Machine Learning-Assisted Unfolding for Neutrino Cross-section Measurements

NuSTEC NuXtract Workshop @ CERN
Andrew Cudd, Masaki Kawaue, Tatsuya Kikawa, Roger Huang,
Ben Nachman, Callum Wilkinson
2023/10/03


University of Colorado **Boulder**

Machine learning (ML) assistance

Neural networks learn to approximate the likelihood ratio when trained to distinguish between the two datasets

(or something monotonically related to it in some known way)

This transforms the problem from **density estimation** (which is hard) to **classification** (which is ~~easy~~ less hard)

Neural nets are naturally unbinned and are well suited to high-dimensional datasets

Note: that other classifiers could be used for this, such as a boosted decision tree

Explained in detail in this paper: A. Andreassen, B. Nachman, PRD RC 101 (2020) 091901, arxiv:1907.08209

Omnifold concept: ML reweighting

Train a fully connected neural network to classify between two datasets A & B:

Using the weighted cross-entropy loss where each event **x** has a weight **w** and a true label **p** gets a prediction **q**

$$L(p_i, q_i) = -w_i \left(p_i \log(q_i) + (1 - p_i) \log(1 - q_i)\right)$$

The predictions from the network **q** then approximate the ratio of the two datasets and can be used to reweight from one to the other:

$$\mathcal{L} = p_A(x_i)/p_B(x_i) \approx q_i/(1 - q_i)$$

In practice this uses real (or fake) data and the MC prediction as inputs to the network ➜ this talk uses a public T2K MC dataset as input to Omnifold

T2K CC0pi event selection

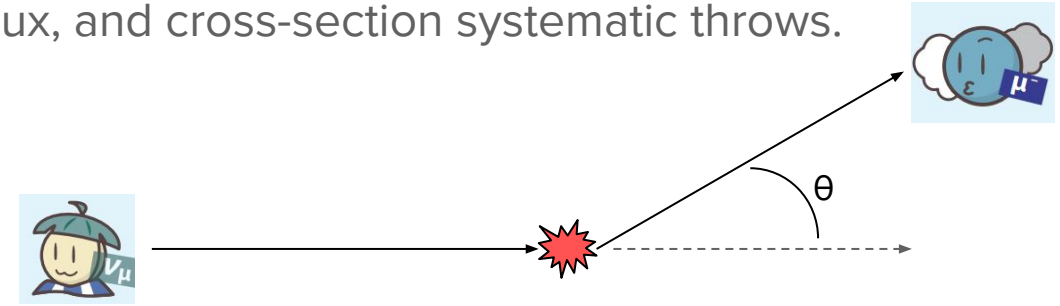
CC0pi signal definition (neutrino mode): **one negatively charged muon**, **zero pions**, and **any number of hadrons** detected in the final state where the vertex was reconstructed in the FGD1 fiducial volume.

Signal samples are categorized by the (sub-)detectors used in the event, and the analysis includes several control samples to constrain background events.

Events in the data set are characterized by muon (proton) kinematics, and also include weights for detector, flux, and cross-section systematic throws.

Corresponding paper:
https://doi.org/10.1103/PhysRevD.101.112001

(see also: https://arxiv.org/abs/2303.14228 or
https://doi.org/10.1103/PhysRevD.101.112004)



Inputs to Omnifold

Omnifold uses the muon kinematics and proton kinematics (if present) for each event parameterized as (log(|p|), cos(θ), φ)

Approx. 200k reconstructed events and 600k truth events

Kinematics are normalized by centering to zero mean and unit standard deviation, and the log of momentum is used

Additionally the sample ID is included as input to the network, but is only used in Step 1 reweighting MC to data ➔ fit both signal and control samples

Some of the background events (e.g. neutral current events) are removed before unfolding due to missing information needed for Omnifold

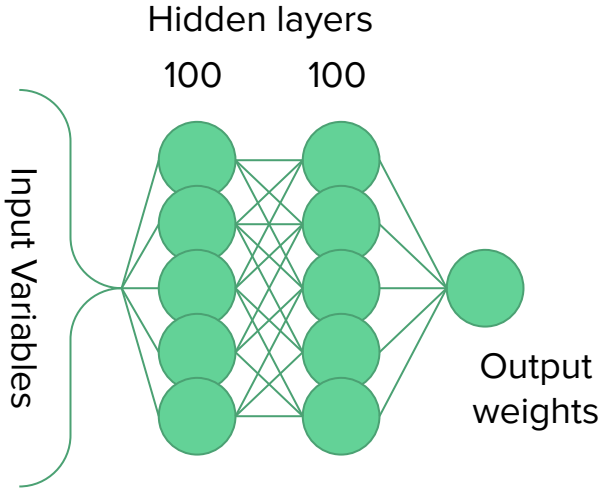
Omnifold network details

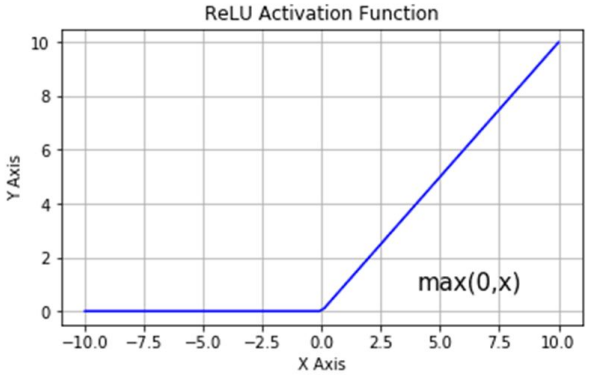
Network architecture (for both Step 1 and 2) is based on two densely connected hidden layers of 100 nodes each with ReLU activation

Single node output layer with sigmoid activation ➔ produce classifier score

Event sample split 75/25 for training and validation respectively

Powered by TensorFlow and (optional) GPU acceleration



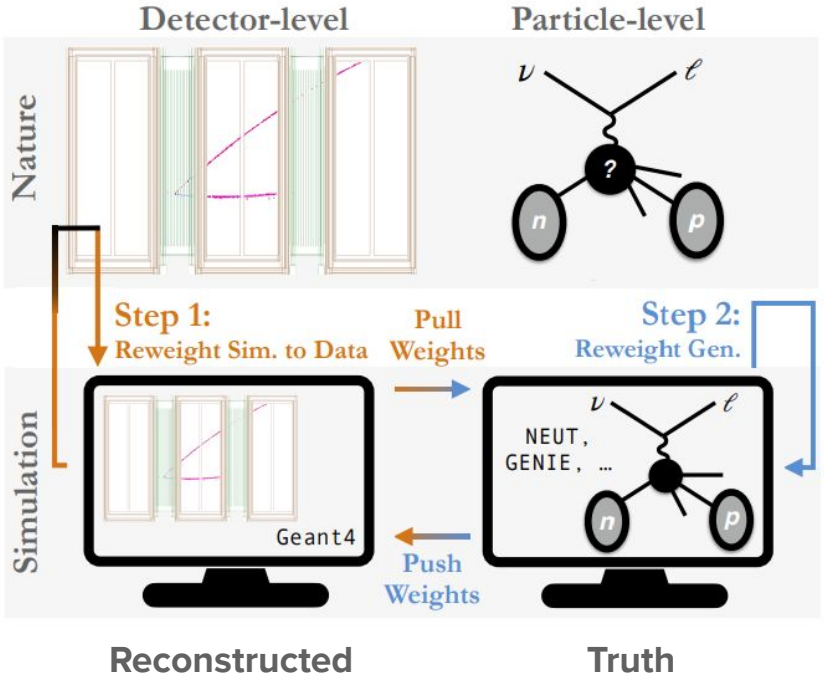


Omnifold procedure

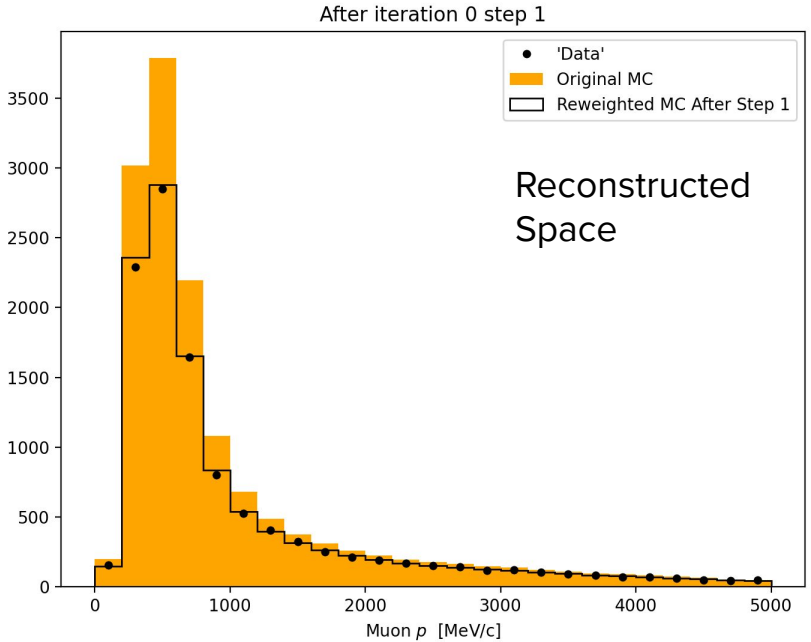
Omnifold is an iterative unfolding procedure performed in two steps:

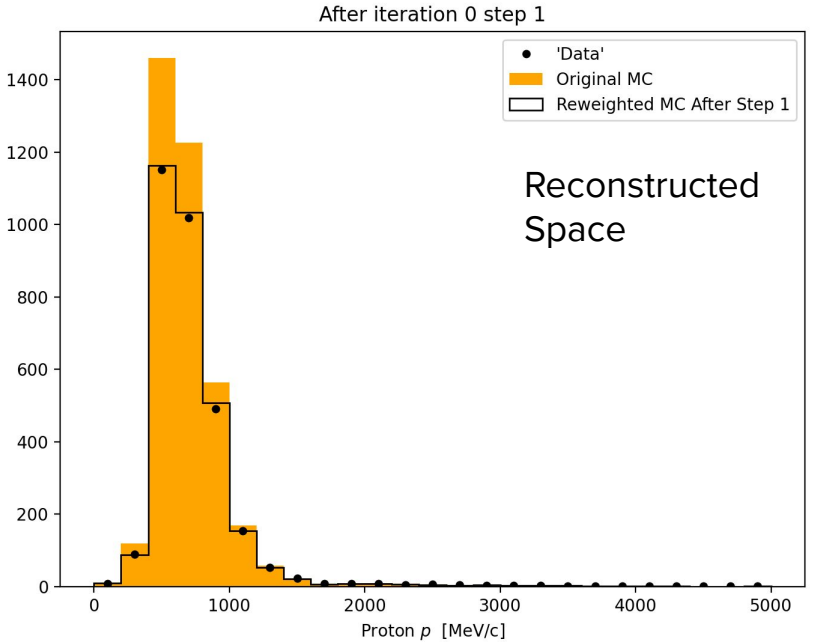
1. Reweight reconstructed MC distribution to (better) match data
2. Reweight nominal truth MC distribution to incorporate information from step 1

This is one iteration, and the method repeats until some convergence criteria is satisfied (or iteration limit is reached)

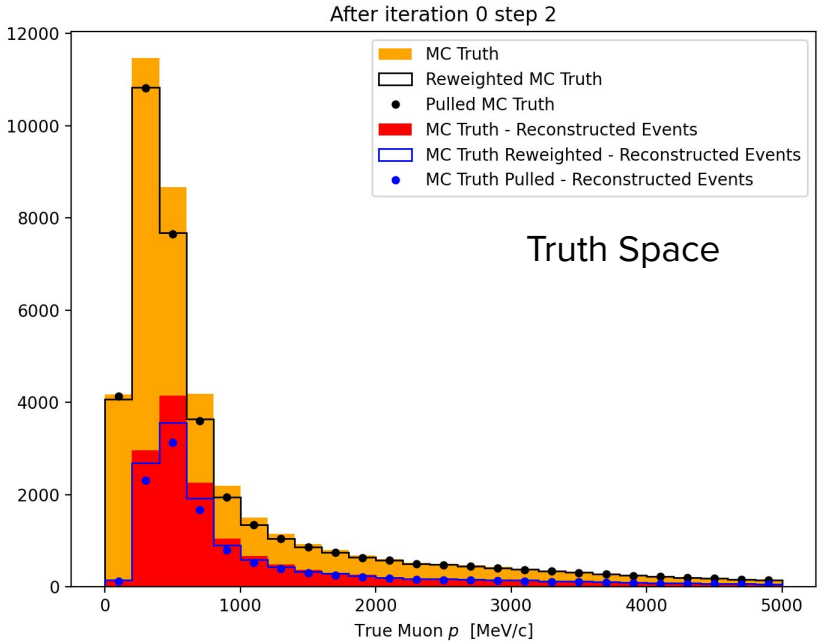


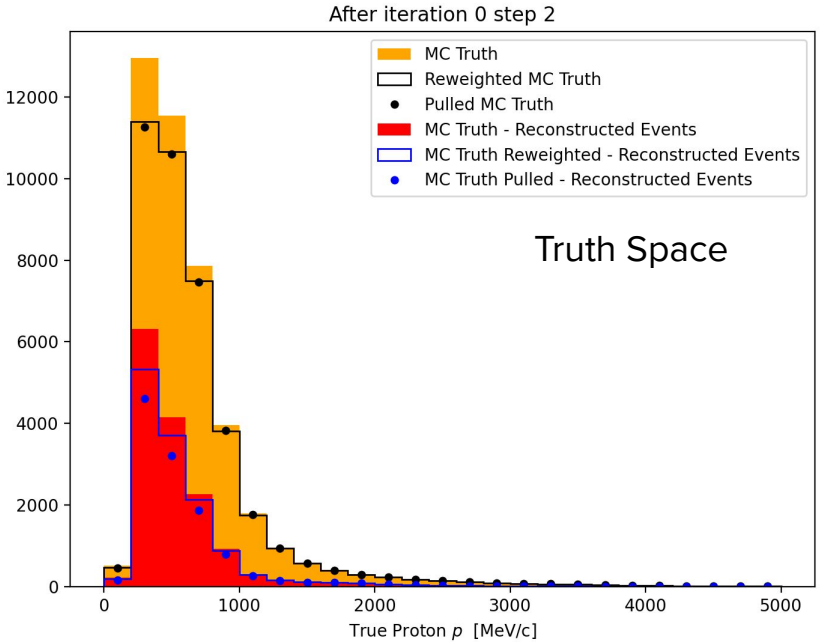
First Iteration – Step 1



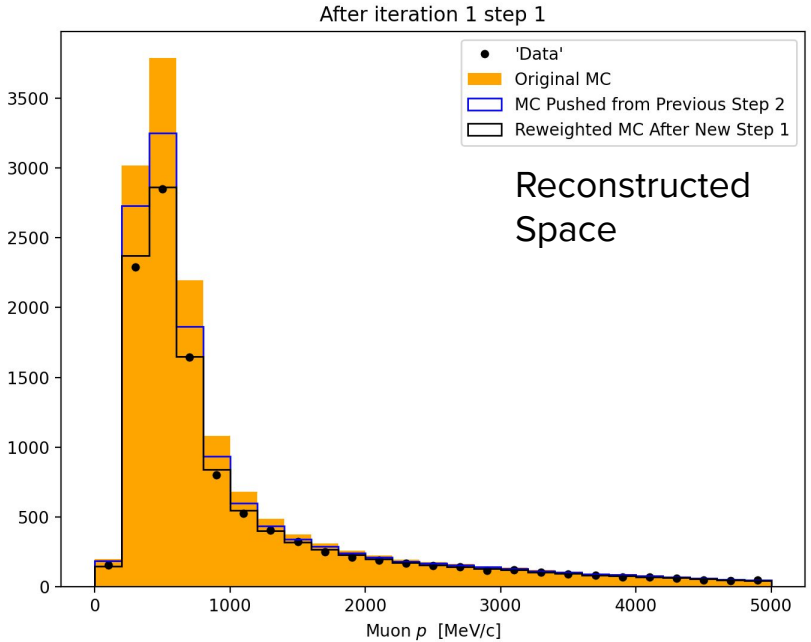


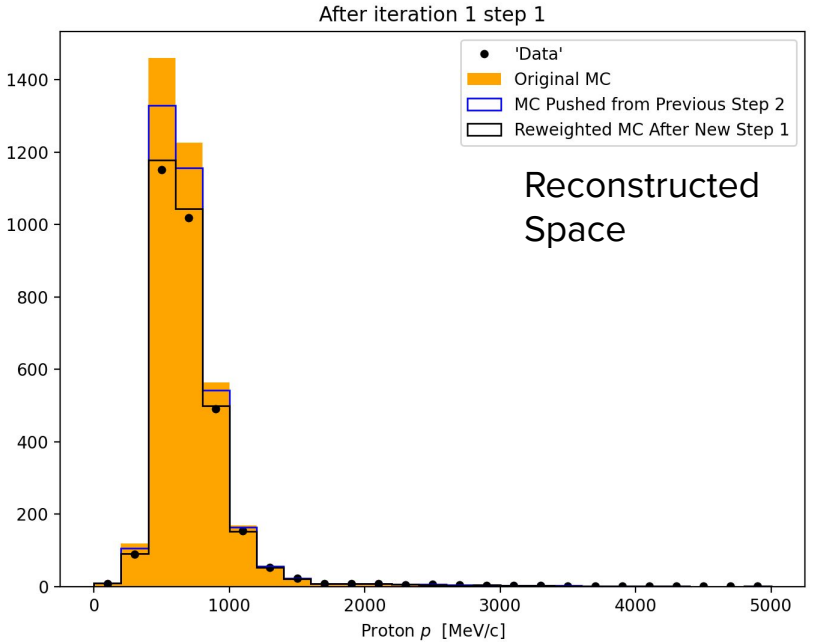
First Iteration – Step 2





Second Iteration – Step 1





Uncertainty, efficiency, and ensembling

Omnifold naturally includes the efficiency correction when performing the unfolding (however this could be separated out and done as another step)

Systematic uncertainty is evaluated through a “universe” (toy throw) method where 100 different systematic variations are processed by Omnifold, and the spread in results gives an uncertainty band

Statistical uncertainty is evaluated by varying the event weights following a Gaussian distribution ➔ N(w, sqrt(w))

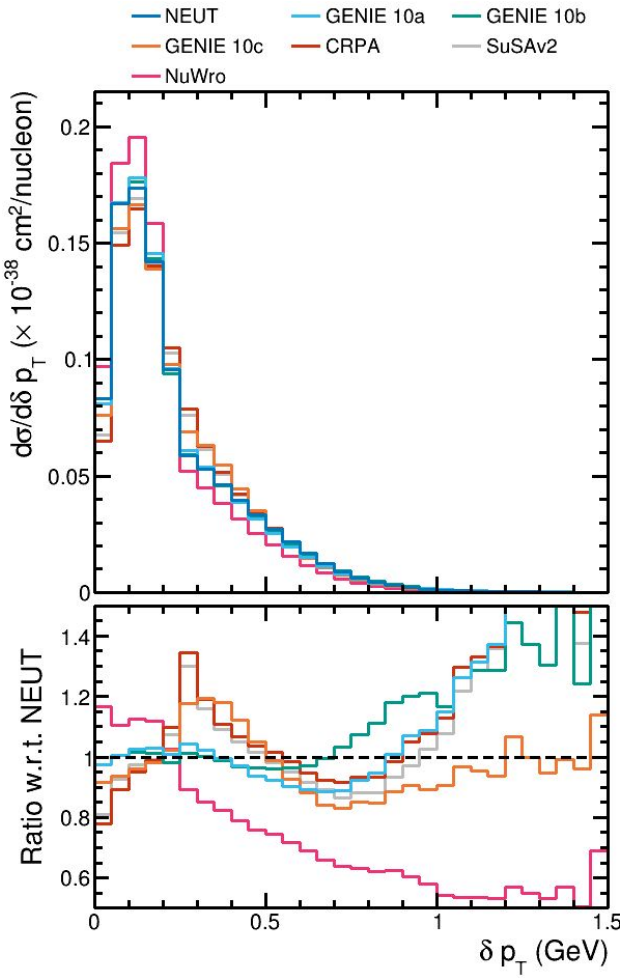
The inherent randomness of training neural networks is mitigated by ensembling using the average result of many trials

Testing Omnifold

Omnifold (and iterative bayesian unfolding) were tested using a fake data study with weights applied to the nominal MC

Weights were derived from the ratio of NuWro to NEUT in transverse momentum imbalance

Note that shape-only weights were used, normalization remains the same



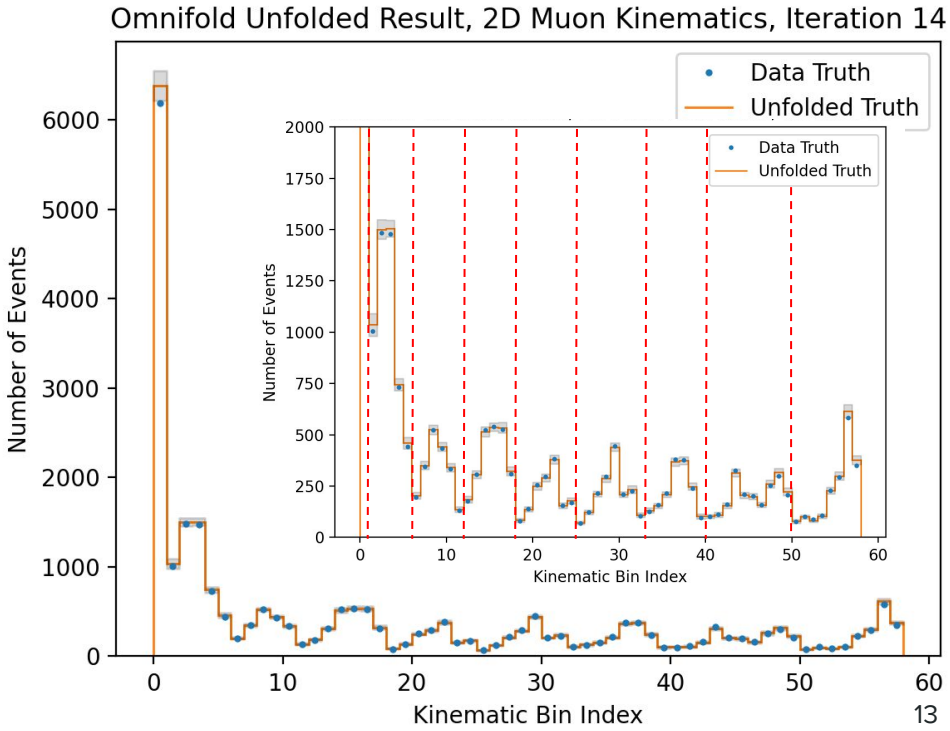
Unfolded Results: Omnifold

Event rate after unfolding with Omnifold compared to the fake data truth

The uncertainty (gray boxes) is evaluated using 100 sys./stat. throws

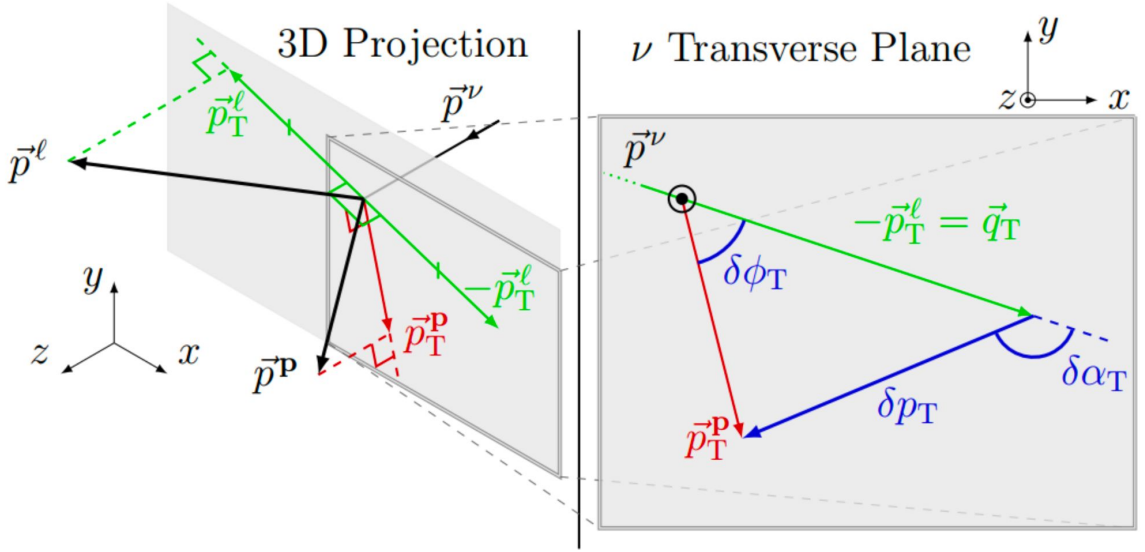
Binned in 2D muon kinematics (p, cosθ) but as a flattened 1D array for plotting purposes

# bins	$\cos(\theta_\mu)$ bin	p_μ bin edges (GeV/c)
1	-1.00, 0.20	0, 30
5	0.20, 0.60	0, 0.3, 0.4, 0.5, 0.6, 30
6	0.60, 0.70	0, 0.3, 0.4, 0.5, 0.6, 0.8, 30
6	0.70, 0.80	0, 0.3, 0.4, 0.5, 0.6, 0.8, 30
7	0.80, 0.85	0, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 30
8	0.85, 0.90	0, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 1.5, 30
7	0.90, 0.94	0, 0.4, 0.5, 0.6, 0.8, 0.8, 1.25, 2.0, 30
10	0.94, 0.98	0, 0.4, 0.5, 0.6, 0.8, 0.8, 1.25, 1.5, 2.0, 3.0, 30
8	0.98, 1.00	0, 0.5, 0.7, 0.9, 1.25, 2.0, 3.0, 5.0, 30



Unfolded Results

Since omnifold is done in the unbinned full phase space, variables which is not trained explicitly can be unfolded.



e.g. Transverse kinematic imbalance

It is important to investigate hadrons to understand the nuclear effect.
It is featured by three variables

- Missing momentum : (δpT, δαT)
- Angle between proton and muon : δφT

$$\delta p_{\mathrm{T}} = |\vec{p}_T^{\,l} + \vec{p}_T^{\,p}|,$$

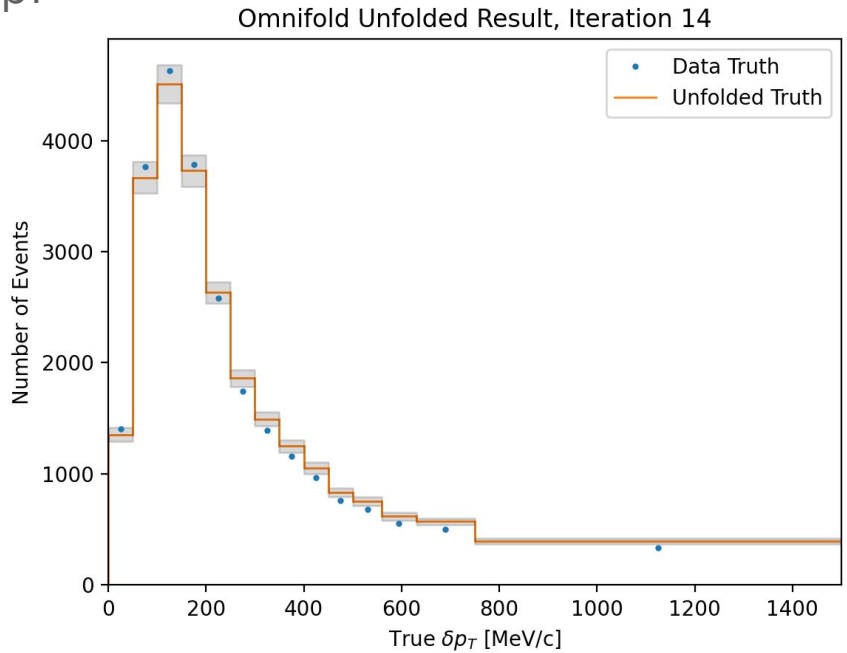
$$\delta\alpha_{\mathrm{T}} = \arccos\frac{-\vec{p}_T^{\,l}\cdot\delta\vec{p}_T}{p_T^l\,\delta p_{\mathrm{T}}},$$

$$\delta\phi_{\mathrm{T}} = \arccos\frac{-\vec{p}_T^{\,l}\cdot\vec{p}_T^{\,p}}{p_T^l\,p_T^p}.$$

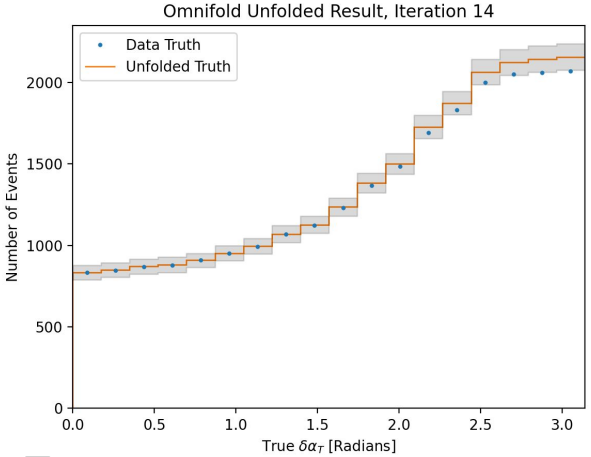
Unfolded Results: Omnifold

Transverse kinematic imbalance

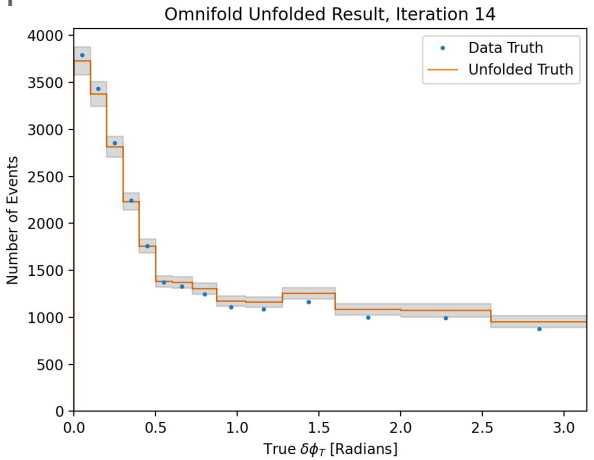
- δpT



- δαT



- δφT



Performance and convergence

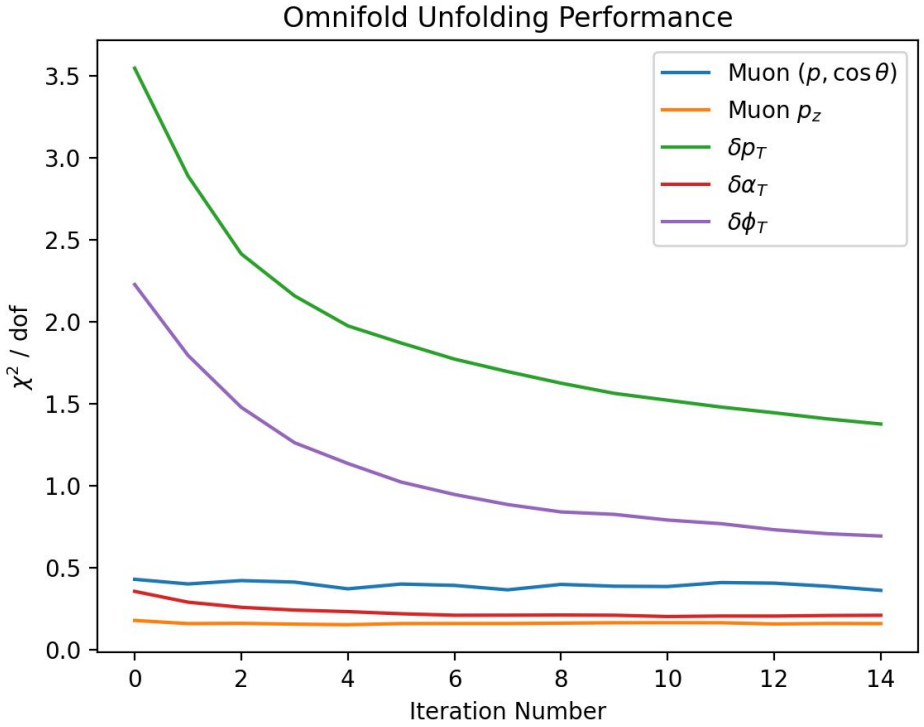
We evaluate performance using a chi-squared metric comparing against the truth-level distributions of our “data”

$$\chi^2 = (\mathbf{p} - \mathbf{q})^T \mathrm{Cov}^{-1} (\mathbf{p} - \mathbf{q})$$

(p : unfolded MC ; q : fake data)

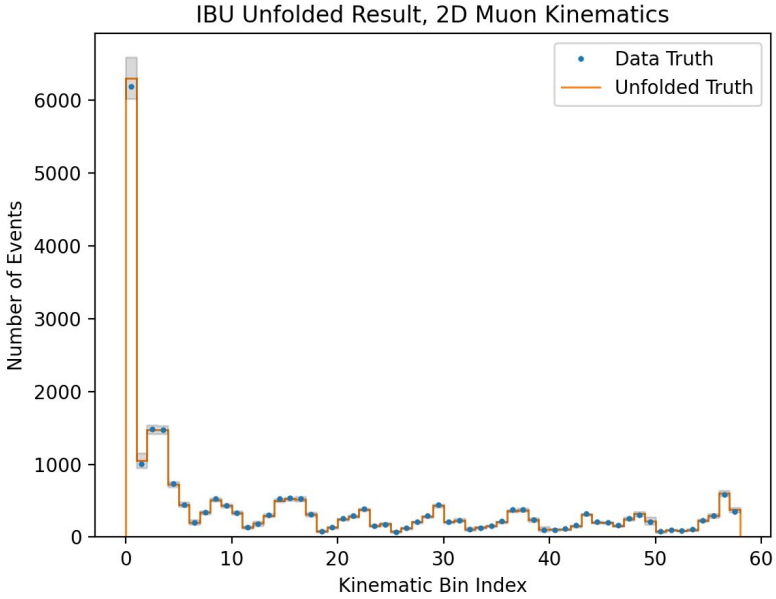
This also gives a look at how the unfolding converges with more iterations

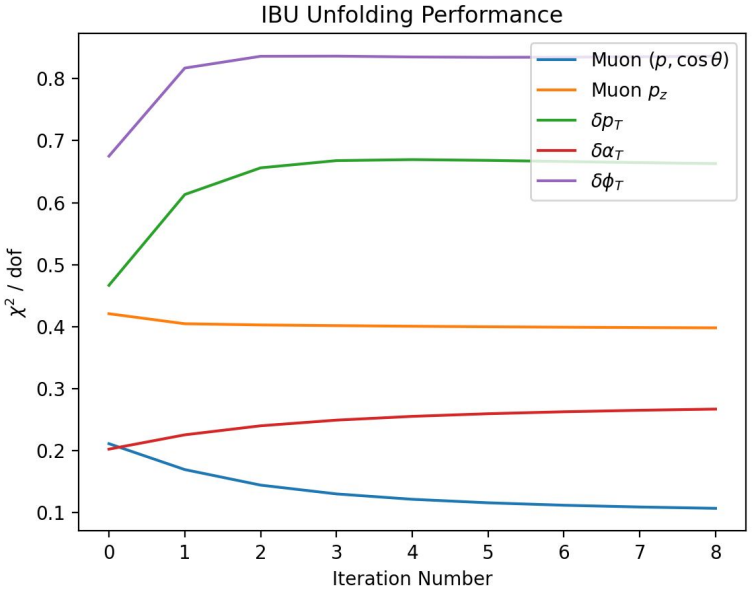
Note that this is just one method to define convergence



Comparison with a conventional method

Comparing to a conventional method (Iterative Bayesian Unfolding / D'Agostini)
The performance of Omnifold is comparable (or better) with that of IBU.





Comparison with a conventional method

Variable	No Unfolding χ^2 / dof	IBU χ^2 / dof	Omnifold χ^2 / dof
2D Muon (p, cos θ)	37.7 / 58	22.9 / 58	21.1 / 58
Muon p_z	5.8 / 12	4.9 / 12	1.9 / 12
p_T	74.7 / 14	8.6 / 14	19.3 / 14
α_T	13.6 / 18	4.1 / 18	3.8 / 18
ϕ_T	46.7 / 14	11.4 / 14	9.7 / 14

Cross section: calculation

Using the number of events, differential cross section with kinematic variables xi is calculated by:

$$\frac{d\sigma}{dx_i} = \frac{N_i}{\Phi T \Delta w_i}$$

Ni : The number of signal events

Φ : Integrated neutrino beam flux

T : The number of targets (nucleon)

Δwi : Bin width correction

NB: Number of signal events calculated using the weights from Omnifold and includes efficiency correction

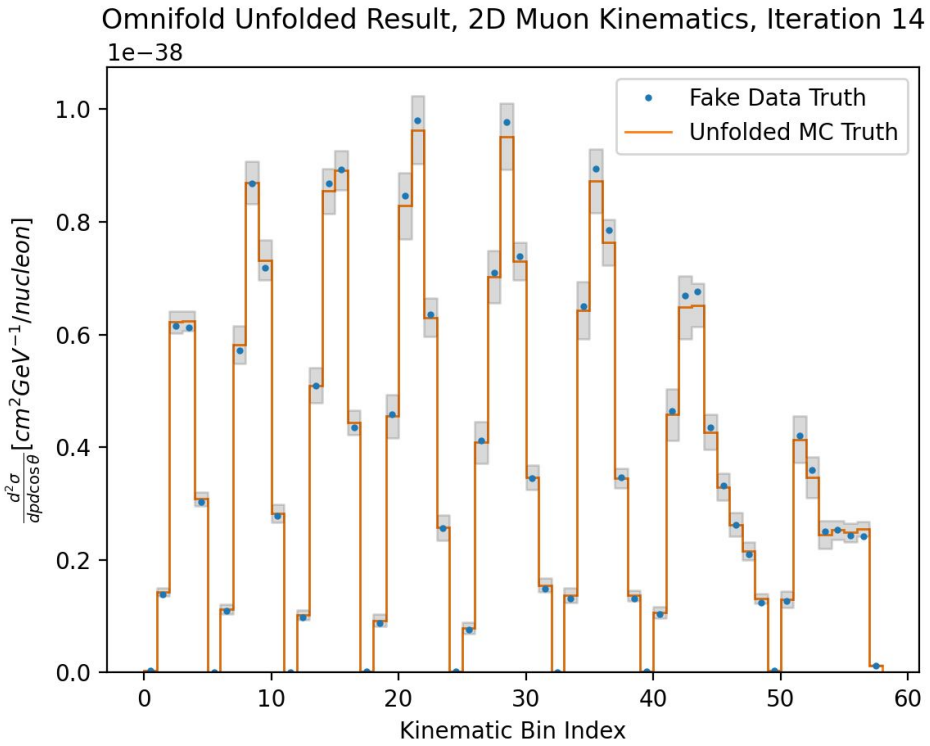
Cross section: Omnifold

Differential cross section with 2D muon kinematics (again as flattened 1D array)

Very preliminary result and some uncertainties are not considered fully

$$\frac{d\sigma}{dx_i} = \frac{N_i}{\Phi T \Delta w_i}$$

Uncertainty from the total flux integral and number of targets not included here



Summary & future work

Omnifold is an unbinned unfolding method utilizing machine learning that naturally work with high dimensionality

Preliminary results using a public T2K data set look promising as an application for neutrino cross-section measurements

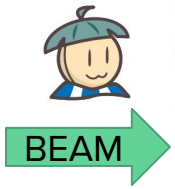
We are studying additional fake data studies to further test the performance of Omnifold and compare to IBU

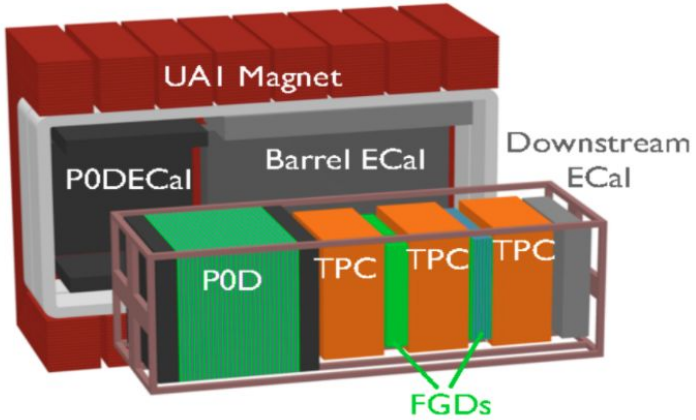
Working on more methods to test convergence, evaluate the various sources of systematic uncertainty, and (as always) fixing various bugs that emerge

Backup slides

T2K near detector: ND280

Off-axis detector (2.5 degrees, 0.6 GeV flux peak)



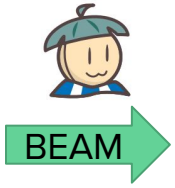


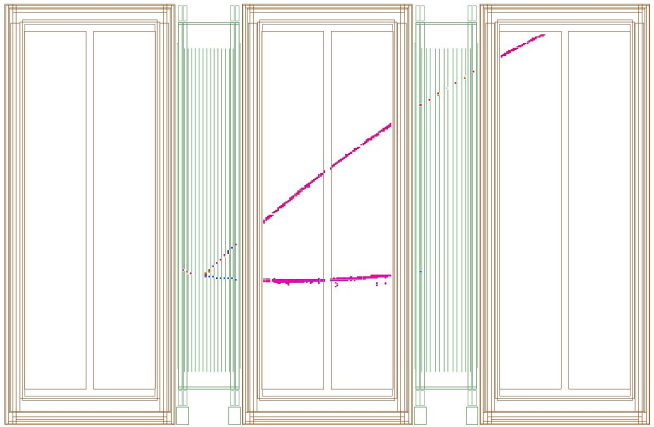
Fine Grained Detectors (FGDs)

- Plastic scintillator tracker
- FGD1 & 2 active carbon (CH) neutrino-interaction target
- FGD2 passive water neutrino-interaction target layers

Time Projection Chambers (TPCs)

- Tracking detectors
- Charged particle momentum
- Particle identification





Highly capable tracking detector with multiple targets. Very good at measuring forward going particles, however is less efficient at high angles.

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: $R_j = \sum_i^N S_{ij}\, T_i \longleftrightarrow T_j = \sum_j^N U_{ij}\, R_i$

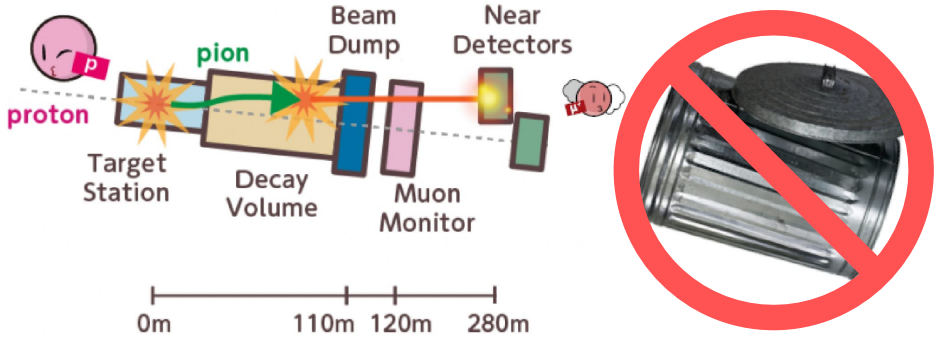
Unfolding is finding the unsmearing matrix **U** given the smearing matrix **S** and removing the detector effects from the measured data (**R** in *j* bins) to get the “true” distribution (**T** in *i* bins)

Simplest method would be to invert **S**, but this is usually regarded as a bad move

Unbinned Reweighting

Given two datasets (where one or both could be MC) where:

- Dataset 1 ➔ sampled from **p(x)**
- Dataset 2 ➔ sampled from **q(x)**



Create or calculate weights **w(x)** = **q(x)** / **p(x)** such that if the weights are applied to dataset 1, it is statistically identical to dataset 2

How do we calculate the weights if we do not know (or can not easily calculate) **p(x)** and **q(x)** – and without estimating them by binning?

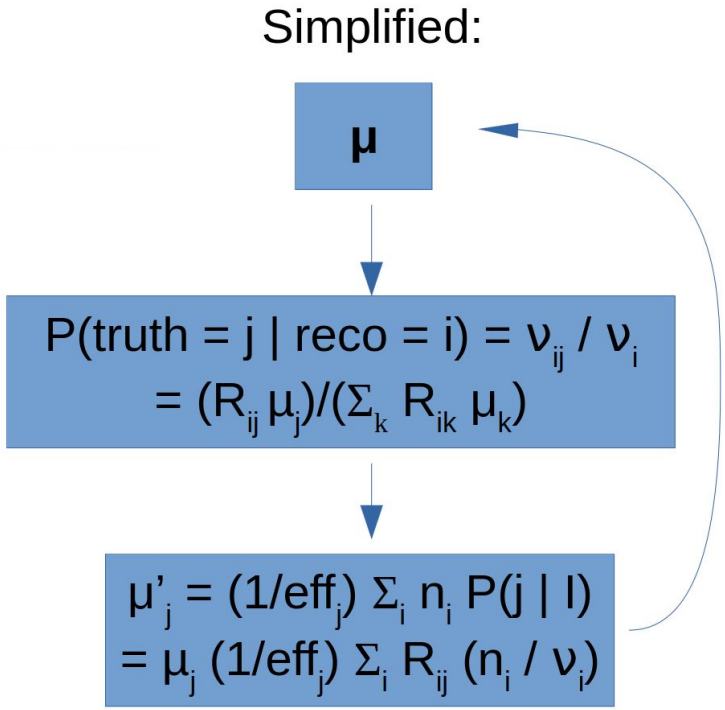
Iterative Bayesian Unfolding

Also known as D'Agonstini unfolding

Iterative procedure for unfolding (for more details refer to https://arxiv.org/abs/1010.0632)

Converges to matrix inversion and/or maximum likelihood as it approaches infinite iterations

Onmifold when binned is equivalent to IBU



From Lukas's slides (Thanks Lukas!)