The surrogate revolution: Generative models for fast detector simulation & more

Prof. Gregor Kasieczka
Email: gregor.kasieczka@uni-hamburg.de
Twitter/X: @GregorKasieczka

U Tokyo HEP Seminar — 12.1.2024

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

KISS

FSP CMS

PUNCH 4NFDI

DASHH

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

UHH Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

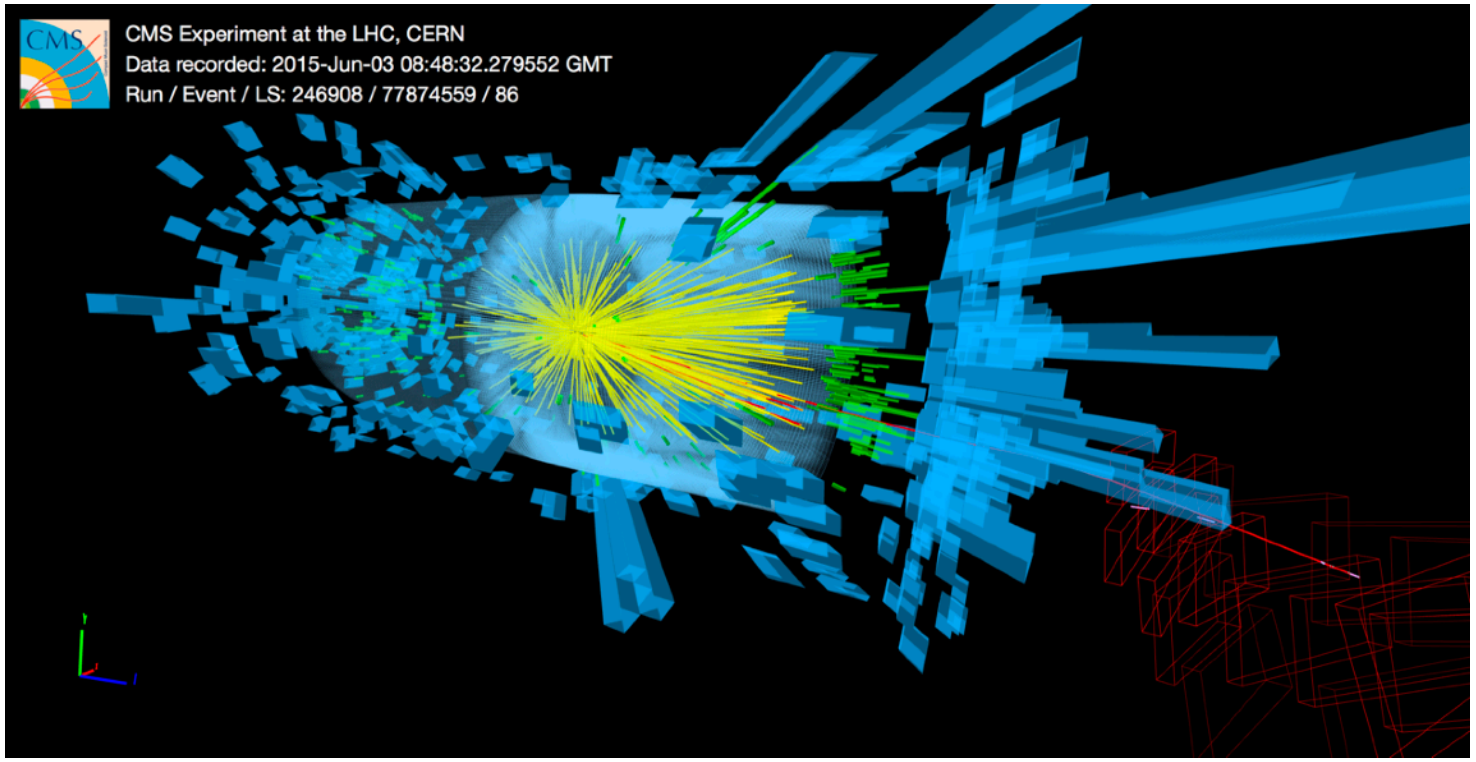
CDCS
CENTER FOR DATA AND COMPUTING IN NATURAL SCIENCES

PIER
Partnership of Universität Hamburg and DESY

Emmy Noether-Programm
Deutsche Forschungsgemeinschaft DFG

Data in HEP

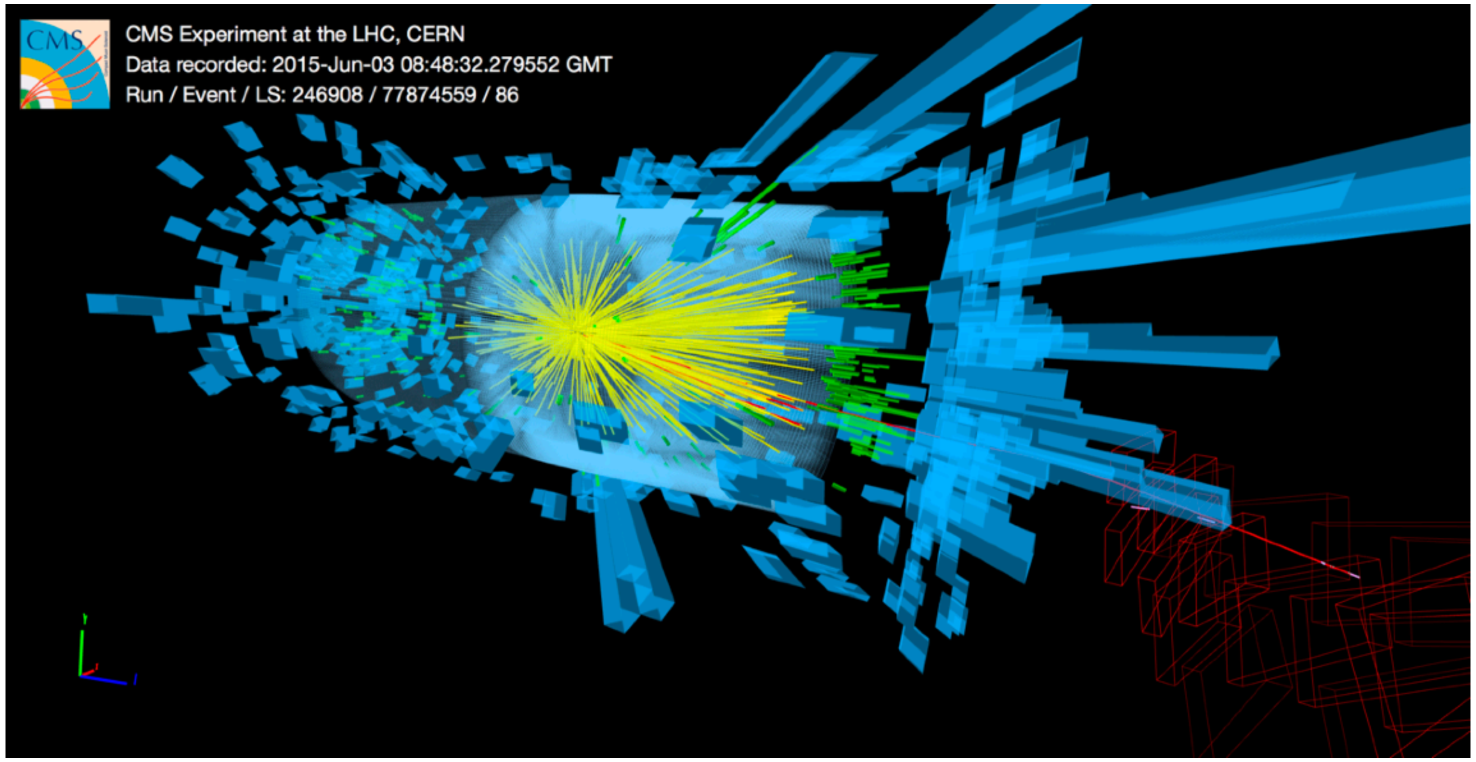
- Collisions:
  ~1 MB/event at 40 MHz
- Reduce to ~1 kHz
  via lossy, irreversible filtering algorithms (Trigger)
- Heterogenous data:
  ~100M low-level readouts



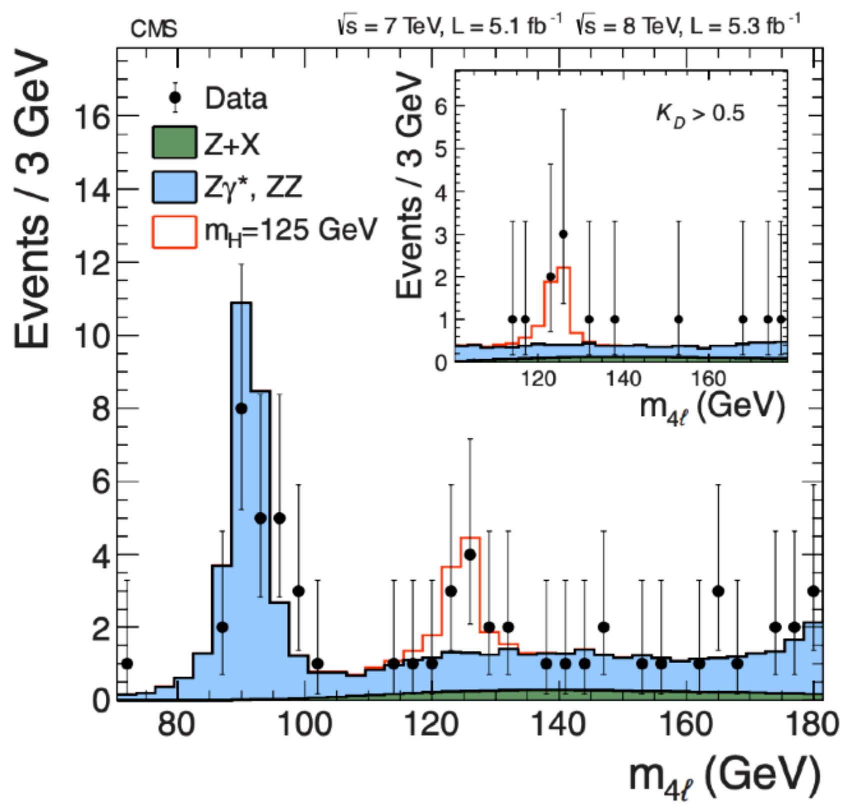
The data

Data in HEP

- Collisions:
  ~1 MB/event at 40 MHz
- Reduce to ~1 kHz
  via lossy, irreversible filtering algorithms (Trigger)
- Heterogenous data:
  ~100M low-level readouts



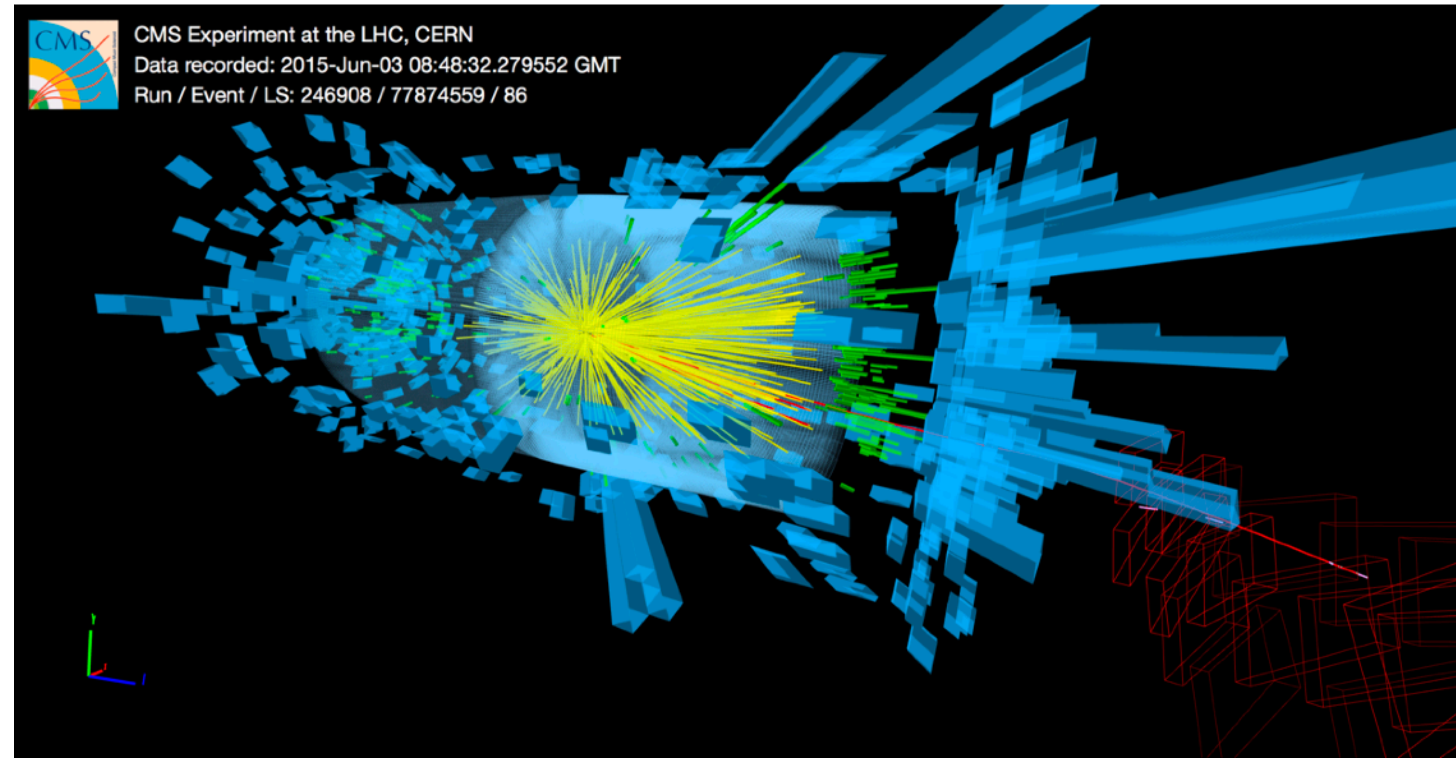
The data



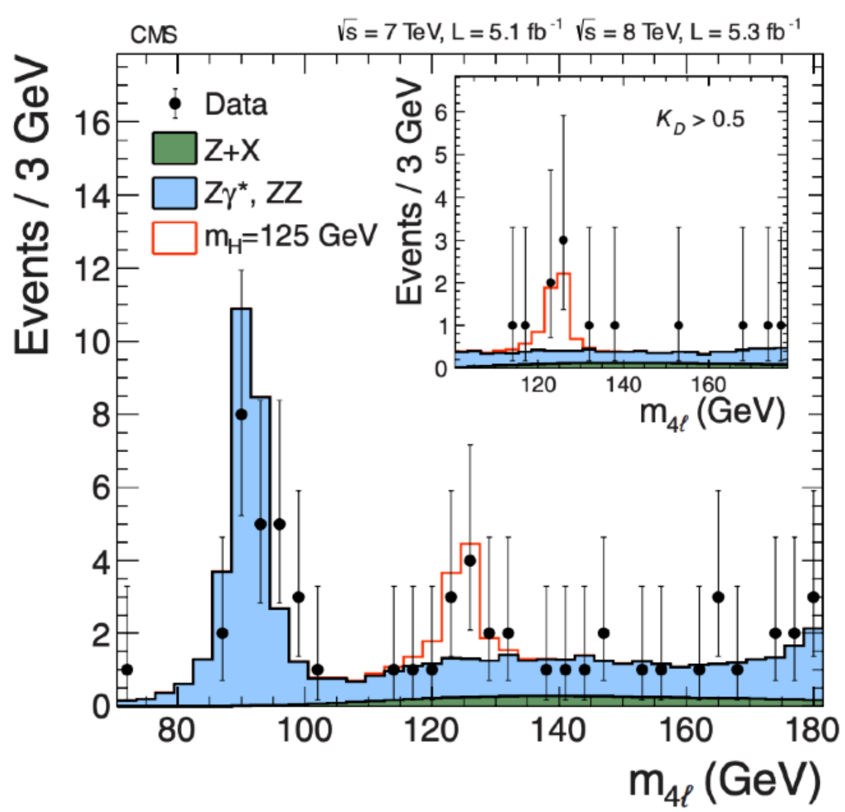
How experimentalists think of the data

Data in HEP

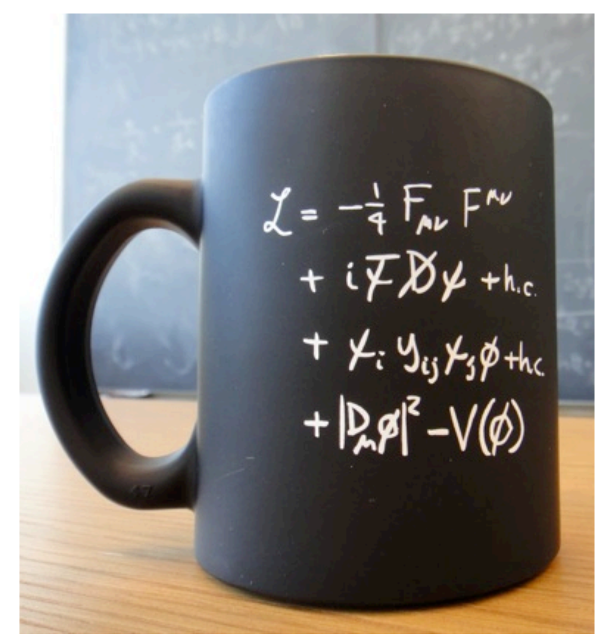
- Collisions:
  ~1 MB/event at 40 MHz
- Reduce to ~1 kHz
  via lossy, irreversible filtering algorithms (Trigger)
- Heterogenous data:
  ~100M low-level readouts



The data



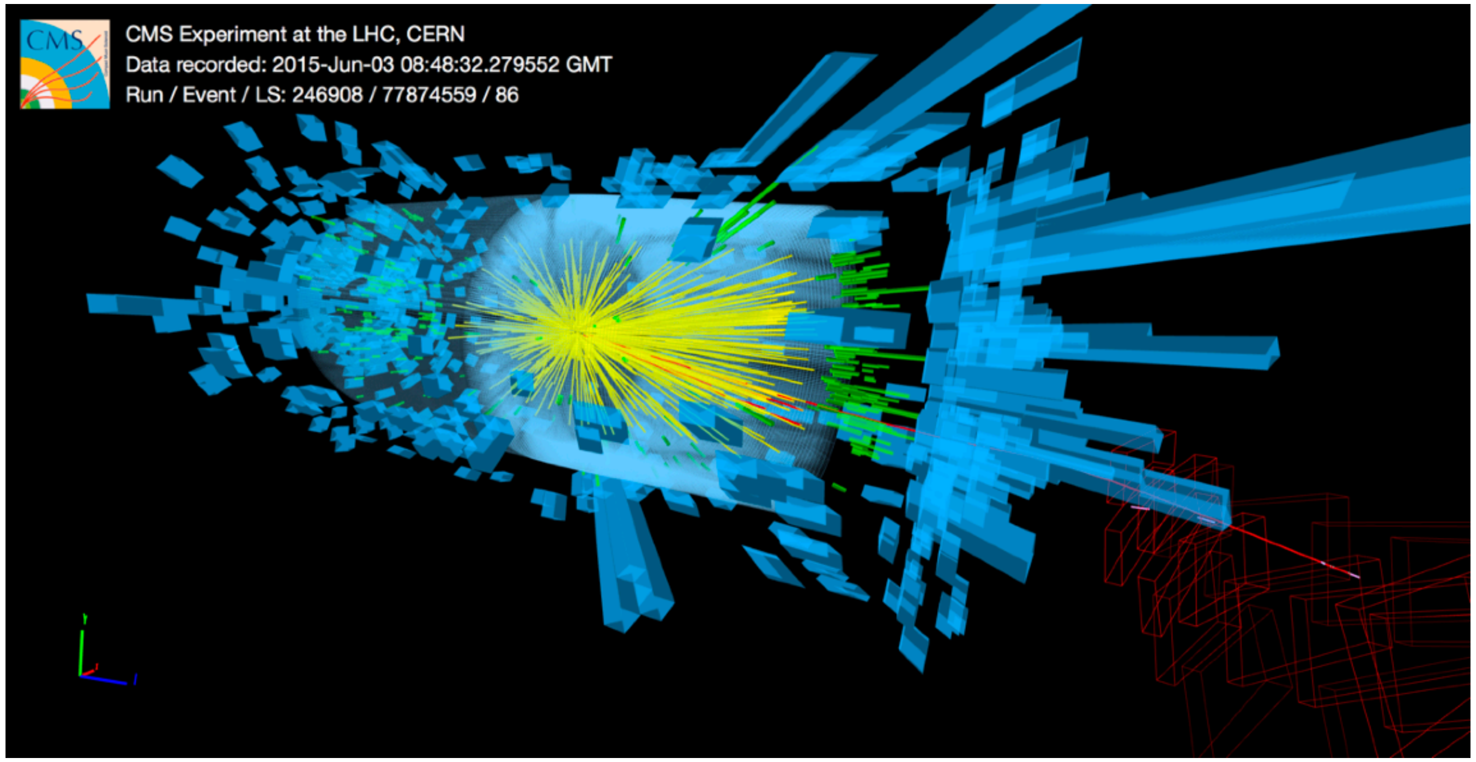
How experimentalists think of the data



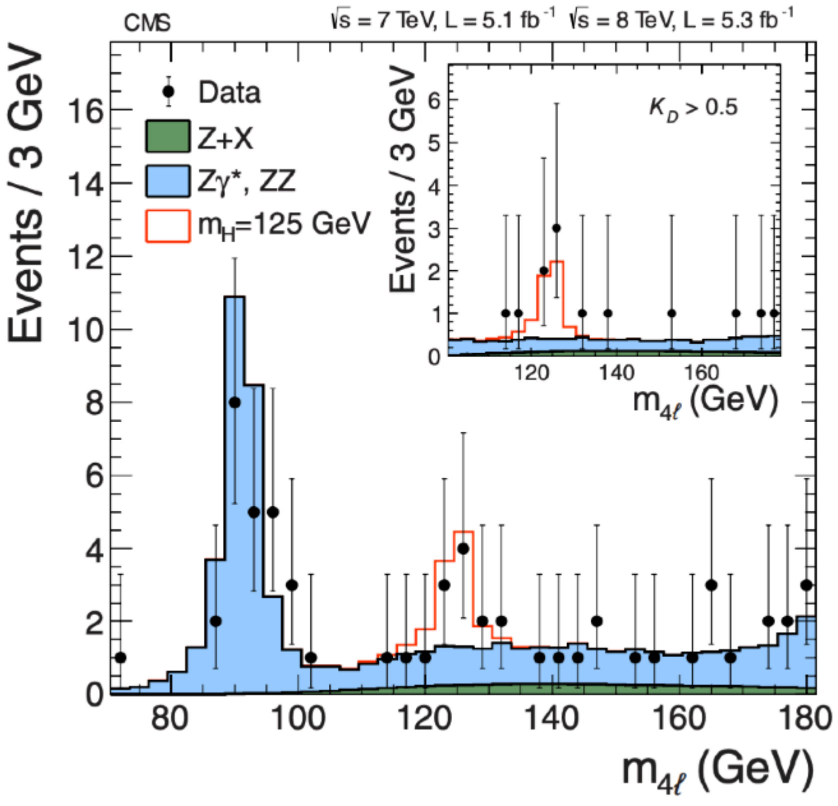
How theorists think of the data

Data in HEP

- Collisions:
  ~1 MB/event at 40 MHz
- Reduce to ~1 kHz
  via lossy, irreversible filtering algorithms (Trigger)
- Heterogenous data:
  ~100M low-level readouts



The data



How experimentalists think of the data

$$p(x)$$

How generative models think of the data

Data in HEP

Sample $X_i \sim p(x)$
to generate datapoints

(Focus of this seminar)

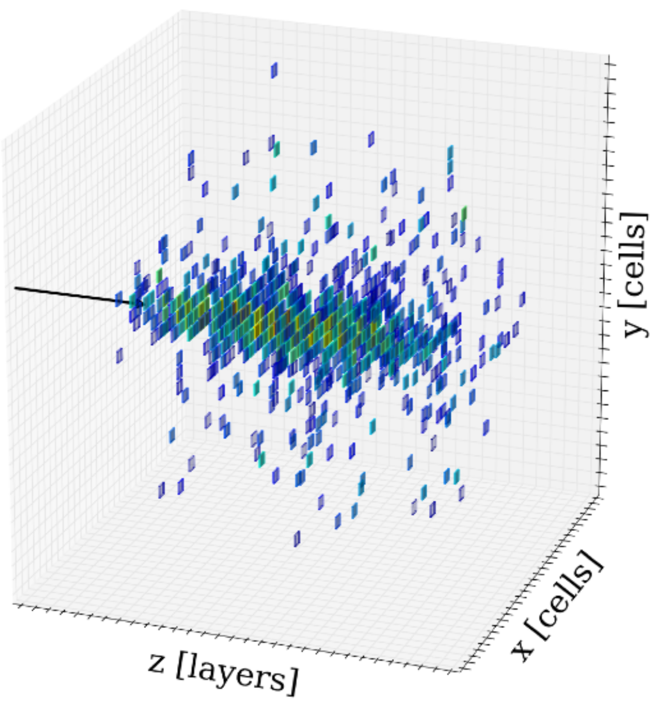
Evaluate p(x) directly as
likelihood

(Less explored, but see
e.g. ANODE / R-ANODE)

$$p(x)$$

How generative models
think of the data

Data in HEP

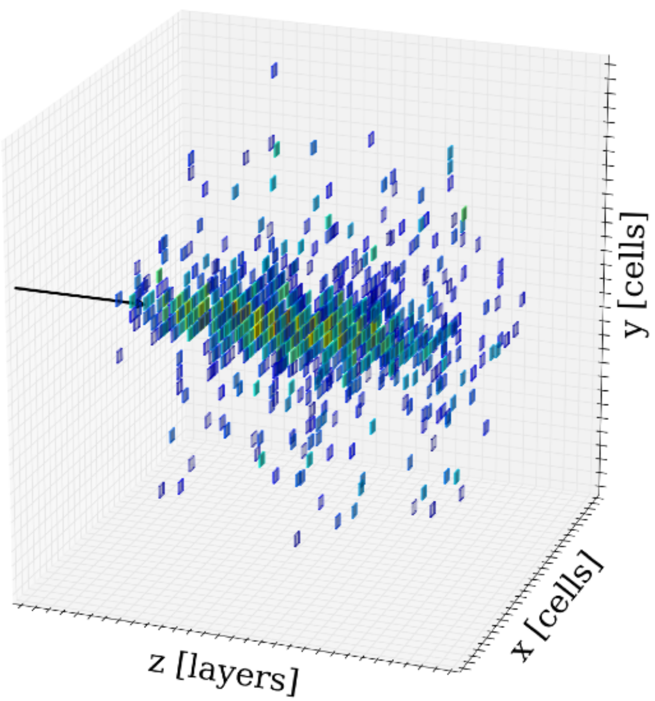


Showers in complex high-resolution calorimeters

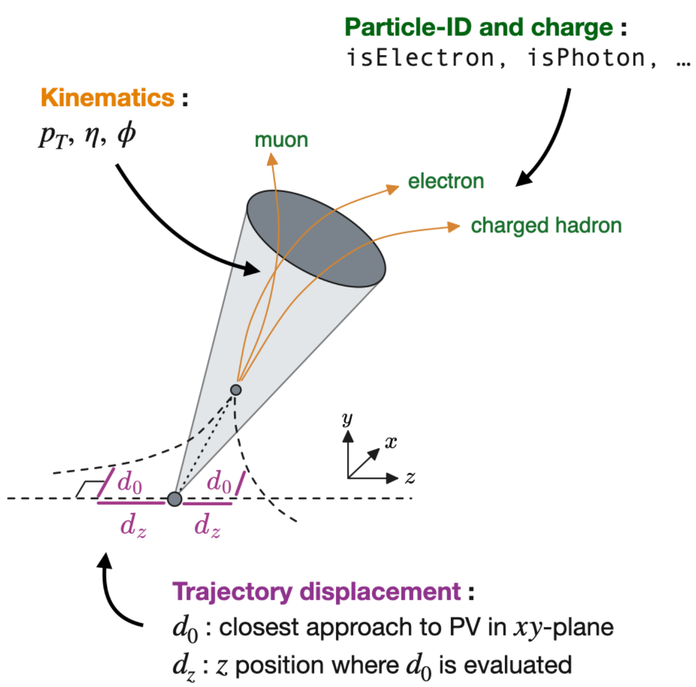
Sample $X_i \sim p(x)$ to generate datapoints

(Focus of this seminar)

Data in HEP



Showers in complex high-resolution calorimeters

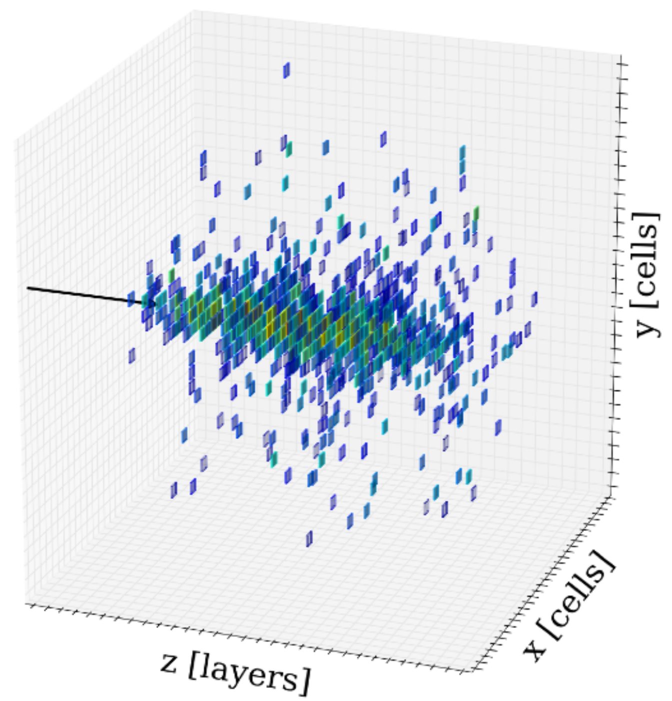


Jet constituents

Sample $X_i \sim p(x)$
to generate datapoints

(Focus of this seminar)

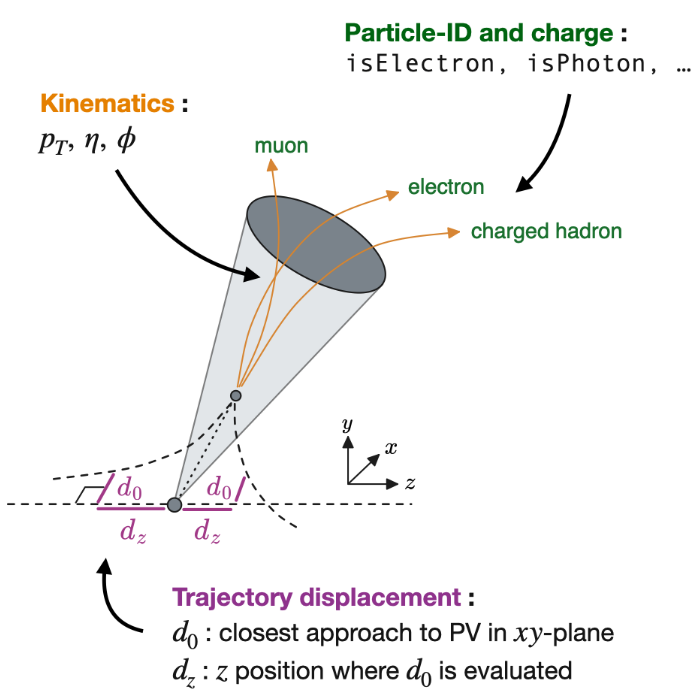
Data in HEP



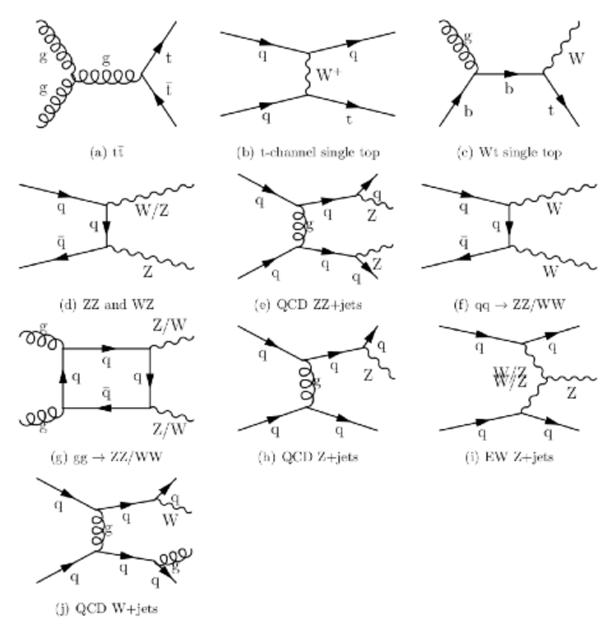
Showers in complex high-resolution calorimeters

Sample $X_i \sim p(x)$ to generate datapoints

(Focus of this seminar)

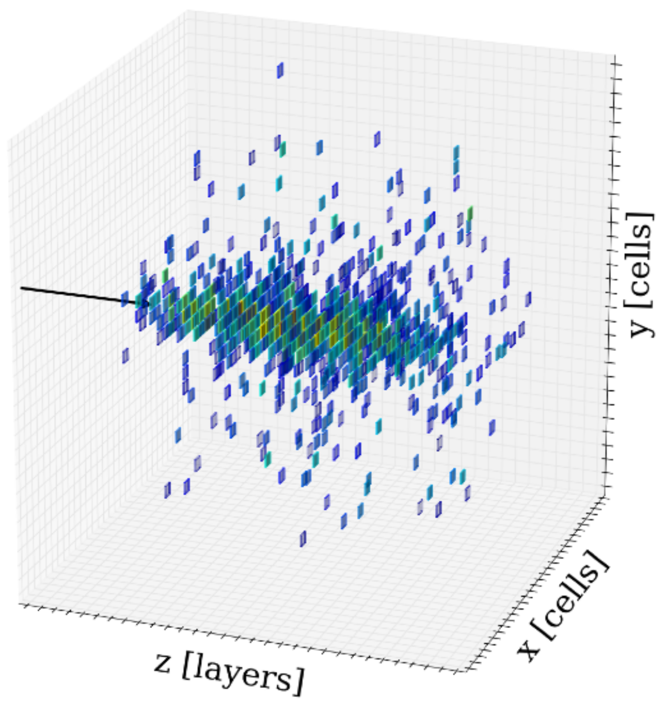


Jet constituents



Event-level kinematics

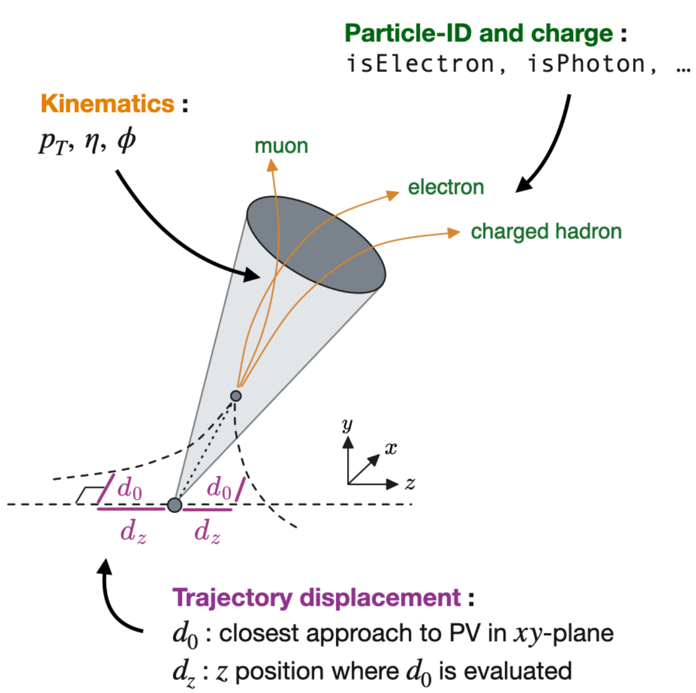
Data in HEP



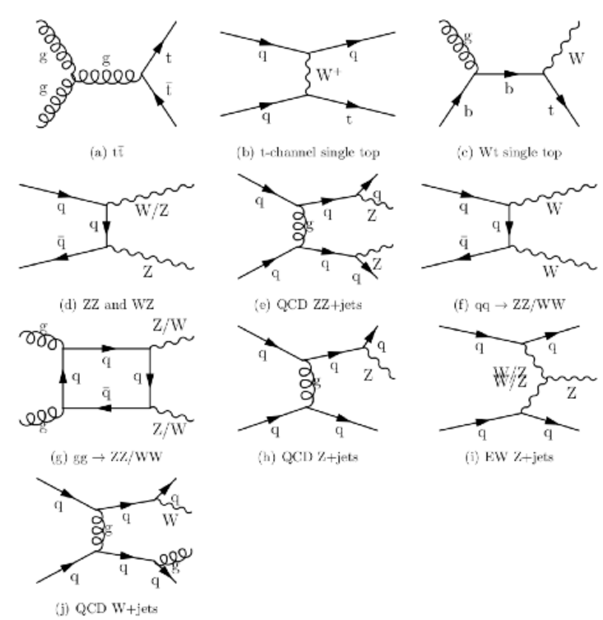
Showers in complex high-resolution calorimeters

Sample $X_i \sim p(x)$
to generate datapoints

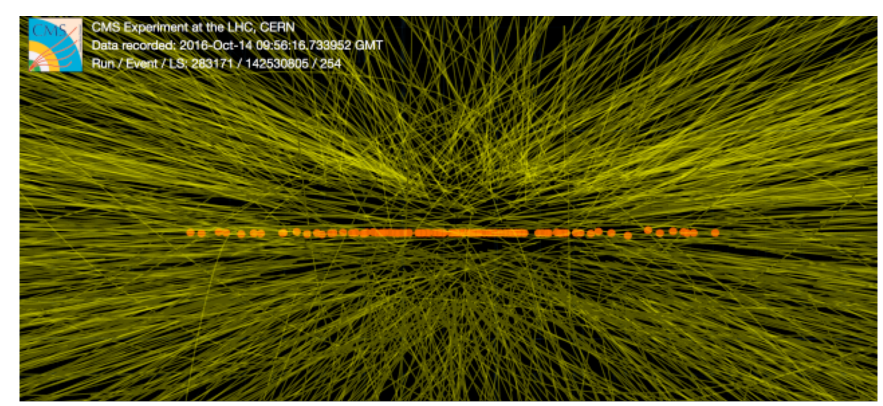
(Focus of this seminar)



Jet constituents

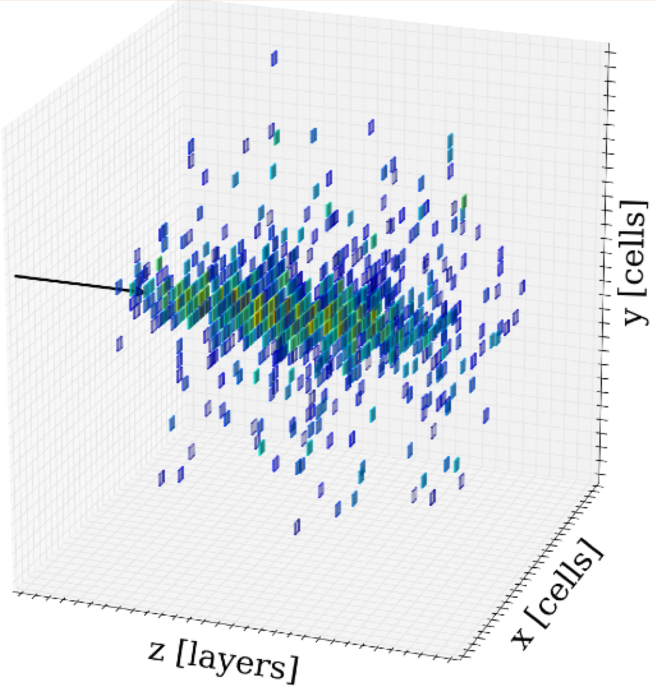


Event-level kinematics



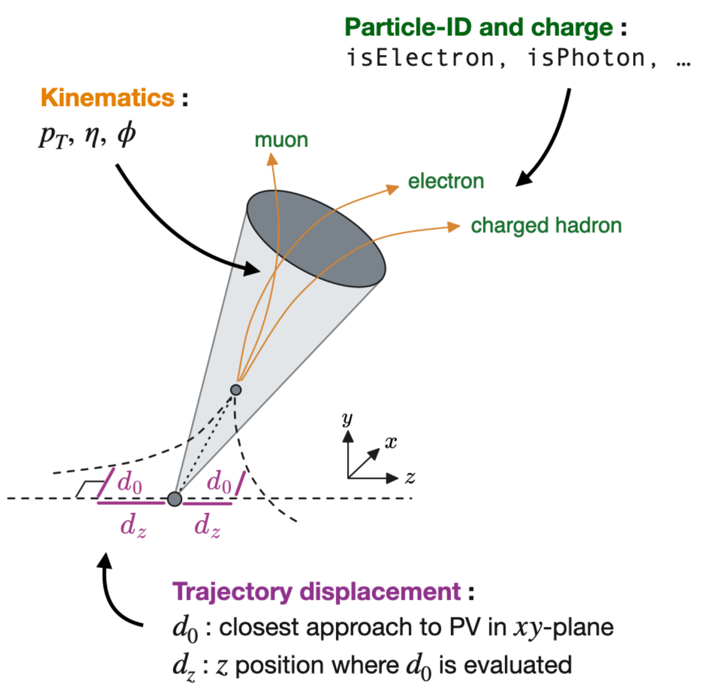
Pile-up Interactions

Data in HEP



Showers in complex high-resolution calorimeters

Cosmic air showers



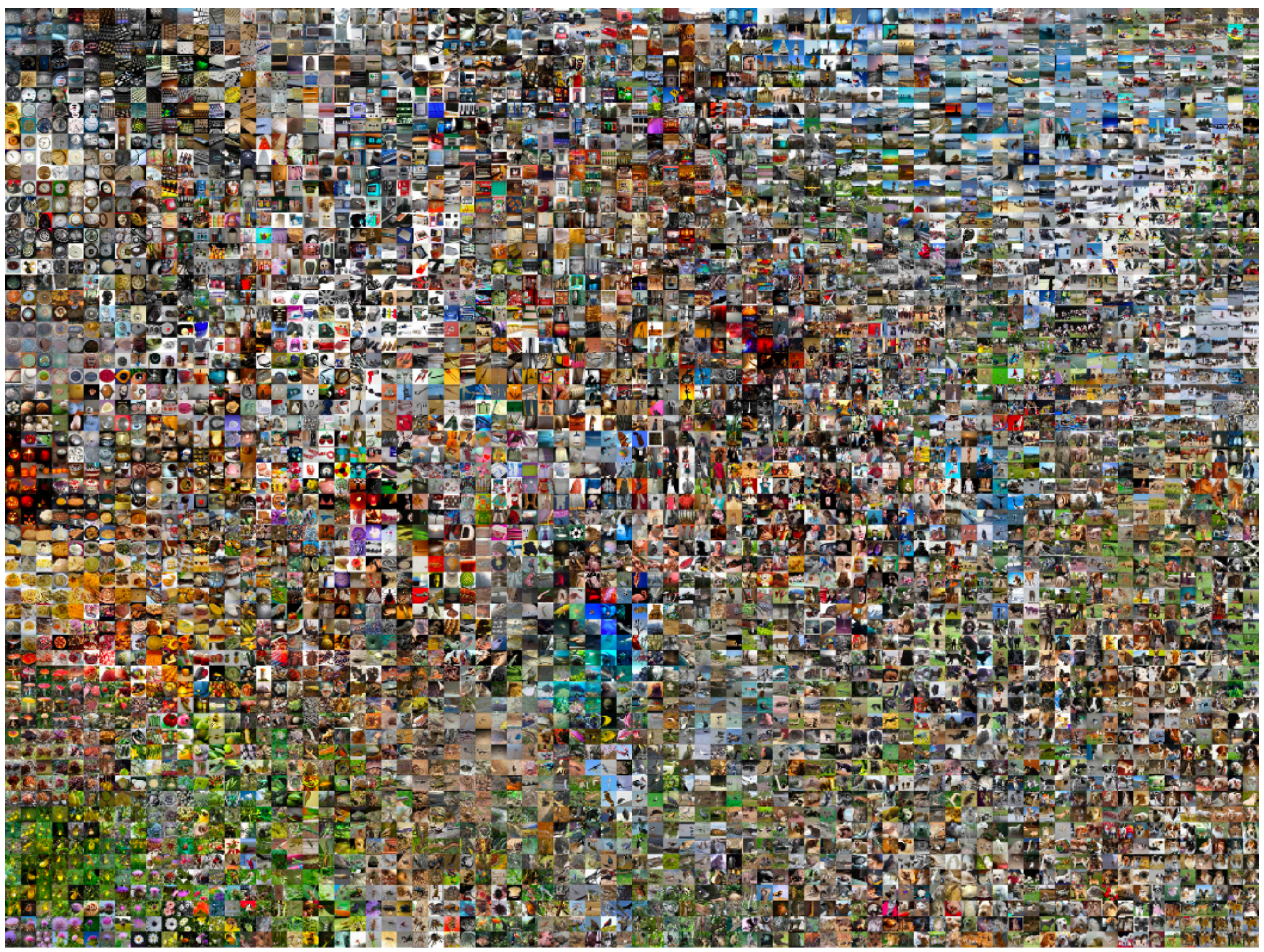
Jet constituents

Event-level kinematics

Pile-up Interactions

Generative AI

Have: input examples (collision events, detector readouts, …)

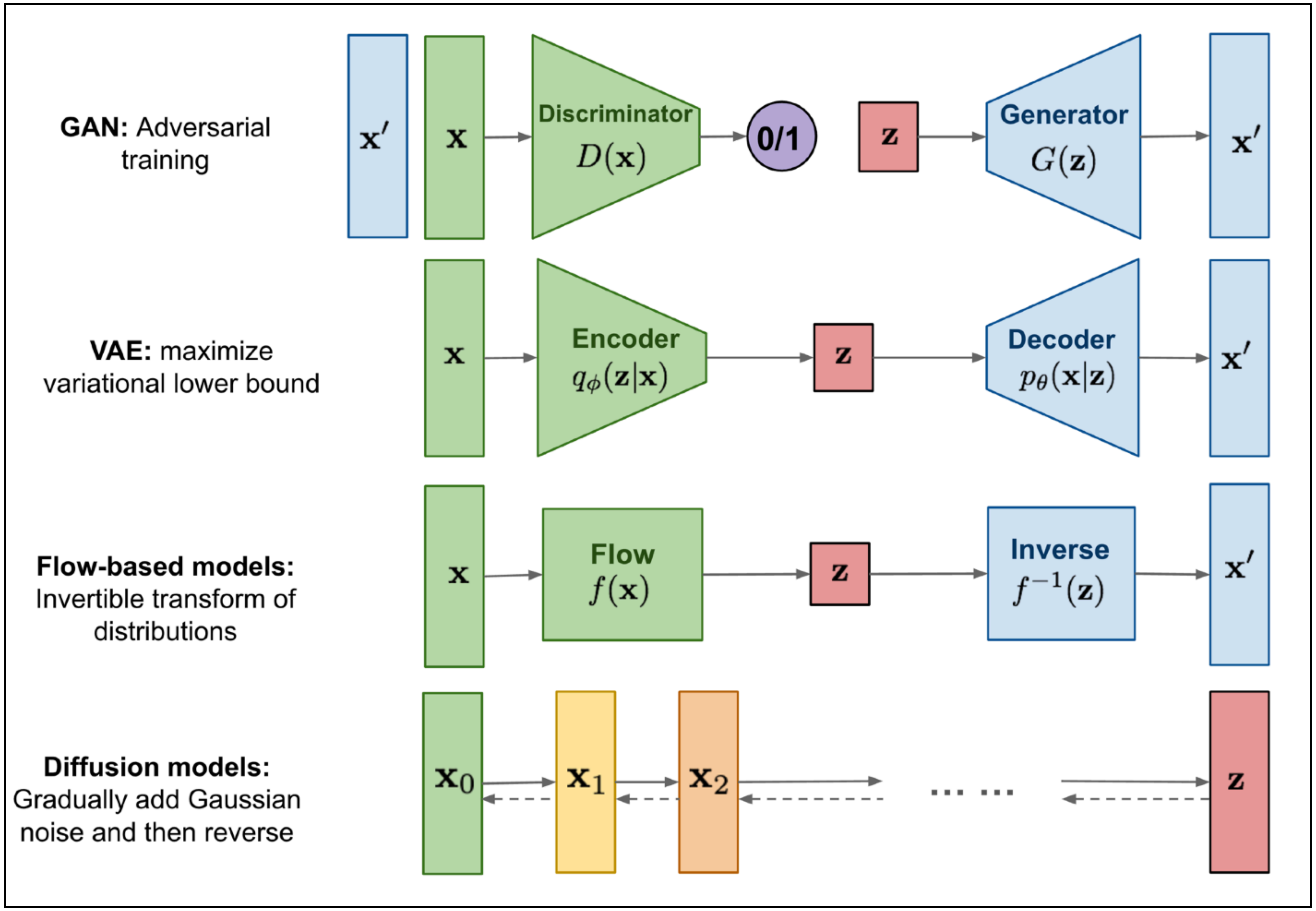


Want: more data

Specifically: new data similar to the input, but not exact copies

How to encode in neural net?

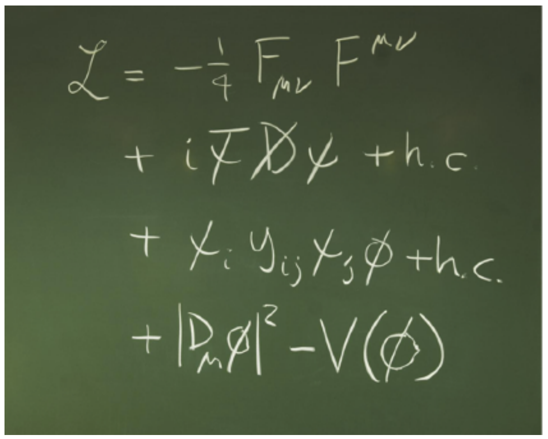
Generative Models

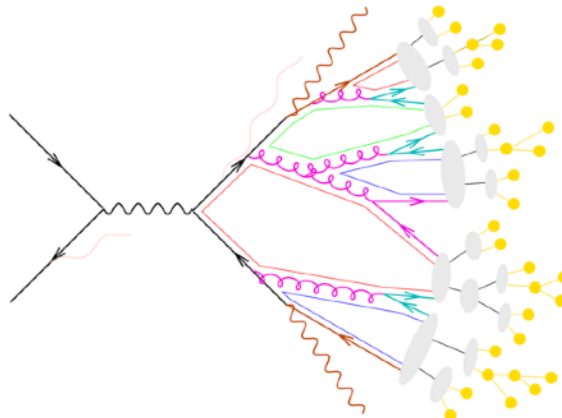


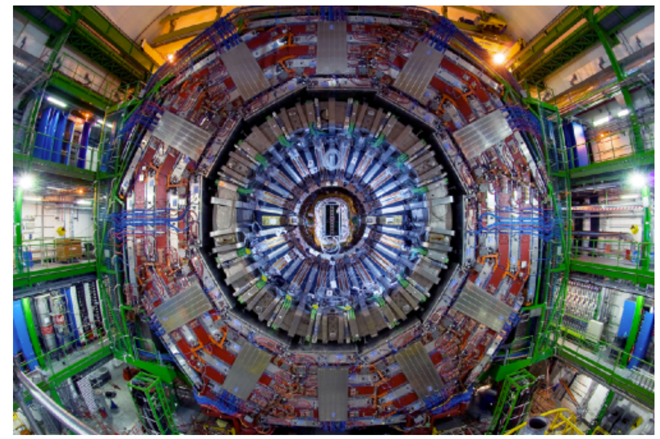
Overview of generative architectures

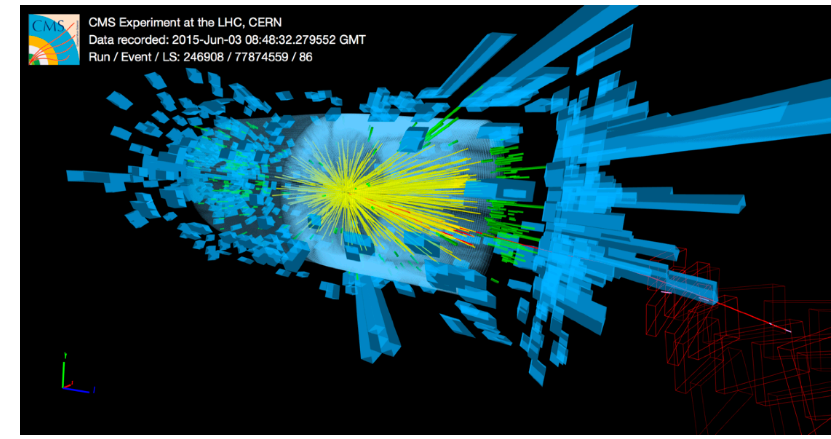
Generative Models

This happens in the experiment







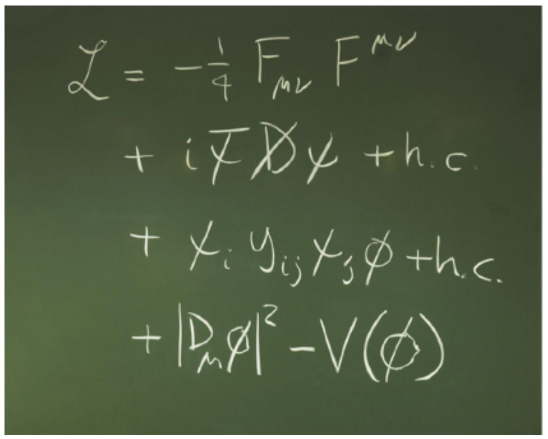


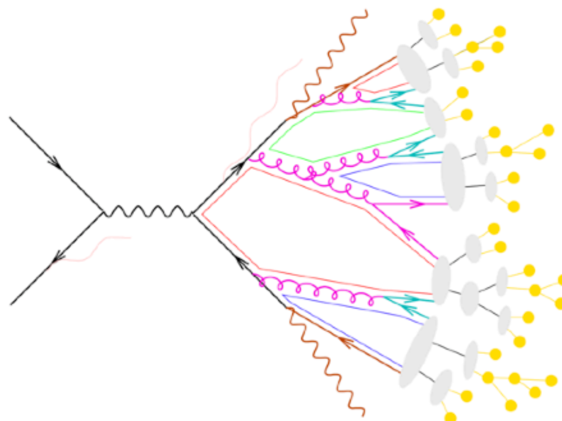
This is what we want to know

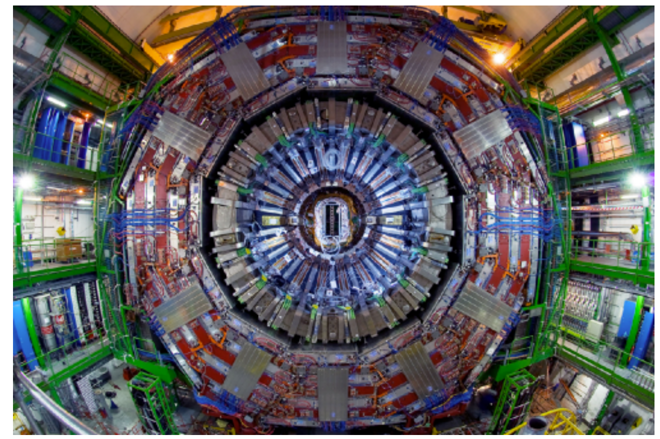
Simulation is crucial to connect experimental data with theory predictions

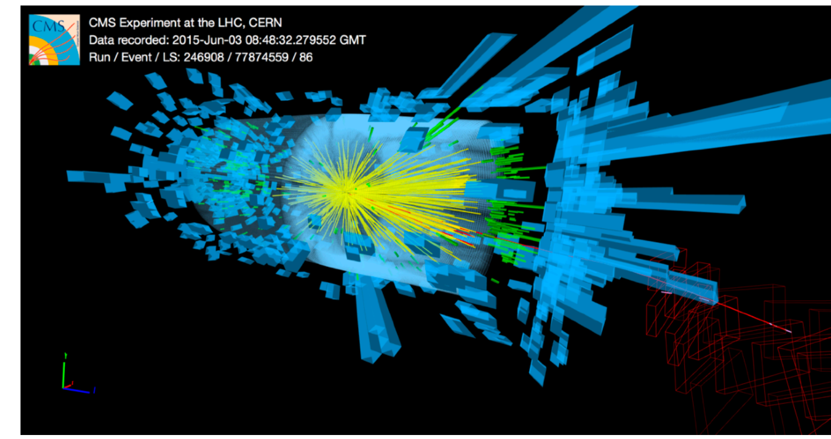
Generative Models

This happens in the experiment



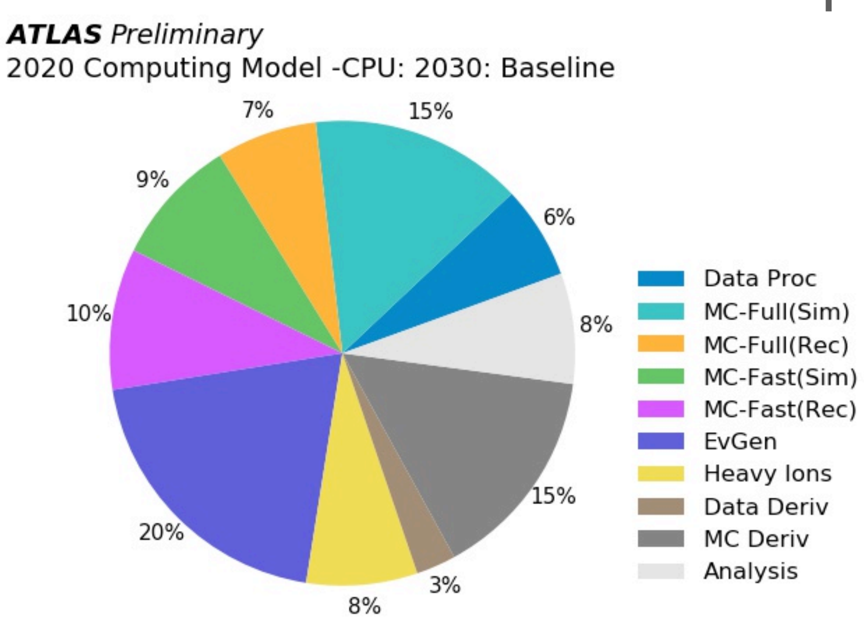


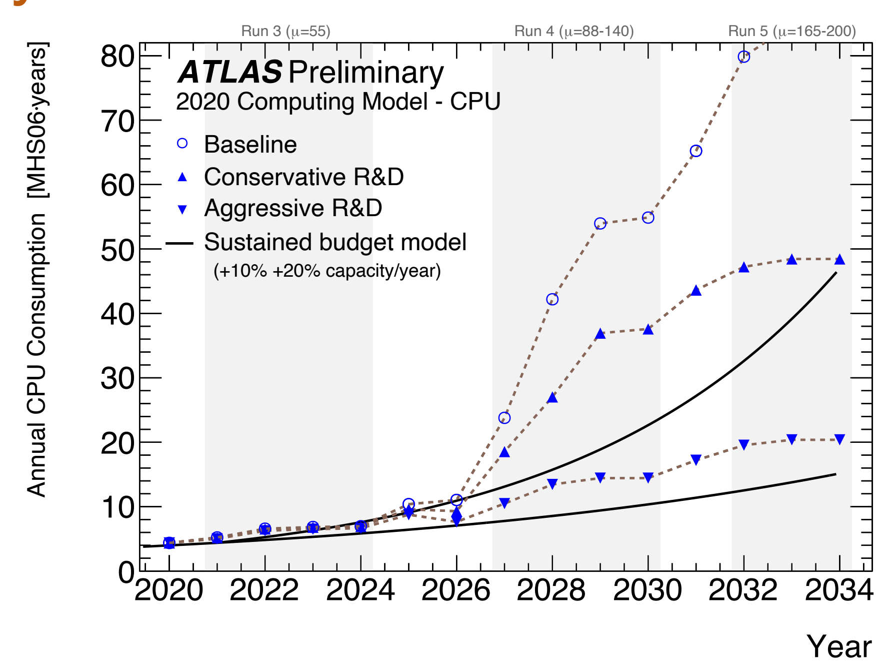




This is what we want to know

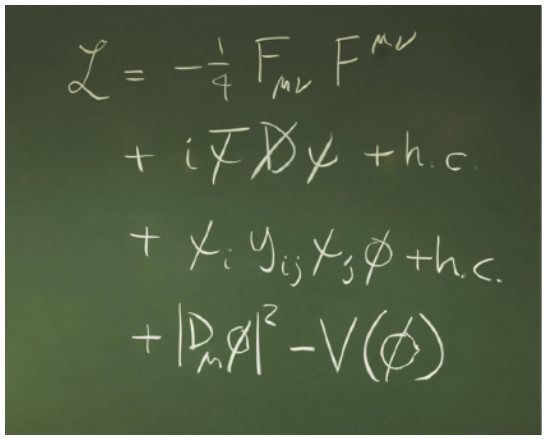
Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

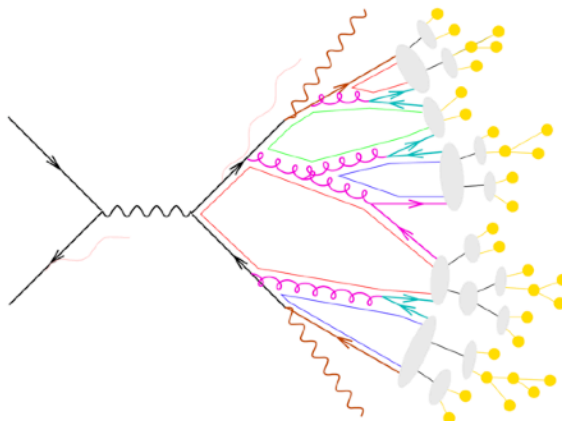
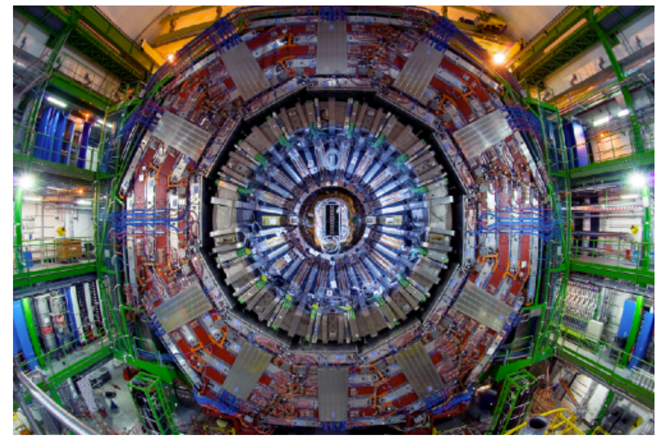
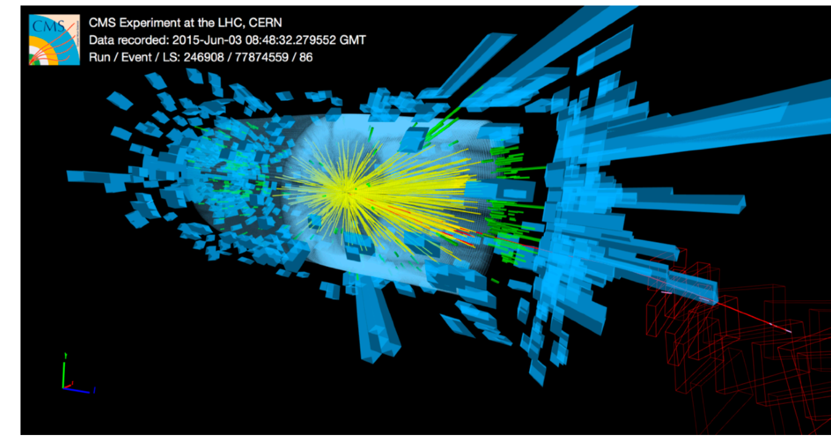




Generative Models

This happens in the experiment



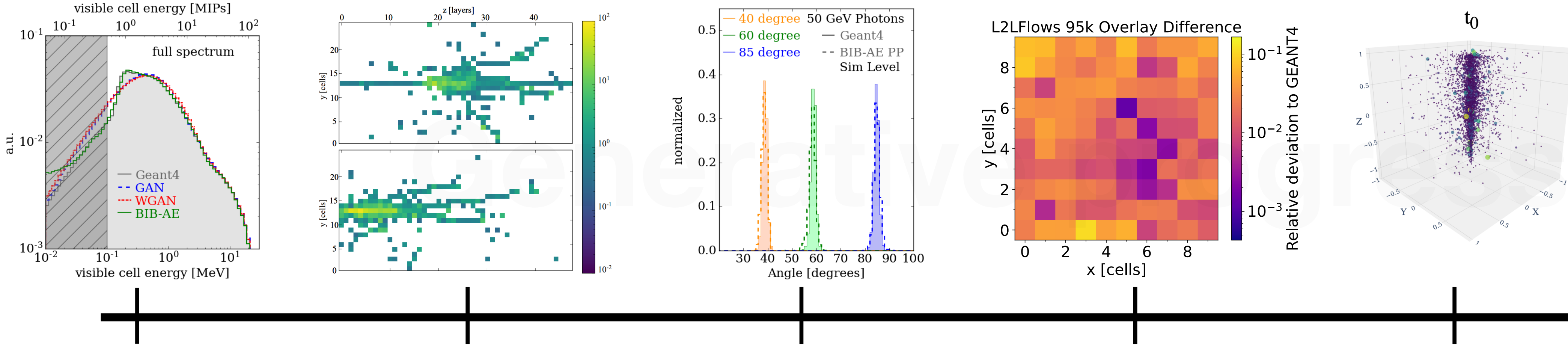




This is what we want to know

Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

→Use generative models trained on simulation or data to augment simulations

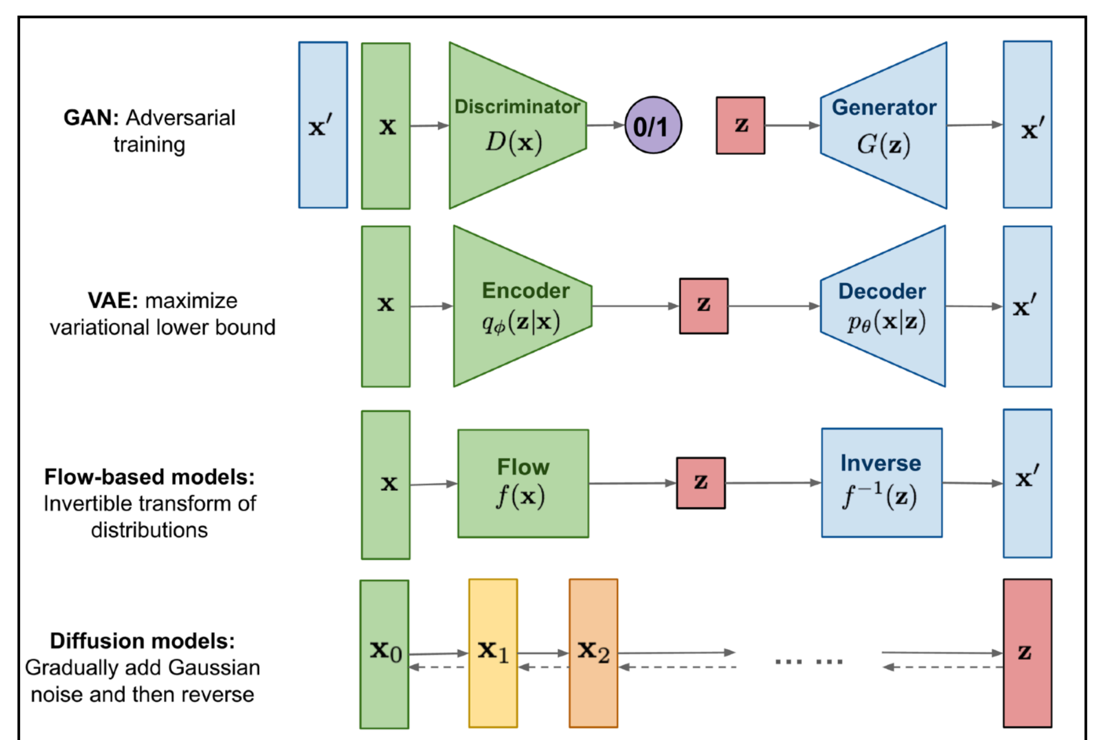
Overview



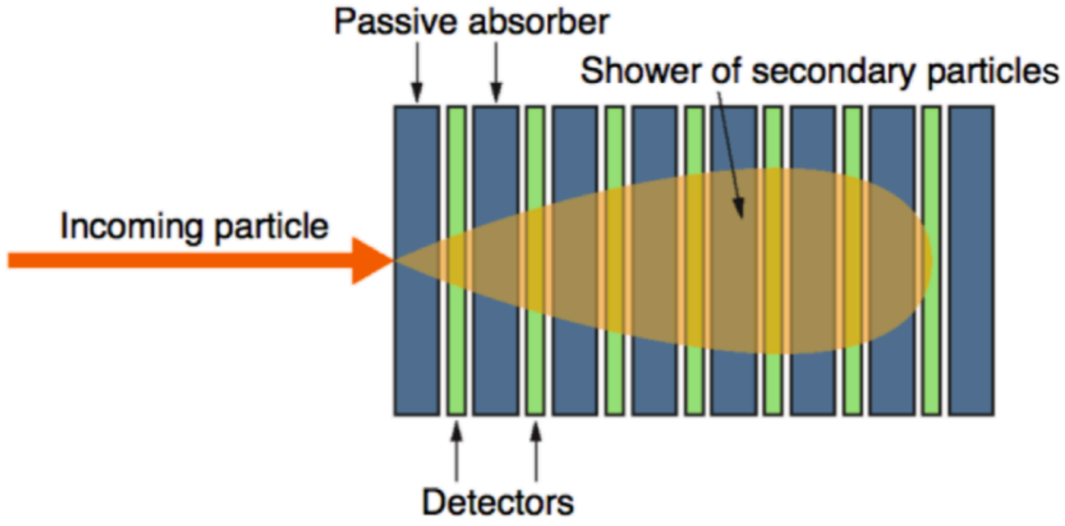
Hardware	Simulator	NFE	Batch Size	Time / Shower [ms]	Speed-up
CPU	Geant4			3914.80 ± 74.09	×1
	CaloClouds	100	1	3146.71 ± 31.66	×1.2
	CaloClouds II	25	1	651.68 ± 4.21	×6.0
	CaloClouds II (CM)	1	1	84.35 ± 0.22	×46
GPU	CaloClouds	100	64	24.91 ± 0.72	×157
	CaloClouds II	25	64	6.12 ± 0.13	×640
	CaloClouds II (CM)	1	64	2.09 ± 0.13	×1873

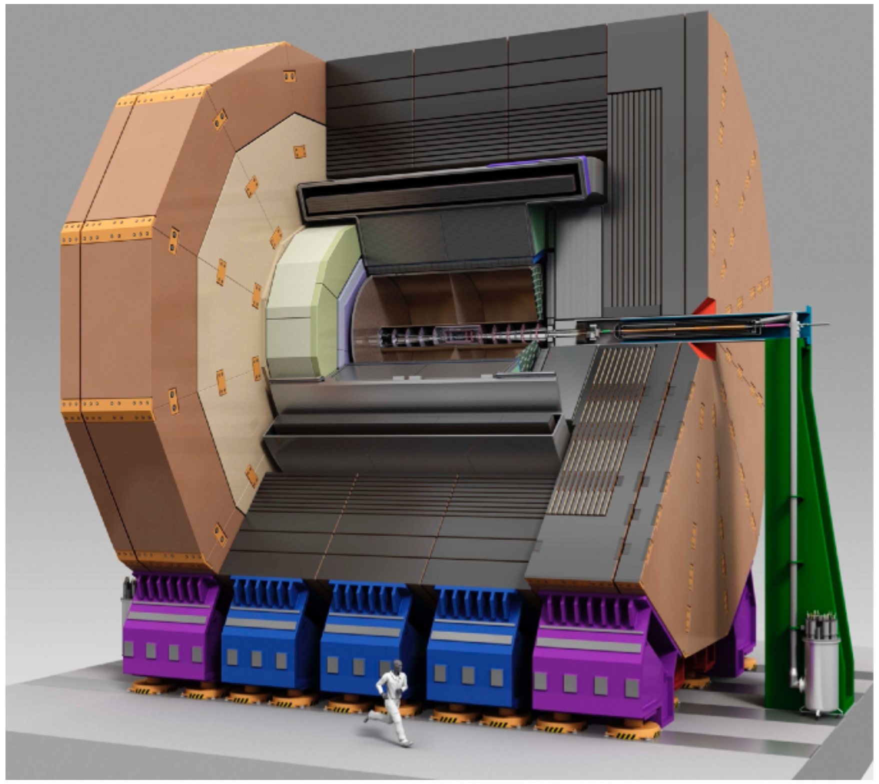
See how better generative architectures improve learning of physics distributions

Introduce new generative architectures as needed

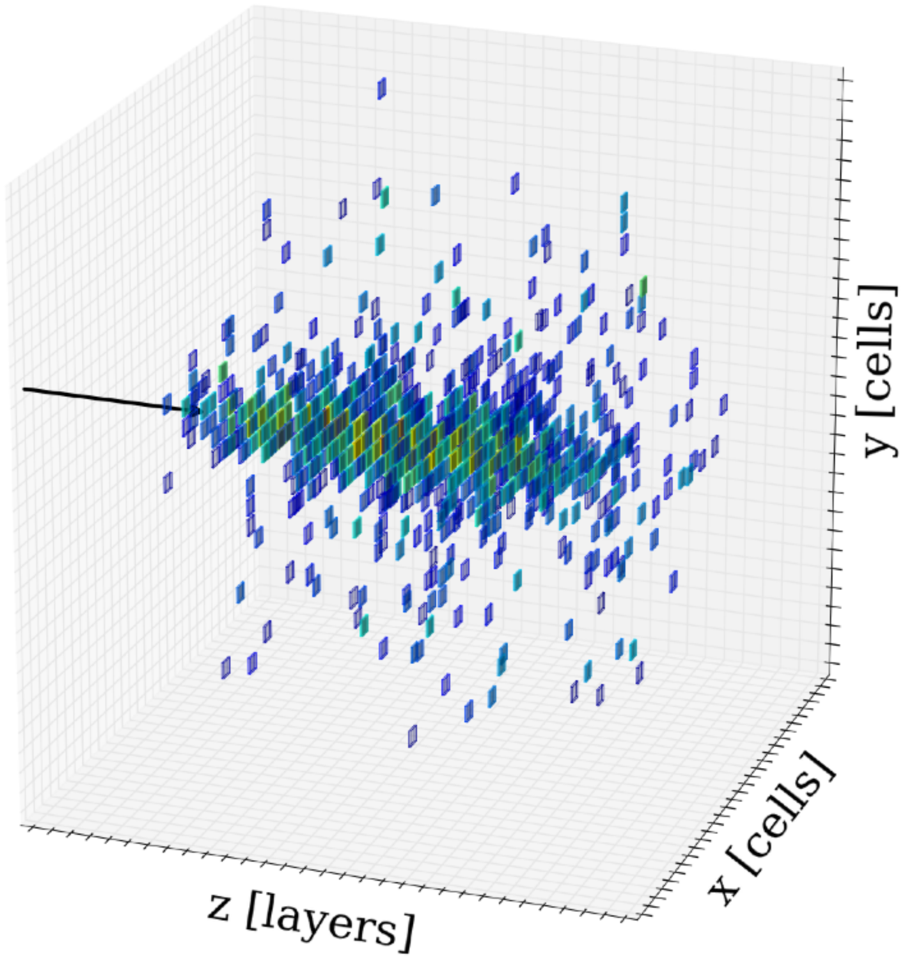


Simulation target



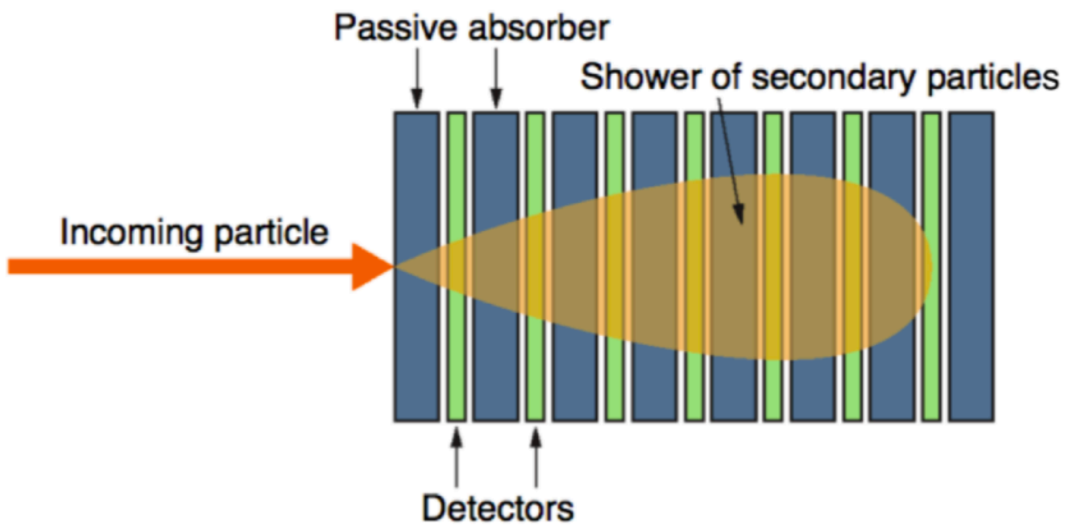


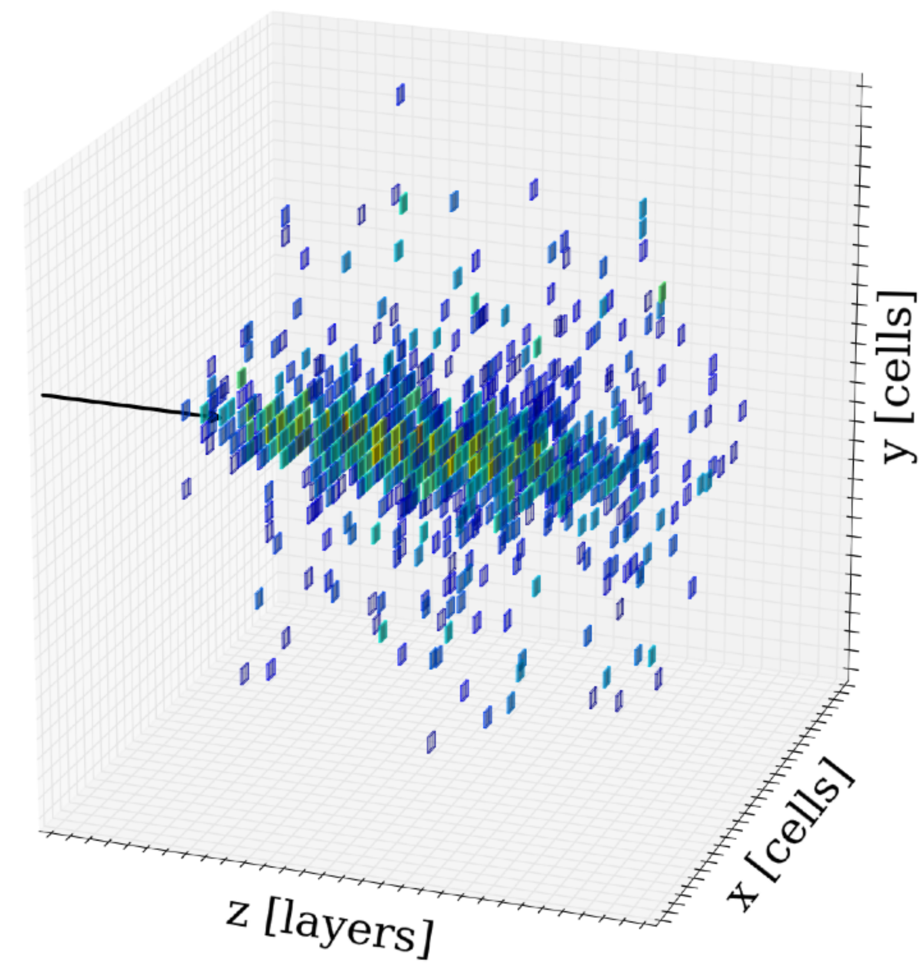
ILD Detector



- Shower in ILD Electromagnetic Calorimeter
- 30x30x30 cells (Si-W)
- Photon energies from 10 to 100 GeV
- Use 950k examples (uniform in energy) created with GEANT4 to train

Simulation target

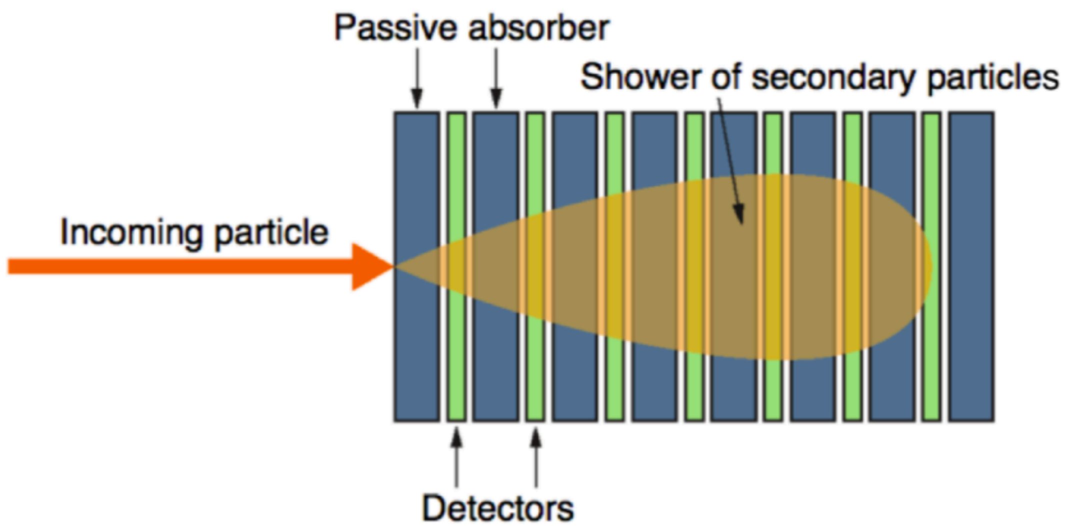


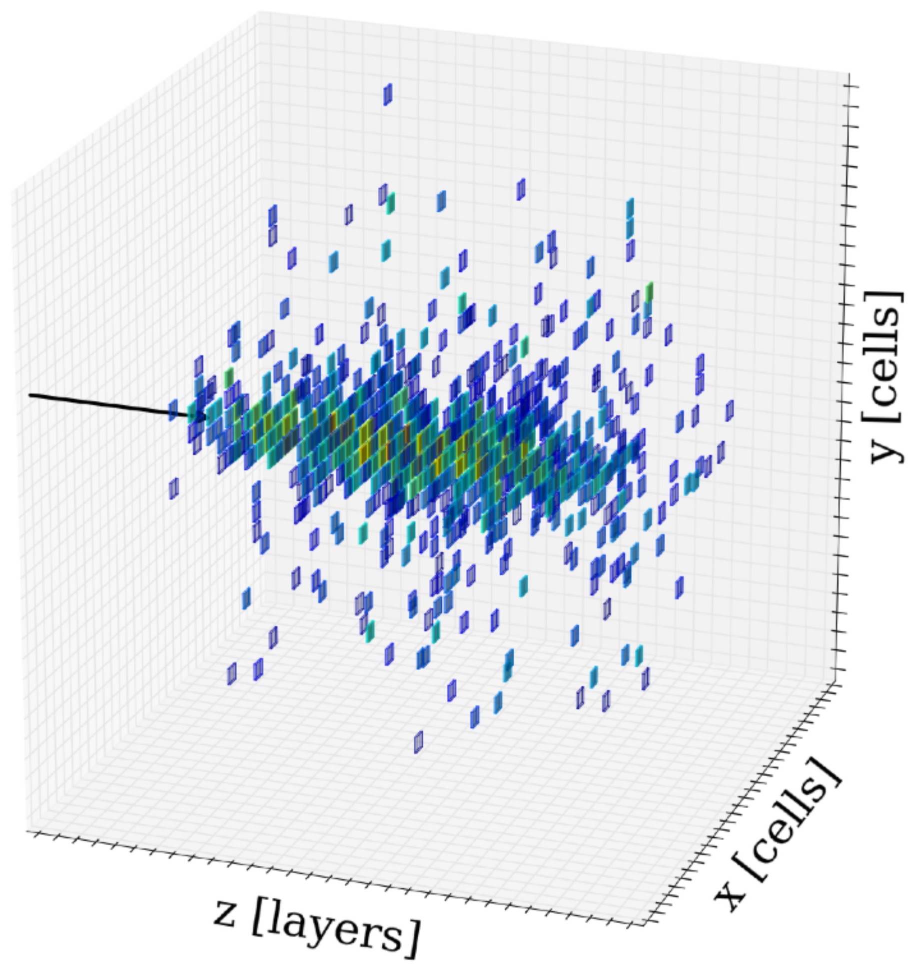


How to represent?

Tabular data:
Easy, insufficient for high-dimensions

Simulation target



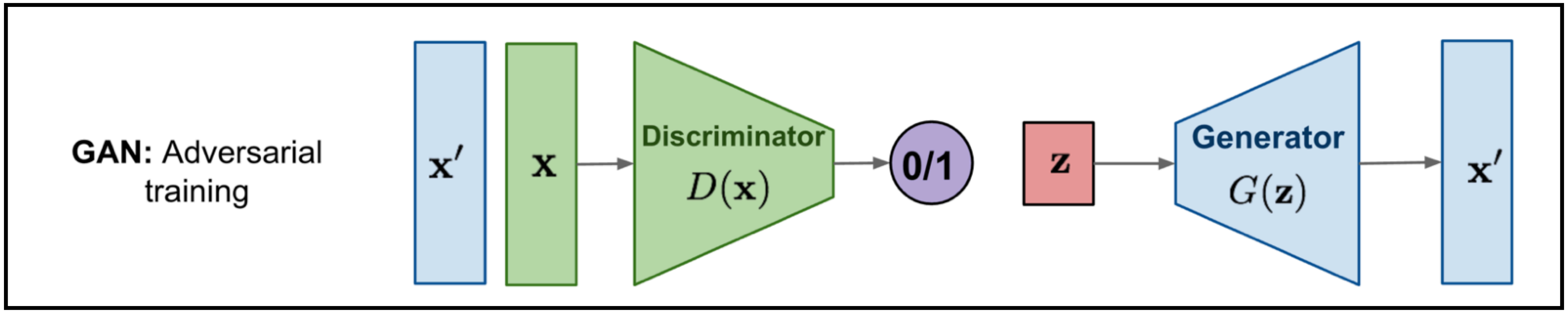


How to represent?

Tabular data

Fixed grid: Voxel image
(allows using e.g. convolutional networks)

Generative Adversarial Networks



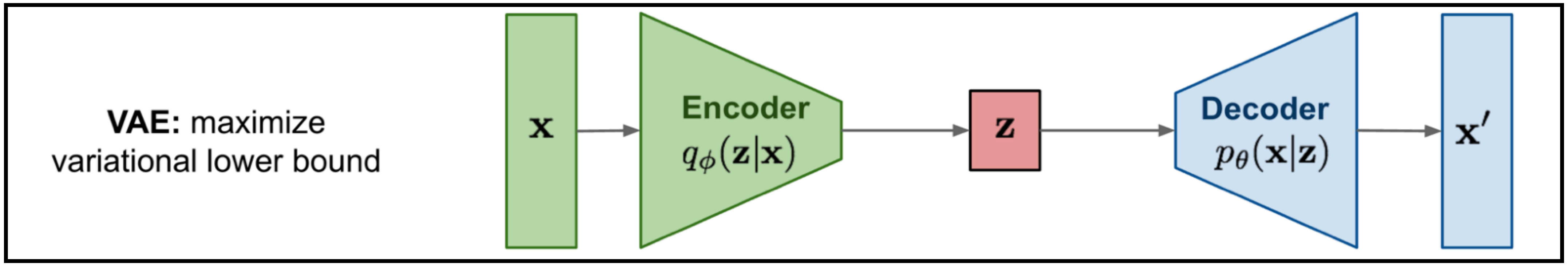
Training objective:
Binary cross entropy

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

At (Nash) equilibrium:
Generator produces realistic examples
Discriminator is maximally confused

1406.2661

Variational Autoencoder



Variational Autoencoder (VAE):
Split latent space
Sample before decoder
Penalty so mean/std are close to unit Gaussian

$$f(x) = (\mu, \sigma)$$

$$z = \text{Gaussian}(\mu, \sigma)$$

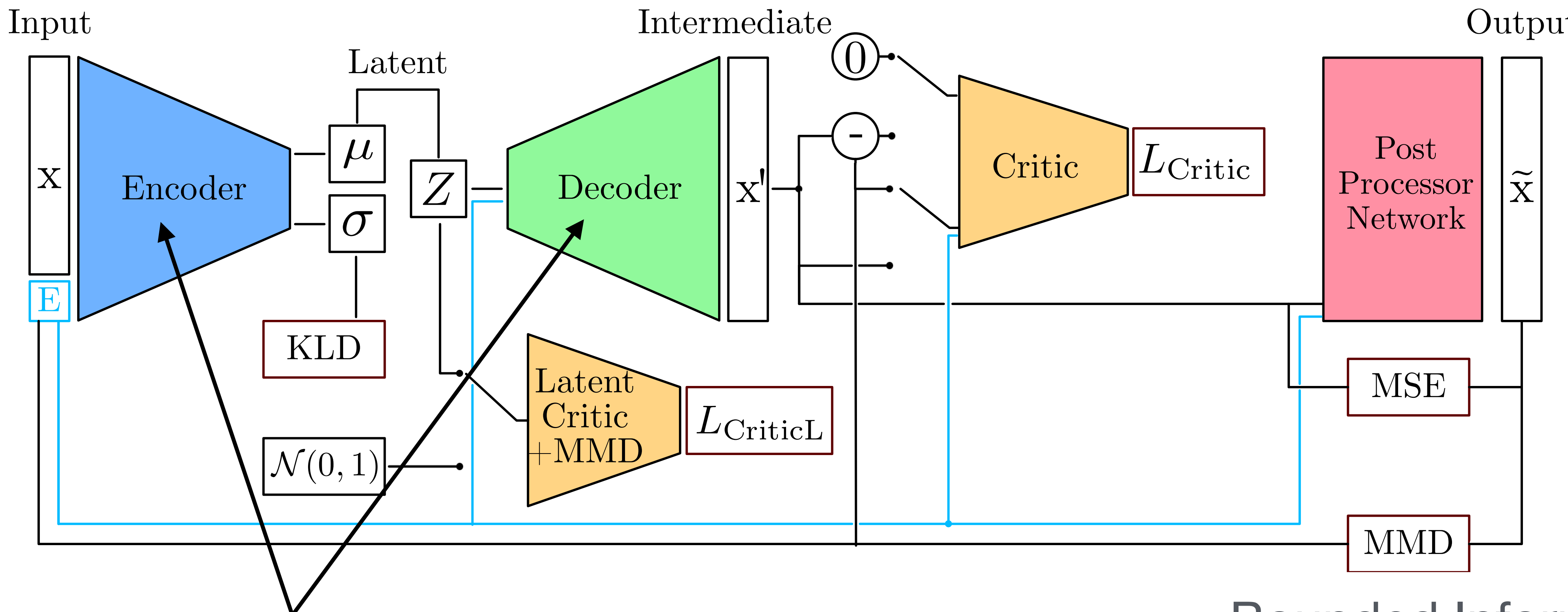
$$x' = g(z)$$

$$L = (x - g(z))^2 + \sigma^2 + \mu^2 - \log(\sigma) - 1$$

(Calculate KL-divergence
between Gaussians)

1312.6114

Generative Architecture



(Transposed) Convolution

Bounded Information Bottleneck AE

BIB-AE (GAN + VAE)

$$
\begin{aligned}
L_{\text{BIB-AE}} = & -\beta_{C_L} \cdot \mathbb{E}[C_L(N_E(x))] \\
& -\beta_C \cdot \mathbb{E}[C_E(D_E(N_E(x)))] \\
& -\beta_{C_D} \cdot \mathbb{E}[C_{D,E}(D_E(N_E(x)) - x)] \\
& +\beta_{\text{KLD}} \cdot \text{KLD}(N_E(x)) \\
& +\beta_{\text{MMD}} \cdot \text{MMD}(N_E(x), \mathcal{N}(0,1)).
\end{aligned}
$$

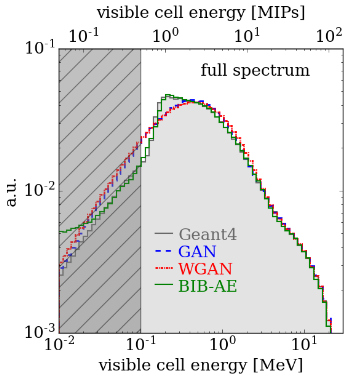
Latent Critic

Critic

Difference Critic

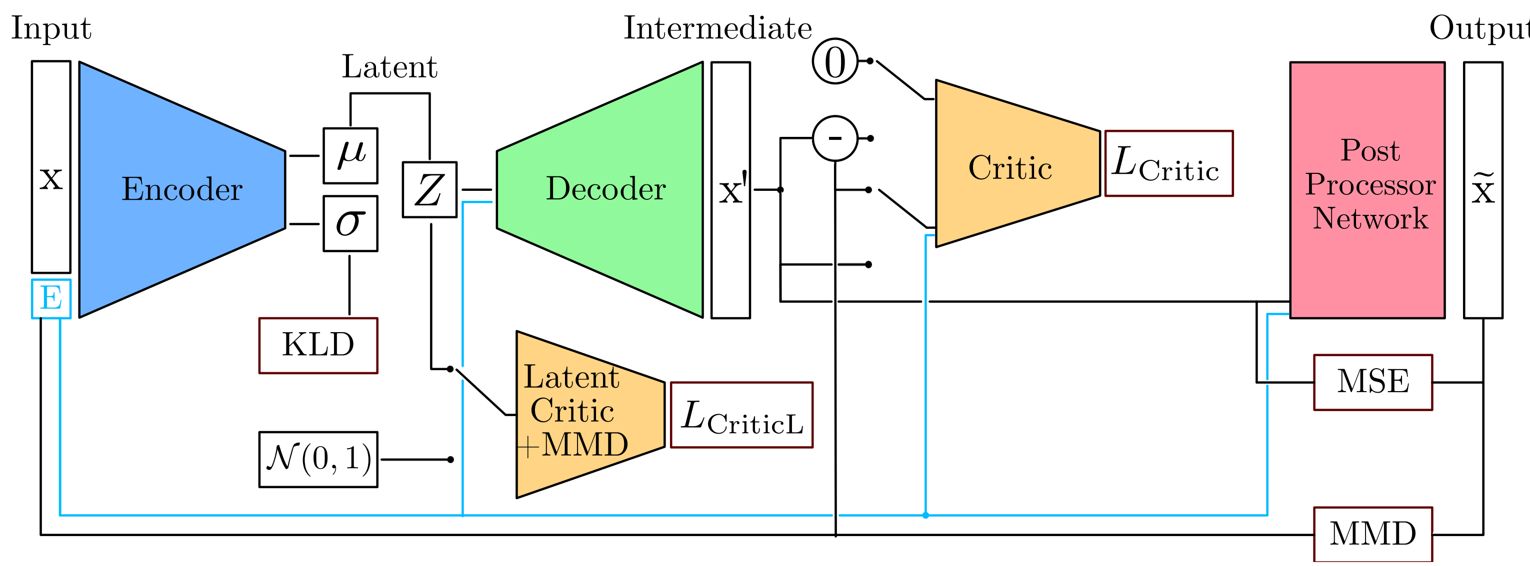
Latent Regularisation

1912.00830, 2005.05334



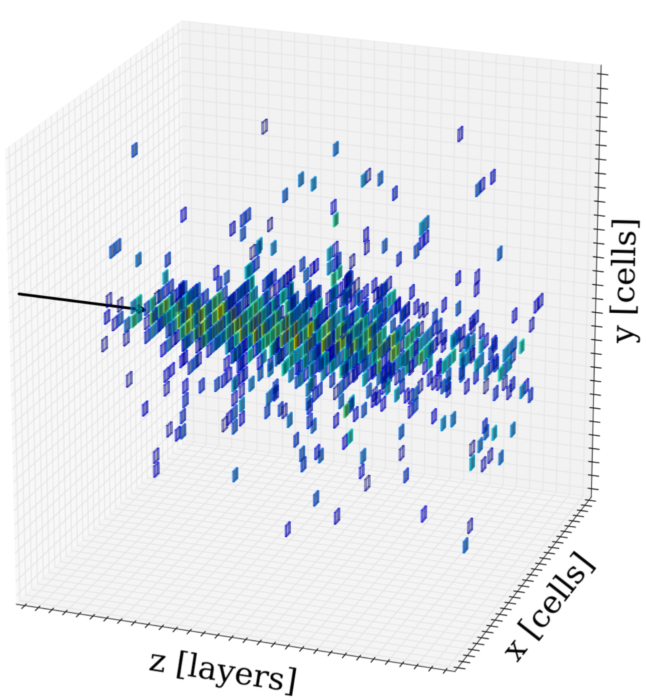
Generative progress

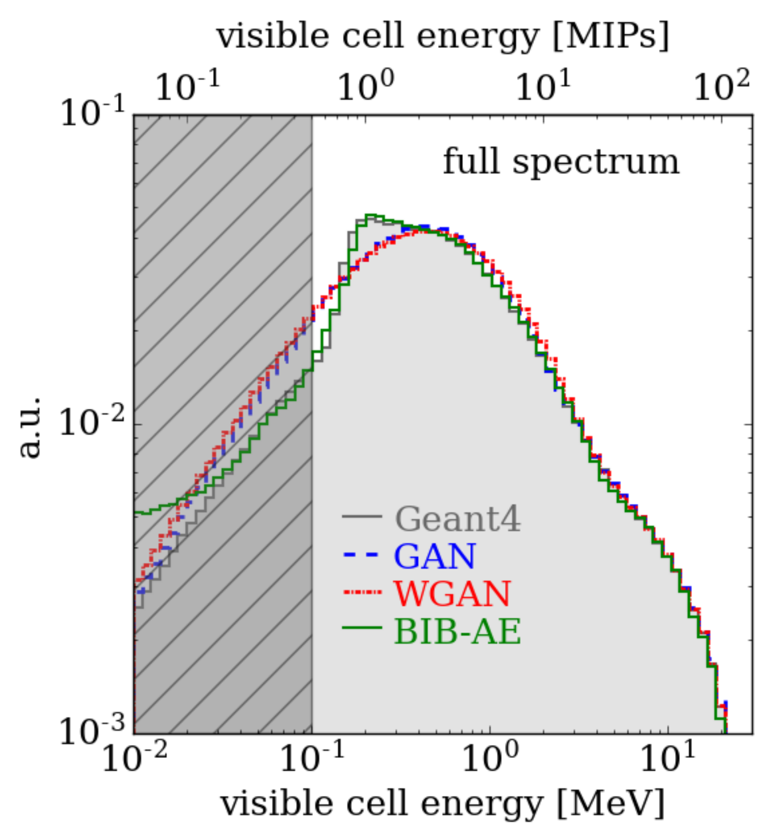
Progress



BIB-AE (GAN + VAE):

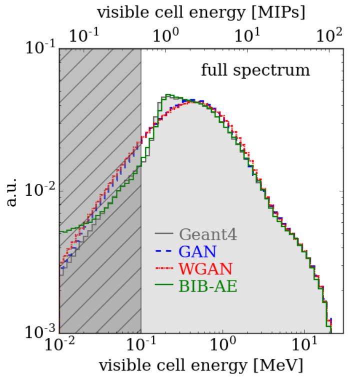
1st simulation of Photon shower in 27k cell calorimeter

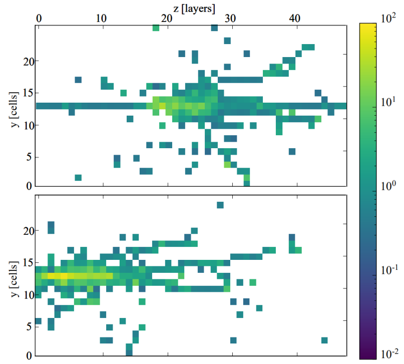




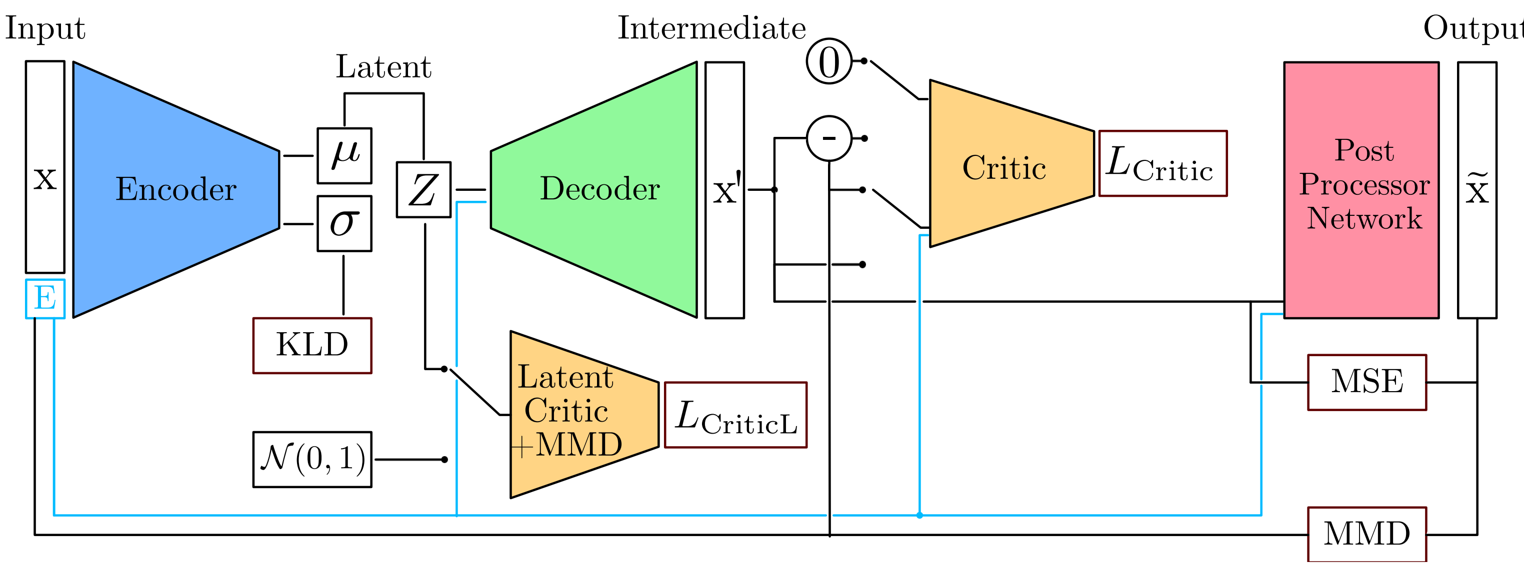
Buhmann, .., **GK** et al 2005.05334

erative progress

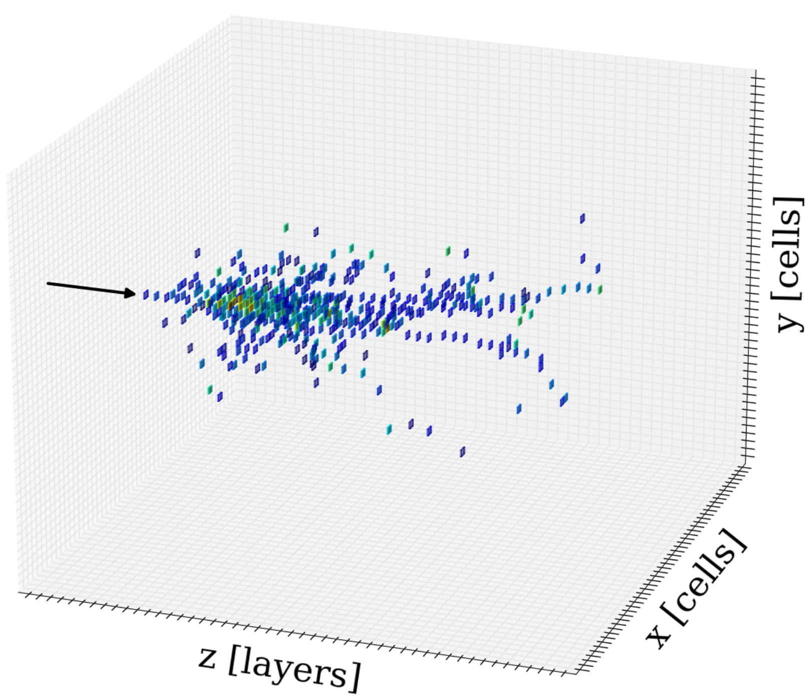


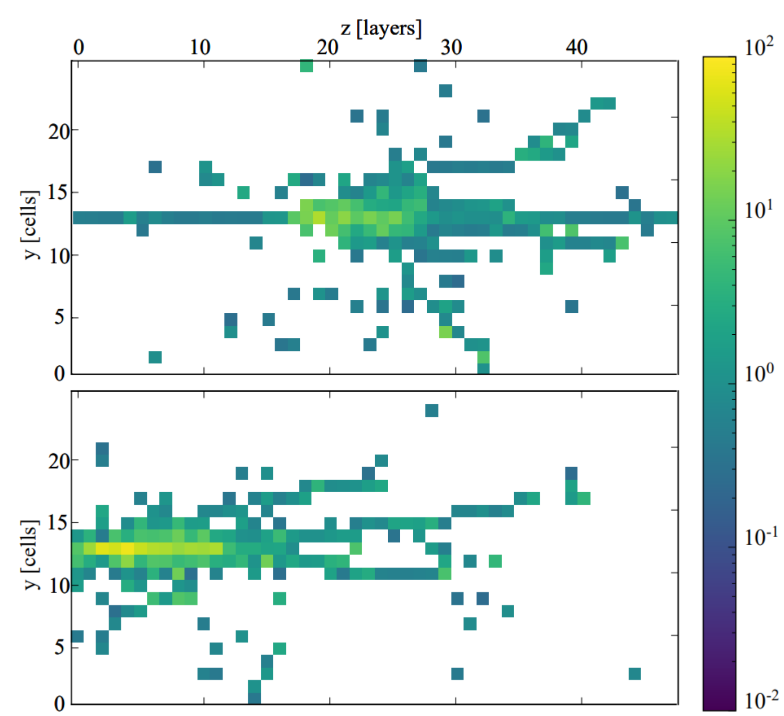


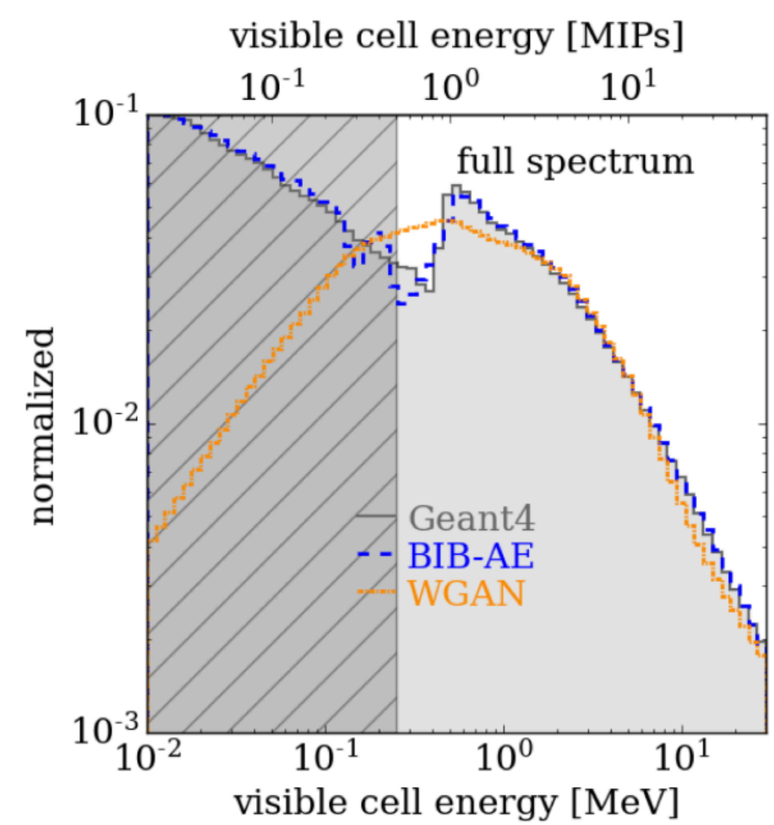
Progress



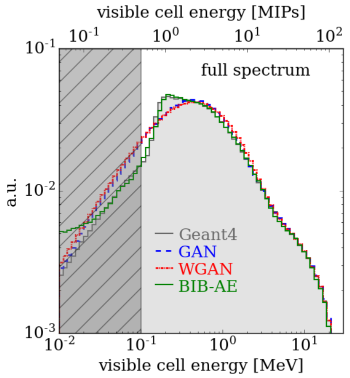
Handle more complex pion showers in hadronic calorimeter

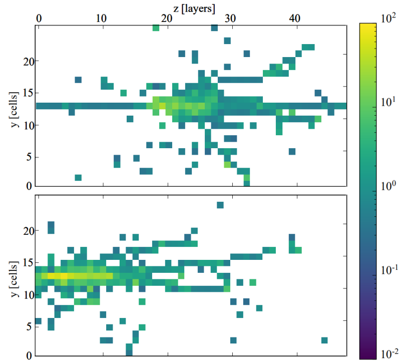


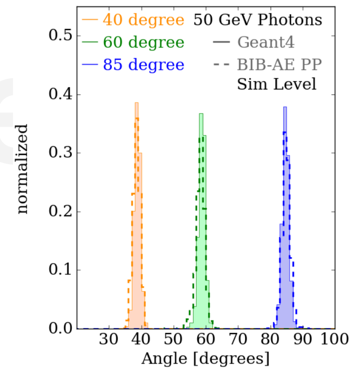




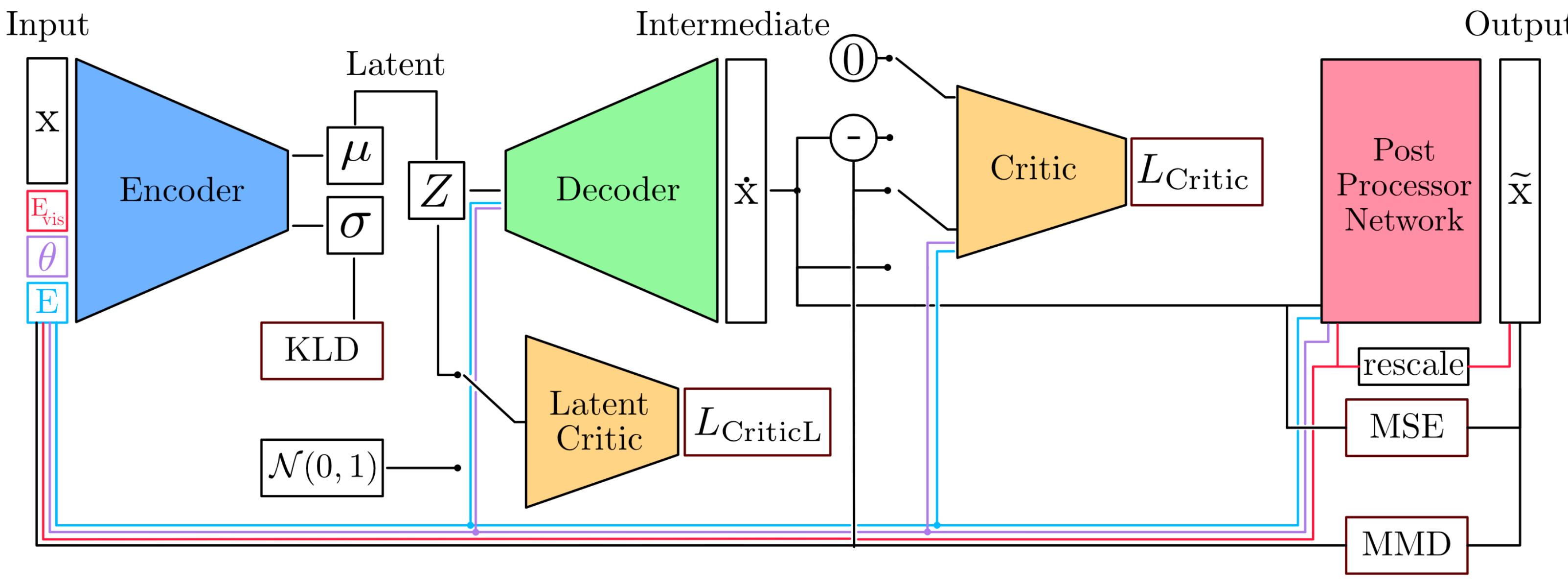
Buhmann, .., **GK** et al 2112.09709;

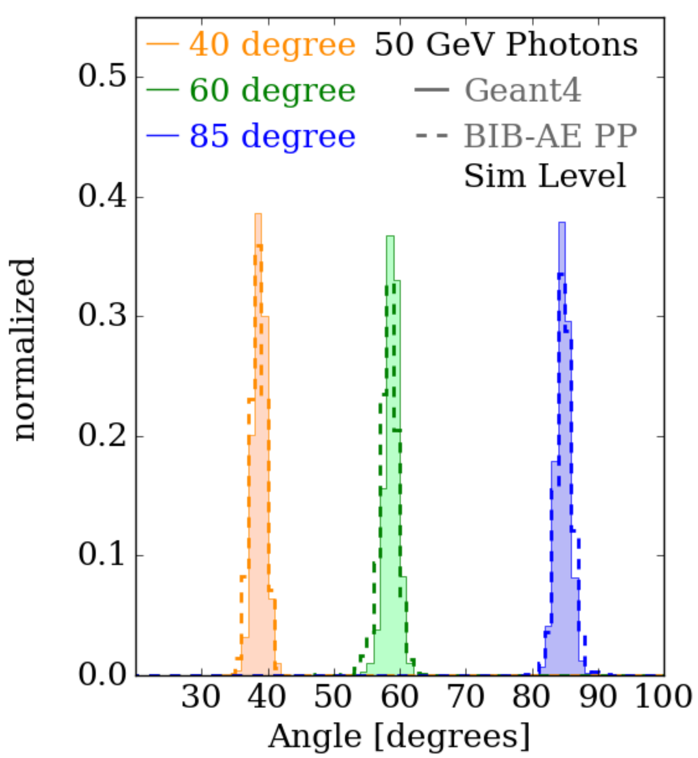


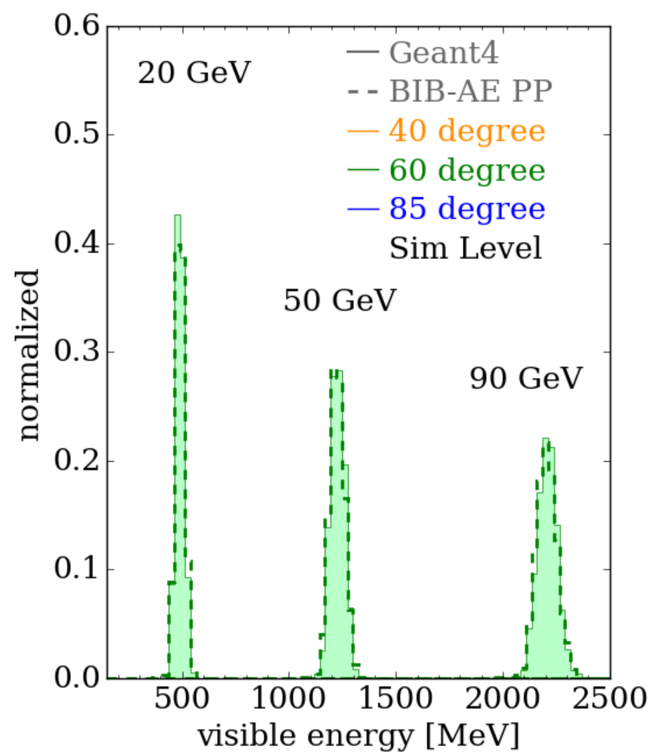




Progress



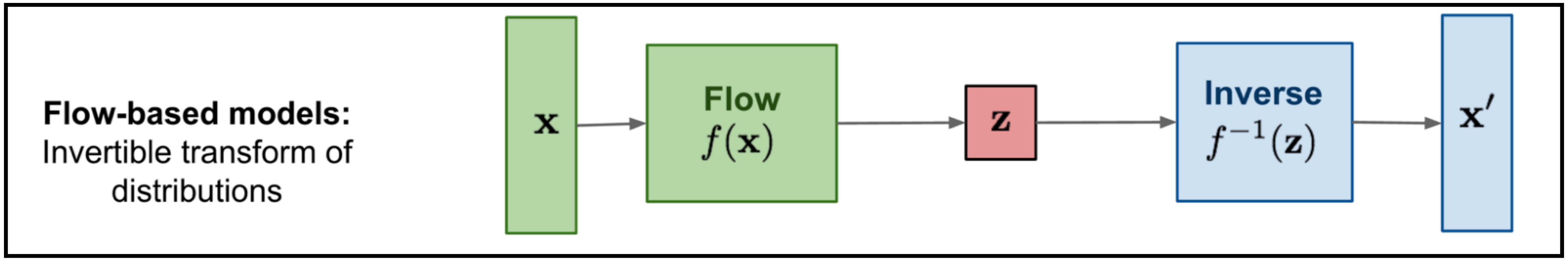




Extend to condition on angles

Diefenbacher, .., **GK** et al 2303.18150

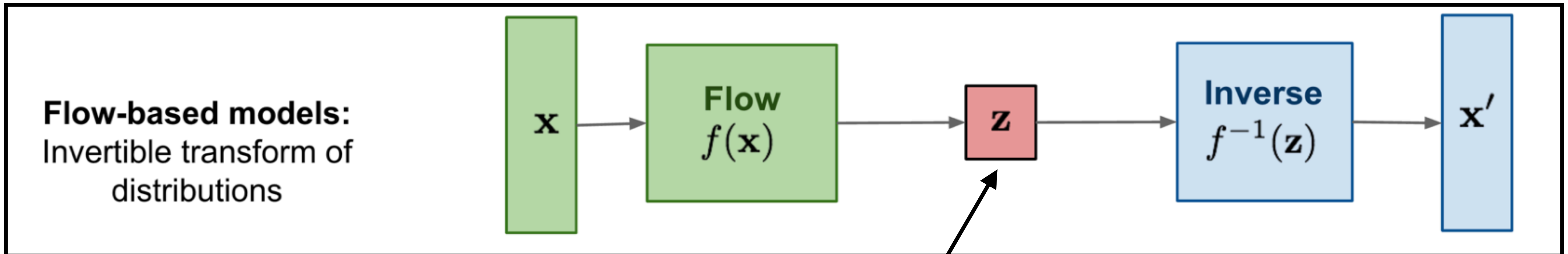
Generative models



In auto-encoders, the decoder learns to 'undo' the encoder

Can we make this exact and directly learn the likelihood?

Generative models

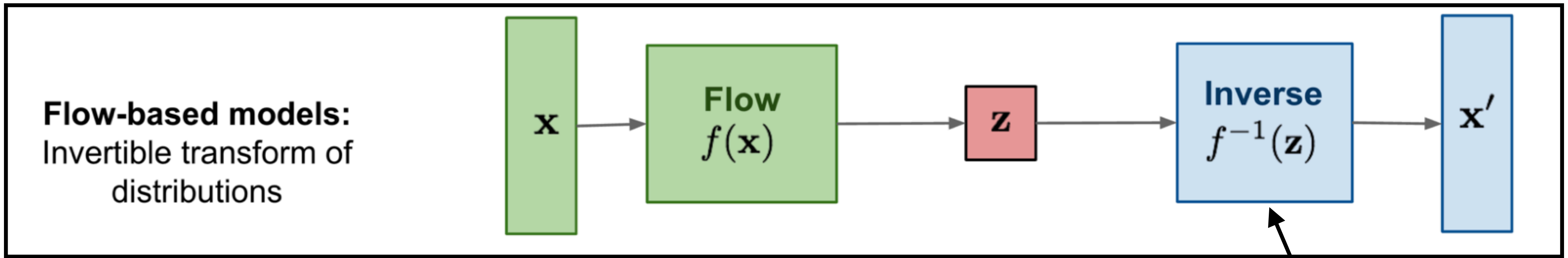


Choose latent space, e.g. standard normal distribution (normalising flow!)
Same dimension as data!

Learn a diffeomorphism between data and latent-space

1505.05770, 1908.09257

Generative models

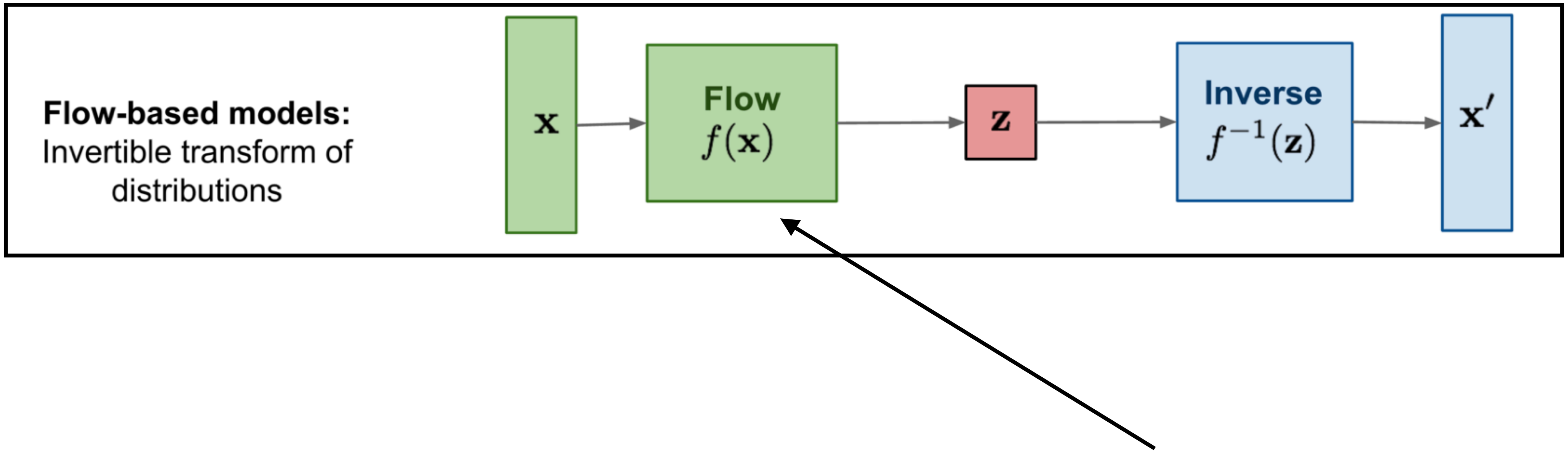


f^{-1} is not a learned inversion, but exact inverse by construction

Learn a diffeomorphism between data and latent-space

Bijective, invertable

Generative models



Take into account Jacobian determinant to evaluate probability density

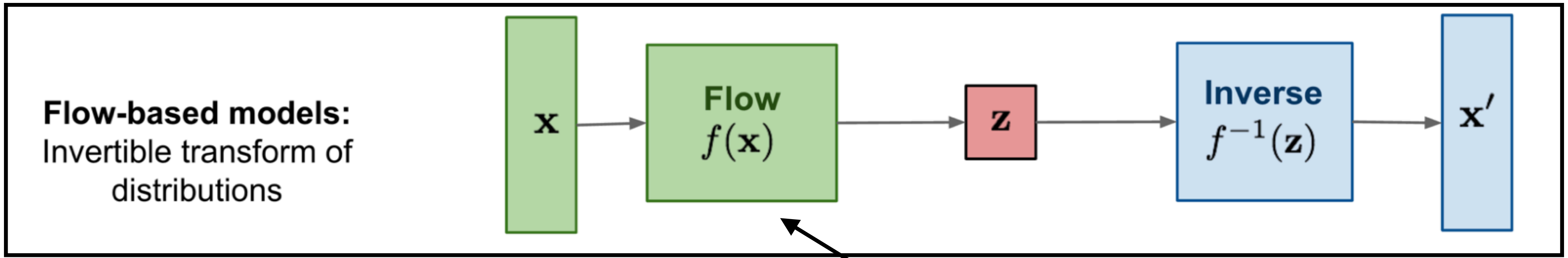
Learn a diffeomorphism between data and latent-space

Bijective, invertable

Learn likelihood of data

1505.05770, 1908.09257

Generative models

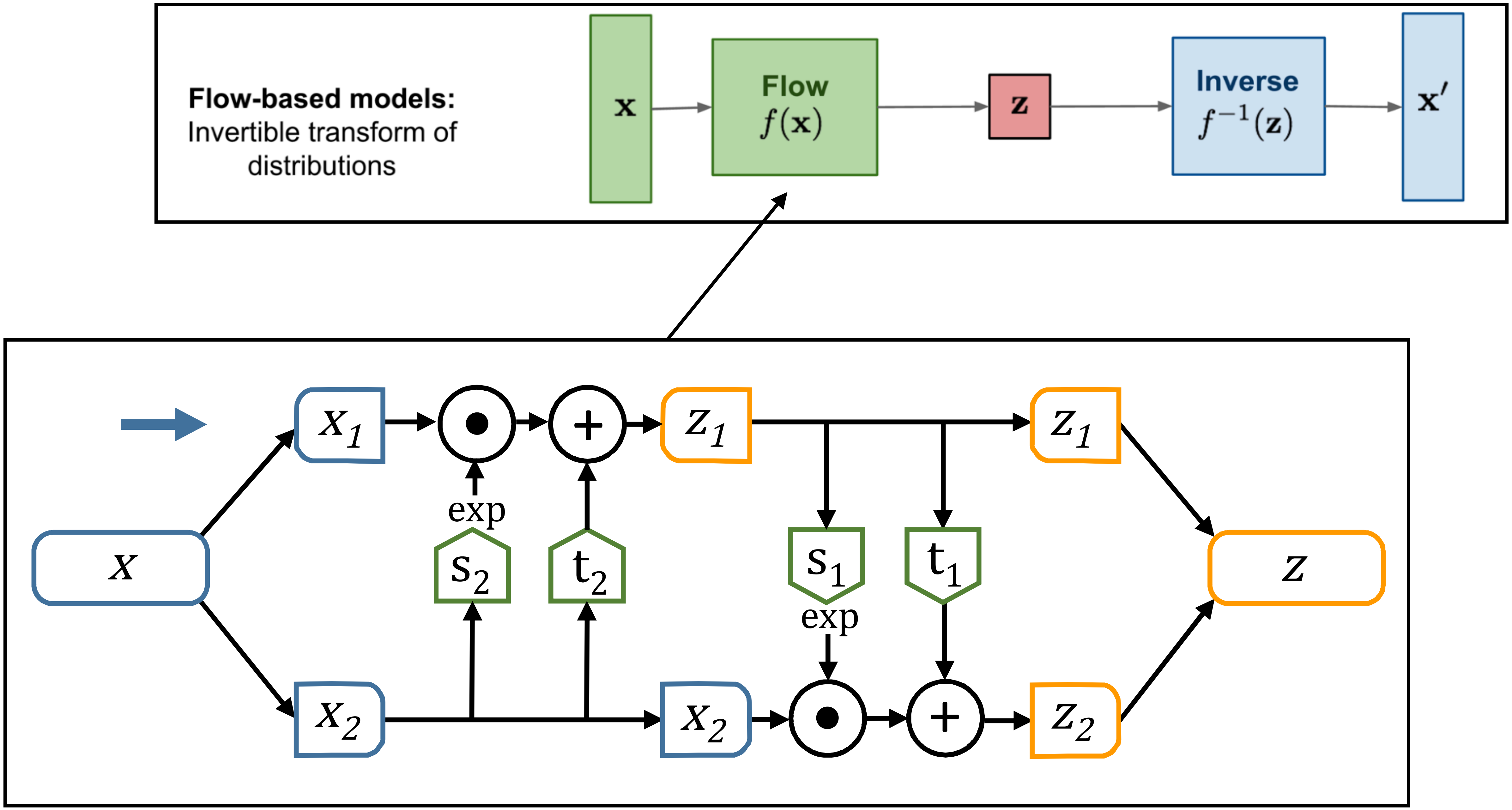


Take into account Jacobian determinant to evaluate probability density

2 challenges:
- Invertible
- Easy-to-calculate Jacobean

1505.05770, 1908.09257

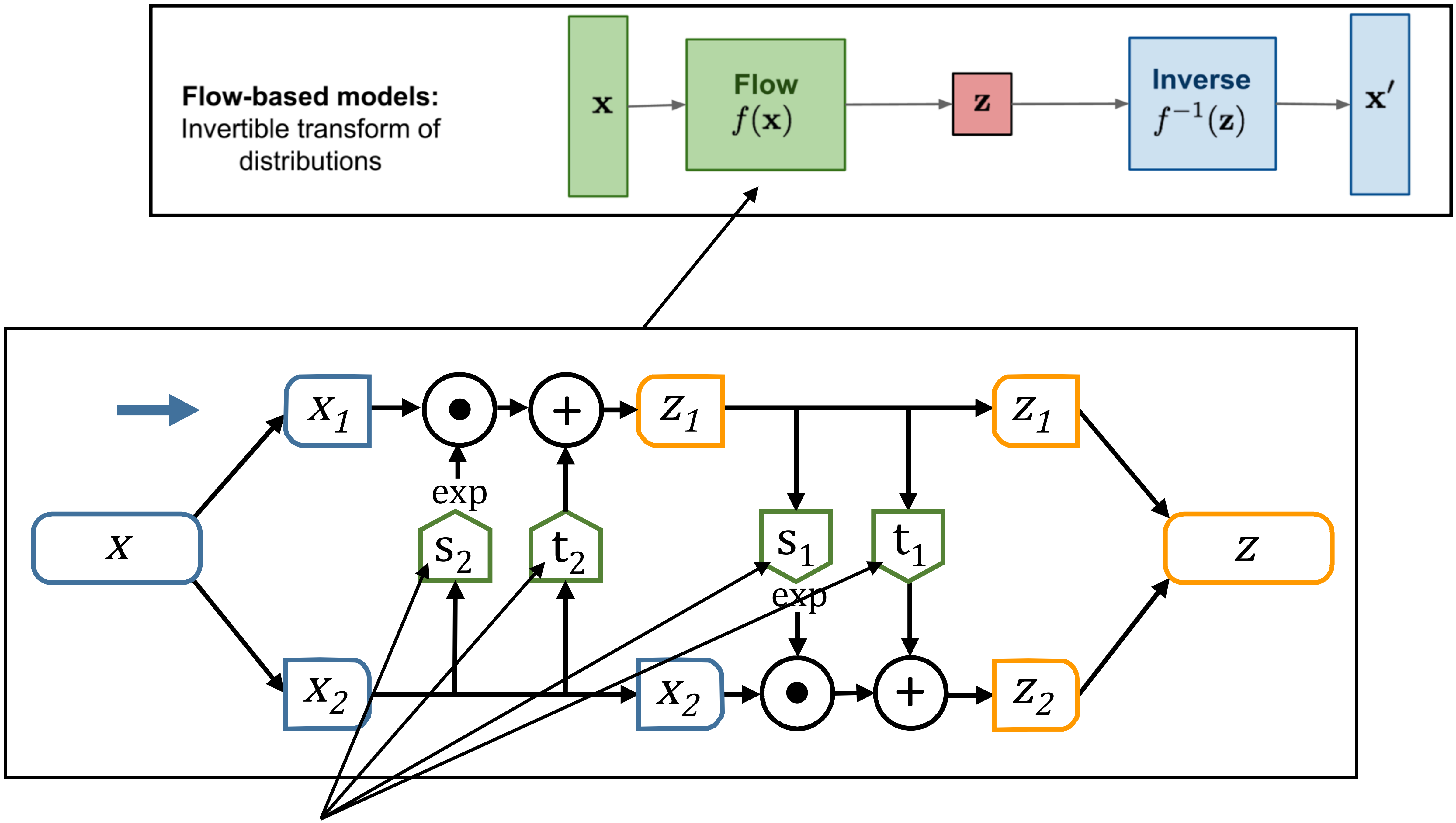
Coupling flows



Coupling layers: Not the most expressive, but useful for illustration/understanding

1505.05770, 1908.09257

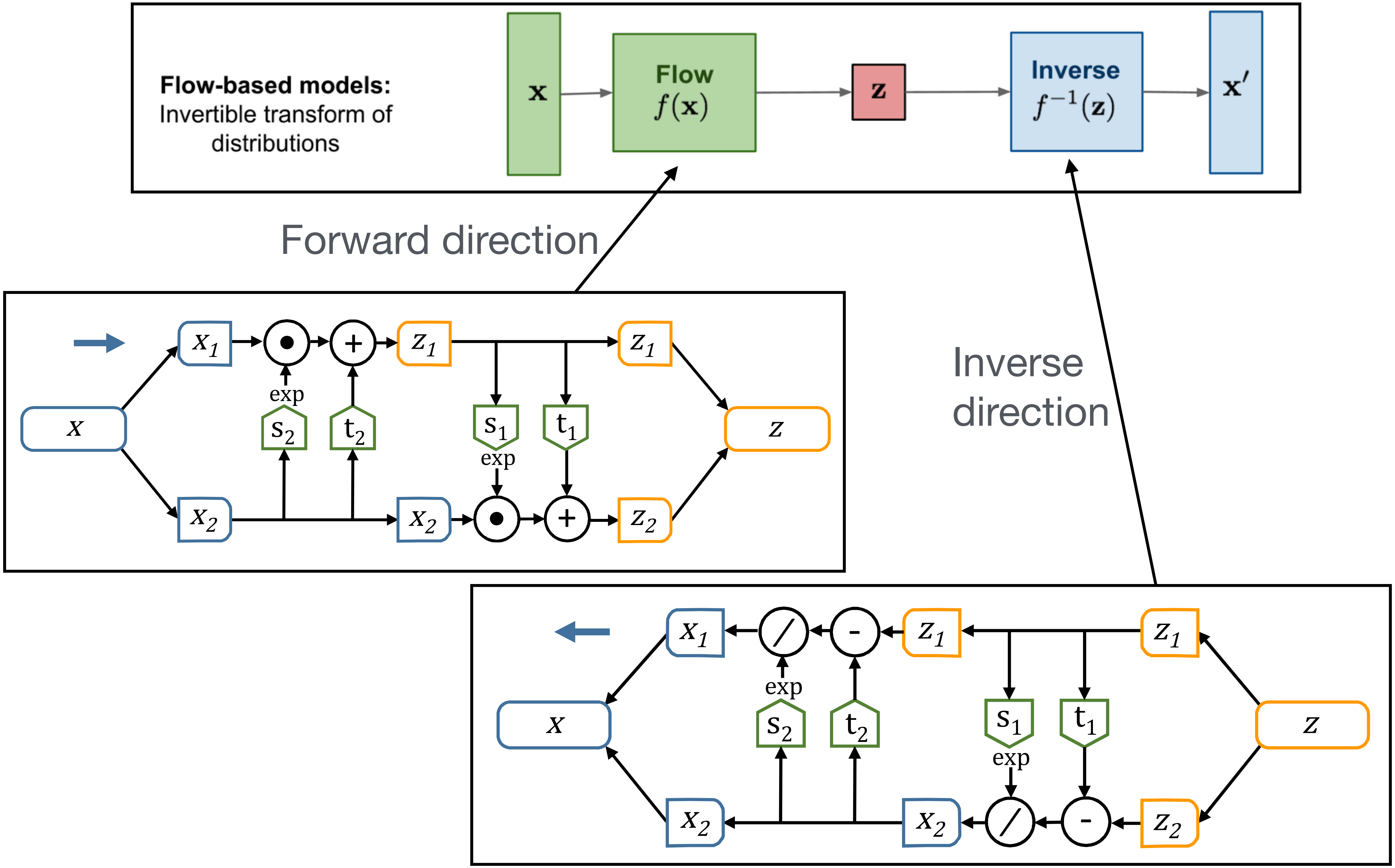
Coupling flows



Simple (e.g. dense)
neural networks

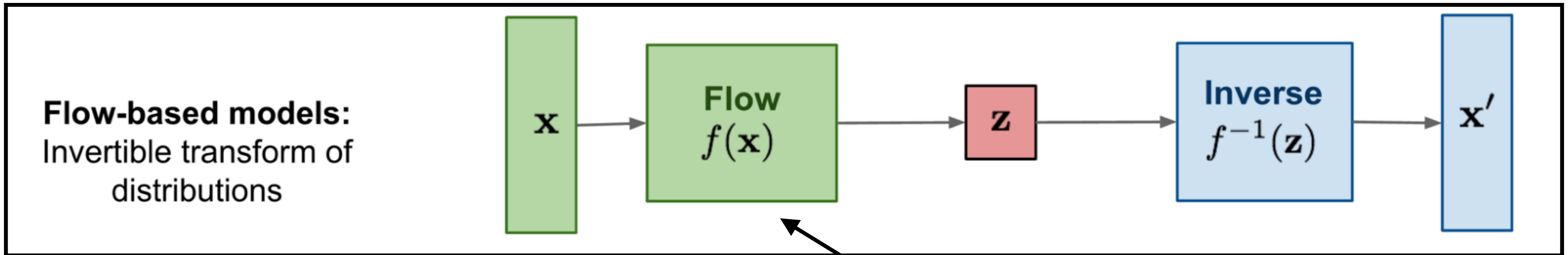
1505.05770, 1908.09257

Coupling flows



1505.05770, 1908.09257

Generative models

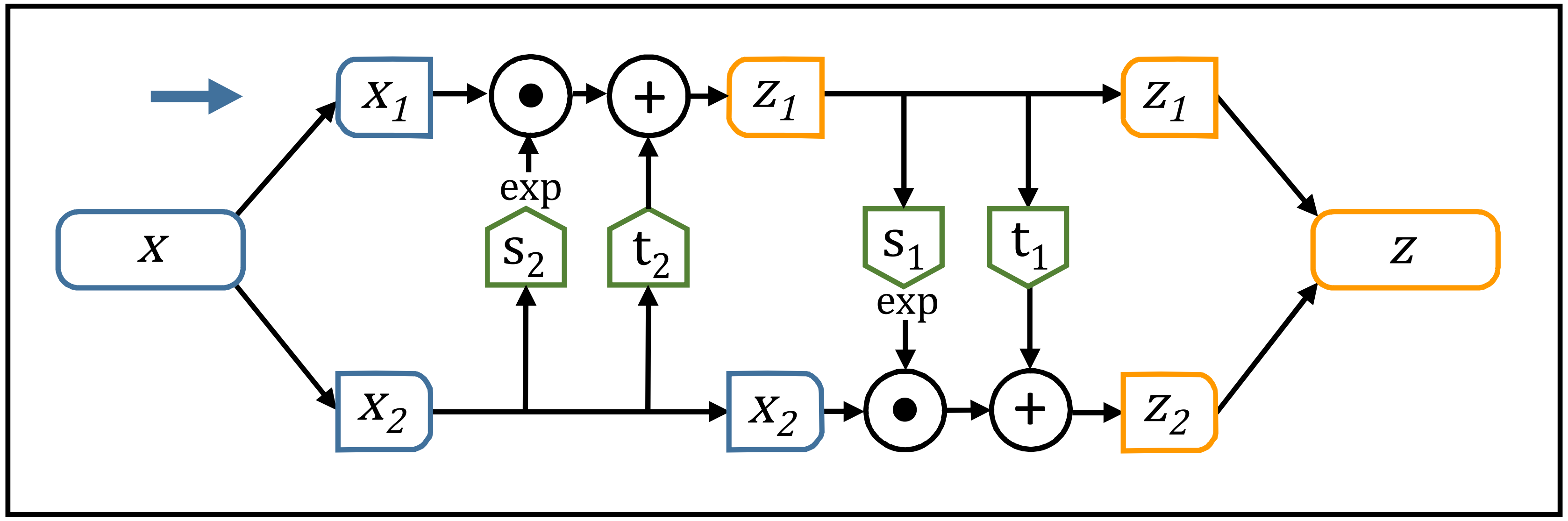


Take into account Jacobian determinant to evaluate probability density

2 challenges:
- Invertible
- Easy-to-calculate Jacobian

1505.05770, 1908.09257

Calculating Jacobian determinant



$$\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} \xrightarrow{f_1} \begin{pmatrix}\mathbf{z}_1\\ \mathbf{x}_2\end{pmatrix} \xrightarrow{f_2} \begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix}$$ with $$\begin{aligned}\mathbf{x}_1 &\xrightarrow{f_1} \mathbf{z}_1 = \mathbf{x}_1 \odot \exp(s_2(\mathbf{x}_2)) + t_2(\mathbf{x}_2)\\ \mathbf{x}_2 &\xrightarrow{f_1} \mathbf{x}_2.\end{aligned}$$

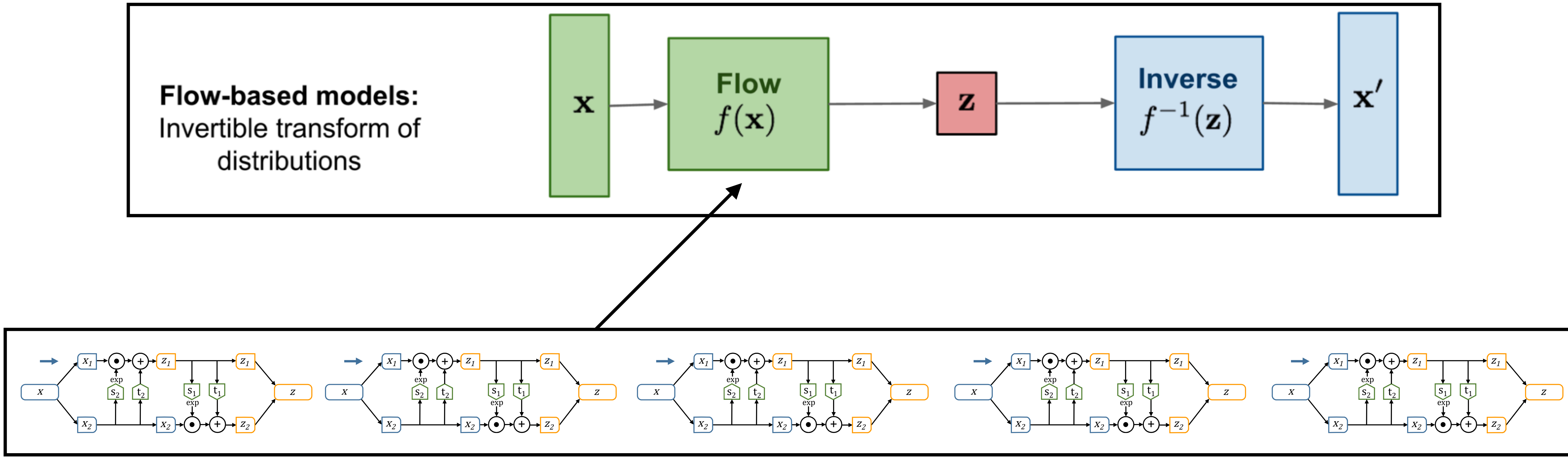
$$\mathbf{J_1} = \begin{pmatrix}\frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_2}\\ \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_2}\end{pmatrix} = \begin{pmatrix}\mathrm{diag}(\exp(s_2(\mathbf{x}_2))) & \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_2}\\ 0 & \mathbb{1}\end{pmatrix}$$

Triangular by construction

$$\det \mathbf{J_1} = \prod \exp(s_2(\mathbf{x}_2)) = \exp\left(\sum s_2(\mathbf{x}_2)\right)$$

1505.05770, 1908.09257

Composition

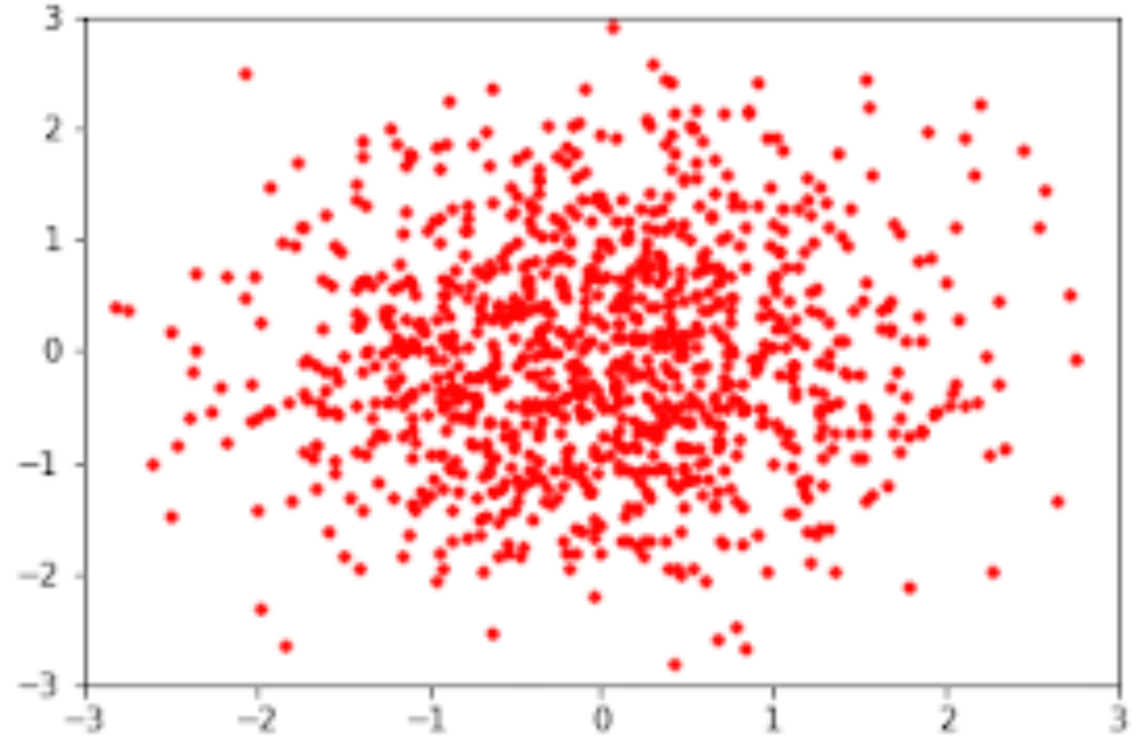


Composition of bijective functions remains bijective

Chain rule: Jacobian determinant of composition is product of determinants

1505.05770, 1908.09257

Animation



How to train NF?

Training objective: Minimise negative log likelihood of data

Sample points from training data

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[-\frac{1}{2}||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x})\right]$$

1505.05770, 1908.09257

How to train NF?

Training objective: Minimise negative log likelihood of data

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[-\frac{1}{2}||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x})\right]$$

Transform into latent space and evaluate probability there

$$\varphi(\mathbf{z}) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\mathbf{z}^2}$$

$$-\log(\varphi(f(\mathbf{x})) = -\log(\frac{1}{\sqrt{2\pi}}) - \log\left(\mathrm{e}^{-\frac{1}{2}f(\mathbf{x})^2}\right) = -\log(\frac{1}{\sqrt{2\pi}}) + \frac{1}{2}f(\mathbf{x})^2$$

How to train NF?

Training objective: Minimise negative log likelihood of data

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[-\frac{1}{2}||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x})\right]$$

Contribution from Jacobian determinant

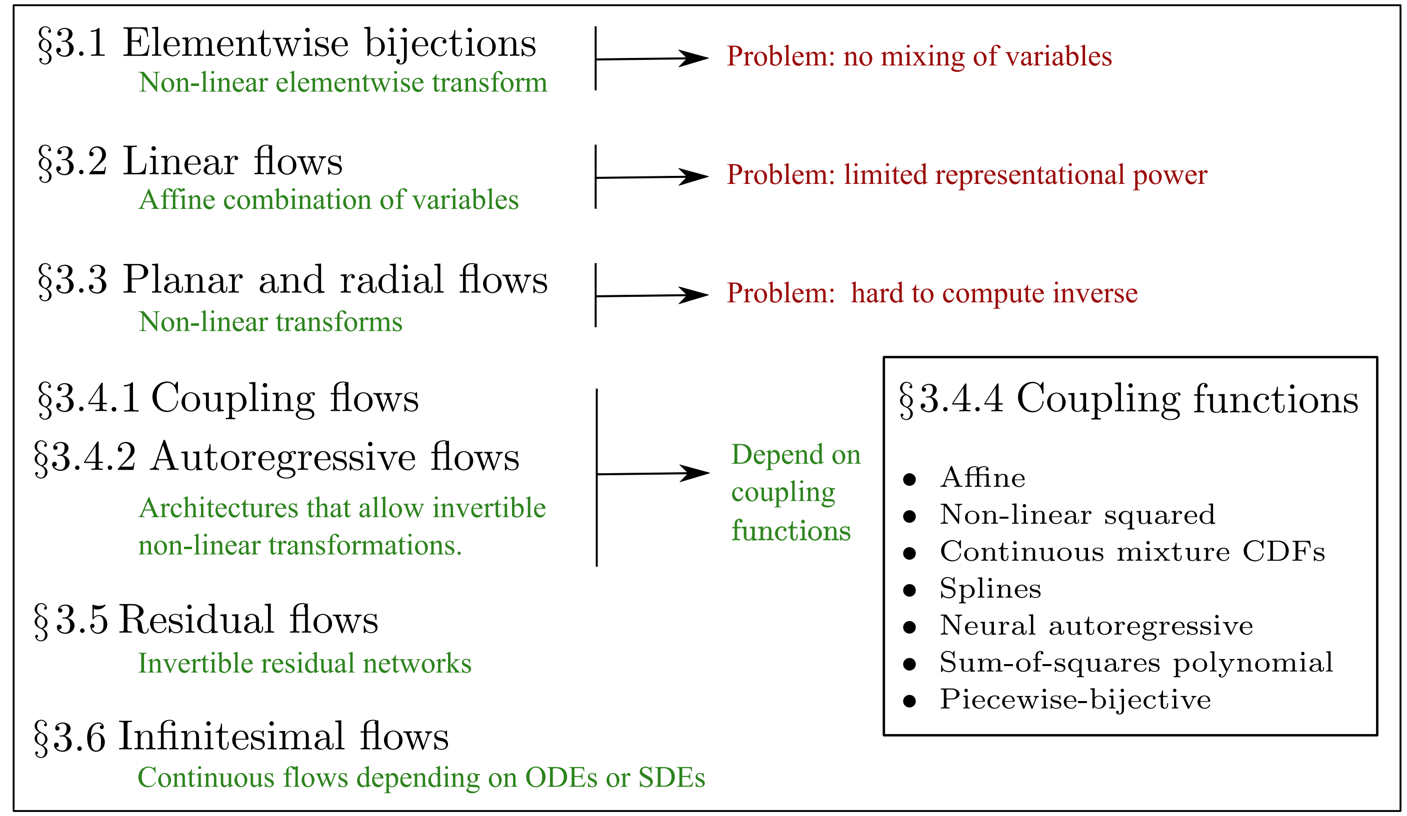
$$\det \mathbf{J} = \exp\left(\sum s(\mathbf{x})\right)$$

$$-\log(\det \mathbf{J}) = -\sum s(\mathbf{x})$$

1505.05770, 1908.09257

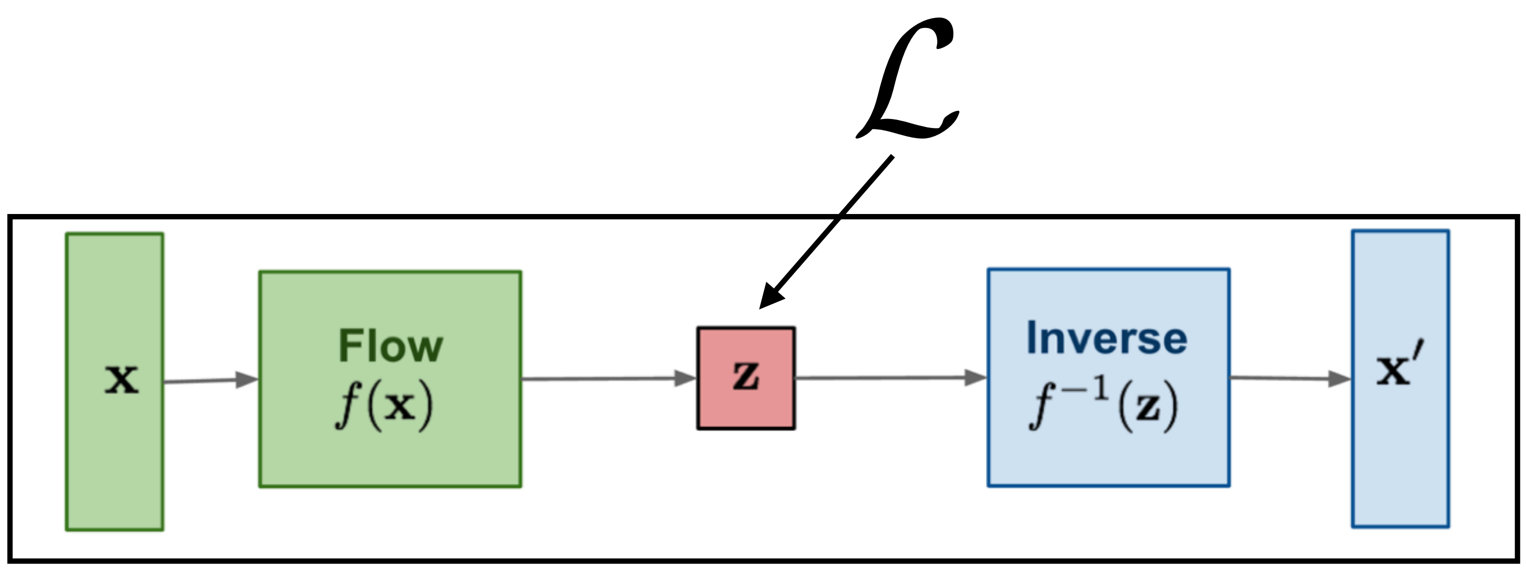
Comments on Flows

Only scratched the surface:
more constructions available



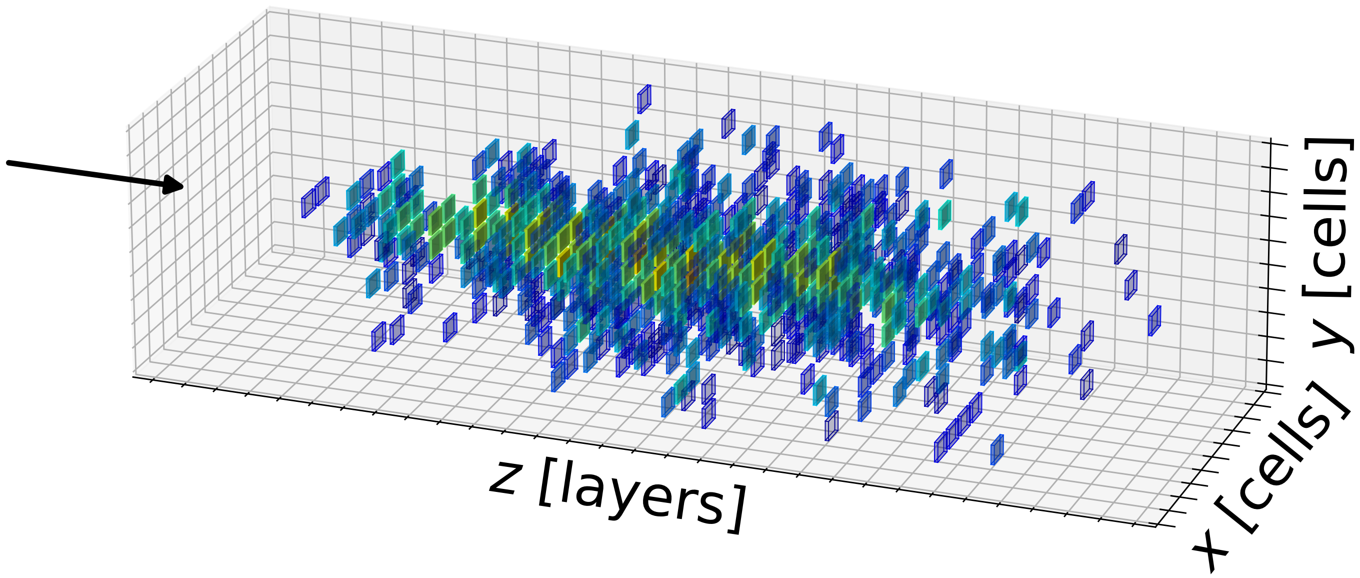
Exact learning of likelihood
→ Better generative fidelity
→ Can evaluate likelihood of data

More complex
→ Slower, choice of fast direction



1908.09257

Flows for detector simulation



10x10 cells / layer
30 layers

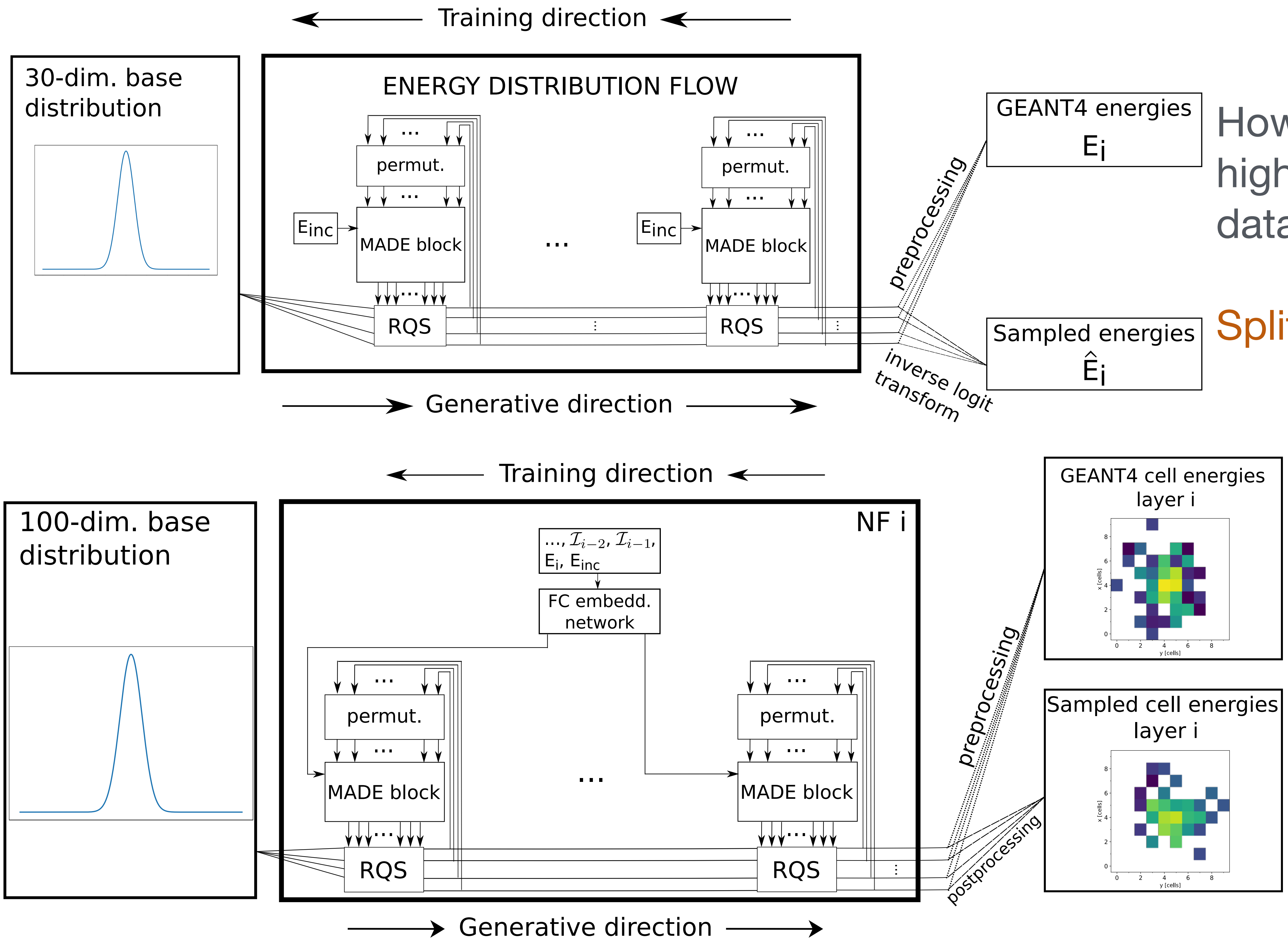
By directly learning the likelihood, flows should be of higher fidelity than GAN/VAE.

But inefficient scaling with data dimension.

How to do flows for high-dimensional data?

2302.11594

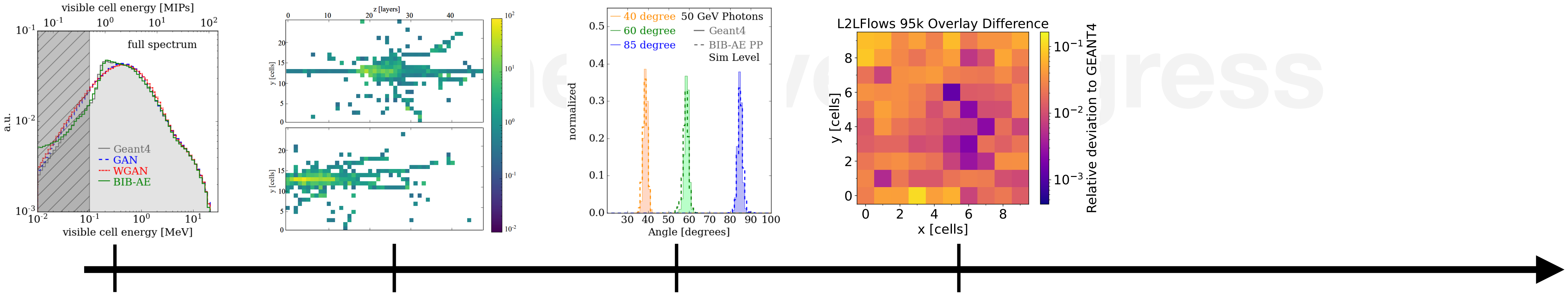
Flows for detector simulation



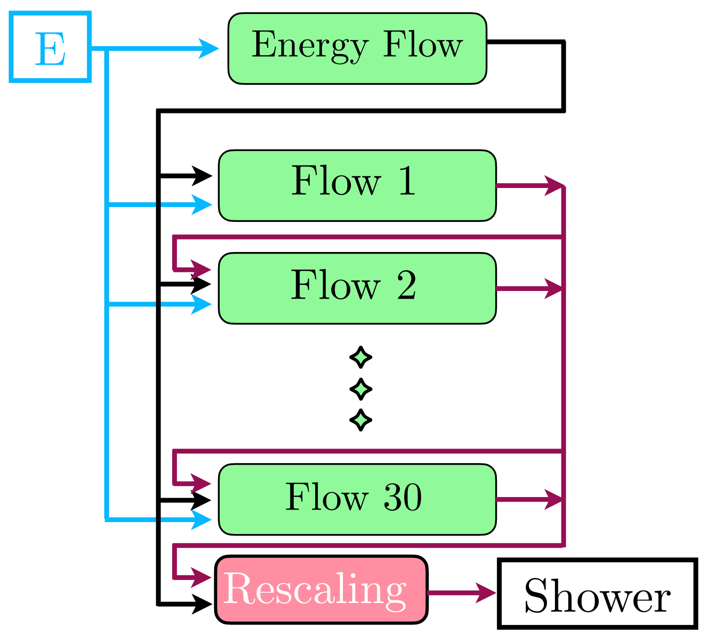
How to flows for high-dimensional data?

Split!

2302.11594

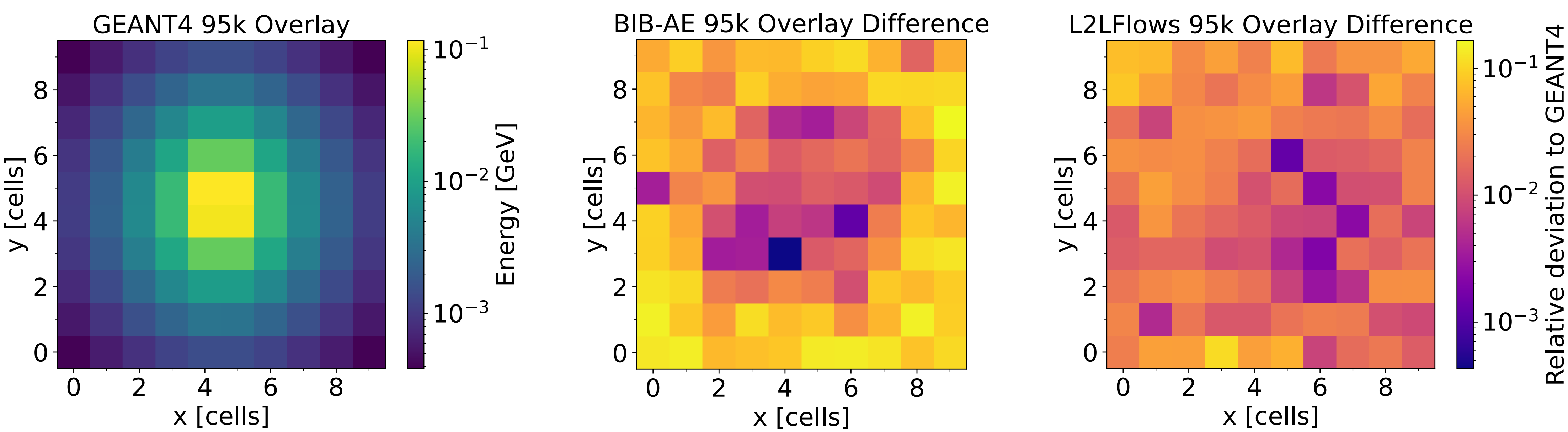


Progress



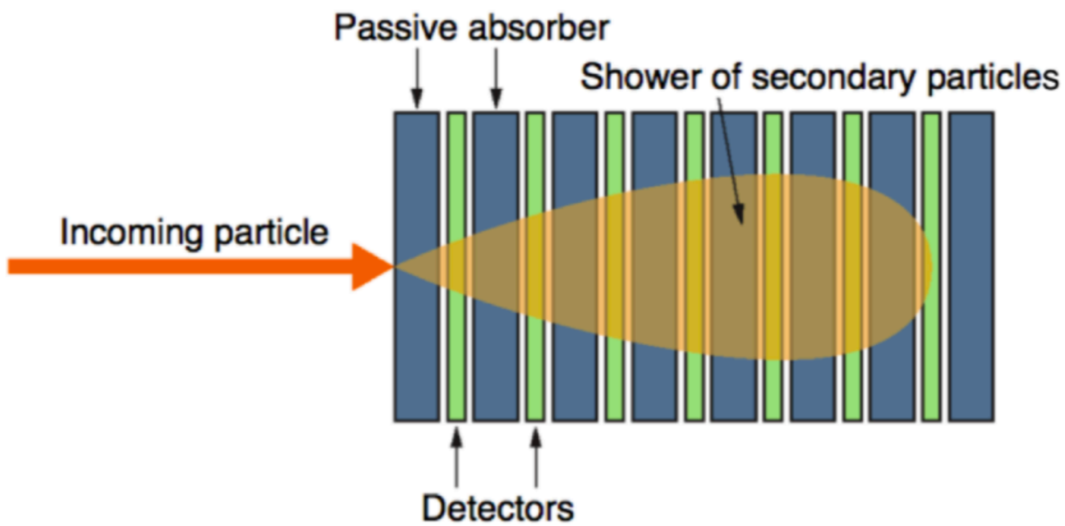
Better convergence of normalising flows:
→better fidelity

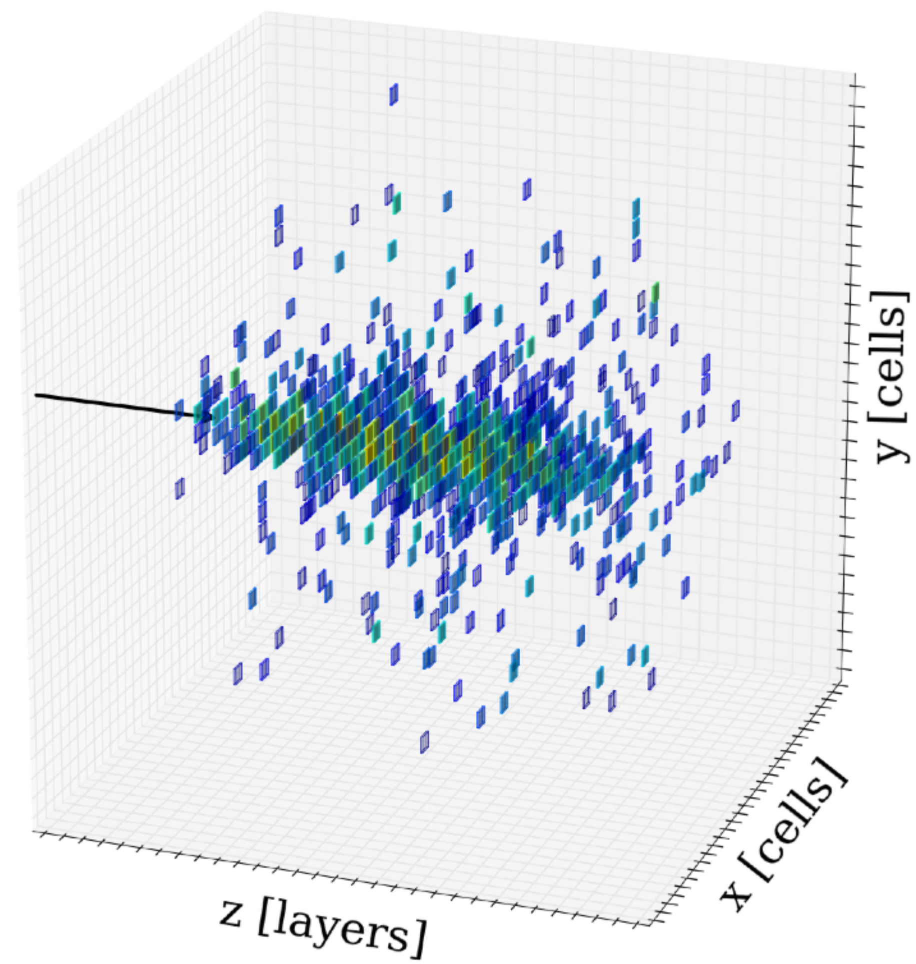
Individual flows per layer for efficiency



Diefenbacher, .., **GK** et al 2302.11594

Simulation targets





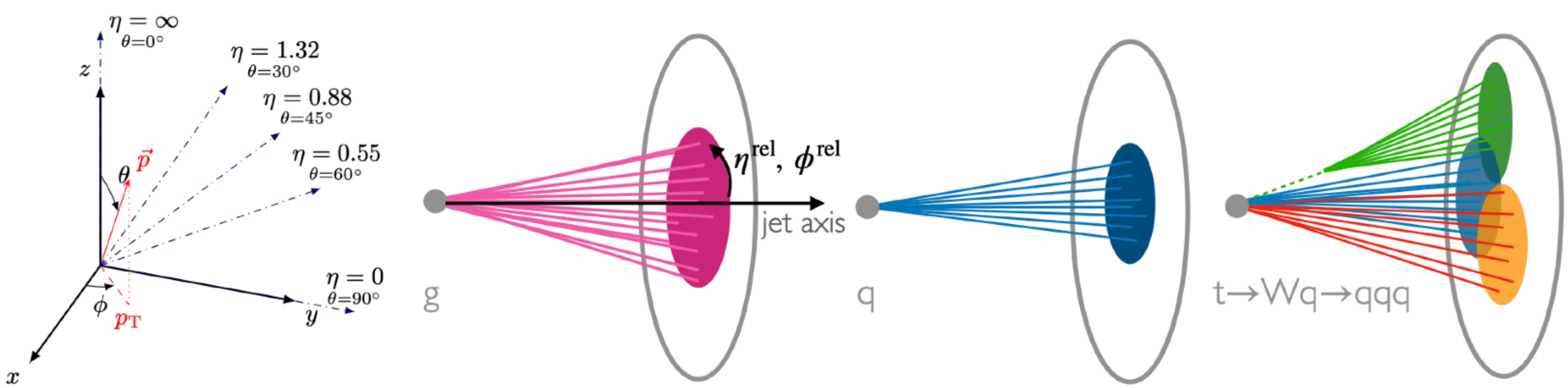
How to represent?

Tabular data

Fixed grid (voxels)
Limiting for high-dimensions (sparse data)

Instead: Point clouds / graphs

Simulation targets



Before tackling showers in calorimeters:
Look at jet constituents (JetNet data):
3 features per constituents
up to 30/150 constituents/jet

Why?
Useful stepping stone
In-situ background

How to represent?

Tabular data

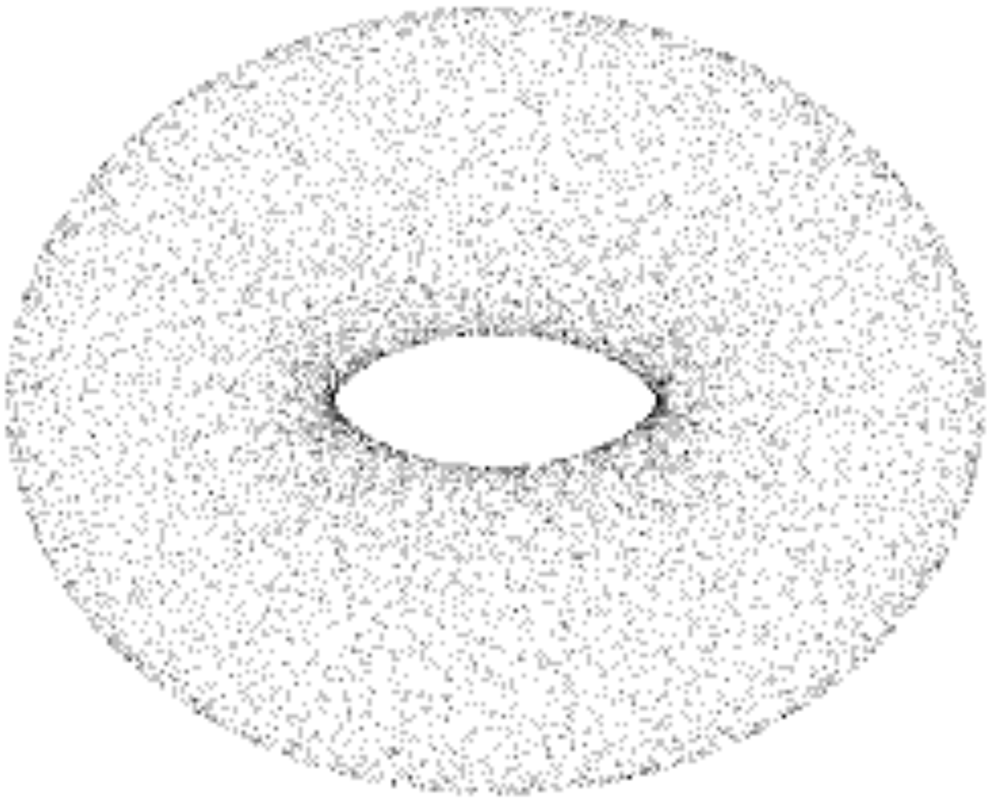
Fixed grid (voxels)
Limiting for high-dimensions (sparse data)

Instead: Point clouds / graphs

2106.11535

Point Clouds

- Example:
  Sensors in a space
  - Fixed grid vs arbitrary positions
  - Potential sparsity of data
- Permutation symmetry
- Can view as trivial graph



$$X = \left\{ \begin{array}{l} \vec{p}_1 = \{r_1, \theta_1, \varphi_1, T_1\} \\ \vdots \\ \vec{p}_L = \{r_L, \theta_L, \varphi_L, T_L\} \end{array} \right\}$$

$$f(X) = g\left(\Sigma_i \, \phi(\vec{p}_i)\right)$$

$$\Uparrow$$

$$f: \mathbb{R}^{4 \times L} \rightarrow \mathbb{R}^1$$

$$\phi: \mathbb{R}^4 \rightarrow \mathbb{R}^S$$

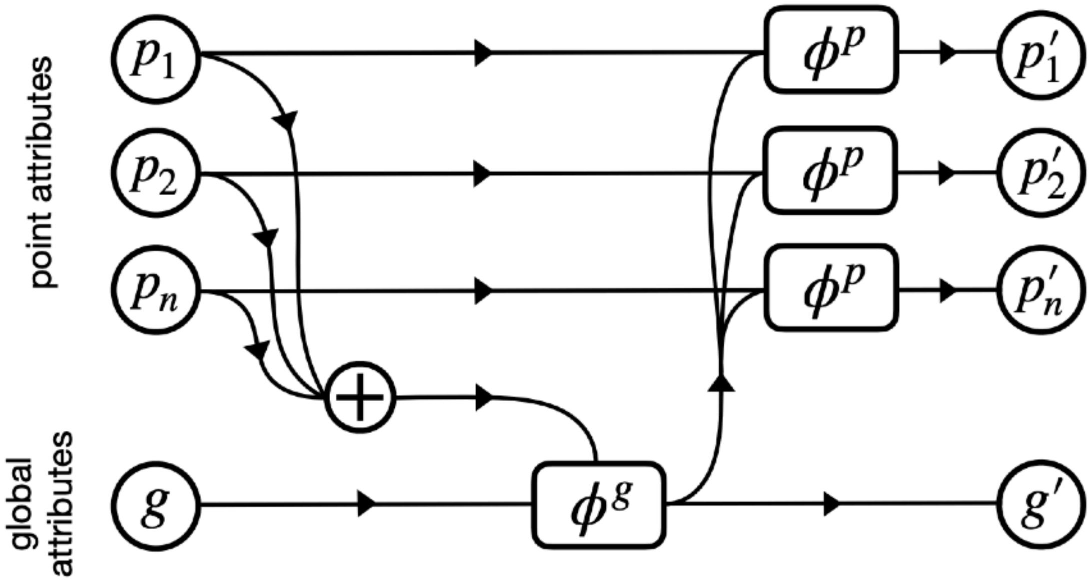
Latent Space Dimension

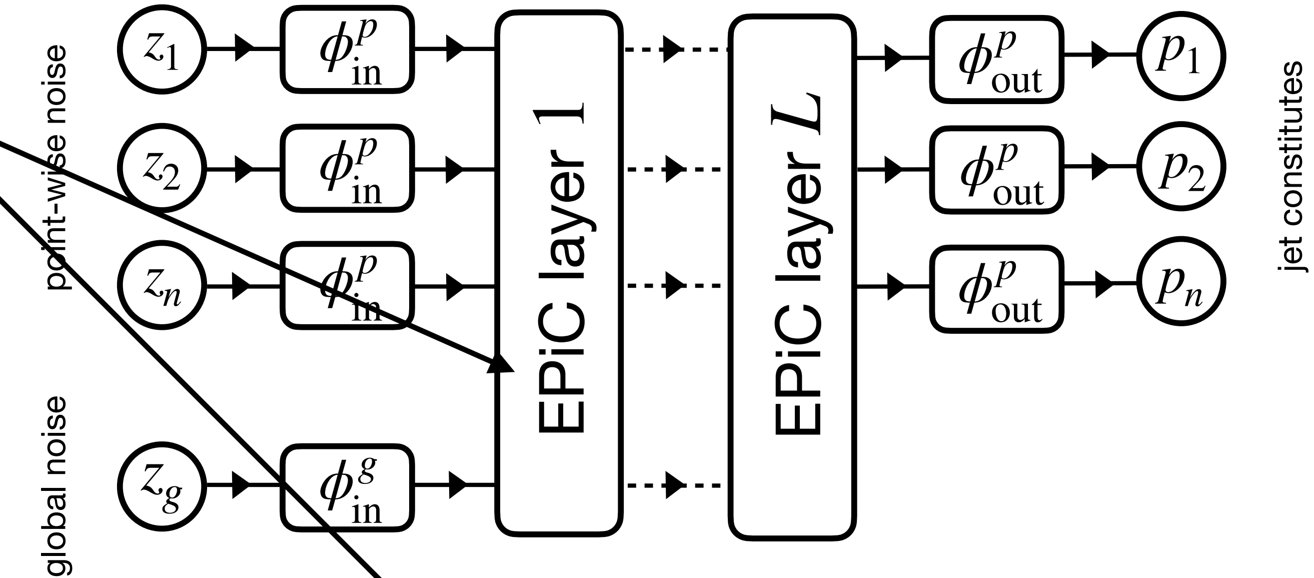
$$\Sigma: \mathbb{R}^{S \times L} \rightarrow \mathbb{R}^S$$

$$g: \mathbb{R}^S \rightarrow \mathbb{R}^1$$

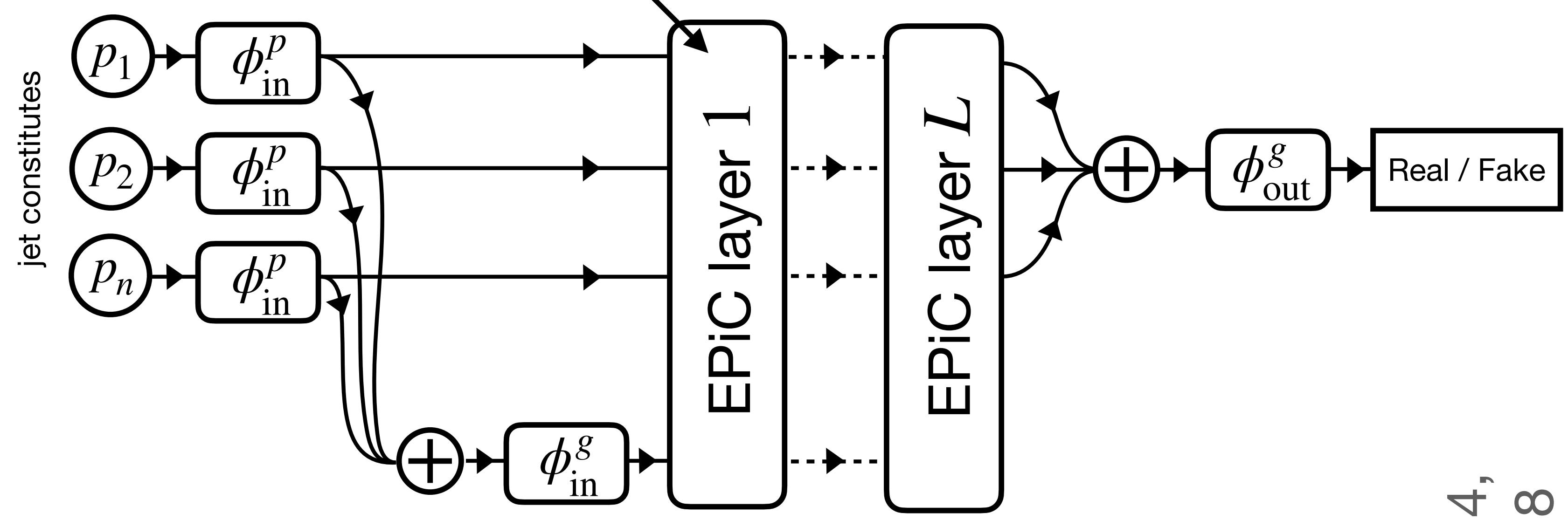
1703.06114, 1810.05165

How to GAN with it





(a) Generator

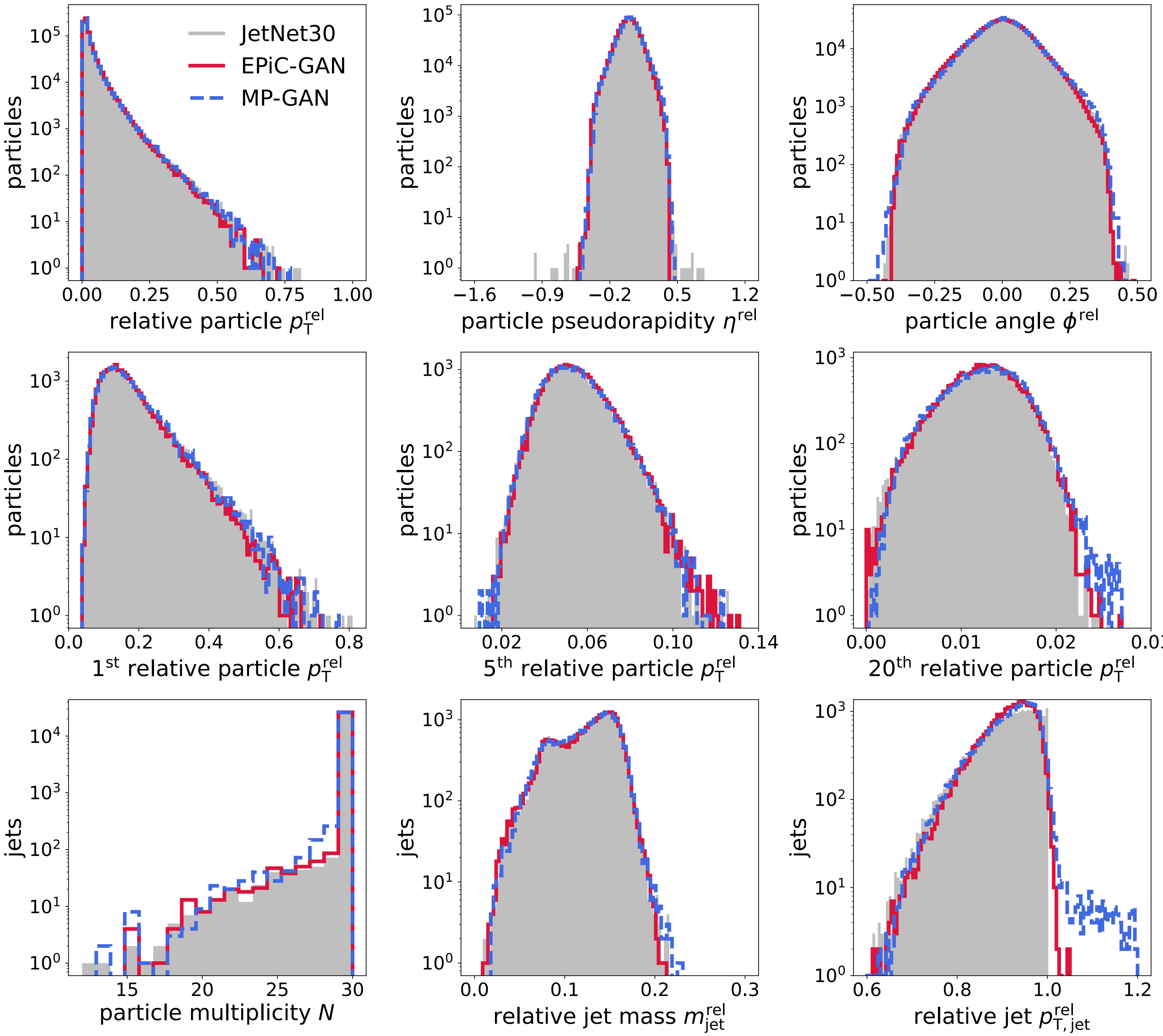


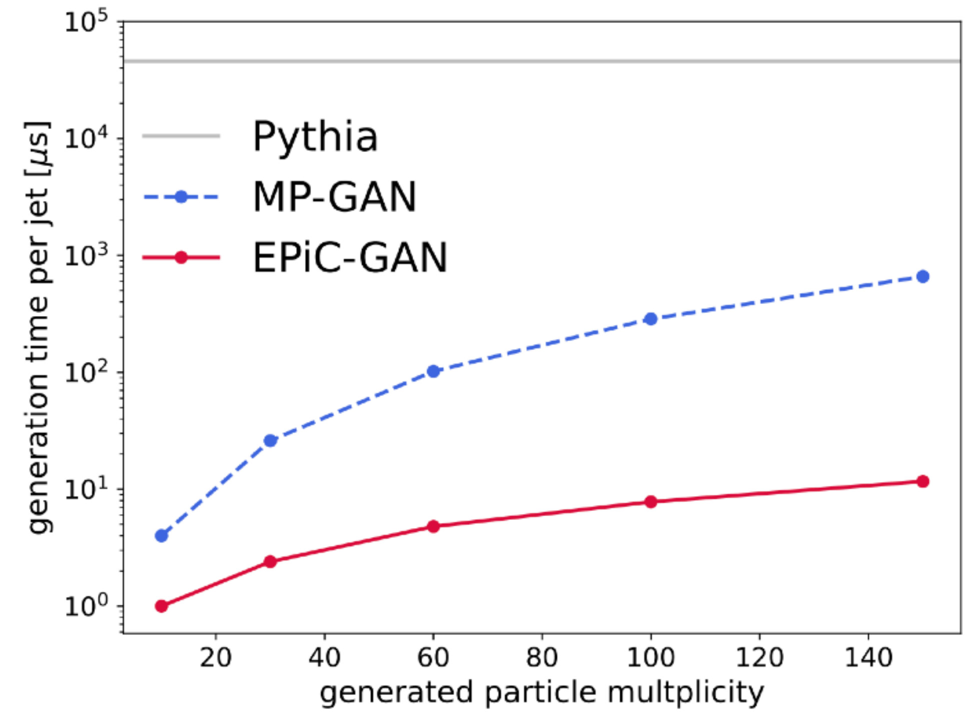
(b) Discriminator

Add permutation symmetry to GAN architecture

1703.06114, 2301.08128

EPIC GAN Result

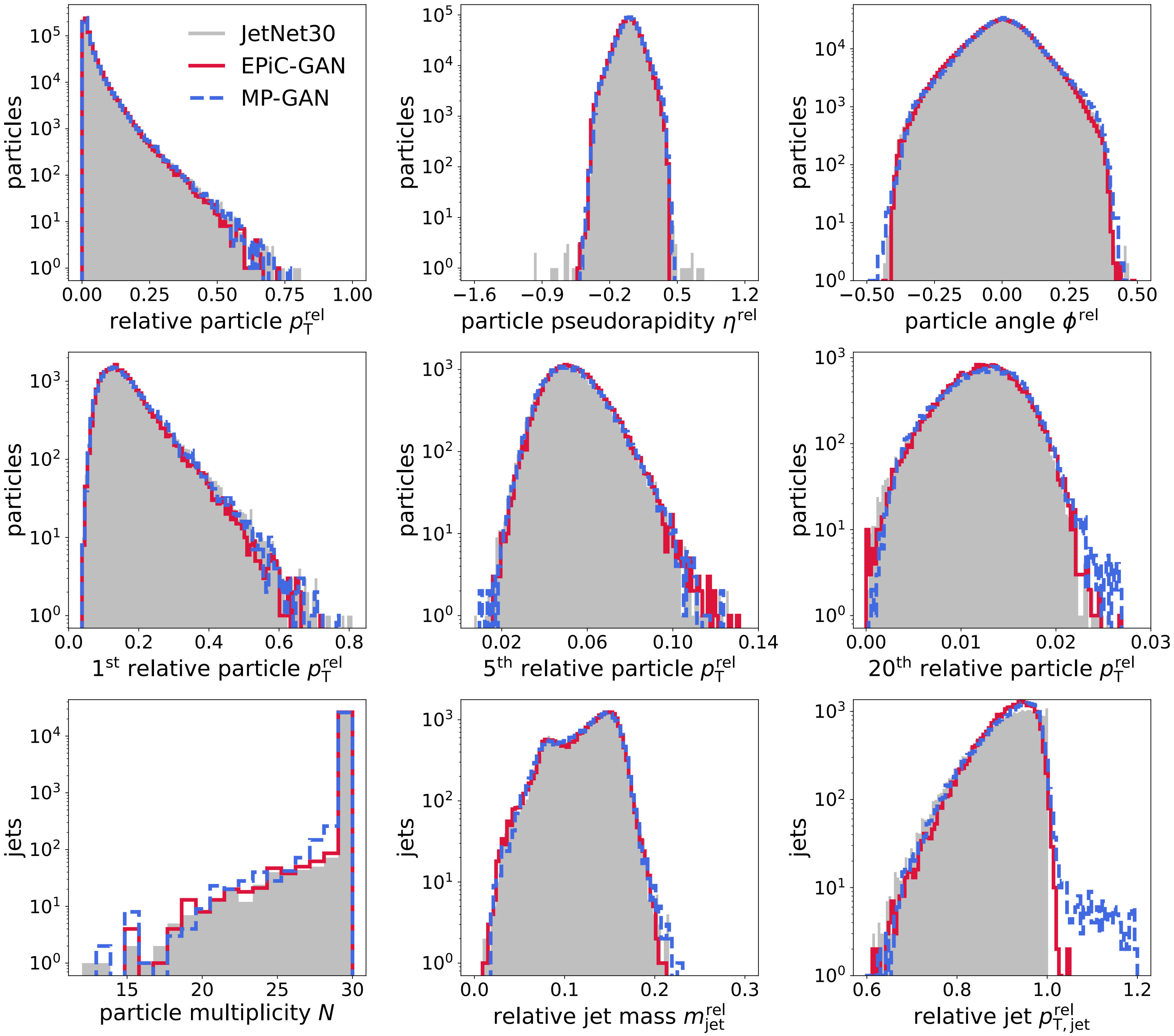


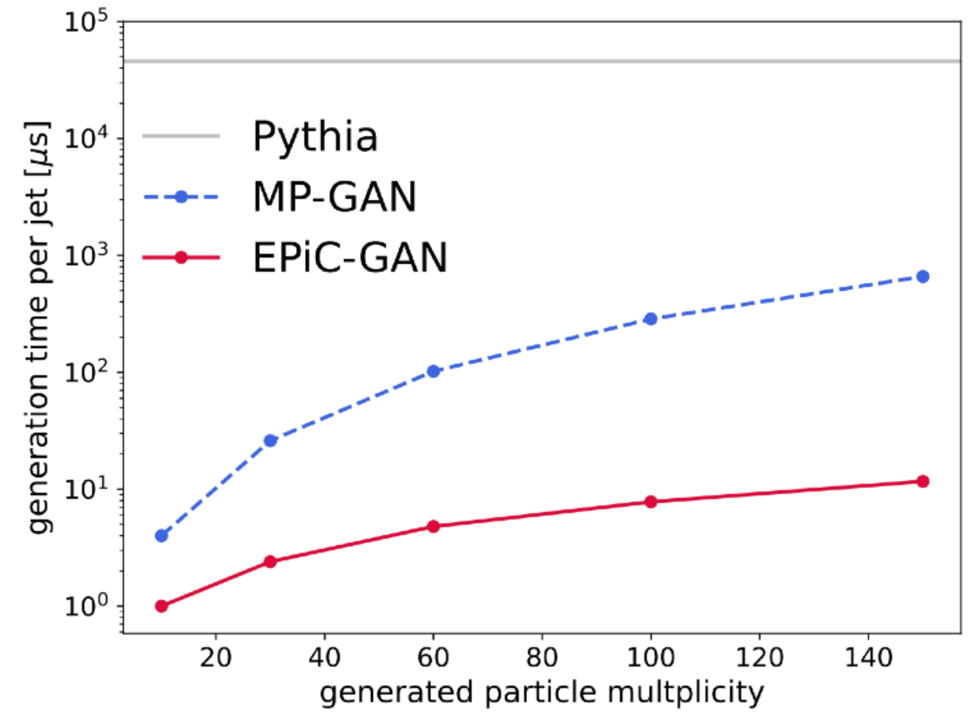


Point clouds/graph GANs successfully generate jet constituents

2301.08128

EPIC GAN Result



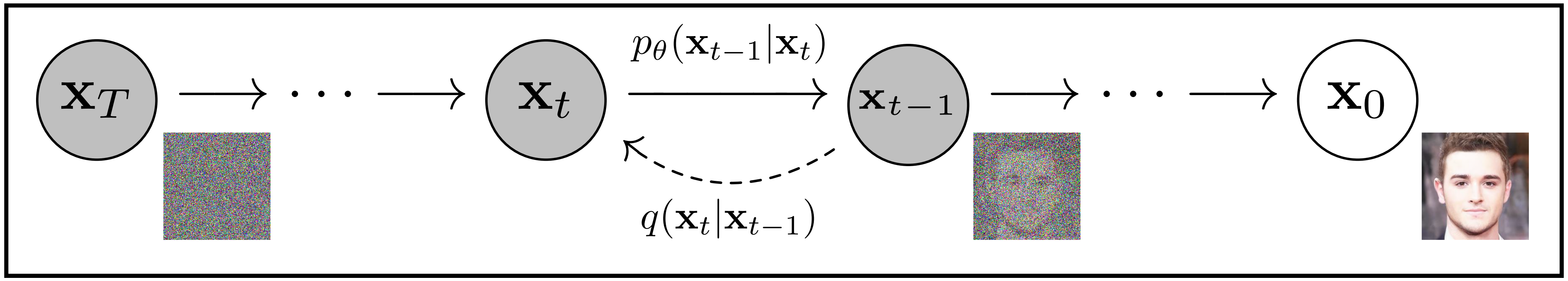


Point clouds/graph GANs successfully generate jet constituents

How to apply for calorimeter simulation?
Need a better architecture than GANs

2301.08128

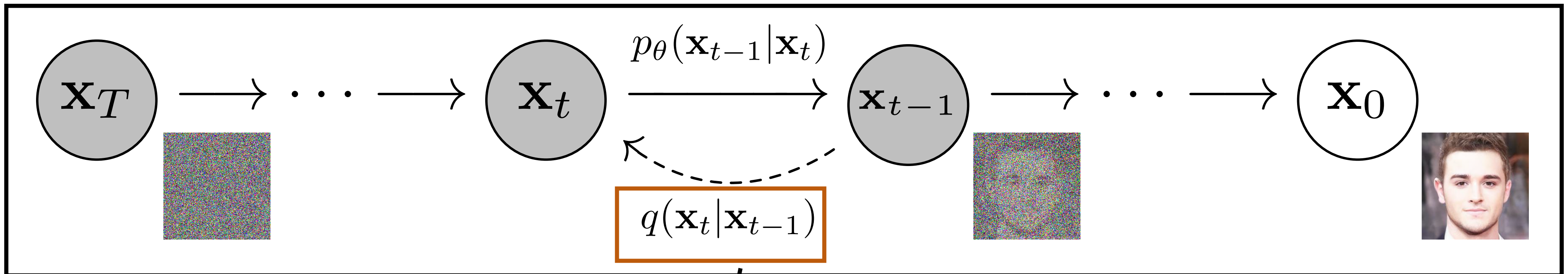
Diffusion Model



Core idea: Stepwise transition from pure noise to data

Markov chain

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Individual step

Noise schedule (hyper-parameter)

Forward (Data → Noise)

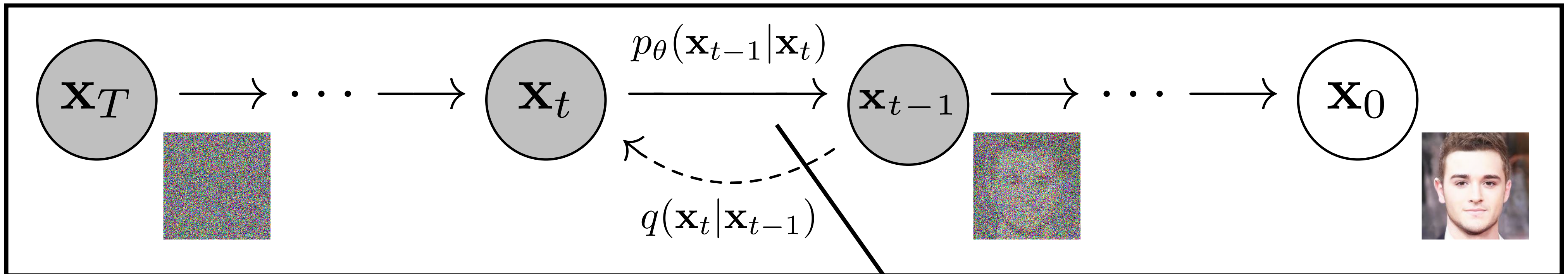
$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boxed{\boldsymbol{\epsilon}} \text{ for } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Rewrite: State at any time

Will try to predict

$$\alpha_t := 1 - \beta_t \qquad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

Diffusion Model



Core idea: Stepwise transition from pure noise to data

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Reverse (Noise → data)

Resulting learning objective

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\| \boldsymbol{\epsilon} - \boxed{\boldsymbol{\epsilon}_\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

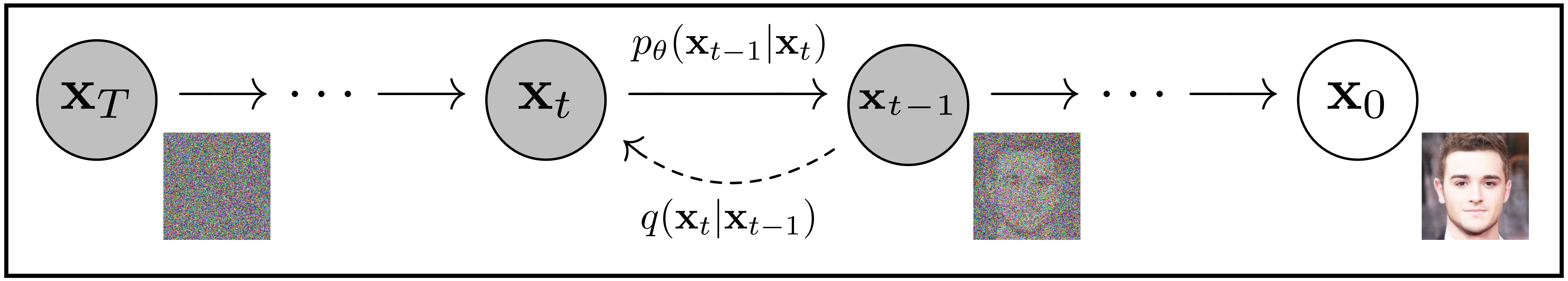
Noisy image

Reminder: Forward diffusion to time t

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

Timestep

Diffusion Model



Core idea: Stepwise transition
from pure noise to data

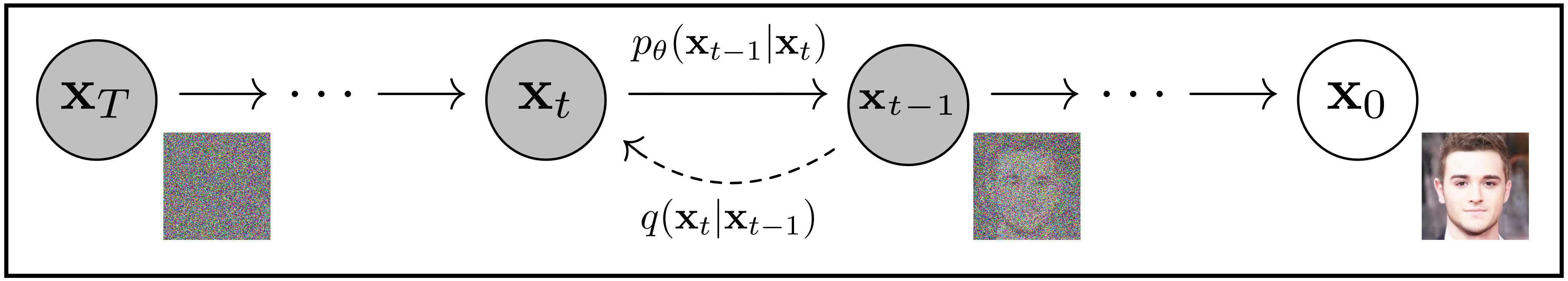
Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
until converged

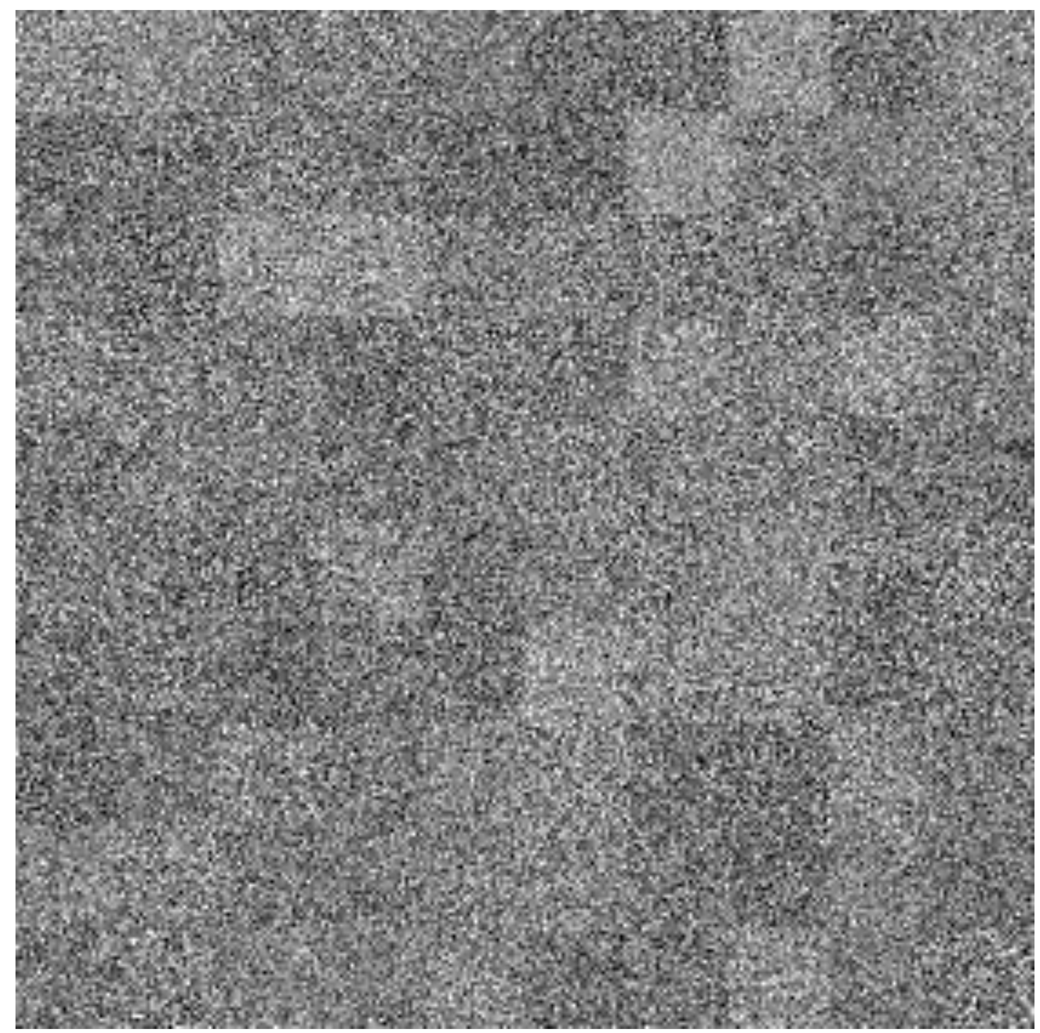
Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Diffusion Model



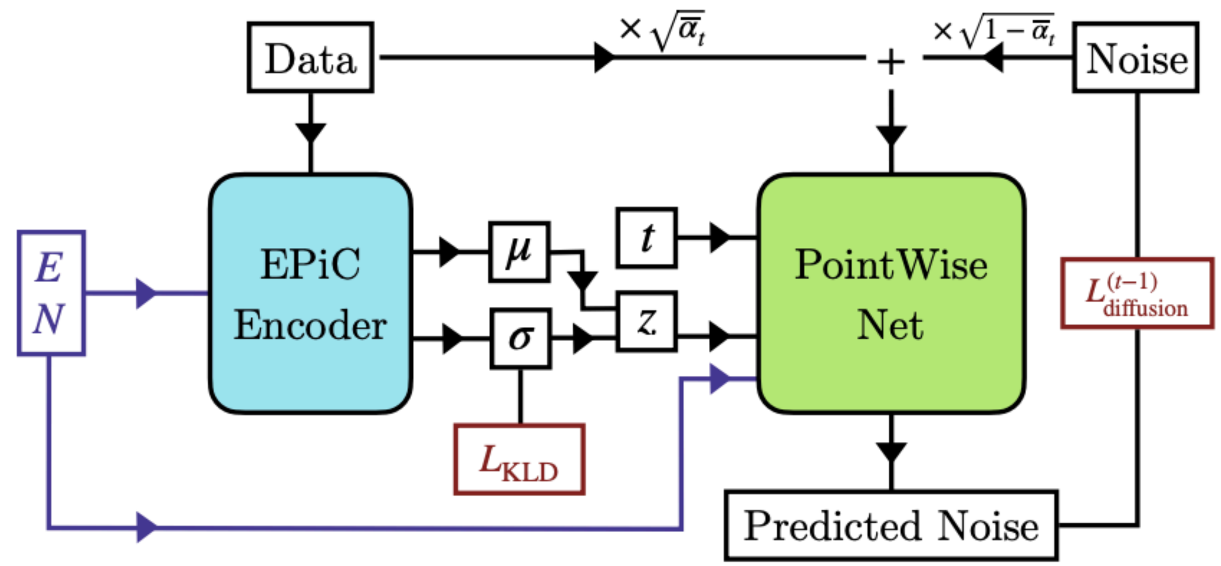
Core idea: Stepwise transition
from pure noise to data



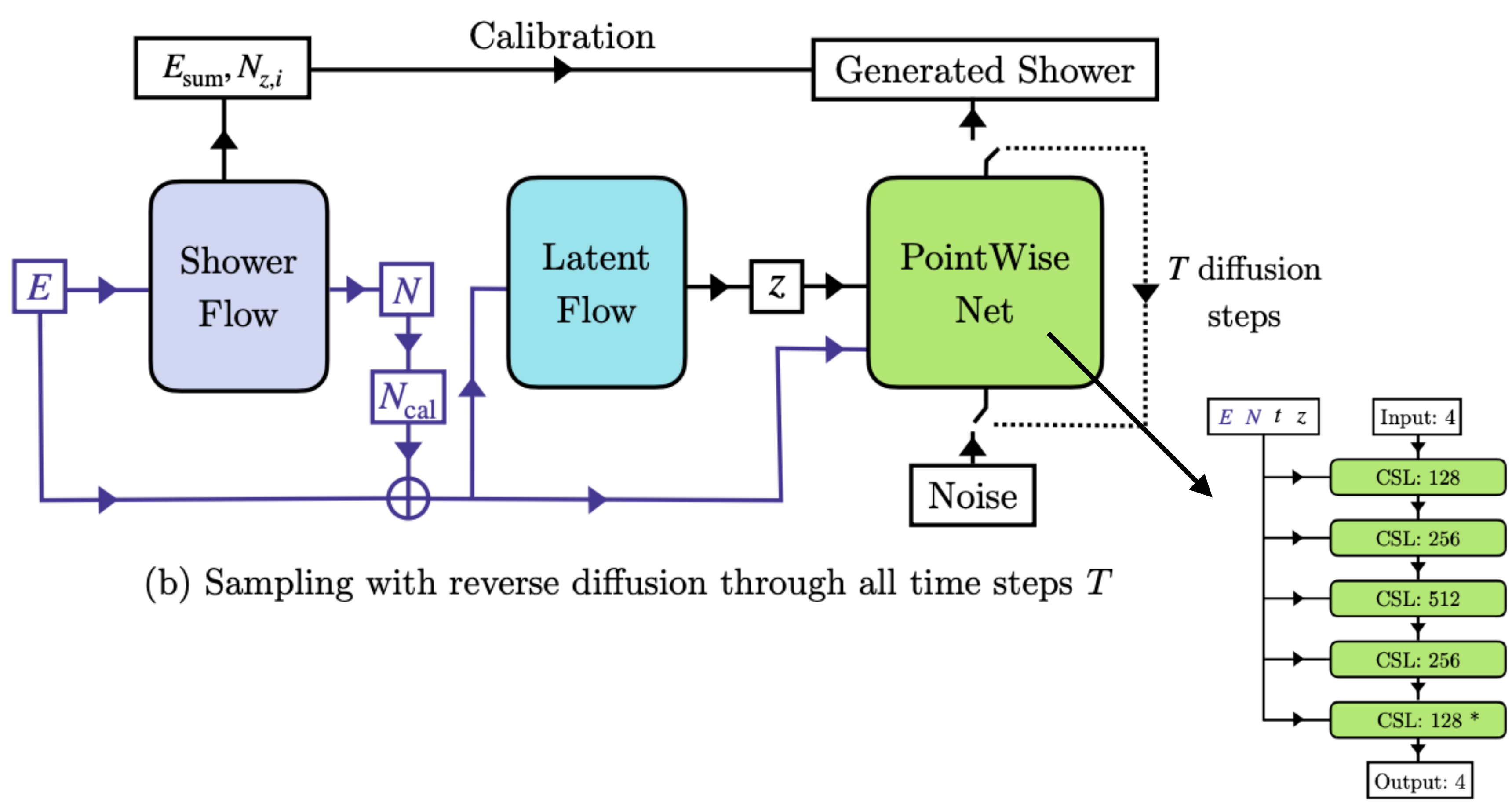
https://medium.com/mlearning-ai/enerating-images-with-ddpms-a-pytorch-implementation-cef5a2ba8cb1

Point Cloud Generation

To improve the generative fidelity, move from GAN to diffusion model



(a) Training at random time step t



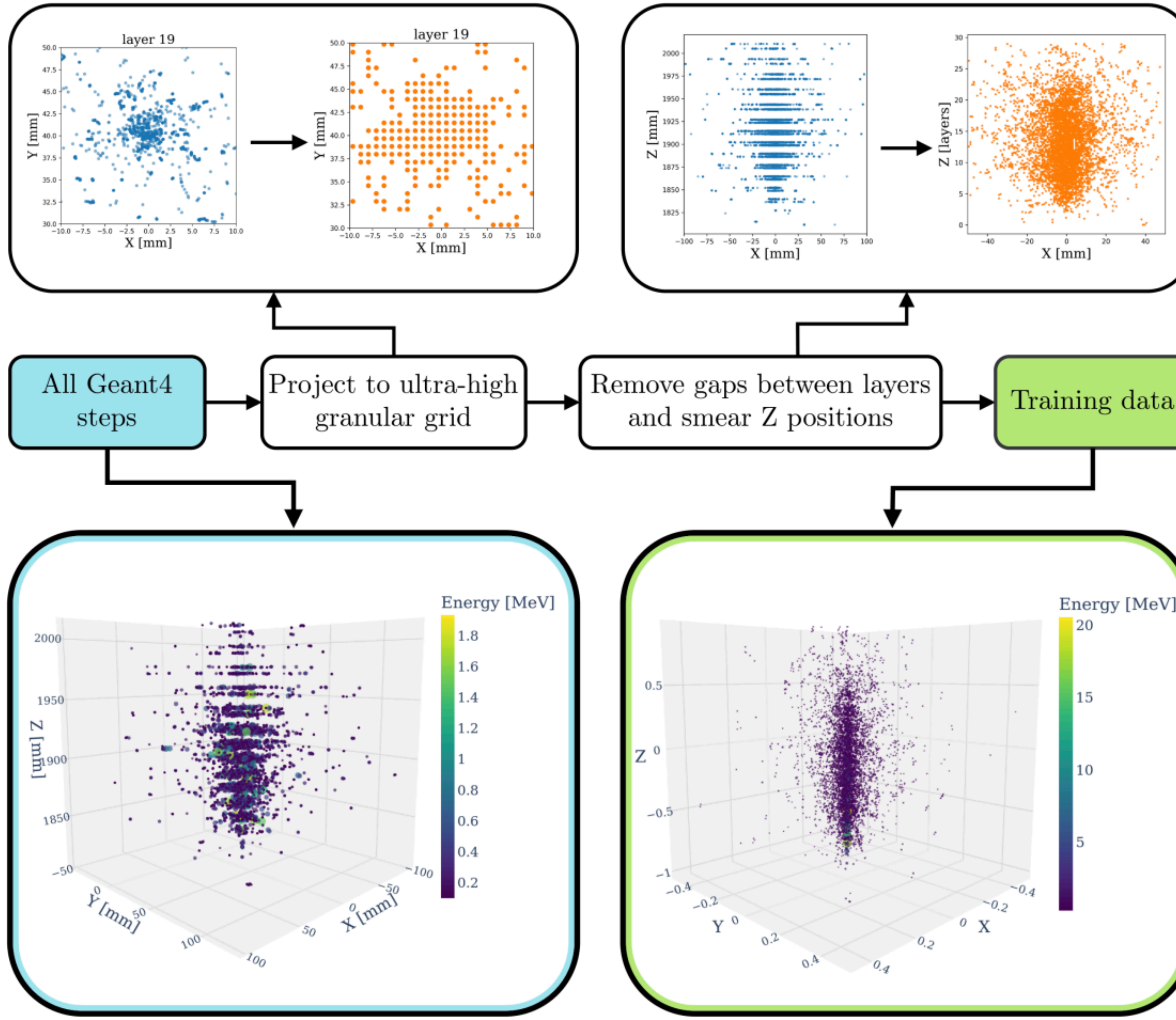
(b) Sampling with reverse diffusion through all time steps T

Buhmann, GK, Thaler 2301.08128; Kansal et al 2106.11535; Käch et al 2211.13630; Buhmann, … GK, et al 2305.04847

Point Cloud Generation

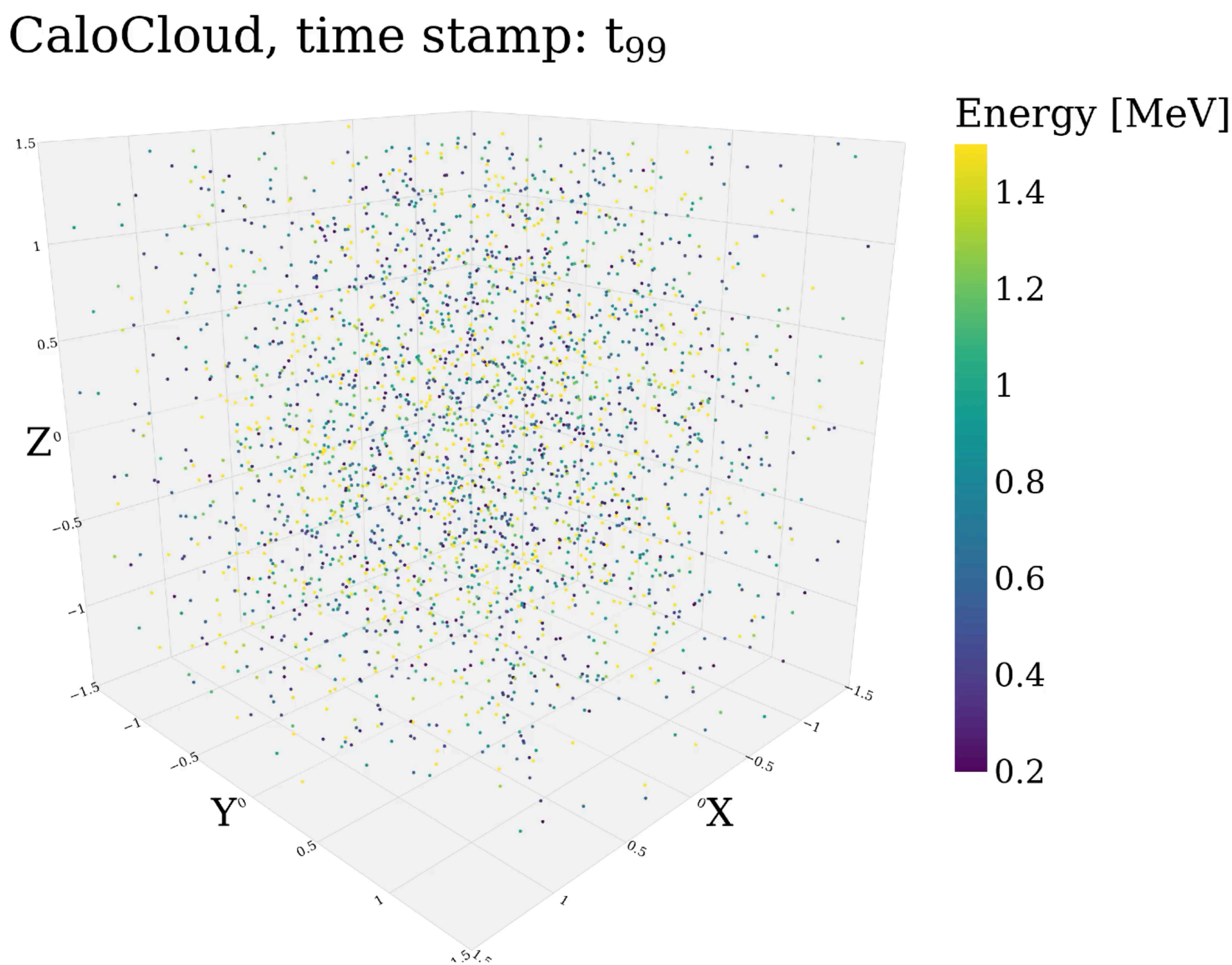
To improve the generative fidelity, move from GAN to diffusion model

Some additional pre-processing

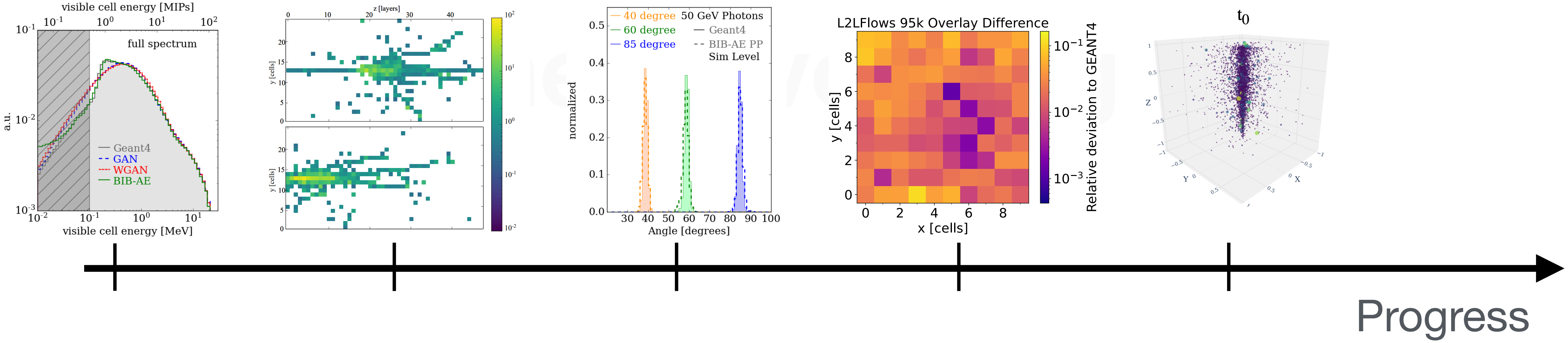


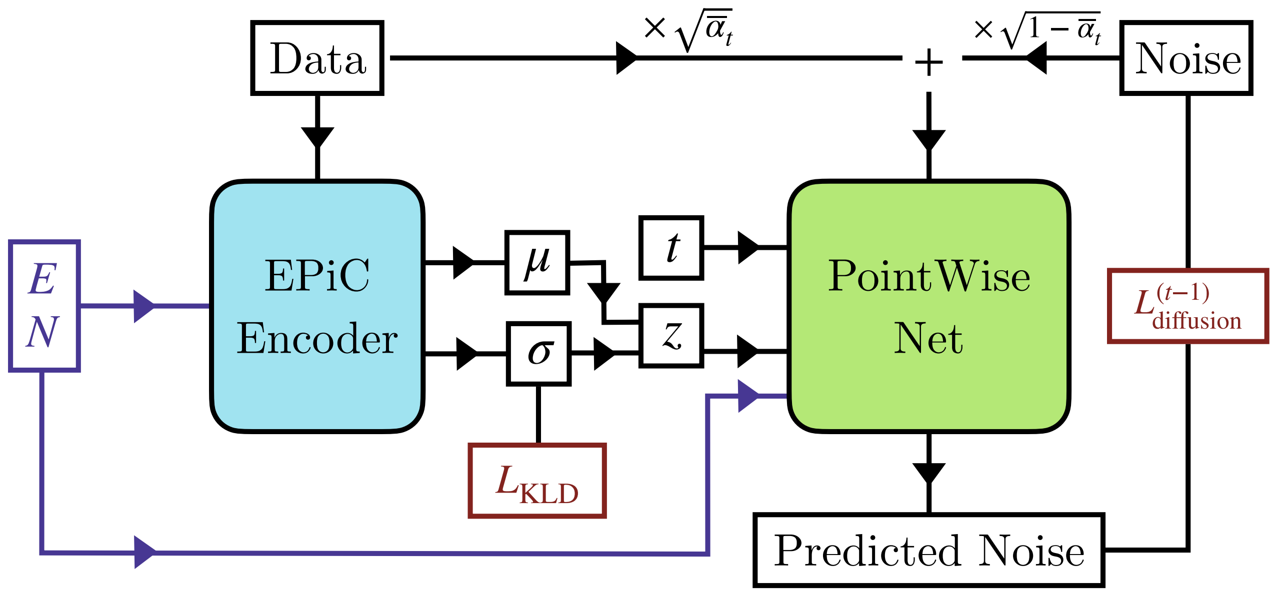
Buhmann, GK, Thaler 2301.08128; Kansal et al 2106.11535; Käch et al 2211.13630; Buhmann, … GK, et al 2305.04847

Diffusion

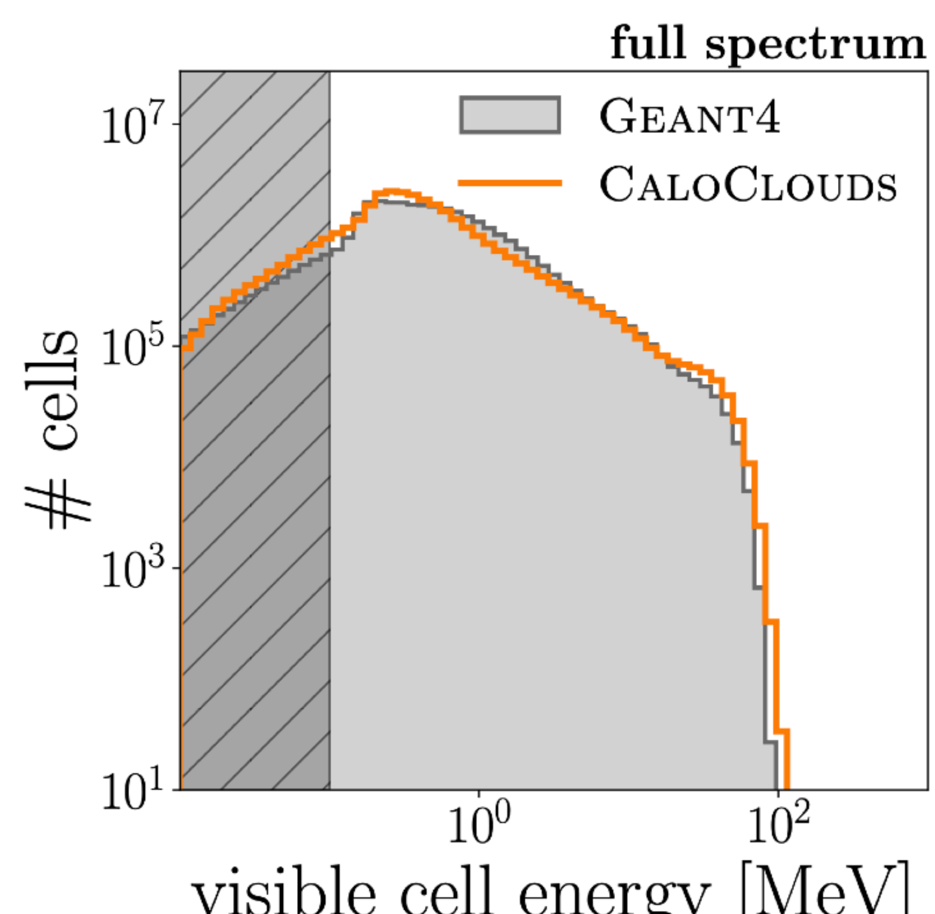


Buhmann, … GK, et al 2305.04847





Add permutation invariance & diffusion:
→simulate as point cloud!

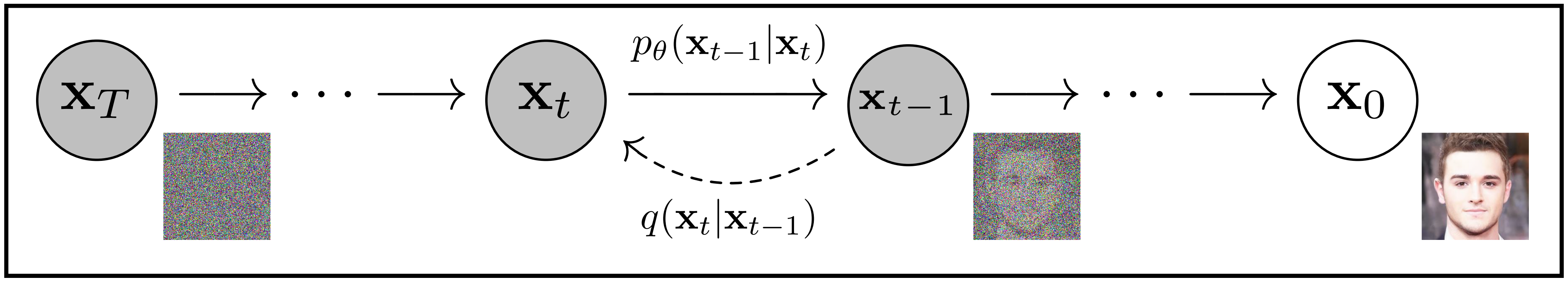


Close in accuracy to fixed grid.
Fairly slow.

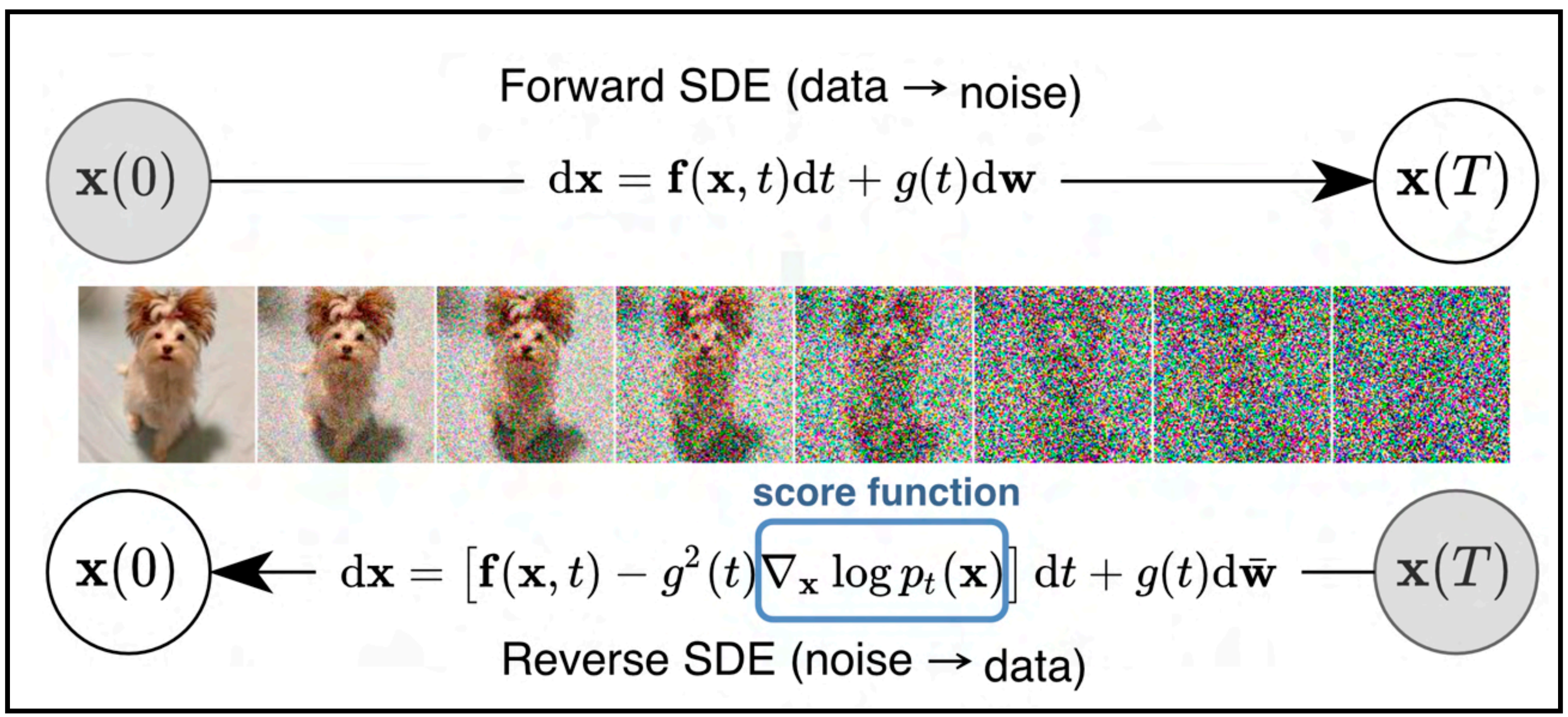
Can we improve further?

Buhmann, .., **GK**, et al 2305.04847

Continuos Diffusion Model



Replace discrete noise steps



with stochastic differential equations (SDEs)

Continuos Diffusion Model

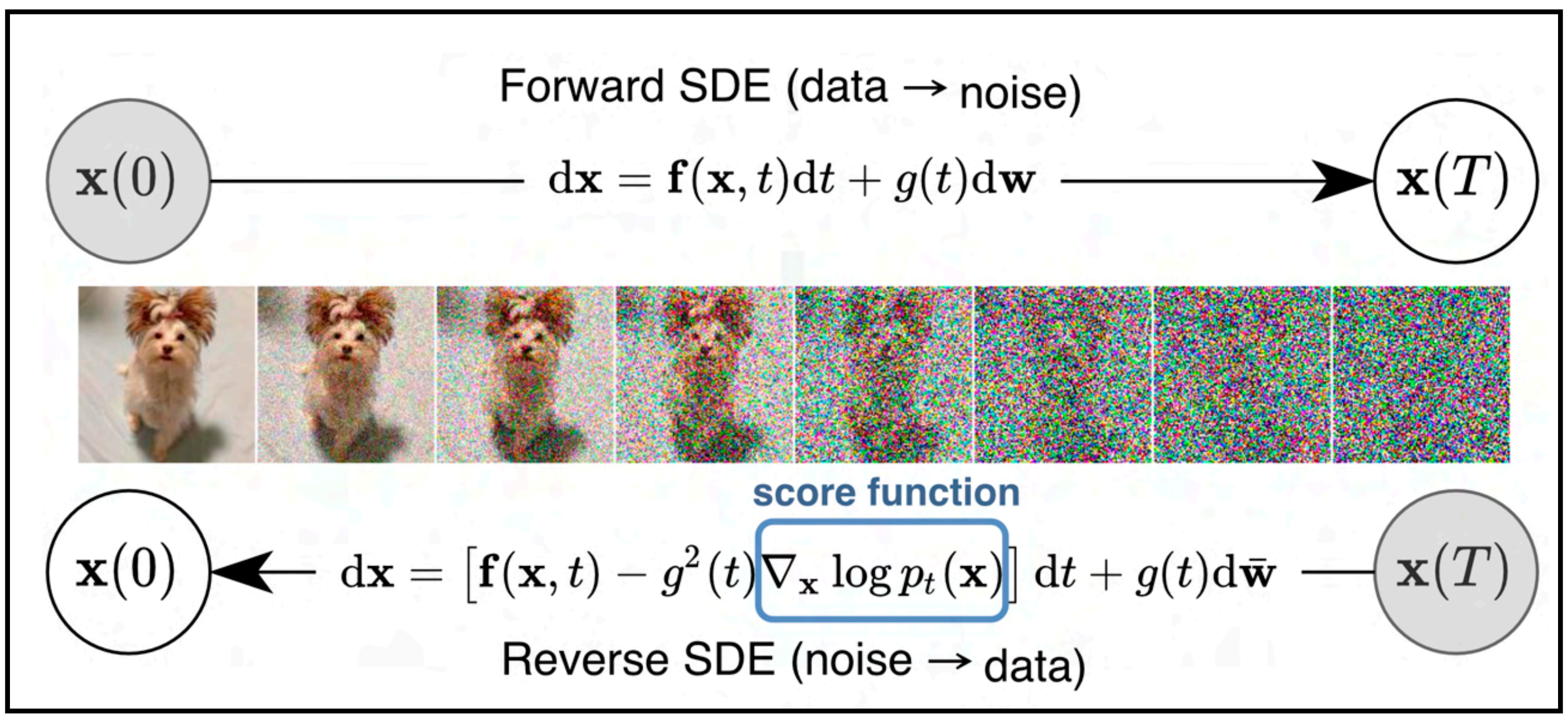
Forward SDE: $$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

Drift term

Diffusion term

Wiener process/ Brownian motion (independent Gaussian increments)

(Correspond to the noise schedule in discrete case)



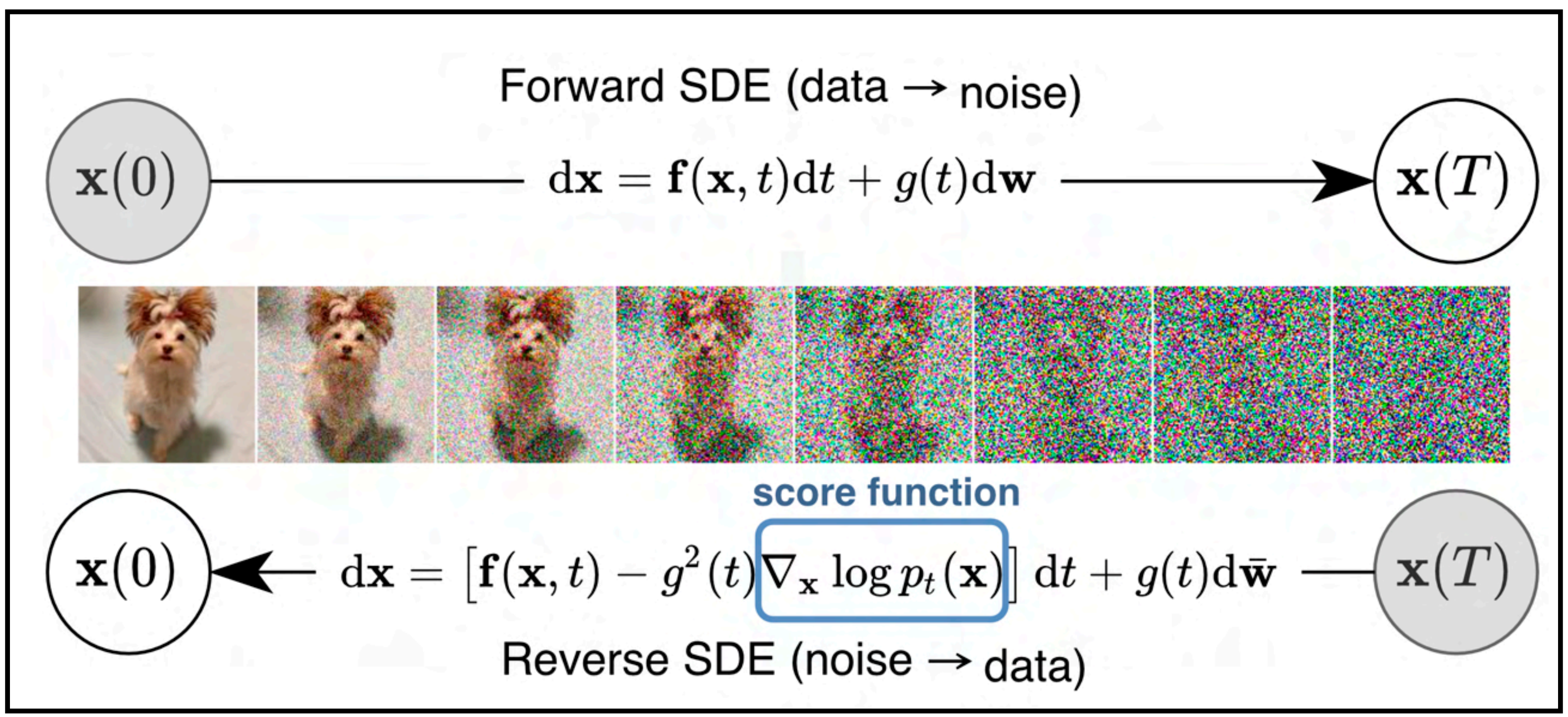
Continuos Diffusion Model

Probability density of x(t)

Reverse SDE: $$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$$

Score function

Reverse of a diffusion process is also a diffusion

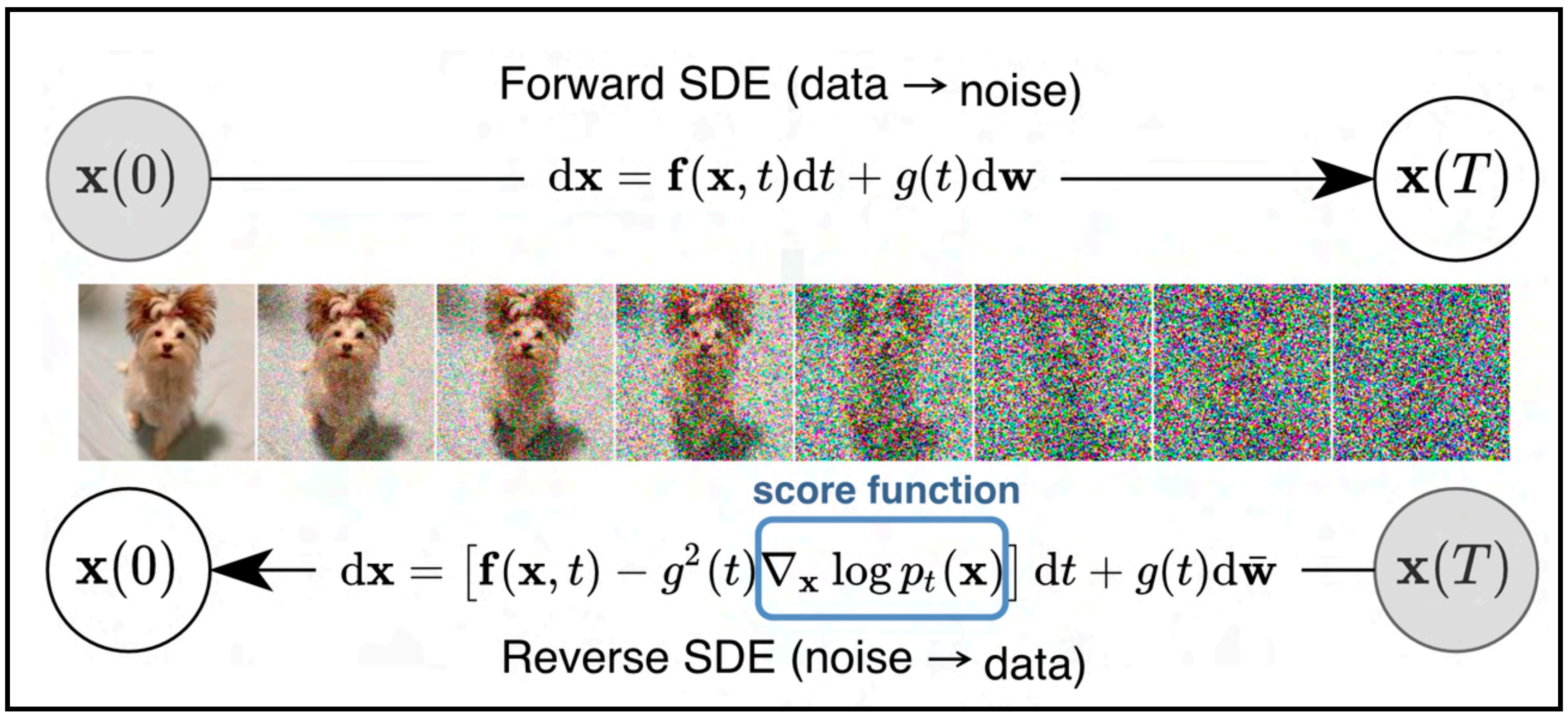


Continuos Diffusion Model

Reverse SDE: $\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2 \boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$

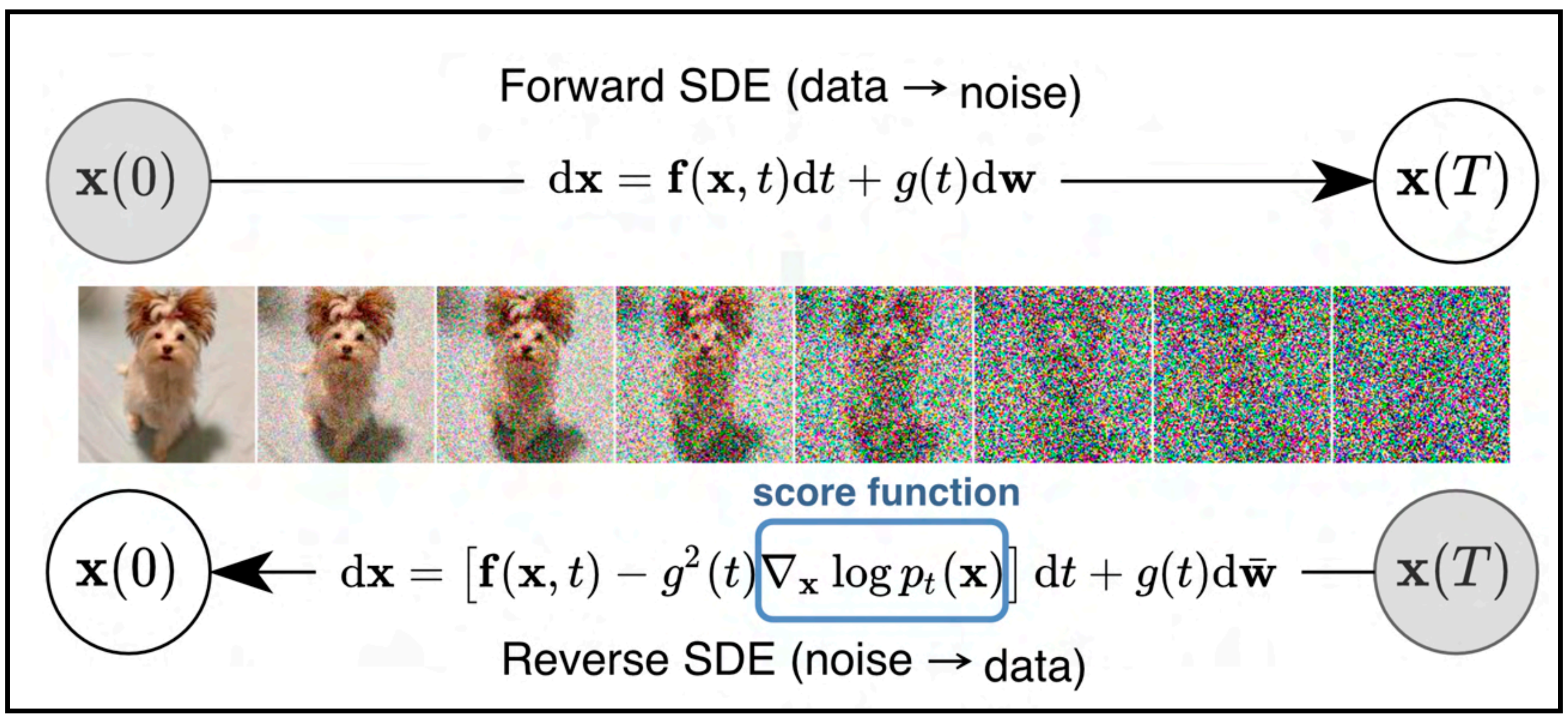
$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\arg\min}\, \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \Big[\big\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \boxed{\nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))} \big\|_2^2 \Big] \Big\}$$

Learn to approximate score function with neural network



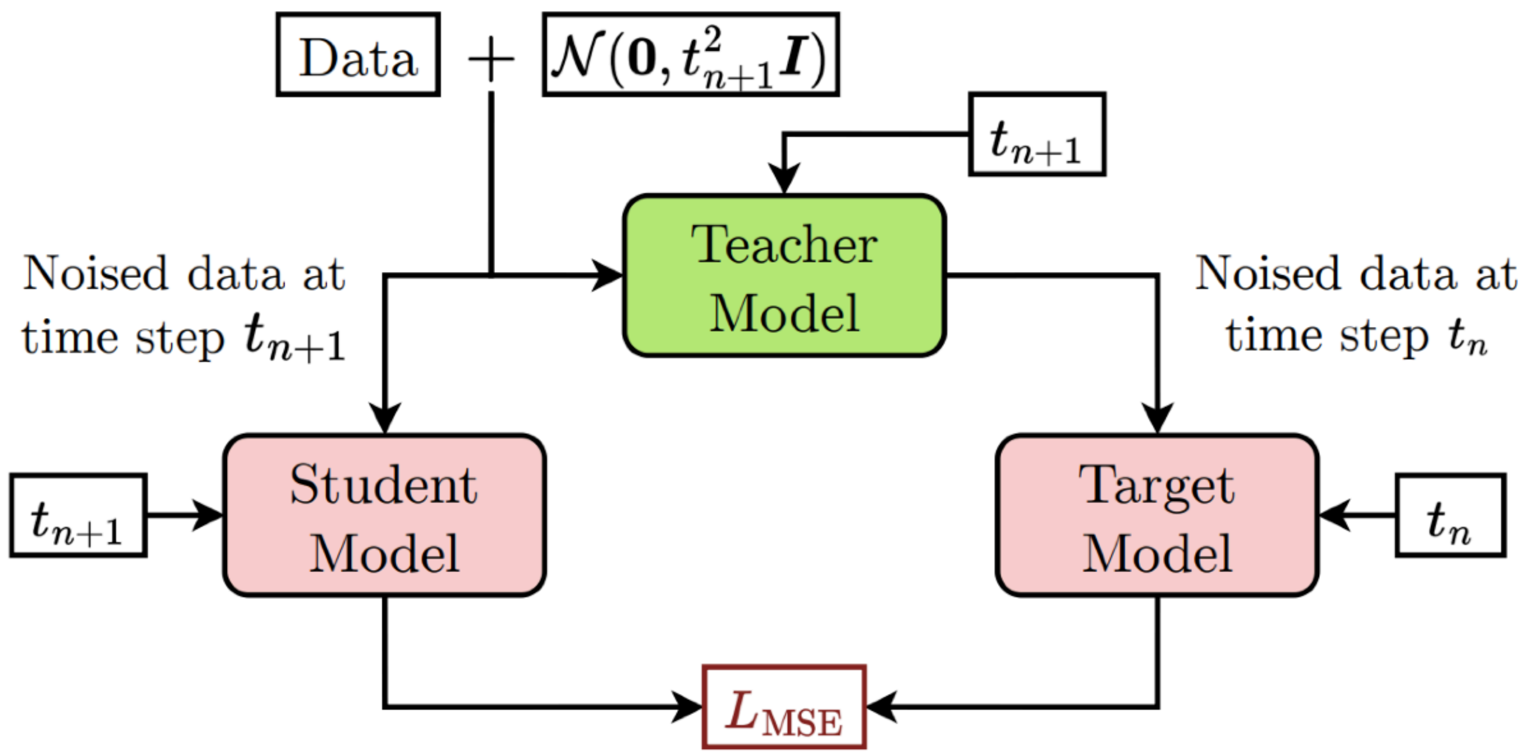
Continuos Diffusion Model

Once trained: Sample latent space and numerically solve SDE to transport to data space

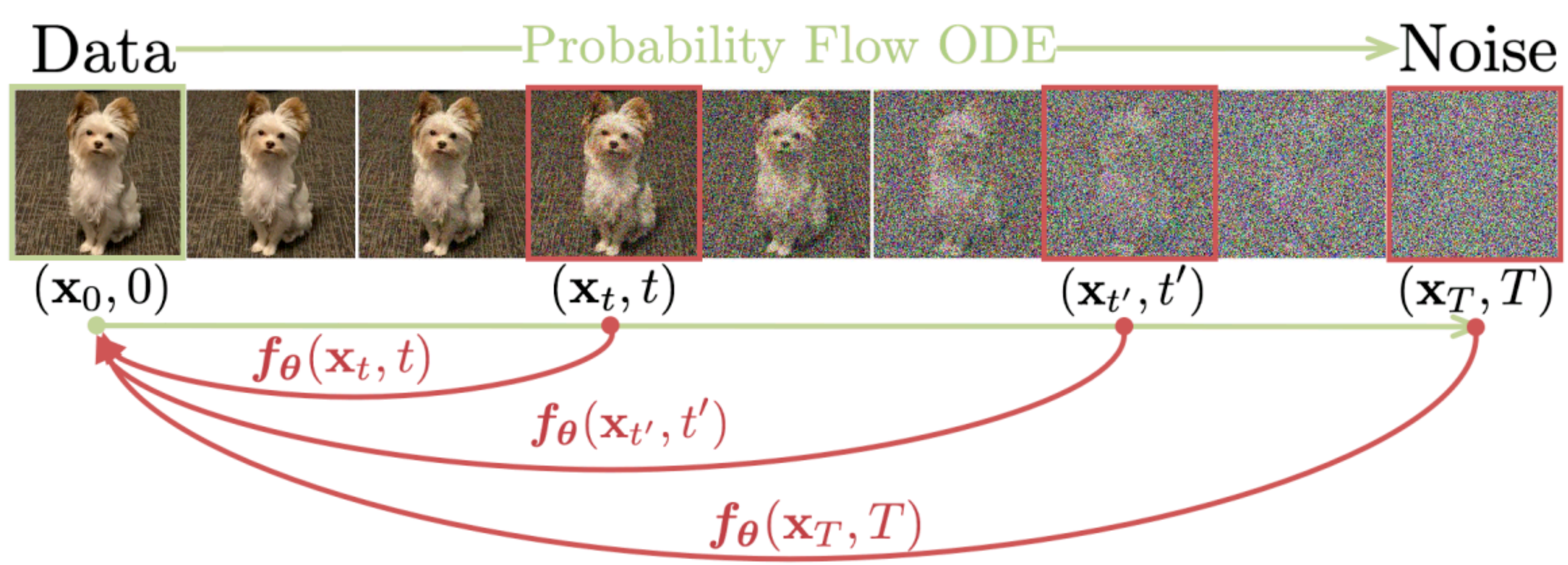


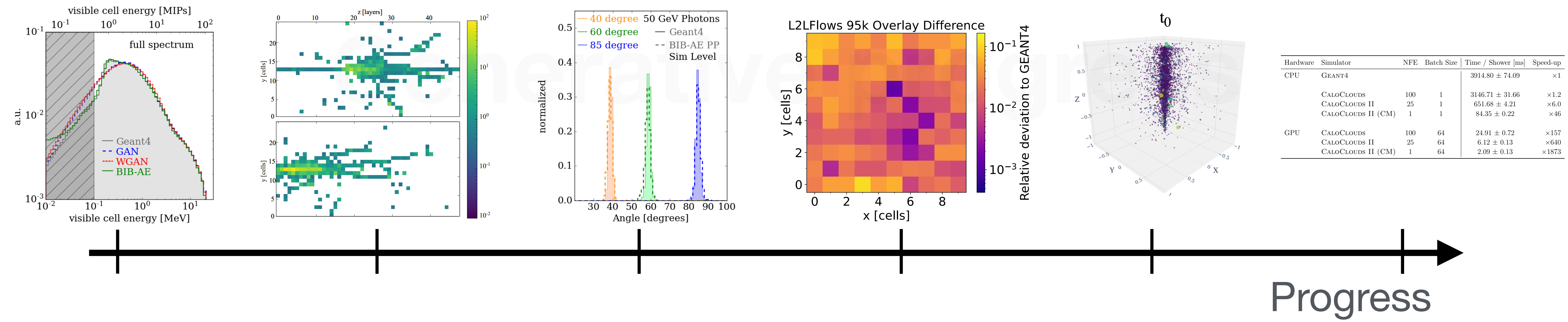
Great generative quality, but tends to be slow. We need to do more

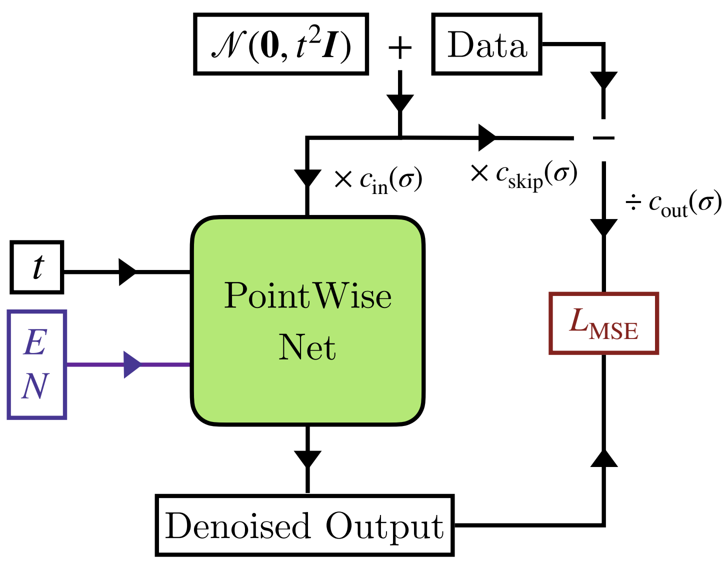
Consistency Distillation

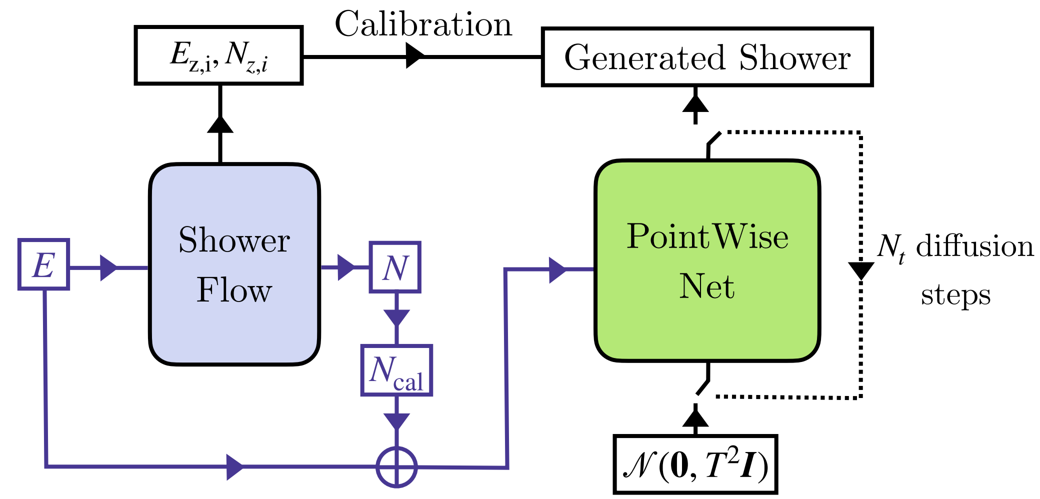


Speed up by training a model to allow single step generation

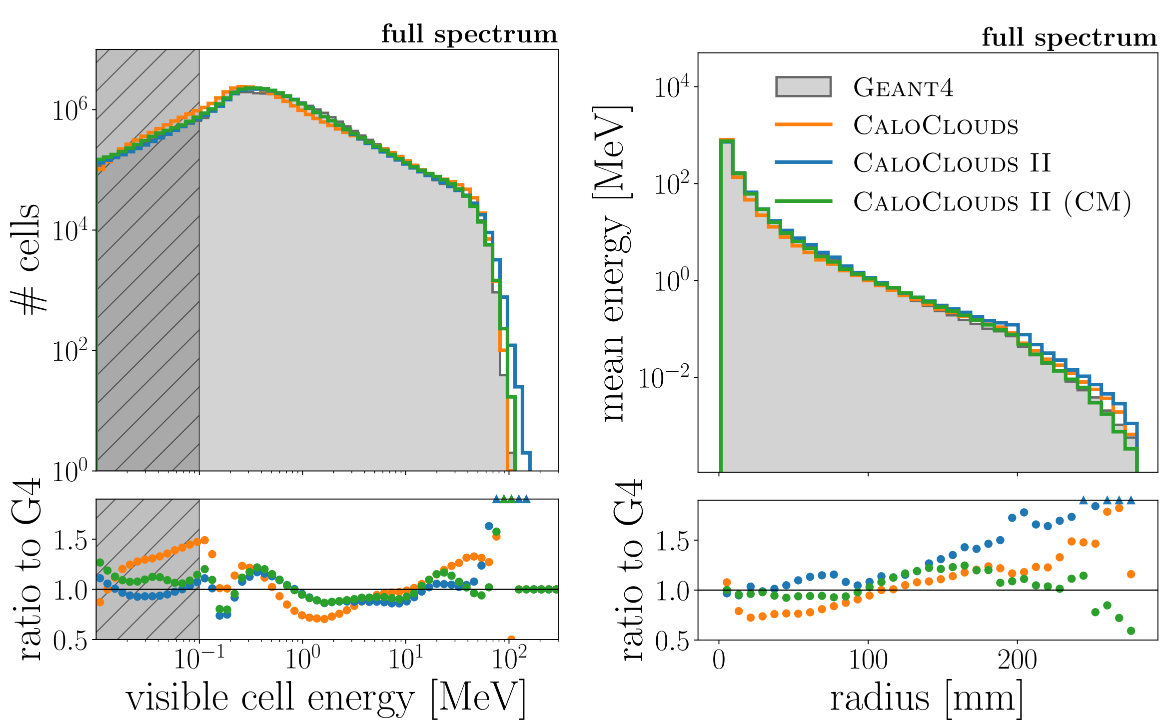








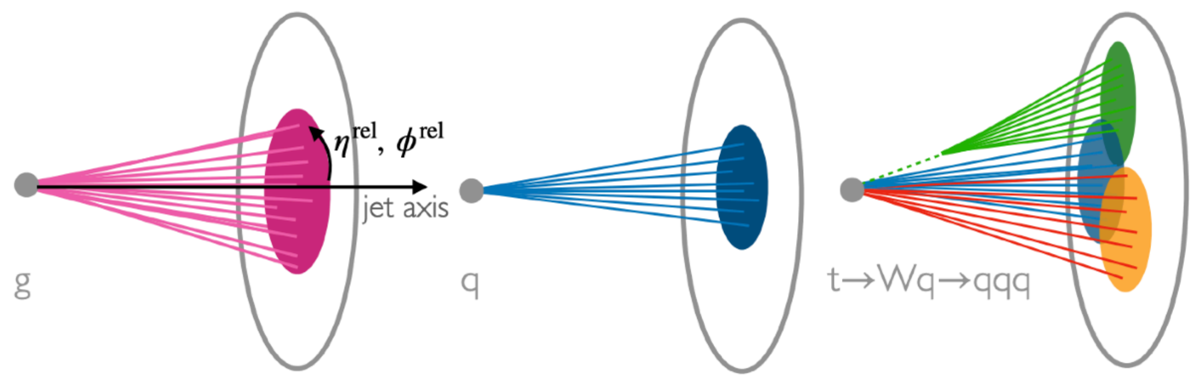
Use continuous time diffusion and consistency distillation: Better quality and faster



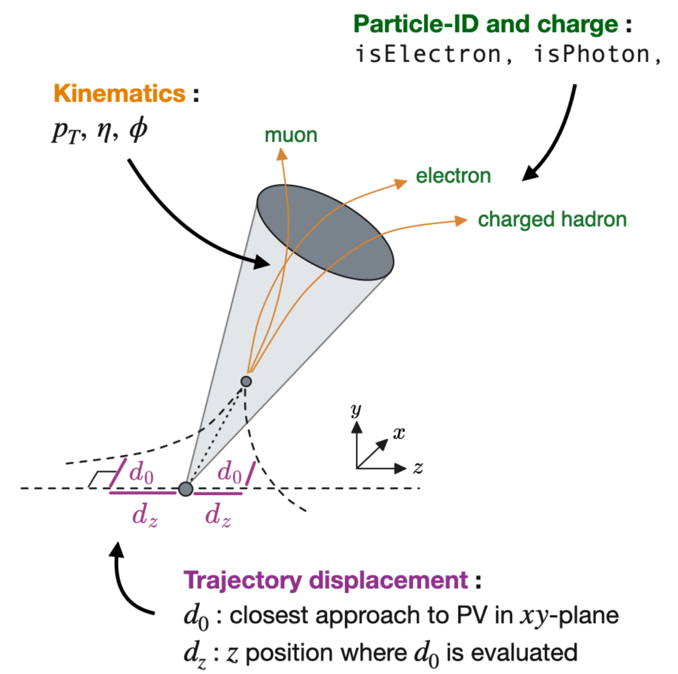
Hardware	Simulator	NFE	Batch Size	Time / Shower [ms]	Speed-up
CPU	Geant4			3914.80 ± 74.09	×1
	CaloClouds	100	1	3146.71 ± 31.66	×1.2
	CaloClouds II	25	1	651.68 ± 4.21	×6.0
	CaloClouds II (CM)	1	1	84.35 ± 0.22	×46
GPU	CaloClouds	100	64	24.91 ± 0.72	×157
	CaloClouds II	25	64	6.12 ± 0.13	×640
	CaloClouds II (CM)	1	64	2.09 ± 0.13	×1873

Buhmann, .., **GK** et al 2309.05704

Aside: Beyond Showes



JetNet

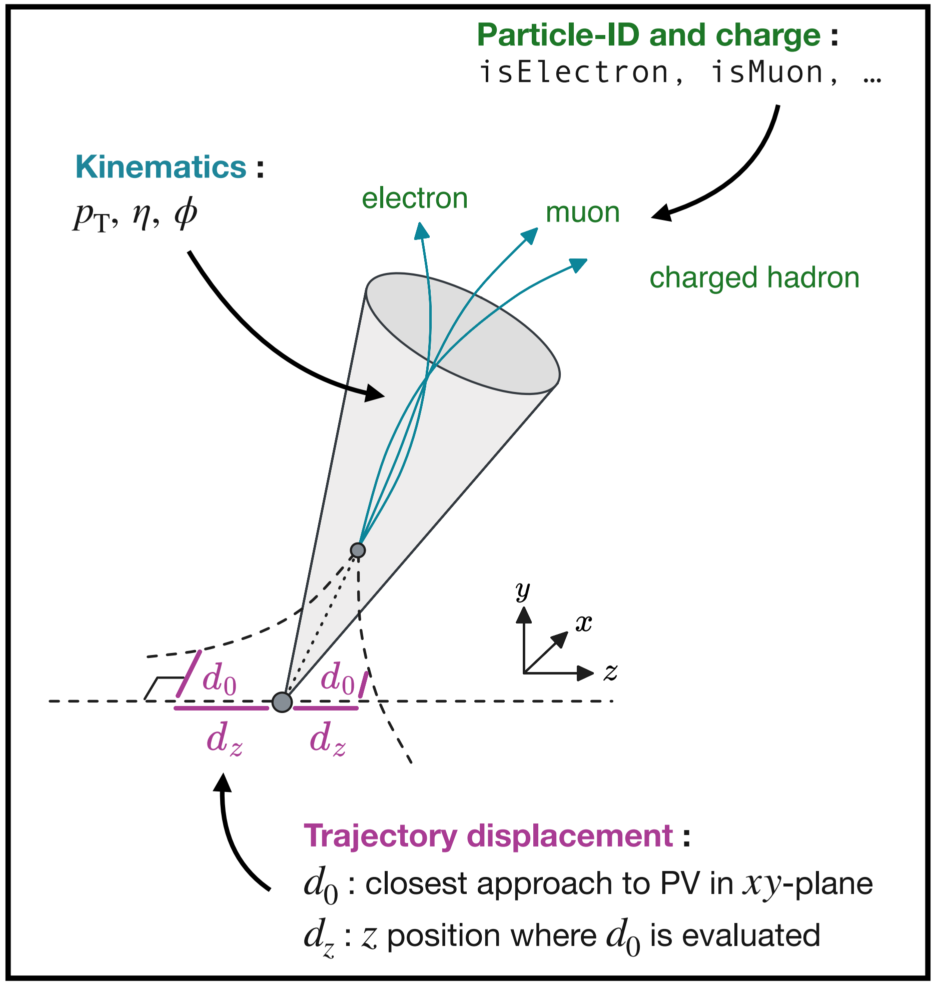


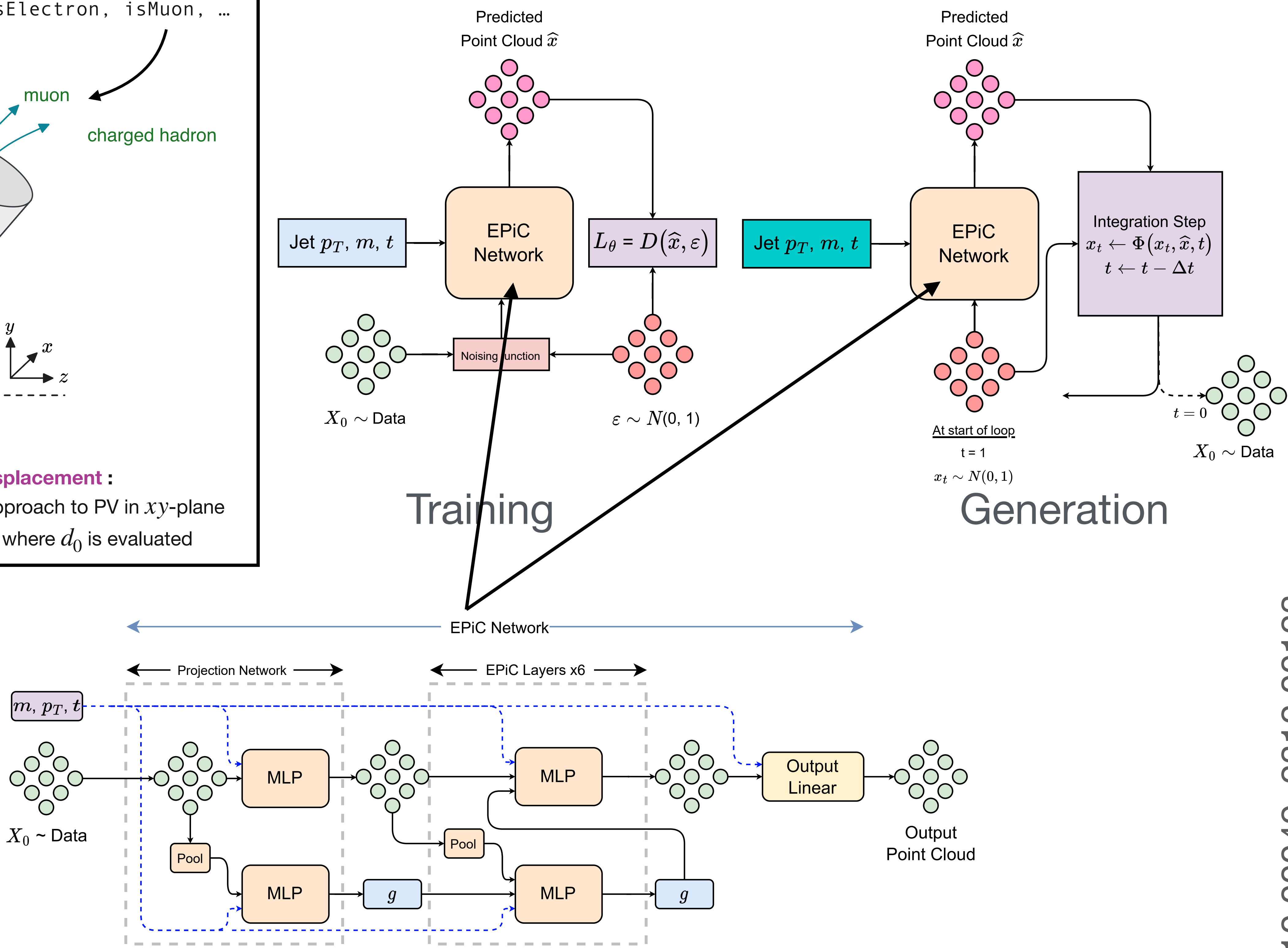
JetClass

	JetNet [3]	**JetClass [1]**
Jet types	5 types	10 types (several decay channels for top and H jets)
Dataset size	180 thousand jets per class	12.5 million jets per class (70x more than JetNet)
Features	Kinematics	Kinematics, Particle-ID and charge, trajectory displacement

2106.11535, 2202.03772

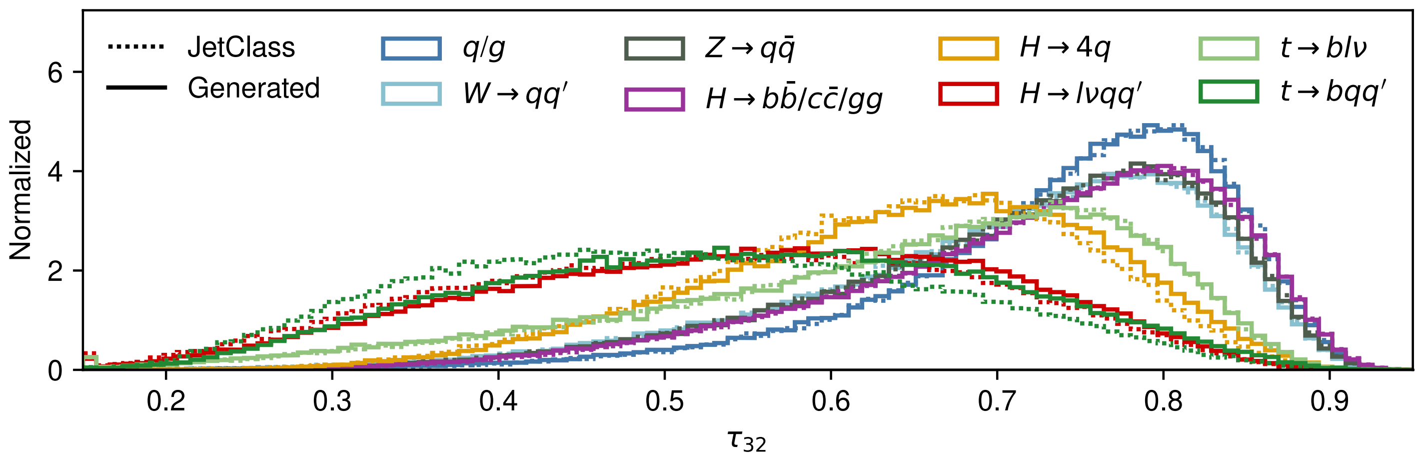
Aside: Beyond showers



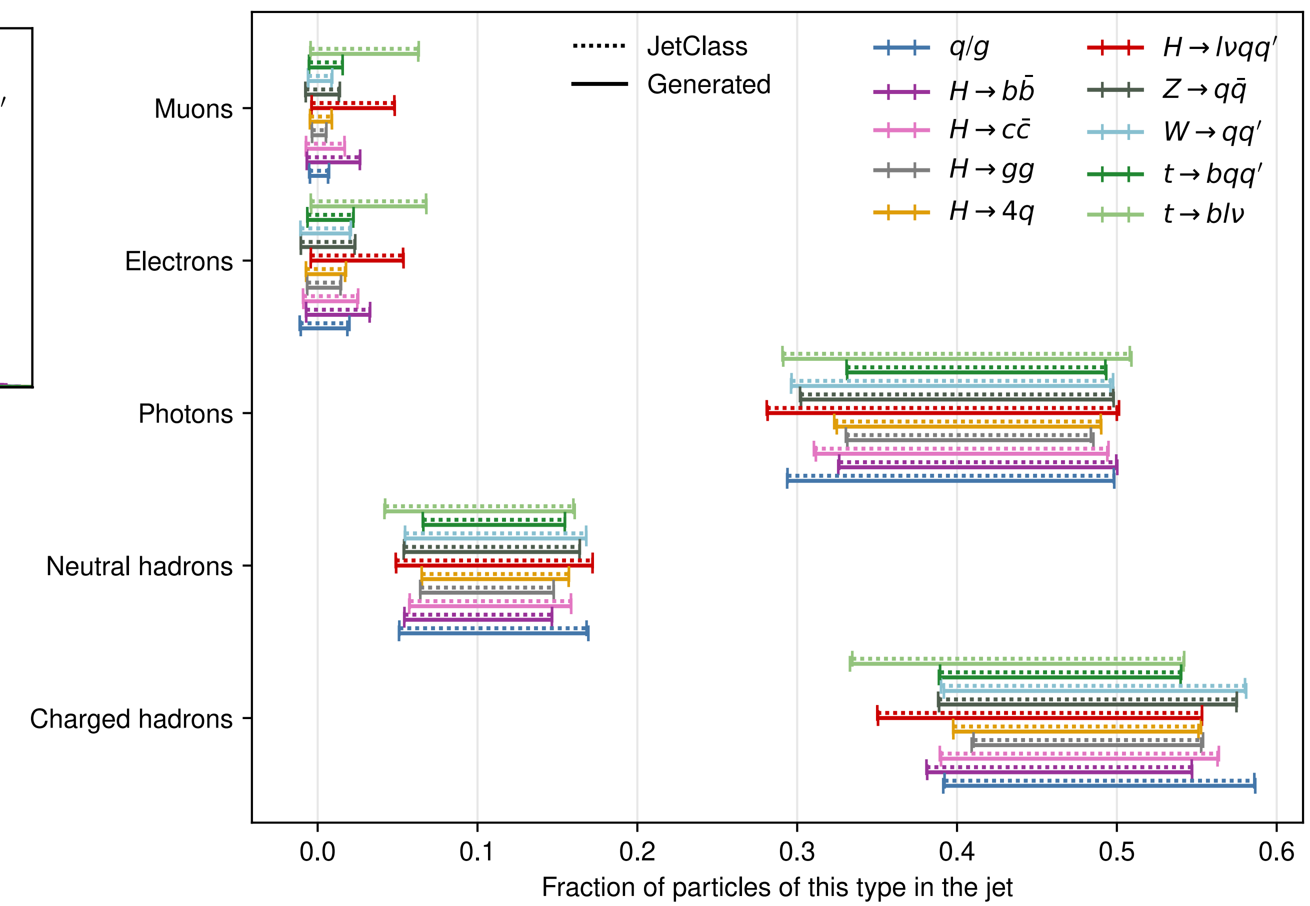


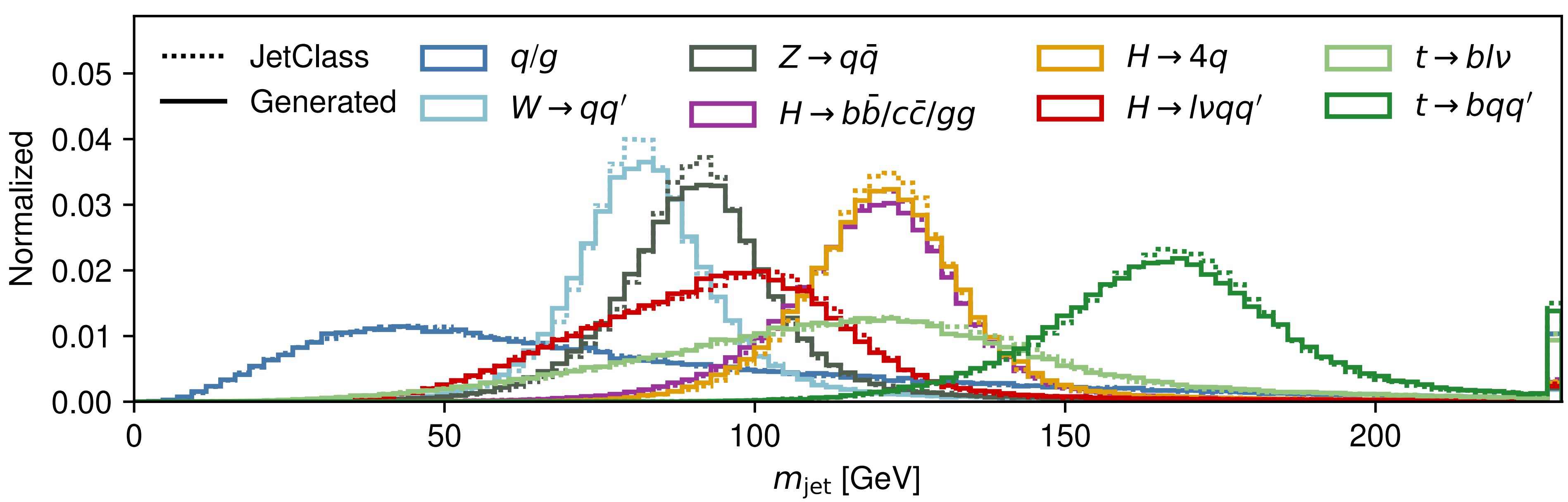
2310.00049, 2312.00123

Aside: Beyond showers



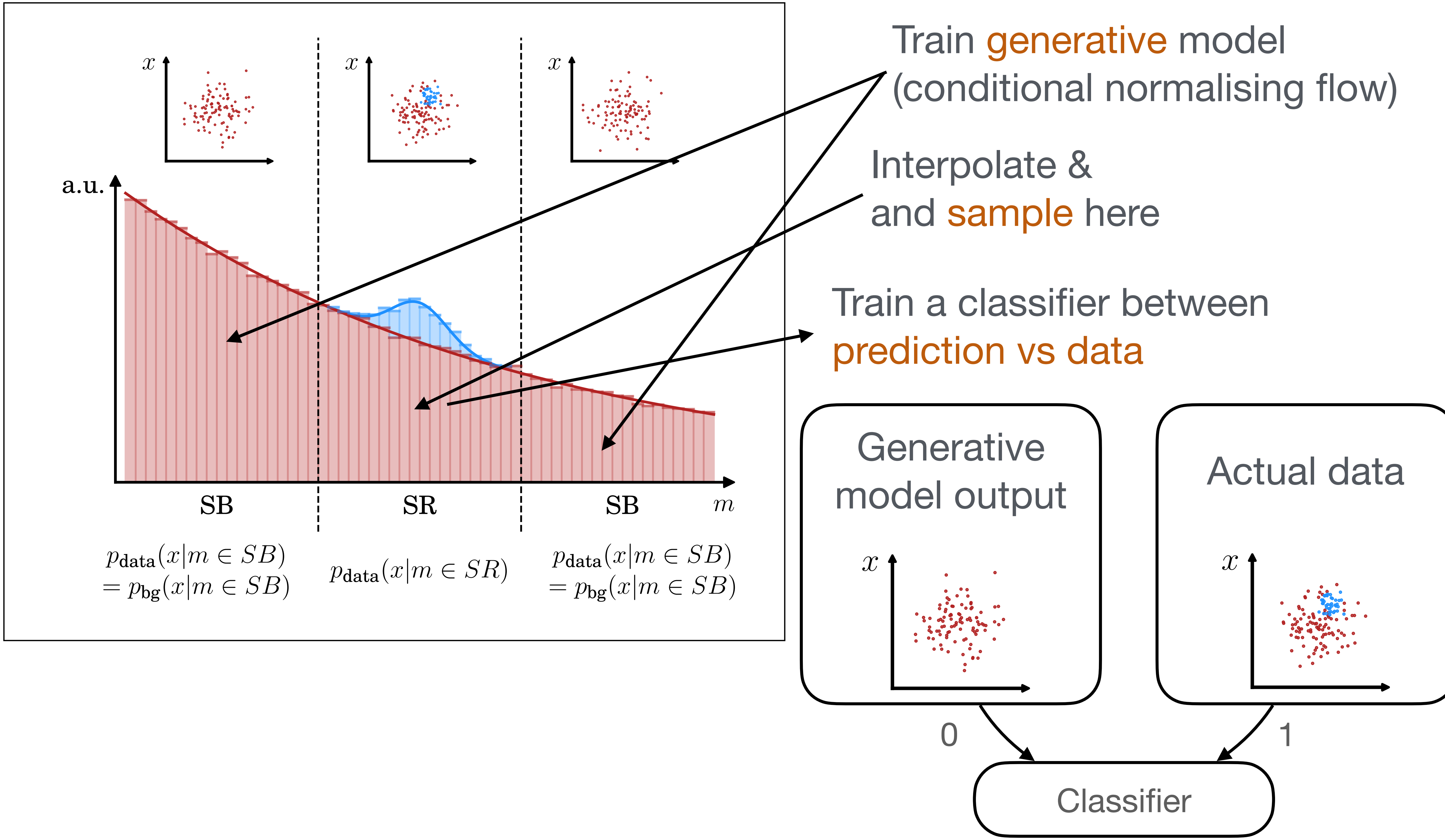
Can use similar setup (flow matching) also for better jet constituent simulation





2310.00049, 2312.00123

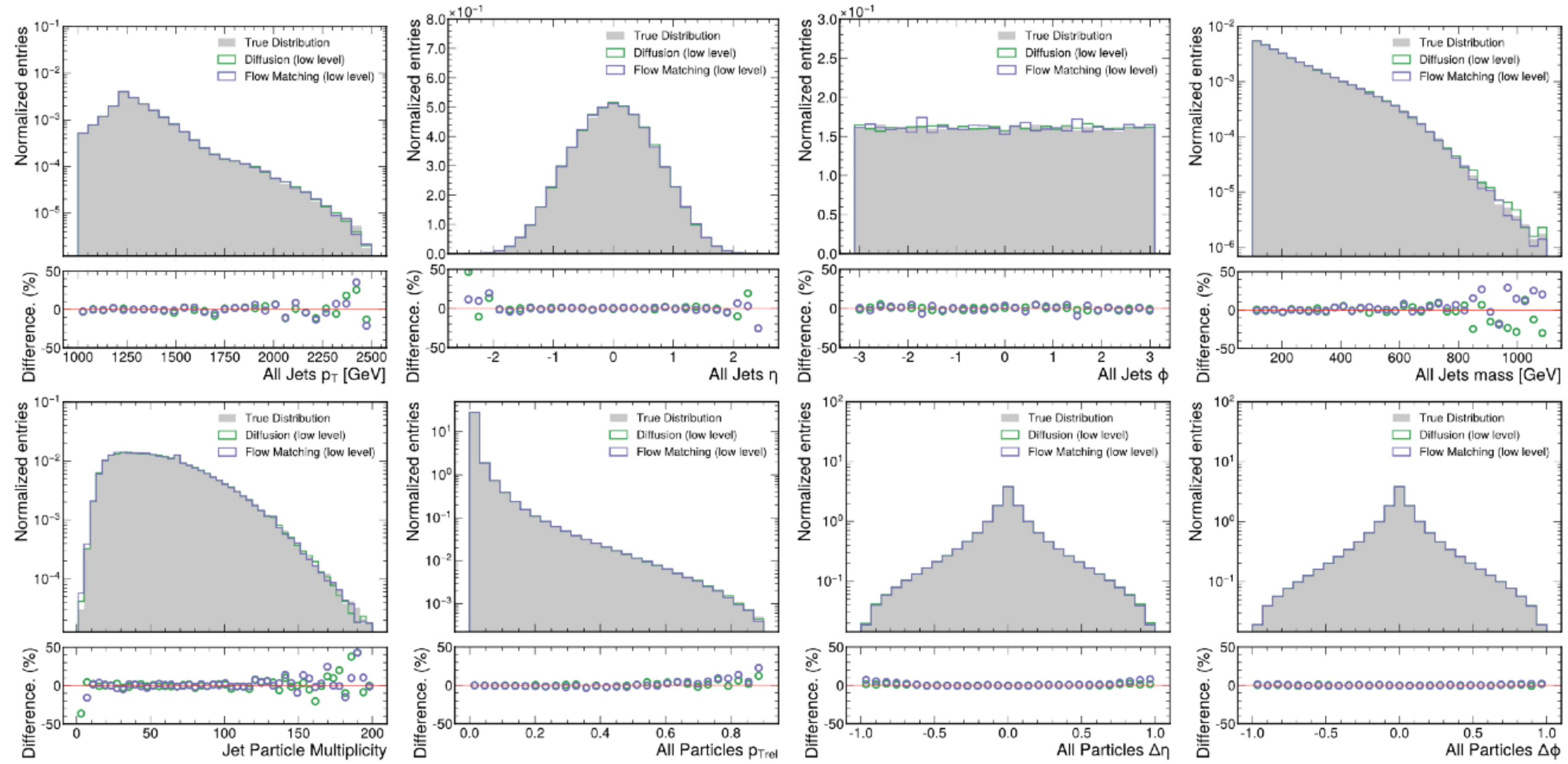
Application: CATHODE

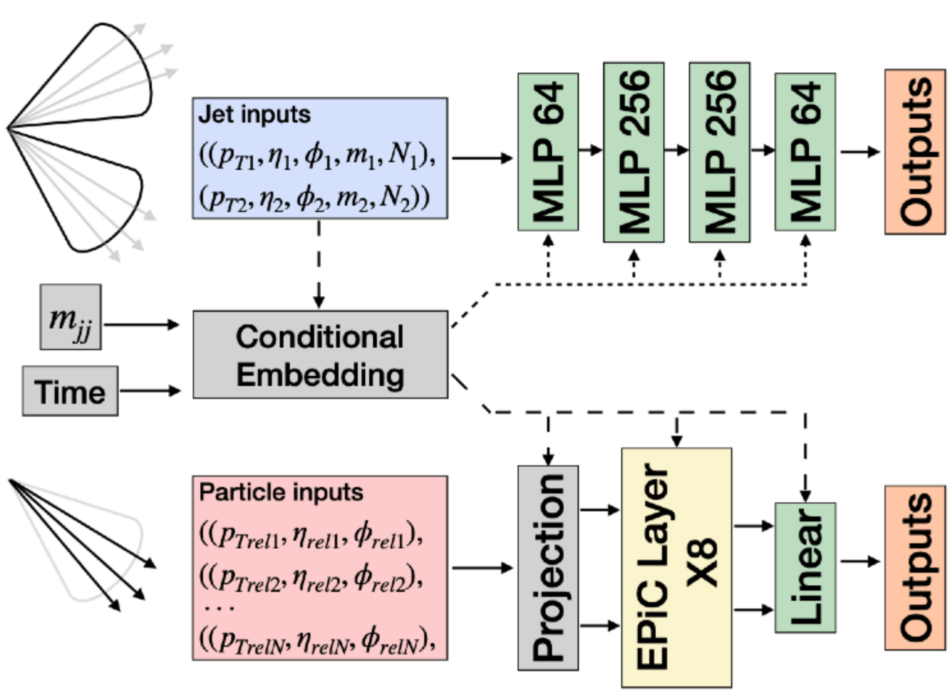


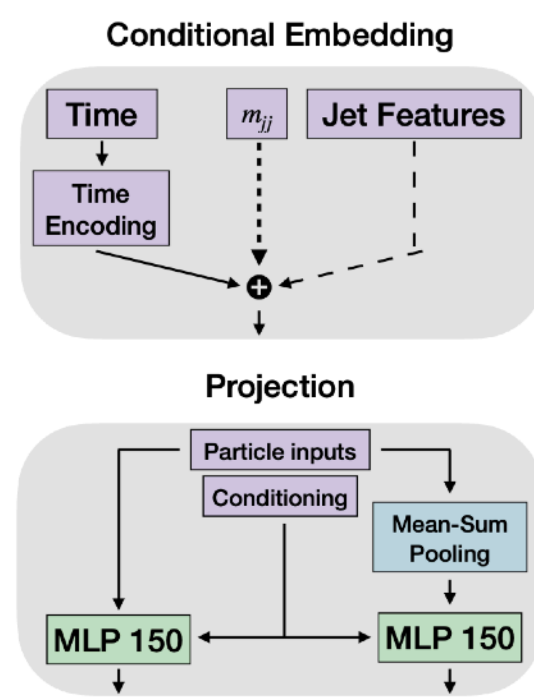
GK, Nachmann, Shih et al 2101.08320; Hallin, .., **GK** et al 2109.00546;

Application: CATHODE

Using all low-level features in-principle includes all properties







Made possible by recent progress in point cloud generative models
(see Vinny's talk for more details)

(Classifier is a transformer)

Buhmann, .., **GK**, Mikuni, et al 2310.06897;

Closing

Better generativer architectures steadily improve simulation of particle showers

Better scaling for large detectors via point cloud approach:

- Can also use for learning jet constituents directly (c.f. “ML at HEP” anomaly talk)

Next big challenge:
Scaling and integration

Fast Calorimeter Simulation Challenge 2022

View on GitHub

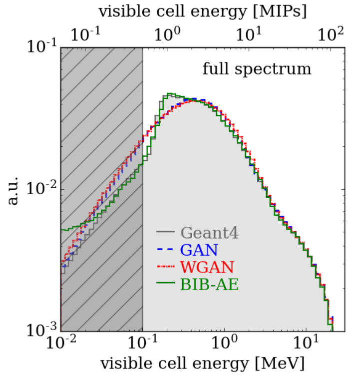
Welcome to the home of the first-ever Fast Calorimeter Simulation Challenge!

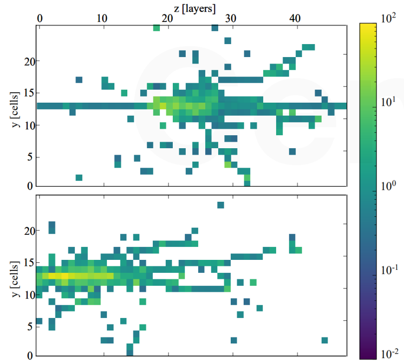
The purpose of this challenge is to spur the development and benchmarking of fast and high-fidelity calorimeter shower generation using deep learning methods. Currently, generating calorimeter showers of interacting particles (electrons, photons, pions, ...) using GEANT4 is a major computational bottleneck at the LHC, and it is forecast to overwhelm the computing budget of the LHC experiments in the near future. Therefore there is an urgent need to develop GEANT4 emulators that are both fast (computationally lightweight) and accurate. The LHC collaborations have been developing fast simulation methods for some time, and the hope of this challenge is to directly compare new deep learning approaches on common benchmarks. It is expected that participants will make use of cutting-edge techniques in generative modeling with deep learning, e.g. GANs, VAEs and normalizing flows.

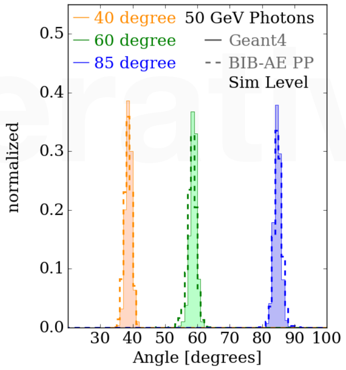
This challenge is modeled after two previous, highly successful data challenges in HEP – the top tagging community challenge and the LHC Olympics 2020 anomaly detection challenge.

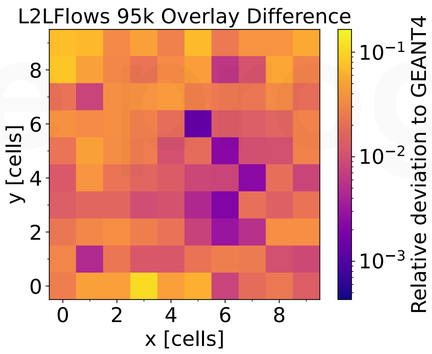
3 datasets of increasing complexity:
DS1: Up to 533 voxels (ATLAS)
DS2: 6480 voxels
DS3: 40500 voxels

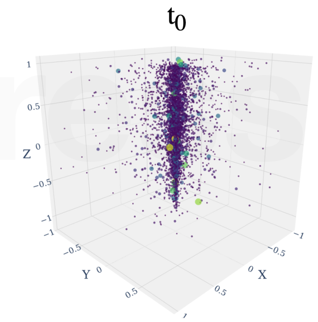
Thank you!











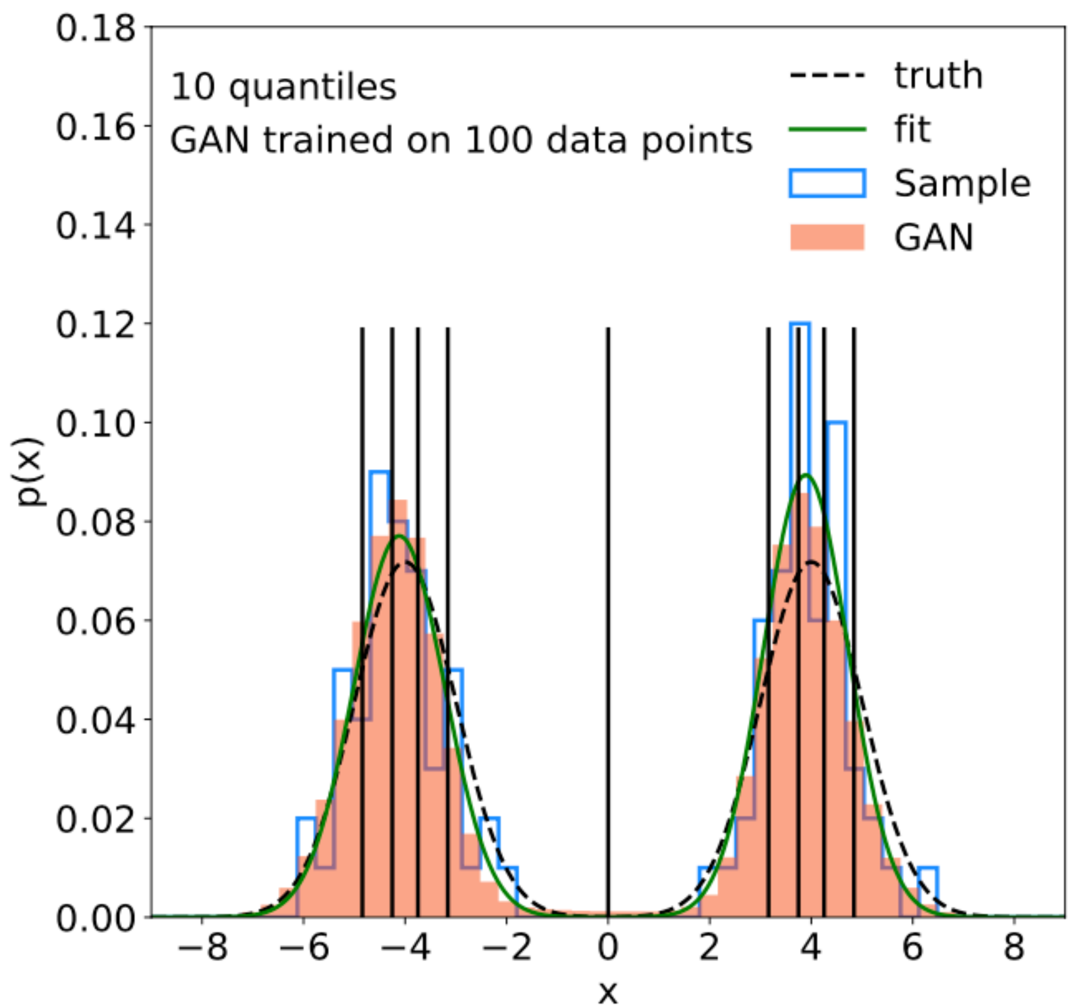
Hardware	Simulator	NFE	Batch Size	Time / Shower [ms]	Speed-up
CPU	GEANT4			3914.80 ± 74.09	×1
	CALOCLOUDS	100	1	3146.71 ± 31.66	×1.2
	CALOCLOUDS II	25	1	651.68 ± 4.21	×6.0
	CALOCLOUDS II (CM)	1	1	84.35 ± 0.22	×46
GPU	CALOCLOUDS	100	64	24.91 ± 0.72	×157
	CALOCLOUDS II	25	64	6.12 ± 0.13	×640
	CALOCLOUDS II (CM)	1	64	2.09 ± 0.13	×1873

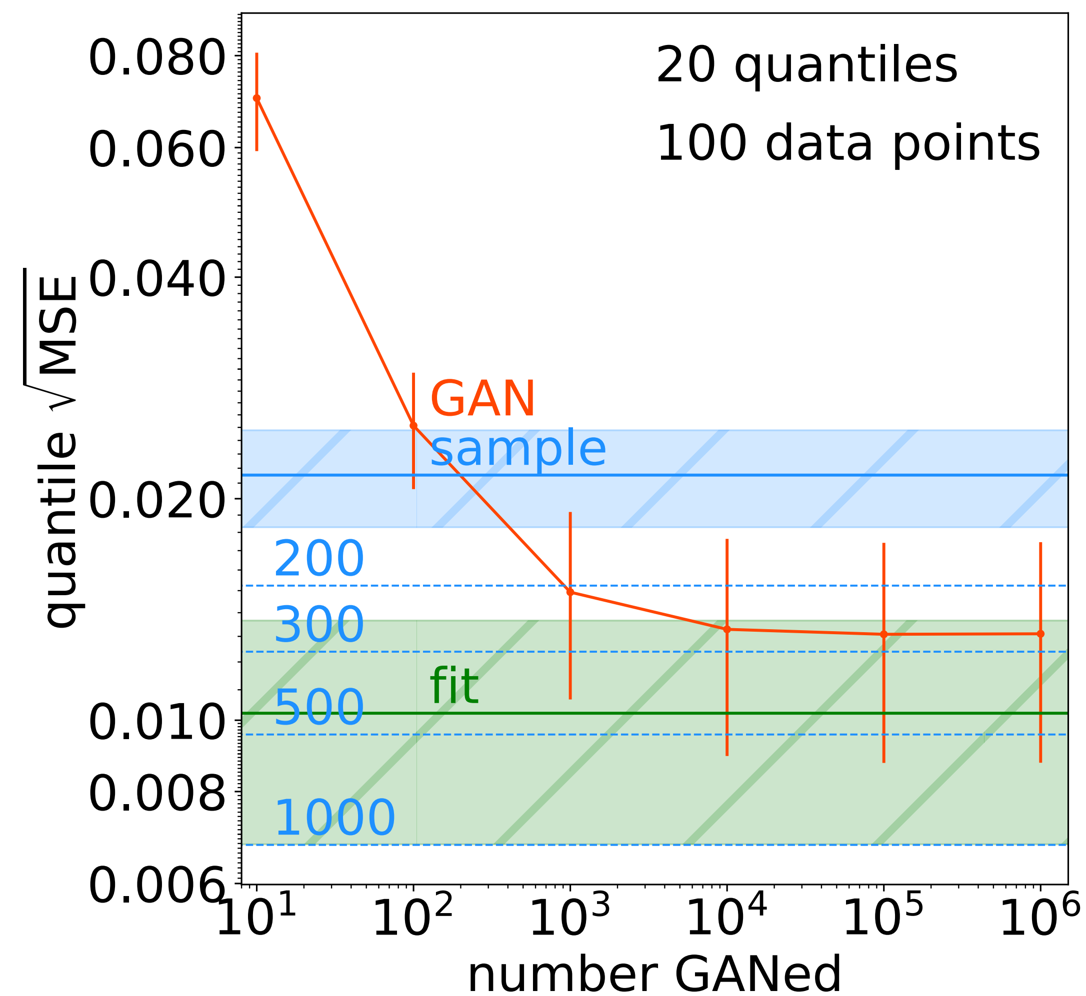
Backup

Statistics

If we train a generator on N data points, and use it to produce M>>N examples, what is the statistical power of the M points?

Compare (known) truth distribution to sample and oversampled data from GAN

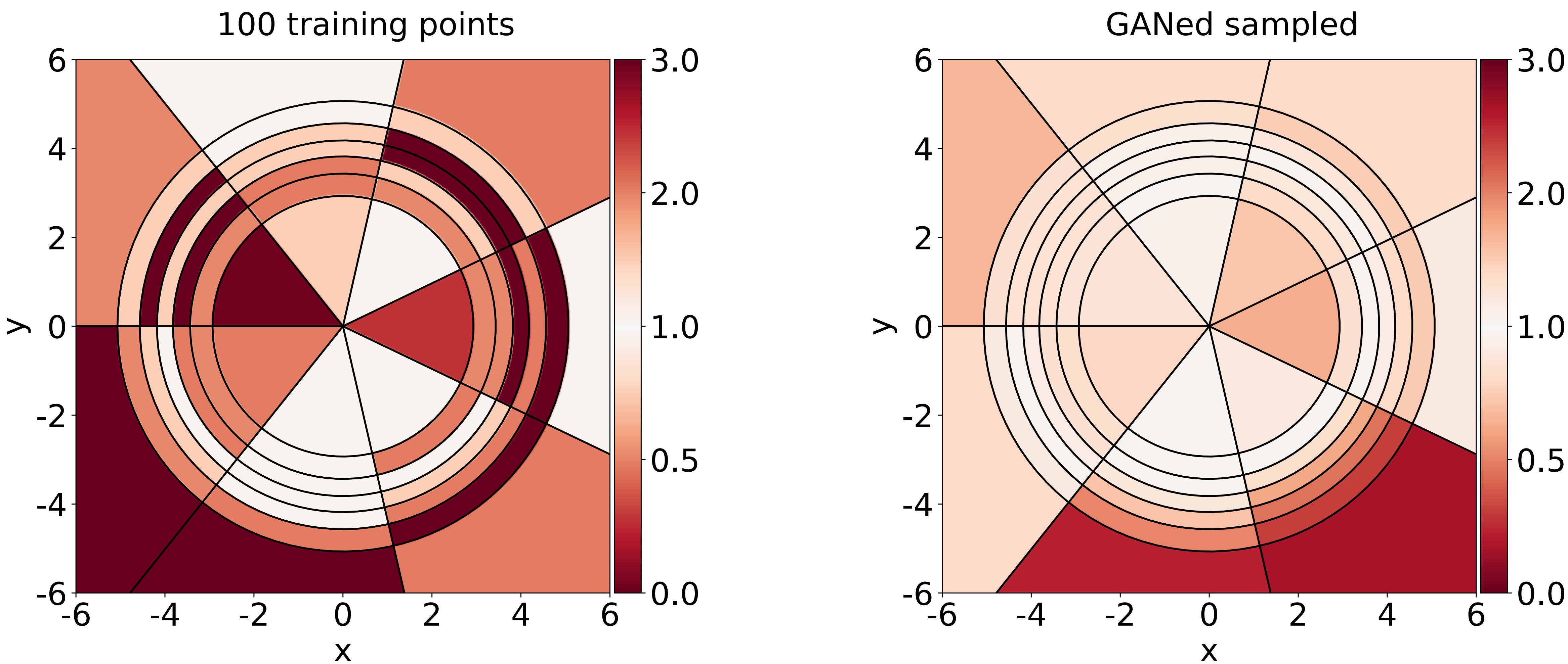




$$\mathrm{MSE} = \frac{1}{N_{\mathrm{quant}}} \sum_{j=1}^{N_{\mathrm{quant}}} \left(x_j - \frac{1}{N_{\mathrm{quant}}} \right)^2$$

Diefenbacher, .., GK et al 2008.06545

Statistics - 2D

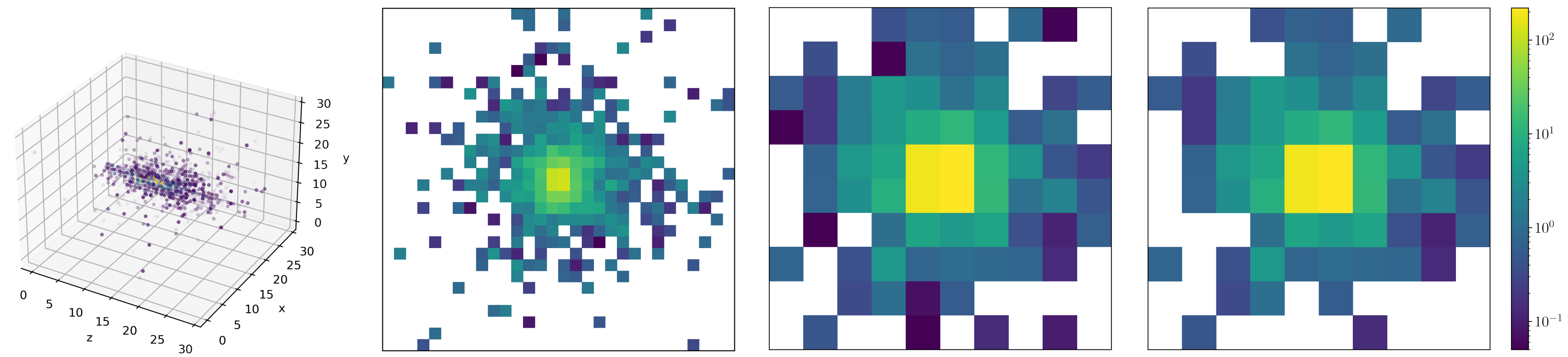


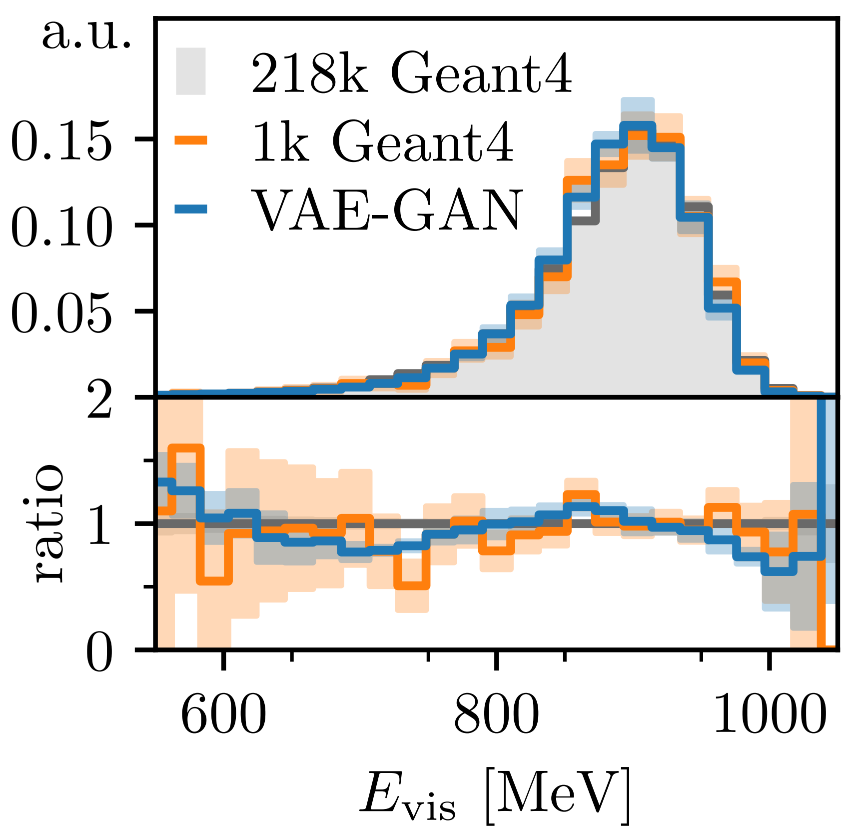
Relative deviation from Gaussian ring distribution

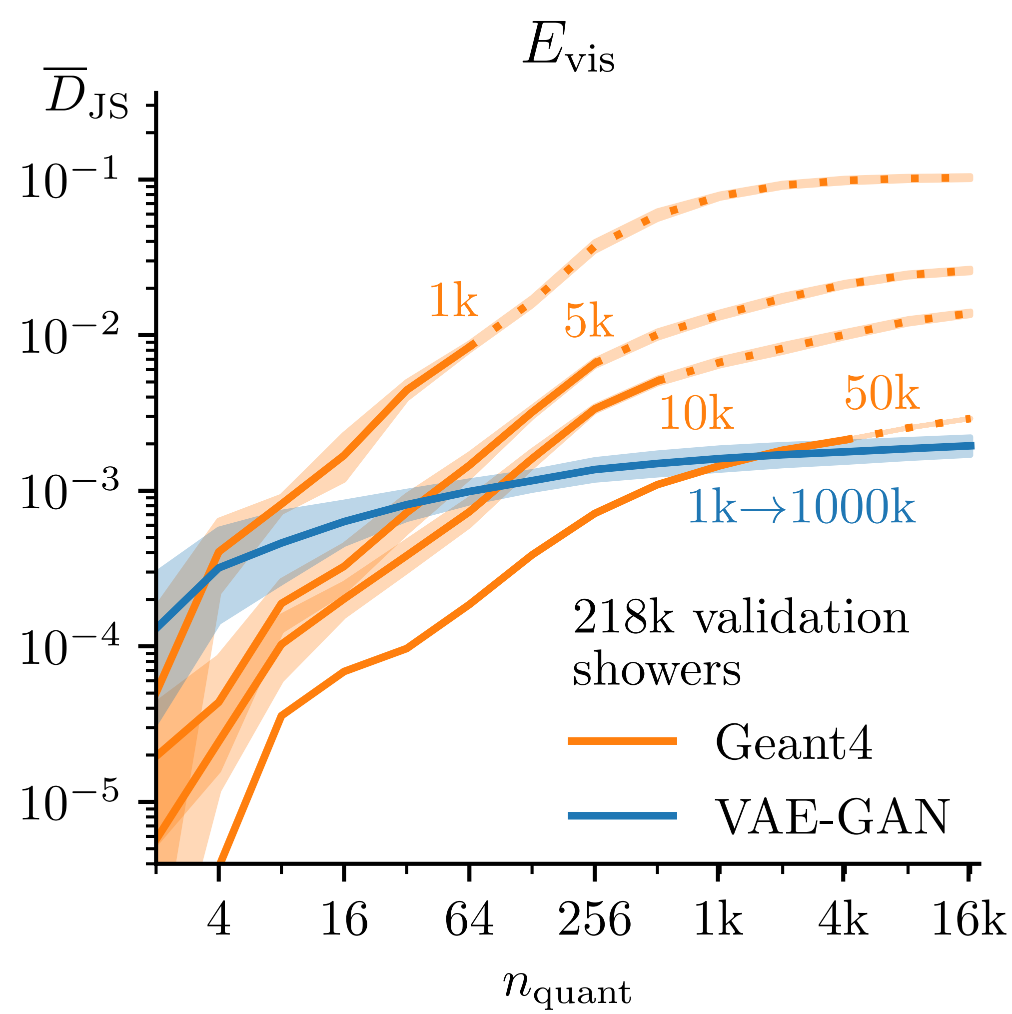
Diefenbacher, .., GK et al 2008.06545

Statistics - Physics

Test the statistical properties of simplified calorimeter showers.







Scaling of difference to ground truth with resolution again better for the generative model.

Bieringer, .., GK et al 2202.07352