Diffusion-based generative models

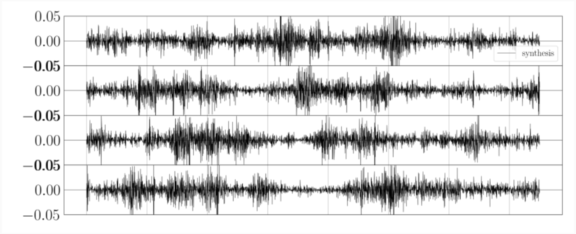
how they work and how to use them

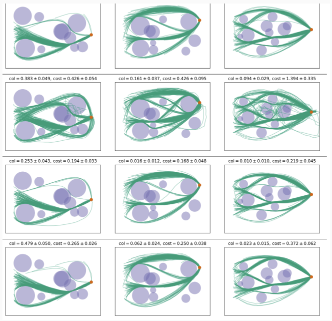
Simon Coste

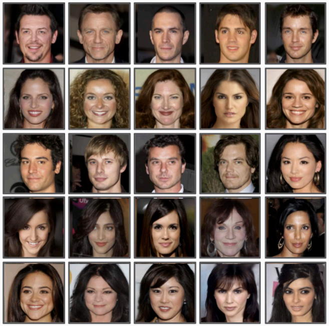
September 27, 2023

Intro: Generative Modelling

$x_*^1, \ldots, x_*^n$: dataset drawn from an unknown distribution ρ_* ("target")







The two goals of generative modelling:

1. Generate ‘new’ samples from ρ_* (direct problem)
2. Find a ‘good’ estimator $\hat{\rho}_*$ for ρ_* (inverse problem)

Examples of generative models: EBMs, GANs, VAEs, Normalizing Flows, Neural ODEs, Noise Contrastive Estimation, Diffusions, Flow matching, Consistency models. . .

The two goals of generative modelling:

1. Generate 'new' samples from ρ_* (direct problem)
2. Find a 'good' estimator $\hat{\rho}_*$ for ρ_* (inverse problem)

Examples of generative models: EBMs, GANs, VAEs, Normalizing Flows, Neural ODEs, Noise Contrastive Estimation, Diffusions, Flow matching, Consistency models...

Diffusion Models
I. Definition

Bridging two distributions

Is there a way to find a random process (X_t) such that

1. $X_0 \sim \rho_* \qquad X_1 \sim N(0, I)$
2. one can easily go from X_1 to X_0 (Markov ? Deterministic ?)
3. (X_t) has nice properties: easy to generate, direct definition, etc.

$X_t = tX_0 + (1-t)\xi$ with $\xi \sim N(0, I)$: too easy!

Bridging two distributions

Is there a way to find a random process (X_t) such that

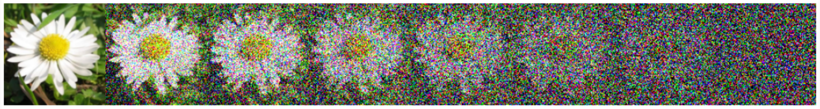
1. $X_0 \sim \rho_*$ $\qquad X_1 \sim N(0, I)$
2. one can easily go from X_1 to X_0 (Markov ? Deterministic ?)
3. (X_t) has nice properties: easy to generate, direct definition, etc.

$X_t = tX_0 + (1-t)\xi$ with $\xi \sim N(0, I)$: too easy!

Vocabulary for the rest of the talk:
Noising/forward process: from ρ_* at time $t = 0$ to $N(0, I)$ at time $t = T$
Forward distribution: p_t = law of X_t
Generative/backward process: from $N(0, I)$ at time $t = 0$ to ρ_* at time $t = T$.
Backward distribution / flow : $q_t = p_{T-t}$

Simple Idea: progressively add noise to the sample.

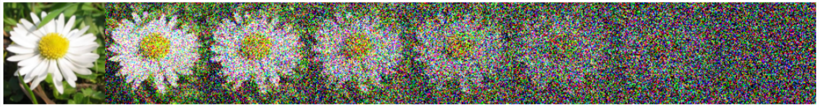
$X_0 \to X_1 \to \cdots \to X_N$ where $X_{k+1} = X_k + \epsilon\xi_k$



Transition probabilities $p(x_{k+1} \mid x_k) = N(x_k, \epsilon^2)$.

Simple Idea: progressively add noise to the sample.

$X_0 \to X_1 \to \cdots \to X_N$ where $X_{k+1} = X_k + \epsilon\xi_k$



Transition probabilities $p(x_{k+1} \mid x_k) = N(x_k, \epsilon^2)$.

We want to reverse the process, but

$$p(x_k \mid x_{k+1}) = \frac{p(x_{k+1}|x_k)p(x_k)}{p(x_{k+1})}$$

is not directly available.

A wild guess:

$$p(x_k \mid x_{k+1}) \approx N(\mu_k(x_k), \epsilon^2)$$

for some (possibly complicated) function μ_k that could be learnt.

Progressive noising

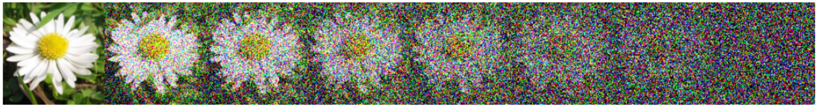
Consider the Ornstein-Uhlenbeck process

$$dX_t = -X_t dt + \sqrt{2} dB_t \qquad X_0 \sim \rho_* \tag{1}$$

$$X_t = e^{-t} X_0 + \sqrt{2} \int_0^t e^{2(s-t)} dBs$$

$$X_t \overset{\text{law}}{=} e^{-t} X_0 + \sqrt{1 - e^{-2t}} \times N(0, I) \to N(0, I)$$

Take T large, say $T \approx 10$. Then $X_T \approx N(0, I)$ (fast mixing).



Formula (1) gives a connection between the target ρ_* and $N(0, I)$.
Can it be reversed?

Generalization

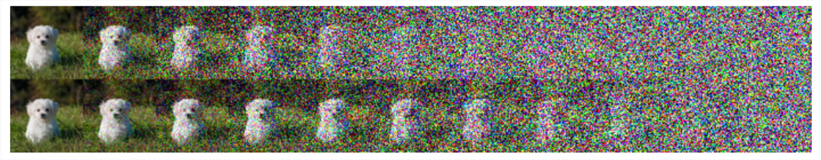
More generally, for any f,

$$dX_t = -\nabla f(X_t) + \sqrt{2}dB_t \qquad X_0 \sim \rho_* \tag{2}$$

gives a connection between ρ_* and e^{-f}/Z where $Z = \int e^{-f(x)}dx$.

$$dX_t = -\alpha(t)X_t dt + \sqrt{2\sigma(t)^2}dB_t$$

$\alpha(t)$: scale schedule $\qquad$ $\sigma(t)$: noise schedule



Time-reversal of continuous Markov processes

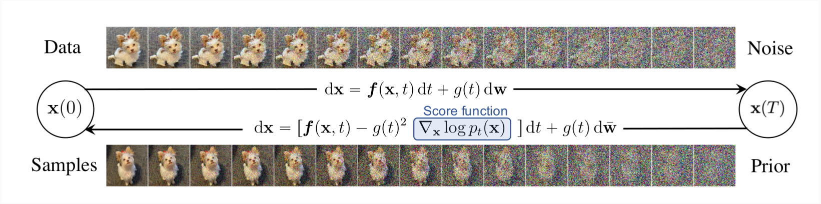
We note p_t the law of X_t and $q_t = p_{T-t}$.

Define a new process by

$$dY_t = (2\nabla \log q_t(Y_t) + Y_t)dt + \sqrt{2}dB_t \qquad Y_0 \sim p_T. \tag{3}$$

$$(X_{T-t})_{t\in[0,T]} \overset{\text{law}}{=} (Y_t)_{t\in[0,T]}.$$

A general paper on time-reversal diffusions: Hausman and Pardoux.


Data
Noise
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \boldsymbol{f}(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
Score function
$\mathrm{d}\mathbf{x} = [\boldsymbol{f}(\mathbf{x}, t) - g(t)^2 \boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\,]\,\mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}$
Samples
Prior

This gives a **generative** process as follows:

1) sample $Y_0 \sim p_T \approx N(0, I)$

2) solve (3) using a numerical scheme until time T

3) the endpoint Y_T should have distribution $\approx \rho_*$.

Problem: in point 2), q_t depends explicitly on ρ_*.

Score-Based Diffusion Models
II. Training

Samples from q_t

Let us recall the **generative** process:

$$dY_t = (2\nabla \log q_t(Y_t) + Y_t)dt + \sqrt{2}dB_t \qquad Y_0 \sim p_T \approx N(0, I). \tag{4}$$

We need access to $\nabla \log q_t$ for every $t \in [0, T]$.

Remember that X_t has the same distribution as $e^{-t}X_0 + N(0, 1 - e^{-2t})$.

Using the samples x_*^i from ρ_*, we get samples from q_t:

$$e^{-t}x_*^i + \sqrt{1 - e^{-2t}}\xi^i \quad \sim p_t = q_{T-t}$$

where ξ^i are iid $N(0, 1)$.

Denoising score matching

q_t is a convolution between ρ_* (rescaled) and a Gaussian!

$\Rightarrow$ we use DNS to estimate $\nabla \log q_t$.

(s_θ): family of parametrized functions (neural networks)

$$\mathrm{DSM}(\theta) = \mathbb{E}\left|s_\theta(X_t) - \varepsilon_t/(1-e^{-2t})\right|^2$$

where $X_t = e^{-t}X_0 + \varepsilon_t$ and $\varepsilon_t \sim N(0,(1-e^{-2t})I)$.

(Reminder: *DSM* has the same minimizers as $\mathbb{E}_{q_t}[|\nabla \log q_t(X_t) - s_\theta(X_t)|^2]$)

For each t this gives an approximation s_{θ_t} of $\nabla \log q_t$.

In practice we use only one network $s(t,x)$.

Writing the loss function

Let $s_\theta : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$ be a family of parametrized functions.

In practice we want to find $\nabla \log q_t$ for all t so we can use the loss

$$\int_0^T \mathbb{E}\left[\left|(1-e^{-2t})^{-1}\varepsilon_t - s_\theta(t, X_t)\right|^2 dt\right] \tag{5}$$

or equivalently, we approximate the integral with a Monte-Carlo method:

$$L(\theta) = \mathbb{E}_{\tau\sim\mathrm{Unif}[0,T]}\mathbb{E}\left[\left|(1-e^{-2\tau})^{-1}\varepsilon_\tau - s_\theta(\tau, X_\tau)\right|^2\right] \tag{6}$$

Clearly if $L(\theta_\star) = 0$ then $s_{\theta_\star}(t, x) = \nabla \log q_t(x)$ for every t, x.

THEORETICAL INTERMEZZO

DM can be seen as EBMs in the path space

- $\mathbb{P}$ = probability law of the process $dX_t = -\alpha_t dt + dB_t$
- $\mathbb{Q}$ = probability law of the process $dX_t = -\beta_t dt + dB_t$

Girsanov's theorem: $\mathbb{P}$ and $\mathbb{Q}$ are equivalent

We can compute the Kullback-Leibler divergence:

$$\mathrm{d}_{\mathrm{KL}}(\mathbb{P} \mid \mathbb{Q}) = \mathbb{E}_{\mathbb{P}}[\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}] = \frac{1}{2}\int_0^T \mathbb{E}[|\alpha_s - \beta_s|^2] ds.$$

DM can be seen as EBMs in the path space

- $\mathbb{P}$ = probability law of the process $dX_t = -\alpha_t dt + dB_t$
- $\mathbb{Q}$ = probability law of the process $dX_t = -\beta_t dt + dB_t$

Girsanov's theorem: $\mathbb{P}$ and $\mathbb{Q}$ are equivalent

We can compute the Kullback-Leibler divergence:

$$\mathrm{d}_{\mathrm{KL}}(\mathbb{P} \mid \mathbb{Q}) = \mathbb{E}_{\mathbb{P}}[\log \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}] = \frac{1}{2}\int_0^T \mathbb{E}[|\alpha_s - \beta_s|^2]ds.$$

Application.

$\mathbb{P}_*$ = true generative process, $\alpha_t(x) = 2\nabla \log q_t(x) + x$

$\mathbb{Q}_\theta$ = our generative process , $\beta_t(x) = 2s_\theta(t, x) + x$

$$\mathrm{d}_{\mathrm{KL}}(\mathbb{P}_* \mid \mathbb{Q}_\theta) = \frac{1}{2}\int_0^T \mathbb{E}[|\nabla \log q_s(X_s) - s_\theta(s, X_s)|^2]ds.$$

Differences with EBMs

In classical EBMs, the distribution of the generative process, $\rho_\theta = e^{-U_\theta}/Z_\theta$, is implicitly defined by U_θ but the normalization constant is not available.

But this normalization constant is necessary to compute the KL divergence!

Differences with EBMs

In classical EBMs, the distribution of the generative process, $\rho_\theta = e^{-U_\theta}/Z_\theta$, is implicitly defined by U_θ but the normalization constant is not available.

But this normalization constant is necessary to compute the KL divergence!

In score-based diffusion models, the law of the generative process is also implicitly defined by the function s_θ.

But due to the structure of the process and Girsanov's theorem, the KL between the model generative process and the target generative process has a simple expression amenable to score matching techniques.

END OF THE THEORETICAL INTERMEZZO

Step-by-step empirical minimization

For each gradient descent step with size η,

1. Draw a batch $x_1^*, \dots, x_n^*$ from the training samples
2. Draw random times $t_1, \dots, t_n$ uniformly on $[0, T]$
3. Draw the corresponding noises $\varepsilon_{t_1}, \dots, \varepsilon_{t_n}$
4. Compute $\mathrm{grad}(\theta_k) = \nabla_\theta \frac{1}{n} \sum_{i=1}^n |\sigma_{t_i}^2 \varepsilon_{t_i} - s_{\theta_k}(t_i, e^{-t_i} x_i^* + \epsilon_{t_i})|^2$
5. $\theta_{k+1} - \theta_k = \eta \times \mathrm{grad}(\theta_k)$ (or any update rule)

Practical matters: scaling + e^{-t} can be dropped

$$X_t = \alpha X_0 + \sqrt{1-\alpha^2}\xi \qquad \text{has the same distribution as} \qquad \frac{X_0 + \sigma\xi}{\alpha^{-1}}$$

with $\sigma^2 = (1-\alpha^2)/\alpha^2$. We note $\tilde{q}_\sigma$ the law of $X_0 + \sigma\xi$.

$$q_t(x) = e^t \tilde{q}_\sigma(xe^t)$$

Learning the family (q_t)

$\Leftrightarrow$

learning the family $(\tilde{q}_\sigma)$ then rescaling

Some important practical points on training Diffusion models.

a Scaling and time parametrization e^{-t} can be dropped

b Pure denoising formulation

c Neural architecture: U-net

Practical matters:Pure denoising formulation

$$L(\theta) = \mathbb{E}_{\tau \sim \mathrm{Unif}[0,T]} \left[|\alpha_\tau^{-1} \varepsilon_\tau - s_\theta(\tau, X_\tau)|^2 \right] \tag{7}$$

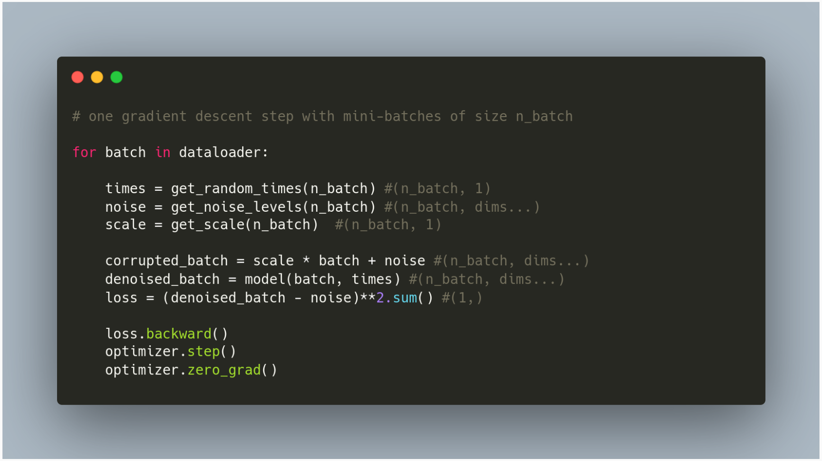
Fact:

$$\begin{aligned} |\alpha^{-1}\varepsilon - \alpha^{-1}s(x+\varepsilon)| &= \alpha^{-1}|\varepsilon + x - x - \alpha^{-1}s(x+\varepsilon)| \\ &= \alpha^{-1}|-x - (\alpha^{-1}s(x+\varepsilon) - (x+\varepsilon))| \\ &= \alpha^{-1}|x - \tilde{s}(x+\varepsilon)| \end{aligned}$$

$$\tilde{L}(\theta) = \mathbb{E}_{\sigma, X, \varepsilon} \left[|X - \tilde{s}_\theta(\sigma, X + \sigma\varepsilon)|^2 \right] \tag{8}$$

$$s_\theta(\sigma, x) = \frac{\tilde{s}_\theta(\sigma, x) - x}{\sigma^2}$$

Formulation (8) is more intuitive and efficient: $\tilde{s}$ is a pure L^2-denoiser.



```
# one gradient descent step with mini-batches of size n_batch

for batch in dataloader:

    times = get_random_times(n_batch) #(n_batch, 1)
    noise = get_noise_levels(n_batch) #(n_batch, dims...)
    scale = get_scale(n_batch)  #(n_batch, 1)

    corrupted_batch = scale * batch + noise #(n_batch, dims...)
    denoised_batch = model(batch, times) #(n_batch, dims...)
    loss = (denoised_batch - noise)**2.sum() #(1,)

    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

Practical matters: neural architecture

Choice of architecture to approximate the score $(t, x) \mapsto \nabla \log \rho_t$?

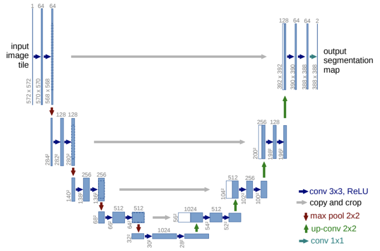


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

- time is embedded into each scale of the U-net
- convolutions + self-attention
- VERY BIG networks

Score-Based Diffusion Models
III. Sampling

SDE sampling

Suppose that we have a good approximation of the score function,

$$s_\theta(t, x) \approx \nabla \log q_t(x).$$

$\Rightarrow$ We simply plug s_θ in the generative process

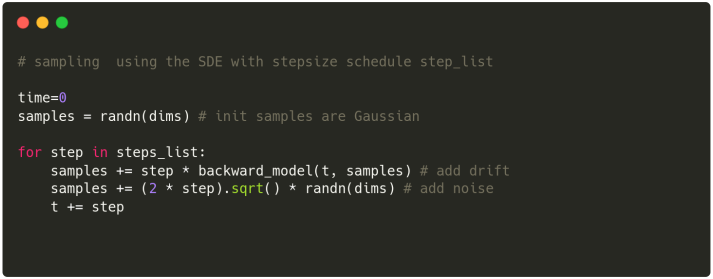
$$dY_t = (2s_\theta(t, Y_t) + Y_t)dt + \sqrt{2}dB_t \qquad Y_0 \sim N(0, I).$$

For solving this SDE we use, for example, an Euler-Maruyama scheme:

$$Y_{k+1} = Y_k + \eta(2s_\theta(k, Y_k) + Y_k) + \sqrt{2\eta}\xi_k$$

- (ξ_k) are iid $N(0, I)$.

- $\eta > 0$ is the stepsize.



```python
# sampling  using the SDE with stepsize schedule step_list

time=0
samples = randn(dims) # init samples are Gaussian

for step in steps_list:
    samples += step * backward_model(t, samples) # add drift
    samples += (2 * step).sqrt() * randn(dims) # add noise
    t += step
```

Is there better to do?

We actually have access to $s_\theta \approx \nabla \log q_t$ for every t. The (q_t) form a connection between ρ_* and $N(0, I)$ in the space of probability measures.

Q: are there *other* processes (X_t) such that $X_t \sim q_t$?

A: yes, a lot.

The SDE does much more than generating a process with q_t as marginals. It also has a very specific structure.

Recast Fokker-Planck as a (fake) Transport Equation

The Fokker-Planck equation associated with

$$dY_t = (2\nabla \log q_t(Y_t) + Y_t)dt + \sqrt{2}dB_t$$

reads

$$\partial_t q_t(x) = \Delta q_t(x) - \mathrm{div}(w_t q_t)$$

with $w_t(x) = 2\nabla \log q_t(x) + x$ and $\mathrm{div} = \nabla\cdot = \sum \partial_i$.

Define $v_t(x) = \nabla \log p_t(x) + x$. Then q_t satisfies the TE

$$\partial_t q_t = -\mathrm{div}(v_t q_t).$$

Proof:

$$\begin{aligned}\Delta q_t - \mathrm{div}(w_t q_t) &= \mathrm{div}\nabla q_t - \mathrm{div}(w_t q_t)\\ &= \mathrm{div}(q_t \nabla \log q_t) - \mathrm{div}(w_t q_t)\\ &= \mathrm{div}(q_t(\nabla \log q_t - w_t)) = -\mathrm{div}(v_t q_t)\end{aligned}$$

ODE sampling: a general transport problem

Let $x: [0,1] \to \mathbb{R}^d$ be the solution of the ODE

$$x' = f_t(x) \qquad\qquad x(0) \sim q_0.$$

where $f_t: \mathbb{R}^d \to \mathbb{R}^d$. Then, its marginal distribution satisfies

$$\partial_t q_t = -\mathrm{div}(f_t q_t)$$

Proof. Let φ be a smooth test function.

$$\begin{aligned}\int \varphi(x)\partial_t p_t(x)dx = \partial_t \mathbb{E}[\varphi(x(t))] &= \mathbb{E}[\nabla\varphi(x(t)) \cdot x'(t)] \\ &= \int \nabla\varphi(x) \cdot f_t(x) p_t(x) dx \\ &= -\nabla\varphi(x)\mathrm{div}(f_t(x)p_t(x))dx.\end{aligned}$$

ODE samplers

Choosing $f_t = v_t$ as above and plugging s_θ instead of $\nabla \log q_t$ yields another generative process with q_t as marginals:

$$X_0 \sim N(0,1) \qquad dX_t = (s_\theta(t, X_t) + X_t)dt.$$

- Only the initial condition is random. No noise is added during the generative process
- ODE solvers are better than SDE samplers (eg Runge-Kutta)
- The flow is *invertible*: two identical initial conditions yield two identical samples.
- Access to $q_T(x) \approx \rho_*(x)$ is feasible (next slide)

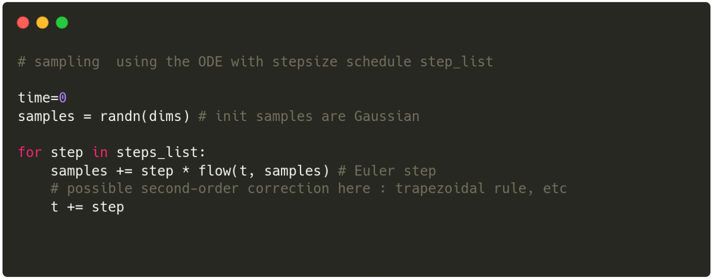
ODE samplers

Choosing $f_t = v_t$ as above and plugging s_θ instead of $\nabla \log q_t$ yields another generative process with q_t as marginals:

$$X_0 \sim N(0,1) \qquad dX_t = (s_\theta(t, X_t) + X_t)dt.$$

- Only the initial condition is random. No noise is added during the generative process
- ODE solvers are better than SDE samplers (eg Runge-Kutta)
- The flow is *invertible*: two identical initial conditions yield two identical samples.
- Access to $q_T(x) \approx \rho_*(x)$ is feasible (next slide)

In practice: it is not clear which samplers are better, ODE or SDEs?



```
# sampling  using the ODE with stepsize schedule step_list

time=0
samples = randn(dims) # init samples are Gaussian

for step in steps_list:
    samples += step * flow(t, samples) # Euler step
    # possible second-order correction here : trapezoidal rule, etc
    t += step
```

Computing exact densities is easier with the ODE

(For simplicity $\dot{a}_t(x)$ = the time derivative and $a'_t(x)$ = space-derivative)

$$\dot{x}_t = v_t(x_t) \qquad \dot{q}_t(x) = -(v_t q_t)'(x)$$

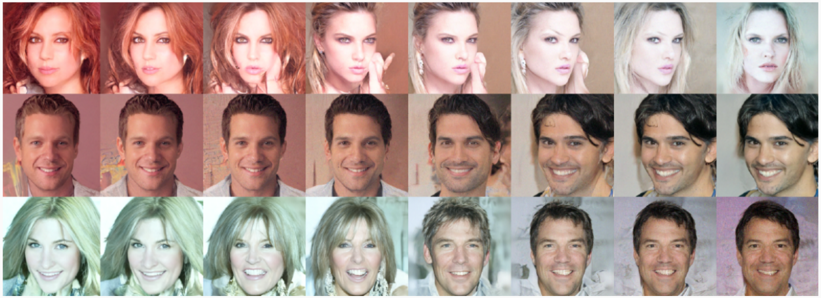
$$\log \rho_*(x) \approx \log q_T(x) = \log q_0(x_0) - dT - \int_0^T \mathrm{div}(s_\theta)(s, x_{T-s}) ds$$

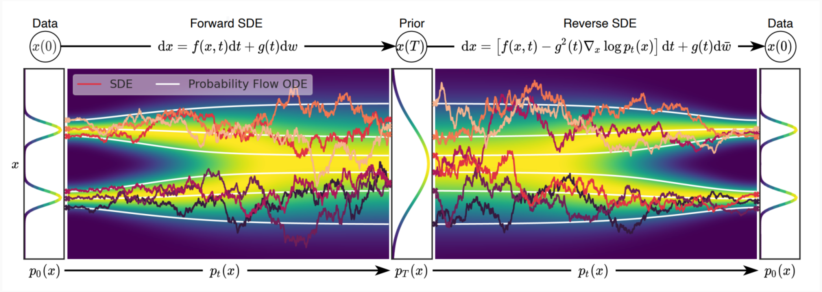
Proof:

$$\begin{aligned}
\log \dot{q_t}(x_t) &= \frac{\dot{q}_t(x_t) + q'_t(x_t)\dot{x}_t}{q_t(x_t)} \\
&= \frac{-(v_t q_t)'(x_t) + q'_t(x_t) v_t(x_t)}{q_t(x_t)} \\
&= \frac{-v'_t(x_t) q_t(x_t) - q'_t(x_t) v_t(x_t) + q'_t(x_t) v_t(x_t)}{q_t(x_t)} \\
&= -v'_t(x_t).
\end{aligned}$$

In our setting $v_t(x) \approx s_\theta(t, x) + x$ so $\mathrm{div}(v_t)(x) \approx \mathrm{div}(s_\theta)(t, x) + d$

Interpolation





Data
Forward SDE
Prior
Reverse SDE
Data
$x(0)$
$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$
$x(T)$
$\mathrm{d}x = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{w}$
$x(0)$
SDE
Probability Flow ODE
x
$p_0(x)$
$p_t(x)$
$p_T(x)$
$p_t(x)$
$p_0(x)$

Extra Techniques

Design choices are everything

- Noise schedule and scale schedule is important depending on the problem
- Predict-correct steps: alternating one ODE steps, one SDE step
- Model Distillation is quite successful
- Don't add noise at the beginning and end
- $s_\theta(t, x) + \lambda \nabla \log D(c|x)$ where D is a trained classifier: classifier-guidance

Thanks for the invitation!

Some references (with links)

The first Diffusion paper by Sohl-Dickstein et al.

The seminal Diffusion paper by Ho et al.

"Diffusion beat GANs", fine engineering by Dhariwal and Nichol

The SDE approach or this paper by the Song and Ermon team

The best blog post on diffusions, by Yang Song

Many point of views on diffusions by Sander Dieleman

Diffusions simplified by a Nvidia team

The Imagen paper (text-driven)