DeGeSim – Pileup Modelling using AI

10-11-23

Stephen Jiggins - DESY

Brief

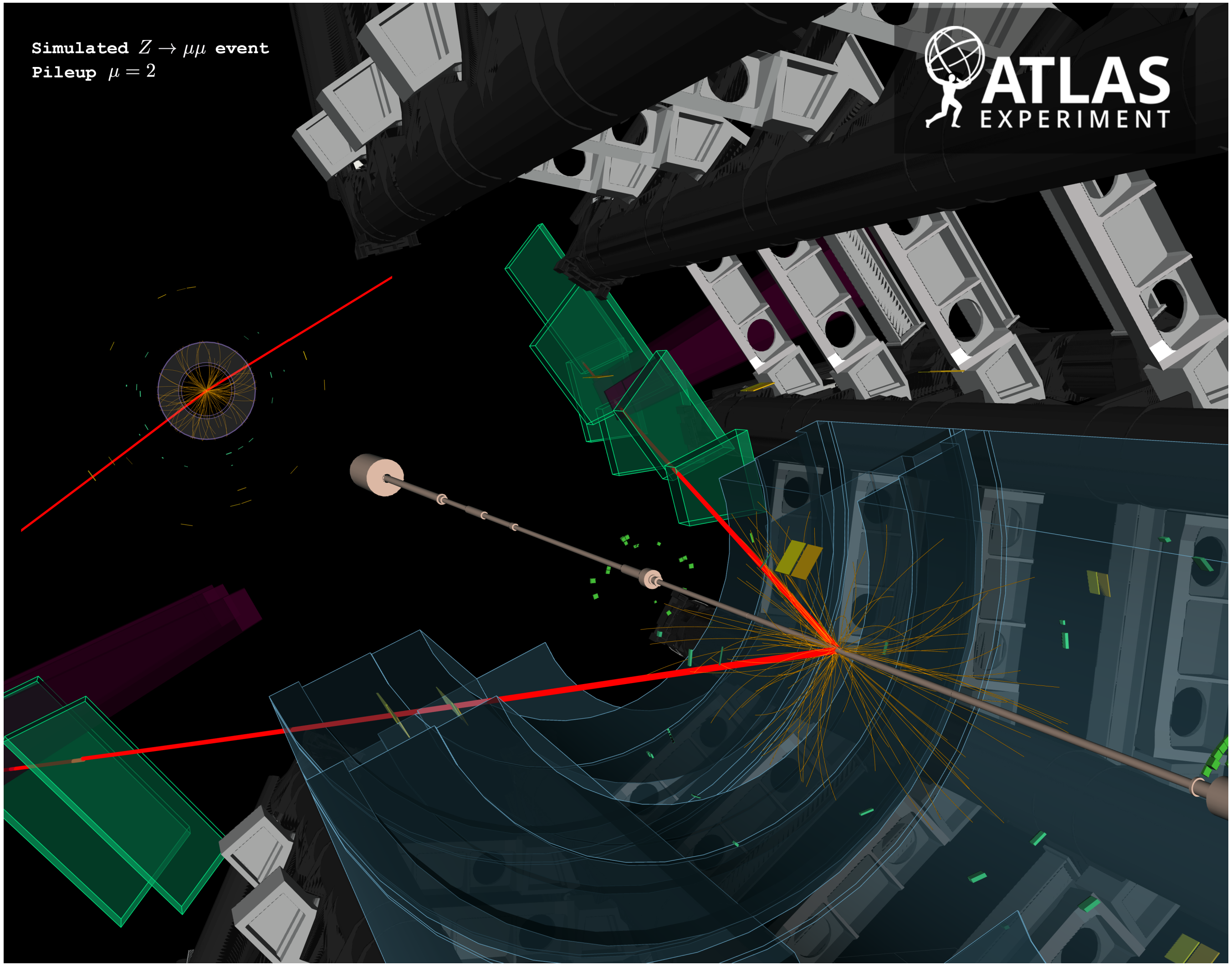
Problem:

Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:

Brief

Problem:

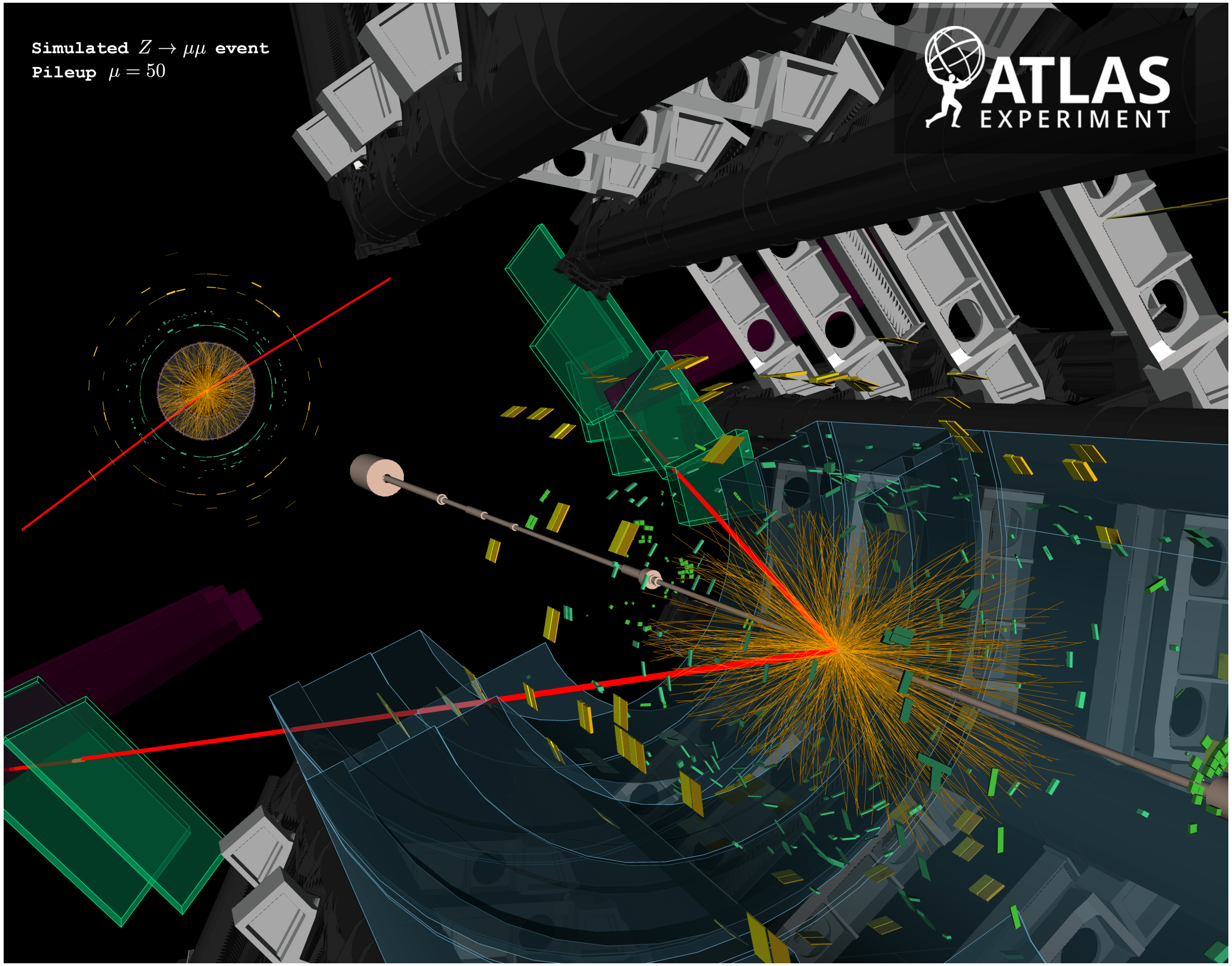
Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:



Brief

Problem:

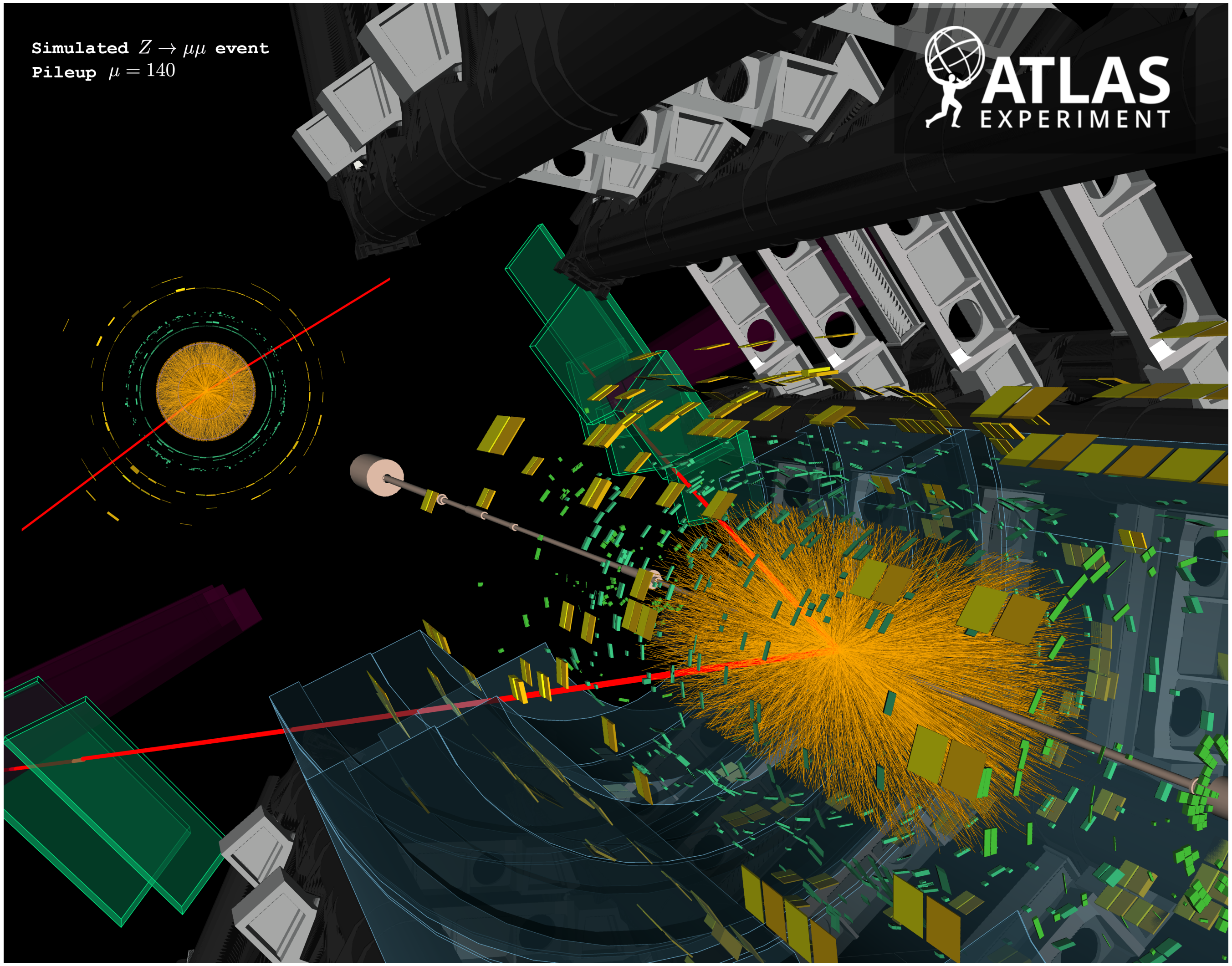
Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:



Brief

Problem:

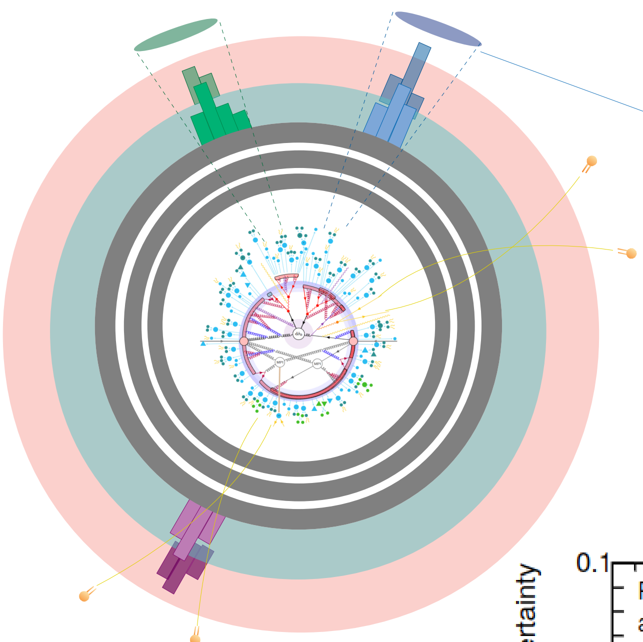
Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:



Brief

Problem:

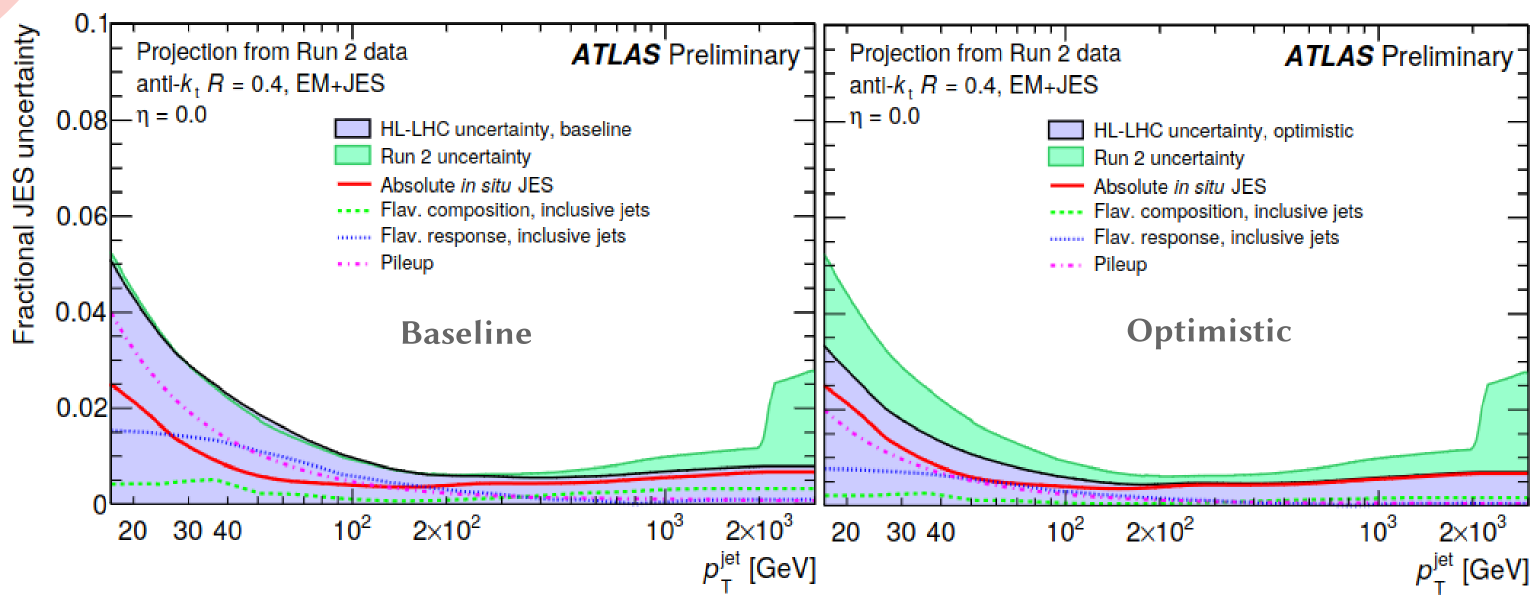
Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:



⟶ HL-LHC jet performance projections for ATLAS 1902.10229:

Baseline ~ Run-2

Optimistic: Requires improved understanding of jet fragmentation & pile-up



Source: 1902.10229

Introduction

Project Aim:
Generative modelling of pileup interactions using zero-bias data instead of MC simulations using ML image synthesis techniques

DeGeSim:
Helmholtz AI funded project for simulating detector simulations in proton-proton colliders using deep generative ML models as joint project with CMS + Juelich + TRIUMF:

HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

ATLAS (DESY)


CMS (DESY)



Juelich

TRIUMF

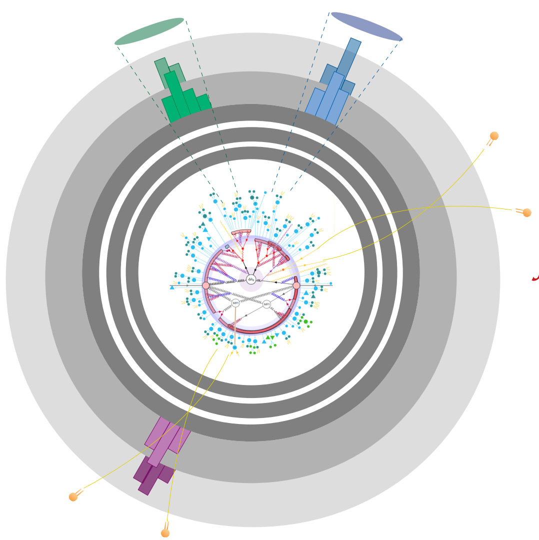
Introduction

Project Aim:

Generative modelling of pileup interactions using zero-bias data instead of MC simulations using ML image synthesis techniques

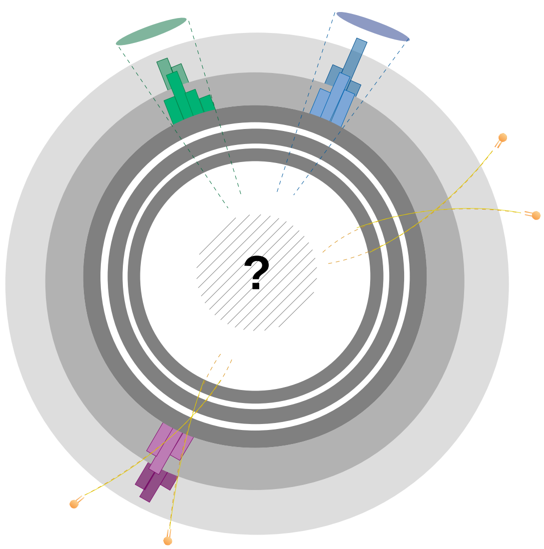
Monte Carlo (MC)

EPOS+Pythia8 simulated
soft-QCD



$f\colon \boldsymbol{X}_{MC} \longrightarrow \boldsymbol{X}_{D}$

Zero-Bias Data



DeGeSim:

Helmholtz AI funded project for simulating detector simulations in proton-proton colliders using deep generative ML models as joint project with CMS + Juelich + TRIUMF:

HELMHOLTZAI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

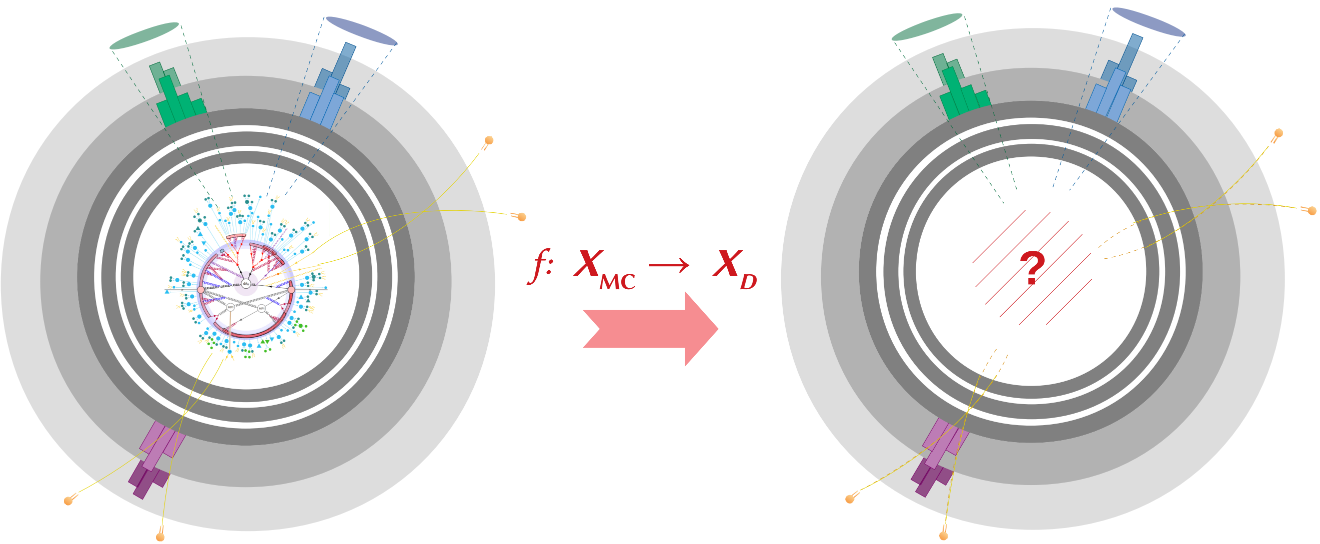
ATLAS (DESY)

Introduction

Project Aim:
Generative modelling of pileup interactions using zero-bias data instead of MC simulations using ML image synthesis techniques

Monte Carlo (MC)
EPOS+Pythia8 simulated soft-QCD

Zero-Bias Data



Boundary Conditions:
In terms of the simulation chain the data domain is limited to the simulated ATLAS tracker + calorimeters + MS

DeGeSim:
Helmholtz AI funded project for simulating detector simulations in proton-proton colliders using deep generative ML models as joint project with CMS + Juelich + TRIUMF:

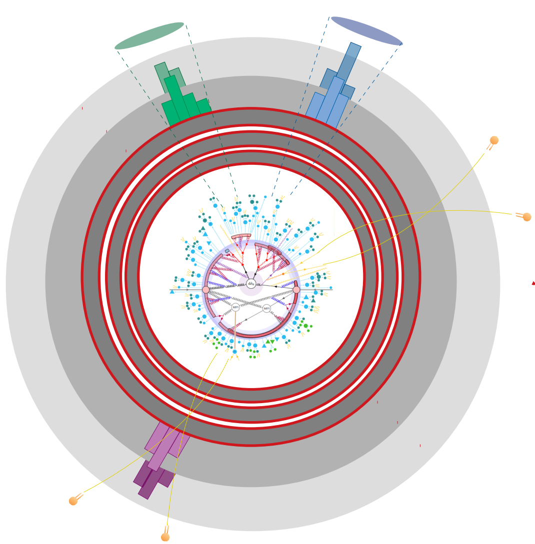
HELMHOLTZAI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

ATLAS (DESY)

Introduction

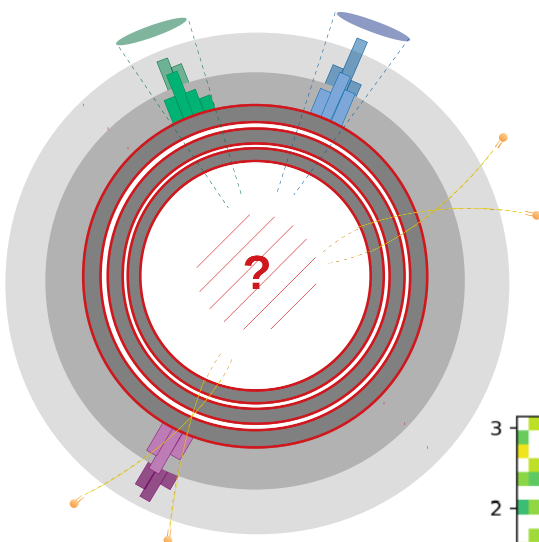
Project Aim:
Generative modelling of pileup interactions using zero-bias data instead of MC simulations using ML image synthesis techniques

Monte Carlo (MC)
EPOS+Pythia8 simulated soft-QCD



$f\colon X_{MC} \rightarrow X_D$

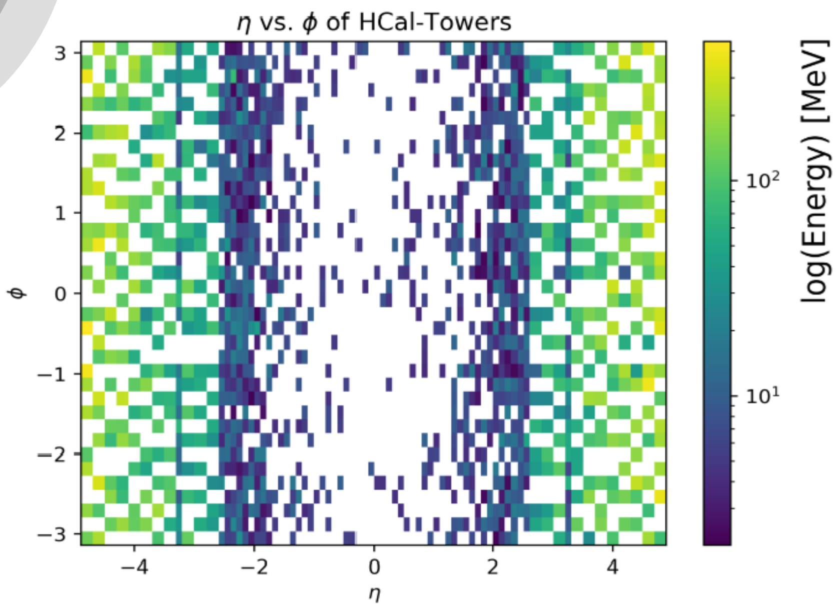
Zero-Bias Data



DeGeSim:
Helmholtz AI funded project for simulating detector simulations in proton-proton colliders using deep generative ML models as joint project with CMS + Juelich + TRIUMF:

HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

ATLAS (DESY)

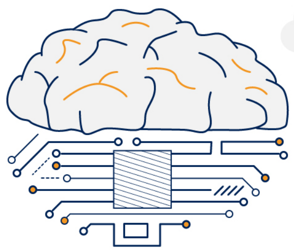


Boundary Conditions:
In terms of the simulation chain the data domain is limited to the simulated ATLAS tracker + calorimeters + MS

Introduction

Project Aim:

Generative modelling of pileup interactions using zero-bias data instead of MC simulations using ML image synthesis techniques



Which model?

DeGeSim:

Helmholtz AI funded project for simulating detector simulations in proton-proton colliders using deep generative ML models as joint project with CMS + Juelich + TRIUMF:

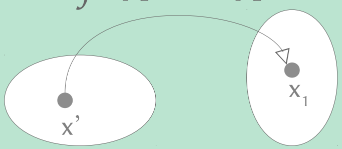
HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

ATLAS (DESY)

Fully Generative

*Sample from a known function $P_s(\overline{x}' \,|\, \overline{\theta})$ to get **X'** and then find a transformation from **X'** to **X**:*

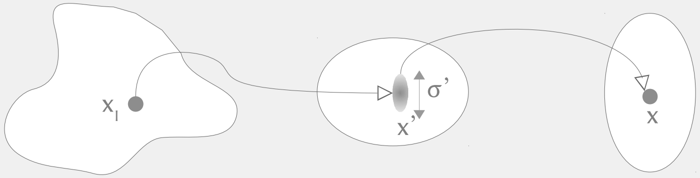
$$f\colon \boldsymbol{X'} \rightarrow \boldsymbol{X}$$



Seeded Generative

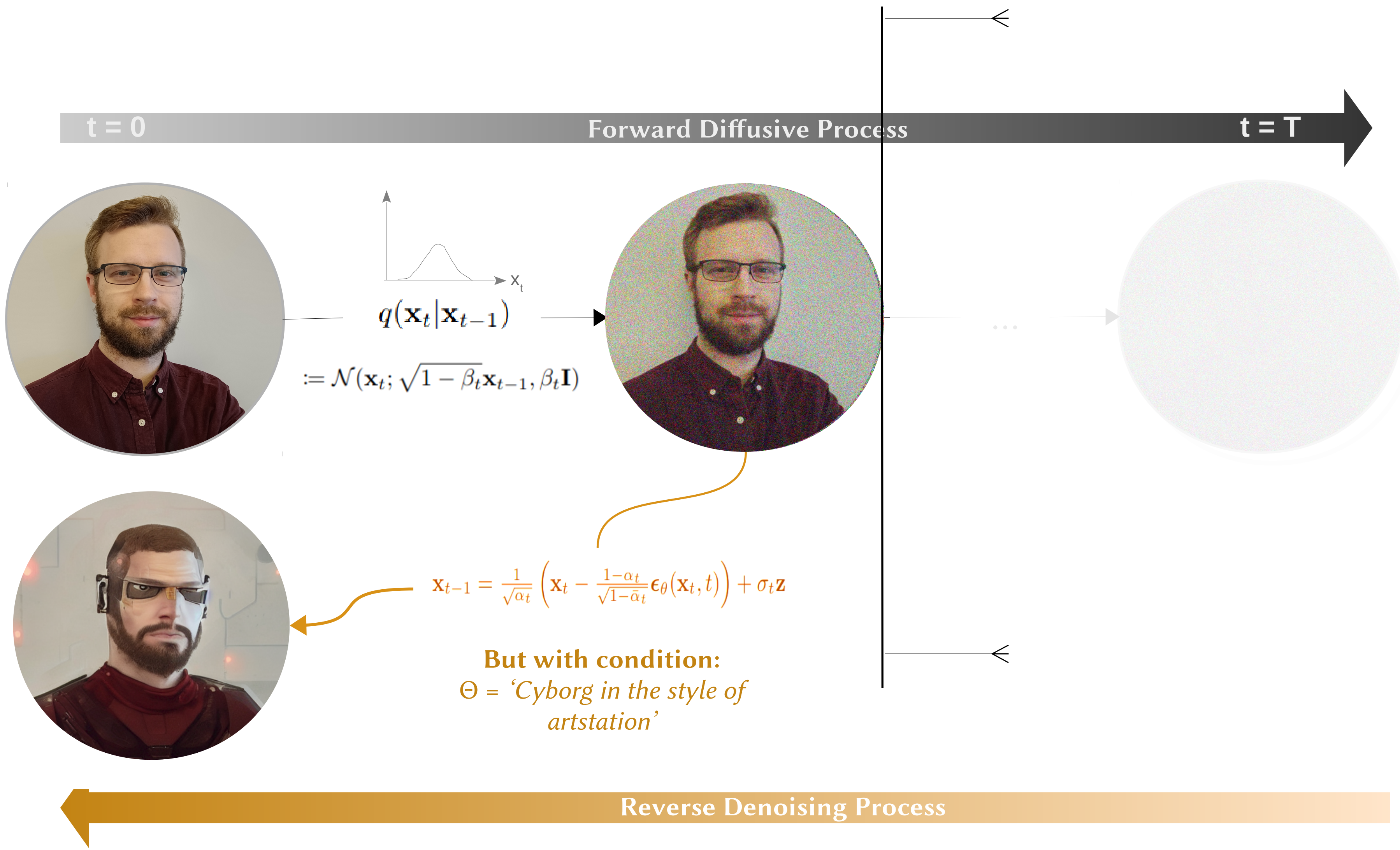
*Transform from a sampled point $\boldsymbol{X_I}$ of a complex function $P_I(\overline{x} \,|\, \overline{\theta})$ and find a transformation from $\boldsymbol{X_I}$ to **X**:*

$$f\colon \boldsymbol{X_I} \rightarrow \boldsymbol{X}$$

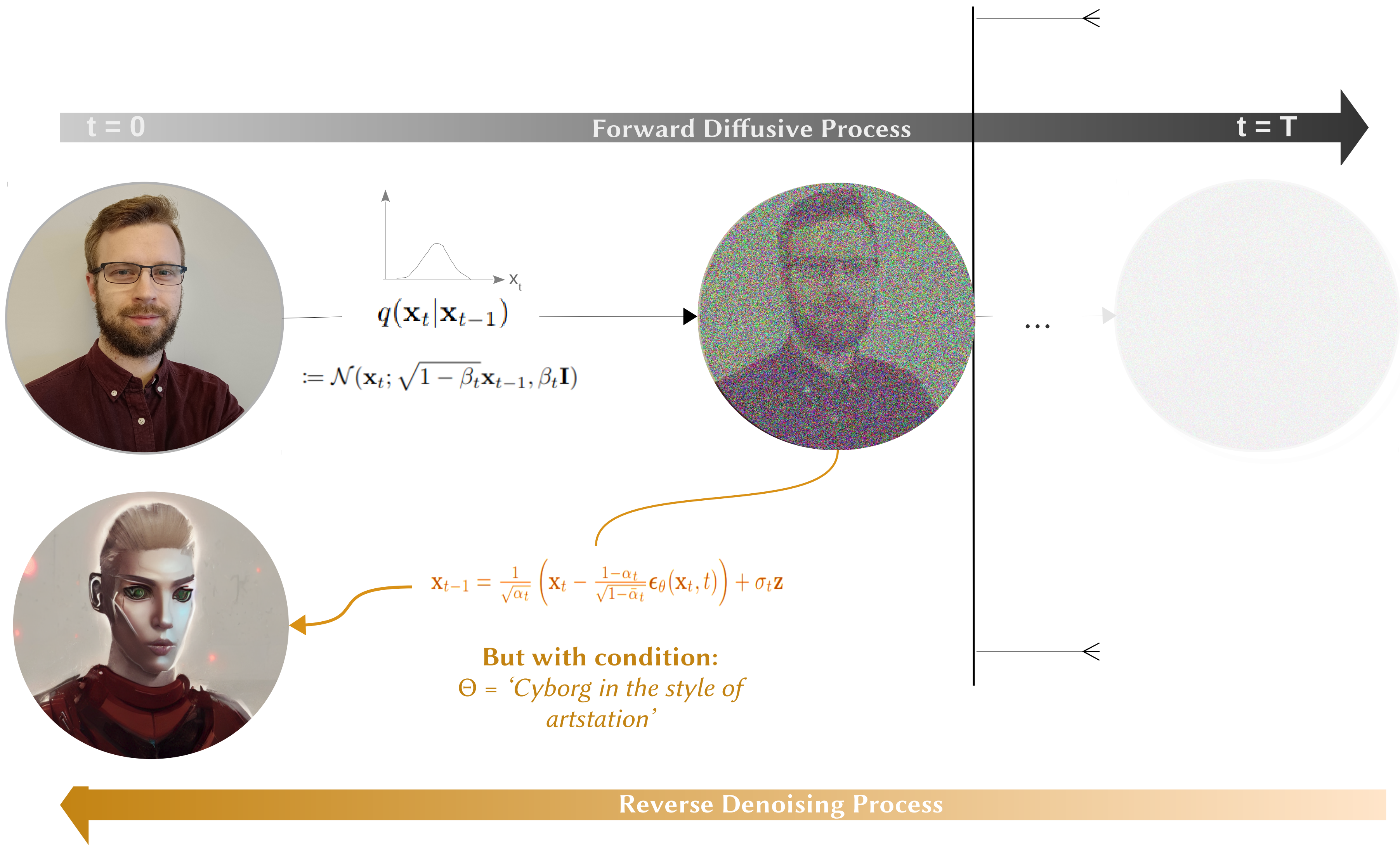


Denoising Diffusion Probabilistic Models

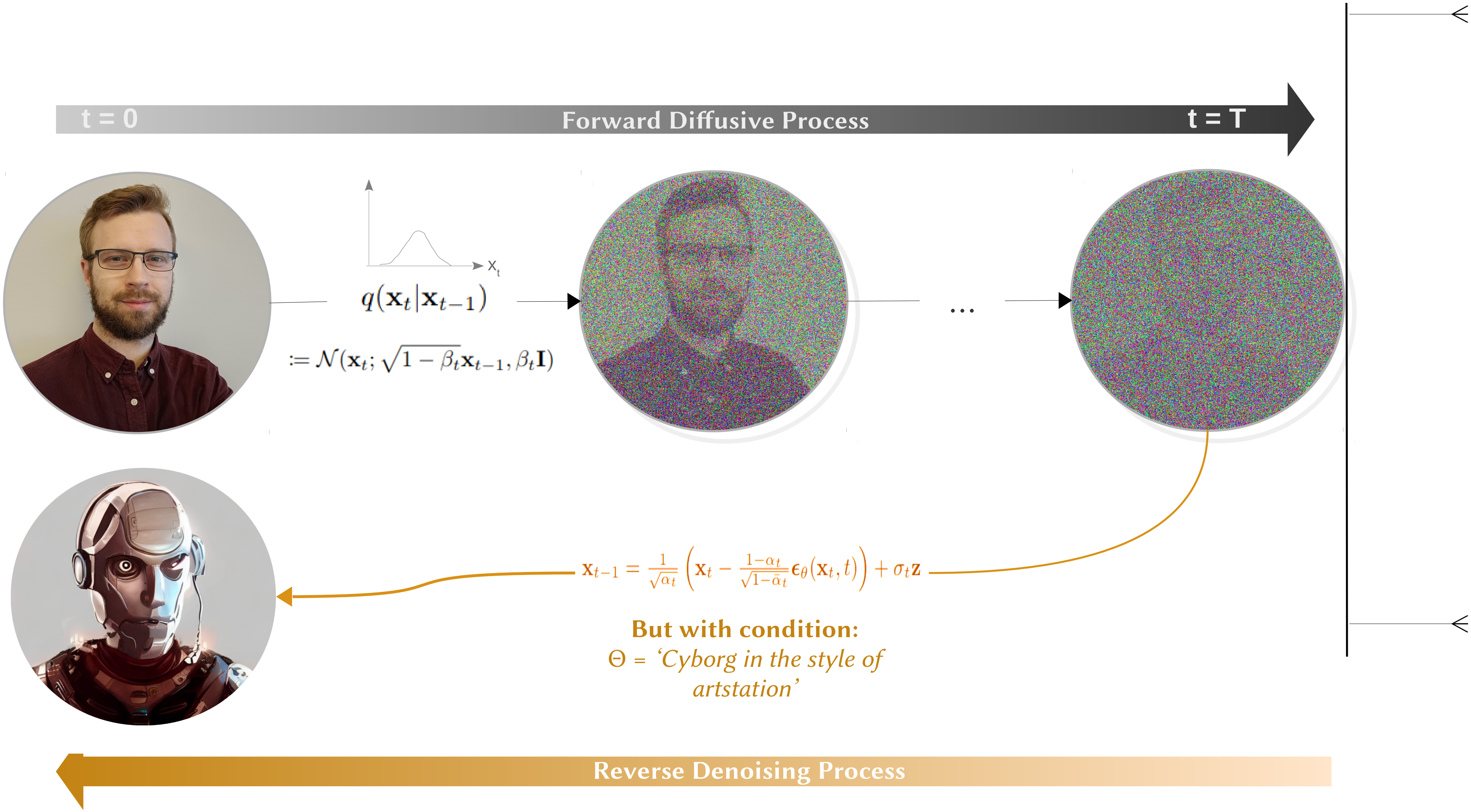
Generative DDPM



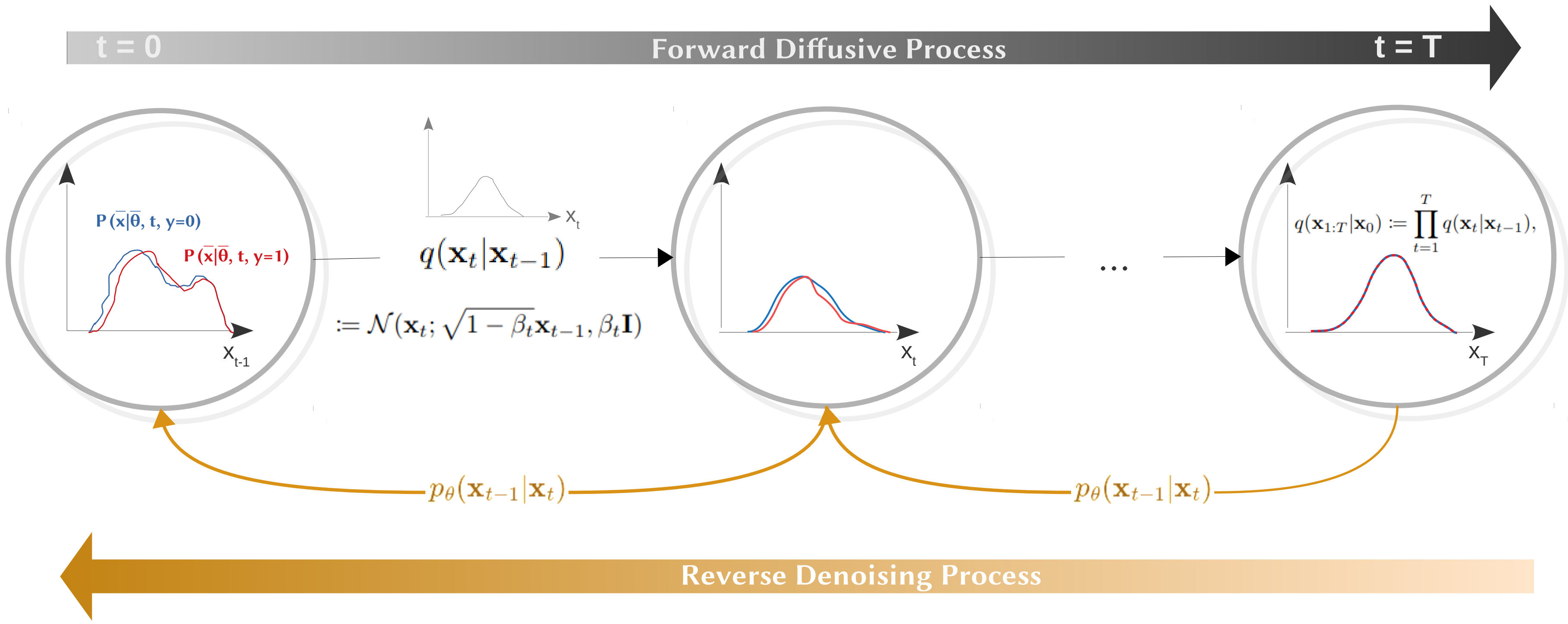
Generative DDPM



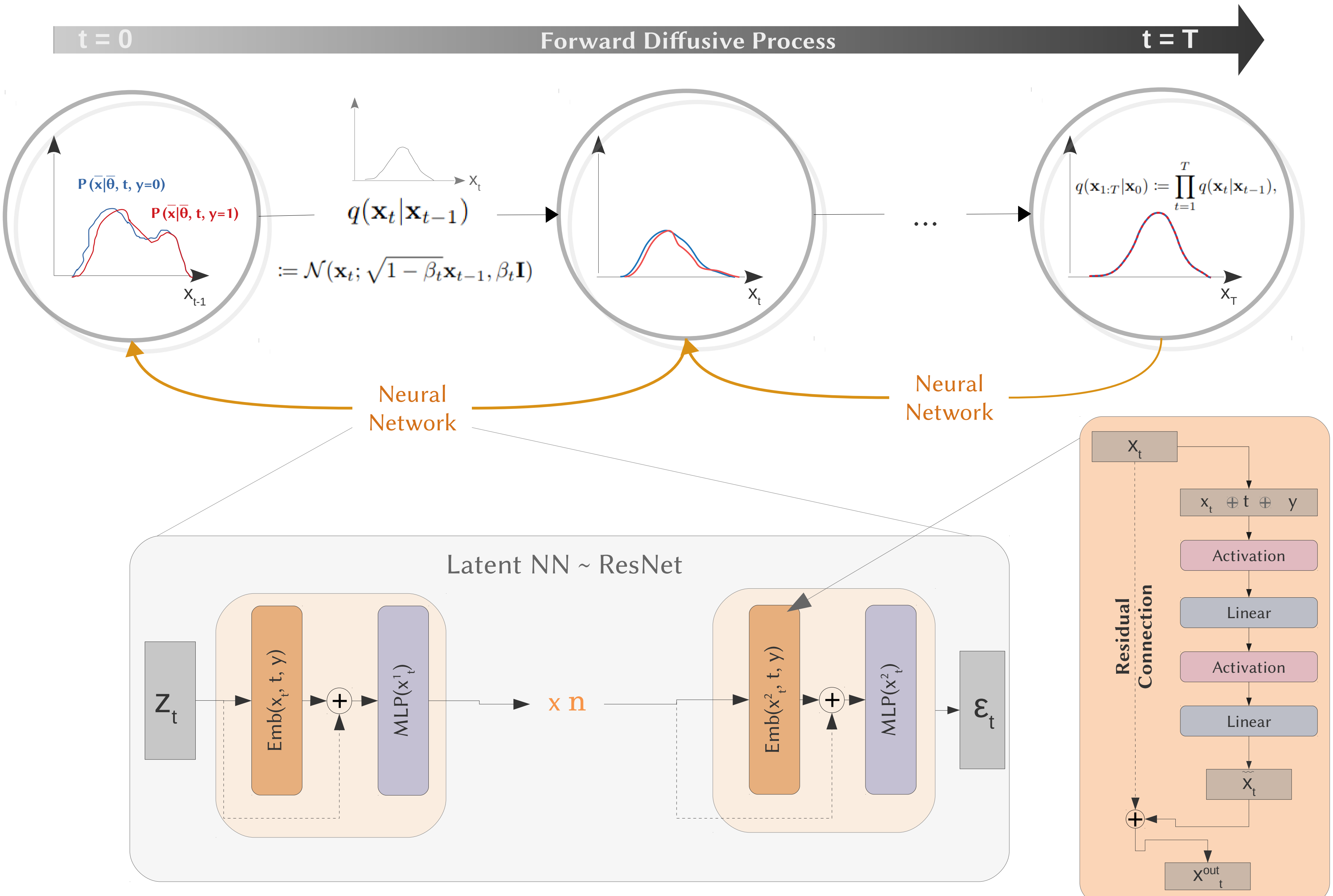
Generative DDPM



Generative DDPM



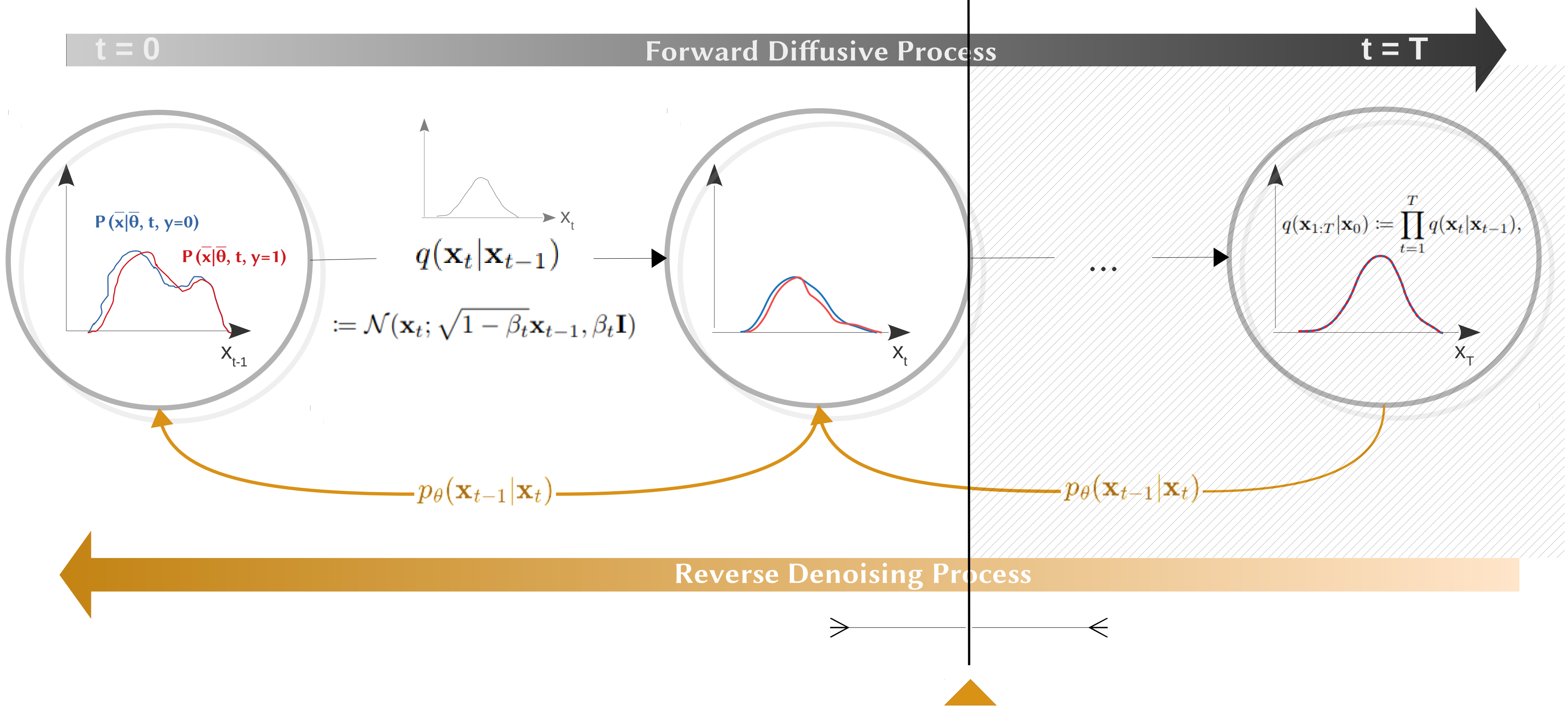
Generative DDPM



Generative DDPM

Partially diffuse the image to retain features from the original image!

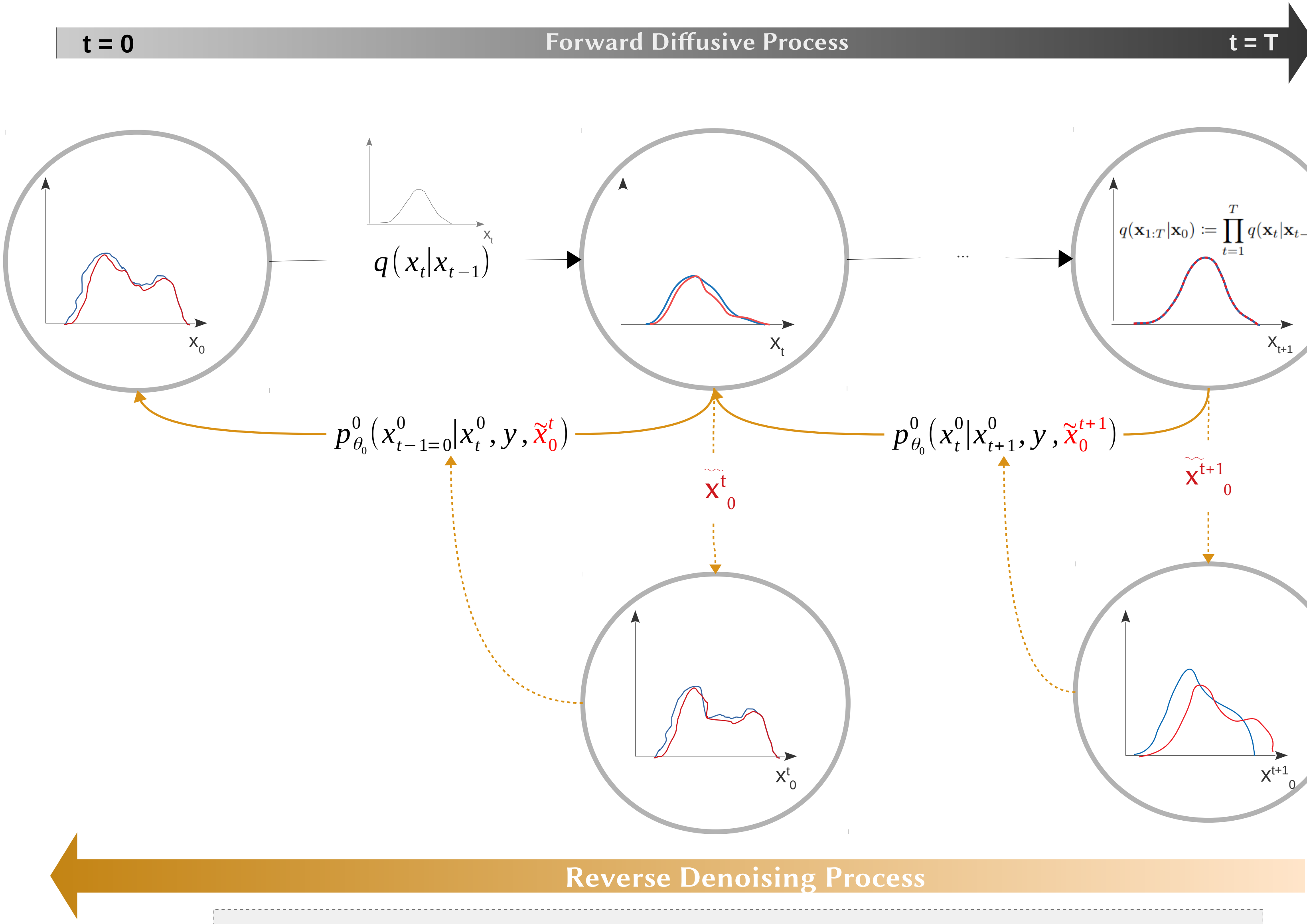
Stop the diffusive process when the two processes are **indistinguishable**



Approach 1

Train a conditional DDPM and partially diffuse a sample to a *release time* **t_r**. Then switch class label ($0 \longrightarrow 1$) and denoise.

Self-Conditioned DDPM



Approach 2

Train a conditional DDPM with concatenated predictions of the origin $\tilde{x}_0$ and partially diffuse a sample to a *release time* **t_r**. Then switch class label ($0 \longrightarrow 1$) and denoise.

JetNet Data Test

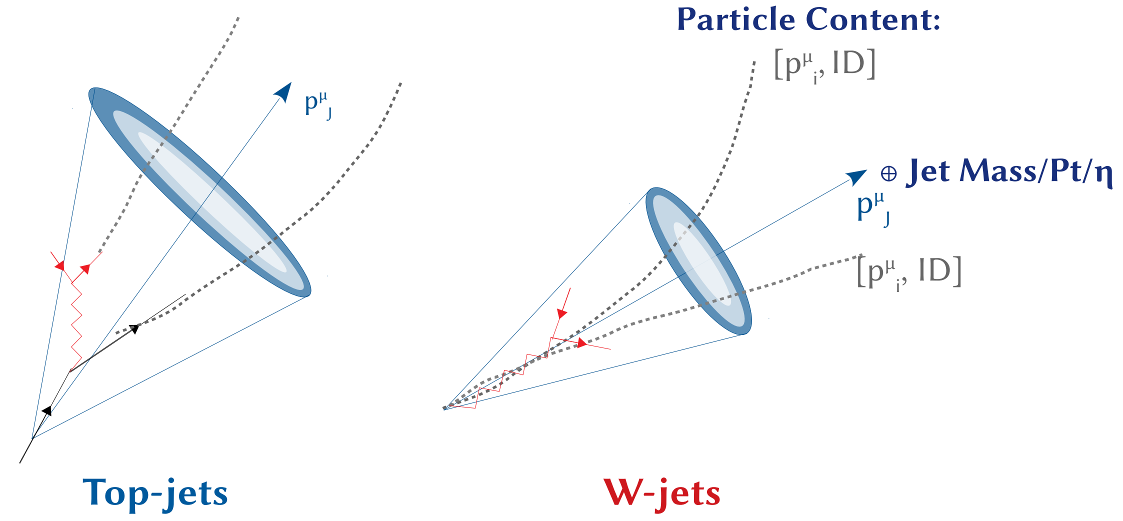
JetNet: Top -vs- W jets

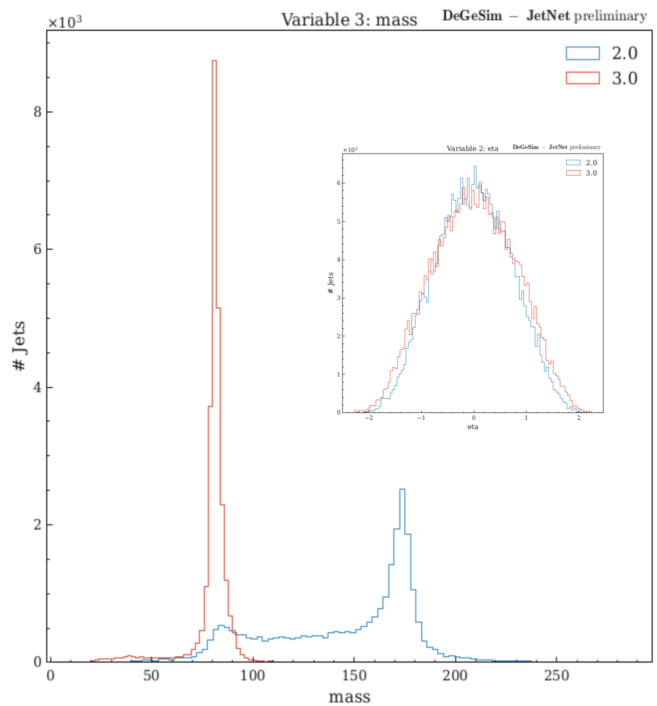
Data - JetNet

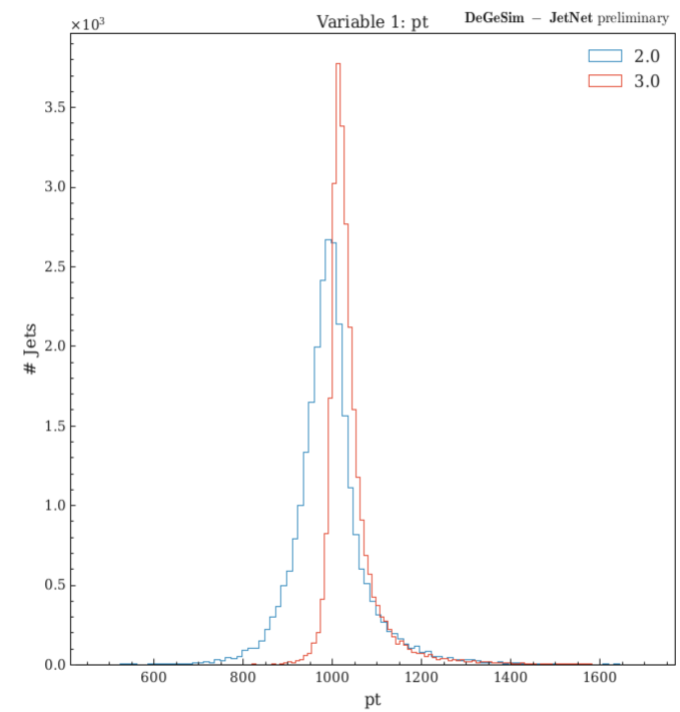
→ **Test using top vs W-jets**
W-jets are a sub-component of top-jets from the top-decay

→ **Implemented using JetNet Python API** - link

→ **Diffusion Model input domain defined by jet features & b-hadrons:**
- $p^\mu_J = [p_T, \eta, m_J]$
- *# of b-hadrons inside jet*







Results

JetNet: Top -vs- W jets

Key

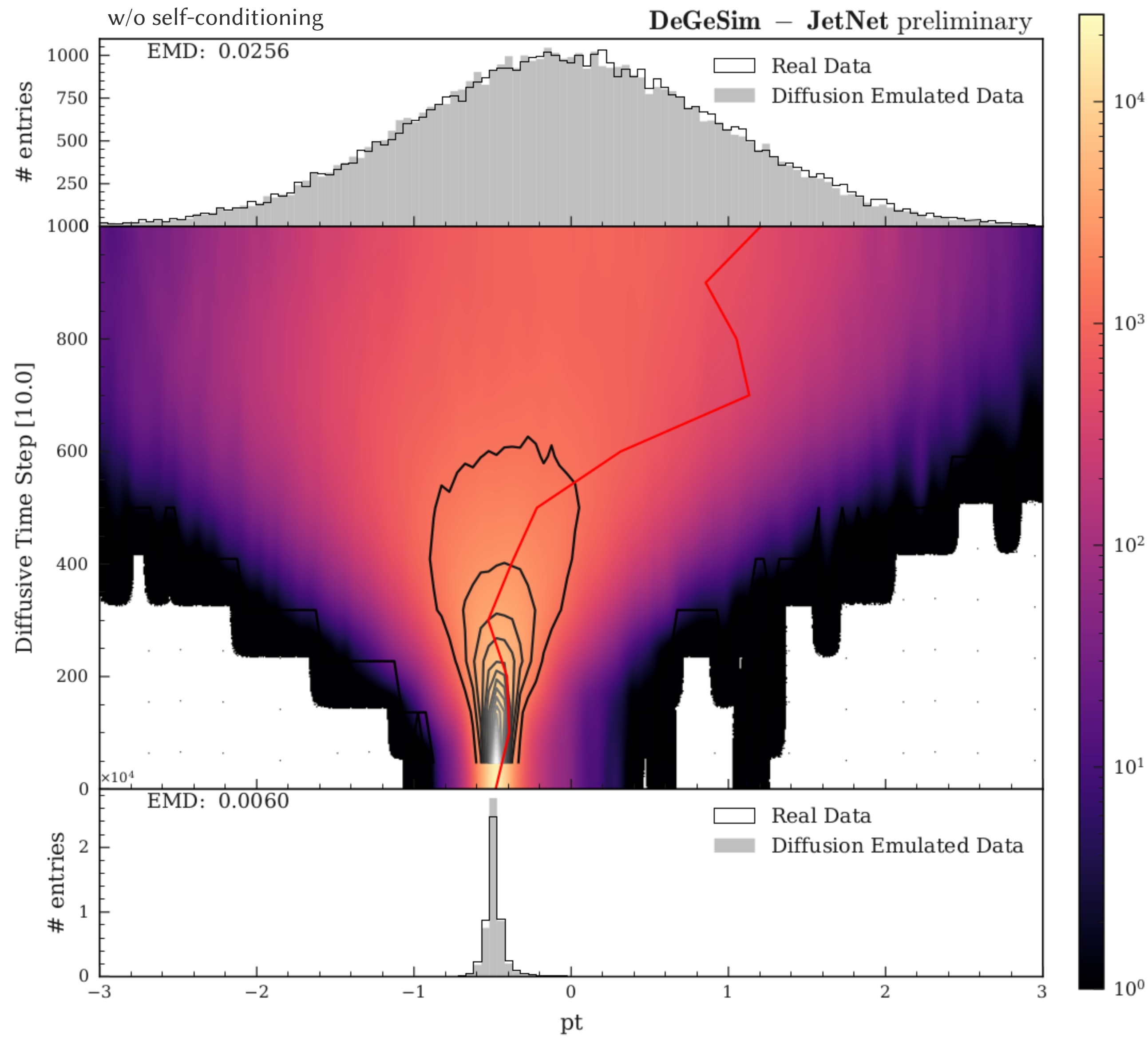
⟶ **Top Pane:** Gaussian emulated (Gen.) sampling with forward diffused *real data*

⟶ **Middle Pane:** Time evolution of emulated (generative) data $p_\theta(x_{t-1}|\ y,\ x_t)$ for time each 10th time step:

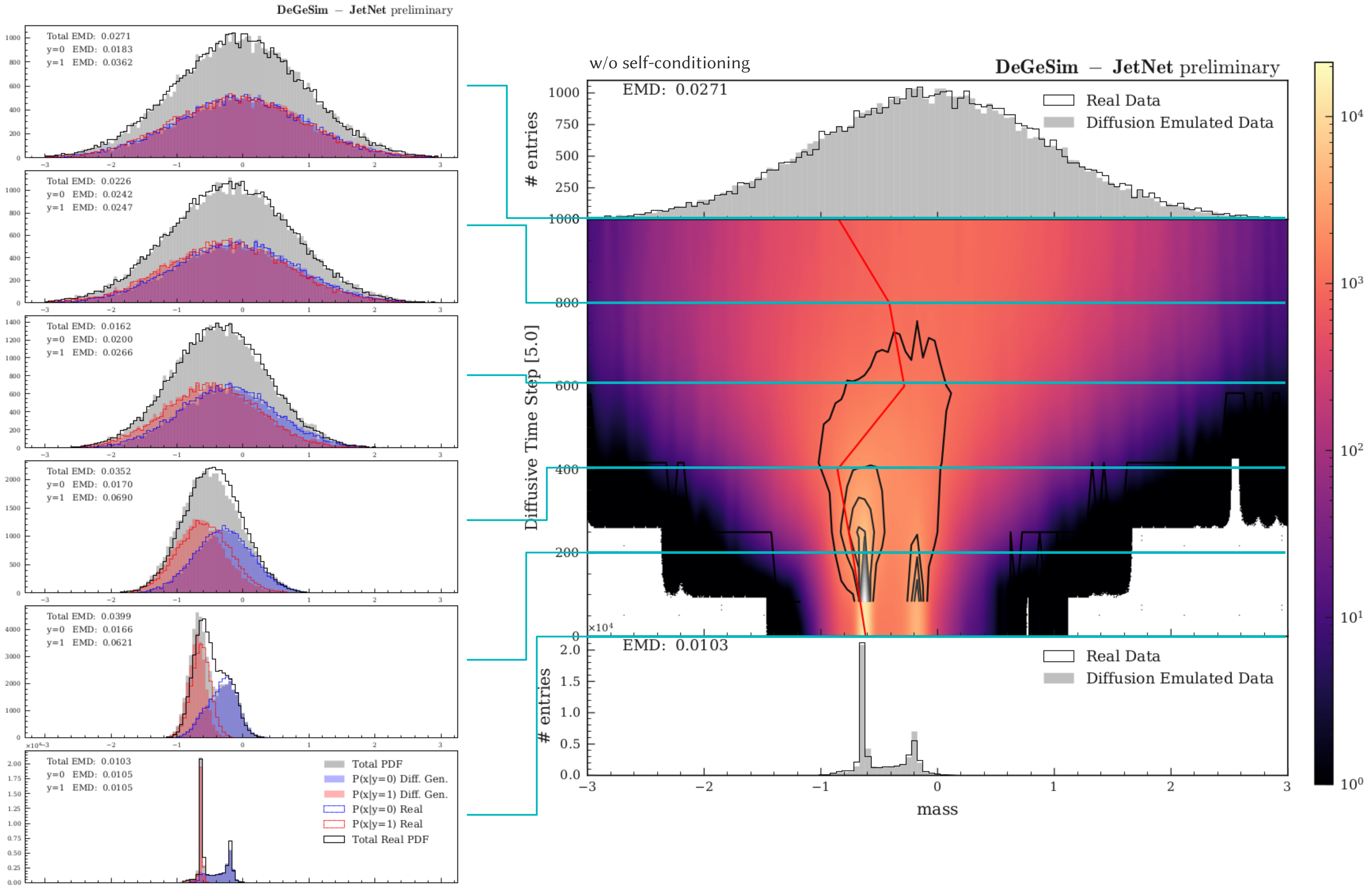
Black contours = equal probability
Red line = random event path

⟶ **Bottom Pane:** Real data vs diffusion emulated data from a guassian prior (top pane).

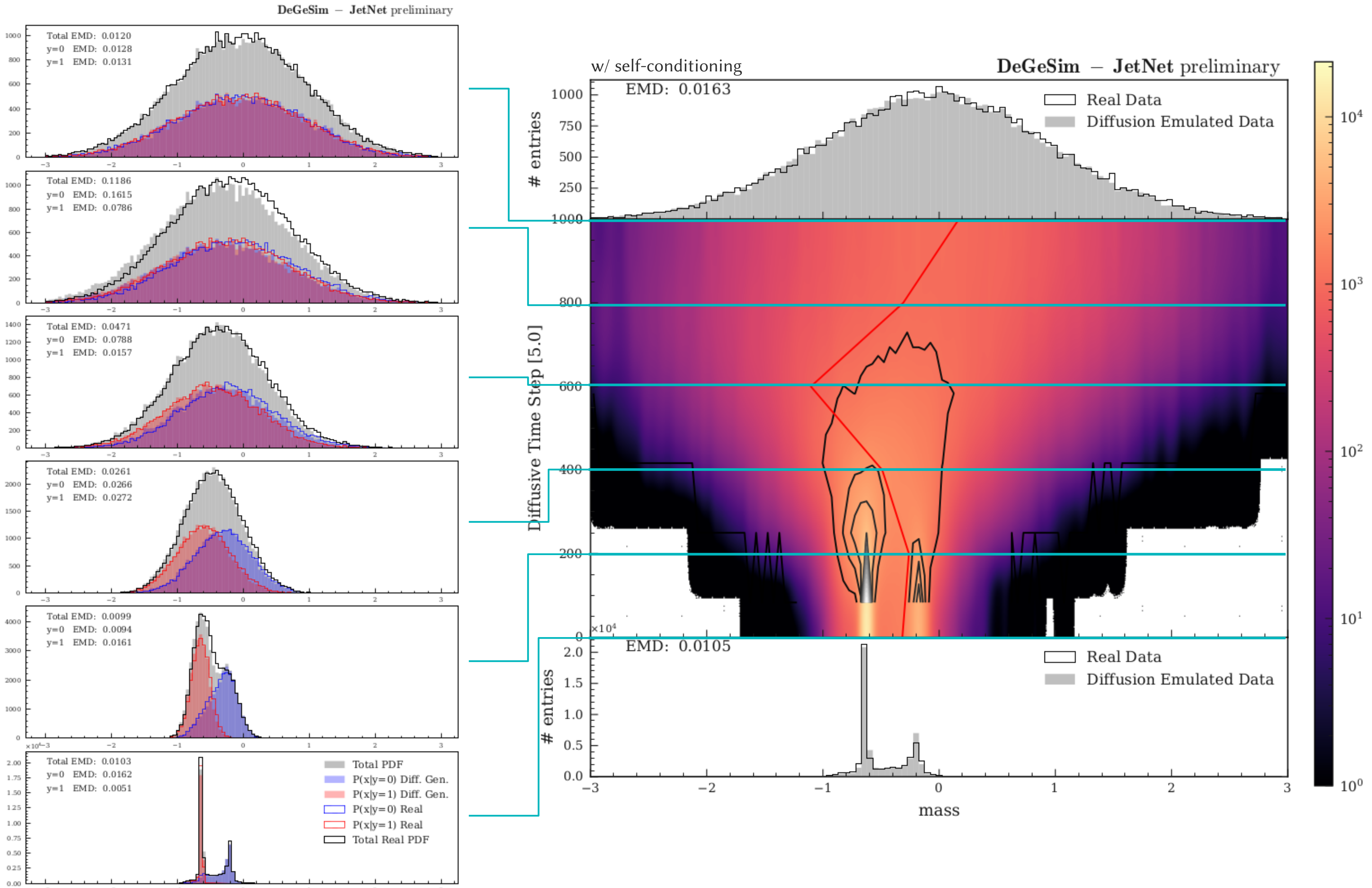
⟶ **Metrics:** Wasserstein distance in 1D projection



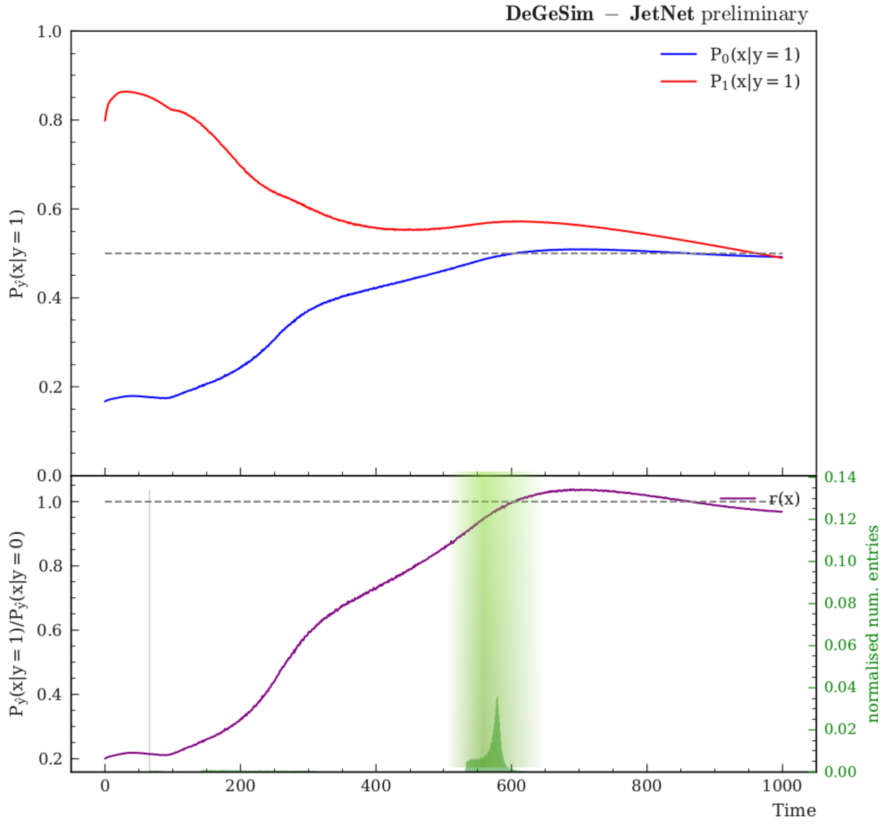
JetNet: Top -vs- W jets



JetNet: Top -vs- W jets



JetNet: Top -vs- W jets

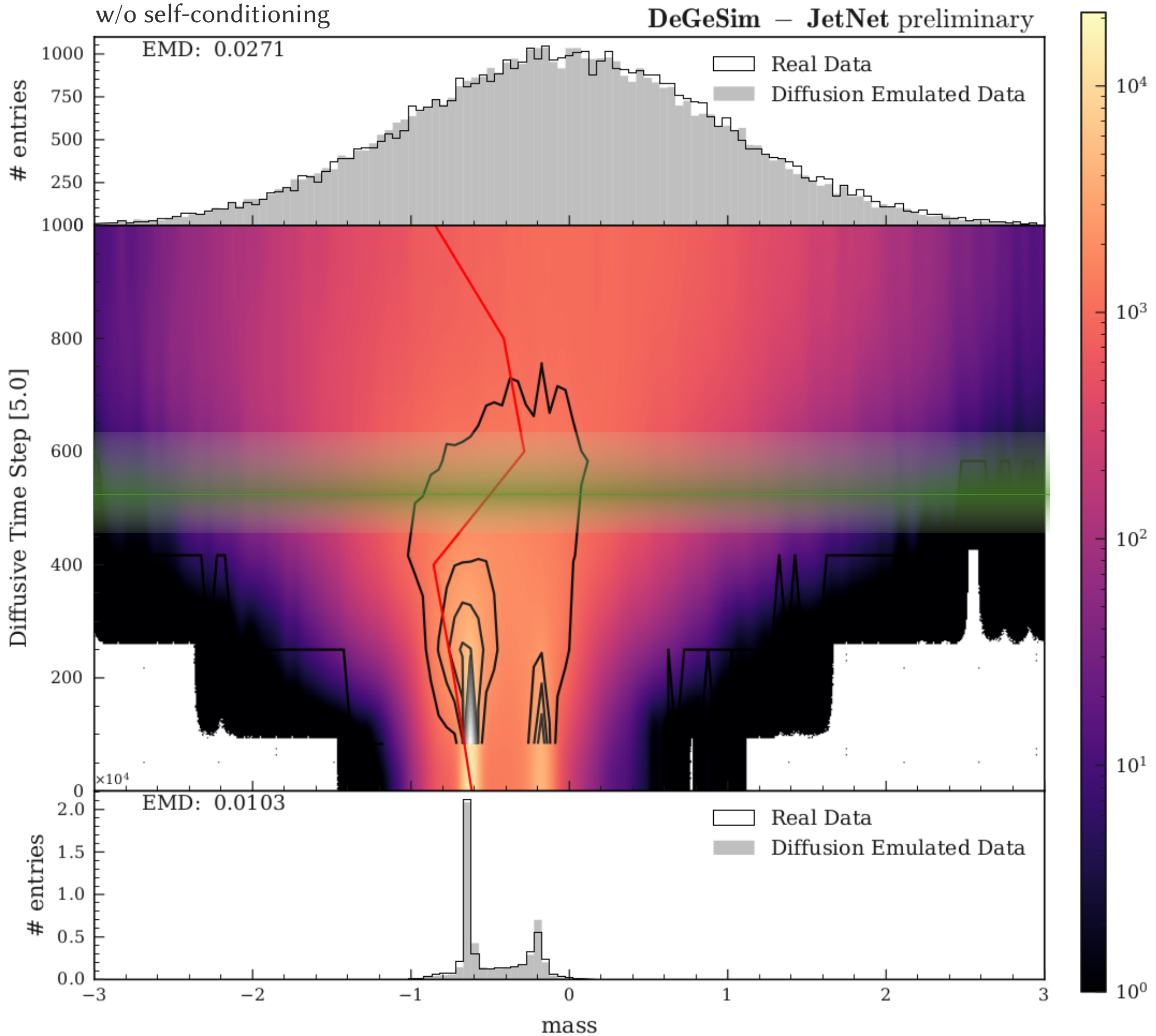


→ The classification neural network approximates the classifier s(x):

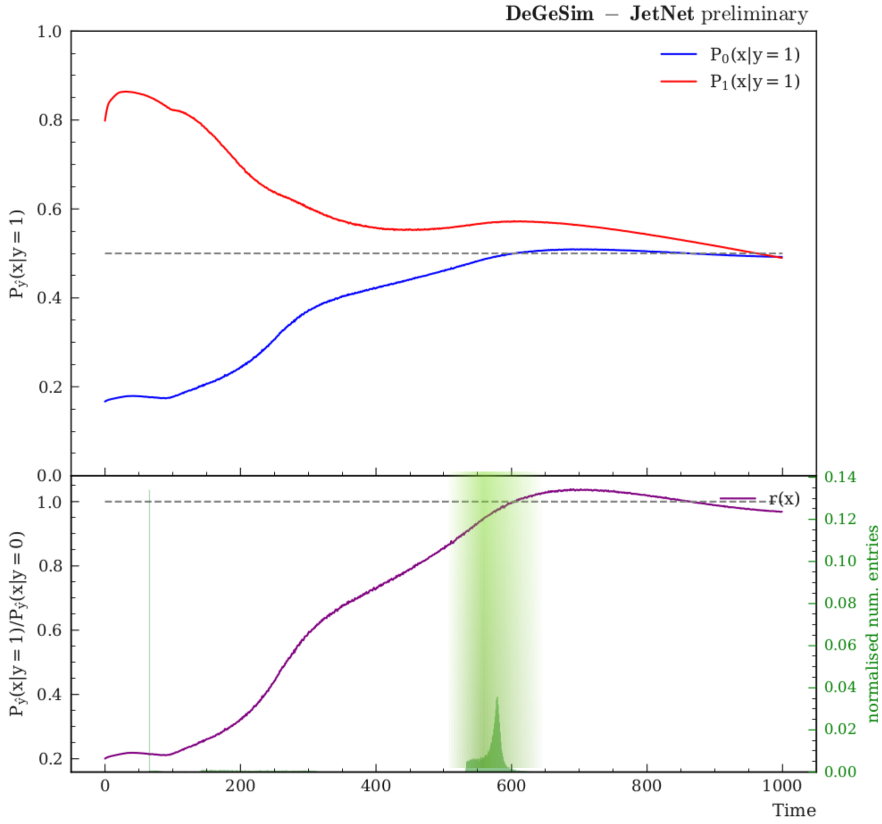
$$s(x,t)=\frac{P(y=1|x,t)}{P(y=0|x,t)+P(y=1|x,t)}$$

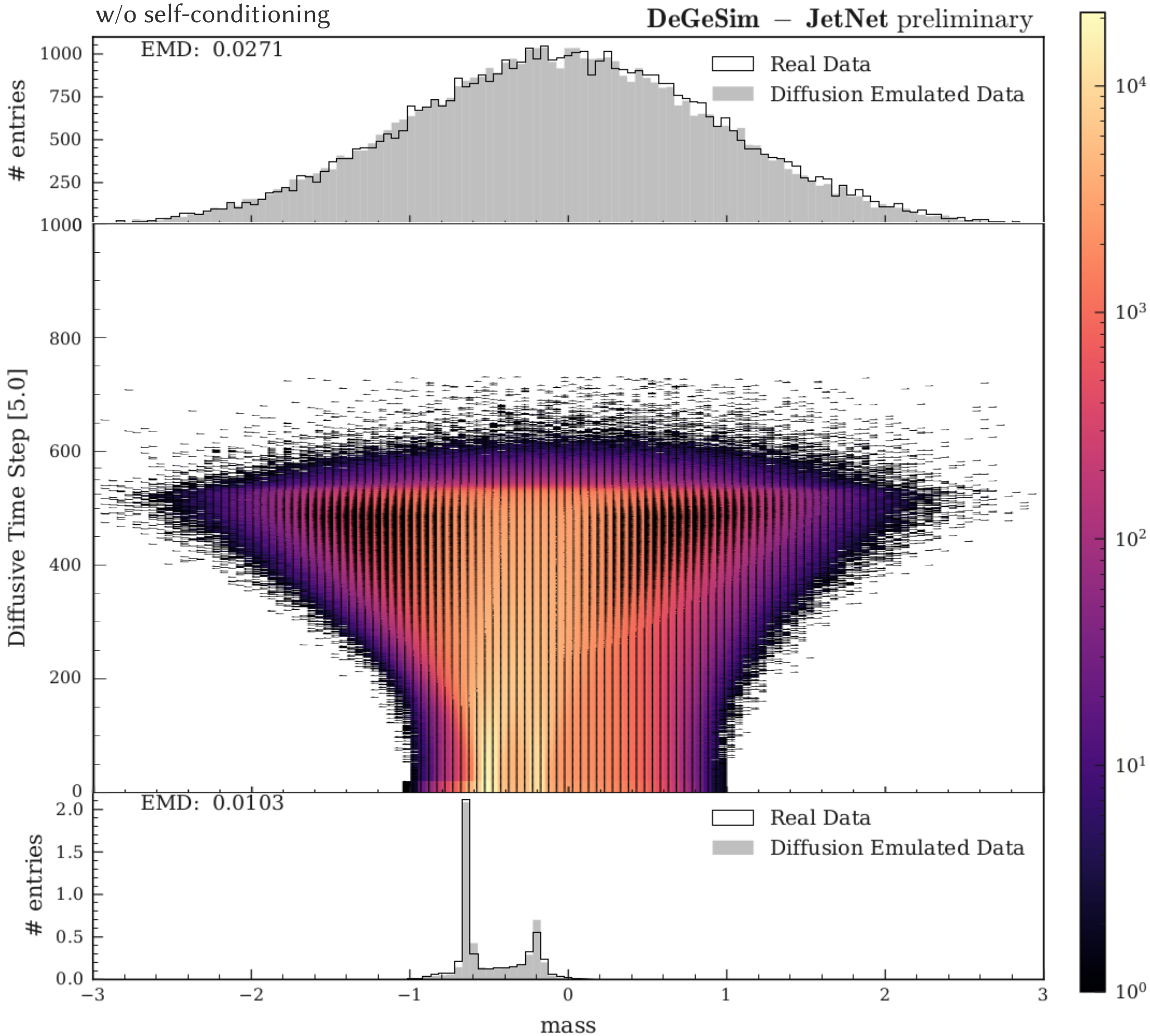
→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x,t_{release})=\frac{s(x,t_{release})}{1-s(x,t_{release})}\approx 1$$



JetNet: Top -vs- W jets





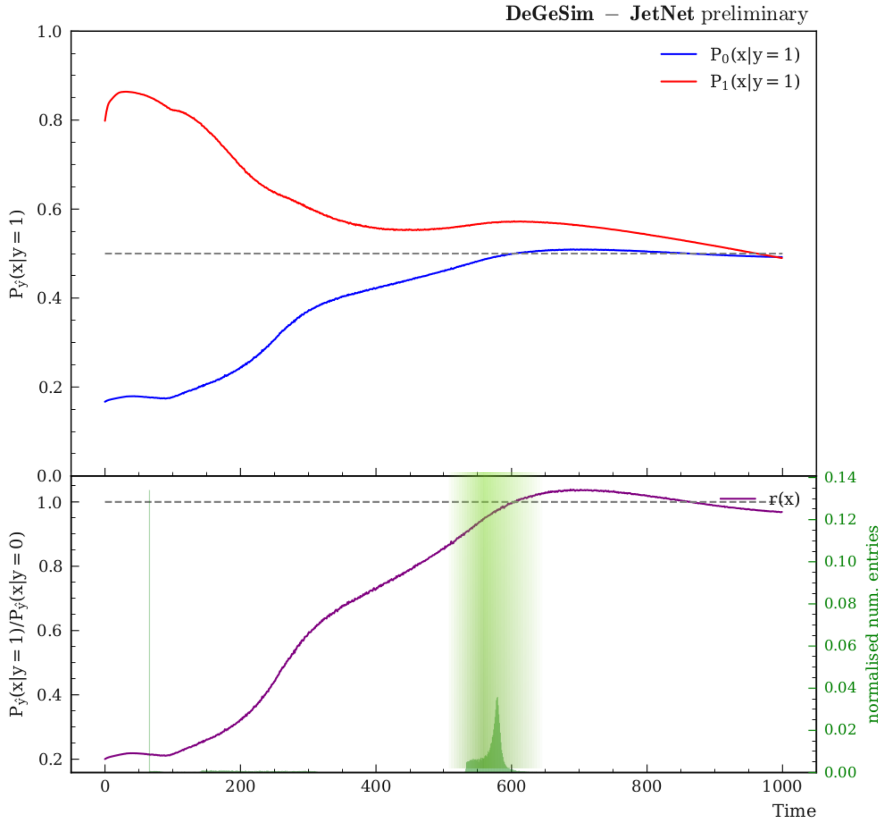
→ The classification neural network approximates the classifier s(x):

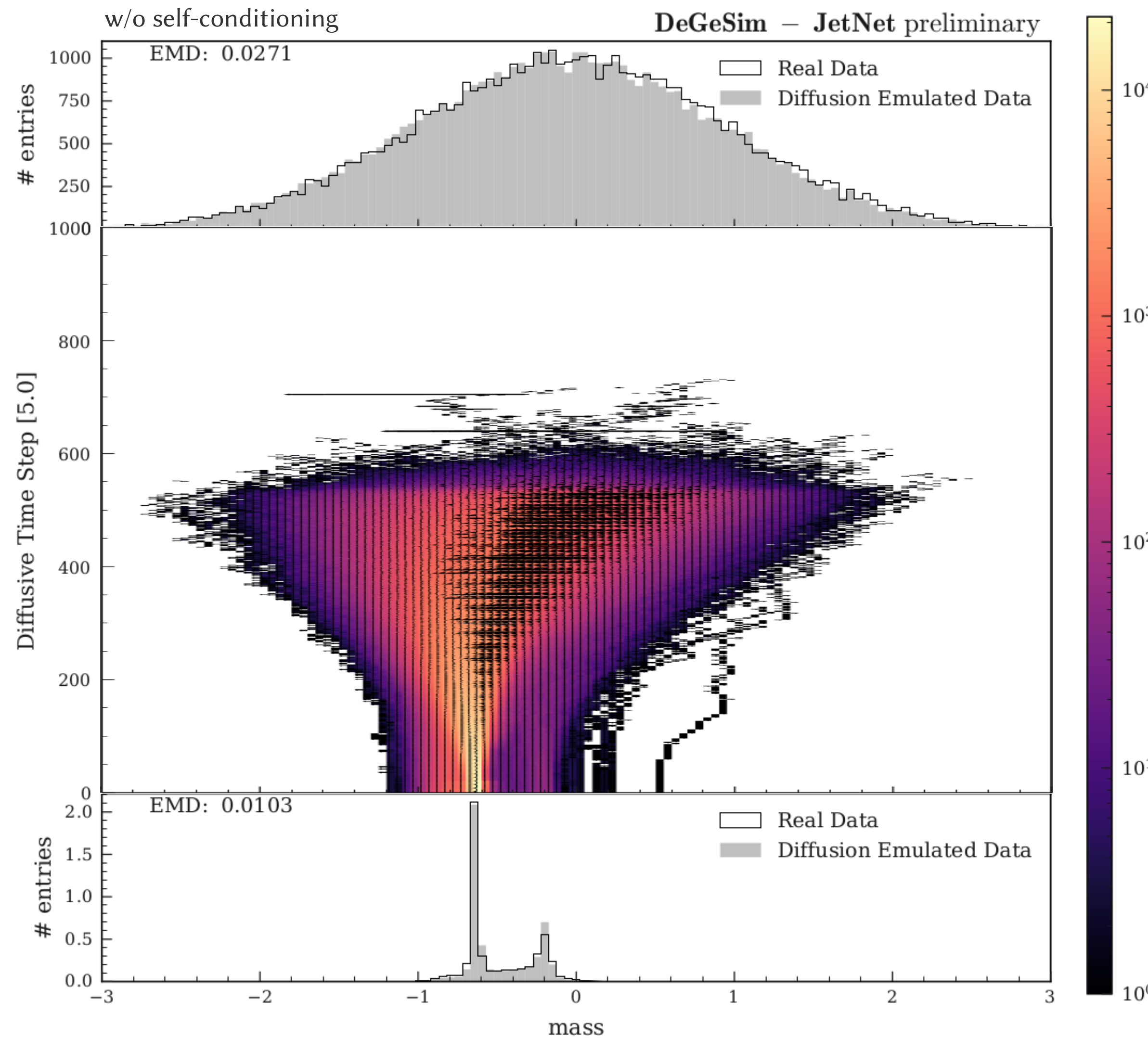
$$s(x,t)=\frac{P(y=1|x,t)}{P(y=0|x,t)+P(y=1|x,t)}$$

→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x,t_{release})=\frac{s(x,t_{release})}{1-s(x,t_{release})}\approx 1$$

JetNet: Top -vs- W jets





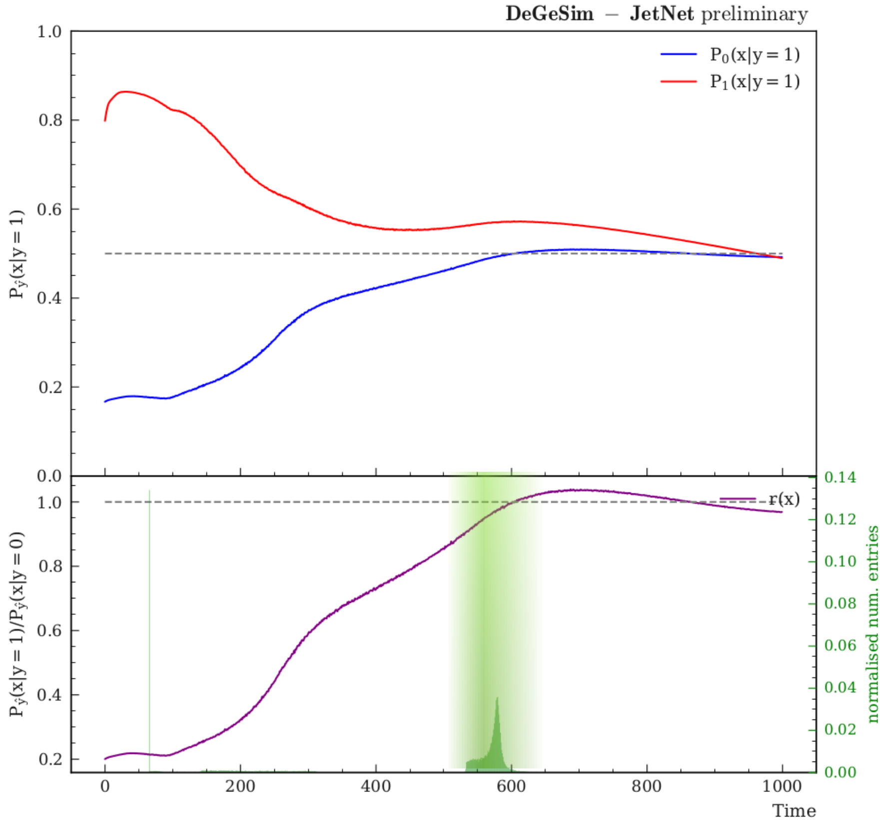
⟶ The classification neural network approximates the classifier s(x):

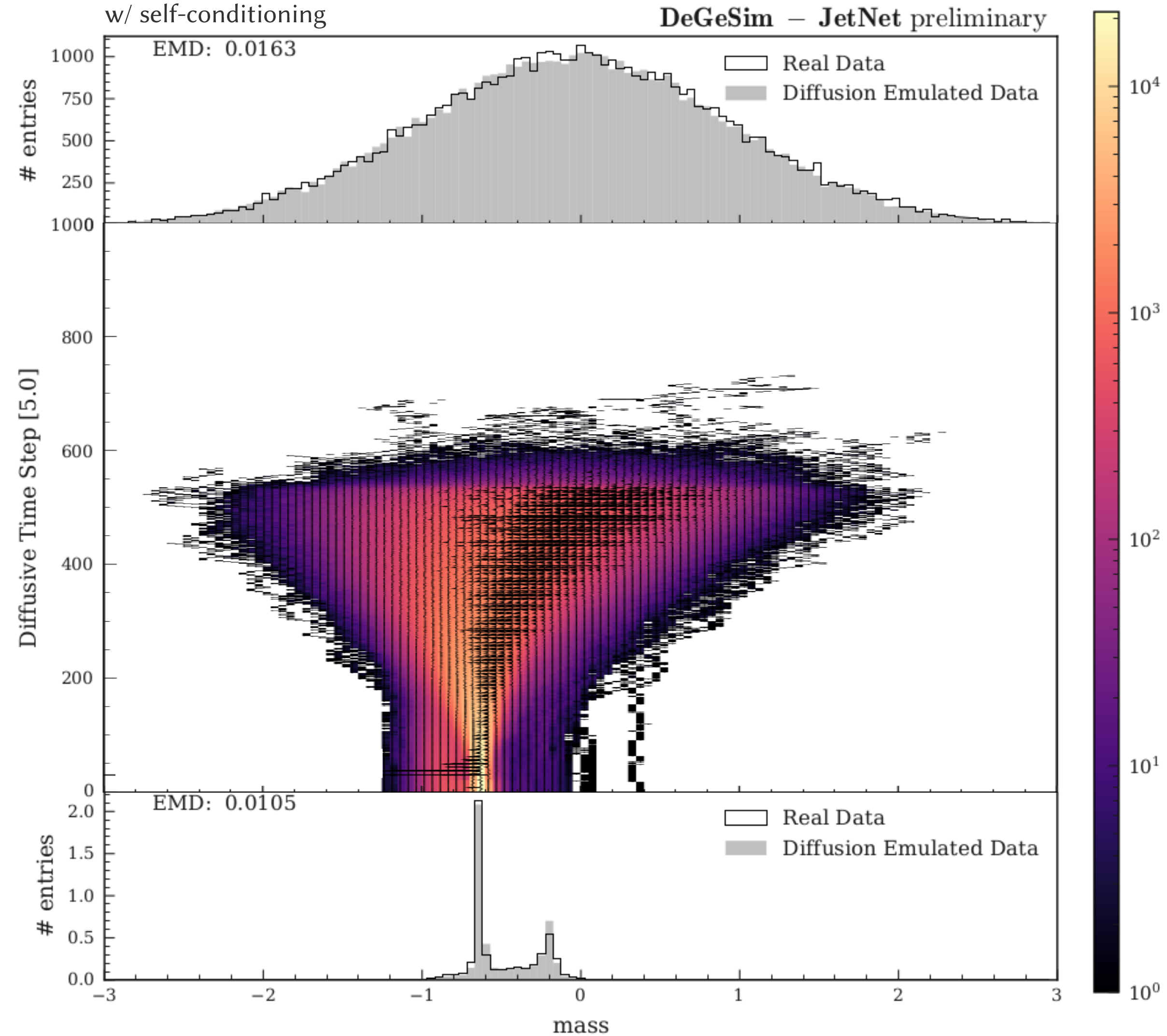
$$s(x,t)=\frac{P(y=1|x,t)}{P(y=0|x,t)+P(y=1|x,t)}$$

⟶ When the densities are equivalent we define the *release time* of each event/data point

$$r(x,t_{release})=\frac{s(x,t_{release})}{1-s(x,t_{release})}\approx 1$$

JetNet: Top -vs- W jets





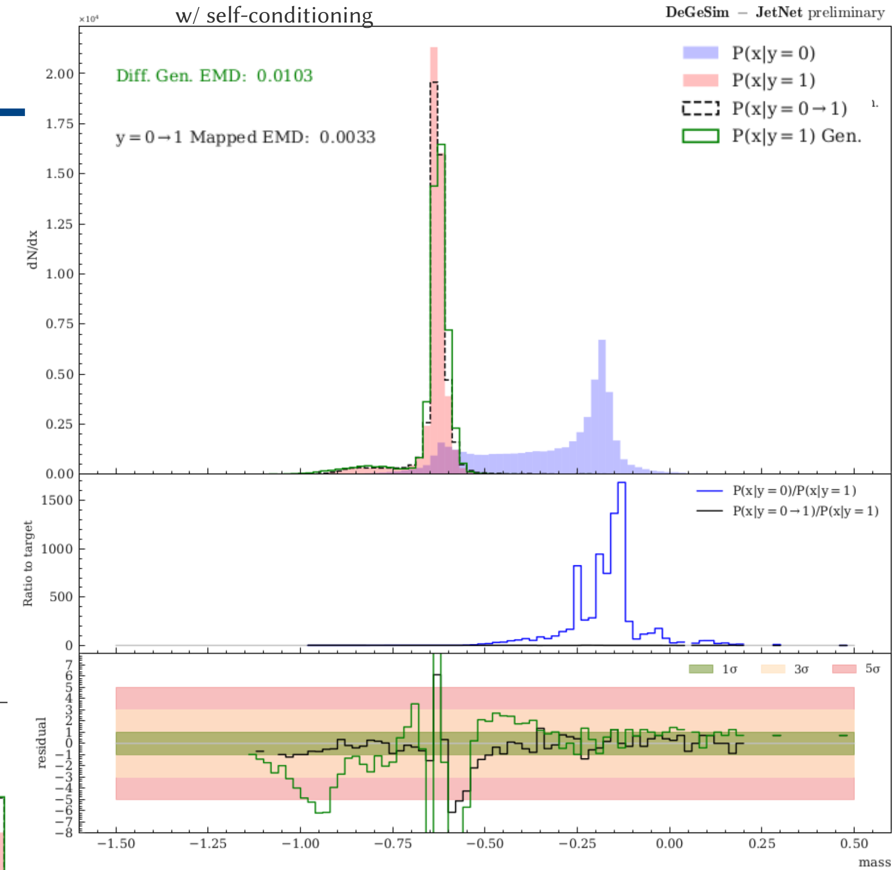
→ The classification neural network approximates the classifier s(x):

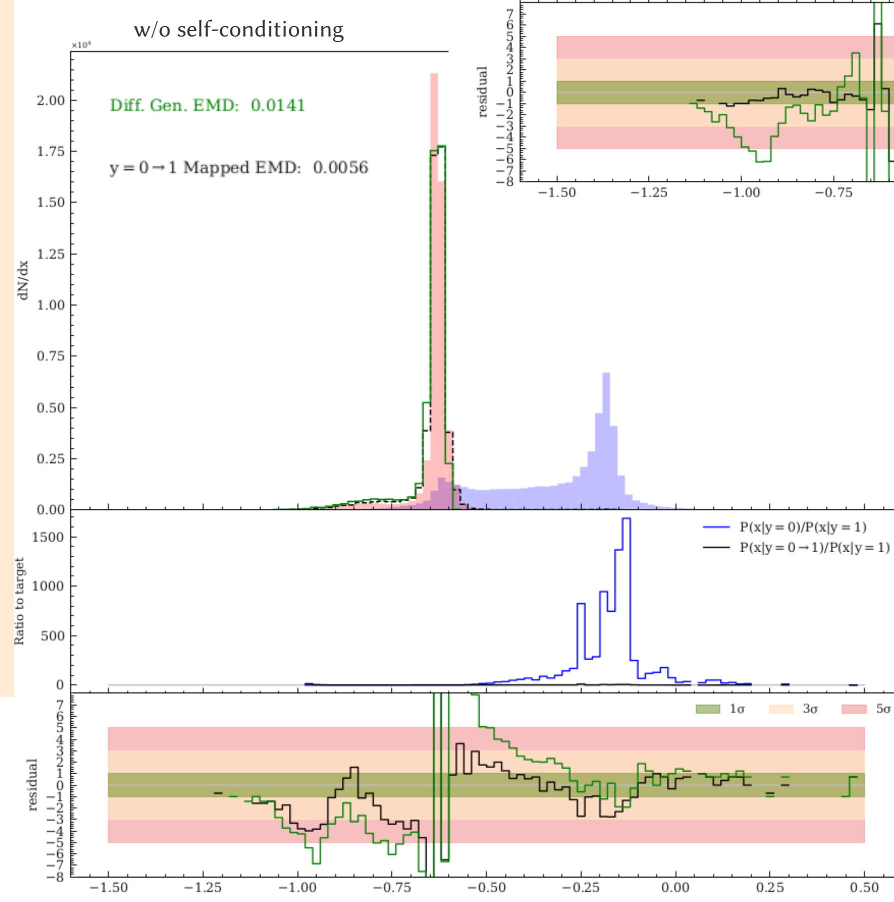
$$s(x,t)=\frac{P(y=1|x,t)}{P(y=0|x,t)+P(y=1|x,t)}$$

→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x,t_{release})=\frac{s(x,t_{release})}{1-s(x,t_{release})}\approx 1$$

JetNet: Top -vs- W jets





Key

→ **Top Pane:** Real data (shaded histograms) split based on class label

y = 0 – Top-Jets

y = 1 - W-Jets

→ **Middle Pane:** Ratio to target (y=1)

Black = Mapped Distrbution

Blue = y=0 / y=1

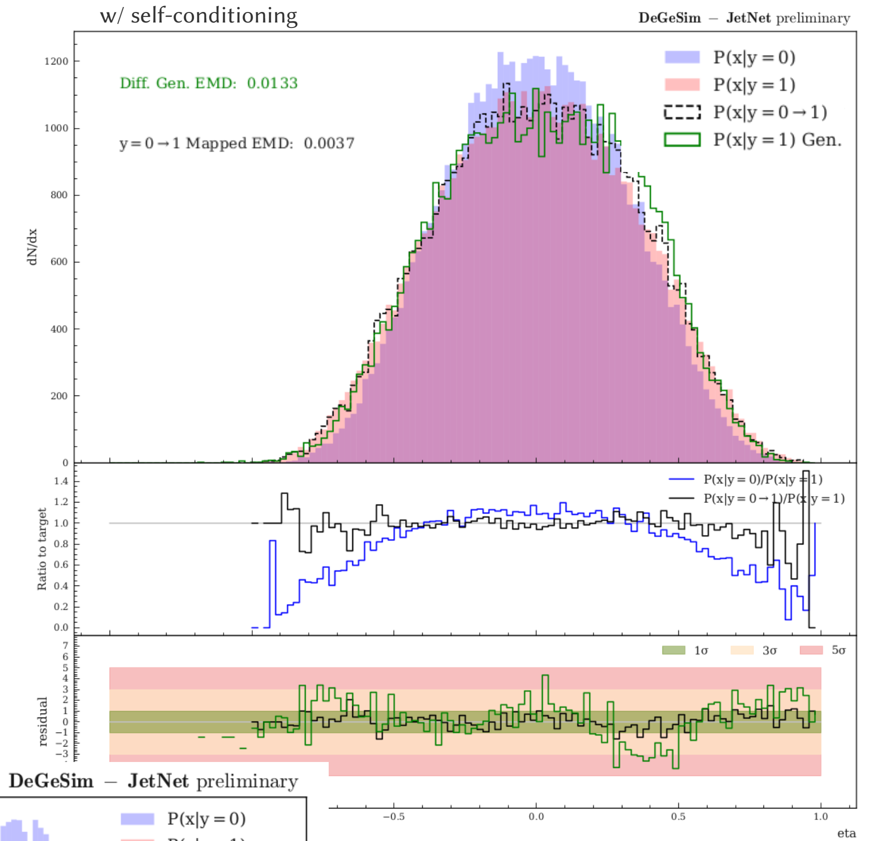
→ **Bottom Pane:** Residuals to target for each component:

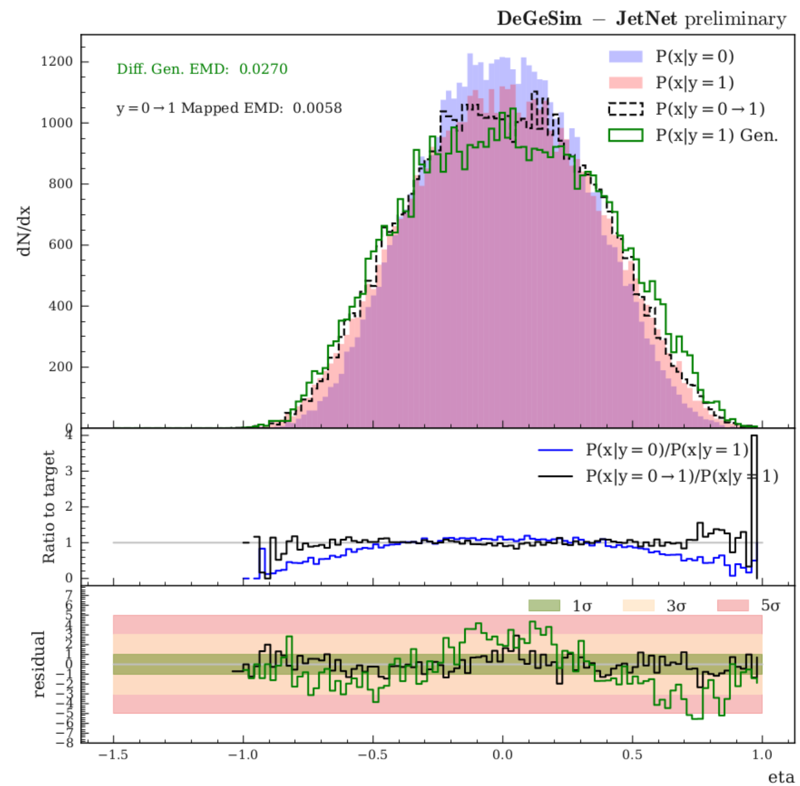
$$Residual = a\text{-}b \,/\, [\sigma_a^{\,2} + \sigma_b^{\,2}]^{1/2}$$

Target is P(x|y=1), i.e. W-boson jets

→ **Metrics:** Wasserstein distance in 1D projection

JetNet: Top -vs- W jets





Key

→ **Top Pane:** Real data (shaded histograms) split based on class label

y = 0 – Top-Jets

y = 1 - W-Jets

→ **Middle Pane:** Ratio to target (y=1)

Black = Mapped Distrbution

Blue = y=0 / y=1

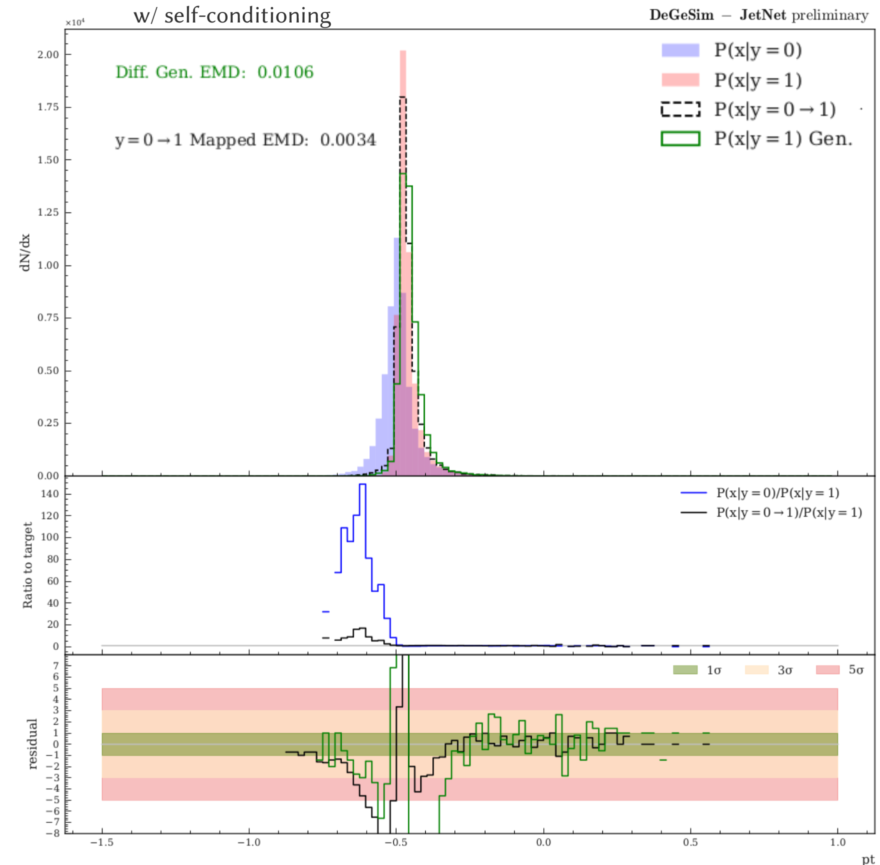
→ **Bottom Pane:** Residuals to target for each component:

$$Residual = a-b / [\sigma_a^2 + \sigma_b^2]^{1/2}$$

Target is P(x|y=1), i.e. W-boson jets

→ **Metrics:** Wasserstein distance in 1D projection

JetNet: Top -vs- W jets



Key

→ **Top Pane:** Real data (shaded histograms) split based on class label

y = 0 – Top-Jets

y = 1 - W-Jets

→ **Middle Pane:** Ratio to target (y=1)

Black = Mapped Distrbution

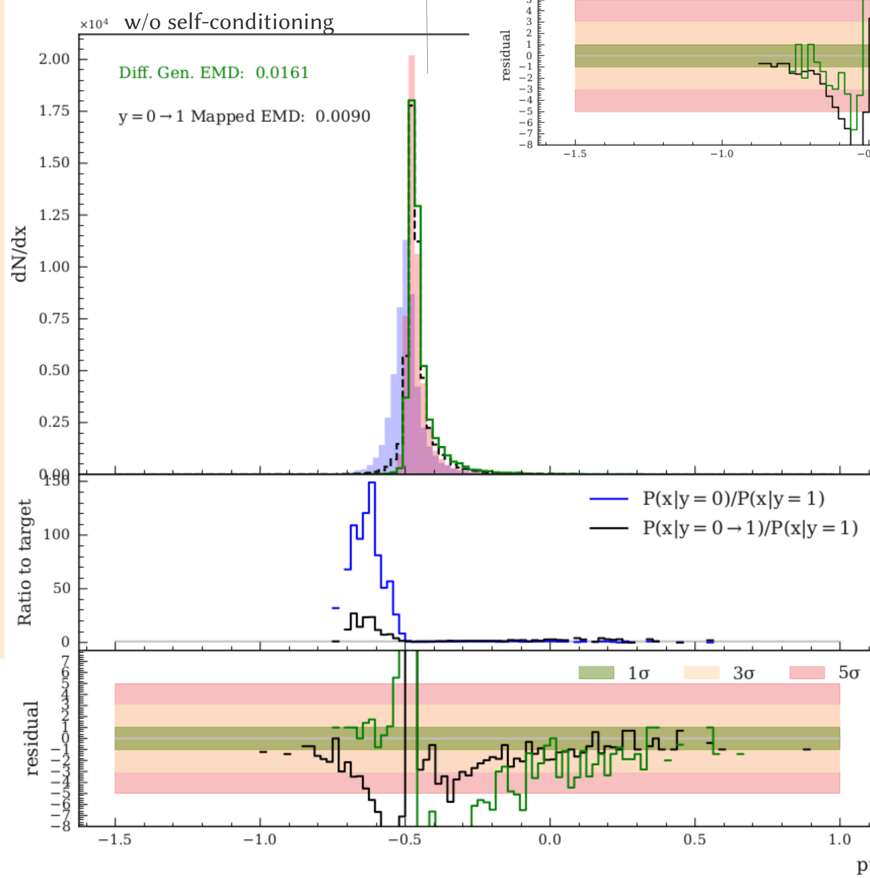
Blue = y=0 / y=1

→ **Bottom Pane:** Residuals to target for each component:

$$Residual = a\text{-}b\ /\ [\sigma_a^{\ 2} + \sigma_b^{\ 2}]^{1/2}$$

Target is P(x|y=1), i.e. W-boson jets

→ **Metrics:** Wasserstein distance in 1D projection



Summary

→ **Goal:** Emulate calorimeter *images* of pileup (minimum/zero bias) seeded from MC based simulations

→ **Status:** Partial diffusion using self-conditioning and classifier free guidance to preserve class-specific features prior to image denoising

→ *Image-to-image (I2I) translation using partial information loss*

→ **Problems:** Null value sparsity injection into calorimeter images:

→ CNNs struggle with noisy images and null-value injection

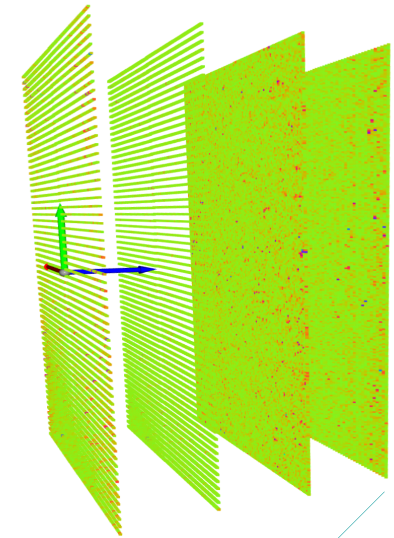
→ Point clouds introduce cell ↔ point assigment ambiguity

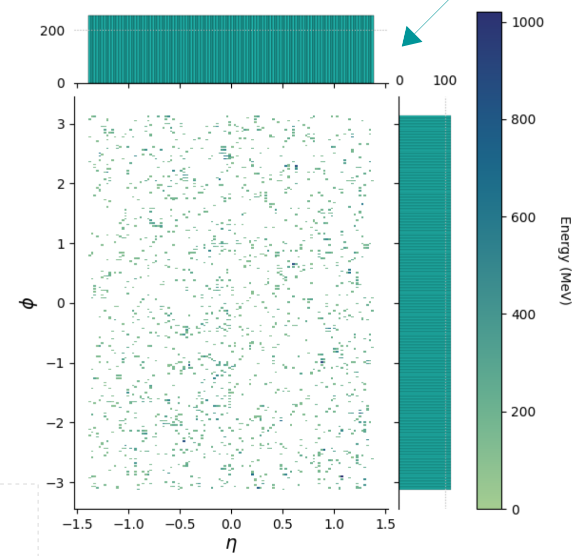
→ Sparse convolutional operations

→ **Further work** will include:

→ Cast discrete state spaces into continuous state space for diffusion (**Analog Bits**)

→ **Cross-domain** conditioning to enhance domain translation duing diffusion





CONTINUOUS & DISCRETE STATE SPACE SAFE CONDITIONAL DENOISING DIFFUSION PROBABILISTIC MODELS USING SELF & CROSS-DOMAIN CONDITIONING FOR MULTI-DIMENSIONAL MAPPING

A PREPRINT

Stephen P.W. Jiggins
Deutsches Elektronnen Synchrotron (DESY)
Notkestraße 85
D-22607, Hamburg
Germany
stephen.jiggins@desy.de

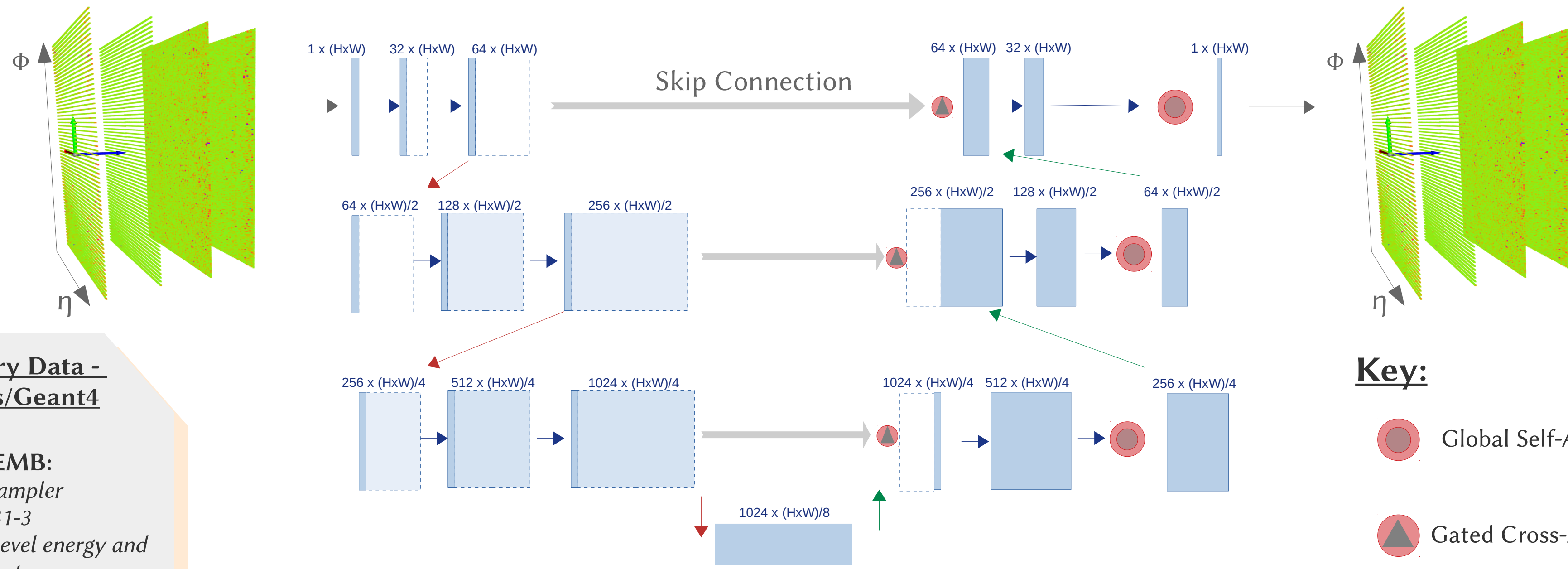
Judith Katzy
Deutsches Elektronnen Synchrotron (DESY)
Notkestraße 85
D-22607, Hamburg
Germany
judith.katzy@desy.de

arXiv coming soon...

Backup

Neural Network Architecture

Latent DDPM



Calorimetry Data - Delphes/Geant4

→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

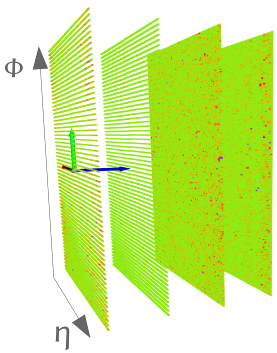
→ **Delphes:**
- → *ECAL*
- → *HCAL*

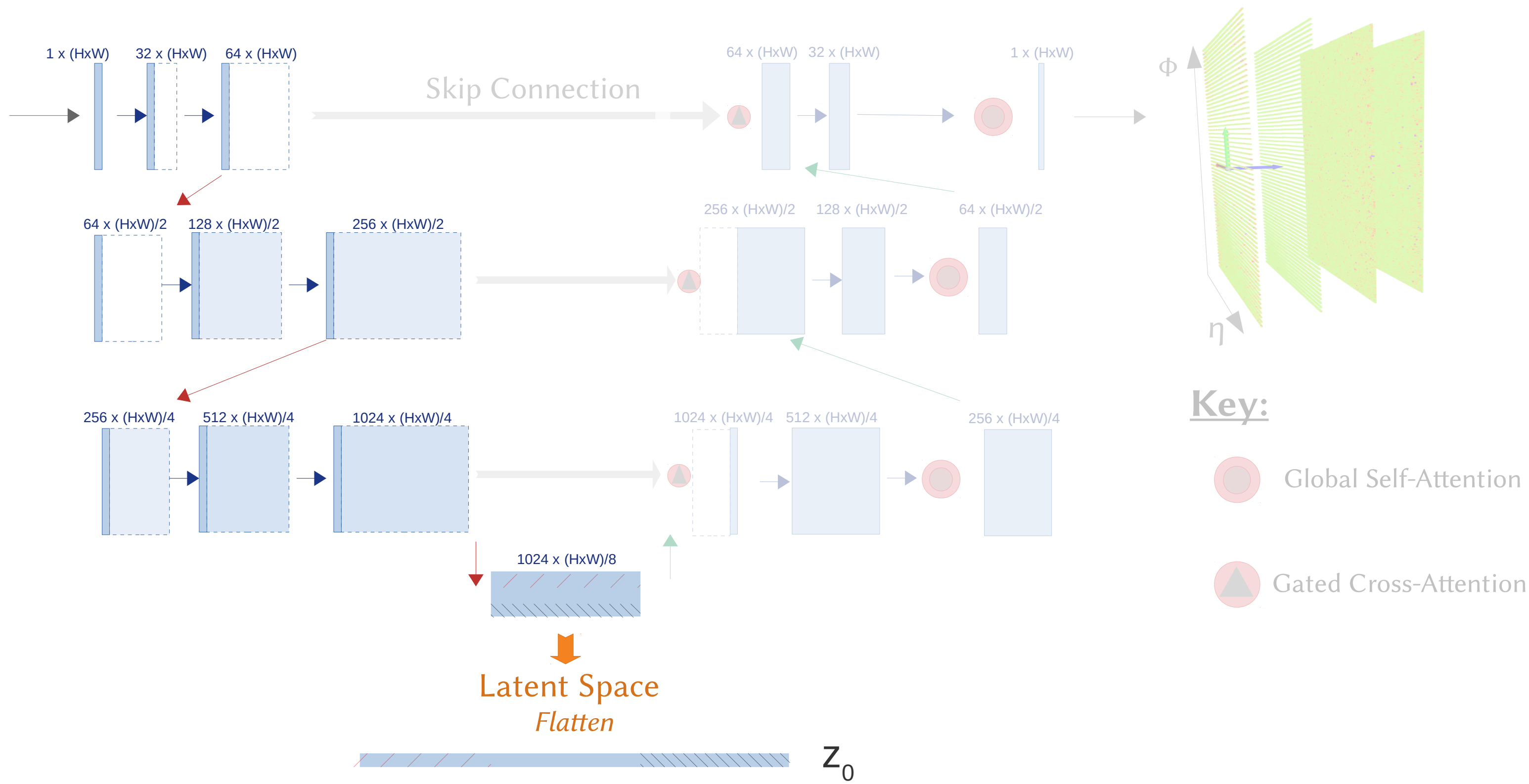
Key:

Global Self-Attention

Gated Cross-Attention

Latent DDPM



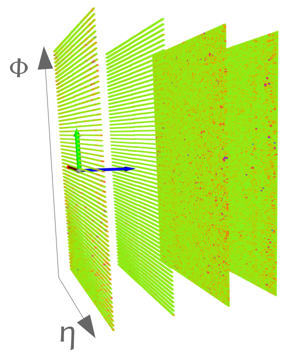


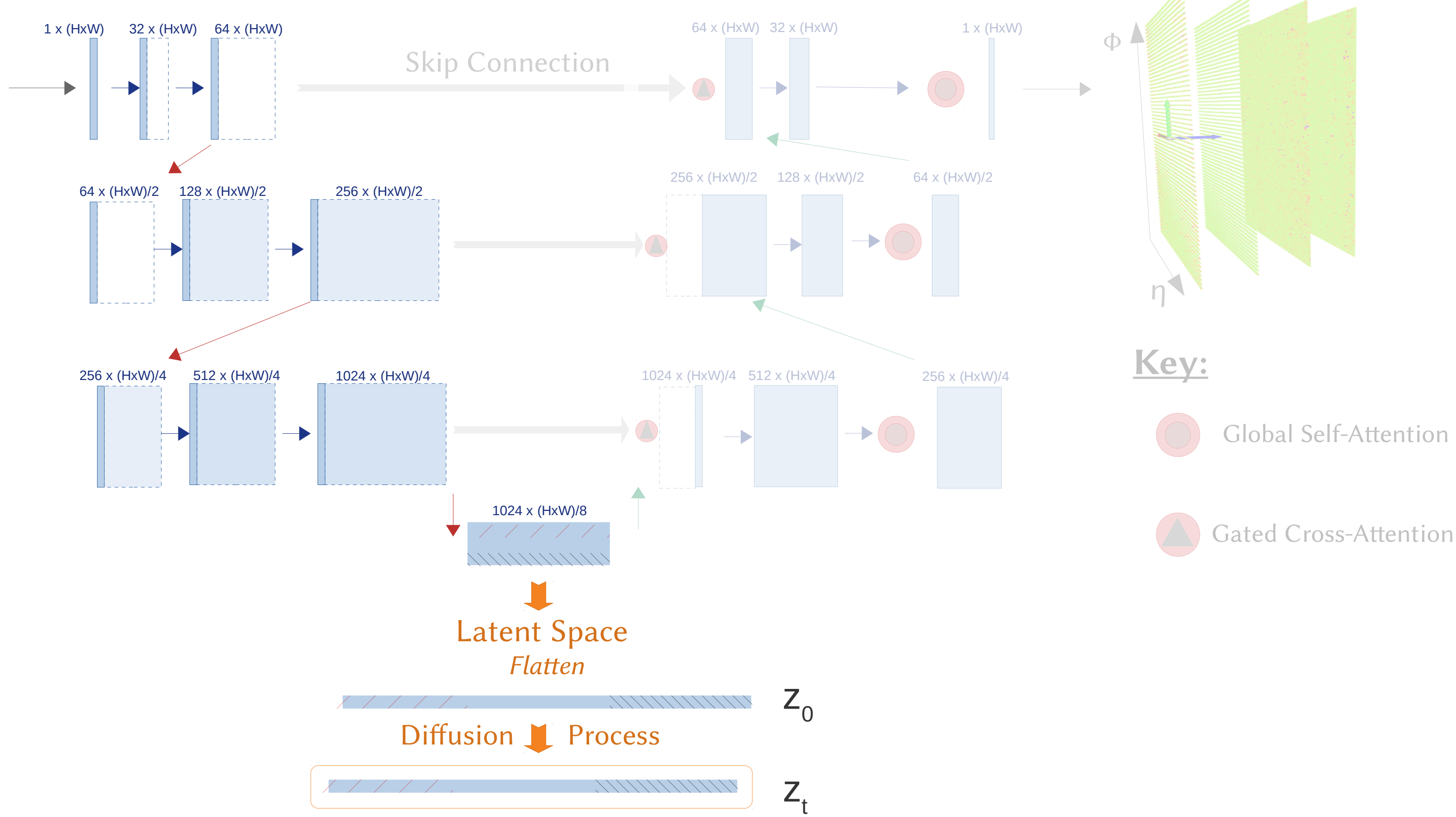
Calorimetry Data - Delphes/Geant4

→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

→ **Delphes:**
- → *ECAL*
- → *HCAL*

Latent DDPM



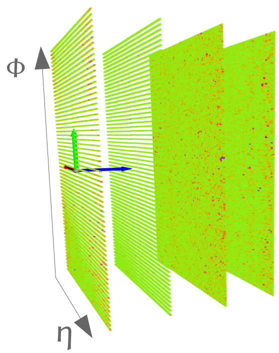


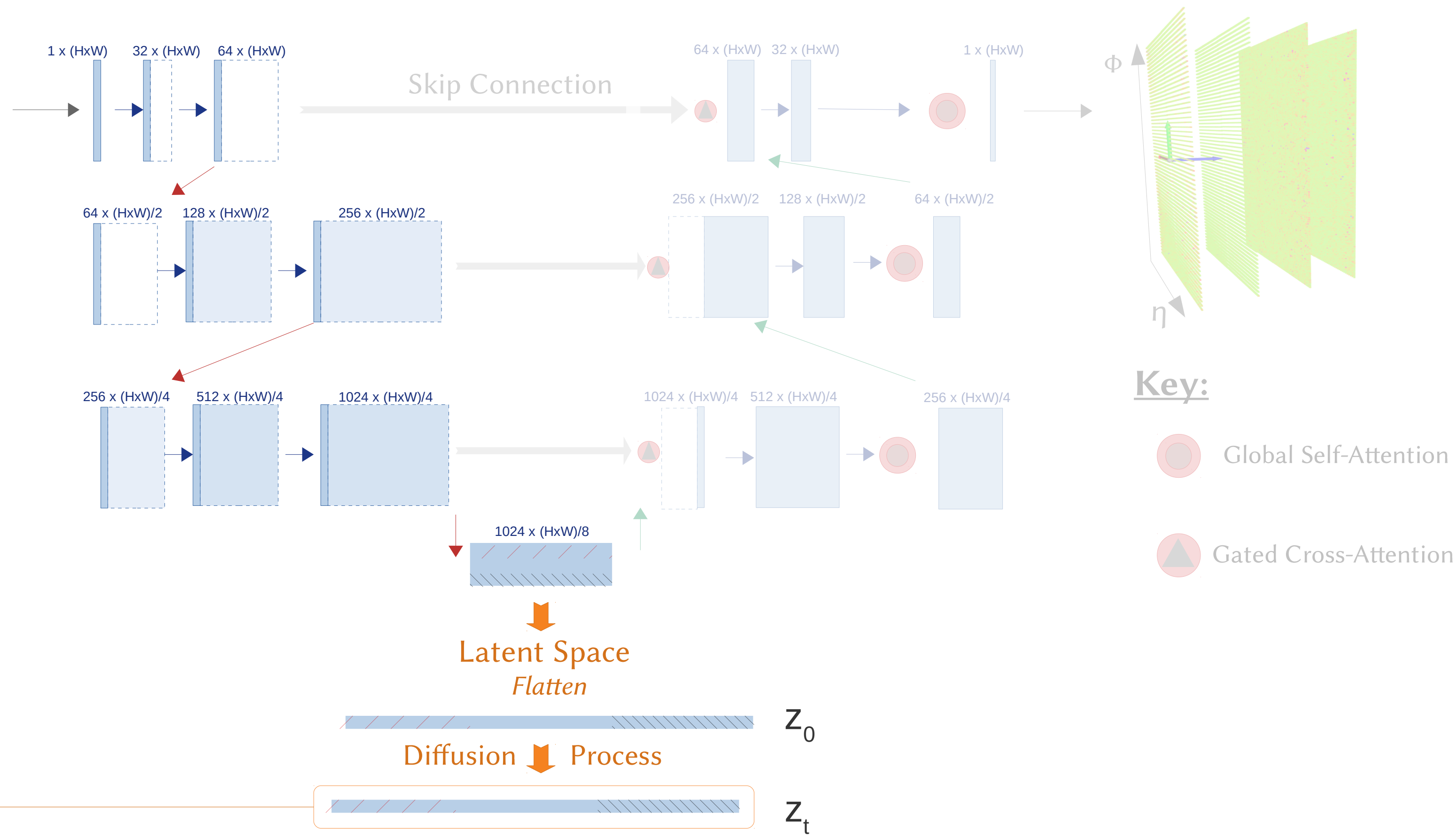
Calorimetry Data - Delphes/Geant4

→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

→ **Delphes:**
- → *ECAL*
- → *HCAL*

Latent DDPM

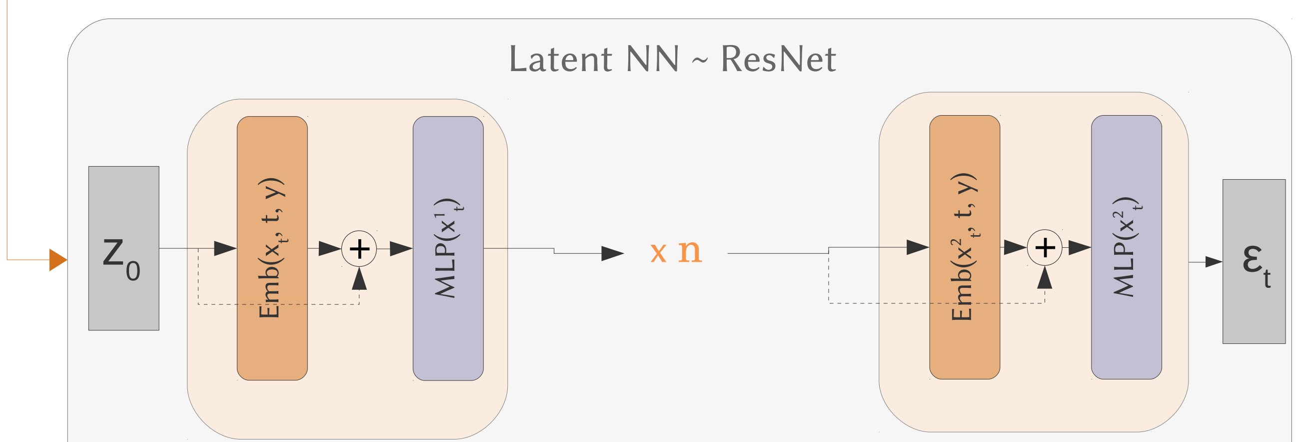




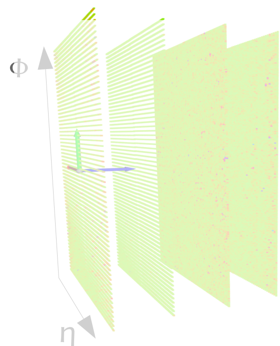
Calorimetry Data - Delphes/Geant4

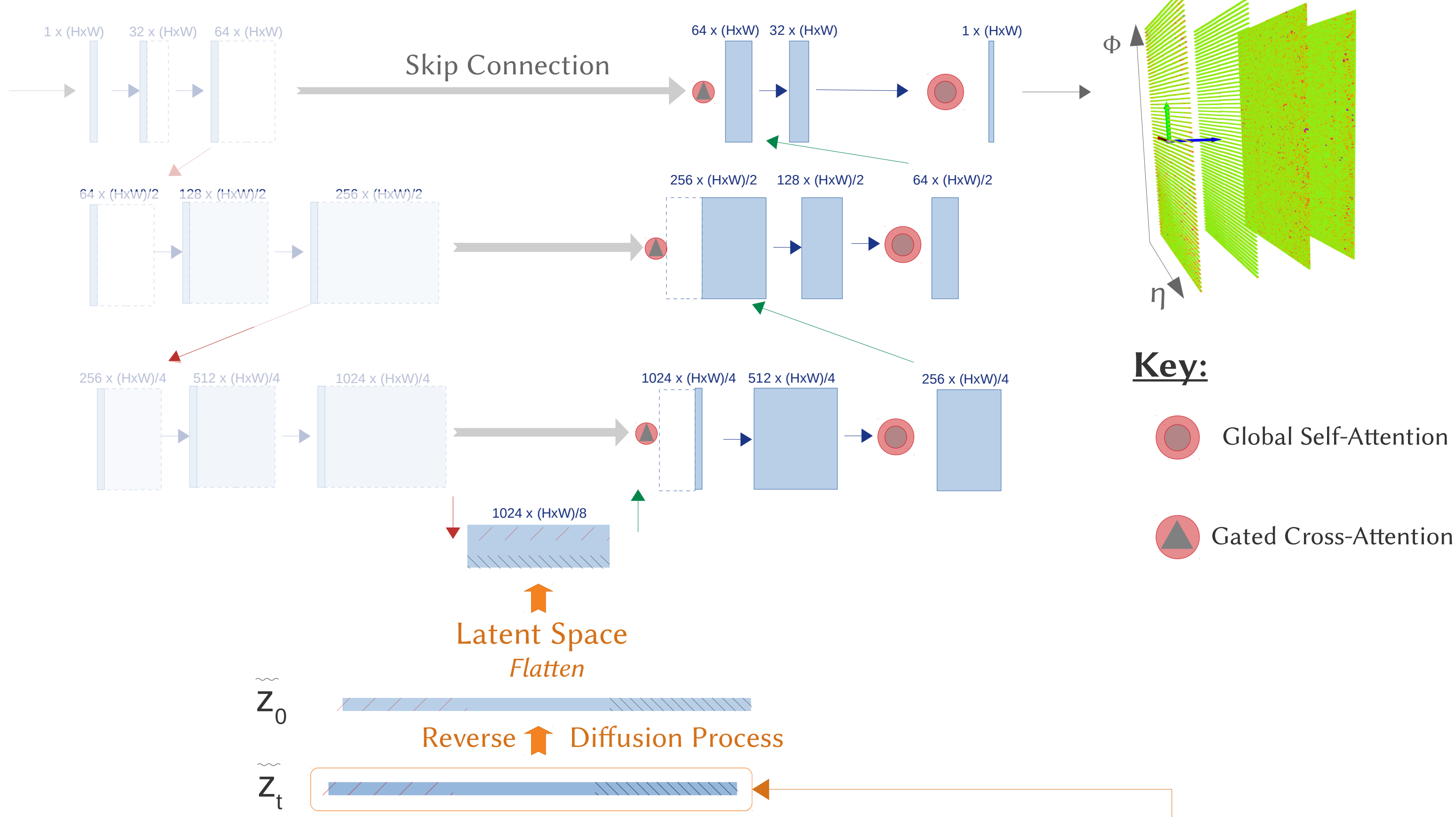
→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

→ **Delphes:**
- → *ECAL*
- → *HCAL*



Latent DDPM

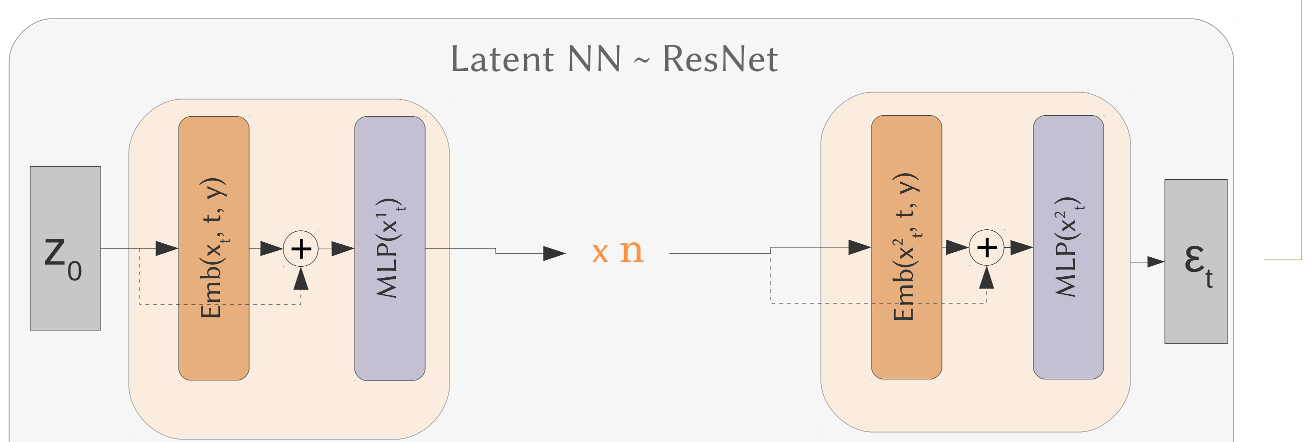




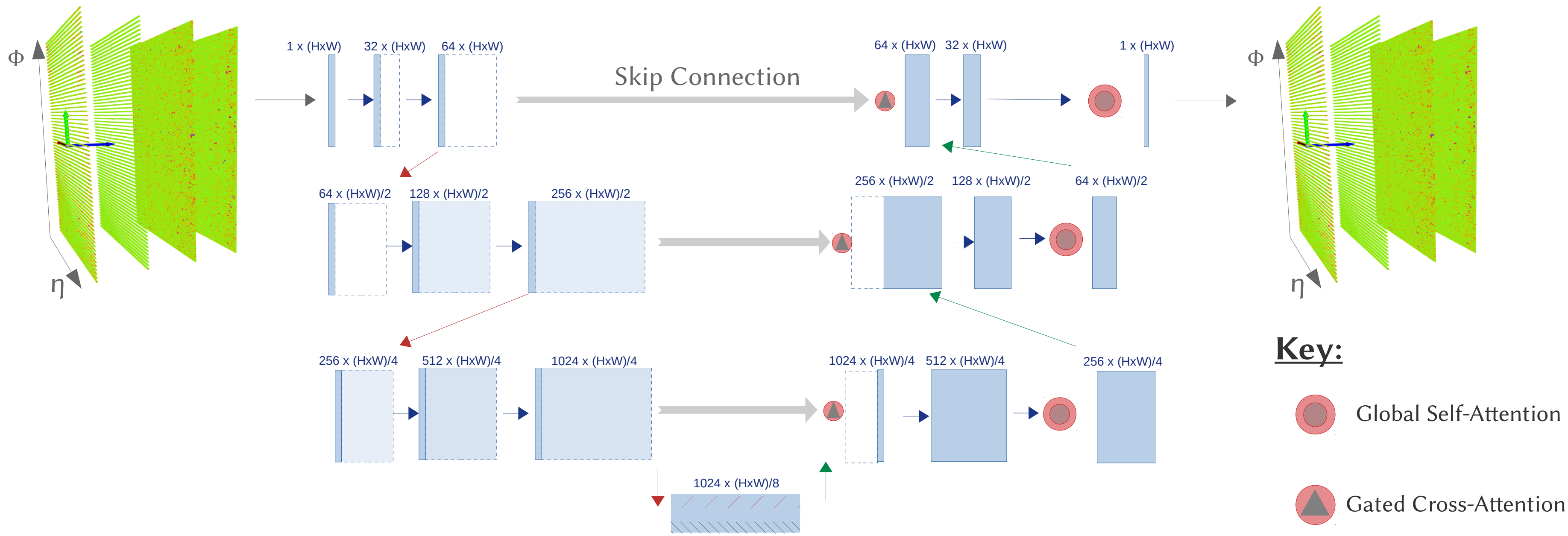
Calorimetry Data - Delphes/Geant4

→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

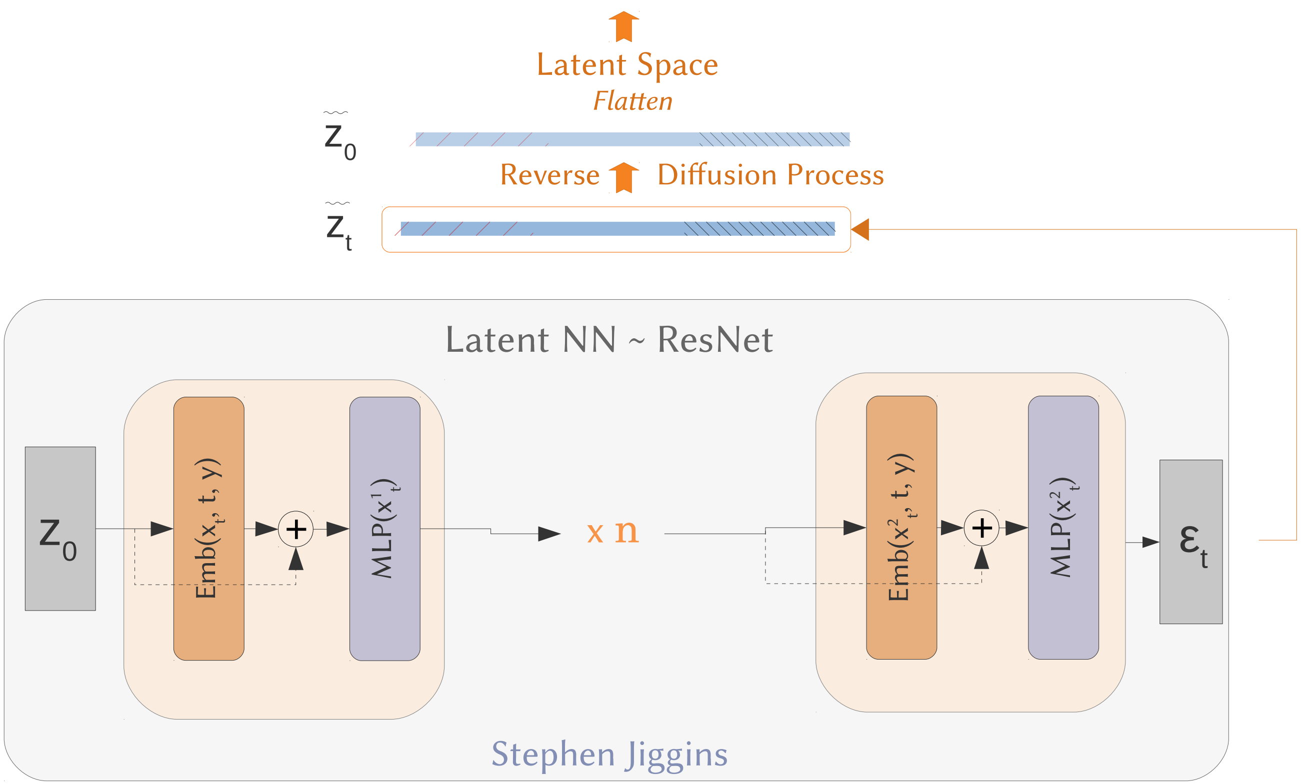
→ **Delphes:**
- → *ECAL*
- → *HCAL*



Latent DDPM



Φ
η
1 x (HxW)
32 x (HxW)
64 x (HxW)
Skip Connection
64 x (HxW)
32 x (HxW)
1 x (HxW)
64 x (HxW)/2
128 x (HxW)/2
256 x (HxW)/2
256 x (HxW)/2
128 x (HxW)/2
64 x (HxW)/2
256 x (HxW)/4
512 x (HxW)/4
1024 x (HxW)/4
1024 x (HxW)/4
512 x (HxW)/4
256 x (HxW)/4
1024 x (HxW)/8
Latent Space
Flatten
Key:
Global Self-Attention
Gated Cross-Attention



z̃0
Reverse Diffusion Process
z̃t
Latent NN ~ ResNet
Z0
Emb(xt, t, y)
MLP(x1t)
x n
Emb(x2t, t, y)
MLP(x2t)
εt

Denoising Diffusion Probabilistic Models

DDPMs – Recipe

→ **Noise Scheduler:**

Cosine β-scheduling noise injection:

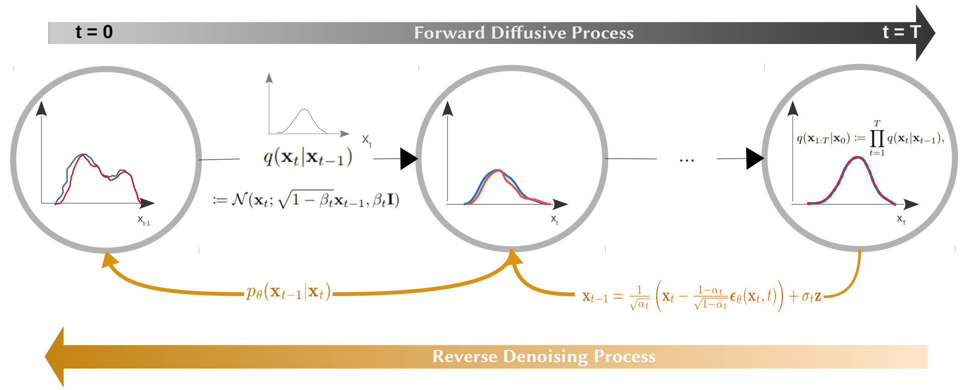
$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \qquad \overline{\alpha}_t = f(t)/f(0) \qquad f(t) = \cos\left(\frac{(t/T+s)\pi}{2.(1+s)}\right)$$

→ **Denoising Score Matching**

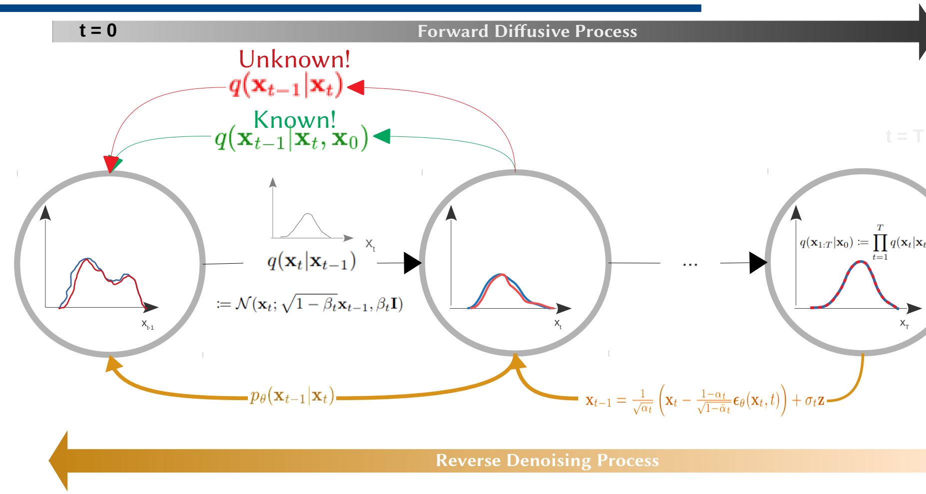
$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),t)^2\right|\right]$$

→ **Density Ratio Estimating Classifer**

$$L_{DRE} = \sum \left[p_c(t,x_0)\log(\widehat{p}_c(t,x_0))\right]$$



Denoising Diffusion Probabilistic Models



DDPMs – Recipe

→ **Noise Scheduler:**

Cosine β-scheduling noise injection:

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}} \qquad \bar{\alpha}_t = f(t)/f(0) \qquad f(t) = \cos\left(\frac{(t/T+s)\pi}{2.(1+s)}\right)$$

→ **Denoising Score Matching**

$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),t)^2\right|\right]$$

→ **Density Ratio Estimating Classifer**

$$L_{DRE} = \sum \left[p_c(t,x_0)\log(\hat{p}_c(t,x_0))\right]$$

→ The *origin* conditional probability is tractable:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

→ Network needs to learn the reverse process:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

→ Lower variational bound objective:

$$L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right]$$

→ DSM needs to learn the term:

$$L_{\text{VLB}} = L_T + \underbrace{L_{T-1} + \cdots}_{} + L_0$$

$$\sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}$$

Denoising Diffusion Probabilistic Models

DDPMs – Recipe

→ **Noise Scheduler:**

Cosine β-scheduling noise injection:

$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \qquad \overline{\alpha}_t = f(t)/f(0) \qquad f(t) = \cos\left(\frac{(t/T+s)\pi}{2.(1+s)}\right)$$

→ **Denoising Score Matching**

$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),t)\right|^2\right]$$

→ **Density Ratio Estimating Classifer**

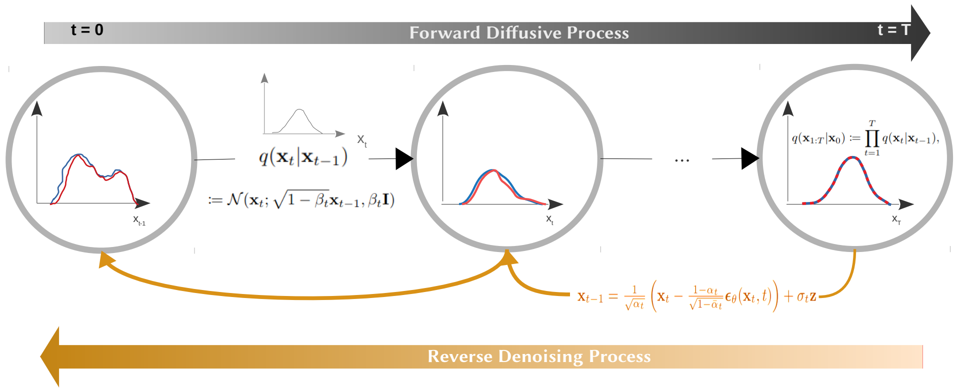
$$L_{DRE} = \sum \left[p_c(t,x_0)\log(\widehat{p}_c(t,x_0))\right]$$

→ **Conditional Embedding**

$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon,y}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),y,t)\right|^2\right]$$

→ **Classifier Free Guidance**

$$\overline{\epsilon}_\theta(x_t, y, t) = (w+1)\,\epsilon_\theta(x_t, y, t) - w.\,\epsilon_\theta(x_t, t)$$



→ Conditional dependency can be integrated:

$$q(\mathbf{x}_{t-1}|x_t, x_{0,y}) = q(\mathbf{x}_t|x_{t-1}, x_{0,y}) \cdot \frac{q(x_{t-1}|x_{0,y})}{q(x_t|x_{0,y})}$$

$$p_\theta(\mathbf{x}_{0:T}|y) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|x_t, y)$$

→ Score estimation naturally extends with this conditionality:

$$\nabla^w_{x_t} \log(p_\theta(\mathbf{x}_t|y)) = (1-w)\nabla_{x_t}\log(p_\theta(\mathbf{x}_t)) + w\nabla_{x_t}\log(p_\theta(\mathbf{x}_t|y))$$

→ Guidance scalar (w) used to control conditional ↔ unconditionally interpolation of score/noise estimators:

$$\overline{\epsilon}(x_t, t, y) = w\,\epsilon_\theta(x_t, t, y) - (1-w)\,\epsilon_\theta(x_t, t)$$

Denoising Diffusion Probabilistic Models

DDPMs – Recipe

→ **Noise Scheduler:**

Cosine β-scheduling noise injection:

$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \qquad \overline{\alpha}_t = f(t)/f(0) \qquad f(t) = \cos\left(\frac{(t/T+s)\pi}{2.(1+s)}\right)$$

→ **Denoising Score Matching**

$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),t)\right|^2\right]$$

→ **Density Ratio Estimating Classifer**

$$L_{DRE} = \sum\left[p_c(t,x_0)\log(\widehat{p}_c(t,x_0))\right]$$

→ **Conditional Embedding**

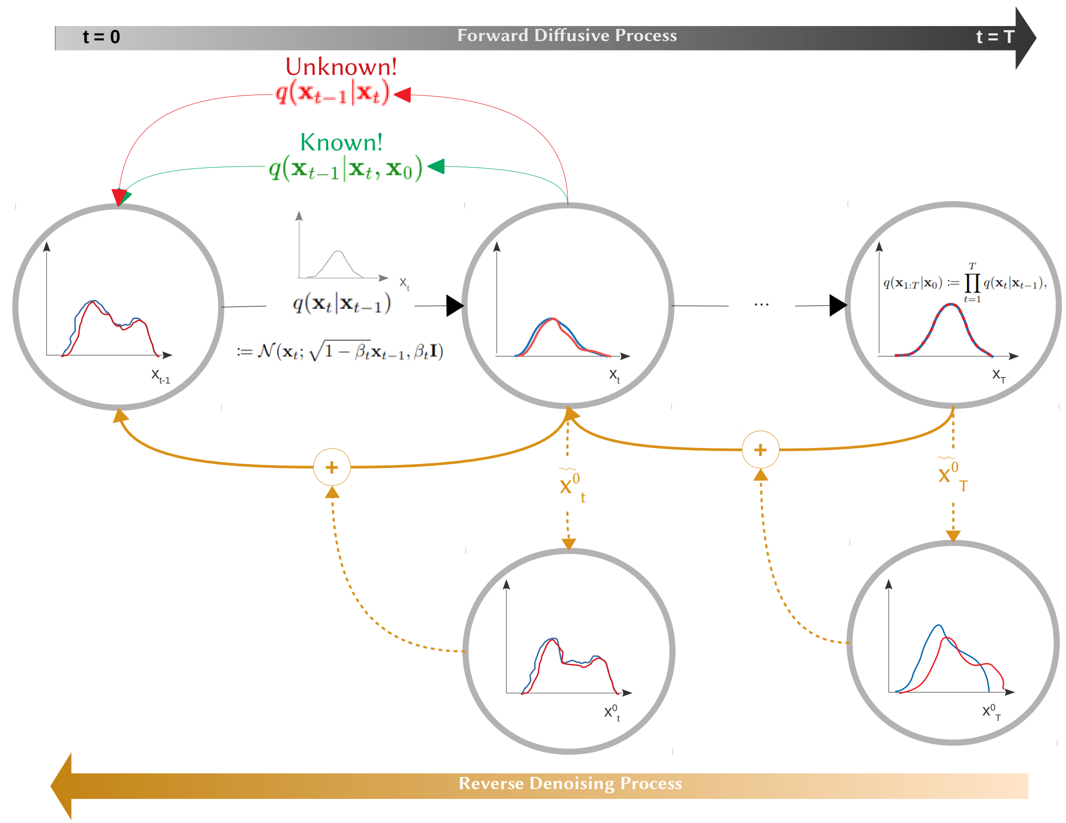
$$L(\theta) = \mathbb{E}_{t,x_0,\epsilon,y}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),y,t)\right|^2\right]$$

→ **Classifier Free Guidance**

$$\overline{\epsilon}_\theta(x_t,y,t) = (w+1)\,\epsilon_\theta(x_t,y,t) - w.\,\epsilon_\theta(x_t,t)$$

→ **Self-Conditioning**

$$L(\theta) = \mathbb{E}_{t,x_0,\widehat{x}_0,\epsilon,y}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon),y,t,\widehat{x}_0)\right|^2\right]$$



→ Self-conditioned dependency can be integrated:

$$p_\theta(\mathbf{x_{0:T}}|y,\tilde{x}_0) = p_\theta(\mathbf{x_T})\prod_{t=1}^{T} p_\theta(\mathbf{x_{t-1}}|x_t,y,\tilde{x}_0)$$

→ Provided that there is a stop gradient on the estimate of the original image $\tilde{x}_0$:

$$\nabla^w_{x_t}\log(p_\theta(\mathbf{x_t}|y,\tilde{x}_0)) = (1-w)\nabla_{x_t}\log(p_\theta(\mathbf{x_t})) + w\nabla_{x_t}\log(p_\theta(\mathbf{x_t}|y,\tilde{x}_0))$$

→ Scalar based guidance becomes:

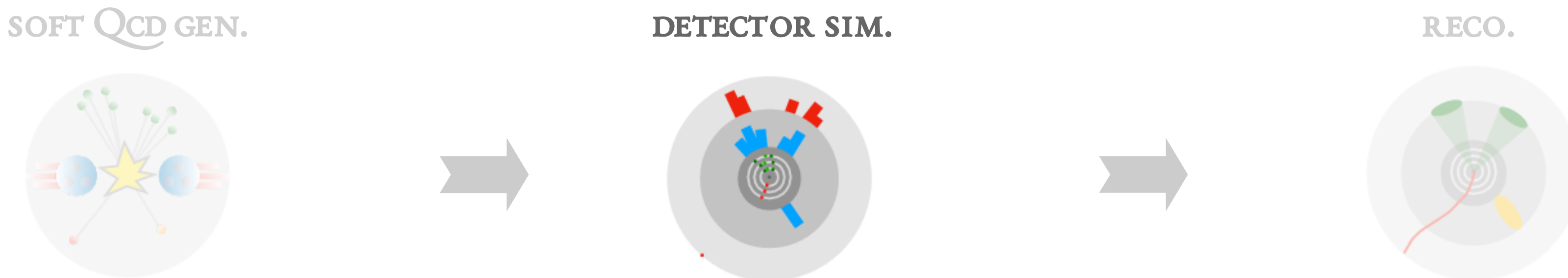
$$\overline{\epsilon}(x_t,t,y) = w\,\epsilon_\theta(x_t,t,y,\tilde{x}_0) - (1-w)\,\epsilon_\theta(x_t,t)$$

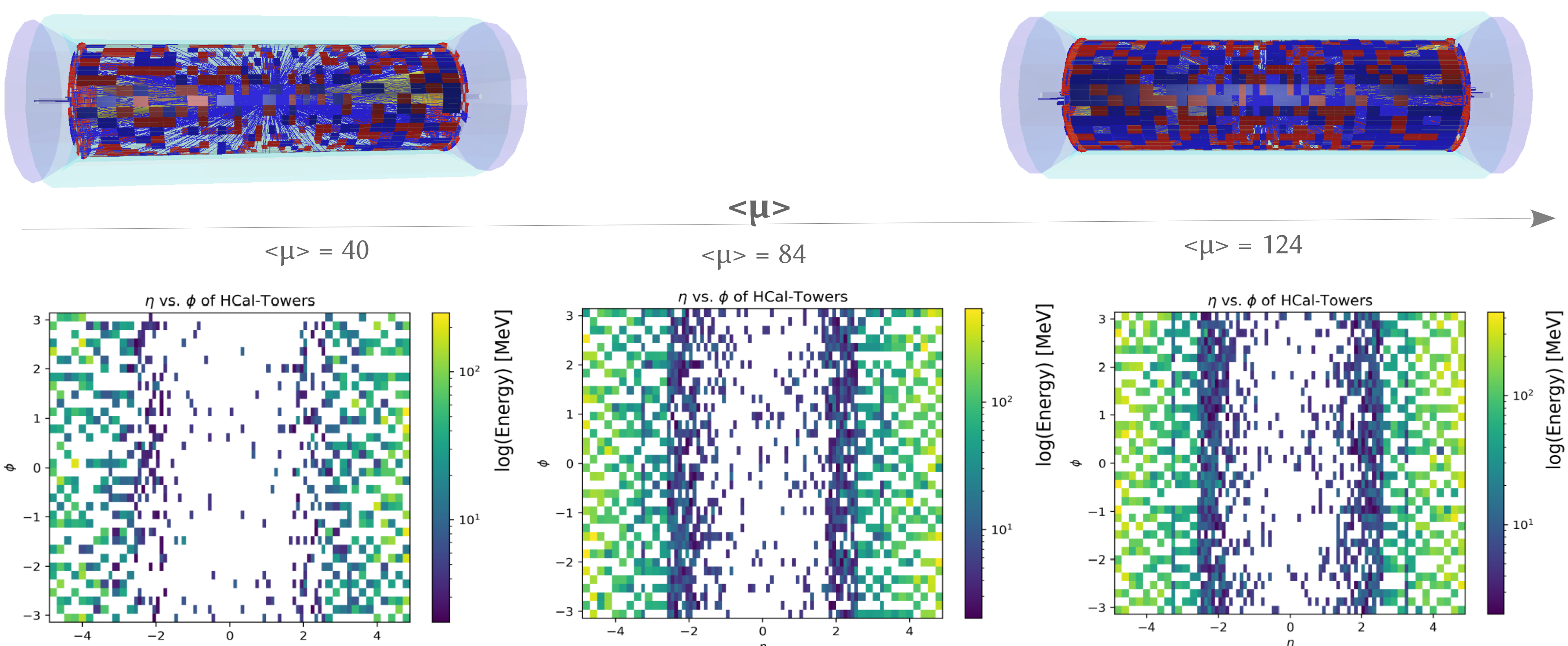
Delphes Pile-up Simulation

Brief

Problem:

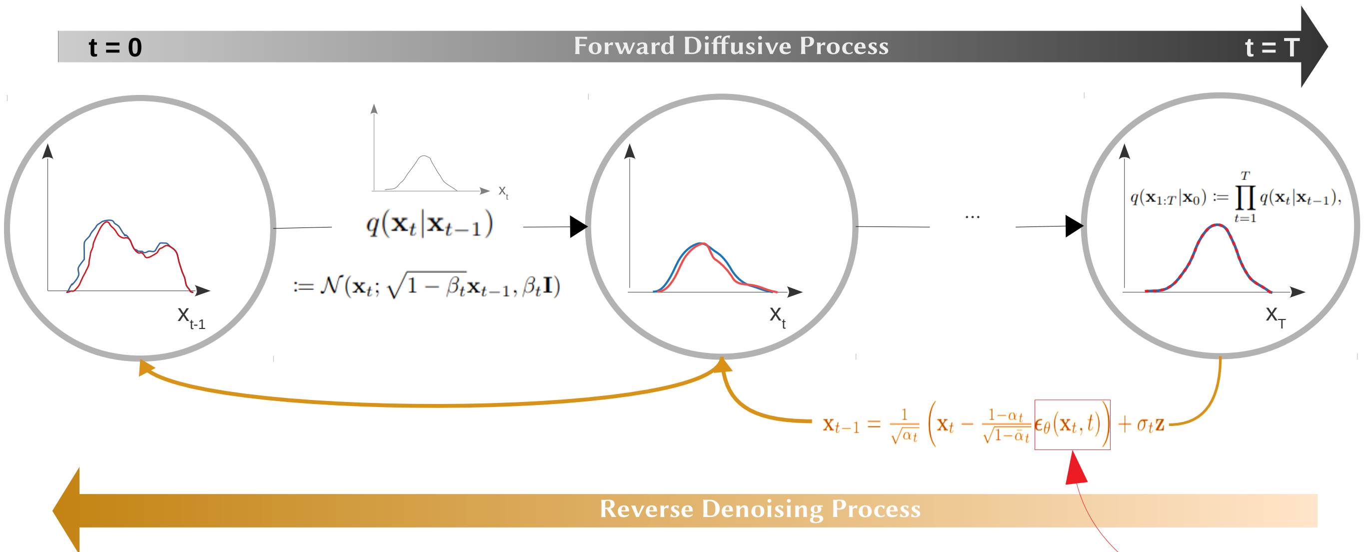
Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:

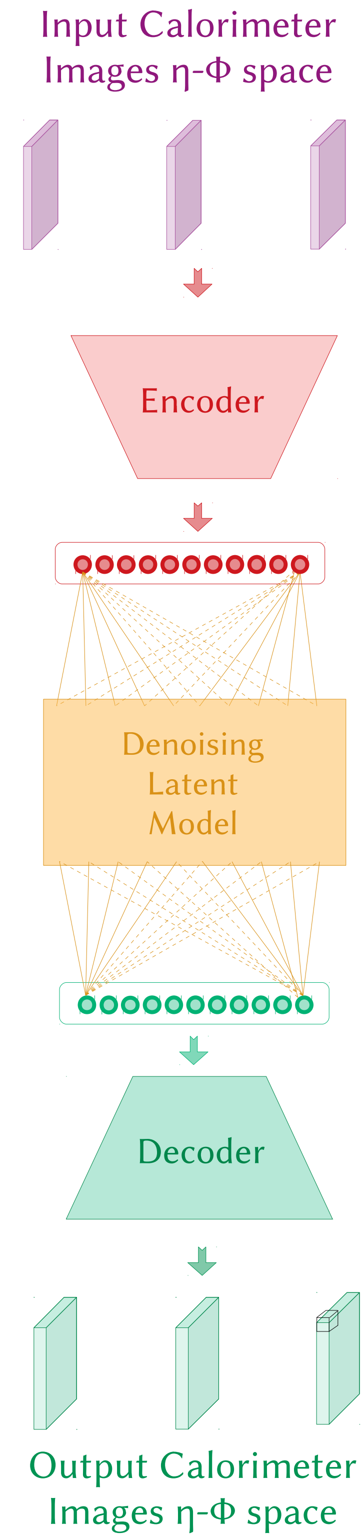




Denoise Score Matching Neural Network

DDPM Model Architecture





Denoising Neural Model

→ Image Reco. Encoder + Decoder:

Using a UNet based architecture with:

- *Cross-Attention*
- *Global Self-Attention*
- *Deep Residual Connections*

Using patch + pixel based reconstruction loss:

→ Denoising Latent Model:

Using a residual block based neural network:

- *Sinusoidal embedding of class+time*
- *Short + Long residual connections*
- *Dense (non-)-linear MLP blocks*

Using noise score matching criteria:

$$L(\theta) = \mathbb{E}_{t,x_0,\hat{x}_0,\epsilon,y}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon), y, t, \hat{x}_0)\right|^2\right]$$

Latent Diffusion Residual Network

Latent Diffusion Resi. Net

→ Residual Neural Network

Deep embedding blocks to prevent mode collapse. Therefore need residual connections to overcome lost gradients

→ Simple Sinusoidal Embedding

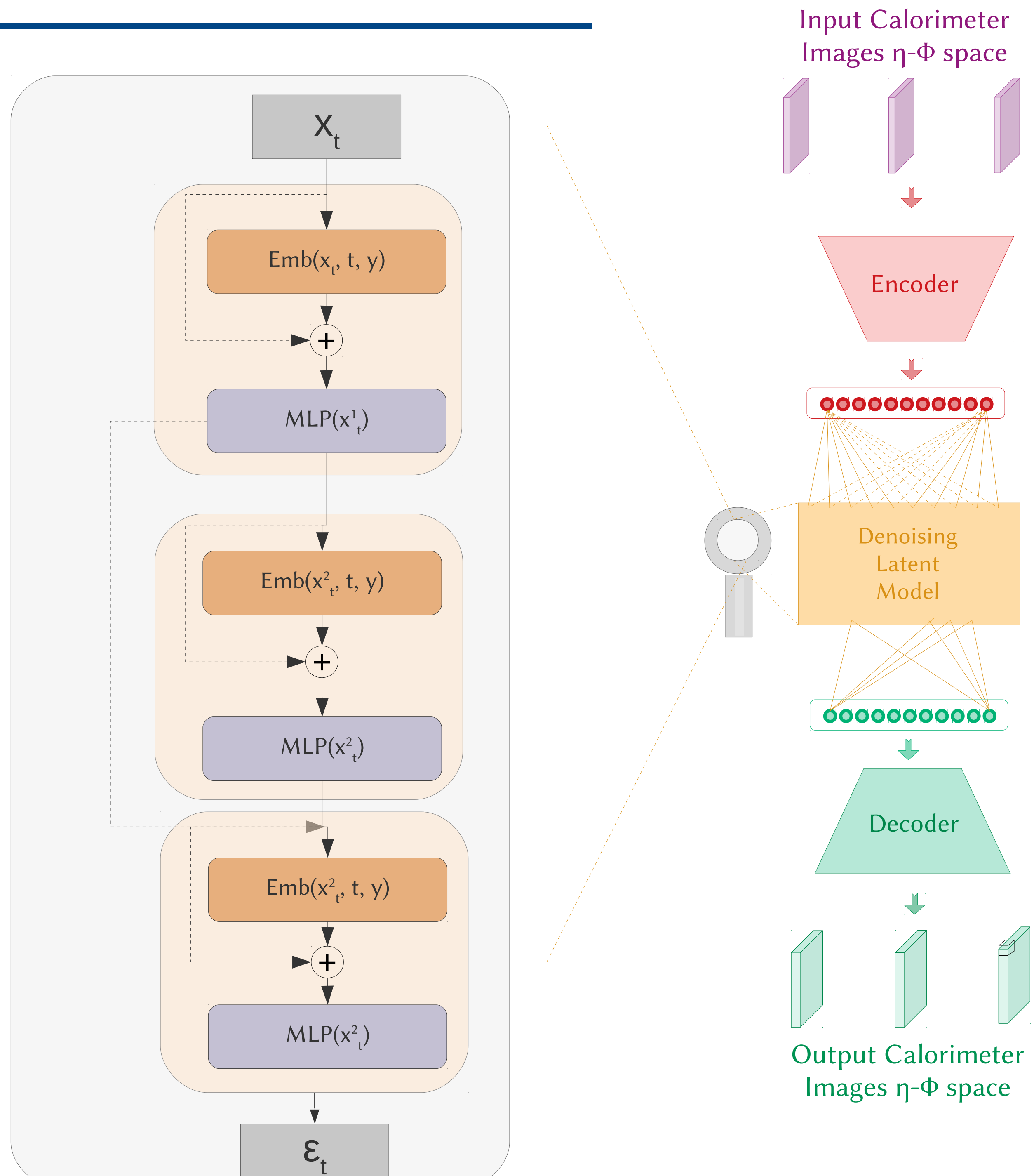
$$PE_{t,2i} = \sin\left(t/\alpha^{2i/d_{emb}}\right)$$

$$PE_{t,2i+1} = \cos\left(t/\alpha^{(2i+1)/d_{emb}}\right)$$

Natural normalisation of conditional variables that scales dimensionally with embedding space

→ FFN Blocks

Simple dense feed-forward neural network blocks for dimensionality scaling and expressibility



Latent Diffusion Residual Network

Latent Diffusion Resi. Net

→ **Residual Neural Network**

Deep embedding blocks to prevent mode collapse. Therefore need residual connections to overcome lost gradients

→ **Simple Sinusoidal Embedding**

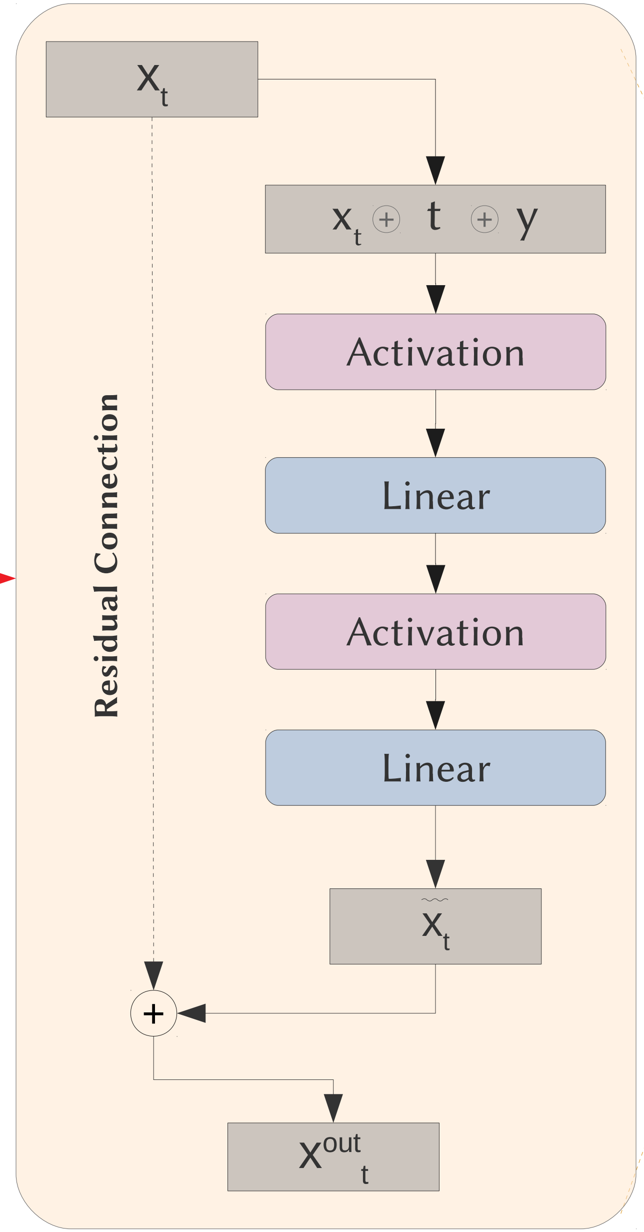
$$PE_{t,2i} = \sin\left(t/\alpha^{2i/d_{emb}}\right)$$

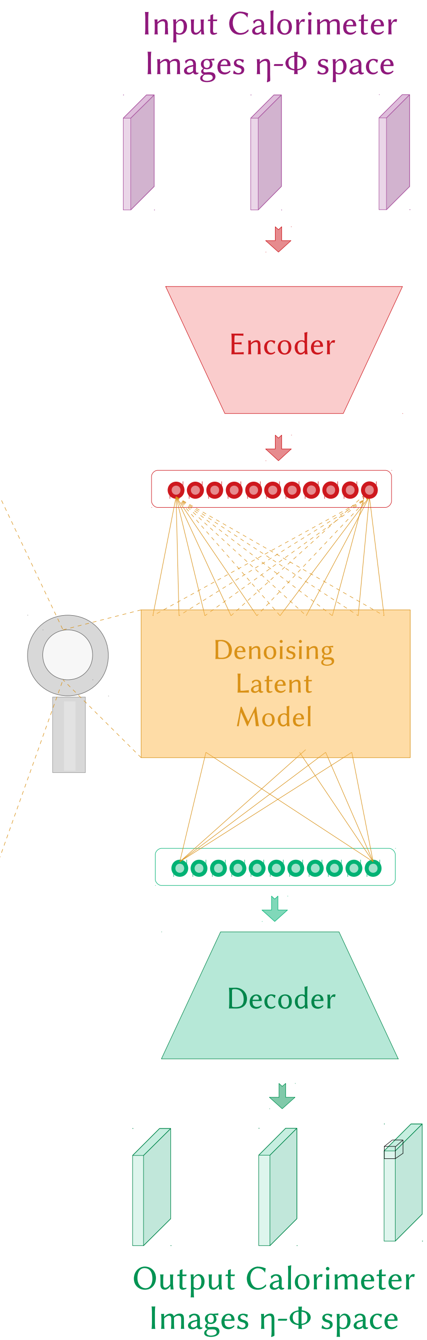
$$PE_{t,2i+1} = \cos\left(t/\alpha^{(2i+1)/d_{emb}}\right)$$

Natural normalisation of conditional variables that scales dimensionally with embedding space

→ **FFN Blocks**

Simple dense feed-forward neural network blocks for dimensionality scaling and expressibility



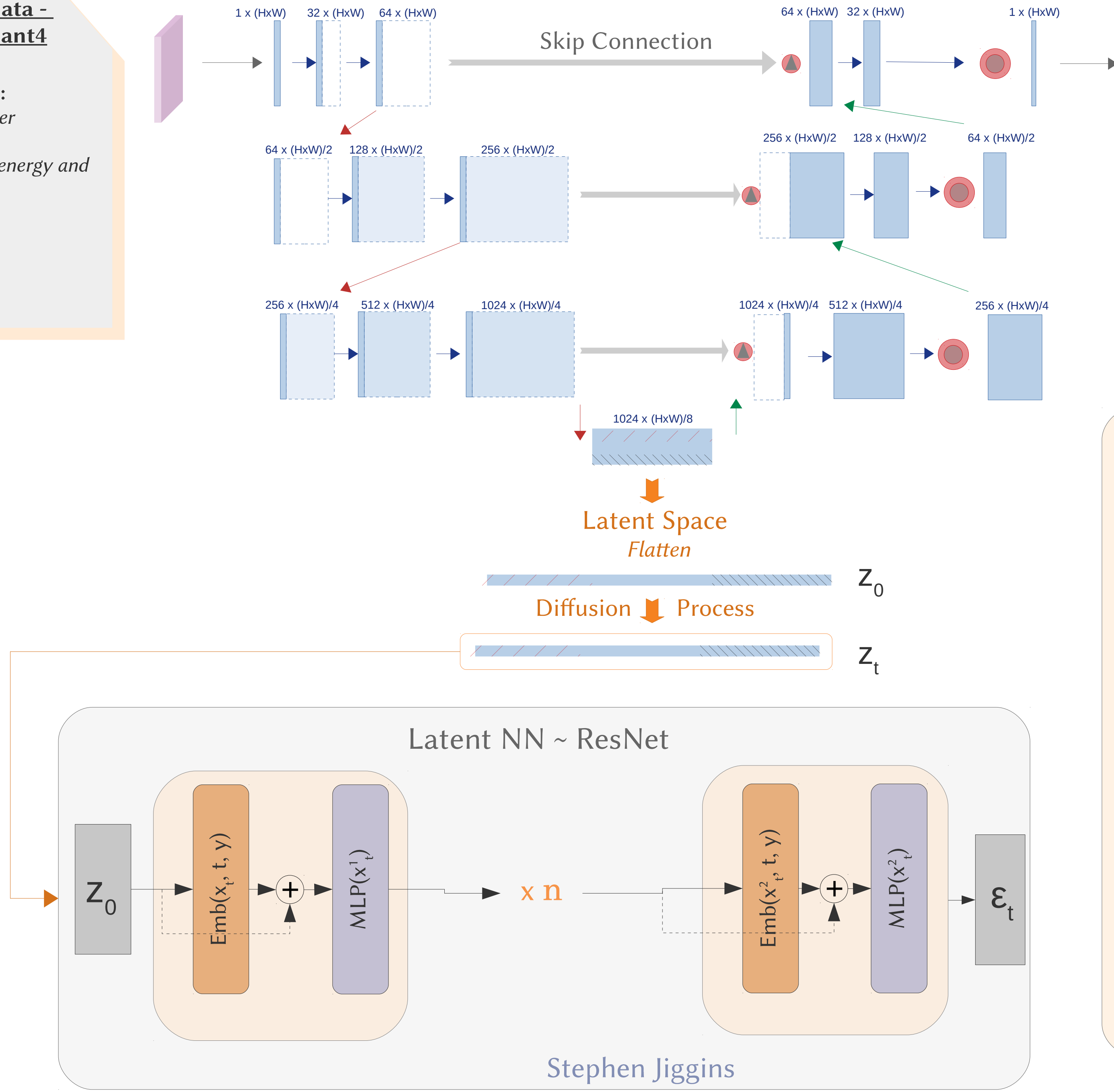


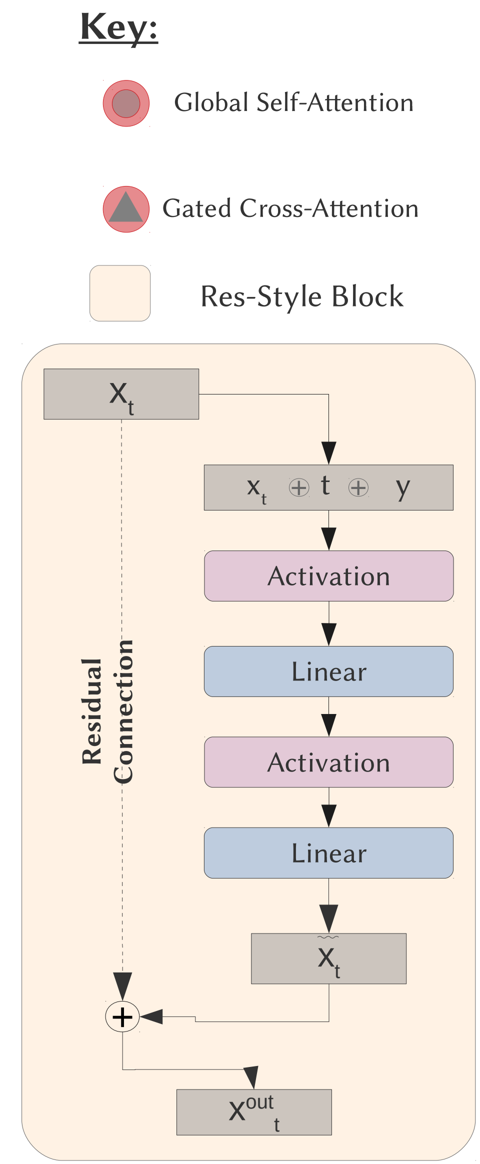
Latent DDPM

Calorimetry Data - Delphes/Geant4

→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

→ **Delphes:**
- → *ECAL*
- → *HCAL*

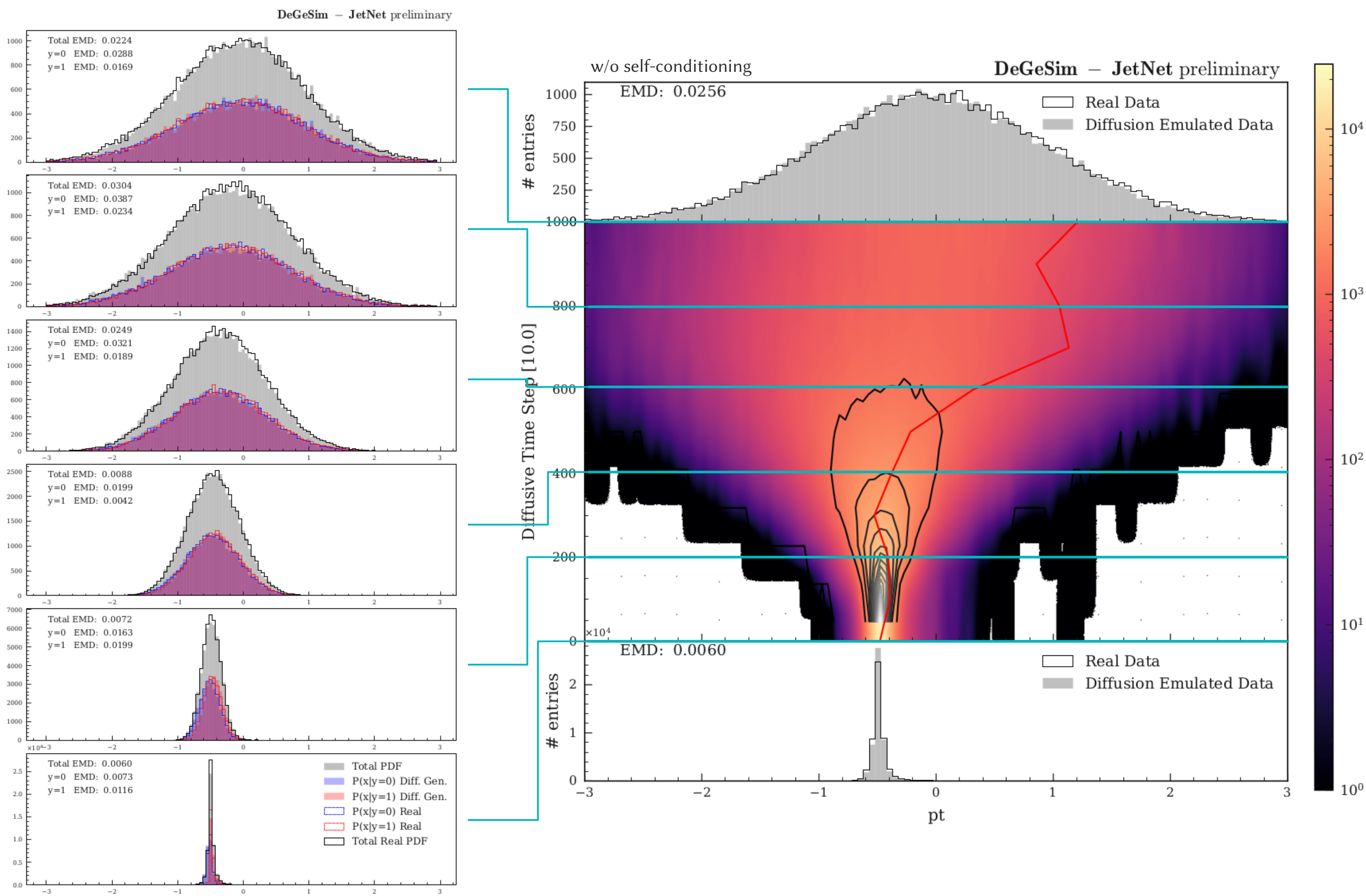




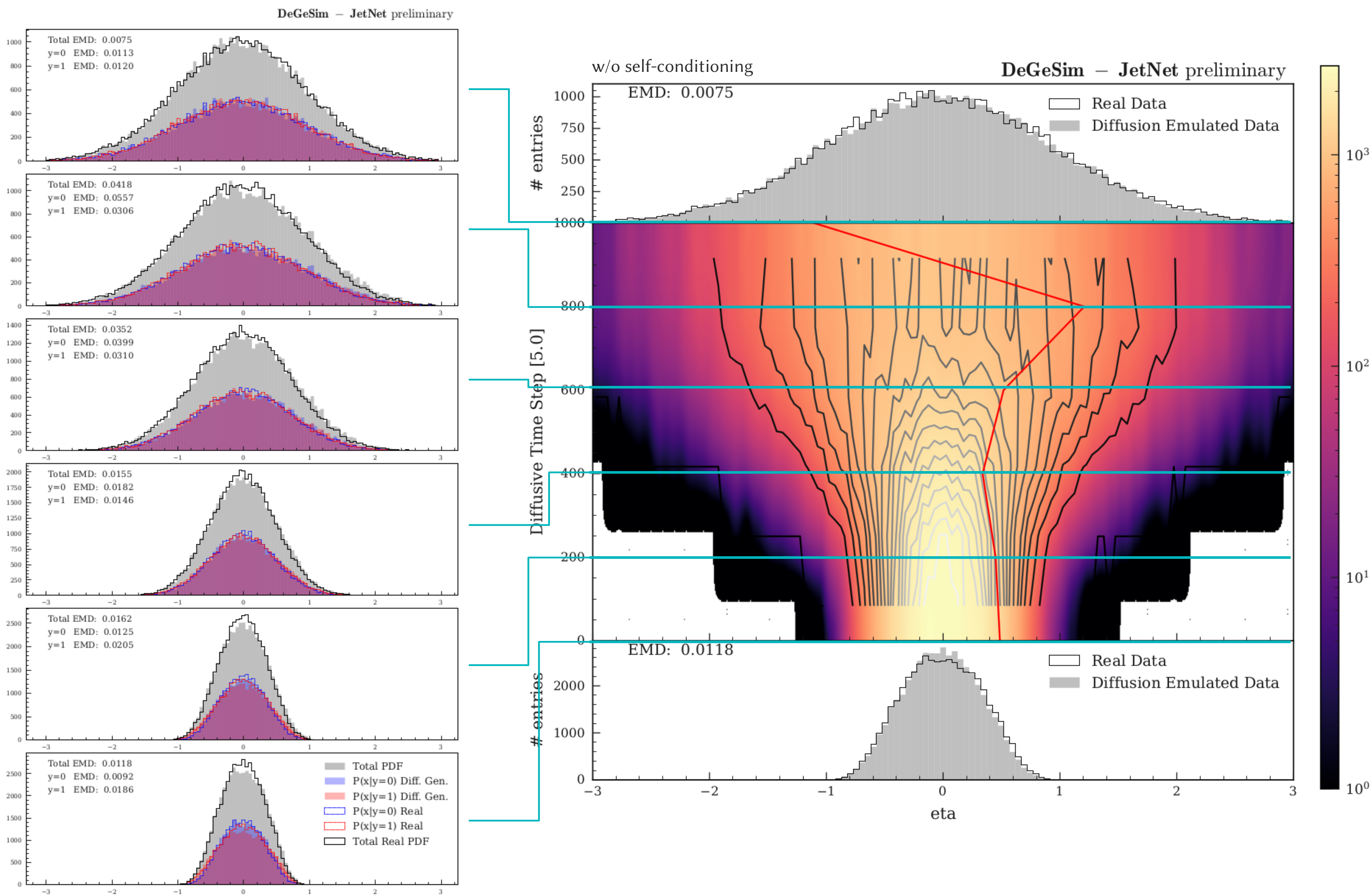
PDF Heat Map – Time Profile

w/ self-conditioning

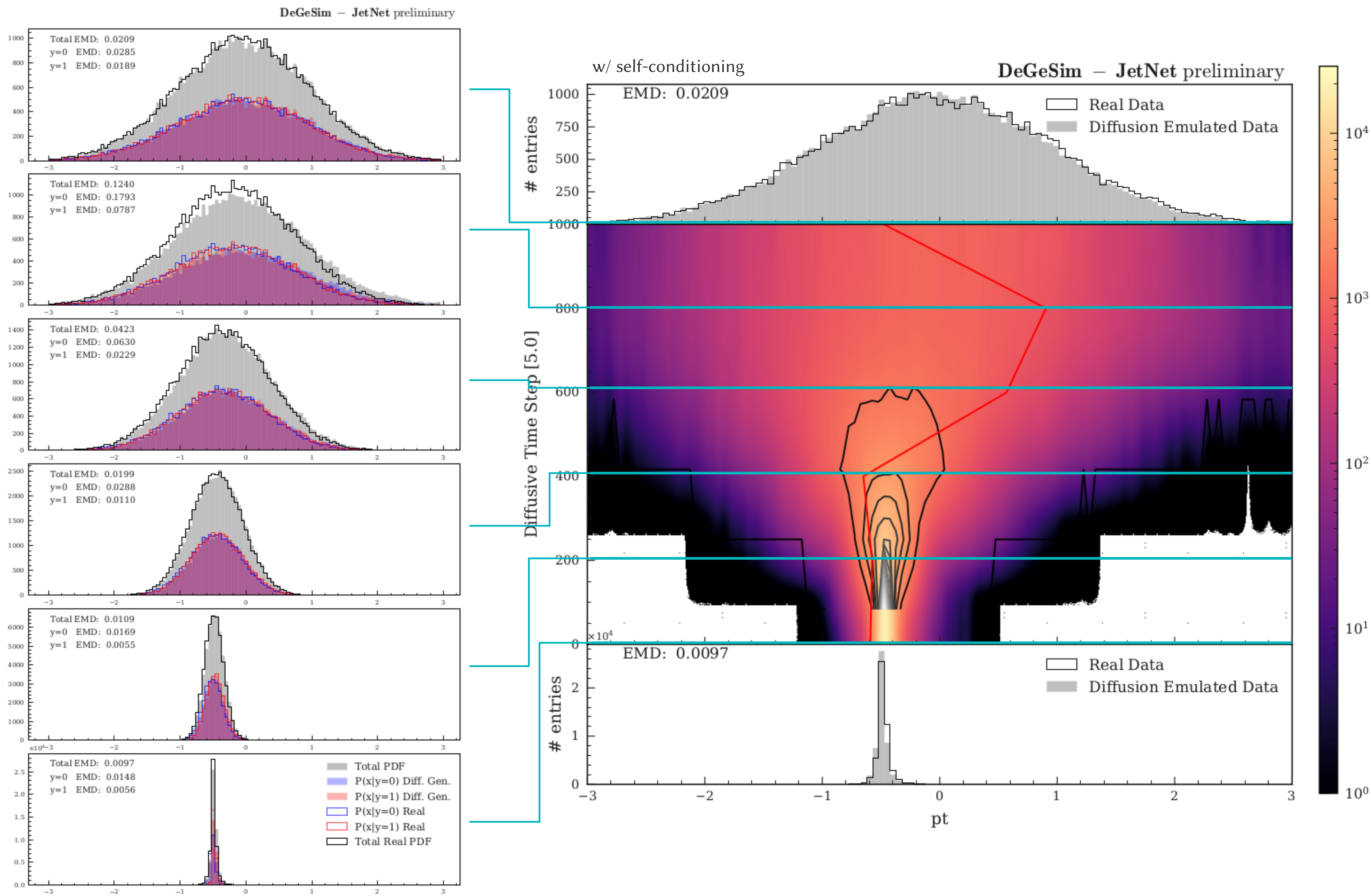
JetNet: Top -vs- W jets



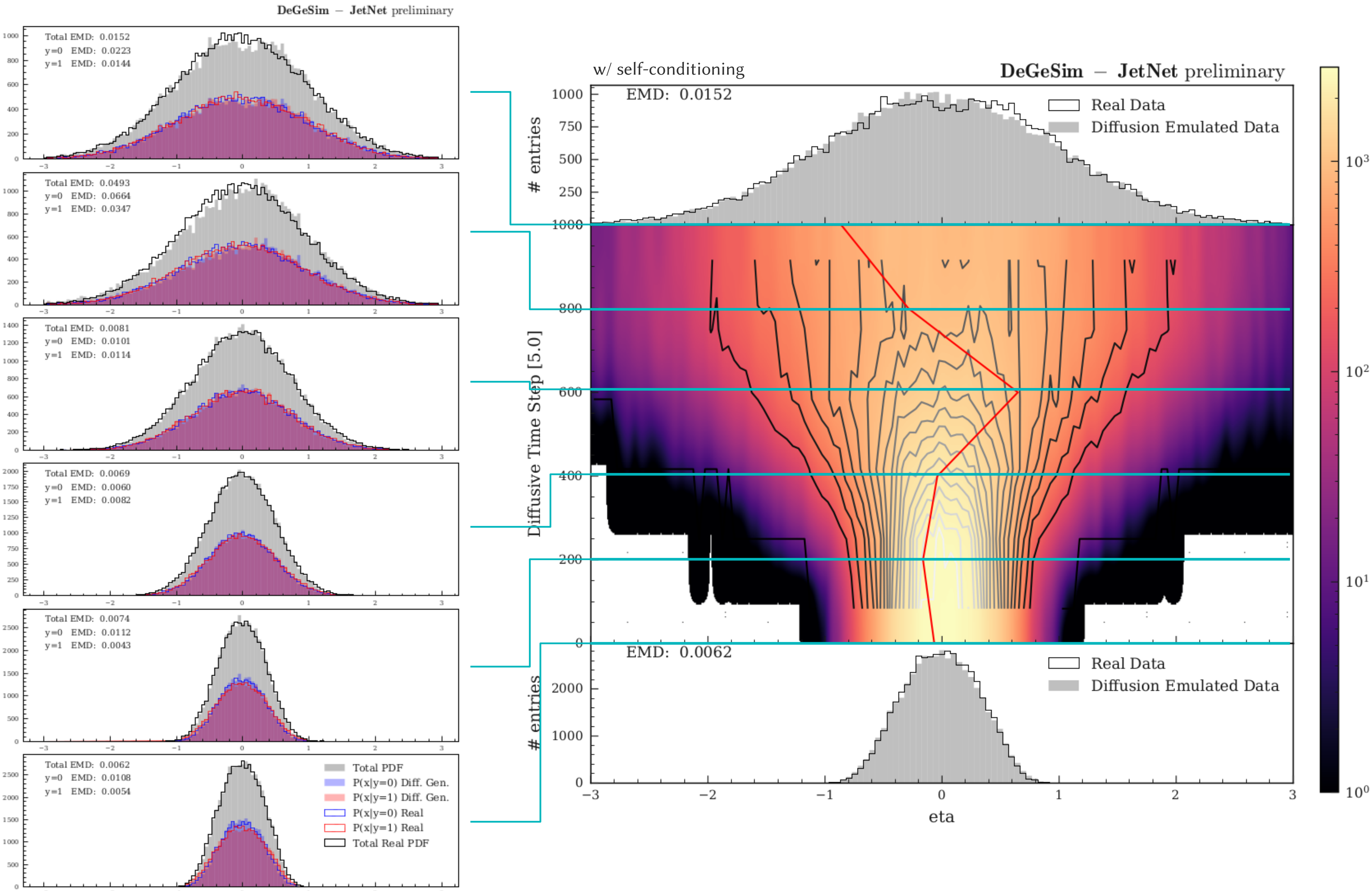
JetNet: Top -vs- W jets



JetNet: Top -vs- W jets

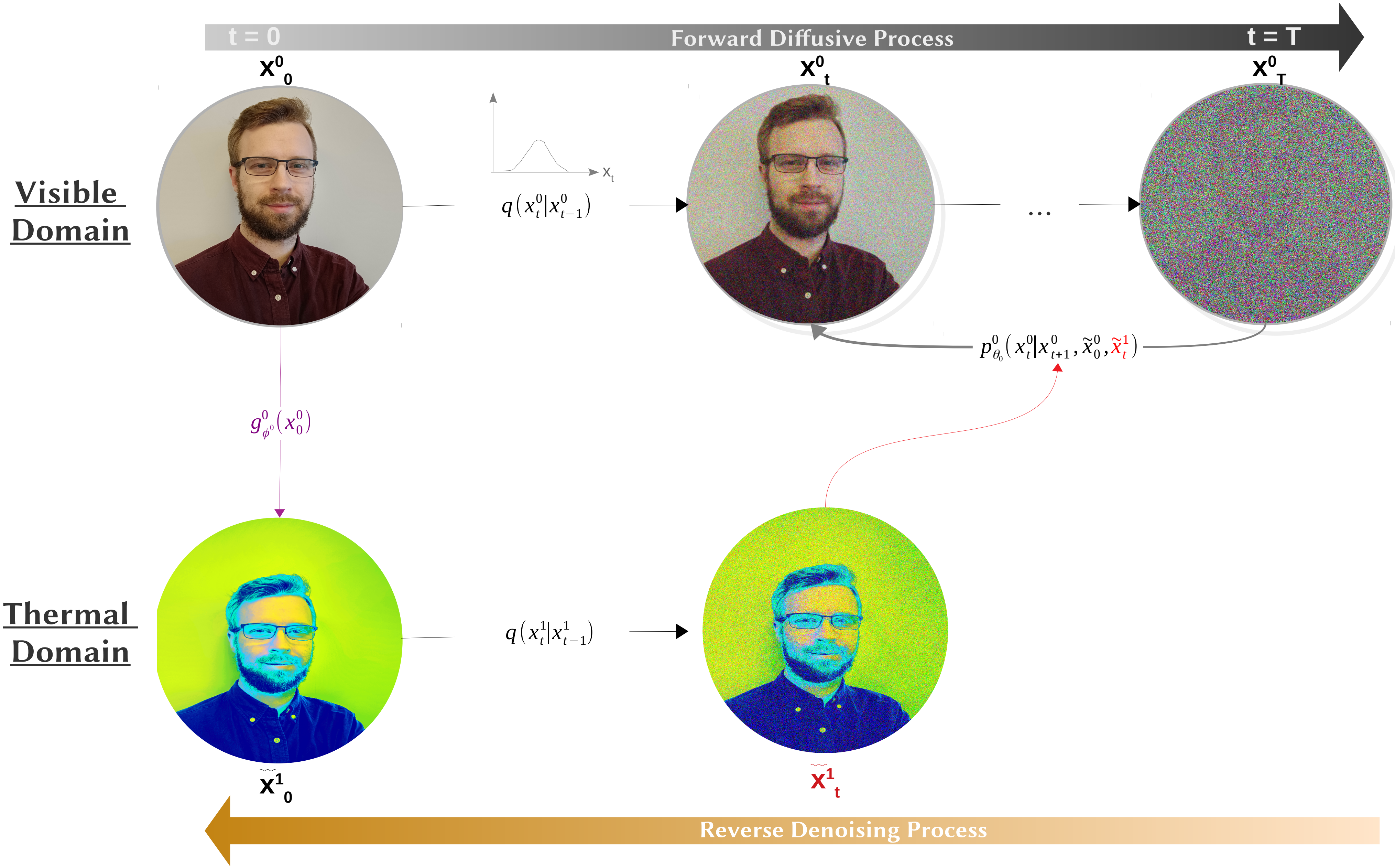


JetNet: Top -vs- W jets



Cross Domain Attention

Unpaired Domain Adaptation DDPM - I2I



Unpaired Domain Adaptation DDPM - I2I