# Applications of Density Ratio Estimation in Experimental Hadron Spectroscopy

Richard Tyson, JLab
Derek Glazier, University of Glasgow
AI for Hadron Spectroscopy at JLab
5th of June 2025

# Density Ratios

Often we require the ratio of two different distributions
⇒Density Ratio.

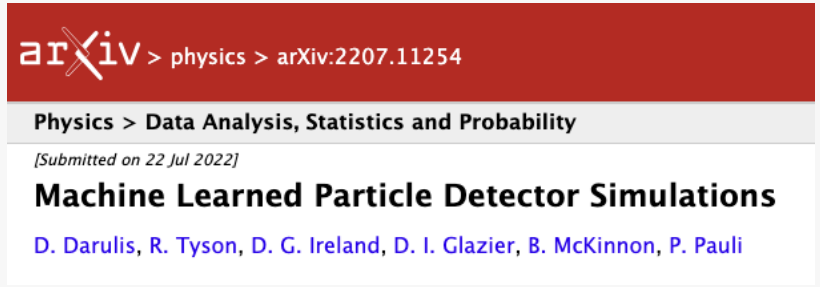
Examples include efficiency calculations, weighting…

In 1D (2D) we may just use 2 histograms and create a 3rd which is their ratio.

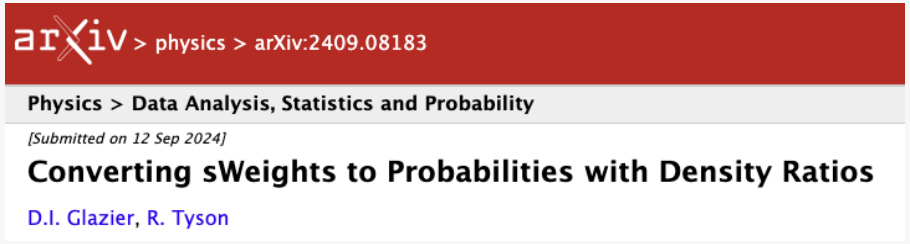
This becomes complicated with more dimensions, or low statistics…

Alternatively we can learn density ratios in N dimensions using binary classification.

## Fast Simulations - arXiv:2207.11254




## Training with sWeights - arXiv:2409.08183



# Motivation: Fast Simulation

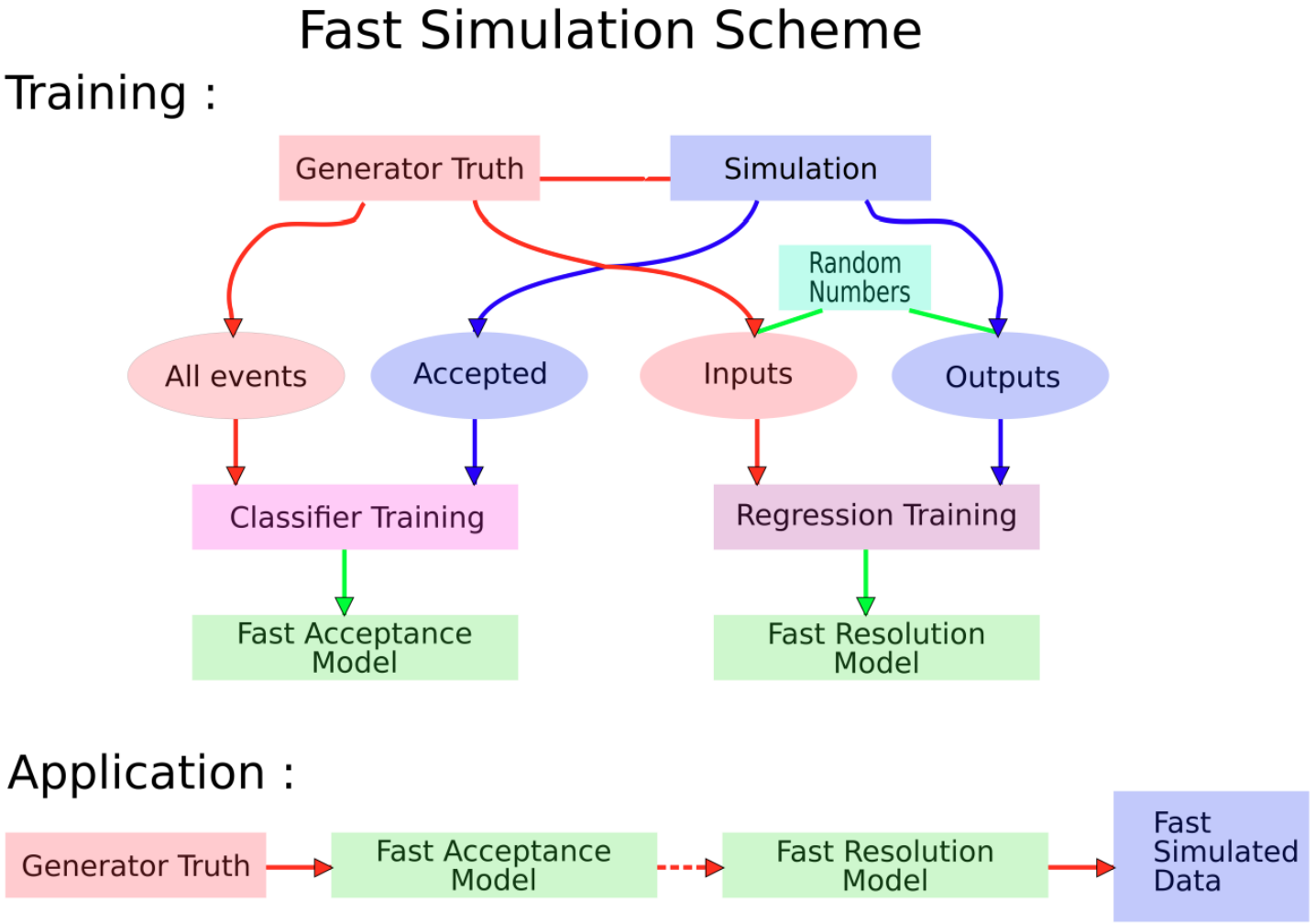
Simulations are key to any High Energy and Nuclear Physics experiment.

Simulations can be computationally expensive. Increased luminosities and detector complexity leads to increased computational overhead.

Instead we want to create ML-based fast MC that can accurately reproduce simulated data.

Two step process:

- Simulate Efficiency with density ratio estimation
- Simulate Resolution



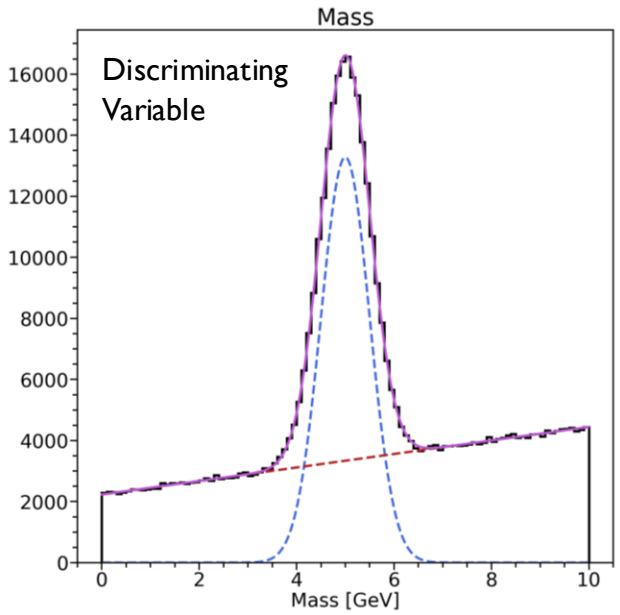
# Motivation: Training with sWeights

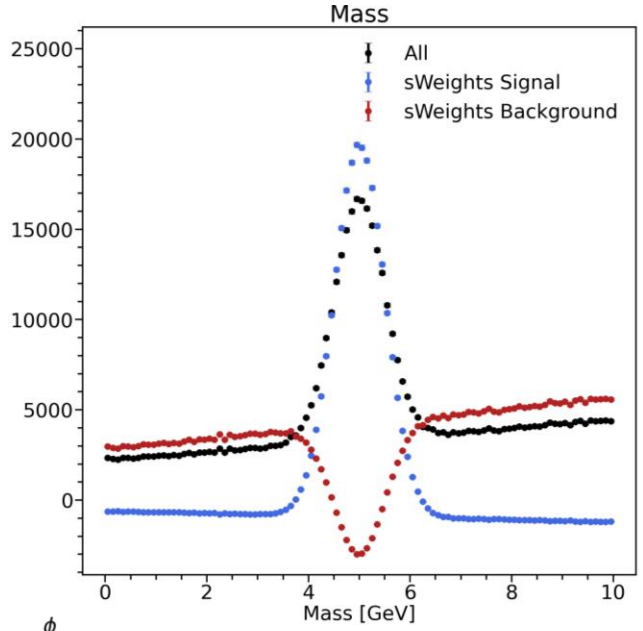
Often rely on simulated data to train ML algorithms, this relies on perfect agreement between simulation and experimental data.

To train ML algorithm with experimental data we need to separate contributions from different event sources.

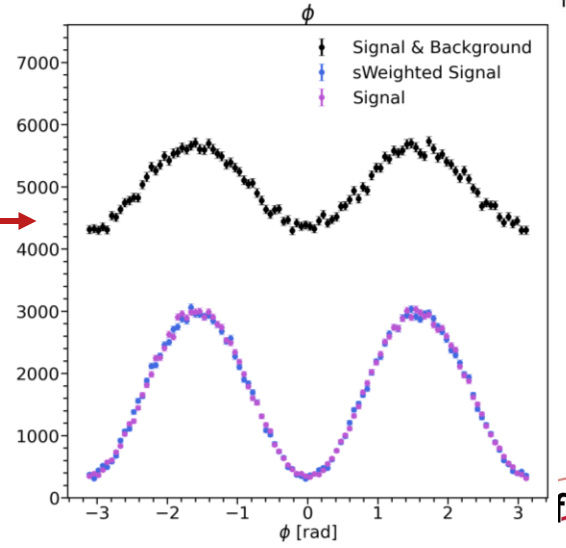
The sPlot formalism aims to unfold the contributions of different event sources to the experimental data.

sWeights are a generalization of side-band subtraction weights to situations where there is no clear region of isolated background which can be used to subtract from the total event sample.




Aim to train ML algorithm on signal.




M. Pivk, F. R. Le Diberder, SPlot: A Statistical tool to unfold data distributions, *Nucl. Instrum. Meth. A*, **555** (2005).

# Motivation: Training with sWeights

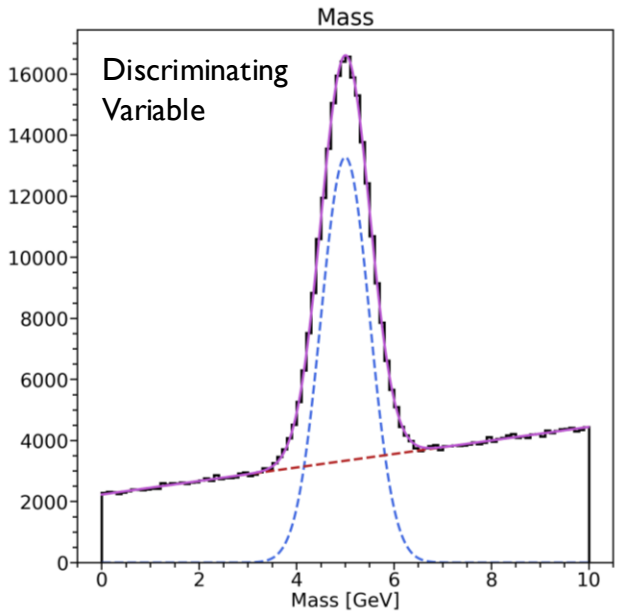
Fit expected pdf to discriminating variables to obtain sWeights that allow to reconstruct distribution of control variables.

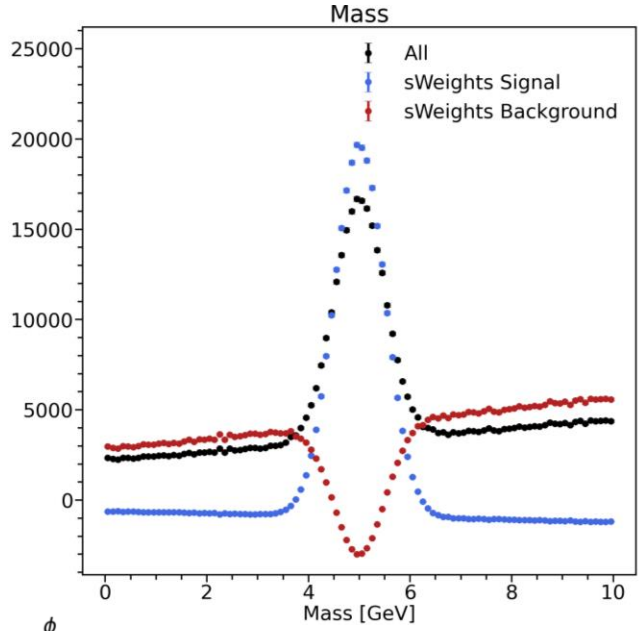
Negative weights are necessary to preserves the statistical properties of the dataset eg correct uncertainties and normalisation. Creates issues for ML training:

$$L\big(f(\mathbf{x})\big) = -\Sigma_i \, w_i \, (y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i)))$$

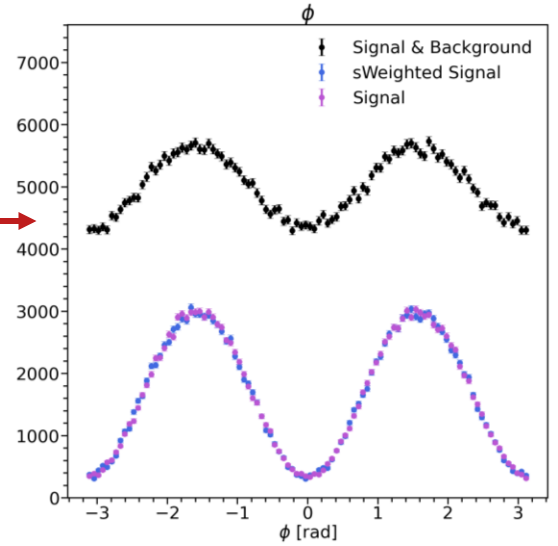
If we recast the weights to positive definite probabilities we can use sWeights to train ML algorithms on experimental data.

This can be done using density ratio estimation.




Aim to train ML algorithm on signal.




M. Pivk, F. R. Le Diberder, SPlot: A Statistical tool to unfold data distributions, *Nucl. Instrum. Meth. A*, **555** (2005).

# Binary Classification for Density Ratios

## Fast Simulation

The efficiency of a detector can be estimated from the ratio of the probability densities for all generated and accepted events:

$$\varepsilon(x) = \frac{D_a(\boldsymbol{x})}{D_{all}(\boldsymbol{x})}$$

Create a training sample with all accepted events as class 1 and all events as class 0. Output for class 1:

$$f(x_i) = \frac{D_a(\mathbf{x})}{D_a(\mathbf{x}) + D_{all}(\mathbf{x})}$$

$$\Rightarrow \varepsilon(x) = \frac{f(x_i)}{1 - f(x_i)}$$

## sWeights

sWeights for a given species are equivalent to the ratio of their probability density over the sum of probability densities of all species in the data:

$$\mathrm{W}_{DR} = \frac{D_s(\mathbf{x})}{D_s(\mathbf{x}) + D_{bg}(\mathbf{x})} = \frac{D_s(\mathbf{x})}{D_{all}(\mathbf{x})}$$

Create a training sample with all events weighted by signal sWeights as class 1 and all events weighted by 1 as class 0:

$$\mathrm{W}_{DR} = \frac{f(x_i)}{1 - f(x_i)}$$

Loss function is now:

$$L\big(f(\mathbf{x})\big) = -\Sigma_i \left(w_i y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i))\right)$$

# Binary Classification for Density Ratios

## Fast Simulation

The efficiency of a detecto
of the probability densit
events:

Create a training
and all events as

$\Rightarrow \varepsilon($

Two key takeaways are:

Creating the training sample with all events as class 0 and signal events as class 1 allows to learn density ratio of class 1 over class 0.

Creating the training sample in such a way allows to use the binary cross-entropy loss function even in the presence of negative weights (and so long as the sum of weights is less negative than the number of events).

## sWeights

hts for a given species are equivalent to the ratio of
ability density over the sum of probability densities
in the data:

$$= \frac{D_s(\mathbf{x})}{D_s(\mathbf{x}) + D_{bg}(\mathbf{x})} = \frac{D_s(\mathbf{x})}{D_{all}(\mathbf{x})}$$

mple with all events weighted by signal
and all events weighted by 1 as class 0:

$$W_{DR} = \frac{f(x_i)}{1 - f(x_i)}$$

is now:

$$) = -\Sigma_i \left(w_i y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i))\right)$$

Similar approach as: B. Nachman, J. Thaler, *Phys. Rev. D*, **102**, 076004 (2020).

# sWeight Dataset

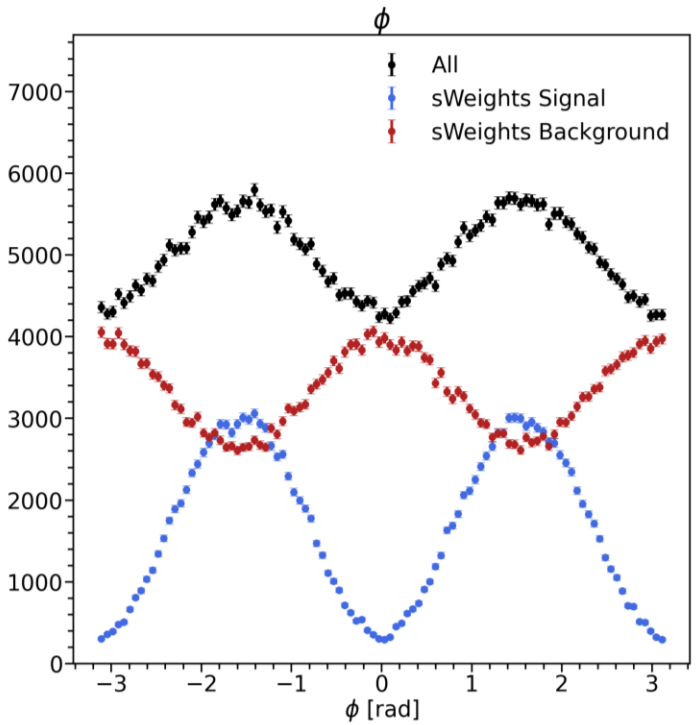
Create toy event generator to produce three dimensional events:

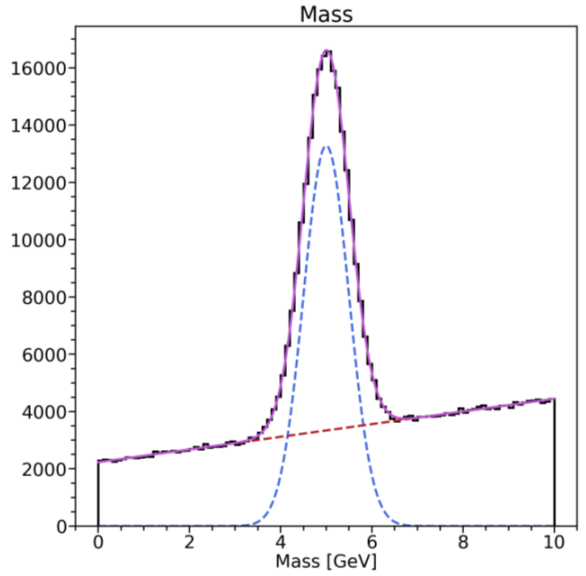
- mass such as an invariant mass as discriminatory variable
- azimuthal (φ) angular distribution
- z = cos θ.

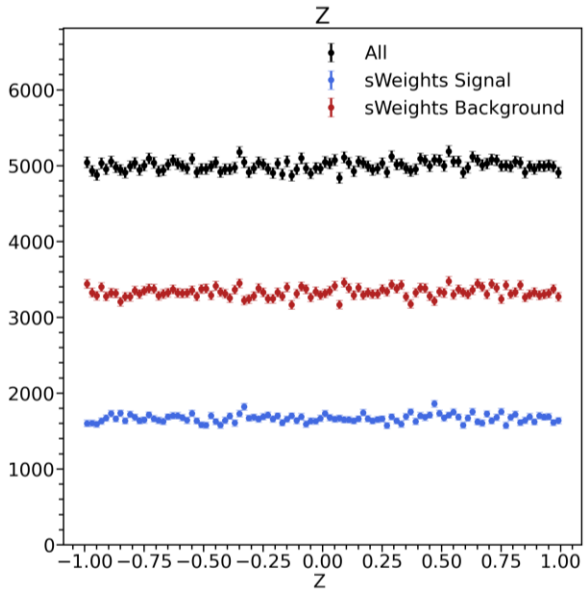
Signal events were generated with a Gaussian distribution in mass and a cos 2φ modulation of amplitude 0.8. Can change eg the frequency

Background events were generated with a linear polynomial distribution in mass and a cos 2φ modulation of amplitude -0.2.

The aim is to measure the signal amplitude in φ by unfolding the signal distribution in the control variables φ and z




See github repo

# sWeight Performance

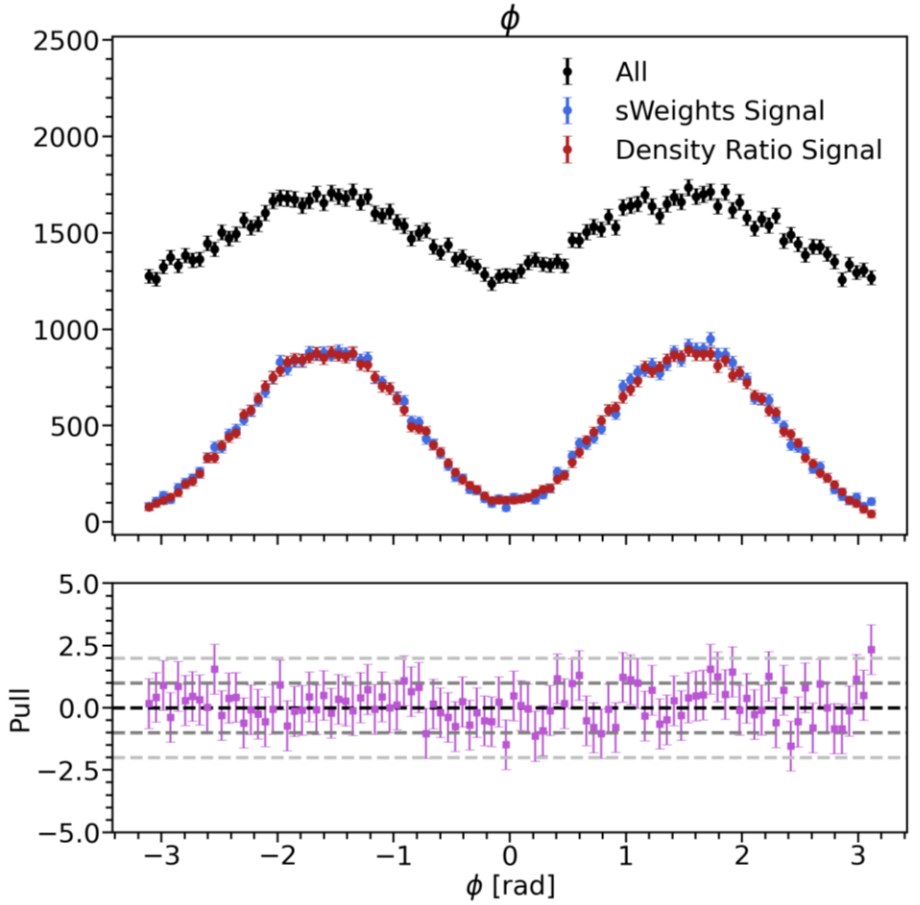
Repeat test 50 times, obtain mean amplitude and uncertainty and standard deviation of the amplitude. The expectations are:

- Mean should be consistent with the nominal value of 0.8
- Mean uncertainty and standard deviation should be numerically similar i.e. the fluctuation of results is consistent with the calculated uncertainty

sWeighted uncertainty is calculated by taking the sum of the squared sWeights.
⇒ This doesn't work with converted weights. Either propagate sWeight uncertainty or convert it with density ratios

Fast training & prediction rates (order of 100s of kHz).

| Method (signal:bg) | Mean | σ | $\frac{\sigma}{\text{uncertainty}}$ |
|---|---|---|---|
| sWeights | | | |
| (1:2) | 0.802 ± 0.0089 | 0.0082 | 0.92 |
| (1:9) | 0.804 ± 0.0274 | 0.0244 | 0.89 |
| drWeights | | | |
| (1:2) | 0.807 ± 0.0092 | 0.0093 | 1.01 |
| (1:9) | 0.793 ± 0.0285 | 0.0260 | 0.91 |

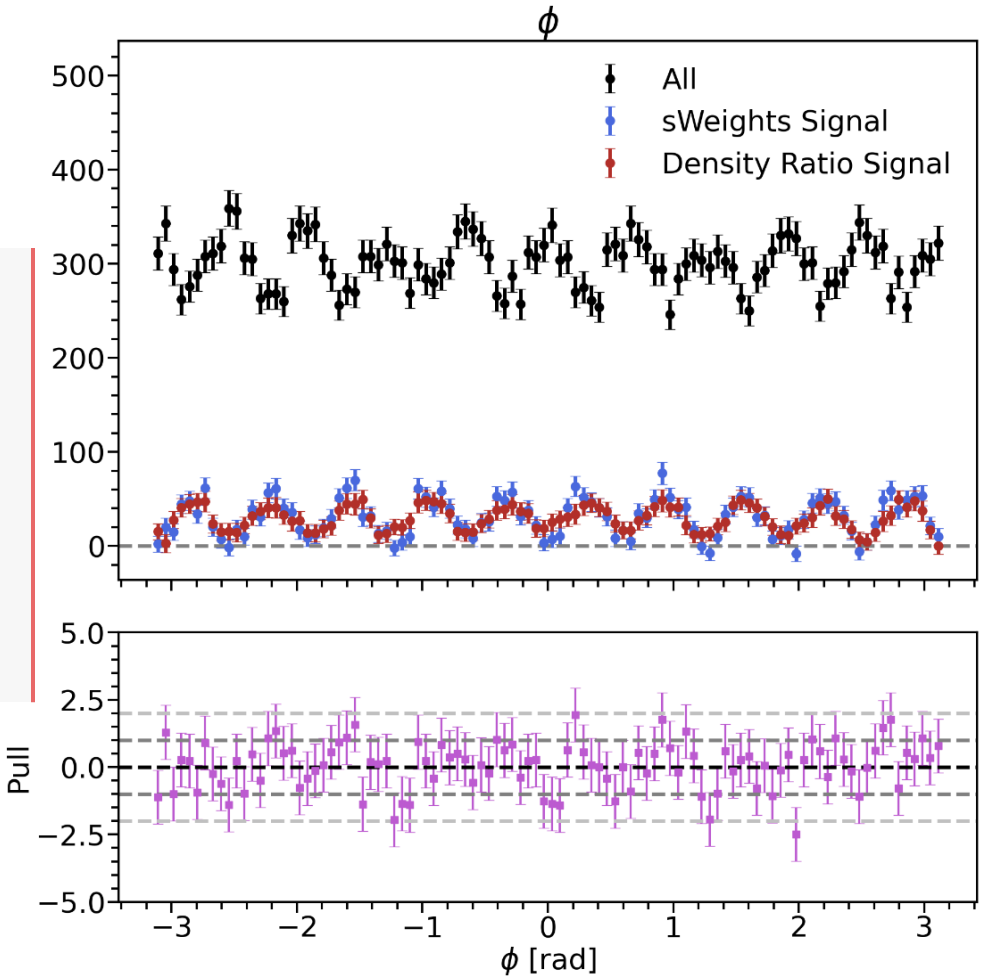


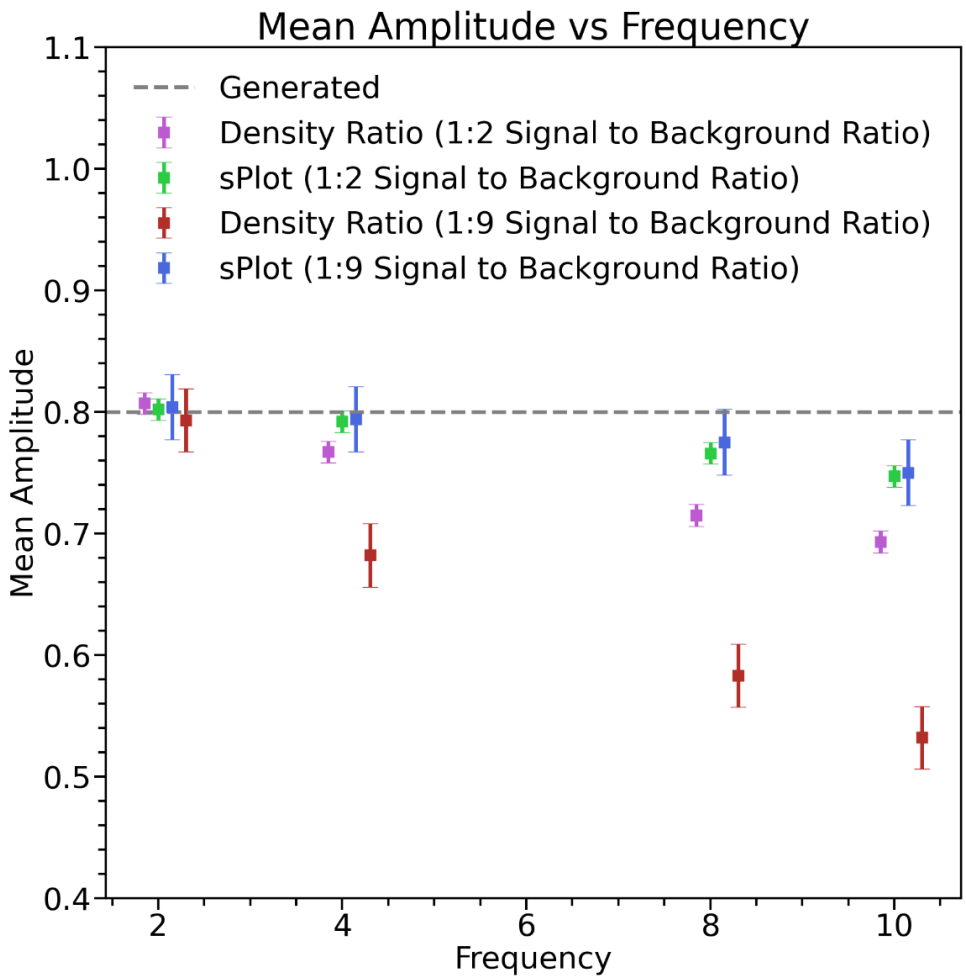

$$\text{Pull} = \frac{(N_{sW} - N_{drW})}{\sqrt{\sigma_{dr}^2 + \sigma_{sW}^2}}$$

# Bias

Learning algorithms introduce some bias. Typically seen as smoothing over of sharp features.

⇒ Increase the frequency of the modulation with a (1:9) signal to background ratio to get sharper features



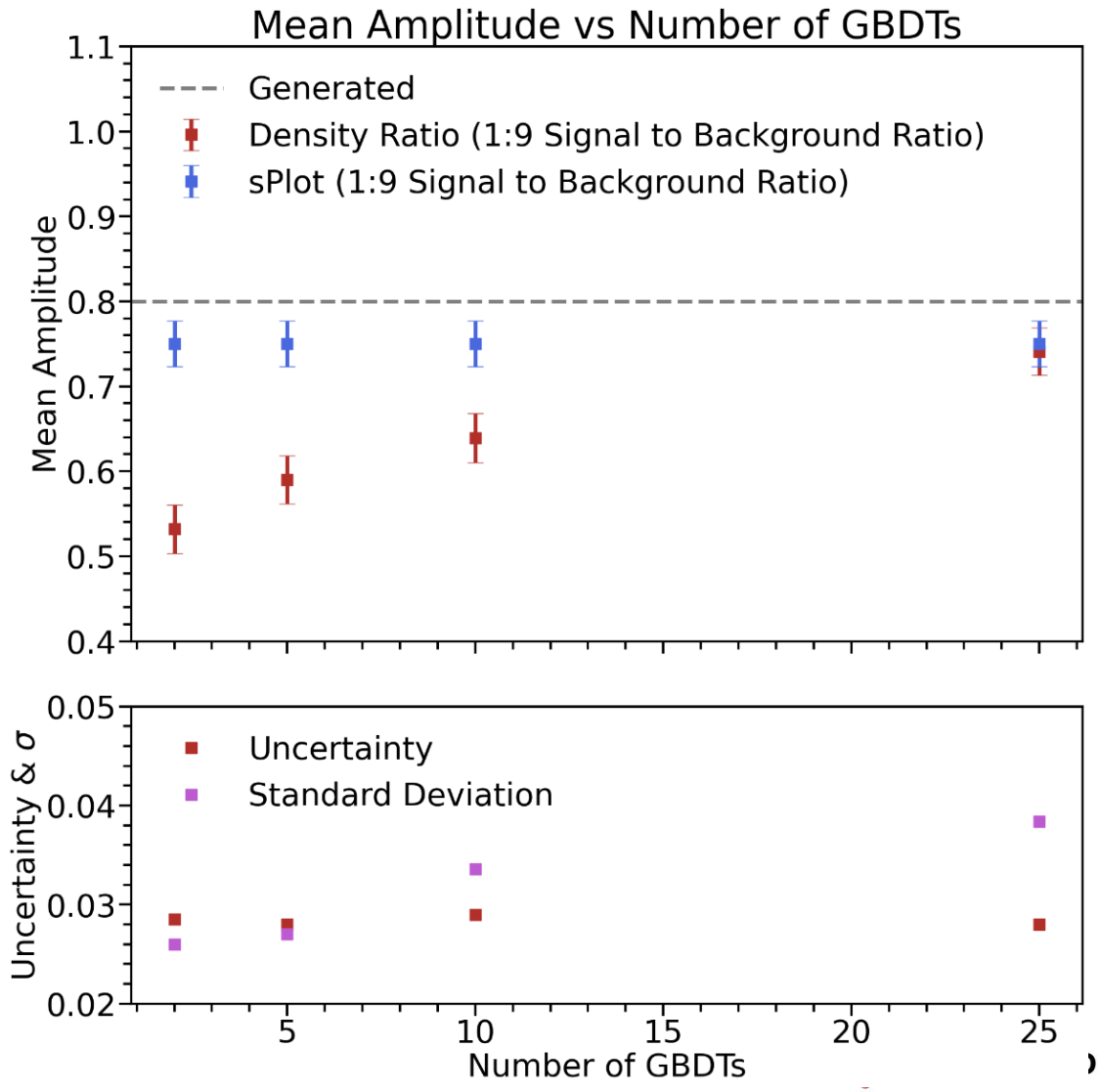

Jefferson Lab

# Correcting the Bias

We can apply the method as many times as we want, ie we correct the learned density ratios. If weights are similar, correcting weights should be 1 (or close to).

Class 0 has all events with weights produced by previous density ratio estimation, predicted weights are product of the weights obtained by all models.

Better performance but training is less stable.



# Fast MC Dataset: CLAS12

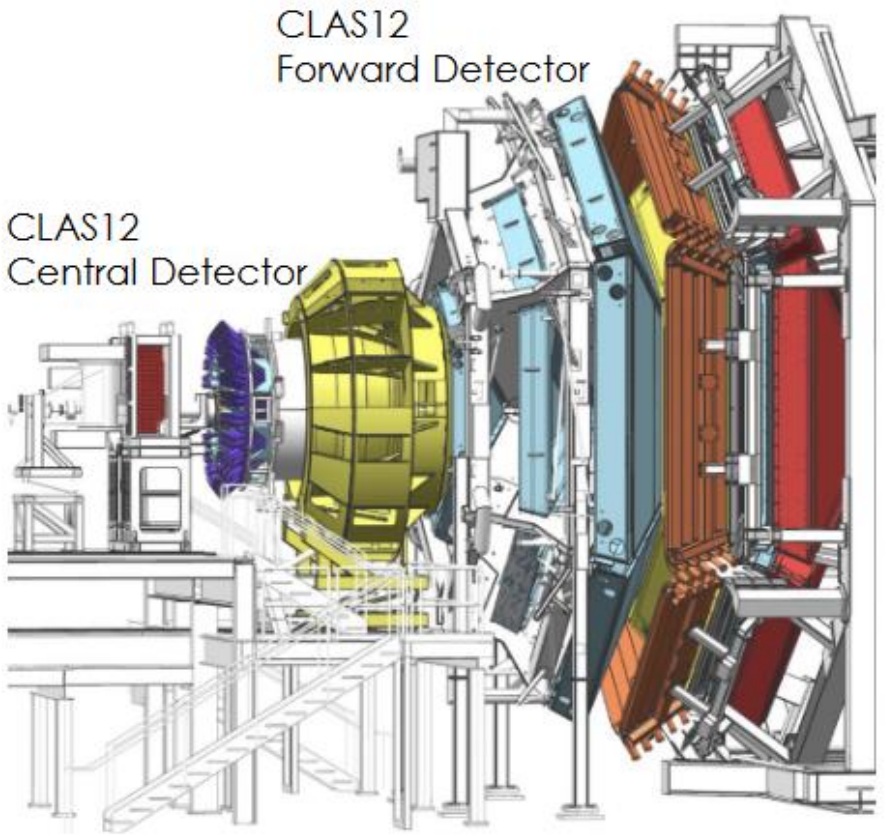
Aim to reproduce simulation from dedicated Geant4 framework GEMC.

Forward Tagger has polar angle coverage of 2.5 to 4 degrees.

The Forward Detector has polar angle coverage of 5 to 35 degrees. Has six sectors in azimuthal angle

The Central Detector has polar angle coverage of 35 to 125 degrees.

Here I'm showing examples based on DVCS $ep \rightarrow e'\gamma p$ and $ep \rightarrow e'nK^+K^-\pi^+$ at CLAS12, also tested with a toy detector (see github repo for all examples).

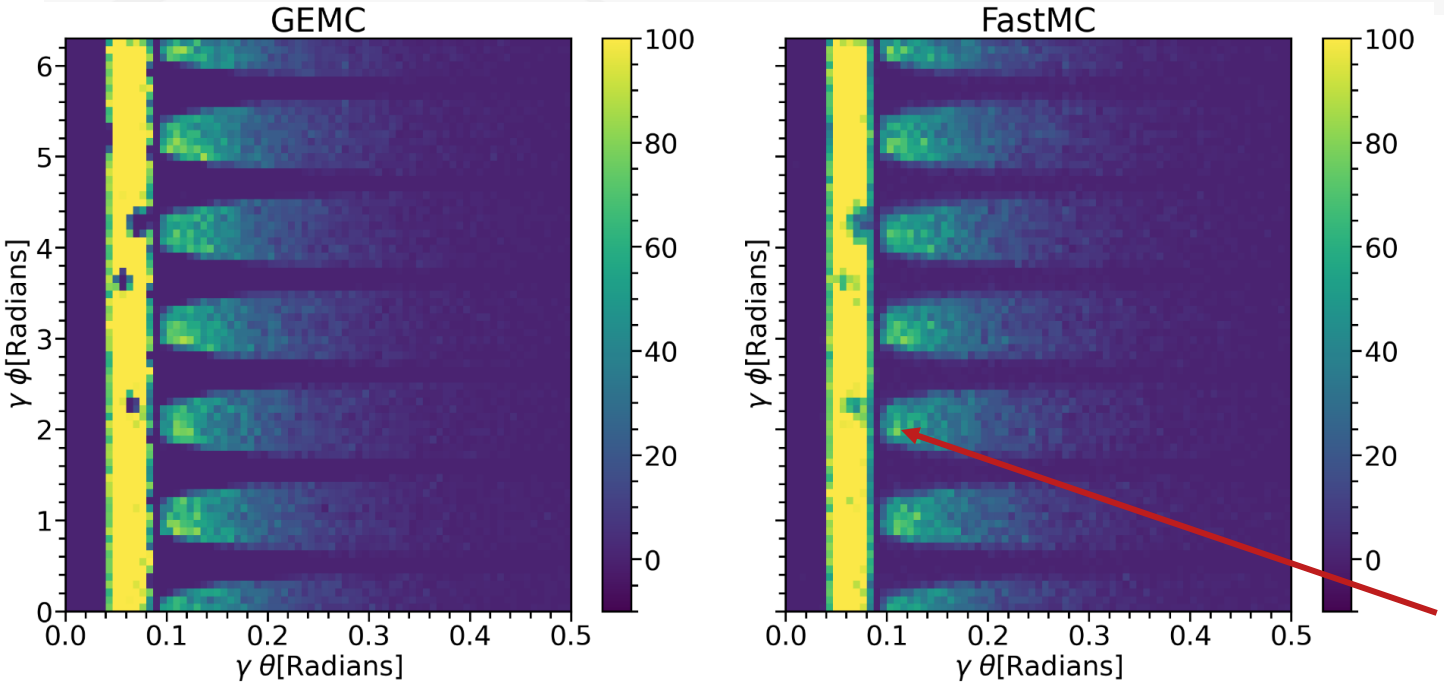


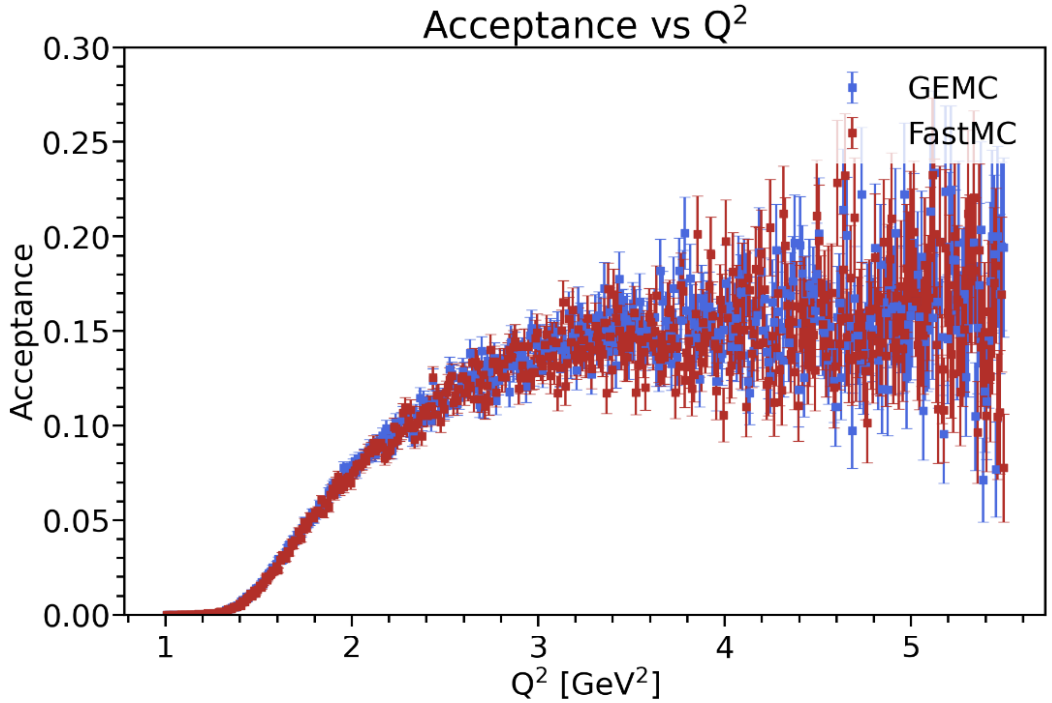
# Fast MC Efficiency

Efficiency Density Ratio estimation using first a neural network then correct with a gradient boosted decision tree.

In DVCS ($ep \rightarrow e'\gamma p$) electron is detected in the forward detector, photon in the forward tagger and forward detector, proton in the central detector.

Training to reproduce GEMC (~1 Hz), achieve ~10 kHz prediction rate.




Sharp Features

Jefferson Lab

# What about Resolution?

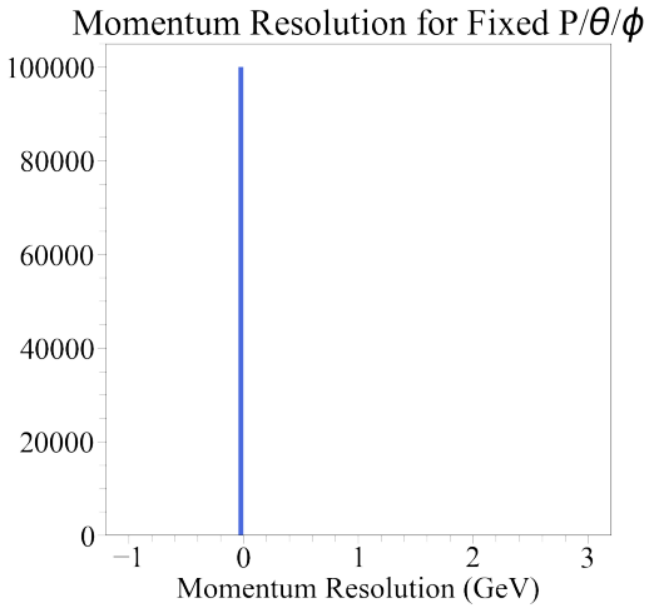
If the detector measures $p_r = p_{true} + \delta p$ , aim to learn the $\delta p$ distribution (and on angles or other quantities).

Overfit decision trees, accurately produces $\delta p$ but they produce one value of $\delta p$ for any one point in the training feature space.

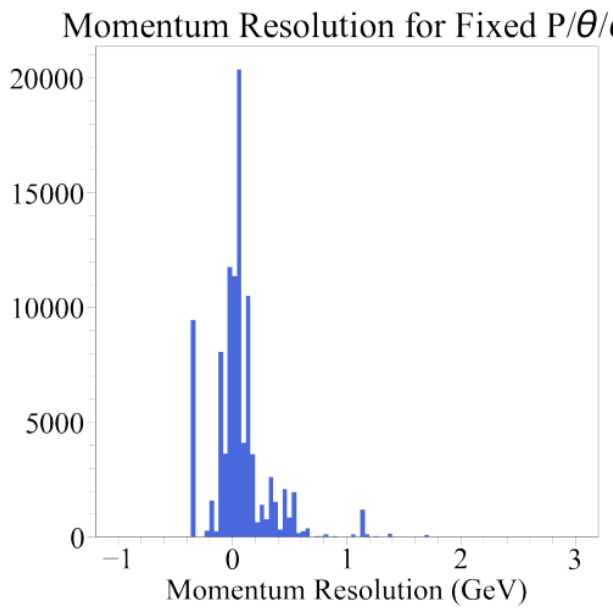
Add random input variables, complexifies input feature space. Now predict different values of $\delta p$.

Train N decision trees with different random inputs, ie N different predictions. Pick DT at random during prediction time.

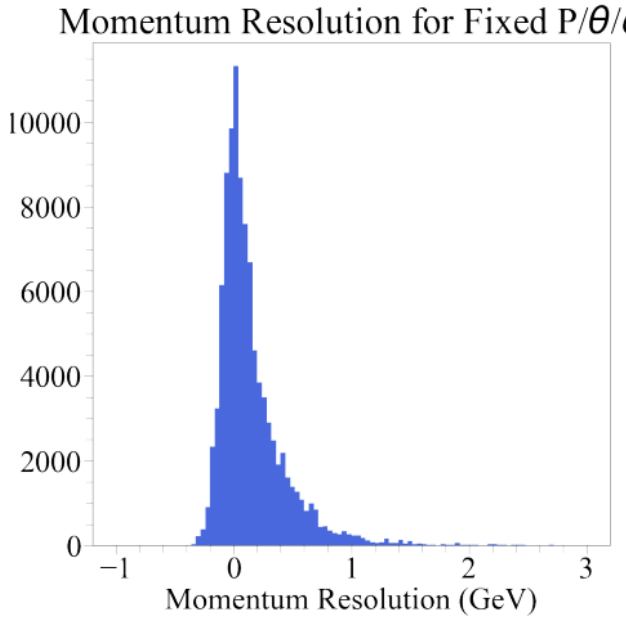
From toy detector



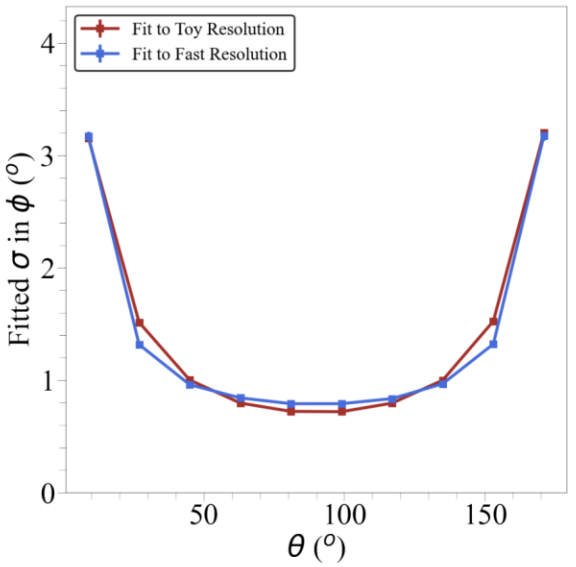

No Random Input, 1 DT




5 Random Input, 1 DT




5 Random Input, 50 DTs




5 Random Input, 50 DTs

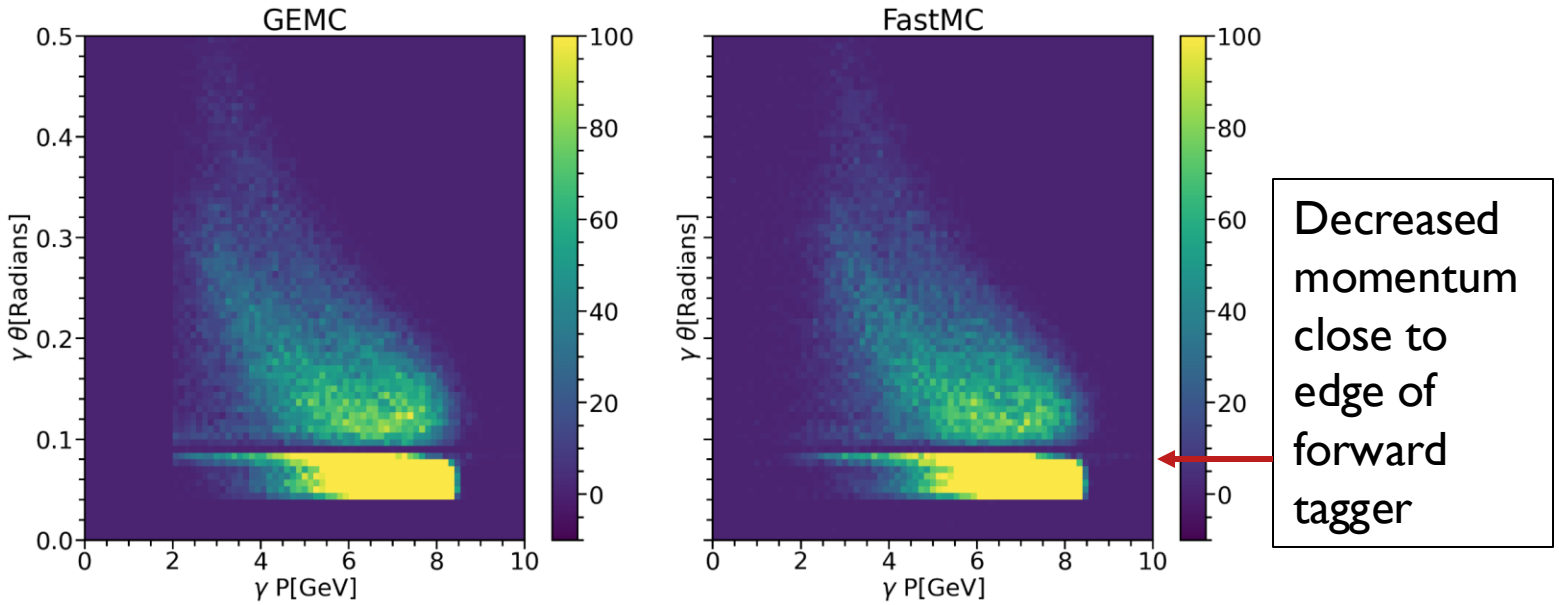
# Fast MC Efficiency and Resolution

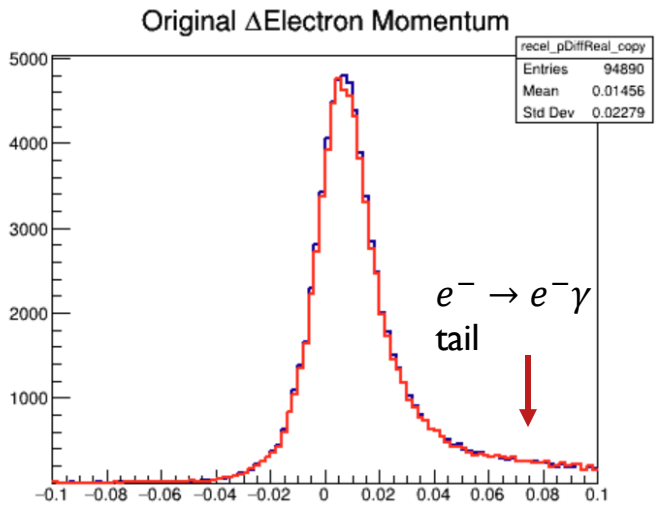
Accurate multidimensional acceptance and resolution.

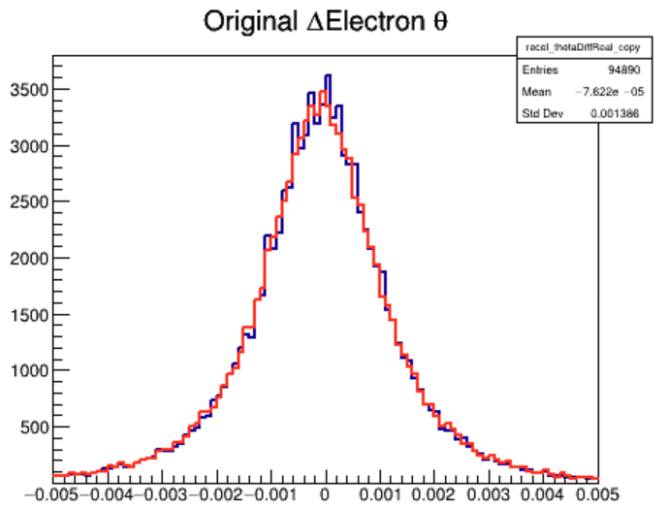
Able to predict any resolution distribution (eg non gaussian, tails etc).

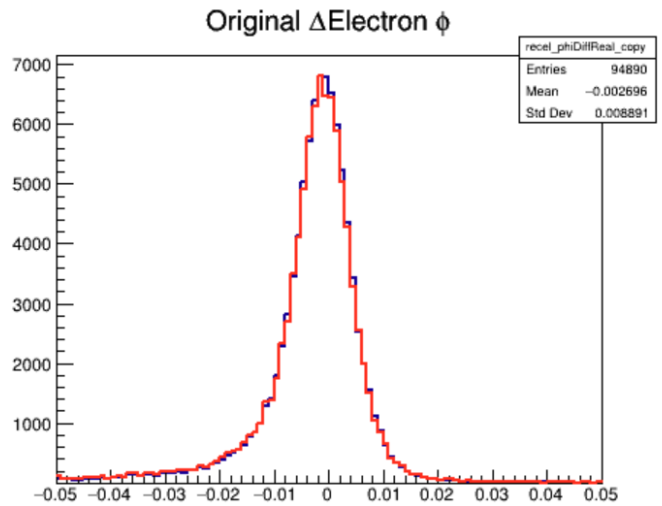
~$10^4$ increase in prediction rate.

Simple algorithms: works out of the box, no further tuning required



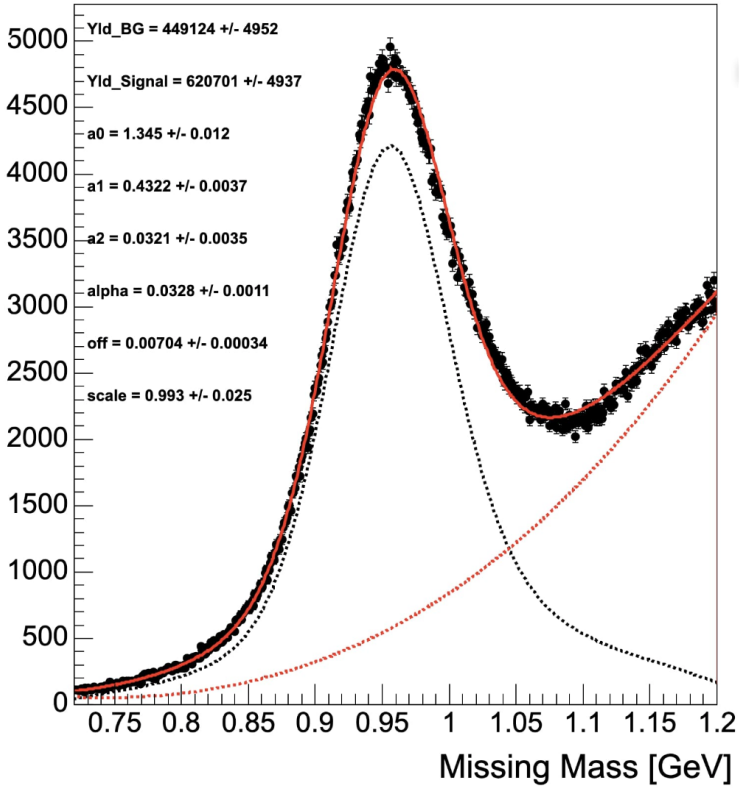



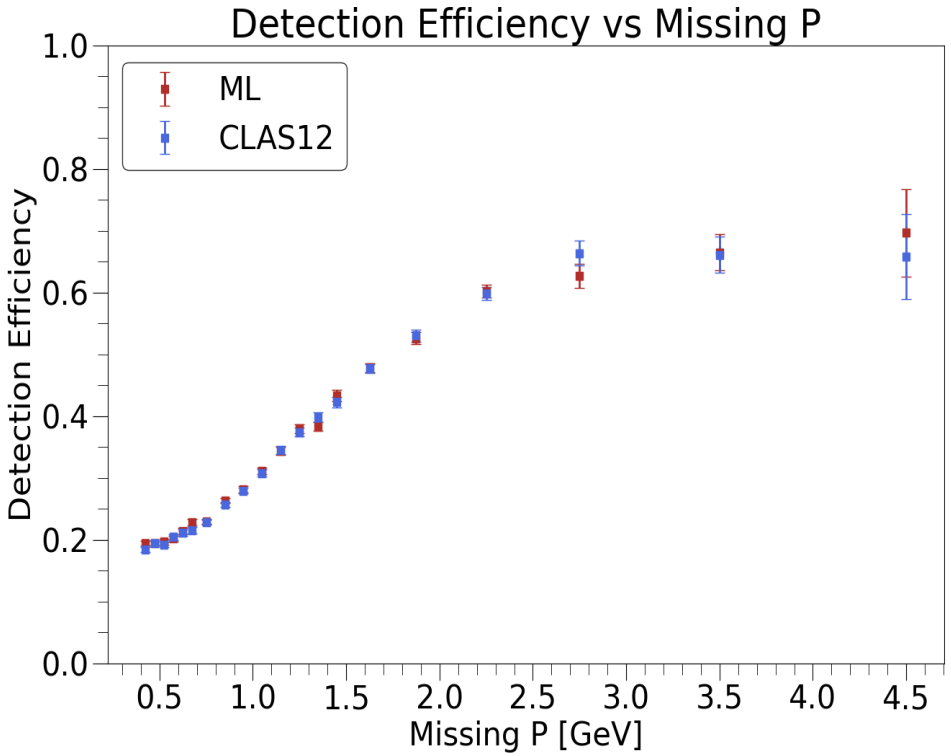
# Learning Efficiencies from Experimental Data

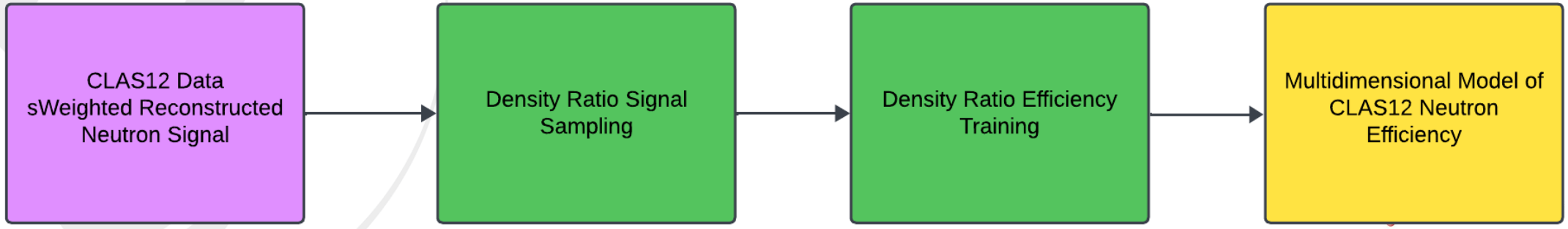
Can use both sWeight and Efficiency Density Ratio learning together.

Aim to learn neutron detection efficiency from ratio of $ep \rightarrow e'\pi^+n$ to $ep \rightarrow e'\pi^+(n)$ without relying on simulation.

Produce sWeights from template fit to neutron peak in missing mass of $ep \rightarrow e'\pi^+(X)$.



# Conclusion

Density ratios can be used to model many common problems in hadron spectroscopy.

Density ratios can be accurately learned using binary classification algorithms.

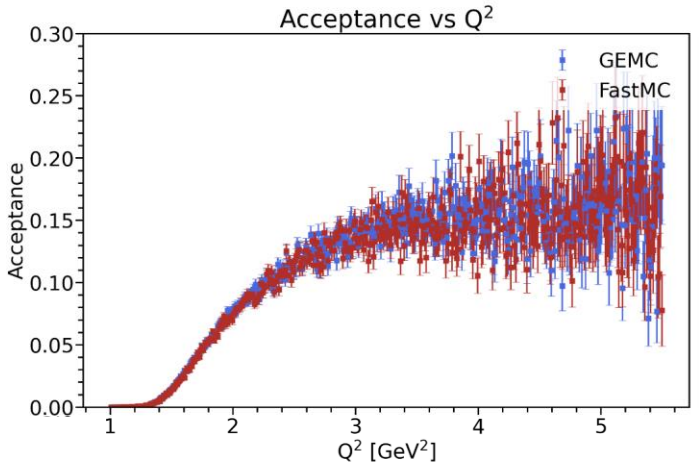
Presented two use cases:

- Efficiency mapping for fast simulation
- Learning sWeights to produce training samples from experimental data

These two use cases can be combined to map detector efficiencies from exclusive reactions in experimental data.

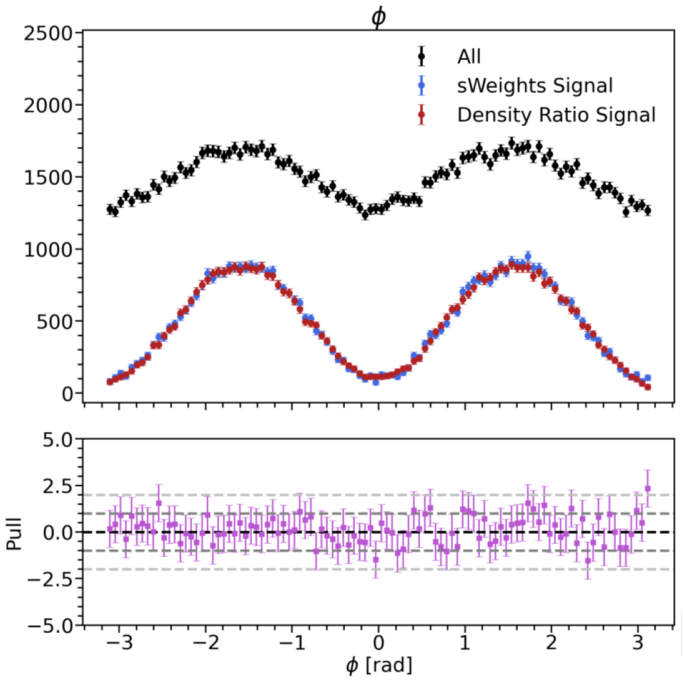
Other applications will exist: eg converting weights to positive probabilities for likelihood fitting.

Any suggestions?




Macparticles fast MC github repo

DR4sWeights github repo



# Back-Up Slides



Jefferson Lab

clas

# Density Ratios

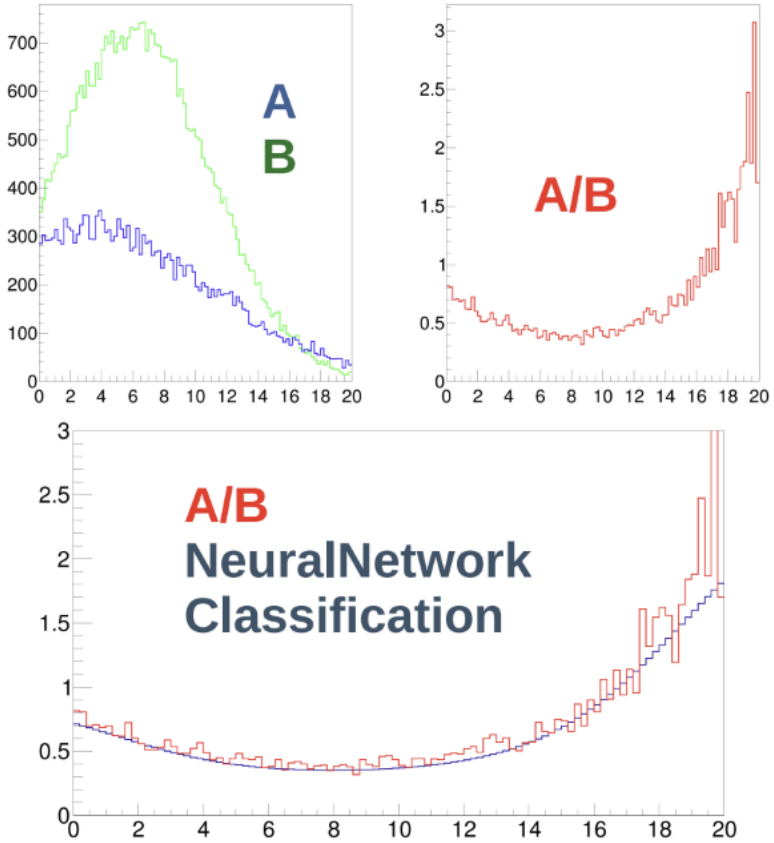
Often we require the ratio of two different distributions ⇒Density Ratio.

Examples include efficiency calculations, weighting…

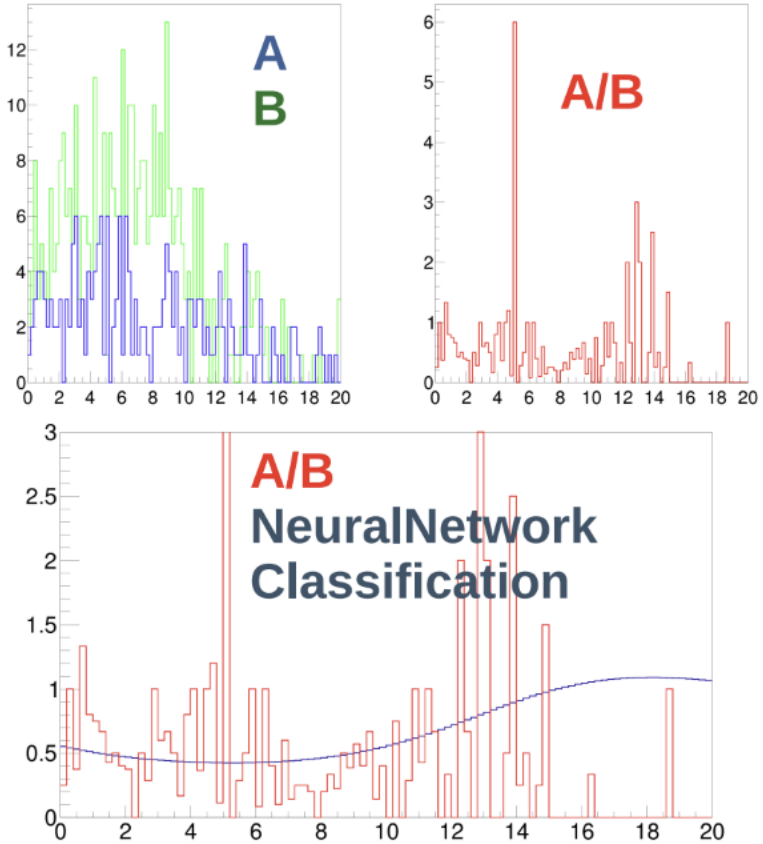
In 1D (2D) we may just use 2 histograms and create a 3rd which is their ratio.

This becomes complicated with more dimensions, or low statistics…

Alternatively we can learn density ratios in N dimensions using binary classification.




100x fewer events



# Multiparticle effects

Naïve expectation is that:

$$A(p_1, p_2, p_3) = A(p_1) \cdot A(p_2) \cdot A(p_3)$$

In reality we have multi particle effects:

$$A(p_1, p_2, p_3) = A(p_1|p_2, p_3) \cdot A(p_2|p_1, p_3) \cdot A(p_3|p_1, p_2)$$

Two solutions, first train acceptance particle per particle or event by event and a correction for multi particle effects:

$$A(p_1, p_2, p_3) \approx A'(p_1) \cdot A'(p_2) \cdot A'(p_3) \cdot C(p_1, p_2, p_3)$$

Or train with all final state particles to learn $A(p_1, p_2, p_3)$ directly.

First method exemplified here with $ep \rightarrow e'nK^+K^-\pi^+$