7
Dec

生成扩散模型漫谈（二十一）：中值定理加速ODE采样

By 苏剑林 | 2023-12-07 | 71766位读者 引用

在生成扩散模型的发展史上，DDIM和同期Song Yang的扩散SDE都称得上是里程碑式的工作，因为它们建立起了扩散模型与随机微分方程（SDE）、常微分方程（ODE）这两个数学领域的紧密联系，从而允许我们可以利用SDE、ODE已有的各种数学工具来对分析、求解和拓展扩散模型，比如后续大量的加速采样工作都以此为基础，可以说这打开了生成扩散模型的一个全新视角。

本文我们聚焦于ODE。在本系列的（六）、（十二）、（十四）、（十五）、（十七）等博客中，我们已经推导过ODE与扩散模型的联系，本文则对扩散ODE的采样加速做简单介绍，并重点介绍一种巧妙地利用“中值定理”思想的新颖采样加速方案“AMED”。

欧拉方法

正如前面所说，我们已经有多篇文章推导过扩散模型与ODE的联系，所以这里不重复介绍，而是直接将扩散ODE的采样定义为如下ODE的求解：

$$\frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \tag{1}$$

其中$t\in[0,T]$，初值条件是$\boldsymbol{x}_T$，要返回的结果是$\boldsymbol{x}_0$。原则上我们并不关心$t\in(0,1)$时的中间值$\boldsymbol{x}_t$，只需要最终的$\boldsymbol{x}_0$。为了数值求解，我们还需要选定节点$0=t_0 < t_1 < t_2 < \cdots < t_N = T$，常见的选择是

$$t_n = \left(t_1^{1/\rho} + \frac{n-1}{N-1}\left(t_N^{1/\rho} - t_1^{1/\rho}\right)\right)^{\rho} \tag{2}$$

其中$\rho > 0$。该形式来自《Elucidating the Design Space of Diffusion-Based Generative Models》（EDM），AMED也沿用了该方案，个人认为节点的选择不算关键要素，因此本文对此不做深究。

最简单的求解器是“欧拉方法”：利用差分近似

$$\left.\frac{d\boldsymbol{x}_t}{dt}\right|_{t=t_{n+1}} \approx \frac{\boldsymbol{x}_{t_{n+1}} - \boldsymbol{x}_{t_n}}{t_{n+1} - t_n} \tag{3}$$

我们可以得到

$$\boldsymbol{x}_{t_n} \approx \boldsymbol{x}_{t_{n+1}} - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})(t_{n+1} - t_n) \tag{4}$$

这通常也直接称为DDIM方法，因为是DDIM首先注意到它的采样过程对应于ODE的欧拉法，继而反推出对应的ODE。

高阶方法

从数值求解的角度来看，欧拉方法属于一阶近似，特点是简单快捷，缺点是精度差，所以步长不能太小，这意味着单纯利用欧拉法不大可能明显降低采样步数并且保证采样质量。因此，后续的采样加速工作都应用了更高阶的方法。

比如，直觉上差分$\frac{\boldsymbol{x}_{t_{n+1}} - \boldsymbol{x}_{t_n}}{t_{n+1} - t_n}$应该更接近中间点的导数而不是边界的导数，所以右端也换成t_n和t_{n+1}的平均应该会有更高的精度：

$$\frac{\boldsymbol{x}_{t_{n+1}} - \boldsymbol{x}_{t_n}}{t_{n+1} - t_n} \approx \frac{1}{2}\left[\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_n}, t_n) + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})\right] \tag{5}$$

由此我们可以得到

$$\boldsymbol{x}_{t_n} \approx \boldsymbol{x}_{t_{n+1}} - \frac{1}{2}\left[\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_n}, t_n) + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})\right](t_{n+1} - t_n) \tag{6}$$

然而，右端出现了$\boldsymbol{x}_{t_n}$，而我们要做的就是计算$\boldsymbol{x}_{t_n}$，所以这样的等式并不能直接用来迭代，为此，我们用欧拉方法“预估”一下$\boldsymbol{x}_{t_n}$，然后替换掉上式中的$\boldsymbol{x}_{t_n}$：

$$\begin{aligned}
\tilde{\boldsymbol{x}}_{t_n} &= \boldsymbol{x}_{t_{n+1}} - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})(t_{n+1} - t_n) \\
\boldsymbol{x}_{t_n} &\approx \boldsymbol{x}_{t_{n+1}} - \frac{1}{2}\left[\boldsymbol{v}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t_n}, t_n) + \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})\right](t_{n+1} - t_n)
\end{aligned} \tag{7}$$

这就是EDM所用的“Heun方法”，是一种二阶方法。这样每步迭代需要算两次$\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$，但精度明显提高，因此可以明显减少迭代步数，总的计算成本是降低的。

二阶方法还有很多变体，比如式(5)的右端我们可以直接换成中间点$t = (t_n + t_{n+1})/2$的函数值，这得到

$$\boldsymbol{x}_{t_n} \approx \boldsymbol{x}_{t_{n+1}} - \boldsymbol{v}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{(t_n+t_{n+1})/2}, \frac{t_n + t_{n+1}}{2}\right)(t_{n+1} - t_n) \tag{8}$$

中间点也有不同的求法，除了代数平均$(t_n + t_{n+1})/2$外，也可以考虑几何平均

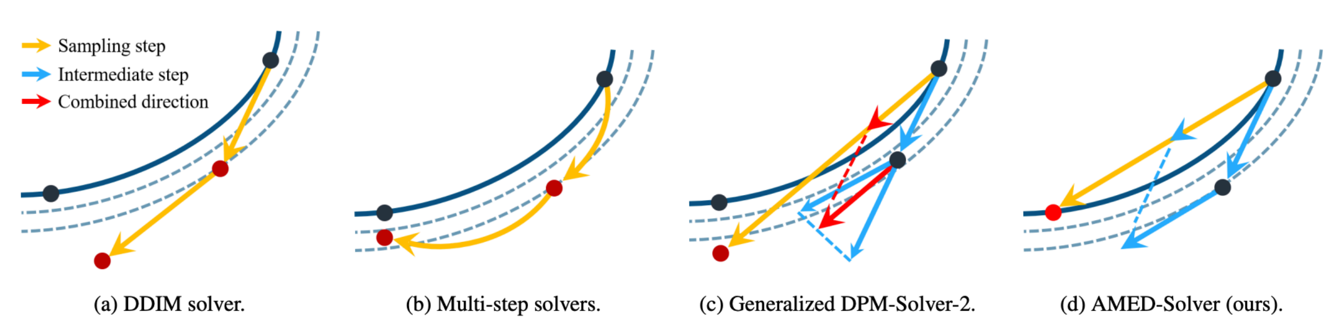
$$\boldsymbol{x}_{t_n} \approx \boldsymbol{x}_{t_{n+1}} - \boldsymbol{v}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{\sqrt{t_n t_{n+1}}}, \sqrt{t_n t_{n+1}}\right)(t_{n+1} - t_n) \tag{9}$$

事实上，式(9)就是DPM-Solver-2的一个特例。

除了二阶方法外，ODE的求解还有不少更高阶的方法，如"Runge-Kutta方法”、“线性多步法”等。然而，不管是二阶方法还是高阶方法，虽然都能一定程度上加速扩散ODE的采样，但由于这些都是“通法”，没有针对扩散模型的背景和形式进行定制，因此很难将采样过程的计算步数降到极致（个位数）。

中值定理

至此，本文的主角AMED登场了，其论文《Fast ODE-based Sampling for Diffusion Models in Around 5 Steps》前两天才放到Arxiv，可谓“新鲜滚热辣”。AMED并非像传统的ODE求解器那样一味提高理论精度，而是巧妙地类比了“中值定理”，并加上非常小的蒸馏成本，为扩散ODE**定制**了高速的求解器。



几种扩散ODE-Solver示意图

首先，我们对方程(1)两端积分，那么可以写出精确的等式：

$$\boldsymbol{x}_{t_{n+1}} - \boldsymbol{x}_{t_n} = \int_{t_n}^{t_{n+1}} \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)dt \tag{10}$$

如果$\boldsymbol{v}$只是一维的标量函数，那么由“积分中值定理”我们可以知道存在点$s_n \in (t_n, t_{n+1})$，使得

$$\int_{t_n}^{t_{n+1}} \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)dt = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{s_n}, s_n) \tag{11}$$

很遗憾，中值定理对一般的向量函数并不成立。不过，在$t_{n+1} - t_n$不太大以及一定的假设之下，我们依然可以类比地写出近似

$$\int_{t_n}^{t_{n+1}} \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)dt \approx \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{s_n}, s_n) \tag{12}$$

于是我们得到

$$\boldsymbol{x}_{t_n} \approx \boldsymbol{x}_{t_{n+1}} - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{s_n}, s_n)(t_{n+1} - t_n) \tag{13}$$

当然，目前还只是一个形式解，s_n和$\boldsymbol{x}_{s_n}$怎么来还未解决。对于$\boldsymbol{x}_{s_n}$，我们依然用欧拉方法进行预估，即$\tilde{\boldsymbol{x}}_{s_n} = \boldsymbol{x}_{t_{n+1}} - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})(t_{n+1} - s_n)$；对于$s_n$，我们则用一个小型的神经网络去估计它：

$$s_n = g_{\boldsymbol{\phi}}(\boldsymbol{h}_{t_{n+1}}, t_{n+1}) \tag{14}$$

其中$\boldsymbol{\phi}$是训练参数，$\boldsymbol{h}_{t_{n+1}}$是U-Net模型$\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_{n+1}}, t_{n+1})$的中间特征。最后，为了求解参数$\boldsymbol{\phi}$，我们采用蒸馏的思想，预先用步数更多的求解器求出精度更高的轨迹点对$(\boldsymbol{x}_{t_n}, \boldsymbol{x}_{t_{n+1}})$，然后最小化估计误差。这就是论文中的AMED-Solver（**A**pproximate **M**ean-**D**irection Solver），它具备常规ODE-Solver的形式，但又需要额外的蒸馏成本，然而这点蒸馏成本相比其他蒸馏加速方法又几乎可以忽略不计，所以笔者将它理解为“定制”求解器。

定制一词非常关键，扩散ODE的采样加速研究由来已久，在众多研究人员的贡献加成下，非训练的求解器大概已经走了非常远，但依然未能将采样步数降到极致，除非未来我们对扩散模型的理论理解有进一步的突破，否则笔者不认为非训练的求解器还有显著的提升空间。因此，AMED这种带有少量训练成本的加速度，既是“剑走偏锋”、“另辟蹊径”，也是“应运而生”、“顺理成章”。

实验结果

在看实验结果之前，我们首先了解一个名为“NFE”的概念，全称是“Number of Function Evaluations”，说白了就是模型$\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$的执行次数，它跟计算量直接挂钩。比如，一阶方法每步迭代的NFE是1，因为只需要执行一次$\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$，而二阶方法每一步迭代的NFE是2，AMED-Solver的$g_{\boldsymbol{\phi}}$计算量很小，可以忽略不计，所以AMED-Solver每一步的NFE也算是2。为了实现公平的比较，需要保持整个采样过程中总的NFE不变，来对比不同Solver的效果。

基本的实验结果是原论文的Table 2：

Method	NFE 3	NFE 5	NFE 7	NFE 9
Multi-step solvers				
DPM-Solver++(3M) [25]	110.0	24.97	6.74	3.42
UniPC [48]	156.3	22.32	6.46	3.47
iPNDM [22, 47]	47.98	13.59	5.08	3.17
Single-step solvers				
DDIM [40]	93.36	49.66	27.93	18.43
DPM-Solver-2 [24]	155.7	57.30	10.20	4.98
EDM [18]	306.2	97.67	37.28	15.76
AMED-Solver (ours)	23.65	17.94	5.40	3.59
AMED-Plugin (ours)	**17.74**	**7.14**	**3.96**	**2.78**

(a) Unconditional generation on CIFAR10 32×32. We show the results of AMED-Plugin applied on iPNDM.

Method	NFE 3	NFE 5	NFE 7	NFE 9
Multi-step solvers				
DPM-Solver++(3M) [25]	91.52	25.49	10.14	6.48
UniPC [48]	107.8	31.84	11.07	6.64
iPNDM [22, 47]	58.53	18.99	9.17	5.91
Single-step solvers				
DDIM [40]	82.96	43.81	27.46	19.27
DPM-Solver-2 [24]	140.2	42.41	12.03	6.64
EDM [18]	249.4	89.63	37.65	16.76
AMED-Solver (ours)	34.70	18.51	9.13	6.56
AMED-Plugin (ours)	**25.34**	**13.75**	**8.44**	**5.65**

(b) Conditional generation on ImageNet 64×64. We show the results of AMED-Plugin applied on iPNDM.

Method	NFE 3	NFE 5	NFE 7	NFE 9
Multi-step solvers				
DPM-Solver++(3M) [25]	86.45	22.51	8.44	4.77
UniPC [48]	126.4	17.89	7.07	**4.53**
iPNDM [22, 47]	45.98	17.17	7.79	4.58
Single-step solvers				
DPM-Solver-2 [24]	266.0	87.10	22.59	9.26
AMED-Solver (ours)	44.94	15.09	7.82	5.89
AMED-Plugin (ours)	**25.18**	**12.54**	**6.57**	4.65

(c) Unconditional generation on FFHQ 64×64. We show the results of AMED-Plugin applied on iPNDM.

Method	NFE 3	NFE 5	NFE 7	NFE 9
Multi-step solvers				
DPM-Solver++(3M) [25]	111.9	23.15	8.87	6.45
UniPC [48]	165.1	26.44	10.01	7.14
iPNDM [22, 47]	80.99	26.65	13.80	8.38
Single-step solvers				
DPM-Solver-2 [24]	210.6	80.60	23.25	9.61
AMED-Solver (ours)	**51.16**	**12.79**	**7.55**	**5.54**
AMED-Plugin (ours)	113.6	24.60	8.88	6.46

(d) Unconditional generation on LSUN Bedroom 256×256. We show the results of AMED-Plugin applied on DPM-Solver-2.

Table 2. Results of image generation. Our proposed AMED-Solver and AMED-Plugin achieve state-of-the-art results among solver-based methods in around 5 NFE.

AMED的实验结果（Table 2）

这个表格有几个值得特别留意的地方。第一，在NFE不超过5时，二阶的DPM-Solver、EDM效果还不如一阶的DDIM，这是因为Solver的误差不仅跟阶次有关，还跟步长$t_{n+1} - t_n$有关，大致上的关系就是$\mathcal{O}((t_{n+1} - t_n)^m)$，其中$m$就是“阶”，在总NFE较小时，高阶方法只能取较大的步长，所以实际精度反而更差，从而效果不佳；第二，同样是二阶方法的SMED-Solver，在小NFE时效果取得了全面SOTA，这充分体现了“定制”的重要性；第三，这里的“AMED-Plugin”是原论文提出的将AMED的思想作为其他ODESolver的“插件”的用法，细节更加复杂一些，但取得了更好的效果。

可能有读者会疑问：既然二阶方法每一步迭代都需要2个NFE，那么表格中怎么会出现奇数的NFE？其实，这是因为作者用到了一个名为“AFS（Analytical First Step）”的技巧来减少了1个NFE。该技巧出自《Genie: Higher-order denoising diffusion solvers》，具体是指在扩散模型背景下我们发现$\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_N}, t_N)$与$\boldsymbol{x}_{t_N}$非常接近（不同的扩散模型表现可能不大一样，但核心思想都是第一步可以直接解析求解），于是在采样的第一

步直接用$\boldsymbol{x}_{t_N}$替代$\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_N}, t_N)$，这就省了一个NFE。论文附录的Table 8、Table 9、Table 10也更详尽地评估了AFS对效果的影响，有兴趣的读者可以自行分析。

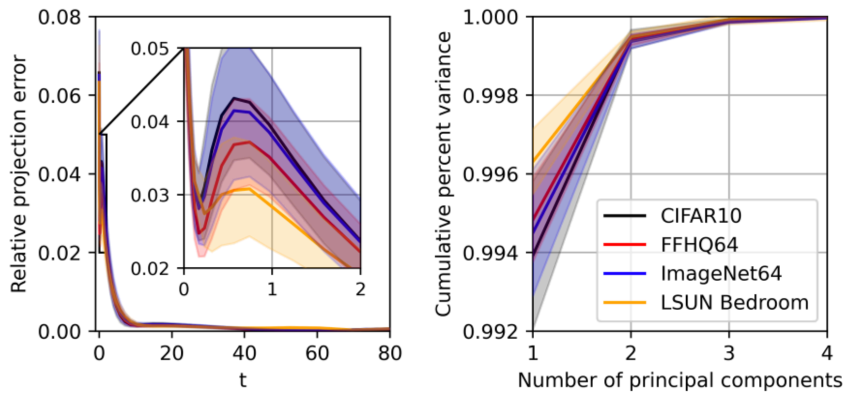
最后，由于AMED使用了蒸馏的方法来训练g_{ϕ}，那么也许会有读者想知道它跟其他蒸馏加速的方案的效果差异，不过很遗憾，论文没有提供相关对比。为此我也邮件咨询过作者，作者表示AMED的蒸馏成本是极低的，CIFAR10只需要在单张A100上训练不到20分钟，256大小的图片也只需要在4张A100上训练几个小时，而相比之下其他蒸馏加速的思路需要的时间是数天甚至数十天，因此作者将AMED视为Solver的工作而不是蒸馏的工作。不过作者也表示，后面有机会也尽可能补上跟蒸馏工作的对比。

假设分析

前面在讨论中值定理到向量函数的推广时，我们提到“一定的假设之下”，那么这里的假设是什么呢？是否真的成立呢？

不难举出反例证明，即便是二维函数积分中值定理都不恒成立，换言之积分中值定理只在一维函数上成立，这意味着如果高维函数成立积分中值定理，那么该函数所描述的空间轨迹只能是一条直线，也就是说采样过程中所有的$\boldsymbol{x}_{t_0}, \boldsymbol{x}_{t_1}, \cdots, \boldsymbol{x}_{t_N}$构成一条直线。这个假设自然非常强，实际上几乎不可能成立，但也侧面告诉我们，要想积分中值定理在高维空间尽可能成立，那么采样轨迹要保持在一个尽可能低维的子空间中。

为了验证这一点，论文作者加大了采样步数得到了较为精确的采样轨迹，然后对轨迹做主成分分析，结果如下图所示：



(a) Relative deviation. (b) Variance explained by top PCs.

Figure 4. We perform PCA to each sampling trajectory $\{x_t\}_{t=\epsilon}^{T}$. (a) These trajectories are projected into a 2D subspace spanned by the top 2 principal components to get $\{\tilde{x}_t\}_{t=\epsilon}^{T}$ and the relative projection error is calculated as $\|x_t - \tilde{x}_t\|_2 \,/\, \|x_t\|_2$. (b) We progressively increase the number of principal components and calculate the cumulative percent variance as $\mathrm{Var}(\{\tilde{x}_t\}_{t=\epsilon}^{T})/\mathrm{Var}(\{x_t\}_{t=\epsilon}^{T})$. The results are obtained by averaging 1k sampling trajectories using EDM sampler [18] with 80 NFE.

扩散ODE采样轨迹的主成分分析

主成分分析的结果显示，只保留top1的主成分，就可以保留轨迹的大部分精度，而同时保留前两个主成本，那么后面的误差几乎可以忽略了，这告诉我们采样轨迹几乎都集中在一个二维子平面上，甚至非常接近这个子平面上的的一个直线，于是在$t_{n+1} - t_n$并不是特别大的时候，扩散模型的高维空间的积分中值定理也近似成立。

这个结果可能会让人比较意外，但事后来看其实也能解释：在《生成扩散模型漫谈（十五）：构建ODE的一般步骤（中）》、《生成扩散模型漫谈（十七）：构建ODE的一般步骤（下）》我们介绍了先指定$\boldsymbol{x}_T$到$\boldsymbol{x}_0$的“伪轨迹”，然后再构建对应的扩散ODE的一般步骤，而实际应用中，我们所构建的“伪轨迹”都是$\boldsymbol{x}_T$与$\boldsymbol{x}_0$的线性插值（关于t可能是非线性的，关于$\boldsymbol{x}_T$和$\boldsymbol{x}_0$则是线性的），于是构建的“伪轨迹”都是直线，这会进一步鼓励真实的扩散轨迹是一条直线，这就解释了主成分分析的结果。

文章小结

本文简单回顾了扩散ODE的采样加速方法，并重点介绍了前两天刚发布的一个名为“AMED”的新颖加速采样方案，该Solver类比了积分中值定理来构建迭代格式，以极低的

蒸馏成本提高了Solver在低NFE时的表现。

如果您需要引用本文，请参考：

苏剑林. (Dec. 07, 2023). 《生成扩散模型漫谈（二十一）：中值定理加速ODE采样 》[Blog post]. Retrieved from https://spaces.ac.cn/archives/9881

```
@online{kexuefm-9881,
        title={生成扩散模型漫谈（二十一）：中值定理加速ODE采样},
        author={苏剑林},
        year={2023},
        month={Dec},
        url={\url{https://spaces.ac.cn/archives/9881}},
}
```