseudo-data", i.e. a validation MC sample with truth- and reco-level quantities that has been reweighted to match data, as well as real data.

Important: Before using this data, please consult the documentation & example notebooks.

The signal process is inclusive Z→μμ production with a fiducial region defined in the boosted regime: p_T^μμ > 200 GeV.

In total, 24 Z+jets kinematic observables are measured:

- p_T, η, and ϕ of each of the two muons (6 observables)
- The p_T and rapidity of the dimuon system: p_T^μμ, y^μμ (2 observables)
- The 4-momenta (p_T, y, ϕ, m) of the two leading charged particle jets (8 observables)
- The number of (charged) constituents and n-subjettiness quantities τ_1, τ_2, and τ_3 for each of the same two jets (8 observables)

The dimuon system p_T and y can be obtained from the muon kinematics, but they are included for convenience. The observables are labeled by 1 and 2 for leading and subleading in p_T, respectively.

64 VIEWS

36 DOWNLOADS

Show more details

Versions

Version v1 Jun 6, 2024

10.5281/zenodo.11507450

Cite all versions? You can cite all versions by using the DOI 10.5281/zenodo.11507449. This DOI represents all versions, and will always resolve to the latest one. Read more.

We have also published a detailed README & several Jupyter notebooks with instructions about how to use the public datasets:

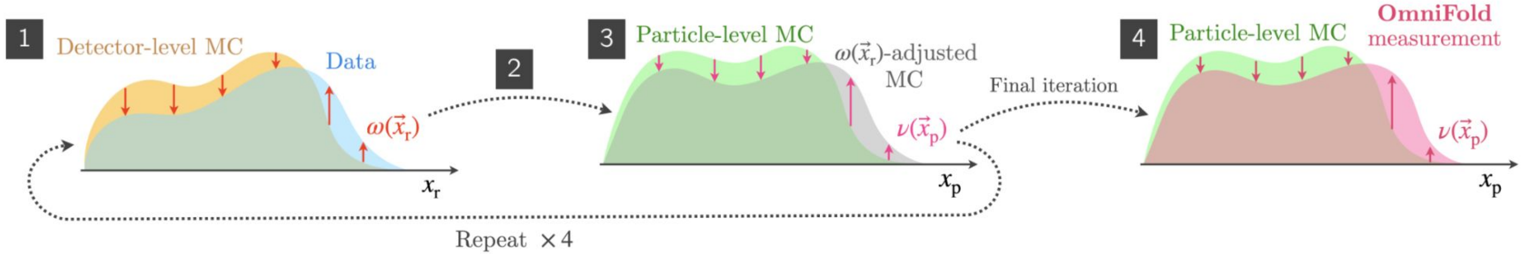
https://gitlab.cern.ch/atlas-physics/public/sm-z-jets-omnifold-2024

1_basics.ipynb	Open in Colab
2_pseudo_results.ipynb	Open in Colab
3_results.ipynb	Open in Colab

A simultaneous unbinned differential cross section measurement of twenty-four Z+jets kinematic observables with the ATLAS detector

CDS CERN-EP-2024-132 arXiv 2405.20041 DOI 10.5281/zenodo.11507450

These notebooks demonstrate how to interact with the unbinned, twenty-four-dimensional ATLAS Z+jets differential cross-section measurement presented in arXiv:2405.20041. This analysis uses OmniFold to mitigate detector effects in data.



Since the data is structured as an unbinned set of events, users can:

- Re-create the differential cross-section distributions (and calculate the associated uncertainties) of the twenty-four measured input observables with a **flexible choice of binnings** (see Fig. 1 below)
- **Modify the measured phase space** on-the-fly (see Fig. 2a & 2b below)
- **Measure new observables** or quantities constructed as a function of the input observables (see Fig. 2 below)

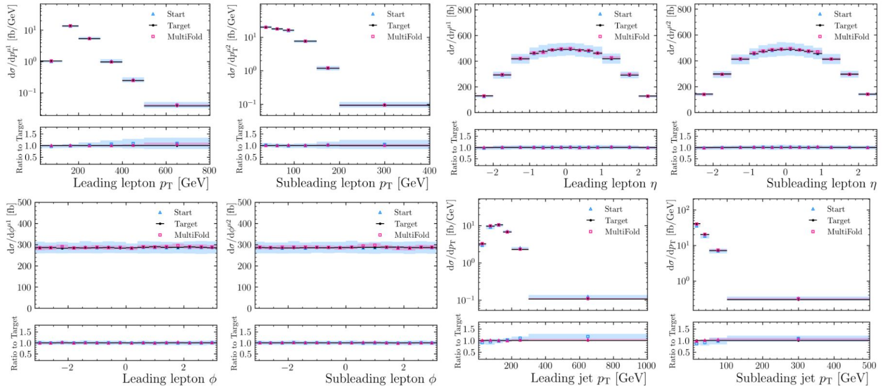
Look at closure of pseudo-data with the known targets:

```
    axs[1].errorbar(bin_centers, np.ones(len(bin_centers)), xerr=bin_widths/2, yerr=uncertainties[var+"_total"]/100, marker=".", linestyle="None", colo
    _ = make_error_boxes(axs[1], bin_centers, start_density/target_density, np.vstack([bin_widths/2,bin_widths/2]), np.vstack([(start_density/target_de
    _ = make_error_boxes(axs[1], bin_centers, multifold_density/target_density, np.vstack([bin_widths/2,bin_widths/2]), np.vstack([(multifold_density/t
    axs[1].set_xlim(bins[0], bins[-1])
    axs[1].set_ylim([0.2,1.8])
    axs[1].xaxis.set_tick_params(labelsize=12, which='both', direction='in', top=True)
    axs[1].yaxis.set_tick_params(labelsize=12, which='both', direction='in', right=True)
    axs[1].set_ylabel('Ratio to Target', fontsize=10, labelpad=2, loc='center')
    axs[1].set_xlabel(plot_labels[var], fontsize=16, labelpad=2, loc='right');

plt.savefig(os.path.join(plot_dir,"nominal_diff_xsec.pdf"))
```

Out [8]:

```
Plots: 100%|██████████| 24/24 [00:55<00:00,  2.31s/it]
```



Calculate p-values:

```
    chi_2 = D.dot(np.linalg.inv(v_total_decorr)).dot(D.T)
    p_value = 1 - stats.chi2.cdf(chi_2, dof)

  print(f"{var:<20} dof: {dof:<7} x2: {chi_2:.5f} \t p value: {p_value:.4f}")
```

Out [14]:

```
pT_l1               dof: 6       x2: 11.01103    p value: 0.0880
pT_l2               dof: 6       x2: 10.68616    p value: 0.0986
eta_l1              dof: 14      x2: 18.19845    p value: 0.1979
eta_l2              dof: 14      x2: 12.03006    p value: 0.6039
phi_l1              dof: 16      x2: 19.79036    p value: 0.2298
phi_l2              dof: 16      x2: 13.41401    p value: 0.6423
pT_trackj1          dof: 6       x2: 13.37187    p value: 0.0375
pT_trackj2          dof: 4       x2: 1.73601     p value: 0.7842
y_trackj1           dof: 18      x2: 10.40634    p value: 0.9178
y_trackj2           dof: 10      x2: 3.60559     p value: 0.9634
phi_trackj1         dof: 16      x2: 4.05499     p value: 0.9988
phi_trackj2         dof: 16      x2: 6.52962     p value: 0.9813
pT_ll               dof: 5       x2: 4.97659     p value: 0.4187
y_ll                dof: 14      x2: 19.70013    p value: 0.1399
Ntracks_trackj1     dof: 6       x2: 6.83001     p value: 0.3369
Ntracks_trackj2     dof: 5       x2: 2.67966     p value: 0.7492
m_trackj1           dof: 6       x2: 9.24668     p value: 0.1602
m_trackj2           dof: 4       x2: 6.91095     p value: 0.1407
tau1_trackj1        dof: 7       x2: 1.94921     p value: 0.9626
tau1_trackj2        dof: 5       x2: 3.92012     p value: 0.5610
tau2_trackj1        dof: 7       x2: 8.74004     p value: 0.2719
tau2_trackj2        dof: 5       x2: 5.93484     p value: 0.3126
tau3_trackj1        dof: 4       x2: 2.63007     p value: 0.6215
tau3_trackj2        dof: 4       x2: 5.48124     p value: 0.2414
```

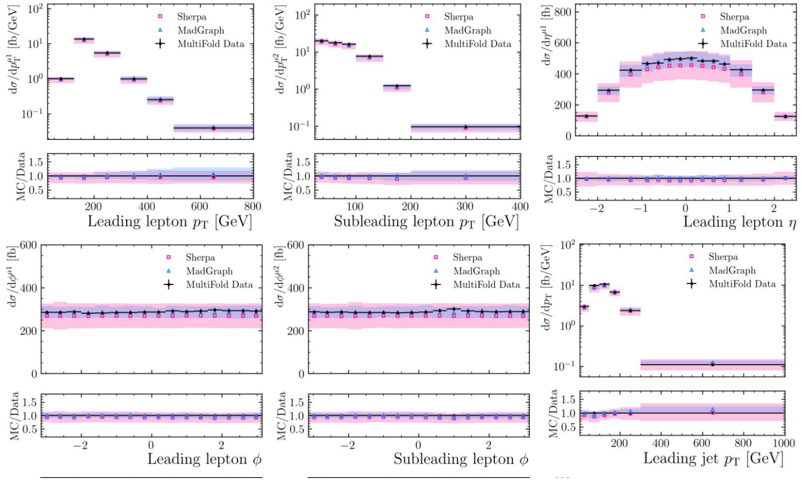
Reproduce the unfolding result and calculate uncertainties:

```
    axs[1].set_xlim(bins[0], bins[-1])
    axs[1].set_ylim([0.2,1.8])
    axs[1].xaxis.set_tick_params(labelsize=12, which='both', direction='in', top=True)
    axs[1].yaxis.set_tick_params(labelsize=12, which='both', direction='in', right=True)
    axs[1].set_ylabel('MC/Data', fontsize=12, labelpad=2, loc='center')
    axs[1].set_xlabel(plot_labels[var], fontsize=16, labelpad=2, loc='right');

plt.savefig(os.path.join(plot_dir,"nominal_diff_xsec.pdf"))
```

Out [8]:

```
Plots: 100%|██████████| 24/24 [00:03<00:00,  6.04it/s]
```

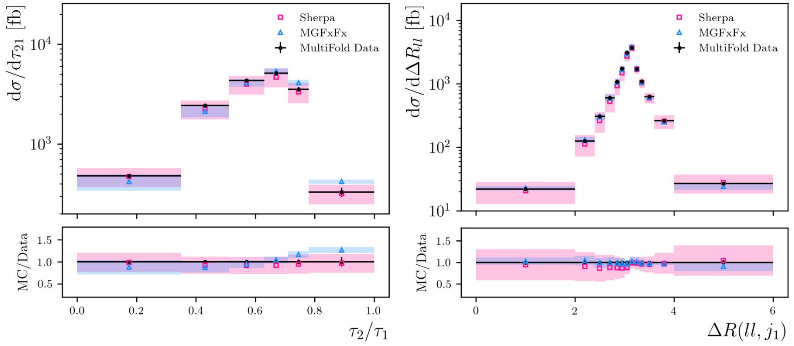


Construct derived variables:

```
    axs[1].set_ylabel('MC/Data', fontsize=10, labelpad=5)
    axs[1].set_xlabel(plot_labels[var], fontsize=14, loc='right');
plt.savefig(os.path.join(plot_dir,"derived_diff_xsec_data.pdf"))
```

Out [13]:

```
Plots: 100%|██████████| 2/2 [00:00<00:00,  4.70it/s]
```



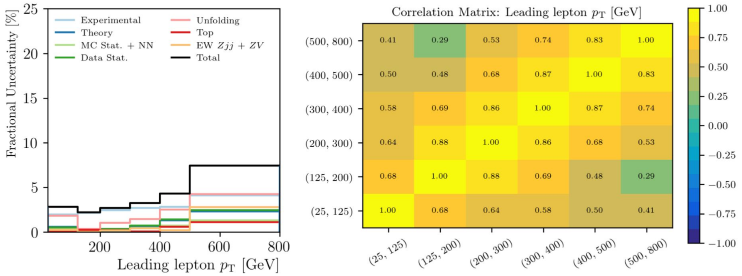
Plot uncertainty correlation matrices:

In [18]:

```
duo_plots([0, 1])
```

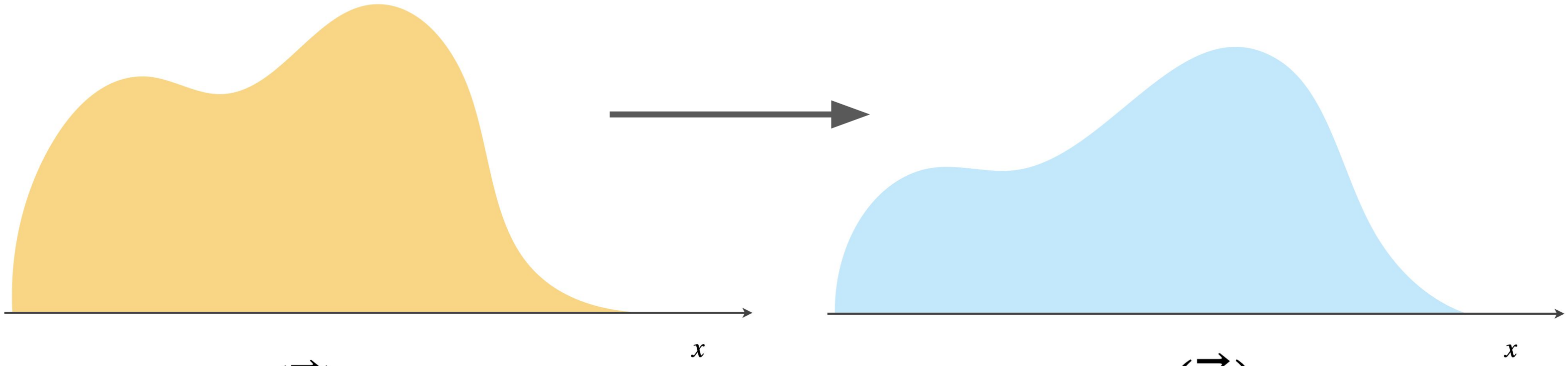
Out [18]:

```
100%|██████████| 2/2 [00:35<00:00, 17.79s/it]
```



Machine learning methodology: OmniFold

Q: How can we adjust one distribution to look like another?



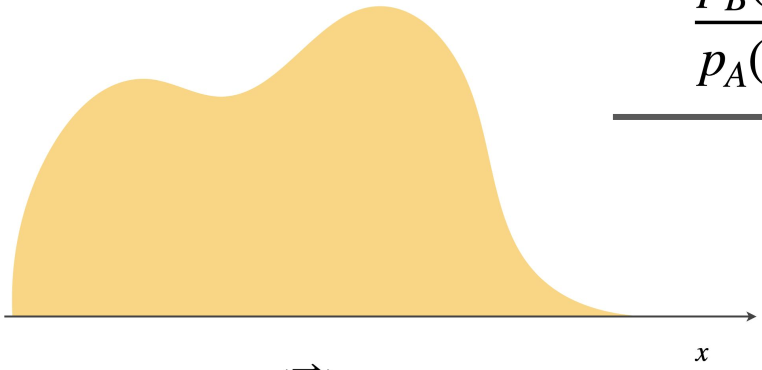
$p_A(\vec{x})$ $p_B(\vec{x})$

Q: How can we adjust one distribution to look like another?

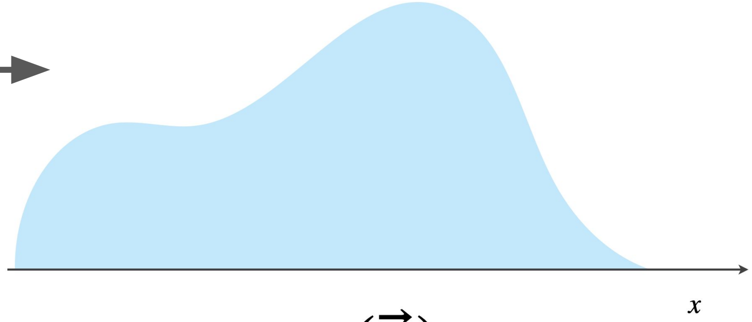
A: Learn a reweighting function based on the ratio of their probability densities.

$$\frac{p_B(\vec{x})}{p_A(\vec{x})}$$

"Likelihood ratio"



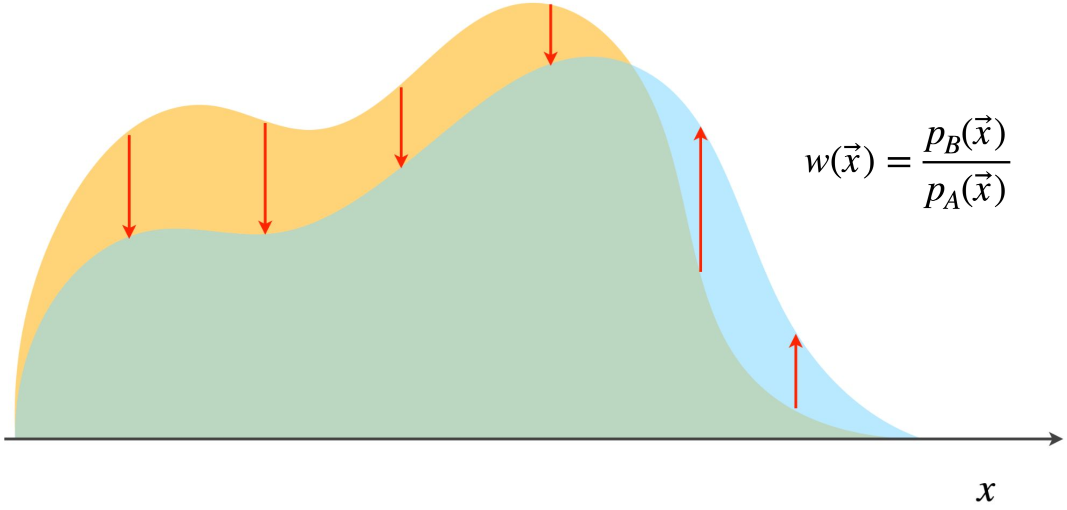
$p_A(\vec{x})$



$p_B(\vec{x})$

Q: How can we adjust one distribution to look like another?

A: Learn a reweighting function based on the ratio of their probability densities.



$$w(\vec{x}) = \frac{p_B(\vec{x})}{p_A(\vec{x})}$$

In practice, calculating individual densities $p_A(x)$ and $p_B(x)$ can be hard.

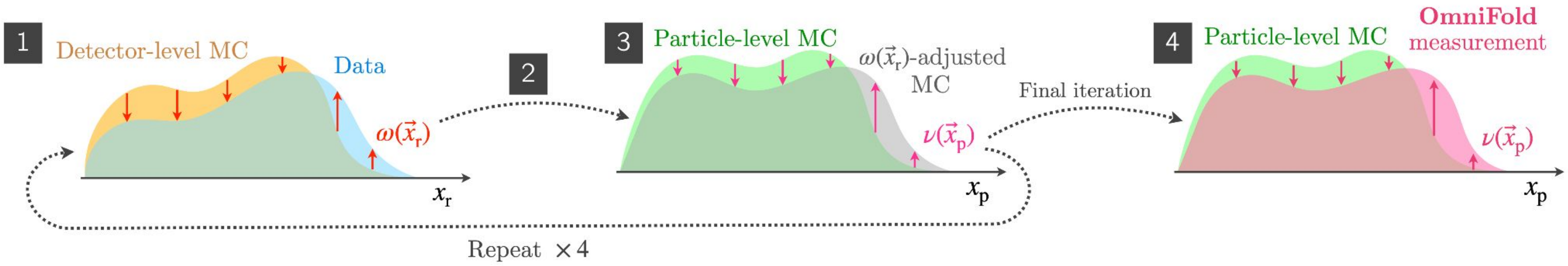
But neural network classifiers can be used to **directly approximate the ratio of the likelihoods.**

OmniFold overview

The OmniFold procedure requires two datasets* as inputs:
(*In practice, we use samples from different MC generators as well as systematically-shifted samples to determine uncertainties.)

- MC sample with events at both detector-level and particle-level
- Real data
  - (In fact, it's the only unfolding method of this kind that has been applied to real data)

In a multi-stage and iterative process, a series of neural networks are trained to learn a reweighting function that maps particle-level MC distributions to particle-level data distributions.



Note: Binned OmniFold reduces to Iterative Bayesian Unfolding (IBU)!

Each iteration consists of 2 reweighting stages.

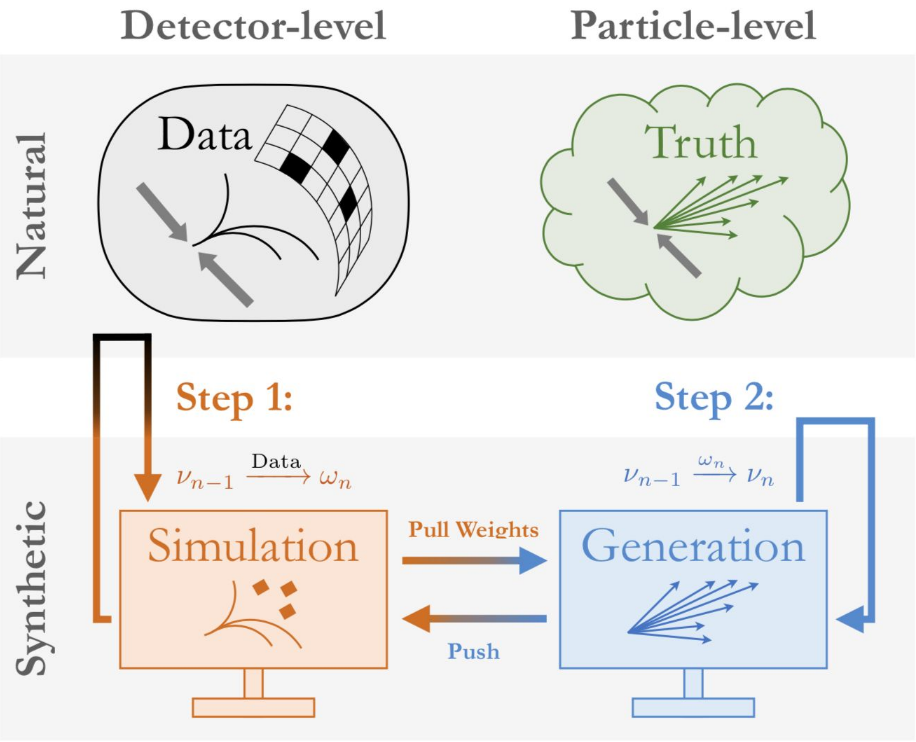
1. Reweight MC Reco to match Data Reco

2: Reweight MC Truth to match the reweighted MC Reco from Step 1

→ Ensures that if two identical particle-level events will be given the same weight, even if they are reconstructed differently

→ Can be thought of as an "averaging" step

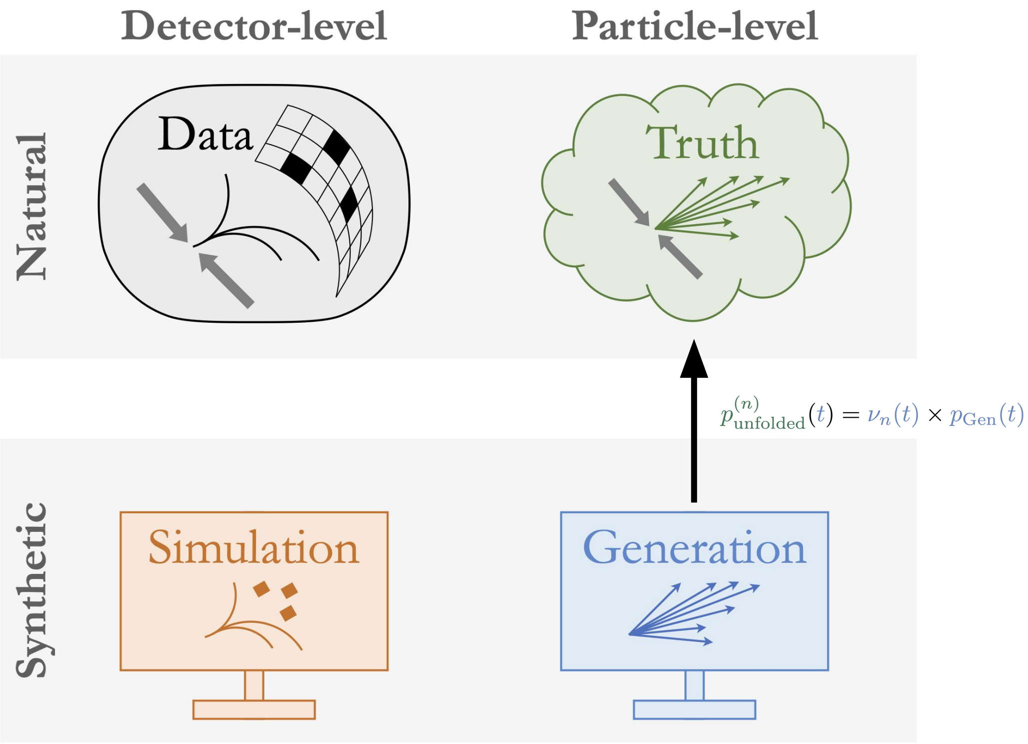
$$p^{(n)}_{\text{unfolded}}(t) = \nu_n(t) \times p_{\text{Gen}}(t)$$



Each iteration consists of 2 reweighting stages.

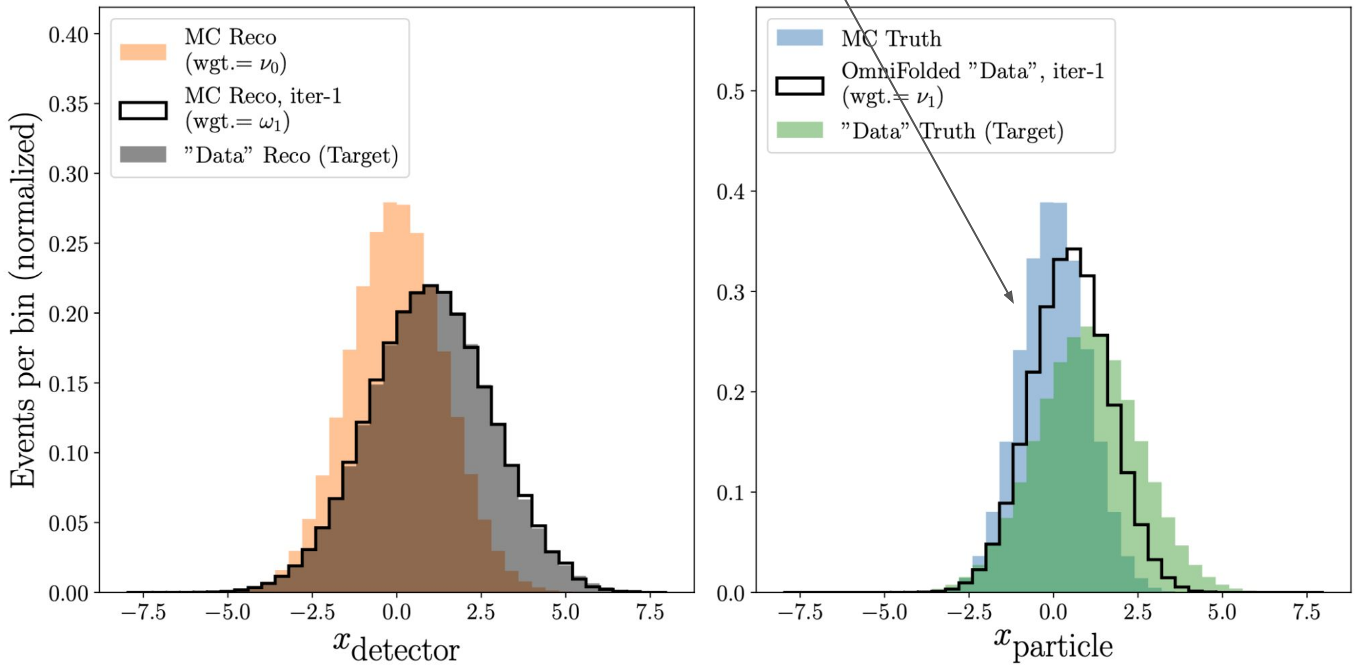
- Iterate this process multiple times to find a function that can **convert truth-level MC to unfolded data.**
- Neural networks are well-suited to this job: can process variable-length, high-dimensional inputs, and can learn a reweighting function by training a classifier & reinterpreting its outputs:

$$\frac{f^*(\vec{x})}{1 - f^*(\vec{x})} \propto \frac{p(\vec{x}|\text{dataset 1})}{p(\vec{x}|\text{dataset 2})}$$



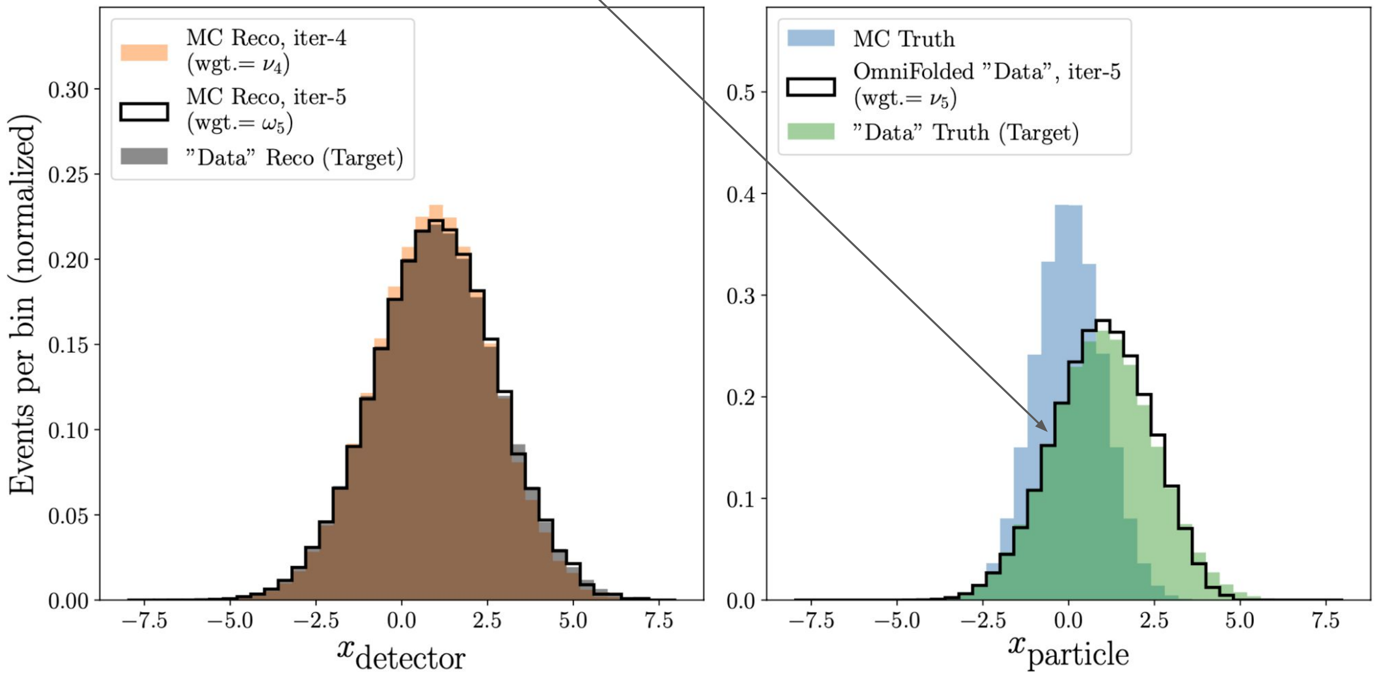
OmniFold reweights truth-level MC to estimate "truth-level" data.

After one iteration of our method, the truth-level MC is **reweighted** to look more like truth-level data.



OmniFold reweights truth-level MC to estimate "truth-level" data.

After five iterations of our method, the **reweighted** truth-level MC is closely aligned with truth-level data.



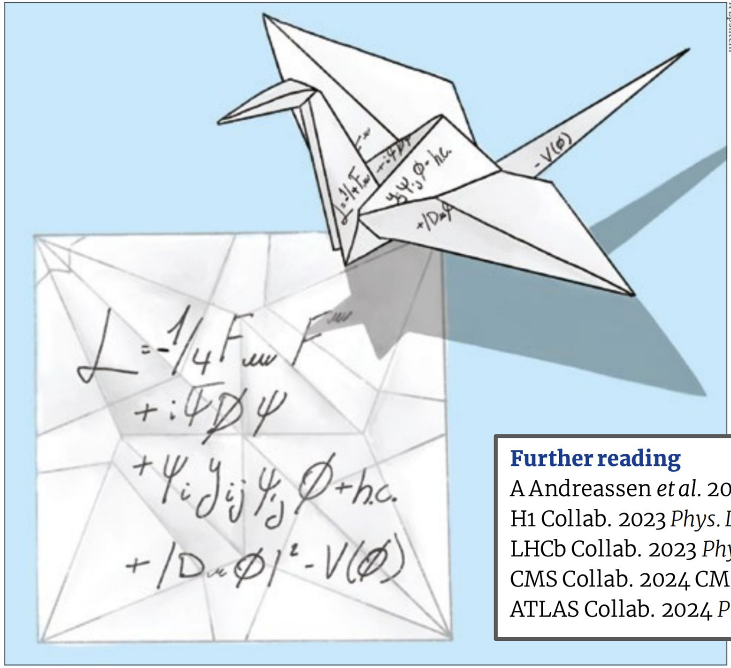
Conclusion

ML-enabled unfolding is allowing us to reimagine how we publish and re-use HEP data.

And it's not just the ATLAS Experiment...

https://cerncourier.com/wp-content/uploads/2025/01/CERNCourier2025JanFeb-digitaledition.pdf

HOW TO UNFOLD WITH AI



A Epshtein

Inspired by high-dimensional data and the ideals of open science, high-energy physicists are using artificial intelligence to reimagine the statistical technique of 'unfolding'.

Further reading

A Andreassen *et al.* 2020 *Phys. Rev. Lett.* **124** 182001.
H1 Collab. 2023 *Phys. Lett. B* **844** 138101.
LHCb Collab. 2023 *Phys. Rev. D* **108** L031103.
CMS Collab. 2024 CMS-PAS-SMP-23-008.
ATLAS Collab. 2024 *Phys. Rev. Lett.* **133** 261803.

Thanks!

- Pip-installable unfolding package:
  - `pip install omnifold` (https://pypi.org/project/unbinned-unfold/)

- CERN Courier feature article:
  https://cerncourier.com/wp-content/uploads/2025/01/CERNCourier2025JanFeb-digitaledition.pdf

- ATLAS result documentation
  - Paper:
    - arXiv: arXiv:2405.20041 [hep-ex]
    - CDS: CERN-EP-2024-132
  - Codebase:
    - https://gitlab.cern.ch/atlas-physics/public/sm-z-jets-omnifold-2024
  - Datasets:
    - https://zenodo.org/records/11507450

Backup

We apply a standard event selection for Z+jets, then restrict $p_T^{\mu\mu} > 200$ GeV.

The resulting dataset is 95% Drell-Yan, 3% diboson (ZV), and 2% EW Z+jets.

Dataset	2015+2016		2017		2018	
$p_{\mathrm{T}}^{\mu\mu}$ cut [GeV]	> 165	> 200	> 165	> 200	> 165	> 200
Data	89167	47544	103286	55132	138251	74197
Strong Z (SHERPA 2.2.1)	79670	43295	95120	51520	125280	67430
Strong Z (MG5 LO)	94430	51817	112450	61860	147310	81040
EW Zjj (POWHEG+PYTHIA8)	1207	786	1431	928	1932	1260
ZV ($V \to jj$)	1438	890	1718	1059	2261	1404
Other VV	511	308	606	367	811	497
$t\bar{t}$, single top	263	88	306	106	393	141
$W(\to \ell\nu)$, $Z(\to \tau\tau)$	3	2	0	0	3	3
Non-strong-Z	3422	2074	4061	2460	5400	3305
Non-strong-Z / data	3.8%	4.4%	3.9%	4.5%	3.9%	4.4%

Table 10: Observed and expected event yields following the event selection described in section 5.1 except using dimuon triggers instead of single muon triggers.

Notable parts of the methodology

- **NN ensembling:**
  - Neural network training introduces a small uncertainty in our result due to its **stochastic nature**.
  - We combat this by **ensembling,** i.e. taking the median result from 100 independent neural networks.
    - The final result required training ~25,000 neural networks!

- **Uncertainties:**
  - Experimental (muon efficiency & calibration, track reconstruction, pileup modelling, luminosity)
  - Theoretical (PDF & α_s variations, QCD scales, generator tune)
  - Statistical uncertainty (MC & data, estimated via bootstrapping)
  - NN uncertainty
  - Top & Non-Strong
  - Unfolding (sensitivity to particle-level shape & sensitivity to modeling of features not included in unfolding)

Validating the method

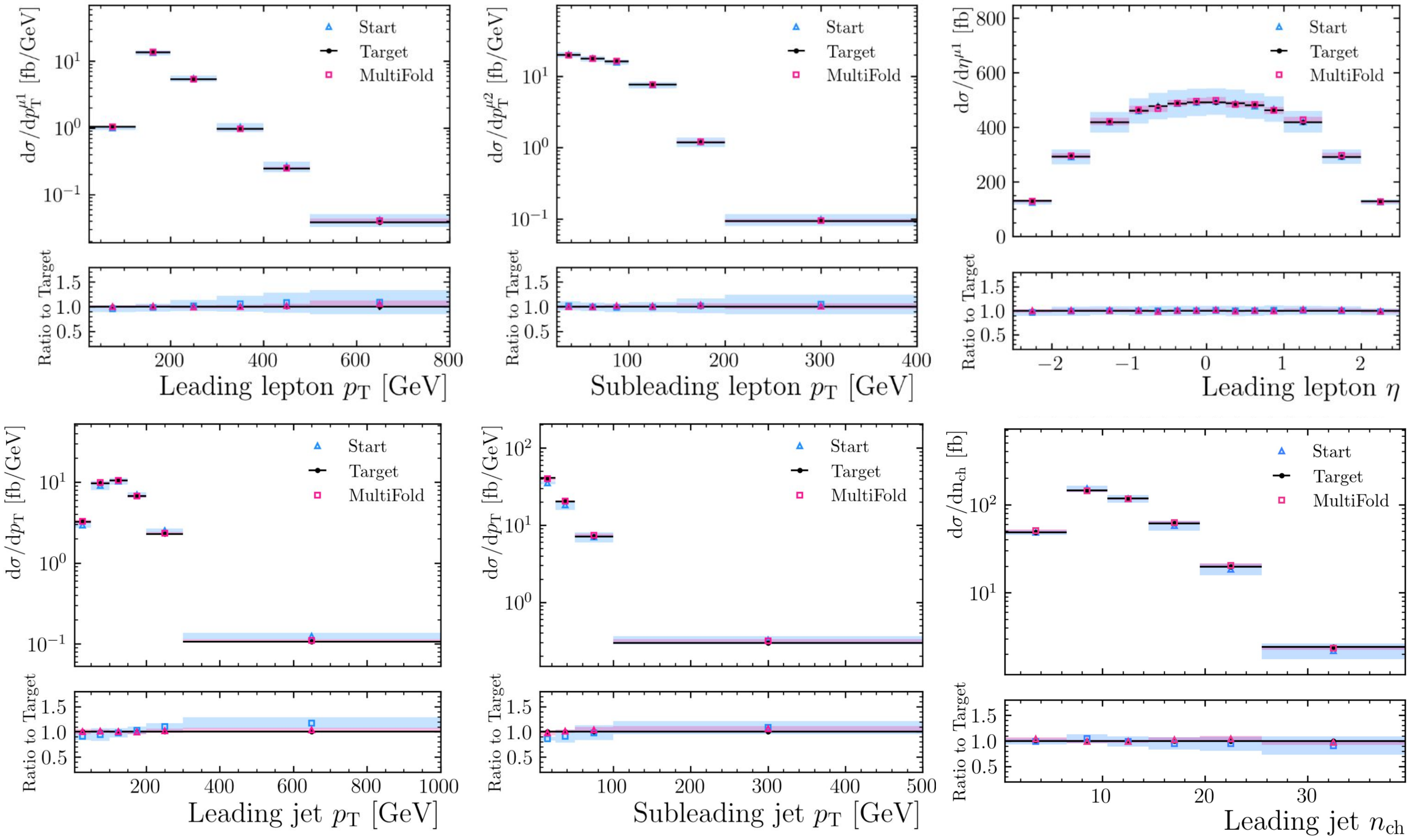
- To validate the method, we first run the full analysis using a **"pseudo-data" sample:**
  - i.e. MC simulation that has been re-weighted to match data
  - This has the advantage of letting us **compare to the truth-level target distribution**

- We run a χ^2 **compatibility test** on the final results:
  - The resulting *p*-values are $>$0.05 for 23 of the variables, but 0.038 for pT_{j1}
  - Analogous tests using IBU (each variable measured one-at-a-time) produces similar results.
  - Further validation was done in dedicated kinematic sub-sections (high dimuon p_T, EW-enhanced, and diboson-enhanced), and similar results were observed.

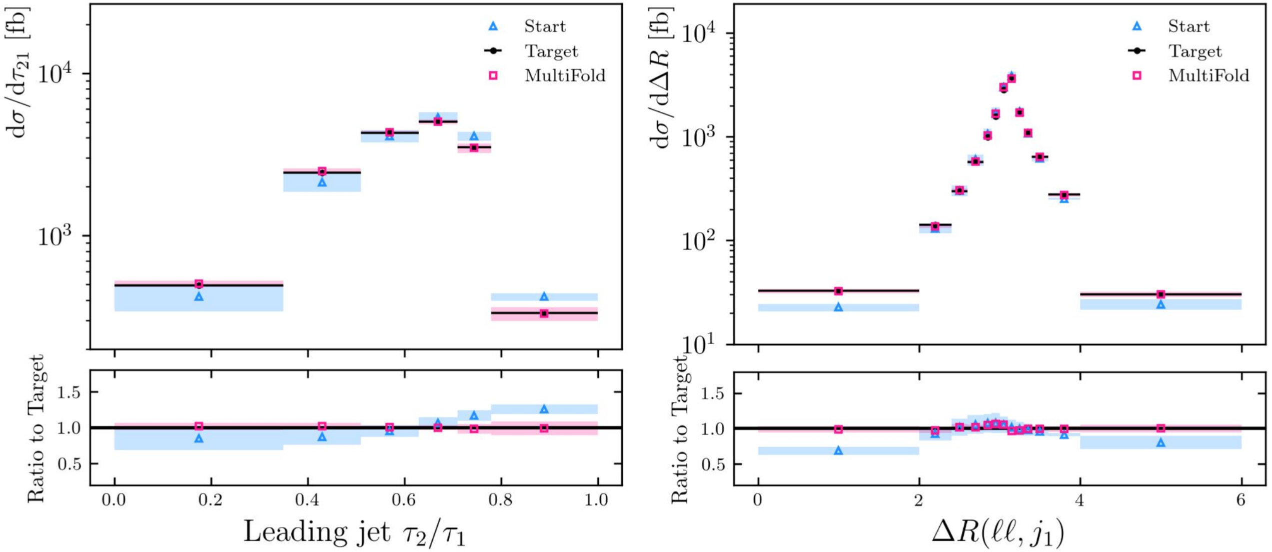
- We also ran "**stress tests**" to test OmniFold under (dramatic) distortions, and no major biases were observed.

- Using these tests, we produced a set of recommendations, e.g. ensure that $N_{eff} > 5{,}000$ events per bin and statistical uncertainty is $< 15\%$ in each bin.

We see excellent closure between the unfolded pseudo-data and the known particle-level target.



(Even for derived variables!)



Classifier functions can be re-used to directly approximate a likelihood ratio.

A vanilla NN classifying between two classes could be trained using **binary cross-entropy loss:**

$$L_{\mathrm{BCE}}[f] = -\int \mathrm{d}x \, \left(p_A(x)\,\log(f(x)) + p_B(x)\,\log(1-f(x))\right)$$

where $f(x)$ is the output of a NN classifier, and our datasets are sampled from these two probability distributions $p_A(x)$ and $p_B(x)$.

Classifier functions can be re-used to directly approximate a likelihood ratio.

A vanilla NN classifying between two classes could be trained using **binary cross-entropy loss:**

$$L_{\mathrm{BCE}}[f] = -\int \mathrm{d}x\, \left(p_A(x)\, \log(f(x)) + p_B(x)\, \log(1-f(x))\right)$$

To find where this is minimized, we need to find the extremum, i.e. differentiate with respect to $f(x)$ and set equal to 0:

$$\begin{aligned}\frac{\partial L}{\partial f} &= -\frac{\partial}{\partial f}\left(p_A(x)\, \log(f(x)) + p_B(x)\, \log(1-f(x))\right) \\ &= -\frac{p_A(x)}{f(x)} + \frac{p_B(x)}{1-f(x)}\end{aligned}$$

$$\frac{\partial L}{\partial f} = 0 \Rightarrow \frac{f(x)}{1-f(x)} = \frac{p_A(x)}{p_B(x)}$$

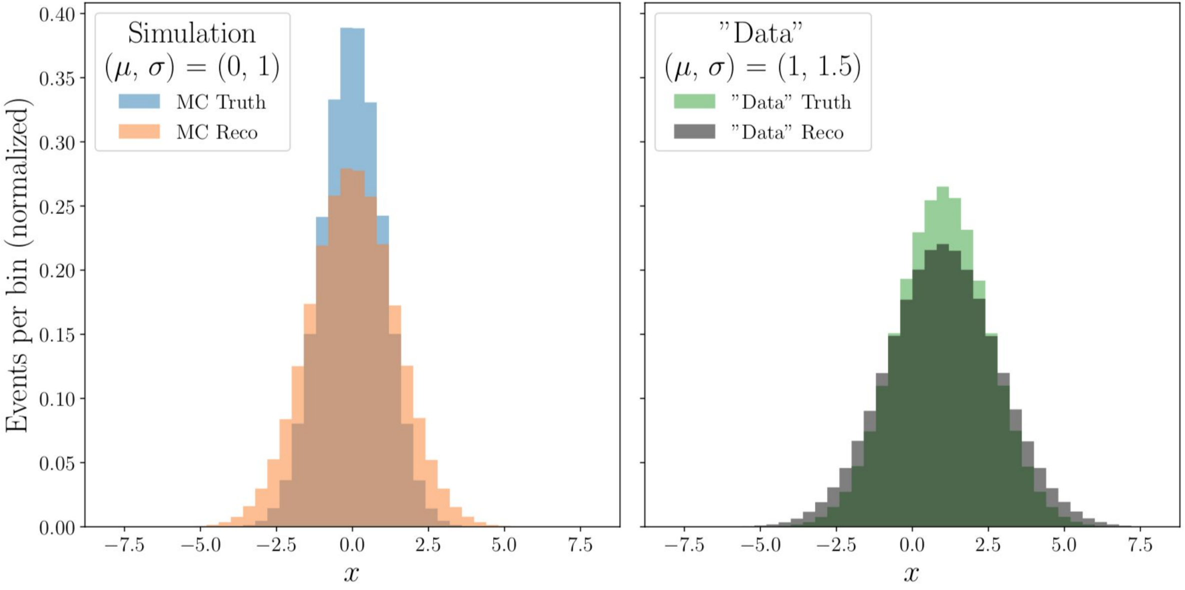
Rescaling of classifier output

Likelihood ratio

OmniFold reweights truth-level MC to estimate “truth-level” data.

Let’s say we have two datasets: our MC simulation $\mathcal{N}(\mu=0,\ \sigma=1)$ and our data $\mathcal{N}(\mu=1,\ \sigma=1.5)$.

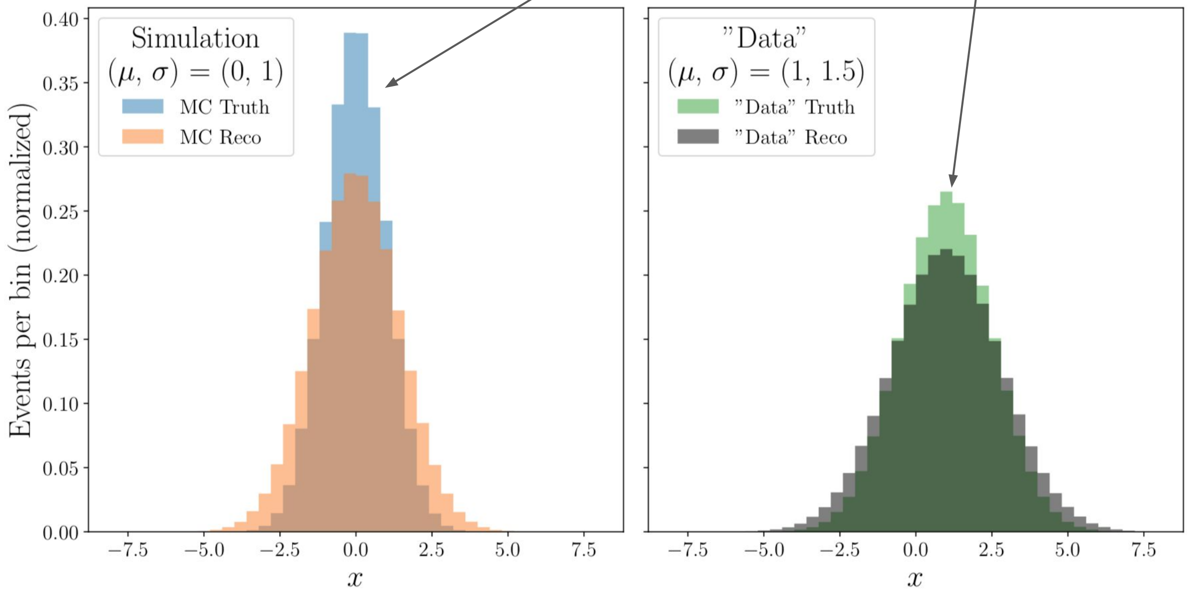
Due to detector effects, the distributions will look different at **truth-level** (sharper peaks) vs. at **reco-level.** (wider peaks)



OmniFold reweights truth-level MC to estimate "truth-level" data.

In reality, we'll never have access to the truth-level data – the best we get is the **reco-level data**.

Our goal, then, is to learn a way to transform our **truth-level MC** into **truth-level data**.



Classifier-based unfolding works well in practice and has several advantages.

In particular, it learns small corrections to MC, meaning it starts from a good baseline solution.

Maximum-likelihood classifier-based unfolding is also prior-independent.