Modern Machine Learning Tools for Unfolding

[arXiv: soon!] Nathan Huetsch, Javier Mariño Villadamigo, Anja Butter, Theo Heimel, Tilman Plehn

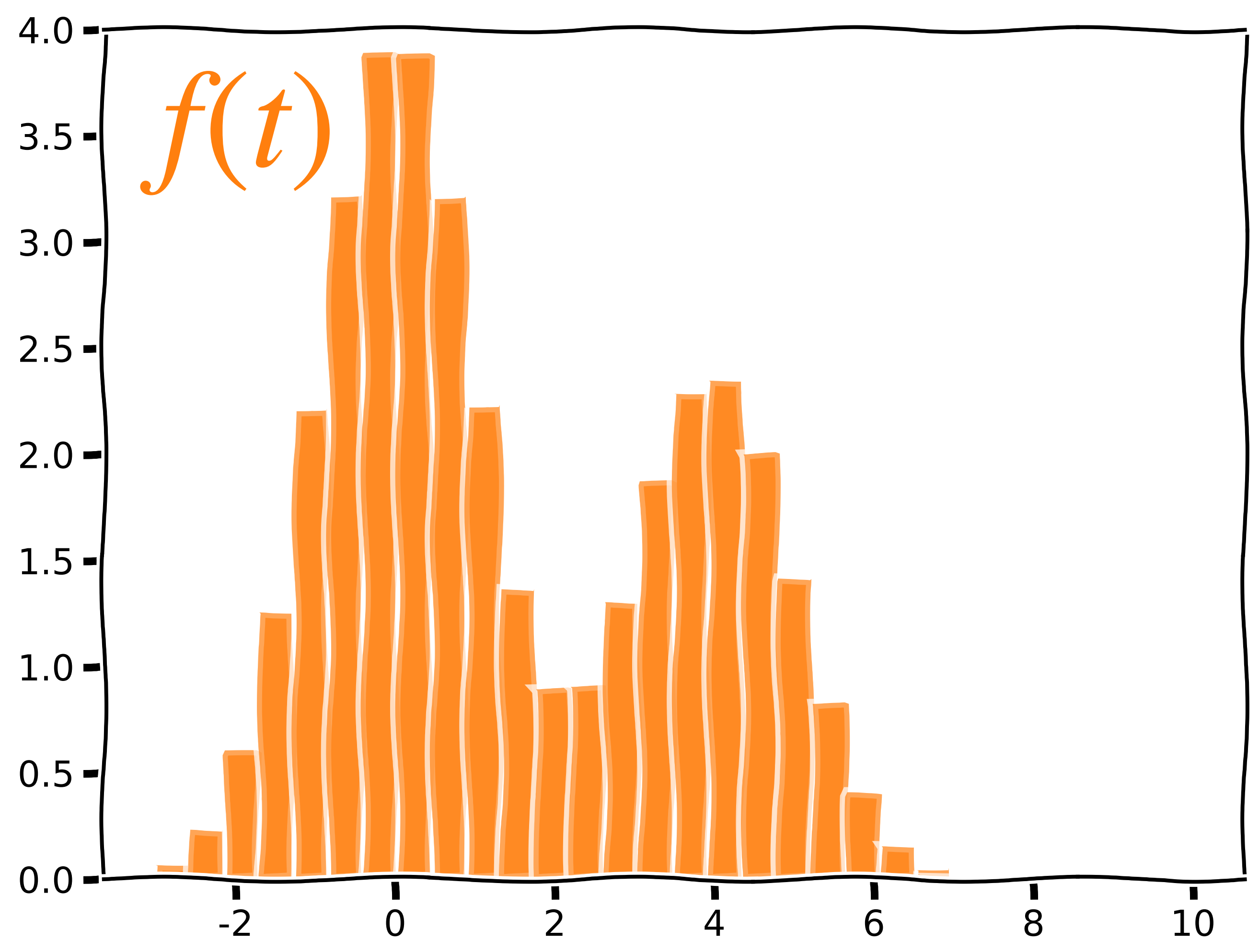
IRN Terascale @ Roma, 16/04/2024

SPONSORED BY THE
Federal Ministry of Education and Research

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386
Institut für Theoretische Physik - University of Heidelberg

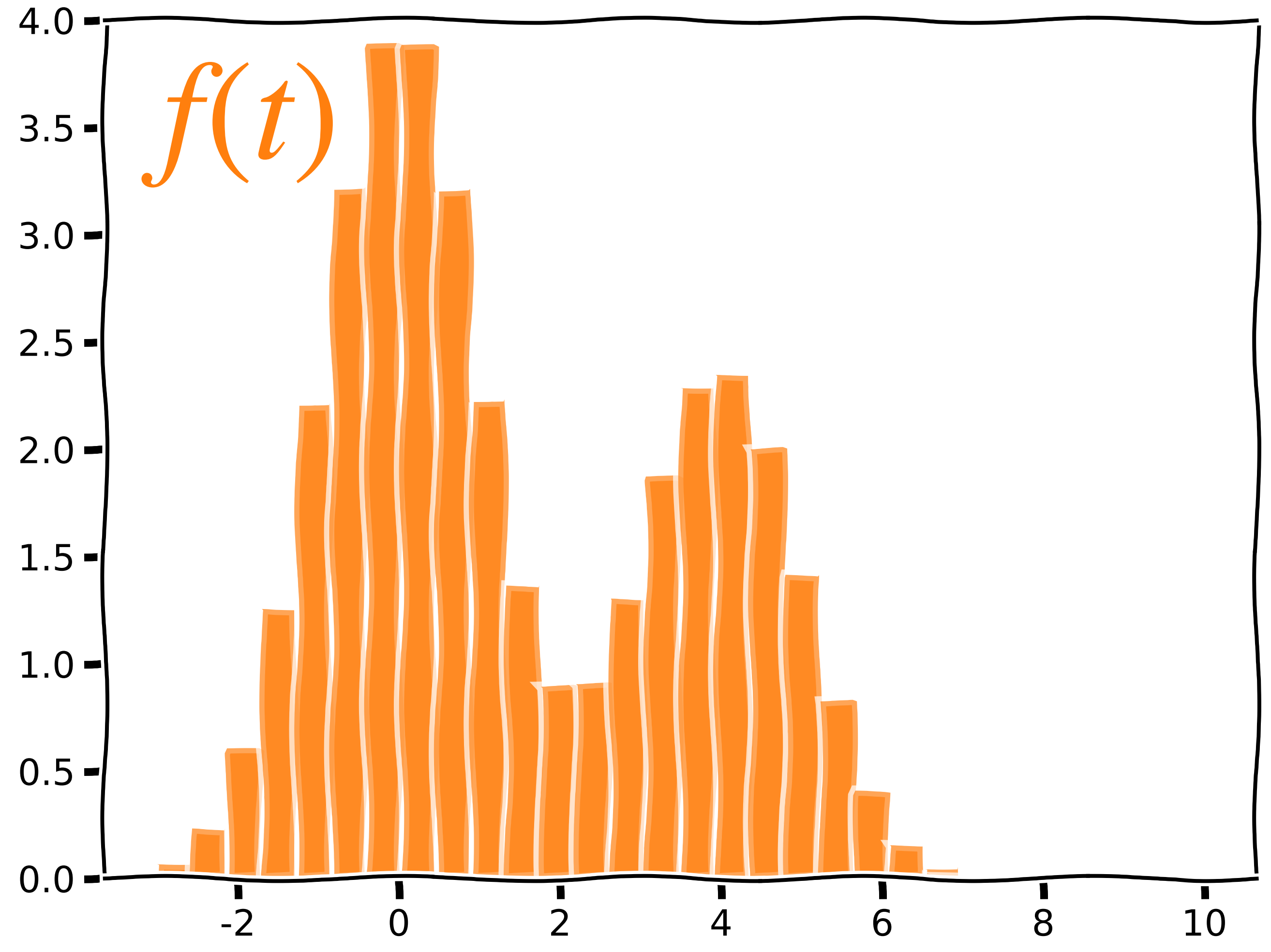
Fundamentals of unfolding

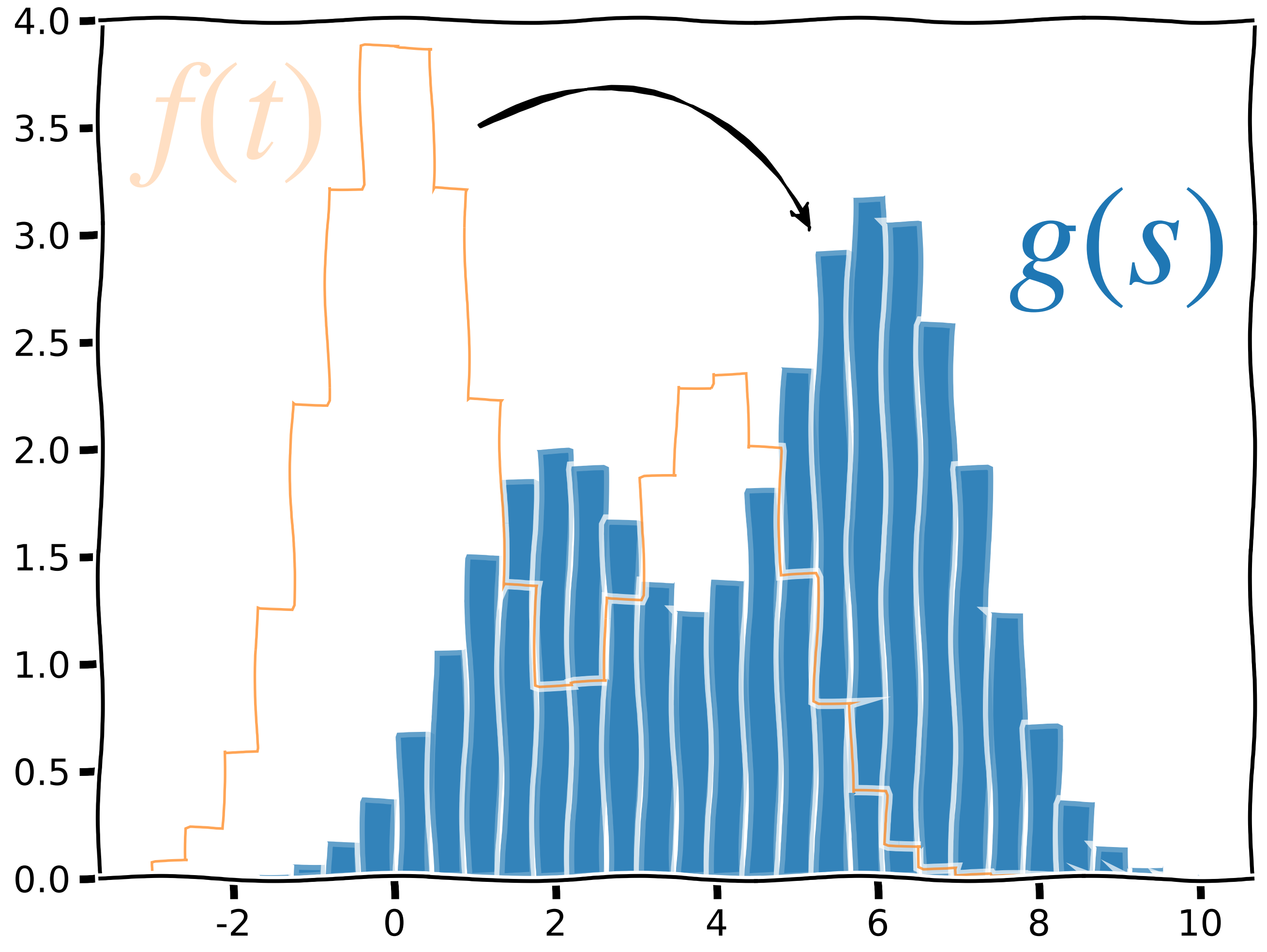
- Distributions $f(t)$ of a physics object t to be measured in experiments are not directly accessible



Fundamentals of unfolding

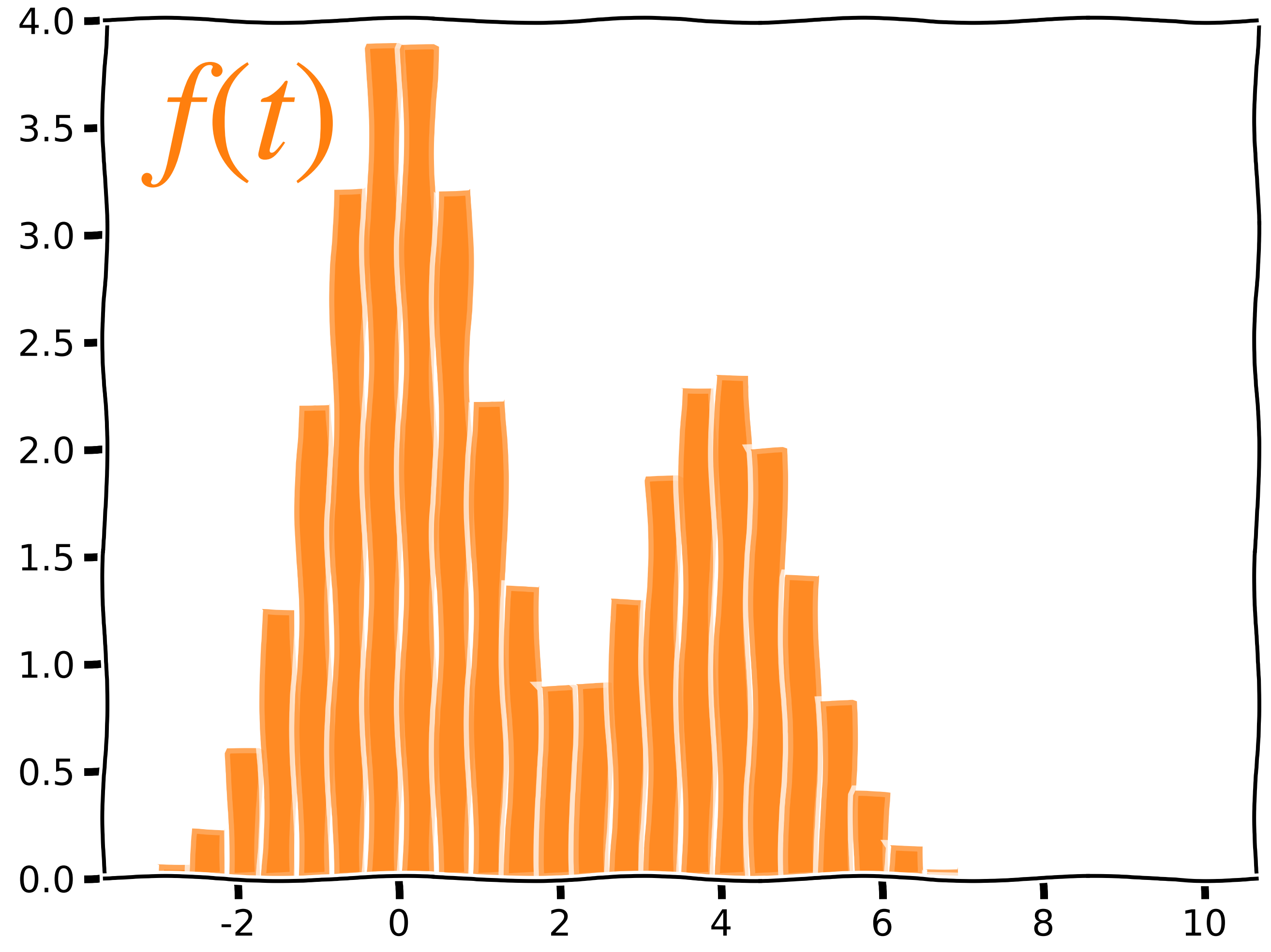
- Distributions $f(t)$ of a physics object t to be measured in experiments are not directly accessible

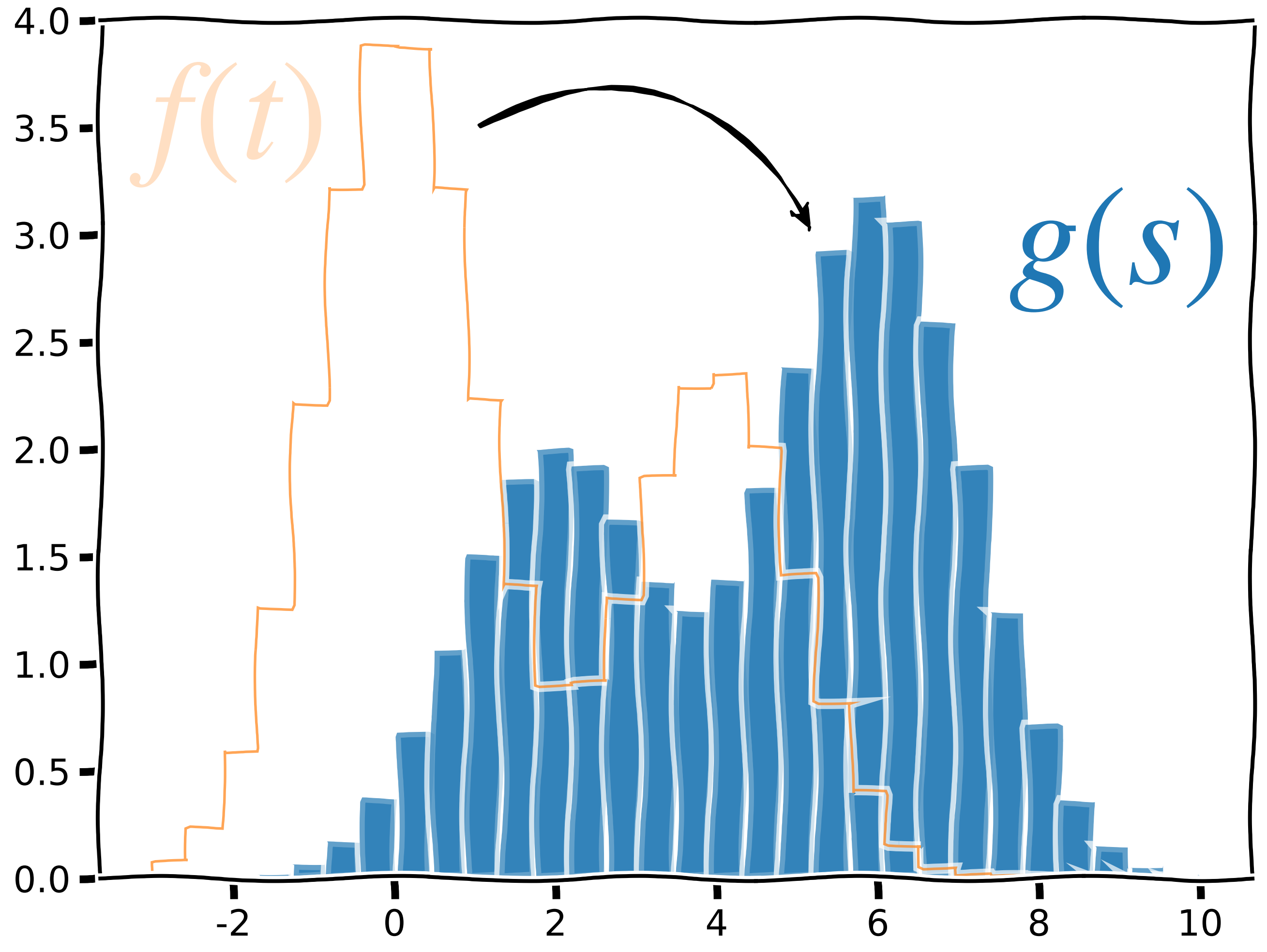




Fundamentals of unfolding

- Distributions $f(t)$ of a physics object t to be measured in experiments are not directly accessible

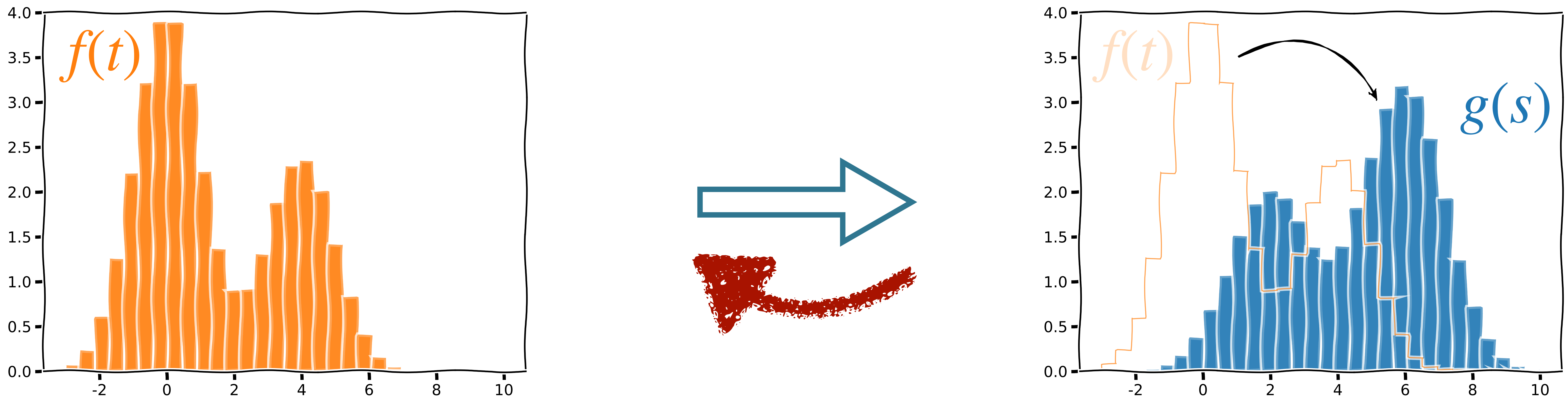




- With MC, the direct process from an assumption $f(t)^{\text{model}}$ to $g(s)$ can be simulated

Fundamentals of unfolding

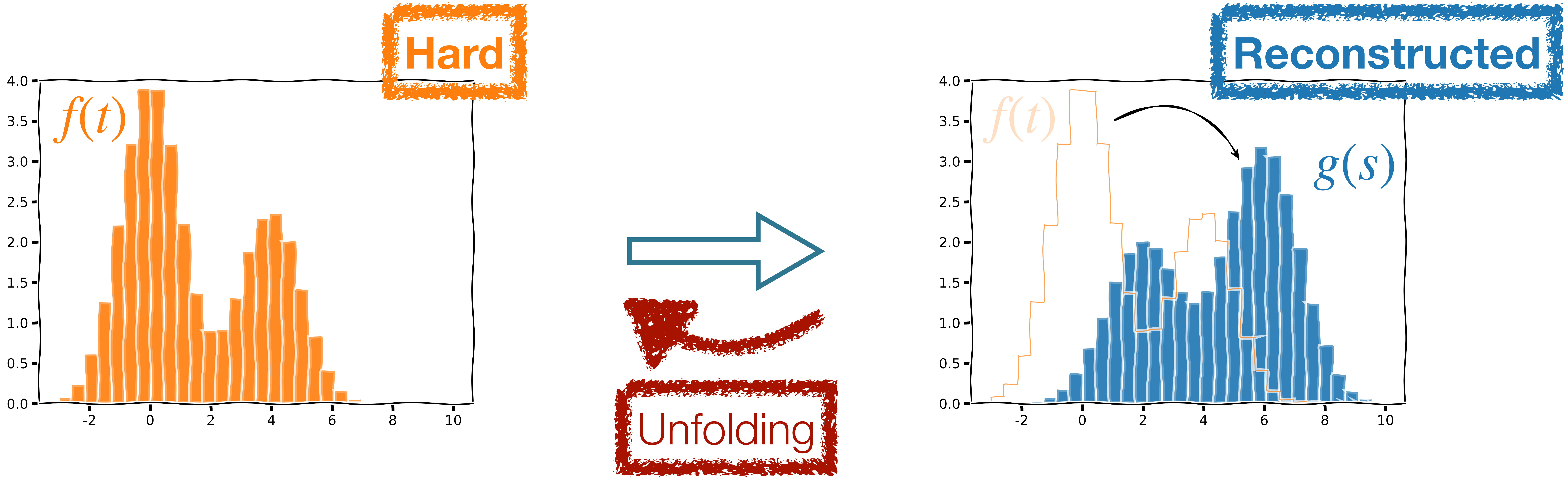
- Distributions $f(t)$ of a physics object t to be measured in experiments are not directly accessible



- With MC, the direct process from an assumption $f(t)^{\mathrm{model}}$ to $g(s)$ can be simulated
- The inverse process is ill-posed: small changes in $g(s)$ can cause large shifts in the $\tilde{f}(t)$

Fundamentals of unfolding

- Distributions $f(t)$ of a physics object t to be measured in experiments are not directly accessible



- With MC, the direct process from an assumption $f(t)^{\text{model}}$ to $g(s)$ can be simulated
- The inverse process is ill-posed: small changes in $g(s)$ can cause large shifts in the $\tilde{f}(t)$

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int R(s \,|\, t)\, f(t)\ \mathrm{d}t$$

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \boxed{R(s\,|\,t)}\, f(t)\ \mathrm{d}t$$

detector
response
matrix

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \boxed{R(s\,|\,t)}\, f(t)\ \mathrm{d}t \quad \xrightarrow{\text{binning}} \quad r_i = \sum_j R_{ij} \cdot t_j$$

detector
response
matrix

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \underbrace{\boxed{R(s \mid t)}}_{\text{detector response matrix}} f(t)\ \mathrm{d}t \xrightarrow{\text{binning}} r_i = \sum_j R_{ij} \cdot t_j$$

  - Various ways to invert the detector response matrix: SVD, IBU, IDS, etc.

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \underbrace{\boxed{R(s\,|\,t)}}_{\text{detector response matrix}} f(t)\ \mathrm{d}t \quad \xrightarrow{\text{binning}} \quad r_i = \sum_j R_{ij} \cdot t_j$$

  - Various ways to invert the detector response matrix: SVD, IBU, IDS, etc.
  - General need for regularization: trade-off between bias and statistical uncertainty

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \underbrace{\boxed{R(s\,|\,t)}}_{\text{detector response matrix}} f(t)\ \mathrm{d}t \quad \xrightarrow[\text{binning}]{} \quad r_i = \sum_j R_{ij} \cdot t_j$$

  - Various ways to invert the detector response matrix: SVD, IBU, IDS, etc.
  - General need for regularization: trade-off between bias and statistical uncertainty
  - Requires binning and can only unfold a few dimensions

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \boxed{R(s\,|\,t)}\, f(t)\ \mathrm{d}t \quad \xrightarrow{\text{binning}} \quad r_i = \sum_j R_{ij} \cdot t_j$$

detector response matrix

  - Various ways to invert the detector response matrix: SVD, IBU, IDS, etc.
  - General need for regularization: trade-off between bias and statistical uncertainty
  - Requires binning and can only unfold a few dimensions

With neural networks:

- **ML-based** unfolding

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \boxed{R(s\,|\,t)}\, f(t)\ \mathrm{d}t \quad \xrightarrow{\text{binning}} \quad r_i = \sum_j R_{ij} \cdot t_j$$

detector response matrix

  - Various ways to invert the detector response matrix: SVD, IBU, IDS, etc.
  - General need for regularization: trade-off between bias and statistical uncertainty
  - Requires binning and can only unfold a few dimensions

With neural networks:

- **ML-based** unfolding
  - Unbinned: advantageous if one wants to derive quantities from the unfolding observables

Why neural networks?

Traditionally:

- **Matrix-based** unfolding

$$g(s) = \int \underbrace{\boxed{R(s\,|\,t)}}_{\text{detector response matrix}} f(t)\;\mathrm{d}t \quad \xrightarrow{\text{binning}} \quad r_i = \sum_j R_{ij} \cdot t_j$$

  - Various ways to invert the detector response matrix: SVD, IBU, IDS, etc.
  - General need for regularization: trade-off between bias and statistical uncertainty
  - Requires binning and can only unfold a few dimensions

With neural networks:

- **ML-based** unfolding
  - Unbinned: advantageous if one wants to derive quantities from the unfolding observables
  - Allows to unfold (and account for correlations in) many dimensions

Several approaches

Event reweighting

- Omnifold [1911.09107]
- (*)

Distribution mapping

- Direct Diffusion [2311.17175]
- Schrödinger Bridge [2308.12351]
- (*)

Conditional phase space sampling

- GANs [1912.00477]
- Latent Diffusion [2305.10399]
- Conditional Flow Matching [2305.10475]
- cINN [2212.08674, 2006.06685]
- (*)

(*) These are not comprehensive lists. For a more extensive catalogue see for example the HEP ML Living Review

Several approaches

Event reweighting

- Omnifold [1911.09107]
- (*)

Distribution mapping

- Direct Diffusion [2311.17175]
- Schrödinger Bridge [2308.12351]
- (*)

Conditional phase space sampling

- GANs [1912.00477]
- Latent Diffusion [2305.10399]
- Conditional Flow Matching [2305.10475]
- cINN [2212.08674, 2006.06685]
- (*)

(*) These are not comprehensive lists. For a more extensive catalogue see for example the HEP ML Living Review

Several approaches

Event reweighting

- Omnifold [1911.09107]
- (*)

Distribution mapping

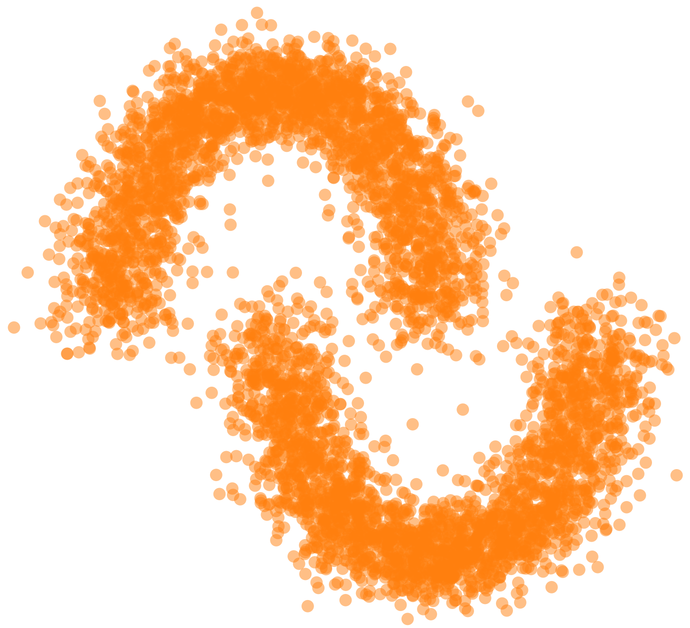
- Direct Diffusion [2311.17175]
- Schrödinger Bridge [2308.12351]
- (*)

Conditional phase space sampling

- GANs [1912.00477]
- Latent Diffusion [2305.10399]
- Conditional Flow Matching [2305.10475]
- cINN [2212.08674, 2006.06685]
- (*)

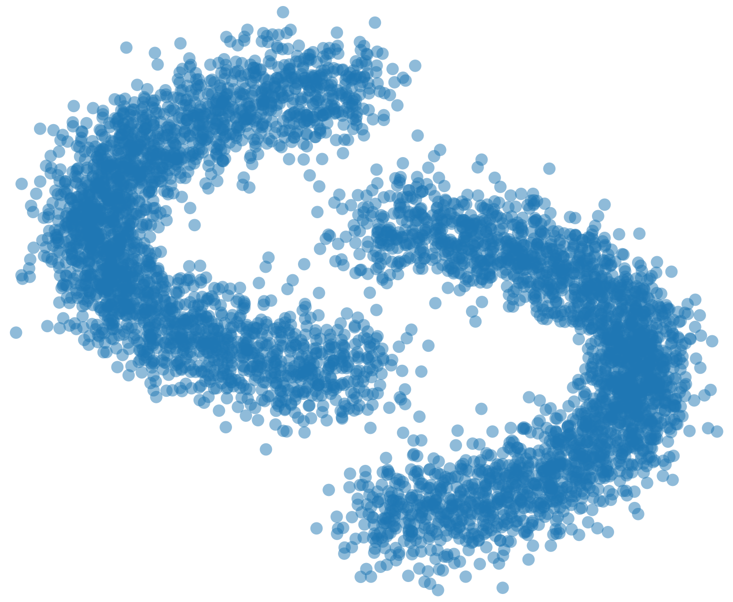
(*) These are not comprehensive lists. For a more extensive catalogue see for example the HEP ML Living Review

Direct Diffusion (DiDi)



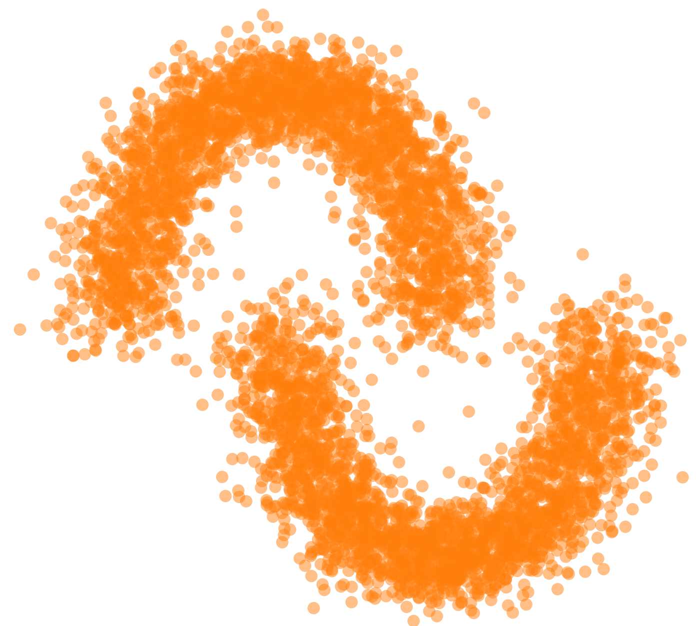
$x_0 \sim p_{\text{model}}(x_{\text{hard}})$

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t)$$

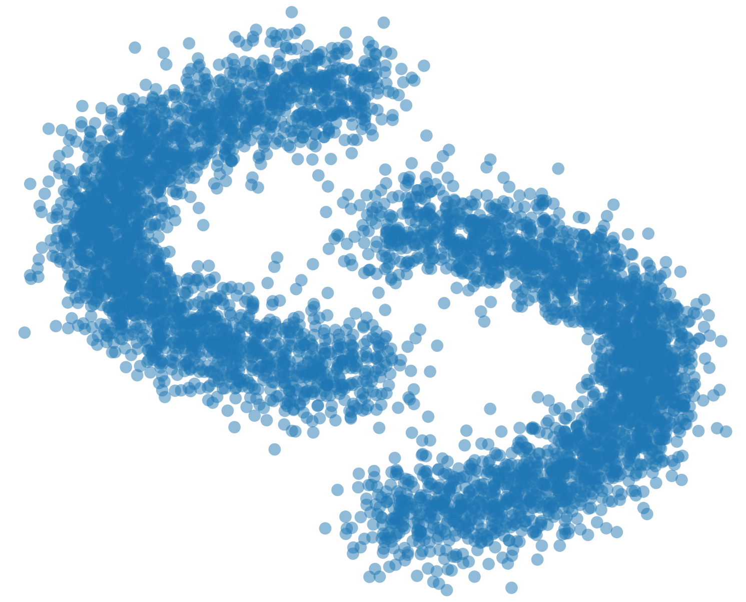


$x_1 \sim p_{\text{reco}}(x_{\text{reco}})$

Direct Diffusion (DiDi)



$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t)$$

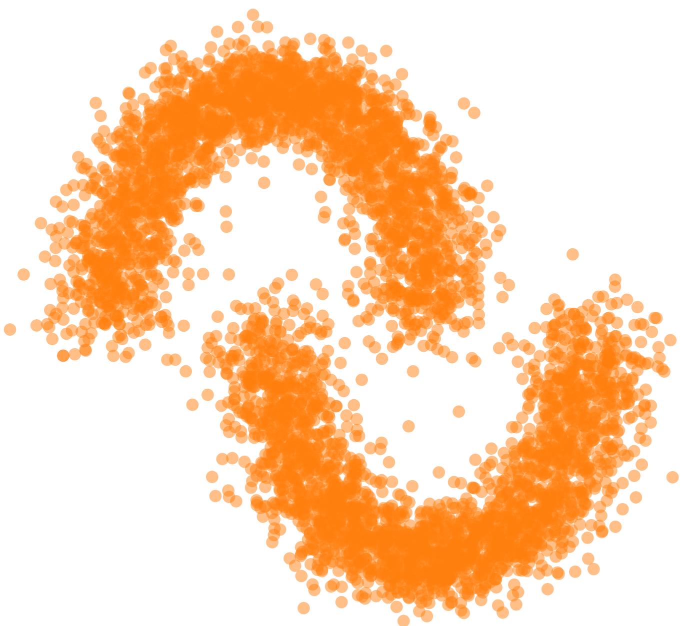


$x_0 \sim p_{\mathrm{model}}(x_{\mathrm{hard}})$

$x_1 \sim p_{\mathrm{reco}}(x_{\mathrm{reco}})$

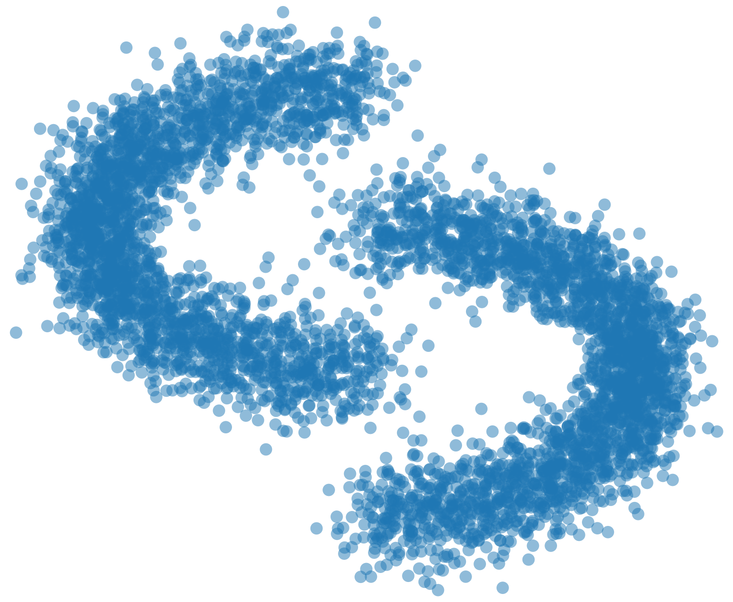
- Connect x_0 and x_1 with a linear trajectory: $x(t) = (1 - t)x_0 + tx_1$

Direct Diffusion (DiDi)



$x_0 \sim p_{\text{model}}(x_{\text{hard}})$

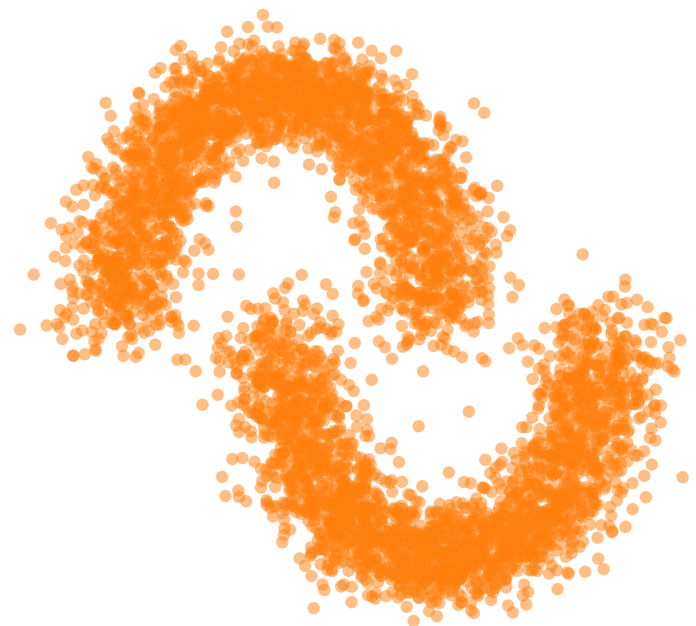
$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t)$$



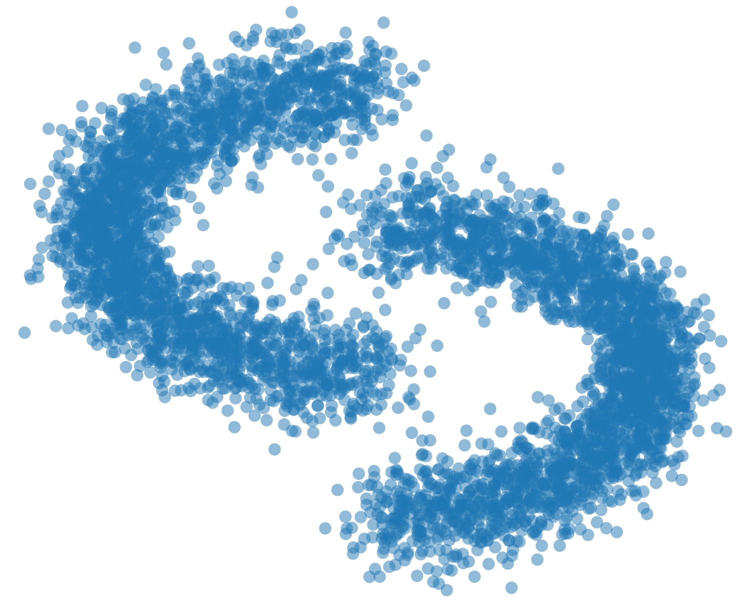
$x_1 \sim p_{\text{reco}}(x_{\text{reco}})$

- Connect x_0 and x_1 with a linear trajectory: $x(t) = (1 - t)x_0 + tx_1$
- The NN is regressed to predict the velocity field: $v_\theta(x(t), t) \approx \frac{\mathrm{dx(t)}}{\mathrm{dt}} = x_1 - x_0$

Direct Diffusion (DiDi)



$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t)$$

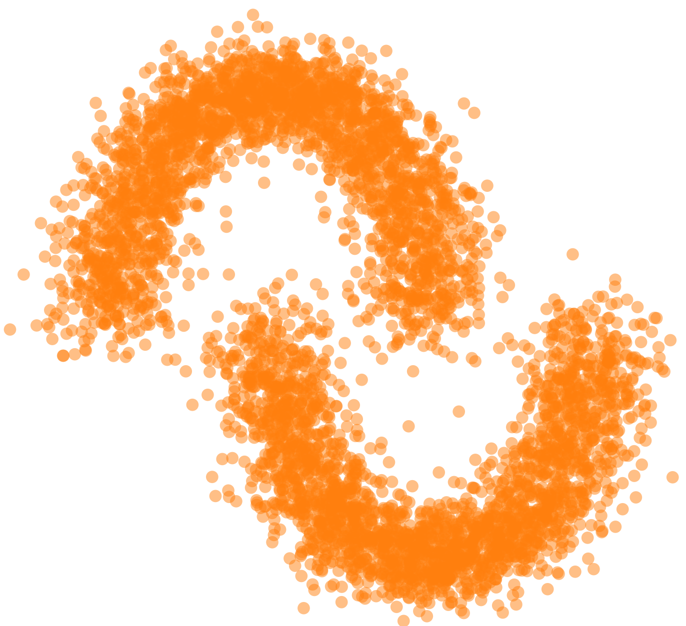


$x_0 \sim p_{\mathrm{model}}(x_{\mathrm{hard}})$

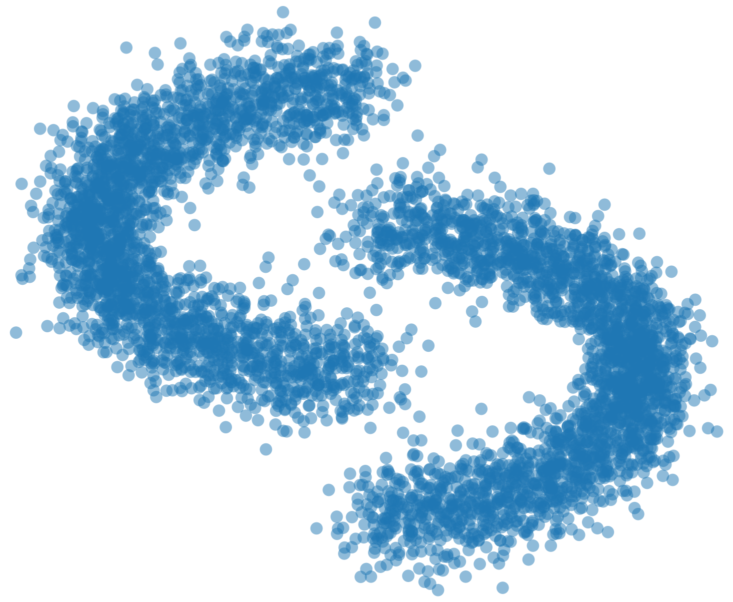
$x_1 \sim p_{\mathrm{reco}}(x_{\mathrm{reco}})$

- Connect x_0 and x_1 with a linear trajectory: $x(t) = (1-t)x_0 + tx_1$
- The NN is regressed to predict the velocity field: $v_\theta(x(t), t) \approx \frac{\mathrm{dx(t)}}{\mathrm{dt}} = x_1 - x_0$
- For sampling, solve ODE starting from x_1: $x_0 = x_1 + \int_1^0 v_\theta(x(t), t)\mathrm{dt}$

Direct Diffusion (DiDi)



$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t)$$



$x_0 \sim p_{\mathrm{model}}(x_{\mathrm{hard}})$

$x_1 \sim p_{\mathrm{reco}}(x_{\mathrm{reco}})$

- Connect x_0 and x_1 with a linear trajectory: $x(t) = (1-t)x_0 + tx_1$
- The NN is regressed to predict the velocity field: $v_\theta(x(t), t) \approx \frac{\mathrm{dx(t)}}{\mathrm{dt}} = x_1 - x_0$
- For sampling, solve ODE starting from x_1: $x_0 = x_1 + \int_1^0 v_\theta(x(t), t)\mathrm{dt}$
- Loss: $\mathscr{L}_{\mathrm{DiDi}} = \left\langle [v_\theta((1-t)x_0 + tx_1,\ t) - (x_1 - x_0)]^2 \right\rangle_{t \sim \mathscr{U}([0,1]),\ (x_0, x_1) \sim p(x_{\mathrm{hard}}, x_{\mathrm{reco}})}$

Several methods

Event reweighting

- Omnifold [1911.09107]
- (*)

Distribution mapping

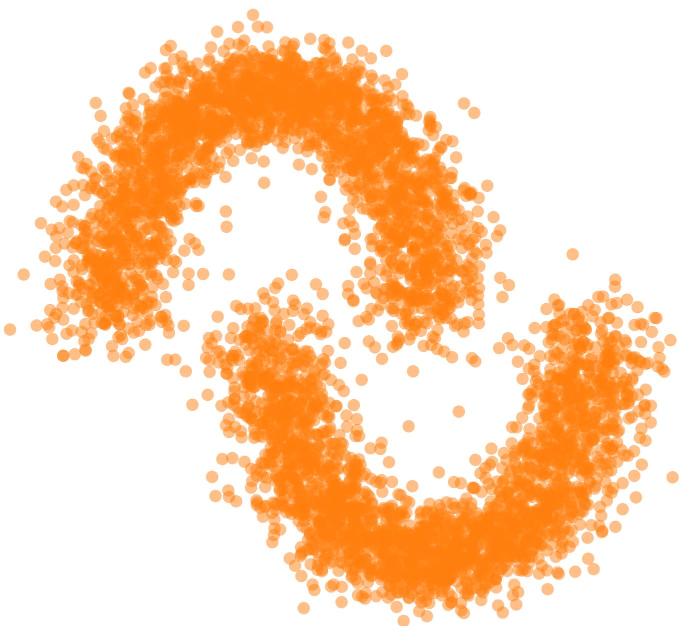
- Direct Diffusion [2311.17175]
- Schrödinger Bridge [2308.12351]
- (*)

Conditional phase space sampling

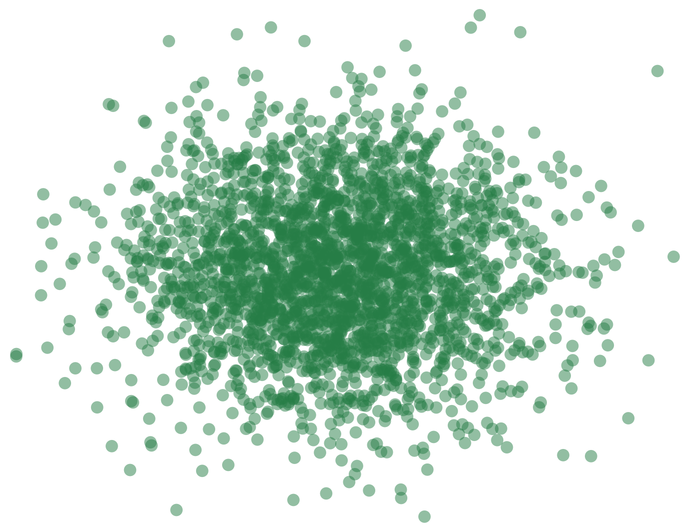
- GANs [1912.00477]
- Latent Diffusion [2305.10399]
- Conditional Flow Matching [2305.10475]
- cINN [2212.08674, 2006.06685]
- (*)

(*) These are not comprehensive lists. For a more extensive catalogue see for example the HEP ML Living Review

Conditional Flow Matching (CFM)



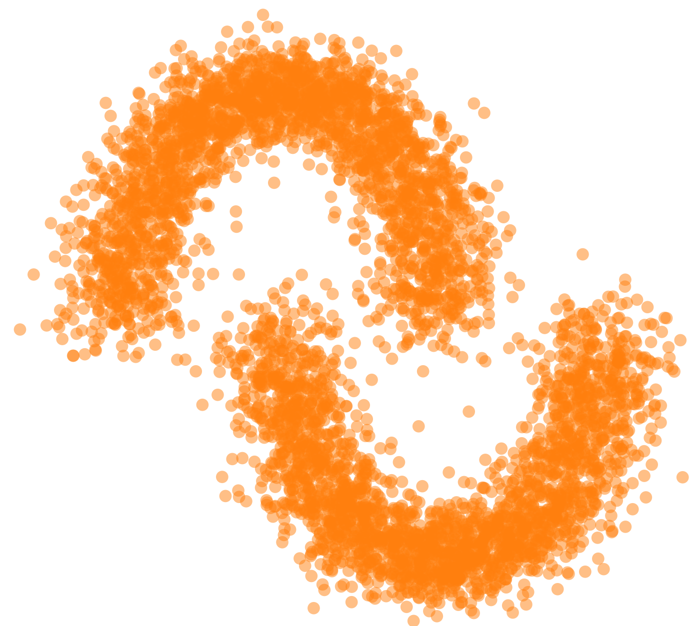
$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t \,|\, x_{\mathrm{reco}})$$



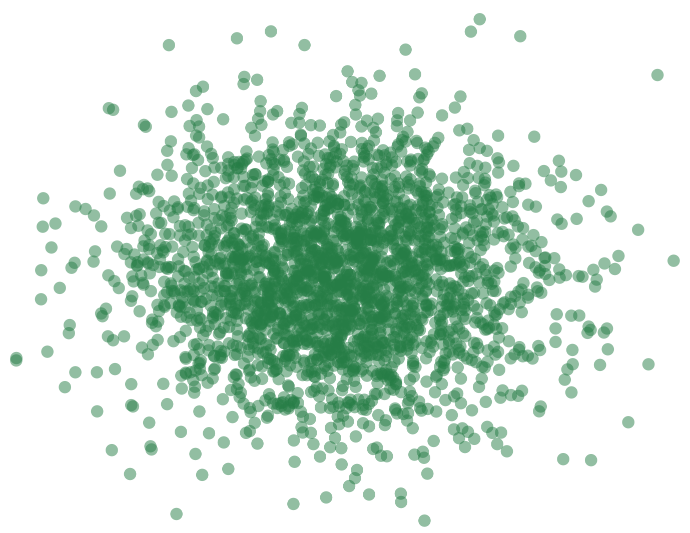
$$x_0 \sim p_{\mathrm{model}}(x_{\mathrm{hard}} \,|\, x_{\mathrm{reco}})$$

$$\color{red}{\epsilon} = z \sim p_{\mathrm{latent}}(z)$$

Conditional Flow Matching (CFM)



$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t \,|\, x_{\text{reco}})$$



$x_0 \sim p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}})$

$\epsilon = z \sim p_{\text{latent}}(z)$

- Connect x_0 and ϵ with a linear trajectory: $x(t) = (1-t)x_0 + t\epsilon$
- The NN is regressed to predict the velocity field: $v_\theta(x(t), t \,|\, x_{\text{reco}}) \approx \dfrac{\mathrm{dx(t)}}{\mathrm{dt}} = \epsilon - x_0$
- For sampling, solve ODE starting from ϵ: $x_0 = \epsilon + \int_1^0 v_\theta(x(t), t \,|\, x_{\text{reco}})\mathrm{dt}$
- Loss: $\mathcal{L}_{\text{CFM}} = \left\langle [v_\theta((1-t)x_0 + t\epsilon,\ t,\ x_{\text{reco}}) - (\epsilon - x_0)]^2 \right\rangle_{t\sim\mathcal{U}([0,1]),\ (x_0, x_{\text{reco}})\sim p(x_{\text{hard}}, x_{\text{reco}}),\ \epsilon\sim\mathcal{N}(0,1)}$

Several methods

Event reweighting

- Omnifold [1911.09107]
- (*)

Distribution mapping

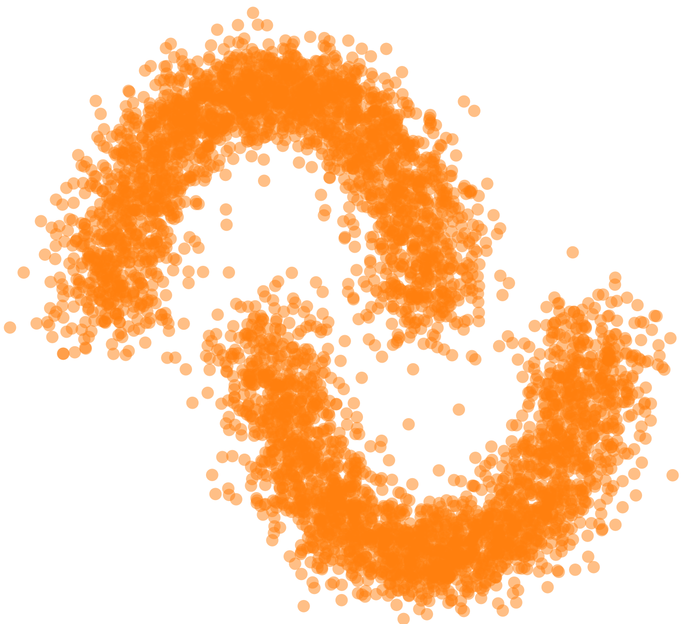
- Direct Diffusion [2311.17175]
- Schrödinger Bridge [2308.12351]
- (*)

Conditional phase space sampling

- GANs [1912.00477]
- Latent Diffusion [2305.10399]
- Conditional Flow Matching [2305.10475]
- cINN [2212.08674, 2006.06685]
- (*)

(*) These are not comprehensive lists. For a more extensive catalogue see for example the HEP ML Living Review

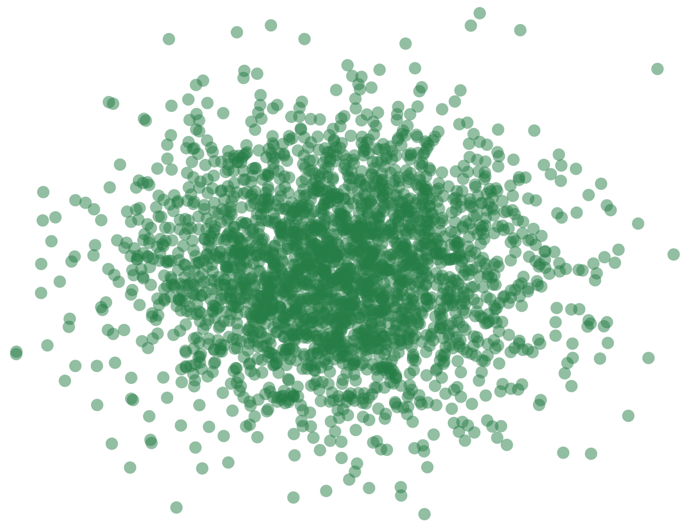
Conditional INN (cINN)



$x_0 \sim p_{\mathrm{model}}(x_{\mathrm{hard}} \,|\, x_{\mathrm{reco}})$

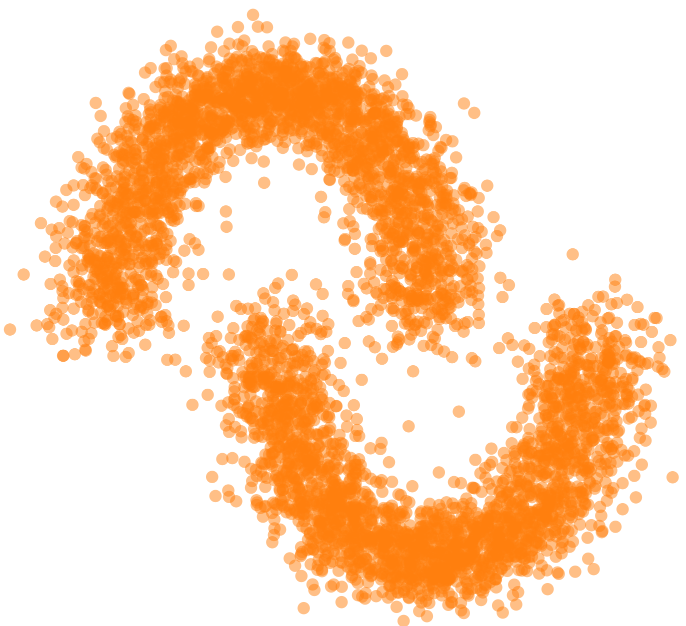
$g_\theta(x_{\mathrm{hard}} \,|\, x_{\mathrm{reco}})$

$g_\theta^{-1}(z \,|\, x_{\mathrm{reco}})$



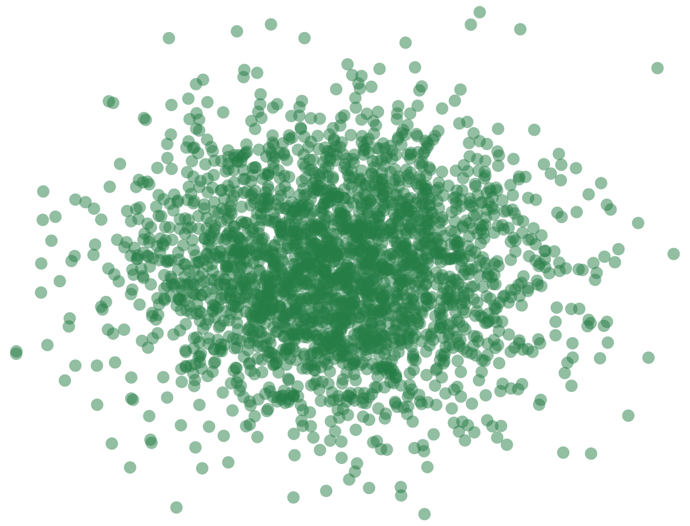
$z \sim p_{\mathrm{latent}}(z)$

Conditional INN (cINN)



$g_\theta(x_{\text{hard}} \,|\, x_{\text{reco}})$

$g_\theta^{-1}(z \,|\, x_{\text{reco}})$



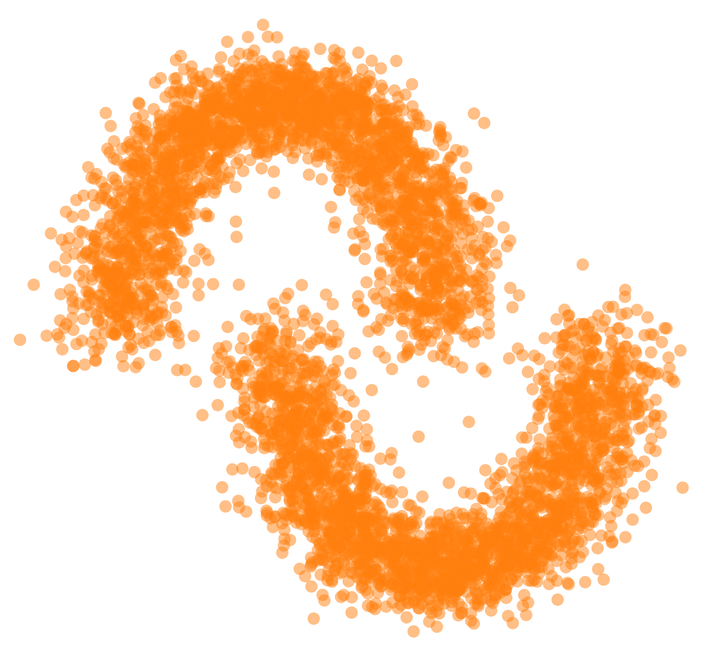
$x_0 \sim p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}})$

$z \sim p_{\text{latent}}(z)$

- Bijective function between $p_{\text{latent}}(z)$ and $p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}})$:

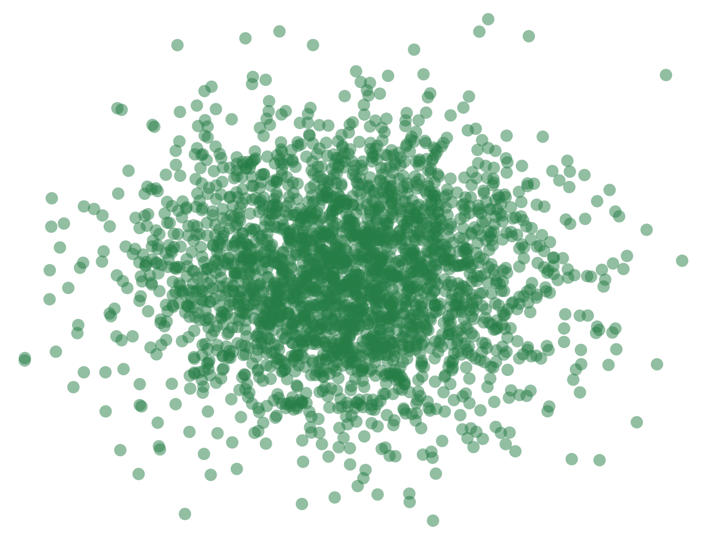
$$p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}}) = p_{\text{latent}}(z) \left| \det \frac{\partial g_\theta(x_{\text{h}}, x_{\text{r}})}{\partial x_{\text{hard}}} \right| = p_{\text{lat.}}(z) \quad \left| \det J_{g_\theta} \right|$$

Conditional INN (cINN)



$g_\theta(x_\mathrm{hard}\,|\,x_\mathrm{reco})$

$g_\theta^{-1}(z\,|\,x_\mathrm{reco})$



$x_0 \sim p_\mathrm{model}(x_\mathrm{hard}\,|\,x_\mathrm{reco})$

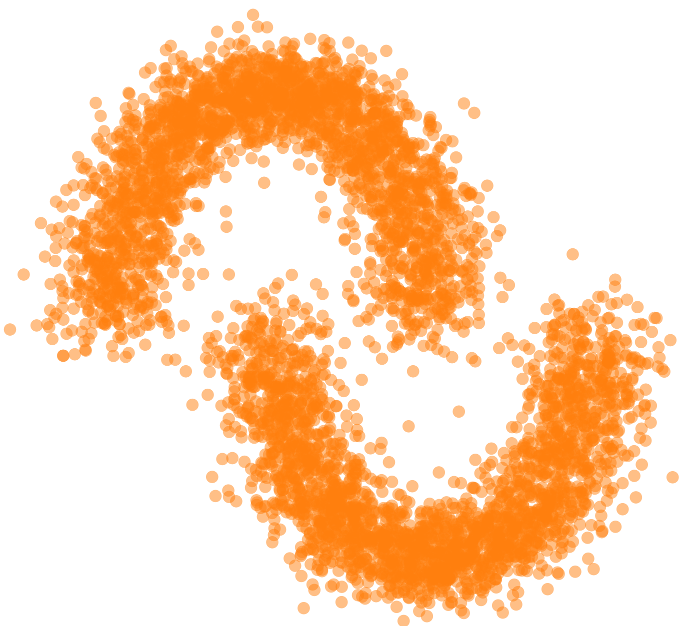
$z \sim p_\mathrm{latent}(z)$

- Bijective function between $p_\mathrm{latent}(z)$ and $p_\mathrm{model}(x_\mathrm{hard}\,|\,x_\mathrm{reco})$:

$$p_\mathrm{model}(x_\mathrm{hard}\,|\,x_\mathrm{reco}) = p_\mathrm{latent}(z)\left|\det \frac{\partial g_\theta(x_\mathrm{h}, x_\mathrm{r})}{\partial x_\mathrm{hard}}\right| = p_\mathrm{lat.}(z)\ \left|\det J_{g_\theta}\right|$$

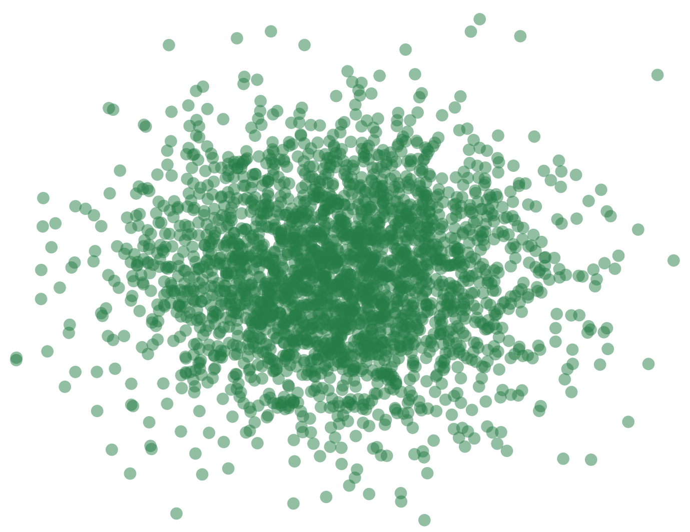
- Pairs $(x_\mathrm{hard},\ x_\mathrm{reco})$ are passed through the NN to the latent space: $z = g_\theta(x_\mathrm{hard}\,|\,x_\mathrm{reco})$

Conditional INN (cINN)



$$g_\theta(x_{\text{hard}}\,|\,x_{\text{reco}})$$

$$g_\theta^{-1}(z\,|\,x_{\text{reco}})$$



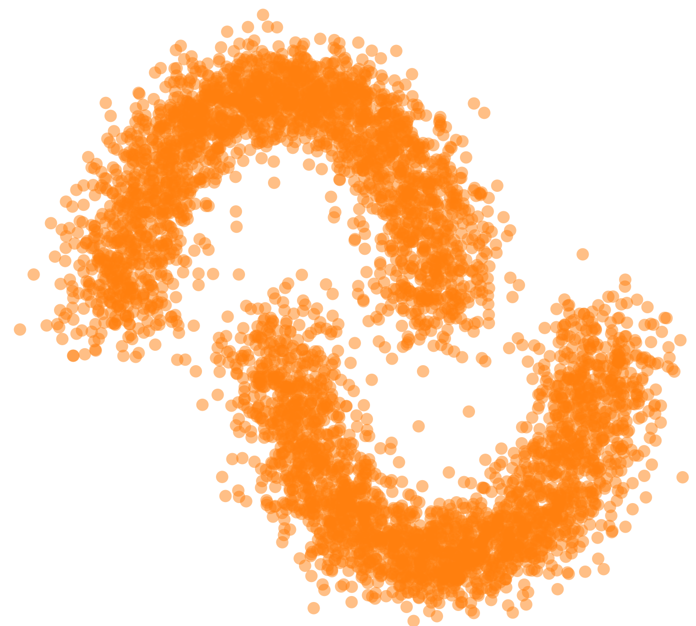
$x_0 \sim p_{\text{model}}(x_{\text{hard}}\,|\,x_{\text{reco}})$ $\qquad$ $z \sim p_{\text{latent}}(z)$

- Bijective function between $p_{\text{latent}}(z)$ and $p_{\text{model}}(x_{\text{hard}}\,|\,x_{\text{reco}})$:

$$p_{\text{model}}(x_{\text{hard}}\,|\,x_{\text{reco}}) = p_{\text{latent}}(z)\left|\det\frac{\partial g_\theta(x_{\text{h}}, x_{\text{r}})}{\partial x_{\text{hard}}}\right| = p_{\text{lat.}}(z)\ \left|\det J_{g_\theta}\right|$$

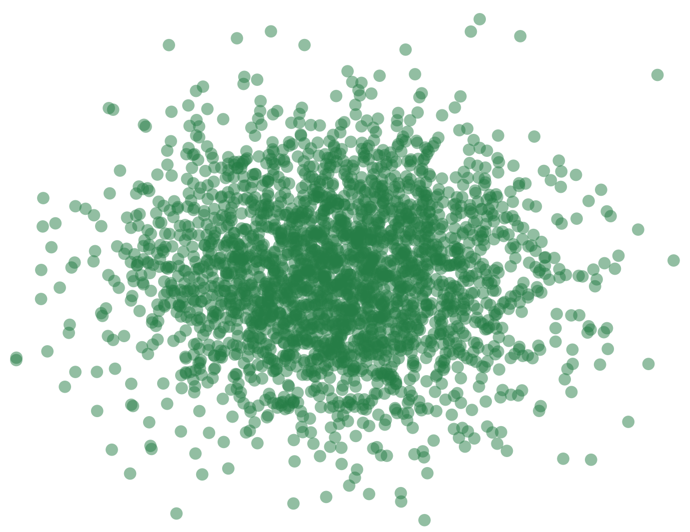
- Pairs $(x_{\text{hard}},\ x_{\text{reco}})$ are passed through the NN to the latent space: $\quad z = g_\theta(x_{\text{hard}}\,|\,x_{\text{reco}})$
- Once trained, one can sample -conditioned on reco- from the latent: $\quad p_{\text{hard}}(x) \approx p_{\text{model}}(x_{\text{hard}}\,|\,x_{\text{reco}})$

Conditional INN (cINN)



$g_\theta(x_{\text{hard}} \,|\, x_{\text{reco}})$

$g_\theta^{-1}(z \,|\, x_{\text{reco}})$



$x_0 \sim p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}})$ $\qquad$ $z \sim p_{\text{latent}}(z)$

- Bijective function between $p_{\text{latent}}(z)$ and $p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}})$:

$$p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}}) = p_{\text{latent}}(z) \left| \det \frac{\partial g_\theta(x_{\text{h}}, x_{\text{r}})}{\partial x_{\text{hard}}} \right| = p_{\text{lat.}}(z) \; \left| \det J_{g_\theta} \right|$$

- Pairs $(x_{\text{hard}},\, x_{\text{reco}})$ are passed through the NN to the latent space: $\quad z = g_\theta(x_{\text{hard}} \,|\, x_{\text{reco}})$
- Once trained, one can sample -conditioned on reco- from the latent: $\quad p_{\text{hard}}(x) \approx p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}})$
- Loss: $\quad \mathscr{L}_{\text{cINN}} = -\langle \log p_{\text{model}}(x_{\text{hard}} \,|\, x_{\text{reco}}) \rangle_{(x_0, x_1) \sim p(x_{\text{hard}}, x_{\text{reco}})}$

Z + jets events

$Z(p_T > 200\ \mathrm{GeV})$ + jets events generated at $\sqrt{s} = 14\ \mathrm{TeV}$ with Pythia 8.244 and Delphes simulation 3.5.0 available on Zenodo. Slight modification from [1911.09107] dataset

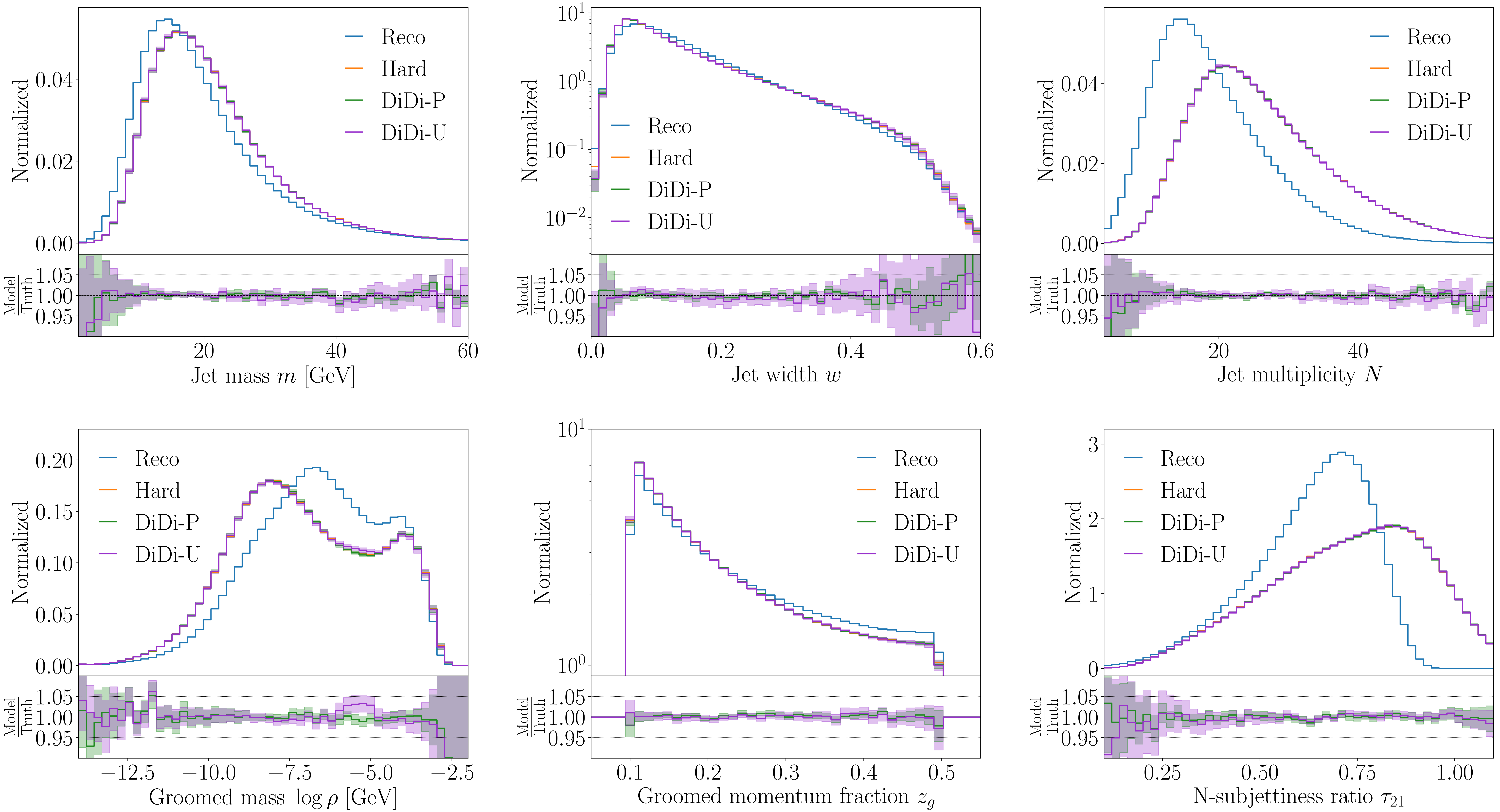
Z + jets events

$Z(p_T > 200\ \mathrm{GeV})$ + jets events generated at $\sqrt{s} = 14\ \mathrm{TeV}$ with Pythia 8.244 and Delphes simulation 3.5.0 available on Zenodo. Slight modification from [1911.09107] dataset

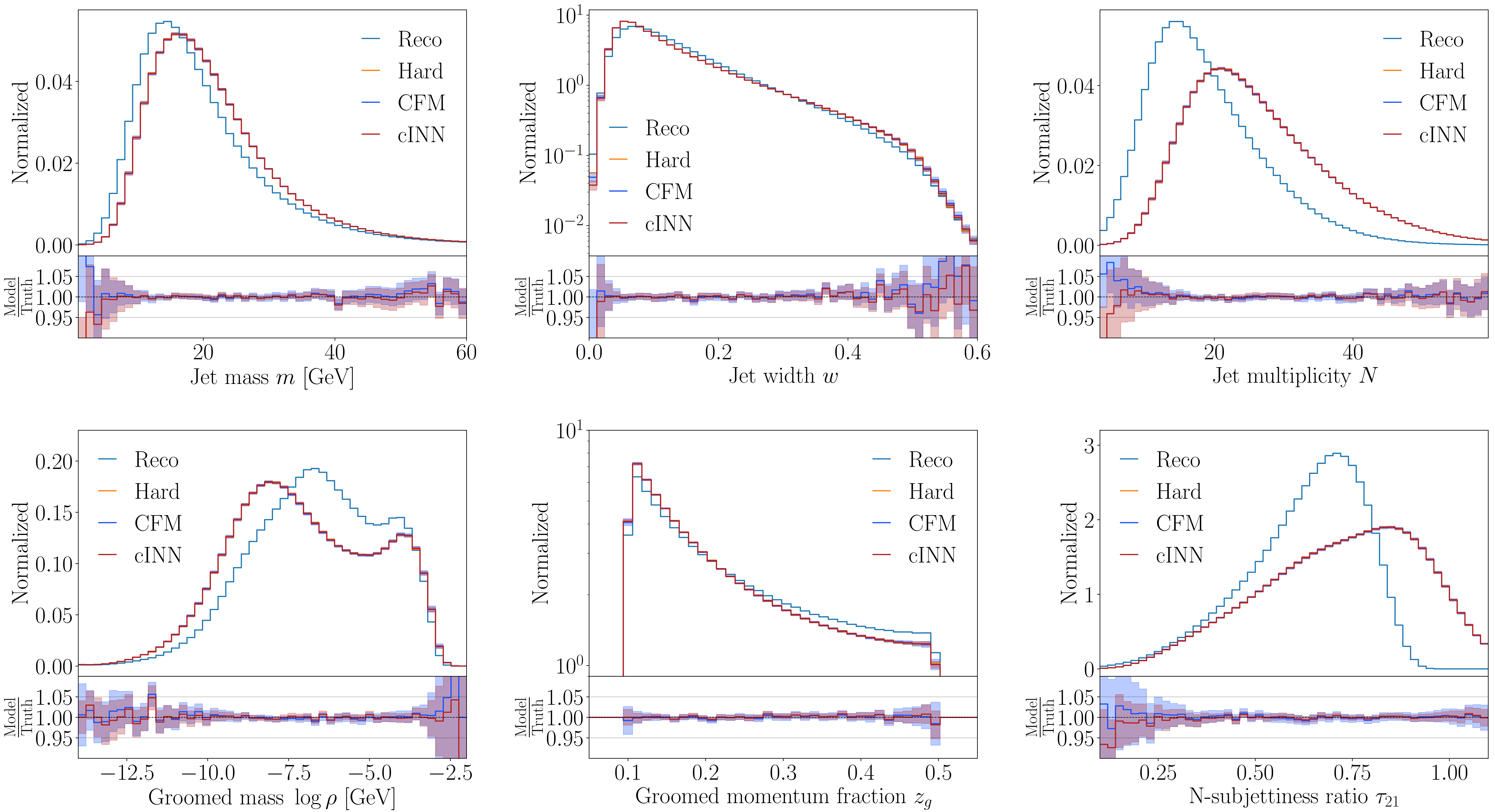
Six widely-used jet substructure observables:

- Jet mass m
- Jet width w
- Jet constituents multiplicity N
- Groomed mass $\log\rho = 2\log\left(m_{\mathrm{SD}}\ /\ p_T\right)$
- Groomed momentum fraction $z_g = \tau_1^{\beta=1}$
- N-subjettiness ratio $\tau_{21} = \tau_2^{\beta=1}/\tau_1^{\beta=1}$

Unfolded observables (DiDi)



Unfolded observables (CFM & cINN)



Top-pair events: unfolding to parton-level

Matrix elements are evaluated at $\sqrt{s} = 13\ \mathrm{TeV}$ using MadGraph_aMC@NLO. Showering and hadronization are simulated with Pythia8, and detector response is simulated with Delphes with the standard CMS card. For a detailed description see [2305.10399].

$$q\bar{q}/gg \to t\bar{t} \to (bl^+\nu_l)(\bar{b}qq)$$

Top-pair events: unfolding to parton-level

Matrix elements are evaluated at $\sqrt{s} = 13\ \mathrm{TeV}$ using MadGraph_aMC@NLO. Showering and hadronization are simulated with Pythia8, and detector response is simulated with Delphes with the standard CMS card. For a detailed description see [2305.10399].

$$q\bar{q}/gg \to t\bar{t} \to (bl^+\nu_l)(\bar{b}qq)$$

Unfolding from 6 final-state particles $(bl\nu)(bqq)$:

- 4 DoFs for the lepton
- 3 DoFs for the missing $\vec{p}^{\nu}$
- 5 DoFs per jet (4-momentum + b-tag)

27 DoFs at reco-level

Top-pair events: unfolding to parton-level

Matrix elements are evaluated at $\sqrt{s} = 13\ \mathrm{TeV}$ using MadGraph_aMC@NLO. Showering and hadronization are simulated with Pythia8, and detector response is simulated with Delphes with the standard CMS card. For a detailed description see [2305.10399].

$$q\bar{q}/gg \to t\bar{t} \to (bl^+\nu_l)(\bar{b}qq)$$

Unfolding from 6 final-state particles $(bl\nu)(bqq)$:

- 4 DoFs for the lepton
- 3 DoFs for the missing $\vec{p}^{\nu}$
- 5 DoFs per jet (4-momentum + b-tag)

27 DoFs at reco-level

- $(p_{T,b_l}, \eta_{b_l}, \phi_{b_l}, p_{T,l}, \eta_l, \phi_l, p_{T,\nu}, \eta_\nu, \phi_\nu, p_{T,b_h}, \eta_{b_h}, \phi_{b_h}, m_{q_1}, p_{T,q_1}, \eta_{q_1}, \phi_{q_1}, p_{T,q_2}, \eta_{q_2}, \phi_{q_2})$

19 DoFs at parton-level

Top-pair events: unfolding to parton-level

Much harder problem:

- Unfolding to parton-level means inverting the entire forward simulation chain

Top-pair events: unfolding to parton-level

Much harder problem:

- Unfolding to parton-level means inverting the entire forward simulation chain
- Faithful modeling of complex correlations at parton-level, i.e., W boson and top mass

Top-pair events: unfolding to parton-level

Much harder problem:

- Unfolding to parton-level means inverting the entire forward simulation chain
- Faithful modeling of complex correlations at parton-level, i.e., W boson and top mass
- Non-trivial combinatorics between physics objects at both levels

Top-pair events: unfolding to parton-level

Much harder problem:

- Unfolding to parton-level means inverting the entire forward simulation chain
- Faithful modeling of complex correlations at parton-level, i.e., W boson and top mass
- Non-trivial combinatorics between physics objects at both levels

Adding transformers:

Top-pair events: unfolding to parton-level

Much harder problem:

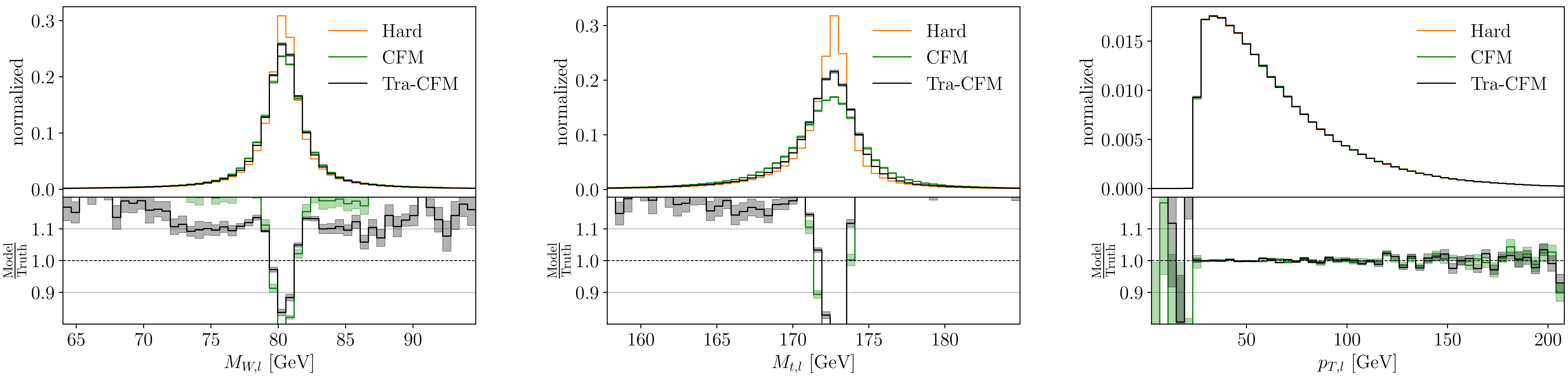
- Unfolding to parton-level means inverting the entire forward simulation chain
- Faithful modeling of complex correlations at parton-level, i.e., W boson and top mass
- Non-trivial combinatorics between physics objects at both levels

Adding transformers:

- Tra-CFM as an extension to CFM [2310.07752]. A transformer is employed to encode correlations at reco and parton-level.

Results: naive parametrization

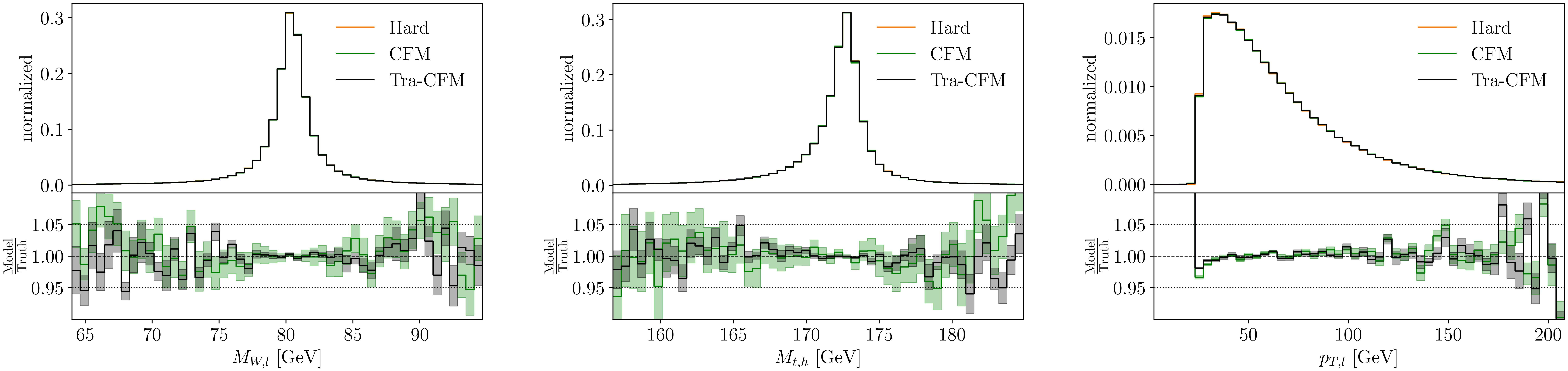
- Unfold: $(p_{T,b_l}, \eta_{b_l}, \phi_{b_l}, p_{T,l}, \eta_l, \phi_l, p_{T,\nu}, \eta_\nu, \phi_\nu, p_{T,b_h}, \eta_{b_h}, \phi_{b_h}, m_{q_1}, p_{T,q_1}, \eta_{q_1}, \phi_{q_1}, p_{T,q_2}, \eta_{q_2}, \phi_{q_2})$



Results: mass parametrization

- Unfold: $(m_t, p_{T,t}^L, \eta_t^L, \phi_t^L, m_W, \eta_W^T, \phi_W^T, (m_{d_1}^W), \eta_{d_1}^W, \phi_{d_1}^W)$

  Originally introduced in [2308.00027]



Summary and outlook

- ML-based unfolding is an unbinned transformative analysis tool

Summary and outlook

- ML-based unfolding is an unbinned transformative analysis tool
- Distribution mapping is relatively fast to train can be used on matched and unmatched data

Summary and outlook

- ML-based unfolding is an unbinned transformative analysis tool
- Distribution mapping is relatively fast to train can be used on matched and unmatched data
- CFM and cINN are best suited to describe complex detector effects, but they are also more complex architectures to train

Summary and outlook

- ML-based unfolding is an unbinned transformative analysis tool
- Distribution mapping is relatively fast to train can be used on matched and unmatched data
- CFM and cINN are best suited to describe complex detector effects, but they are also more complex architectures to train
- Parton-level unfolding is a reasonably complicated task, but transformers and adequate parametrization help greatly in accounting for correlations and resonances

Summary and outlook

- ML-based unfolding is an unbinned transformative analysis tool
- Distribution mapping is relatively fast to train can be used on matched and unmatched data
- CFM and cINN are best suited to describe complex detector effects, but they are also more complex architectures to train
- Parton-level unfolding is a reasonably complicated task, but transformers and adequate parametrization help greatly in accounting for correlations and resonances
- All the tools within percent level precision across many observables

Summary and outlook

- ML-based unfolding is an unbinned transformative analysis tool
- Distribution mapping is relatively fast to train can be used on matched and unmatched data
- CFM and cINN are best suited to describe complex detector effects, but they are also more complex architectures to train
- Parton-level unfolding is a reasonably complicated task, but transformers and adequate parametrization help greatly in accounting for correlations and resonances
- All the tools within percent level precision across many observables

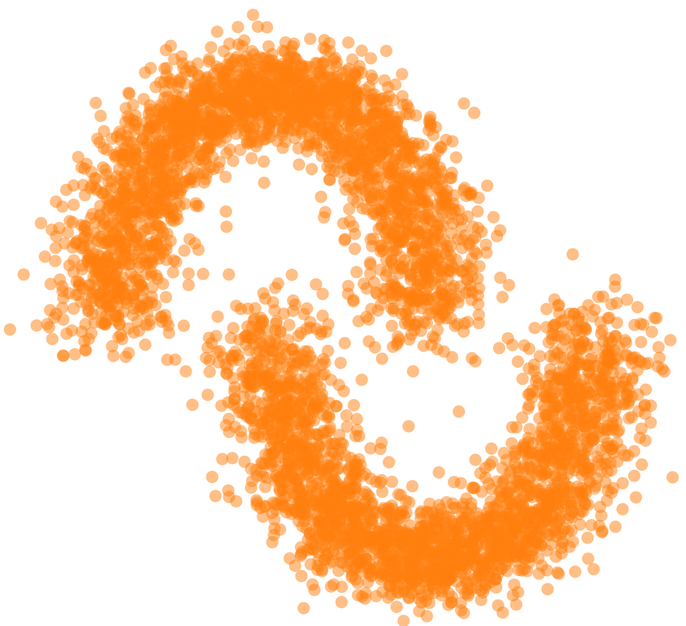
What is next?

- Compare to other methods, single-event unfolding, model dependence, and more…

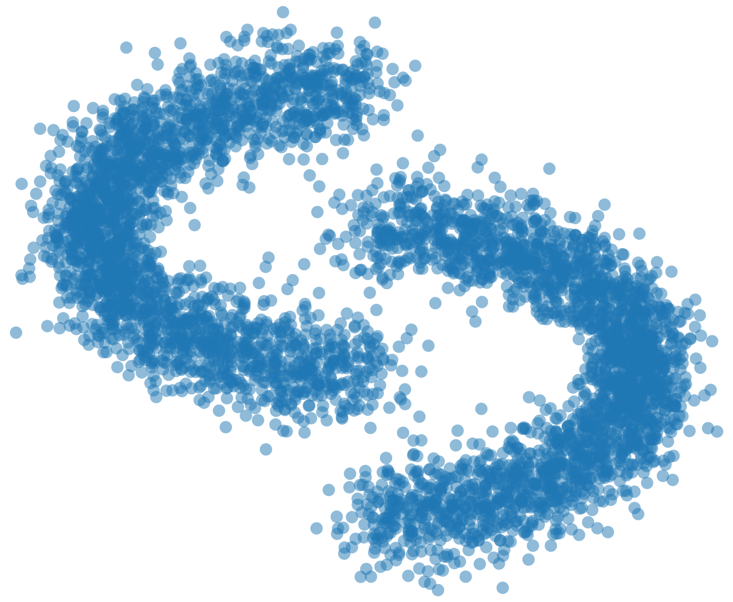
Thanks for your attention!

Backup

Direct Diffusion (DiDi)



$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_\theta(x(t), t)$$



$x_0 \sim p_{\mathrm{model}}(x_{\mathrm{hard}})$ $\qquad$ $x_1 \sim p_{\mathrm{reco}}(x_{\mathrm{reco}})$

- Connect x_0 and x_1 with a linear trajectory: $\quad x(t) = (1-t)x_0 + tx_1$
- The NN is regressed to predict the velocity field: $\quad v_\theta(x(t), t) \approx \frac{\mathrm{dx(t)}}{\mathrm{dt}} = x_1 - x_0$
- For sampling, solve ODE starting from x_1: $\quad x_0 = x_1 + \int_1^0 v_\theta(x(t), t)\mathrm{dt}$
- Loss:

$$\mathscr{L}_{\mathrm{DiDi-P}} = \left\langle [v_\theta((1-t)x_0 + tx_1,\ t) - (x_1 - x_0)]^2 \right\rangle_{t\sim\mathscr{U}([0,1]),\ (x_0,x_1)\sim p(x_{\mathrm{hard}},x_{\mathrm{reco}})}$$

$$\mathscr{L}_{\mathrm{DiDi-U}} = \left\langle [v_\theta((1-t)x_0 + tx_1,\ t) - (x_1 - x_0)]^2 \right\rangle_{t\sim\mathscr{U}([0,1]),\ x_0\sim p(x_{\mathrm{hard}}),x_1\sim p(x_{\mathrm{reco}})}$$

Z + jets events

$Z(p_T > 200\ \mathrm{GeV})$ + jets events generated at $\sqrt{s} = 14\ \mathrm{TeV}$ with Pythia 8.244 and Delphes simulation 3.5.0 available on Zenodo

Six widely-used jet substructure observables:

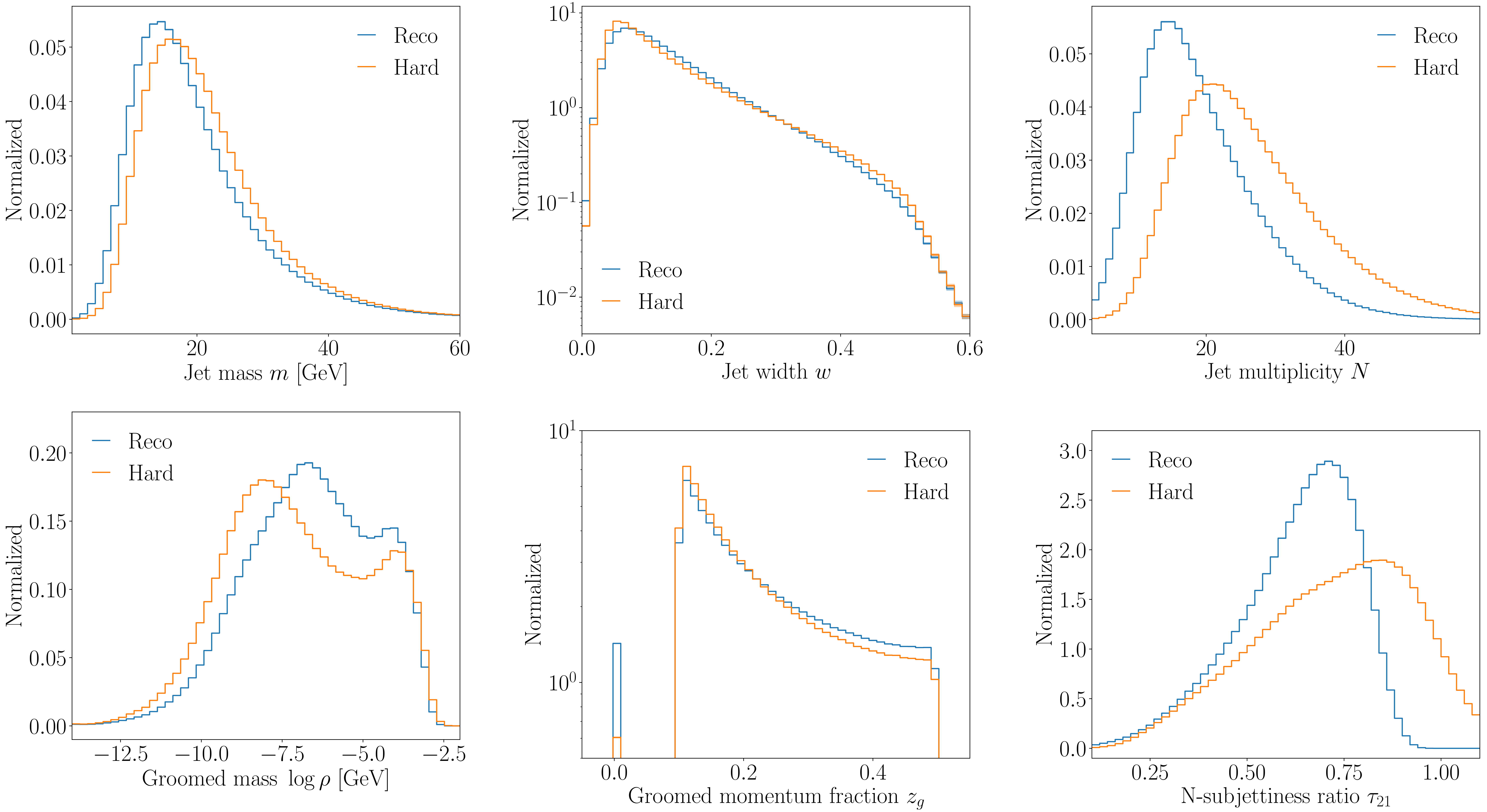
- Jet mass m
- Jet width w
- Jet constituents multiplicity N

Networks of ~3M parameters

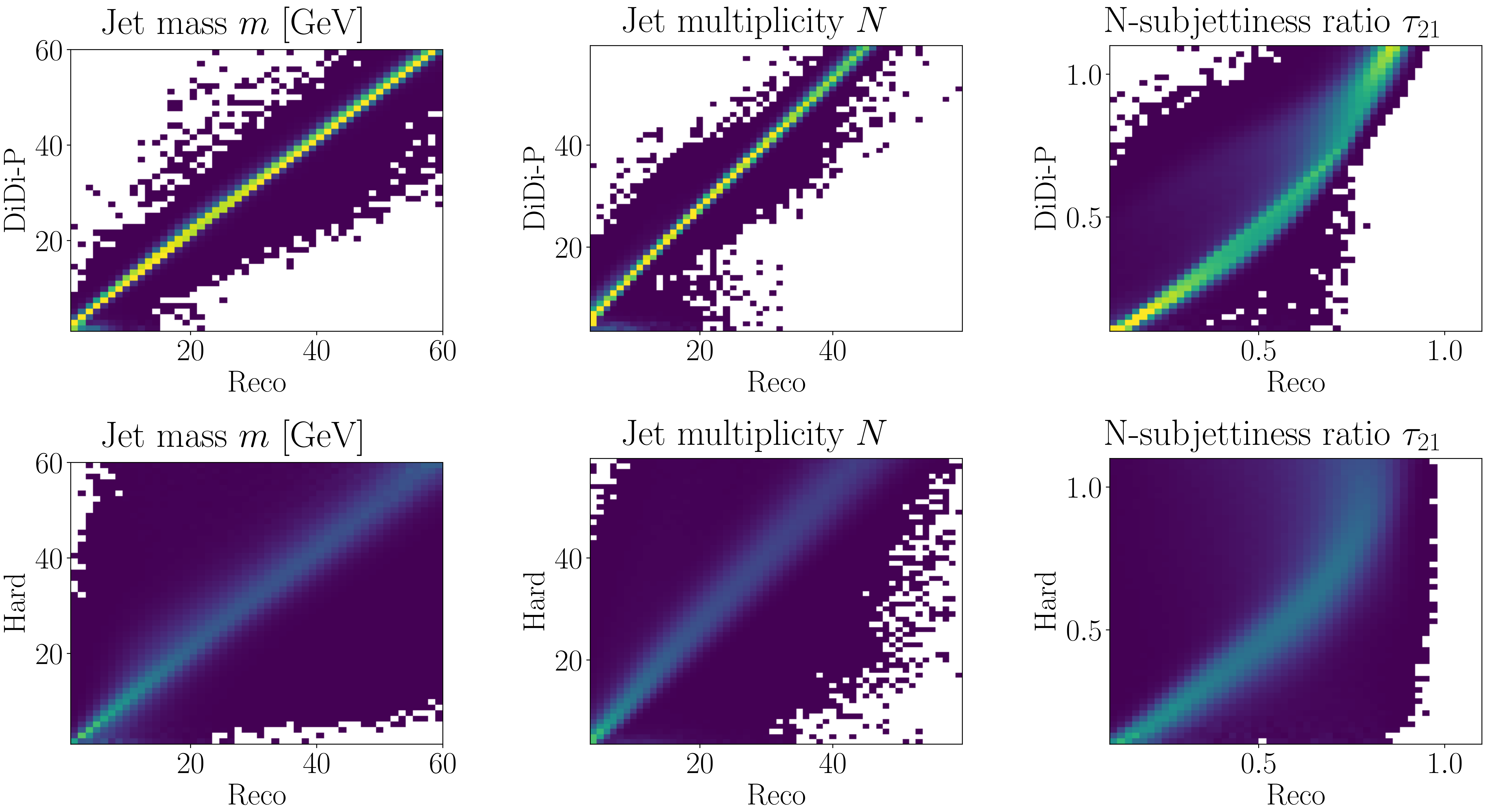
19M training events and 1M validation events

~4M events for testing

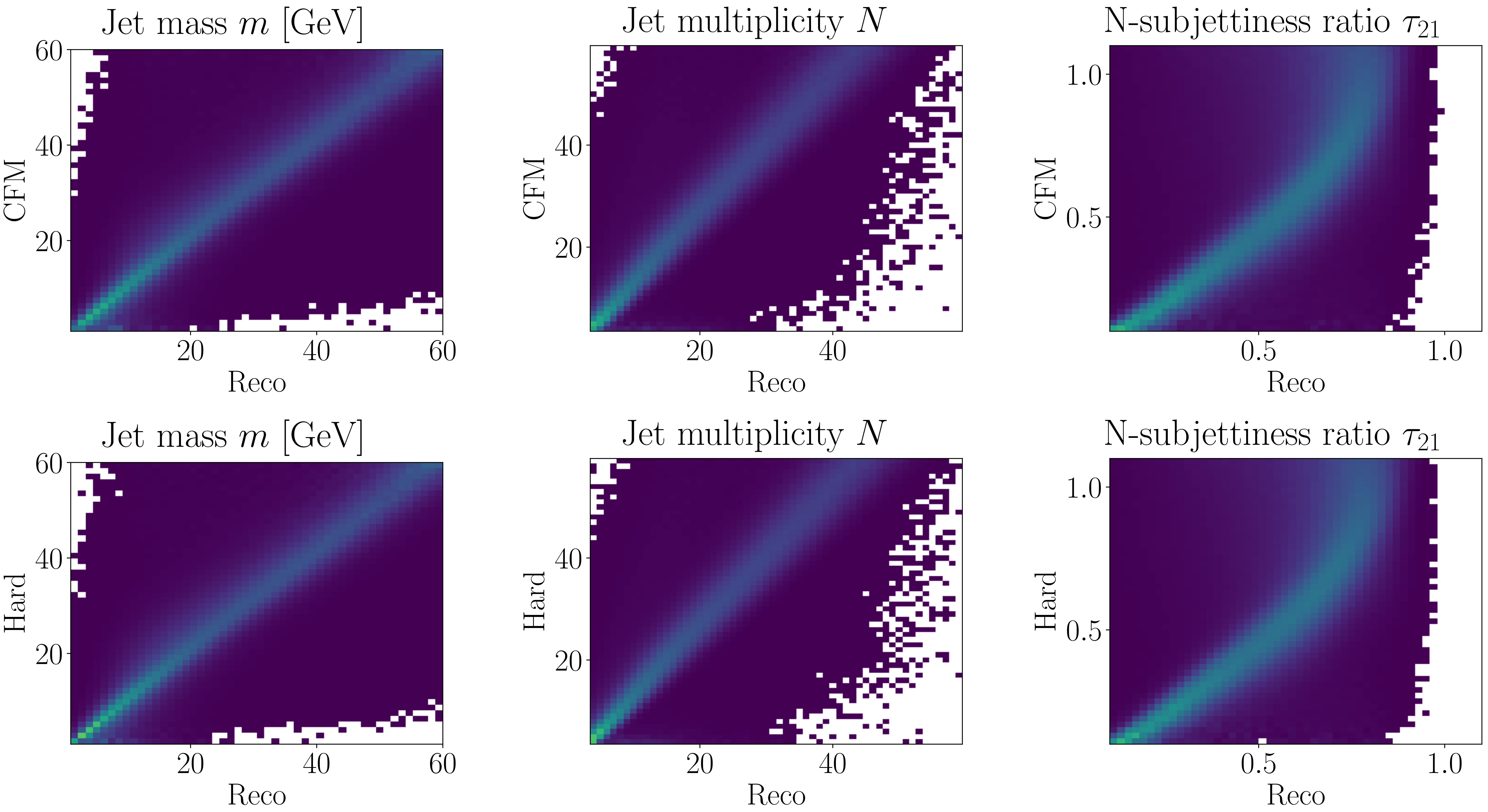
The dataset


Reco
Hard
Normalized
Jet mass m [GeV]
Jet width w
Jet multiplicity N
Groomed mass $\log \rho$ [GeV]
Groomed momentum fraction z_g
N-subjettiness ratio τ_{21}

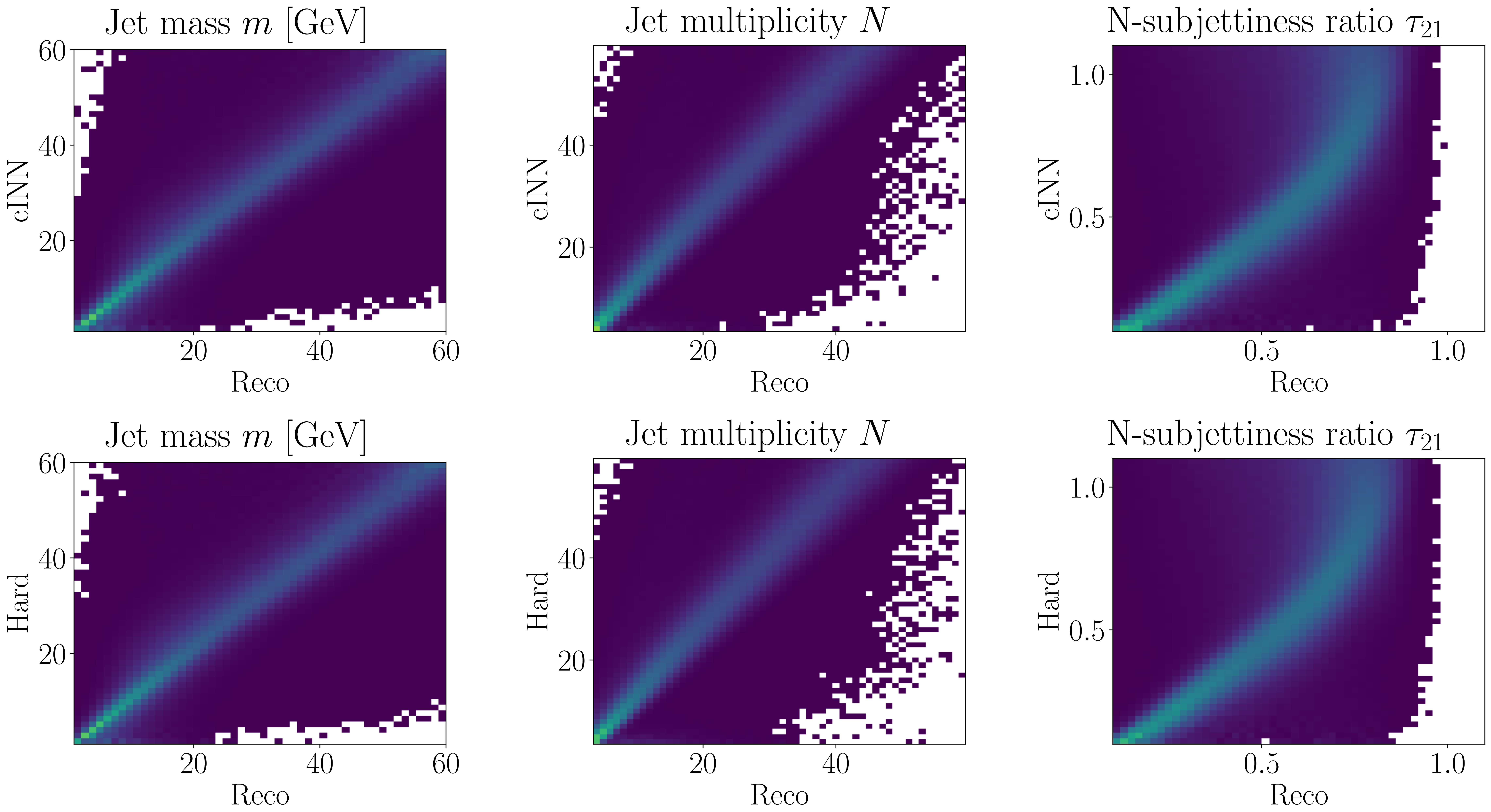
Optimal transport (DiDi)



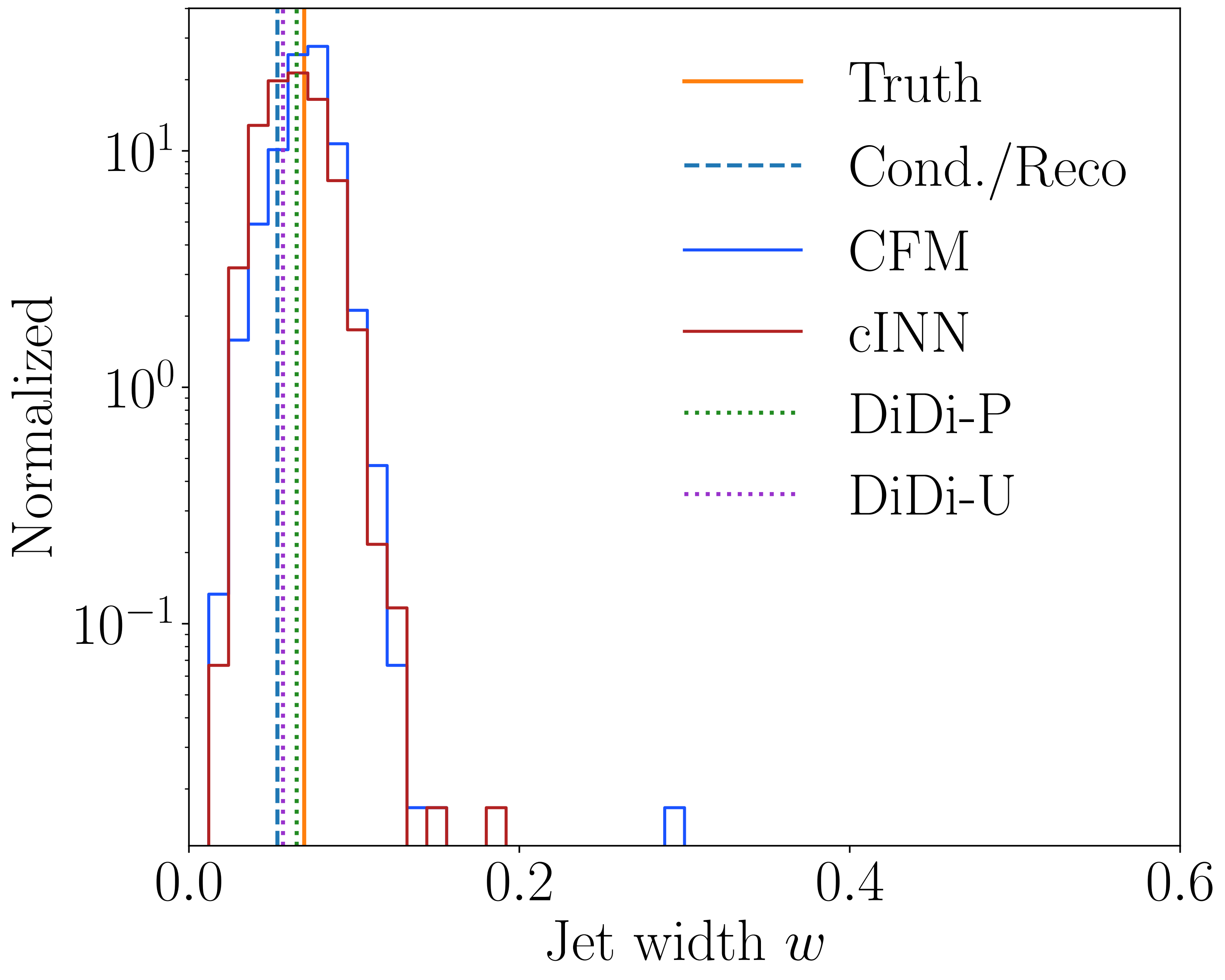
Optimal transport (CFM)

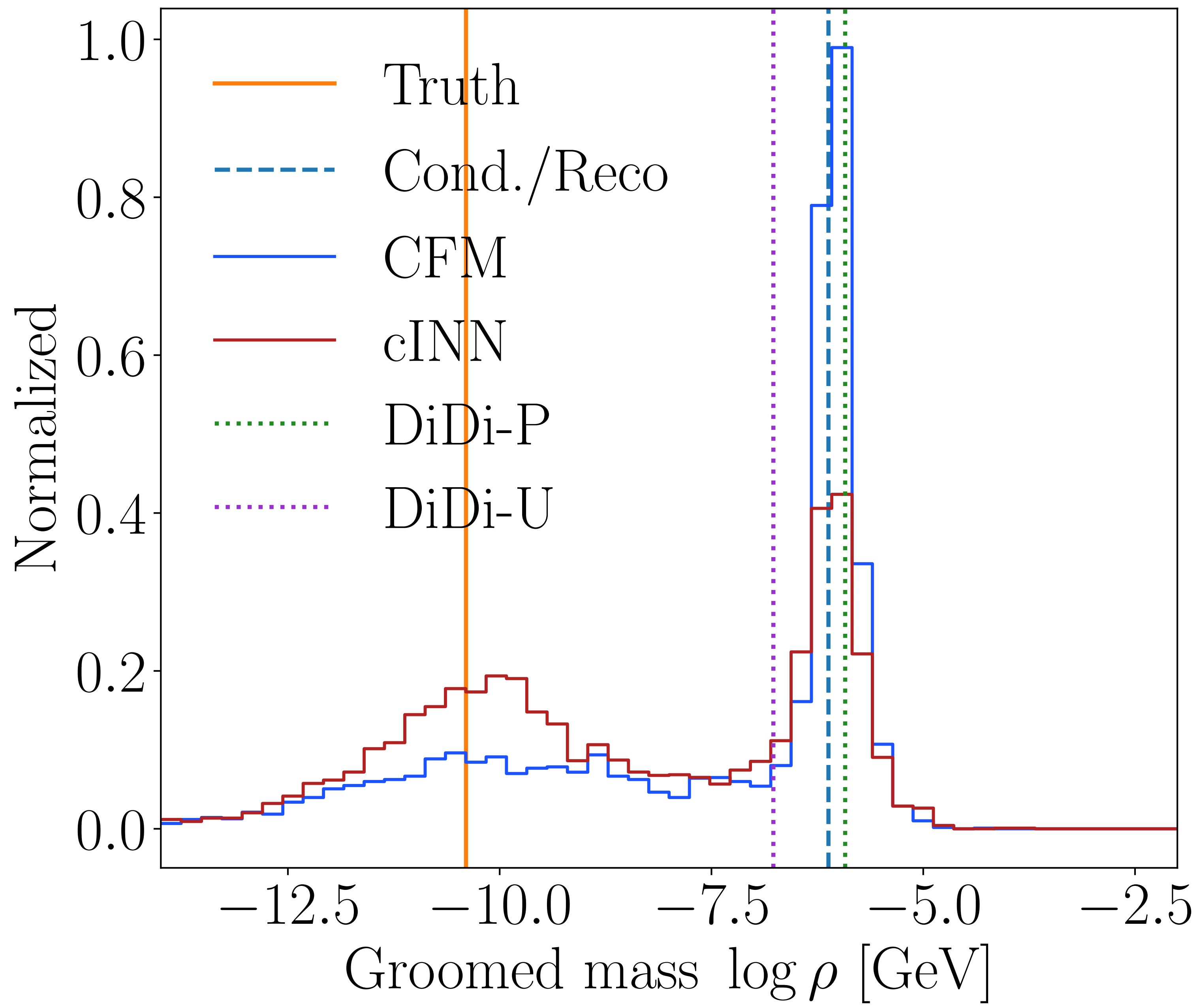


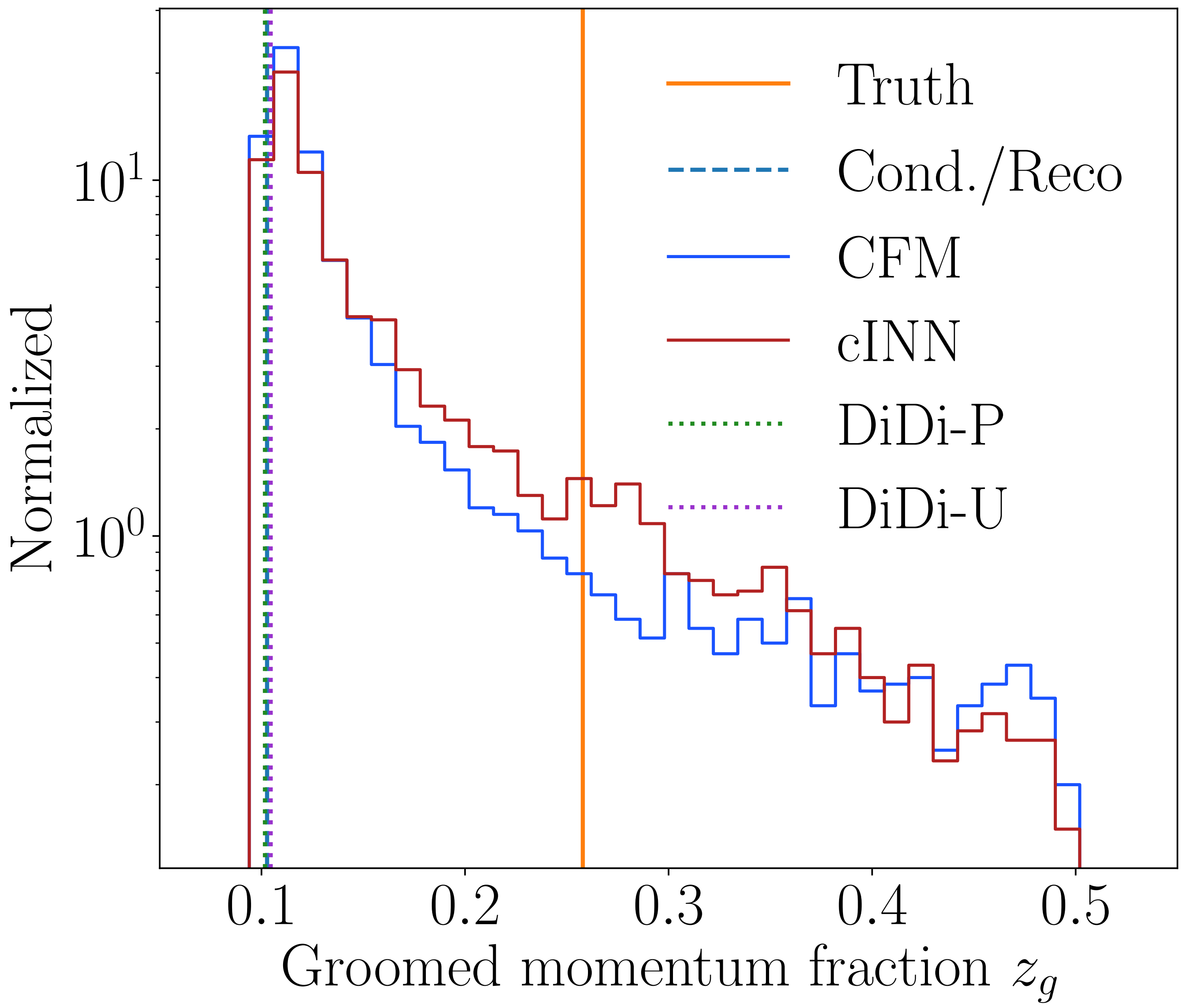
Optimal transport (cINN)



Single events

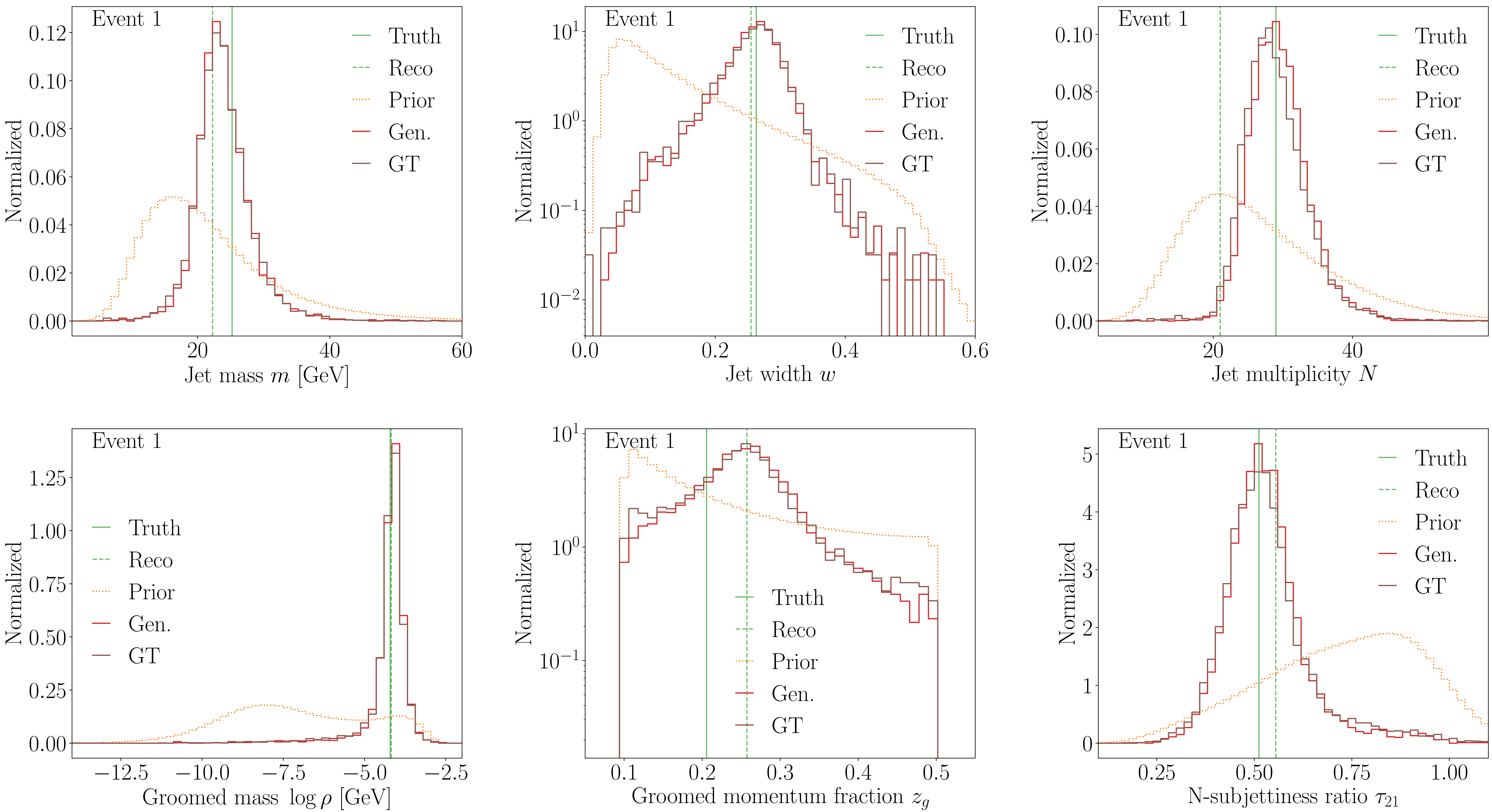




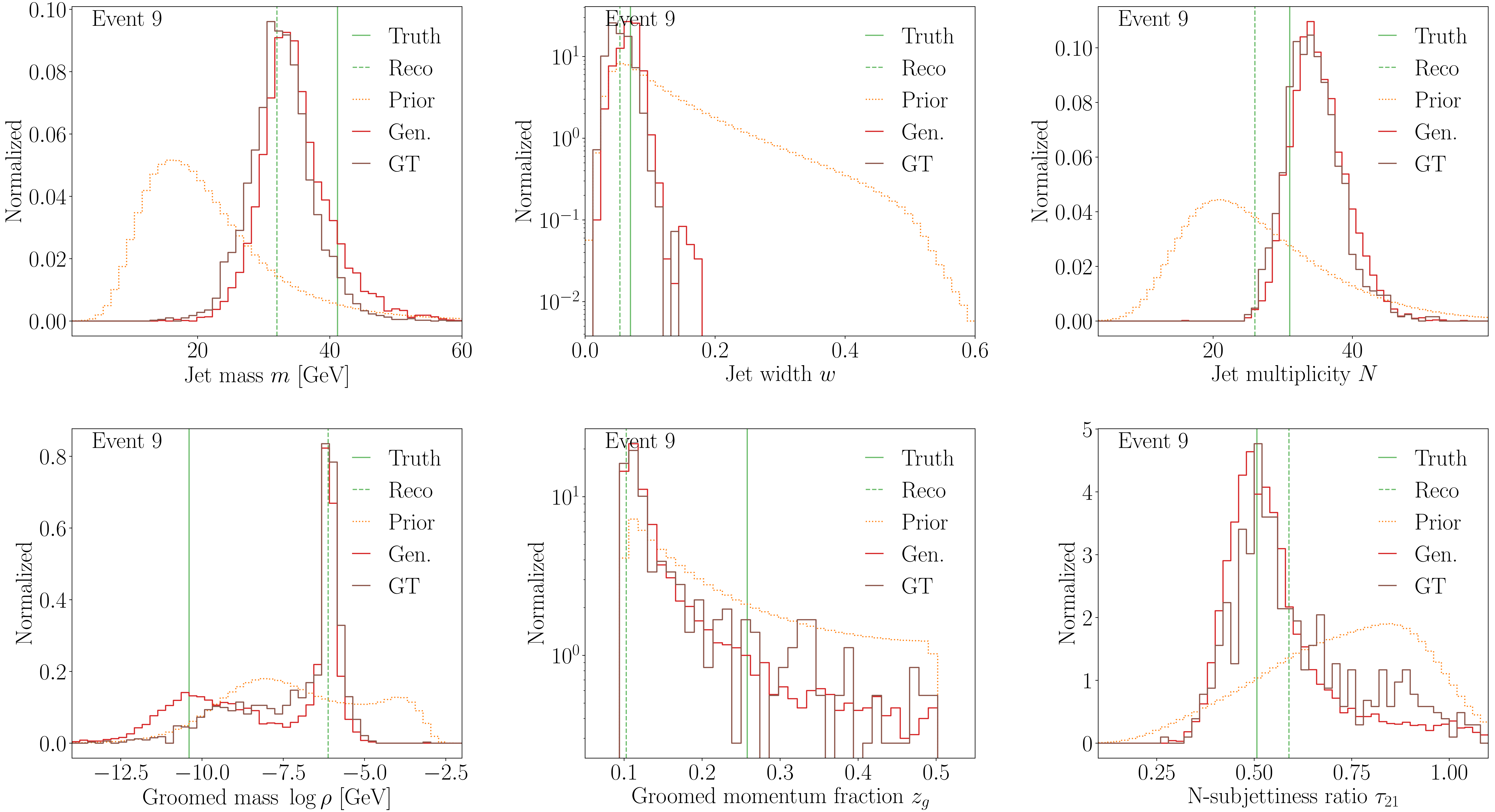


Single event unfolding

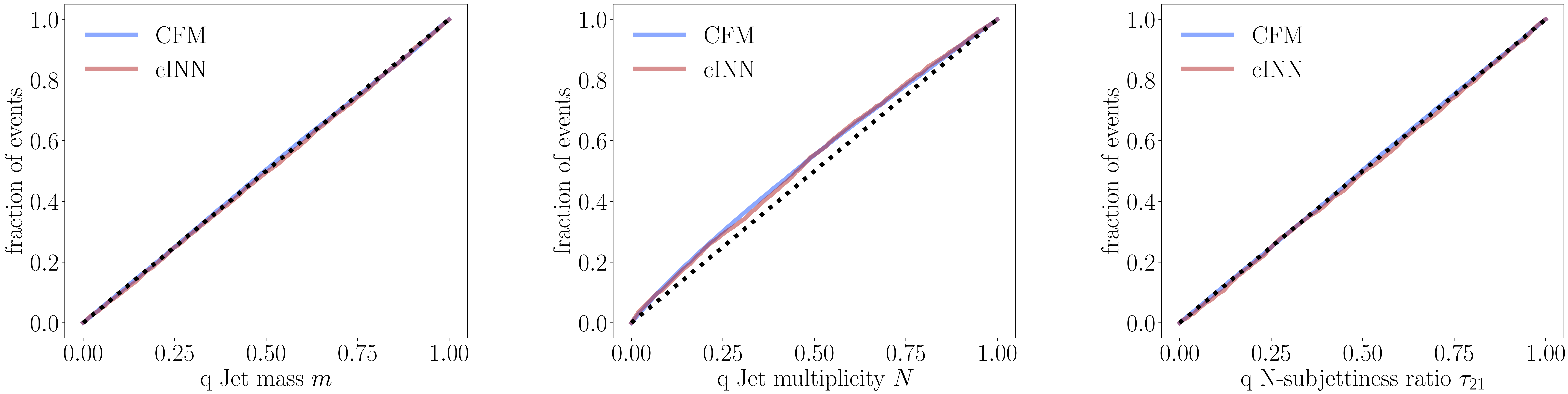
Single event unfolding



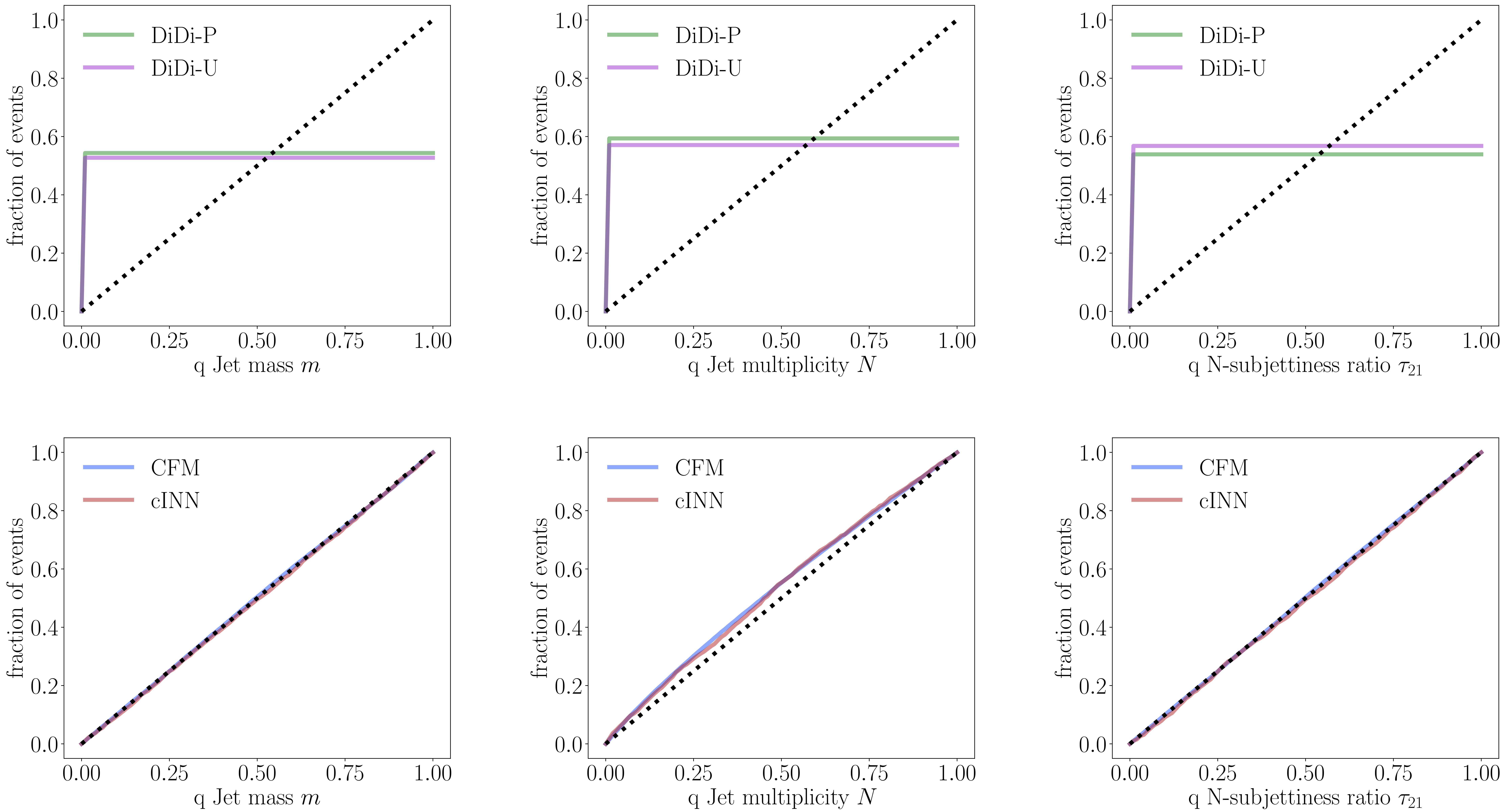
Single event unfolding



Calibration



Calibration



Top-pair events: unfolding to parton-level

Matrix elements are evaluated at $\sqrt{s} = 13\ \mathrm{TeV}$ using MadGraph_aMC@NLO. Showering and hadronization are simulated with Pythia8, and detector response is simulated with Delphes with the standard CMS card. For a detailed description see [2305.10399].

Unfolding from 6 final-state particles $(bl\nu)(bqq)$:

- 4 DoFs for the lepton
- 3 DoFs for the missing p_T^ν
- 5 DoFs per jet (4-momentum + b-tag)

Total: 27 DoFs at reco-level and 19 DoFs at parton-level

Non-bayesian networks

cINN ~ 8M parameters, CFM ~ 6M, Tra-CFM, Transfermer ~ 3M

10M training events and 1M testing events

Top-pair events: unfolding to parton-level

Adding transformers:

- For transfermer, likelihoods are factorized autoregressively on all previous parton-level dimensions and reco-level event:

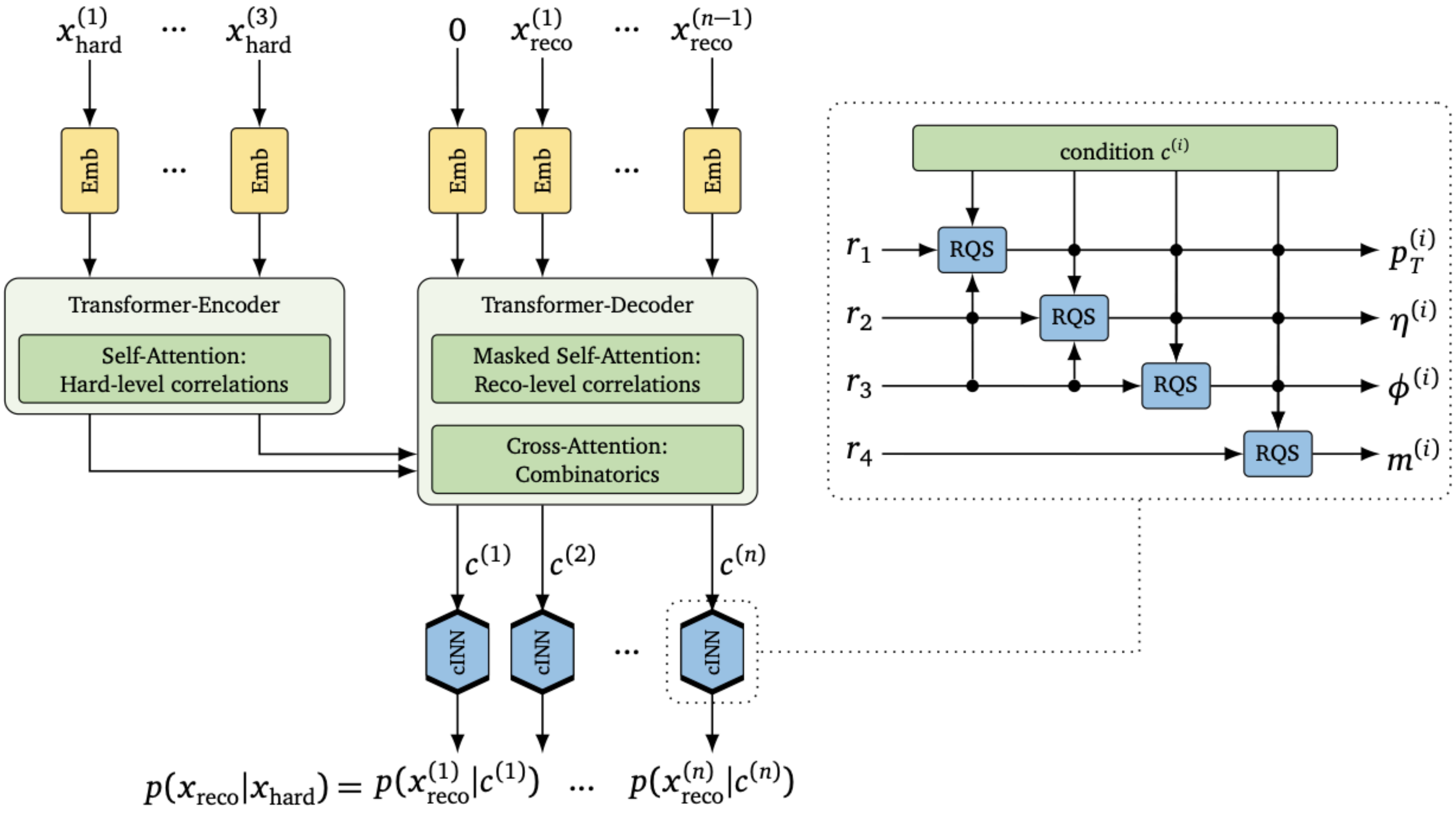
$$p_{\text{model}}(x_{\text{part}} \,|\, x_{\text{reco}}) = \prod_{i=1}^{n} p_{\text{model}}(x_{\text{part}}^{(i)} \mid c(x_{\text{part}}^{(0)}, \ldots, x_{\text{part}}^{(i-1)}, x_{\text{reco}}))$$

- For Tra-CFM, the transformer is made time-dependent and a small CFM predicts velocities at each different dimension:

$$v(x_{\text{part}}(t), t \,|\, x_{\text{reco}}) = \left(v^{(1)}(c^{(1)}, t), \ \ldots, v^{(n)}(c^{(n)}, t)\right)$$

Top-pair events: unfolding to parton-level

Transfermer



Top-pair events: unfolding to parton-level

Tra-CFM