Simulation & Generative Models

Gregor Kasieczka
Email: gregor.kasieczka@uni-hamburg.de
Twitter/X: @GregorKasieczka
COFI Winter School 2023

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

KISS

CDCS
CENTER FOR DATA AND COMPUTING
IN NATURAL SCIENCES

FSP CMS

PUNCH 4NFDI

DASHH

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

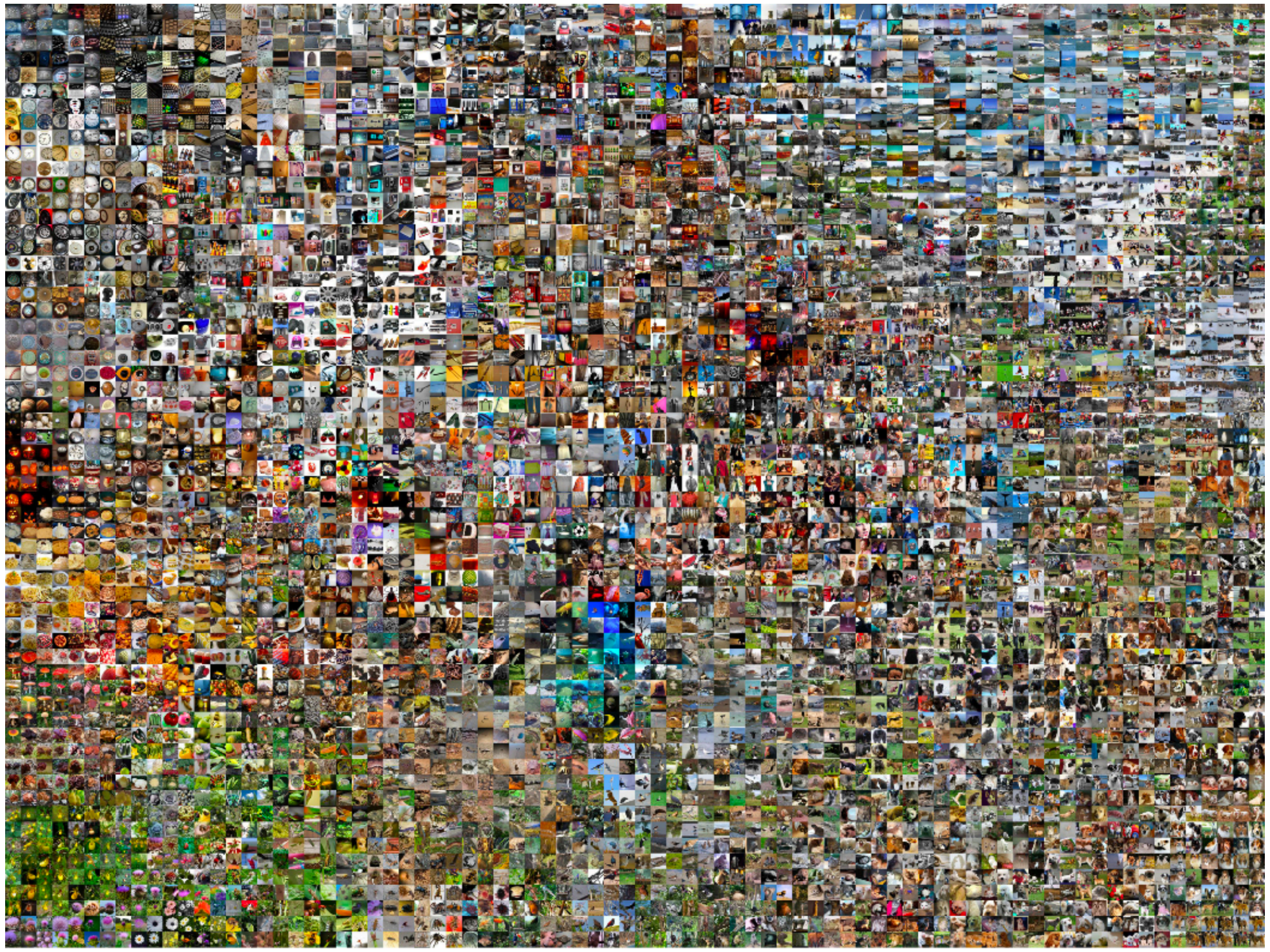
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

PIER
Partnership of Universität Hamburg and DESY

Emmy Noether-Programm
Deutsche Forschungsgemeinschaft DFG

Motivation

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



Want: more data

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

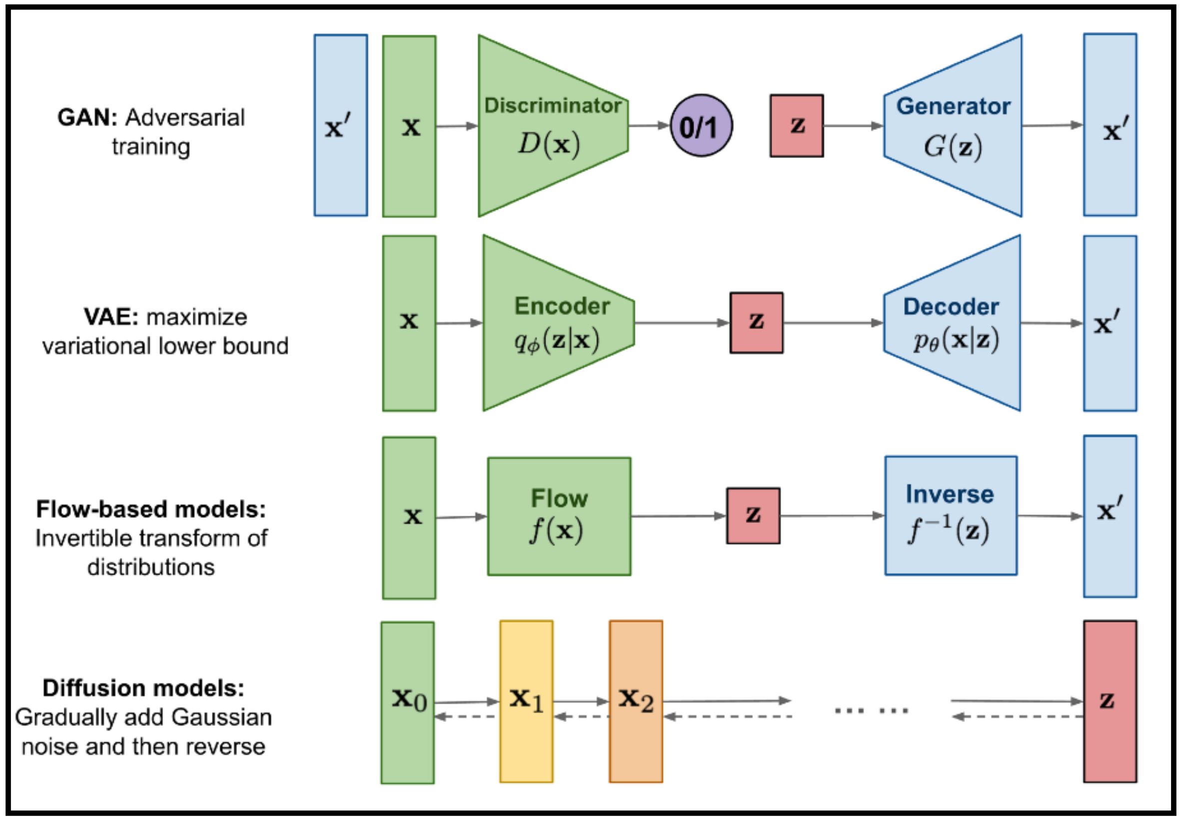
How to encode in neural net?

Uses:

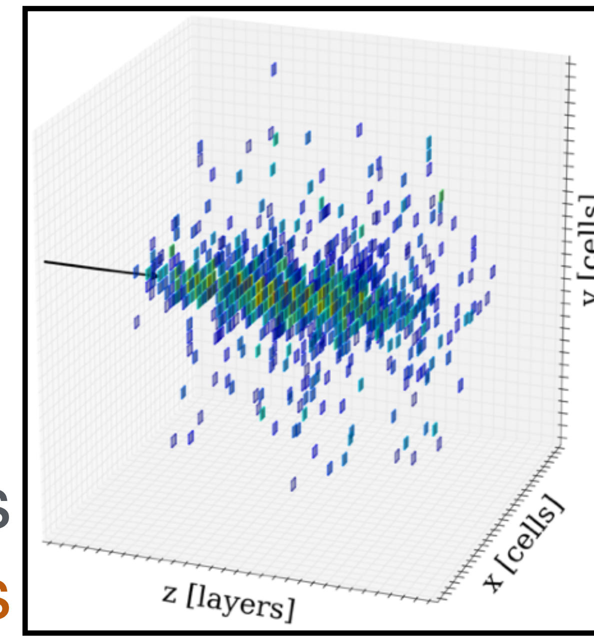
- Detector Simulation
- In-situ background estimation
- Surrogate models
- …

Overview

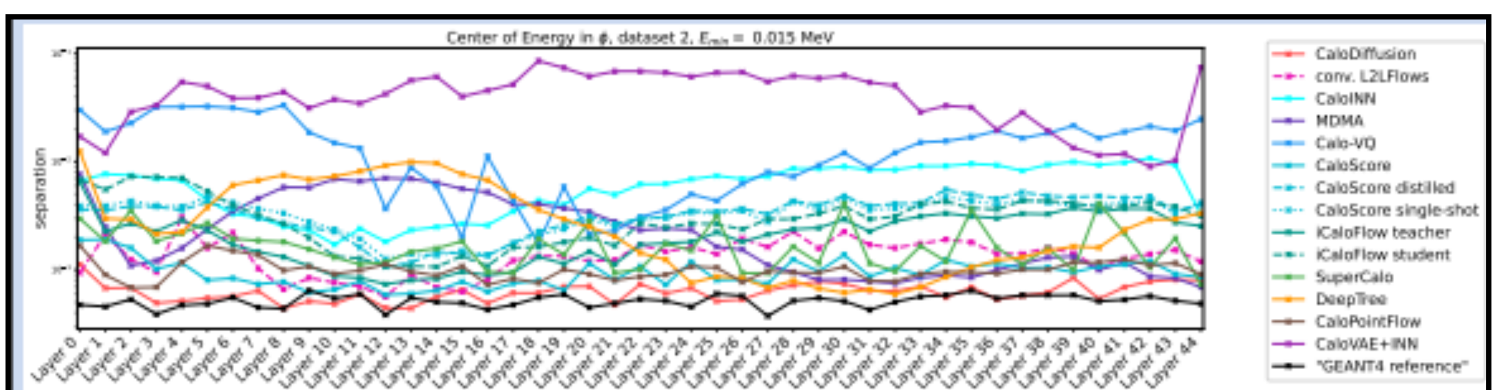
1. Common architectures*
   -> GANs, VAEs, NF today
   -> Diffusion & CNF tomorrow



2. Physics applications



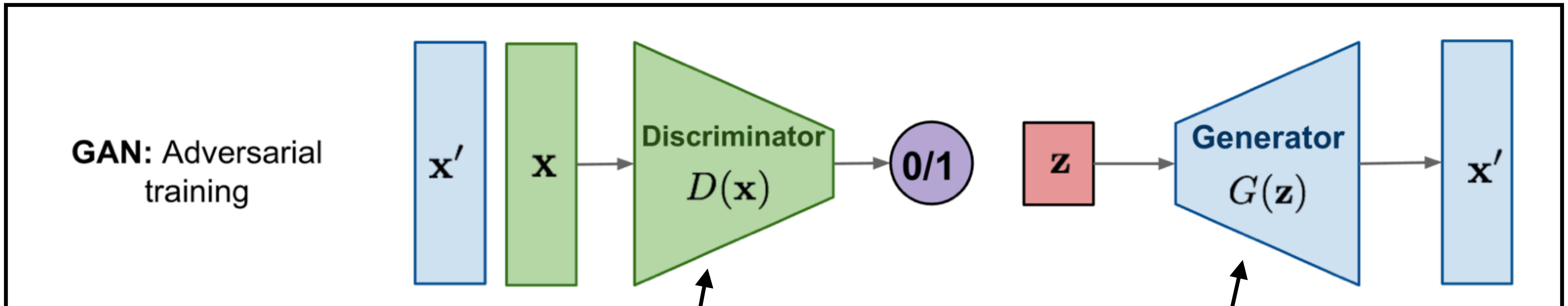
3. Quality metrics



*excluding transformers

Generative Adversarial Networks

Generative Adversarial Networks



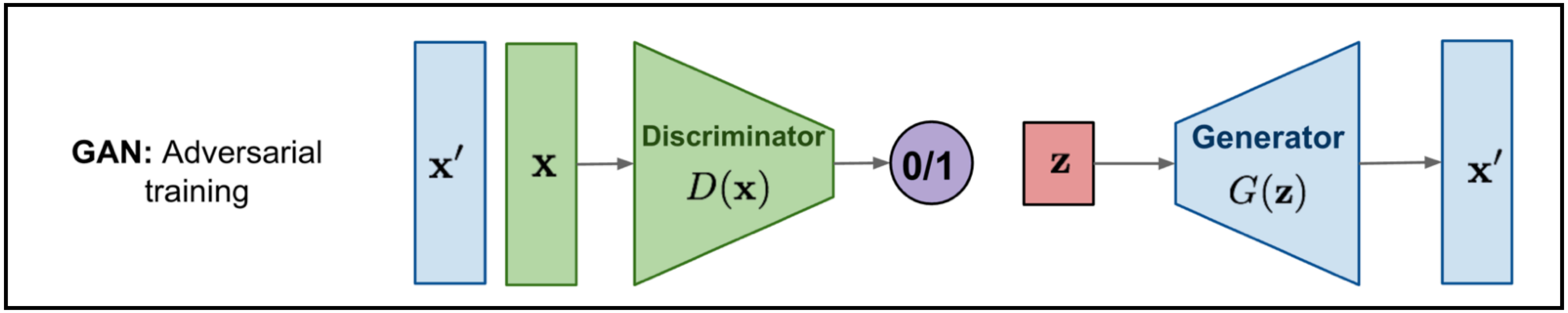
Generative Adversarial Networks (GANs) consist of 2 networks

Maps random noise to realistic examples

Provides feedback on quality of examples

1406.2661
lilianweng.github.io

Generative Adversarial Networks



Training objective:

Binary cross entropy

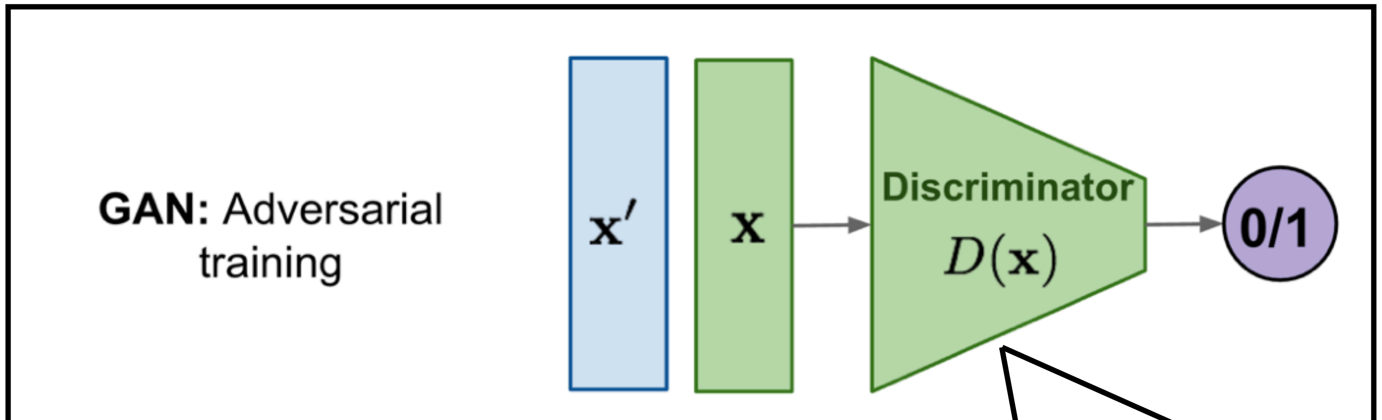
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

True examples

Fake examples

1406.2661

Generative Adversarial Networks



Training objective:

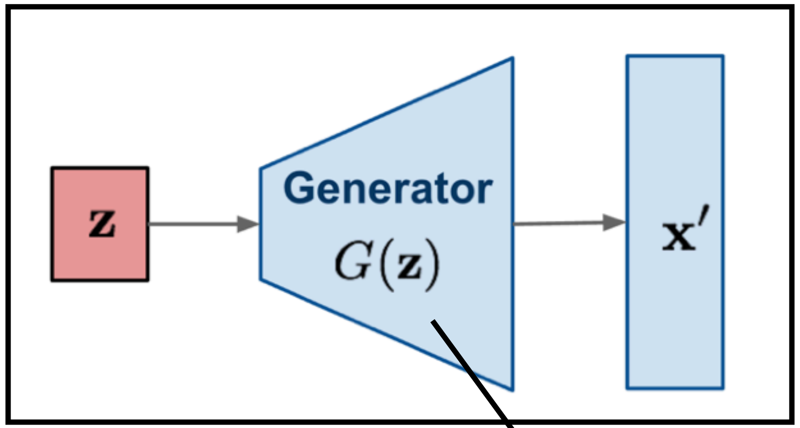
Binary cross entropy

Maximise for discriminator

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

1406.2661

Generative Adversarial Networks



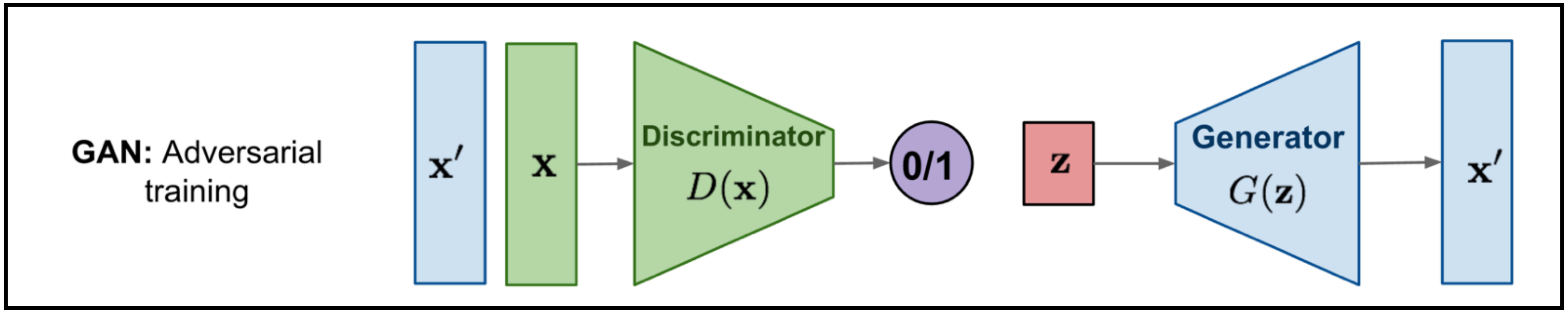
Training objective:
Binary cross entropy

Minimise for generator

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

1406.2661

Generative Adversarial Networks



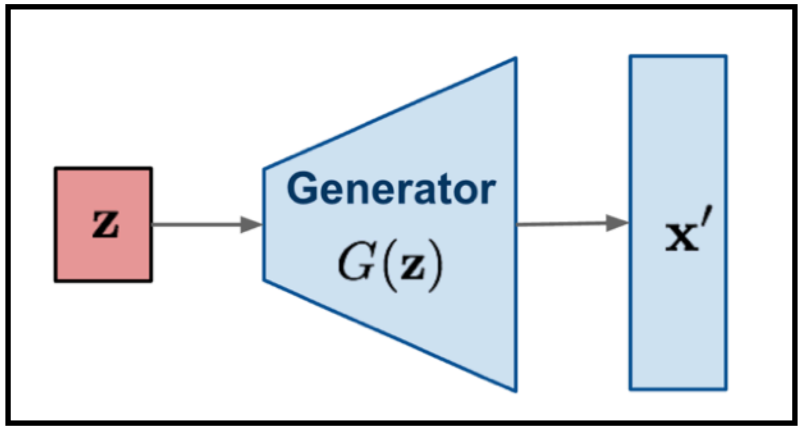
Training objective:
Binary cross entropy

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

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

1406.2661

Generative Adversarial Networks



Training objective:
Binary cross entropy

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

For generation:
Sample from Generator
Discard Discriminator

1406.2661

Comments on GANs

Architecture:

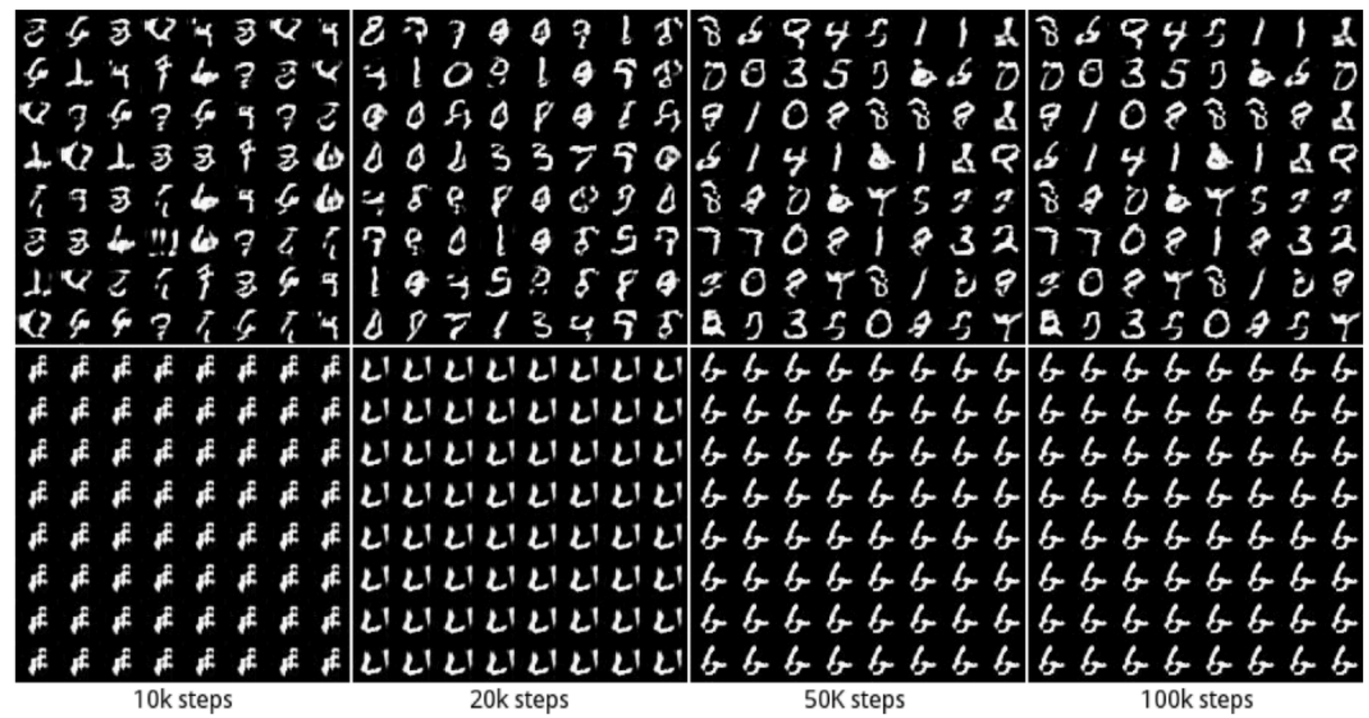
- Low complexity, fast and adaptable

Learning:

- Unstable training
- Matching of generator/discriminator (vanishing gradients)
- Mode collapse
- Loss function not interpretable

Maturity:

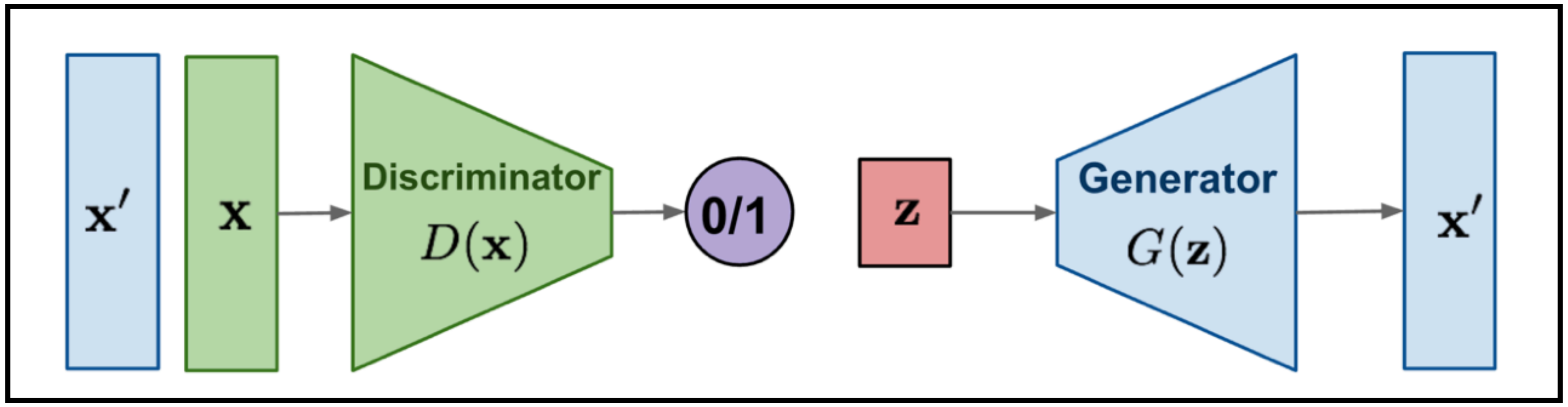
- Well established, many variants and extensions



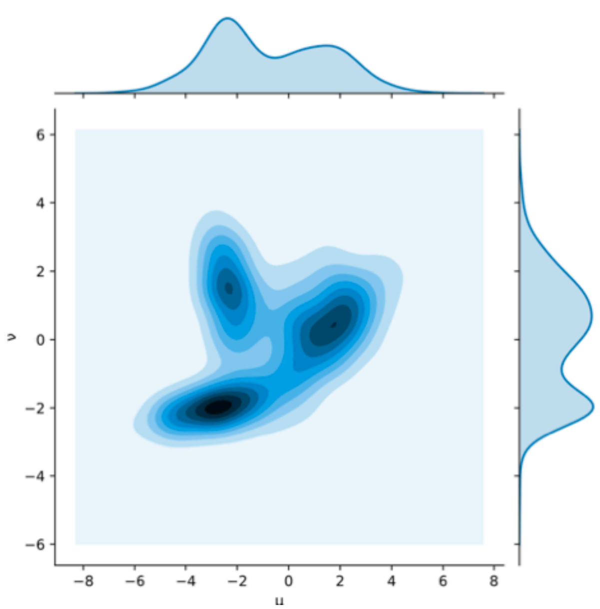
Mode collapse

1611.02163

Wasserstein GAN



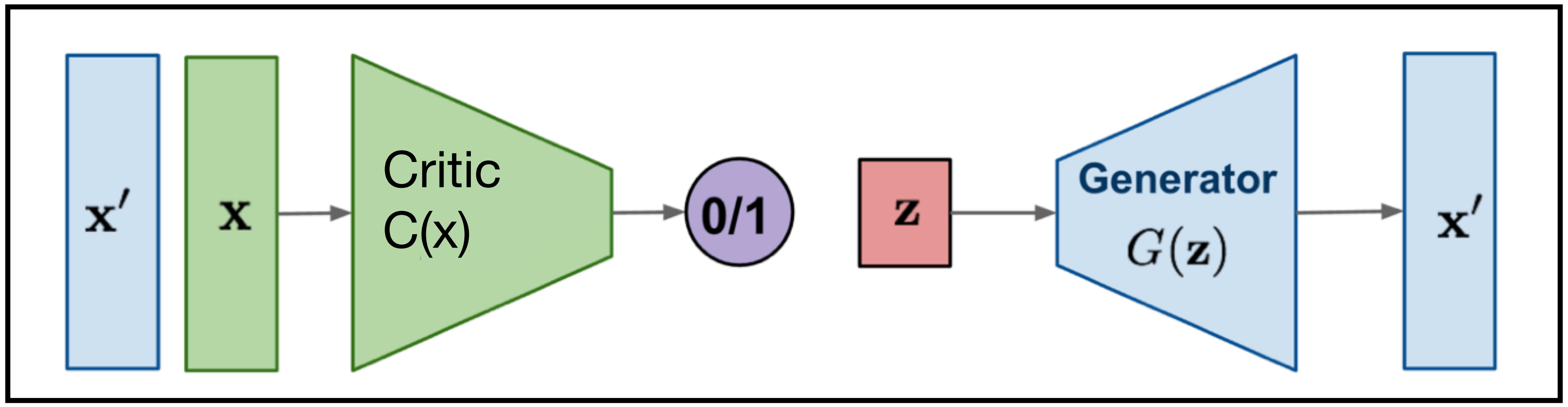
- Standard GANs minimise Jensen-Shannon divergence of generator output and true data
  - Not best measure, e.g. for non-overlapping distributions
- Replace with Wasserstein / Earth-Mover-Distance



$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \mathbf{E}_{(x,y)\sim\gamma} d(x,y)^p \right)^{1/p}$$

1704.00028

Wasserstein GAN



GAN loss:

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\boldsymbol{x}}))]$$

Wasserstein GAN loss*:

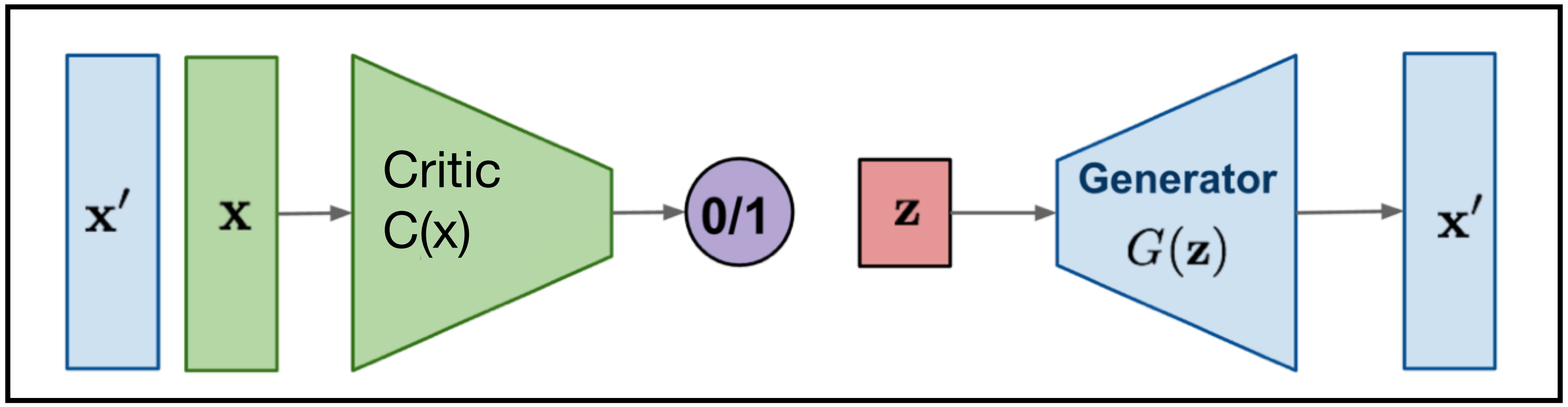
$$\min_G \max_{D\in\mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\big[D(\boldsymbol{x})\big] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}\big[D(\tilde{\boldsymbol{x}}))\big]$$

Requires bounded Lipschitz norm, e.g. via term in loss

* Some mathematics involved from earth mover distance to here

1704.00028

Wasserstein GAN



GAN loss:

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\boldsymbol{x}}))]$$

Wasserstein GAN loss:

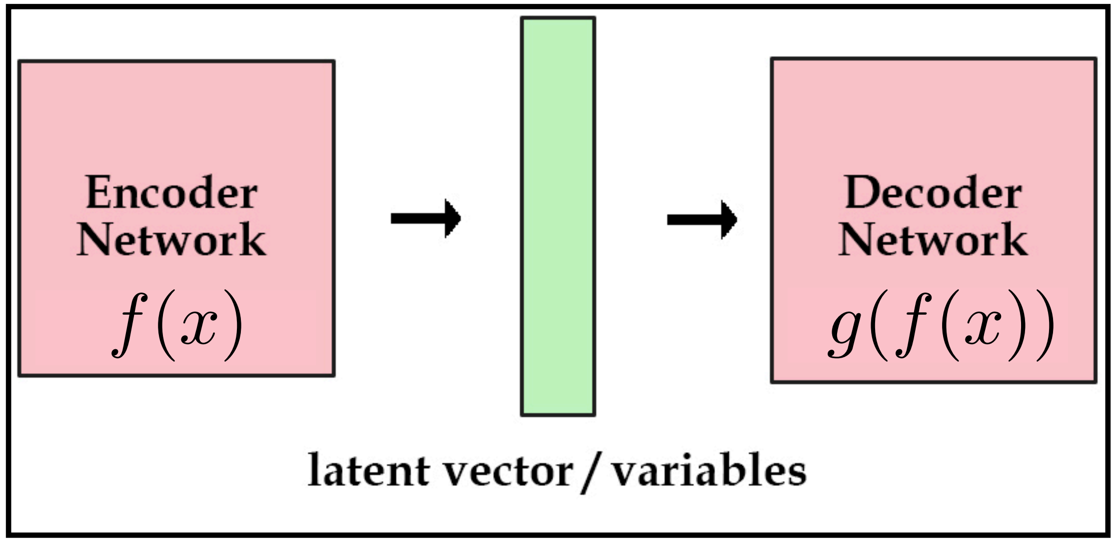
$$\min_G \max_{D\in\mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\big[D(\boldsymbol{x})\big] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}\big[D(\tilde{\boldsymbol{x}}))\big]$$

Improves training stability and sample quality (e.g. mode collapse)

1704.00028

Variational Autoencoders

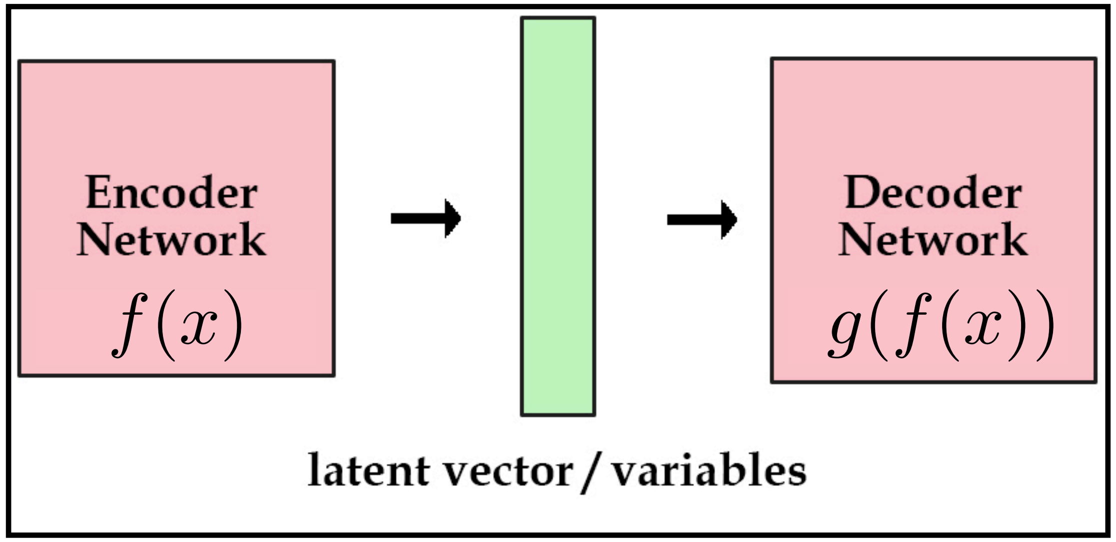
Autoencoder



Two networks
Encoder: data → latent space
Decoder: latent space → data

1312.6114
lilianweng.github.io

Autoencoder



Two networks
Encoder: data → latent space
Decoder: latent space → data

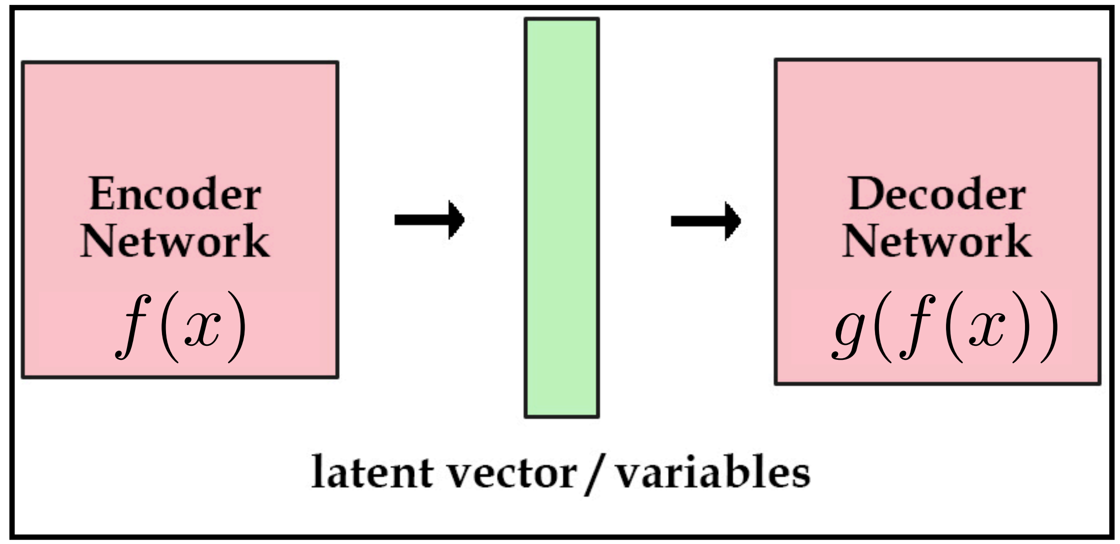
Decoder Encoder

$$L = (x - f(g(x)))^2$$

Training objective:
Minimise input/output difference

1312.6114

Autoencoder



Two networks
Encoder: data → latent space
Decoder: latent space → data

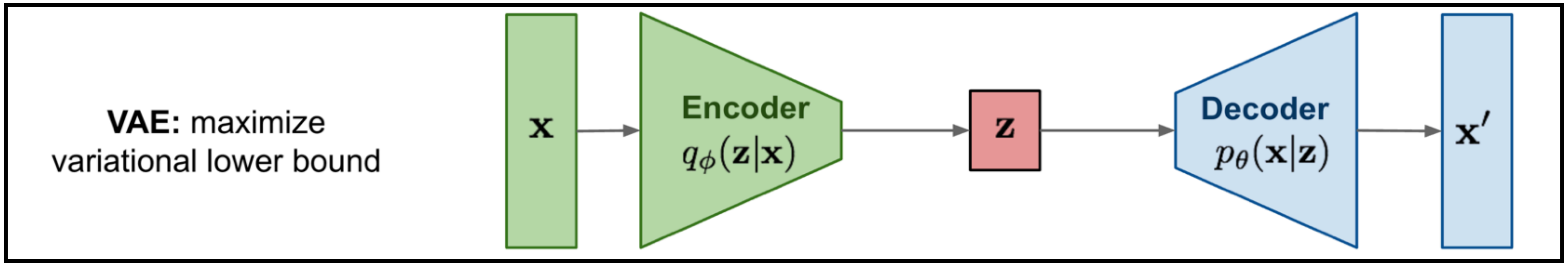
Training objective:
Minimise input/output difference

$$L = (x - \overset{\text{Decoder}}{f}(\overset{\text{Encoder}}{g}(x)))^2$$

Uses:
Dimension reduction
Denoising
Anomaly detection
Generation?

1312.6114

Variational Autoencoder

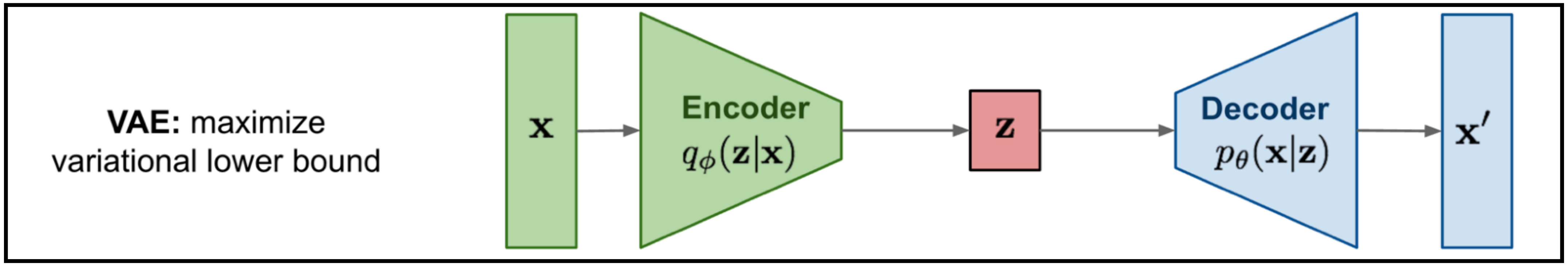


$$f(x) = (\mu, \sigma)$$

Variational Autoencoder (VAE):
Split latent space

1312.6114

Variational Autoencoder



Variational Autoencoder (VAE):
Split latent space
Sample before decoder

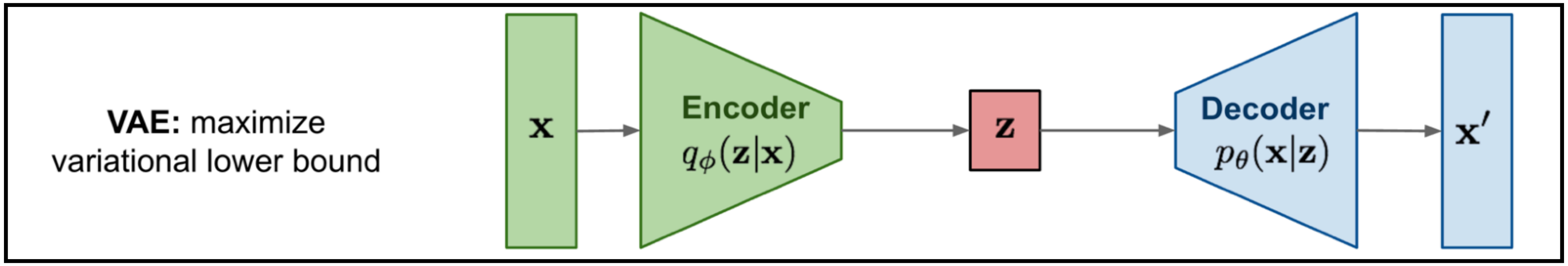
$$f(x) = (\mu, \sigma)$$

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

$$x' = g(z)$$

1312.6114

Variational Autoencoder



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

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

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

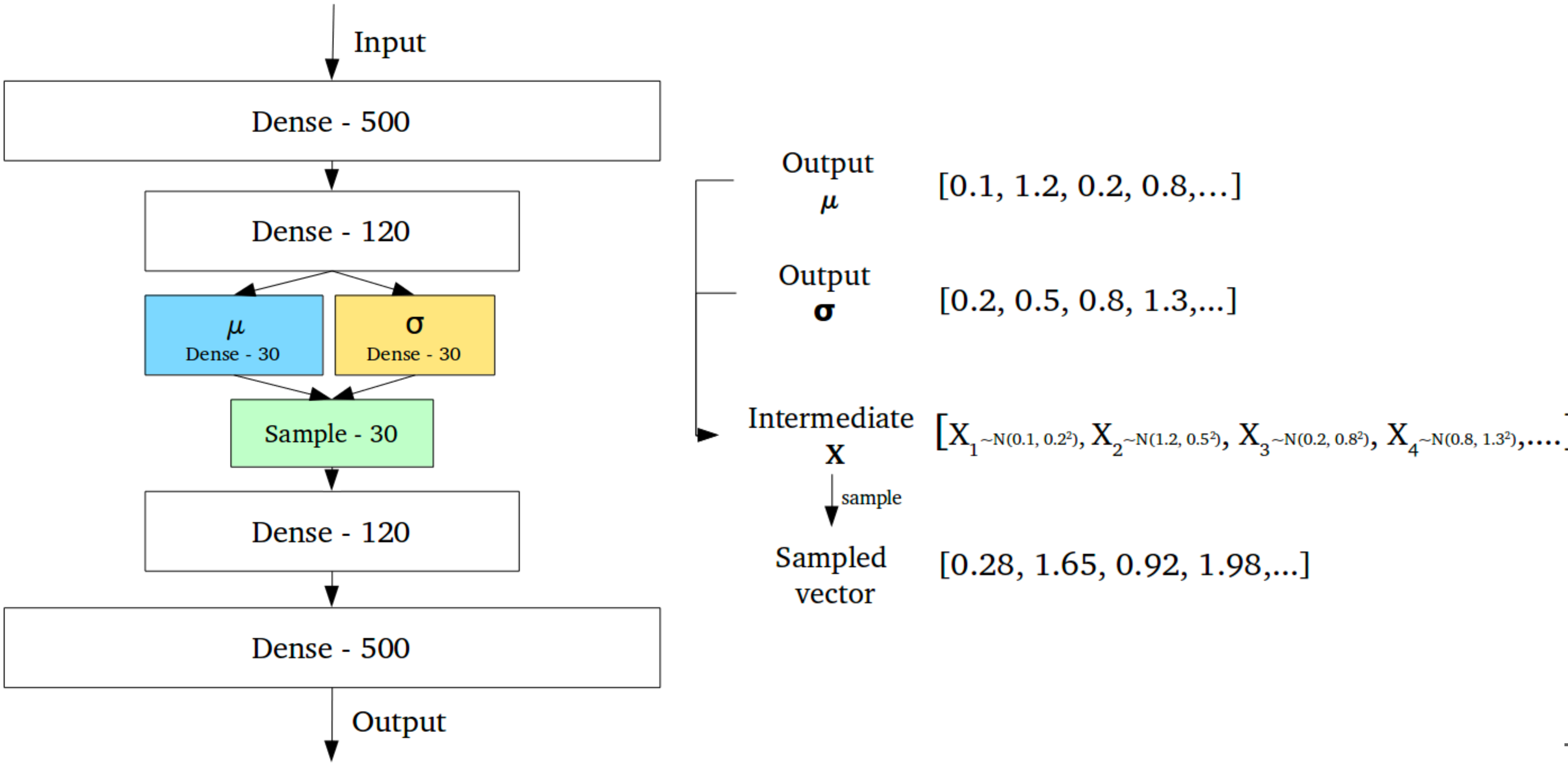
$$x' = g(z)$$

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

(Calculate KL-divergence between Gaussians)

1312.6114

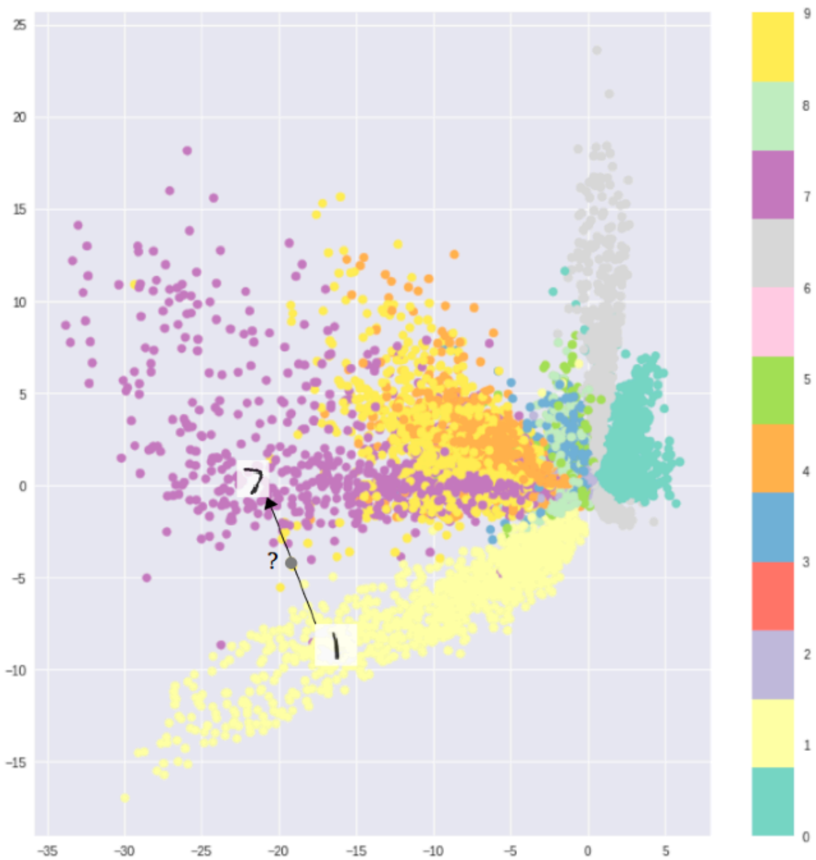
VAE Example



towardsdatascience.com

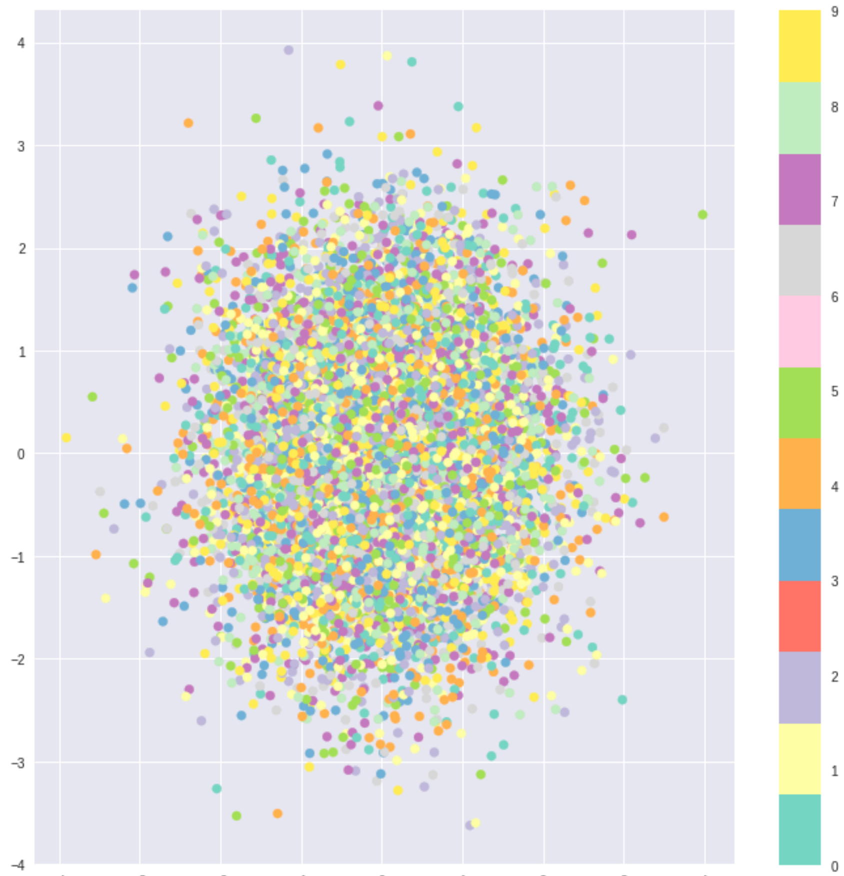
Loss terms

Latent space of MNIST VAE



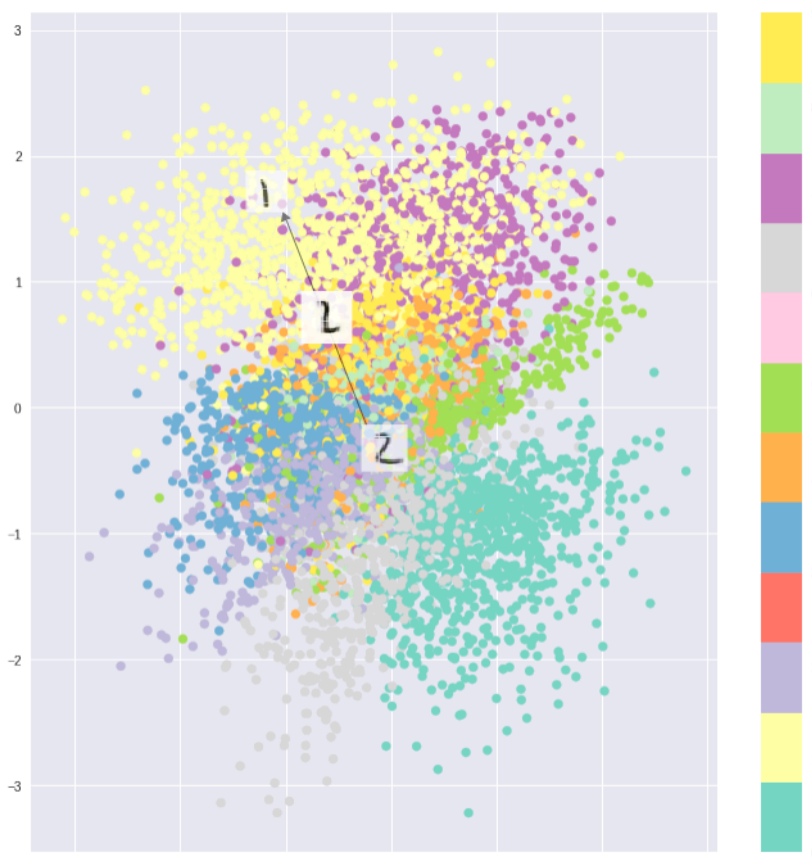
$$(x - g(z))^2$$

Reconstruction



$$\sigma^2 + \mu^2 - \log(\sigma) - 1$$

Regularisation



Both terms

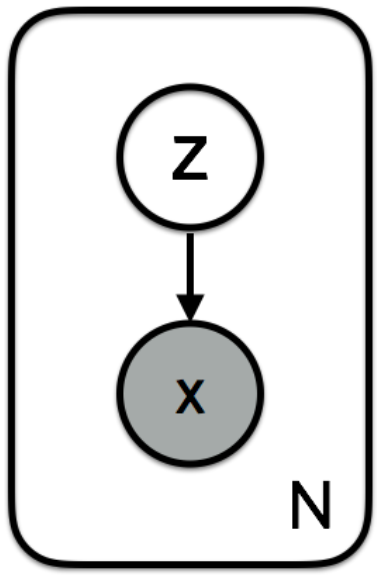
towardsdatascience.com

Loss terms

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

How did we get here?

Loss terms



Sample from latent variables z

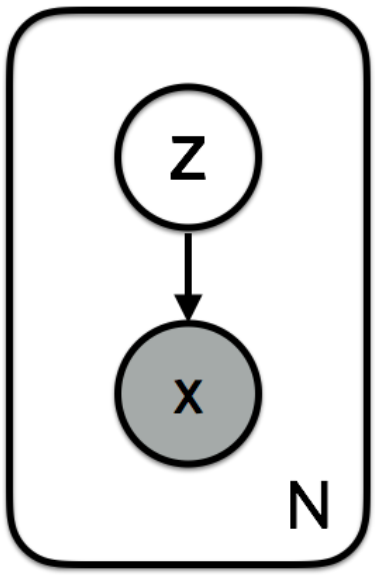
$$z_i \sim p(z)$$

Produce data points x

$$x_i \sim p(x \mid z)$$

https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Loss terms



Sample from latent variables z

$$z_i \sim p(z)$$

Produce data points x

$$x_i \sim p(x \mid z)$$

To choose correct latent distribution given data, could use Bayes theorem:

Conditional Prior

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

Difficult due to p(x)

Evidence

Loss terms

To choose correct latent distribution given data, could use Bayes theorem:

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

Instead, approximate with family of posterior distributions (variational inference):

$$\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

And find optimal approximation:

$$q_\lambda^*(z \mid x) = \arg\min_\lambda \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

Still difficult due to (hidden) p(x) term!

Loss terms

$$\mathbb{KL}(q(x)\,||\,p(x)) = -\int \mathrm{d}x\, q(x) \log \frac{p(x)}{q(x)}$$

Kullback-Leibler definition

To choose correct latent distribution given data, could use Bayes theorem:

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

Instead, approximate with family of posterior distributions (variational inference):

$$p(x, z) = p(z|x)p(x)$$

Reminder

$$\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

And find optimal approximation:

$$q^*_\lambda(z \mid x) = \arg\min_\lambda \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

Still difficult due to p(x) term!

Loss terms

$$\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

Introduce

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$

Rewrite

$$\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

As KL is >=0, ELBO is a lower limit for p(X)
ELBO: Evidence Lower Bound

Loss terms

Maximise

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x,z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$

Rewrite for samples, using neural networks:

$$ELBO_i(\theta,\phi) = \mathbb{E}q_\theta(z \mid x_i)[\log p_\phi(x_i \mid z)] - \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

Reconstruction term

Regularisation term

Assume normal distribution

Difference between normal and standard normal

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

Loss terms

Maximise

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$

Rewrite for samples, using neural networks:

$$ELBO_i(\theta, \phi) = \mathbb{E}q_\theta(z \mid x_i)[\log p_\phi(x_i \mid z)] - \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

Reconstruction term

Regularisation term

Assume normal distribution

Difference between normal and standard normal

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

Comments on VAEs

Architecture:

- Low complexity, fast and adaptable
- Target: Maximise lower bound on likelihood

Learning:

- Stable training
- Average prediction → blurrier output
- Interpretable latent space

Maturity:

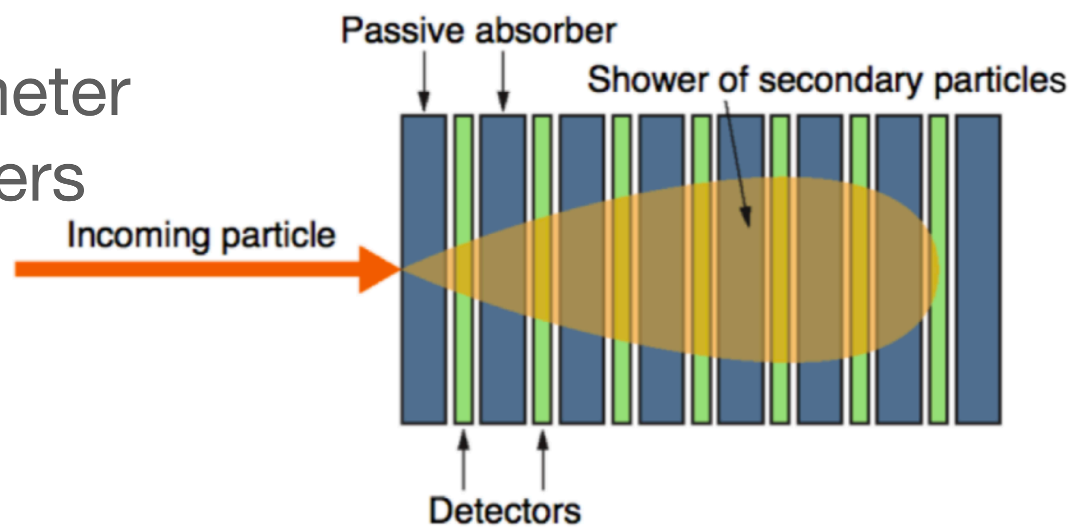
- Well established,
  many variants and extensions

1607.07539

Applications I

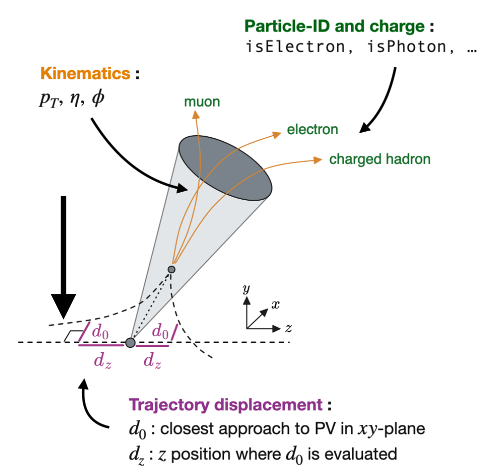
(Some) Simulation targets

Calorimeter Showers



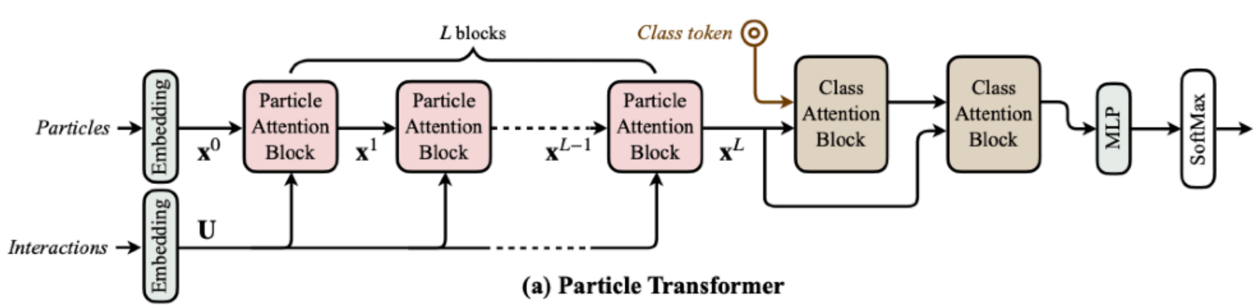
Reduce computational bottleneck

Jet Constituents



Predict background from data

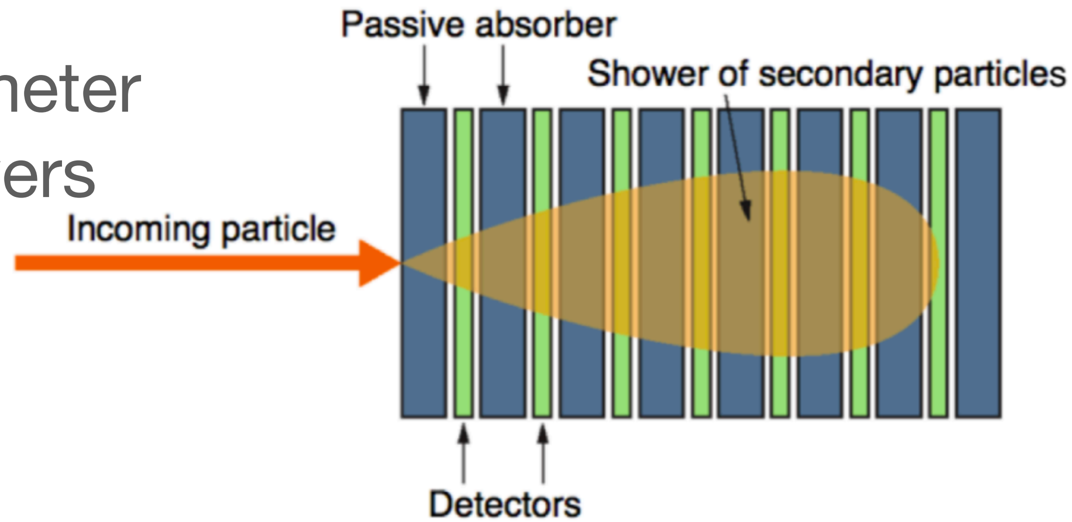
Classification and Reconstruction tasks



Act as surrogate models

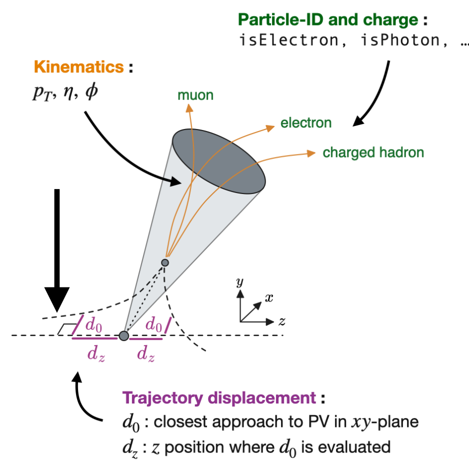
(Some) Simulation targets

Calorimeter Showers



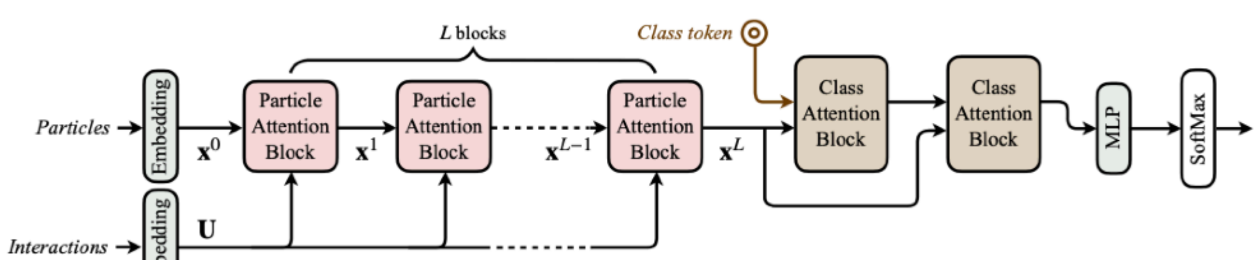
Reduce computational bottleneck

Jet Constituents



Predict background from data

Classification and Reconstruction tasks

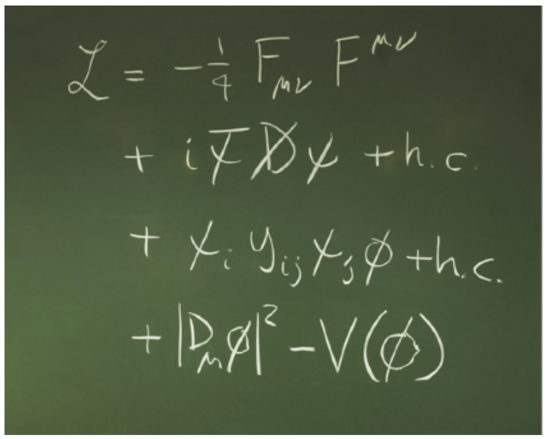


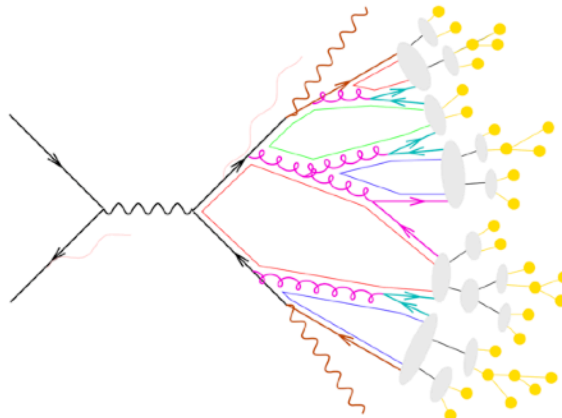
(a) Particle Transformer

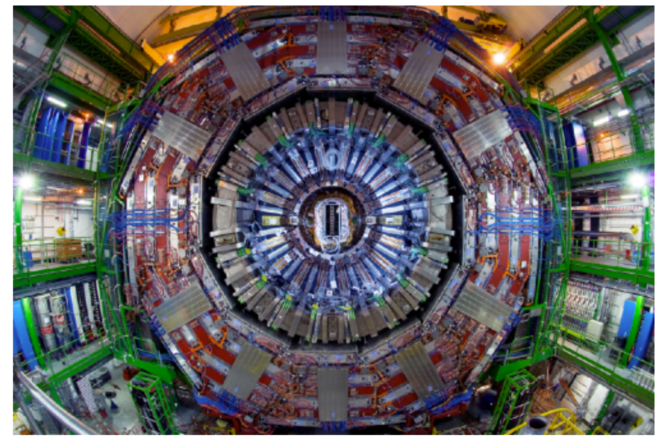
Act as surrogate models

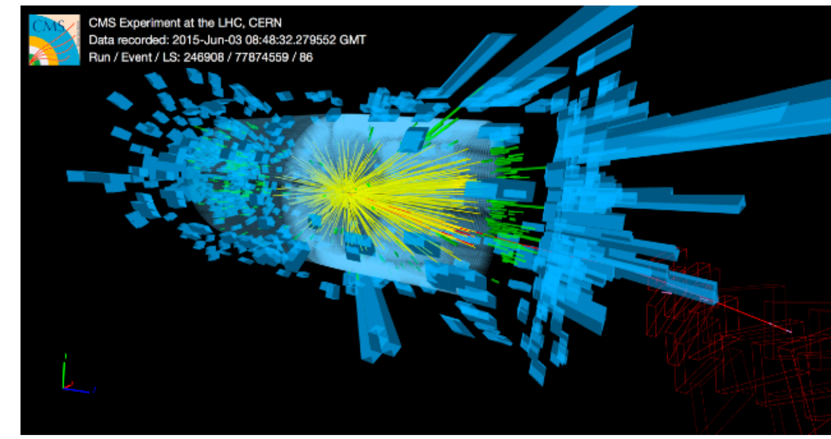
Generative Models

This happens in the experiment







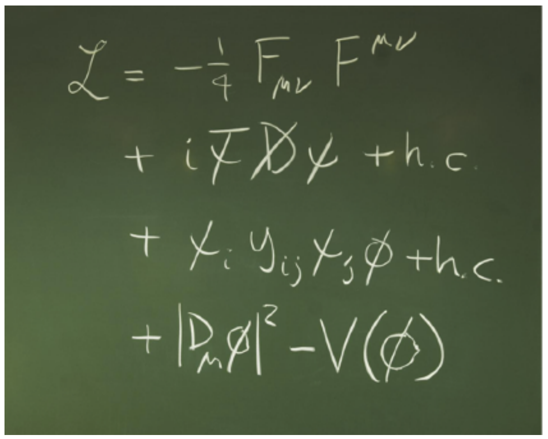


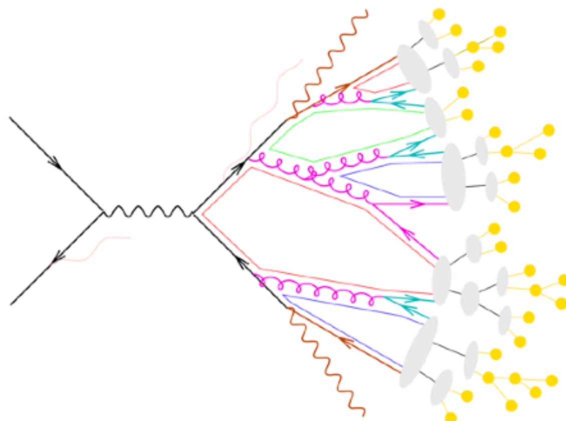
This is what we want to know

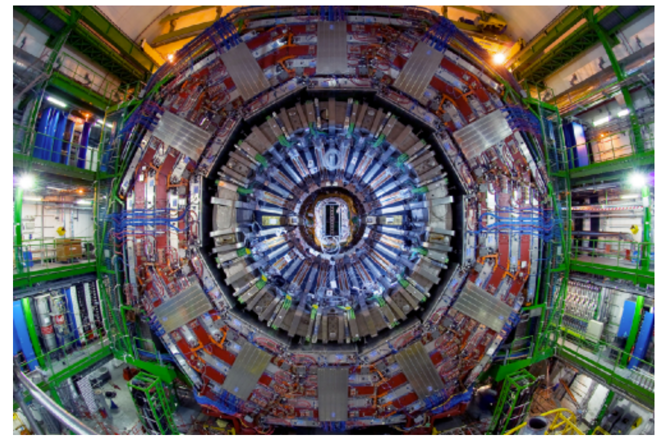
Simulation is crucial to connect experimental data with theory predictions

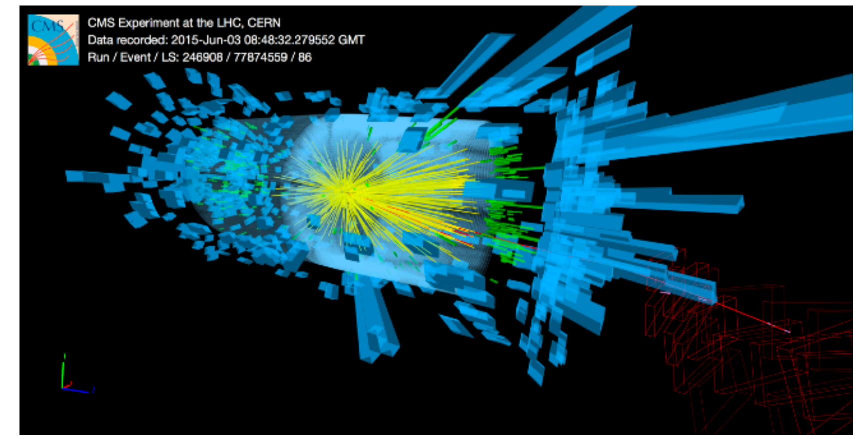
Generative Models

This happens in the experiment



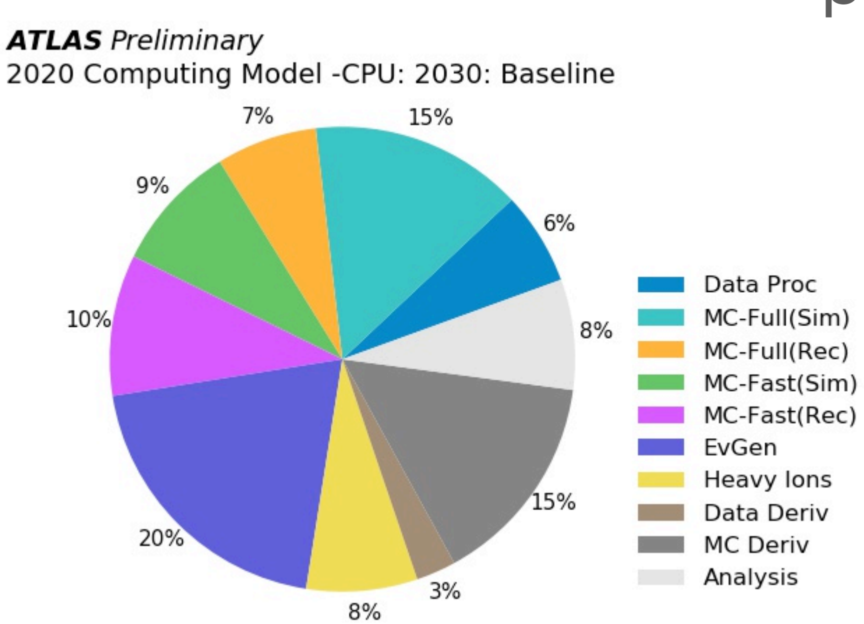


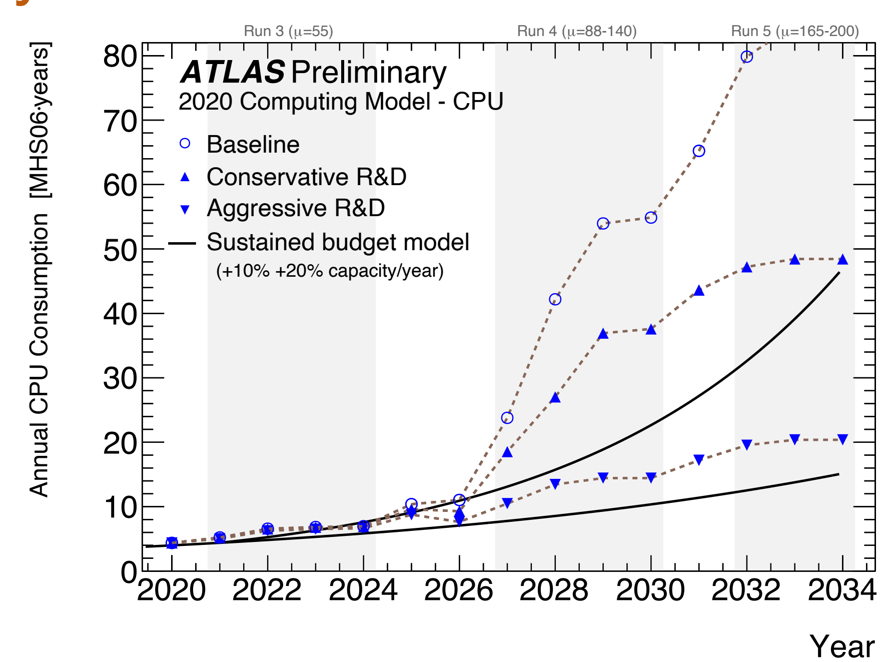




This is what we want to know

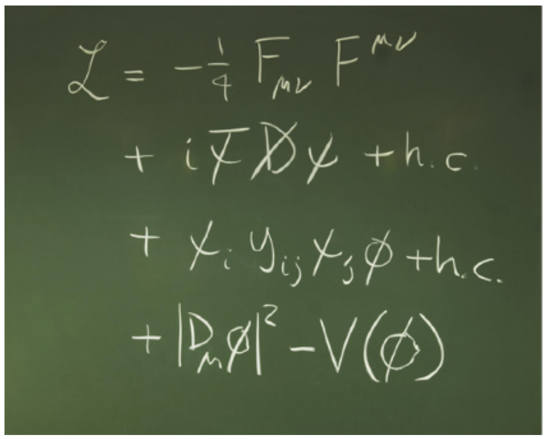
Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

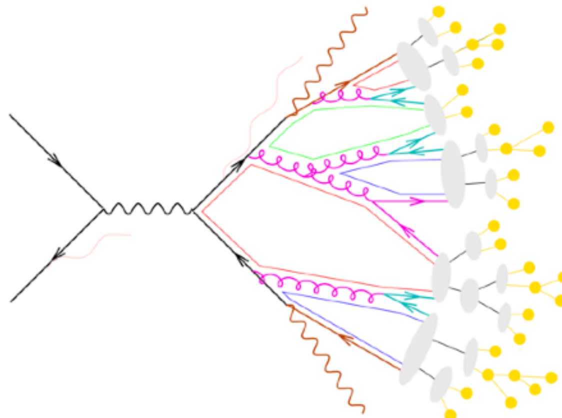


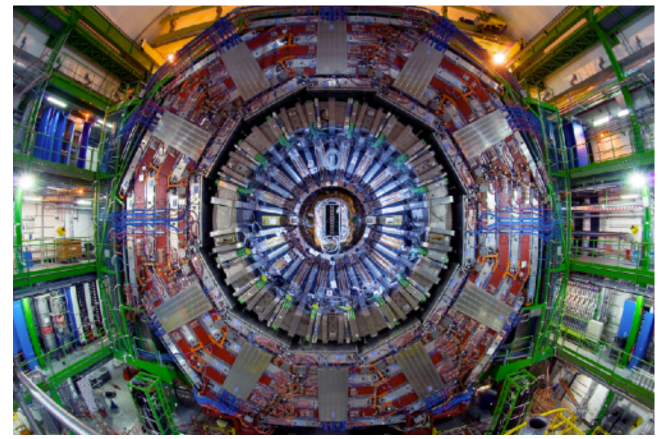


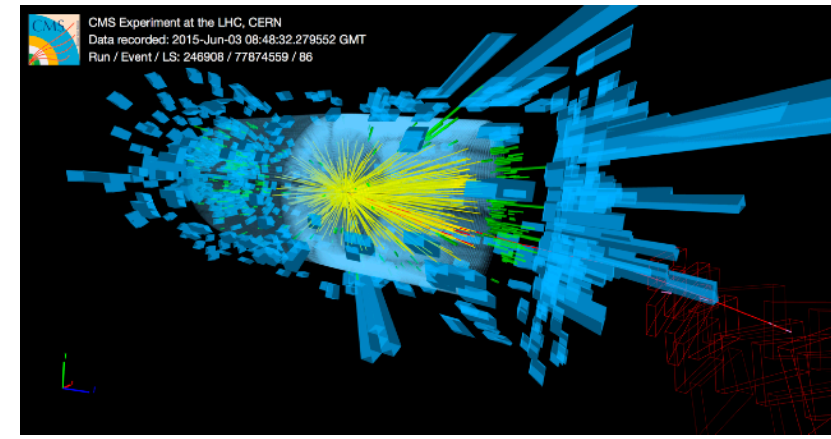
Generative Models

This happens in the experiment





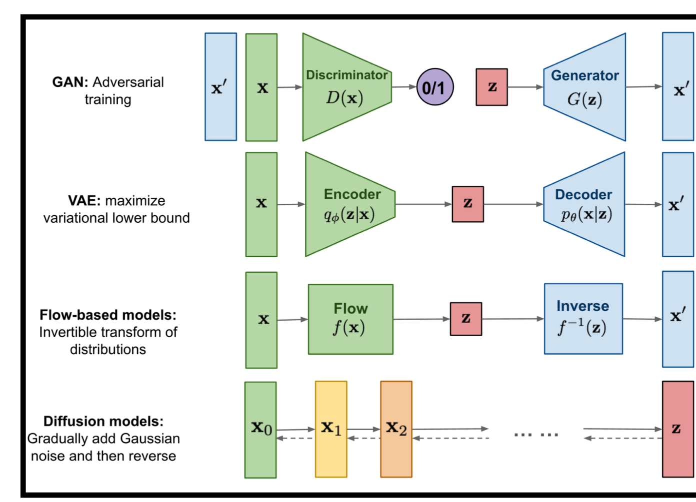




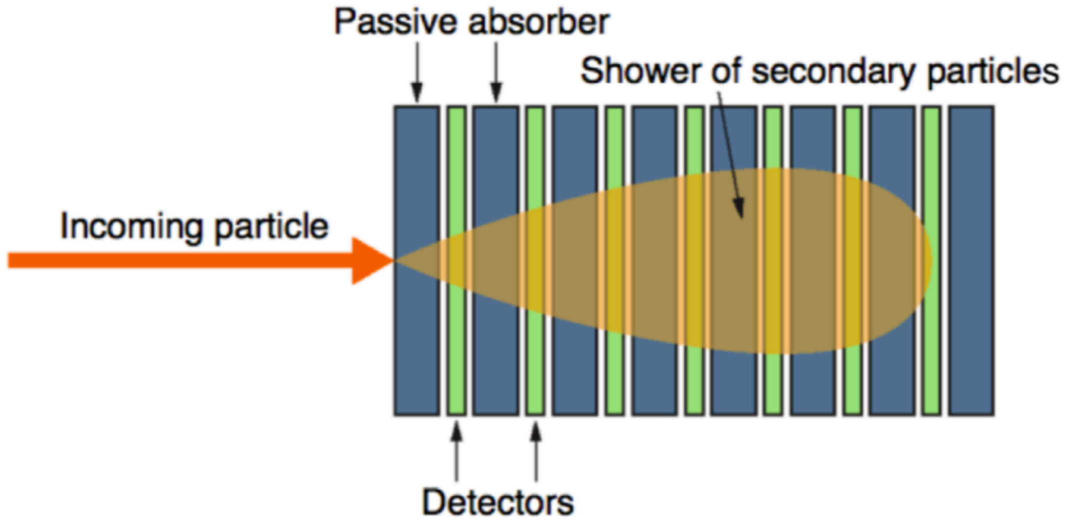
This is what we want to know

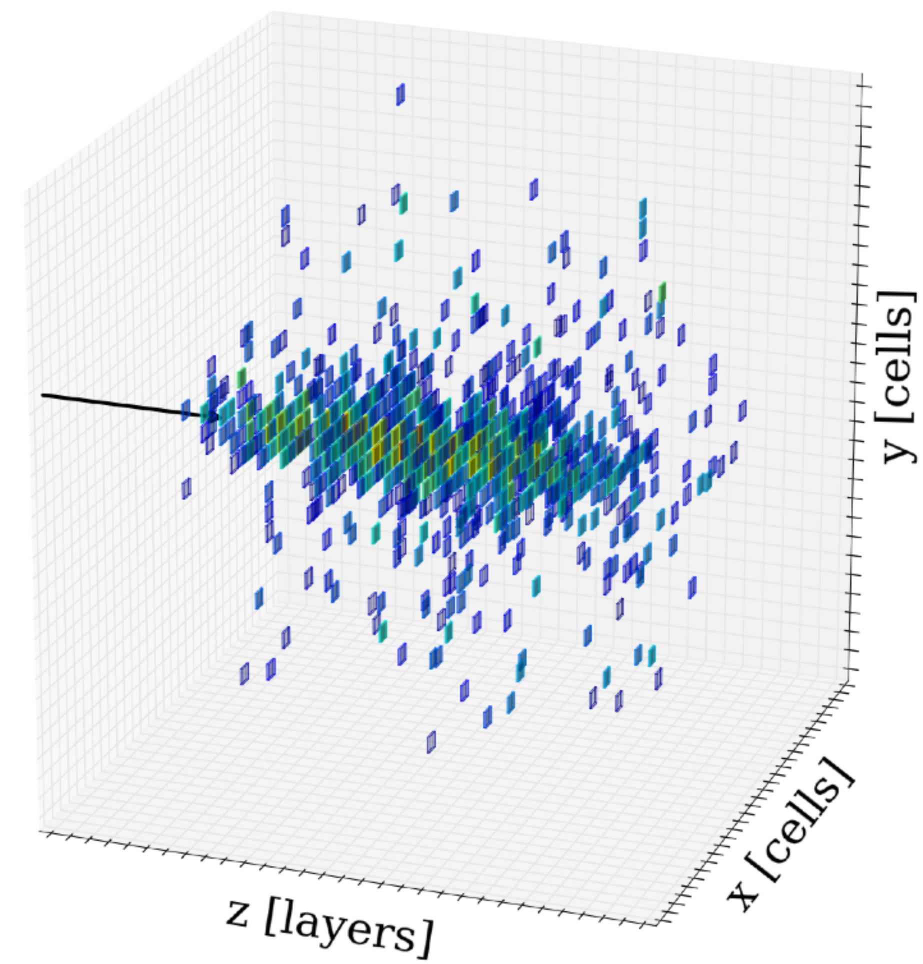
Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

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



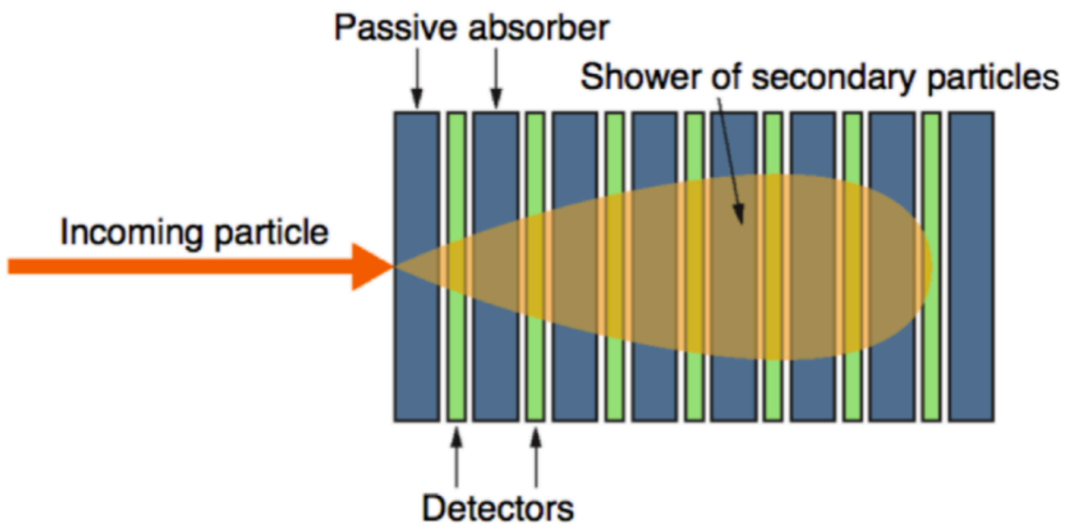
Simulation targets

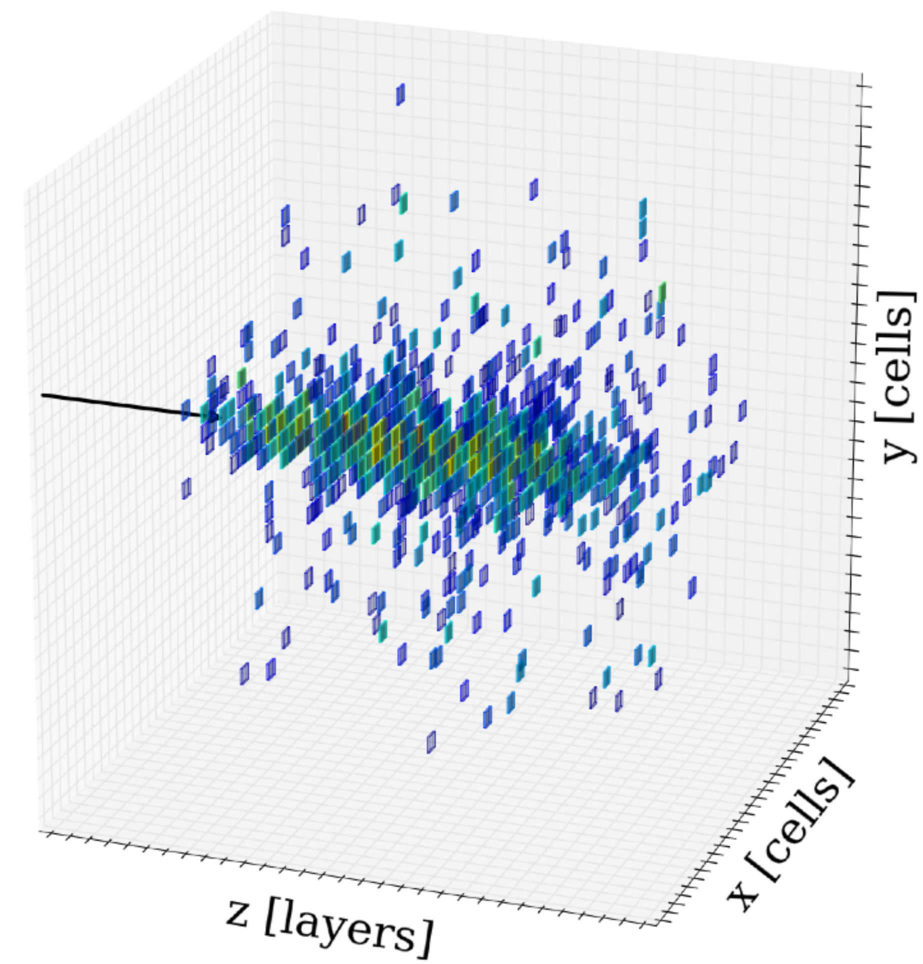




How to represent?

Simulation targets

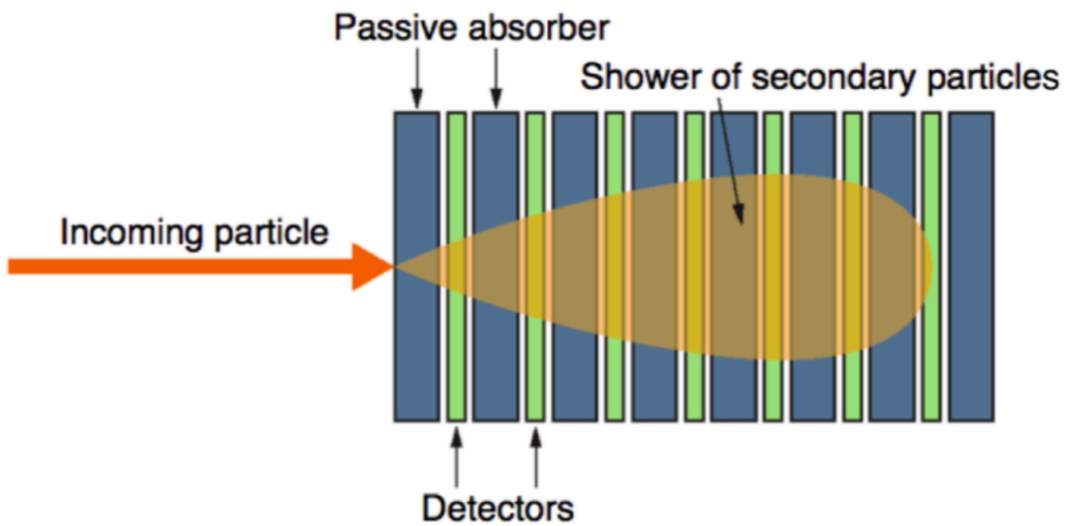


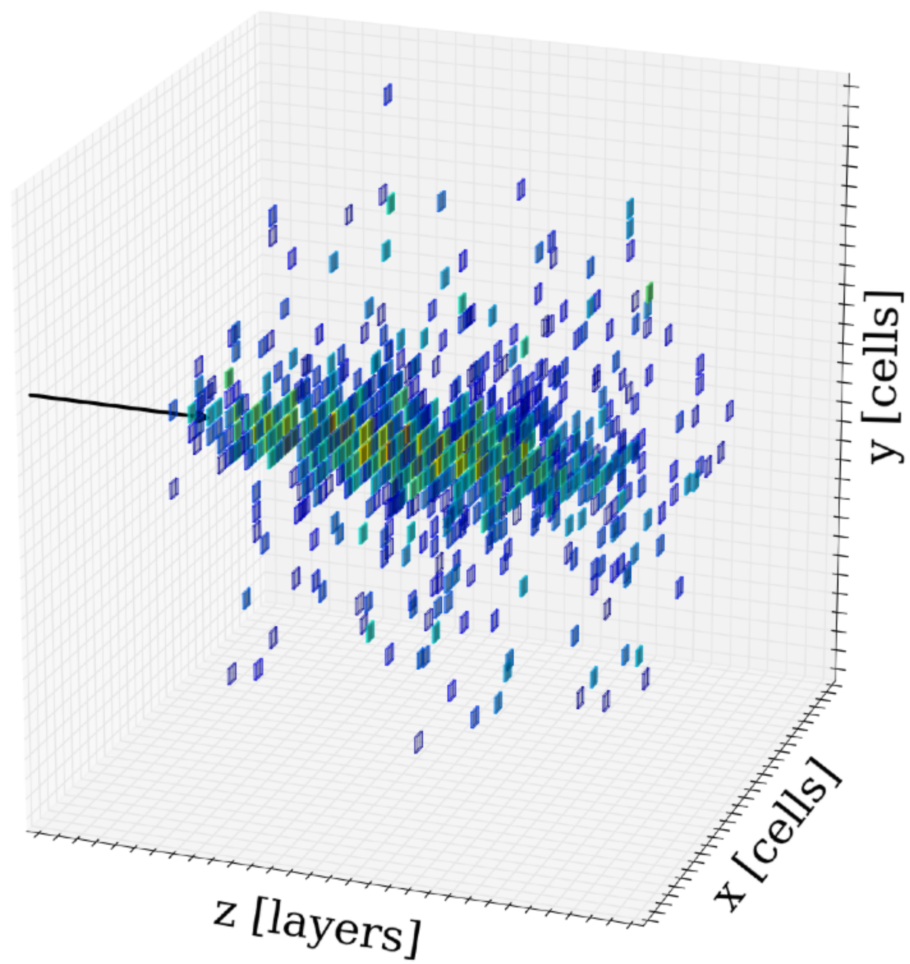


How to represent?

Tabular data:
Easy, insufficient for high-dimensions

Simulation targets



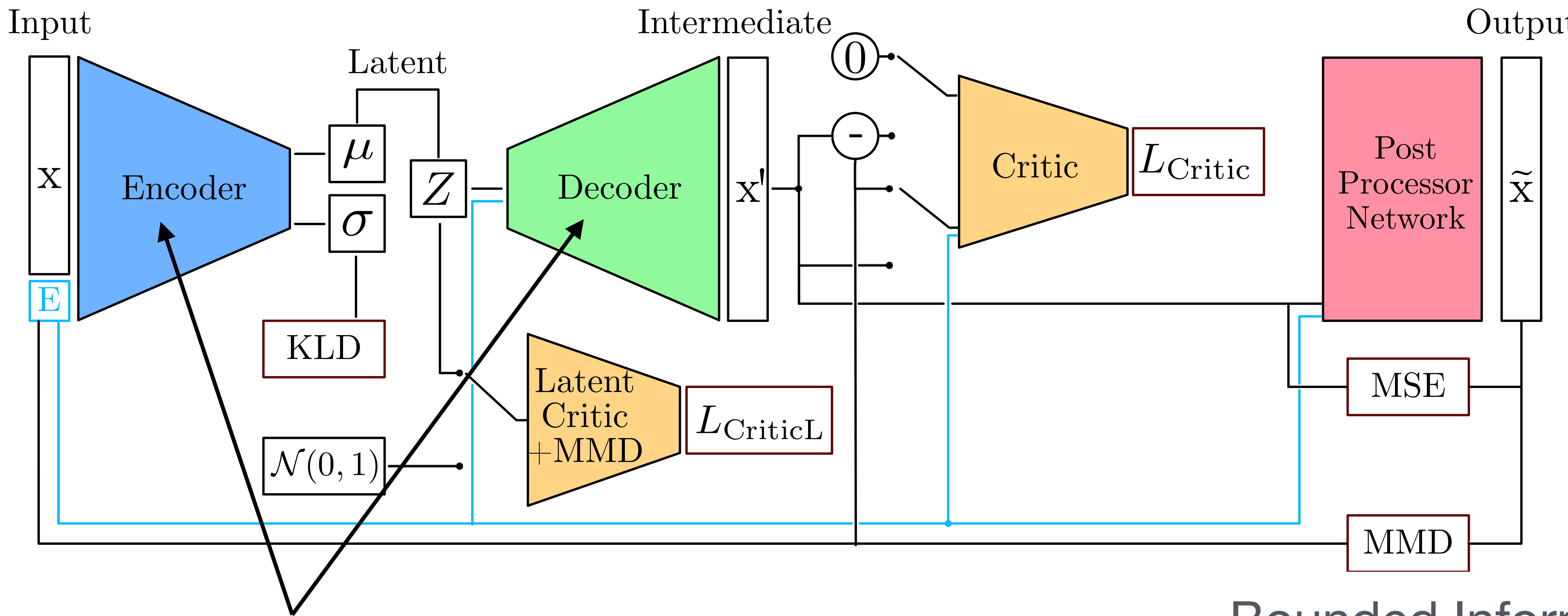


How to represent?

Tabular data

Fixed grid (voxels)

Generative results



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

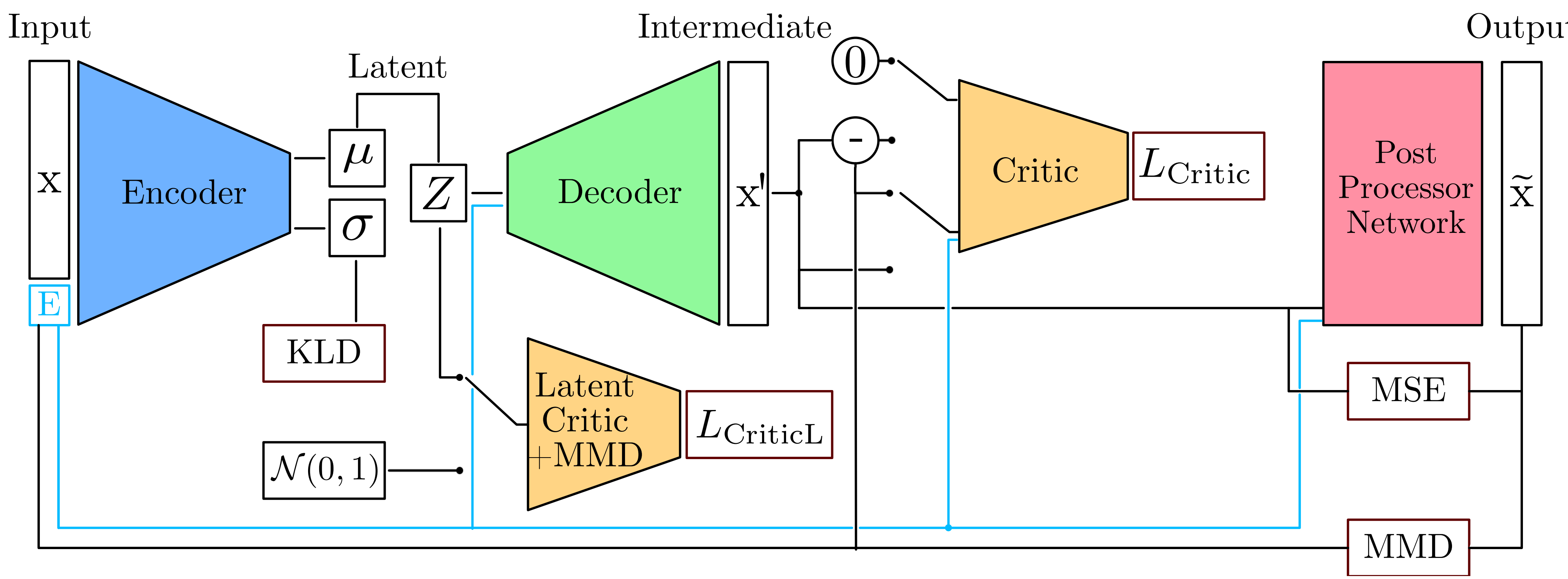
Critic

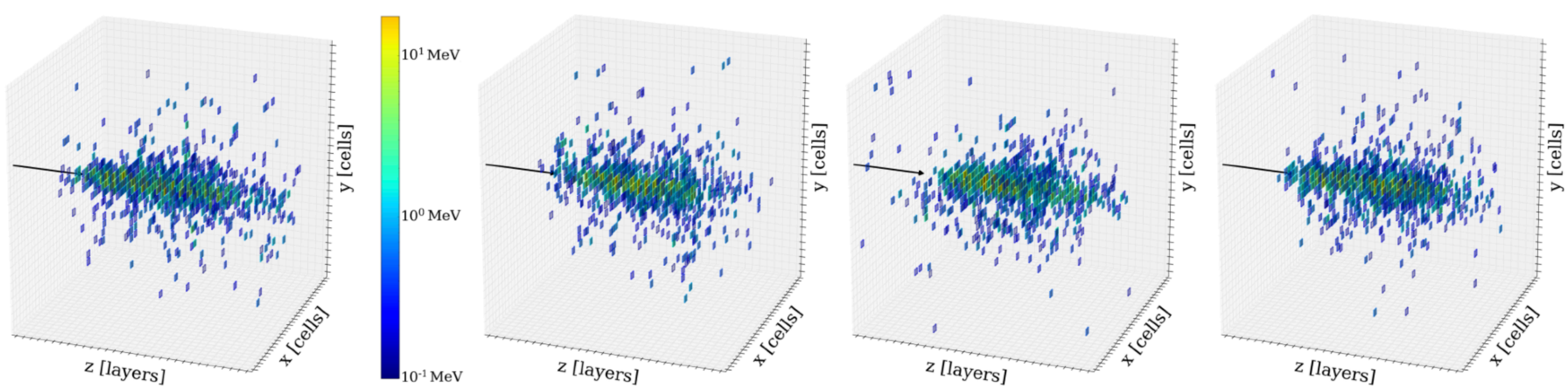
Difference Critic

Latent Regularisation

1912.00830, 2005.05334

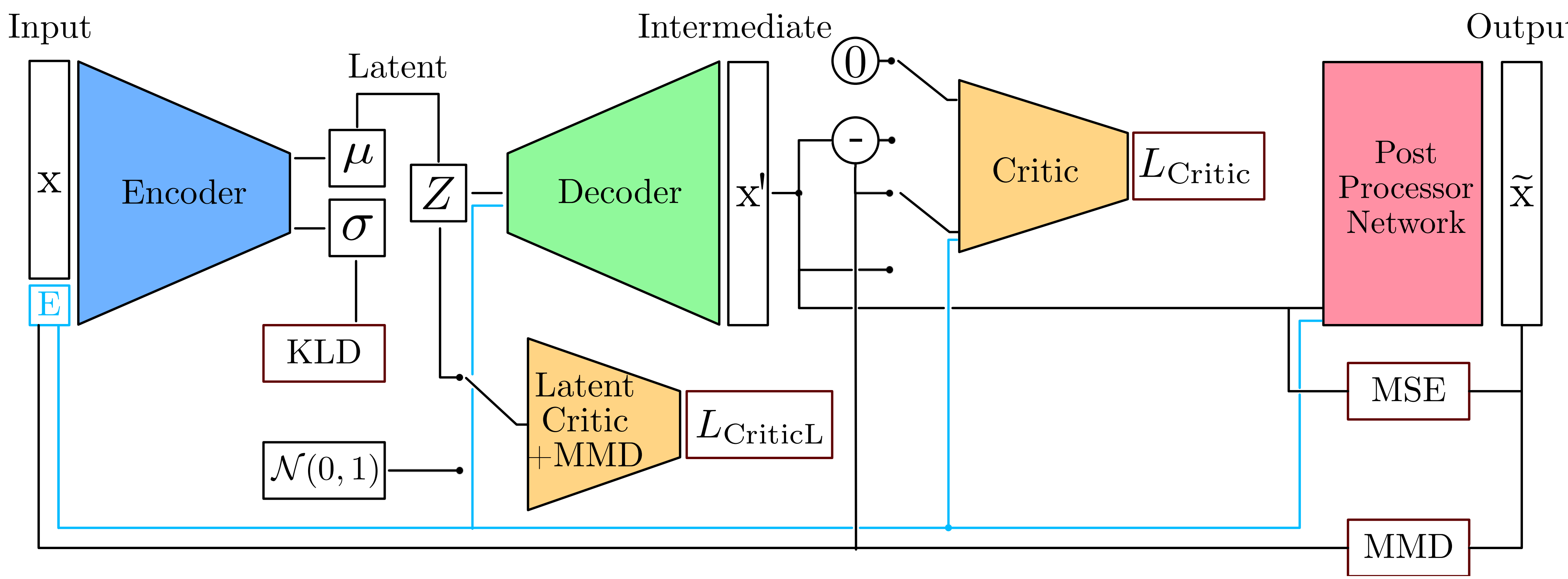
Generative results

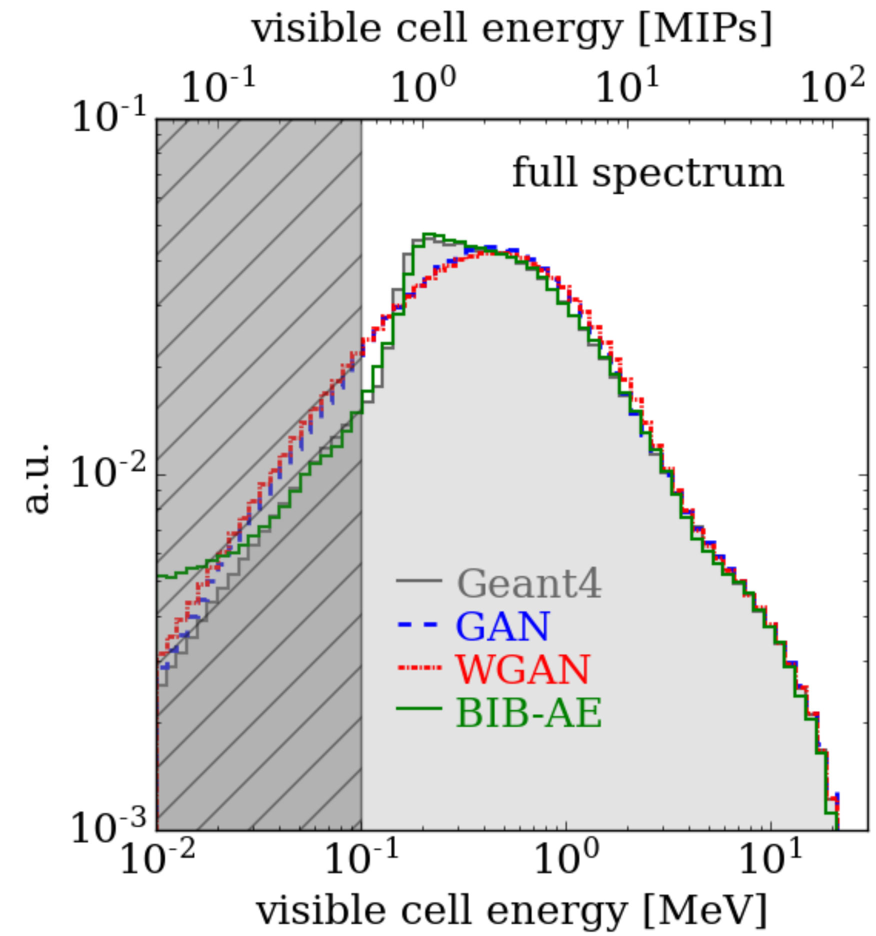




2005.05334

Generative results



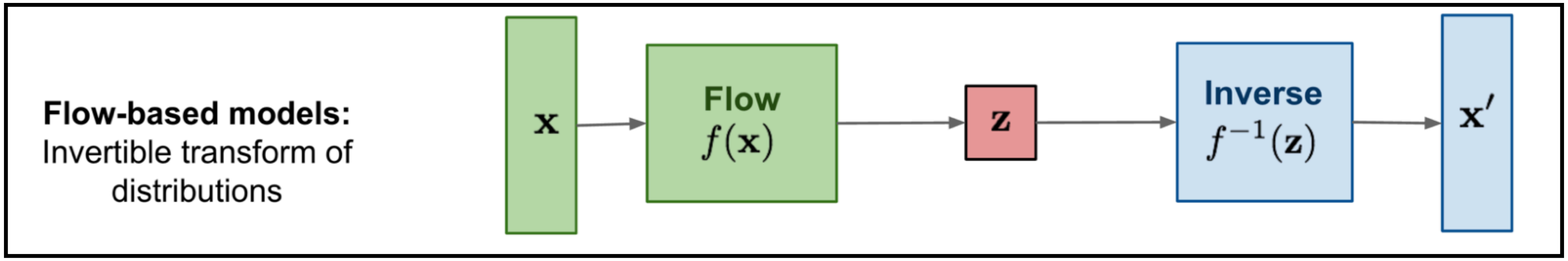


2005.05334

Go with the…

(Normalising) Flows

Generative models

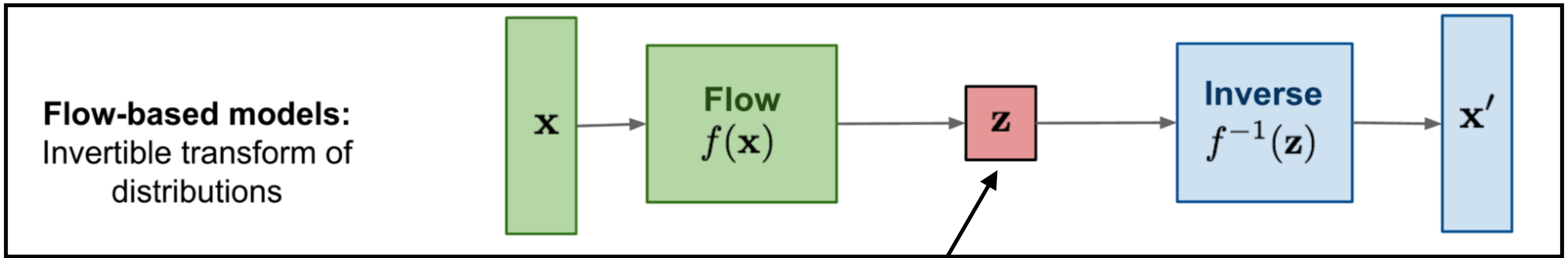


In auto-encoders, the decoder learns to 'undo' the encoder

Can we make this exact?

lilianweng.github.io

Generative models

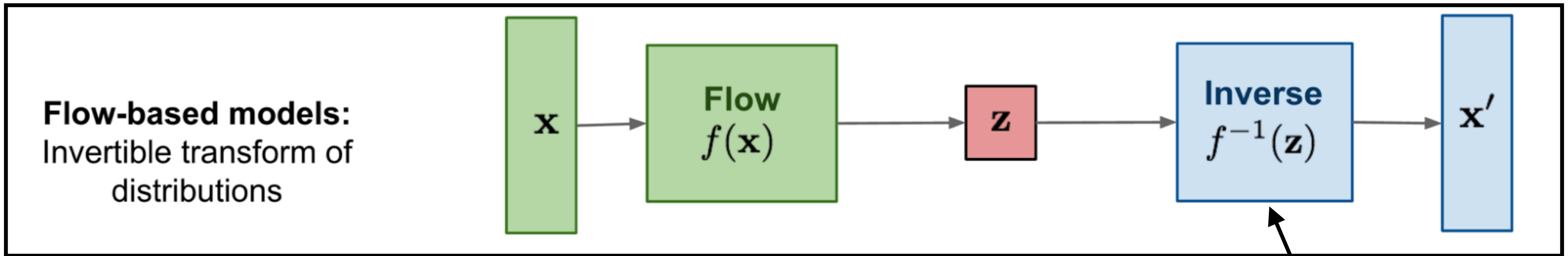


Choose latent space, e.g. standard normal distribution (normalising flow!)
Same dimension as data!

Learn a diffeomorphism between data and latent-space

1505.05770, 1908.09257

Generative models

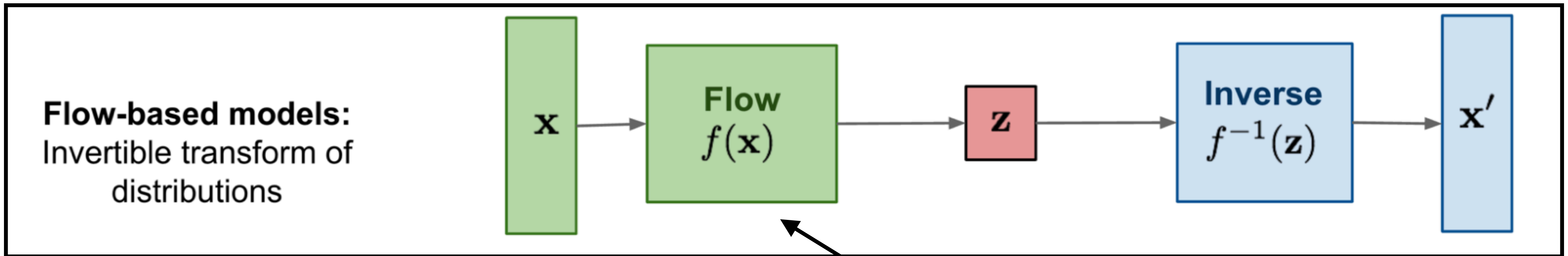


f^{-1} is not a learned inversion, but exact inverse by construction

Learn a diffeomorphism between data and latent-space

Bijective, invertable

Generative models



Take into account Jacobian determinant to evaluate probability density

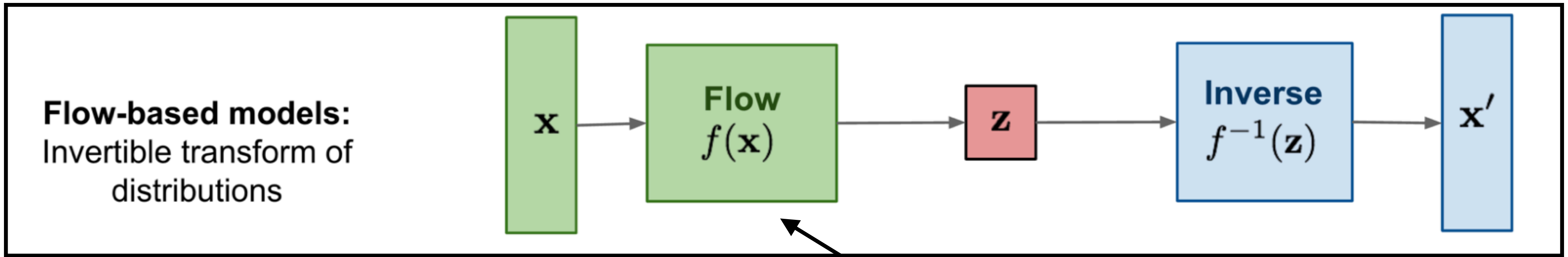
Learn a diffeomorphism between data and latent-space

Bijective, invertable

Learn likelihood of data

1505.05770, 1908.09257

Generative models

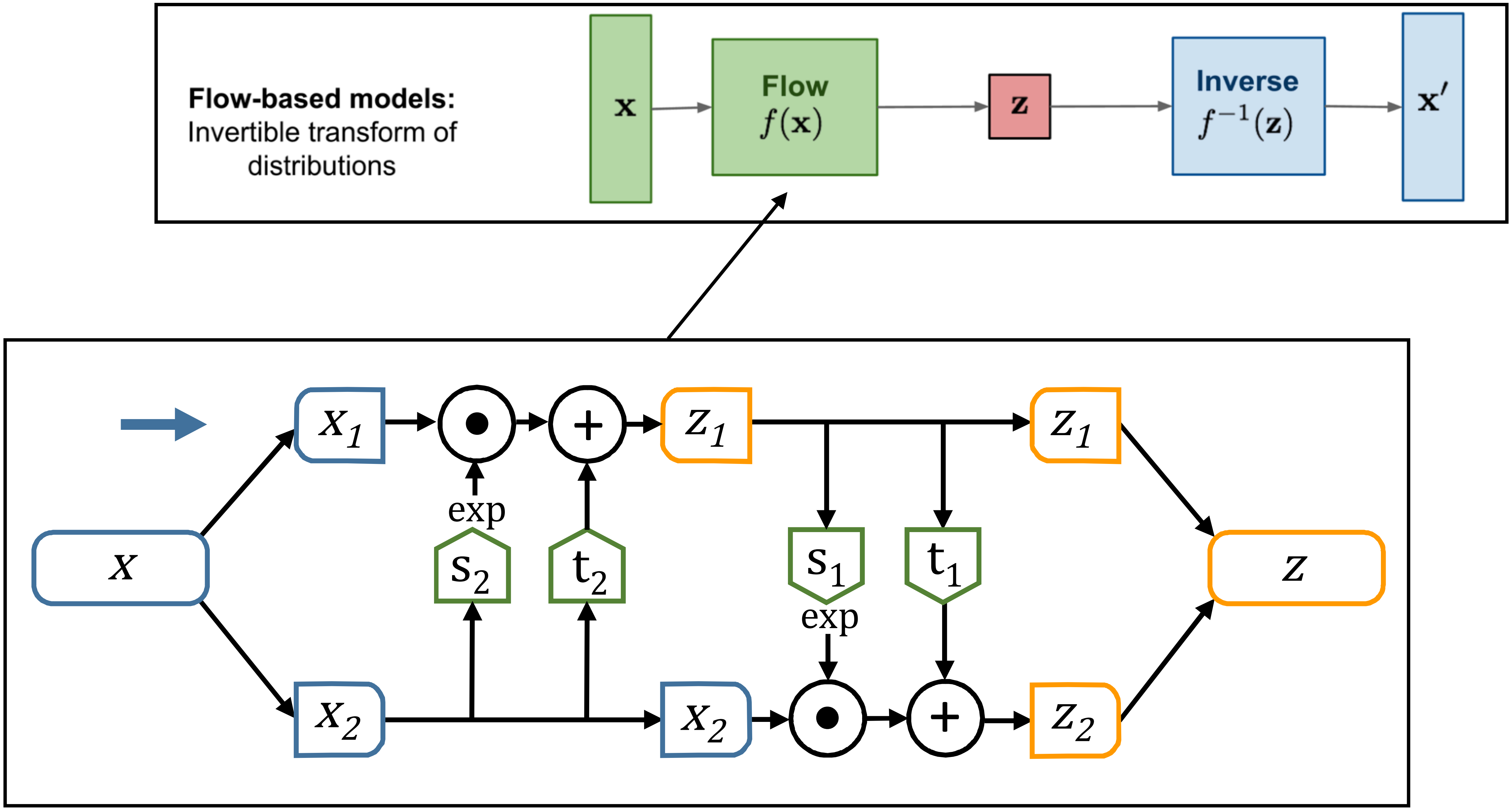


Take into account Jacobian determinant to evaluate probability density

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

1505.05770, 1908.09257

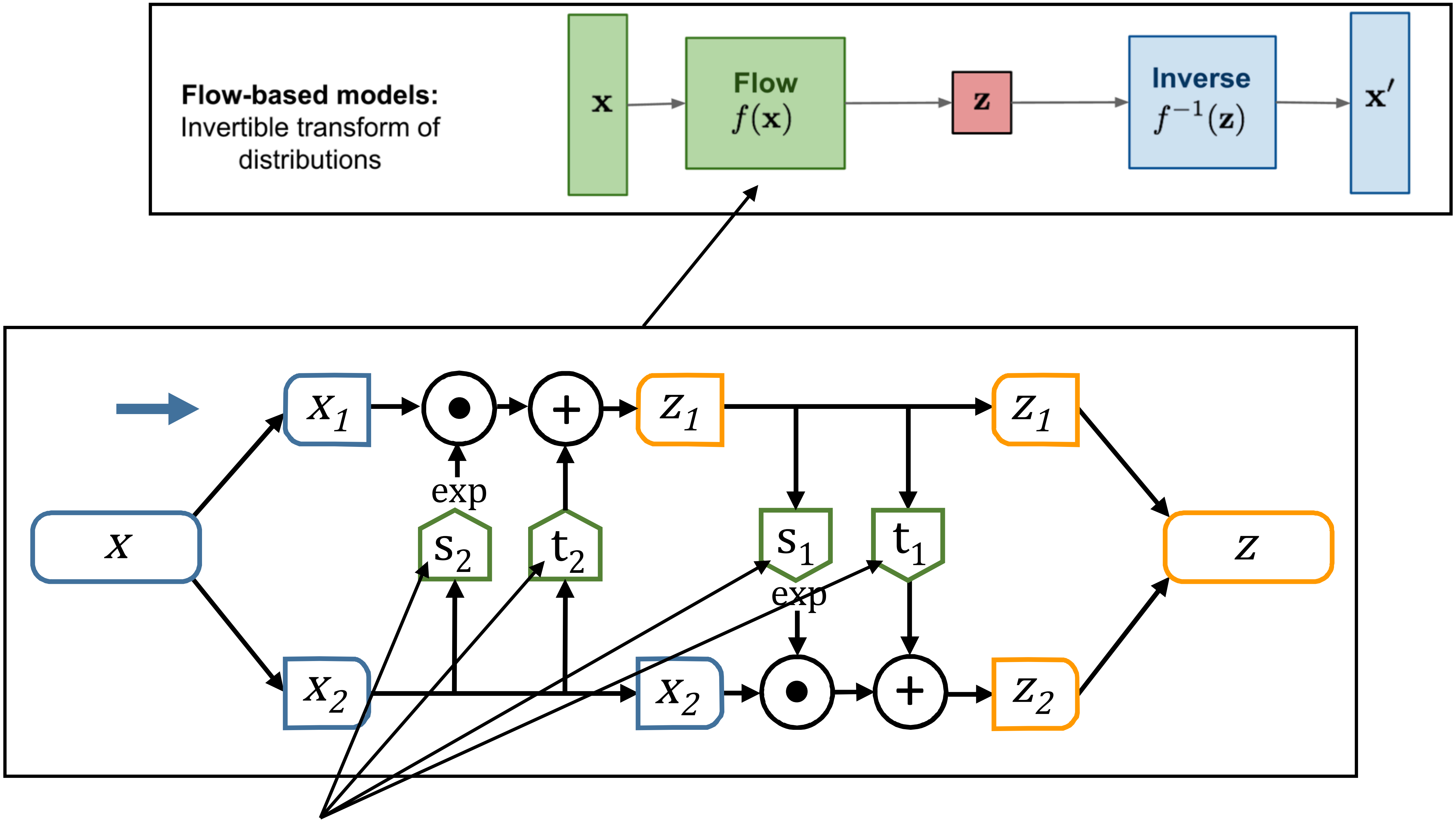
Coupling flows



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

1505.05770, 1908.09257

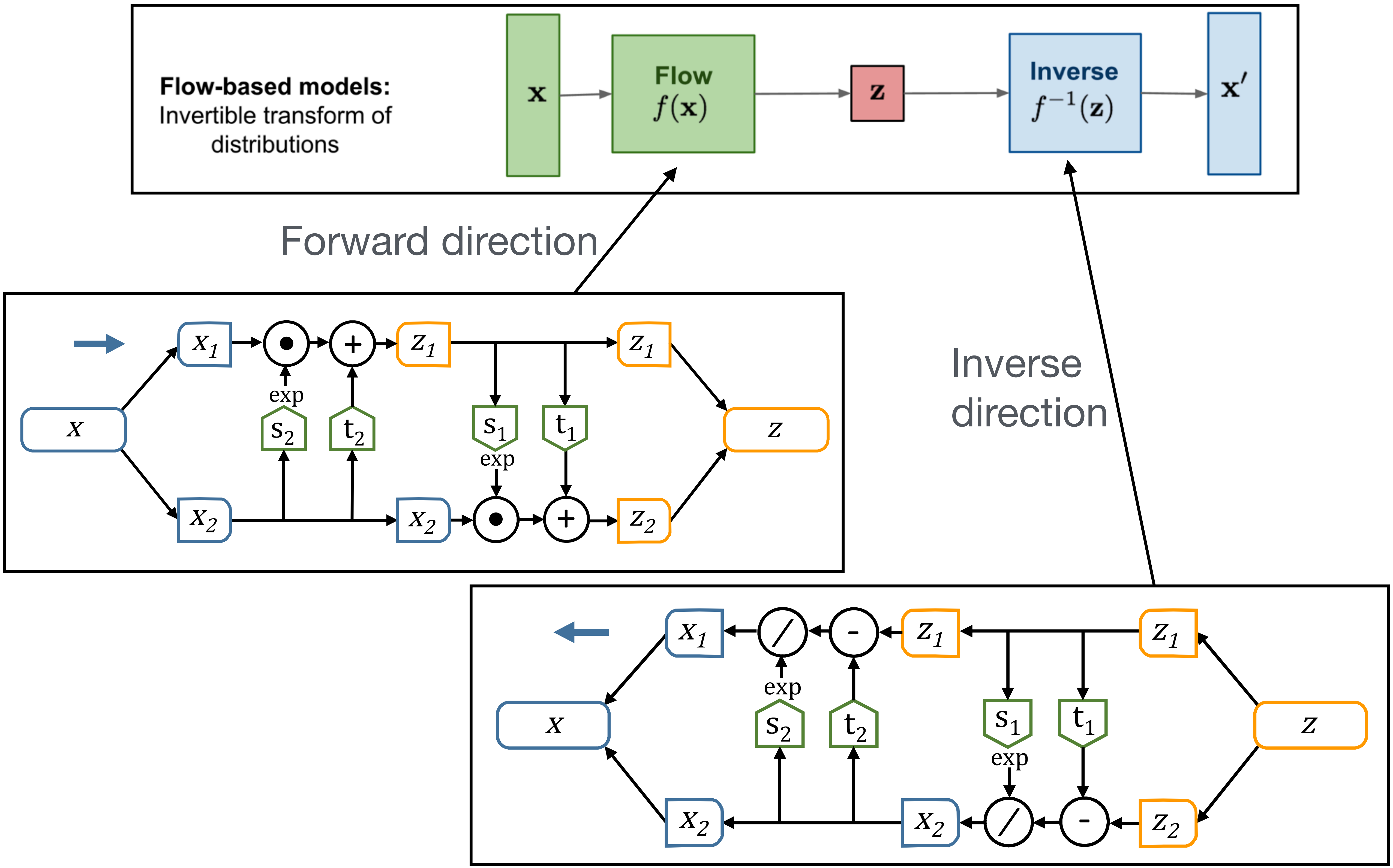
Coupling flows



Simple (e.g. dense) neural networks

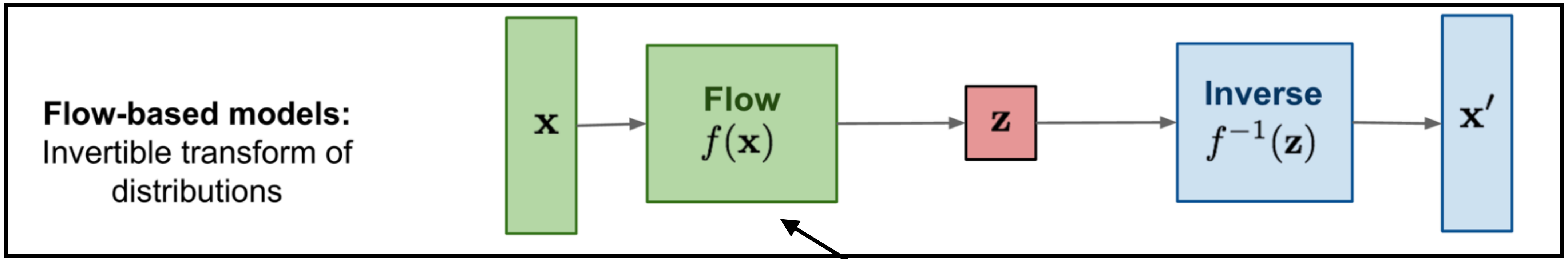
1505.05770, 1908.09257

Coupling flows



1505.05770, 1908.09257

Generative models

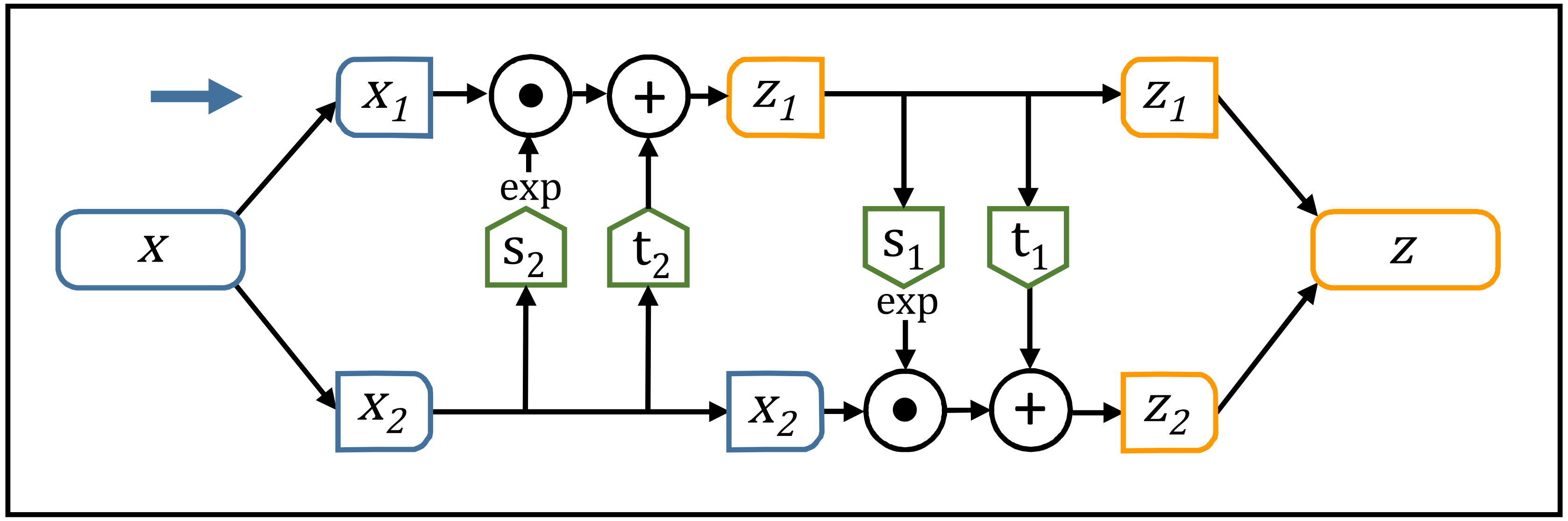


Take into account Jacobian determinant to evaluate probability density

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

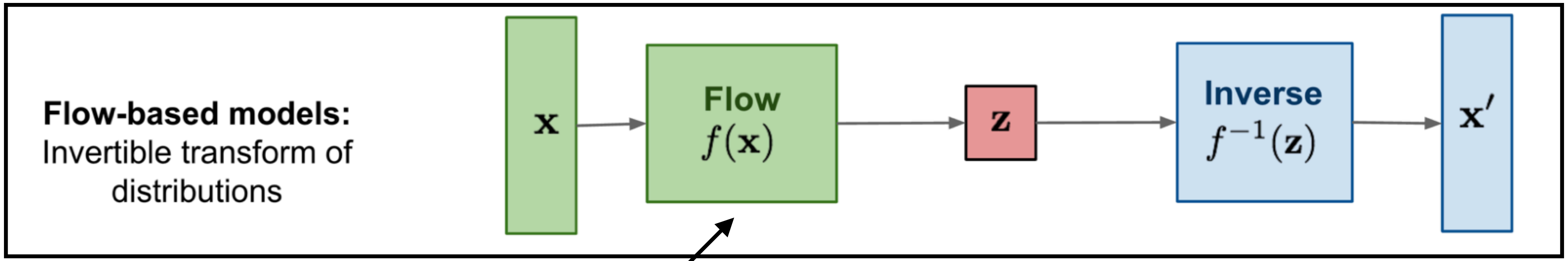
$$\mathbf{J_1} = \begin{pmatrix} \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{z}_1}{\partial \mathbf{x}_2} \\ \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_2} \end{pmatrix} = \begin{pmatrix} \mathrm{diag}(\exp(s_2(\mathbf{x}_2)))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

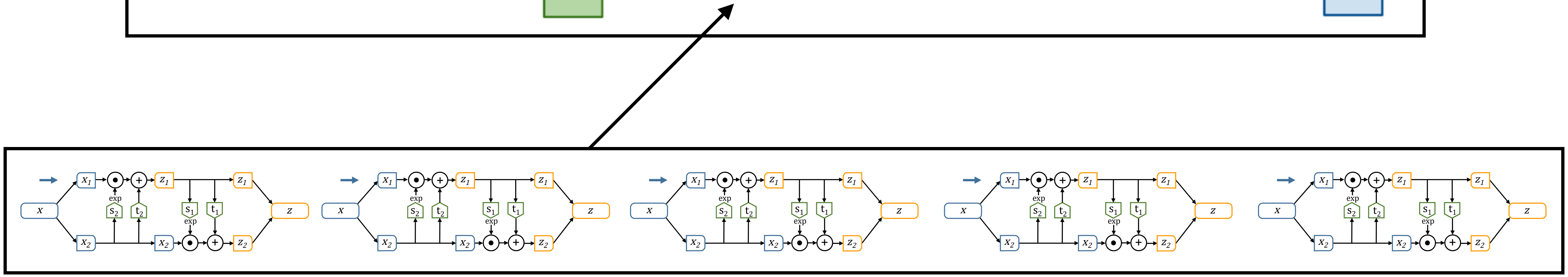
Triangular by construction

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

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

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

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

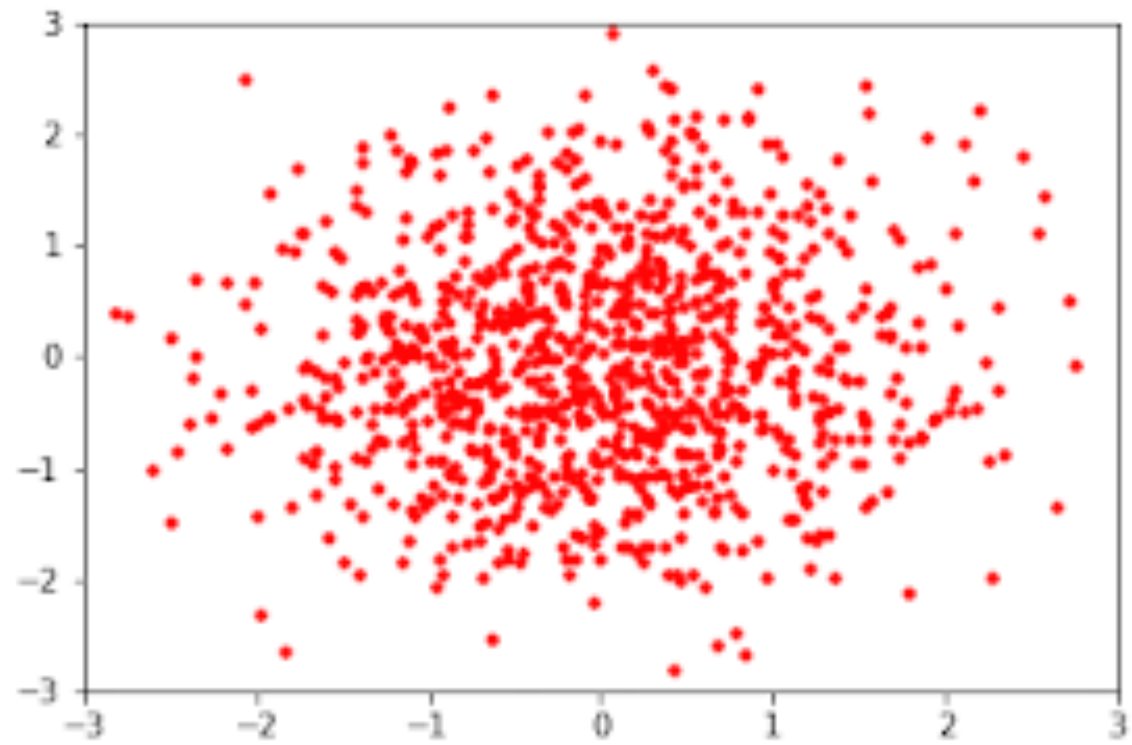
Contribution from Jacobian determinant

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

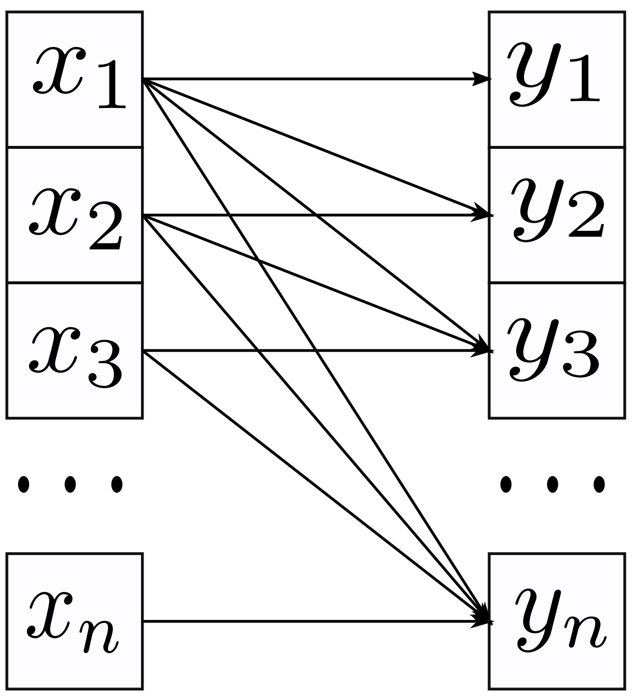
$$-\log(\det \mathbf{J}) = -\sum s(\mathbf{x})$$

1505.05770, 1908.09257

Animation



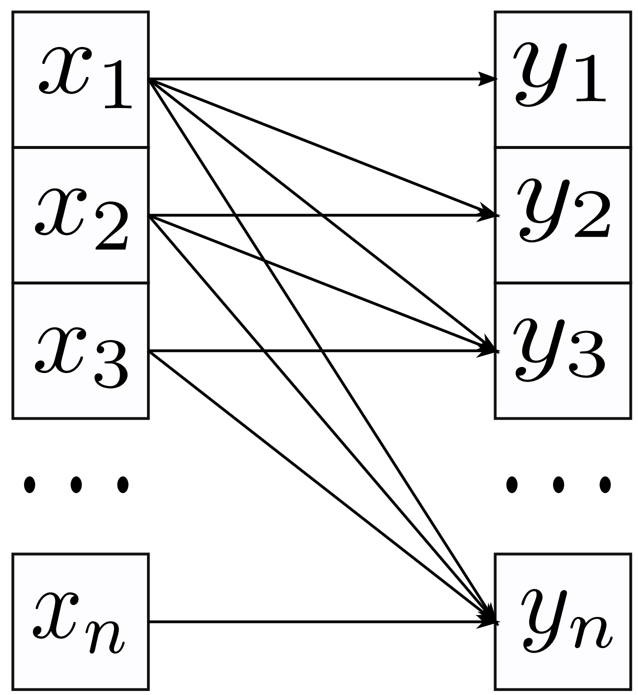
Autoregressive Flows



Alternative to coupling flows:
Outputs conditioned on previous inputs

1505.05770, 1908.09257

Autoregressive Flows



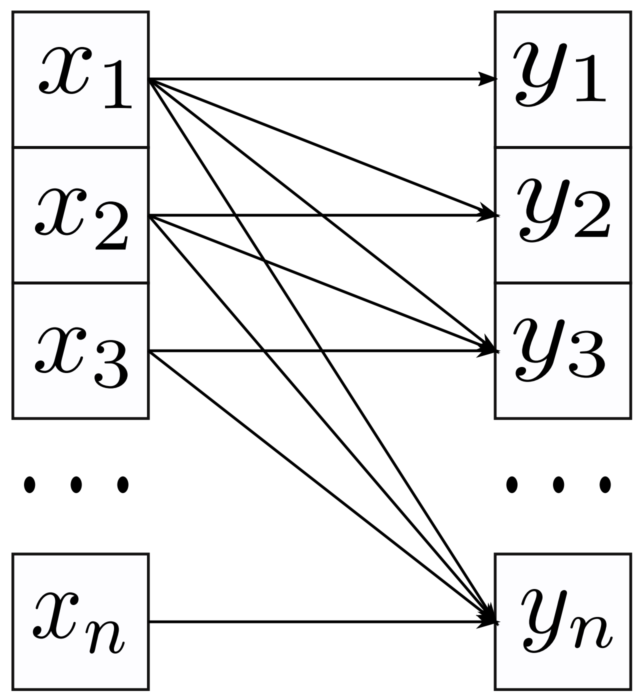
Bijective function   Parameters

$$y_t = h(x_t; \Theta_t(\mathbf{x}_{1:t-1}))$$

Alternative to coupling flows:
Outputs conditioned on previous inputs

Again: simple Jacobian and invertible functions

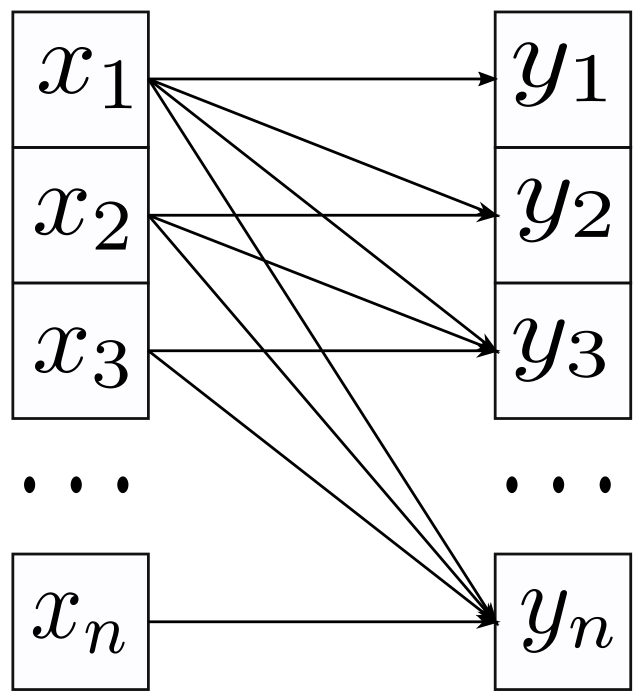
Autoregressive Flows

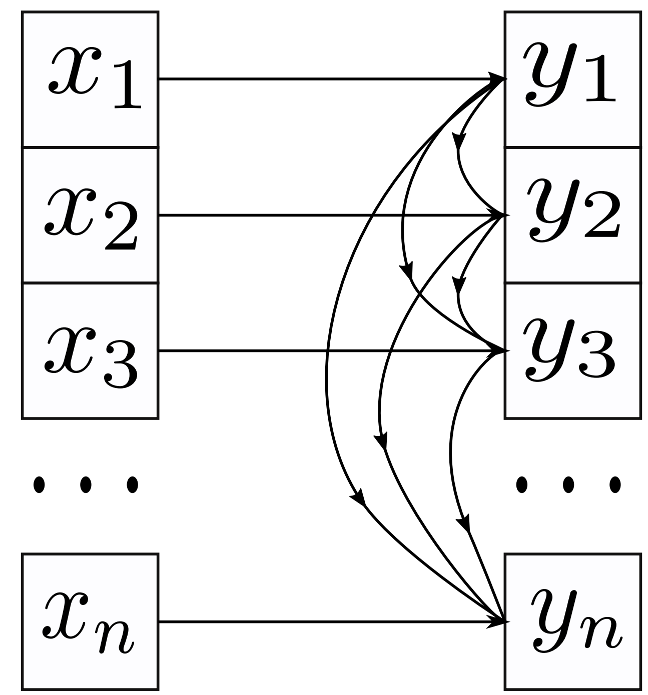


$$y_t = h(x_t; \Theta_t(\mathbf{x}_{1:t-1}))$$

Masked autoregressive flow (MAF):
Fast: Data → latent space
Slow: Latent space → data

Autoregressive Flows





$$y_t = h(x_t; \Theta_t(\mathbf{x}_{1:t-1}))$$

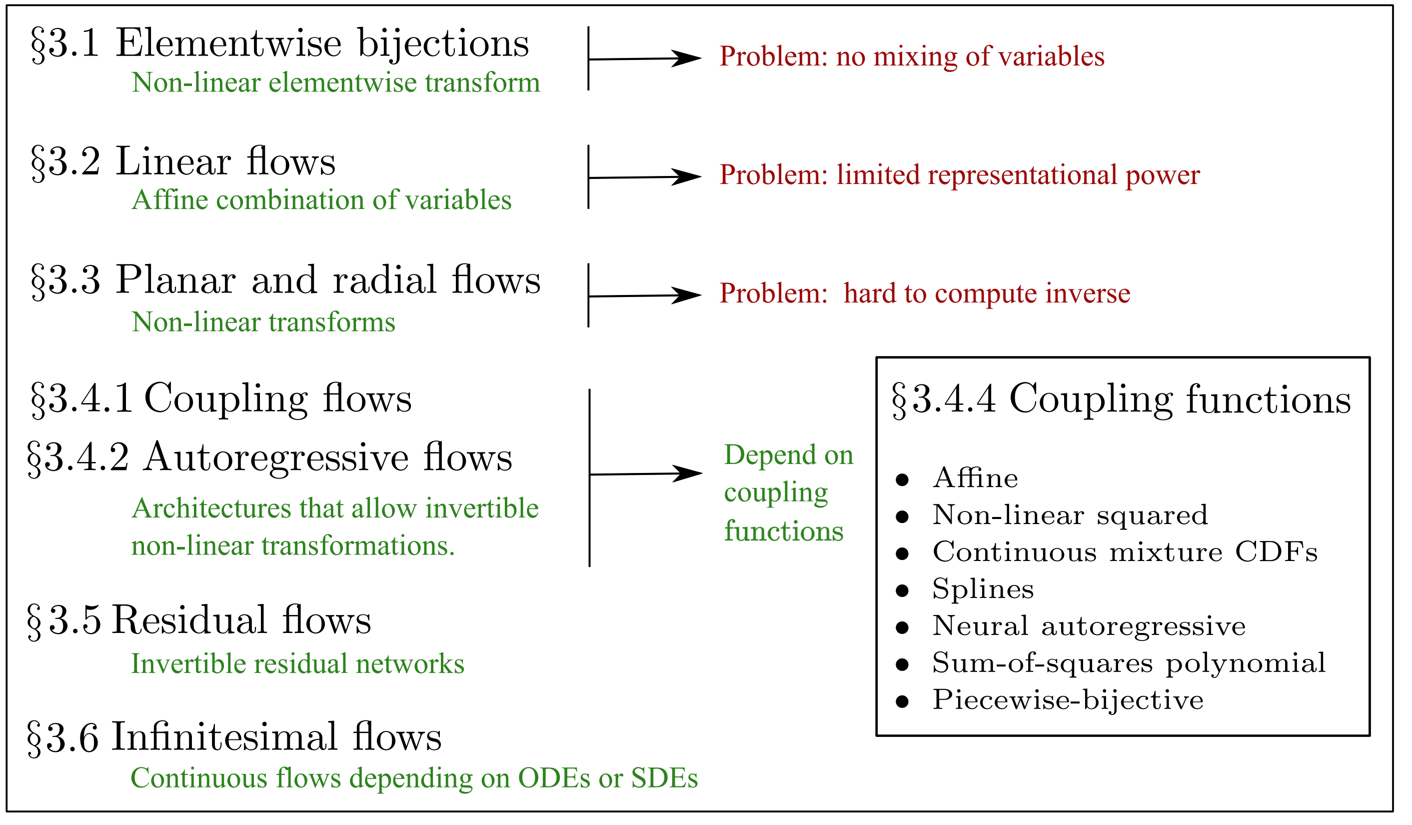
$$y_t = h(x_t; \theta_t(\mathbf{y}_{1:t-1}))$$

Masked autoregressive flow (MAF):
Fast: Data → latent space
Slow: Latent space → data

Inverse autoregressive flow (IAF):
Slow: Data → latent space
Fast: Latent space → data

Comments on Flows

Only scratched the surface:
more constructions available



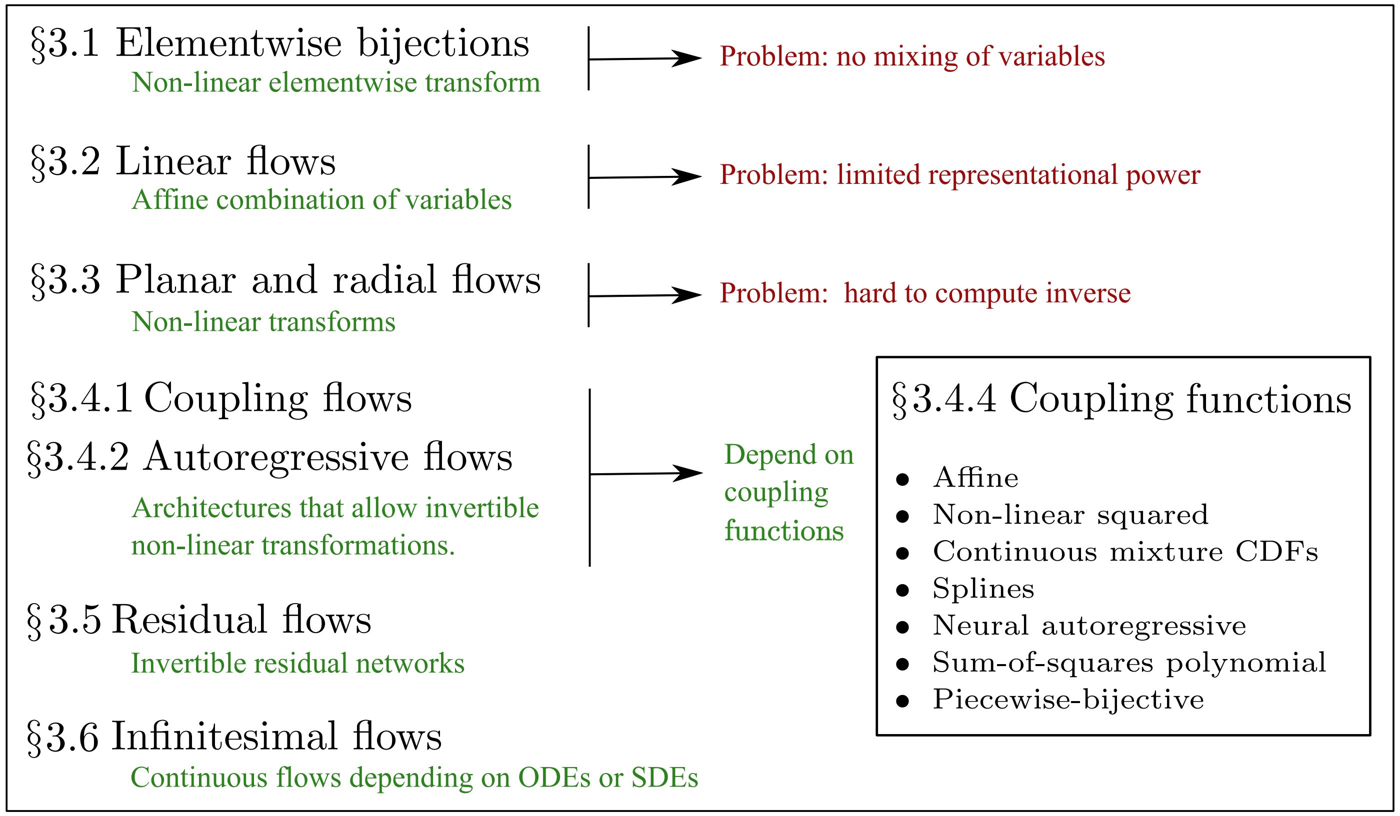
1908.09257

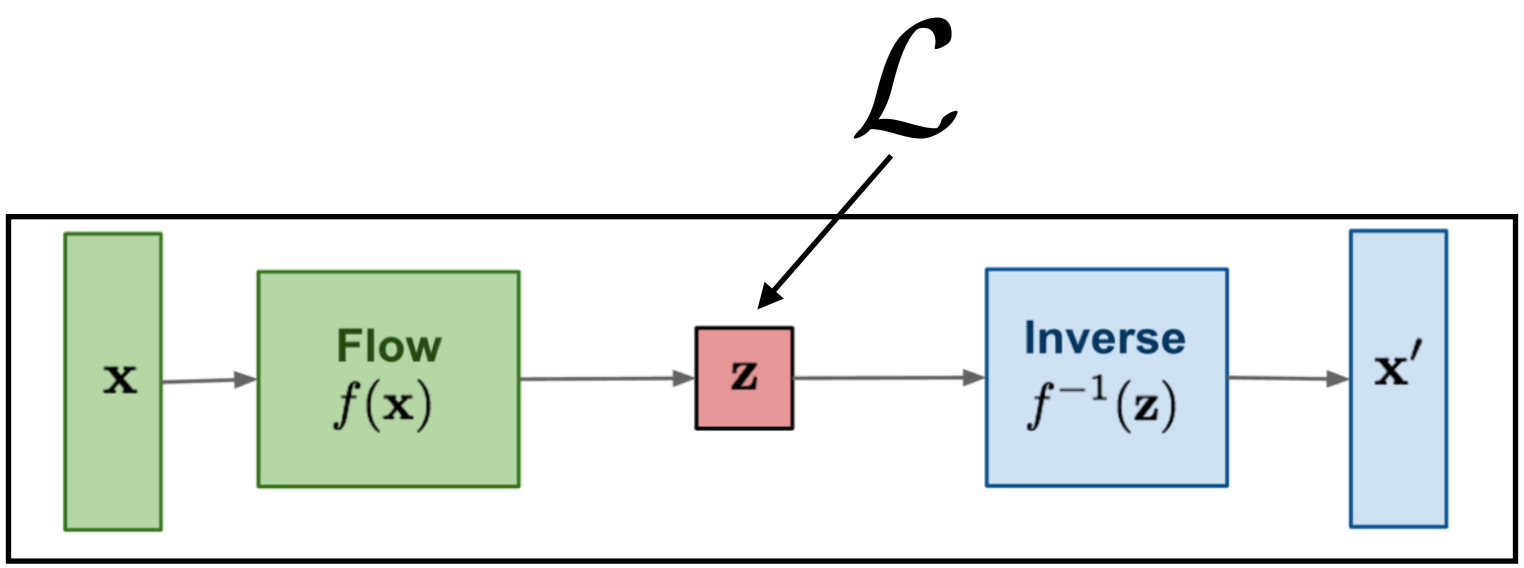
Comments on Flows

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

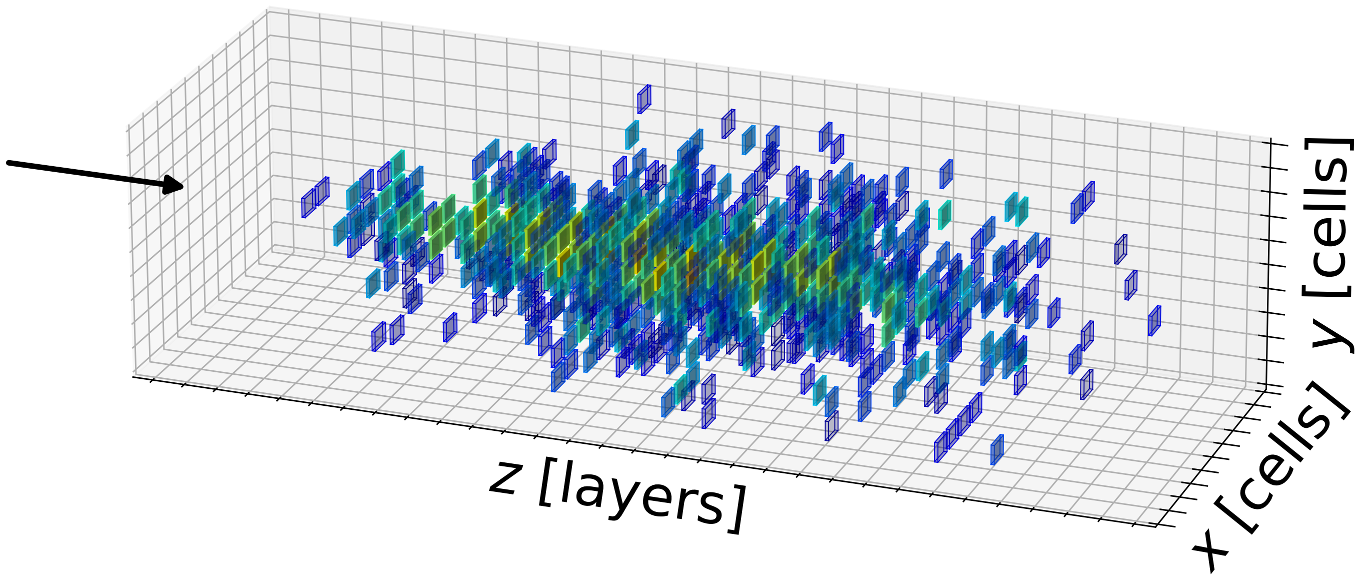




1908.09257

Applications II

Generative results II

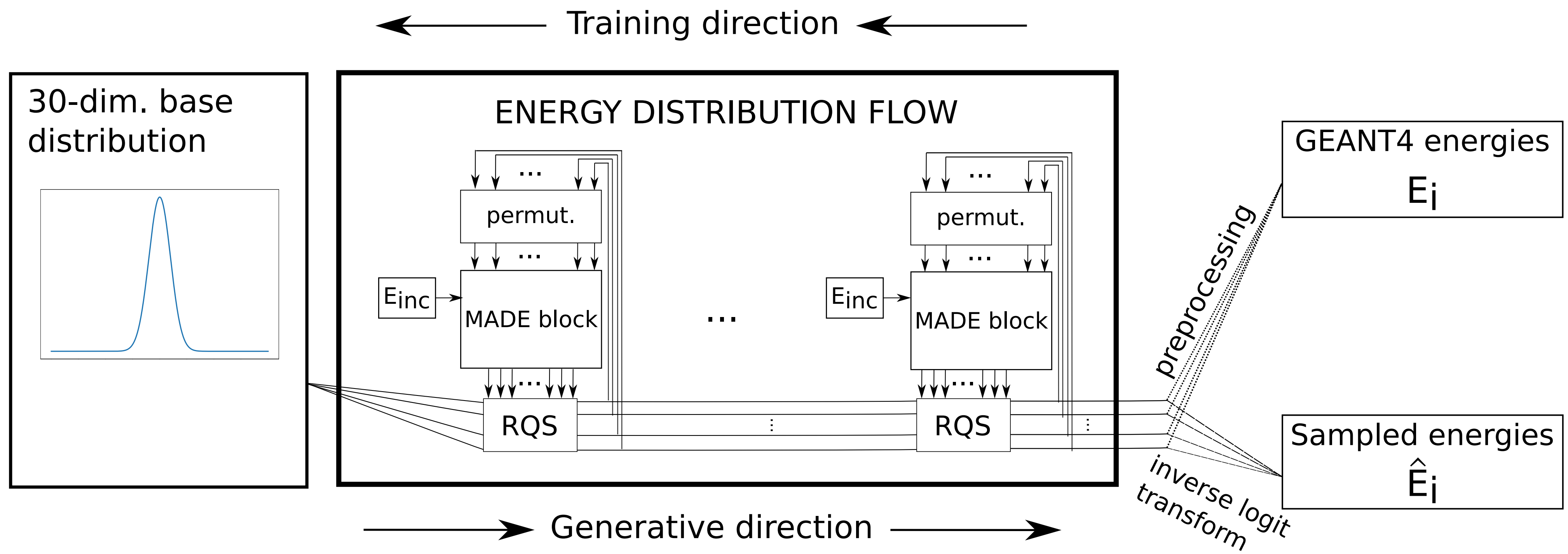


10x10 cells / layer
30 layers

How to flows for high-dimensional data?

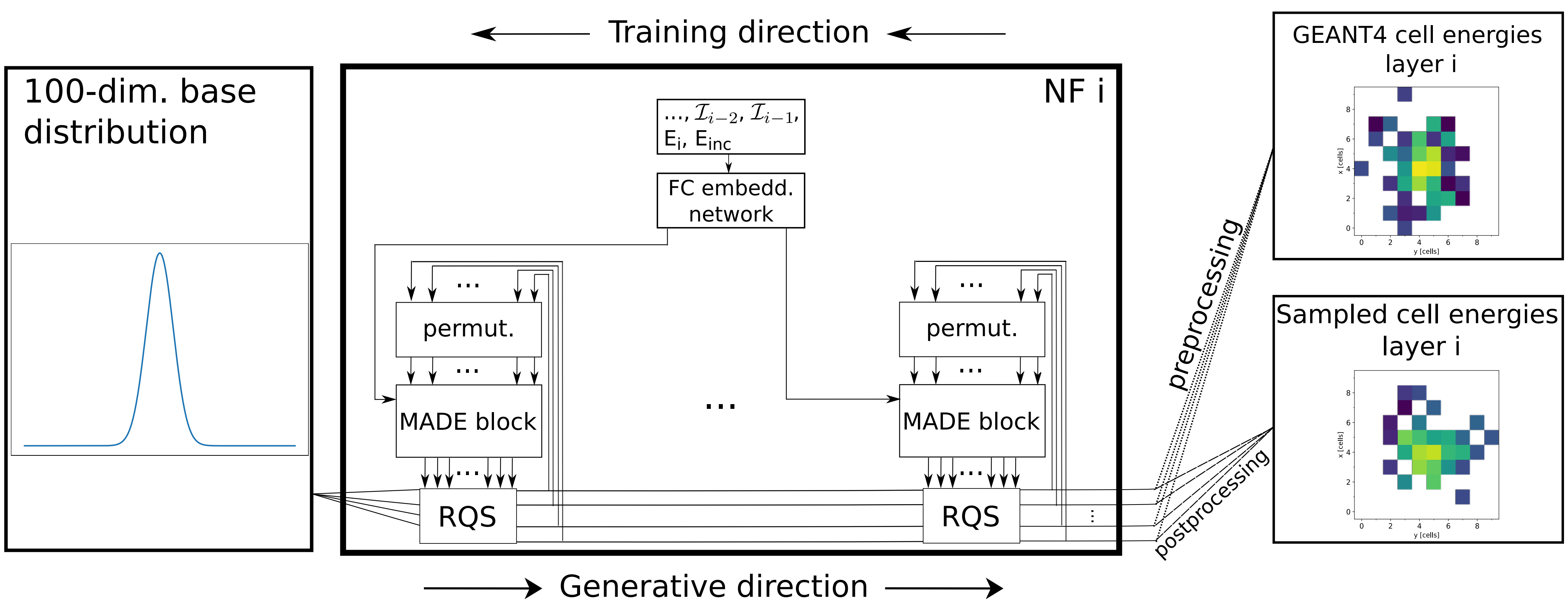
2302.11594

Generative results II



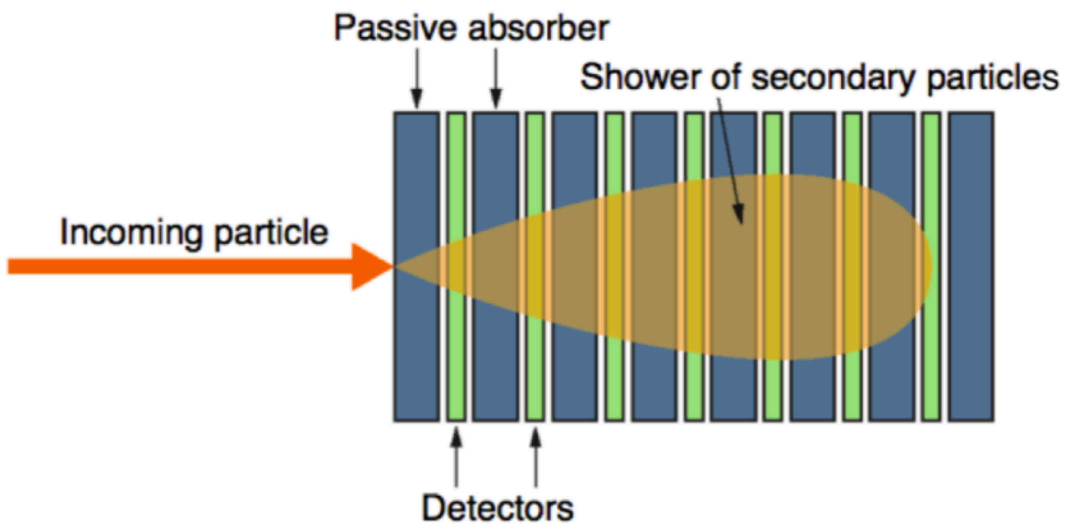
How to flows for high-dimensional data?

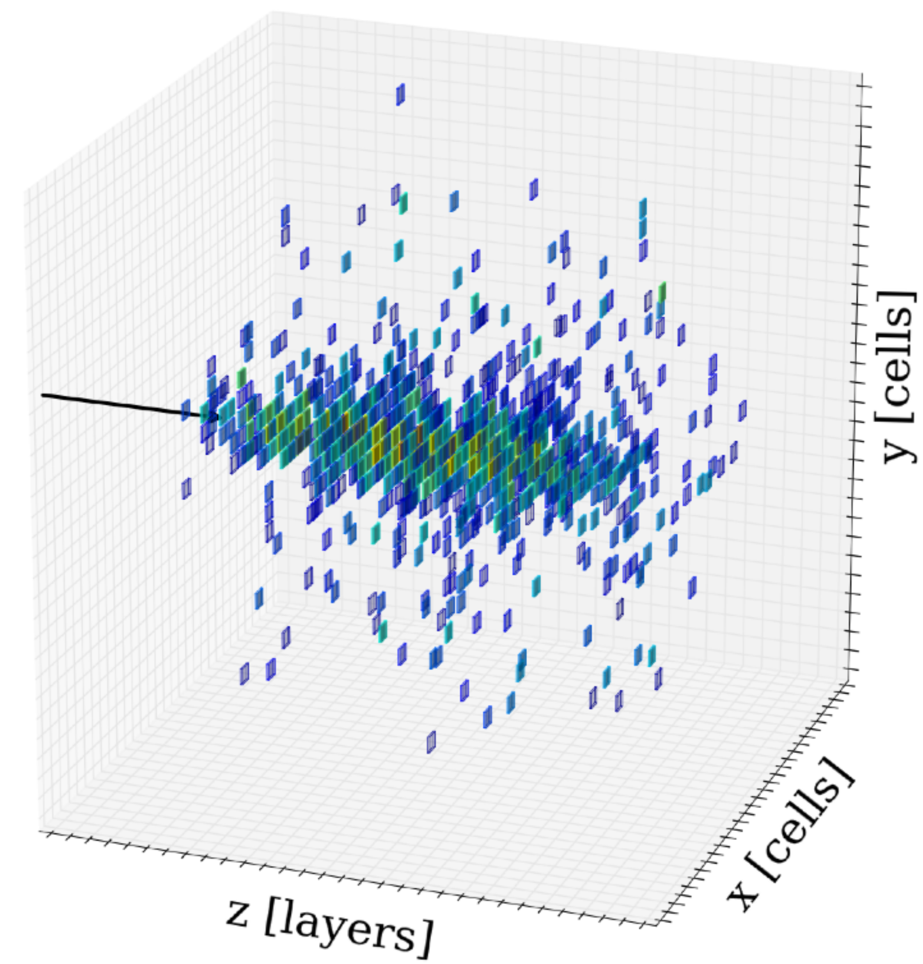
Split!



2302.11594

Simulation targets





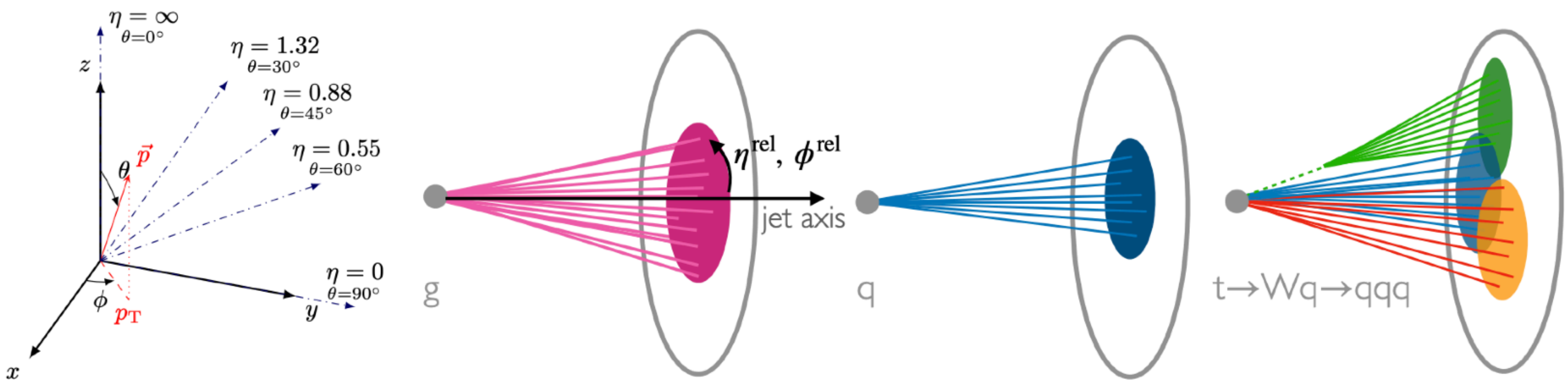
How to represent?

Tabular data

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

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

How to represent?

Tabular data

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

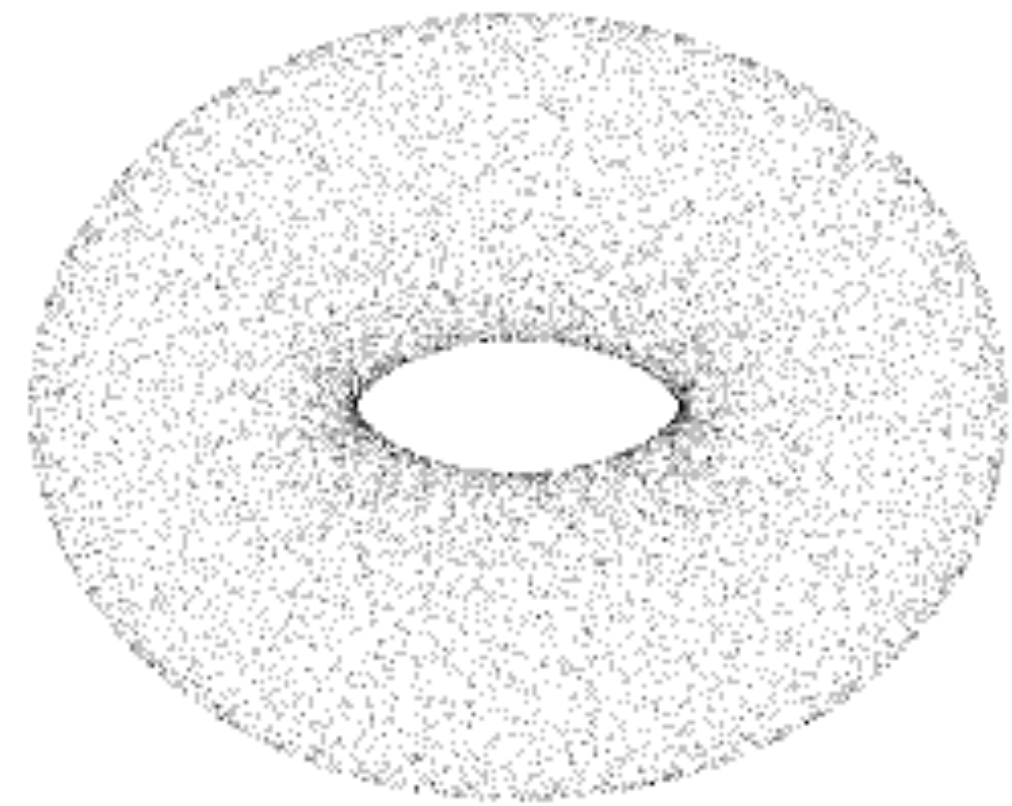
Point clouds / graphs

Why?
Useful stepping stone
In-situ background

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



Examples

Total data $\{x^j\}_{j=1 \dots N}$

with $x^j = \{\vec{p}_1^{\,j}, \dots \vec{p}_i^{\,j}, \dots, \vec{P}_{L(j)}^{\,j}\}$

and $\vec{p}_i \in \mathbb{R}^D$

Deep Sets

Theorem 7 *Let $f : [0,1]^M \to \mathbb{R}$ be a permutation invariant continuous function iff it has the representation*

$$f(x_1, ..., x_M) = \rho\left(\sum_{m=1}^{M} \phi(x_m)\right) \tag{18}$$

for some continuous outer and inner function $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ respectively. The inner function ϕ is independent of the function f.

$$X = \left.\begin{matrix} \vec{p}_1 = \{r_1, \theta_1, \varphi_1, T_1\} \\ \vdots \qquad\qquad \vdots \\ \vec{p}_L = \{r_L, \theta_L, \varphi_L, T_L\} \end{matrix}\right\} \qquad f(x) = \rho\left(\Sigma_i\, \phi(\vec{p}_i)\right)$$

$$\Uparrow$$

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

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

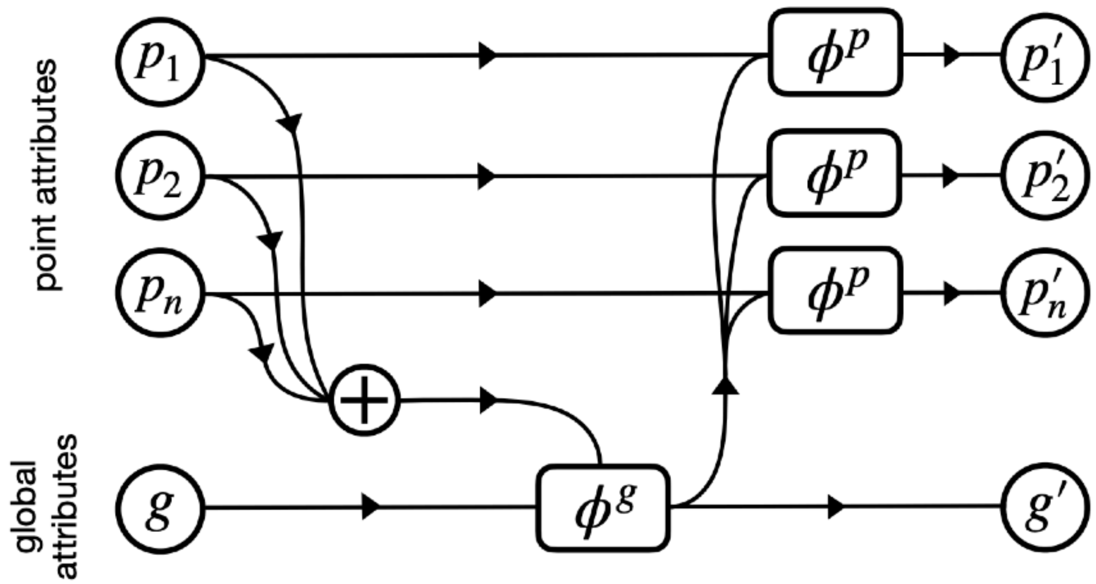
Latent Space Dimension

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

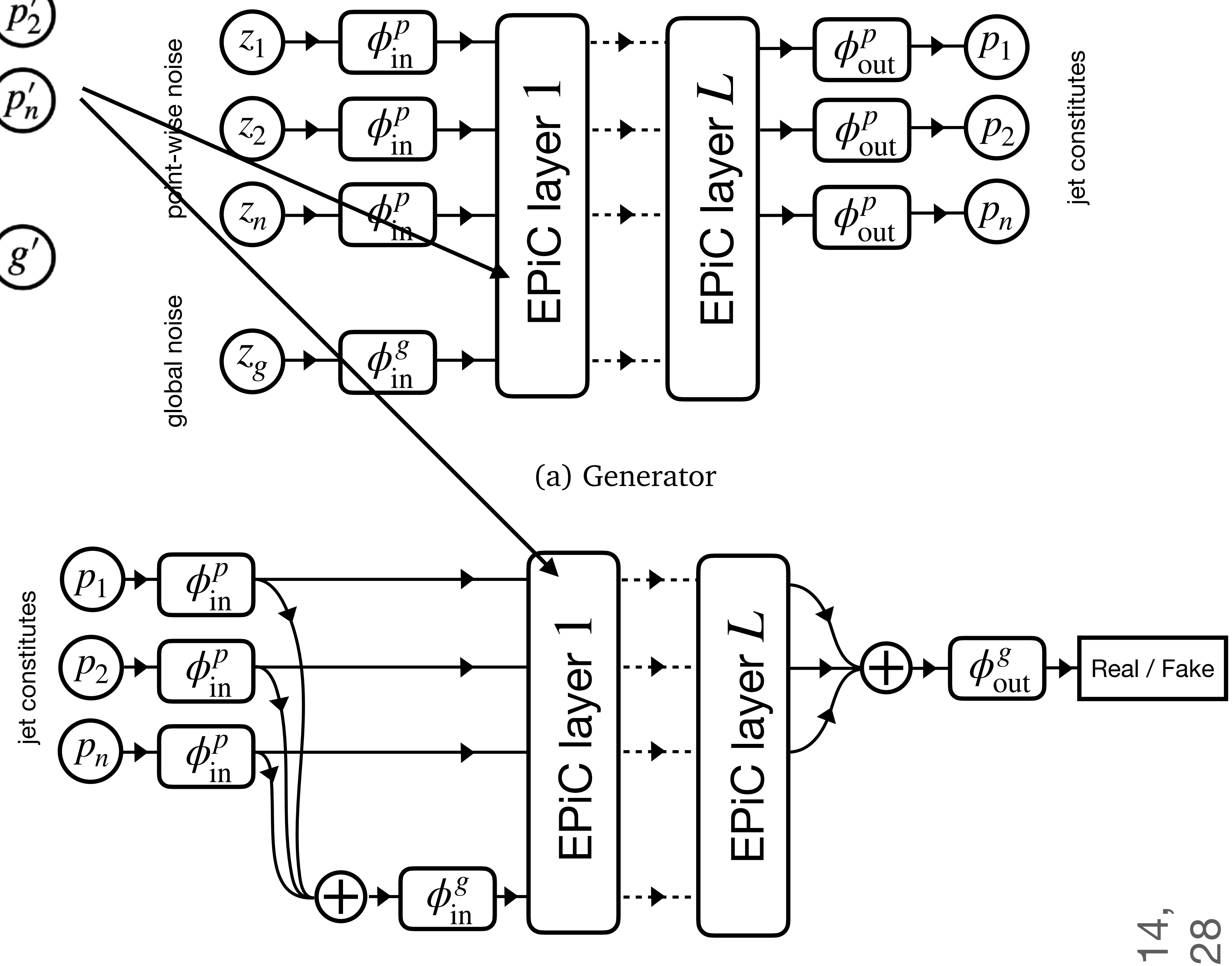
$$\rho : \mathbb{R}^S \to \mathbb{R}^1$$

1703.06114

How to GAN with it



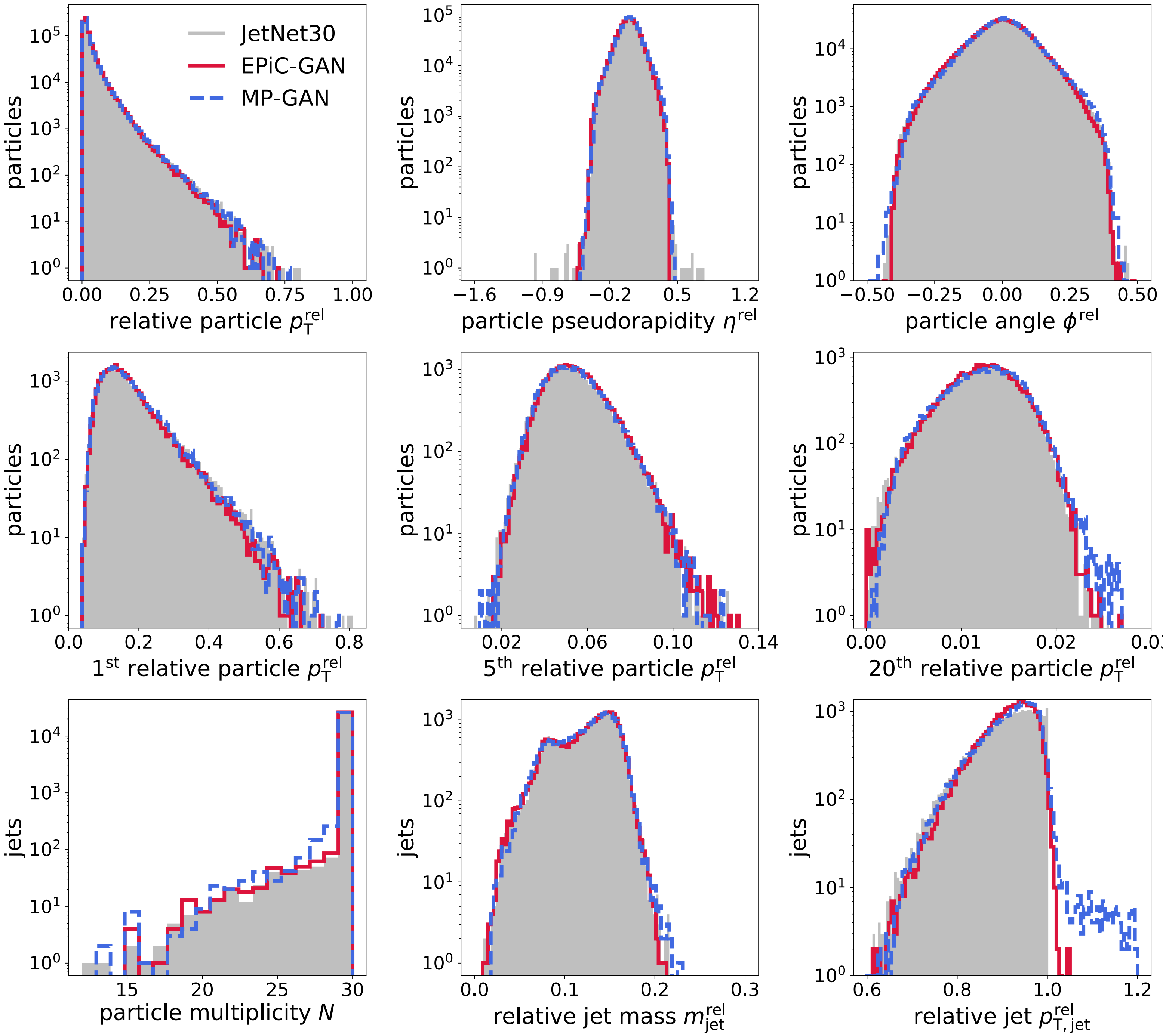
Add permutation symmetry to GAN architecture

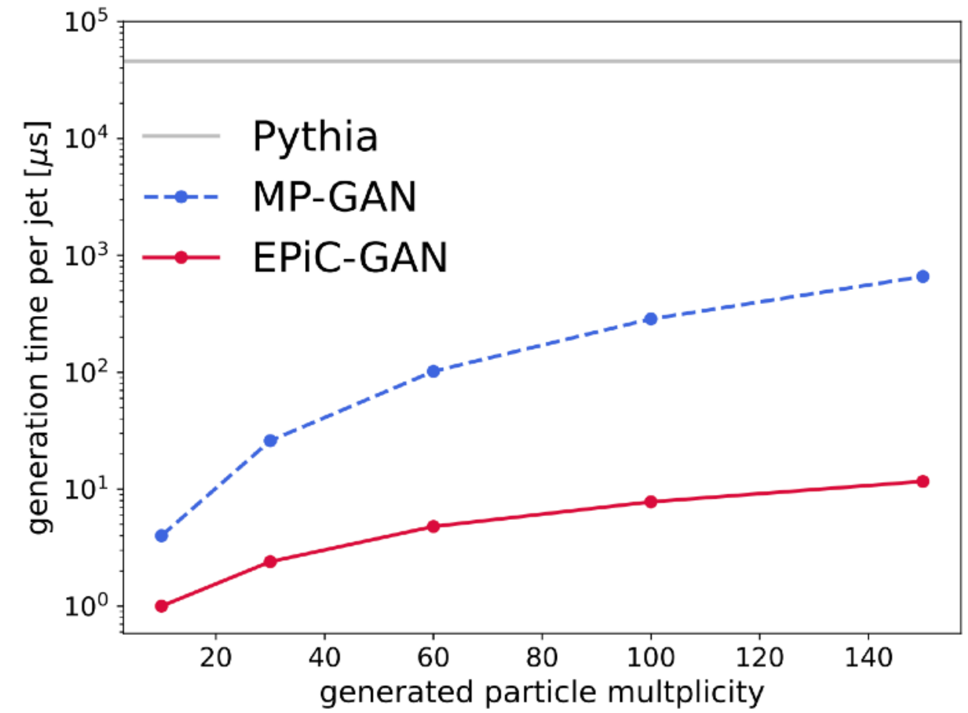


(b) Discriminator

1703.06114, 2301.08128

Generative results III



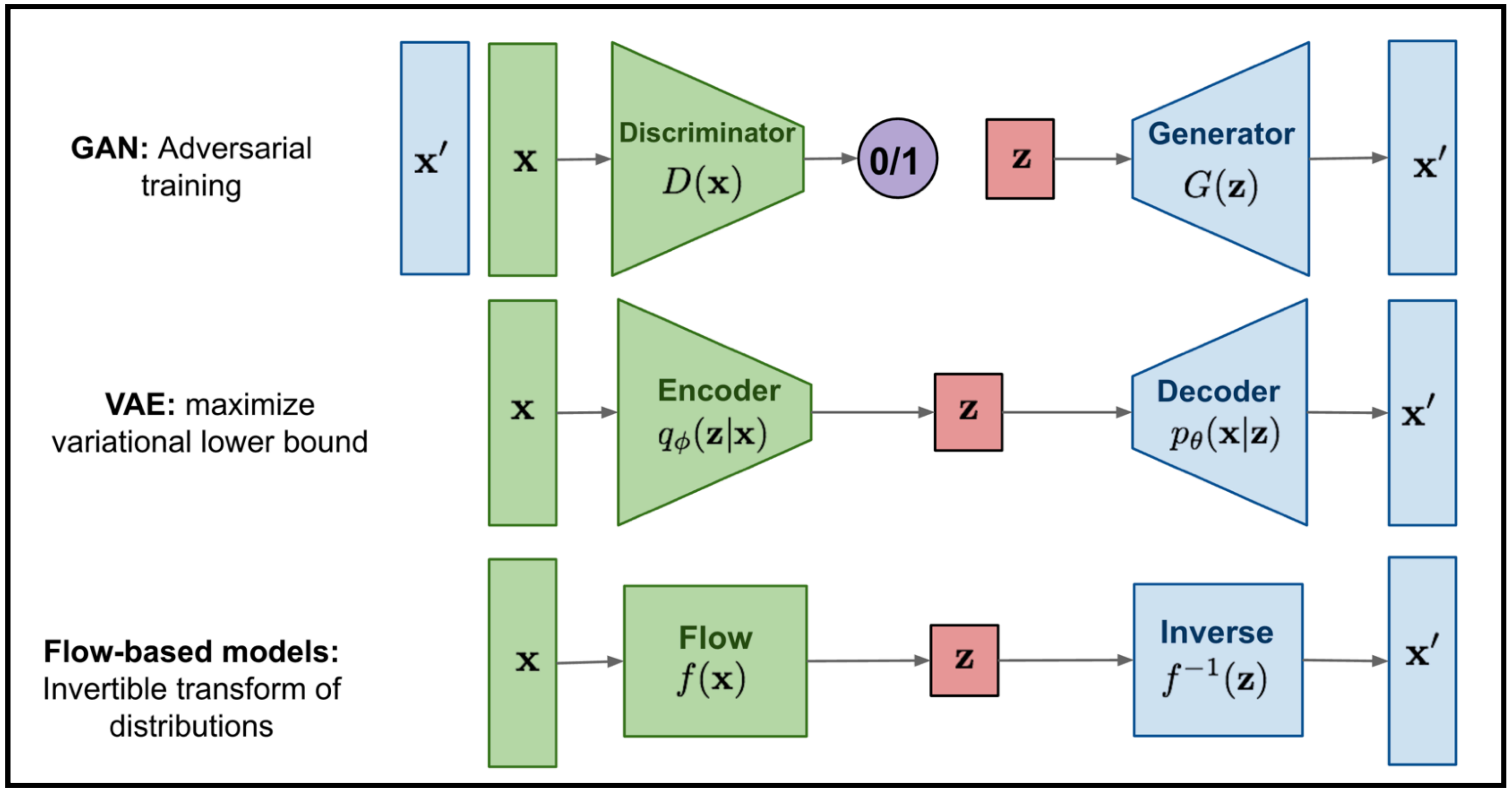


Point clouds/graph GANs successfully generate jet constituents

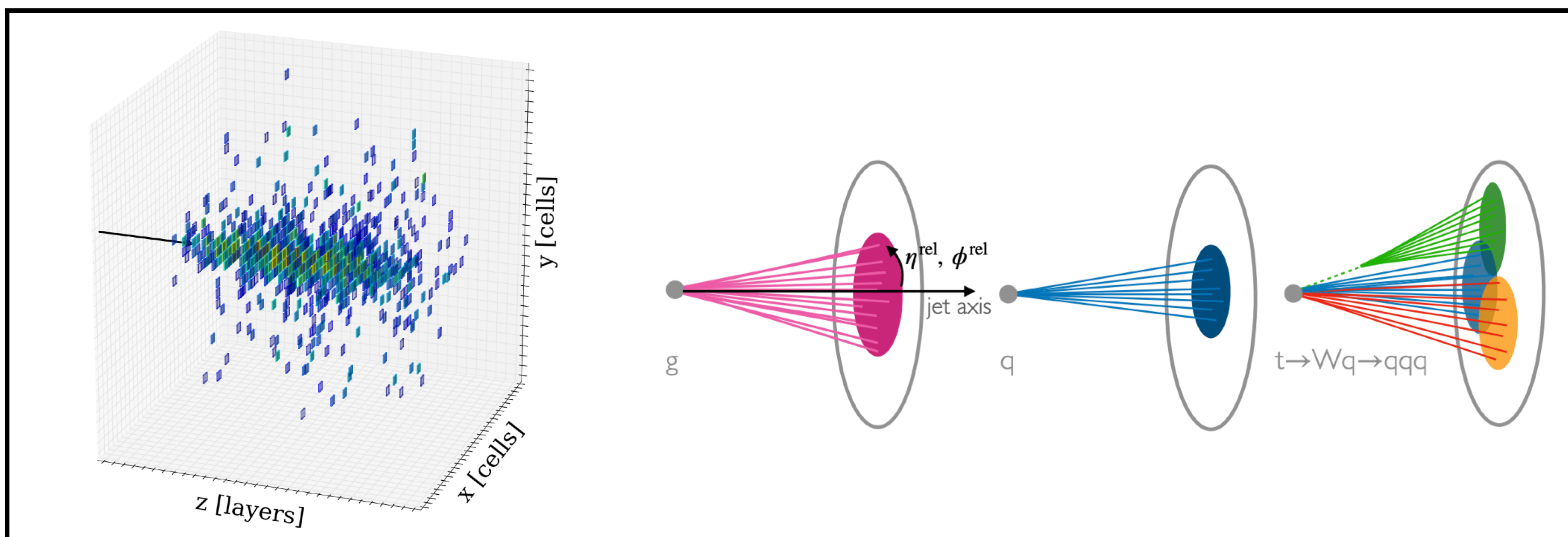
2301.08128

Closing I

Closing



Understand three basic generative architectures



First look at simulating fixed grid and point-cloud data