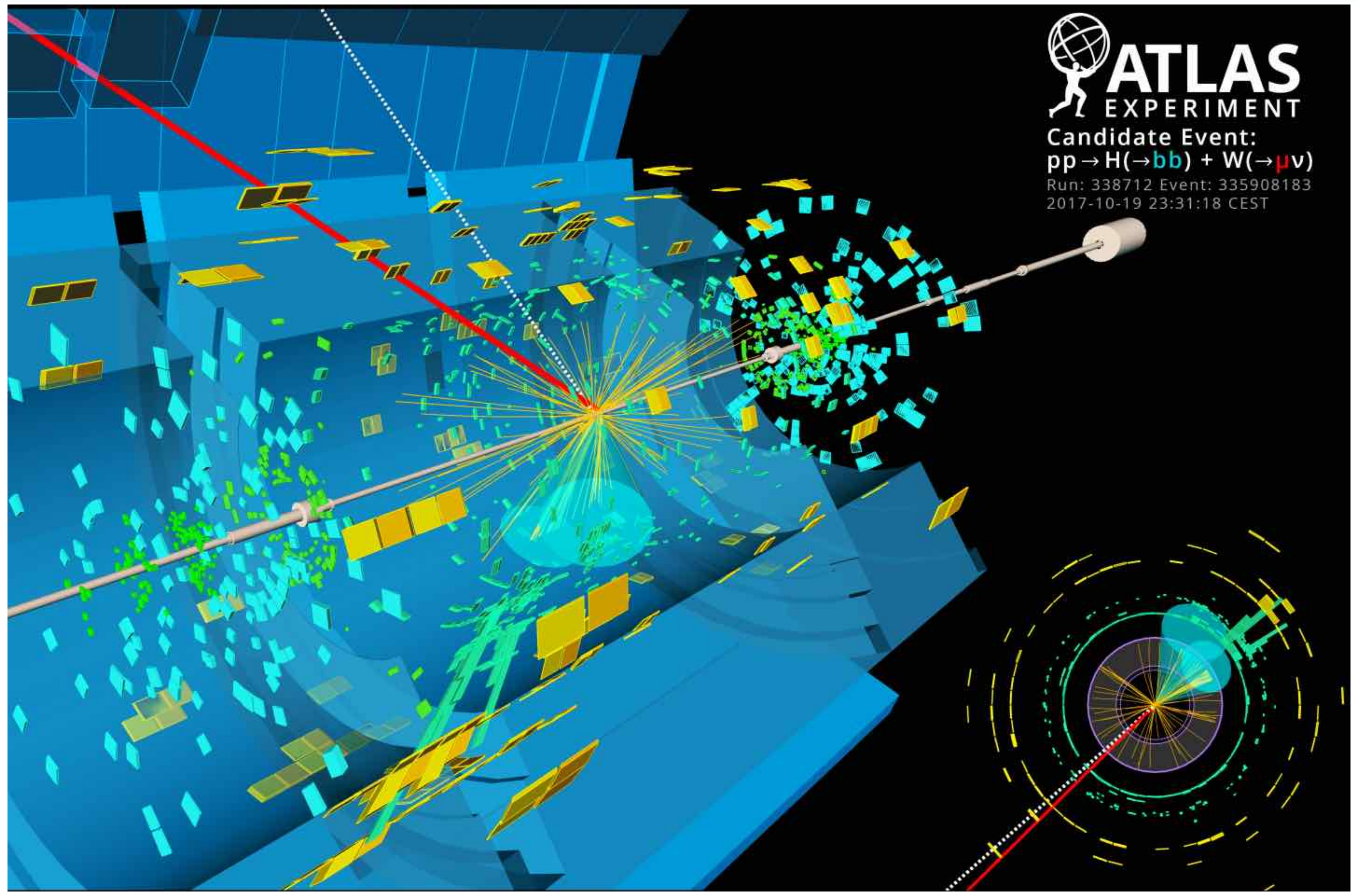
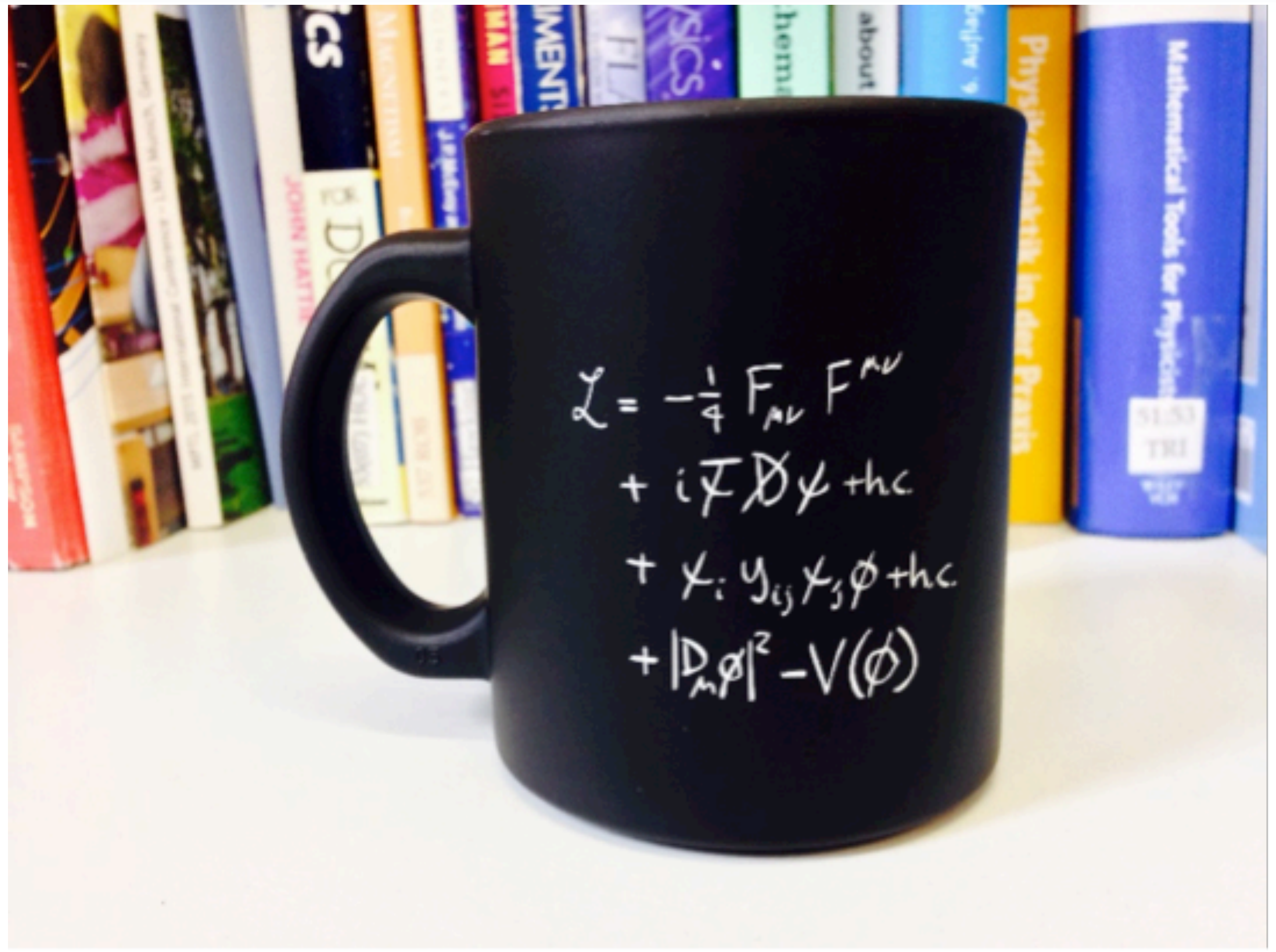
Up and down the simulation chain with neural networks

Anja Butter, LPNHE, ITP

LPNHE
PARIS

Ideal conditions for ML research




$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$
$+ i\bar{\psi} \not{D} \psi + h.c.$
$+ \chi_i y_{ij} \chi_j \phi + h.c.$
$+ |D_\mu \phi|^2 - V(\phi)$

Data

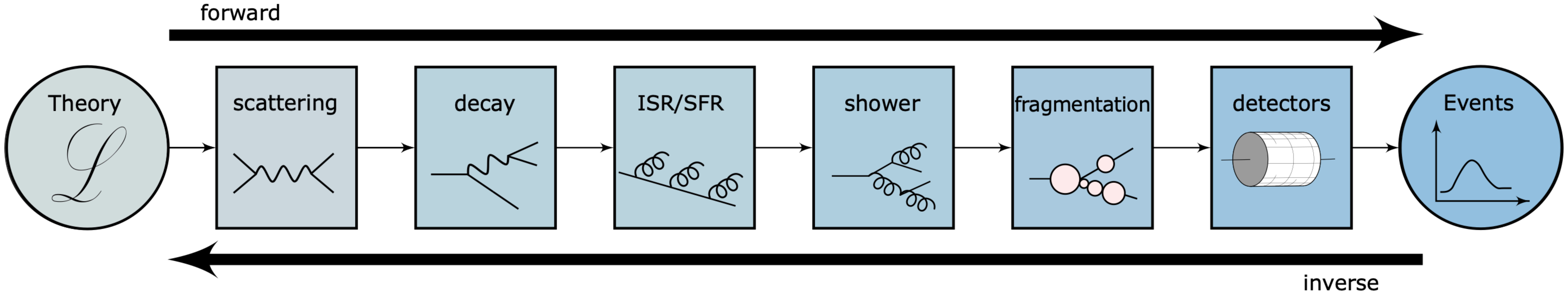
- **Huge** dataset ~1Pb/s before trigger selection
- High-dim. kinematic distributions, jet substructures …

Control

- Understand full dataset from **1st principles**
- Precision measurements of the (B)SM

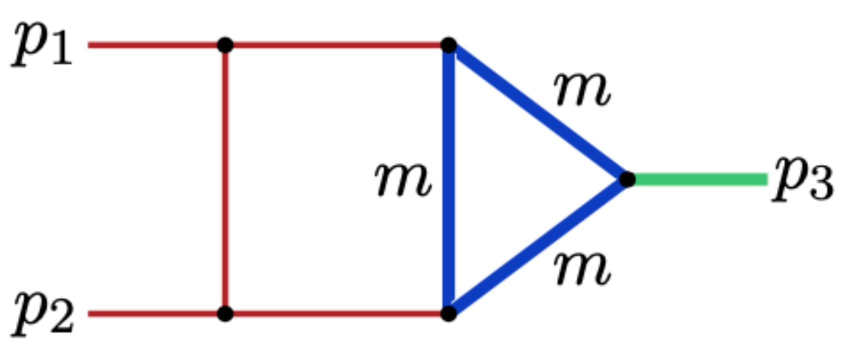
How can ML help to exploit all information in the dataset?

Machine learning up and down the simulation chain

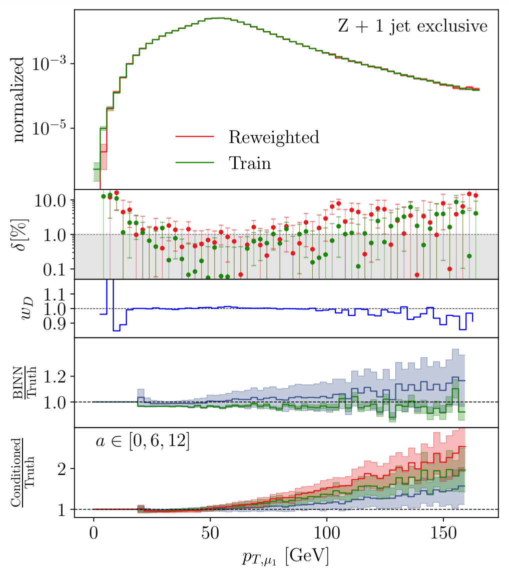


Machine learning up and down the simulation chain

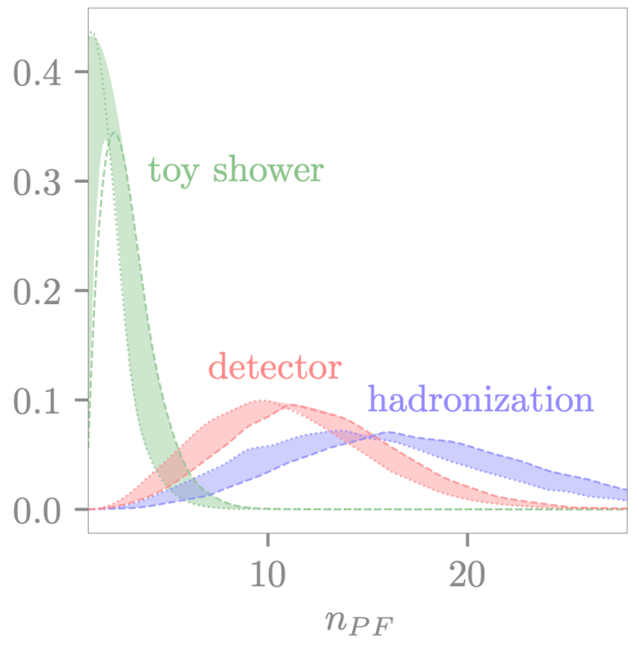
Loop amplitudes



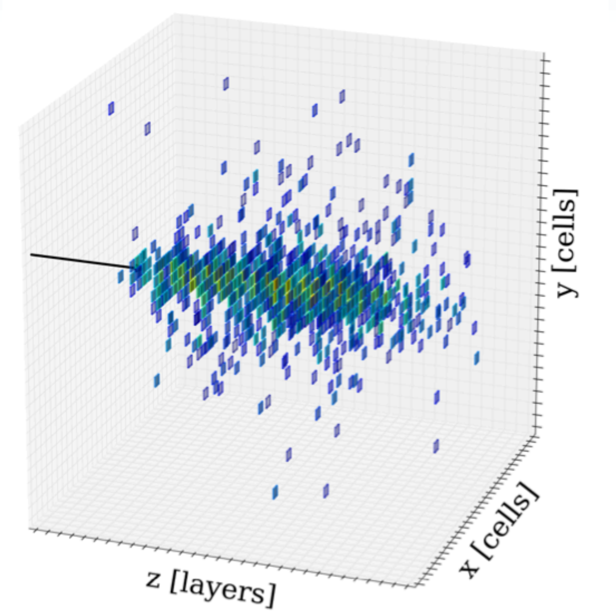
Event generation

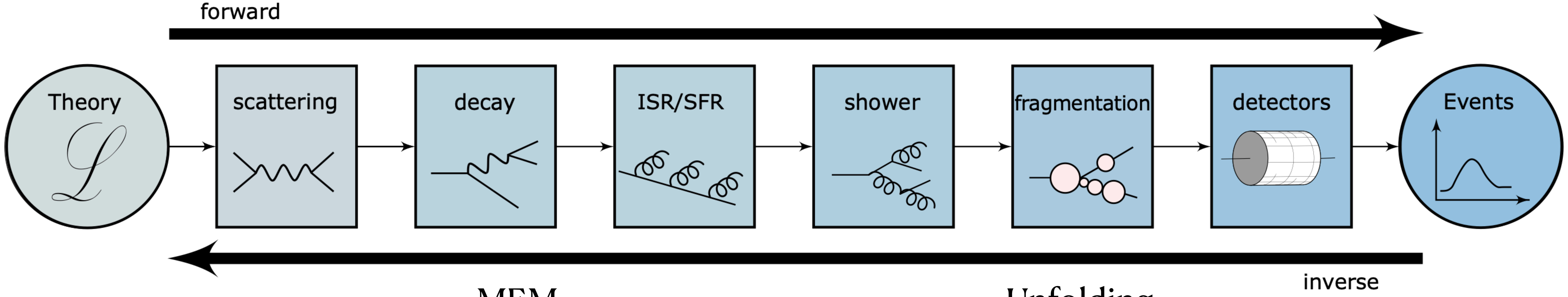


Shower

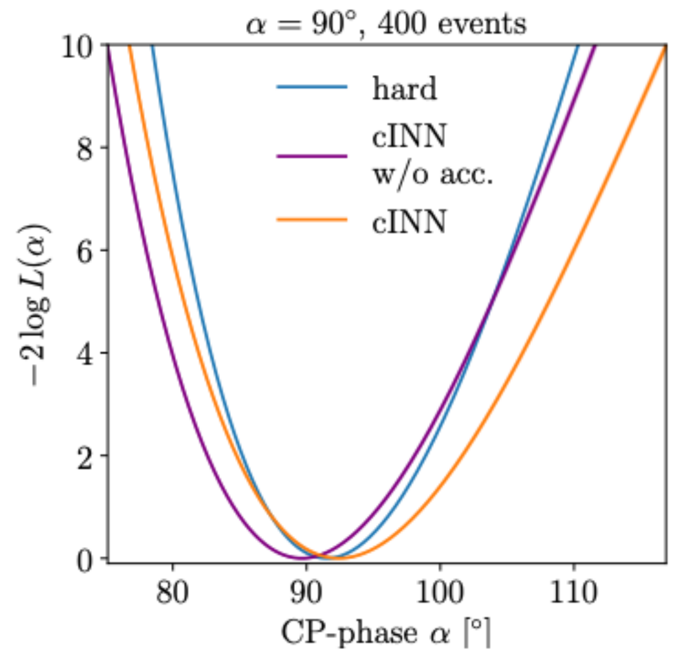


Detector simulation

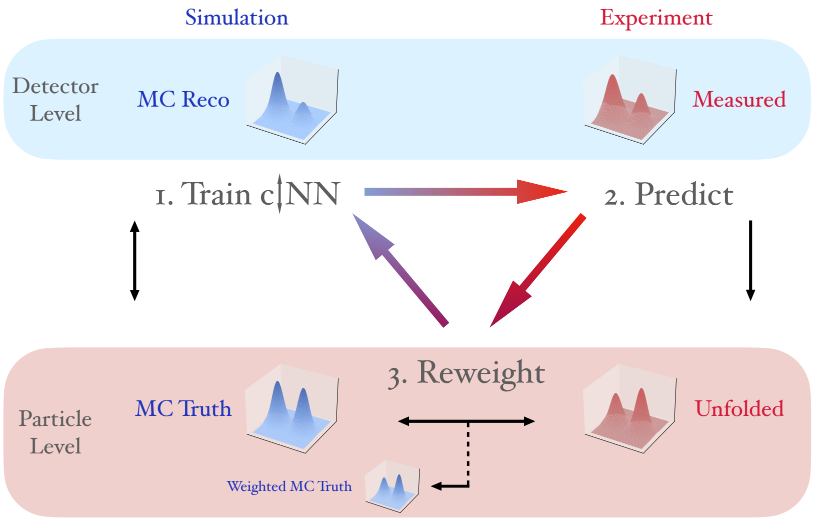




MEM



Unfolding



Monte carlo event generation

1. Generate phase space points

→ 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
     - ➡ $J(p_i(r)) = (f \times \mathscr{M})^{-1}$

How to include computing intensive amplitudes in MC simulations?

Interpolating amplitudes with neural networks

Standard approach

Training data

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

Loss

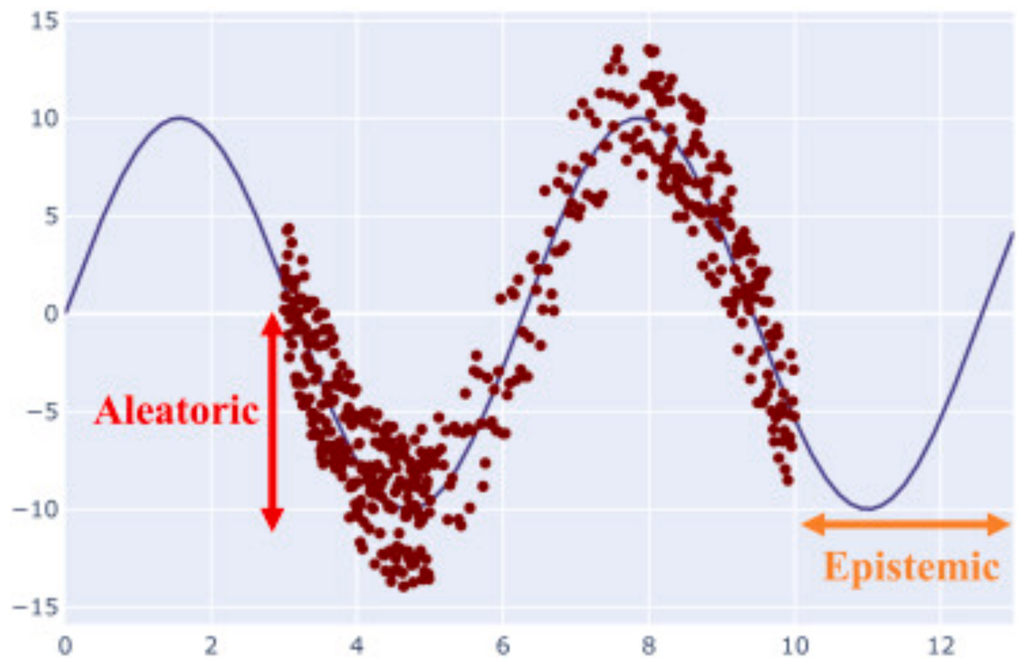
$\mathscr{L} = (A'(x) - NN(x))^2$

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

⇒

→ Instead find $p(A \,|\, x, T)$ (from now on x is implicit)

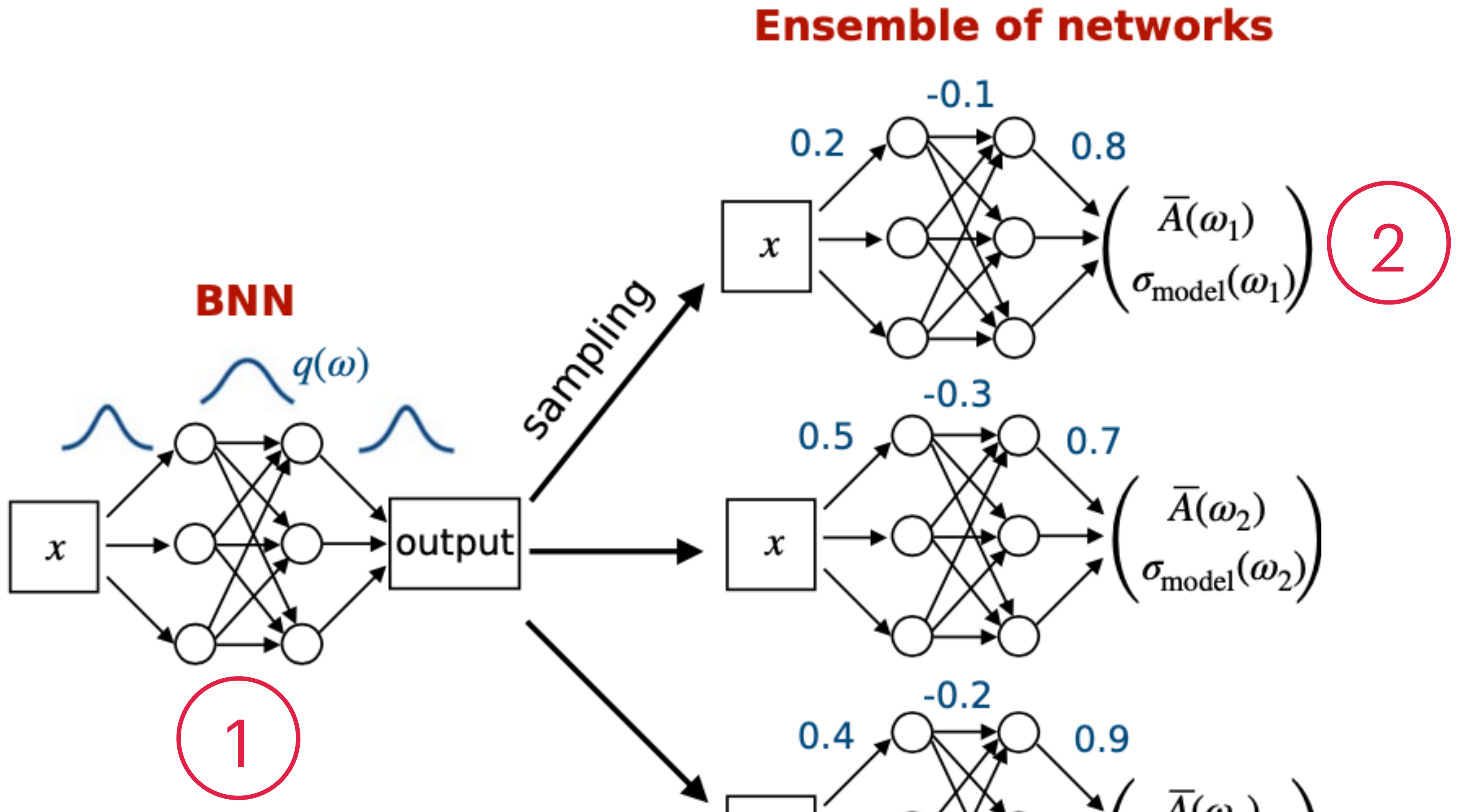
→ $p(A) = \int \mathrm{d}w \; p(A \,|\, w) p(w \,|\, T)$



Capturing probabilities with Bayesian networks

$$\boxed{p(A) = \int \mathrm{d}w \; p(A\,|\,w)p(w\,|\,T) \approx \int \mathrm{d}w \; \overset{2}{p(A\,|\,w)}\,\overset{1}{q(w)}}$$

Bayesian networks



Distribution over
likely weight config.

Building the loss function

Approximate $q(w)$ by minimizing KL divergence

$$\mathscr{L}_{BNN} = \mathrm{KL}[q(w), p(w)] - \int dw \; q(w) \;\; \log p(T\,|\,w)$$

1

Gaussian prior

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

2

Gaussian uncertainty

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

Results - out of the box

A. Butter, et al. [2206.14831]

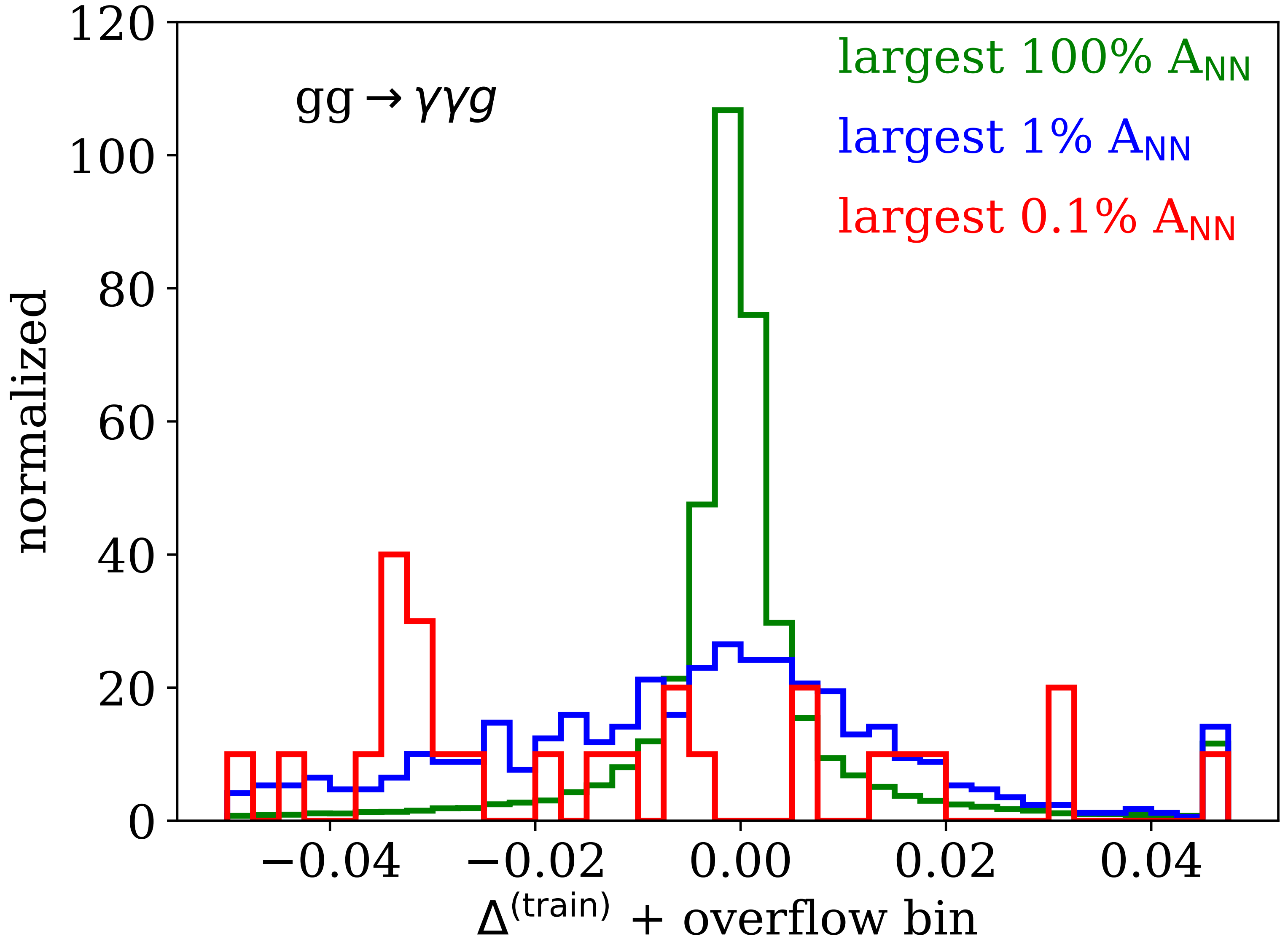
+ Deviations at 1 percent level

Example

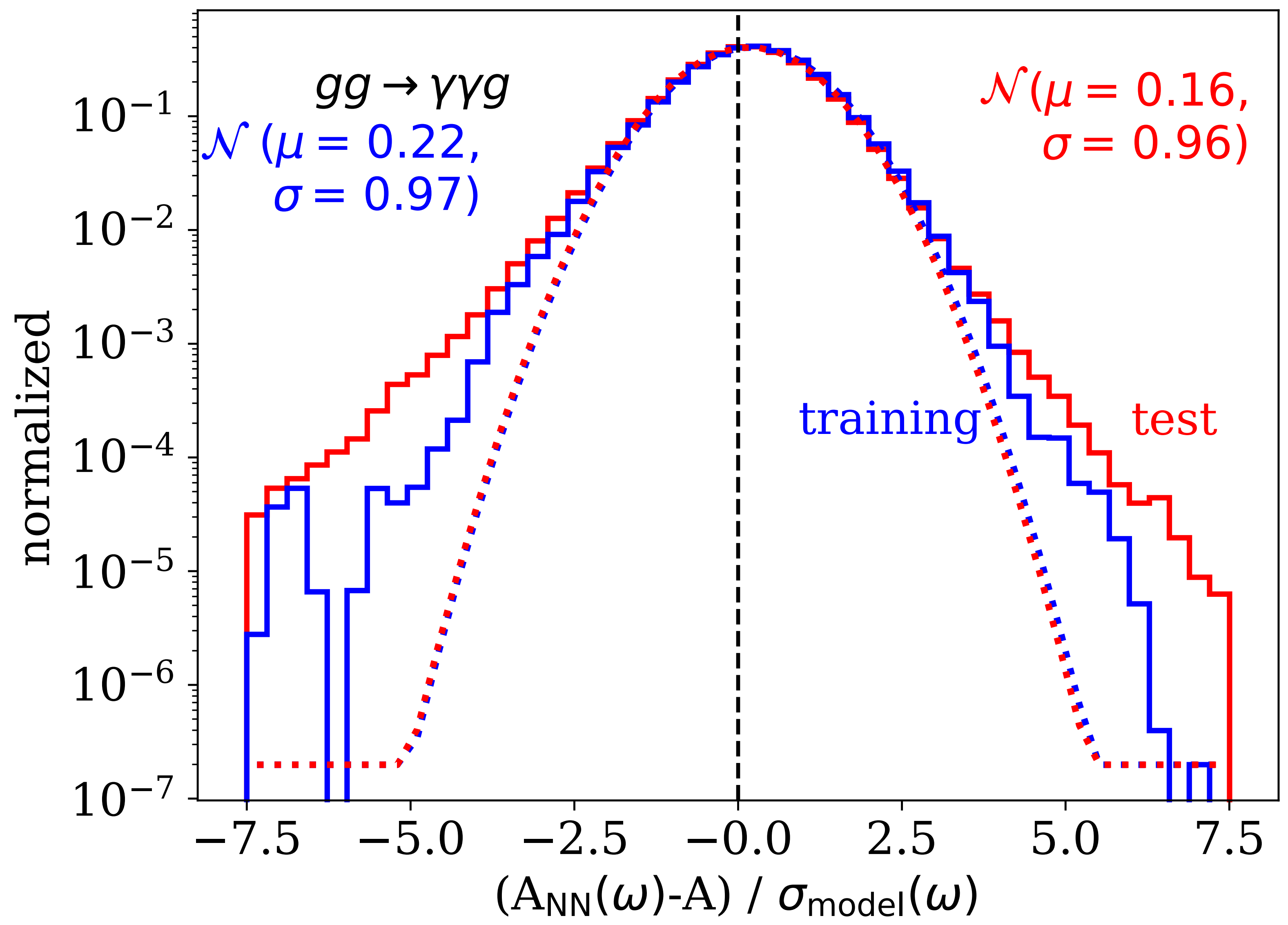
$gg \to \gamma\gamma g(g)$ @LO

90k training amplitudes
870k test amplitudes

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

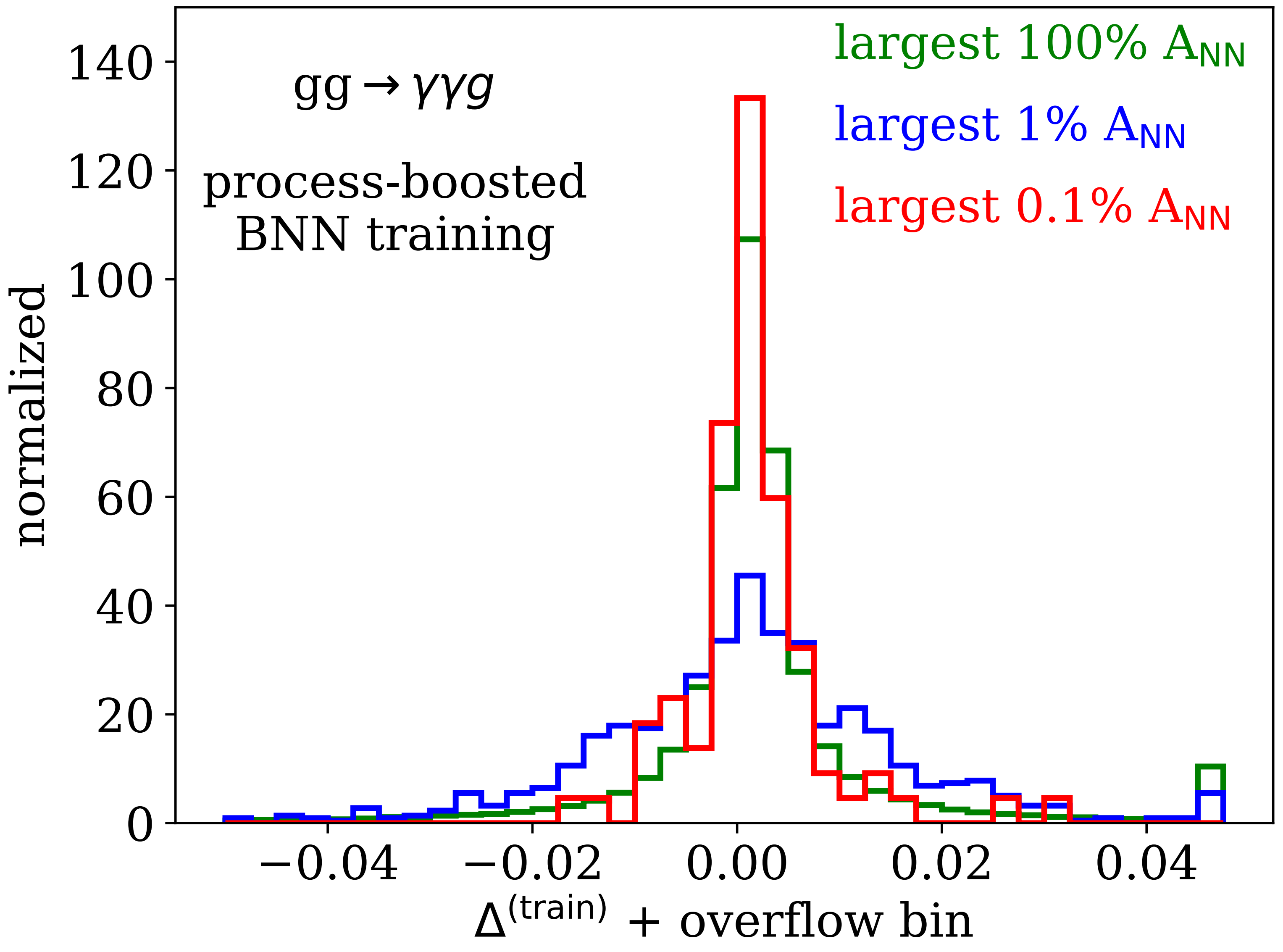
Boosting performance & calibration

Example

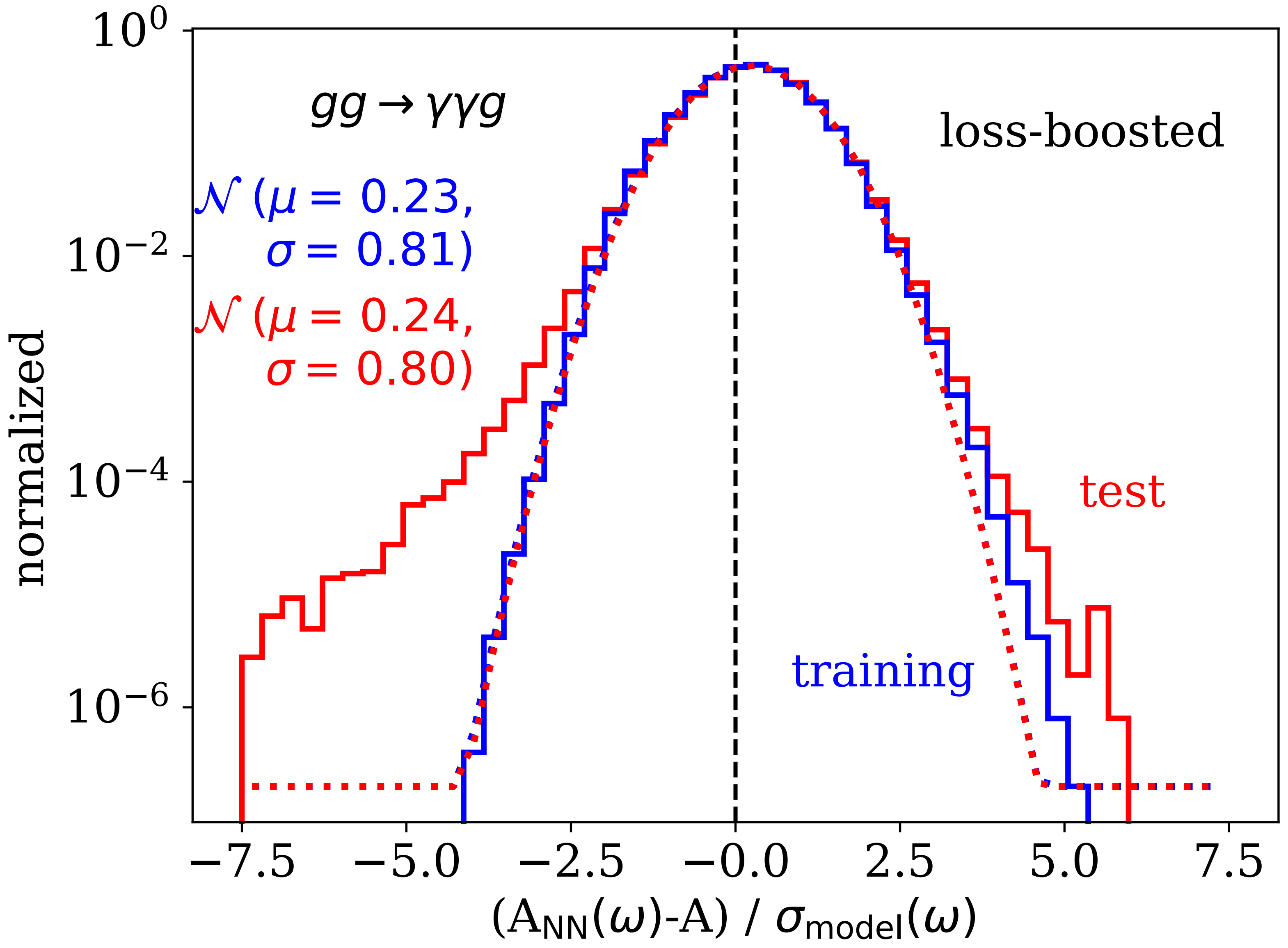
$gg \to \gamma\gamma g(g)$ @LO

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

→ 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
     - ➡ $J(p_i(r)) = (f \times \mathscr{M})^{-1}$

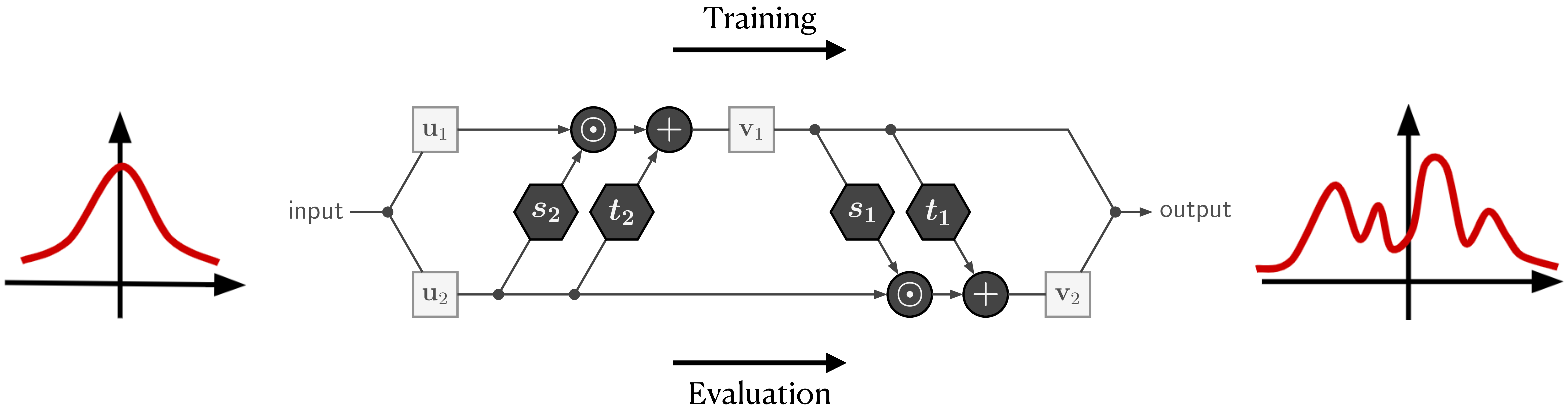
Normalizing flows

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

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

Requires evaluation of $t(x)$

Training on samples x

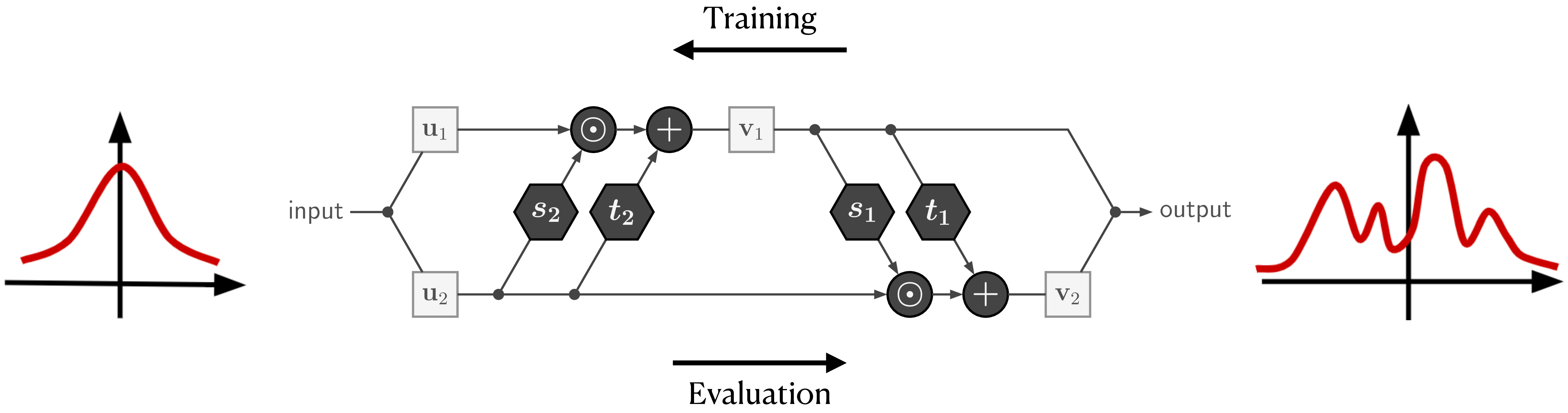
→ Maximize the log-likelihood

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

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



Training on density $t(x)$

→ Minimize difference

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

Requires evaluation of $t(x)$

Training on samples x

→ Maximize the log-likelihood

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

Normalizing flows

Invertible networks for complex transformations

- Bijective mapping
- Tractable Jacobian → $p_x(x) = p_z(z) \cdot J_{NN}$
- Fast evalua

Optimized training strategy in MadNIS [2212.06172]

-> Implemented in MadGraph

T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn, R. Winterhalder [2311.01548]

Training on density $t(x)$

→ Minimize difference

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

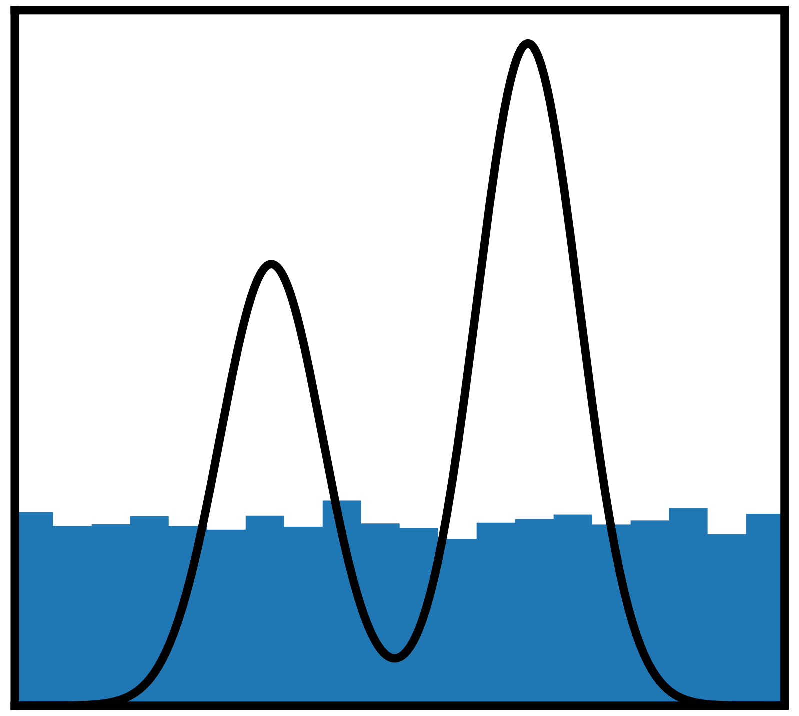
Training on samples x

→ Maximize the log-likelihood

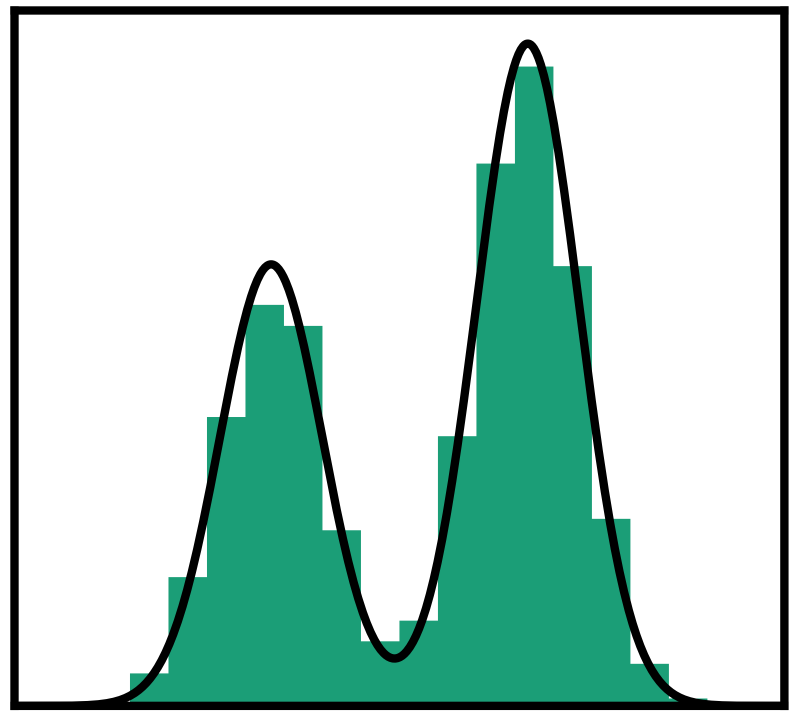
$$\mathscr{L} = \log p(\theta|x)$$
$$= \log p(z|\theta) + \log J_{NN} + p(\theta)$$

MadNIS — Neural importance sampling

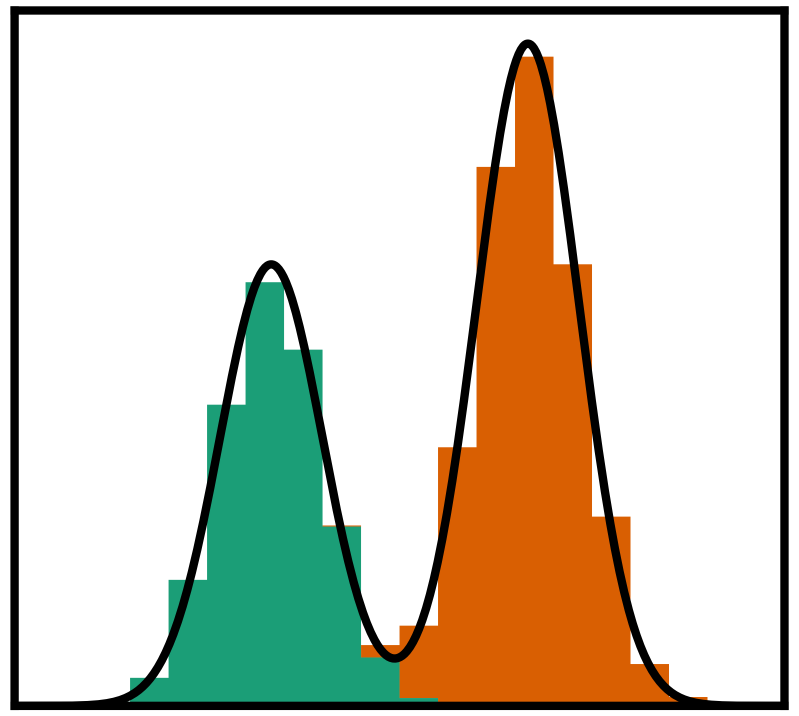
[2212.06172, 2311.01548, 2408.01486]



Flat sampling



Importance Sampling

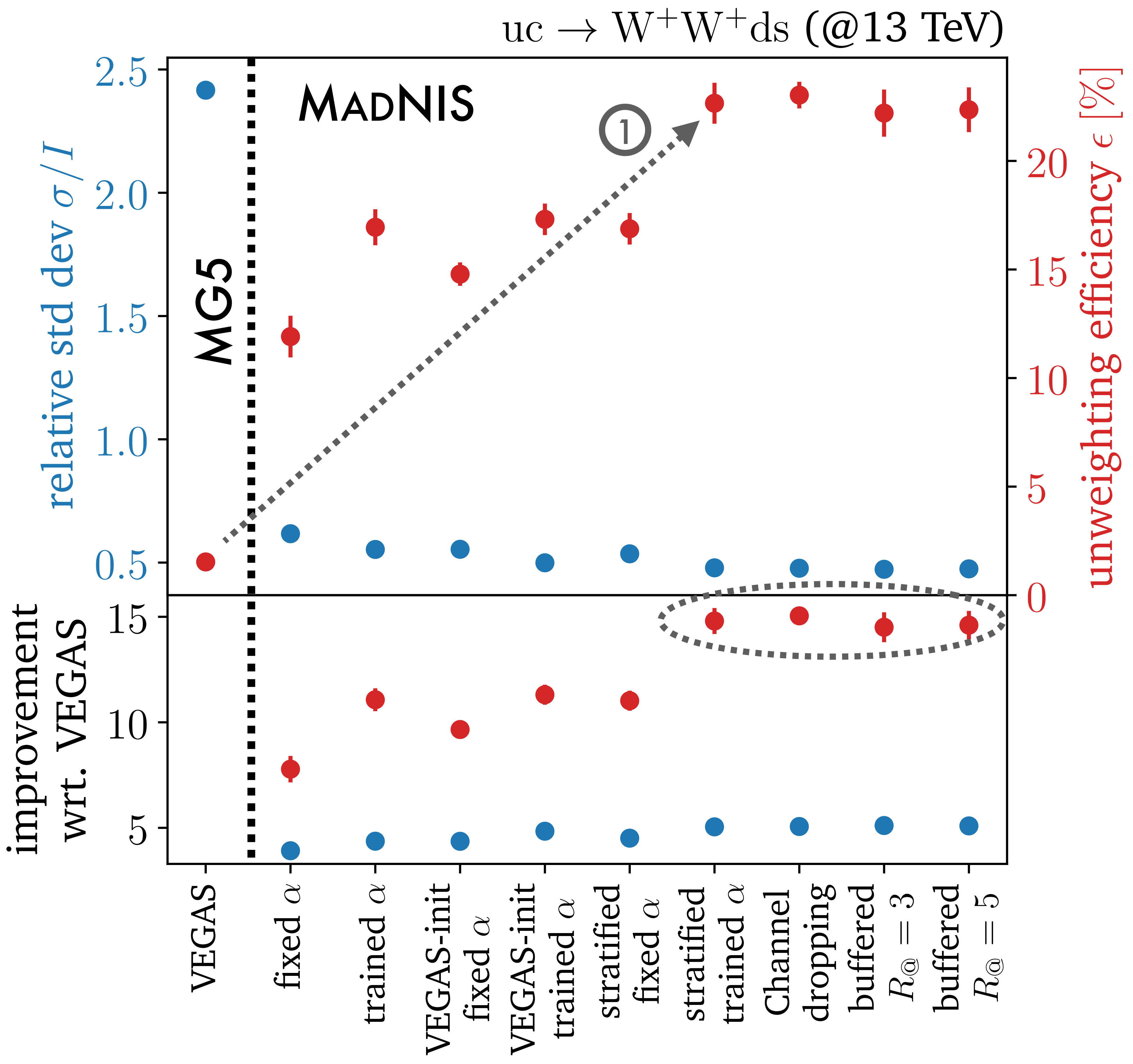


Multi-channel

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(x)}{g_i(x)} \right\rangle_{x \sim g_i(x)}$$

Parametrize with **NN**

Parametrize with **Normalizing Flow**

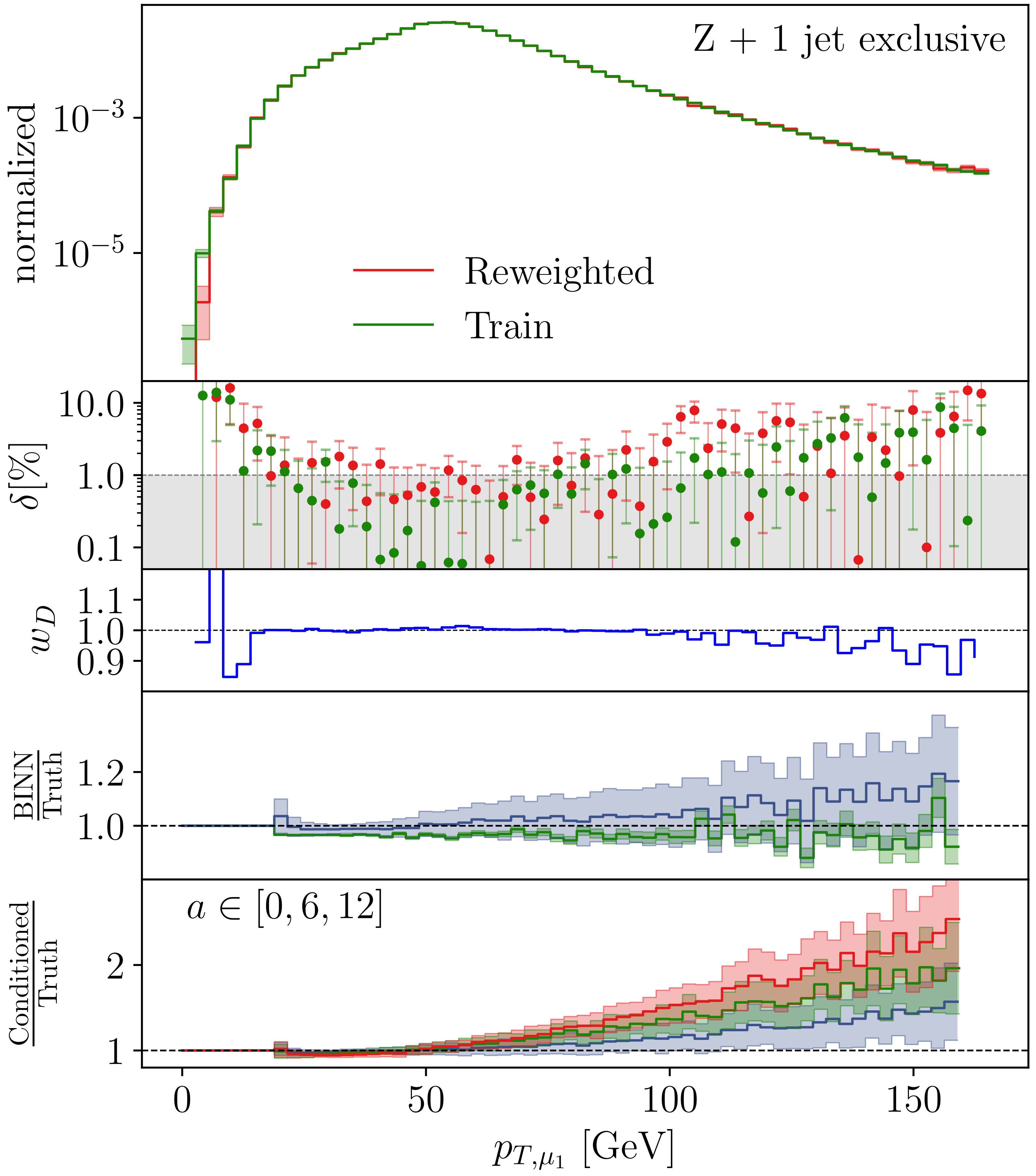


① excellent results with all features

Slide by R. Winterhalder

Keeping neural networks under control

A. Butter, et al. [2110.13632]



- Basis: INN
  - Phase space symmetries in architecture
- Control via classifier D
  - $$\frac{p_{\mathrm{truth}}(x)}{p_{\mathrm{INN}}(x)} = \frac{D(x)}{1 - D(x)}$$
- 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 for loop calculations ✓

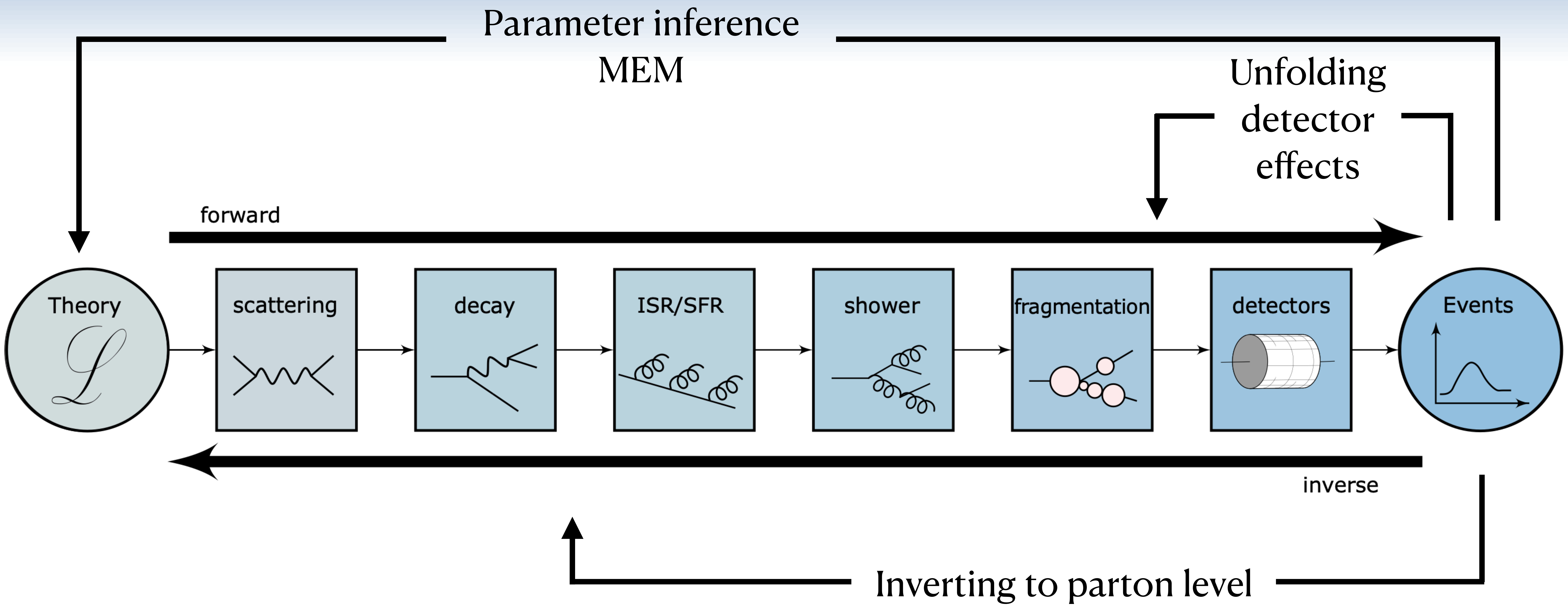
Phase space sampling ✓

→ Precision ≡ efficiency

Transferable to detecor simulation ✓

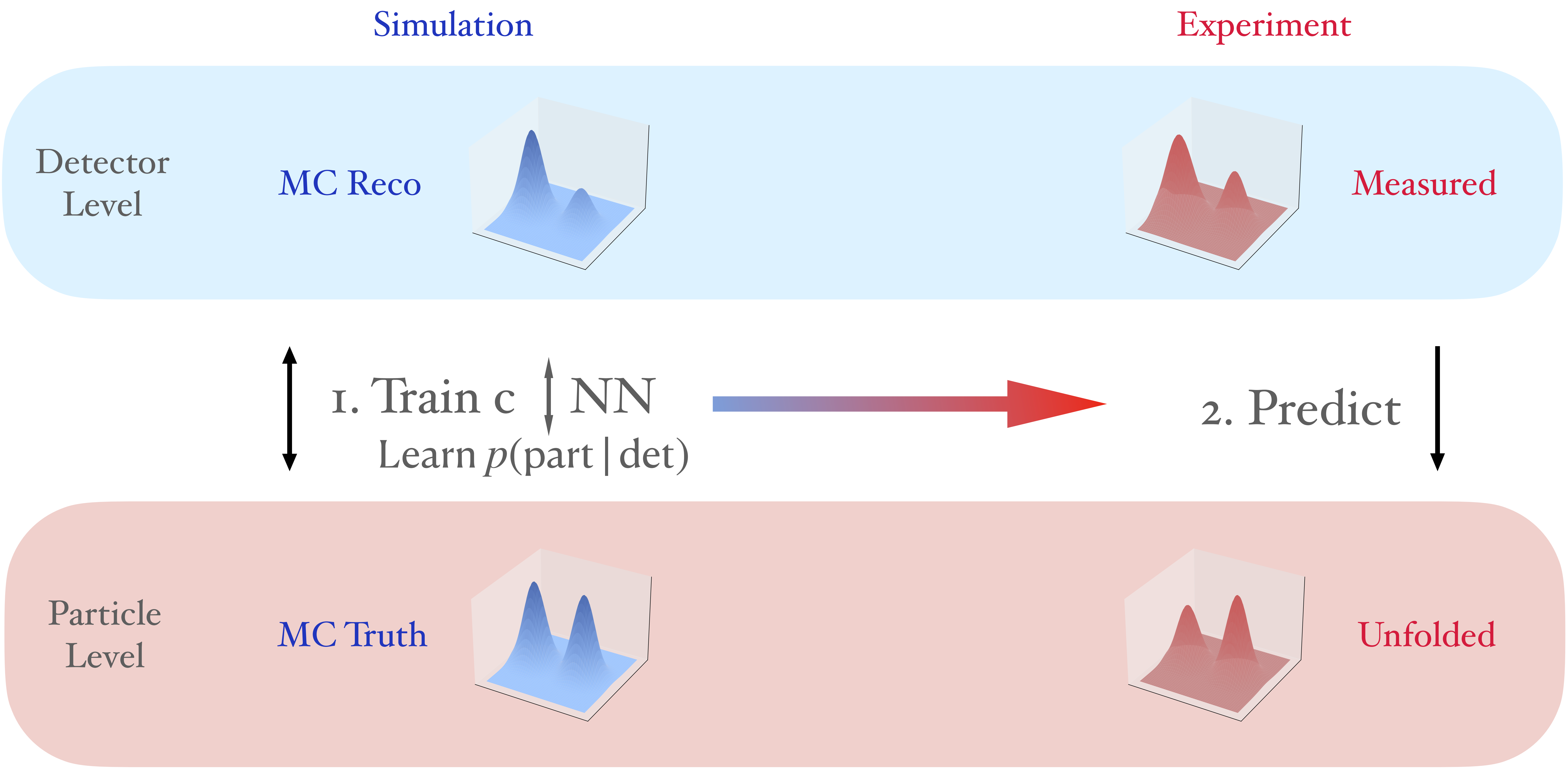
What about inversion?

Inverting the simulation chain



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

Unfolding with generative networks



Simulation

Experiment

Detector Level

MC Reco

Measured

1. Train c ↕ NN

Learn $p(\text{part}\,|\,\text{det})$

2. Predict

Particle Level

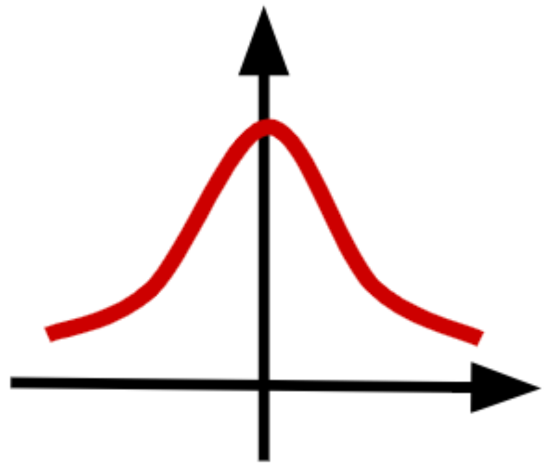
MC Truth

Unfolded

cINN unfolding

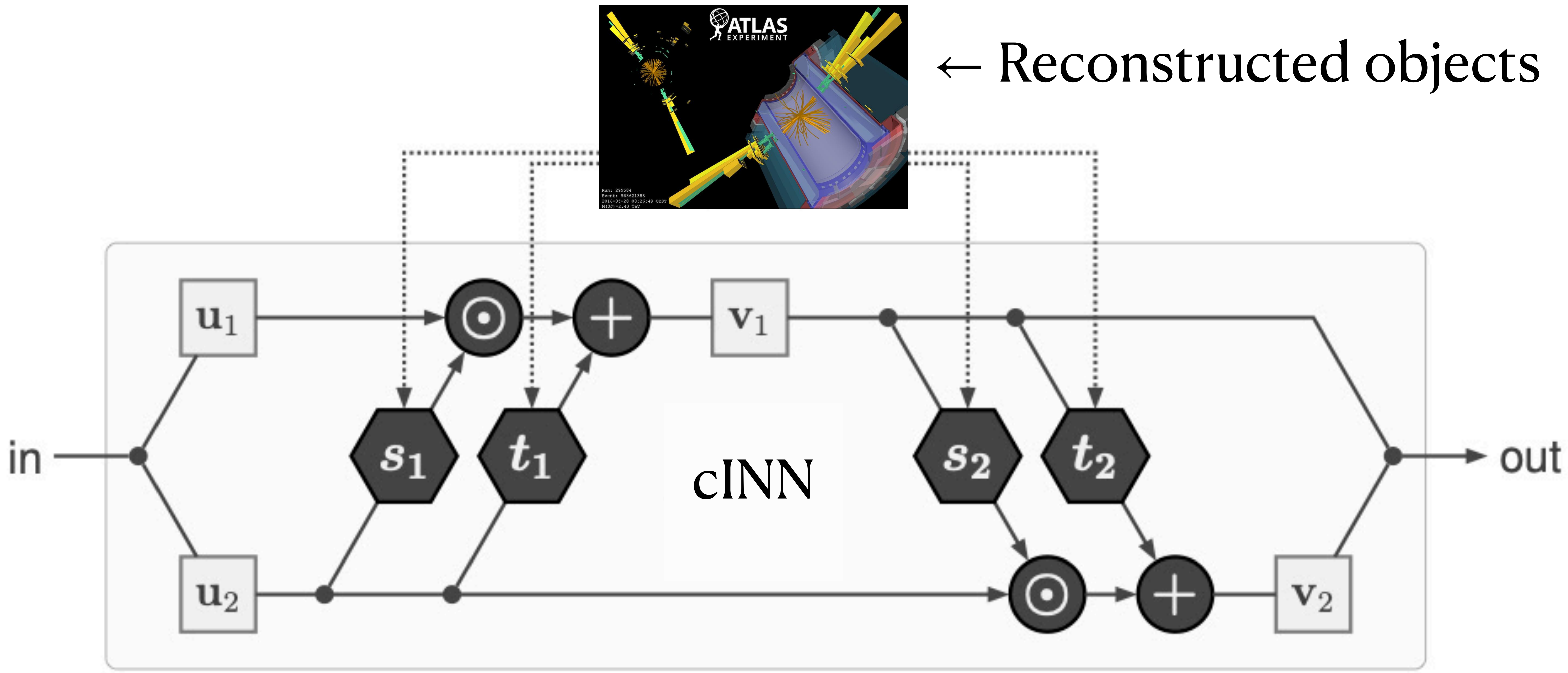
High-dimensional. Bin independent. Robust.

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

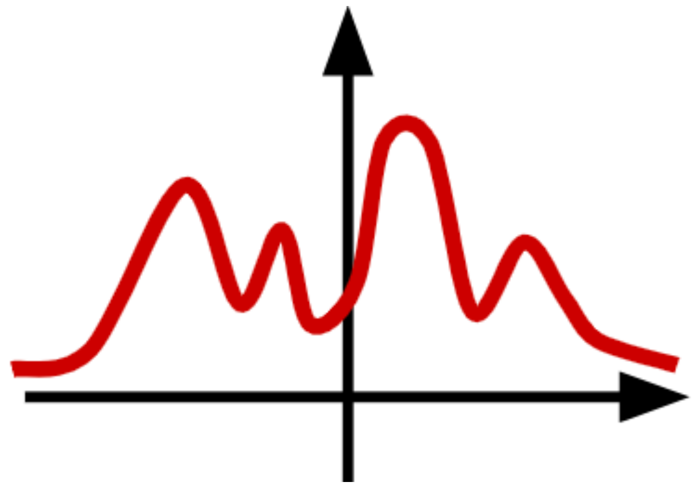


Normal distribution

$z \sim \mathcal{N}$



← Reconstructed objects



Probability distribution

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

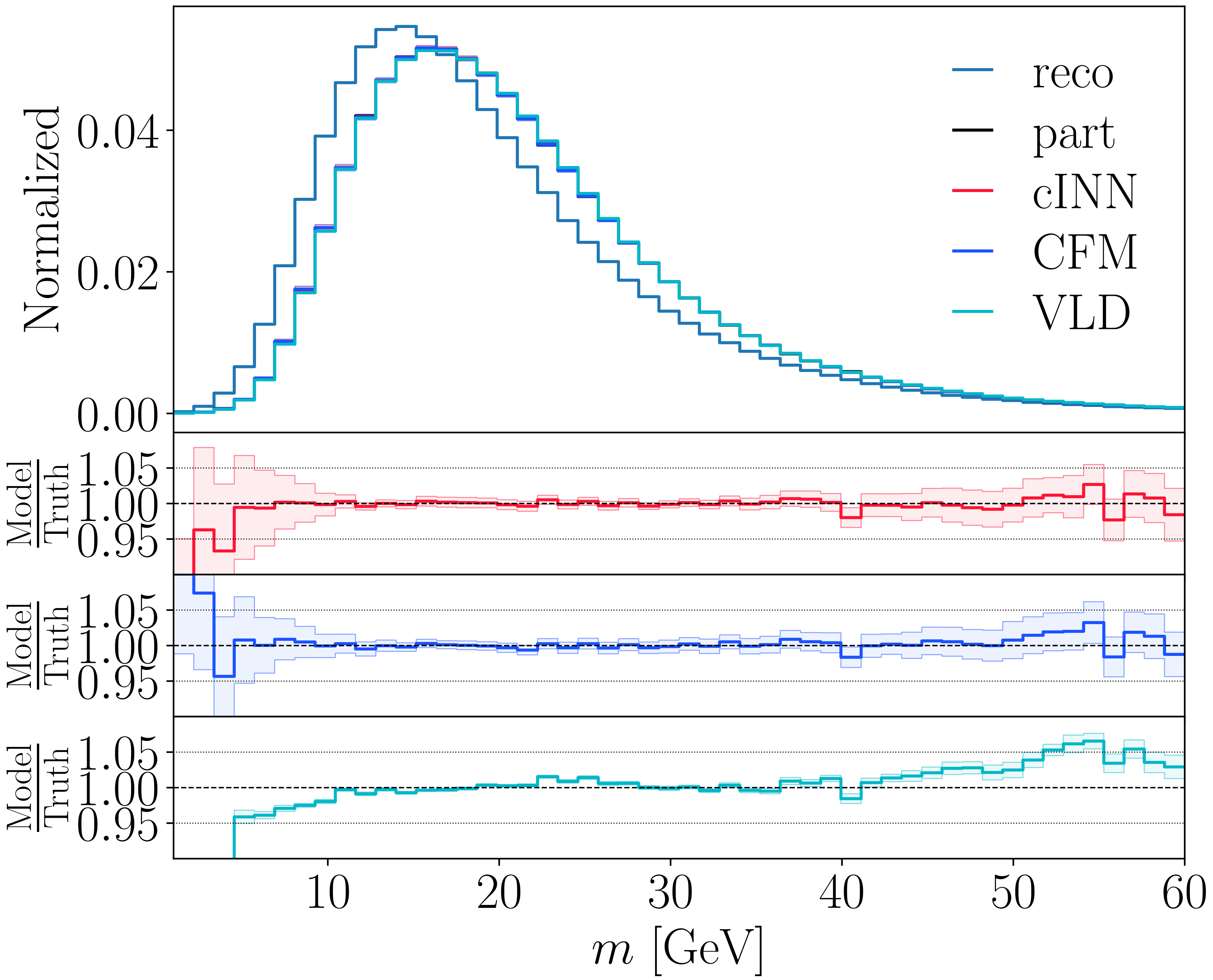
Training

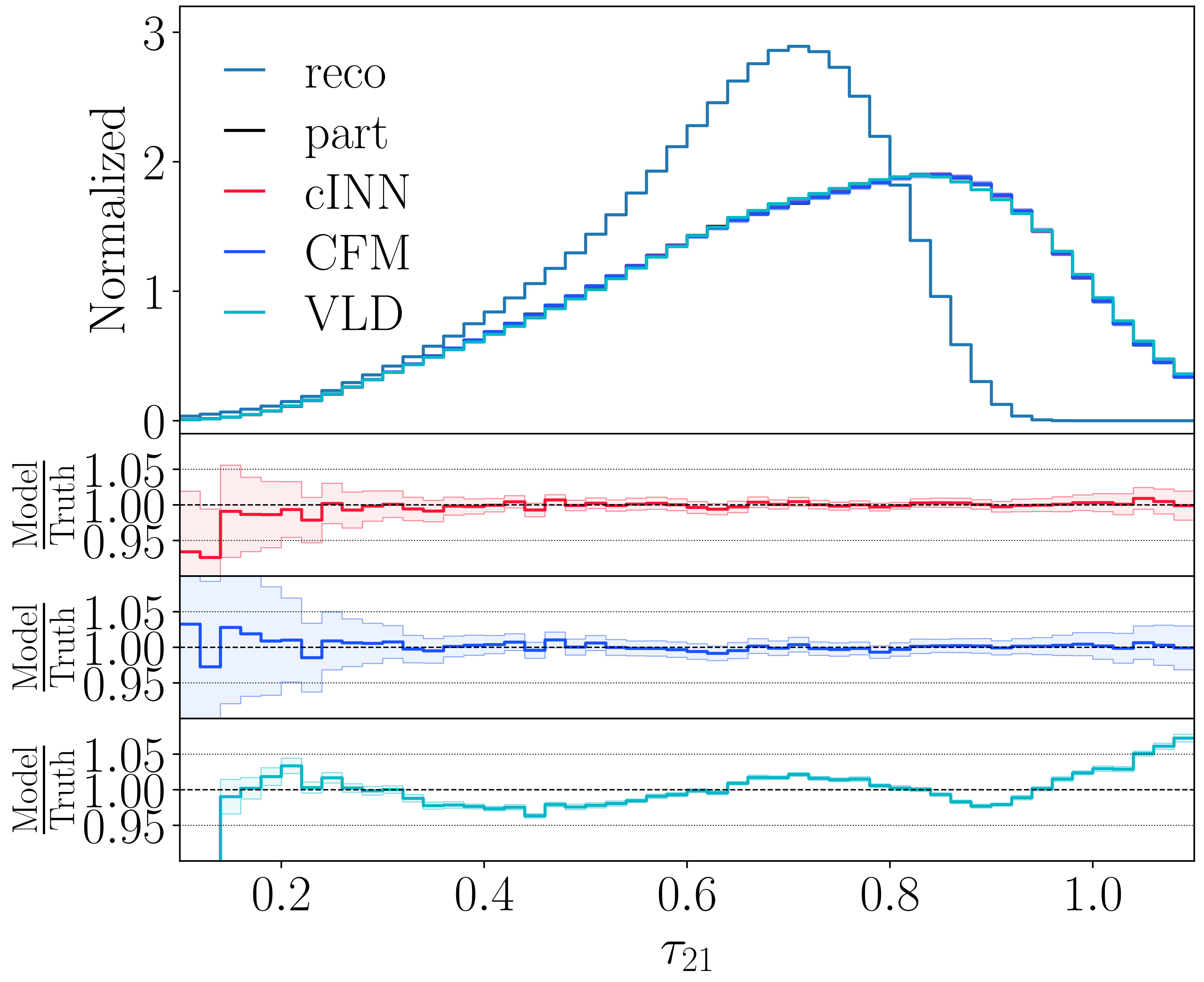
Unfolding

$$\begin{aligned}\mathscr{L} &= \log p(\theta \,|\, x, reco) \\ &= \log p(z \,|\, \theta, reco) + \log J_{NN} + p(\theta)\end{aligned}$$

Unfolding Z+jets events with cINN and diffusion networks

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

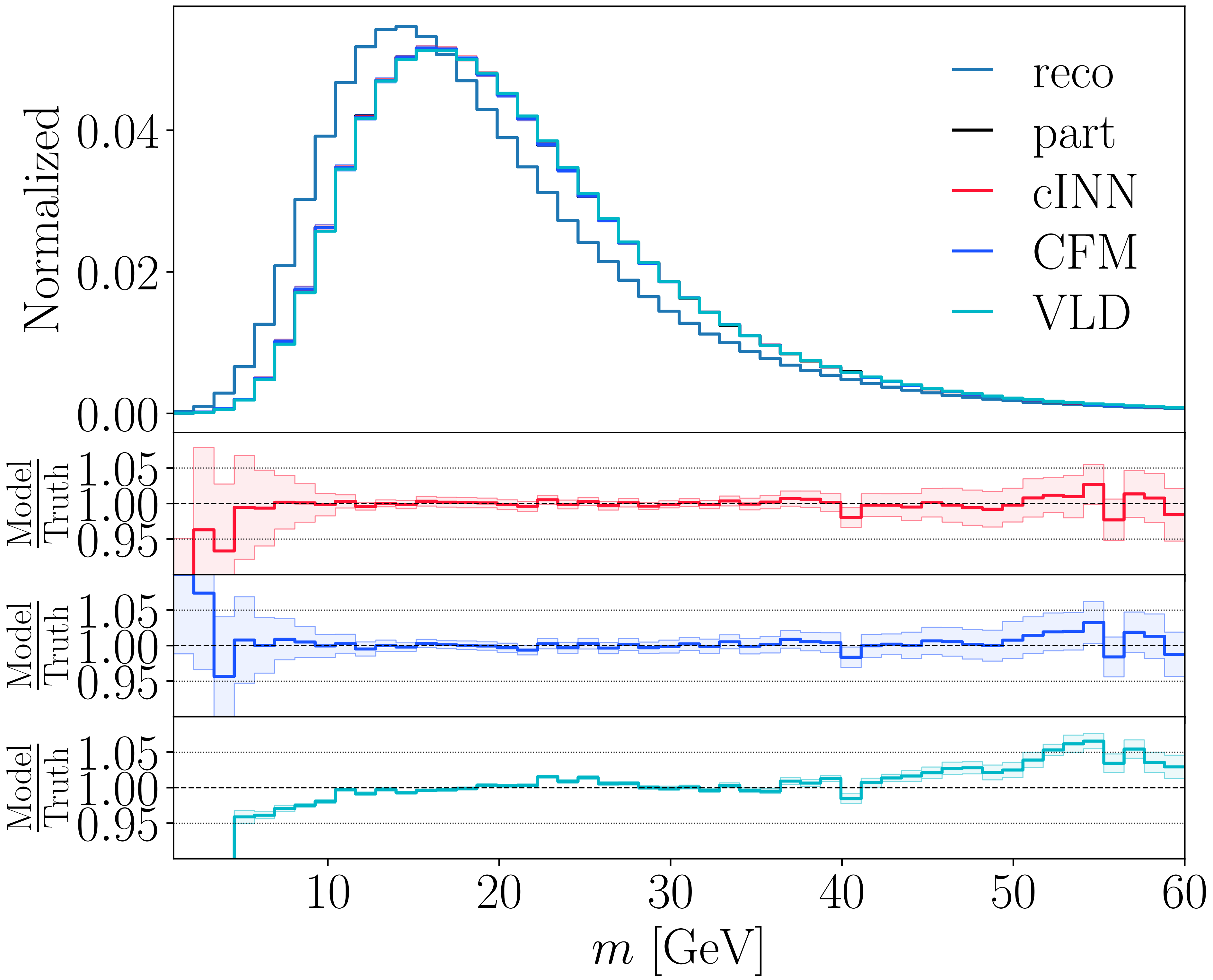


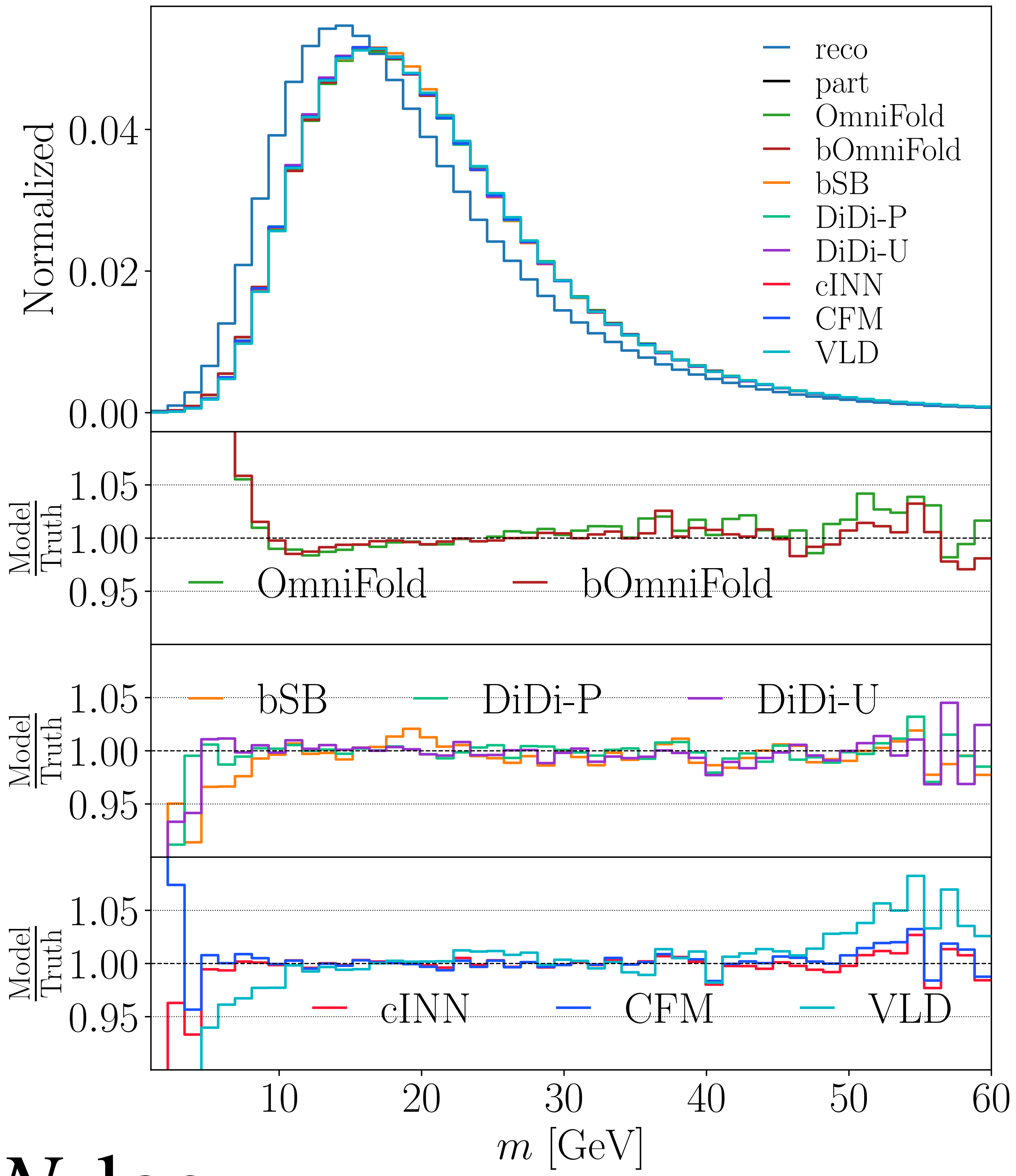


Observables $m, \tau_{21}, w, N, \log \rho, z_g$

Unfolding Z+jets events

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

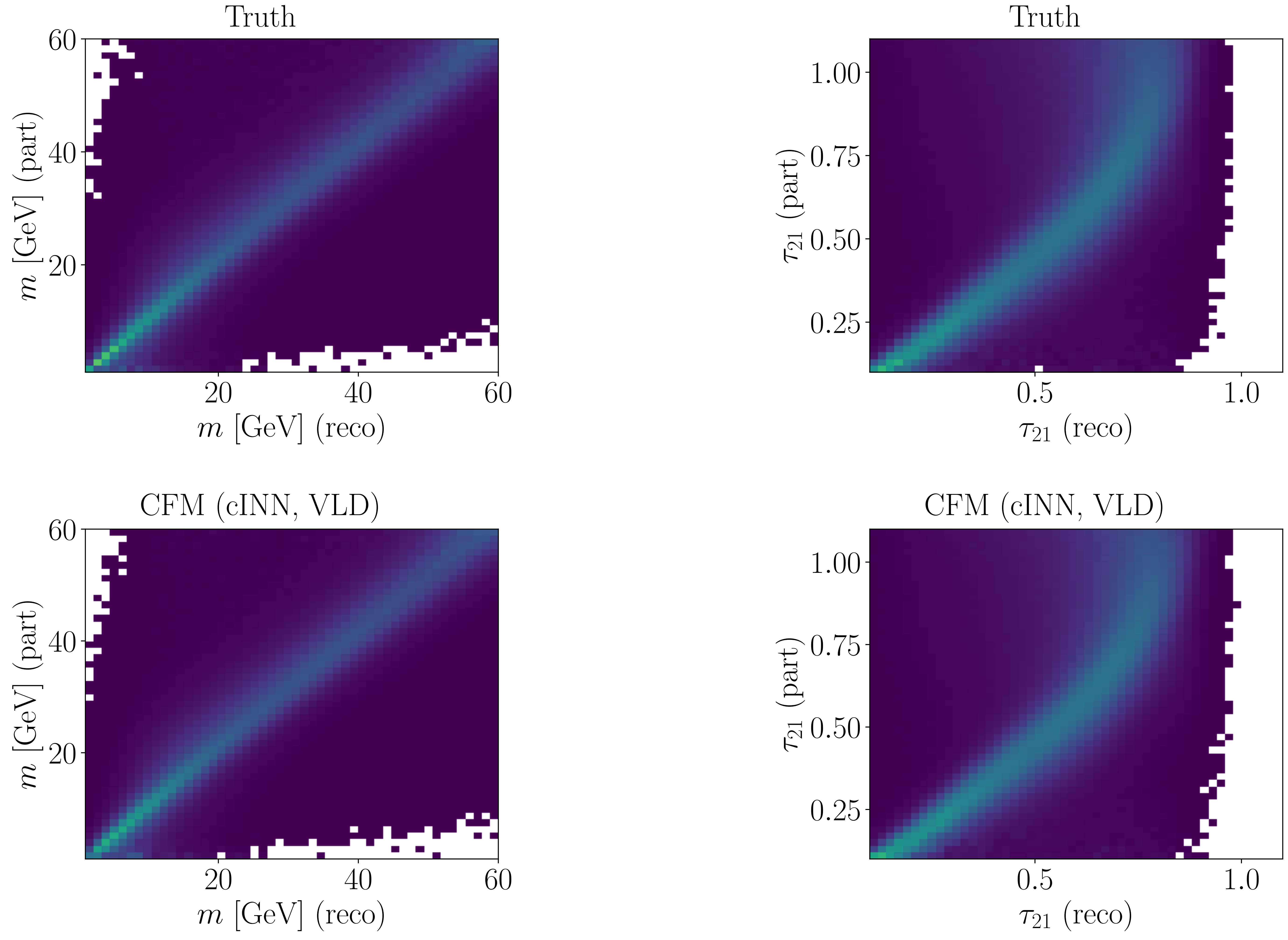




Observables $m, \tau_{21}, w, N, \log \rho, z_g$

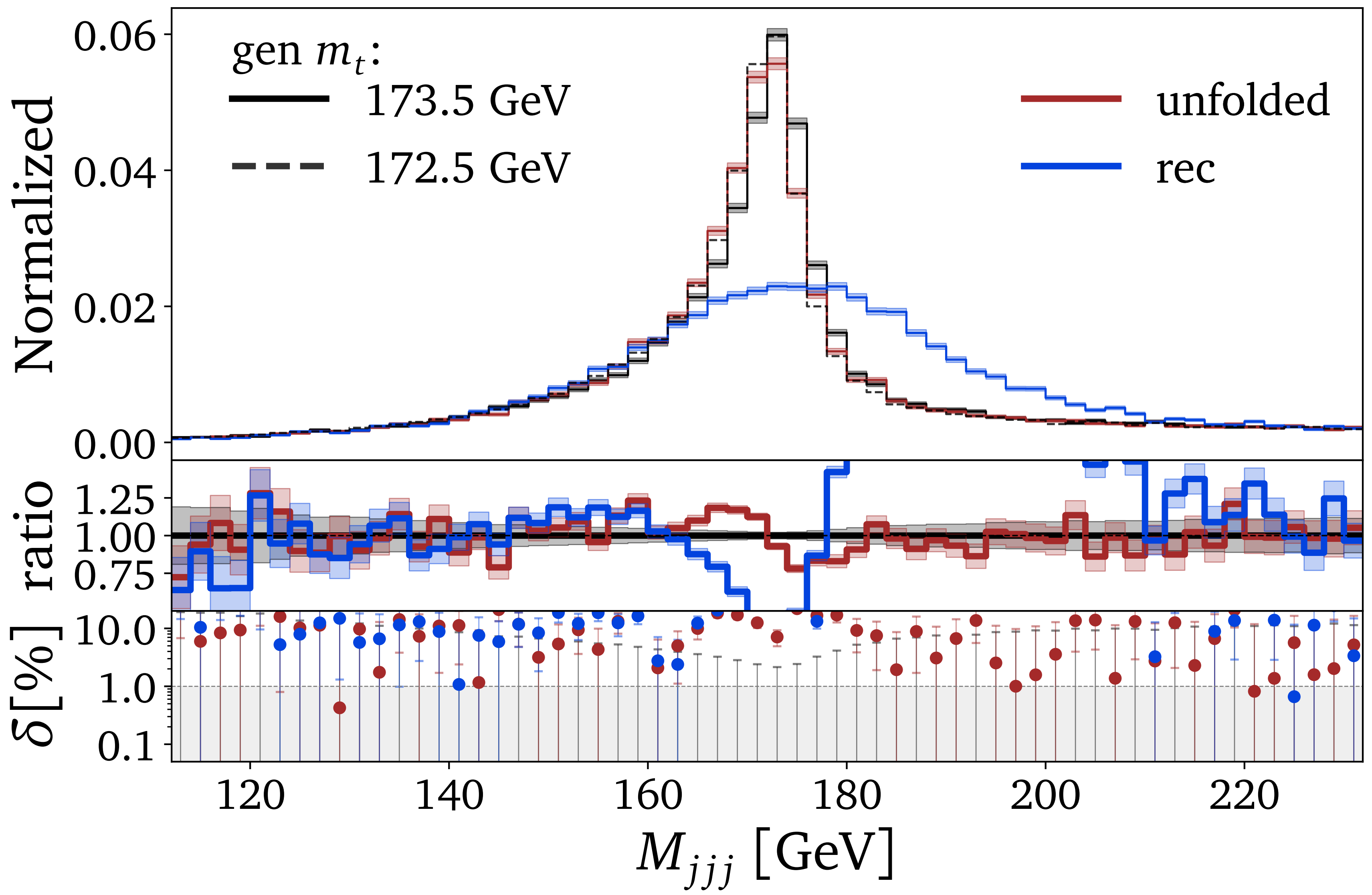
Correlations

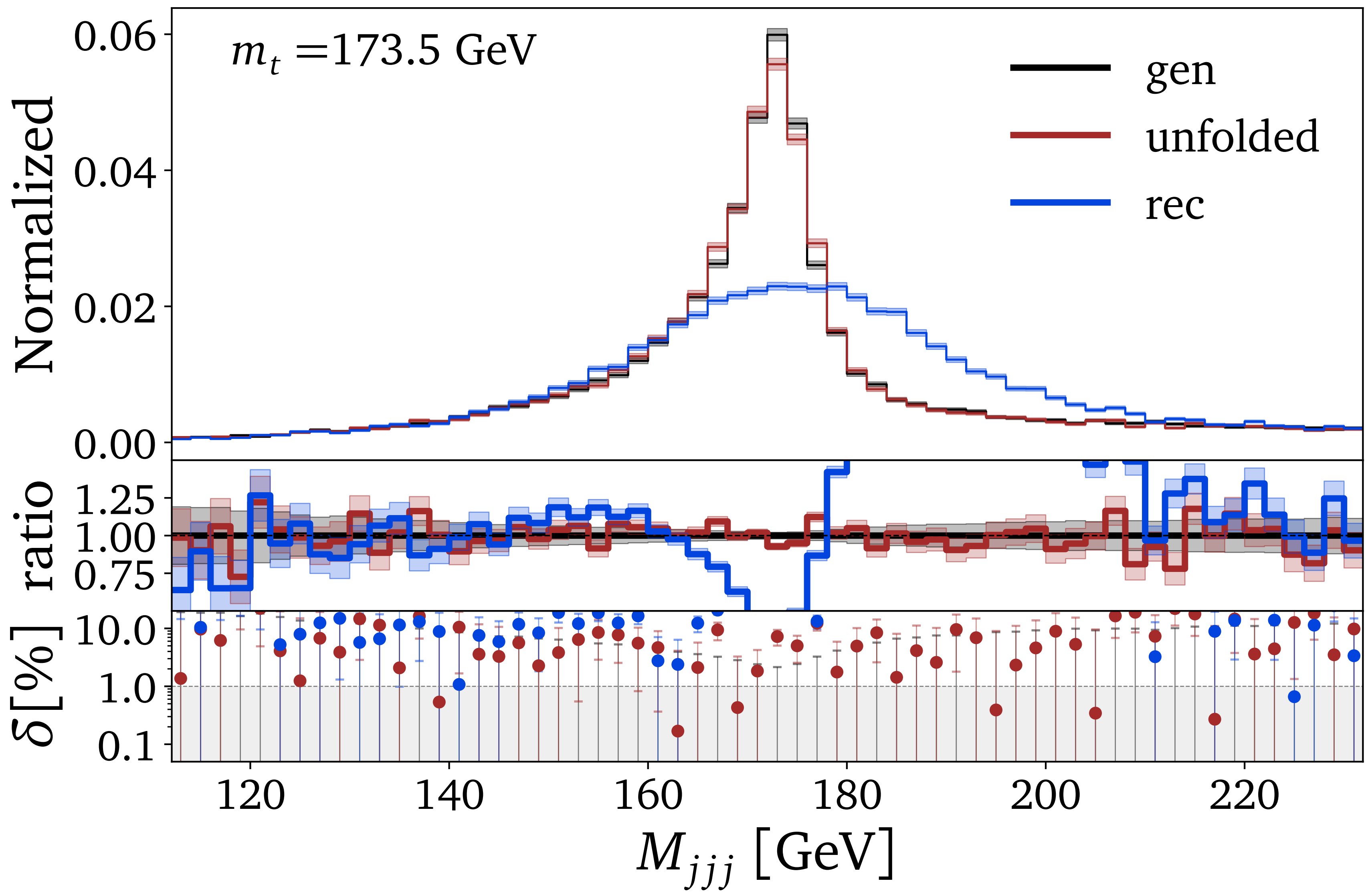
Z+jets: reco vs particle level, jet mass & subjettiness ratio



Application in top mass measurement

L. Favaro, R. Kogler, A. Paasch, S. Palacios, T. Plehn, D. Schwarz





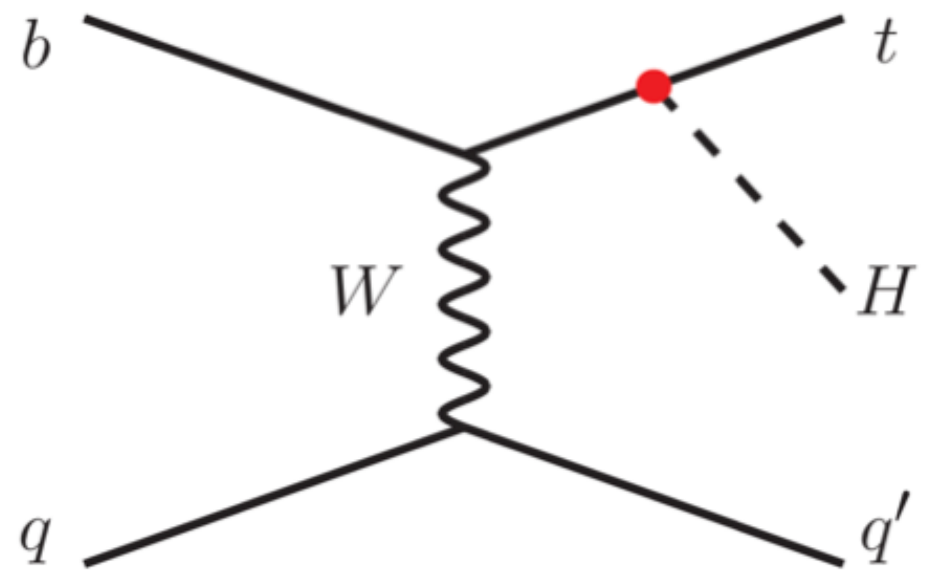
Trained on $m_t = 172.5$ GeV
Applied to $m_t = 173.5$ GeV
-> strongly biased

Unbiased unfolding
Training on $m_t = 169.5, 172.5$ & 175.5 GeV

Additional input (training & evaluation):
Batch-wise weighted median of M_{jjj}@reco

Beyond unfolding: Enabling the MEM

T. Heimel, N. Huetsch, R. Winterhalder, A. Butter, T. Plehn [2210.00019, 2310.07752]



Matrix element method is based on **untractable** likelihood

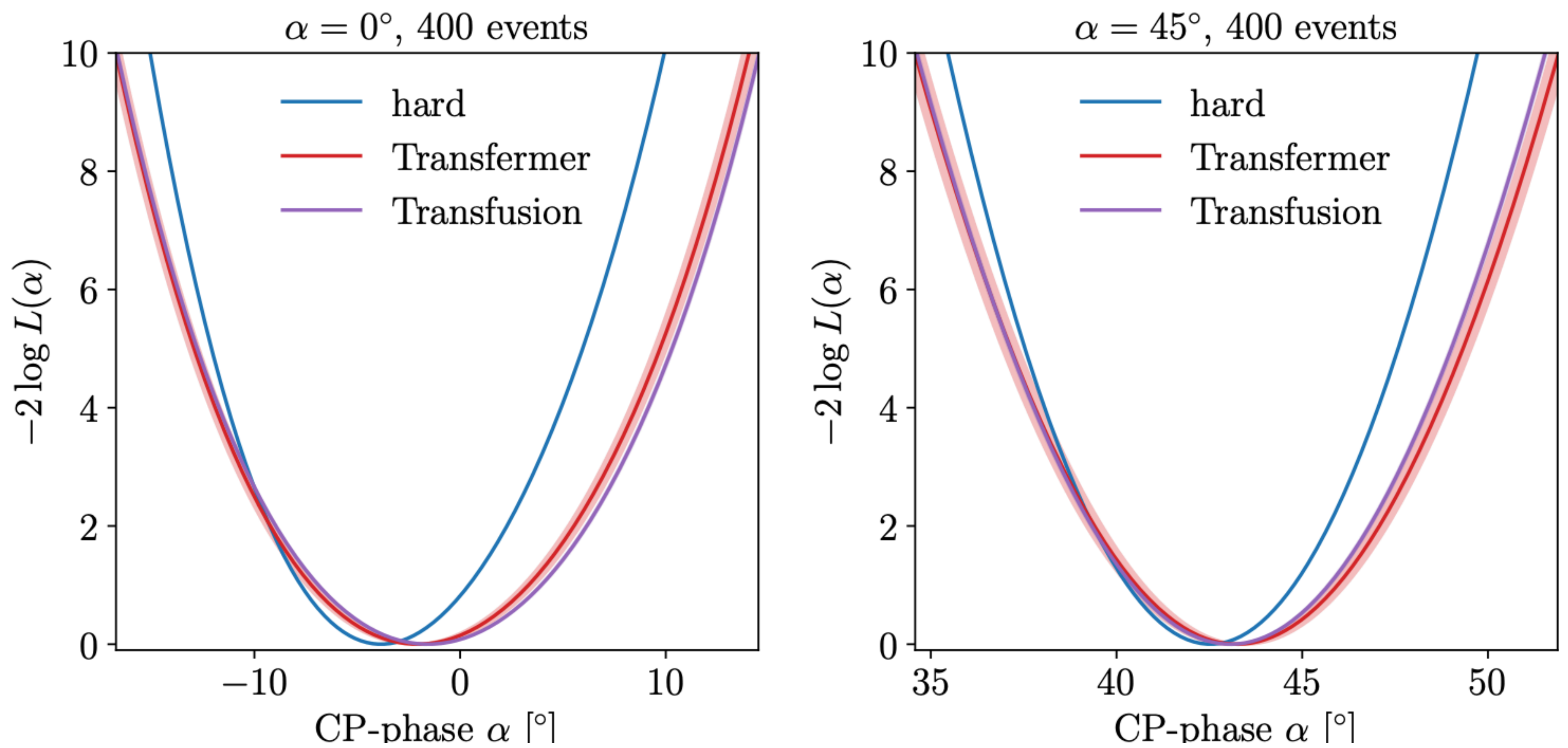
$$p(x_{\text{reco}}|\alpha) = \int \mathrm{d}x_{\text{hard}}\ \underbrace{p(x_{\text{hard}}|\alpha)}_{\text{diff. CS}}\ \underbrace{p(x_{\text{reco}}|x_{\text{hard}},\alpha)}_{\text{estimate with network}}$$

Single Higgs production
with anomalous non-CP conserving Higgs coupling

Problem: integration over full phase space of the hard scattering

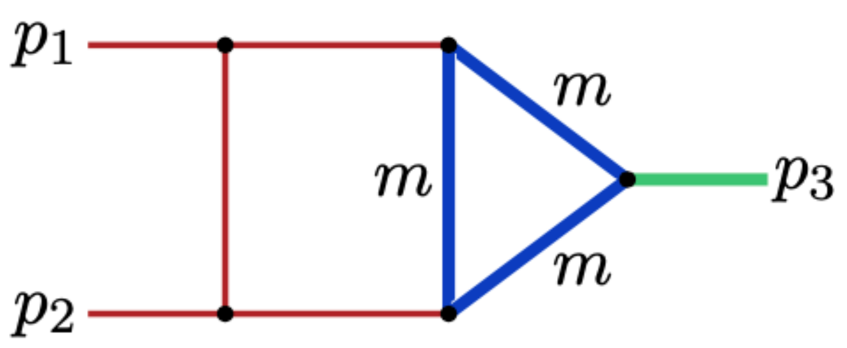
Solution: Use unfolding cINN to sample x_{hard}

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

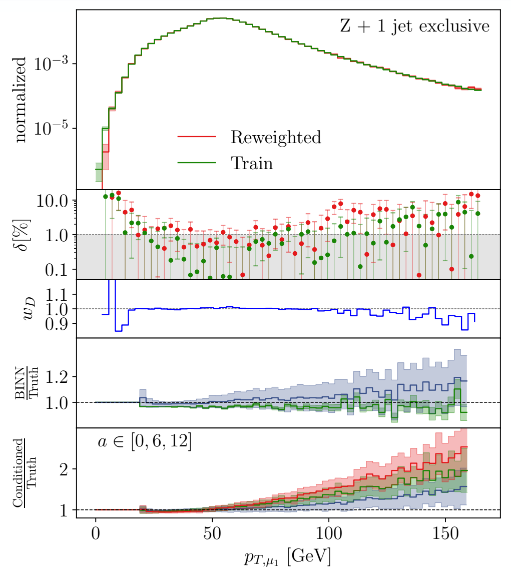


Machine learning up and down the simulation chain

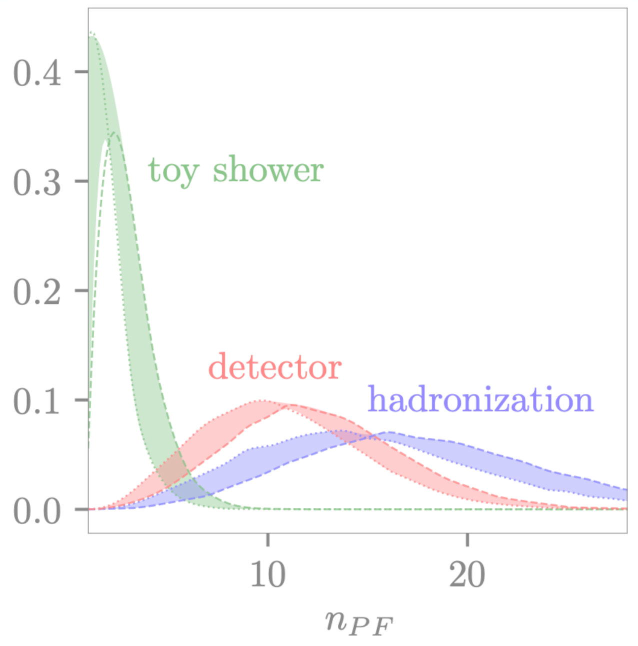
Loop amplitudes



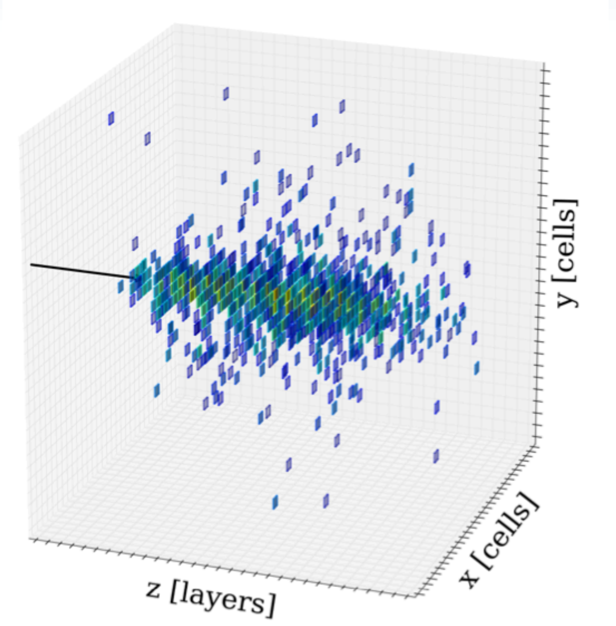
Event generation

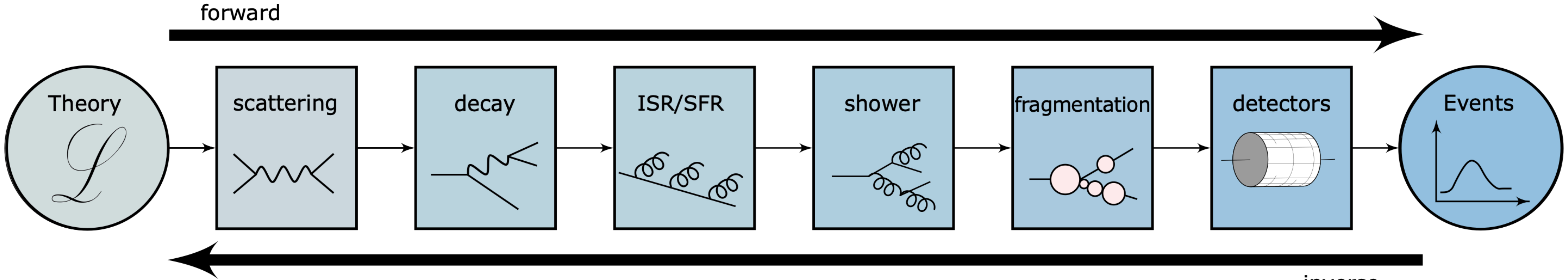


Shower



Detector simulation





forward

Theory

scattering

decay

ISR/SFR

shower

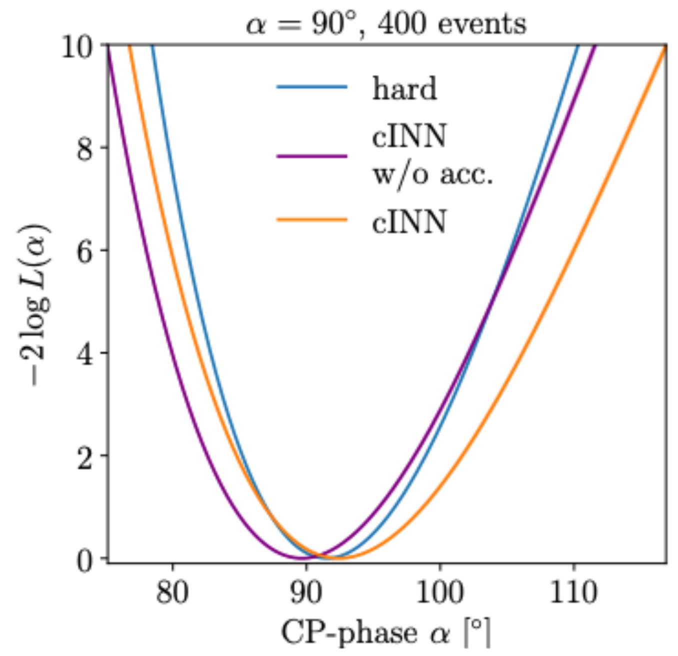
fragmentation

detectors

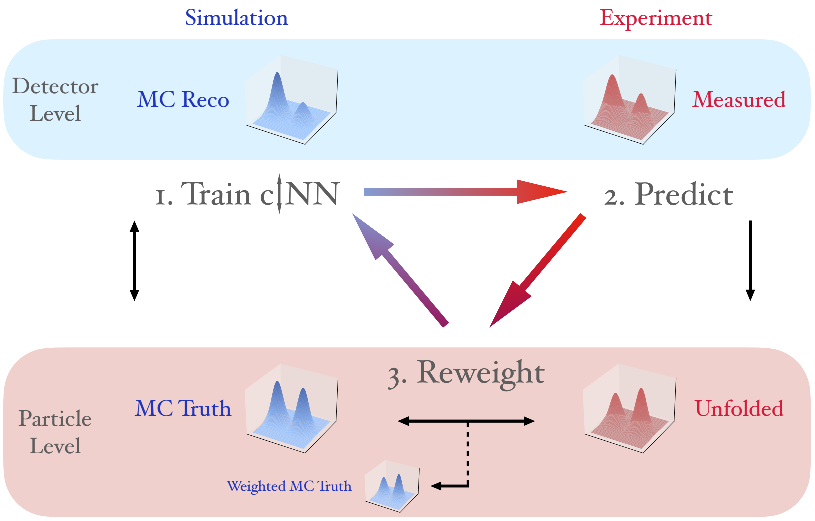
Events

inverse

MEM



Unfolding



Simulation

Experiment

Detector Level

MC Reco

Measured

1. Train c|NN

2. Predict

3. Reweight

Particle Level

MC Truth

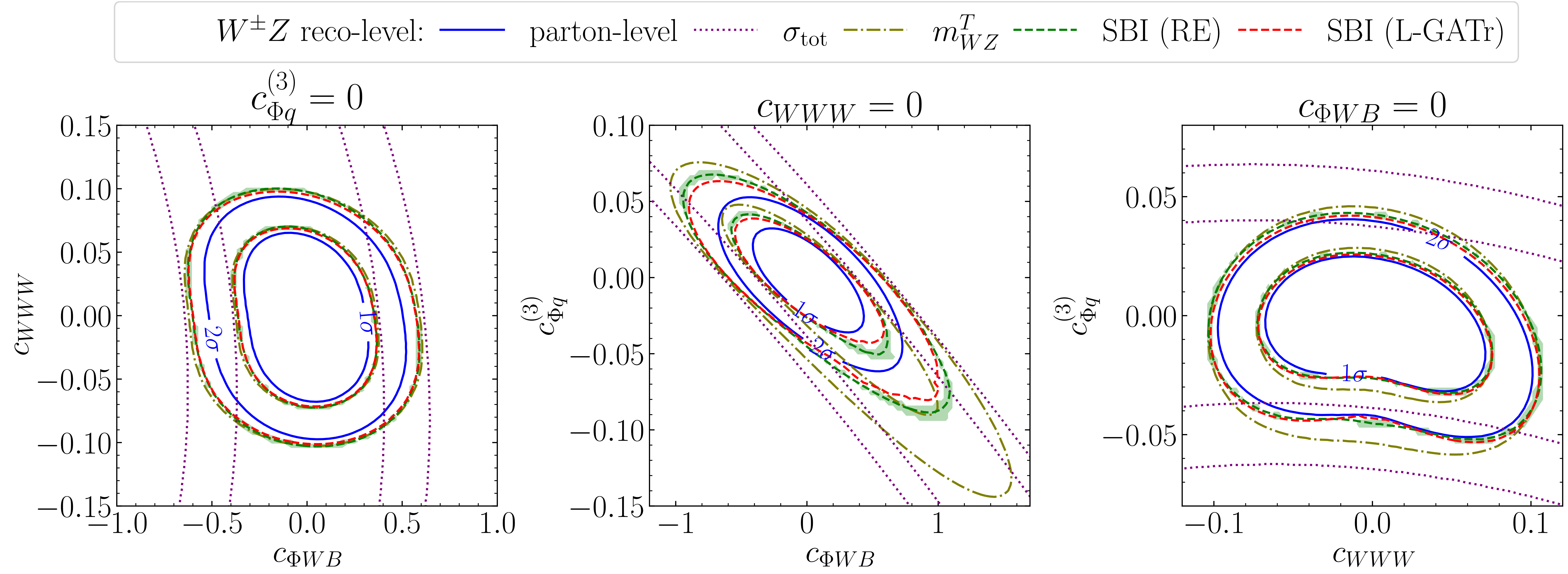
Unfolded

Weighted MC Truth

Ready for more data ...

SMEFT analyses

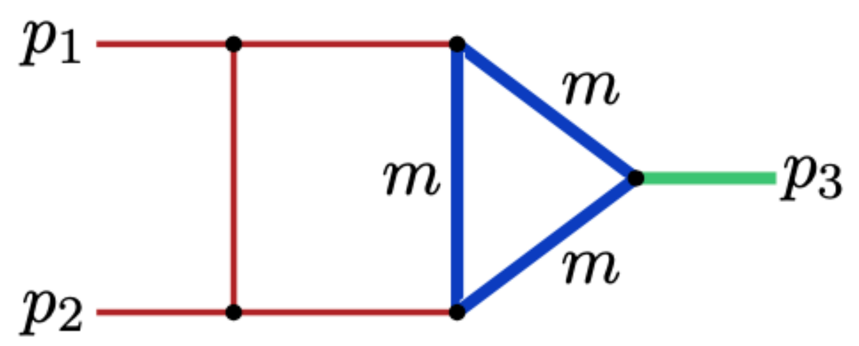
[2409.XXXXX]



Multi-loop calculations with INNs

Profiting from the Jacobian

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

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

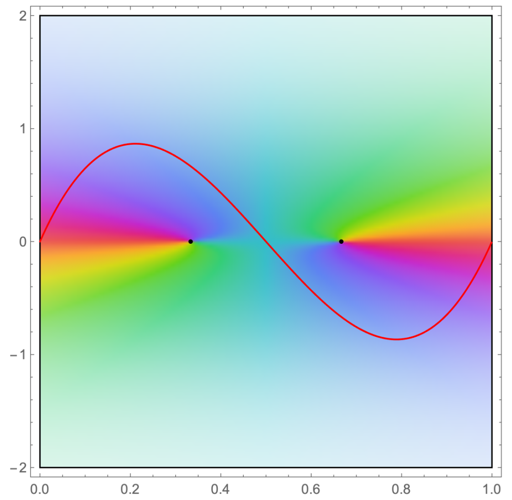
$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j-1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, I(\vec{x})$$

Rewrite with Feynman parameters

Still contains singularities

Solved by contour deformation due to Cauchy's theorem

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

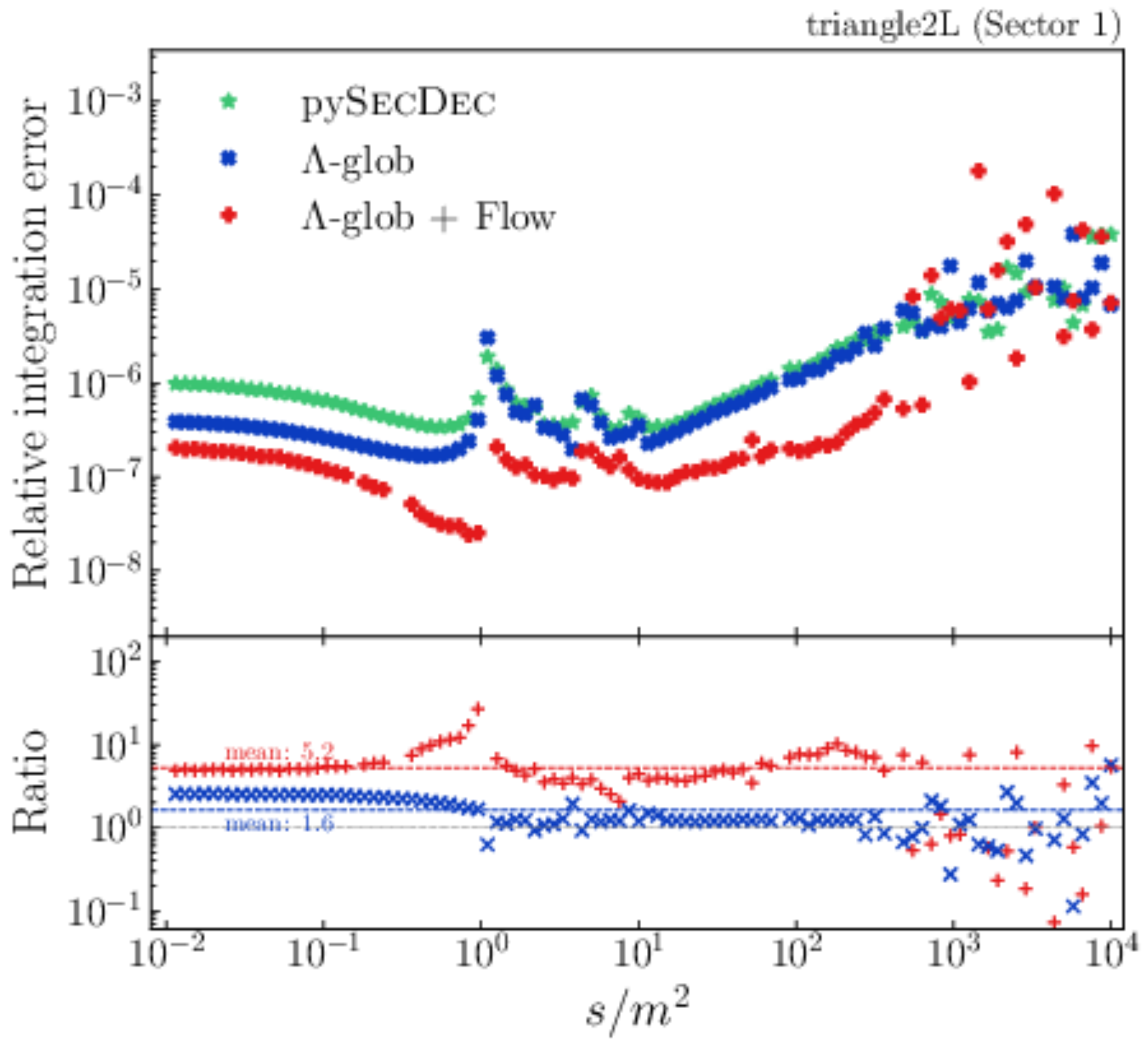


Optimal parametrization = minimal variance

Turn it into an ML Problem

Parametrization → $z = \mathrm{INN}(x)$

Variance → $\mathscr{L}$



R. Winterhalder, et al. [2112.09145]