12. Diffusion models.

Ch. 25. Probabilistic Machine Learning: Advanced Topics (Murphy, 2025)

Probabilistic Machine Learning Reading Group

Alberto Suárez

April 15, 2026

Computer Science Department
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Generative AI: diffusion models (DMs)

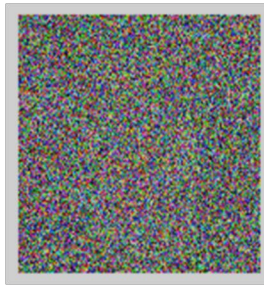
Consider the sample $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}; \quad \mathbf{x}_i \sim p_0(\mathbf{x})$

Data distribution (unknow)

E.g., images of cats

Goal: Generate $\mathbf{x}_{new} \sim p_0(\mathbf{x})$ starting from a sample of $\pi(\mathbf{x})$

Simple distribution E.g., $N(\mathbf{0}, \mathbf{I})$



Source: https://cvpr2022-tutorial-diffusion-models.github.io/ (retrieved 2023-08-15)

Generative AI: diffusion models (DMs)

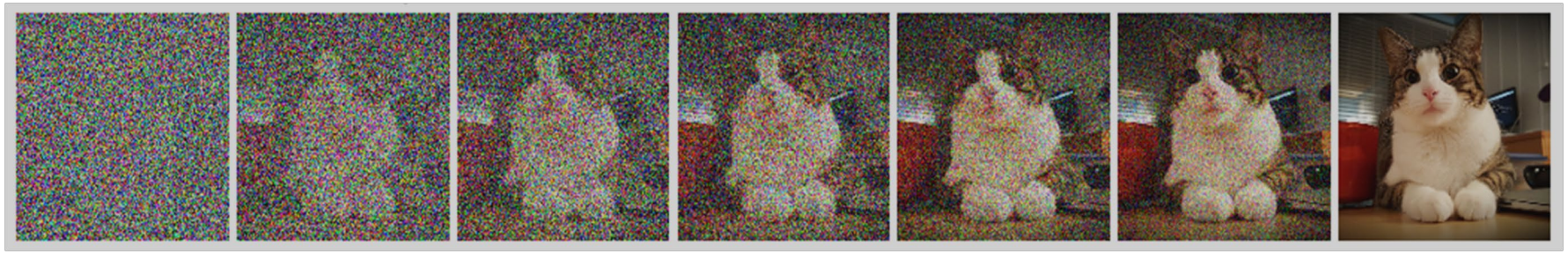
Consider the sample $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}; \quad \mathbf{x}_i \sim p_0(\mathbf{x})$

Data distribution (unknow)

E.g., images of cats

Goal: Generate $\mathbf{x}_{new} \sim p_0(\mathbf{x})$ starting from a sample of $\pi(\mathbf{x})$

Simple distribution E.g., $N(\mathbf{0}, \mathbf{I})$



Source: https://cvpr2022-tutorial-diffusion-models.github.io/ (retrieved 2023-08-15)

Then, reverse time

Image from https://cvpr2022-tutorial-diffusion-models.github.io/ [accessed 2023-10-25]

DMs: Time-reversal of a noise injection process

Forward process (noise injection)

Reverse process (noise removal)

Image from https://cvpr2022-tutorial-diffusion-models.github.io/ [accessed 2023-10-25]

Denoising diffusion probabilistic models (DDPMs)

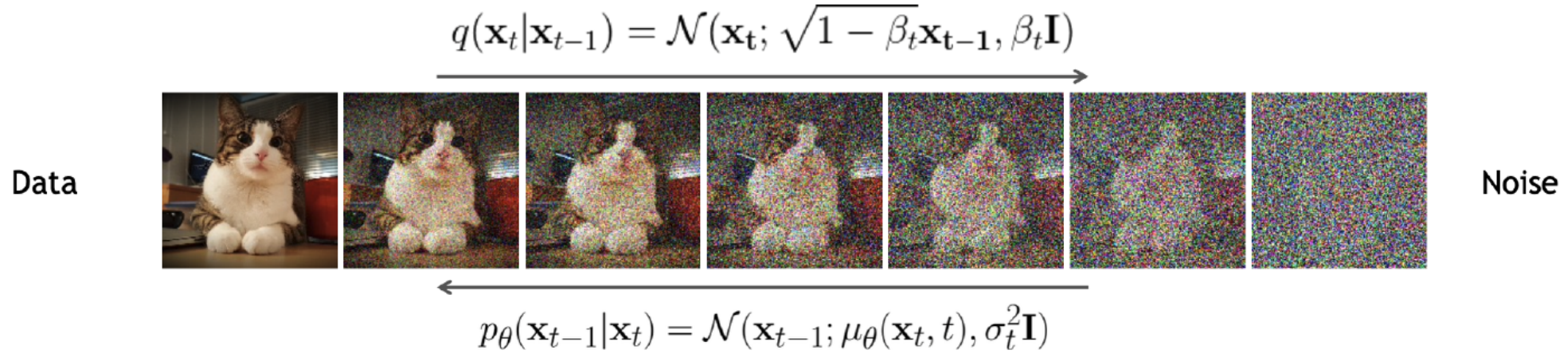


Figure 25.1: Denoising diffusion probabilistic model. The forwards diffusion process, $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, implements the (non-learned) inference network; this just adds Gaussian noise at each step. The reverse diffusion process, $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, implements the decoder; this is a learned Gaussian model. From Slide 16 of [KGV22]. Used with kind permission of Arash Vahdat.

Denoising diffusion probabilistic models (DDPMs)

Discrete-time Markov chains:

- **Forward process (noise injection)**

$$\mathbf{x}_0 \sim p_0(\mathbf{x})$$

$$\mathbf{x}_t = \sqrt{1-\beta_t}\,\mathbf{x}_{t-1} + \sqrt{\beta_t}\,\mathbf{Z}_t; \qquad \mathbf{Z}_t \sim N(\mathbf{0}, \mathbf{I}), \quad t = 1, 2, \ldots, T$$

Limiting distribution: $\mathbf{x}_\infty = \lim_{t\to\infty} \mathbf{x}_t \sim N(\mathbf{0}, \mathbf{I})$

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

iid Gaussian white noise

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_t; \quad \alpha_t = 1 - \beta_t$$

Independent of $\mathbf{x}_0$

- **Reverse process (noise removal)**

$$\mathbf{x}_T \sim N(\mathbf{0}, \mathbf{I})$$

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}(\mathbf{x}_t, t; \boldsymbol{\theta}) + \sigma_t\,\mathbf{Z}_t; \qquad \mathbf{Z}_t \sim N(\mathbf{0}, \mathbf{I}), \quad t = T, T-1, \ldots, 1$$

iid Gaussian white noise

- For sufficiently large T, and well-trained model $\boldsymbol{\mu}(\mathbf{x}_t, t; \boldsymbol{\theta})$

$$(\mathbf{x}_{t-1} | \mathbf{x}_T) \sim p_0(\mathbf{x})$$

DDPM: Noise injection (forward diffusion)

- Transition probability kernel:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t\middle|\sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right), \qquad t = 1, \ldots, T$$

- Trajectory:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} \mathcal{N}\left(\mathbf{x}_t\middle|\sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right)$$

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

- Marginal conditioned to $\mathbf{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t\middle|\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)$$

- Unconditional marginal

$$p_t(\mathbf{x}_t) = \int p_0(\mathbf{x}_0)\, q(\mathbf{x}_t|\mathbf{x}_0)\, d\mathbf{x}_0$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \mathcal{N}\left(\mathbf{x}_t\middle|\sqrt{\bar{\alpha}_t}\,\mathbf{x}_n, (1-\bar{\alpha}_t)\mathbf{I}\right)$$

Mixture of Gaussians

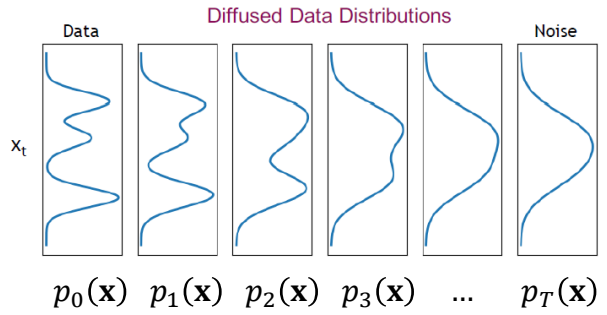


Figure 25.2: Illustration of a diffusion model on 1d data. The forwards diffusion process gradually transforms the empirical data distribution $q(\boldsymbol{x}_0)$ into a simple target distribution, here $q(\boldsymbol{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. To generate from the model, we sample a point $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then run the Markov chain backwards, by sampling $\boldsymbol{x}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ until we get a sample in the original data space, $\boldsymbol{x}_0$. From Slide 19 of [KGV22]. Used with kind permission of Arash Vahdat.

DDPM: Noise removal (reverse diffusion)

- Transition probability kernel:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}\middle|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}\right)$$

Markov property: $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1})$

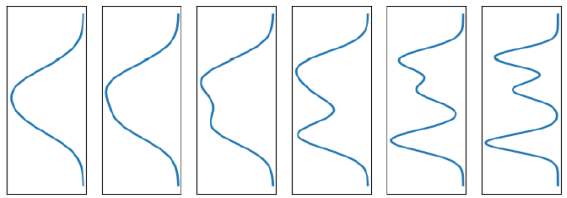
Bayes' theorem + Markov property:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t; \quad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t; \quad t = 1, \ldots, T$$

- Model $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx p(\mathbf{x}_{t-1}|\mathbf{x}_t; \boldsymbol{\theta})$

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



It is common to asume that $\boldsymbol{\Sigma}(\mathbf{x}_t, t; \boldsymbol{\theta}) = \sigma_t^2\mathbf{I}$

- Either $\sigma_t^2 = \beta_t$ (upper bound of reverse process entropy)
- Or $\sigma_t^2 = \tilde{\beta}_t$ (lower bound of reverse process entropy)

DDPM: Training by maximizing **ELBO** (**E**vidence **L**ower **BO**und)

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

Contribution of $\mathbf{x}_0 \sim p_0(\mathbf{x})$ to the evidence (marginal or type II likelihood)

$$\log p_0(\mathbf{x}_0;\boldsymbol{\theta}) = \log \int d\mathbf{x}_{1:T}\, p_{0:T}(\mathbf{x}_{0:T};\boldsymbol{\theta}) = \log \int d\mathbf{x}_{1:T}\, q(\mathbf{x}_{1:T}|\mathbf{x}_0)\frac{p_{0:T}(\mathbf{x}_{0:T};\boldsymbol{\theta})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}$$

Jensen's inequality

ELBO: lower bound of the evidence

$$\geq \int d\mathbf{x}_{1:T}\, q(\mathbf{x}_{1:T}|\mathbf{x}_0)\log\frac{p_{0:T}(\mathbf{x}_{0:T};\boldsymbol{\theta})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

$\mathcal{L}(\boldsymbol{\theta};\mathbf{x}_0)$: *minus* ELBO

$$= \int d\mathbf{x}_T\, q(\mathbf{x}_T|\mathbf{x}_0)\log p_T(\mathbf{x}_T) + \int d\mathbf{x}_{1:T}\, q(\mathbf{x}_{1:T}|\mathbf{x}_0)\sum_{t=1}^{T}\log\frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t;\boldsymbol{\theta})}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \stackrel{\text{def}}{=} -\mathcal{L}(\boldsymbol{\theta};\mathbf{x}_0)$$

$$\mathcal{L}(\boldsymbol{\theta};\mathbf{x}_0) = D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)|p_T(\mathbf{x}_T)) + \textstyle\sum_{t=2}^{T} L_{t-1}(\boldsymbol{\theta};\mathbf{x}_0) - \int d\mathbf{x}_T\, q(\mathbf{x}_1|\mathbf{x}_0)\log p(\mathbf{x}_0|\mathbf{x}_1;\boldsymbol{\theta})$$

Markov property + Bayes' theorem:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\, q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}$$

$$L_{t-1}(\boldsymbol{\theta};\mathbf{x}_0) \stackrel{\text{def}}{=} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)|p(\mathbf{x}_{t-1}|\mathbf{x}_t;\boldsymbol{\theta}))$$

DDPM: Training

$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_{t-1}\middle|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}\right)$

$p(\mathbf{x}_{t-1}|\mathbf{x}_t;\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}(\mathbf{x}_t, t;\boldsymbol{\theta}), \sigma_t^2\boldsymbol{I})$

$$L_{t-1}(\boldsymbol{\theta};\mathbf{x}_0) \stackrel{\text{def}}{=} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)|p(\mathbf{x}_{t-1}|\mathbf{x}_t;\boldsymbol{\theta}))$$

$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t;\ \ \boldsymbol{\epsilon}_t \sim N(0,1)$
$\Rightarrow \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 = \mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t;$

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) = \frac{\beta_t}{1-\bar{\alpha}_t}\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t = \frac{\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

$\bar{\alpha}_t = \bar{\alpha}_{t-1}\alpha_t$
$\Rightarrow \sqrt{\bar{\alpha}_{t-1}} = \frac{1}{\sqrt{\alpha_t}}\sqrt{\bar{\alpha}_t}$

$$= \frac{\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_t + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

$\alpha_t + \beta_t = 1$

$\bar{\alpha}_{t-1}\alpha_t = \bar{\alpha}_t$

$$= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\frac{\beta_t + \alpha_t - \alpha_t\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$$

Model the injected noise with NNET $\boldsymbol{\epsilon}(\mathbf{x}_t, t;\boldsymbol{\theta})$

$$\boldsymbol{\mu}(\mathbf{x}_t, t;\boldsymbol{\theta}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}(\mathbf{x}_t, t;\boldsymbol{\theta})\right)$$

Mean mismatch term in $D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)|p(\mathbf{x}_{t-1}|\mathbf{x}_t;\boldsymbol{\theta}))$

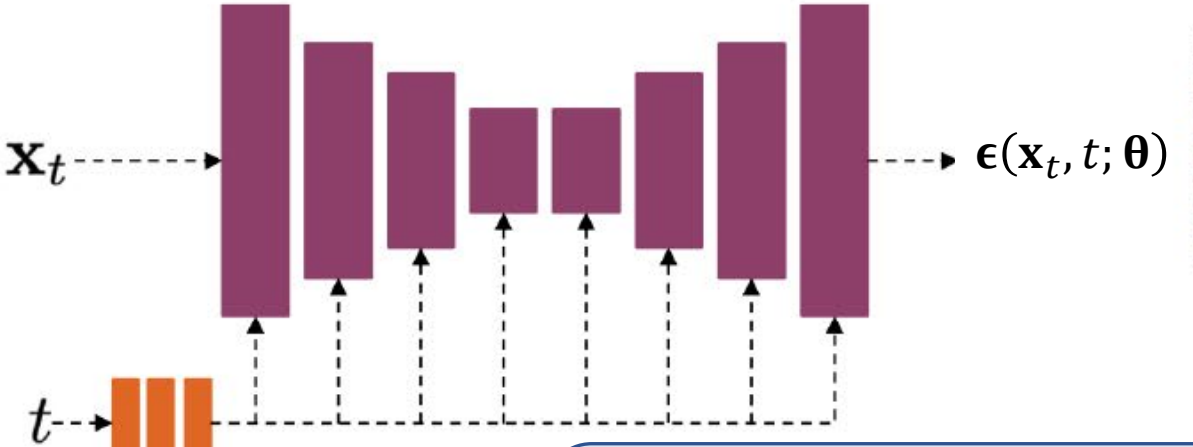
$$\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}(\mathbf{x}_t, t;\boldsymbol{\theta})\|^2$$

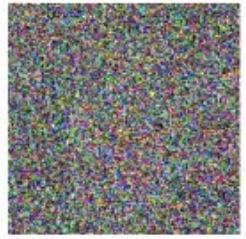
DDPM: U-NET for denoising

Other architectures are possible (e.g., transformers) [PX22, Li+22, Bao+22a]

Image perturbed with fwd noise injection
$\mathbf{x}_0 \sim p_0(\mathbf{x})$; $\boldsymbol{\epsilon}_t \sim N(\mathbf{0}, \mathbf{I})$
$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t$;

Estimation of the noise injected
$\boldsymbol{\epsilon}(\mathbf{x}_t, t; \boldsymbol{\theta}) \sim \boldsymbol{\epsilon}_t$





Larger t corresponds to higher noise levels

Encode t (using, e.g., sinusoidal positional encoding or random Fourier features with fixed frequencies) and incorporate it into the network (e.g. by spatial addition or by conditioning the group norm layers)

Figure 25.3: Illustration of the U-net architecture used in the denoising step. From Slide 26 of [KGV22]. Used with kind permission of Arash Vahdat.

DDPM: Loss function

$$\lambda(t) = \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)}$$

Empirically, instead of using this value, which corresponds to maximing the likelihood by maximing the ELBO, the replacement $\lambda(t) \longrightarrow 1$ works better, in general.

$$L_{t-1}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{\substack{\mathbf{x}_0 \sim p_0(\mathbf{x}) \\ \boldsymbol{\epsilon}_t \sim N(\mathbf{0},\mathbf{I})}} [\lambda(t) \, \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}(\mathbf{x}_t, t; \boldsymbol{\theta})\|^2]$$

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

$\lambda(t) \longrightarrow 1$

$$\Longrightarrow \quad L_{simple}(\boldsymbol{\theta}) = \mathbb{E}_{t \sim U[0,1]} \left[\mathbb{E}_{\substack{\mathbf{x}_0 \sim p_0(\mathbf{x}) \\ \boldsymbol{\epsilon}_t \sim N(\mathbf{0},\mathbf{I})}} \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}(\mathbf{x}_t, t; \boldsymbol{\theta})\|^2 \right]$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}_0 \sim p_0(\mathbf{x})} \left[\mathbb{E}_{t \sim U[0,1]} \left[\mathbb{E}_{\boldsymbol{\epsilon}_t \sim N(\mathbf{0},\mathbf{I})} \left[\left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}\left(\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \boldsymbol{\epsilon}_t, t; \boldsymbol{\theta} \right) \right\|^2 \right] \right] \right]$$

Mimimize $L_{simple}(\boldsymbol{\theta})$

Algorithm 25.1: Training a DDPM model with L_{simple}.

1 **while** *not converged* **do**
2 $\quad \mathbf{x}_0 \sim p_0(\mathbf{x})$ — Sample from original data
3 $\quad t \sim \text{Unif}(\{1, \ldots, T\})$
4 $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5 $\quad$ Take gradient descent step on $\nabla_{\boldsymbol{\theta}} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}} \left(\sqrt{\bar{\alpha}_t} \boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}, t \right) \|^2$

DDPM: Generation of new samples

- Ancestral sampling with $p(\mathbf{x}_{t-1}|\mathbf{x}_t;\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}(\mathbf{x}_t,t;\boldsymbol{\theta}),\sigma_t^2\boldsymbol{I})$

Algorithm 25.2: Sampling from a DDPM model.

1 $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2 **foreach** $t = T,\ldots,1$ **do**
3 $\quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
4 $\quad \boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)\right) + \sigma_t\boldsymbol{\epsilon}_t$
5 Return $\boldsymbol{x}_0$

$$\boldsymbol{\mu}(\mathbf{x}_t,t;\boldsymbol{\theta}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}(\mathbf{x}_t,t;\boldsymbol{\theta})\right)$$

Sampling from a PDF: Langevin dynamics

GOAL: Generate iid samples $\mathbf{x} \sim p_0(\mathbf{x})$

Difficult to sample / no explicit form

INPUT: Samples from a prior $\pi(\mathbf{x})$; for instance, $N(\mathbf{0}, \mathbf{I})$

Easy to sample

METHOD: Simulate trajectories of an SDE whose stationary solution is $p_0(\mathbf{x})$

$$d\mathbf{x}(t) = \varepsilon \, \nabla_{\mathbf{x}} \log p_0(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}(t)} \, dt + \sqrt{2\varepsilon} \, d\mathbf{W}(t);$$

$$d\mathbf{W}(t) = \sqrt{dt} \, \mathbf{Z}; \quad \mathbf{Z} \sim N(\mathbf{0}, \mathbf{I})$$

Continuous t

Score function

Properties:

1. $p_0(\mathbf{x})$ is the stationary distribution of the corresponding Fokker-Planck equation.
2. $\mathcal{L}(\mathbf{x}(t))$, the distribution of $\mathbf{x}(t)$ approaches $p_0(\mathbf{x})$ in the limit $t \longrightarrow \infty$

Numerical simulation of Langevin dynamics

Euler-Maruyama numerical integration scheme:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \alpha \nabla_{\mathbf{x}} \log p_0(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_{t-1}} + \sqrt{2\alpha}\,\mathbf{Z}_t; \quad \mathbf{Z}_t \sim N(\mathbf{0}, \mathbf{I})$$

Discrete t

$$\mathbf{x}_0 \sim \pi(\mathbf{x})$$

$$\mathbf{x}_t \stackrel{\text{def}}{=} \mathbf{x}(t\,\Delta t), \quad t = 1,2,\ldots,T$$

$$\Delta t \to 0^+$$

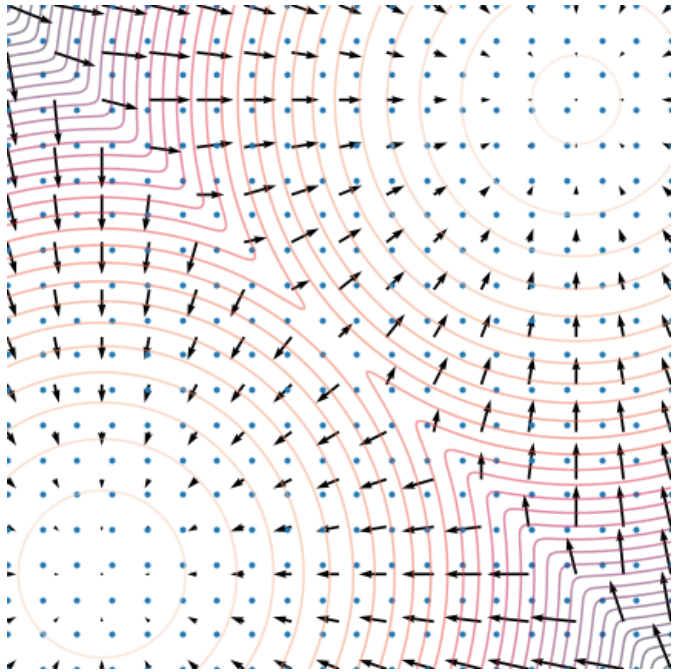
$$\alpha = \varepsilon\,\Delta t \to 0^+$$

- In the limit $T \to \infty, \quad \mathbf{x}_T \sim p_0(\mathbf{x})$

Langevin dynamics: Mixture of two Gaussians

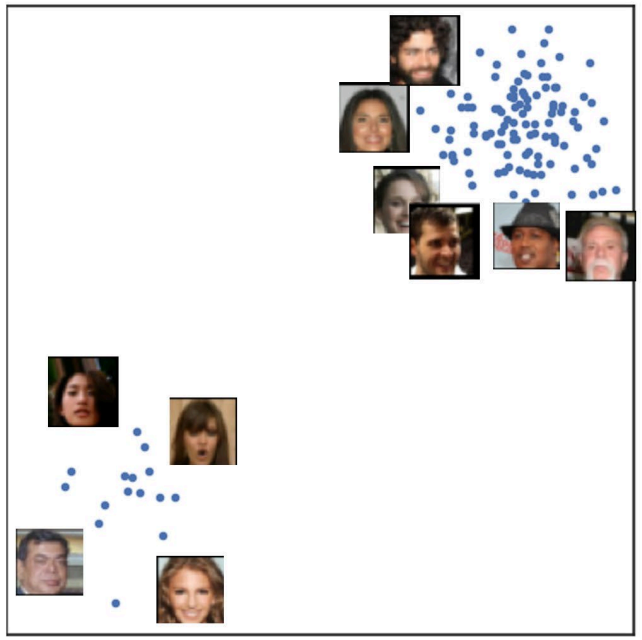
$\pi(\mathbf{x})$: Uniform

$p_0(\mathbf{x})$: Mixture of Gaussians



Source: https://yang-song.net/blog/2021/score/

Generative model: Score matching (learning) + Langevin dynamics (simulation)

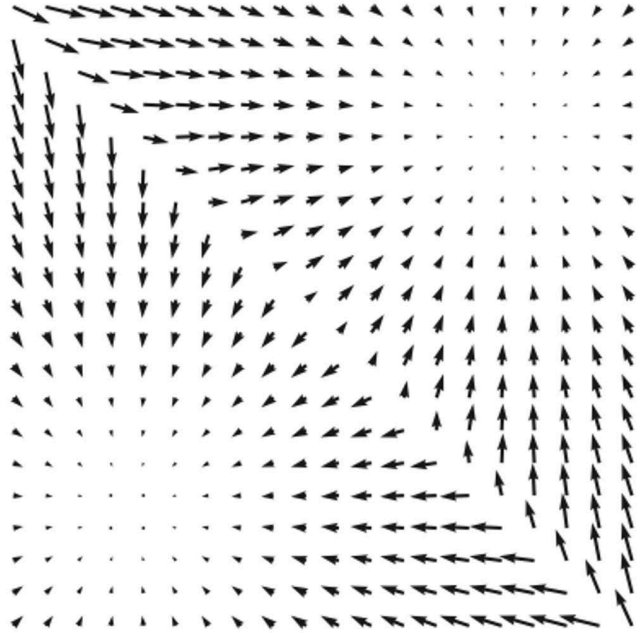


Data samples

$$\{\mathbf{x}_n\}_{n=1}^{N} \underset{iid}{\sim} p_0(\mathbf{x})$$

score matching

Learning

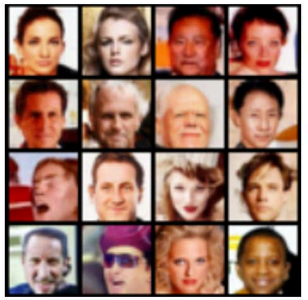


Scores

$$s(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\mathbf{x}} \log p_0(\mathbf{x})$$

Langevin dynamics

Simulation



New samples

Source: https://yang-song.net/blog/2021/score/

Score matching

$\mathbf{s}(\mathbf{x};\boldsymbol{\theta})$: model of the score (e.g., a neural network)

Approximate the score function: $\nabla_{\mathbf{x}} \log \pi^*(\mathbf{x}) \approx \mathbf{s}(\mathbf{x};\boldsymbol{\theta}^*)$

Fisher divergence

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}\sim p_0(\mathbf{x})}[\|\mathbf{s}(\mathbf{X};\boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log p_0(\mathbf{x})|_{\mathbf{x}=\mathbf{X}}\|_2^2]$$

$$= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}\sim p_0(\mathbf{x})}[\|\mathbf{s}(\mathbf{X};\boldsymbol{\theta})\|_2^2 + 2\, Tr(\nabla_{\mathbf{x}}\mathbf{s}(\mathbf{x};\boldsymbol{\theta})|_{\mathbf{x}=\mathbf{X}})]$$

$$\approx \arg\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{n=1}^{N} \left(\|\mathbf{s}(\mathbf{x}_n;\boldsymbol{\theta})\|_2^2 + 2\, Tr\left(\nabla_{\mathbf{x}}\mathbf{s}(\mathbf{x};\boldsymbol{\theta})|_{\mathbf{x}=\mathbf{x}_n}\right)\right)$$

Sample estimate
$\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$
$\mathbf{x}_n \sim p_0(\boldsymbol{x})$

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(Apr):695–709, 2005.

Difficulties: Low data density regions

$$\boldsymbol{\theta}^* \approx \arg\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{n=1}^{N}\left(\|\mathbf{s}(\mathbf{x}_n;\boldsymbol{\theta})\|_2^2 + 2\,Tr\left(\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{s}(\mathbf{x};\boldsymbol{\theta})|_{\mathbf{x}=\mathbf{x}_n}\right)\right)$$

- **Manifold hypothesis**: For real-world problems $p_0(\mathbf{x})$, the distribution the data is likely to be concentrated on low dimensional manifolds.
- Since in the cost function that is optimized, the squared deviations are weighted by $p_0(\mathbf{x})$, the **estimates of the score outside of this manifold are typically rather poor**.
- However, the initial values for the Langevin dynamics are sampled from a prior $\pi(\mathbf{x})$ that is, in general, different from $p_0(\mathbf{x})$.
- Therefore, most of these initial values will be in low data density regions.
- As a result, the Langevin dynamics will be misled already in these early first steps because of the poor estimates of the score and will fail to converge to samples that are representative of the original data.

Typically, $\pi(\mathbf{x})$ is very different from $p_0(\mathbf{x})$ $\Longrightarrow$ inaccurate score estimates in regions more likely to be sampled with $\pi(\mathbf{x})$.

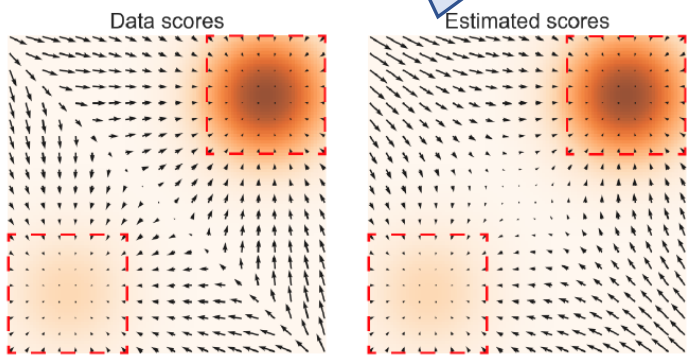


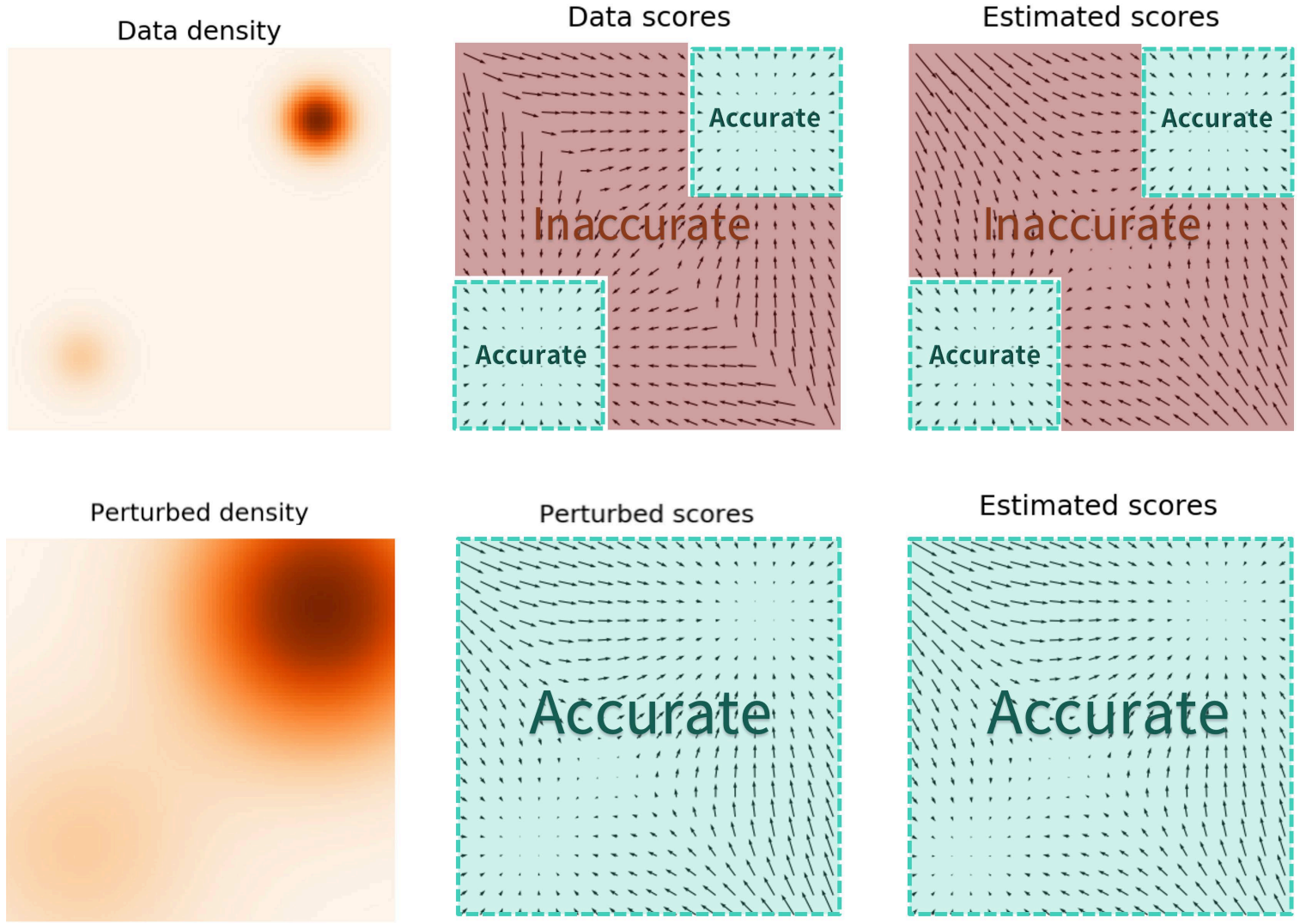
Figure 2: **Left**: $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$; **Right**: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$. The data density $p_{\text{data}}(\mathbf{x})$ is encoded using an orange colormap: darker color implies higher density. Red rectangles highlight regions where $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) \approx \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})$.

Source: Song & Ermon (2019)

Langevin dynamics with noise perturbation

Noise injection

Tradeoff:
The larger the noise injection, the more accurate the estimates of the perturbed model's score
but
the more different the learned model from the original one.

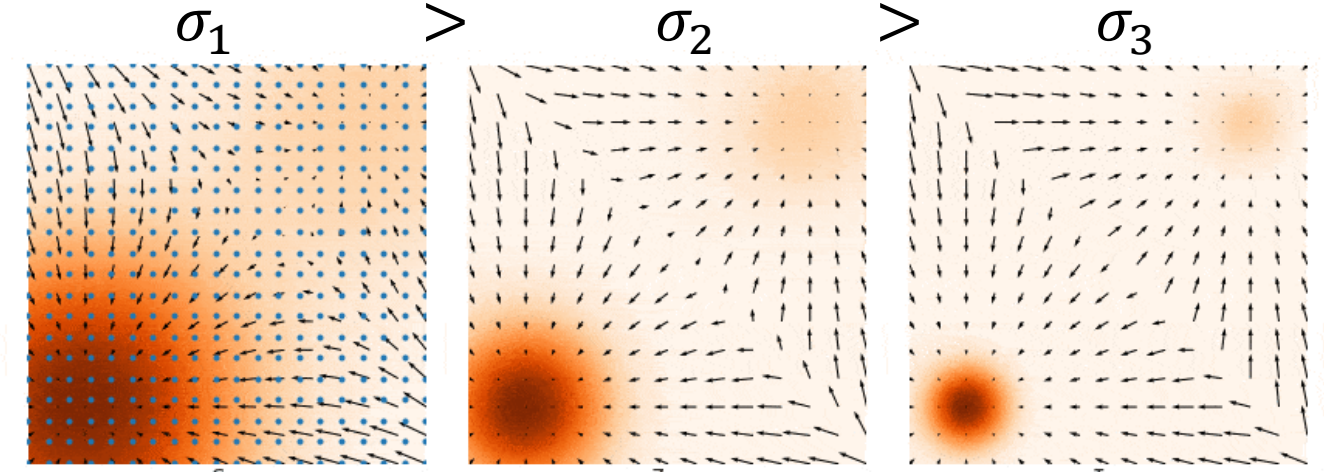


Source: https://yang-song.net/blog/2021/score/

Noise-conditional Score-based Model

Multiple noise perturbation + annealed Langevin dynamics

- Fix a sequence of decreasing noise levels: $\sigma_1 > \sigma_2 > \cdots > \sigma_L$
- Contaminated data distribution: $p_{\sigma_i}(\mathbf{x}) = \int p_0(\mathbf{x}')\, \phi\left(\mathbf{x}; \mathbf{x}', \sigma_i^2 \mathbf{I}\right) d\mathbf{x}, \quad i = 1, 2, \ldots, L$
- Noise-conditional score networks: $\mathbf{s}(\mathbf{X}; \boldsymbol{\theta}, \sigma_i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}), \quad i = 1, 2, \ldots, L$
- Training objective: $\sum_{i=1}^{L} \lambda_i\, \mathbb{E}_{\mathbf{X} \sim p_{\sigma_i}(\mathbf{x})} \left[\left\| \mathbf{s}(\mathbf{X}; \boldsymbol{\theta}, \sigma_i) - \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) \big|_{\mathbf{x}=\mathbf{X}} \right\|_2^2 \right]$



Source: https://yang-song.net/blog/2021/score/

Weighted sum of Fisher divergences.
Weight function: λ_i
Common choice: $\lambda_i = \sigma_i^2$

Estimate by averaging over contaminated samples:
$\mathbf{x}_n(\sigma_i) = \mathbf{x}_n + \sigma_i \mathbf{Z}_n; \quad \mathbf{Z}_n \sim N(\mathbf{0}, \mathbf{I})$
$n = 1, 2, \ldots, N; i = 1, 2, \ldots, L$

Annealed Langevin dynamics

Algorithm 1 Annealed Langevin dynamics [1]

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\mathbf{x}_0$
for $i \leftarrow 1$ to L **do**
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ▷ α_i is the step size.
    for $t \leftarrow 1$ to T **do**
        Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
        $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \alpha_i \, \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}, \sigma_i) + \sqrt{2\alpha_i} \, \mathbf{z}_t$
    $\mathbf{x}_0 \leftarrow \mathbf{x}_T$
if denoise $\mathbf{x}_T$ **then**
    return $\mathbf{x}_T + \sigma_T^2 \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_T, \sigma_T)$
else
    return $\mathbf{x}_T$

Source: Yang Song and Stefano Ermon. 2020. Improved techniques for training score-based generative models. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020). Curran Associates Inc., Red Hook, NY, USA, Article 1043, 12438–12448. [https://arxiv.org/abs/2006.09011]

Diffusion-based generative models

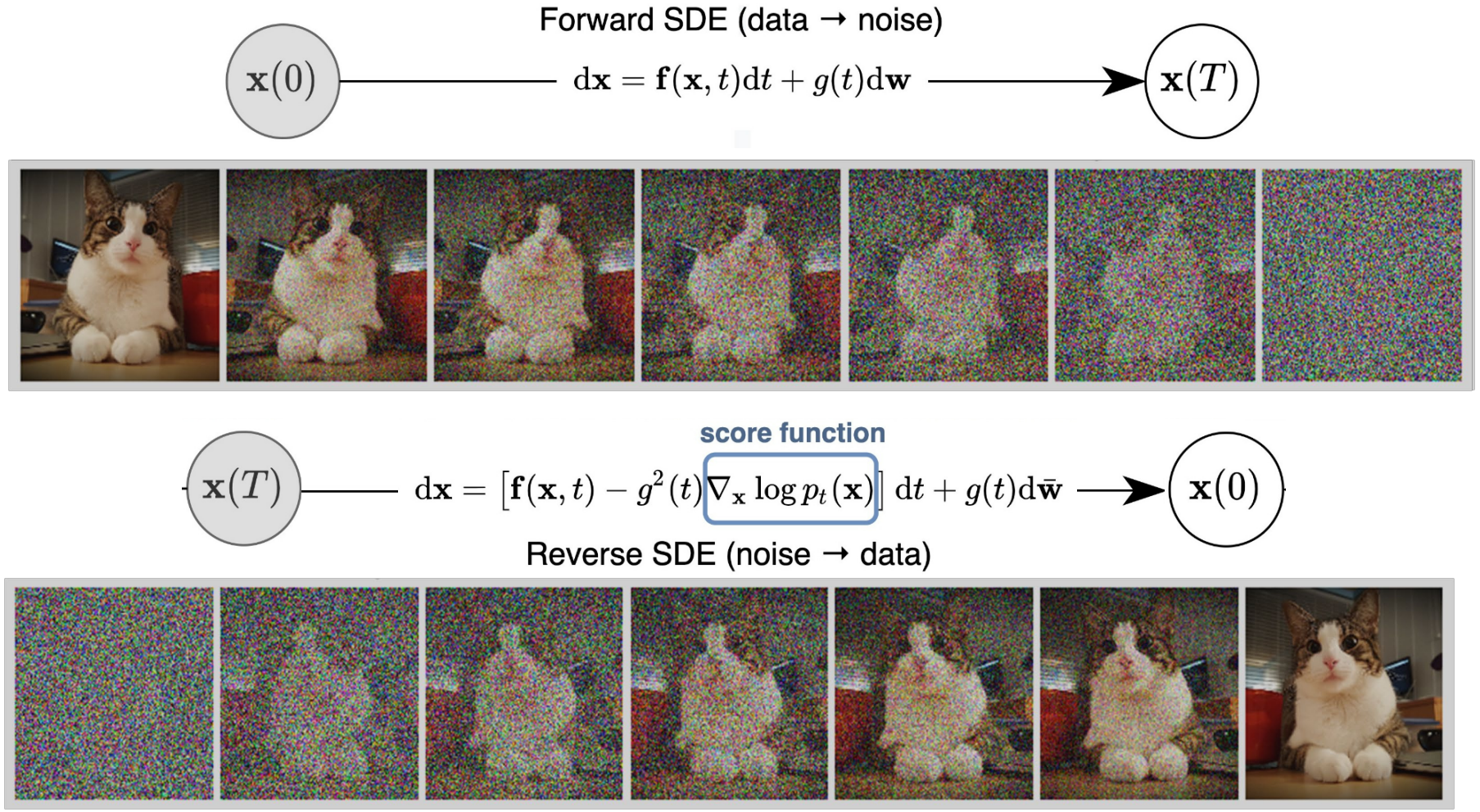


Image from https://cvpr2022-tutorial-diffusion-models.github.io/ [accessed 2023-10-25]

Score-based generative AI: Diffusion models

Learn to **invert a stochastic process**

- Simulate foward process in $[0, T]$:

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)dt + g(t)d\mathbf{W}(t)$$

$$\mathbf{x}(0) \sim p_0(\mathbf{x}) \Rightarrow \mathbf{x}(t) \sim p_t(\mathbf{x})$$

$$\mathbf{x}(T) \sim p_T(\mathbf{x}) \xrightarrow[T\to\infty]{} \pi_T(\mathbf{x})$$

Noise injection

Drift term: $\mathbf{f}(\mathbf{x}(t), t)$
Diffusion term: $g(t)\,\mathbf{I}$

Model the score function $\mathbf{s}(\mathbf{x}, t; \boldsymbol{\theta}^*) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

Initial distribution (unknown)

Limiting reference distribution (known & easy to sample)

- Simulate reverse process in $[T, 0]$:

$$d\overleftarrow{\mathbf{x}}(t) = (\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\overleftarrow{\mathbf{x}}(t)))dt + g(t)\,d\mathbf{W}(t)$$

$$\overleftarrow{\mathbf{x}}(T) \sim \pi_T(\mathbf{x})$$

$$\overleftarrow{\mathbf{x}}(0) \approx \mathbf{x} \sim p_0(\mathbf{x})$$

Denoising

Drift term: $\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\overleftarrow{\mathbf{x}}(t))$
Diffusion term: $g(t)\,\mathbf{I}$

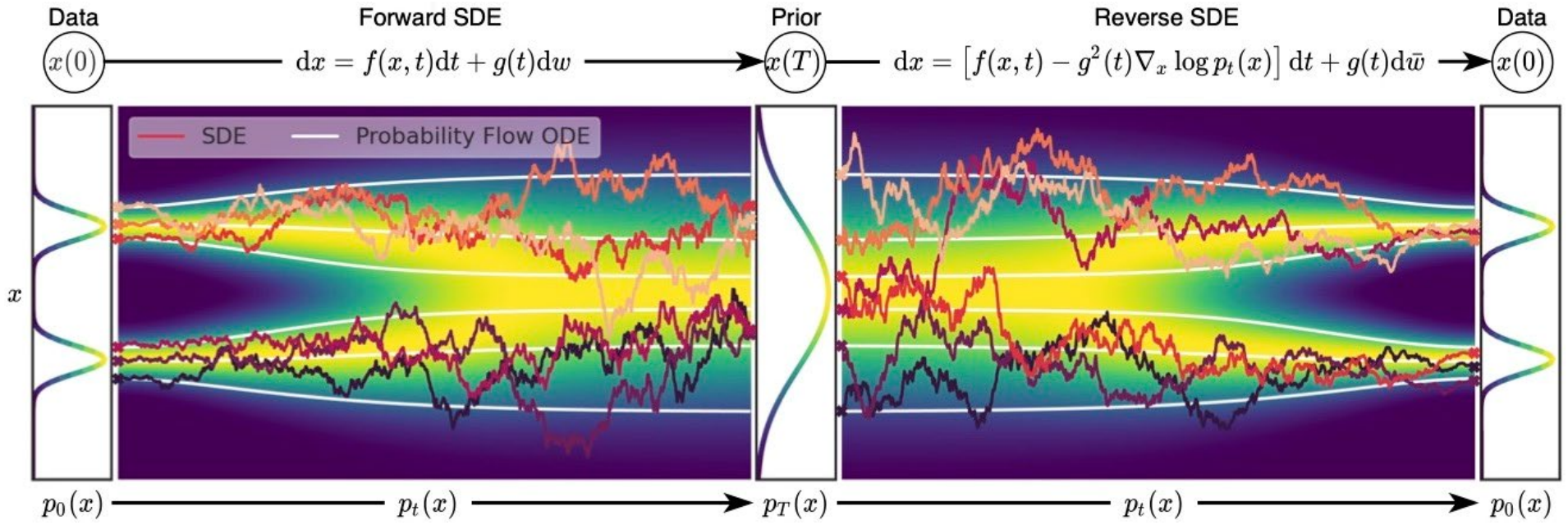
Approximation: $p_T(\mathbf{x}) \approx \pi_T(\mathbf{x})$

Approximation:
Numerical integration of the reverse SDE

Approximation: Use learned $\mathbf{s}(\mathbf{x}, t;\ \boldsymbol{\theta}^*) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

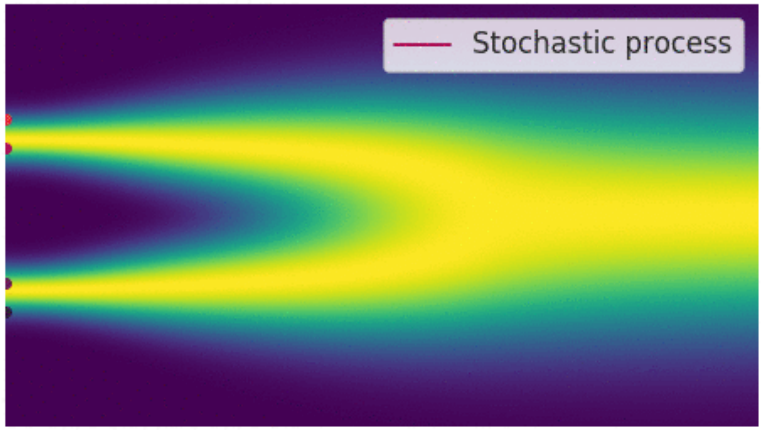
Synthetic sample from approximation of $p_0(\mathbf{x})$

Generative modeling using stochastic differential equations



Source: Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole "Score-Based Generative Modeling through Stochastic Differential Equations", International Conference on Learning Representations (2021) https://doi.org/10.48550/arXiv.2011.13456

Forward stochastic process: Noise injection



Generate noisy images by integrating the forward SDE in $[0, T]$

T is sufficiently large

$$d\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)dt + g(t)d\mathbf{W}(t)$$

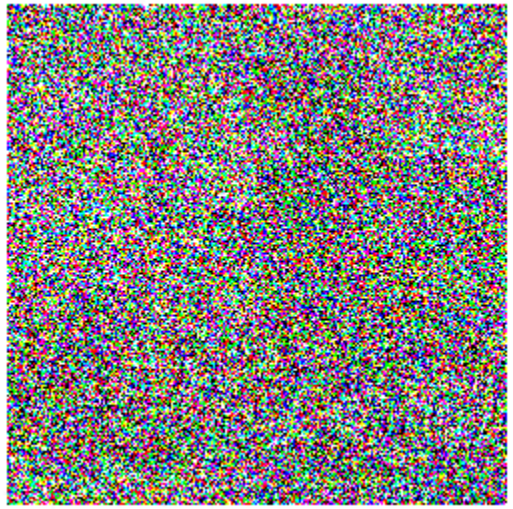
$$\mathbf{x}(0) \sim p_0(\mathbf{x})$$

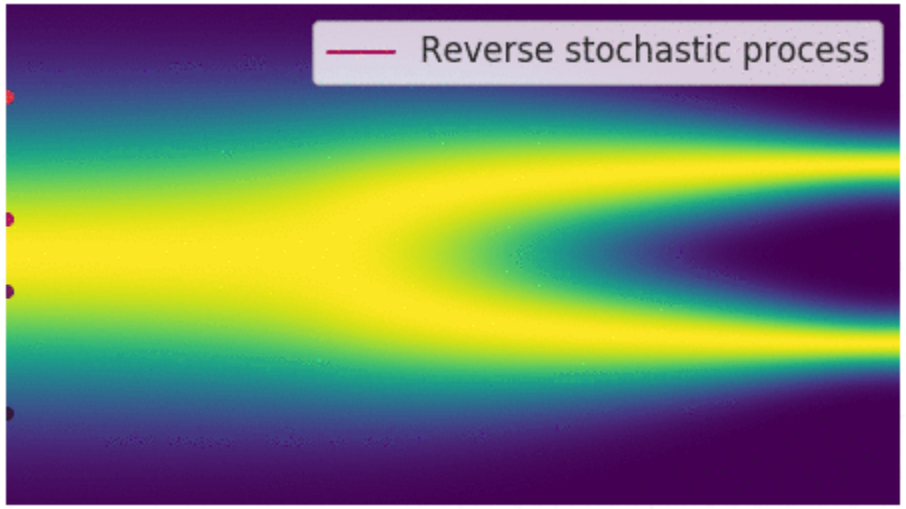
$d\mathbf{W}(t) = \sqrt{dt}\,\mathbf{Z}; \mathbf{Z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$

- Choose $\mathbf{f}(\mathbf{x}(t), t)$, the drift coefficient, so that $\mathbf{x}(t)|\mathbf{x}(0)$ can be expressed in closed form.

Source: https://*yang-song*.net/blog/2021/score/

Reverse stochastic process: Generation of synthetic samples





Generation of images: Simulate reverse SDE in $[T, 0]$ from $\overleftarrow{\mathbf{x}}(T) \sim \pi_T(\mathbf{x})$

$$d\overleftarrow{\mathbf{x}}(t) = \left(\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - g^2(t)\, \boldsymbol{\nabla}_{\mathbf{x}} \log p_t\left(\overleftarrow{\mathbf{x}}(\mathrm{t})\right)\right)dt + g(t)d\mathbf{W}(t); \implies \quad \overleftarrow{\mathbf{x}}(0) \approx \mathbf{x} \sim p_0(\mathbf{x})$$

- Euler-Maruyana
- Predictor-corrector
- Equivalent probability flow ODE: $d\overleftarrow{\mathbf{x}}(t) = \left(\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - \frac{1}{2} g^2(t)\, \boldsymbol{\nabla}_{\mathbf{x}} \log p_t\left(\overleftarrow{\mathbf{x}}(\mathrm{t})\right)\right) dt$

Same marginals $p_t\left(\mathbf{x}(\mathrm{t})\right)$ as forward SDE in $[0, T]$ from $\vec{\mathbf{x}}(0) \sim p_0(\mathbf{x})$
reverse SDE in $[T, 0]$ from $\overleftarrow{\mathbf{x}}(T) \sim p_T(\mathbf{x})$

Source: https://yang-song.net/blog/2021/score/

Brownian motion (BM): Variance exploding (VE)

Drift term: $\mathbf{0}$
Diffusion term: $g(t)\mathbf{I}$

- Forward process SDE: Integrate in $[0, T]$ from $\mathbf{x}(0)$

$$d\mathbf{x}(t) = g(t)\, d\mathbf{W}(t)$$

$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \sigma(t)\,\mathbf{Z} \sim \mathcal{N}(\mathbf{x}(0), \sigma^2(t)\mathbf{I})$

Exact

$$\boldsymbol{\mu}_{\mathbf{x}(0)}(t) = \mathbb{E}[\mathbf{x}(t)\,|\mathbf{x}(0)] = \mathbf{x}(0)$$

$$\sigma^2(t) = \int_0^t g^2(s)\, ds$$

For instance, $g(t) = \sigma^t$

$$\sigma^2(t) = \int_0^t g^2(s)\, ds$$

$$= \int_0^t \sigma^{2s} ds = \frac{1}{2\,\log\sigma}(\sigma^{2t} - 1)$$

- Reverse process SDE: Integrate in $[T, 0]$ from $\overleftarrow{\mathbf{x}}(T) \sim \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$

$\pi_T(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$

$$d\overleftarrow{\mathbf{x}}(t) = -g^2(t)\,\nabla_{\mathbf{x}} \log p_t\big(\overleftarrow{\mathbf{x}}(t)\big)\, dt + g(t) d\mathbf{W}(t); \quad \Longrightarrow \quad \overleftarrow{\mathbf{x}}(0) \approx \mathbf{x} \sim p_0(\mathbf{x})$$

Drift coefficient: $-g^2(t)\,\nabla_{\mathbf{x}} \log p_t(\overleftarrow{\mathbf{x}}(t))$
Diffusion coefficient: $g(t)\mathbf{I}$

Approximation: Use learned $\mathbf{s}(\mathbf{x}, t; \boldsymbol{\theta}^*) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

Ornstein-Uhlenbeck (OU) process: Variance preserving (VP) & sub-VP

- Forward process SDE: Integrate in $[0, T]$ from $\mathbf{x}(0)$

$$d\mathbf{x}(t) = -\frac{1}{2}\beta(t)\,\mathbf{x}(t)dt + g(t)d\mathbf{W}(t)$$

Drift term: $-\frac{1}{2}\beta(t)\ \mathbf{x}(t)$

Diffusion term: $g(t)\,\mathbf{I}$

Solution: $\mathbf{x}(t) = \boldsymbol{\mu}_{\mathbf{x}(0)}(t) + \sigma(t)\,\mathbf{Z} \underset{t\to\infty}{\longrightarrow} \sigma_\infty\,\mathbf{Z}$

$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

For instance, $g(t) = \sqrt{\beta(t)}$ (VP)

or $g(t) = \sqrt{\beta(t)\left(1 - e^{-2\int_0^t \beta(s)ds}\right)}$

(sub-VP: sub-variance preserving)

$$\boldsymbol{\mu}_{\mathbf{x}(0)}(t) = \mathbb{E}[\mathbf{x}(t)\,|\mathbf{x}(0)] = \mathbf{x}(0)\,e^{-\frac{1}{2}\int_0^t \beta(s)ds} \underset{t\to\infty}{\longrightarrow} \mathbf{0}$$

$$\sigma^2(t) = \int_0^t e^{-\int_s^t \beta(u)du} g^2(s)\,ds \quad \underset{t\to\infty}{\longrightarrow} \quad \sigma_\infty^2 = \int_0^\infty e^{-\int_s^\infty \beta(u)du} g^2(s)\,ds < \infty$$

- Reverse process SDE: Integrate in $[T, 0]$ from $\overleftarrow{\mathbf{x}}(T) \sim \mathcal{N}(\mathbf{0}, \sigma^2(T)\,\mathbf{I})$

$\pi_T(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \sigma^2(T)\,\mathbf{I})$

$$d\overleftarrow{\mathbf{x}}(t) = -\left(\frac{1}{2}\beta(t)\mathbf{x}(t) + g^2(t)\,\boldsymbol{\nabla}_{\mathbf{x}} \log p_t\left(\overleftarrow{\mathbf{x}}(\mathrm{t})\right)\right)dt + g(t)d\mathbf{W}(t); \quad \Longrightarrow \quad \overleftarrow{\mathbf{x}}(0) \approx \mathbf{x} \sim p_0(\mathbf{x})$$

Drift term: $-\left(\frac{1}{2}\beta(t)\mathbf{x}(t) + g^2(t)\,\boldsymbol{\nabla}_{\mathbf{x}} \log p_t(\overleftarrow{\mathbf{x}}(\mathrm{t}))\right)$

Diffusion term: $g(t)\,\mathbf{I}$

Noise schedules

- Constant schedule: $\beta(t) = \beta$
- Linear schedule: $\beta(t) = \beta_{min} + (\beta_{max} - \beta_{min})\frac{t}{T}$
- Cosine schedule:

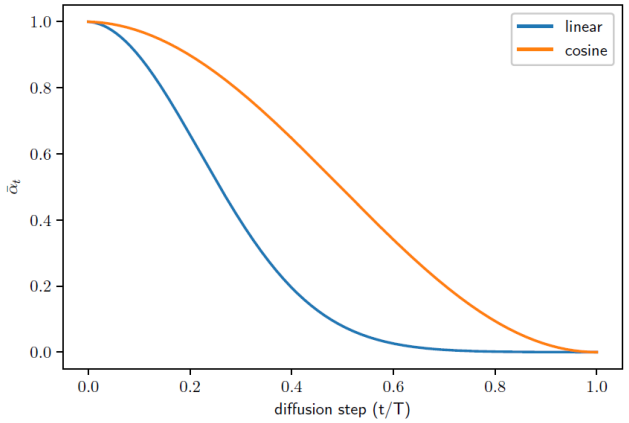


Figure 5. $\bar{\alpha}_t$ throughout diffusion in the linear schedule and our proposed cosine schedule.

$$\beta_t = \min\left(0.999,\quad 1 - \frac{\overline{\alpha_t}}{\overline{\alpha_{t-1}}}\right); \qquad \overline{\alpha_t} = \frac{\cos^2\left(\frac{\pi}{2}\,\frac{\frac{t}{T}+s}{1+s}\right)}{\cos^2\left(\frac{\pi}{2}\,\frac{s}{1+s}\right)}; \quad t = 0,1,2,\ldots.,T$$

$$\beta(t) = -\frac{d}{dt}\log\bar{\alpha}(t) = \frac{\pi}{T\,(1+s)}\tan\left(\frac{\pi}{2}\,\frac{\frac{t}{T}+s}{1+s}\right); \quad t \in [0,T]$$

$$T = 1;\ dT = 0.001, s = 0.008 \Longrightarrow \sqrt{\beta(0)\,dT} = 0.062 \leq \frac{2}{255} = 0.078 \text{ (pixel bin size)}$$

Zhehao Guo, Jiedong Lang, Shuyu Huang, Yunfei Gao, and Xintong Ding (2025)
A Comprehensive Review on Noise Control of Diffusion Model
https://arxiv.org/abs/2502.04669
Alexander Quinn Nichol, Prafulla Dhariwal "Improved Denoising Diffusion Probabilistic Models"
Proceedings of Machine Learning Research (ICML 2021) https://arxiv.org/abs/2102.09672

Stochastic Euler scheme: Regular grid

$$d\mathbf{x}(t) = \mathbf{a}(\mathbf{x}(t), t)dt + \mathbf{B}(\mathbf{x}(t), t)d\mathbf{W}(t)$$

Drift term: $\mathbf{a}(\mathbf{x}(t), t)$
Diffusion term: $\mathbf{B}(\mathbf{x}(t), t)$

- Initial condition: $\mathbf{x}(t_0) = \mathbf{x}_0$
- Integration interval $[t_0\,, t_0 + T]$

$$\Delta T = \frac{T}{N} \to 0^+$$

- Regular grid of integration points : $t_n = t_0 + n\Delta T; \quad n = 0, 1, \ldots, N$
- Approximate solution: $M \longrightarrow \infty$ trajectories

$$\mathbf{x}_0^{(m)} = \mathbf{x}_0; \quad m = 1, \ldots, M.$$

$$\mathbf{x}_{n+1}^{(m)} = \mathbf{x}_n^{(m)} + \mathbf{a}\left(\mathbf{x}_n^{(m)}, t_n\right)\Delta T + \mathbf{B}\left(\mathbf{x}_n^{(m)}, t_n\right)\sqrt{\Delta T}\,\mathbf{Z}_n^{(m)}$$

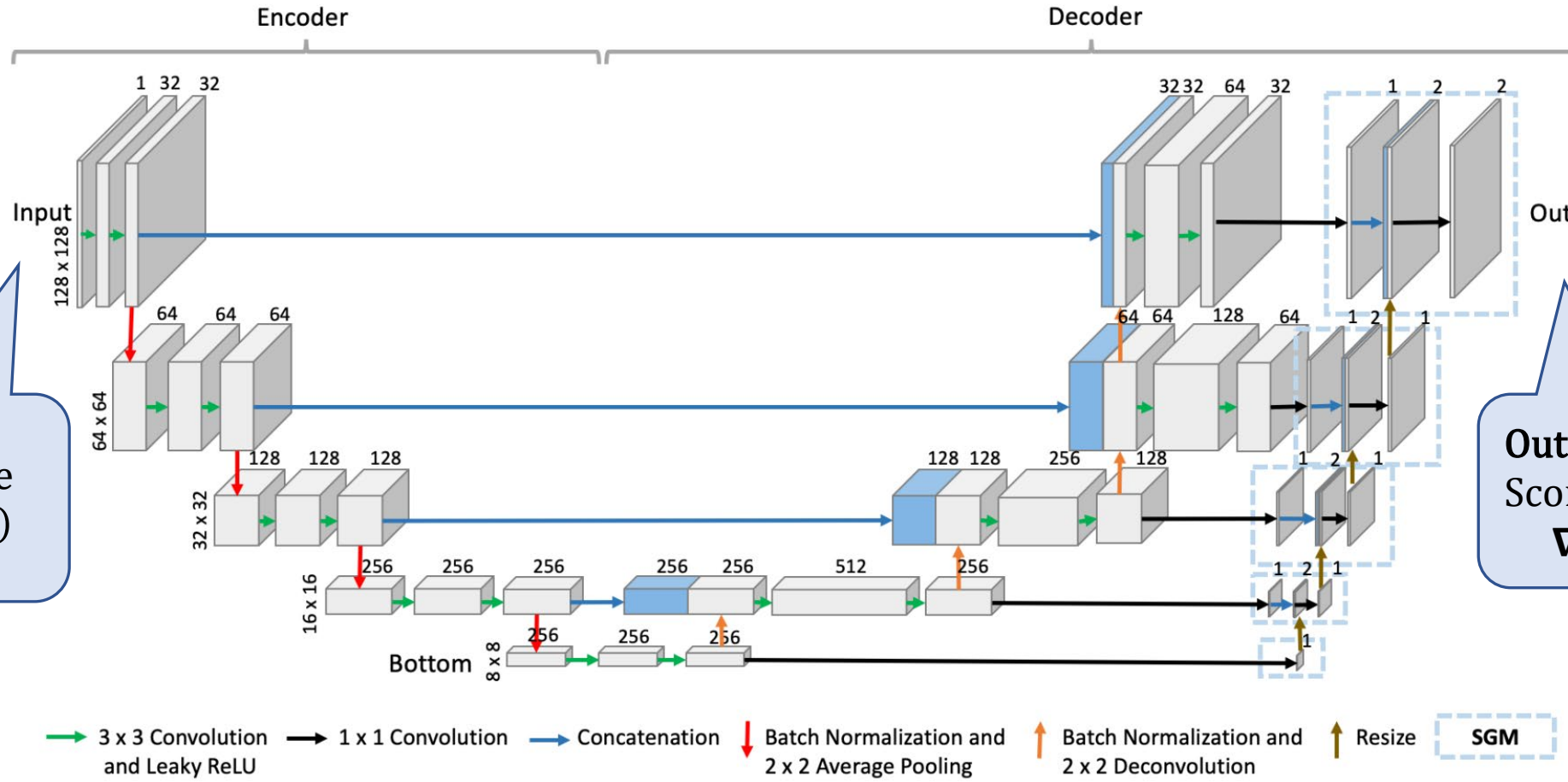
$$n = 0,1, \ldots, (N-1); \qquad m = 1, \ldots, M.$$

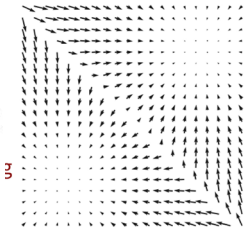
$\mathbf{x}_n^{(m)} \stackrel{\text{def}}{=} \mathbf{x}^{(m)}(t_n); \qquad \Delta T \stackrel{\text{def}}{=} \frac{T}{N} \longrightarrow 0^+; \left\{\mathbf{Z}_n^{(m)} \sim N(\mathbf{0}, \mathbf{I})\right\}$ iidrv's

Value of the mth trajectory at instant t_n

Approximating the score function: The U-Net

Input:
- Noisy image $\mathbf{x}(t) \sim p_t(\mathbf{x})$
- Time: t





Output:
Score function
$\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$

Image source: H. Pan, Q. Zhou and L. J. Latecki, "SGUNET: Semantic Guided UNET For Thyroid Nodule Segmentation," *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, Nice, France, 2021, pp. 630-634 https://doi.org/10.1109/ISBI48211.2021.9434051

Learning the score: Score matching

$\lambda(t)$: weighting function

Weighted combination of Fisher divergences

Score of $p_t(\mathbf{x})$

Unknown!

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim U[0,1]}\left[\lambda(t)\left[\mathbb{E}_{\mathbf{x}\sim p_t(\mathbf{x})}\left[\|\mathbf{s}(\mathbf{x},t;\boldsymbol{\theta}) - \nabla_{\mathbf{x}}\log p_t(\mathbf{x})\|_2^2\right]\right]\right]$$

$$= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim U[0,1]}\left[\lambda(t)\mathbb{E}_{\mathbf{x}_0\sim p_0(\mathbf{x})}\left[\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x},t|\mathbf{x}_0,0)}\left[\left\|\mathbf{s}(\mathbf{x},t;\boldsymbol{\theta}) - \nabla_{\mathbf{x}}\log p_{t|0}(\mathbf{x},t|\mathbf{x}_0,0)\right\|_2^2\right]\right]\right]$$

Weighted combination of denoising score-matching objectives

Score of $p_{t|0}(\mathbf{x},t|\mathbf{x}_0,0)$

Known!

P. Vincent, "A Connection Between Score Matching and Denoising Autoencoders,"
in *Neural Computation*, vol. 23, no. 7, pp. 1661-1674, July 2011, doi: 10.1162/NECO_a_00142.
https://www.iro.umontreal.ca/~vincentp/Publications/smdae_techreport.pdf

Loss function for a Gaussian diffusion process

A difusión process s.t. $p_{t|0}(\mathbf{x}(t), t|\mathbf{x}_0, 0)$ is Gaussian

- Initial condition: $\mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^D$
- Forward solution: $\mathbf{x}(t) \mid \mathbf{x}_0 = \boldsymbol{\mu}_{\mathbf{x}_0}(t) + \sigma(t)\,\mathbf{Z} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{x}_0}(t), \sigma^2(t)\,\mathbf{I}\right)$

$\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$p_{t|0}(\mathbf{x}(t), t|\mathbf{x}_0, 0) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2\sigma^2(t)}\left(\mathbf{x}(t) - \boldsymbol{\mu}_{\mathbf{x}_0}(t)\right)^{\mathrm{T}}\left(\mathbf{x}(t) - \boldsymbol{\mu}_{\mathbf{x}_0}(t)\right)\right)$$

$$\nabla_{\mathbf{x}} \log p_{t|0}(\mathbf{x}(t), t|\mathbf{x}_0, 0) = -\frac{1}{\sigma^2(t)}\left(\mathbf{x}(t) - \boldsymbol{\mu}_{\mathbf{x}_0}(t)\right) = -\frac{1}{\sigma(t)}\mathbf{Z}$$

$\mathbf{x}(t) - \boldsymbol{\mu}_{\mathbf{x}_0}(t) = \sigma(t)\,\mathbf{Z}$

$\mathbf{x}(t) \in \mathbb{R}^D$

$$p_t(\mathbf{x}) = \int p_{t|0}(\mathbf{x}, t|\mathbf{x}_0, 0) p_0(\mathbf{x}_0)\, d\mathbf{x}_0 \approx \frac{1}{N}\sum_{n=1}^{N} p_{t|0}(\mathbf{x}, t|\mathbf{x}_n, 0)$$

is a **mixture of Gaussians**!

$$loss(\boldsymbol{\theta}) = \mathbb{E}_{t\sim U[0,1]}\left[\lambda(t)\mathbb{E}_{\mathbf{x}_0\sim p_0(\mathbf{x})}\left[\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x},t|\mathbf{x}_0,0)}\left[\|\mathbf{s}(\mathbf{x}, t; \boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log p(\mathbf{x}, t|\mathbf{x}_0, 0)\|_2^2\right]\right]\right]$$

$$= \mathbb{E}_{t\sim U[0,1]}\left[\lambda(t)\mathbb{E}_{\mathbf{x}_0\sim p_0(\mathbf{x})}\left[\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x},t|\mathbf{x}_0,0)}\left[\left\|\mathbf{s}(\mathbf{x}, t; \boldsymbol{\theta}) + \frac{1}{\sigma(t)}\mathbf{Z}\right\|_2^2\right]\right]\right]$$

$$= \mathbb{E}_{t\sim U[0,1]}\left[\mathbb{E}_{\mathbf{x}_0\sim p_0(\mathbf{x})}\left[\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x}(t),t|\mathbf{x}_0,0)}\left[\frac{\lambda(t)}{\sigma^2(t)}\|\sigma(t)\mathbf{s}(\mathbf{x}, t; \boldsymbol{\theta}) + \mathbf{Z}\|_2^2\right]\right]\right]$$

$$\lambda(t) \propto \frac{1}{\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x}(t),t|\mathbf{x}_0,0)}\left[\left\|\nabla_{\mathbf{x}} \log p_{t|0}(\mathbf{x}, t|\mathbf{x}_0, 0)\right\|_2^2\right]} \propto \sigma^2(t)$$

Loss function: Empirical estimate for a Gaussian diffusion process

Training data: $\{\mathbf{x}_i\}_{i=1}^N$

- Initial condition: $\mathbf{x}_i(0) = \mathbf{x}_i \in \mathbb{R}^D;\ i = 1, \ldots, N$
- Forward solution: $\mathbf{x}_i(t)|\mathbf{x}_i(0) = \boldsymbol{\mu}_{\mathbf{x}_i}(t) + \sigma(t)\,\mathbf{Z} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{x}_i}(t), \sigma^2(t)\,\mathbf{I}\right) \xrightarrow[t\to\infty]{} \mathcal{N}(\mathbf{0}, \sigma^2(t)\,\mathbf{I})$

In the limit $t \to \infty$ $\boldsymbol{\mu}_{\mathbf{x}_i}(t) \ll \sigma(t)$

$$loss(\boldsymbol{\theta}) = \mathbb{E}_{t\sim U[0,1]}\left[\mathbb{E}_{\mathbf{x}_0\sim p_0(\mathbf{x})}\left[\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x},\,t|\mathbf{x}_0,0)}\left[\frac{\lambda(t)}{\sigma^2(t)}\left\|\sigma(t)\,\mathbf{s}(\mathbf{x},t;\boldsymbol{\theta}) + \mathbf{Z}\right\|_2^2\right]\right]\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\frac{\lambda(t_i)}{\sigma^2(t_i)}\left\|\sigma(t_i)\,\mathbf{s}(\mathbf{x}_i(t_i),t_i;\boldsymbol{\theta}) + \mathbf{Z}_i\right\|_2^2 \propto \frac{1}{N}\sum_{i=1}^{N}\left\|\sigma(t_i)\,\mathbf{s}(\mathbf{x}_i(t_i),t_i;\boldsymbol{\theta}) + \mathbf{Z}_i\right\|_2^2$$

Sample estimate:
$t_i \sim U[0,1]$
$\mathbf{Z}_i \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
$\mathbf{x}_i(t_i) = \boldsymbol{\mu}_{\mathbf{x}_i}(t_i) + \sigma(t_i)\mathbf{Z}_i$

$$\lambda(t) \propto \frac{1}{\mathbb{E}_{\mathbf{x}\sim p_{t|0}(\mathbf{x}(t),t|\mathbf{x}_0,0)}\left[\left\|\nabla_{\mathbf{x}}\log p_{t|0}(\mathbf{x},t|\mathbf{x}_0,0)\right\|_2^2\right]} \propto \sigma^2(t)$$

Flow matching

Connections with optimal transport:
Tong, A., Fatras, K., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Wolf, G., & Bengio, Y. (2024). Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 1-34.
https://openreview.net/forum?id=CD9Snc73AW
https://arxiv.org/abs/2302.00482

Learn a smooth differentiable map from $p_0(\mathbf{x})$ to $p_1(\mathbf{x})$

Neural ODE

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{v}(\mathbf{x}(t), t; \boldsymbol{\theta}), \qquad t \in [0, 1]$$

Initial condition: $\mathbf{x}_0 \sim p_0(\mathbf{x})$

Not necessarily Gaussian

$\mathrm{I}(\mathbf{x}_0, \mathbf{x}_1, t)$: Interpolant between $\mathbf{x}_0 \sim p_0(\mathbf{x})$ and $\mathbf{x}_1 \sim p_1(\mathbf{x})$

Velocity field

$$\mathbf{v}(\mathbf{x}(t), t; \boldsymbol{\theta}) \approx \frac{d}{dt}\mathrm{I}(\mathbf{x}_0, \mathbf{x}_1, t) = \frac{d}{dt}(\alpha(t)\mathbf{x}_0 + \beta(t)\mathbf{x}_1)$$

$\alpha(0) = 1;\ \alpha(1) = 0$ (e.g., $\alpha(t) = 1 - t$)
$\beta(0) = 0;\ \beta(1) = 1$ (e.g., $\beta(t) = t$)

- Generation: $\mathbf{x}_{new} = \int_0^1 \mathbf{v}(\mathbf{x}(t), t; \boldsymbol{\theta})dt \sim p_1(\mathbf{x})$
- Likelihood computation:

Divergence of the velocity field

$$\frac{\partial}{\partial t}p_t(\mathbf{x}) = -\boldsymbol{\nabla}_{\mathbf{x}} \cdot \left(p_t(\mathbf{x})\, \mathbf{v}(\mathbf{x}, t; \boldsymbol{\theta})\right) \Longrightarrow \frac{d}{dt}\log p_t\big(\mathbf{x}(t)\big) = -\boldsymbol{\nabla}_{\mathbf{x}} \cdot \mathbf{v}(\mathbf{x}(t), t; \boldsymbol{\theta})$$

Liouville (continuity transport) equation

$$\Longrightarrow\ p_1(\mathbf{x}) \approx p_0(\mathbf{x}) \exp\left(-\int_0^1 \boldsymbol{\nabla}_{\mathbf{x}} \cdot \mathbf{v}(\mathbf{x}(t), t; \boldsymbol{\theta})dt\right)$$

Reverse SDE vs probability flow ODE

Generally faster.
Can be used to compute likelihoods (determinitic, invertible mapping)

$\mathbf{s}(\overleftarrow{\mathbf{x}}(t), t) = \nabla_{\mathbf{x}} \log p_t(\overleftarrow{\mathbf{x}}(t))$

- Probability flow ODE: $d\overleftarrow{\mathbf{x}}(t) = \left(\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - \frac{1}{2} g^2(t)\, \mathbf{s}(\overleftarrow{\mathbf{x}}(t), t)\right) dt$
- Reverse SDE:

$$d\overleftarrow{\mathbf{x}}(t) = \left(\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - g^2(t)\, \nabla_{\mathbf{x}} \log p_t(\overleftarrow{\mathbf{x}}(t))\right) dt + g(t) d\mathbf{W}(t)$$

$$= \left(\mathbf{f}(\overleftarrow{\mathbf{x}}(t), t) - \frac{1}{2} g^2(t)\, \mathbf{s}(\overleftarrow{\mathbf{x}}(t), t)\right) dt + \left(-\frac{1}{2} g^2(t) \mathbf{s}(\overleftarrow{\mathbf{x}}(t), t) dt + g(t) d\mathbf{W}(t)\right)$$

Slower but, typically, better quality samples

Probability flow ODE

Langevin dynamics

Combined scheme

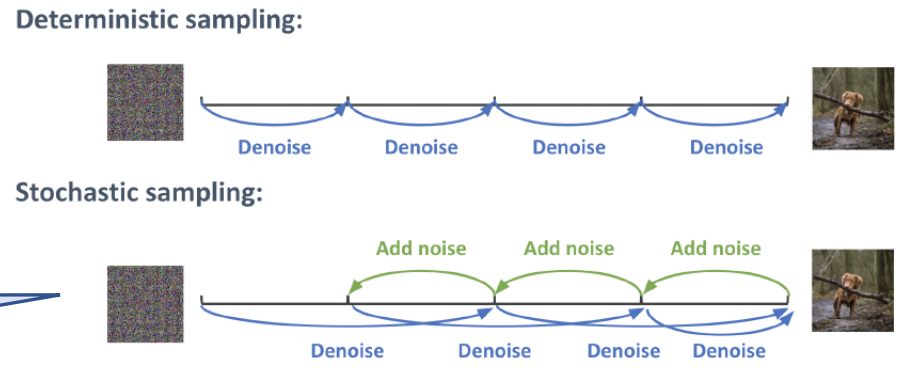


Figure 25.9: Generatng from the reverse diffusion process using 4 steps. (Top) Deterministic sampling. (Bottom) A mix of deterministic and stochastic sampling. Used with kind permission of Ruiqi Gao.

Diffusion in latent space: VAE + Diffusion model

Latent space provides

- **Compressed representation:**
  - dimensionality reduction
  - simpler networks
  - more efficient training and generation.
- **High-level semantic structure:**
  - More **coherent and meaningful outputs.**
  - More robust to noise
  - **Simpler conditioning:** Conditioning (text, style, etc.) is easier to integrate with **structured latent features.**

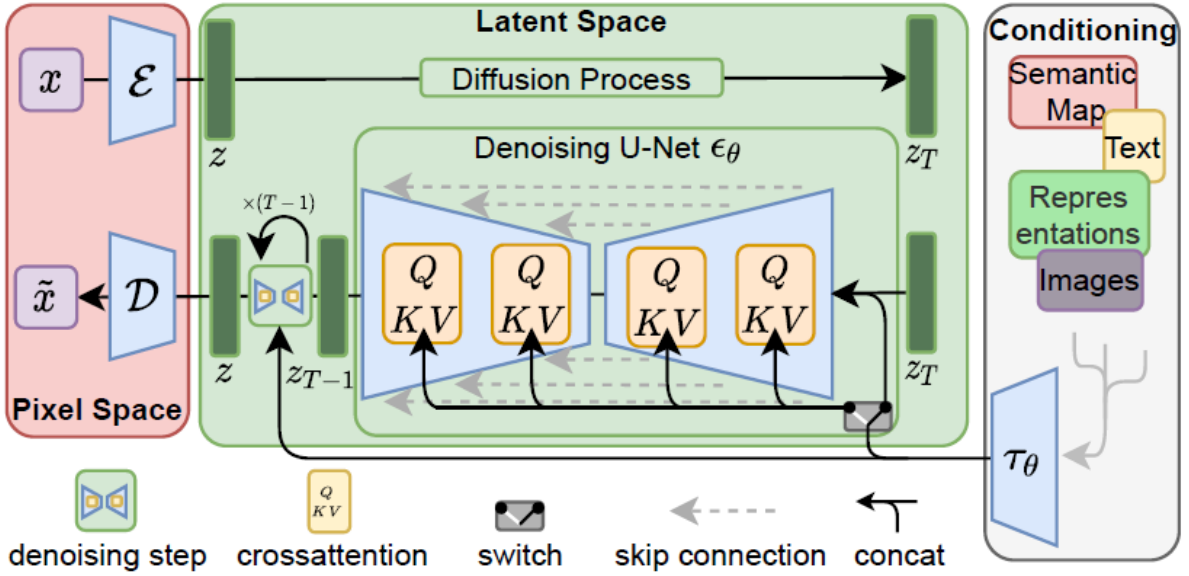


Figure 25.13: Combining a VAE with a diffusion model. Here $\mathcal{E}$ and $\mathcal{D}$ are the encoder and decoder of the VAE. The diffusion model conditions on the inputs either by using concatentation or by using a cross-attention mechanism. From Figure 3 of [Rom+22]. Used with kind permission of Robin Rombach.

Conditional generation: Conditional diffusion model

sample $\mathcal{D} = \{\mathbf{x}_i, c_i\}_{i=1}^{N}$ $\mathbf{x}_i \in \mathbb{R}^D$, $c_i \in \mathcal{C}$ (additional information)

- A class label
- An image
- A text prompt

- Conditional diffusion model:

  Train on $\mathcal{D}$ to maximize the conditional likelihood $p_0(\mathbf{x}|c)$.
  - c is a scalar (e.g., a class label): Map it to an embedding vector
    - Incorporate it into the network by spatial addition
    - Use to modulate normalization layers
  - c is an image: Concatenate with $\mathbf{x}$ as additional channels
  - c is a text prompt: Map it to an embedding vector
    - Incorporate it into the network by spatial addition
    - Use cross-attention

Conditional generation: Classifier guidance

P. Dhariwal A. Q. Nichol. *Diffusion Models Beat GANs on Image Synthesis* In: NIPS. May 2021.

Sample $\mathcal{D} = \{\mathbf{x}_i, c_i\}_{i=1}^N \qquad \mathbf{x}_i \in \mathbb{R}^D, \quad c_i \in \mathcal{C}$

- Leverage pre-trained discriminative classifier of the form $p_t(c|\mathbf{x};\boldsymbol{\phi})$
- Objective function

Different classifiers for different t (noise level)

$$\log p_t(\mathbf{x}|c;\boldsymbol{\theta},\boldsymbol{\phi}) = \log p_t(c|\mathbf{x},\boldsymbol{\phi}) + \log p_t(\mathbf{x};\boldsymbol{\theta}) - \log p(c)$$

  - Score function

Models for $p_t(c|\mathbf{x},\boldsymbol{\phi})$ and $p_t(\mathbf{x};\boldsymbol{\theta})$ and learned separately

$$s(\mathbf{x}|c;\boldsymbol{\theta},\boldsymbol{\phi}) = \nabla_{\mathbf{x}} \log p(\mathbf{x};\boldsymbol{\theta}) + \nabla_{\mathbf{x}} \log p(c|\mathbf{x},\boldsymbol{\phi})$$

  - Weighted score function

$w > 0$: Amplification of the conditioning signal ($w = 10$ in [DN21b])

$$s_w(\mathbf{x}|c;\boldsymbol{\theta},\boldsymbol{\phi}) = \nabla_{\mathbf{x}} \log p(\mathbf{x};\boldsymbol{\theta}) + w\,\nabla_{\mathbf{x}} \log p(c|\mathbf{x},\boldsymbol{\phi})$$

Could lead to poor results because of contraposed objectives (similar problem to adversarial methods)

Sampling scheme:

$$\mathbf{x}_{t-1} \sim \mathcal{N}\left(\mathbf{x}_t \middle| \boldsymbol{\mu}(\mathbf{x}_t, t;\boldsymbol{\theta}) + w\,\boldsymbol{\Sigma}(\mathbf{x}_t, t;\boldsymbol{\theta})\nabla_{\mathbf{x}_t} \log p_t(c|\mathbf{x}_t,\boldsymbol{\phi}), \boldsymbol{\Sigma}(\mathbf{x}_t, t;\boldsymbol{\theta})\right)$$

Conditional generation: Classifier-free guidance

J. Ho and T. Salimans (2021)
Classifier-Free Diffusion Guidance
In: NIPS Workshop on Deep Generative Models and Downstream Applications

- Learn two generative models
  - Uncoditional: $p_t(\mathbf{x};\boldsymbol{\theta})$
  - Conditional: $p_t(\mathbf{x}|c,\boldsymbol{\phi})$

Unconditional classifier could simply be $p_t(\mathbf{x}|c=\emptyset,\boldsymbol{\phi})$

- Build implicit classifier from the two generative models

$$\log p_t(c|\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi}) = \log p_t(\mathbf{x}|c,\boldsymbol{\phi}) - \log p_t(\mathbf{x};\boldsymbol{\theta}) + \log p(c)$$

- Use weighted score function

$$s_w(\mathbf{x}|c;\boldsymbol{\theta},\boldsymbol{\phi}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}|c,\boldsymbol{\phi}) + w\,\nabla_{\mathbf{x}} \log p_t(c|\mathbf{x},\boldsymbol{\theta},\boldsymbol{\phi})$$

$$= (1+w)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|c,\boldsymbol{\phi}) - w\,\nabla_{\mathbf{x}} \log p_t(\mathbf{x};\boldsymbol{\theta})$$

Typically, better individual sample quality, but lower diversity

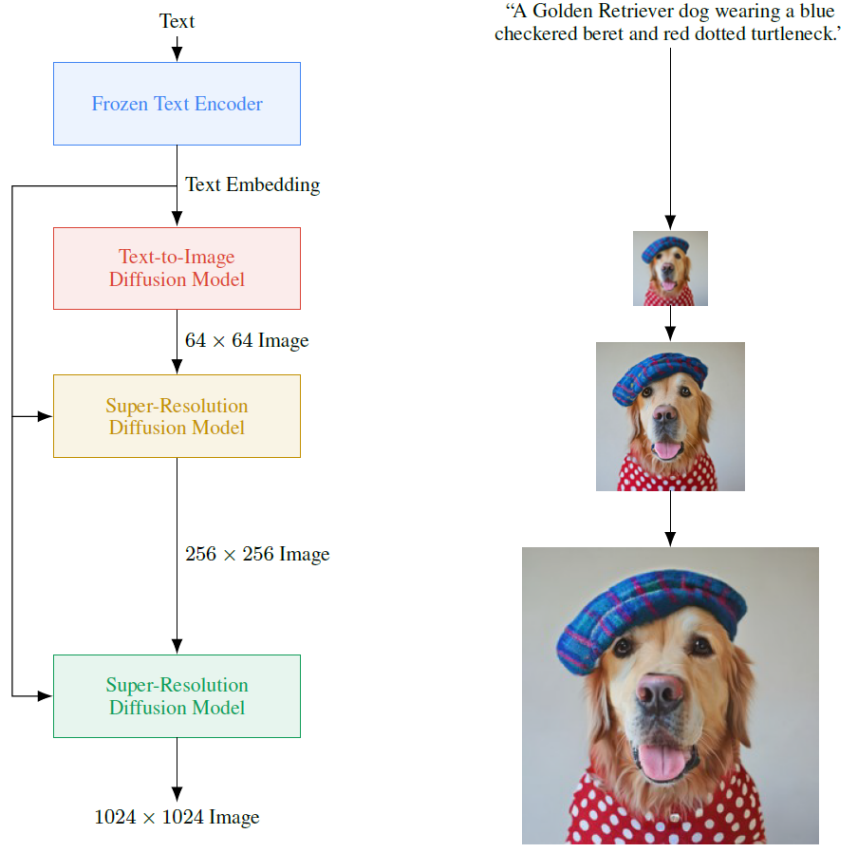
Conditional vs. unconditional generation

(a) (b) (c)

Figure 25.4: Some sample images generated by a small variational diffusion model trained on EMNIST for about 30 minutes on a K40 GPU. (a) Unconditional sampling. (b) Conditioned on class label. (c) Using classifier-free guidance (see Section 25.6.3). Generated by diffusion_emnist.ipynb. Used with kind permission of Alex Alemi.

Generation of higher resolution images

Google's **Imagen** model
C. Saharia et al. (2022)
Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding
https://arxiv.org/abs/2205.11487

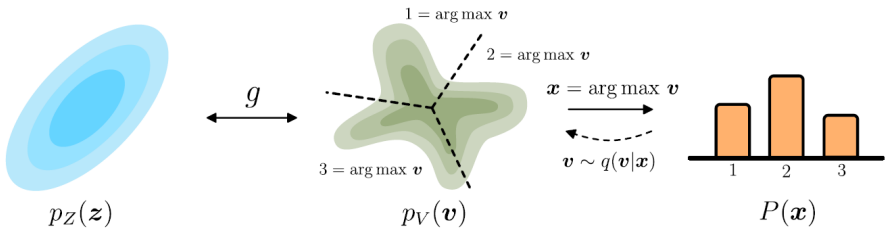


Cascade generation [Ho+21]:
1. Text to low resolution image diffusion model.
2. Super-resolution diffusion model.

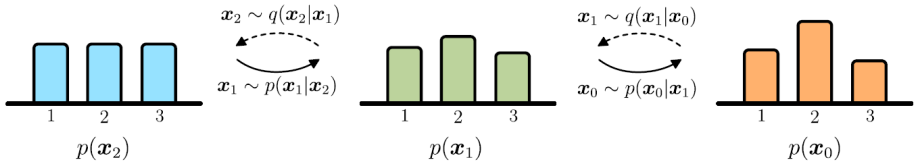
Figure A.4: Visualization of Imagen. Imagen uses a frozen text encoder to encode the input text into text embeddings. A conditional diffusion model maps the text embedding into a 64×64 image. Imagen further utilizes text-conditional super-resolution diffusion models to upsample the image, first $64 \times 64 \rightarrow 256 \times 256$, and then $256 \times 256 \rightarrow 1024 \times 1024$.

Discrete data: Multinomial diffusion

E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling. *Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions*. In: NIPS. Feb. 2021. https://arxiv.org/pdf/2102.05379



(a) Argmax Flow: Composition of a flow $p(\boldsymbol{v})$ and argmax transformation which gives the model $P(\boldsymbol{x})$. The flow maps from a base distribution $p(\boldsymbol{z})$ using a bijection g.



(b) Multinomial Diffusion: Each step $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ denoises the signal starting from a uniform categorical base distribution which gives the model $p(\boldsymbol{x}_0)$.

Figure 1: Overview of generative models.

Multinomial diffusion for image segmentation

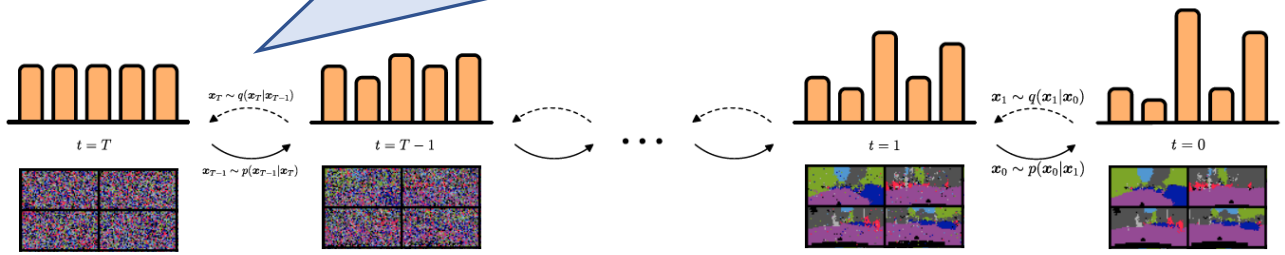


Figure 2: Overview of multinomial diffusion. A generative model $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ learns to gradually denoise a signal from left to right. An inference diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ gradually adds noise form right to left.

Discrete denoising diffusion probabilistic model (D3PM)

Diffusion model designed for discrete data (e.g., text).

1. Inject noise in the original examples through a sequence of probabilistic transitions (such as masking or random token replacement) until a simple limiting distribution is obtained.

2. Use a trained neural network to reverse this process in a stepwise manner.

3. The network is typically trained using cross-entropy losses.

This process allows generation of discrete structures in a non-autoregressive and globally coherent way.

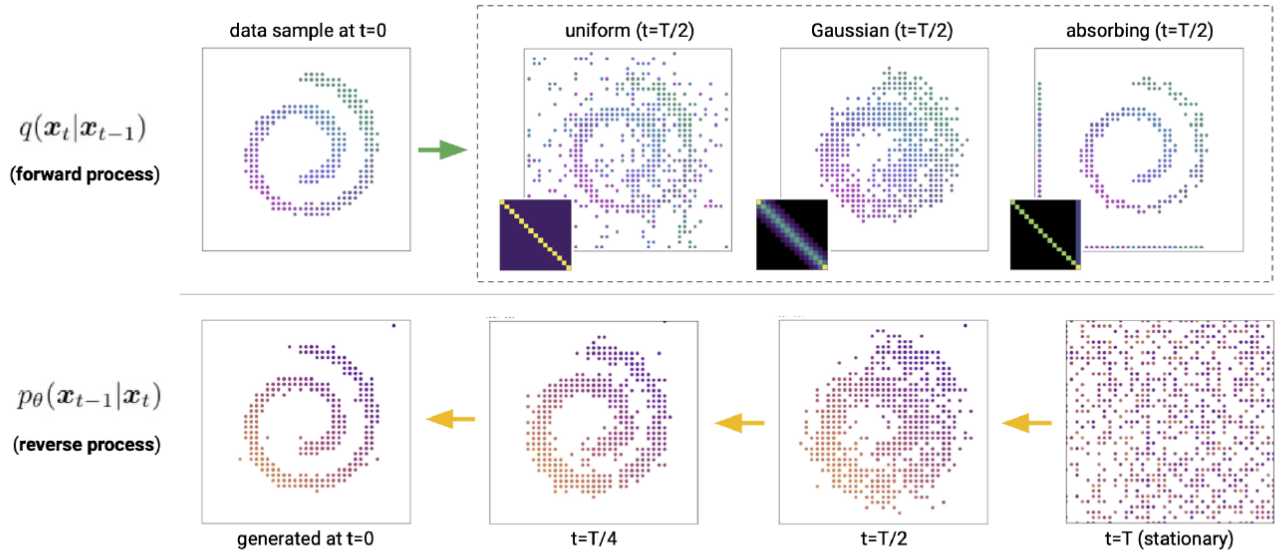


Figure 1: D3PM forward and (learned) reverse process applied to a quantized swiss roll. Each dot represents a 2D categorical variable. Top: samples from the uniform, discretized Gaussian, and absorbing state D3PM model forward processes, along with corresponding transition matrices $\boldsymbol{Q}$. Bottom: samples from a learned discretized Gaussian reverse process.

J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. *Structured Denoising Diffusion Models in Discrete State-Spaces.* In: NIPS. July 2021. https://arxiv.org/abs/2107.03006

Generative AI: diffusion models

Diffusion models are a family of **generative** models based on **reversing a noise-injection process to** produce **samples whose characteristics are similar to those of the training data**.

Basic idea: **Break up the generative process** into **small denoising steps that produce images with a lower level of noise from noisier images**.

- **Training** consists in **learning to remove noise from perturbed images created by injecting noise into the original images**.
- **Generation. To generate a new sample:**
  - **Sample from the limiting distribution of the forward process**.
  - Remove noise in a **stepwise** procedure corresponding to the **time-reversal of the noise-injection process**.

Leveraging this strategy, diffusion models can be used to **generate high-quality, diverse samples of structured data**.

The **conceptual simplicity** of diffusion models, their **scalability**, and the **stability of their training process** have contributed to their widespread adoption in the **synthesis of audio, images, video, and other forms of complex data**.

Online resources

- Kevin P. Murphy (2023), *Probabilistic Machine Learning: Advanced Topics* (version online 2025-Dec-10) MIT Press
  http://probml.github.io/book2
- Lilian Weng. (July, 2021). What are diffusion models? Lil'Log.
  https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- Yang Song (May, 2021) *Generative Modeling by Estimating Gradients of the Data Distribution*.
  https://yang-song.net/blog/2021/score/
  https://colab.research.google.com/drive/120kYYBOVa1i0TD85RjlEkFjaWDxSFUx3?usp=sharing
- Karsten Kreis, Ruiqi Gao, Arash Vahdat (2022)
  https://cvpr2022-tutorial-diffusion-models.github.io/
- Paper Collection on Diffusion Models
  https://github.com/zelaki/Reading-Diffusion

Thanks for your attention!

alberto.suarez@uam.es