DeGeSim – Pileup Modeling using Conditional Denoising Probabilistic Models

14-06-24

Stephen Jiggins – DESY
Judith Katzy - DESY

HELMHOLTZ AI
SUPPORTED PROJECT

Contents

Contents

01

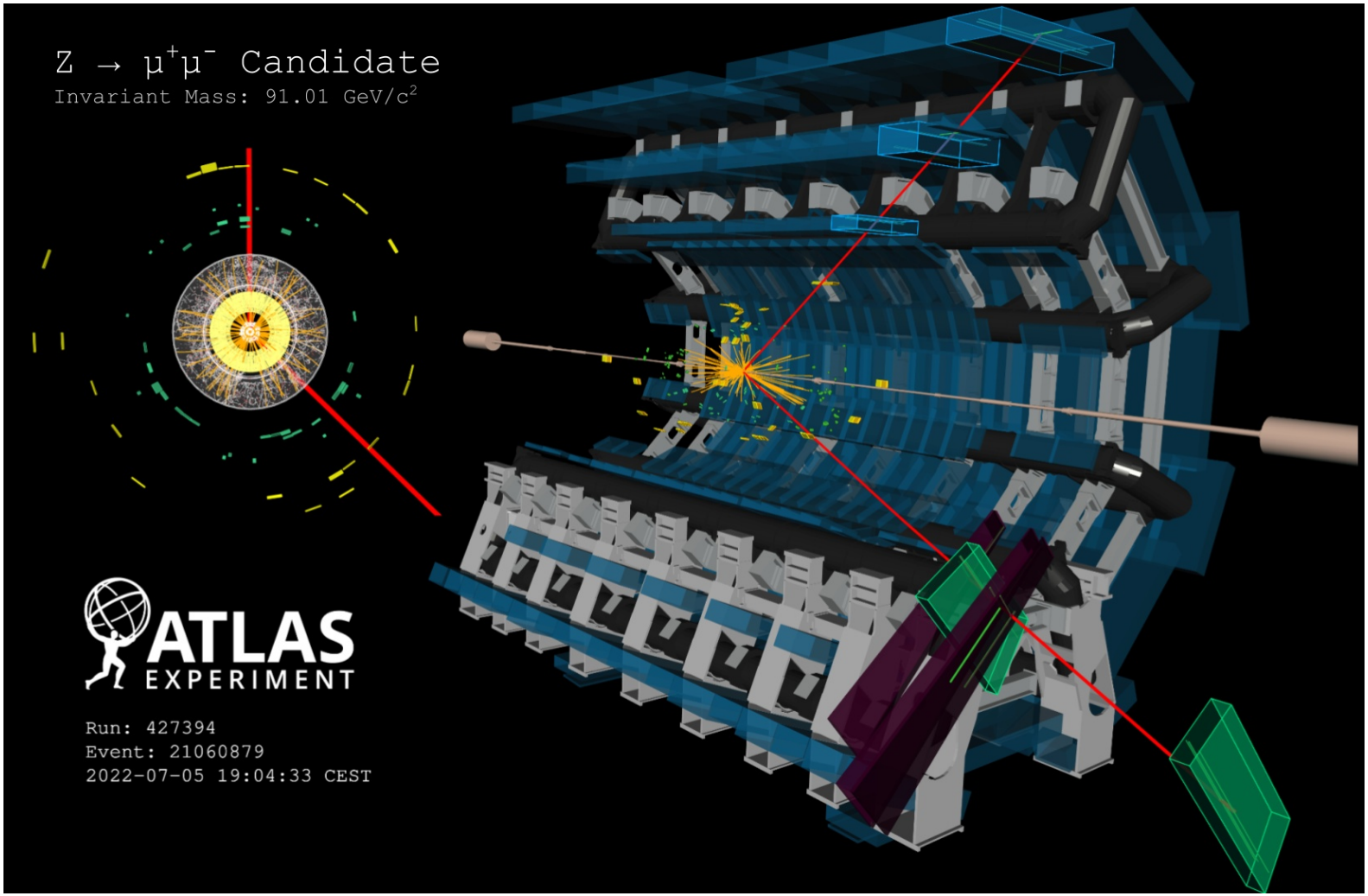
Introduction:
Pileup ~ Noise

Brief

Problem:

The Large Hadron Collider (LHC) smashes bunches of protons (~10^{11}) approximately every 25ns. At present this yields ~40 collisions per crossing, in the High Luminosity LHC this will increase to ~200

⟶ This is pileup ~ noise

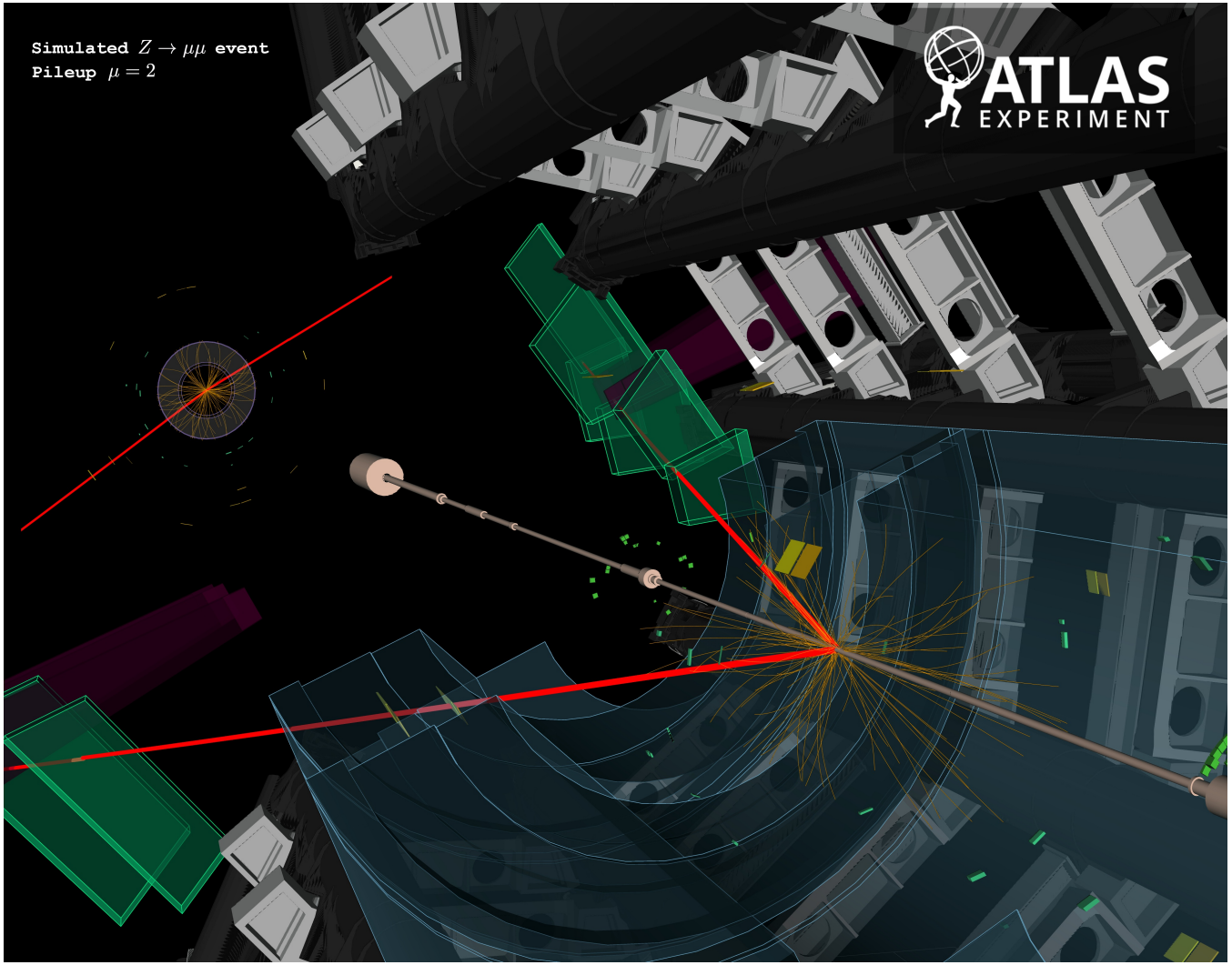


Brief

Problem:

The Large Hadron Collider (LHC) smashes bunches of protons (~10^{11}) approximately every 25ns. At present this yields ~40 collisions per crossing, in the High Luminosity LHC this will increase to ~200

⟶ This is pileup ~ noise

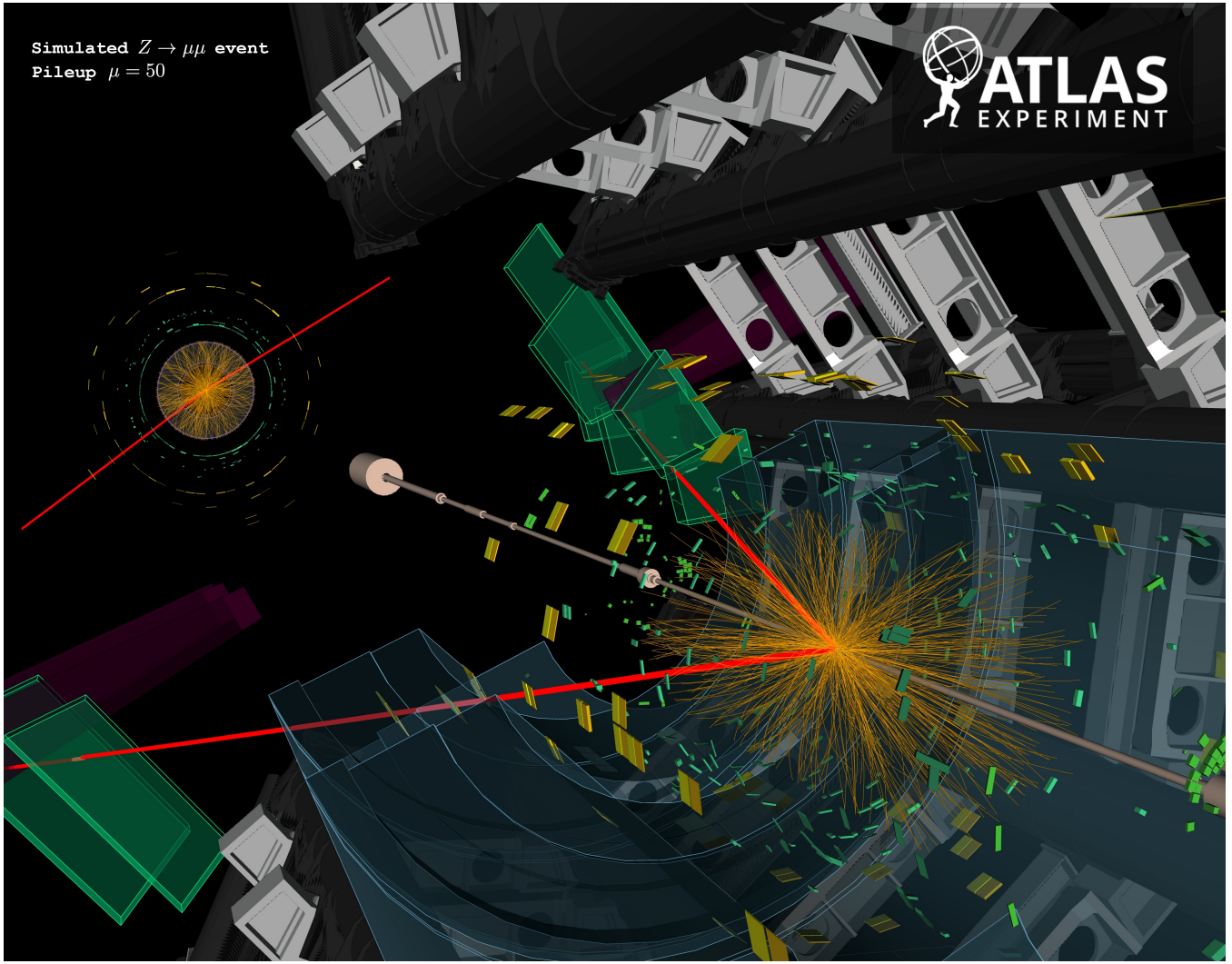


Brief

Problem:

The Large Hadron Collider (LHC) smashes bunches of protons (~10^{11}) approximately every 25ns. At present this yields ~40 collisions per crossing, in the High Luminosity LHC this will increase to ~200

⟶ This is pileup ~ noise

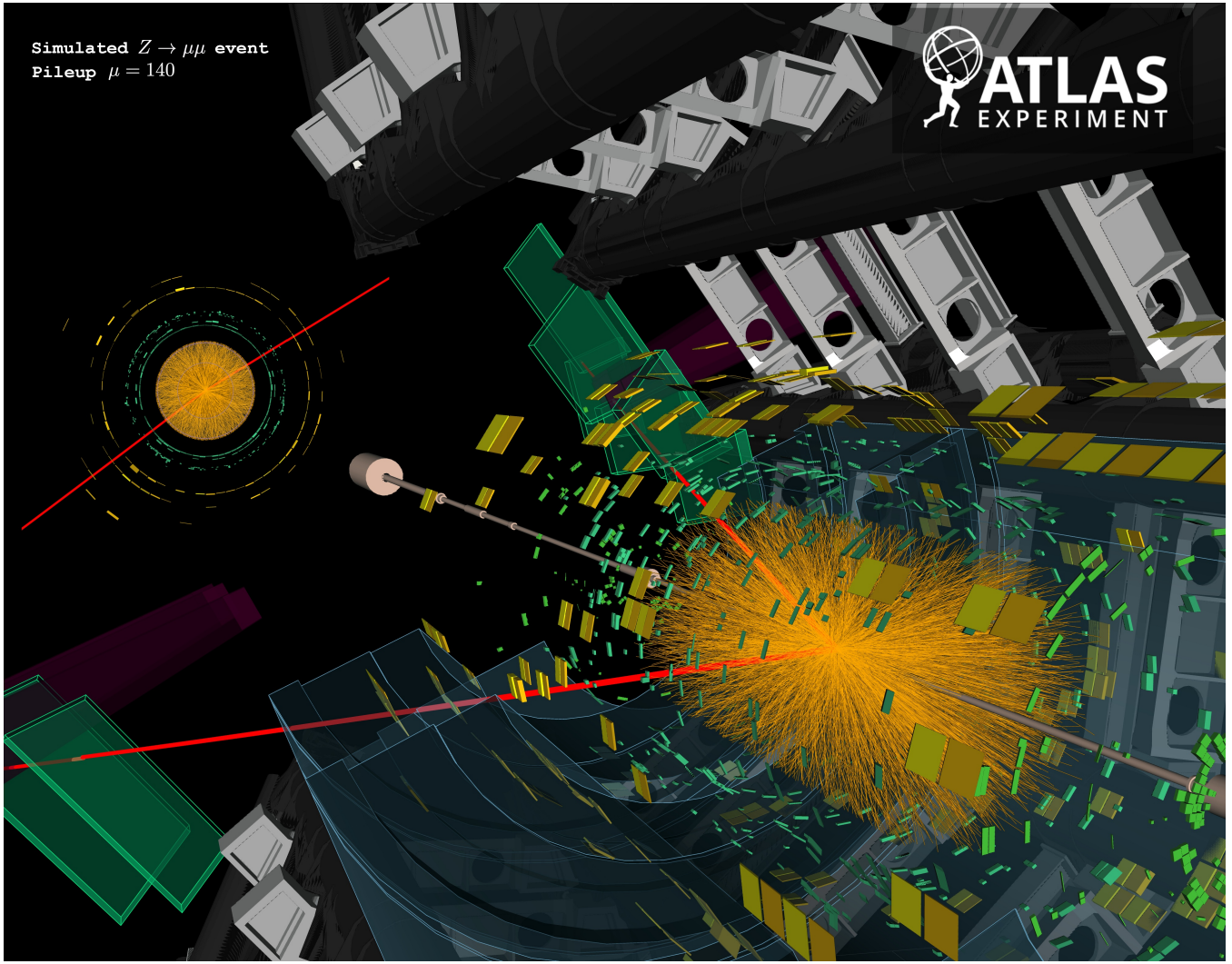


Brief

Problem:

The Large Hadron Collider (LHC) smashes bunches of protons (~10^{11}) approximately every 25ns. At present this yields ~40 collisions per crossing, in the High Luminosity LHC this will increase to ~200

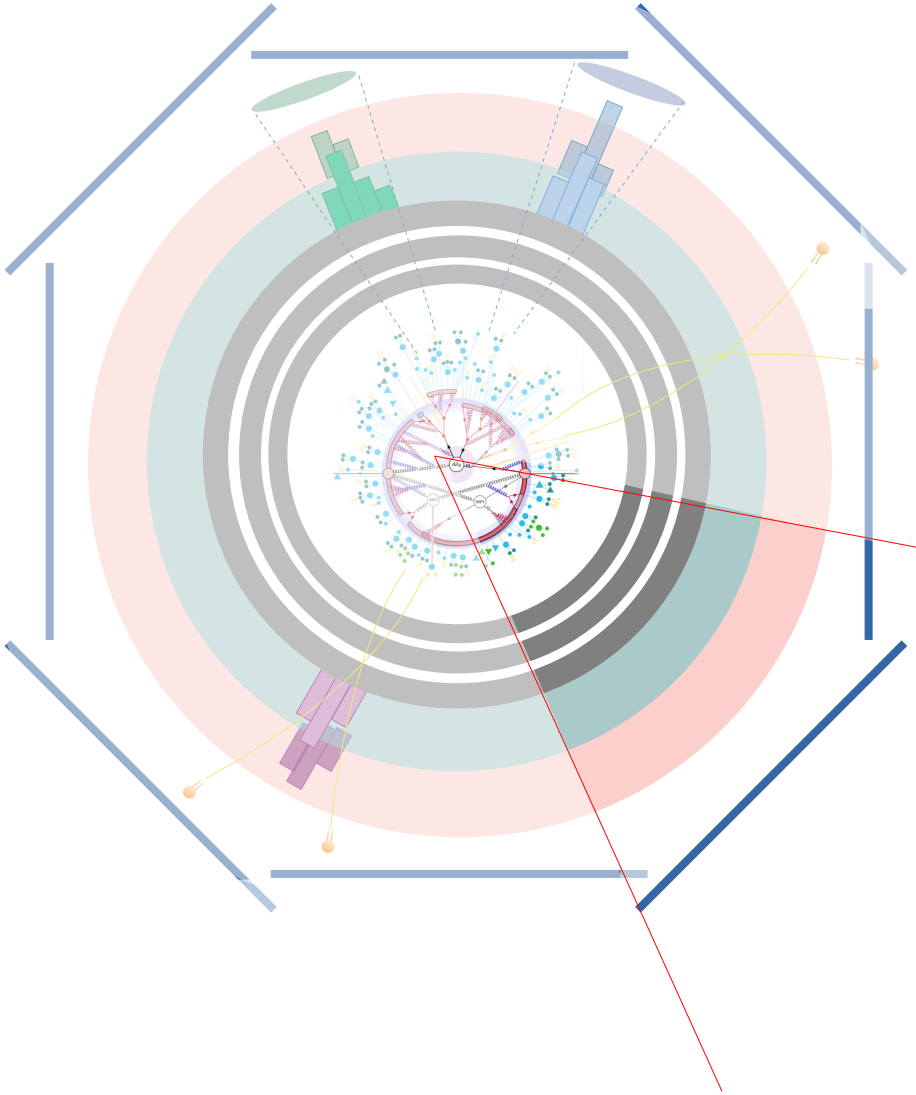
⟶ This is pileup ~ noise

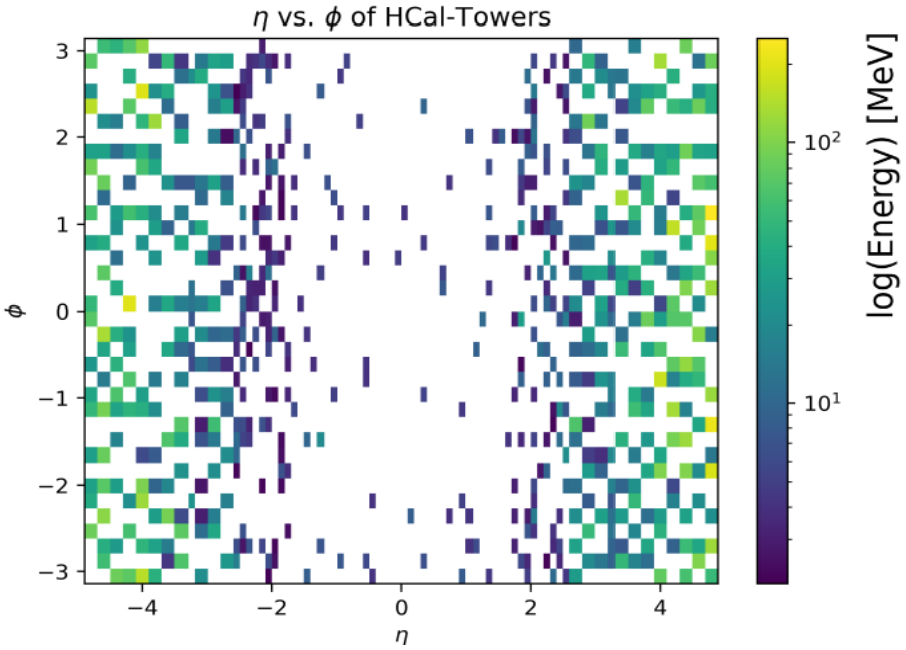


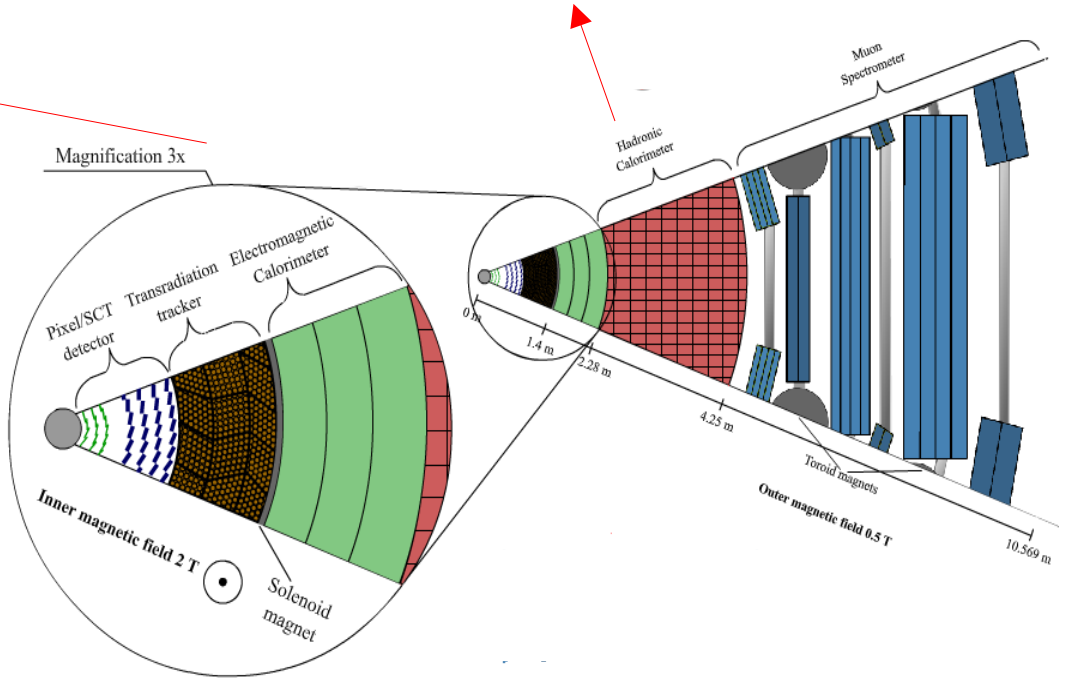
DeGeSim

Project Aim:

Generative modeling of ATLAS calorimeter response due to pileup interactions using ML image synthesis techniques







DeGeSim

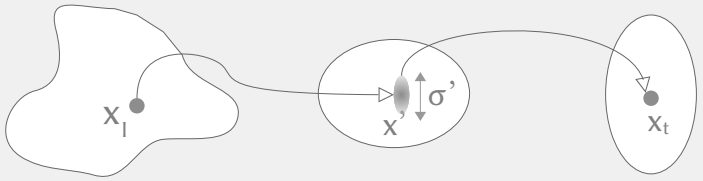
Project Aim:

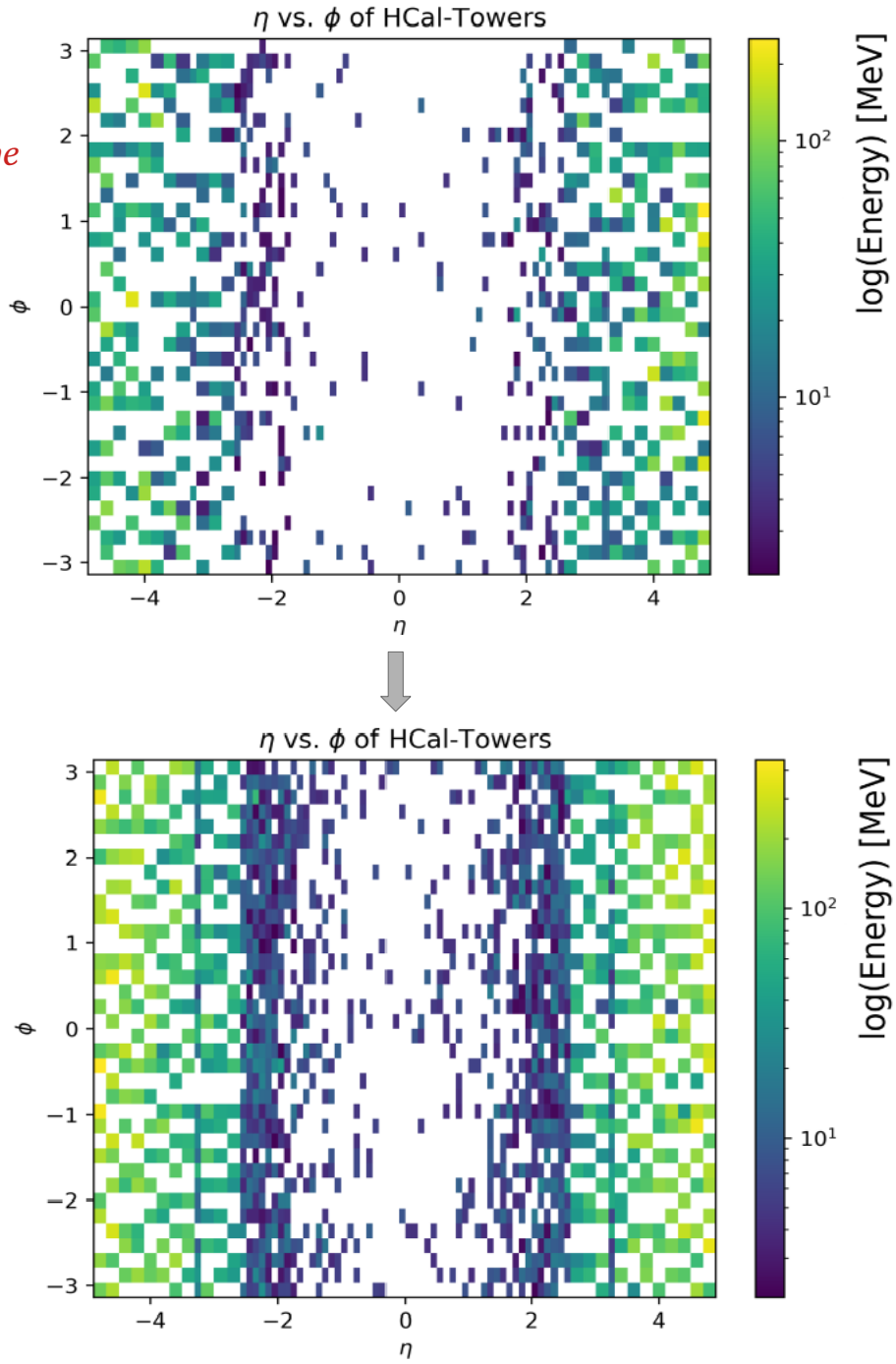
Generative modeling of ATLAS calorimeter response due to pileup interactions using ML image synthesis techniques, but map from one image class to another

Image-to-Image Generation

Transform from a image $\boldsymbol{X}_{\mathrm{I}}$ *sampled from a distribution* $P_{\mathrm{I}}(\bar{x}\,|\,\bar{\theta})$ *to a target image* $\boldsymbol{X}_{\mathrm{t}}$ sampled from $P_{\mathrm{t}}(\bar{x}\,|\,\bar{\theta})$*:*

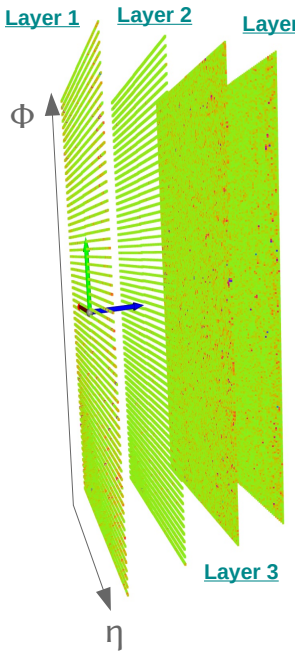
$$f:\ \boldsymbol{X}_{\mathrm{I}} \longrightarrow \boldsymbol{X}_{\mathrm{t}}$$

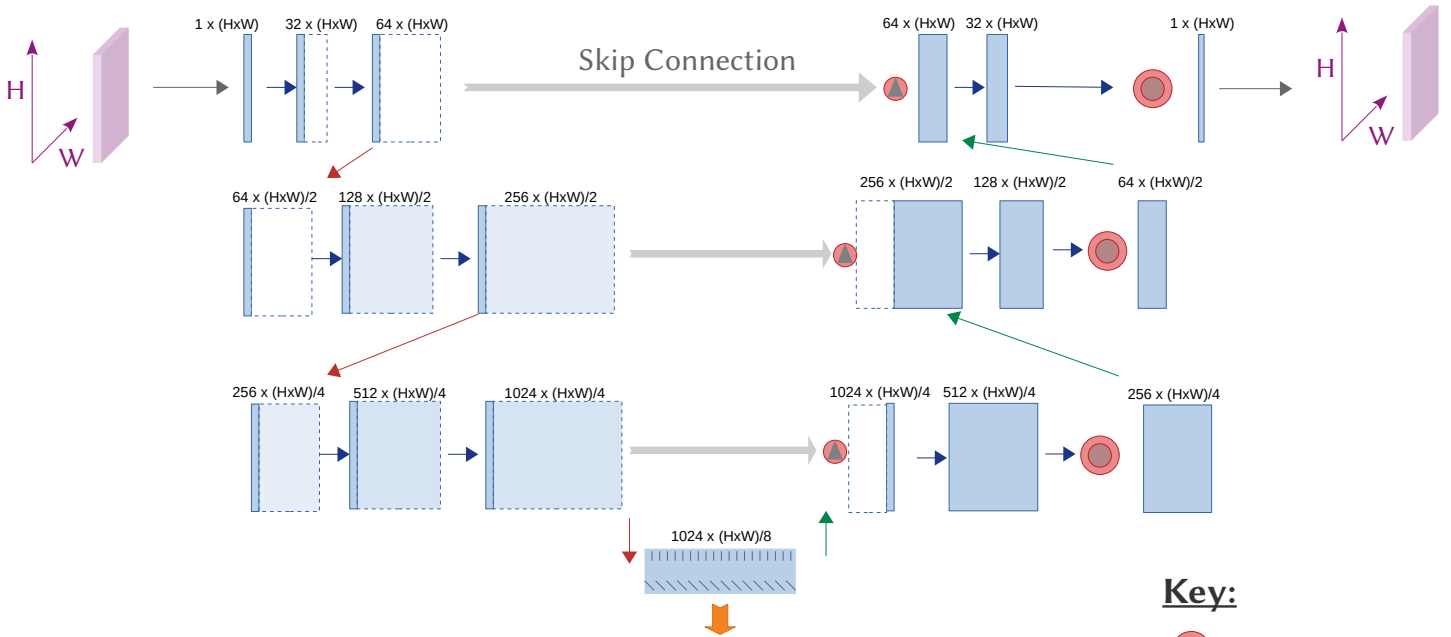




02 Model Overview

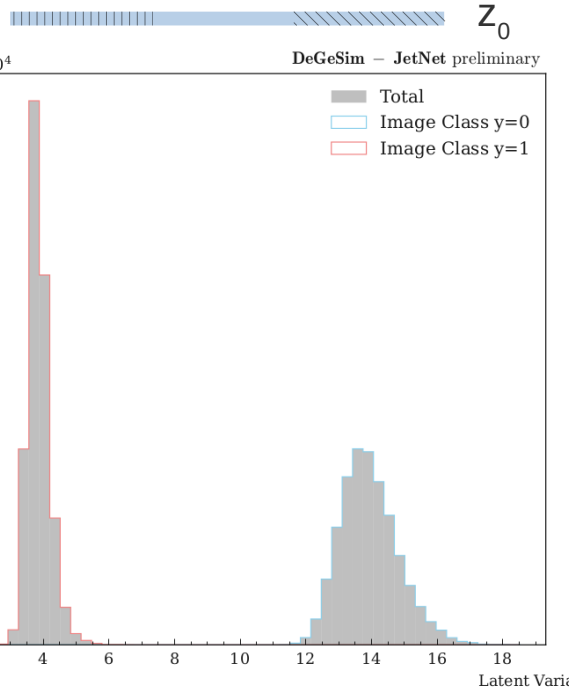
Calorimeter Image Model





Key:

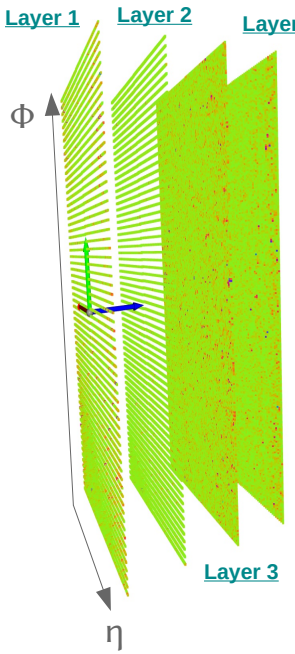
- Global Self-Attention
- Gated Cross-Attention

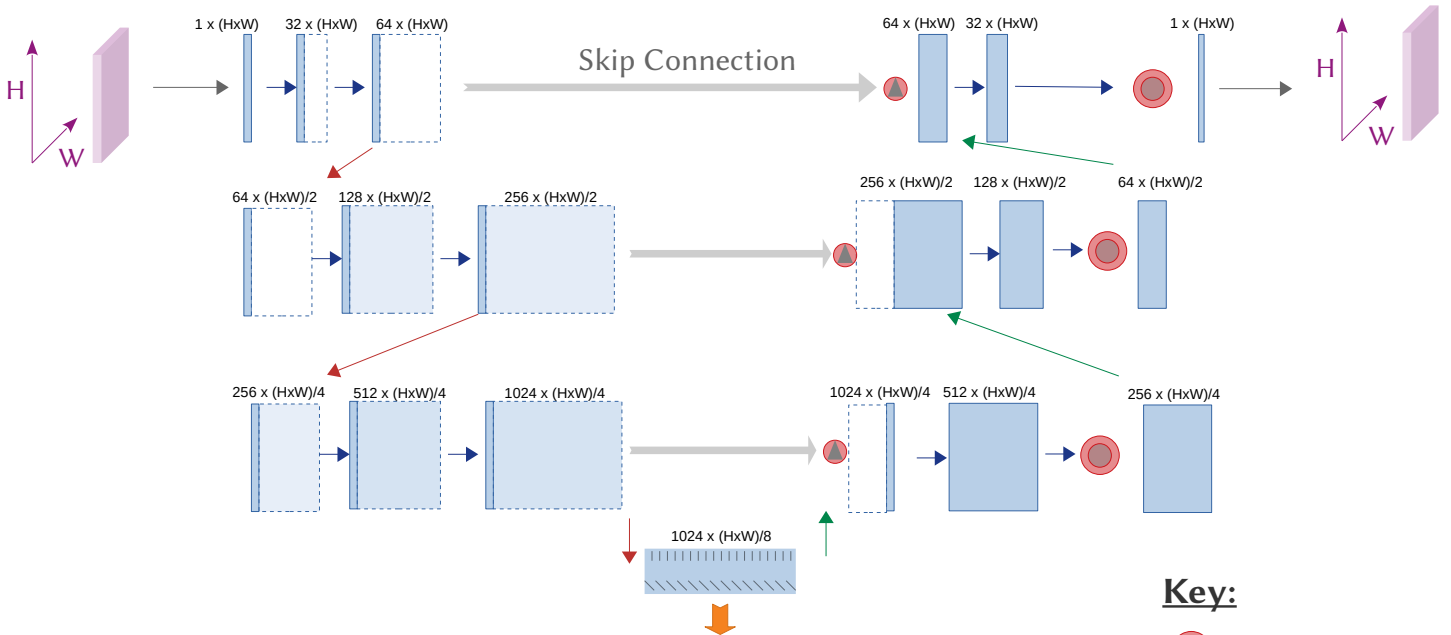


→ **Image Structure:**
- → 3D image
- → Approximately175,000 cells ~ pixels

→ **Image Compression Model:**
- → Encoder → Latent → Decoder
- → 2D planar CNN based model

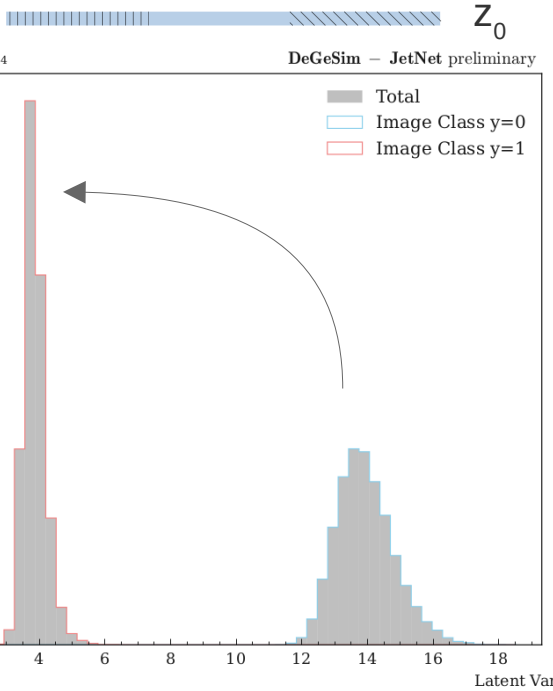
Calorimeter Image Model





Key:

- Global Self-Attention
- Gated Cross-Attention



→ **Image Structure:**
- → 3D image
- → Approximately175,000 cells ~ pixels

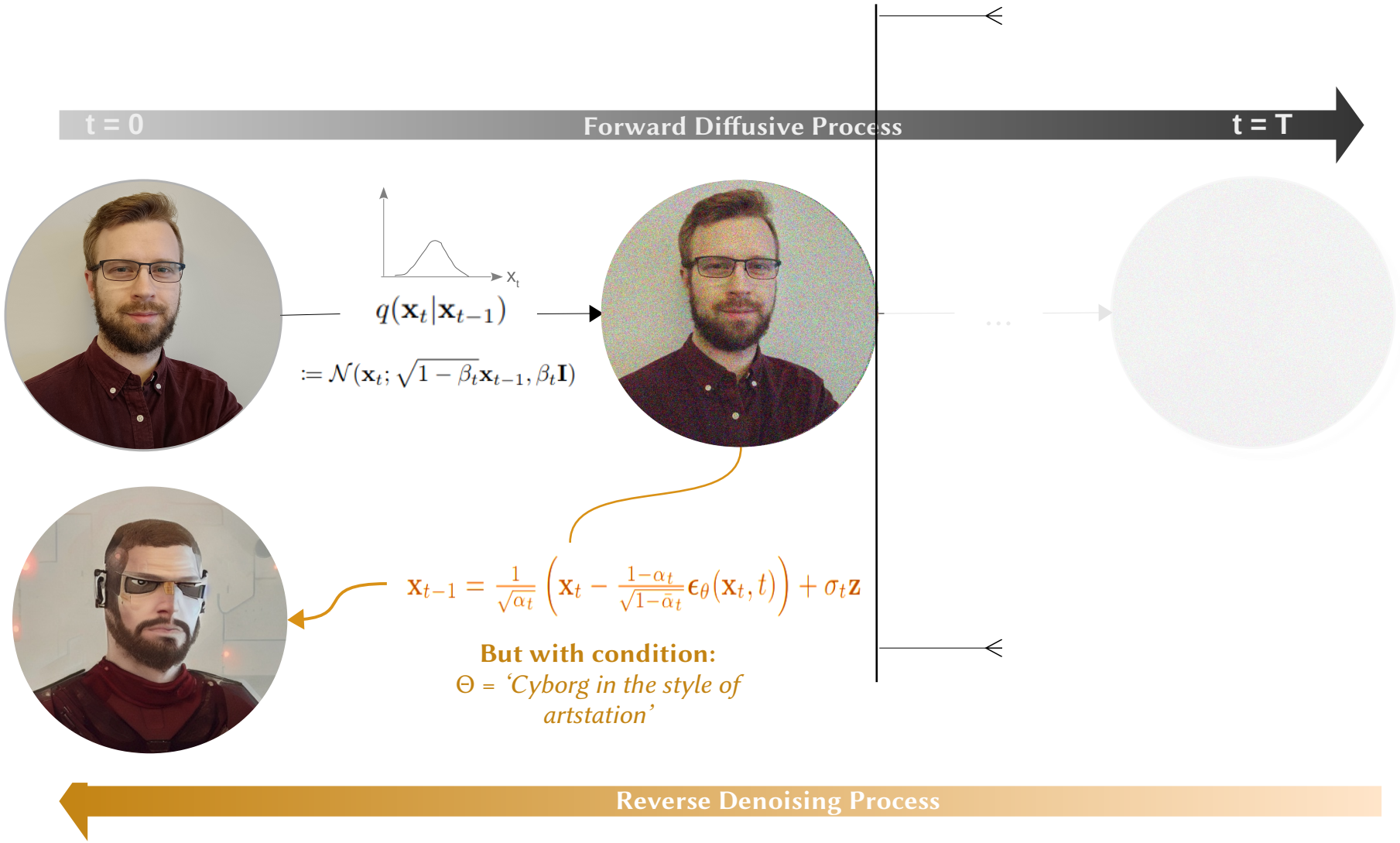
→ **Image Compression Model:**
- → Encoder → Latent → Decoder
- → 2D planar CNN based model

The task is to make *blue* look like *red*:

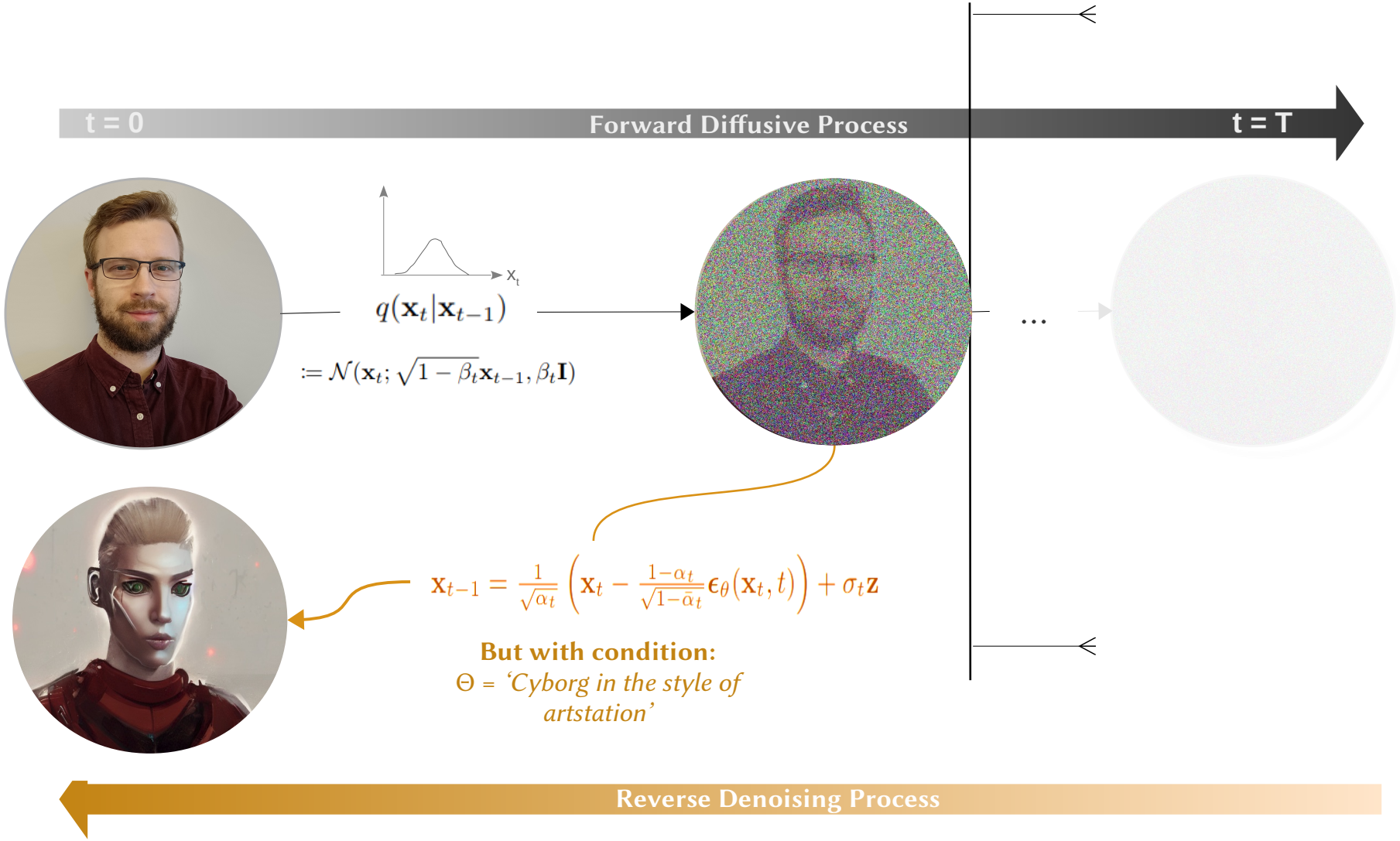
$f: X_0 \rightarrow X_t$

03 Latent Denoising Diffusion Probabilistic Models

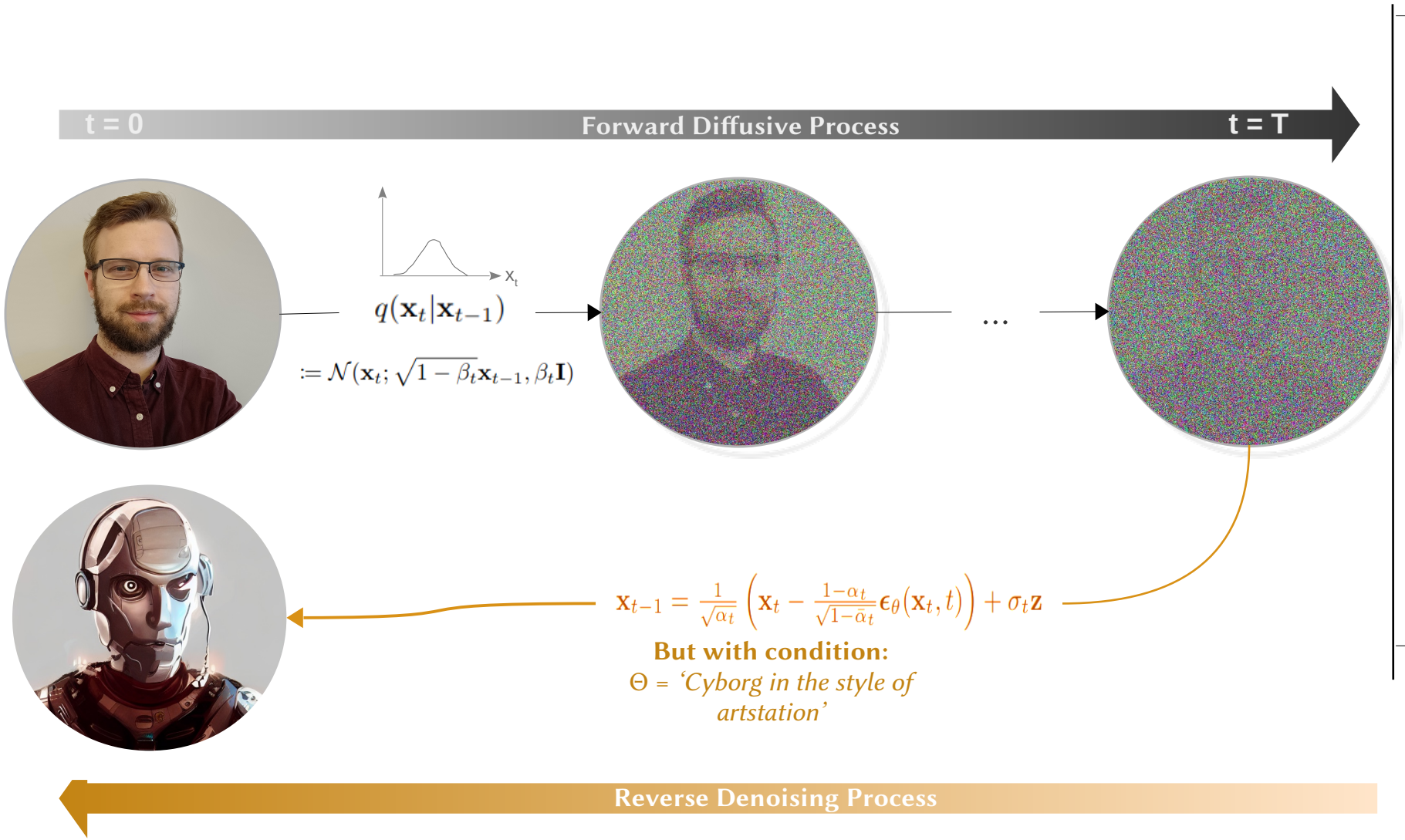
Partial Diffusion Probabilistic Model



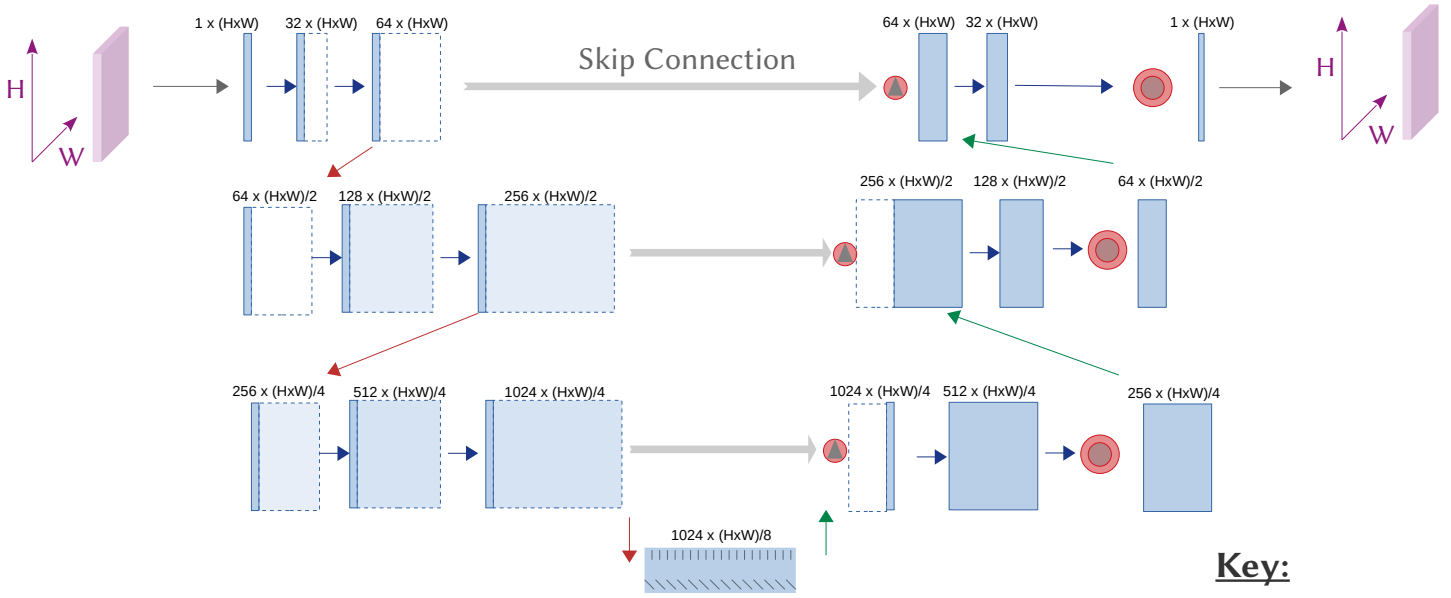
Partial Diffusion Probabilistic Model



Partial Diffusion Probabilistic Model



Latent DDPM Calo. Imaging



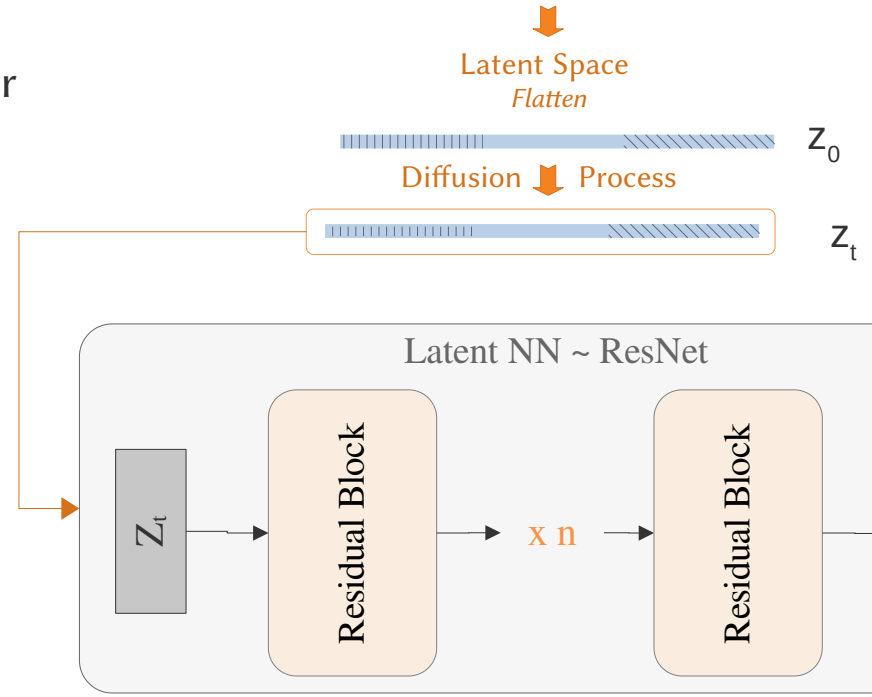
→ **Image Compression Model:**

Encoder → Latent → Decoder

2D planar CNN based model

→ **Latent Denoising Model:**

+ Residual neural network
  - Critical for time + class conditioning
+ Classifier-free Guidance
+ Self-Conditioning

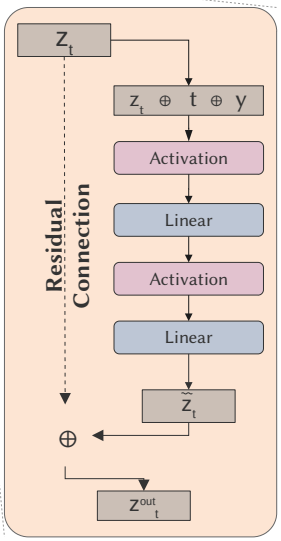


Key:

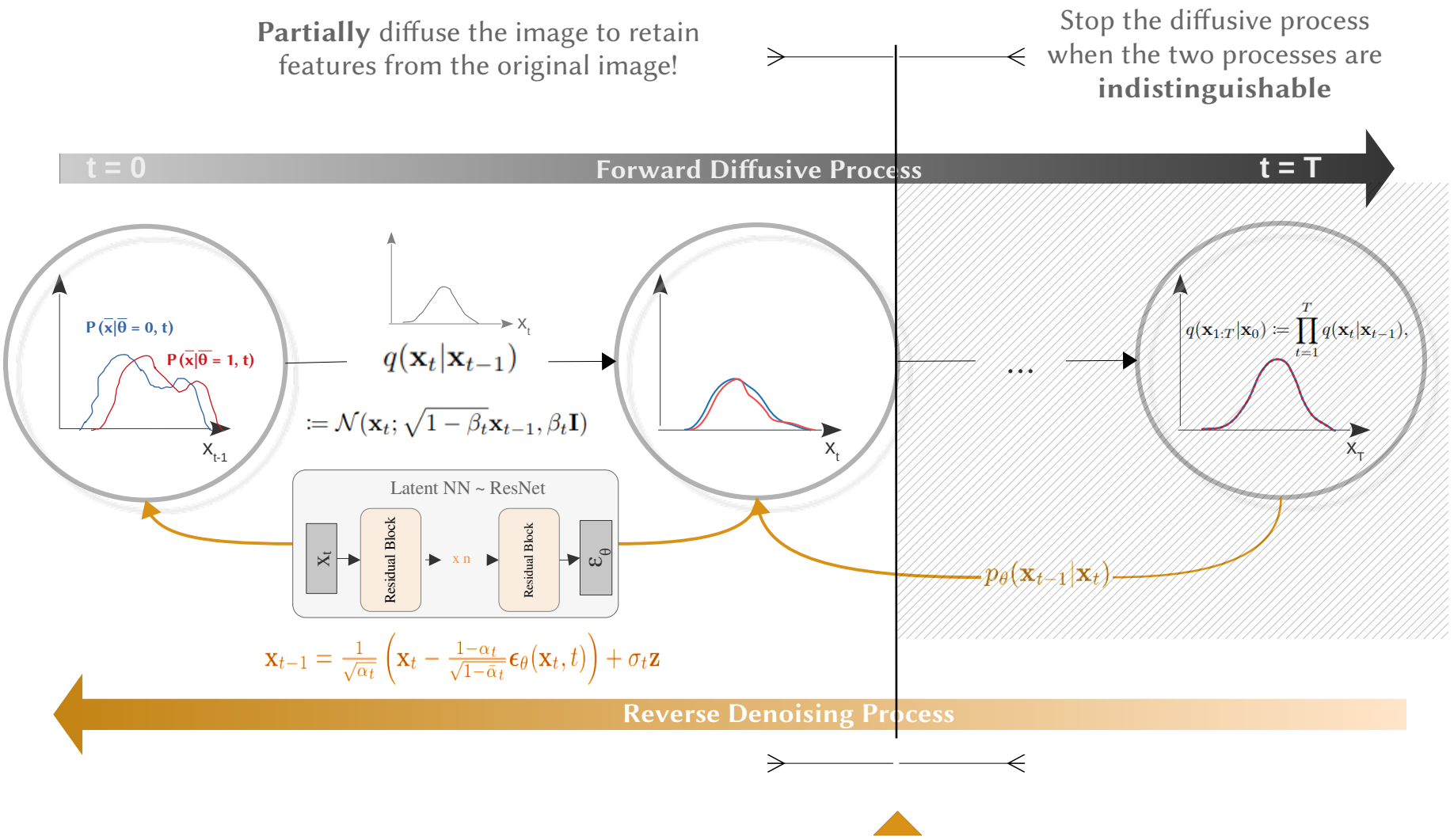
Global Self-Attention

Gated Cross-Attention

Residual Block



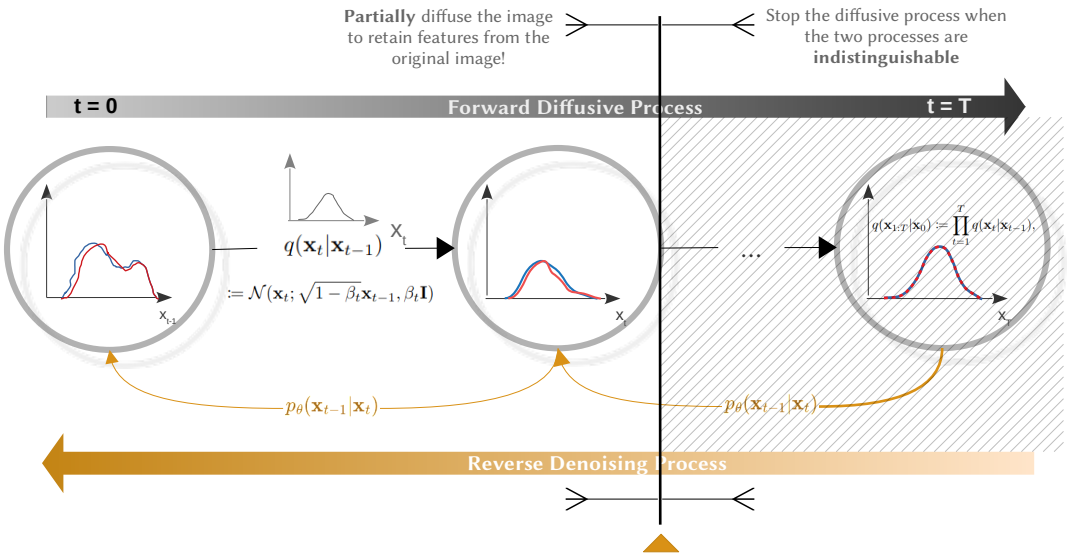
Partial Diffusion Probabilistic Model



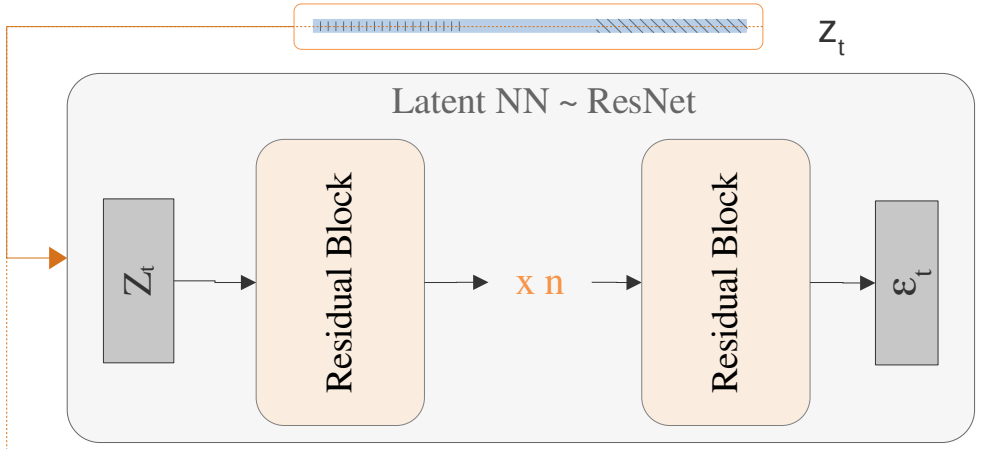
Partially Diffused DDPM Generation

Train a conditional DDPM and partially diffuse a sample to a *release time* $\mathbf{t}_r$. Then switch class label (0 ⟶ 1) and denoise.

Partially Diffused DDPM Operation



→ How do we determine the release time t_r ?



$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)}$$

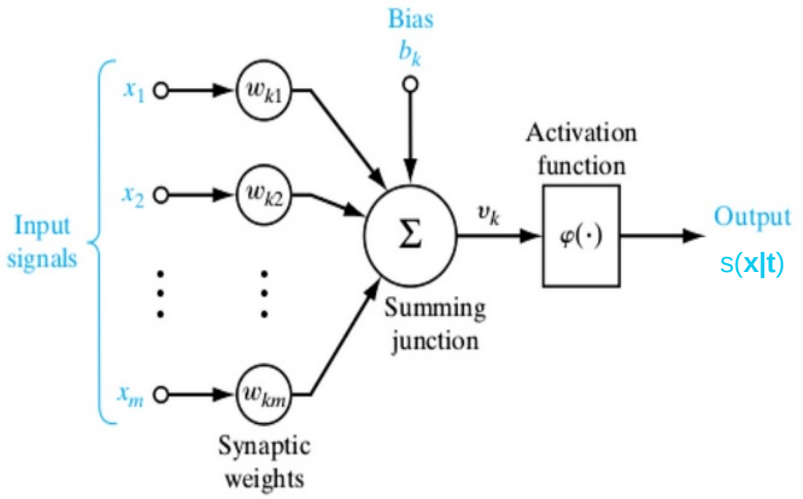
Neural Likelihood Ratio Estimation

For a neural network with configurable set $\Phi = \{w_i\}^N_{i=1}$ parameters and a given loss functional $\mathcal{L}(s)$ of the form:

$$\mathcal{L}(s) = -\int_{\mathbb{R}^n} d^n x \, (p(x|t,\theta_0) \cdot \ln(s(x|t)) + p(x|t,\theta_1) \cdot \ln(1 - s(x|t)))$$

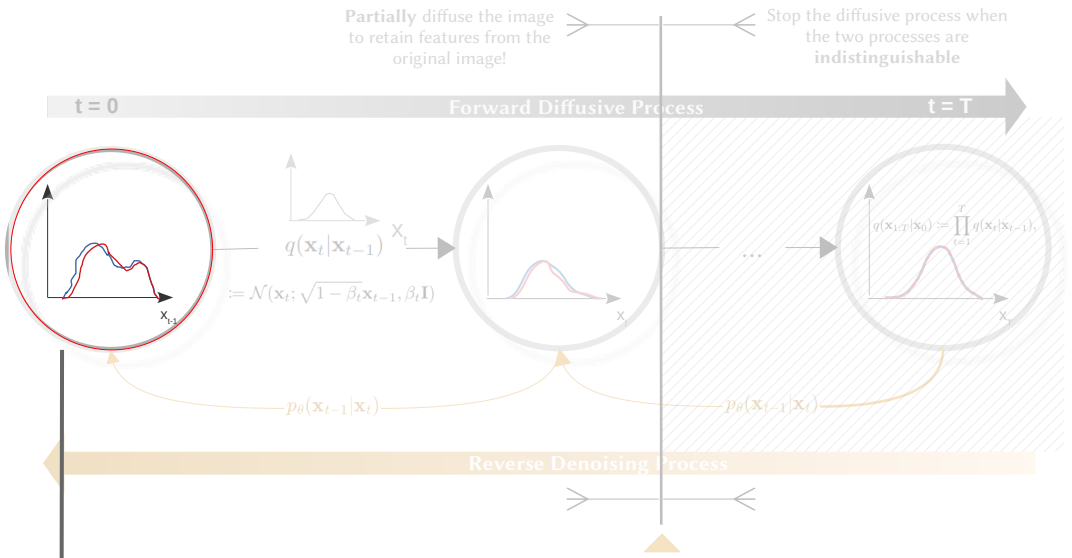
The extrema ($\delta\mathcal{L}(sh)/\delta s = 0$) of this general loss yields:

$$\frac{1 - s(x|t)}{s(x|t)} = \frac{p(x|t,\theta_1)}{p(x|t,\theta_0)} = r(x|t,\theta_0,\theta_1)$$



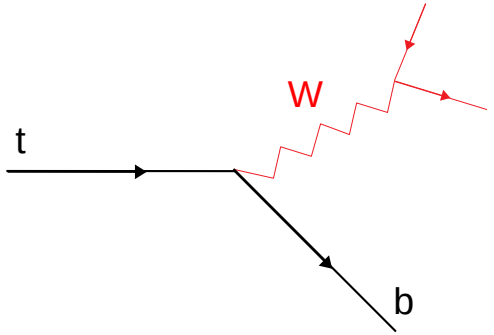
04 Partial Diffusion Alg. Prototype

Prototype



→ **Physics Example:** *Top -vs- W-boson Particles* [1]

→ For this example a *top-quark* decays into a *W-boson*



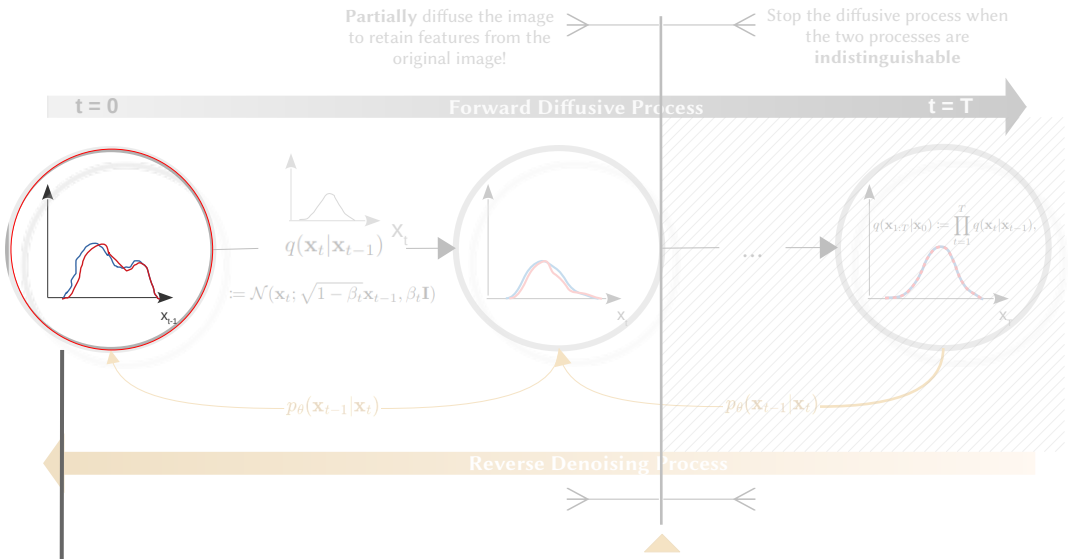
→ In particle physics we identify these particles by their mass:

Top Mass (m_t) ~ 175 GeV

W-Boson Mass (m_W) ~ 81 GeV

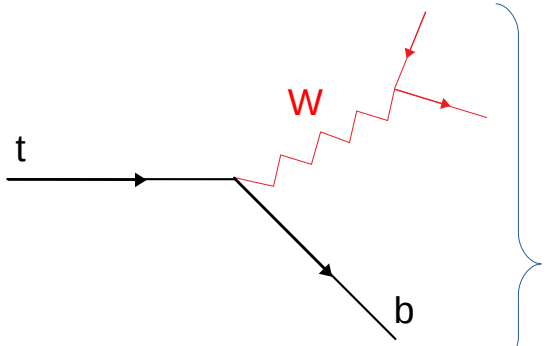
[1] JetNet: A Python package for accessing open datasets and benchmarking machine learning methods in high energy physics, https://joss.theoj.org/papers/10.21105/joss.05789

Prototype



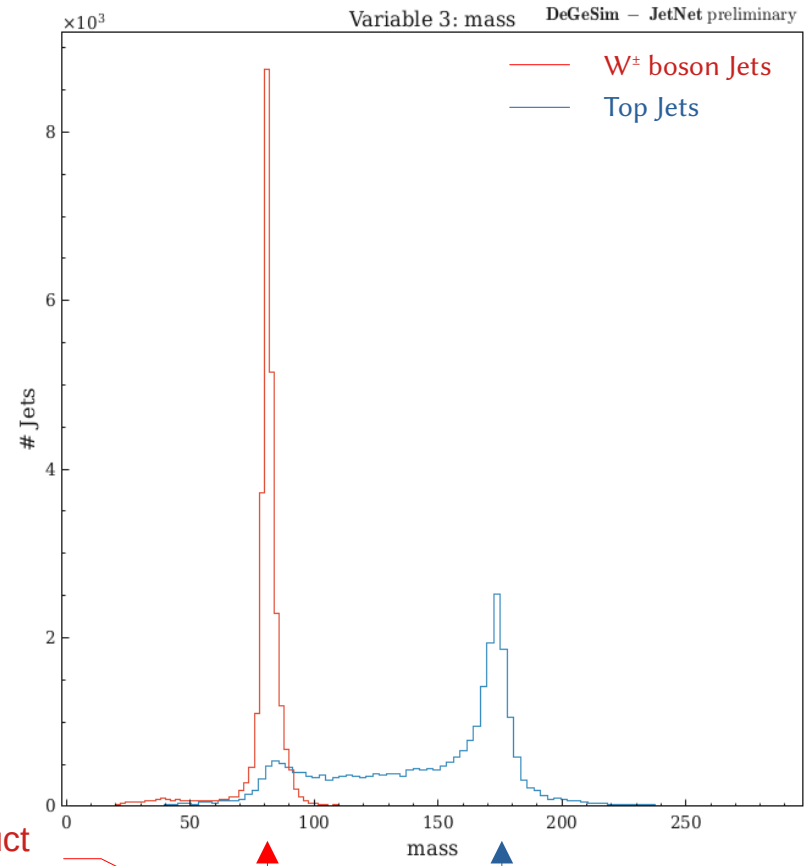
→ **Physics Example:** *Top -vs- W-boson Particles* [1]

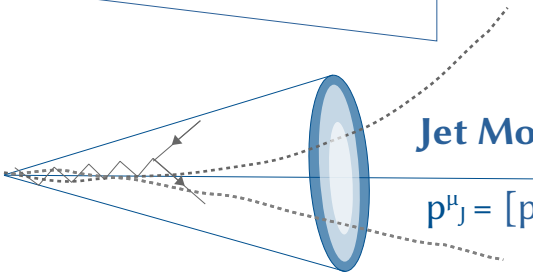
→ For this example a *top-quark* decays into a *W-boson*



If you reconstruct only this

If you reconstruct both: W+b = t





Jet Momentum Vector

$p^{\mu}_{J} = [p_T, \eta, \text{mass}, \phi]$

Jet of Particles

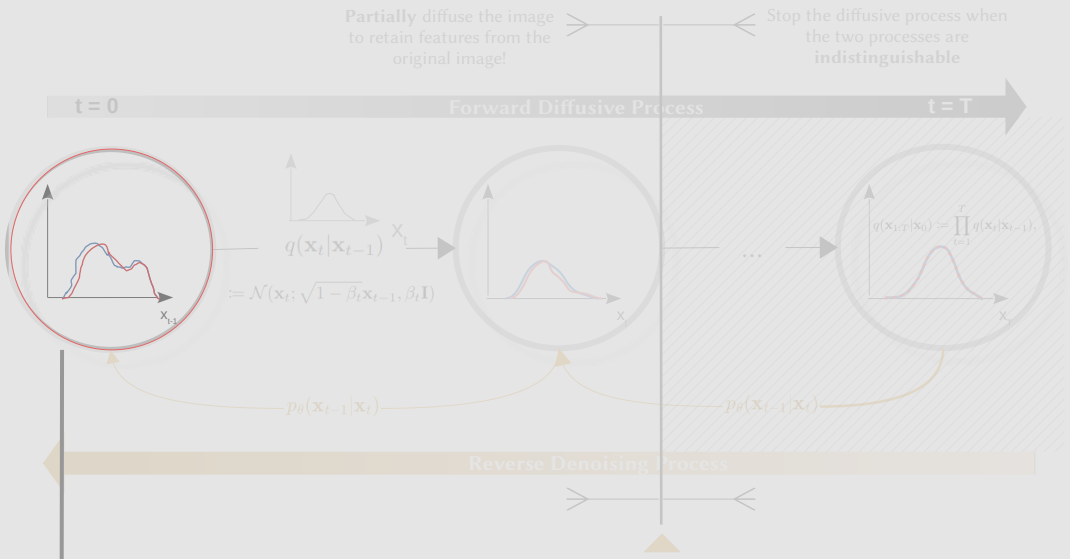
→ In particle physics we identify these particles by their mass:

Top Mass (m_t) ~ 175 GeV

W-Boson Mass (m_W) ~ 81 GeV

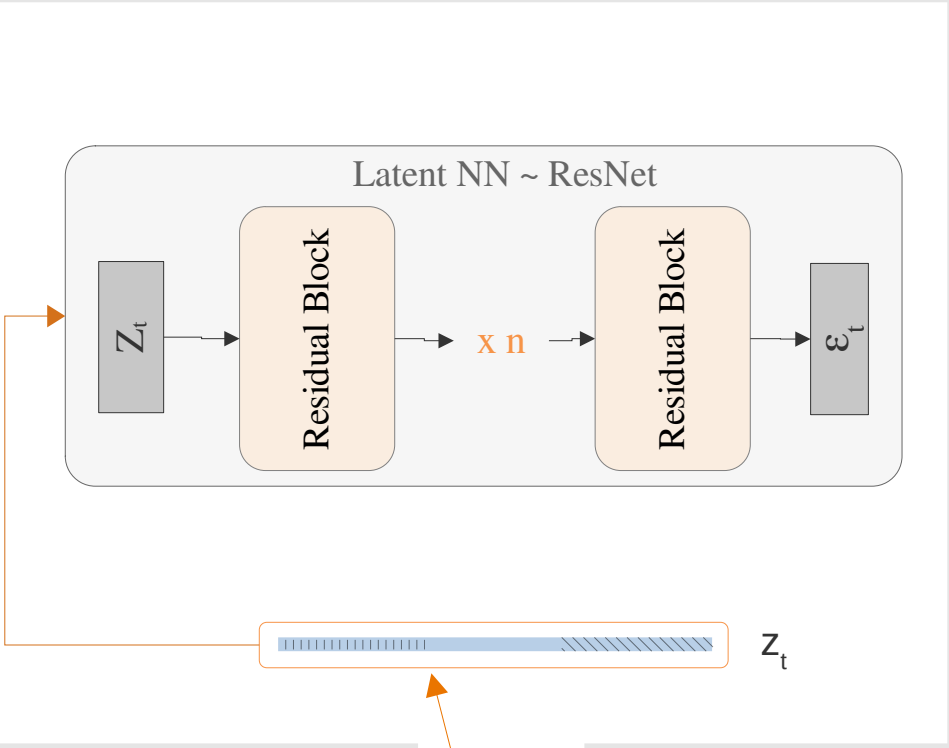
[1] JetNet: A Python package for accessing open datasets and benchmarking machine learning methods in high energy physics, https://joss.theoj.org/papers/10.21105/joss.05789

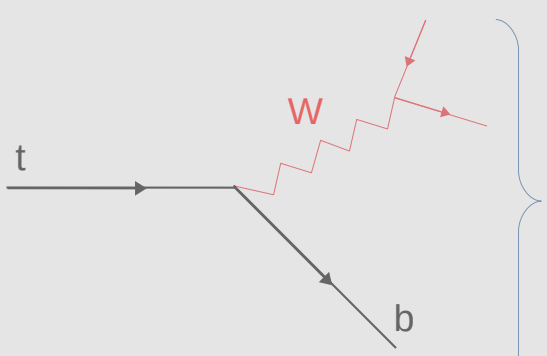
Prototype



→ **Physics Example:** *Top -vs- W-boson Particles*

→ For this example a *top-quark* decays into a *W-boson*





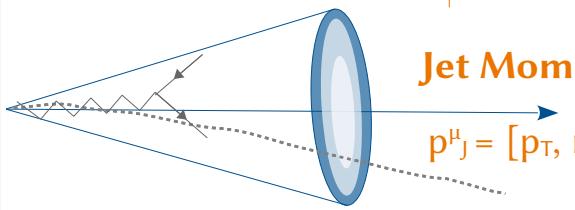
If you reconstruct only this

If you reconstruct both: W+b = t

→ In particle physics we identify these particles by their mass:

Top Mass (m_t) ~ 175 GeV

W-Boson Mass (m_W) ~ 81 GeV



Jet Momentum Vector

$p^{\mu}_{J} = [p_T, \eta, \text{mass}, \phi]$

Jet of Particles

JetNet: Top -vs- W jets

Key

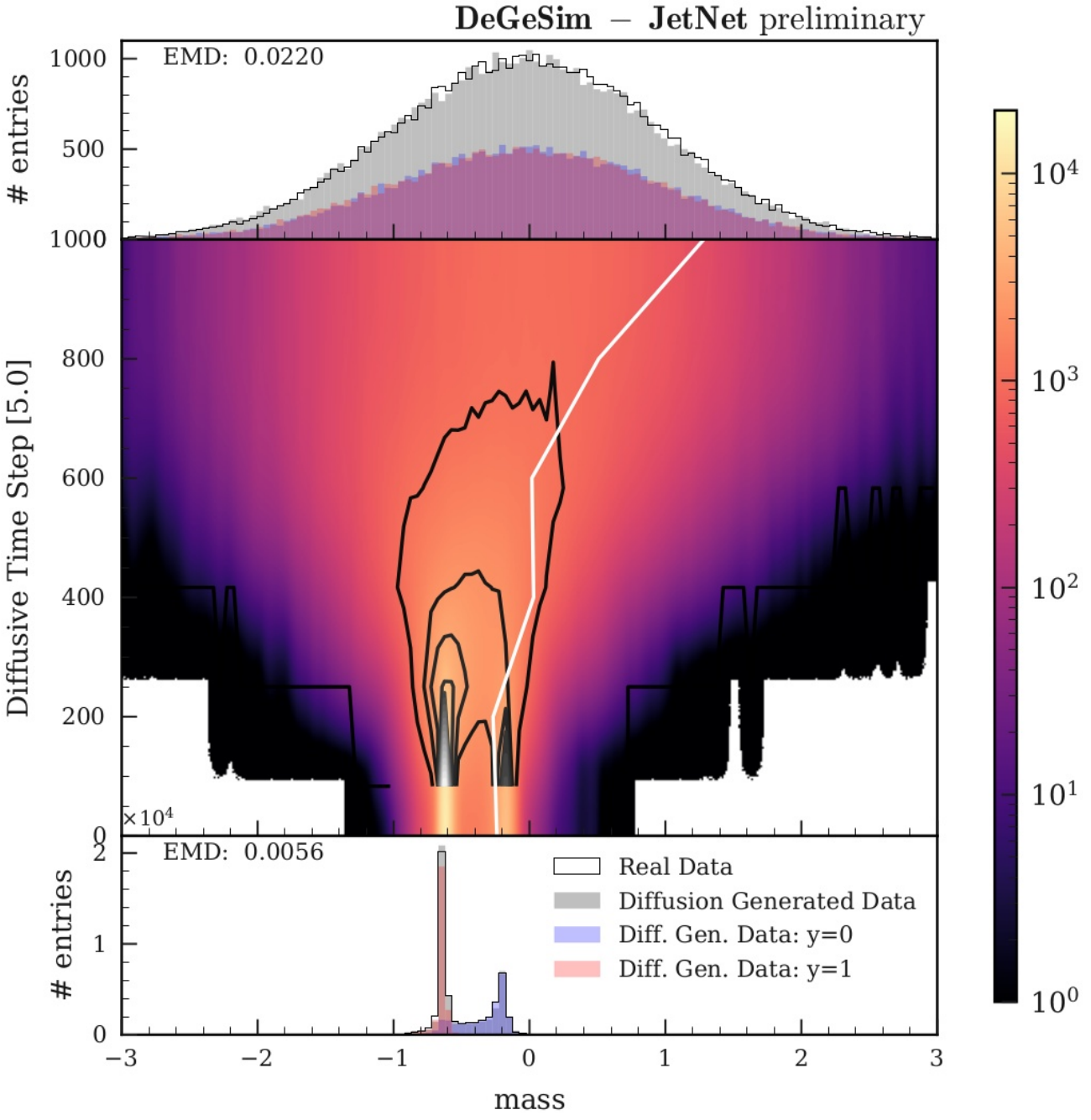
→ **Top Pane:** Gaussian generated (Gen.) sampling with forward diffused *real data*

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

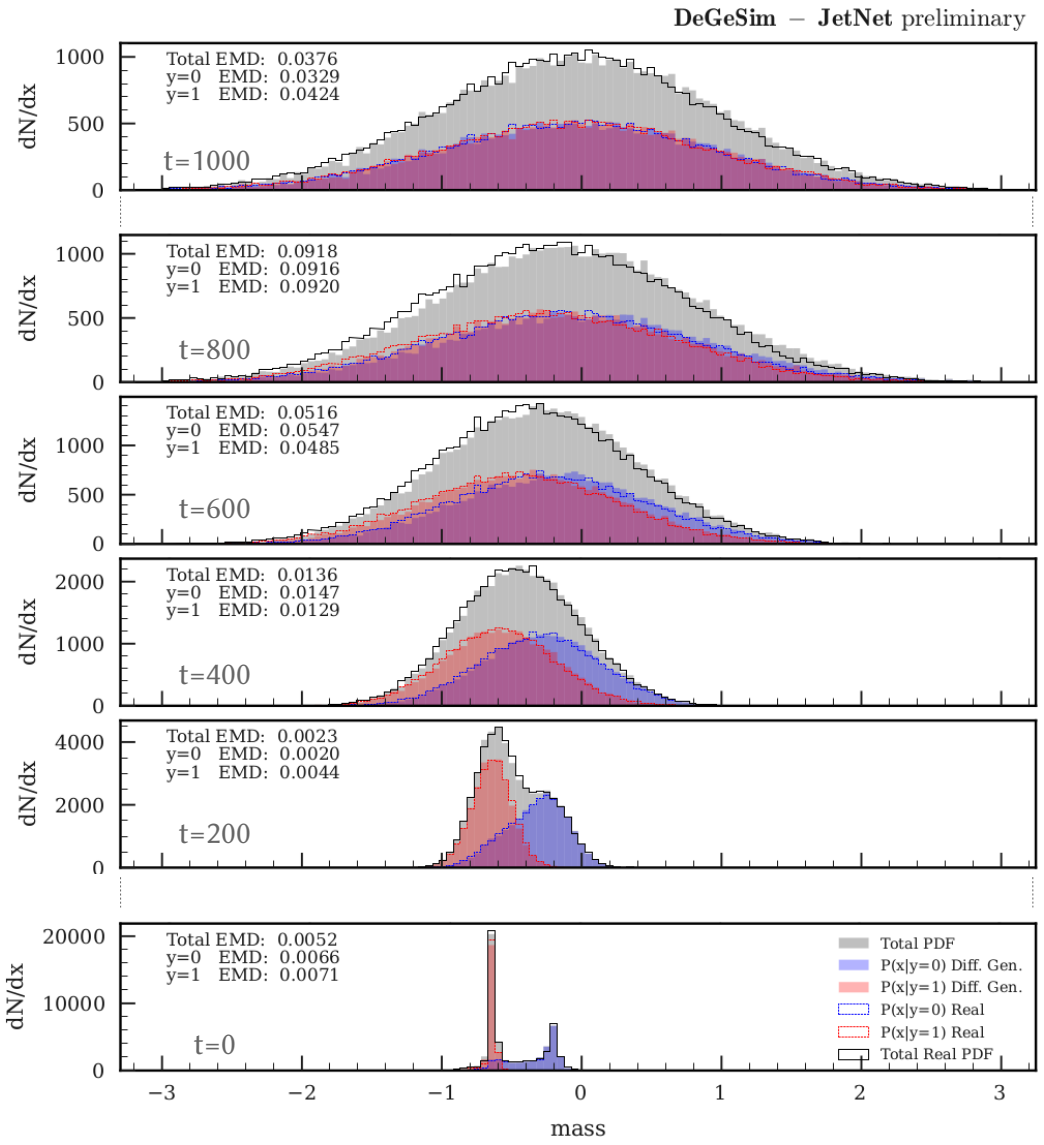
Black contours = equal probability
White line = random event path

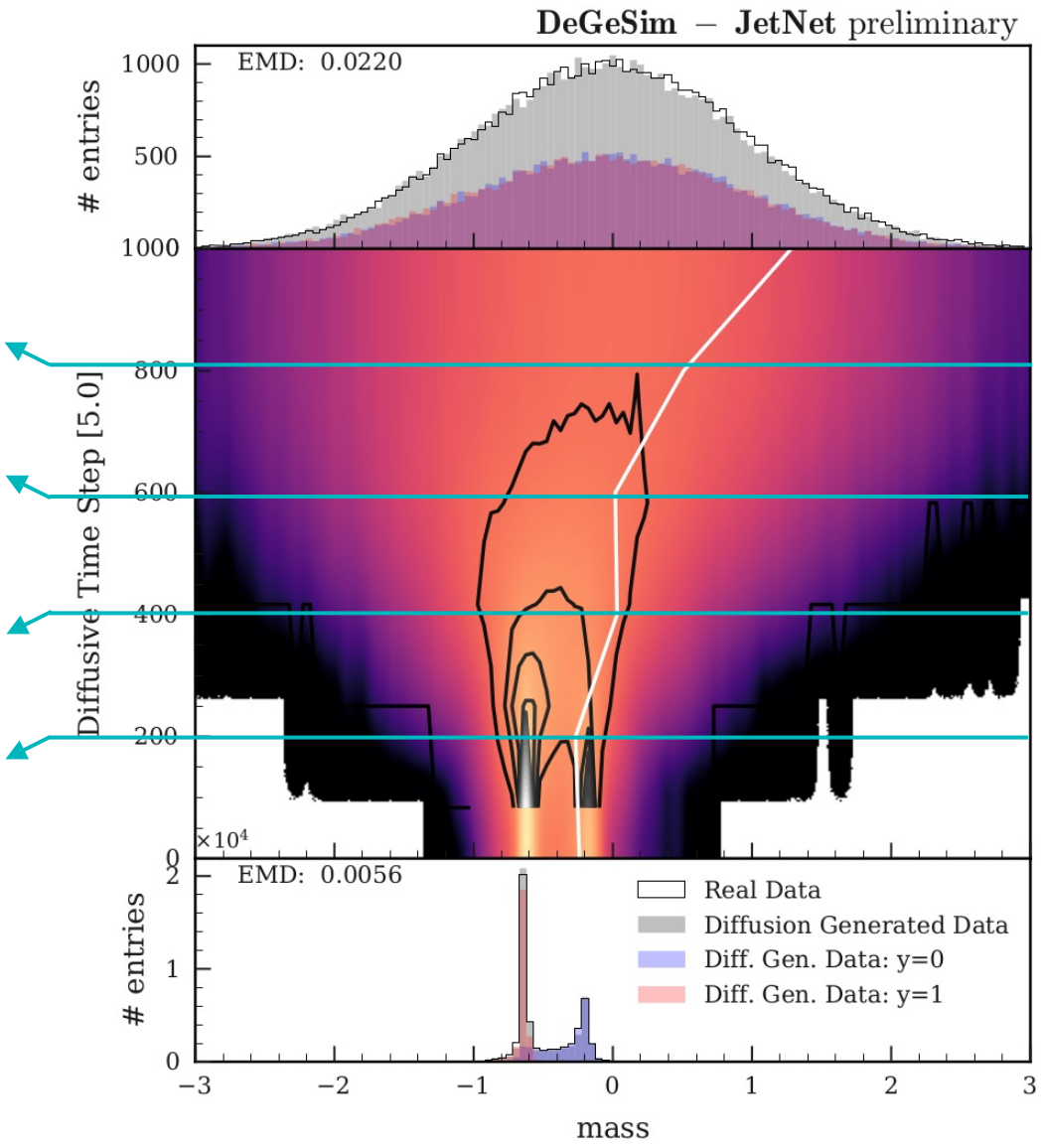
→ **Bottom Pane:** Real data vs diffusion generated data from a guassian prior (top pane).

→ **Metrics:** Wasserstein distance in 1D projection between total real and total generated data

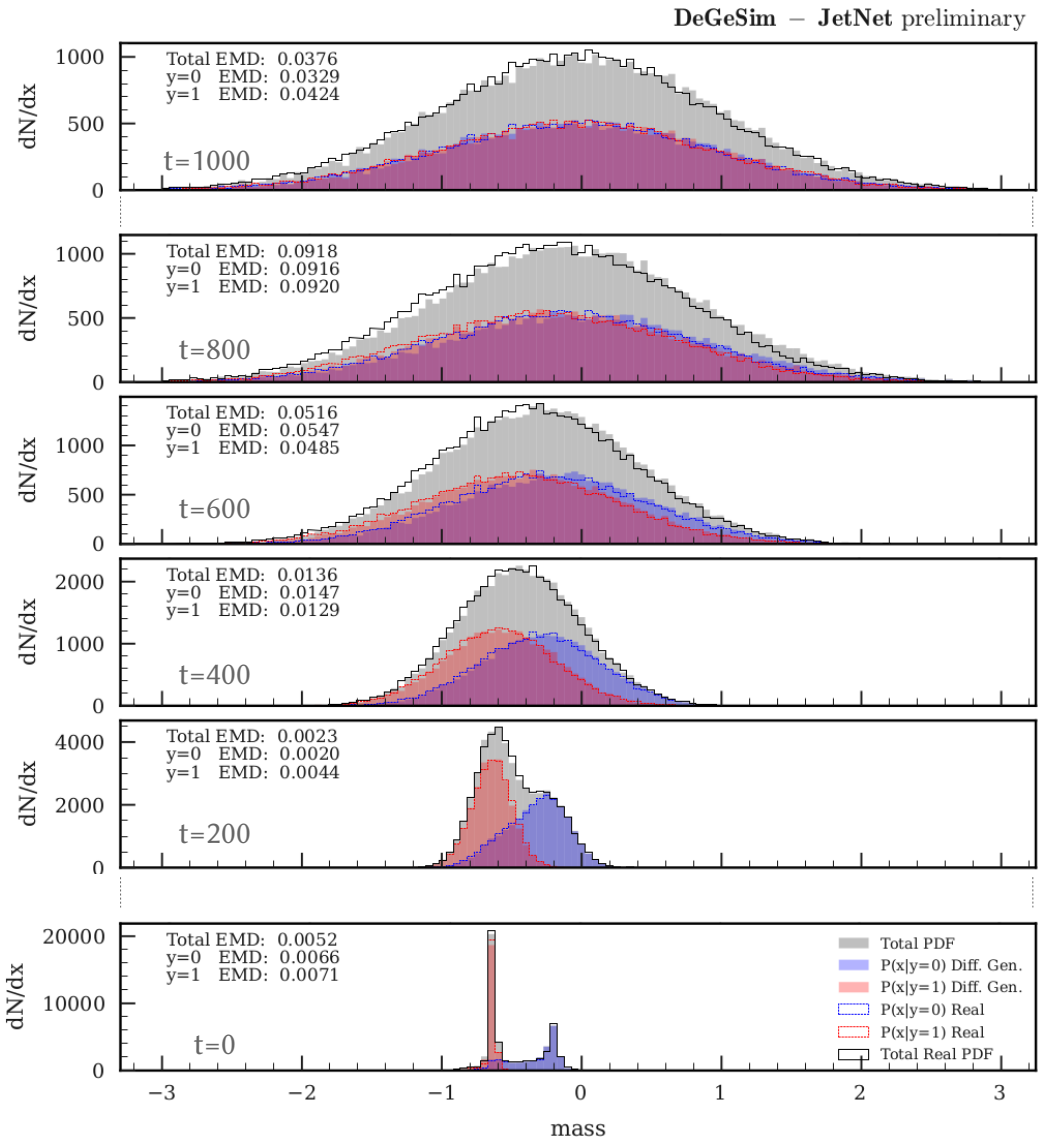


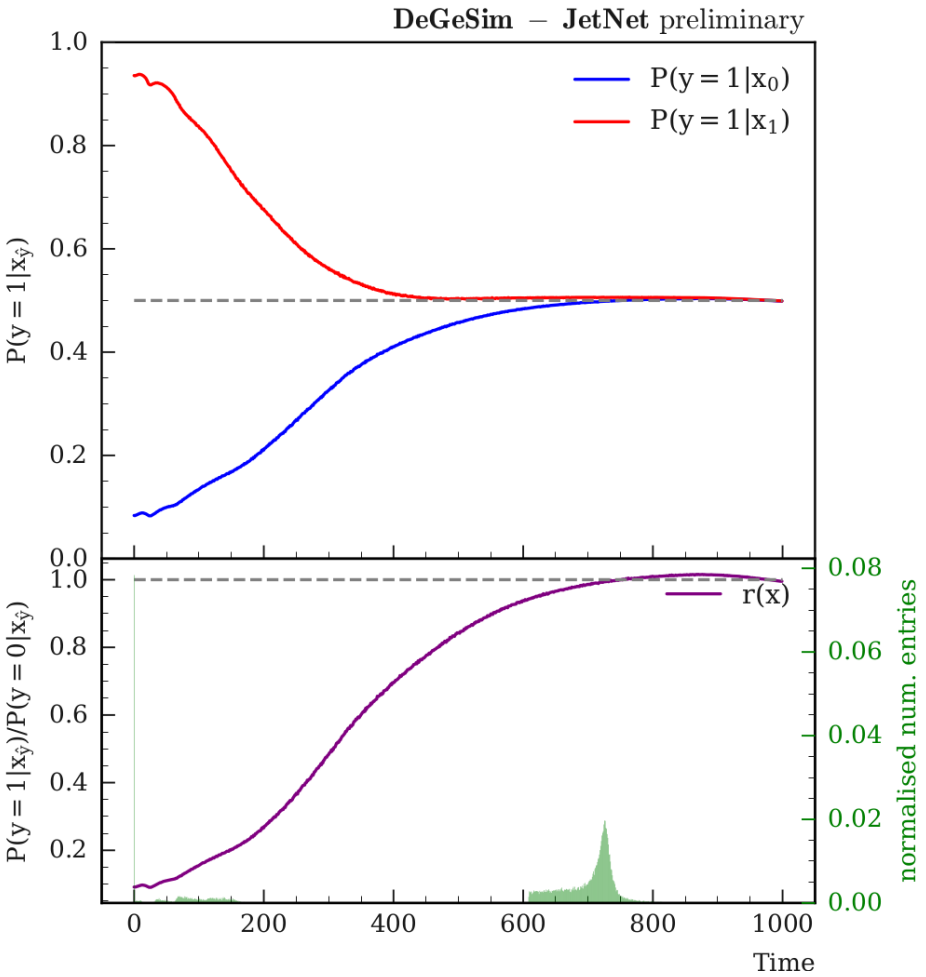
JetNet: Top -vs- W jets





JetNet: Top -vs- W jets

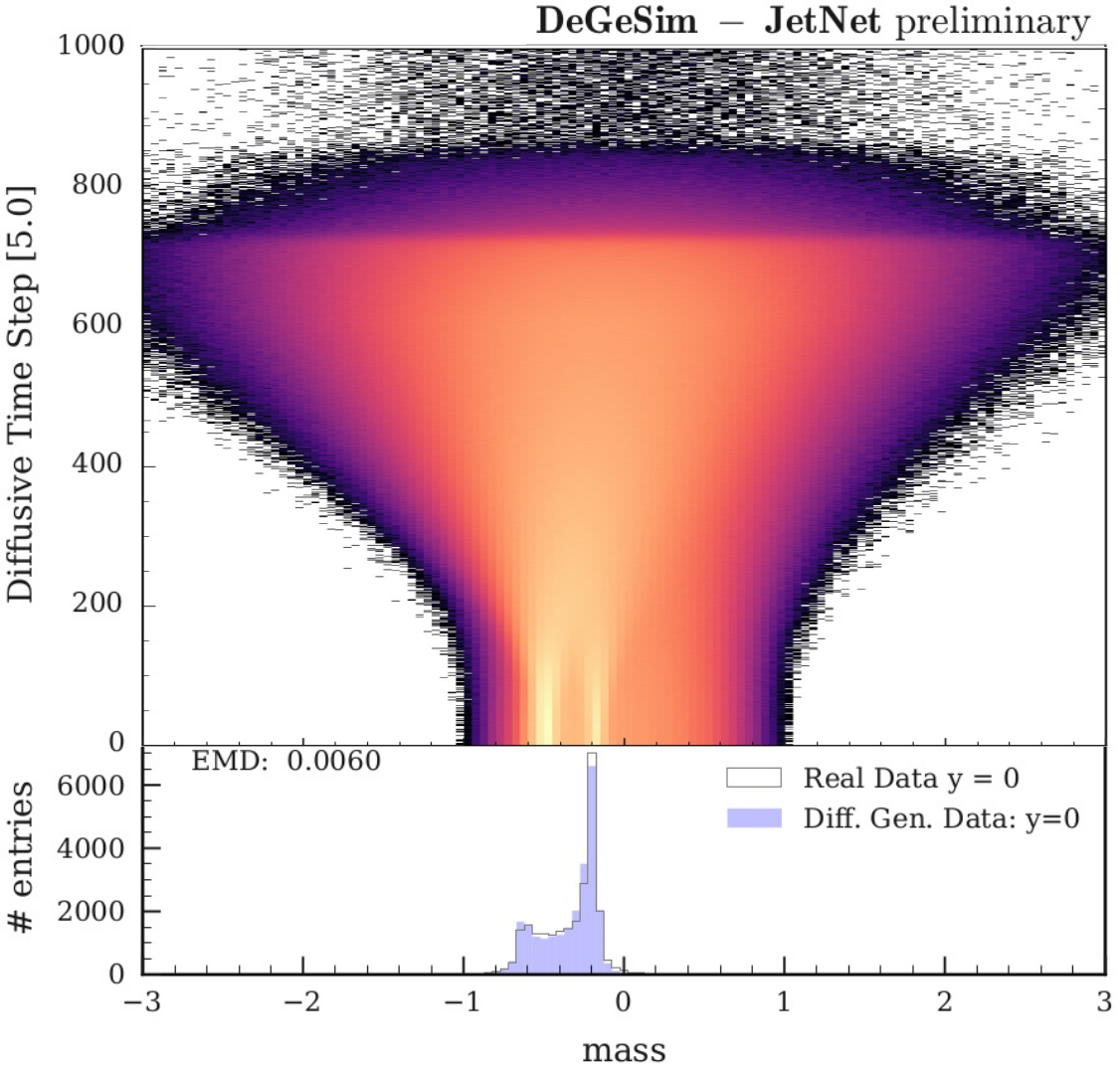


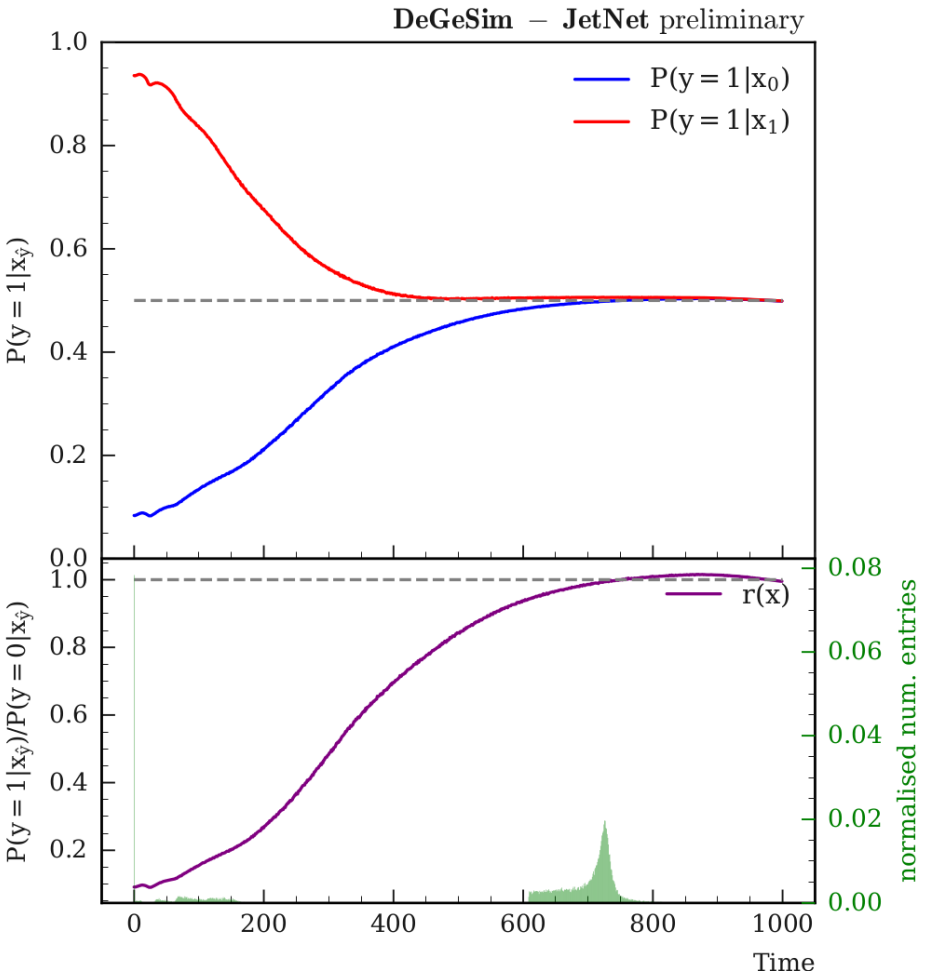


→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$

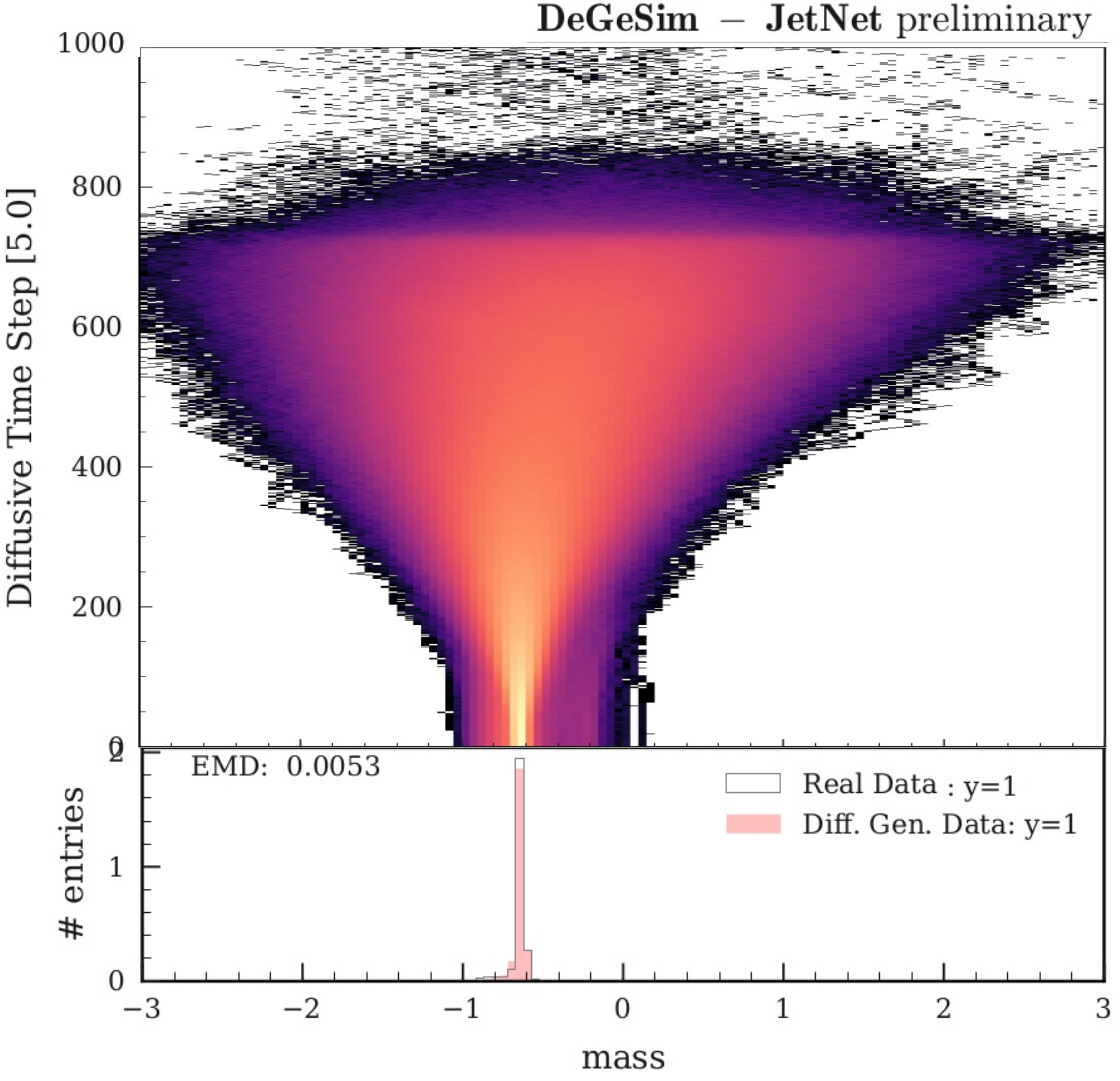
JetNet: Top -vs- W jets

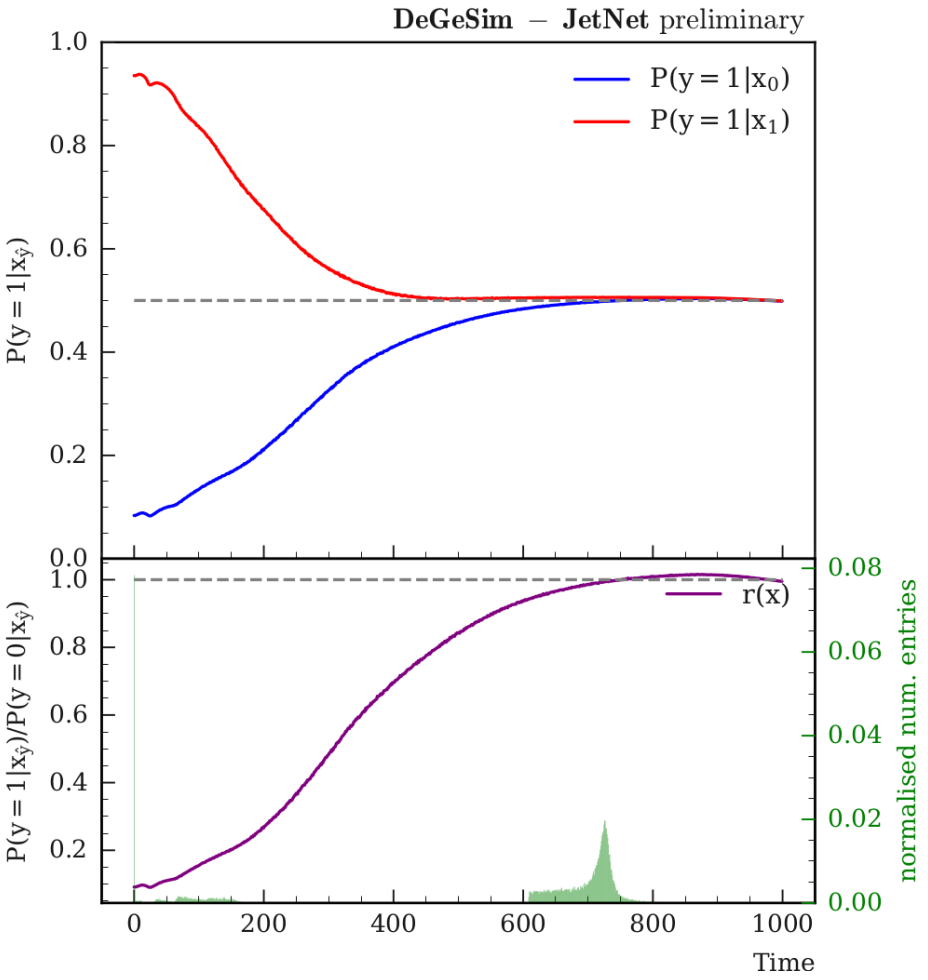




→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$

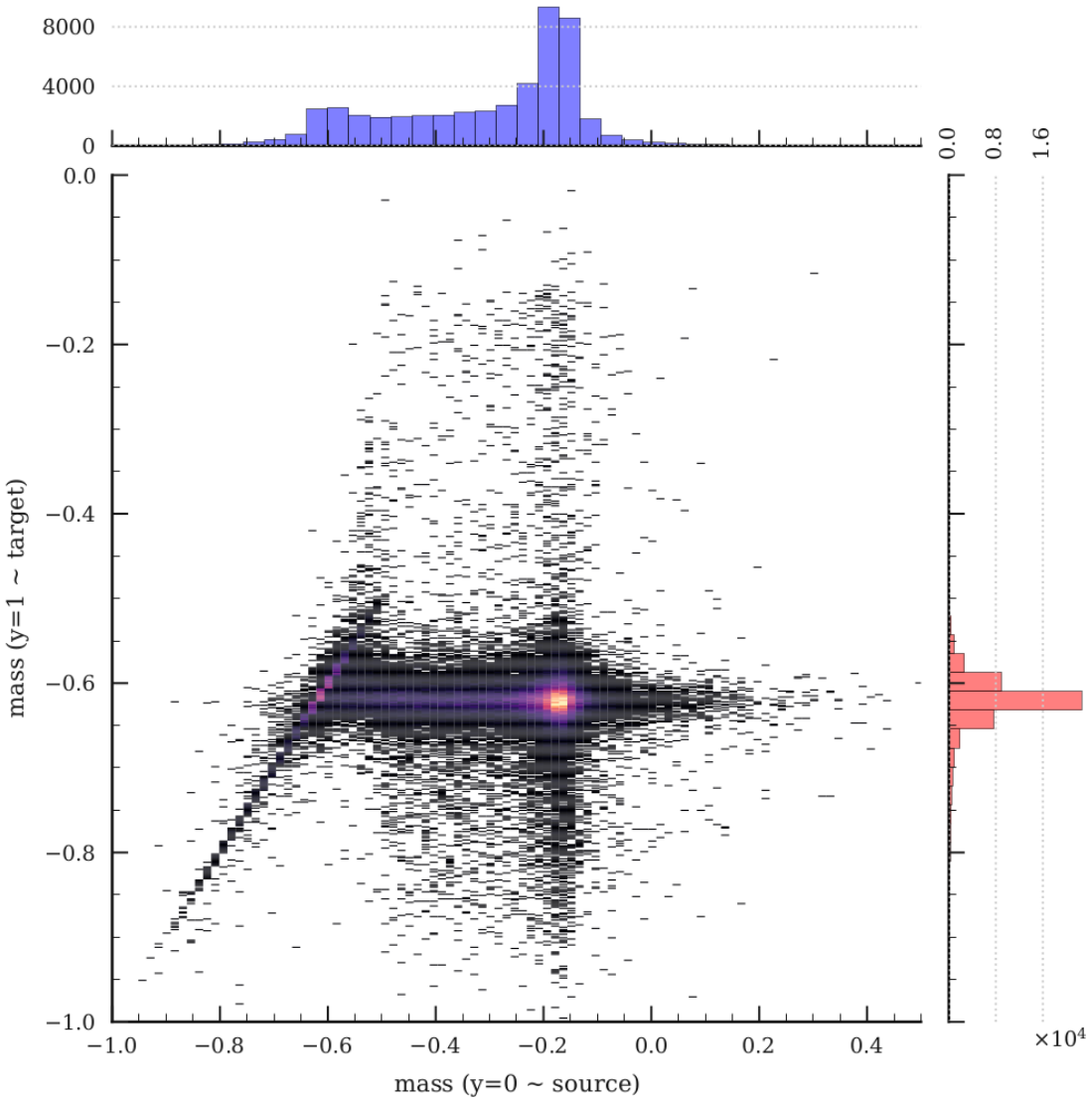


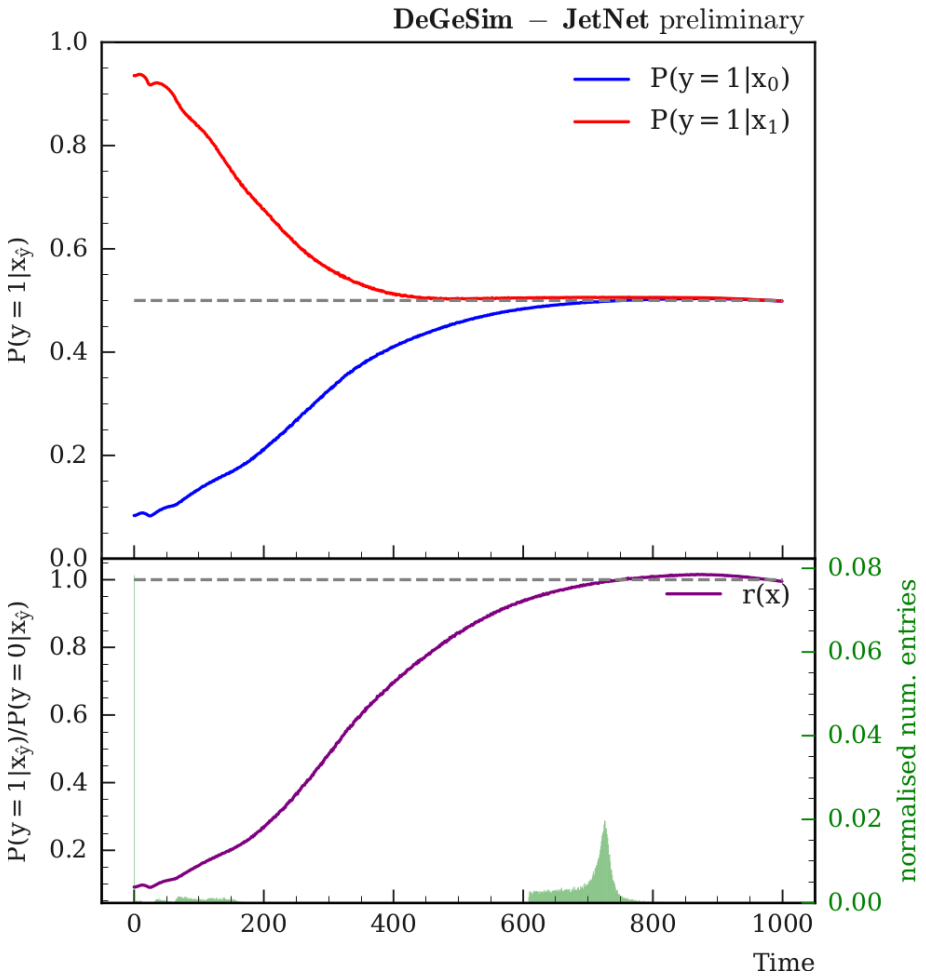


→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$

JetNet: Top -vs- W jets

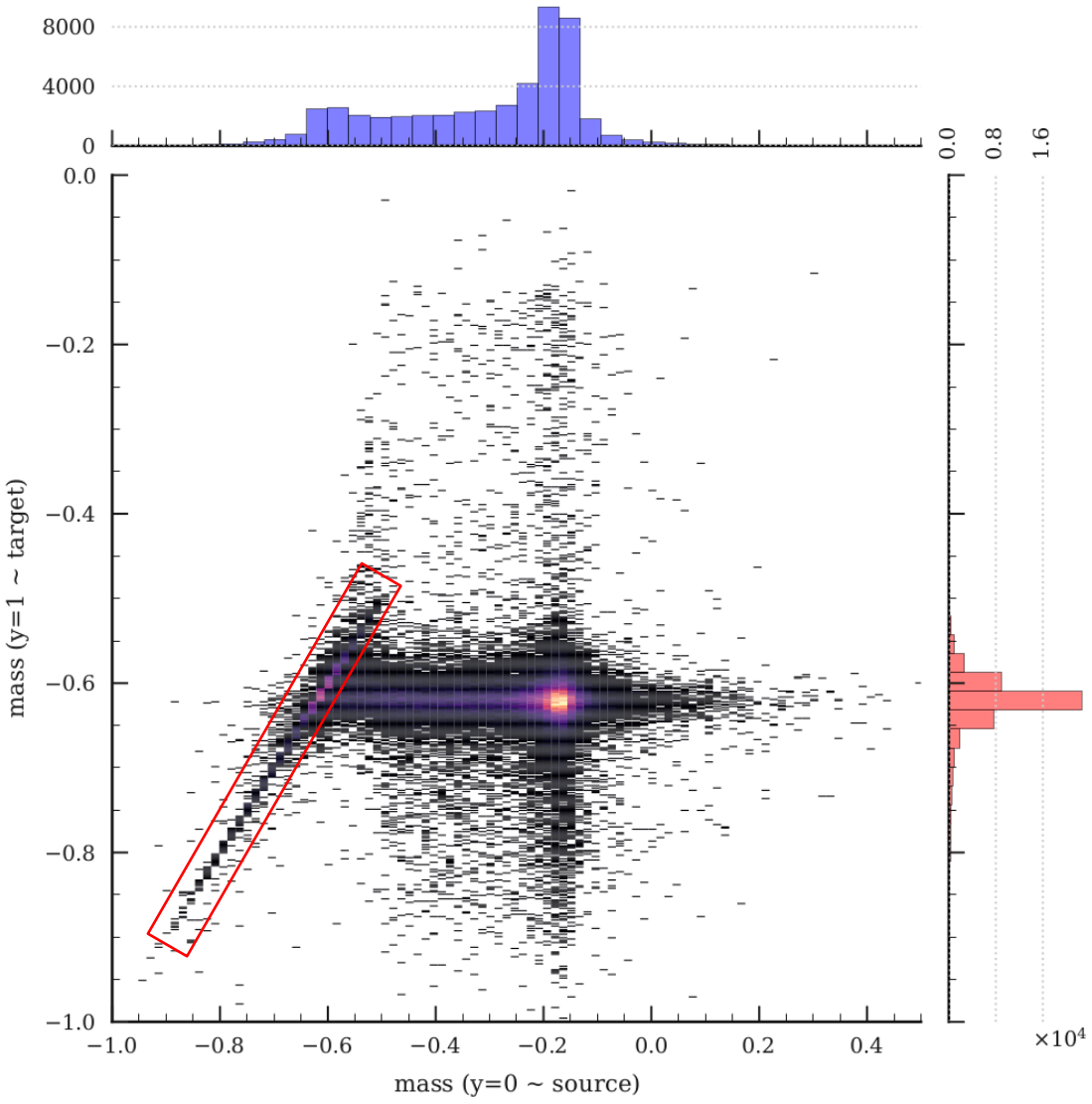


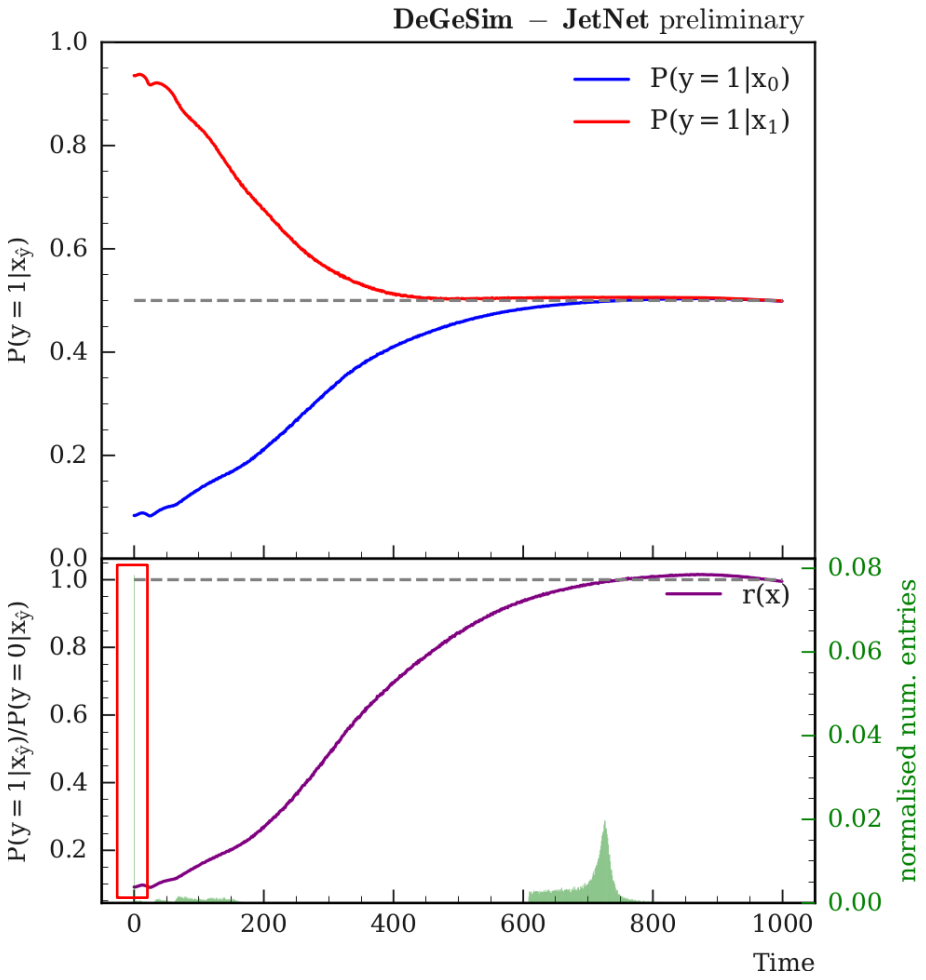


→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$

JetNet: Top -vs- W jets

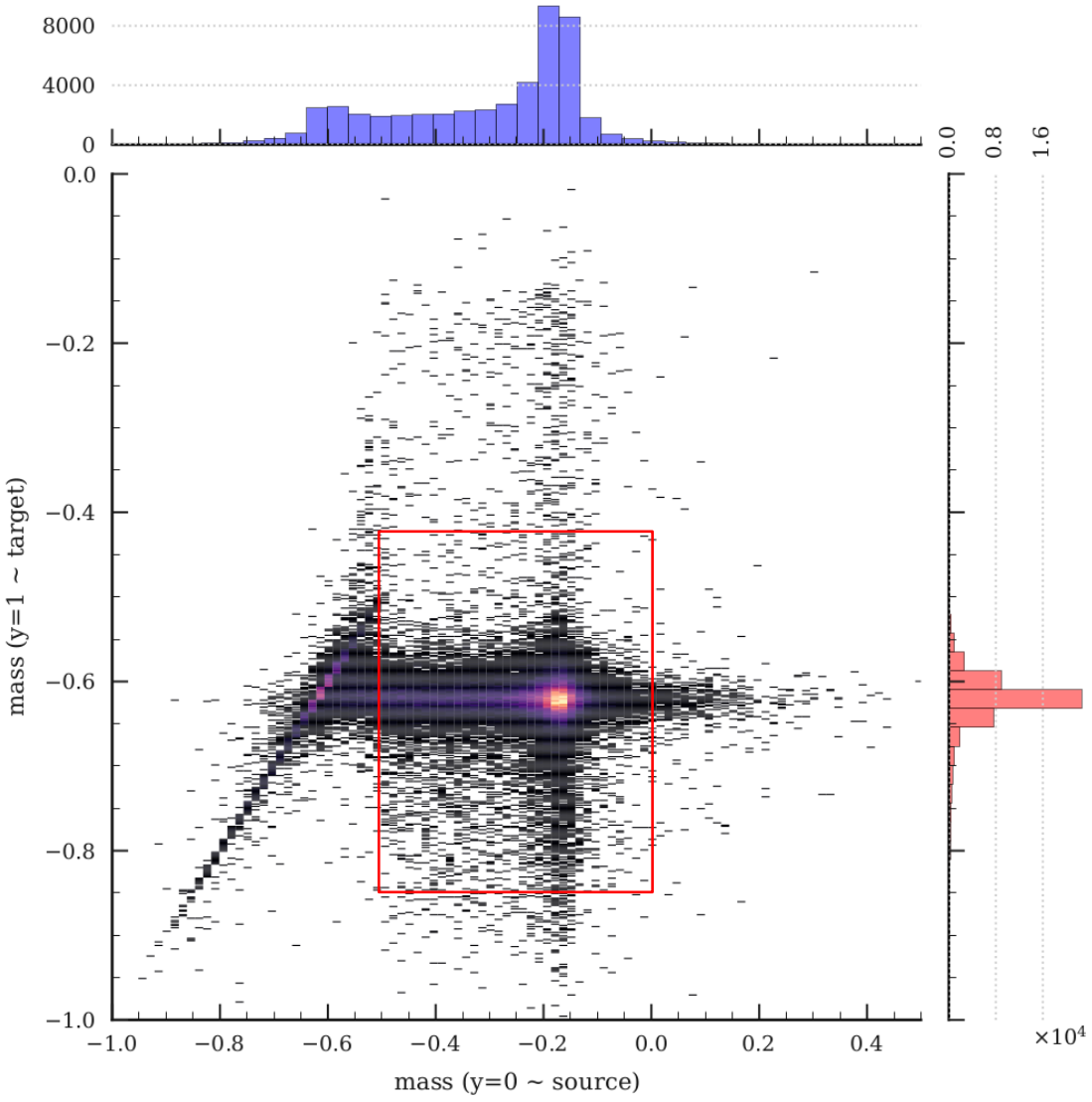


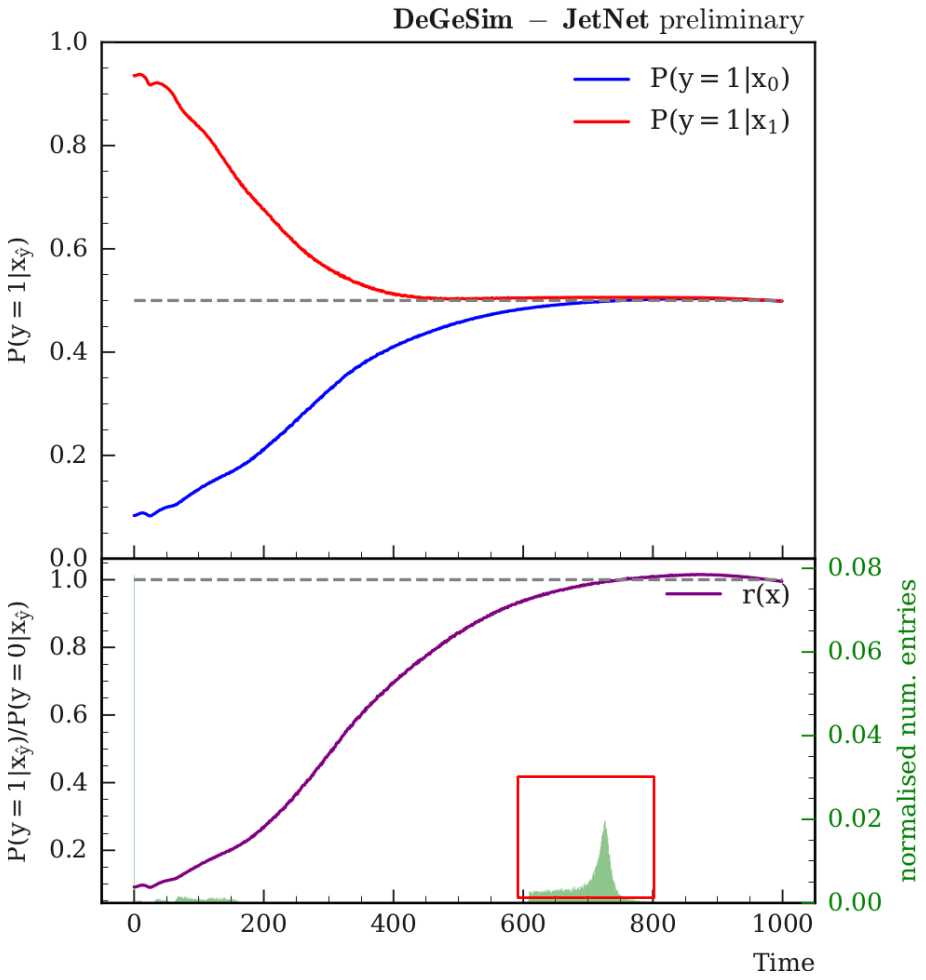


→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$

JetNet: Top -vs- W jets

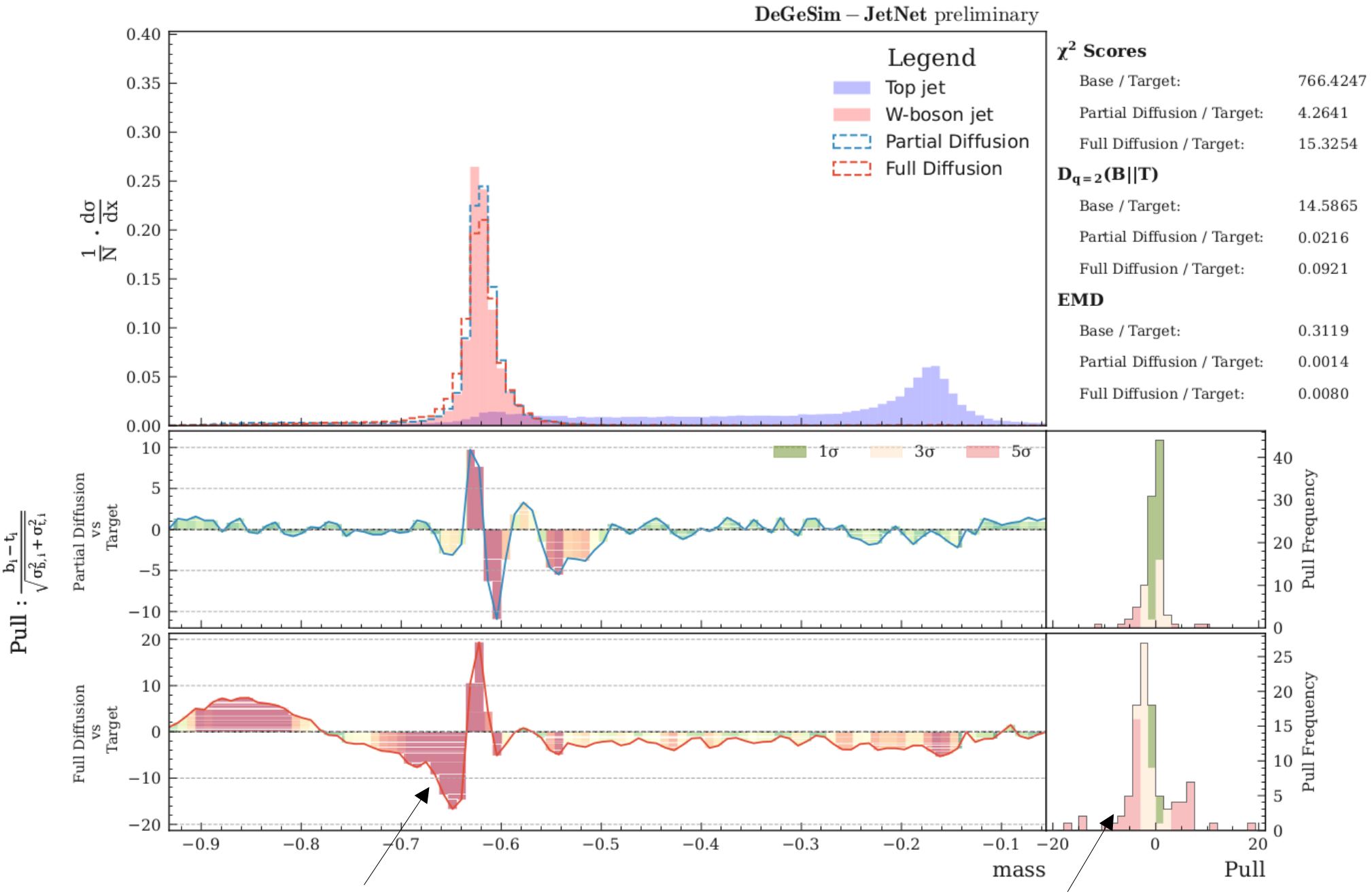




→ When the densities are equivalent we define the *release time* of each event/data point

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$

JetNet: Top -vs- W jets

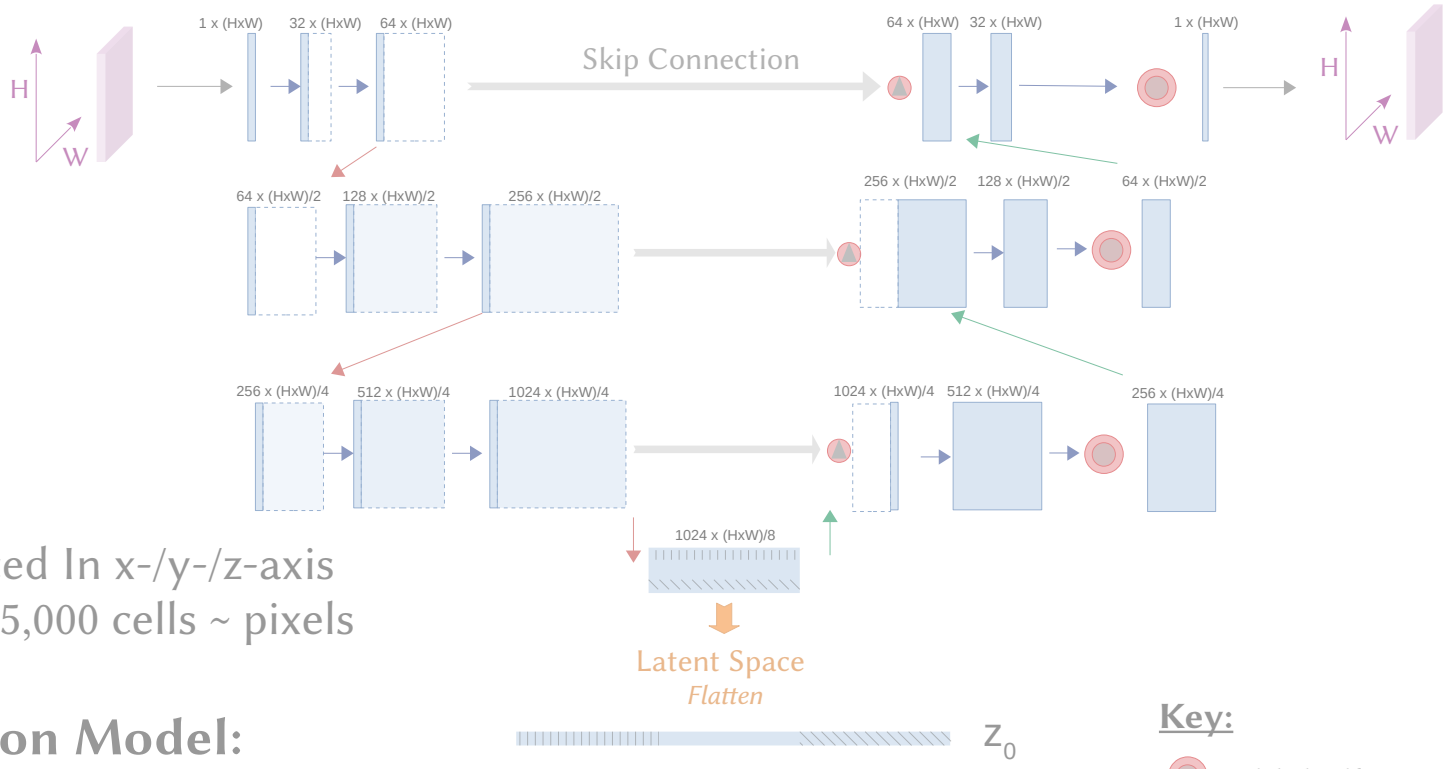


Per bin pull between the reference and the target

Distribution of pulls across all bins.
Ideally want it to be Gaussian

05 Conclusions

Calorimeter Image Model



→ **Image Structure:**
- → 3D image displaced In x-/y-/z-axis
- → Approximately175,000 cells ~ pixels

→ **Image Compression Model:**
- → Encoder → Latent → Decoder
- → 2D planar CNN based model

→ **R&D Hurdles:**

(1) Image Sparsity:

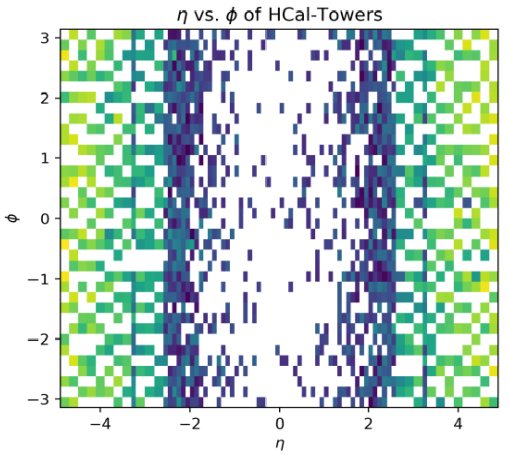
Pile-up calorimeter response is noisy in ϕ, but structured in η

Key:

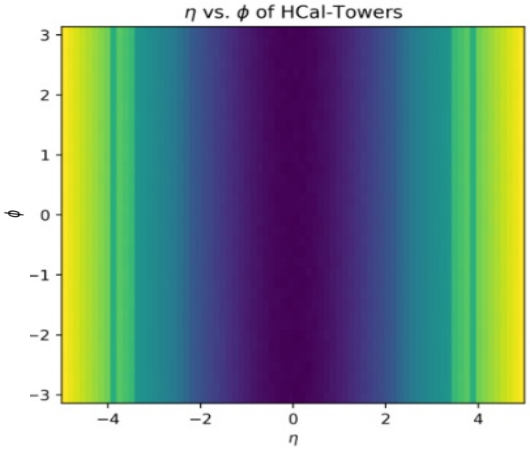
Global Self-Attention

Gated Cross-Attention

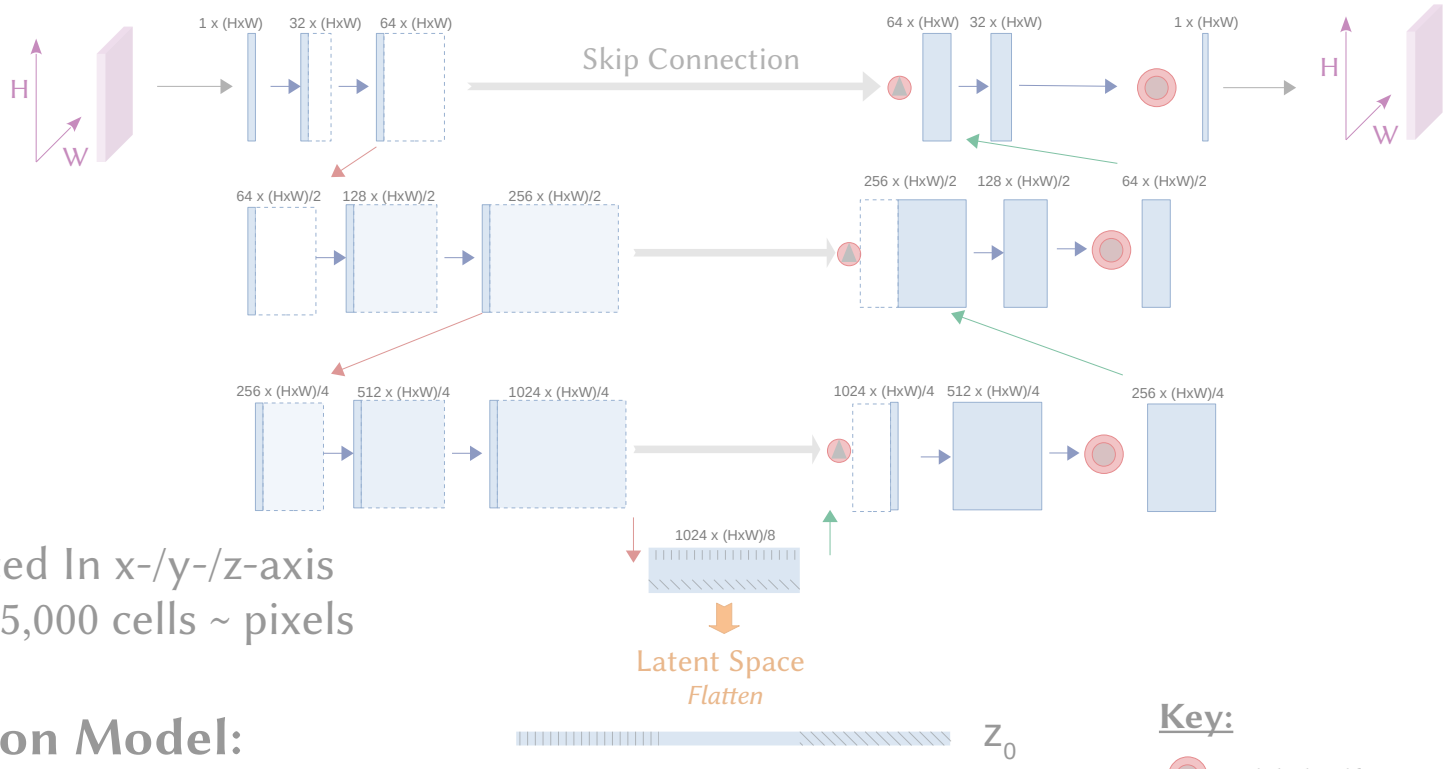
Real



Generated



Calorimeter Image Model



→ **Image Structure:**
- → 3D image displaced In x-/y-/z-axis
- → Approximately175,000 cells ~ pixels

→ **Image Compression Model:**
- → Encoder → Latent → Decoder
- → 2D planar CNN based model

→ **R&D Hurdles:**

(2) Data Volume & Ambiguity:

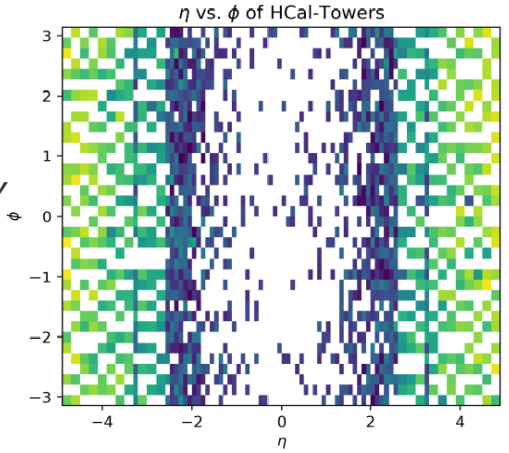
3D point cloud has better data compression. But yields an abiguity issue when mapping back to 2D representations

Key:

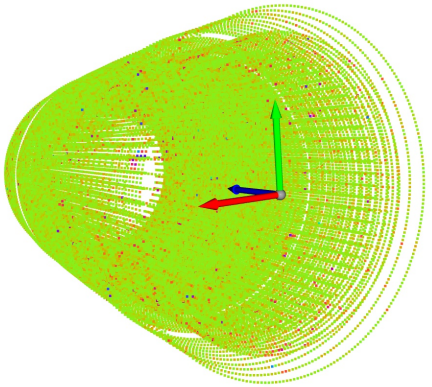
Global Self-Attention

Gated Cross-Attention

2D Planar



3D Point Cloud



Summary

→ **Goal:** Emulate calorimeter *images* of pileup 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 noise injection into calorimeter images:
  → CNNs struggle with noisy images and null noise injection
  → Point clouds introduce cell ↔ point assigment ambiguity

→ **Further work** will include:
  → Consistency models for faster inferencing
  → **Cross-domain** conditioning to enhance domain translation duing diffusion

Project Aim:

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

HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

ATLAS (DESY)


CMS (DESY)



Juelich

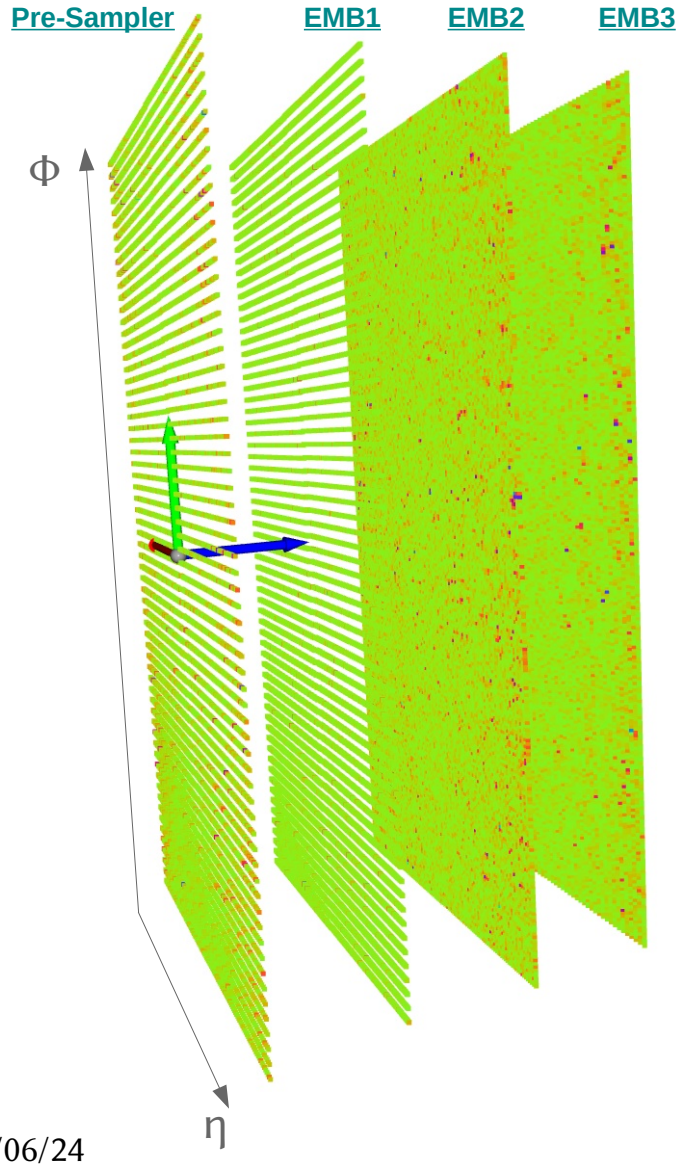
TRIUMF

Backup

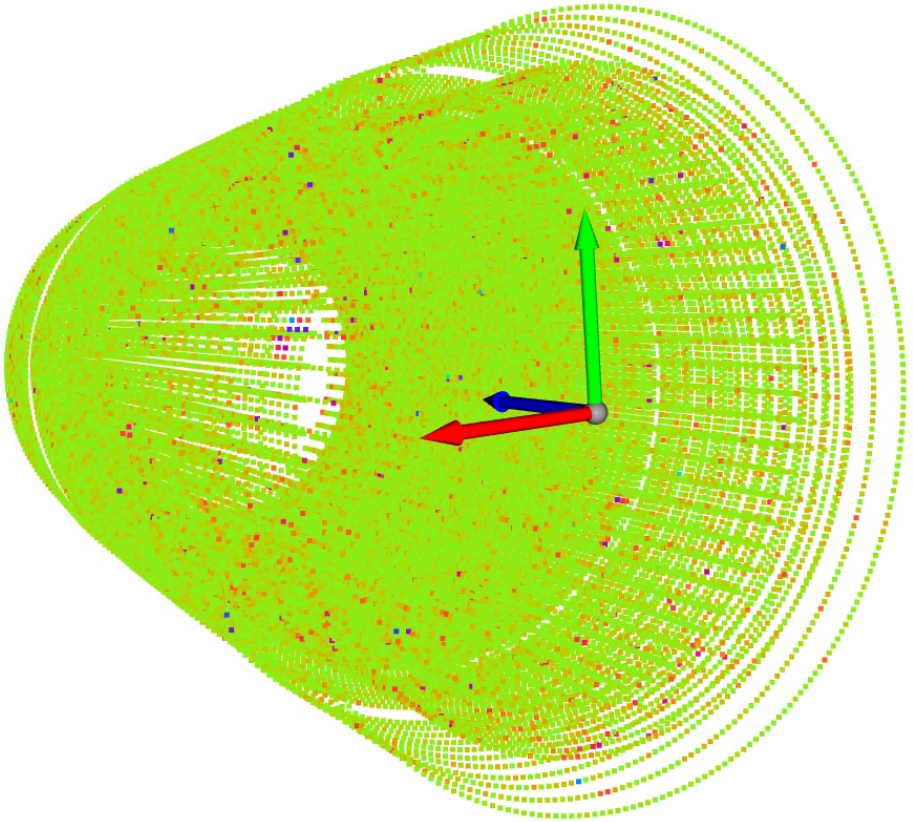
Detector Image – Point Cloud

→ **Point cloud** is actually the only way forward due to *simple geometry* used in the ATLAS simulation chain due to the irregular shape of the ECAL/HCAL

→ Artifacts induced due to projection will be problematic therefore simplest approach to solve the **data volume & geometry issues**



Move to euclidean space



Visualisation and point cloud creator available

→ Next steps is to create sparse PC:

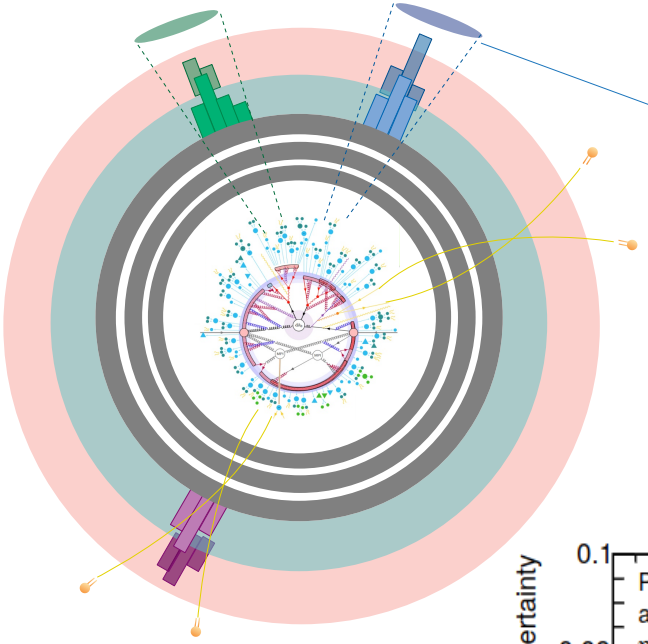
→ **Visualisation:** Open3d

→ **ML Graph:** Open3d ML

Brief

Problem:

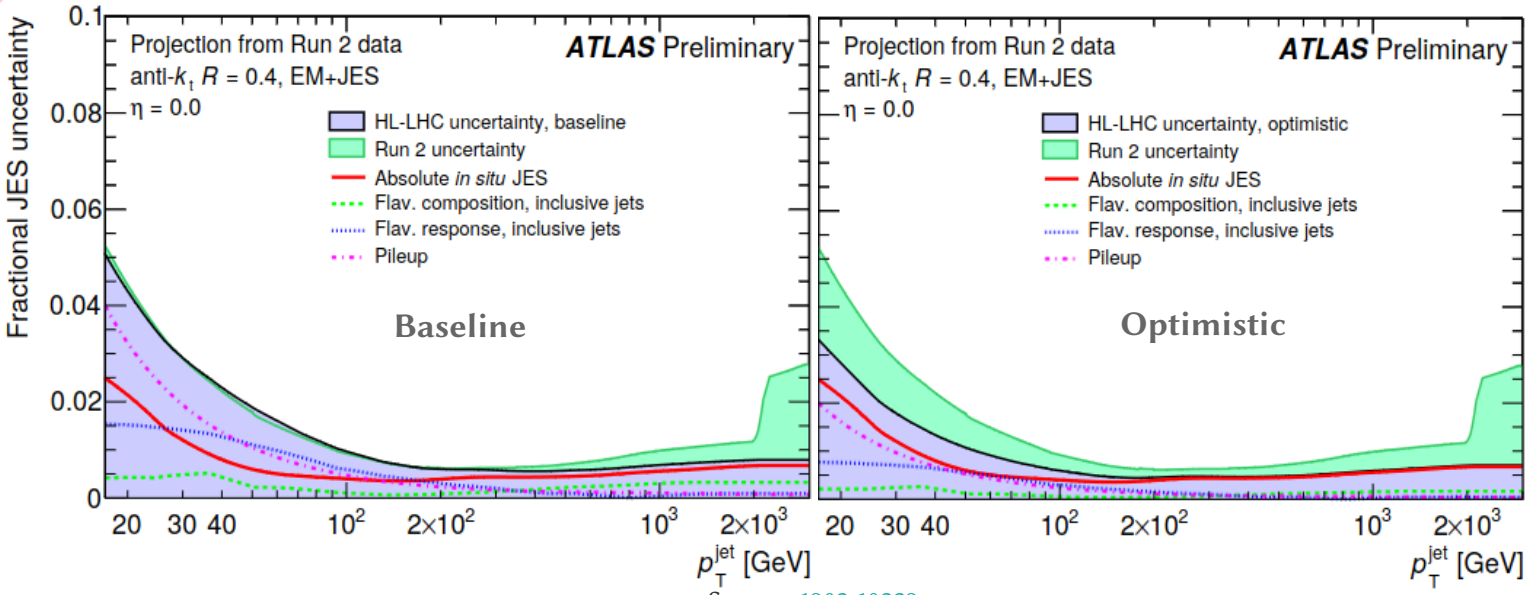
Pileup from multiple proton interactions currently miss-model data. This trend worsens with the number of interactions per bunch crossing leading to an explosive issue moving towards the HL-LHC:



→ HL-LHC jet performance projections for ATLAS 1902.10229:

Baseline ~ Run-2

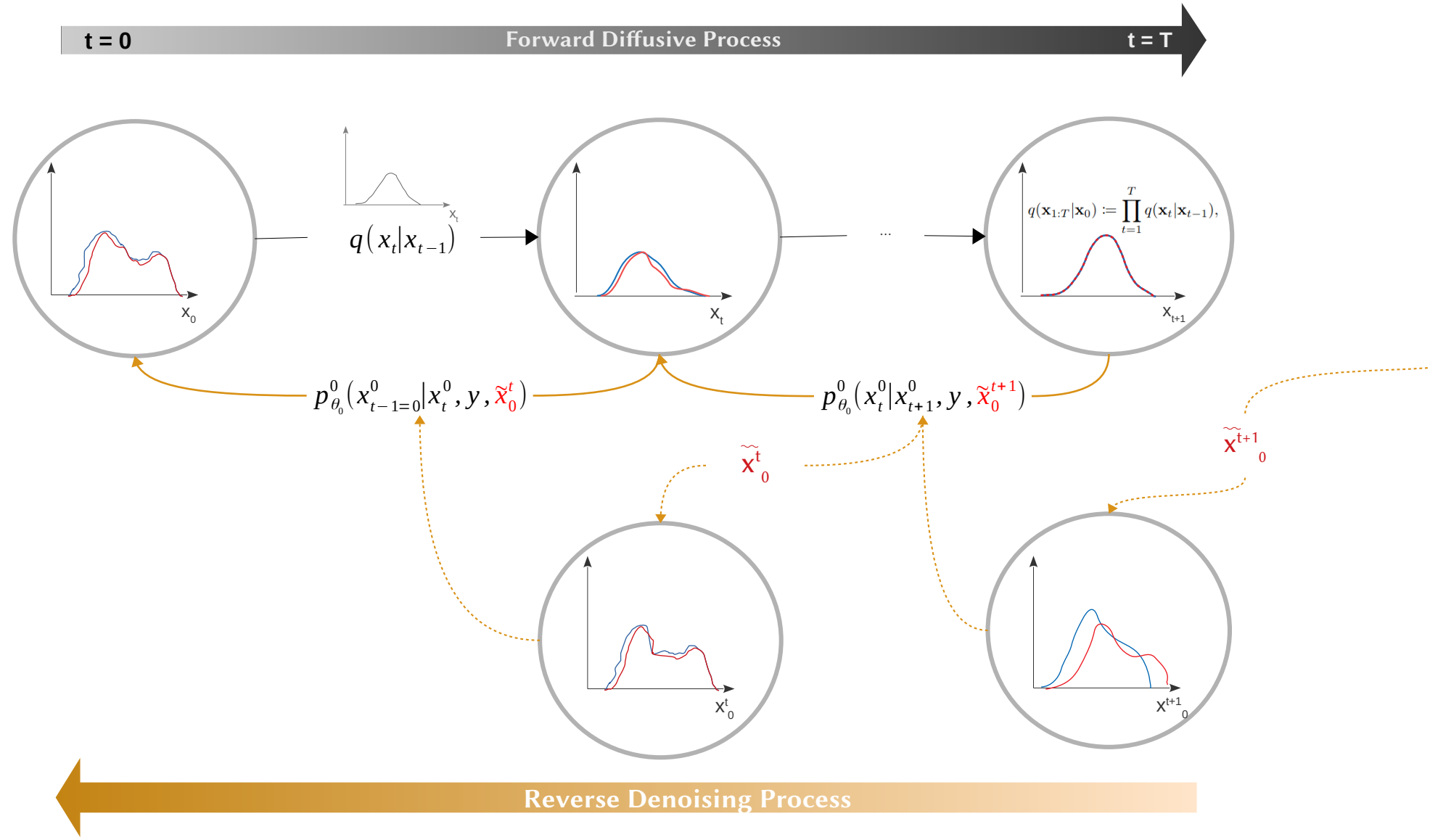
Optimistic: Requires improved understanding of jet fragmentation & pile-up



Source: 1902.10229

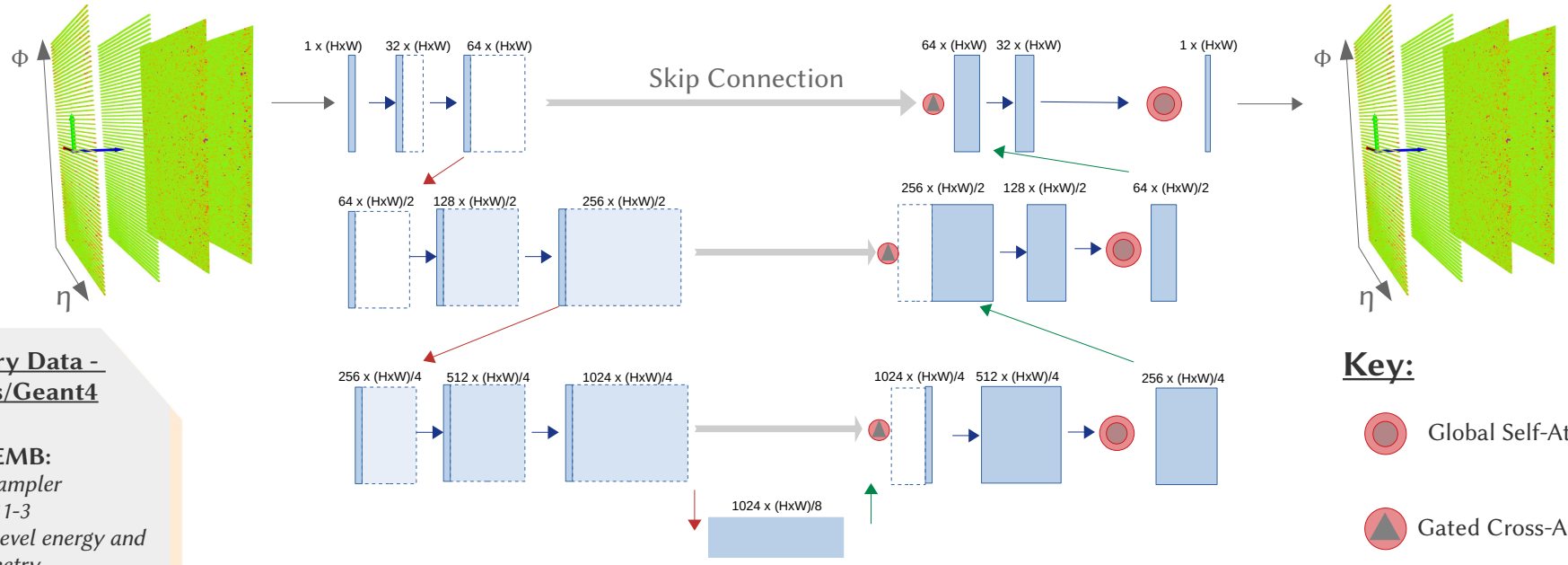
Self-Conditioning

Self-Conditioned DDPM



Neural Network Architecture

Latent DDPM



Calorimetry Data - Delphes/Geant4

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

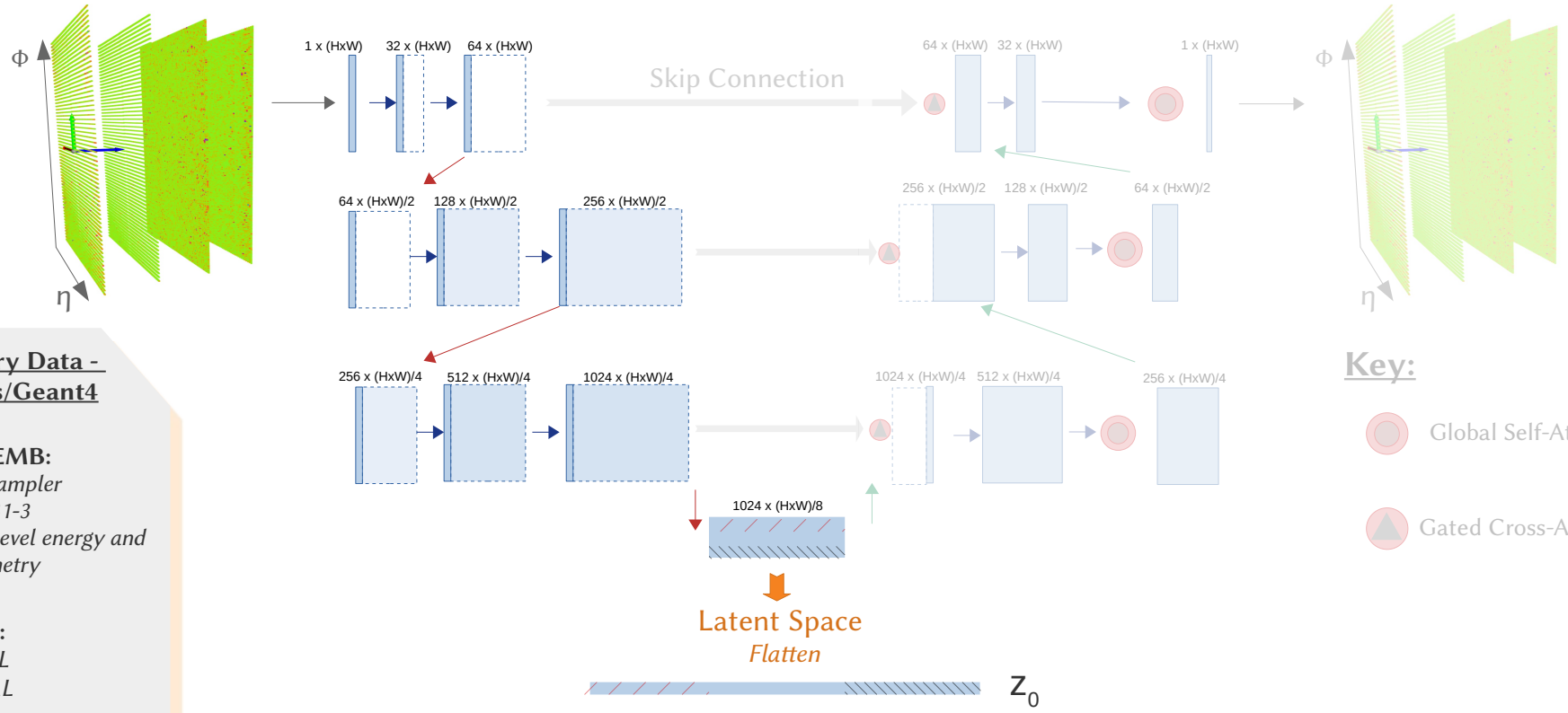
→ **Delphes:**
- → *ECAL*
- → *HCAL*

Key:

Global Self-Attention

Gated Cross-Attention

Latent DDPM

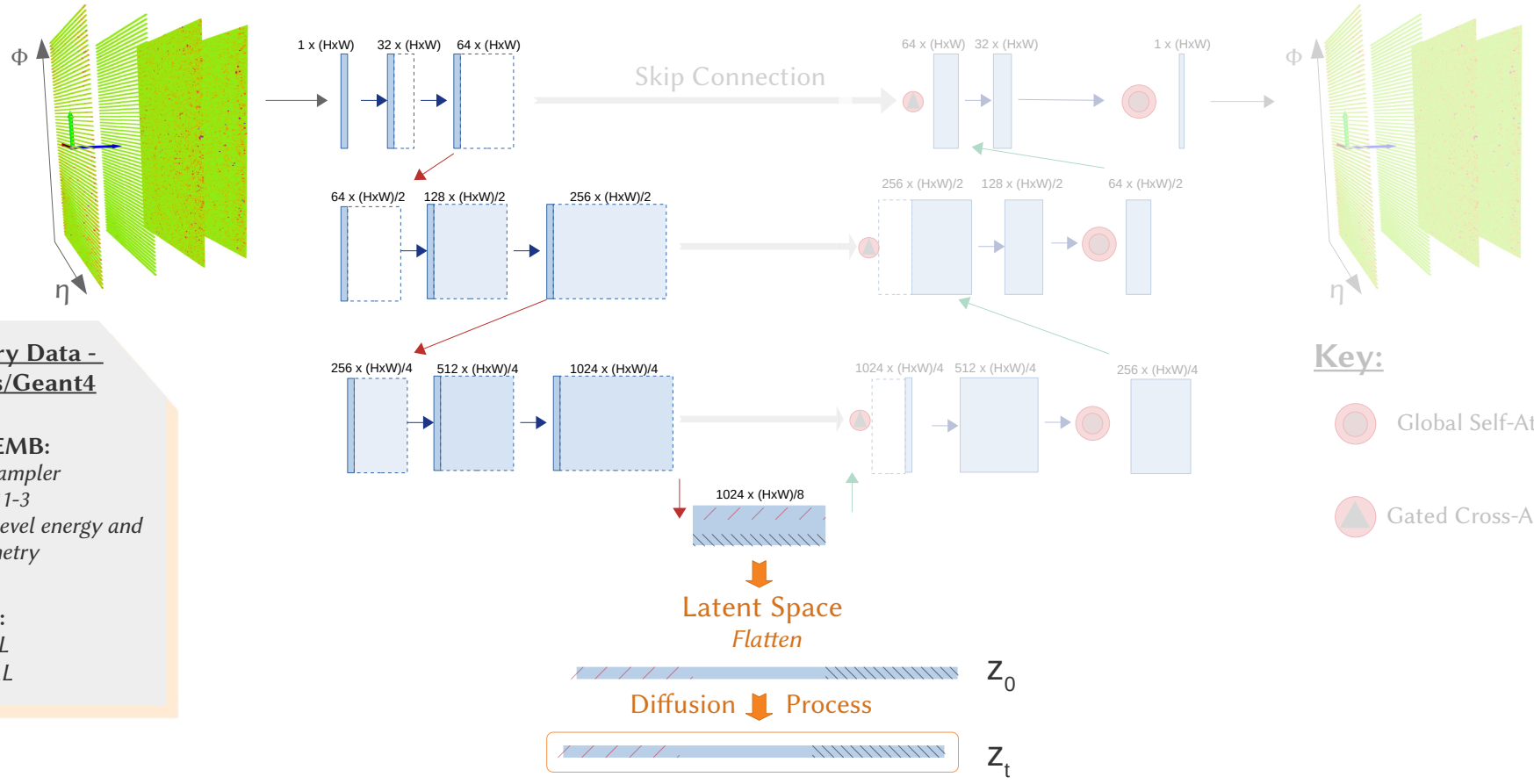


Calorimetry Data - Delphes/Geant4

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

→ **Delphes:**
- → *ECAL*
- → *HCAL*

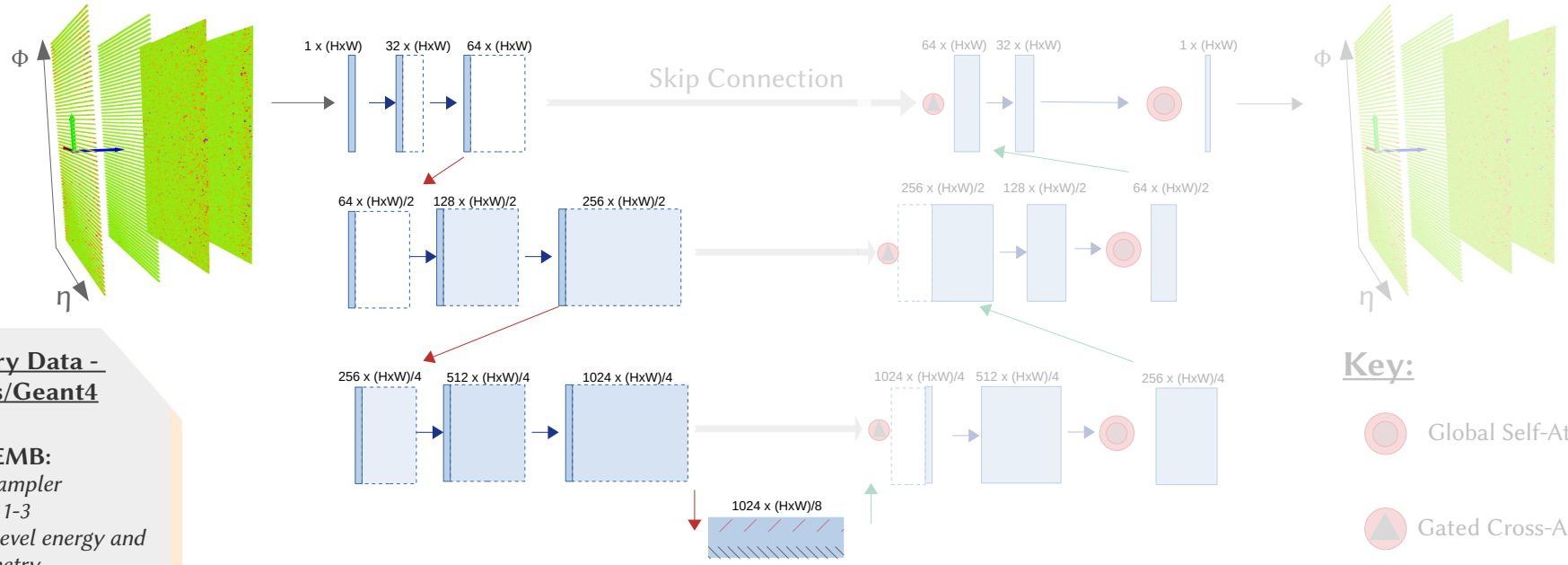
Latent DDPM



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*

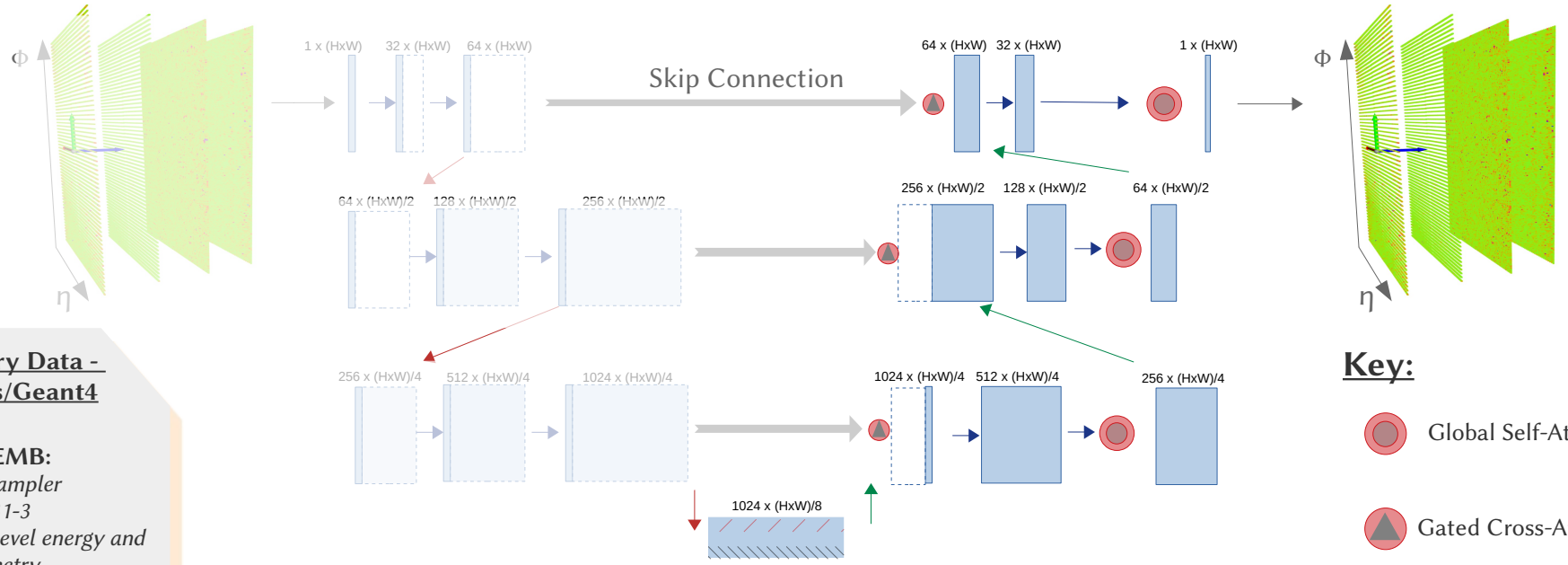
→ **Delphes:**
- → *ECAL*
- → *HCAL*

Key:

Global Self-Attention

Gated Cross-Attention

Latent DDPM



Calorimetry Data - Delphes/Geant4

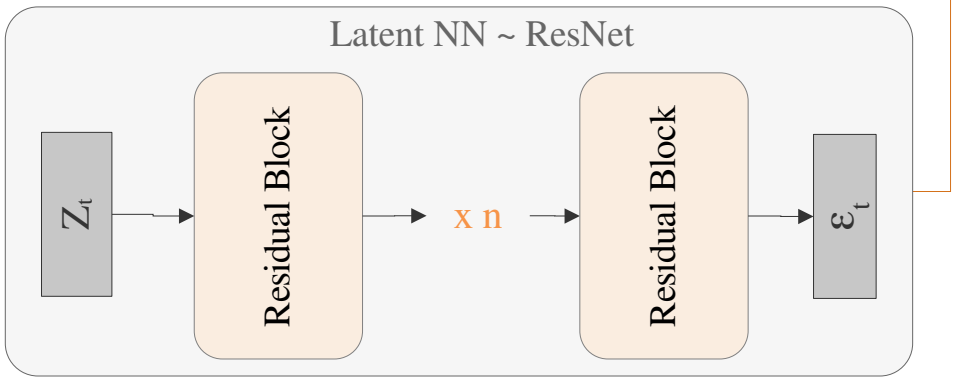
→ **ATLAS EMB:**
- → *Pre-sampler*
- → *EMB1-3*
- → *Cell level energy and geometry*

→ **Delphes:**
- → *ECAL*
- → *HCAL*

Key:

Global Self-Attention

Gated Cross-Attention



Denoising Diffusion Probabilistic Models

DDPMs – Recipe

→ **Noise Scheduler:**

Cosine β-scheduling noise injection:

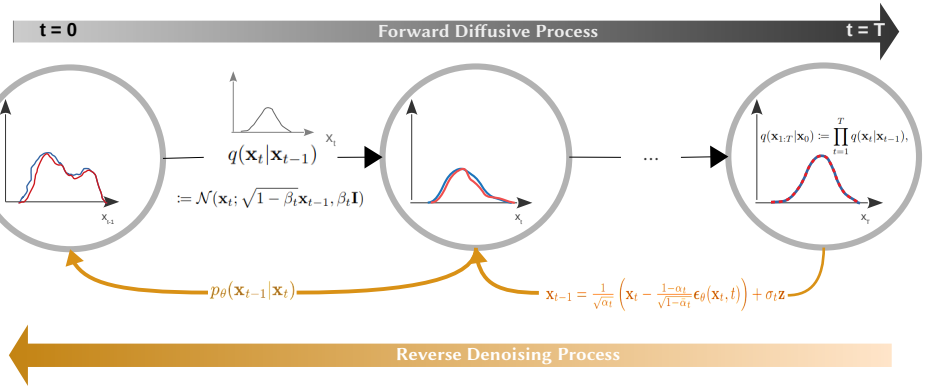
$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \qquad \overline{\alpha}_t = f(t)/f(0) \qquad f(t) = \cos\left(\frac{(t/T + s)\pi}{2.(1+s)}\right)$$

→ **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**

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



Denoising Diffusion Probabilistic Models

DDPMs – Recipe

→ **Noise Scheduler:**

Cosine β-scheduling noise injection:

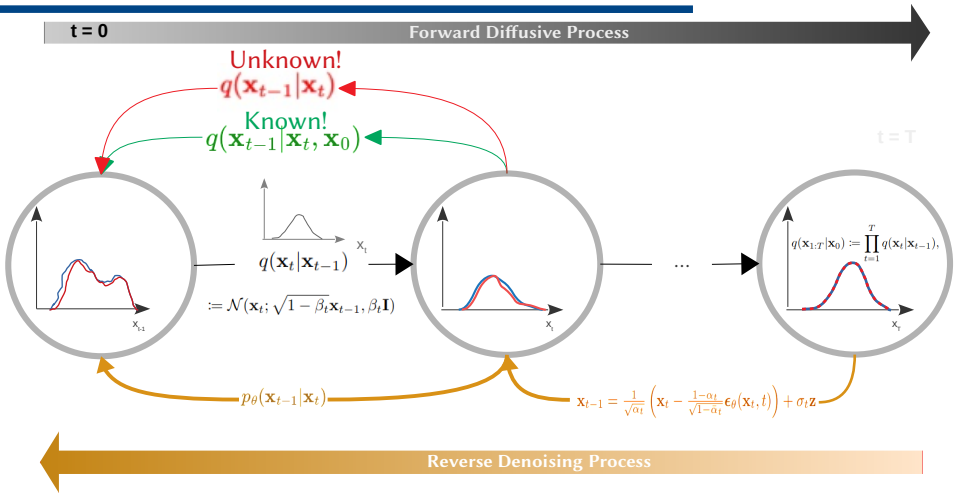
$$\beta_t = 1 - \frac{\overline{\alpha}_t}{\overline{\alpha}_{t-1}} \qquad \overline{\alpha}_t = f(t)/f(0) \qquad f(t) = \cos\left(\frac{(t/T + s)\pi}{2.(1+s)}\right)$$

→ **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**

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



→ 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:

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

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

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**

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

→ **Density Ratio Estimating Classifer**

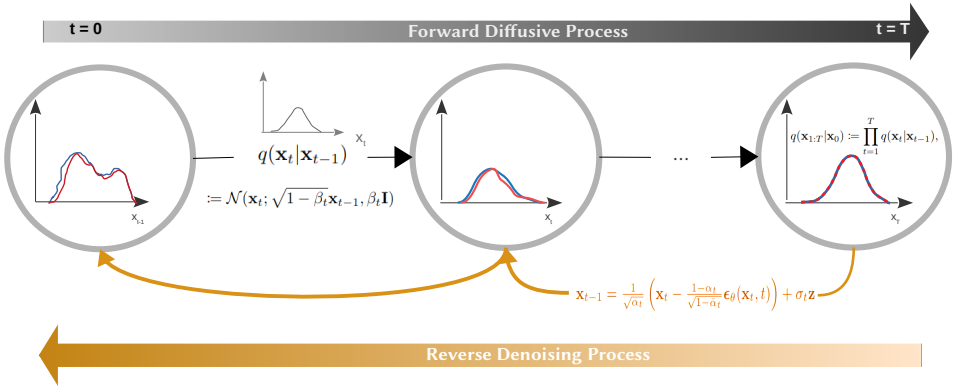
$$L_{DRE} = \sum [p_c(t, x_0) \log(\widehat{p}_c(t, x_0))]$$

→ **Conditional Embedding**

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

→ **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:

$$q(\boldsymbol{x_{t-1}}|x_t, x_{0,y}) = q(\boldsymbol{x_t}|x_{t-1}, x_{0,y}) \cdot \frac{q(x_{t-1}|x_{0,y})}{q(x_t|x_{0,y})}$$

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

→ Score estimation naturally extends with this conditionality:

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

→ 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**

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

→ **Density Ratio Estimating Classifer**

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

→ **Conditional Embedding**

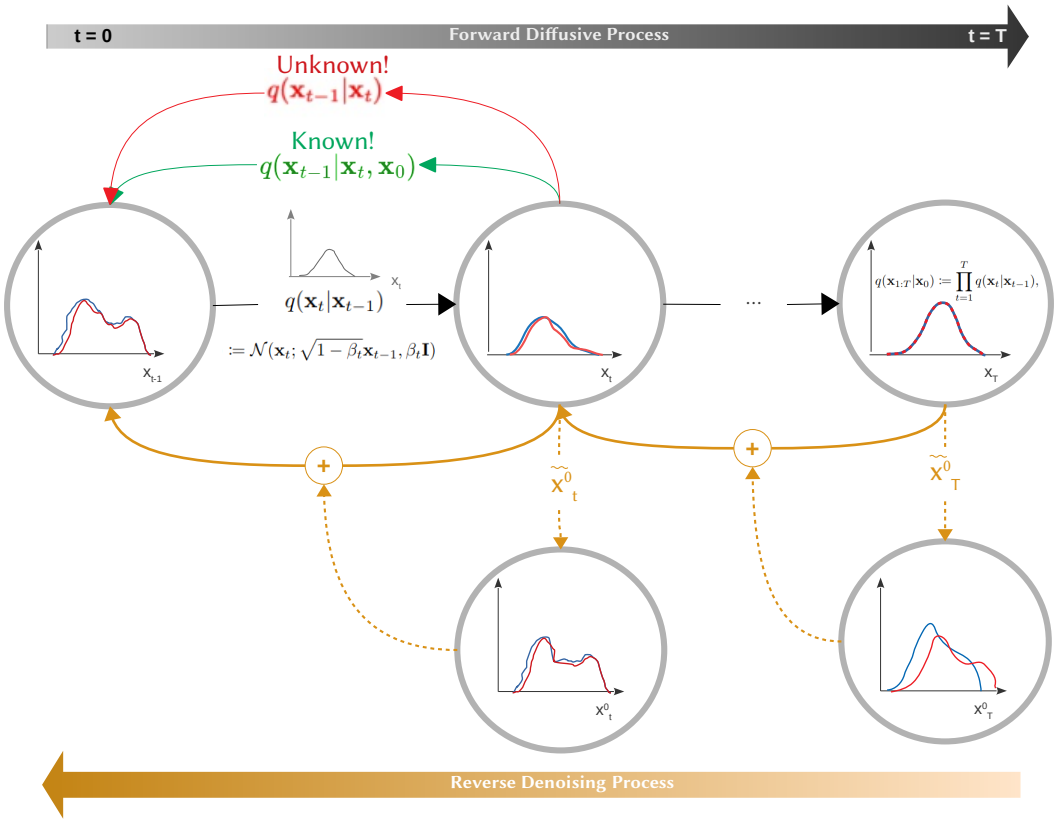
$$L(\theta) = \mathbb{E}_{t, x_0, \epsilon, y}\left[\left|\epsilon - \epsilon_\theta(x_t(x_0, \epsilon), y, t)^2\right|\right]$$

→ **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**

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



→ 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:

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

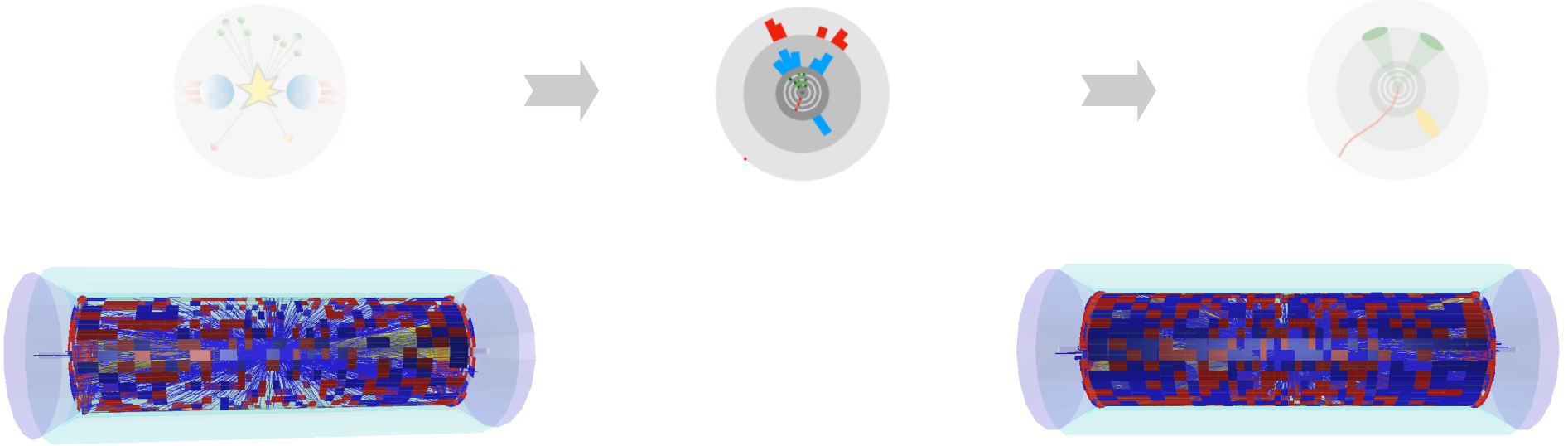
Delphes Pile-up Simulation

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:

SOFT QCD GEN. → DETECTOR SIM. → RECO.



Denoise Score Matching Neural Network

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**

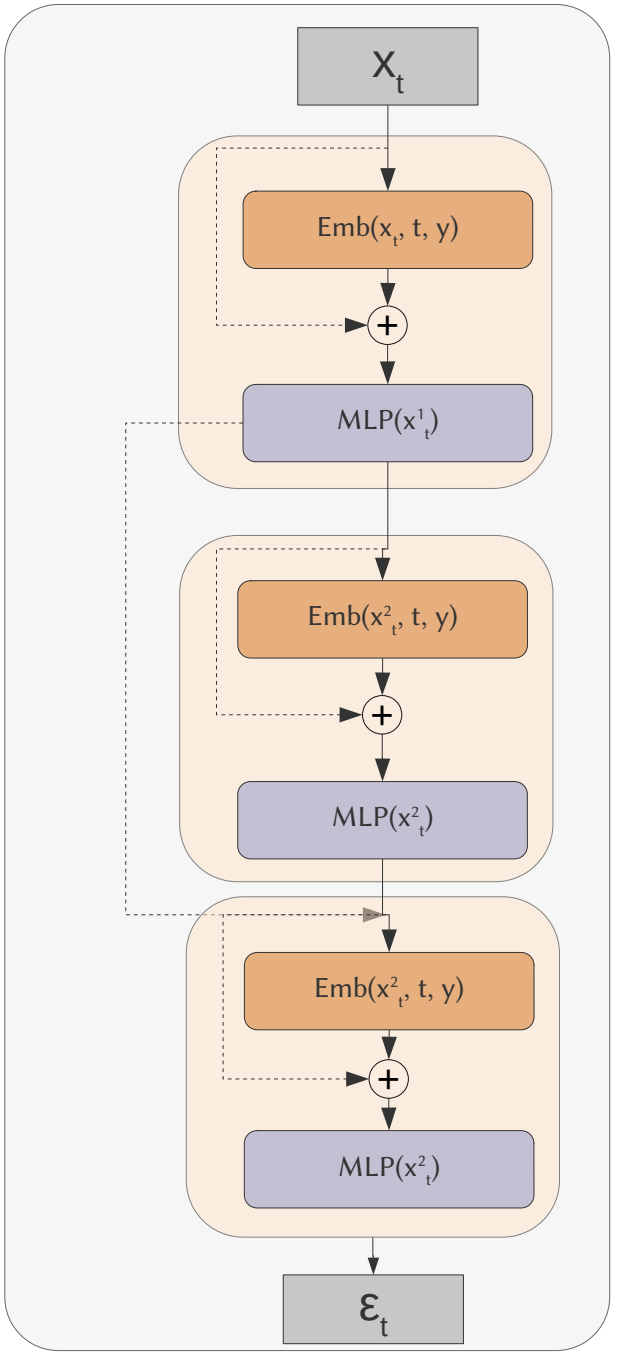
$$PE_{t,2i} = \sin\left(t / \alpha^{2i/d_{emb}}\right)$$

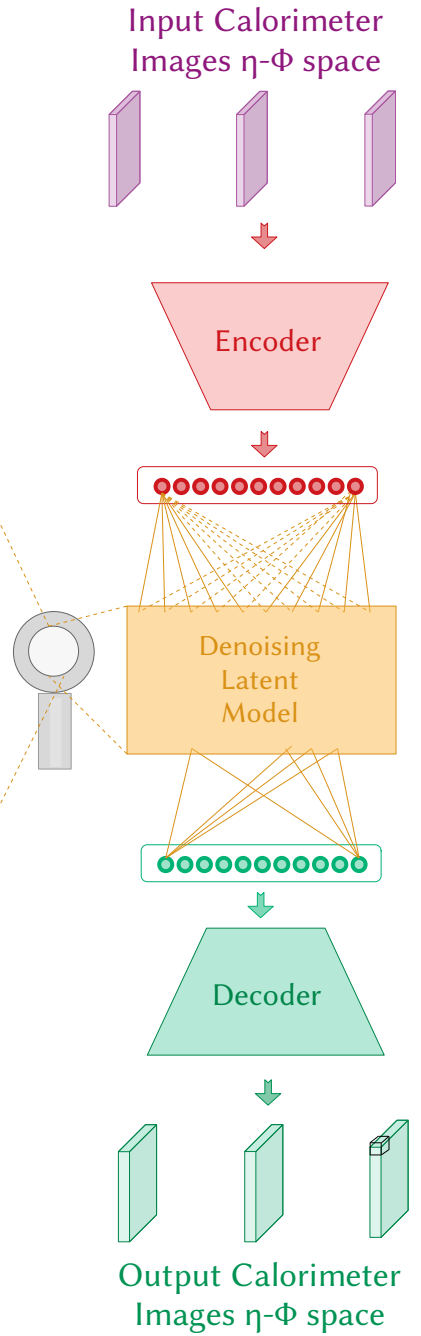
$$PE_{t,2i+1} = \cos\left(t / \alpha^{(2i+1)/d_{emb}}\right)$$

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**

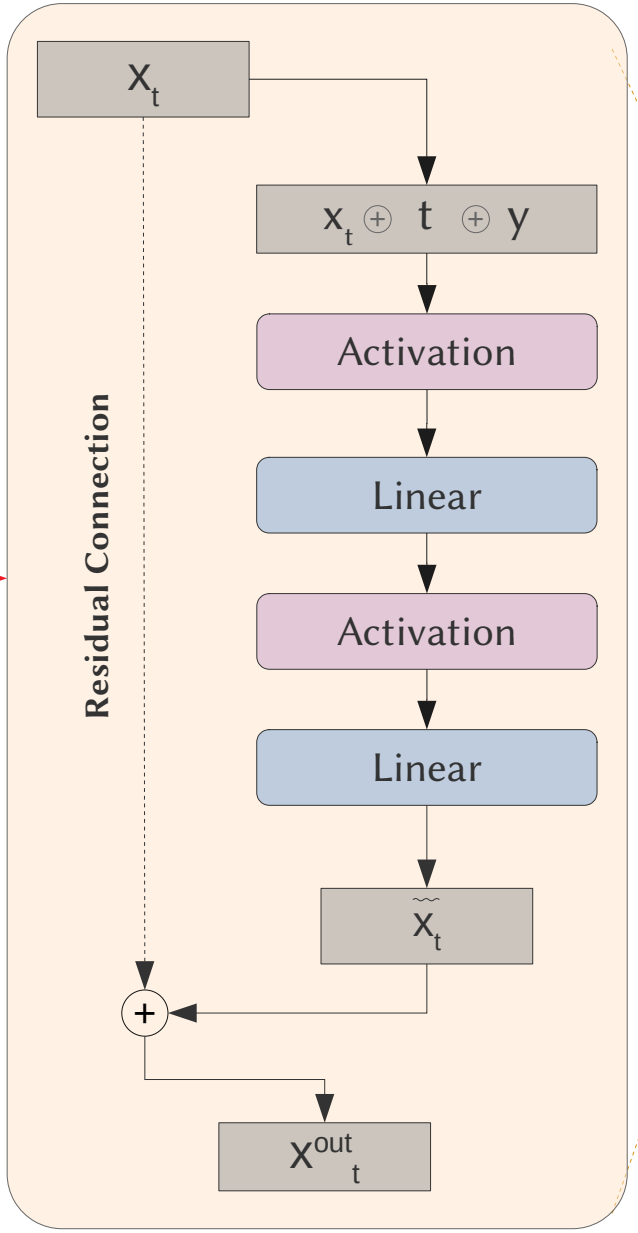
$$PE_{t,2i} = \sin\left(t/\alpha^{2i/d_{emb}}\right)$$

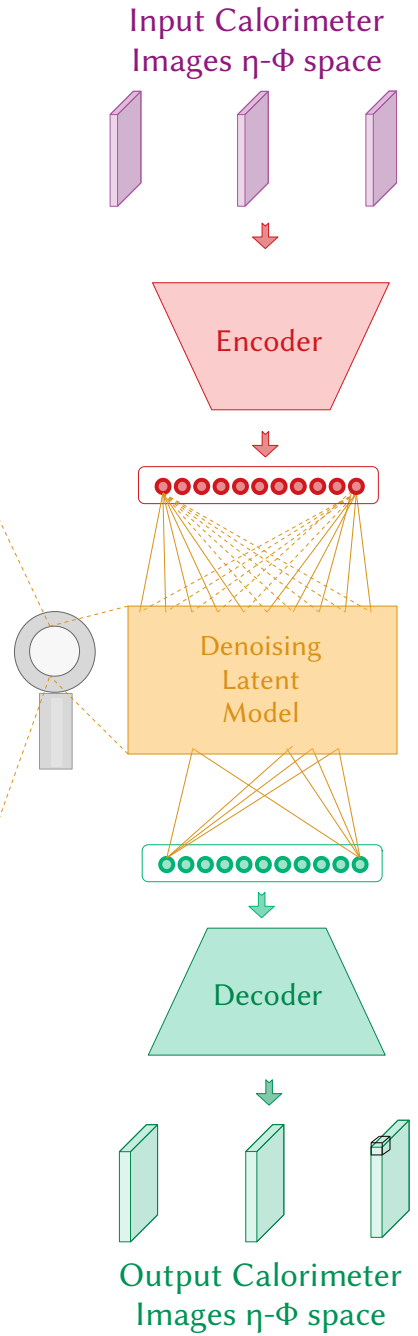
$$PE_{t,2i+1} = \cos\left(t/\alpha^{(2i+1)/d_{emb}}\right)$$

Natural normalisation of conditional variables that scales dimensionally with embedding space

→ **FFN Blocks**

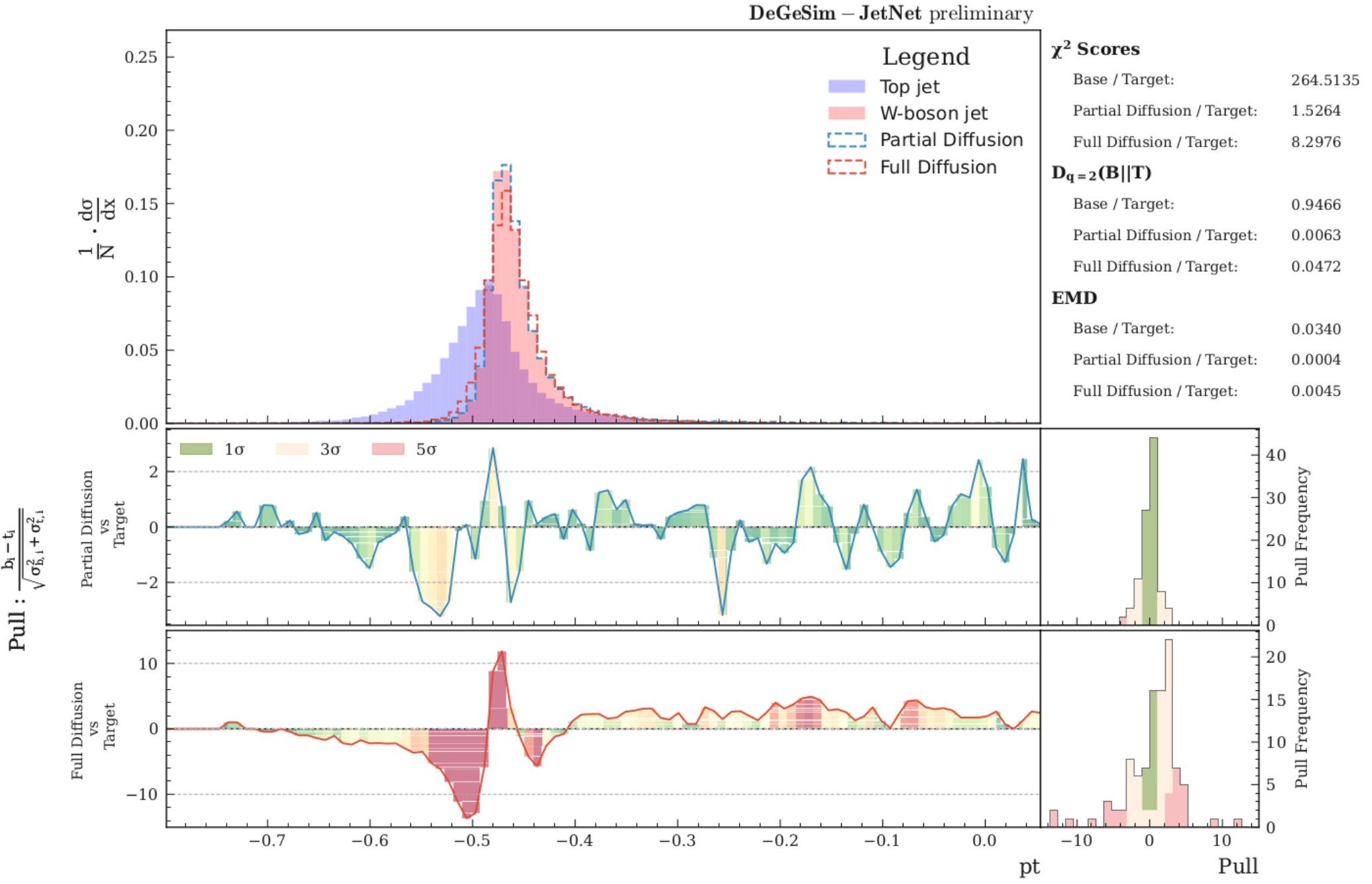
Simple dense feed-forward neural network blocks for dimensionality scaling and expressibility



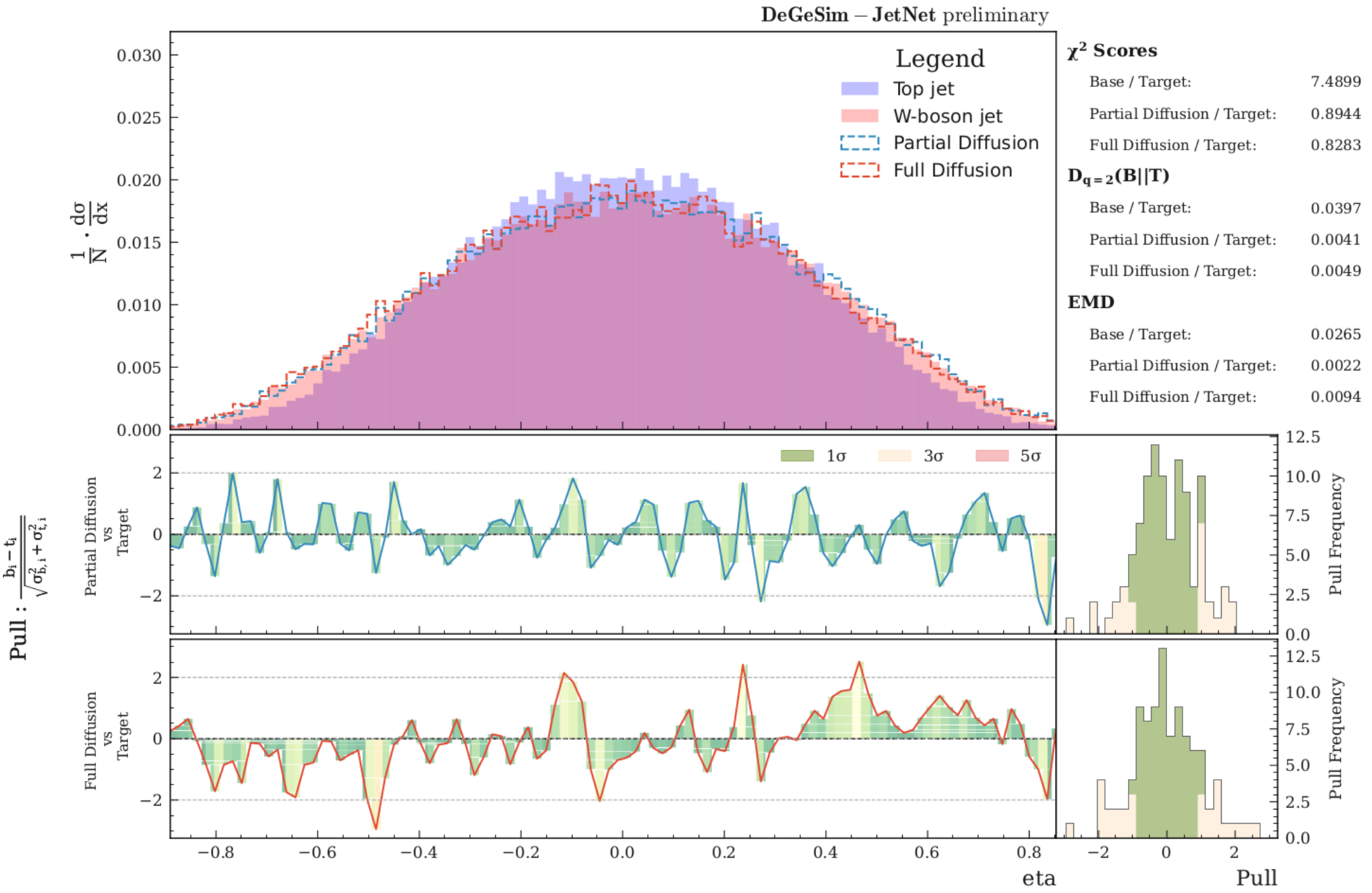


JetNet Data

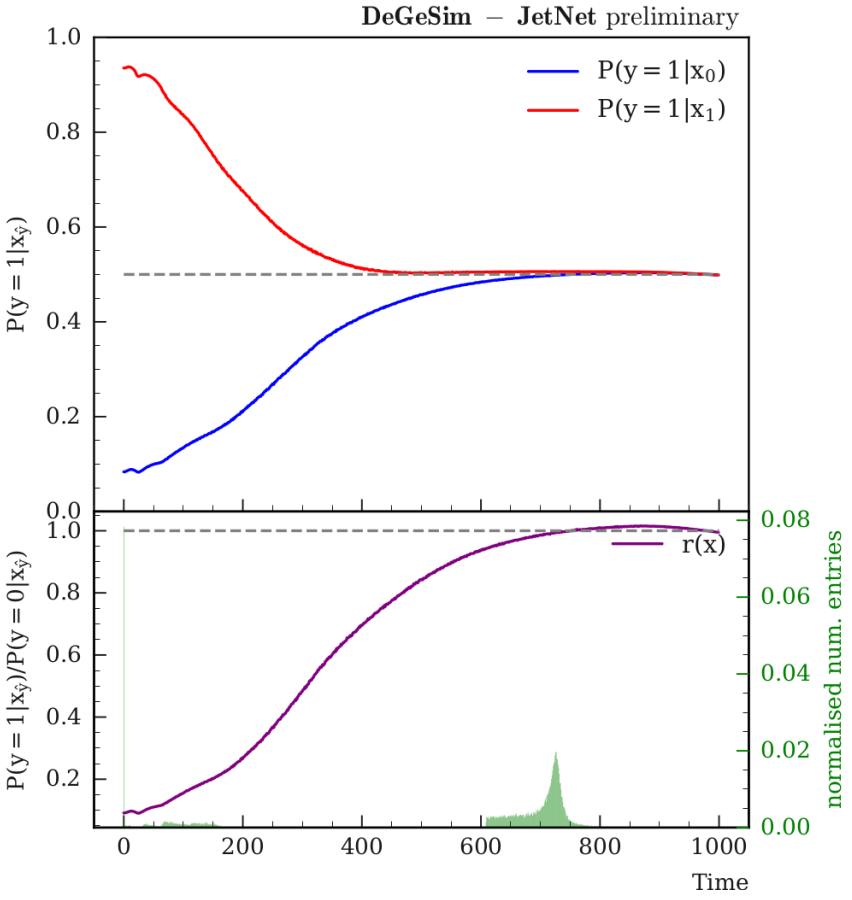
JetNet: Top -vs- W jets



JetNet: Top -vs- W jets



JetNet: Top -vs- W jets



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

$$r(x|t,\theta_0,\theta_1) = \frac{\mathcal{L}(\theta_1|x,t)}{\mathcal{L}(\theta_0|x,t)} = 1$$