Denoising Diffusion Implicit Models

Jiaming Song, Chenlin Meng, Stefano Ermon

ICLR 2021

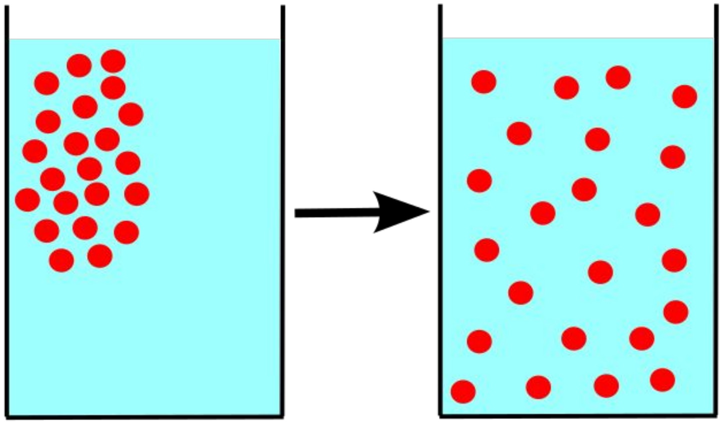
Table of Contents

What is Diffusion Model?

What is Diffusion Model?

Diffusion

- Diffusion is the net movement of anything (for example, atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.



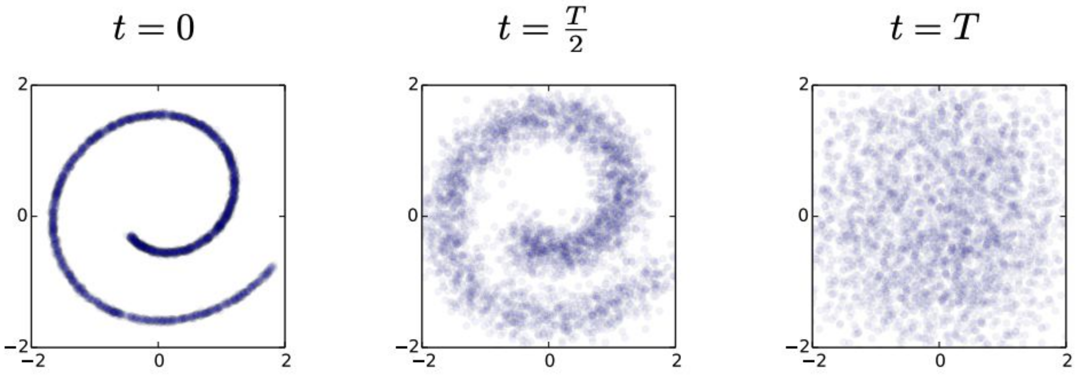
Reference: https://en.wikipedia.org/wiki/Diffusion

What is Diffusion Model?

Diffusion Process

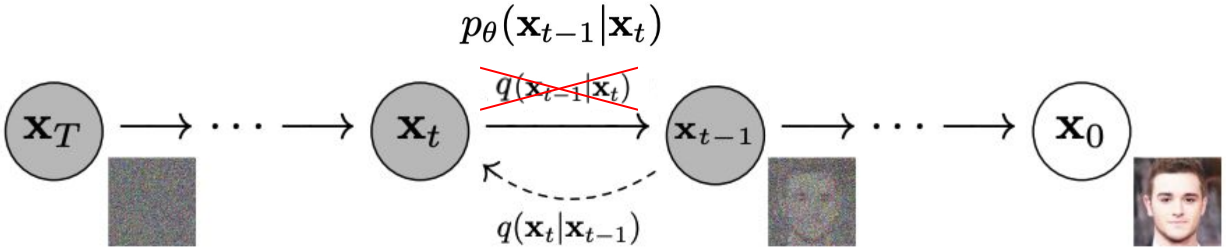
- A (forward) diffusion process converts any complex data distribution into a simple, tractable, distribution.

$$q\left(\mathbf{x}^{(0\cdots T)}\right)$$



Reference: Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML, 2015

Diffusion Model



Diffusion Process

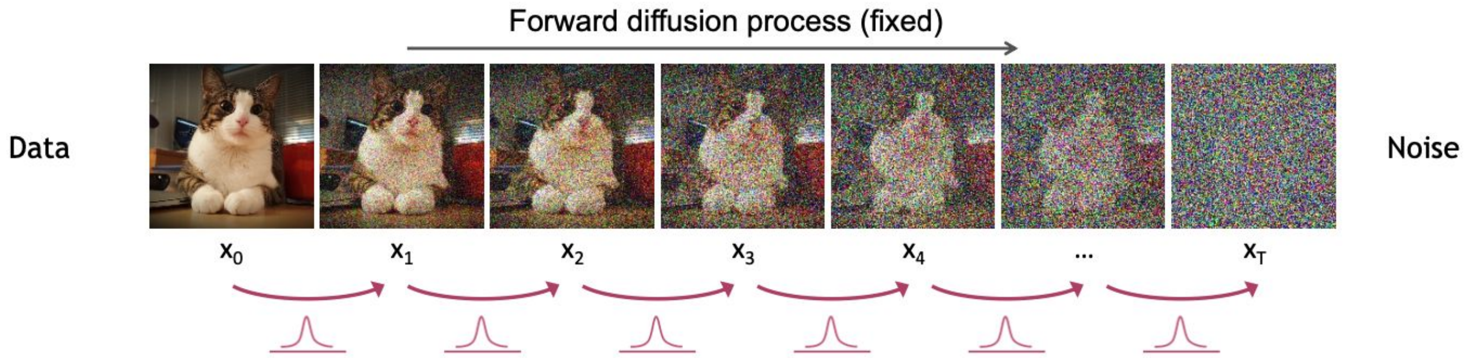
Reverse Process

- Diffusion Process: gradually adds noise to input
- Reverse Process: learns to generate data by denoising

Reference: Denoising Diffusion Probabilistic Models, NeurIPS, 2020

Denoising Diffusion Probabilistic Models

Diffusion Process



$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

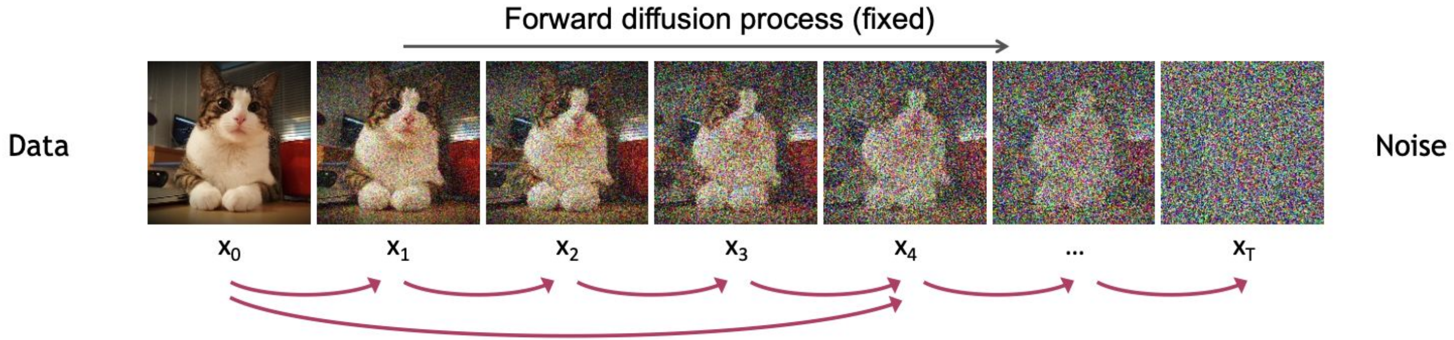
β_t increased linearly: $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$

Reference: https://cvpr2023-tutorial-diffusion-models.github.io/

Denoising Diffusion Probabilistic Models

Diffusion Process

- How to sample x_t?



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$\alpha_t := 1 - \beta_t \text{ and } \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

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

Reparameterization

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$$

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

Reference: https://cvpr2023-tutorial-diffusion-models.github.io/

MI2RL Medical Imaging Intelligent Reality Lab

Diffusion Process

- Codes

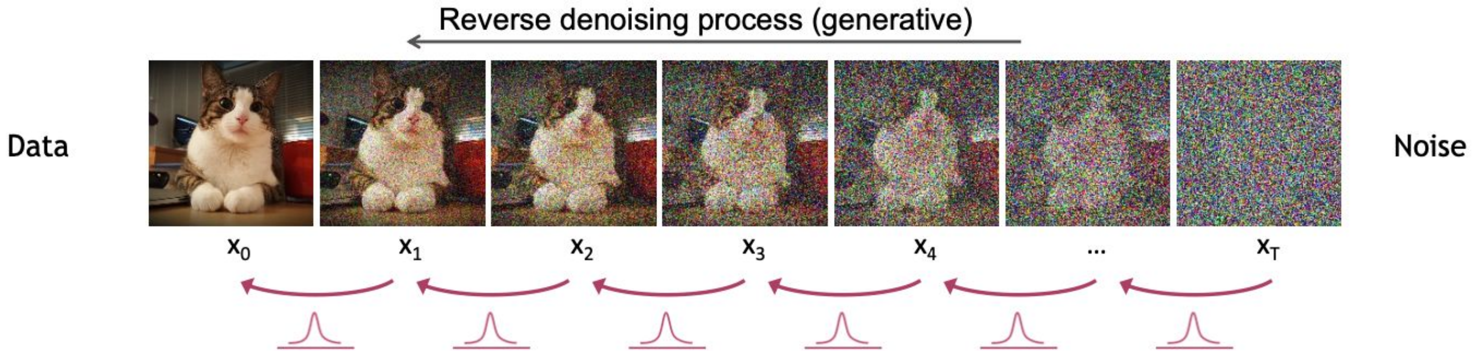
```
alphas = 1. - betas
alphas_cumprod = torch.cumprod(alphas, dim=0)
register_buffer('sqrt_alphas_cumprod', torch.sqrt(alphas_cumprod))
register_buffer('sqrt_one_minus_alphas_cumprod', torch.sqrt(1. - alphas_cumprod))
```

```
@autocast(enabled = False)
def q_sample(self, x_start, t, noise = None):
    noise = default(noise, lambda: torch.randn_like(x_start))

    return (
        extract(self.sqrt_alphas_cumprod, t, x_start.shape) * x_start +
        extract(self.sqrt_one_minus_alphas_cumprod, t, x_start.shape) * noise
    )
```

Reference: https://github.com/lucidrains/denoising-diffusion-pytorch/blob/main/denoising_diffusion_pytorch/denoising_diffusion_pytorch.py#L757

Reverse Process



$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)}_{\text{Trainable network (U-net, Denoising Autoencoder)}}, \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

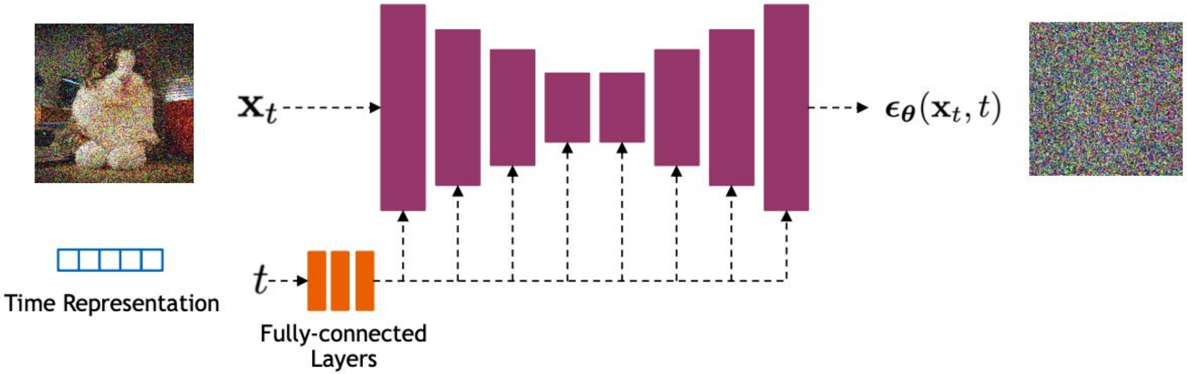
$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2$$

$$\sigma_t^2 = \beta_t$$

Reference: https://cvpr2023-tutorial-diffusion-models.github.io/

Reverse Process with Diffusion Model

- Diffusion model consists of U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$.



- We can generate $\mathbf{x}_{t-1}$ image with $\mathbf{x}_t$ and $\epsilon_\theta(\mathbf{x}_t, t)$.

Reference: https://cvpr2023-tutorial-diffusion-models.github.io/

Medical Imaging Intelligent Reality Lab

Reverse Process with Predicted Noise

Bayes' Rule

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \times q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid x_0)}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

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

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

$$\text{where} \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \frac{1}{\sqrt{2\pi\beta_t}}\exp\left(-\frac{(\mathbf{x}_t - \sqrt{1-\beta_t}\mathbf{x}_{t-1})^2}{2\beta_t}\right)$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \frac{1}{\sqrt{2\pi(1-\bar{\alpha}_t)}}\exp\left(-\frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{2(1-\bar{\alpha}_t)}\right)$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \frac{1}{\sqrt{2\pi(1-\bar{\alpha}_{t-1})}}\exp\left(-\frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{2(1-\bar{\alpha}_{t-1})}\right)$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \approx \frac{1}{\sqrt{2\pi\underbrace{\boxed{\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}}_{\tilde{\beta}_t}}}\exp\left(-\frac{1}{2\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}\left[\mathbf{x}_{t-1} - \underbrace{\boxed{\left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}\mathbf{x}_t\right)}}_{\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)}\right]^2\right)$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

Reference: https://cvpr2023-tutorial-diffusion-models.github.io/

Medical Imaging Intelligent Reality Lab

Reverse Process with Predicted Noise

Set as β_t

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boxed{\mathbf{x}_0} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \boxed{\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t}$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon \qquad \text{where} \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \longrightarrow \boxed{x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \times x_t - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \times \epsilon}$$

- Above equation also can be used for predicted $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$

Reference: https://cvpr2023-tutorial-diffusion-models.github.io/

MI2RL Medical Imaging Intelligent Reality Lab

Reverse Process

```python
@torch.inference_mode()
def p_sample(self, x, t: int, x_self_cond = None):
    b, *_, device = *x.shape, self.device
    batched_times = torch.full((b,), t, device = device, dtype = torch.long)
    model_mean, _, model_log_variance, x_start = self.p_mean_variance(x = x, t = batched_times, x_self_cond = x_self_cond, clip_denoised = True)
    noise = torch.randn_like(x) if t > 0 else 0. # no noise if t == 0
    pred_img = model_mean + (0.5 * model_log_variance).exp() * noise
    return pred_img, x_start

@torch.inference_mode()
def p_sample_loop(self, shape, return_all_timesteps = False):
    batch, device = shape[0], self.device

    img = torch.randn(shape, device = device)
    imgs = [img]

    x_start = None

    for t in tqdm(reversed(range(0, self.num_timesteps)), desc = 'sampling loop time step', total = self.num_timesteps):
        self_cond = x_start if self.self_condition else None
        img, x_start = self.p_sample(img, t, self_cond)
        imgs.append(img)

    ret = img if not return_all_timesteps else torch.stack(imgs, dim = 1)

    ret = self.unnormalize(ret)
    return ret
```

Reference: https://github.com/lucidrains/denoising-diffusion-pytorch/blob/main/denoising_diffusion_pytorch/denoising_diffusion_pytorch.py#L660-L686

Diffusion Loss

Variational Bound

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \boxed{\mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]}$$

Proof: Appendix A

$$\mathbb{E}_q\Bigg[\cancel{\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}_{L_T}} + \boxed{\sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}} \cancel{\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}}\Bigg]$$

Denoising Matching

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

$$L_{\mathrm{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Reference: Denoising Diffusion Probabilistic Models, NeurIPS, 2020

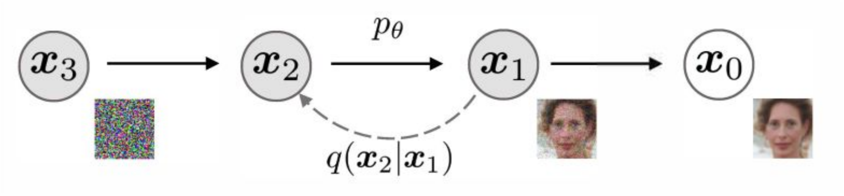
Denoising Diffusion Implicit Model

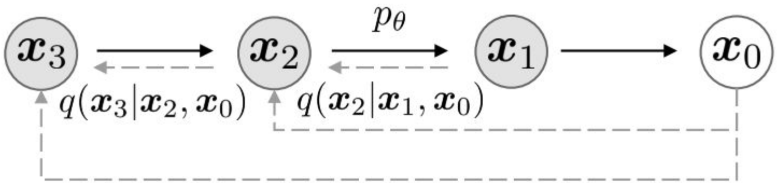
Main Idea

- Denoising diffusion probabilistic models (DDPMs) have achieved high quality image generation without adversarial training, yet they **require simulating a Markov chain for many steps in order to produce a sample**.
- Denoising diffusion implicit models (DDIMs) used **Non-Markovian processes** that can correspond to generative processes that are deterministic, giving rise to implicit models that **produce high quality samples much faster**.

Denoising Diffusion Implicit Model

Non-Markovian Diffusion Process





$$\boxed{q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$$

$$\cancel{q(x_1|x_0)}q(x_2|x_1, x_0)q(x_3|x_2, x_0)\ldots q(x_T|x_{T-1}, x_0)$$

Bayes' Rule

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

$$\frac{q_\sigma(x_1|x_2, x_0)\cancel{q_\sigma(x_2|x_0)}}{\cancel{q_\sigma(x_1|x_0)}} \times \frac{q_\sigma(x_2|x_3, x_0)\cancel{q_\sigma(x_3|x_0)}}{\cancel{q_\sigma(x_2|x_0)}} \times \cdots \times \frac{q_\sigma(x_{T-1}|x_T, x_0)q_\sigma(x_T|x_0)}{\cancel{q_\sigma(x_{T-1}|x_0)}}$$

Reference: Denoising Diffusion Implicit Model, ICLR, 2021

Non-Markovian Diffusion Process

$$q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$$

Proof: Appendix B

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\boldsymbol{I}\right)$$

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{" predicted } \boldsymbol{x}_0\text{"}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{"direction pointing to } \boldsymbol{x}_t\text{"}} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}}$$

Based on DDPM

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \times x_t - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \times \epsilon_\theta(x_t, t)$$

$$\epsilon_\theta(x_t, t) = \frac{x_t - \sqrt{\bar{\alpha}_t} \times x_0}{\sqrt{1-\bar{\alpha}_t}}$$

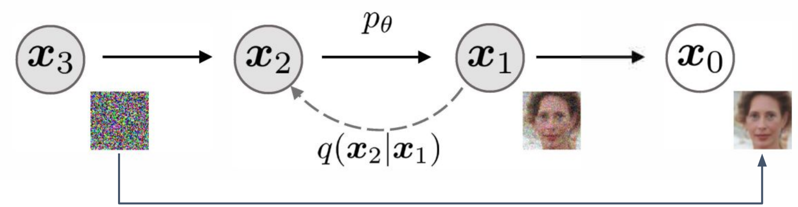
$$x_t = \sqrt{\bar{\alpha}_t} \times x_0 - \sqrt{1-\bar{\alpha}_t} \times \epsilon_\theta(x_t, t)$$

Reference: Denoising Diffusion Implicit Model, ICLR, 2021

Deterministic Generative Process

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\,\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}} \right)}_{\text{“ predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}} + \underbrace{\cancel{\sigma_t \epsilon_t}}_{\text{random noise}}$$

$\sigma_t = 0 \longrightarrow$



1:1 Mapping

Accelerated Generation Processes

$$q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$$

Proof: Appendix C.1

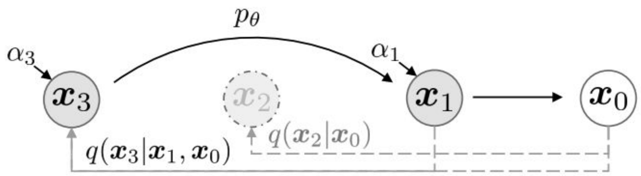


Figure 2: Graphical model for accelerated generation, where $\tau = [1, 3]$.

In the previous sections, the generative process is considered as the approximation to the reverse process; since of the forward process has T steps, the generative process is also forced to sample T steps. However, as the denoising objective L_1 does not depend on the specific forward procedure as long as $q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is fixed, we may also consider forward processes with lengths smaller than T, which accelerates the corresponding generative processes without having to train a different model.

Experiments

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 (32×32) 10	20	50	100	1000	CelebA (64×64) 10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	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	**3.17**	299.71	183.83	71.71	45.20	**3.26**

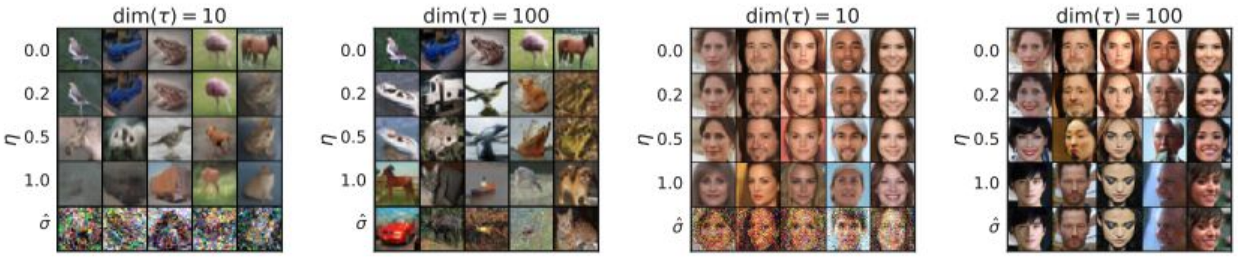
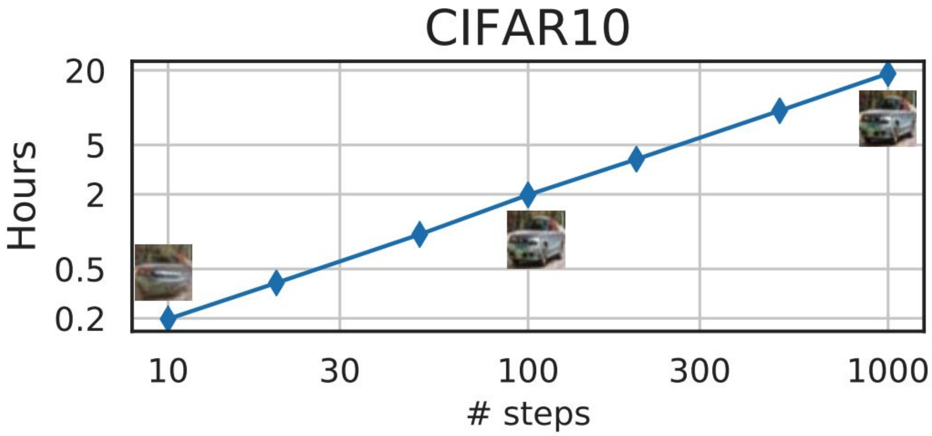
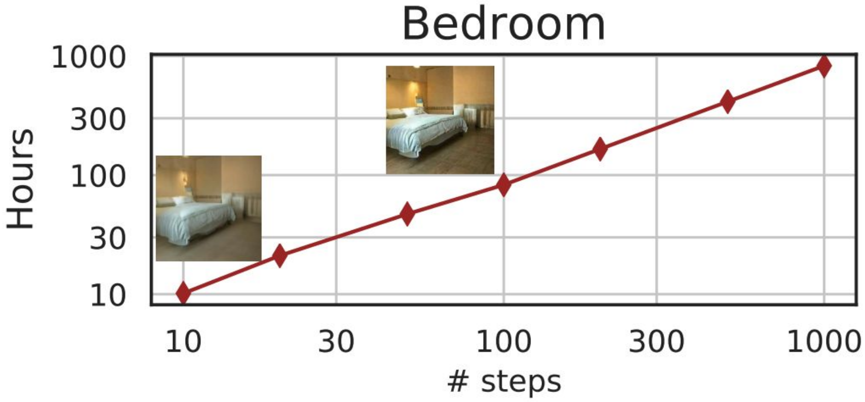


Figure 3: CIFAR10 and CelebA samples with $\dim(\tau) = 10$ and $\dim(\tau) = 100$.

Experiments





Experiments

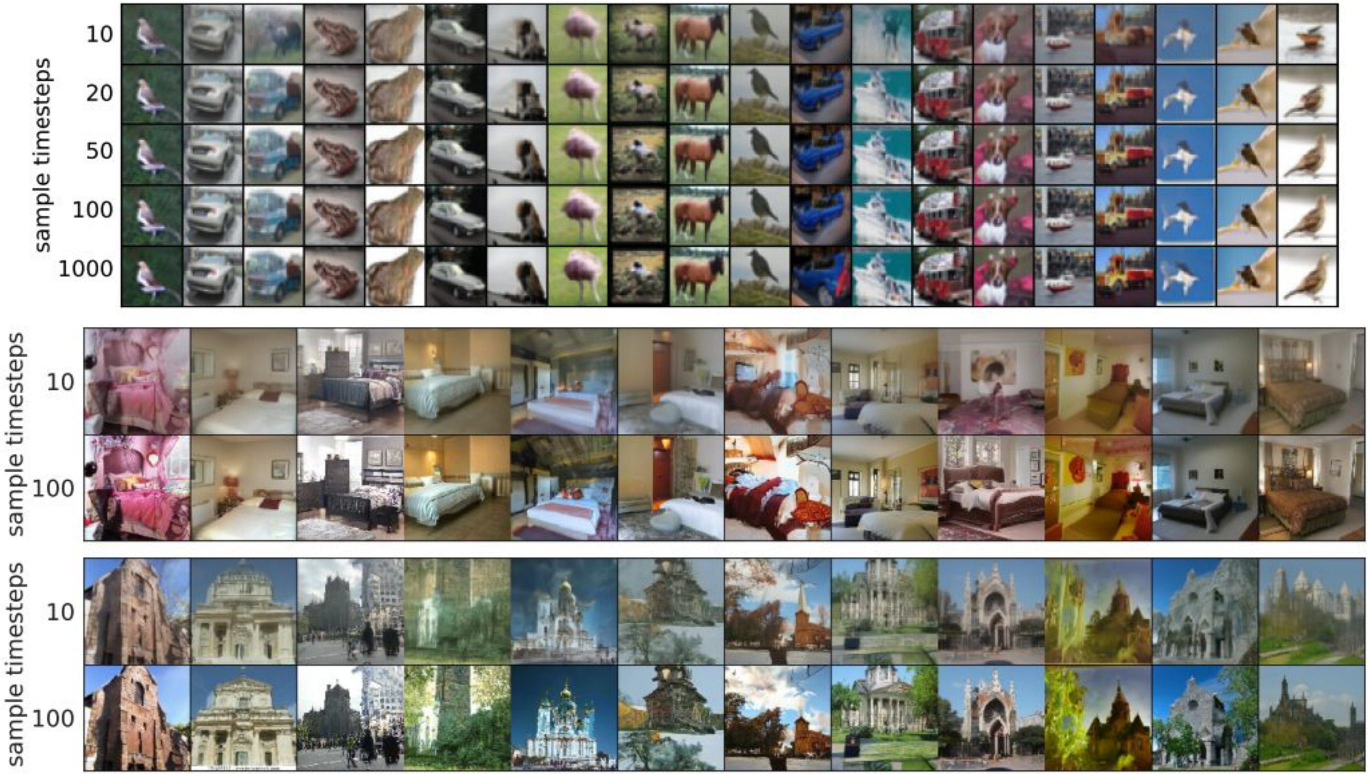


Figure 5: Samples from DDIM with the same random x_T and different number of steps.

Experiments

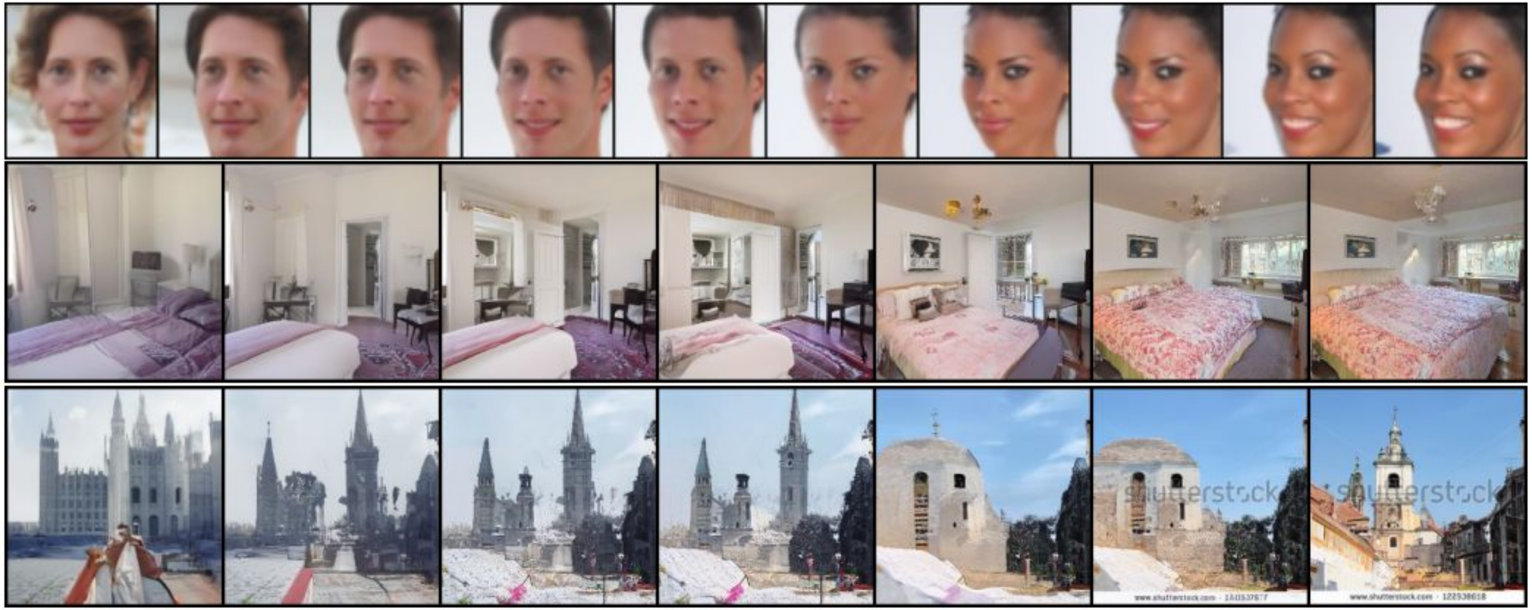


Figure 6: Interpolation of samples from DDIM with $\dim(\tau) = 50$.

Conclusions

Conclusions

- DDIMs can **produce high quality samples 10× to 50× faster** in terms of wall-clock time compared to DDPMs, allow us to trade off computation for sample quality applying non-Markovian processes.

- After **training with DDPM**, we can use **DDIM for inference**, because they have the same training procedure.

ASAN
Medical Center

Thank you

MI2RL
Medical
Imaging
Intelligent
Reality
Lab

Appendix

Appendix C.1

C.1 Accelerated sampling processes

In the accelerated case, we can consider the inference process to be factored as:

$$q_{\sigma,\tau}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = q_{\sigma,\tau}(\boldsymbol{x}_{\tau_S}|\boldsymbol{x}_0)\prod_{i=1}^{S} q_{\sigma,\tau}(\boldsymbol{x}_{\tau_{i-1}}|\boldsymbol{x}_{\tau_i}, \boldsymbol{x}_0)\prod_{t\in\bar{\tau}} q_{\sigma,\tau}(\boldsymbol{x}_t|\boldsymbol{x}_0) \tag{52}$$

where τ is a sub-sequence of $[1, \ldots, T]$ of length S with $\tau_S = T$, and let $\bar{\tau} := \{1, \ldots, T\} \setminus \tau$ be its complement. Intuitively, the graphical model of $\{\boldsymbol{x}_{\tau_i}\}_{i=1}^{S}$ and $\boldsymbol{x}_0$ form a chain, whereas the graphical model of $\{\boldsymbol{x}_t\}_{t\in\bar{\tau}}$ and $\boldsymbol{x}_0$ forms a star graph. We define:

$$q_{\sigma,\tau}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\boldsymbol{I}) \quad \forall t \in \bar{\tau} \cup \{T\} \tag{53}$$

$$q_{\sigma,\tau}(\boldsymbol{x}_{\tau_{i-1}}|\boldsymbol{x}_{\tau_i}, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{\tau_{i-1}}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{\tau_{i-1}} - \sigma_{\tau_i}^2}\cdot\frac{\boldsymbol{x}_{\tau_i} - \sqrt{\alpha_{\tau_i}}\boldsymbol{x}_0}{\sqrt{1-\alpha_{\tau_i}}}, \sigma_{\tau_i}^2\boldsymbol{I}\right) \quad \forall i \in [S]$$

where the coefficients are chosen such that:

$$q_{\sigma,\tau}(\boldsymbol{x}_{\tau_i}|\boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_{\tau_i}}\boldsymbol{x}_0, (1-\alpha_{\tau_i})\boldsymbol{I}) \quad \forall i \in [S] \tag{54}$$

i.e., the "marginals" match.

Appendix C.1

The corresponding "generative process" is defined as:

$$p_\theta(\boldsymbol{x}_{0:T}) := \underbrace{p_\theta(\boldsymbol{x}_T) \prod_{i=1}^{S} p_\theta^{(\tau_i)}(\boldsymbol{x}_{\tau_{i-1}} | \boldsymbol{x}_{\tau_i})}_{\text{use to produce samples}} \times \underbrace{\prod_{t \in \bar{\tau}} p_\theta^{(t)}(\boldsymbol{x}_0 | \boldsymbol{x}_t)}_{\text{in variational objective}} \tag{55}$$

where only part of the models are actually being used to produce samples. The conditionals are:

$$p_\theta^{(\tau_i)}(\boldsymbol{x}_{\tau_{i-1}} | \boldsymbol{x}_{\tau_i}) = q_{\sigma,\tau}(\boldsymbol{x}_{\tau_{i-1}} | \boldsymbol{x}_{\tau_i}, f_\theta^{(\tau_i)}(\boldsymbol{x}_{\tau_{i-1}})) \quad \text{if } i \in [S], i > 1 \tag{56}$$

$$p_\theta^{(t)}(\boldsymbol{x}_0 | \boldsymbol{x}_t) = \mathcal{N}(f_\theta^{(t)}(\boldsymbol{x}_t), \sigma_t^2 \boldsymbol{I}) \quad \text{otherwise,} \tag{57}$$

where we leverage $q_{\sigma,\tau}(\boldsymbol{x}_{\tau_{i-1}} | \boldsymbol{x}_{\tau_i}, \boldsymbol{x}_0)$ as part of the inference process (similar to what we have done in Section 3). The resulting variational objective becomes (define $\boldsymbol{x}_{\tau_{L+1}} = \varnothing$ for conciseness):

$$J(\epsilon_\theta) = \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q_{\sigma,\tau}(\boldsymbol{x}_{0:T})}[\log q_{\sigma,\tau}(\boldsymbol{x}_{1:T} | \boldsymbol{x}_0) - \log p_\theta(\boldsymbol{x}_{0:T})] \tag{58}$$

$$= \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q_{\sigma,\tau}(\boldsymbol{x}_{0:T})}\Bigg[\sum_{t \in \bar{\tau}} D_{\mathrm{KL}}(q_{\sigma,\tau}(\boldsymbol{x}_t | \boldsymbol{x}_0) \| p_\theta^{(t)}(\boldsymbol{x}_0 | \boldsymbol{x}_t) \tag{59}$$

$$+ \sum_{i=1}^{L} D_{\mathrm{KL}}(q_{\sigma,\tau}(\boldsymbol{x}_{\tau_{i-1}} | \boldsymbol{x}_{\tau_i}, \boldsymbol{x}_0) \| p_\theta^{(\tau_i)}(\boldsymbol{x}_{\tau_{i-1}} | \boldsymbol{x}_{\tau_i})))\Bigg]$$

where each KL divergence is between two Gaussians with variance independent of θ. A similar argument to the proof used in Theorem 1 can show that the variational objective J can also be converted to an objective of the form L_γ.