去噪扩散概率模型 (DDPM) 原理详解

郭东昊　高歌

2025 年 6 月 9 日

1　引言 (Introduction)

近年来，生成模型（Generative Models）在机器学习领域取得了突破性进展，其在图像生成、风格迁移、数据增强等任务中展现了惊人的能力。在众多生成模型中，去噪扩散概率模型（Denoising Diffusion Probabilistic Models, DDPM）作为一种新兴且强大的模型，受到了广泛关注。
DDPM 的核心思想源于对非平衡热力学的观察。它包含两个关键过程：

1. **前向过程（Forward Process）**：也称为扩散过程（Diffusion Process）。在这个过程中，模型通过数千个时间步（steps），逐渐地向原始数据（如一张清晰的图片）中添加微量的高斯噪声。随着时间的推移，原始数据最终会变成一个完全无规律的、符合标准正态分布的噪声图像。这个过程是固定的，不涉及任何模型学习。
2. **逆向过程（Reverse Process）**：这是模型的核心学习部分。模型学习如何系统地“撤销”前向过程中添加的噪声。从一个纯粹的噪声图像开始，模型利用一个深度神经网络（通常是 U-Net 架构）在每个时间步预测并去除一小部分噪声，逐步将噪声图像还原成一张清晰、真实的图像。

通过学习这个逆向的去噪过程，DDPM 能够从随机噪声中生成高质量、高保真的数据样本。本报告将详细阐述 DDPM 的数学原理，并介绍其依赖的相关技术。

2　相关工作 (Related Work)

2.1　U-Net 架构

U-Net 是一种最初为生物医学图像分割而设计的卷积神经网络架构，但由于其出色的性能，现已广泛应用于各种图像到图像（Image-to-Image）的转换任务中，包括 DDPM 中的噪声预测。
U-Net 的结构呈“U”形，由两个主要部分组成：

- **收缩路径（Contracting Path）**：也被称为编码器（Encoder）。它由一系列标准的卷积层和最大池化层（Max Pooling）组成。在收缩路径中，输入图像的空间维度（高度和宽度）被逐渐减小，而特征图的通道数（深度）则相应增加。这个过程旨在捕捉图像的上下文信息和高级特征。
- **扩张路径（Expansive Path）**：也被称为解码器（Decoder）。它通过上采样（Up-sampling）或转置卷积（Transposed Convolution）将特征图的空间维度逐步恢复到原始输入图像的大小。

扩张路径的关键创新在于 **跳跃连接（Skip Connections）**。这些连接将收缩路径中对应层级的特征图直接拼接到扩张路径的相应层级上。这一操作使得解码器在恢复空间细节时，能够直接利用编码器捕捉到的低级、高分辨率的特征，从而极大地提高了定位精度和细节还原能力。

在 DDPM 中，U-Net 的任务是在给定当前时间步 t 的噪声图像 x_t 的情况下，预测出该图像中包含的噪声 ϵ。U-Net 的输入是噪声图像 x_t 和当前时间步 t 的嵌入表示，输出是一个与输入图像尺寸完全相同的噪声预测图。其强大的特征融合能力使其能够精确地预测噪声，从而在逆向过程中逐步恢复出清晰图像。

2.2 ReLU 激活函数

修正线性单元（Rectified Linear Unit, ReLU）是深度学习中最常用的激活函数之一。它的数学定义非常简单：

$$f(x) = \max(0, x)$$

ReLU 的主要优点在于：

- **解决梯度消失问题**：对于正输入，ReLU 的导数为 1，这有助于在深层网络中维持梯度的流动，避免了 Sigmoid 和 Tanh 等函数在饱和区（输入值过大或过小）梯度接近于零的问题。
- **计算效率高**：ReLU 的计算只涉及一个简单的阈值操作，比 Sigmoid 和 Tanh 等函数中的指数运算要快得多。
- **引入稀疏性**：当输入为负时，ReLU 的输出为零，这可以使网络中的一部分神经元处于非激活状态，从而引入稀疏性，有助于模型更好地提取特征。

2.3 变分自编码器 (Variational Autoencoder, VAE)

变分自编码器（VAE）是一种重要的生成模型，其核心思想与 DDPM 的损失函数推导密切相关。VAE 同样由编码器和解码器组成，但它引入了概率论的视角。

- **编码器** $Q(z|X)$：将输入数据 X 编码为一个概率分布，通常是高斯分布 $\mathcal{N}(\mu, \Sigma)$，而不是一个固定的向量。
- **解码器** $P(X|z)$：从编码器产生的分布中采样一个隐变量 z，然后解码器学习从 z 重建原始数据 X。

VAE 的训练目标是最大化证据下界（Evidence Lower Bound, ELBO），也称为变分下界（Variational Lower Bound, VLB）。ELBO 由两部分组成：

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{z \sim Q(z|X)}[\log P(X|z)] - D_{KL}(Q(z|X)||P(z))$$

其中，第一项是 **重建损失**，鼓励解码器精确地重建输入数据。第二项是 **KL 散度**，作为一个正则化项，它衡量编码器输出的分布 $Q(z|X)$ 与一个预设的先验分布 $P(z)$（通常是标准正态分布 $\mathcal{N}(0, I)$）之间的相似性。这个正则化项使得隐空间（latent space）变得平滑和连续，从而让模型具备从隐空间中随机采样并生成新数据的能力。DDPM 的损失函数正是基于最大化对数似然的变分下界这一思想推导出来的。

3 DDPM 核心原理

3.1 前向过程 (Forward Process)

前向过程是一个固定的、不可学习的马尔可夫链，它逐步向原始数据 x_0 中添加高斯噪声。假设整个过程有 T 个时间步，在每个时间步 t，我们向 x_{t-1} 中加入一个方差为 β_t 的高斯噪声，得到 x_t。这个单步转移可以表示为：

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

其中，$\{\beta_t\}_{t=1}^T$ 是一个预先设定的方差表（variance schedule），通常是从一个较小的值（如 10^{-4}）线性增加到一个较大的值（如 0.02）。令 $\alpha_t = 1 - \beta_t$ 和 $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$，通过重参数化技巧（reparameterization trick），我们可以推导出 x_t 与 x_0 之间的直接关系：

$$\begin{aligned}
x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}) + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}$$

其中 $\epsilon \sim \mathcal{N}(0, I)$。这个性质非常重要，它意味着我们可以在任意时间步 t 直接从原始数据 x_0 采样得到 x_t，而无需迭代计算。因此，从 x_0 到 x_t 的采样分布为：

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

3.2 逆向过程 (Reverse Process)

逆向过程是 DDPM 的核心学习任务，其目标是学习一个模型来逆转上述加噪过程，即从一个纯噪声 $x_T \sim \mathcal{N}(0, I)$ 出发，逐步去除噪声，最终恢复出原始数据 x_0。这个过程也是一个马尔可夫链，由一个神经网络 p_θ 参数化：

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

其中 $p_\theta(x_{t-1}|x_t)$ 表示在给定 x_t 的情况下，模型预测的 x_{t-1} 的分布。我们将其建模为高斯分布：

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

为了训练这个逆向过程，我们需要知道 x_{t-1} 的真实后验分布。利用贝叶斯定理，我们可以推导出给定 x_t 和 x_0 时 x_{t-1} 的后验分布：

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

由于前向过程是马尔可夫链，$q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1})$，因此：

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

前向过程中的各个条件概率都是高斯分布，因此这个后验分布也是高斯分布：

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

对高斯分布进行代数运算后，我们可以得到这个后验分布的均值和方差：

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

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

然而，在实际的逆向过程中，我们无法获取真实的 x_0。我们可以利用前向过程的性质，从 x_t 反解出 x_0 的估计值。回顾前向过程的公式：

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

重新整理，我们可以表示 x_0 为：

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

因此，如果我们能预测在时间步 t 添加的噪声 ϵ，我们就能估计 x_0。这就是为什么我们训练一个神经网络 $\epsilon_\theta(x_t, t)$ 来预测噪声。将 x_0 的估计值代入 $\tilde{\mu}_t$ 的公式中，我们得到：

$$\begin{aligned}\tilde{\mu}_t(x_t, x_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\cdot\frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)}x_t - \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_t)}\epsilon_\theta + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t\end{aligned}$$

经过代数简化，我们可以得到模型预测的均值 μ_θ：

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

因此，学习逆向过程等价于训练一个神经网络 ϵ_θ 来从 x_t 预测噪声 ϵ。这将扩散模型的训练简化为一个噪声预测问题，这也是为什么在 DDPM 中，U-Net 的目标是预测噪声而不是直接预测去噪后的图像。

3.3 损失函数 (Loss Function)

DDPM 的训练目标是最大化数据的对数似然 $\log p_\theta(x_0)$。通过变分推断，我们可以优化其证据下界 (ELBO)：

$$\begin{aligned}L_{VLB} &= \mathbb{E}_{q(x_{0:T})}\left[\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \\ &= \mathbb{E}_{q(x_{0:T})}\left[\log\frac{p(x_T)\prod_{t=1}^{T}p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T}q(x_t|x_{t-1})}\right] \\ &= \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T) + \sum_{t=1}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] \\ &= \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T) - \log\frac{q(x_T|x_0)}{p(x_T)} - \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)\right] \\ &= -D_{KL}(q(x_T|x_0)||p(x_T)) - \sum_{t=2}^{T}\mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right] + \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)]\end{aligned}$$

上述表达式中，我们使用了贝叶斯规则将条件概率 $q(x_t|x_{t-1})$ 转换为 $q(x_{t-1}|x_t, x_0)$，并利用 KL 散度的定义进行了转换。最终得到的变分下界由三部分组成：

$$L_{VLB} = \underbrace{-D_{KL}(q(x_T|x_0)||p(x_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{-\mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))\right]}_{L_{t-1}} + \underbrace{\mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)]}_{L_0}$$

Ho et al. (2020) 的工作发现，通过特定的参数化选择，可以极大地简化这个复杂的损失函数。损失中的 L_{t-1} 项衡量的是真实的后验分布 $q(x_{t-1}|x_t, x_0)$ 与我们学习的模型 $p_\theta(x_{t-1}|x_t)$ 之间的 KL 散度。由于这两个分布都是高斯分布，它们的 KL 散度可以被解析地计算出来，主要取决于它们均值之间的差异。忽略一个不依赖于 θ 的权重系数，这一项可以简化为：

$$L_{t-1} \propto \mathbb{E}_q \left[||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2\right]$$

将前面推导出的均值表达式代入，我们发现目标变成了最小化真实噪声 ϵ 和预测噪声 ϵ_θ 之间的均方误差：

$$L_{t-1} \propto \mathbb{E}_{x_0,\epsilon} \left[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2\right]$$

最终，DDPM 使用了一个非常简洁的、在所有时间步上均匀采样的简化版损失函数：

$$L_{\text{simple}}(\theta) = \mathbb{E}_{t\sim[1,T], x_0, \epsilon\sim\mathcal{N}(0,I)} \left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right]$$

其中 $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$。这个损失函数直观、易于实现，并且在实验中取得了非常出色的效果。

3.4 采样过程 (Sampling Process)

在训练完成后，DDPM 通过逆向过程生成新样本。采样过程从纯高斯噪声 $x_T \sim \mathcal{N}(0, I)$ 开始，然后逐步应用学习到的转移函数 $p_\theta(x_{t-1}|x_t)$ 来获取 x_{t-1}，直到得到 x_0。具体来说，采样算法如下：

1. 从标准正态分布中采样 $x_T \sim \mathcal{N}(0, I)$
2. 对于 $t = T, T-1, \ldots, 1$:
   - 如果 $t > 1$，从 $\mathcal{N}(0, I)$ 中采样 z
   - 计算 $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$
   - 计算 $x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z$，其中 $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
3. 返回 x_0 作为生成样本

在实际应用中，为了在生成时提高效率，Ho 等人提出了一种无噪声的确定性采样方法，即在采样时将方差 σ_t 设为 0。此外，后续研究还提出了更高效的采样策略，如 DDIM (Denoising Diffusion Implicit Models)，它能够在保持生成质量的同时，显著减少所需的采样步骤。

3.5 方差表和参数化选择 (Variance Schedule and Parameterization)

DDPM 的性能在很大程度上依赖于方差表 $\{\beta_t\}_{t=1}^T$ 的选择。Ho 等人 (2020) 在原始论文中使用了线性增长的方差表，从 $\beta_1 = 10^{-4}$ 到 $\beta_T = 0.02$。Nichol 和 Dhariwal(2021) 在后续工作中提出了基于余弦函数的方差表，它在训练过程的中间部分提供了接近线性的下降，而在开始和结束阶段则变化较小：

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad \text{其中} \quad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

其中 s 是一个小的偏移量，防止 $\bar{\alpha}_t$ 在接近 T 时变得太小。实验表明，这种基于余弦的方差表能够显著提高模型的性能。

除了前向过程的方差表外，逆向过程中的方差 $\Sigma_\theta(x_t, t)$ 的参数化选择也很重要。Ho 等人选择将其固定为常数 $\sigma_t^2 I$，而非学习参数，其中 $\sigma_t^2 = \beta_t$ 或 $\sigma_t^2 = \tilde{\beta}_t$。他们发现学习对角方差矩阵会导致训练不稳定和样本质量下降。

Nichol 和 Dhariwal 提出了一种插值方法来学习 Σ_θ，将其设置为 β_t 和 $\tilde{\beta}_t$ 之间的插值：

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1-v) \log \tilde{\beta}_t)$$

其中 v 是由模型预测的混合系数。这种方法能够在一定程度上提高模型的性能，特别是在对数似然方面。

3.6 条件生成 (Conditional Generation)

DDPM 最初被设计为无条件生成模型，但后续研究表明它可以很好地扩展到条件生成任务。条件生成允许我们控制生成过程，例如基于类别标签、描述性文本或其他图像生成特定的图像。

最简单的条件生成方法是在训练和采样过程中将条件信息 y（如类别标签）作为额外输入提供给模型。在这种情况下，噪声预测网络变为 $\epsilon_\theta(x_t, t, y)$，它同时考虑当前噪声图像、时间步和条件信息。

Dhariwal 和 Nichol(2021) 提出了分类器引导（Classifier Guidance）方法，它利用一个在噪声数据上训练的分类器 $p(y|x_t)$ 来引导扩散过程朝向特定类别。具体来说，它通过梯度 $\nabla_{x_t} \log p(y|x_t)$ 来修改噪声预测：

$$\tilde{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t} \nabla_{x_t} \log p(y|x_t)$$

为了控制引导强度，可以添加一个权重系数 s：

$$\tilde{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) - s \cdot \sqrt{1-\bar{\alpha}_t} \nabla_{x_t} \log p(y|x_t)$$

Ho 和 Salimans(2021) 进一步提出了无分类器引导（Classifier-Free Guidance）方法，它不需要单独训练分类器，而是通过同时训练条件模型 $\epsilon_\theta(x_t, t, y)$ 和无条件模型 $\epsilon_\theta(x_t, t)$ 来实现条件生成。在实践中，这通常通过在训练期间随机丢弃条件信息来实现。无分类器引导的采样公式为：

$$\tilde{\epsilon}_\theta(x_t, t, y) = \epsilon_\theta(x_t, t) + s \cdot (\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t))$$

这些条件生成技术大大扩展了 DDPM 的应用范围，使其能够处理更多样化和复杂的生成任务。