UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

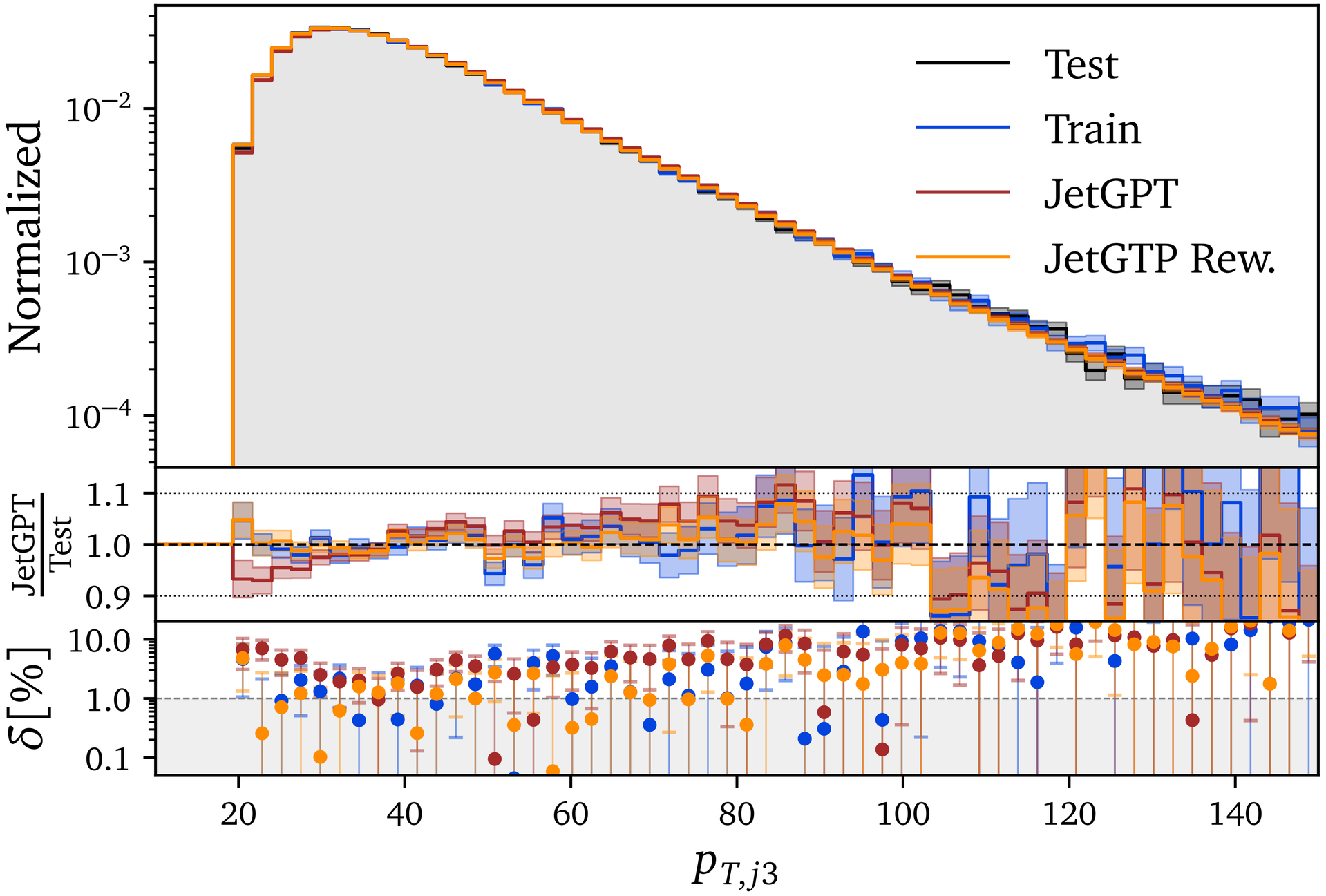
High Multiplicity with JetGPT

LHC Event Generation with Autoregressive Transformers

Jonas Spinner

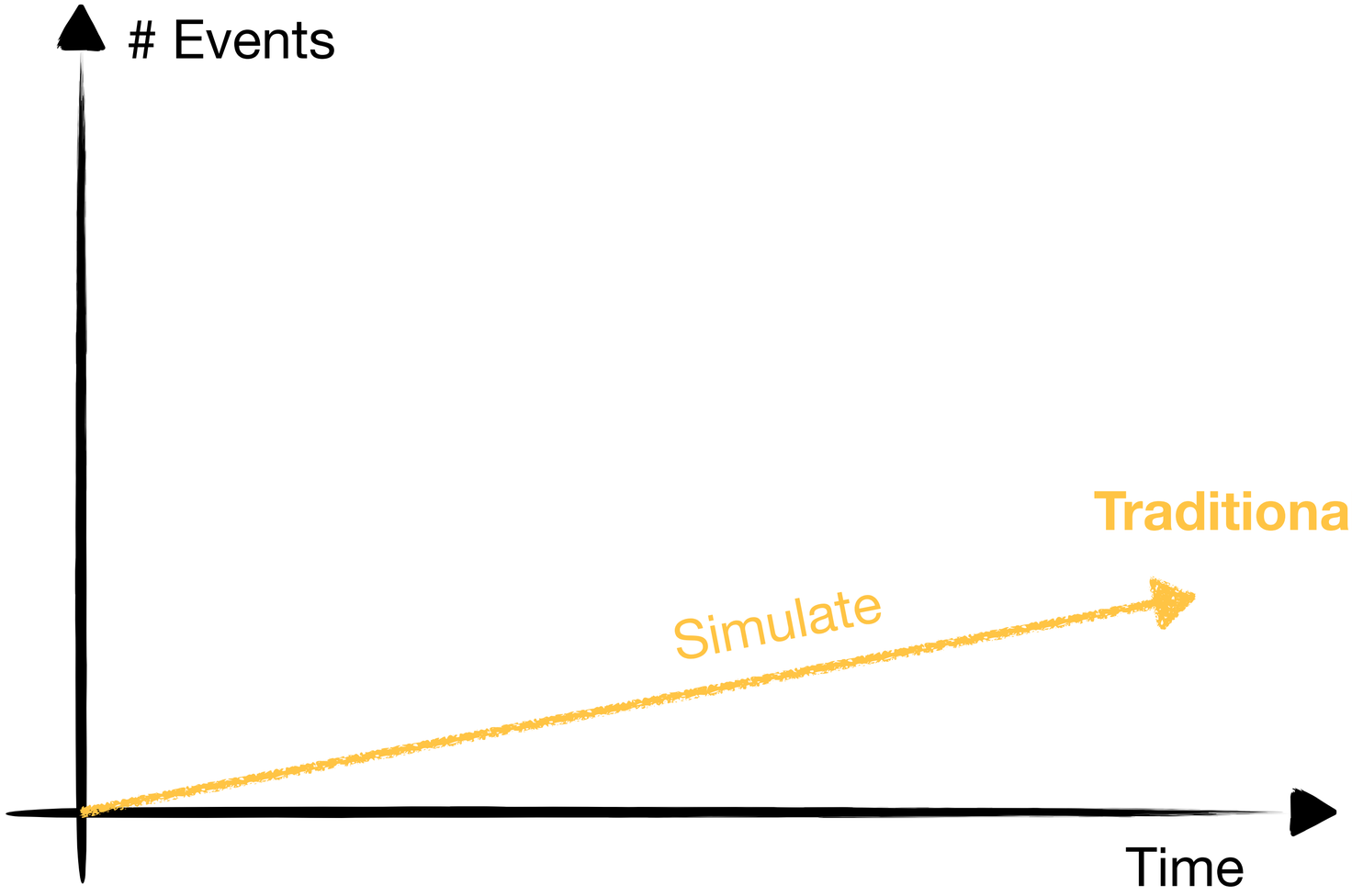
Based on work in collaboration with:
Anja Butter, Nathanael Ediger, Nathan Hütsch,
Maeve Madigan, Sofia Palacios and Tilman Plehn
2305.10475

ML4Jets 2023


Normalized
10^{-2}
10^{-3}
10^{-4}
Test
Train
JetGPT
JetGTP Rew.
$\frac{\text{JetGPT}}{\text{Test}}$
1.1
1.0
0.9
$\delta[\%]$
10.0
1.0
0.1
20
40
60
80
100
120
140
$p_{T,j3}$

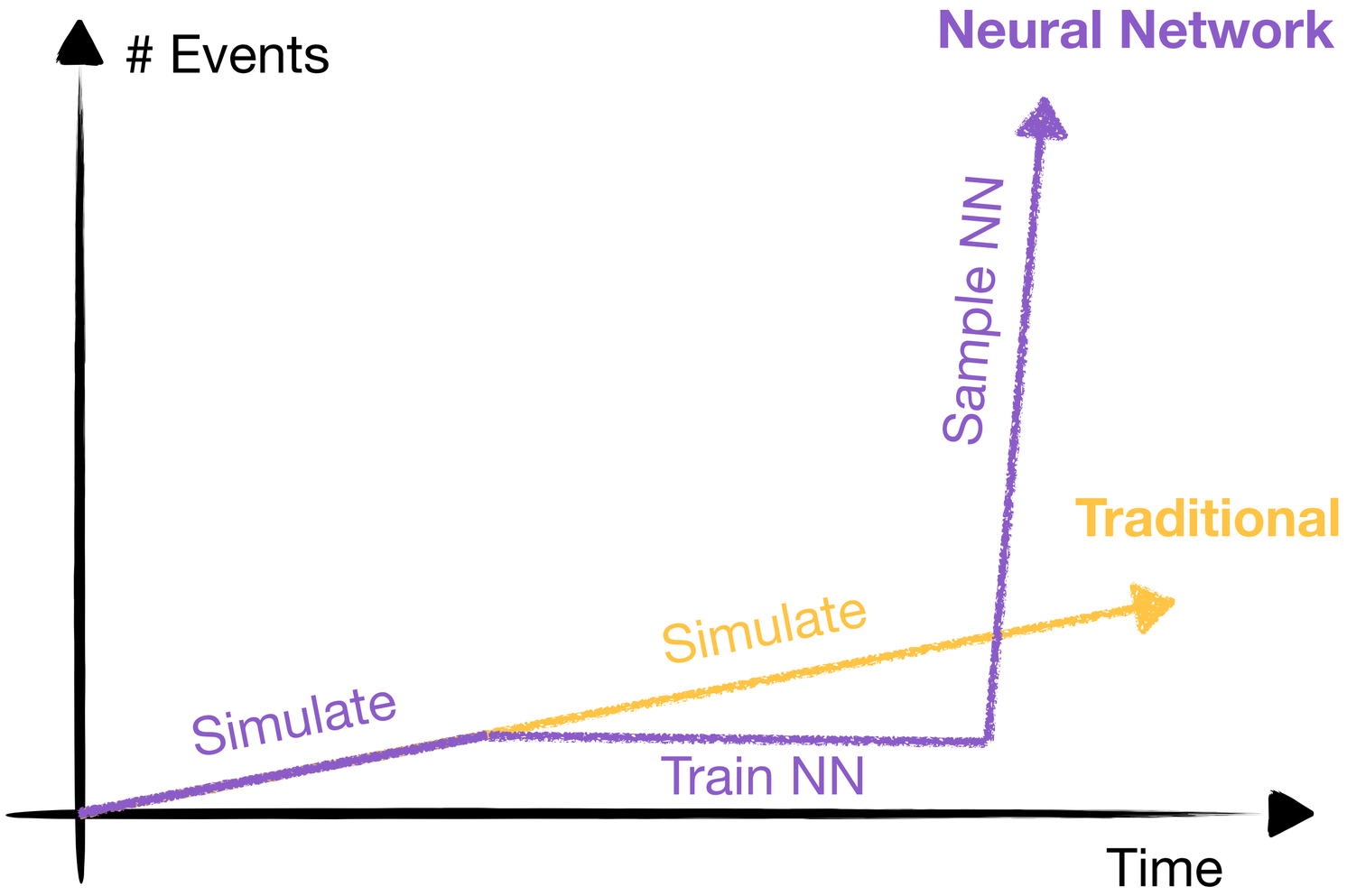
Motivation

End-to-End-Generation with Neural Networks



Motivation

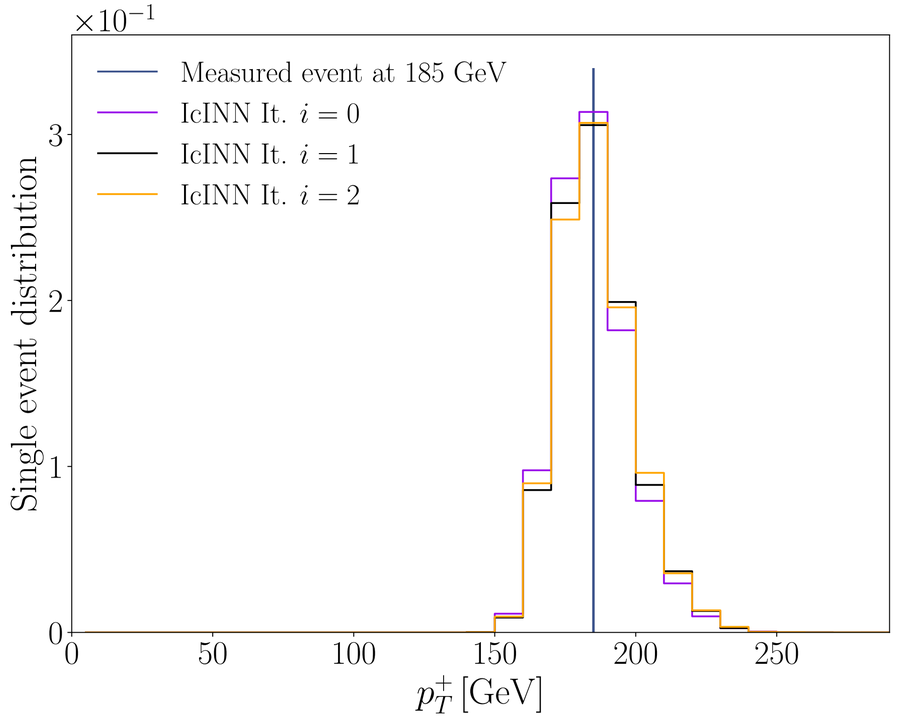
End-to-End-Generation with Neural Networks



- **Faster** when many events are required
- NNs are a more **efficient** encoding of distributions
- NNs **scale better** towards complex processes

Motivation

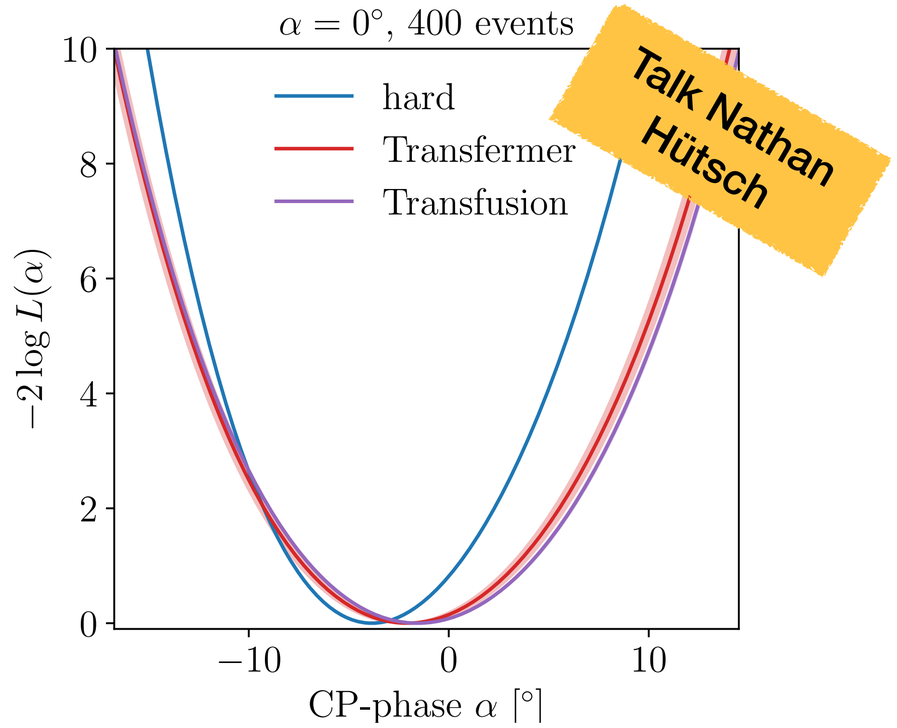
Inference with Generative Neural Networks



2006.06685 2305.10399

(Generative) Unfolding

1912.00477 2212.08674 2308.12351



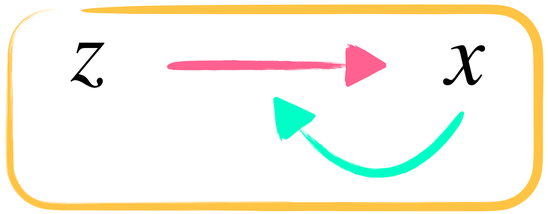
2210.00019 2310.07752

(Generative) Matrix Element Method

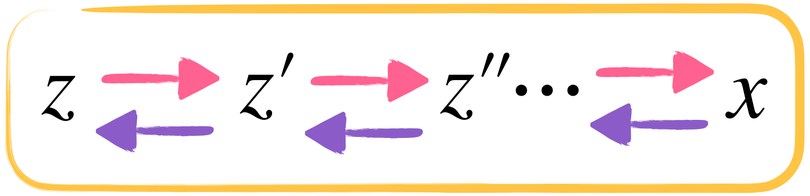
Motivation

Generative Neural Networks

GANs

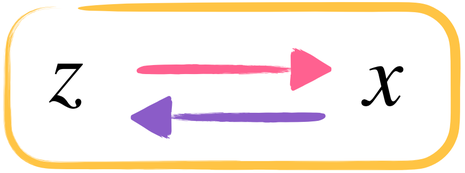


Diffusion Models



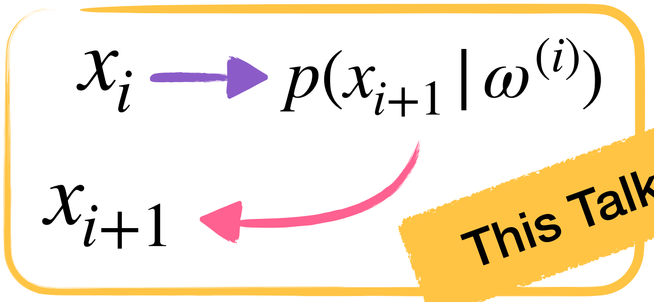
The ***Precise***

Normalizing Flows



The ***Fast***

Autoregressive Transformers



The ***Flexible***

x **Phase Space**

z **Latent Space**

Sampling

Density Estimation

Classifier

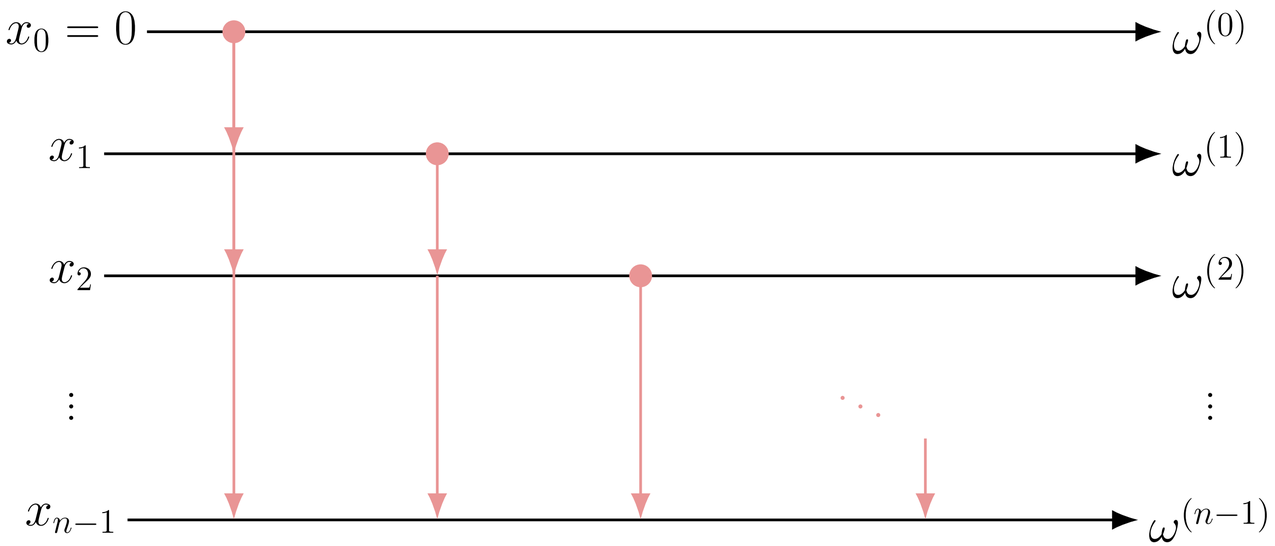

Autoregressive
Transformers

Autoregressive Transformer

Generating Events

Autoregression

$$
\begin{aligned}
p(x_1, x_2 \cdots x_n) &= p(x_1) && p(x_2 \mid x_1) && \cdots && p(x_n \mid x_1 \cdots x_{n-1}) \\
&= p(x_1 \mid \omega^{(0)}) && p(x_2 \mid \omega^{(1)}) && \cdots && p(x_n \mid \omega^{(n-1)})
\end{aligned}
$$

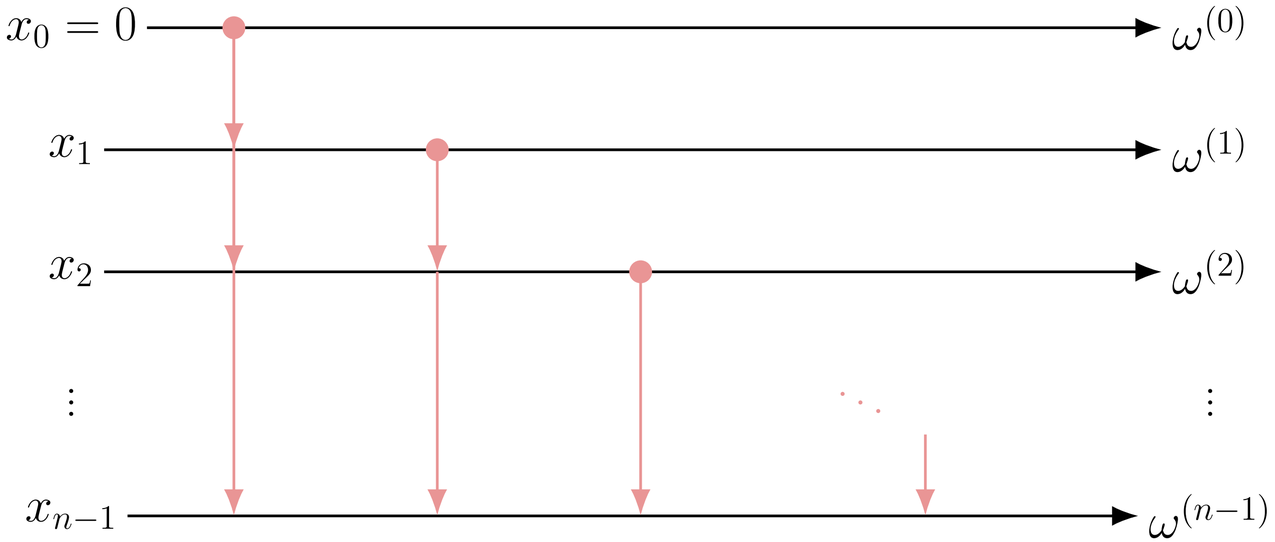


Autoregressive Transformer

Generating Events

Autoregression

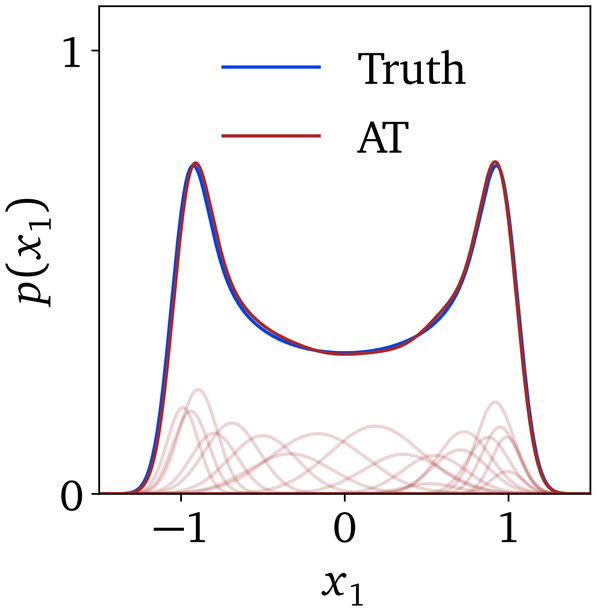
$$p(x_1, x_2 \cdots x_n) = p(x_1) \quad p(x_2 \,|\, x_1) \quad \cdots \quad p(x_n \,|\, x_1 \cdots x_{n-1})$$
$$= p(x_1 \,|\, \omega^{(0)}) \quad p(x_2 \,|\, \omega^{(1)}) \quad \cdots \quad p(x_n \,|\, \omega^{(n-1)})$$



Gaussian Mixture Model

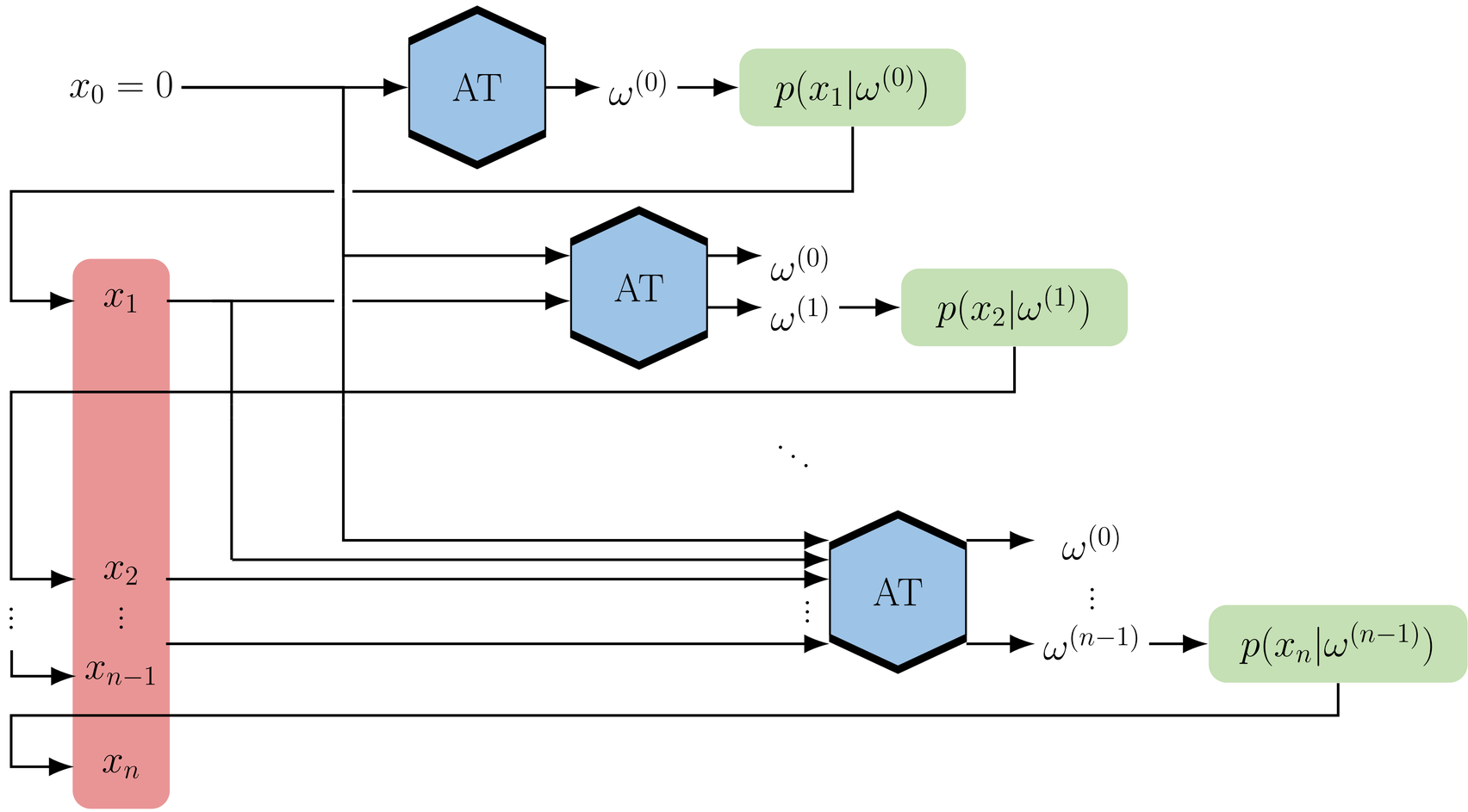
$$\omega^{(i)} = \{w_j^{(i)}, \mu_j^{(i)}, \sigma_j^{(i)}\}$$

$$p(x_{i+1} \,|\, \omega^{(i)}) = \sum_j w_j^{(i)} \mathcal{N}(x_{i+1}; \mu_j^{(i)}, \sigma_j^{(i)})$$



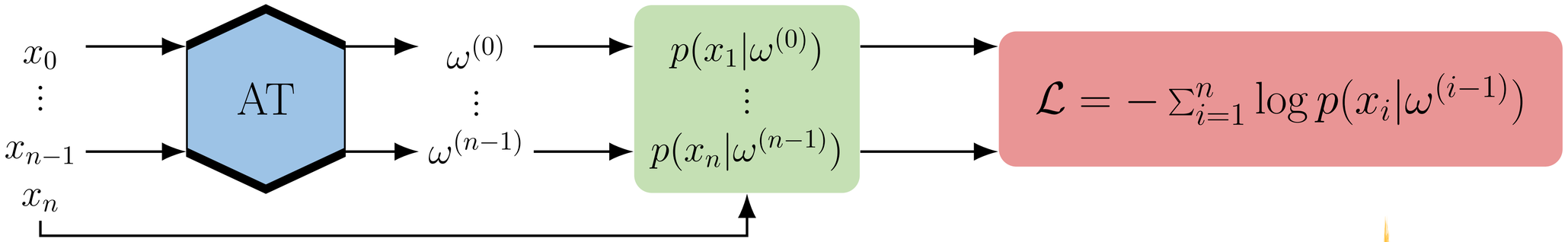
Autoregressive Transformer

(Slow) Sampling



Autoregressive Transformer

(Fast) Density Estimation



$$\mathscr{L} = \Big\langle -\log p(x) \Big\rangle_{x \sim p_{\text{data}}} = \sum_{i=1}^{n} \Big\langle -\log p(x_i \,|\, \omega^{(i-1)}) \Big\rangle_{x \sim p_{\text{data}}}$$

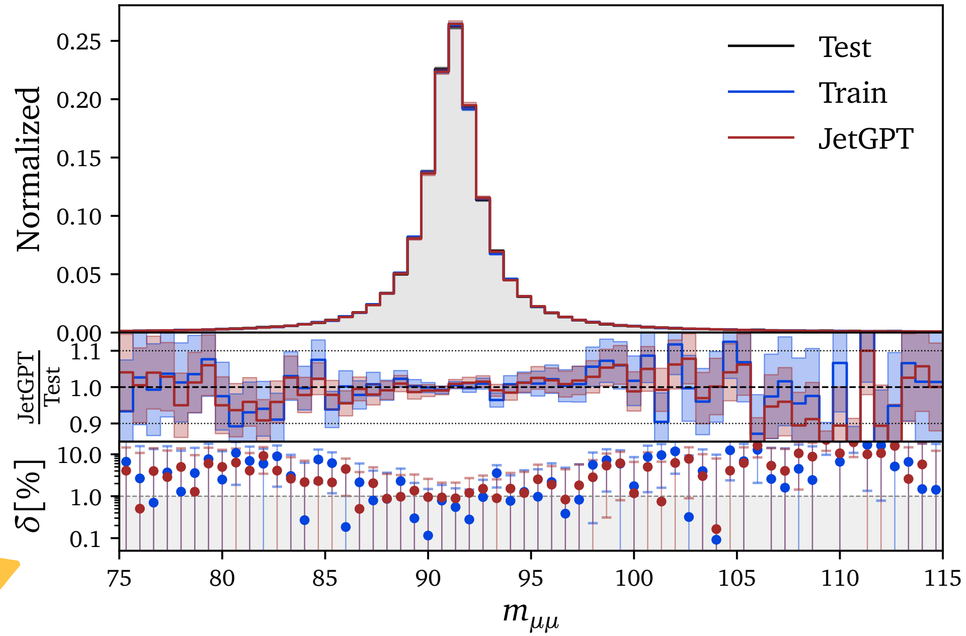

Generating
LHC Events

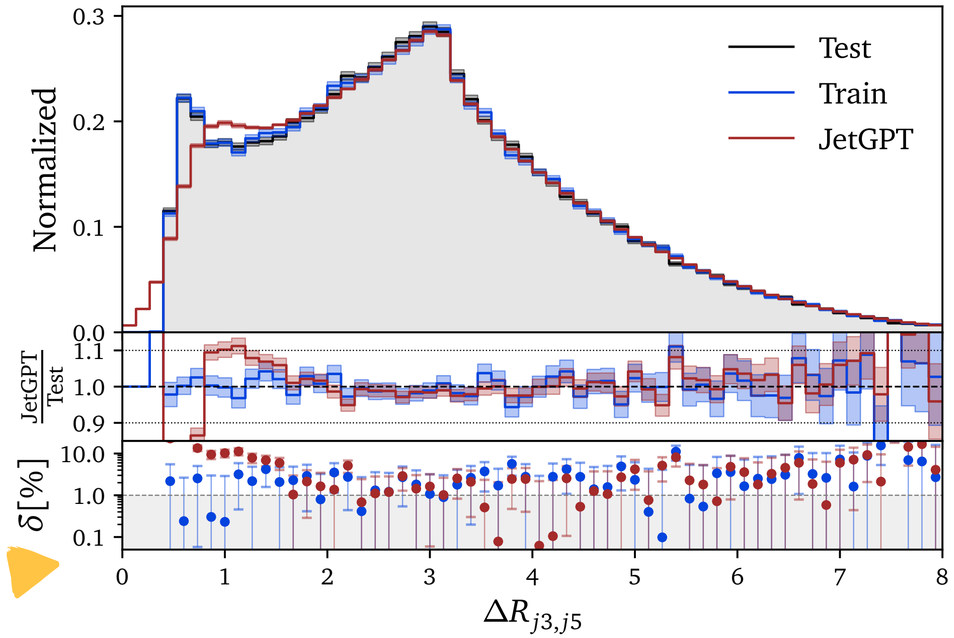
Generating LHC Events

Dataset: $Z(\mu\mu) + \text{jets}$

- MadGraph + Pythia
- Events with 3-5 jets (5M, 1M, 200k)
- Autoregressive Ordering:

$$\left\{ m_Z, \underbrace{\phi_j, \eta_j}_{\Delta R_{jj}}, \phi_Z, \eta_Z, p_T, m_j \right\}$$

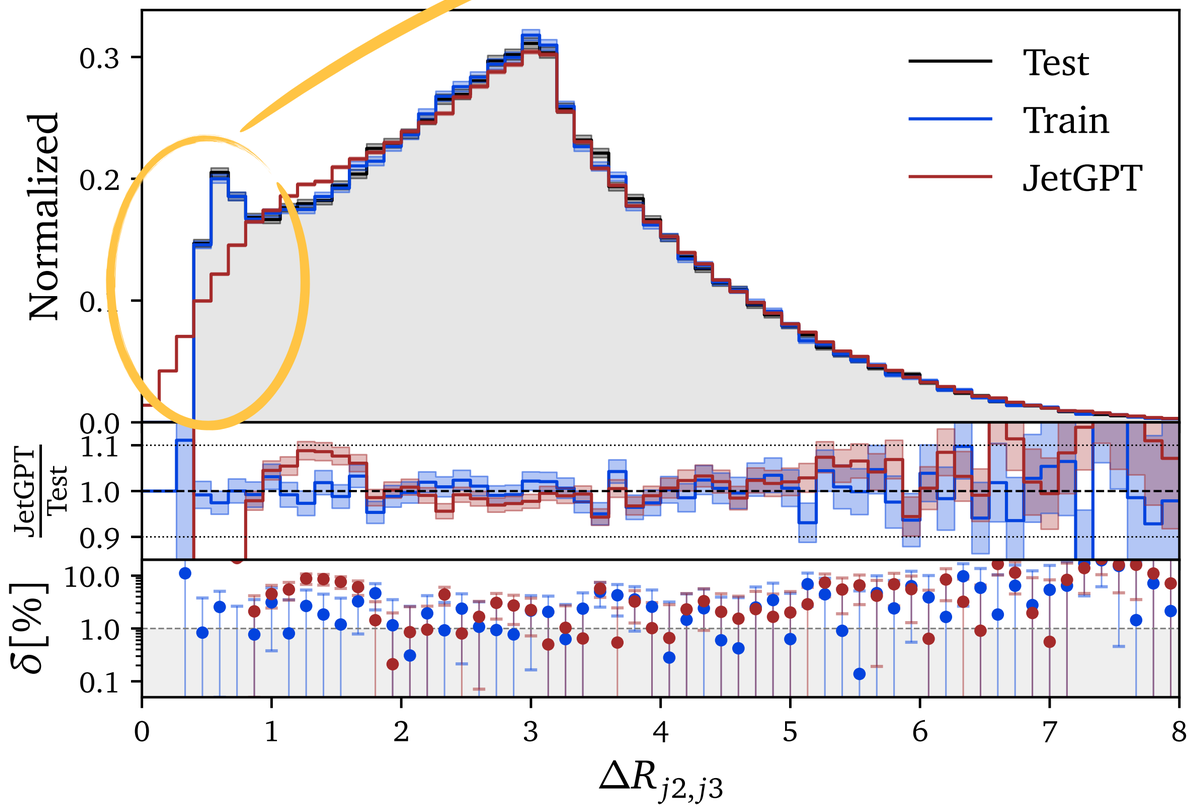




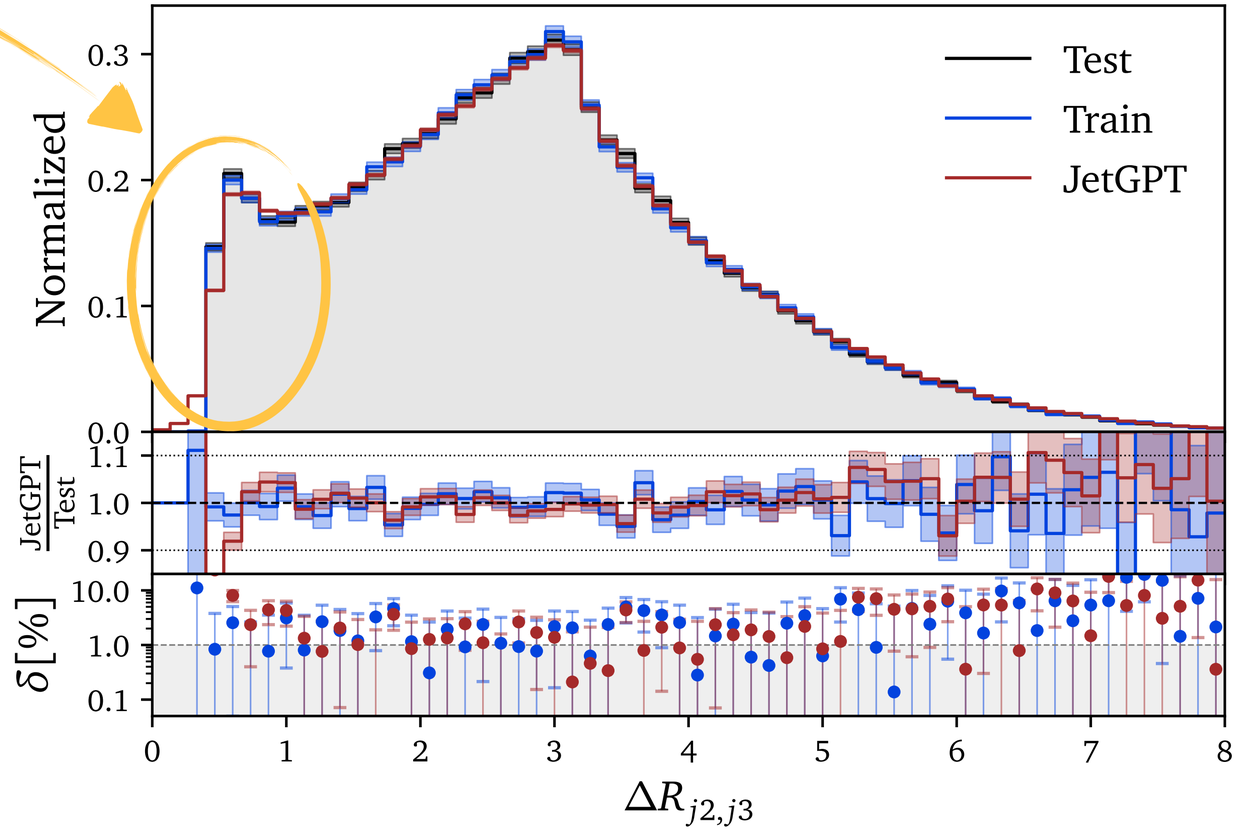
Generating LHC Events

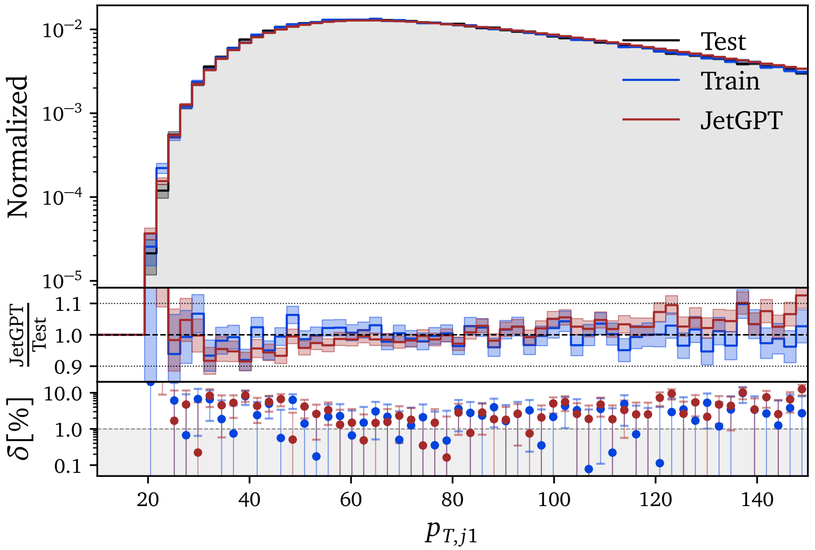
Joint Training

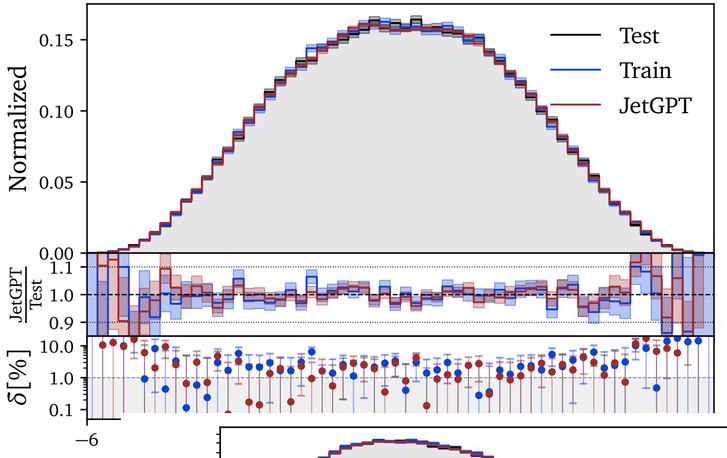
Naive Training (5j only)

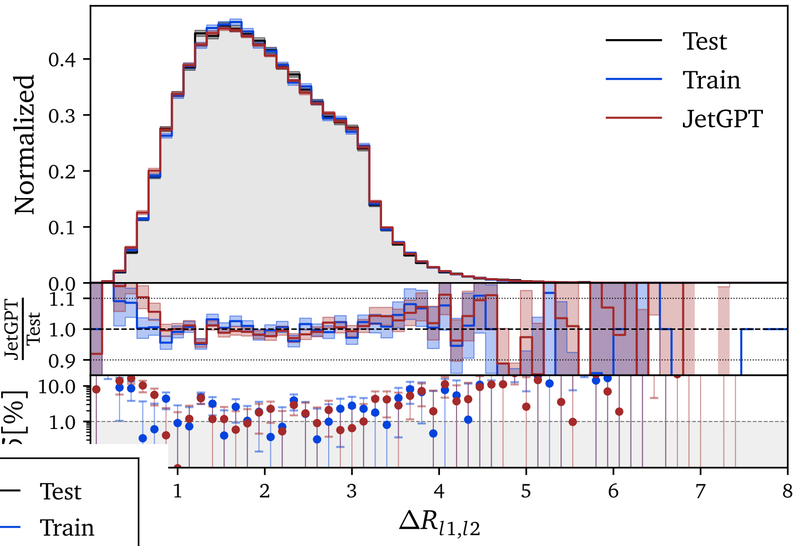


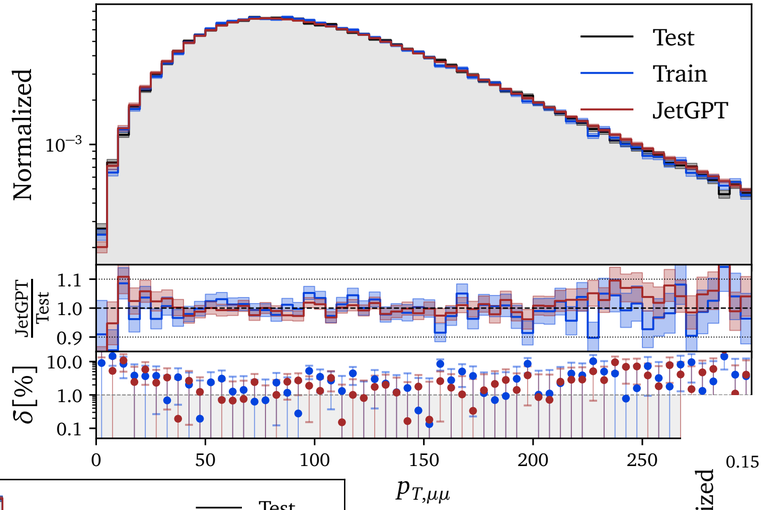
Joint Training (3j-5j)

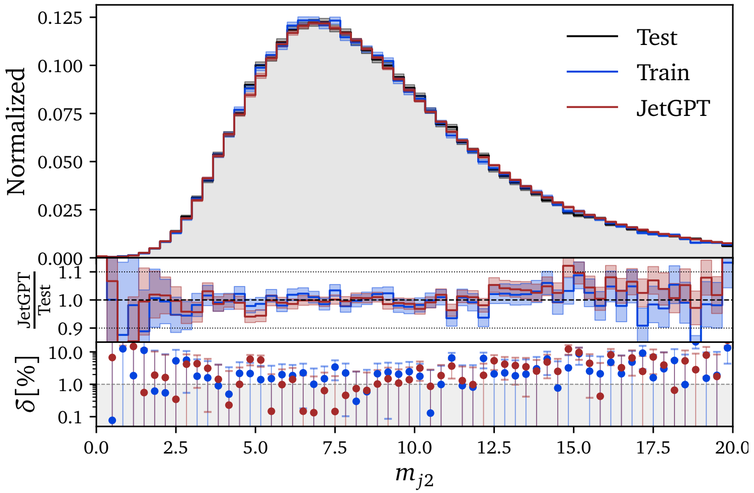


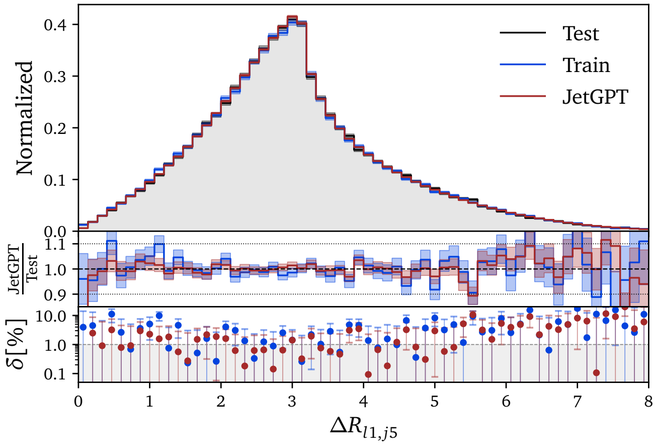

Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]
$p_{T,j1}$

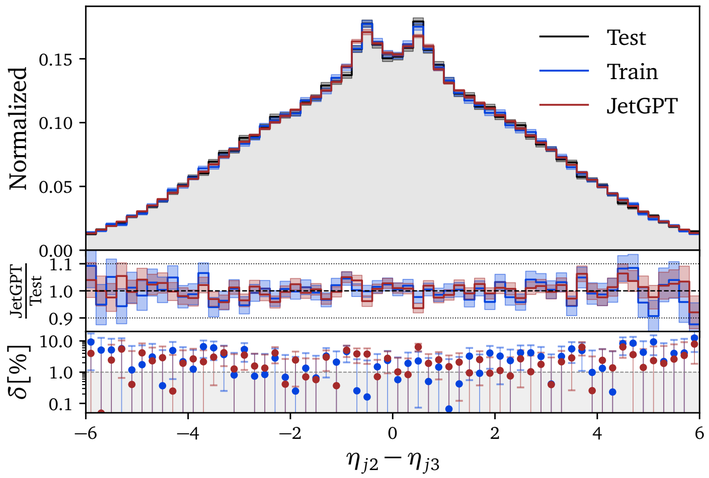

Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]


Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]
$\Delta R_{l1,l2}$


Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]
$p_{T,\mu\mu}$


Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]
m_{j2}


Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]
$\Delta R_{l1,j5}$


Normalized
Test
Train
JetGPT
JetGPT / Test
δ[%]
$\eta_{j2} - \eta_{j3}$


Classifier
Reweighting

Classifier Reweighting

Likelihood Ratio Trick

$$\mathscr{L}_{\text{BCE}} = -\left\langle \log D(x) \right\rangle_{x\sim p_{\text{data}}} - \left\langle \log(1-D(x)) \right\rangle_{x\sim p_{\text{model}}}$$

$$= -\int dx\; p_{\text{data}} \log D - \int dx\; p_{\text{model}} \log(1-D)$$

Equations of Motion

$$0 = \frac{\delta \mathscr{L}_{\text{BCE}}}{\delta D} = \frac{p_{\text{data}}}{D} - \frac{p_{\text{model}}}{1-D}$$

$$\frac{p_{\text{data}}}{p_{\text{model}}} = \frac{D}{1-D}$$

Classification

$$w(x) = \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)}$$

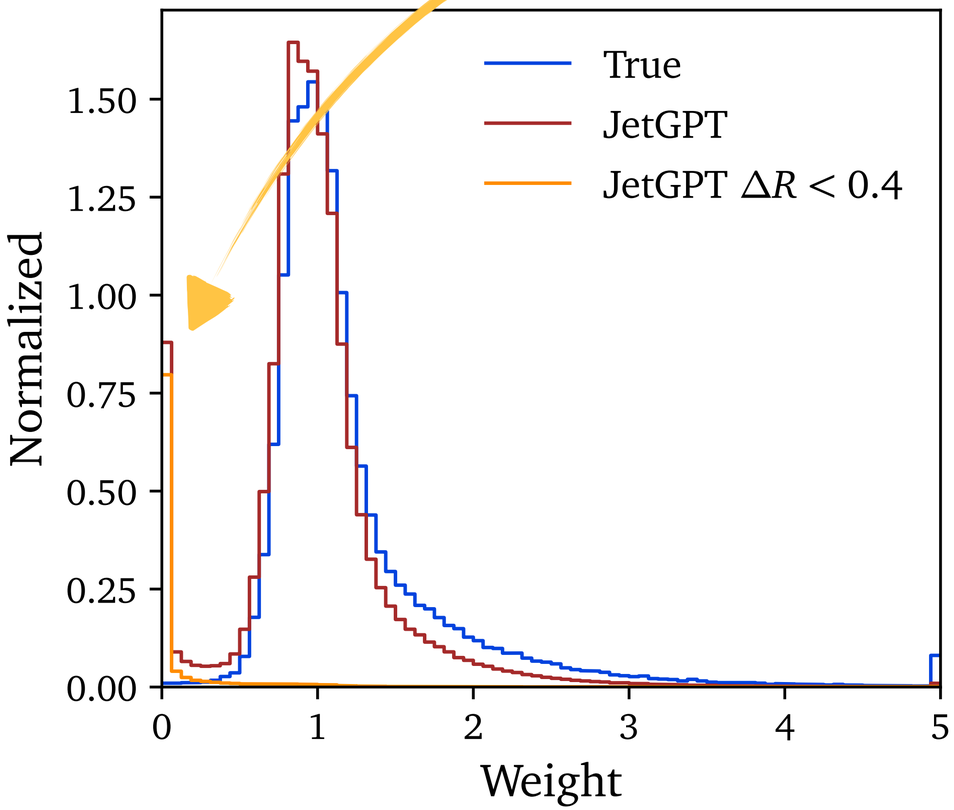
Reweighting

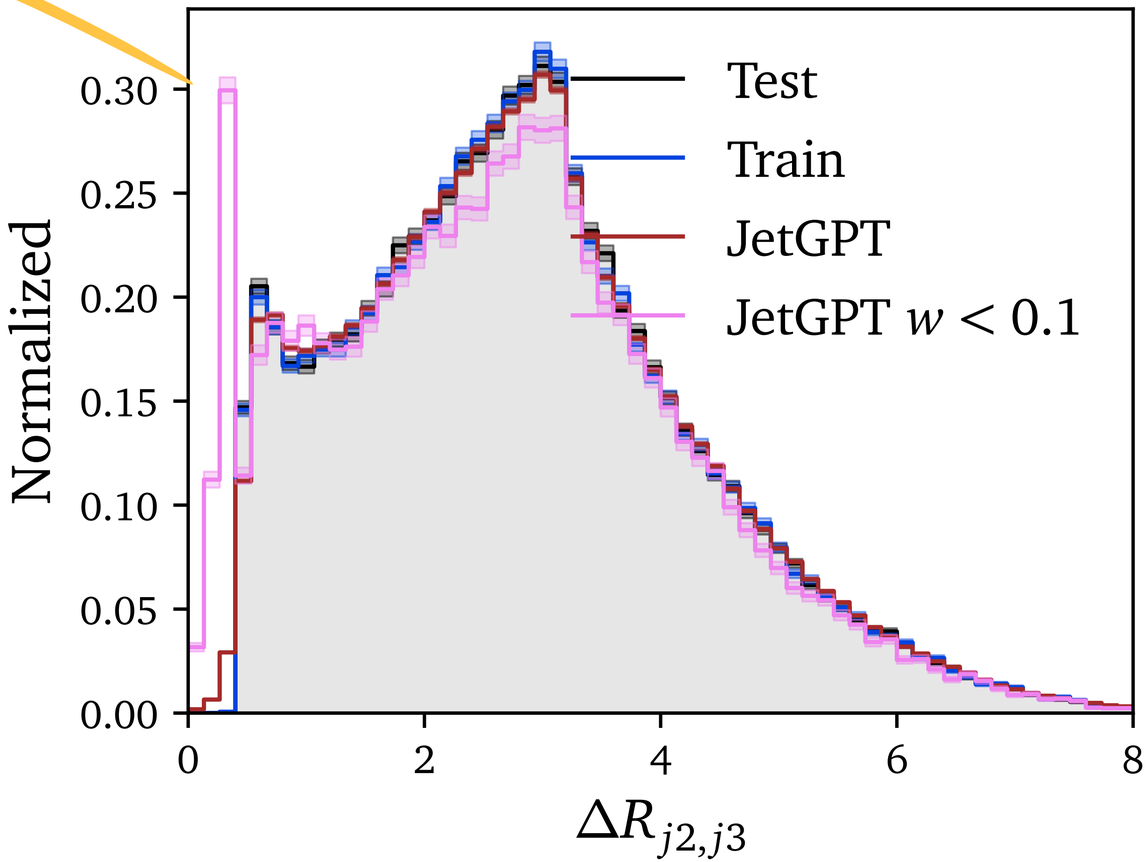
$$p_{\text{data}} = p_{\text{model}} \times \frac{p_{\text{data}}}{p_{\text{model}}}$$

Classifier Reweighting

Track the limitations

$$w(x) = \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)}$$

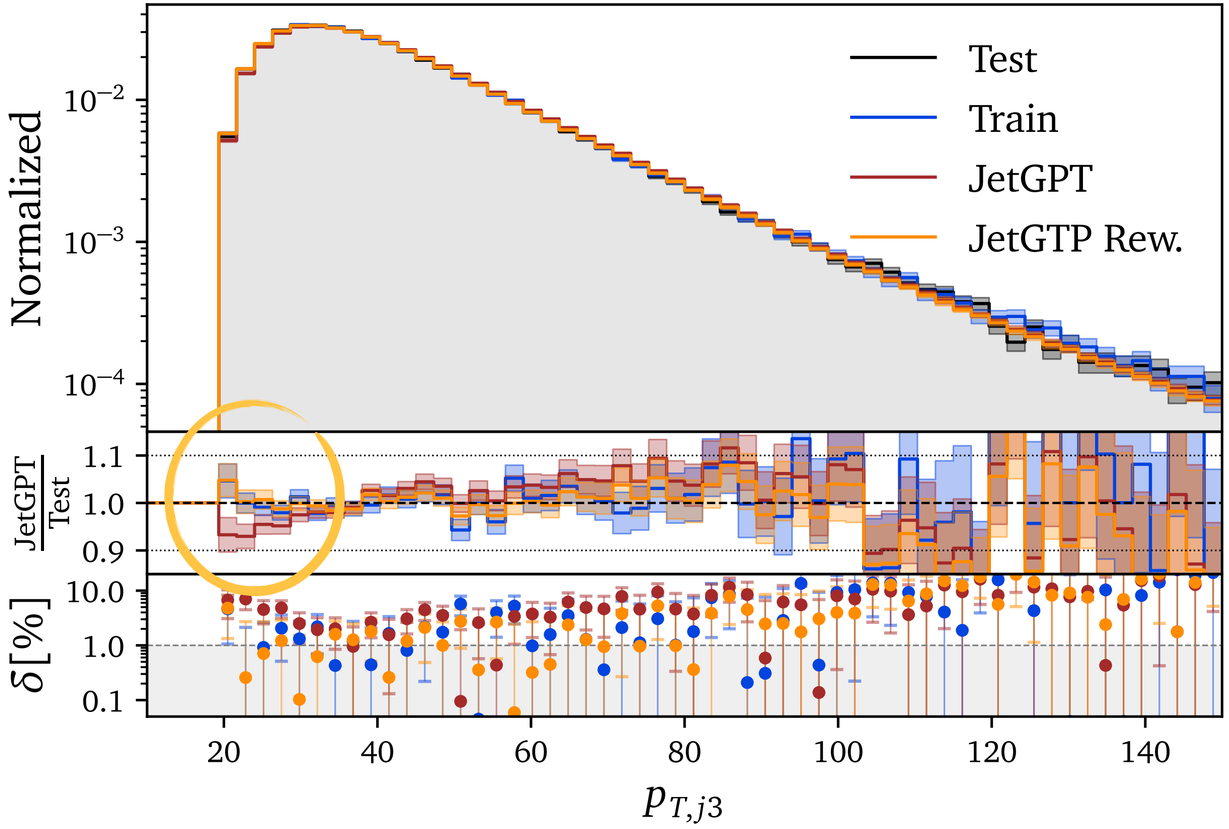


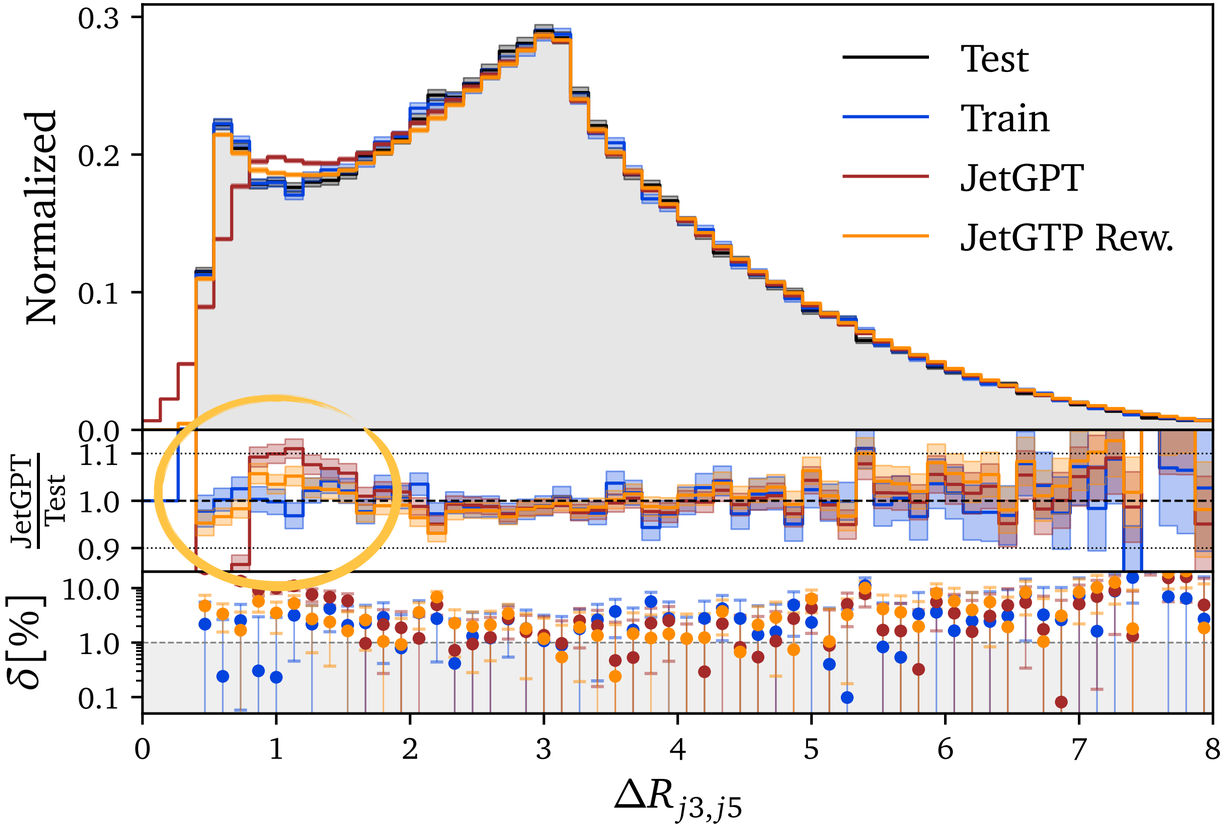


Classifier Reweighting

Overcome the limitations

$$p_{\text{data}} = p_{\text{model}} \times \frac{p_{\text{data}}}{p_{\text{model}}}$$





Conclusions

- Neural Networks can generate LHC events with **percent-level** accuracy
- Neural Network Classifiers can **find and reweight** remaining discrepancies
- Transformers can be **trained jointly** on high-multiplicity datasets
- **Autoregressive ordering** as powerful handle to provide implicit bias

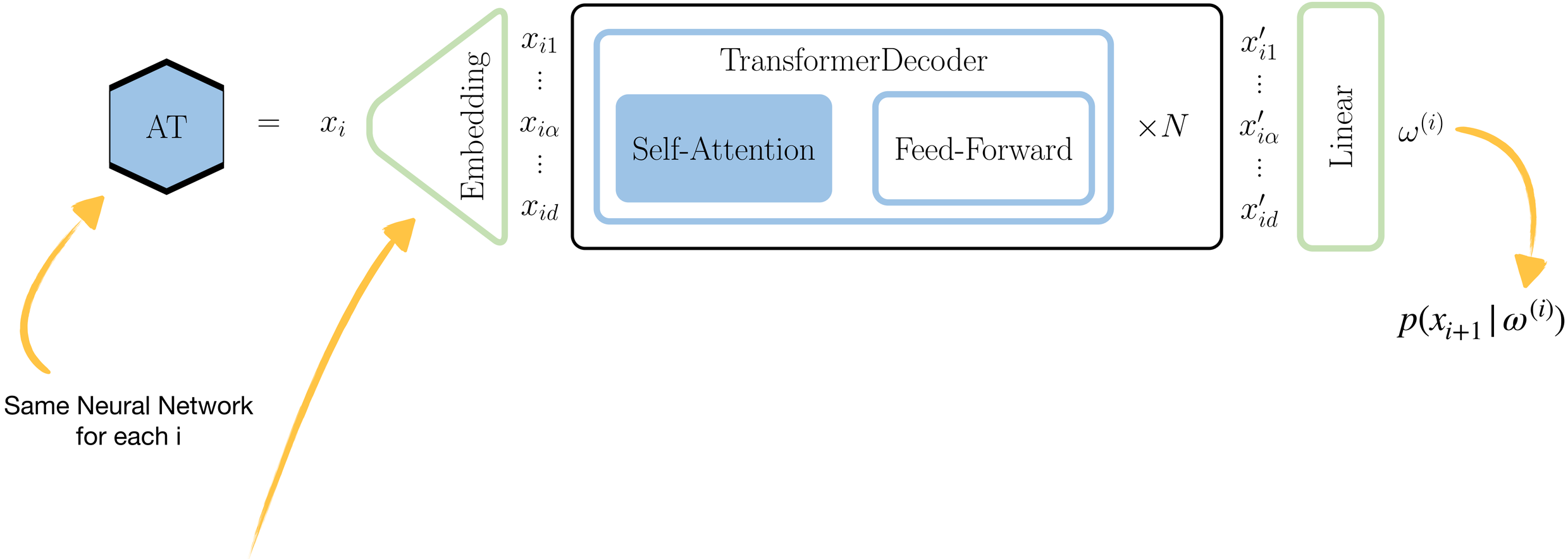

Backup

Autoregressive Transformer

Transformer Architecture

Autoregressive Transformer

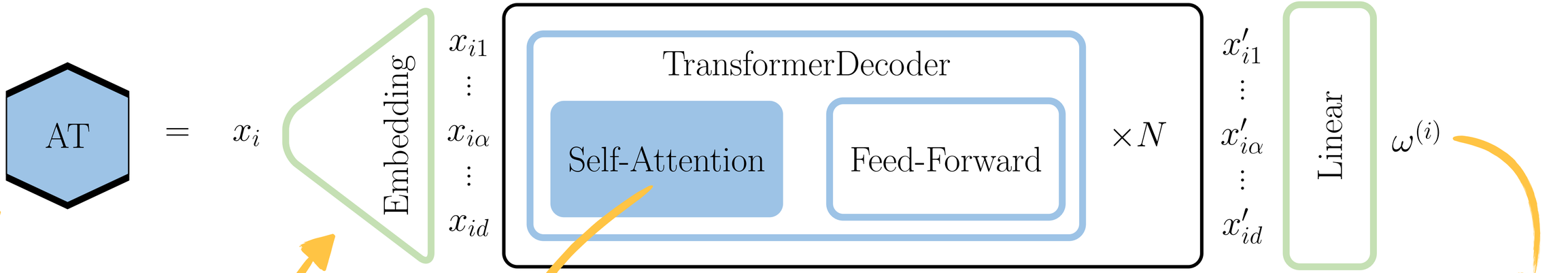
Transformer Architecture



$$x_{i\alpha} = \mathrm{TE}_\alpha(x_i) + \mathrm{PE}_\alpha(i) + \mathrm{PE}_\alpha(i_{\mathrm{mult}})$$

Autoregressive Transformer

Transformer Architecture



Same Neural Network for each i

$$x_{i\alpha} = \mathrm{TE}_\alpha(x_i) + \mathrm{PE}_\alpha(i) + \mathrm{PE}_\alpha(i_{\mathrm{mult}})$$

Learnable

$$\mathrm{SelfAttention}_{i\alpha}(x) = \mathrm{Softmax}_j\left(\frac{x_{i\gamma} W^Q_{\delta\gamma} W^K_{\delta\sigma} x_{j\sigma}}{\sqrt{d}}\right) W^V_{\alpha\beta} x_{j\beta}$$

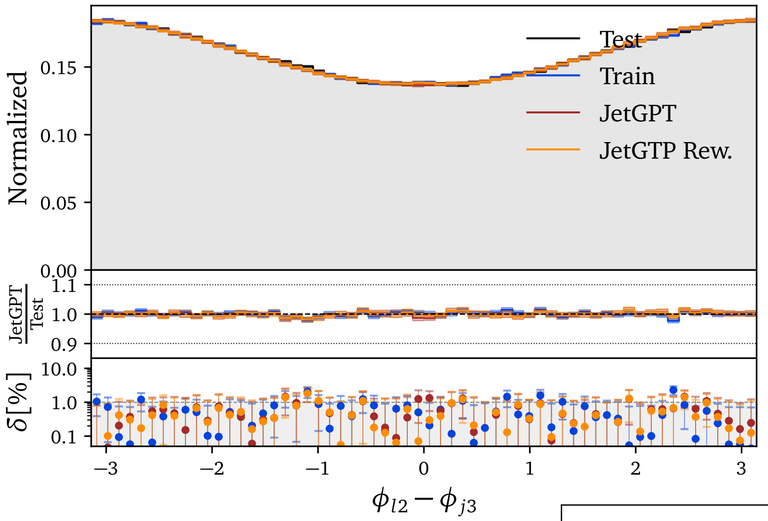
$$= A_{ij}(x) \, W^V_{\alpha\beta} \, x_{j\beta}$$

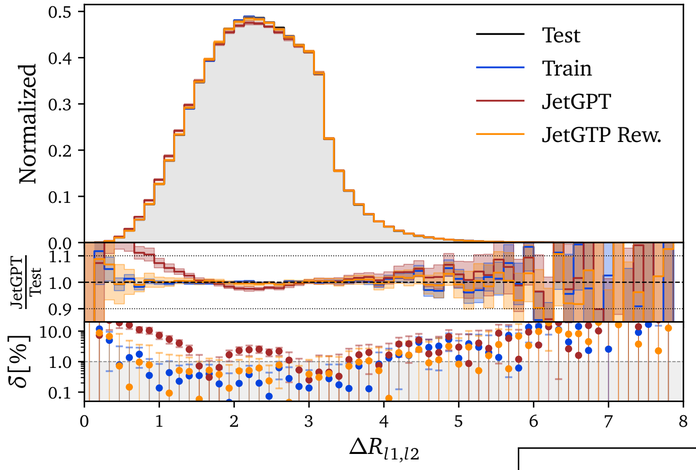
Feature Space Latent Space

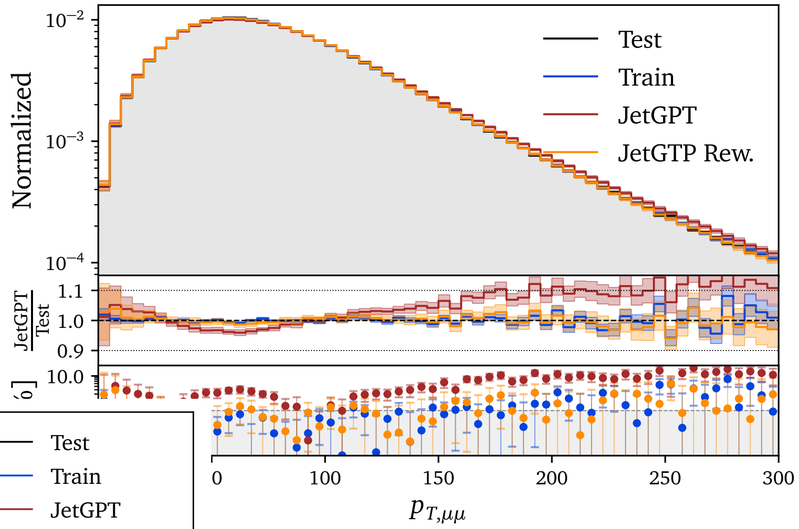
Generating LHC Events

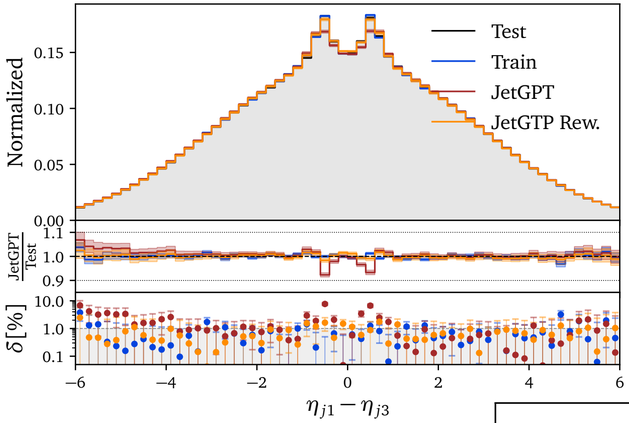
Preprocessing

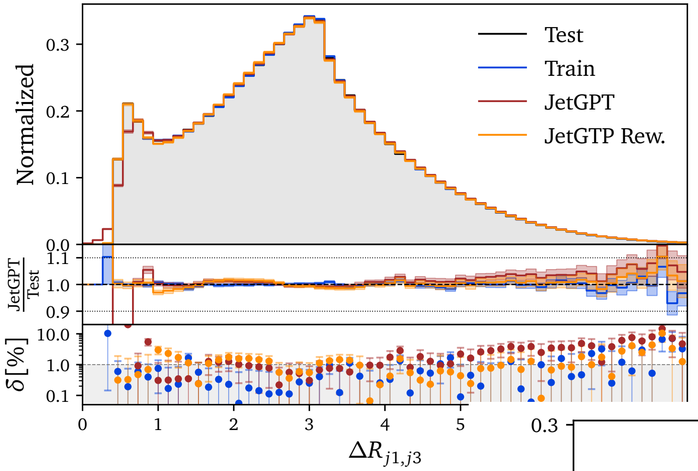
$$p^\mu \rightarrow \begin{matrix} p_T \\ \phi \\ \eta \\ m \end{matrix} \rightarrow \begin{matrix} \log(p_T - p_{T,\min}) \\ \operatorname{arctanh}\frac{\phi}{\pi} \\ \eta \\ \log m \end{matrix} = y_i \rightarrow x_i = \frac{y_i - \bar{y}_i}{\sigma_{y_i}}$$

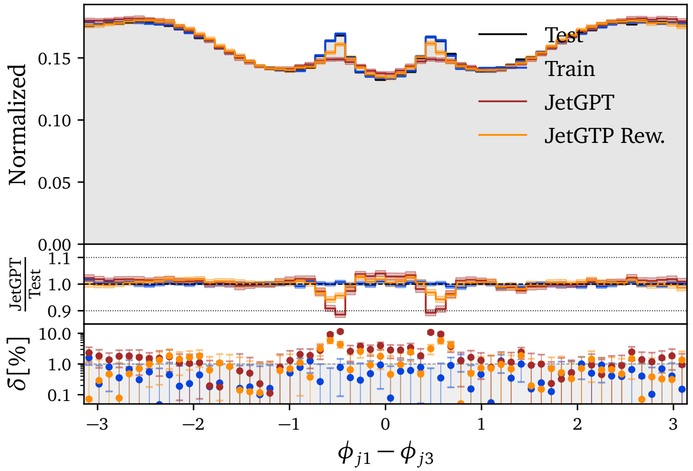

Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\phi_{l2} - \phi_{j3}$

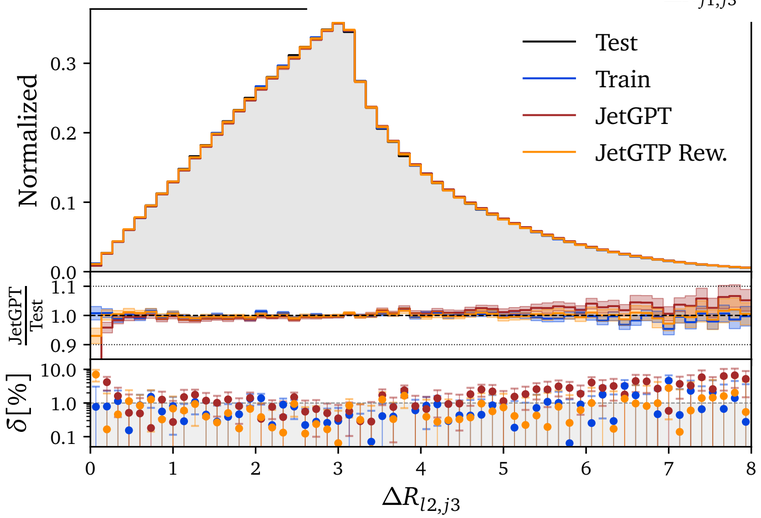

Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\Delta R_{l1,l2}$

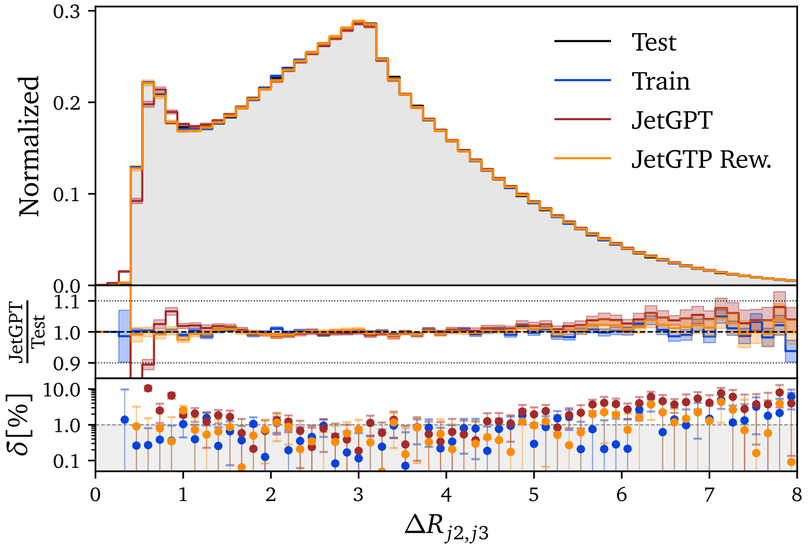

Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$p_{T,\mu\mu}$


Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\eta_{j1} - \eta_{j3}$


Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\Delta R_{j1,j3}$


Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\phi_{j1} - \phi_{j3}$


Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\Delta R_{l2,j3}$


Normalized
Test
Train
JetGPT
JetGTP Rew.
JetGPT / Test
δ[%]
$\Delta R_{j2,j3}$