SDE-based Generative Models

Yuanzhi Zhu
SP at CVL

01/Aug/2022

Content

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

Noise Conditional Score Network (NCSN)*

Motivation: Learning the score function $s_\theta(x) \sim \nabla_x \log p(x)$ instead

Training Objective: Score Matching for Score Estimation

$$\frac{1}{2}\mathbb{E}_{p_{data}}\left[|| \, s_\theta(x) - \nabla_x \log p_{data}(x) \, ||_2^2\right] \longrightarrow \mathbb{E}_{p_{data}}\left[\underbrace{\mathrm{tr}(\nabla_x s_\theta(x))}_{\text{expensive}} + \frac{1}{2} \, || \, s_\theta(x) \, ||_2^2\right]$$

Sampling with Langevin Dynamics

$$x_t = x_{t-1} + \frac{\epsilon}{2}\underbrace{\nabla_x \log p(x_{t-1})}_{\text{score}} + \sqrt{\epsilon} z_t, \quad \text{where } z_t \sim \mathcal{N}(0, \mathbf{I})$$

*[1907.05600v3] Generative Modeling by Estimating Gradients of the Data Distribution (arxiv.org)

Noise Conditional Score Network (NCSN)

Cheaper Score Matching

pre-specified noise distribution, i.e. Gaussian

1. **Denoising Score Matching (DSM)**

   Matching perturbed data distribution $q_\sigma(\tilde{x}) = \int q_\sigma(\tilde{x}|x) p_{data}(x) dx$

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x}|x)p_{data}(x)}\left[|| \; s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) \; ||_2^2\right]$$

2. **Sliced Score Matching:** random projections to approximate $\mathrm{tr}(\nabla_x s_\theta(x))$

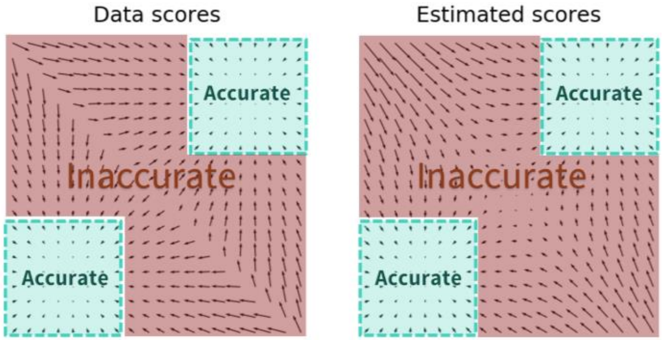
$$\mathbb{E}_{p_v}\mathbb{E}_{p_{data}}\left[v^T s_\theta(x) v - \frac{1}{2} \; || \; s_\theta(x) \; ||_2^2\right]$$

Noise Conditional Score Network (NCSN)

Low Density Regions Pitfalls: learning the score of p_{data} only

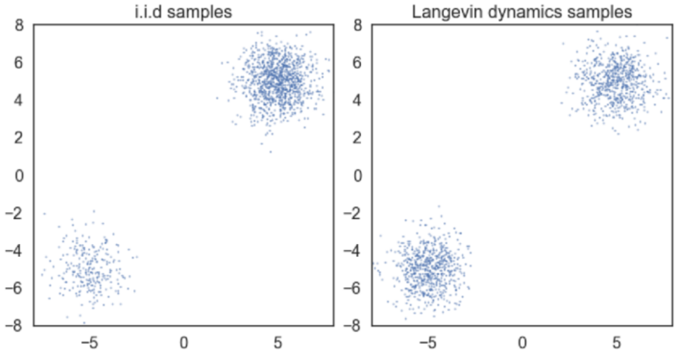
1. Inaccurate score estimation in low data density regions

   For regions with $p_{data} \approx 0$, we do not have sufficient data samples for accurate estimation.



2. Slow mixing of Langevin dynamic

$$p_{data}(x) = \pi p_1(x) + (1-\pi) p_2(x)$$
$$\nabla_x \log p_{data}(x) = \nabla_x \log p_1(x)$$
$$\nabla_x \log p_{data}(x) = \nabla_x \log p_2(x)$$



Noise Conditional Score Network (NCSN)

Training Objective: Score Matching a *Sequence of Noise-levels*

Large noise: perturbate the data sufficiently to better estimate the low density regions
Small noise: be able to converge to the true data distribution

Denoising score matching objective for given σ

$$q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x} \mid x, \sigma^2\mathbf{I}) \longrightarrow \nabla_x \log q_\sigma(\tilde{x}|x) = -(\tilde{x} - x)/\sigma^2$$

$$\ell(\theta;\sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{data}(x)}\mathbb{E}_{\tilde{x}\sim\mathcal{N}(x,\sigma^2\mathbf{I})} \, || \, s_\theta(\tilde{x},\sigma) + \frac{\tilde{x} - x}{\sigma^2} \, ||_2^2$$

Final objective

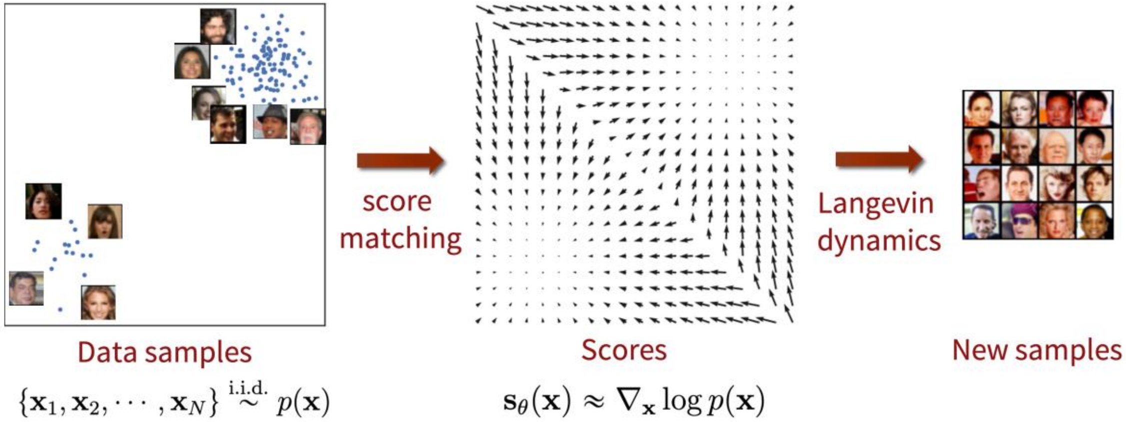
$$\mathcal{L}(\theta;\{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L}\sum_{i=1}^{L} \underbrace{\lambda(\sigma_i)}_{\text{coefficient function}} \ell(\theta;\sigma_i)$$

Noise Conditional Score Network (NCSN)

NCSN Inference: via Annealed Langevin Dynamics

A sequence of positive noise scales $\sigma_{min} = \sigma_1 < \sigma_2 < \cdots < \sigma_L = \sigma_{max}$

$p_{\sigma_{min}}(x) \approx p_{data}(x)$ $\qquad$ $p_{\sigma_{max}}(x) \approx \mathcal{N}(x; 0, \sigma_{max}^2 \mathbf{I})$



Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow L$ to 1 **do**
    $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\qquad$ ▷ α_i is the step size.
    for $t \leftarrow 1$ to T **do**
        Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
        $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
    end for
    $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

Outer loop: responsible for transitioning to next noise levels

Inner loop: takes T steps to guarantee the samples are from p_{σ_i}

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

Deep Unsupervised Learning using Nonequilibrium Thermodynamics*

First Attempt from *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*

Main Ideas:

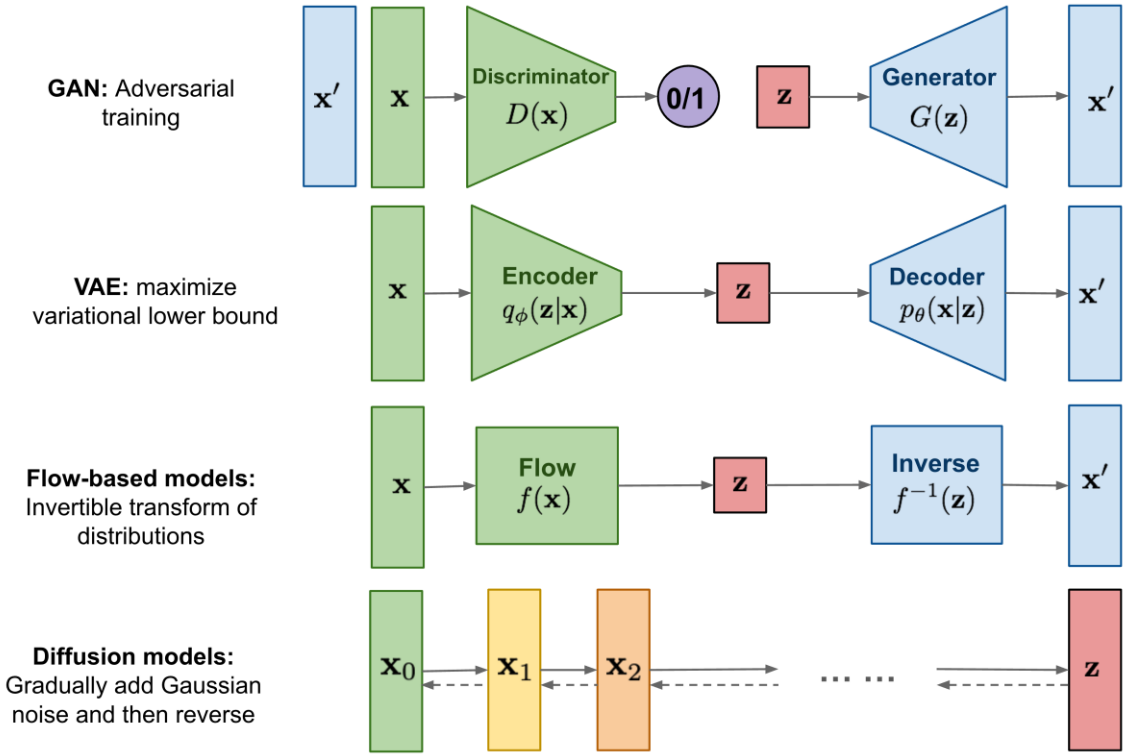
- inspired by non-equilibrium statistical physics
- systematically and slowly destroy structure in a data distribution (iterative forward diffusion)
- then **learn a reverse diffusion process** that restores structure in data(restore data distribution)

Math come soon in DDPM

Jascha Sohl-Dickstein is also author of *RealNVP* and *Score-based generative model through SDE*, now is working on ML theory and NLP

*[1503.03585] Deep Unsupervised Learning using Nonequilibrium Thermodynamics (arxiv.org)

DDPM: Denoising Diffusion Probabilistic Models*



*[2006.11239] Denoising Diffusion Probabilistic Models (arxiv.org)

DDPM: Denoising Diffusion Probabilistic Models

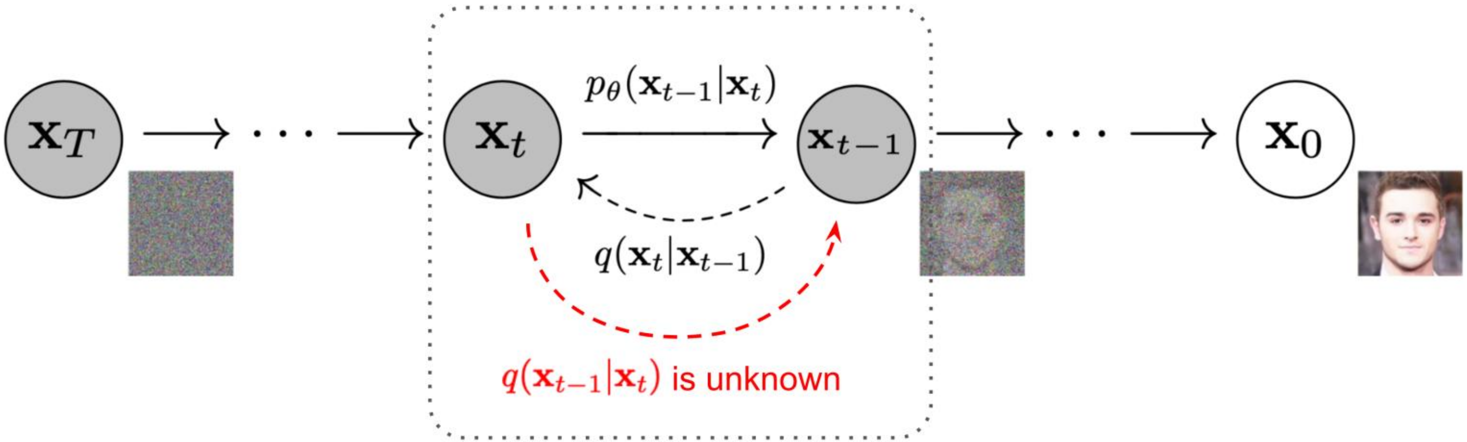
True data dist. : $x_0 \sim q(x_0)$

Forward process: $q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$

Reverse process: $p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$

Markov Assumption

Use variational lower bound



DDPM: Denoising Diffusion Probabilistic Models

Forward Diffusion Process

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$$

Each Step

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \underline{\sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}}) \quad \text{or} \quad x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}z_{t-1}$$

norm invariant

variance schedule β_t controls the diffusion processing

For arbitrary *t*

$$q(x_t|x_0) \quad = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$\alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$$

$$x_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot z_t$$

DDPM: Denoising Diffusion Probabilistic Models

Reverse Diffusion Process

if β_t is small enough, $q(x_{t-1}|x_t)$ will also be Gaussian

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t))$$

Reverse when condition on x_0

$$\begin{aligned}
& q(x_{t-1}|x_t, x_0) \\
&= q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\
&\propto \exp\Big(-\frac{1}{2}\big(\frac{(x_t-\sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1}-\sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(x_t-\sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\
&= \exp\Big(-\frac{1}{2}\big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})x_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0)x_{t-1} + C(x_t, x_0)\big)\Big) \\
&\longrightarrow q(x_{t-1}|x_t, x_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})
\end{aligned}$$

$$\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\Big)$$

$$\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

z_t is the noise between x_t and x_0

DDPM: Denoising Diffusion Probabilistic Models

Negative Log Likelihood to Variational Lower Bound

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)} \Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0) \Big] \\
&= \mathbb{E}_q \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \Big] \\
\text{Let } L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)
\end{aligned}
$$

DDPM: Denoising Diffusion Probabilistic Models

Parameterization for Training Loss

$$L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] = \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big]$$

$$= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big]$$

$$= \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_t} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]$$

Known

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t))$$

$$\begin{cases} \boldsymbol{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} z_\theta(x_t, t)) \\ \boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2 \mathbf{I} \end{cases}$$

Model The Noise(Residual)

$$L_t^{\text{simple}} = \mathbb{E}_{x_0, z_t}\Big[\|z_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t, t)\|^2\Big]$$

DDPM: Denoising Diffusion Probabilistic Models

Nonetheless, it is just another parameterization of $p_\theta(x_{t-1}|x_t)$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t))$$

$$\begin{cases} \boldsymbol{\mu}_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} z_\theta(x_t, t)\right) \\ \boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2 \mathbf{I} \end{cases}$$

two options† $\sigma^2 = \begin{cases} \beta_t \\ \frac{1-\hat{\alpha}_{t-1}}{1-\hat{\alpha}_t}\beta_t \end{cases}$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \mathbf{z}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

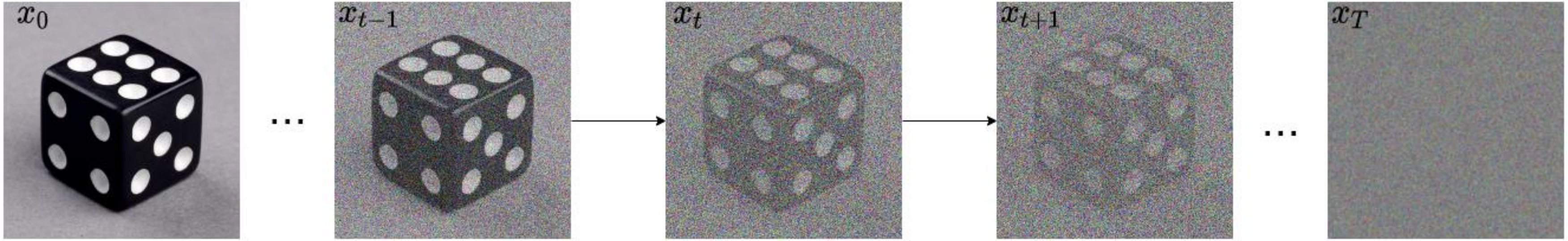
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

† Covariance has analytical optimal form (Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models)

DDPM: Denoising Diffusion Probabilistic Models

$x_0 \sim q(x_0)$ ⟹ $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ ⟹ $q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$



$p_\theta(x_0) = \int p(x_T) \prod_{i=1}^{T} p_\theta(x_{t-1}|x_t) dx_{1:T}$ ⟸ $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ ⟸ $x_T \sim \mathcal{N}(0, I)$

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

DDIM: Denoising Diffusion Implicit Models*

Variational Inference for *Non-Markovian* Forward Processes

$$-\log p_\theta(\mathbf{x}_0) \leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0))$$

Model this directly

$$q_\sigma(x_{1:T}|x_0) = q_\sigma(x_T|x_0) \prod_{t=2}^{T} q_\sigma(x_{t-1}|x_t, x_0)$$

Reverse Process: deterministic given x_t, x_0

$$x_{t-1} = \sqrt{\alpha_{t-1}} x_0 + \sqrt{1-\alpha_{t-1}} \cdot \epsilon_{t-1} = \sqrt{\alpha_{t-1}} x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_t + \sigma_t \epsilon$$

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}} x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{x_t - \sqrt{\alpha_t} x_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 \mathbf{I}\right)$$

Forward: still Gaussian (non-Markovian)

$$q_\sigma(x_t|x_{t-1}, x_0) = \frac{q_\sigma(x_{t-1}|x_t, x_0) q_\sigma(x_t|x_0)}{q_\sigma(x_{t-1}|x_0)}$$

[2010.02502] Denoising Diffusion Implicit Models (arxiv.org)

α_t in DDIM is $\bar{\alpha}_t$ in DDPM.

DDIM: Denoising Diffusion Implicit Models

Model difference between x_0 and x_t

Given: noisy observation x_t

Prediction of the corresponding $\hat{x}_0(x_t) = f_\theta^{(t)}(x_t) = (x_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(x_t))/\sqrt{\alpha_t}$

$$p_\theta^{(t)}(x_{t-1}|x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_1), \sigma_1^2 I) & \text{if } t = 1 \\ q_\sigma(x_{t-1}|x_t, f_\theta^{(t)}(x_t)) & \text{otherwise} \end{cases}$$

Variational Inference Objective (equivalent to objective in DDPM for certain weights)

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

$$= \mathbb{E}_{\boldsymbol{x}_{0:T} \sim q_\sigma(\boldsymbol{x}_{0:T})}\left[\log q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0) + \sum_{t=2}^{T} \log q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) - \sum_{t=1}^{T} \log p_\theta^{(t)}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) - \log p_\theta(\boldsymbol{x}_T)\right]$$

Surrogate Objective $L_t^{\text{simple}} = \mathbb{E}_{x_0, z_t}\left[\|z_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t, t)\|^2\right]$ Same as in DDPM!

DDIM: Denoising Diffusion Implicit Models

Sampling from Generalized Generative Processes $p_\theta^{(t)}(x_{t-1}|x_t)$

$$x_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{predicted } x_0} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t)}_{\text{direction pointing to } x_t} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

$$\sigma_t := \eta \sqrt{(1-\alpha_{t-1})/(1-\alpha_t)} \sqrt{1-\alpha_t/\alpha_{t-1}}$$

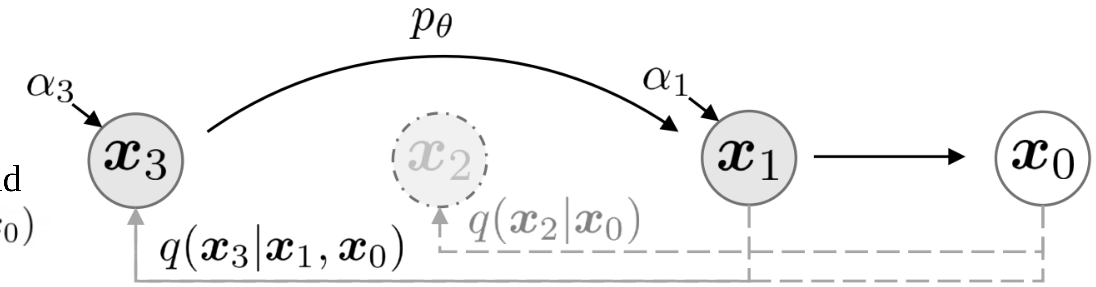
- **DDPM:** $\eta = 1$ (forward process becomes Markovian (different noise schedule from vanilla DDPM))
- **DDIM:** $\eta = 0$ (forward process becomes deterministic)

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**

DDIM: Denoising Diffusion Implicit Models

Accelerated Generation Processes



Denoising surrogate objective does not depend on the specific forward procedure $q_\sigma(x_{t-1}|x_0)$

Consider the forward process as defined on a subset $\tau = [\tau_1, \tau_2, \ldots, \tau_{\dim(\tau)}] \subset [1, 2, \cdots, T]$

$$q_{\sigma,\tau}(x_{\tau_{i-1}}|x_{\tau_t}, x_0) = \mathcal{N}(x_{\tau_{i-1}}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\frac{x_{\tau_i} - \sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})$$

The generative process now samples latent variables according to reversed(τ), which we term (sampling) *trajectory*

→ Train a model with arbitrary number forward steps but only sample from some of them in the generative process

DDIM: Denoising Diffusion Implicit Models

Relevance to Neural ODEs

$$x_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{"predicted } x_0 \text{"}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t)}_{\text{"direction pointing to } x_t \text{"}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

$$\Longrightarrow \frac{x_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} - \frac{x_t}{\sqrt{\alpha_t}} = \left(\sqrt{\frac{1-\alpha_{t-\Delta t}}{\alpha_{t-\Delta t}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}} \right) \epsilon_\theta^{(t)}(x_t)$$

$$\sigma := \sqrt{\frac{1-\alpha}{\alpha}} \qquad \bar{x} := \frac{x}{\sqrt{\alpha}}$$

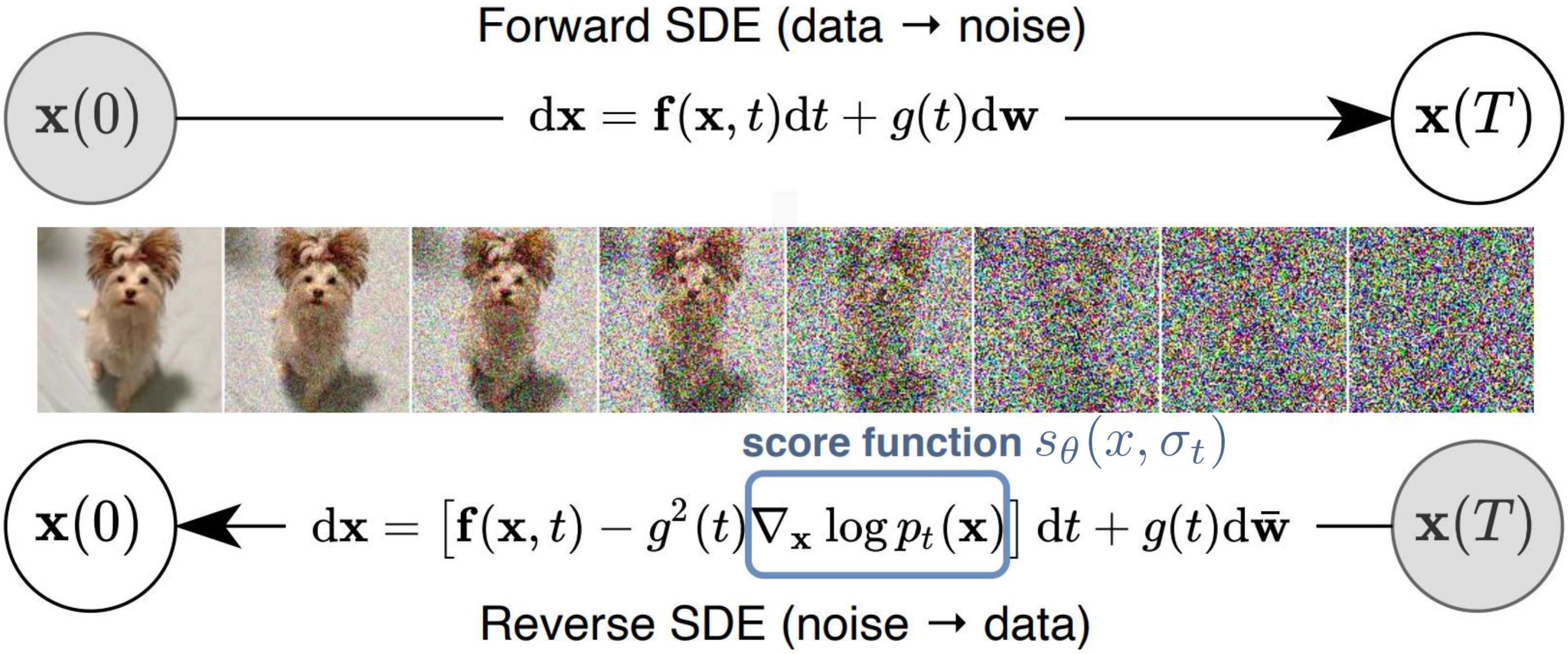
$$\Longrightarrow \mathrm{d}\bar{x}(t) = \epsilon_\theta^{(t)} \left(\frac{\bar{x}(t)}{\sqrt{\sigma^2+1}} \right) \mathrm{d}\sigma(t)$$

Variance-Exploding SDE soon

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

SDE-based Generative Models: A Unified Framework*



*[2011.13456] Score-Based Generative Modeling through Stochastic Differential Equations (arxiv.org)

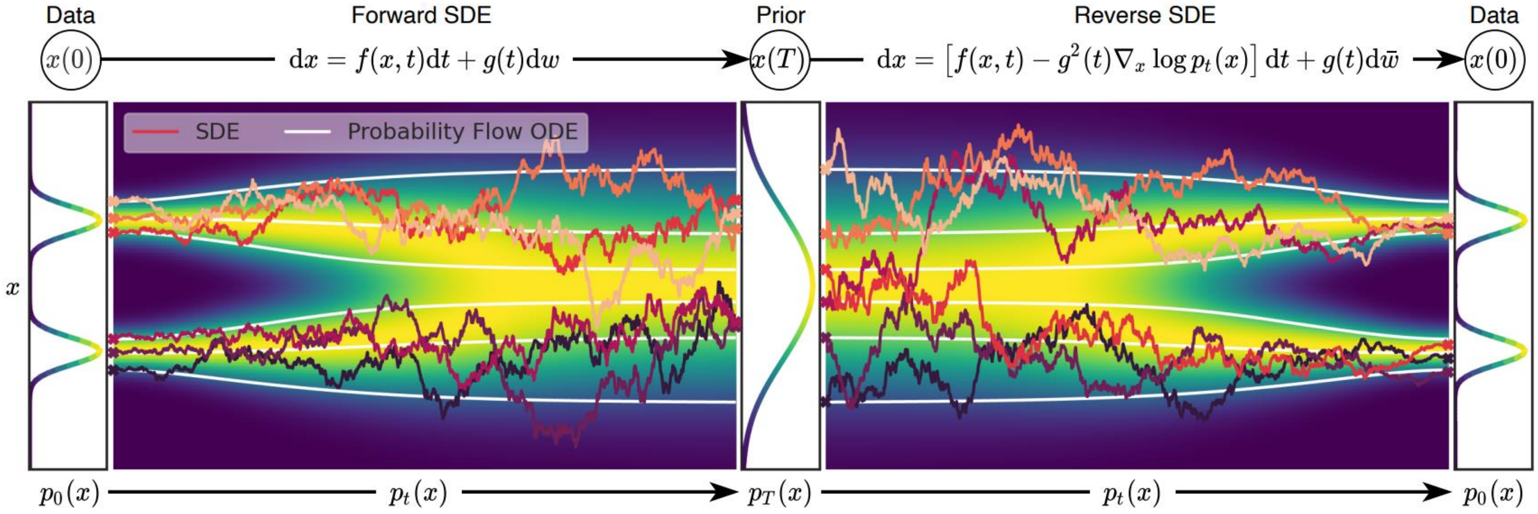
SDE-based Generative Models: A Unified Framework

Model diffusion process as solution of Itô SDE (continuous time)

$$dx = f(x,t)dt + g(t)dw$$

Generating samples by reversing the SDE

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



SDE-based Generative Models: A Unified Framework

Training Objective (DSM)

known Gaussian when $f(x,t)$ if affine

$$\theta^* = arg\min \mathbb{E}_{t\sim U(0,T)} \{\lambda(t)\mathbb{E}_{x(0)}\mathbb{E}_{x(t)|x(0)}[||s_\theta(x(t),t) - \nabla_{x(t)} \log p_{0t}(x(t)|x(0))||_2^2]\}$$

Discretizations

$$dx = f(x,t)dt + g(t)dw$$

SDE Form	Discrete Markov Chain	SDE Expression
Variance Exploding (VE) SDE (SMLD)	$x_i = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} z_{i-1}$	$\mathrm{d}x = \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\mathrm{d}w$
Variance Preserving (VP) SDE (DDPM)	$x_i = \sqrt{1-\beta_i} x_{i-1} + \sqrt{\beta_i} z_{i-1}$	$\mathrm{d}x = \frac{1}{2}\beta(t)x\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}w$

SDE-based Generative Models: A Unified Framework

Reverse SDE Discretization $x_i = x_{i+1} - f_{i+1}(x_{i+1}) + g_{i+1}g_{i+1}^T s_\theta(x_{i+1}, i+1) + g_{i+1}z_{i+1}$

Predictor-Corrector (PC) Samplers

1. **Predictor:** general-purpose numerical SDE solvers
2. **Corrector:** score-based MCMC (i.e. Langevin MCMC)

Ensure samples on the desired manifold

DDPM: predictor only
SMLD: corrector only

Algorithm 2 PC sampling (VE SDE)

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

Algorithm 3 PC sampling (VP SDE)

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i = N-1$ **to** 0 **do**
  $\mathbf{x}'_i \leftarrow (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\theta^*}(\mathbf{x}_{i+1}, i+1)$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}}\,\mathbf{z}$
  for $j = 1$ **to** M **do**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta^*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\,\mathbf{z}$
return $\mathbf{x}_0$

Predictor

Corrector

SDE-based Generative Models: A Unified Framework

Relationship between Bayesian Posterior and Reverse SDE

$$\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t = \boldsymbol{f}_t(\boldsymbol{x}_t)\Delta t + g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

$$\begin{aligned} p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t) &= \mathcal{N}\left(\boldsymbol{x}_{t+\Delta t}; \boldsymbol{x}_t + \boldsymbol{f}_t(\boldsymbol{x}_t)\Delta t, g_t^2\Delta t\, \boldsymbol{I}\right) \\ &\propto \exp\left(-\frac{\|\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t)\Delta t\|^2}{2g_t^2\Delta t}\right) \end{aligned}$$

$$\begin{aligned} p(\boldsymbol{x}_t|\boldsymbol{x}_{t+\Delta t}) &= \boxed{\frac{p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)}{p(\boldsymbol{x}_{t+\Delta t})}} = p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t)\exp(\log p(\boldsymbol{x}_t) - \log p(\boldsymbol{x}_{t+\Delta t})) \\ &\propto \exp\left(-\frac{\|\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t)\Delta t\|^2}{2g_t^2\Delta t} + \log p(\boldsymbol{x}_t) - \log p(\boldsymbol{x}_{t+\Delta t})\right) \end{aligned}$$

$$\log p(\boldsymbol{x}_{t+\Delta t}) \approx \log p(\boldsymbol{x}_t) + (\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t)\cdot\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t) + \Delta t\frac{\partial}{\partial t}\log p(\boldsymbol{x}_t)$$

$$\begin{aligned} p(\boldsymbol{x}_t|\boldsymbol{x}_{t+\Delta t}) &\propto \exp\left(-\frac{\|\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t - \left[\boldsymbol{f}_t(\boldsymbol{x}_t) - g_t^2\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)\right]\Delta t\|^2}{2g_t^2\Delta t}\right) \\ &\approx \exp\left(-\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_{t+\Delta t} + \boxed{\left[\boldsymbol{f}_{t+\Delta t}(\boldsymbol{x}_{t+\Delta t}) - g_{t+\Delta t}^2\nabla_{\boldsymbol{x}_{t+\Delta t}}\log p(\boldsymbol{x}_{t+\Delta t})\right]}\Delta t\|^2}{2g_{t+\Delta t}^2\Delta t}\right) \end{aligned}$$

$$d\boldsymbol{x} = \left[\boldsymbol{f}_t(\boldsymbol{x}) - g_t^2\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right]dt + g_t d\boldsymbol{w}$$

SDE-based Generative Models: A Unified Framework

Sampling: DDPM and SDE point of views (equivalent up to first order)

The ancestral sampling of DDPM matches its reverse diffusion counterpart when $\beta_i \approx 0$ for all i

Bayesian Posterior — Reverse SDE

$$\begin{aligned}
\mathbf{x}_i &= \frac{1}{\sqrt{1-\beta_{i+1}}}(\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&= \left(1 + \frac{1}{2}\beta_{i+1} + o(\beta_{i+1})\right)(\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&\approx \left(1 + \frac{1}{2}\beta_{i+1}\right)(\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1)) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&= \left(1 + \frac{1}{2}\beta_{i+1}\right)\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \frac{1}{2}\beta_{i+1}^2\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&\approx \left(1 + \frac{1}{2}\beta_{i+1}\right)\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&= \left[2 - \left(1 - \frac{1}{2}\beta_{i+1}\right)\right]\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&\approx \left[2 - \left(1 - \frac{1}{2}\beta_{i+1}\right) + o(\beta_{i+1})\right]\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1} \\
&= (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}^*}(\mathbf{x}_{i+1}, i+1) + \sqrt{\beta_{i+1}}\mathbf{z}_{i+1}.
\end{aligned}$$

SDE-based Generative Models: A Unified Framework

Model: DDPM and SDE point of views

Score in score-based model is affine transformation of predicted noise in DDPM

$$\boldsymbol{x_t} = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \boldsymbol{\varepsilon}$$ Equivalent one step forward

$$\begin{aligned} \boldsymbol{s}_\theta(\boldsymbol{x}_t, t) &\approx \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) \\ &= -\frac{\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0}{1-\bar{\alpha}_t} \\ &= -\frac{\boldsymbol{\varepsilon}}{\sqrt{1-\bar{\alpha}_t}} \\ &\approx -\frac{\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}} \end{aligned}$$

Denoising score matching

Gaussian assumption

SDE-based Generative Models: A Unified Framework

ODE form

Fokker-Plank function associated with forward diffusion

$$\frac{\partial}{\partial t} p_t(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} \cdot [\boldsymbol{f}_t(\boldsymbol{x}) p_t(\boldsymbol{x})] + \frac{1}{2} g_t^2 \nabla_{\boldsymbol{x}} \cdot \nabla_{\boldsymbol{x}} p_t(\boldsymbol{x})$$

With $\sigma_t^2 \leq g_t^2$ and $\begin{aligned} \boldsymbol{f}_t &\longrightarrow \boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) \\ g_t &\longrightarrow \sigma_t \end{aligned}$

$$d\boldsymbol{x} = \boldsymbol{f}_t(\boldsymbol{x})dt + g_t d\boldsymbol{w} \quad \overset{equivalent}{\longleftrightarrow} \quad d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}(g_t^2 - \sigma_t^2)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right) dt + \sigma_t d\boldsymbol{w}$$

Reverse

$$d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}(g_t^2 + \sigma_t^2)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right) dt + \sigma_t d\boldsymbol{w}$$

ODE

$$d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2} g_t^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right) dt$$

Comparing with SDEs, ODEs can be solved with larger step sizes as they have no randomness.

SDE-based Generative Models: A Unified Framework

Controllable Generation

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

Bayesian

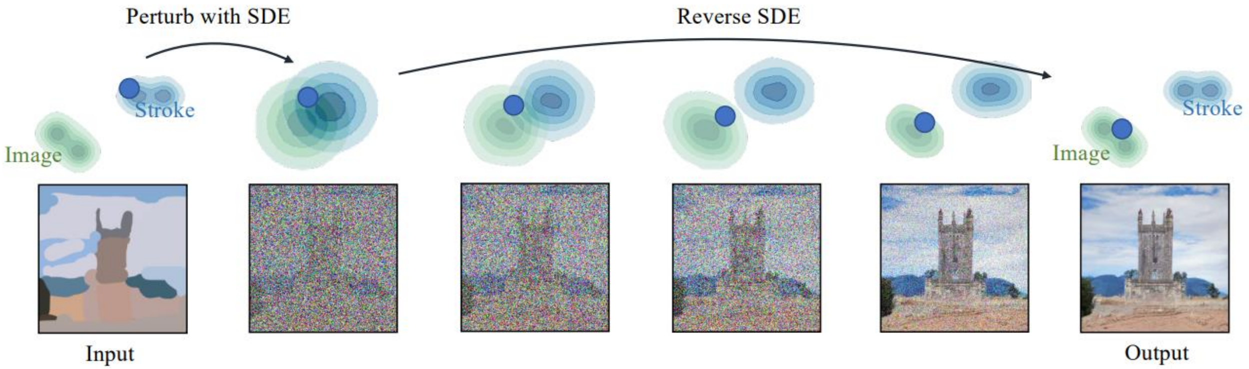
time-dependent classifier

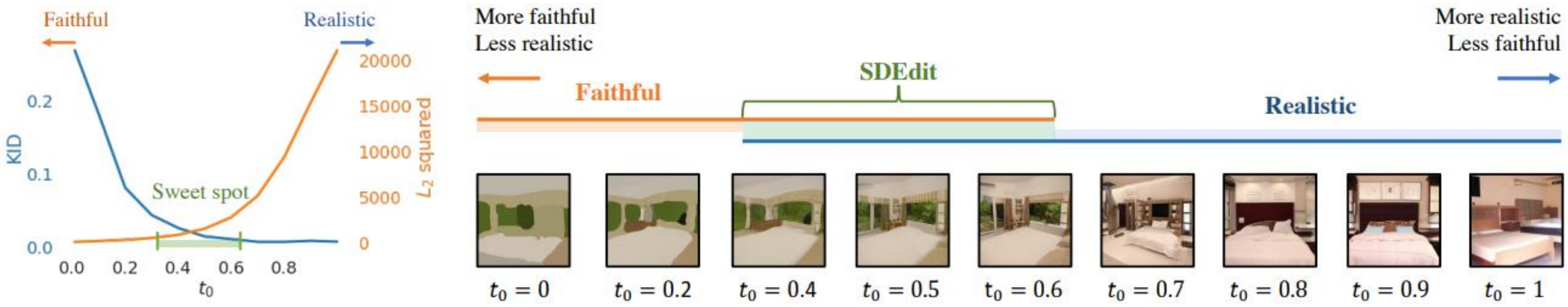
$$dx = \{f(x,t) - g(t)^2 [\nabla_x \log p_t(x) + \nabla_x \log p_t(y|x)]\}dt + g(t)d\bar{w}$$

Content

- NCSN
- DDPM
- DDIM
- SDE-based
- Applications

SDEdit: Guided Image Synthesis and Editing with SDE*

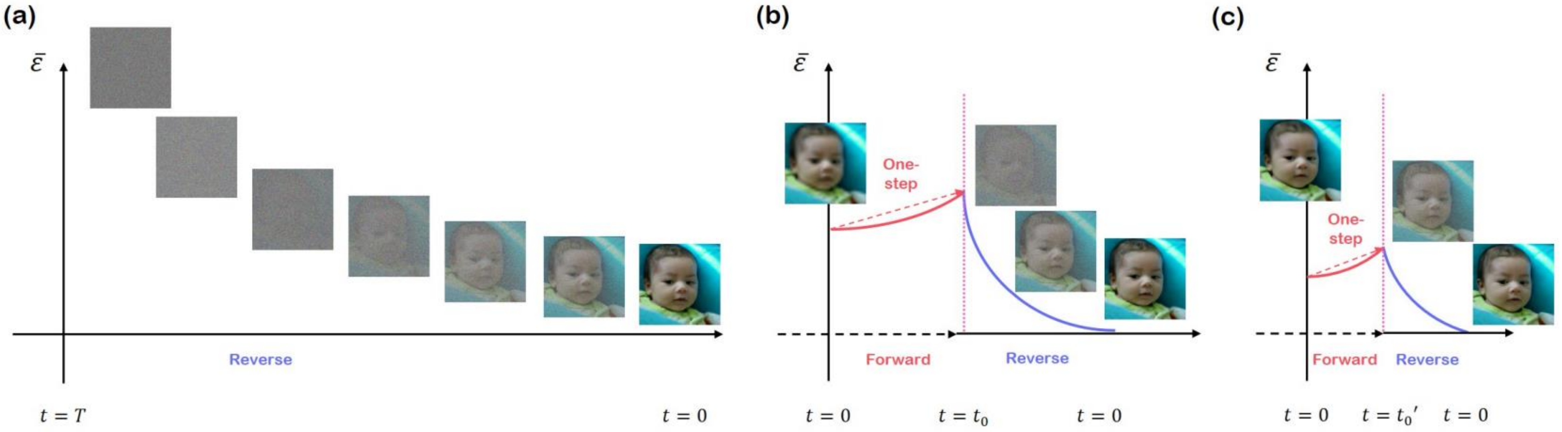




*[2108.01073] SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations (arxiv.org)

Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction*

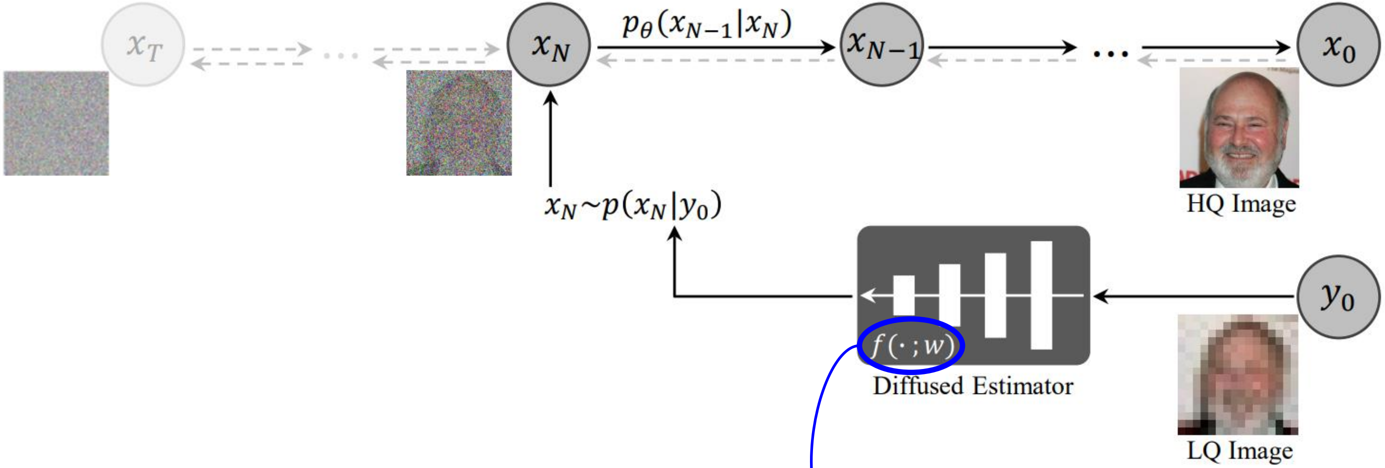
Same idea but different downstream tasks: super-resolution (SR), inpainting, and MRI reconstruction



*[2112.05146] Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction

DifFace: Blind Face Restoration with Diffused Error Contraction*

Same idea but different downstream task: Blind face (easier) restoration



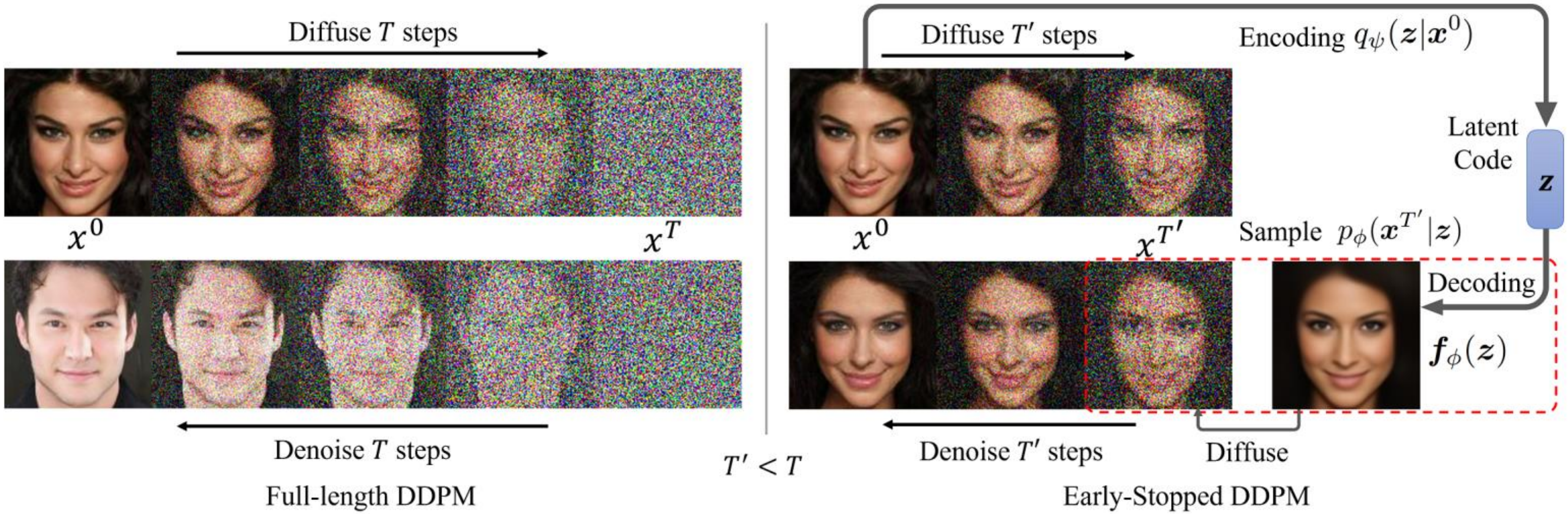
$$\boldsymbol{y} = \left\{ [(\boldsymbol{x} * \boldsymbol{k}_l) \downarrow_s + \boldsymbol{n}_\sigma]_{\text{JPEG}_q} \right\} \uparrow_s$$

two classical network architectures
as the backbone SRCNN and SwinIR

*[2212.06512] DifFace: Blind Face Restoration with Diffused Error Contraction (arxiv.org)

Accelerating Diffusion Models via Early Stop of the Diffusion Process*

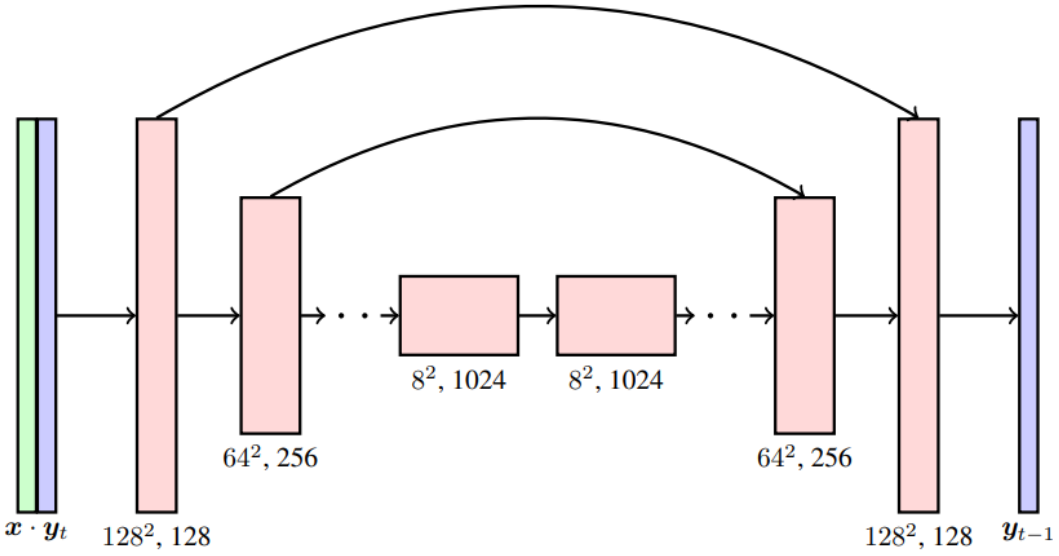
Get not fully noising image by diffusing output from pre-trained models like GAN and VAE



*[2205.12524] Accelerating Diffusion Models via Early Stop of the Diffusion Process (arxiv.org)

Image Super-Resolution via Iterative Refinement*

The condition is concatenated with y_t along the channel dimension (cascaded)

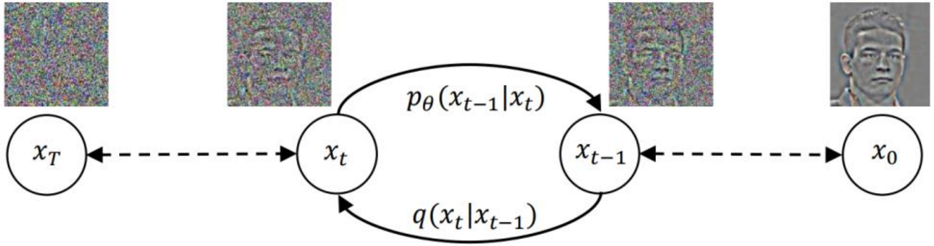


Same author also proposed palette for multi-tasks†, same architecture used for cascaded diffusion‡

*[2104.07636] Image Super-Resolution via Iterative Refinement (arxiv.org)
†[2111.05826] Palette: Image-to-Image Diffusion Models (arxiv.org)
‡[2106.15282] Cascaded Diffusion Models for High Fidelity Image Generation (arxiv.org)

SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models*

Learn the residual with condition encoded LR (fused as 2D CNN block outputs)



Algorithm 1 Training

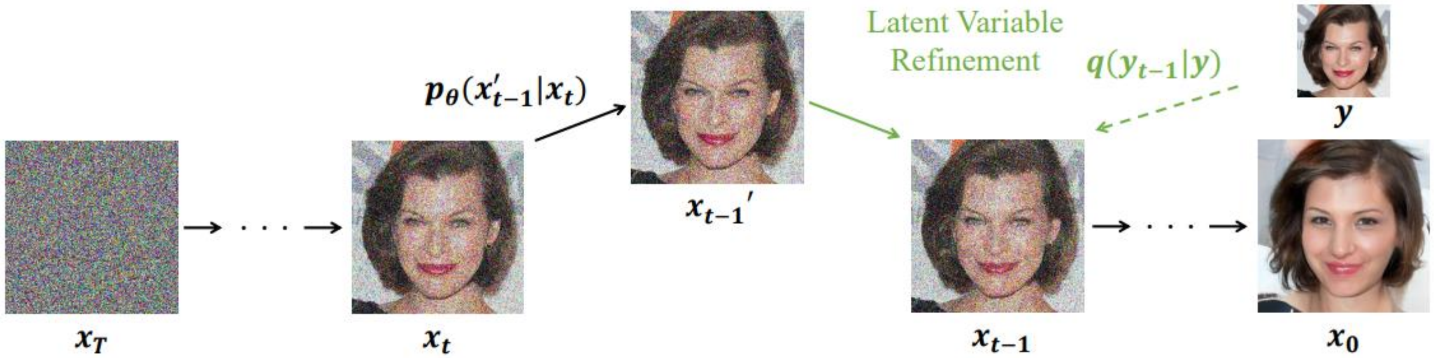
Input: LR image and its corresponding HR image pairs $P = \{(x_L^k,\ x_H^k)\}_{k=1}^K$, total diffusion step T
Initialize: randomly initialized conditional noise predictor ϵ_θ and pretrained LR encoder $\mathcal{D}$
repeat
  Sample $(x_L,\ x_H) \sim P$
  Upsample x_L as $up(x_L)$, compute $x_r = x_H - up(x_L)$
  Encode LR image x_L as $x_e = \mathcal{D}(x_L)$
  Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $t \sim \text{Uniform}(\{1, \cdots, T\})$
  Take gradient step on
  $\nabla_\theta \|\epsilon - \epsilon_\theta(x_t, x_e, t)\|,\ x_t = \sqrt{\bar{\alpha}_t} x_r + \sqrt{1 - \bar{\alpha}_t}\epsilon$
until converged

Algorithm 2 Inference

Input: LR image x_L, total diffusion step T
Load: conditional noise predictor ϵ_θ and LR encoder $\mathcal{D}$
Sample $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
Upsample x_L to $up(x_L)$
Encode LR image x_L as $x_e = \mathcal{D}(x_L)$
for $t = T, T-1, \cdots, 1$ **do**
  Sample $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $z = 0$
  Compute x_{t-1} using Eq. (7):
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, x_e, t)\right) + \sigma_\theta(x_t, t)z$
end for
return $x_0 + up(x_L)$ as SR prediction

[2104.14951] SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models (arxiv.org)

ILVR: Conditioning Method for DDPM*



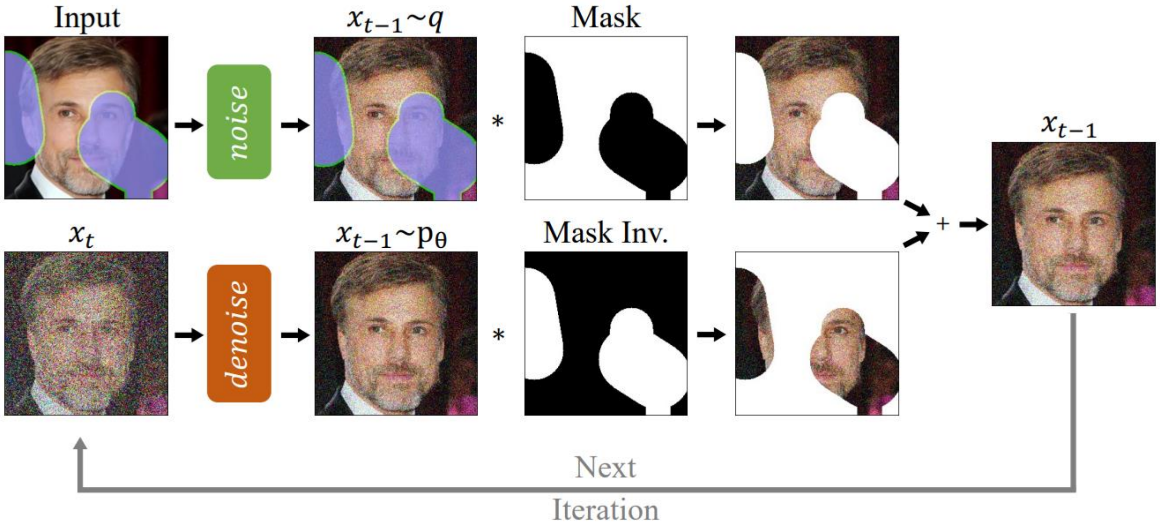
Algorithm 1 Iterative Latent Variable Refinement

Input: Reference image y
Output: Generated image x
$\phi_N(\cdot)$: low-pass filter with scale N
Sample $x_T \sim N(\mathbf{0}, \mathbf{I})$
for $t = T, ..., 1$ **do**
    $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$
    $x'_{t-1} \sim p_\theta(x'_{t-1}|x_t)$ ▷ unconditional proposal
    $y_{t-1} \sim q(y_{t-1}|y)$ ▷ condition encoding
    $x_{t-1} \leftarrow \phi_N(y_{t-1}) + x'_{t-1} - \phi_N(x'_{t-1})$
end for
return x_0

*[2108.02938] ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models (arxiv.org)

RePaint: Inpainting using Denoising Diffusion Probabilistic Models*

Same idea but different downstream tasks from ILVR



Algorithm 1 Inpainting using our RePaint approach.

$x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  for $u = 1, \ldots, U$ **do**
    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\epsilon = \mathbf{0}$
    $x_{t-1}^{\text{known}} = \sqrt{\bar{\alpha}_t} x_0 + (1 - \bar{\alpha}_t)\epsilon$
    $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
    $x_{t-1}^{\text{unknown}} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(x_t, t) \right) + \sigma_t z$
    $x_{t-1} = m \odot x_{t-1}^{\text{known}} + (1 - m) \odot x_{t-1}^{\text{unknown}}$
    if $u < U$ and $t > 1$ **then**
      $x_t \sim \mathcal{N}(\sqrt{1 - \beta_{t-1}} x_{t-1}, \beta_{t-1} \mathbf{I})$
    end if
  end for
end for
return x_0

*[2201.09865] RePaint: Inpainting using Denoising Diffusion Probabilistic Models (arxiv.org)

Noise Estimation for Generative Diffusion Models*

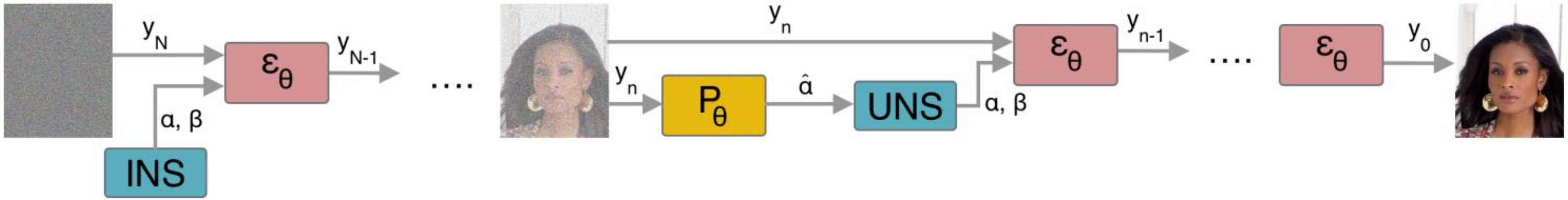


Figure 1: An overview of our generative process. INS and UNS are respectively the functions initializeNoiseSchedule() and updateNoiseSchedule($\hat{\alpha}$).

Algorithm 3: P_θ training procedure

```
repeat
    y_0 ~ q(y_0)
    s ~ U({1, ..., N})
    sqrt(ᾱ) ~ U([l_{s-1}, l_s])
    ε ~ N(0, I)
    y_s = sqrt(ᾱ) y_0 + sqrt(1 - |ᾱ|) ε
    α̂ = P_θ(y_s)
    Take gradient descent step on:
    || log(1 - ᾱ) - log(1 - α̂) ||_2
until converged
```

- **repeat**
  - $y_0 \sim q(y_0)$
  - $s \sim \mathcal{U}(\{1, ..., N\})$
  - $\sqrt{\bar{\alpha}} \sim \mathcal{U}([l_{s-1}, l_s])$
  - $\varepsilon \sim \mathcal{N}(0, I)$
  - $y_s = \sqrt{\bar{\alpha}} y_0 + \sqrt{1 - |\bar{\alpha}|}\varepsilon$
  - $\hat{\alpha} = P_\theta(y_s)$
  - Take gradient descent step on: $\| \log(1 - \bar{\alpha}) - \log(1 - \hat{\alpha}) \|_2$
- **until** converged

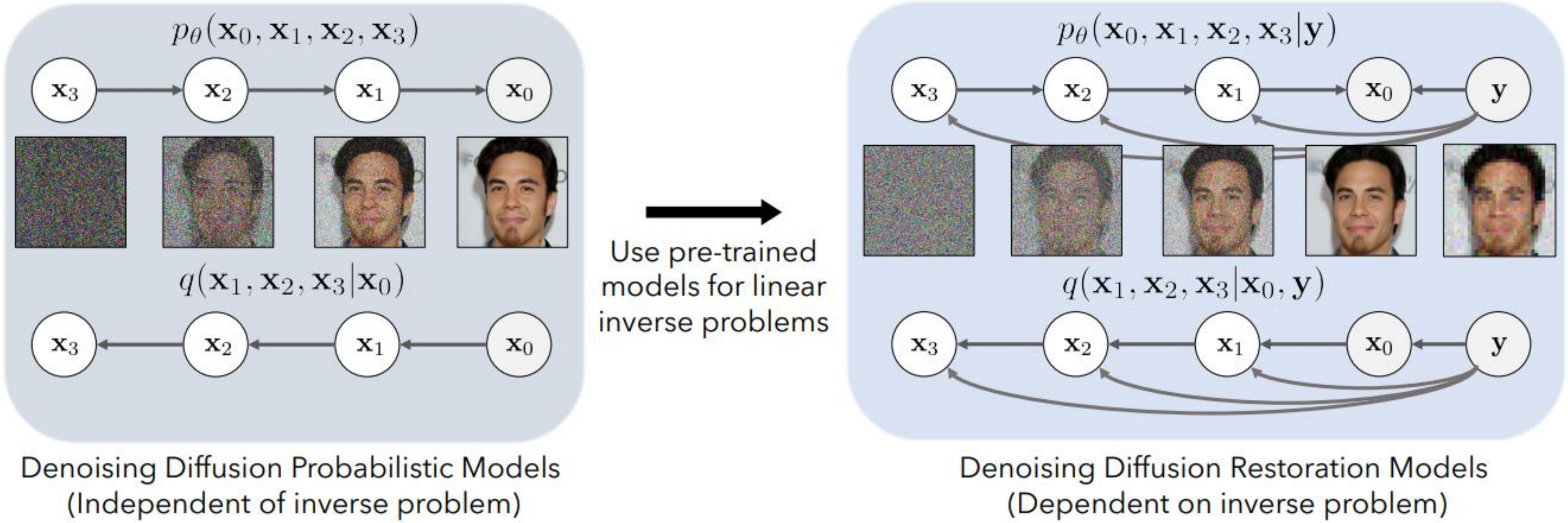
Algorithm 4: Model inference procedure

- N Number of iterations
- $y_N \sim \mathcal{N}(0, I)$
- α, β = initialNoiseSchedule()
- **for** n= N, ..., 1 **do**
  - $z \sim \mathcal{N}(0, I)$
  - $\hat{\varepsilon} = \varepsilon_\theta(y_n, \sqrt{\bar{\alpha}_n})$ or $\varepsilon_\theta(y_n, t)$ where $\bar{\alpha}_n \in [l_t, l_{t-1}]$
  - $y_{n-1} = \dfrac{y_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\hat{\varepsilon}}{\sqrt{\alpha_n}}$
  - **if** $n \in U$ **then**
    - $\hat{\alpha} = P_\theta(y_{n-1})$
    - α, β, τ = updateNoiseSchedule($\hat{\alpha}, n$)
  - **end if**
  - **if** $n \neq 1$ **then**
    - $y_{n-1} = y_{n-1} + \sigma_n z$
  - **end if**
- **end for**
- **return** y_0

*[2104.02600] Noise Estimation for Generative Diffusion Models (arxiv.org)

Denoising Diffusion Restoration Models (DDRM)*

An efficient, unsupervised posterior sampling method



*[2201.11793] Denoising Diffusion Restoration Models (arxiv.org)

Dual Diffusion Implicit Bridges for Image-to-Image Translation*

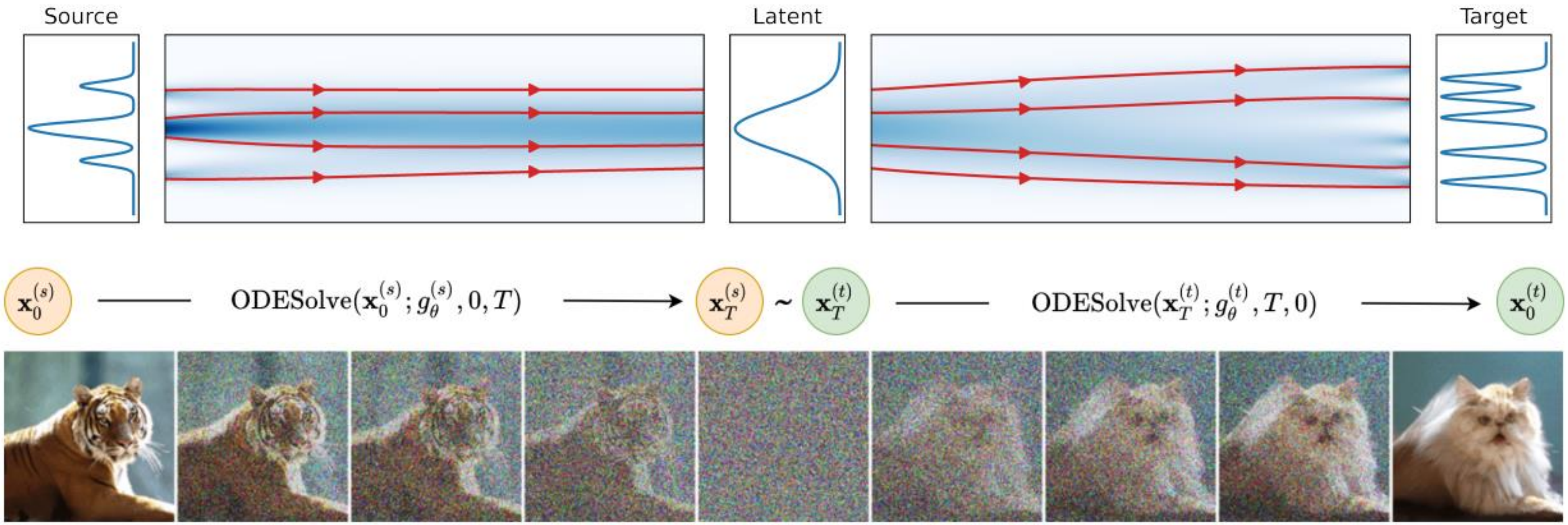
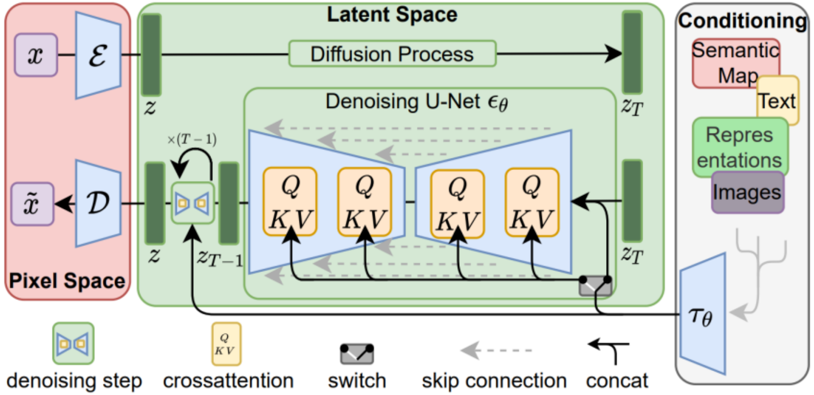


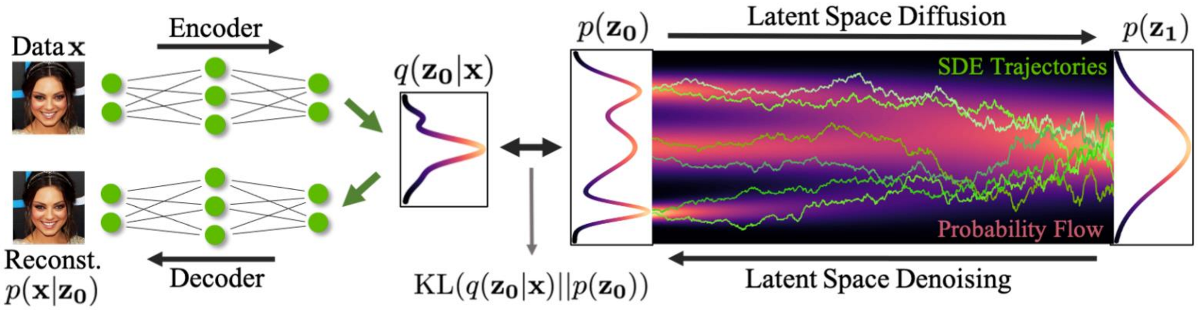
Figure 1: **Dual Diffusion Implicit Bridges**: DDIBs leverage two ODEs for image translation. Given a source image $\mathbf{x}_0^{(s)}$, the source ODE runs in the forward direction to convert it to the latent $\mathbf{x}_T^{(s)}$, while the target, reverse ODE then constructs the target image $\mathbf{x}_0^{(t)}$. (*Top*) Illustration of the DDIB idea between two one-dimensional distributions. (*Bottom*) DDIB from a tiger to a cat using a pre-trained conditional diffusion model.

*[2203.08382] Dual Diffusion Implicit Bridges for Image-to-Image Translation (arxiv.org)

Score-based Generative Modeling in Latent Space*

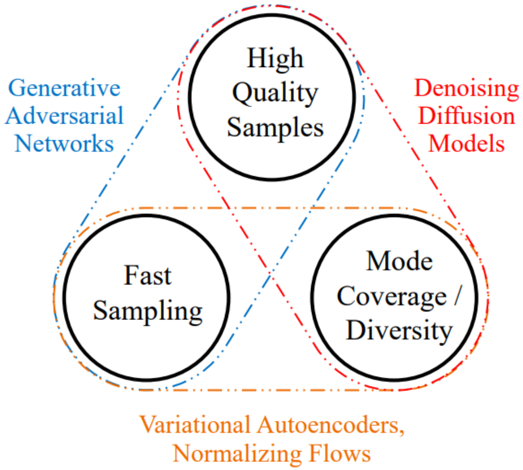
Faster diffusion in latent space

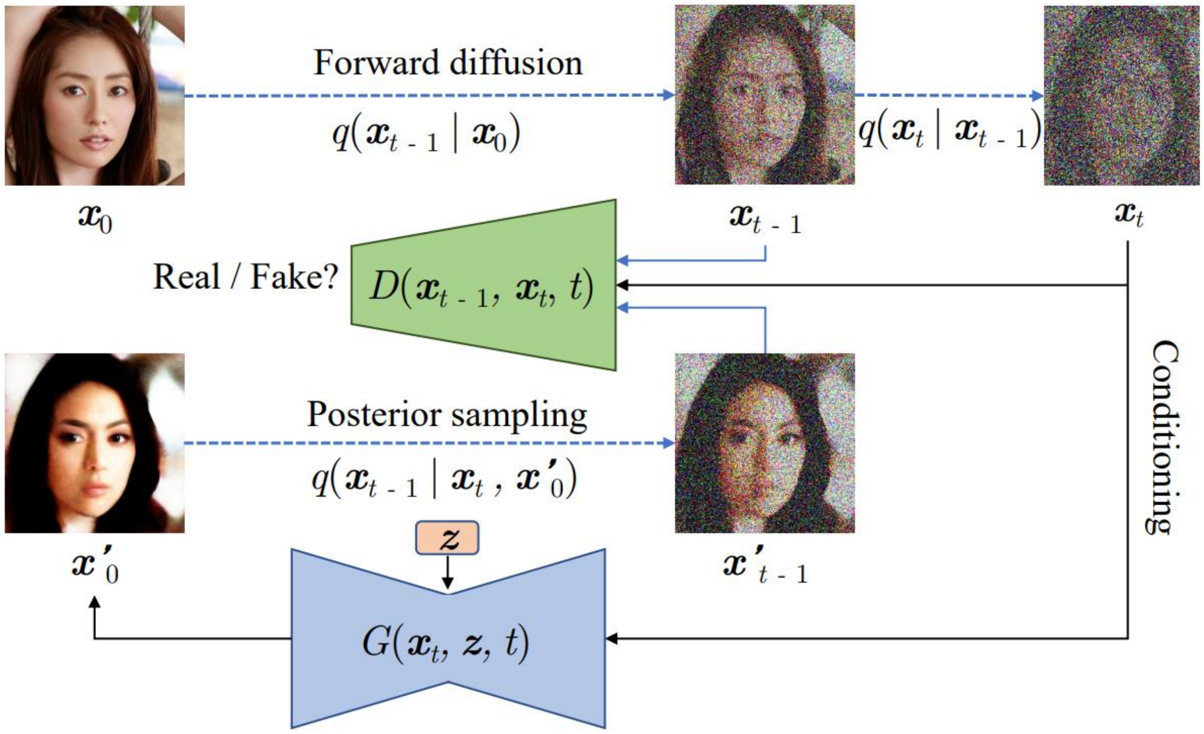
†


*


†[2112.10752] High-Resolution Image Synthesis with Latent Diffusion Models (arxiv.org)
*[2106.05931] Score-based Generative Modeling in Latent Space (arxiv.org)

Tackling the Generative Learning Trilemma with Denoising Diffusion GANS*





*[2112.07804] Tackling the Generative Learning Trilemma with Denoising Diffusion GANs (arxiv.org)

Thank You!

Useful Resources

- What's the score? – Review of latest Score Based Generative Modeling papers
- zhangbaijin/Diffusion-model-low-level (github.com)
- What are Diffusion Models? | Lil'Log (lilianweng.github.io)
- Generative Modeling by Estimating Gradients of the Data Distribution | Yang Song
- Diffusion Models as a kind of VAE
- yang-song/score_sde_pytorch
- Denoising Diffusion Probabilistic Models (DDPM) (labml.ai)
- Denoising Diffusion-based Generative Modeling: Foundations and Applications