Deep generative models

A latent variable model perspective

51st SLAC Summer Institute

August 15, 2023

Gilles Louppe
g.louppe@uliege.be

Outline

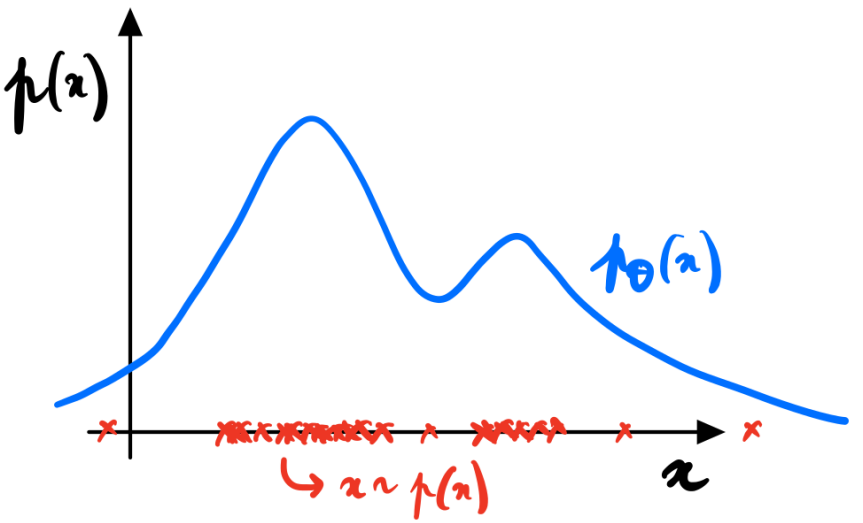
1. Deep generative models
2. Variational auto-encoders
3. Diffusion models
4. Normalizing flows

Deep generative models

Generative models

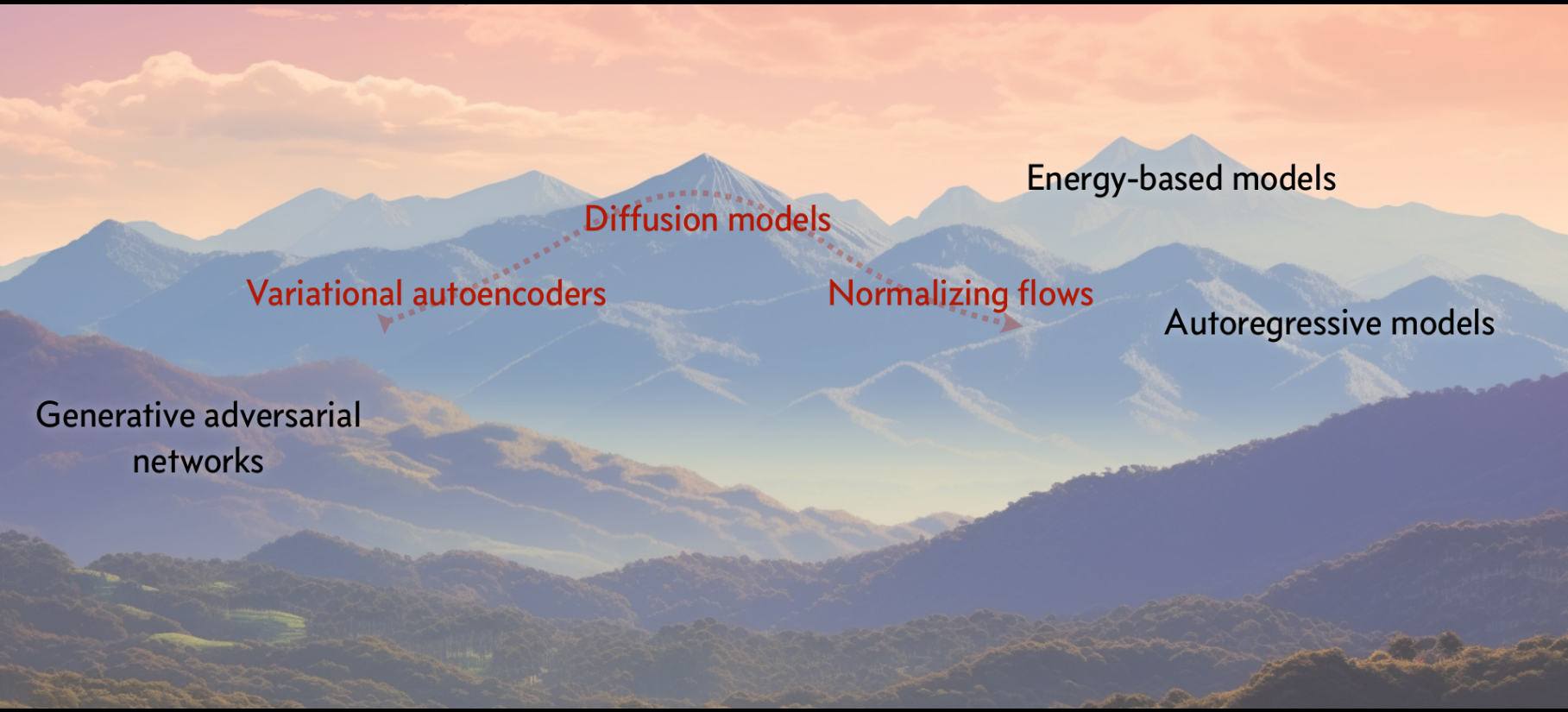
A (deep) generative model is a probabilistic model p_θ that can be used as a simulator of the data.

Formally, a generative model defines a probability distribution $p_\theta(\mathbf{x})$ over the data $\mathbf{x} \in \mathcal{X}$, parameterized by θ.



Variational auto-encoders
(Kingma and Welling, 2013)

Diffusion models
(Midjourney, 2023)



—

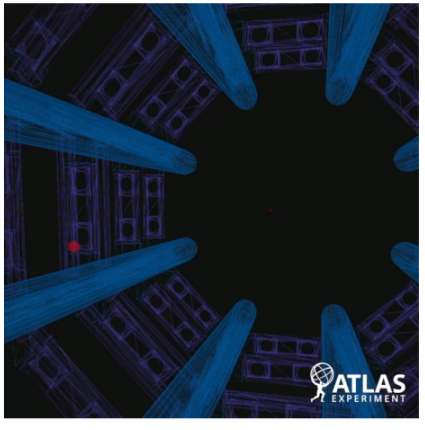
Credits: Karsten et al, 2022; Siddharth Mishra-Sharma, 2023.

Simulators

A simulator prescribes a generative model that can be used to simulate data $\mathbf{x}$.

Collider data

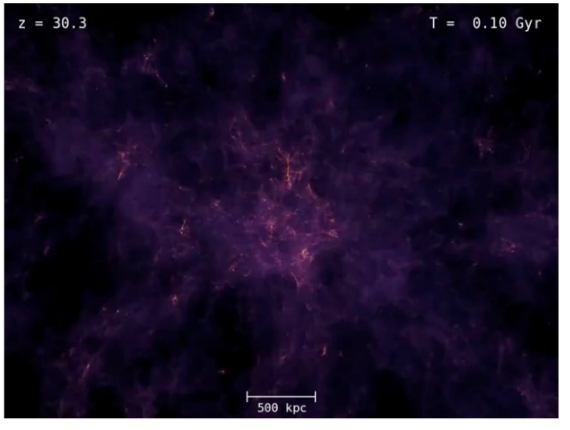
$\text{particles} \sim p(\text{particles})$



[C. Cesarotti with ATLAS]

Cosmology data

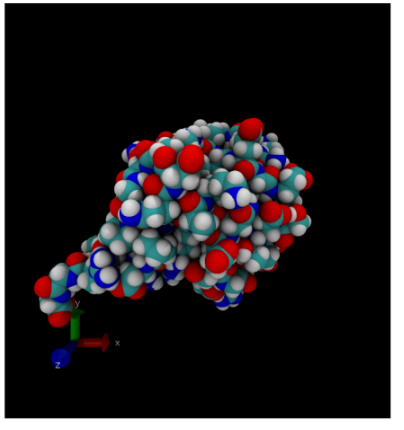
$\text{particles} \sim p(\text{particles})$



[Aquarius simulation]

Molecular dynamics

$\text{configurations} \sim p(\text{configurations})$



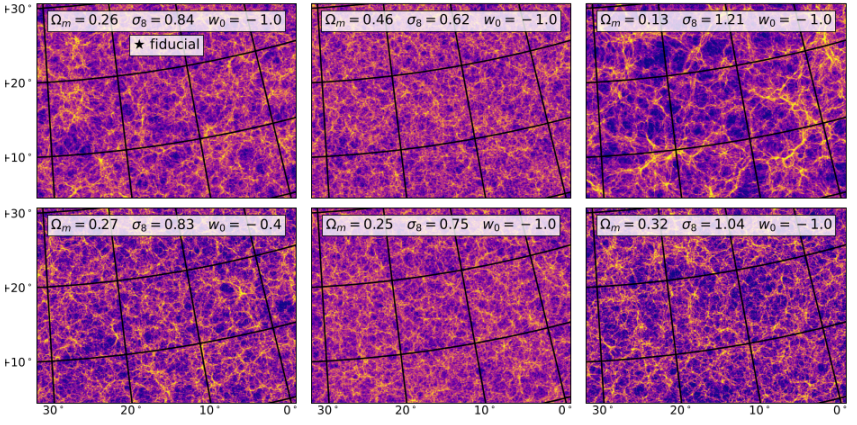
[E. Cances et al]

Conditional simulators

A conditional simulator prescribes a way to sample from the likelihood $p(\mathbf{x}|\vartheta)$, where ϑ is a set of conditioning variables or parameters.

Cosmology data

$$\text{map} \sim p(\text{map} \mid \{\Omega_m, \sigma_8, w_0\})$$



[Kacprzak et al 2022]

$$x \sim p(x; \mathscr{M})$$

Model

or

$$x \sim p(x \mid \theta)$$

Model parameters

$$p(z_p|\vartheta)$$

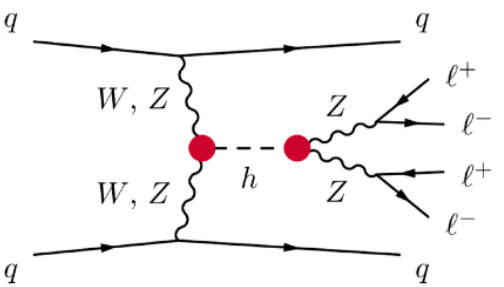
Latent variables

Parameters of interest

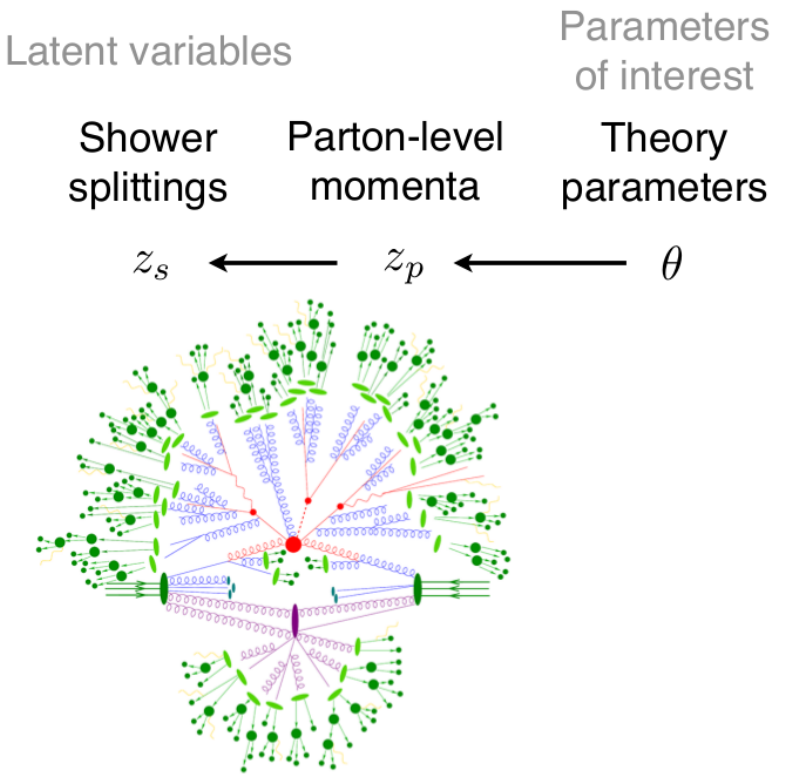
Parton-level momenta

Theory parameters

$z_p \longleftarrow \theta$



$$p(z_s|\vartheta) = \int p(z_p|\vartheta) p(z_s|z_p) dz_p$$



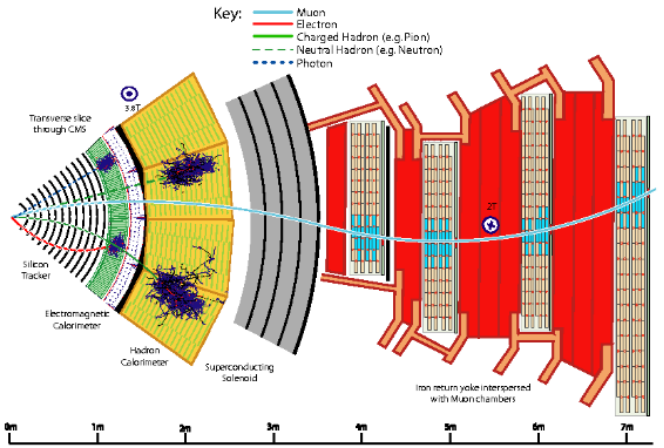
$$p(z_d|\vartheta) = \iint p(z_p|\vartheta)p(z_s|z_p)p(z_d|z_s)dz_p dz_s$$

Latent variables

Parameters of interest

Detector interactions ← Shower splittings ← Parton-level momenta ← Theory parameters

$z_d \longleftarrow z_s \longleftarrow z_p \longleftarrow \theta$

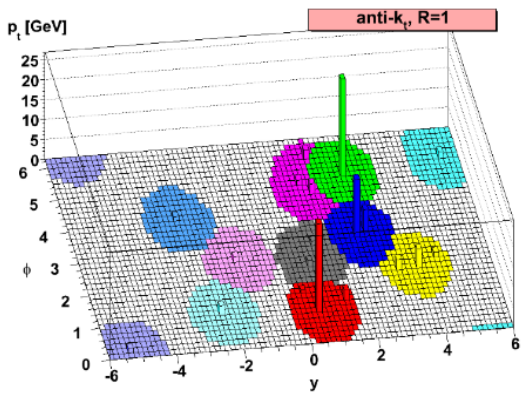


$$p(x|\vartheta) = \iiint p(z_p|\vartheta)p(z_s|z_p)p(z_d|z_s)p(x|z_d)dz_p dz_s dx$$

Features | Latent variables | Parameters of interest

Observables | Detector interactions | Shower splittings | Parton-level momenta | Theory parameters

$x \leftarrow z_d \leftarrow z_s \leftarrow z_p \leftarrow \theta$



[Image source: M. Cacciari, G. Salam, G. Soyez 0802.1189]

What can we do with generative models?

Produce samples

$$\mathbf{x} \sim p(\mathbf{x}|\vartheta)$$

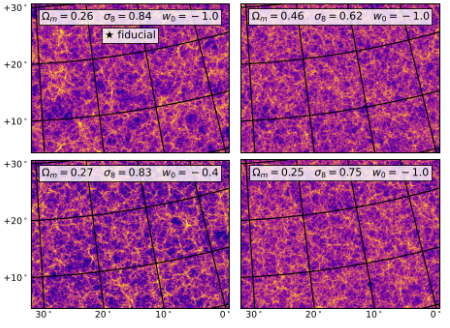
Evaluate densities

$$p(\mathbf{x}|\vartheta)$$

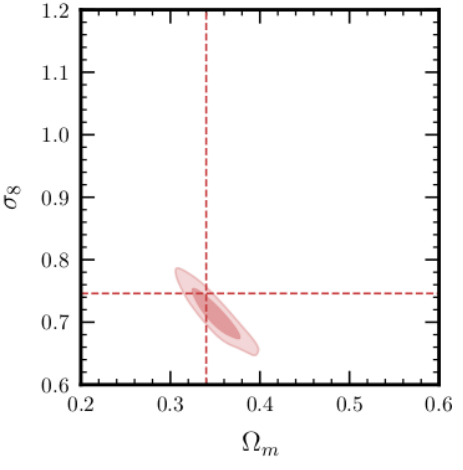
Encode complex priors

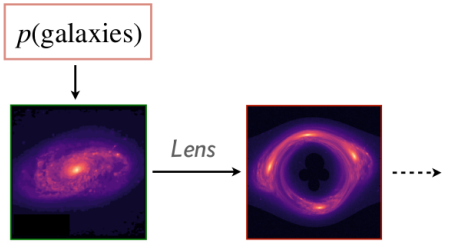
$$p(\mathbf{x})$$

$$p(\vartheta|\mathbf{x}) = \frac{p(\mathbf{x}|\vartheta)p(\vartheta)}{p(\mathbf{x})}$$



[Kacprzak et al 2022]

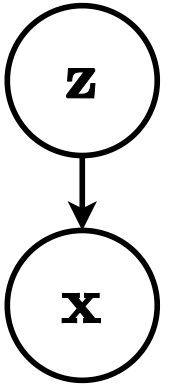




—

Variational auto-encoders

Latent variable model



Consider for now a prescribed latent variable model that relates a set of observable variables $\mathbf{x} \in \mathcal{X}$ to a set of unobserved variables $\mathbf{z} \in \mathcal{Z}$.

The probabilistic model defines a joint probability distribution $p_\theta(\mathbf{x}, \mathbf{z})$, which decomposes as

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}).$$

How to fit a latent variable model?

$$
\begin{aligned}
\theta^* &= \arg\max_\theta p_\theta(\mathbf{x}) \\
&= \arg\max_\theta \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \arg\max_\theta \mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right] d\mathbf{z} \\
&\approx \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} p_\theta(\mathbf{x}|\mathbf{z}_i)
\end{aligned}
$$

How to fit a latent variable model?

$$
\begin{aligned}
\theta^* &= \arg\max_\theta p_\theta(\mathbf{x}) \\
&= \arg\max_\theta \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \arg\max_\theta \mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right] d\mathbf{z} \\
&\approx \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} p_\theta(\mathbf{x}|\mathbf{z}_i)
\end{aligned}
$$

The curse of dimensionality will lead to poor estimates of the expectation.

Variational inference

Let us instead consider a variational approach to fit the model parameters θ.

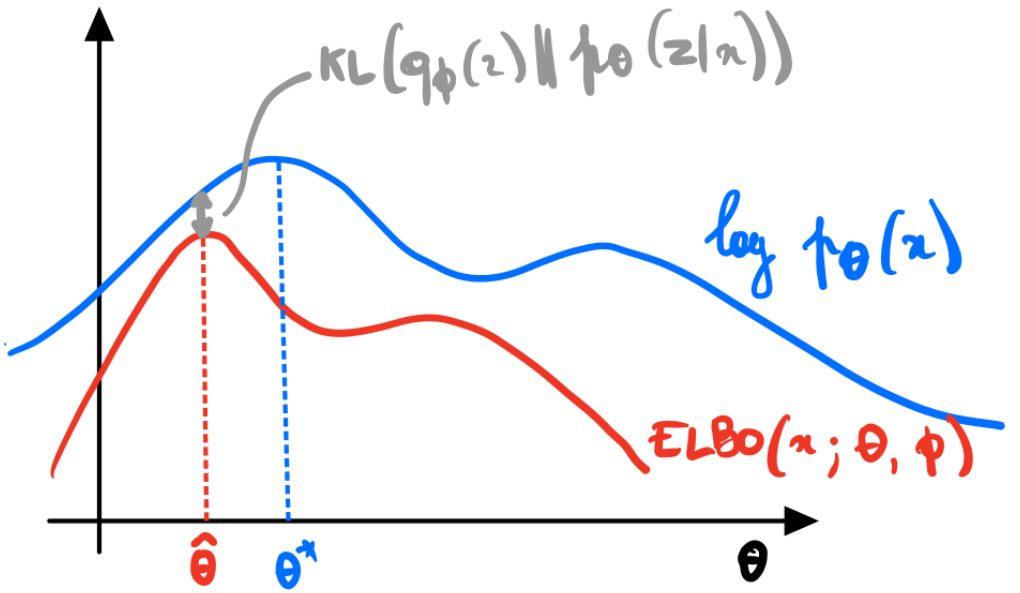
Using a variational distribution $q_\phi(\mathbf{z})$ over the latent variables $\mathbf{z}$, we have

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right] \\
&= \log \mathbb{E}_{q_\phi(\mathbf{z})} \left[\frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z})} \right] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z})} \right] \quad (\mathrm{ELBO}(\mathbf{x}; \theta, \phi)) \\
&= \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathrm{KL}(q_\phi(\mathbf{z}) || p(\mathbf{z}))
\end{aligned}$$

Using the Bayes rule, we can also write

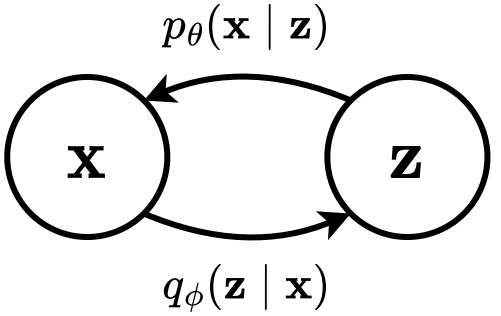
$$\begin{aligned}
\mathrm{ELBO}(\mathbf{x};\theta,\phi) &= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z})}\frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})}p_\theta(\mathbf{x})\right] \\
&= \log p_\theta(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})).
\end{aligned}$$

Therefore, $\log p_\theta(\mathbf{x}) = \mathrm{ELBO}(\mathbf{x};\theta,\phi) + \mathrm{KL}(q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x}))$.



Provided the KL gap remains small, the model parameters can now be optimized by maximizing the ELBO,

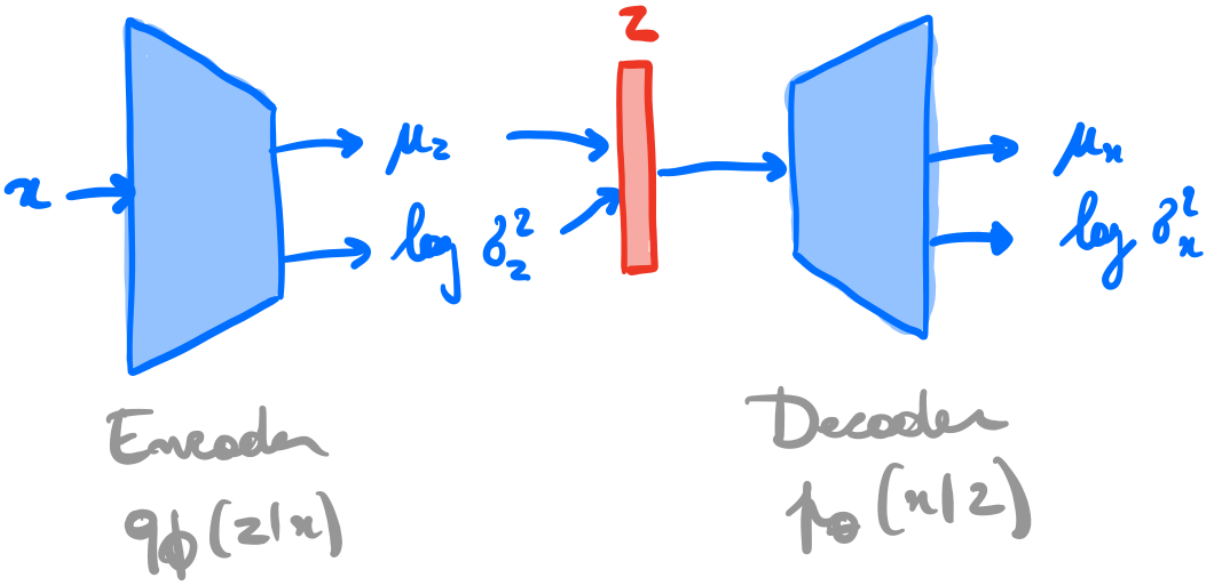
$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \mathrm{ELBO}(\mathbf{x}; \theta, \phi).$$



So far we assumed a prescribed probabilistic model motivated by domain knowledge. We will now directly learn a stochastic generating process $p_\theta(\mathbf{x}|\mathbf{z})$ with a neural network.

We will also amortize the inference process by learning a second neural network $q_\phi(\mathbf{z}|\mathbf{x})$ approximating the posterior, conditionally on the observed data $\mathbf{x}$.

Variational auto-encoders

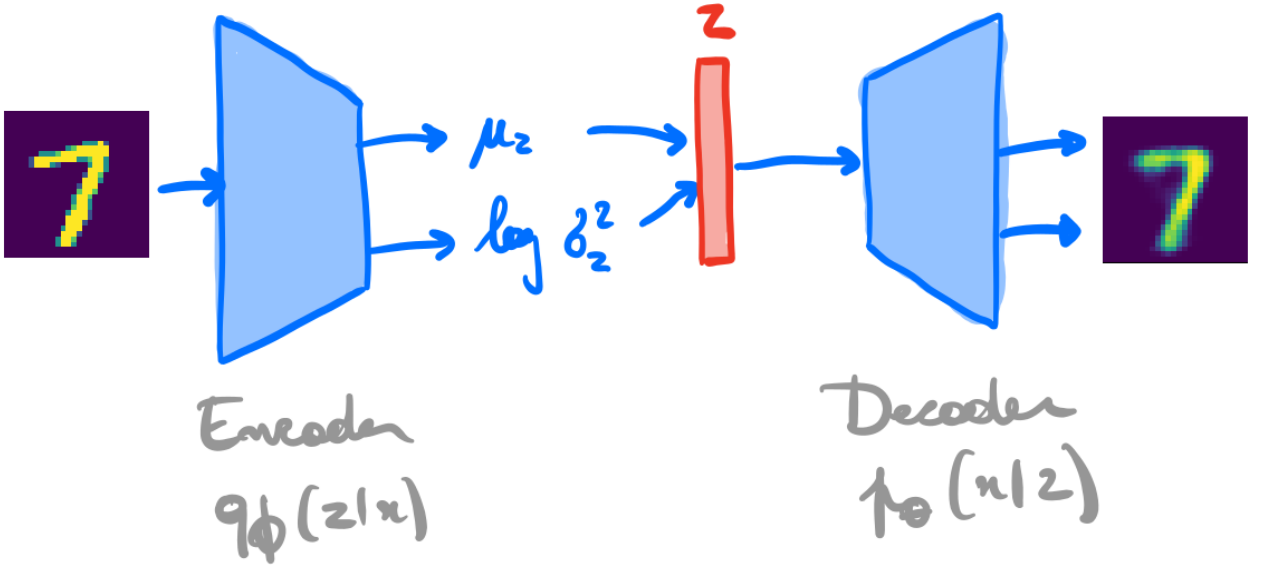


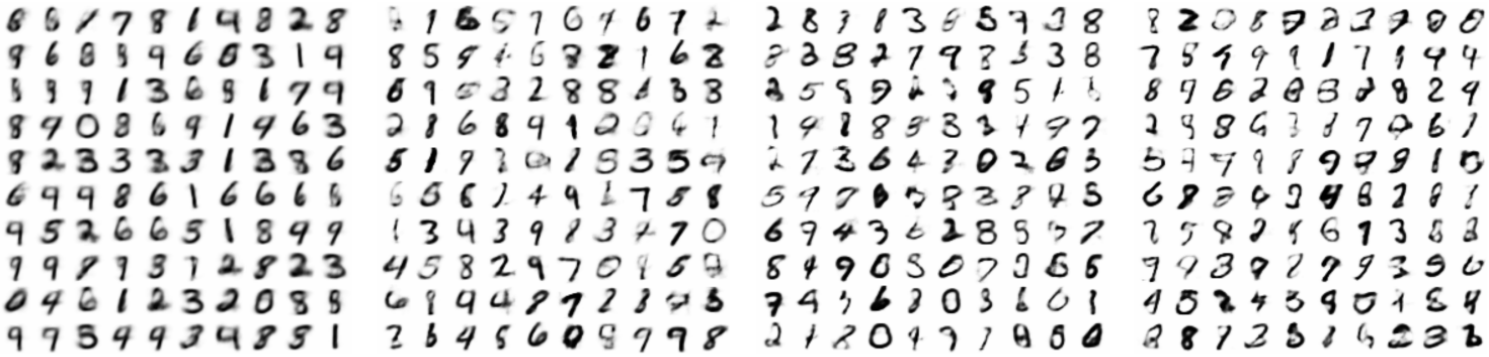
As before, we can use variational inference to jointly optimize the generative and the inference networks parameters θ and ϕ:

$$
\begin{aligned}
\theta^*, \phi^* &= \arg\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \left[\text{ELBO}(\mathbf{x}; \theta, \phi) \right] \\
&= \arg\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}] \right] \\
&= \arg\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z})) \right] .
\end{aligned}
$$

Step-by-step example

Consider as data $\mathbf{d}$ the MNIST digit dataset:


z
μ_z
$\log \sigma_z^2$
Encoder
$q_\phi(z|x)$
Decoder
$p_\theta(x|z)$

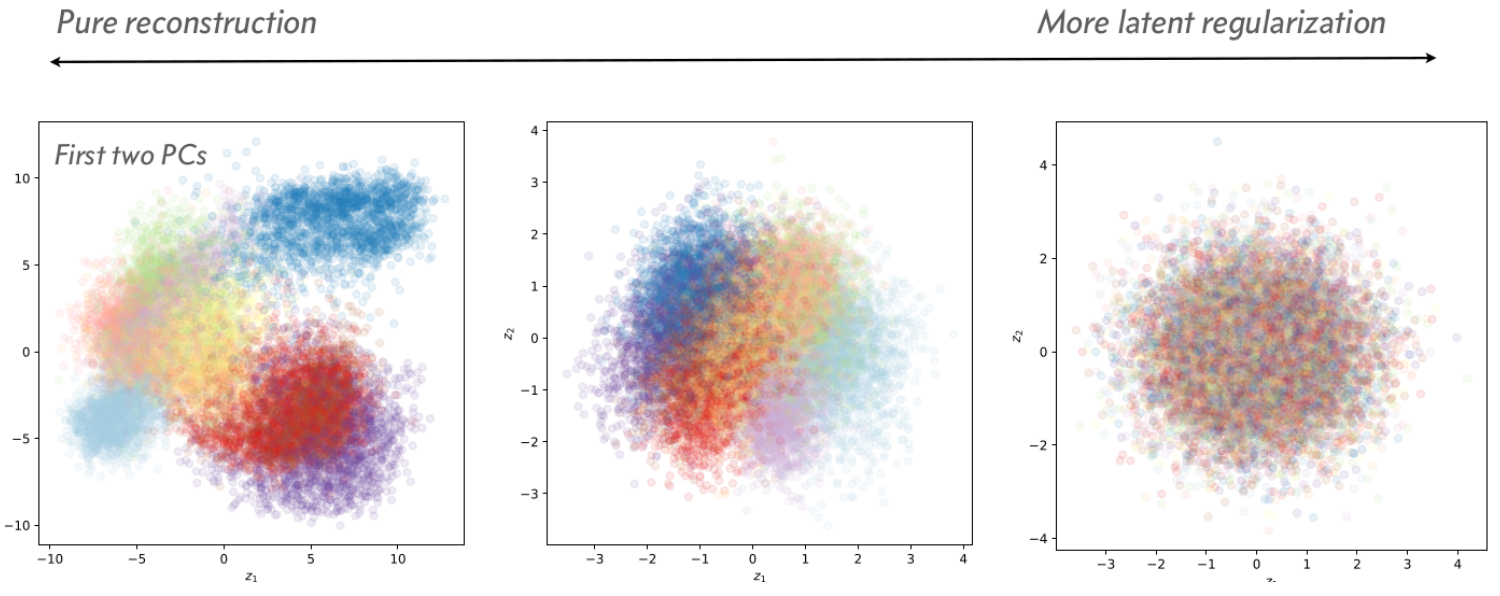


(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

(Kingma and Welling, 2013)

A semantically meaningful latent space

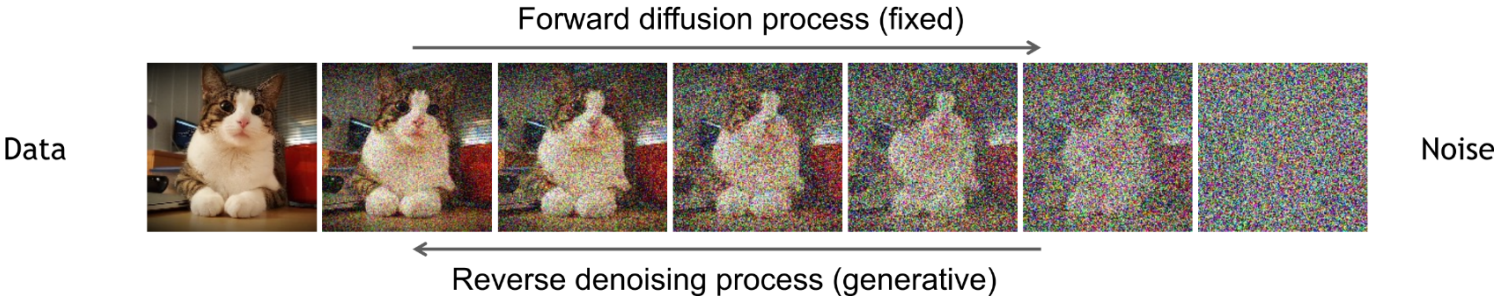
The prior-matching term $\mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$ enforces simplicity in the latent space, encouraging learned semantic structure and disentanglement.



—

Credits: Siddharth Mishra-Sharma, 2023.

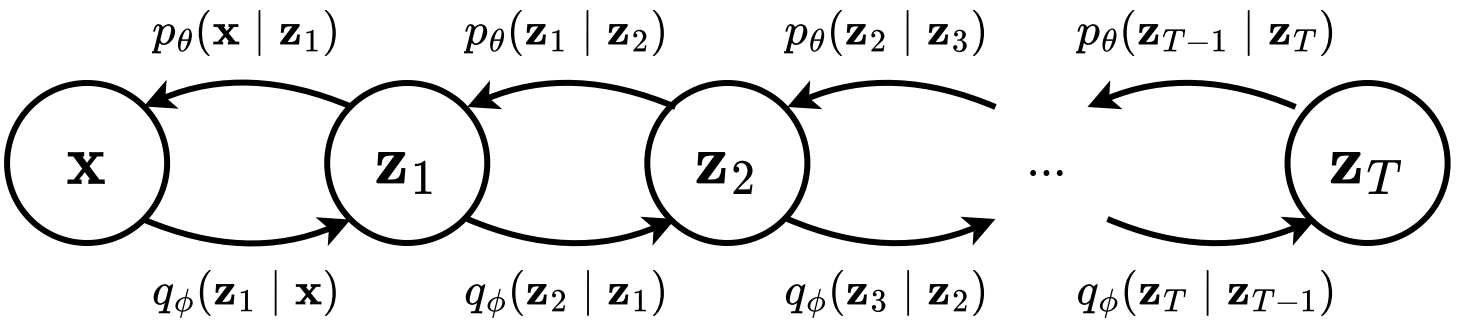
Diffusion models



—

Credits: Kreis et al, 2022.

Markovian Hierarchical VAEs



Similarly to VAEs, training is done by maximizing the ELBO, using a variational distribution $q_\phi(\mathbf{z}_{1:T}|\mathbf{x})$ over all levels of latent variables:

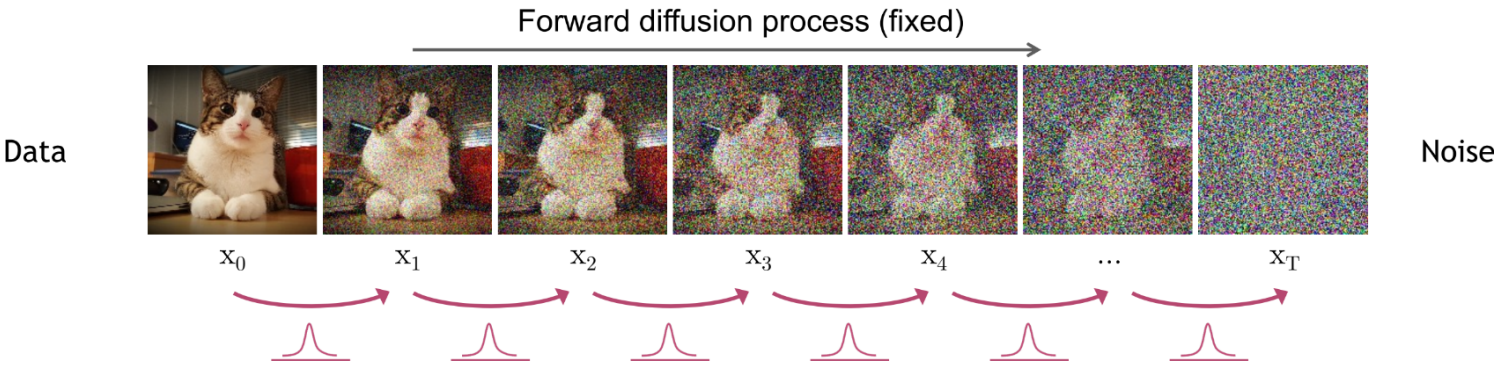
$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T}|\mathbf{x})}\right]$$

Diffusion models

Diffusion models are Markovian HVAEs with the following constraints:

- The latent dimension is the same as the data dimension.
- The encoder is fixed to linear Gaussian transitions $q(\mathbf{x}_t|\mathbf{x}_{t-1})$.
- The hyper-parameters are set such that $q(\mathbf{x}_T|\mathbf{x}_0)$ is a standard Gaussian.

Forward diffusion process

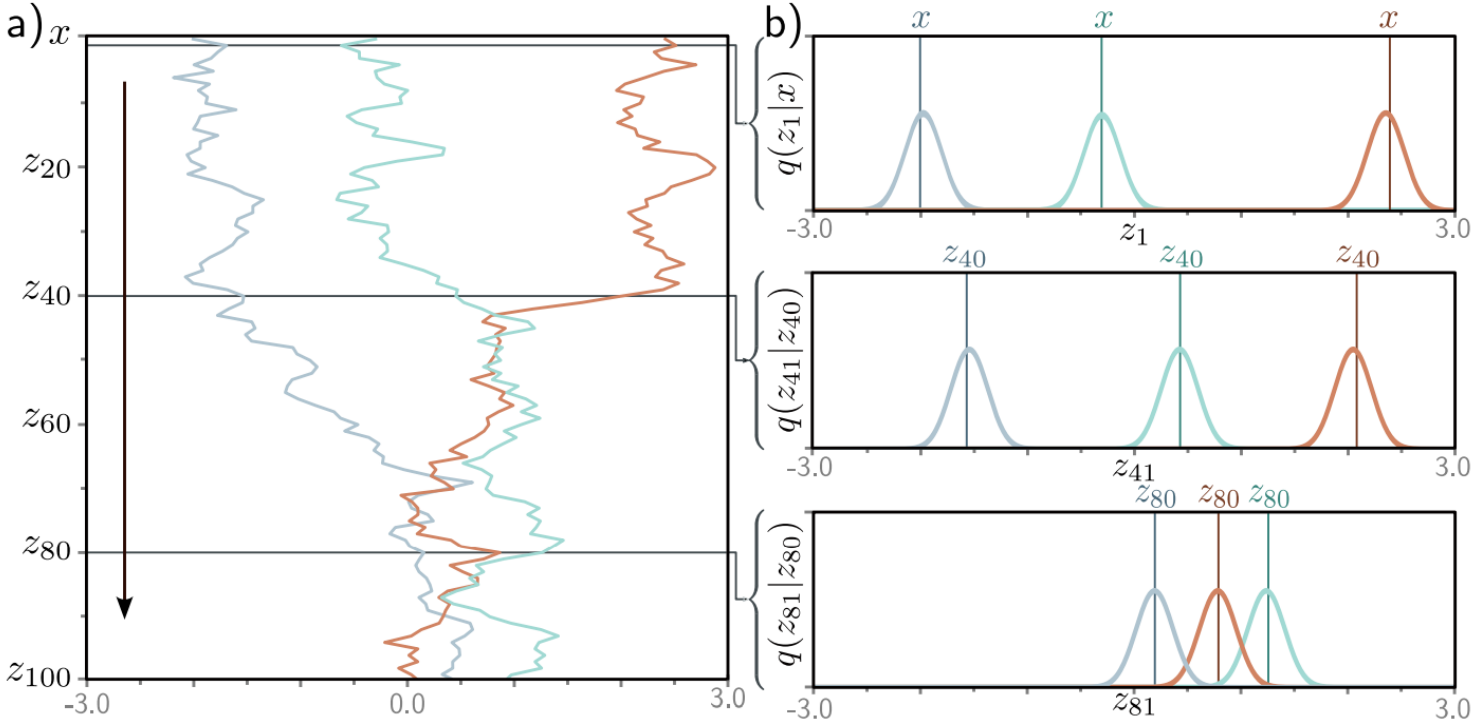


With $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon$$
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

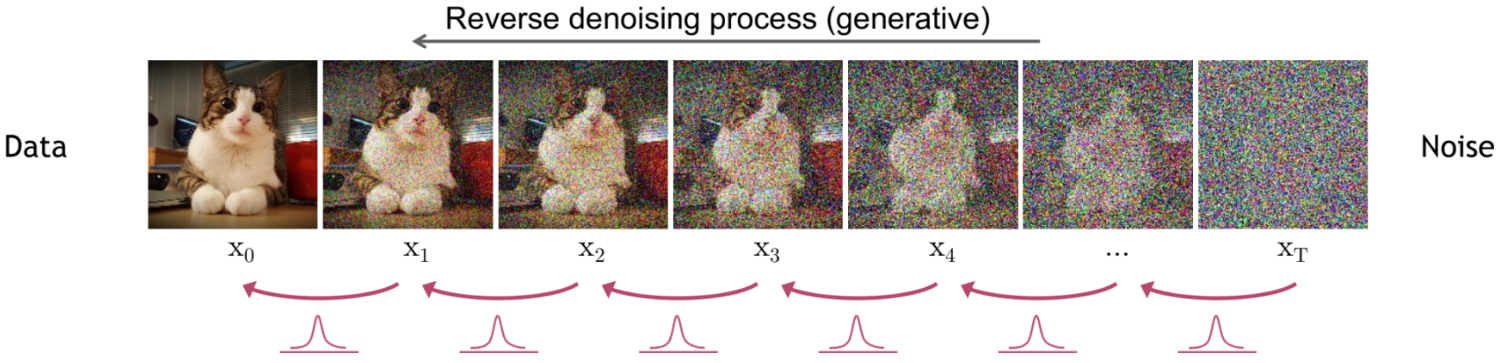
—

Credits: Kreis et al, 2022.


a)
x
z_{20}
z_{40}
z_{60}
z_{80}
z_{100}
-3.0
0.0
3.0
b)
x
x
x
$q(z_1|x)$
-3.0
z_1
3.0
z_{40}
z_{40}
z_{40}
$q(z_{41}|z_{40})$
-3.0
z_{41}
3.0
z_{80}
z_{80}
z_{80}
$q(z_{81}|z_{80})$
-3.0
z_{81}
3.0

—

Reverse denoising process



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

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, I)$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta^2(\mathbf{x}_t, t)\mathbf{I})$$

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_\theta(\mathbf{x}_t, t)\mathbf{z}$$

with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

—

Credits: Kreis et al, 2022.

Training

For learning the parameters θ of the reverse process, we can form a variational lower bound on the log-likelihood of the data as

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

This objective can be rewritten as

$$
\begin{aligned}
L &= \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_0)}\left[L_0 - \sum_{t>1} L_{t-1} - L_T\right]
\end{aligned}
$$

where

- $L_0 = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]$ can be interpreted as a reconstruction term. It can be approximated and optimized using a Monte Carlo estimate.
- $L_{t-1} = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$ is a denoising matching term. The transition $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ provides a learning signal for the reverse process, since it defines how to denoise the noisified input $\mathbf{x}_t$ with access to the original input $\mathbf{x}_0$.
- $L_T = \mathrm{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T))$ represents how close the distribution of the final noisified input is to the standard Gaussian. It has no trainable parameters.

(Some calculations later...)

$$\arg\min_\theta L_{t-1}$$

$$= \arg\min_\theta \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \frac{1}{2\sigma_t^2} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2} ||\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t) - \mathbf{x}_0||_2^2$$

Interpretation 1: Denoising. Training a diffusion model amounts to learning a neural network that predicts the original ground truth $\mathbf{x}_0$ from a noisy input $\mathbf{x}_t$.

$$
\begin{aligned}
&\arg\min_\theta L_{t-1} \\
&= \arg\min_\theta \mathbb{E}_{\mathcal{N}(\epsilon;\mathbf{0},I)} \frac{1}{2\sigma_t^2} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} ||\epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon}_{\mathbf{x}_t}, t) - \epsilon||_2^2 \\
&\approx \arg\min_\theta \mathbb{E}_{\mathcal{N}(\epsilon;\mathbf{0},I)} ||\epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon}_{\mathbf{x}_t}, t) - \epsilon||_2^2
\end{aligned}
$$

Interpration 2: Noise prediction. Training a diffusion model amounts to learning a neural network that predicts the noise ϵ that was added to the original ground truth $\mathbf{x}_0$ to obtain the noisy $\mathbf{x}_t$.

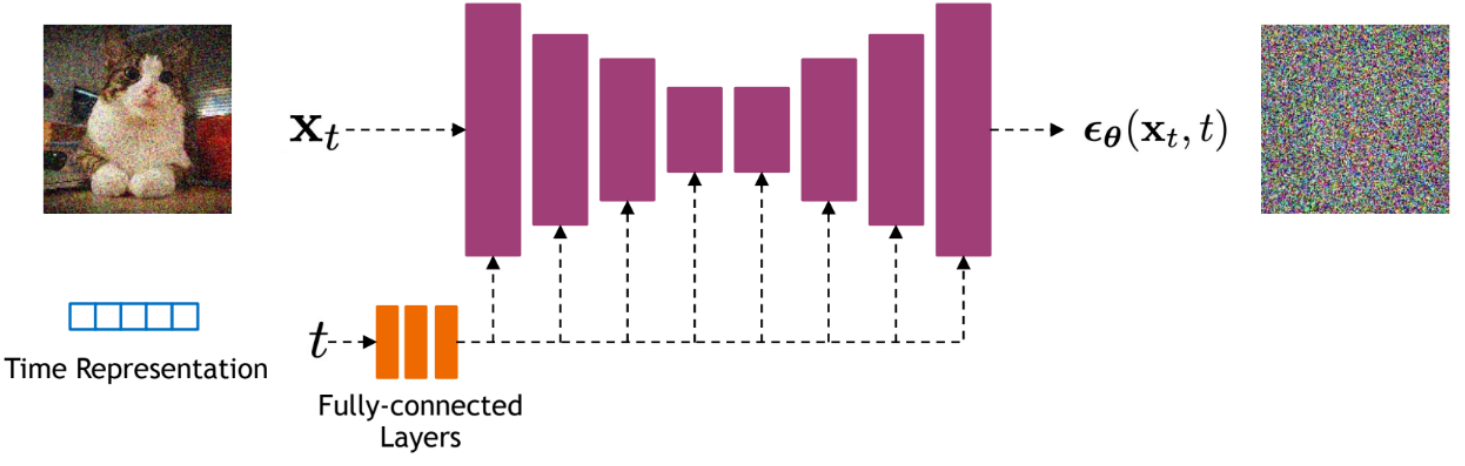
$$\arg\min_\theta L_{t-1}$$

$$= \arg\min_\theta \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \frac{1}{2\sigma_t^2} \frac{(1-\alpha_t)^2}{\alpha_t} ||s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0)||_2^2$$

Interpretation 3: Denoising score matching. Training a diffusion model amounts to learning a neural network that predicts the score $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0)$ of the tractable posterior.

Network architectures

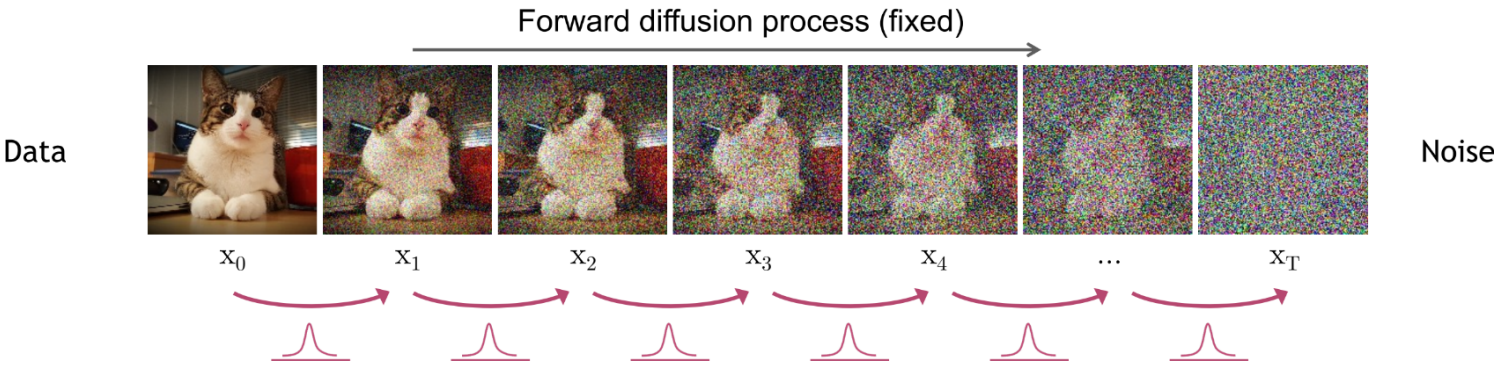
Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t)$, $\epsilon_\theta(\mathbf{x}_t, t)$ or $s_\theta(\mathbf{x}_t, t)$.



—

Credits: Kreis et al, 2022.

Continuous-time diffusion models



With $\beta_t = 1 - \alpha_t$, we can rewrite the forward process as

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{1-\beta(t)\Delta_t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

—

Credits: Kreis et al, 2022.

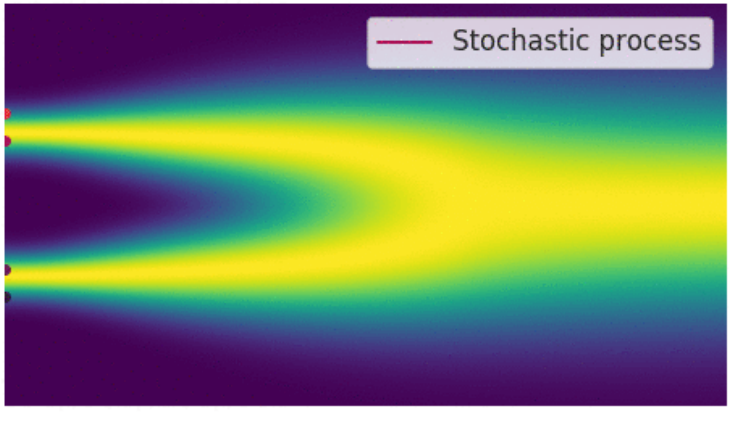
When $\Delta_t \to 0$, we can further rewrite the forward process as

$$\begin{aligned}\mathbf{x}_t &= \sqrt{1-\beta(t)\Delta_t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta_t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta_t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}.$$

This last update rule corresponds to the Euler-Maruyama discretization of the stochastic differential equation (SDE)

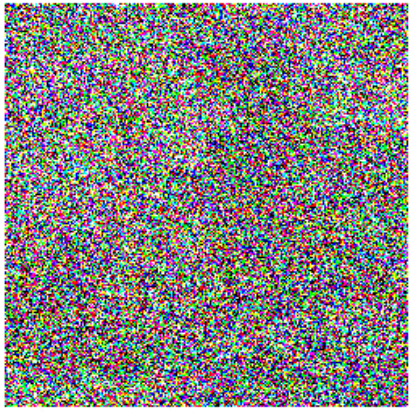
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\mathbf{w}_t$$

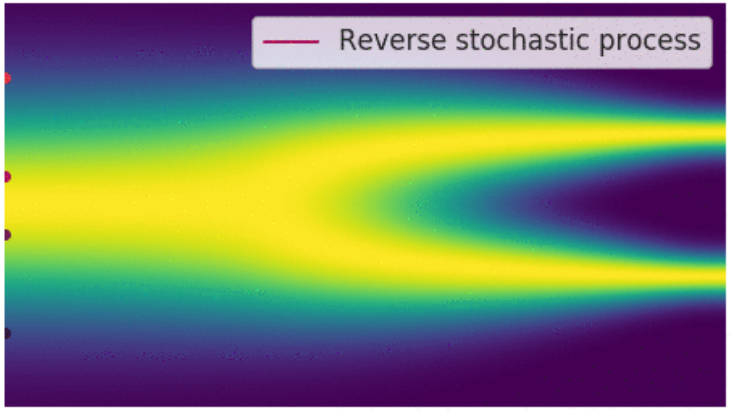
describing the diffusion in the infinitesimal limit.



The reverse process satisfies a reverse-time SDE that can be derived analytically from the forward-time SDE and the score of the marginal distribution $q(\mathbf{x}_t)$, as

$$\mathrm{d}\mathbf{x}_t = \left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) \right] \mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\mathbf{w}_t.$$



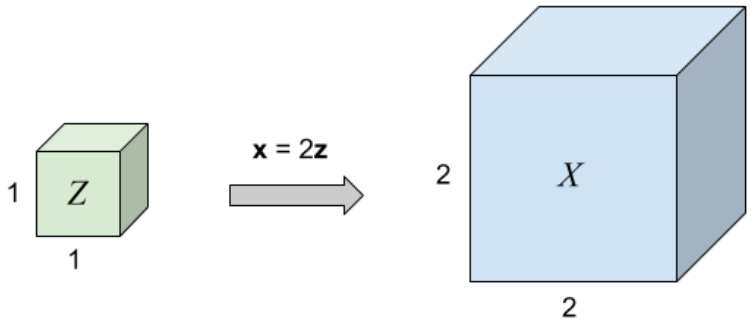


—

Credits: Song, 2021.

Normalizing flows

Change of variables



Assume $p(\mathbf{z})$ is a uniformly distributed unit cube in $\mathbb{R}^3$ and $\mathbf{x} = f(\mathbf{z}) = 2\mathbf{z}$. Since the total probability mass must be conserved,

$$p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z})\frac{V_{\mathbf{z}}}{V_{\mathbf{x}}} = p(\mathbf{z})\frac{1}{8},$$

where $\frac{1}{8} = \left|\det \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}\right|^{-1}$ represents the inverse determinant of the linear transformation f.

What if f is non-linear?

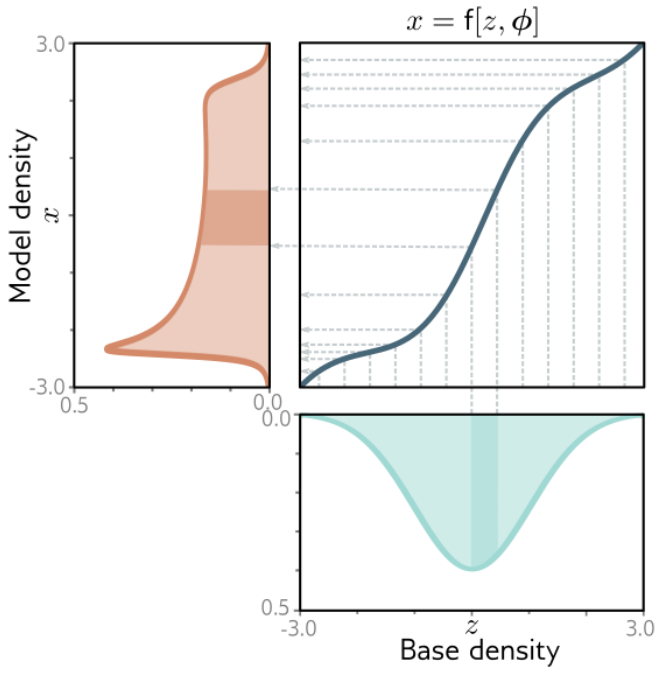


Figure 16.2 Transforming distributions. The base density (cyan, bottom) passes through a function (blue curve, top right) to create the model density (orange, left). Consider dividing the base density into equal intervals (gray vertical lines). The probability mass between adjacent lines must remain the same after transformation. The cyan-shaded region passes through a part of the function where the gradient is larger than one, so this region is stretched. Consequently, the height of the orange-shaded region must be lower so that it retains the same area as the cyan-shaded region. In other places (e.g., $z = -2$), the gradient is less than one, and the model density increases relative to the base density.

—

Image credits: Simon J.D. Prince, Understanding Deep Learning, 2023.

Change of variables theorem

If f is non-linear,

- the Jacobian $J_f(\mathbf{z})$ of $\mathbf{x} = f(\mathbf{z})$ represents the infinitesimal linear transformation in the neighborhood of $\mathbf{z}$;
- if the function is a bijective map, then the mass must be conserved locally.

Therefore, the local change of density yields

$$p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z}) \left|\det J_f(\mathbf{z})\right|^{-1} .$$

Similarly, for $g = f^{-1}$, we have

$$p(\mathbf{x}) = p(\mathbf{z} = g(\mathbf{x})) \left|\det J_g(\mathbf{x})\right| .$$

Example: coupling layers

Assume $\mathbf{z} = (\mathbf{z}_a, \mathbf{z}_b)$ and $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$. Then,

- Forward mapping $\mathbf{x} = f(\mathbf{z})$:

$$\mathbf{x}_a = \mathbf{z}_a, \quad \mathbf{x}_b = \mathbf{z}_b \odot \exp(s(\mathbf{z}_a)) + t(\mathbf{z}_a),$$

- Inverse mapping $\mathbf{z} = g(\mathbf{x})$:

$$\mathbf{z}_a = \mathbf{x}_a, \quad \mathbf{z}_b = (\mathbf{x}_b - t(\mathbf{x}_a)) \odot \exp(-s(\mathbf{x}_a)),$$

where s and t are arbitrary neural networks.

For $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, the log-likelihood is

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) \left| \det J_f(\mathbf{z}) \right|^{-1}$$

where the Jacobian $J_f(\mathbf{z}) = \frac{\partial \mathbf{x}}{\partial \mathbf{z}}$ is a lower triangular matrix

$$\begin{pmatrix} \mathbf{I} & 0 \\ \frac{\partial \mathbf{x}_b}{\partial \mathbf{z}_a} & \operatorname{diag}(\exp(s(\mathbf{z}_a))) \end{pmatrix},$$

such that $\left| \det J_f(\mathbf{z}) \right| = \prod_i \exp(s(\mathbf{z}_a))_i = \exp(\sum_i s(\mathbf{z}_a)_i)$.

Therefore, the log-likelihood is

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) - \sum_i s(\mathbf{z}_a)_i$$

Normalizing flows

A normalizing flow is a change of variable f that transforms a base distribution $p(\mathbf{z})$ into $p(\mathbf{x})$ through a discrete sequence of invertible transformations.

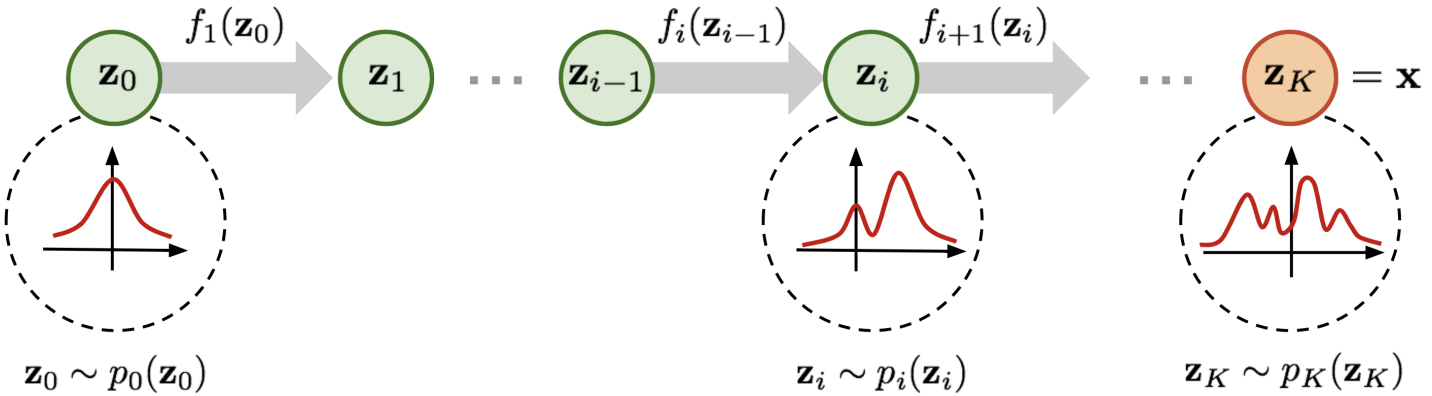


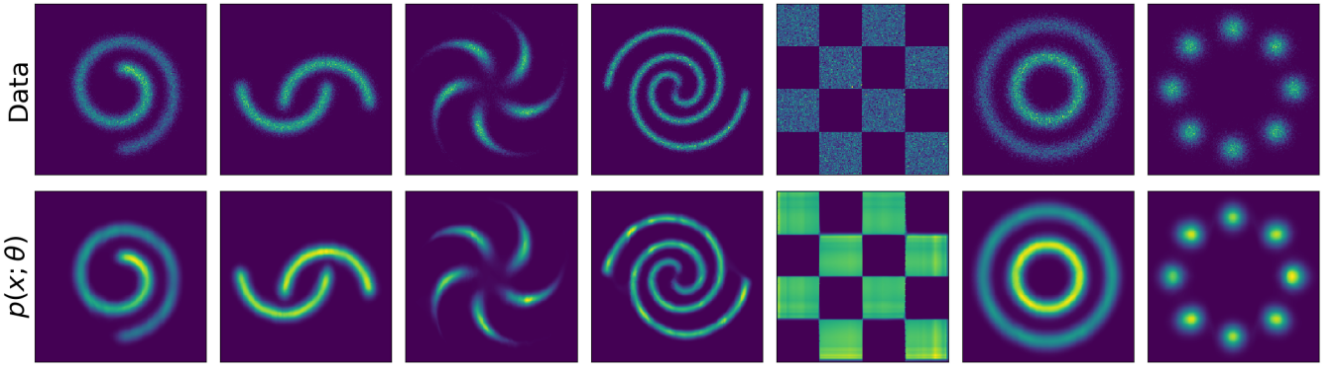
Image credits: Lilian Weng, 2018.

Formally,

$$\begin{aligned}
&\mathbf{z}_0 \sim p(\mathbf{z}) \\
&\mathbf{z}_k = f_k(\mathbf{z}_{k-1}), \quad k = 1, ..., K \\
&\mathbf{x} = \mathbf{z}_K = f_K \circ ... \circ f_1(\mathbf{z}_0).
\end{aligned}$$

The change of variable theorem yields

$$\log p(\mathbf{x}) = \log p(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(\mathbf{z}_{k-1}) \right| .$$



Normalizing flows can fit complex multimodal discontinuous densities.

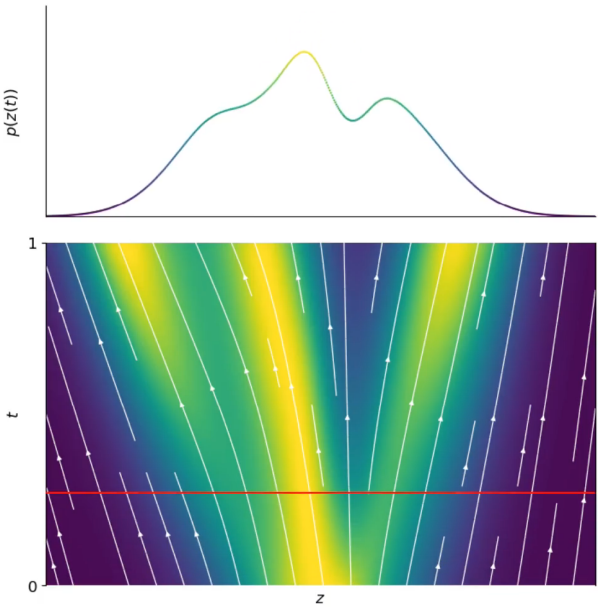
—

Image credits: Wehenkel and Louppe, 2019.

Continuous-time normalizing flows

Replace the discrete sequence of transformations with a neural ODE with reversible dynamics such that

$$\mathbf{z}_0 \sim p(\mathbf{z})$$
$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \theta)$$
$$\mathbf{x} = \mathbf{z}(1) = \mathbf{z}_0 + \int_0^1 f(\mathbf{z}(t), t) dt.$$



The instantaneous change of variable yields

$$\log p(\mathbf{x}) = \log p(\mathbf{z}(0)) - \int_0^1 \mathrm{Tr}\left(\frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}(t)}\right) dt.$$

—

Image credits: Grathwohl et al, 2018.

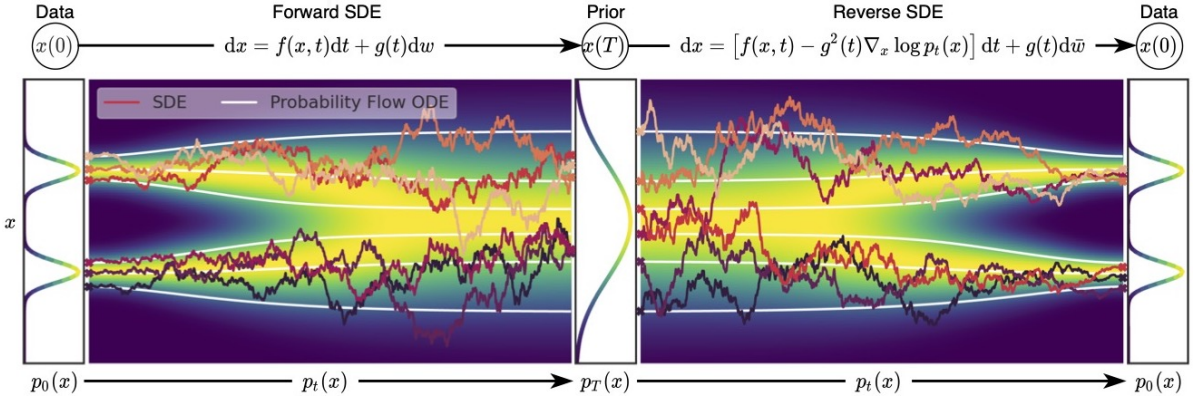
Probability flow ODE

Back to diffusion: For any diffusion process, there exists a corresponding deterministic process

$$\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(t, \mathbf{x}_t) - \frac{1}{2} g^2(t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)\right] \mathrm{d}t$$

whose trajectories share the same marginal densities $p(\mathbf{x}_t)$.

Therefore, when $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ is replaced by its approximation $s_\theta(\mathbf{x}_t, t)$, the probability flow ODE becomes a special case of a neural ODE. In particular, it is an example of continuous-time normalizing flows!



—

Credits: Song, 2021.

The end.