



भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

COV877

Special Module on Visual Computing

Generative AI for Visual Content Creation: Image, Video, and 3D

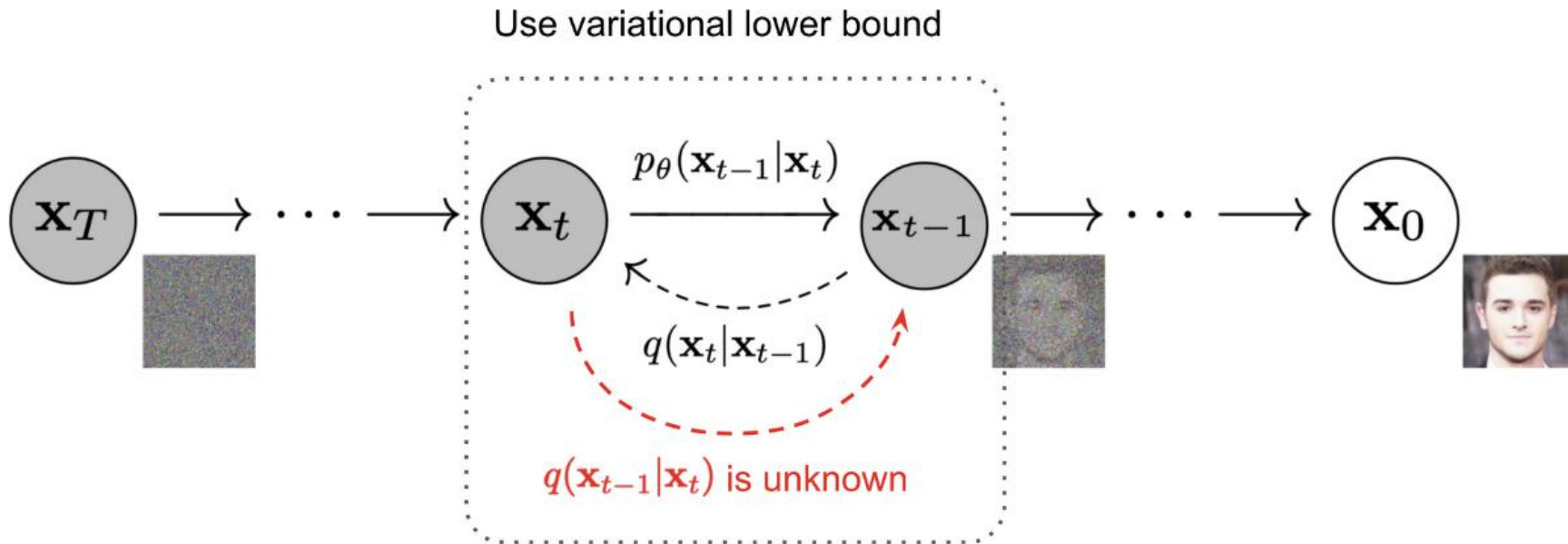
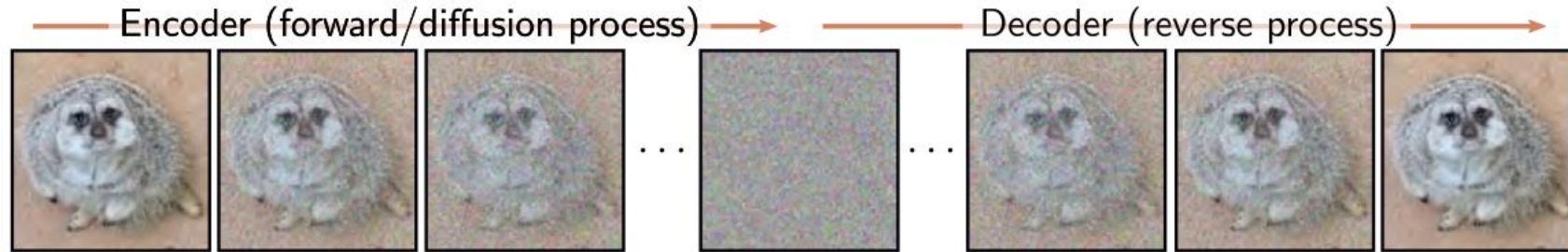
DDPM and Conditional Generation

Instructor:

Dr. Lokender Tiwari

Research Scientist

Denoising Diffusion Probabilistic Model (DDPM)



- [1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.
- [2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>
- [3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.
- [4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Forward Process

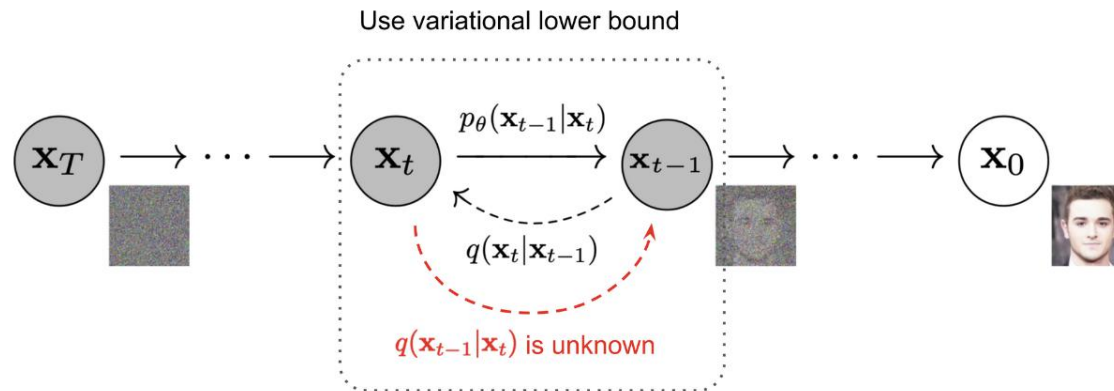
Sample a data point from a real data distribution and add small amount of Gaussian noise to it in T in steps

$$\mathbf{x}_0 \sim q(\mathbf{x})$$

$$\mathbf{x}_1, \dots, \mathbf{x}_T$$

Step size controlled by a variance schedule

$$\{\beta_t \in (0, 1)\}_{t=1}^T$$



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

When $T \rightarrow \infty$, data sample loses all its features, and end up to complete Gaussian distribution

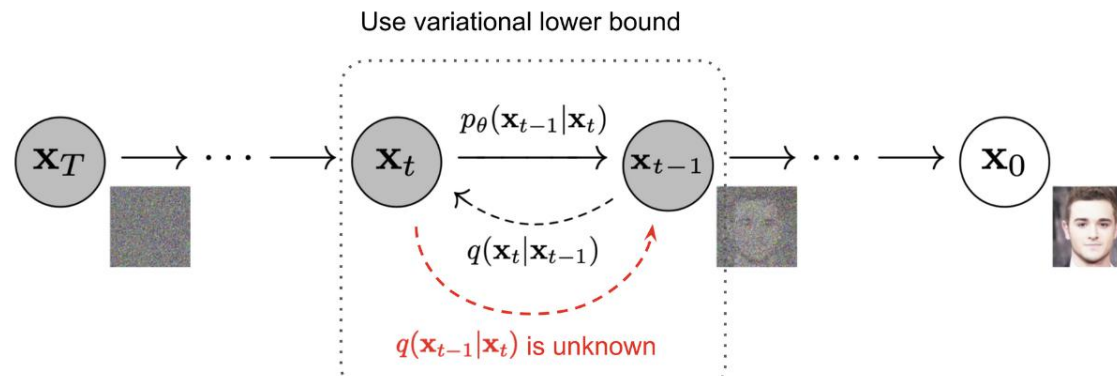
[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Forward Process



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

For get any intermediate sample $\mathbf{x}_t$ do we have to sample all previous $\mathbf{x}_t$?

No, we can sample at any time step t using reparameterization trick

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

$$\epsilon_{t-1}, \epsilon_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1 - \alpha_t}\epsilon_{t-1}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-2}$$

$$= \dots$$

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

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

• merged Gaussians

$$\beta_1 < \beta_2 < \dots < \beta_T$$

Merging Two Gaussians

$$\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$$

$$\mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I})$$

Merged Distribution

$$\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$$

Merged STD

$$\sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t\alpha_{t-1}}$$

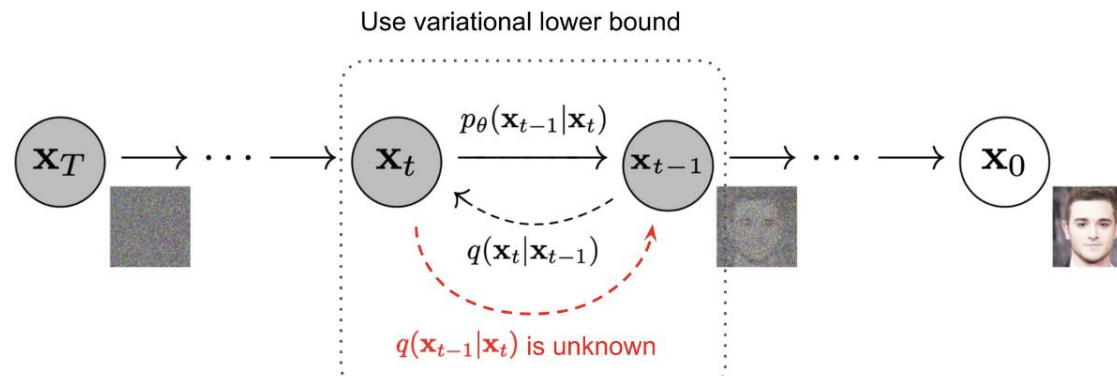
[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Forward Process



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

For get any intermediate sample $\mathbf{x}_t$ do we have to sample all previous $\mathbf{x}_t$?

No, we can sample at any time step t using reparameterization trick

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

$$\epsilon_{t-1}, \epsilon_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1 - \alpha_t}\epsilon_{t-1}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-2}$$

$$= \dots$$

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

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

• merged Gaussians

$$\beta_1 < \beta_2 < \dots < \beta_T$$

Merging Two Gaussians

$$\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$$

$$\mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I})$$

Merged Distribution

$$\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$$

Merged STD

$$\sqrt{(1 - \alpha_t) + \alpha_t(1 - \alpha_{t-1})} = \sqrt{1 - \alpha_t\alpha_{t-1}}$$

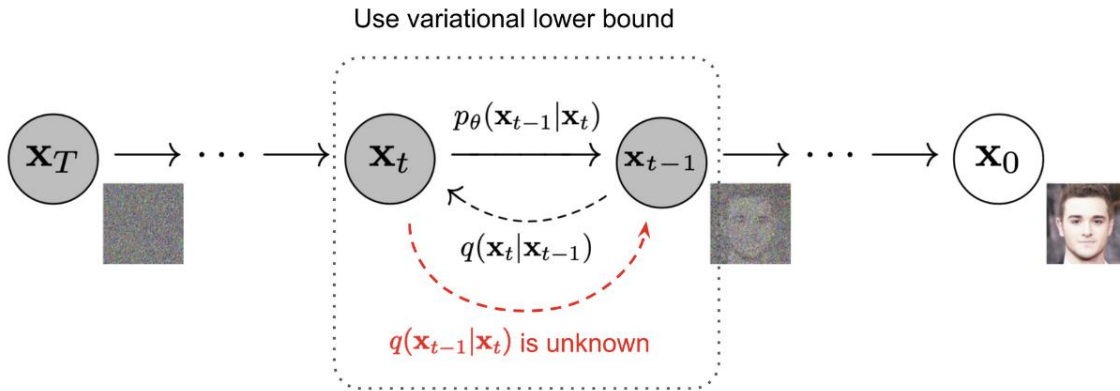
[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Reverse Process $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$



Sample from conditional probability $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

Generate sample from Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

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

Approximate conditional probabilities by learning a model

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

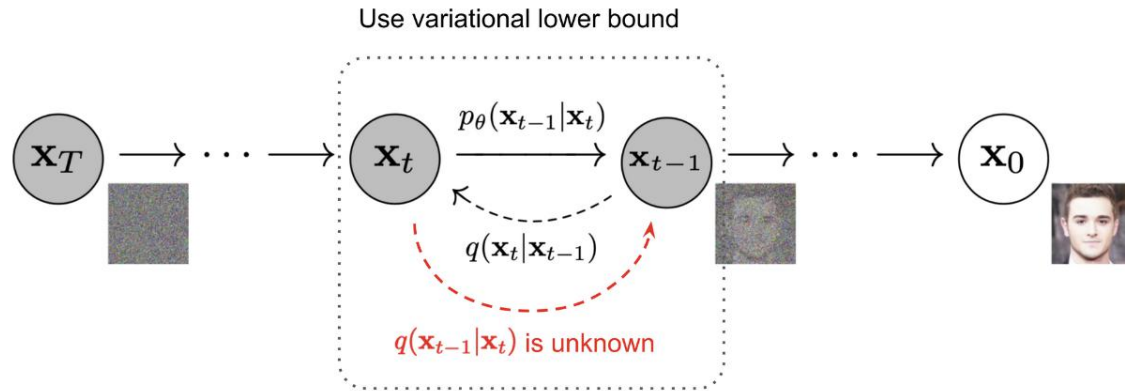
[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training



Training is done by optimizing the variational bound on negative log likelihood

$$\begin{aligned}\mathbb{E} [-\log p_\theta(\mathbf{x}_0)] &\leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \\ &= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] =: L\end{aligned}$$

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training

$$\mathbb{E} [-\log p_{\theta}(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] =: L$$
$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right]$$

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training

$$\begin{aligned}
 L &= \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \\
 &= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
 &= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right] \\
 &= \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right] \\
 &= \mathbb{E}_q \left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t > 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \\
 &= \mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t > 1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right]
 \end{aligned}$$

• Non-tractable

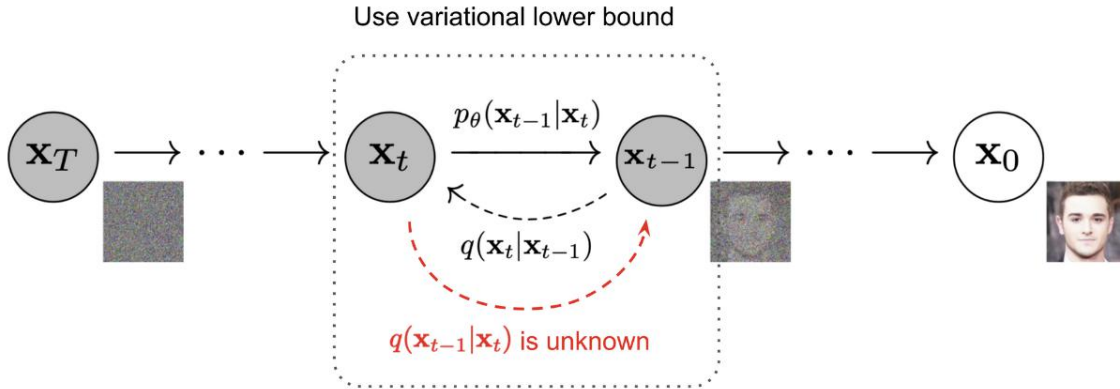
[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training



- It is shown in the [1] that reverse conditional probability is tractable when conditioned on original sample

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

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

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training

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

Since the approximate posterior q has no learnable parameters
This loss term is ignored

- This can be learnable through reparametrization.
- In practice this is fixed

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

$$\beta_1 = 10^{-4} \text{ to } \beta_T = 0.02$$

- [1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.
[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>
[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.
[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training

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

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \text{ for } 1 < t \leq T$$

Parametrization of STD

$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I} \quad \bullet \text{ This is to untrain the time dependent constants}$$

Both gives similar results

$$\sigma_t^2 = \beta_t \quad \text{Suitable for } \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\sigma_t^2 = \tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \quad \text{Suitable if our data sample is a deterministic point}$$

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

predict posterior mean that matches with

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training

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

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$$

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C$$

$$\begin{aligned} L_{t-1} - C &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon), \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \sqrt{1 - \bar{\alpha}_t} \epsilon) \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right] \end{aligned}$$

Must predict,
for given $\mathbf{x}_t$

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

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

Parametrization of Mean

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t)) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Denoising Diffusion Probabilistic Model (DDPM) - Training

Parametrization of Mean

$$\mu_{\theta}(\mathbf{x}_t, t) = \tilde{\mu}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_{\theta}(\mathbf{x}_t, t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_{\theta}(\mathbf{x}_t, t)\right)$$

function approximator intended to predict ϵ from $\mathbf{x}_t$

Sampling $\mathbf{x}_{t-1} \sim p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_{\theta}(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}, \text{ where } \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

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

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_{\theta}(\mathbf{x}_t, t)\|^2 \right] + C$$

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

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

Denoising Diffusion Probabilistic Model (DDPM) - Training Algorithm

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
$$\nabla_{\theta} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) \right\|^2$$
 - 6: **until** converged
-

Denoising Diffusion Probabilistic Model (DDPM) - Sampling Algorithm

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:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

Resembles Langevin dynamics with $\boldsymbol{\epsilon}_\theta$ as a learned gradient of the data density

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

resembles denoising score matching over multiple noise scales indexed by t

Denoising Diffusion Probabilistic Model (DDPM) - Results

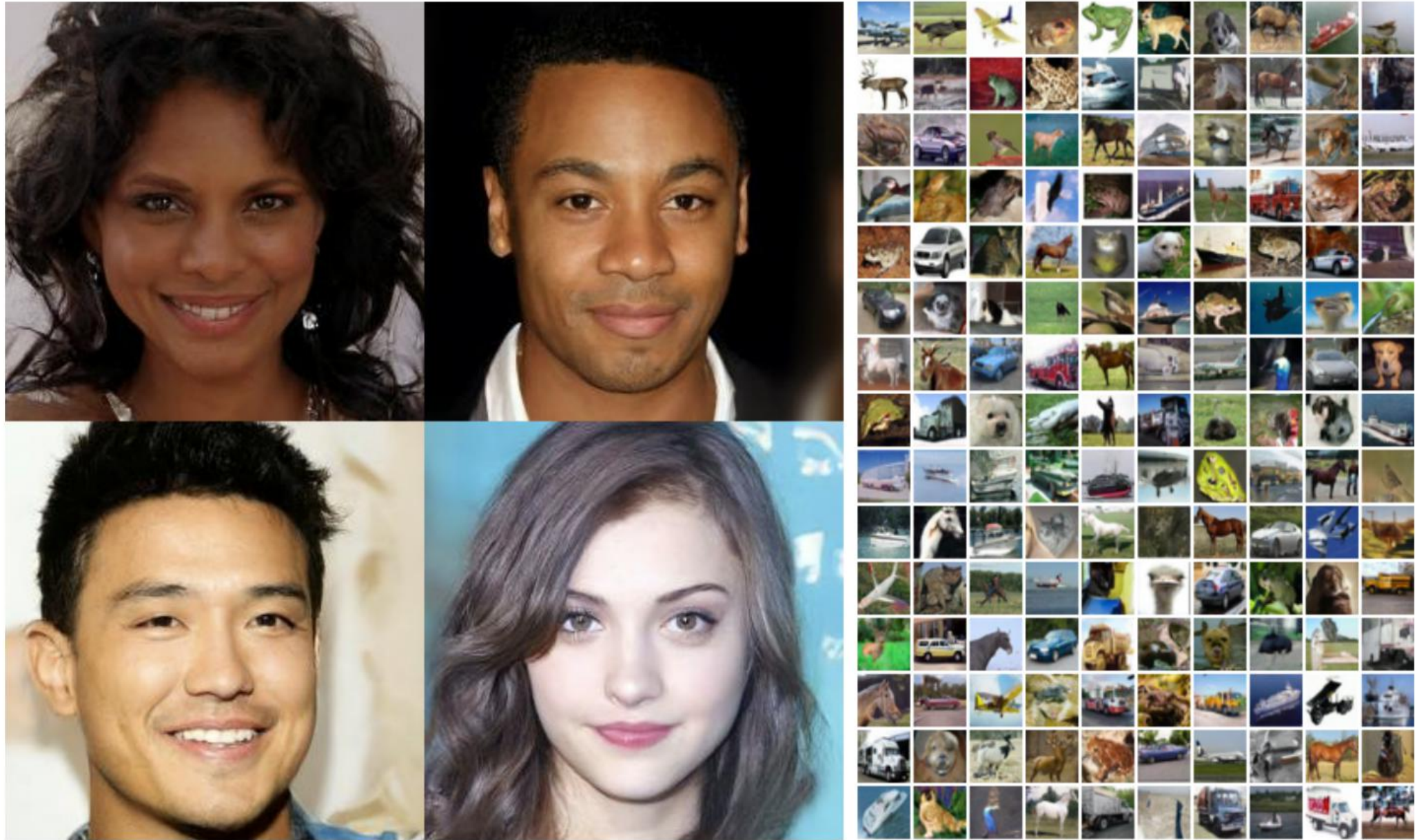


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Denoising Diffusion Probabilistic Model (DDPM) - Results

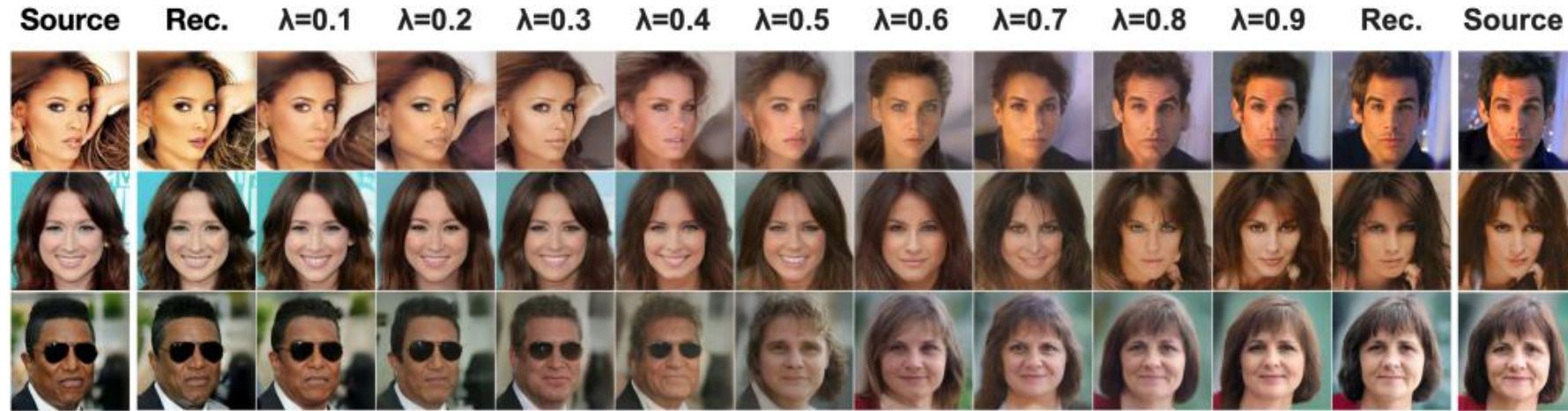
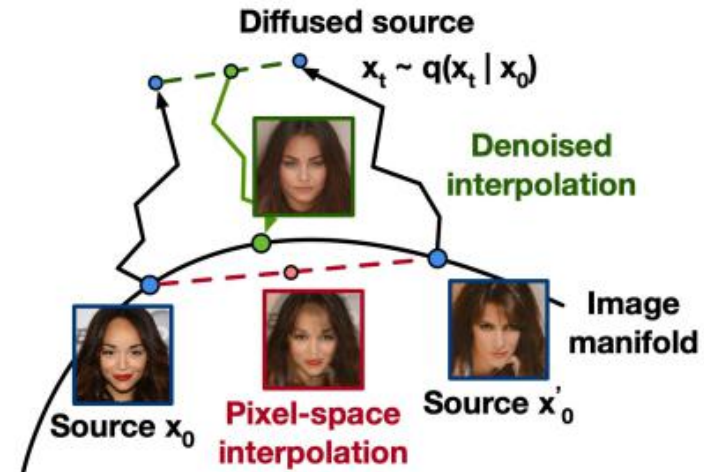


Figure 8: Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion.

Denoising Diffusion Probabilistic Model (DDPM) and NCSN

$\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$ score-based generative method learns the gradients of the data distribution using score matching

$\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2 \mathbf{I})$ Given Gaussian distribution

Gradient of log PDF

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}) = \nabla_{\mathbf{x}} \left(-\frac{1}{2\sigma^2} (\mathbf{x} - \mu)^2 \right) = -\frac{\mathbf{x} - \mu}{\sigma^2} = -\frac{\epsilon}{\sigma} \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

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

$$\begin{aligned} \mathbf{s}_\theta(\mathbf{x}_t, t) &\approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = \mathbb{E}_{q(\mathbf{x}_0)} [\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0)] \\ &= \mathbb{E}_{q(\mathbf{x}_0)} \left[-\frac{\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \right] = -\frac{\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \end{aligned}$$

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] Prince, Simon JD. Understanding deep learning. MIT press, 2023.

[4] <https://yang-song.net/blog/2021/score/>

Conditional Generation - Classifier Guided Generation

Include class information in the diffusion process

$f_\phi(y|\mathbf{x}_t, t)$ train a classifier on the noisy images

$$\nabla_{\mathbf{x}} \log f_\phi(y|\mathbf{x}_t)$$

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \quad \text{Previous slide}$$

Score function of joint distribution $q(\mathbf{x}_t, y)$

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t, y) &= \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) \\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)) \end{aligned}$$

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$$

$$\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \quad \text{Weighted classifier guidance}$$

[1] Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in neural information processing systems 34 (2021): 8780-8794.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] <https://yang-song.net/blog/2021/score/>

Conditional Generation - Classifier Guided Generation

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s .

Input: class label y , gradient scale s

$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$

for all t from T to 1 **do**

$\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$

$x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$

end for

return x_0

[1] Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in neural information processing systems 34 (2021): 8780-8794.

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] <https://yang-song.net/blog/2021/score/>

Conditional Generation - Classifier Guided Generation

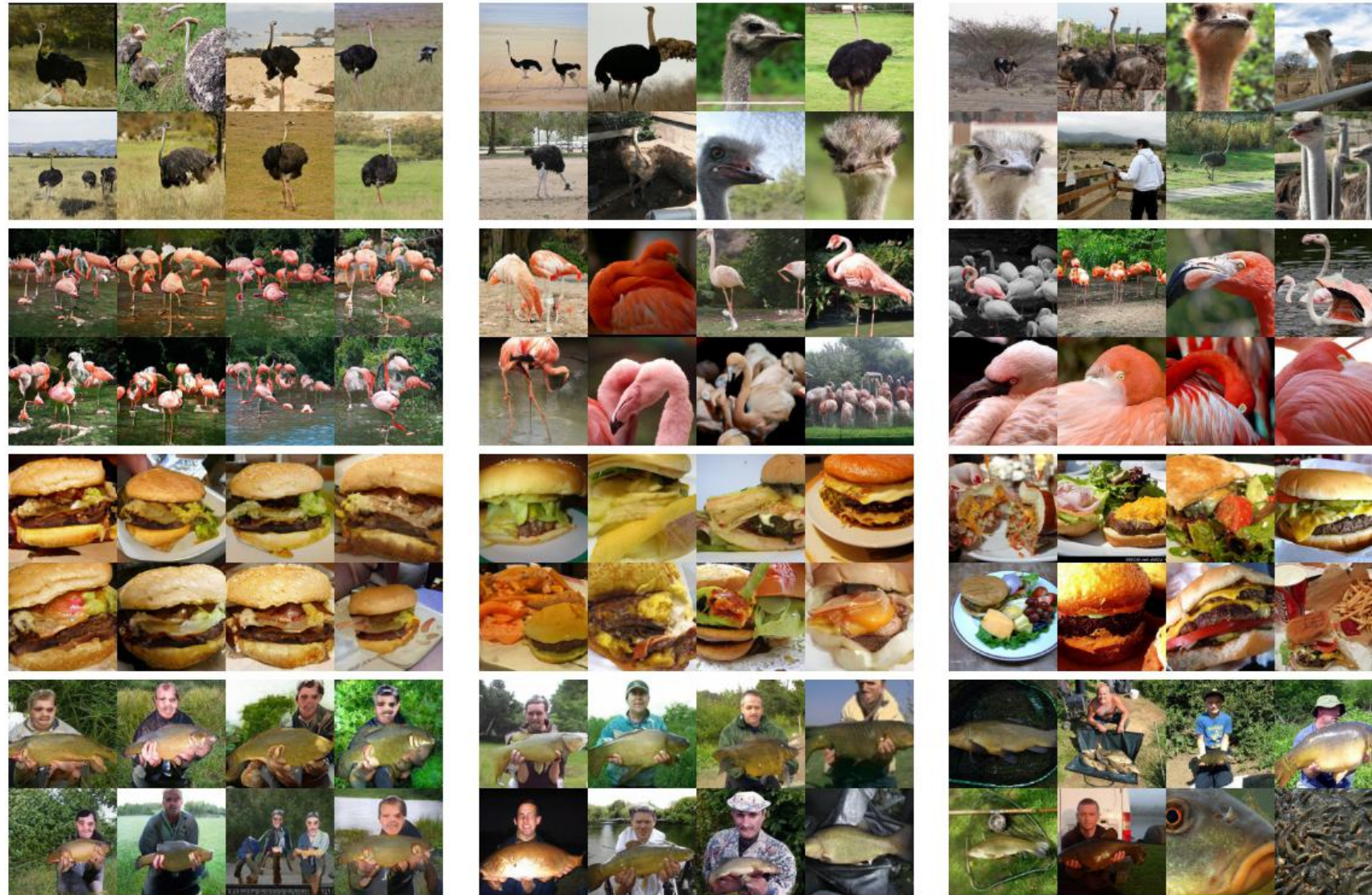


Figure 6: Samples from BigGAN-deep with truncation 1.0 (FID 6.95, left) vs samples from our diffusion model with guidance (FID 4.59, middle) and samples from the training set (right).

Conditional Generation - Classifier Free Guidance

$p_\theta(\mathbf{x})$ $\epsilon_\theta(\mathbf{x}_t, t)$ Unconditional diffusion model

$p_\theta(\mathbf{x}|y)$ $\epsilon_\theta(\mathbf{x}_t, t, y)$ Conditional diffusion model

- Learn both of them jointly using paired data

$$p_\theta(\mathbf{x}|y) \quad (\mathbf{x}, y)$$

- Periodically discard conditioning information

$$\epsilon_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t, y = \emptyset)$$

The gradient of an implicit classifier can be represented with conditional and unconditional score estimators

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1 - \bar{\alpha}_t}} \left(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t) \right) \end{aligned}$$

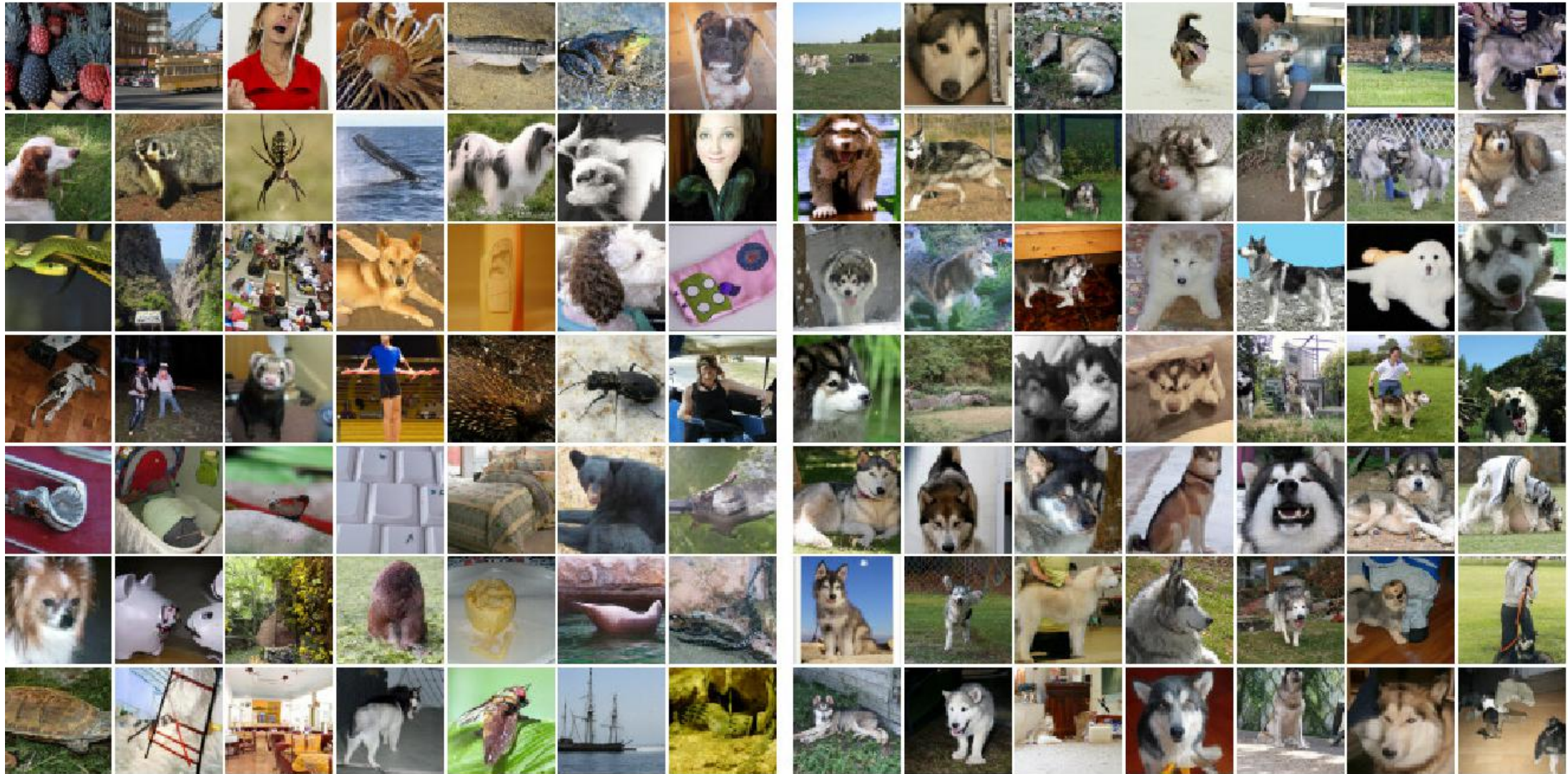
$$\begin{aligned} \bar{\epsilon}_\theta(\mathbf{x}_t, t, y) &= \epsilon_\theta(\mathbf{x}_t, t, y) - \sqrt{1 - \bar{\alpha}_t} w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\ &= \epsilon_\theta(\mathbf{x}_t, t, y) + w \left(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t) \right) \\ &= (w + 1) \epsilon_\theta(\mathbf{x}_t, t, y) - w \epsilon_\theta(\mathbf{x}_t, t) \end{aligned}$$

[1] Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." arXiv preprint arXiv:2207.12598 (2022).

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] <https://yang-song.net/blog/2021/score/>

Conditional Generation - Classifier Free Guidance



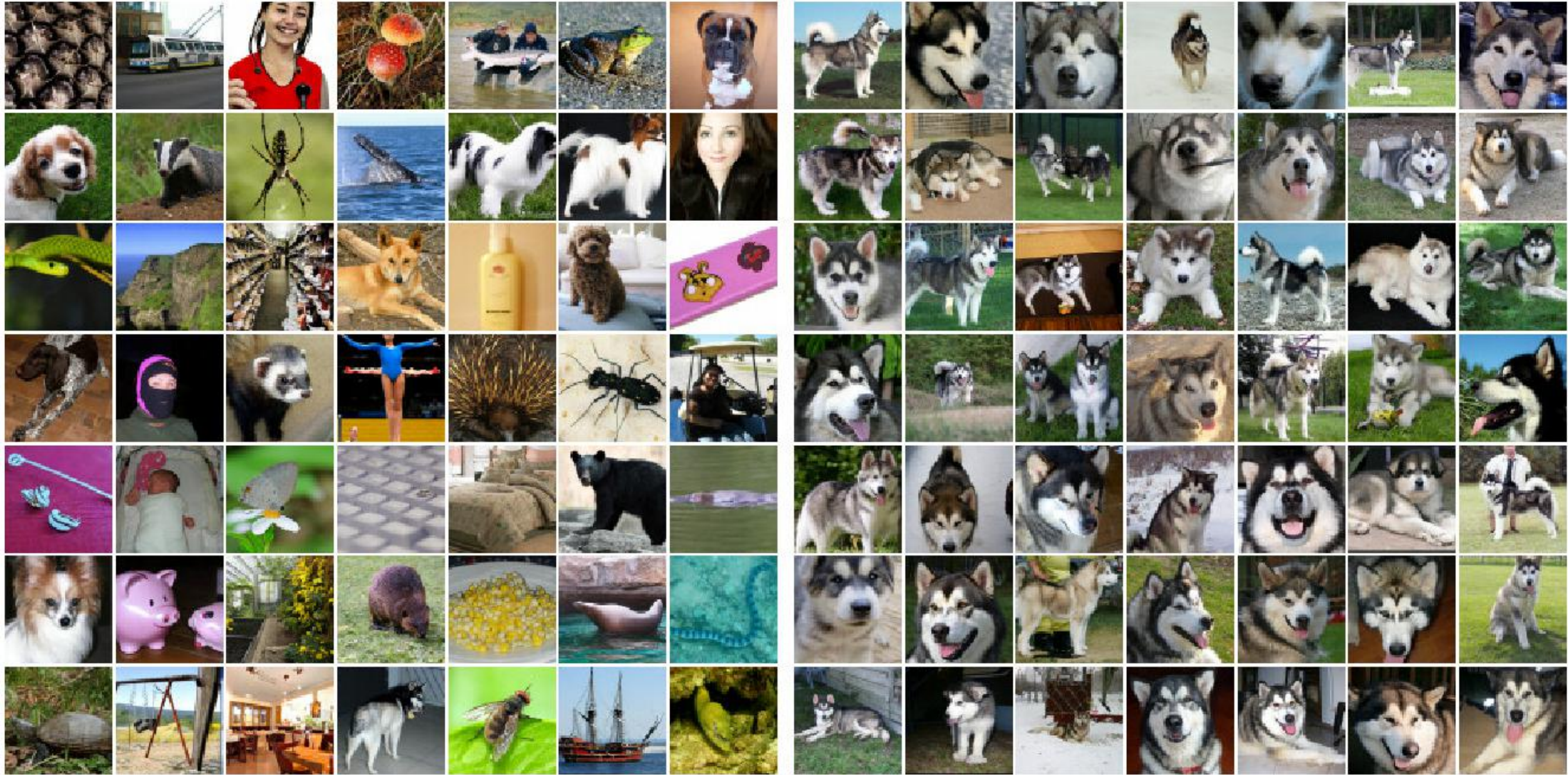
(a) Non-guided conditional sampling: FID=1.80, IS=53.71

[1] Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." arXiv preprint arXiv:2207.12598 (2022).

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] <https://yang-song.net/blog/2021/score/>

Conditional Generation - Classifier Free Guidance



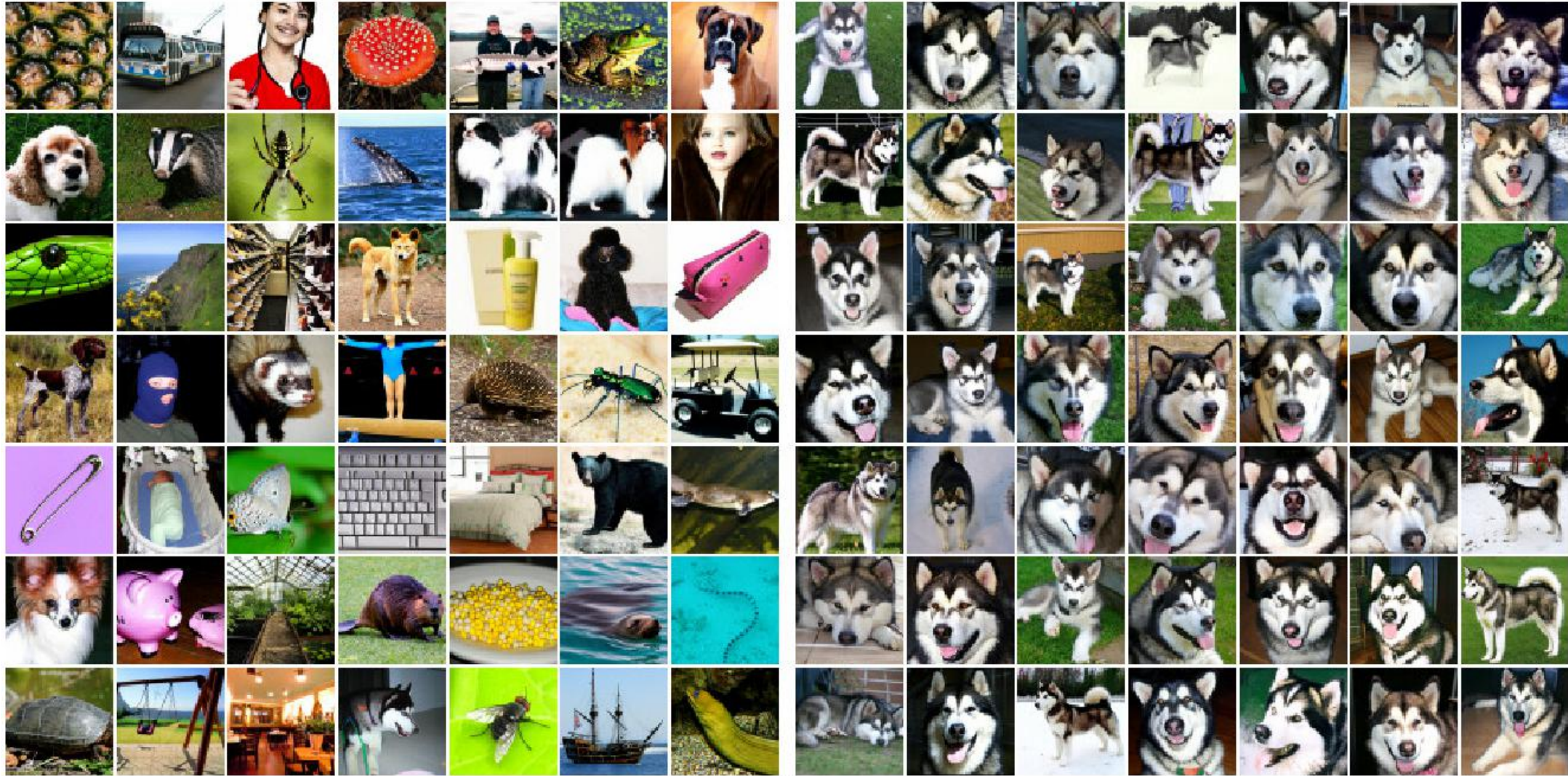
(b) Classifier-free guidance with $w = 1.0$: FID=12.6, IS=170.1

[1] Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." arXiv preprint arXiv:2207.12598 (2022).

[2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

[3] <https://yang-song.net/blog/2021/score/>

Conditional Generation - Classifier Free Guidance



(c) Classifier-free guidance with $w = 3.0$: FID=24.83, IS=250.4

- [1] Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." arXiv preprint arXiv:2207.12598 (2022).
 [2] <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>
 [3] <https://yang-song.net/blog/2021/score/>