

# Diffusion Study Group #4

Tanishq Abraham

10/1/2022

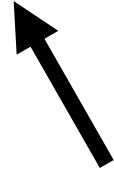
A recap

# Stochastic Differential Equations (SDEs)

Like ODEs... *but with noise!*

$$\frac{d\mathbf{x}}{dt} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\substack{\text{drift} \\ \text{coefficient}}} + \underbrace{g(t)}_{\substack{\text{diffusion} \\ \text{coefficient}}} \frac{d\mathbf{w}}{dt} \rightarrow d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$$

Wiener process  
(Brownian motion)



Numerical solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + g(t)\mathbf{z}_t$$

where  $\mathbf{z}_t \sim \mathcal{N}(0, I)$

# SDE formulation of score-based generative modeling

In the limit of  $\Delta t \rightarrow 0$ , our diffusion process can now be described by  $\mathbf{x}(t)$ , indexed by a continuous time variable  $t \in [0, T]$ .

$\mathbf{x}(0) \sim p_0(x)$  which is the data distribution

$\mathbf{x}(T) \sim p_T(x)$  which is the prior distribution

Let's denote the probability density of  $\mathbf{x}(t)$  as  $p_t(\mathbf{x})$  and the transition kernel as  $p_{st}(\mathbf{x}(t)|\mathbf{x}(s))$

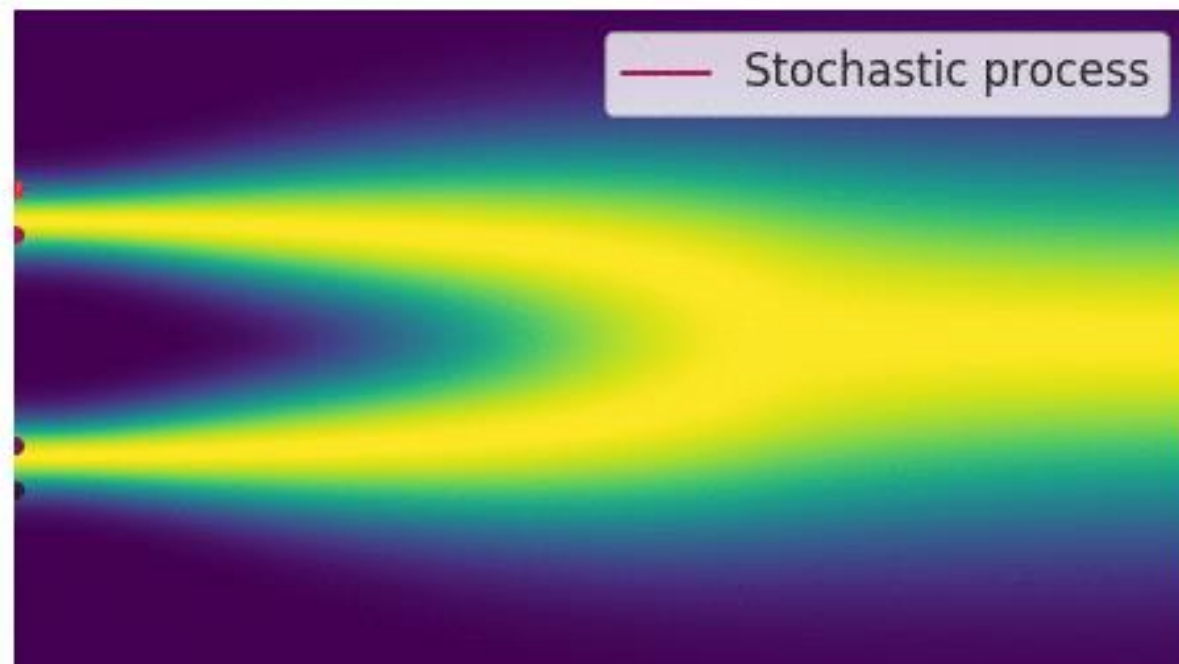
# An important property

The reverse of a diffusion process is also a diffusion process, described by what is known as the Anderson reverse-time SDE:

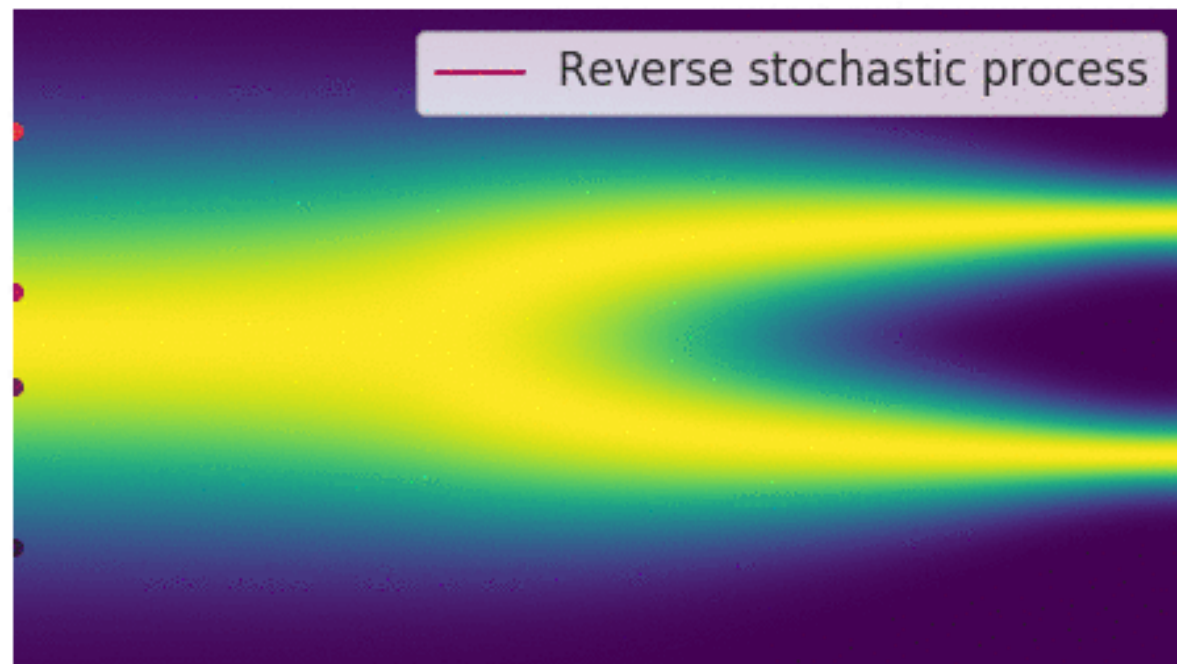
$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}$$

where  $\bar{\mathbf{w}}$  is the Wiener process for time flowing backward from  $T$  to 0

# Forward process as modeled by an SDE



# Reverse process as modeled by a reverse-time SDE



# Different score SDEs

NSCN (VE SDE):

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} d\mathbf{w}$$

DDPM (VP SDE):

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x} dt + \sqrt{\beta(t)}d\mathbf{w}$$

Sub-VP SDE:

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



# Training the score function for the reverse SDE

The training objective with denoising score matching is as follows:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \left[ \left\| \mathbf{s}_{\theta}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) \right\|_2^2 \right] \right\}.$$

$$p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) = \begin{cases} \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}), & \text{(VE SDE)} \\ \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2} \int_0^t \beta(s) ds}, \mathbf{I} - \mathbf{I}e^{-\int_0^t \beta(s) ds}) & \text{(VP SDE)} \\ \mathcal{N}(\mathbf{x}(t); \mathbf{x}(0)e^{-\frac{1}{2} \int_0^t \beta(s) ds}, [1 - e^{-\int_0^t \beta(s) ds}]^2 \mathbf{I}) & \text{(sub-VP SDE)} \end{cases}.$$

Let's continue

# Solving the Reverse SDE

Given the SDE:

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

We have the reverse-time SDE:

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

Which we discretize as such:

$$\mathbf{x}_i = \mathbf{x}_{i+1} - f_{i+1}(\mathbf{x}_{i+1}) + (g_{i+1})^2 \mathbf{s}_{\theta}(\mathbf{x}_{i+1}, i+1) + g_{i+1} \mathbf{z}_{i+1}$$

DDPM ancestral sampling works out to be quite similar to this discretization for VP SDE.

# Predictor-Corrector Sampler

We can utilize the score information!

- Score functions are used to sample from distributions with (Markov Chain Monte Carlo) MCMC approaches

Predictor-Corrector setup:

- Predictor – Numerical SDE solver gives estimate of sample at next timestep
- Corrector – MCMC approach corrects the estimated sample

# Predictor-Corrector Sampler

---

**Algorithm 2** PC sampling (VE SDE)

---

```
1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ 
2: for  $i = N - 1$  to  $0$  do
3:    $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2) \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, \sigma_{i+1})$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2} \mathbf{z}$ 
6:   for  $j = 1$  to  $M$  do
7:      $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9: return  $\mathbf{x}_0$ 
```

---

---

**Algorithm 3** PC sampling (VP SDE)

---

```
1:  $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $i = N - 1$  to  $0$  do
3:    $\mathbf{x}'_i \leftarrow (2 - \sqrt{1 - \beta_{i+1}}) \mathbf{x}_{i+1} + \beta_{i+1} \mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i + 1)$ 
4:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:    $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}} \mathbf{z}$  Predictor
6:   for  $j = 1$  to  $M$  do Corrector
7:      $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
8:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i} \mathbf{z}$ 
9: return  $\mathbf{x}_0$ 
```

---

# Predictor-Corrector sampler results

Table 1: Comparing different reverse-time SDE solvers on CIFAR-10. Shaded regions are obtained with the same computation (number of score function evaluations). Mean and standard deviation are reported over five sampling runs. “P1000” or “P2000”: predictor-only samplers using 1000 or 2000 steps. “C2000”: corrector-only samplers using 2000 steps. “PC1000”: Predictor-Corrector (PC) samplers using 1000 predictor and 1000 corrector steps.

		Variance Exploding SDE (SMLD)				Variance Preserving SDE (DDPM)			
FID↓	Sampler	P1000	P2000	C2000	PC1000	P1000	P2000	C2000	PC1000
	ancestral sampling	4.98 ± .06	4.88 ± .06		<b>3.62</b> ± .03	3.24 ± .02	3.24 ± .02		<b>3.21</b> ± .02
	reverse diffusion	4.79 ± .07	4.74 ± .08	20.43 ± .07	<b>3.60</b> ± .02	3.21 ± .02	3.19 ± .02	19.06 ± .06	<b>3.18</b> ± .01
	probability flow	15.41 ± .15	10.54 ± .08		<b>3.51</b> ± .04	3.59 ± .04	3.23 ± .03		<b>3.06</b> ± .03

# Probability Flow ODEs

Given the SDE:

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

A corresponding “probability flow ODE” can be found:

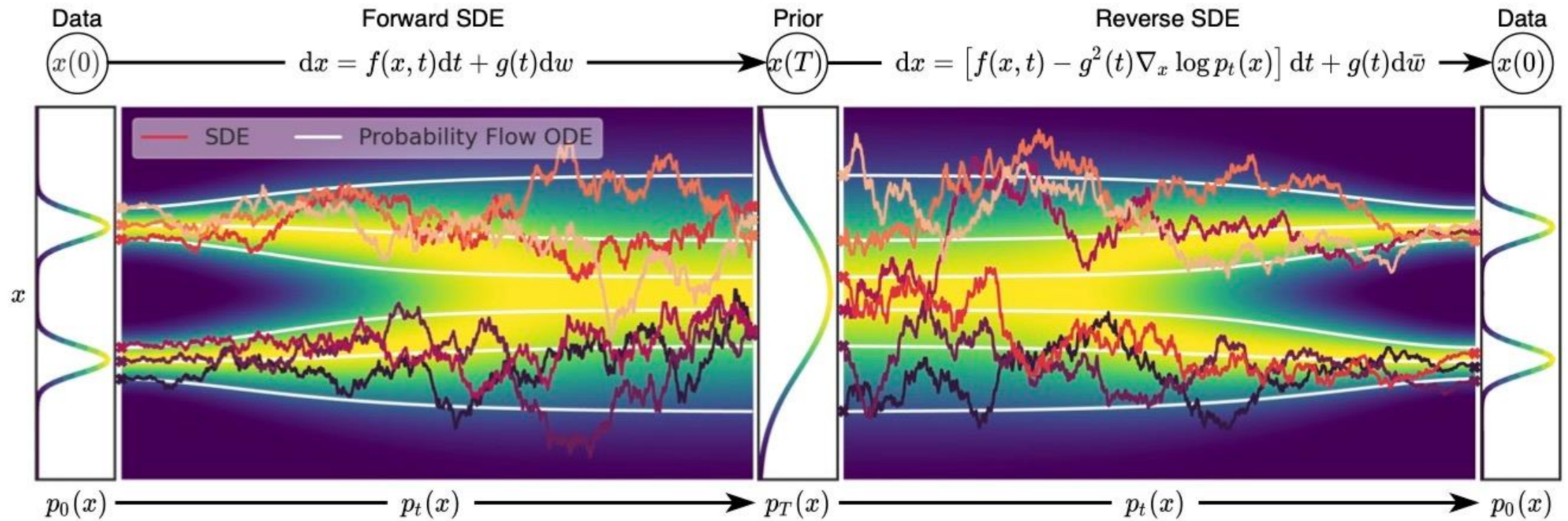
$$d\mathbf{x} = \left[ \mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt$$

Describes the evolution of the probability distribution over time

We can numerically solve the ODE in reverse to generate samples!

Connection to neural ODEs/continuous normalizing flows enable exact log-likelihood calculation

# Probability flow ODEs





# Uniquely identifiable encodings

By integrating over the ODE, we can map an input image  $\mathbf{x}(0)$  to a uniquely identifiable encoding  $\mathbf{x}(T)$

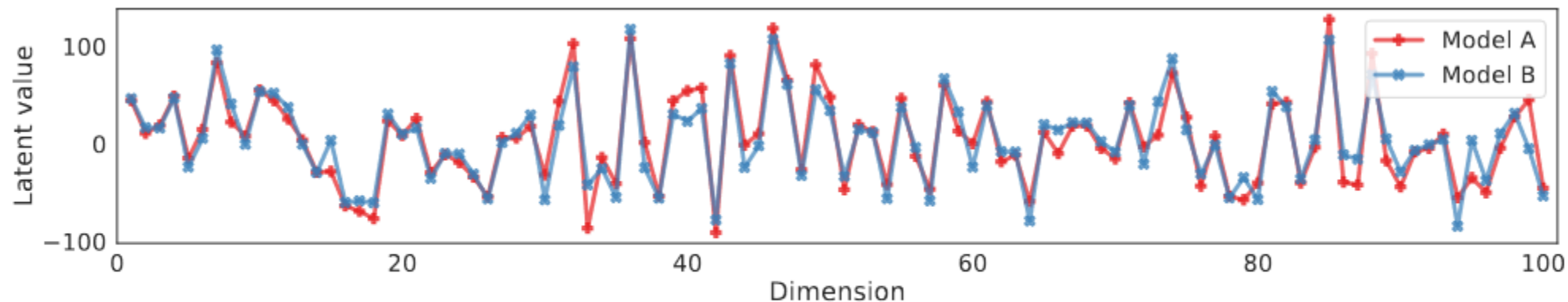
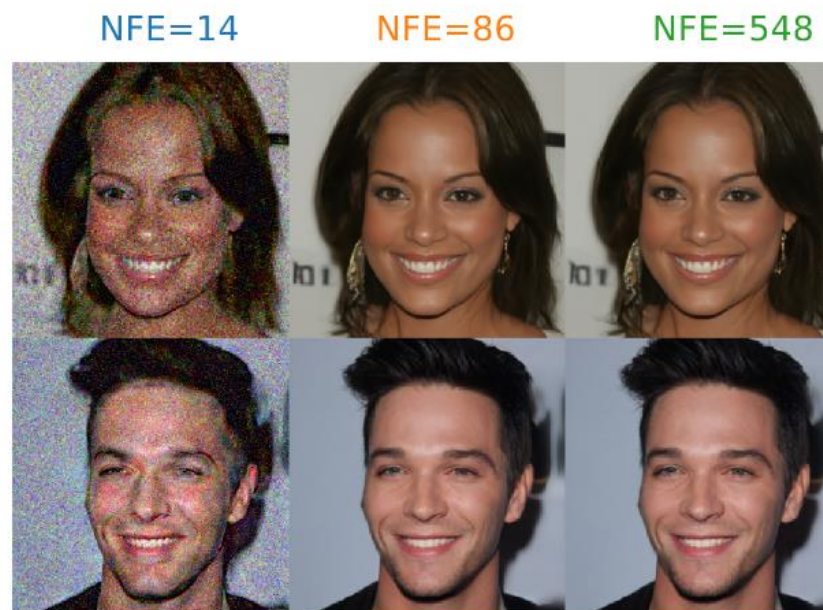
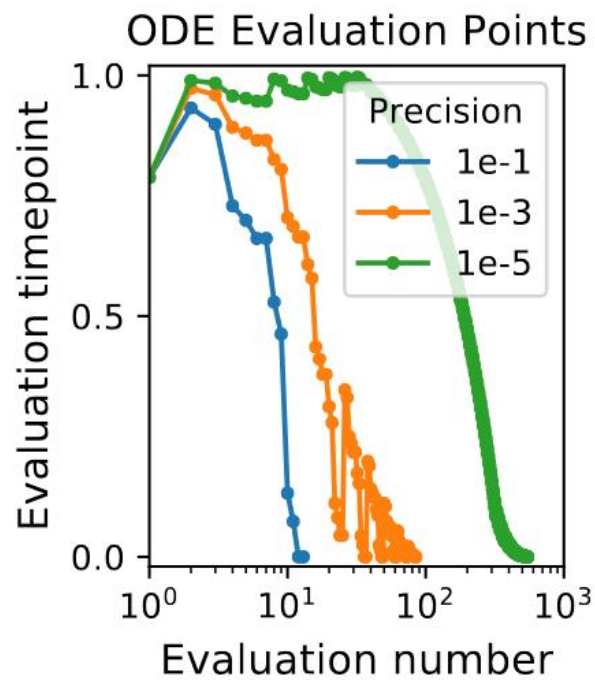


Figure 7: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. “Model A” and “Model B” are separately trained with different architectures.

# ODE solvers enable faster sampling and controllable error tolerance



# Architectural improvements – NCSN++/DDPM++

1. Upsampling and downsampling images with anti-aliasing based on Finite Impulse Response (FIR) (Zhang, 2019). We follow the same implementation and hyper-parameters in StyleGAN-2 (Karras et al., 2020b).
2. Rescaling all skip connections by  $1/\sqrt{2}$ . This has been demonstrated effective in several best-in-class GAN models, including ProgressiveGAN (Karras et al., 2018), StyleGAN (Karras et al., 2019) and StyleGAN-2 (Karras et al., 2020b).
3. Replacing the original residual blocks in DDPM with residual blocks from BigGAN (Brock et al., 2018).
4. Increasing the number of residual blocks per resolution from 2 to 4.
5. Incorporating progressive growing architectures. We consider two progressive architectures for input: “input skip” and “residual”, and two progressive architectures for output: “output skip” and “residual”. These progressive architectures are defined and implemented according to StyleGAN-2.
6. Switching to the continuous training objective and timestep conditioning on random Fourier features improves sample quality

# Improved performance with architectural improvements

Probability flow ODE

Table 2: NLLs and FIDs (ODE) on CIFAR-10.

Model	NLL Test ↓	FID ↓
RealNVP (Dinh et al., 2016)	3.49	-
iResNet (Behrmann et al., 2019)	3.45	-
Glow (Kingma & Dhariwal, 2018)	3.35	-
MintNet (Song et al., 2019b)	3.32	-
Residual Flow (Chen et al., 2019)	3.28	46.37
FFJORD (Grathwohl et al., 2018)	3.40	-
Flow++ (Ho et al., 2019)	3.29	-
DDPM ( $L$ ) (Ho et al., 2020)	$\leq 3.70^*$	13.51
DDPM ( $L_{\text{simple}}$ ) (Ho et al., 2020)	$\leq 3.75^*$	3.17
DDPM	3.28	3.37
DDPM cont. (VP)	3.21	3.69
DDPM cont. (sub-VP)	3.05	3.56
DDPM++ cont. (VP)	3.16	3.93
DDPM++ cont. (sub-VP)	3.02	3.16
DDPM++ cont. (deep, VP)	3.13	3.08
DDPM++ cont. (deep, sub-VP)	<b>2.99</b>	<b>2.92</b>

SDE

Table 3: CIFAR-10 sample quality.

Model	FID↓	IS↑
<b>Conditional</b>		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	<b>2.42</b>	<b>10.14</b>
<b>Unconditional</b>		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	$8.87 \pm .12$
NCSNv2 (Song & Ermon, 2020)	10.87	$8.40 \pm .07$
DDPM (Ho et al., 2020)	3.17	$9.46 \pm .11$
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	<b>2.20</b>	<b>9.89</b>

# Controllable generation

Reverse-time SDE for controllable guidance:

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

Class-conditional sampling with guidance from a time-dependent classifier

Can be extended to inpainting and colorization



# Examples of controllable generation

Class-conditional



Inpainting



Colorization



# Denoising Diffusion Implicit Models

# Reviewing DDPM yet again!

Notation difference

$$\begin{aligned} q(\mathbf{x}_t | \mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \rightarrow \\ q(\mathbf{x}_t | \mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t / \alpha_{t-1}} \mathbf{x}_{t-1}, (1 - \alpha_t / \alpha_{t-1}) \mathbf{I}) \end{aligned}$$

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \rightarrow \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t} \mathbf{x}_0, (1 - \alpha_t) \mathbf{I})$$

$$\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



# Let us analyze the loss function

Our loss function:

$$L_\gamma(\epsilon_\theta) := \sum_{t=1}^T \gamma_t \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ \|\epsilon_\theta^{(t)}(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon_t) - \epsilon_t\|_2^2 \right]$$

We can see it only depends on  $q(\mathbf{x}_t | \mathbf{x}_0)$  and not directly on  $q(\mathbf{x}_{1:T} | \mathbf{x}_0)$

So we can try to find inference processes (non-Markovian) that use the same marginal distribution!

# Non-Markovian forward processes

A family of joint distributions indexed by a real vector  $\sigma$ :

$$q_{\sigma}(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := q_{\sigma}(\mathbf{x}_T \mid \mathbf{x}_0) \prod_{t=2}^T q_{\sigma}(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$
$$q_{\sigma}(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1 - \alpha_t}}, \sigma^2 \mathbf{I}\right)$$

This form is chosen since it ensures  $q_{\sigma}(\mathbf{x}_T \mid \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\mathbf{x}_0, (1 - \alpha_T)\mathbf{I})$

# Non-Markovian forward processes

The forward process is therefore given by Bayes' rule:

$$q_{\sigma}(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q_{\sigma}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q_{\sigma}(\mathbf{x}_t | \mathbf{x}_0)}{q_{\sigma}(\mathbf{x}_{t-1} | \mathbf{x}_0)}$$

Note that the forward process is no longer Markovian. Additionally, note that the magnitude of  $\sigma$  controls how stochastic the process is. It becomes completely deterministic at  $\sigma = 0$ .

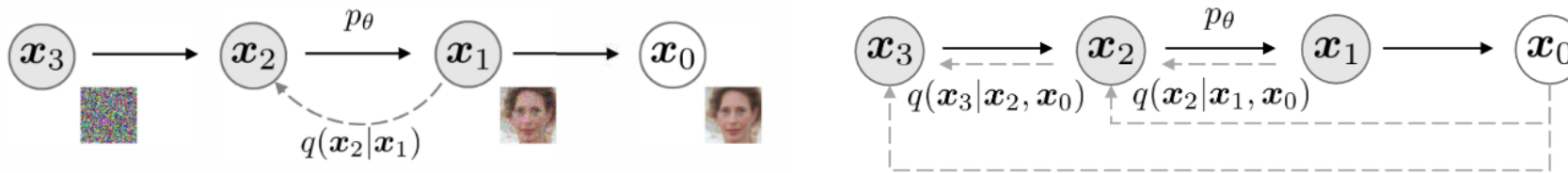


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

# Key insight

If we want to train our reverse process model  $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ , we get the following objective:

$$\begin{aligned} J_\sigma(\epsilon_\theta) &:= \mathbb{E}_{\mathbf{x}_{0:T} \sim q_\sigma(\mathbf{x}_{0:T})} [\log q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) - \log p_\theta(\mathbf{x}_{0:T})] \\ &= \mathbb{E}_{\mathbf{x}_{0:T} \sim q_\sigma(\mathbf{x}_{0:T})} \left[ \log q_\sigma(\mathbf{x}_T|\mathbf{x}_0) + \sum_{t=2}^T \log q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) - \sum_{t=1}^T \log p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) - \log p_\theta(\mathbf{x}_T) \right] \end{aligned} \quad (11)$$

However it can be shown that if the model is time-dependent, minimizing this objective is equivalent to minimizing the simplified DDPM objective!

DDIM is simply a novel sampling process for diffusion models!

# DDIM sampling

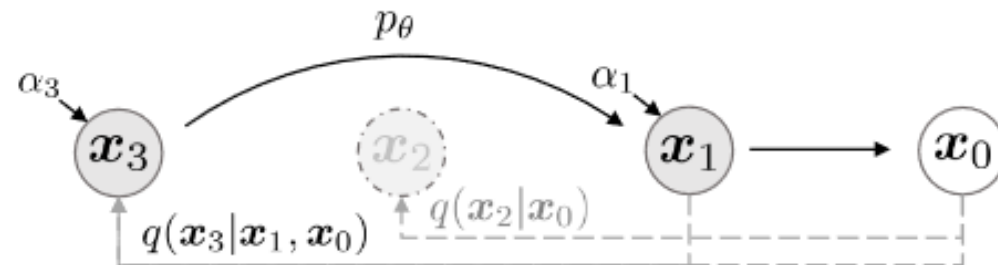
Sampling procedure:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \left( \frac{\mathbf{x}_t - \sqrt{1 - \alpha_t} \epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \epsilon_\theta(\mathbf{x}_t, t) + \sigma_t \epsilon_t$$

where  $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Specific parameterization of  $\sigma_t$  gives DDPM and  $\sigma_t = 0$  gives DDIM.

Accelerated generation:



# Continuous version of DDIM

Rewrite the previous deterministic iterative procedure as such:

$$\frac{\mathbf{x}_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} = \frac{\mathbf{x}_t}{\sqrt{\alpha_t}} + \left( \sqrt{\frac{1 - \alpha_{t-\Delta t}}{\alpha_{t-\Delta t}}} - \sqrt{\frac{1 - \alpha_t}{\alpha_t}} \right) \epsilon_{\theta}^{(t)}(\mathbf{x}_t)$$

Setting  $(\sqrt{1 - \alpha}/\alpha) = \sigma$  and  $\mathbf{x}/\sqrt{\alpha} = \bar{\mathbf{x}}$ , this is an Euler integration of the following ODE:

$$d\bar{\mathbf{x}}(t) = \epsilon_{\theta}^{(t)} \left( \frac{\bar{\mathbf{x}}(t)}{\sqrt{\sigma^2 + 1}} \right) d\sigma(t),$$

# Continuous version of DDIM

The ODE is equivalent to Song et al.'s probability flow ODE for the VE SDE, but their discretization is not equivalent:

$$\frac{\mathbf{x}_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} = \frac{\mathbf{x}_t}{\sqrt{\alpha_t}} + \frac{1}{2} \left( \frac{1 - \alpha_{t-\Delta t}}{\alpha_{t-\Delta t}} - \frac{1 - \alpha_t}{\alpha_t} \right) \cdot \sqrt{\frac{\alpha_t}{1 - \alpha_t}} \cdot \epsilon_{\theta}^{(t)}(\mathbf{x}_t)$$

Just like Song et al.'s ODEs, DDIM ODE provides unique identifiable encodings

# Experiments and Results

## Pretrained DDPM models used!

Table 1: CIFAR10 and CelebA image generation measured in FID.  $\eta = 1.0$  and  $\hat{\sigma}$  are cases of **DDPM** (although [Ho et al. \(2020\)](#) only considered  $T = 1000$  steps, and  $S < T$  can be seen as simulating DDPMs trained with  $S$  steps), and  $\eta = 0.0$  indicates **DDIM**.

$S$	CIFAR10 ( $32 \times 32$ )					CelebA ( $64 \times 64$ )					
	10	20	50	100	1000	10	20	50	100	1000	
$\eta$	0.0	<b>13.36</b>	<b>6.84</b>	<b>4.67</b>	<b>4.16</b>	4.04	<b>17.33</b>	<b>13.73</b>	<b>9.17</b>	<b>6.53</b>	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$	367.43	133.37	32.72	9.99	<b>3.17</b>	299.71	183.83	71.71	45.20	<b>3.26</b>	

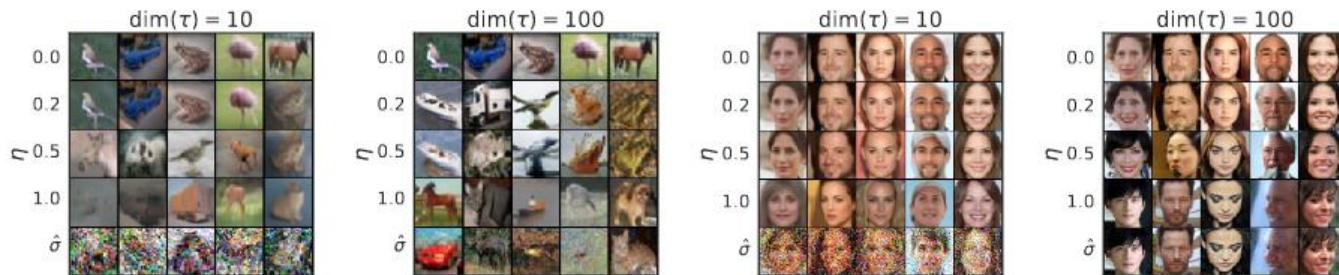


Figure 3: CIFAR10 and CelebA samples with  $\dim(\tau) = 10$  and  $\dim(\tau) = 100$ .

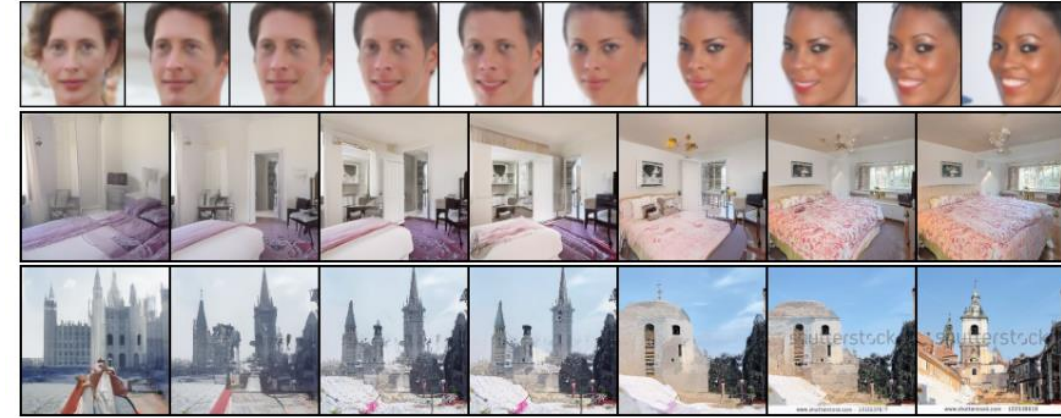


Figure 6: Interpolation of samples from DDIM with  $\dim(\tau) = 50$ .

Table 2: Reconstruction error with DDIM on CIFAR-10 test set, rounded to  $10^{-4}$ .

$S$	10	20	50	100	200	500	1000
Error	0.014	0.0065	0.0023	0.0009	0.0004	0.0001	0.0001