Contents

1 Denoising Diffusion Probabilisitic Models (DDPM)

1.1 Forward Diffusion Process (Inference)

A *forward diffusion process* can be thought of intuitively as turning a data point into noises. Formally, denote the data point sampled from a real distribution as $\boldsymbol{x_0} \sim q(\boldsymbol{x})$. At each time step t, we add a small amount of Gaussian noise to the data point at previous time $\boldsymbol{x_{t-1}}$ to get $\boldsymbol{x_t}$,

$$\boldsymbol{x_0} \to \boldsymbol{x_1} \to \ldots \to \boldsymbol{x_{t-1}} \to \boldsymbol{x_t} \to \ldots \to \boldsymbol{x_T}$$

where T denotes the total time steps (here we assume discrete). The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$ (here we treat as hyperparamater). The forward process is a Markov process

$$q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}) = \prod_{t=1}^{T} q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}) \tag{1}$$

where

$$q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{1-\beta_t}\boldsymbol{x_{t-1}}, \beta_t \boldsymbol{I}) \tag{2}$$

is a diagonal Gaussian with mean $\sqrt{1-\beta_t}\boldsymbol{x_{t-1}}$ and variance $\beta_t \boldsymbol{I}$. Note that we want β_t increases over time

$$\beta_1 < \beta_2 < \ldots < \beta_T$$

and when $T \to \infty$, $\beta_T \to 1$ and the mean and variance becomes 0 and $\boldsymbol{I}$, so

$$q(\boldsymbol{x_T}|\boldsymbol{x_0}) \approx \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

In practice, the total time steps T is in the order of thousands.

1.2 Reverse Diffusion Process (Generative)

We then want to reverse the above process, hopefully being able to generate a true sample from Gaussian noise. Formally, we want to sample Gaussian noise $\boldsymbol{x_T} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and gradually denoise it with model $p_\theta(\boldsymbol{x_{0:T}})$ (see equation 3). Our goal is to recover a believable data point $\boldsymbol{x_0}$. Note that if β_t is sufficiently small (i.e. $\beta_t \ll 1$), the variance $\beta_t \boldsymbol{I}$ of the diagonal Gaussian is small and the reverse process will have the same functional form as the forward process [1]. So $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$ will also be diagonal Gaussian.

$$\boldsymbol{x_0} \leftarrow \boldsymbol{x_1} \leftarrow \ldots \leftarrow \boldsymbol{x_{t-1}} \leftarrow \boldsymbol{x_t} \leftarrow \ldots \leftarrow \boldsymbol{x_T}$$

We model this as

$$p_\theta(\boldsymbol{x_{0:T}}) = p(\boldsymbol{x_T}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}) \tag{3}$$

where

$$p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}) = \mathcal{N}(\boldsymbol{x_{t-1}}; \boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x_t}, t)) \tag{4}$$

[1] Feller, 1949

is a Gaussian model with mean $\boldsymbol{\mu_\theta}$ and variance $\boldsymbol{\Sigma_\theta}$ parameterized by θ, which are also the targets to be learnt in MLP. Note that the mean $\boldsymbol{\mu_\theta}$ and variance $\boldsymbol{\Sigma_\theta}$ are not just functions of $\boldsymbol{x_t}$ but also time step t, meaning the models are time dependent. Also, $p(\boldsymbol{x_T}) = \mathcal{N}(\boldsymbol{x_T}; \boldsymbol{0}, \boldsymbol{I})$ is simply a Gaussian distribution with $\boldsymbol{x_T}$ sampled as Gaussian noise.

1.3 Sampling using reparamaterization

We are going to sample from the distributions in the forward and reverse process. Let's say we want to sample $x \sim \mathcal{N}(\mu, \sigma^2\boldsymbol{I})$. We can instead apply the reparamaterization trick by first sampling $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and then compute

$$x = \mu + \sigma\epsilon$$

Now if we want to sample $\boldsymbol{x_t} \sim q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{1-\beta_t}\boldsymbol{x_{t-1}}, \beta_t\boldsymbol{I})$ in equation (2), we can apply the same trick and get

$$\boldsymbol{x_t} = \sqrt{1-\beta_t}\boldsymbol{x_{t-1}} + \sqrt{\beta_t}\boldsymbol{\epsilon} \tag{5}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Now let $\alpha_t = 1 - \beta_t$, we have $\boldsymbol{x_t} = \sqrt{\alpha_t}\boldsymbol{x_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$. As $\boldsymbol{x_{t-1}} = \sqrt{\alpha_{t-1}}\boldsymbol{x_{t-2}} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}$, plugging in we have

$$\boldsymbol{x_t} = \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\boldsymbol{x_{t-2}} + \underbrace{\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}}_{\sigma_{\boldsymbol{x_{t-1}}|\boldsymbol{x_{t-2}}}}\boldsymbol{\epsilon} + \underbrace{\sqrt{1-\alpha_t}}_{\sigma_{\boldsymbol{x_t}|\boldsymbol{x_{t-1}}}}\boldsymbol{\epsilon}$$

Recall that for two independent random variables X and Y, $Var[X+Y] = Var[X] + Var[Y]$, so the new variance for the two merged Gaussian is

$$\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t) = 1 - \alpha_t\alpha_{t-1}$$

and

$$\boldsymbol{x_t} = \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x_{t-2}} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}$$

Now let $\overline{\alpha}_t = \prod_{i=1}^{t}\alpha_i$. By induction, we can go directly from $\boldsymbol{x_0}$ to $\boldsymbol{x_t}$ with given variance scehdule $\{\beta_t \in (0,1)\}_{t=1}^{T}$ without having to run the forward process by the time step t. That is

$$\boldsymbol{x_t} = \sqrt{\overline{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon} \tag{6}$$

meaning

$$q(\boldsymbol{x_t}|\boldsymbol{x_0}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{\overline{\alpha}_t}\boldsymbol{x_0}, (1-\overline{\alpha}_t)\boldsymbol{I}) \tag{7}$$

is a Gaussian distribution with mean $\sqrt{\overline{\alpha}_t}\boldsymbol{x_0}$ and variance $(1-\overline{\alpha}_t)\boldsymbol{I}$.

1.4 Training Objective

By treating the samples $\boldsymbol{x_{1:T}}$ (except $\boldsymbol{x_0}$) as latent variables, we can apply the objective function in latent variable models, which is the Evidence Lower Bound (ELBO).

Consider the generic setting of a latent variable model, where z denotes a latent variable,

$$p_\theta(x) = \int p_\theta(x, z)dz$$

Applying importance sampling with $q(z)$,

$$p_\theta(x) = \int \frac{q(z)}{q(z)} p_\theta(x, z)dz = E_{q(z)}\left[\frac{p_\theta(x, z)}{q(z)}\right]$$

and as usual we care about maximizing the marginal log-likelihood $\ln p_\theta(x)$ which is intractable in this case, we apply the Jensen's Inequality to get the ELBO

$$\ln p_\theta(x) = \ln E_{q(z)}\left[\frac{p_\theta(x, z)}{q(z)}\right] \geq E_{q(z)}\left[\ln \frac{p_\theta(x, z)}{q(z)}\right] = \text{ELBO}$$

If $q(z) = p_\theta(z|x)$ (the posterior), then the ELBO *becomes*

$$E_{p_\theta(z|x)}\left[\ln \frac{p_\theta(z|x)p_\theta(x)}{p_\theta(z|x)}\right] = \ln p_\theta(x)$$

which is the marginal log-likelihood.

Now with the given data point $\boldsymbol{x_0}$, we can treat $\boldsymbol{x_{1:T}}$ as latent variables z and $q(z)$ simply as the posterior $q(z|x)$, we get the ELBO as

$$E_q\left[\ln \frac{p_\theta(\boldsymbol{x_{0:T}})}{q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0})}\right]$$

By equation (3) and (1), we can factor the ELBO into components corresponding to each time step t,

$$E_q\left[\ln p(\boldsymbol{x_T}) + \sum_{t\geq 1} \ln \frac{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})}\right] \tag{8}$$

By equation (2), $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}) = \mathcal{N}(\boldsymbol{x_t}; \sqrt{1-\beta_t}\boldsymbol{x_{t-1}}, \beta_t \boldsymbol{I})$ in the forward diffusion process, so the expectation can be sampled from the forward process.

We will now further expand the ELBO into three terms with the Bayes rule. We first take out the part in the summation for the initial time step,

$$\begin{aligned}
\text{ELBO} &= E_q\left[\ln p(\boldsymbol{x_T}) + \sum_{t\geq 1} \ln \frac{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})}\right] \\
&= E_q\left[\ln p(\boldsymbol{x_T}) + \sum_{t>1} \ln \frac{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})} + \ln \frac{p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1})}{q(\boldsymbol{x_1}|\boldsymbol{x_0})}\right]
\end{aligned}$$

then with Bayes rules (conditioned on $\boldsymbol{x_0}$),

$$q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) = \frac{q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{x_0})q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})}{q(\boldsymbol{x_t}|\boldsymbol{x_0})} \tag{9}$$

we plug in the forward posterior $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ for $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$,

$$\text{ELBO} = E_q\left[\ln p(\boldsymbol{x_T}) + \sum_{t>1} \ln \frac{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})} \cdot \frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})}{q(\boldsymbol{x_t}|\boldsymbol{x_0})} + \ln \frac{p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1})}{q(\boldsymbol{x_1}|\boldsymbol{x_0})}\right]$$

Note that

$$\sum_{t>1} \ln \frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})}{q(\boldsymbol{x_t}|\boldsymbol{x_0})} = \ln \prod_{t=2}^{T} \frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0})}{q(\boldsymbol{x_t}|\boldsymbol{x_0})} = \ln \frac{q(\boldsymbol{x_1}|\boldsymbol{x_0})}{q(\boldsymbol{x_T}|\boldsymbol{x_0})}$$

and so the ELBO finally becomes

$$\text{ELBO} = E_q\left[\ln \frac{p(\boldsymbol{x_T})}{q(\boldsymbol{x_T}|\boldsymbol{x_0})} + \sum_{t>1} \ln \frac{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})} + \ln p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1})\right]$$

Rewriting it as KL-divergence (with a negative sign pulled out), we have

$$\begin{aligned}\text{ELBO} = -E_q\Bigg[&\underbrace{D_{\text{KL}}(q(\boldsymbol{x_T}|\boldsymbol{x_0}) \parallel p(\boldsymbol{x_T}))}_{L_T} \\ &+ \sum_{t>1} \underbrace{D_{\text{KL}}(q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) \parallel p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}))}_{L_{t-1}} \underbrace{- \ln p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1})}_{L_0}\Bigg] \qquad (10)\end{aligned}$$

We now look closer into each term. As discussed in previous section, $q(\boldsymbol{x_T}|\boldsymbol{x_0}) \approx \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ when $t \to \infty$ (i.e. $\beta_T \to 1$) and that $p(\boldsymbol{x_T}) = \mathcal{N}(\boldsymbol{x_T}; \boldsymbol{0}, \boldsymbol{I})$ is also Gaussian, so we can expect the first $L_T \to 0$ when $T \to \infty$. Also, both distributions q and p are *not* parameterized by θ, so we can safely ignore the first L_T term.

1.4.1 The L_{t-1} term

Note that the two distributions in the L_{t-1} term has the same form $\boldsymbol{x_{t-1}}|\boldsymbol{x_t}$ (the q in forward process conditioned more on $\boldsymbol{x_0}$). This is primarily the reason we applied Bayes rule to expand the ELBO. We now show $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ is a tractable Gaussian, meaning the L_{t-1} term is a divergence between two Gaussians, for which there is a closed form solution for it (though we will simplify it later on).

Forward posterior

By equation (2) and (7), each distribution in the Bayes rule equation (9) is just Gaussian. We expand them with the Gaussian exponentials,

$$\begin{aligned}&q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) \\ &= \frac{1}{N} \exp\left(-\frac{1}{2}\left(\frac{(\boldsymbol{x_t} - \sqrt{\alpha_t}\boldsymbol{x_{t-1}})^2}{\beta_t} + \frac{(\boldsymbol{x_{t-1}} - \sqrt{\overline{\alpha}_{t-1}}\boldsymbol{x_0})}{1 - \overline{\alpha}_{t-1}} - \frac{(\boldsymbol{x_t} - \sqrt{\overline{\alpha}_t}\boldsymbol{x_0})}{1 - \overline{\alpha}_t}\right)\right)\end{aligned}$$

where the normalizing constant $N = \sqrt{2\pi\beta_t} \cdot \sqrt{2\pi(1 - \overline{\alpha}_{t-1})}/\sqrt{2\pi(1 - \overline{\alpha}_t)}$. Expand the power terms with $(a+b)^2 = a^2 + 2ab + b^2$ and grouping $\boldsymbol{x_{t-1}}$ degree terms we get

$$\frac{1}{N} \exp\left(-\frac{1}{2}\left[\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \overline{\alpha}_{t-1}}\right)\boldsymbol{x_{t-1}^2} - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x_t} + \frac{2\sqrt{\overline{\alpha}_{t-1}}}{1 - \overline{\alpha}_{t-1}}\boldsymbol{x_0}\right)\boldsymbol{x_{t-1}} + C(\boldsymbol{x_t}, \boldsymbol{x_0})\right]\right)$$

which is a quadratic equation on $\boldsymbol{x_{t-1}}$ with $C(\boldsymbol{x_t}, \boldsymbol{x_0})$ as constant. Let

$$J = \frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}} = \frac{\alpha_t - \alpha_t\overline{\alpha}_{t-1} + 1 - \alpha_t}{\beta_t(1-\overline{\alpha}_{t-1})} = \frac{1-\overline{\alpha}_t}{\beta_t(1-\overline{\alpha}_{t-1})}$$

and factor J out (note that $J = 1/J^{-1}$) we get,

$$\frac{1}{N}\exp\left(-\frac{1}{2}\frac{1}{J^{-1}}\left[\boldsymbol{x}_{\boldsymbol{t-1}}^2 - 2\left(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x_t} + \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}\boldsymbol{x_0}\right)J^{-1}\boldsymbol{x_{t-1}} + C(\boldsymbol{x_t}, \boldsymbol{x_0})J^{-1}\right]\right)$$

we can further factor the square bracket terms,

$$\frac{1}{N}\exp\left(-\frac{1}{2}\frac{1}{J^{-1}}\left[\boldsymbol{x_{t-1}} - \left(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x_t} - \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}\boldsymbol{x_0}\right)J^{-1}\right]^2\right)\underbrace{\exp\left(-\frac{1}{2}\frac{1}{J^{-1}}D(\boldsymbol{x_t}, \boldsymbol{x_0})\right)}_{\text{constant}}$$

where all the non-$\boldsymbol{x_{t-1}}$ terms are grouped into $D(\boldsymbol{x_t}, \boldsymbol{x_0})$ and taken out in the exponential as constant. Now we can see the variance, denote as $\tilde{\beta}_t$, is

$$\tilde{\beta}_t = J^{-1} = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t \tag{11}$$

and the mean, denote as $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x_t}, \boldsymbol{x_0})$, is

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x_t}, \boldsymbol{x_0}) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\boldsymbol{x_t} + \frac{\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_{t-1}}\boldsymbol{x_0}\right)J^{-1} \\ &= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{x_t} + \frac{\beta_t\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_t}\boldsymbol{x_0}\end{aligned} \tag{12}$$

Recall in equation (6) we have $\boldsymbol{x_0} = \frac{1}{\sqrt{\overline{\alpha}_t}}\left(\boldsymbol{x_t} - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}\right)$, plugging back to equation (12), we have $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ now only as a function of $\boldsymbol{x_t}$ and $\boldsymbol{\epsilon}$,

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}} &= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{x_t} + \frac{\beta_t\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_t}\frac{1}{\sqrt{\overline{\alpha}_t}}\left(\boldsymbol{x_t} - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}\right) \\ &= \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\boldsymbol{x_t} + \frac{1-\alpha_t}{(1-\overline{\alpha}_t)\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}\right) \\ &= \frac{\alpha_t - \overline{\alpha}_t + 1 - \alpha_t}{(1-\overline{\alpha}_t)\sqrt{\alpha_t}}\boldsymbol{x_t} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon} \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}\right)\end{aligned} \tag{13}$$

To wrap things up, we have the forward posterior

$$q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) = \mathcal{N}(\boldsymbol{x_{t-1}}; \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}, \tilde{\beta}_t\boldsymbol{I}) \tag{14}$$

as a Gaussian with mean $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ and variance $\tilde{\beta}_t\boldsymbol{I}$ following equation (13) and (11) respectively. Intuitively, the L_{t-1} term is essentially "pushing" the (reverse) model distribution $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$ towards the (forward) posterior $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$ (conditioned on $\boldsymbol{x_0}$).

Model distribution

We now turn to the model distribution $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$. In the original paper [2], the authors set fixed $\boldsymbol{\Sigma}_\theta(\boldsymbol{x_t}, t) = \sigma_t^2\boldsymbol{I}$ as a time dependent constant and only learn $\boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t)$ in the model $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}) = \mathcal{N}(\boldsymbol{x_{t-1}}; \boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x_t}, t))$. Experimentally, settting $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$. gives similar results to setting $\sigma_t^2 = \beta_t$.

To train $\boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t)$ to model $\boldsymbol{\tilde{\mu}_t} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}\right)$ as in equation (13), note that $\boldsymbol{x_t}$ is given as input during training, we can parameterize the noise term $\boldsymbol{\epsilon}$ with θ as $\boldsymbol{\epsilon}_\theta(\boldsymbol{x_t}, t)$ (predicting the noise $\boldsymbol{\epsilon}$ from $\boldsymbol{x_t}$) and have

$$\boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x_t}, t)\right) \tag{15}$$

To sample $\boldsymbol{x_{t-1}} \sim p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$, we can apply reparameterization to compute

$$\boldsymbol{x_{t-1}} = \boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t) + \sigma_t\boldsymbol{z} \tag{16}$$

where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Since we are modelling only the mean, we can parameterized the L_{t-1} loss term as an expectation of the squared error of the forward posterior mean $\boldsymbol{\tilde{\mu}_t}(\boldsymbol{x_t}, \boldsymbol{x_0})$ and the reverse model mean $\boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t)$,

$$L_{t-1} = E_q\left[\frac{1}{2\sigma_t^2} \parallel \boldsymbol{\tilde{\mu}_t}(\boldsymbol{x_t}, \boldsymbol{x_0}) - \boldsymbol{\mu}_\theta(\boldsymbol{x_t}, t) \parallel^2\right] \tag{17}$$

Note that this expectation is very similar to expanding the KL-divergence (i.e. the log-likelihood of the Gaussian forms of $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$ and $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0})$). We can further simplify it by substituting equation (13) and (15),

$$L_{t-1} = E_{\boldsymbol{x_0},\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha}_t)} \parallel \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x_t}, t)^2 \parallel\right] \tag{18}$$

where $\boldsymbol{x_t}$ is reparameterized in (6) as $\boldsymbol{x_t} = \sqrt{\overline{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

Finally, the authors found experimentally that it is (1) beneficial to the sample quality and (2) simpler to implement by dropping the weighing terms and simply have the loss term as

$$L_{t-1}^{\text{simple}}(\theta) = E_{t,\boldsymbol{x_0},\boldsymbol{\epsilon}}\left[\parallel \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\boldsymbol{x_t}, t\right)^2 \parallel\right] \tag{19}$$

where $\boldsymbol{x_t}$ is reparameterized as above and $t \sim \text{Uniform}(\{1, \ldots, T\})$. Now the squared error depends on the noise $\boldsymbol{\epsilon}$ and the predicted noise $\boldsymbol{\epsilon}_\theta$.

1.4.2 The L_0 term

We assume the data point $\boldsymbol{x_0}$ as an image with pixel values $\in \{0, 1, \ldots, 255\}$ which is scaled linearly to $[-1, 1]$. This ensures the reverse process operates on the same scale

[2]Ho et al., 2020

starting from the Gaussian prior $p(\boldsymbol{x_T})$. We set the last term of the reverse process to an independent *discrete* decoder derived from Gaussian $\mathcal{N}(\boldsymbol{x_0}; \boldsymbol{\mu}_\theta(\boldsymbol{x_1}, 1), \sigma_1^2 \boldsymbol{I})$.

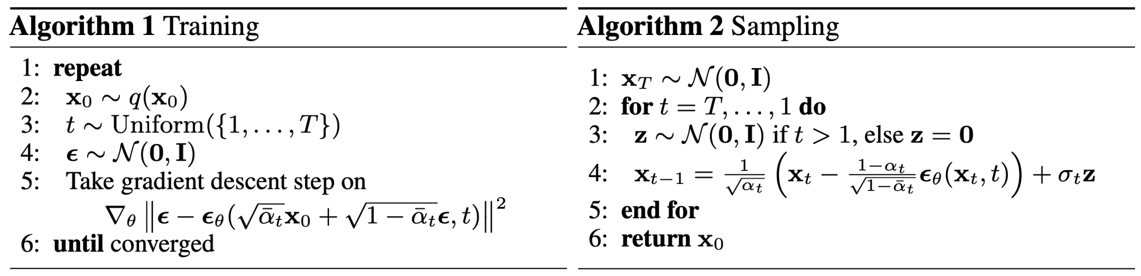
$$p(\boldsymbol{x_0}|\boldsymbol{x_1}) = \prod_{i=1}^{D} \int_{\delta_-(\boldsymbol{x_0}^i)}^{\delta_+(\boldsymbol{x_0}^i)} \mathcal{N}(x; \mu_\theta^i(\boldsymbol{x_1}, 1), \sigma_1^2) dx$$

where D denotes the dimensionality of the image data, i denotes a specific coordinate with pixel value $\in [-1, 1]$ and

$$\delta_+(x) = \begin{cases} \infty & \text{for } x = 1 \\ x + \frac{1}{255} & \text{for } x < 1 \end{cases} \qquad \delta_-(x) = \begin{cases} -\infty & \text{for } x = -1 \\ x - \frac{1}{255} & \text{for } x > -1 \end{cases}$$

One can think of the definite integral as finding the sum of probability within the discretized bound $\left(\boldsymbol{x_0}^i - \frac{1}{255}, \boldsymbol{x_0}^i + \frac{1}{255}\right)$ for one pixel, where $\boldsymbol{x_0}^i$ denotes the pixel value $\in [-1, 1]$ for a given image data $\boldsymbol{x_0}$. Then we compute the joint conditional probability $p(\boldsymbol{x_0}|\boldsymbol{x_1})$ by multiplying all such definite integrals for the full dimension D.

1.5 Algorithm



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \text{Uniform}(\{1, \ldots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```

Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Figure 1.1: Training and Sampling algorithm (Image source: Ho et al., 2020)

1.6 Connection to Langevin dynamics in score matching

1.7 Parameterization of β_t

1.8 Parameterization of the reverse process variance Σ_θ

1.9 Interpolation of images

2 Noise Conditional Score Network (NCSN)

Given a dataset $\{\boldsymbol{x_i} \in \mathbb{R}^D\}_{i=1}^N$, where D denotes the dimensionality of the data $\boldsymbol{x_i}$ and N denotes the number of data, such that $\boldsymbol{x_i} \sim p(\boldsymbol{x})$, the unknown underlying probability distribution. To model $p(\boldsymbol{x})$, we can parameterize it with θ. Let $f_\theta(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$. Then $p_\theta(\boldsymbol{x})$ can be defined as

$$p_\theta(\boldsymbol{x}) = \frac{e^{f_\theta(\boldsymbol{x})}}{Z_\theta}$$

where $Z_\theta > 0$ is a normalizing constant dependent on θ to ensure $\int p_\theta(\boldsymbol{x}) d\boldsymbol{x} = 1$. Such $f_\theta(\boldsymbol{x})$ is called an unnormalized probabilistic model or energy-based model [3].

As usual, we want to maximize the log-likelihood $\log p_\theta(\boldsymbol{x})$. But this involves the generally intractable normalizing constant Z_θ. One way to side-step this constant Z_θ is to consider the score function.

2.1 Score Matching

The (Stein) *score function* of a probability density $p(x)$ is given as

$$\boldsymbol{s}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) \tag{1}$$

which is simply the *gradient of the log-likelihood.* This is a vector field pointing in the direction where the log-likelihood increases the most. We then learn a score model $\boldsymbol{s_\theta}(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}^D$ to approximate the true score function $s(\boldsymbol{x})$ such that

$$\boldsymbol{s_\theta}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}) - \underbrace{\nabla_{\boldsymbol{x}} \log Z_\theta}_{0} = \nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}) \tag{2}$$

and hopefully through training $\boldsymbol{s_\theta}(\boldsymbol{x}) \approx \boldsymbol{s}(\boldsymbol{x})$. This is called a *score-based model.*

Given a probability density $p(\boldsymbol{x})$, one can compute the score function $\boldsymbol{s}(\boldsymbol{x})$ by taking the derivative with respect to the random vector $\boldsymbol{x}$. Conversely, one can compute $p(\boldsymbol{x})$ when given $s(\boldsymbol{x})$ by

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{x_0}) + \int_0^1 \boldsymbol{s}(\boldsymbol{x_0} + t(\boldsymbol{x} - \boldsymbol{x_0}))^T (\boldsymbol{x} - \boldsymbol{x_0})\, dt \tag{3}$$

where $\log p(\boldsymbol{x_0})$ can be determined from the normalizing constant

$$\int p(\boldsymbol{x})\, d\boldsymbol{x} = 1.$$

To make sense of (3), let $\boldsymbol{y} = \log p(\boldsymbol{x_0} + t(\boldsymbol{x} - \boldsymbol{x_0}))$ and $\boldsymbol{z} = \boldsymbol{x_0} + t(\boldsymbol{x} - \boldsymbol{x_0})$, then

$$\frac{d\boldsymbol{y}}{d\boldsymbol{z}} = \boldsymbol{s}(\boldsymbol{x_0} + t(\boldsymbol{x} - \boldsymbol{x_0})), \qquad \frac{d\boldsymbol{z}}{dt} = \boldsymbol{x} - \boldsymbol{x_0}$$

[3] In the context of physics, the exponential part is often written as $-f_\theta(x)$, deonting the energy. But the negative sign is of no practical interest to us here.

and one can view the integrand as

$$\frac{d\boldsymbol{y}}{dt} = \frac{d\boldsymbol{y}}{d\boldsymbol{z}} \cdot \frac{d\boldsymbol{z}}{dt}$$

Therefore, $\log p(\boldsymbol{x}) - \log p(\boldsymbol{x_0})$ is the result from the fundamental theorem of calculus by substituting $t = 1$ and $t = 0$ into $\boldsymbol{y} = \log p(\boldsymbol{x} + t(\boldsymbol{x} - \boldsymbol{x_0}))$.

2.1.1 Training Objective

The objective is simple. We minimize the *Fisher divergence*

$$J(\boldsymbol{\theta}) = \frac{1}{2} E_p \left[\| \ \boldsymbol{s}(\boldsymbol{x}) - \boldsymbol{s}_\theta(\boldsymbol{x}) \ \|_2^2 \right] \tag{4}$$

which is just the 2-norm between the true score function $\boldsymbol{s}(\boldsymbol{x})$ and the score model $\boldsymbol{s}_\theta(\boldsymbol{x})$. Thus, our score-matching estimator of the parameter $\boldsymbol{\theta}$ is simply

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Directly computing $J(\boldsymbol{\theta})$ is infeasible because the true score function $\boldsymbol{s}(\boldsymbol{x})$ is unknown (that's why we do score matching in the first place). A. Hyvarinen, (2005) showed that with some derivation, (4) does not depend on $\boldsymbol{s}(\boldsymbol{x})$. To simplify discussion, let's rewrite it in $D = 1$,

$$\frac{1}{2} E_p \left[(\nabla_x \log p(x) - \nabla_x f_\theta(x))^2 \right]$$

Expanding and rewriting the expectations (assume finite) as integral,

$$\underbrace{\frac{1}{2} \int p(x)(\nabla_x \log p(x))^2 \, dx}_{A} + \underbrace{\frac{1}{2} \int p(x)(\nabla_x f_\theta(x))^2 \, dx}_{B} \underbrace{- \int p(x) \nabla_x \log p(x) \nabla_x f_\theta(x) \, dx}_{C}$$

The last term C can be derived as (assume $p(x)$ is differentiable)

$$C = -\int p(x) \left(\frac{1}{p(x)} \nabla_x p(x) \right) \nabla_x f_\theta(x) \, dx = -\int \nabla_x p(x) \nabla_x f_\theta(x) \, dx$$

Applying integration by parts with $d(p(x)) = \nabla_x p(x) dx$,

$$C = -p(x) \nabla_x f_\theta(x) \Big|_{-\infty}^{\infty} + \int p(x) \nabla_x^2 f_\theta(x) \, dx$$

If we assume $p(x) \to 0$ when $|x| \to \infty$, then the first term $= 0$ and we are left with the second term. Note that A is not parameterized by θ, so A is just a constant k (i.e. an expectation with the empircal data). Putting everything together we have

$$\begin{aligned} &\frac{1}{2} \int p(x)(\nabla_x f_\theta(x))^2 \, dx + \int p(x) \nabla_x^2 f_\theta(x) \, dx + k \\ &= E_p \left[\frac{1}{2} (\nabla_x f_\theta(x))^2 + \nabla_x^2 f_\theta(x) \right] + k \end{aligned} \tag{5}$$

For multi-dimension D, (5) can be generalized as

$$E_p\left[\frac{1}{2}\parallel \nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}) \parallel_2^2 + \operatorname{tr}\left(\nabla_{\boldsymbol{x}}^2 f_\theta(\boldsymbol{x})\right)\right] + k \tag{6}$$

where $\nabla_{\boldsymbol{x}}^2$ denotes the Hessian with respect to $\boldsymbol{x}$. This is the score matching objective that does not involve estimating the true score $s(\boldsymbol{x})$ [4]. If we do not care about $f_\theta(\boldsymbol{x})$ and directly model $s_\theta(\boldsymbol{x})$ (perhaps with a neural network), then the objective is better expressed as

$$E_p\left[\frac{1}{2}\parallel \boldsymbol{s}_\theta(\boldsymbol{x}) \parallel_2^2 + \operatorname{tr}\left(\nabla_{\boldsymbol{x}} \boldsymbol{s}_\theta(\boldsymbol{x})\right)\right] + k \tag{7}$$

Notice we have assumed three conditions (back to $\mathbb{R}^D$): (1) finite expectations, (2) differentiable $p(\boldsymbol{x})$, and (3) $p(\boldsymbol{x}) \to 0$ as $|x| \to 0$. They are easily satisfied in practice. For (3), with large datasets, we can apply CLT and conclude $p(\boldsymbol{x})$ to be Gaussian.

2.1.2 Denoising Score Matching

The problem with 7 is that it involves the computation of the trace of the Hessian matrix, which makes it an unscalable method (i.e. computationally intensive). A way to deal with it is the denoising score matching method.

We perturb a given data $\boldsymbol{x}$ to form $\tilde{\boldsymbol{x}}$. Then $\tilde{\boldsymbol{x}} \sim q(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$.

$$\begin{aligned}
&\frac{1}{2} E_{\tilde{\boldsymbol{x}} \sim p}\left[\parallel \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}) - \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}}) \parallel_2^2\right] \\
&= \frac{1}{2} E_{\boldsymbol{x} \sim p(\boldsymbol{x})} E_{\tilde{\boldsymbol{x}} \sim q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})}\left[\parallel \boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}) - \nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) \parallel_2^2\right]
\end{aligned}$$

where $q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \sigma^2 \boldsymbol{I})$ and $q(\tilde{\boldsymbol{x}}) = \int p(\boldsymbol{x}) q_\sigma(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) d\boldsymbol{x}$.

2.1.3 Sliced Score Matching

$$\frac{1}{2} E_{\boldsymbol{v} \sim p(\boldsymbol{v})} E_{\boldsymbol{x} \sim p(\boldsymbol{x})}\left[\left(\boldsymbol{v}^T s(\boldsymbol{x}) - \boldsymbol{v}^T s_\theta(\boldsymbol{x})\right)^2\right]$$

where $\boldsymbol{v}$ denotes the random projection vector with distribution $p(\boldsymbol{v})$. By the same integration by parts as the generic score matching, we have

$$E_{\boldsymbol{v} \sim p(\boldsymbol{v})} E_{\boldsymbol{x} \sim p(\boldsymbol{x})}\left[\frac{1}{2}\left(\boldsymbol{v}^T \boldsymbol{s}_\theta(\boldsymbol{x})\right)^2 + \boldsymbol{v}^T \nabla_{\boldsymbol{x}} \boldsymbol{s}_\theta(\boldsymbol{x}) \boldsymbol{v}\right]$$

The advantages of using a score model is (1) there is no need for adversarial optimization (like GANs), (2) there is no restriction on the form of the model $s_\theta(\boldsymbol{x})$

[4]See A. Hyvarinen, 2005 for details of proof for the multivariate case. The author showed in the paper, using multivariate Gaussian density as example, that score matching gives exactly the same estimator as maximum likelihood estimation (MLE). This shows the consistency of score matching as MLE.

2.2 Sampling with Langevin Dynamics

$$\tilde{\boldsymbol{x}}_{t+1} \leftarrow \tilde{\boldsymbol{x}}_t + \frac{\epsilon}{2}\boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}_t) + \underbrace{\sqrt{\epsilon}\boldsymbol{z}_t}_{\text{noise}}$$

where $\boldsymbol{z}_t, \tilde{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{I})$ with $t = 0, 1, \ldots, T$. As $T \to \infty$ and $\epsilon \to 0$, $\tilde{\boldsymbol{x}}_T \sim p(\boldsymbol{x})$. Intuitively, we can think of the gradient $\boldsymbol{s}_\theta(\tilde{\boldsymbol{x}}_t)$ as a directional guide for $\tilde{\boldsymbol{x}}_t$ to move closer to $\tilde{\boldsymbol{x}}_{t+1}$, while $\frac{\epsilon}{2}$ is a scaling controlling the magnitude of movement and the noise term gives the stochastic nature of the sampling.

2.3 Challenges and Pitfalls

(a) Manifold Hypothesis

(b) Low Density Region

(c) SLowing mixing of Langevin dynamics between data modes

2.4 Noise Perturbation and annealed Langevin Dynamics

To work around with the pitfalls, we introduce the idea of perturbing the data points with Gaussian noise. We set a specific number of time steps T, and with increasing time step, we perturb the data points with increasing noise, until we obtain a full Gaussian noise. (Here, the introduction of noise and time step connects score models to diffusion models.)

Formally, we perturb the data with isotropic Guassian noise and let there be a total of L increasing standard deviations $\sigma_1 < \sigma_2 < \cdots < \sigma_L$. Hence, we obtain a noise-perturbed distribution

$$p_{\sigma_i}(\boldsymbol{x}) = \int p(\boldsymbol{y})\mathcal{N}(\boldsymbol{x}; \boldsymbol{y}, \sigma_i^2\boldsymbol{I})\, d\boldsymbol{y}$$

To draw samples from this $p_{\sigma_i}(\boldsymbol{x})$, we can apply the reparameterization trick: we first draw $\boldsymbol{x} \sim p(\boldsymbol{x})$ and then compute $\boldsymbol{x} + \sigma_i\boldsymbol{z}$ with $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$.

Now, there are mutiple score functions, i.e. $\nabla_{\boldsymbol{x}} \log p_{\sigma_i}(\boldsymbol{x})$ for each σ_i, and we train a noise conditional score model $\boldsymbol{s}_\theta(\boldsymbol{x}, i)$ (also called *Noise Conditional Score Network*, or *NCSN*, when we parameterize it with a neural network) for all $i = 1, 2, \ldots, L$. The training objective is a weighted sum of Fisher divergence for all i which is a slight modifcation of (4),

$$\sum_{i=1}^{L} \lambda(i) E_{p_{\sigma_i}} \left[\| \nabla_{\boldsymbol{x}} \log p_{\sigma_i}(\boldsymbol{x}) - \boldsymbol{s}_\theta(\boldsymbol{x}, i) \|_2^2 \right] \tag{8}$$

where $\lambda(i) \in \mathbb{R}^+$ is a positive weighting function, often chosen simply as the variance σ_i^2. This objective (8) can also be obtimized with score matching, similar to the naive score model with objective (4).

2.5 Stochastic Differential Equation (SDE) and reverse SDE

By generalizing the number of time steps to infinity, we are dealing with stocahstic differential equations. We can then obtain sample with higher quality, compute exact log-likelihood and controllable generation for inverse problems.

Perturbing data with stochastic process for $t : 0 \to T$,

$$d\boldsymbol{x_t} = \boldsymbol{f}(\boldsymbol{x_t}, t)\, dt + g(t)\, d\boldsymbol{w_t} \tag{9}$$

where $\boldsymbol{f}(\cdot, t) : \mathbb{R}^D \to \mathbb{R}^D$ is a vector-valued function called the (deterministic) drift coefficient, $g(t) : [0, 1] \to \mathbb{R}$ is a real function called the diffusion coefficient, and $d\boldsymbol{w_t}$ can be viewed as infinitesimal white noise.

Generation via reverse stochastic process for $t : T \to 0$,

$$d\boldsymbol{x_t} = -\sigma(t)^2 \boldsymbol{s}_\theta(\boldsymbol{x}, t)\, dt + \sigma(t)\, d\bar{\boldsymbol{w}}_{\boldsymbol{t}} \tag{10}$$

where $\boldsymbol{s}_\theta(\boldsymbol{x}, t) = \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x_t})$ is the score function at time t.

Sampling is done numerical SDE solver, called the Euler-Maruyama method.

$$\begin{aligned} \boldsymbol{x} &\leftarrow \boldsymbol{x} - \sigma(t)^2 \boldsymbol{s}_\theta(\boldsymbol{x}, t)\Delta t + \sigma(t)\boldsymbol{z} \\ t &\leftarrow t + \Delta t \end{aligned} \tag{11}$$

where $\boldsymbol{z} \sim \mathcal{N}(0, |\Delta t|\boldsymbol{I})$.

2.6 Converting SDE to ODE

We can think of it as a normalizing flow with *continuous time* and *infinite depth.*

2.7 Connection to Diffusion Models

3 Applications