UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

High Multiplicity with JetGPT

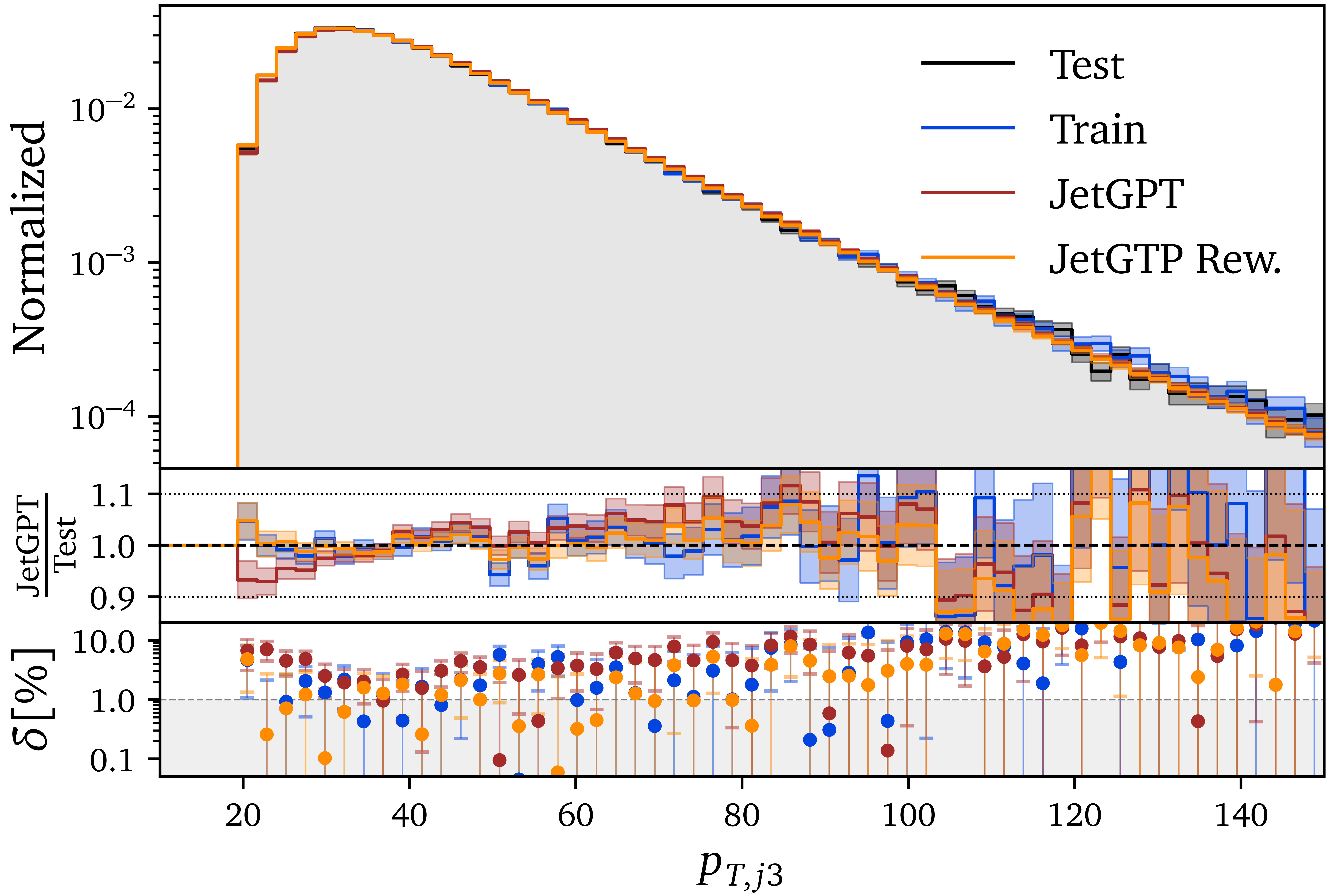
LHC Event Generation with Autoregressive Transformers

Jonas Spinner

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

2305.10475

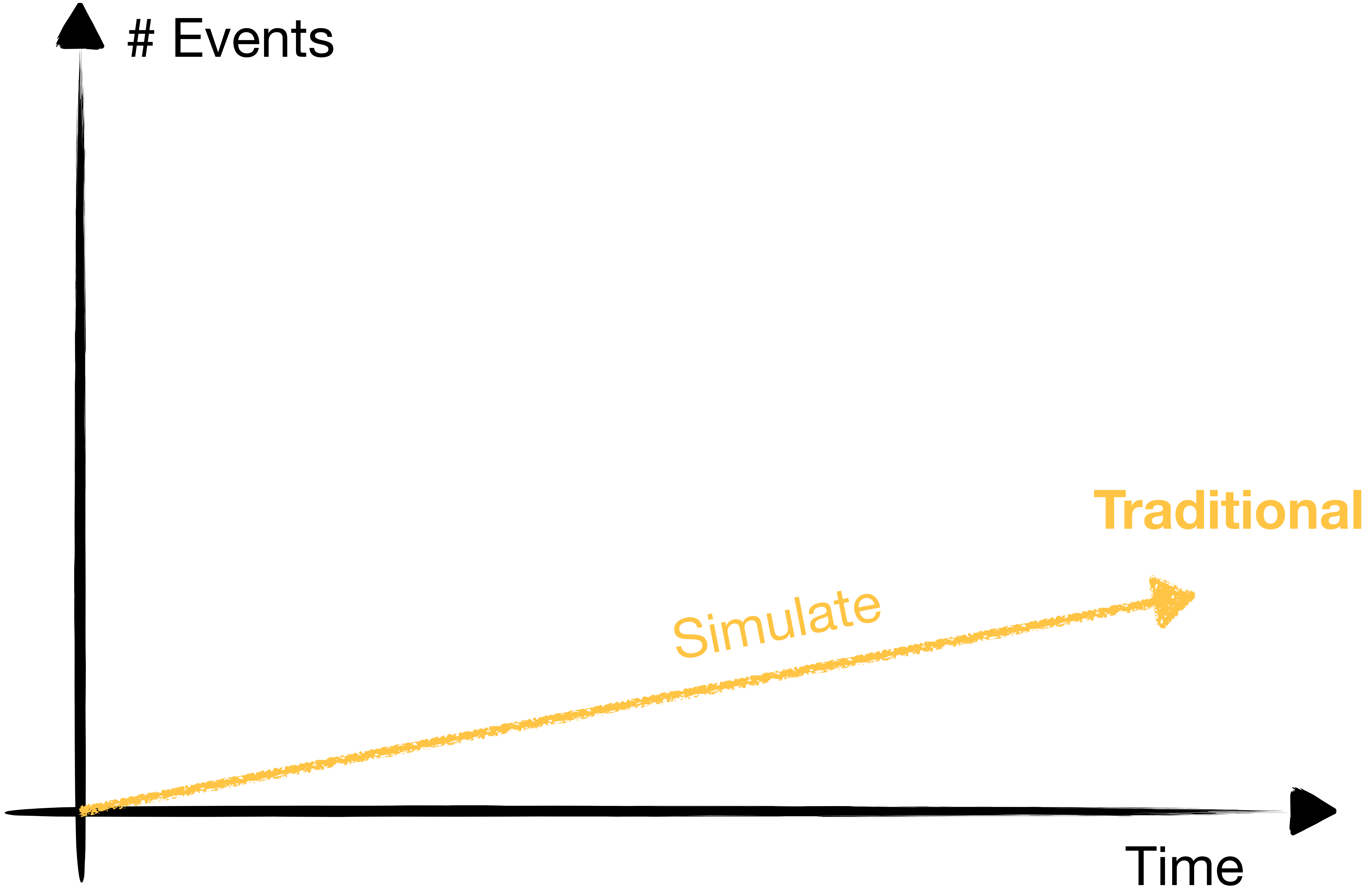
IRN Terascale Marseille 2023



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

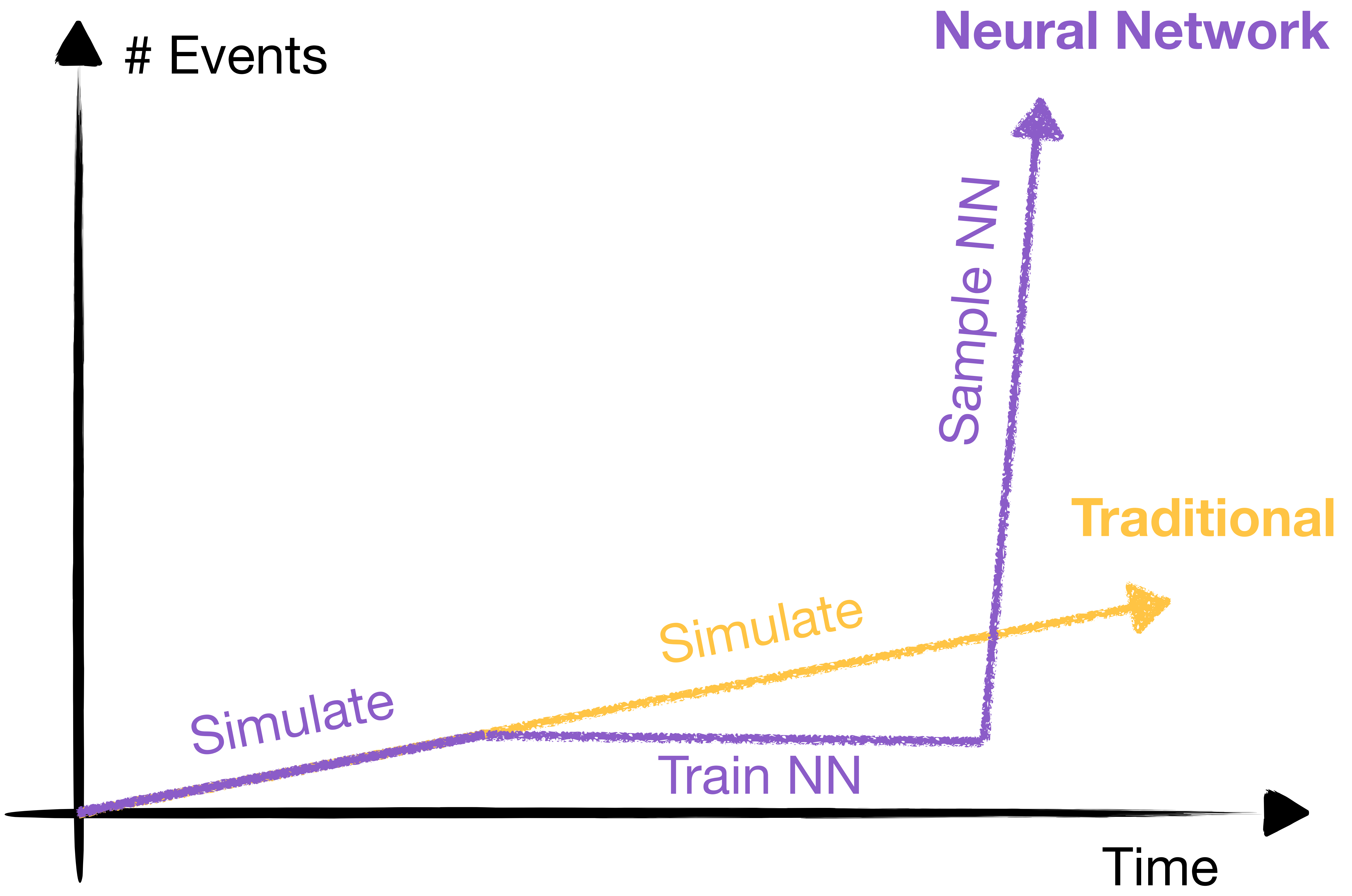
Motivation

End-to-End-Generation with Neural Networks



Motivation

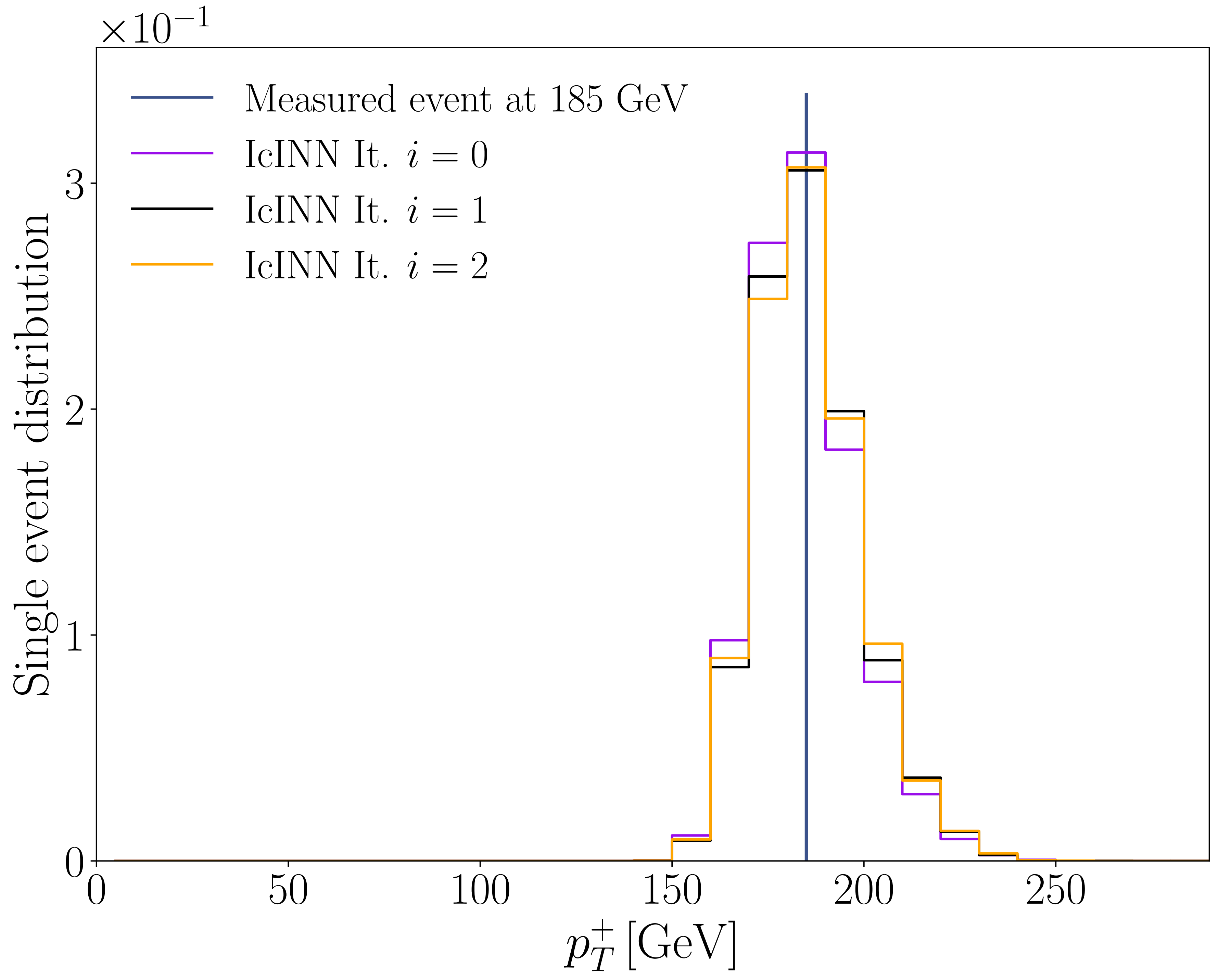
End-to-End-Generation with Neural Networks



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

Motivation

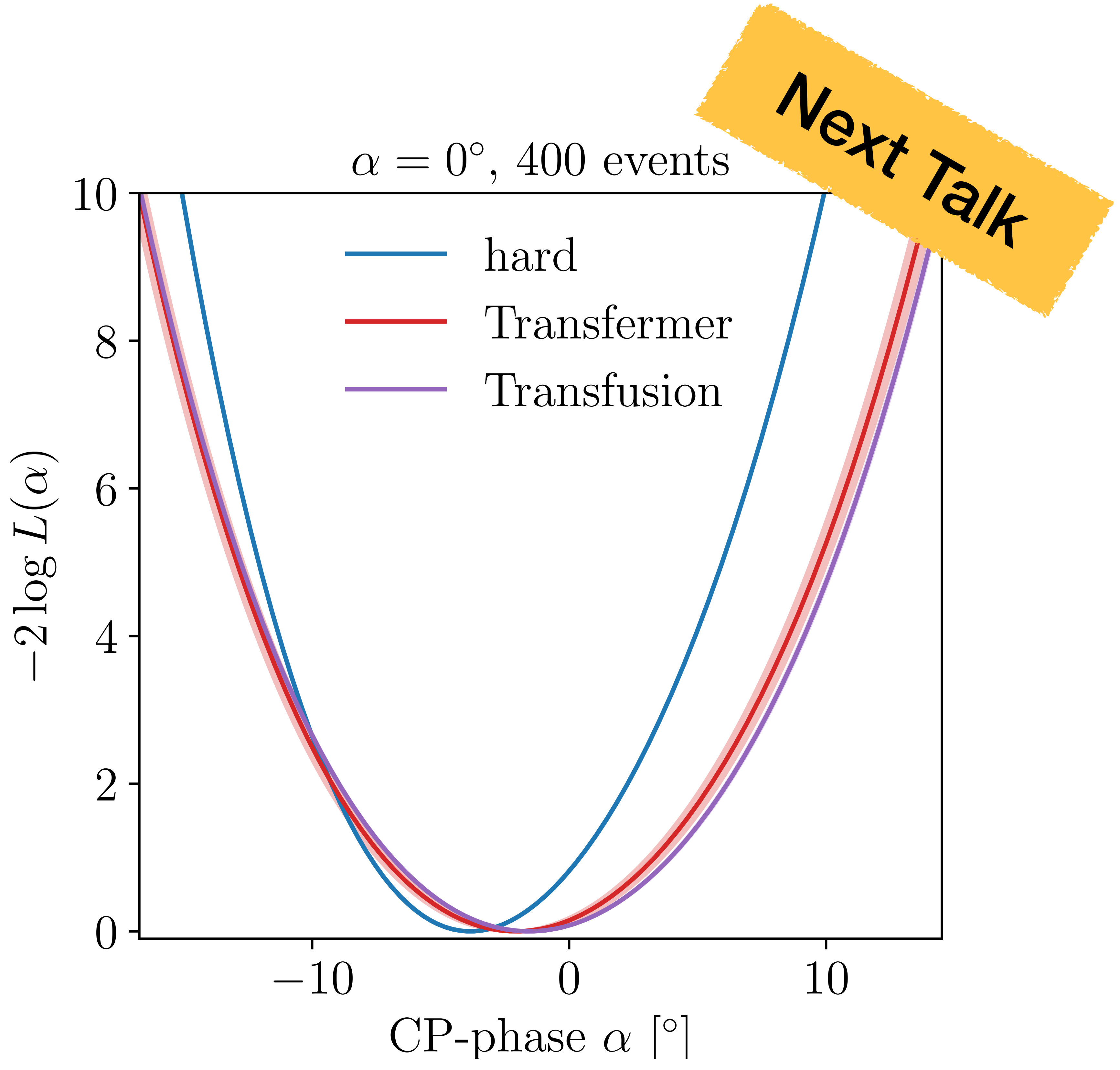
Inference with Generative Neural Networks



2006.06685

Unfolding

2212.08674



Matrix Element Method

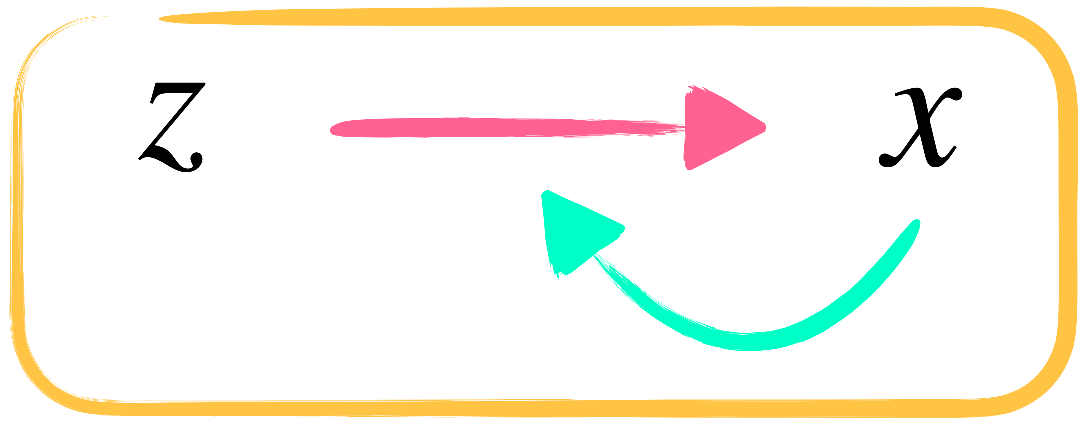
2210.00019

2310.07752

Motivation

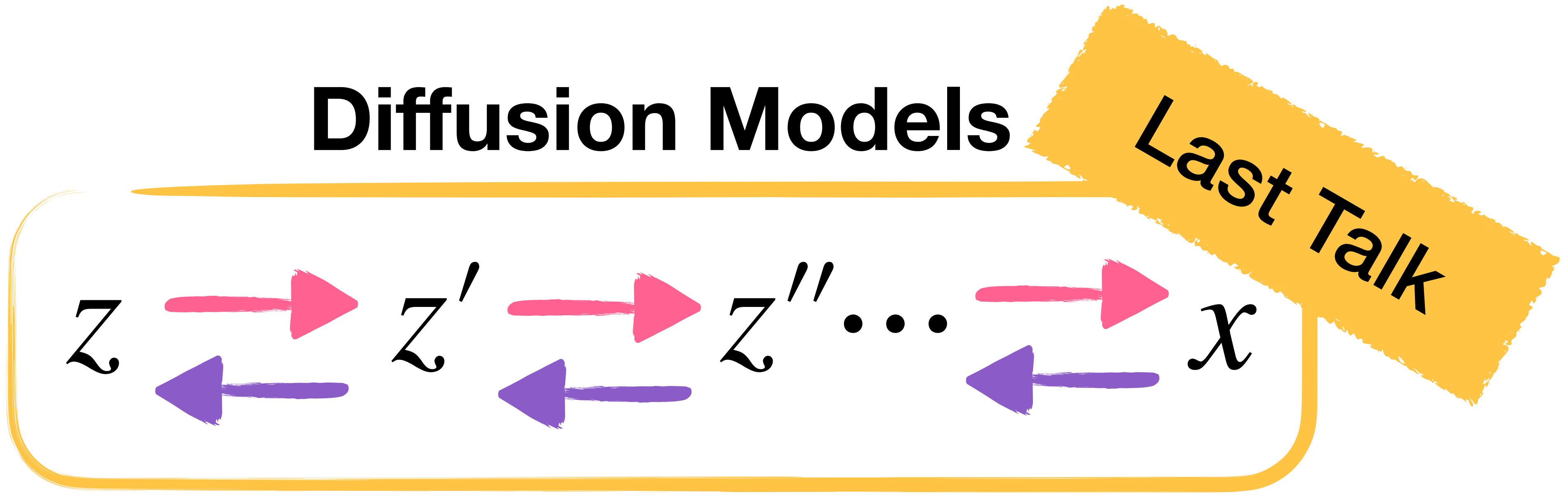
Generative Neural Networks

GANs



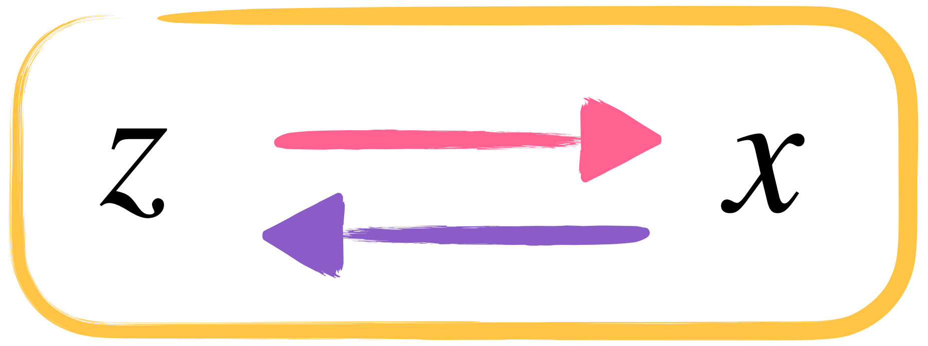
1907.03764

Diffusion Models



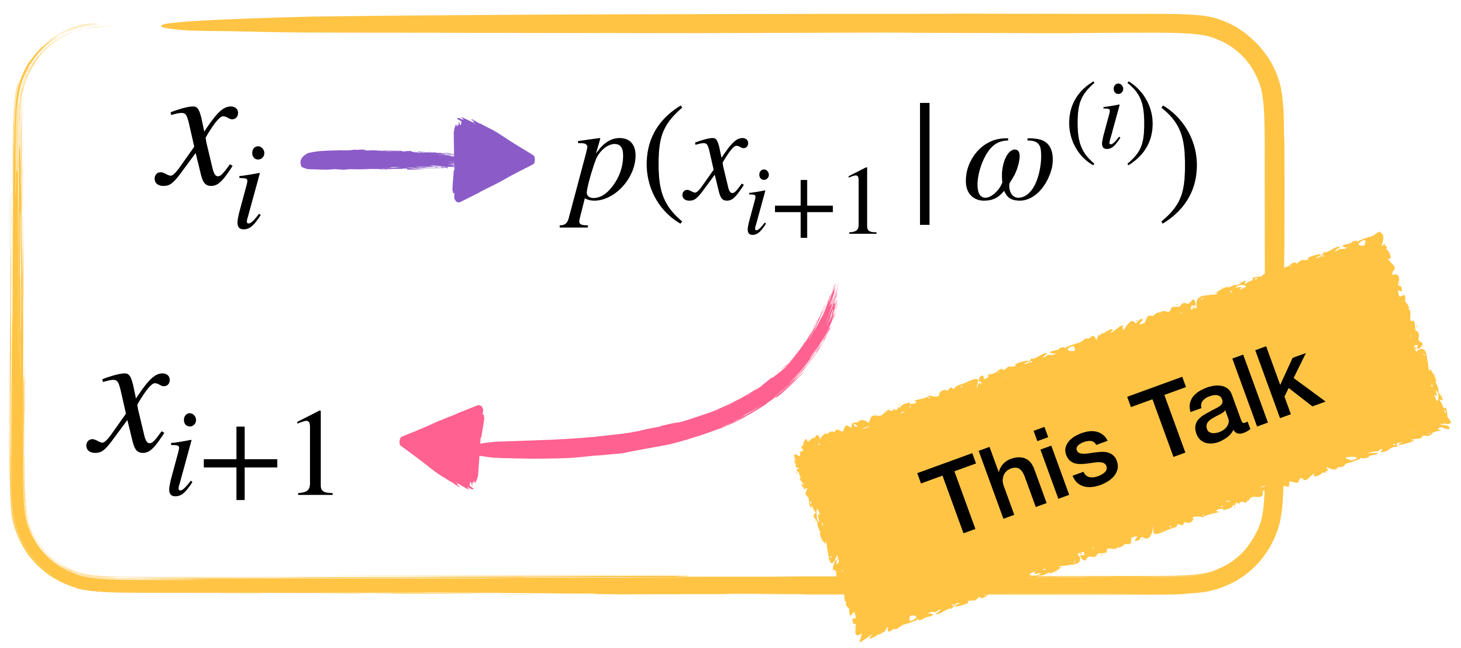
The ***Precise*** 2305.10475

Normalizing Flows



2110.13632 The ***Fast***

Autoregressive Transformers



The ***Flexible*** 2305.10475

x **Phase Space**

z **Latent Space**

Sampling

Density Estimation

Classifier

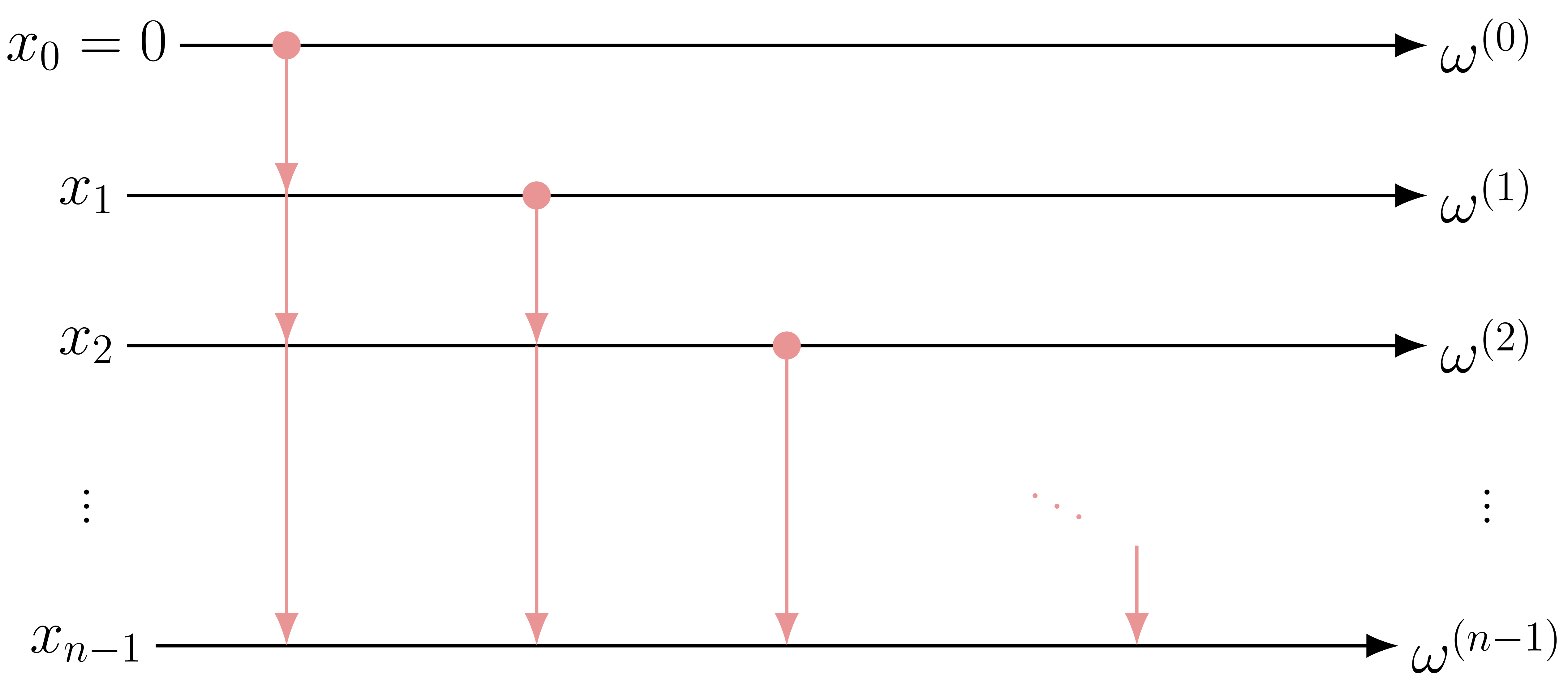

Autoregressive
Transformers

Autoregressive Transformer

Generating Events

Autoregression

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



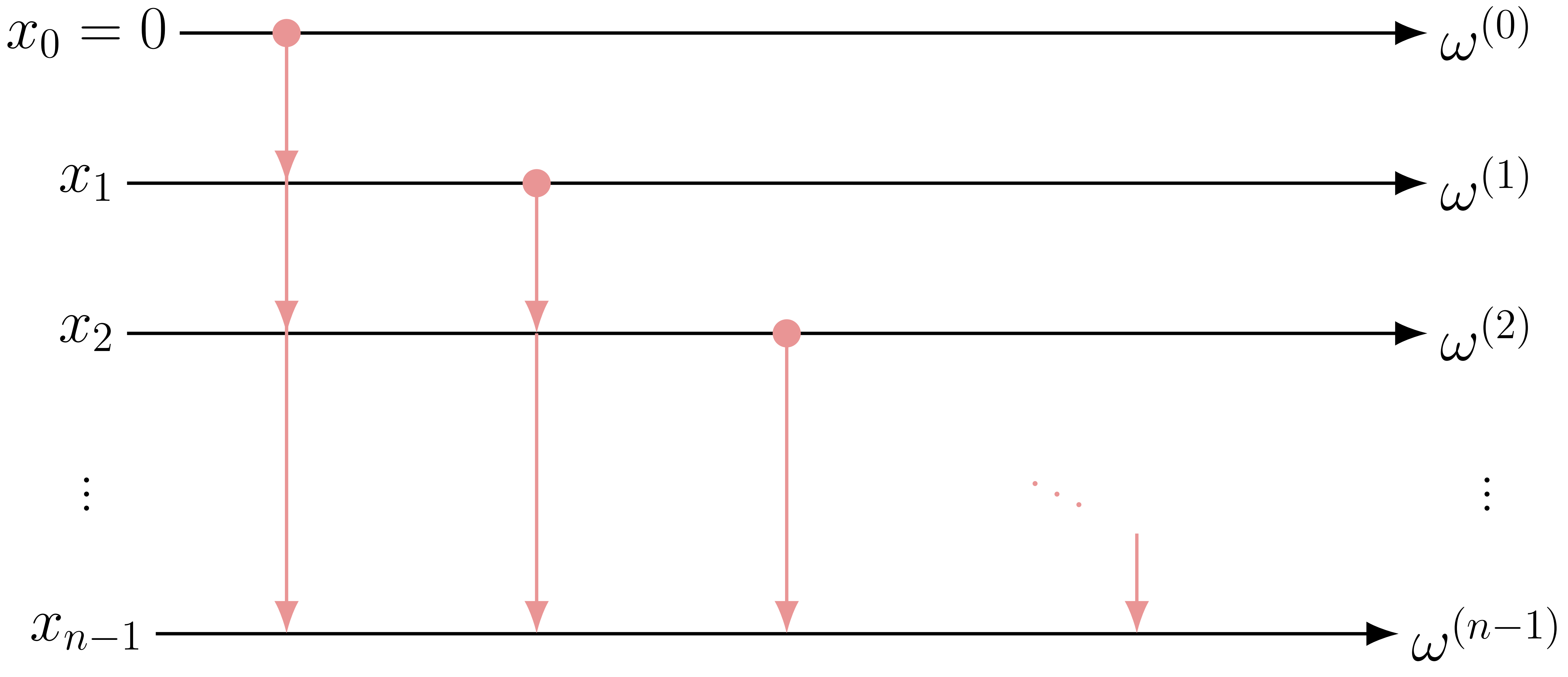
Autoregressive Transformer

Generating Events

Autoregression

$$p(x_1, x_2 \cdots x_n) = p(x_1) \qquad p(x_2 | x_1) \qquad \cdots \qquad p(x_n | x_1 \cdots x_{n-1})$$

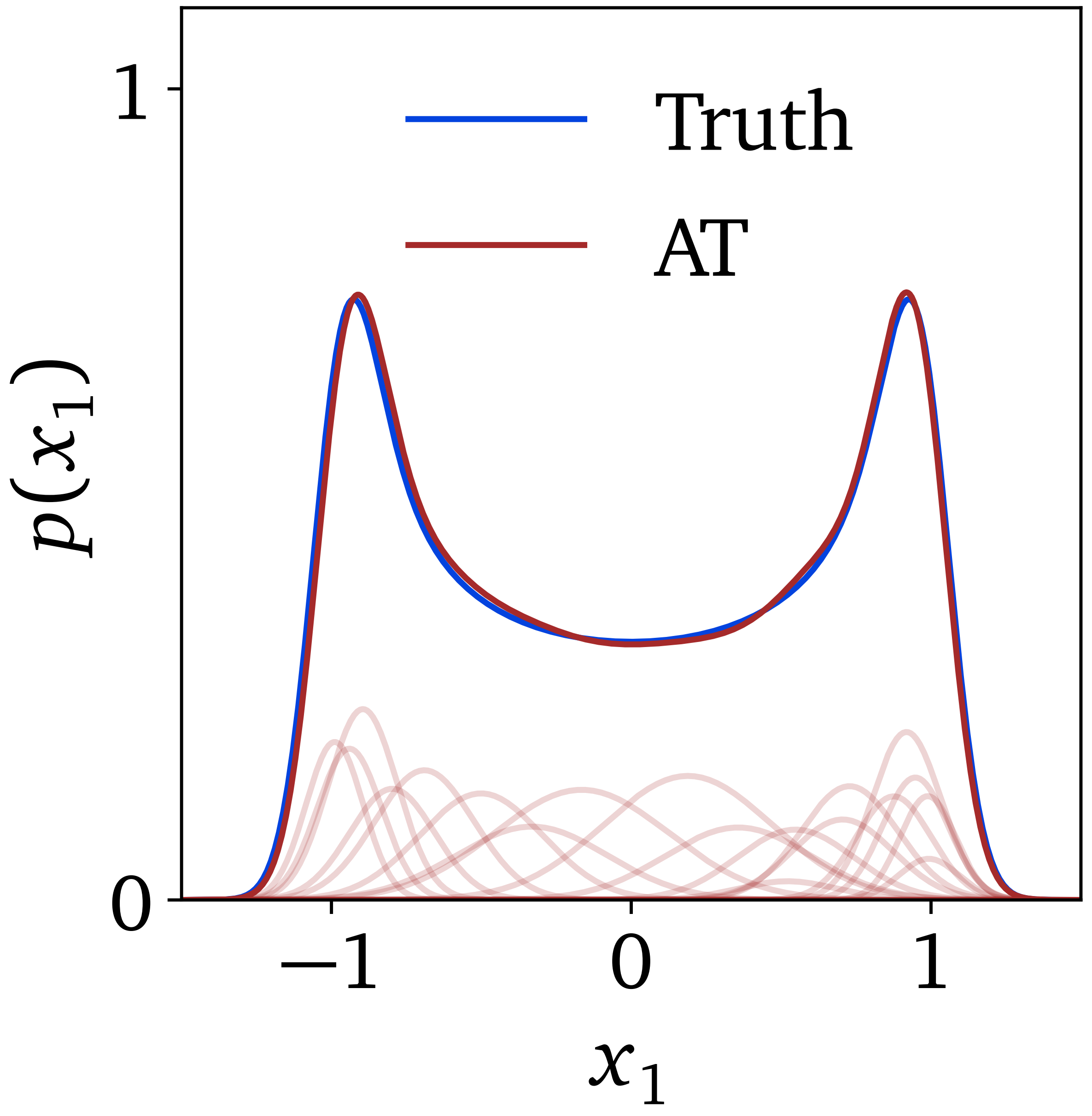
$$= p(x_1 | \omega^{(0)}) \qquad p(x_2 | \omega^{(1)}) \qquad \cdots \qquad p(x_n | \omega^{(n-1)})$$



Gaussian Mixture Model

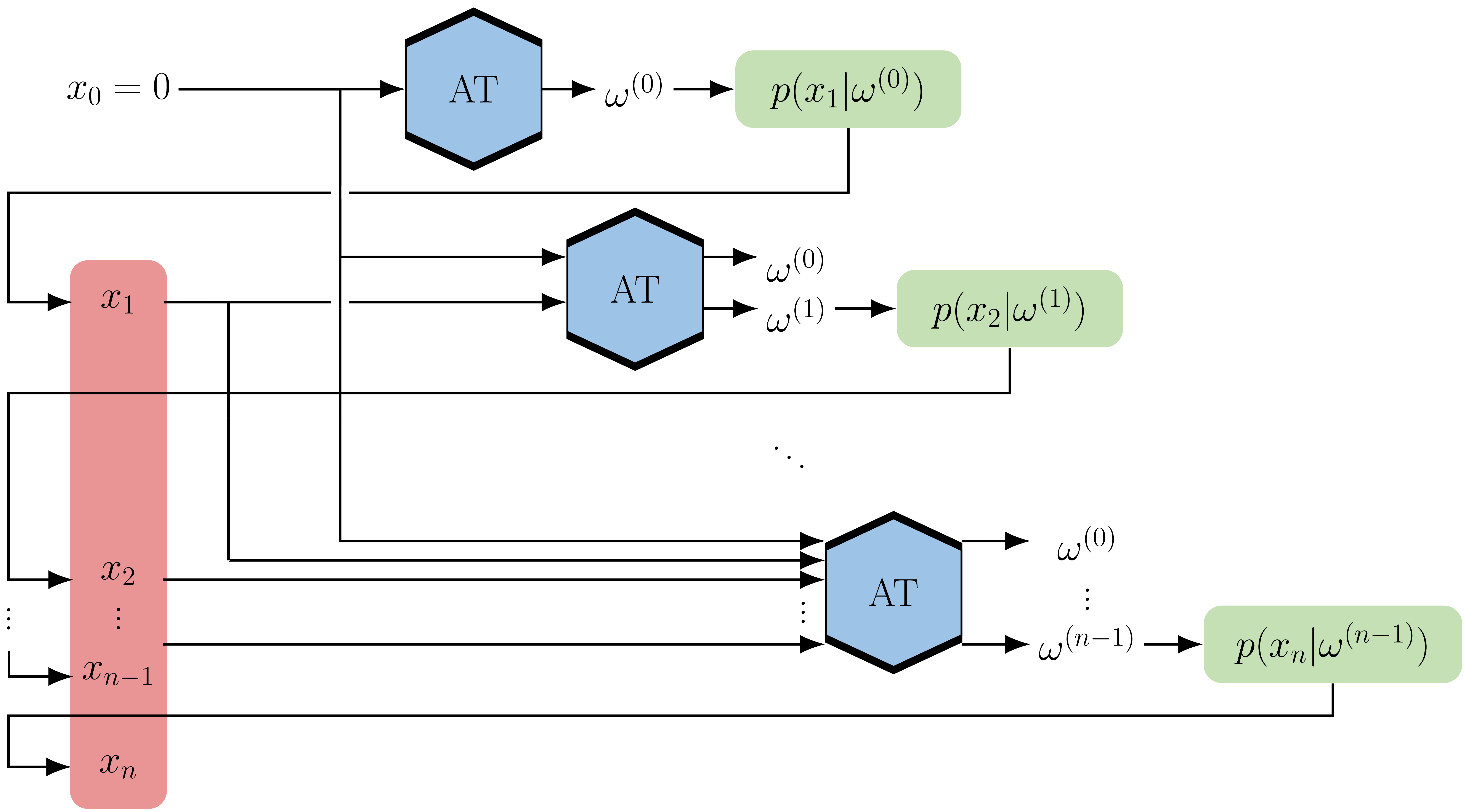
$$\omega^{(i)} = \{w_j^{(i)}, \mu_j^{(i)}, \sigma_j^{(i)}\}$$

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



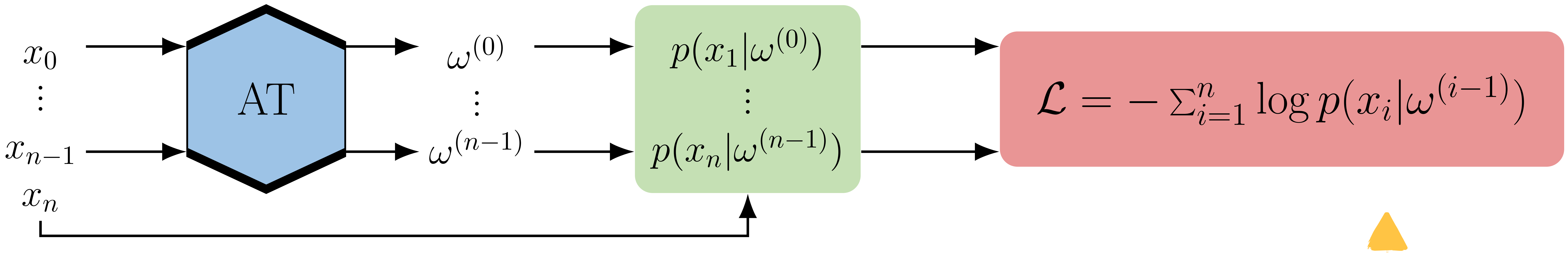
Autoregressive Transformer

Slow Sampling



Autoregressive Transformer

Fast Training



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

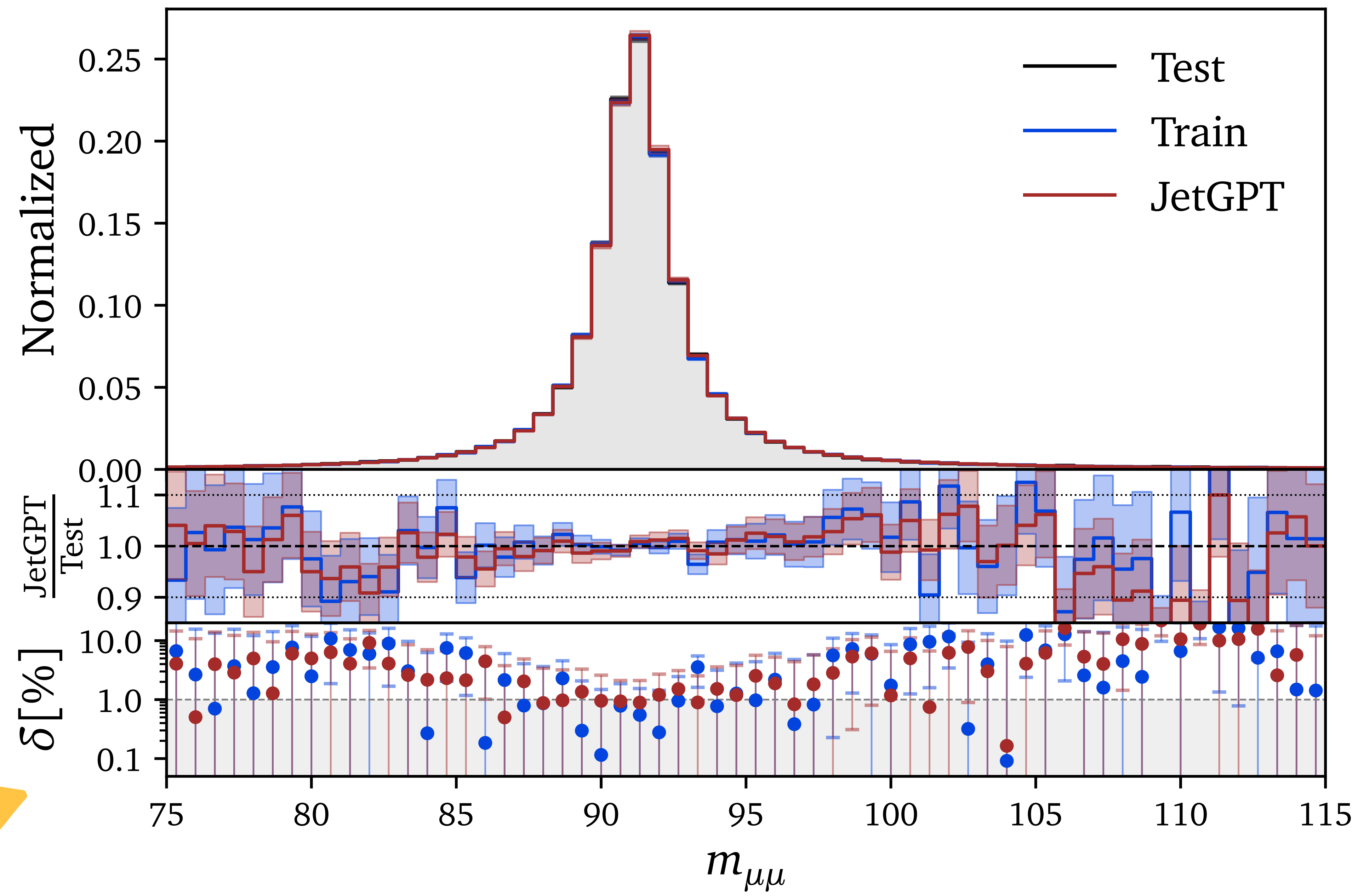
Generating LHC Events

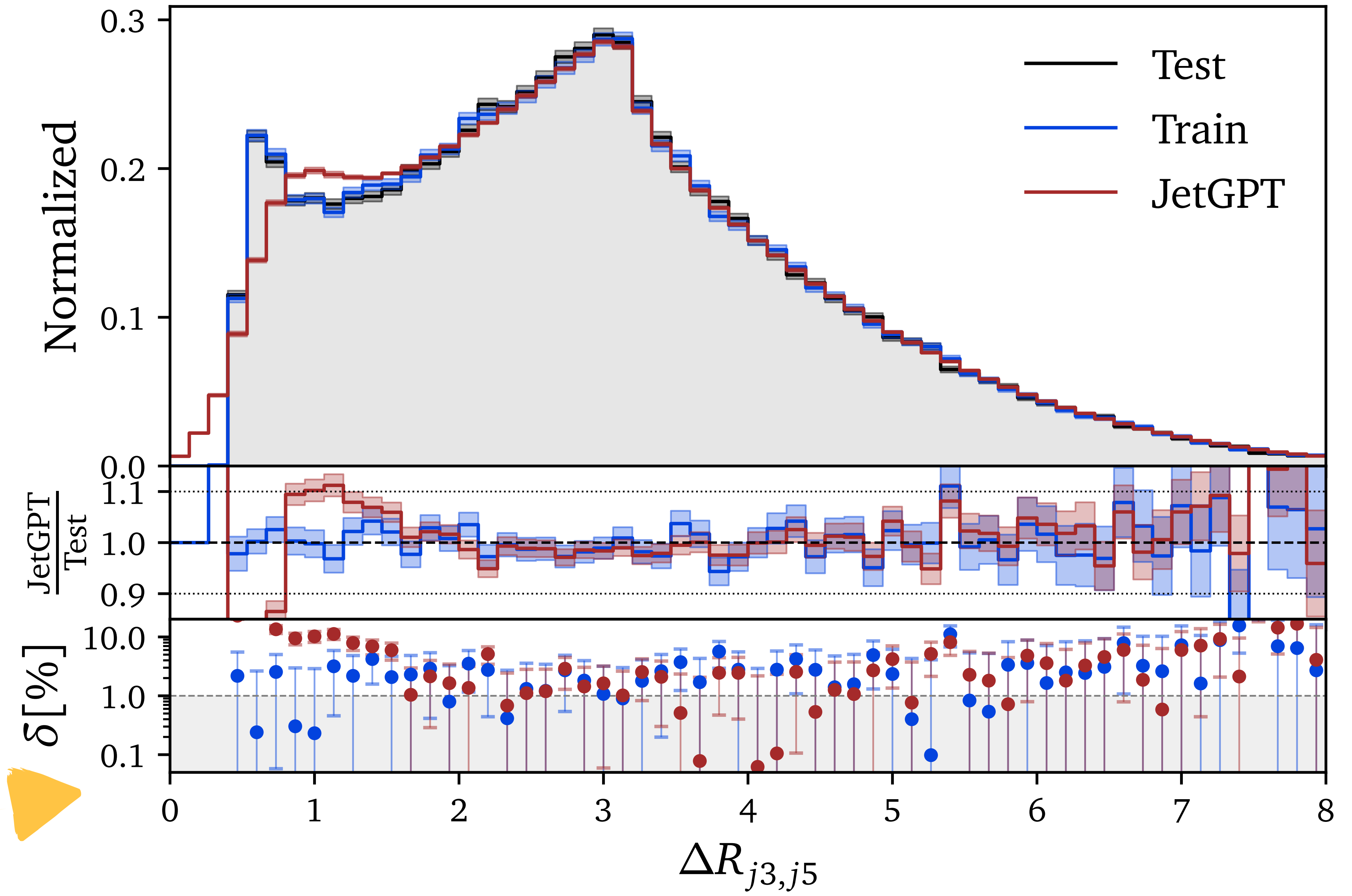
Generating LHC Events

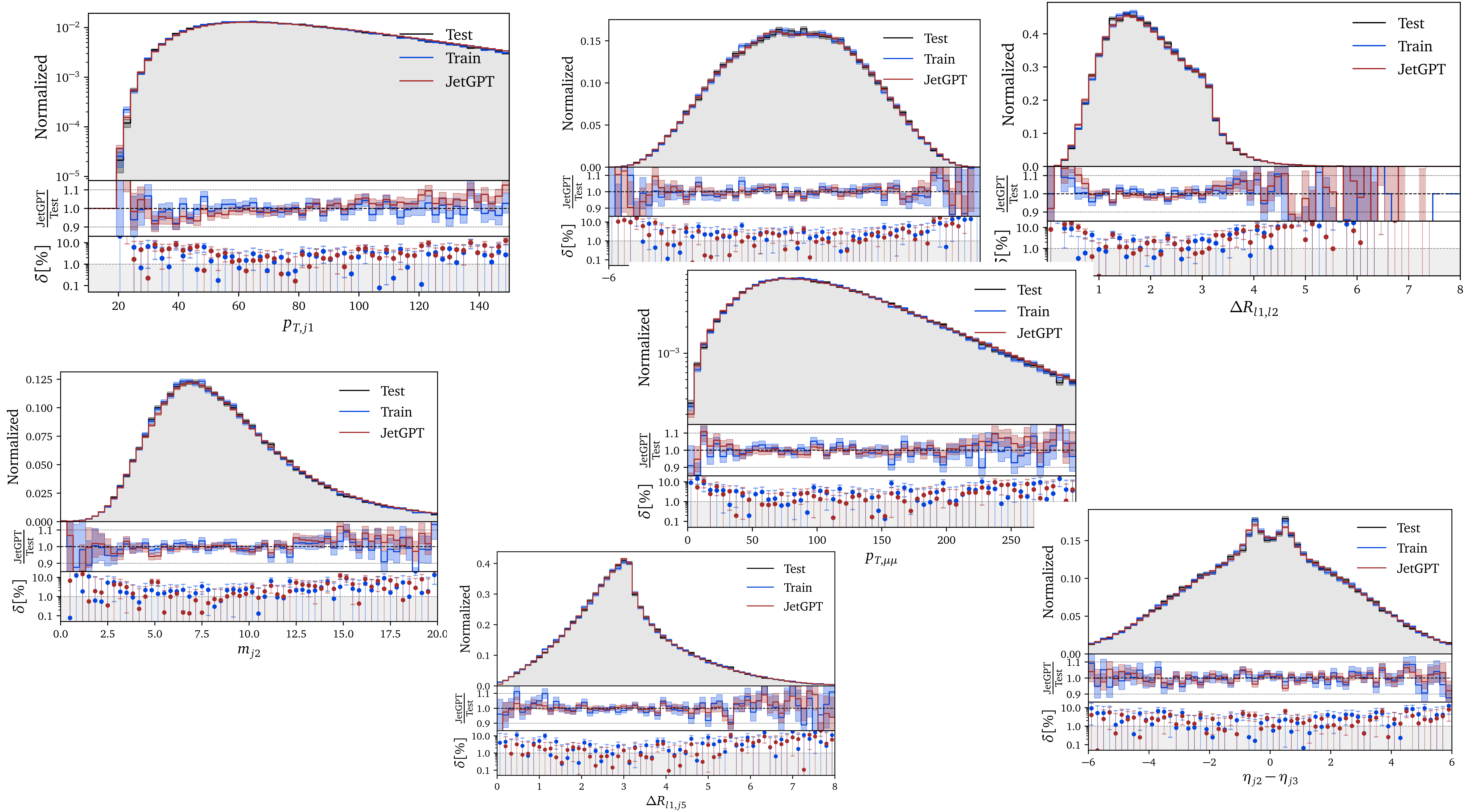
Dataset: $Z(\mu\mu) + \text{jets}$

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

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



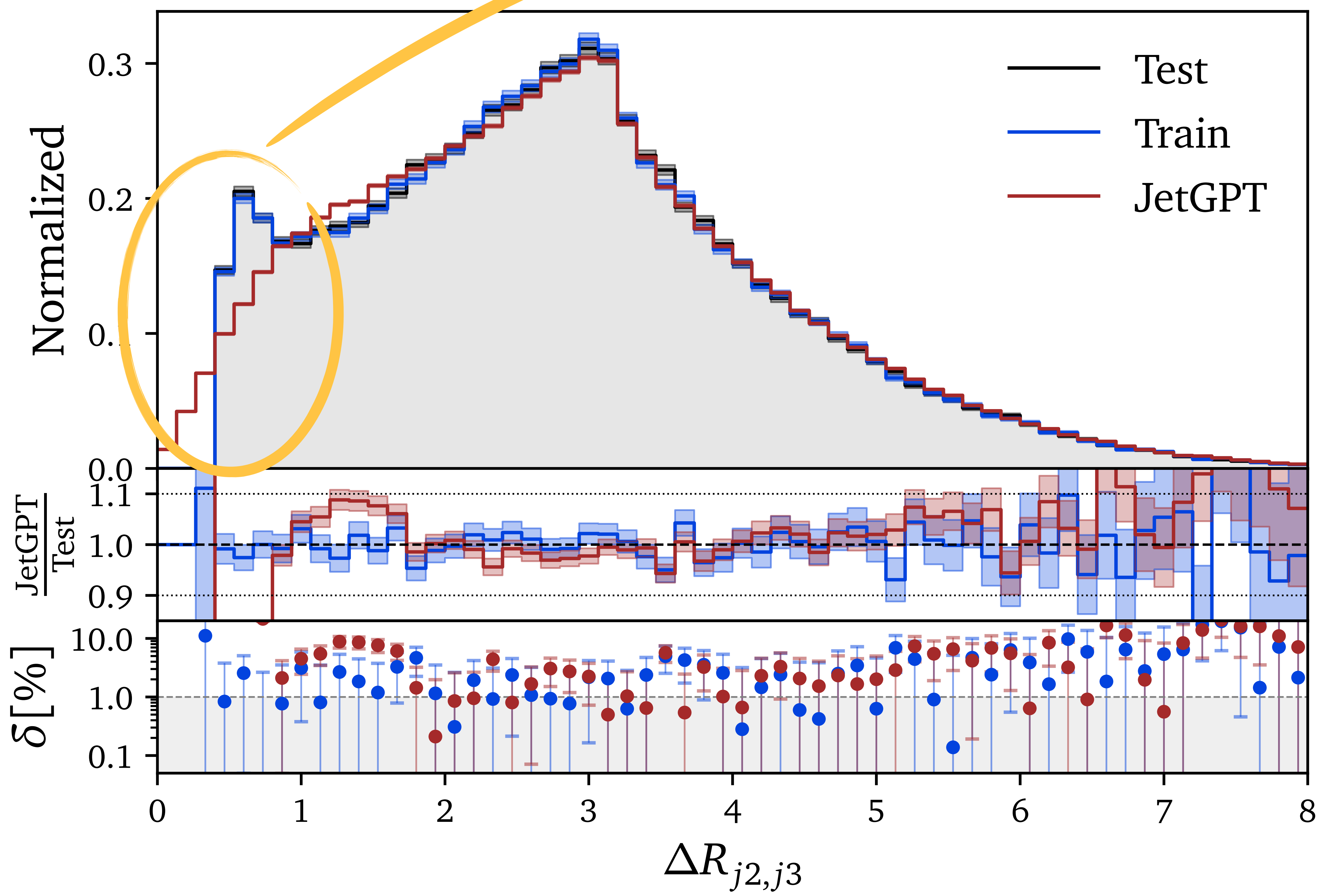



Test
Train
JetGPT
Normalized
JetGPT/Test
δ[%]
$p_{T,j1}$
$\Delta R_{l1,l2}$
$p_{T,\mu\mu}$
m_{j2}
$\Delta R_{l1,j5}$
$\eta_{j2} - \eta_{j3}$

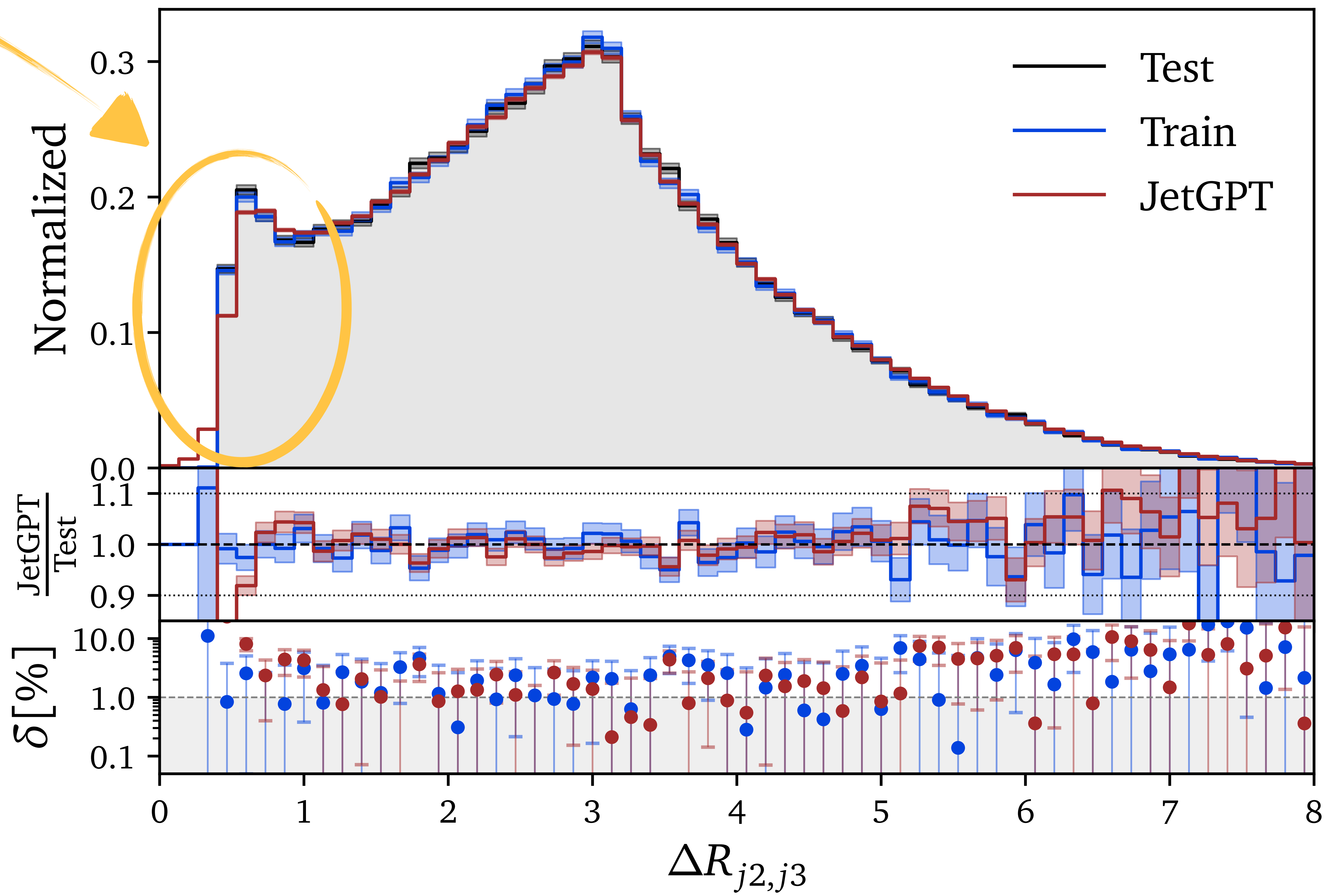
Generating LHC Events

Joint Training helps

Naive Training (5j only)



Joint Training (3j-5j)



Classifier Reweighting

Classifier Reweighting

Likelihood Ratio Trick

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

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

Equations of Motion

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

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

Classification

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

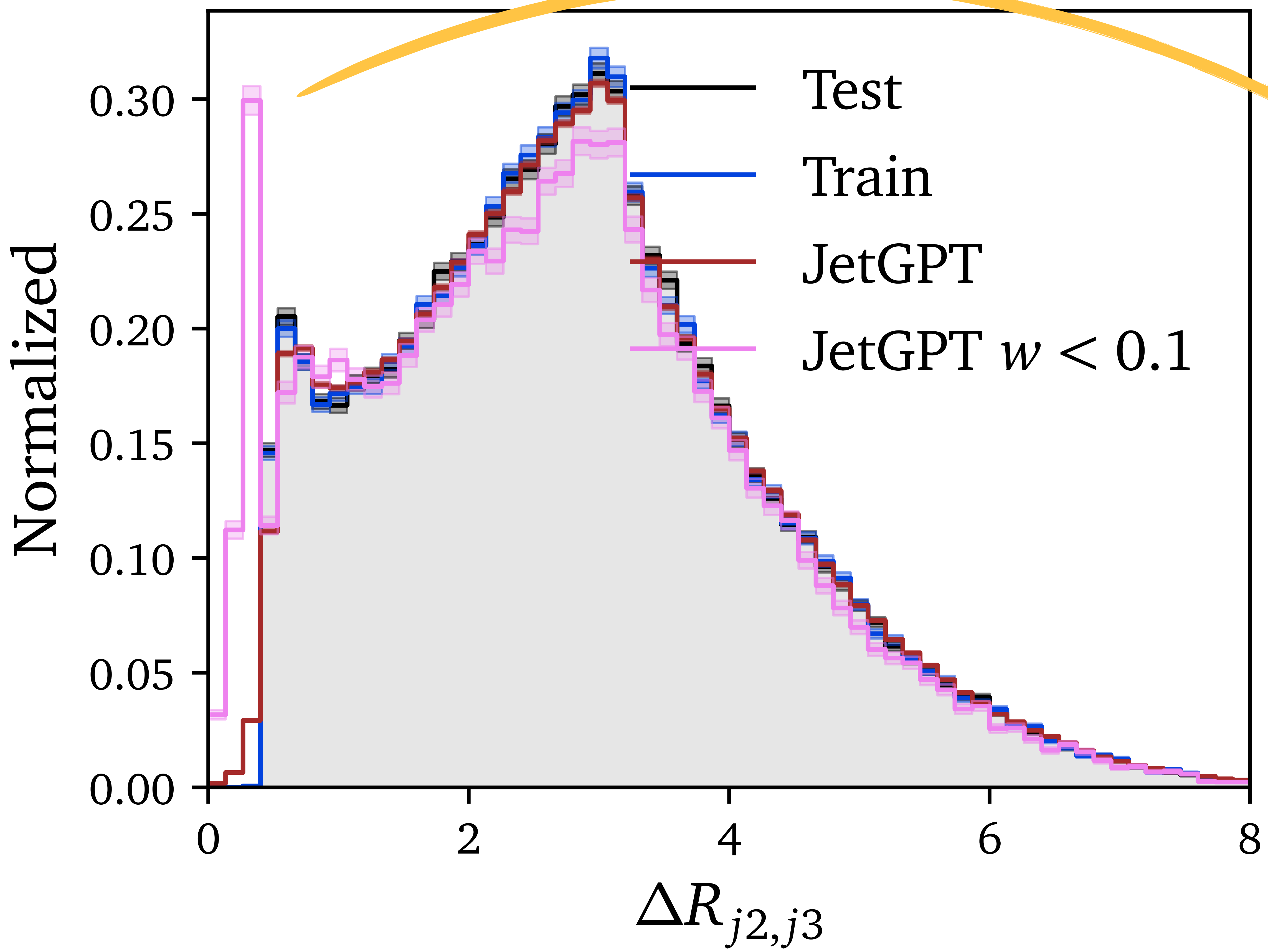
Reweighting

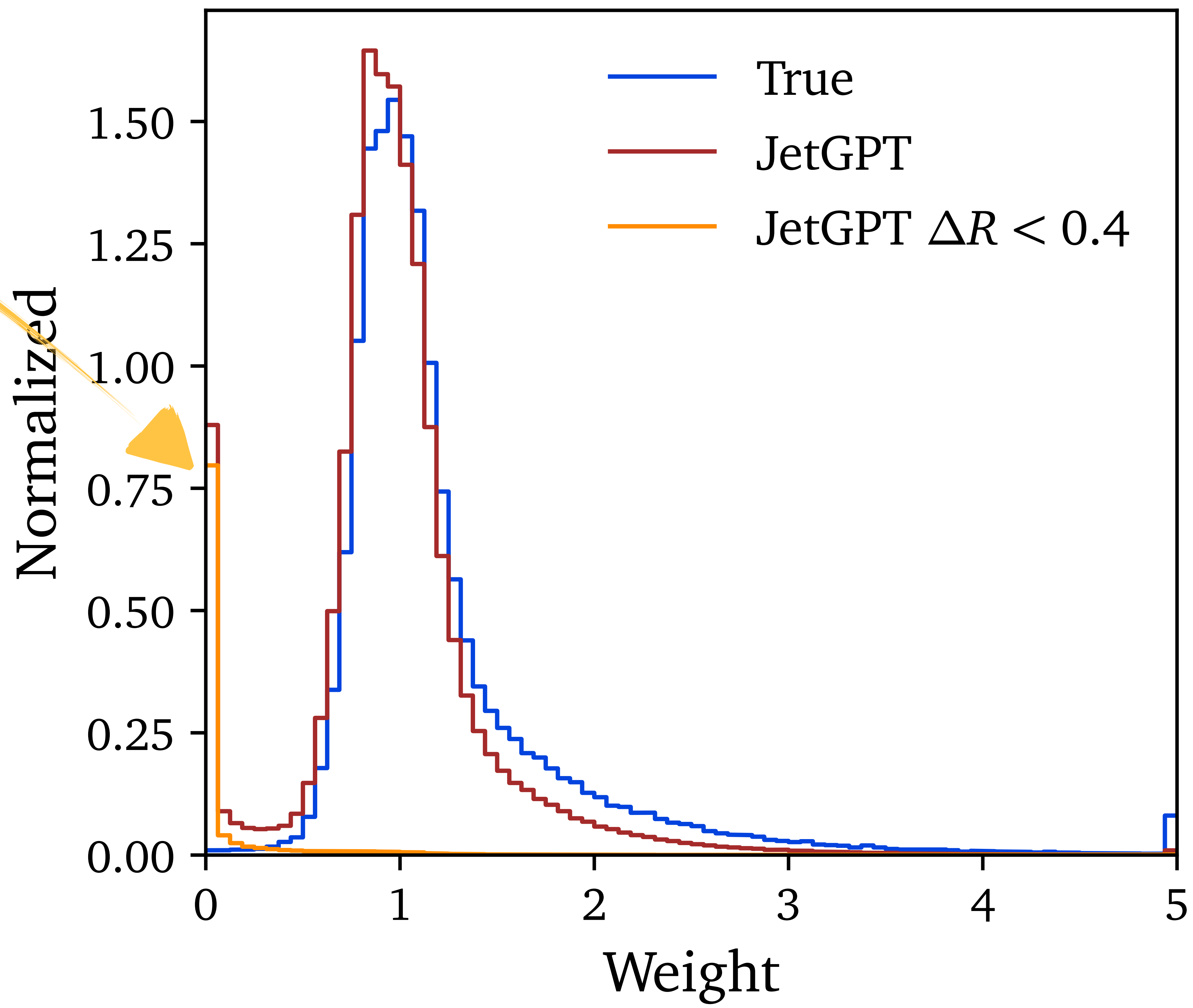
$$p_{\mathrm{data}} = p_{\mathrm{model}} \times \frac{p_{\mathrm{data}}}{p_{\mathrm{model}}}$$

Classifier Reweighting

Track the limitations

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



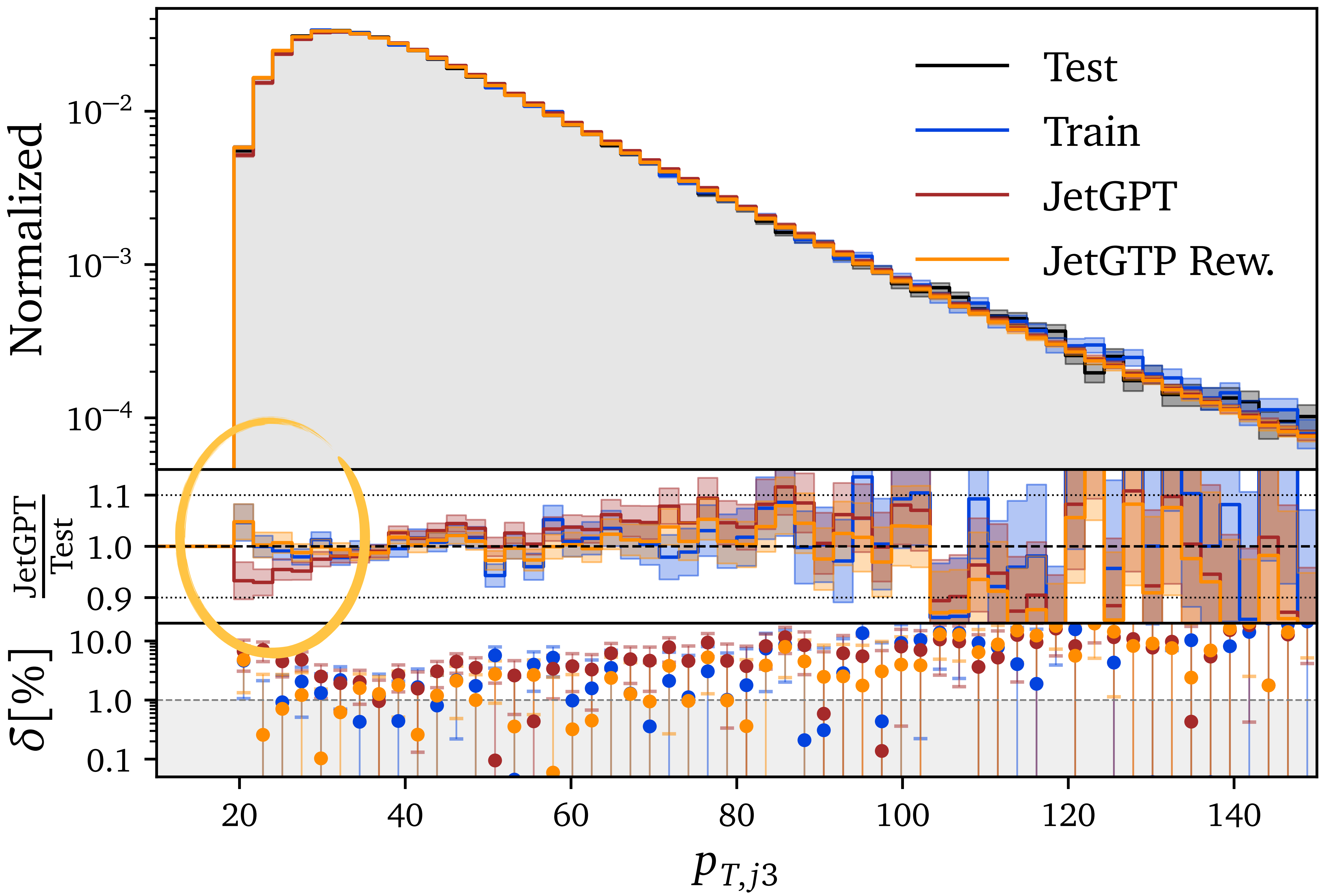


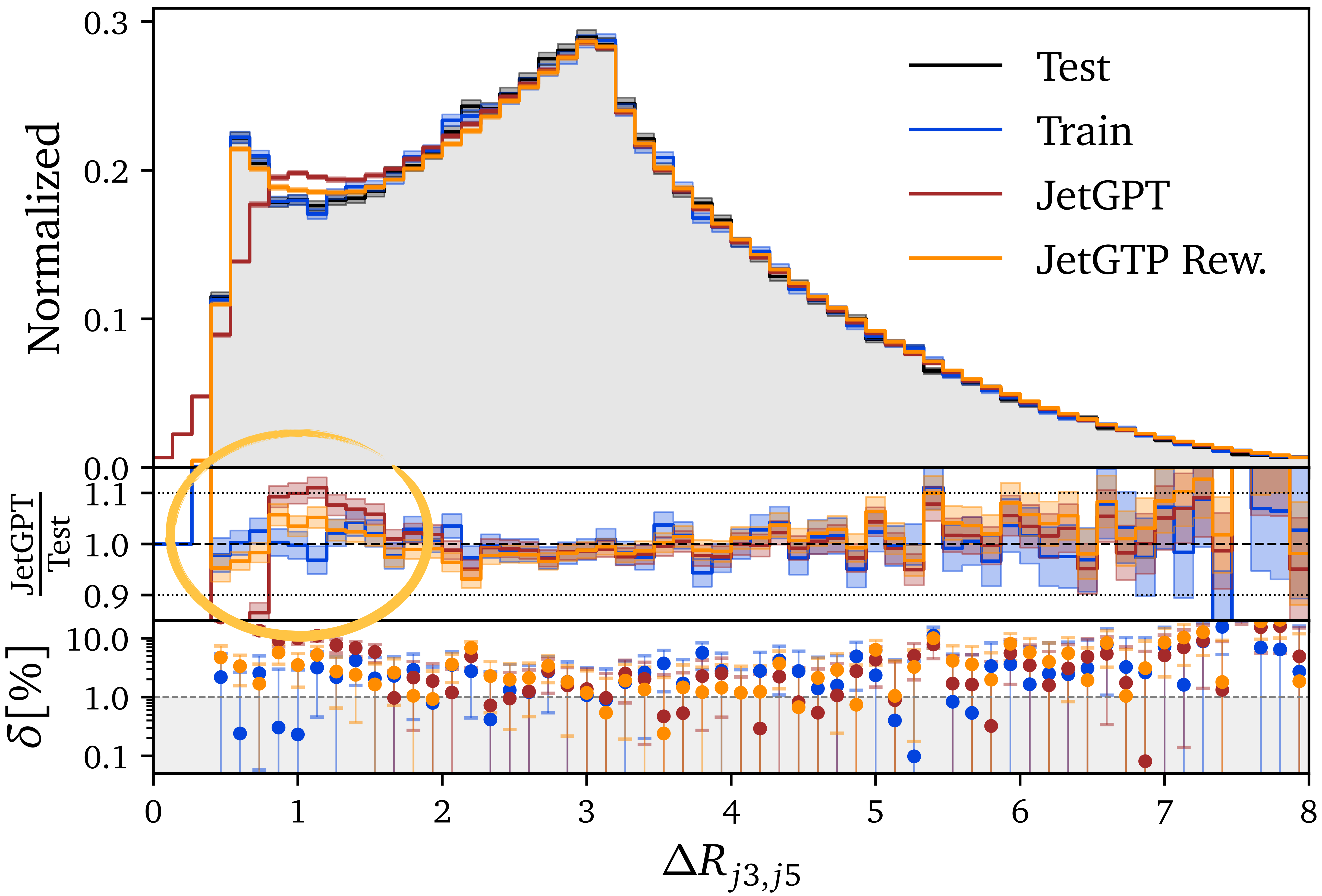
Classifier Reweighting

Overcome the limitations

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

Generator + Classifier





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

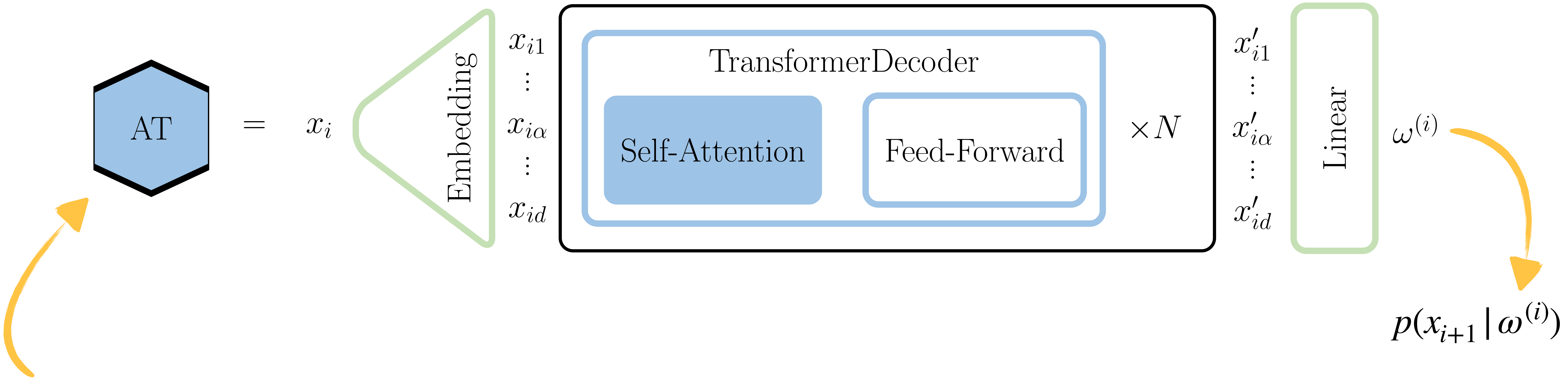
Backup

Autoregressive Transformer

Transformer Architecture

Autoregressive Transformer

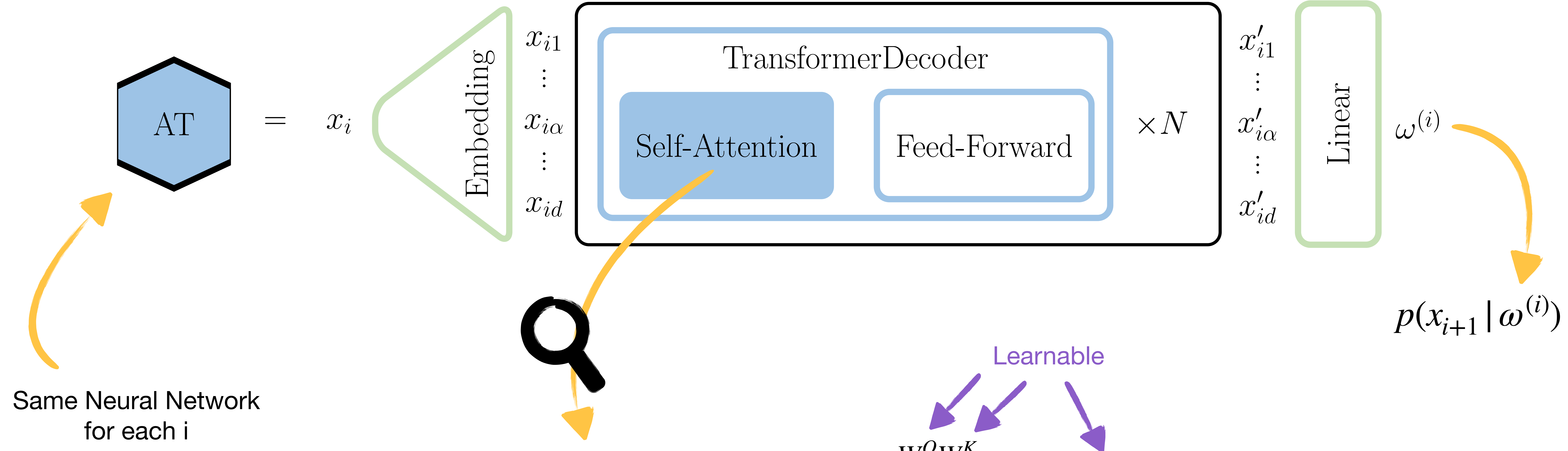
Transformer Architecture



Same Neural Network
for each i

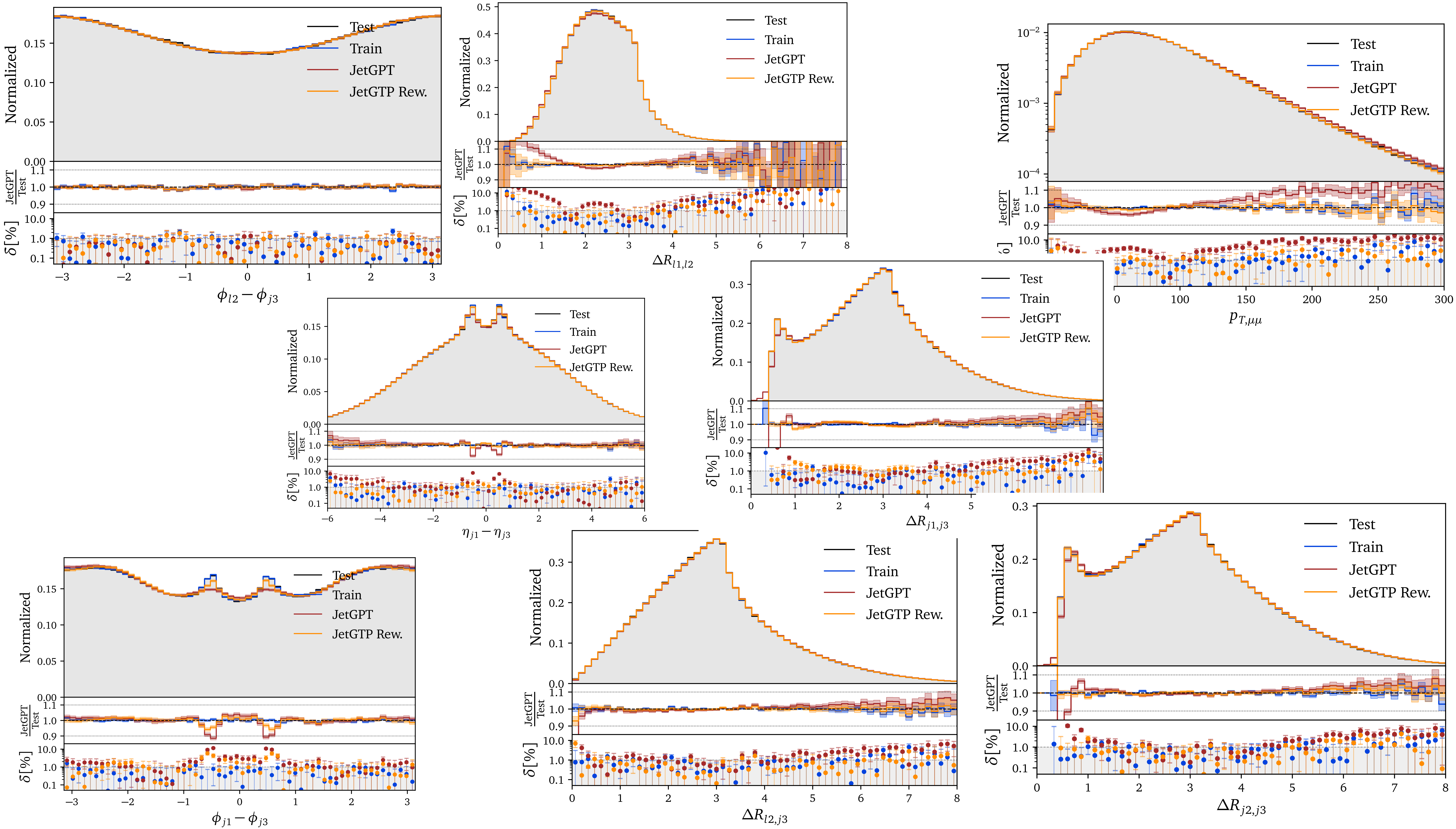
Autoregressive Transformer

Transformer Architecture



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

$$= \quad \underbrace{A_{ij}(x)}_{\text{Feature Space}} \quad \underbrace{W^V_{\alpha\beta}}_{\text{Latent Space}} \quad x_{j\beta}$$



Test
Train
JetGPT
JetGTP Rew.
Normalized
JetGPT / Test
δ[%]
$\phi_{l2} - \phi_{j3}$
$\Delta R_{l1,l2}$
$p_{T,\mu\mu}$
$\eta_{j1} - \eta_{j3}$
$\Delta R_{j1,j3}$
$\phi_{j1} - \phi_{j3}$
$\Delta R_{l2,j3}$
$\Delta R_{j2,j3}$