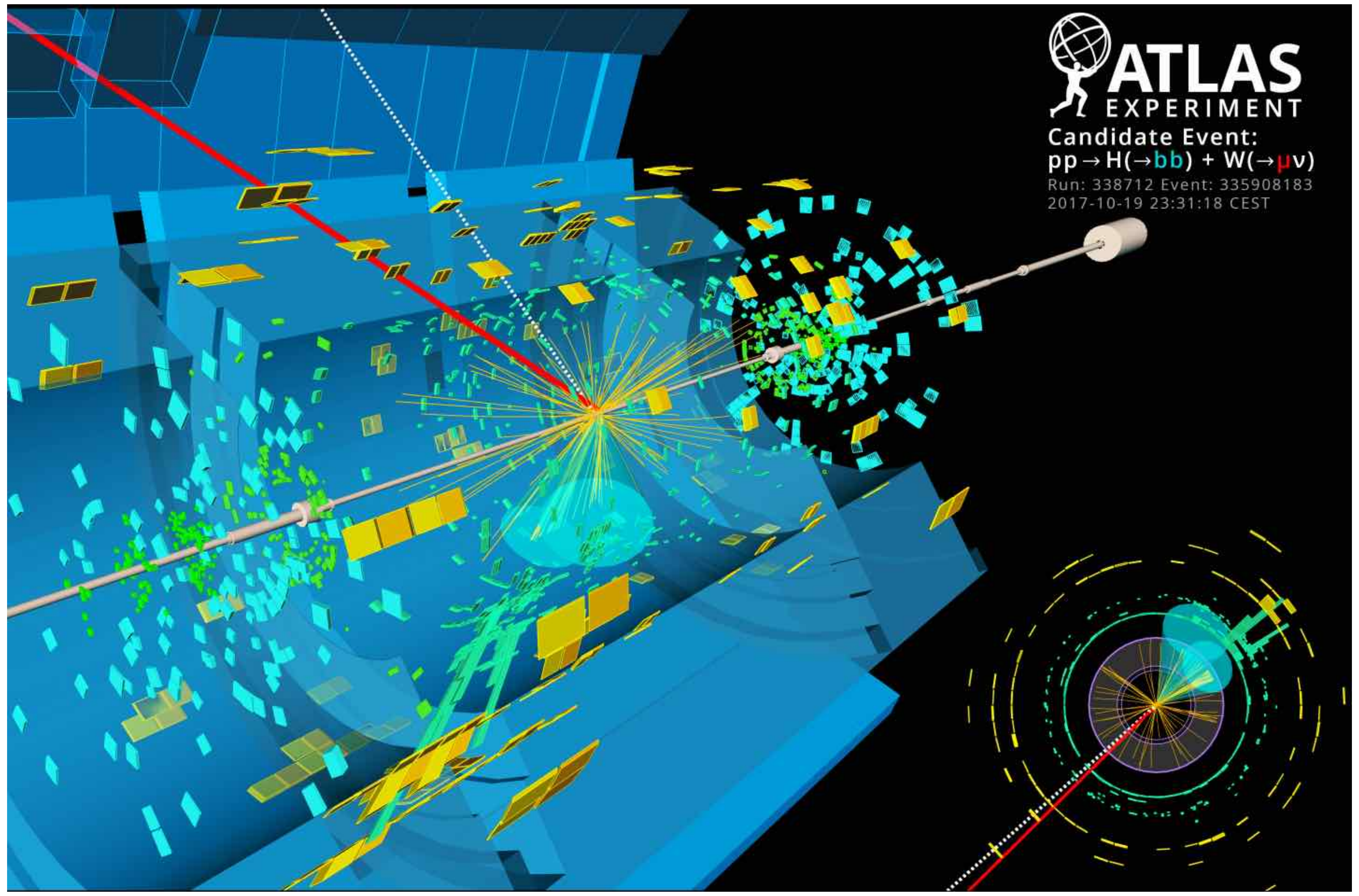
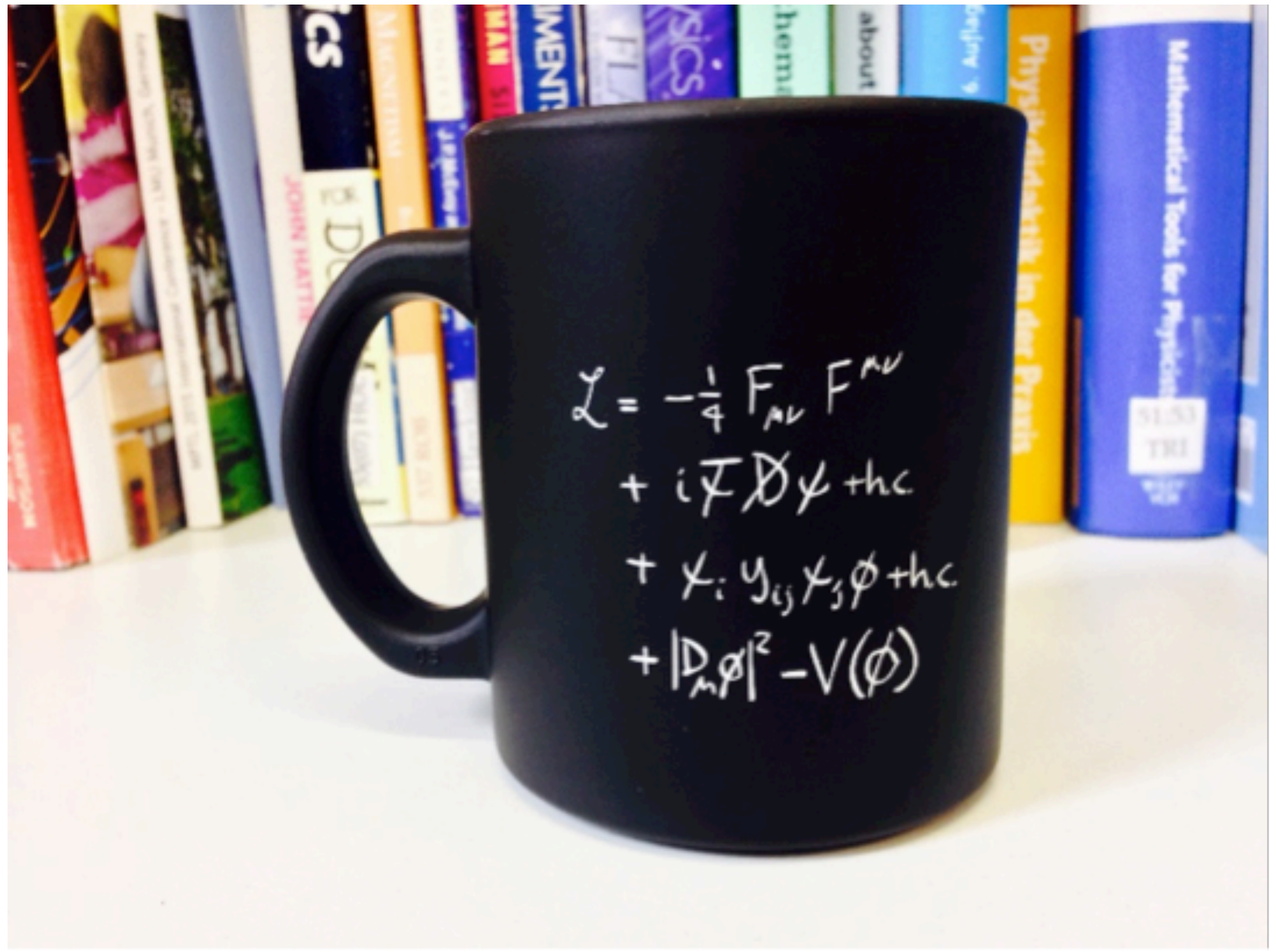
Boosting event generation & unfolding with generative networks

Simtech Summer School in Stuttgart

Anja Butter, ITP Heidelberg

A short overview on LHC physics





Setting

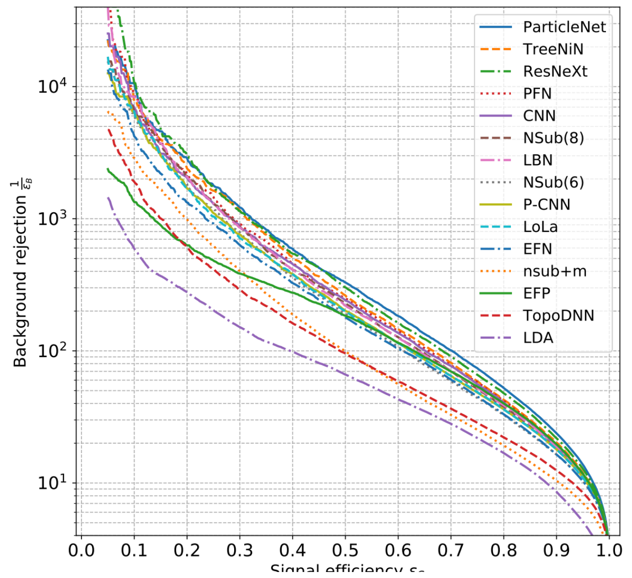
- Large Hadron Collider at CERN
- Proton collisions at 13 TeV
- **Huge** dataset ~1Pb/s before trigger selection

Goal

- Understand full dataset from **1st principles**
- Find signs of new physics (1 in 10^{10} or less)
- .... and it looks exactly like the background

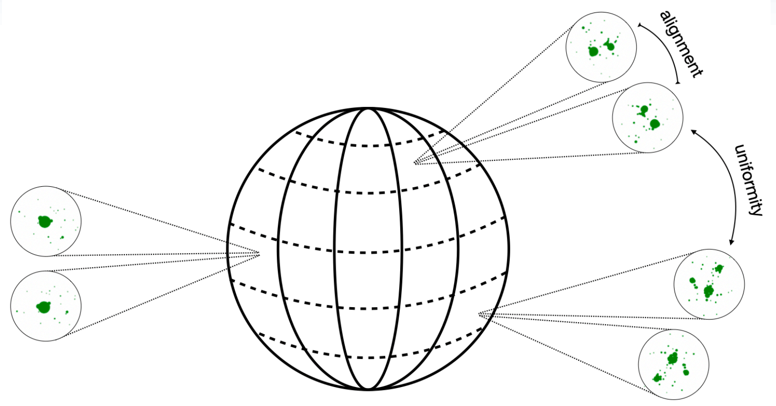
ML for big data in particle physics

Top tagging



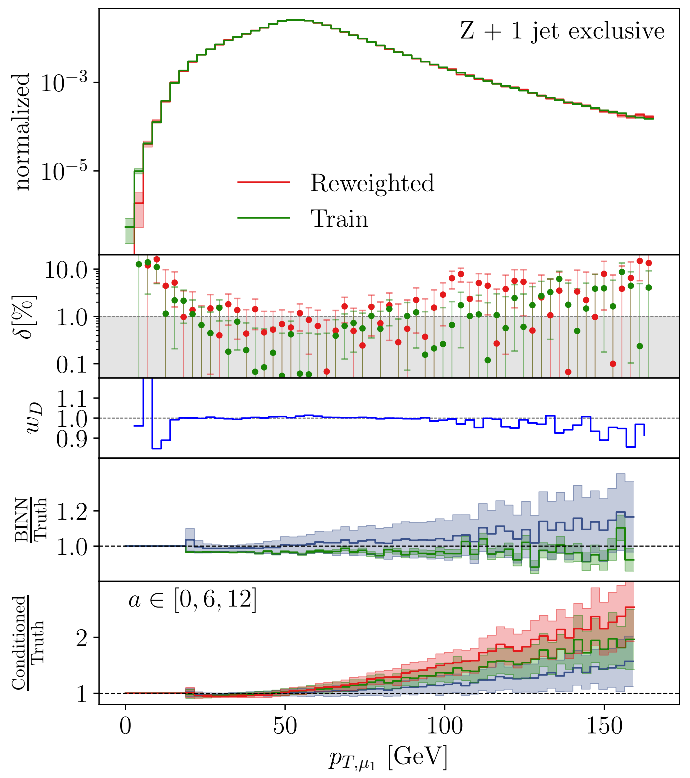
G. Kasieczka et al. [1902.09914]

Anomaly detection



B. Dillon et al. [2108.04253]

Event generation



A. Butter et al. [2110.13632]

Classification

Generative models

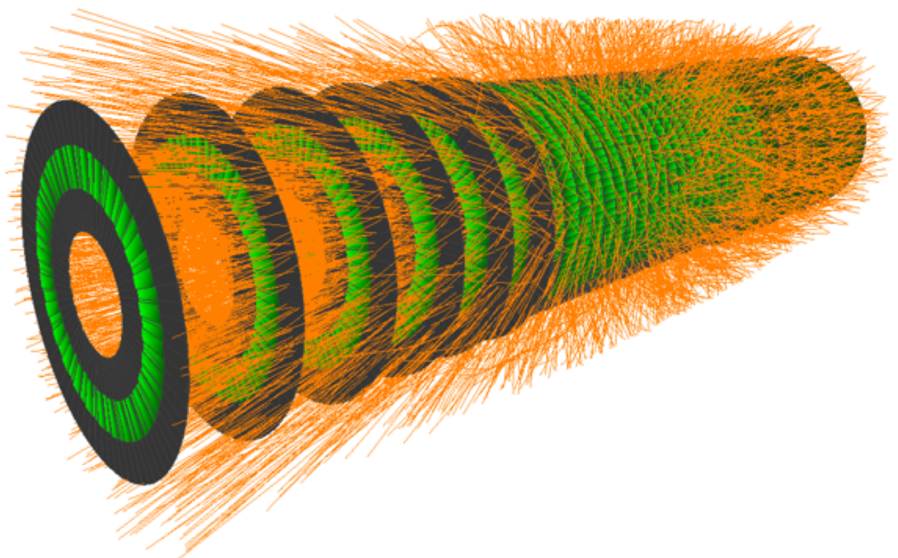
Graph networks

Bayesian networks

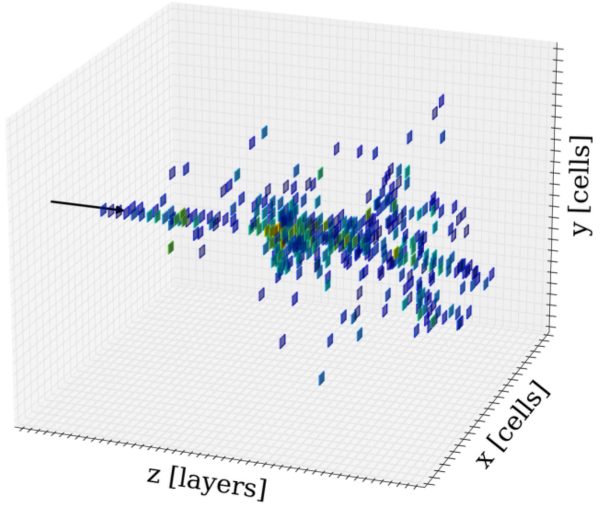
Regression

Track reconstruction

Kaggle challenge

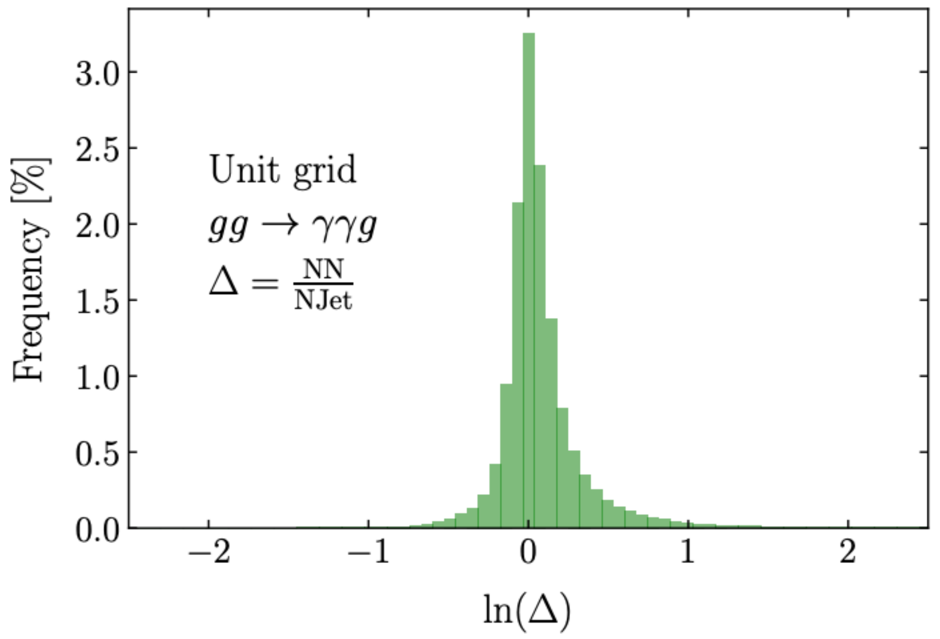


Detector simulation



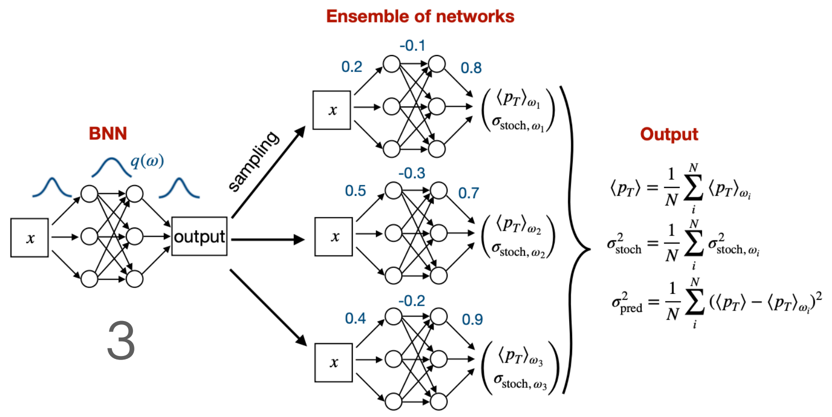
E. Buhmann et al. [2112.09709]

Amplitude estimation



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

Jet calibration & uncertainties

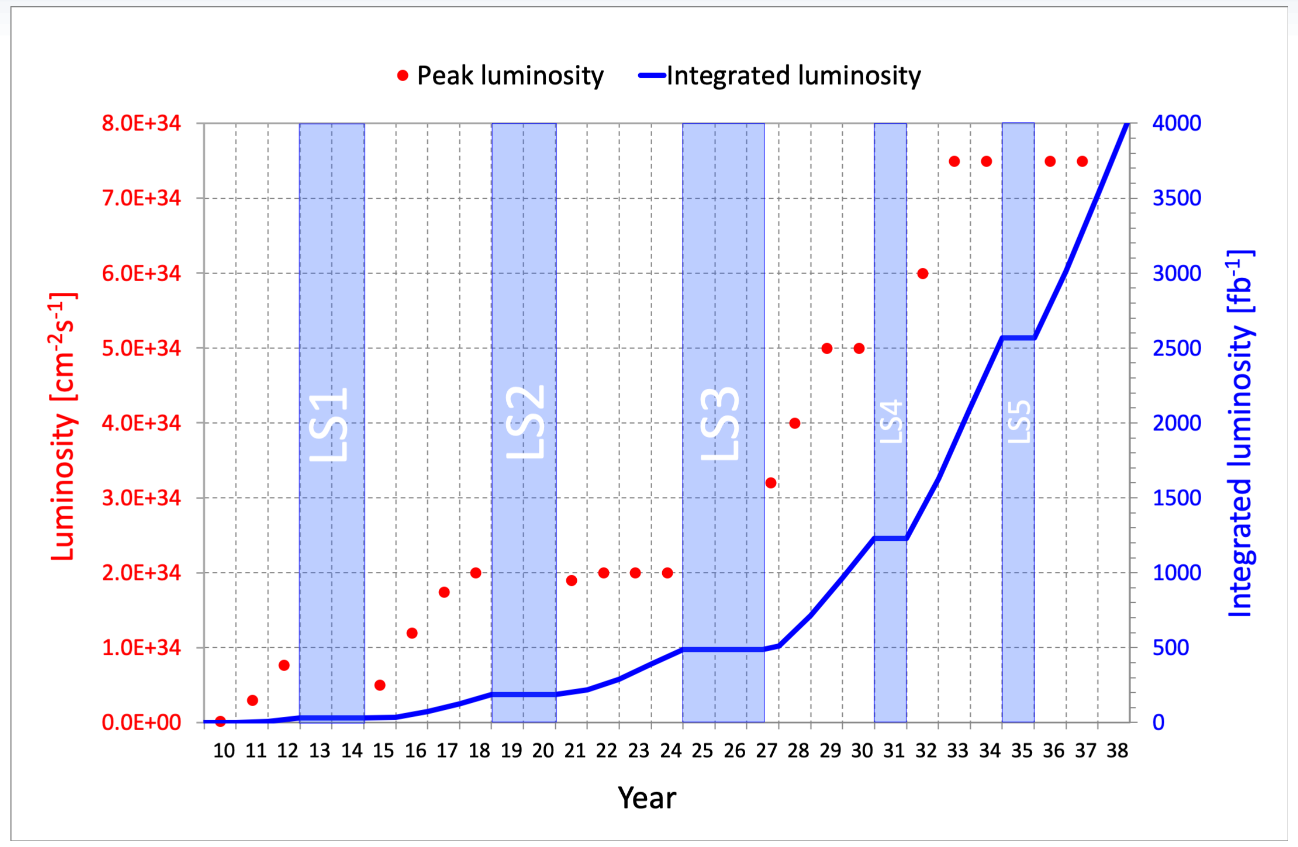


G. Kasieczka et al. [2003.11099]

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

Open questions towards HL-LHC

A biased selection



- Facing **25 times** the amount of data
- What do we need to understand the data? (*read:* find new physics)

- **Precision predictions**
  - Higher order calculations
  - Event generation
  - Detector simulation

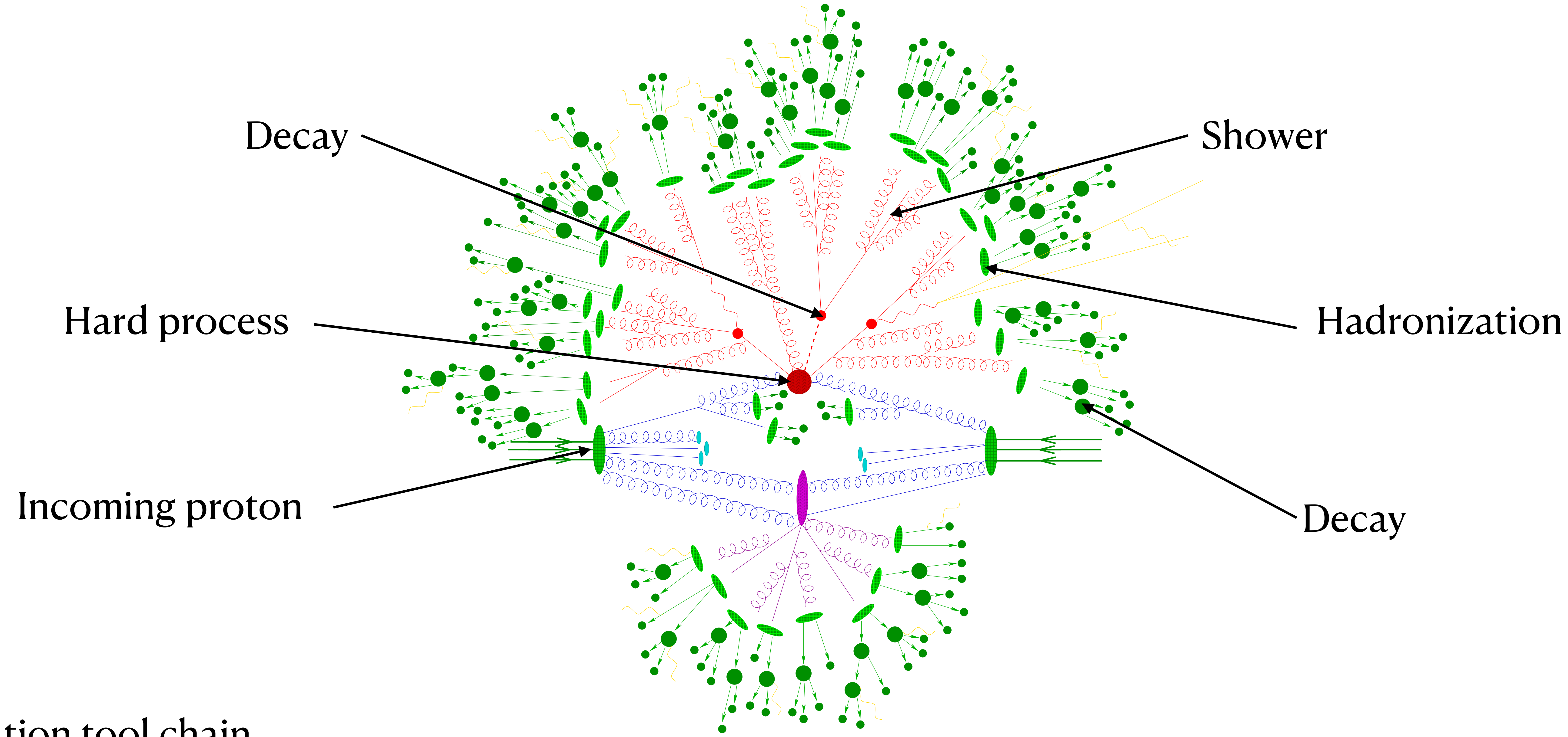
- **Optimized analysis for high-dimensional data**
  - Likelihood free inference
    - Unfolding = Inverse problem
  - How to get most out of data

Problems beyond supervised classification/regression → How can machine learning help?

How to
generate LHC events

Event generation at the LHC

A theorist's perspective



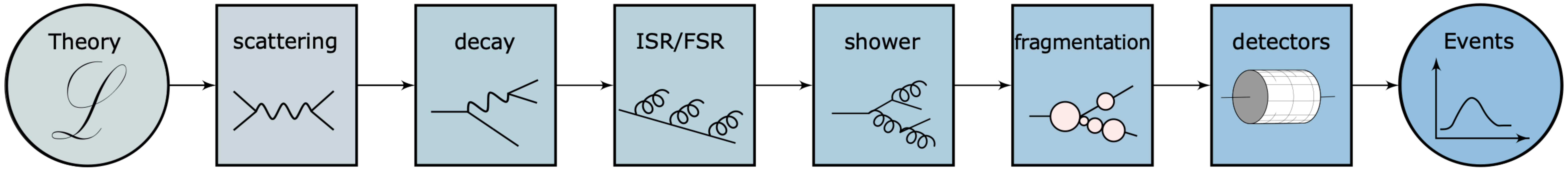
Every line is a particle!

4-vector representation

(E, p_x, p_y, p_z)

with $E^2 = m^2 + \vec{p}^{\,2}$

Simulation tool chain



Monte carlo event generation

1. Generate phase space points

$\rightarrow$ set of four-momenta p_i

2. Calculate event weight

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

Parton density function

Matrix element †

Phase space mapping

3. Unweighting †

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

† 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

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

Approximating Amplitudes

- Approximate Amplitude $\mathcal{M}(x_1, x_2, p_1, \ldots, p_n)$ with NN

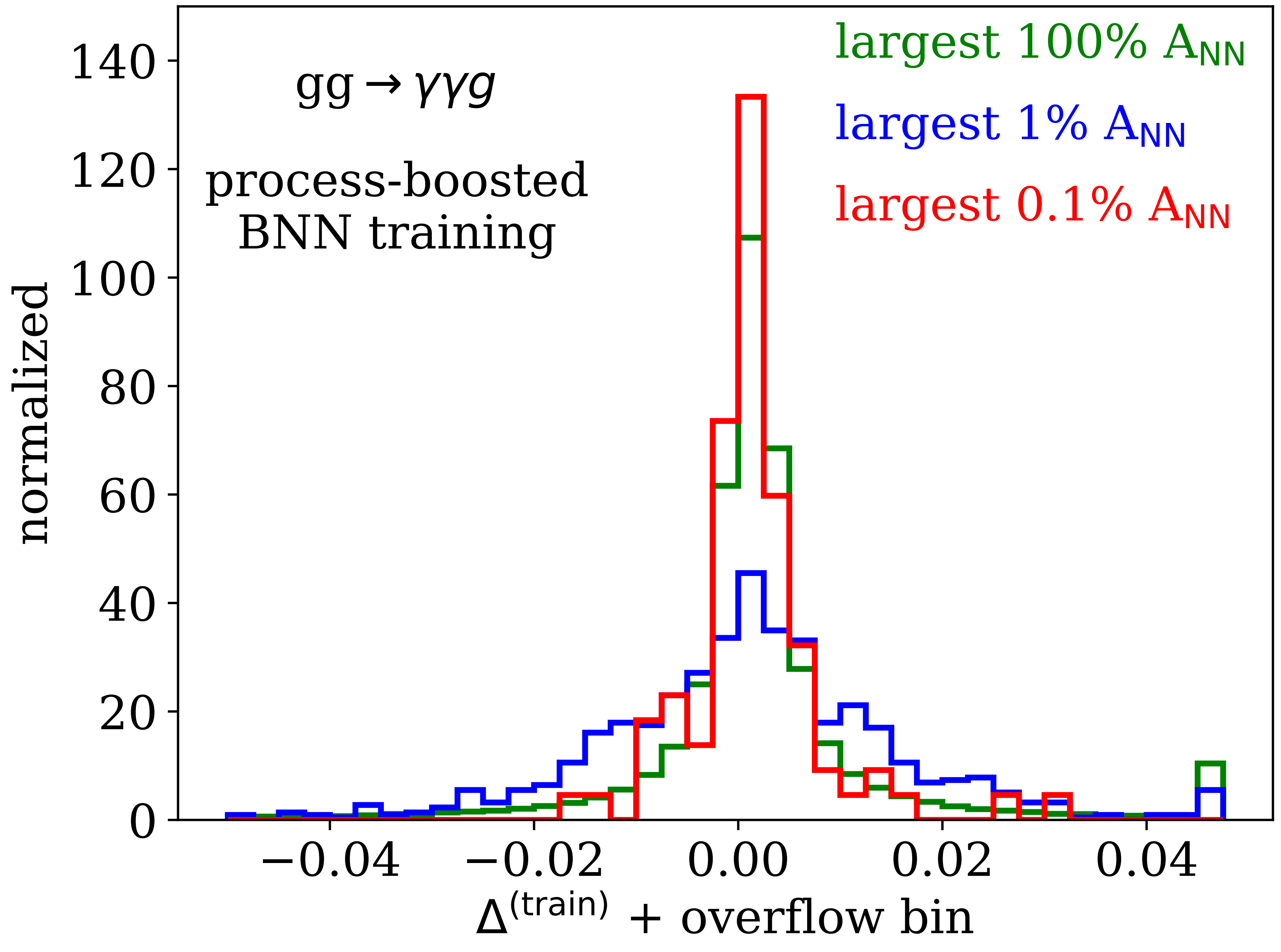
- Regression problem with NN

+ Generalization of interpolation
+ Better scaling than grids for large dimensions

- Minimize distance between prediction and truth

$$\Rightarrow \mathcal{L} = \left(NN(p_i) - \mathcal{M}(p_i)\right)^2 / 2\sigma^2 + \mathcal{L}_{BNN}$$

- **Twist**: Boosting techniques to improve precision
  - Retrain on bad samples
  - 1% precision
  - Boost for reliable uncertainties or small deviations



Monte carlo event generation

1. Generate phase space points

$\rightarrow$ set of four-momenta p_i

2. Calculate event weight

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

Parton density function — 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
     ➡ $J(p_i(r)) = (f \times \mathcal{M})^{-1}$

Monte carlo event generation

1. Generate phase space points

$\rightarrow$ set of four-momenta p_i

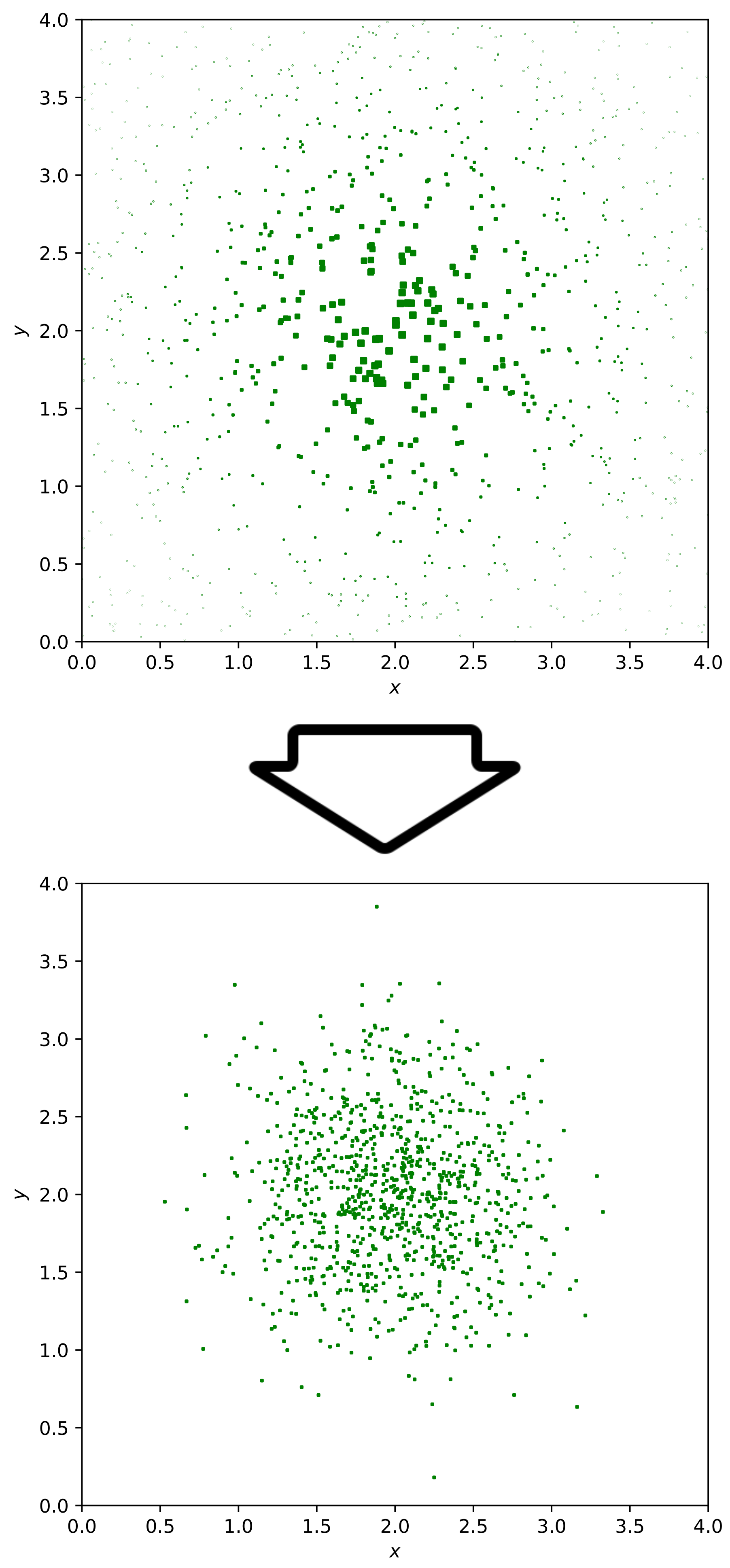
2. Calculate event weight

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

Parton density function — Matrix element † — Phase space mapping

3. Unweighting †

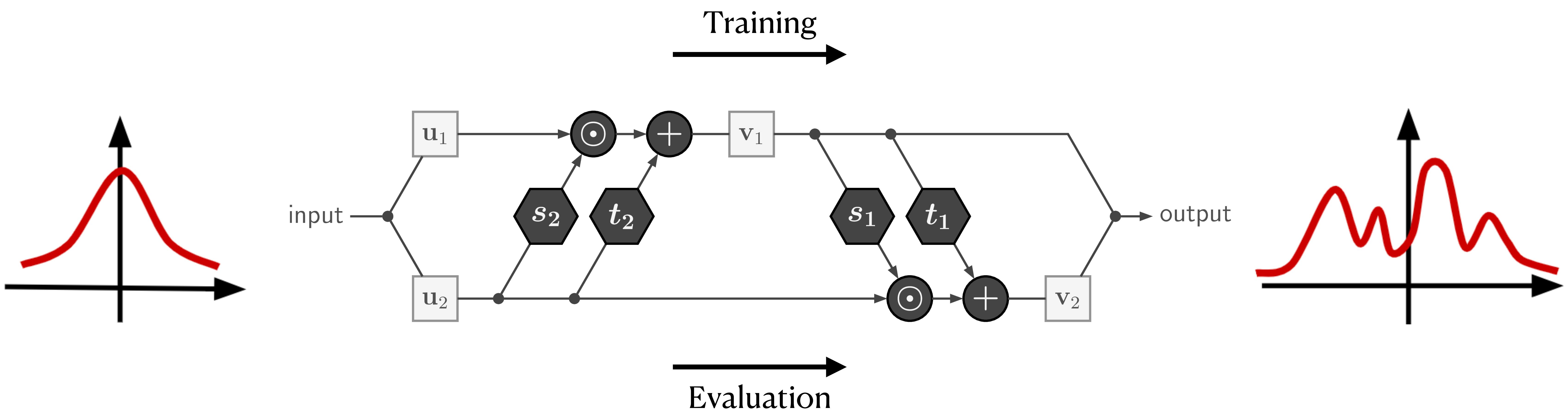
keep events with $\frac{w_i}{w_{\max}} > r \in [0,1]$



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

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

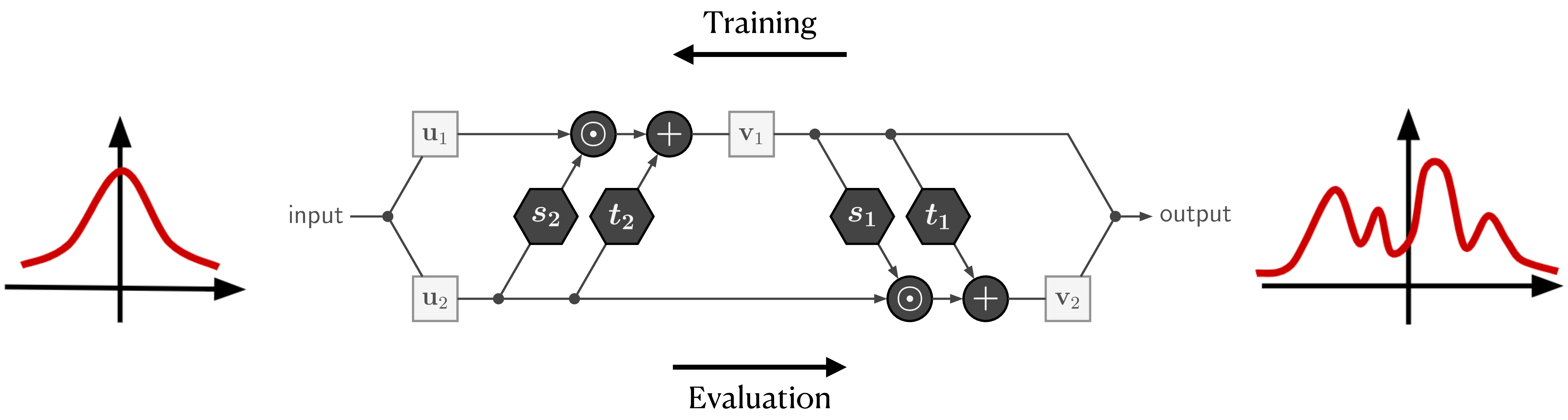
Training on samples x
→ Maximize the log-likelihood

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

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

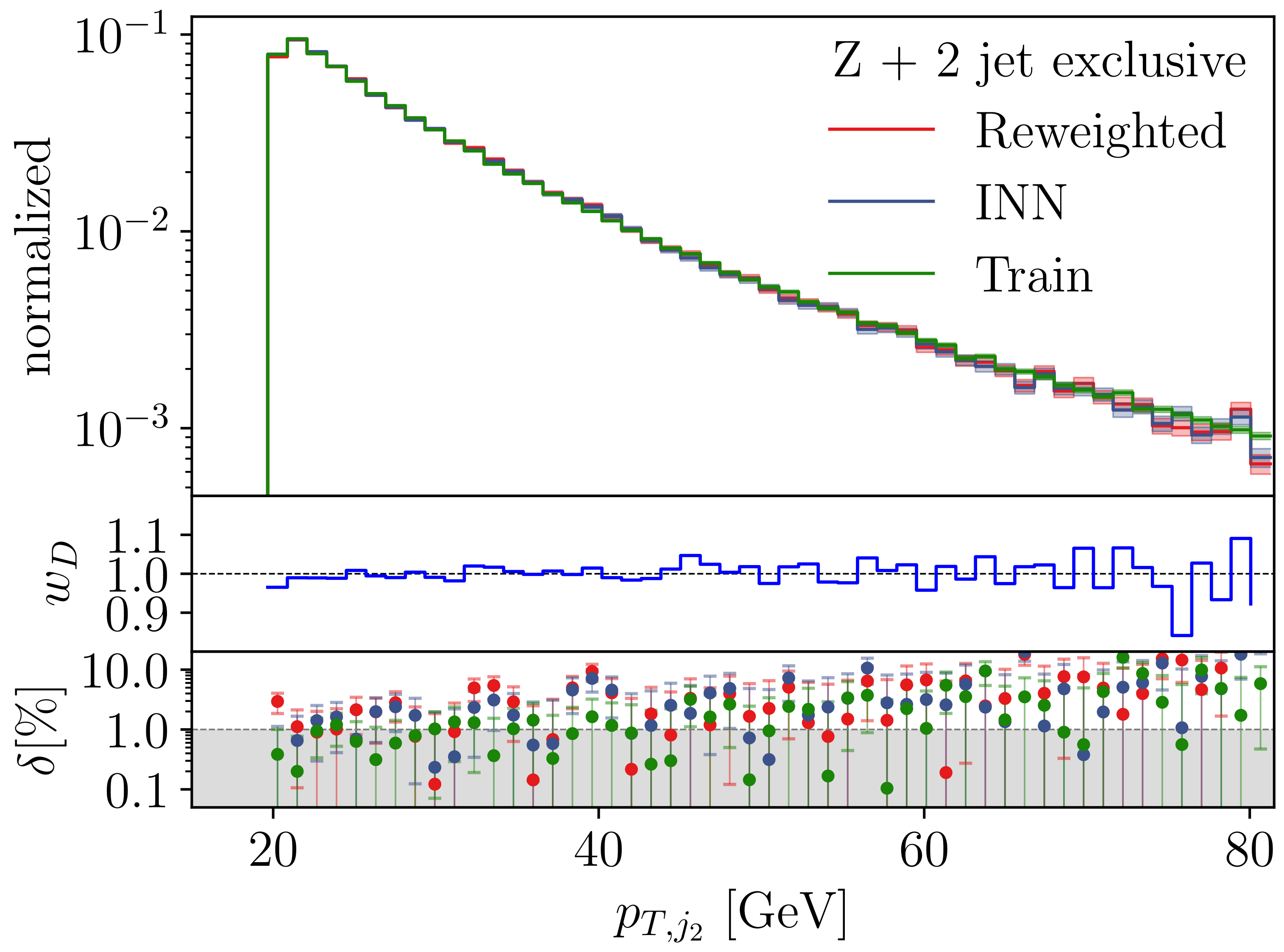
$$\mathscr{L} = \log p_x(x)/t(x)$$
$$= \log p_z(z(x))\ J_{NN}\ /\ t(x)$$

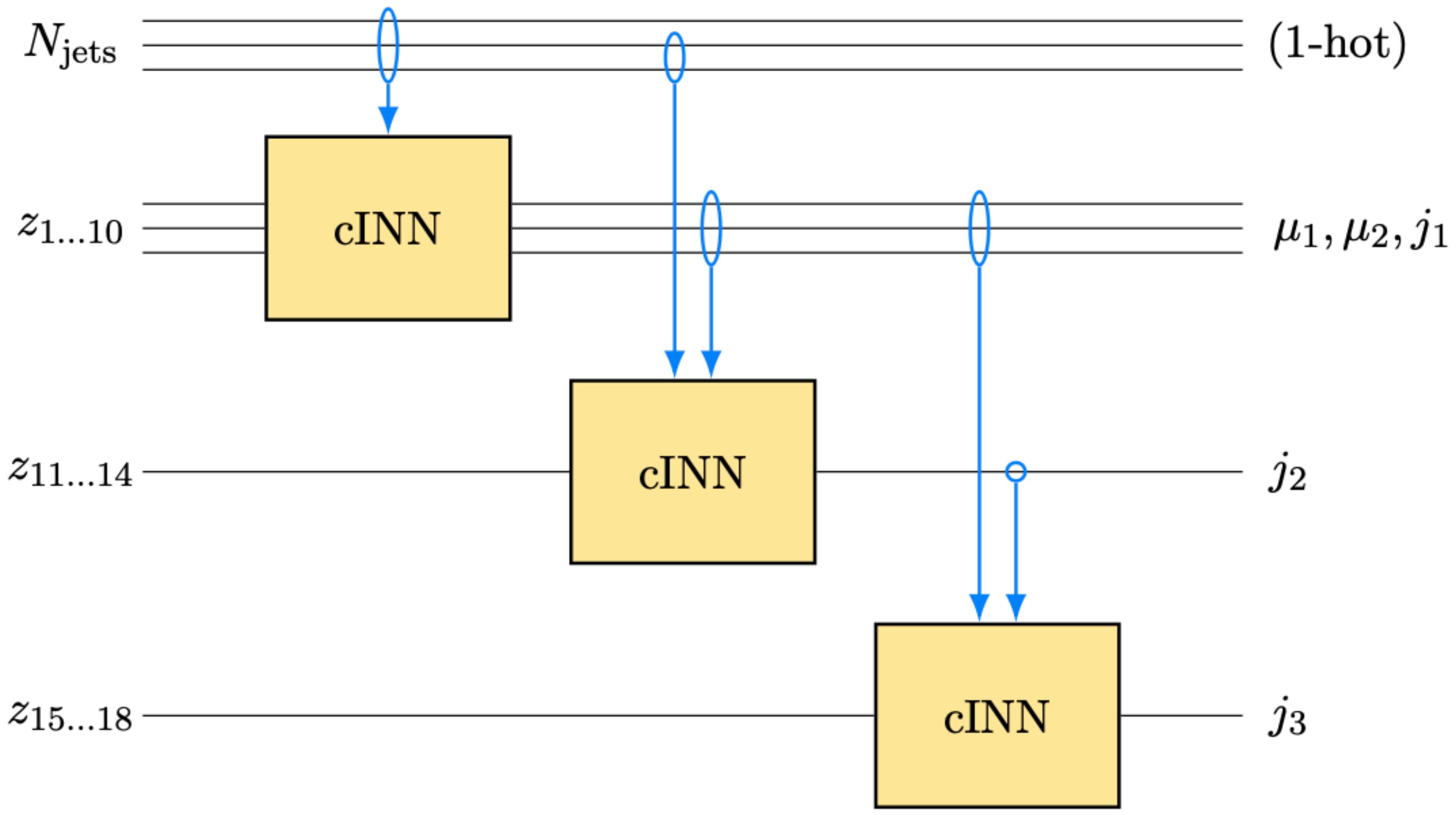
Training on samples x
$\rightarrow$ Maximize the log-likelihood

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

Putting flows to work

Event generation



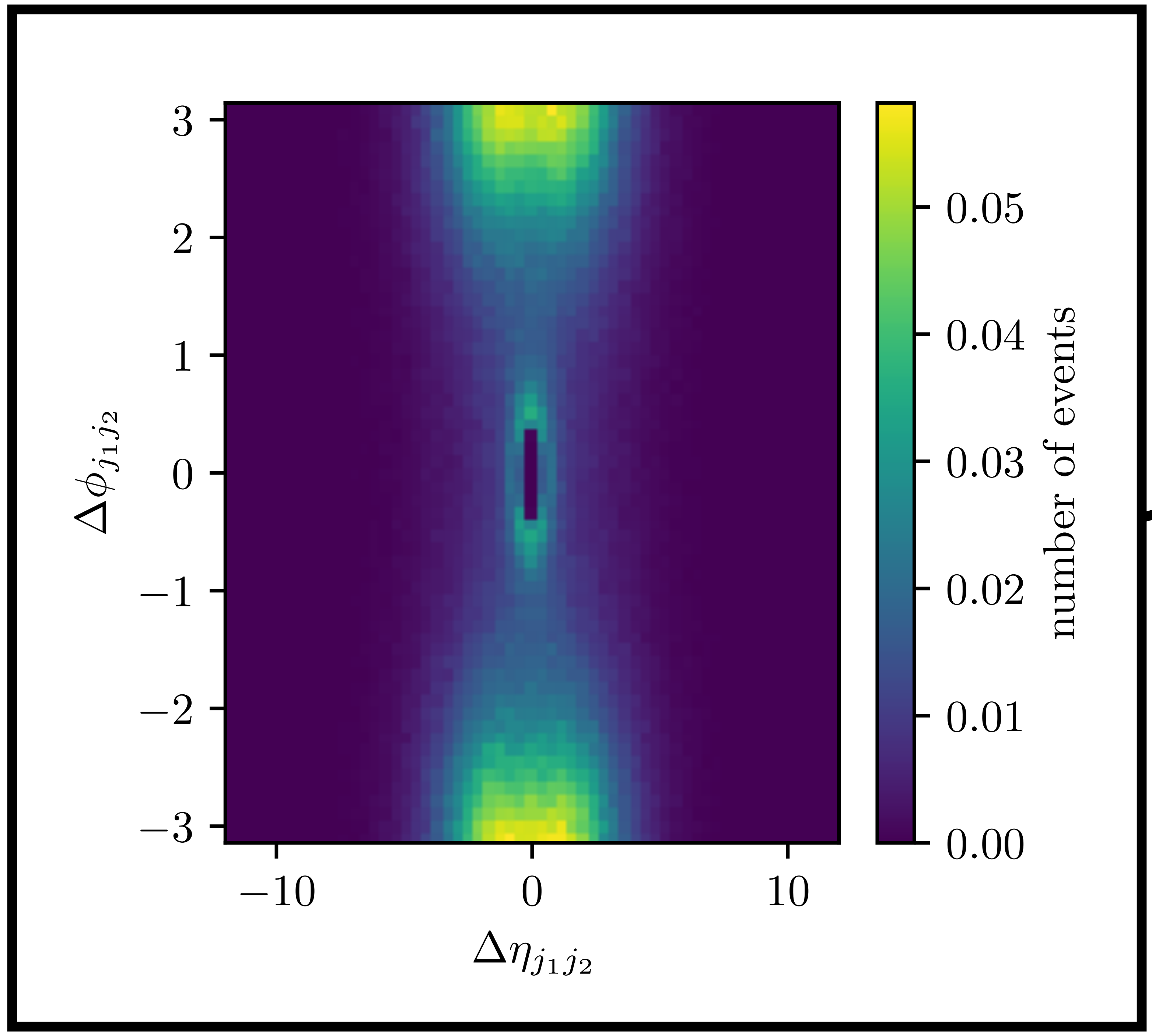


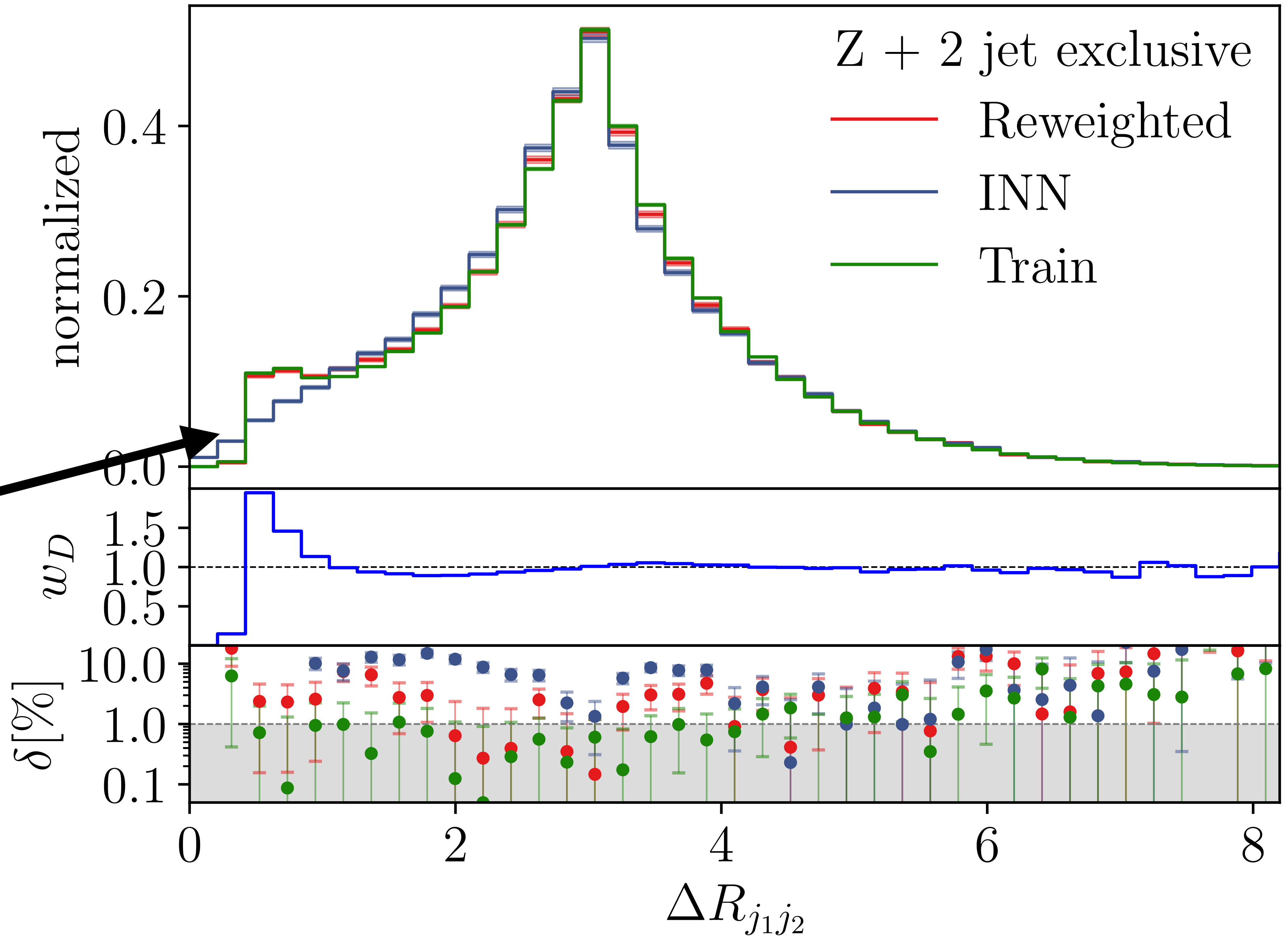
- Train normalizing flow on 4-momenta
- Include symmetries in feature representation
- Excellent performance for direct output

- 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

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

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

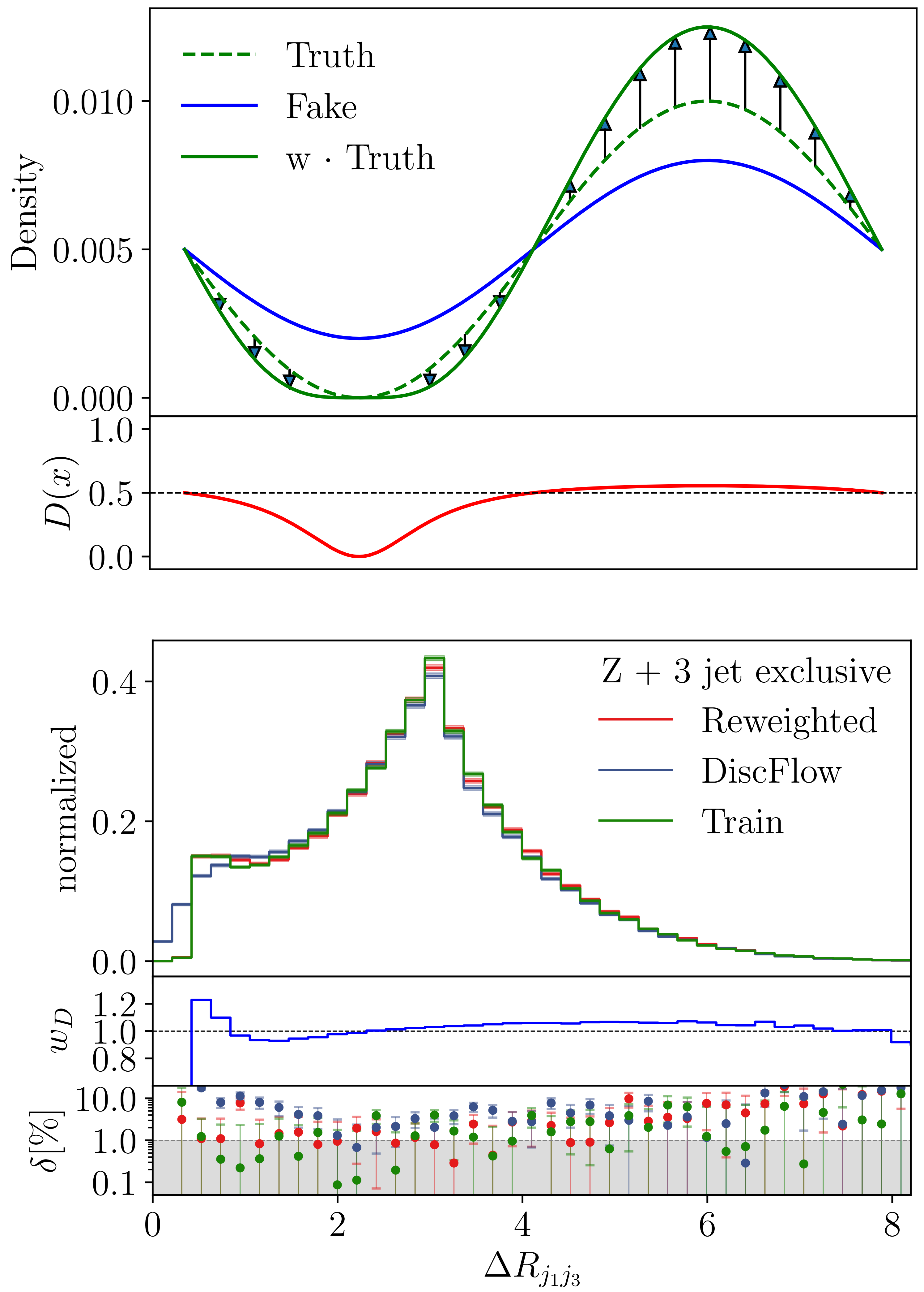
- Upon convergence obtain **reweighting factor**

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

- Use classifier feedback to enhance gradients

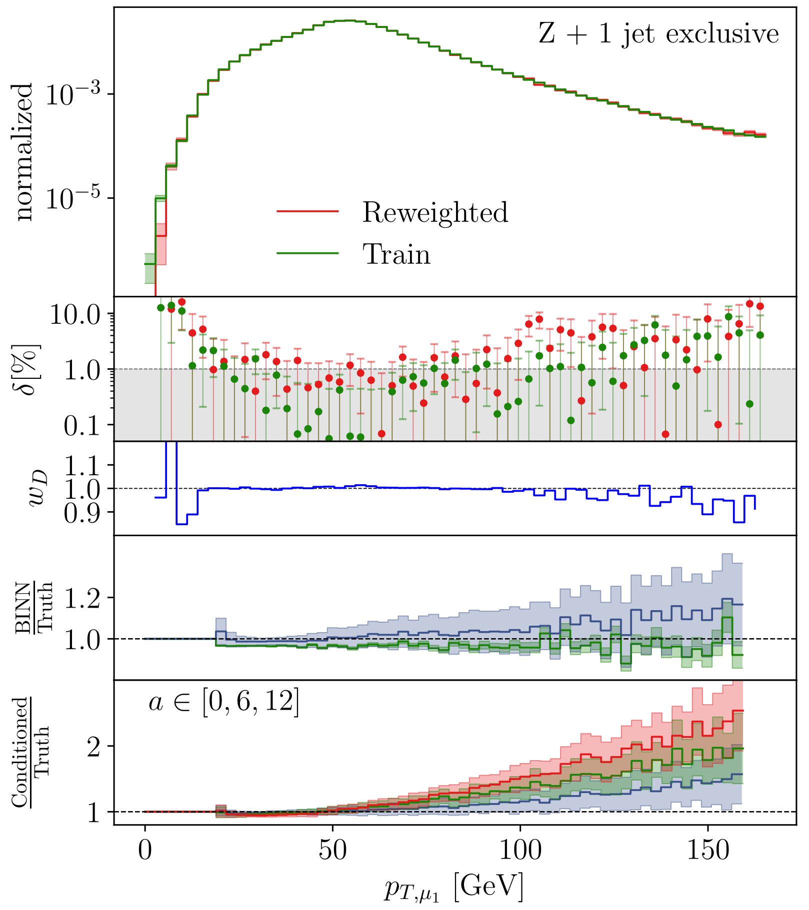
$$\mathscr{L}_{\text{DiscFlow}} \approx \int dx \underbrace{w_D(x)^{\alpha} P(x)}_{\text{reweighted truth}} \left(\frac{\psi(x;c)^2}{2} - \log J(x) \right)$$

$\Rightarrow$ Reduces range of reweighting factors



Forward simulations with generative networks

Event generation



→ Otten et al.

→ Gao et al.

→ Bothmann et al.

→ Stienen et al.

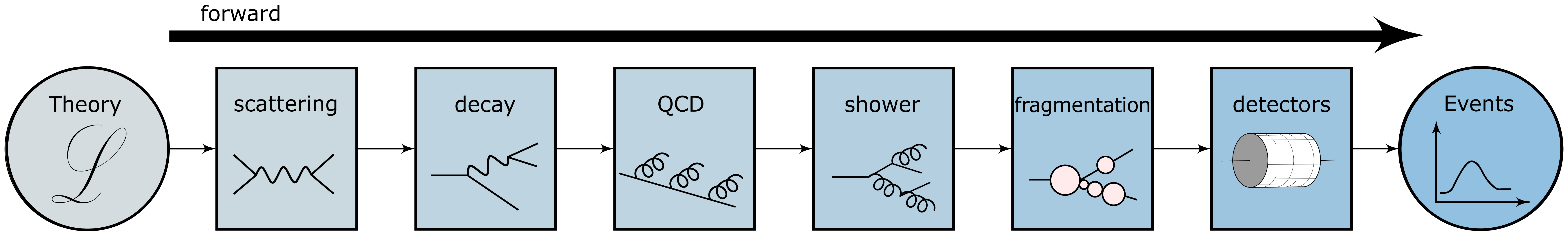
→ AB, et al.

→ and many more

Applications

- Phase space sampling
- End to end learning
- Data compression
- Amplification

What about the uncertainties ???



ML Uncertainties

When do we (not) need them?

- Predictions with poorly trained NNs are sub-***optimal*** but not ***wrong***

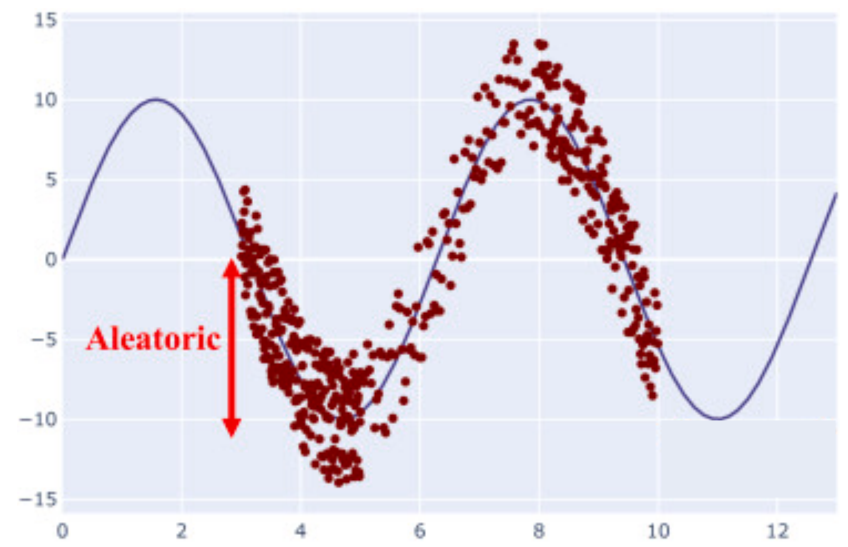
- Example 1: **Phase space sampling**
  - Mapping induces Jacobian
  - Events obtain weight from ME × J
  - Bad mapping → small unweighting efficiency

- Example 2: INN for **integration**
  - Sub-optimal contour deformation
  - **High variance** of integral
  - Not efficient but not wrong

➡ **Control** comes from **simulation** !

- Example 3: Data compression
  - Direct use of generator output

- Example 4:
  - Control over network training



modified from M. Abdar [doi.org/10.1016/j.inffus.2021.05.008]

➡ How can we estimate this uncertainty?

Defining the loss function

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

Approximate $q(w)$ by minimizing KL divergence

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

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

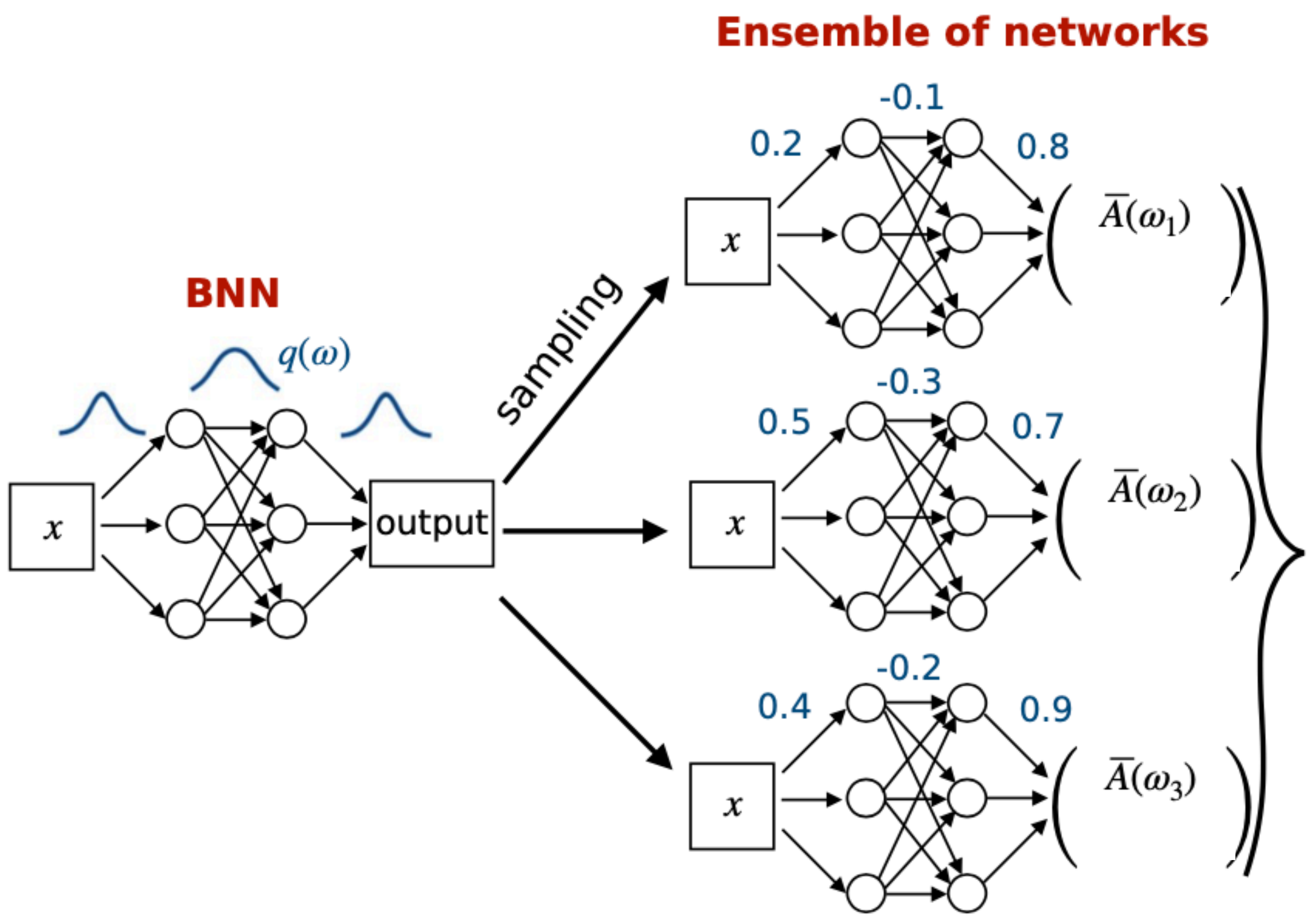
$$= \int dw\ q(w)\ \log \frac{q(w)p(T)}{p(w)p(T\,|\,w)}$$

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

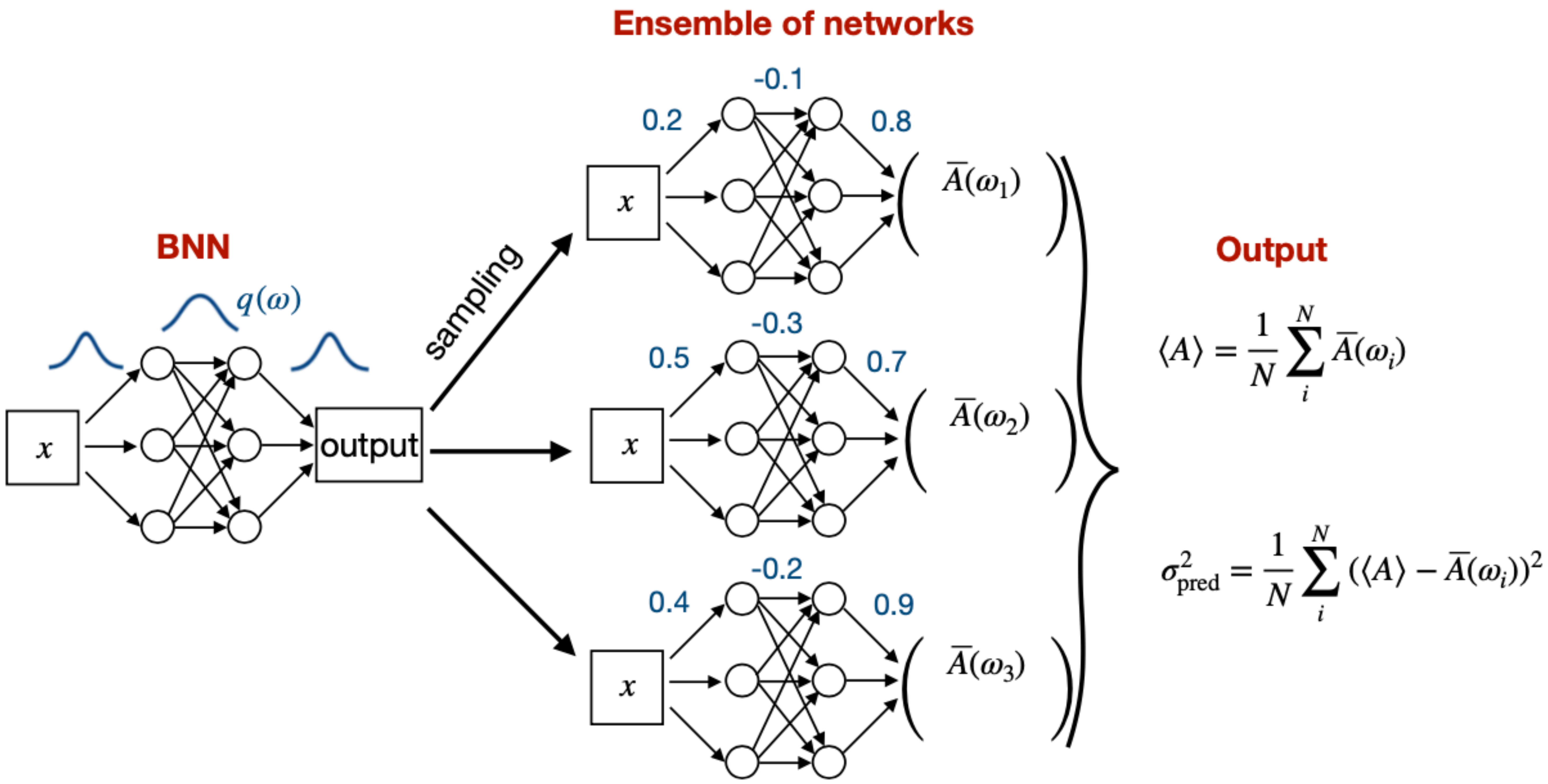
Gaussian prior

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

Standard loss



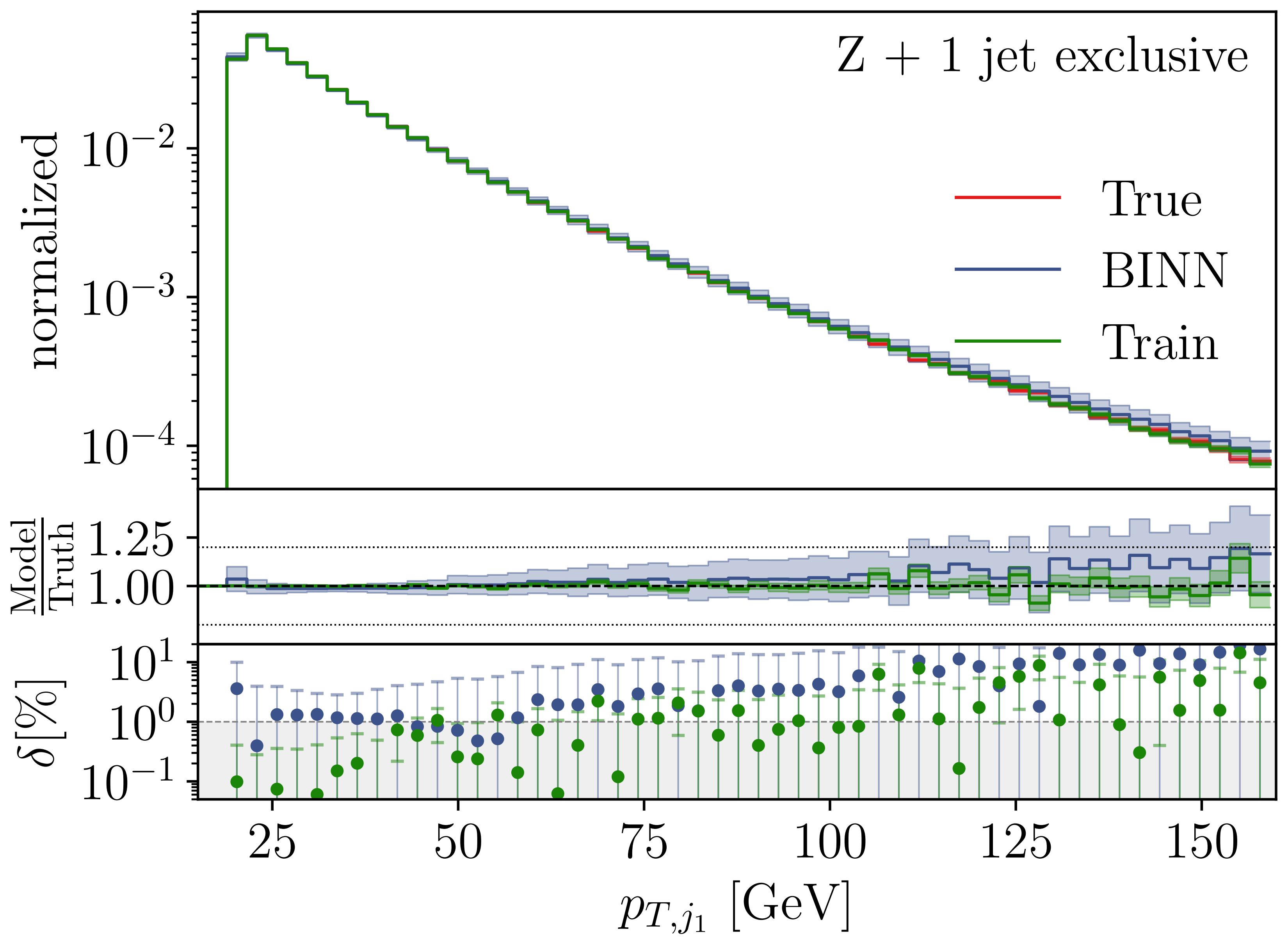
Bayesian Neural Network

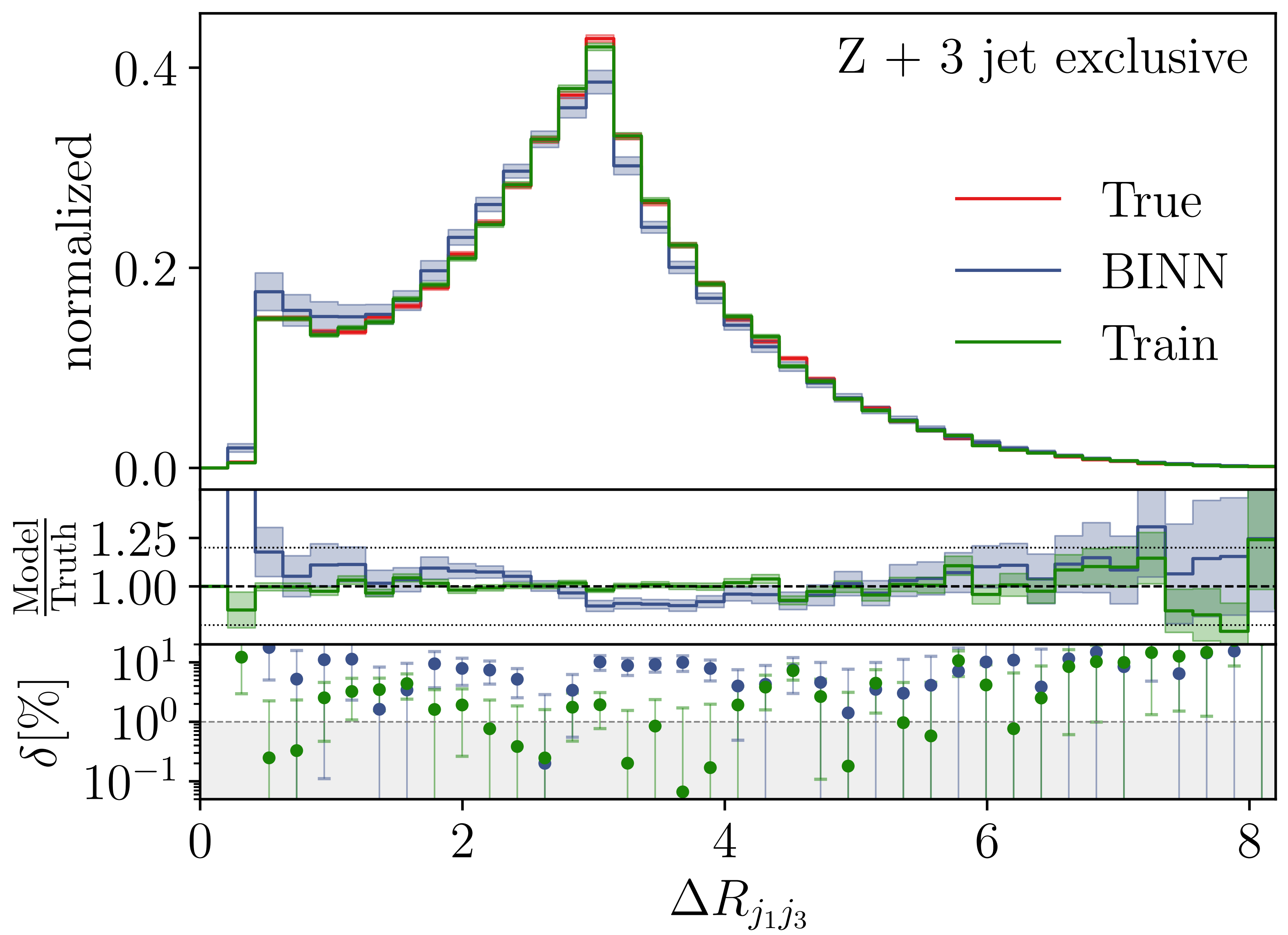


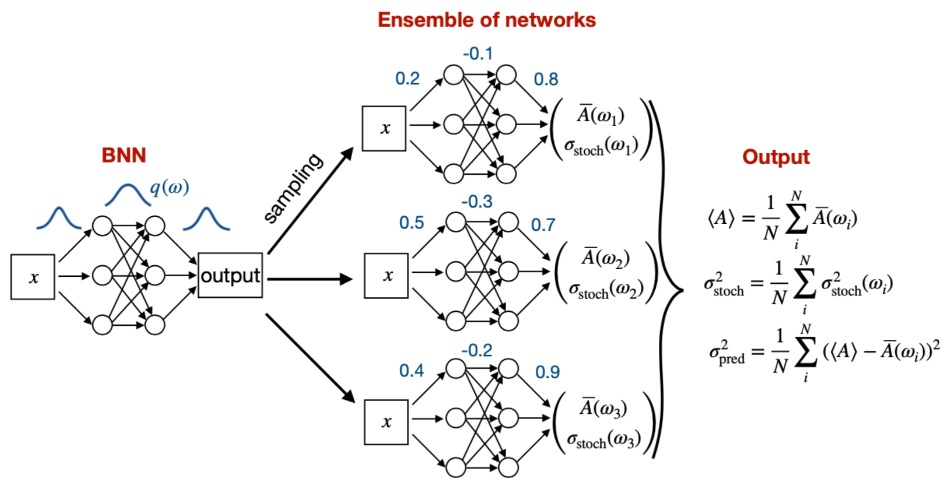
$$\mathscr{L} = \mathscr{L}_{INN} + KL_{prior}$$

$$= \sum_{n=1}^{N} \langle \log p_X(x_n \,|\, \theta) \rangle_{\theta \sim q_\Phi(\theta)} - KL(q_\Phi(\theta), p(\theta))$$

Bayesian generative networks





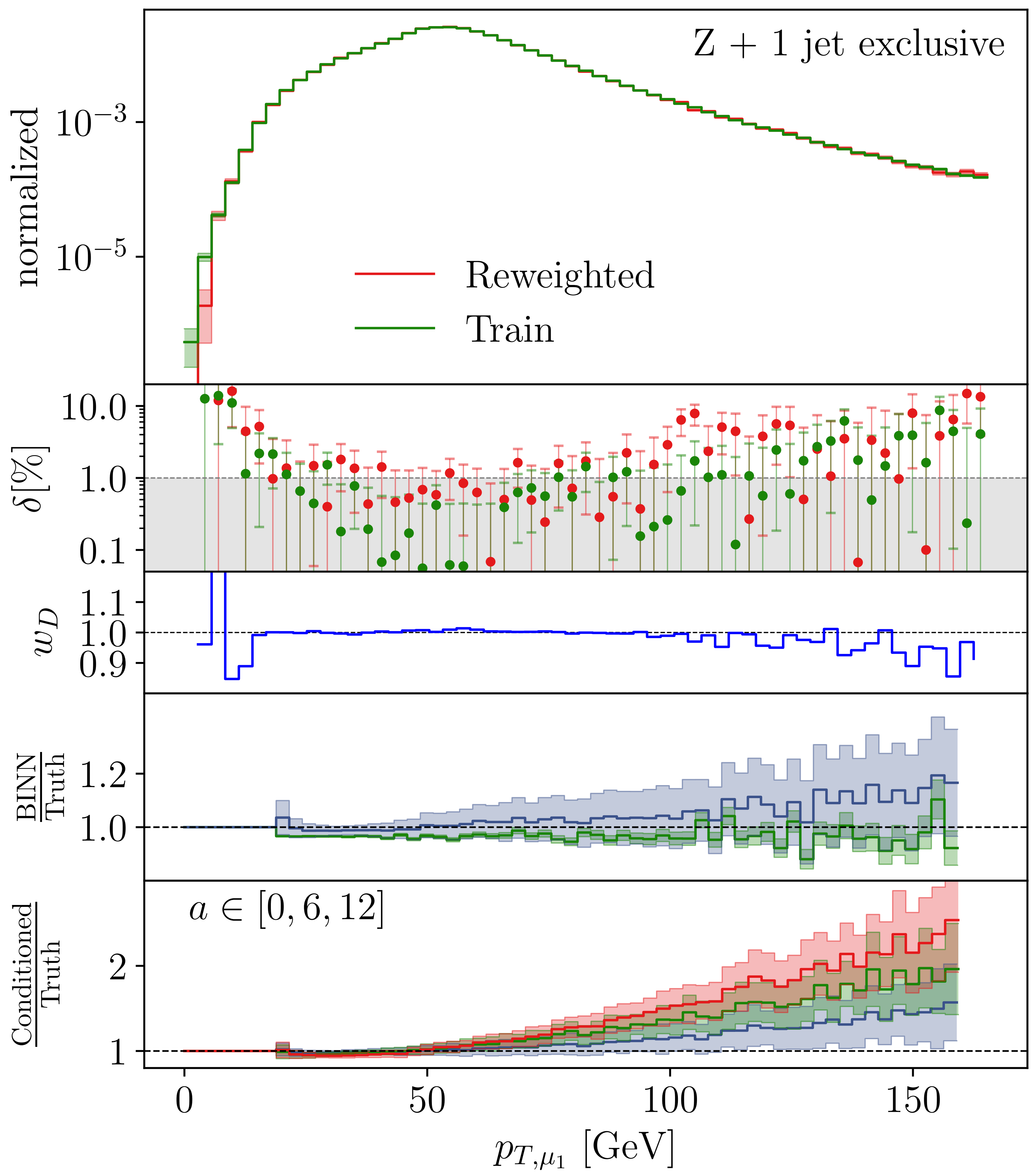


⇒ BINN captures uncertainty related to convergence and statistical uncertainties

⇒ BINN does not capture lack of expressiveness

Putting flows to work

Event generation



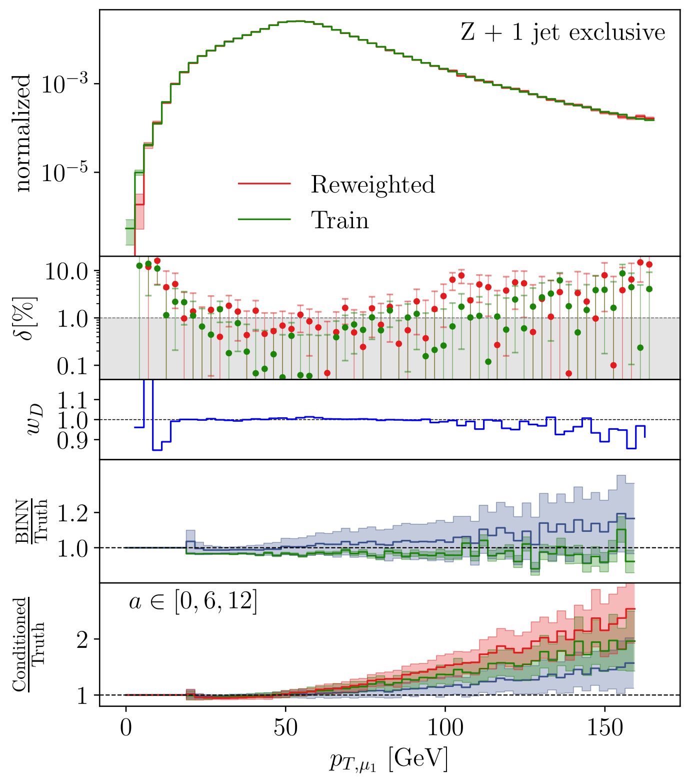
- Basis: INN
  - Phase space symmetries in architecture
- Control via classifier D
  - $$\frac{p_{\text{truth}}(x)}{p_{\text{INN}}(x)} = \frac{D(x)}{1 - D(x)}$$
- Precision via reweighting
  - Correct deviations of p_{INN}

➡ Uncertainty estimation via Bayesian NN

➡ Uncertainty propagation via conditioning

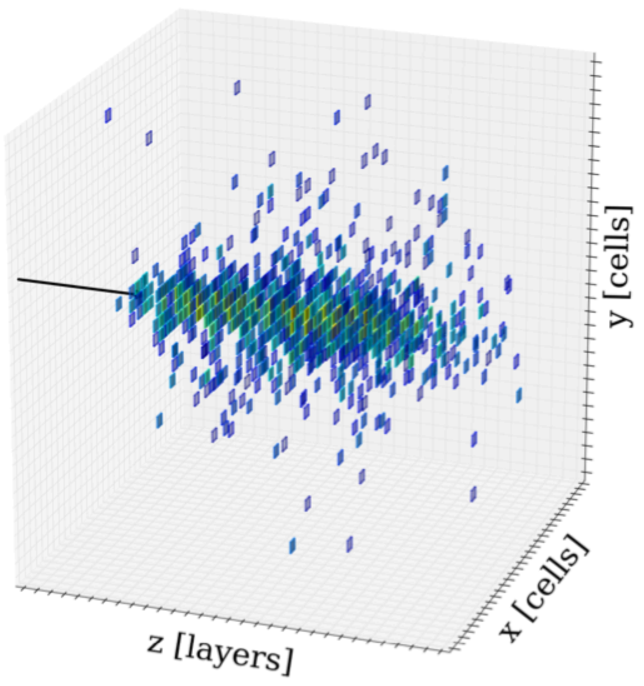
Forward simulations with generative networks

Event generation

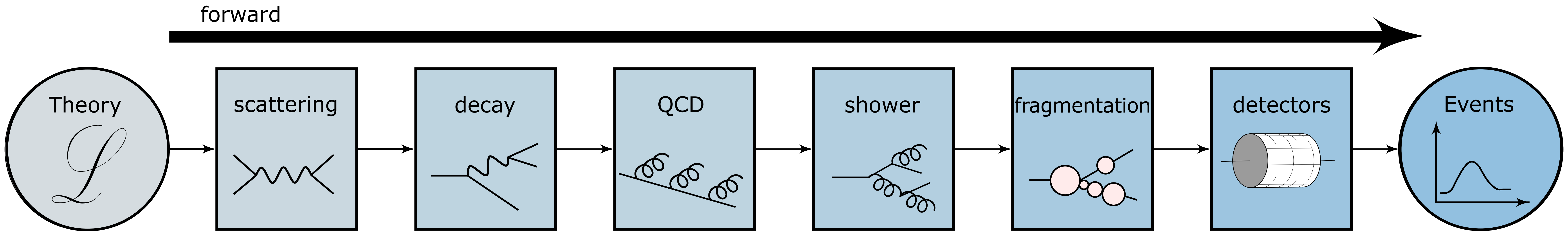


→ Otten et al.
→ Gao et al.
→ Bothmann et al.
→ Stienen et al.
→ AB, et al.
→ and many more

Detector simulation



→ CaloGAN by M. Paganini et al.
→ BIBAE by E. Buhman, S. Diefenbacher et a
→ CaloFlow by C. Krause , D. Shih
→ and many more



Applications

- Phase space sampling
- End to end learning
- Data compression
- Amplification

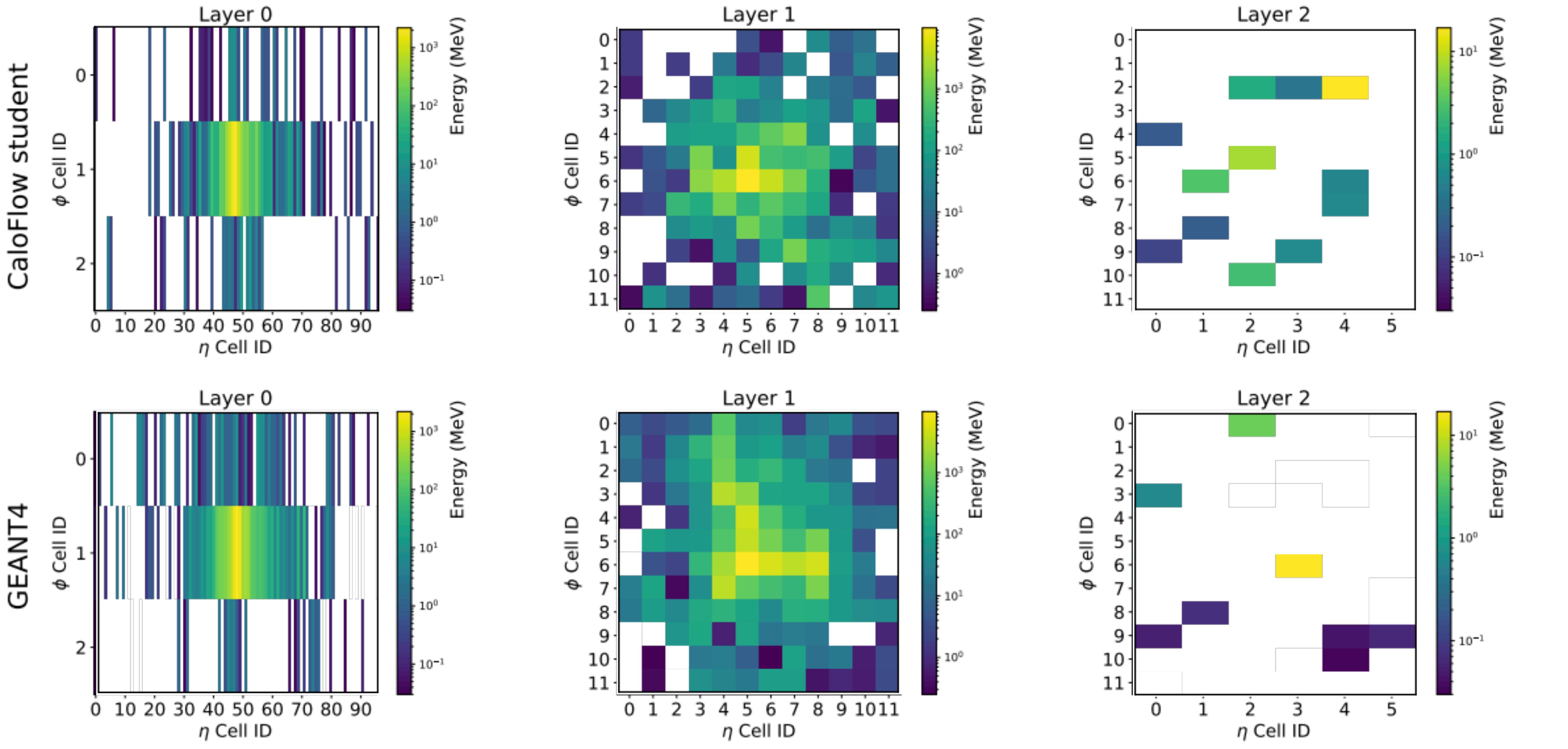
Phase space sampling with generative networks (GAN, VAE, NF)

Particularly promising architecture
→ Normalizing flows

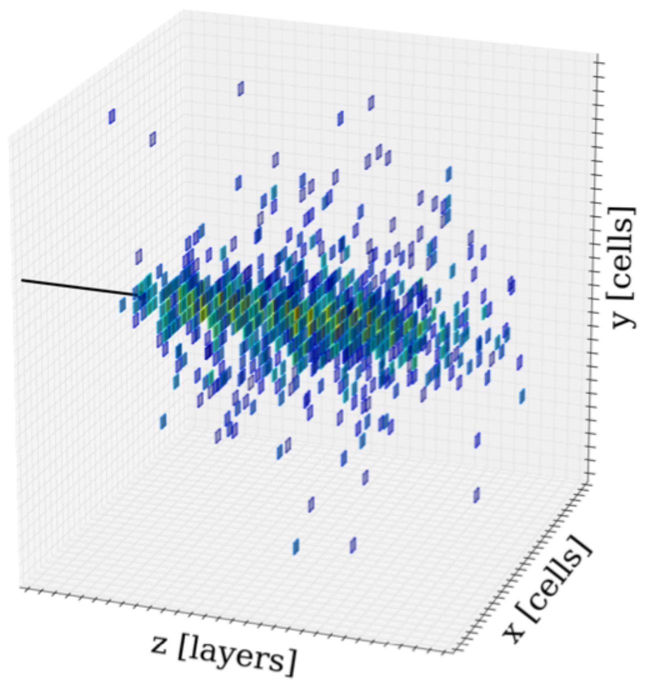
Putting flows to work

Detector simulation

Challenge: large dimensionality ($3 \times 96, 12 \times 12, 12 \times 6$)



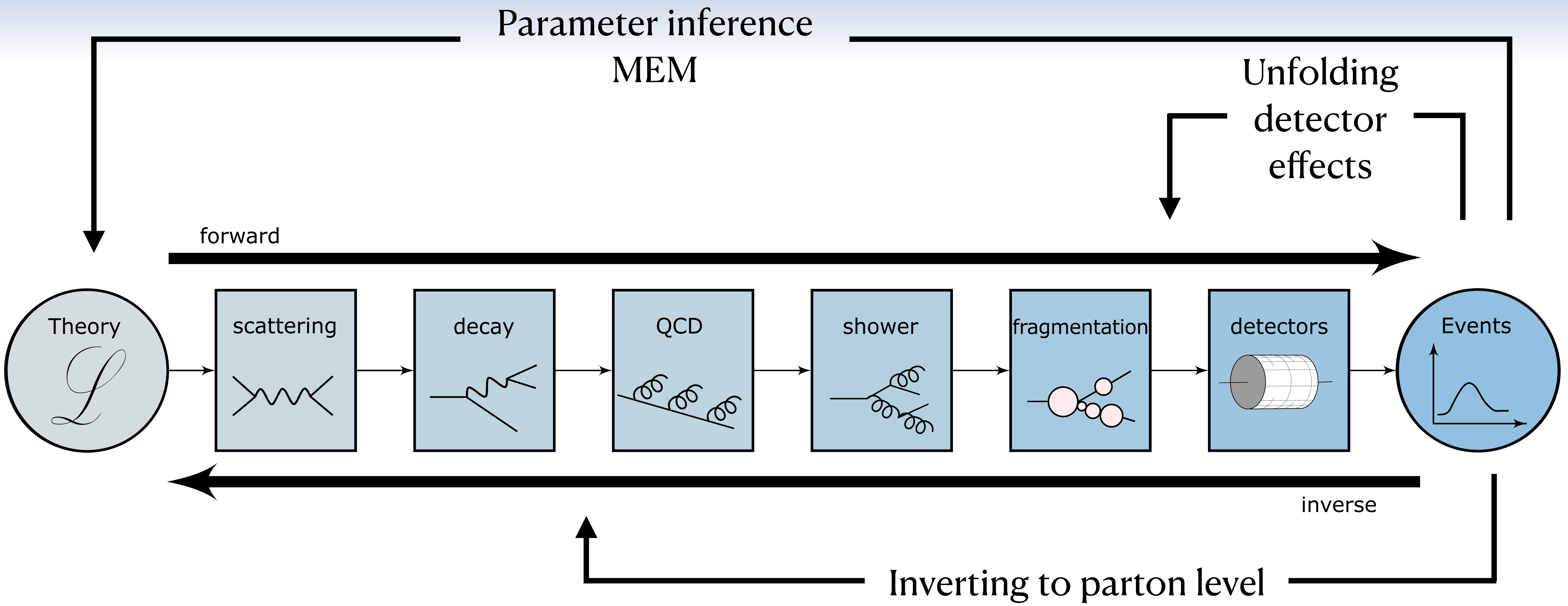
30000 pixels



C. Krause & D. Shih [2110.11377]

π^+ shower individual & average

Inverting the simulation chain

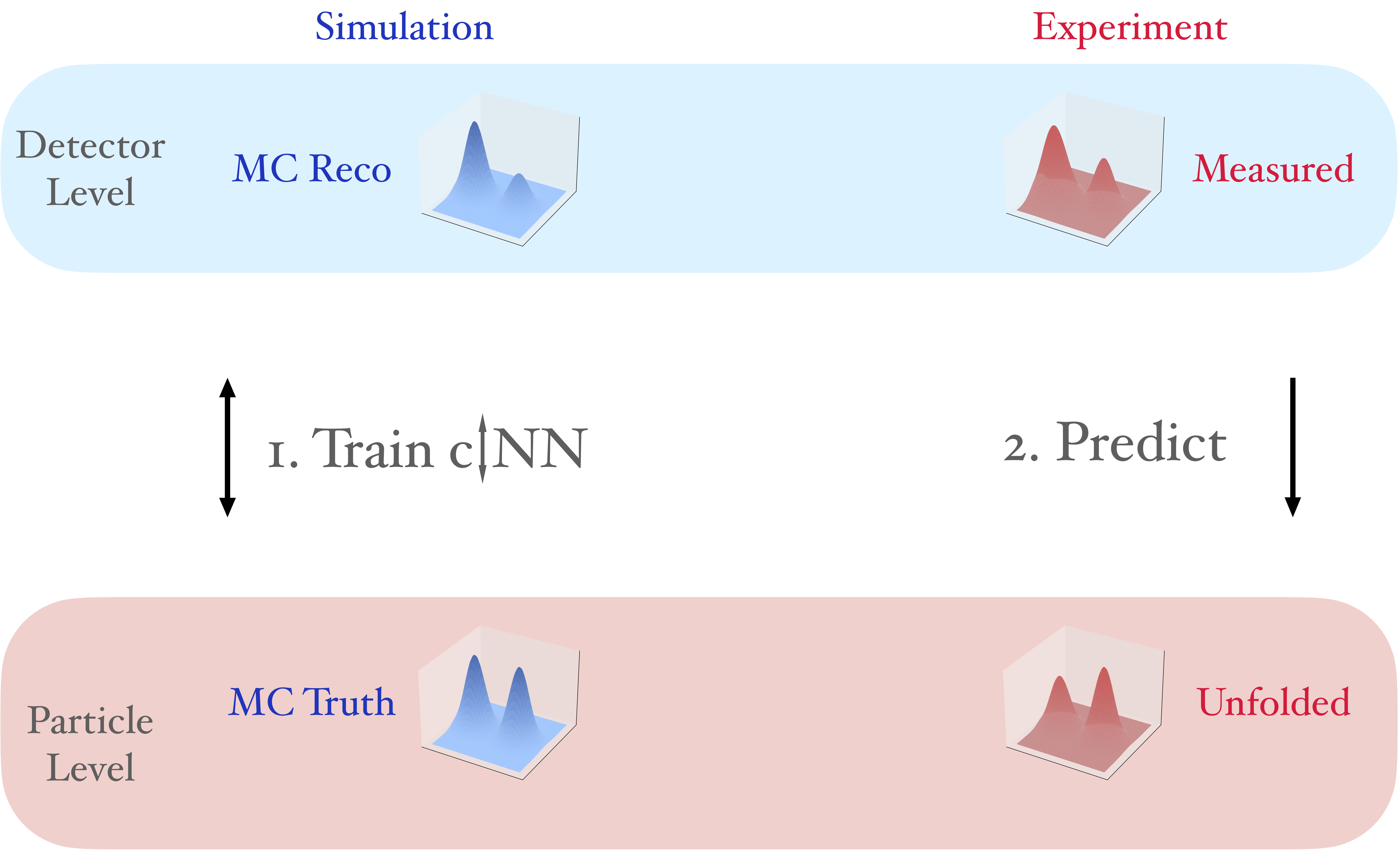


Requirements

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

ML unfolding methods

High-dimensional. Bin independent. Robust.

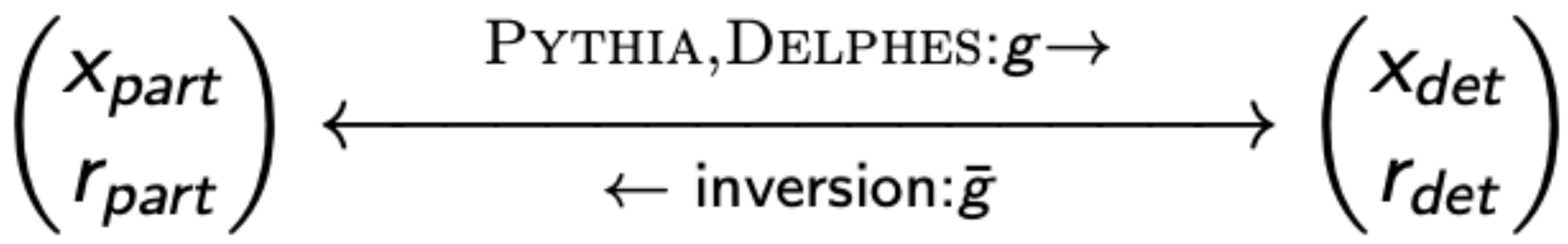


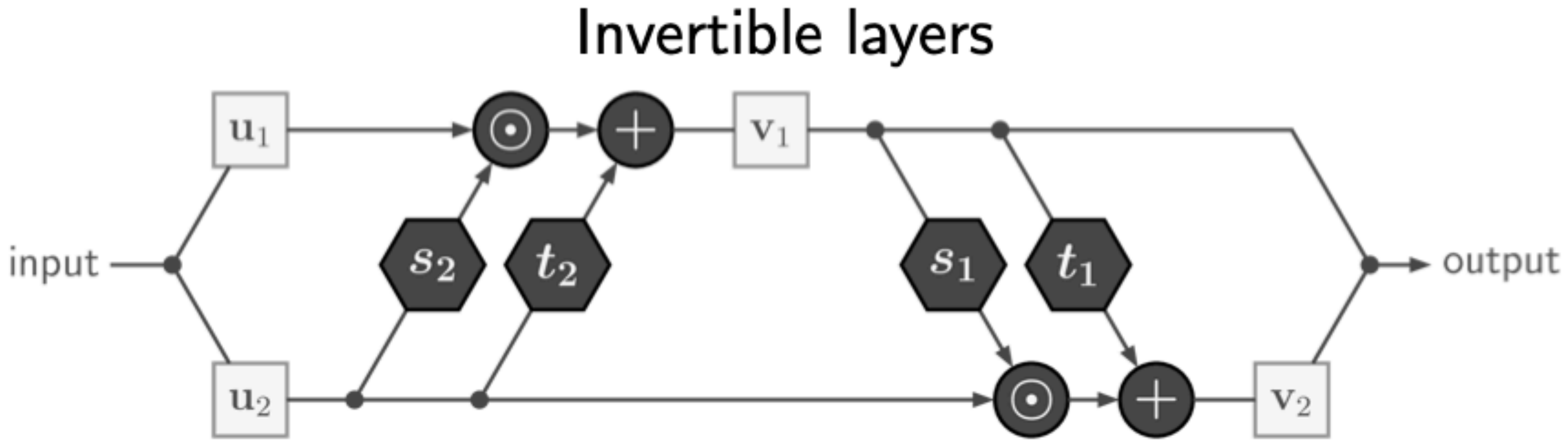
Density based approach

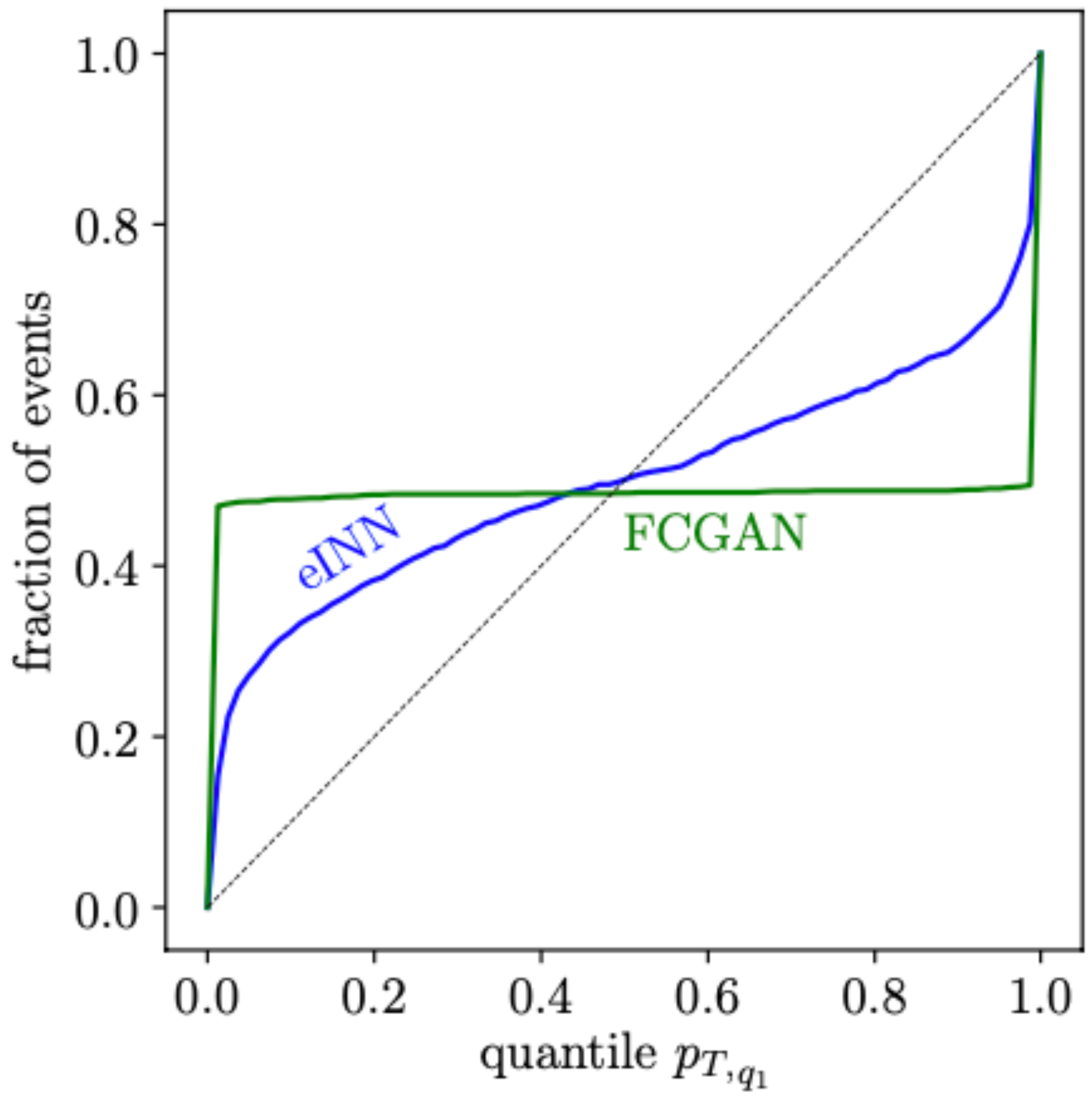
Output: probability density per unfolded event

$$p(x_{part} | x_{reco})$$

Inverting with invertible networks

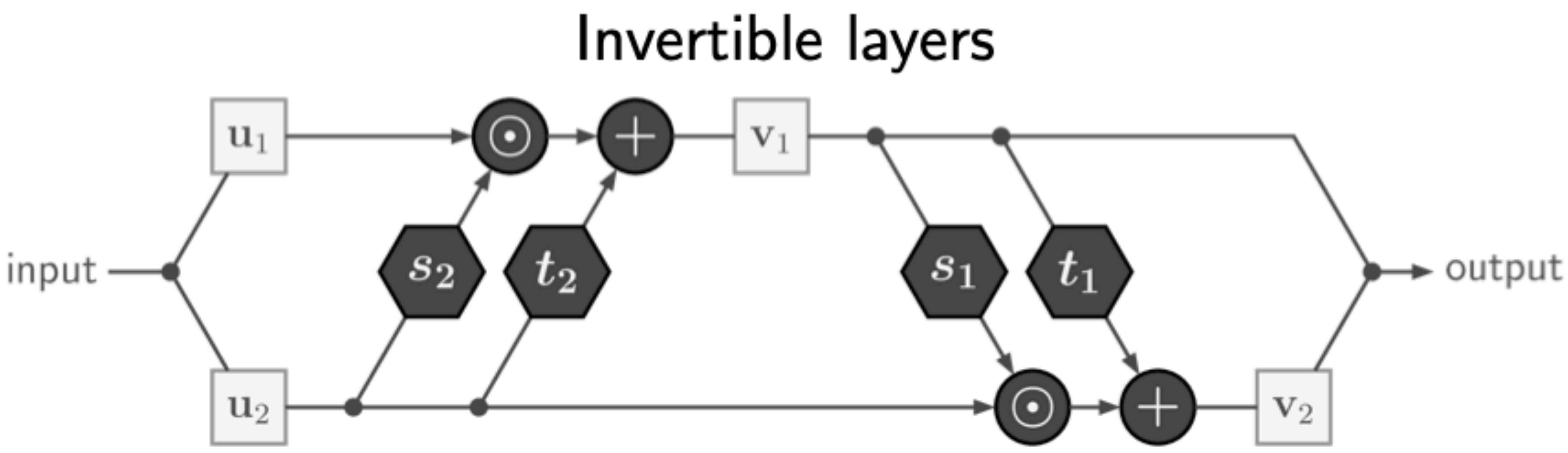






Inverting with invertible networks

$$\begin{pmatrix} x_{part} \\ r_{part} \end{pmatrix} \xleftrightarrow[\leftarrow \text{ inversion:}\bar{g}]{\text{PYTHIA,DELPHES:}g\rightarrow} \begin{pmatrix} x_{det} \\ r_{det} \end{pmatrix}$$



$$\mathcal{L} = \mathcal{L}_{part} + \mathcal{L}_{det} + \mathcal{L}_r$$

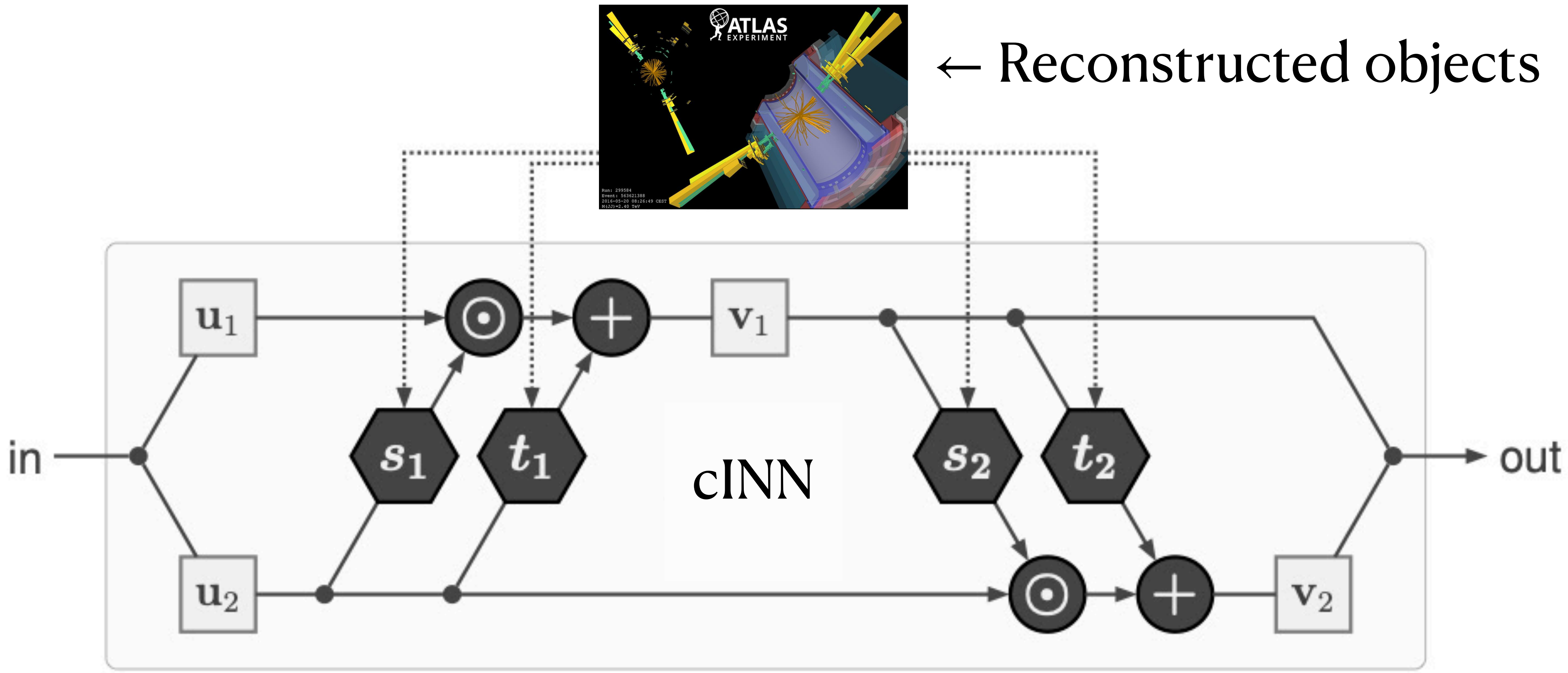
$$\mathcal{L}_{part} = \lambda_1 \, ||x_{part} - \bar{g}_{part}(x_{det}, r_{det})|| + \lambda_2 \, \text{MMD}(x_{part}, \bar{g}_{part}(x_{det}, r_{det})$$

$$\mathcal{L}_{det} = \lambda_3 \, ||x_{det} - g_{det}(x_{part}, r_{part})|| + \lambda_4 \, \text{MMD}(x_{det}, g_{det}(x_{part}, r_{part})$$

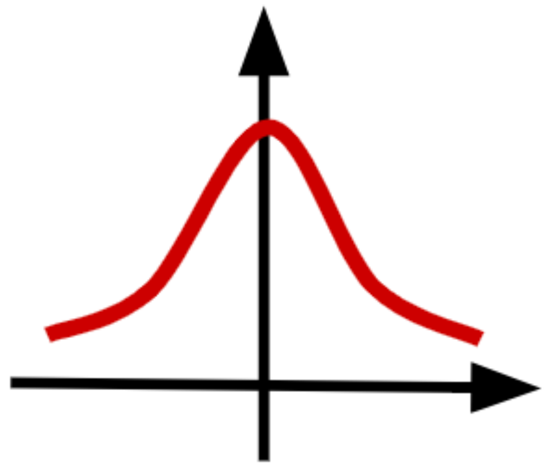
$$\mathcal{L}_r = \lambda_5 \, \text{MMD}(\bar{g}_r(x_{det}, r_{det}), \mathcal{N}) + \lambda_6 \, \text{MMD}(g_r(x_{part}, r_{part}), \mathcal{N})$$

cINN unfolding

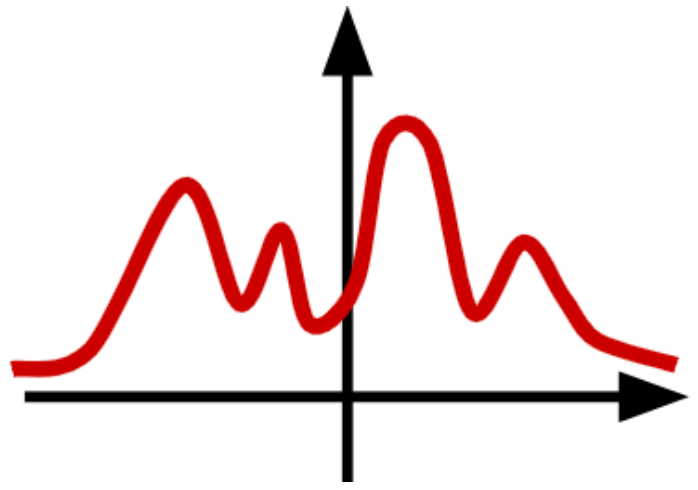
Given a reconstructed event:
What is the probability distribution at particle level?



← Reconstructed objects



Normal distribution
$z \sim \mathcal{N}$



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

Training

Unfolding

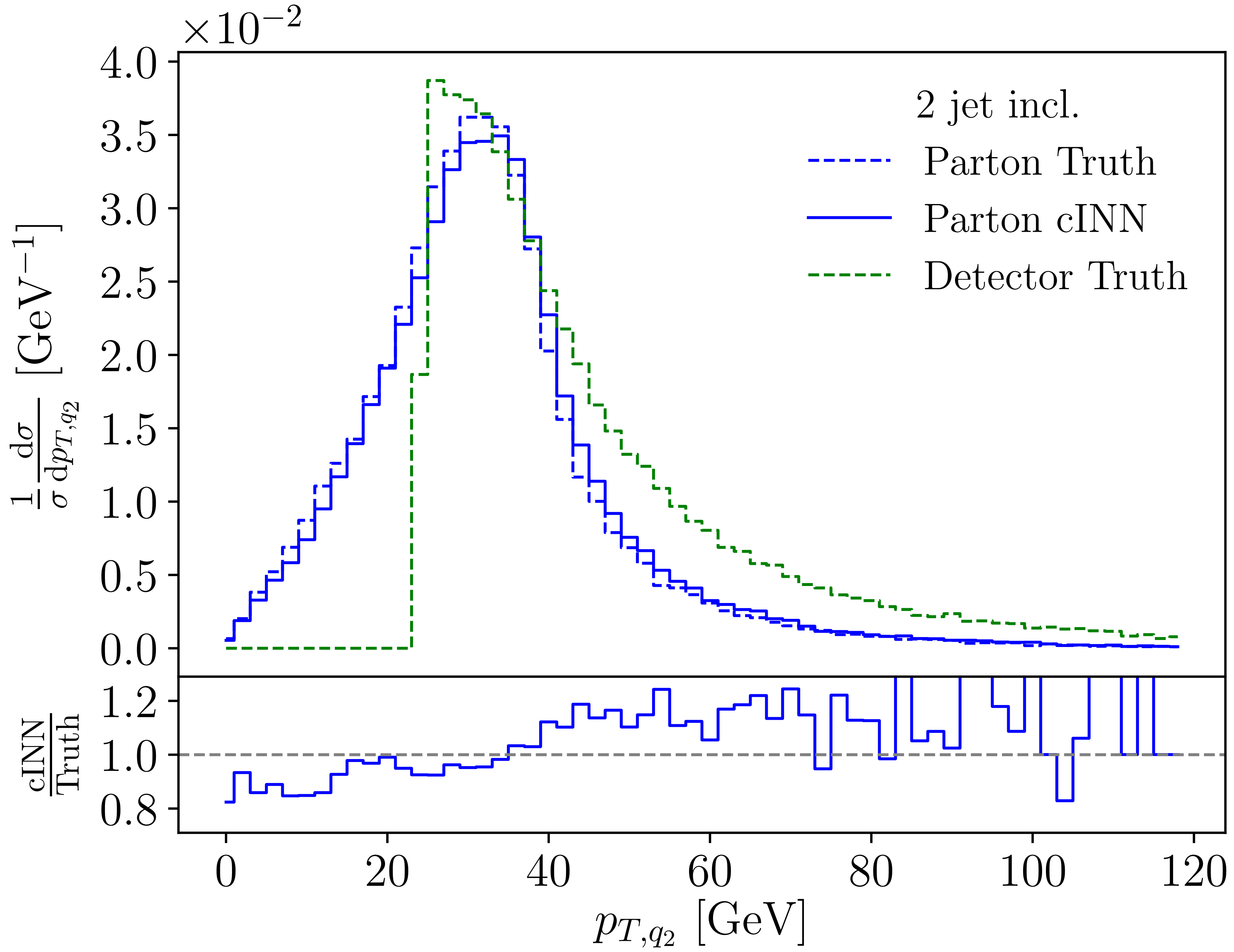
→ Maximize the log-likelihood

$$\mathscr{L} = \log p(\theta \,|\, x_{part}, x_{reco})$$
$$= \log p(z \,|\, \theta, x_{reco}) + \log J_{NN} + p(\theta)$$

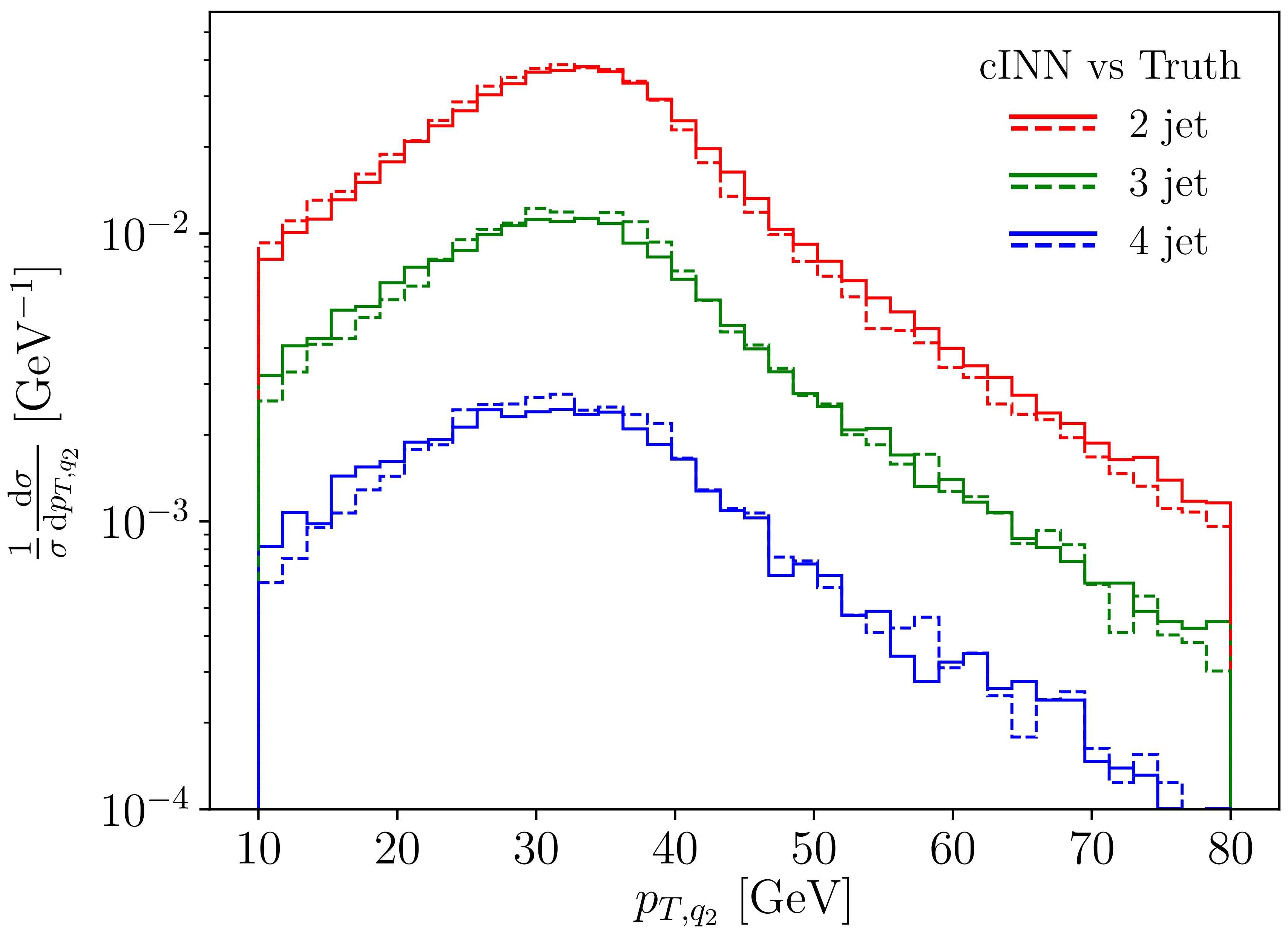
Inverting inclusive distributions

$pp > WZ > q\bar{q}l^+l^- +$ ISR → 2/3/4 jet events

Training on inclusive dataset



Evaluate exclusive 2/3/4 jet events



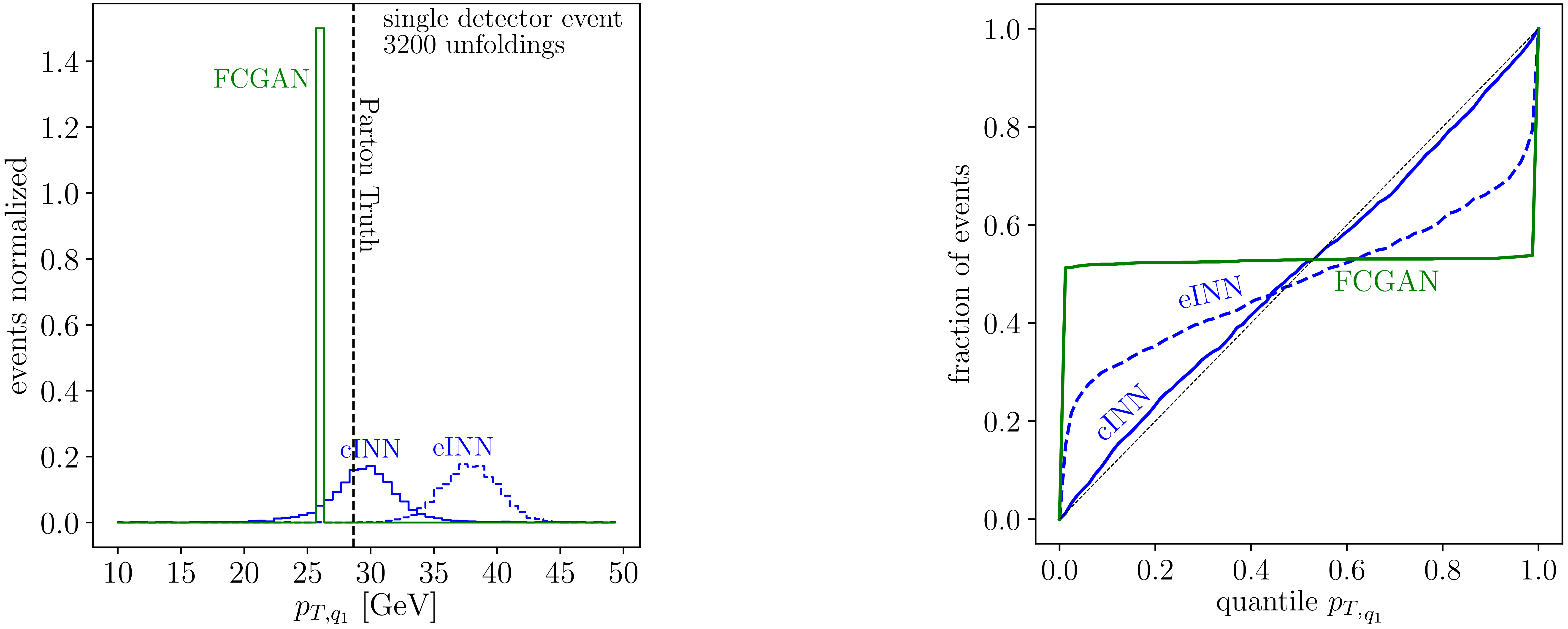
M. Bellagente et al. [2006.06685]

- ☑ High-dimensional
- ☑ Bin-independent
- ☐ Statistically well defined ?

Event-wise unfolding

No deterministic mapping!

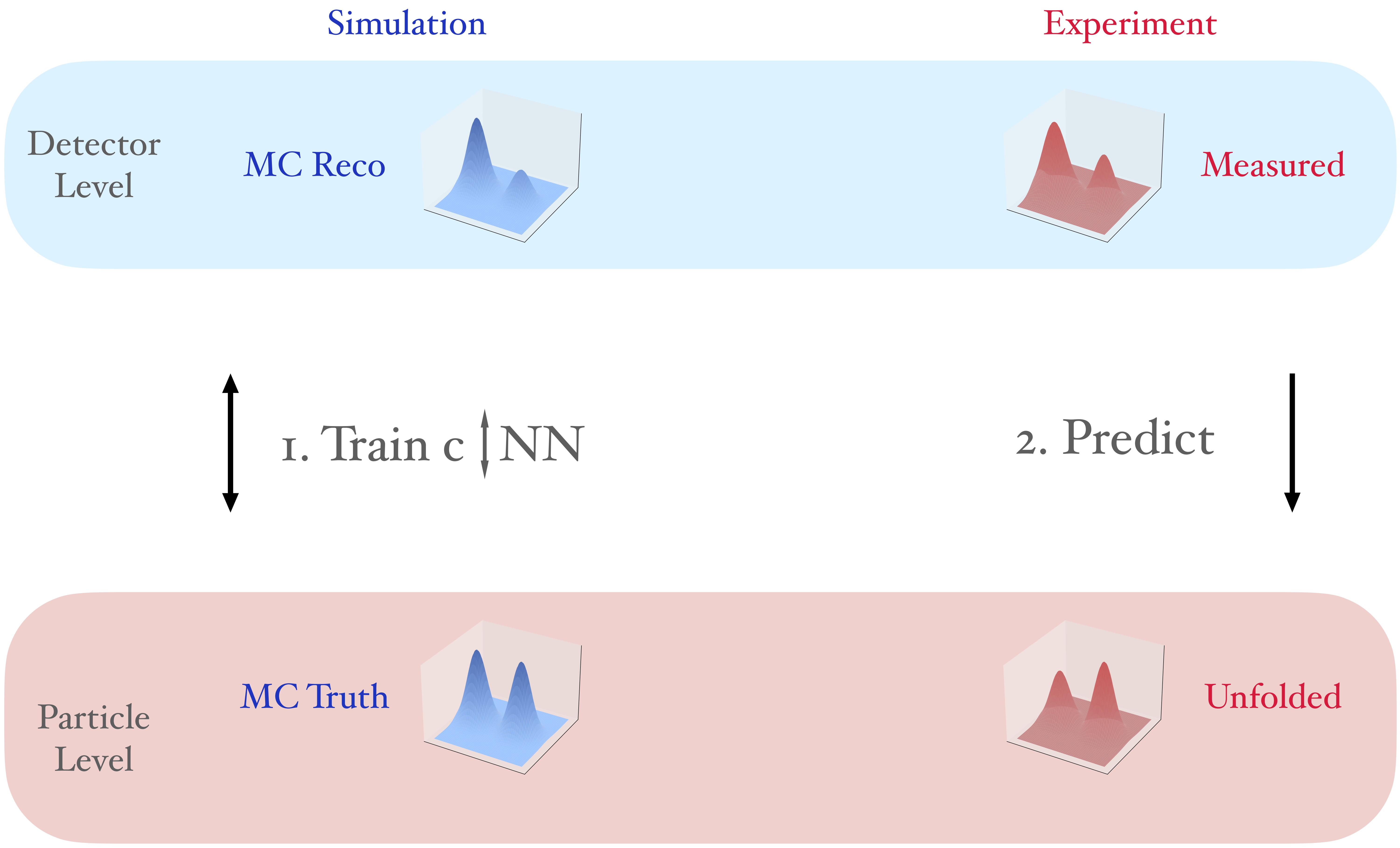
Check calibration of probability density for individual event unfolding



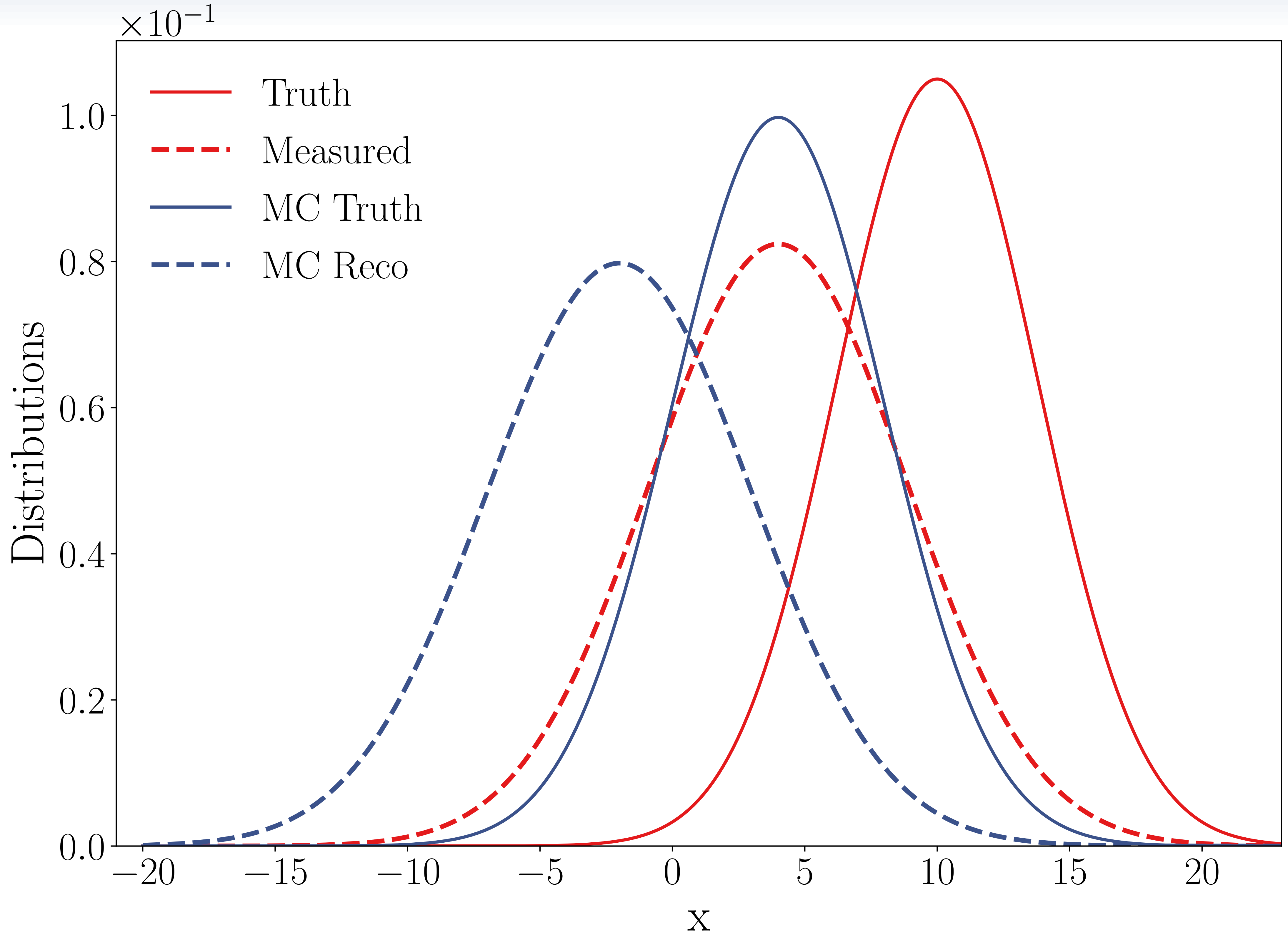
- ☑ High-dimensional
- ☑ Bin-independent
- ☑ Statistically well defined

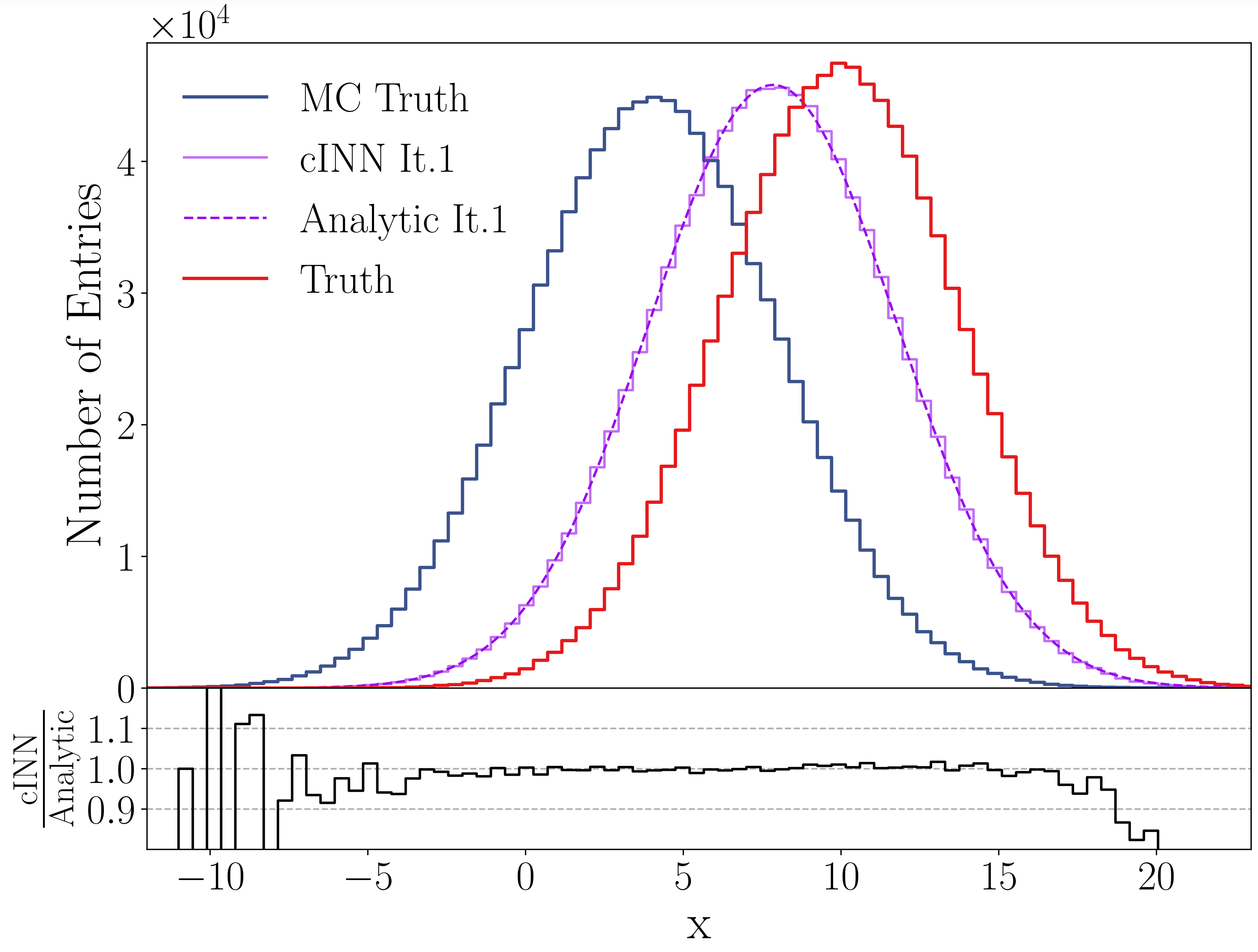
M. Bellagente et al. [2006.06685]

One problem remains (work in progress)



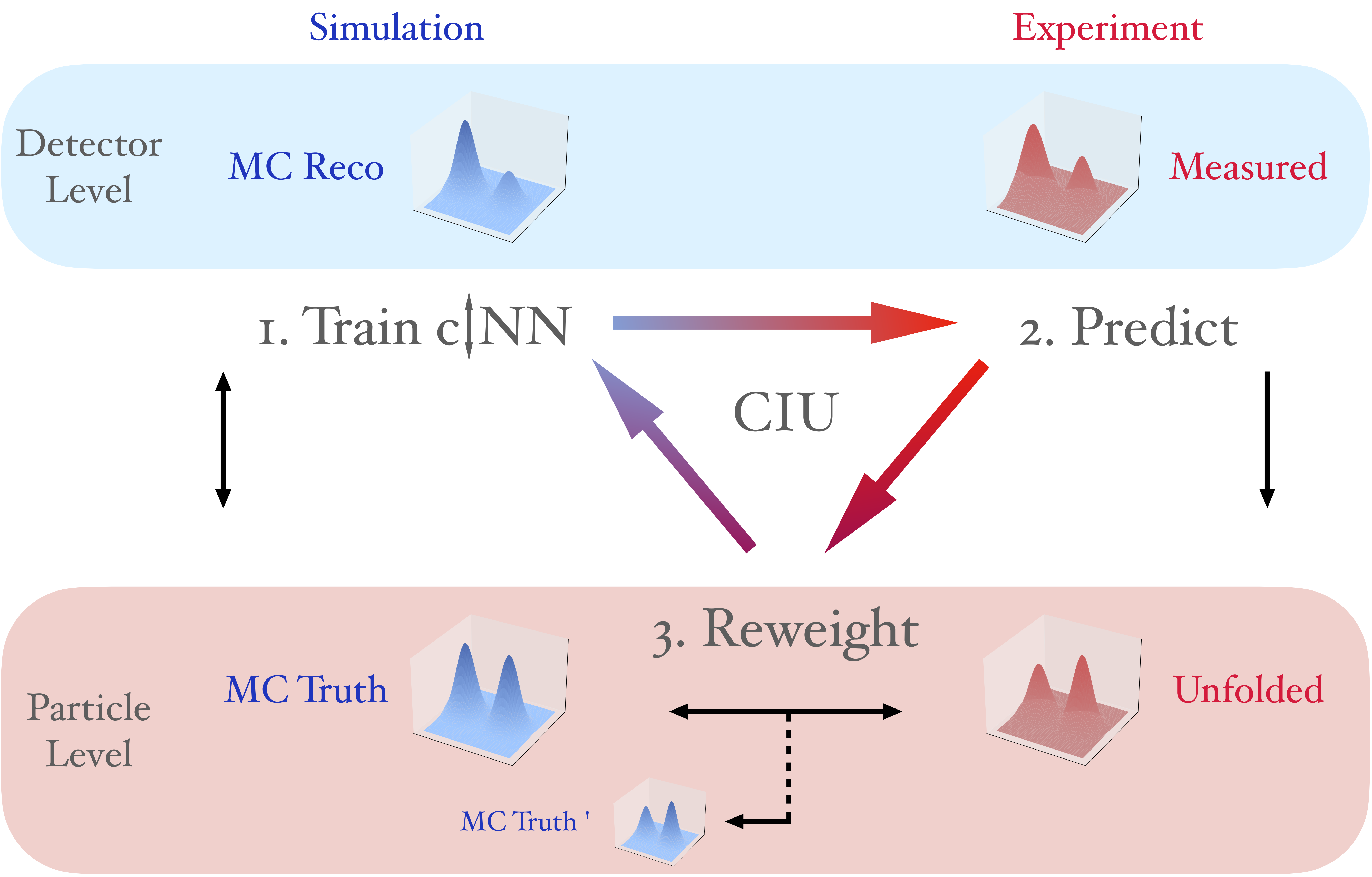
One problem remains

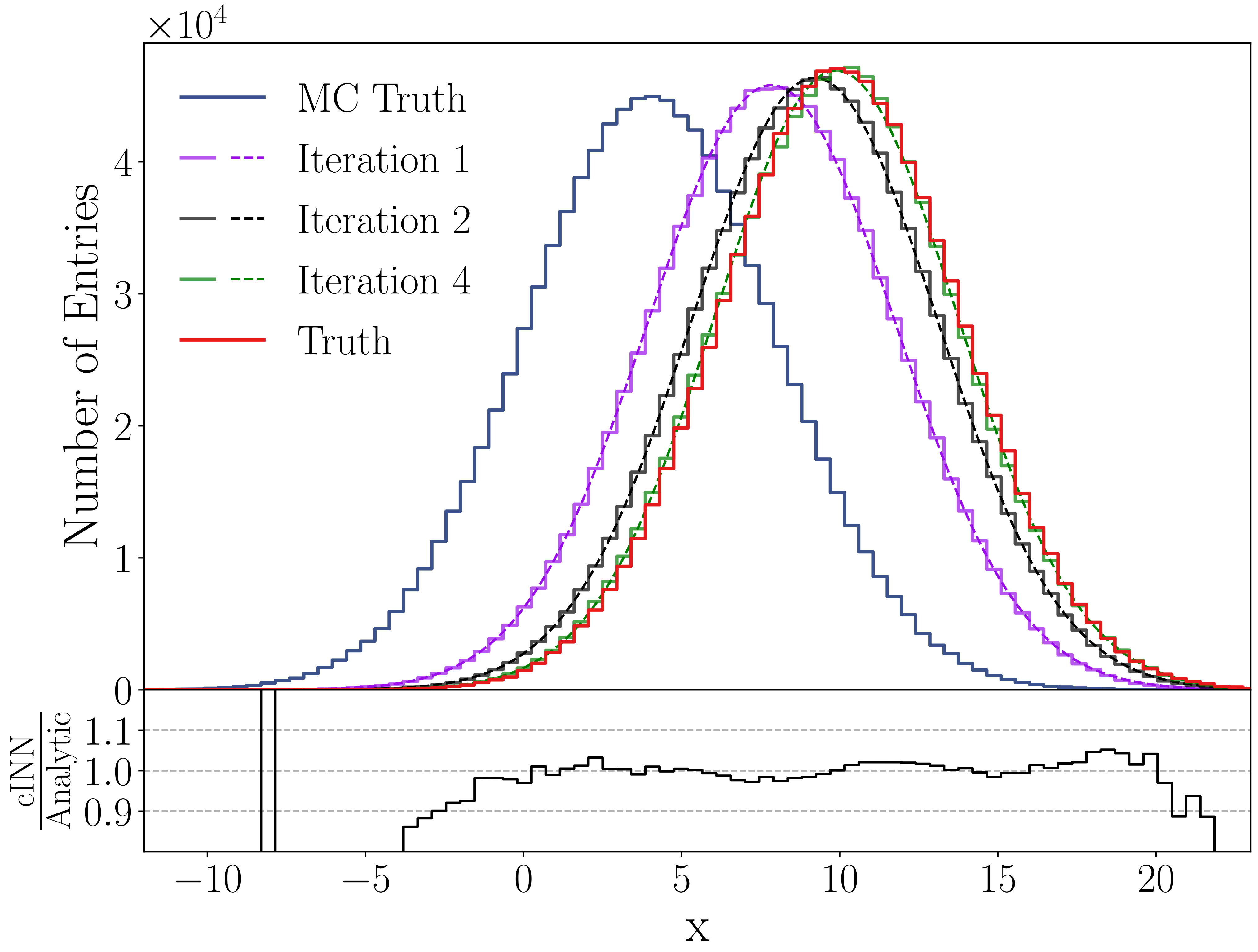




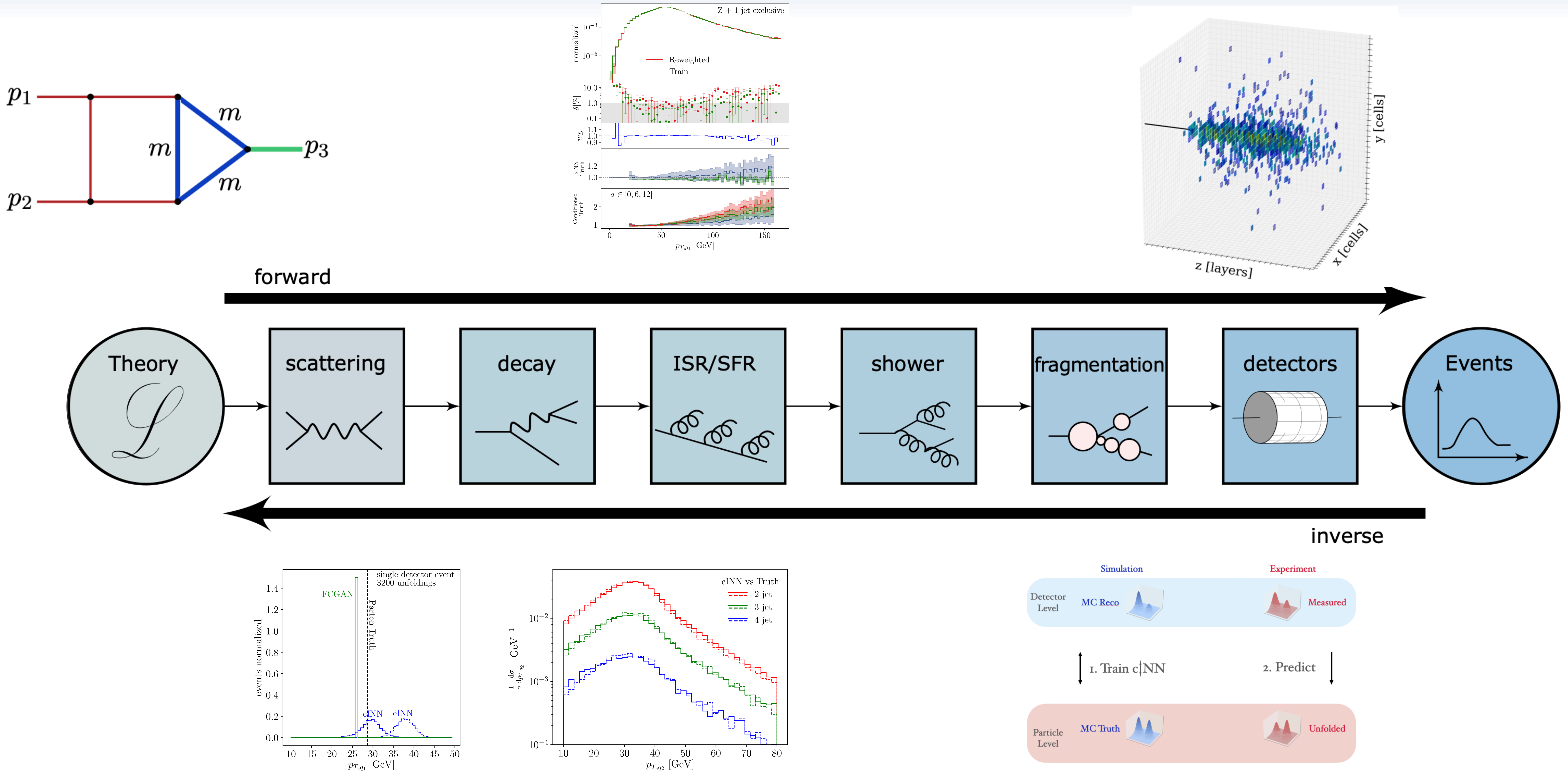
$p(x_{reco}\,|\,x_{parton})$ **is the same, but** $p(x_{parton}\,|\,x_{reco})$ **is not**

One problem remains





Generative networks for better LHC physics

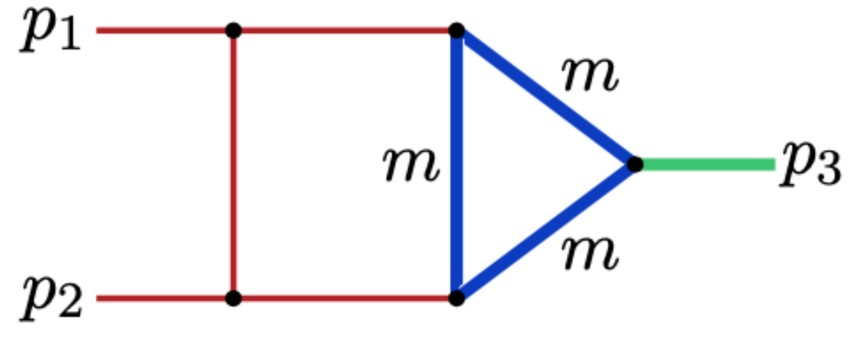


New data are currently on their way...

BACK UP

Multi-loop calculations with NNs

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

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

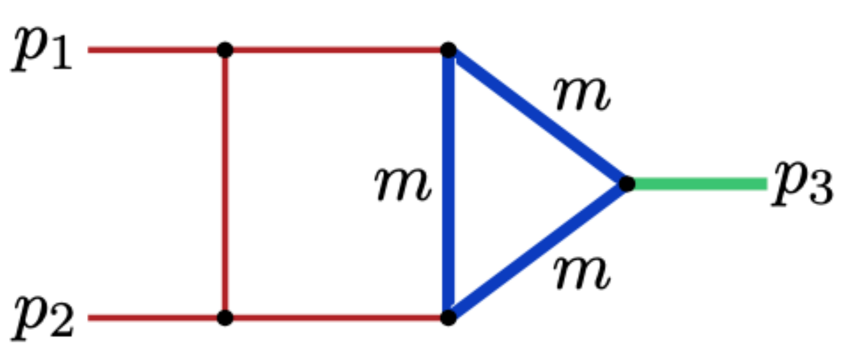
$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \; I(\vec{x})$$

Rewrite with Feynman parameters

Still contains singularities

Multi-loop calculations with NNs

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

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

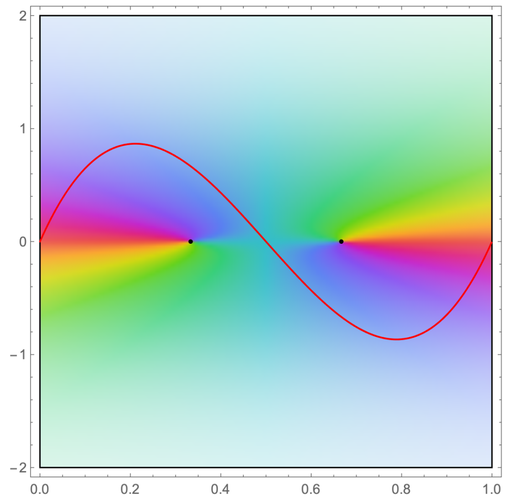
$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, I(\vec{x})$$

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

Integration with normalizing flows

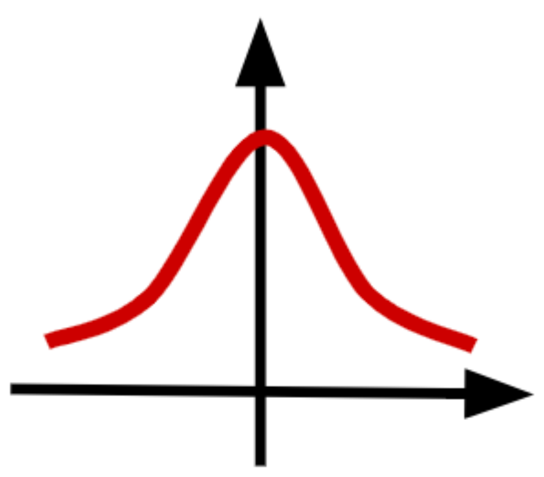
Numeric evaluation of integral $G = \int_0^1 \mathrm{d}x_j \; \det\left(\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right) I(\vec{z}(\vec{x}))$

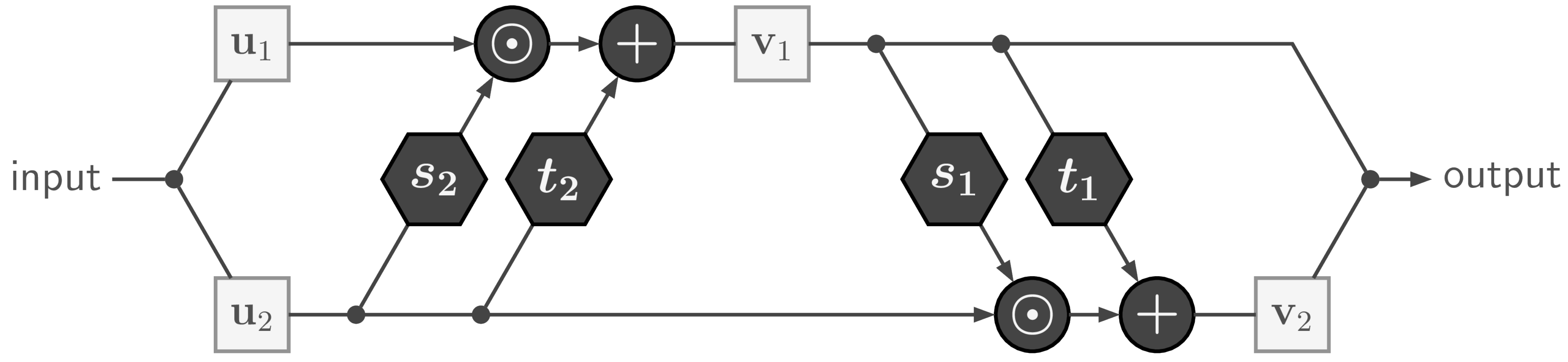
Parametrization $\rightarrow z = \mathrm{INN}(x)$

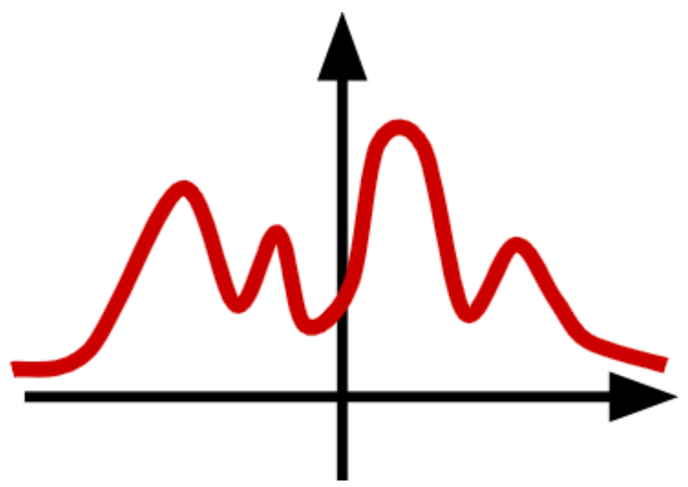
Minimize **variance** $\rightarrow$ loss $\mathscr{L} = \sigma_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left| \det\left(\frac{\partial \vec{z}(\vec{x}_{(i)})}{\partial \vec{x}_{(i)}}\right) I(\vec{z}(\vec{x}_{(i)})) - \langle I \rangle \right|^2$

Normalizing flow networks

+ Bijective mapping
+ Tractable Jacobian
+ Combine many blocks



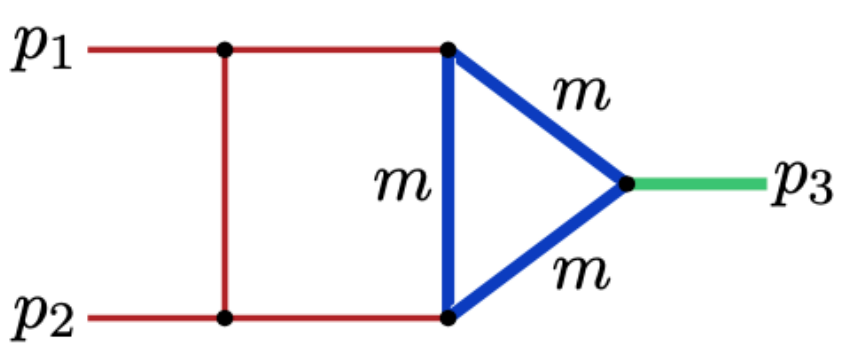




Multi-loop calculations with INNs

Profiting from the Jacobian

Precision predictions based on loop diagrams



Analytic expression for loop amplitude

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

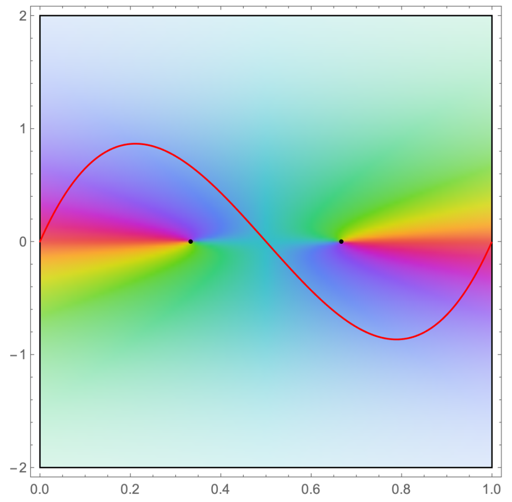
$$= \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \, x_j^{\nu_j - 1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu - LD/2}} = \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j \; I(\vec{x})$$

Rewrite with Feynman parameters

Still contains singularities

Solved by contour deformation due to Cauchy's theorem

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

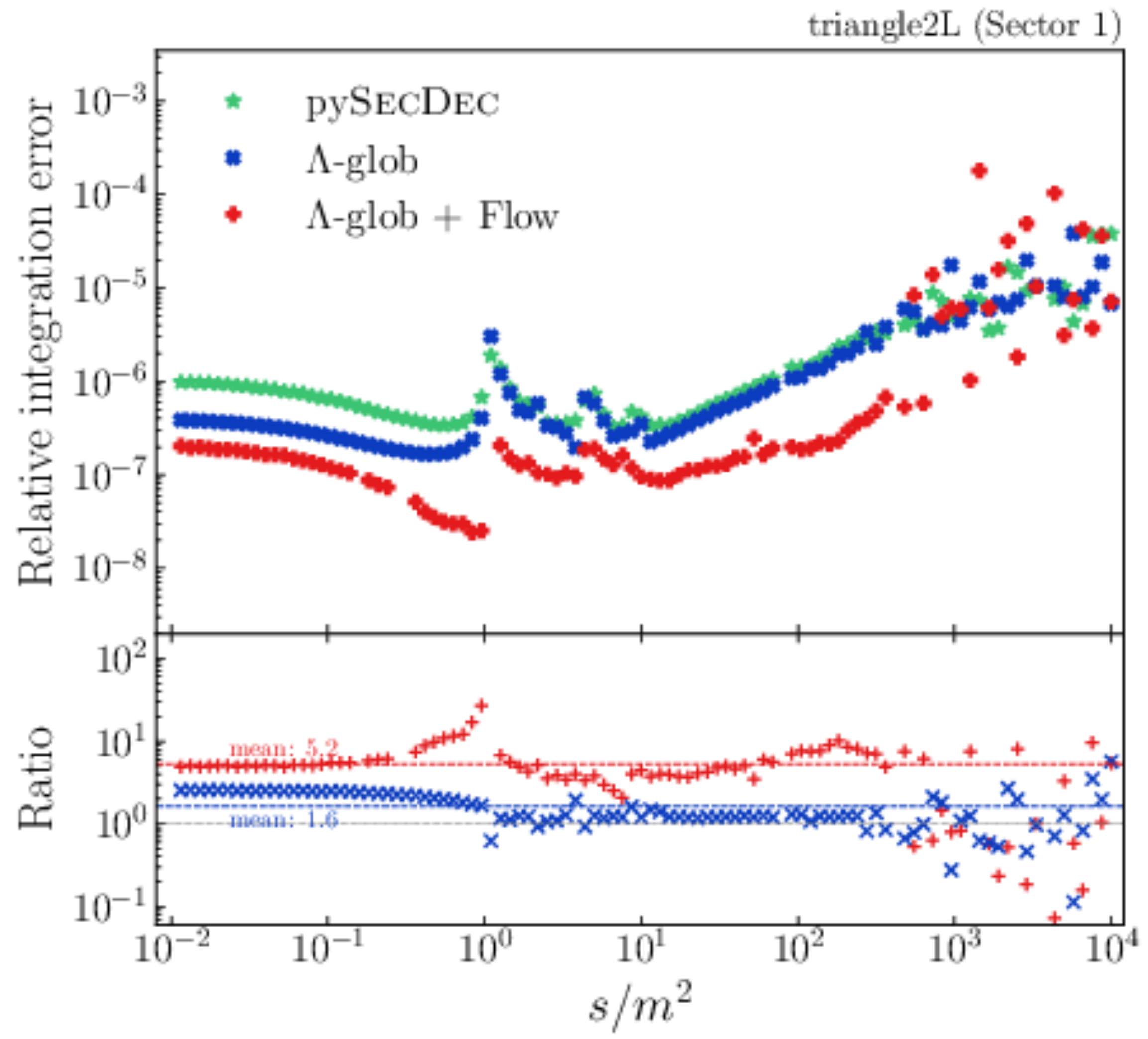


Optimal parametrization = minimal variance

Turn it into an ML Problem

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

Variance $\rightarrow \mathscr{L}$



R. Winterhalder, et al. [2112.09145]