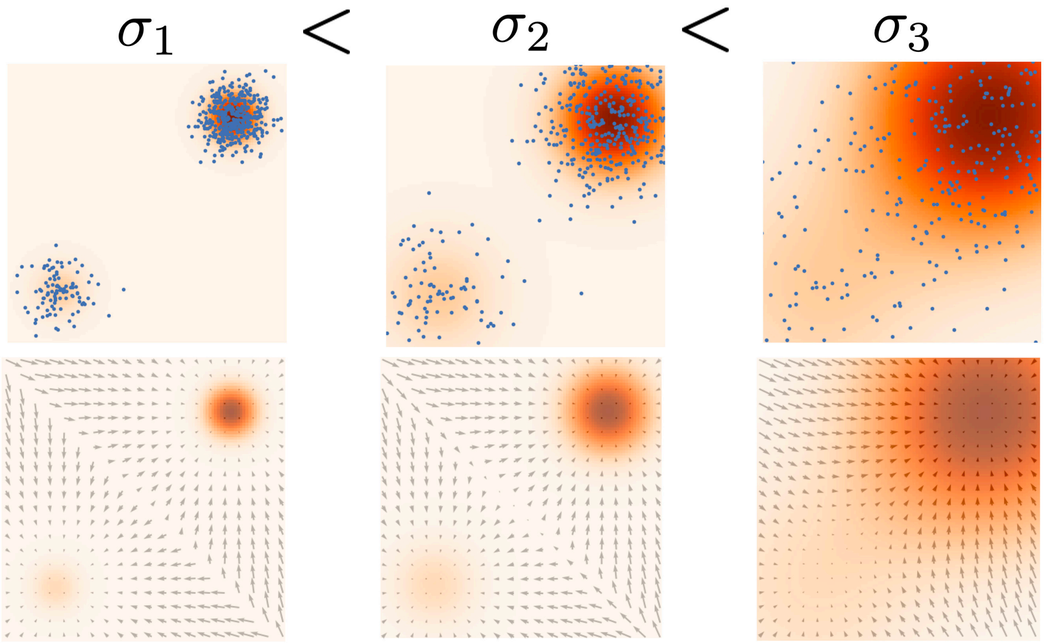
Score-based models with SDEs



- Adding noise fills up low-density regions
- But why "hardcode" to $\sigma_1, \sigma_2, \ldots$?
- With infinite noise scales, we can have
  - Higher quality samples
  - Exact log-likelihood computation
  - Controllable generation with inverse problem solving
- A stochastic process defines a process of generating infinite noise scales

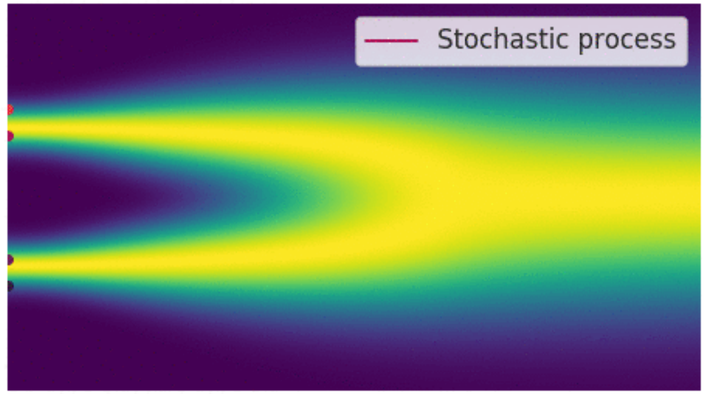
Y. Song, J. Sohl-Dickstein, D. Kingma, A. Kumar, S. Ermon, B. Poole, Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2019

Stochastic processes via SDEs

- Stochastic processes as solutions of stochastic differential equations

$$\mathrm{d}\mathbf{x} = f(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \quad f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d, g \in \mathbb{R}$$

- Change in our RV governed by a function of RV and time (drift) plus noise whose scale is a function of time (diffusion)
- $\mathbf{w}$ is Brownian motion, $\mathrm{d}\mathbf{w}$ is infinitesimal white noise (Gaussian distribution)



Solutions to SDEs

- SDEs usually solved with numerical methods ($\mathbf{x}_{t+1} = \mathbf{x}_t + \dots$) over small time steps
- Solutions of SDEs are stochastic RVs $\{\mathbf{x}(t)\}_{t\in[0,T]} \rightarrow$ trajectories over time
- The pdf of $\mathbf{x}(t)$ is $p_t(\mathbf{x})$ (like $p_{\sigma_i}(\mathbf{x})$ for the discrete case)
- $p_0(\mathbf{x})$ means the distribution in the data space, i.e., $p_0(\mathbf{x}) = p(\mathbf{x})$
- $p_T(\mathbf{x})$ is the distribution after all the noising up for period T until we end up to our prior distribution for our data generation process, i.e., $p_T(\mathbf{x}) = \pi(\mathbf{x})$

Perturbing data with noise from SDEs

- In score-matching the SDE is the generalisation of the finite scaling $\sigma_0, \ldots, \sigma_L$
- In the discrete case perturb with a handpicked (geometric) progression of scales
- Now, noise scale controlled by SDE, where we (manually) select the governing SDE
- If $\mathrm{d}\mathbf{x} = e^t \mathrm{d}\mathbf{w}$, our Gaussian noise $\mathrm{d}\mathbf{w}$ grows exponentially with time

What SDE?

- Noise-conditional score-matching (the discrete case)

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1} \Rightarrow \mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\sigma^2(\mathbf{t})}{\mathrm{d}t}}\mathrm{d}\mathbf{w}$$

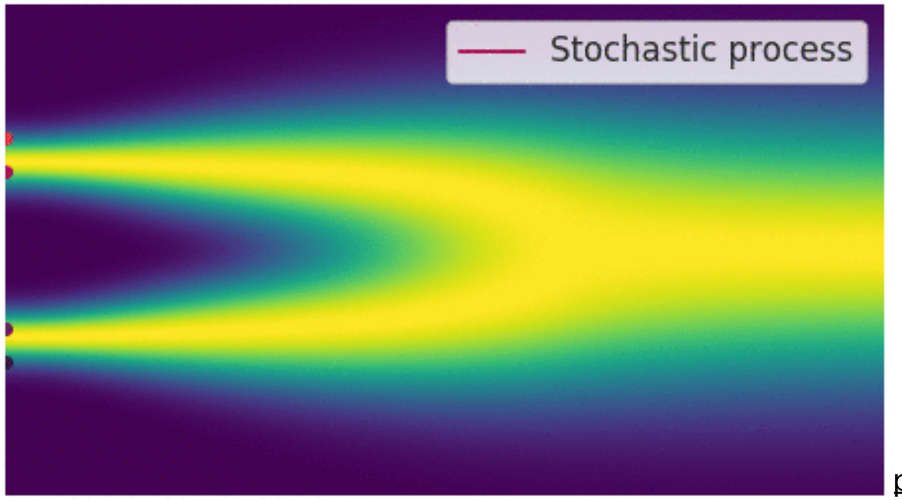
- Another SDE that worked pretty well

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)\left(1 - \exp(-2\int_0^t \beta(s)\mathrm{d}s)\right)}\,\mathrm{d}\mathbf{w}$$

- Might look scary but we merely need to pass it to our numerical solver

From data to SDE noise

- We pick a sample $\mathbf{x}$ from $p_{data}(x)$, e.g., an image from our training set
- We add again and again noise until data looks like a standard Gaussian $\pi(\mathbf{x})$
- We can always make noise from data
- Million dollar question: can we make data from noise (reverse process)?

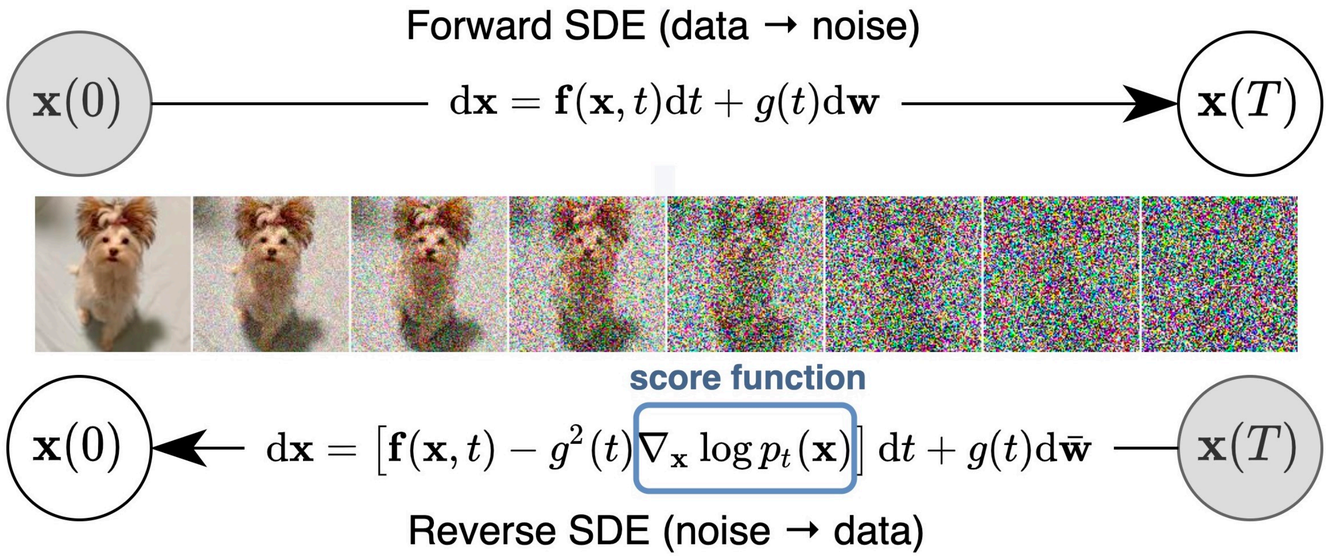


From reverse SDE noise to data

- For any SDE there is a reverse SDE with the reverse trajectories

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

- For the reverse SDE we need precisely the score function of $p_t(\mathbf{x})$



Learning the reverse SDE

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

- A neural network approximates the score function
- Take an initial sample from the prior distribution $\mathbf{x}(T) \sim \pi(\mathbf{x})$
- Solve reverse SDE to get $\mathbf{x}(t)$, $\forall t \in (T,0]$ till model matches data distribution $p_\theta \approx p_0$
- With $\lambda(t) = g^2(t)$ we have

$$\mathrm{KL}(p_0(\mathbf{x}) \| p_\theta(\mathbf{x}))) \leq \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{x\sim p_t(x)}\Big[\Big\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - s_\theta(\mathbf{x})\Big\|_2^2\Big] + \mathrm{KL}(p_T(\mathbf{x}) \| \pi(\mathbf{x})))$$

- With perfect score-matching, matching the prior implies approximating data distribution

Time-dependent score-matching

- Train a neural network for score-matching that depends on time

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(x)}\left[\lambda(t)\left\|\nabla_{\mathbf{x}}\log p_t(\mathbf{x}\,|\,\mathbf{x}_0)-s_\theta(\mathbf{x},t)\right\|_2^2\right]$$

where typically $\lambda(t)\propto 1/\mathbb{E}\left[\|\nabla_{\mathbf{x}(t)}\log p(\mathbf{x}(t)\,|\,x(0))\|_2^2\right]$

- Randomly sample time steps
- Then sample data from training set
- Then optimise your score matching approximation

Solving the reverse SDE

- Once we have trained the score-matching function, we can solve the reverse SDE from the prior π all the way to our data distribution p_0 to generate new data
- Solving means running numerical solver to compute the trajectory (like a decoder)

$$\Delta \mathbf{x} \leftarrow \Big[f(\mathbf{x}, t) - g^2(t) s_\theta(\mathbf{x}, t)\Big]\Delta t + g(t)\sqrt{|\,\Delta t\,|}\,\mathbf{z}_t, \;\; \mathbf{z}_t \sim \mathcal{N}(0, I)$$

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta \mathbf{x}$$

$$t \leftarrow t + \Delta t$$

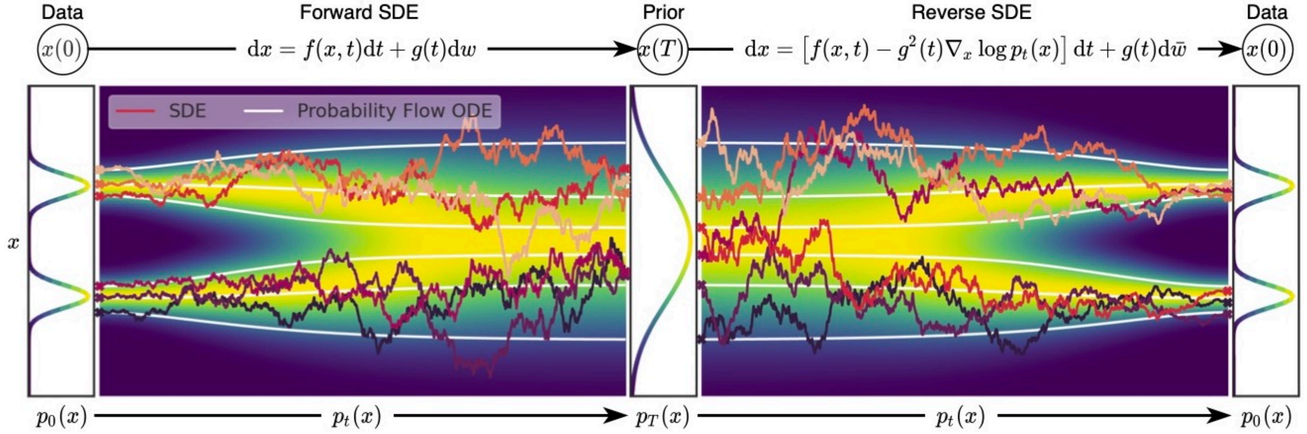
- We solve the SDE (getting the trajectory) for each new sample
- With better sampling procedures and architectures → SoTA generation

Probability flow ODE

- With Langevin MCMC samplers and SDE solvers we can't get exact log-likelihoods
- We can convert the SDE to a corresponding "probability flow" ODE without changing the marginal distributions $\{p_t(\mathbf{x})\}_{t\in[0,T]}$

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

- Solving the ODE, we get the exact log-likelihood



Qualitative examples