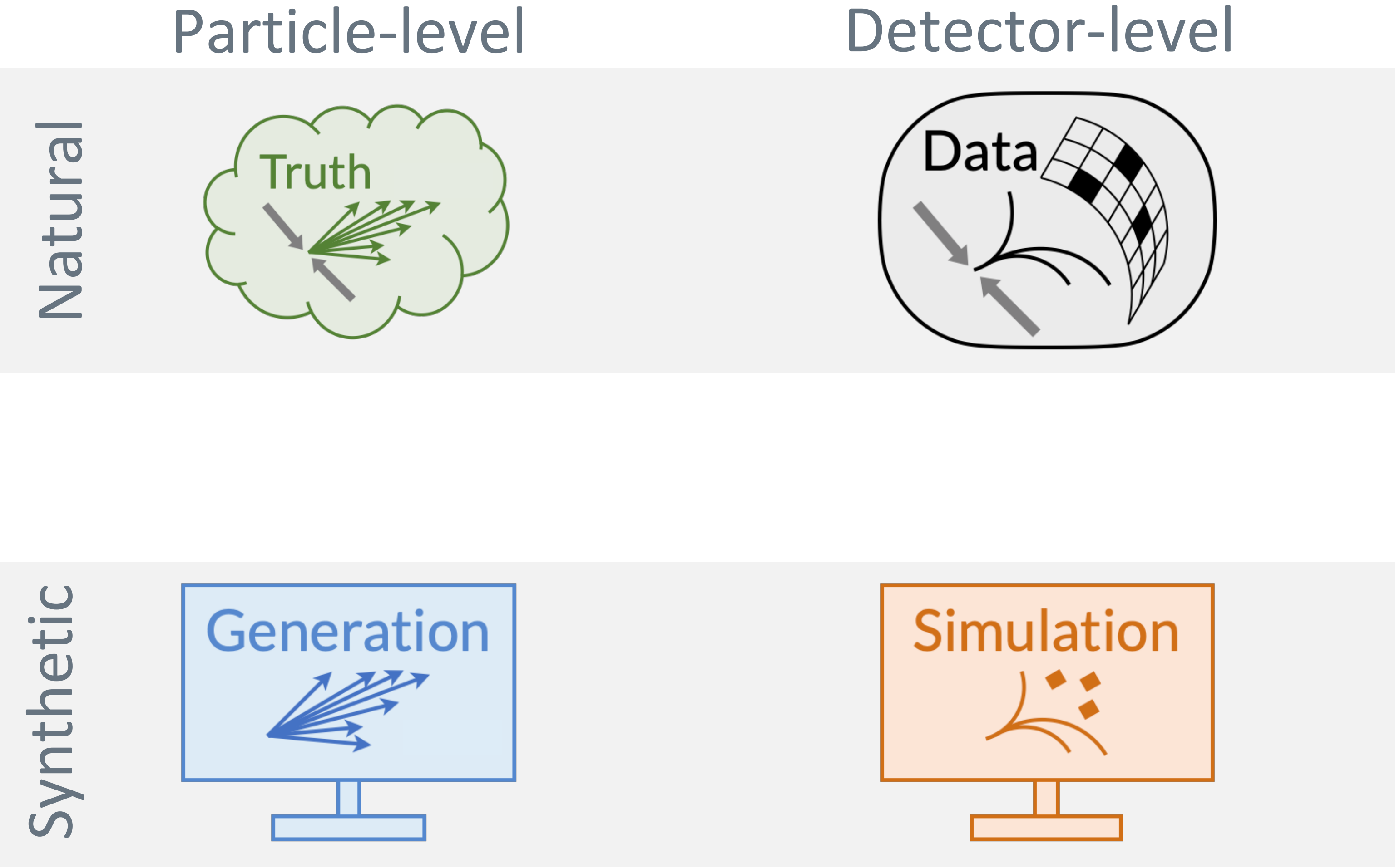
Scaffolding Simulations with Deep Learning for High-dimensional Deconvolution

Adi Suresh (UC Berkeley)

In collaboration with **Anders Andreassen** (Google), **Patrick Komiske** (MIT), **Benjamin Nachman** (LBNL), **Eric Metodiev** (MIT), and **Jesse Thaler** (MIT)

2021 CMS Machine Learning Town Hall, July 26-30



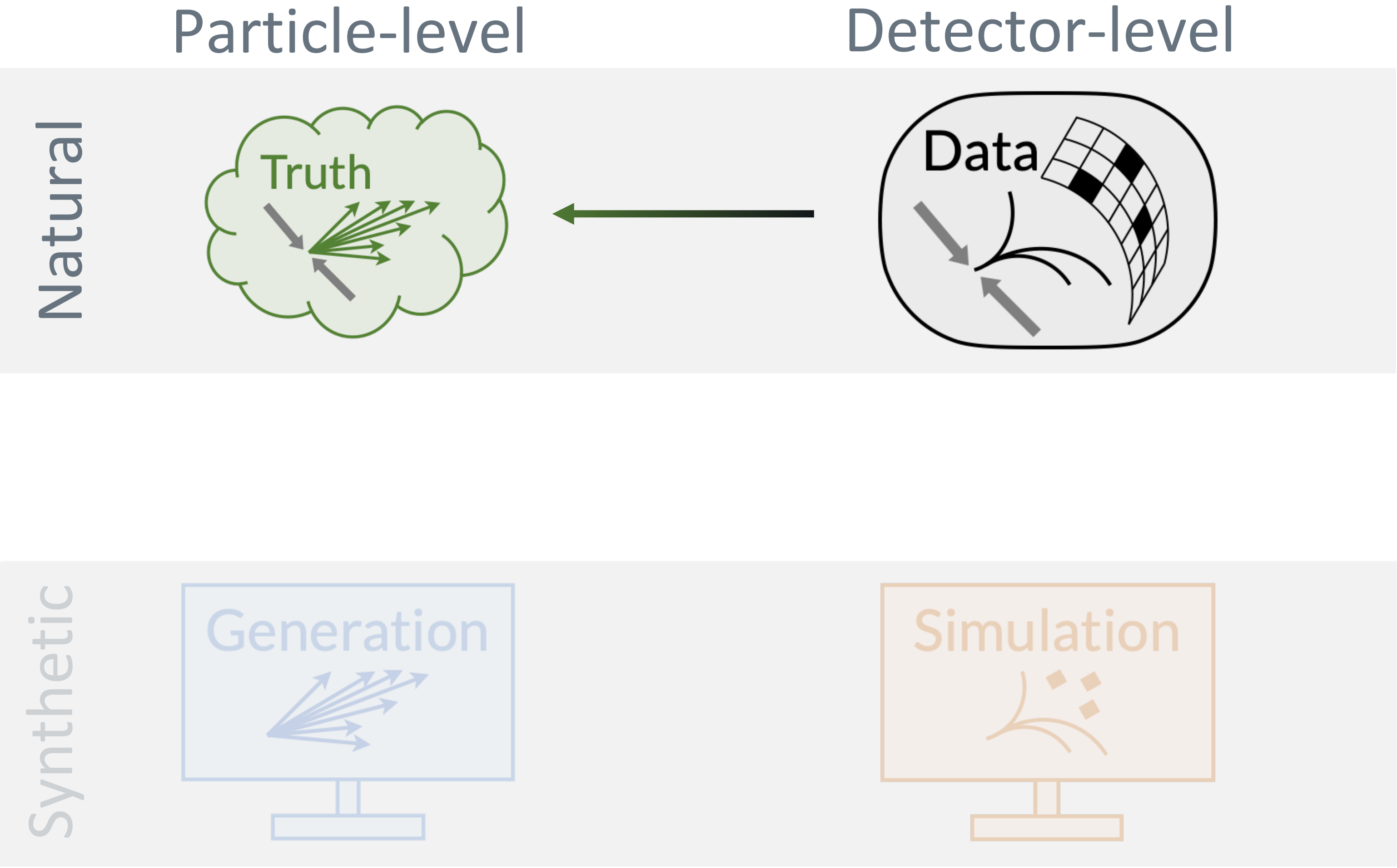
Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

Nomenclature

Deconvolution (AKA Unfolding)

- Goal: take **data** (at detector-level) and **infer truth** (particle-level)
  - i.e. remove detector effects
- Key for comparing **theoretical predictions** to what we **observe** in the collider and for comparing between experiments



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

The Goal

Current Unfolding Drawbacks

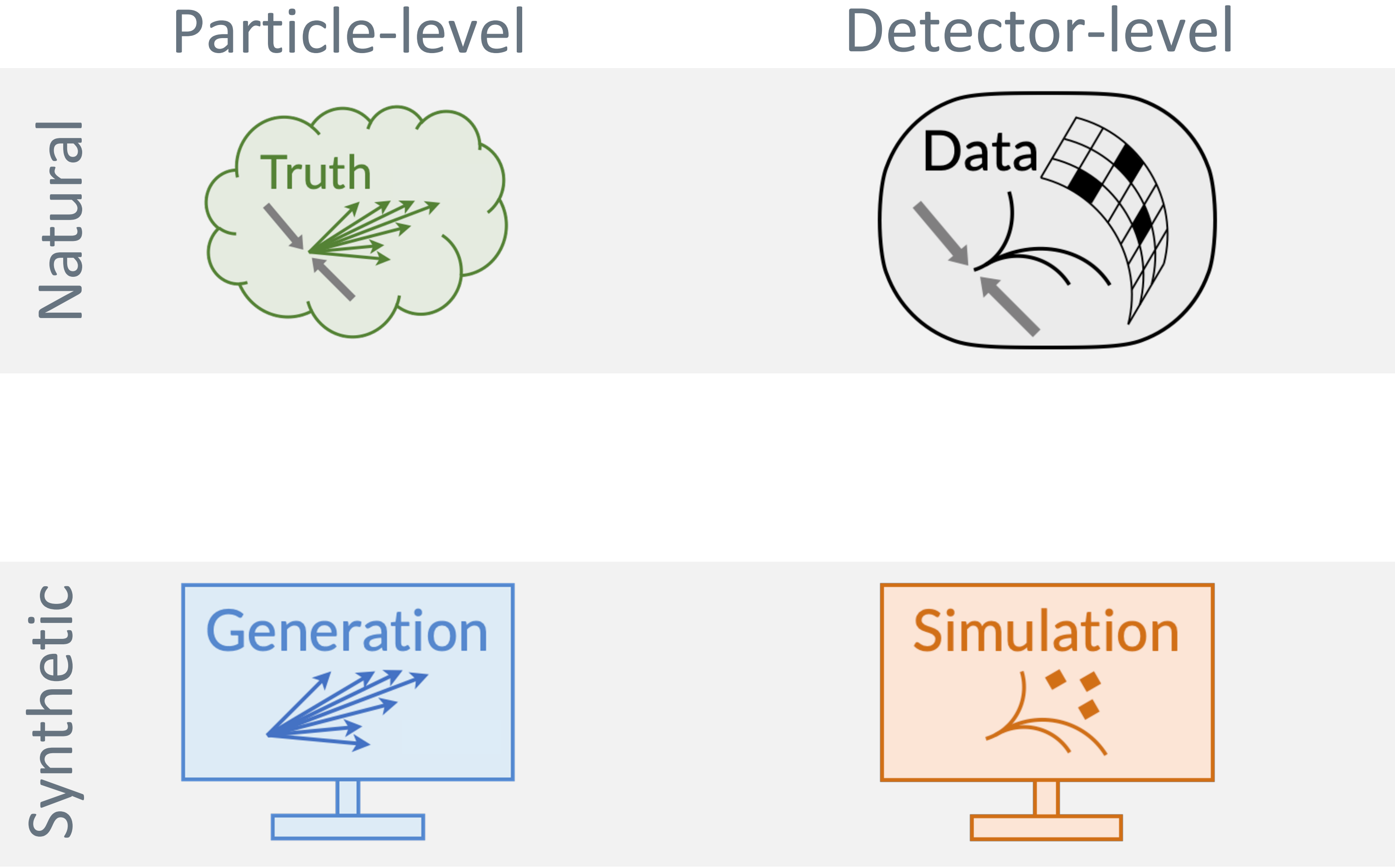
- In HEP, the current standard unfolding methods (e.g. Richardson-Lucy AKA Iterative Bayesian Unfolding (IBU))
  - Take only **binned data**
  - Are unfeasible for unfolding **in more than a small number of dimensions**
  - May not capture and remove **all salient detector effects**
- Output is a 1D histogram

OmniFold

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

- OmniFold is a simulation-based, maximum likelihood procedure which uses deep learning to do unfolding
  - Specifically, iterative neural network reweighting (information in the backup slides)



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

The Starting Point for OmniFold

OmniFold

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

- In contrast to other methods, with NNs, OmniFold…
  - Can take **unbinned data**
  - Easily extendable to **multidimensional** observables or even the **full phase-space**
    - And thus can capture all detector effects
- The output of OmniFold is a per event **reweighting function** that reweights Generation to Truth

Flavors of OmniFold

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

- UniFold: events represented by **one** observable
  - 1-D input to NN
  - Equivalent to unbinned version of standard methods
- MultiFold: events are represented by **multiple** observables
  - n-D input to NN
- OmniFold: **full phase space** events
  - Events are represented as unordered lists of particles and their features
  - Variable dimensional input to NN

Four Functions of a Complete Unfolding Algorithm

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

1. Background subtraction
2. Resolution effects
3. Fake factors
4. Efficiency factors

Four Functions of a Complete Unfolding Algorithm

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

1. **Background subtraction**
2. Resolution effects
3. **Fake factors**
4. **Efficiency factors**

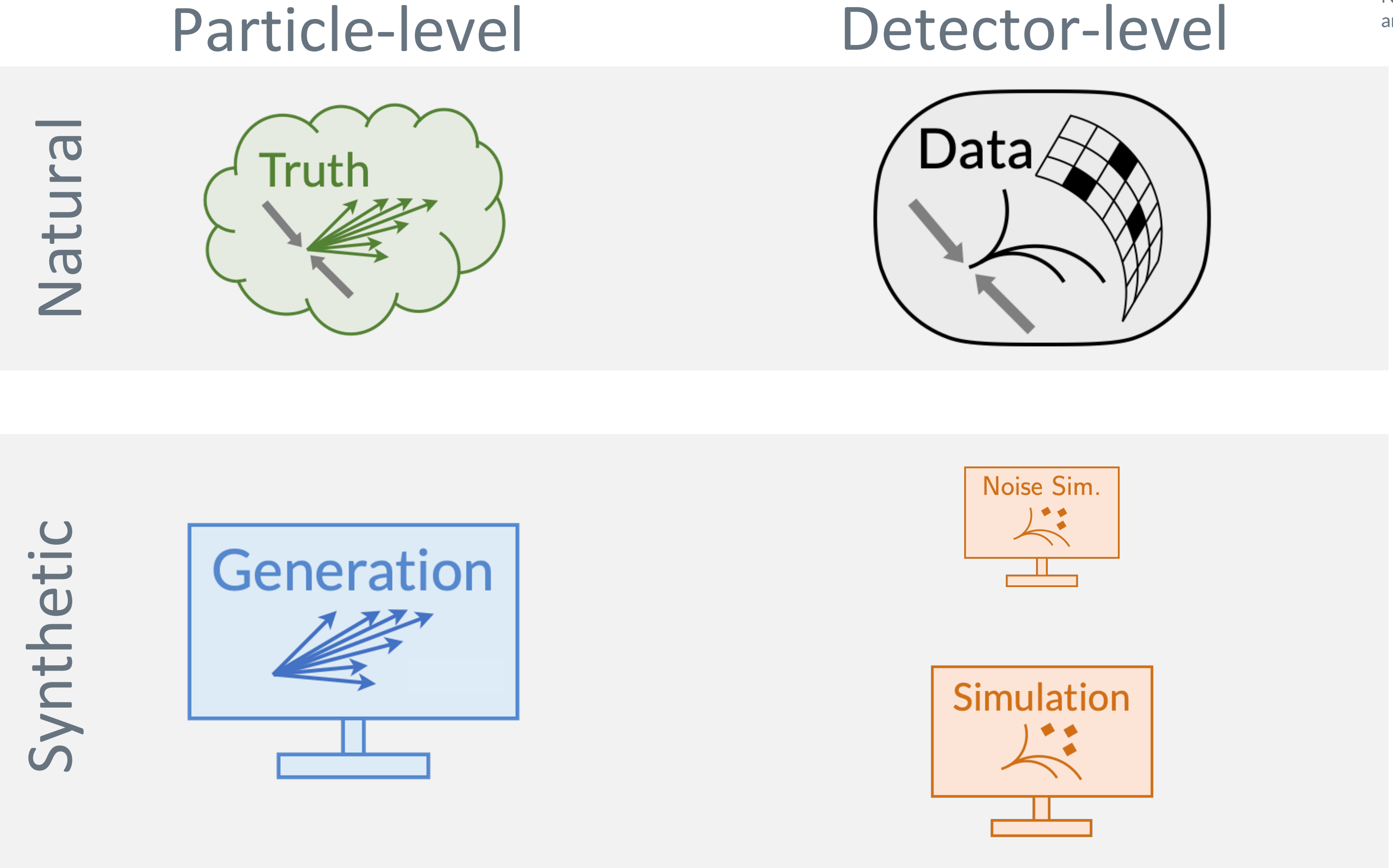
Fleshing out **1**, **3**, and **4** are **improvements** made in this work, where the original paper only addressed **2**.

1. Background Subtractions

Using Neural Positive Reweighting

Neural Positive Reweighting

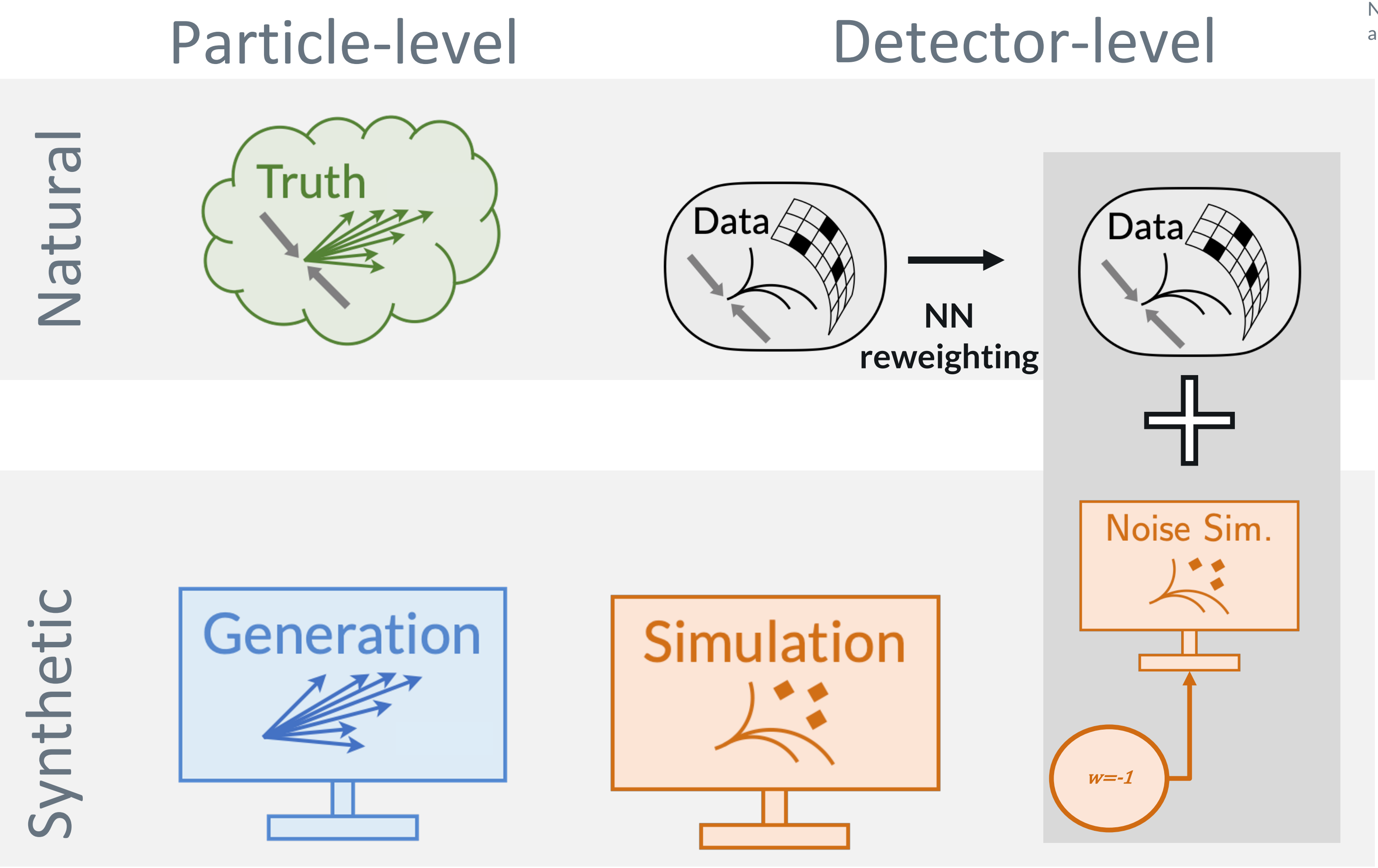
- To remove background noise processes from the data, we employ neural positive reweighting **before** dealing with any detector resolution effects.
- This requires producing another detector simulation of **only the background noise processes**



Additional detector simulation of noise only

Neural Positive Reweighting

- We assign **event weights of -1** to the background simulation, and then **concatenate** this sample with the data sample
  - This effectively **removes** the background noise from the data
- However, to avoid dealing with negative weights, we then reweight the nominal data sample to the new concatenated sample with **only positive event weights**



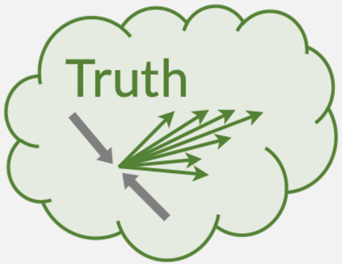
Neural positive reweighting to remove noise from data

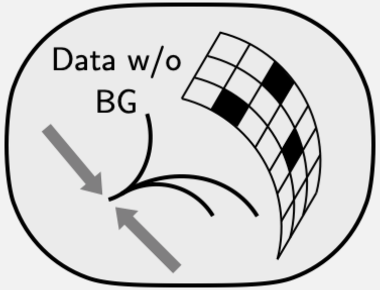
Nachman, Thaler
arXiv:2105.04448

Particle-level

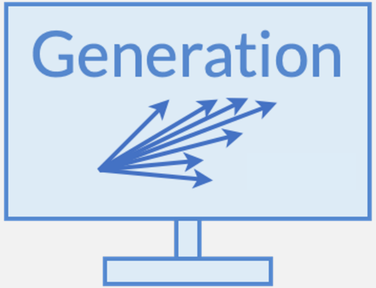
Detector-level

Natural





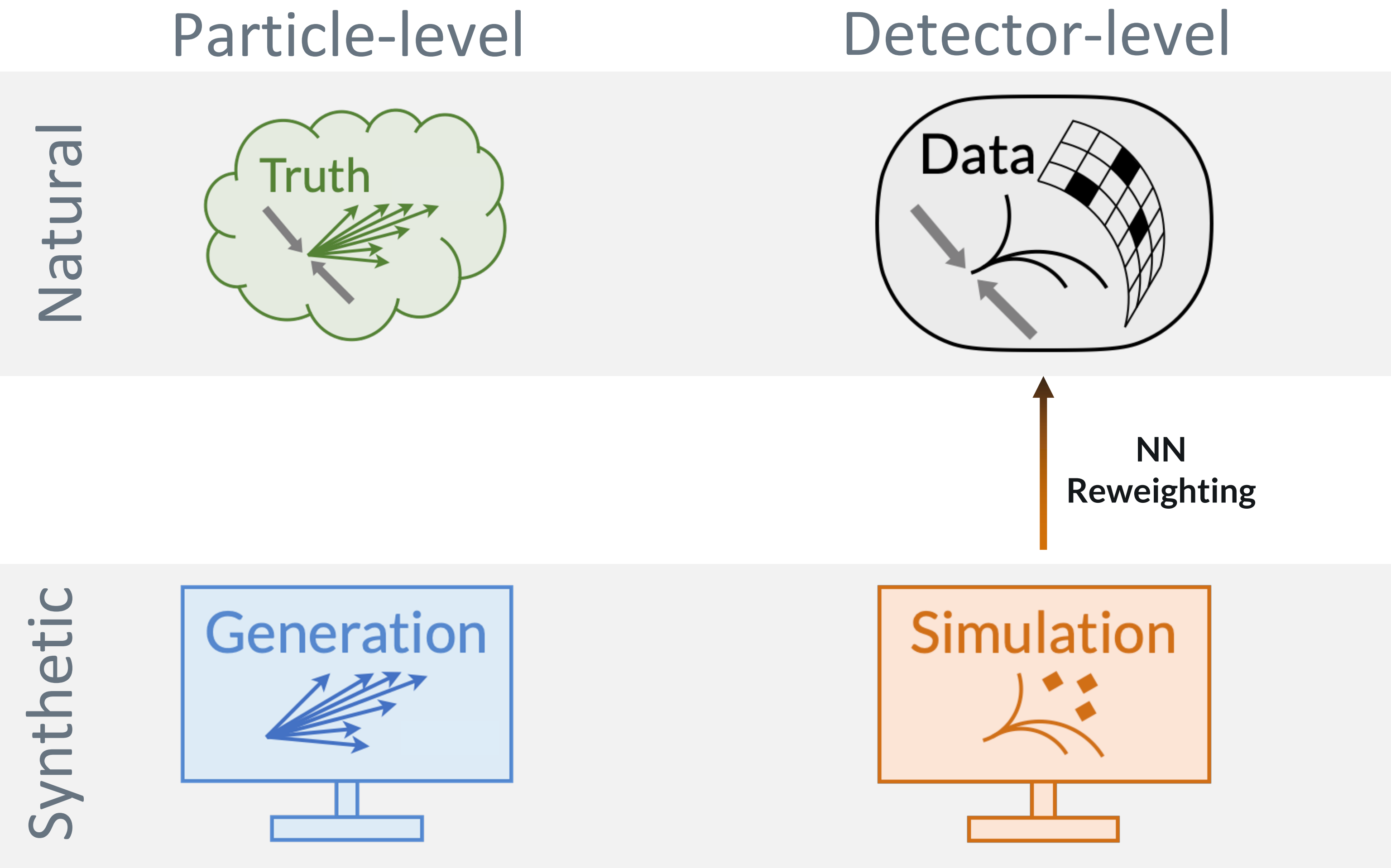
Synthetic



The New Starting Point for OmniFold

2. Resolution Effects

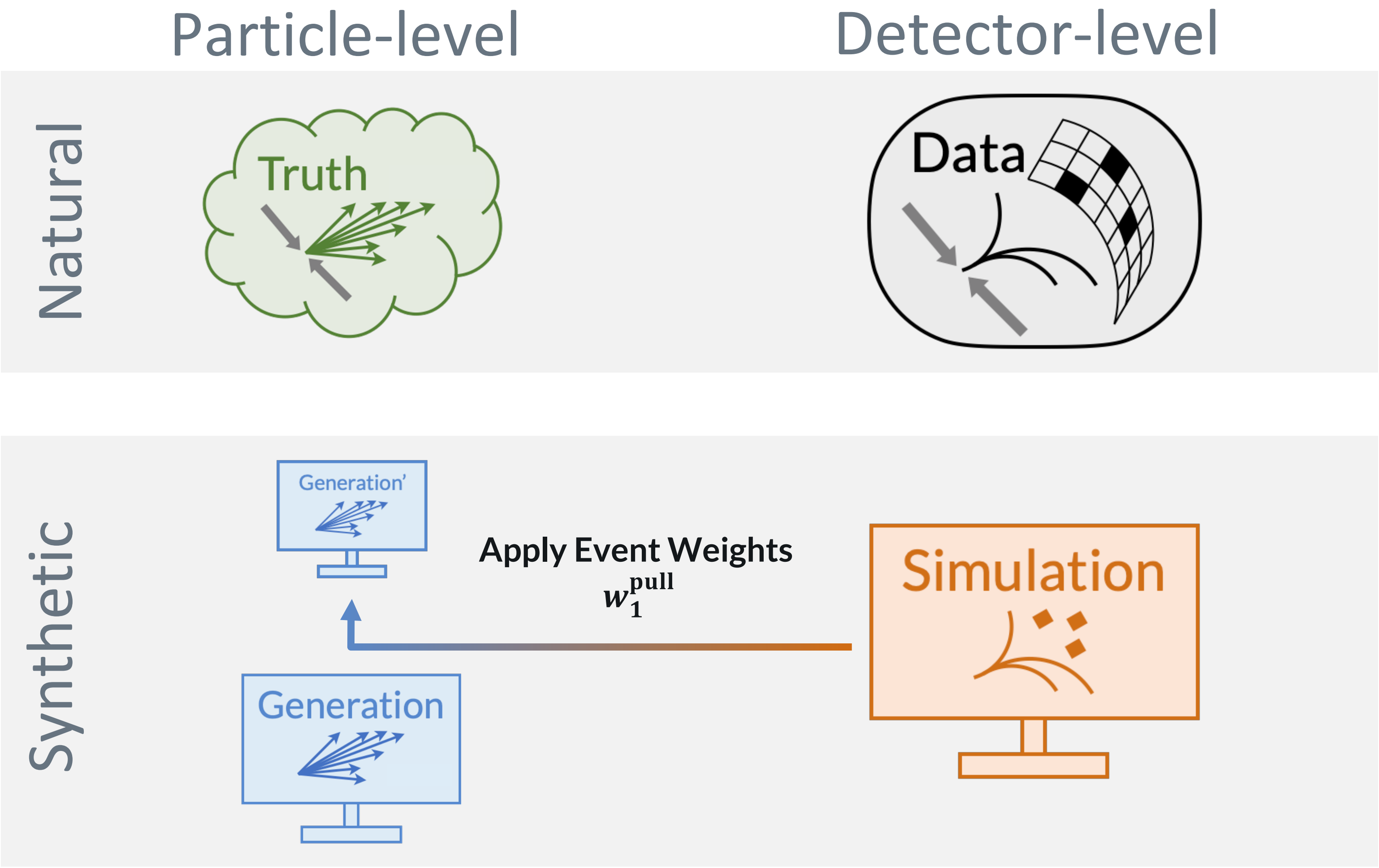
Using Iterative Neural Network Reweighting



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

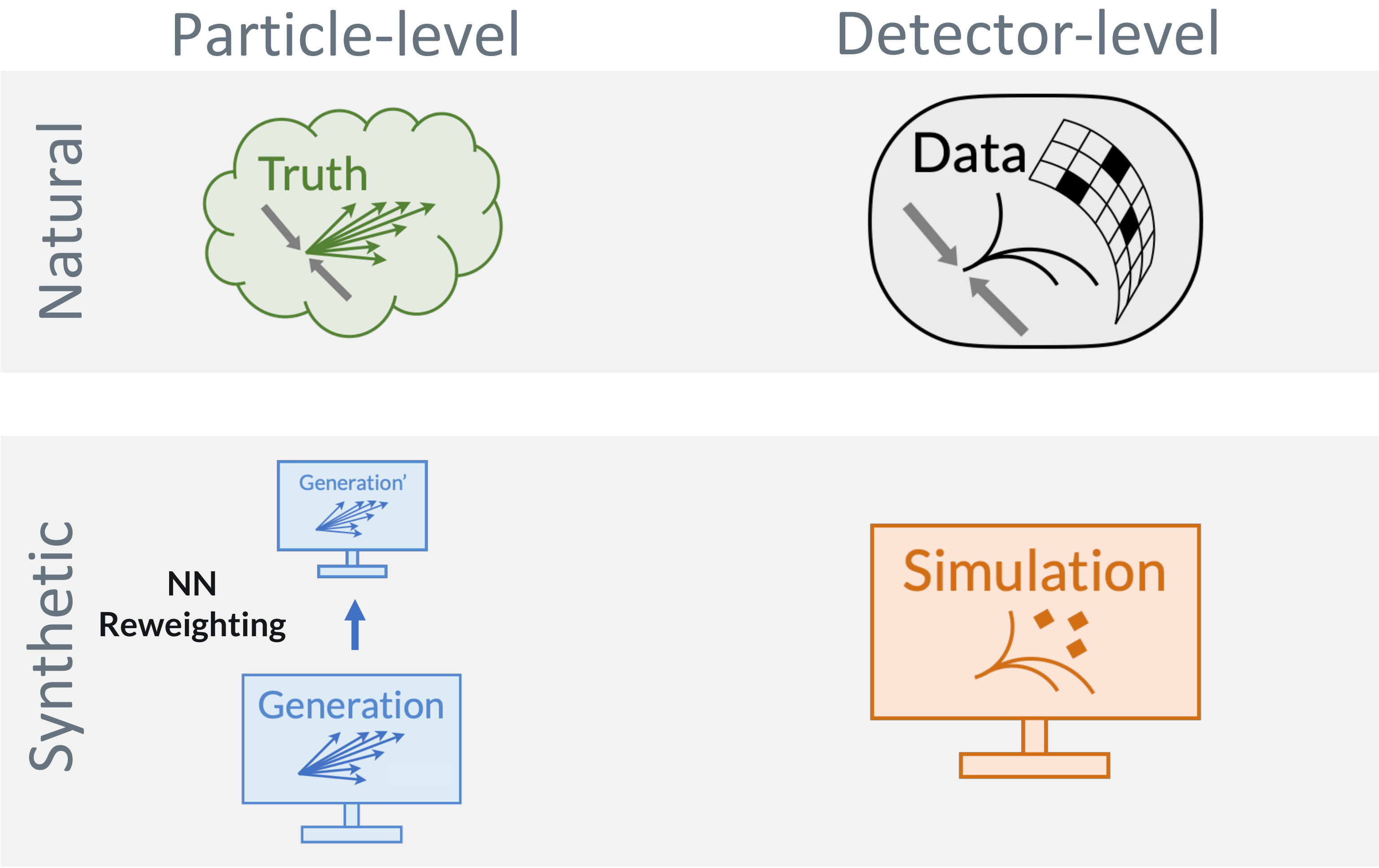
Step 1: Reweight **Sim.** to **Data**



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

Pull weights back to **Gen.** to get new Gen. sample: **Gen.'**

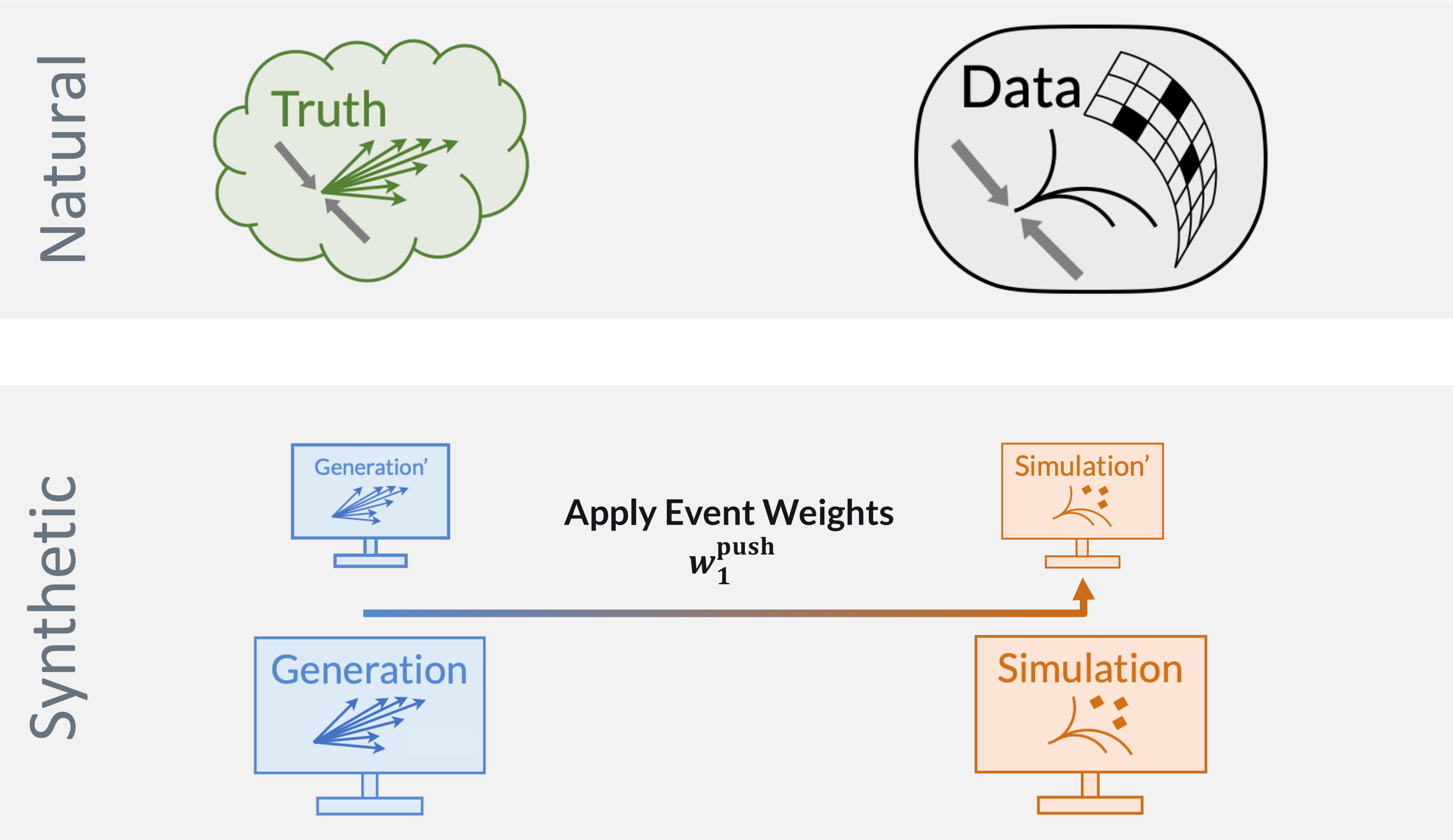


Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

Step 2: Reweight **Gen**. to **Gen.'**

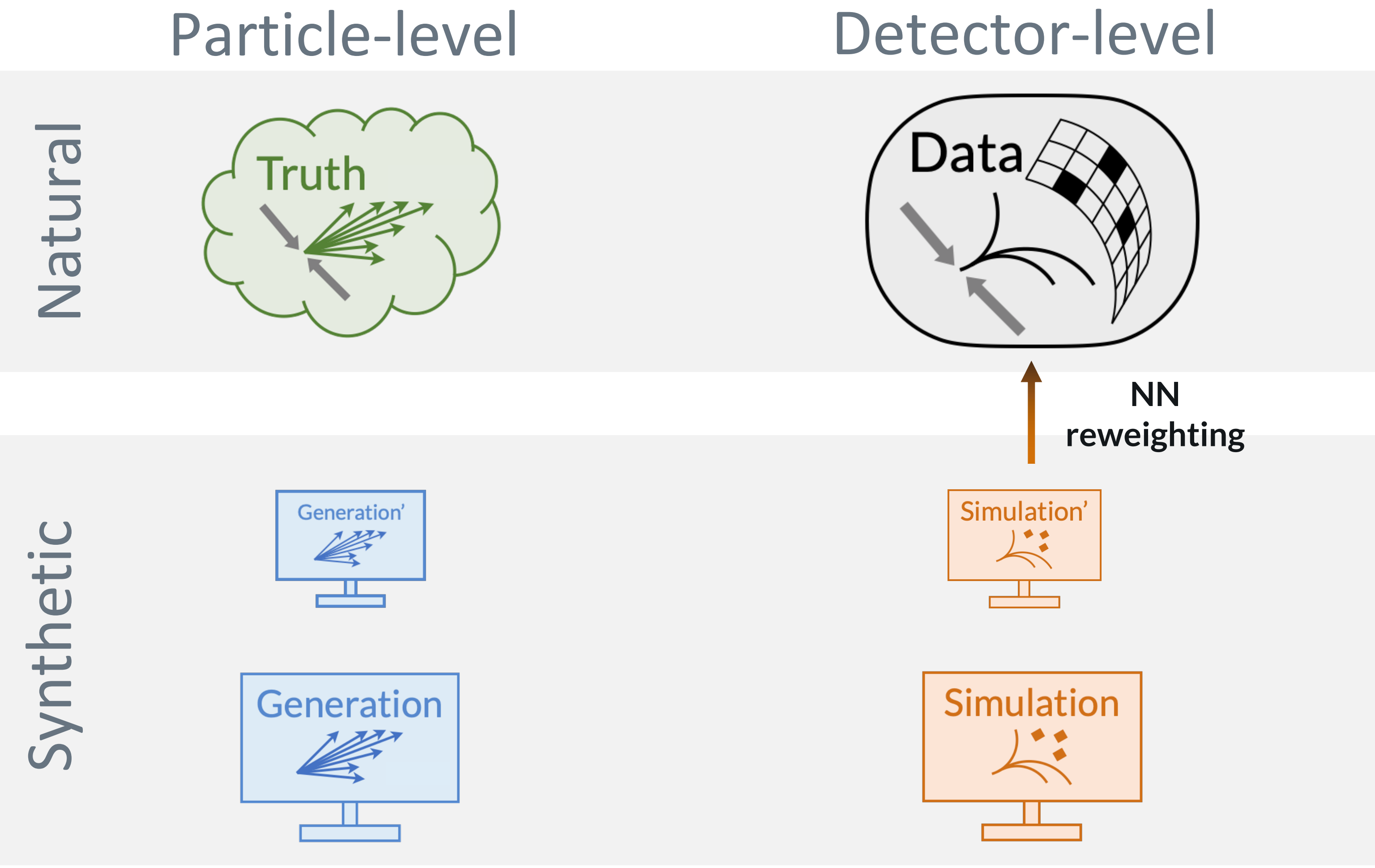
Particle-level Detector-level



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

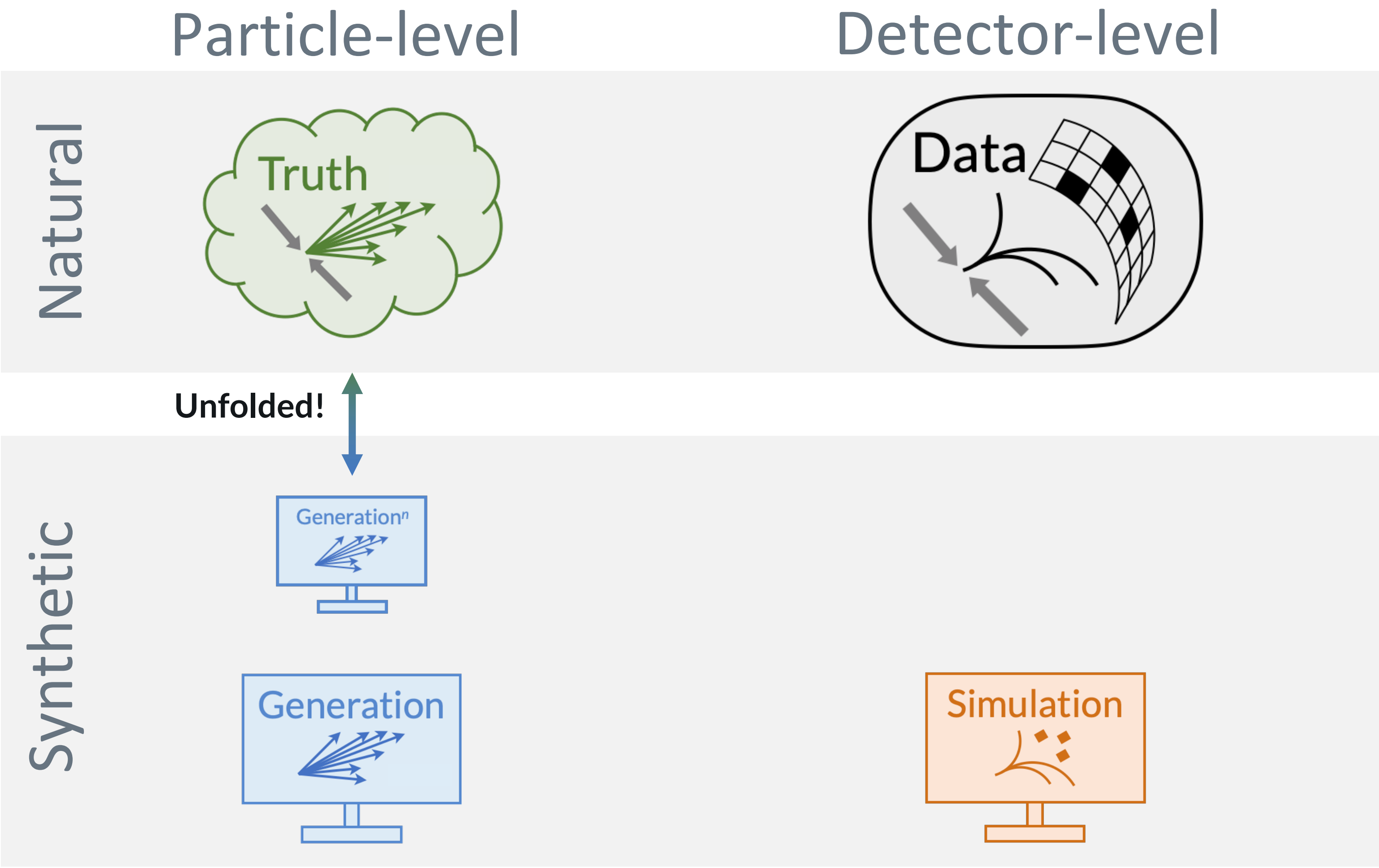
Push weights back to **Sim.** To get new Sim. Sample: **Sim.'**



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

Iterate… Step 1: Reweight **Sim.'** to **Data**



Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

After n iterations, take reweighted Gen^n as the unfolded distribution

3. Fake Factors
&
4. Efficiency Factors

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

What are fake and efficiency factors?

- Fake factors are events that pass the detector-level selection but not the particle-level selection
- Efficiency factors are events that pass the particle-level selection but not the detector-level selection

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

How to deal with fake/efficiency factors

- Simple: assign a **dummy value** to either the particle-level information (for efficiency factors) or the detector-level information (for fake factors) to flag them

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

How to deal with fake/efficiency factors

- Two options when we encounter a dummy value when doing reweighting:
  1. Take the weight of the prior $(w = 1)$
  2. Take the average weight
     - $\langle w|x_{\text{particle}}\rangle$ for efficiency factors
     - $\langle w|x_{\text{detector}}\rangle$ for fake factors

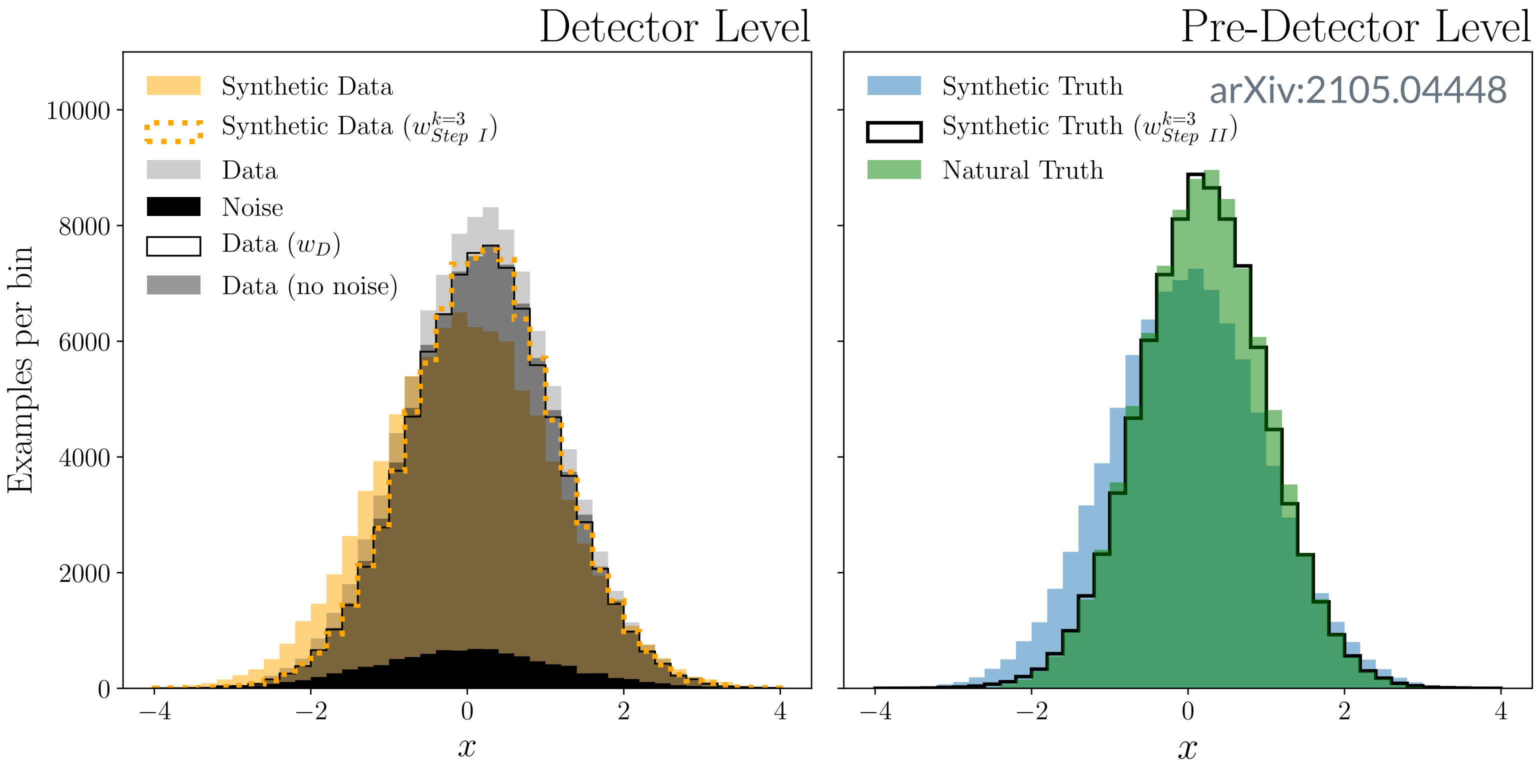
Gaussian Toy Example

Andreassen, Komiske, Metodiev, Nachman, Suresh, Thaler arXiv:2105.04448

Unfolding a Gaussian Distribution
(from arXiv:2105.04448)

- $X_T \sim \mathcal{N}(0.2, 0.8)$ ("Truth"/Target)
- $X_G \sim \mathcal{N}(0, 1)$ (Generation)
- $X_D \sim X_T + Z$ ("Data")
- $X_S \sim X_G + Z$ (Simulation)
- $Z \sim \mathcal{N}(0, 0.5)$ (Detector distortion)
- 10% of X_D are background noise processes $\mathcal{N}(0, 1.1)$
- 10% of X_D and X_T have fake or efficiency factors

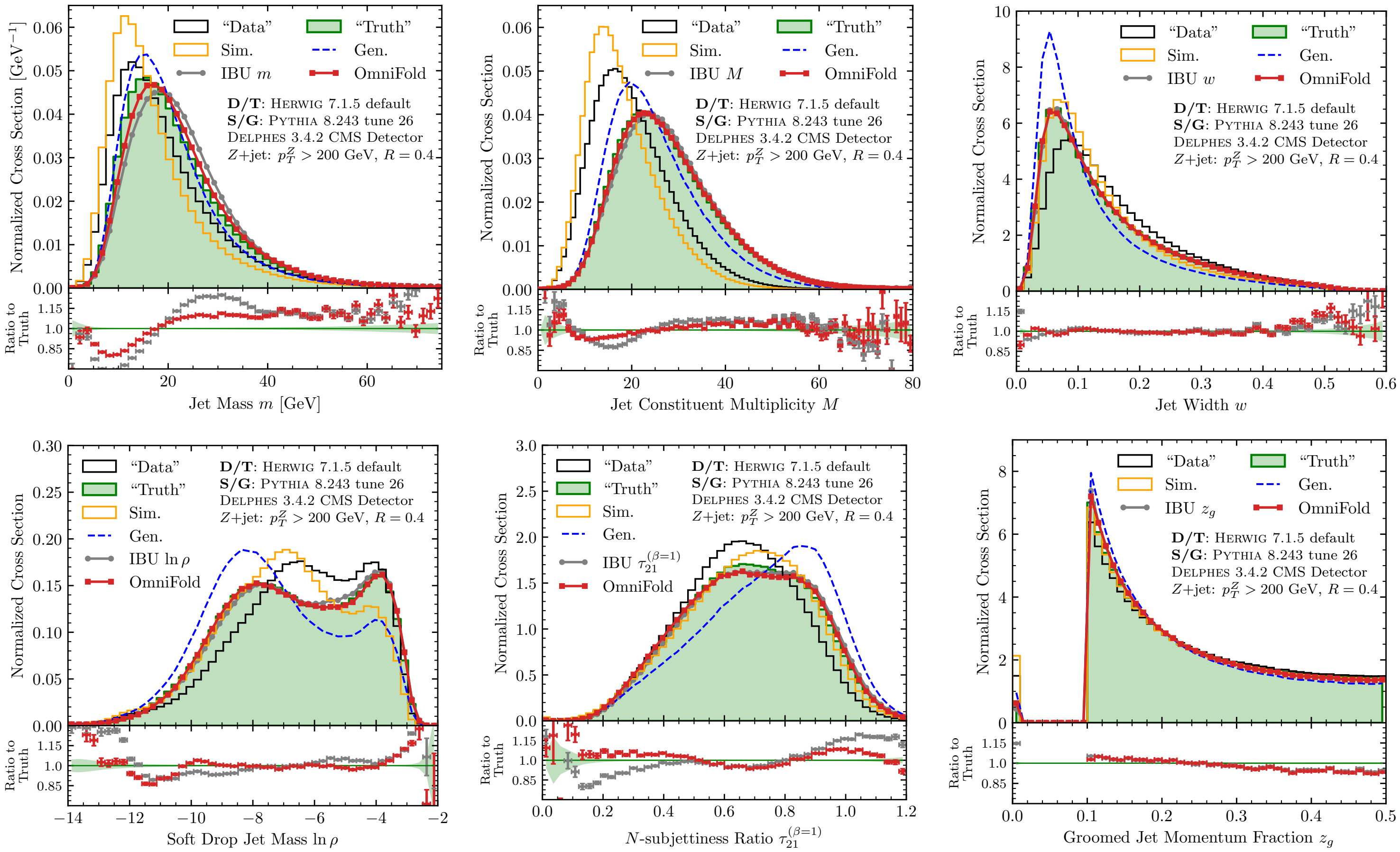
Andreassen,
Komiske,
Metodiev,
Nachman, Suresh,
Thaler
arXiv:2105.04448



OmniFold on a Gaussian distribution after 3 iterations.

HEP Example

$$pp \to \boldsymbol{Z} + \mathbf{jets}$$

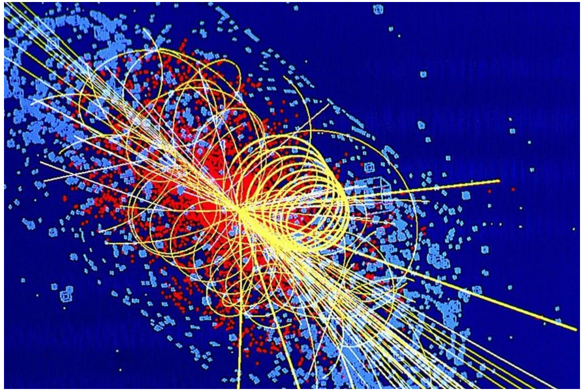


Andreassen, Komiske, Metodiev, Nachman, Thaler PRL 124 (2020) 182001

The unfolding results for six jet substructure observables, using Herwig 7.1.5 ("Data"/"Truth") and Pythia 8.243 tune 26 (Sim./Gen.), unfolded with OmniFold and compared to IBU

THANK YOU!

Any questions?



You can contact me at:

» adisurtya@berkeley.edu

Software at:

» https://github.com/hep-lbdl/OmniFold

Backup:
How NN Reweighting Works

DCTR – Deep Neural Networks using **Classification** for **Tuning** and **Reweighting**

A. Andreassen, B Nachman
PRD RC 101 (2020) 091901

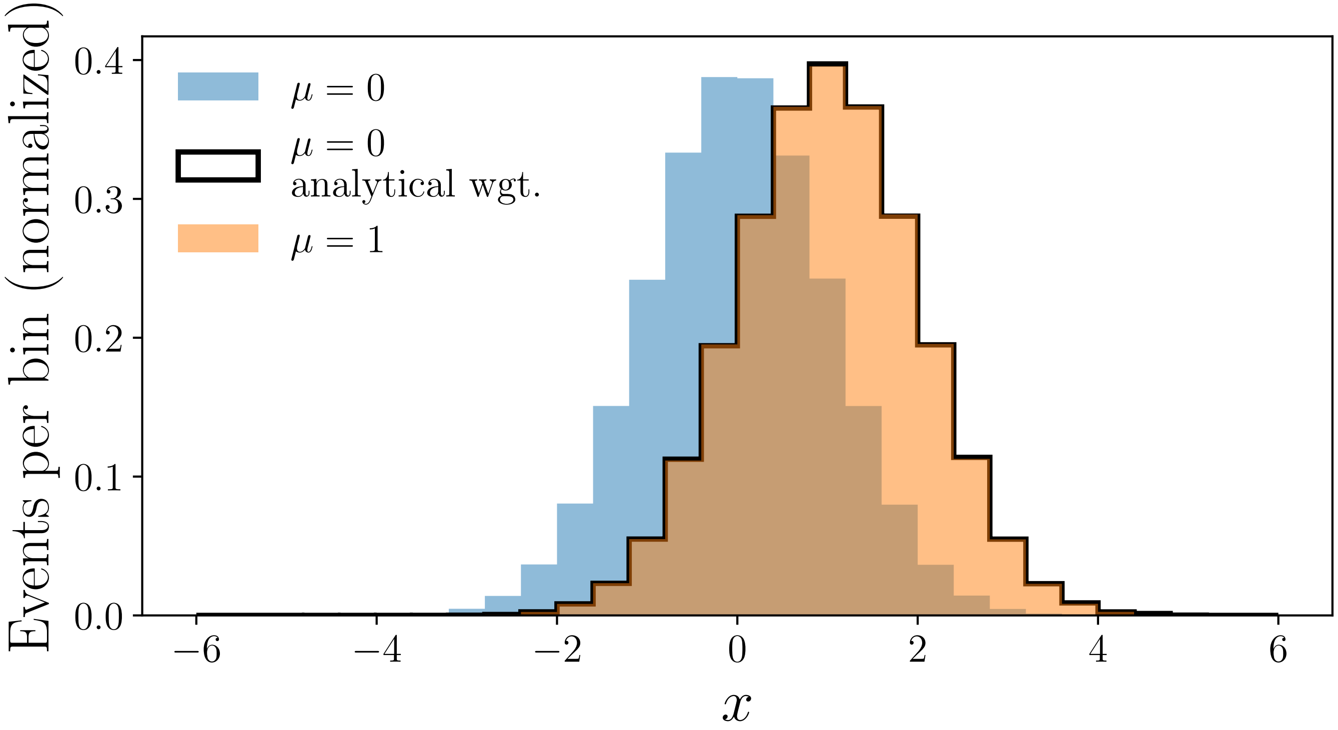
Reweighting from One Sample to Another

- Suppose Sample A is drawn from $p_0(x)$
- Suppose Sample B is drawn from $p_1(x)$
- Reweighting function
  - Takes an event x from Sample A and reweights it to an event from Sample B
  - $w_{A\to B}(x) = \frac{p_1(x)}{p_0(x)}$

A. Andreassen, B Nachman
PRD RC 101 (2020) 091901

Example: Gaussian Distribution

- Suppose Sample A is drawn from $\mathcal{N}(0,1)$
  - $p_0(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$
- Suppose Sample B is drawn from $\mathcal{N}(1,1)$
  - $p_1(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-1)^2}$
- Then the reweighting function is:
  - $w_{A\rightarrow B}(x) = e^{-\frac{1}{2}((x-1)^2 - x^2)}$

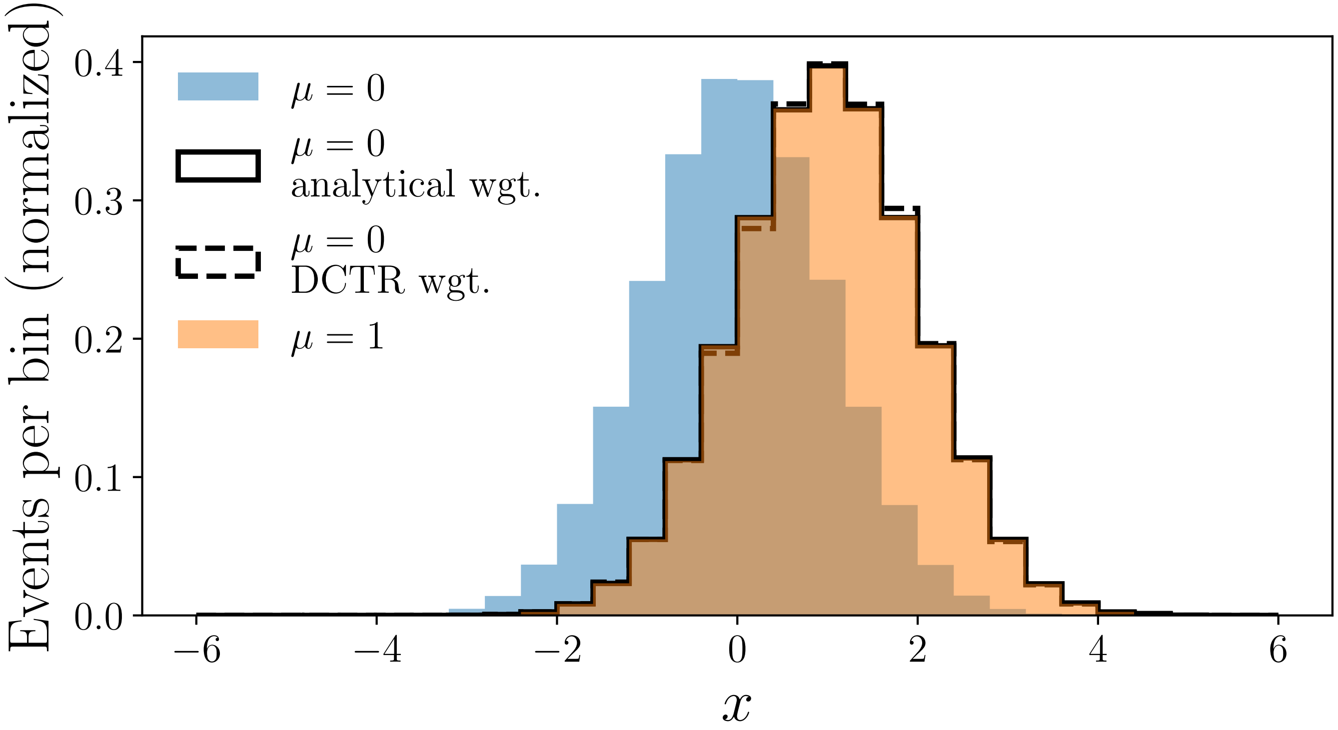


Analytical Reweighting from a Gaussian centered at 0 to a Gaussian centered at 1

Neural Networks and Reweighting

- Reweighting is useful because neural networks can easily approximate the likelihood ratio:
  - Let $f(x) \in [0,1]$ be a binary classifier trained to distinguish Sample A from Sample B
  - $w_{A\to B}(x) = \frac{p_1(x)}{p_0(x)} \approx \frac{f(x)}{1-f(x)}$
    - (**DCTR** Reweighting)

A. Andreassen, B Nachman
PRD RC 101 (2020) 091901



Different reweightings from a Gaussian centered at 0 to a Gaussian centered at 1

Simultaneous and Full Phase-space Reweighting

- An event x may be multidimensional and/or of variable dimension
  - Could contain multiple observables
  - Could contain the full phase-space of information
- HEP full phase-space
  - $x = \begin{pmatrix} m_1 & p_{T,1} & \eta_1 & \phi_1 \\ \vdots & \vdots & \vdots & \vdots \\ m_n & p_{T,n} & \eta_n & \phi_n \end{pmatrix}$

A. Andreassen, B Nachman
PRD RC 101 (2020) 091901

EnergyFlow: Particle Flow Networks

- Collider data is of **variable dimension**; there is no fixed number of particles per event
  - So how do we structure our neural networks if the input is a different size every time?
- EnergyFlow's **Particle Flow Networks**
  - Python package with a TensorFlow/Keras backend

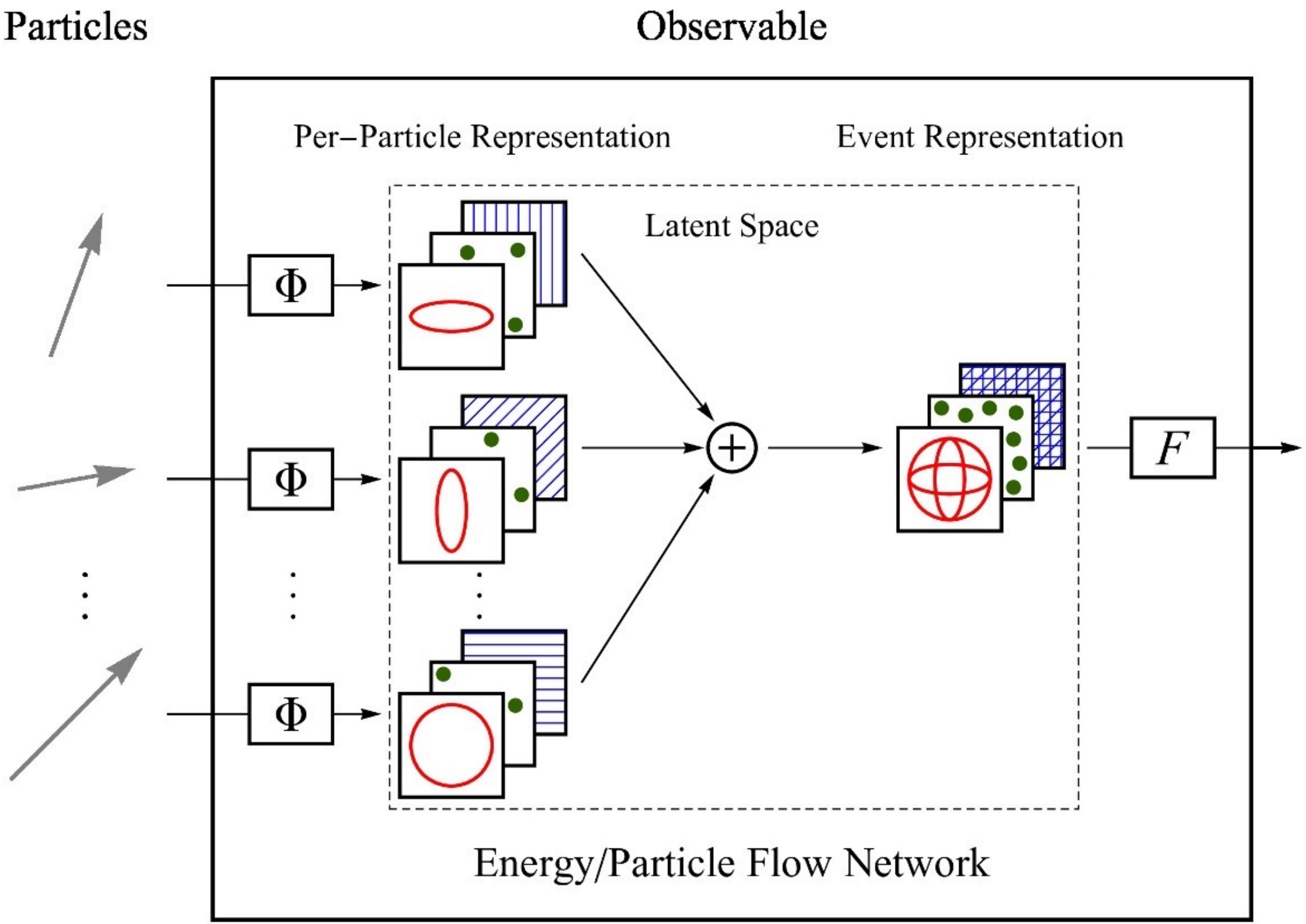
Komiske, Metodiev, Thaler
JHEP 01 (2019) 121

Particle Flow Networks and Deep Sets

- $\text{PFN} = F(\sum_{i=1}^{M} \Phi(p_i))$ (from **Deep Sets**)
  - $p_i \in \mathbb{R}^d$ – relevant per particle info (e.g. four-vector)
  - $\Phi: \mathbb{R}^d \rightarrow \mathbb{R}^l$ – per particle mapping
    - $\mathbb{R}^l$ is a **latent** space
  - $\text{F}: \mathbb{R}^l \rightarrow \mathbb{R}$ – continuous function
    - $\text{F}: \mathbb{R}^l \rightarrow [0, 1]$ (binary classifier output)
- Because of the **sum**, also respects the **permutation invariance** of collider data!

Komiske, Metodiev, Thaler
JHEP 01 (2019) 121

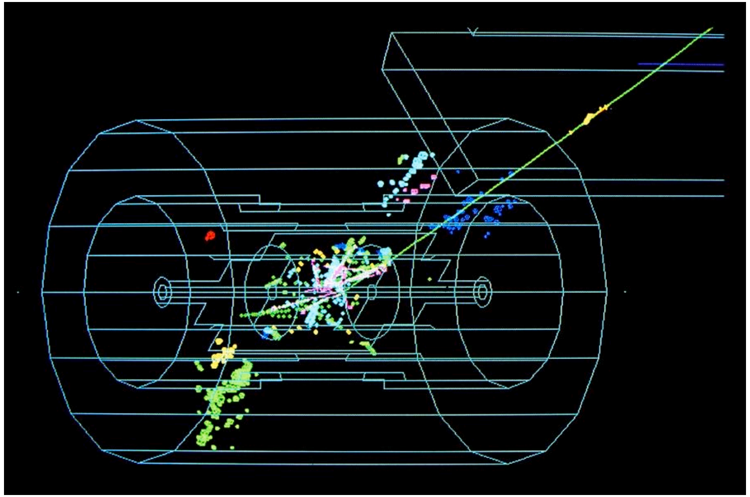
Komiske, Metodiev, Thaler
JHEP 01 (2019) 121

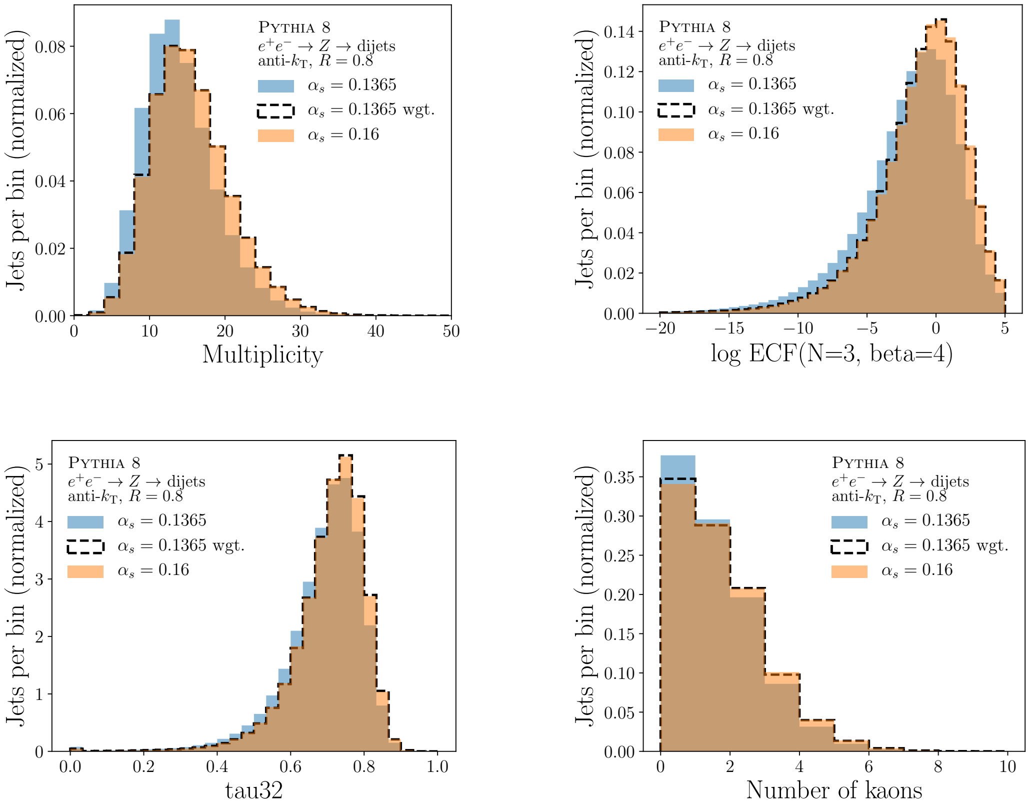


Particle Flow Network Schematic Diagram

HEP Example: electron-positron collisions

- $e^+e^- \to Z \to \text{dijets}$
- It is difficult to visualize such high-dimensional information, so we **construct several observables** to **visualize** and **probe** the phase space!





Full phase-space reweighting from $\alpha_S = 0.1365 \to 0.1600$ for particle level $e^+e^- \to Z \to$ dijets