从零开始生成模型

by shj_ren

如果有任何问题请联系(shj_ren@foxmail.com)

要讲扩散模型，不得不提VAE。VAE和GAN一样，都是从隐变量Z生成目标数据X。它们假设隐变量服从某种常见的概率分布（比如正态分布），然后希望训练一个模型G，这个模型将原来的概率分布映射到训练集的概率分布，也就是分布的变换。注意，VAE和GAN的本质都是概率分布的映射。
大致意思就是先用某种分布随机生成一组隐变量，然后这个隐变量会经过一个生成器生成一组目标数据。但是这种方法本质上是难以work的，因为“尽量接近”并没有一个确定的关于X和$\hat{X}$的相似度的评判标准。换句话说，这种方法的难度就在于，必须去猜测“它们的分布相等吗”这个问题，而缺少真正interpretable(可说明的，可解释的)的价值判断。有聪明的同学会问，KL散度不就够了吗？不行，因为KL散度是针对两个已知的概率分布求相似度的，而X和$\hat{X}$的概率分布目前都是未知。
GAN的做法就是直接把这个度量标准也学过来就行，相当生猛。但是这样做的问题在于依然不interpretable，非常不优雅。VAE的做法就优雅很多了，我们先来看VAE是怎么做的，理解了VAE以后再去理解Diffussion就很自然了。

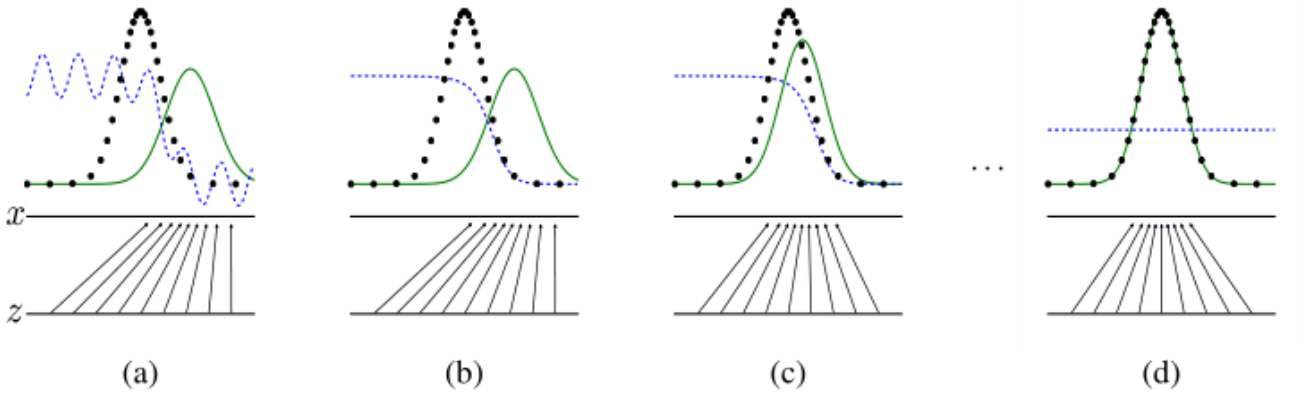
生成模型

我们拿到一批sample(称为X)，想要用X学到它的分布P(X)，这样就能同时学到没被sample到的数据了，用这个分布P(X)就能随意采样，然后获得生成结果。但是这个分布十分复杂，根本不可能直接获得。所以绕个弯，找一个隐变量Z(什么是隐变量:我们可以这么认为，对于一个很复杂的分布往往内部都有几个决定性变量来决定数据的情况，比如一个人的隐变量可能有年龄，经历，性别这在很大程度上决定了人的复杂行动)，这东西可以生成X。不妨假设
Z满足正态分布(为什么用正态分布，当然非正态分布也可以因为很多基函数都可以拟合任何一个分布)，那就可以先从正态分布里面随便取一个Z，然后用Z和X的关系(也就是解码器DECODER)算出X。
也就是$P(x) = \Sigma_z P(X|Z)P(Z)$,其中P(X|Z)被称为似然分布，P(Z)称为隐变量分布(什么是先验什么是后验？后验就是由果求因，先验就是我们知道的结果的总体概率，也就是我们的常识)

GAN

对于依赖性极强同时概率极为复杂的数据，使用深度学习的网络直接生成是十分困难的，例如我们想直接生成一张超大的图片。
所以我们引入GAN的方式，也就是一种对抗的思路，用判别器来判断输入的模型使从数据输入的还是模型生成的。
我们首先定义一个先验噪声变量z，然后使用一个多层感知机G，然后我们使用多层感知机将数据映射到x，也就是$G(z)$,然后定义一个判别器D来判断这个数据是真实的还是使用G生成的，也就是$D(G(z))$
我们的损失函数是

$min_G max_G V(D,G) = E_{x->p_{data}(x)}[log(D(x))] + E_{z->p_z(z)[log[1-D(G(z))]]}$
这个损失函数可以拆分为两部分
首先第一部分：$max_D V(D,G) = E_{x->p(data)(x)}[logD(x)] + E_{z->p_z(z)}[log(1-D(z))]$
这一部分可以理解为我们对于真实数据x，我们要尽可能的增大判别为真的概率，对于虚假数据$G(z)$,我们要尽可能的让这些数据正确的概率小。
第二部分：$minE_{z->p_z}[log(1-D(G(z)))]$
这一部分可以理解为，我们要让我们的映射函数尽可能的生成真实数据，迷惑我们的判别器
很容易的想到，我们可以使用识别器和生成器交替优化的方式来训练我们的模型。
同时因为在早期我们的生成模型是非常弱的也就是G非常弱，$log(1-D(G(z)))$是趋向于正无穷的，所以我们不使用这个损失函数，我们仅仅可以修改成最大化$log(D(G(z)))$.



在训练对抗生成网络的时候，我们同时更新判别函数分布(D,蓝色虚线)，使得D能区分真实数据分布P(x)(黑色虚线)和生成器的数据分布G(z)(绿色实线)。我们下面的黑线是均匀采样获得的z，然后箭头是通过G进行一次变换而获得的数据$G(x)$
G在P(x)高密度区域收缩，且在P(x)的低密度区域扩散。(a)考虑一个接近收敛的对抗的模型对P(G(z))与p(x)相似，且D是个部分准确的分类器。(b)算法的内循环中，训练D来判别数据中的样本，收敛到$D^*(x) = \frac{P(x)}{P(x)+P(G(x))}$.(c)在G的1次更新后，D的梯度引导G(z)流向更可能分类为数据的区域。(d)训练若干步后，如果G和D性能足够，它们接近某个稳定点并都无法继续提高性能，因为此时pg=pdata。判别器将无法区分训练数据分布和生成数据分布，即$D(x) = \frac{1}{2}$
接下来我们来证明为什么更新是真实数据除以真实数据和生成数据之和:
首先:
$V(G,D) = \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz =$
$\int_x p_{data}(x)log(D(x))dx + p_G(x)log(1-D(x))dx$
其中后半部分是我们通过G生成的x的分布

alog(Dx)+blog(1-Dx)我们知道当Dx等于$\frac{a}{a+b}$时取最小值，然后我们就可以通过推导出来的式子进行修改.

$C(G) = max_D V(G,D) = E_{x->p(x)}[logD^*_G(x)] + E_{z->p_z}[log(1-D^*_G(G(z)))]$
$= E_{x->p(x)}[logD^*_G(x)] + E_{z->p_z}[log(1-D^*_G(x))]$
$= E_{x->p(x)}[log\frac{P_{data}(x)}{P_{data}(x)+P_g(x)}] + E_{z->p_z}[log\frac{P_g(x)}{P_{data}(x)+P_g(x)}]$

全局最小值就可以获得了这样-log4，这是十分简单的
然后我们把-log4提取出来，就可以获得
$C(G) = -log4 + KL(p_{data}||\frac{P_{data}+p_g}{2}) + KL(p_g||\frac{P_{data}+p_g}{2})$
要注意一个问题，就是我们的G也就是生成模型不能计算太多次，因为如果计算太多次，会出现和和输入独立完全自主生成单一真实图像的问题,我们可以称之为退化问题。

KL散度

数据编码所包含的信息量，也可以i结尾编码信息所需要的平均数码位数，我们可以直观的理解信息熵，那就是当小概率时间发生时所包含的信息量一定比大概率事件的信息量大（例如今天太阳从东方升起，和今天太阳从西边升起）
然后我们来为什么的人们选择的对数作为信息熵的公式
对于两个互不相关的事件 h(x,y) = h(x) + h(y) p(x,y) = p(x)*p(y)这个式子我们可以十分轻松的理解，前面的式子是两个相互独立的事件的信息熵的和，的第二个式子是两个事件同时发生的概率，所以我们轻松的看出了为什么信息熵使用的是对数，因为对数可以将乘法转化为加法
现在我们可以考虑kl散度kl散度可以理解为信息量的损失程度，即我们有一个真实分布，也有一个拟合分布，将拟合分布和真实分布的各项信息熵相减便可以得出kl散度的结果
虽然不少人认为kl散度是个距离但是实际情况并不是，我们通过一个简单的例子来验证这个结果是错误的，即我们用正态分布去拟合二项分布和用二项分布去拟合正态分布结果是不同的
$D_{KL}(p||q) = \Sigma p(x)(logp(x) - logq(x))$
对于两个高斯分布的KL散度为$KL(p,q) = log\frac{\sigma_1^2}{\sigma_2^2} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$

JS散度

JS散度
为什么我们引入JS散度，因为KL散度具有不对称性的问题，为了解决这个问题，我们引入JS散度

由于KL散度是非对称的，对其稍加修改，便能转化为**对称**的JS散度
首先，我们设$M = \frac{1}{2}(P + Q)$，则

$$JSD(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M)$$

如果我们把**KL散度公式带入展开**的话，结果如下

$$JSD(P||Q) = \frac{1}{2}\sum p(x)\log(\frac{p(x)}{\frac{p(x)+q(x)}{2}}) + \frac{1}{2}\sum q(x)\log(\frac{q(x)}{\frac{p(x)+q(x)}{2}})$$

我们接下来把**log中的$\frac{1}{2}$放到分母**上

$$JSD(P||Q) = \frac{1}{2}\sum p(x)\log(\frac{2p(x)}{p(x)+q(x)}) + \frac{1}{2}\sum q(x)\log(\frac{2q(x)}{p(x)+q(x)})$$

接着**把2提出**

$$JSD(P||Q) = \frac{1}{2}\sum p(x)\log(\frac{p(x)}{p(x)+q(x)}) + \frac{1}{2}\sum q(x)\log(\frac{q(x)}{p(x)+q(x)}) + \log 2$$

这是因为$\sum p(x) = \sum q(x) = 1$

但是这个分布也有问题，如果两个分布不重叠，散度的值就会为常数，没有梯度，无法更新,我们只需要证明不重叠右边为0即可

参数重整化

我们从高斯分布$N(\mu, \sigma^2)$,我们可以先从$N(0,1)$采样出一个点z，然后计算$\sigma * z + \mu$,为什么要这样因为假设我们直接采样，这样是不可导的，也就是梯度无法缠脖，我们的参数不能进行计算

证据似然下界

假设我们将观察到的数据和潜在的潜变量想象成一个联合分布，$P(x,z)$。我们对于这种方式的建模一般是使用似然法进行建模，也就是我们学习一个模型对于所有的观测x的分布$p(x)$我们让这个分布最大化，也就是我们说的极大似然估计，我们有两种方式来将潜在变量增加到这个数据称为联合分布
1.$p(x) = \int p(x,z)dz$
2.$p(x) = \frac{p(x,z)}{p(z|x)}$
直接计算和最大似然是比较困难的，因为在1.中我们要找到所有的z来进行积分。在2.中我们需要一个真实的编码器来获得条件分布。
所以我们使用另外的方式，也就是变分,通过这个我们就可以找到一个下界，我们将下界推高，比如a>b,我们将b变大，a也就变大了。b就称为ELBO
形式上ELBO的公式为$E_{q(z|x)}[log\frac{p(x,z)}{q(z|x)}]$
也就是我们可以获得一个不等式 $logp(x) >= E_{q(z|x)}[log\frac{p(x,z)}{q(z|x)}]$
在这里q就是我们要寻找的最优分布，换句话说，他就是用来逼近真实的后验的$p(z|x)$。

接下来我们首先使用1来推导ELBO
$logp(x) = log \int p(x,z)dz = log \int \frac{p(x,z)q(z|x)}{q(z|x)} = logE_{q(z|x)} = \frac{p(x,z)}{q(z|x)} >= E_{q(z|x)}(log\frac{p(x,z)}{q(z|x)})$
最后一步是使用了jensen不等式来获得的。

接下来我们使用公式2来推导公式捏
$$
\begin{aligned}
&\log p(\boldsymbol{x}) \\
&= \log p(\boldsymbol{x}) \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) dz \quad (\text{Multiply by1} = \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) dz) \\
&= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{x}) dz \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log p(\boldsymbol{x}) \right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} \right] \\
&= \mathbb{E}_{q_\phi(z|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z}) q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x}) q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] \quad (\text{Multiply by1} = \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] + D_{\mathrm{KL}} \left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z}|\boldsymbol{x}) \right) \\
&\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] \quad (\text{KL Divergence always} \geq 0)
\end{aligned}
$$
其实这些公式在后边会再进行推导的，其实不会也没事，我们只需要记住这个内容

我们发现在第一个推导中jensen不等式使用了魔法，消除了这个KL散度，但是这个KL散度是十分重要的，这是我们vae的关键
因为KL散度永远是一个非负的，所以如果我们增大第一项，那么第二项就会越来越接近我们的真实的结果，同时真实分布p是不变的，也就是我们的q后验和$p(z|x)$就会变得越来越相同
然后我们将第一项最大，我们的第一项就是我们的损失函数，我们要最大化第一项

VAE

我们来手动推导一下VAE，我们的假设在前面的生成模型中已经说明了。也就是隐变量是符合正态分布的
我们的损失函数是什么呢，是我们隐变量标准正态分布的KL散度(其实我们只需要预测均值和方差)和ENCODER和DECODER之后的噪声损失。
变分的根本思路就是对于真实的数据分布$P(x)$是很难计算的，所以我们使用一个新的q分布来近似这个分布，我们本质上就是要求P(x|z)和q(z|x),其实我们如果求得了P(x|z)之后P(z|x)也就求得了，因为我们假设了$P(z)$,然后$P(X)$我们可以通过样本获得来训练。
我们来推导我们的损失函数：
$maxL = \Sigma_x logP(x)$
$logP(x) =$

$\int_z q(z|x)logP(x)dz$（$q(z|x)$可以是任何分布因为积分后是1，不影响等式成立）$=$
$\int_z q(z|x)log(\frac{P(z,x)}{P(z|x)})dz = \int_z q(z|x)log(\frac{P(z,x)q(z|x)}{q(z|x)P(z|x)})dz = \int_z q(z|x)log(\frac{P(z,x)}{q(z|x)}dz) +$
$\int_z q(z|x)log(\frac{q(z|x)}{P(z|x)})dz = \int_z q(z|x)log(\frac{P(z,x)}{q(z|x)})dz + KL(q(z|x)||P(z|x))$
$\int_z q(z|x)log(\frac{P(z,x)}{q(z|x)})dz >= q(z|x)\int_z log(\frac{P(x|z)P(z)}{q(z|x)})dz$
这一步是jassion不等式，这就是我们的置信下界，也就是$E_{z->q_{q(z|x)}}[log(\frac{P(x|z)P(z)}{q(z|x)})]$，我们可以用这个方式获得也，当然下面的推导也是正确的，但是这里更好理解,然后我们可以简单的理解分子，其实就是还原原始数据和KL散度。
第二项就是KL散度，我们现在推到出损失函数了，我们的目的是求P(z|x),但是P(z|x)是固定的，所以这个logP(x)是一个常数，我们为了优化KL散度，也就是我们的目的求出P(z|x),我们的目的是最大化第一项。
这个现象从宏观上来看也是很有意思，调节P(x|z)就是在调节Decoder，调节q(z|x)就是在调节Encoder。于是，VAE的训练逻辑就变成了，Decoder每前进一步，Encoder就调节成与其一致的样子，并且站在那拿“枪”顶住Decoder，这样Decoder在下次训练的时候就只能前进，不能退步了。
然后我们继续推理，我们想让第一步最大：
$L_b = \int_z q(z|x)log(\frac{P(z,x)}{q(z|x)})dz = \int_z q(z|x)log(\frac{P(x|z)P(z)}{q(z|x)})dz = \int_z q(z|x)log(\frac{P(z)}{q(z|x)})dz +$
$\int_z q(z|x)log(P(x|z))dz = -KL(P(z)||q(z|x)) + \int_z q(z|x)log(P(x|z))dz$

第二项就是也就是表明在给定q(z|x)（编码器输出）的情况下P(x|z)（解码器输出）的值尽可能高，这其实就是一个类似于Auto-Encoder的损失函数（方差忽略不计的话）也就是输入和输出的相似度
第一项我们使用正态分布，如果有基础的话，那么第一项就是一个高斯混合模型，高斯混合模型就是用一系列高斯分布重建真实的分布

为了使得分布的学习尽量接近，我们希望噪声越小越好，所以我们会尽量使得方差趋于0。但是方差不能为0，因为我们还想要给模型一些训练难度。如果方差为 0，模型永远只需要学习高斯分布的均值，这样就丢失了随机性，我们希望方差能够持续存在，从而带来噪声。
对于多层VAE推导也是一样的，无非是多几个变量

diffusion

在上面呢VAE中我们的似然估计$P(X|Z)$是怎么获得的，是通过DECODER恢复出来的，而$P(Z)$是通过ENCODER计算出来的。但是VAE的问题在于$P(X|Z)$的表达能力和计算代价是不能同时获得的(简单的似然拟合效果并不好，复杂的拟合复杂度非常大，也就是提取隐变量非常困难，透过现象去抓住本质本来就不是简单的)
Diffusion本质就是借鉴了GAN这种训练目标单一的思路和VAE这种不需要判别器的隐变量变分的思路
Diffusion本质就是借鉴了GAN这种训练目标单一的思路和VAE这种不需要判别器的隐变量变分的思路
变分后验分布的表达能力也就是$q(z|x)$与计算代价的权衡一直是VAE领域的核心痛点。
反观GAN，只需要一个“生成器”，先采样高斯噪声，然后用“生成器”把这个高斯噪声映射到数据分布就完事了。而且我们其实只关心生成(也就是模型的边缘分布$P(x|z)$)，大多数情况下并不关心这个后验分布($P(z|x)$)到底是啥。但是GAN也有别的缺陷，比如GAN还需要额外训练判别器，这导致训练很困难，同时容易退化，因为模型很容易只输出一张图片而和输入无关，VAE中的方差噪声其实弥补了这种困难；那么

问题来了，能不能有一种深度生成模型，只需要训练“生成器”，训练目标函数简单，而且不需要训练别的网络（判别器/后验分布等），并且这个生成器没啥限制，可以随便选表达能力极强的神经网络？答案就是diffusion model。
在VAE中难题是如何求得ENCODER $P(z|x)$,我们首先定义了生成器也就是$P(x|z)$,然后定义变分后验来适配这个生成器$p(z|x)$,但是其实可以同时来进行训练，我们可以交替训练当然其实是无所谓的。那么能不能反过来呢？我们能否先定义一个简单的“变分后验”（这里是不准确的描述），再定义“生成器”去适配它呢？如果可以做到，我们就可以避免优化变分后验，而是直接优化生成器！回忆一下，“生成器”想要做的是把标准高斯分布映射到数据分布，那么反过来，“变分后验”其实就是想要把数据分布映射到标准高斯。换句话说，我们能否先定义某种简单的过程，把数据分布映射到标准高斯？这样一来，我们的生成器只需要去拟合这个过程对应的逆过程即可。同时我们的ENCODER也不需要很复杂的训练。VAE因为我们要训练ENCODER和DECODER会导致搜索空间超级大，是呈指数上升。
如果我们先定义了一个简单的过程，把数据分布映射到高斯，那我们的生成器就可以抄作业，直接拟合这个过程每一小步的逆过程即可！这就是diffusion model的核心思想：匹配简单前向过程对应的逆过程的每一小步。

马尔科夫链

首先马尔可夫链是没有后效性的(类似dp)，我们构造适当的马尔科夫链使得不管从什么分布出发，我们的一直采样，最终的分布就是我们想要的平稳分布。我们可以定义一个简单的矩阵变换，然后我们一直传递采样最终结果我们可以求解$Ax = x$，这在最终情况下就平稳了，当然可能会有波动，我们可以定义一个波动范围来限制。

我们继续回到diffusion，这一部分是变分扩散模型

变分扩散模型可以理解为具有三个限制的马尔可夫变分自编码器
1.潜在维度和数据维度完全相等，和我们vae的降维不同
2.每个部分的编码器都是定义好的，我们定义是线性高斯
3.潜在编码器参数是不同的，但是我们最终要将其变成标准高斯

去噪模型和score function

梦开始的地方,现在我们来读开山之作DDPM

介绍

本文介绍了扩散概率模型的研究进展。扩散概率模型(为了简洁起见，我们称其为“扩散模型”)是一个参数化的马尔科夫链，使用变分推理训练，在有限时间后产生与数据匹配的样本。通过学习这个链的过渡来逆转一个扩散过程，扩散过程是一个马尔可夫链，在采样的相反方向上逐渐给数据添加噪声，直到信号被破坏。当扩散由少量高斯噪声组成时，将采样链过渡也设置为条件高斯就足够了，这允许一个特别简单的神经网络参数化，之所以简单是因为我们仅仅需要预测均值和方差即可。
扩散模型定义简单，训练效率高，但据我们所知，还没有证据表明它们能够生成高质量的样本。这篇论文

就是证明扩散模型可以生成质量非常高的数据样本。次啊外我们还展示了在训练过程中对多个噪声使用去噪。
我们还发现大多数的模型的生成部分都用来描述很难察觉的模型细节，也就是模型为了共呢个搞得得分去关注了对人类来说并不重要的部分。
对于扩散模型的损失函数
$E_q[D_{KL}(q(x_T|x_0)||p(X_T)) + \Sigma D_{KL}(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t)) - logp_\theta(x_0|x_1)]$
我们如何理解这个损失函数，第一个是最后一项趋近正态分布，第二项是我们的逆过程趋近于标准的逆过程，我们可以理解成恢复状态,我们不用求逆过程，我们完全可以求前向后向的相似性，和是否能恢复数据。
其中$q(x_{t-1}|x_t,x_0) = N(x_{t-1};\hat{u_t}(x_t,x_0),\hat{\beta_t}I)$
$\hat{u_t}(x_t,x_0) := \frac{\sqrt{\hat{a}_{t-1}}\beta_t}{1-\hat{a_t}}x_0 + \frac{\sqrt{a_t}(1-\hat{a_{t-1}})}{1-\hat{a_t}}x_t$
$\hat{\beta}_t := \frac{1-\hat{a_{t-1}}}{1-\hat{a}_t}\beta_t$

概率扩散模型和去噪自动编码器

前向过程

扩散模型可以看成一种受限制的寻找潜变量的模型。我们必须选择正向过程的方差β_t和高斯分布参数化。
我们将去噪模型和扩散模型建立了新的联系。
我们忽略了正向过程的方差是可以变化的，将其固定为常数(影响不大)，所以我们的近似后验是没有可学习参数的也就是我们损失函数的第一项
我们前向传播就是不断扩散采样的过程
要注意的是我们的正向过程是不含参数的，我们添加的高斯噪声是固定的，不需要训练，只需要你提前选择好就可以。

后向过程，逆过程

我们的本质就是求后验
我们在前边推到说过逆过程也假设是高斯分布我们写为
$p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t,t),\Sigma_\theta(x_t,t))$
$p_\theta(x_{0:T}) = p(x_T)\Pi p_\theta(x_{t-1}|x_t)$
后验扩散条件概率为$q(x_{t-1}|x_t,x_0)$,为什么我们需要x_0,因为虽然有随机过程，但是x_0决定开始条件
我们假设
$q(x_{t-1}|x_t,x_0) = N(\mu(x_t,x_0),\hat{\beta}_t I)$
我们可以跟之前推导的一样
$q(x_{t-1}|x_t,x_0) = q(x_t|x_{t-1},x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}$
我们将式子带入即可

$$
\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\
&\propto \exp\Big(-\frac{1}{2}\big(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\alpha_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\
&= \exp\left(-\frac{1}{2}((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})x_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\alpha_t}}{1-\bar{\alpha}_t}x_0)x_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\right)\Big)
\end{aligned}
$$

我们把上面的式子配方，其实就是二次函数的另一种写法，把中轴线求出来$ax^2+bx+c=a(x+\frac{b}{2a})^2+c$,就有

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

$$
\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0)/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0
$$

由之前的公式我们可以知道$x_0 = \frac{1}{\sqrt{\hat{\alpha}_t}}(x_t - \sqrt{1-\hat{\alpha}_t}z_t)$

我们带入到我们的后验就有

$$
\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t) \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\right)
\end{aligned}
$$

似然函数

我们求上界，这是负对数似然

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\Big[\log\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_q\left[\log\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
\text{Let } L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_0:T)}\Big[\log\frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)}\log p_\theta(\mathbf{x}_0)
\end{aligned}
$$

其中KL散度其实就是一个类似均值

下面用到的公式

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

$$
\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_0:T)}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T)^\star + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\cdot\frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^{T}\log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big] \\
&= \mathbb{E}_q[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{L_0}
\end{aligned}
$$

我们忽略了正向过程的方差是可以变化的，将其固定为常数(影响不大)，所以我们的近似后验是没有可学习参数的也就是我们损失函数的第一项.中间就是逆变换的条件分布，当然我们最后一项可以拿到中间里边

在论文中将$P_\theta(x_{t-1}|x_t)$的方差也设置为了一个常数，所以我们的损失只用拟合均值

所以我们根据之前的KL散度就是$KL(p,q)=\frac{(\mu_1-\mu_2)^2}{2\sigma_2^2}$

获得了损失函数我们由多种建模目标

1.让网络输出等于前向过程的后验分布

2.直接还原我们的图像

3.预测随机噪声的方法，也就是去噪

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

$$
\begin{aligned}
L_{t-1} - C &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}), \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})\right) - \mu_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0,\boldsymbol{\epsilon}),t)\right\|^2\right]
\end{aligned}
$$

对于下面的部分，就是预测噪声的部分，对于下面的式子θ，因为是模型预测的部分，我们不可能手动推

导，所以我们仅仅需要将参数合理的输入即可，为什么我们要让μ_θ和μ_t有相同的形式，因为他们本来就是对应同一种过程我们使用这种近似拟合形式更具有可解释性，也降低了拟合的难度

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right)$$

$$\mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right) \right\|^2 \right]$$

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right) \right\|^2 \right]$$

其中$x_t(x_0, \epsilon)$,就是将x_t用x_0表示

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t)$$

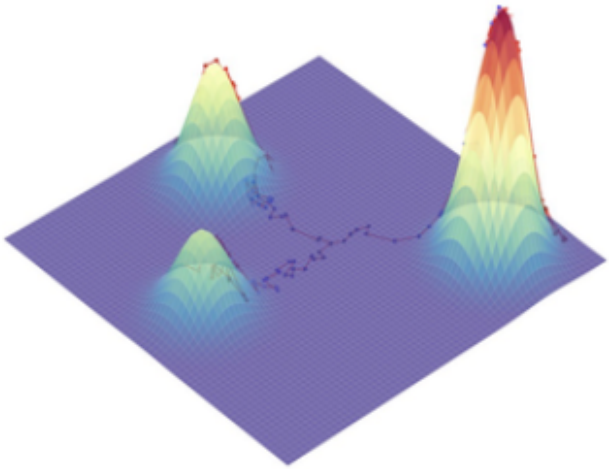
$$= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t \right)$$

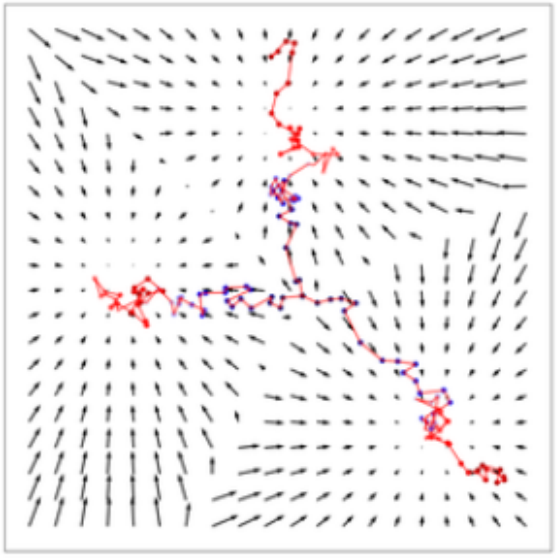
对于这个部分n_θ表示含参的网络，而n_t是我们的推导值

Score Diffusion

这一部分写的不好当时理解不到位，现在也懒得修改了，如果有意修改和增加的在此表示感谢

SCORE





我们有一组样本x_i,概率密度函数的分布是$\nabla logp(x)$,分数网络是$R^D -> R^D$,来逼近分布的分数，我们生成模型的目标是能够生成新的样本，同时我们可以和我们的数据相似，我们的核心是分数匹配和郎之万动力学

郎之万动力学

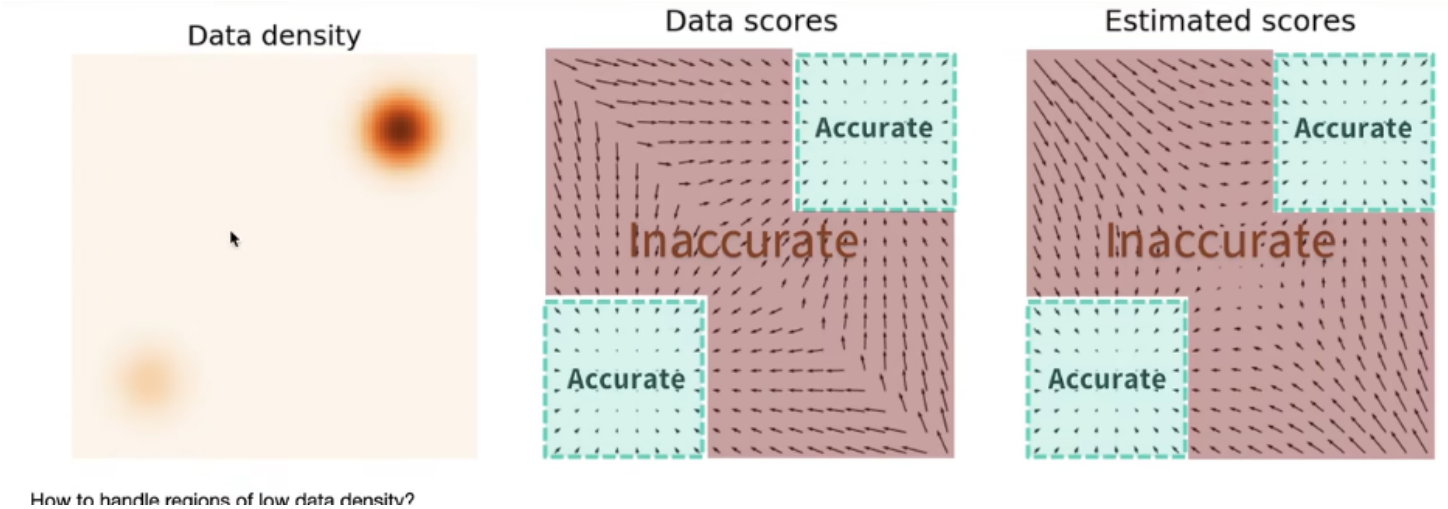
郎之万采样就是仅仅使用得分网络来从我们的分布中采样。
对于我们给定的ϵ(步长)和$\pi(x)$(先验分布)
$\hat{x_t} = \hat{x_{t-1}} + \frac{\epsilon}{2}\nabla p(\hat{x_{t-1}}) + \sqrt{\epsilon}z_t$
其中z_t服从标准正态分布，ϵ接近0，$t->\infty$的时候我们终止采样。
为了采样，我们可以首先训练一个分数网络

分数匹配

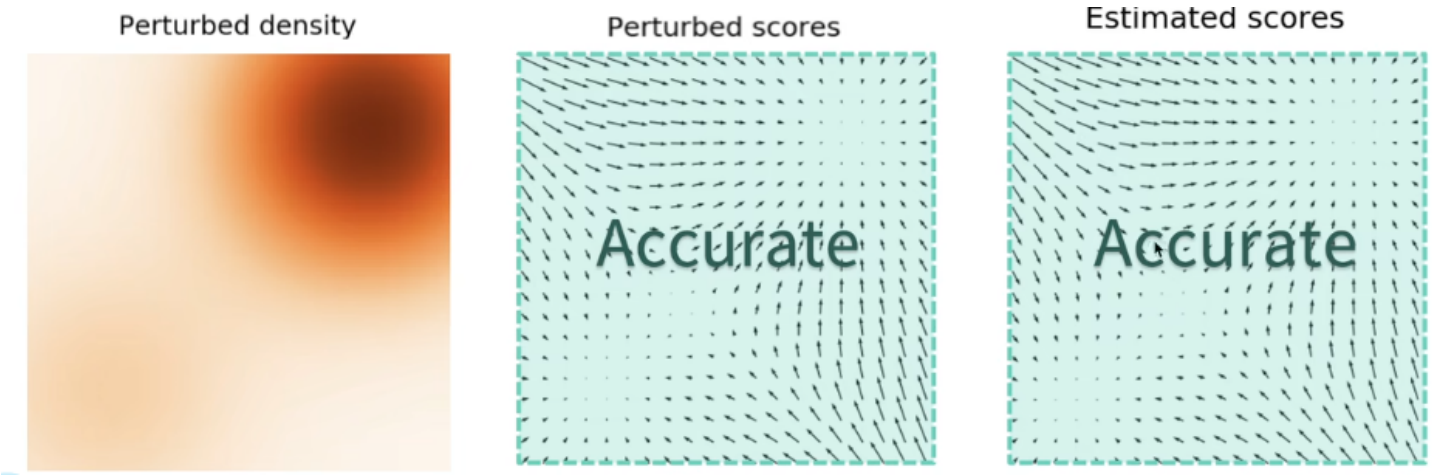
本开是用来归一化未归一化的统计样本
我们的损失是$E_{p_{data(x)}}[||s_\theta - \nabla lop_{data}||_2^2]$
因为我们梯度是不知道的
我们可以等价为$E_{p_{data}}[tr(\nabla s_\theta(x)) + \frac{1}{2}||s_\theta(x)||_2^2]$
tr表示的是雅可比矩阵，但是如果数据集太大，维度过高雅可比矩阵式很难计算的
所以我们引出

噪声分数去噪

我们为了绕过雅可比矩阵引入噪声分数匹配，当然也有随机向量投影的优化。
我们首先定义一个噪声分布q_σ,我们对数据增加一个噪声样本，我们之后的数据分布就是$q_\sigma(\hat{x}) = \int q_\sigma(\hat{x}|x)p_{data}(x)dx$,其中$\hat{x}$是加噪后的
$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})}[\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2].$
当我们噪声非常小之后，我们的就能近似的认为分数网络并没有变化
问题：在数据密度较低的区域，分数估计并不准确，郎之万采样更加不准确



Noise Conditional Score Networks



如何加噪，过大分数可能不对，过小可能分布不合理

1.使用不同量级的分数

2.使用独立的条件分数网络来预测不同条件下的分数

定义一组等比数列σ_i_{i=1}^{L},然后我们对加噪后的数据$q_\sigma(x) = \int p_{data}(t)N(t, \sigma^2 I)dt$

我们的目标就是拟合分数网络$s_\theta(x, \sigma) -> \nabla log q_\sigma(x)$，也就是一招鲜吃遍天，噪声就是条件网络

如何训练NCSN

我们可以选择正态分布$N(x, \sigma^2 I)$作为噪声分布

我们计算梯度就是$\nabla log q_\sigma(\hat{x}|x) = -(\hat{x} - x)\sigma^2$

我们的目标函数就变为

$$\ell(\boldsymbol{\theta}; \sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{\text{dan}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}\right\|_2^2\right]$$

我们的总体目标函数为

$$\mathcal{L}(\boldsymbol{\theta}; \{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L}\sum_{i=1}^L \lambda(\sigma_i)\ell(\boldsymbol{\theta}; \sigma_i)$$

其中λ是一个权重系数，用来平衡噪声大小，所以只要我们训练出一个最好的分数网络，就证明我们的分数网络和真实的数值已经一样了，如果真实分数网络知道之后，后面的常数是可以计算出来的

$\lambda = \sigma^2$

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$

为什么我们选择这个步长，因为要固定信噪比，见原文

本质上扩散模型都是SDE

现在我们应该理解大一统了
就是一阶常微分方程的扩散布朗运动，或者水流之类的运动
首先将变分扩散模型（VDM）推导为马尔可夫分层变分自编码器的特殊情况，通过三个关键假设实现了易于计算和可扩展优化下ELBO的可行性。然后我们证明，优化 VDM 可归结为学习一个神经网络，以预测三个潜在目标之一：从任意噪化输入中恢复原始源输入、从任意噪化的输入中恢复原始源噪声，或者在任意噪声水平下计算噪化输入的分数函数。
基于分数的生成模型是高度相关的；他们不学习对能量函数本身进行建模，而是将基于能量的模型的分数作为神经网络进行学习。扩散模型既有基于似然的解释，也有基于得分的解释。

对于马尔可夫蒙特卡洛，如果我们不限制采样的步数，将数据分布映射到标准高斯分布是比较容易的：我们只需要构造一个平稳分布是标准高斯分布的马尔科夫链即可(当然我不会构造和证明，可以自己学习)，对于离散时间我们构造下面的转移概率

$q(x_t|x_{t-1}) = N(\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$其中$0 < \beta_t < 1$

其中$q_0(x_0)$为数据分布，因为这个条件高斯分布的均值和方差都是关于条件变量的线性函数，我们可以证明，任何时刻关于初始分布的条件分布以掩饰均值和方差都是线性的高斯分布

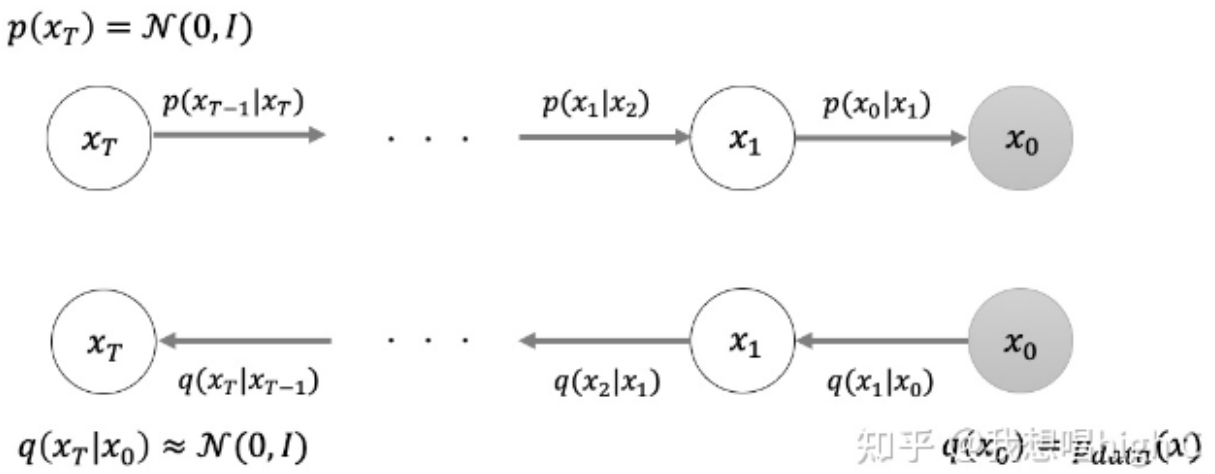
$q(x_t, x_0) = N(\sqrt{\hat{a}_t x_0}, (1-\hat{a}_t)I)$
其中$\hat{a}_t$是一个关于t递减的函数，如果合适的选择β_t，我们还可以得到$lim\hat{a}_t = 0$，也就是意味着
$lim_{t->\infty}q(x_t|x_0) = N(0,1)$

这样一来我们只要许纳区一个足够大的步骤次数N，我们就可以把数据分布x_0映射到一个非常接近高斯分布的x_N,我们称这个过程为前向过程,并且由于$q(x_t|x_{t-1})$的不呢之上就是把x_{t-1},放缩后增加一个噪声，我们也可以把这个过程理解为逐渐给数据增加噪声的过程，这样的过程也被称为扩散过程。

然后我们要考虑DECODER，我们前向过程ENCODER的逆过程也可以近似为高斯分布，准确的来说如果我们选择一个比较合适的β_t，这样x_t和x_{t-1}之间对应的分布变化十分微小的时候，我们可以使用高斯分布来拟合他的逆过程$q(x_{t-1}|x_t)$,我们从不太严谨的角度来看待这个问题

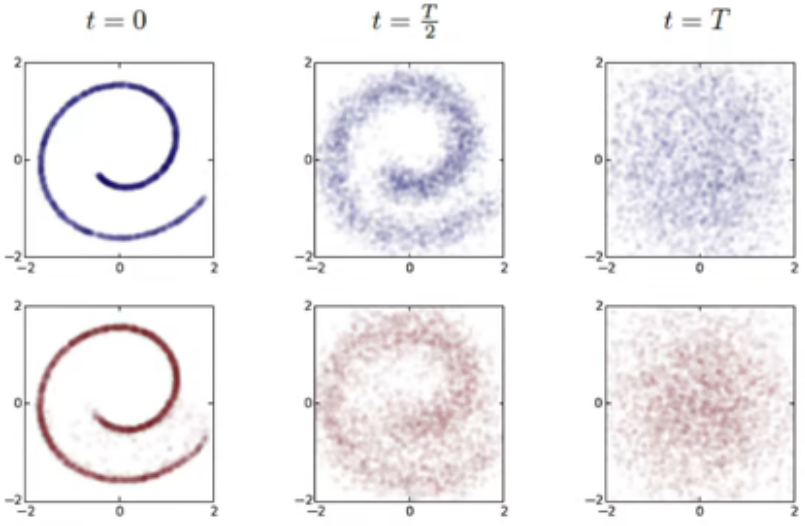
$q(x_{t-1}|x_t)$
$->q(x_t|x_{t-1})exp(logp_{t-1}(x_{t-1})-logp_t(x_t))$
$->q(x_t|x_{t-1})exp(logp_t(x_{t-1})-logp_t(x_t))$
这一步是每一个部分增加的都是一个很小的噪声所以是可以近似相等的
$->q(x_t|x_{t-1})exp((x_{t-1}-x_t)\nabla logp_t(x_t))$
$=\frac{1}{\sqrt{2\pi}\beta_t}exp(\frac{||x_t-\sqrt{1-\beta_t}x_{t-1}||_2^2}{2\beta_t^2}+(x_{t-1}-x_t\nabla logp_t(x_t)))$
我们上面有几个近似，一个是$q_{t-1}->q_t$,还有是使用导数来近似变化。
现在我们函数唯一不知道的就是$\nabla_x logp_t(x_t)$,这个函数也被称为“score function”。

这也是为什么diffusion model还有个角度是score-based generative model，就是因为逆过程里有score function。



我们可以用一个直观的图来感受一下。diffusion model定义了一个简单的前向过程（下面那行），不断地加噪来把真实数据映射到标准高斯；然后又定义一个逆向过程来去噪（上面那行），并且逆向过程的每一步只需要是一个很简单的高斯分布即可。总而言之，从理论角度来看，diffusion model的成功在于我们训练的模型只需要“模仿”一个简单的前向过程对应的逆向过程，而不需要像其它模型那样“黑盒”地搜索模型。并且，这个逆向过程的每一小步都非常简单，只需要用一个简单的高斯分布来拟合。这为diffusion model的优化带来了诸多便利，这也是它empirical performance非常好的原因之一。

接下来一部分是不太重要的，同时是比较严格的推导，可以不看这一部分，直接去看SD模型
效果如图



当然我们其实不需要不断采样迭代出来，我们任意时刻的数据都可以基于x_0, β_t，求出来
我们令$a_t = 1 - \beta_t, \hat{a}_t = \Pi_{i=1}^{T} a_i$
推导$x_t = \sqrt{a_t} x_{t-1} + \sqrt{1 - a_t} z_{t-1} = \sqrt{a_t a_{t-1}} x_{t-2} + \sqrt{a_t - a_t a_{t-1}} z_{t-2} + \sqrt{1 - a_t} z_{t-1} = ... = \sqrt{\hat{a}_t} x_0 + \sqrt{1 - \hat{a}_t} z$

这个推导我们用到N_t, N_{t-1}都是近似标准高斯分布，同时所有的采样分布都是独立的，所以两个分布的叠加为$N(\mu_1 + \mu_2, a^2\sigma_1 + b^2\sigma_2)$

最终高斯分布就是$N(\sqrt{\hat{a}_t} x_0, (1 - \hat{a}_t) I)$
这个次数主要是用来确定迭代次数，实际用可能会有误差

现在我们开始看严格的推导

随机微分
首先我们来看前向过程
我们考虑一个微分方程
$dx = f_t(x)dt + g_t dw$
$x_{t+1} - x_t = f(x)\Delta t + g_t \sqrt{\Delta t}\epsilon$其中$\epsilon$满足标准正态，这个就是我们那个采样过程很好理解

也就是相邻两项的差值就是上面的方程，这很好理解，我们也很好理解其实这个因为我们加噪的过程就是增加噪声的过程

首先我们看前向过程
$p(x_{t+1}|x_t) = N(x_{t+1}; x_t + f(x)\Delta t, g_t^2 \Delta I)$
$exp(-\frac{||x_{t+1} - x_t - f_t(x_t \Delta t)||^2}{2g_t^2 \Delta t})$

然后我们使用贝叶斯定理

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t+\Delta t}) = \frac{p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t)p(\boldsymbol{x}_t)}{p(\boldsymbol{x}_{t+\Delta t})} = p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t)\exp(\log p(\boldsymbol{x}_t) - \log p(\boldsymbol{x}_{t+\Delta t}))$$
$$\propto \exp\left(-\frac{\|\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t - \boldsymbol{f}_t(\boldsymbol{x}_t)\Delta t\|^2}{2\boldsymbol{g}_t^2\Delta t} + \log p(\boldsymbol{x}_t) - \log p(\boldsymbol{x}_{t+\Delta t})\right)$$

当t足够小的时候，我们使用泰勒展开，也就是用导数近似

$$\log p(\boldsymbol{x}_{t+\Delta t}) \approx \log p(\boldsymbol{x}_t) + (\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t)\cdot\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t) + \Delta t\frac{\partial}{\partial t}\log p(\boldsymbol{x}_t)$$

然后我们就有

$$\log p(\boldsymbol{x}_{t+\Delta t}) \approx \log p(\boldsymbol{x}_t) + (\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t)\cdot\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t) + \Delta t\frac{\partial}{\partial t}\log p(\boldsymbol{x}_t)$$

因为最后一项t的二阶导在t->0时几乎为0

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t+\Delta t}) \propto \exp\left(-\frac{\|\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t - \left[\boldsymbol{f}_t(\boldsymbol{x}_t) - g_t^2\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)\right]\Delta t\|^2}{2g_t^2\Delta t}\right)$$
$$\approx \exp\left(-\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_{t+\Delta t} + \left[\boldsymbol{f}_{t+\Delta t}(\boldsymbol{x}_{t+\Delta t}) - g_{t+\Delta t}^2\nabla_{\boldsymbol{x}_{t+\Delta t}}\log p(\boldsymbol{x}_{t+\Delta t})\right]\Delta t\|^2}{2g_{t+\Delta t}^2\Delta t}\right)$$

我们有

$$d\boldsymbol{x} = \left[\boldsymbol{f}_t(\boldsymbol{x}) - g_t^2\nabla_x\log p_t(\boldsymbol{x})\right]dt + g_t d\boldsymbol{w}$$

其中第一项为均值，第二项系数为方差

现在我们来看分数匹配的过程

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \lim_{\Delta t\to 0}\int\cdots\iint p(\boldsymbol{x}_t|\boldsymbol{x}_{t-\Delta t})p(\boldsymbol{x}_{t-\Delta t}|\boldsymbol{x}_{t-2\Delta t})\cdots p(\boldsymbol{x}_{\Delta t}|\boldsymbol{x}_0)d\boldsymbol{x}_{t-\Delta t}\boldsymbol{x}_{t-2\Delta t}\cdots\boldsymbol{x}_{\Delta t}$$

假设p是线性的，也就是我们之前粗略的讲线性高斯的时候，我们有，因为是线性高斯所以可以叠加出一个新高斯

$$p(\boldsymbol{x}_t) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_0)\tilde{p}(\boldsymbol{x}_0)d\boldsymbol{x}_0 = \mathbb{E}_{\boldsymbol{x}_0}\left[p(\boldsymbol{x}_t|\boldsymbol{x}_0)\right]$$
$$\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t) = \frac{\mathbb{E}_{\boldsymbol{x}_0}\left[\nabla_{\boldsymbol{x}_t}p(\boldsymbol{x}_t|\boldsymbol{x}_0)\right]}{\mathbb{E}_{\boldsymbol{x}_0}[p(\boldsymbol{x}_t|\boldsymbol{x}_0)]} = \frac{\mathbb{E}_{\boldsymbol{x}_0}\left[p(\boldsymbol{x}_t|\boldsymbol{x}_0)\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t|\boldsymbol{x}_0)\right]}{\mathbb{E}_{\boldsymbol{x}_0}[p(\boldsymbol{x}_t|\boldsymbol{x}_0)]}$$

实际上我们的最后结果是导数的加权，因为我们假设$p(x_t|x)$是一个不断线性加噪，所以我们是可以直接估算的，但是导数比较困难，所以我们直接使用神经网络拟合导数

$$\mathbb{E}_{\boldsymbol{x}_0}\left[p(\boldsymbol{x}_t|\boldsymbol{x}_0)\|\boldsymbol{s}_\theta(\boldsymbol{x}_t,t) - \nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t|\boldsymbol{x}_0)\|^2\right]$$

这就是我们的前向过程了，因为分母部分是个常数，所以我们只需要优化分子

统一部分:

首先我们直到正态分布的联合分布还是正态分布，我们假设我们知道$p(x_t|x_0)$，因为是个正态分布

我们还知道$p(x_{t+\Delta t}|x_0) = \int p(x_{t+\Delta t}|x_t)p(x_t|x_0)dx_t$

我们将微分方程形式带入dx

记号	含义	采样
$p(\boldsymbol{x}_{t+\Delta t}\vert\boldsymbol{x}_0)$	$\mathcal{N}(\boldsymbol{x}_t;\bar{\alpha}_{t+\Delta t}\boldsymbol{x}_0,\bar{\beta}_{t+\Delta t}^2\boldsymbol{I})$	$\boldsymbol{x}_{t+\Delta t}=\bar{\alpha}_{t+\Delta t}\boldsymbol{x}_0+\bar{\beta}_{t+\Delta t}\boldsymbol{\varepsilon}$
$p(\boldsymbol{x}_t\vert\boldsymbol{x}_0)$	$\mathcal{N}(\boldsymbol{x}_t;\bar{\alpha}_t\boldsymbol{x}_0,\bar{\beta}_t^2\boldsymbol{I})$	$\boldsymbol{x}_t=\bar{\alpha}_t\boldsymbol{x}_0+\bar{\beta}_t\boldsymbol{\varepsilon}_1$
$p(\boldsymbol{x}_{t+\Delta t}\vert\boldsymbol{x}_t)$	$\mathcal{N}(\boldsymbol{x}_{t+\Delta t};(1+f_t\Delta t)\boldsymbol{x}_t,g_t^2\Delta t\,\boldsymbol{I})$	$\boldsymbol{x}_{t+\Delta t}=(1+f_t\Delta t)\boldsymbol{x}_t+g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}_2$
$\int p(\boldsymbol{x}_{t+\Delta t}\vert\boldsymbol{x}_t)p(\boldsymbol{x}_t\vert\boldsymbol{x}_0)d\boldsymbol{x}_t$		$\begin{aligned}&\boldsymbol{x}_{t+\Delta t}\\&=(1+f_t\Delta t)\boldsymbol{x}_t+g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}_2\\&=(1+f_t\Delta t)(\bar{\alpha}_t\boldsymbol{x}_0+\bar{\beta}_t\boldsymbol{\varepsilon}_1)+g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}_2\\&=(1+f_t\Delta t)\bar{\alpha}_t\boldsymbol{x}_0+((1+f_t\Delta t)\bar{\beta}_t\boldsymbol{\varepsilon}_1+g_t\sqrt{\Delta t}\boldsymbol{\varepsilon}_2)\end{aligned}$

$\bar{\alpha}_{t+\Delta t}=(1+f_t\Delta t)\bar{\alpha}_t$

$\bar{\beta}_{t+\Delta t}^2=(1+f_t\Delta t)^2\bar{\beta}_t^2+g_t^2\Delta t$

令$\Delta t=0$

$f_t=\frac{d}{dt}(\ln\bar{\alpha}_t)=\frac{1}{\bar{\alpha}_t}\frac{d\bar{\alpha}_t}{dt},\quad g_t^2=\bar{\alpha}_t^2\frac{d}{dt}\left(\frac{\bar{\beta}_t^2}{\bar{\alpha}_t^2}\right)=2\bar{\alpha}_t\bar{\beta}_t\frac{d}{dt}\left(\frac{\bar{\beta}_t}{\bar{\alpha}_t}\right)$

假设$\alpha^2+\beta^2=1$

我么就有

$s_\theta=\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t\vert\boldsymbol{x}_0)=-\frac{\boldsymbol{x}_t-\bar{\alpha}_t\boldsymbol{x}_0}{\bar{\beta}_t^2}=-\frac{\varepsilon}{\bar{\beta}_t}$

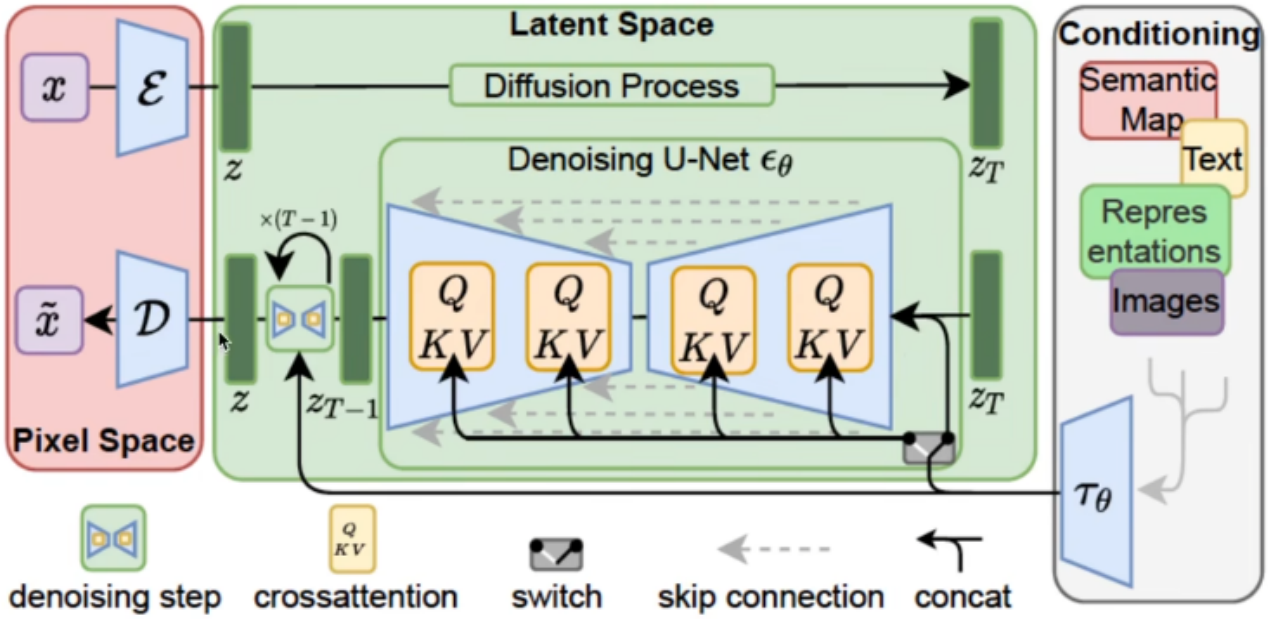
第二个等号是因为$x_t=\alpha x_0+\beta\epsilon$

我们代入score的前向过程就可以获得了DDPM的前向过程

$\frac{1}{\bar{\beta}_t^2}\mathbb{E}_{\boldsymbol{x}_0\sim\tilde{p}(\boldsymbol{x}_0),\boldsymbol{\varepsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\left\|\boldsymbol{\epsilon}_\theta(\bar{\alpha}_t\boldsymbol{x}_0+\bar{\beta}_t\boldsymbol{\varepsilon},t)-\boldsymbol{\varepsilon}\right\|^2\right]$

SD模型

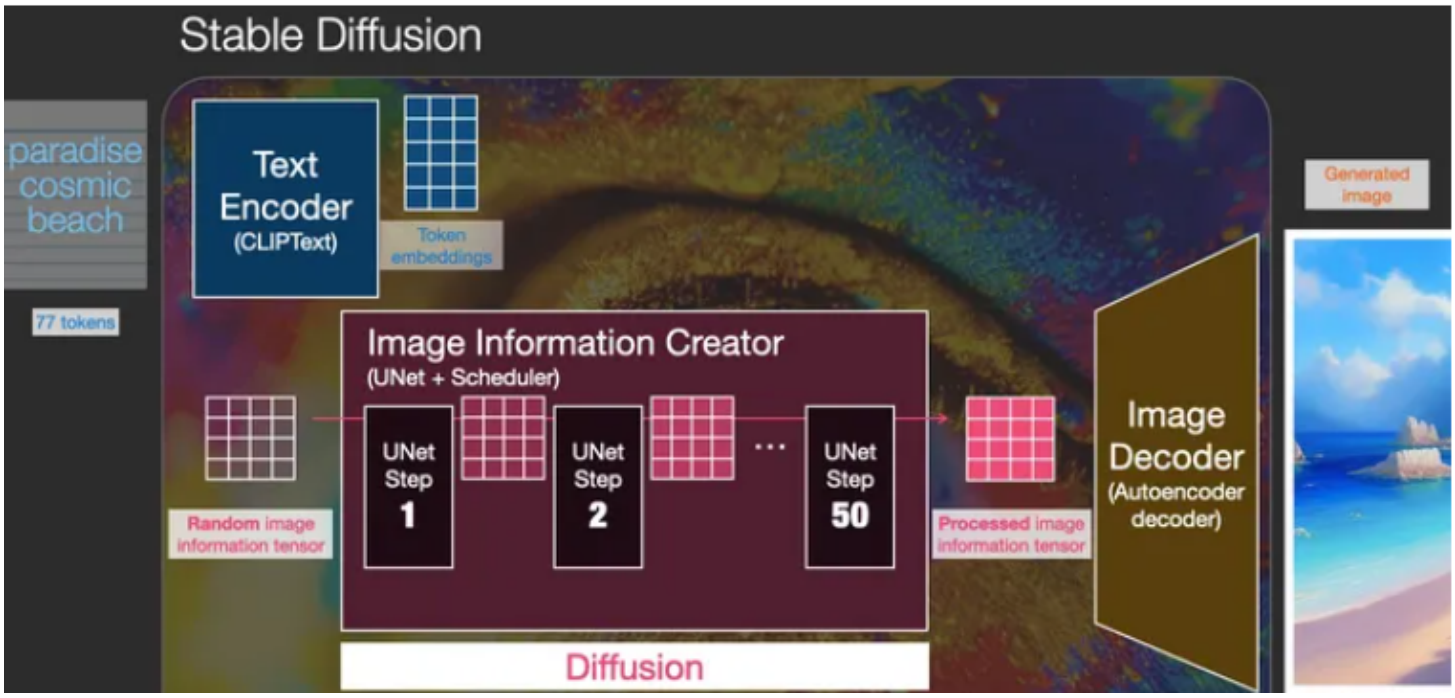
SD模型其实对于加噪和去噪并没有那么的有可解释性，因为他是完全恢复原来的图像而不是预测噪声。有可能不可解释的更work

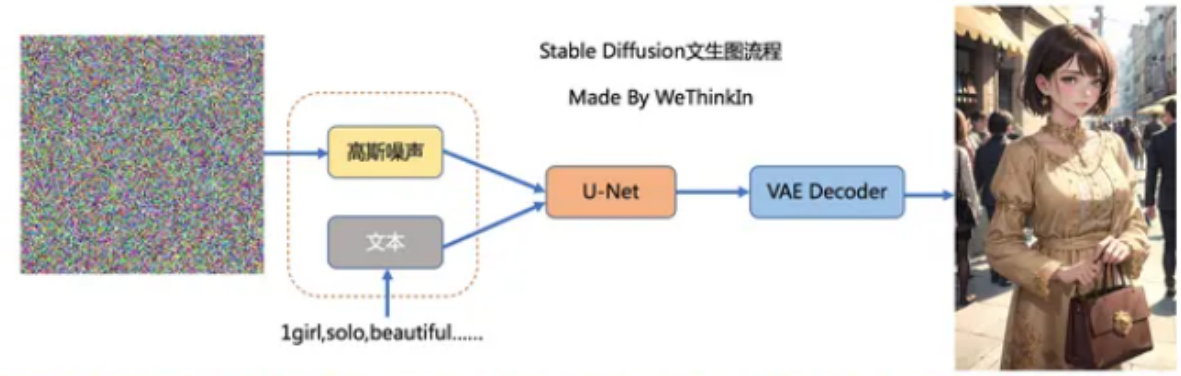


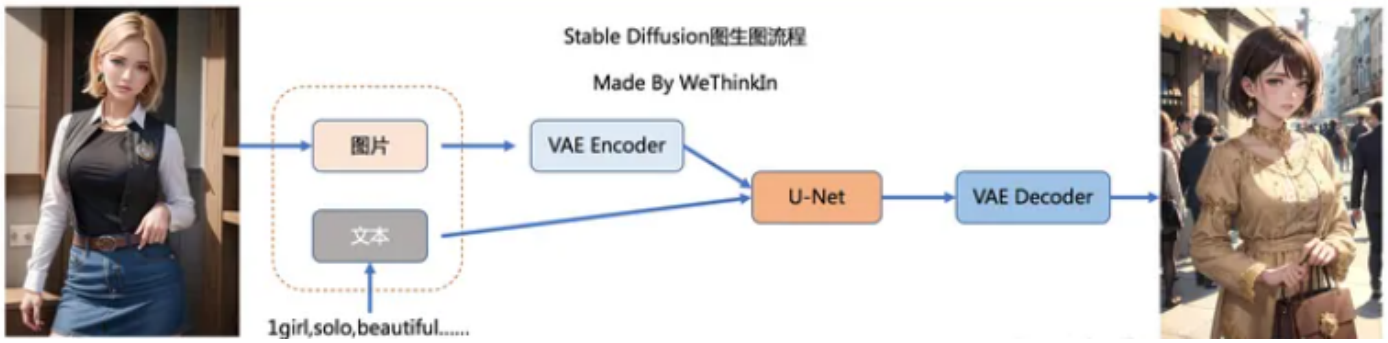
stable duffsion是首先使用vae之类的进行一个encoder获得隐变量，然后通过duffsion对隐变量进行扩散，对于条件的引入，我们使用attention机制来引入的QKV
模型简单来说就是 文本->encoder->图像生成

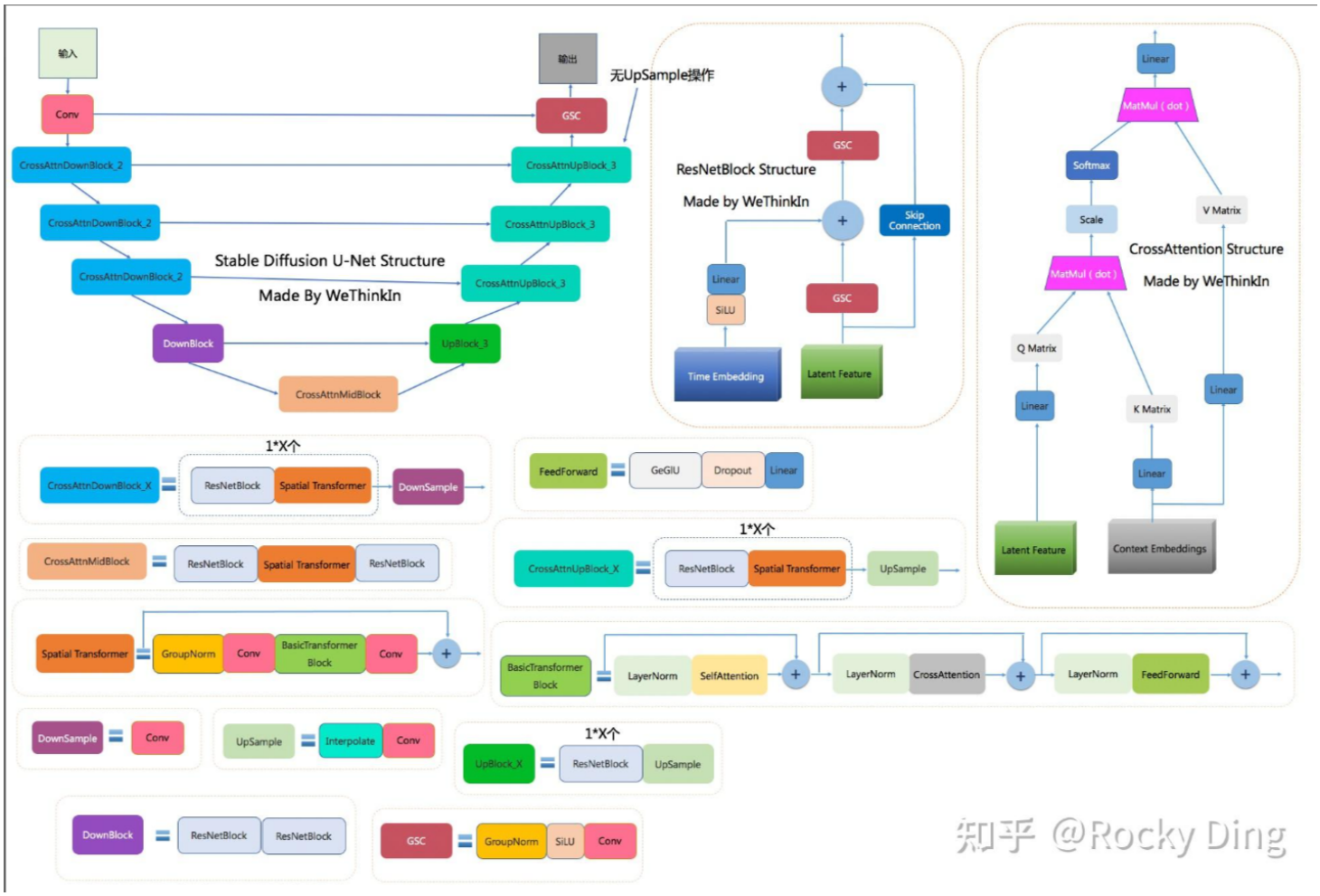
图像优化模块

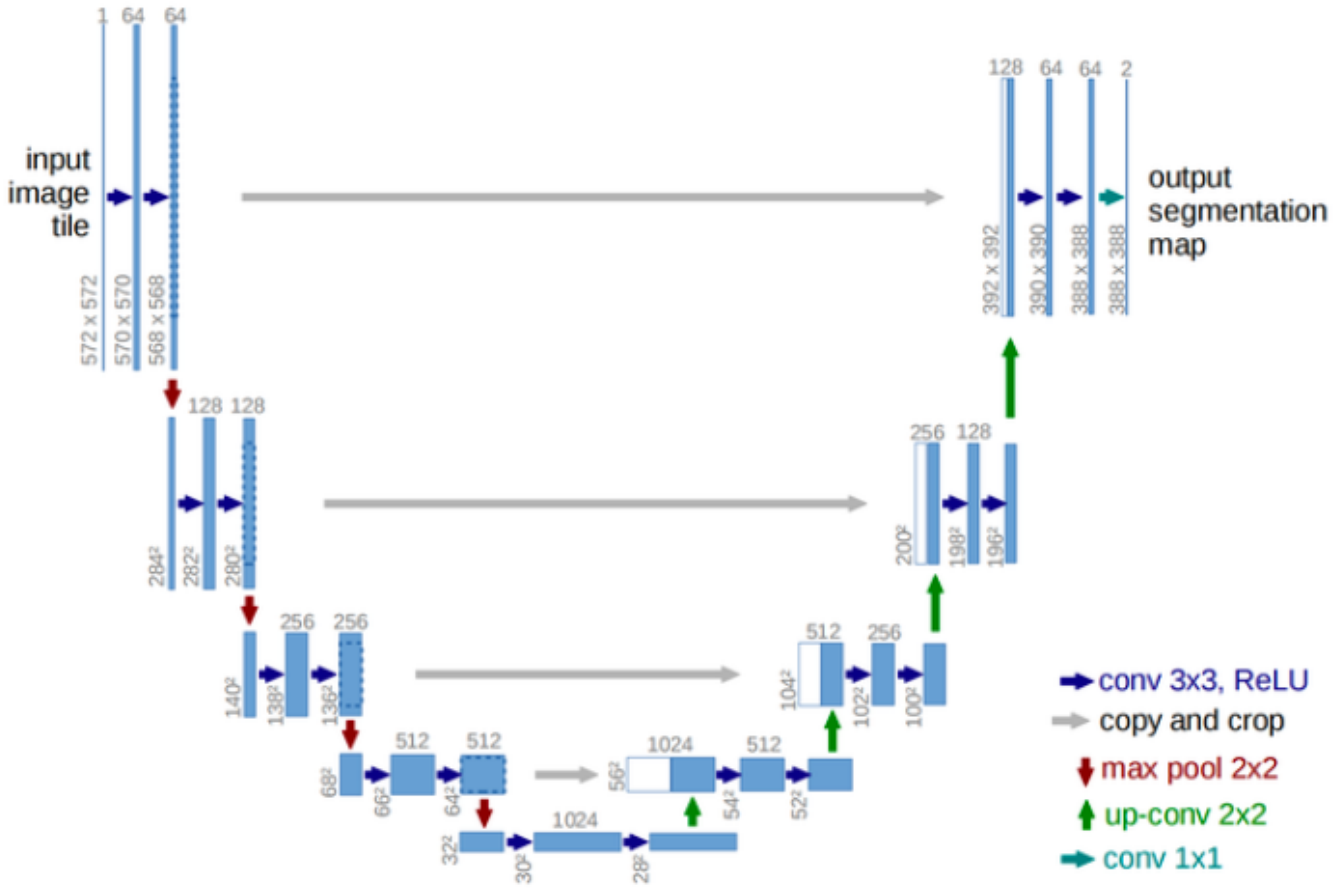
图像优化部分优2两部分组成，分别是U-NET和一个Scheduler








输入
Conv
CrossAttnDownBlock_2
CrossAttnDownBlock_2
CrossAttnDownBlock_2
DownBlock
CrossAttnMidBlock
Stable Diffusion U-Net Structure
Made By WeThinkIn
输出
GSC
CrossAttnUpBlock_3
CrossAttnUpBlock_3
CrossAttnUpBlock_3
UpBlock_3
无UpSample操作
ResNetBlock Structure
Made by WeThinkIn
GSC
Skip Connection
Linear
SiLU
GSC
Time Embedding
Latent Feature
Linear
MatMul (dot)
Softmax
Scale
MatMul (dot)
V Matrix
CrossAttention Structure
Made by WeThinkIn
Q Matrix
K Matrix
Linear
Linear
Linear
Latent Feature
Context Embeddings
1*X个
CrossAttnDownBlock_X
ResNetBlock
Spatial Transformer
DownSample
FeedForward
GeGlU
Dropout
Linear
CrossAttnMidBlock
ResNetBlock
Spatial Transformer
ResNetBlock
1*X个
CrossAttnUpBlock_X
ResNetBlock
Spatial Transformer
UpSample
Spatial Transformer
GroupNorm
Conv
BasicTransformer Block
Conv
BasicTransformer Block
LayerNorm
SelfAttention
LayerNorm
CrossAttention
LayerNorm
FeedForward
DownSample
Conv
UpSample
Interpolate
Conv
1*X个
UpBlock_X
ResNetBlock
UpSample
DownBlock
ResNetBlock
ResNetBlock
GSC
GroupNorm
SiLU
Conv
知乎 @Rocky Ding



致谢

苏剑南

Generative Modeling by Estimating Gradients of the Data Distribution

Understanding Diffusion Models: A Unified Perspective

以及我参考的所有资料创作者