Diffusion models (theory & extensions)

Valentin De Bortoli

March 5, 2023

Summary of the previous lecture

- The previous lecture was an **introduction** to **diffusion models**:
  - **Ancestral sampling** and discrete **time-reversal**.
  - **Efficient** and **stable** implementation.
  - Links with **annealed Langevin**.
  - Connections with **EBMs** and **VAEs**.
- Recall the basics of **diffusion model**:
  - Sample a **forward trajectory**, noising the distribution.

$$\boxed{X_{k+1} = X_k - \gamma X_k + \sqrt{2\gamma} Z_{k+1} \ .}$$

  - Sample a **backward trajectory** via **ancestral sampling**.

$$\boxed{X_k = X_{k+1} + \gamma\{X_{k+1} + \mathbf{s}_\theta(k\gamma, X_{k+1})\} + \sqrt{2\gamma} Z_{k+1} \ .}$$

  - Backward sampling relies on learning the **score** (**score-matching**)

$$\boxed{\mathbf{s}_{\theta^\star}(k\gamma, \cdot) = \arg\min_\theta \{\mathbb{E}[\|\mathbf{s}_\theta(k\gamma, X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2] \ : \ f \in \mathrm{L}^2(p_k)\} \ .}$$

Continuous diffusion models

Continuous forward process

- Recall that in classical **diffusion models**, the forward dynamics is given by the following **Markov chain**

$$\boxed{X_{k+1} = X_k - \gamma X_k + \sqrt{2\gamma} Z_{k+1} \ .}$$

- This is the **Euler-Maruyama** discretization of the **Ornstein-Ulhenbeck** process.

$$\boxed{\mathrm{d}\mathbf{X}_t = -\mathbf{X}_t \mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- A technical point Ikeda and Watanabe (2014): two different definitions.
  - A **strong solution**: given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a Brownian motion $(\mathbf{B}_t)_{t\geq 0}$ with filtration $(\mathcal{F}_t)_{t\geq 0}$ we want to find $(\mathbf{X}_t)_{t\geq 0}$ which is $(\mathcal{F}_t)_{t\geq 0}$ adapted and for any $t \geq 0$

$$\boxed{\mathbf{X}_t = \mathbf{X}_0 + \int_0^t b(s, \mathbf{X}_s)\mathrm{d}s + \int_0^t \sigma(s, \mathbf{X}_s)\mathrm{d}\mathbf{B}_s \ .}$$

  - A **weak solution**: we just need that *there exists* a probability space such that such a process exists.

Technical point continued

- The weak formulation is equivalent to a **martingale** problem, see (Stroock and Varadhan, 1997, Chapter 6).
- We introduce the **infinitesimal generator** $\mathscr{A} : \mathbb{R}_+ \times \mathrm{C}^2(\mathbb{R}^d) \to \mathcal{F}(\mathbb{R}^d)$ such that for any $f \in \mathrm{C}^2(\mathbb{R}_+ \times \mathbb{R}^d)$ and $t \geq 0$

$$\boxed{\mathscr{A}(t,f)(x) = \langle b(t,x), \nabla f(t,x)\rangle + (1/2)\langle \Sigma(t,x), \nabla^2 f(t,x)\rangle \ .}$$

  - $\Sigma = \sigma^\top \sigma$.
  - Morally, $\mathbb{E}[\mathscr{A}(t,f)(\mathbf{X}_t)] = \lim_{h\to 0}(\mathbb{E}[f(\mathbf{X}_{t+h})] - \mathbb{E}[f(\mathbf{X}_t)])/h$.
- The existence of a weak solution is equivalent to the existence of a process $(\mathbf{X}_t)_{t\geq 0}$ such that for any $f \in \mathrm{C}^2(\mathbb{R}_+ \times \mathbb{R}^d)$, the process $(\mathbf{M}_t^f)_{t\geq 0}$ is a $(\mathcal{F}_t)_{t\geq 0}$-**martingale** where

$$\boxed{\mathbf{M}_t^f = f(t,\mathbf{X}_t) - f(0,\mathbf{X}_0) - \int_0^t (\partial_s f(s,\mathbf{X}_s) + \mathscr{A}(s,f)(\mathbf{X}_s))\mathrm{d}s \ .}$$

Analysis of the Ornstein-Ulhenbeck process

- Back to the **Ornstein-Ulhenbeck** (OU) process, **strong solutions** exist. The OU process is the *nicest* process:
  - $(\mathbf{X}_t)_{t\geq 0}$ is a Gaussian process
  - $(\mathbf{X}_t)_{t\geq 0}$ admits a closed form solution

$$\boxed{\mathbf{X}_t = \mathrm{e}^{-t}\mathbf{X}_0 + \mathbf{B}_{1-\mathrm{e}^{-2t}} \ .}$$

  - From the previous equation it is clear that

$$\boxed{\mathbf{W}_2(\mathcal{L}(\mathbf{X}_t), \mathrm{N}(0, \mathrm{Id})) \leq \mathrm{e}^{-t}\{\mathbb{E}^{1/2}[\|\mathbf{X}_0\|^2] + d^{1/2}\} \ .}$$

- We can also control the **Kullback-Leibler** divergence between $\mathcal{L}(\mathbf{X}_t)$ and $\mathrm{N}(0, \mathrm{Id})$ using that $\mathrm{N}(0, \mathrm{Id})$ satisfies a **logarithmic Sobolev** inequality Bakry et al. (2014).
- We can also control theχ^2 divergence between $\mathcal{L}(\mathbf{X}_t)$ and $\mathrm{N}(0, \mathrm{Id})$ using that $\mathrm{N}(0, \mathrm{Id})$ satisfies a **Poincaré** inequality Bakry et al. (2014).
- **Geometric** convergence rates **independent of the dimension**.
- Very fast mixing compared to Langevin dynamics targeting **non-convex** potentials.

Time reversal

- In discrete time we consider the **ancestral sampling** of the discretized **Ornstein-Ulhenbeck**.
- In the continuous-time setting we need to compute the **time-reversal** of the Ornstein-Ulhenbeck.
  - More precisely: does $(\mathbf{Y}_t)_{t\in[0,T]} = (\mathbf{X}_{T-t})_{t\in[0,T]}$ also satisfies a **Stochastic Differential Equation** (SDE)?
- The answer is *yes* under conditions and $(\mathbf{Y}_t)_{t\in[0,T]}$ is a (weak) solution of the following SDE

$$\boxed{\mathrm{d}\mathbf{Y}_t = \{\mathbf{Y}_t + 2\nabla \log p_{T-t}(\mathbf{Y}_t)\}\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- Note that for any $t \in [0, T]$, p_t is the density of $\mathcal{L}(\mathbf{X}_t)$ w.r.t. the Lebesgue measure, where we recall that $(\mathbf{X}_t)_{t\in[0,T]}$ is the **forward** noising process, here a **Ornstein-Ulhenbeck** process.
- A few remarks:
  - First found in Anderson (1982); Haussmann and Pardoux (1986).
  - The time-reversal formula is valid for **more complicated diffusions**.
  - Previous formula is valid in an **abstract setting** Cattiaux et al. (2021).

Time Reversal and Comparison with ancestral sampling

- Recall that in the **discrete-time** setting we have (denoting $\{Y_k\}_{k=0}^N = \{X_{N-k}\}_{k=0}^N$)

$$\boxed{Y_{k+1} = Y_k + \gamma\{Y_k + 2\nabla \log p_{N-k}(Y_k)\} + \sqrt{2\gamma} Z_{k+1} \ .}$$

- In the **continuous-time** setting we have

$$\boxed{\mathrm{d}\mathbf{Y}_t = \{\mathbf{Y}_t + 2\nabla \log p_{T-t}(\mathbf{Y}_t)\}\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- There is a clear link between the two formulations with **Euler-Maruyama** discretization.
- Note that p_{N-k} is the density of $\mathcal{L}(X_{N-k})$ while p_{T-t} is the density of $\mathcal{L}(\mathbf{X}_{T-t})$ but these two densities are close if the **stepsize is small**.
- In practice the **Stein score** is approximated using **score-matching**.
  - **DSM** and **ISM** losses can be defined in continuous-time.
  - Continuous losses can be used in practice because we can **exactly** sample from the **Ornstein-Ulhenbeck** process.

A first theoretical result (setting)

- We want to know how close the generative model is to the true **data distribution**.
- In order to do so we consider the **total variation** distance between $\mathcal{L}(Y_N)$ and p (the data distribution).
  - Recall that $\|\mu - \nu\|_{\mathrm{TV}} = \sup\{\mu(\mathsf{A}) - \nu(\mathsf{A}) \ : \ \mathsf{A} \in \mathcal{B}(\mathbb{R}^d)\}$
  - If μ, ν admit densities f, g w.r.t the Lebesgue measure we have that $\|\mu - \nu\|_{\mathrm{TV}} = \|f - g\|_1$.
  - This distance is stronger than the **weak convergence**.
  - This distance can be seen as a Wasserstein distance with cost ℓ_0.
- We are going to consider the **Markov chain**

$$\boxed{Y_{k+1} = Y_k + \gamma\{Y_k + 2\mathbf{s}_{\theta^\star}((N-k)\gamma, Y_k)\} + \sqrt{2\gamma}Z_{k+1} \ .}$$

- This Markov chain is initialized with $Y_0 \sim \mathrm{N}(0, \mathrm{Id})$.
- The **data distribution** is denoted by π.

Theoretical result

Convergence of diffusion models (De Bortoli et al., 2021)

- Assume there exists $M \geq 0$ such that for any $t \in [0, T]$ and $x \in \mathbb{R}^d$

$$||\mathbf{s}_{\theta^\star}(t, x) - \nabla \log p_t(x)|| \leq M \ ,$$

with $\mathbf{s}_{\theta^\star} \in C([0, T] \times \mathbb{R}^d, \mathbb{R}^d)$ and regularity conditions on the density of π w.r.t. the Lebesgue measure and its gradients.

- Then there exist $B, C, D \geq 0$ s.t. for any $N \in \mathbb{N}$ and $\{\gamma_k\}_{k=1}^N$ the following hold:

$$\boxed{\|\mathcal{L}(Y_N) - \pi\|_{\mathrm{TV}} \leq B \exp[-T] + C(M + \gamma^{1/2}) \exp[DT] \ .}$$

where $T = N\gamma$.

- **A few remarks**:
  - The assumption on π is *not* satisfied if π defined on a **manifold** of $\mathbb{R}^d$ with dimension $p < d$, see De Bortoli (2022)
  - The approximation assumption is strong and could be **relaxed**.
  - The term $\exp[DT]$ can be improved to a **polynomial dependency**.
  - Extension & Improvements Lee et al. (2022); Chen et al. (2022, 2023).

Sketch of the proof

- The central decomposition

$$\begin{aligned}\|\mathcal{L}(Y_N) - \pi\|_{\mathrm{TV}} &= \|\pi_0 \hat{\mathrm{R}}_N - \pi\|_{\mathrm{TV}} \\ &= \|\pi_0 \hat{\mathrm{R}}_N - p_T \mathrm{Q}_T\|_{\mathrm{TV}} \\ &\leq \|\pi_0 \hat{\mathrm{R}}_N - \pi_0 \mathrm{Q}_T\|_{\mathrm{TV}} + \|p_T \mathrm{Q}_T - \pi_0 \mathrm{Q}_T\|_{\mathrm{TV}} \\ &\leq \|\pi_0 \hat{\mathrm{R}}_N - \pi_0 \mathrm{Q}_T\|_{\mathrm{TV}} + \|\pi \mathrm{P}_T - \pi_0\|_{\mathrm{TV}} \ ,\end{aligned}$$

  where
  - $\pi_0 = \mathrm{N}(0, \mathrm{Id})$, π is the data distribution.
  - $(\mathrm{P}_t)_{t\in[0,T]}$ is the **forward** Ornstein-Ulhenbeck semi-group.
  - $(\mathrm{Q}_t)_{t\in[0,T]}$ is the **backward** Ornstein-Ulhenbeck semi-group.
  - $(\hat{\mathrm{R}}_n)_{n\in\{1,\dots,N\}}$ is the iterated kernel associated with the backward Markov chain.
- $\|\pi_0 \hat{\mathrm{R}}_N - \pi_0 \mathrm{Q}_T\|_{\mathrm{TV}}$ is the **approximation error**.
  - We first use the **Pinsker theorem** and control $\mathrm{KL}(\pi_0 \hat{\mathrm{R}}_N | \pi_0 \mathrm{Q}_T)$.
  - Then we use the **transfer theorem** and **Girsanov theory**, similarly to Durmus and Moulines (2017); Dalalyan (2017).
- $\|\pi \mathrm{P}_T - \pi_0\|_{\mathrm{TV}}$ is the **mixing time**.
  - This is simply the **geometric ergodicity** of the **Ornstein-Ulhenbeck**.

Bottleneck of score-matching methods

- We end up with

$$\boxed{\|\mathcal{L}(Y_N) - \pi\|_{\mathrm{TV}} \leq B\exp[-T] + C(M + \gamma^{1/2})\exp[DT]\ .}$$

- **A few remarks:**
  - The mixing time of the **reverse** is the same as the mixing time of the **forward**. This is a remarkable property because the **backward** targets a potentially complicated distribution.
  - Where is the trick? The **bottleneck** is in the **computation of the drift**.
  - To compare it with the **Unadjusted Langevin Algorithm** (ULA) (statistical sampling). The mixing time depends on the **convexity** of the target but the drift is **tractable**.
  - Here the mixing time is independent of the **convexity** of the target but the drift is **approximated**.
  - Note that the two algorithms *do not* solve the same problem. ULA addresses the **statistical sampling** problem while diffusion model addresses the **generative modeling** problem.

Likelihood computation and continuous normalizing flows

Outline of the section

- In this section:
  - We present a link with **Normalizing Flows**.
  - We highlight the **training differences** with normalizing flows.
  - We present **likelihood computation** results.
  - We show the advantages of a **deterministic** transform.

Figure 1: Interpolation with ODE flow. Image extracted from Song et al. (2021).

Recap on Continuous Normalizing Flows

- One **problem** of **normalizing flows**: the set of valid transformation is restricted by the **tractability of the log-Jacobian**.

- Moving from the **discrete** time setting to the **continuous** time setting allows **greater flexibility** (Chen et al. (2018); Grathwohl et al. (2018)).
  - Normalizing flow: $\mathcal{O}(d^3)$ computation.
  - **C**ontinuous **N**ormalizing **F**low (CNF): $\mathcal{O}(d^2)$ computation Chen et al. (2018).
  - CNF with trace estimator: $\mathcal{O}(d)$ computation Grathwohl et al. (2018).

- We introduce a **continuous** evolution:
  - A (stochastic) **dynamics** $\mathrm{d}\mathbf{X}_t = b(t, \mathbf{X}_t)\mathrm{d}t + \sigma(t, \mathbf{X}_t)\mathrm{d}\mathbf{B}_t$ with $\mathbf{X}_0 \sim \pi_0$.
  - $(\mathbf{B}_t)_{t\geq 0}$ is a d-dimensional Brownian motion.
  - $b : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$ smooth, π_0 has density p_0.
  - What is the evolution of $t \mapsto \log(p_t(\mathbf{X}_t))$, where $p_t = \mathcal{L}(\mathbf{X}_t)$?

- This is equivalent to a continuous **change of variable**.

Recap on the Fokker-Planck equation

- We have the **Fokker-Planck** equation

$$\boxed{\partial_t p_t(x) = -\mathrm{div}(b(t,\cdot)p_t)(x) + \tfrac{1}{2}\sum_{i,j=1}^{d} \partial_{i,j}\{\Sigma_{i,j}(t,\cdot)p_t\}(x)\ ,}$$

where $\mathrm{div}(a(t,x)) := \sum_{i=1}^{d} \partial_{x_i} a_i(t,x)$.

- This equation describes the **evolution of the density**.
- Some special cases:
  - Case $\sigma = 0$ (**deterministic dynamics**)

$$\boxed{\partial_t p_t(x) = -\mathrm{div}(b(t,\cdot)p_t)(x)\ .}$$

  - Case $\sigma = c^{1/2}\,\mathrm{Id}$ $(c > 0)$ (**Langevin dynamics**)

$$\boxed{\partial_t p_t(x) = -\mathrm{div}(b(t,\cdot)p_t)(x) + \tfrac{c}{2}\Delta p_t(x) = -\mathrm{div}(\{b(t,\cdot) - \tfrac{c}{2}\nabla \log p_t\}p_t)(x)\ .}$$

- As a consequence the two following dynamics have the **same** marginal densities.
  - $\mathrm{d}\mathbf{X}_t = b(t,\mathbf{X}_t)\mathrm{d}t + c^{1/2}\mathrm{d}\mathbf{B}_t$
  - $\mathrm{d}\mathbf{X}_t = \{b(t,\mathbf{X}_t) - \tfrac{c}{2}\nabla \log p_t(\mathbf{X}_t)\}\mathrm{d}t.$
  - One is **deterministic**, the other is **stochastic**.

Log-likelihood evaluation

- Assume that the **deterministic** process $(\mathbf{X}_t)_{t\in[0,T]}$ satisfies

$$\boxed{\mathrm{d}\mathbf{X}_t = b(t,\mathbf{X}_t)\mathrm{d}t\ .}$$

- The associated **Fokker-Planck** equation is

$$\boxed{\partial_t p_t(x) = -\mathrm{div}(b(t,\cdot)p_t)(x)\ .}$$

- **Evolution of the logarithm**

$$\partial_t \log p_t(x) = -\mathrm{div}(b(t,\cdot))(x) - \langle b(t,x), \nabla \log p_t(x)\rangle\ .$$

- Combining the two dynamics

$$\begin{aligned}\mathrm{d}\log p_t(\mathbf{X}_t) &= [\partial_t \log p_t(\mathbf{X}_t) + \langle \mathrm{d}\mathbf{X}_t, \nabla \log p_t(\mathbf{X}_t)\rangle]\mathrm{d}t\\ &= [-\mathrm{div}(b(t,\cdot))(\mathbf{X}_t) - \langle b(t,\mathbf{X}_t), \nabla \log p_t(\mathbf{X}_t) + \langle b(t,\mathbf{X}_t), \nabla \log p_t(\mathbf{X}_t)]\mathrm{d}t\\ &= -\mathrm{div}(b(t,\cdot))(\mathbf{X}_t)\mathrm{d}t\ .\end{aligned}$$

- Hence, we have the following **log-likelihood** computation

$$\boxed{\log(p_0(\mathbf{X}_0)) = \log p_T(\mathbf{X}_T) + \int_0^T \mathrm{div}(b(t,\cdot))(\mathbf{X}_t)\mathrm{d}t\ .}$$

From score-based models to normalizing flows

- Recall that the **continuous**-time version of **diffusion model** is given by

$$\boxed{\mathrm{d}\mathbf{Y}_t = \{\mathbf{Y}_t + 2\nabla \log p_{T-t}(\mathbf{Y}_t)\}\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- It is the **backward** dynamics associated with the **forward Ornstein-Ulhenbeck**

$$\boxed{\mathrm{d}\mathbf{X}_t = -\mathbf{X}_t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- For any $t \geq 0$, $\mathcal{L}(\mathbf{X}_t)$ has the same distribution as $\mathcal{L}(\hat{\mathbf{X}}_t)$ where for any $t \geq 0$

$$\boxed{\mathrm{d}\hat{\mathbf{X}}_t = \{-\hat{\mathbf{X}}_t - \nabla \log p_t(\hat{\mathbf{X}}_t)\}\mathrm{d}t \ .}$$

- The **backward** of this **deterministic** dynamics is $(\hat{\mathbf{Y}}_t)_{t\in[0,T]}$ given by

$$\boxed{\mathrm{d}\hat{\mathbf{Y}}_t = \{\hat{\mathbf{Y}}_t + \nabla \log p_{T-t}(\hat{\mathbf{Y}}_t)\}\mathrm{d}t \ .}$$

- Comparing $(\mathbf{Y}_t)_{t\in[0,T]}$ and $(\hat{\mathbf{Y}}_t)_{t\in[0,T]}$
  - $(\mathbf{Y}_t)_{t\in[0,T]}$ is **stochastic** and $(\hat{\mathbf{Y}}_t)_{t\in[0,T]}$ is **deterministic**.
  - Both are given by the **time-reversal** of a **forward dynamics**.
  - They both approximate the **data distribution** at time $t = T$.
  - Notice the difference of a **factor two** in the drift!

Stochastic and deterministic trajectories

- Recall that the **forward Ornstein-Ulhenbeck** is given by

$$\boxed{\mathrm{d}\mathbf{X}_t = -\mathbf{X}_t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- Its **deterministic** counterpart is given by

$$\boxed{\mathrm{d}\hat{\mathbf{X}}_t = \{-\hat{\mathbf{X}}_t - \nabla \log p_t(\hat{\mathbf{X}}_t)\}\mathrm{d}t \ .}$$

- By the **Fokker-Planck** equation (and its uniqueness) we have that for any $t \geq 0$, $\mathcal{L}(\mathbf{X}_t) = \mathcal{L}(\hat{\mathbf{X}}_t)$.
- Is this property still true for the **backward** dynamics $(\mathbf{Y}_t)_{t\in[0,T]}$ and $(\hat{\mathbf{Y}}_t)_{t\in[0,T]}$?
- Both dynamics are started at $\mathcal{L}(\mathbf{Y}_0) = \mathcal{L}(\hat{\mathbf{Y}}_0)$. The question is then: do the distributions $(\mathcal{L}(\mathbf{Y}_t))_{t\in[0,T]}$ and $(\mathcal{L}(\hat{\mathbf{Y}}_t))_{t\in[0,T]}$ follow the same dynamics?
- We are going to compute the associated **Fokker-Planck** evolution equations.

Fokker-Planck of the stochastic backward dynamics

- Recall that the **stochastic** time-reversal is given by

$$\boxed{\mathrm{d}\mathbf{Y}_t = \{\mathbf{Y}_t + 2\nabla \log p_{T-t}(\mathbf{Y}_t)\}\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- Denote $(q_t)_{t\in[0,T]}$ the distribution of $(\mathbf{Y}_t)_{t\in[0,T]}$.
- We have the following evolution equation

$$\begin{aligned}\partial_t q_t(x) &= -\mathrm{div}(q_t(x)\{x + 2\nabla \log p_{T-t}(x)\}) + \Delta q_t \\ &= -\mathrm{div}(q_t(x)\{x + 2\nabla \log p_{T-t}(x) - \nabla \log q_t(x)\}) \ .\end{aligned}$$

- Recall that the **deterministic** time-reversal is given by

$$\boxed{\mathrm{d}\hat{\mathbf{Y}}_t = \{\hat{\mathbf{Y}}_t + \nabla \log p_{T-t}(\hat{\mathbf{Y}}_t)\}\mathrm{d}t \ .}$$

- Denote $(\hat{q}_t)_{t\in[0,T]}$ the distribution of $(\hat{\mathbf{Y}}_t)_{t\in[0,T]}$.
- We have the following evolution equation

$$\partial_t \hat{q}_t = -\mathrm{div}(\hat{q}_t(x)\{x + \nabla \log p_{T-t}(x)\}) \ .$$

- The **evolutions are equal** if $p_{T-t} = q_t = \hat{q}_t$, i.e. $\mathcal{L}(\mathbf{Y}_0) = \mathcal{L}(\hat{\mathbf{Y}}_0) = \mathcal{L}(\mathbf{X}_T)$.
- Hence as $T \to +\infty$ the two dynamics get **closer**, see De Bortoli et al. (2022).

Difference between stochastic and deterministic dynamics

- We have a **stochastic** and a **deterministic** representation of $\mathcal{L}(\mathbf{X}_t)$ where for any $t \geq 0$

$$\boxed{\mathrm{d}\mathbf{X}_t = -\mathbf{X}_t + \sqrt{2}\mathrm{d}\mathbf{B}_t \ .}$$

- Are the **dynamics** the same? The answer is **no**. They can be really different.
- An example:

- $\mathcal{L}(\mathbf{X}_0) = \mathrm{N}(m, \mathrm{Id})$ with $m \in \mathbb{R}^d$.
- $\mathbf{X}_t = \mathrm{e}^{-t}\mathbf{X}_0 + \mathbf{B}_{1-\mathrm{e}^{-2t}}$, $p_t = \mathrm{N}(\mathrm{e}^{-t}m, \mathrm{Id})$
- Recall that the **deterministic** dynamics is given by

$$\boxed{\mathrm{d}\hat{\mathbf{X}}_t = \{-\hat{\mathbf{X}}_t - \nabla \log p_t(\hat{\mathbf{X}}_t)\}\mathrm{d}t = \{-\hat{\mathbf{X}}_t + \hat{\mathbf{X}}_t - m\mathrm{e}^{-t}\}\mathrm{d}t = -m\mathrm{e}^{-t}\mathrm{d}t \ .}$$

- $\hat{\mathbf{X}}_t = \mathbf{X}_0 - m(1 - \mathrm{e}^{-t})$.
- No **forgetting of the initial condition** with the **deterministic** dynamics.
- Khrulkov and Oseledets (2022) show that the deterministic dynamics yield the **optimal transport** between the Gaussian distributions.

Interpolation with normalizing flows

- Having a **deterministic** model is useful for:
  - **Likelihood computation**
  - **Interpolation**
  - **Temperature scaling**
- We can explore the **latent structure**.

Figure 2: Interpolation with ODE. Image extracted from Song et al. (2021).

Temperature scaling with normalizing flows

Figure 3: Temperature scaling with ODE Image extracted from Song et al. (2021).

Faster diffusion models

Few-step sampling: a well-known problem

- For **high-quality** image sampling **vanilla** diffusion models are notably **slow**.

A critical drawback of these models is that they require many iterations to produce a high quality sample. For DDPMs, this is because that the generative process (from noise to data) approximates the reverse of the forward *diffusion process* (from data to noise), which could have thousands of steps; iterating over all the steps is required to produce a single sample, which is much slower compared to GANs, which only needs one pass through a network. For example, it takes around 20

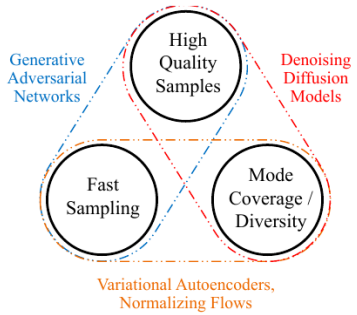
control the generation sample. To obtain high-quality synthesis, a large number of denoising steps is used (i.e. 1000 steps). A notable property of the diffusion process is a closed-form formulation of

network). Although very powerful, score-based models generate data through an undesirably long iterative process; meanwhile, other state-of-the-art methods such as GANs generate data from a single forward pass of a neural network. Increasing the speed of the generative process is thus an active area of research.

denoises the samples under the fixed noise schedule. However, DDPMs often need hundreds-to-thousands of denoising steps (each involving a feedforward pass of a large neural network) to achieve

However, GANs are typically much more efficient than DDPMs at generation time, often requiring a single forward pass through the generator network, whereas DDPMs require hundreds of forward passes through a U-Net model. Instead of learning a generator directly, DDPMs learn to convert

A major downside to score-based generative models is that they require performing expensive MCMC sampling, often with a thousand steps or more. As a result, they can be up to three orders of magnitude slower than GANs, which only require a single network evaluation. To address this issue, Denoising Diffusion Implicit Models, or DDIMs, have been



A myriad of methods

- A bunch of **approaches** have been proposed to solve this problem:
  - **D**enoising **D**iffusion **I**mplicit **M**odels (**DDIM**) Song et al. (2020).
  - **Discovering the stepsizes** Watson et al. (2022, 2021).
  - Knowledge **distillation** Luhman and Luhman (2021); Salimans and Ho (2022).
  - Improved **SDE solvers** Jolicoeur-Martineau et al. (2021); Liu et al. (2022).
  - **Non-Gaussian** diffusions Nachmani et al. (2021).
  - **Multiscale** denoising Saharia et al. (2021).
  - **GAN jumps** Xiao et al. (2021).

- We present two of these approaches:
  - DDIM and **subsampling** Song et al. (2020).
  - Discovering the stepsize and **dynamic programming** Watson et al. (2021).

Non-Markov models and subsampling

From DDPM to DDIM

- **DDIM** (**D**enoising **D**iffusion **I**mplicit **M**odel) is based on **DDPM** (**D**enoising **D**iffusion **P**robabilistic **M**odel).
- Both are based on the **variational approach** to recover **diffusion models**.
- The contributions of **DDIM**:
  - A **non-Markov** noising process (interpolating between **deterministic** and **stochastic** dynamics).
  - An **accelerated** variational formulation.
  - **Combining** these techniques lead to high-quality accelerated models.

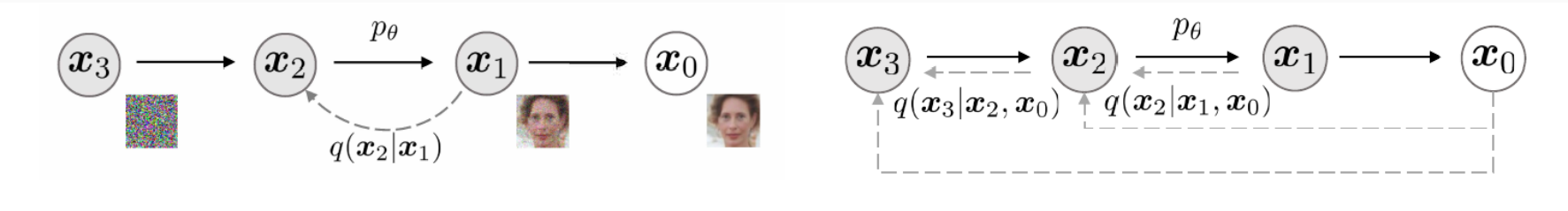


Figure 4: DDPM/DDIM comparison. Image extracted from Song et al. (2020).

Recap on DDPM (1/2)

- We start by deriving an **ELBO** for the **score-based generative models**. Note that such a derivation was already obtained by Sohl-Dickstein et al. (2015).
- Similar to VAE we maximize the **log-likelihood**

$$\begin{aligned}\log(p_{\theta,0}(x_0)) &= \log\left(\int_{(\mathbb{R}^d)^N} \prod_{k=0}^{N-1} p_{\theta,k|k+1}(x_k|x_{k+1}) p_N(x_N) \mathrm{d}x_{1:N}\right) \\ &= \log\left(\int_{(\mathbb{R}^d)^N} \prod_{k=0}^{N-1} p_{\theta,k|k+1}(x_k|x_{k+1}) p_N(x_N) / q(x_{1:N}|x_0) q(x_{1:N}|x_0) \mathrm{d}x_{1:N}\right) \\ &\geq \int_{(\mathbb{R}^d)^N} \log\left(\prod_{k=0}^{N-1} p_{\theta,k|k+1}(x_k|x_{k+1}) p_N(x_N) / q(x_{1:N}|x_0)\right) q(x_{1:N}|x_0) \mathrm{d}x_{1:N} \ .\end{aligned}$$

- The last inequality is obtained the **concavity** of the logarithm.
- We now choose the **variational distribution** $q(x_{1:N}|x_0)$:
  - **Factorization** $q(x_{1:N}|x_0) = q_{N|0}(x_N|X_0) \prod_{k=1}^{N-1} q_{k|k+1,0}(x_k|x_{k+1}, x_0)$.
  - **Tractability** of $q_{k|k+1,0}$.
  - We choose a tractable (Gaussian) decomposition

  $q(x_{1:N}|x_0) = \prod_{k=0}^{N-1} q_{k+1|k}(x_{k+1}|x_k)$.
- Here, we consider

$$\boxed{q_{k+1|k}(x_{k+1}|x_k) = (4\pi\gamma)^{-d/2} \exp[-\|x_{k+1} - x_k + \gamma x_k\|^2/(4\gamma)] \ .}$$

- This is a slightly different discretization from the one of Ho et al. (2020).

Recap on DDPM (2/2)

- Recall that we have $\log(p_{\theta,0}(x_0)) \geq \mathcal{L}$ with

$$\boxed{\mathcal{L} = \int_{(\mathbb{R}^d)^N} \log\left(\prod_{k=0}^{N-1} p_{\theta,k|k+1}(x_k|x_{k+1}) p_N(x_N) / q(x_{1:N}|x_0)\right) q(x_{1:N}|x_0) \mathrm{d}x_{1:N} \ .}$$

- We use the **backward decomposition** of $q(x_{1:N}|x_0)$ and we get

$$\boxed{\mathcal{L} = \mathcal{L}_N + \sum_{k=1}^{N-1} \mathcal{L}_k + \mathcal{L}_0 \ ,}$$

- with:
  - $\mathcal{L}_N = \int_{\mathbb{R}^d} \log(p_N(x_N)/q(x_N|x_0)) q_{N|0}(x_N|x_0) \mathrm{d}x_N$.
  - $\mathcal{L}_k = \int_{\mathbb{R}^d} \log(p_{\theta,k|k+1}(x_k|x_{k+1})/q_{k|k+1,0}(x_k|x_{k+1}, x_0)) q_{k,k+1|0}(x_k, x_{k+1}|x_0) \mathrm{d}x_k$.
  - $\mathcal{L}_0 = \int_{\mathbb{R}^d} \log(p_{\theta,0|1}(x_0|x_1)) q_{1|0}(x_1|x_0) \mathrm{d}x_1$.
- **The different terms**:
  - $\mathcal{L}_N$ does not depend on θ.
  - $\mathcal{L}_k$ is related to **score-matching**.
  - $\mathcal{L}_0$ is a **reconstruction** term.

Minimal requirements on the noising distribution

- We have $\log(p_{\theta,0}(x_0)) \geq \mathcal{L}$ with

$$\boxed{\mathcal{L} = \int_{(\mathbb{R}^d)^N} \log\left(\prod_{k=0}^{N-1} p_{\theta,k|k+1}(x_k|x_{k+1}) p_N(x_N) / q(x_{1:N}|x_0)\right) q(x_{1:N}|x_0) \mathrm{d}x_{1:N} \ .}$$

- We use the **backward decomposition** of $q(x_{1:N}|x_0)$ and we get

$$\boxed{\mathcal{L} = \mathcal{L}_N + \sum_{k=1}^{N-1} \mathcal{L}_k + \mathcal{L}_0 \ ,}$$

- with:
  - $\mathcal{L}_N = \int_{\mathbb{R}^d} \log(p_N(x_N)/q(x_N|x_0)) q_{N|0}(x_N|x_0) \mathrm{d}x_N.$
  - $\mathcal{L}_k = \int_{\mathbb{R}^d} \log(p_{\theta,k|k+1}(x_k|x_{k+1})/q_{k|k+1,0}(x_k|x_{k+1}, x_0)) q_{k,k+1|0}(x_k, x_{k+1}|x_0) \mathrm{d}x_k.$
  - $\mathcal{L}_0 = \int_{\mathbb{R}^d} \log(p_{\theta,0|1}(x_0|x_1)) q_{1|0}(x_1|x_0) \mathrm{d}x_1.$
- What do we need on the **variational distribution** $q(x_{1:N}|x_0)$?
  - **Sampling**: $q_{k|0}$ is an **easy-to-sample** distribution.
  - **Tractability** of $q_{k|k+1,0}$.
- In **DDPM** we choose q by specifying **Gaussian transition** (discretization of the Ornstein-Ulhenbeck process) for $q_{k+1|k}$.

From DDPM to DDIM

- In **DDPM** we choose q by specifying **Gaussian transition** (discretization of the Ornstein-Ulhenbeck process) for $q_{k+1|k}$.
- This choice specifies $q_{k|0}$ (because $q_{k+1|k}$ is Gaussian with **linear** mean).
- Therefore we have
  $q_{k|k+1,0}(x_k|x_{k+1}, x_0) = q_{k+1|k}(x_{k+1}|x_k)q_{k|0}(x_k|x_0)/q_{k+1|0}(x_{k+1}|x_0)$.
- In **DDIM** we choose a different approach.
  - We specify $q_{k|k+1,0}(x_k|x_{k+1}, x_0) = \mathrm{N}(C_k x_{k+1} + D_k x_0, \bar{\sigma}_k^2)$.
  - $\{\bar{\sigma}_k\}_{k=1}^{N-1}$ will be **fixed by the user**.
  - The parameters $\{C_k\}_{k=1}^{N-1}$, $\{D_k\}_{k=1}^{N-1}$ are chosen such that $q_{k|0} = \mathrm{N}(\alpha_k x_0, \sigma_k^2)$.
- The **parameters** $\{\alpha_k\}_{k=0}^{N-1}$ and $\{\sigma_k\}_{k=0}^{N-1}$ match the ones of **DPPM**.
- We present here a derivation of **DDIM** inspired from Song et al. (2020).

DDIM derivation (1/2)

- We assume that there exist $C_k, D_k, \bar{\sigma}_k \geq 0$ such that

$$\boxed{q_{k|k+1,0}(x_k|x_{k+1}, x_0) \propto \exp[-\|x_k - (C_k x_{k+1} + D_k x_0)\|^2/(2\bar{\sigma}_k^2)] \ .}$$

- We assume that $q_{N|0}(x_N|x_0) = \mathrm{N}(\alpha_N x_0, \sigma_N^2)$.
- We are going to proceed by recursion. We assume that for any $j \in \{k+1, \dots, N\}$, $q_{j|0}(x_j|x_0) = \mathrm{N}(\alpha_j x_0, \sigma_j^2)$.
- **Question**: what **conditions** on $C_k, D_k, \bar{\sigma}_k$ should we impose so that $q_{k|0} = \mathrm{N}(\alpha_k x_0, \sigma_k^2)$?
- We are going to use that for any $x_k \in \mathbb{R}^d$

$$\boxed{q_{k|0}(x_k|x_0) = \int_{\mathbb{R}^d} q_{k|k+1,0}(x_k|x_{k+1}, x_0) q_{k+1|0}(x_{k+1}|x_0) \mathrm{d}x_{k+1} \ .}$$

  - **Computing** the integral.
  - **Finding** the parameter of the resulting Gaussian.

DDIM derivation (2/2)

- We have the following:
  - $X_k = C_k X_{k+1} + D_k X_0 + \bar{\sigma}_k \bar{Z}_k$,
  - $X_{k+1} = \alpha_{k+1} X_0 + \sigma_{k+1} Z_{k+1}$.
  - With $\bar{Z}_k, Z_{k+1} \sim \mathrm{N}(0, \mathrm{Id})$ independent.
- Hence, we get that

$$\begin{aligned} X_k &= (C_k \alpha_{k+1} + D_k) X_0 + \bar{\sigma}_k \bar{Z}_k + C_k \sigma_{k+1} Z_{k+1} \\ &= (C_k \alpha_{k+1} + D_k) X_0 + (\bar{\sigma}_k^2 + C_k^2 \sigma_{k+1}^2)^{1/2} Z_k \ . \end{aligned}$$

- Therefore, we obtain the following **parameters**:
  - $\alpha_k = C_k \alpha_{k+1} + D_k$.
  - $\sigma_k^2 = \bar{\sigma}_k^2 + C_k^2 \sigma_{k+1}^2$.
  - **Solving the system**: setting $\bar{\sigma}_k$ implies setting C_k implies setting D_k.
  - Choosing $\bar{\sigma}_k = (\alpha_{k+1|k}^2 / \sigma_{k+1|k}^2 + (1/\sigma_k^2))^{-1}$ gives **DDPM**.
- Recall that the **discretized Ornstein-Ulhenbeck** gives
  - $\alpha_k = (1-\gamma)^k$.
  - $\sigma_k^2 = (1 - (1-\gamma)^{2k})/(1 - \gamma/2)$.

Choice of the parameters

- Recall that we have:
  - $\alpha_k = C_k \alpha_{k+1} + D_k$.
  - $\sigma_k^2 = \bar{\sigma}_k^2 + C_k^2 \sigma_{k+1}^2$.
  - **Solving the system**: setting $\bar{\sigma}_k$ implies setting C_k implies setting D_k.
  - Choosing $\bar{\sigma}_k = (\alpha_{k+1|k}^2 / \sigma_{k+1|k}^2 + (1/\sigma_k^2))^{-1}$ gives **DDPM**.
- We can choose $\bar{\sigma}_k$ **arbitrary small**.
- Choosing $\bar{\sigma}_k \to 0$, we get **D**enoising **D**iffusion **I**implicit **M**odels (**DDIM**).
- We end up with a **deterministic** model.
  - Recall that when we train the model we estimate X_0, i.e. $\mathbf{t}_\theta(k+1, X_{k+1}) \approx X_0$.
  - The recursion at **sampling** times is then given by

$$\boxed{X_k = C_k X_{k+1} + \mathbf{t}_\theta(k+1, X_{k+1}) + \bar{\sigma}_k \bar{Z}_k \ .}$$

Interpolation with DDIM

- An alternative to **normalizing flows**.
- Like any **deterministic** method we can **interpolate**.

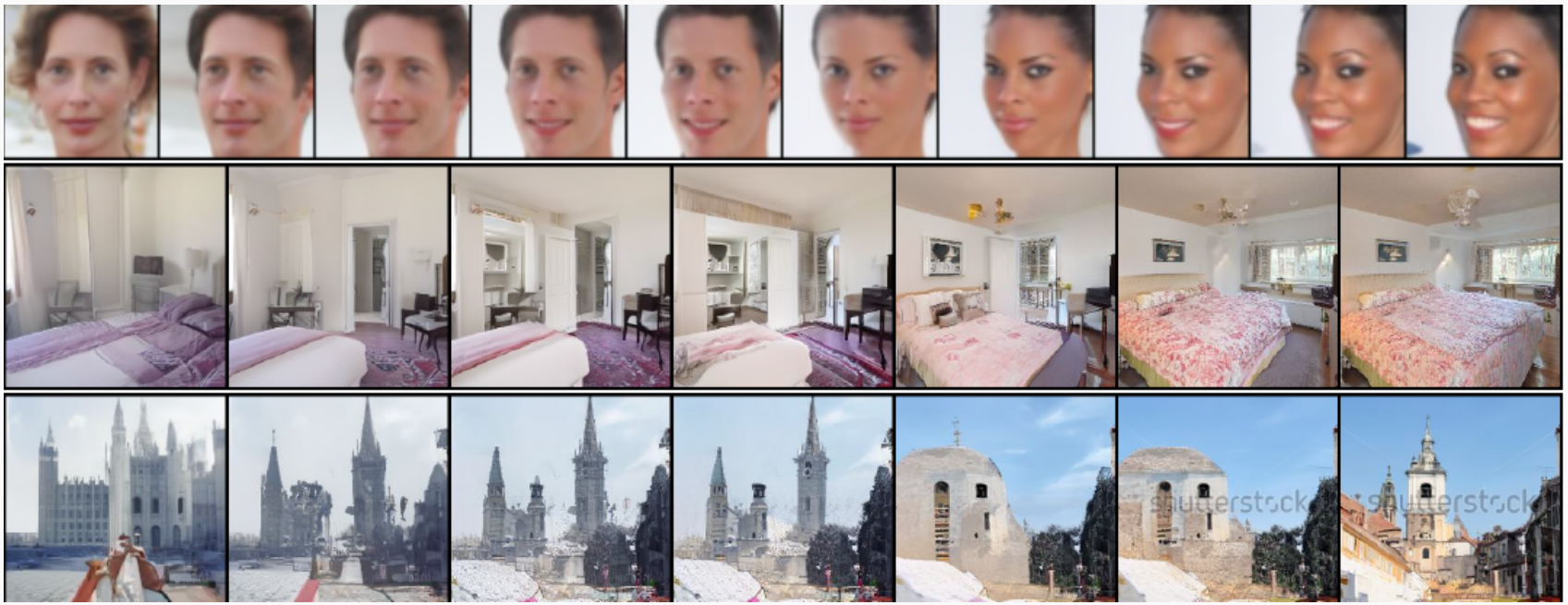
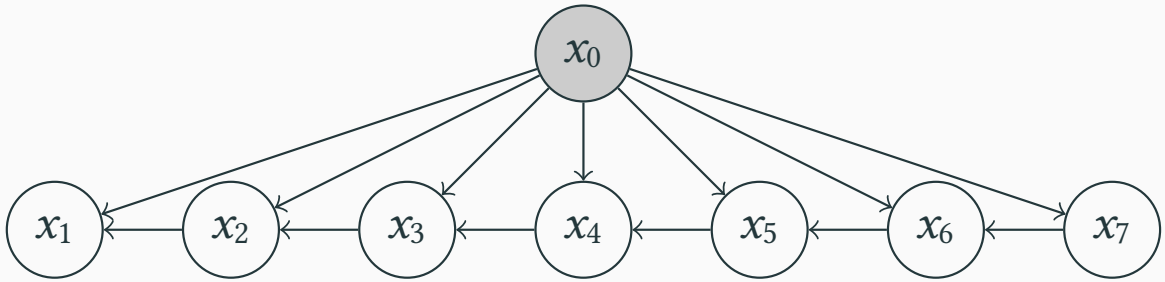


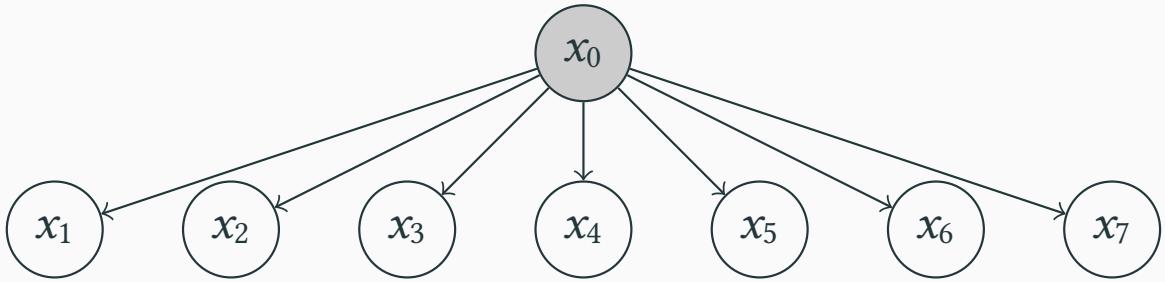
Figure 5: DDIM interpolation. Image extracted from Song et al. (2020).

Subsampling with DDIM

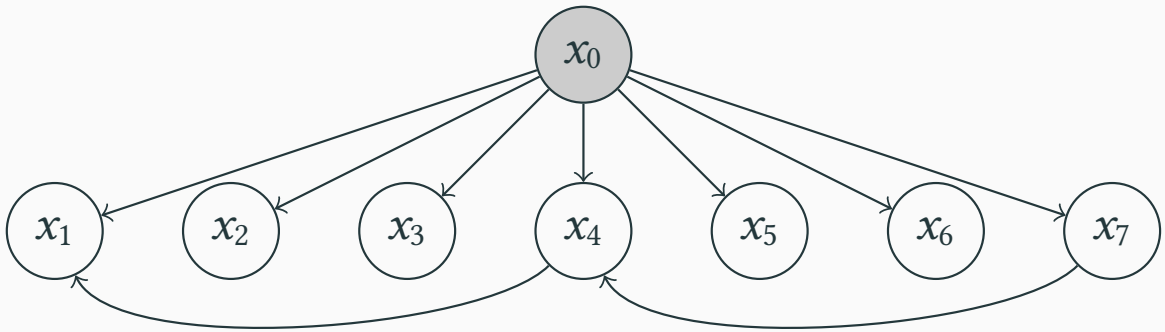
- **Chain model**: (classical **DDIM**/**DDPM**)



- **Star graph model**: (multiple **encoder**/**decoder**)



- **Subsampled model**:



Subsampled model

- In the **chain model** we parameterize $q(x_{1:N}|x_0)$ by

$$\boxed{q(x_{1:N}|x_0) = q_N(x_N|x_0) \prod_{k=1}^{N-1} q_{k|k+1,0}(x_k|x_{k+1}, x_0) \ .}$$

- The **backward chain** $p_\theta(x_{0:N})$ is then given by

$$\boxed{p_\theta(x_{0:N}) = p(x_N) \prod_{k=0}^{N-1} p_{\theta,k|k+1}(x_k|x_{k+1}) \ .}$$

- In the **subsampled model** we parameterize $q(x_{1:N}|x_0)$ by

$$\boxed{q(x_{1:N}|x_0) = \prod_{i=1}^{M-1} q_{k_i|k_{i+1},0}(x_{k_i}|x_{k_{i+1}}, x_0) \prod_{k\in\mathcal{N}\setminus\mathcal{M}} q_{k|0}(x_k|x_0) \ ,}$$

- where $\mathcal{N} = \{0, \dots, N-1\}$ and $\mathcal{M} = \{k_1, \dots, k_{M-1}\}$ with $k_M = N$
- The **backward chain** $p_\theta(x_{0:N})$ is then given by

$$\boxed{p_\theta(x_{0:N}) = p(x_N) \prod_{i=1}^{M-1} p_{\theta,k_i|k_{i+1}}(x_{k_i}|x_{k_{i+1}}) \prod_{k\in\mathcal{N}\setminus\mathcal{M}} \tilde{p}_{\theta,0|k}(x_0|x_k) \ .}$$

Susbsampled ELBO

- The **backward chain** $p_\theta(x_{0:N})$ is then given by

$$\boxed{p_\theta(x_{0:N}) = p(x_N) \prod_{i=1}^{M-1} p_{\theta,k_i|k_{i+1}}(x_{k_i}|x_{k_{i+1}}) \prod_{k \in \mathcal{N}\setminus\mathcal{M}} \tilde{p}_{\theta,0|k}(x_0|x_k)\ .}$$

- Recall that $\log(p_\theta(x_0)) \geq \mathcal{L} = \int_{(\mathbb{R}^d)^N} \log(q(x_{1:N}|x_0)/p_\theta(x_{0:N}))q(x_{1:N}|x_0)\mathrm{d}x_{1:N}$.
- We use the **backward decomposition** of $q(x_{1:N}|x_0)$ and we get

$$\boxed{\mathcal{L} = \mathcal{L}_N + \sum_{i=1}^{M-1} \mathcal{L}_i + \sum_{k \in \mathcal{N}\setminus\mathcal{M}} \mathcal{L}_k\ ,}$$

- with:
  - $\mathcal{L}_N = \int_{\mathbb{R}^d} \log(p_N(x_N)/q(x_N|x_0))q_{N|0}(x_N|x_0)\mathrm{d}x_N$.
  - The **backward terms**:

$\mathcal{L}_i = \int_{\mathbb{R}^d} \log(p_{\theta,k_i|k_{i+1}}(x_{k_i}|x_{k_i+1})/q_{k_i|k_{i+1},0}(x_{k_i}|x_{k_{i+1}}, x_0))q_{k_i,k_{i+1}|0}(x_{k_i}, x_{k_{i+1}}|x_0)\mathrm{d}x_{k_i}$.

  - The **decoder terms**: $\mathcal{L}_k = \int_{\mathbb{R}^d} \log(\tilde{p}_{\theta,0|k}(x_0|x_k))q_{k|0}(x_k|x_0)\mathrm{d}x_k$.

DDIM results

	S	CIFAR10 (32 × 32) 10	20	50	100	1000	CelebA (64 × 64) 10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$		367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

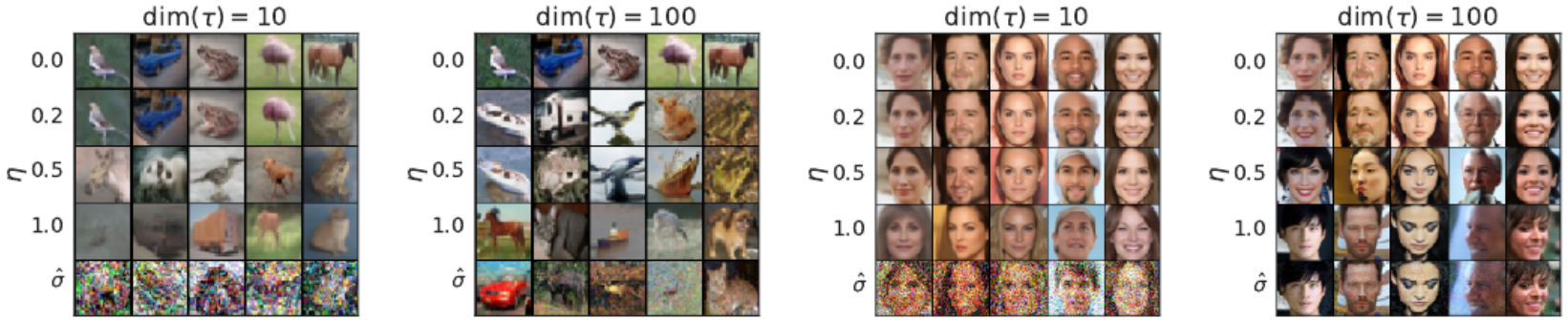


Figure 6: DDIM results. Image extracted from Song et al. (2020).

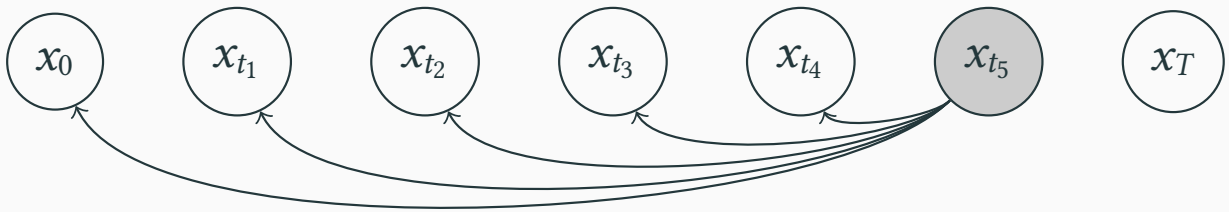
Dynamic programming and stepsize selection

Choosing the stepsizes

- Given a sequence of steps $\mathcal{T} = \{t_0 = 0, t_1, \dots, t_N = T\}$ and a budget of K steps which **subsampled** sequence is the **best**?
- The "best" in a **variational sense**.
- We recall that in the variational setting we set $p_{\theta^\star, t_i | t_{i+1}}(x_{t_i} | x_{t_{i+1}}) = q_{t_i | t_{i+1}, 0}(x_{t_i} | x_{t_{i+1}}, \mathbf{t}_{\theta^\star}(t_{i+1}, x_{t_{i+1}}))$.
- $\mathbf{t}_{\theta^\star}(t_{i+1}, \cdot)$ is trained with the loss

$$\boxed{\ell_{t_{i+1}}(\theta) = \mathbb{E}[\|\mathbf{t}_\theta(t_{i+1}, X_{t_{i+1}}) - X_0\|^2] \ .}$$

- **Take-home** message: *all* $p_{\theta^\star, t_i | t_{i+1}}(x_{t_i} | x_{t_{i+1}})$ with $t_i < t_{i+1}$ can be computed using $\mathbf{t}_{\theta^\star}(t_{i+1}, x_{t_{i+1}})$.



Choosing the best subsample

- Given a **subsample** $t_{k_0} = 0 < t_{k_1} < \cdots < t_{k_K} = T$ ($\mathcal{K} = \{t_{k_0}, \ldots, t_{k_K}\}$) we have a **variational lower bound**

$$\boxed{\mathcal{L}_{\mathcal{K}} = \mathcal{L}_T + \sum_{i=1}^{K-1} \mathcal{L}_{t_{k_{i+1}}, t_{k_i}} + \mathcal{L}_{t_{k_1}, 0} \ .}$$

- where:
  - $\mathcal{L}_T = \int_{\mathbb{R}^d} \log(p_T(x_T)/q_{T|0}(x_T|x_0)) q(x_T|x_0) \mathrm{d}x_T$.
  - $\mathcal{L}_{t_{k_{i+1}}, t_{k_i}} = \int_{\mathbb{R}^d} \log(p_{\theta, t_{k_i}|t_{k_{i+1}}}(x_{t_{k_i}}|x_{t_{k_{i+1}}})/q(x_{t_{k_i}}|x_{t_{k_{i+1}}}, x_0)) q(x_{t_{k_i}}|x_{t_{k_{i+1}}}, x_0) \mathrm{d}x_{t_{k_i}}$.
  - $\mathcal{L}_{t_{k_1}, 0} = \int_{\mathbb{R}^d} \log(p_{\theta, 0, t_{k_1}}(x_0|x_{t_{k_1}})) q(x_{t_{k_1}}|x_0) \mathrm{d}x_{t_{k_1}}$.
- To compute $\{\mathcal{L}_{t,s} \ : \ s, t \in \mathcal{T}, \ s < t\}$, we only need $\mathcal{T}$ **forward call** to the network $\mathbf{t}_{\theta^\star}$.
- A **budget** of K steps, which **subsample** $\mathcal{L}_{\mathcal{K}}$ yields the **higher variational bound**?
- If $|\mathcal{K}| \ll |\mathcal{T}|$ then sampling time is **heavily reduced**.

Dynamic programming

- Create matrices $\mathrm{C} \in \mathbb{R}^{K+1 \times N+1}$, $\mathrm{D} \in \mathbb{R}^{K+1 \times N+1}$ (recall that $|\mathcal{K}| = K+1$ and $|\mathcal{T}| = N+1$)
- where:
  - $\mathrm{C}(m, n)$ is the **minimal cost** to reach t_{n-1} in m steps, i.e.

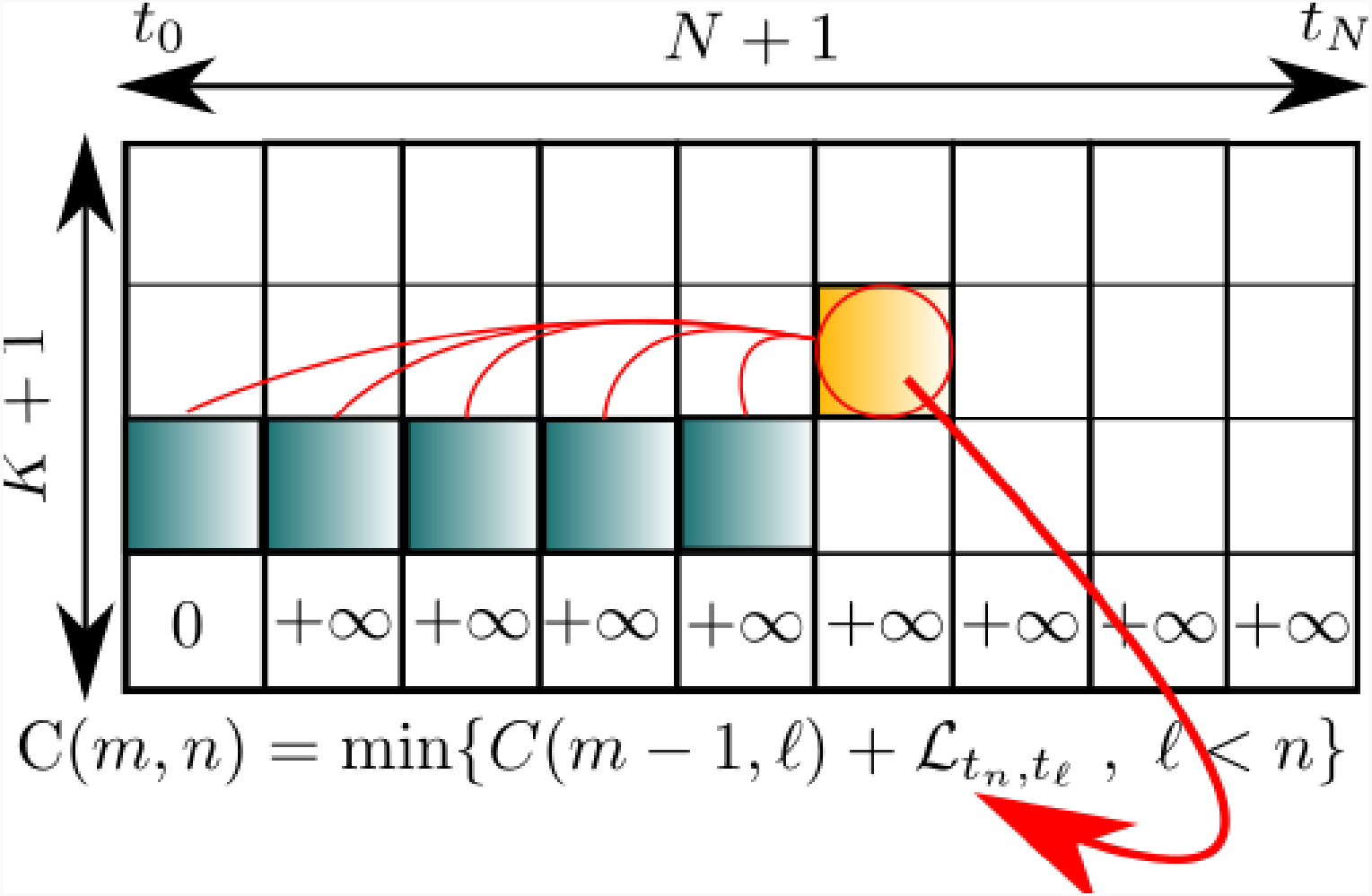
$$\boxed{\mathrm{C}(m,n) = \min\{\textstyle\sum_{i=1}^{m-1} \mathcal{L}_{t_{k_{i+1}}, t_{k_i}} + \mathcal{L}_{t_{k_1}, 0} \ : \ \mathcal{K} = \{t_{k_0} = 0, \ldots, t_{k_m} = t_{n-1}\}\} \ .}$$

  - $\mathrm{D}(m, n) = k_{m-1}$ in the decomposition $\mathcal{K} = \{t_{k_0} = 0, t_{k_1}, \ldots, t_{k_{m-1}}, t_{k_m} = t_n\}$ which minimizes the **previous cost**.
- We can **fill the lines** recursively using that

$$\mathrm{C}(m,n) = \min\{C(m-1,\ell) + \mathcal{L}_{t_n, t_\ell} \ : \ \ell \in \{0, \ldots, N\}, \ \ell < n\} \ ,$$
$$\mathrm{D}(m,n) = \arg\min\{C(m-1,\ell) + \mathcal{L}_{t_n, t_\ell} \ : \ \ell \in \{0, \ldots, N\}, \ \ell < n\} \ .$$

- Then, the **optimal decomposition** is given by $\mathcal{K} = \{t_{k_0}, \ldots, t_{k_K}\}$, where for any $i \in \{0, \ldots, K-1\}$, $k_{K-1-i} = D(K-i, k_{K-i})$, with $k_K = N$.
- This is a **dynamic programming** (bottom-up) procedure Dijkstra et al. (1959).

An illustration of the bottom-up procedure



Subsampling results

- Running the **dynamic program** we find the following decomposition

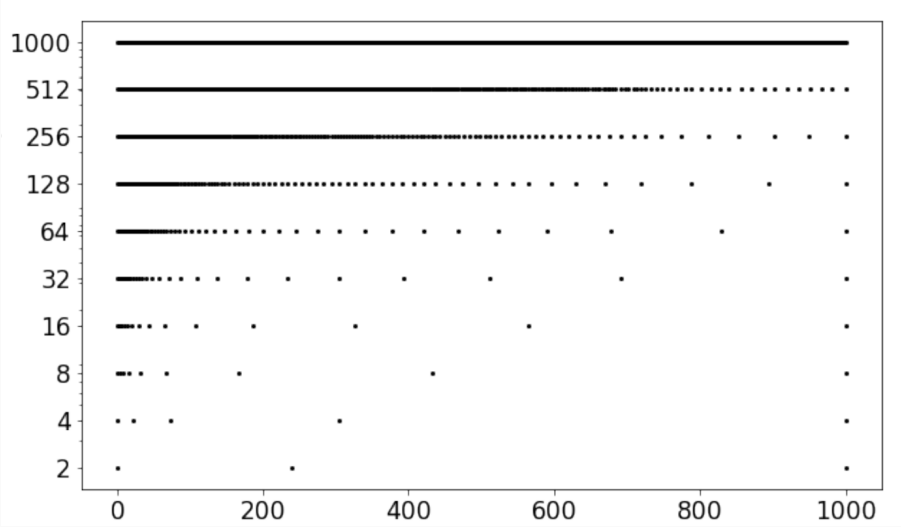


Figure 7: Budget of stepsizes. Image extracted from Watson et al. (2021).

- **Consequences**:
  - **More stepsize** near the **data distribution**.
  - **Larger stepsizes** near the **easy-to-sample** distribution.

Limitations and Extensions

- **Optimizing** the **ELBO** does *not* always correlate with better **FID**.

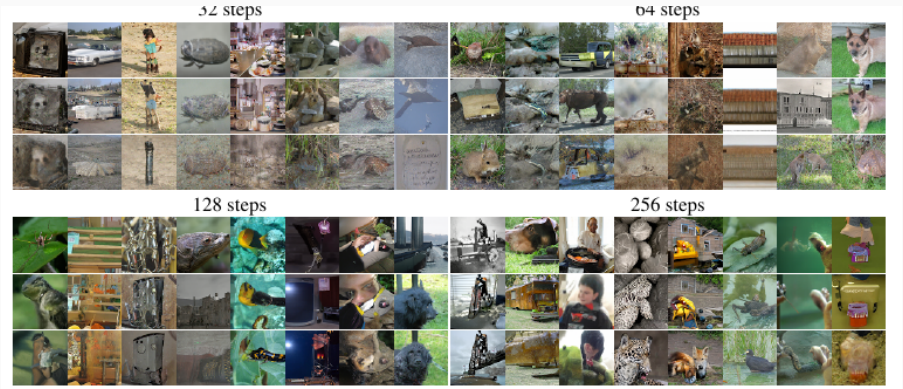


Figure 8: Top and middle are samples (Imagenet) obtained using preset stride strategies. Bottom is the discovering strategy from Watson et al. (2021). Image extracted from Watson et al. (2021).

- Improvement Watson et al. (2022) by differentiating through sample quality (Differentiable Diffusion Sampler Search).
- **Differenciation** through the sampling process. **DDSS** optimize the **Kernel Inception Distance** Bińkowski et al. (2018).

Knowledge distillation

Teacher/Student and progressive distillation

- First train a **teacher** diffusion model. Then train a **student** diffusion model.
- We follow Salimans and Ho (2022)

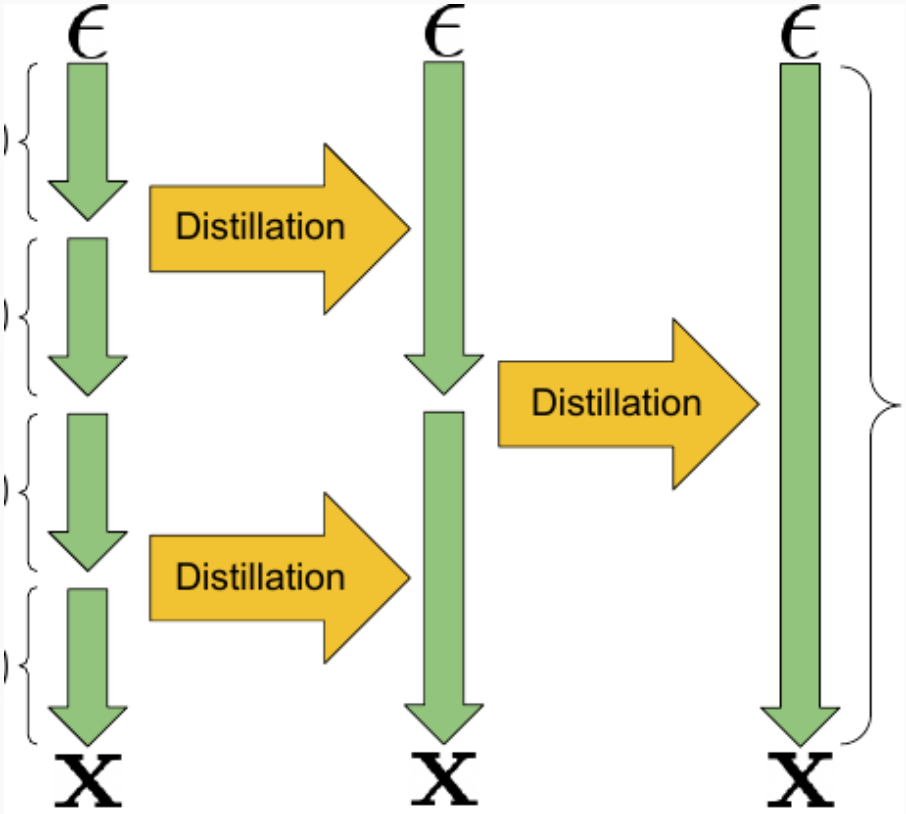


Figure 9: Progressive distillation. Image extracted from Salimans and Ho (2022).

Progressive distillation algorithm

Algorithm 1 Standard diffusion training

Require: Model $\hat{\mathbf{x}}_\theta(\mathbf{z}_t)$ to be trained
Require: Data set $\mathcal{D}$
Require: Loss weight function $w()$

while not converged **do**
  $\mathbf{x} \sim \mathcal{D}$ ▷ Sample data
  $t \sim U[0, 1]$ ▷ Sample time
  $\epsilon \sim N(0, I)$ ▷ Sample noise
  $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$ ▷ Add noise to data

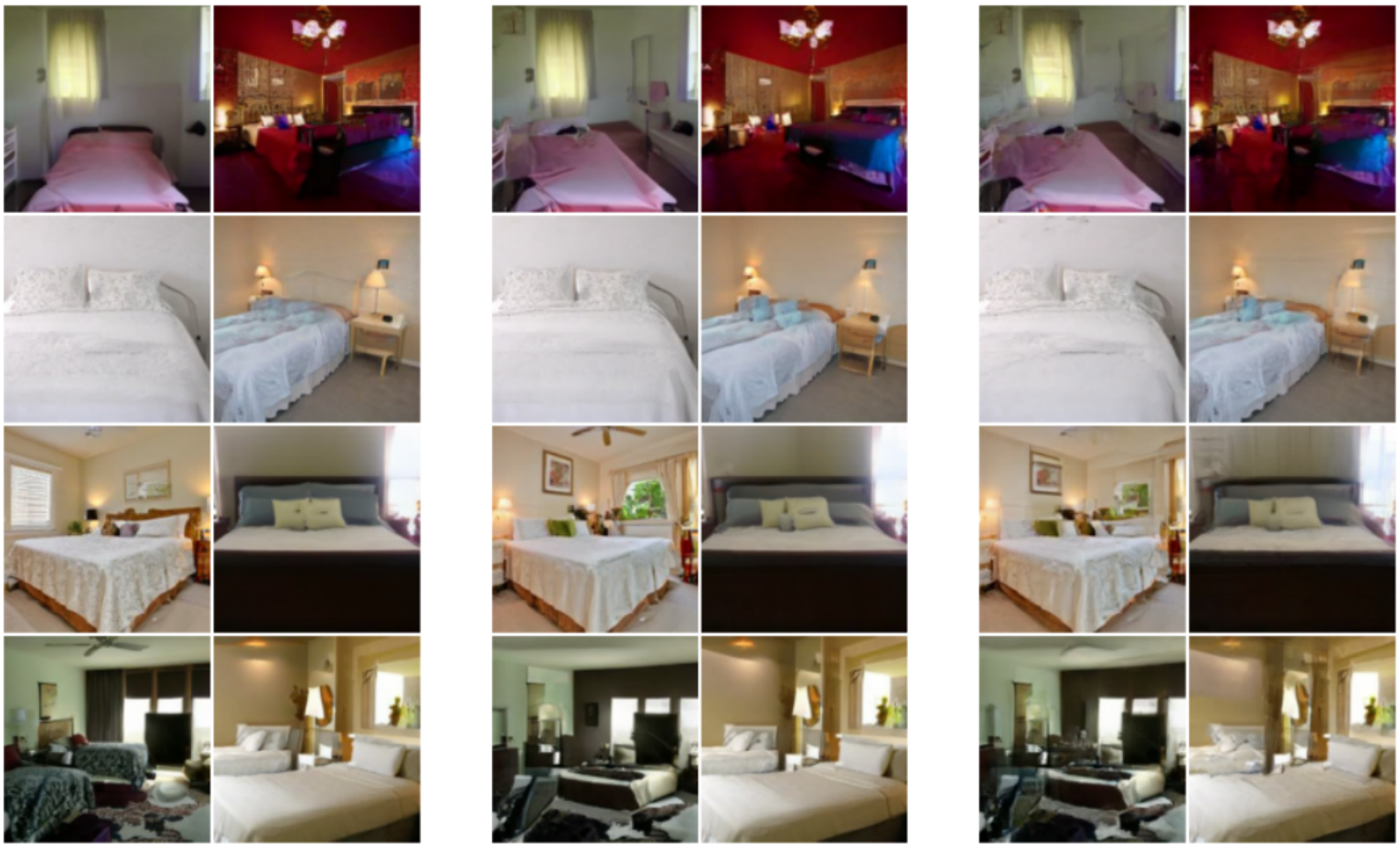
  $\tilde{\mathbf{x}} = \mathbf{x}$ ▷ Clean data is target for $\hat{\mathbf{x}}$
  $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$ ▷ log-SNR
  $L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$ ▷ Loss
  $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$ ▷ Optimization
end while

Algorithm 2 Progressive distillation

Require: Trained teacher model $\hat{\mathbf{x}}_\eta(\mathbf{z}_t)$
Require: Data set $\mathcal{D}$
Require: Loss weight function $w()$
Require: Student sampling steps N
for K iterations **do**
  $\theta \leftarrow \eta$ ▷ Init student from teacher
  while not converged **do**
    $\mathbf{x} \sim \mathcal{D}$
    $t = i/N, \quad i \sim Cat[1, 2, \ldots, N]$
    $\epsilon \sim N(0, I)$
    $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$
    `# 2 steps of DDIM with teacher`
    $t' = t - 0.5/N, \quad t'' = t - 1/N$
    $\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t}(\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$
    $\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}}(\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$
    $\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t)\mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t)\alpha_t}$ ▷ Teacher $\hat{\mathbf{x}}$ target
    $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$
    $L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$
    $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$
  end while
  $\eta \leftarrow \theta$ ▷ Student becomes next teacher
  $N \leftarrow N/2$ ▷ Halve number of sampling steps
end for

Figure 10: Progressive distillation algorithm. Image extracted from Salimans and Ho (2022).

Some results



(a) 256 sampling steps (b) 4 sampling steps (c) 1 sampling step

Figure 11: From 256 to 1 sampling steps. Image extracted from Salimans and Ho (2022).

Some extensions

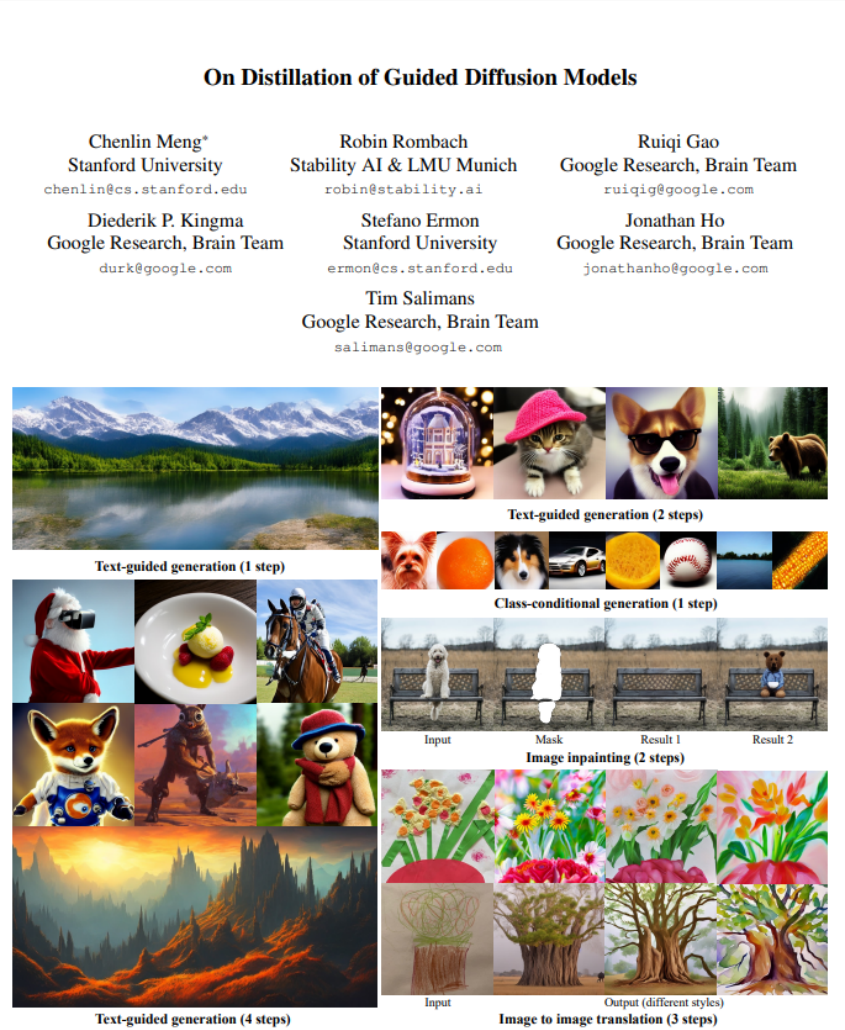


Figure 12: Distillation for guided model. Image extracted from Meng et al. (2022).

- Other distillation methods Luhman and Luhman (2021); Meng et al. (2022); Song et al. (2023).

Conclusion

Conclusion

- We have introduced **diffusion models**.
  - diffusion models in **continuous time** and results.
  - **Normalizing flows** and **Likelihood computation**.
  - **Acceleration** of diffusion models.

References

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Dominique Bakry, Ivan Gentil, Michel Ledoux, et al. *Analysis and geometry of Markov diffusion operators*, volume 103. Springer, 2014.

Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

Patrick Cattiaux, Giovanni Conforti, Ivan Gentil, and Christian Léonard. Time reversal of diffusion processes under a finite entropy condition. *arXiv preprint arXiv:2104.07708*, 2021.

Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. *arXiv preprint arXiv:2302.07194*, 2023.

References ii

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):651–676, 2017.

Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *arXiv preprint arXiv:2208.05314*, 2022.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modeling. *NeurIPS*, 2022.

Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

References iii

Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *Ann. Appl. Probab.*, 27(3):1551–1587, 2017. ISSN 1050-5164. doi: 10.1214/16-AAP1238. URL https://doi.org/10.1214/16-AAP1238.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Ulrich G Haussmann and Etienne Pardoux. Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205, 1986.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Nobuyuki Ikeda and Shinzo Watanabe. *Stochastic differential equations and diffusion processes*. Elsevier, 2014.

Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

References iv

Valentin Khrulkov and Ivan Oseledets. Understanding ddpm latent codes through optimal transport, 2022.

Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence for score-based generative modeling with polynomial complexity. *arXiv preprint arXiv:2206.06227*, 2022.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds, 2022.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022.

Eliya Nachmani, Robin San Roman, and Lior Wolf. Non gaussian denoising diffusion models. *arXiv preprint arXiv:2106.07582*, 2021.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

References v

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023. doi: 10.48550/ARXIV.2303.01469. URL https://arxiv.org/abs/2303.01469.

Daniel W Stroock and SR Srinivasa Varadhan. *Multidimensional diffusion processes*, volume 233. Springer Science & Business Media, 1997.

References vi

Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021.

Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality, 2022.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.