

# A Survey on Diffusion Model Through SDE

周添文

# Table of Contents

- Diffusion Model and SDE
- Discretizing Forward SDE
- Discretizing Reverse SDE
- Probability Flow ODE
- Accelerating Sampling Process

**生成式模型：**从给定的服从分布  $p(x)$  的训练数据中学习数据的概率分布  $p_{\theta}(x)$  (可能是隐式的)，并用这种分布来生成与训练数据相似的新样本。对于图像生成任务而言，我们希望模型学习到自然世界中真实图像的统计特征，例如边缘、纹理、颜色等，然后利用这些特征生成逼真的新的图像

常见的生成式模型有 GAN<sup>1</sup>，VAE<sup>2</sup>等，而 Diffusion Model(DM) 同样为一种生成式模型。其模拟物理学中的分子扩散模型，通过向训练集图像添加噪声来破坏训练数据 (Forward Process)，然后通过 (Training) 学习反转的去噪过程来恢复数据 (Reverse Process) 获得去噪模型，从而能够从噪音中采样生成出清晰图像 (Sampling Process)

---

<sup>1</sup>I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

<sup>2</sup>D. P. Kingma and M. Welling. Auto-encoding variational bayes. In ICLR, 2014

# Diffusion Model and SDE

Diffusion Model 的想法来自物理学中的分子扩散模型，而物理学中的扩散过程是一个关于时间连续的随机过程，其粒子的运动情况可以用一个随机微分方程 (Stochastic Differential Equation, SDE) 进行描述，即为

$$dx_t = f(x, t)dt + g(t)dw_t$$

对于图像处理问题而言， $x_t$  代表时刻  $t$  的图像， $f(x, t)$  称为 Drift Coefficient，代表期望的变化方向， $w_t$  表示一个标准 Gauss 噪音， $g(t)$  称为 Diffuse Coefficient，代表 Gauss 噪音的方差（尺度）。而文献 [1] 进一步证明了，在加噪过程中，Drift Coefficient 是线性的<sup>1</sup>，即上式可以改写为：

$$dx_t = f(t)x_t dt + g(t)dw_t \quad (1)$$

---

<sup>1</sup>D. P. Kingma, T. Salimans, B. Poole, and J. Ho, “Variational diffusion models,” in Advances in Neural Information Processing Systems, 2021.

我们的目的是输入一个原始图像  $x_0$ ，得到加噪后的图像  $x_t$ ，这一过程可以看做求解上述 SDE，即得到其从输入到输出的一条解轨迹。在求解上述 SDE 之前，我们先引入如下的概念。

信噪比 (SNR, Signal-to-Noise Ratio) 描述了包含图像真实信息的信号强度 (能量)  $\alpha_t$  和噪声信号强度 (能量)  $\sigma_t$  的比例关系，其定义为：

$$SNR(t) = \frac{\alpha_t^2}{\sigma_t^2}$$

其中  $\alpha_t$  可以看做是输入图像信号的 Amplitude， $\sigma_t^2$  则是噪音的方差。显然，随着时间  $t$  的推移，我们需要让加噪后的图像噪声权重越来越大，原始图像的权重越来越小，因此信噪比应该是一个单调减少的函数。

# SDE 参数的确定

事实上，我们的目的是让经过前向过程加噪后的图像  $q(x_t|x_0)$  可以满足 Gauss 分布  $N(\alpha_t x_0, \sigma_t^2 \mathbf{I})$ ，即  $q(x_t|x_0) = N(\alpha_t x_0, \sigma_t^2 \mathbf{I})$  (2) 因此，我们需要确定前向过程中的参数  $f(t), g(t)$ 。而最终得到的 Gauss 分布，正是由前面所定义的信噪比决定的  
我们断言，这两个参数的表达式为：<sup>1</sup>

$$f(t) = \frac{d \log \alpha_t}{dt}, g^2(t) = \frac{d \sigma_t^2}{dt} - 2 \frac{d \log \alpha_t}{dt} \sigma_t^2$$

我们下面给出证明：事实上，方程 (1) 可以写作如下的积分形式，即：

$$x_t = x_0 + \int_0^t f(s) x_s ds + \int_0^t g(s) dw_s \quad (3)$$

---

<sup>1</sup>Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022.

# SDE 参数的确定

我们对方程 (3) 两侧同时求期望，得：

$$\mathbb{E}[x_t] = \mathbb{E}[x_0] + \int_0^t \mathbb{E}[f(s)x_t] ds + 0 \quad (4)$$

其中，由于  $\int_0^t g(s)dw_t$  描述的是标准 Gauss 噪音项，因此其期望为 0  
方程 (4) 可以化为

$$\mathbb{E}[x_t] = \mathbb{E}[x_0] + \int_0^t \mathbb{E}[x_t]f(s) ds \quad (5)$$

同理，只需考虑  $\text{Var}[x_t] = \mathbb{E}[x_t^2] - (\mathbb{E}[x_t])^2$ ，即可得到

$$\text{Var}[x_t] = \sigma_t^2$$

至此，我们证明了  $f(t) = \frac{d \log \alpha_t}{dt}$ ,  $g^2(t) = \frac{d\sigma_t^2}{dt} - 2 \frac{d \log \alpha_t}{dt} \sigma_t^2$  的选取是合理的，其可以保证原始图像在加噪一段时间  $t$  后服从 Gauss 分布

$N(\alpha_t x_0, \sigma_t^2 \mathbf{I})$

# 求解 SDE 的数值方法

想要求随机微分方程的解，我们通常可以使用 Euler-Maruyama 法进行离散化求解，这一方法是由常微分方程中 Euler 折线法演变而来的。

- **Euler 折线法:** 对于初值问题:

$$\frac{dy}{dx} = f(x, y), a \leq x \leq b$$

$$y(a) = \eta$$

求数值解是指：取定一个步长  $h$ ，在一系列点

$\{x_0 = a, x_1 = a + h, x_2 = a + 2h, \dots\}$  上求  $y_i = y(x_i)$  而 Euler 法指的是，考虑在区间  $[x_n, x_{n+1}]$  上对  $y'(x) = f(x, y(x))$  求积分，并用矩形进行近似，有：

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \\ &\approx y(x_n) + hf(x_n, y(x_n)) \\ &= y_n + hf(x_n, y_n) \end{aligned}$$

给定一个初值，以此公式迭代求解即可。

# 求解 SDE 的数值方法

- **Euler-Maruyama 法**: Euler-Maruyama 法和 Euler 折线法类似，都是从一个初值出发，选定一个固定的时间步  $\Delta t \approx 0$  进行反复迭代即可。其将 SDE 转化为离散时间差分方程来逼近数值解，在每个时间步长上，Euler-Maruyama 法使用当前时间的状态和随机项（例如布朗运动）来估计下一个时间步长上的状态。因此，其离散化得到的是一个 Markov Chain。  
对于一个形如：

$$dx_t = a(x_t, t)dt + b(x_t, t)dw_t$$

的 SDE，Euler-Maruyama 法用一个 Markov Chain  $Y$  逼近方程的解  $X$

- 首先，将区间  $[0, T]$  分割为  $N$  份，步长为  $\Delta t > 0$   
 $0 = \tau_0 < \tau_1 < \dots < \tau_N = T$ ，且  $\Delta t = T/N$
- 赋初值  $Y_0 = x_0$
- 对于  $0 \leq n \leq N-1$ ，迭代计算  
 $Y_{n+1} = Y_n + a(Y_n, \tau_n)\Delta t + b(Y_n, \tau_n)\Delta w_n$   
其中  $\Delta W_n = W_{\tau_{n+1}} - W_{\tau_n}$

# VP-SDE 与 DDPM 的前向过程

上述选定好参数的 SDE 即为 Diffusion Model 的前向加噪过程 (Forward Process), 这一过程的目的是把原始图像转换成纯噪音图。一类特殊的 Forward Process SDE 称为 Variance Preserving SDE (VP-SDE)

<sup>1</sup> VP SDE 如下:

$$dx = \sqrt{1 - \sigma_t^2} x dt + \sqrt{\sigma_t^2} dw$$

由前面的系数公式可知, 其为选定了  $\alpha_t = \sqrt{1 - \sigma_t^2}$  得到的, 结果服从 Gauss 分布  $N(\sqrt{1 - \sigma_t^2} x_0, \sigma_t^2 \mathbf{I})$  这一 SDE 之所以称作 Variance-Preserving 的, 是因为由式 (5) 可知, 我们只要通过 Rescale 初始图像  $x_0$ , 保证  $\mathbb{E}[x_0] = 0$  且  $\mathbb{E}[x_0^2] = 1$ , 即可知  $\mathbb{E}[x_t] = 0$ ,  $\mathbb{E}[x_t^2] = 1$ , 进而知  $V[x_t] = 1$   
因此, 经过这一前向过程, 我们将原始图像  $x_0$  转化为  $x_T \sim N(0, 1)$

---

<sup>1</sup>Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in International Conference on Learning Representations, 2021

# VP-SDE 与 DDPM 的前向过程

连续的 SDE 能够更加本质地描述 Diffusion 的过程，但为了能够对方程进行求解，我们需要对上述 VP-SDE 进行离散化。

经 Euler-Maruyama 离散化后，上述 VP SDE 得到一个具有 Markov 性的随机过程：

$$x_i = \sqrt{1 - \sigma_t^2} x_{i-1} + \sqrt{\sigma_t^2} z_{i-1}$$

其中， $z_{i-1} \sim N(0, 1)$  由于  $\beta_i$  通常较小，因此由 Taylor's Formula，可以将上式化成：

$$x_i = -\frac{1}{2} \sigma_t^2 x_{i-1} + \sqrt{\sigma_t^2} z_{i-1}$$

这一离散化后的 Markov Chain 即为 DDPM<sup>1</sup> 的前向加噪过程

---

<sup>1</sup>J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6840–6851.

# VE-SDE 与 NCSN 的前向过程

同样地，NCSN<sup>1</sup>的前向加噪过程也可以视作一种特殊的随机微分方程，即 Variance Exploding SDE (VE)，经 Euler-Maruyama 离散化得到的 VE SDE 如下：

$$dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}} dw$$

事实上，其为选定了信噪比中图像信号强度  $\alpha_t^2 = 1$  所得到的结果。按照我们前文中对  $f(t), g(t)$  形式的推导，此时有：

$$f(t) = \frac{d \log \alpha_t}{dt} = 0, g^2(t) = \frac{d\sigma_t^2}{dt} - 2 \frac{d \log \alpha_t}{dt} \sigma_t^2 = \frac{d\sigma_t^2}{dt}$$

在这里，我们选取  $\sigma(t) = \sigma_{\min} \left( \frac{\sigma_{\max}}{\sigma_{\min}} \right)^t, t \in (0, 1]$ ，其中  $\sigma_{\max}, \sigma_{\min}$  为所添加噪音尺度的最大、最小值。由于  $\sigma(t)$  包含关于  $t$  的指数式，因此其称为 Variance Exploding SDE

<sup>1</sup>Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Advances in Neural Information Processing Systems, pp. 11895–11907, 2019

上式经 Euler-Maruyama 离散化后, 得到:

$$x_i = x_{i-1} + \sqrt{\frac{d(\sigma^2(t))}{dt}} \Delta t z_{i-1} = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} z_{i-1}$$

即为 NCSN 的前向过程

在 Sampling 过程中，我们的目的是生成数据，也就是从随机噪声  $x(T) \sim p(T)$  反向生成原始图像。这一过程与前面的 Forward Process 是相反的，因此也称为 Reverse Process。从 SDE 的角度来看，我们的目的就是要找到 Forward SDE 关于时间的逆过程。由文献<sup>1</sup>对每一个 SDE

$$dx_t = f(x, t)dt + g(t)dw_t$$

都有一个关于时间反向的 Reverse SDE，其式为

$$dx = [f(x, t) - g^2(t)\nabla_x \log p_t(x)]dt + g(t)d\bar{w}(6)$$

同样地， $\bar{w}$  也是一个标准的布朗运动其中， $\nabla_x \log p_t(x)$  称为 Score Function，是一个向量，其代表了图像分布的对数梯度方向。

---

<sup>1</sup>Brian D O Anderson. Reverse-time diffusion equation models. Stochastic Process. Appl., 12(3): 313–326, May 1982.

我们可以从物理模型的角度直观地理解上述 (6) 式, 对于现实世界的分子扩散过程, 其应该是顺着浓度梯度方向进行的。如果想找到扩散过程的逆过程, 理应逆着浓度梯度方向进行。而 SDE 的 Drift Coefficient 正是描述分布的期望变化的, 因此应在 Drift Coefficient 的基础上添加浓度梯度的相反方向。

由于在 Reverse Process 中, 我们是从纯噪音图像  $x_T$  出发的, 因此我们对真实图像的分布  $p_{data}(x)$  的梯度  $\nabla_x \log p_{data}(x)$  并不清楚, 从而没办法直接求出依赖真实图像分布的 Score Function。所以, 我们需要利用如下的 Score-Matching 算法<sup>1</sup>, 使用分步积分等初等积分方法, 对 Score Function 进行估计, 得到 Score-Based Models $_{\theta}(x, t)$ , 这也就是我们前文所说的 Training 过程

---

<sup>1</sup>Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019, pp. 204, 2019a.

# Training of Score-based Model

在训练 Score-based Model 时，我们的目标就是通过训练得到一个  $s_\theta(x, t)$  拟合真实分布的 Score-Function  $\nabla_x \log p_{data}(x)$ 。因此，我们可以用最小化 Fisher Divergence 作为训练目标。即目标为：

$$\hat{\theta} = \arg \min_{\theta} \mathbb{E}_{p_{data}} [\|\nabla_x \log p_{data}(x) - s_\theta(x, t)\|_2^2]$$

但是，输入数据的真实分布  $p_{data}(x)$  是未知的，因此仅凭这样的 Loss Function，我们无法对  $s_\theta(x, t)$  进行优化。在这里，我们应用 Score-Matching<sup>1</sup> 对上式进行化简，经过分部积分等操作，上述损失函数可以化简为

$$\begin{aligned} L &= \mathbb{E}_{p_{data}} [\|\nabla_x \log p_{data}(x) - s_\theta(x, t)\|_2^2] \\ &= 2\mathbb{E}_{p_{data}} [\nabla_x s_\theta(x, t) + \frac{1}{2} \|s_\theta(x, t)\|_2^2] + const \end{aligned}$$

---

<sup>1</sup>A. Hyvärinen. Estimation of Non-Normalized Statistical Models by Score Matching. Journal

# Training of Score-based Model

有了上述结论，我们在实际采样时便不再需要真实的数据分布  $p_{data}(x)$  在实际采样时，我们可以从输入数据中采样  $T$  个样本

$$x_1, x_2, \dots, x_T \sim p_{data}(x)$$

而采集了这些样本后，我们的损失函数  $\tilde{L}$  为（舍去常数 const）：

$$\tilde{L} = 2 \frac{1}{T} \sum_{i=1}^T [\nabla_x s_\theta(x_i, t) + \frac{1}{2} \|s_\theta(x_i, t)\|_2^2]$$

这一损失函数  $\tilde{L}$  是可以供我们对神经网络进行训练的（求使  $L$  最小的  $\theta$  即可）。而由大数定律， $\tilde{L}$  依概率收敛于  $L$ ，可以认为二者几乎必然 (almost sure) 相等，因此我们可以依此训练得到 Score-based Model  $s_\theta(x, t)$

# Equivalence of Different Training Objects

事实上，上述训练的目标函数  $\mathbb{E}_{p_{data}}[\|\nabla_x \log p_{data}(x) - s_\theta(x, t)\|_2^2]$  既可以看作是学习了一个 **Score-based Model**，也可以看作一个噪声估计模型 (Noise Prediction Model)  $\epsilon_\theta(x, t)$ ，还可以看作是一个用于估计当前图像所对应原始图像的模型  $x_\theta(x, t)$ ，我们下面证明上述三种模型的训练是等价的。

由上述损失函数  $L$  出发：

$$\begin{aligned} L &= \mathbb{E}_{p_{data}}[\|\nabla_x \log p_{data}(x, t) - s_\theta(x_t, t)\|_2^2] \\ &= \mathbb{E}_{p_{data}}\left[\frac{1}{\sigma_t^2} \|\sigma_t \nabla_x \log p_{data}(x, t) - \sigma_t s_\theta(x_t, t)\|_2^2\right] \end{aligned}$$

# Equivalence of Different Training Objects

而我们知道，经过 Forward Process 的图像满足  $p(x_t) = N(\alpha_t x_0, \sigma_t^2 \mathbf{I})$ ，因此  $x_t = \exp[-\frac{1}{2} \frac{(x_t - \alpha_t x_0)^2}{\sigma_t^2}]$ ，我们由此计算  $\nabla_x \log p_{data}(x, t)$

$$\nabla_x \log p_{data}(x, t) = -\frac{x_t - \alpha_t x_0}{\sigma_t^2}$$

而由于  $x_t = \alpha_t x_0 + \sigma_t \epsilon$ ，上式可以化为：

$$\nabla_x \log p_{data}(x, t) = -\frac{x_t - \alpha_t x_0}{\sigma_t^2} = -\frac{\epsilon(x_t, t)}{\sigma_t}$$

其中， $\epsilon(x_t, t)$  为当前图像中所注入的噪声的 Ground Truth 值  
因此，我们有

$$\begin{aligned} L &= \mathbb{E}_{p_{data}} \left[ \frac{1}{\sigma_t^2} \|\sigma_t \nabla_x \log p_{data}(x_t, t) - \sigma_t s_\theta(x_t, t)\|_2^2 \right] \\ &= \frac{1}{\sigma_t^2} \mathbb{E}_{p_{data}} \left[ \|\epsilon(x_t, t) - s_\theta(x_t, t)\|_2^2 \right] \end{aligned}$$

# Equivalence of Different Training Objects

因此，上式右端的  $-\sigma_t s_\theta(x_t, t)$  可以看作是对图像中噪声  $\epsilon(x_t, t)$  的估计模型，我们记其为  $\epsilon_\theta(x_t, t) := -\sigma_t s_\theta(x_t, t)$ ，这一模型正是 DDPM<sup>1</sup> 的训练目标。

除此以外，我们还可以对上述损失函数  $L$  进行如下变换，以此得到估计当前图像所对应原始图像模型  $x_\theta(x, t)$ ：

$$\begin{aligned} L &= \frac{1}{\sigma_t^2} \mathbb{E}_{p_{data}} [\| \epsilon_\theta(x_t, t) - \epsilon(x_t, t) \|_2^2] \\ &= \frac{1}{\sigma_t^2} \mathbb{E}_{p_{data}} [\| (x_t - \alpha_t x_0) - (x_t - \alpha_t x_\theta(x_t, t)) \|_2^2] \\ &= \frac{\alpha_t^2}{\sigma_t^2} \mathbb{E}_{p_{data}} [\| x_0 - x_\theta(x_t, t) \|_2^2] \end{aligned}$$

由此可见，通过这样得到的  $x_\theta(x, t)$  即为对原始图像  $x_0$  的估计。综上，我们证明了上述三种模型的训练是等价的。

<sup>1</sup>J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6840–6851.

训练好 Score-Based Models  $s_\theta(x, t)$  后，我们的目的是采样生成数据，也就是从随机噪音  $x(T) \sim p(T)$ ，通过求解 Reverse SDE 得到  $x(0)$ ，在这一求解的过程中，我们同样可以使用上述 Euler-Maruyama 离散法对 (6) 式进行处理。

经上述方法离散化后，我们得到：

$$x_i = x_{i+1} - f(x_{i+1}) + g_{i+1}^2 s_\theta(x_{i+1}, i+1) + g_{i+1} z_{i+1}$$

其中  $z_t \sim N(0, I)$

我们将前文中 DDPM 模型的系数代入 Reverse Process SDE, 得到

$$dx = -\frac{1}{2}\sigma_t^2(x_t - \nabla_{x_t} \log q_t(x_t))dt + \sqrt{\sigma_t^2}d\bar{w}$$

经 Euler-Maruyama 离散化后, 得到:

$$\begin{aligned}x_{t-1} &= x_t + \frac{1}{2}\sigma_t^2(x_t - s_\theta(x_t, t)) + \sigma_t^2 s_\theta(x_t, t) + \sqrt{\sigma_t^2}z_t \\ &= (1 + \frac{1}{2}\sigma_t^2)x_t + \frac{1}{2}\sigma_t^2 s_\theta(x_t, t) + \sqrt{\sigma_t^2}z_t\end{aligned}$$

此即为 DDPM 的采样过程, 即 Ancestral Sampling<sup>1</sup>

---

<sup>1</sup>J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models” in Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 6840–6851. 

同样地，对于 NCSN 而言，我们代入前文确定的系数后，其 Reverse Process 的 SDE 如下：

$$dx = -\frac{d(\sigma_t^2)}{dt} s_\theta(x, t) dt + \frac{d(\sigma_t^2)}{dt} d\bar{w}$$

经 Euler-Maruyama 离散化后，得到：

$$x_{t-1} = x_t + \frac{d(\sigma_t^2)}{dt} s_\theta(x, t) + \sqrt{\frac{d(\sigma_t^2)}{dt}} z_t$$

即为 NCSN 中的退火 Langevin 采样过程<sup>1</sup>

---

<sup>1</sup>Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Advances in Neural Information Processing Systems, pp. 11895–11907, 2019

在 Sampling 过程中，我们想要从纯噪音  $x_T$  得到原始图像  $x_0$ ，则需要用离散化的数值解方法求解上述 Reverse Process SDE。但由于 SDE 具有随机性，因此往往不能用较大的步长进行离散化，导致采样一次需要迭代 1000~2000 步，效率较低。因此，我们可以采用下面的 Probability Flow ODE<sup>1</sup>对采样过程进行优化，以此减少单次采样所需的步数。对于一个一般的 SDE，其描述的是粒子的随机运动，单独的粒子具有随机性，而其对应的分布则是稳定的，可以用一个 PDE，即 **Fokker-Planck** 方程进行描述<sup>2</sup>

---

<sup>1</sup>Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in International Conference on Learning Representations, 2021.

<sup>2</sup>Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker-planck equations through gradient-log-density estimation. arXiv preprint arXiv:2006.00702, 2020.

# Probability Flow ODE

对于一个一般的 SDE

$$dx = f(x, t)dt + g(t)dw$$

其对应的 Fokker-Planck 方程是：

$$\frac{\partial p(x, t)}{\partial t} = -\sum_i \frac{\partial}{\partial x_i} [f_i(x, t)p(x, t)] + \frac{1}{2} \sum_i \frac{\partial^2}{\partial x_i^2} [g^2(t)p(x, t)] \quad (7)$$

对上式 (7) 进行等价变换，得到：

$$\begin{aligned} \frac{\partial p(x, t)}{\partial t} &= -\sum_i \frac{\partial}{\partial x_i} [f_i(x, t)p(x, t)] + \frac{1}{2} \sum_i \frac{\partial^2}{\partial x_i^2} [(g^2(t) - \sigma^2(t))p(x, t)] \\ &\quad + \frac{1}{2} \sum_i \frac{\partial^2}{\partial x_i^2} [\sigma^2(t)p(x, t)] \\ &= -\sum_i \frac{\partial}{\partial x_i} [f_i(x, t)p(x, t)] + \frac{1}{2} \sum_i \frac{\partial}{\partial x_i} [(g^2(t) - \sigma^2(t)) \frac{\partial}{\partial x_i} p(x, t)] \\ &\quad + \frac{1}{2} \sum_i \frac{\partial^2}{\partial x_i^2} [\sigma^2(t)p(x, t)] \end{aligned}$$

做等价替换

$$\frac{\partial}{\partial x_i} p(x, t) = p(x, t) \frac{\partial}{\partial x_i} \log p(x, t)$$

原式

$$\begin{aligned} &= -\sum_i \frac{\partial}{\partial x_i} [f_i(x, t) p(x, t)] + \frac{1}{2} \sum_i \frac{\partial}{\partial x_i} [(g^2(t) - \sigma^2(t)) p(x, t) \frac{\partial}{\partial x_i} \log p(x, t)] \\ &+ \frac{1}{2} \sum_i \frac{\partial^2}{\partial x_i^2} [\sigma^2(t) p(x, t)] \\ &= -\sum_i \frac{\partial}{\partial x_i} [(f_i(x, t) - \frac{1}{2}(g^2(t) - \sigma^2(t)) \frac{\partial}{\partial x_i} \log p(x, t)) p(x, t)] \\ &+ \frac{1}{2} \sum_i \frac{\partial^2}{\partial x_i^2} [\sigma^2(t) p(x, t)] \quad (8) \end{aligned}$$

而按照 Ito Process 和 Fokker-Planck 方程的对应关系, (8) 式对应着如下 Ito Process

$$dx = (f(x, t) - \frac{1}{2}(g^2(t) - \sigma^2(t))\nabla_x \log p(x, t))dt + \sigma(t)dw$$

由于 Fokker-Planck 方程进行的是等价变换, 因此上述 Ito Process 在任意  $\sigma^2$  下与

$$dx = f(x, t)dt + g(t)dw$$

在各个时刻的边缘分布  $p(x_t)$  是一致的  
因此, 取  $\sigma^2 = 0$  即可得到如下的 ODE:

$$dx = (f(x, t) - \frac{1}{2}g^2(t)\nabla_x \log p(x, t))dt \quad (9)$$

其仍然与 (8) 式有着相同的边缘分布。(9) 的解，是一个确定性的轨迹，其从原始图像分布  $p_0(x)$  指向纯噪音分布  $p_T(x)$ ，且是可逆的。这其实与 Flow-based Model 的思想是一致的（即通过一个可逆变换将噪声变成样本），是一种 Continuous Normalizing Flow<sup>1</sup>，因此也称为 Probability Flow 由于 (9) 的解是可逆的，因此其既可以用于描述前向传播过程，也可以用于描述 **Reverse Process**<sup>2</sup>，其也是前文中我们提到过的 Reverse-time SDE

$$dx = [f(x, t) - g^2(t)\nabla_x \log p_t(x)]dt + g(t)d\bar{w}$$

对应的 Probability Flow ODE

---

<sup>1</sup>Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9916–9926, 2019.

<sup>2</sup>Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022.

# Semi-Linear Structure

对于上面的 (9) 式, 我们可以用首先用前文中提到的 Noise Prediction Model 替换 Score-based Model

$$\nabla_x \log p(x, t) dt \approx s_\theta(x, t) = -\frac{1}{2\sigma_t} \epsilon_\theta(x_t, t)$$

可以将 (9) 式变为:

$$\frac{dx_t}{dt} = h_\theta(x, t) := f(t)x_t + \frac{g^2(t)}{2\sigma_t} \epsilon_\theta(x_t, t) \quad (10)$$

可以观察到, 式 (10) 中包括一个线性部分  $f(t)x_t$ , 为关于  $x_t$  的线性方程; 而后面的  $\frac{g^2(t)}{2\sigma_t} \epsilon_\theta(x_t, t)$  由于神经网络的存在, 是非线性的。因此, 我们称式 (10) 为 Semi-Linear ODE.

想要求解上式 (9), 我们先用常数变易法将其转化成积分的形式, 这一步的计算是精确的:

$$x_t = e^{\int_s^t f(\tau) d\tau} x_s + \int_s^t (e^{\int_\tau^t f(r) dr} \frac{g^2(\tau)}{2\sigma_\tau} \epsilon_\theta(x_\tau, \tau)) d\tau \quad (11)$$

在求解上述式 (10) 时，我们当然可以选择常规的求 ODE 数值解的方法，如 Euler 法或 Runge-Kutta 法，首先将式 (10) 写成积分的形式：

$$x_t = x_s + \int_s^t h_\theta(x_\tau, \tau) d\tau = x_s + \int_s^t (f(\tau)x_\tau + \frac{g^2(\tau)}{2\sigma_\tau} \epsilon_\theta(x_\tau, \tau)) d\tau \quad (12)$$

在 Song et al.2021<sup>1</sup>中，作者就是使用 Runge-Kutta 法<sup>2</sup>估计式 (12) 中积分部分的值，进行求数值解。

这种方法之所以称为 Black-Box 方法，是因为其在求解时并未考虑方程半线性性的结构特点，将整体  $h_\theta(x_t, t)$  作为输入项。但是从 (11) 式可以发现，在实际的解中，线性项  $f(t)x_t$  的系数是指数次的，因此只能用较小的步长，较多的步数进行多次迭代近似，否则这一项带来的误差将会过大，因此这种 Black-Box 模型的效率并不高。

<sup>1</sup>Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” in International Conference on Learning Representations, 2021.

<sup>2</sup>Peter E Kloeden and Eckhard Platen. Numerical solution of stochastic differential equations, volume 23. Springer Science & Business Media, 2013.

# Observations of Probability Flow ODE

- 首先，由于我们在式 (11) 中分离出了含有神经网络  $\epsilon_\theta(x, t)$  的部分，式 (11) 右侧的前半部分  $e^{\int_s^t f(\tau) d\tau} x_s$  是不含有估计项的，而  $f(t)$  的表达式我们已经给出，因而可以精确地计算这一部分 (Observation 1)
- 其次，我们进一步处理式 (11) 的非线性部分（即包含神经网络的部分），令  $\lambda_t := \log(\alpha_t/\sigma_t)$ ，回顾上文中关于信噪比的定义，我们可以发现  $\lambda_t$  就是信噪比对数的一半。
- 由前文所述，信噪比是一个关于时间  $t$  单调减少的函数，因此  $\lambda_t := \log(\alpha_t/\sigma_t)$  也应随着时间  $t$  的增大而减少，进而由反函数定理知，时间  $t$  必拥有关于  $\lambda(t)$  的反函数  $t = t_\lambda(\lambda(t))$

因此，我们可以将  $g^2(t)$  改写为：

$$g^2(t) = \frac{d\sigma_t^2}{dt} - 2\frac{d\log \alpha_t}{dt}\sigma_t^2 = 2\sigma_t^2\left(\frac{d\log \sigma_t}{dt} - \frac{d\log \alpha_t}{dt}\right) = -2\sigma_t^2\frac{d\lambda_t}{dt} \quad (13)$$

# Observations of Probability Flow ODE

考虑到

$$f(t) = \frac{d \log \alpha_t}{dt}$$

我们将上式与式 (13) 带入式 (11) 即可得到:

$$x_t = \frac{\alpha_t}{\alpha_s} x_s - \alpha_t \int_s^t \left( \frac{d\lambda_\tau}{d\tau} \right) \frac{\sigma_\tau}{\alpha_\tau} \epsilon_\theta(x_\tau, \tau) d\tau \quad (14)$$

同时, 由于时间  $t$  拥有关于  $\lambda(t)$  的反函数  $t = t_\lambda(\lambda(t))$ , 因此我们可以通过换元法, 令  $\hat{x}_\lambda := x_{t_\lambda(\lambda(t))}$ ,  $\hat{\epsilon}_\theta(\hat{x}_\lambda(t), \lambda(t)) := \epsilon_\theta(x_{t_\lambda(\lambda(t))}, t_\lambda(\lambda(t)))$ , 且回忆  $\lambda(t) = \log \frac{\sigma_t}{\alpha_t}$ , 经过换元后可以得到 (**Observation 2**):

$$x_t = \frac{\alpha_t}{\alpha_s} x_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_\theta(\hat{x}_\lambda(t), \lambda(t)) d\lambda(t) \quad (15)$$

此时, 式 (15) 从对  $t$  积分改为了对  $\lambda$  积分, 这样的操作是为了能够得到  $\int_{\lambda_s}^{\lambda_t} e^{-\lambda} \hat{\epsilon}_\theta(\hat{x}_\lambda(t), \lambda(t)) d\lambda(t)$  这一积分形式, 我们称其为对  $\hat{\epsilon}_\theta$  的**指数加权积分** (Exponentially Weighted Integral)

# Observations of Probability Flow ODE

为了求解上述式 (15) 的数值解, 我们在时间  $T$  输入初始的噪音值  $x_T$ , 并用  $M+1$  时间步  $\{t_i\}_{i=0}^M$  从  $t_0 = T$  到  $t_M = 0$  进行迭代, 最后一次迭代的  $\hat{x}_{t_M}$  作为  $t = 0$  时刻的预测值, 也就是预测的原始图像。从第  $i-1$  步到第  $i$  步过程中, 解轨迹的真值为: (注意, 到目前为止我们所表示的都是真值, 但我们后续需要用各种数值方法去拟合这一真值)

$x_{t_{i-1} \rightarrow t_i}$  代表解轨迹, 则有

$$x_{t_{i-1} \rightarrow t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \hat{x}_{t_{i-1}} - \alpha_{t_i} \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \hat{\epsilon}_\theta(\hat{x}_\lambda, \lambda) d\lambda \quad (16)$$

为了能够近似求解 (16) 式得到  $\hat{x}_{t_i}$ , 我们需要近似计算的是 (16) 式的后半部分, 即  $\hat{\epsilon}_\theta$  从  $\lambda_{t_{i-1}}$  到  $\lambda_{t_i}$  的指数加权积分  $\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \hat{\epsilon}_\theta(\hat{x}_\lambda, \lambda) d\lambda$

# Observations of Probability Flow ODE

我们记

$$h_i := \lambda_{t_i} - \lambda_{t_{i-1}}$$

及  $n$  阶导

$$\hat{\epsilon}_\theta^{(n)}(\hat{x}_\lambda, \lambda) := \frac{d^n \hat{\epsilon}_\theta(\hat{x}_\lambda, \lambda)}{d\lambda^n}$$

由 Taylor 公式, 有:

$$\hat{\epsilon}_\theta(\hat{x}_\lambda, \lambda) = \sum_{n=0}^{k-1} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} \hat{\epsilon}_\theta^{(n)}(\hat{x}_{\lambda_{i-1}}, \lambda_{t_{i-1}}) + O((\lambda - \lambda_{t_{i-1}})^k) \quad (17)$$

将 (17) 式代入 (16) 式, 有:

$$\begin{aligned} x_{t_{i-1} \rightarrow t_i} &= \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \hat{x}_{t_{i-1}} \\ &- \alpha_{t_i} \sum_{n=0}^{k-1} \hat{\epsilon}_\theta^{(n)}(\hat{x}_{\lambda_{i-1}}, \lambda_{t_{i-1}}) \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} d\lambda + O(h_i^{k+1}) \quad (18) \end{aligned}$$

# Observations of Probability Flow ODE

对于上面的 (18) 式, 右侧的  $\frac{\alpha_{t_j}}{\alpha_{t_{j-1}}} \hat{x}_{t_{j-1}}$  可以精确计算, 而积分号下的

$\int_{\lambda_{t_{j-1}}}^{\lambda_{t_j}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{j-1}})^n}{n!} d\lambda$  可以由  $n$  次分步积分进行精确计算

(Observation3), 即:

$$\begin{aligned} \int_{\lambda_{t_{j-1}}}^{\lambda_{t_j}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{j-1}})^n}{n!} d\lambda &= - \int_{\lambda_{t_{j-1}}}^{\lambda_{t_j}} \frac{(\lambda - \lambda_{t_{j-1}})^n}{n!} d(e^{-\lambda}) \\ &= \left( -\frac{(\lambda - \lambda_{t_{j-1}})^n}{n!} e^{-\lambda} \right) \Big|_{\lambda_{t_{j-1}}}^{\lambda_{t_j}} + \int_{\lambda_{t_{j-1}}}^{\lambda_{t_j}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{j-1}})^{n-1}}{(n-1)!} d\lambda \\ &= \dots \end{aligned}$$

因此, 整个 (18) 式中, 我们需要近似的只有  $\hat{\epsilon}_{\theta}^{(n)}(\hat{x}_{\lambda_{t_{j-1}}}, \lambda_{t_{j-1}})$ , 而在对其进行计算的时候, 我们可以用一些传统的数值方法 (比如微分中值定理等等), 并不需要真正地进行求导运算 (Observation4)。比如, 在  $n = 1$  时, 由微分中值定理, 有

$$\hat{\epsilon}_{\theta}^{(1)}(\hat{x}_{\lambda_{t_{j-1}}}, \lambda_{t_{j-1}}) \approx \frac{\hat{\epsilon}_{\theta}(\hat{x}_{s_i}, s_i) - \hat{\epsilon}_{\theta}(\hat{x}_{\lambda_{t_{j-1}}}, \lambda_{t_{j-1}})}{s_i - \lambda_{t_{j-1}}}$$

# Observations of Probability Flow ODE

综上所述，DPM-Solver 基于以上 4 个 Observation，做到了尽可能地准确计算所有已知项，只对神经网络部分做近似，因此最大程度地减小了离散化带来的误差，进而可以用更大的步长去进行迭代，以此加快采样的速率

$$\mathbf{x}_{t_{i-1} \rightarrow t_i} = \frac{\alpha_{t_i} \tilde{\mathbf{x}}_{t_{i-1}}}{\alpha_{t_{i-1}}} - \alpha_{t_i} \sum_{n=0}^{k-1} \hat{\epsilon}_{\theta}^{(n)}(\hat{\mathbf{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}) \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} \frac{(\lambda - \lambda_{t_{i-1}})^n}{n!} d\lambda + \mathcal{O}(h_i^{k+1})$$

<b>Linear term</b>	<b>Derivatives</b>	<b>Coefficients</b>	<b>High-order errors</b>
<b>Exactly Computed</b>	<b>Approximated</b>	<b>Exactly computed</b>	<b>Omitted</b>
(Observation 1)	(Observation 4)	(Observation 2 & 3)	

当  $k = 1$  时，我们重写上述式子，可以发现其为：

$$x_{t_{i-1} \rightarrow t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \hat{x}_{t_{i-1}} - \alpha_{t_i} \epsilon_{\theta}(\hat{x}_{\lambda_{i-1}}, \lambda_{t_{i-1}}) \int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda} d\lambda + O((\lambda_{t_i} - \lambda_{t_{i-1}})^2) \quad (19)$$

回忆  $\lambda$  的定义， $\lambda_t := \log(\alpha_t / \sigma_t)$ ，因此  $\frac{\sigma_{t_{i-1}}}{\alpha_{t_{i-1}}} = e^{-\lambda_{t_{i-1}}}$ ，故舍弃高阶项后，(19) 式可以的估计值可以写为：

$$\hat{x}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}} \hat{x}_{t_{i-1}} - \alpha_{t_i} \left( \frac{\sigma_{t_i}}{\alpha_{t_i}} - \frac{\sigma_{t_{i-1}}}{\alpha_{t_{i-1}}} \right) \epsilon_{\theta}(\hat{x}_{t_{i-1}}, t_{i-1})$$

而 DDIM 的 Sampling 过程则是下式 (20) 在方差  $\sigma = 0$  时的特例:

$$x_{t-1} = \sqrt{\alpha_{t-1}} \left( \frac{x_t - \sqrt{1 - \alpha_t} \epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \epsilon_\theta(x_t, t) + \sigma_t \epsilon_t(20)$$

即:

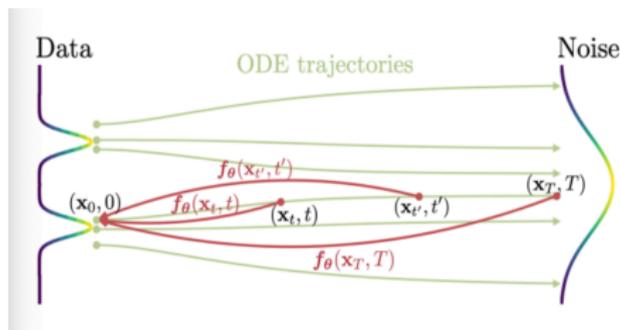
$$x_{t-1} = \sqrt{\alpha_{t-1}} \left( \frac{x_t - \sqrt{1 - \alpha_t} \epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1}} \epsilon_\theta(x_t, t)(21)$$

可以看出, 只需令  $\sigma_t = 1 - \alpha_t$ , 二者即为等价的形式, 这也证明了 DDIM 是 DPM-Solver 的一阶解, 因而同样可以加速 Sampling 过程

# Consistency Model

总的来说，DPM-Solver 是在传统的 Black-Box 模型基础上，考虑了所求数值解微分方程的结构，最大程度地减小了离散化误差。因此只要噪音模型训练的足够好，就可以用更大的步长进行迭代计算。

而 Consistency Model 则是直接学习 Probability Flow ODE 的解函数，由于 Probability Flow ODE 是一个确定性的轨迹，其同一条轨迹上任意一个点应该都有相同的起终点，因此称其为 Consistency Model<sup>1</sup>



<sup>1</sup>Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, “Con-sistency models,” ArXiv, vol. abs/2303.01469, 2023.3.1.1

# Consistency Model

Consistency Model 的是用一个神经网络  $f_\theta(x, t), t \in (\epsilon, T]$  去学习上述 Probability Flow ODE 的解函数  $f(x, t)$ , 其应为可优化学习的神经网络。由于这一解函数是一条由原始图像  $x_0$  指向噪音图像  $x_T$  的轨迹, 因此其应满足边界条件

$$f(x_\epsilon, \epsilon) = x_\epsilon, (\epsilon \rightarrow 0)$$

在训练这一 Consistency Model 的时候, 可用其“一致性”进行监督, 即保证同一个 ODE 解轨迹上的两个样本对  $(\hat{x}_t, x_t)$  的函数值是一样的。其中,  $\hat{x}_t$  是由 Euler 法等 Solver 对 Probability Flow ODE 进行离散化估计得到的,  $x_t$  则是由前向加噪过程的 SDE 准确计算得到的。因此, 我们的损失函数  $L$  为:

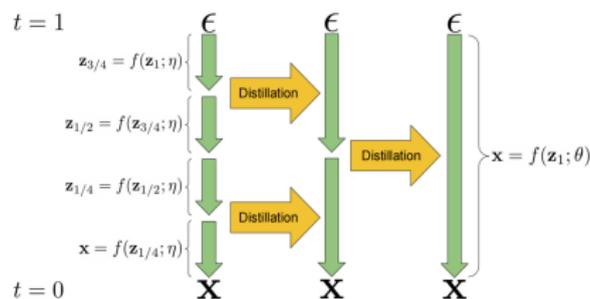
$$L = \lambda(t) \mathbb{E}[|f_\theta(x_t, t) - f_\theta(\hat{x}_t, t)|]$$

其中  $\lambda(t)$  为一个权重参数

这样训练得到的  $f_\theta(x, t)$  即为对 Probability Flow ODE 解函数的估计, 理论上讲, 我们只需输入解轨迹上的任意一个图像, 便可沿着解轨迹一步找到该图像对应的噪音图或者原始图像。

# Progressive Distillation

渐进式蒸馏 (Progressive Distillation)<sup>1</sup>是另一种加快 Sampling Process 的方法，其目标是将一个步骤很多的教师扩散模型蒸馏为一个步骤较少的学生模型，从而提升采样的速度，这一过程一般通过反复迭代的方式进行



在这部分，我们与原文一致，用  $z_t$  表示第  $t$  步加噪后的图像，用  $x$  表示未被噪音污染的原始图像

<sup>1</sup>Progressive distillation for fast sampling of diffusion models, Tim Salimans & Jonathan Ho



训练得到教师模型  $\hat{x}_\eta(z_t)$  后，我们进一步蒸馏得到学生模型。渐进蒸馏算法与标准扩散模型训练方法的主要区别在于如何确定去噪模型的目标值。在标准扩散训练中，去噪的目标是干净的数据，而在渐进式蒸馏中，学生去噪模型需要预测的目标是教师模型采样后的输出。

取  $t' = t - \frac{0.5}{N}, t'' = t - \frac{1}{N}$  为前后两个时间步，由 DDIM 的采样步骤可知，给定第  $t$  步的被噪音污染的图像  $z_t$ ，我们可以采样出  $t', t''$  步的图像：

$$z_{t'} = \alpha_{t'} \hat{x}_\eta(z_t) + \frac{\sigma_{t'}}{\sigma_t} (z_t - \alpha_t \hat{x}_\eta(z_t))$$

$$z_{t''} = \alpha_{t''} \hat{x}_\eta(z_t) + \frac{\sigma_{t''}}{\sigma_t} (z_t - \alpha_t \hat{x}_\eta(z_t))$$

# Progressive Distillation

而学生模型  $\tilde{x}$  同样是对尝试通过一步 DDIM 采样得到  $z_{t''}$ , 用  $\tilde{x}$  表示学生模型的预测结果, 用  $\tilde{z}_{t''}$  表示其一步采样的样本, 同样应用 DDIM 可得:

$$\tilde{z}_{t''} = \alpha_{t''} \tilde{x} + \frac{\sigma_{t''}}{\sigma_t} (z_t - \alpha_t \tilde{x})$$

在蒸馏学生模型时, 我们核心需要保证的就是其一步 DDIM 得到的结果与教师模型两步得到的相同, 因此我们需要限制  $\tilde{z}_{t''} = z_{t''}$

$$\begin{aligned}\tilde{z}_{t''} &= \alpha_{t''} \tilde{x} + \frac{\sigma_{t''}}{\sigma_t} (z_t - \alpha_t \tilde{x}) = z_{t''} \\ &= \left( \alpha_{t''} - \frac{\sigma_{t''}}{\sigma_t} \alpha_t \right) \tilde{x} + \frac{\sigma_{t''}}{\sigma_t} z_t = z_{t''} \\ \left( \alpha_{t''} - \frac{\sigma_{t''}}{\sigma_t} \alpha_t \right) \tilde{x} &= z_{t''} - \frac{\sigma_{t''}}{\sigma_t} z_t \\ \tilde{x} &= \frac{z_{t''} - \frac{\sigma_{t''}}{\sigma_t} z_t}{\alpha_{t''} - \frac{\sigma_{t''}}{\sigma_t} \alpha_t}\end{aligned}$$

# Progressive Distillation

因此，只要让  $\hat{x}_\theta(z_t)$  足够接近上述  $\tilde{x}$ ，我们则可以直接用学生模型  $\hat{x}_\theta(z_t)$  一步的训练结果来代替教师模型两步的训练结果了。因此，我们的损失函数为

$$L_\theta = w(\lambda) \|\tilde{x} - \hat{x}_\theta(z_t)\|$$

## Algorithm 1 Standard diffusion training

**Require:** Model  $\hat{x}_\theta(z_t)$  to be trained  
**Require:** Data set  $\mathcal{D}$   
**Require:** Loss weight function  $w(\cdot)$

**while** not converged **do**

$\mathbf{x} \sim \mathcal{D}$   $\triangleright$  Sample data  
 $t \sim U[0, 1]$   $\triangleright$  Sample time  
 $\epsilon \sim N(0, I)$   $\triangleright$  Sample noise  
 $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$   $\triangleright$  Add noise to data

$\tilde{\mathbf{x}} = \mathbf{x}$   $\triangleright$  Clean data is target for  $\hat{\mathbf{x}}$   
 $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$   $\triangleright$  log-SNR  
 $L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$   $\triangleright$  Loss  
 $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$   $\triangleright$  Optimization

**end while**

## Algorithm 2 Progressive distillation

**Require:** Trained teacher model  $\hat{x}_\eta(z_t)$   
**Require:** Data set  $\mathcal{D}$   
**Require:** Loss weight function  $w(\cdot)$   
**Require:** Student sampling steps  $N$

**for**  $K$  iterations **do**

$\theta \leftarrow \eta$   $\triangleright$  Init student from teacher

**while** not converged **do**

$\mathbf{x} \sim \mathcal{D}$   
 $t = i/N, i \sim \text{Cat}[1, 2, \dots, N]$   
 $\epsilon \sim N(0, I)$   
 $\mathbf{z}_t = \alpha_t \mathbf{x} + \sigma_t \epsilon$   
**# 2 steps of DDIM with teacher**  
 $t' = t - 0.5/N, t'' = t - 1/N$   
 $\mathbf{z}_{t'} = \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_t) + \frac{\sigma_{t'}}{\sigma_t} (\mathbf{z}_t - \alpha_t \hat{\mathbf{x}}_\eta(\mathbf{z}_t))$   
 $\mathbf{z}_{t''} = \alpha_{t''} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}) + \frac{\sigma_{t''}}{\sigma_{t'}} (\mathbf{z}_{t'} - \alpha_{t'} \hat{\mathbf{x}}_\eta(\mathbf{z}_{t'}))$   
 $\tilde{\mathbf{x}} = \frac{\mathbf{z}_{t''} - (\sigma_{t''}/\sigma_t) \mathbf{z}_t}{\alpha_{t''} - (\sigma_{t''}/\sigma_t) \alpha_t}$   $\triangleright$  Teacher  $\hat{\mathbf{x}}$  target  
 $\lambda_t = \log[\alpha_t^2 / \sigma_t^2]$   
 $L_\theta = w(\lambda_t) \|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)\|_2^2$   
 $\theta \leftarrow \theta - \gamma \nabla_\theta L_\theta$

**end while**

$\eta \leftarrow \theta$   $\triangleright$  Student becomes next teacher

$N \leftarrow N/2$   $\triangleright$  Halve number of sampling steps

**end for**