Deep Learning

Lecture 12: Diffusion models

Prof. Gilles Louppe
g.louppe@uliege.be

Today

- VAEs
- Variational diffusion models
- Score-based generative models

Caution: See also the side notes derived in class.

Applications

A few motivating examples.

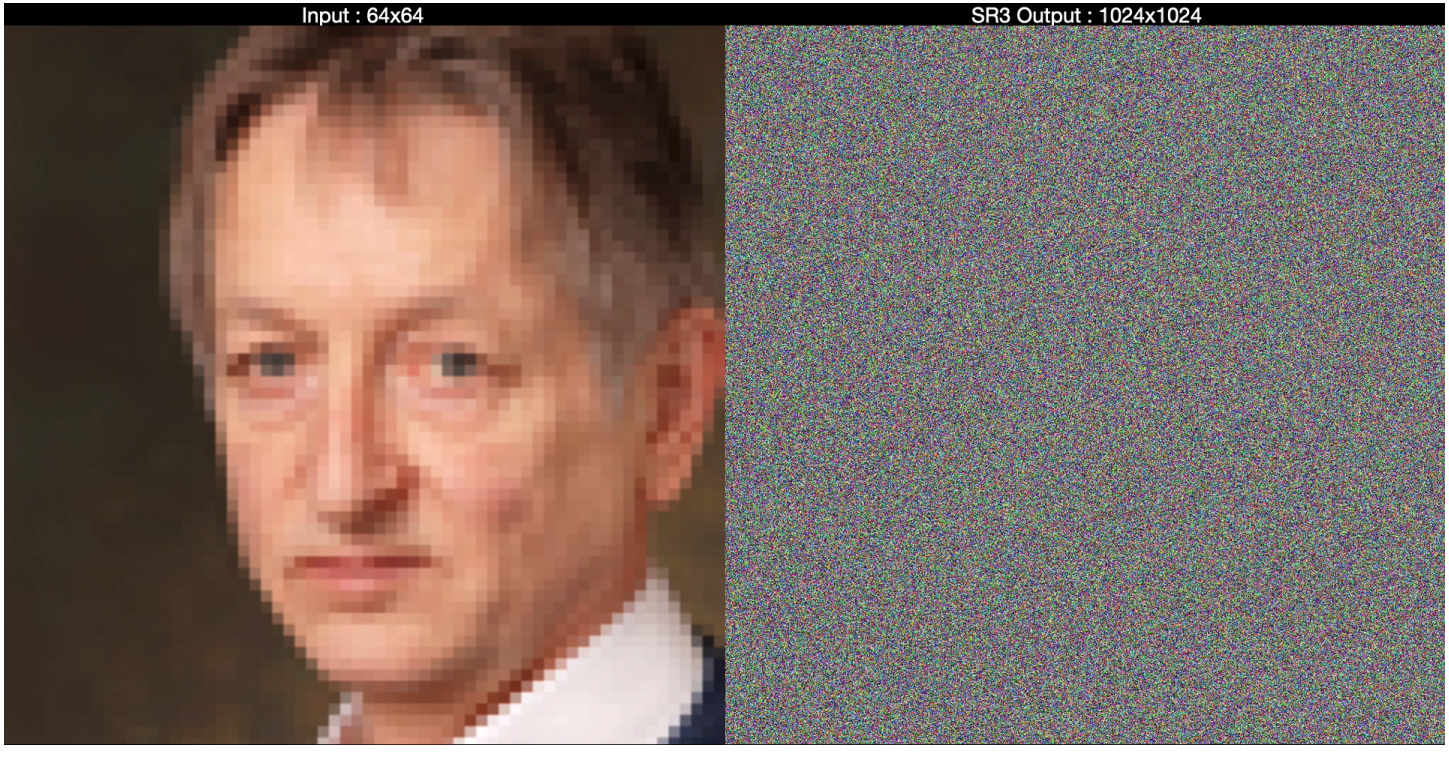
Content generation

Diffusion models have emerged as powerful generative models, beating previous state-of-the-art models (such as GANs) on a variety of tasks.

—

Credits: Dhariwal and Nichol, 2021; Ho et al, 2021.

Image super-resolution



—

Credits: Saharia et al, 2021.

Text-to-image generation

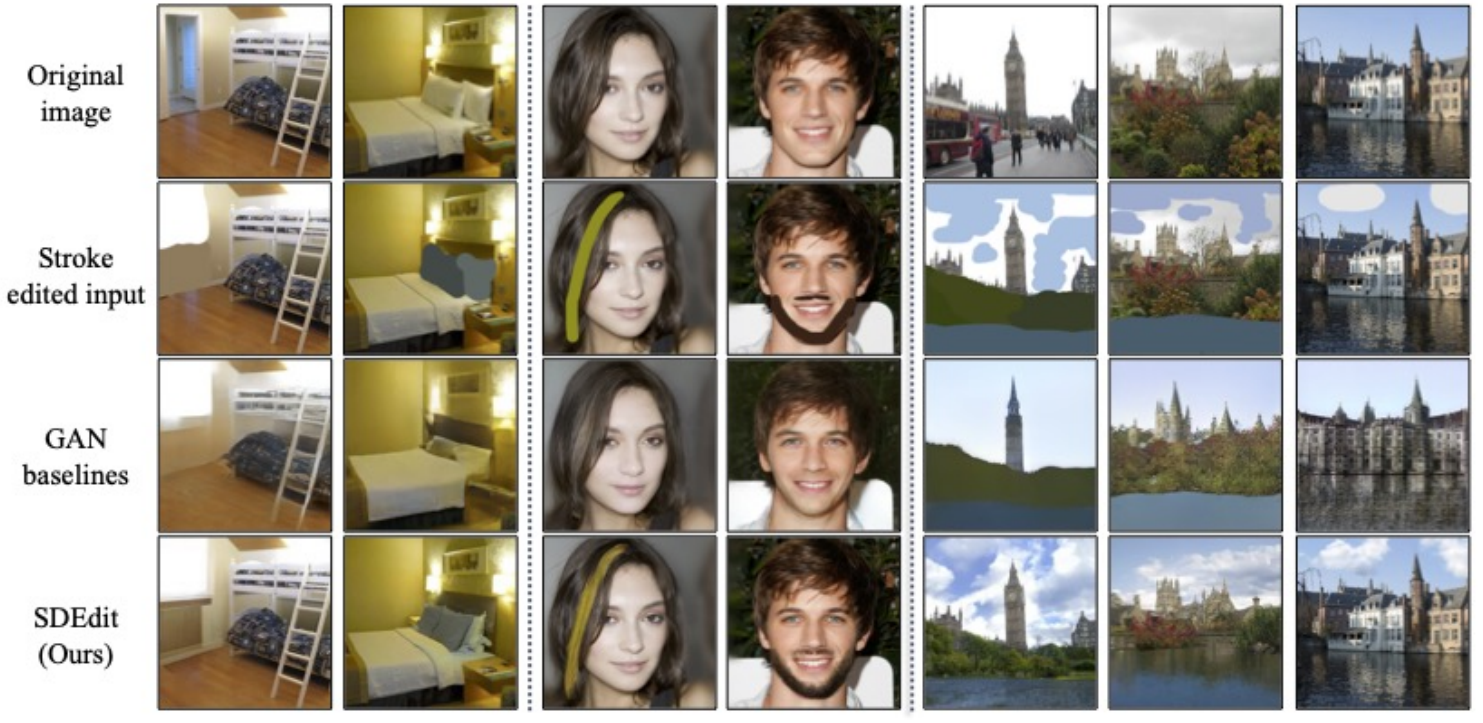
A group of teddy bears in suite in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.

—

Credits: Saharia et al, 2022.

... or deepfakes.

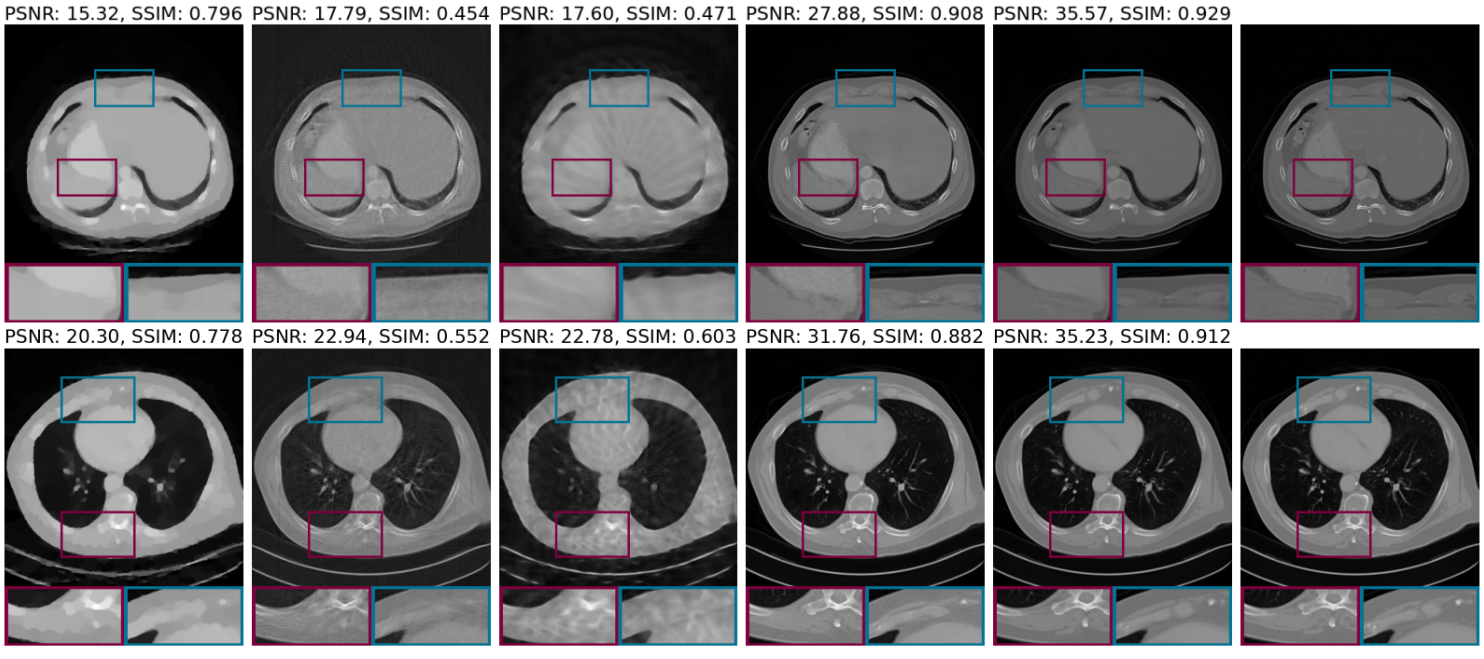
Artistic tools and image editing



—

Credits: Meng et al, 2021.

Inverse problems in medical imaging



—

Credits: Song et al, 2021.

Data assimilation in ocean models

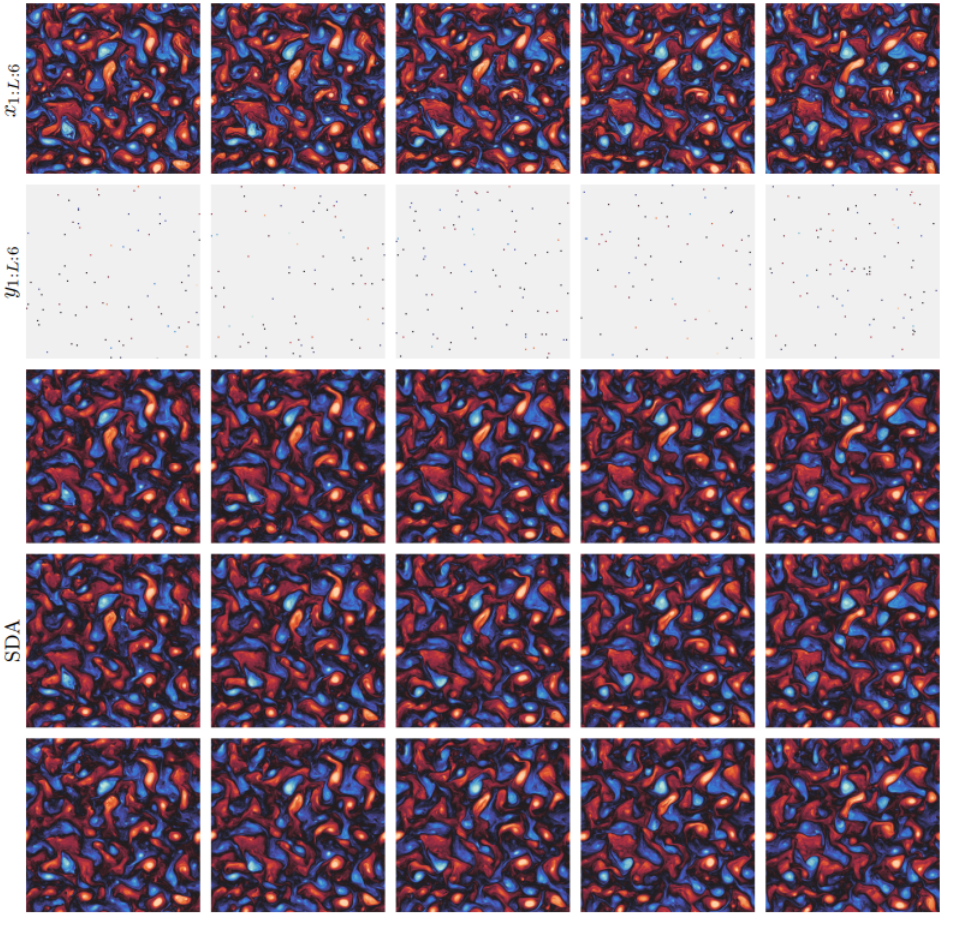


Figure 3: Example of sampled trajectories for a spatially sparse observation. The observation y corresponds to a random (uniform) sampling of ± 80 bins of the velocity fields (u^1, v^1, u^2, v^2) with medium Gaussian noise ($\Sigma_y = 0.1^2 I$). SDA generates trajectories similar to the original one, despite the limited amount of information available in the observation. The three trajectories present slight physically plausible variations, as expected from sampling from a narrow posterior. We observe that the trajectories exhibit less small-scale details than the original one.

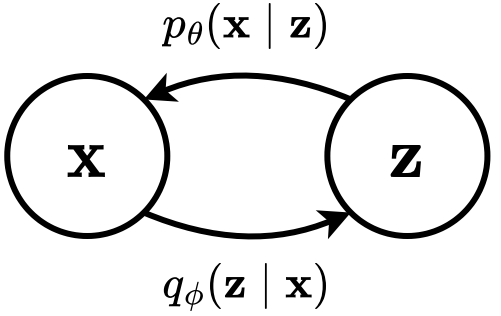
—

Credits: Rozet and Louppe, 2023.

VAEs

A short recap.

Variational autoencoders



Training

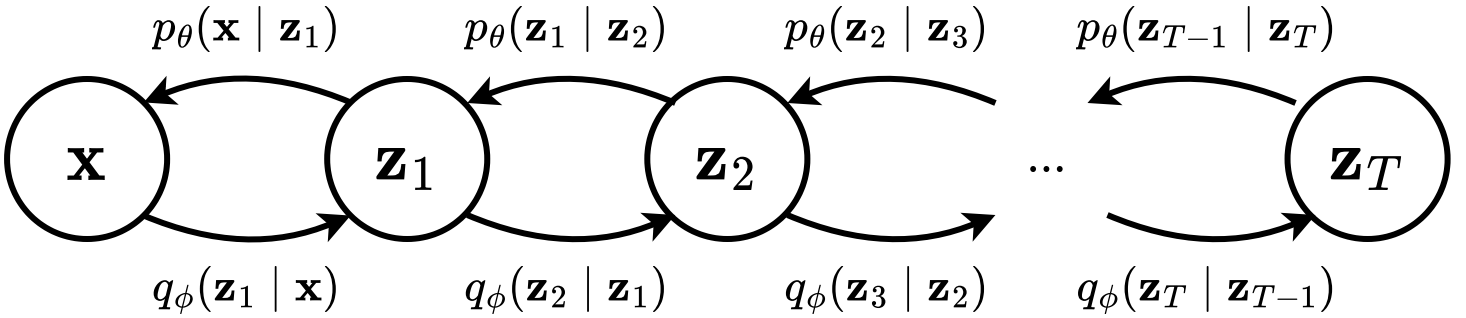
$$
\begin{aligned}
\theta^*, \phi^* &= \arg\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \mathrm{ELBO}(\mathbf{x}; \theta, \phi) \\
&= \arg\max_{\theta,\phi} \mathbb{E}_{p(\mathbf{x})} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}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

Issue: The prior matching term limits the expressivity of the model.

Solution: Make $p(\mathbf{z})$ a learnable distribution.

(Markovian) Hierarchical VAEs

The prior $p(\mathbf{z})$ is itself a VAE, and recursively so for its own hyper-prior.



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:

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

Variational diffusion models

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

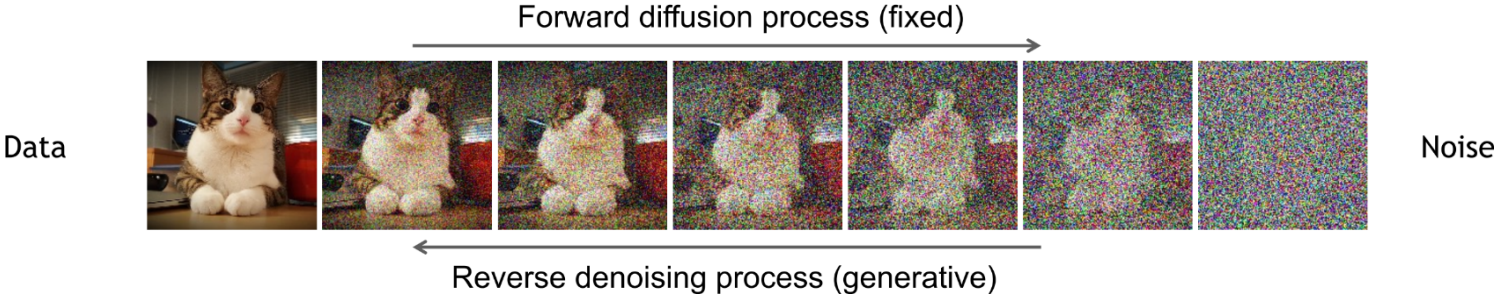
A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference imple-

these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

Variational diffusion models are Markovian HVAEs with the following constraints:

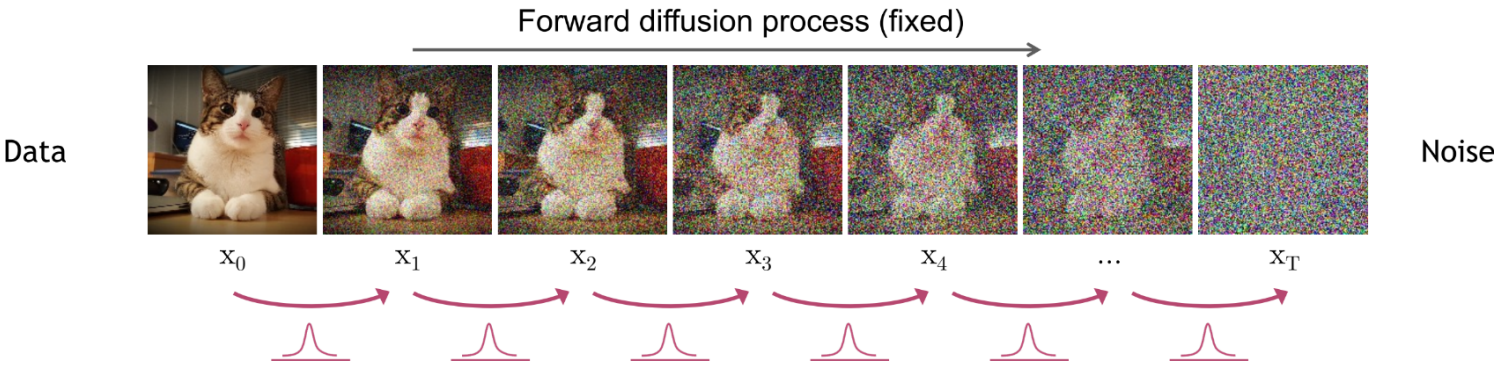
- The latent dimension is the same as the data dimension.
- The encoder is fixed to linear Gaussian transitions $q(\mathbf{x}_t|\mathbf{x}_{t-1})$.
- The hyper-parameters are set such that $q(\mathbf{x}_T|\mathbf{x}_0)$ is a standard Gaussian.



—

Credits: Kreis et al, 2022.

Forward diffusion process

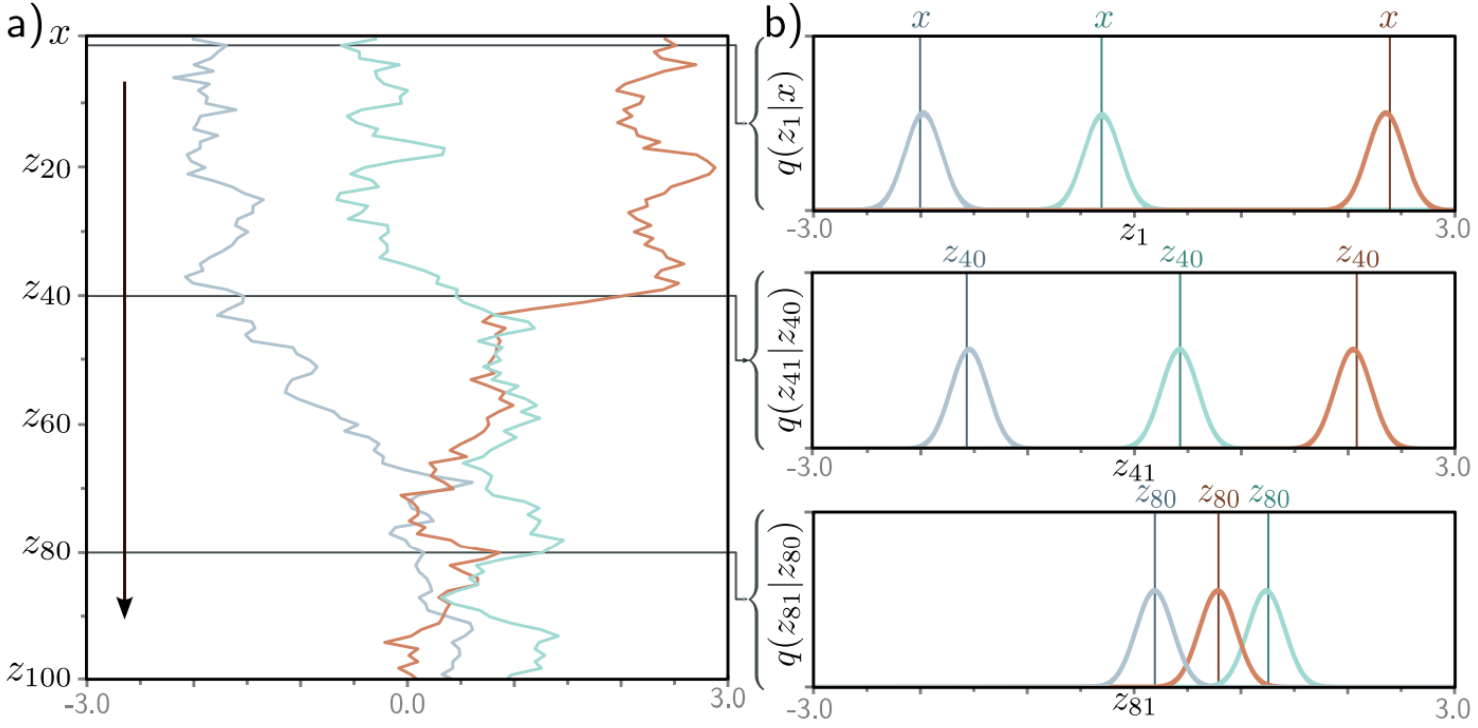


With $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have

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

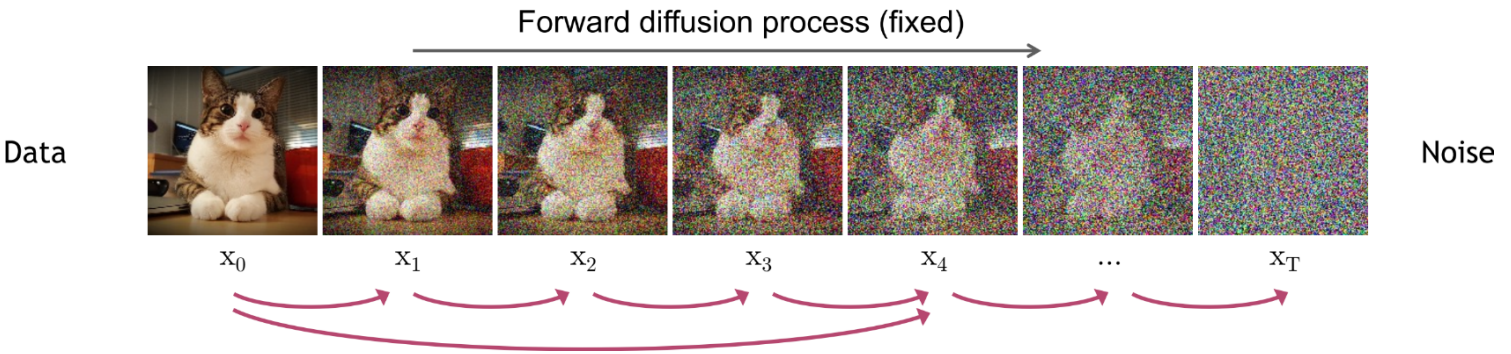
—

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

—

Diffusion kernel

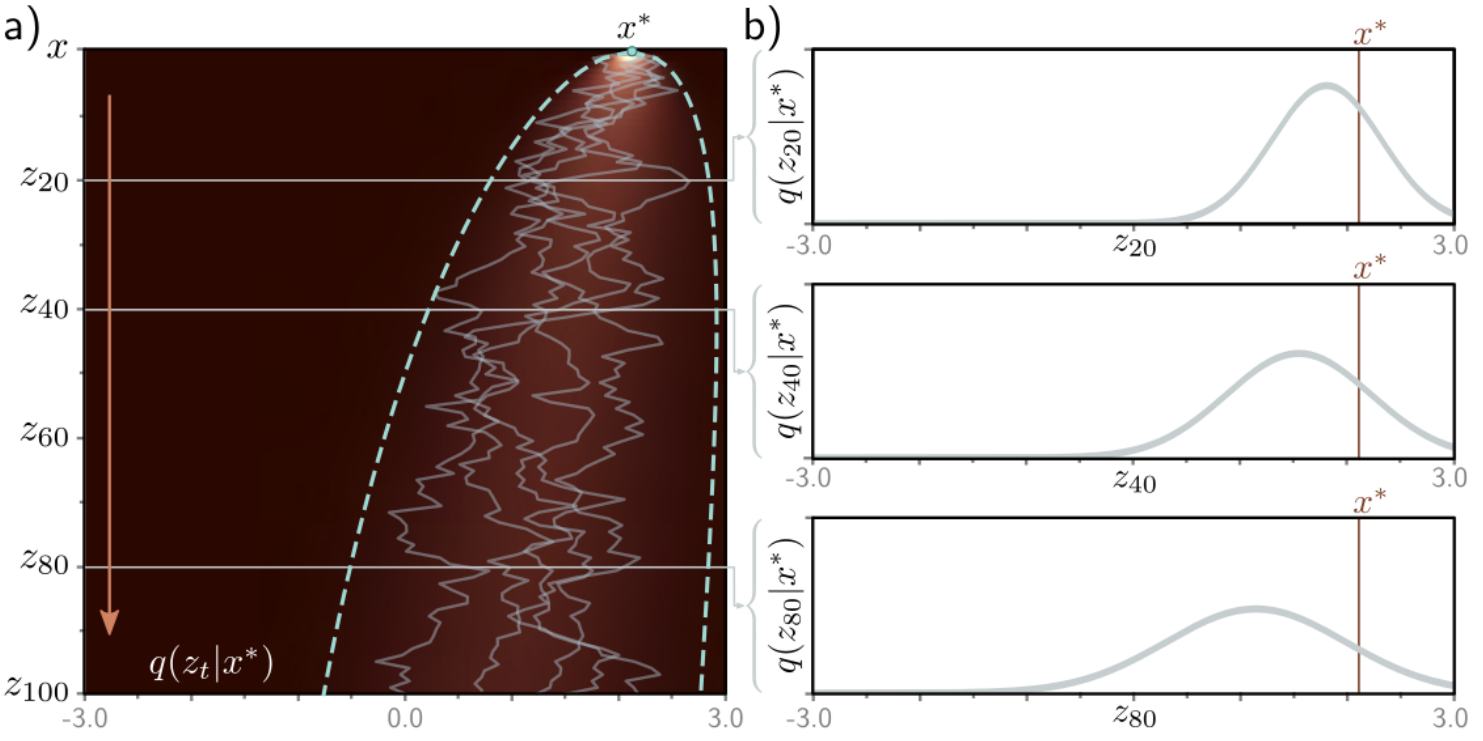


With $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

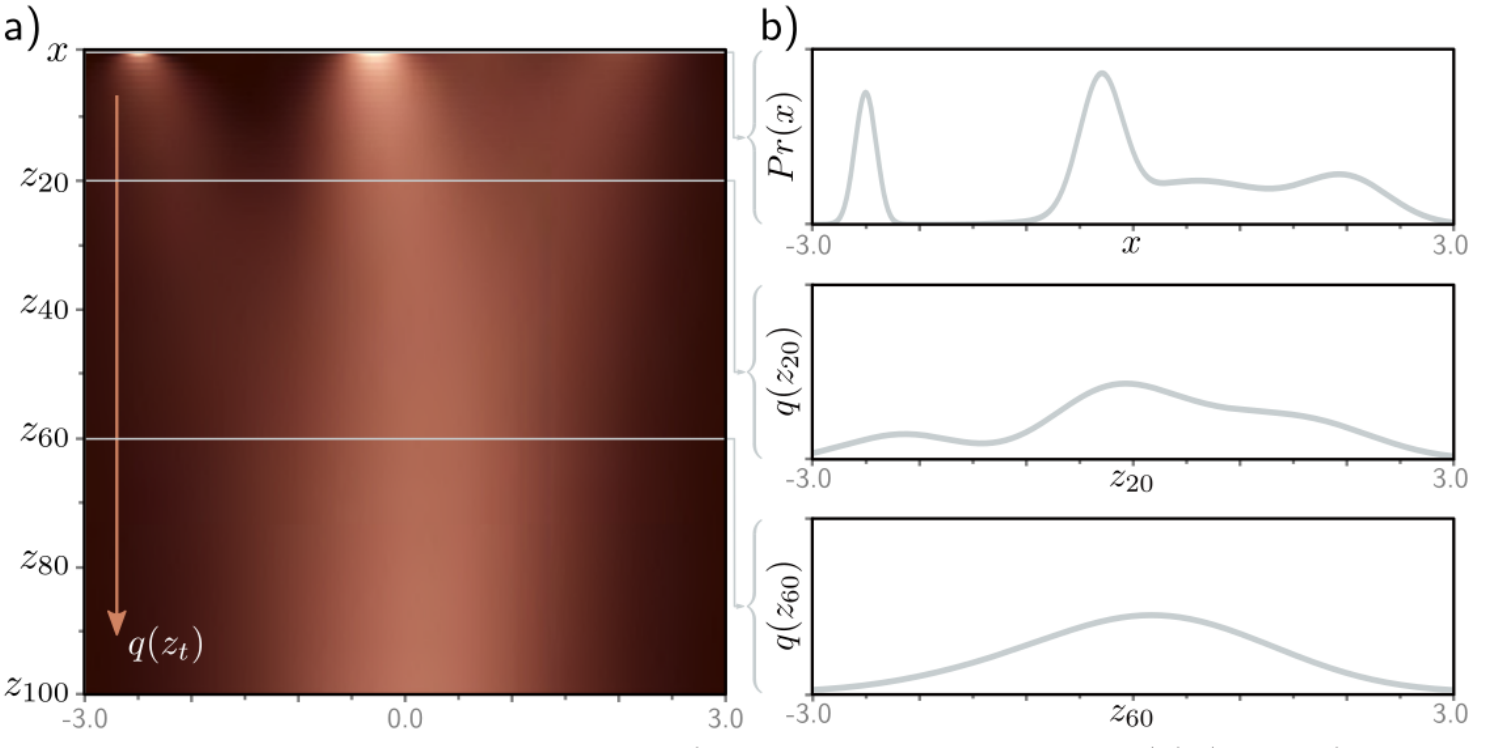
—

Credits: Kreis et al, 2022.



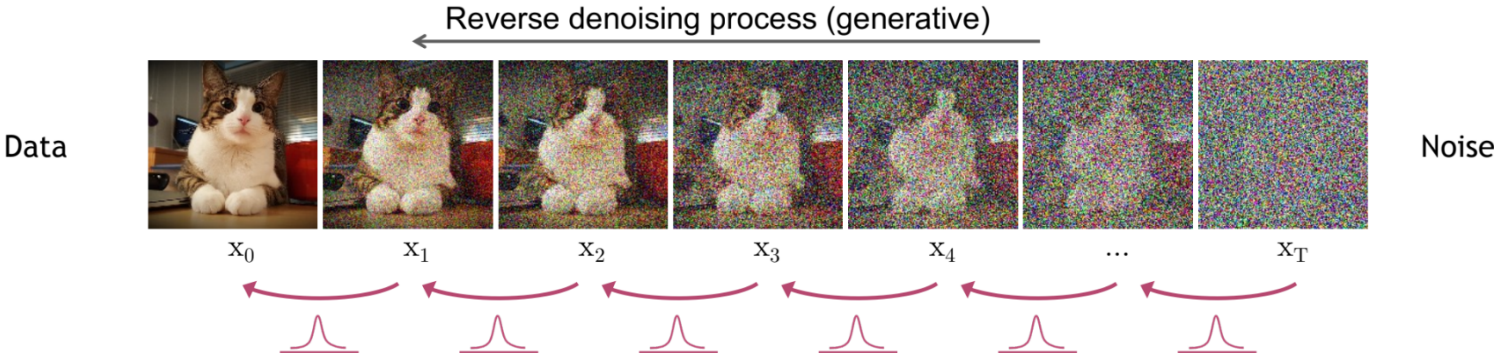
Diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ for different noise levels t.

—



Marginal distribution $q(\mathbf{x}_t)$.

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

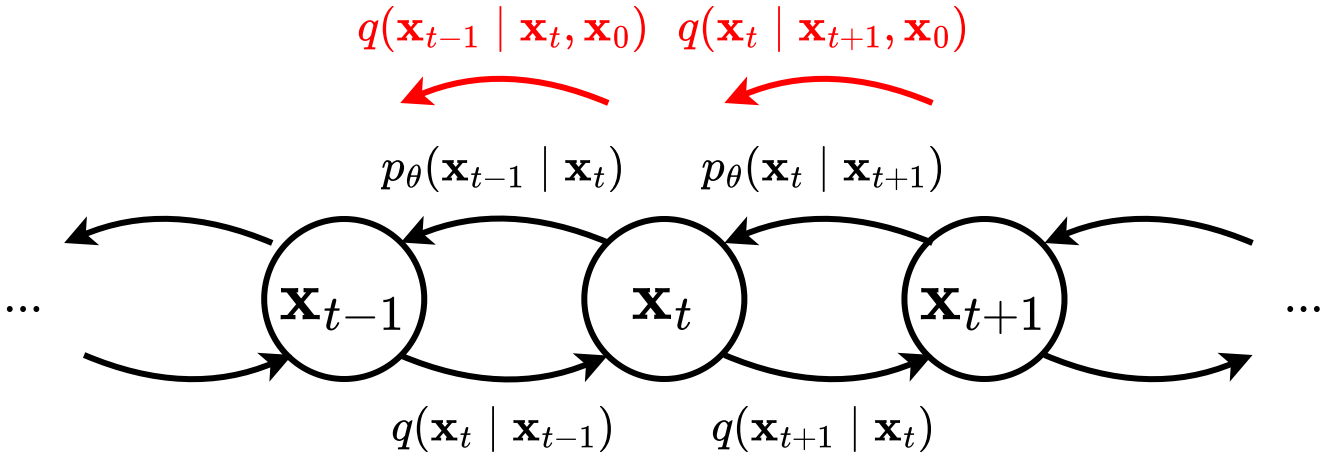
$$\mathbb{E}_{q(\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0)\right] \geq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] := L$$

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.



The distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the tractable posterior distribution

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0, t), \sigma_t^2 I) \end{aligned}$$

where

$$\begin{aligned} \mu_q(\mathbf{x}_t, \mathbf{x}_0, t) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\ \sigma_t^2 &= \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \end{aligned}$$

Interpretation 1: Denoising

To minimize the expected KL divergence L_{t-1}, we need to match the reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to the tractable posterior. Since both are Gaussian, we can match their means and variances.

By construction, the variance of the reverse process can be set to the known variance σ_t^2 of the tractable posterior.

For the mean, we reuse the analytical form of $\mu_q(\mathbf{x}_t, \mathbf{x}_0, t)$ and parameterize the mean of the reverse process using a **denoising network** as

$$\mu_\theta(\mathbf{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t).$$

Under this parameterization, the minimization of expected KL divergence L_{t-1} can be rewritten as

$$
\begin{aligned}
&\arg\min_{\theta} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \\
&= \arg\min_{\theta} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \frac{1}{2\sigma_t^2} ||\mu_\theta(\mathbf{x}_t, t) - \mu_q(\mathbf{x}_t, \mathbf{x}_0, t)||_2^2 \\
&= \arg\min_{\theta} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \frac{1}{2\sigma_t^2} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2} ||\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t) - \mathbf{x}_0||_2^2
\end{aligned}
$$

Optimizing a VDM amounts to learning a neural network that predicts the original ground truth $\mathbf{x}_0$ from a noisy input $\mathbf{x}_t$.

Finally, minimizing the summation of the L_{t-1} terms across all noise levels t can be approximated by minimizing the expectation over all timesteps as

$$\arg\min_{\theta} \mathbb{E}_{t\sim U\{2,T\}} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)).$$

Interpretation 2: Noise prediction

A second interpretation of VDMs can be obtained using the reparameterization trick. Using

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}},$$

we can rewrite the mean of the tractable posterior as

$$\begin{aligned}
\mu_q(\mathbf{x}_t, \mathbf{x}_0, t) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}} \\
&= \dots \\
&= \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)\alpha_t}}\epsilon
\end{aligned}$$

Accordingly, the mean of the reverse process can be parameterized with a **noise-prediction network** as

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)\alpha_t}}\epsilon_\theta(\mathbf{x}_t, t).$$

Under this parameterization, the minimization of the expected KL divergence L_{t-1} can be rewritten as

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

Optimizing a VDM 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 $\mathbf{x}_t$.

In summary, training and sampling thus eventually boils down to:

Algorithm 1 Training

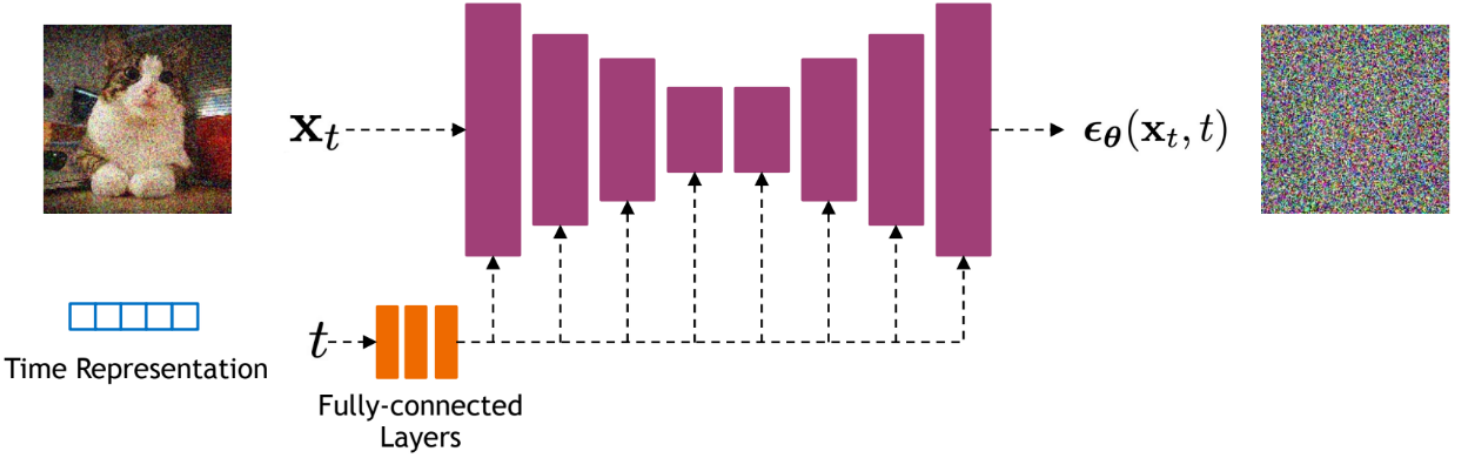
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

Network architectures

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



—

Credits: Kreis et al, 2022.

Score-based generative models

Score-based models

Maximum likelihood estimation for energy-based probabilistic models

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \exp(-f_\theta(\mathbf{x}))$$

can be intractable when the partition function Z_θ is unknown. We can sidestep this issue with a score-based model

$$s_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$$

that approximates the (Stein) **score function** of the data distribution. If we parameterize the score-based model with an energy-based model, then we have

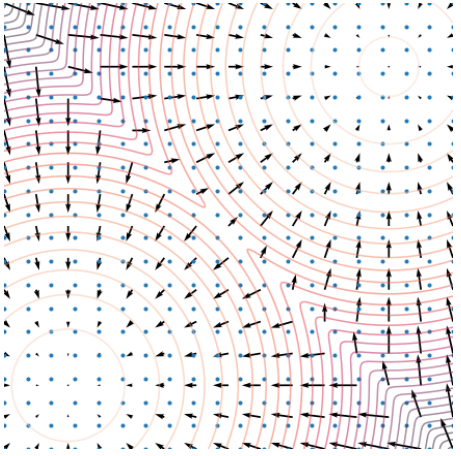
$$s_\theta(\mathbf{x}) = \nabla_\mathbf{x} \log p_\theta(\mathbf{x}) = -\nabla_\mathbf{x} f_\theta(\mathbf{x}) - \nabla_\mathbf{x} \log Z_\theta = -\nabla_\mathbf{x} f_\theta(\mathbf{x}),$$

which discards the intractable partition function and expands the family of models that can be used.

The score function points in the direction of the highest density of the data distribution. It can be used to find modes of the data distribution or to generate samples by **Langevin dynamics** by iterating the following sampling rule

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}_i} \log p(\mathbf{x}_i) + \sqrt{2\epsilon}\mathbf{z}_i,$$

where ϵ is the step size and $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. When ϵ is small, Langevin dynamics will converge to the data distribution $p(\mathbf{x})$.



—

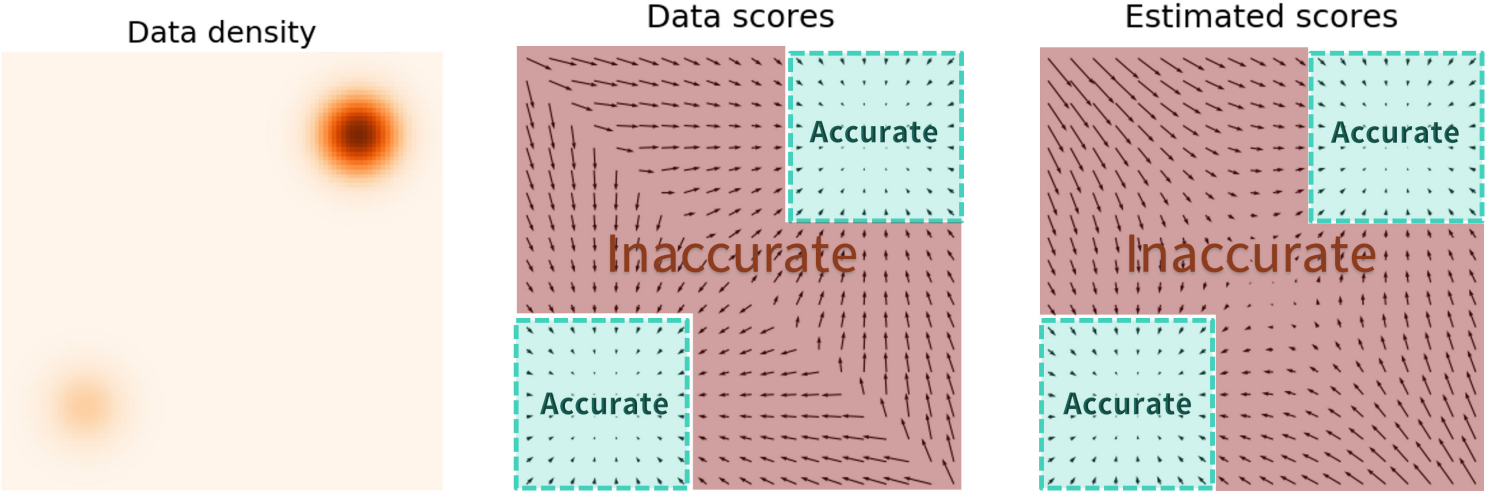
Credits: Song, 2021.

Similarly to likelihood-based models, score-based models can be trained by minimizing the **Fisher divergence** between the data distribution $p(\mathbf{x})$ and the model distribution $p_\theta(\mathbf{x})$ as

$$\mathbb{E}_{p(\mathbf{x})}\left[||\nabla_{\mathbf{x}} \log p(\mathbf{x}) - s_\theta(\mathbf{x})||_2^2\right].$$

Unfortunately, the explicit score matching objective leads to inaccurate estimates in low-density regions, where few data points are available to constrain the score.

Since initial sample points are likely to be in low-density regions in high-dimensional spaces, the inaccurate score-based model will derail the Langevin dynamics and lead to poor sample quality.



—

Credits: Song, 2021.

To address this issue, **denoising score matching** can be used to train the score-based model to predict the score of increasingly noisified data points.

For each noise level t, the score-based model $s_\theta(\mathbf{x}_t, t)$ is trained to predict the score of the noisified data point $\mathbf{x}_t$ as

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

where $p_t(\mathbf{x}_t)$ is the noise-perturbed data distribution

$$p_t(\mathbf{x}_t) = \int p(\mathbf{x}_0) \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2 \mathbf{I}) d\mathbf{x}_0$$

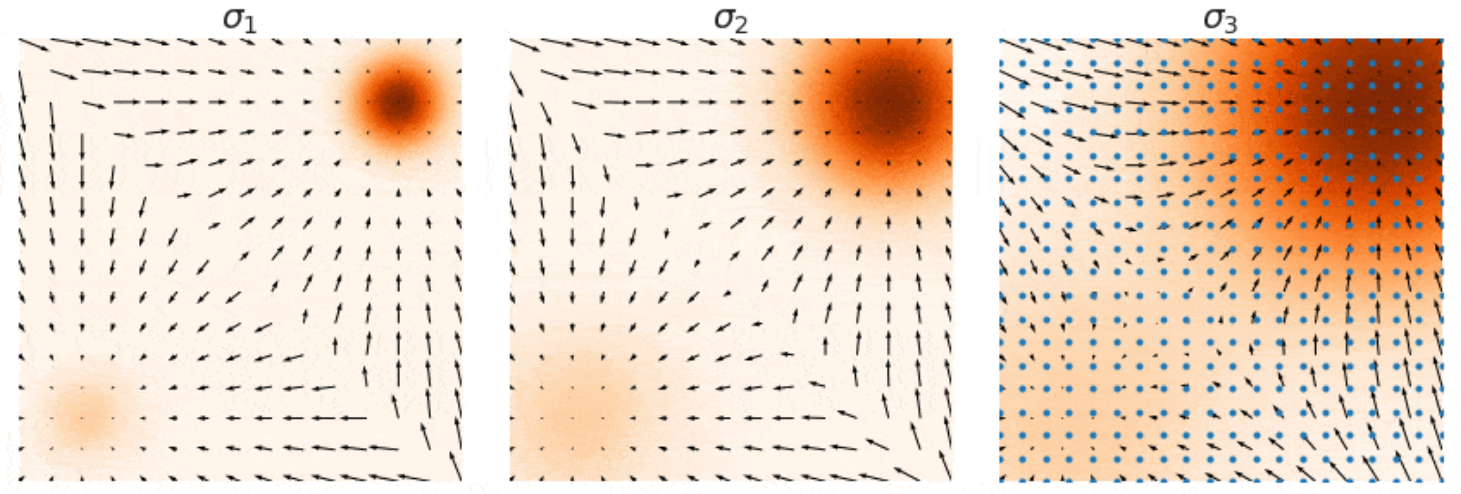
and σ_t^2 is an increasing sequence of noise levels.

The training objective for $s_\theta(\mathbf{x}_t, t)$ is then a weighted sum of Fisher divergences for all noise levels t,

$$\sum_{t=1}^{T} \lambda(t) \mathbb{E}_{p_t(\mathbf{x}_t)} \left[||\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) - s_\theta(\mathbf{x}_t, t)||_2^2 \right]$$

where $\lambda(t)$ is a weighting function that increases with t to give more importance to the noisier samples.

Finally, annealed Langevin dynamics can be used to sample from the score-based model by running Langevin dynamics with decreasing noise levels $t = T, ..., 1$.



—

Credits: Song, 2021.

Interpretation 3: Denoising score matching

A third interpretation of VDMs can be obtained by reparameterizing $\mathbf{x}_0$ using Tweedie's formula, as

$$\mathbf{x}_0 = \frac{\mathbf{x}_t + (1-\bar{\alpha}_t)\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0)}{\sqrt{\bar{\alpha}_t}},$$

which we can plug into the the mean of the tractable posterior to obtain

$$\begin{aligned}\mu_q(\mathbf{x}_t, \mathbf{x}_0, t) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\ &= \dots \\ &= \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0).\end{aligned}$$

The mean of the reverse process can be parameterized with a **score network** as

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha}_t}\mathbf{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} s_\theta(\mathbf{x}_t, t).$$

Under this parameterization, the minimization of the expected KL divergence L_{t-1} can be rewritten as

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

Optimizing a score-based 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.

Since $s_\theta(\mathbf{x}_t, t)$ is learned in expectation over the data distribution $q(\mathbf{x}_0)$, the score network will eventually approximate the score of the marginal distribution $q(\mathbf{x}_t)$, for each noise level t, that is

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t).$$

Conditional sampling

To turn a diffusion model $p_\theta(\mathbf{x}_{0:T})$ into a conditional model, we can add conditioning information y at each step of the reverse process, as

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

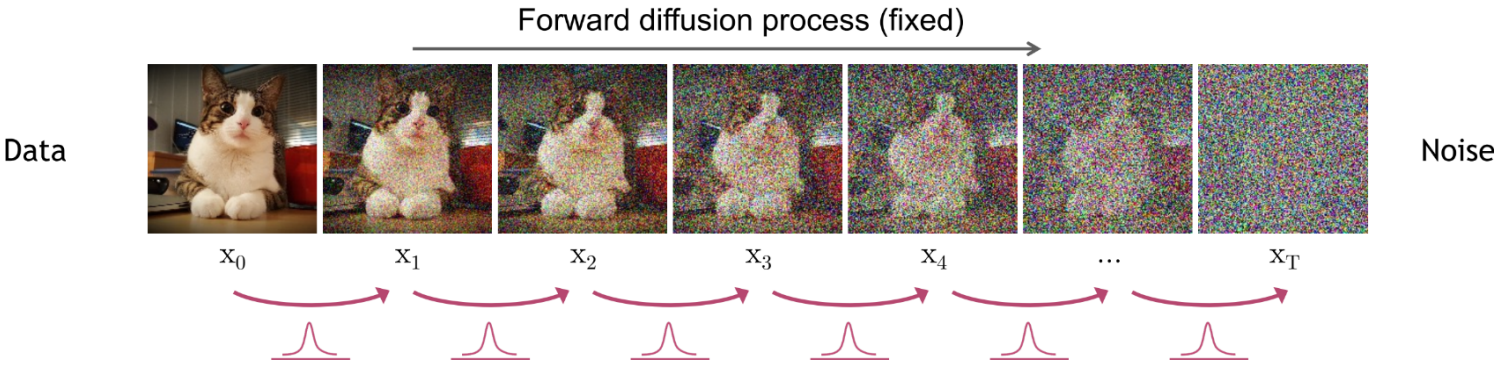
With a score-based model however, we can use the Bayes rule and notice that

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t),$$

where we leverage the fact that the gradient of $\log p(y)$ with respect to $\mathbf{x}_t$ is zero.

In other words, controllable generation can be achieved by adding a conditioning signal during sampling, without having to retrain the model. E.g., train an extra classifier $p(y|\mathbf{x}_t)$ and use it to control the sampling process by adding its gradient to the score.

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.

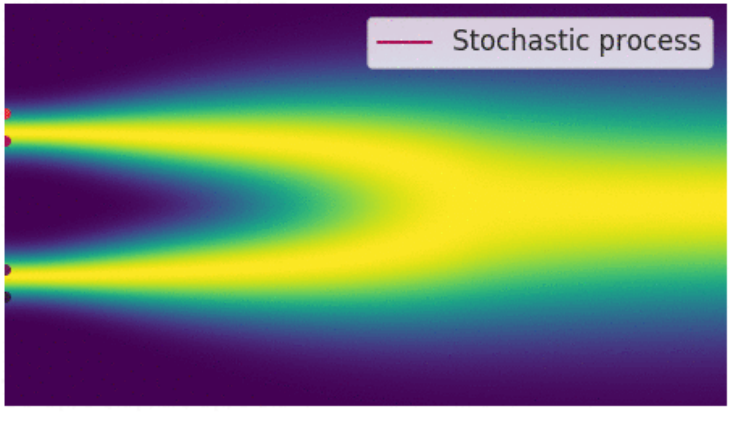
When $\Delta_t \to 0$, we can further rewrite the forward process as

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

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

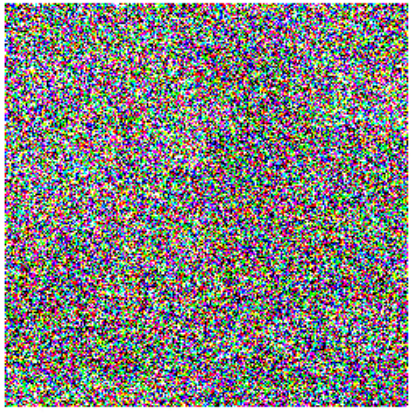
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\mathbf{w}_t$$

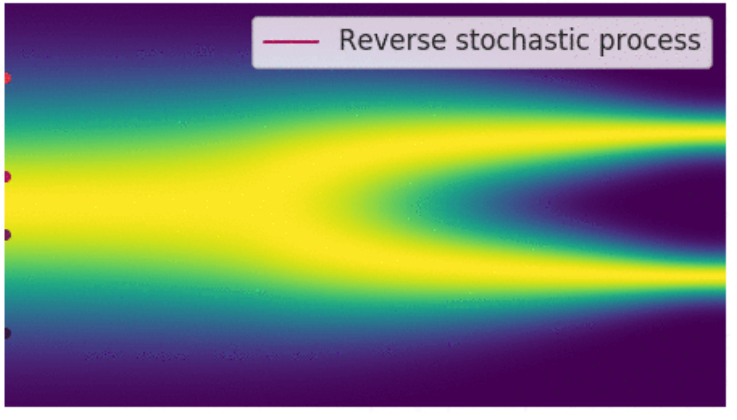
describing the diffusion in the infinitesimal limit.



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.

The score $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ of the marginal diffused density $q(\mathbf{x}_t)$ is not tractable, but can be estimated using denoising score matching (DSM) by solving

$$\arg\min_{\theta} \mathbb{E}_{q(\mathbf{x}_0)} \mathbb{E}_{t \sim U[0,T]} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} ||s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0)||_2^2,$$

which will result in $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ because of the outer expectation over $q(\mathbf{x}_0)$.

This is just the **same objective** as for VDMs! (See Interpretation 3)

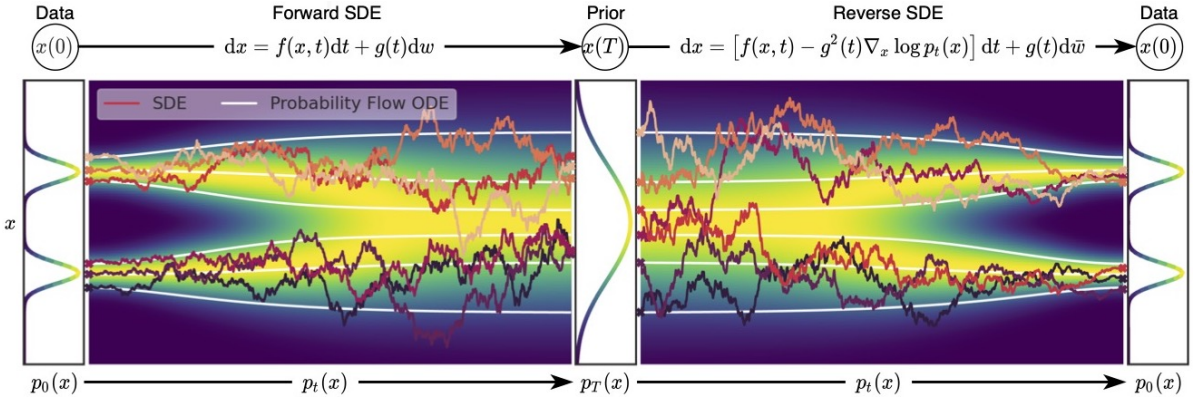
Probability flow ODE

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!



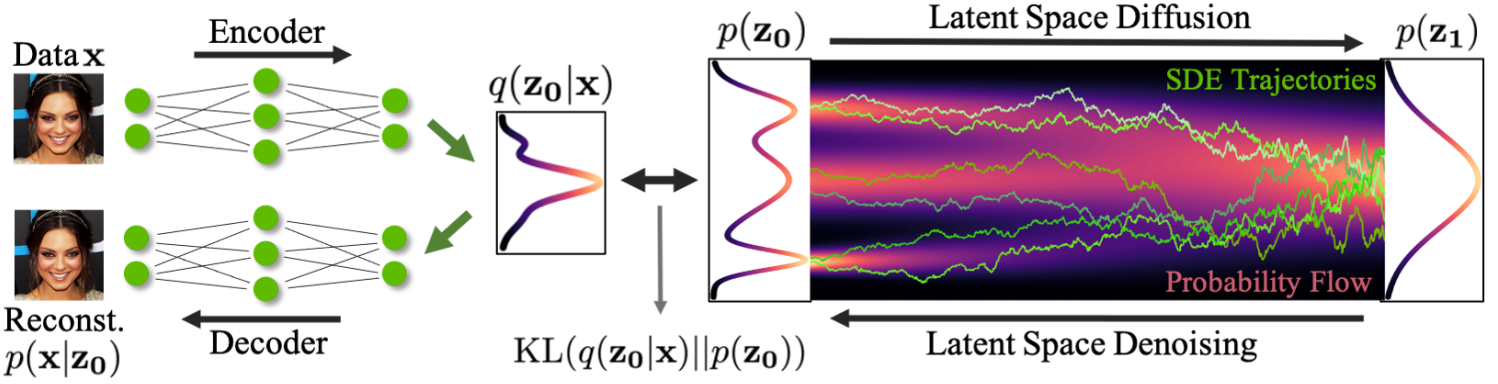
—

Credits: Song, 2021.

Latent-space diffusion models

Directly modeling the data distribution can be make the denoising process difficult to learn. A more effective approach is to combine VAEs with a diffusion prior.

- The distribution of latent embeddings is simpler to model.
- Diffusion on non-image data is possible with tailored autoencoders.



—

Credits: Vahdat et al, 2021.

The end.

—

Credits: Blattmann et al, 2023. Prompt: "A teddy bear is playing the electric guitar, high definition, 4k."