Diffusion Models and Applications

Tri Nguyen

Internal Reading Meeting
Oregon State University

February 24, 2023

Task

Generate new data

$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are sampled i.i.d. from an unknown $\mathcal{P}_{\mathcal{X}}$. How to sample $\boldsymbol{x} \sim \mathcal{P}_{\mathcal{X}}$?

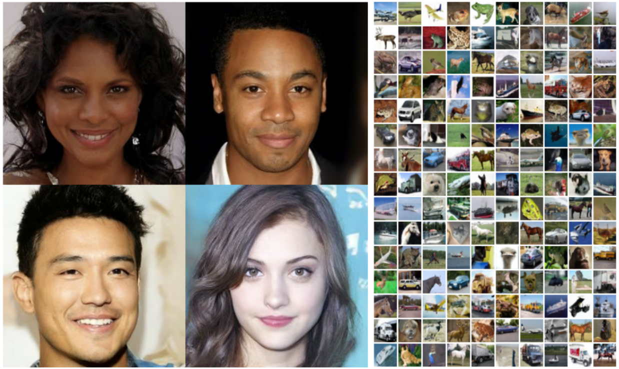

Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Figure: From [Ho et al. 2020].

VAE Approach

VAE [Kingma and Welling 2013] makes some assumptions about family distribution to which $\mathcal{P}_{\mathcal{X}}$ belongs:

- Existence of latent factors $\boldsymbol{z}$: $P(\boldsymbol{x}, \boldsymbol{z}) = P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})$
- $P(\boldsymbol{x}|\boldsymbol{z}) = f_1(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}_1)$ where function class f_1 is some known distribution (in $\boldsymbol{x}$).
- $P(\boldsymbol{z}) = f_2(\boldsymbol{z}; \boldsymbol{\theta}_2)$ where function class f_2 is some known distribution (in $\boldsymbol{z}$).

Maximum likelihood principle suggests to maximize:

$$\log P(\boldsymbol{x}) = \log \left(\sum_{\boldsymbol{z} \in \mathcal{Z}} P(\boldsymbol{x} \mid \boldsymbol{z}) P(\boldsymbol{z}) \right) = \log \left(\sum_{\boldsymbol{z} \in \mathcal{Z}} f_1(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}_1) f_2(\boldsymbol{z}; \boldsymbol{\theta}_2) \right)$$

We can try to maximize its lower bound: For *any* distribution $Q(\boldsymbol{z})$,

$$\log P(\boldsymbol{x}) = D_{\mathrm{kl}}(Q(\boldsymbol{z}) \parallel P(\boldsymbol{z} \mid \boldsymbol{x})) + \mathcal{L}(Q), \quad \text{where } \mathcal{L}(Q) = \underset{\boldsymbol{z} \sim Q(\boldsymbol{z})}{\mathbb{E}} \left[\log \frac{P(\boldsymbol{x}, \boldsymbol{z})}{Q(\boldsymbol{z})} \right]$$

Let $Q(\boldsymbol{z}) = f_3(\boldsymbol{z}; \boldsymbol{\theta}_3)$, and the lower bound can be maximized with respect to $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3$.

Could it be tractable without any assumption on $\mathcal{P}_{\mathcal{X}}$?

Diffusion Model

Given an observed $\boldsymbol{x}_0 \sim \mathcal{P}_{\mathcal{X}}$, define a sequence of RVs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ [1]:

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{z}_{t-1}, \quad t = 1, \ldots, T-1,$$

where

- $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T$ are i.i.d, and $\boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
- $0 < \beta_0, \ldots, \beta_T < 1$ are predefined.

Claims

1. The sequence $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T$ satisfies Markov property: $P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_0) = P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$.
2. If T is large enough, $P(\boldsymbol{x}_T \mid \boldsymbol{x}_0)$ is approximately a Gaussian distribution *regardless* of $\mathcal{P}_{\mathcal{X}}$.
3. The backward direction of the chain also satisfies Markov property. In particular,

$$\boldsymbol{x}_{t-1} = \left(2 - \sqrt{1-\beta_{t+1}}\right)\boldsymbol{x}_t + \beta_t \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t) + \sqrt{\beta_{t+1}}\boldsymbol{z}_t$$

[1] abuse of notation

Implication

Claims

1. The sequence $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T$ satisfies Markov property: $P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_0) = P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$.
2. If T is large enough, $P(\boldsymbol{x}_T \mid \boldsymbol{x}_0)$ is approximately a Gaussian distribution *regardless* of $\mathcal{P}_{\mathcal{X}}$.
3. The backward direction of the chain also satisfies Markov property. In particular,

$$\boldsymbol{x}_{t-1} = \left(2 - \sqrt{1 - \beta_{t+1}}\right) \boldsymbol{x}_t + \beta_t \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t) + \sqrt{\beta_{t+1}} \boldsymbol{z}_t$$

In VAE's world,

- Latent factor $\boldsymbol{z} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$
- Family of $P(\boldsymbol{z})$ is **well defined** without assumption,

$$P(\boldsymbol{z}) = \sum_{\boldsymbol{x}_0 \in \mathcal{X}} P(\boldsymbol{z}, \boldsymbol{x}_0) = \sum_{\boldsymbol{x}_0 \in \mathcal{X}} \underbrace{P(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)}_{\mathcal{N}(\boldsymbol{\mu}_1, \sigma_1^2 \boldsymbol{I})} \cdots \underbrace{P(\boldsymbol{x}_{T-1} \mid \boldsymbol{x}_T)}_{\mathcal{N}(\boldsymbol{\mu}_T, \sigma_T^2 \boldsymbol{I})} \underbrace{P(\boldsymbol{x}_T)}_{\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})}$$

- Family of $P(\boldsymbol{x} \mid \boldsymbol{z})$ is **well defined** without assumption,

$$P(\boldsymbol{x} \mid \boldsymbol{z}) = P(\boldsymbol{x}_0 \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = P(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{\mu}_1, \sigma_1^2 \boldsymbol{I})$$

Proof of property 2

Claims

- If T is large enough, $P(\boldsymbol{x}_T \mid \boldsymbol{x}_0)$ is approximately a Gaussian distribution *regardless* of $\mathcal{P}_{\mathcal{X}}$.

One step transition is

$$P(\boldsymbol{x}_{i+1} \mid \boldsymbol{x}_i) \sim \mathcal{N}(\boldsymbol{x}_i \sqrt{(1-\beta_{i+1})}, \sqrt{\beta_{i+1}} \boldsymbol{I}),$$

similarly, t steps transition is

$$P(\boldsymbol{x}_{i+t} \mid \boldsymbol{x}_i) = \mathcal{N}\left(\boldsymbol{x}_i \sqrt{\prod_{j=i+1}^{i+t} (1-\beta_j)}, \left(1 - \prod_{j=i+1}^{i+t} (1-\beta_j)\right) \boldsymbol{I}\right),$$

Therefore, when T is large enough,

$$P(\boldsymbol{x}_T \mid \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_0 \sqrt{\prod_{j=1}^{T} (1-\beta_j)}, \left(1 - \prod_{j=1}^{T} (1-\beta_j)\right) \boldsymbol{I}\right) \quad \sim \quad \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),$$

Define some constants

$$P(\boldsymbol{x}_{i+t} \mid \boldsymbol{x}_i) = \mathcal{N}\left(\boldsymbol{x}_i \sqrt{\prod_{j=i+1}^{i+t} (1-\beta_j)}, \left(1 - \prod_{j=i+1}^{i+t} (1-\beta_j)\right) \boldsymbol{I}\right),$$

Define some notation

$$\alpha_t \triangleq 1 - \beta_t,$$

$$\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Proof of property 3

- $\boldsymbol{w}(t) \in \mathbb{R}^d$ is the standard Wiener process (or Brownian motion).

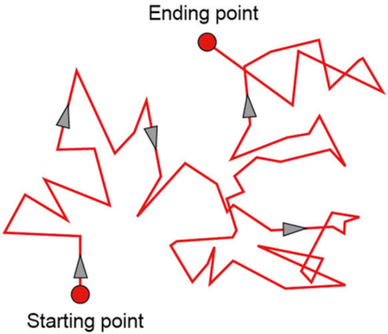

Figure: Brownian motion describes position of a random moving object, e.g., particles in water.

The increment $\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}$ is Gaussian with mean zero and variance $t_2 - t_1$.

Informal proof of property 3

Diffusion process. Given

- $\boldsymbol{x}(t) : \mathbb{R}^+ \to \mathbb{R}^d$ is a function of $t \geq 0$.
- $\boldsymbol{w}(t) \in \mathbb{R}^d$ is the standard Wiener process.
- $\boldsymbol{f}(\boldsymbol{x}, t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$: drift coefficient of $\boldsymbol{x}(t)$.
- $g(\cdot) : \mathbb{R} \to \mathbb{R}$: diffusion coefficient of $\boldsymbol{x}(t)$.

then a diffusion process is governed by a stochastic differential equation (SDE)

$$x(0) \sim \mathcal{P}_{\boldsymbol{X}} \tag{1a}$$

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)dt + g(t)d\boldsymbol{w} \tag{1b}$$

By starting from samples of $\boldsymbol{x}_T \sim p_T$, and reverse process, we can obtain $\boldsymbol{x}(0) \sim \mathcal{P}_{\mathcal{X}}$.
Remarkable result from [x]: the reverse process is also a diffusion process, i.e,

$$\boldsymbol{x}(T) \sim p_T \tag{2a}$$

$$d\boldsymbol{x}(t) = \left(f(\boldsymbol{x}(t), t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}(t))\right) dt + g(t)d\overline{\boldsymbol{w}}, \tag{2b}$$

where $\overline{\boldsymbol{w}}$ is another standard Wiener process.

Proof of property 3

Based on Yang Song et al. "Score-based generative modeling through stochastic differential equations". In: *arXiv preprint arXiv:2011.13456* [2020].

Proof.

- Discrete the forward SDE $d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)dt + g(t)d\boldsymbol{w}$:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{f}_t(\boldsymbol{x}) + g_t.$$

- By choosing $\boldsymbol{f}_t(\boldsymbol{x}) \triangleq \left(\sqrt{1-\beta_{t+1}} - 1\right)\boldsymbol{x}, \quad g_t \triangleq \sqrt{\beta_{t+1}}$,

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{z}_{t-1}, \quad t = 1, \ldots, T-1. \quad \text{(our original chain)}$$

- Discrete the backward SDE $d\boldsymbol{x}(t) = \left(\boldsymbol{f}(\boldsymbol{x}(t), t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}(t))\right) dt + g(t)d\overline{\boldsymbol{w}}$:

$$\boldsymbol{x}_{t-1} = \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t) + g_t^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t) + g_t \boldsymbol{z}_t.$$

- Plug in $\boldsymbol{f}_t, g_t$:

$$\boldsymbol{x}_{t-1} = (2 - \sqrt{1-\beta_{t+1}})\boldsymbol{x}_t + \beta_{t+1}\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t) + \sqrt{\beta_{t+1}}\boldsymbol{z}_t$$

The SDE Framework

Forward SDE

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

Backward SDE

$$d\boldsymbol{x}(t) = \left(\boldsymbol{f}(\boldsymbol{x}(t), t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right) dt + g(t)d\overline{\boldsymbol{w}}$$

By choosing $\boldsymbol{f}(\boldsymbol{x}, t), g(t)$, we can design various diffusion process where $\boldsymbol{x}_T \sim p_T$ is in our control.
The remaining is to learn score function

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

- Time t is uniform sampled over $[0, T]$
- $\lambda(t) : [0, T] \leftarrow \mathbb{R}^+$ is a weighting function
- $\boldsymbol{x}(0) \sim \mathcal{P}_{\mathcal{X}}$ and $\boldsymbol{x}(t) \sim P(\boldsymbol{x}(t) \mid \boldsymbol{x}(0))$ where $P(\boldsymbol{x}(t) \mid \boldsymbol{x}(0))$ is Gaussian if $\boldsymbol{f}(\boldsymbol{x}, t)$ is affine in $\boldsymbol{x}$.
- Expressiveness power of deep neural network is fully exploited here.

The SDE Framework

Forward SDE

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

Backward SDE

$$d\boldsymbol{x}(t) = \big(\boldsymbol{f}(\boldsymbol{x}(t), t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\big)\, dt + g(t)d\overline{\boldsymbol{w}}$$

Once $\boldsymbol{s}_{\boldsymbol{\theta}^*}(\boldsymbol{x}, t)$ is learned, we can derive the reverse diffusion process from the backward SDE

$$d\boldsymbol{x}(t) = \big(\boldsymbol{f}(\boldsymbol{x}(t), t) - g(t)^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\big)\, dt + g(t)d\overline{\boldsymbol{w}}$$

and simulate it to sample $\boldsymbol{x}_0 \sim \mathcal{P}_{\mathcal{X}}$.

- Solve the backward SDE using numerical SDE solver
- Ancestor sampling method ...

The whole training and parameterization can be implemented under a probabilistic model, like in VAE.

Training Process Under Probabilistic View

Based on Jonathan Ho et al. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 [2020], pp. 6840–6851.
Define a generative model $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_1 \leftarrow \ldots \leftarrow \boldsymbol{x}_T$ as

$$P(\boldsymbol{x}_T) \triangleq \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{3}$$

$$P_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T) \triangleq P(\boldsymbol{x}_T) \prod_{t=1}^{T} P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t), \quad P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) \triangleq \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)) \tag{4}$$

$$P(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T \mid \boldsymbol{x}_0) \triangleq \prod_{t=1}^{T} P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}), \quad P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \triangleq \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta, \boldsymbol{I}) \tag{5}$$

With large T, there always exists $\boldsymbol{\theta}$ such that $\boldsymbol{x}_0 \sim \mathcal{P}_{\boldsymbol{X}}, \boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
How to perform inference on this model efficiently?

Diffusion Model Inference

By maximum likelihood principle, we want to minimize

$$\mathop{\mathbb{E}}_{\boldsymbol{x}_0 \sim P_{\boldsymbol{\theta}}(\boldsymbol{x}_0)} [-\log P_{\boldsymbol{\theta}}(\boldsymbol{x}_0)] \leq \mathop{\mathbb{E}}_{P_{\boldsymbol{\theta}}(\boldsymbol{x}_0)} \left[-\log P(\boldsymbol{x}_T) - \sum_{t=1}^{T} \log \frac{P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)}{P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})} \right]$$

$$= \mathop{\mathbb{E}}_{P_{\boldsymbol{\theta}}(\boldsymbol{x}_0)} \left[\text{const} + \sum_{t>1}^{T} \underbrace{D_{\text{kl}}(P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t))}_{L_{t-1}} - \text{almost const} \right],$$

This expression is better since $P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$ is tractable, i.e.,

$$P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \widetilde{\beta}_t \boldsymbol{I}),$$

$$\widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) \triangleq \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1 - \overline{\alpha}_t} \boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1 - \overline{\alpha}_{t-1})}{1 - \overline{\alpha}_t} \boldsymbol{x}_t$$

We mostly only need to take care of L_{t-1} for $t = 1, \ldots, T$.
The remaining part is how to parameterize

$$P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I})$$

Diffusion Model Inference

Recall that we want to minimize

$$L_{t-1} \triangleq \underset{\boldsymbol{x}_0 \sim P_{\boldsymbol{\theta}}(\boldsymbol{x}_0)}{\mathbb{E}} \left[D_{\mathrm{kl}}(P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \parallel P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)) \right]$$

$$= \underset{\boldsymbol{x}_0 \sim P_{\boldsymbol{\theta}}(\boldsymbol{x}_0)}{\mathbb{E}} \left[\frac{1}{2\sigma^2} \left\| \widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right\|^2 \right] + C$$

Using reparameterization trick on $P(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, (1 - \overline{\alpha}\boldsymbol{I}))$,

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

Plug in that $\boldsymbol{x}_t(\boldsymbol{x}_0, \epsilon)$,

$$L_{t-1} - C = \underset{\boldsymbol{x}_0, \boldsymbol{\epsilon}}{\mathbb{E}} \left[\frac{1}{2\sigma^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1 - \overline{\alpha}_t}} \boldsymbol{\epsilon} \right) - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right\|^2 \right]$$

This clearly suggest a parameterization

$$\boldsymbol{\mu}_{\theta}(\boldsymbol{x}_t, t) \triangleq \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1 - \overline{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right),$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is neural network predicts true noise $\boldsymbol{\epsilon}$ from input $\boldsymbol{x}_t$ and time t.

Diffusion Model Inference

Finally, the loss function would be

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

And the sampling process is to sample recursively $\boldsymbol{x}_{t-1} \sim P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$,

$$\boldsymbol{x}_{t-1}=\frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t-\frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)\right)$$

Take home points

- Expressively powerful as there is almost no assumption about family of $\mathcal{P}_{\mathcal{X}}$ while being **tractable**.
- Error occurs in choosing T, choosing function class of score function $\boldsymbol{s_\theta}$, learning $\boldsymbol{s_\theta}$
- No assumption about input structure (vs VAE): 1d, 2d. . . , image, text, ...

Diffusion Model in Action.

- High resolution image generation.
- Conditional generative model.
- Inverse problem.

Applications: High resolution image generation

Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695. This paper is the core of Stable Diffusion.

Diffusion process on image space is too expensive.

- Find a good latent space, a good encoder $\mathcal{E}$ and decoder $\mathcal{D}$
- Project all data to this latent space $\boldsymbol{z}_i = \mathcal{E}(\boldsymbol{x}_i) \sim \mathcal{P}_{\mathcal{Z}}$
- Run diffusion to sample new latent vector $\boldsymbol{z} \sim \mathcal{P}_{\mathcal{Z}}$
- Decode the latent vector to get new image $\mathcal{D}(\boldsymbol{z})$

Conditional generation setting.

Setting:

- We have a list of paired data $(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)$ where $\boldsymbol{y}_i$ is additional information about $\boldsymbol{x}_i$, such as class label, text describing the image.
- Later, we want to sample new $\boldsymbol{x}$ given particular $\boldsymbol{y}$.

Define a generative model $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_1 \leftarrow \ldots \leftarrow \boldsymbol{x}_T$ as

$$P(\boldsymbol{x}_T \mid \boldsymbol{y}) \triangleq \mathcal{N}(\mathbf{0}, \boldsymbol{I}), P_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T \mid \boldsymbol{y}) \triangleq P(\boldsymbol{x}_T \mid \boldsymbol{y}) \prod_{t=1}^{T} P_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{y}), \tag{6}$$

$$P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{y}) \triangleq \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \boldsymbol{y}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t, \boldsymbol{y})) \tag{7}$$

$$P(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T \mid \boldsymbol{x}_0, \boldsymbol{y}) \triangleq \prod_{t=1}^{T} P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}), \quad P(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}) \triangleq \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta, \boldsymbol{I}) \tag{8}$$

The loss function would be

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

Applications: Image Restoration

Yinhuai Wang et al. "Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model". In: *arXiv preprint arXiv:2212.00490* [2022]

Given

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n},$$

where $\boldsymbol{y}$ is observed signal, $\boldsymbol{A}$ is known linear operator (dow-sampling of an image, sampling matrix in compressed sensing, ...), $\boldsymbol{x}$ is the original signal that we wish to recover, $\boldsymbol{n}$ is nonlinear noise.

Existing approaches are

- Domain knowledge-based regularization

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \sum_{i=1}^{N} \|\boldsymbol{y}_i - \boldsymbol{A}\boldsymbol{x}_i\|^2 + \lambda \mathcal{R}(\boldsymbol{x}_i)$$

- Then deep learning comes in: data distribution-based regularization

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{A}\mathcal{G}(\boldsymbol{w}) - \boldsymbol{y}\|^2 + \lambda \mathcal{R}(\boldsymbol{w}),$$

Solution of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is

$$\widehat{\boldsymbol{x}} = \boldsymbol{A}^{\dagger}\boldsymbol{y} + (\boldsymbol{I} - \boldsymbol{A}^{\dagger}\boldsymbol{A})\overline{\boldsymbol{x}}, \quad \forall \overline{\boldsymbol{x}}$$

So the idea is to find $\overline{\boldsymbol{x}}$ such that $P(\widehat{\boldsymbol{x}}; \overline{\boldsymbol{x}}) = \mathcal{P}_{\mathcal{X}}$.
In order to do so, and note that the requirement of *data consistency* is only required on $\boldsymbol{x}_0$, not all the other $\boldsymbol{x}_i$'s (during the sampling process).

- Sample $\boldsymbol{x}_T$
- Sample $\boldsymbol{x}_{t-1}$ based on $\boldsymbol{x}_t$
- Infer $\boldsymbol{x}_0$ from $\boldsymbol{x}_t$
- Rectify $\boldsymbol{x}_0$ to get $\widehat{\boldsymbol{x}}_0$ such that $\boldsymbol{A}\widehat{\boldsymbol{x}_0} = \boldsymbol{y}$ (so it satisfies data consistency)
- Get the "rectify" version of $\boldsymbol{x}_{t-1}$, namely $\widehat{\boldsymbol{x}}_{t-1}$
- Back to step 2

In details

Recall that

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

Suppose we have $\boldsymbol{x}_t$, a good trained diffusion model gives us $\boldsymbol{\epsilon}_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{x}_t, t) \approx \boldsymbol{\epsilon}$, then an estimate of $\boldsymbol{x}_0$ given $\boldsymbol{x}_t$ is

$$\widehat{\boldsymbol{x}}_{0|t} = \frac{1}{\sqrt{\overline{\alpha}_t}} \left(\boldsymbol{x}_t - \boldsymbol{\epsilon}_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{x}_t, t)\sqrt{1 - \overline{\alpha}_t}\right)$$

Modify this to satisfy data consistency,

$$\widetilde{\boldsymbol{x}}_{0|t} = \boldsymbol{A}^\dagger \boldsymbol{A}\boldsymbol{y} + (\boldsymbol{I} - \boldsymbol{A}^\dagger \boldsymbol{A})\widehat{\boldsymbol{x}}_{0|t}$$

Then we can sample $\boldsymbol{x}_{t-1}$ as $\boldsymbol{x}_{t-1} \sim P(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \widetilde{\boldsymbol{x}}_{0|t})$,

$$\boldsymbol{x}_{t-1} = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1 - \overline{\alpha}_t}\widetilde{\boldsymbol{x}}_{0|t} + \frac{\sqrt{\alpha_t}(1 - \overline{\alpha}_{t-1})}{1 - \alpha_t}\boldsymbol{x}_t + \sigma_t\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

Some thoughts

- We only need to train diffusion model once, and then freeze it.
- But we need to know linear operator $\boldsymbol{A}$ in advance

Something else

- Nicholas Carlini et al. "Extracting training data from diffusion models". In: *arXiv preprint arXiv:2301.13188* [2023]
- And measure performance of generative model is still a controversial topic
- Energy-based models.
- Discrete latent.