

# Flow-based generative models

how they work and how to use them

---

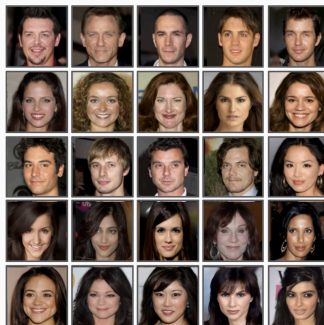
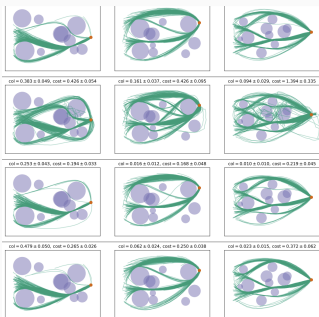
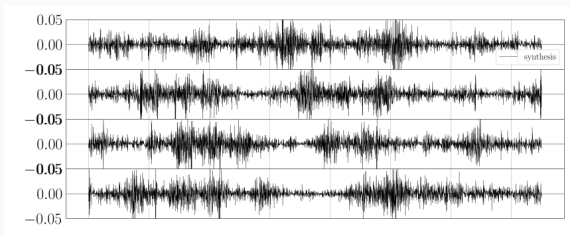
Simon Coste

June 26, 2024

# Intro: Generative Modelling

---

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



The two goals of generative modelling:

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

# Diffusion Models

## I. Definition

---

# Bridging two distributions

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

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

# Bridging two distributions

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

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

Vocabulary for the rest of the talk:

**Noising/forward process:** from  $\rho_*$  at time  $t = 0$  to  $N(0, I)$  at time  $t = T$

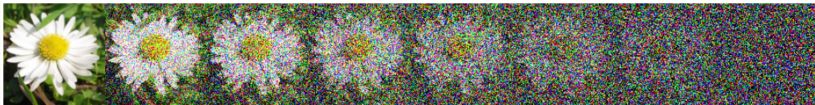
**Forward distribution:**  $p_t = \text{law of } X_t$

**Generative/backward process:** from  $N(0, I)$  at time  $t = 0$  to  $\rho_*$  at time  $t = T$ .

**Backward distribution:**  $q_t = p_{T-t}$

Simple Idea: progressively add noise to the sample.

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

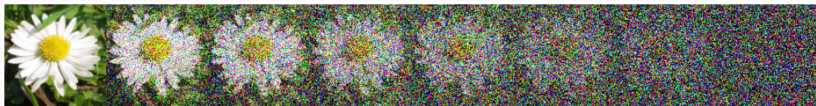


Transition probabilities  $p(x_{k+1} \mid x_k) = N(x_k, \epsilon^2)$ .



Simple Idea: progressively add noise to the sample.

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



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

We want to reverse the process, but

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

is not directly available.

A wild guess:

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

for some (possibly complicated) function  $\mu_k$  that could be learnt.

# Continuous Progressive noising

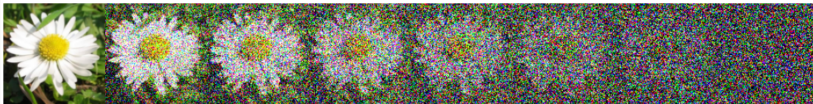
Consider the Ornstein-Uhlenbeck process

$$dX_t = -X_t dt + \sqrt{2} dB_t \quad X_0 \sim \rho_* \quad (1)$$

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

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

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



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

# Generalization

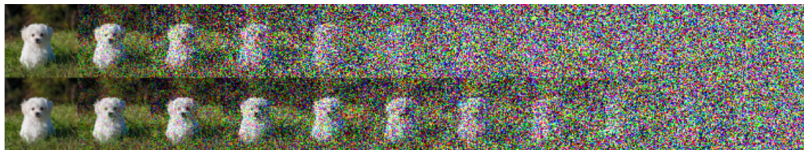
More generally, for any  $f$ ,

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

gives a connection between  $\rho_*$  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                       $\sigma(t)$ : noise schedule



# Time-reversal of continuous Markov processes

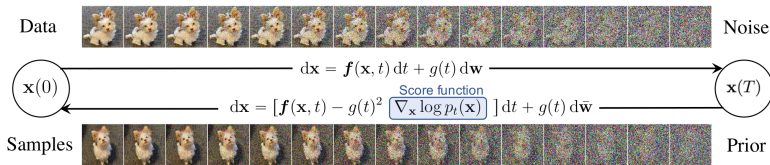
We note  $p_t$  the law of  $X_t$  and  $q_t = p_{T-t}$ .

Define a new process by

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

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

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



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  $\rho_*$ .

# Score-Based Diffusion Models

## II. Training

---

## Samples from $q_t$

Let us recall the **generative** process:

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

We need access to  $\nabla \log q_t = \nabla \log p_{T-t}$  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  $\rho_*$ , we get samples from  $p_t$ :

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

where  $\xi^i$  are iid  $N(0, 1)$ .

# Denoising score matching

$q_t$  is a convolution between  $\rho_*$  (rescaled) and a Gaussian!

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

$(s_\theta)$ : family of parametrized functions (neural networks)

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

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

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

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

In practice we use only one network  $s_\theta(t, x)$ .



# Writing the loss function

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

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

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

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

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

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

## *THEORETICAL INTERMEZZO*

## DM can be seen as EBM 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:

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

# DM can be seen as EBM's 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:

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

*Application.*

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

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

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

*END OF THE THEORETICAL INTERMEZZO*

# Step-by-step empirical minimization

For each gradient descent step with size  $\eta$ ,

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  $\text{grad}(\theta_k) = \nabla_{\theta} \frac{1}{n} \sum_{i=1}^n |\sigma_{t_i}^2 \varepsilon_{t_i} - s_{\theta_k}(t_i, e^{-t_i} x_i^* + \epsilon_{t_i})|^2$
5.  $\theta_{k+1} - \theta_k = \eta \times \text{grad}(\theta_k)$  (or any update rule)

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

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

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



## Practical matters: Pure denoising formulation

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

Fact:

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

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

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

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

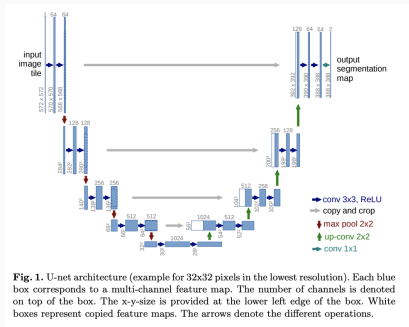
for batch in dataloader:
    times = get_random_times(n) #(n,1)
    noise = get_random_noise(n)  #(n,dim)
    scale = get_scales(times) #(n,1)

    corrupted_batch = batch + scale * noise
    denoised_batch = model(corrupted_batch, times)
    loss = ((denoised_batch - batch)**2).mean() # mean square error

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

# Practical matters: neural architecture

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



- time is embedded into each scale of the U-net
- convolutions + self-attention
- VERY BIG networks (Stable Diffusion 3 has 2 billion parameters)

# Score-Based Diffusion Models

## III. Sampling

---

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

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


⇒ We simply plug  $s_\theta$  in the generative process

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

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

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

- $(\xi_k)$  are iid  $N(0, I)$ .
- $\eta > 0$  is the stepsize.



```
# 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  $\rho_*$  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.

*(Beware: mathematics incoming)*

# 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

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

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

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

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

**Proof:**

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



# ODE sampling: a general transport problem

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

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

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

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

**Proof.** Let  $\varphi$  be a smooth test function.

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

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


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

- 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)

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

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

- 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)



```
# 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)

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

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

**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  $\operatorname{div}(v_t)(x) \approx \operatorname{div}(s_\theta)(t, x) + d$

# Interpolation



## Leveraging the ODE: Flow Matching

---

Let us revert the point of view on ODE sampling.

There is a velocity field  $v_t$  such that the ODE

$$\dot{X}_t = v_t(X_t) \quad x_0 \sim N(0, I) \quad (9)$$

satisfies  $X_T \sim \rho_*$ .

In practice, with the reverse-SDE formulation,

$$v_t(x) = x + \nabla \log q_t(x).$$

By learning  $\nabla \log q_t$ , we were able to learn the velocity field  $v_t$ .

Are there other fields  $v_t$  mapping  $N(0, I)$  to  $\rho_*$  through flow (9)?  
If so, could we **directly** learn such a flow  $v_t$ ?



# Can you directly learn the flow?

Let  $v_t$  be any velocity field connecting  $X_0 \sim q_0$  to  $X_1 \sim \rho_*$ . We could learn approximate  $v_t$  by a NN  $s_\theta(t, x)$  using the loss

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

**Problem:** finding an explicit  $v_t$  transporting  $q_0$  to  $q_1$  is not doable in general.

# Can you directly learn the flow?

Let  $v_t$  be any velocity field connecting  $X_0 \sim q_0$  to  $X_1 \sim \rho_*$ . We could learn approximate  $v_t$  by a NN  $s_\theta(t, x)$  using the loss

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

**Problem:** finding an explicit  $v_t$  transporting  $q_0$  to  $q_1$  is not doable in general.

Exemple: for diffusion maps,  $v_t(x) = \nabla \log q_t(x) + x$  and  $\nabla \log q_t$  depends on  $\rho_*$ .

Let  $\psi_t^x$  be a flow conditioned on ending at  $x$ :

$$\psi_t^x(z) = (1 - t)z + tx.$$

Note that if  $x_t = \psi_t^x(x_0)$  then  $x_t$  is a solution of the ODE

$$\dot{x}_t = x - \frac{x_t - tx}{1 - t} = v_t^x(x_t)$$

with

$$v_t^x(z) = x - (z - tx)/(1 - t).$$

We thus have a family of **conditional** velocity fields, with the ODE associated to  $v_t^x$  conditioned on ending at  $x$ .

Clearly, the averaged flows

$$v_t(y) = \mathbb{E}_{x \sim \rho_*}[v_t^x(y)]$$

transport any starting distribution  $x_0 \sim q_0$  to the target distribution  $x_1 \sim \rho_*$ .

Let  $q_t^X$  be the probability path associated with  $v_t^X$ .

Then, the mean path  $q_t = \mathbb{E}q_t^X$  is the probability path associated to the field

$$v_t(z) = \mathbb{E} \left[ \frac{v_t^X(z) q_t(z|X)}{q_t(z)} \right].$$

**Proof:** Check that  $q_t$  satisfies the Transport Equation with  $v_t$ !

$$\begin{aligned} \partial_t q_t(y) &= \partial_t \mathbb{E}_* q_t^X(y) = \mathbb{E}_* \partial_t q_t^X(y) \\ &= -\mathbb{E}_* \nabla \cdot v_t^X q_t^X(y) \\ &= -\mathbb{E}_* \nabla \cdot \left( \frac{v_t^X q_t^X(y)}{q_t(y)} \right) q_t(y) \\ &= -\nabla \cdot \left( \mathbb{E}_* \frac{v_t^X q_t^X(y)}{q_t(y)} \right) q_t(y) \\ &= -\nabla \cdot v_t q_t(y) \end{aligned}$$

# Flow Matching [many teams in 2022]

Learning the flow  $v_t$  (intractable) and learning the conditional flows  $v_t^x$  (tractable) lead to the same loss function!

$$\begin{aligned} & \int_0^1 \mathbb{E}_{X_t \sim q_t} \left[ |s_\theta(t, X_t) - v_t(X_t)|^2 \right] dt \\ &= \\ & \text{cst} + \int_0^1 \mathbb{E}_{X_t \sim q_t} \left[ |s_\theta(t, X_t) - v_t^x(X_t)|^2 \right] dt \end{aligned}$$

# Flow Matching [many teams in 2022]

Learning the flow  $v_t$  (intractable) and learning the conditional flows  $v_t^x$  (tractable) lead to the same loss function!

$$\begin{aligned} & \int_0^1 \mathbb{E}_{X_t \sim q_t} \left[ |s_\theta(t, X_t) - v_t(X_t)|^2 \right] dt \\ &= \\ & \text{cst} + \int_0^1 \mathbb{E}_{X_t \sim q_t} \left[ |s_\theta(t, X_t) - v_t^x(X_t)|^2 \right] dt \end{aligned}$$

This leads to a simpler, clearer way of learning ODE samplers.

## 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
- 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: helps controlling the sampling towards something
- Choosing the right ODE scheme can be tricky (Heun ? Runge-Kutt?)
- Classifier-free guidance was the game-changer for text-to-image generation
- Model Distillation is quite successful for Diffusion models and Flows!



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

Flow Matching I: Albergo and Vanden-Eijnden

Flow Matching II: Lipman et al.

Flow Matching III: Liu et al.