Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

扩散模型的应用和发展

加速采样的极致演变

硕士研究生 万韵伟

2023年04月16日

内容提要

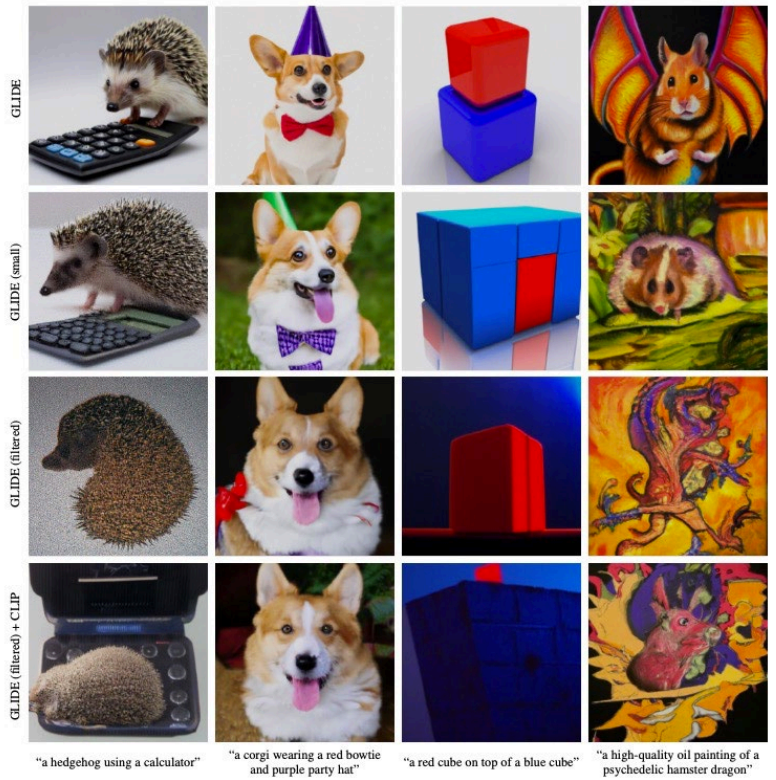
- 背景简介
  - DDPM 回顾
  - 加速采样
- 算法原理
  - DDIM
  - PNDM
- 应用
  - Stable Diffusion
- 总结
- 前沿发展
- 参考文献

内容提要

- 预期收获
  - 1.了解扩散模型在数据生成中的实际应用
  - 2.理解扩散模型的加速采样原理
  - 3.学会运用加速采样进行生成优化
  - 4.了解扩散模型在实际应用中的优化与发展

背景简介

- 扩散模型 —— 文本生成图像
  - 2021 年 1 月，OpenAI 公布首个文本生成图像模型 DALL-E
  - 2021 年 12 月底，OpenAI 提出 GLIDE，能够生成比 DALL-E 更复杂、更丰富的图像



背景简介

- 扩散模型 —— 文本生成图像
  - 2022 年 4 月，OpenAI 提出 DALL-E 2，能够生成真实或者艺术的图像
  - 2022 年 5 月，Google 发表 Imagen，在写实性上击败 DALL-E 2

A chromeplated cat sculpture placed on a Persian rug.

Android Mascot made from bamboo.

A transparent sculpture of a duck made out of glass.

A raccoon wearing cowboy hat and black leather jacket is behind the backyard window. Rain droplets on the window.

背景简介 DDPM

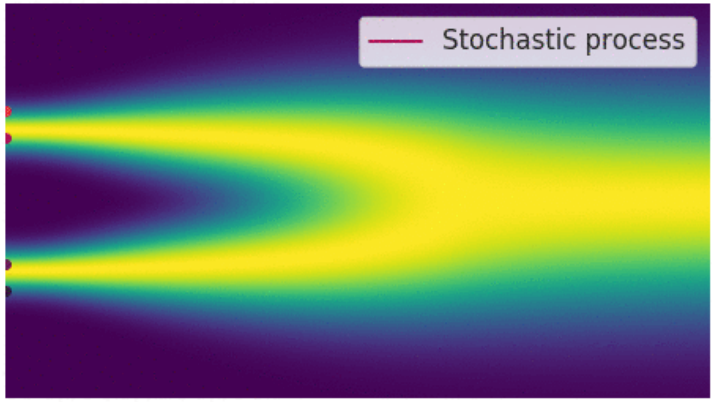
- 扩散过程
  - 从真实数据中取样数据点 $\mathbf{x}_0 \sim q(\mathbf{x})$，连续添加 T 次高斯噪声（方差为$\{\beta_1, \ldots, \beta_T\}$），产生噪声样本 $\mathbf{x}_1, \ldots, \mathbf{x}_T$

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

  - 数据样本 $\mathbf{x}_0$ 随 t 增大而逐渐失去可辨别特征。当 $T \to \infty$ 时，$\mathbf{x}_T$ 为各向同性的高斯分布

- 可以在任意阶段 t 采样获得 $\mathbf{x}_t$

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



背景简介 DDPM

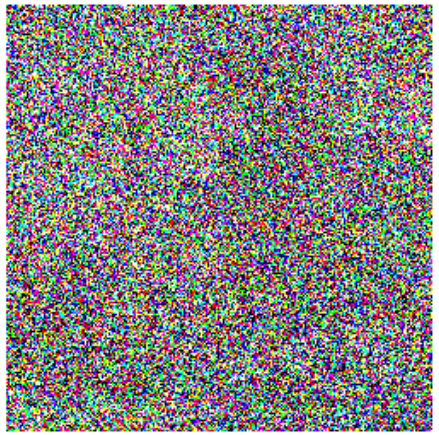
- 逆扩散过程
  - 当 β_t 足够小的时候， $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ 可以近似为高斯过程

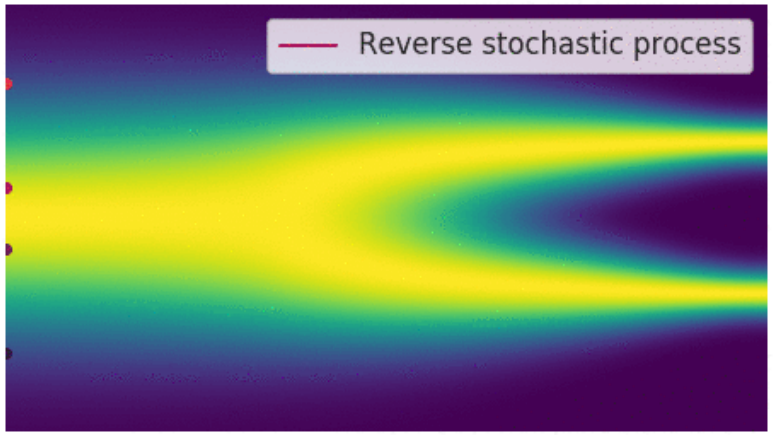
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = N\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right)$$

  - 使用深度模型预测噪声 ϵ_θ，可从逆向分布中进行采样，获取前一步的数据
  - 循环采样所有时间步，得到合成图像

Algorithm 2 Sampling

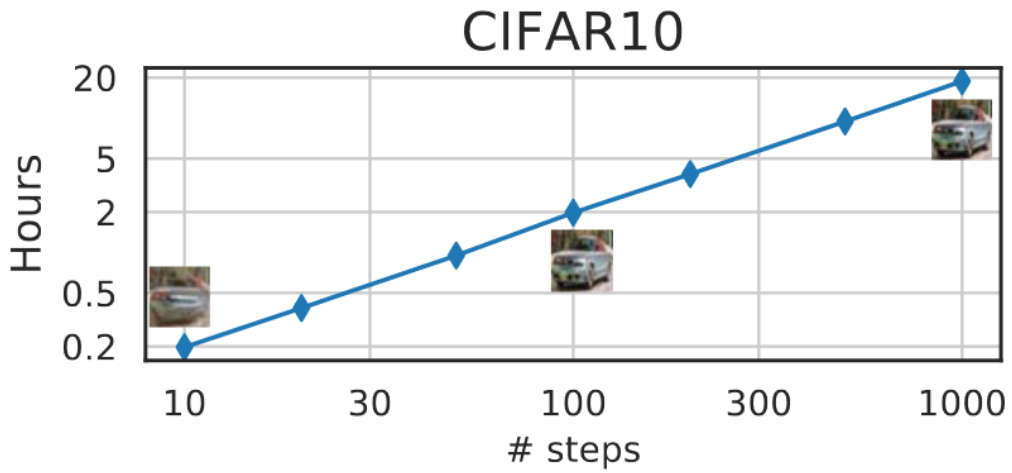
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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

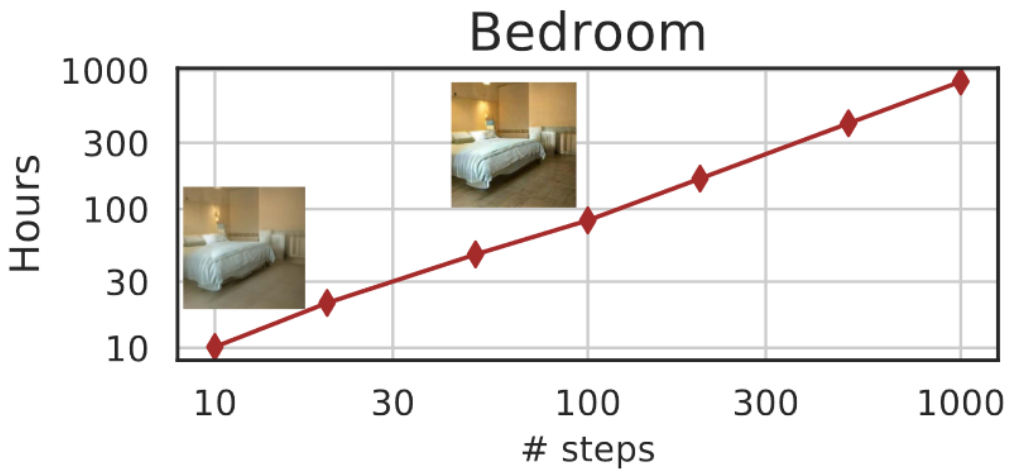




背景简介

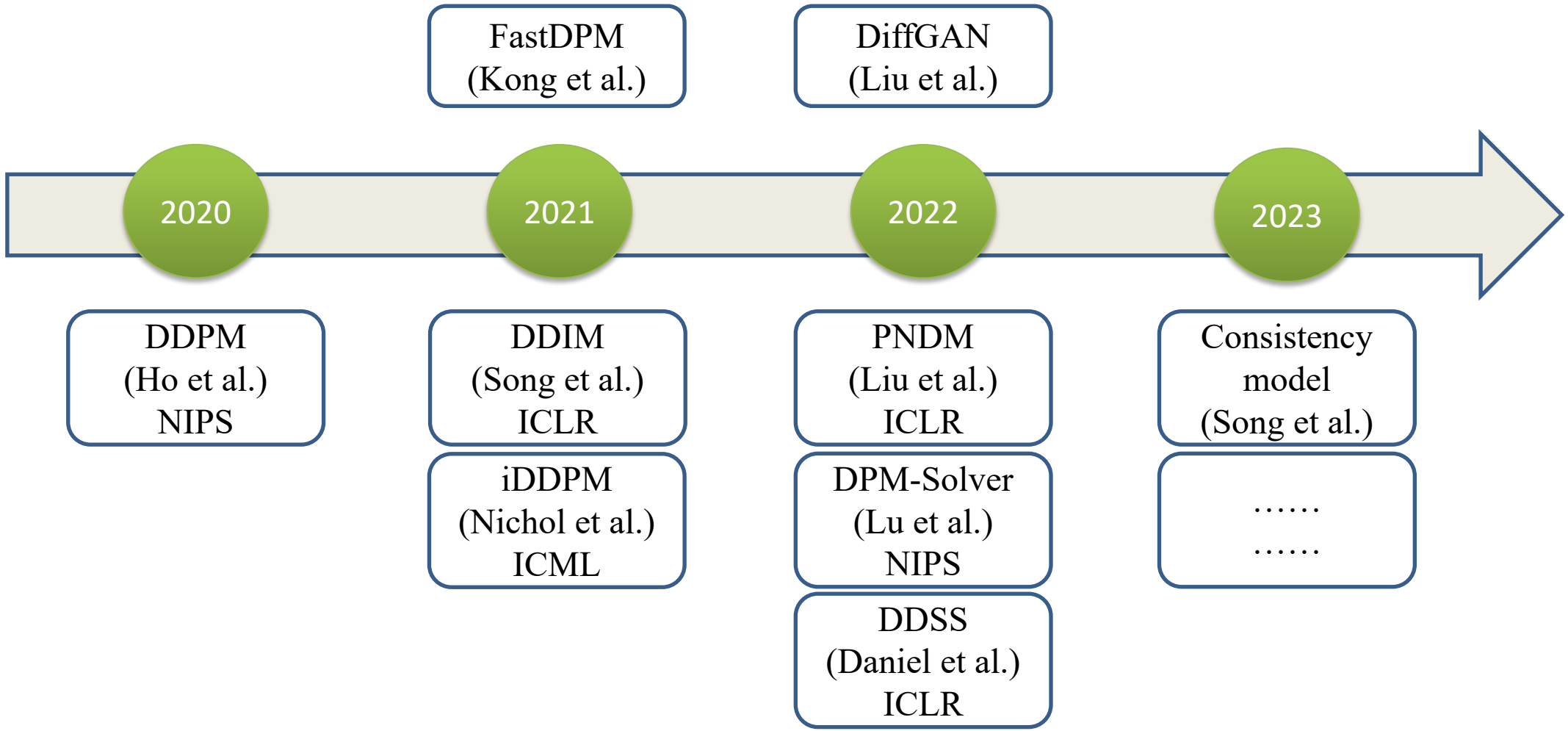
- 采样缓慢?
  - 为了达到较好的数据生成效果，通常需要几千次的加噪和采样，成本较高，在实际应用的过程中困难重重
    - DDPM 对大小为 32 x 32 的 50 KB 图像采样，CPU 需要约 20 小时，GPU 2080 Ti 需要一分钟





研究历史 加速采样

- 扩散模型 加速采样



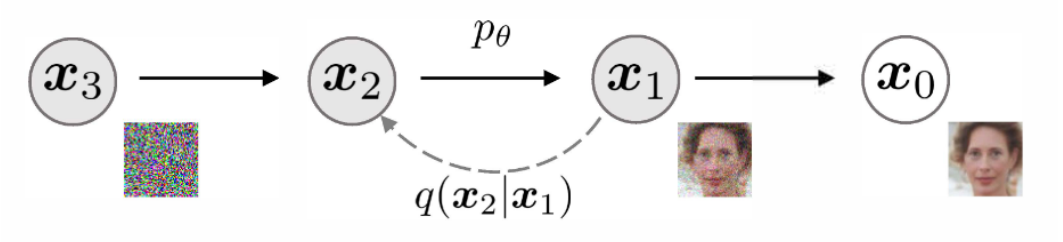
加速采样

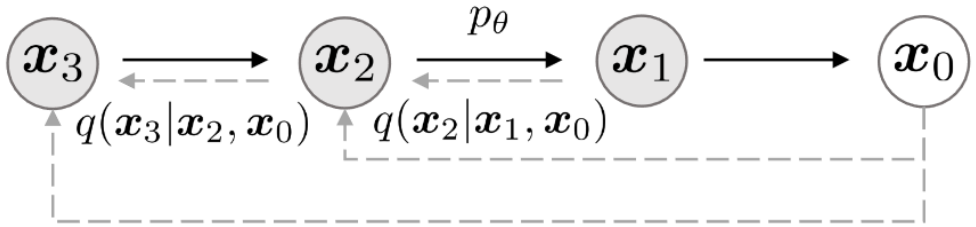
T	目标	生成高质量合成图像
I	输入	原始输入图像
P	处理	1.对输入图像进行加噪 2.解噪器根据条件预测噪声 3.根据预测噪声快速采样，生成高质量图片
O	输出	高质量合成图片

P	问题	扩散模型采样步长大，采样时间长
C	条件	输入图片的大小一致
D	难点	如何在保证数据生成质量的情况下，提升扩散模型的采样速度
L	水平	ICLR 2021 2022

算法原理 DDIM

- DDIM (ICLR 2021)：无随机项迭代法
  - 通过定义子序列跳步骤采样
  - 将图像的采样过程定义为非马尔可夫链：





  - 定义采样过程：

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\boldsymbol{\epsilon}_\theta + \sigma_t \mathbf{z}_t, \qquad \sigma_t = 0$$

  - 当 $\sigma_t = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}} \times \sqrt{1-\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}$ 时，为 DDPM 采样过程

算法原理 DDIM

- 从 DDPM 到 DDIM：去除随机项
- Diffusion SDE：从扩散模型到随机微分方程
  - 扩散随机微分方程：扩散模型的迭代可以关于 t 取极限后转化为随机微分方程

$$\mathrm{d}(\mathbf{x}_t) = \left(f(t)\mathbf{x}_t + \frac{g^2(t)}{\sigma_t}\epsilon_\theta(\mathbf{x}_t, t) \right)\mathrm{d}t + \sigma_t \mathbf{z}_t$$

- Diffusion ODE：从扩散模型到常微分方程
  - 扩散常微分方程：

$$\frac{\mathrm{d}(\mathbf{x}_t)}{\mathrm{d}t} = f(t)\mathbf{x}_t + \frac{g^2(t)}{\sigma_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$$

  - DDPM 是对 SDE 的一阶离散化，DDIM 是对 ODE 的一阶离散化
  - 将采样优化过程转化为常微分方程的快速求解

评价指标

- FID（Fréchet Inception Distance）
  - FID 是从原始图像的计算机视觉特征的统计方面，来衡量两组图像的相似度
  - 使用 Inception 网络，去除最后的全连接层，得到 2048 维特征
  - 计算两个特征的 Fréchet 距离

$$FID(x, g) = \left|\left|\mu_x - \mu_g\right|\right|_2^2 + Tr\left(\Sigma_x + \Sigma_g - 2\left(\Sigma_x\Sigma_g\right)^{0.5}\right)$$

  - 较低的分数与较高质量的图像有很高的相关性
- FID变体
  - 适应时序数据：Context-FID
  - 将 Inception 转换为 MTS 编码器

实验结果 DDIM

- 加速采样效果
  - η 为随机系数， $\eta = 1$ 为 DDPM， $\eta = 0$ 为 DDIM
  - 降低随机系数，可以提升低采样步数下的生成效果
  - 可以进行有损采样，在牺牲一定生成图像质量的情况下提升采样速度

		CIFAR10 (32 × 32)					CelebA (64 × 64)				
	S	10	20	50	100	1000	10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
	$\hat{\sigma}$	367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

实验结果 DDIM

- DDIM (ICLR 2021)
  - 代码实现

```
# predict noise eps
eps = self._predict_eps_from_xstart(x, t, out["pred_xstart"])
alpha_bar = _extract_into_tensor(self.alpha, t, x.shape)
alpha_bar_prev = _extract_into_tensor(self.alpha_prev, t, x.shape)
sigma = (
    eta
    * torch.sqrt((1 - alpha_bar_prev) / (1 - alpha_bar))
    * torch.sqrt(1 - alpha_bar / alpha_bar_prev)
) # ddpm: eta=1, ddim: eta=0
# Equation of sample x_{t-1}
noise = torch.randn_like(x)
mean_pred = (
    out["pred_xstart"] * torch.sqrt(alpha_bar_prev)
    + torch.sqrt(1 - alpha_bar_prev - sigma ** 2) * eps
)
nonzero_mask = (
    (t != 0).float().view(-1, *([1] * (len(x.shape) - 1)))
)
sample = mean_pred + nonzero_mask * sigma * noise
return {"sample": sample,"pred xstart": out["pred_xstart"]}
```

计算随机项

计算 x_{t-1}

```
# ddim
num_steps = space_timesteps(self.num_steps, 'ddim8')
last_alpha_cumprod = 1.0
new_betas = []
for i, alpha_cumprod in enumerate(self.alpha):
    if i in num_steps:
        new_betas.append(1 - alpha_cumprod / last_alpha_cumprod)
        last_alpha_cumprod = alpha_cumprod
self.beta = np.asarray(new_betas)
```

根据压缩步长，重新生成噪声链

$\{\beta_1, \beta_2, \ldots, \beta_t\}$

```
Energy
2023-04-14 20:46:48.915 | INFO     | __main__:<module>:95 - Use existed model t
2023-04-14 20:46:49.207 | INFO     | __main__:<module>:109 - Generate samples o
2023-04-14 20:46:49.207 | INFO     | utils:evaluate:108 - Evaluation: generate
  0%|                                                | 6/19711 [00:13<12:43:51,
```

```
Energy
2023-04-14 20:45:15.302 | INFO     | __main__:<module>:95 - Use existed model
2023-04-14 20:45:15.586 | INFO     | __main__:<module>:109 - Generate samples
2023-04-14 20:45:15.587 | INFO     | utils:evaluate:108 - Evaluation: generate
  0%|                                                | 32/19711 [00:13<1:47:35,
```

算法发展 加速采样

- 优势
  - 将随机微分方程的采样过程转化为常微分方程，提供了加速采样的可能性
  - 可以在保持一定生成质量的情况下，减少采样步数，加速采样
- 局限性
  - 一阶离散化方法收敛速度缓慢，为提升采样速度，往往通过对 Diffusion ODE 使用高阶求解器进行加速
  - 传统高阶求解器将原本离散的扩散模型迭代先连续化为微分方程，再反过来通过数值方法离散化求解
  - 如何权衡生成质量和速度性能
- 发展：PNDM (ICLR 2022)，DPM-Solver (NIPS 2022)

算法原理 PNDM

- PNDM (ICLR 2022)
  - 融合高阶数值方法和 DDIM 的优点
  - 在原始 DDIM 的方法上，对其中的 ϵ 项使用 线性多步法 PLMS 做四阶修正
  - 无法自启动，使用 Runge-Kutta 方法 (PRK) 计算前三步采样
  - e_t 当前时间步的预测值，$x_{t+\delta}$ 下一步采样值

$$
\begin{cases}
e_t = \epsilon_\theta(x_t, t) \\
e'_t = \frac{1}{24}(55e_t - 59e_{t-\delta} + 37e_{t-2\delta} - 9e_{t-3\delta}) \\
x_{t+\delta} = \phi(x_t, e'_t, t, t+\delta).
\end{cases}
$$

$$
\begin{cases}
e_t^1 = \epsilon_\theta(x_t, t) \\
x_t^1 = \phi(x_t, e_t^1, t, t+\frac{\delta}{2}) \\
e_t^2 = \epsilon_\theta(x_t^1, t+\frac{\delta}{2}) \\
x_t^2 = \phi(x_t, e_t^2, t, t+\frac{\delta}{2}) \\
e_t^3 = \epsilon_\theta(x_t^2, t+\frac{\delta}{2}) \\
x_t^3 = \phi(x_t, e_t^3, t, t+\delta) \\
e_t^4 = \epsilon_\theta(x_t^4, t+\delta) \\
e'_t = \frac{1}{6}(e_t^1 + 2e_t^2 + 2e_t^3 + e_t^4) \\
x_{t-\delta} = \phi(x_t, e'_t, t, t+\delta).
\end{cases}
$$

算法原理 PNDM

- PNDM (ICLR 2022)
  - 融合高阶数值方法和 DDIM 的优点
  - 在原始 DDIM 的方法上，对其中的 ϵ 项使用 线性多步法 PLMS 做四阶修正
  - 无法自启动，使用 Runge-Kutta 方法 (PRK) 计算前三步采样
  - e_t 当前时间步的预测值，$x_{t+\delta}$ 下一步采样值

Algorithm 1 DDIMs

$x_T \sim \mathcal{N}(0, I)$
for $t = T-1, \cdots, 1, 0$ **do**
  $x_t = \phi(x_{t+1}, \epsilon_\theta(x_{t+1}, t+1), t+1, t)$
end for
return x_0

Algorithm 2 PNDMs

$x_T \sim \mathcal{N}(0, I)$
for $t = T-1, T-2, T-3$ **do**
  $x_t, e_t = \text{PRK}(x_{t+1}, t+1, t)$
end for
for $t = T-4, \cdots, 1, 0$ **do**
  $x_t, e_t = \text{PLMS}(x_{t+1}, \{e_p\}_{p>t}, t+1, t)$
end for
return x_0

实验结果 PNDM

- 图像生成效果
  - S-PNDM：二阶修正
  - F-PNDM：四阶修正
  - 可以在保证生成图像效果的情况下，将采样步长降低到 50

- time：每步采样的平均计算成本，单位：秒
  - RTX 3090

dataset	FID \ step / model	10	20	50	100	250	1000	time
Cifar10	DDIM	13.4	6.84	4.67	4.16		4.04	
	PF		13.8	3.89	3.69	3.71	3.72	
Cifar10 (linear)	DDIM*	18.5	10.9	6.99	5.52	4.52	4.00	0.337
	FON	13.1	7.41	5.26	4.65	4.12	3.71	0.390
	S-PNDM	11.6	7.56	5.18	4.34	3.91	3.80	0.344
	F-PNDM	7.03	5.00	3.95	3.72	**3.60**	3.70	0.391
Cifar10 (cosine)	DDIM	14.5	8.79	5.86	4.92	4.30	3.69	0.505
	S-PNDM	8.64	5.77	4.46	3.94	3.71	3.38	0.517
	F-PNDM	7.05	4.61	3.68	3.53	3.49	**3.26**	0.595
CelebA	DDIM	17.3	13.7	9.17	6.53		3.51	
CelebA (linear)	DDIM*	16.9	13.4	8.95	6.36	4.44	3.41	1.237
	FON	16.0	11.6	8.13	6.70	5.14	4.17	1.431
	S-PNDM	12.2	9.45	5.69	4.03	3.19	2.99	1.258
	F-PNDM	7.71	5.51	3.34	2.81	**2.71**	2.86	1.433

实验结果 PNDM

- 代码实现
  - Hugging face 提供接口：diffusers

Scheduler	Paper
ddim	Denoising Diffusion Implicit Models
ddpm	Denoising Diffusion Probabilistic Models
singlestep_dpm_solver	Singlestep DPM-Solver
multistep_dpm_solver	Multistep DPM-Solver
heun	Heun scheduler inspired by Karras et. al paper
dpm_discrete	DPM Discrete Scheduler inspired by Karras et. al paper
dpm_discrete_ancestral	DPM Discrete Scheduler with ancestral sampling inspired by Karras et. al paper
stochastic_karras_ve	Variance exploding, stochastic sampling from Karras et. al
lms_discrete	Linear multistep scheduler for discrete beta schedules
pndm	Pseudo numerical methods for diffusion models (PNDM)

```python
import torch
from diffusers import PNDMScheduler

scheduler = PNDMScheduler()
scheduler.set_timesteps(50)
# input the image
image = torch.randn([24, 6])

for t in scheduler.timesteps:
    # calculate the predict noise
    model_output = torch.randn([24, 6])
    image = scheduler.step(model_output, t, image).prev_sample

print(image)
```

设置采样步长

循环采样

直接调用 PNDM 进行加速采样

Stable Diffusion

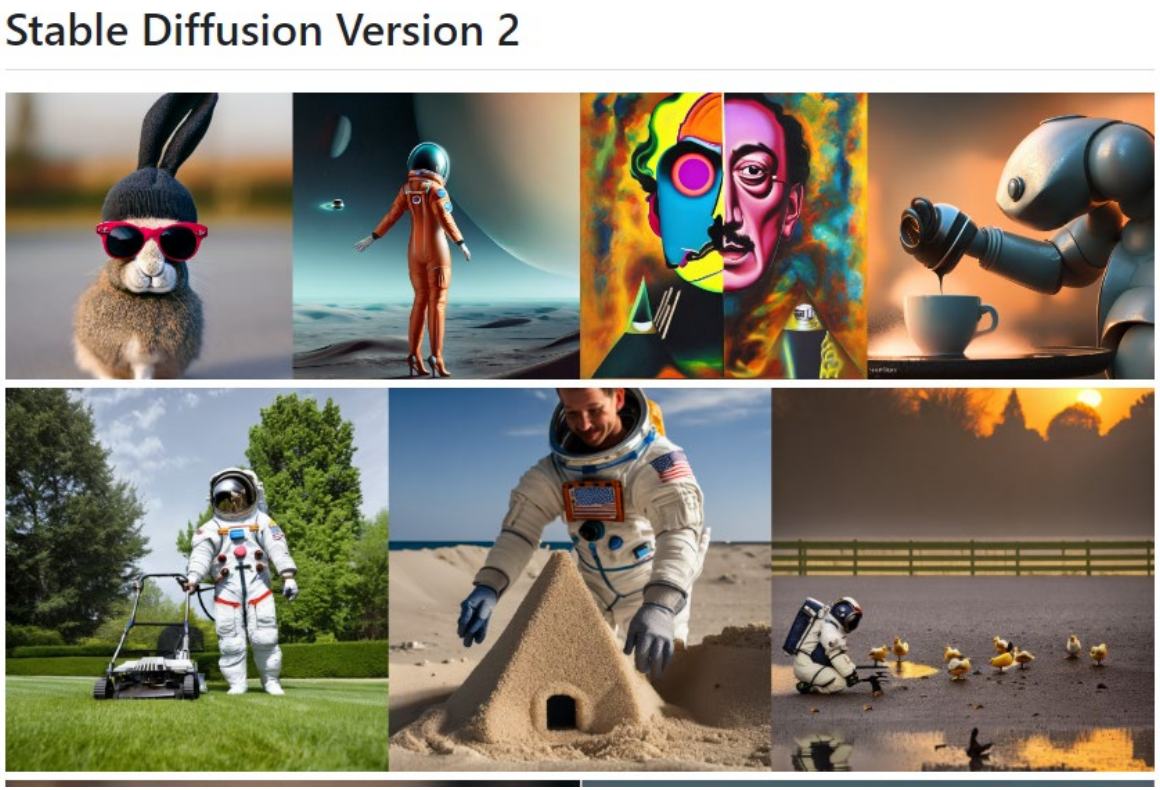
Stable Diffusion

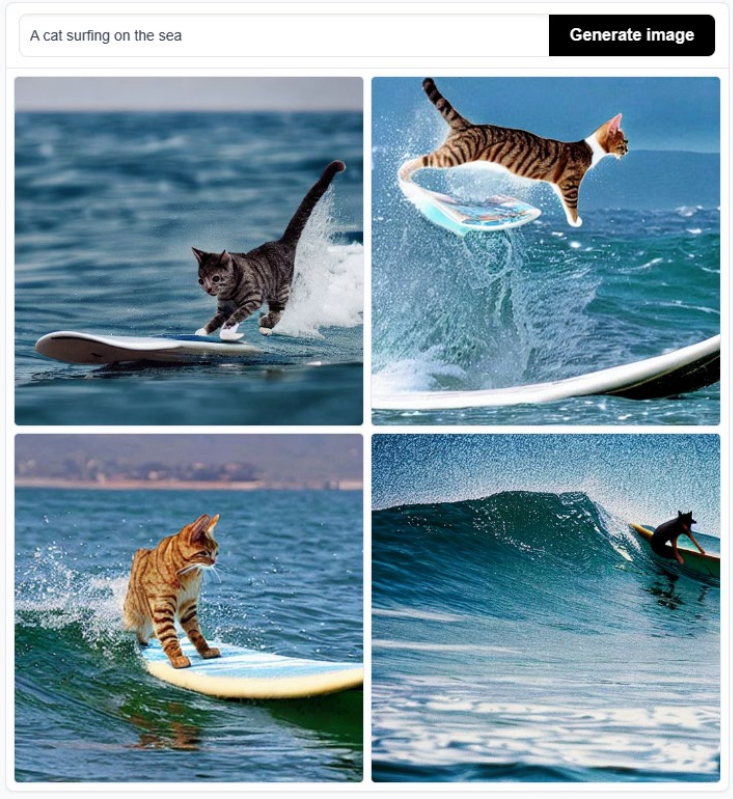
T	目标	根据输入文本生成高质量图片（Text-to-Image）
I	输入	提示文本，输入图像
P	处理	1.使用自编码器将图像编码到隐空间，并进行加噪 2.使用文本编码器对提示文本编码 3.解噪器根据文本编码预测噪声 4.根据预测噪声快速采样，生成高质量图片
O	输出	高质量合成图片

P	问题	合成大型图片存在大量的算力开销
C	条件	输入图片的大小一致
D	难点	如何在保证数据生成质量的情况下，减小模型的资源开销
L	水平	CVPR 2022

Stable Diffusion

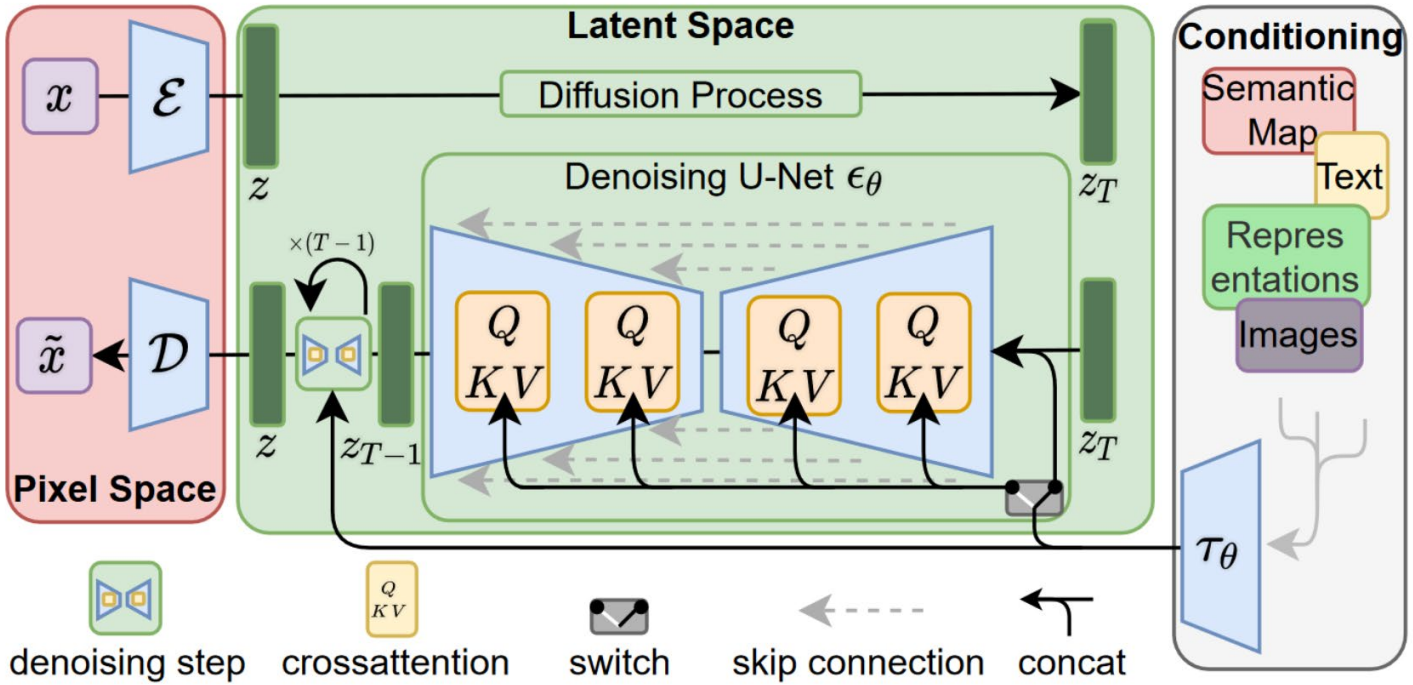
- Stable Diffusion
  - StabilityAI 公司于 2022 年开源的 AI 图像生成算法
  - 在 Stable Diffusion Online 官网上可以体验文本生成图像





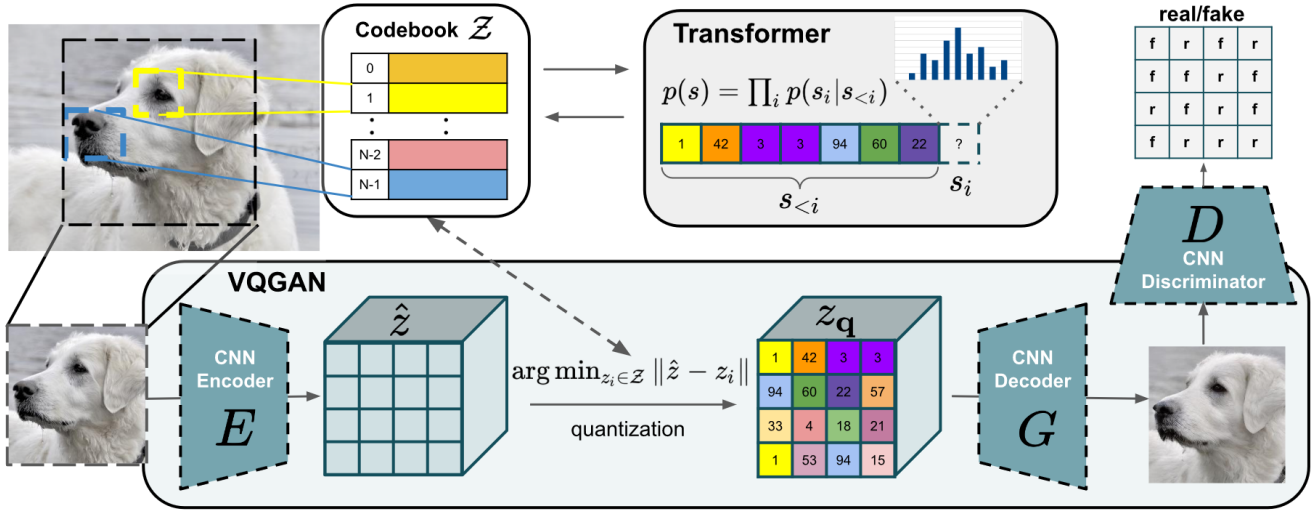
算法原理 Stable Diffusion

- Stable Diffusion
  - 使用自编码器对原始图像进行压缩
  - 使用潜在扩散模型对潜空间进行采样生成图像的隐式表达
  - 条件信息：融合不同模态的信息作为条件输入



算法原理 Stable Diffusion

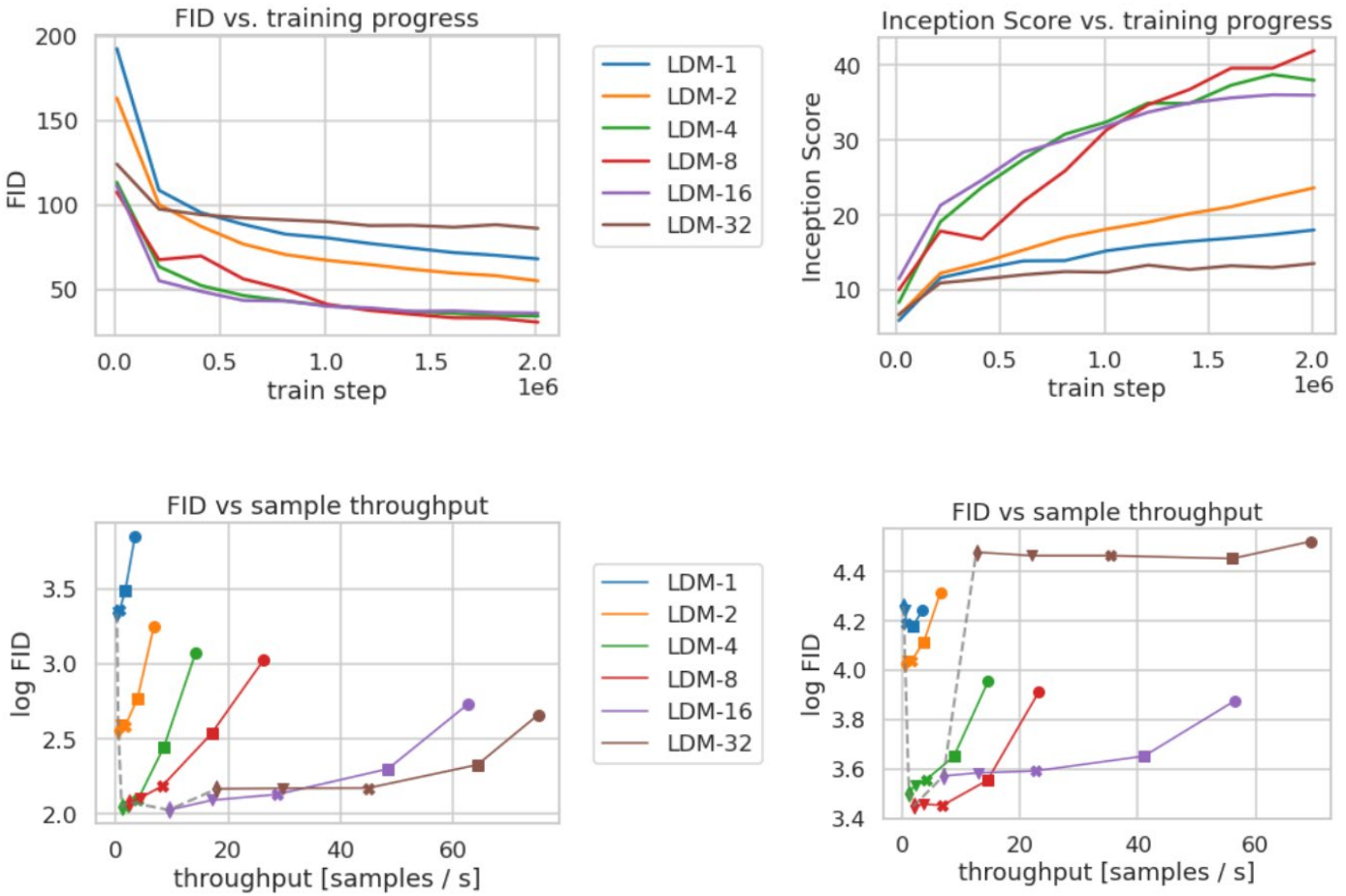
- 图片感知压缩
  - 对高分辨率图片（高维），需要高维的潜空间训练
  - 使用自编码模型对原图片进行处理，忽略图片中的高频信息
    - 添加两种正则化项：KL-reg，VQ-reg。将潜空间拟合到正态分布，避免高方差的潜空间，防止搜索空间过大
    - 使用基于 patch 的判别器，辅助提升图片压缩效果



算法原理 Stable Diffusion

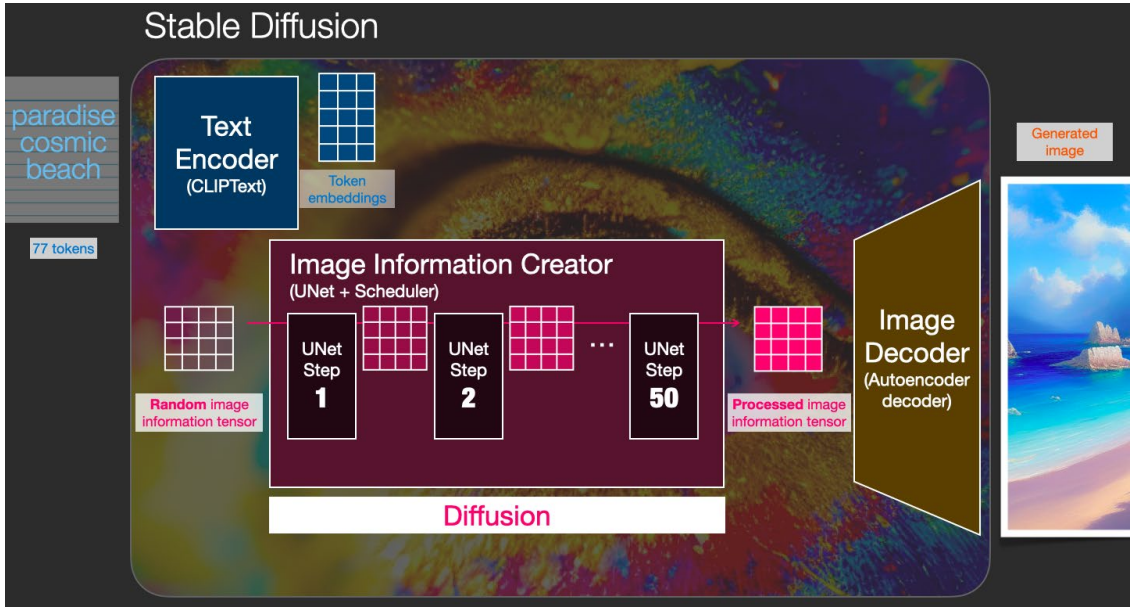
- 感知压缩权衡
  - 下采样因子 $f = \frac{H}{h} = \frac{W}{w}$，如果 f 越大，信息压缩越严重，丢失信息更多
  - $f = \{4 \sim 16\}$ 可以较好平衡压缩效率和视觉感知效果

- 推荐 LDM-4，LDM-8



算法原理 Stable Diffusion

- LDM 潜在扩散模型
  - 模型训练
    - 前向扩散：对图像的隐式表达进行前向扩散过程
    - 逆向解噪：UNetModel 接收图像的隐式表达以及提示文本编码，在训练时以文本编码作为条件，使用注意力机制学习文本和图像的匹配关系
  - 采样生成
    - 三种高速采样模式：
      - PNDM（默认）
      - DDIM
      - K-LMS
    - 采样 1 张/秒



实验效果

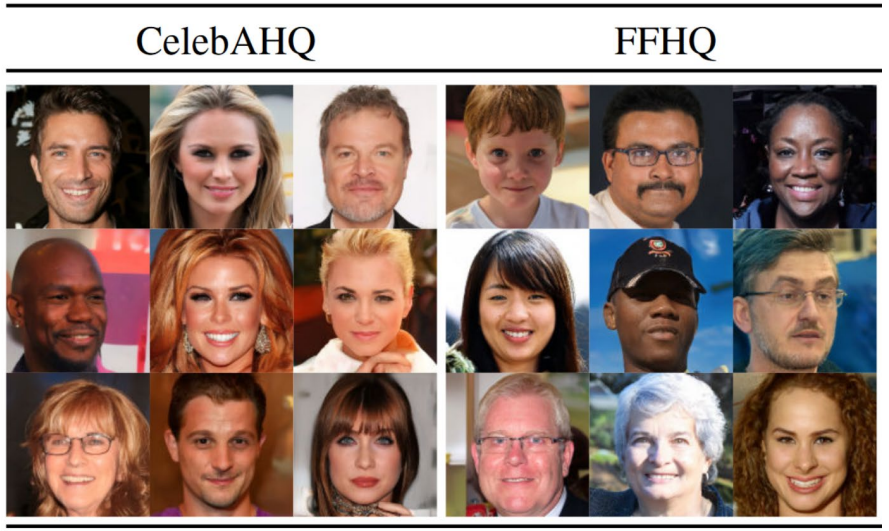
- LDM 生成效果

CelebA-HQ 256 × 256				FFHQ 256 × 256			
Method	FID ↓	Prec. ↑	Recall ↑	**Method**	FID ↓	Prec. ↑	Recall ↑
DC-VAE [61]	15.8	-	-	ImageBART [21]	9.57	-	-
VQGAN+T. [23] (k=400)	10.2	-	-	U-Net GAN (+aug) [75]	10.9 (7.6)	-	-
PGGAN [38]	8.0	-	-	UDM [42]	5.54	-	-
LSGM [90]	7.22	-	-	StyleGAN [40]	4.16	0.71	0.46
UDM [42]	7.16	-	-	ProjectedGAN [74]	**3.08**	0.65	0.46
LDM-4 (ours, 500-s†)	**5.11**	0.72	0.49	*LDM-4* (ours, 200-s)	4.98	**0.73**	**0.50**

LSUN-Churches 256 × 256				LSUN-Bedrooms 256 × 256			
Method	FID ↓	Prec. ↑	Recall ↑	**Method**	FID ↓	Prec. ↑	Recall ↑
DDPM [29]	7.89	-	-	ImageBART [21]	5.51	-	-
ImageBART [21]	7.32	-	-	DDPM [29]	4.9	-	-
PGGAN [38]	6.42	-	-	UDM [42]	4.57	-	-
StyleGAN [40]	4.21	-	-	StyleGAN [40]	2.35	0.59	0.48
StyleGAN2 [41]	3.86	-	-	ADM [15]	1.90	**0.66**	**0.51**
ProjectedGAN [74]	**1.59**	0.61	0.44	ProjectedGAN [74]	**1.52**	0.61	0.34
*LDM-8** (ours, 200-s)	4.02	**0.64**	**0.52**	*LDM-4* (ours, 200-s)	2.95	**0.66**	0.48

Text-Conditional Image Synthesis

	DALL-E† [64]	CogView† [17]	Lafite† [105]	*LDM-KL-8*	*LDM-KL-8-G**
FID ↓	27.50	27.10	26.94	23.35	**12.61**
IS ↑	17.90	18.20	26.02	19.93±0.35	**26.62**±0.38



总结

应用总结

- 算法的应用领域
  - 广泛应用于 DDPM 类型的所有算法，兼容 DDPM 的训练过程
  - 模型无关的架构特性，使得算法的可扩展性和应用前景十分广泛
- 优势
  - 相比于 DDPM，有效地缓解了采样步数大，时间过长的问题
  - 加速采样的过程有明确的数学原理，解释性强
- 劣势
  - 削减采样步数会降低生成样本的质量，需要权衡生成质量和采样速度

前沿发展

- Consistency models
  - 支持 one-step 生成，同时允许 few-step 采样，以权衡计算量和样本质量
  - 支持零样本（zero-shot）数据编辑，例如图像修复、着色和超分辨率，而无需针对这些任务进行具体训练
  - 3.5 秒生成 64 张 256×256 的图片，平均 18 张 / 秒

参考文献

[1] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2020, pp. 6840–6851.

[2] J. Song, C. Meng, and S. Ermon, “Denoising Diffusion Implicit Models,” presented at the International Conference on Learning Representations, Apr. 2023.

[3] L. Liu, Y. Ren, Z. Lin, and Z. Zhao, “Pseudo Numerical Methods for Diffusion Models on Manifolds,” presented at the International Conference on Learning Representations, Jan. 2022.

[4] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps,” presented at the Advances in Neural Information Processing Systems, Oct. 2022.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-Resolution Image Synthesis With Latent Diffusion Models,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10684–10695.

[6] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, “Consistency Models.” arXiv, Mar. 02, 2023. doi: 10.48550/arXiv.2303.01469.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！