An introduction to Score-based Generative Models
Lecture 2: Introduction to score-based generative models

Giovanni Conforti, Alain Durmus
École polytechnique
with the help of Valentin De Bortoli, Marta Gentiloni-Silveri, Emmanuel Gobet, Yazid Janati, Éric Moulines, Maxence Noble, Tom Sander, and many others...

February 21, 2024

Outline

- **Goal of today's course**:
  - Score matching / Denoising score matching
  - Introduce SGM with time-reversal (without relying on stochastic calculus).

- **Outline of the course**:
  - Introduction of SGM with discrete-time reversal following Sohl-Dickstein et al. (2015).
  - Introduction of SGM with variational approaches following Ho et al. (2020).

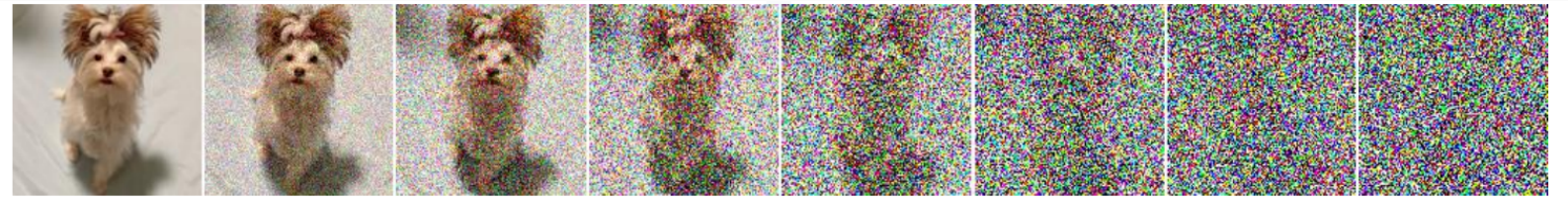

Figure 1: Noising process in SGM. Image extracted from Song et al. (2020b).

Score matching and denoising score matching

Reminder on EBM

- Consider the case $\lambda = \mathrm{Leb}$ and $\mathsf{X} = \mathbb{R}^d$.
- EBM consists in defining a family $\{\mu_\theta \,:\, \theta \in \Theta\}$ directly from a family of potential/energy functions $\{U_\theta \,:\, \theta \in \Theta\}$: for $x \in \mathbb{R}^d$

$$p_\theta(x) = (\mathrm{d}\mu_\theta/\mathrm{dLeb})(x) = \exp[-U_\theta(x)]/\mathfrak{Z}(\theta) \,,$$
$$\mathfrak{Z}(\theta) = \int_{\mathbb{R}^d} \exp[-U_\theta(\tilde{x})]\mathrm{d}\tilde{x} \,.$$

- U_θ is typically a **neural network** ($\theta \in \Theta$ is a set of parameters).
- We saw in the last course how to train an EBM based on maximum likelihood estimation (MLE).

Score matching for EBM

- An alternative to MLE is score matching (SM)
- It consists in minimizing the Fisher divergence:

$$\mathbf{D}_F(\mu^\star|\mu_\theta) = \int \|\nabla_x \log p_\theta - \nabla_x \log p^\star\|^2 \, \mathrm{d}\mu^\star$$

- Supposing $p^\star \ll \mathrm{Leb}$ **with a smooth density** $p^\star$
- General principle in statistics not restrained on EBM... Hyvärinen (2005)

Score matching for EBM

- SM consists in minimizing the Fisher divergence:

$$\mathbf{D}_F(\mu^\star|\mu_\theta) = \int \|\nabla_x \log p_\theta - \nabla_x \log p^\star\|^2 \, \mathrm{d}\mu^\star$$

- Problem: we do not have access to $\nabla_x \log p^\star$...
- How we can show that **minimizing $\theta \mapsto \mathbf{D}_F(\mu^\star|\mu_\theta)$ is equivalent to a more tractable problem.**

Score matching

- SM consists in minimizing (w.r.t. θ) the Fisher divergence:

$$\mathbf{D}_F(\mu^\star|\mu_\theta) = \int \|\nabla_x \log p_\theta - \nabla_x \log p^\star\|^2 \, \mathrm{d}\mu^\star$$

- Problem: we do not have access to $\nabla_x \log p^\star$...
- It is equivalent to minimizing (w.r.t. θ)

$$\int \|\nabla_x \log p_\theta\|^2 \, \mathrm{d}\mu^\star + 2 \int \Delta_x \log p_\theta \mathrm{d}\mu^\star$$

- An empirical counterpart of the above functions is:

$$N^{-1} \sum_{i=1}^{N} \{ \left\| \nabla_x \log p_\theta(x^i) \right\|^2 + \Delta_x \log p_\theta(x^i) \} \ ,$$

where $\{x^i\}_{i=1}^N$ are observations of $\mu^\star$.

Denoising Score Matching

- In case $\mu^\star$ do not admit a smooth density
- We can consider instead $\mu_\varepsilon^\star = \mu^\star * \varphi_\varepsilon$, where φ_ε is the density of $\mathrm{N}(0, \varepsilon\,\mathrm{Id})$
- It has the smooth density:

$$p_\varepsilon^\star(y) = \int \mathrm{d}\mu^\star(x)\varphi_\varepsilon(y-x)\ .$$

- It turns out that minimizing (w.r.t. θ) $\mathbf{D}_F(\mu_\varepsilon^\star|\mu_\theta)$ is equivalent to minimizing

$$\int \mathrm{d}\mu^\star \varphi_\varepsilon(y-x)\left\|\nabla_x \log \varphi_\varepsilon(y-x) - \nabla_x \log p_\theta(y)\right\|^2\ .$$

Discrete time-reversal and score-based generative modeling

Outline of the section

- In this section we introduce SGM in a **"direct" manner**.
- A bit of "history":
  - First paper Sohl-Dickstein et al. (2015).
  - First successful application Song and Ermon (2019).
  - Concurrently (variational approach) Ho et al. (2020).
- We present some **techniques** to train SGM.
- In what follows:
  - **Time-reversal in discrete-time**.
  - Links with **annealed Langevin**.
  - **A (very) few Implementation details and tricks**.

Discrete-time

Principles of SGM

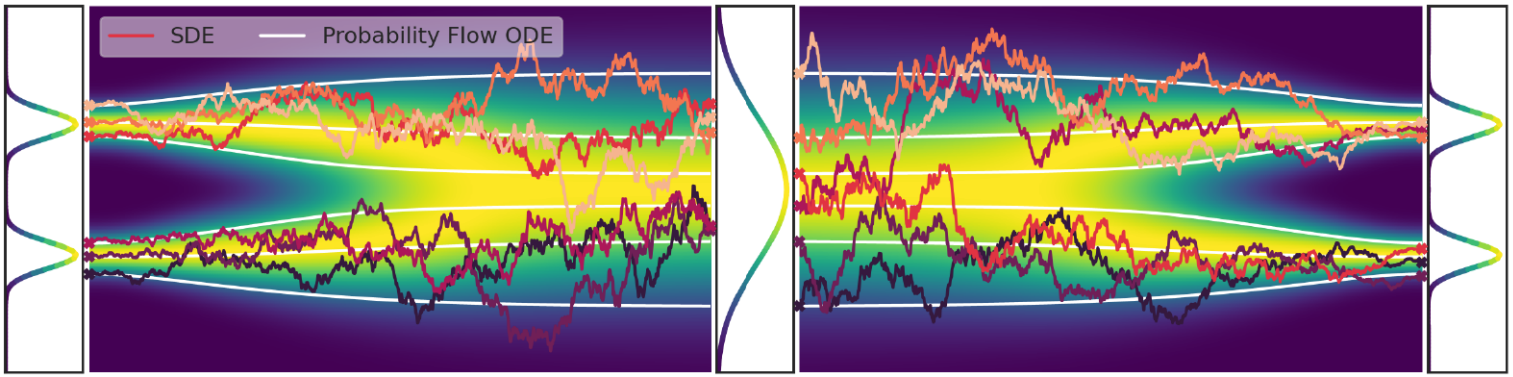

Figure 2: Noising and generative processes in SGM. Image extracted from Song et al. (2020b).

- **Interpolating** between two distributions:
  - The data distribution is denoted $\mu^\star \in \mathcal{P}(\mathbb{R}^p)$ with density $p^\star$.
  - The easy-to-sample distribution is denoted $\nu_0 \in \mathcal{P}(\mathbb{R}^p)$ with density q_0.
  - ν_0 is usually the standard multivariate Gaussian distribution.
- Going from the data to the easy-to-sample distribution: **noising process**.
- Going from the easy-to-sample to the data distribution: **generative process**.
- How to **invert** the forward noising process?

A Markov model

- **First idea**:
  - **Progressively going from ν_0 to $\mu^\star$ in $n_s \in \mathbb{N}^*$ steps**,
  - defining $\{X_i\}_{i=0}^{n_s}$ such that $X_0 \sim \mu^\star$ and $X_{n_s} \sim \nu_0$.
  - How to define $\{X_i\}_{i=0}^{n_s}$?
- **Second idea**:
  - Consider $\{X_i\}_{i=0}^{n_s}$ **as a non-homogeneous Markov chain** starting from $X_0 \sim \mu^\star$,
  - using a sequence of transition densities $\{p_{k+1|k}\}_{k=0}^{N-1}$ corresponding to $X_{k+1}|X_k \sim p_{k+1|k}(\cdot|X_k)$,
  - and **which converges to ν_0**: X_{n_s} has distribution $\approx \nu_0$.
- $(X_0, \dots X_{n_s})$ has for density for any $x_{0:n_s} = \{x_k\}_{k=0}^{n_s}$

$$\boxed{p_{0:n_s}(x_{0:n_s}) = p_0(x_0) \prod_{k=0}^{n_s-1} p_{k+1|k}(x_{k+1}|x_k)\ ,}$$

  denoting for simplicity $p^\star$ by p_0.
- $(X_0, \dots X_{n_s})$ is called the noising process.

Ancestral/backward sampling I

- $(X_0, \dots X_{n_s})$ has for density for any $x_{0:n_s} = \{x_k\}_{k=0}^{n_s}$

$$\boxed{p_{0:n_s}(x_{0:n_s}) = p_0(x_0) \prod_{k=0}^{n_s-1} p_{k+1|k}(x_{k+1}|x_k)\ ,}$$

  denoting for simplicity $p^\star$ by p_0.

- Define the marginal associated with X_k:

$$p_k(x_k) = \ ?$$

- Let us pretend that the distribution of X_{n_s}, p_{n_s}, **is approximately** ν_0
- Question: starting from a sample Y_0 from $p_{n_s} \approx \nu_0$, can we obtain a sample from $\mu^\star$?
- Alternative/equivalent question: **can we sample from** $p_{0:n_s}$ **starting from** $p_{n_s} \approx \nu_0$**?**

Ancestral/backward sampling II

- $(X_0, \dots X_{n_s})$ has for density for any $x_{0:n_s} = \{x_k\}_{k=0}^{n_s}$

$$\boxed{p_{0:n_s}(x_{0:n_s}) = p_0(x_0) \prod_{k=0}^{n_s-1} p_{k+1|k}(x_{k+1}|x_k) \ ,}$$

  denoting for simplicity $p^\star$ by p_0.
- Define the marginal associated with X_k: p_k.
- Alternative/equivalent question: **can we sample from $p_{0:n_s}$ starting from $p_{n_s} \approx \nu_0$?**
- Main remark: ?

Ancestral/backward sampling II

- $(X_0, \dots X_{n_s})$ has for density for any $x_{0:n_s} = \{x_k\}_{k=0}^{n_s}$

$$\boxed{p_{0:n_s}(x_{0:n_s}) = p_0(x_0) \prod_{k=0}^{n_s-1} p_{k+1|k}(x_{k+1}|x_k)\ ,}$$

  denoting for simplicity $p^\star$ by p_0.
- Define the marginal associated with X_k: p_k.
- Alternative/equivalent question: **can we sample from $p_{0:n_s}$ starting from $p_{n_s} \approx \nu_0$?**
- Main remark: ?
- Idea: **sample iteratively $Y_{k+1}|Y_k$ with conditional distribution $X_k|X_{k+1}$!**

Ancestral/backward sampling II

- $(X_0, \dots X_{n_s})$ has for density for any $x_{0:n_s} = \{x_k\}_{k=0}^{n_s}$

$$\boxed{p_{0:n_s}(x_{0:n_s}) = p_0(x_0) \prod_{k=0}^{n_s-1} p_{k+1|k}(x_{k+1}|x_k) \ ,}$$

  denoting for simplicity $p^\star$ by p_0.

- Alternative/equivalent question: **can we sample from $p_{0:n_s}$ starting from $p_{n_s} \approx \nu_0$?**
- Main remark: the **backward** decomposition

$$\boxed{p_{0:n_s}(x_{0:n_s}) = p_{n_s}(x_{n_s}) \prod_{k=0}^{n_s-1} p_{k|k+1}(x_k|x_{k+1}) \ .}$$

- **Questions**:
  - Choice for $\{p_{k+1|k}\}_{k=0}^{n_s-1}$?
  - Estimation of $\{p_{k|k+1}\}_{k=0}^{n_s-1}$?

Choice for $\{p_{k+1|k}\}_{k=0}^{n_s-1}$

- How do we go from the data distribution to the easy-to-sample distribution?
  - Take inspiration from **autoregressive process**:

$$X_{k+1} = \alpha X_k + \sqrt{1-\alpha^2} Z_{k+1}$$

    for $\{Z_k\}_{k\in\mathbb{N}}$ i.i.d. $\mathrm{N}(0, \mathrm{Id})$ Gaussian and $\alpha < 1$.
  - $(X_k)_{k\in\mathbb{N}^*} \to \mathrm{N}(0, \mathrm{Id})$ exponentially fast as.
  - **Ornstein-Ulhenbeck** process (continuous counterpart of AR):

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

  - **Euler-Maruyama** discretization: $X_{k+1} = (1-\gamma)X_k + \sqrt{2\gamma} Z_{k+1}$, $\gamma > 0$ is the stepsize.
  - **Euler-Maruyama** discretization of the **Ornstein-Ulhenbeck** process converges **exponentially** fast towards $\mathrm{N}(0, \mathrm{Id}/(1-\gamma/2))$ if $\gamma < 1$.

Inverting the noising process (1/3)

- Now let us try to **invert** the forward noising process
- **Difficulty comes from the initial distribution $\mu^\star$**
- If $\mu^\star$ is Gaussian, $(X_k)_{k=0}^{n_s}$ is a Gaussian vector: $p_{k|k+1}(x_k|x_{k+1})$ is a Gaussian density
- Can we still have a **Gaussian approximation**?
- Let us try to approximate γ small

$$p_{k|k+1}(x_k|x_{k+1}) = \ ?$$

Inverting the noising process (2/2)

- Hence, we get that

$$p_{k|k+1}(\cdot|x_{k+1}) \approx \mathrm{N}(\cdot; x_{k+1} + \gamma\{x_{k+1} + 2\nabla \log p_k(x_{k+1})\}, 2\gamma\,\mathrm{Id}) \ .$$

  - The **approximation** is up to a term of order γ in the exponential.
- **Sampling** from the **backward chain**: $Y_0 \sim \nu_0 = \mathrm{N}(0, \mathrm{Id}\,/(1-\gamma/2))$

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

- $\nabla \log p_k$ is **untractable**. We are going to approximate this term.

Score-matching (1/4)

- Recall
$$p_k(x_k) = \int p_{0:k-1}(x_{0:k}) \mathrm{dLeb}(x_0 \dots x_{k-1}) \ .$$
- The term $\nabla \log p_k$ is the called the **(Stein) score**.
- Literature on **score matching**: Hyvärinen (2005); Vincent (2011)
- We have the Fisher identity; see e.g., Efron (2011)

$$\nabla \log p_k(x_k) = \ ?$$

Score-matching (1/4)

- Recall
$$p_k(x_k) = \int p_{0:k-1}(x_{0:k}) \mathrm{dLeb}(x_0 \dots x_{k-1}) \ .$$
- The term $\nabla \log p_k$ is the called the **(Stein) score**.
- Literature on **score matching**: Hyvärinen (2005); Vincent (2011)
- We have the Fisher identity; see e.g., Efron (2011)

$$\nabla \log p_k(x_k) = \ ?$$

- An **intermediate expression**:
  - $\nabla \log p_{k|0}(x_k|x_0)$ is tractable (forward transition).
  - The **conditional expectation** is not (backward conditional).
- We are going to use the property of the conditional expectation to obtain a **loss function**.

Score matching (2/4)

- We have

$$\boxed{\nabla \log p_k(X_k) = \mathbb{E}[\nabla \log p_{k|0}(X_k|X_0)|X_k]\ .}$$

- Using properties of the **conditional expectation** we have

$$\nabla \log p_k = \ ?$$

Score matching (2/4)

- We use the following properties of the **conditional expectation**:
- $Y = \mathbb{E}[X\,|U]$ if $Y = f(U)$, with $f = \arg\min\{\mathbb{E}[\|X - f(U)\|^2] \,:\, f \in \mathrm{L}^2(U)\}$.
- Recall that we have

$$\boxed{\nabla \log p_k(X_k) = \mathbb{E}[\nabla \log p_{k|0}(X_k|X_0)|X_k]\,.}$$

- Using the previous property we have

$$\boxed{\nabla \log p_k = \arg\min\{\mathbb{E}[\|f(X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2] \,:\, f \in \mathrm{L}^2(p_k)\}\,.}$$

- We obtain a **loss function**:
  - $\nabla \log p_{k|0}(x_k|x_0)$ is tractable (forward transition).
  - The expectation can be approximated with **Monte Carlo** (joint distribution).
- Note that this is valid for $k \in \{0, \ldots, n_s - 1\}$.

Score matching (3/4)

- Recall that the loss function is given by

$$\boxed{\nabla \log p_k = \arg\min\{\mathbb{E}[\|f(X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2] \,:\, f \in \mathrm{L}^2(p_k)\}\,.}$$

- This **loss function** is called the **D**enoising **S**core **M**atching loss.
  - $\nabla \log p_{k|0}(X_k|X_0) = ?$.

Score matching (3/4)

- Recall that the loss function is given by

$$\boxed{\nabla \log p_k = \arg\min\{\mathbb{E}[\|f(X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2] \ : \ f \in \mathrm{L}^2(p_k)\} \ .}$$

- This **loss function** is called the **D**enoising **S**core **M**atching loss.
  - $\nabla \log p_{k|0}(X_k|X_0) = ?$.
  - f tries to **predict the residual noise from X_k**.

Score matching (4/4)

- Another formulation: the loss satisfies

$$\mathbb{E}[\|f(X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2] \\ = \mathbb{E}[\|f(X_k)\|^2 + 2\mathrm{div}(f(X_k))] + \mathbb{E}[\|\nabla \log p_{k|0}(X_k|X_0)\|^2] \,.$$

- We obtain the **I**mplicit **S**core **M**atching loss function

$$\boxed{\nabla \log p_k = \arg\min\{\mathbb{E}[1/2\|f(X_k)\|^2 + \mathrm{div}(f(X_k))] \;:\; f \in \mathrm{L}^2(p_k)\} \,.}$$

- Comparison between ISM/DSM:
  - **DSM**: access to $\nabla \log p_{k|0}$.
  - **ISM**: no need of the **transition density** but computation of a **divergence**.
  - Approximation with the **Hutchinson estimator**: ?

Training algorithm

- We choose the **DSM** or **ISM** loss for all $k \in \{1, \dots, n_s\}$
  - $\mathrm{DSM}_k(f) = \mathbb{E}[\|f(X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2]$.
  - $\mathrm{ISM}_k(f) = \mathbb{E}[1/2\|f(X_k)\|^2 + \mathrm{div}(f(X_k))]$.

- Defining the **integrated** loss for $f : \mathbb{R}_+ \times \mathbb{R}^p \to \mathbb{R}^p$.
  - $\ell^{\mathrm{DSM}}(f) = \sum_{k=1}^{n_s} \lambda_k \mathrm{DSM}_k(f(k\gamma, \cdot))$,
  - $\ell^{\mathrm{ISM}}(f) = \sum_{k=1}^{n_s} \lambda_k \mathrm{ISM}_k(f(k\gamma, \cdot))$.
  - We define a **weighting** function $\lambda_k \geq 0$.

- Let $\{\mathbf{s}_\theta\}_{\theta \in \Theta}$ a **parametric family of functions** such that $\mathbf{s}_\theta : \mathbb{R}_+ \times \mathbb{R}^p \to \mathbb{R}^p$.
  - Usually $\{\mathbf{s}_\theta\}_{\theta \in \Theta}$ is a family of **neural networks**.
  - We optimize $\ell^{\mathrm{DSM}}(\theta) = \ell^{\mathrm{DSM}}(\mathbf{s}_\theta)$ or $\ell^{\mathrm{ISM}}(\theta) = \ell^{\mathrm{ISM}}(\mathbf{s}_\theta)$.

Backward sampling

- Recall the goal:
  - Sample from $p_{0:n_s}(x_{0:n_s}) = p_{n_s}(x_{n_s}) \prod_{k=0}^{n_s-1} p_{k|k+1}(x_k|x_{k+1})$ (**ancestral sampling**).
  - **Approximate backward**

$$\boxed{p_{k|k+1}(x_k|x_{k+1}) \approx \mathrm{N}(x_k; x_{k+1} + \gamma\{x_{k+1} + 2\nabla \log p_k(x_{k+1})\}, 2\gamma\,\mathrm{Id}).}$$

  - **Approximation of the score** (**DSM** or **ISM** losses).
- Once $\mathbf{s}_{\theta^\star}$ is learned via DSM or ISM losses, i.e. $\mathbf{s}_{\theta^\star}(k, \cdot) \approx \nabla \log p_k$.
- **Sampling scheme**:
  - $Y_0 \sim \mathrm{N}(0, \mathrm{Id})$ (approximate sampling from p_{n_s}).
  - **Approximate ancestral sampling**

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

  - Y_{n_s} is approximately distributed according to the **data-distribution**.
- Some remarks:
  - Time-reversal can be obtained in **continuous-time**.
  - Original approach relies on **annealed Langevin** Song and Ermon (2019).
  - Other approaches Ho et al. (2020); Gao et al. (2020).

Links with annealed Langevin

Failure of the score-estimation

- We now present (one) **original approach** by Song and Ermon (2019).
- Goal: **sampling** from the **data distribution** $p^\star$.
  - **Langevin algorithm**: $X_{k+1} = X_k + \gamma \nabla \log p^\star(X_k) + \sqrt{2\gamma} Z_{k+1}$.
  - Estimation of the **Stein score** $\nabla \log p^\star$ with ISM

$$\boxed{\nabla \log p^\star = \arg\min\{\mathbb{E}[1/2\|f(X)\|^2 + \mathrm{div}(f(X))] \ : \ f \in \mathrm{L}^2(p^\star)\} \ .}$$

  - $X \sim p^\star$.
- **Problems**:
  - **Slow mixing** with Langevin algorithm (non-convexity).
  - **Bad score** approximation.

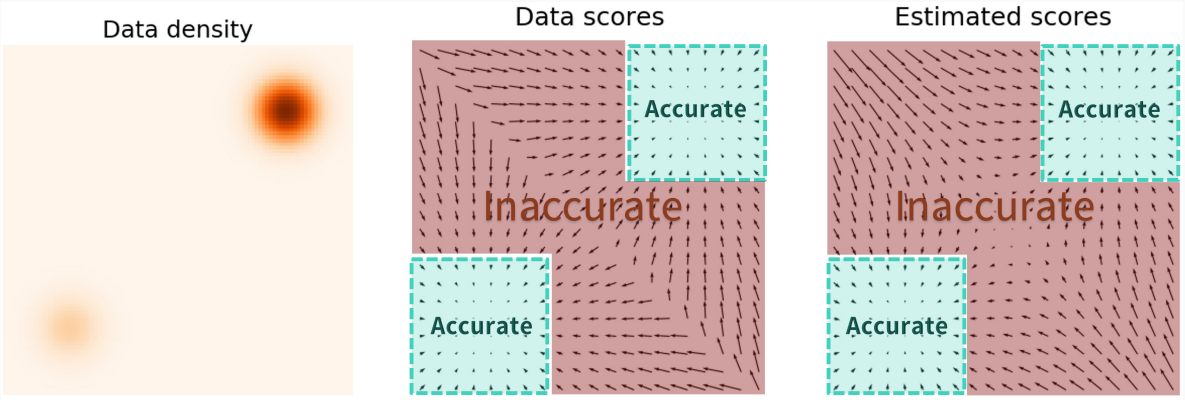

Figure 3: Image extracted from an online tutorial blogpost.

The power of smoothing

- A solution: **smoothing** the density.
  - **Spreading** the observations lead to **better score estimations**.
  - Smoothing leads to **better landscapes** of the potential and **faster mixing** (removal of spurious minima).
- Problem: we do not target the right density.
  - $p_\sigma = p * \mathrm{N}(0, \sigma^2)$.
  - We have that $\mathrm{Var}(p_\sigma) = \mathrm{Var}(p) + \sigma^2$.
- **Trade-off**:
  - **Small value** of σ: close to $p^\star$, hard to sample.
  - **Large value** of σ: far from $p^\star$, easy to sample.

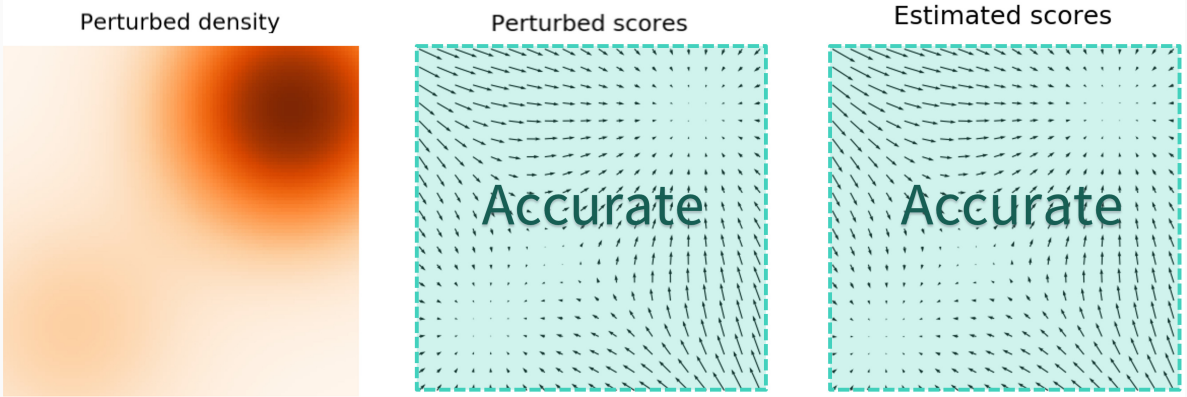

Figure 4: Image extracted from an online tutorial blogpost.

The best of both worlds

- The main of idea of Song and Ermon (2019): **annealed Langevin** dynamics.
  - Starting from a large value of σ_T, sample easily using the **Langevin dynamics**.
  - Reduce the value of $\sigma_T > \sigma_{T-1}$ and **warm-start** the new Langevin dynamics with the previous samples.
  - Repeat the procedure with σ_0 very small (close to the target density).
  - This is an **annealed** procedure.

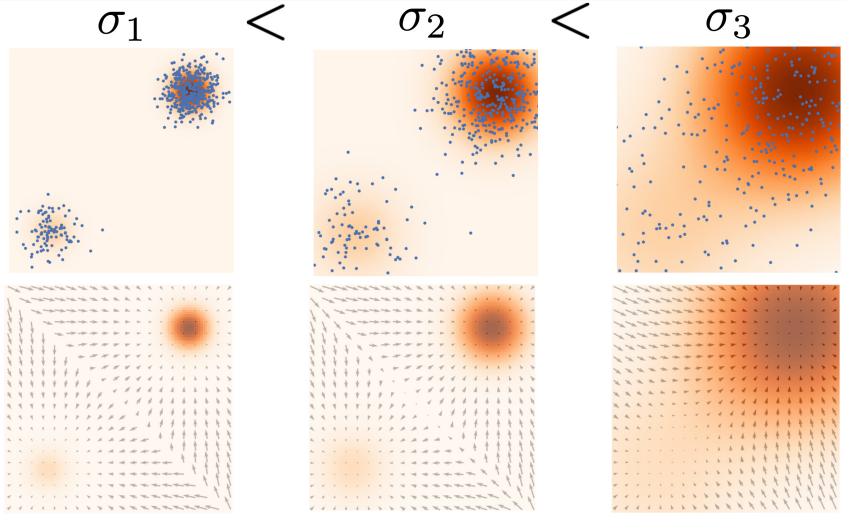

Figure 5: Image extracted from an online tutorial blogpost.

Training algorithm

- Consider $\{\sigma_t\}_{t=1}^{n_s}$ a sequence of increasing variances and set $p_t = p^\star * \mathrm{N}(0, \sigma_t^2)$.
- Denote by $\{X_t\}_{t=0}^{n_s}$ by $X_0 \sim \mu^\star$,

$$X_{t+1} = X_t + (\sigma_{t+1}^2 - \sigma_t^2)^{1/2} Z_{t+1} \ , \{Z_t\}_{t=1}^{n_s} \overset{\text{iid}}{\sim} \mathrm{N}(0, \mathrm{Id}) \ .$$

- We choose the **DSM** or **ISM** loss for all $t \in \{1, \dots, n_s\}$
  - $\mathrm{DSM}_t(f) = \mathbb{E}[\|f(X_t) - \nabla \log p_t(X_t|X_0)\|^2]$.
  - $\mathrm{ISM}_t(f) = \mathbb{E}[{}^1\!/_2\|f(X_t)\|^2 + \mathrm{div}(f(X_t))]$.
- Defining the **integrated** loss for $f : \mathbb{R}_+ \times \mathbb{R}^p \to \mathbb{R}^p$.
  - $\ell^{\mathrm{DSM}}(f) = \sum_{t=1}^{n_s} \lambda_t \mathrm{DSM}_t(f(t\gamma, \cdot))$,
  - $\ell^{\mathrm{ISM}}(f) = \sum_{t=1}^{n_s} \lambda_t \mathrm{ISM}_t(f(t\gamma, \cdot))$.
  - We define a **weighting** function $\lambda_t \geq 0$.
- Let $\{\mathbf{s}_\theta\}_{\theta \in \Theta}$ a **parametric family of functions** such that $\mathbf{s}_\theta : \mathbb{R}_+ \times \mathbb{R}^p \to \mathbb{R}^p$.
  - Usually $\{\mathbf{s}_\theta\}_{\theta \in \Theta}$ is a family of **neural networks**.
  - We optimize $\ell^{\mathrm{DSM}}(\theta) = \ell^{\mathrm{DSM}}(\mathbf{s}_\theta)$ or $\ell^{\mathrm{ISM}}(\theta) = \ell^{\mathrm{ISM}}(\mathbf{s}_\theta)$.

Annealing algorithm

Algorithm 1 Sampling of annealing Langevin dynamics

Input: $\{\sigma_t\}_{t=1}^{n_s}$, $\{\gamma\}_{t=1}^{n_s}$, K
Initialize $Y_{n_s}^0 \sim \mathcal{N}(0, \sigma_{n_s}\,\mathrm{Id})$.
for $t = n_s$ to 1 **do**
  for $k = 0$ to $K-1$ **do**
    Sample $Y_t^{k+1} = Y_t^k + \gamma_t \mathbf{s}_\theta(\sigma_t, Y_t^k) + \sqrt{2\gamma_t} Z_t^{k+1}$
  end for
  $Y_{t-1}^0 = Y_t^K$
end for
Return Y_0^0.

- If $K = 1$ then it is **equivalent to the time-reversal** except that:
  - $\{\gamma_t\}_{t=1}^{n_s}$ is *a priori* unrelated to $\{\sigma_t\}_{t=1}^{n_s}$ contrary to the time-reversal approach where we would have $\gamma_t = \gamma$ and $\sigma_t^2 = t\gamma$.
  - Main difference is that the **forward** process is the discretization of a **Brownian motion** and not a **Ornstein-Ulhenbeck** process.
  - $X_{k+1} = X_k - \gamma X_k + \sqrt{2\gamma} Z_{k+1}$ in the Ornstein-Ulhenbeck setting and $X_{k+1} = X_k + \sqrt{2\gamma} Z_{k+1}$ in the Brownian case.

Implementation details and tricks

Careful implementation is necessary

- Originally these models were **hard** to train Song and Ermon (2019), see also this blogpost.
- In what follows we describe a **series of tricks** which greatly facilitate the training of these models. These tricks can be found in Song et al. (2020b); Song and Ermon (2020); Nichol and Dhariwal (2021); Ho and Salimans (2021); De Bortoli et al. (2021a).
- We *do not* discuss the **architecture** here.

Ornstein-Ulhenbeck and discretization

- Here **SGM** as the discretization of a **OU** process:
  - **Target measure** is $\mathrm{N}(0, \mathrm{Id})$ (approximately), the data should be centered and reduced.
  - **Constant** stepsize discretization is *not* what is done in practice.
- In practice we consider a **schedule** on the stepsize:

$$\boxed{Y_{k+1} = Y_k + \gamma_k\{Y_k + 2\mathbf{s}_\theta(\textstyle\sum_{j=0}^{k}\gamma_j, Y_k)\} + \sqrt{2\gamma_k}Z_{k+1}\ .}$$

  - Linear schedule $\gamma_k = \gamma_{\min} + (\gamma_{\max} - \gamma_{\min})(n_s - k)/n_s$
  - **Intuition**: we need more stepsizes near the data distribution.
  - Different schedules Song and Ermon (2019); Ho et al. (2020); Nichol and Dhariwal (2021)

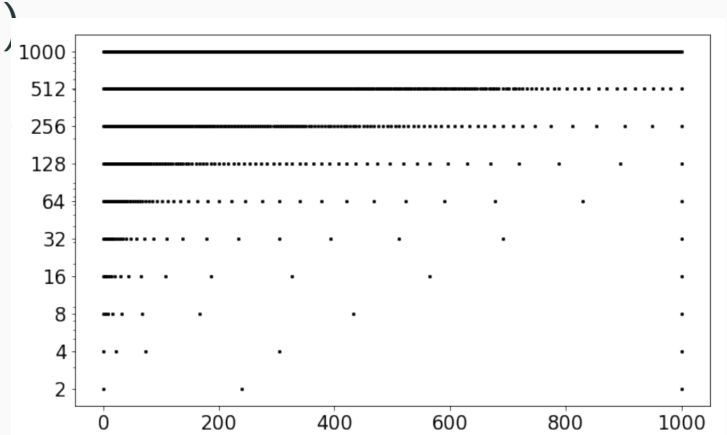

Figure 6: Budget of stepsizes. Image extracted from Watson et al. (2021).

Loss function weighting

- In practice a **weighted** version of the **DSM** loss is used.
  - Recall that the **DSM** loss is given by $\mathrm{DSM}_k(f) = \mathbb{E}[\|f(X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2]$.
  - $\ell^{\mathrm{DSM}}(f) = \sum_{k=1}^{n_s} \lambda_k \mathrm{DSM}_k(f(k, \cdot))$.
  - $\nabla \log p_{k|0}(X_k|X_0) = -\hat{Z}_k/\sigma_k$
- **Intuition**: λ_k function of σ_k to **stabilize** the loss Song et al. (2020b).
- Additional remarks:
  - Changing the **discretization schedule** is equivalent to do a **time-change** in the original Ornstein-Ulhenbeck process then a **fixed discretization**.

Exponential Moving Average

- The training of the network is **unstable**.
- To regularize this we consider an **E**xponential **M**oving **A**verage of weights.

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

  - The parameter m corresponds to the **forgetting** of the initial conditions.
  - The parameters $\bar{\theta}_{\mathrm{K}}$ are used at **sampling** times (K is the number of training steps).

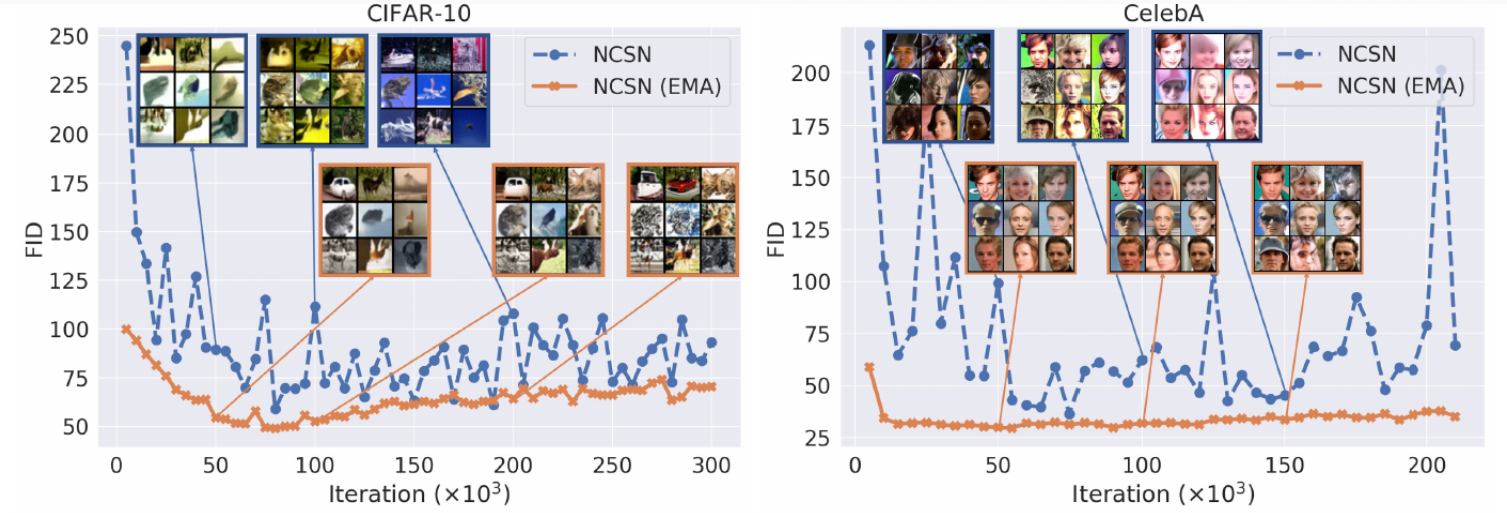

Figure 7: Training instabilities. Image extracted from Song and Ermon (2020).

Predictor-corrector

- Recall that we consider the following **Euler-Maruyama discretization**

$$\boxed{Y_{k+1} = Y_k + \gamma_k \{ Y_k + 2\mathbf{s}_\theta(\textstyle\sum_{j=0}^{k} \gamma_j, Y_k) \} + \sqrt{2\gamma_k} Z_{k+1} \ .}$$

- We can also correct the Euler-Maruyama scheme using the **time-reversal** property.
  - We must have $\mathcal{L}(Y_k) \approx p_k$.
  - Hence we go from Y_k to $\hat{Y}_{k+1}$ with the Euler-Maruyama scheme (**predictor**).
  - We refine $\hat{Y}_{k+1}$ by running a Langevin chain targeting p_{k+1} (**corrector**).

$$\boxed{Y_{k+1}^0 = \hat{Y}_{k+1} \ , \qquad Y_{k+1}^{\ell+1} = Y_{k+1}^{\ell} + \delta_{k+1} \mathbf{s}_\theta(\textstyle\sum_{j=0}^{k+1} \gamma_j, Y_{k+1}^{\ell}) + \sqrt{2\delta_{k+1}} Z_{k+1}^{\ell+1} \ .}$$

  - $\{\delta_k\}_{k=0}^{n_s}$ is a sequence of stepsizes and we set $Y_{k+1} = Y_{k+1}^L$ ($L \in \mathbb{N}$).

Conditional sampling and classifier-free guidance

- If the data distribution contains **classes** (like MNIST, CIFAR-10, LSUN, ImageNet or CelebA when classifying by attributes) then we can exploit this extra **structure**.
- Define a **conditional score** $\mathrm{DSM}_k(f) = \mathbb{E}[\|f(X_k^c, c) - \nabla \log p_{k|0}(X_k^c | X_0^c)\|^2]$.
  - $c \in \{1, \dots, C\}$ is the class of the image.
  - Then, we can (approximately) sample from the class c by considering $Y_0^c \sim \mathrm{N}(0, \mathrm{Id})$

$$\boxed{Y_{k+1}^c = Y_k^c + \gamma_k \{ Y_k^c + 2\mathbf{s}_\theta(\textstyle\sum_{j=0}^k \gamma_j, Y_k^c, c)\} + \sqrt{2\gamma_k} Z_{k+1} \ .}$$

Figure 8: Class conditional generation. Image extracted from Song et al. (2020b).

- Other improvements with **unconditional guidance** Ho and Salimans (2021) or **classifier guidance** Dhariwal and Nichol (2021).

Other approaches

Links with other models

- Until now we have presented two approaches to derive **score-based generative models** (SGMs) :
  - A discrete-time **time-reversal** approach.
  - An **annealed Langevin** approach.
- The time-reversal approach is now widely used Song et al. (2020b).

- We now present links with other **generative models**:
  - SGMs as **variational autoencoders** Ho et al. (2020).
- The connection with variational autoencoders allows for:
  - **Extension** to discrete settings
  - **Acceleration** of the sampling dynamics Watson et al. (2021)

Connections with Variational AutoEncoders

A variational perspective

- We follow the approach of Ho et al. (2020).
- **Variational approach** offers great flexibility:
  - Optimization of the stepsize Watson et al. (2021).
  - Learning of the covariance matrix Nichol and Dhariwal (2021).
  - Non-Markov dynamics Song et al. (2020a).
- Ho et al. (2020) was the first to propose a **discretized Ornstein-Ulhenbeck** Markov chain as a forward process.

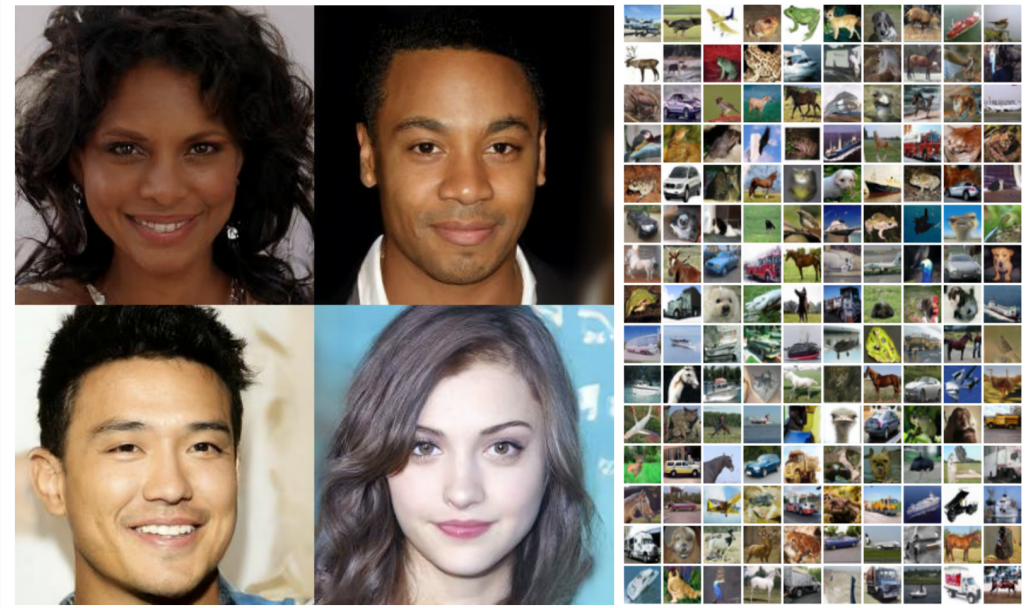

Figure 9: CelebA and CIFAR10 results. Image extracted from Ho et al. (2020).

An Evidence Lower BOund (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** for some well chosen latent Markov model

$$\log(p_0^\theta(x_0)) = ?$$

An Evidence Lower BOund (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** for some well chosen latent Markov model

$$\log(p_0^\theta(x_0)) = ?$$

- We now choose the **variational distribution** $q^\phi(x_{1:n_s}|x_0)$:
  - We choose a tractable Markovian (Gaussian) decomposition $q^\phi(x_{1:n_s}|x_0) = \prod_{k=0}^{n_s-1} q^\phi_{k+1|k}(x_{k+1}|x_k)$.
  - **Factorization** $q^\phi(x_{1:n_s}|x_0) = q^\phi_{n_s|0}(x_{n_s}|x_0) \prod_{k=1}^{n_s-1} q^\phi_{k|k+1,0}(x_k|x_{k+1}, x_0)$.
  - **Tractability** of $q^\phi_{k|k+1,0}$.
- Here, we consider

$$\boxed{q^\phi_{k+1|k}(x_{k+1}|x_k) = \mathrm{N}(x_{k+1}; (1-\gamma)x_k, 2\gamma\,\mathrm{Id}).}$$

- In Ho et al. (2020): auto-regressive process

An Evidence Lower BOund (2/2)

- Recall that we have $\log(p_0^\theta(x_0)) \geq \mathcal{L}$ with

$$\boxed{\mathcal{L} = \int_{(\mathbb{R}^p)^{n_s}} \log(\prod_{k=0}^{n_s-1} p_{k|k+1}^\theta(x_k|x_{k+1}) p_{n_s}^\theta(x_{n_s}) / q^\phi(x_{1:n_s}|x_0)) q^\phi(x_{1:n_s}|x_0) \mathrm{d}x_{1:n_s} \ .}$$

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

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

- with:
  - $\mathcal{L}_{n_s} = \int_{\mathbb{R}^p} \log(p_{n_s}^\theta(x_{n_s}) / q_{n_s|0}^\phi(x_{n_s}|x_0)) q_{n_s|0}^\phi(x_{n_s}|x_0) \mathrm{d}x_{n_s}$.
  - $\mathcal{L}_k = \int_{\mathbb{R}^p} \log(p_{k|k+1}^\theta(x_k|x_{k+1}) / q_{k|k+1,0}^\phi(x_k|x_{k+1}, x_0)) q_{k,k+1|0}^\phi(x_k, x_{k+1}|x_0) \mathrm{d}x_k$.
  - $\mathcal{L}_0 = \int_{\mathbb{R}^p} \log(p_{0|1}^\theta(x_0|x_1)) q_{1|0}^\phi(x_1|x_0) \mathrm{d}x_1$.
- **The different terms**:
  - $\mathcal{L}_{n_s}$ does not depend on θ if we choose $p_{n_s}^\theta = \mathrm{N}(0, \mathrm{Id})$.
  - $\mathcal{L}_k$ is related to **score-matching**.
  - $\mathcal{L}_0$ is more complicated and will be dealt with later.

The backward $q^{\phi}_{k|k+1,0}$ (1/2)

- To compute $\mathcal{L}_k$ we need to compute $q^{\phi}_{k|k+1,0}$.
- We know that $q^{\phi}_{k|k+1,0}$ is **Gaussian** with **diagonal covariance** and just need to compute its parameter.
- $q^{\phi}_{k|0} = \mathrm{N}(\alpha_k x_0, \sigma_k \,\mathrm{Id})$ and $q^{\phi}_{k+1|k} = \mathrm{N}(\alpha_{k+1|k}, \sigma_{k+1|k} \,\mathrm{Id})$.
- **Computing the parameters**:
  - $\alpha_{k+1|k} = 1 - \gamma$, $\sigma^2_{k+1|k} = 2\gamma$.
  - $X_{k+1} = (1-\gamma)^k X_0 + \sqrt{2\gamma} \sum_{j=1}^{k} (1-\gamma)^{k-j} Z_{j+1} = (1-\gamma)^k X_0 + \sigma_{k+1} \hat{Z}_{k+1}$.
  - $\hat{Z}_{k+1} \sim \mathrm{N}(0, \mathrm{Id})$ and $\sigma^2_{k+1} = (1 - (1-\gamma)^{2k})/(1 - \gamma/2)$.
  - $\alpha_{k+1} = (1-\gamma)^k$, $\sigma^2_{k+1} = (1 - (1-\gamma)^{2k})/(1 - \gamma/2)$.
- We have that

$$\boxed{q^{\phi}_{k|k+1,0}(x_k|x_{k+1}, x_0) = q^{\phi}_{k+1|k}(x_{k+1}|x_k) q^{\phi}_{k|0}(x_k|x_0) / q^{\phi}_{k+1|0}(x_{k+1}|x_0) \ .}$$

  - We can **discard the denominator** (normalizing constant).
  - We can focus on $\log(q^{\phi}_{k+1|k}(x_{k+1}|x_k) q^{\phi}_{k|0}(x_k|x_0))$.

The backward $q^{\phi}_{k|k+1,0}$ (2/2)

- We have that

$$-2\log(q^{\phi}_{k+1|k}(x_{k+1}|x_k)q^{\phi}_{k|0}(x_k|x_0))$$
$$= \|x_k - A_{k|k+1}x_{k+1} + B_{k|k+1}\hat{z}_{k+1}\|^2/(2\sigma^2_{k|k+1}) + D\ .$$

where

$$\hat{z}_{k+1} = (x_{k+1} - x_0)/\sigma_{k+1}\ .$$

  - D is a constant **independent** from x_k.
  - $\sigma^2_{k|k+1} = (\alpha^2_{k+1|k}/\sigma^2_{k+1|k} + (1/\sigma^2_k))^{-1}$.
  - $A_{k|k+1} = \alpha_{k+1|k}(\sigma_{k|k+1}/\sigma_{k+1|k})^2 + \alpha_k\sigma^2_{k|k+1}/(\alpha_{k+1}\sigma^2_k)$.
  - $B_{k|k+1} = (\alpha_k\sigma_{k+1}\sigma^2_{k+1|k})/(\alpha_{k+1}\sigma^2_k)$.

- Therefore, we **choose the family**

$$\boxed{-\log(p^{\theta}_{k|k+1}(x_k|x_{k+1})) = \|x_k - A_{k|k+1}x_{k+1} + B_{k|k+1}\hat{z}_{\theta,k+1}(x_{k+1})\|^2/(2\sigma^2_{k|k+1}) + E\ .}$$

- E is a constant, $\hat{z}_{\theta,k+1}(x_{k+1})$ is a function of x_{k+1} (**estimator of the noise**).

Sampling from the model

- How to **train** and **sample** the model?
- Recall that we have set

$$\boxed{-\log(p^{\theta}_{k|k+1}(x_k|x_{k+1})) = \|x_k - A_{k|k+1}x_{k+1} + B_{k|k+1}\hat{z}_{\theta,k+1}(x_{k+1})\|^2/(2\sigma^2_{k|k+1}) + E}$$

- Recall that $p^{\theta}_{n_s} = \mathrm{N}(0, \mathrm{Id})$. To **sample from the model**:
  - We sample $Y_0 \sim \mathrm{N}(0, \mathrm{Id})$
  - We consider the **backward update**

$$\boxed{Y_{k+1} = A_{k|k+1}Y_k - B_{k|k+1}\hat{z}_{\theta,k+1}(Y_k) + \sigma_{k|k+1}Z_{k+1}\ .}$$

- To **train the model** (without the therm $\mathcal{L}_{1|0}$):
  - Minimize $\sum_{k=1}^{n_s}\mathcal{L}_k(\theta)$, with

$$\boxed{-\mathcal{L}_k(\theta) = \mathbb{E}[\|Y_{k+1} - A_{k|k+1}Y_k + B_{k|k+1}\hat{z}_{\theta,k+1}(Y_k)\|^2]/(2\sigma^2_{k|k+1})\ .}$$

Taylor expansion and comparison with SGM (2/2)

- The model is already **similar to SGM**:
  - We sample from $\mathrm{N}(0, \mathrm{Id})$ and use **ancestral sampling**.
  - We train part of the **drift term**.
- The analogy becomes even stronger when considering **Taylor expansion** of $A_{k|k+1}$, $B_{k|k+1}$ and $\sigma_{k|k+1}$:
  - $A_{k|k+1} = 1 + \gamma + o(\gamma)$.
  - $B_{k|k+1} = 2\gamma + o(\gamma)$.
  - $\sigma^2_{k|k+1} = 2\gamma + o(\gamma)$.
- Hence

$$Y_{k+1} = A_{k|k+1} Y_k - B_{k|k+1} \hat{z}_{\theta,k+1}(Y_k) + \sigma_{k|k+1} Z_{k+1} \ ,$$

- becomes (up to the first order)

$$\boxed{Y_{k+1} = (1+\gamma) Y_k - 2\gamma \hat{z}_{\theta,k+1}(Y_k) + \sqrt{2\gamma} Z_{k+1} \ .}$$

- We can identify this recursion with the one of SGM if $\hat{z}_{\theta,k+1} \approx -\nabla \log p^{\theta}_{k+1}$, i.e., the neural network approximates the **score**.

Taylor expansion and comparison with SGM (2/2)

- We want to show that $\hat{z}_{\theta,k+1} \approx -\nabla \log p^{\theta}_{k+1}$, i.e. the neural network approximates the **score**.
- Recall that we minimize the sum of the following **loss functions**

$$-\mathcal{L}_k(\theta) = \mathbb{E}[\| Y_{k+1} - A_{k|k+1} Y_k + B_{k|k+1} \hat{z}_{\theta,k+1}(Y_k)\|^2]/(2\sigma^2_{k|k+1}) \ .$$

- Up to the first order we get that

$$\boxed{-\mathcal{L}_k(\theta) = \mathbb{E}[\| Y_{k+1} - (1+\gamma) Y_k + 2\gamma \hat{z}_{\theta,k+1}(Y_k)\|^2]/(2\gamma) \ .}$$

- But we have $(1+\gamma)X_{k+1} = (1-\gamma^2)X_k + \sqrt{2\gamma}(1+\gamma)Z_{k+1}$ and therefore $(1+\gamma)Y_k = (1-\gamma^2)Y_{k+1} + \sqrt{2\gamma}(1+\gamma)Z_{n_s-k}$.
- Hence, up to the first order we get that

$$\boxed{-\mathcal{L}_k(\theta) = \mathbb{E}[\| \sqrt{2\gamma} Z_{n_s-k} + 2\gamma \hat{z}_{\theta,k+1}(Y_{k+1})\|^2]/(2\gamma) \ ,}$$

- This is exactly the **Denoising Score Matching** loss (up to a minus term) times λ_k (the **weighting function** appearing score-based models being fixed to $\lambda_k = 2\gamma$).

The term $\mathcal{L}_0$

- The previous recursion is valid up to $k = 1$.
- p_θ is an **independent decoder** on the pixel of the image.
- We assume that $x_0 \in [-1, 1]^d$

$$\boxed{p_\theta(x_0|x_1) = \prod_{i=1}^{p} \int_{a(x_0^i)}^{b(x_0^i)} \exp[-\|x - \mu_\theta(x_1)\|^2 / \sigma_1^2] / (2\pi\sigma_1^2)^p \mathrm{d}x \ .}$$

- $a(t) = t + 1/255$ if $t < 1$ and $+\infty$ otherwise.
- $b(t) = t - 1/255$ if $t > -1$ and $-\infty$ otherwise.
- We could also have chosen the classical (non-discrete) **decoding Gaussian of the VAE**.

Figure 10: CelebA results. Image extracted from Ho et al. (2020).

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.

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

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**.

Time Reversal and Comparison with ancestral sampling

- Recall that in the **discrete-time** setting we have

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

- 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 $\mathbf{s}_{\theta^\star}(\gamma(n_s - k), \cdot)$ is supposed to be close to $p_{n_s - k}$, the density of $X_{n_s - k}$.
- p_{T-t} is the density of $\mathbf{X}_{T-t}$ 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.

References

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

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

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.

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.

Valentin De Bortoli, Arnaud Doucet, Jeremy Heng, and James Thornton. Simulating diffusion bridges with score matching. *arXiv preprint arXiv:2111.07243*, 2021a.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34, 2021b.

References ii

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

Garland B Durham and A Ronald Gallant. Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*, 20(3):297–338, 2002.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.

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

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

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

Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34, 2021.

References iii

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

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

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

References iv

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

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33: 12438–12448, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34, 2021.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

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