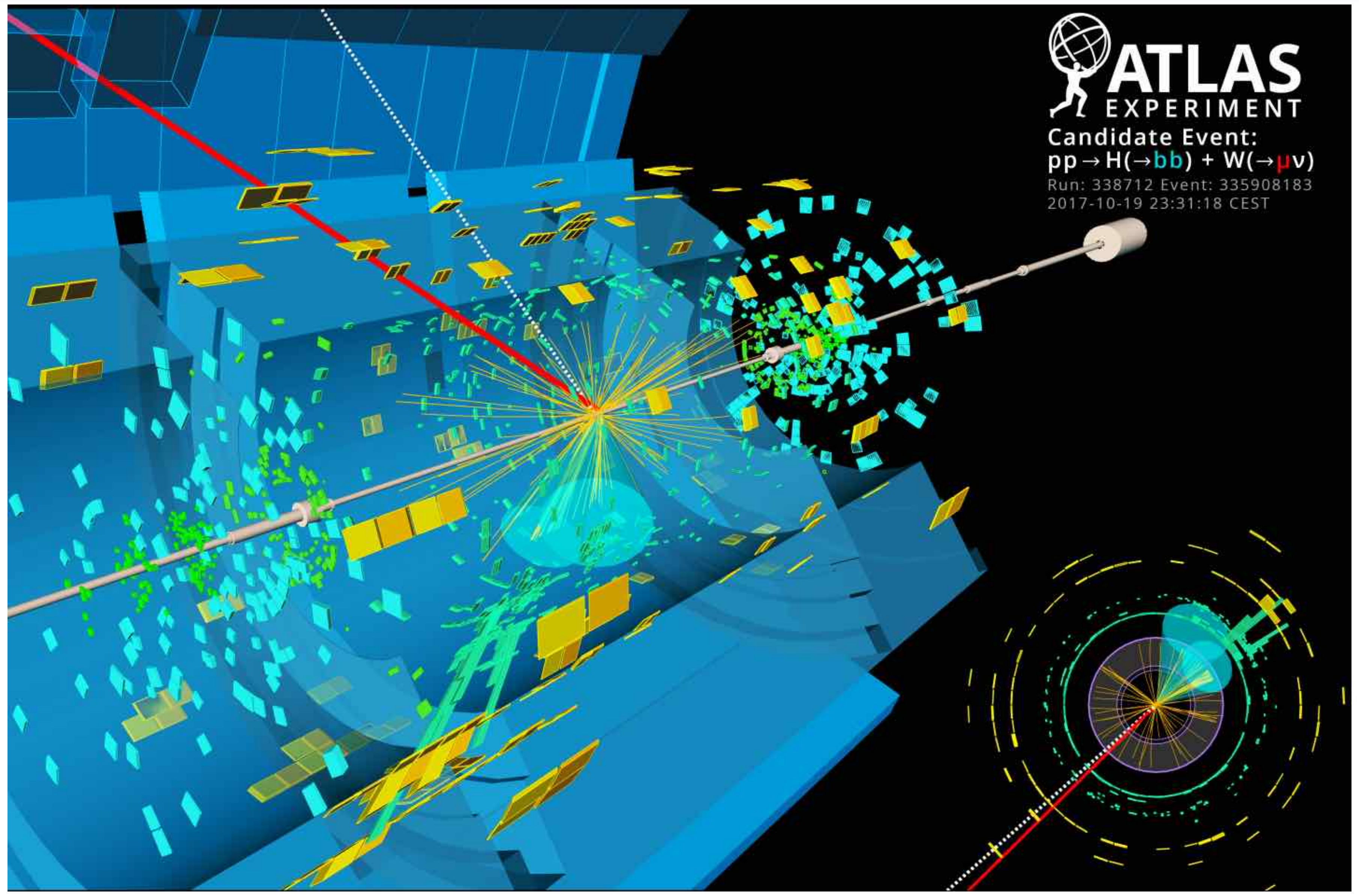
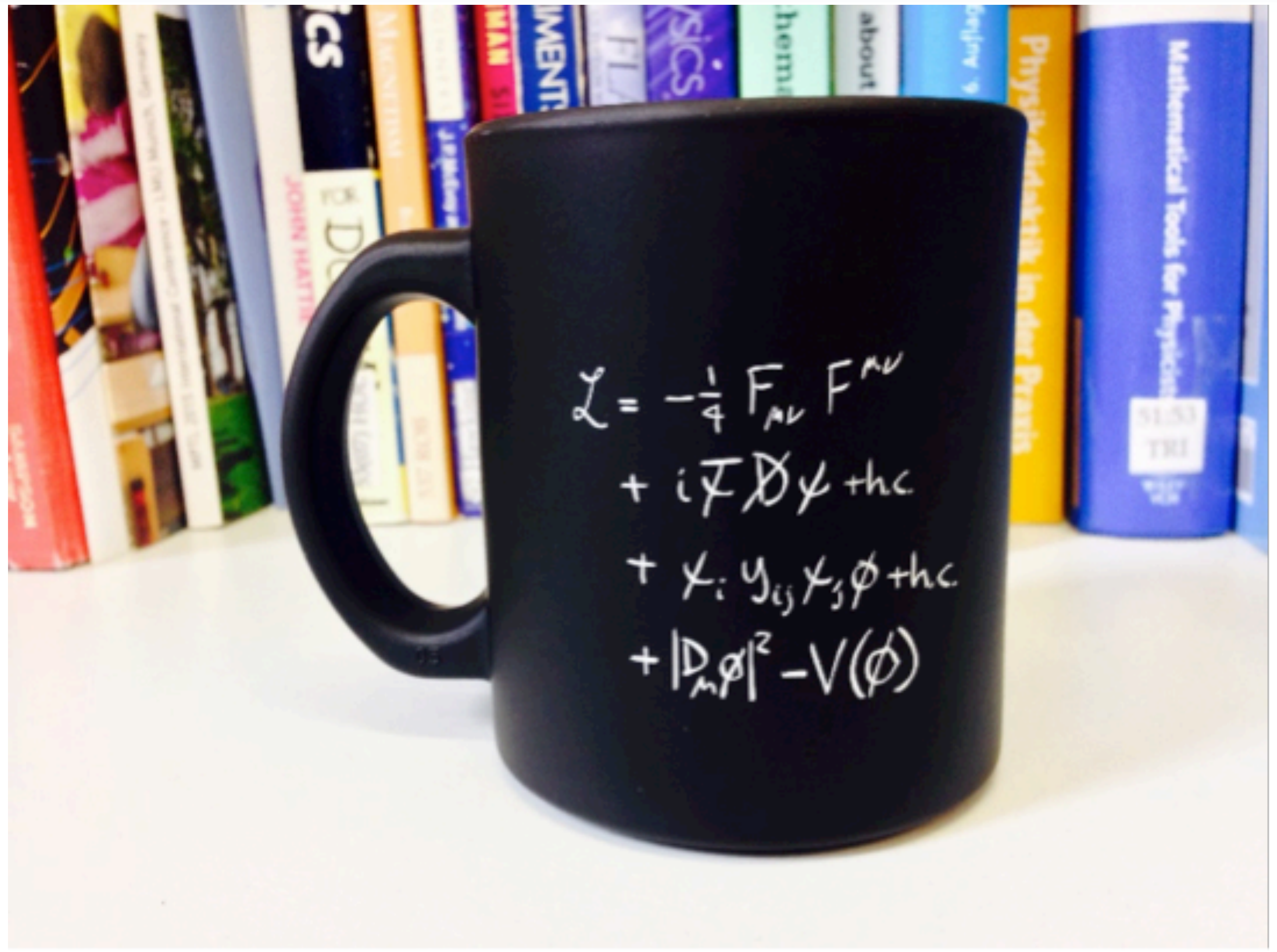
Machine Learning for Better Precision Simulations & New Approaches to Old Analysis Methods

Anja Butter, LPNHE

LPNHE PARIS

A biased view on LHC physics





Setting

- Proton collisions at 13 TeV
- **Huge** dataset ~1Pb/s before trigger selection
- High-dim. kinematic distributions, jet substructures …

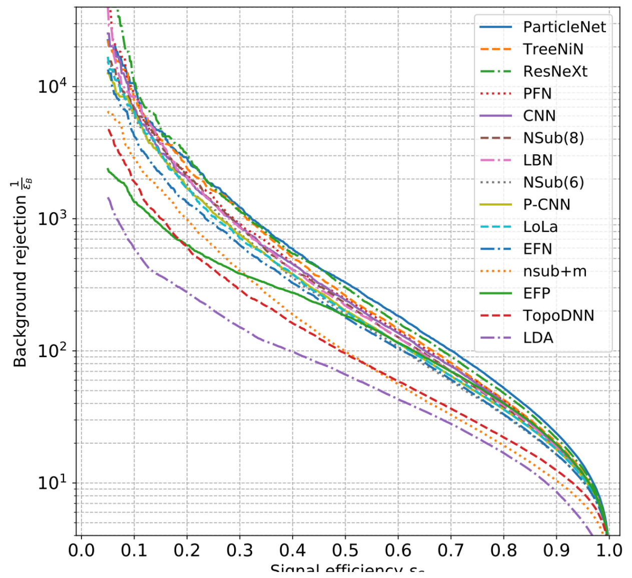
Goal

- Understand full dataset from **1st principles**
- Precision measurements of the SM
- Find signs of new physics (eg dark matter)

How can ML help to exploit all information in the dataset?

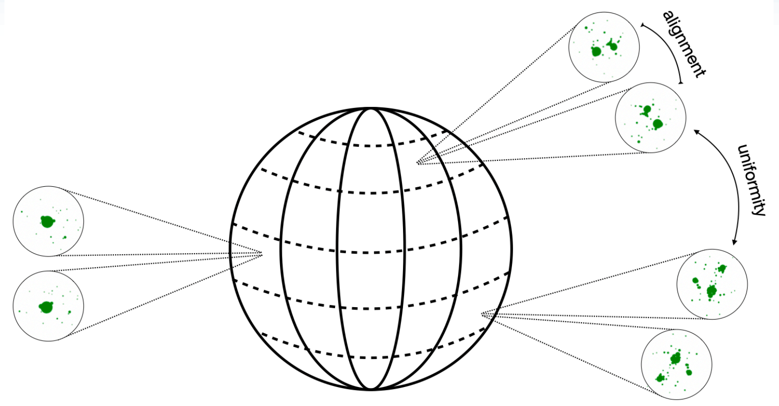
ML for data science in particle physics

Top tagging



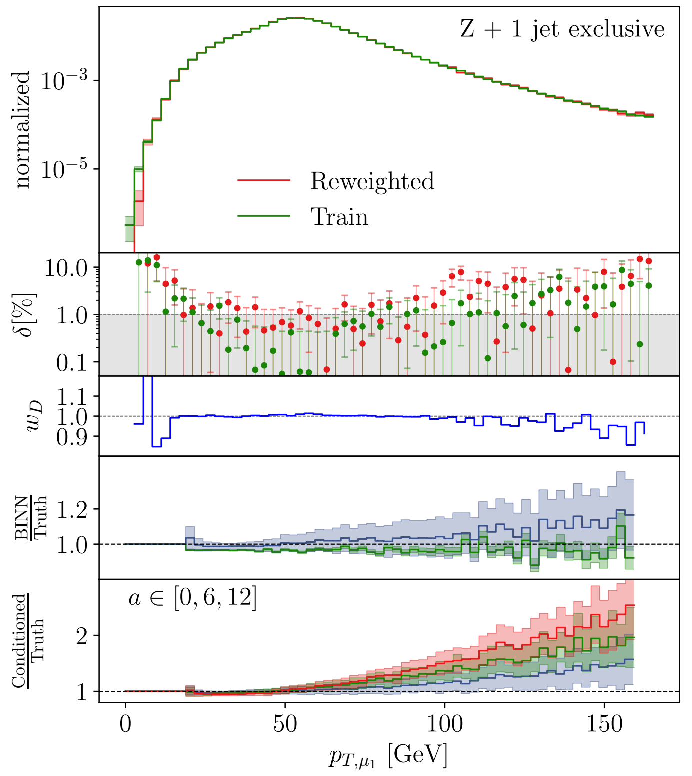
G. Kasieczka et al. [1902.09914]

Anomaly detection



B. Dillon et al. [2108.04253]

Event generation



A. Butter et al. [2110.13632]

Classification

Generative models

Graph networks

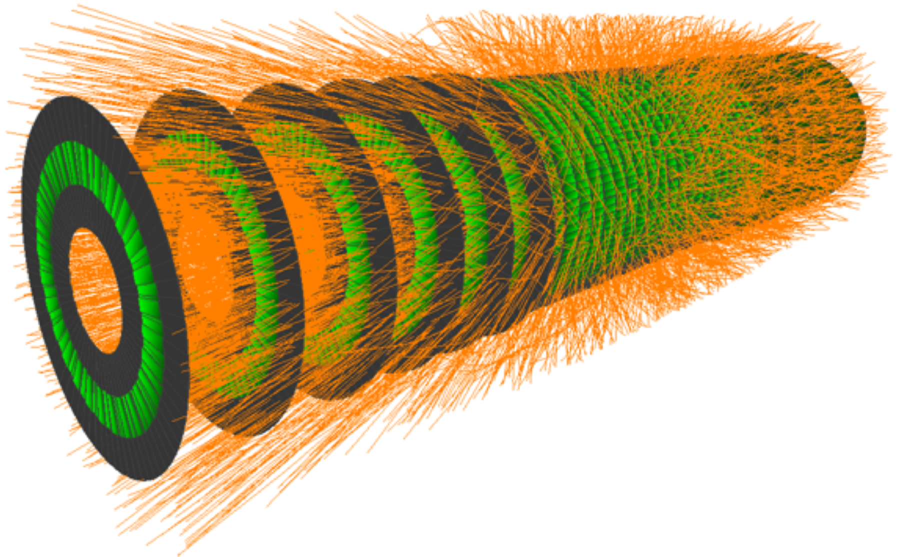
Unsupervised learning

Regression

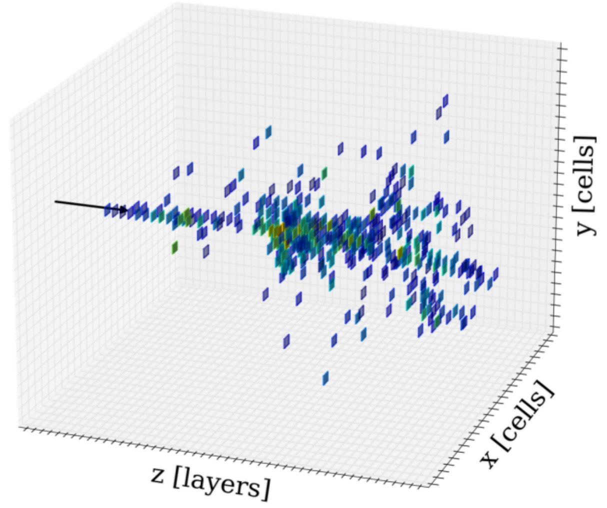
Bayesian networks

Track reconstruction

Kaggle challenge

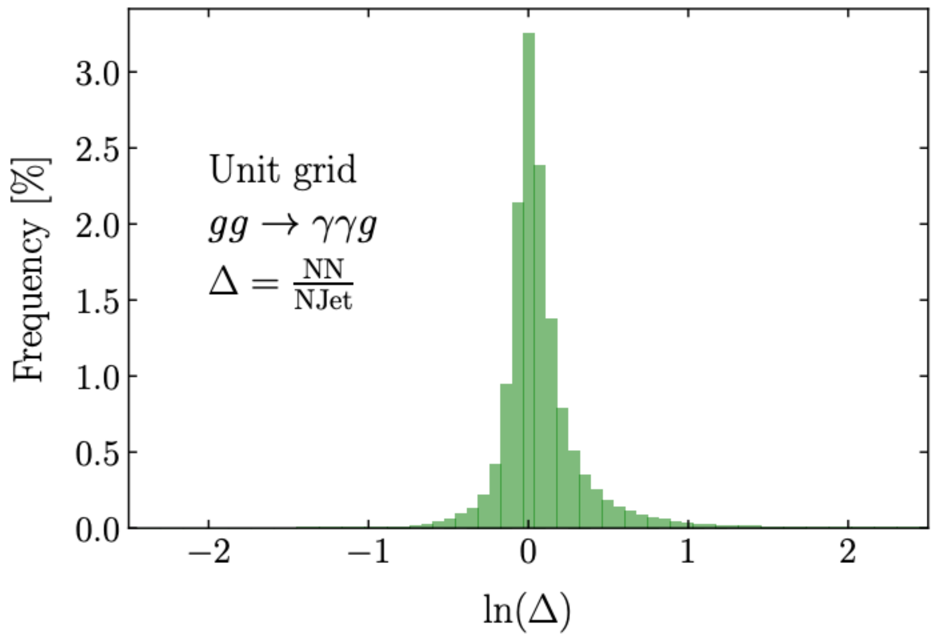


Detector simulation



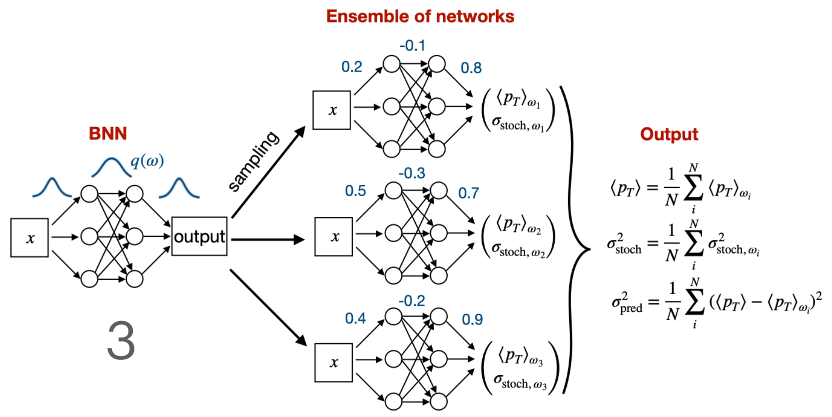
E. Buhmann et al. [2112.09709]

Amplitude estimation



J. Aylett-Bullock, et al. [2106.09474]

Jet calibration & uncertainties



G. Kasieczka et al. [2003.11099]

Complete citations $\mathcal{O}(800)$
https://iml-wg.github.io/HEPML-LivingReview/

Monte carlo event generation

1. Generate phase space points

$\rightarrow$ set of four-momenta p_i

2. Calculate event weight

$$w_{\text{event}} = f(x_1, Q^2) f(x_1, Q^2) \times \mathcal{M}(x_1, x_2, p_1, \ldots, p_n) \times J(p_i(r))$$

PDF — Matrix element — Phase space mapping

3. Unweighting

keep events with $\frac{w_i}{w_{\max}} > r \in [0,1]$

Bottlenecks

1. Slow **matrix element** calculation
   - Complexity grows exponentially with
     - # final state particles
     - Precision (LO, NLO, NNLO, ...)
2. Low **unweighting** efficiency
   - Discard most events if $w_i \ll w_{\max}$
   - Optimize phase space mapping
     ➡ $J(p_i(r)) = (f \times \mathcal{M})^{-1}$

Estimating amplitudes with neural networks

A standard regression problem (?)

Standard approach

Training data

$T = $ (phase space points x, Amplitudes $A'(x)$)

Loss

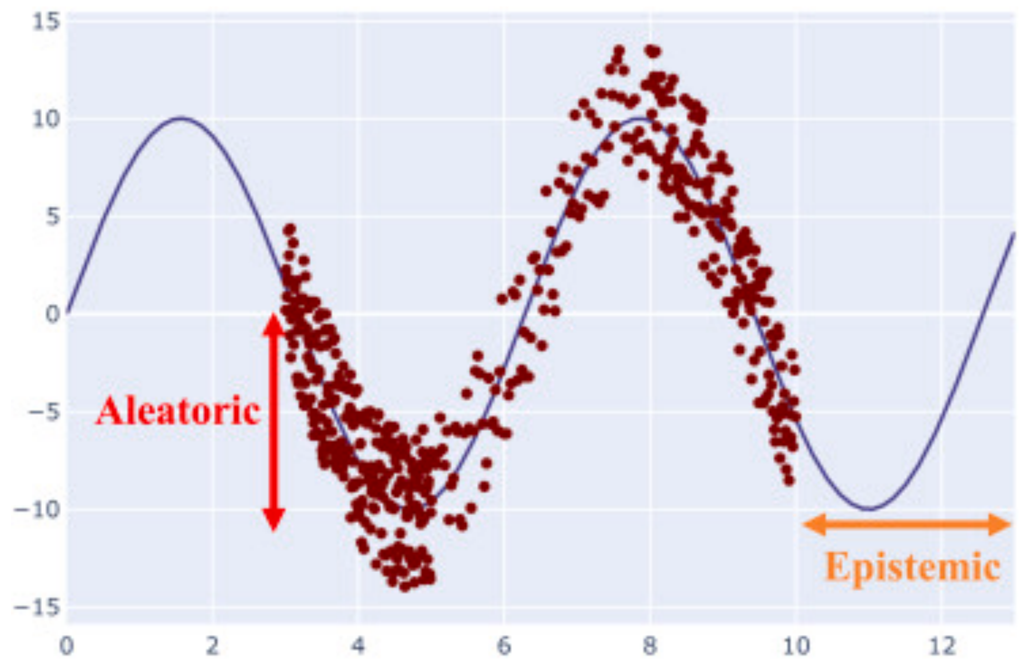
$\mathscr{L} = (A'(x) - NN(x))^2$

PROBLEM: For limited data there is **no unique solution**

→ Need better formulation of the problem

→ Find $p(A \mid x, T)$ (from now on x is implicit)

→ $p(A) = \int \mathrm{d}w \; p(A \mid w) p(w \mid T)$

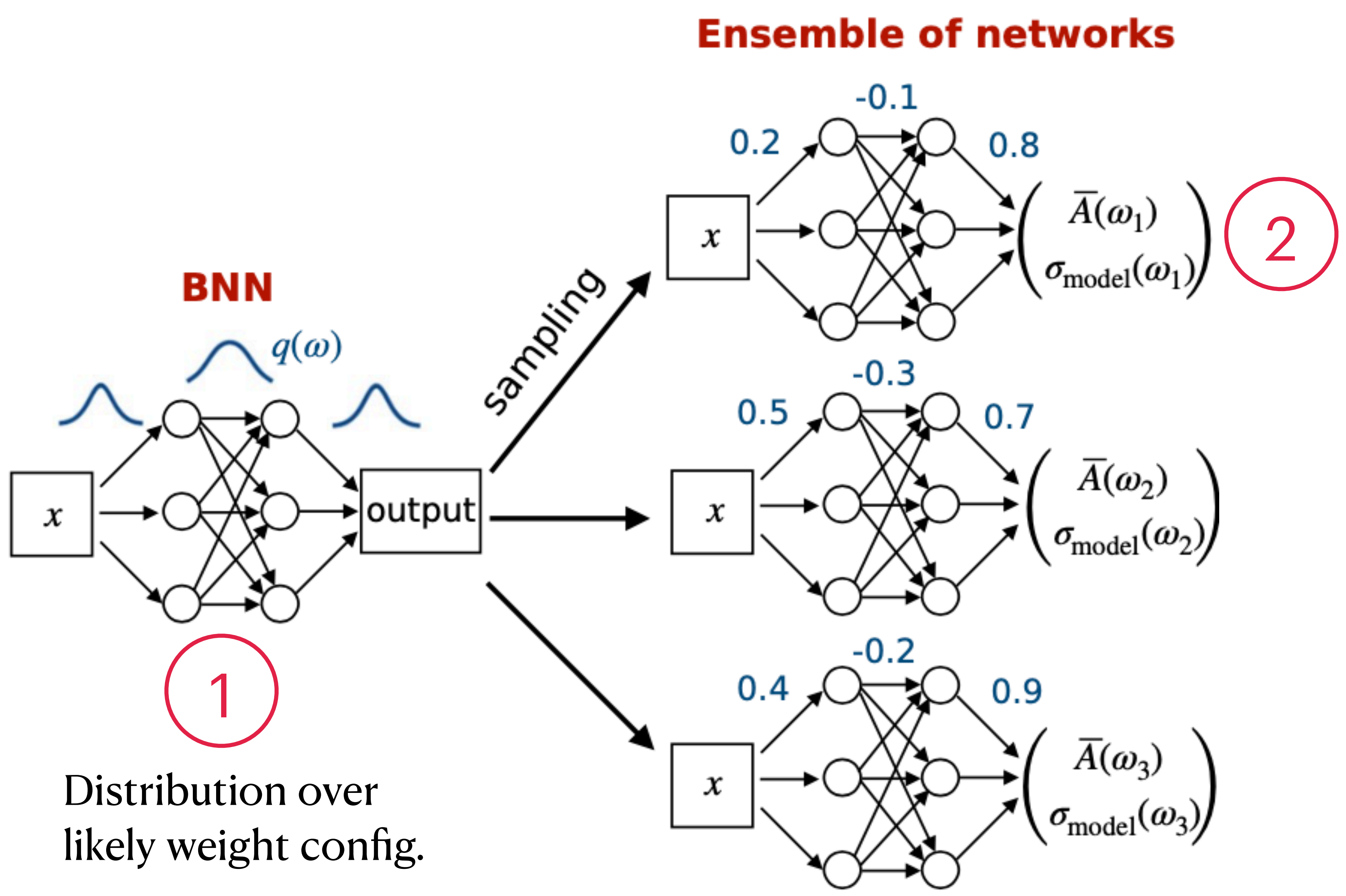


Capturing probabilities with Bayesian networks

$$p(A) = \int dw\; p(A\,|\,w)p(w\,|\,T) \approx \int dw\; p(A\,|\,w)q(w)$$

(2) (1)

Bayesian network



Building the loss function

Approximate $q(w)$ by minimizing KL divergence

$$\mathscr{L}_{BNN} = \text{KL}[q(w), p(w\,|\,T)]$$

$$= \int dw\; q(w)\; \log \frac{q(w)}{p(w\,|\,T)}$$

$$= \int dw\; q(w)\; \log \frac{q(w)p(T)}{p(w)p(T\,|\,w)}$$

$$= \text{KL}[q(w), p(w)] - \int dw\; q(w)\; \log p(T\,|\,w)$$

(1) Gaussian prior

$$\frac{\sigma_q^2 - \sigma_p^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} + \log \frac{\sigma_p}{\sigma_q}$$

(2) Gaussian uncertainty

$$\frac{\left|\bar{A}_j(\omega) - A_j^{(\text{truth})}\right|^2}{2\sigma_{\text{model},j}(\omega)^2} + \log \sigma_{\text{model},j}(\omega)$$

Results - out of the box

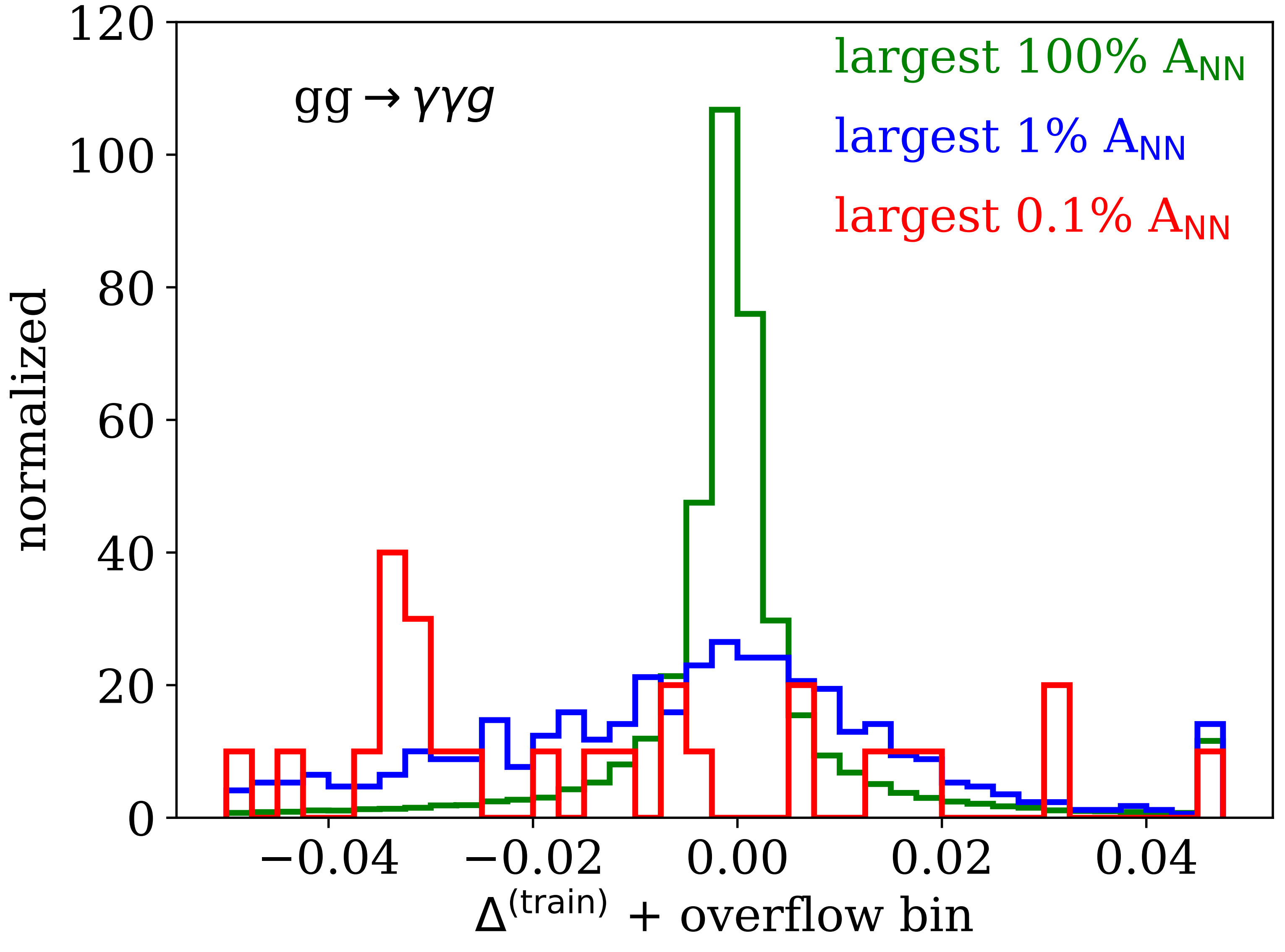
Example

$gg \to \gamma\gamma g(g)$ @LO

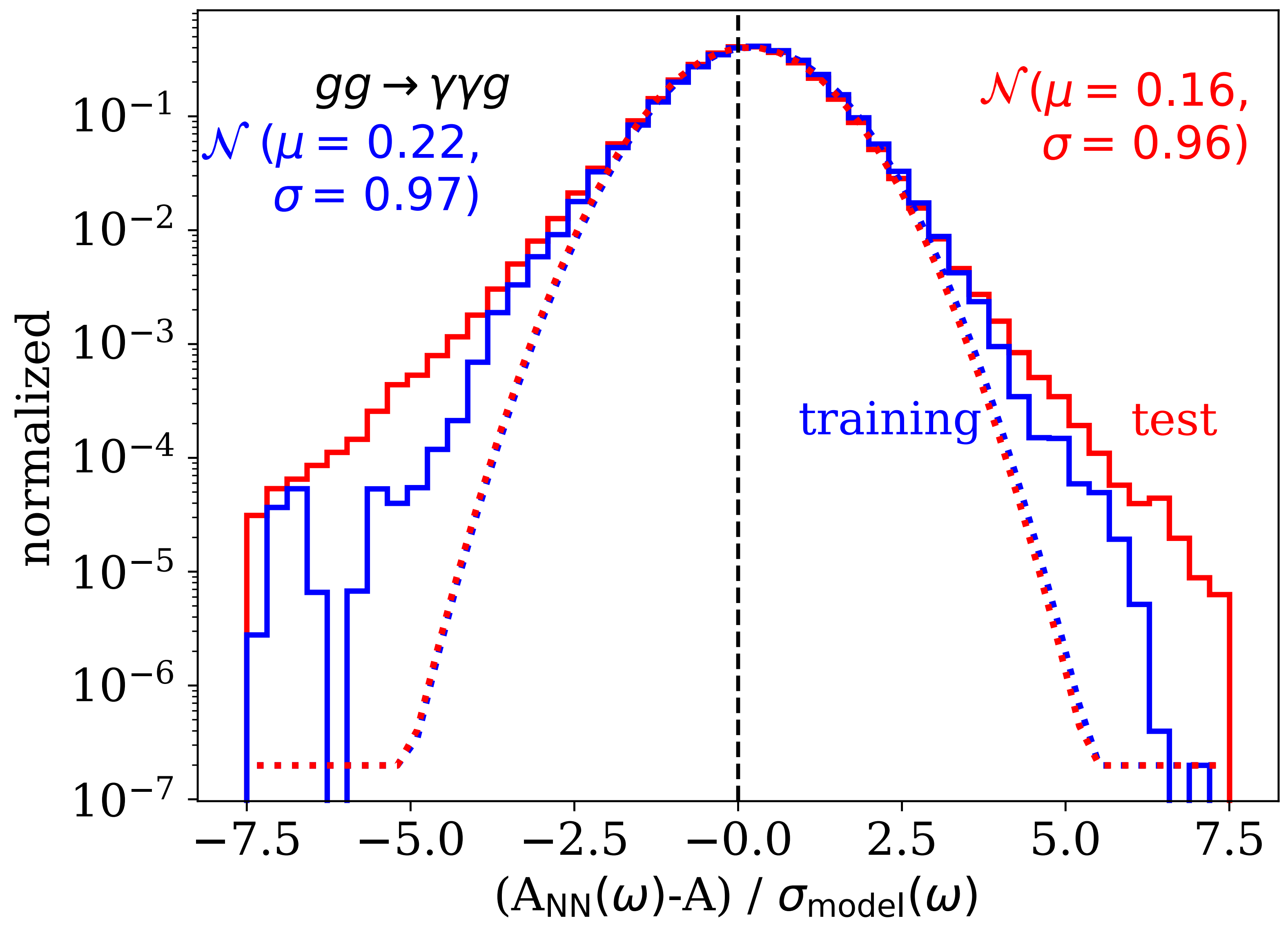
90k training amplitudes
870k test amplitudes

+ Deviations at 1 percent level

Precision $\Delta^{(train)} = \frac{A_{NN} - A_{train}}{A_{NN}}$



Calibration $\Delta^{(train)} = \frac{A_{NN} - A_{train}}{\sigma}$



Performance worse for rare points with large amplitudes (collinear)

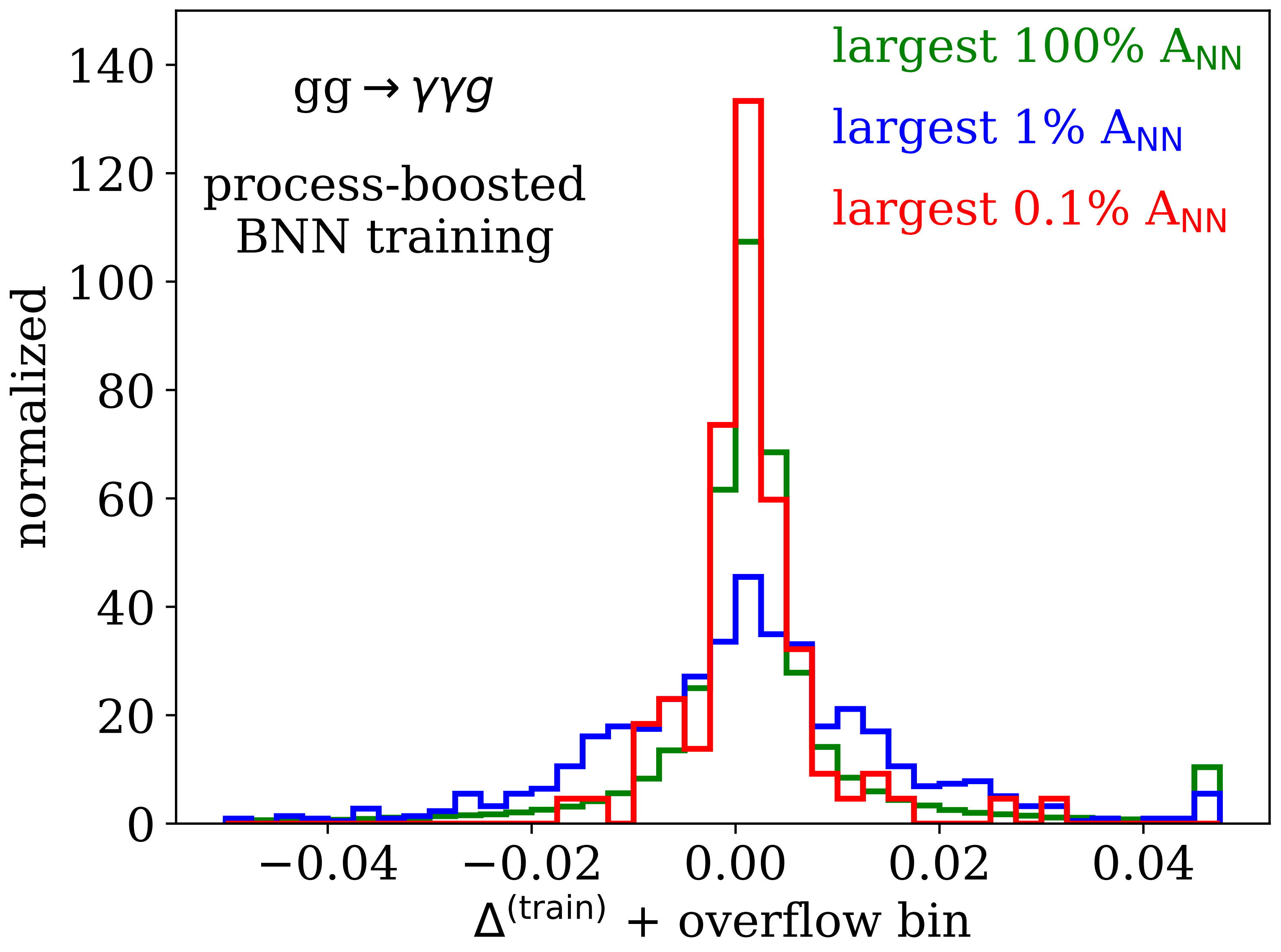
Roughly Gaussian but enhanced tails

Boosting performance & calibration

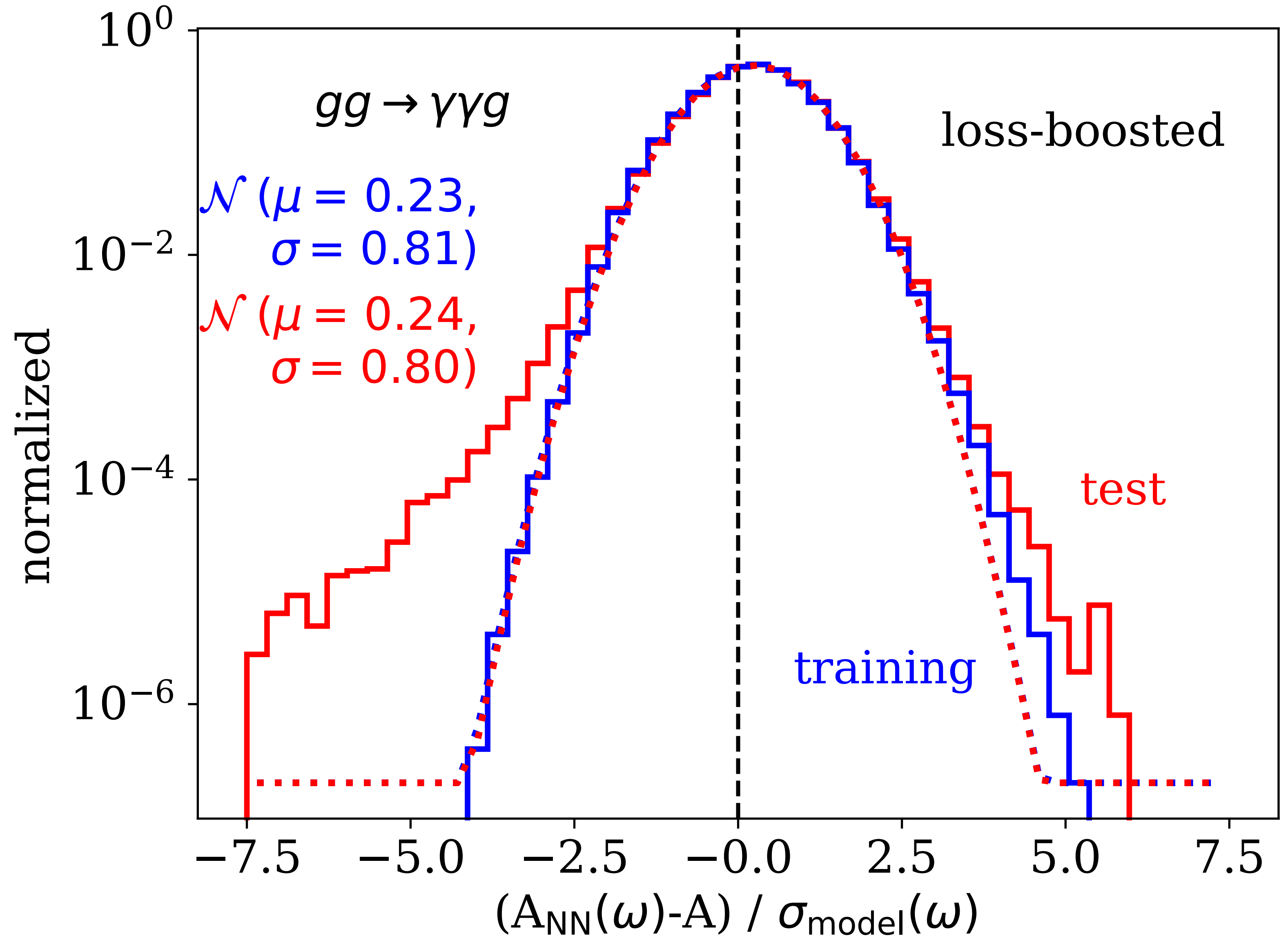
Example
$gg \to \gamma\gamma g(g)$ @LO

90k training amplitudes
870k test amplitudes

Enforce training on samples with
(a) largest σ_{tot} or (b) $\Delta A > 2\sigma$



(a) Significant improvement in performance



(b) Tails reproduced for training data
Improvement for test data

Monte carlo event generation

1. Generate phase space points

$\rightarrow$ set of four-momenta p_i

2. Calculate event weight

$$w_{\text{event}} = f(x_1, Q^2) f(x_1, Q^2) \times \mathscr{M}(x_1, x_2, p_1, \ldots, p_n) \times J(p_i(r))$$

PDF — Matrix element — Phase space mapping

3. Unweighting

keep events with $\dfrac{w_i}{w_{\text{max}}} > r \in [0,1]$

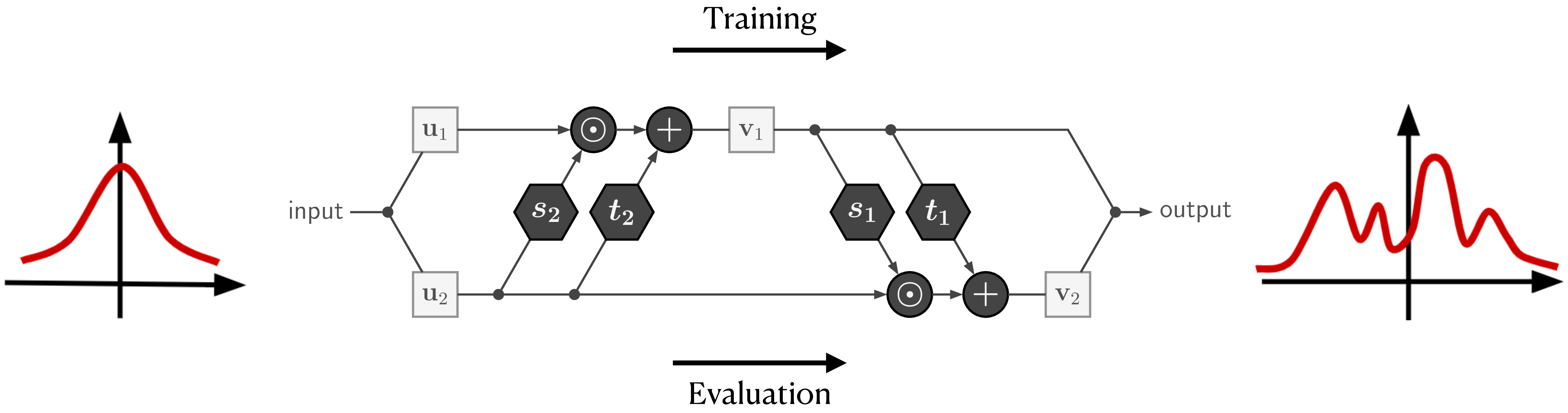
Bottlenecks

1. Slow **matrix element** calculation
   - Complexity grows exponentially with
     - # final state particles
     - Precision (LO, NLO, NNLO, ...)
2. Low **unweighting** efficiency
   - Discard most events if $w_i \ll w_{\text{max}}$
   - Optimize phase space mapping
     ➡ $J(p_i(r)) = (f \times \mathscr{M})^{-1}$

Normalizing flows

Invertible networks for complex transformations

+ Bijective mapping
+ Tractable Jacobian → $p_x(x) = p_z(z) \cdot J_{NN}$
+ INN → flow with fast evaluation in both direction



Training on density $t(x)$

→ Minimize difference

$$\mathscr{L} = \log p_x(x)/t(x)$$
$$= \log p_z(z(x))\, J_{NN} \,/\, t(x)$$

Requires evaluation of $t(x)$

Training on samples x

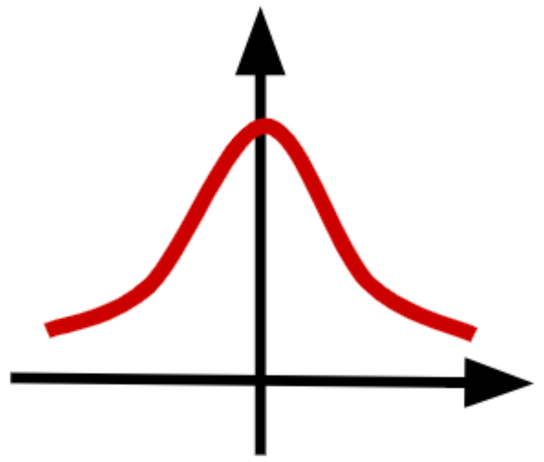
→ Maximize the log-likelihood

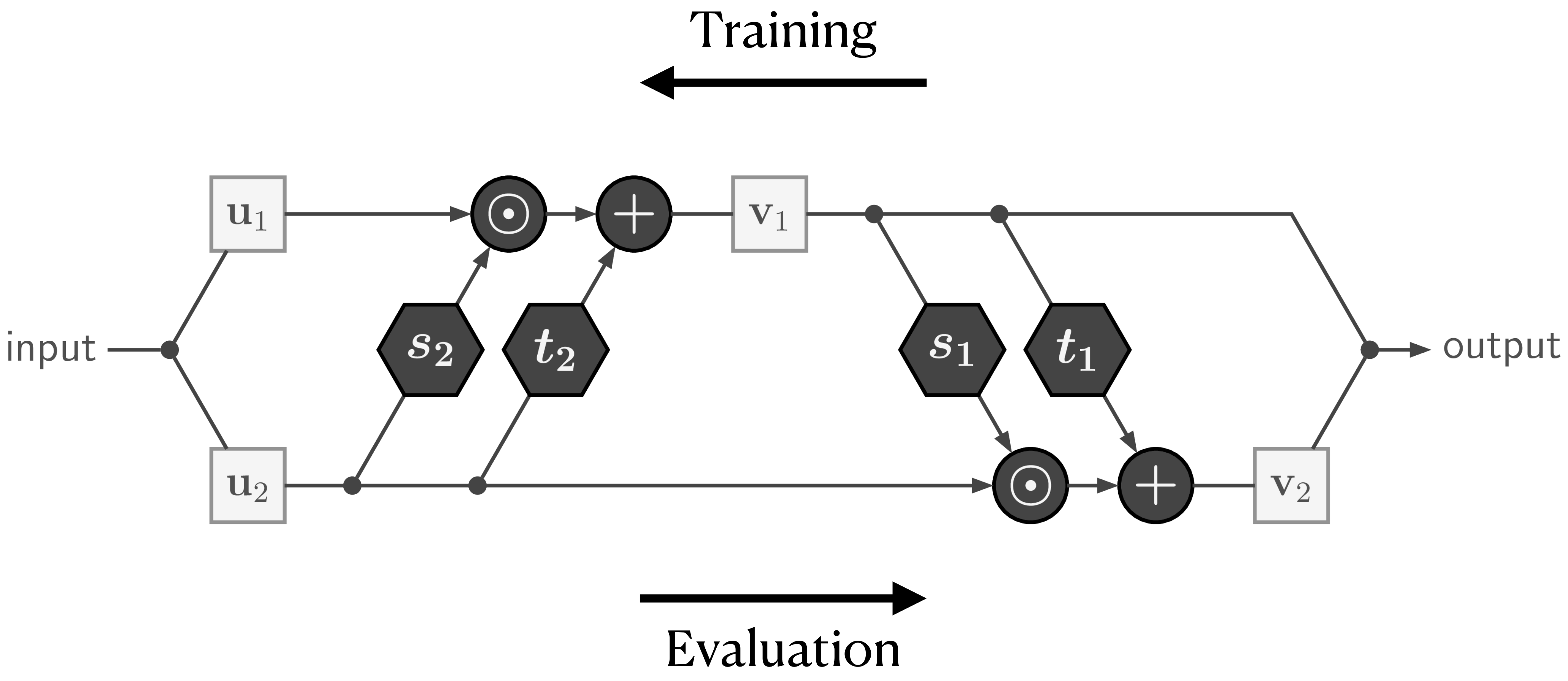
$$\mathscr{L} = \log p(\theta\,|\,x)$$
$$= \log p(x\,|\,\theta) + \log p(\theta) + \text{const}$$
$$= \log p(z\,|\,\theta) + \log J_{NN} + p(\theta) + \text{const}$$

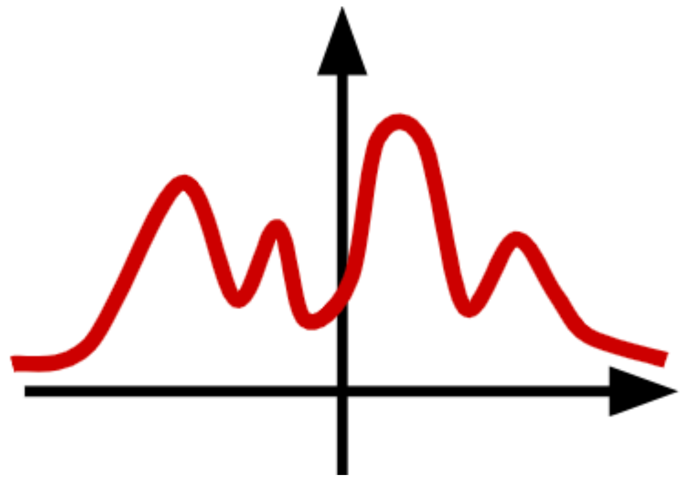
Normalizing flows

Invertible networks for complex transformations

+ Bijective mapping
+ Tractable Jacobian $\rightarrow p_x(x) = p_z(z) \cdot J_{NN}$
+ Fast evaluation in both direction







Training on density $t(x)$
$\rightarrow$ Minimize difference

$$\mathcal{L} = \log p_x(x)/t(x)$$
$$= \log p_z(z(x))\, J_{NN} \,/\, t(x)$$

Requires evaluation of $t(x)$

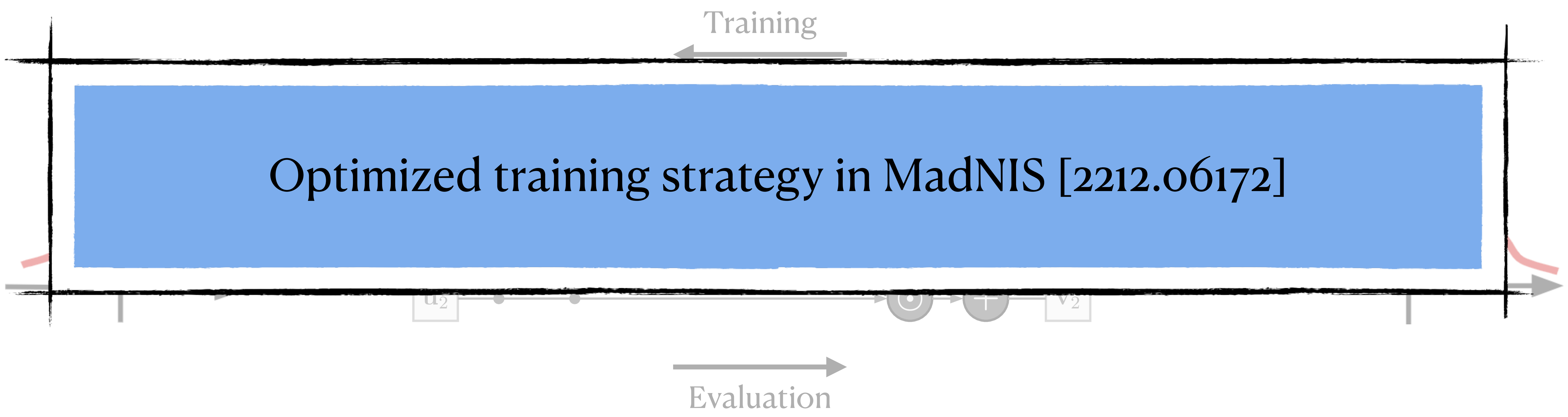
Training on samples x
$\rightarrow$ Maximize the log-likelihood

$$\mathcal{L} = \log p(\theta\,|\,x)$$
$$= \log p(x\,|\,\theta) + \log p(\theta) + \text{const}$$
$$= \log p(z\,|\,\theta) + \log J_{NN} + p(\theta) + \text{const}$$

Normalizing flows

Invertible networks for complex transformations

+ Bijective mapping
+ Tractable Jacobian → $p_x(x) = p_z(z) \cdot J_{NN}$
+ Fast evaluation in both direction



Optimized training strategy in MadNIS [2212.06172]

Training on density $t(x)$

→ Minimize difference

$$\mathcal{L} = \log p_x(x)/t(x)$$
$$= \log p_z(z(x))\, J_{NN} \,/\, t(x)$$

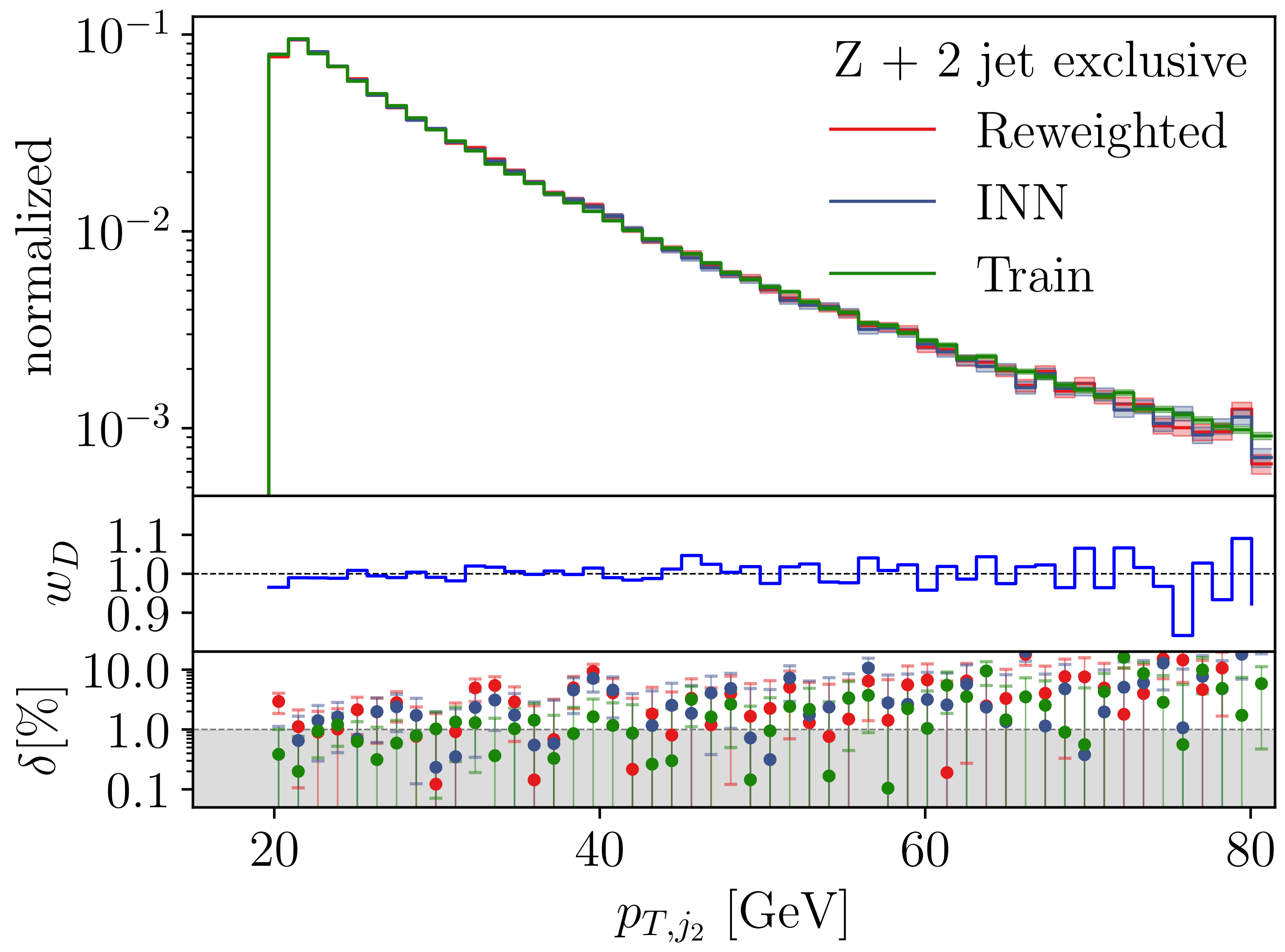
Training on samples x

→ Maximize the log-likelihood

$$\mathcal{L} = \log p(\theta | x)$$
$$= \log p(z | \theta) + \log J_{NN} + p(\theta)$$

Putting flows to work

Event generation

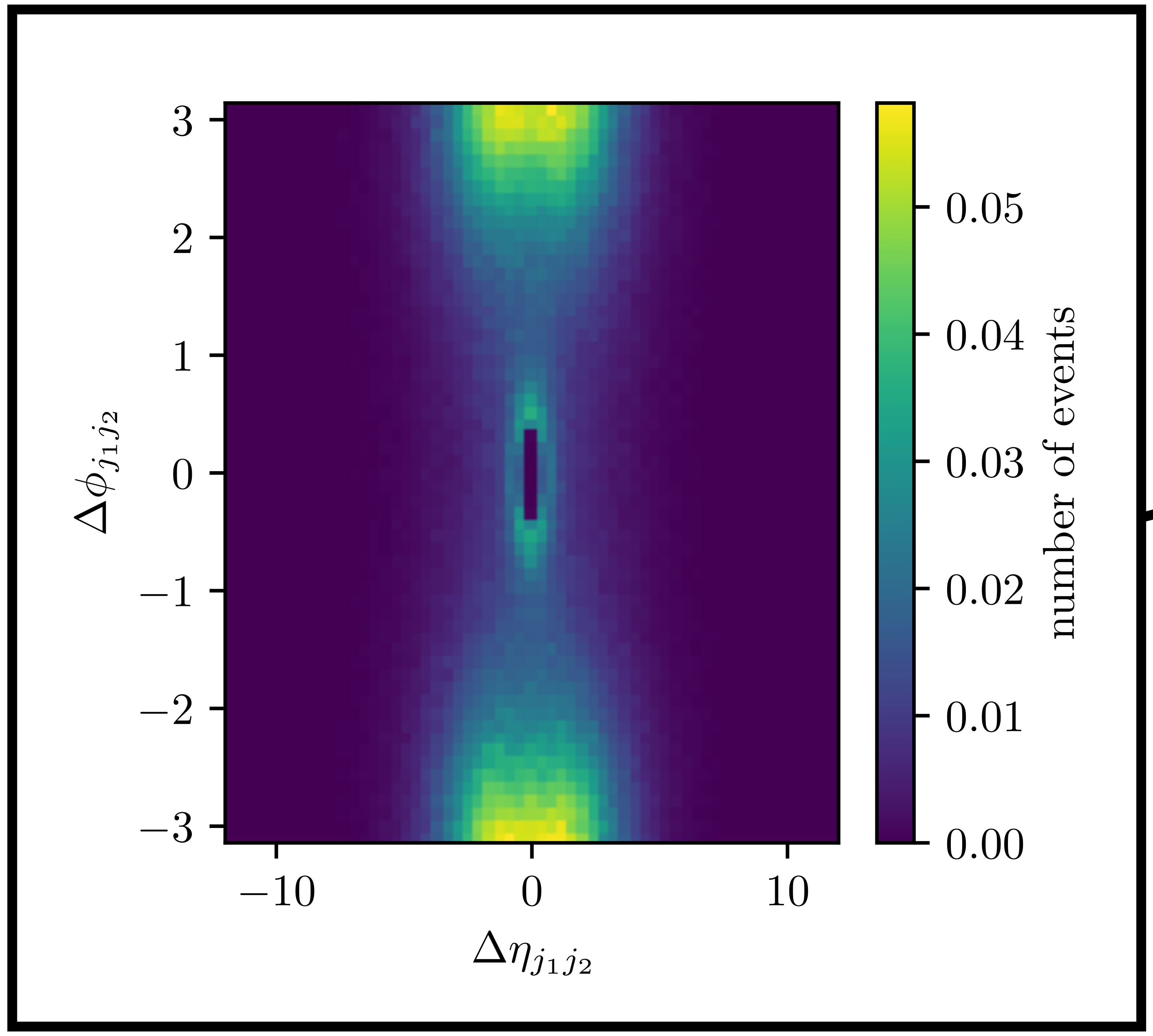


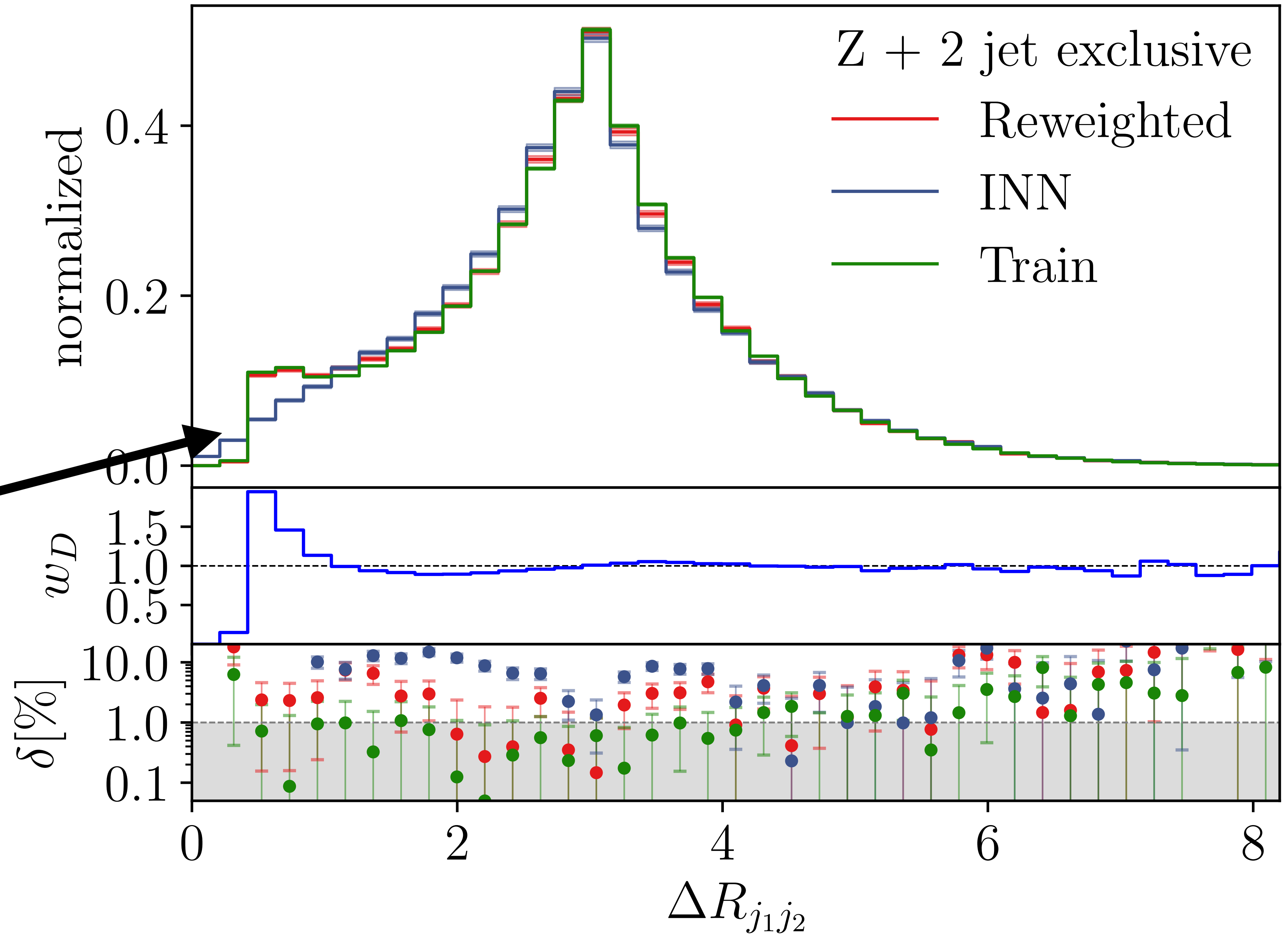
- Train normalizing flow on 4-momenta
- Include symmetries in feature representation
- Excellent performance for direct output

- Extend setup vor variable jet multiplicity

Challenges for normalizing flows

- Narrow features
- Topological holes (eg ΔR cuts)
  - no bijecive mapping possible
  - can only be approximated





Reweighting for Precision

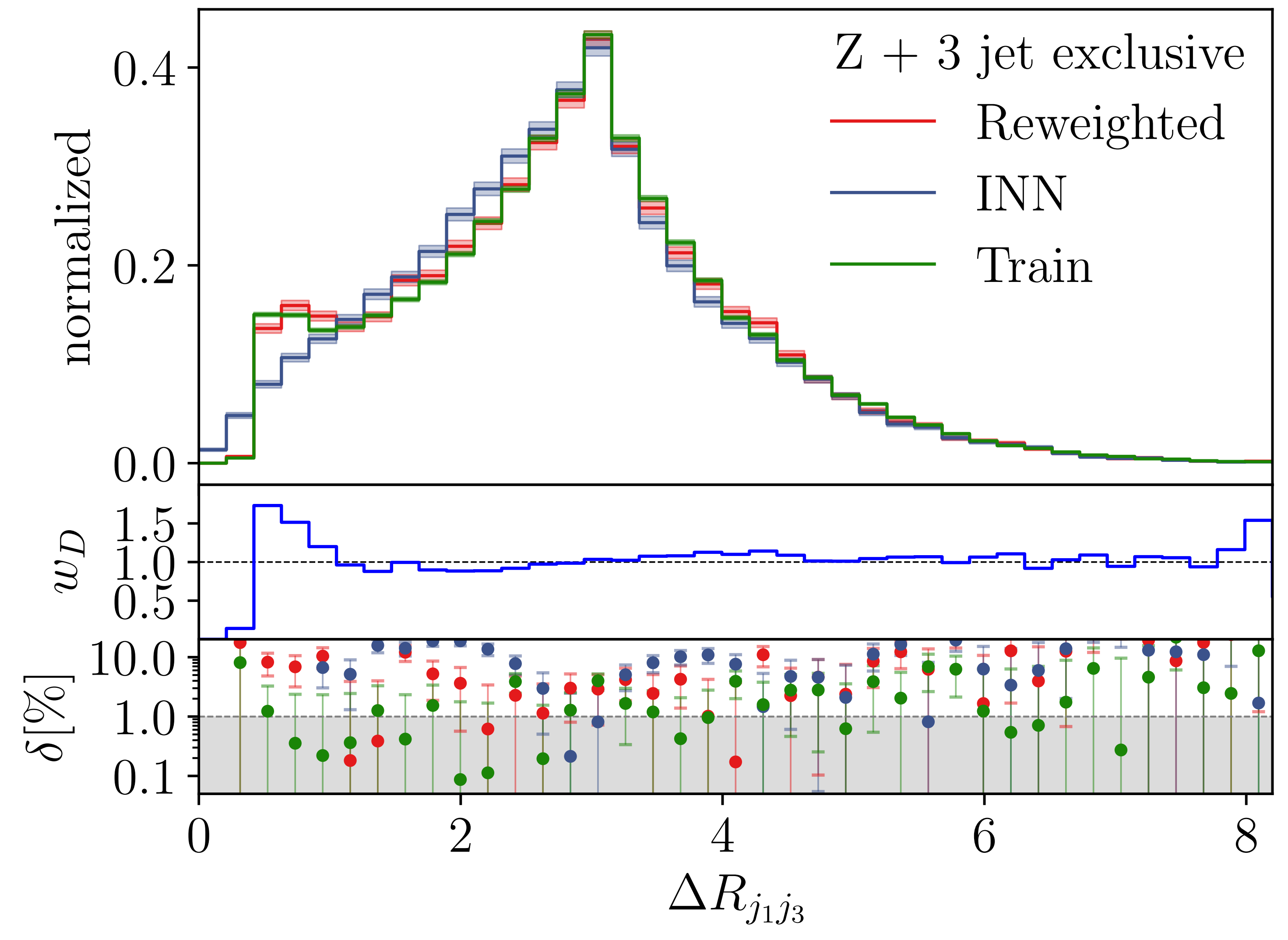
- Classifier loss

$$\mathscr{L} = -\sum_{x\sim p_{data}} \log(D(x)) - \sum_{x\sim p_{INN}} \log(1 - D(x))$$

$$= -\int \mathrm{d}x\, p_{data}(x)\, \log(D(x)) + p_{INN}(x)\, \log(1 - D(x))$$

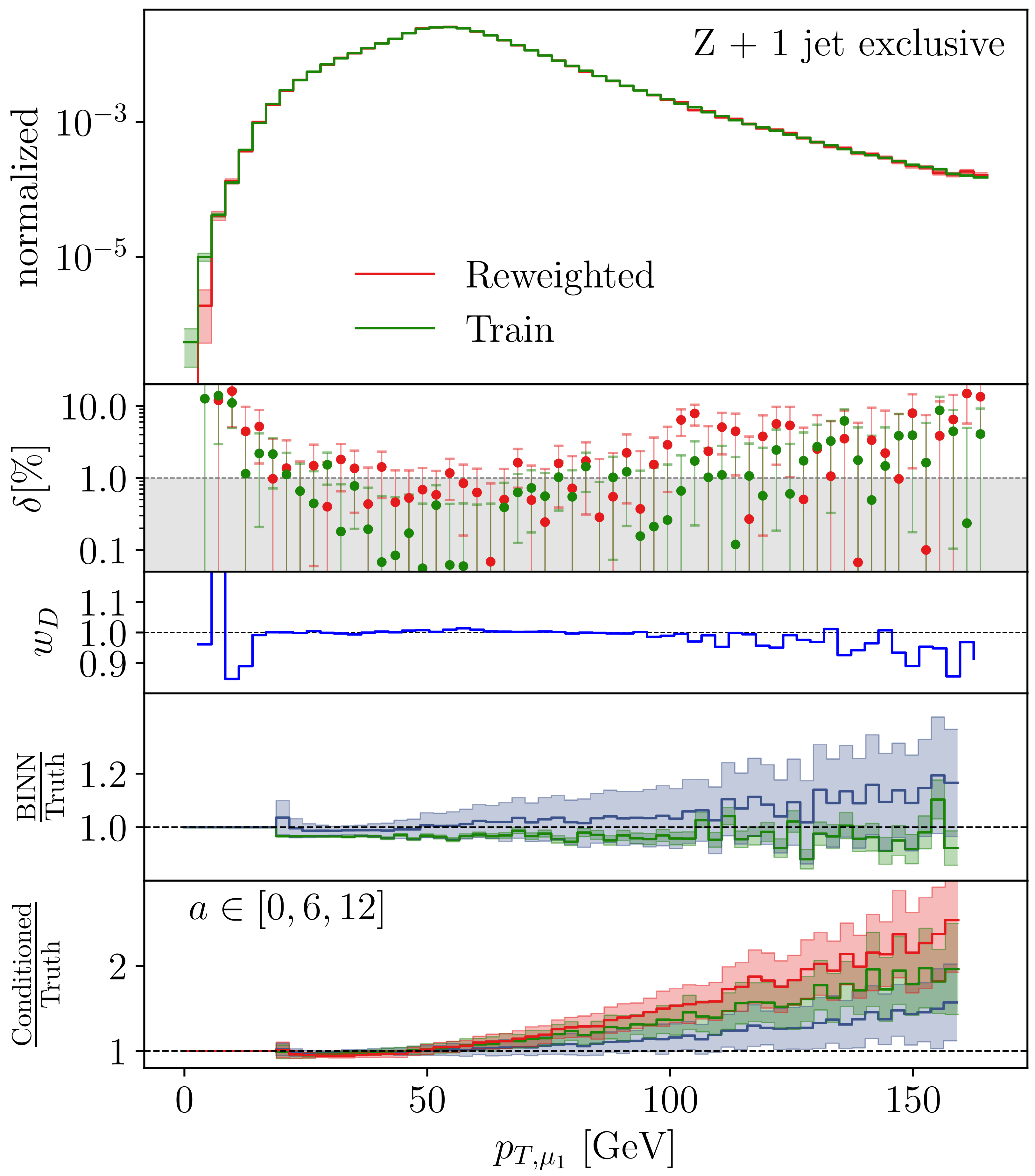
- Upon convergence obtain **reweighting factor**

$$\Rightarrow \frac{p_{data}(x)}{p_{INN}(x)} = \frac{D(x)}{1 - D(x)} = w_D$$



Putting flows to work

Event generation



- Basis: INN
  - Phase space symmetries in architecture
- Control via classifier D
  - $$\frac{p_{\text{truth}}(x)}{p_{\text{INN}}(x)} = \frac{D(x)}{1 - D(x)}$$
- Precision via reweighting
  - Correct deviations of p_{INN}

➡ Uncertainty estimation via Bayesian NN

➡ Uncertainty propagation via conditioning

How can networks improve predictions?

Amplitude interpolation with uncertainties ✓

Bijective mapping for reparametrization ✓

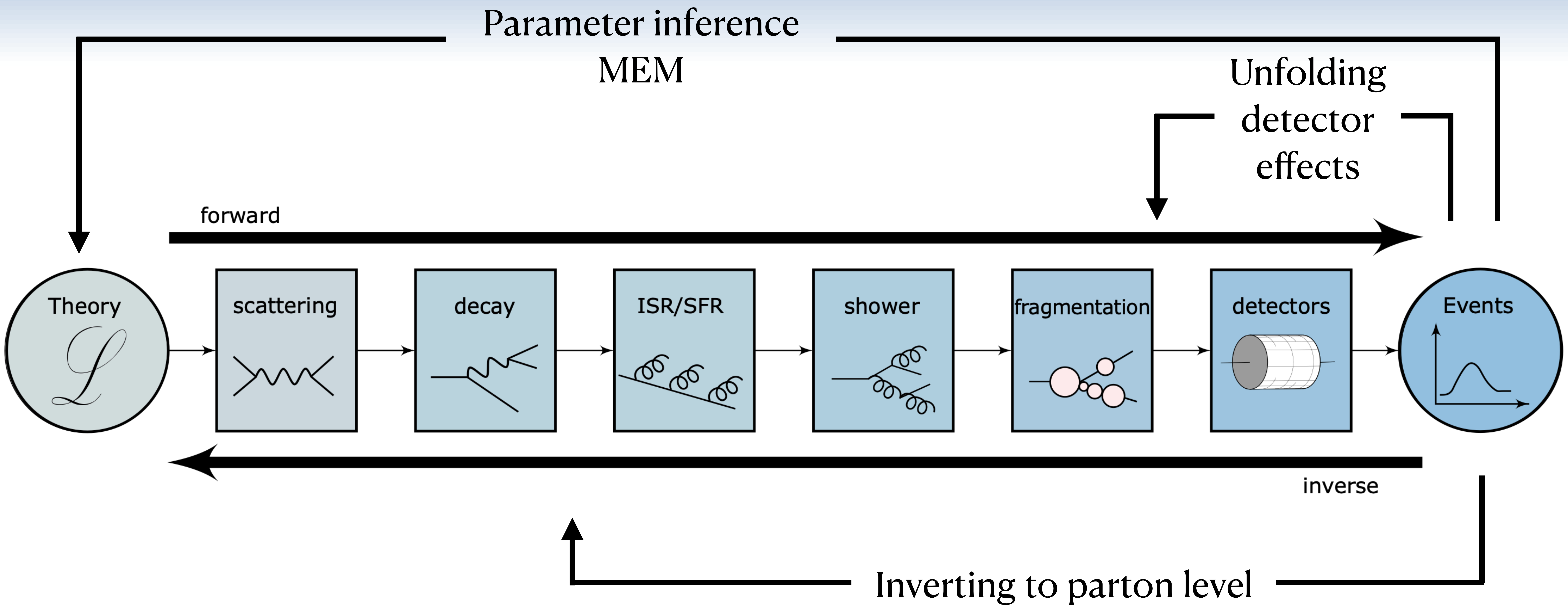
Phase space sampling ✓

$\rightarrow$ Precision $\equiv$ efficiency

Unweighting/ Operations on distribution samples ✓

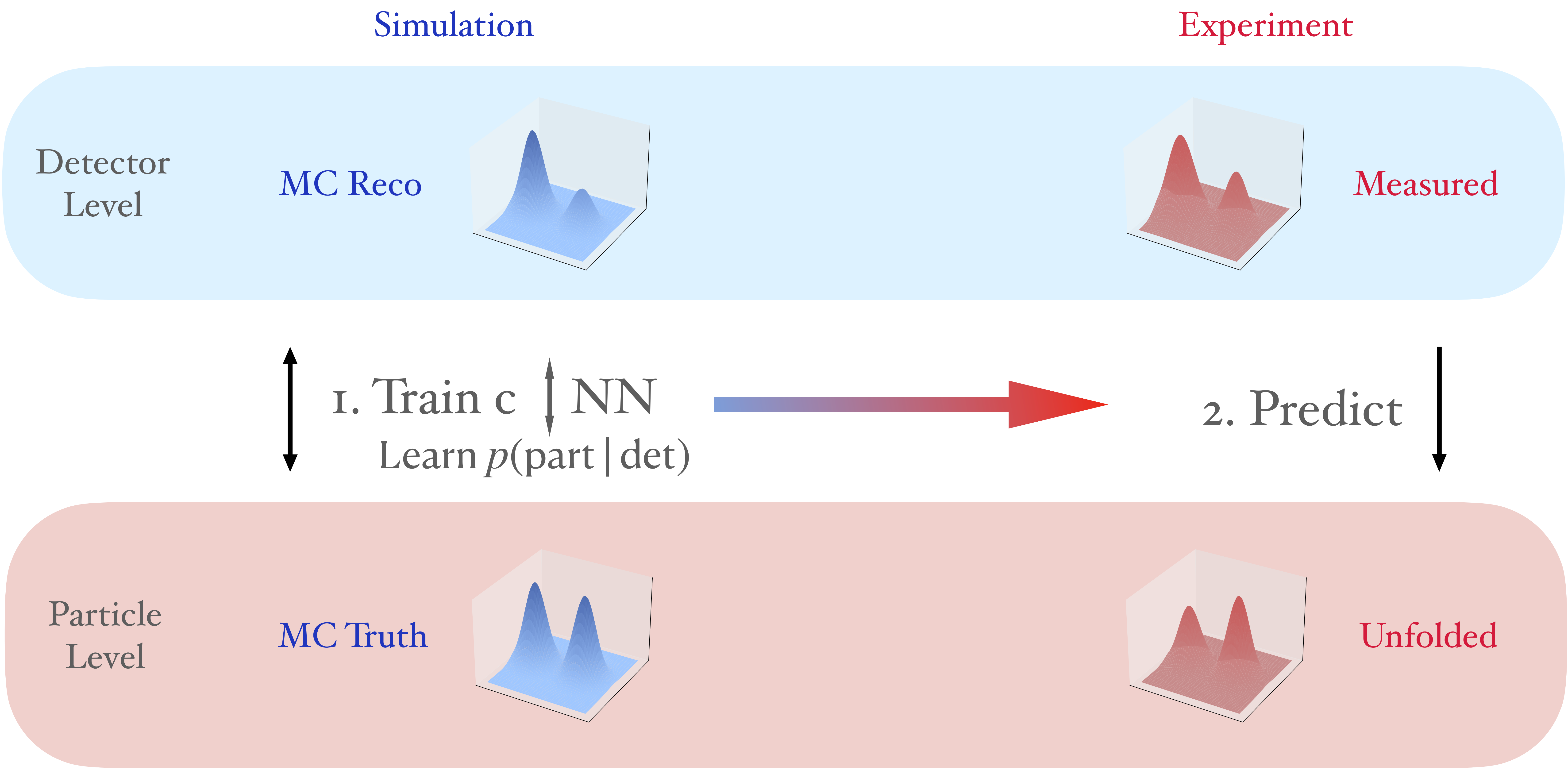
What about inversion?

Inverting the simulation chain



Requirements
- ☐ High - dimensional
- ☐ Bin - independent
- ☐ Statistically well defined

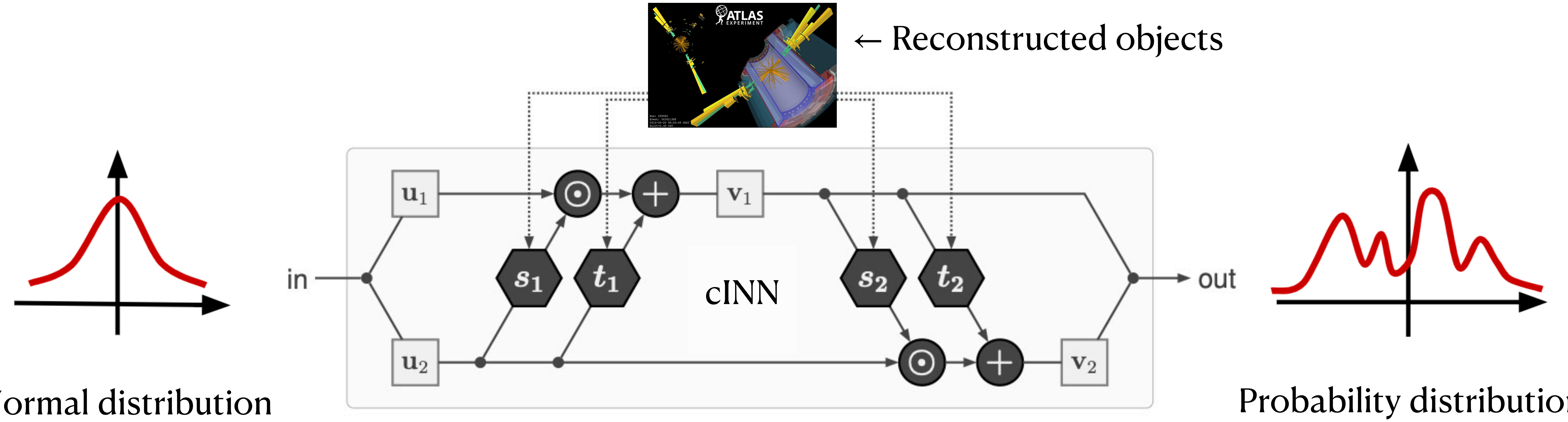
Flow based unfolding methods



cINN unfolding

High-dimensional. Bin independent. Robust.

Given a reconstructed event:
What is the probability distribution at particle level?



Normal distribution

$z \sim \mathcal{N}$

Probability distribution

$x \sim P_{\text{parton}}$

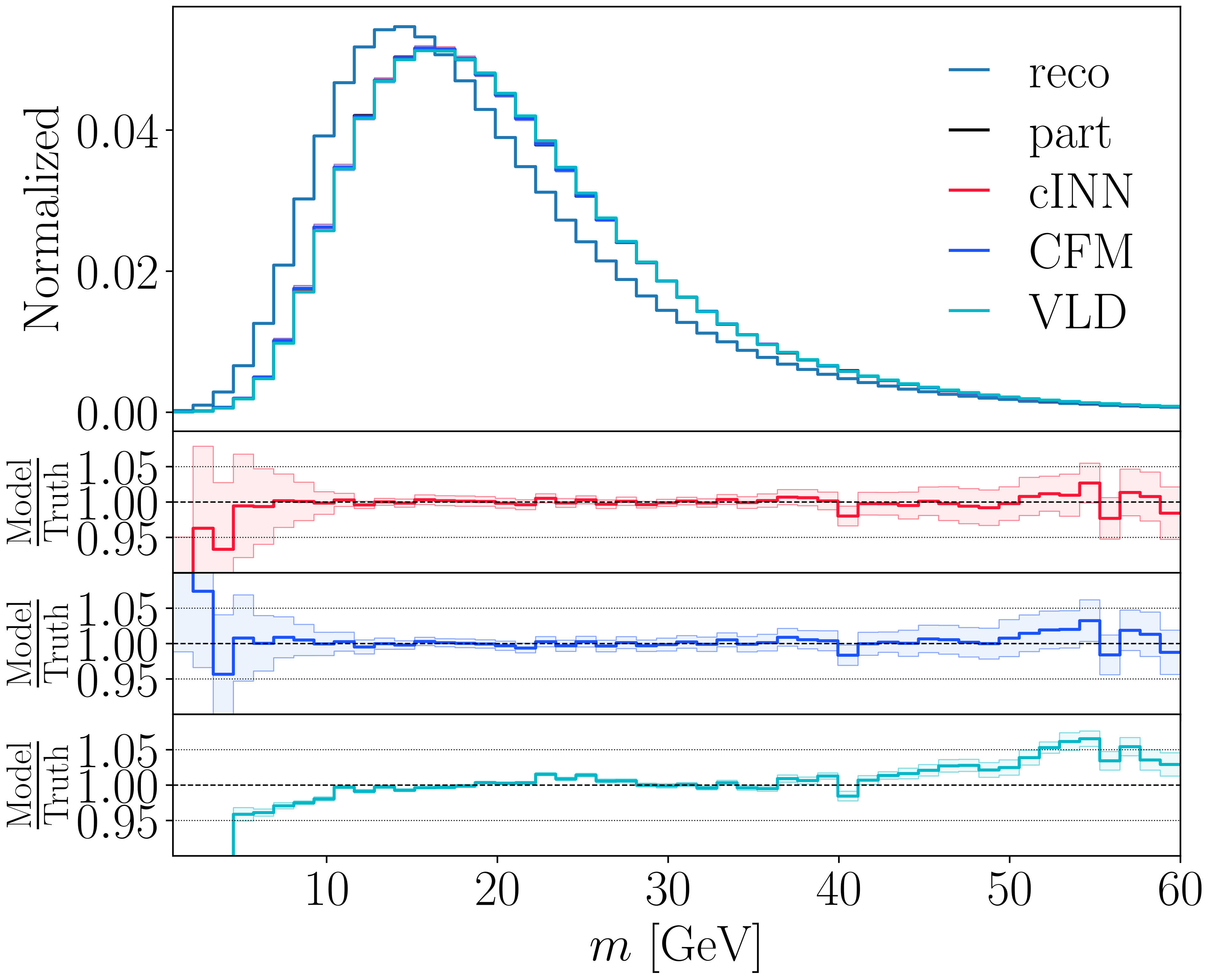
Training

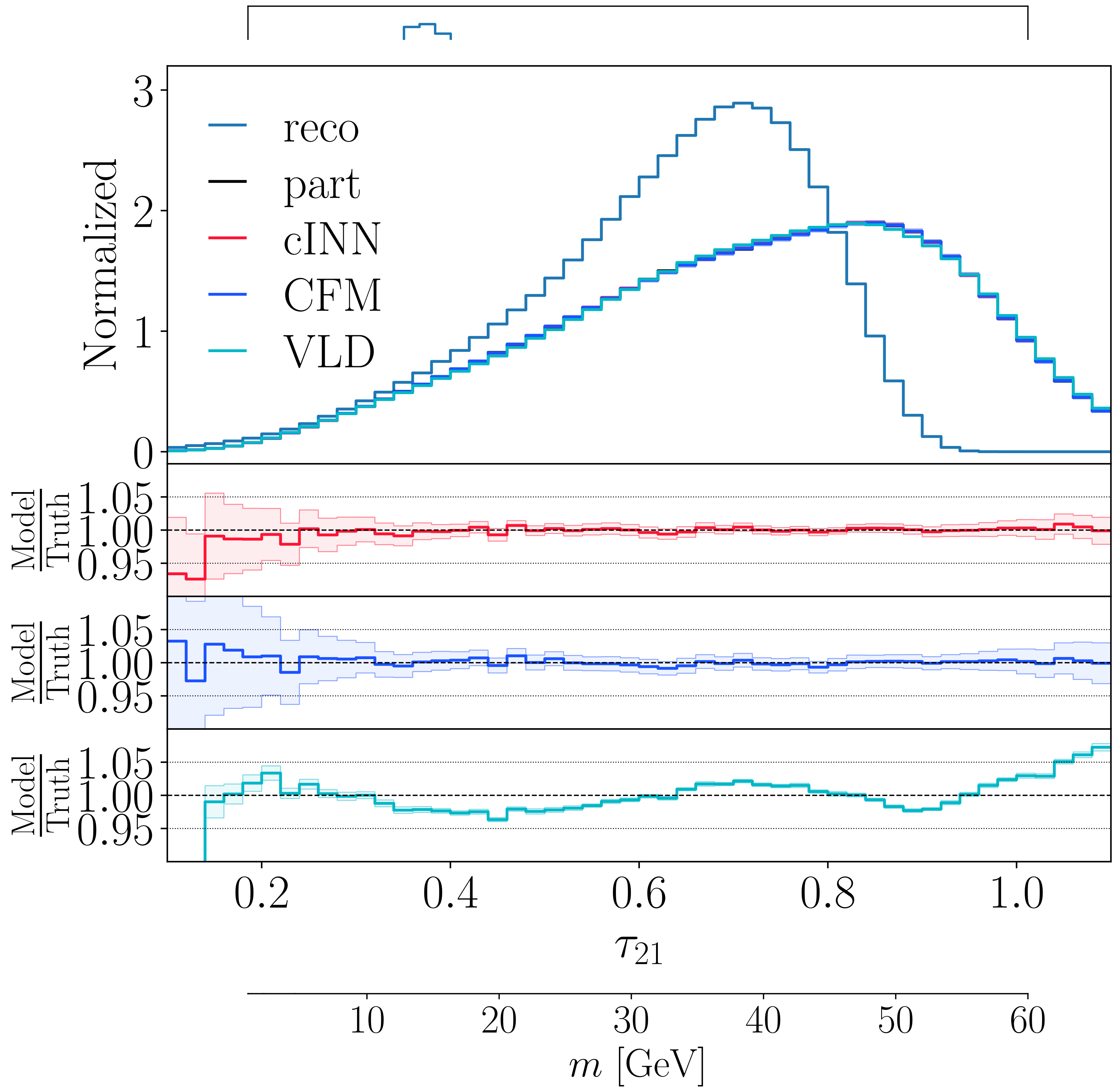
Unfolding

$$\mathcal{L} = \log p(\theta \,|\, x, reco)$$
$$= \log p(z \,|\, \theta, reco) + \log J_{NN} + p(\theta)$$

Unfolding Z+jets events

Observables $m, \tau_{21}, w, N, \log\rho, z_g$

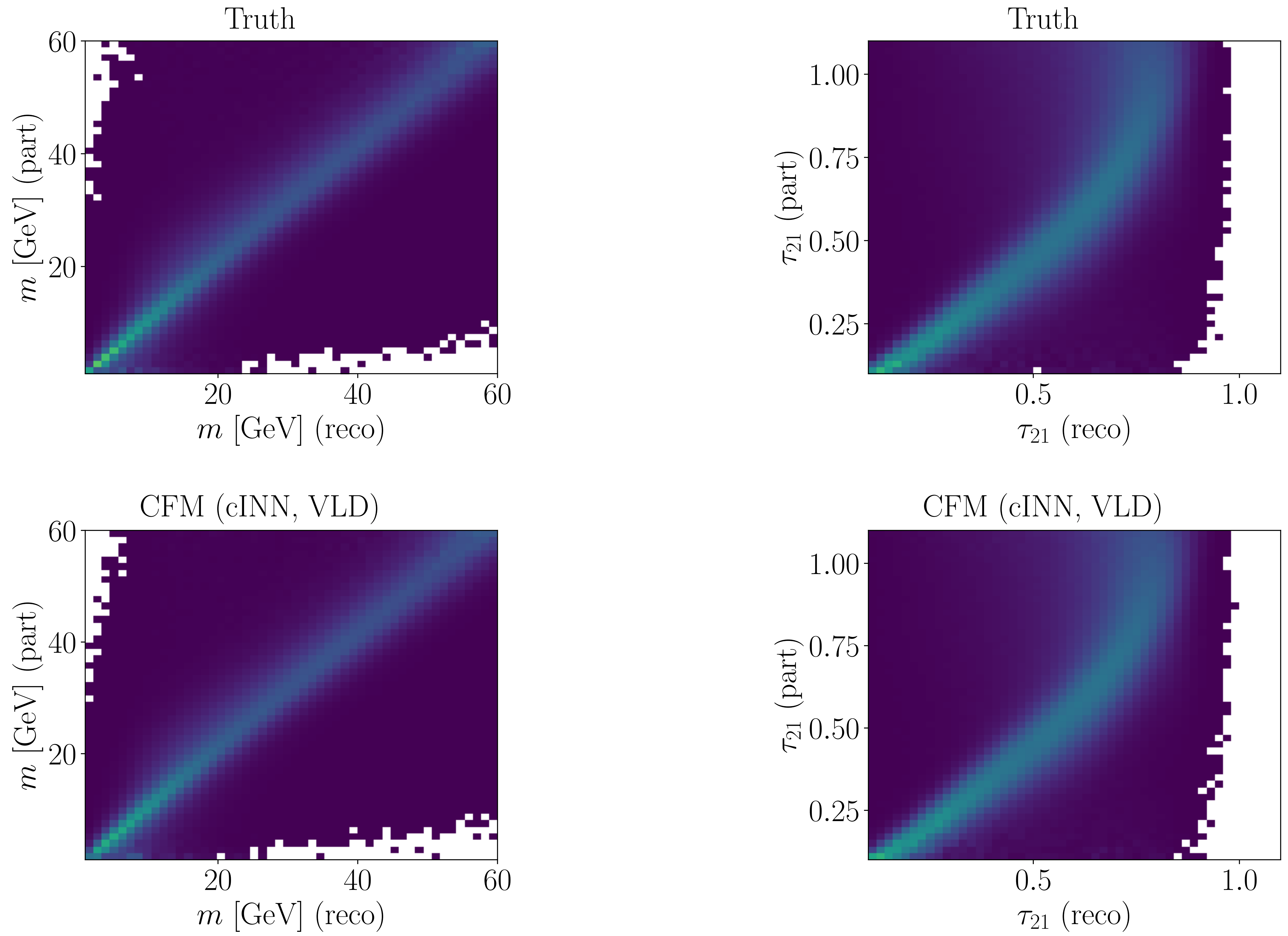




The Landscape of Unfolding with ML **[2404.18807] N. Hütsch, et al.**

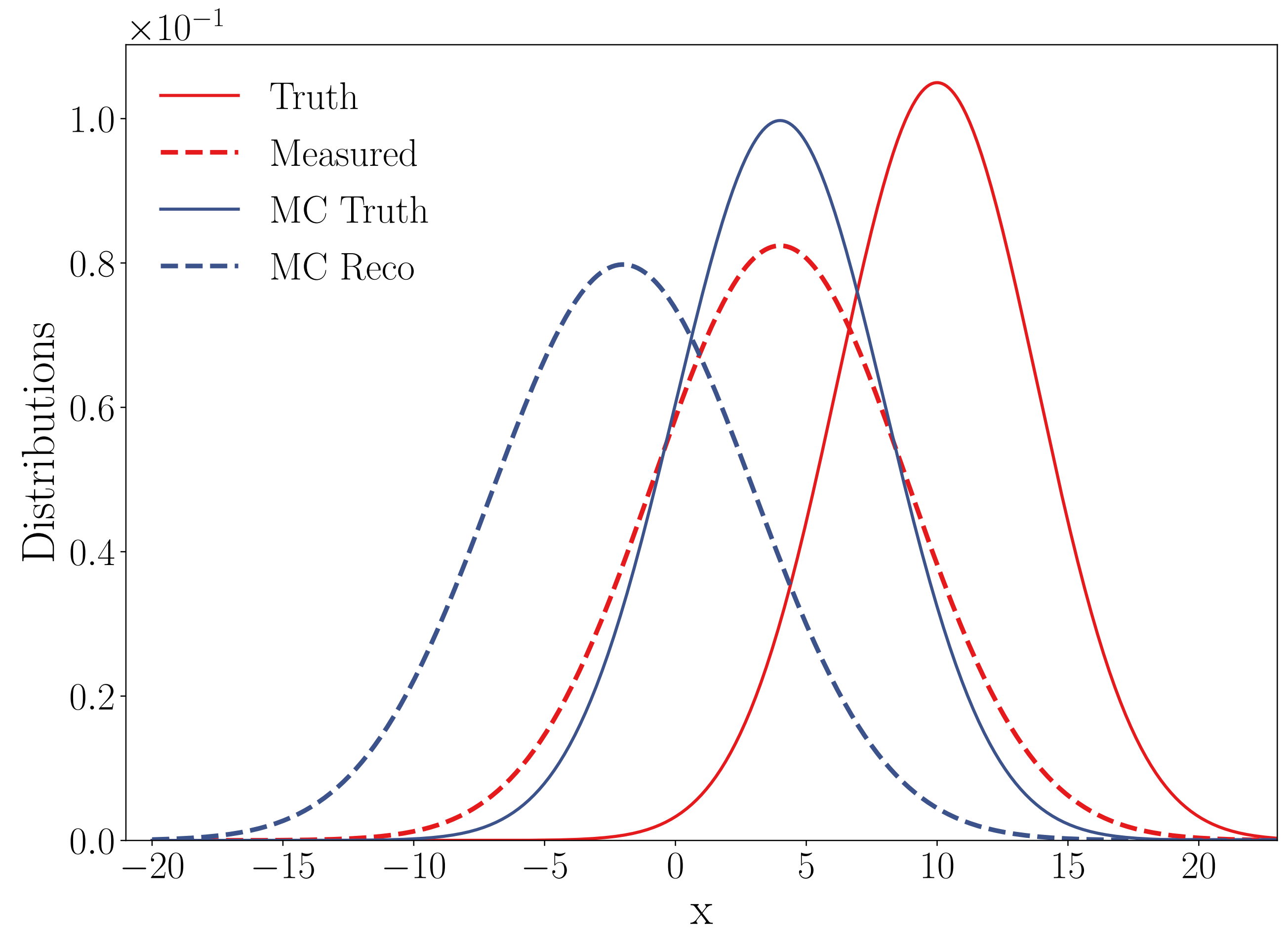
Correlations

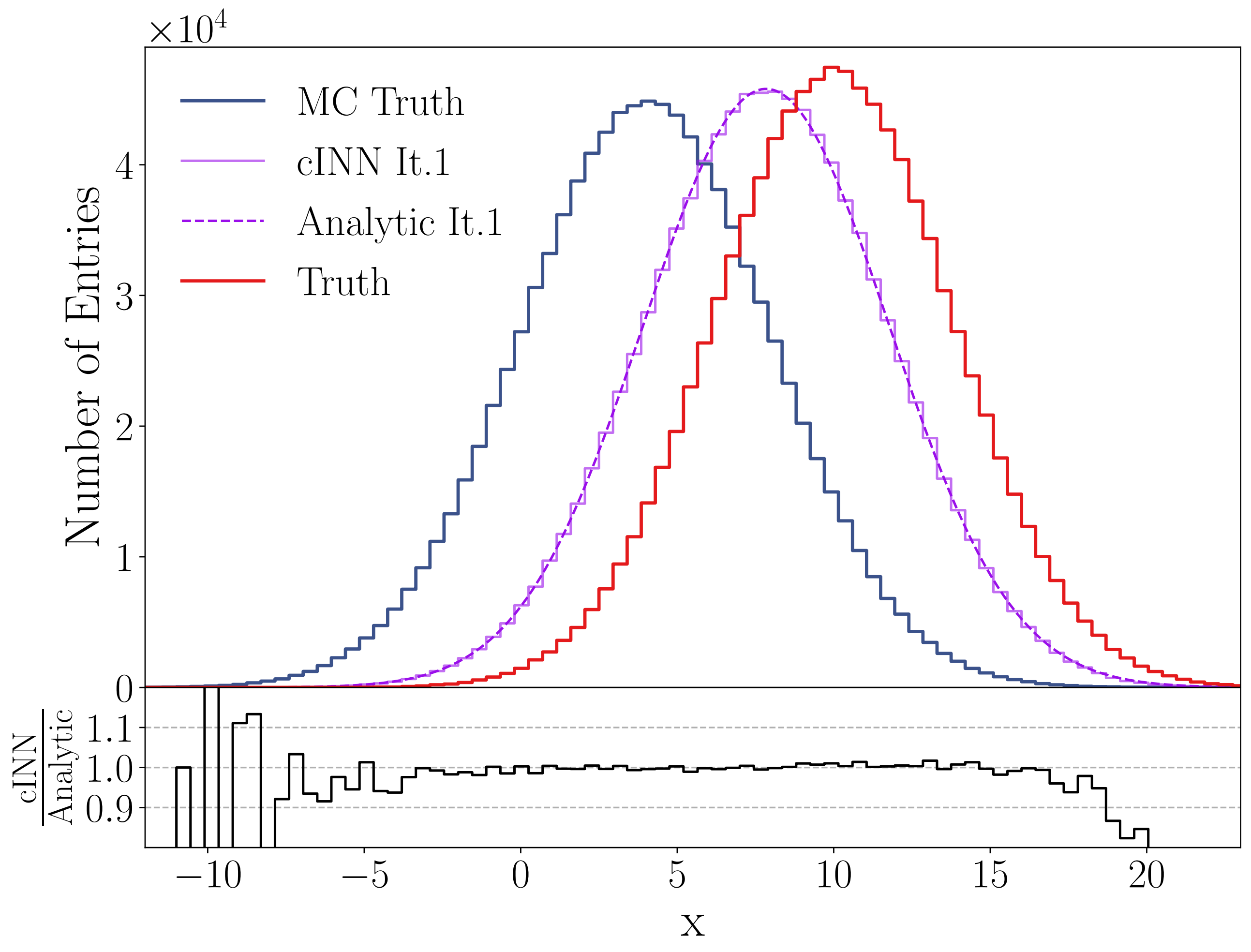
Z+jets: reco vs particle level, jet mass & subjettiness ratio

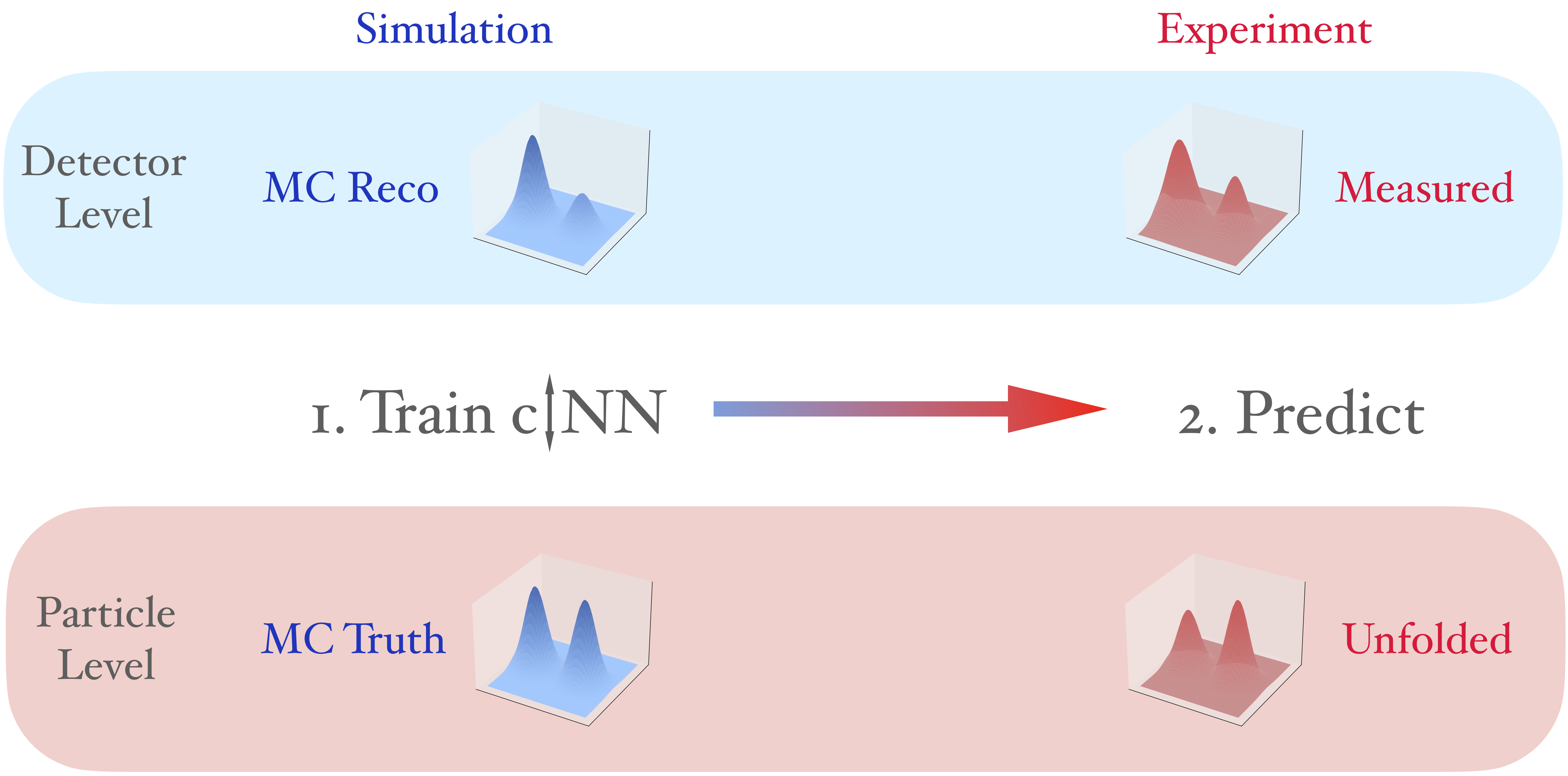


Limitations of direct unfolding with generative NNs

Prior dependence



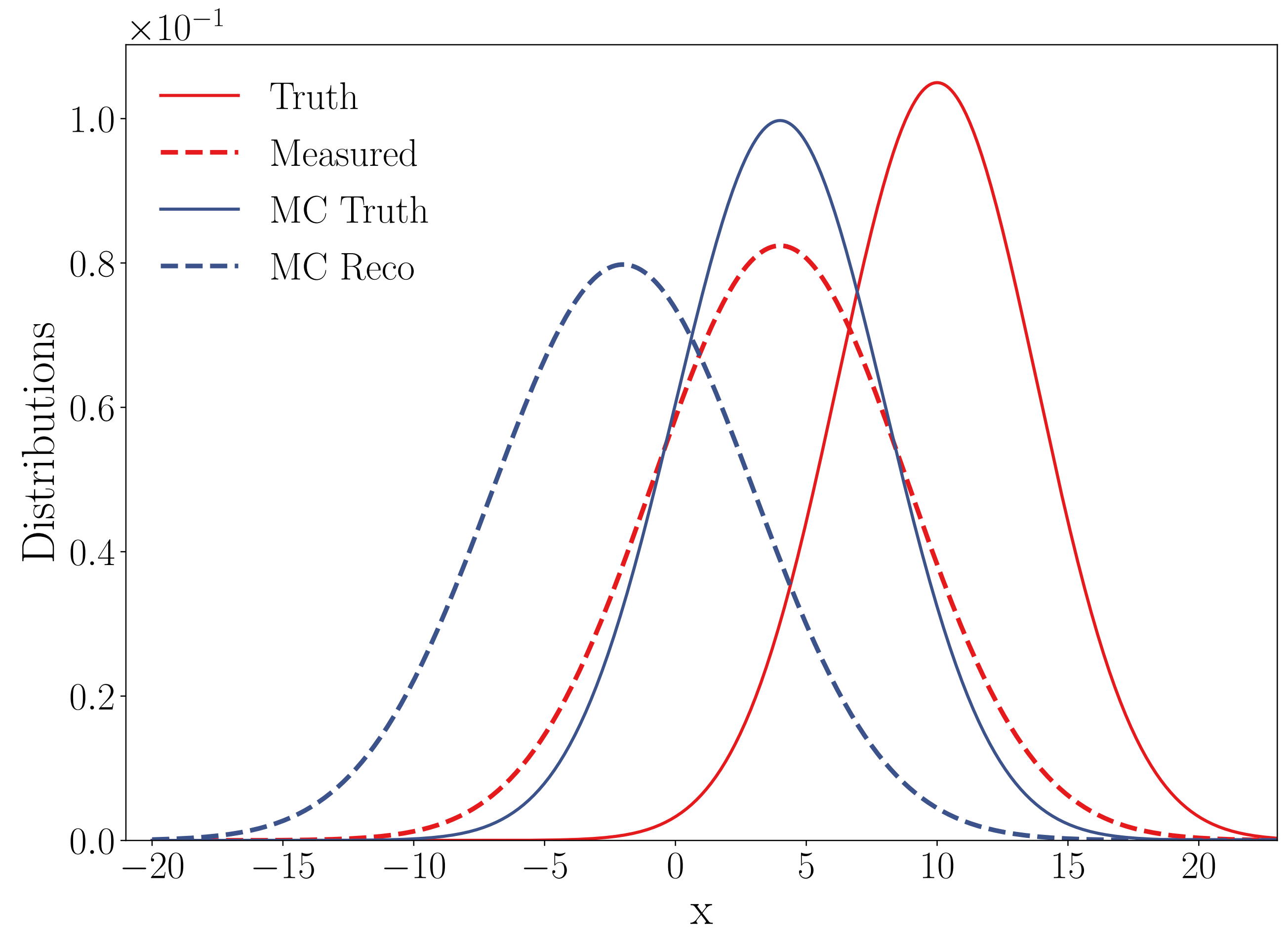


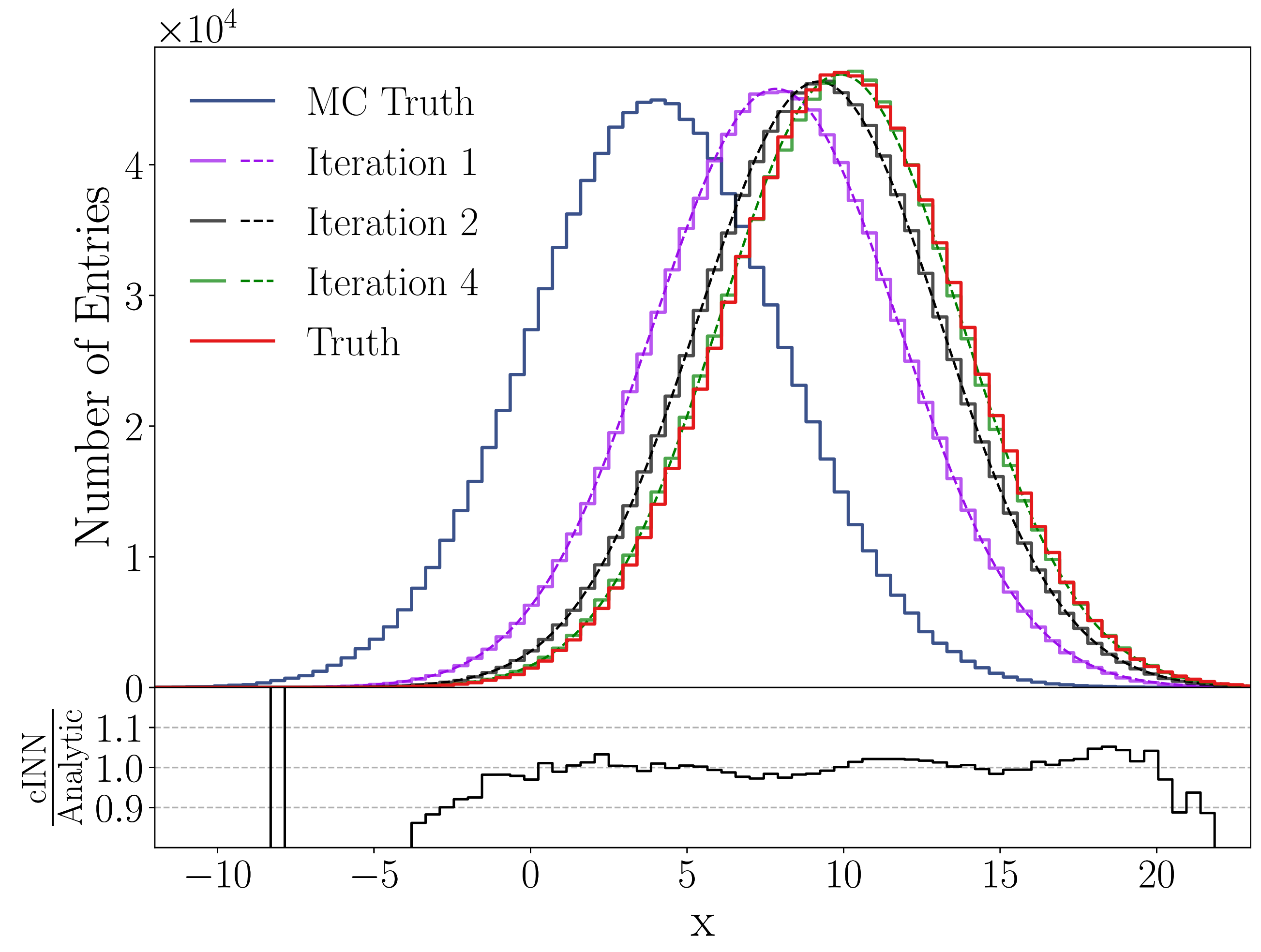


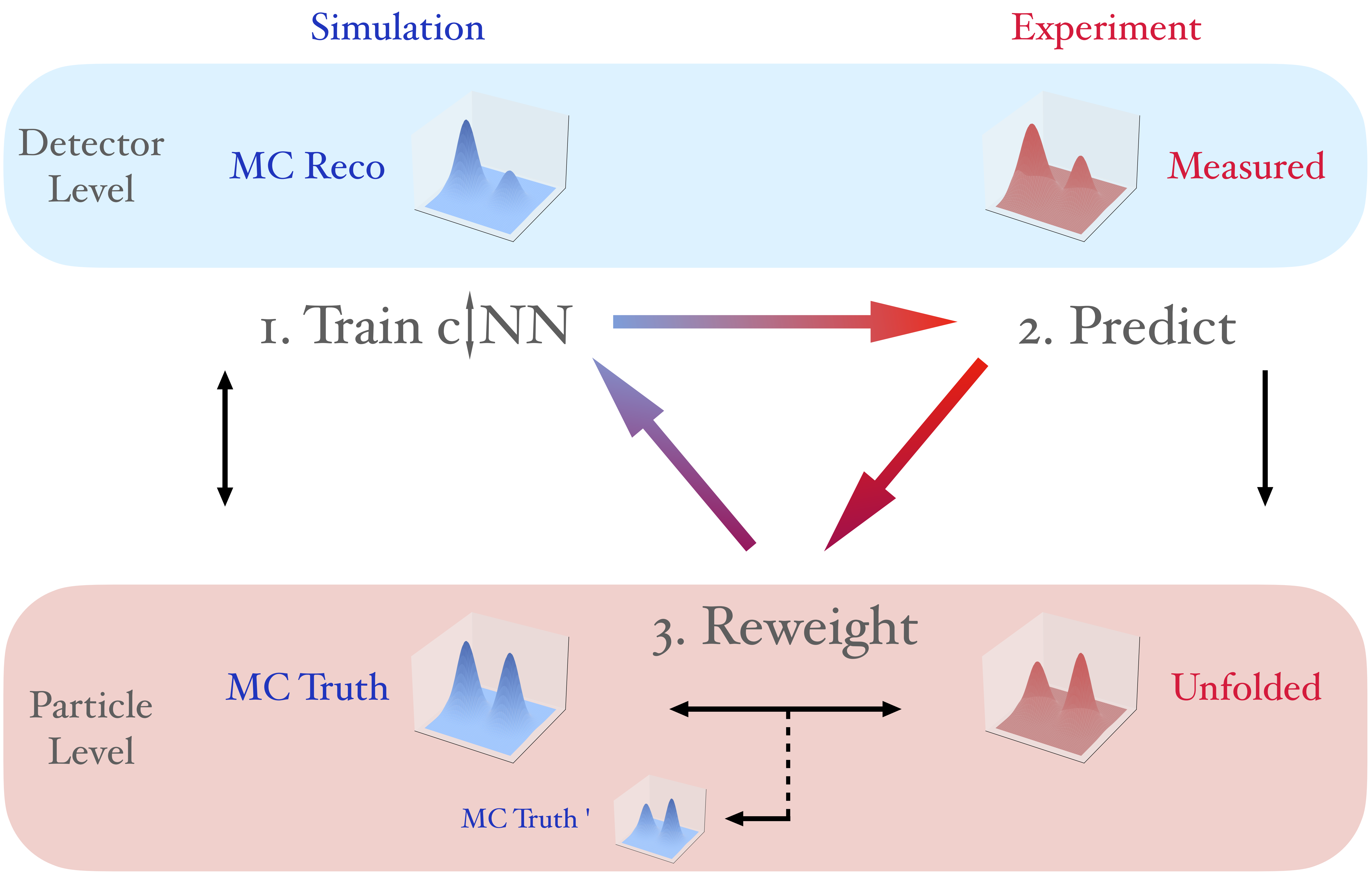
$$p(\text{part}\,|\,\text{det}) = \frac{p(\text{det}\,|\,\text{part}) \cdot p(\text{part})}{p(\text{det})}$$

Conditional iterative unfolding

[] M. Backes, et al.







Beyond unfolding: Enabling the MEM

[2210.00019, 2310.07752]

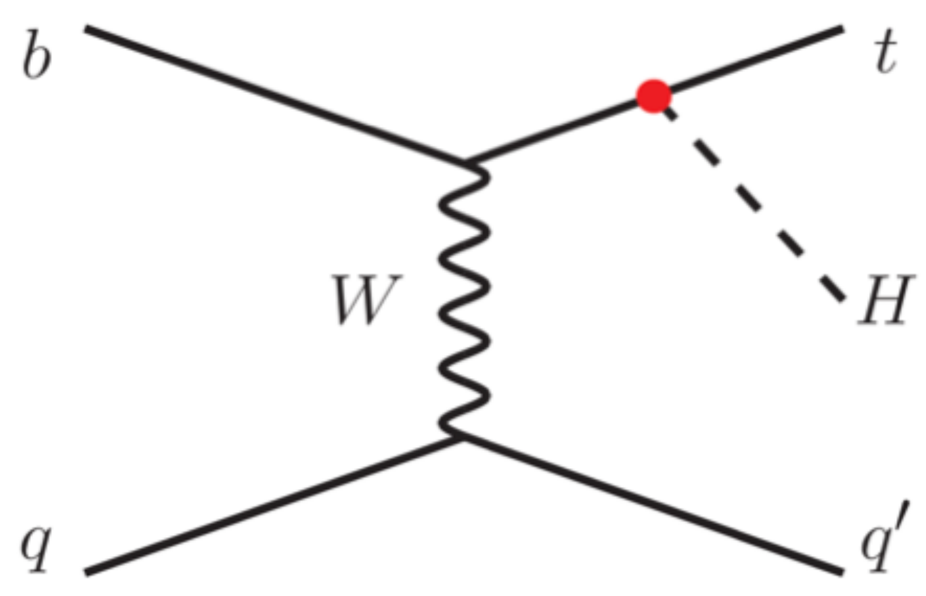
Matrix element method is based on untractable likelihood

$$p(x_{\text{reco}}|\alpha) = \int dx_{\text{hard}} \underbrace{p(x_{\text{hard}}|\alpha)}_{\text{diff. CS}} \underbrace{p(x_{\text{reco}}|x_{\text{hard}}, \alpha)}_{\text{estimate with network}}$$

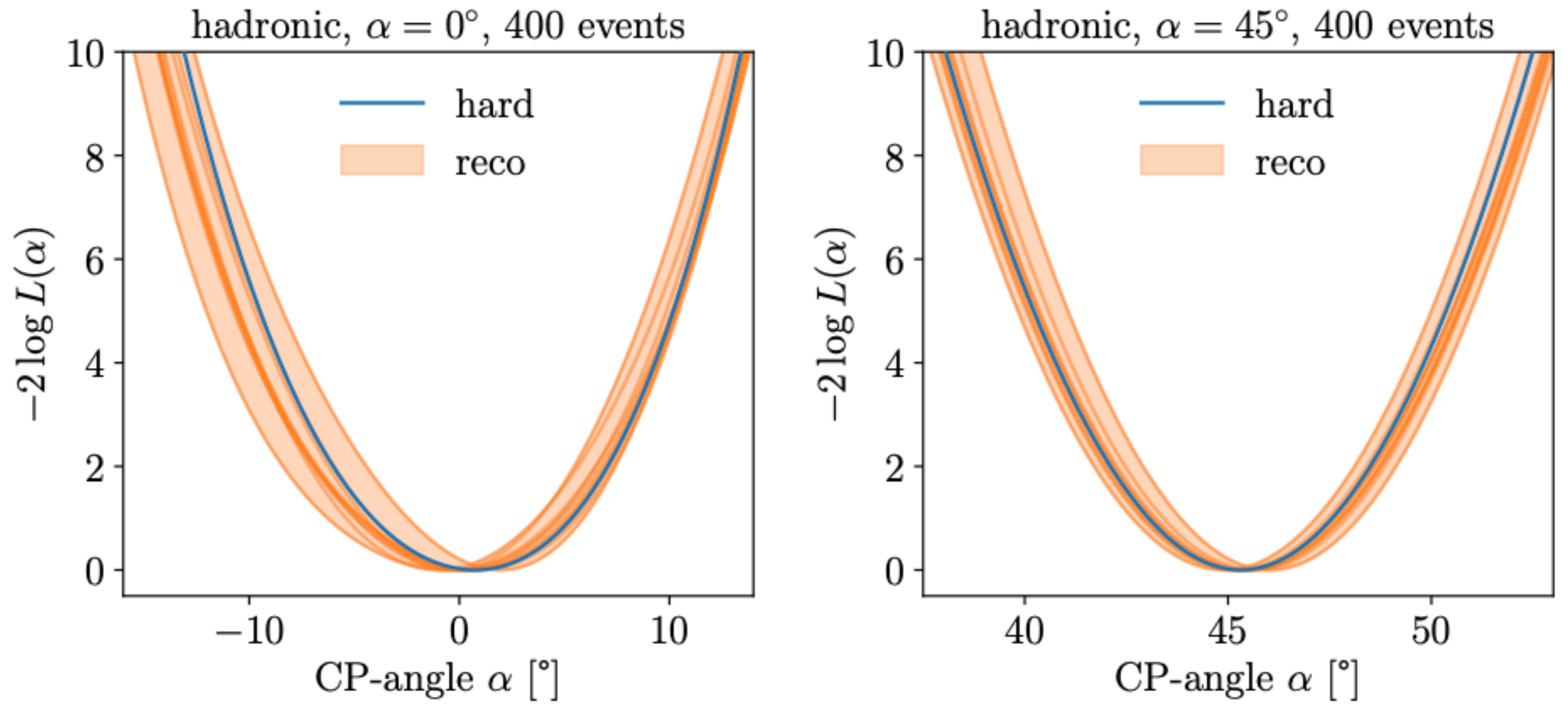
Problem: integration over full phase space of the hard scattering

Solution: Use unfolding cINN to sample x_{hard}

$$p(x_{\text{reco}}|\alpha) = \left\langle \frac{1}{q(x_{\text{hard}})} p(x_{\text{hard}}|\alpha)\, p(x_{\text{reco}}|x_{\text{hard}}, \alpha) \right\rangle_{x_{\text{hard}} \sim q(x_{\text{hard}})}$$

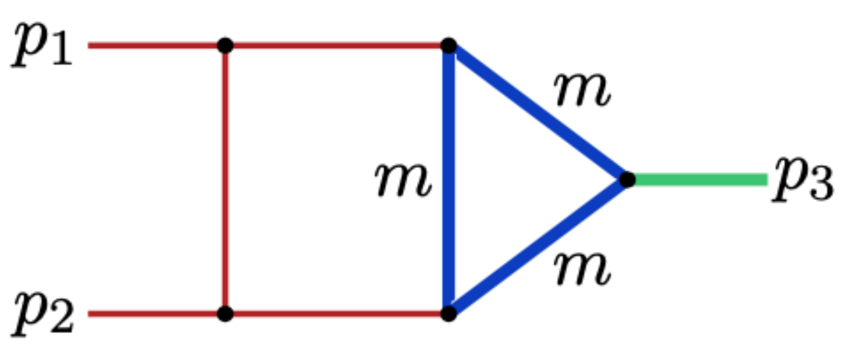


Single Higgs production
with anomalous non-CP conserving Higgs coupling



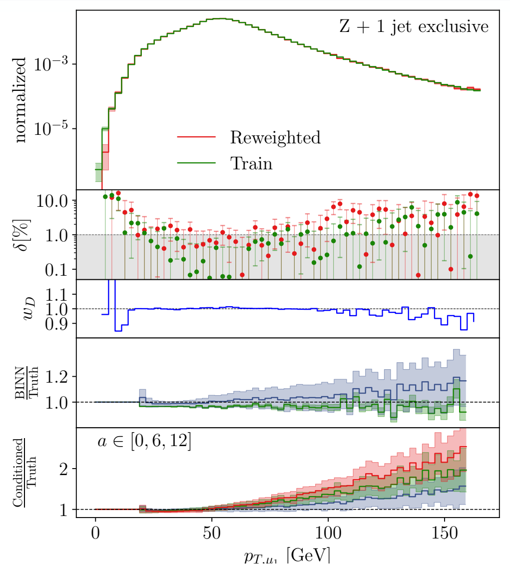
Machine learning up and down the simulation chain

Loop amplitudes



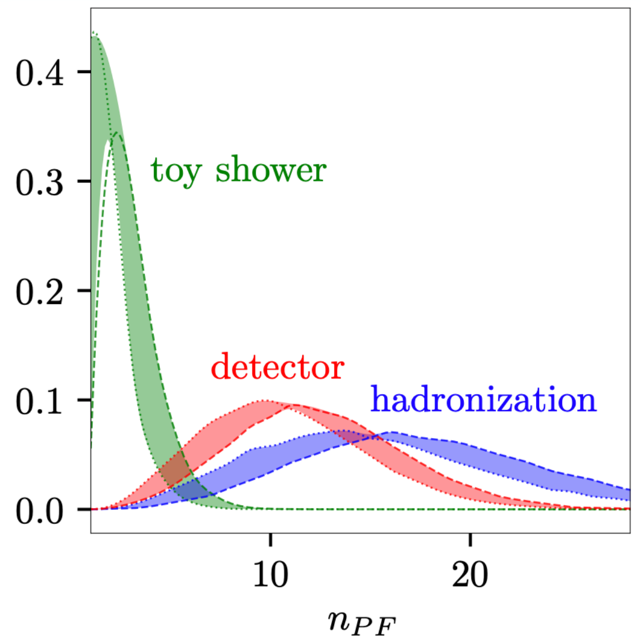
[2112.09145]

Event generation



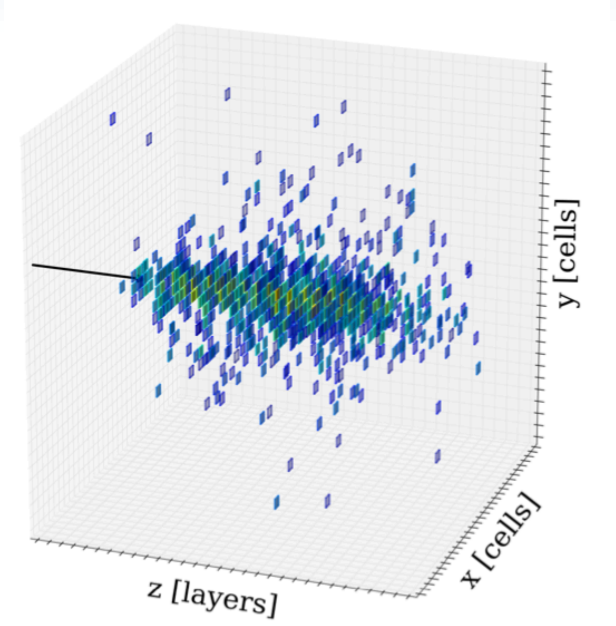
[2110.13632, 2012.07873, 2008.08558, 2008.06545, 1912.08824, 1907.03764]

Shower



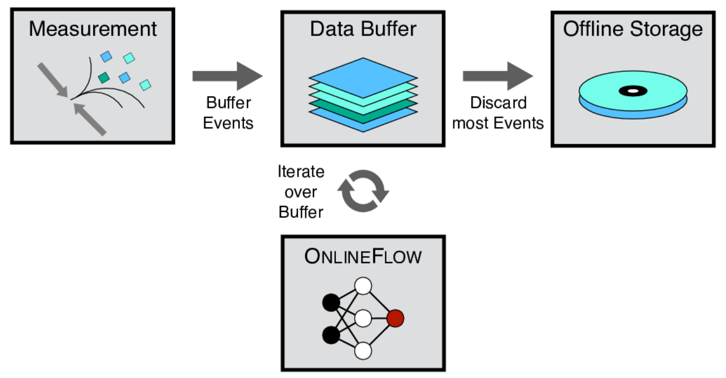
[2012.09873]

Detector simulation

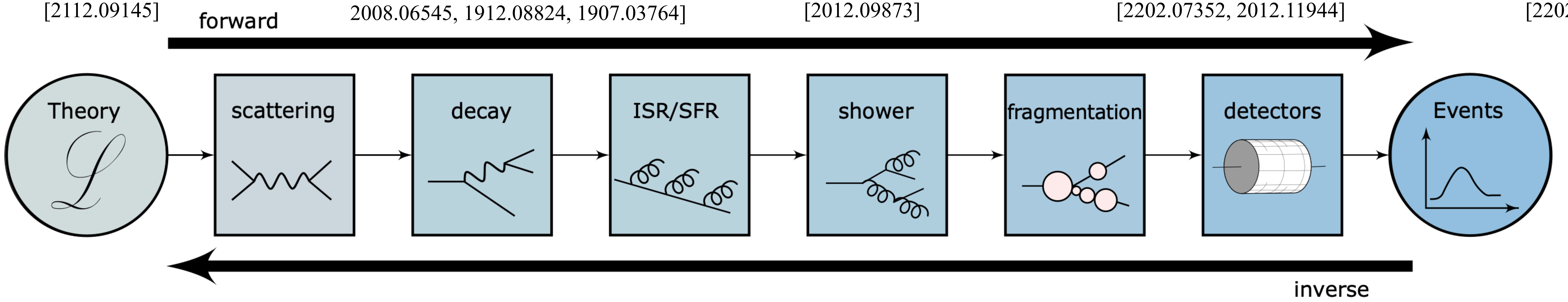


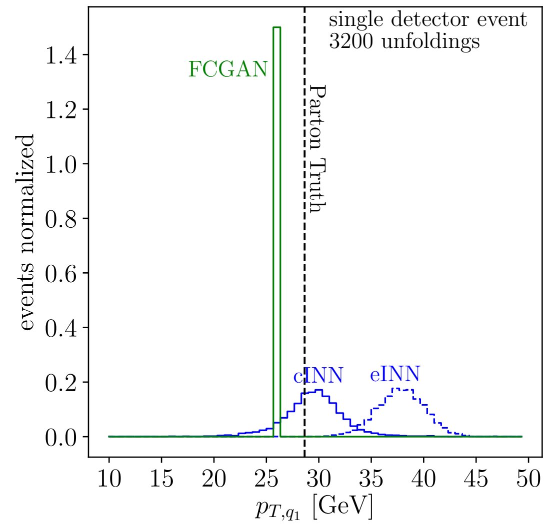
[2202.07352, 2012.11944]

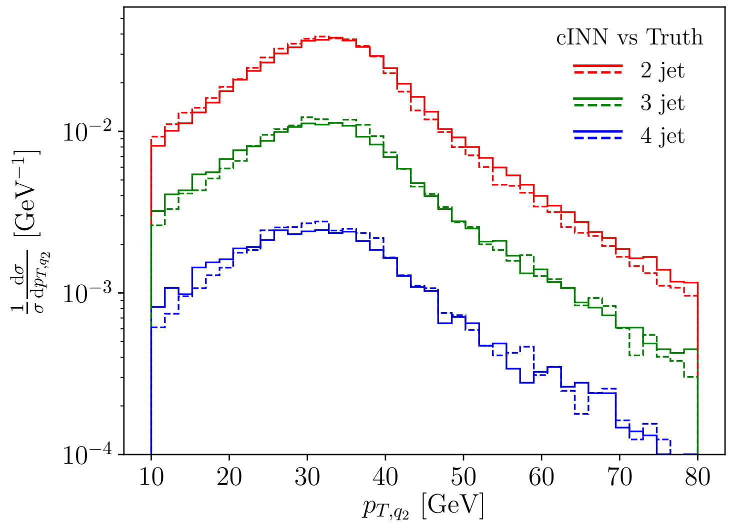
Data compression



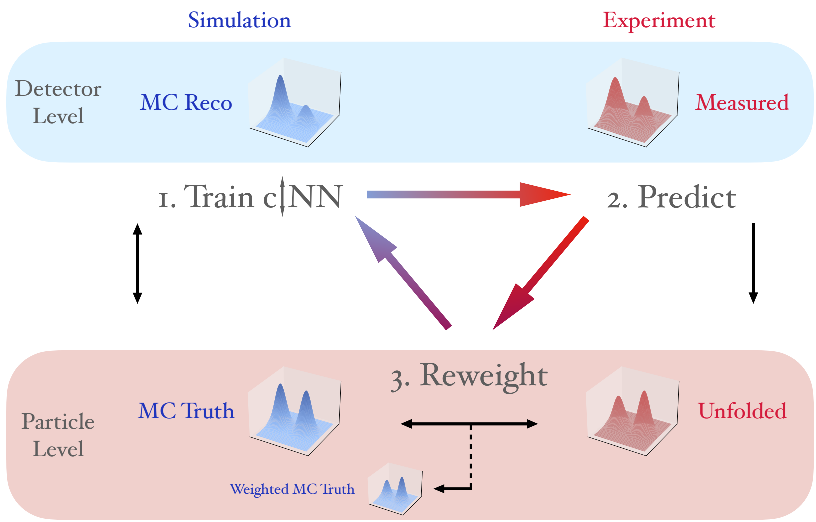
[2202.09375]







[2006.06685, 1912.00477]

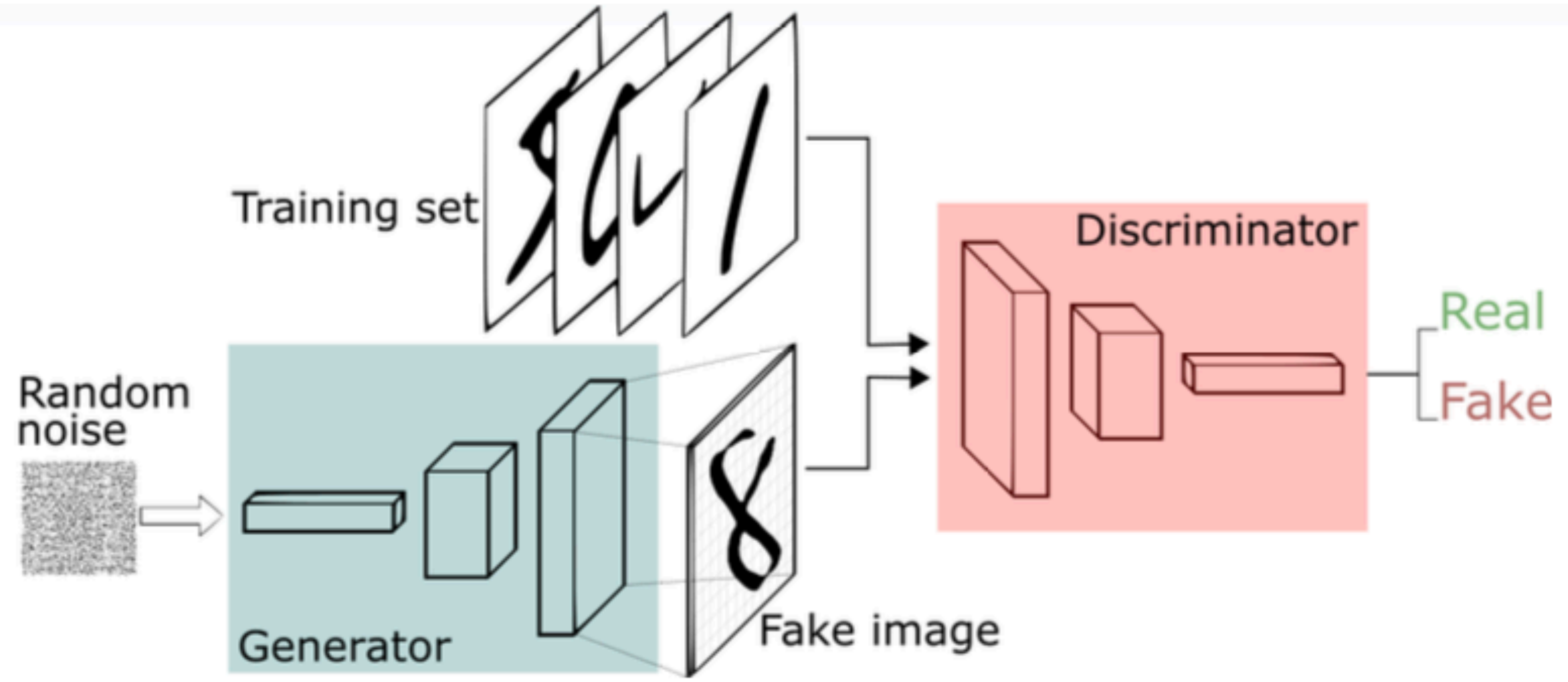


Getting ready for more data ...

Define your own

FLOW

GANs



Discriminator $[D(x_T) \to 1,\ D(x_G) \to 0]$

$$L_D = \langle -\log D(x) \rangle_{x \sim P_{Truth}} + \langle -\log(1 - D(x)) \rangle_{x \sim P_{Gen}} \to -2 \log 0.5$$

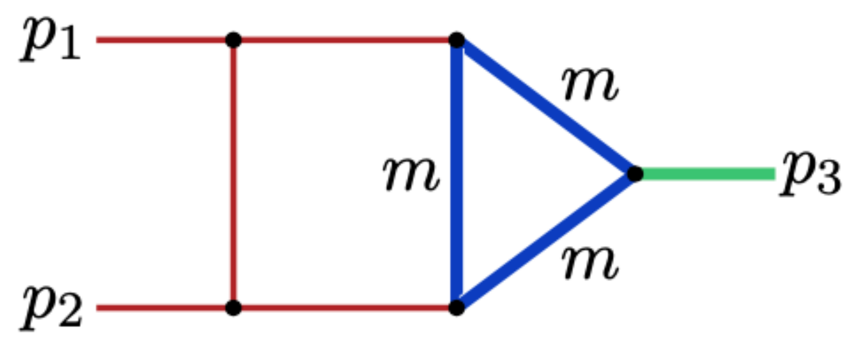
Generator $[D(x_G) \to 1]$

$$L_G = \langle -\log D(x) \rangle_{x \sim P_{Gen}}$$

Multi-loop calculations with NNs

Collaboration with G. Heinrich, et al.

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

$$G = \int_{-\infty}^{\infty} \left(\prod_{l=1}^{L} \frac{\mathrm{d}^D k_l}{i\pi^{\frac{D}{2}}} \right) \prod_{j=1}^{N} \frac{1}{(q_j^2 - m_j^2 + i\delta)^{\nu_j}}$$

$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \, \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \; I(\vec{x})$$

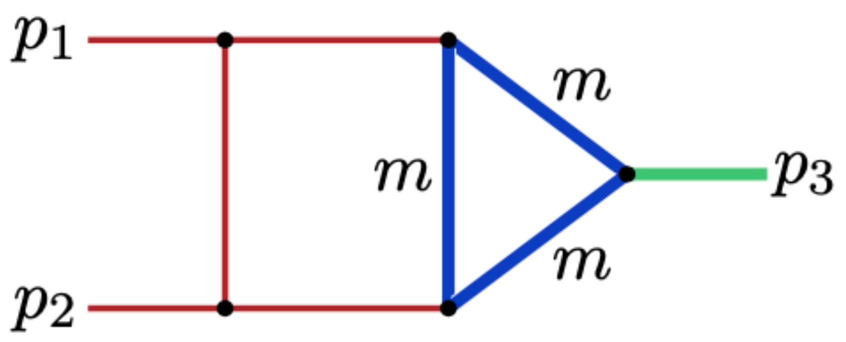
Rewrite with Feynman parameters

Still contains singularities

Multi-loop calculations with NNs

Collaboration with G. Heinrich, et al.

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

$$G = \int_{-\infty}^{\infty} \left(\prod_{l=1}^{L} \frac{\mathrm{d}^D k_l}{i\pi^{\frac{D}{2}}} \right) \prod_{j=1}^{N} \frac{1}{(q_j^2 - m_j^2 + i\delta)^{\nu_j}}$$

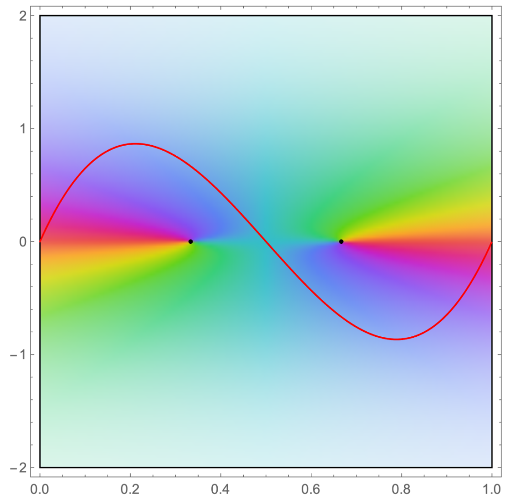
$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu - LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, I(\vec{x})$$

Rewrite with Feynman parameters

Still contains singularities

Solved by contour deformation due to Cauchy's theorem

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, I(\vec{x}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \det\left(\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}} \right) I(\vec{z}(\vec{x}))$$



How to parametrize $z(x)$?

Optimal parametrization = minimal variance

Turn it into an ML Problem

Integration with normalizing flows

Numeric integral evaluation $\rightarrow G = \int_0^1 \mathrm{d}x_j \, \det\left(\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right) I(\vec{z}(\vec{x}))$

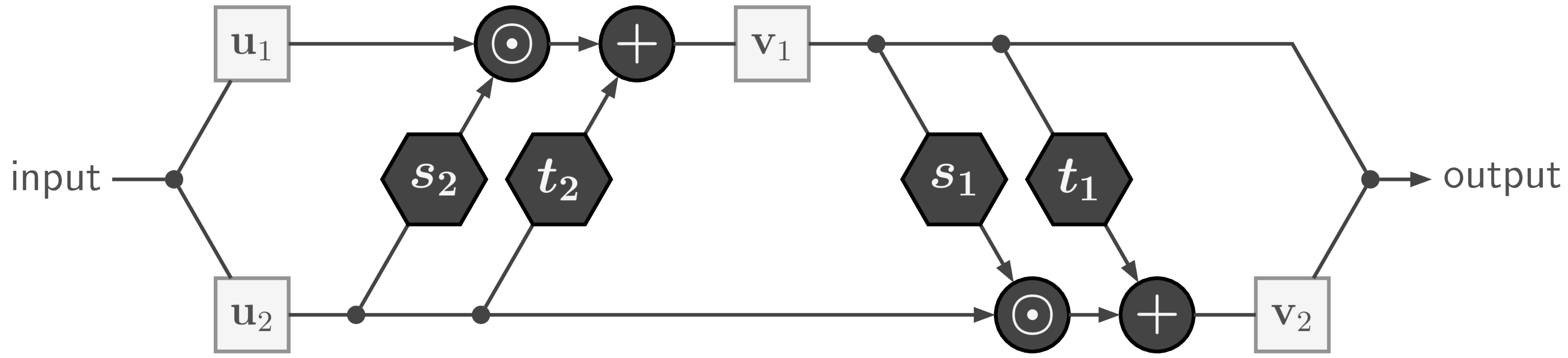
Parametrization $\rightarrow \vec{z} = \mathrm{NN}(\vec{x})$ more precisely $z_i = y_i - i\lambda y_i(1 - y_i)\frac{\partial F(\vec{y})}{\partial y_i}$ with $\vec{y} = \mathrm{NN}(\vec{x})$

Minimize **variance** $\rightarrow \text{loss } \mathscr{L} = \sigma_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left|\det\left(\frac{\partial \vec{z}(\vec{x}_{(i)})}{\partial \vec{x}_{(i)}}\right) I(\vec{z}(\vec{x}_{(i)})) - \langle I \rangle\right|^2$

Normalizing flow networks

+ Bijective mapping
+ Tractable Jacobian
+ Combine many blocks

$\vec{x}$

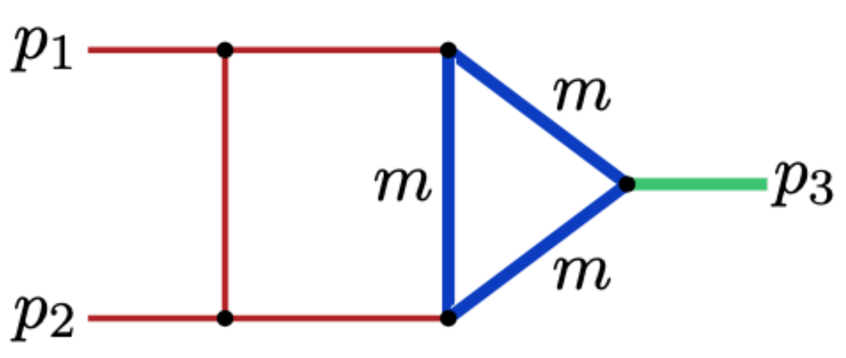


$\vec{z}(\vec{x})$

Multi-loop calculations with INNs

Profiting from the Jacobian

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

$$G = \int_{-\infty}^{\infty} \left(\prod_{l=1}^{L} \frac{d^D k_l}{i\pi^{\frac{D}{2}}} \right) \prod_{j=1}^{N} \frac{1}{(q_j^2 - m_j^2 + i\delta)^{\nu_j}}$$

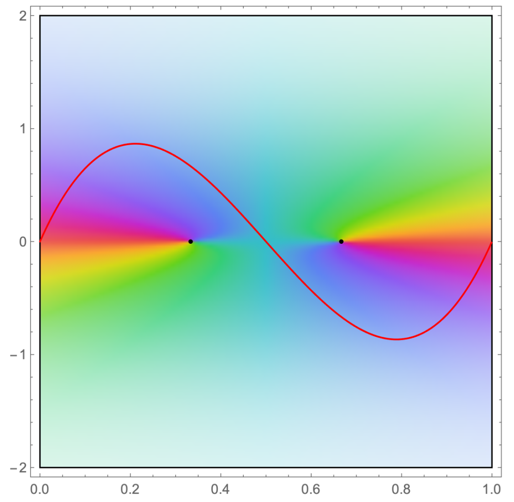
$$= \int_0^1 \prod_{j=1}^{N-1} dx_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} dx_j \, I(\vec{x})$$

Rewrite with Feynman parameters

Still contains singularities

Solved by contour deformation due to Cauchy's theorem

$$\int_0^1 \prod_{j=1}^{N} dx_j \, I(\vec{x}) = \int_0^1 \prod_{j=1}^{N} dx_j \, \det\left(\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}} \right) I(\vec{z}(\vec{x}))$$

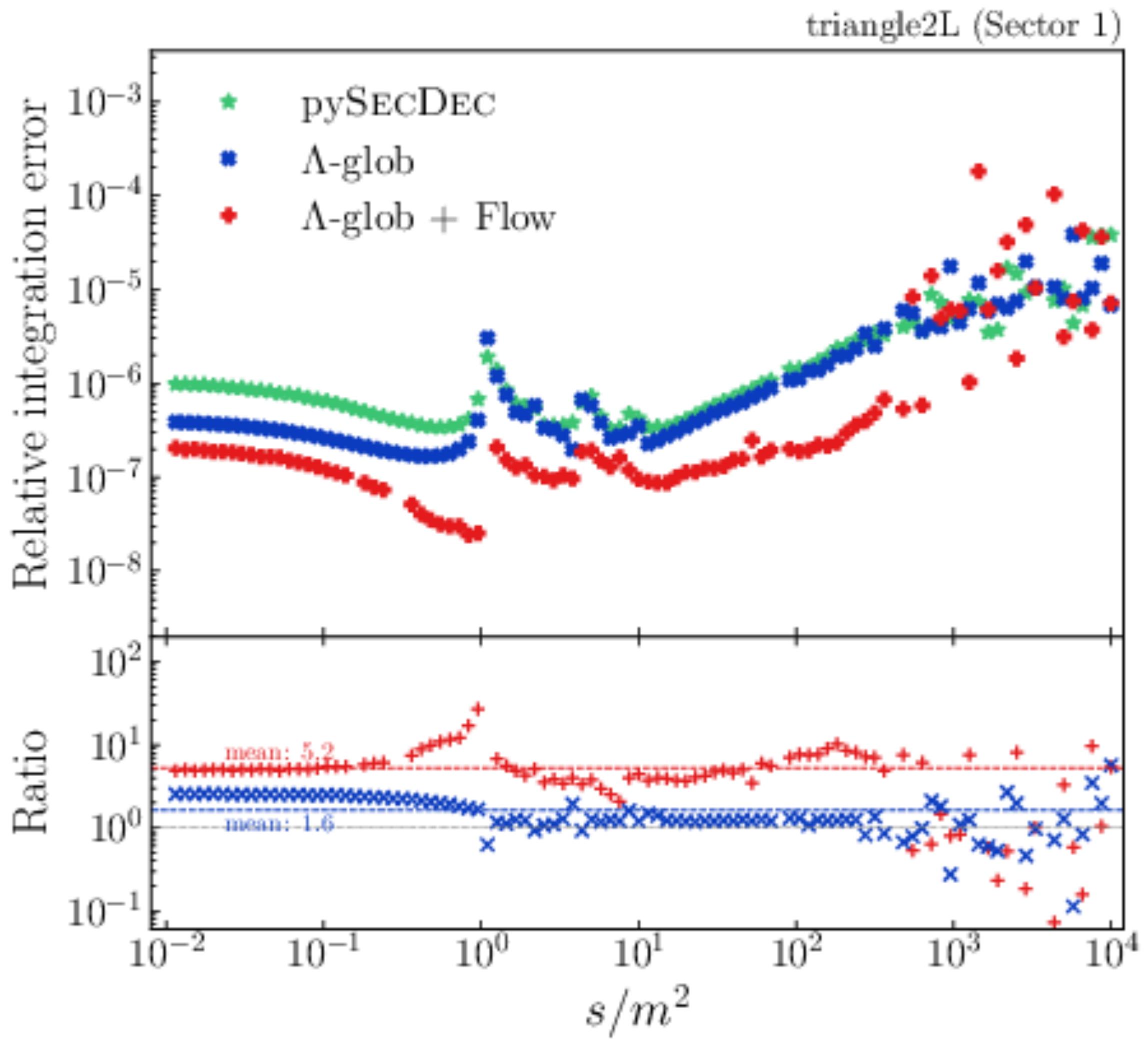


Optimal parametrization = minimal variance

Turn it into an ML Problem

Parametrization $\rightarrow z = \text{INN}(x)$

Variance $\rightarrow \mathscr{L}$



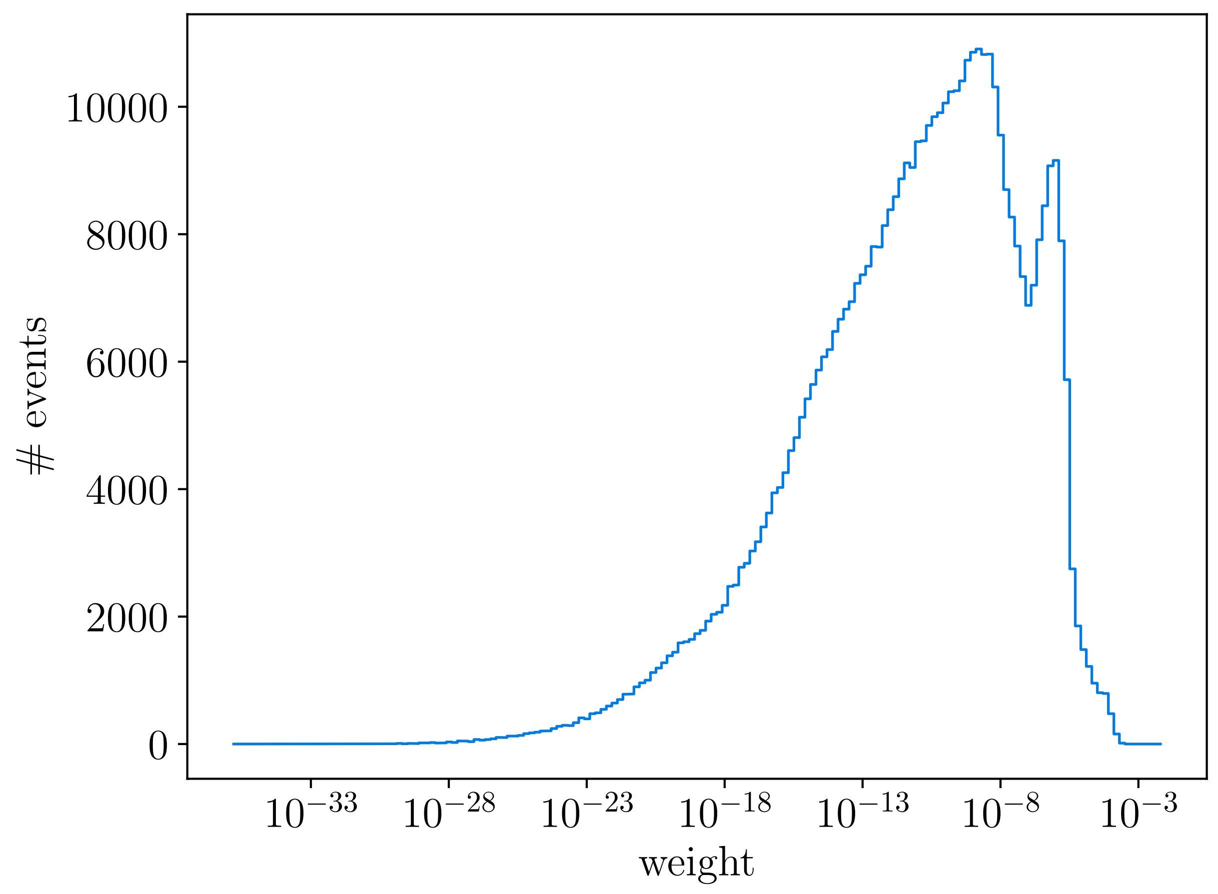
R. Winterhalder, et al. [2112.09145]

How to train on weighted events?

$$\mathscr{L}_{\text{INN weighted}} = \int dx\, w(x)\, P(x) \left(\frac{\text{INN}(x)^2}{2} - \log J(x) \right) \approx \sum_{x_i} w(x_i) \left(\frac{\text{INN}(x_i)^2}{2} - \log J(x_i) \right)$$

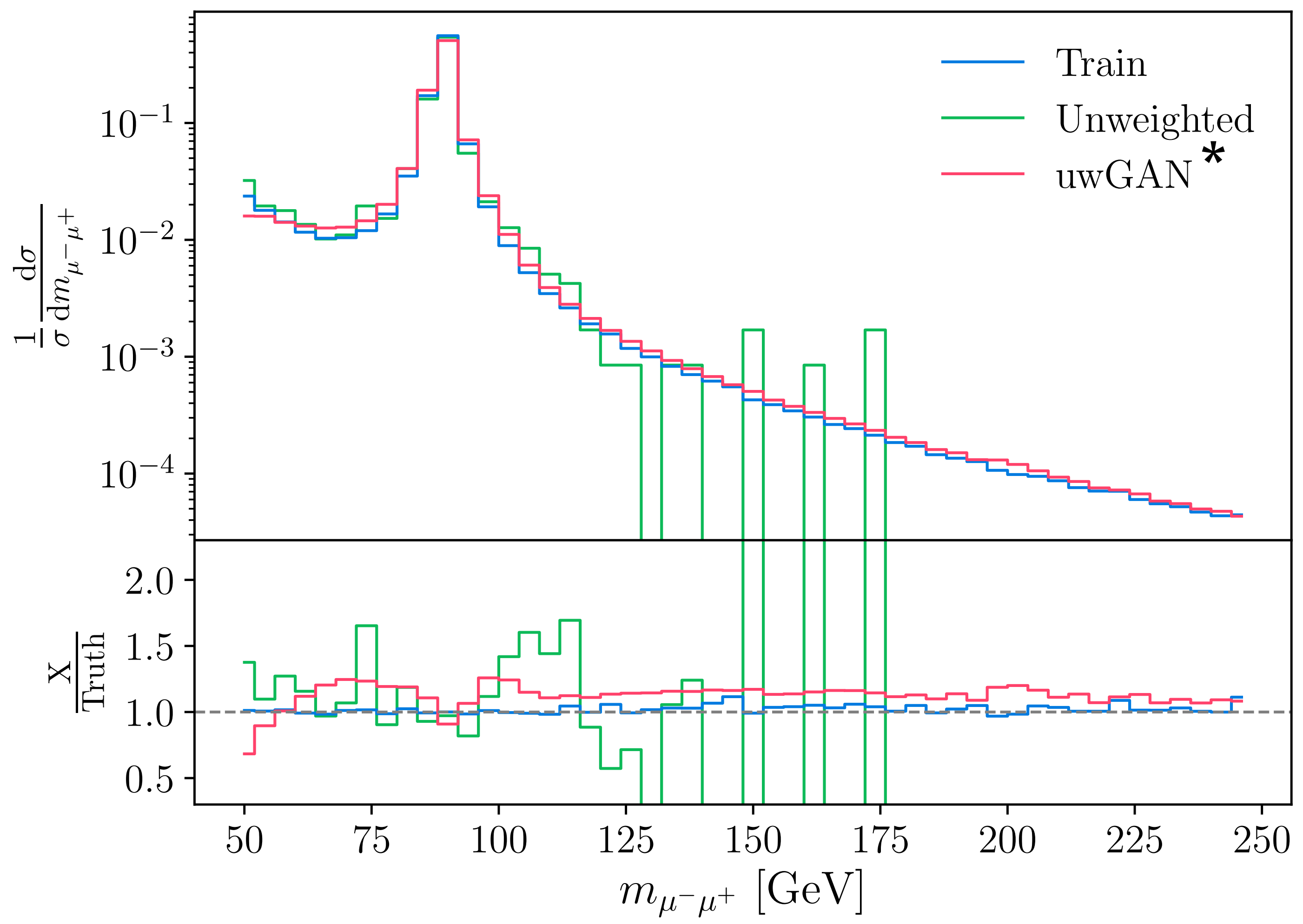
⇒ Outputs unweighted events !

Applied to
Drell Yan process $Z \rightarrow \mu^+\mu^-$ with naive RAMBO setup



⇒

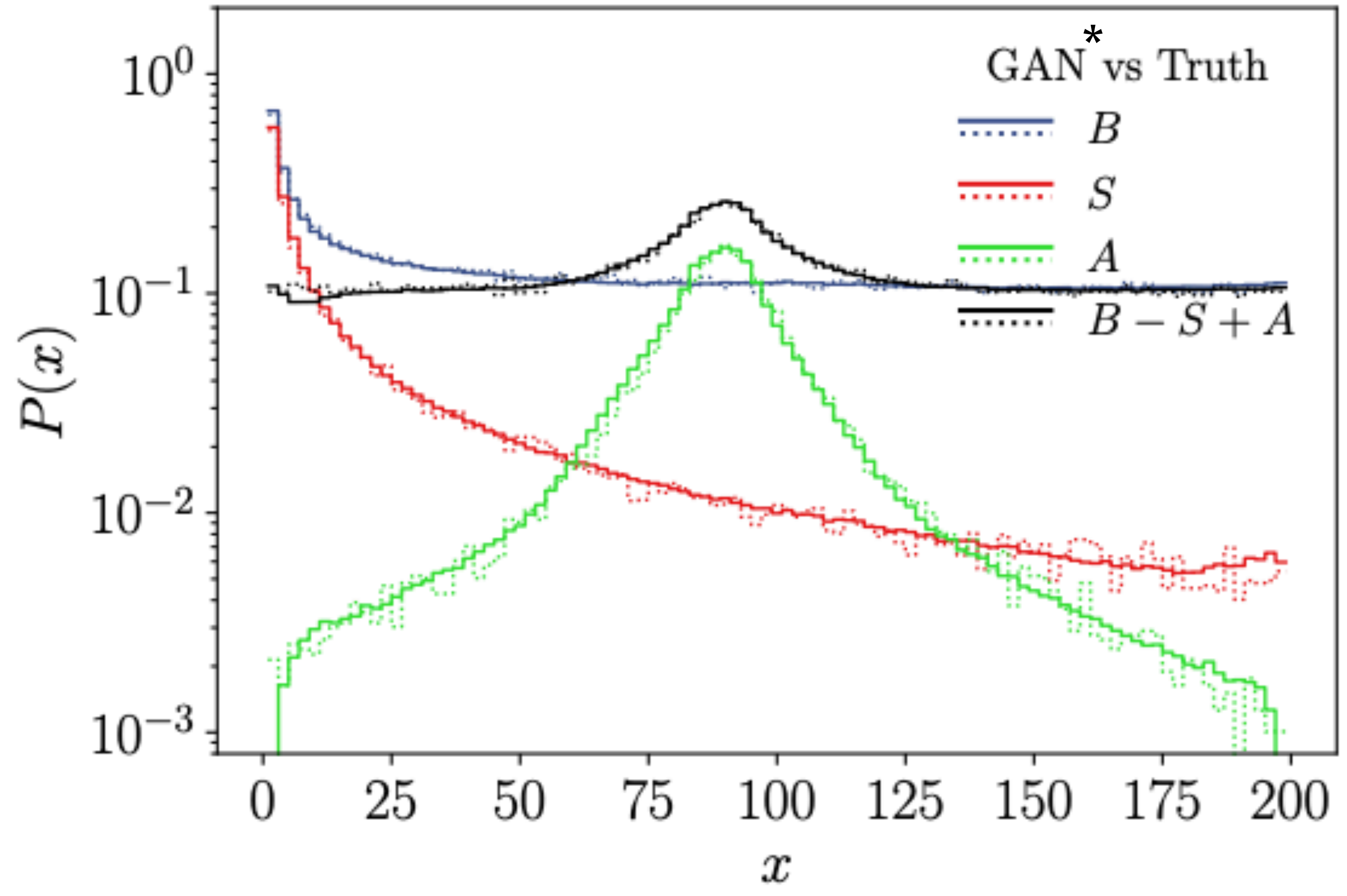
Generated events (w=1) reproduce weighted training data



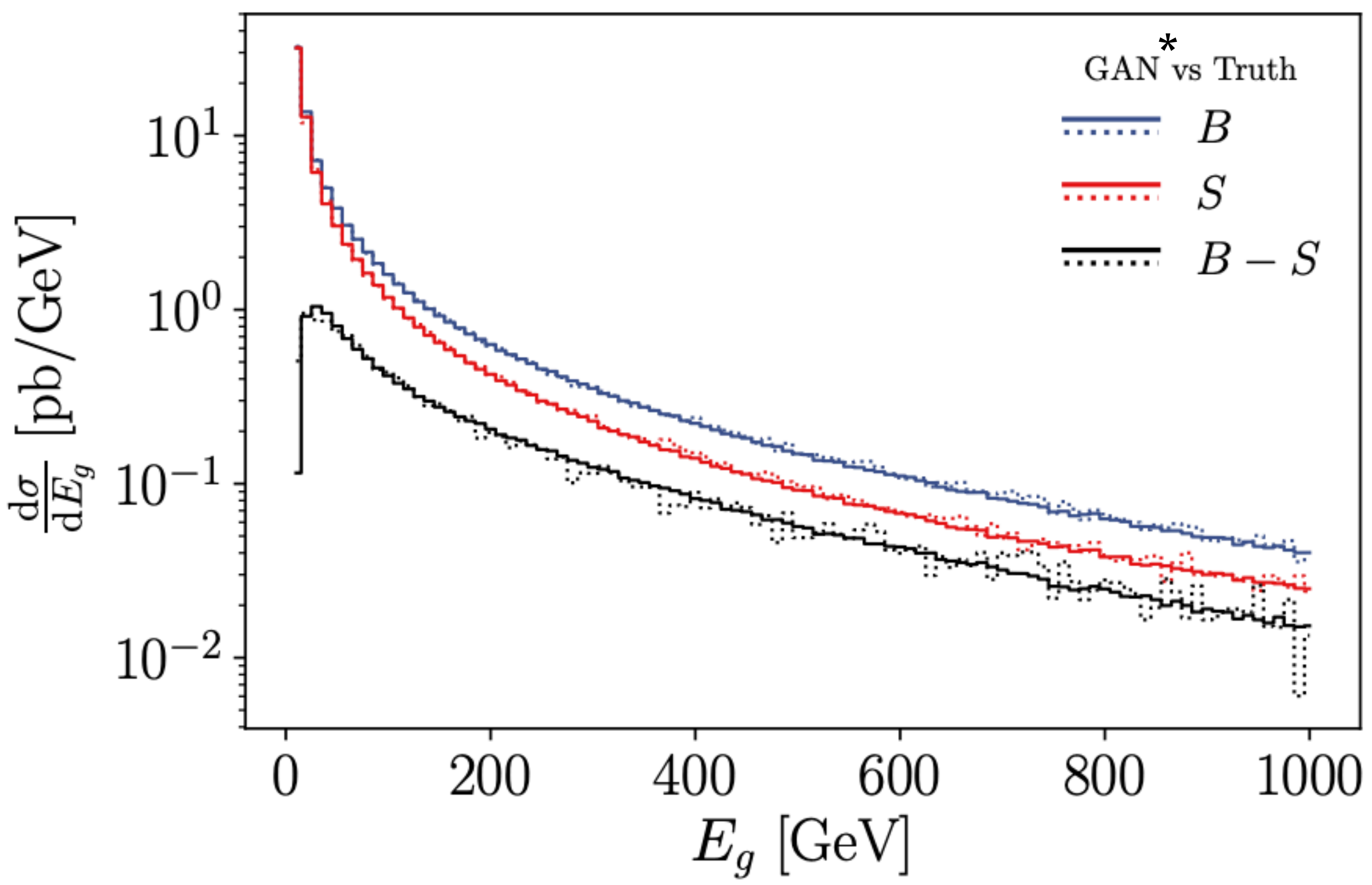
* older generative model - potentially worse performance than flows

Combination with negative weights

Toy example for addition and subtraction



Zg collinear divergence
subtraction of a shifted dipole



* older generative model - potentially worse performance than flows