

基于AIGC的3D生成模型基础与最新发展

张继耀 PB20000204 少年班学院

2023 年 8 月 20 日

摘要

我们在本文中对AIGC的3D生成方向最新进展进行综述性研究。我们首先介绍了AIGC的发展历程以及背后的基础技术,例如Diffusion model[18]、GAN[15]、NeRF[29]等。其次,我们重点介绍了一下Dreamfusion[28]以及Prolific Dreamer[31]这两个前沿的工作。

目录

1 介绍	2
1.1 引言	2
1.2 发展历程	3
2 已有技术	4
2.1 GAN	4
2.1.1 整体框架	4
2.1.2 训练过程	5
2.1.3 算法流程	6
2.1.4 存在的问题	6
2.2 Diffusion Model	7
2.3 DDPM	7
2.3.1 正向过程	7
2.3.2 逆向过程	8
2.4 SGM	11
2.4.1 分数匹配	11
2.4.2 朗之万动力学	13
2.5 3D Generation	14
2.6 NeRF	14
2.6.1 场景表示	15
2.6.2 体积渲染	15
2.6.3 优化	16

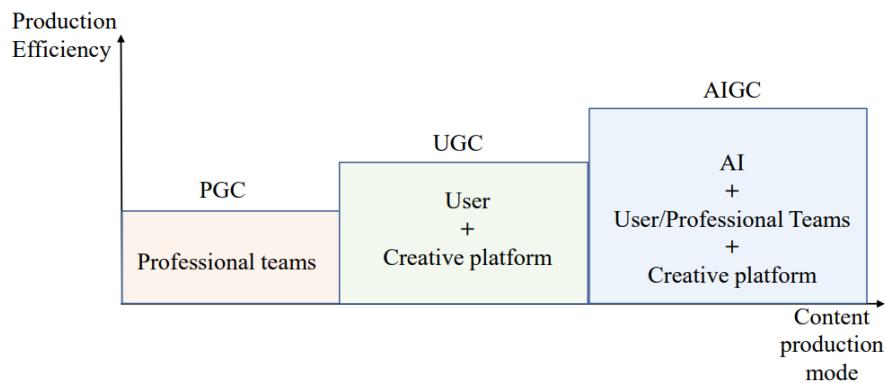
3 3D生成代表解决方案	17
3.1 分类	17
3.2 DreamFusion	18
3.2.1 算法流程	18
3.2.2 SDS	19
3.3 ProlificDreamer	20
3.3.1 VSD	21
3.3.2 算法流程	22
4 总结与展望	22
4.1 AIGC的优点与不足	22
4.2 AIGC的发展前景	23

1 介绍

1.1 引言

AIGC(AI-generated content)凭借ChatGPT等强大的工具逐渐走进人们的生活,代表着AI的一个新时代即将到来。目前,AIGC已经被证明是一个十分高效、有前景的生成工具。了解AIGC的发展历程以及它背后的原理、优点与不足,以便在未来的生活中更好的利用它,对我们每一个人来说还是十分必要的。

与传统的人工智能主要目标是分析已有的数据不同,AIGC更注重于利用AI来生成新的内容。现有的内容创作领域已经历了三次变革,从Web1.0时代的PGC、Web2.0时代的UGC,再到如今Web3.0时代的AIGC。如下图所示:



在PGC模式下,内容是由专业的团队来生成的(见[1],[2])。PGC模式的优点在于生成的绝大多数内容都是高质量的,但往往生产周期过长,难以达到需求的数量。

在UGC模式下,用户可以利用许多工具来自主生成内容(见[3],[4])。UGC的优势在于这些创造性的工具可以降低门槛和成本,提高用户的创作热情。它的劣势就是难以保证内容的质量,因为创作者的水平参差不一。

而AIGC可以在数量和质量方面克服PGC和UGC的缺点,它被人们期待可以在未来成为主要内容生成方式。在AIGC模式中, AI技术利用专业知识来提高生成内容的质量,同时也节约了时间。许多

企业已经计划利用AIGC产品来自动完成生产任务,大大节约了时间与成本。总体上来说,AIGC可以被归类成文本、图像和视频生成方面:

- 文本生成:结构化写作,创意写作和对话写作是其主要的子领域 ([5],[6],[7])。结构化写作主要产生文本基于特定场景的内容,例如新闻。然而,创意写作涉及到生成具有更高程度开放性的文本,这需要个性化以及创造能力。创意写作非常适合营销文案、社交媒体和博客。对话写作主要用于聊天机器人,通过文本。这些机器人往往是用来回答问题的。
- 图像生成:利用AIGC,用户可以以他们的图片为基础更改和添加新的元素,([8],[9],[10])。这使得用户更容易和更有效地编辑图像,而不需要高级技能或知识。此外, AIGC可以独立生成满足特定要求的图像。AIGC的另一个令人兴奋的应用是从2D图像创建3D模型[11]。
- 视频生成:AIGC在视频剪辑中得到了应用,例如制作预告片和宣传片([12],[13])。工作流类似于图像生成,其中每个在帧级处理视频的帧,然后进行AI算法被用来检测视频片段。AIGC的先进的功能越来越受欢迎,它可能继续革新视频内容的创作方式和销售。

1.2 发展历程

AIGC的历史可以追溯到上个世纪50年代,总体上来说,AIGC的发展历程大致可以分为三个阶段,如下图所示:

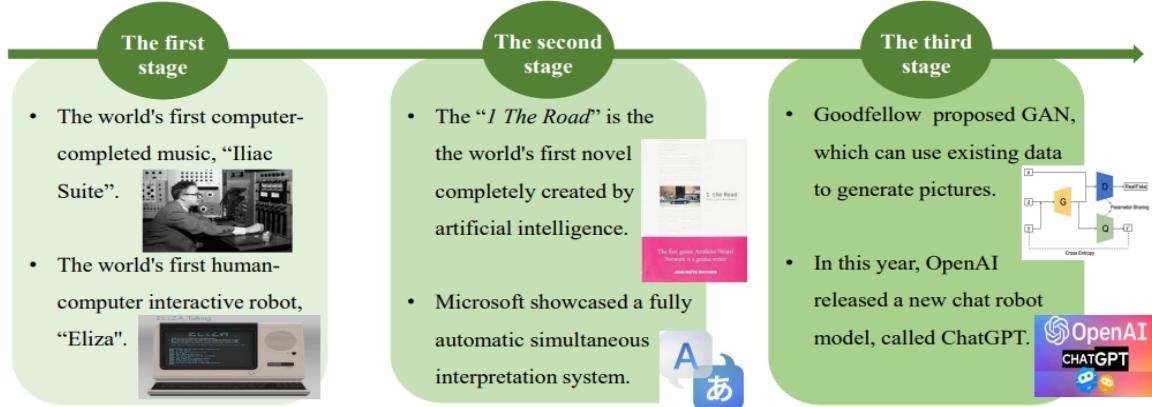


图 1: 发展历程

在第一阶段,研究者通过最原始的编程技术控制计算机实现内容输出。1957年,Hiller以及Isaacson完成了世界上第一个完全由计算机生成的音乐,“*Iliac Suite*”。接下来,世界上第一个人类与计算机交互的机器人——*Eliza*诞生了。*Eliza*展现出通过模式匹配和智能短语寻找合适答案的能力,但尚不具备理解能力。然而,许多人仍把*Eliza*视作当年人工智能的灵感来源。接下来的二十年就是一个沉淀的阶段了。

AIGC的第二阶段进展得益于大量数据库可用性的提高以及计算设备性能的进步。Road是世界上第一本完全由人工智能创作的小说。在那之后,微软还做出了一款全自动同声传译机口译系统,能够在短时间内以较高的准确率将英语翻译成汉语[14]。然而, 算法上的瓶颈极大的限制了AIGC生成丰富内容的能力。

第三阶段开始于2010年,此时的AIGC进入了一个快速发展的阶段。Goodfellow提出了生成对抗网络(GAN[15]),可以利用已有的数据去生成图片。2022年,OpenAI提出了一个新的聊天机器人模型——ChatGPT。到2023年1月时,ChatGPT每天的活跃人数已经超过了1300万人次!这些代表性的产品(例如ChatGPT)已经展现出了巨大的应用潜力和商业价值,引起了各个领域的广泛关注,包括企业家、投资者、学者和公众层面。

现在的AIGC生成内容的质量已经有了显著的进步。此外,在生成的内容上也丰富了许多,包括文本、图像、视频、代码等等。而这些都要得益于技术的进步,下面我们重点讲解一下最前沿的Diffusion模型以及GAN模型。

2 已有技术

在这一节中,我们重点关注AIGC背后的技术。我们可以将AIGC的基本技术分为两大类:生成技术和创造技术。具体来说,创造技术指的就是能够通过它来生成各种内容的技术,例如GAN和扩散模型。同时,生成技术例如Transformer不能生成内容,但对于AIGC的发展来说也是必不可少的。我们把目光重点放在创造技术上。

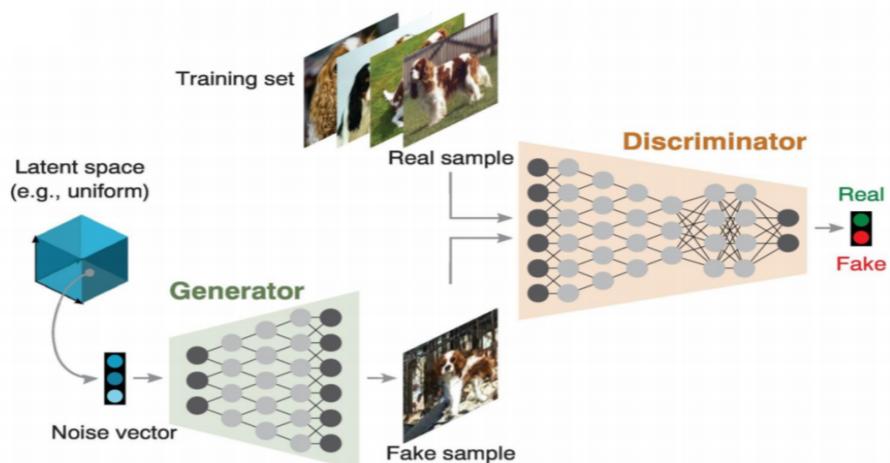
深度生成模型(Deep generative models)是一种利用神经网络生成样本的概率模型,大体上可以分为两类:基于似然的,例如autoregressive models[16]、flow models[17]等。以及基于能量的。由于基于能量的模型以非归一化概率为特征,一般来说会更加灵活,但难以训练。

下面介绍的GAN和Diffusion model与基于能量的模型高度相关。

2.1 GAN

2.1.1 整体框架

GAN(Generative Adversarial Network)全名为生成对抗网络。这一概念由Ian Goodfellow于2014年提出[15],并迅速成为了一个非常火热的研究话题。目前,关于GAN相关的变种的研究已有上千种。2019年图灵奖得主Yann LeCun也曾说:“GAN及其变种是数十年来机器学习领域最有趣的想法。”流程如下图所示:



下面我们来详细解释一下GAN。GAN框架通过一个对抗来估计生成模型,该框架同时训练两个模型:

- 生成模型(Generative model):G用于捕获数据分布;G主要做的是:使得虚假样本尽可能的欺骗判别模型D,使其辨别不出来。
- 判别模型(Discriminative model):D用于识别真实数据;D要做的是:将生成模型G生成的虚假样本与真实样本识别出来。

通俗来说,GAN的训练就是一个生成器与判别器互相博弈的过程。最终达到的理想效果应该是:生成器G生成的图像在判别器D的判别结果为0.5,即不知道到底是真图还是假图。

下面我们具体描述一下GAN的训练过程。

2.1.2 训练过程

GAN的训练过程就是找到生成器和判别器的最优参数,使得下面的值函数具有极小极大值,这个函数也就是GAN的目标函数:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.1)$$

其中,G代表生成器,D代表判别器,x代表真实数据, p_{data} 代表真实数据概率密度分布,z代表随机的输入数据,该数据是随机的高斯噪声。

我们详细解释一下这个过程。对上面这个式子,从判别器D的角度来说,判别器D希望能尽可能区分真实样本x和虚假样本G(z),因此 $D(x)$ 必须尽可能大,且 $D(G(z))$ 尽可能小,也就是 $V(D, G)$ 整体尽可能大。

从生成器G的角度来看,生成器G希望自己生成的数据 $G(z)$ 可以尽可能骗过判别器D,也就是希望 $D(G(z))$ 尽可能大, $V(D, G)$ 整体尽可能小。GAN的两个模块在相互对抗、相互训练,最后达到全局最优。

下图是GAN的训练过程示意图。平行线代表噪声z,它映射到了x,蓝色点线代表判别器D的输出,黑色圆点代表真实数据分布 p_{data} ,绿色实线代表生成器G的虚假数据的概率分布 p_g 。

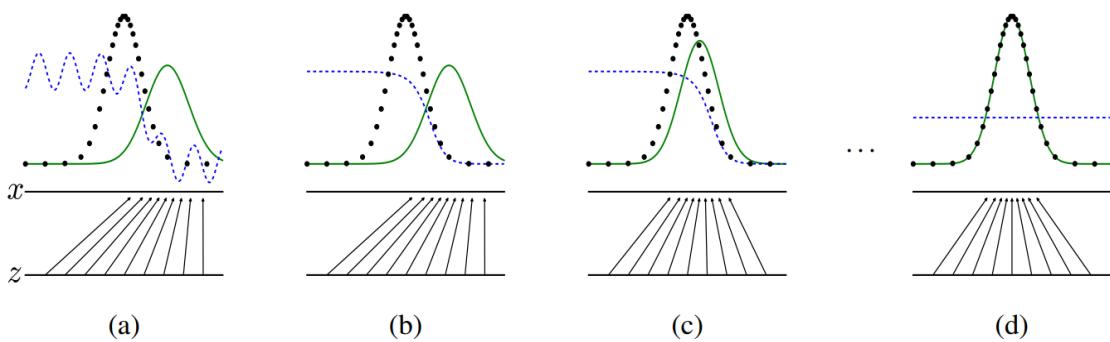


图 2: 训练过程

从上面的图示也可以看出,GAN的训练过程中,生成器G的概率密度分布慢慢逼近真实数据集的概率密度分布。最后,黑线与绿线几乎重合,这时 $D(G(z)) = 0.5$,即分不清输入图像是真实的还是生成器伪造的。此时就达到了我们期望的最优状态。

2.1.3 算法流程

下图是GAN具体的算法流程:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```

for number of training iterations do
  for  $k$  steps do
    • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
    • Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating distribution  $p_{data}(x)$ .
    • Update the discriminator by ascending its stochastic gradient:
      
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right].$$

  end for
  • Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
  • Update the generator by descending its stochastic gradient:
    
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)}))).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

```

图 3: 算法流程

下面我们给出GAN相关的数学理论而不加以证明,详情见论文[15]:

命题 2.1. For G fixed, the optimal discriminator D is

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

命题 2.2. If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G , and p_g is updated so as to improve the criterion

$$\mathbb{E}_{p \sim p_{data}} [\log D_G^*(x)] + \mathbb{E}_{x \sim p_g} [\log (1 - D_G^*(x))]$$

then p_g converges to p_{data}

定理 2.3. The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.

至此,我们以及从数学上证明了GAN的理论基础。论文[15]中也指出,在给定足够的训练数据和实验环境下,训练过程将收敛到最优 G 。

2.1.4 存在的问题

尽管如此,GAN模型也存在着一些明显的问题:

- 不收敛问题:由于GAN是两个神经网络之间的博弈,是一个对抗的过程。因此GAN在实际搭建过程对各种参数十分敏感,GAN的收敛性也一直是个问题。总需要精心设计才能完成一次训练任务。
- 崩溃问题:正是由于GAN模型的收敛不稳定问题,它在实际的训练过程中生成器可能会出现生成相同样本点的情况。从而导致判别器无法继续学习,整个模型崩溃。
- 无法处理复杂场景:基于上面的原因,GAN模型往往无法应对复杂的场景。只能处理一些简单的图片,如人像等。

2.2 Diffusion Model

扩散模型是一类概率生成模型,它的思路是先通过注入噪声破坏原图像,然后再通过去噪生成目标图像。具体的过程可参考下图:

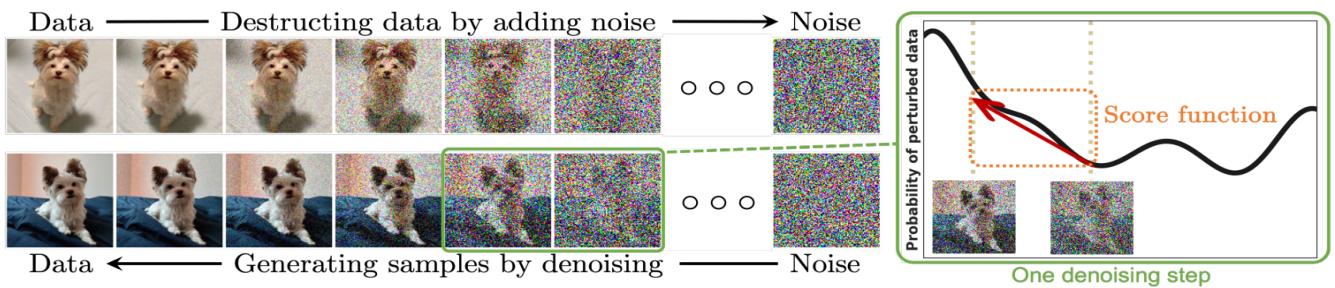


图 4: 生成过程

当前关于扩散模型已经有了很多的工作,下面我们主要介绍DDPM([18],[19],[20])和SGM([21],[22])两类扩散模型。

2.3 DDPM

2.3.1 正向过程

DDPM的构造主要利用了两个马尔科夫链:扩散过程将采样数据转化为噪声,逆向过程将噪声转化为生成的图像。这个思路也是借鉴了热力学中的想法。

扩散(Diffusion)在热力学中指细小颗粒从高密度区域扩散至低密度区域,在统计领域,扩散则指将复杂的分布转换为一个简单的分布的过程。Diffusion模型定义了一个概率分布转移模型T,能将原始数据的复杂分布 $p_{complex}$ 转换为一个简单的已知参数的先验分布 p_{prior} :

$$x_0 \sim p_{complex} \Rightarrow T(x_0) \sim p_{prior}$$

以上的这个概率转移模型是利用马尔科夫链来构造的。具体说,就是先给定起始的数据分布 $x_0 \sim q(x_0)$,正向过程会产生一系列的随机变量 x_1, \dots, x_T ,具有转移核 $q(x_t|x_{t-1})$ 。由概率的链式法则以及马氏性,我们可以将 x_1, x_2, \dots, x_T 在 x_0 下的条件分布,记作 $q(x_1, \dots, x_T|x_0)$,转化为:

$$q(x_1, \dots, x_T|x_0) = \prod_{t=1}^T q(x_t|x_{t-1}) \quad (2.2)$$

我们希望,当 $T \rightarrow \infty$ 时, x_T 的分布趋近于已知的先验分布 p_{prior} 。一个简单的想法就是利用正态分布,即:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I) \quad (2.3)$$

其中 $\beta_t \in (0, 1)$ 是一个预先定义的参数。这里的转移核是由Sohl-Dickstein et al.在[20]中的观察得到的。同样我们也可以选用其他的转移核,但为了方便讨论就选用此转移核。

在上述条件下,可以对 $\forall t \in \{0, 1, \dots, T\}$ 得到 $q(x_t|x_0)$ 的解析形式。令 $\alpha_t = 1 - \beta_t$ 以及 $\bar{\alpha}_t = \prod_{s=0}^t \alpha_s$,利用2.3以及正态分布的性质则有:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I) \quad (2.4)$$

给定 x_0 的情况下,可以直接得到 x_t 的一个采样,利用Gauss随机向量 $\epsilon \sim \mathcal{N}(0, I)$:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad (2.5)$$

因为 $\beta_t \in (0, 1)$,当 T 很大时则有 $\bar{\alpha}_t \rightarrow 0$, x_T 近似为标准高斯正态分布,因此我们有:

$$q(x_t) = \int q(x_T|x_0)q(x_0)dx_0 \approx \mathcal{N}(x_T; 0, I) \quad (2.6)$$

总结一下正向过程的思路,就是在重复地给原始数据分布添加噪声,直到变成一个简单固定分布为止。

2.3.2 逆向过程

因为扩散生成模型的最终目标是从 p_{prior} 中采样一个样本,转化为原始数据中的一个样本。我们可以先从先验分布中采样,然后通过一个马尔可夫链逐渐去除噪声的过程。显然,逆转上一节中的正向过程即可完成目标。先选择先验分布 $p(x_T) = \mathcal{N}(x_T; 0, I)$,因为正向过程中有 $q(x_T) \approx \mathcal{N}(x_T; 0, I)$ 。

如果可以直接逆转上面的正向过程即得所求。但 $q(x_{t-1}|x_t)$ 的分布不易给出,可以用分布 $p_\theta(x_{t-1}|x_t)$ 来近似 $q(x_{t-1}|x_t)$:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)) \quad (2.7)$$

其中 μ_θ 和 σ_θ 都是将要学习的函数,接受 x_t, t 作为参数。 θ 代表模型的参数。这样,在连续迭代多次后,可以得到近似的真实数据分布 $p_\theta(x_0)$ 为:

$$p_\theta(x_0) = \int p_\theta(x_{0:T})dx_{1:T}$$

其中 $p_\theta(x_{0:T})$ 为 x_0, x_1, \dots, x_T 的联合概率分布。由条件概率公式:

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

这样在确定好 μ_θ 以及 σ_θ 后,就可以确定 $p_\theta(x_{t-1}|x_t)$ 就被确认了下来。逆向过程如下图所示:

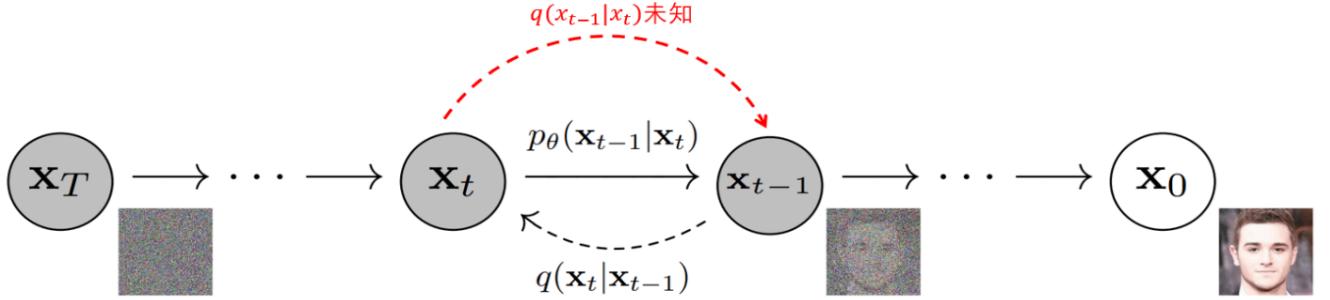


图 5: 逆向过程

下面就要确定理想的 μ_θ 和 σ_θ 。因为我们要调节参数 θ 的取值,使得 $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)$ 接近于向前过程 $q_{x_0:T} := q(x_0) \prod_{t=1}^T q(x_t|x_{t-1})$ 。这可以通过最小化预测 $p_{data} = q(x_0)$ 和 $p_\theta(x_0)$ 的交叉熵来达到:

$$\mathcal{L} = E_{q(x_0)}[-\log p_\theta(x_0)]$$

但我们没法直接写出 $p_\theta(x_0)$ 的表达式,没法直接计算,可以对上式做一些数学上的处理:

$$\mathcal{L} = -E_{q(x_0)} \log p_\theta(x_0) \quad (2.8)$$

$$= -E_{q(x_0)} \log (E_{q(x_{1:T}|x_0)} \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}) \quad (2.9)$$

$$\leq -E_{q(x_{0:T})} \log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \quad (2.10)$$

$$= E_{q(x_{0:T})} [\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] \quad (2.11)$$

其中 $q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$, (2.8)到(2.9)那一步利用了Jensen不等式。根据(2.10),为了优化 \mathcal{L} ,可以转而优化其上界 L_{VLB} :

$$L_{VLB} = E_{q(x_{0:T})} [\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] \quad (2.12)$$

$$= E_q [\log \frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p_\theta(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)}] \quad (2.13)$$

$$= E_q [\log \frac{q(x_T|x_0)}{p_\theta(x_T)} + \sum_{t=2}^T \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)] \quad (2.14)$$

$$= E_{q(x_{0:T})} [\log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}] \quad (2.15)$$

$$= E_q [\underbrace{-\log p_\theta(x_0|x_1)}_{L_0} + \sum_{t=2}^T \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t))}_{L_{t-1}} + \underbrace{D_{KL}(q(x_T|x_0) || p_\theta(x_T))}_{L_T}] \quad (2.16)$$

上面公式的推导中用到了贝叶斯公式:

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

以及马尔可夫链的性质: $q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1})$ 。注意到(2.15)中的 L_T, x_T 和 x_0 一个是先验分布,一个是数据分布,都是固定的,故 L_T 为常数,最小化 L_{VLB} 时忽略。只去研究 L_0 和 $L_t, t \in \{1, 2, 3, \dots, T-1\}$ 。

下面我们先研究 $q(x_{t-1}|x_t, x_0)$ 和分布 $p_\theta(x_{t-1}|x_t)$ 之间的KL散度。由2.7, 分布 $p_\theta(x_{t-1}|x_t)$ 是一个高斯分布, 可以直接算出它的平均值和方差。而分布 $q(x_{t-1}|x_t, x_0)$ 可以根据贝叶斯定律, 推导得:

$$q(x_{t-1}|x_t, x_0) = q(x_t|x_{t-1}, x_0) \frac{q(x_{t-1})|x_0}{q(x_t|x_0)} \quad (2.17)$$

$$= q(x_t|x_{t-1}) \frac{q(x_{t-1})|x_0}{q(x_t|x_0)} \quad (2.18)$$

$$\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t}\right)\right) \quad (2.19)$$

继续推导可以发现 $q(x_{t-1}|x_t, x_0)$ 同样是一个高斯分布。假设:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$$

其中这两个新变量为:

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

$$\tilde{\mu}_t(x_t, x_0) = \left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}}x_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}}\right)$$

因此, 最小化 L_t 这个KL损失实际上就是拉近这两个高斯分布的距离, 而多元正态分布之间的KL散度可以直接计算出来:

$$L_t = E_q\left[\frac{1}{2\|\sum_\theta(x_t, t)\|_2^2}\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2\right] + C$$

上式中C是一个不依赖于 θ 的常数, 为使模型简单一点, 可以令 $\sum_\theta(x_t, t) = \sigma_t^2 I$, 其中 σ_t^2 可以设置为 β_t 或 $\tilde{\beta}_t$ 。更进一步的, 将 $x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}z_t)$ 代入, 有:

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}z_t)$$

为了降低学习的难度, 直接定义:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}z_\theta(x_t, t))$$

这样就把上面的公式简化为了:

$$L_t - C = E_{x_0, z_t}\left[\frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2\right] \quad (2.20)$$

$$= E_{x_0, z_t}\left[\frac{\beta_t^2}{2\alpha_t(1 - \bar{\alpha}_t)\sigma_t^2}\|z_t - z_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1 - \bar{\alpha}_t}z_t, t)\|^2\right] \quad (2.21)$$

最后把上面的系数去掉, 就得到了简化的训练目标:

$$L_t = E_{t, x_0, \epsilon_t}[\|\epsilon_t - z_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1 - \bar{\alpha}_t}z_t, t)\|^2]$$

具体的算法流程可参考下图:

Algorithm 1 Training	Algorithm 2 Sampling
<pre> 1: repeat 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 3: $t \sim \text{Uniform}(\{1, \dots, T\})$ 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 5: Take gradient descent step on $\nabla_{\theta} \ \epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\ ^2$ 6: until converged </pre>	<pre> 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: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{1 - \bar{\alpha}_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 5: end for 6: return \mathbf{x}_0 </pre>

图 6: 训练流程

2.4 SGM

分数模型(Score-based models)作为扩散模型的一种,其主要目标是估计与数据分布相关的梯度(即分数,后文会介绍定义)并使用郎之万动力学的方法从估计的数据分布中进行采样来生成新的样本。

假设我们有一组数据 $\{x_1, x_2, \dots, x_N\} \sim p_{data}(x)$ 。其中 p_{data} 的分布是未知的。对于这个数据集,生成模型的目标就是利用它来训练一个模型,可以从样本中随意采样来生成新的数据点。

分数模型的框架主要包括两个关键步骤,一个是分数匹配,还有一个是郎之万动力学。整体思路如下图:

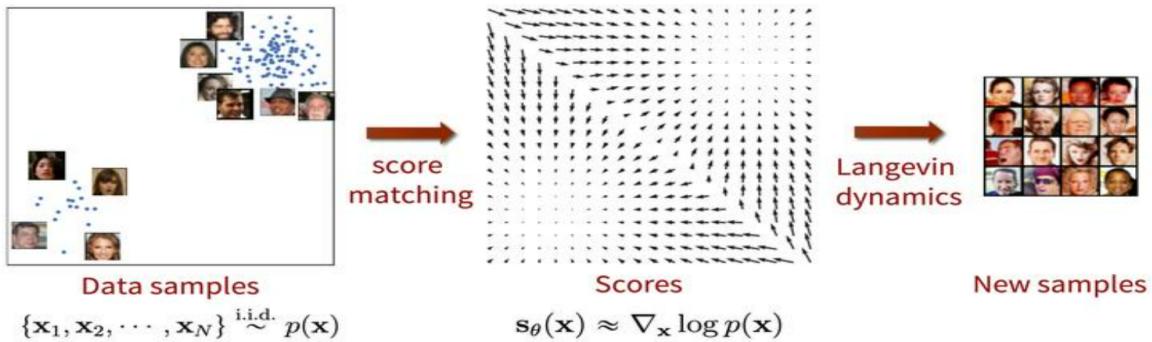


图 7: 思路框架

2.4.1 分数匹配

我们首先介绍一些概念:

- 分数函数:通过对分数函数建模,可以避免难以处理的常数归一化的困难。其中分数的定义为:

$$\frac{\partial(\log p(x))}{\partial x} = \nabla_x \log p(x)$$

解释一下上面分数的定义。一般来说,实际场景下的数据往往是多维的。分数就是一个这样的矢量场,方向是:对于输入数据(样本)来说,其对数概率密度增长最快的方向。如下图所示:

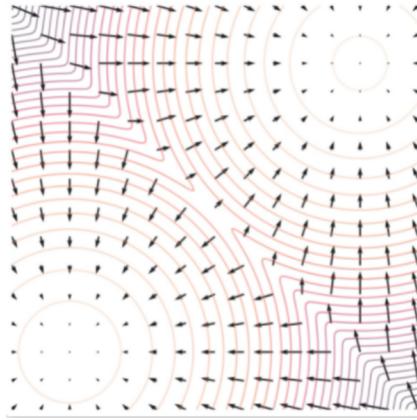


图 8: 图示

- 分数模型(score-based model):一个用来学习分数分布模型的神经网络,用 $s_\theta(x)$ 来表示。
- 分数匹配(Score Matching):类似于基于似然估计的模型,可以通过最小化模型和数据分布之间的Fisher散度来训练基于分数的模型,其表达式为:

$$\mathbb{E}_{p(x)}[\|s_\theta(x) - \nabla_x \log p(x)\|_2^2] = \int_{x \sim X} p(x) \|s_\theta(x) - \nabla_x \log p(x)\|_2^2 dx$$

但我们没法直接计算出这个式子。因为分布 $p(x)$ 是未知的。而Score matching方法就可以让我们在不知道真实的 $p(x)$ 的情况下最小化这个loss。推导如下:我们先把上面的期望写开,二次项打开,可以得到

$$\mathcal{L} = \mathbb{E}_{p(x)}[\|\nabla_x \log p(x) - s_\theta(x)\|^2] \quad (2.22)$$

$$= \int p(x)[\|\nabla_x \log p(x)\|^2 + \|s_\theta(x)\|^2 - 2(\nabla_x \log p(x))^T s_\theta(x)]dx \quad (2.23)$$

第一项对于 θ 来说是常数可以忽略。

第二项为

$$\int p(x) \|s_\theta(x)\|^2 dx.$$

对第三项,若 x 的维度为N,由分部积分可得:

$$-2 \int p(x)(\nabla_x \log p(x))^T s_\theta(x) dx = -2 \int p(x) \sum_{i=1}^N \frac{\partial \log p(x)}{\partial x_i} s_{\theta i}(x) dx \quad (2.24)$$

$$= -2 \sum_{i=1}^N \int \frac{\partial p(x)}{\partial x_i} s_{\theta i}(x) dx \quad (2.25)$$

$$= 2 \sum_{i=1}^N - \int \frac{\partial p(x) s_{\theta i}(x)}{\partial x_i} dx + \int p(x) \frac{\partial s_{\theta i}(x)}{\partial x_i} dx \quad (2.26)$$

$$= 2 \int p(x) \text{tr}(\nabla_x s_\theta(x)) dx \quad (2.27)$$

所以最后的loss可以写成:

$$\mathcal{L} = \int p(x) \|s_\theta(x)\|^2 dx + 2 \int p(x) \text{tr}(\nabla_x s_\theta(x)) dx \quad (2.28)$$

$$= \mathbb{E}_{p(x)}[\|s_\theta(x)\|^2 + 2 \text{tr}(\nabla_x s_\theta(x))]. \quad (2.29)$$

但是还存在一个问题就是:当遇到高维数据时,Jacobian矩阵在计算上十分复杂,需要经历多次反向传播来求偏微分。有两种方法来解决:去噪分数匹配(denoising score matching)和切片分数匹配(sliced score matching)。

去噪分数匹配:设原始数据为 x ,对原始数据(训练样本)加入噪声后记为 \tilde{x} (通过预先设定的分布 $q_\sigma(\tilde{x}|x) \sim \mathcal{N}(\tilde{x}; x, \sigma^2 I)$ 来扰动原始数据,方差 σ^2 提前规定)。那么加噪声后的数据分布为;

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

那么此时的训练函数为

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

在去噪分数匹配方法下,只需要从标准高斯分布 $\mathcal{N}(0, I)$ 中采取随机高斯噪声 ϵ ,然后通过重参数化方法 $\tilde{x} = x + \epsilon$ 就可以得到加入噪声后的样本。加入噪声后的数据分布满足 $q_\sigma(\tilde{x}|x) \sim \mathcal{N}(\tilde{x}; x, \sigma^2 I)$ 。在这种方法下,训练的网络估计的是 $q_\sigma(\tilde{x}|x)$ 而不是原始数据分布,就要求加入的噪声方差 σ 足够小才行。

切片分数匹配(sliced score matching):前文提及Jacobian矩阵的迹 $\text{tr}(\nabla_x s_\theta(x))$ 在高维数据的情况下需要很大的计算量,因此可以用一个服从简单分布(高斯分布)的随机变量 v 将网络分数 $\nabla_x s_\theta(x)$ 随机映射为一个Jacobian向量,即:

$$v^T \frac{\partial s_\theta(x)}{\partial x} v = v^T \frac{\partial [v^T s_\theta(x)]}{\partial x}$$

此时的训练损失函数变为

$$\mathbb{E}_v \mathbb{E}_{p_{data}} [v^T \frac{\partial [v^T s_\theta(x)]}{\partial x} + \frac{1}{2} \|s_\theta(x)\|_2^2]$$

但这个方法相比于去噪分数匹配的缺陷就在于计算量会大很多。

2.4.2 朗之万动力学

当我们已经通过神经网络学习到了数据分布的分数函数时,接下来可以用朗之万动力学采样方法从分布中生成新的样本:假设初始数据满足先验分布 $x_0 \sim \pi(x)$,使用迭代过程:

$$x_{t+1} = x_t + \frac{\epsilon}{2} \nabla_x \log p(x) + \sqrt{\epsilon} z_t, \quad z_t \sim \mathcal{N}(0, I), \quad t = 0, 1, 2, \dots, T$$

其中, $z_t \sim \mathcal{N}(0, I)$ 为标准高斯分布。理论上,当 $K \rightarrow \infty$ 且 $\epsilon \rightarrow 0$ 时,最终生成的样本 x_T 将会服从原数据分布 $p_{data}(x)$ 。

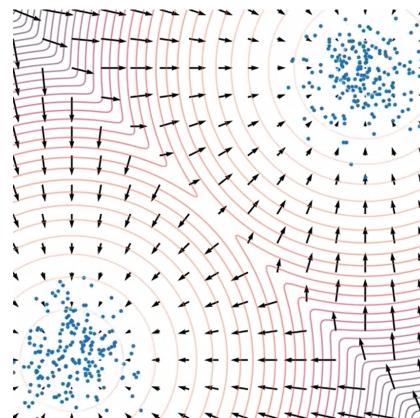


图 9: 最终采样的结果

2.5 3D Generation

生成模型在2D领域上的巨大成功,促使着研究人员继续探索3D数据的生成。这事实上就是一个真实物理世界的模拟。与2D数据的单一表达形式不同,3D物体具有深度图像、体素[23]、点云[24] [25]、网格[26]和神经场[27]等多种表达形式,也分别具有不同的优点和缺点。

但由于3D缺乏足够的3D训练数据以及合适的架构,目前的3D生成仍然具有挑战性。基于扩散模型, DreamFusion[28]提出通过预先训练的文本到2D模型来解决这个问题。

3D生成领域的另一个分支是从单视图图像或多视图图像重建三维物体,称为Image-to-3D。隐式神经辐射场(Neural Radiance Fields, NeRF)是多视图三维重建的一个新分支[29]。接下来具体讲解一下NeRF技术:

2.6 NeRF

NeRF(Neural Radiance Fields神经辐射场)本质上就是用深度学习完成了图形学中的3D渲染任务,是一种隐式的场景表示。NeRF要处理的任务是新视角的合成。具体来说,就是利用2D的posed images作为监督,即可生成多视角的复杂的3D场景。可参考下图:

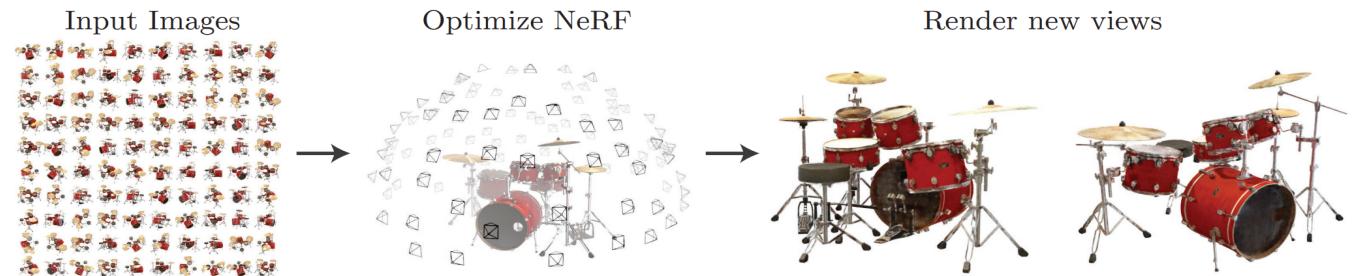


图 10: 图解

下面来介绍一下NeRF具体的思路:如下图所示,主要分成场景表示、体积渲染、优化三个部分,如下图所示

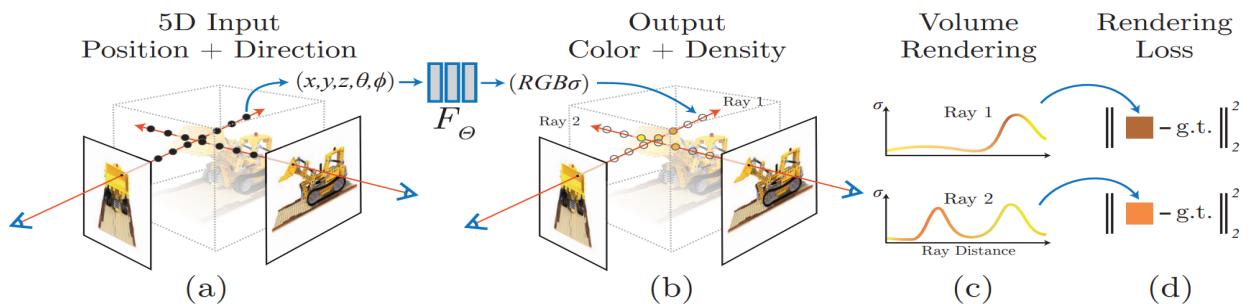


图 11: 算法流程

2.6.1 场景表示

要解决的第一个问题是如何用NeRF来表示3D场景。NeRF将一个连续的场景表示为一个输入为5D向量的函数,包括一个空间点的3D坐标位置 $x = (x, y, z)$,以及视角方向 $d = (\theta, \phi)$ 。而输出则为颜色 $c = (r, g, b)$ 以及体积密度 σ 。用一个MLP神经网络可以写作:

$$F_\Theta : (x, d) \rightarrow (c, \sigma)$$

我们可以优化该网络的权重,使得对于每个输入5D坐标,都会得到对应的密度和有方向的发射颜色。

论文中还提到,NeRF约束了预测体积密度 σ 的网络的输入仅仅是位置 X ,而预测RGB颜色 C 的网络的输入是位置和视角方向。通过这种方式可以鼓励网络学习到多视角连续的表示。

2.6.2 体积渲染

下面讨论的是如何用NeRF来渲染2D图像。NeRF函数最终得到的是一个3D空间点的颜色和密度信息。当用一个相机去对这个场景成像时,所得到的2D图像上的一个像素实际上对应了一条从相机出发的射线上的所有连续空间点。所以我们需要通过渲染算法来得到最终渲染的颜色。

论文中首先介绍了经典的体素渲染volume rendering方法。体素密度 $\sigma(x)$ 可以被理解为:一条射线 r 在经过 x 处的一个无穷小的粒子时被终止的概率,并且这个概率是可微的。也可以说是这个点的不透明度。我们可以用积分的方法来求解最终的渲染颜色。

将一个相机射线标记为 $r(t) = o + td$,其中 o 是射线原点, d 是相机射线角度, t 的近端和远端边界分别为 t_n 和 t_f 。那么这条射线的颜色 $C(r)$ 为:

$$C(r) = \int_{t_n}^{t_f} T(t) \cdot \sigma(r(t)) \cdot c(r(t), d) dt$$

此处 $T(t)$ 是射线从 t_n 到 t 这一段路径上的累积透明度,可以被理解为这条射线从 t_n 到 t 一路上没有击中任何粒子的概率。其具体形式为:

$$T(t) = \exp\left(-\int_{t_n}^t \sigma(r(s)) ds\right)$$

但是在计算机中我们无法直接求这个积分,只能采用数值近似的方式:最直接的想法是在需要求积的区域内均匀地采样N个点进行近似计算。但论文中指出这样的方法具有局限性,因为MLP只需要学习一系列离散点的信息,最终生成的结果不是很清晰。文中采用了分层抽样的方法:把 $[t_n, t_f]$ 分成均匀分布的小区段,接着对每个小区段进行均匀采样

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right]$$

我们使用采样的样本,用积分法则来计算 $C(r)$:

$$\hat{C}(r) = \sum_{i=1}^N T_i \cdot (1 - \exp(-\sigma_i \cdot \delta_i)) \cdot c_i$$

其中, $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$, $\delta_i = t_{i+1} - t_i$ 是相邻样本之间的距离。

2.6.3 优化

论文中也指出,前面所述的核心算法,还不够实现最先进的效果。因此我们还引入了两项优化措施,以帮助模型能够表征更高分辨率的复杂场景。第一个是输入坐标的位置编码,以帮助MLP能够表示高频函数;第二个是层次化的采样方案,可以有效地采样模型的高频表示。

尽管神经网络理论上可以逼近任意函数,但实验发现如果直接用网络 F_θ 直接处理输入坐标 $xyz\theta\phi$,很难表现出高频的细节。这个结果与Rahaman等人近期的工作是一致的。他们证明了深度网络倾向于学习到频率较低的函数。他们还证明,使用高频函数,把输入映射到更高维的空间中,再喂给神经网络,可以更好地拟合具有高频变化的数据。

我们运用这个理论,重新构建函数 F_θ ,作为两个函数的组合函数: $F_\theta = F'_\theta \circ \gamma$,该方法显著提高了性能。

其中 γ 表示一个从 \mathbb{R} 到高维空间 \mathbb{R}^{2L} 的映射, F'_θ 为普通的MLP。我们使用的编码函数为:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \dots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$$

该函数分别应用于 X 的三个坐标,以及视角方向单位向量 d 的三个坐标。

另一个方面是优化之前的采样方案。因为之前的渲染过程计算量很大,每条射线会采样很多点。但实际上很多都是空白和遮挡的区域,没有贡献。所以我们借鉴了立体渲染的工作,提出层次化的采样:根据期望的渲染效果,来按比例的分配采样点。

为此我们采用了一种”coarse to fine”的策略。同时优化coarse网络和fine网络。首先对于coarse网络,采样较为稀疏的 N_c 个点。将前述的离散求和函数重新表示为:

$$\hat{C}_c(r) = \sum_{i=1}^N \omega_i c_i, \quad \omega_i = T_i \cdot (1 - \exp(-\sigma_i \cdot \delta_i))$$

接下来对 ω_i 做归一化:

$$\hat{\omega}_i = \frac{\omega_i}{\sum_{j=1}^{N_c} \omega_j}$$

此处的 $\hat{\omega}_i$ 可以看作沿着射线的概率密度函数,如下图所示:

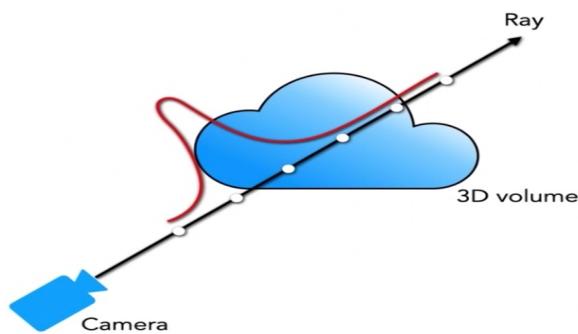


图 12: 分布情况

接下来,基于得到的概率密度函数来采样 N_f 个点,并且用这 N_f 个点和前面的 N_c 个点一同计算fine网络的渲染结果 $\hat{C}_f(r)$ 。亦如下图所示。

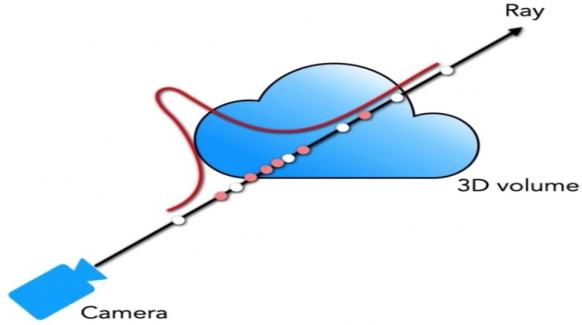


图 13: 采样

最后补充一个细节,训练时定义的损失函数:直接定义在渲染结果上的 L_2 损失(同时优化coarse和fine):

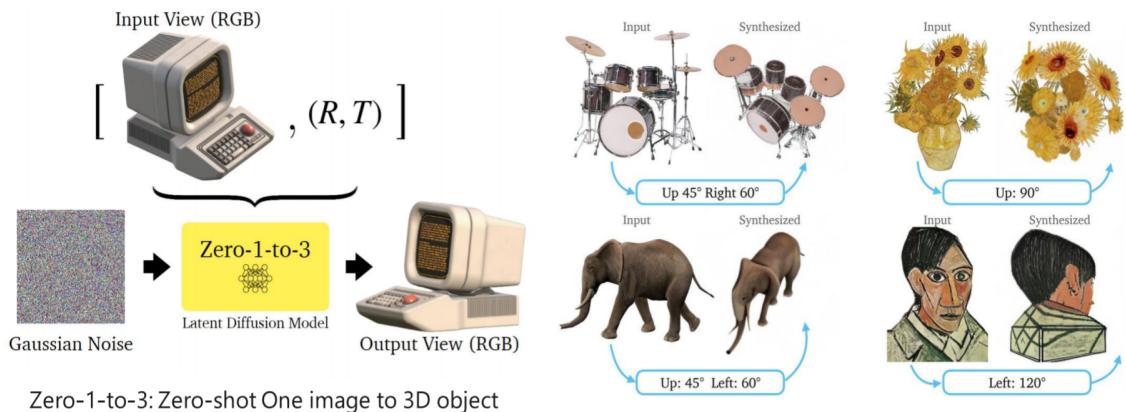
$$\mathcal{L} = \sum_{r \in R} [\|\hat{C}_c(r) - C(r)\|_2^2 - \|\hat{C}_f(r) - C(r)\|_2^2]$$

3 3D生成代表解决方案

3.1 分类

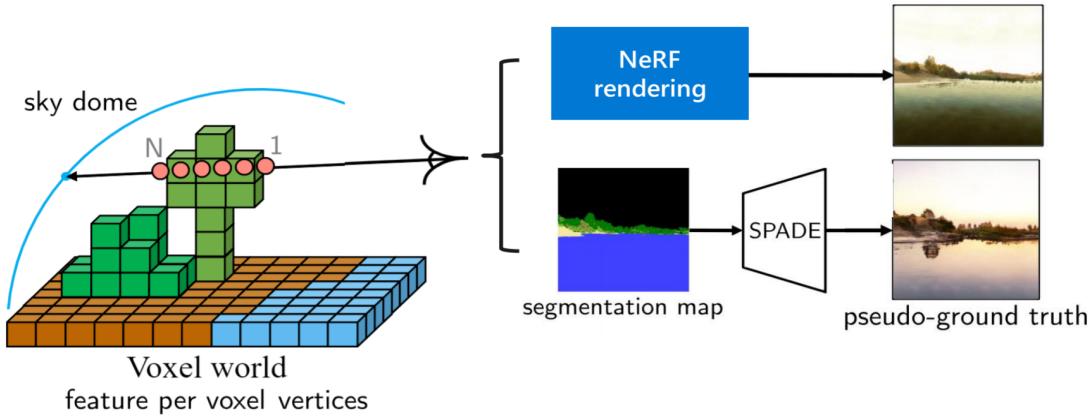
由上文提到的,目前的3D生成尚没有取得2D领域如此大的进展,一方面也是因为3D数据的匮乏。这逼迫着人们必须利用现有的为数不多的数据来做3D生成,从而涌现出了许多十分有趣的工作和想法。现在的3D生成工作大体上可以分为如下这几类:

- Pseudo 3D generation(仿3D派)的思路是不去直接建模一个3D的物体,而是利用现有的2D的成熟模型,每次产生一张不同视角下的图像,如下图所示:



但由于其本质上还是一个2D层面的生成,难免会有一些闪烁、不连续的地方。

- Primitive to 3D translation(转化3D派)的思路是先利用一些现有的表达(例如体素等)做过渡,再去转化成我们希望的3D场景。例如2021年提出的GANcraft,如下图所示:



具体的细节可以参考论文[30],篇幅有限,在这里不做赘述。

- Lift 2D to 3D(升维派)的思路是利用现有的成熟的预训练的2D模型(DDPM)去做3D生成,利用2D的图像生成3D物体。例如Dream field、DreamFusion、ProlificDreamer等等。

下面的几节我们以DreamFusion[28]和后续改进的ProlificDreamer[31]为代表,讲解一下它们背后的思路和数学原理。

3.2 DreamFusion

3.2.1 算法流程

得益于扩散模型等成熟的工具, Text2Image这一领域已经取得了巨大的进展,AIGC也因此得到了人们的许多关注。但与此同时,Text2Shape还存在着许多的困难,一个主要的难题就是3D数据的稀缺性。在现有的开源数据集中,3D数据的规模比成熟的2D数据还是小了好几个数量级,这一定程度上限制了3D生成的发展。

因此,人们想到,能否不利用任何3D数据,并结合已有的成熟工具(Diffusion Modle)来做3D生成呢?在这个背景下,DreamFusion应运而生。与此同时,也有着许多类似的工作,如DreamFields,DreamCLIP等。它们的核心思想就是利用预训练的text-to-image扩散模型作为监督信号,不断优化3D物体,最终达到一个比较好的3D生成结果。

总体上来说,DreamFusion就是扩散模型与NeRF的一个很好的结合。它的流程图如下所示:

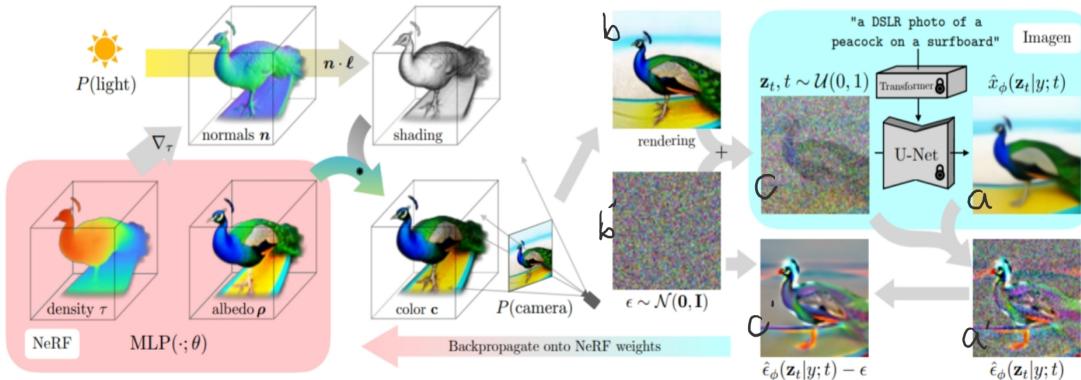


图 14: DreamFusion流程图

在生成阶段,DreamFusion采用的DM模型出自谷歌的Imagen(见论文),Imagen通过DM可以实现text-to-image的效果。于是DreamFusion将Imagen作为文生图的工具,希望在给定的文本text作为输入的情况下,得到文本对应的图像。在图像的生成过程中,不同视角的生成受到文本中与方向有关的描述所控制。

对于3D表示方面,DreamFusion采用了Mip-NeRF,这样就把当下流行的两个模型很好的结合了起来。下面来具体描述一下DreamFusion工作的流程:

- 第一步,给定一段输入的文本(例如上图中的”a DSLR photo of a peacock on a surfboard”),并将其送到训练好的Imagen中,此时可以得到该文本所对应的生成图像,记作图像a。
- 另一方面,我们随机初始化一个NeRF场,并利用它rendering出一张图,记作图像b。注意这里我们要控制render的图像分辨率也是64*64的,从而保证与Imagen的输出保持一致。
- 最后一步,我们对图像b加上高斯噪声b',得到图像c。然后利用训练好的Diffusion model对这个图像c执行去噪过程,得到图像a'。把此处预测的噪声记作 $\hat{\epsilon}_\phi(z_t|y; t)$,然后对估计的噪声和添加的噪声之间求loss,并用求得的结果来更新NeRF的参数,使得render的结果b更加真实。如此迭代下去,就能达到text-to-3D的效果。

而如何通过噪声之间的loss来更新NeRF的参数呢?这里就要用到下文提到的SDS。

3.2.2 SDS

SDS是该论文提出的一种计算loss的方式,下面用一张图来解释一下此过程:

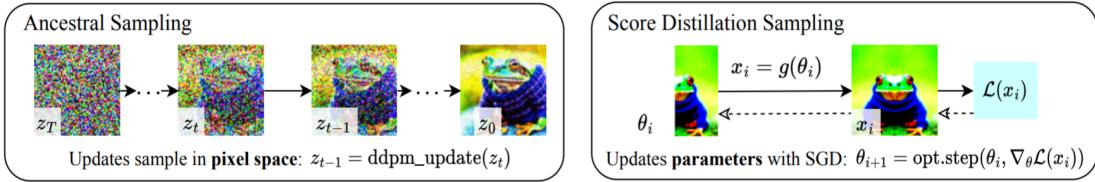


Figure 2: Comparison of 2D sampling methods from a text-to-image diffusion model with text “*a photo of a tree frog wearing a sweater.*” For score distillation sampling, as an example we use an image generator that restricts images to be symmetric by having $\mathbf{x} = (\text{flip}(\theta), \theta)$.

图 15: SDS流程图

详细描述一下这个过程。对于用 θ_i 表示的NeRF参数化模型,然后通过可微的图像生成器 $g(\theta_i)$ 得到了变换后的 x_i (NeRF的可微渲染过程)。再将生成器 $g(\theta_i)$ 得到的 x_i 与真值的 \hat{x}_i 求损失 $L(x_i)$,并通过梯度下降的方式更新 θ_i ,从而使 θ_i 满足预期要求。

下面用数学公式描述一下:

$$\mathcal{L}_{Diff}(\phi, x) = \mathbb{E}_{t \sim \mathcal{U}(0,1), \epsilon \sim \mathcal{N}(0,I)} [\omega(t) \|\epsilon_\phi(\alpha_t x + \sigma_t \epsilon; t) - \epsilon\|_2^2]$$

上面是扩散模型的基础公式,对上式求梯度可得:

$$\nabla_\theta \mathcal{L}_{Diff}(\phi, \mathbf{x} = g(\theta)) = \mathbb{E}_{t,\epsilon} \left[w(t) \underbrace{(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon)}_{\text{Noise Residual}} \underbrace{\frac{\partial \hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\mathbf{z}_t}}_{\text{U-Net Jacobian}} \underbrace{\frac{\partial \mathbf{x}}{\partial \theta}}_{\text{Generator Jacobian}} \right] \quad (2)$$

因为此时的 x 变成了可微的 $g(\theta)$,在求梯度时就要计算多出的两项。另外论文作者也发现,忽略U-Net的Jacobian项有助于提升 $g(\theta)$ 的优化效率,所以有了下面的公式,也就是本文最终用到的SDS。

$$\nabla_\theta \mathcal{L}_{SDS}(\phi, x = g(\theta)) = \mathbb{E}_{t,\epsilon} [\omega(t) (\hat{\epsilon}_\phi(z_t; y, t) - \epsilon) \frac{\partial x}{\partial \theta}]$$

但SDS也存在着很多的不足,例如过饱和、过平滑、缺少细节等,下面的ProlificDreamer对这些问题做了改进,并提出了新的VSD算法。

3.3 ProlificDreamer

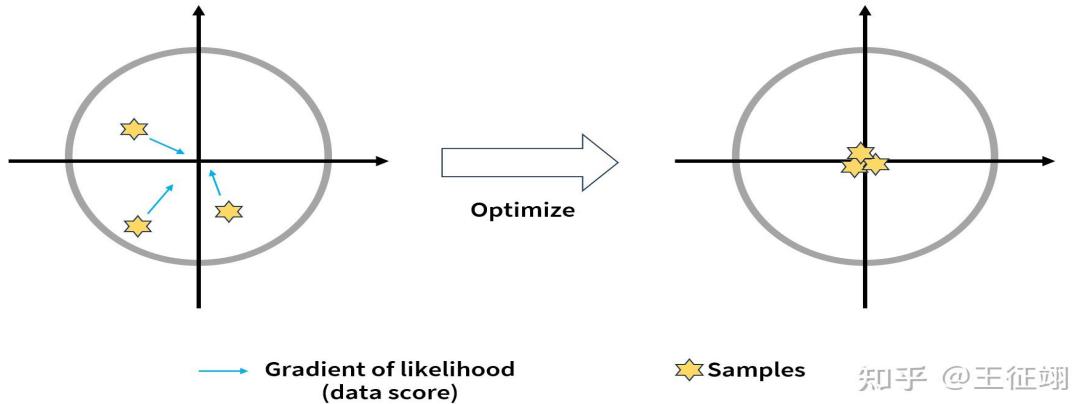
让我们首先回顾一下SDS的优化过程:对于一个用 θ 进行参数化的3D表示,给定文本 y ,通过对随机采样的视角 c 下渲染的2D图像,优化SDS loss,就能让3D物体越来越逼真。其中, g 代表可微渲染方程。

$$\nabla_\theta \mathcal{L}_{SDS}(\phi, x = g(\theta)) = \mathbb{E}_{t,\epsilon} [\omega(t) (\hat{\epsilon}_\phi(z_t; y, t) - \epsilon) \frac{\partial x}{\partial \theta}]$$

SDS公式中 $\hat{\epsilon}_\phi(z_t; y, t)$ 这一项主要是由预训练的图像扩散模型提供的,它的作用是让渲染出来的2D图像尽可能逼真。这一项事实上就是扩散模型预测出来的噪声大小,而噪声通过一个线性变换可以变为分布的score,可就是分布的似然的梯度。这个过程就是不断地最大化渲染出来图像的似然函数。

ϵ 项是一个均值为零的高斯噪声,可以认为这一项在期望下不起作用。因此SDS实际上就是在不断优化样本的似然。

但是SDS的方法也存在着几个明显的问题。第一是似然最高不一定代表生成的质量好。SDS的优化过程会让样本偏离“典型样本”,并且由于不同的起始点可能收敛到同一个mode,多样性也比较差。



另一个因素是SDS的优化目标对于 $t \sim Uniform(0, 1)$ 做了一个集成。这样做的主要目的是方便优化,但实际上只需要对 $p_0(x_0|y)$ 这个分布找mode。对于一个比较大的 t ,分布 $p_t(x_t|y)$ 会比较平滑,又导致SDS出来的结果在细节上比较缺失。为了缓解过平滑的问题,SDS一般采用一个非常大的CFG来增加细节,但这样又会导致SDS的生成结果过于饱和。

3.3.1 VSD

为了解决上面提到的SDS的问题,VSD方法对SDS进行了一些小改动。同时优化多个样本,把样本们视为一个变分分布。如下图所示,把SDS更新方向的第二项由零均值的高斯换成了变分分布的Score。

$$\text{SDS/SJC: } \nabla_{\theta} \mathcal{L}_{\text{SDS}}(\theta) \triangleq \mathbb{E}_{t, \epsilon, c} \left[\omega(t) (\underbrace{\epsilon_{\text{pretrain}}(\mathbf{x}_t, t, y) - \epsilon}_{\text{Gaussian noise (zero-mean)}}) \frac{\partial \mathbf{g}(\theta, c)}{\partial \theta} \right]$$

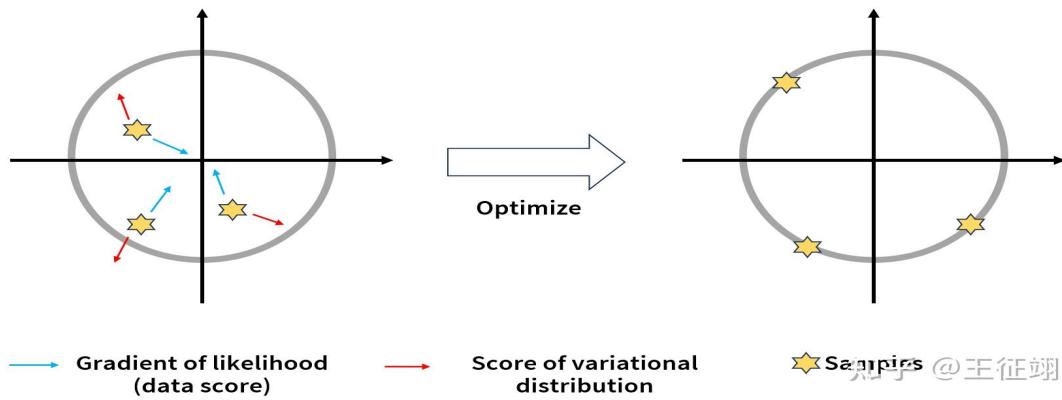
$$\text{VSD: } \nabla_{\theta} \mathcal{L}_{\text{VSD}}(\theta) \triangleq \mathbb{E}_{t, \epsilon, c} \left[\omega(t) (\underbrace{\epsilon_{\text{pretrain}}(\mathbf{x}_t, t, y) - \epsilon_{\phi}(\mathbf{x}_t, t, c, y)}_{\text{Variational Score (encourage diversity)}}) \frac{\partial \mathbf{g}(\theta, c)}{\partial \theta} \right]$$

↓ ↓

Pretrained **Unknown**

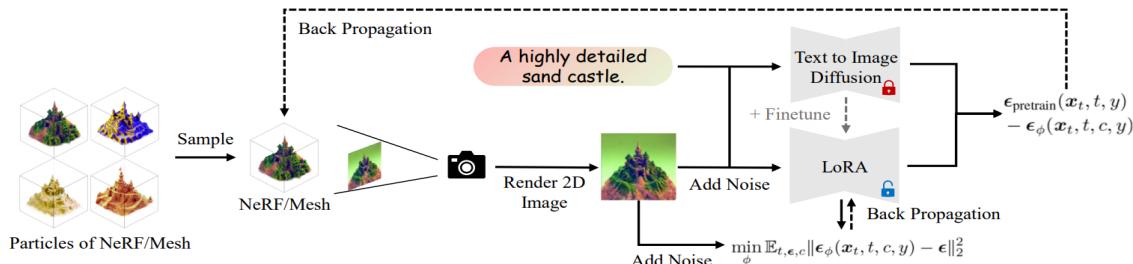
知乎 @王征翊

这样做的动机主要是,加入变分分布后可以让样本收敛到图中灰色圆环上的典型样本,同时增大样本的多样性。



另一方面就是,论文中发现VSD的CFG大小可以设置为正常的值。相比之下SDS往往只能设置成很大的值。因此,我们的VSD很大程度上缓解了SDS带来的过饱和问题。

3.3.2 算法流程



如上图所示,VSD的算法流程可以总结成这样一张示意图。在每一个迭代过程中,我们会把3D物体渲染出2D的图片,把这个2D图片送到diffusion以及lora中,利用VSD算法算出一个更新的方向,再反传给3D物体来优化。同时,这个2D图片也会被用来训练变分分布的score。

4 总结与展望

从之前的内容中我们可以看到AIGC的巨大潜力,本节是一个对AIGC的总结与未来的展望。包括AIGC的主要优点与不足、面临的挑战、未来发展的前景等等。

4.1 AIGC的优点与不足

我们列举了AIGC的优点与缺点,如下两张图所示:

Pros	Description
Efficiency and scalability	AIGC can provide many benefits over traditional human writing, including speed and language localization. Another benefit of AIGC is its ability to create personalized social media posts for various sites.
Help scientific research	AI can assist in analyzing large datasets through machine learning algorithms to identify patterns and correlations that might not be easily visible to humans.
For search engine optimization	AI can analyze the content on a website and suggest changes to make it more SEO-friendly.
Overcome writer's block	AI tools can create detailed outlines and key points to help the writer determine what should be included in the article.

Cons	Description
Ethics and trust	Due to the lack of intended tone and personality, the generated answers may be filtered out.
Exacerbate social imbalances	Some people can use AI tools to complete the original tasks at various speeds, whereas others may need to spend a significant amount of time thinking and creating content.
Negative effects on education	AIGC may lack the human touch and personalization that are necessary for effective learning.
Inadequate empathy	For instance, AI-generated music might not have the same emotional depth and authenticity as music performed and composed by humans.
Human involved	People still need to be involved and articles quality-checked.
Missing creativity	It is hard for AIGC to come up with new content with the latest, trending ideas and topics.

4.2 AIGC的发展前景

AIGC在各个领域产生现实和多样化的产出方面已经取得了显著的成功，但在实际应用中仍然存在许多挑战。除了需要大量的训练数据和计算资源以外,我们还列出了一些最重要的挑战如下:

(1)缺乏可解释性。虽然AIGC模型可以产生令人印象深刻的输出,但理解模型如何得到输出仍然具有挑战性。当模型产生不希望的结果时,由于缺乏可解释性,很难控制输出。

(2)伦理和法律问题。AIGC模型容易出现数据偏差。例如,语言模型对英语文本的训练可能会对西方文化产生偏见。侵犯版权和隐私同样是不可忽视的潜在法律问题。此外, AIGC模型还具有潜在的恶意使用。例如, 学生可以利用这些工具在论文作业中作弊。

(3)特定领域的技术挑战。在当前和不久的将来,不同的领域需要它们独特的AIGC模型。每个领域仍然面临着其独特的挑战。例如,Stable Diffusion, 流行的文本到图像的AIGC工具,偶尔会生成与用户期望相差甚远的输出,例如把人画成动物、把一个人画成两个人等等。另一方面,聊天机器人偶尔也会犯事实性错误。

生成人工智能仍处于早期阶段。接下来我们将介绍AIGC在不久的将来可能如何演变。

(1)更灵活的控制。AIGC任务的一个主要趋势是实现更灵活的控制。例如进行图像生成,早期基于GAN的模型可以生成高质量的图像,但几乎没有控制。最近的在大型文本图像数据上训练的扩散模型可以通过文本指令进行控制。这有利于生成更符合用户需求的图像。尽管如此,当前的文本到图像模型仍然需要更精细的控制,以便以更灵活的方式生成图像

(2)与微调有关。目前, ChatGPT等AIGC模型的开发主要集中在 pretraining阶段。相应的技术相对成熟;然而,如何微调这些模型是一个尚未开发的领域。不同于从零开始训练一个模型, 调优需要在基础模型原有的通用能力和它的适应性之间进行权衡。

(3)从大型科技公司到创业公司。目前, AIGC技术主要开发的是大型科技公司, 比如谷歌和Meta。在大型科技公司的支持下,一些初创公司开始崭露头角, 比如OpenAI(由微软支持)和DeepMind(由谷歌支持)。焦点开始从核心技术开发向应用转型。由于不断增长的需求,预计未来会出现更多的创业公司。

总体来说,AIGC是一个年轻的且充满前景的领域,我们十分期待在未来AIGC可以真正意义上改变我们的生活。

参考文献

- [1] G. Li, B. Chen, L. Zhu, Q. He, H. Fan, and S. Wang, “PUGCQ: A large scale dataset for quality assessment of professional user-generated content,” in *29th ACM International Conference on Multimedia*, 2021, pp. 3728–3736.
- [2] J. Kim, “The institutionalization of YouTube: From user-generated content to professionally generated content,” *Media, Culture & Society*, vol. 34, no. 1, pp. 53–67, 2012.
- [3] S. Gao, Y. Liu, Y. Kang, and F. Zhang, “User-generated content: A promising data source for urban informatics,” *Urban Informatics*, pp. 503–522, 2021.
- [4] D. J. Welbourne and W. J. Grant, “Science communication on YouTube: Factors that affect channel and video popularity,” *Public Understanding of Science*, vol. 25, no. 6, pp. 706–718, 2016.
- [5] Z. Li, J. Cai, S. He, and H. Zhao, “Seq2seq dependency parsing,” in *27th International Conference on Computational Linguistics*, 2018, pp. 3203–3214.
- [6] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” in *55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 1073–1083.
- [7] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient,” in *AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017, pp. 2852–2858.
- [8] H. Ren, H. Dai, Z. Dai, M. Yang, J. Leskovec, D. Schuurmans, and B. Dai, “Combiner: Full attention transformer with sparse computation cost,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 470–22 482, 2021.
- [9] E. Hoogeboom, A. A. Gritsenko, J. Bastings, B. Poole, R. v. d. Berg, and T. Salimans, “Autoregressive diffusion models,” in *10th International Conference on Learning Representations*, 2021, pp. 1–23.
- [10] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, “Training with quantization noise for extreme model compression,” in *9th International Conference on Learning Representations*, 2020, pp. 1–20.
- [11] M. Pollefeys and L. V. Gool, “From images to 3D models,” *Communications of the ACM*, vol. 45, no. 7, pp. 50–55, 2002.
- [12] S. Loeschcke, S. Belongie, and S. Benaim, “Text-driven stylization of video objects,” in *Computer Vision – ECCV Workshops*, 2022, pp. 594–609.
- [13] O. Texler, D. Futschik, M. Kucera, O. Jamriska, S. Sochorova, M. Chai, S. Tulyakov, and D. Sykora, “Interactive video stylization using few- shot patch-based training,” *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 73–1, 2020.
- [14] R. M. Joshi, S. Tao, P. Aaron, and B. Quiroz, “Cognitive component of componential model of reading applied to different orthographies,” *Journal of Learning Disabilities*, vol. 45, no. 5, pp. 480–486, 2012.

- [15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial net works," *Communications of the ACM*, vol. 63,no. 11, pp. 139-144, 2020.
- [16] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
- [17] Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. Nice: Non-linear independent components estimation. *ICLR 2015 Workshop Track* (2015).
- [18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, Vol. 33. 6840–6851.
- [19] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*. 8162–8171
- [20] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015, Deep unsupervised learning using nonequilibrium thermodynamics, In *International Conference on Machine Learning*. 2256–2265.
- [21] Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, Vol. 32.
- [22] Yang Song and Stefano Ermon. 2020. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, Vol. 33. 12438–12448.
- [23] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.
- [24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*.
- [25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).
- [26] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.
- [27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3D reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4460–4470.
- [28] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022).
- [29] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

- [30] Zekun Hao, Arun Mallya, Serge Belongie, Ming-Yu Liu. 2021. GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. *arXiv:2104.07659* (2021).
- [31] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, Jun Zhu. 2023. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. *arXiv:2305.16213*(2023).
- [32] Chaoning Zhang, Chenshuang Zhang, Sheng Zheng, Yu Qiao, Chenghao Li, Mengchun Zhang, Sumit Kumar Dam, Chu Myaet Thwal, Ye Lin Tun, Le Luang Huy, Donguk kim, Sung-Ho Bae, Lik-Hang Lee, Yang Yang, Heng Tao Shen, In So Kweon, Choong Seon Hong. A Complete Survey on Generative AI (AIGC): Is ChatGPT from GPT-4 to GPT-5 All You Need?. *arXiv:2303.11717*(2023).
- [33] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, Hong Lin. AI-Generated Content (AIGC): A Survey. *arXiv:2304.06632*(2023).