

Generative models for images II

Bruno Galerne

`bruno.galerie@univ-orleans.fr`

ETICS 2024: École Thématique sur les Incertitudes en Calcul Scientifique

Saissac

Institut Denis Poisson

Université d'Orléans, Université de Tours, CNRS

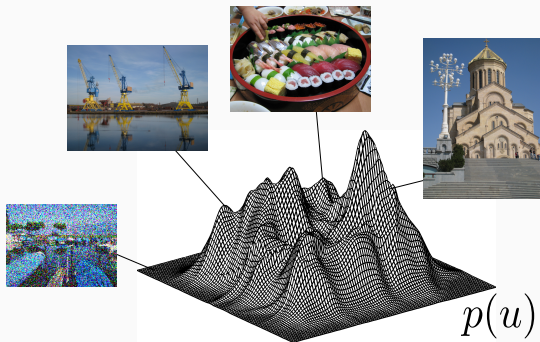
Institut universitaire de France (IUF)

Material for the course is here:

<https://www.idpoisson.fr/galerie/etics2024/index.html>

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.



(source: Charles Deledalle)

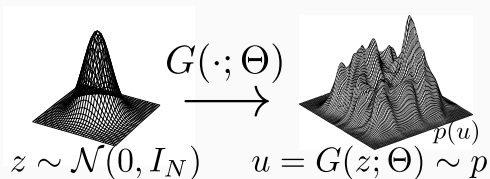
The images may represent:

- different instances of the same texture image,
- all images naturally described by a dataset of images,
- any image

2. Generate samples from this distribution.

Generative models

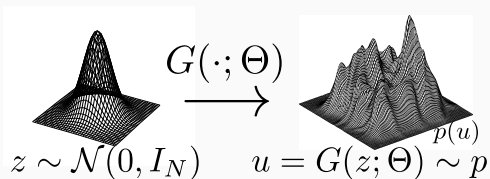
1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot; \Theta)$ is known, then $p = G(\cdot; \Theta)_\# \mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot; \Theta)$ is known, then $p = G(\cdot; \Theta)_{\#} \mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

The generator $G(\cdot; \Theta)$ can be:

- A deterministic function (e.g. convolution operator),
- A neural network with learned parameter,
- An iterative optimization algorithm (gradient descent,...),
- A stochastic sampling algorithm (e.g. MCMC, Langevin diffusion,...).

Basics on diffusion models

Adding noise to images

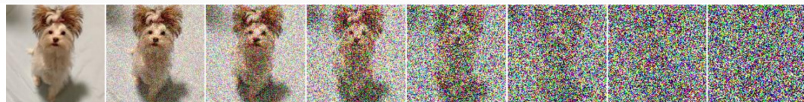
- We are given an input dataset

$$\mathcal{D} = \{\mathbf{x}^{(i)}, i = 1, \dots, N\} \subset \mathbb{R}^d$$

- We assume that these images are independent samples of a common distribution p_0 over \mathbb{R}^d .
- Consider the random process that consists of adding noise to images:

$$\mathbf{x}_t = \mathbf{x}_0 + \mathbf{w}_t, \quad t \in [0, T]$$

where $\mathbf{x}_0 \sim p_0$ is a sample image and \mathbf{w}_t is a Brownian motion (also called Wiener process).



(source: [\(Song et al., 2021b\)](#))

Real-valued: A standard (real-valued) **Brownian motion** (also called **Wiener process**) is a stochastic process $(w_t)_{t \geq 0}$ such that

- $w_0 = 0$.
- With probability one, the function $t \mapsto w_t$ is continuous.
- The process $(w_t)_{t \geq 0}$ has stationary independent increments.
- $w_t \sim \mathcal{N}(0, t)$.

Direct consequences:

- For $s < t$, w_s and $w_t - w_s$ are independent and $w_{t-s} \sim \mathcal{N}(0, t - s)$.
- Markovian random field.

\mathbb{R}^d -valued: A standard \mathbb{R}^d -valued Brownian motion $(\mathbf{w}_t)_{t \geq 0}$ is made of d independent real-valued Brownian motions

$$\mathbf{w}_t = (w_{t,1}, \dots, w_{t,d}) \in \mathbb{R}^d.$$

Ito integral on $[0, T]$:

Given a process $(\mathbf{x}_t)_{t \in [0, T]}$ adapted to the filtration $\mathcal{F}_t = \sigma(\mathbf{w}_s, s \leq t)$, one defines

$$\int_0^t \mathbf{x}_s d\mathbf{w}_s \quad \text{as the } L^2 \text{ limit of } \sum_{j=0}^{k-1} \mathbf{x}_{t_j} \odot (\mathbf{w}_{t_{j+1}} - \mathbf{w}_{t_j})$$

when the minimal step of the partition $0 \leq t_0 \leq \dots \leq t_k \leq T$ tends to 0.

- In particular, for a deterministic function $s \mapsto g(s)$, $\int_0^t g(s) d\mathbf{w}_s$ is a normal variable with mean 0 and variance $\sigma^2 = \int_0^t g^2(s) ds$.

- Adding noise to images: $\mathbf{x}_t = \mathbf{x}_0 + \mathbf{w}_t$, $t \in [0, T]$.
- This corresponds to the stochastic differential equation (SDE):

$$d\mathbf{x}_t = d\mathbf{w}_t \quad \text{with initial condition } \mathbf{x}_0 \sim p_0.$$

- We denote by p_t the distribution of \mathbf{x}_t at time $t \in [0, T]$. What is p_t ?

$$p_t = p_0 * \mathcal{N}(\mathbf{0}, tI_d)$$

- This corresponds to applying the heat equation starting from p_0 :

$$\partial_t p_t(\mathbf{x}) = \frac{1}{2} \Delta_{\mathbf{x}} p_t(\mathbf{x}) \quad \text{with } p_{t=0} = p_0.$$

This PDE is called the **Fokker-Planck equation** associated with the SDE.

- This is an example of diffusion equation.

Diffusion SDE and Fokker-Planck equation

- More generally we will consider diffusion SDE of the form (Song et al., 2021b):

$$dx_t = f(x_t, t)dt + g(t)dw_t$$

where

- $f : \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^d$ is called the **drift**: External deterministic force that drives x_t in the direction $f(x_t, t)$,
 - $g : [0, T] \rightarrow [0, +\infty)$ is the **diffusion coefficient**.
- The corresponding Fokker-Planck equation is

$$\partial_t p_t(\mathbf{x}) = -\operatorname{div}_{\mathbf{x}} (f(\mathbf{x}, t)p_t(\mathbf{x})) + \frac{1}{2}g(t)^2 \Delta_{\mathbf{x}} p_t(\mathbf{x})$$

that is,

$$\partial_t p_t(\mathbf{x}) = -\sum_{k=1}^d \partial_{x_k} [f_k(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}g(t)^2 \sum_{k=1}^d \partial_{x_k}^2 p_t(\mathbf{x}).$$

$$dx_t = f(x_t, t)dt + g(t)dw_t$$

Example 1: Variance exploding diffusion (VE-SDE)

SDE: $dx_t = dw_t$

Solution: $x_t = x_0 + w_t$

Variance: $\text{Var}(x_t) = \text{Var}(x_0) + t$

Example 2: Variance preserving diffusion (VP-SDE)

SDE: $dx_t = -\beta_t x_t dt + \sqrt{2\beta_t} dw_t$

Solution: $x_t = e^{-B_t} x_0 + \int_0^t e^{B_s - B_t} \sqrt{2\beta_s} dw_s$ with $B_t = \int_0^t \beta_s ds$

Variance: $\text{Var}(x_t) = e^{-2B_t} \text{Var}(x_0) + 1 - e^{-2B_t} = 1$ if $\text{Var}(x_0) = 1$.

Both variants have the form $x_t = a_t x_0 + b_t Z_t$: x_t is a rescaled noisy version of x_0 and the noise is more and more predominant as time grows.

$$dx_t = f(x_t, t)dt + g(t)d\mathbf{w}_t$$

In general we do not have a close form formula for x_t .

Diffusion SDEs can be approximately simulated using numerical schemes such as the **Euler-Maruyama scheme**:

- Using the time step $h = T/N$ with $N + 1$ times $t_n = nh$, $n \in \{0, \dots, N\}$, define $X_0 = x_0$ and

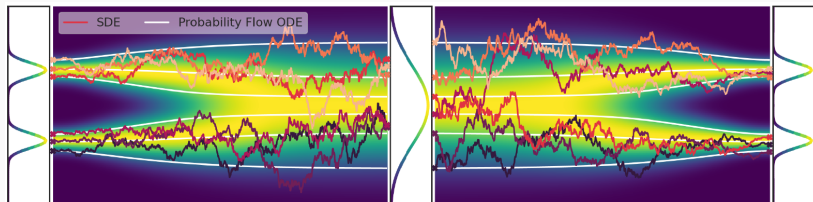
$$X_{n+1} = X_n + f(X_n, t_n)h + g(t_n) (\mathbf{w}_{t_{n+1}} - \mathbf{w}_{t_n}), \quad n = 1, \dots, N - 1.$$

- Remark that $\mathbf{w}_{t_{n+1}} - \mathbf{w}_{t_n} \sim \mathcal{N}(\mathbf{0}, hI_d)$ and is independent of X_n :

$$X_{n+1} = X_n + f(X_n, t_n)h + g(t_n)\sqrt{h}\mathbf{Z}_n, \quad \text{with } \mathbf{Z}_n \sim \mathcal{N}(\mathbf{0}, I_d), \quad n = 1, \dots, N - 1.$$

Reversed diffusion

- For diffusion SDEs, as t grows p_t is closer and closer to a normal distribution.
- We will consider that at the final time $t = T$ large enough so that p_T can be considered to be a normal distribution.
- For generative modeling, we want to reverse the process:
 - Start by generating $\mathbf{x}_T \sim p_T \approx \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I}_d)$.
 - Simulate $(\mathbf{x}_{T-t})_{t \in [0, T]}$ such that $\mathbf{x}_{T-t} \sim p_{T-t}$.



(source: (Song and Ermon, 2020))

Reversed diffusion: What is the SDE satisfied by \mathbf{x}_{T-t} ?

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

has the associated Fokker-Planck equation

$$\partial_t p_t(\mathbf{x}) = -\operatorname{div}_x (\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})) + \frac{1}{2}g(t)^2 \Delta_x p_t(\mathbf{x}).$$

Let us derive the Fokker-Planck equation for $q_t = p_{T-t}$ the distribution function of $\mathbf{y}_t = \mathbf{x}_{T-t}$.

$$\begin{aligned}\partial_t q_t(\mathbf{x}) &= -\partial_t p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)p_{T-t}(\mathbf{x})) - \frac{1}{2}g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)q_t(\mathbf{x})) - \frac{1}{2}g(T-t)^2 \Delta_x q_t(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)q_t(\mathbf{x})) + \left(-1 + \frac{1}{2}\right)g(T-t)^2 \Delta_x q_t(\mathbf{x})\end{aligned}$$

$$\begin{aligned}\partial_t q_t(\mathbf{x}) &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)q_t(\mathbf{x})) + \left(-1 + \frac{1}{2}\right)g(T-t)^2\Delta_x q_t(\mathbf{x}) \\ &= \operatorname{div}_x \left(\mathbf{f}(\mathbf{x}, T-t)q_t(\mathbf{x}) - g(T-t)^2\nabla_x q_t(\mathbf{x})\right) + \frac{1}{2}g(T-t)^2\Delta_x q_t(\mathbf{x}) \\ &= \operatorname{div}_x \left(\left[\mathbf{f}(\mathbf{x}, T-t) - g(T-t)^2\frac{\nabla_x q_t(\mathbf{x})}{q_t(\mathbf{x})}\right]q_t(\mathbf{x})\right) + \frac{1}{2}g(T-t)^2\Delta_x q_t(\mathbf{x}) \\ &= -\operatorname{div}_x \left(\left[-\mathbf{f}(\mathbf{x}, T-t) + g(T-t)^2\nabla_x \log q_t(\mathbf{x})\right]q_t(\mathbf{x})\right) + \frac{1}{2}g(T-t)^2\Delta_x q_t(\mathbf{x})\end{aligned}$$

This is the Fokker-Planck equation associated with the diffusion SDE:

$$dy_t = \left[-\mathbf{f}(\mathbf{y}_t, T-t) + g(T-t)^2\nabla_x \log p_{T-t}(\mathbf{y}_t)\right] dt + g(T-t)d\mathbf{w}_t.$$

Reversed diffusion

Forward diffusion:

$$dx_t = f(x_t, t)dt + g(t)dw_t$$

Backward diffusion: $y_t = x_{T-t}$

$$dy_t = \left[-f(y_t, T-t) + g(T-t)^2 \nabla_x \log p_{T-t}(y_t) \right] dt + g(T-t)dw_t.$$

- Same diffusion coefficient.
- Opposite drift term with additional distribution correction:

$$g(T-t)^2 \nabla_x \log p_{T-t}(y_t)$$

drives the diffusion in regions with high p_{T-t} probability.

- $x \mapsto \nabla_x \log p_t(x)$ is called the (Stein) **score** of the distribution.
- Rigorous results from SDE literature ((Anderson, 1982) (Hausmann and Pardoux, 1986)) (measurability issues, the filtration is also reversed...).

Reversed diffusion

Forward diffusion:

$$dx_t = f(x_t, t)dt + g(t)dw_t$$

Backward diffusion: $y_t = x_{T-t}$

$$dy_t = \left[-f(y_t, T-t) + g(T-t)^2 \nabla_x \log p_{T-t}(y_t) \right] dt + g(T-t)dw_t.$$

- Same diffusion coefficient.
- Opposite drift term with additional distribution correction:

$$g(T-t)^2 \nabla_x \log p_{T-t}(y_t)$$

drives the diffusion in regions with high p_{T-t} probability.

- $x \mapsto \nabla_x \log p_t(x)$ is called the (Stein) **score** of the distribution.
- Rigorous results from SDE literature ((Anderson, 1982) (Haussmann and Pardoux, 1986)) (measurability issues, the filtration is also reversed...).
- Can we simulate this backward diffusion using Euler-Maruyama ?

$$X_{n+1} = X_n + f(X_n, t_n)h + g(t_n)\sqrt{h}Z_n, \quad \text{with } Z_n \sim \mathcal{N}(\mathbf{0}, I_d), \quad n = 1, \dots, N-1.$$

Learning the score function: Denoising score matching

- **Goal:** Estimate the score $\mathbf{x} \mapsto \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ using only available samples $(\mathbf{x}_0, \mathbf{x}_t)$.
- For the models of interests, $\mathbf{x}_t = a_t \mathbf{x}_0 + b_t \mathbf{Z}_t$ is a rescaled noisy version of \mathbf{x}_0 (both a_t and b_t have known analytical expressions).
- Explicit conditional distribution: $p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(a_t \mathbf{x}_0, b_t^2 I_d)$.

$$\begin{aligned} p_t(\mathbf{x}_t) &= \int_{\mathbb{R}^d} p_{0,t}(\mathbf{x}_0, \mathbf{x}_t) d\mathbf{x}_0 = \int_{\mathbb{R}^d} p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t) &= \int_{\mathbb{R}^d} \nabla_{\mathbf{x}_t} p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) &= \frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t)}{p_t(\mathbf{x}_t)} = \int_{\mathbb{R}^d} \nabla_{\mathbf{x}_t} p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) \frac{p_0(\mathbf{x}_0)}{p_t(\mathbf{x}_t)} d\mathbf{x}_0 \\ &= \int_{\mathbb{R}^d} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)] p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) \frac{p_0(\mathbf{x}_0)}{p_t(\mathbf{x}_t)} d\mathbf{x}_0 \\ &= \int_{\mathbb{R}^d} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)] p_{0|t}(\mathbf{x}_0|\mathbf{x}_t) d\mathbf{x}_0 \end{aligned}$$

Conclusion:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_0 \sim p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)] = \mathbb{E} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) | \mathbf{x}_t]$$

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_0 \sim p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)] = \mathbb{E} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)|\mathbf{x}_t]$$

- $\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$ is explicit (forward transition): For $\mathbf{x}_t|\mathbf{x}_0 \sim \mathcal{N}(\alpha_t \mathbf{x}_0, \beta_t^2 I_d)$,

$$\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = \nabla_{\mathbf{x}_t} \left[-\frac{1}{2\beta_t^2} \|\mathbf{x}_t - \alpha_t \mathbf{x}_0\|^2 + C \right] = -\frac{1}{\beta_t^2} (\mathbf{x}_t - \alpha_t \mathbf{x}_0) = -\frac{1}{\beta_t} \mathbf{Z}_t$$

- But the distribution $p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)$ is not explicit (backward conditional)!

$$\mathbb{E} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)|\mathbf{x}_t] = -\frac{1}{\beta_t^2} (\mathbf{x}_t - \alpha_t \mathbb{E}[\mathbf{x}_0|\mathbf{x}_t])$$

- $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t]$ is the best estimate of the initial noise-free \mathbf{x}_0 given its noisy version \mathbf{x}_t .

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_0 \sim p_{0|t}(\mathbf{x}_0|\mathbf{x}_t)} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)] = \mathbb{E} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)|\mathbf{x}_t]$$

We use the following properties of the **conditional expectation**.

- $Y = \mathbb{E}[X|\mathcal{F}]$ if and only if $Y = \operatorname{argmin}\{\mathbb{E}\|X - Z\|^2, Z \in L^2(\mathcal{F})\}$.
- $Y \in \sigma(X)$ iff there exists $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ (measurable) with $Y = f(X)$.
- $Y = \mathbb{E}[X|U]$ if $Y = f(U)$ with $f = \operatorname{argmin}\{\mathbb{E}\|X - f(U)\|^2, f \in L^2(U)\}$.

Hence the function $\mathbf{x}_t \mapsto \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the solution

$$\nabla_{\mathbf{x}_t} \log p_t = \operatorname{argmin}\{\mathbb{E}_{p_{0,t}} \|f(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\|^2, f \in L^2(p_t)\}$$

- We obtain a **loss function** to learn the function f using Monte Carlo approximation with samples $(\mathbf{x}_0, \mathbf{x}_t)$ for the expectation.

Learning the score function: Denoising score matching

$$\nabla_{\mathbf{x}_t} \log p_t = \operatorname{argmin} \{ \mathbb{E}_{p_{0,t}} \|f(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\|^2, f \in L^2(p_t) \}$$

- $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ will be approximated with a neural network such as a (complex) U-net (Ho et al., 2020).
- But we need to have an approximation of $\nabla_{\mathbf{x}_t} \log p_t$ for all time t (at least for the times t_n in our Euler-Maruyama scheme).
- In practice we share the same network architecture for all time t : one learns a network $s_\theta(\mathbf{x}, t)$ such that

$$s_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, t \in [0, T].$$

Final loss for denoising score matching: (Song et al., 2021b)

$$\theta^* = \operatorname{argmin} \mathbb{E}_t \left(\lambda_t \mathbb{E}_{(x_0, x_t)} \|s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\|^2 \right)$$

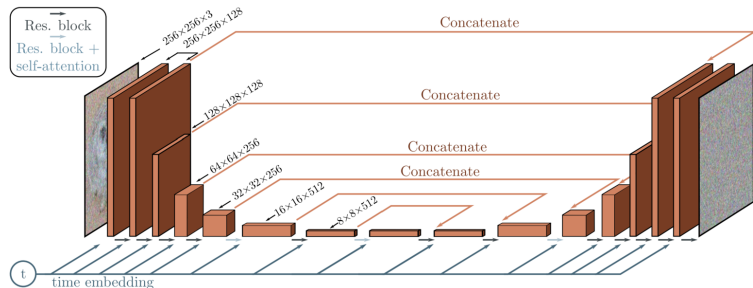
where t is chosen uniformly in $[0, T]$ and $t \mapsto \lambda_t$ is a weighting term to balance the importance of each t .

Practical aspects of diffusion models: Training and sampling

Score architecture

$$\theta^* = \operatorname{argmin} \mathbb{E} \left(\lambda_t \mathbb{E}_{(x_0, x_t)} \|s_\theta(x_t, t) - \nabla_{x_t} \log p_{t|0}(x_t|x_0)\|^2 \right)$$

- $s_\theta : \mathbb{R}^d \times [0, T] \rightarrow \mathbb{R}^d$ is a (complex) U-net (Ronneberger et al., 2015), eg in (Ho et al., 2020) “All models have two convolutional residual blocks per resolution level and self-attention blocks at the 16×16 resolution between the convolutional blocks”.
- Diffusion time t is specified by adding the Transformer sinusoidal position embedding into each residual block (Vaswani et al., 2017).



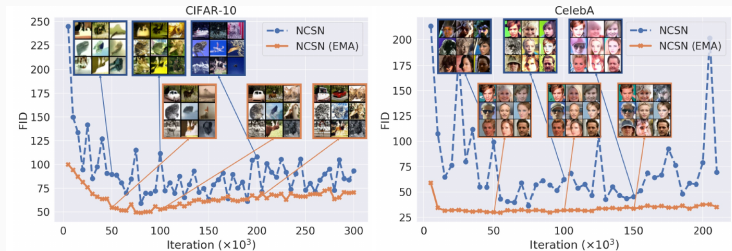
(source: learnopencv.com)

Exponential Moving Average

- Several choices for $t \mapsto \lambda_t$, linked to ELBO and data augmentation (Kingma and Gao, 2023).
- Training using Adam algorithm (Kingma and Ba, 2015), but still **unstable**.
- To regularize: **Exponential Moving Average** (EMA) of weights.

$$\bar{\theta}_{n+1} = (1 - m)\bar{\theta}_n + m\theta_n.$$

- Typically $m = 10^{-4}$ (more than 10^4 iterations are averaged).
- The final averaged parameters $\bar{\theta}_K$ are used at **sampling**.



Training instabilities

(source: (Song and Ermon, 2020))

Sampling strategy

- The score function of a distribution is generally used for Langevin sampling (ULA or MALA):

$$X_{n+1} = X_n + \gamma \nabla_x \log p(X_n) + \sqrt{2\gamma} Z_n$$

- (Song et al., 2021b) propose to add one step of Langevin diffusion (same $t = t_n$) after each step Euler-Maruyama step (t_n to t_{n+1}).
- This means that we jump from one trajectory to the other, but we correct some defaults from the Euler scheme.
- This is called a Predictor-Corrector sampler.

Algorithm 2 PC sampling (VE SDE)

```
1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ 
2: for  $i = N - 1$  to  $0$  do
3:    $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2) \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, \sigma_{i+1})$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2} \mathbf{z}$ 
6: for  $j = 1$  to  $M$  do
7:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9: return  $\mathbf{x}_0$ 
```

Algorithm 3 PC sampling (VP SDE)

```
1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $i = N - 1$  to  $0$  do
3:    $\mathbf{x}'_i \leftarrow (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i + 1)$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}} \mathbf{z}$  Predictor
6: for  $j = 1$  to  $M$  do Corrector
7:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9: return  $\mathbf{x}_0$ 
```

(source: (Song et al., 2021b))

- (Song et al., 2021b) achieved SOTA in terms of FID for CIFAR-10 unconditional sampling.
- Very good results for 1024×1024 portrait images.
- See also “Diffusion Models Beat GANs on Image Synthesis” (Dhariwal and Nichol, 2021) (self-explanatory title).



(source: FFHQ 1024×1024 samples (Song et al., 2021b))

Many approximations in the full generative pipelines:

- The final distribution p_T is not exactly a normal distribution.
- The learnt Unet model s_θ is far from being the exact score function: Sample-based, limitations from the architecture...
- Discrete sampling scheme (Euler-Maruyama, Predictor-Corrector,...).
- Score function may behave badly near $t = 0$ (irregular density in case of manifold hypothesis).

But we do have theoretical guarantees if all is well controlled!

Theorem (Convergence guarantees (De Bortoli, 2022))

Let p_0 be the data distribution having a compact manifold support and let q_T be the generator distribution from the reversed diffusion. Under suitable hypotheses, the 1-Wasserstein distance $W_1(p_0, q_T)$ can be explicitly bounded and tends to zero when all the parameters are refined (more Euler steps, better score learning, etc.).

**The deterministic approach:
Probability flow ODE**

We derived the Fokker-Planck equation for $q_t = p_{T-t}$ of reversed diffusion $\mathbf{y}_t = \mathbf{x}_{T-t}$.

$$\begin{aligned}\partial_t q_t(\mathbf{x}) &= -\partial_t p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t) p_{T-t}(\mathbf{x})) - \frac{1}{2} g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t) p_{T-t}(\mathbf{x})) + \left(-1 + \frac{1}{2}\right) g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= -\operatorname{div}_x \left(\left[-\mathbf{f}(\mathbf{x}, T-t) + g(T-t)^2 \nabla_x \log p_{T-t}(\mathbf{x}) \right] p_{T-t}(\mathbf{x}) \right) + \frac{1}{2} g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x})\end{aligned}$$

This is the Fokker-Planck equation associated with the diffusion SDE:

$$d\mathbf{y}_t = \left[-\mathbf{f}(\mathbf{y}_t, T-t) + g(T-t)^2 \nabla_x \log p_{T-t}(\mathbf{y}_t) \right] dt + g(T-t) d\mathbf{w}_t.$$

We derived the Fokker-Planck equation for $q_t = p_{T-t}$ of reversed diffusion
 $\mathbf{y}_t = \mathbf{x}_{T-t}$.

$$\begin{aligned}\partial_t q_t(\mathbf{x}) &= -\partial_t p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)p_{T-t}(\mathbf{x})) - \frac{1}{2}g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)p_{T-t}(\mathbf{x})) + \left(-1 + \frac{1}{2}\right)g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x})\end{aligned}$$

We derived the Fokker-Planck equation for $q_t = p_{T-t}$ of reversed diffusion

$$\mathbf{y}_t = \mathbf{x}_{T-t}.$$

$$\begin{aligned}\partial_t q_t(\mathbf{x}) &= -\partial_t p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)p_{T-t}(\mathbf{x})) - \frac{1}{2}g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t)p_{T-t}(\mathbf{x})) + \left(-\frac{1}{2} + 0\right)g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x})\end{aligned}$$

We derived the Fokker-Planck equation for $q_t = p_{T-t}$ of reversed diffusion $\mathbf{y}_t = \mathbf{x}_{T-t}$.

$$\begin{aligned}\partial_t q_t(\mathbf{x}) &= -\partial_t p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t) p_{T-t}(\mathbf{x})) - \frac{1}{2} g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= \operatorname{div}_x (\mathbf{f}(\mathbf{x}, T-t) p_{T-t}(\mathbf{x})) + \left(-\frac{1}{2} + 0\right) g(T-t)^2 \Delta_x p_{T-t}(\mathbf{x}) \\ &= -\operatorname{div}_x \left(\left[-\mathbf{f}(\mathbf{x}, T-t) + \frac{1}{2} g(T-t)^2 \nabla_x \log p_{T-t}(\mathbf{x}) \right] p_{T-t}(\mathbf{x}) \right)\end{aligned}$$

This is the Fokker-Planck equation associated with the diffusion SDE:

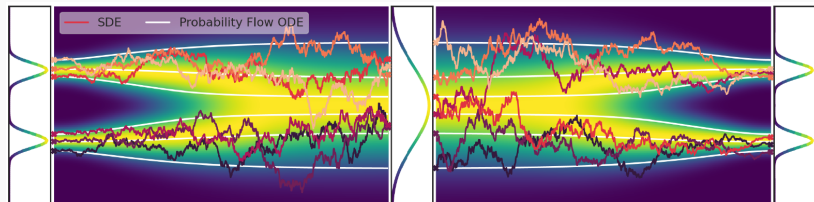
$$d\mathbf{y}_t = \left[-\mathbf{f}(\mathbf{y}_t, T-t) + \frac{1}{2} g(T-t)^2 \nabla_x \log p_{T-t}(\mathbf{y}_t) \right] dt.$$

which is an **Ordinary Differential Equation (ODE)** (no stochastic term) !

Reverse diffusion via an ODE

$$dy_t = \left[-f(y_t, T - t) + \frac{1}{2}g(T - t)^2 \nabla_x \log p_{T-t}(y_t) \right] dt.$$

This ODE is called a **probability flow ODE**.



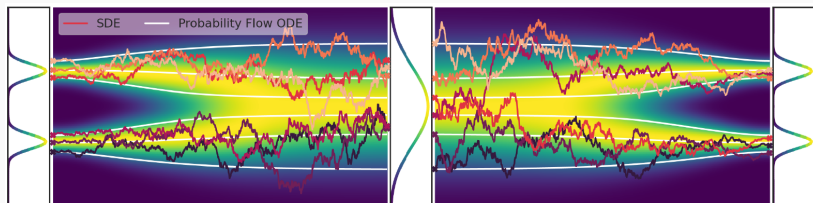
(source: (Song and Ermon, 2020))

- Like with normalizing flows, we get a deterministic mapping between initial noise and generated images.
- We do not simulate the (chaotic) path of the stochastic diffusion **but we still have the same marginal distribution** p_t .
- We can use **any ODE solver**, with higher order than Euler scheme.

Reverse diffusion via an ODE

$$dy_t = \left[-f(y_t, T - t) + \frac{1}{2}g(T - t)^2 \nabla_x \log p_{T-t}(y_t) \right] dt.$$

This ODE is called a **probability flow ODE**.

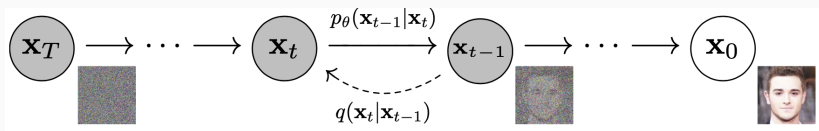


(source: (Song and Ermon, 2020))

- From (Karras et al., 2022) “Through extensive tests, we have found Heun’s 2nd order method (a.k.a. improved Euler, trapezoidal rule) [...] to provide an excellent tradeoff between truncation error and NFE.”
- Requires much less NFE than stochastic samplers (eg around 50 steps instead of 1000), see also Denoising Diffusion Implicit Models (DDIM) (Song et al., 2021a) for a deterministic approach.

**The discrete approach for diffusion
models:
Denoising Diffusion Probabilistic
Models**

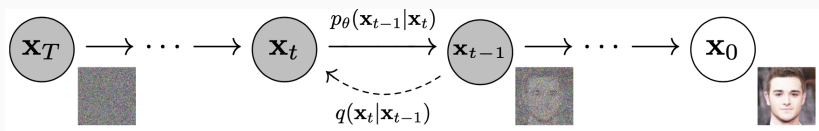
Denosing Diffusion Probabilistic Models



(source: (Ho et al., 2020))

Denosing Diffusion Probabilistic Models (**DDPM** (Ho et al., 2020)) is a discrete model with a fixed number of $T = 10^3$ steps that performs discrete diffusion.

Denosing Diffusion Probabilistic Models



(source: (Ho et al., 2020))

Denosing Diffusion Probabilistic Models (**DDPM** (Ho et al., 2020)) is a discrete model with a fixed number of $T = 10^3$ steps that performs discrete diffusion.

WARNING: Slight change of notation

Forward model: Discrete variance preserving diffusion

- Distribution of samples: $q(x_0)$.
- Conditional Gaussian noise: $q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1 - \beta_t}x_{t-1}, \beta_t I_d)$

$$x_t = \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}z_t$$

where the variance schedule $(\beta_t)_{1 \leq t \leq T}$ is fixed.

- One step noising $q(x_t|x_0)$: With $\alpha_t = 1 - \beta_t$ and $\bar{\alpha} = \text{cumprod}(\alpha)$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}z \quad \text{where } z \text{ is standard.}$$

- We consider the diffusion as a fixed stochastic encoder
- We want to learn a **stochastic decoder** p_θ :

$$p_\theta(\mathbf{x}_{0:T}) = \underbrace{p(\mathbf{x}_T)}_{\text{fixed latent prior}} \prod_{t=1}^T \underbrace{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{learnable backward transitions}} .$$

$$\text{with } p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \beta_t I_d)$$

$$\text{Compare with: } q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t I_d)$$

- Recall same diffusion coefficient, new backward drift to be learnt,...
- Oversimplified version compare to (Ho et al., 2020), there are ways to also learn the variance for each pixel, see (Nichol and Dhariwal, 2021).
- Then we look for training the decoder by maximizing an **ELBO**.

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

We have

$$L = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t=1}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right]$$

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

We have

$$L = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t=1}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right]$$

= ... (see [\(Ho et al., 2020\)](#) Appendix A)

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

We have

$$\begin{aligned} L &= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t=1}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\ &= \dots \quad (\text{see (Ho et al., 2020) Appendix A}) \\ &= \mathbb{E}_q \left[D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T)) + \sum_{t=2}^T D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) \right] \end{aligned}$$

Computation of $D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))$

By Bayes rule,

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

Computation shows that this is a normal distribution $\mathcal{N}(\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t I_d)$ with

$$\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t.$$

Using the expression of the KL-divergence between Gaussian distributions,

$$D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \frac{1}{\beta_t} \|\mu_{\theta}(\mathbf{x}_t, t) - \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 + C$$

$$L_t = \mathbb{E}_q [D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))] = \frac{1}{\beta_t} \mathbb{E}_q \left[\|\mu_{\theta}(\mathbf{x}_t, t) - \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right] + C$$

DDPM: Noise reparameterization

Rewrite everything **in function of the added standard noise ε** :

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

Then $\mu_\theta(\mathbf{x}_t, t)$ must predict

$$\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon \right)$$

If we parameterize

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(\mathbf{x}_t, t) \right)$$

Then the loss is simply

$$\begin{aligned} L_t &= \frac{\beta_t}{1 - \bar{\alpha}_t} \mathbb{E}_q \left[\|\varepsilon_\theta(\mathbf{x}_t, t) - \varepsilon\|^2 \right] + C \\ &= \frac{\beta_t}{1 - \bar{\alpha}_t} \mathbb{E}_{\mathbf{x}_0, \varepsilon} \left[\|\varepsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon, t) - \varepsilon\|^2 \right] + C \end{aligned}$$

That is we must predict the noise ε added to \mathbf{x}_0 (without knowing \mathbf{x}_0).

DDPM: Training and sampling

$$L = \mathbb{E}_q \left[D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)) + \sum_{t=2}^T D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right]$$
$$= \sum_{t=2}^T L_t + L_1 + C$$

- The L_1 term is dealt differently (to account for discretization of \mathbf{x}_0).
- (Ho et al., 2020) proposes to simplify the loss (no constants):

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

Algorithm 1 Training

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

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))

DDPM: Denoiser

The Unet $\varepsilon_\theta(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

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

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_\theta(\mathbf{x}_t, t).$$



x_t

\hat{x}_0

x_0

$t = 11$

DDPM: Denoiser

The Unet $\varepsilon_\theta(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

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

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_\theta(\mathbf{x}_t, t).$$



\mathbf{x}_t



$\hat{\mathbf{x}}_0$



\mathbf{x}_0

$t = 100$

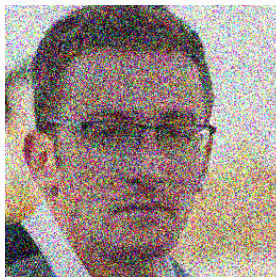
DDPM: Denoiser

The Unet $\varepsilon_{\theta}(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

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

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_{\theta}(\mathbf{x}_t, t).$$



\mathbf{x}_t



$\hat{\mathbf{x}}_0$



\mathbf{x}_0

$t = 200$

DDPM: Denoiser

The Unet $\varepsilon_\theta(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

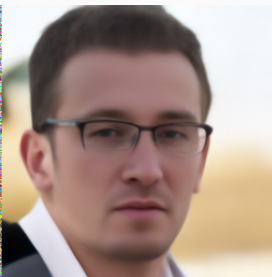
$$\mathbf{x}_t(\mathbf{x}_0, \varepsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon.$$

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_\theta(\mathbf{x}_t, t).$$



\mathbf{x}_t



$\hat{\mathbf{x}}_0$



\mathbf{x}_0

$t = 400$

DDPM: Denoiser

The Unet $\varepsilon_\theta(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

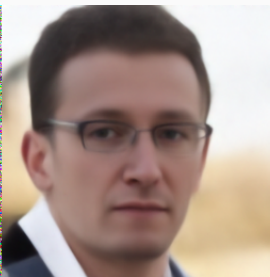
$$\mathbf{x}_t(\mathbf{x}_0, \varepsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon.$$

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_\theta(\mathbf{x}_t, t).$$



\mathbf{x}_t



$\hat{\mathbf{x}}_0$



\mathbf{x}_0

$t = 500$

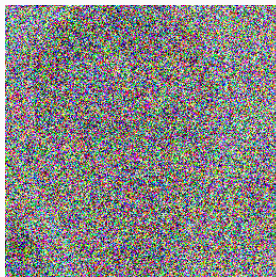
DDPM: Denoiser

The Unet $\varepsilon_{\theta}(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

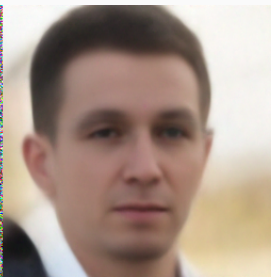
$$\mathbf{x}_t(\mathbf{x}_0, \varepsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon.$$

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_{\theta}(\mathbf{x}_t, t).$$



\mathbf{x}_t



$\hat{\mathbf{x}}_0$



\mathbf{x}_0

$t = 600$

DDPM: Denoiser

The Unet $\varepsilon_{\theta}(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

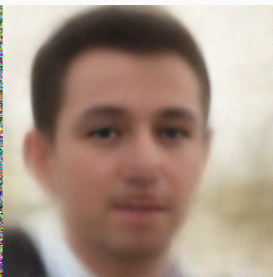
$$\mathbf{x}_t(\mathbf{x}_0, \varepsilon) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon.$$

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_{\theta}(\mathbf{x}_t, t).$$



\mathbf{x}_t



$\hat{\mathbf{x}}_0$



\mathbf{x}_0

$t = 700$

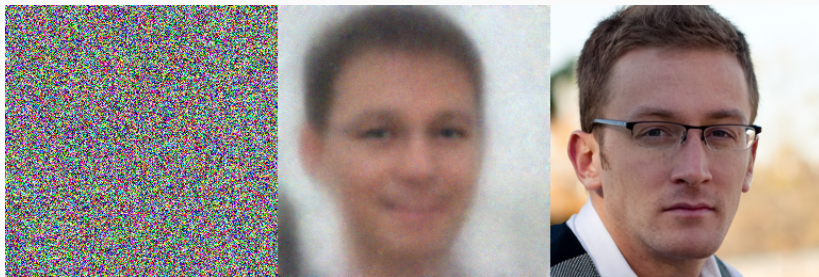
DDPM: Denoiser

The Unet $\varepsilon_{\theta}(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

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

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_{\theta}(\mathbf{x}_t, t).$$



x_t

\hat{x}_0

x_0

$t = 800$

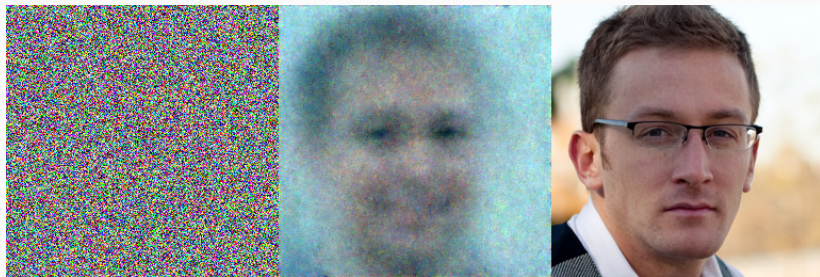
DDPM: Denoiser

The Unet $\varepsilon_{\theta}(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

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

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_{\theta}(\mathbf{x}_t, t).$$



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

\mathbf{x}_0

$t = 900$

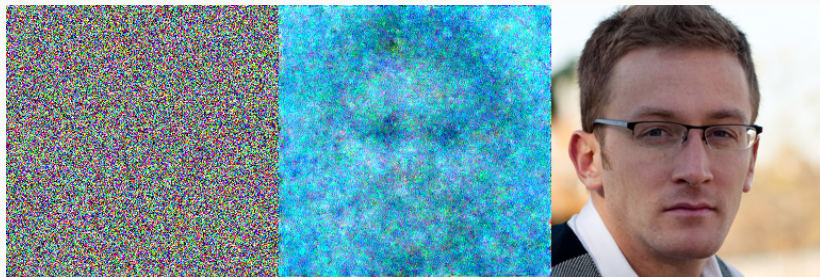
DDPM: Denoiser

The Unet $\varepsilon_{\theta}(\mathbf{x}_t, t)$ is a (residual) denoiser that gives an estimation of the noise ε from

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

We get the associated estimation of \mathbf{x}_0 :

$$\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\mathbf{x}_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\varepsilon_{\theta}(\mathbf{x}_t, t).$$



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

\mathbf{x}_0

$t = 1000$

DDPM: Sampling

Algorithm 1 Training

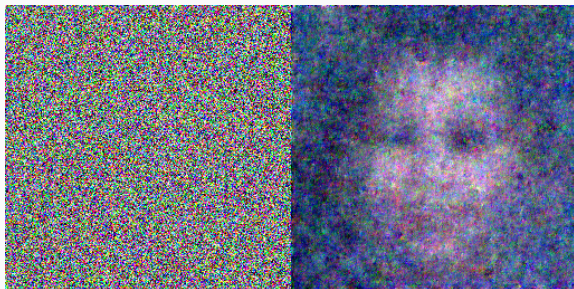
- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
$$\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}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

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 999$

DDPM: Sampling

Algorithm 1 Training

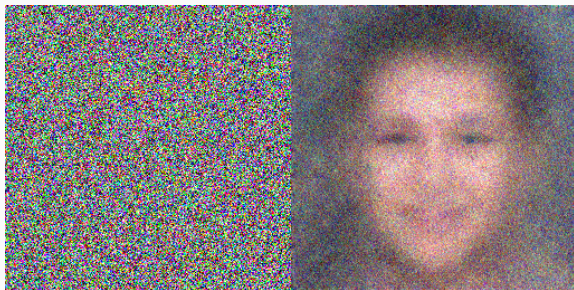
- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$
- 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return** \mathbf{x}_0

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 900$

DDPM: Sampling

Algorithm 1 Training

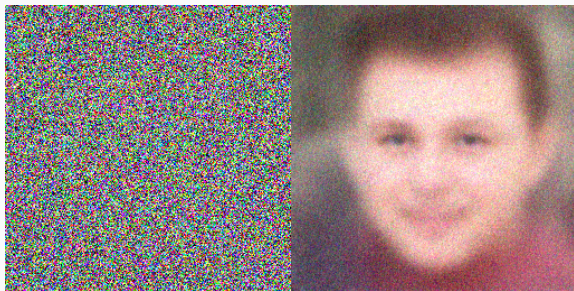
- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon,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

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 800$

DDPM: Sampling

Algorithm 1 Training

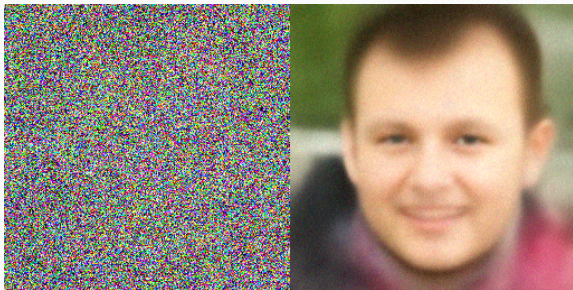
- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$
- 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return** \mathbf{x}_0

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 700$

DDPM: Sampling

Algorithm 1 Training

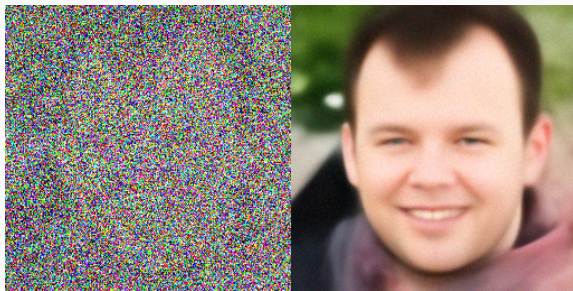
- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon,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

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 600$

DDPM: Sampling

Algorithm 1 Training

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

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 500$

DDPM: Sampling

Algorithm 1 Training

- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$
- 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return** \mathbf{x}_0

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 400$

DDPM: Sampling

Algorithm 1 Training

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

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 300$

DDPM: Sampling

Algorithm 1 Training

- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$
- 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return** \mathbf{x}_0

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 200$

DDPM: Sampling

Algorithm 1 Training

- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
 $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$
- 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return** \mathbf{x}_0

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

$\hat{\mathbf{x}}_0$

$t = 100$

Algorithm 1 Training

- 1: **repeat**
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \dots, T\})$
- 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 5: Take gradient descent step on
$$\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$$
- 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
- 5: **end for**
- 6: **return** \mathbf{x}_0

$\sigma_t = \sqrt{\beta_t}$ here.

(source: (Ho et al., 2020))



\mathbf{x}_t

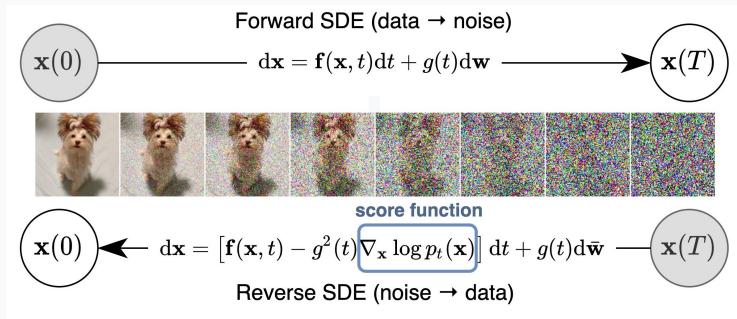
$\hat{\mathbf{x}}_0$

$t = 0$

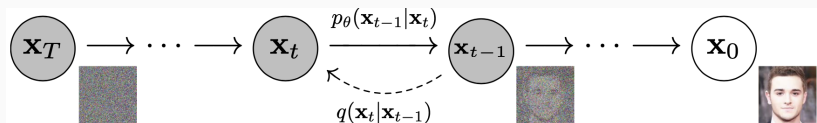
Continuous and discrete diffusion models

Recap on diffusion models

Diffusion model via SDE: (Song et al., 2021b)



Diffusion model via Denoising Diffusion Probabilistic Models (DDPM): (Ho et al., 2020) Discrete model with a fixed number of $T = 10^3$.



Forward diffusion:

$$dx_t = f(x_t, t)dt + g(t)dw_t$$

Backward diffusion: $y_t = x_{T-t}$

$$dy_t = \left[-f(y_t, T-t) + g(T-t)^2 \nabla_x \log p_{T-t}(y_t) \right] dt + g(T-t)dw_t.$$

- Learn score by denoising score matching:

$$\theta^* = \operatorname{argmin} \mathbb{E}_t \left(\lambda_t \mathbb{E}_{(x_0, x_t)} \|s_\theta(x_t, t) - \nabla_{x_t} \log p_{t|0}(x_t|x_0)\|^2 \right) \quad \text{with } t \sim \operatorname{Unif}([0, T])$$

- Generate samples by SDE discrete scheme (e.g. Euler-Maruyama):

$$Y_{n-1} = Y_n - hf(Y_n, t_n) + hg(t_n)^2 s_\theta(Y_n, t_n) + g(t_n) \sqrt{h} Z_n \quad \text{with } Z_n \sim \mathcal{N}(\mathbf{0}, I_d)$$

- Associated deterministic probability flow:

$$dy_t = \left[-f(y_t, T-t) + \frac{1}{2}g(T-t)^2 \nabla_x \log p_{T-t}(y_t) \right] dt$$

Denoising Diffusion Probabilistic Models (DDPM)

Forward diffusion:

$$q(x_{0:T}) = \underbrace{q(x_0)}_{\text{data distribution}} \prod_{t=1}^T \underbrace{q(x_t|x_{t-1})}_{\text{fixed forward transitions}} \quad \text{with} \quad q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1 - \beta_t}x_{t-1}, \beta_t I_d)$$

Backward diffusion: **stochastic decoder** p_θ :

$$p_\theta(x_{0:T}) = \underbrace{p(x_T)}_{\text{fixed latent prior}} \prod_{t=1}^T \underbrace{p_\theta(x_{t-1}|x_t)}_{\text{learnt backward transitions}} \quad \text{with} \quad \underbrace{p_\theta(x_{t-1}|x_t)}_{\text{Gaussian approximation of } q(x_{t-1}|x_t)} = \mathcal{N}(\mu_\theta(x_t, t), \beta_t I_d)$$

- Learn the score by minimizing the ELBO (like for VAE): This boils down to denoising the diffusion iterations $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$:

$$\theta^* = \operatorname{argmin} \sum_{t=1}^T \frac{\beta_t}{1 - \bar{\alpha}_t} \mathbb{E}_q \left[\|\epsilon_\theta(x_t, t) - \epsilon\|^2 \right] + C$$

- Sampling through the stochastic decoder with

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

Posterior mean training: Recall that $\mu_\theta(\mathbf{x}_t, t)$ minimizes

$$\mathbb{E}_q [D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))] = \frac{1}{\beta_t} \mathbb{E}_q \left[\|\mu_\theta(\mathbf{x}_t, t) - \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right] + C$$

where $\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)$ is the mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Hence ideally,

$$\mu_\theta(\mathbf{x}_t, t) = \mathbb{E} [\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0)|\mathbf{x}_t] = \mathbb{E} [\mathbb{E} [\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0]|\mathbf{x}_t] = \mathbb{E} [\mathbf{x}_{t-1}|\mathbf{x}_t].$$

Noise prediction training: $\varepsilon_\theta(\mathbf{x}_t, t)$ minimizes

$$\mathbb{E}_q \left[\|\varepsilon_\theta(\mathbf{x}_t, t) - \varepsilon\|^2 \right]$$

where ε is a function of $(\mathbf{x}_t, \mathbf{x}_0)$ (since $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$). Hence ideally,

$$\varepsilon_\theta(\mathbf{x}_t, t) = \mathbb{E} [\varepsilon|\mathbf{x}_t]$$

Score matching training: Ideally,

$$s_\theta(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E} [\nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)|\mathbf{x}_t]$$

We derived the formulas for DDPM training without considering the score function... but denoising and score functions are linked by **Tweedie formulas**:

Theorem (Tweedie formulas)

If $Y = aX + \sigma Z$ with $Z \sim \mathcal{N}(\mathbf{0}, I_d)$ independent of X , $a > 0$, $\sigma > 0$, then

Tweedie denoiser: $\mathbb{E}[X|Y] = \frac{1}{a} \left(Y + \sigma^2 \nabla_y \log p_Y(Y) \right)$

Tweedie noise predictor: $\mathbb{E}[Z|Y] = -\sigma \nabla_y \log p_Y(Y)$

If $Y = aX + \sigma Z$, **Tweedie denoiser:** $\mathbb{E}[X|Y] = \frac{1}{a} \left(Y + \sigma^2 \nabla_y \log p_Y(Y) \right)$

Tweedie noise predictor: $\mathbb{E}[Z|Y] = -\sigma \nabla_y \log p_Y(Y)$

Tweedie for noise prediction: Predict the noise ε from \mathbf{x}_t :

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon \Rightarrow \mathbb{E}[\varepsilon | \mathbf{x}_t] = -\sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

Tweedie for one-step denoising: Predict \mathbf{x}_{t-1} from \mathbf{x}_t :

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \mathbf{z}_t \Rightarrow \mathbb{E}[\mathbf{x}_{t-1} | \mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t + \beta_t \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right)$$

$$\mathbb{E}[\mathbf{x}_{t-1} | \mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbb{E}[\varepsilon | \mathbf{x}_t] \right)$$

If $Y = aX + \sigma Z$, **Tweedie denoiser:** $\mathbb{E}[X|Y] = \frac{1}{a} \left(Y + \sigma^2 \nabla_y \log p_Y(Y) \right)$

Tweedie noise predictor: $\mathbb{E}[Z|Y] = -\sigma \nabla_y \log p_Y(Y)$

Tweedie for noise prediction: Predict the noise ε from \mathbf{x}_t :

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon \Rightarrow \mathbb{E}[\varepsilon | \mathbf{x}_t] = -\sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

Tweedie for one-step denoising: Predict \mathbf{x}_{t-1} from \mathbf{x}_t :

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \mathbf{z}_t \Rightarrow \mathbb{E}[\mathbf{x}_{t-1} | \mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t + \beta_t \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) \right)$$

$$\mathbb{E}[\mathbf{x}_{t-1} | \mathbf{x}_t] = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \mathbb{E}[\varepsilon | \mathbf{x}_t] \right)$$

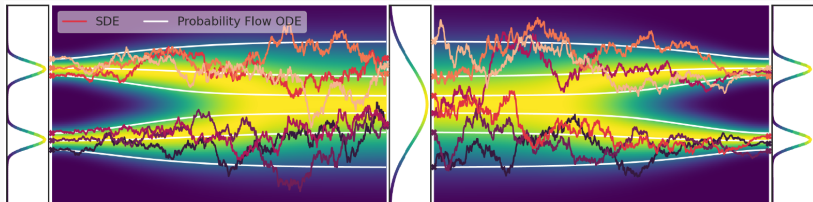
$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(\mathbf{x}_t, t) \right)$$

Remarks: We recover the expression of $\mu_\theta(\mathbf{x}_t, t)$ without using the one of

$$\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon \right)$$

To sum up:

- The three trainings strategies are the same (up to weighting constants).
- The only difference between the continuous SDE model and the discrete DDPM model are the time values: $t \in [0, T]$ VS. $t = 1, \dots, T = 10^3$.
- **Good news:** We can train a DDPM and use it for a deterministic probability flow ODE (this is what is done by the DDIM model (Song et al., 2021a)).



(source: (Song and Ermon, 2020))

Diffusion models for imaging inverse problems

Diffusion posterior sampling

We present **Diffusion Posterior Sampling (DPS)** for general noisy inverse problems ([Chung et al., 2023](#))

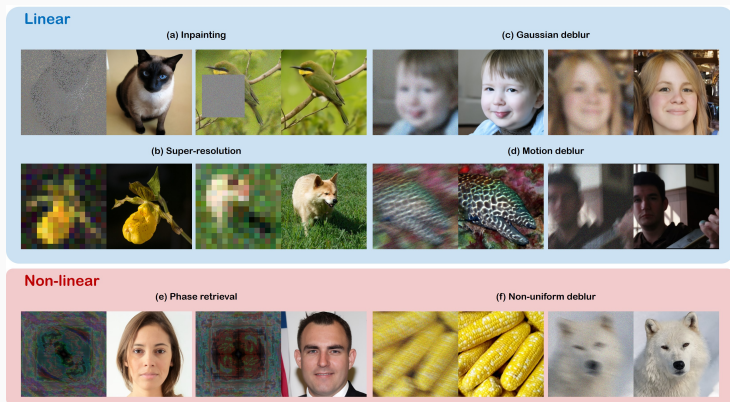


Figure 1: Solving noisy linear, and nonlinear inverse problems with diffusion models. Our reconstruction results (right) from the measurements (left) are shown.

(source: [Chung et al., 2023](#))

See also ([Song et al., 2023](#)), ([Kawar et al., 2022](#)) for alternative methods.

Let A be a linear operator from an inverse problem (masking operator for inpainting, blur operator for deblurring, subsampling for SR, ...).

Given some observation

$$\mathbf{y} = A\mathbf{x}_{\text{unknown}} + \mathbf{n}$$

where \mathbf{n} is some additive white Gaussian noise with variance σ^2 , we would like to sample

$$p_0(\mathbf{x}_0 | A\mathbf{x}_0 + \mathbf{n} = \mathbf{y}) = p_0(\mathbf{x}_0 | \mathbf{y})$$

to estimate $\mathbf{x}_{\text{unknown}}$ in accordance with the prior of the generative model.

Conditional sampling

From (Song et al., 2021b), we can consider the SDE for the conditional distribution $p_0(\mathbf{x}_0|\mathbf{y})$:

Backward diffusion for VP-SDE: $\mathbf{y}_t = \mathbf{x}_{T-t}$

$$d\mathbf{y}_t = [\beta_{T-t}\mathbf{y}_t + \beta_{T-t}\nabla_{\mathbf{x}=\mathbf{y}_t} \log p_{T-t}(\mathbf{y}_t)] dt + \beta_{T-t}d\mathbf{w}_t.$$

Conditional backward diffusion for VP-SDE: $\mathbf{y}_t = \mathbf{x}_{T-t}$

$$d\mathbf{y}_t = [\beta_{T-t}\mathbf{y}_t + \beta_{T-t}\nabla_{\mathbf{x}=\mathbf{y}_t} \log p_{T-t}(\mathbf{y}_t|\mathbf{y})] dt + \beta_{T-t}d\mathbf{w}_t.$$

By Bayes rule:

$$\log p_{T-t}(\mathbf{y}_t|\mathbf{y}) = \log p_{T-t}(\mathbf{y}|\mathbf{y}_t) + \log(p_{T-t}(\mathbf{y}_t)) - \log(p_{T-t}(\mathbf{y}))$$

Thus,

$$\nabla_{\mathbf{x}=\mathbf{y}_t} \log p_{T-t}(\mathbf{y}_t|\mathbf{y}) = \underbrace{\nabla_{\mathbf{x}=\mathbf{y}_t} \log p_{T-t}(\mathbf{y}|\mathbf{y}_t)}_{\text{intractable}} + \underbrace{\nabla_{\mathbf{x}=\mathbf{y}_t} \log(p_{T-t}(\mathbf{y}_t))}_{\text{usual score function}}$$

For clarity, let us write the new term with forward notation:

$$\nabla_{\mathbf{x}=\mathbf{y}_t} \log p_{T-t}(\mathbf{y}|\mathbf{y}_t) = \nabla_{\mathbf{x}=\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t)$$

(Chung et al., 2023) propose the following approximation:

$$\log p_t(\mathbf{y}|\mathbf{x}_t) \approx \log p_t(\mathbf{y}|\mathbf{x}_0 = \hat{\mathbf{x}}_0(\mathbf{x}_t, t))$$

with $\hat{\mathbf{x}}_0(\mathbf{x}_t, t)$ the estimate of the original image from the network.

Since

$$p(\mathbf{y}|\mathbf{x}_0) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\left(-\frac{\|\mathbf{y} - A\mathbf{x}_0\|^2}{2\sigma^2}\right)$$

we finally approximate

$$\nabla_{\mathbf{x}=\mathbf{x}_t} \log p_t(\mathbf{y}|\mathbf{x}_t) = -\frac{1}{2\sigma^2} \nabla_{\mathbf{x}_t} \|\mathbf{y} - A\hat{\mathbf{x}}_0(\mathbf{x}_t, t)\|^2$$

- Computing $\nabla_{\mathbf{x}_t} \|\mathbf{y} - A\hat{\mathbf{x}}_0(\mathbf{x}_t, t)\|^2$ involves a backpropagation through the Unet.
- One can expect this approximate conditional sampling to be twice as long as the sampling procedure.

Algorithm 1 DPS - Gaussian

Require: $N, \mathbf{y}, \{\zeta_i\}_{i=1}^N, \{\tilde{\sigma}_i\}_{i=1}^N$

1: $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

2: **for** $i = N - 1$ **to** 0 **do**

3: $\hat{\mathbf{s}} \leftarrow \mathbf{s}_\theta(\mathbf{x}_i, i)$

4: $\hat{\mathbf{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\mathbf{x}_i + (1 - \bar{\alpha}_i)\hat{\mathbf{s}})$

5: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

6: $\mathbf{x}'_{i-1} \leftarrow \frac{\sqrt{\bar{\alpha}_i(1-\bar{\alpha}_{i-1})}}{1-\bar{\alpha}_i}\mathbf{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}\beta_i}}{1-\bar{\alpha}_i}\hat{\mathbf{x}}_0 + \tilde{\sigma}_i\mathbf{z}$

7: $\mathbf{x}_{i-1} \leftarrow \mathbf{x}'_{i-1} - \zeta_i \nabla_{\mathbf{x}_i} \|\mathbf{y} - \mathcal{A}(\hat{\mathbf{x}}_0)\|_2^2$

8: **end for**

9: **return** $\hat{\mathbf{x}}_0$

(source: (Chung et al., 2023))

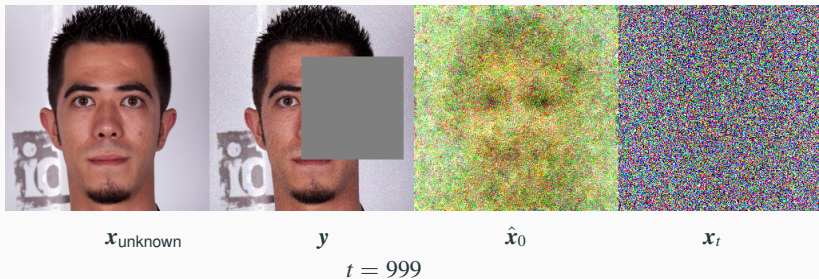
- Usual DDPM sampling (notation with $\hat{\mathbf{x}}_0(\mathbf{x}_t, t)$ instead of $\varepsilon_\theta(\mathbf{x}_t, t)$).

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(\mathbf{x}_t, t) \right) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}\beta_t}}{1 - \bar{\alpha}_t} \hat{\mathbf{x}}_0(\mathbf{x}_t, t)$$

- Add a correction term to drive $A\hat{\mathbf{x}}_0(\mathbf{x}_t, t)$ close to \mathbf{y} .
- In practice $\zeta_i = \zeta_t \propto \|\mathbf{y} - A\hat{\mathbf{x}}_0(\mathbf{x}_t, t)\|^{-1}$.

- Very good results in terms of perceptual metric (LPIPS).

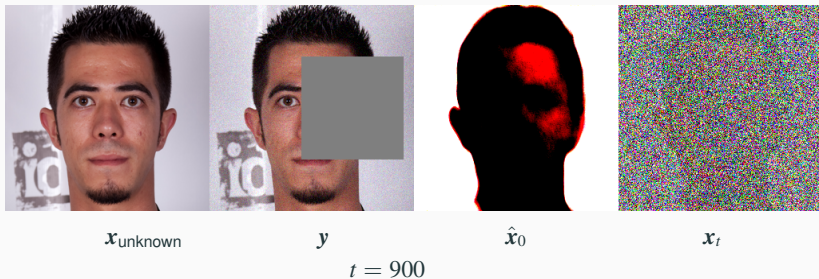
Inpainting:



Diffusion posterior sampling: Results

- Very good results in terms of perceptual metric (LPIPS).

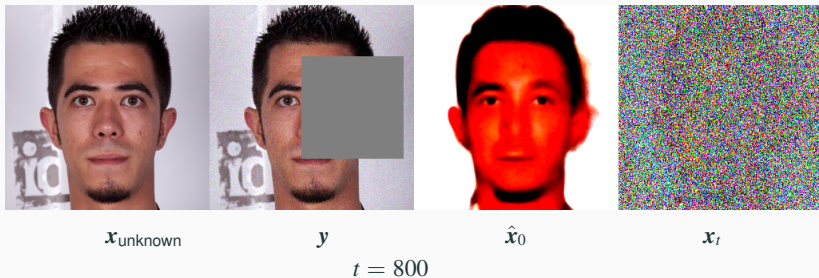
Inpainting:



Diffusion posterior sampling: Results

- Very good results in terms of perceptual metric (LPIPS).

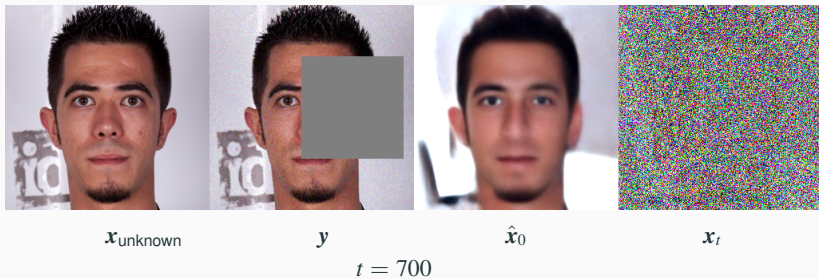
Inpainting:



Diffusion posterior sampling: Results

- Very good results in terms of perceptual metric (LPIPS).

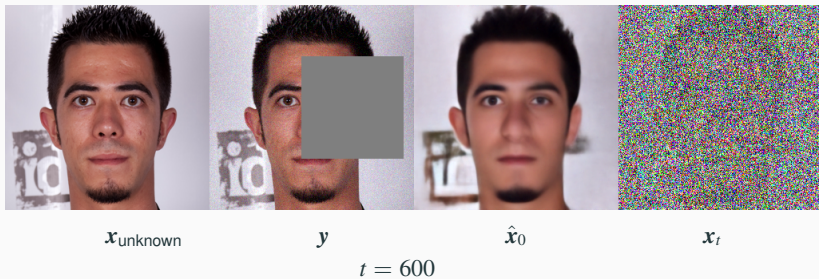
Inpainting:



Diffusion posterior sampling: Results

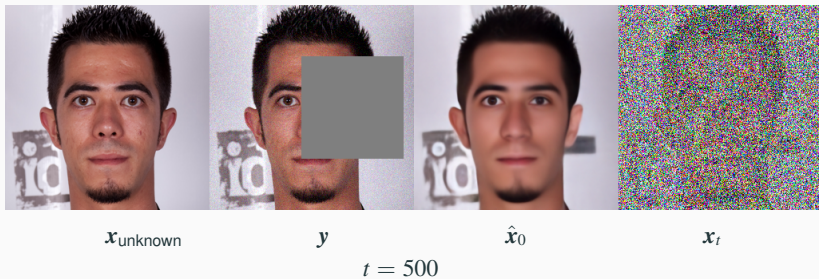
- Very good results in terms of perceptual metric (LPIPS).

Inpainting:



- Very good results in terms of perceptual metric (LPIPS).

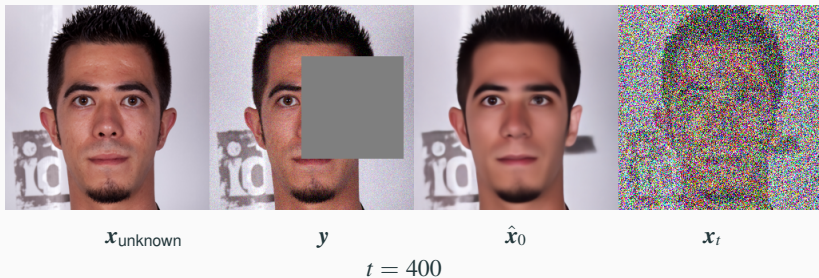
Inpainting:



Diffusion posterior sampling: Results

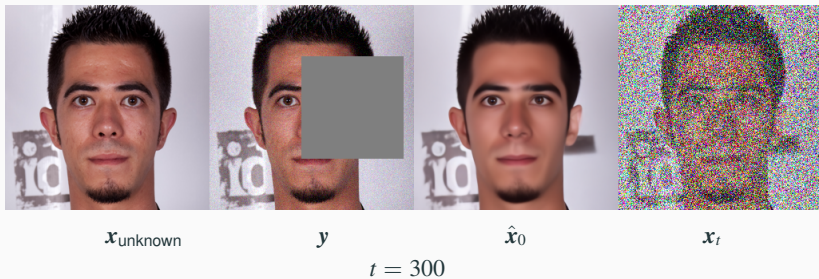
- Very good results in terms of perceptual metric (LPIPS).

Inpainting:



- Very good results in terms of perceptual metric (LPIPS).

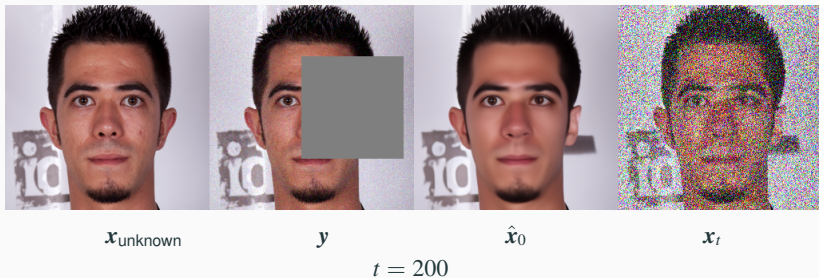
Inpainting:



Diffusion posterior sampling: Results

- Very good results in terms of perceptual metric (LPIPS).

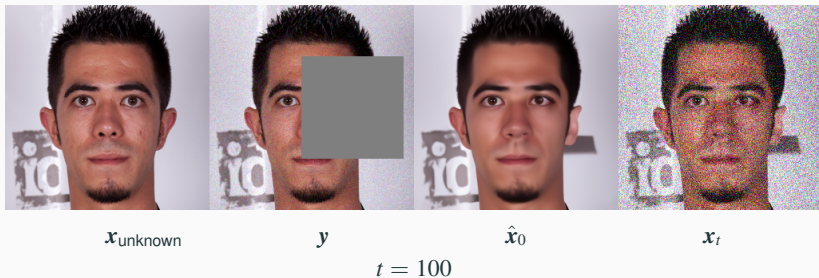
Inpainting:



Diffusion posterior sampling: Results

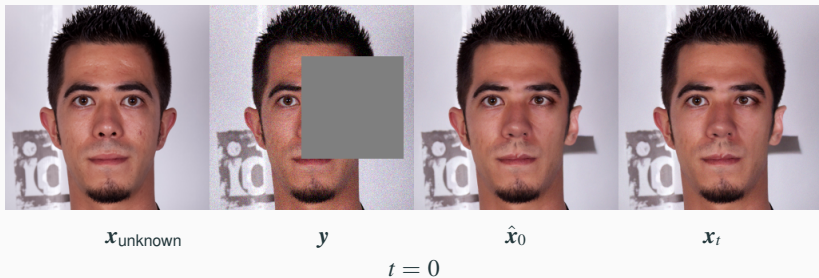
- Very good results in terms of perceptual metric (LPIPS).

Inpainting:



- Very good results in terms of perceptual metric (LPIPS).

Inpainting:

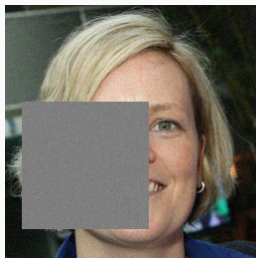


Diffusion posterior sampling: Inpainting results

- Very good results in terms of perceptual metric (LPIPS).
- Lack of symmetry.
- It can sometimes be really bad though!



original x_{unknown}



input y



output x_0

Diffusion posterior sampling: Inpainting results

- Very good results in terms of perceptual metric (LPIPS).
- Lack of symmetry.
- It can sometimes be really bad though!



original x_{unknown}



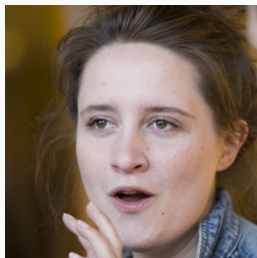
input y



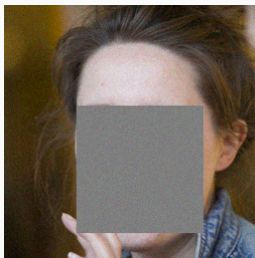
output x_0

Diffusion posterior sampling: Inpainting results

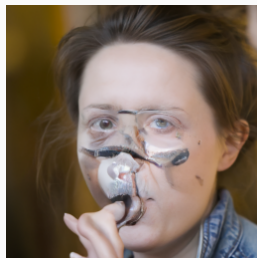
- Very good results in terms of perceptual metric (LPIPS).
- Lack of symmetry.
- It can sometimes be really bad though!



original x_{unknown}



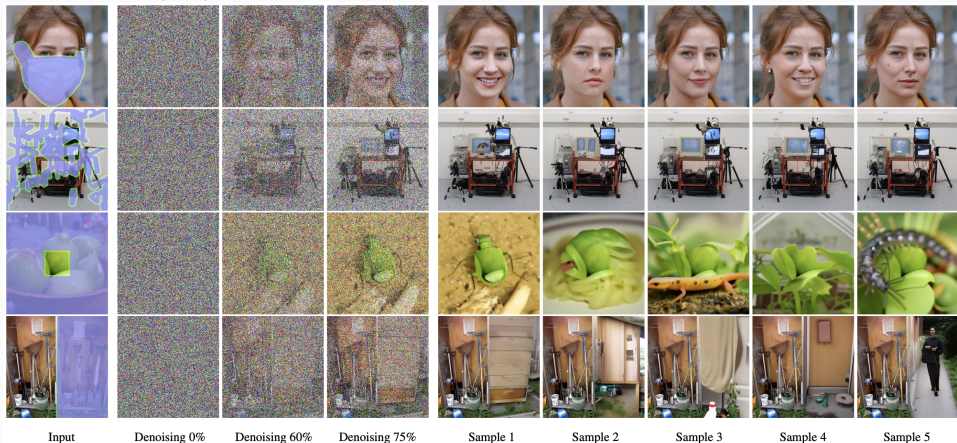
input y



output x_0

Diffusion posterior sampling: Inpainting results

- For inpainting it can help to go back and forth in the diffusion process (Lugmayr et al., 2022).



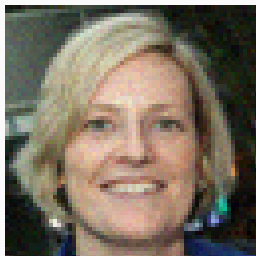
(source: (Lugmayr et al., 2022))

Diffusion posterior sampling: Super-resolution results

- Super-resolution with a factor $\times 4$.
- Very good results in terms of perceptual metric (LPIPS).
- Loss of details (skin defaults, etc.).



original x_{unknown}



input y



output x_0

Diffusion posterior sampling: Super-resolution results

- Super-resolution with a factor $\times 4$.
- Very good results in terms of perceptual metric (LPIPS).
- Loss of details (skin defaults, etc.).



original x_{unknown}



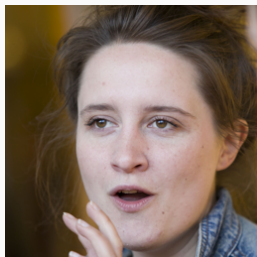
input y



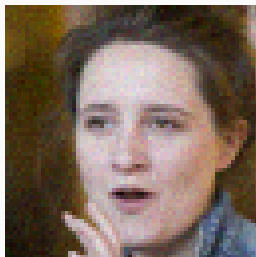
output x_0

Diffusion posterior sampling: Super-resolution results

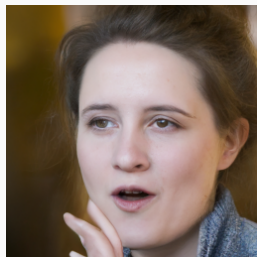
- Super-resolution with a factor $\times 4$.
- Very good results in terms of perceptual metric (LPIPS).
- Loss of details (skin defaults, etc.).



original x_{unknown}



input y



output x_0

Conditional DDPM for super-resolution

- Super-resolution is often used to improve the quality of generated images.
- One can train a specific DDPM for this task by conditioning the Unet with the low resolution image $\varepsilon_{\theta}(x_t, y_{LR}, t)$.

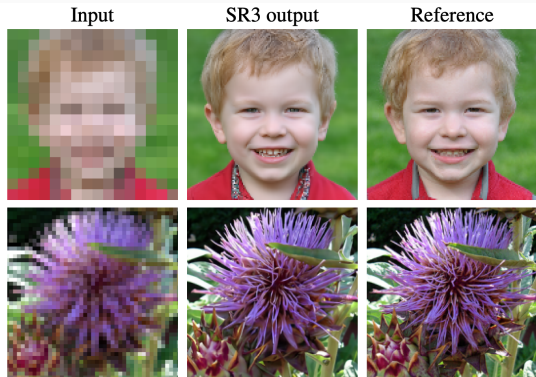
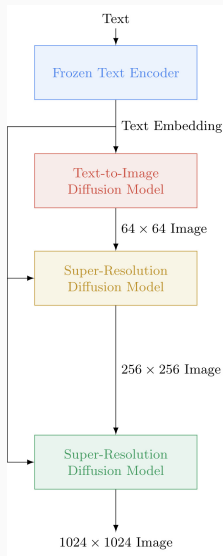


Figure 1: Two representative SR3 outputs: (top) $8\times$ face super-resolution at $16\times 16 \rightarrow 128\times 128$ pixels (bottom) $4\times$ natural image super-resolution at $64\times 64 \rightarrow 256\times 256$ pixels.

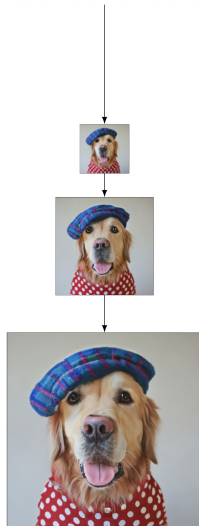
From (Saharia et al., 2023):
“To condition the model on the input y_{LR} , we upsample the low-resolution image to the target resolution using bicubic interpolation. The result is concatenated with x_t along the channel dimension.”

Conditional DDPM for super-resolution

Imagen pipeline:
Text conditioning
&
Conditional
super-resolution
via DDPM



“A Golden Retriever dog wearing a blue checkered beret and red dotted turtleneck.”



(source: (Saharia et al., 2022))

References

References

- Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.
- Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*.
- De Bortoli, V. (2022). Convergence of denoising diffusion models under the manifold hypothesis. *Transactions on Machine Learning Research*. Expert Certification.
- Dhariwal, P. and Nichol, A. Q. (2021). Diffusion models beat GANs on image synthesis. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.
- Hausmann, U. G. and Pardoux, E. (1986). Time reversal of diffusions. *The Annals of Probability*, 14(4):1188 – 1205.

- Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.
- Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.
- Kawar, B., Elad, M., Ermon, S., and Song, J. (2022). Denoising diffusion restoration models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 23593–23606. Curran Associates, Inc.
- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

- Kingma, D. P. and Gao, R. (2023). Understanding diffusion objectives as the ELBO with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. (2022). Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471.
- Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR.
- Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

- Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding.
- Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2023). Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726.
- Song, J., Meng, C., and Ermon, S. (2021a). Denoising diffusion implicit models. In *International Conference on Learning Representations*.
- Song, J., Vahdat, A., Mardani, M., and Kautz, J. (2023). Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*.
- Song, Y. and Ermon, S. (2020). Improved techniques for training score-based generative models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12438–12448. Curran Associates, Inc.

- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.