ELEG 5491: Introduction to Deep Learning
Diffusion Models

Prof. LI Hongsheng

Office: SHB 428
e-mail: hsli@ee.cuhk.edu.hk
web: https://dl.ee.cuhk.edu.hk

Department of Electronic Engineering
The Chinese University of Hong Kong

April 2023

- There are different categories of generative neural models
- Generative Adversarial Networks (GAN) were mostly famous before, which is now replaced by diffusion models
- Diffusion models define a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise

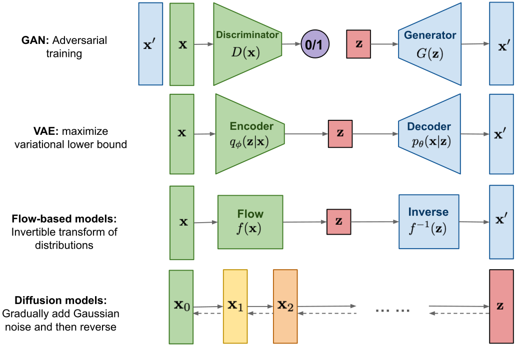


Figure: Overview of different types of generative models.

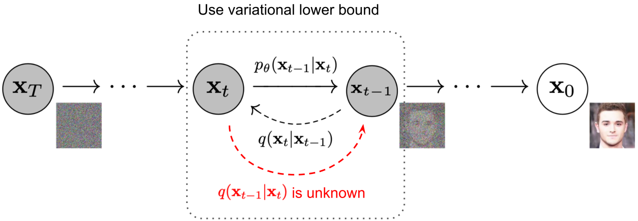
Diffusion model is the state-of-the-art generative model

Figure: Generated images by Midjourney v4.

Forward Diffusion Process (adding noise)

- Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$ define a forward diffusion process in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$
- The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$

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



- The data sample $\mathbf{x}_0$ gradually loses its distinguishable features as the step t becomes larger
- Eventually when $T \to \infty, \mathbf{x}_T$ is equivalent to an isotropic Gaussian distribution.

Forward Diffusion Process

- We can sample $\mathbf{x}_t$ at any arbitrary time step t in a closec form using reparameterization trick
- Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}; && \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t \alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\bar{\epsilon}_{t-2}; && \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians } (*) \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\
q(\mathbf{x}_t | \mathbf{x}_0) &= \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)
\end{aligned}
$$

- We also have $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t\right)$
- (*) When merging two Gaussians $\mathcal{N}\left(\mathbf{0}, \sigma_1^2\mathbf{I}\right)$ and $\mathcal{N}\left(\mathbf{0}, \sigma_2^2\mathbf{I}\right)$, the new distribution is $\mathcal{N}\left(\mathbf{0}, \left(\sigma_1^2 + \sigma_2^2\right)\mathbf{I}\right)$. Here the merged standard deviation is

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

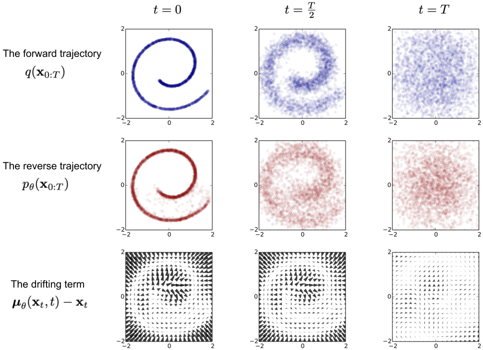
- Usually, a larger update step is used, when the sample gets noiser, $\beta_1 < \beta_2 < \cdots < \beta_T$ and therefore $\bar{\alpha}_1 > \cdots > \bar{\alpha}_T$

Reverse diffusion process (removing noise)

- If we can reverse the above process and sample from $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$, we can recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Note that if β_t is small enough, $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$ will also be Gaussian
- Unfortunately, we cannot easily estimate $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$
- We therefore need to learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process

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

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



- the reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

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

- Using Bayes' rule:

$$\begin{aligned}
q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right) &= q\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0\right)\frac{q\left(\mathbf{x}_{t-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_t|\mathbf{x}_0\right)}\\
&\propto \exp\left(-\frac{1}{2}\left(\frac{\left(\mathbf{x}_t-\sqrt{\alpha_t}\mathbf{x}_{t-1}\right)^2}{\beta_t}+\frac{\left(\mathbf{x}_{t-1}-\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_{t-1}}-\frac{\left(\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2-2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1}+\alpha_t\mathbf{x}_{t-1}^2}{\beta_t}+\frac{\mathbf{x}_{t-1}^2-2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1}+\bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}}-\frac{\left(\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2-\left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t+\frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1}+C\left(\mathbf{x}_t,\mathbf{x}_0\right)\right)\right)
\end{aligned}$$

where $C\left(\mathbf{x}_t,\mathbf{x}_0\right)$ is some function not involving $\mathbf{x}_{t-1}$ and details are omitted

Reverse diffusion process

- Following the standard Gaussian density function, the mean and variance can be parameterized as follows ($\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$)

$$\tilde{\beta}_t = 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = 1/\left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t\left(1-\bar{\alpha}_{t-1}\right)}\right) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right) \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\
&= \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0
\end{aligned}$$

- We can represent $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t\right)$ and plug it into the above equation and obtain

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t\right) \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right)
\end{aligned}$$

- Recall the VAE lower bound is modeled as

$$-L_{\mathrm{VAE}} = \log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})\right) \leq \log p_\theta(\mathbf{x})$$

- The setup is very similar to VAE and thus we can use the variational lower bound to optimize the negative log-likelihood

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

- The objective can be further rewritten to be a combination of several KL-divergence and entropy terms

Reverse diffusion process

$$
\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \mathbb{E}_q\left[\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)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log\left(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\right) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log 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)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log 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 \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[\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)\right] \\
&= \mathbb{E}_q[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}
$$

Reverse diffusion process

- We can label each component in the variational lower bound loss separately

$$
\begin{aligned}
L_{\mathrm{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_T\right)\right) \\
L_t &= D_{\mathrm{KL}}\left(q\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}\right)\right) \text{ for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)
\end{aligned}
$$

- Every KL term in L_{VLB} (except for L_0) compares two Gaussian distributions and therefore they can be computed in closed form (as shown in the chapter of VAE)
- L_T is constant and be ignored during training because $\mathbf{q}$ has no learnable parameters
- $\mathbf{x}_T$ is a Gaussian noise. Ho et al. 2020 models L_0 using a separate discrete decoder derived from $\mathcal{N}\left(\mathbf{x}_0; \boldsymbol{\mu}_\theta\left(\mathbf{x}_1, 1\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_1, 1\right)\right)$

Parameterization of L_t for Training Loss

- Recall that we need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta\left(\mathbf{x}_{t-1} | \mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$
- We train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$
- Because $\mathbf{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict $\boldsymbol{\epsilon}_t$ from the input $\mathbf{x}_t$ at time step t:

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

Thus $\mathbf{x}_{t-1} = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$

Parameterization of L_t for Training Loss

- The loss term L_t is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$:

$$
\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\left\|\boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t,t\right)\right\|_2^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t,\mathbf{x}_0\right)-\boldsymbol{\mu}_\theta\left(\mathbf{x}_t,t\right)\right\|^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\left\|\boldsymbol{\Sigma}_\theta\right\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)-\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t,t\right)\right)\right\|^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\left(1-\alpha_t\right)^2}{2\alpha_t\left(1-\bar{\alpha}_t\right)\left\|\boldsymbol{\Sigma}_\theta\right\|_2^2}\left\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t,t\right)\right\|^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\left(1-\alpha_t\right)^2}{2\alpha_t\left(1-\bar{\alpha}_t\right)\left\|\boldsymbol{\Sigma}_\theta\right\|_2^2}\left\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t,t\right)\right\|^2\right]
\end{aligned}
$$

Parameterization of L_t for Training Loss

- **Simplification:** Ho et al. (2020) found that training the diffusion model works better with a simplified objective that ignores the weighting term

$$
\begin{aligned}
L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right] \\
&= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t\right)\right\|^2\right]
\end{aligned}
$$

- The final simplified objective is

$$
L_{\text{simple}} = L_t^{\text{simple}} + C
$$

C is a constant not depending on θ

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Figure: The training and sampling algorithms in DDPM (Ho et al. 2020).

Parameterization of β_t

- The forward variances are set to be a sequence of linearly increasing constants in Ho et al. (2020), from from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$, which are small compared with normalized pixel values in $[-1, 1]$
- Diffusion models in their experiments showed high-quality samples but still could not achieve competitive model log-likelihood as other generative models
- Nichol & Dhariwal (2021) proposed several improvement techniques to improve diffusion models. One of the improvements is to use a cosine-based variance schedule:

$$\beta_t = \text{clip}\left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999\right) \quad \bar{\alpha}_t = \frac{f(t)}{f(0)} \quad \text{where } f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2$$

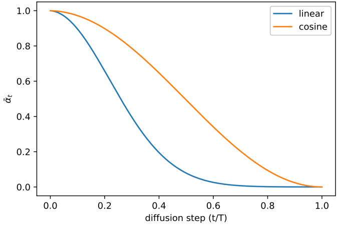
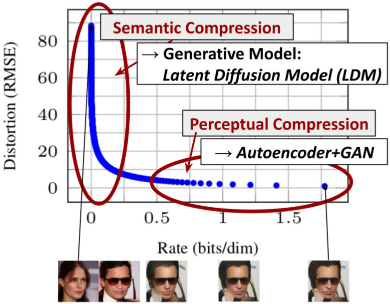


Figure: Comparison of linear and cosine-based scheduling of β_t during training.

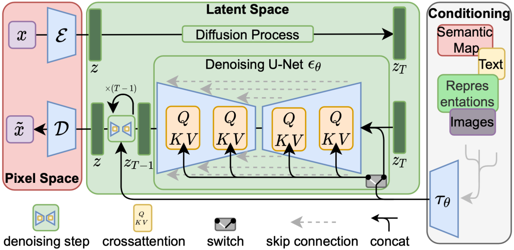
Latent diffusion model

- Latent diffusion model (LDM; Rombach & Blattmann, et al. 2022) runs the diffusion process in the latent space instead of pixel space, making training cost lower and inference speed faster
- Most bits of an image contribute to perceptual details and the semantic and conceptual composition still remains after aggressive compression
- LDM loosely decomposes the perceptual compression and semantic compression with generative modeling learning by first trimming off pixel-level redundancy with autoencoder and then manipulate/generate semantic concepts with diffusion process on learned latent



Latent diffusion model

- The perceptual compression process relies on an autoencoder model
- An encoder $\mathcal{E}$ compress the input image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ to a smaller $2D$ latent vector $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{h \times w \times c}$, where the downsampling rate $f = H/h = W/w = 2^m, m \in \mathbb{N}$
- The downsampling rate $f = H/h = W/w = 2^m, m \in \mathbb{N}$. Then an decoder $\mathcal{D}$ reconstructs the images from the latent vector, $\tilde{\mathbf{x}} = \mathcal{D}(\mathbf{z})$
- Two types of regularization in autoencoder training are used to avoid arbitrarily high-variance in the latent spaces
  - **KL-reg:** A small KL penalty towards a standard normal distribution over the learned latent, similar to VAE.
  - **VQ-reg** uses a vector quantization layer within the decoder, like VQVAE but the quantization layer is absorbed by the decoder



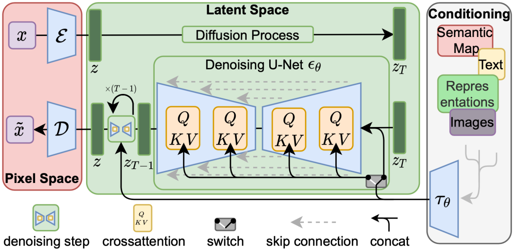
Latent diffusion model

- The diffusion and denoising processes happen on the latent vector $\mathbf{z}$
- The denoising model is a time-conditioned U-Net, augmented with the cross-attention to handle flexible conditioning information(e.g. class labels, semantic maps, blurred variants of an image).)
- Each type of condition is paired with a domain-specific encoder τ_θ to project the conditioning input y to an intermediate representation that can be mapped into cross-attention component, $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \cdot \mathbf{V}$$

where $\mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i), \mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y), \mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y)$

and $\mathbf{W}_Q^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_\tau}, \varphi_i(\mathbf{z}_i) \in \mathbb{R}^{N \times d_\epsilon^i}, \tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$



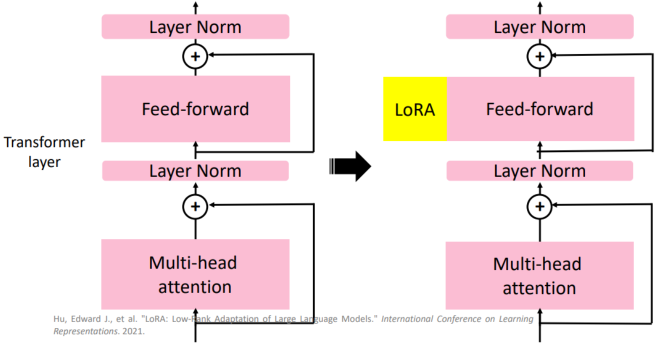
Stable Diffusion

- Stable Diffusion is a latent text-to-image diffusion model
- It is trained on 512×512 images from a subset of the LAION-5B database
- The model uses a frozen CLIP ViT-L/14 text encoder to condition the model on text prompts
- It uses a 860M UNet and 123M text encoder, is relatively lightweight, and can run on a GPU with at least 10GB memory

Figure: Results of Stable Diffusion model.

LoRA: Low-Rank Adaptation

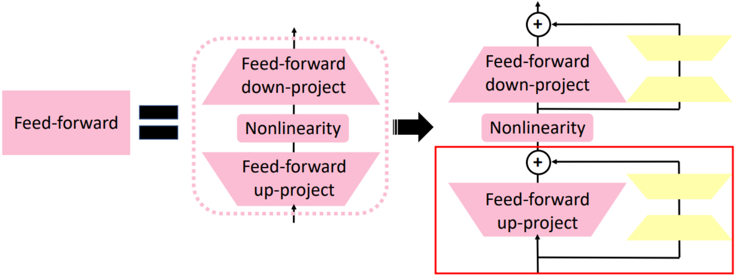
- LoRA was originally introduced for fine-tuning large-language models, which are too expensive to be fine-tuned (e.g., GPT-3 has billions of parameters)
- LoRA proposes to freeze pre-trained model weights and inject trainable layers (rank-decomposition matrices) in each transformer block
- LoRA can be applied to the cross-attention layers that relate the image generation with text prompts



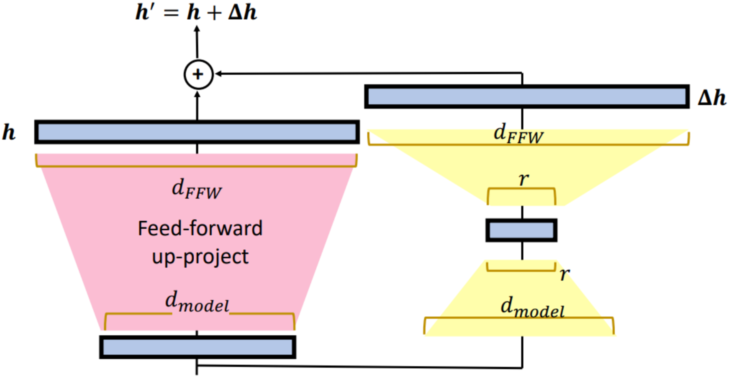
Hu, Edward J., et al. "LoRA: Low-Rank Adaptation of Large Language Models." *International Conference on Learning Representations*. 2021.

LoRA: Low-Rank Adaptation

- Additive sub-networks are added to each of the feedfoward layer
- During finetuning, residual features are predicted by the new sub-networks



- Additive sub-networks are added to each of the feedfoward layer
- During finetuning, residual features are predicted by the new sub-networks

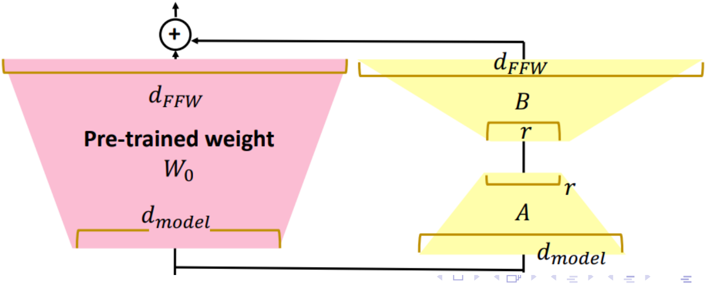


LoRA: Low-Rank Adaptation

- After finetuing, the FFN weights W are updated as

$$W = W + \alpha \Delta W = W + \alpha BA, \text{ rank } r \ll \min\left(d_{FFW}, d_{\text{model}}\right)$$

  $\alpha \in [0, 1]$ is a hyper-parameter
- The Stable Diffusion with LoRA finetuing allows finetuning with just a few new training images
- Trained weights are much, much smaller. Because the original model is frozen and we inject new layers to be trained, we can save the weights for the new layers as a single file that weighs in at 3 MB in size. This is about one thousand times smaller than the original size of the UNet model



- **Pros:** Tractable models can be analytically evaluated and cheaply fit data, but they cannot easily describe the structure in rich datasets
- Flexible models can fit arbitrary structures in data, but evaluating, training, or sampling from these models is usually expensive
- Diffusion models are both analytically tractable and flexible
- **Cons:** Diffusion models rely on a long Markov chain of diffusion steps to generate samples, so it can be quite expensive in terms of time and compute

References

- https://d223302.github.io/AACL2022-Pretrain-Language-Model-Tutorial/lecture_material/AACL_2022_tutorial_PLMs.pdf
- https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- Jonathan Ho et al. "Denoising diffusion probabilistic models." arxiv Preprint arxiv:2006.11239, 2020.
- Yang Song & Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." NeurIPS 2019.
- Rombach & Blattmann, et al. "High-Resolution Image Synthesis with Latent Diffusion Models." CVPR 2022.
- Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models", arXiv:2106.09685, 2021.