Machine Learning Assisted Reweighting and Unfolding for Neutrino Analyses

Roger Huang

10/31/25

NPML 2025, University of Tokyo

The Likelihood Ratio “Trick”

A classifier trained to distinguish between two datasets **learns an approximation to their likelihood ratio***

Example: Train a classifier using weighted binary cross entropy loss, where each event $\mathbf{x_i}$ with weight $\mathbf{w_i}$ has a true label $\mathbf{p_i} \subset \{0, 1\}$ and gets a network prediction of $\mathbf{q_i}$:

$$\mathbf{Loss(p_i, q_i) = -w_i * (p_i * log(q_i) + (1-p_i) * log(1-q_i))}$$

If we train with dataset ***A*** with labels 1 and dataset ***B*** with labels 0, then we can reweight each of the events $\mathbf{x_i}$ in ***B*** by the likelihood ratio:

$$\mathbf{\mathscr{L}\,[A,B](x_i) = p_A(x_i) \,/\, p_B(x_i) = q_i \,/\, (1-q_i)}$$

This lets us avoid directly doing **multidimensional density estimation**, which is hard

Instead, we can just do **classification**, which is easy-ish

*J. High Energ. Phys. 2024, 136 (2024)

Measurements and Unfolding

Measure **selected number** of **events** in a **reconstructed variable** -- what the detector saw.

Efficiency | Background | Unfolding

Want the **total number** of **signal events** in a **true variable** -- what physically happened.

Assuming no background: $R_j = \sum_i^N S_{ij}\, T_i \quad \longleftrightarrow \quad T_j = \sum_j^N U_{ij}\, R_i$

Unfolding is finding the unsmearing matrix **U** given the smearing matrix **S** and removing the detector effects from the measured data (**R** in *j* bins) to get the “true” distribution (**T** in *i* bins)

Simply inverting **S** is a bad idea since it is generally ill-conditioned, as a result of very different true distributions being able to map to very similar reconstructed distributions

“Traditional” Unfolding

Several common unfolding techniques currently used for neutrino physics are:

- iterative Bayesian unfolding (aka D’Agostini)
- SVD unfolding (including Wiener SVD)
- template likelihood unfolding (e.g. recent T2K analyses)

These methods (generally) **require that the distributions are binned**, and work best with a **small set of variables** (around 1 to 4)

However many of the **corrections (e.g. efficiency) can have high-dimensional dependence**, and this is difficult to capture with only a few variables

In all cases the reconstructed MC distribution is reweighted to better match the data, and this is propagated to the truth MC distribution

OmniFold Concept: ML reweighting

With some given generator and detector simulation, we can train classifiers using the likelihood ratio trick to do **event-by-event reweighting** of the generated events to fit the observed data

- ML-based classifiers are effectively unrestricted in the number of variables they can use in decisions, allowing us to unfold in very high dimensional space

Automatically get background subtraction and efficiency correction

From the reweighted generator events, we can then extract **unbinned unfolded results of any observable**

More info in original paper PRL 124 (2020) 182001
and code release https://github.com/hep-lbdl/OmniFold

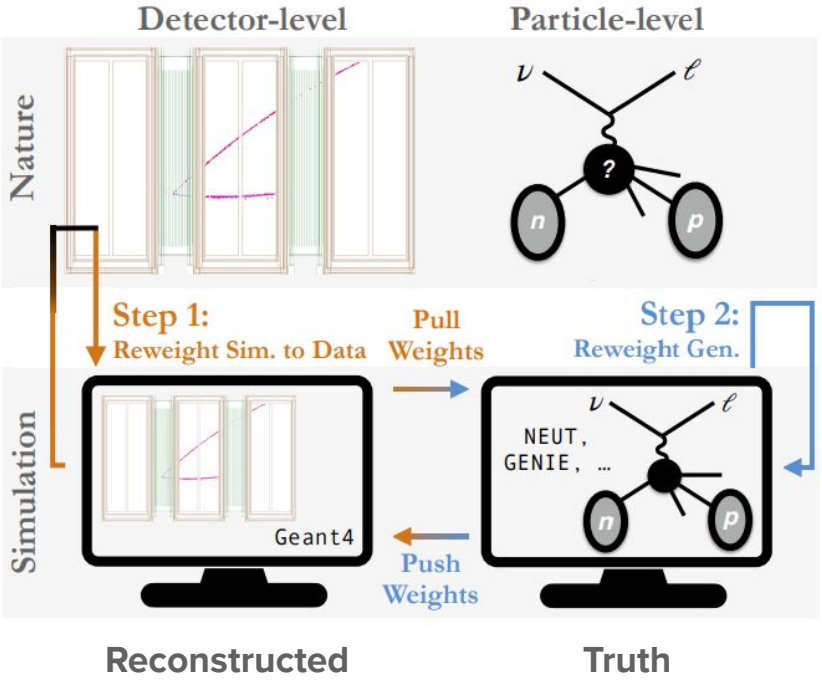
OmniFold Procedure

OmniFold is an **iterative unfolding procedure** performed in two steps:

1. Reweight reconstructed MC distribution to (better) match data, yielding **pull weights $\omega_n(m)$** for each reconstructed **m**
2. Reweight nominal truth MC distribution to incorporate information from step 1, yielding push weights $\nu_n(t)$ for each true value **t**

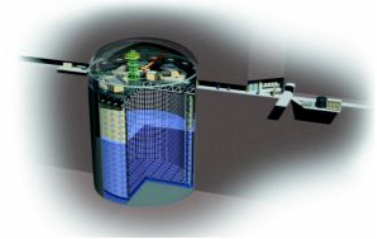
This is one iteration, and the method repeats until some convergence criteria is satisfied

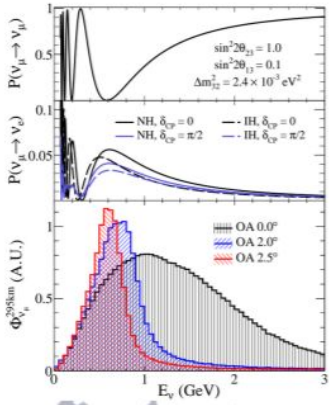
1. $\omega_n(m) = \nu_{n-1}^{\text{push}}(m)\, L[(1, \text{Data}), (\nu_{n-1}^{\text{push}}, \text{Sim.})](m),$
2. $\nu_n(t) = \nu_{n-1}(t)\, L[(\omega_n^{\text{pull}}, \text{Gen.}), (\nu_{n-1}, \text{Gen.})](t).$

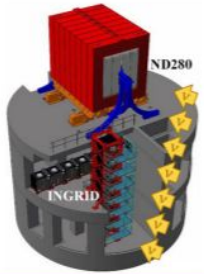


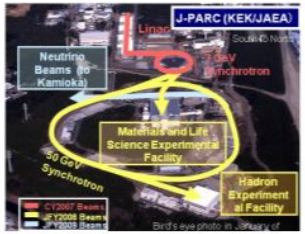
The T2K Experiment

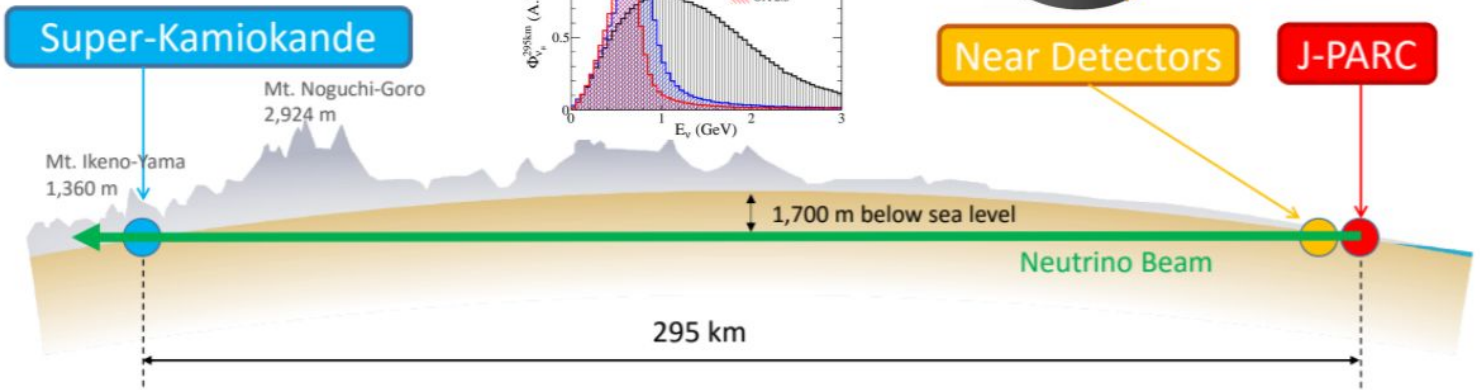
T2K is a long-baseline neutrino oscillation experiment in Japan that has been accumulating data since 2010



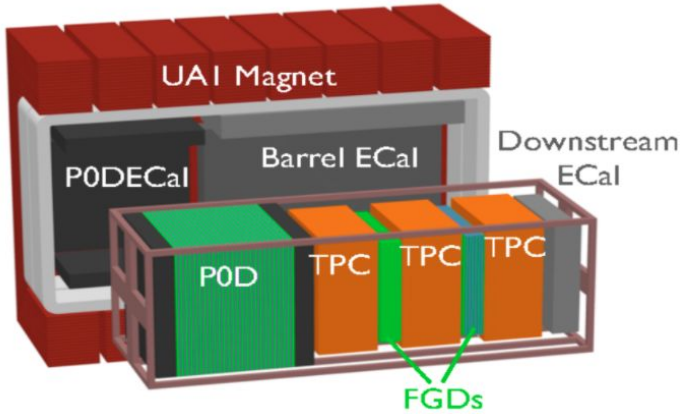








Test Setup - ND280 Public Dataset

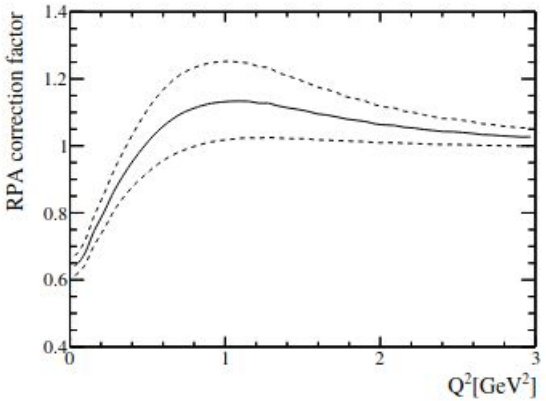


Public dataset of **~1.2 million simulated ND280 events** intended for ν_μ CC0π 2D differential cross-section

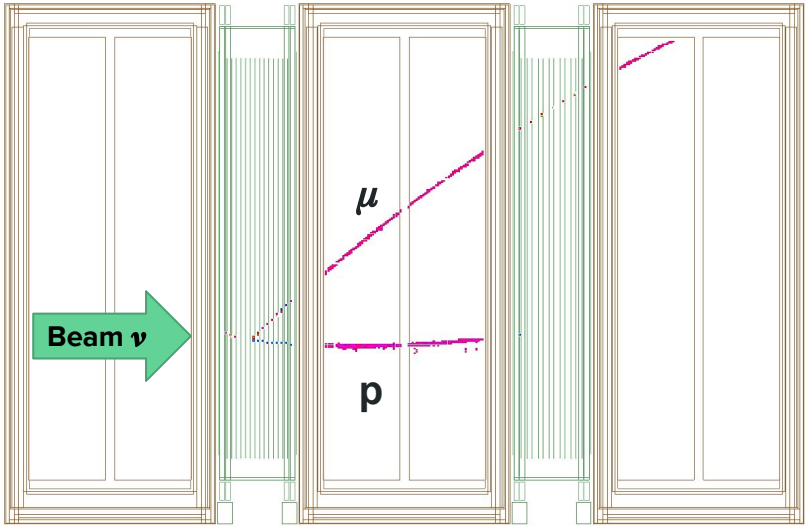
- Corresponds to about **20k measured events**

Dataset includes **muon and leading proton info**per event

Create **fake data** with a reweighting as a function of Q^2

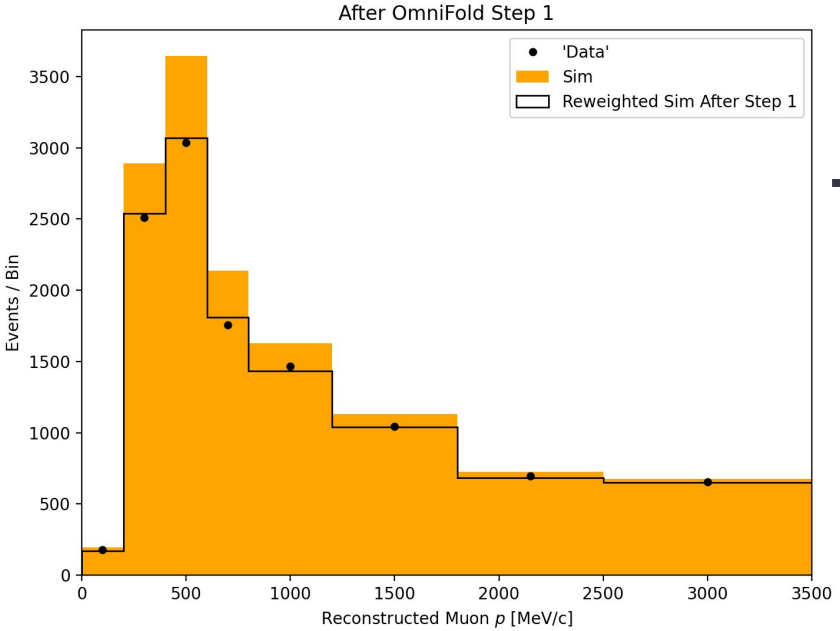


- Extract CC0π differential xsecs like with real data and compare against the data-truth values after unfolding to evaluate

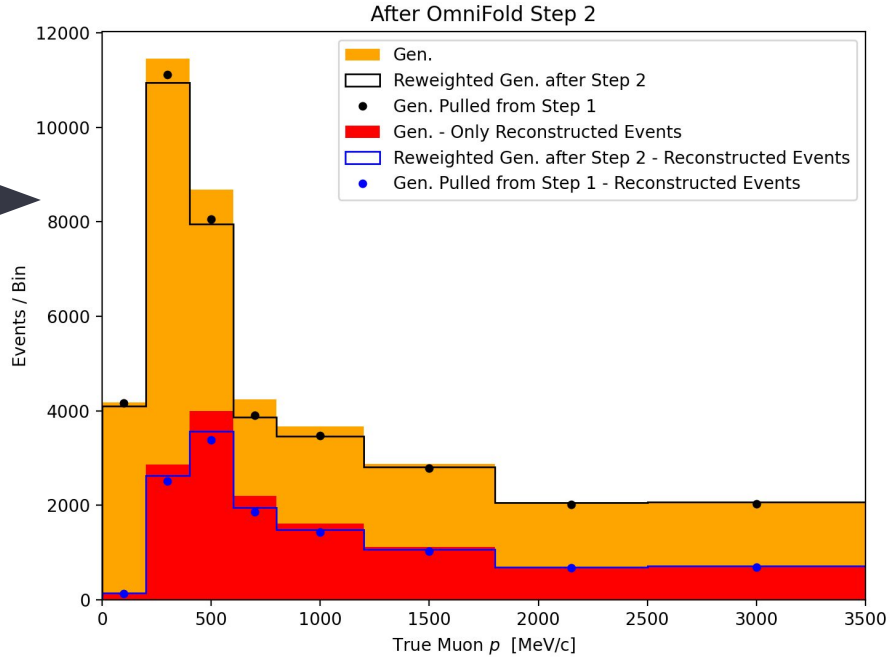


OmniFold Procedure Example

Step 1: reweight simulated reconstructed MC to observed data



Step 2: reweight generator-level MC to itself with pulled reweighting factors from step 1



Reminder: the reweighting procedure is unbinned!

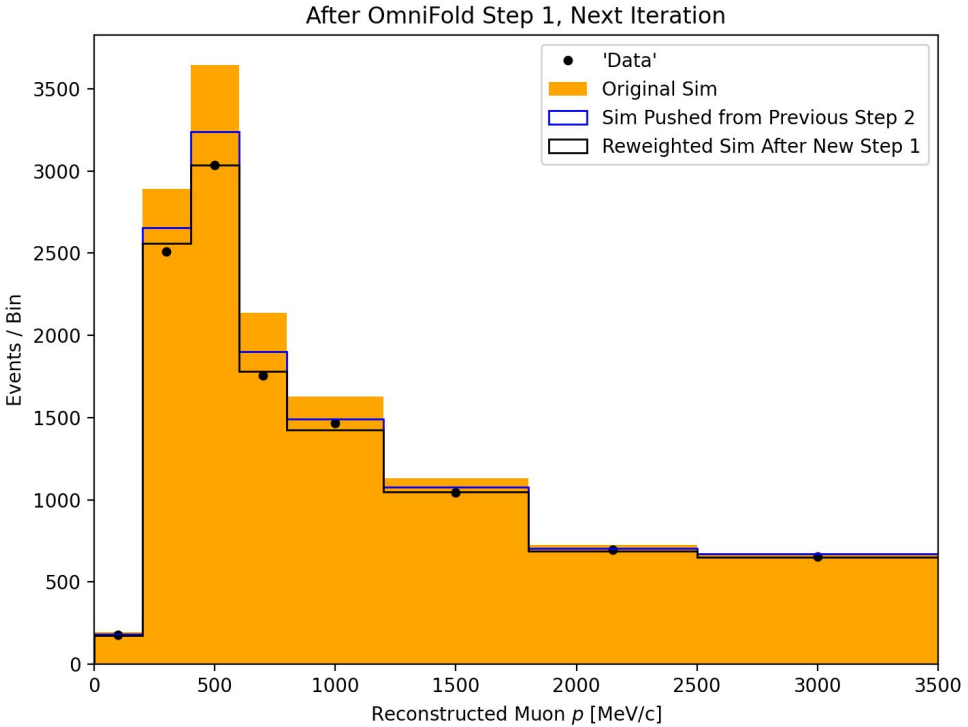
OmniFold Procedure Example

One iteration is a step 1 + step 2 combo

On a new iteration, go to step 1 again, but **starting with pushed reweighting factors** on the simulated reconstructed events from the previous iteration's step 2 result

Repeat for any number of iterations

- **Regularization** comes from a cutoff on the number of iterations, based on some chosen convergence criterion



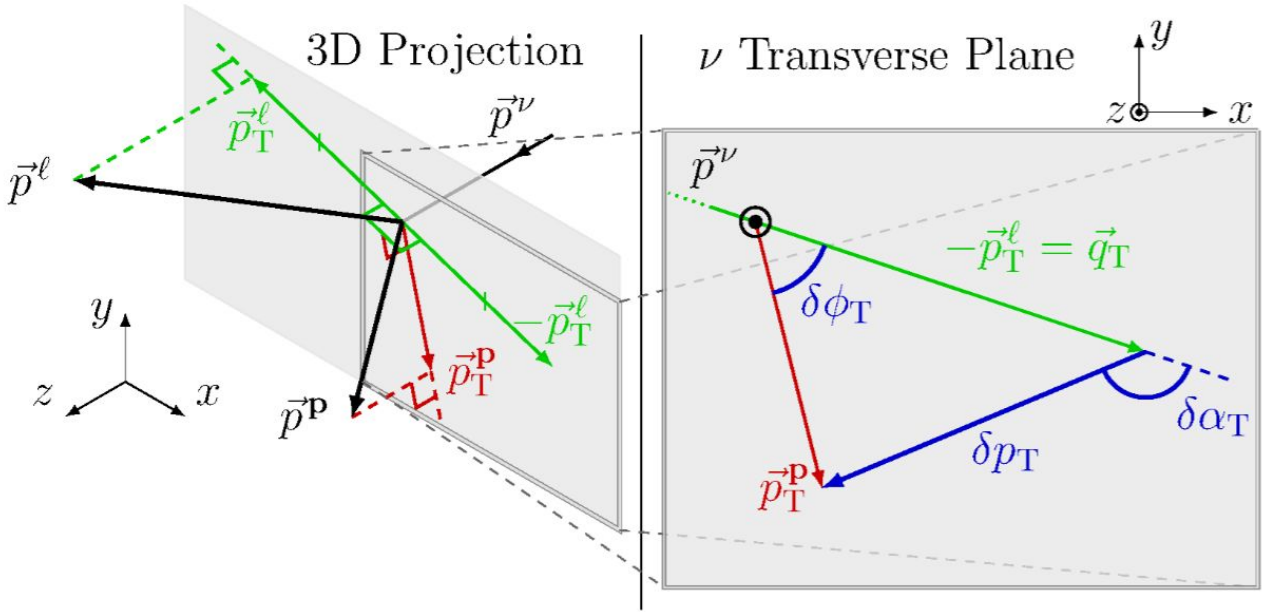
Variables of Interest

Evaluate performance of unfolding methods with 4 variables of interest:

- (p_μ, $\cos\theta_\mu$): muon momentum and forward angle
- $\delta\vec{p}_{\mathrm{T}} \equiv \vec{p}^{\,\mathrm{T}}_{\ell'} + \vec{p}^{\,\mathrm{T}}_{\mathrm{h}'}$

$$\delta\phi_{\mathrm{T}} \equiv \arccos\left(\frac{-\vec{p}^{\,\mathrm{T}}_{\ell'}\cdot\delta\vec{p}_{\mathrm{T}}}{p^{\mathrm{T}}_{\ell'}\delta p_{\mathrm{T}}}\right)$$

$$\delta\alpha_{\mathrm{T}} \equiv \arccos\left(\frac{-\vec{p}^{\,\mathrm{T}}_{\ell'}\cdot\vec{p}^{\,\mathrm{T}}_{\mathrm{h}'}}{p^{\mathrm{T}}_{\ell'}p^{\mathrm{T}}_{\mathrm{h}'}}\right)$$



These last 3 are the single transverse variables (STVs)

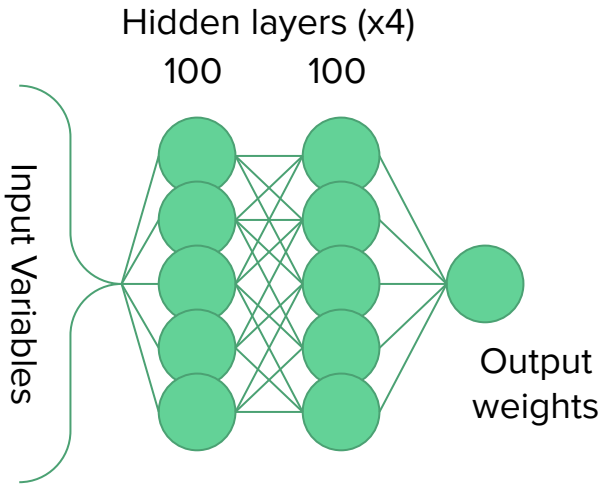
OmniFold Inputs & Network

OmniFold is agnostic to the choice of classifier

We use a simple MLP with 4 hidden layers of 100 nodes each

Input variables:

- Various **kinematic observables** (details on following slides). Generally standardized to mean 0 and unit variance, with values of 0 when the variable does not exist
- For detector space only: **Detector sample ID** (1-hot encoded, out of 8 possible sample IDs)
- For truth space only: **Interaction topology** (1-hot encoded, out of 5 possibilities: CC0π0p, CC0π1p, CC0πNp, CC1π, CCOther)

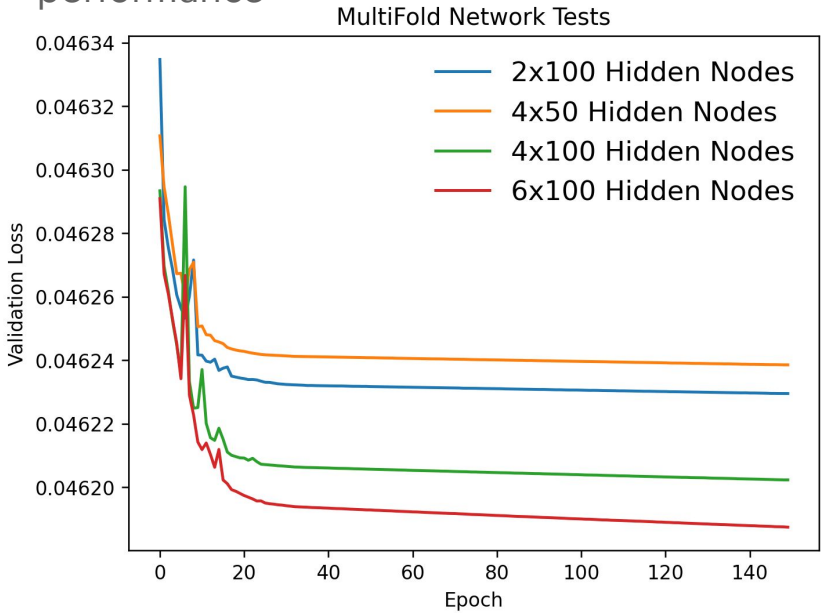


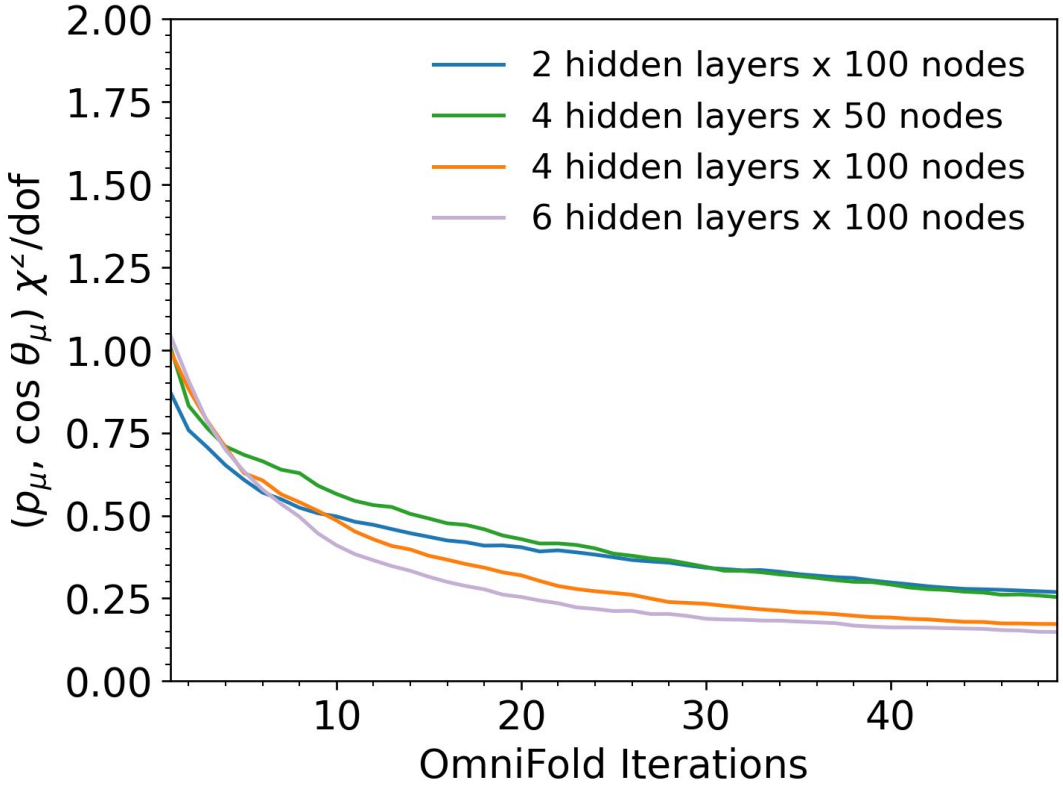
Using one NVIDIA A100 on a NERSC Perlmutter node, << 1 minute per OmniFold iteration on one set of data/MC

Neural Network Sizes

OmniFold analyses usually don't need very complicated networks

But we can observe small networks limiting performance





Testing OmniFold

Every setup includes detector sample ID for reco-space and topology ID for truth-space (one-hot encoded in both cases)

Conventional-like unfolding setups:

- **IBU-UniFold**: using the OmniFold machinery, but the inputs to the neural network are limited to the bin indices of whatever variable we're unfolding (e.g. an event is only identified by saying it's in bin #3 of the dpT binning).
  - **This is mathematically equivalent to IBU**
- **Binned UniFold**: also binned inputs, but now identifying events by the value of the center of the bin they fall into (e.g. an event gets an input of 100 MeV if it's in the bin centered on 100 MeV, instead of getting an input saying it's in bin #2)
  - This is the same amount of info as IBU-UniFold, but in a format that's harder for a neural network to learn

Testing OmniFold

OmniFold-type unfolding setups (unbinned inputs):

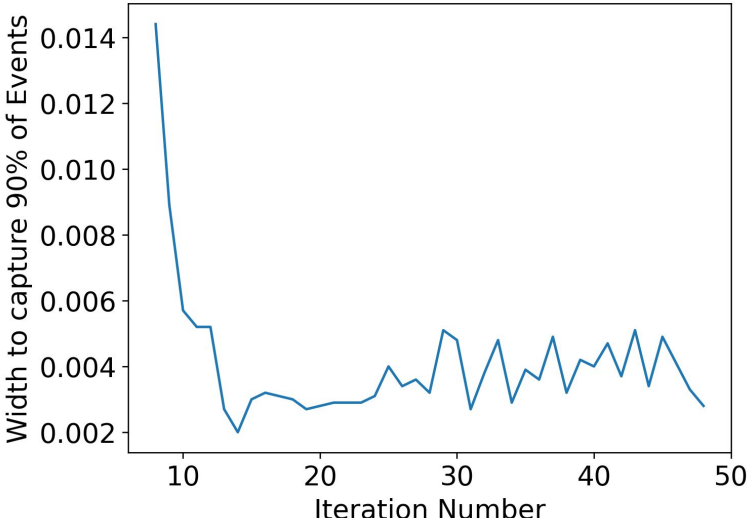
- **UniFold**: only receive the unfolding variable in question as kinematic input. Run a separate version for each variable we're unfolding
- **MultiFold**: use (p_μ, cos θ_μ, p_p, δp_T, $\delta\alpha_T$, $\delta\phi_T$) as input. This includes every observable of interest, and should be the "easiest" way to unfold them all at once
- **OmniFold**: use muon and leading proton kinematics (p_μ, cos θ_μ, ϕ_μ, p_p, cos θ_p, ϕ_p) as input. This is the most general input we can supply given the available data, and in principle everything is derivable from these values

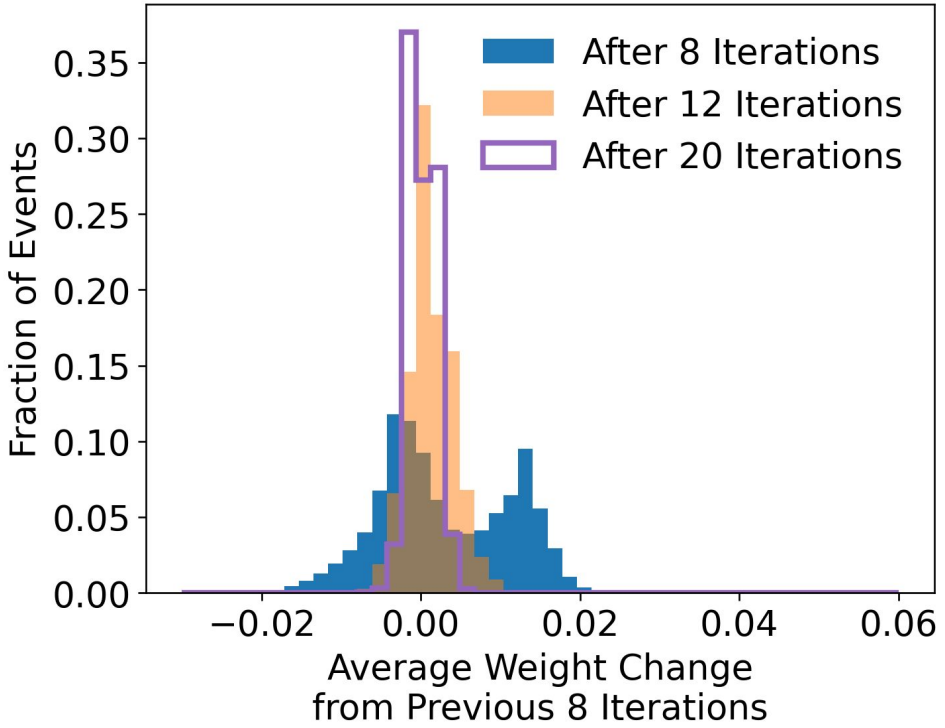
Each method is run **500 times**, once for each of the syst/stat throws

Unbinned Convergence Metrics

A convergence metric for OmniFold is ideally independent of binning and observable choices

Plot average **weight change of each event** over the last N iterations, which should cluster around 0 as it converges





Results

- Binned UniFold is worse than IBU-UniFold => **caused by NN training effects**
- UniFold performs similarly to Binned UniFold => not much gain from going unbinned without additional info (bins are already quite fine)
- For χ^2, MultiFold is similar to IBU and better than OmniFold
- For bias (triangular discriminator), MultiFold/OmniFold are both better than IBU

	χ^2			
Method	$(p_\mu, \cos\theta_\mu)$ DoF=58	δp_T DoF=8	$\delta \alpha_T$ DoF=8	$\delta \phi_T$ DoF=8
Prior	298.2	2.3	5.9	4.9
IBU-UniFold	2.1	0.2	0.4	0.1
Binned UniFold	21.4	1.4	0.9	0.5
UniFold	27.1	1.1	0.6	1.1
MultiFold	3.1	0.3	0.2	0.3
OmniFold	10.0	0.8	1.1	0.4

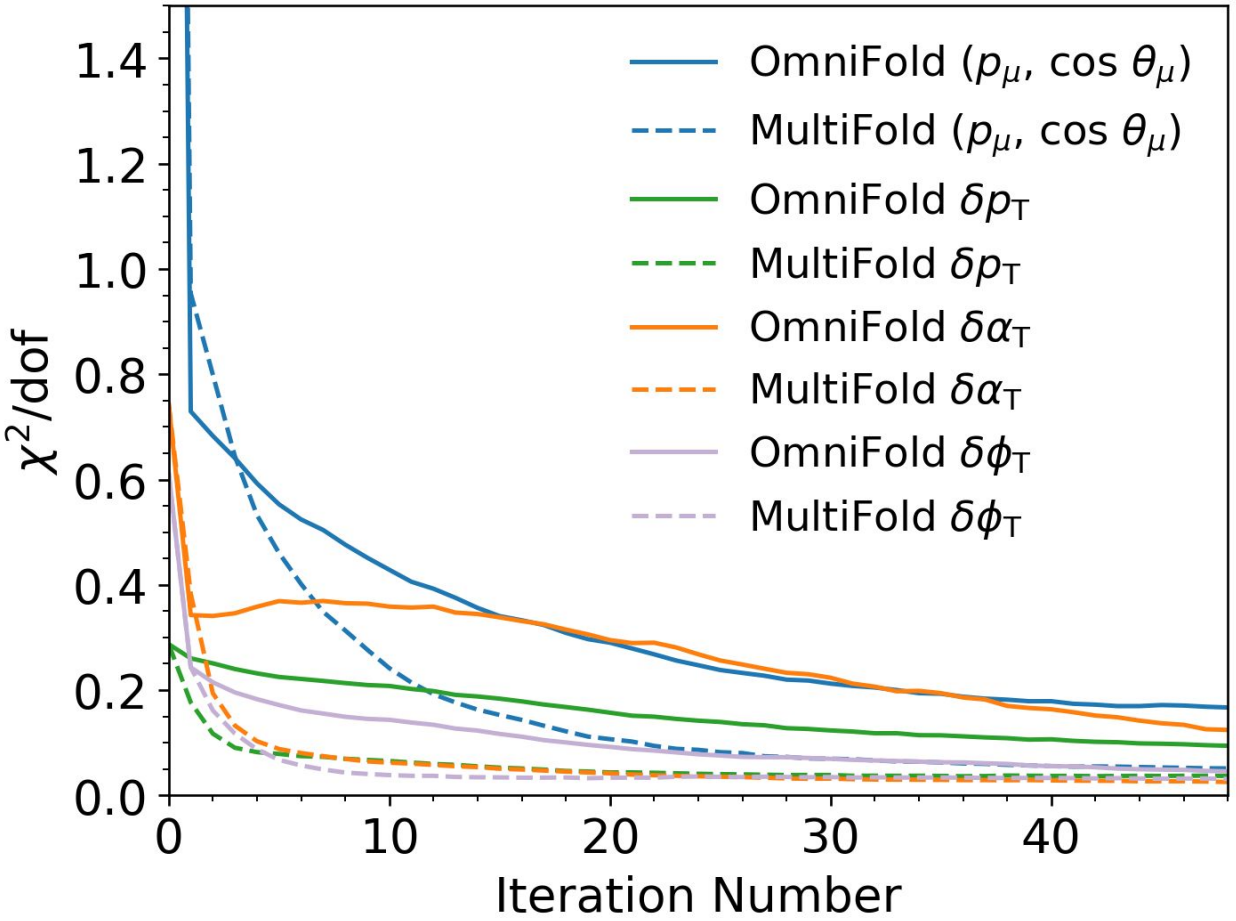
	Triangular Discriminator			
Method	$(p_\mu, \cos\theta_\mu)$	δp_T	$\delta \alpha_T$	$\delta \phi_T$
Prior	545.6	27.5	31.2	26.7
IBU-UniFold	17.1	1.9	3.4	0.8
Binned UniFold	29.9	2.8	6.0	1.9
UniFold	17.3	5.7	5.8	1.7
MultiFold	2.7	0.7	0.6	0.6
OmniFold	9.4	1.7	3.0	2.1

Input Comparison

MultiFold (dashed lines, using unfolding variables as direct input) **outperforms OmniFold** (solid lines, general muon/proton kinematics input)

Again indicates imperfect neural network training

- Statistics are low (20k data events) relative to many applications

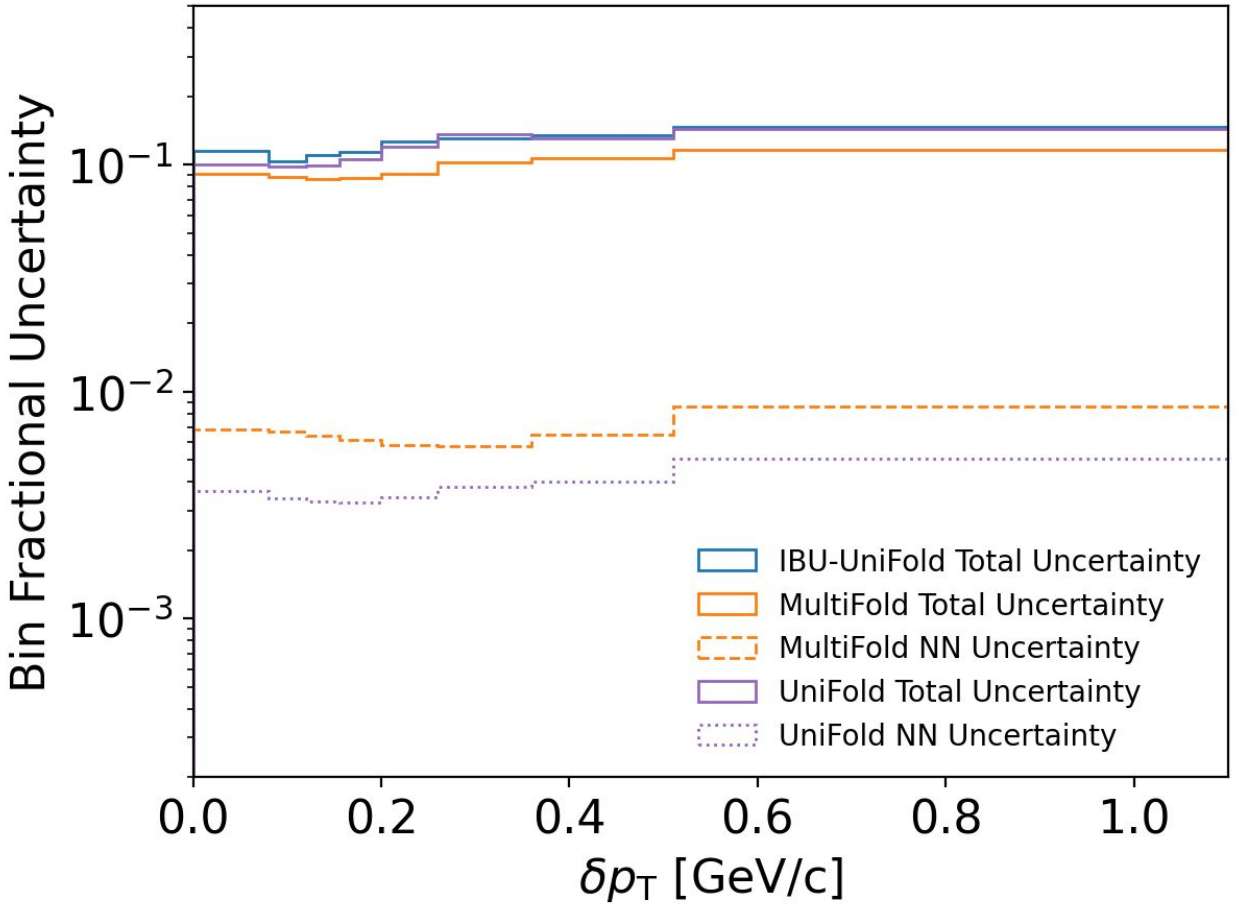


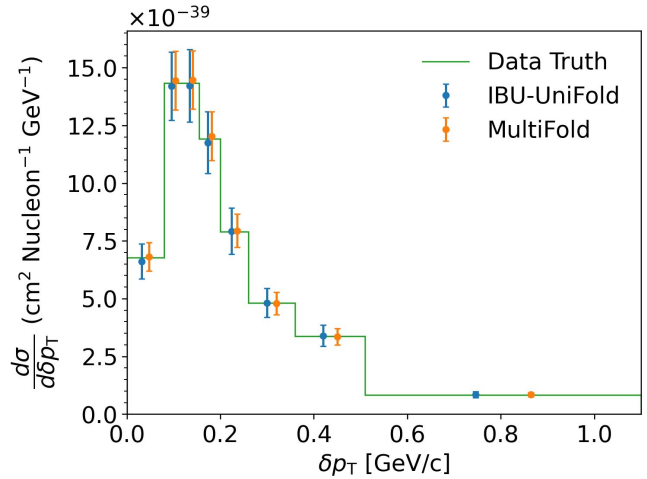
Uncertainties

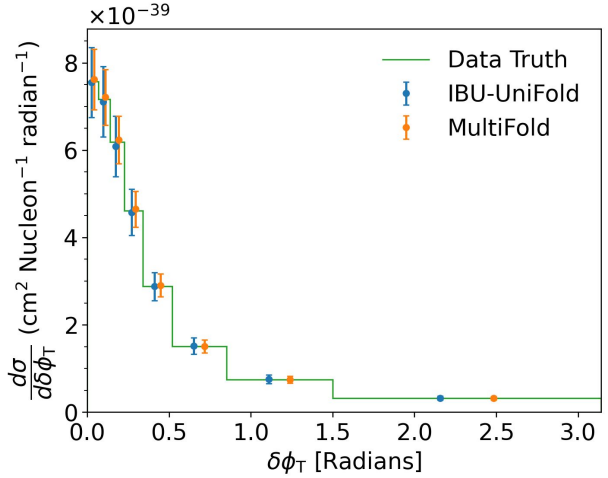
MultiFold achieves smaller bias and lower uncertainties than IBU

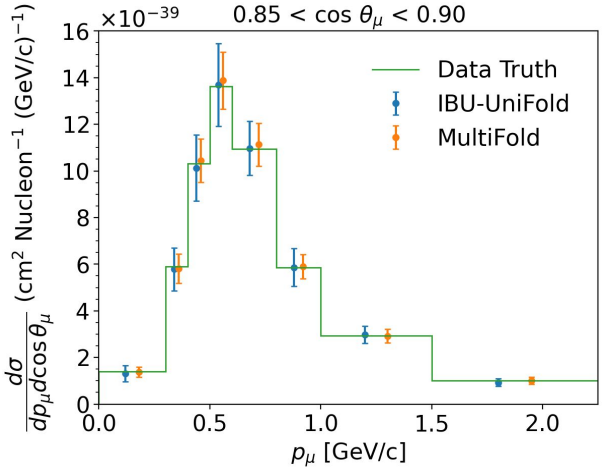
- Neural network training is a form of regularization

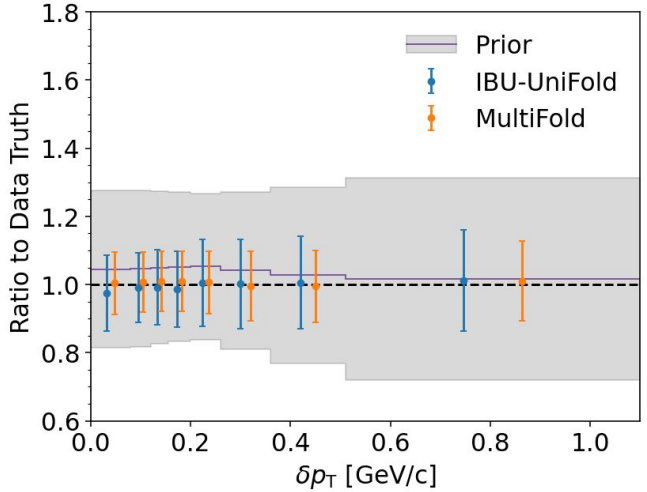
For the STVs, UniFold has smaller NN uncertainty than MultiFold, due to the relative simplicity of the network

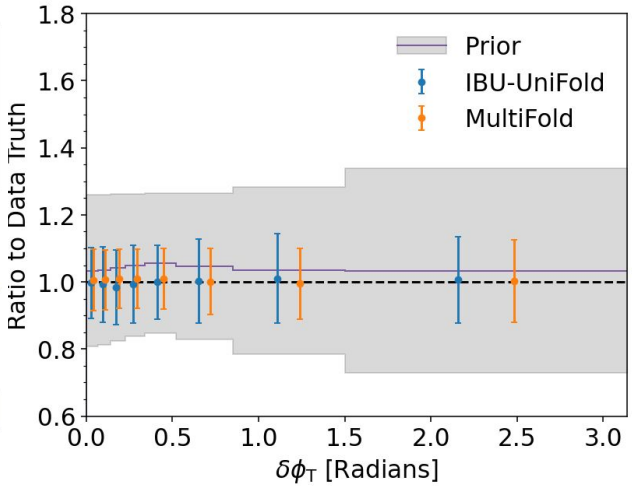


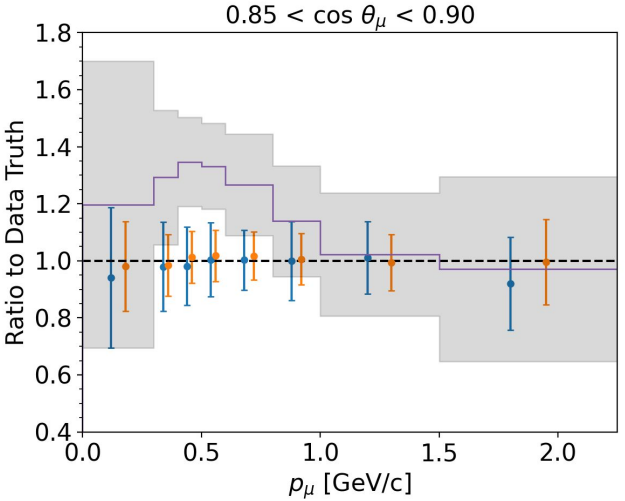

×10⁻³⁹
Data Truth
IBU-UniFold
MultiFold
$\frac{d\sigma}{d\delta p_T}$ (cm² Nucleon⁻¹ GeV⁻¹)
δp_T [GeV/c]


×10⁻³⁹
Data Truth
IBU-UniFold
MultiFold
$\frac{d\sigma}{d\delta\phi_T}$ (cm² Nucleon⁻¹ radian⁻¹)
$\delta\phi_T$ [Radians]


×10⁻³⁹
0.85 < cos θ_μ < 0.90
Data Truth
IBU-UniFold
MultiFold
$\frac{d\sigma}{dp_\mu d\cos\theta_\mu}$ (cm² Nucleon⁻¹ (GeV/c)⁻¹)
p_μ [GeV/c]


Prior
IBU-UniFold
MultiFold
Ratio to Data Truth
δp_T [GeV/c]


Prior
IBU-UniFold
MultiFold
Ratio to Data Truth
$\delta\phi_T$ [Radians]


0.85 < cos θ_μ < 0.90
Ratio to Data Truth
p_μ [GeV/c]

OmniFold Results Summary

More details on this T2K study with OmniFold: Phys. Rev. D 112, 012008

General takeaways:

- **OmniFold obtains similar χ^2 and less bias** than IBU, but we get to unfold everything at once and in an unbinned way
- IBU vs Binned UniFold and MultiFold vs OmniFold comparisons show importance of choice of inputs in combination with model architecture/training and available statistics
- The relative **lack of detail in the public dataset** limits OmniFold performance, but this can be fixed with real data
- **Low statistics** is limiting the performance too, but this will also be a problem for real data

Potential OmniFold Applications

OmniFold is useful for any analysis where there is **a lot of information that is relevant** to the unfolding problem, or there are **many observables of interest**.

Many neutrino analyses could benefit from the advantages of OmniFold:

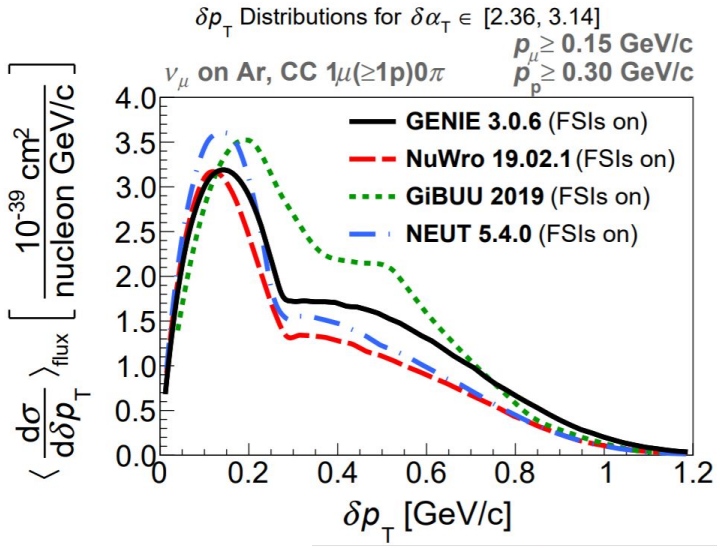
- Simply improving the smearing/efficiency corrections by being unbinned and high-dimensional
- Simultaneous unfolding of several samples, and checking correlations among the results
- Samples where many observables are useful proxies for underlying physics, and we'd like to project our results into many directions at once

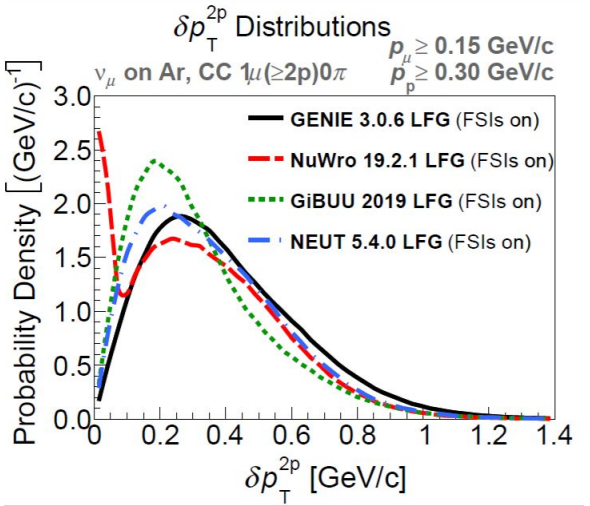
More thoughts on practical considerations for using OmniFold: arXiv:2507.09582

Neutrino Event Generators

There are many neutrino event generators in circulation and being actively used by various analyses right now

Due to our still relatively poor understanding of neutrino interactions in general right now, the generators often disagree quite noticeably with each other





Plots: arXiv:2201.04664

Neutrino Event Generators

Often useful to check how the choice of generator impacts the result of an analysis

- Don’t want to be overly sensitive to modeling uncertainties

But it’s very expensive to propagate another set of generator events through detector simulation/reconstruction again

- Realistically, you won’t have a full statistics MC dataset for every generator you want to test against

If we could quickly reweight the results of one generator to replicate another’s, we could more rapidly iterate on these kinds of studies

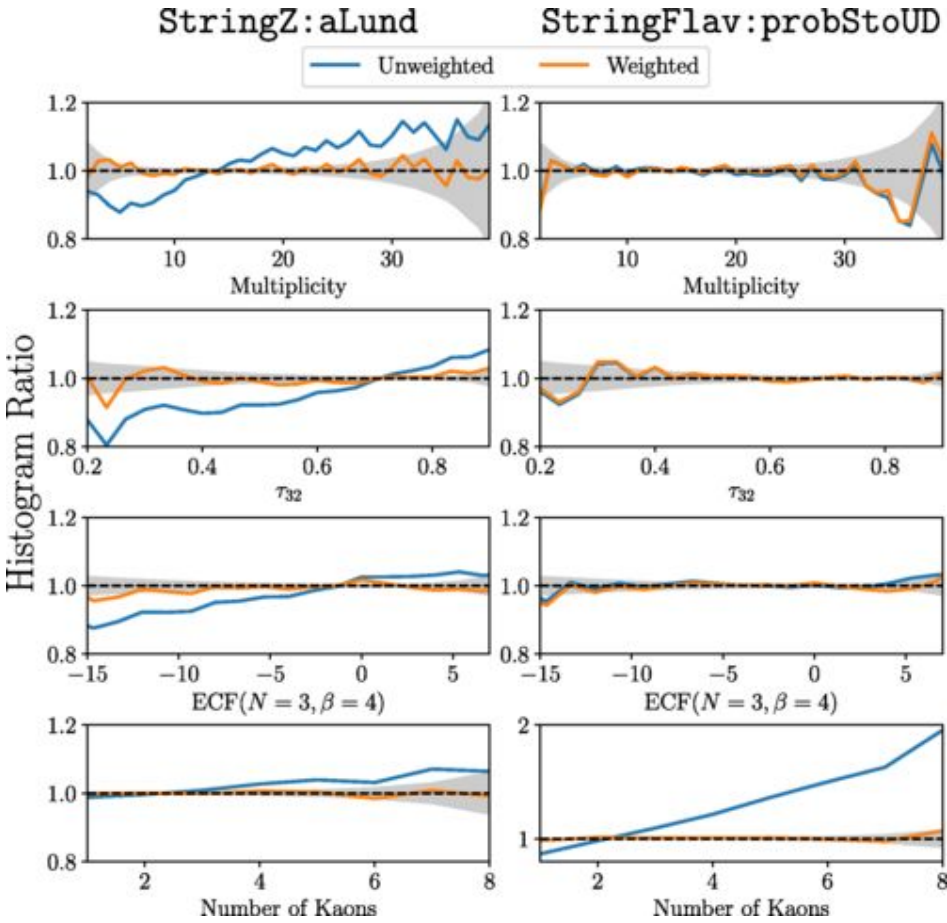
- Conventionally this relies on splines or histograms, only valid in a limited phase space…

Generator Reweighting

Reweighting one generator's output to match another's is **finding the likelihood ratio** between their distributions

Likelihood ratio trick offers a path to converting between generator outputs with **validity over much of the phase space**

- Range of validity will depend on which generator outputs are included as part of the reweighting function



Plot: Phys. Rev. D 101, 091901

Neural Network Architecture

Test setup: train classifier between **GENIE 20i** events and **NUWRO** events

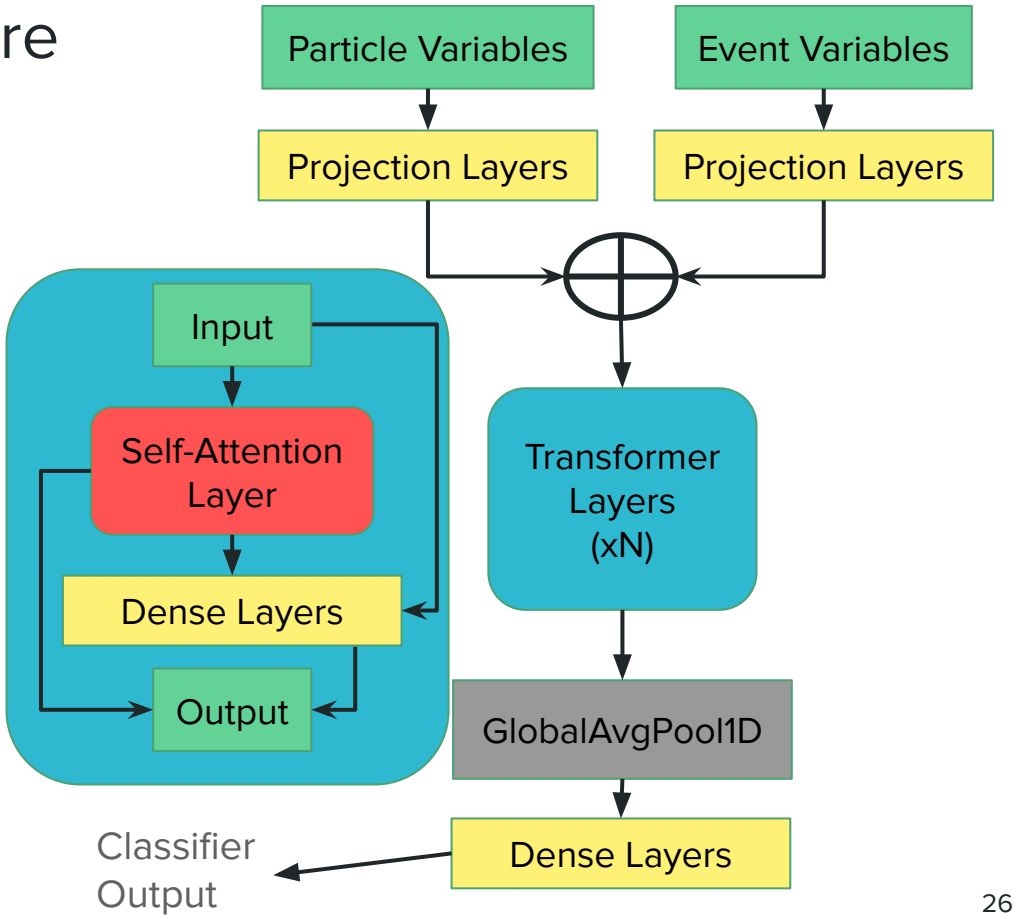
Want a reweighting that's a function of as much of the generator outputs as possible

- Need to deal with large and variable-length particle stacks

Use 4-momenta + PID for particles

Event variables include quantities like q_0, q_3, Q^2, W, E_ν

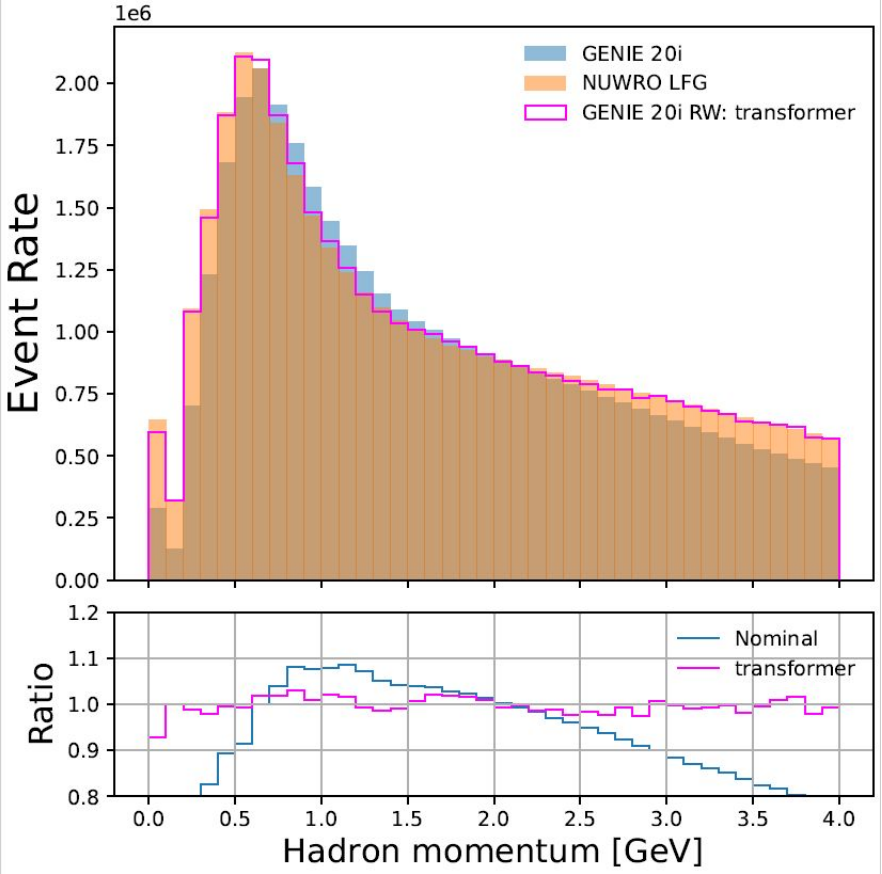
From Andrew Cudd

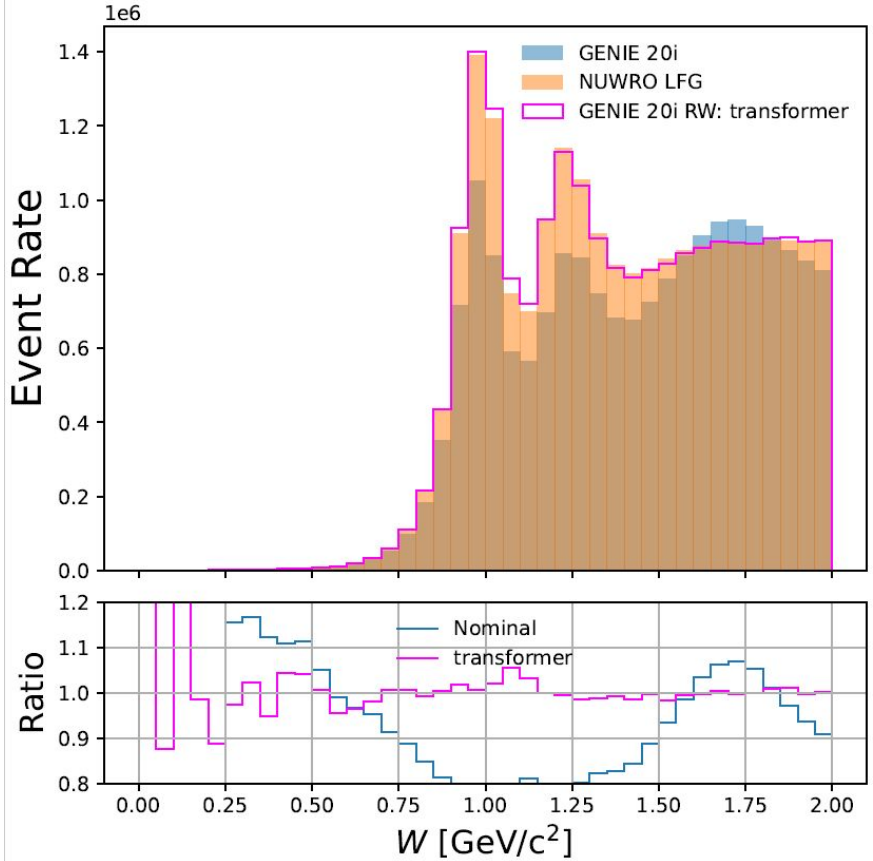


Reweighting Results

From Andrew Cudd

Particle-level and event-level kinematics





Summary

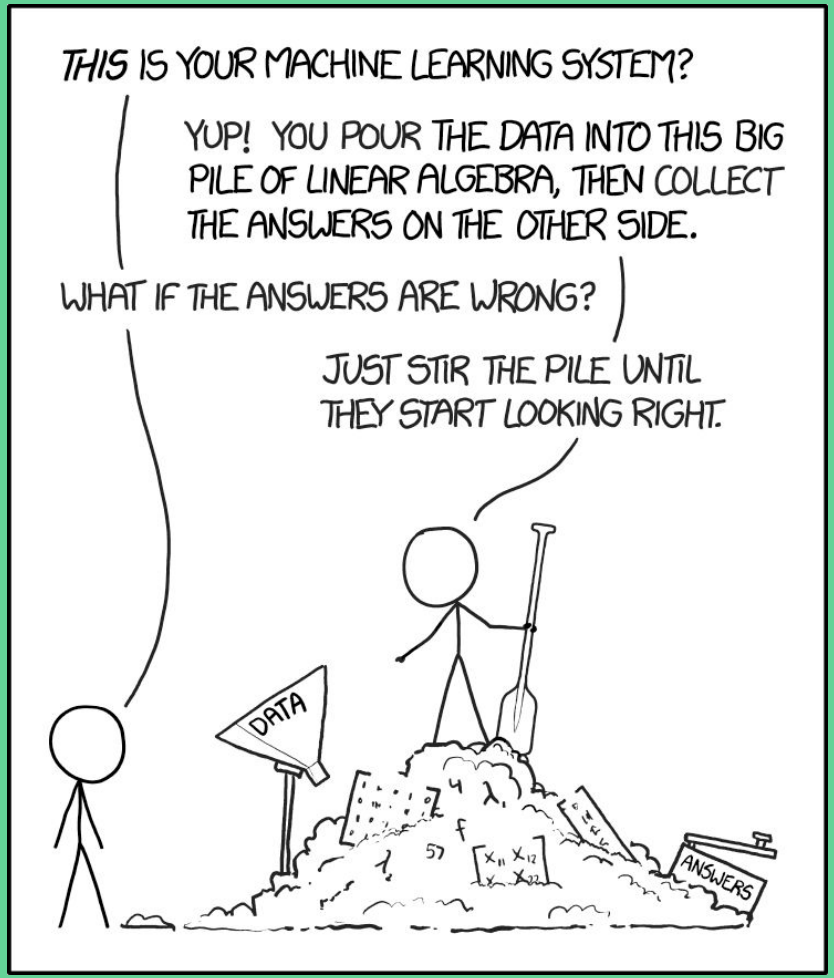
After building a good detector and good reconstruction algorithms, we should construct analyses that can use all the available information

- ML-based unfolding and generator reweighting both help with this

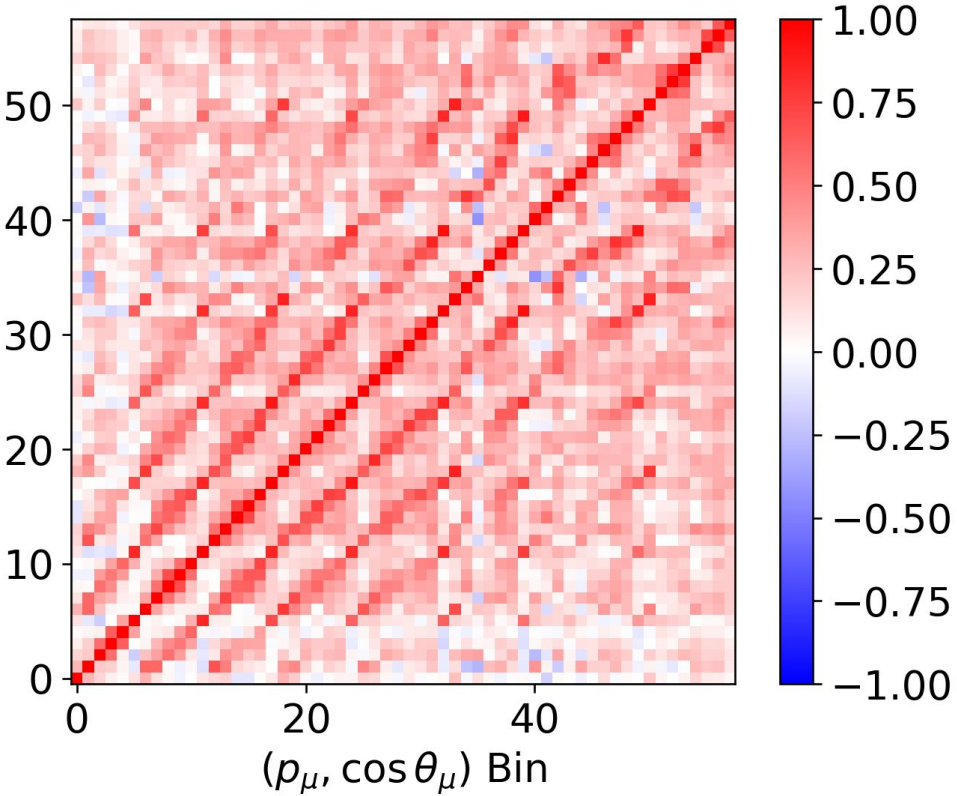
But as we incorporate more variables into the analysis, remember to watch out for:

- (Sub x N)-leading values that are not well-vetted for validity
- What quantities we actually have systematic uncertainties for
- Poorly constrained regions of phase space that shouldn’t be extrapolated into

Backup


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

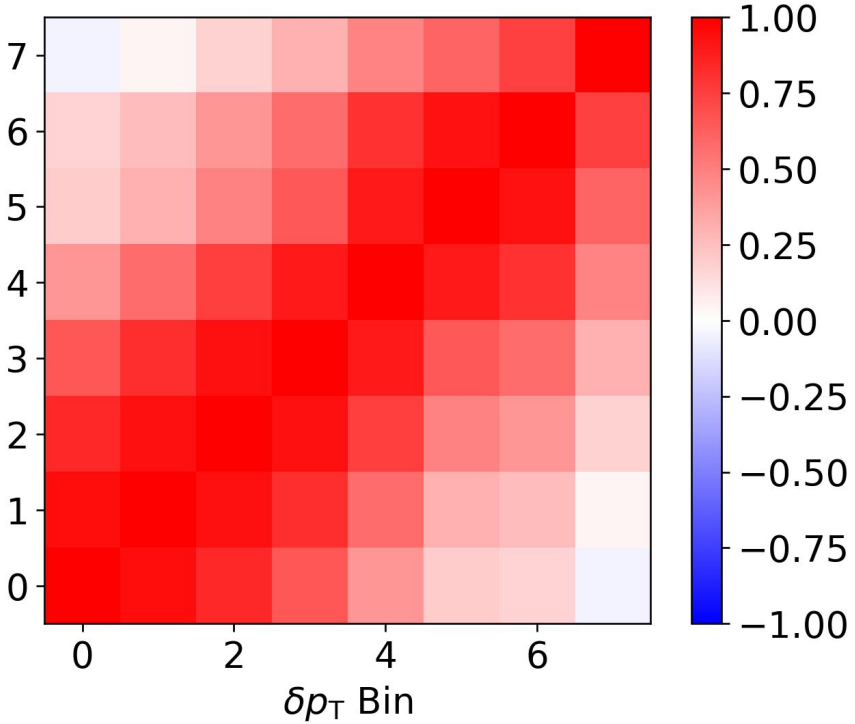
MultiFold Correlation Matrices

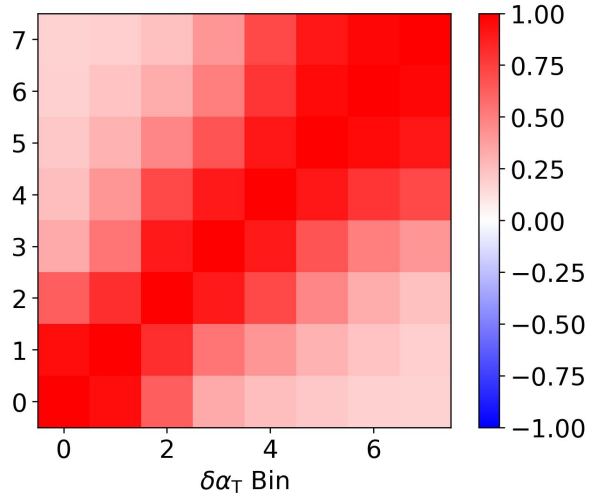


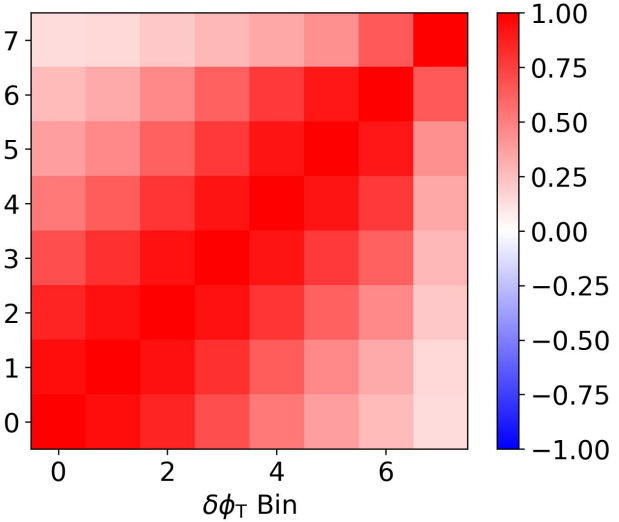
Correlation matrix excluding flux uncertainty

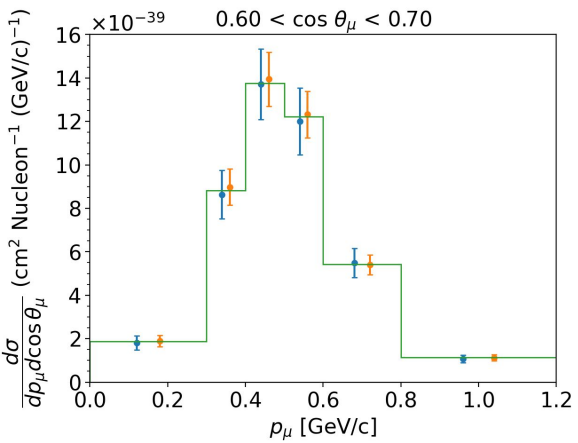
- Actual χ^2 is evaluated including effects of flux uncertainty in the covariance matrix

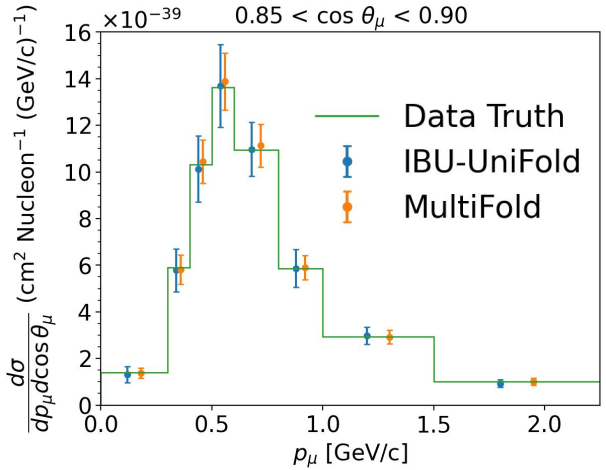
MultiFold Correlation Matrices

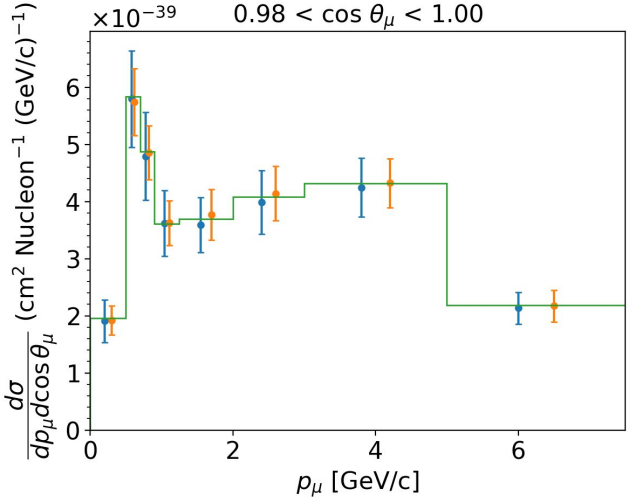


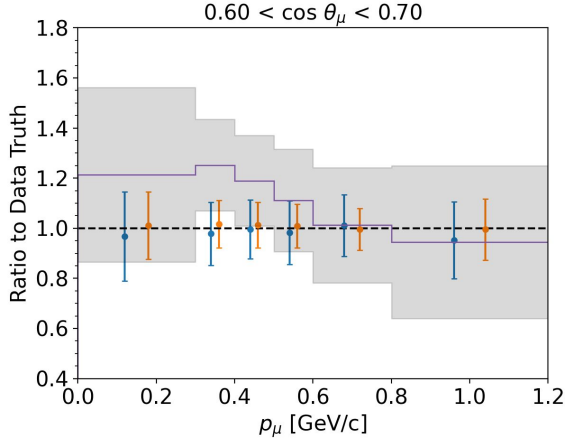


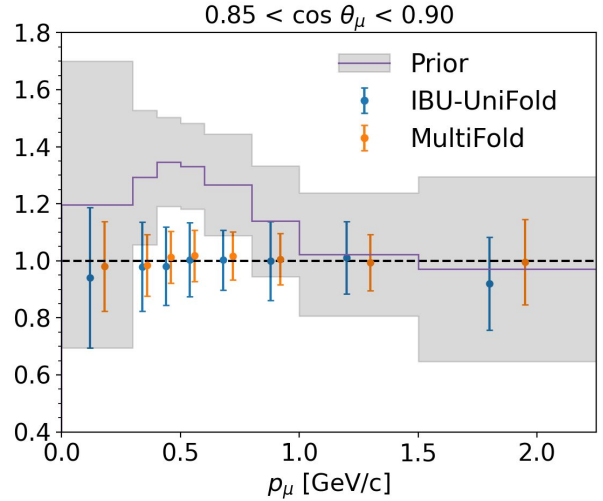


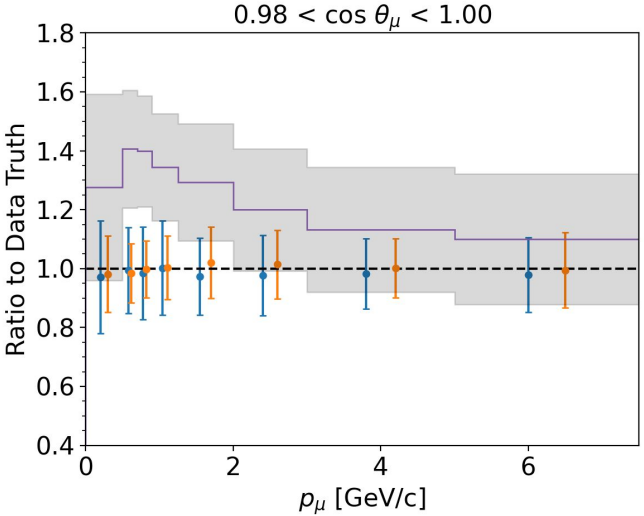

×10⁻³⁹
0.60 < cos θμ < 0.70
dσ/dpμdcosθμ (cm² Nucleon⁻¹ (GeV/c)⁻¹)
pμ [GeV/c]

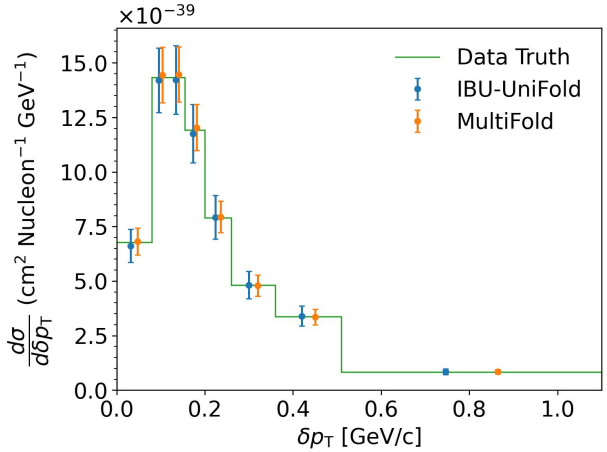

×10⁻³⁹
0.85 < cos θμ < 0.90
Data Truth
IBU-UniFold
MultiFold
dσ/dpμdcosθμ (cm² Nucleon⁻¹ (GeV/c)⁻¹)
pμ [GeV/c]

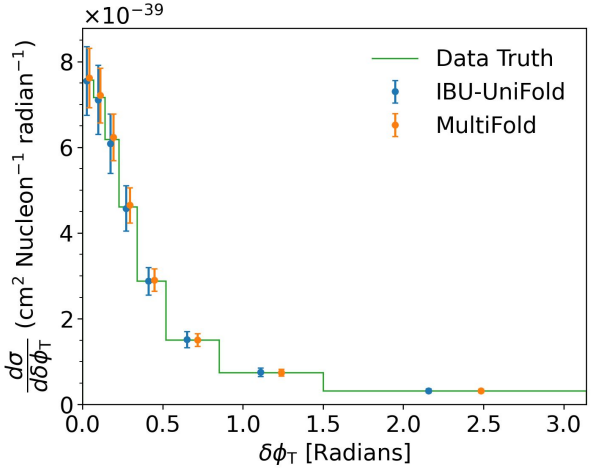

×10⁻³⁹
0.98 < cos θμ < 1.00
dσ/dpμdcosθμ (cm² Nucleon⁻¹ (GeV/c)⁻¹)
pμ [GeV/c]

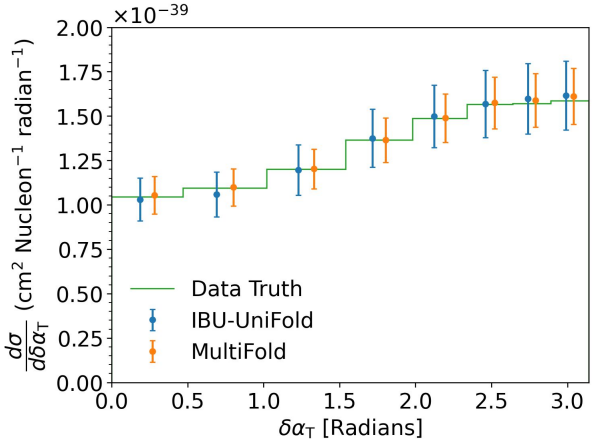

0.60 < cos θμ < 0.70
Ratio to Data Truth
pμ [GeV/c]

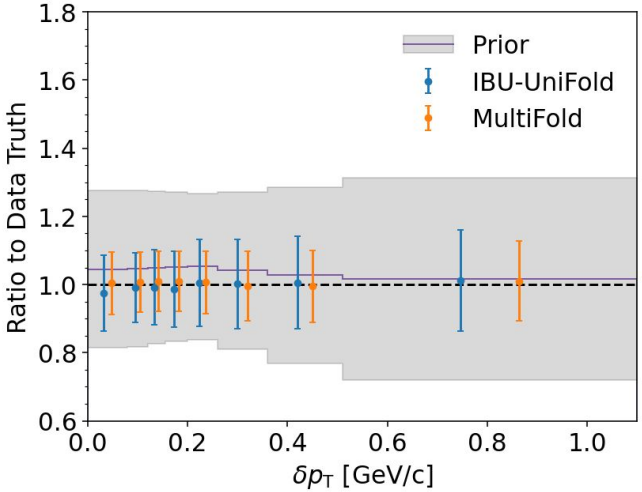

0.85 < cos θμ < 0.90
Prior
IBU-UniFold
MultiFold
pμ [GeV/c]

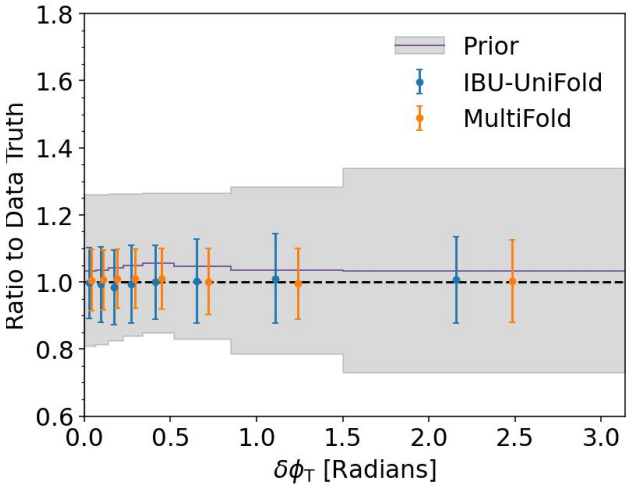

0.98 < cos θμ < 1.00
Ratio to Data Truth
pμ [GeV/c]

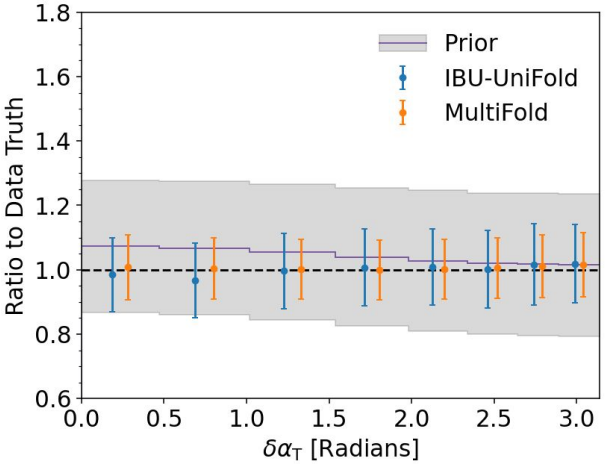

×10⁻³⁹
Data Truth
IBU-UniFold
MultiFold
$\frac{d\sigma}{d\delta p_T}$ (cm² Nucleon⁻¹ GeV⁻¹)
δp_T [GeV/c]


×10⁻³⁹
Data Truth
IBU-UniFold
MultiFold
$\frac{d\sigma}{d\delta\phi_T}$ (cm² Nucleon⁻¹ radian⁻¹)
$\delta\phi_T$ [Radians]


×10⁻³⁹
Data Truth
IBU-UniFold
MultiFold
$\frac{d\sigma}{d\delta\alpha_T}$ (cm² Nucleon⁻¹ radian⁻¹)
$\delta\alpha_T$ [Radians]

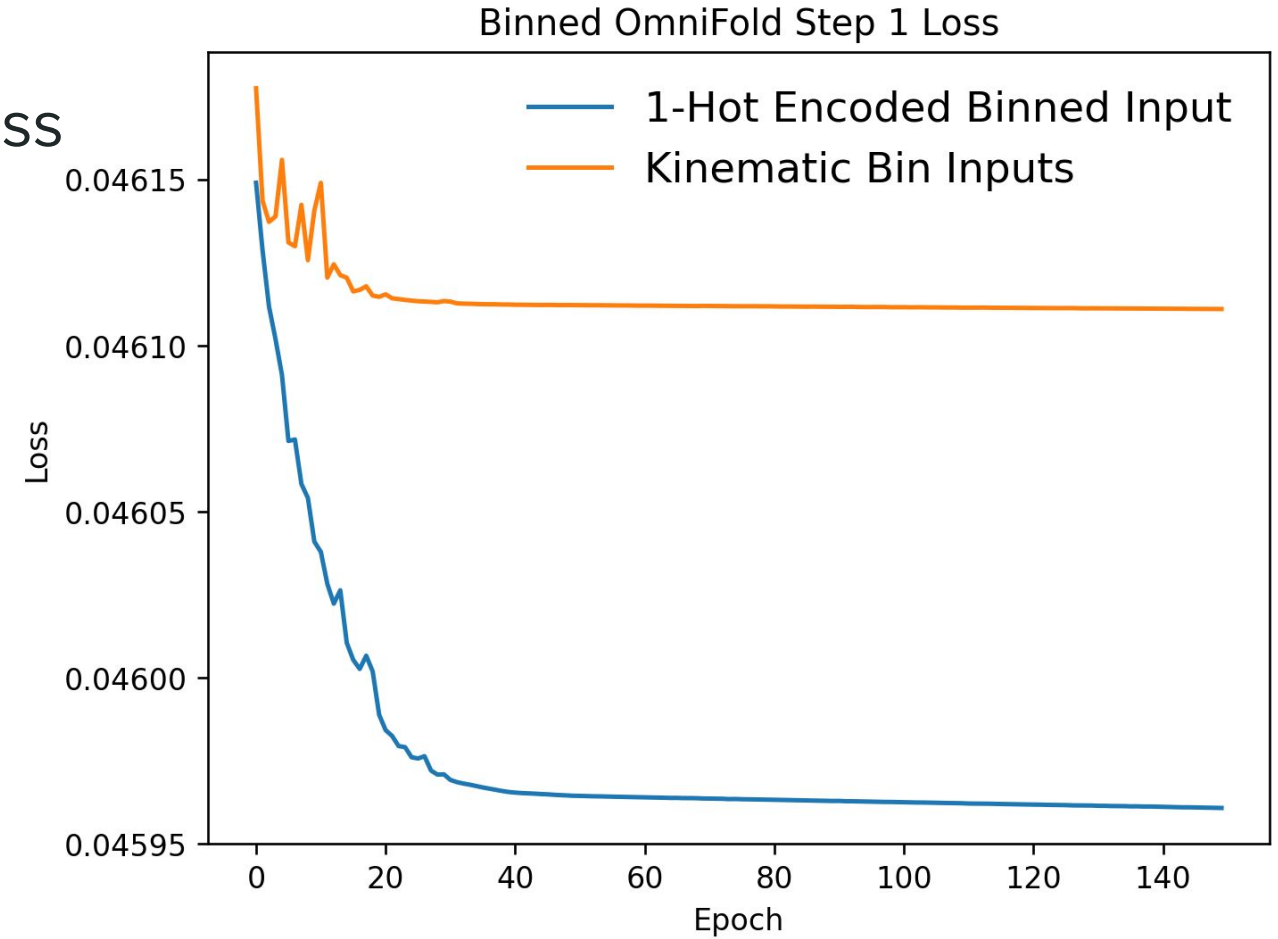

Prior
IBU-UniFold
MultiFold
Ratio to Data Truth
δp_T [GeV/c]


Prior
IBU-UniFold
MultiFold
Ratio to Data Truth
$\delta\phi_T$ [Radians]

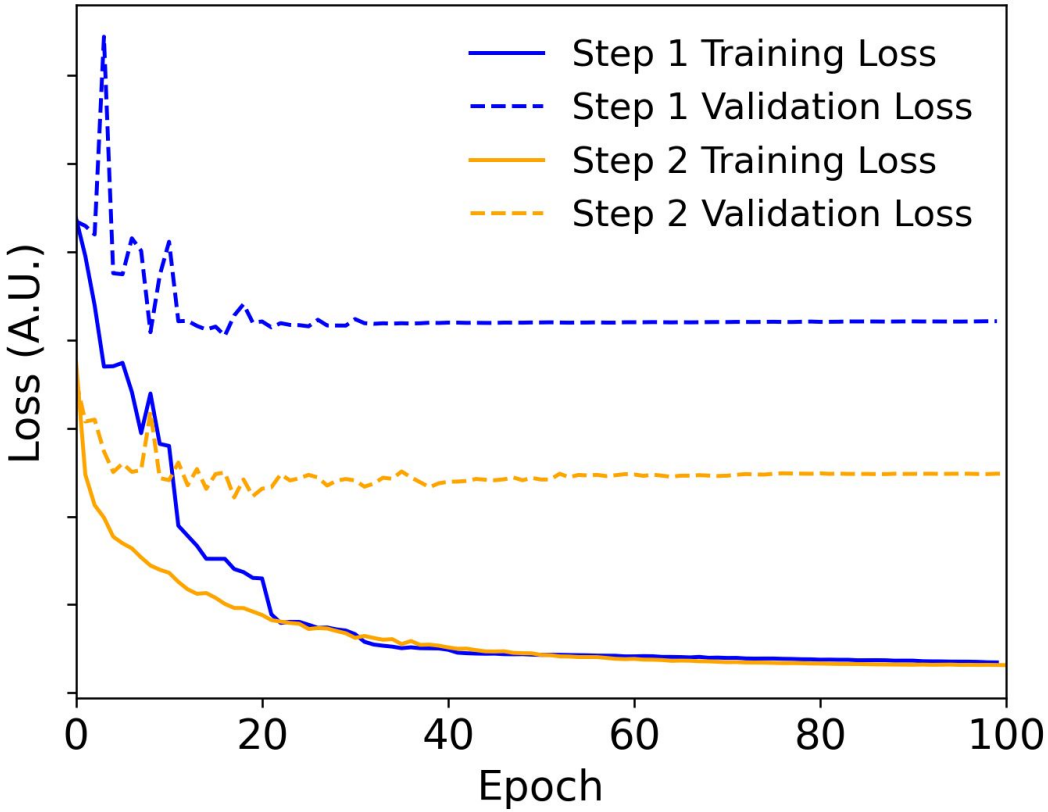

Prior
IBU-UniFold
MultiFold
Ratio to Data Truth
$\delta\alpha_T$ [Radians]

Binned Training Loss

Difference in performance for IBU-UniFold (1-hot encoded) vs Binned-UniFold (kinematic bins) visible in training loss curves



Training/Validation Loss



Example of training/validation loss curves for a step 1 + step 2 iteration of OmniFold

- Arbitrary offset on y-axis for each curve for plotting convenience

Actual result would be terminated after 15 epochs of no improvement in validation loss

Training Parameters

Our results are of course with the “best” NN training settings

But in our scans we have found many settings that yield worse performance

- Must be careful about tuning parameters before applying to real data!