DDPM

diffusion-based generative models were first introduced in 2015 and popularized in 2020 when Ho et al. published the paper “Denoising Diffusion Probabilistic Models” (DDPMs).

In DDPMs, the authors changed the formulation and model training procedures which helped to improve and achieve ***“image fidelity”***(图像的保真度) rivaling(媲美) GANs and established the validity of these new generative algorithms.

The best approach to completely understanding ***“Denoising Diffusion Probabilistic Models”*** is by 复习理论（+ 一些数学）和底层代码 。

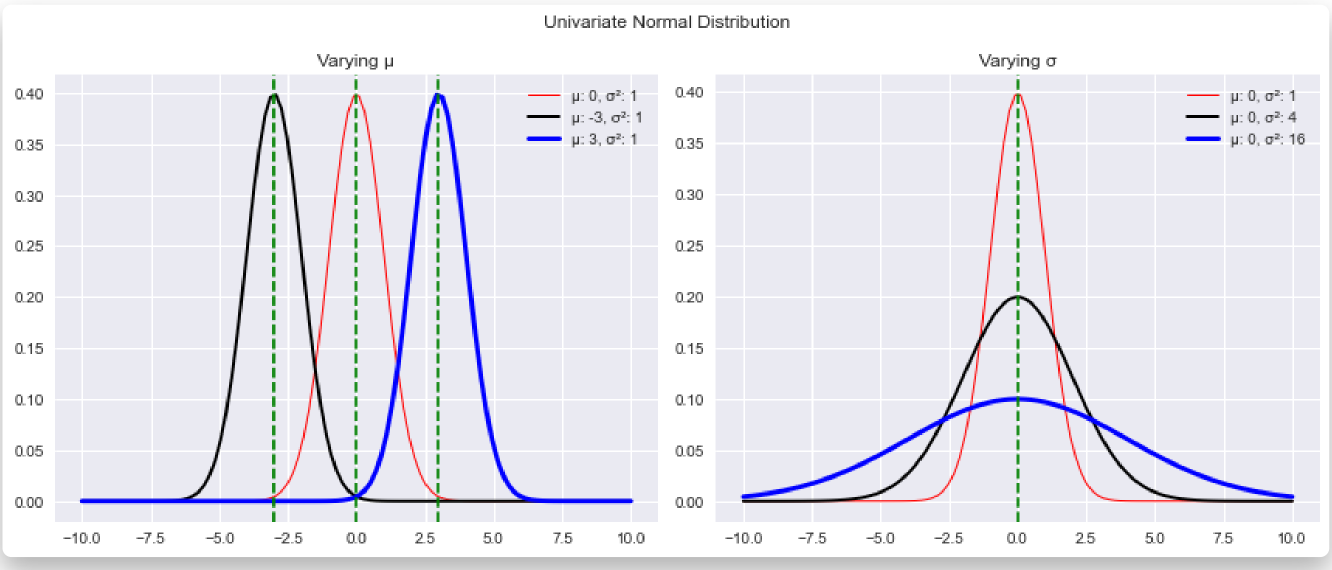
The job of image-based generative models is to generate **new images that are similar,** in other words, “representative” of our original set of images.(基于图像的生成模型的工作是生成相似的新图像，换句话说，“代表” 我们的原始图像集。)

We need to create and train generative models because the set of all possible images that can be represented by, say, just (256x256x3) images is enormous. **An image must have the right pixel value combinations to represent something meaningful** (something we can understand).

For example, for the above image to represent a “Sunflower”, the pixels in the image need to be in the right configuration (they need to have the right values). And the space where such images exist is just a fraction of the entire set of images that can be represented by a (256x256x3) image space. (除了image的像素要正确外，这些256x256x3组合过的图片只有一小部分是被认为是准确)

The probability distribution function or,more precisely, probability density function (PDF) that captures/models this (data) subspace **remains unknown** and most likely **too complex** to make sense. This is why we need ‘Generative models — To figure out the underlying likelihood function our data satisfies.

Every PDF has a set of parameters that determine the shape and probabilities of the distribution. The shape of the distribution changes as the parameter values change. For example, in the case of a normal distribution, we have mean μ and variance σ that control the distribution’s center point and spread.



DDPM基本过程

Diffusion models are a class of generative models inspired by an idea in Non-Equilibrium Statistical Physics(非平衡统计物理学), which states:

“We can gradually convert one distribution into another using a Markov chain”

- *Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015*

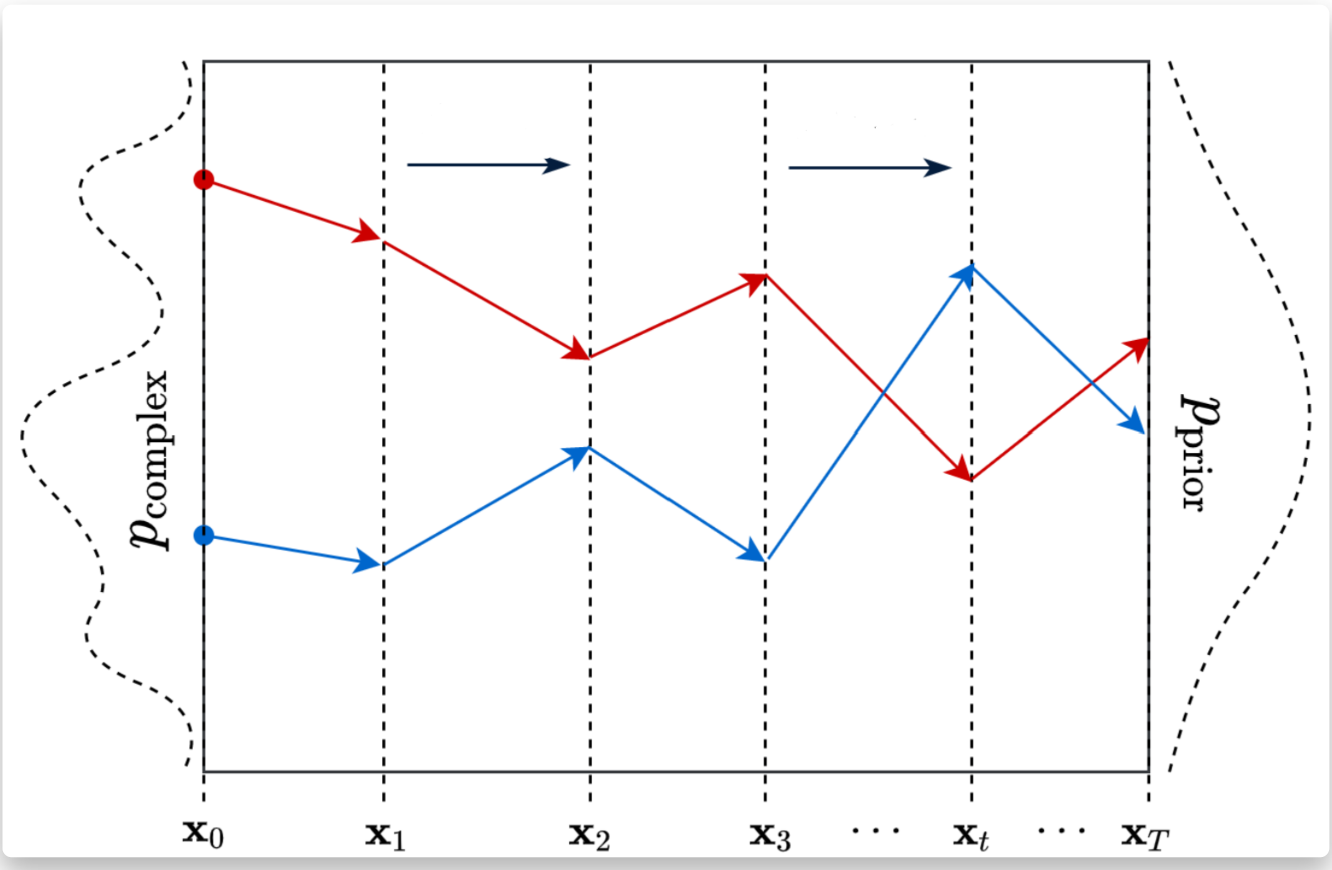
Diffusion generative models are composed of two opposite processes: **Forward & Reverse Diffusion Process**.(正向扩散和逆向扩散)

Forward Diffusion Process:

"It's easy to destroy but hard to create" – Pearl S. Buck

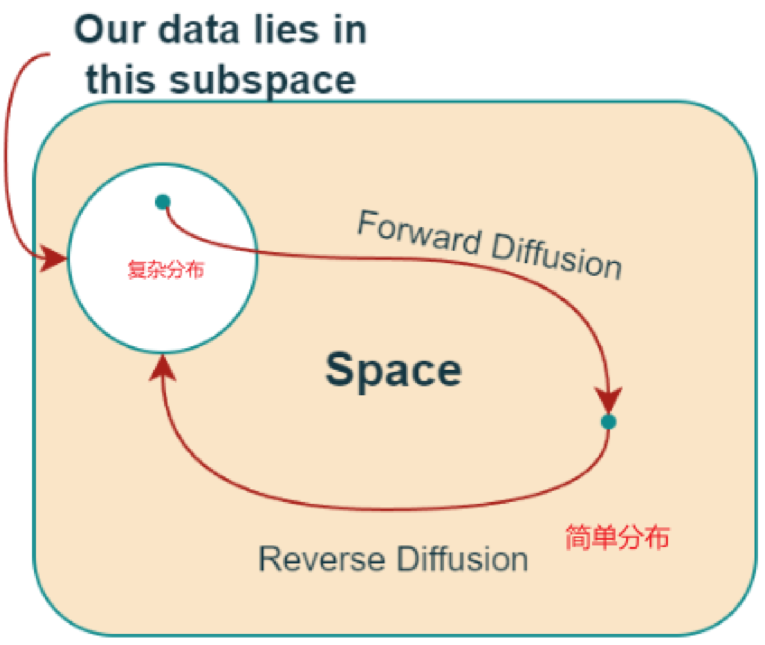
(正向是不断的加正态分布采样的噪声).

- In the "Forward Diffusion" process, we **slowly and iteratively(迭代) add noise to (corrupt) the images in our training set** such that **they "move out or move away" from their existing subspace.**
- What we are doing here is converting the unknown and complex distribution that our training set belongs to into one that is easy for us to sample a (data) point from and understand.(我们这里要做的是将我们的train数据所属于的复杂分布转换为易于我们从中采样(数据)并理解的分布)。
- At the end of the forward process, the images become entirely unrecognizable . The complex data distribution is wholly transformed into a (chosen) simple distribution. Each image gets mapped to a space outside the data subspace.



Reverse Diffusion Process:

By decomposing the image formation process into a sequential application of denoising autoencoders, diffusion models (DMs) achieve state-of-the-art synthesis results on image data and beyond.



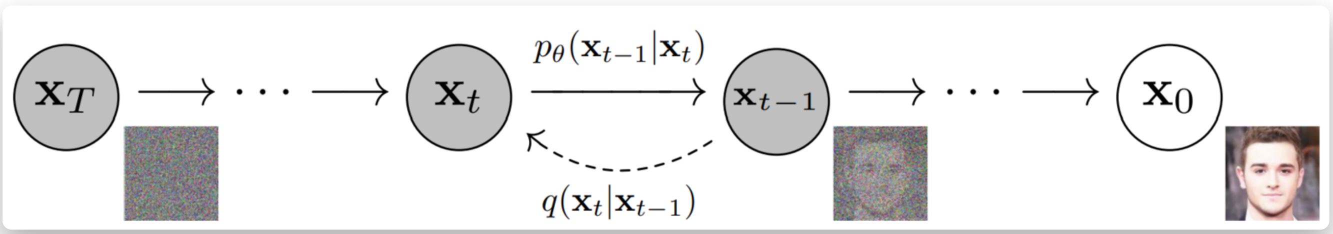
A high-level conceptual overview of the entire image space.

01. In the “Reverse Diffusion process,” the idea is to reverse the forward diffusion process.
02. We slowly and iteratively try to reverse the corruption performed on images in the forward process.
03. The reverse process starts where the forward process ends.
04. The benefit of starting from a simple space is that we know how to get/sample a point from this simple distribution (think of it as any point outside the data subspace(真实的复杂空间的点)).
05. **And our goal here is to figure out how to return to the data subspace.**
06. However, the problem is that we can take infinite paths starting from a point in this “simple” space, but only a fraction of them will take us to the “data” subspace. （我们可以从这个“简单”空间中的一个点开始，有无数条路径，但只有一小部分会把我们带到“数据”子空间）
07. In diffusion probabilistic models, this is done by referring to the small iterative steps taken during the forward diffusion process. （通过参考前向扩散过程中采取的小的迭代步骤来实现）
08. The PDF that satisfies the corrupted images **in the forward process** differs slightly at each step.
09. Hence, **in the reverse process,** we use a deep-learning model at each step **to predict the PDF parameters of the forward process.**
10. **And once we train the model（训练完模型）, we can start from any point in the simple space and use the model to iteratively take steps to lead us back to the data subspace.**
11. In reverse diffusion, we iteratively perform the **“denoising”** in small steps, starting from a noisy image. （小步子去噪，把噪声空间带回子空间）

12. This approach for training and generating new samples is much **more stable than GANs** and **better than** previous approaches like variational autoencoders (**VAE**) and **normalizing flows.**

Since their introduction in 2020, DDPMs has been the foundation for cutting-edge image generation systems, including DALL-E 2, Imagen, Stable Diffusion, and Midjourney.

DPPM 公式流程


$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$

There are two terms mentioned on the arrows:

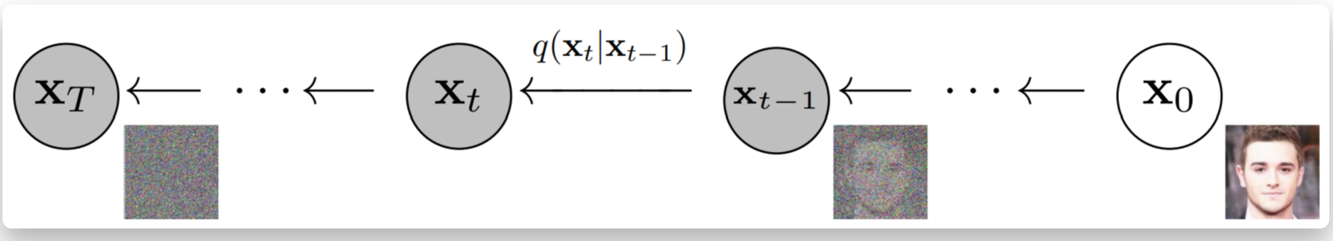
$q\left(x_t \mid x_{t-1}\right)$

- This term is also known as the **forward diffusion kernel (FDK).**
- It defines the PDF of an image at timestep*t* in the forward diffusion process x_t given x_{t-1}.
- It denotes the "transition function" applied at each step in the **forward diffusion process**.

$p_\theta\left(x_{t-1} \mid x_t\right)$

- Similar to the forward process, it is known as the **reverse diffusion kernel (RDK).**
- It stands for the PDF of x_{t-1} given x_i as parameterized by θ . The θ means that the parameters of the distribution of the reverse process are learned using a neural network.
- It's the "transition function"(过渡函数) applied at each step in the **reverse diffusion process**.

前向的数学过程



The distribution q in the forward diffusion process is defined as ***Markov Chain*** given by:

$$q\left(x_1, \ldots, x_T \mid x_0\right) := \prod_{t=1}^{T} q\left(x_t \mid x_{t-1}\right)$$
$$q\left(x_t \mid x_{t-1}\right) := \mathcal{N}\left(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I\right)$$

01. We begin by taking an image from our dataset: x_0. Mathematically it's stated as sampling a data point from **the original (but unknown) data distribution**(复杂分布): $x_0 \sim q\left(x_0\right)$
02. **The PDF of the forward process** is the product of individual distribution(单个分布的乘积) starting from timestep $1 \rightarrow T$.
03. The forward diffusion process is fixed and known.
04. All the intermediate noisy images starting from timestep **1** to **T** are also called "latents." （中间噪声图像也被称为 "潜在"）The dimension of the latents is the same as the original image.
05. The PDF used to define the FDK is a "**Normal/Gaussian distribution"** .
06. At each timestep t, the parameters that define the distribution of image x_t are set as:
    - Mean: $\sqrt{1-\beta_t} x_{t-1}$
    - Covariance: $\beta_t I$
07. The term β (beta) is known as the **"diffusion rate"** and is precalculated(预先)using a "variance scheduler". （β 能被预先计算出来） The term I is an identity matrix. Therefore, the distribution at each time step is called **Isotropic Gaussian.** (各向同性的高斯分布（**球形高斯分布**）指的是各个方向方差都一样的多维高斯分布，协方差为正实数与 identity matrix 相乘)
08. The original image is corrupted at each time step by adding a small amount of **gaussian noise**(ε) (标准高斯分布) .The amount of noise added is regulated by the scheduler.(添加的噪声量由scheduler控制 -> β由variance scheduler决定)
09. By choosing **sufficiently large timesteps** and defining **a well-behaved schedule of** β_t ***the repeated application of FDK gradually converts the data distribution to be nearly an isotropic gaussian distribution.*** (最后被破坏的图像分布接近 isg 分布)

H6 如何从x_{t-1}得到图像x_t，以及如何在每个时间步骤中加入噪声 (关键)

we can easily sample image x_t from a normal distribution as:

$$x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon$$
$$\text{; where } \epsilon \sim \mathcal{N}(0, I)$$

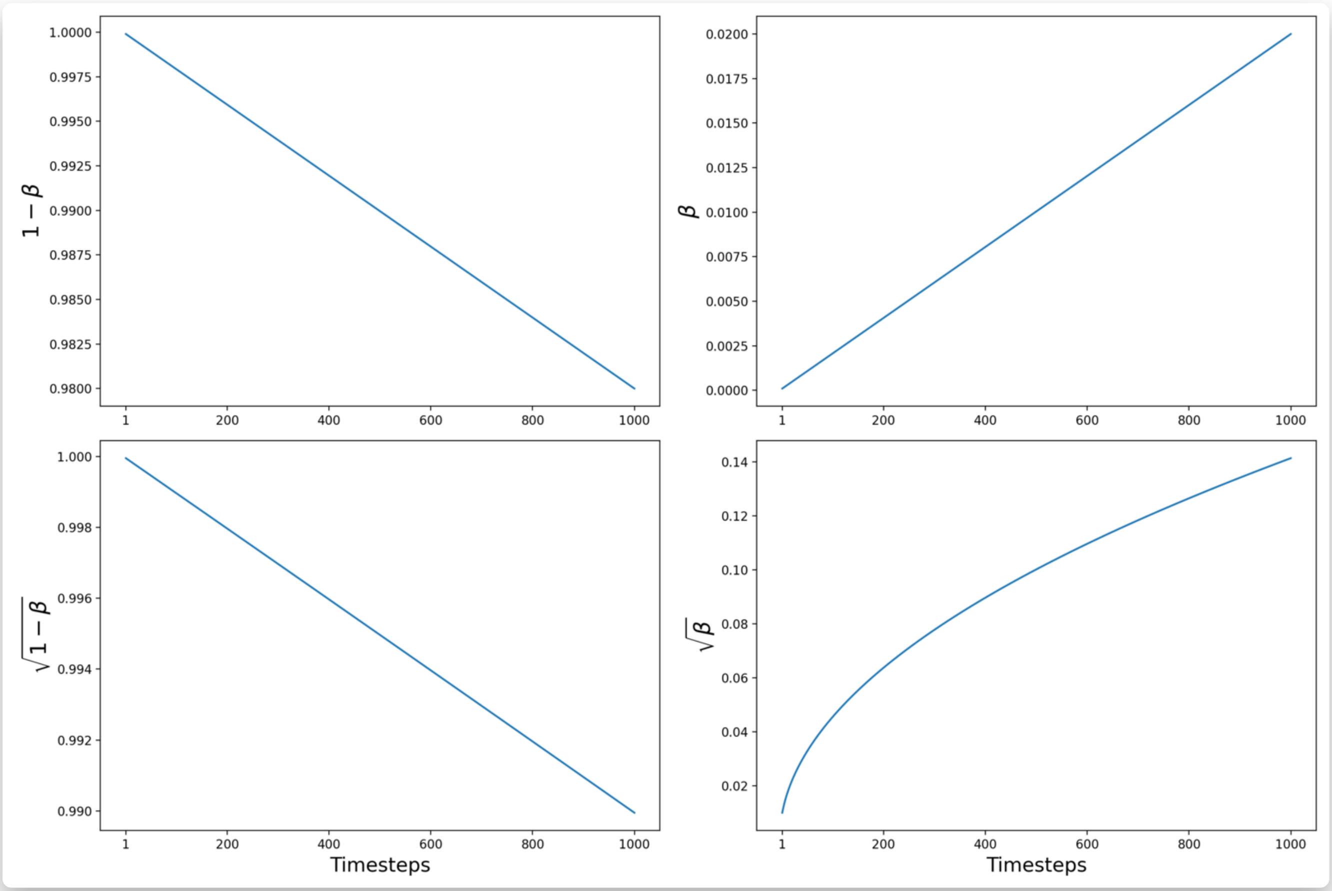
01. the epsilon ε is the "**noise**" term that is randomly sampled from the standard gaussian distribution and is first scaled and then added (scaled) x_{t-1} (ε 是从标准高斯分布中随机采样的“噪声”项，首先被缩放，然后添加（缩放)x_{t-1} 项。
02. In this way, starting from x_0, the original image is iteratively corrupted from $t = 1 \ldots T$

linear variance scheduler

In practice, the authors of DDPMs use a “***linear variance scheduler***” and define β in range **[0.0001,0.02]** and set total timesteps $T = 1000$

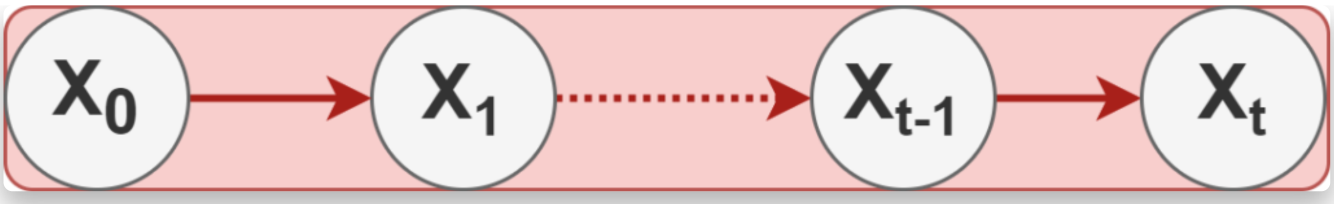
Diffusion models scale down the data with each forward process step (按比例缩小数据) **(by a $\sqrt{1-\beta_t}$ factor) so that variance does not grow when adding noise**. (在添加噪声时方差不会增加)

β 可以提前被计算 下图



There’s a problem here, which results in an inefficient forward process

Whenever we need a latent sample x at timestep t, we have to perform $t-1$ steps （t=1. . .t-1）in the Markov chain.



To fix this, the authors of the DDPM reformulated the kernel to directly go from timestep 0 (i.e., from the original image) to timestep t in the process. (从 t=0 直接到 t)

To do so, two additional terms are defined:

$$\alpha_t := 1 - \beta_t$$
$$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

where $\bar{\alpha}_t$ is a cumulative product(累乘) of α from 1 to t.

用高斯分布的加法特性将β 用$\bar{\alpha}$ 替换掉 （具体推导？）

$$q\left(x_t \mid x_0\right) := \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t} x_{t-1}, (1-\bar{\alpha}_t) I\right)$$
$$x_t := \sqrt{\overline{\alpha_t}} x_0 + \sqrt{1-\bar{\alpha}_t} \epsilon$$

🚀 Using the above formulation, we can sample at any arbitrary timestep t in the Markov chain.

注： 高斯分布的加法特性

$$X \sim N\left(\mu_X, \sigma_X^2\right)$$
$$Y \sim N\left(\mu_Y, \sigma_Y^2\right)$$
$$Z = X + Y$$

then

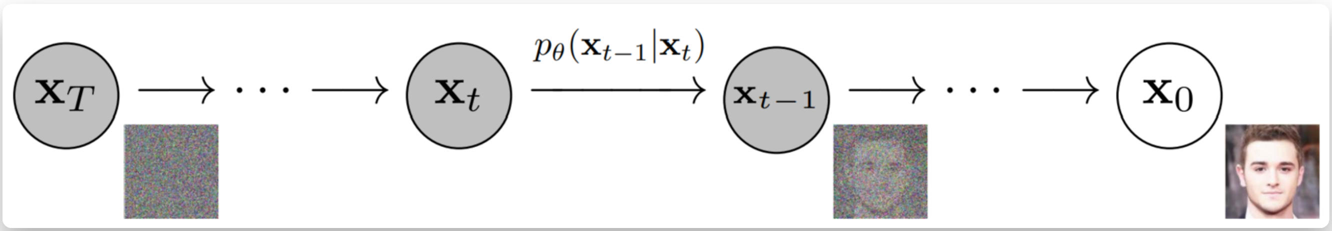
$$Z \sim N\left(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2\right)$$

THE REVERSE DIFFUSION 数学细节

“In the reverse diffusion process, the task is to learn a finite-time (within T timesteps) reversal of the forward diffusion process.

This basically means that we have to “undo” the forward process , to remove the noise added in the forward process iteratively. It is done using a neural network model.（我们必须 “撤消” 前向过程，即迭代地去除前向过程中添加的噪声。 它是使用神经网络模型完成的。）

In the forward process, the transitions function q was defined using **a Gaussian**, so what function should be used for the reverse process p ? ***What should the neural network learn?***



- In 1949, W. Feller showed that, for gaussian distributions, the diffusion process's reversal has the same functional form as the forward process.
- This means that similar to the FDK, which is defined as a normal distribution, we can use **the same functional form (a gaussian distribution)** to define the reverse diffusion kernel.
- The reverse process is also a Markov chain where a **neural network predicts the parameters for the reverse diffusion kernel at each timestep.**
- During training, the learned of the parameters (reverse) should be close to the parameters of the FDK's posterior at each timestep.
- **We want this because if we follow the forward trajectory in reverse, we may return to the original data distribution.**
- we would also learn **how to generate new samples that closely match the underlying (中间层)data distribution**, **starting from a pure gaussian noise** (we do not have access to the forward process during inference).

数学细节

01. The Markov chain for the reverse diffusion starts from where the forward process ends, at timestep **T**, where the data distribution has been converted into (nearly an) **isotropic gaussian distribution.**(X_T 的位置)

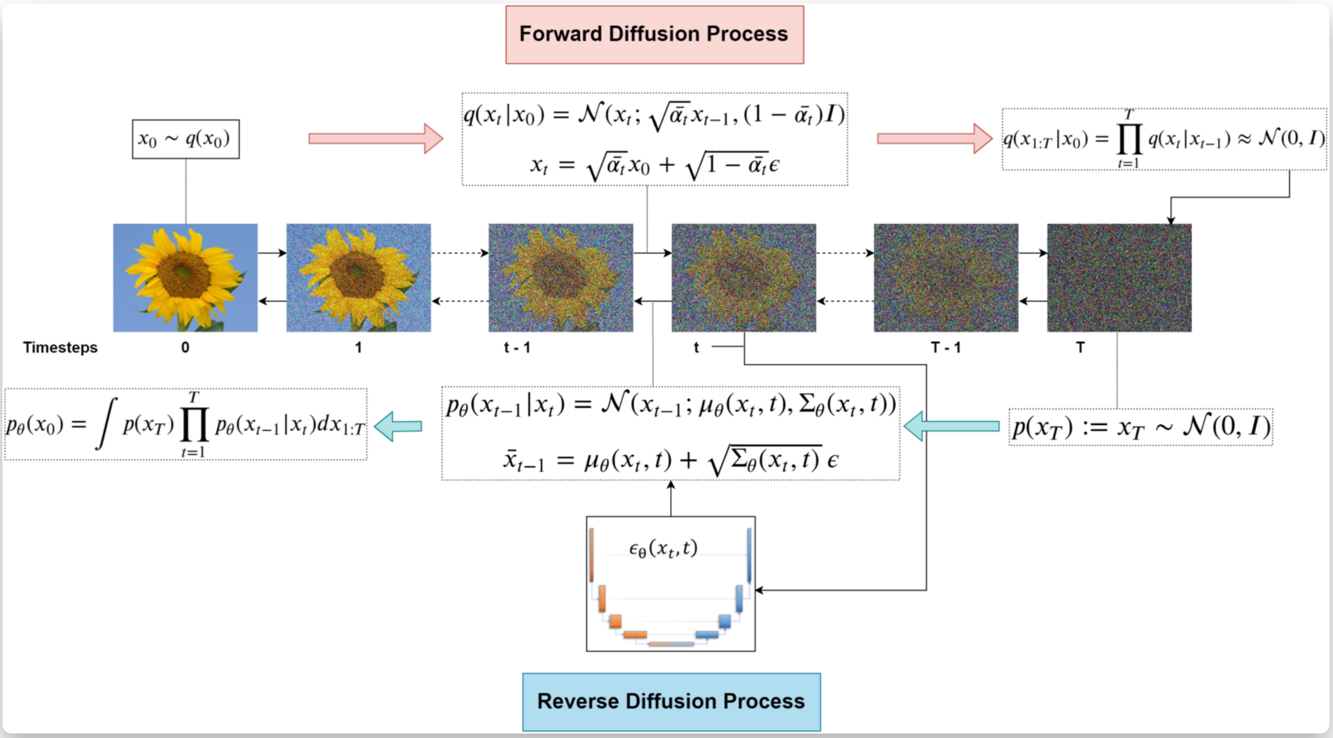
$$q\left(x_T\right) \approx \mathcal{N}\left(x_t; 0, I\right)$$
$$p\left(x_T\right) := \mathcal{N}\left(x_t; 0, I\right)$$

02. The PDF of the reverse diffusion process is an "integral" over all the possible pathways we can take to arrive at a data sample (in the same distribution as the original) starting from pure noise x_T.

(反向扩散过程的PDF是对所有可能的路径的 "积分"，我们可以从纯噪声x_T开始到达一个数据样本（与原始分布相同））

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

$$p_\theta\left(\mathbf{x}_{0:T}\right) := p\left(\mathbf{x}_T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right), \quad p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right)$$


Forward Diffusion Process
$x_0 \sim q(x_0)$
$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_{t-1}, (1-\bar{\alpha}_t)I)$
$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \approx \mathcal{N}(0, I)$
Timesteps
0
1
t - 1
t
T - 1
T
$p_\theta(x_0) = \int p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)dx_{1:T}$
$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$
$\bar{x}_{t-1} = \mu_\theta(x_t,t) + \sqrt{\Sigma_\theta(x_t,t)}\,\epsilon$
$p(x_T) := x_T \sim \mathcal{N}(0, I)$
$\epsilon_\theta(x_t, t)$
Reverse Diffusion Process

优化目标（损失函数）

基于扩散的生成模型的训练目标相当于 “最大化生成的样本（在反向过程结束时）（x）属于原始数据分布的对数似然”。

我们将扩散模型中的transition functions 定义为 “高斯函数”。 为了最大化高斯分布的对数似然，就是尝试找到分布 $(\boldsymbol{\mu}, \sigma^2)$的参数，使其最大化属于与原始数据分布相同的（生成的）数据的 “似然” 数据。

为了训练我们的神经网络，我们将损失函数 (L) 定义为目标函数的负值。 所以 p$\boldsymbol{\theta}$(x$_0$)的高值意味着低损失，反之亦然 。

$$p_\theta\left(x_0\right):=\int p_\theta\left(x_{0:T}\right) d x_{1:T}$$
$$L=-\log \left(p_\theta\left(x_0\right)\right)$$

事实证明，这是棘手的，因为我们需要在非常高维（像素）空间上积分以获得 T 时间步长的连续值。（VAE）or ELBO.

 作者从 VAE 中汲取灵感，并使用变分下界 (VLB)（也称为 “证据下界”(ELBO)）重新制定训练目标，这就是这个看起来很吓人的方程👻:

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right]=\mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right)-\sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right]=: L$$

经过一些简化后，DDPM 作者得出了最终的 L_{vlb}—— 变分下界损失项:

$$\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T}+\sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}]$$

我们可以将上述 L_{vlb} 损失项分解为单独的时间步长，如下所示:

$$\begin{aligned}
L_{\mathrm{vlb}} & :=L_0+L_1+\ldots+L_{T-1}+L_T \\
L_0 & :=-\log p_\theta\left(x_0 \mid x_1\right) \\
L_{t-1} & :=D_{KL}\left(q\left(x_{t-1} \mid x_t, x_0\right) \| p_\theta\left(x_{t-1} \mid x_t\right)\right) \\
L_T & :=D_{KL}\left(q\left(x_T \mid x_0\right) \| p\left(x_T\right)\right)
\end{aligned}$$

这个损失函数很大,但是 DDPM 的作者通过忽略他们简化的损失函数中的一些项来进一步简化它。

-> 去掉L_0 and LT

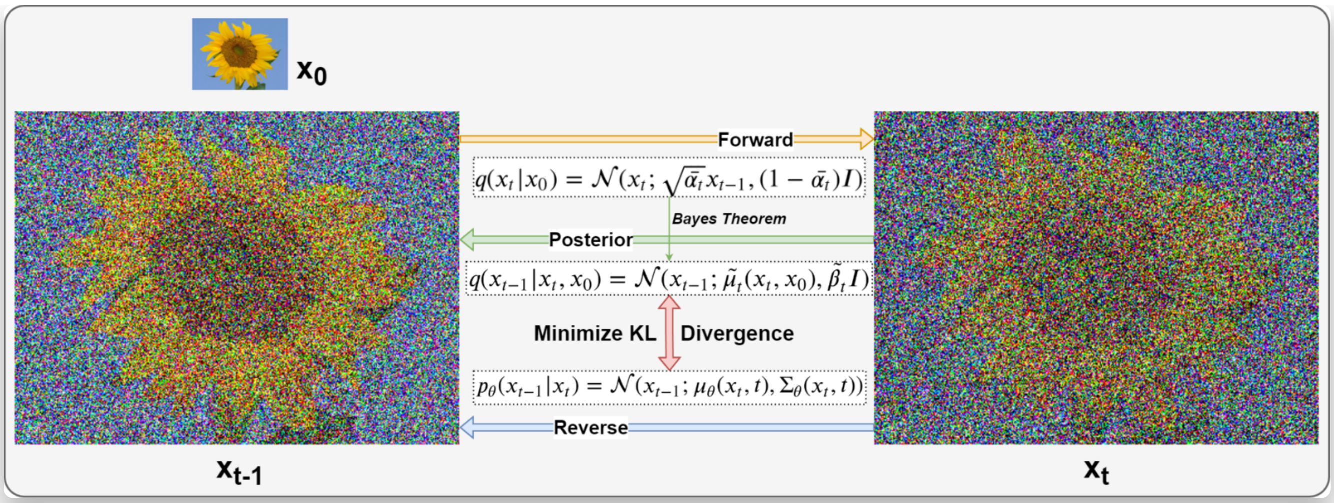
01. L_0 – The authors got better results without this.
02. L_T – 这就是正向过程中的 最后隐层的分布和反向过程中的第一个潜势分布之间的 KL相似性。然而，这里没有涉及神经网络参数，所以除了定义一个好的variance scheduler和使用大的large timesteps，使它们都代表一个an Isotropic Gaussian distribution，我们对此无能为力。

So L_{t-1} is the only loss 它是前向过程（以 x_t 和初始样本 x_0 为条件）的 “后验” 与参数化反向扩散过程之间的 KL 散度。

$$L_{vlb}:=L_{t-1}:=D_{KL}\left(q\left(x_{t-1} \mid x_t, x_0\right) \| p_\theta\left(x_{t-1} \mid x_t\right)\right)$$

The term $q\left(x_{t-1} \mid x_t, x_0\right)$ is referred to as ***"forward process posterior distribution."***

在训练期间的工作是近似 / 估计此（高斯）后验的参数，以使 KL 散度尽可能小。



x_0
Forward
$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_{t-1}, (1-\bar{\alpha}_t)I)$
Bayes Theorem
Posterior
$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$
Minimize KL Divergence
$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$
Reverse
x_{t-1}
x_t

The parameters of the posterior distribution are as follows:

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_{t-1} ; \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right), \tilde{\beta}_t \mathbf{I}\right),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right):=\frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1-\bar{\alpha}_t} \mathbf{x}_0+\frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t} \mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t:=\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$$

To further simplify the task of the model, **the authors decided to fix the variance to a constant** β_t

模型只需要学习预测上面的方程。 并且反向扩散内核被修改为:

$$\begin{gathered}p_\theta\left(x_{t-1} \mid x_t\right)=\mathcal{N}\left(x_{t-1} ; \mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right) \\ \downarrow \\ p_\theta\left(x_{t-1} \mid x_t\right)=\mathcal{N}\left(x_{t-1} ; \mu_\theta\left(x_t, t\right), \sigma^2 I\right)\end{gathered}$$

由于我们保持方差不变，最小化 KL 散度就像最小化两个高斯分布 q 和 p 的均值 ($\boldsymbol{\mu}$) 之间的差异（或距离）一样简单:

$$L_{t-1}=\mathbb{E}_q\left[\frac{1}{2 \sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right)-\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right]+C$$