Adventures in OmniFold:

Multivariable Unfolding of Jet-Level Observables with STAR Data

Hannah Harrison-Smith

For the STAR Collaboration

University of Kentucky

RHIC/AGS Users Meeting

6/11/24

Supported in part by U.S. DEPARTMENT OF ENERGY

Motivation

Extracting Fragmentation Functions (FF)

- Fragmentation functions provide valuable information about the final-state parton in a collision.

- Vacuum FF typically extracted from
  1. e^+e^- *collisions*
  2. *SIDIS and* pp $(p\bar{p})$ *collisions*

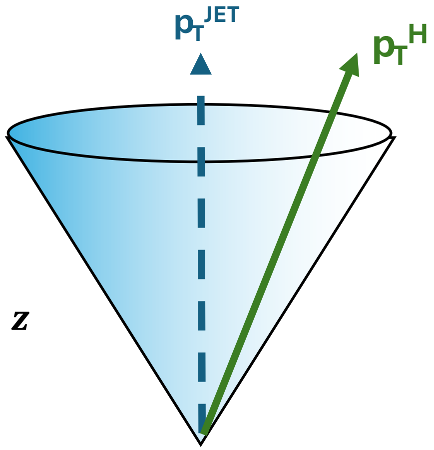
- We can complement e^+e^- by studying hadrons in jets in pp collisions.

Collinear

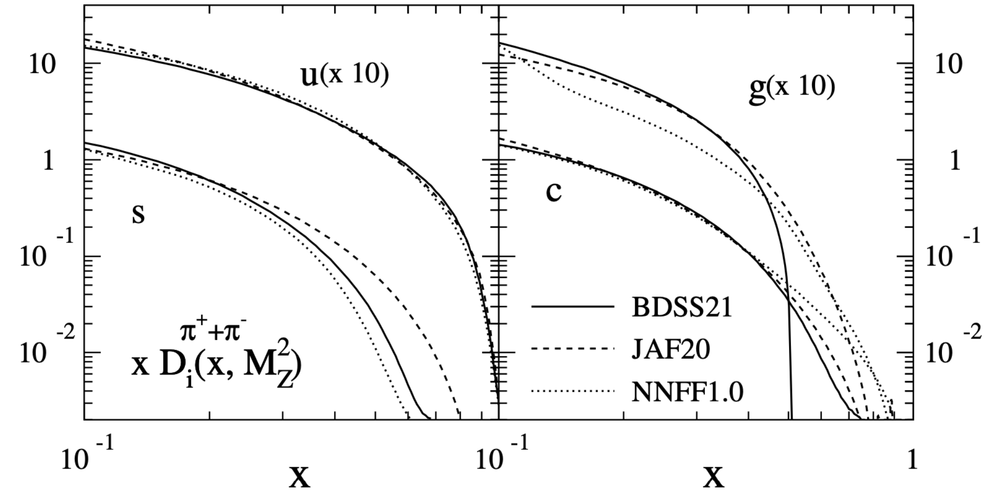
$$\boxed{\frac{d\sigma^{pp\to(jeth)+X}}{dp_T^{jet}\,d\eta^{jet}\,dz_h}}$$

Collinear
Hadron-in-Jet Cross Section
Phys. Rev. D 101, 079901 (2020)

- Collinear FFs are sensitive to both quark and gluon FF.
  - pp provides direct constraints on the gluon FF, especially at high x where SIDIS and e^+e^- are scarce.



$$z_h = \frac{p_T^{H}}{p_T^{JET}} \cong z$$



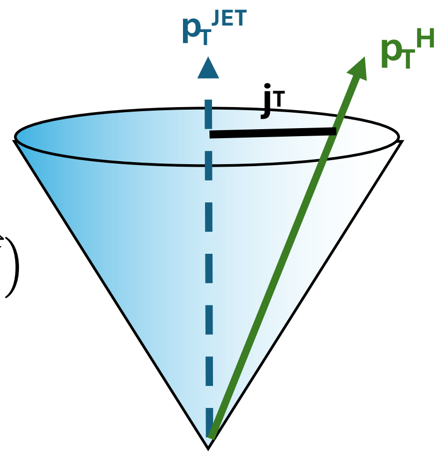
PGD. PTEP 2022 (2022), 083C01

Motivation

Extracting Fragmentation Functions (FF)

$$z_h = \frac{p_T{}^H}{p_T{}^{JET}} \cong z$$

$$j_T = \perp Proj_{\overrightarrow{p_T^{jet}}}\left(\overrightarrow{p_T^H}\right)$$



Transverse-Momentum-Dependent FF (TMD)

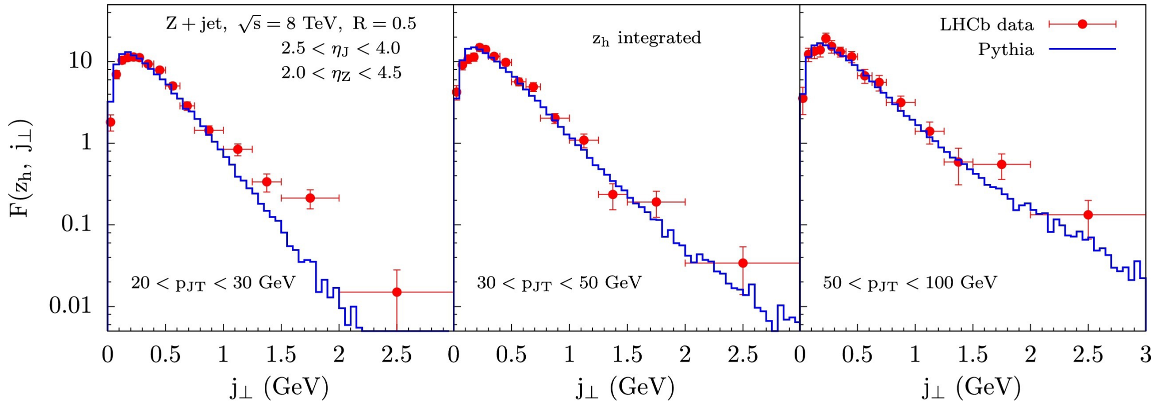
$$\boxed{F(z_h, j_T; pT, \eta, R) = \frac{d\sigma^{pp\to(jeth)+X}}{dp_T^{jet}\, d\eta^{jet}\, dz_h\, d^2 j_T} \Big/ \frac{d\sigma^{pp\to jet+X}}{dp_T^{jet}\, d\eta^{jet}}}$$

TMD FF

Kang, Z.-B., Liu, X., Ringer, F., Xing, H.
JHEP 1711 (2017) 068

- Unlike Collinear, takes into account transverse momentum component of fragmenting hadron.

- Looking at TMD FFs on different energy scales ($\sqrt{s}$) allows study of evolution effects.
  - pp, unlike SIDIS and e^+e^- , provides more direct access to gluon TMD FFs.
  - Compared to SIDIS, pp allows access to TMD FFs at higher Q^2.



Example TMDs: shown as functions of j_T, integrated over all z_h. Shown for 3 jet p_T ranges.
Kang, Z.-B., Lee, K., Terry, J., & Xing, H. Phys. Letters B, 798, 134978 (2019)

***GOAL**: extract charged-pion jet fragmentation functions in STAR Run15 proton-proton collisions at $\sqrt{s} = 200\ GeV$ (pp200).*

****** Work shown here is all in-progress, uncorrected, does not yet include all uncertainties/statistical errors. ****

Analysis

- TMD and Collinear FF extracted from Yield Ratios…

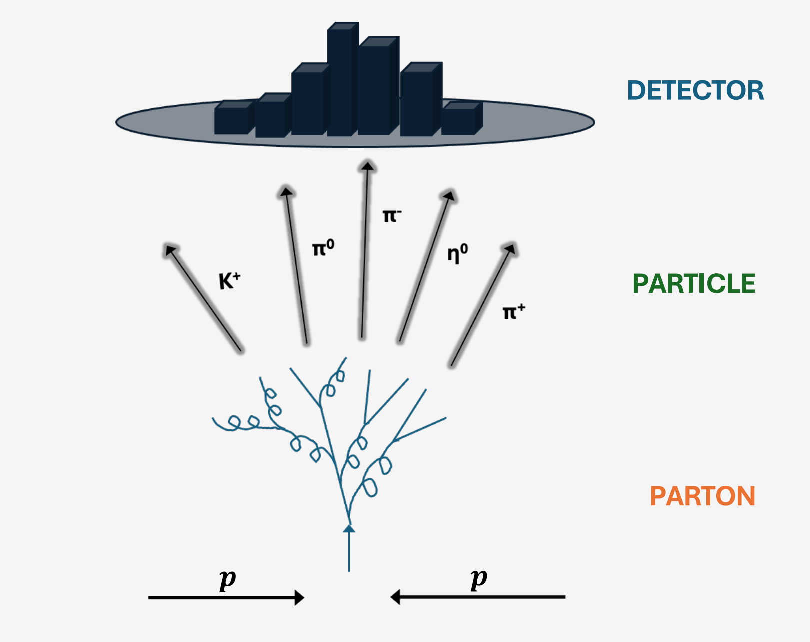
$$R_{collinear} = \frac{Njets^{\pi}}{Njets^{tot.}\,(z_h^{\pi}\ bin\ width)}$$

Collinear Yield Ratio
Kaufmann, T., Asmita M., Werner V. Phys. Rev. D 101, 079901 (2020)

$$R_{TMD} = \frac{Njets^{\pi}}{Njets^{tot.}\,(z_h^{\pi}\ bin\ width)2\pi\langle j_T^{\pi}\rangle(j_T^{\pi}\ bin)}$$

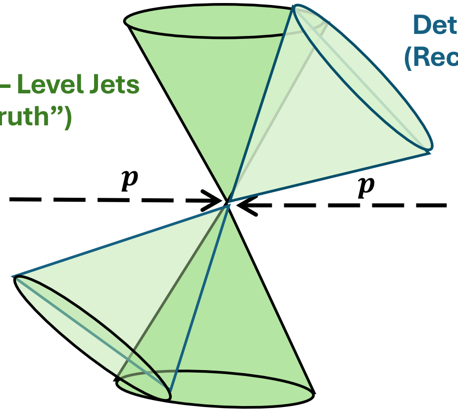
TMD Yield Ratio
Kang, Z.-B., Liu, X., Ringer, F., Xing, H. JHEP 1711 (2017) 068

- STAR Run15 pp200 Minbias triggers (SSDMB-5)



Steps

- Jet Reconstruction
  - Anti-k_T Jet-Finding Algorithm (*R= 0.6*).
  - Apply jet-level experimental cuts that isolate events of interest.
  - When required in simulation, match detector-level jets to closest particle-level jet and require jet axes to be separated by $\Delta R < 0.2$.
- Charged Pion Identification
  - Select charged pions via detector-level cuts (TPC, TOF, $n\sigma_{\pi}$).
- Underlying Event Correction
  - Apply 5GeV cut to reconstructed detector jet pT.
  - Correct jet pT for "underlying event" or peripheral events that did not contribute to the event of interest (Off-Axis Cone Method).



- **<u>Next Step:</u> Data corrections!**

Data Corrections

Several corrections to data that must be accounted for…

- **Bin Migration:**

  -- Accounts for bin migration due to detector effects.

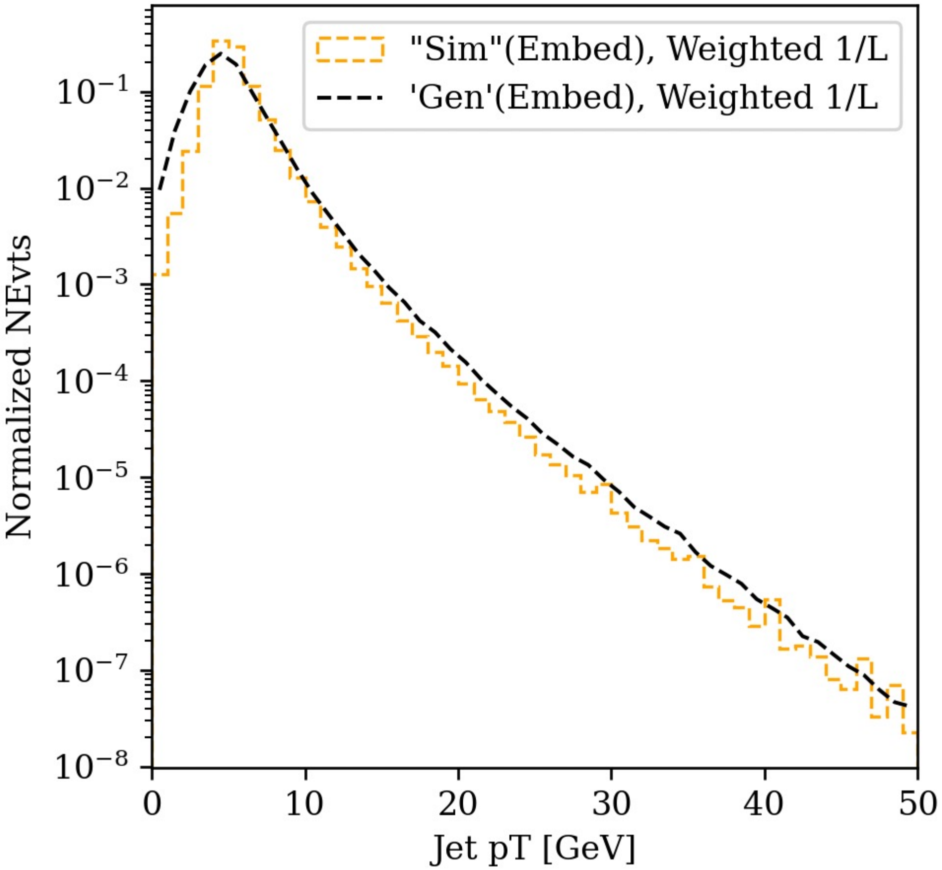
  - Need to "unfold"/account for bin migration in observables **pion z_h, j_T, and jet p_T.**

  -- Multi-observable unfolding using OmniFold **

  ** *Andreassen et al., PRL. 124, 182001 (2020)*

- **background / "fakes":** Detector-level ("reconstructed") jet and hadron events with *no* particle-level ("true") match.

- **detector efficiency:** Particle-level jet and hadron events ("true") with *no* detector-level ("reconstructed") match.



- Backgrounds and Bin Migration will be accounted for in OmniFold.
  - Background correction is applied by weighting data with factor w_{data}.
  - w_{data} are calculated prior to unfolding. They are fed into OmniFold as an input.

- Efficiency will be accounted for after applying OmniFold.
  - Not discussed in this talk, as it doesn't directly involve OmniFold.

Unfolding

OmniFold: A "New" Unfolding Method at STAR

- Many existing methods used for unfolding.
  - Iterative Bayesian Unfolding (IBU)
  - Bin-by-bin
  - Singular Value Decomposition (SVD)
- Drawbacks to existing methods:
  - Difficult to unfold multiple variables at the same time
  - Dependent on how data/embedding is binned (histograms)
- I employ the OmniFold method.

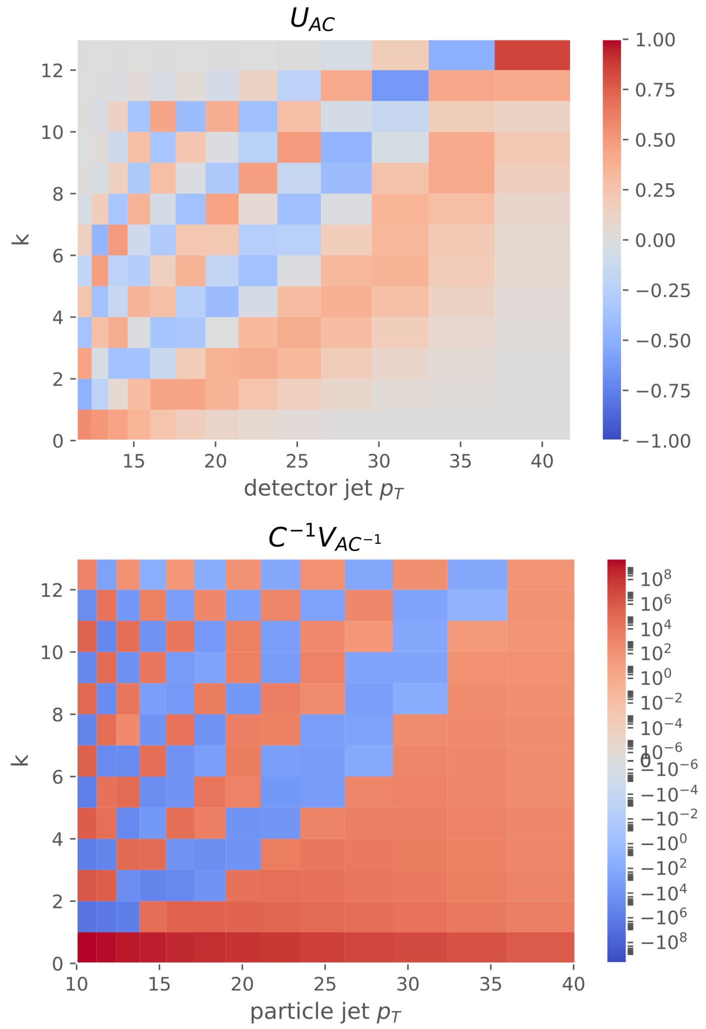
  Advantages
  - Unfolds all observables at once.
  - Isn't dependent on binning.

  Challenges
  - This method is being used in STAR, but not widely.
    - Currently being used on STAR jet substructure measurements to study parton showers.
  - Initially unsure if method would work for unfolding FFs.
    - Lots of closure tests!
- OmniFold algorithm can be further discussed in two categories…

  UniFold: Using OmiFold algorithm for single-variable unfolding

  MultiFold: Using OmniFold algorithm for multi-variable unfolding.

$$M = U\,S\,V^T$$



Example of SVD unfolding scheme.
K represents eigenvalues of S.
Dmitry Kalinkin, STAR

Unfolding Using OmniFold

Goal: ML obtains approximation for "truth" by a series of reweighting.

Inputs:

- Embedding

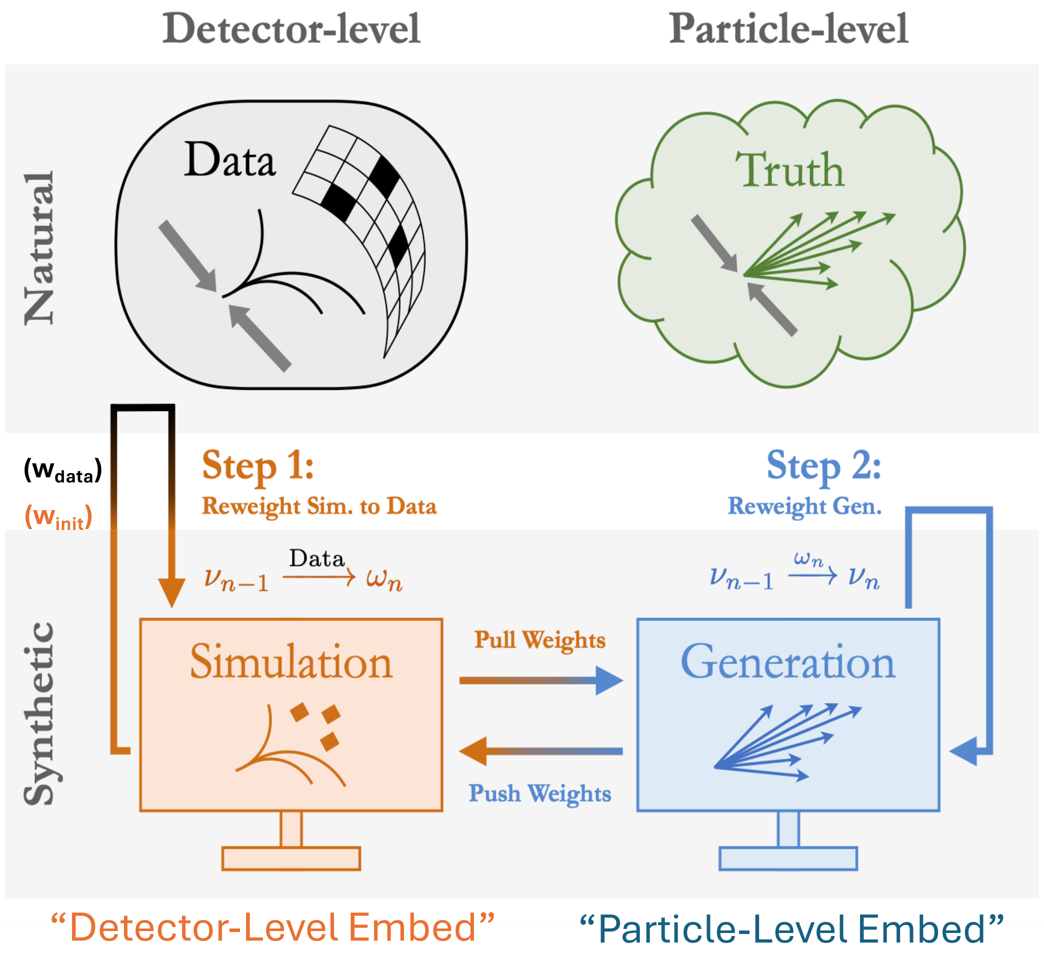
  Simulation "embedded" with sampling of random detector-level events.
  - *Detector-Level Embedding ("sim")*
  - *Particle-Level Embedding ("gen")*
- *Data (detector-level)*

- *Starting embedding weights ($w_{init} = {}^{1}/_{\mathcal{L}}$, inverse luminosity)*
- *Starting data weights (w_{data})*
  - Used to account for "backgrounds": Detector-level ("reconstructed") jets with *no* particle-level ("true") match.

$$w_{data\ i} = \frac{NDetEvt_{Matched\ i}}{NDetEvt_i}$$

Outputs:

- Weights for particle-level embedding (gen) which gives best approximation of truth. These I call w_{out}.
- *Reported result will be a Monte Carlo distribution: "gen" weighted with w_{out} ("truth")*



Andreassen et al., PRL. 124, 182001 (2020)

OmniFold: Closure Test

- Since truth is not known, it is difficult to know if OmniFold adequately models the given data/embed.
- There are also many parameters that affect how the ML algorithm fits embed → data.
- Giving OmniFold a "known truth" allows the algorithm to be studied.
  - In this sense, "giving ML the answer".
  - Allows investigation of different OmniFold optimization parameters.

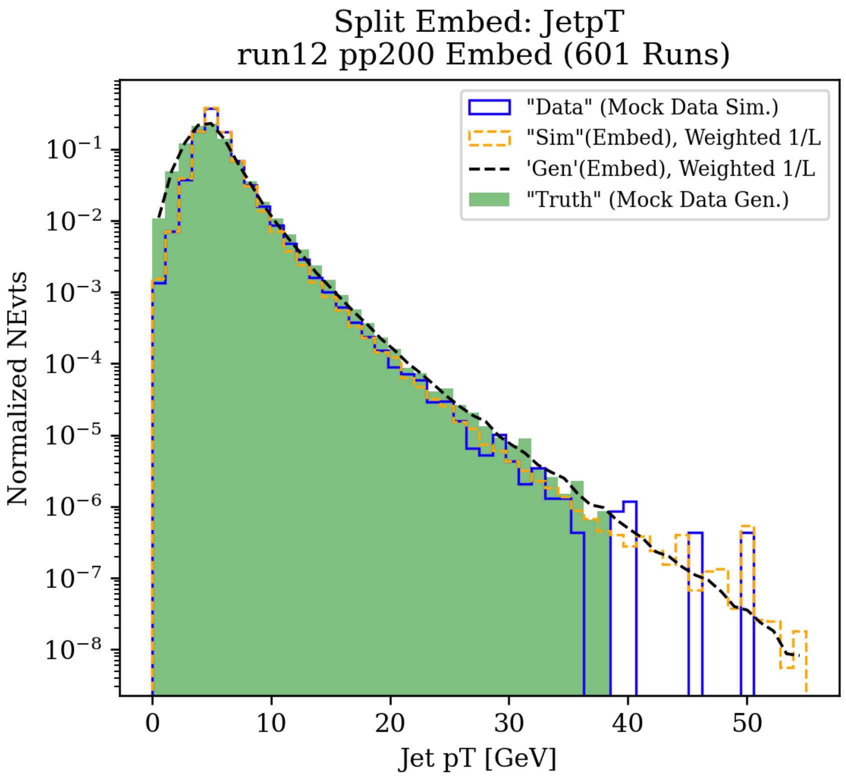
"Split Embedding"

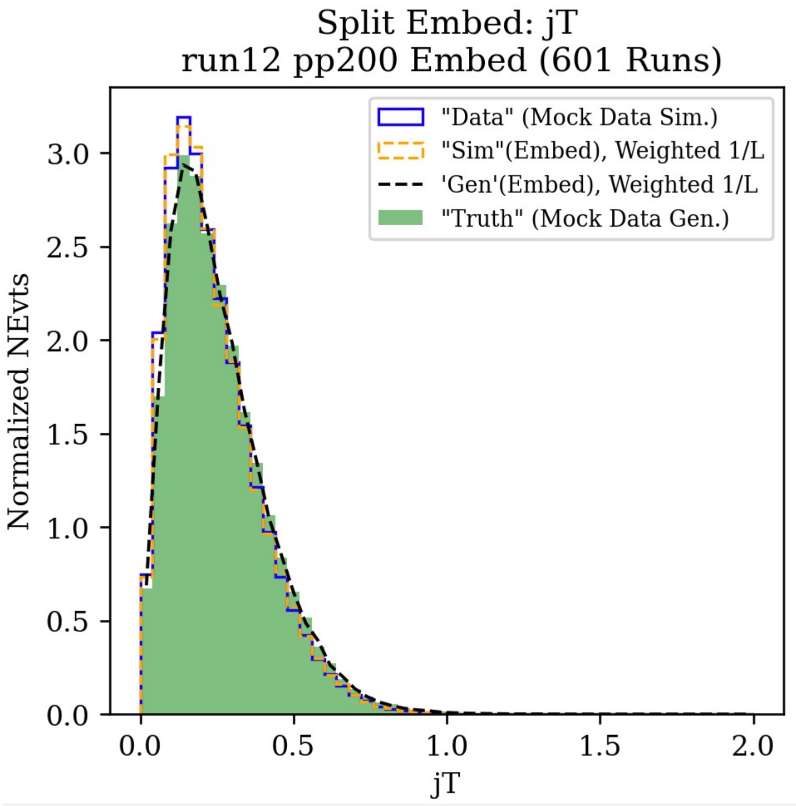
- Proof of closure is stronger if "embedding" (gen/sim) and "data" (data/truth) are truly independent data sets.
  - For this reason, mock data was generated.
  - *Half of embedding is used as "embedding" (300 runs), while the other half is used to generate mock data (301 runs).*

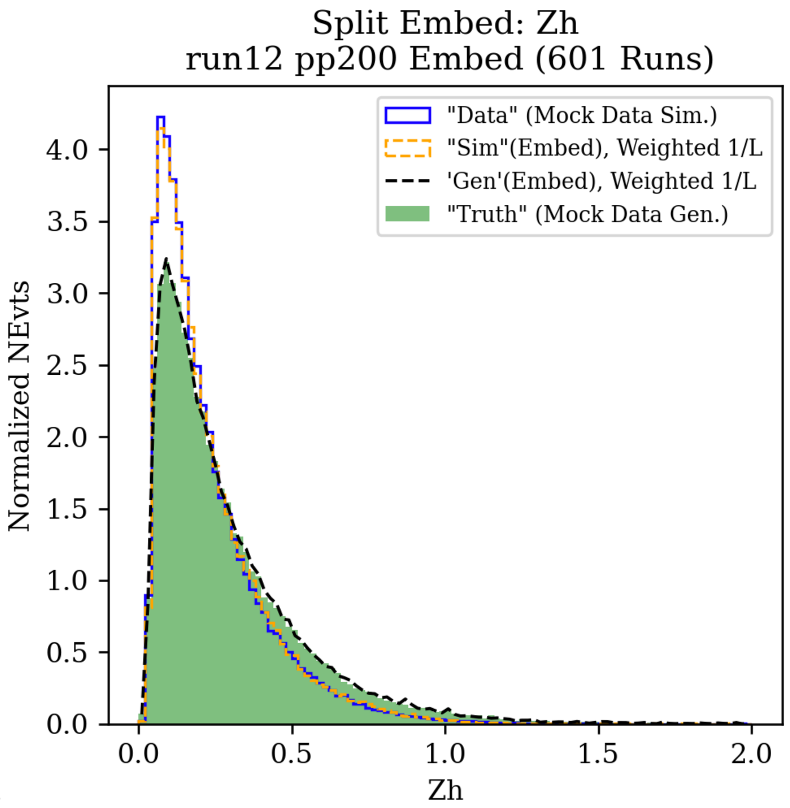
"Data" – *Detector-Level Mock Data*
Sim. – *Detector-Level Embedding, weighted by 1/L*
Gen. – *Particle-Level Embedding, weighted by 1/L*
"Truth" - *Particle-Level Mock Data*
MultiFold – *Particle-Level Embedding, weighted by W_{output}*
IBU – *Iterative Bayesian Unfolding (built-in to MultiFold)*

Mock Data

- Using run12 pp200 embedding sample (601 runs, ~3M events).
- Split embedding to create 2 independent data sets.
    - 1st Half of Embed Runlist: “Sim”, “Gen”
    - 2nd Half of Embed Runlist: ”Mock Data”, “Truth”
- **Will show these overlaid with OmniFold results on next few slides (red).**





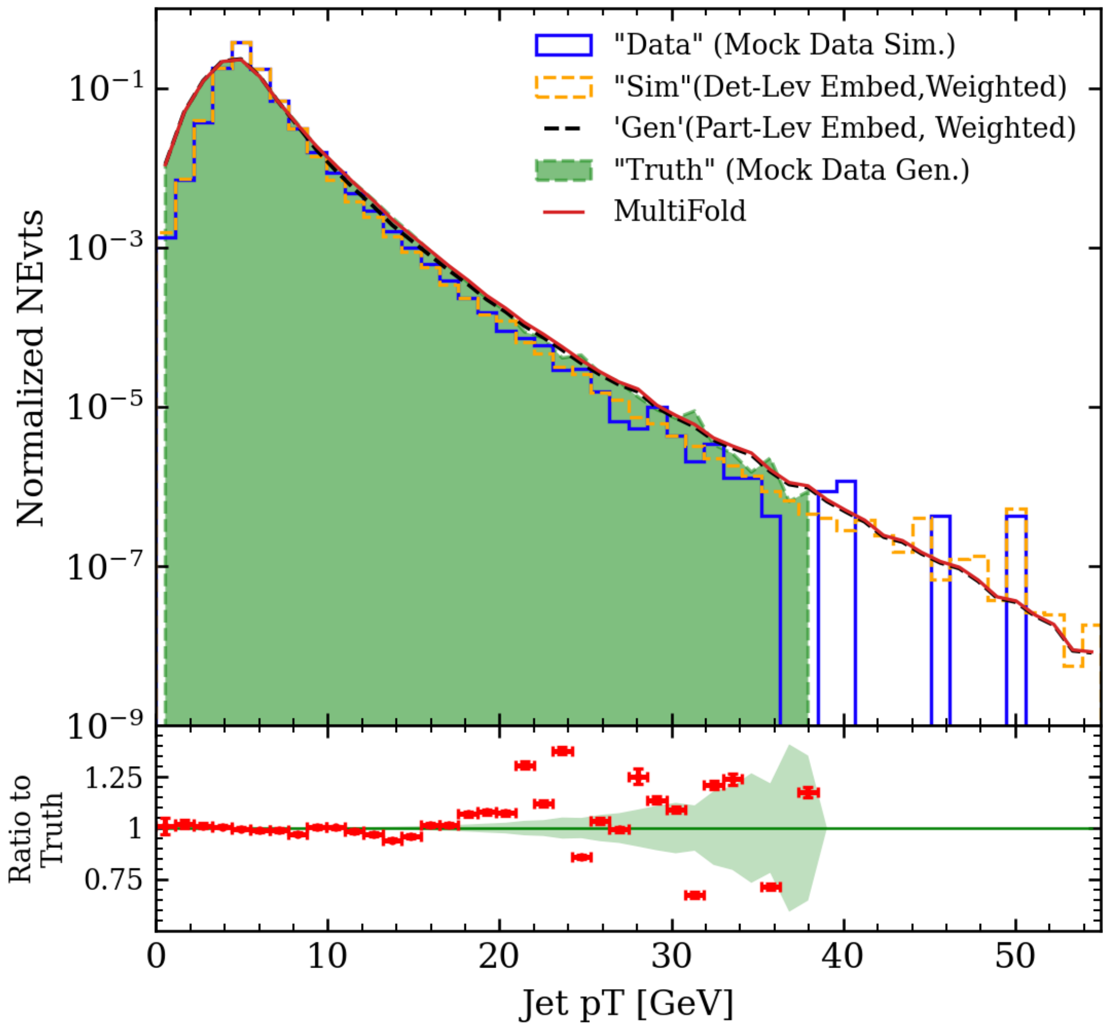


Closure Test

Closure Test: MultiFold Jet pT

- Now using run12 pp200 embedding sample (601 runs, ~3M events).
- Split embedding to create 2 independent data sets.
  - 1st Half of Embed Runlist: “Sim”, “Gen”
  - 2nd Half of Embed Runlist: ”Mock Data”, “Truth”

- MultiFold Closure Test was successful!



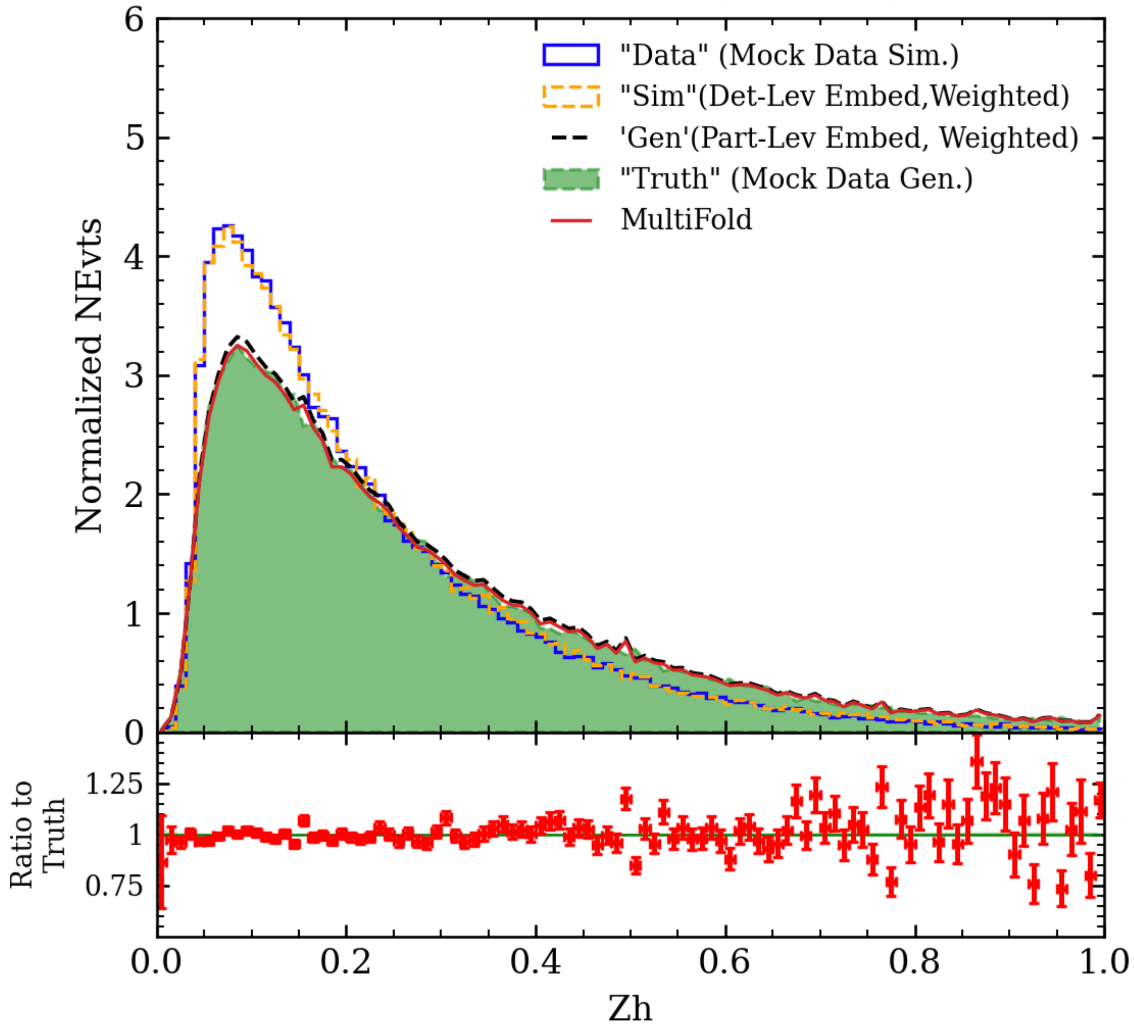
”Data” – *Detector-Level Mock Data*
Sim. – *Detector-Level Embedding, weighted by 1/L*
Gen. – *Particle-Level Embedding, weighted by 1/L*
”Truth”- *Particle-Level Mock Data*
MultiFold – *Particle-Level Embedding, weighted by W_{output}*

Closure Test

Closure Test: MultiFold Zh

- Now using run12 pp200 embedding sample (601 runs, ~3M events).
- Split embedding to create 2 independent data sets.
  - 1st Half of Embed Runlist: "Sim", "Gen"
  - 2nd Half of Embed Runlist: "Mock Data", "Truth"

- MultiFold Closure Test was successful!



"Data" – *Detector-Level Mock Data*

Sim. – *Detector-Level Embedding, weighted by 1/L*

Gen. – *Particle-Level Embedding, weighted by 1/L*

"Truth"- *Particle-Level Mock Data*

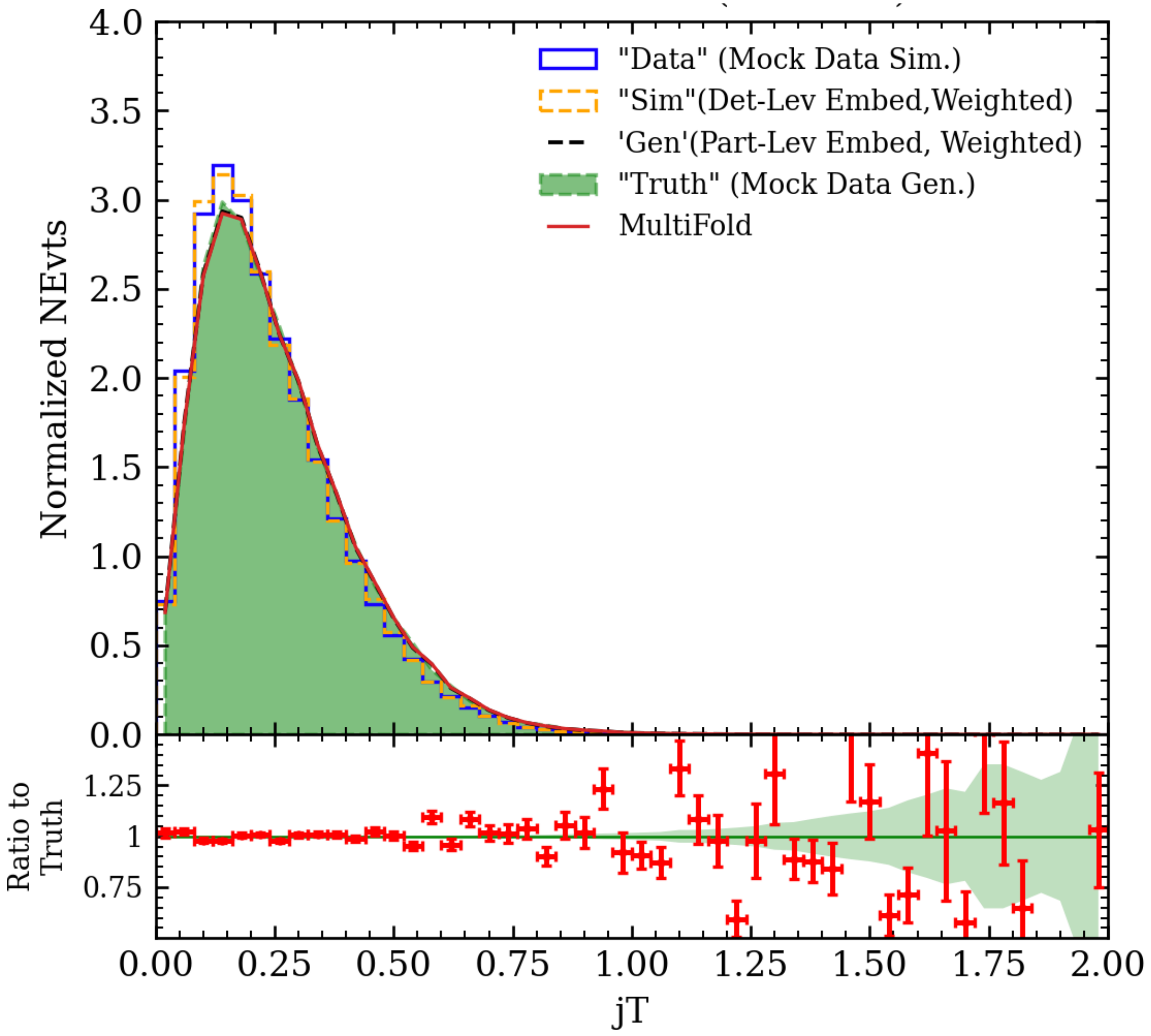
MultiFold – *Particle-Level Embedding, weighted by W_{output}*

Closure Test

Closure Test: MultiFold jT

- Now using run12 pp200 embedding sample (601 runs, ~3M events).
- Split embedding and sampled it to create 2 independent data sets.
    - 1st Half of Embed Runlist: “Sim”, “Gen”
    - 2nd Half of Embed Runlist: ”Mock Data”, “Truth”

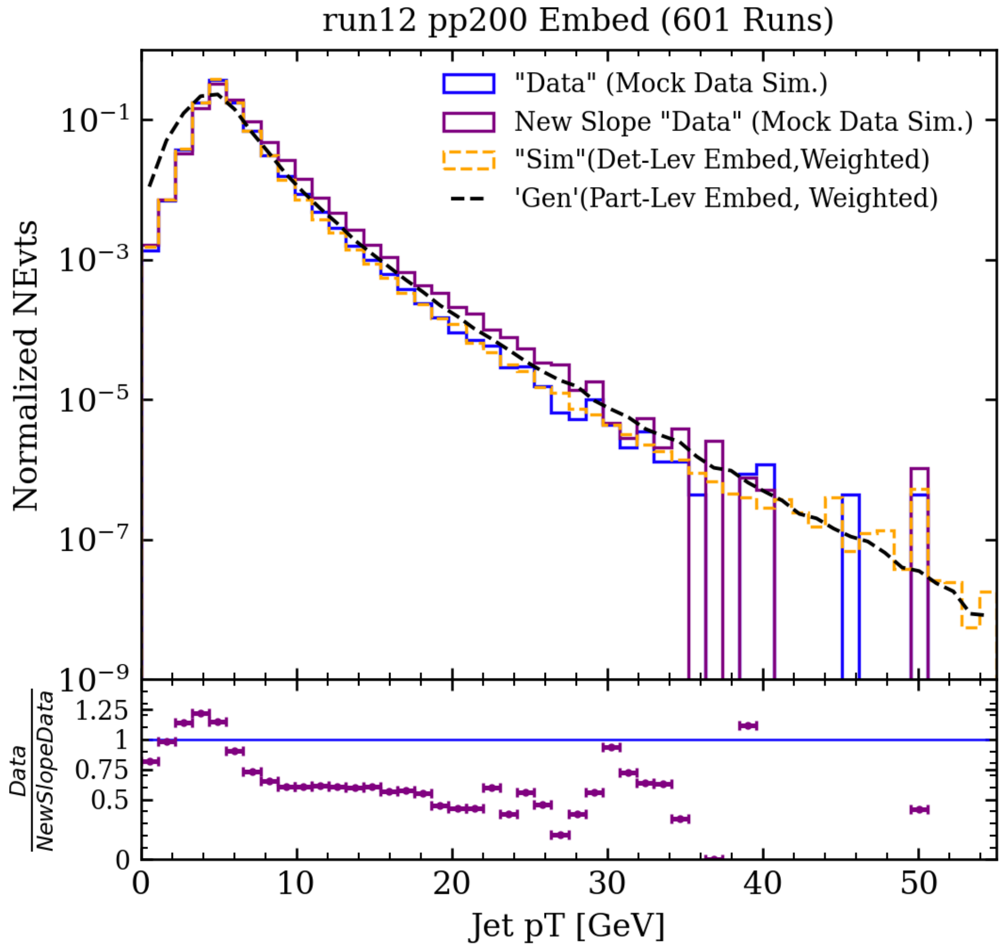
- MultiFold Closure Test was successful!



”Data” – *Detector-Level Mock Data*
Sim. – *Detector-Level Embedding, weighted by 1/L*
Gen. – *Particle-Level Embedding, weighted by 1/L*
”Truth”- *Particle-Level Mock Data*
MultiFold – *Particle-Level Embedding, weighted by W_{output}*

Second Closure Test

- First closure test assumed data and simulation ("Data" and "Sim") have same shape. What if this isn't the case?
- Performed second closure test where data/simulation have different slopes.
  - A second "new slope" mock data set was constructed by weighting original mock data to give it some shape.

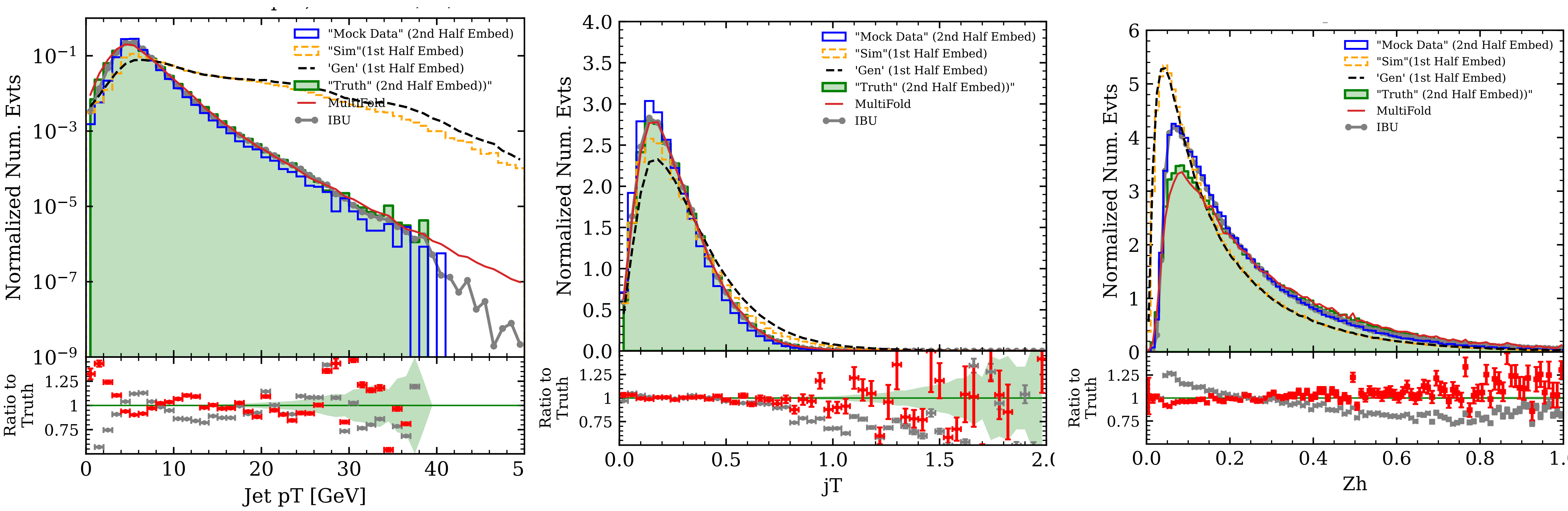


Second Closure Test

"Reweighted" Closure Test

- For Jet pT, OmniFold and IBU both break down for pT < 5GeV.
- For both Jet p_T > 5GeV, j_T, and Z_h, OmniFold improves.
- *This raises the question: How different will simulation and data actually be?*

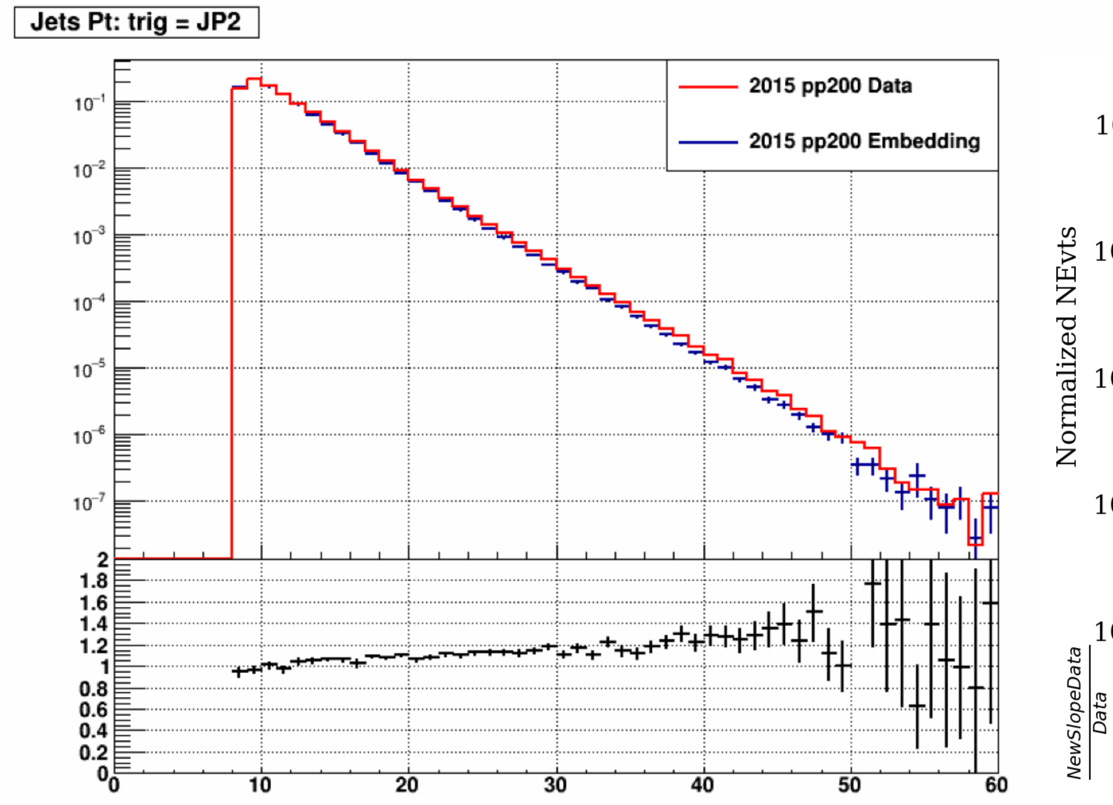
"Data" – "new" *Run15 pp200- like mock data*
Sim. – *Detector-Level Embedding, weighted by 1/L*
Gen. – *Particle-Level Embedding, weighted by 1/L*
"Truth"- *Particle-Level Mock Data*
MultiFold – *Particle-Level Embedding, weighted by W_{output}*
IBU– *Iterative Bayesian Unfolding Method*



Data/Sim Comparison

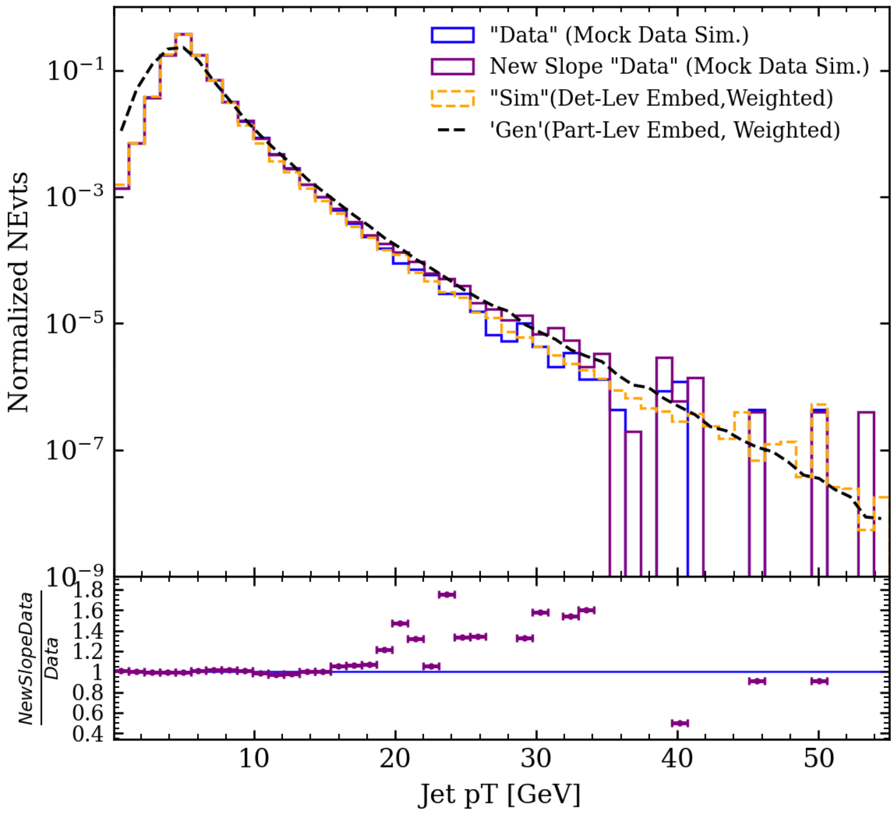
A New Run15 Mock Data Model

- How different will data/simulation (embedding) actually be?
- With a few changes to my previous Mock Data Model, a new Run15 Mock Data Model (right) can be constructed to more closely model an existing STAR model of this (left).



Ratio of 2015 data to 2015 embedding

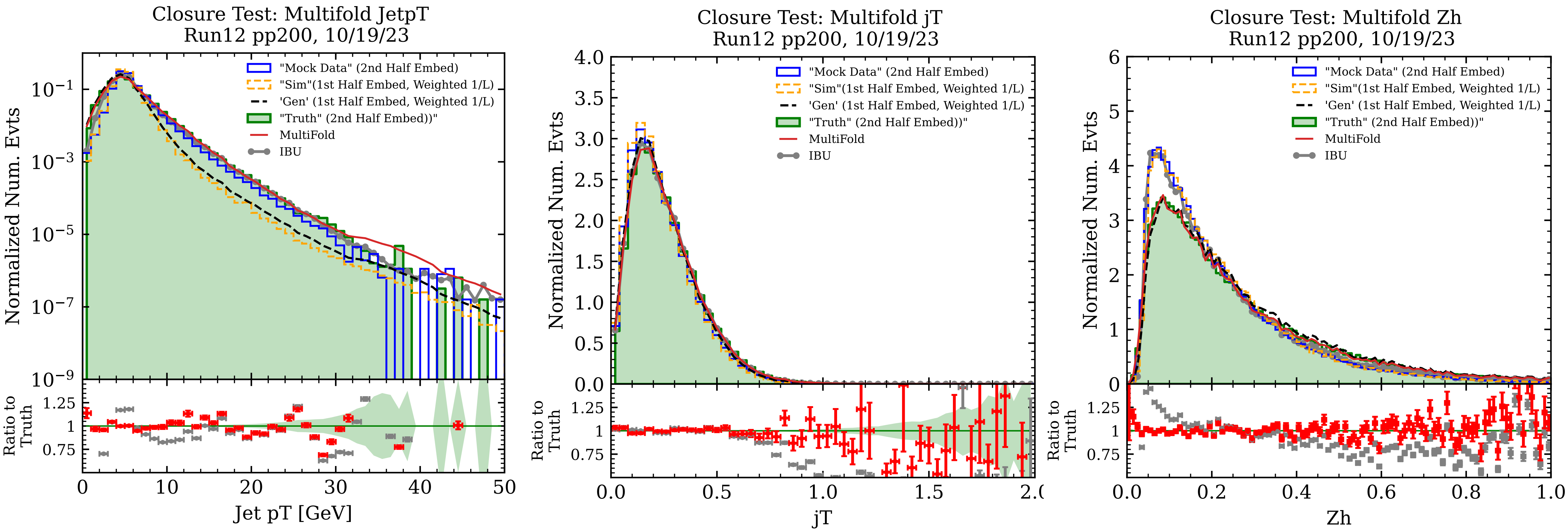
Ting Lin, STAR



Data/Sim Comparison

Run15 Mock Data Model: Closure Test

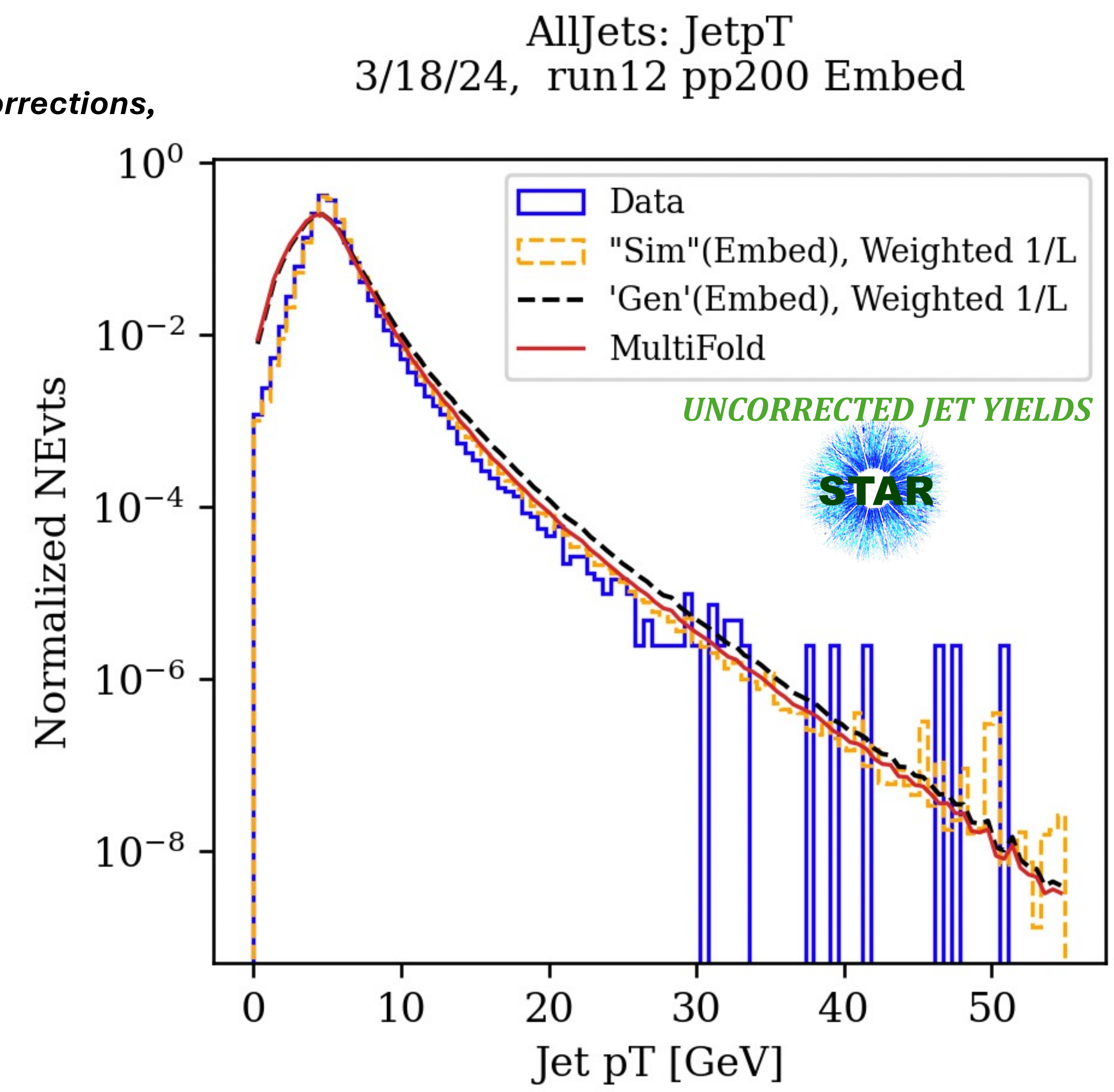
- This new distribution unfolds much better under closure test, in its ability to reconstruct the truth.



Unfolding Data: $Njets_{tot.}(p_T)$

****** Unfolding is a work-in-progress and doesn't yet include all corrections, uncertainties, and errors ******

- This jet yield is proportional to the denominator of my FF yields.

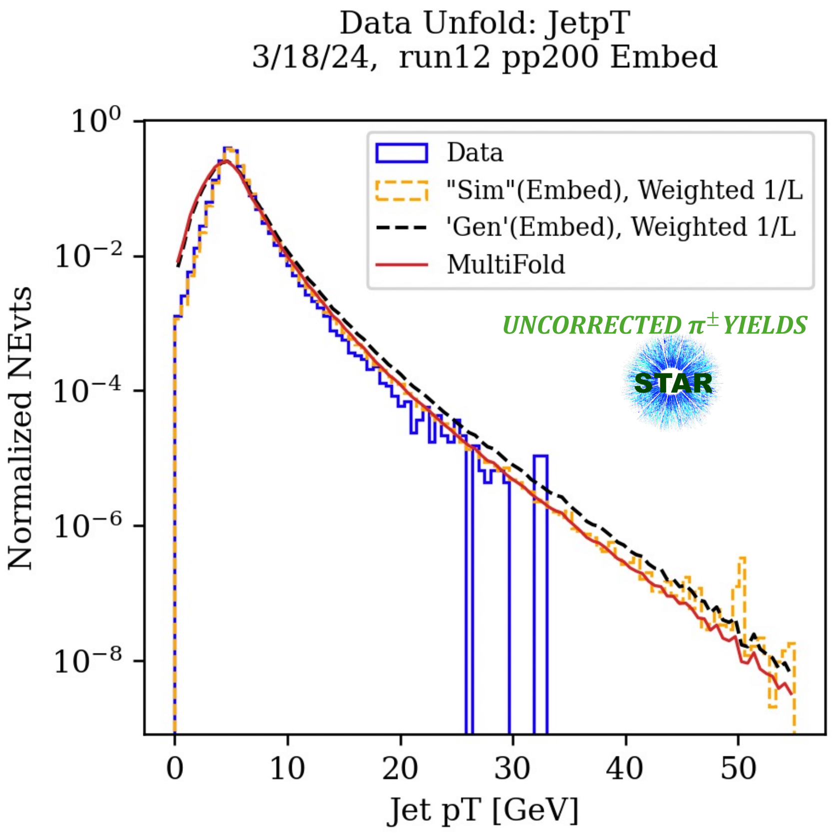


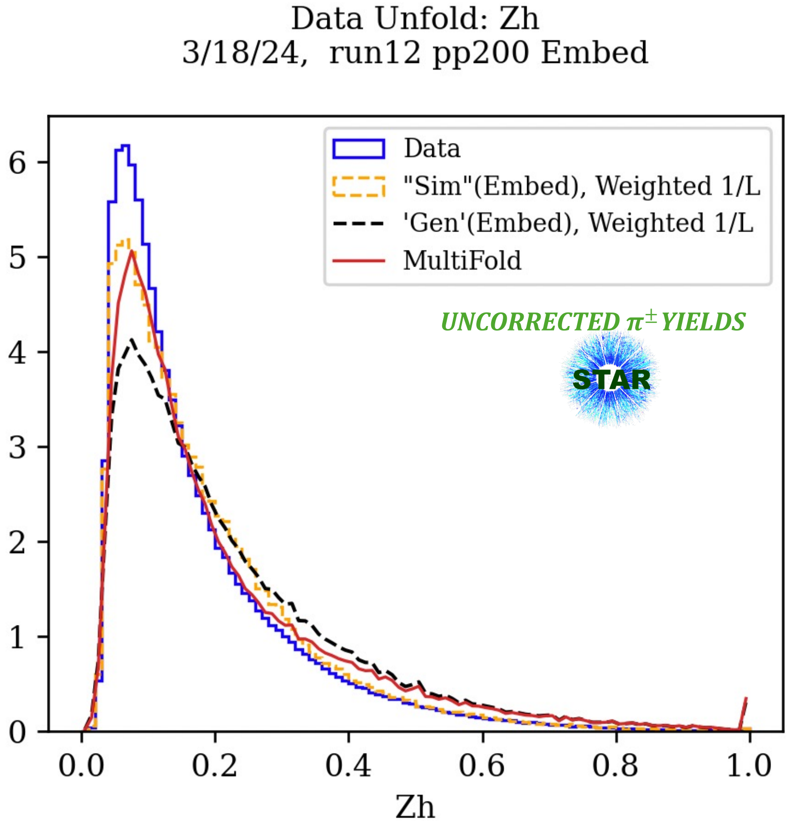
"Data" – *Detector-Level run15 Data*
Sim. – *Detector-Level run12 Embedding, weighted by 1/L*
Gen. – *Particle-Level run12 Embedding, weighted by 1/L*
MultiFold – *Particle-Level Embedding, weighted by W_{output}*
IBU – *Iterative Bayesian Unfolding (built-in to MultiFold)*

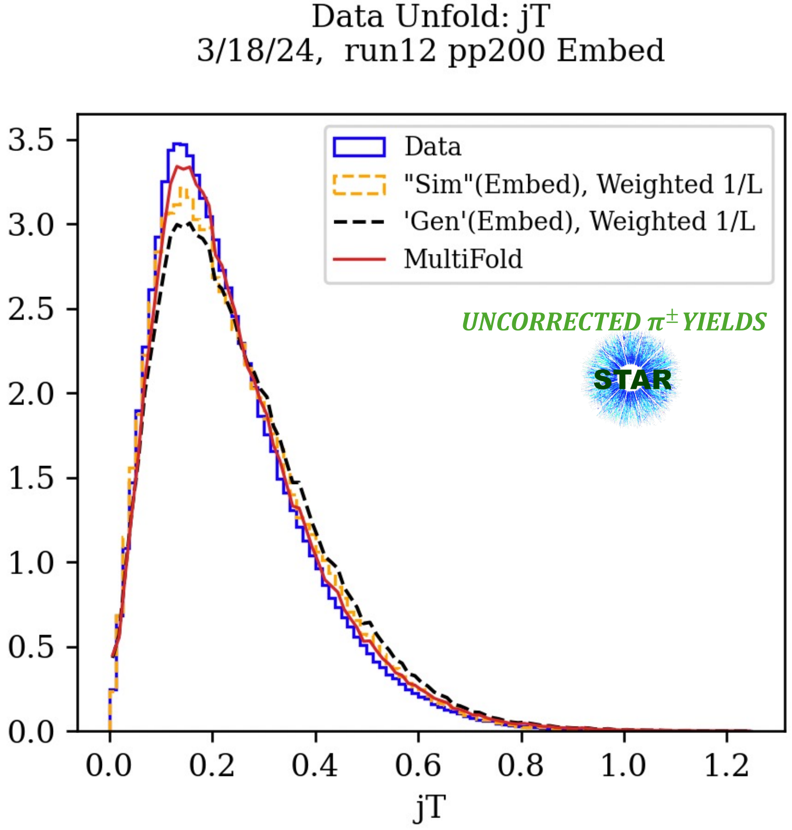
Unfolding: $Njets_{\pi}\ (p_T, z_h^{\pi})$

- ****** Unfolding is a work-in-progress and doesn't yet include all corrections, uncertainties, and errors ******
- Thes pion yields represent the the numerator of my FF yields.

"Data" – *Detector-Level run15 Data*
Sim. – *Detector-Level run12 Embedding, weighted by 1/L*
Gen. – *Particle-Level run12 Embedding, weighted by 1/L*
MultiFold – *Particle-Level Embedding, weighted by W_{output}*



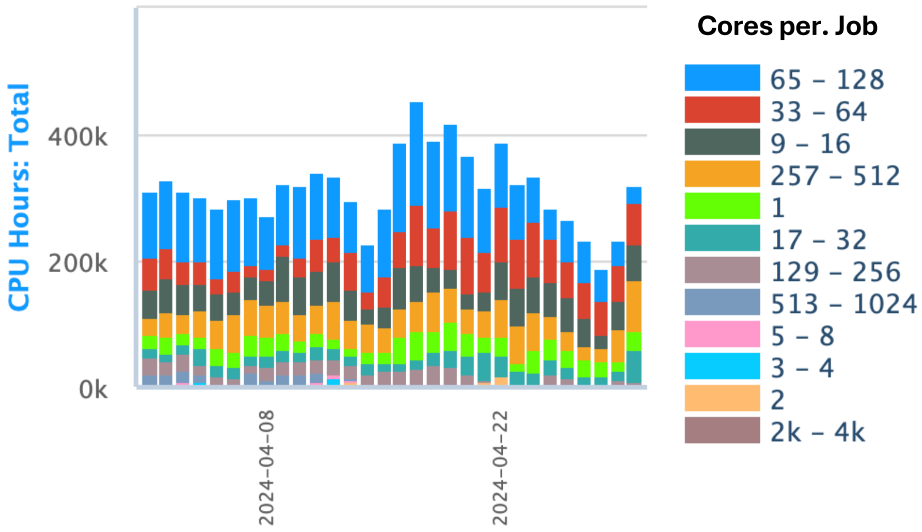




UKentucky LCC

- Unfolding was done using the University of Kentucky Center for Computational Sciences **Lipscomb Computing Cluster (LCC).**
    - LCC came online beginning in Fall 2019.
    - Intel CPUs, batch processing.
    - The LCC is used by researchers from 75+ research groups across the universities.
        - Primarily physics, engineering, biology, chemistry.
    - On average, a single MultiFold closure test takes:
        - **5-7 hrs** to run the full training algorithm.
        - **1 node**
        - **1 CPU**

UK LCC By the Numbers	
Users	*204*
Computing Nodes	*198*
Cores per Node	*32 - 48*
Average Run Time (per job)	*14 hrs*
Average Wait Time (per job)	*0.47 hrs*



Center for
Computational Sciences

Conclusions

- Simplest proof-of-concept OmniFold closure tests have been passed.

- OmniFold presents as good an unfolding result as IBU
  - Disagreement between data/embed and its effect on OmniFold has been explored.

- OmniFold performance depends on
  - Magnitude of disagreement
  - How well simulation replicates experimental conditions.

- OmniFold is shown to be a viable option for multi-dimensional unfolding of STAR data, specifically with applications to jet fragmentation function analysis.

Next Steps and In-Progress

- Finish correcting for backgrounds and inefficiencies.
- Apply systematic and statistical uncertainties to unfolded FF results.

Backup

Correcting for Fakes: W_{data}

- Fakes are accounted for by applying a weight (w_{data}) to each data point in OmniFold.
- These w_{data} are an input to OmniFold.

How do we know what amount of data were "fakes"?

- W_{data} is computed from what I call embedding "matched rate".
  - How many events in embedding had a detector jet/particle jet match *and* a pion-level match?
  - **In other words, what fraction of events did the detector see that came from "real" events?**

$$\boxed{Matched\ Rate_{(bin)} = \frac{NDetJet_{Matched\ (bin)}}{NDetJet_{Tot\ (bin)}}}$$

- *NDetJets$_{Tot}$* includes
  - Events from detector jets that didn't match to a particle jet ("no jet match").
  - Events where there was a jet match, but pions within the jet didn't match ("no pion match").

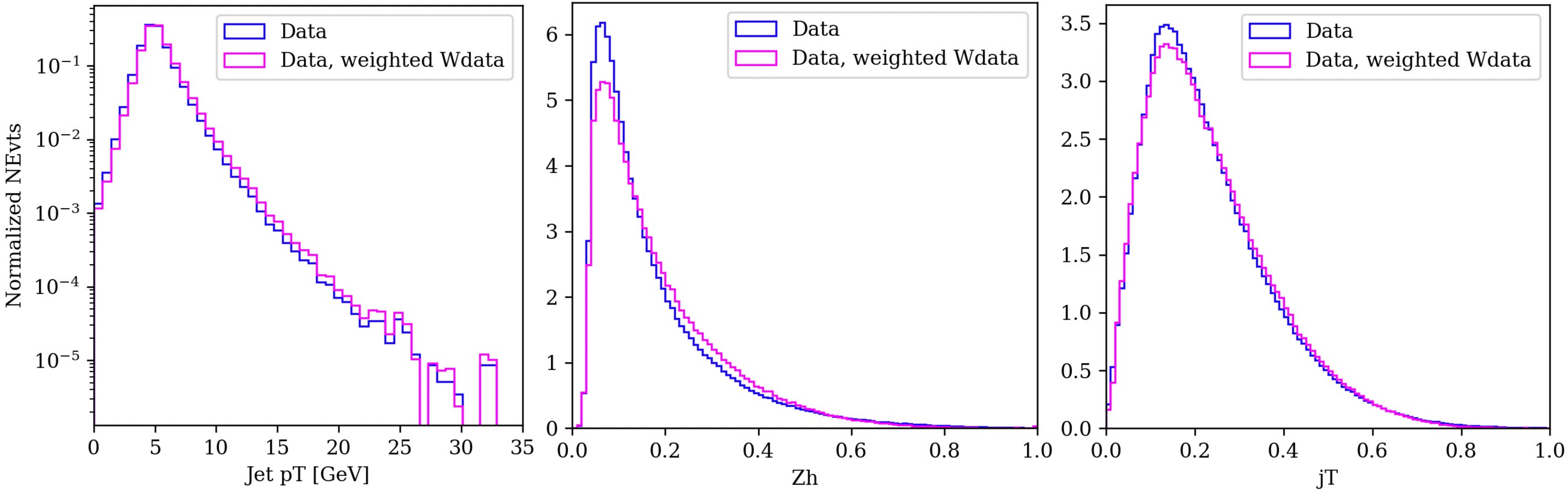
- Wdata is obtained by sampling embedding "matched rate" at each *data* point.

Correcting for Fakes: W_{data}

- Fakes are accounted for by weighting by data with some weight (w_{data}) when inputting to unfolding.
- In general, w_{data} is computed by sampling histograms of matched detector jets and total detector jets…

$$w_{data\ (event\ i)} = \frac{NDetJet_{Matched\ (event\ i)}}{NDetJet_{(event\ i)}}$$

- ***Application of this is still a work-in-progress.***



Analysis and Cuts

Steps

- Jet Reconstruction
  - Anti-k_T Jet-Finding Algorithm (*R= 0.6)*.
  - Apply jet-level experimental cuts that isolate events of interest.
  - When required in simulation, match detector-level jets to closest particle-level jet and require jet axes to be separated by $\Delta R < 0.2$.
- Charged Pion Identification
  - Apply hadronic cuts that further isolate events with charged pions.
- Underlying Event Correction
  - Apply 5GeV cut to reconstructed detector jet pT.
  - Correct jet pT for "underlying event" or peripheral events that did not contribute to the event of interest (Off-Axis Cone Method).

pp200 **Data Cut Summary**	
Jet-Level	***Pion-Level***
$\lvert Ver\ Z \rvert$< 30 R_T^{jet}< 0.95 $\lvert \eta_{jet} \rvert$ < 1 $\lvert \eta_{jet\ det} \rvert$< 0.8 $Sum\ Track\ p_T$ > 0.5	-1 < $n\sigma(\pi)_{TPC}$ < 2.5 -4 < $n\sigma(\pi)_{TOF}$ < 4 $Hits\ Fit(TPC)$ > 20

Reweighting with Machine Learning

- OmniFold is a Python-based machine-learning unfolding method, which trains a neural network.
  - Keras, TensorFlow, EnergyFlow
- Trains neural network *f(x)* using Categorical Cross-Entropy Loss Function, which has known result

$$\boxed{w(x) \approx \frac{f(x)}{1 - f(x)} \approx \frac{p_0(x)}{p_1(x)}}$$

(Andreassen and Nachman PRD 101, 091901 (2020))

- $p_0(x)$ and $p_1(x)$ give probability densities for embedding and data.
- $w(x)$ is the weighting parameter used to train one data/sim. set to another. This is what Keras obtains.

- Input:
  - **Winit:** initial values for Keras to use for w(x).
  - **Data:** detector-level
  - **Embedding:** Pairs of matched detector-level and particle-level jets
    - Select best-match jets by requiring R<=0.2

Machine Learning Params

- There are several main parameters that tell OmniFold how to process the given data sets:

  - **Batch Size (1,000)**
    - Data/embed are broken into ”batches” to be analyzed
    - Batch size tells *how many data points* should be in one batch.
  - **Iterations (4)**
    - Number of times a batch is passed through the algo.
    - After each *iteration,* the ML algo. outputs a set of approximations for w(x).
  - **Model Layer Size ([100, 100, 100])**
    - Size of the neural network to “train”.
  - **Seed (ON, Seed=43)**
    - Some ML algorithms make use of random number generators. Setting a “seed” assigns these random number generators the same value each time, to minimize fluctuations in w(x).

- Setting these parameters correctly for the given data sets are key to optimizing OmniFold, and having it work effectively.