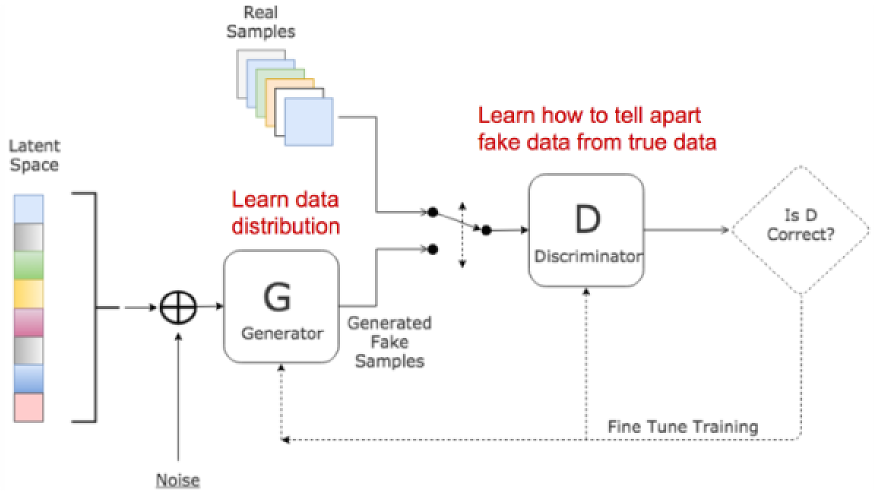
目录

- Recap: GAN, VAE and Flow-based Models
- Intro to Diffusion Models
- Dive into Diffusion Models
- Advanced Topics of Diffusion Models
- Applications and Tools

目录

- Recap: GAN, VAE and Flow-based Models
- Intro to Diffusion Models
- Dive into Diffusion Models
- Advanced Topics of Diffusion Models
- Applications and Tools

Recap: **GAN**, VAE and Flow-based Models

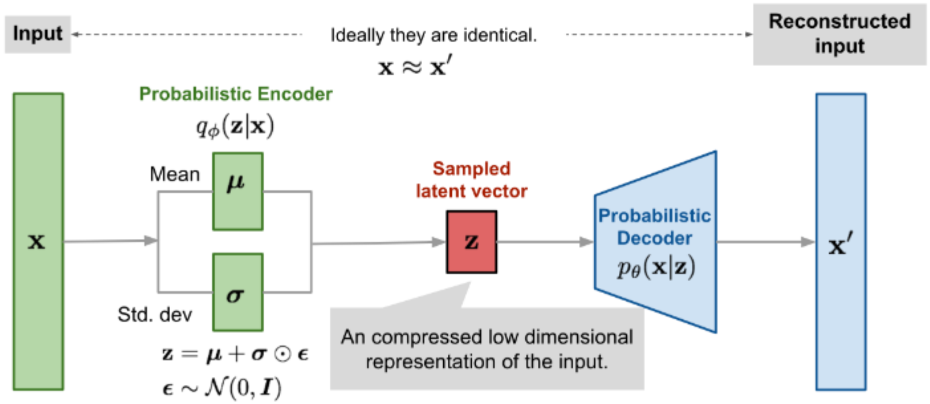


- A discriminator D estimates the probability of a given sample coming from the real dataset or the synthesized data
- A generator G learns to capture the real data distribution so that it synthesizes samples as real as possible
- D and G are playing a minimax game in which we wish to optimize the following loss function:

$$\begin{aligned}\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]\end{aligned}$$

- Problems: hard to achieve Nash equilibrium, low dimensional supports, vanishing gradient, mode collapse
- Improvements: feature matching, minibatch discrimination, historical averaging, label smoothing, virtual batch normalization, adding noise, using better distribution similarity (WGAN)

Recap: GAN, **VAE** and Flow-based Models



- In VAEs, the input data is sampled from a parametrized distribution and the encoder and decoder are jointly trained such that the output minimizes a reconstruction error in the sense of KL divergence between the posterior and its parametric approximation
- It minimizes the following lower bound (ELBO) with the reparameterization trick:

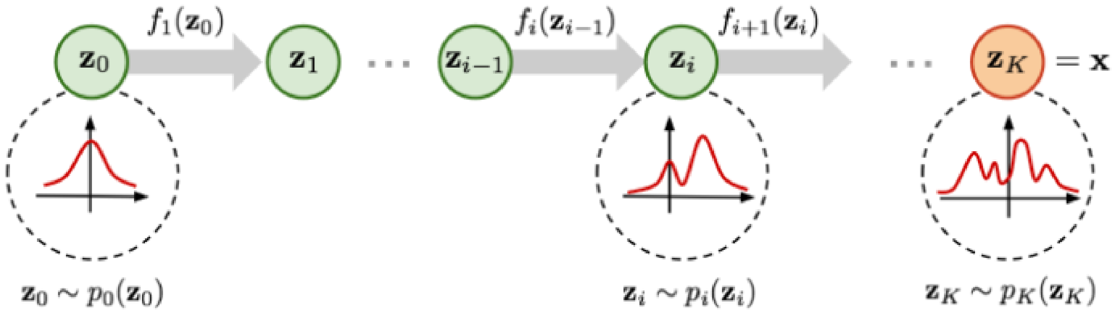
$$
\begin{aligned}
L_{\mathrm{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)} \boldsymbol{I}) \\
&\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}) \qquad \text{; Reparameterization trick.}
\end{aligned}
$$

- Improvements: Beta-VAE (disentangled latent factors), VQ-VAE, TD-VAE (for sequential data)

$$
L_{\mathrm{BETA}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))
$$

Recap: GAN, VAE and **Flow-based Models**



- A flow-based model is constructed by a sequence of invertible transformations. It explicitly learns the data distribution and therefore the loss function is simply the negative log-likelihood
- Flow-based models use the following process to calculate the data distribution:

$$
\begin{aligned}
\mathbf{x} &= \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) \\
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left|\det \frac{df_{K-1}}{d\mathbf{z}_{K-2}}\right| - \log \left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|
\end{aligned}
$$

For function f:
1) It is easily invertible
2) Its Jocobian determinant is easy to compute

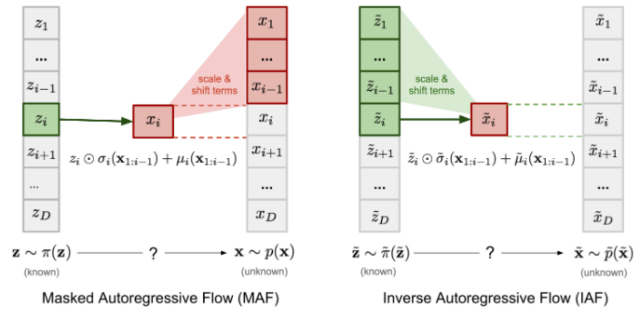
- Implementations:

$$
\begin{aligned}
\mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\
\mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})
\end{aligned}
$$

RealNVP

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j: \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j: \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W}: [c \times c]$. See Section 3.2.	$\forall i,j: \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j: \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

Glow



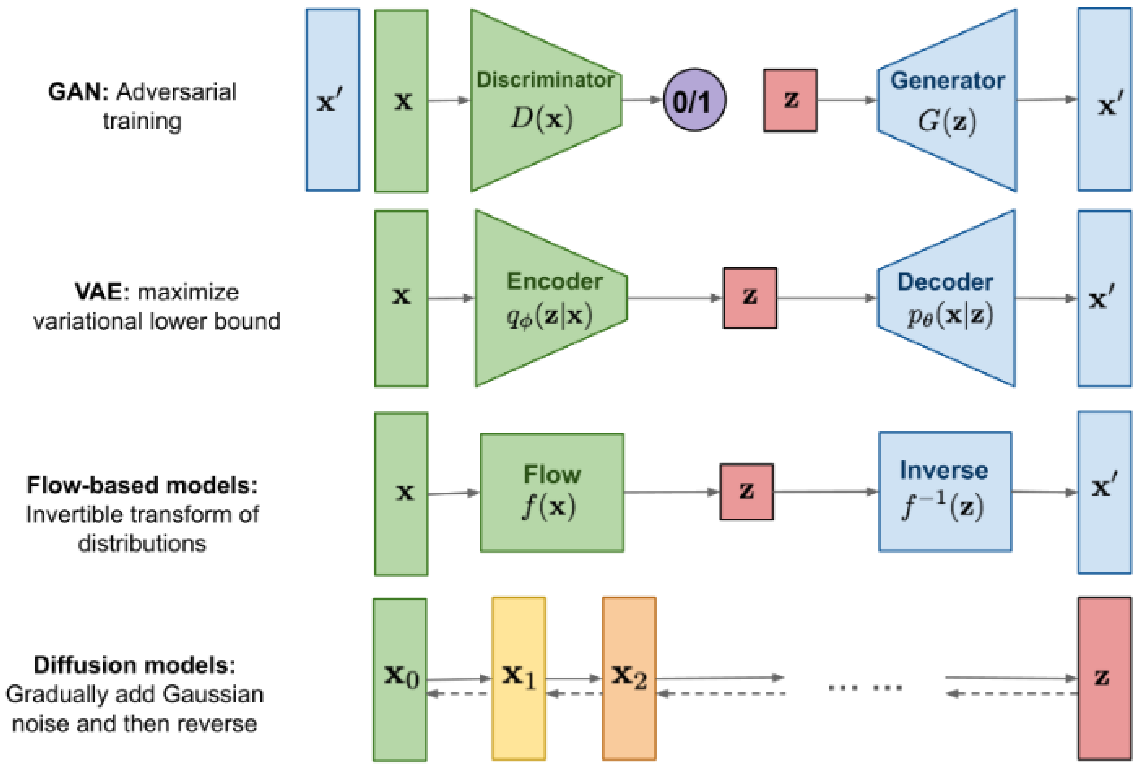
MAF & IAF

目录

- Recap: GAN, VAE and Flow-based Models
- Intro to Diffusion Models
- Dive into Diffusion Models
- Advanced Topics of Diffusion Models
- Applications and Tools

Intro to Diffusion Models

- GANs suffer from unstable training and are bad at generating diverse results; VAEs rely on a surrogate loss; Flow-based models have to use specified architectures to construct reversible transform
- Diffusion models slowly inject random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise, at the benefits of a fixed procedure and high-dimensional latent variables



目录

- Recap: GAN, VAE and Flow-based Models
- Intro to Diffusion Models
- Dive into Diffusion Models
- Advanced Topics of Diffusion Models
- Applications and Tools

Dive into Diffusion Models

- ***Forward diffusion***: iteratively add small amount of Gaussian noise, producing a sequence of noisy samples

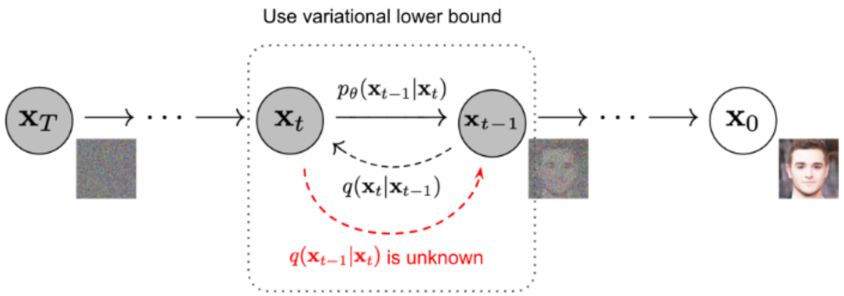
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

- We can sample data at any time step t using reparameterization trick:

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

- We can afford a larger update step when the sample gets noiser:

$$\beta_1 < \beta_2 < \dots < \beta_T \qquad \bar{\alpha}_1 > \dots > \bar{\alpha}_T.$$



Dive into Diffusion Models

- ***Reverse diffusion*** : recover the real data from Gaussian noise by learning a model to approximate the probability

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

*Note: we cannot reversely use the reparameterization trick to sample $\mathbf{x}_{t-1}$ from $\mathbf{x}_t$ because a final probability value is needed to calculate the loss

- In order to make the reverse conditional probability tractable, we further condition it on $\mathbf{x}_0$:

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

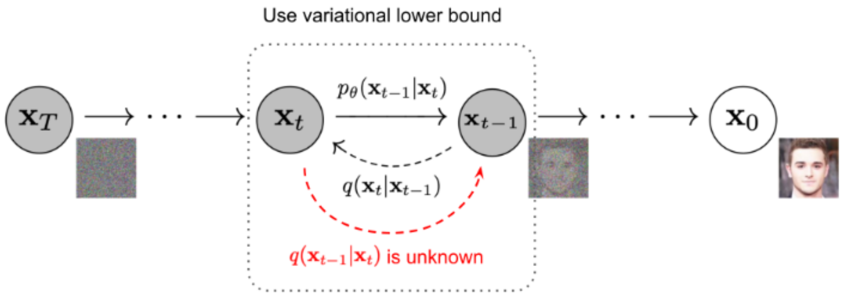
Using Bayes' rule, we have:

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

Therefore we have:

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

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



You can verify the expression in exp is a square number

Dive into Diffusion Models

- The reparameterization trick implies $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t)$, we can plug it into the mean:

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

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

- We can use the variational lower bound similar to VAEs to optimize the NLL:

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

The right-hand side is the standard cross entropy loss

Dive into Diffusion Models

$$\text{Let } L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)$$

- The VLB can be further decomposed into a combination of several KL divergence losses:

$$\begin{aligned}
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 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)}\Big] \\
&= \mathbb{E}_q\Big[-\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)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \Big(\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)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\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)}\Big] \\
&= \mathbb{E}_q\Big[-\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)}\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-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}$$

which leads to:

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

Dive into Diffusion Models

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

- The next question is how to parameterize L_t. We would like to train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big)$
- Because $\mathbf{x}_t$ is available during training, we can instead make the model to predict $\boldsymbol{\epsilon}_t$ from $\mathbf{x}_t$:

$$
\begin{aligned}
\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big) \\
\text{Thus } \mathbf{x}_{t-1} &= \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))
\end{aligned}
$$

- Then, L_t is parameterized to minimize the difference:

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

- It can be simplified to make the diffusion model work better:

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

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 $\mathbf{x}_0$

Dive into Diffusion Models

- The variance matrix in $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)=\mathcal{N}(\mathbf{x}_{t-1};\tilde{\boldsymbol{\mu}}(\mathbf{x}_t,\mathbf{x}_0),\tilde{\beta}_t\mathbf{I})$ is found to be important for stable training. An effective method is to interpolate between β_t and $\tilde{\beta}_t$ by predicting a mixing vector $\mathbf{v}$:

$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)=\exp(\mathbf{v}\log\beta_t+(1-\mathbf{v})\log\tilde{\beta}_t)$$

- However, L_t^{simple} does not depend on $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)$. A feasible way is to combine L_t^{simple} with L_{VLB}, leading to $L_{\text{hybrid}}=L_{\text{simple}}+\lambda L_{\text{VLB}}$ with small λ. A time-averaging smoothed version of L_{VLB} is leveraged to better optimize the loss.

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

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

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

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

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0\sim q(\mathbf{x}_0)$
3: $t\sim\text{Uniform}(\{1,\ldots,T\})$
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

Algorithm 2 Sampling

1: $\mathbf{x}_T\sim\mathcal{N}(\mathbf{0},\mathbf{I})$
2: **for** $t=T,\ldots,1$ **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** $\mathbf{x}_0$

目录

- Recap: GAN, VAE and Flow-based Models
- Intro to Diffusion Models
- Dive into Diffusion Models
- Advanced Topics of Diffusion Models
- Applications and Tools

Advanced Topics of Diffusion Models

Speed up Diffusion Model Samping

- It could be very slow to generate samples following the Markov chain of the reverse diffusion process as T can be up to few thousands of steps.
- One simple way is to use a strided sampling schedule, i.e., update every $\lceil T/S \rceil$ steps to reduce the process from T to S steps.
- Another approach is to re-parameterize $\mathbf{x}_t$ as:

$$\begin{aligned}
\mathbf{x}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\boldsymbol{\epsilon}_t + \sigma_t\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\boldsymbol{\epsilon} \\
q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})
\end{aligned}$$

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

- Let $\sigma_t^2 = \eta \cdot \tilde{\beta}_t$ such that we can control the sampling stochasticity. $\eta = 0$ makes the model deterministic. Such a model is called *denoising diffusion implicit model* (DDIM). It maps noise back to the original data samples.
- During inference, we only sample S diffusion steps and the process becomes(*):

$$q_{\sigma,\tau}(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_t}, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{\tau_{i-1}}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_{\tau_i} - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})$$

(*) Currently I do not fully understand the mechanism behind DDIM. For readers who are interested in DDIM, please refer to the original paper: https://arxiv.org/abs/2010.02502

Advanced Topics of Diffusion Models

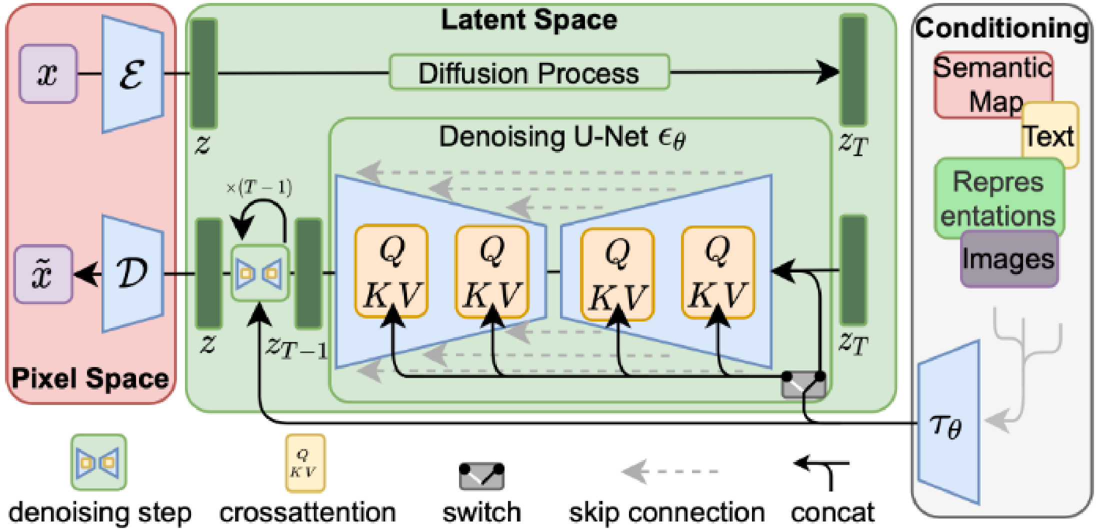
Speed up Diffusion Model Samping

- Latent diffusion model (LDM) runs the diffusion process in the latent space rather than pixel space, making training cost lower and inference speed faster. It first trims off pixel-level redundancy with autoencoder and then manipulate /generate semantic concepts with diffusion process on learned latent.
- More concretely, an encoder E is employed to compress the input image to a latent vector $\mathbf{z} = E(\mathbf{x})$. Then a decoder D reconstructs the image from the latent vector $\mathbf{x}' = D(\mathbf{z})$. The diffusion and denoising processes happen on the latent vector $\mathbf{z}$.

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

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

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



Memory-efficient array redistribution through portable collective communication, https://arxiv.org/abs/2112.01075

Advanced Topics of Diffusion Models

Conditional Generation

- To explicitly incorporate class info into the diffusion process, we can train a classifier $f_\phi(y|\mathbf{x}_t, t)$ on noisy image $\mathbf{x}_t$ and use gradients $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t)$ to guide the diffusion sampling process toward the conditioning information y.
- The joint distribution $q(\mathbf{x}_t, y)$ can be rewritten as follows:

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

- A new classifier-guided predictor $\bar{\boldsymbol{\epsilon}}_\theta$ would take the following form:

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

- The resulting model is called the *ablated diffusion model* (ADM).

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, 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 $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$
end for
return x_0

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $f_\phi(y|x_t)$, 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**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log f_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

Diffusion Models Beat GANs on Image Synthesis, https://arxiv.org/abs/2105.05233

Advanced Topics of Diffusion Models

Conditional Generation

- The unconditional denoising diffusion model $p_\theta(\mathbf{x})$ parameterized through a score estimator $\epsilon_\theta(\mathbf{x}_t, t)$ and the conditional model $p_\theta(\mathbf{x}|y)$ parameterized through $\epsilon_\theta(\mathbf{x}_t, t, y)$. These two models can be learned via a single NN.
- Precisely, $p_\theta(\mathbf{x}|y)$ is trained on paired data, where the label gets discarded periodically at random such that the model knows how to generate images unconditionally, i.e., $\epsilon_\theta(\mathbf{x}_t, t) = \epsilon_\theta(\mathbf{x}_t, t, y = \varnothing)$.

- The gradient can be derived as:

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

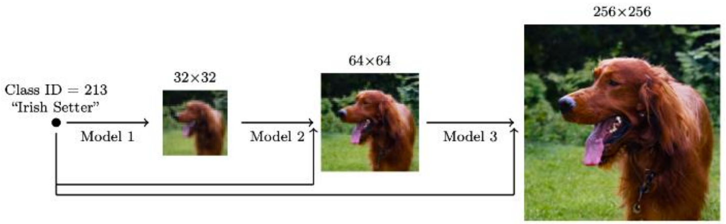
- The gradient is represented with conditional and unconditional score estimates, which contain no dependency on a separate classifier.

Classifier-Free Diffusion Guidance, https://openreview.net/pdf?id=qw8AKxfYbI
GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, https://arxiv.org/abs/2112.10741

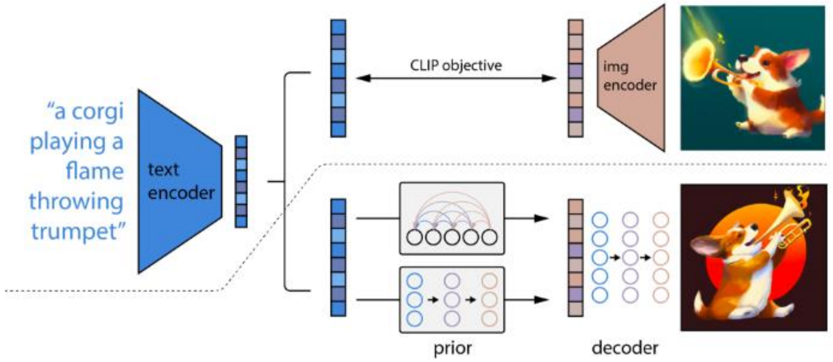
Advanced Topics of Diffusion Models

Scale up Generation Resolution and Quality

- We can use a pipeline of multiple diffusion models at increasing resolutions to generate high-quality images. *Noise conditioning augmentation* between pipeline models is crucial to the final quality. It helps reduce compounding error in the pipeline setup. U-net is a common choice of model architecture.



- The two-stage diffusion model unclip utilizes CLIP text encoder to produce text-guided images at high quality. It learns two models in parallel:
  - A prior model $P(\mathbf{c}^i|y)$ outputs CLIP image embedding given the text
  - A decoder $P(\mathbf{x}|\mathbf{c}^i, [y])$ generates the image given CLIP image embedding and optionally the original text



Cascaded Diffusion Models for High Fidelity Image Generation, https://arxiv.org/abs/2106.15282
Hierarchical Text-Conditional Image Generation with CLIP Latents, https://arxiv.org/abs/2204.06125

Advanced Topics of Diffusion Models

Scale up Generation Resolution and Quality

- Imagen uses a pretrained LM to encode text for image generation, such as T5-XXL
- When applying classifier-free guidance, increasing w may lead to better image-text alignment but worse image fidelity because of train-test mismatch
- To mitigate this issue, two thresholding strategies are introduced:
  - Static thresholding: clip $\mathbf{x}$ prediction to [-1, 1]
  - Dynamic thresholding: at each step, compute s as a certain percentile absolute pixel value; if $s > 1$, clip the prediction to $[-s, s]$ and divide by s

- Imagen modifies several designs in U-Net to make it efficient U-Net:
  - Shift model parameters from high resolution blocks to low resolution by adding more residual locks for the lower resolutions
  - Scale the skip connections by $\frac{1}{\sqrt{2}}$
  - Reverse the order of downsampling (move it before convolutions) and upsampling operations (move it after convoluions) in order to improve the speed of forward pass

Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding, https://arxiv.org/abs/2205.11487

目录

- Recap: GAN, VAE and Flow-based Models
- Intro to Diffusion Models
- Dive into Diffusion Models
- Advanced Topics of Diffusion Models
- Applications and Tools: AI Art Generator

Applications and Tools: AI Art Generator

Recent AI Art Generators

- 近段时间来，AI绘画成为游戏圈的一大热点，基于深度学习生成模型，AI绘画工具能够根据用户输入的文本、关键词和参数生成风格多样的高质量2D图像

Applications and Tools: AI Art Generator

Recent AI Art Generators

- 当前一些常用的开放AI绘画工具有
  - Dream by Wombo：一款2022年初发布的移动端APP，用户可以输入文字描述和手动选择绘画风格，APP会快速出图，但限制是移动端专有，且分辨率较低，不推荐
  - Disco Diffusion：免费、高清大图、艺术性高、无版权限制、可本地部署，但限制是使用需要具备一定的代码基础，且没有网页UI界面和本地应用程序，不推荐
  - Stable Diffusion：高清大图、无版权限制、**开源模型**，官方网页按图收费，但可以通过比较复杂的步骤本地部署，推荐
  - Midjourney：目前最火的AI绘画工具，高清大图、快速出图、无版权限制、**价格公道**（免费就别想了），且部署在Discord上，**有UI界面**，推荐

- 下面分别介绍Midjourney和Stable Diffusion的使用/部署方法及效果

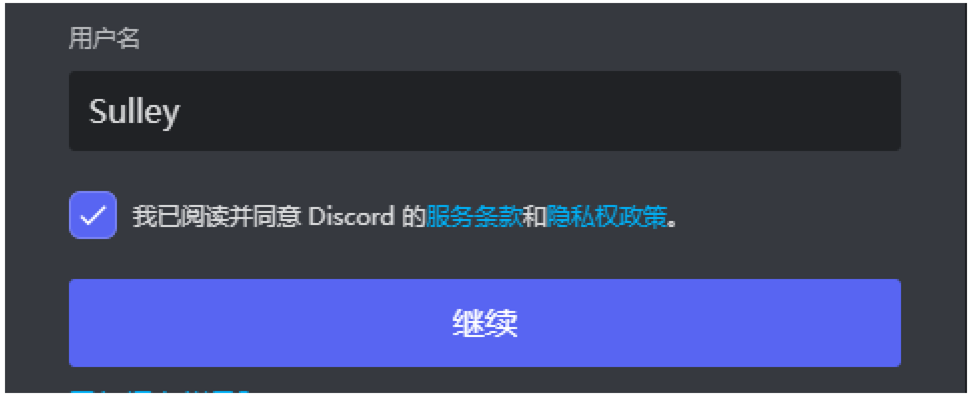
https://bytexd.com/get-started-with-disco-diffusion-to-created-ai-generated-art/
https://replicate.com/nightmareai/disco-diffusion
https://www.howtogeek.com/830179/how-to-run-stable-diffusion-on-your-pc-to-generate-ai-images/
https://www.howtogeek.com/832491/how-to-run-stable-diffusion-locally-with-a-gui-on-windows/
https://discord.com/app/invite-with-guild-onboarding/midjourney

Applications and Tools: AI Art Generator

Usage of Midjourney

- 1) 登陆官网https://www.midjourney.com/home/，点击Join the beta

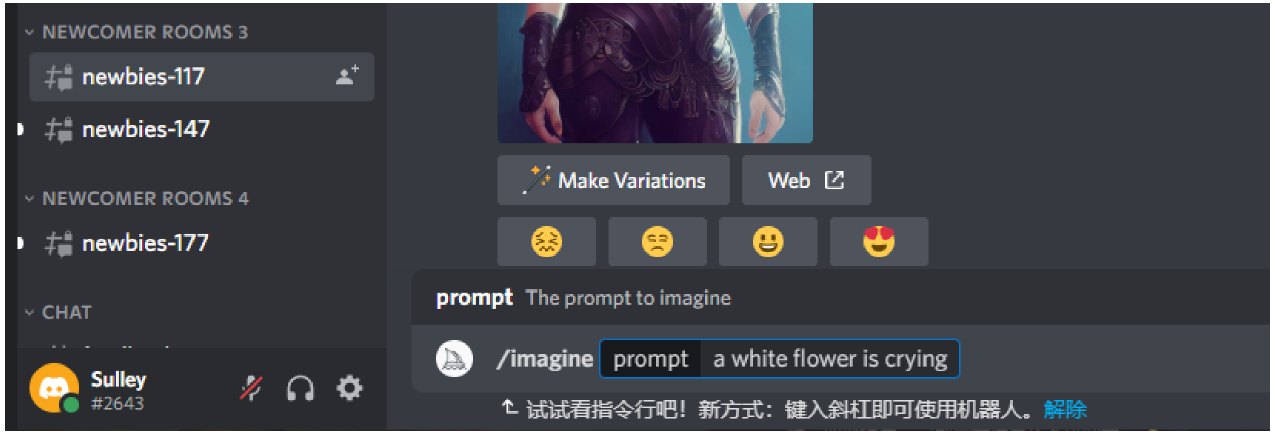
- 2）进入后输入昵称，加入Discord，如果你没有discord，可能需要根据提示注册一个，之后进入服务器

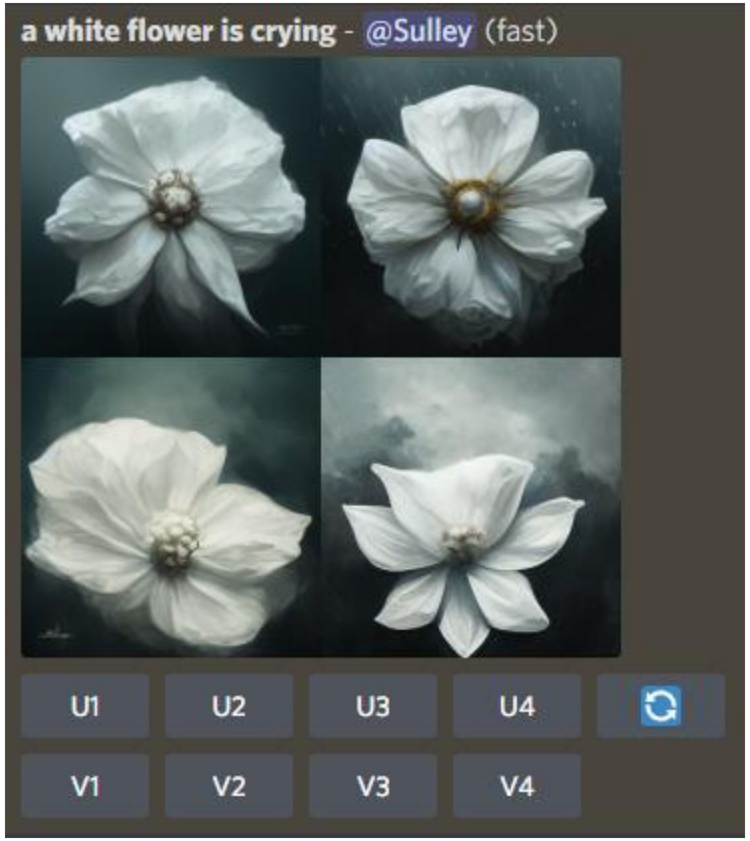


Applications and Tools: AI Art Generator

Usage of Midjourney

- 3）进入一个以#newbies开头的频道，比如#newbies-117，然后在下方的输入框中输入/imagine，此时就能在弹出来的prompt框中输入你想要生成图片的文本描述了，比如我这里输入的是a white flower is crying，稍等片刻，就能在聊天框中看到生成的4张图像了

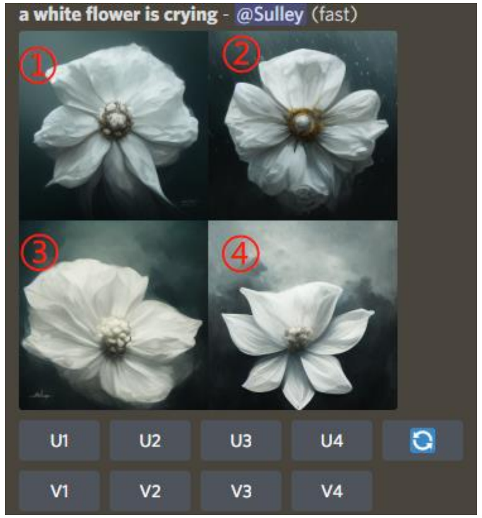


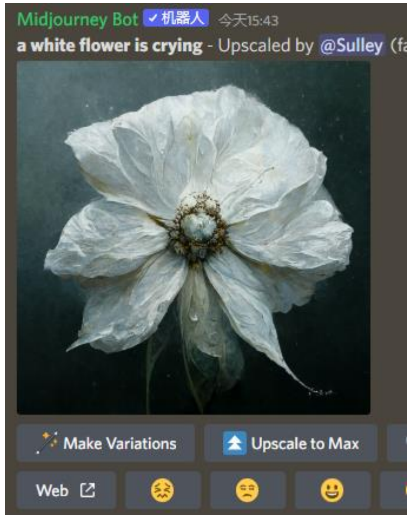


Applications and Tools: AI Art Generator

Usage of Midjourney

- 4）除了生成的4张图像外，下方还有两行按钮，分别是U1/U2/U3/U4和V1/V2/V3/V4，分别表示增大每张图的分辨率，以及为每张图重新随机生成
  - 在点击增大分辨率之后，对应大图会重新发送在频道中，下方也会随之出现几个新按钮，见字如义
- 每个新账号可以免费生成25张图，之后只能付费使用，当前有三种付费方式
  - Basic：每月10刀、200分钟Fast GPU时间、无Relax GPU时间，能生成约200张图，此后则需要按时付费
  - Standard：每月30刀、15小时Fast GPU时间、无限Relax GPU时间，能生成无限张图，只是快慢有别
  - Corporate：每年600刀（每月50刀）、120小时Fast GPU时间、无限Relax GPU时间，且生成的图片仅自己能见





	Free trial	Basic	Standard	Corporate***
Price	Free	$10 / month	$30 / month	$600 / **year**
Fast GPU time	25 min*/ lifetime	200 min*/ month	15 hrs*/ month	120 hrs*/ **year**
Relax GPU time	No	No	Unlimited	Unlimited
Metered mode**	No	Yes	Yes	Yes
Personal Bot Chat	No	**Yes**	**Yes**	**Yes**
Private visibility	No	for +$20 / month	for +$20 / month	Yes

* 1 generation job roughly takes 1 gpu minute, upscales are longer, variants are quicker
**Metered mode can be enabled to use additional fast time for $4 / gpu hour
***Corporate plan is required for employees of companies with revenues over $1Million / year

https://midjourney.gitbook.io/docs/billing

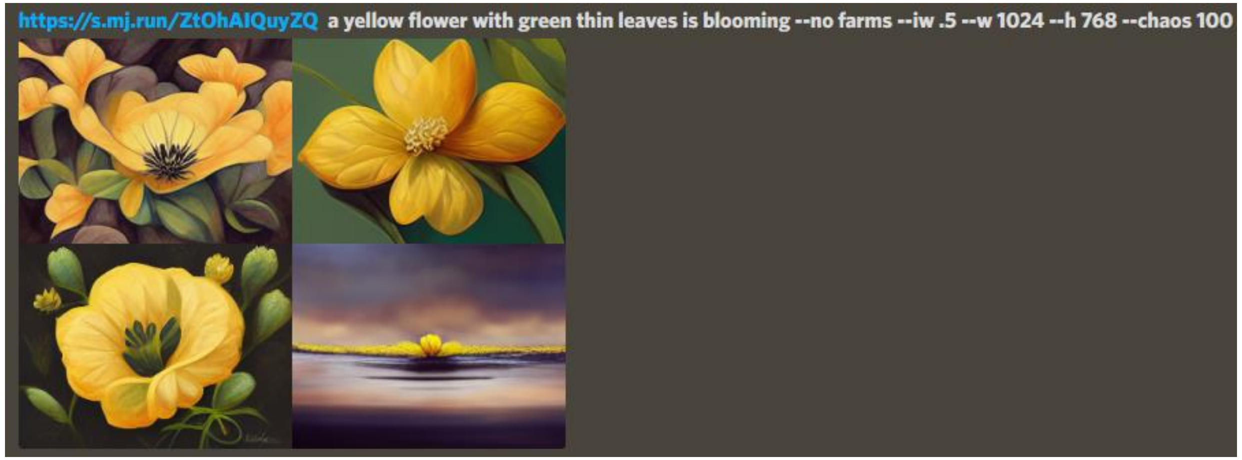
Applications and Tools: AI Art Generator

Usage of Midjourney

详细的使用方法见：https://midjourney.gitbook.io/docs/user-manual，下面进行简要总结

参数	功能
/imagine	呼出prompt，根据文本描述生成四张图片
/info	查看当前正在运行的任务
/fast(/relax)	切换为使用Fast/Relax GPU时间
/private	切换为private模式，其他人不可见你的图片
/public	切换为public模式，其他人可见你的图片
--hd	使用旧算法，适用于抽象和风景图，图片分辨率更高
--ar <n:n>	显式指定图片的宽高比，比如 --ar 16:9
--w <n>	显式指定图片的宽度，比如 --w 320
--h <n>	显式指定图片的高度，比如 --h 256
--seed <n>	显式指定种子数
--no <s>	生成的排除该关键词，比如--no plants为去掉文本中的"plants"
--iw <f>	设置prompt中的图片/文本权重比，默认0.25
--s <n>	指定生成图片的风格化程度，值越大，图片越"抽象"
--q <f>	指定图片质量，默认为1，值越大，细节越多，但耗时越长
--chaos <n>	指定图片的随机性，值越大，生成图片越多样，范围[0,100]
--fast	更快地生成图片，但质量会更低，近似于--q 0.5或--q 0.25
--stop <n>	在n%的时候停止终止生成
--uplight	在Upscale的时候用light版本，增加更少的细节，与原图更接近

- Prompt由三部分组成：
  - Image Prompt: 多个图片链接组成，所生成的图片的风格和内容会尽量接近所提供的image prompt，每张图片通常以.png或.jpg结尾，可以通过参--iw调整image prompt的权重
  - Text Prompt: 一个字符串，跟在最后一个image prompt之后
  - Parameters: 若干可选的参数，如左表所示



https://midjourney.gitbook.io/docs/imagine-parameters

Applications and Tools: AI Art Generator

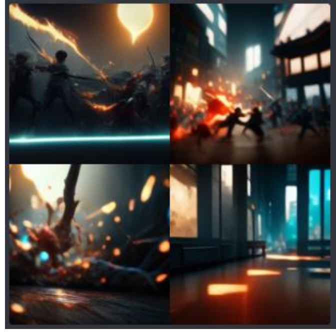
Some Nice Examples of Midjourney

medium shot of an old woman, brown eyes, beautiful, detailed eyes, cinematic epic angle, dynamic lighting, photo realistic, 8k post, cinematic colour photorealistic, high details, ultra HD, hyperrealistic, highly detailed, scene hyper realistic, very detailed, high resolution, cinematic lighting, --test --creative --upbeta --upbeta --upbeta --ar 10:16 - @Tenok (fast)

monster creepy organic body skeleton bones metal in gothic church , hyperrealistic, 8k, 4k, ultra detail, ultra realistic, vfx, epic, ray tracing, octane render, unreal engine, 3dsmax + V-Ra, award winning photograph, --ar 2:3 --upbeta --test --creative --upbeta --upbeta - Upscaled (Beta) by @ydees (fast)

https://i.mj.run/99ebbd18-280e-44fd-9d19-8a7bb1a9f4e9/0_2.png moebius，Ghibli，Alejandro Burdisio，matte painting，A diesel powered bulky exoskeleton power armor， and part of the chitinous's limbs are added to the armor，8k, detailed, intricate, realistic, dieselpunk - Upscaled by @yuhan163 (fast)

epic anime fight, cinematic epic angle, dynamic lighting, photo realistic, 8k post, cinematic colour photorealistic, high details, ultra HD, hyperrealistic, highly detailed, scene hyper realistic, very detailed, high resolution, cinematic lighting, - @Zedy Beny (fast)

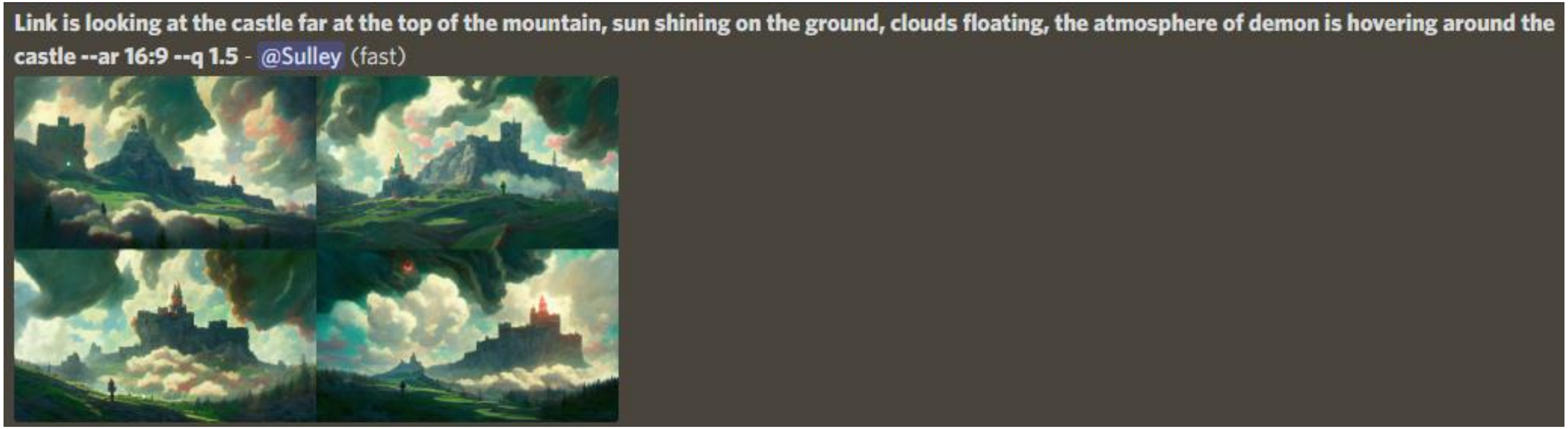


yemen in 2030 , beautiful, cinematic epic angle, dynamic lighting, photo realistic, 8k post, cinematic colour photorealistic, high details, ultra HD, hyperrealistic, highly detailed, scene hyper realistic, very detailed, high resolution, cinematic lighting, --test --creative --upbeta --upbeta --upbeta --ar 16:9 - @AhmdBdr (fast)

medium shot of an young man, brown eyes, beautiful, detailed eyes, cinematic epic angle, dynamic lighting, photo realistic, 8k post, cinematic colour photorealistic, high details, ultra HD, hyperrealistic, highly detailed, scene hyper realistic, very detailed, high resolution, cinematic lighting, - @Mayu bro (fast)

Applications and Tools: AI Art Generator

Some Nice Examples of Midjourney



所生成图片的艺术性很强!

Applications and Tools: AI Art Generator

Usage of Stable Diffusion

下面介绍Stable Diffusion的部署方法（步骤1-8）

- 1）下载Python 3.10.6：https://www.python.org/downloads/release/python-3106/，在安装的时候一定要勾选**把安装路径加入到环境变量**中!

Files

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and late
Windows embeddable package (32-bit)	Windows	
Windows embeddable package (64-bit)	Windows	
Windows help file	Windows	
Windows installer (32-bit)	Windows	
Windows installer (64-bit)	Windows	Recommended

Applications and Tools: AI Art Generator

Usage of Stable Diffusion

- 2）下载并安装Git：https://git-scm.com/download/win，都使用默认选项即可

Download for Windows

Click here to download the latest (**2.37.3**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **20 days ago**, on 2022-08-30.

Other Git for Windows downloads

Standalone Installer

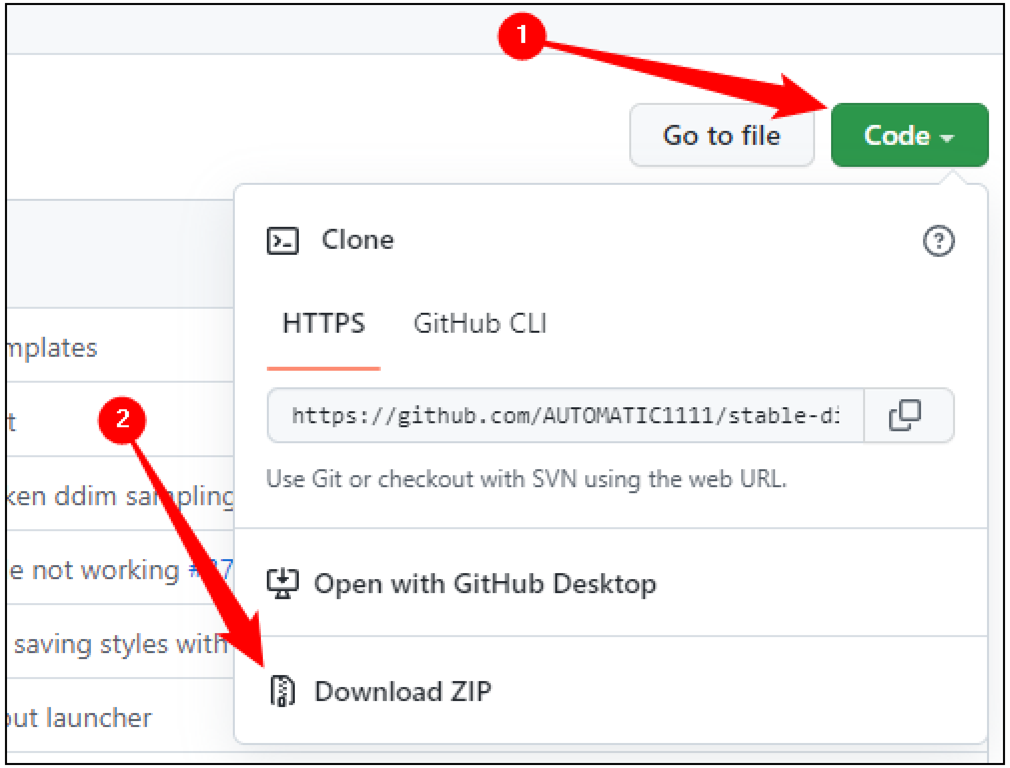
32-bit Git for Windows Setup.

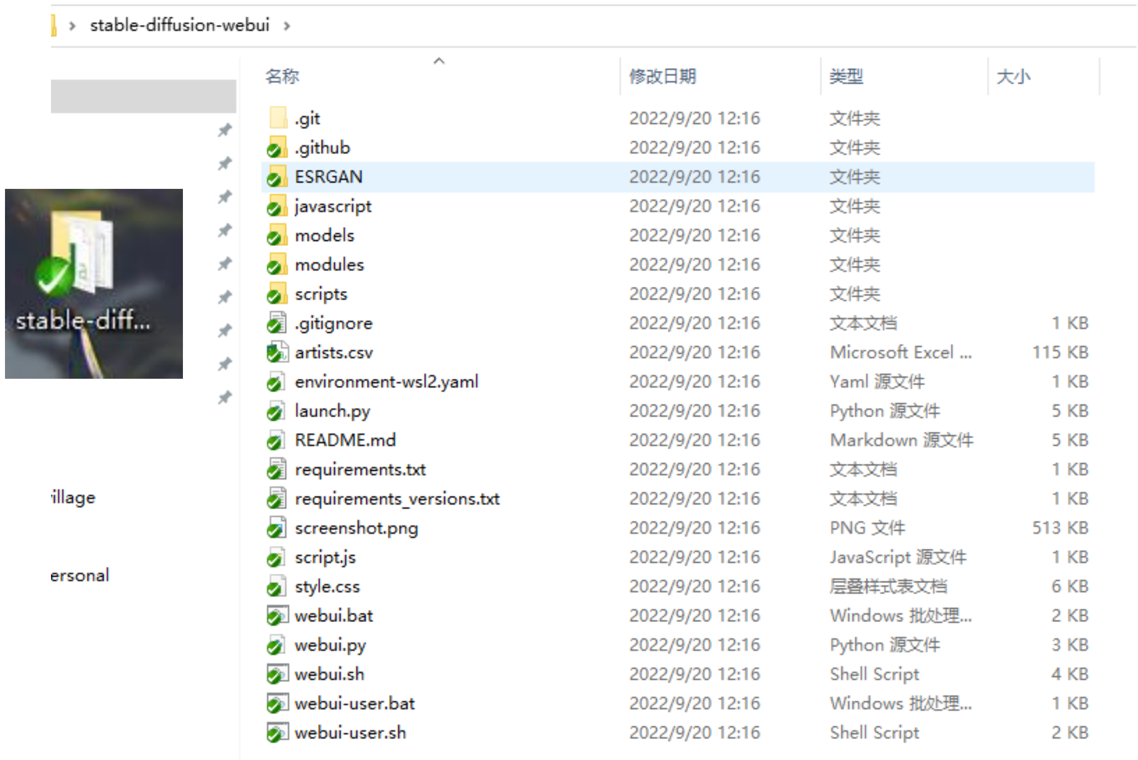
64-bit Git for Windows Setup.

Applications and Tools: AI Art Generator

Usage of Stable Diffusion

- 3）下载UI界面所需要的文件：https://github.com/AUTOMATIC1111/stable-diffusion-webui，下载完毕之后把文件夹解压到你想要的目录，比如这里我放到了桌面

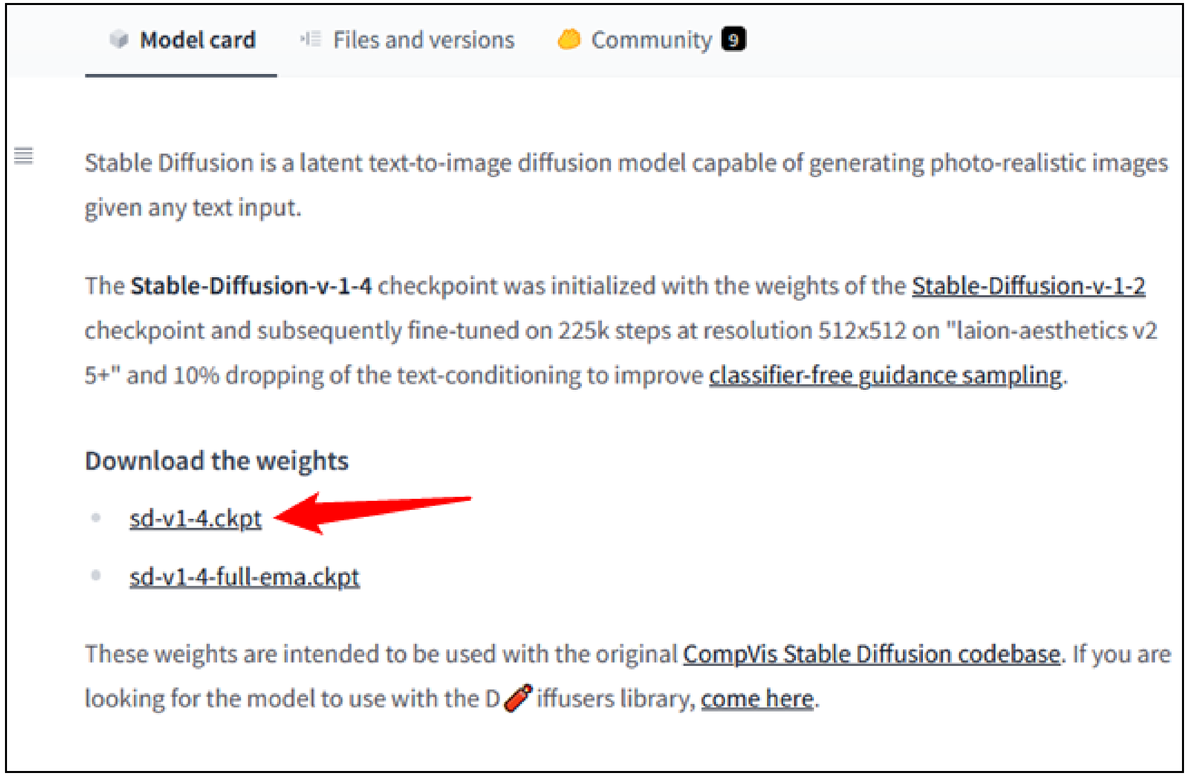




Applications and Tools: AI Art Generator

Usage of Stable Diffusion

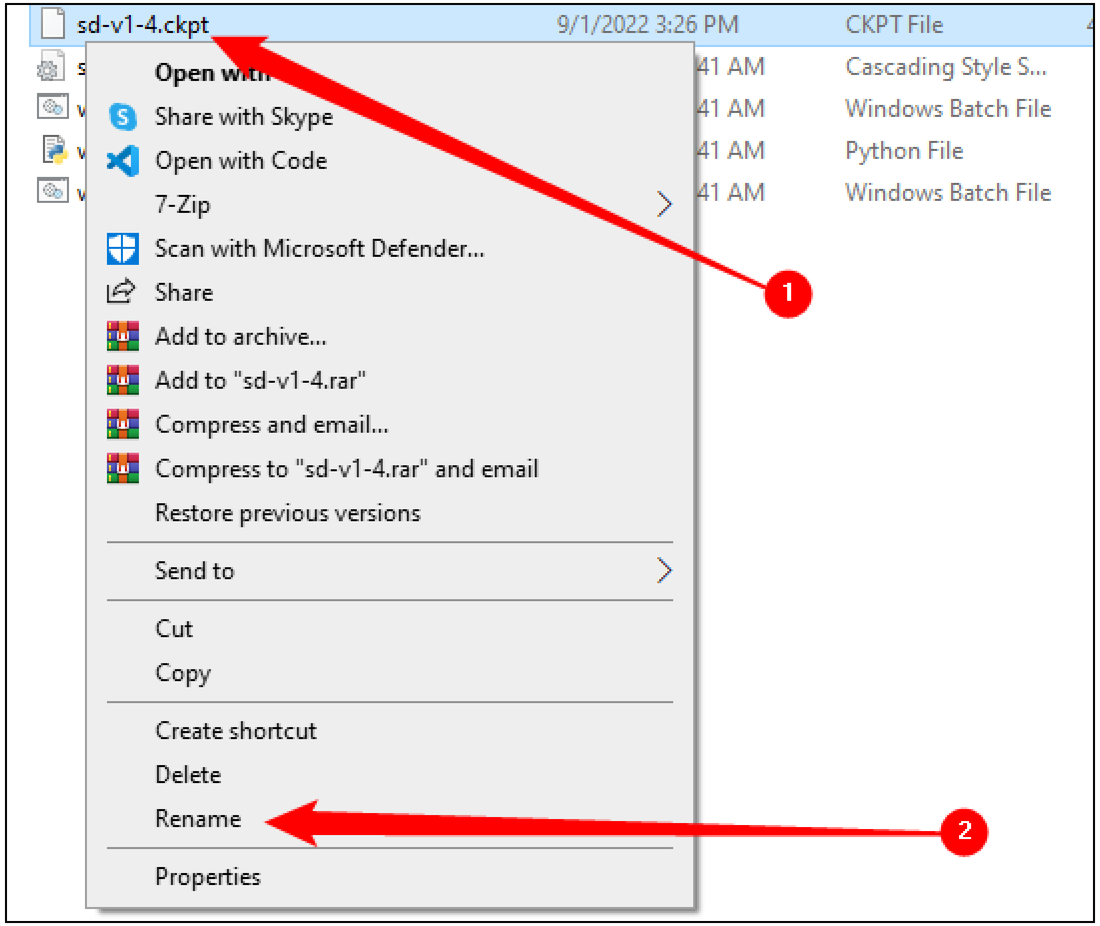
- 4）下载Stable Diffusion模型文件：https://huggingface.co/CompVis/stable-diffusion-v-1-4-original，你可能需要先注册一个HuggingFace账号。由于文件较大，所以下载需要一些时间



Applications and Tools: AI Art Generator

Usage of Stable Diffusion

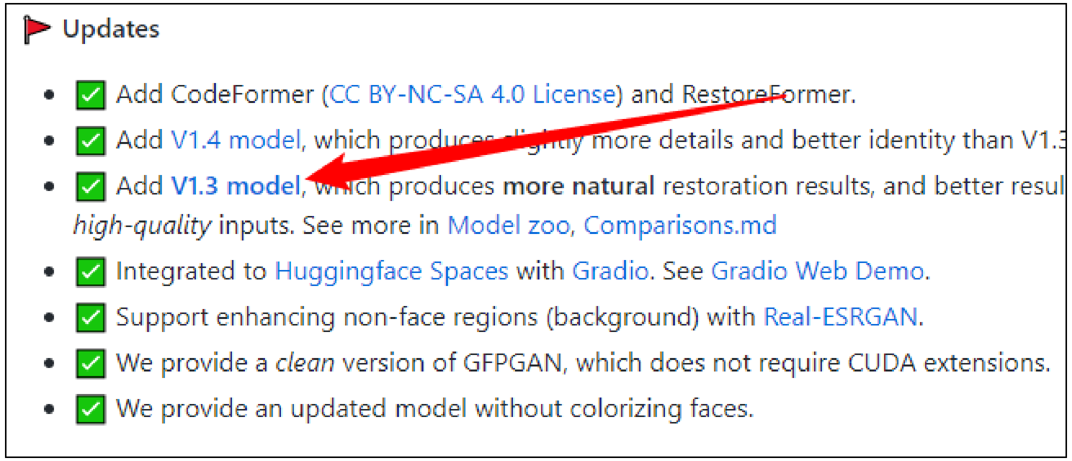
- 5）把下载完之后的文件放到之前下载的文件夹中，然后重命名为model.ckpt

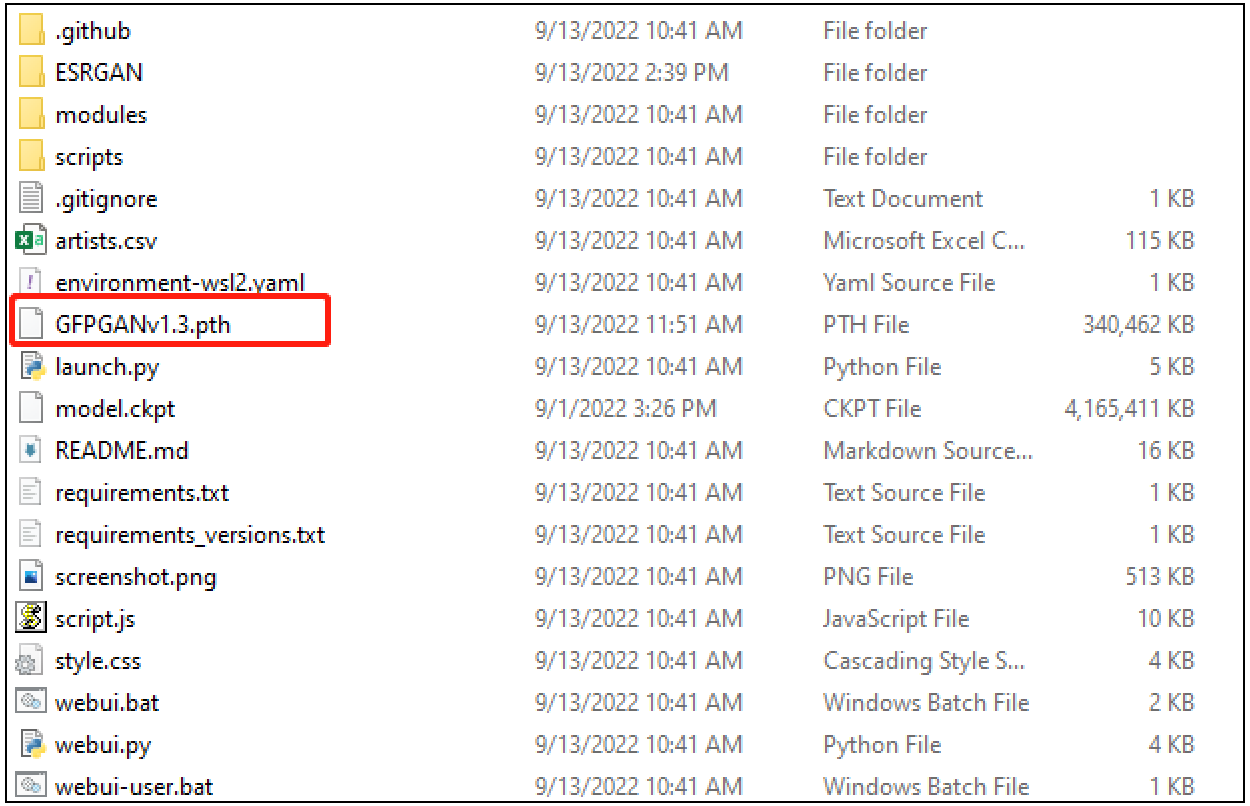


Applications and Tools: AI Art Generator

Usage of Stable Diffusion

- 6）下载GFPGAN：https://github.com/TencentARC/GFPGAN，V1.3和V1.4都可以，下载完之后仍然把文件放到之前的文件夹里，但这次不需要重命名了





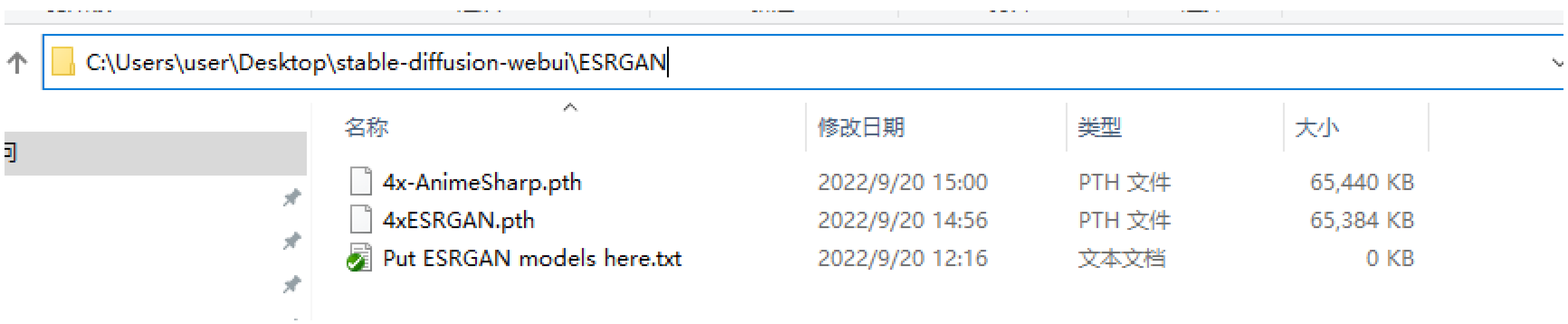
.github	9/13/2022 10:41 AM	File folder	
ESRGAN	9/13/2022 2:39 PM	File folder	
modules	9/13/2022 10:41 AM	File folder	
scripts	9/13/2022 10:41 AM	File folder	
.gitignore	9/13/2022 10:41 AM	Text Document	1 KB
artists.csv	9/13/2022 10:41 AM	Microsoft Excel C...	115 KB
environment-wsl2.yaml	9/13/2022 10:41 AM	Yaml Source File	1 KB
GFPGANv1.3.pth	9/13/2022 11:51 AM	PTH File	340,462 KB
launch.py	9/13/2022 10:41 AM	Python File	5 KB
model.ckpt	9/1/2022 3:26 PM	CKPT File	4,165,411 KB
README.md	9/13/2022 10:41 AM	Markdown Source...	16 KB
requirements.txt	9/13/2022 10:41 AM	Text Source File	1 KB
requirements_versions.txt	9/13/2022 10:41 AM	Text Source File	1 KB
screenshot.png	9/13/2022 10:41 AM	PNG File	513 KB
script.js	9/13/2022 10:41 AM	JavaScript File	10 KB
style.css	9/13/2022 10:41 AM	Cascading Style S...	4 KB
webui.bat	9/13/2022 10:41 AM	Windows Batch File	2 KB
webui.py	9/13/2022 10:41 AM	Python File	4 KB
webui-user.bat	9/13/2022 10:41 AM	Windows Batch File	1 KB

Applications and Tools: AI Art Generator

Usage of Stable Diffusion

- 7）下载ESRGAN模型： https://upscale.wiki/wiki/Model_Database，你可以根据下载多个ESRGAN模型。下载好之后，在之前的文件夹中新建一个子文件夹（若没有），命名为ESRGAN，把你下载的所有ESRGAN模型文件（.pth）文件放到这个子文件夹里即可，比如我这里下载了两个模型

注：ESRGAN模型是一类超分辨率模型，旨在将低分辨率的图片还原为高分辨率图片，上述每个模型都有各自适用的领域



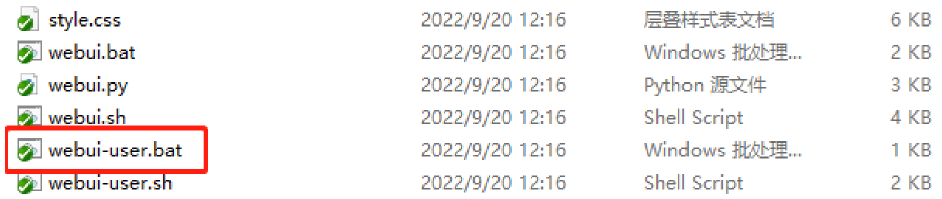
Applications and Tools: AI Art Generator

Usage of Stable Diffusion

- 8）双击启动文件夹中的文件webui-user.bat，之后会打开一个命令行并自动运行，你只需要等待运行结束即可（时间较长）。当结束的时候，命令行上会显示：

  Running on local URL: http://127.0.0.1:7860
  To create a public link, set `share=True` in `launch()`

  如果出现RPC failed报错，则重新打开即可

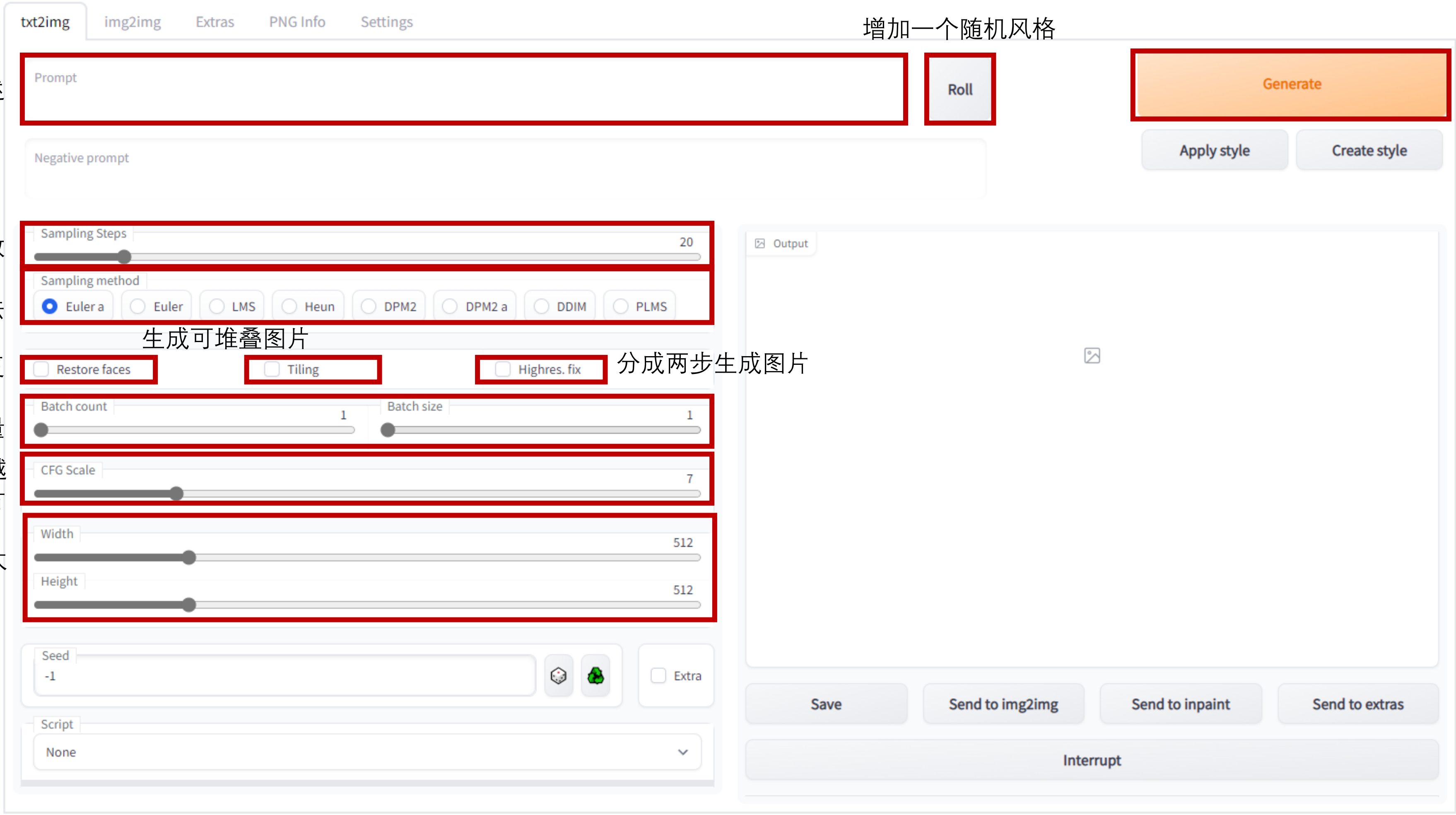


- 最后，打开浏览器，输入地址http://127.0.0.1:7860即可进入UI界面

Applications and Tools: AI Art Generator

Usage of Stable Diffusion

- 下面简要介绍一些重要参数



文本描述

生成步数

生成方法

人脸修复

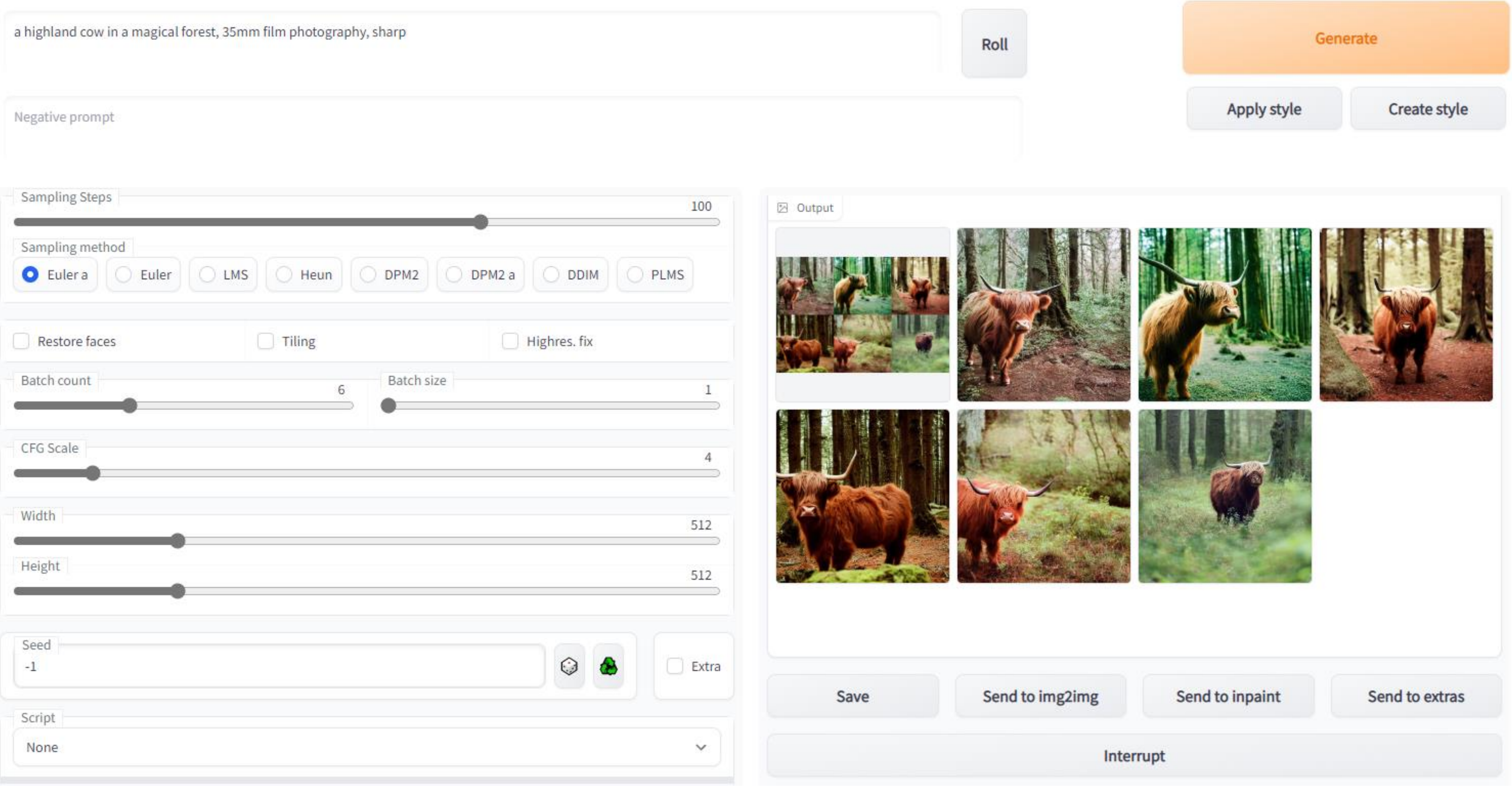
生成数量

值越小，生成越具创造力的图片

分辨率，不要太大，显存会炸

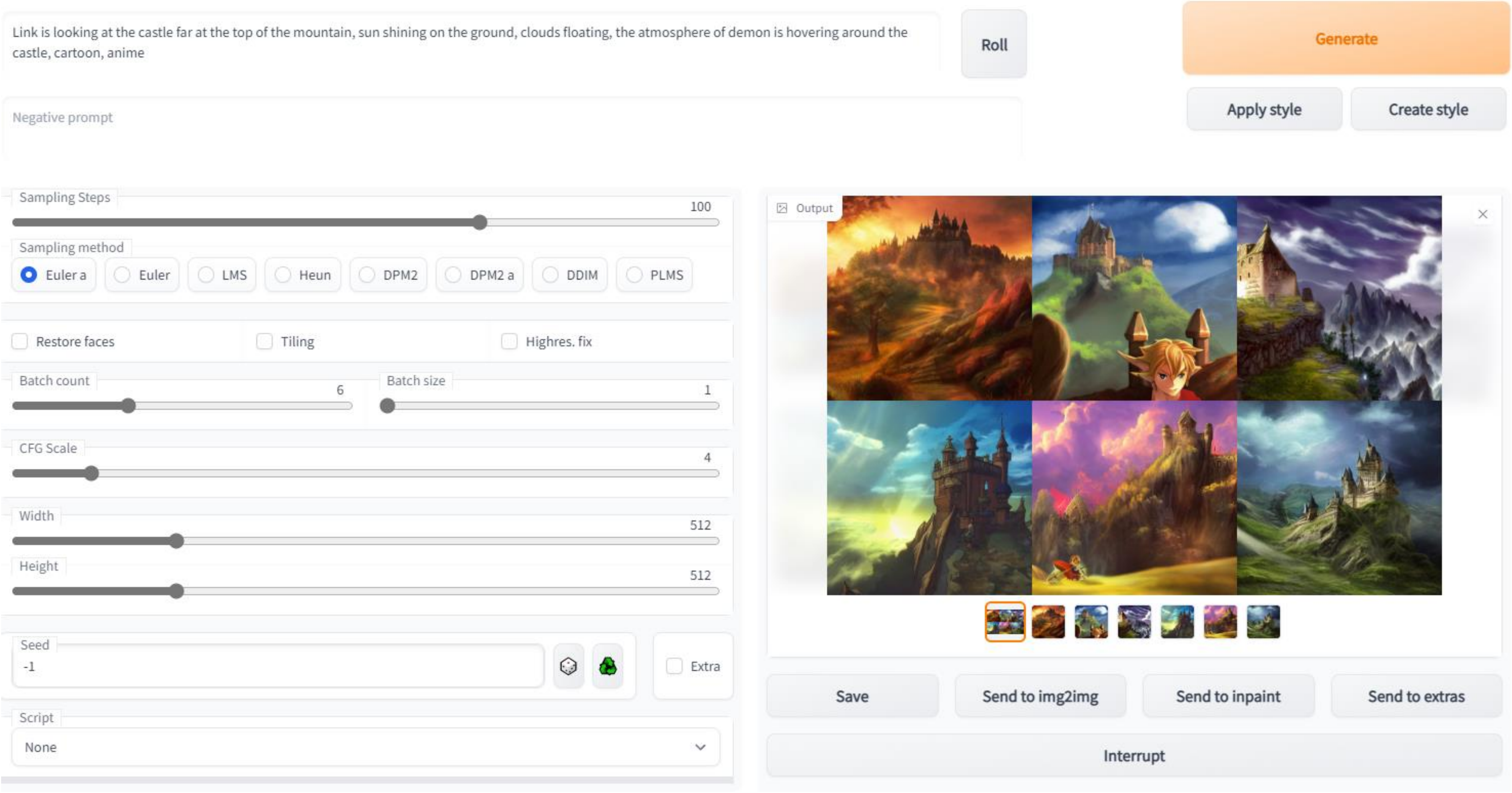
Applications and Tools: AI Art Generator

Examples of Stable Diffusion



Applications and Tools: AI Art Generator

Examples of Stable Diffusion



生成的图片更加写实，同时创造性更低

Applications and Tools: AI Art Generator

Comparison

	Stable Diffusion	Midjourney
图像风格	偏写实	偏艺术
图像质量	较低	较高
生成速度	快	较快
是否付费	免费	付费
部署方式	可本地部署	需在指定网页端/桌面端操作
推荐指数	☆☆☆	☆☆☆☆

References

- From GAN to WGAN, https://lilianweng.github.io/posts/2017-08-20-gan/
- From Autoencoder to Beta-VAE, https://lilianweng.github.io/posts/2018-08-12-vae/
- Flow-based Deep Generative Models, https://lilianweng.github.io/posts/2018-10-13-flow-models/
- What are Diffusion Models? https://lilianweng.github.io/posts/2021-07-11-diffusion-models/#forward-diffusion-process
- Diffusion Models: A Comprehensive Survey of Methods and Applications, https://arxiv.org/abs/2209.00796
- 扩散模型与其在文本生成图像领域的应用， https://zhuanlan.zhihu.com/p/546311167