Score-Based Generative Modeling Through Stochastic Differential Equations

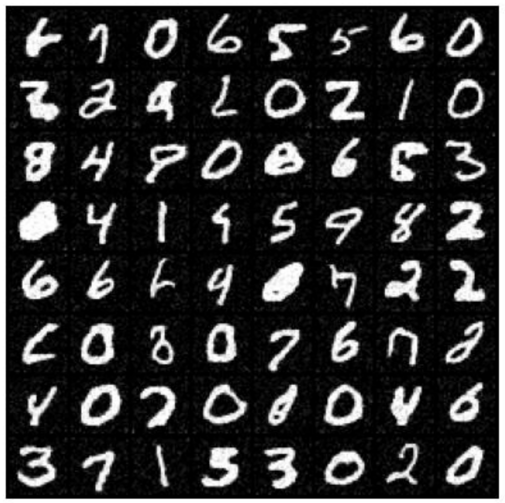
MATHÉMATIQUES VISION APPRENTISSAGE

école normale supérieure paris—saclay

Objective: given i.i.d. samples, learn *to sample from* their distribution

Discrete Langevin Dynamics [1]

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1}) + \sqrt{\epsilon}\mathbf{z}_t$$



Score Denoising Technique [3]

Explicit Score Matching $J^{\text{naive}}(\theta) = \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|^2\right]$ Intractable because we don't know the log-density

$\downarrow$

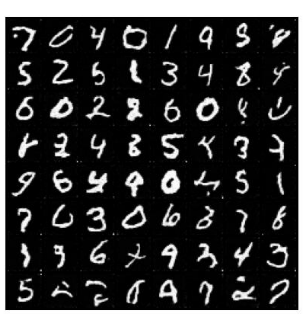
Denoising Score Matching $J^{\text{denoising}}(\theta) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x}), \tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|^2\right]$ SGD compatible

Continuous diffusion via SDEs [2]

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

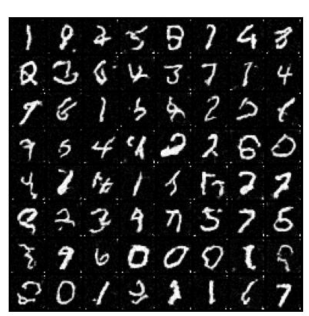
Reverse SDE

$$d\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - g(t)^2\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\right]dt + d\bar{\mathbf{w}}_t$$



Probability flow ODE

$$d\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)\right]dt$$



[1] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution.* Advances in Neural Information Processing Systems 32 (NeurIPS), 2019.

[2] 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, 2021.

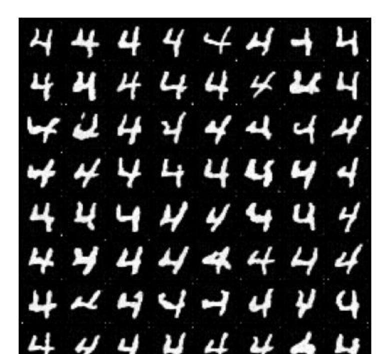
[3] Pascal Vincent. *A Connection Between Score Matching and Denoising Autoencoders.* Neural Computation, 23(7):1661–1674, 2011.

Controllable Generation [2]

Score-based framework is very flexible.
We just need to compute $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y})$
We give two examples.

Class-conditional generation

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y}) = \underbrace{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}_{\approx s_\theta(\mathbf{x},t)} + \underbrace{\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})}_{\text{estimated with backpropagation on time-conditional classifier } \hat{p}(\mathbf{y}|\mathbf{x},t)}$$



Imputation

$$\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$$

$\Omega(\mathbf{x}_0)$ visible; $\bar{\Omega}(\mathbf{x}_0)$ masked

Reverse diffusion on the masked part

$$\nabla_{\bar{\Omega}(\mathbf{x}_t)} \log p_t([\bar{\Omega}(\mathbf{x}_t); \hat{\Omega}(\mathbf{x}_t)])$$

$$\hat{\Omega}(\mathbf{x}_t) \sim p_t(\Omega(\mathbf{x}_t)|\Omega(\mathbf{x}_0) = \mathbf{y})$$