Lecture Notes: Diffusion Generative Models

1 Generative Model

What is generative model? Consider a generative model that produces realistic image of human face. Suppose the model always outputs the same specific image of Albert, then we may not want such generative model.

A good generative model should generate realistic output (of course), however, at the same time, it should be able to mimic the diversity of input. Your human face generator should be able to output of various types of faces (race, age, sex, etc). In other words, generative model is not only learning realistic samples, it is learning the distribution of the dataset.

On the other hand, sampling (or generating sample) is not always trivial even when we know the distribution. Suppose you have the following density:

$$f(x) = \frac{1}{\pi} \frac{\gamma}{x^2 + \gamma^2} \tag{1}$$

for some $\gamma > 0$. This is called Cauchy distribution with parameter $\gamma > 0$.

Exercise 1. *Prove that $f(x)$ is a valid probability density.*

In this case, it is non-trivial to generate samples $X \sim f(x)$ even when we have a complete characterization of distribution. However, in the scalar case, we can use *inverse transform sampling* based on the uniform distribution.

Suppose F_X be a cumulative density function of random variable X, and $U \sim Unif[0, 1]$ be an uniform distribution. Since F_X is non-decreasing and $\Pr(U \leq u) = u$ for $0 \leq u \leq 1$, we have:

$$\Pr(F_X^{-1}(U) \leq x) = \Pr(U \leq F_X(x)) = F_X(x),$$

which implies:

$$F_X^{-1}(U) \sim f(x).$$

In other words, if we sample uniform random variable U, then apply F_X^{-1}, then we can achieve a sample that has a distribution $f(x)$.

Then how do we generate a uniform random variable? One way is to approximate it using binary expansion:

$$U \approx 0.U_1U_2U_3\cdots U_{n(2)}, \quad \text{where} \quad U_i \sim \text{Bernoulli}\left(\frac{1}{2}\right)$$

By generating a sequence of fair coin flips, we can construct a random number that closely follows the uniform distribution on $[0, 1]$.

However, for multidimensional random vector $\mathbf{X} = (X_1, \ldots, X_n)$, it is hard to sample from distribution even when we have an exact formula of probability density.

In most generative models, which "learns distribution of dataset", does not explicitly learn the probability density function. It often learns a "transformation" from samples (that are easy to generate) to a target distribution (e.g., Generative Adversarial Networks).

Notation: Uppercase X usually denotes a random variable, and lowercase x often denotes a realization of it. For example, a probability density function $f_X(x)$ takes lowercase x as an input, since it provides a density value at specific point x.

2 1D Diffusion Model

2.1 Variance Exploding

For simplicity let start from 1-dimensional case. Let $X \in \mathbb{R}$ be a random variable that we want to learn. The density function $f_X(x)$ is unknown but we have samples (or we can sample X). Consider the following random process (called forward process)

$$X^{(n)} = X^{(n-1)} + Z^{(n)} \tag{2}$$

where $X^{(0)} = X$ and $Z^{(n)} \sim \mathcal{N}(0, \sigma^2)$ is independent Gaussian noise with mean zero and variable σ^2. It is equivalent to where we are gradually adding noise to the input image.

Remark 1. *Note that the forward process does not have to be computed sequentially. We can corrupt a sample $X^{(0)}$ with n-times in a single step*

$$X^{(n)} = X^{(0)} + \sqrt{n}Z \tag{3}$$

where $Z \sim \mathcal{N}(0, \sigma^2)$. This fact is useful in training the denoiser f_θ.

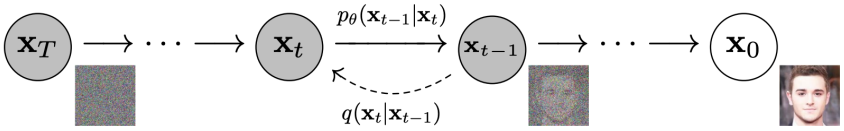

Figure 1: The directed graphical model considered in DDPM [5].

When n grows, the nZ term dominates and $X^{(n)}$ behaves like Gaussian. Yes, this is informal, but we will see the formal proof for variance preserving (VP) process. The above process is called variance exploding (VE) process as variance of $X^{(n)}$ explodes.

For large enough N, we can approximate the distribution of $X^{(N)}$ by

$$X^{(N)} \sim \mathcal{N}(\tilde{\mu}, \tilde{\sigma}). \tag{4}$$

Clearly, $\tilde{\mu} = \mu_0$ and $\tilde{\sigma}^2 = \sigma_0^2 + N\sigma^2$ where μ_0 and σ_0^2 are mean and variance of X_0.

Remark 2. *Estimating $\tilde{\mu}$ and $\tilde{\sigma}$ is much easier than estimating the distribution. Moreover, we will have explicit formula for $\tilde{\mu}$ and $\tilde{\sigma}$ for the VP process.*

The key here is that, we can sample from the distribution $\mathcal{N}(\tilde{\mu}, \tilde{\sigma})$, Let $\tilde{X}^{(N)}$ be the sample from the distribution $\mathcal{N}(\tilde{\mu}, \tilde{\sigma})$. Our goal is to reverse the process from $X^{(N)}$ to obtain $\tilde{X}^{(N-1)}, \ldots, \tilde{X}^{(1)}, \tilde{X}^{(0)}$ and hoping $\tilde{X}^{(0)} \overset{(d)}{\approx} X$ (similar in distribution).

Now, consider the reverse sampling (also referred to as the reverse process). For each step $1 \leq n \leq N$, we aim to model the reverse conditional distribution $P_{X^{(n-1)}|X^{(n)}}$. (From this point forward, we denote the density function of X by P_X rather than f_X.)

For notational simplicity, let $X = \tilde{X}^{(n-1)}$, $Z = \tilde{Z}^{(n)}$, and $Y = \tilde{X}^{(n)}$, so that $Y = X + Z$ and $Y|X \sim \mathcal{N}(0, \sigma^2)$. Our goal is to characterize the distribution of $X|Y$. By Bayes' theorem, we have:

$$P_{X|Y}(x|y) = \frac{P_{Y|X}(y|x)P_X(x)}{P_Y(y)}.$$

To evaluate this expression, we begin by approximating $P_X(x)$ and $P_Y(y)$:

$$\begin{aligned} P_X(x) &= P_X(y) + P_X'(y)(x-y) + O(\|x-y\|^2) \\ &\approx P_X(y) + P_X'(y)(x-y), \end{aligned}$$

where the approximation relies on the assumption that $\sigma^2 \ll 1$.

Next, we approximate the marginal density $P_Y(y)$ as follows:

$$\begin{aligned} P_Y(y) &= \int P_{Y|X}(y|x)P_X(x)\,dx \\ &\approx \int \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)\big(P_X(y) + P'_X(y)(x-y)\big)\,dx \\ &= P_X(y) + P'_X(y)\,\mathbb{E}_{X\sim\mathcal{N}(y,\sigma^2)}[X-y] \\ &= P_X(y), \end{aligned}$$

Substituting these approximations into Bayes' rule yields:

$$\begin{aligned} P_{X|Y}(x|y) &= \frac{P_{Y|X}(y|x)P_X(x)}{P_Y(y)} \\ &\approx \frac{P_{Y|X}(y|x)\,(P_X(y) + P'_X(y)(x-y))}{P_X(y)} \\ &= P_{Y|X}(y|x)\left(1 + \frac{P'_X(y)}{P_X(y)}(x-y)\right) \\ &\approx \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)\exp\left(\frac{P'_X(y)}{P_X(y)}(x-y)\right) \\ &\approx \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left((x-y)^2 - 2\sigma^2\frac{P'_X(y)}{P_X(y)}(x-y) + \left(\sigma^2\frac{P'_X(y)}{P_X(y)}\right)^2\right)\right) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(x - y - \sigma^2\frac{P'_X(y)}{P_X(y)}\right)^2\right). \end{aligned}$$

Noting that $\frac{P'_X(y)}{P_X(y)} = \frac{\partial}{\partial y}\log P_X(y)$, we conclude that

$$X|Y \overset{\text{approx.}}{\sim} \mathcal{N}\left(Y + \sigma^2\frac{\partial}{\partial y}\log P_X(Y),\ \sigma^2\right).$$

This implies that when the noise variance σ^2 is small, the reverse conditional distribution can be well-approximated by a Gaussian. Based on this, we can generate samples from the reverse process via:

$$\tilde{x}^{(n-1)} = \tilde{x}^{(n)} + \sigma^2\frac{\partial}{\partial \tilde{x}^{(n)}}\log P_{\tilde{X}^{(n)}}(\tilde{x}^{(n)}) + \tilde{z}^{(n)},$$

where $\tilde{z}^{(n)} \sim \mathcal{N}(0, \sigma^2)$ is an independent Gaussian noise.

Remark 3. *One might wonder whether we can generate samples by subtracting noise instead of adding it. That is, consider the process:*

$$\tilde{X}^{(n-1)} = \tilde{X}^{(n)} - \tilde{Z}^{(n)}. \tag{5}$$

However, this approach does not constitute a denoising process. Since the noise distribution is symmetric, i.e., $-\tilde{Z}^{(n)} \overset{(d)}{=} \tilde{Z}^{(n)}$*, subtracting noise is distributionally equivalent to adding it. In other words, the process still corresponds to further corrupting the sample with noise, not reversing it.*

Remark 4. *It is different from optimal denoising. For example, let* X *be a random variable and* $Y = X + N$ *where* N *be independent noise. Then, the minimum mean squared estimation that minimizes*

$$\mathbb{E}\left[(\hat{X}(Y) - X)^2\right] \tag{6}$$

is achieved by $\hat{X}(Y) = \mathbb{E}[X|Y]$*, i.e., deterministic. In the above example, the reverse process is different from the optimal denoising step. The key here is "distribution matching" which is seemingly similar to denoising when* σ *is small.*

Exercise 2. *Show that the minimum mean square estimate is* $\hat{X}(Y) = \mathbb{E}[X|Y]$*. You can start from*

$$\mathbb{E}\left[(\hat{X}(Y) - X)^2\right] = \mathbb{E}\left[((\hat{X}(Y) - \mathbb{E}[X|Y]) + (\mathbb{E}[X|Y] - X))^2\right] \tag{7}$$

Consider the single sample $x^{(0)} = x$ and its corrupted version $x^{(N)}$. If we sample from $X^{(N)} = x^{(N)}$ and generate $X^{(0)}$ through optimum denoising, we may not (and will not) get sample that is similar to original x. This is also an indication of the reverse diffusion is not an optimum denoising (in mean squared error sense).

Example 1. *Let* $X \sim \mathcal{N}(0, \sigma_x^2)$ *and* $Y = X + Z$ *where* $Z \sim \mathcal{N}(0, \sigma_z^2)$*. Then, the optimal denoiser is*

$$\hat{X}(Y) = \mathbb{E}[X|Y] \tag{8}$$

$$= \frac{\sigma_x^2}{\sigma_x^2 + \sigma_z^2} Y. \tag{9}$$

Since $Y \sim \mathcal{N}(0, \sigma_x^2 + \sigma_z^2)$*, we have*

$$\mathbb{E}[X|Y] \sim \mathcal{N}\left(0, \frac{\sigma_x^4}{\sigma_x^2 + \sigma_z^2}\right) \tag{10}$$

which does not match to the original distribution of X. However, if we generate

$$\tilde{X} = \mathbb{E}[X|Y] + \tilde{Z} \tag{11}$$

where $\tilde{Z} \sim \mathcal{N}(0, \frac{\sigma_x^2 \sigma_z^2}{\sigma_x^2 + \sigma_z^2})$, then the distribution of $\tilde{X} \sim \mathcal{N}(0, \sigma^2)$ matches to the original distribution.

Remark 5. *As we will see in the simulation results, we do not need any neural networks to be trained for reverse process if we have full information of "score function" $\frac{\partial}{\partial \tilde{x}^{(n)}} \log P_{\tilde{X}^{(n)}}(\tilde{x}^{(n)})$ for all n and all $\tilde{x}^{(n)}$. However, it is (of course) not the case in practice, and that is why we need to train a neural network to learn this score function.*

Suppose we do not have access to the probability density function or the score function defined as

$$s(x^{(n)}; n) = \frac{\partial}{\partial x^{(n)}} \log P_{\tilde{X}^{(n)}}(x^{(n)}),$$

where the score function explicitly depends on n due to the time-dependent marginal distribution $P_{\tilde{X}^{(n)}}$.

In such a case, we can instead train a neural network $f_\theta(x^{(n)}; n)$ to approximate the original (clean) data point $x^{(0)}$ from its noisy counterpart $x^{(n)}$, obtained after n steps of noising. By minimizing the expected squared error between the network output and the original data,

$$\|f_\theta(x^{(n)}; n) - x^{(0)}\|^2,$$

the trained network learns to estimate the conditional expectation:

$$f_\theta(x^{(n)}; n) \approx \mathbb{E}[X^{(0)}|X^{(n)} = x^{(n)}].$$

Notably, this learning process does not require knowledge of the underlying data distribution or the score function.

Exercise 3 (Tweedie's Formula [4])**.** *Let X be a random variable and $Z \sim \mathcal{N}(0, \sigma^2)$ be an independent Gaussian noise. Let $Y = X + Z$ be noise corrupted version of X, show that*

$$\mathbb{E}[X|Y = y] = y + \sigma^2 \frac{\partial}{\partial y} \log P_Y(y). \tag{12}$$

You can start from

$$P_Y(y) = \int P_X(x) P_Z(y - x)\, dz \tag{13}$$

and compare it with $\frac{\partial}{\partial y} P_Y(y)$

Using Tweedie's formula, we have

$$f_\theta(x;n) \approx x + n\sigma^2 \frac{\partial}{\partial x} \log P_{X^{(n)}}(x). \tag{14}$$

Thus, we can obtain an approximation for the score function as

$$s_\theta(x;n) = \frac{f_\theta(x;n) - x}{n\sigma^2}. \tag{15}$$

2.2 Variance Preserving

The above noise addition process will have gigantic variance of X_N, and this is why it is called variance exploding (VE) process. Clearly, adding noise at each time will keep increasing the variance. We can have alternative formula that preserves the variance by scaling. Let

$$X^{(n)} \sim \mathcal{N}\left(\sqrt{1-\beta_n} X^{(n-1)}, \beta_n\right) \tag{16}$$

or

$$X^{(n)} = \sqrt{1-\beta_n} X^{(n-1)} + \sqrt{\beta_n} Z^{(n)} \tag{17}$$

where $0 < \beta_n < 1$ for $1 \le n \le N$ and $Z^{(n)} \sim \mathcal{N}(0,1)$ is an independent Gaussian noise.

Exercise 4. *Show that the variance is preserved in the above relation. What will be the variance of $X^{(N)}$ when $N \to \infty$?*

Exercise 5. *For fixed $\beta_n \equiv \beta$, show that $X^{(N)}$ converges to Gaussian distribution as $N \to \infty$.*

Proof. Consider the moment generating function

$$\begin{aligned}
M_{X^{(n)}}(t) =& \mathbb{E}\left[\exp(tX^{(n)})\right] && (18)\\
=& \mathbb{E}\left[\exp(t\sqrt{1-\beta_n} X^{(n-1)})\right] \mathbb{E}\left[\exp(t\sqrt{\beta_n} Z^{(n)})\right] && (19)\\
=& \mathbb{E}\left[\exp(t\sqrt{1-\beta_n} X^{(n-1)})\right] \exp\left(\frac{\sigma^2\beta_n}{2} t^2\right) && (20)\\
=& \cdots && (21)\\
=& \mathbb{E}\left[\exp(t\sqrt{\bar{\alpha}_n} X^{(0)})\right] \exp\left(\frac{(1-\bar{\alpha}_n)\sigma^2}{2} t^2\right) && (22)
\end{aligned}$$

where $\bar{\alpha}_n = \prod_{i=1}^n (1-\beta_i)$. For fixed $\beta_i \equiv \beta$, it is clear that $\bar{\alpha}_n \to 0$ and the moment generating function converges to that of standard Gaussian. Also, even when β_n is nonuniform, as long as $\bar{\alpha}_n \to 0$, the random variable $X^{(N)}$ converges to standard Gaussian in distribution. □

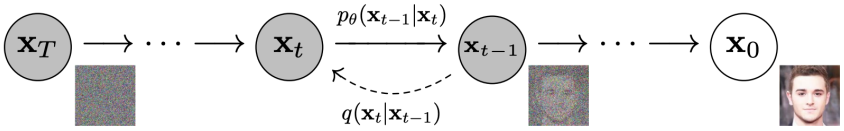

Figure 2: The directed graphical model considered in DDPM [5].

Remark 6. *Note that the forward process does not have to be computed sequentially. We can corrupt a sample $X^{(0)}$ with n-times in a single step*

$$X^{(n)} = \sqrt{\bar{\alpha}_n} X^{(0)} + \sqrt{1-\bar{\alpha}_n} Z \tag{23}$$

where $Z \sim \mathcal{N}(0,1)$. Again, this is useful in training f_θ.

This is called Denoising Diffusion Probabilistic Models (DDPM) [5] (note that it has $>$ 20000 citations). In the original paper, the authors claimed that the true reverse distribution can be approximated (for small β_n)

$$X^{(n-1)}|X^{(n)} \sim \mathcal{N}(\mu(X^{(n)}, n), \beta_n) \tag{24}$$

where

$$\mu(X^{(n)}, n) = \frac{1}{\sqrt{1-\beta_n}} \left(X^{(n)} + \beta_n \frac{\partial}{\partial x} \log P_{X^{(n)}}(X^{(n)}) \right). \tag{25}$$

Note that the $\mu(X^{(n)}, n)$ is from the Tweedie's formula, and we can similarly show that the variance of $X^{(n-1)}|X^{(n)}$ is also β_n with a negligible high order term. Similarly, since we do not know the true score function, it estimates

$$\mu_\theta(X^{(n)}, n) = \frac{1}{\sqrt{1-\beta_n}} \left(X^{(n)} + \beta_n s_\theta(X^{(n)}, n) \right). \tag{26}$$

The above model is called Denoising Diffusion Probabilistic Models (DDPM) [5]. DDPM is discretized version of variance preserving diffusion model (we will see the continuous version of it soon). Recall that the forward and reverse processes are

$$X^{(n)}|X^{(n-1)} \sim \mathcal{N}(\sqrt{1-\beta_n} X^{(n-1)}, \beta_n) \tag{27}$$

$$X^{(n-1)}|X^{(n)} \approx \mathcal{N}(\mu(X^{(n)}, n), \beta_n) \tag{28}$$

where

$$\mu(X^{(n)}, n) = \frac{1}{\sqrt{1-\beta_n}} (X_n + \beta_n \frac{\partial}{\partial X^{(n)}} \log P_{X^{(n)}}(X^{(n)})). \tag{29}$$

The DDPM learns

$$\mu_\theta(X^{(n)}, n) = \frac{1}{\sqrt{1-\beta_n}}(X_n + \beta_n s_\theta(X^{(n)}, n)). \tag{30}$$

The DDPM loss can also be achieved via variational lower bound. Let $X^{(i:j)} = (X^{(i)}, \dots, X^{(j)})$, and q be the distribution of the forward process while p_θ be the distribution of the learned reverse process. Then, the forward distribution is

$$X^{(0)} \sim q(X^{(0)}) \tag{31}$$

$$q(X^{(1:N)}|X^{(0)}) = \prod_{n=1}^{N} q(X^{(n)}|X^{(n-1)}) \tag{32}$$

where the reverse distribution is

$$X^{(N)} \sim p(X^{(N)}) = \mathcal{N}(0,1) \tag{33}$$

$$p_\theta(X^{(0:N)}) = p(X^{(N)}) \prod_{n=1}^{N} p_\theta(X^{(n-1)}|X^{(n)}) \tag{34}$$

$$p_\theta(X^{(0)}) = \int p_\theta(X^{(0:N)})\, dX^{(1:N)}. \tag{35}$$

The goal of generative model is minimizing negative log-likelihood $-\log p_\theta(X^{(0)})$. (Why? See Exercise 6.)

$$-\log p_\theta(X^{(0)}) = -\log \int p_\theta(X^{(0:N)})\, dX^{(1:N)} \tag{36}$$

$$= -\log \left[\int \frac{p_\theta(X^{(0:N)})}{q(X^{(1:N)}|X^{(0)})} q(X^{(1:N)}|X^{(0)})\, dX^{(1:N)} \right] \tag{37}$$

$$= -\log \mathbb{E}_q \left[\frac{p_\theta(X^{(0:N)})}{q(X^{(1:N)}|X^{(0)})} \middle| X^{(0)} \right] \tag{38}$$

$$\leq \mathbb{E}_q \left[-\log \frac{p_\theta(X^{(0:N)})}{q(X^{(1:N)}|X^{(0)})} \middle| X^{(0)} \right]. \tag{39}$$

The final inequality follows from Jensen's inequality. The idea is that instead of minimizing neagtive log-likelihood directly, we minimize the variational lower bound (which is indeed an upperbound in this case):

Exercise 6 (Maximum Likelihood)**.** *Let p_θ be Berounoulli distribution with unknown parameter θ. Suppose we have samples $C_1, \dots, C_n \sim p_\theta$. What is the maximum likelihood estimate of θ?*

Then, the sample mean of likelihood is an approximation of $\mathbb{E}_q[-\log p_\theta(X^{(0)})]$, which is upperbounded by

$$\mathbb{E}_q\left[-\log p_\theta(X^{(0)})\right] \tag{40}$$

$$\leq \mathbb{E}_q\left[-\log \frac{p_\theta(X^{(0:N)})}{q(X^{(1:N)}|X^{(0)})}\right] \tag{41}$$

$$= \mathbb{E}_q\left[-\log p(X^{(N)}) - \sum_{n=1}^{N} \log \frac{p_\theta(X^{(n-1)}|X^{(n)})}{q(X^{(n)}|X^{(n-1)})}\right] \tag{42}$$

$$= \mathbb{E}_q\left[-\log p(X^{(N)}) - \sum_{n=2}^{N} \log \frac{p_\theta(X^{(n-1)}|X^{(n)})}{q(X^{(n)}|X^{(n-1)})} - \log \frac{p_\theta(X^{(0)}|X^{(1)})}{q(X^{(1)}|X^{(0)})}\right] \tag{43}$$

Note that the expectation is with respect to $X^{(0:T)} \sim q$. From Markov property, we have $q(X^{(n)}|X^{(n-1)})q(X^{(n-1)}|X^{(0)}) = q(X^{(n-1)}|X^{(n)}, X^{(0)})q(X^{(n)}|X^{(0)})$.

Exercise 7. *Show that* $q(X^{(n)}|X^{(n-1)})q(X^{(n-1)}|X^{(0)}) = q(X^{(n-1)}|X^{(n)}, X^{(0)})q(X^{(n)}|X^{(0)})$.

Thus,

$$\mathbb{E}_q\left[-\log p_\theta(X^{(0)})\right] \tag{44}$$

$$= \mathbb{E}_q\left[-\log p(X^{(N)}) - \sum_{n=2}^{N} \log \frac{p_\theta(X^{(n-1)}|X^{(n)})}{q(X^{(n-1)}|X^{(n)}, X^{(0)})} \frac{q(X^{(n-1)}|X^{(0)})}{q(X^{(n)}|X^{(0)})} - \log \frac{p_\theta(X^{(0)}|X^{(1)})}{q(X^{(1)}|X^{(0)})}\right] \tag{45}$$

$$= \mathbb{E}_q\left[-\log \frac{p(X^{(N)})}{q(X^{(N)}|X^{(0)})} - \sum_{n=2}^{N} \log \frac{p_\theta(X^{(n-1)}|X^{(n)})}{q(X^{(n-1)}|X^{(n)}, X^{(0)})} - \log p_\theta(X^{(0)}|X^{(1)})\right] \tag{46}$$

The first term of (46) becomes

$$L_N \triangleq \mathbb{E}_q\left[-\log \frac{p_\theta(X^{(N)})}{q(X^{(N)}|X^{(0)})}\right] = \mathbb{E}_{X^{(0)}\sim q}\left[D_{\mathrm{KL}}(q(X^{(N)}|X^{(0)})\|p(X^{(N)}))\right]. \tag{47}$$

where the expectation is with respect to $X^{(0)}$.

The second term of (46) becomes

$$\mathbb{E}_q\left[-\sum_{n=2}^{N} \log \frac{p_\theta(X^{(n-1)}|X^{(n)})}{q(X^{(n-1)}|X^{(n)}, X^{(0)})}\right] \tag{48}$$

$$= \sum_{n=2}^{N} \mathbb{E}_q \left[D_{\mathrm{KL}}(q(X^{(n-1)}|X^{(n)}, X^{(0)}) \| p_\theta(X^{(n-1)}|X^{(n)}) \right] \tag{49}$$

$$= \sum_{n=2}^{N} \mathbb{E}_q \left[D_{\mathrm{KL}}(q(X^{(n-1)}|X^{(n)}, X^{(0)}) \| p_\theta(X^{(n-1)}|X^{(n)}) \right] \tag{50}$$

$$\stackrel{\Delta}{=} \sum_{n=2}^{N} L_{n-1} \tag{51}$$

where the expectation is on conditional $X^{(n)}$ and $X^{(0)}$ for each expectation.

Finally, the last term of (46) is

$$L_0 = \mathbb{E}_q \left[-\log p_\theta(X^{(0)}|X^{(1)}) \right]. \tag{52}$$

Since L_N is independent of θ and L_0 is negligible (compared to $L_1, \ldots, L_{N-1}$), we often consider the second term only:

$$L = \sum_{n=2}^{N} L_{n-1} \tag{53}$$

Exercise 8. *Show that*

$$q(X^{(n-1)}|X^{(n)}, X^{(0)}) = \mathcal{N}(\mu_n(X^{(n)}|X^{(0)}), \beta_n) \tag{54}$$

where

$$\mu_n(X^{(n)}|X^{(0)}) = \frac{1}{\sqrt{1-\beta_n}} \left(X^{(n)} + \beta_n \frac{\partial}{\partial_{X^{(n)}}} \log q(X^{(n)}|X^{(0)}) \right) \tag{55}$$

Hint: Given $X^{(0)}$, all distributions are Gaussian, and you may start with

$$q(X^{(n-1)}|X^{(n)}, X^{(0)}) = \frac{q(X^{(n)}|X^{(n-1)}, X^{(0)}) q(X^{(n-1)}|X^{(0)})}{q(X^{(n)}|X^{(0)})}. \tag{56}$$

Also note that we have an analytic formula for

$$\frac{\partial}{\partial X^{(n)}} \log q(X^{(n)}|X^{(0)}) \tag{57}$$

The following exercise provides an explicit formula for KL divergence between Gaussian distributions

Exercise 9. *Show that*

$$D_{\mathrm{KL}}\left(\mathcal{N}(\mu_0, \sigma_0^2 I) \| \mathcal{N}(\mu_1, \sigma_1^2 I)\right) = \frac{1}{2\sigma_1^2}\|\mu_1 - \mu_0\|^2 + \frac{d}{2}\left(\frac{\sigma_0^2}{\sigma_1^2} - 1\right) + d\log\frac{\sigma_1}{\sigma_0}. \tag{58}$$

where d is dimension of multidimensional Gaussian distribution and $\mu_0, \mu_1 \in \mathbb{R}^d, \sigma_1, \sigma_2 > 0$.

Thus, we have

$$\begin{aligned}
L_{n-1} =& \frac{1}{2\beta_n}\|\mu_n(X^{(n)}|X^{(0)}) - \mu_\theta(X^{(n)}, n)\|^2 && (59)\\
=& \frac{\beta_n}{2(1-\beta_n)}\left\|\frac{\partial}{\partial X^{(n)}}\log q_{n|0}(X^{(n)}|X^{(0)}) - s_\theta(X^{(n)}, n)\right\|^2 && (60)\\
=& \frac{\beta_n}{2(1-\beta_n)}\left\|\frac{X^{(n)} - \sqrt{\bar\alpha_n}X^{(0)}}{1-\bar\alpha_n} - \frac{\epsilon_\theta(X^{(n)}, n)}{\sqrt{1-\bar\alpha_n}}\right\|^2 && (61)\\
=& \frac{\beta_n}{2(1-\beta_n)(1-\bar\alpha_n)}\left\|\epsilon_n - \epsilon_\theta(\sqrt{\bar\alpha_n}X^{(0)} + \sqrt{1-\bar\alpha_n}\epsilon_n, n)\right\|^2 && (62)
\end{aligned}$$

where $X^{(n)} \overset{(d)}{=} \sqrt{\bar\alpha_n}X^{(0)} + \sqrt{1-\bar\alpha_n}\epsilon_n$ with $\epsilon_n \sim \mathcal{N}(0,1)$.

Thus, the training procedure is simply iteratively appying

1. Pick $X^{(0)}$ from data
2. Sample $n \sim \mathrm{Unif}(\{1, \ldots, N\})$
3. Sample $\epsilon \sim \mathcal{N}(0,1)$
4. Compute $X^{(n)} = \sqrt{\bar\alpha_n}X^{(0)} + \sqrt{1-\bar\alpha_n}\epsilon$.
5. Call optimizer to minimize (apply SGD) $\tilde\lambda_n\|\epsilon - \epsilon_\theta(X^{(n)}, n)\|^2$

Note that for small β_n's and large N (so that $\bar\alpha_N$ is small), the terminal variable $X^{(N)}$ is approximately $\mathcal{N}(0,1)$. Thus, with learned $\epsilon_\theta(X^{(n)}, n)$, the sampling process of DDPM is

1. Sample $X^{(N)} \sim \mathcal{N}(0,1)$
2. For $n = N, N-1, \ldots, 1$
   (a) Sample $Z^{(n)} \sim \mathcal{N}(0,1)$
   (b) Compute $X^{(n-1)} = \frac{1}{\sqrt{1-\beta_n}}\left(X^{(n)} - \frac{\beta_n}{\sqrt{1-\bar\alpha_n}}\epsilon_\theta(X^{(n)}, n)\right) + \sqrt{\beta_n}Z^{(n)}$

Remark 7. *There are couple of sources of error: 1) reverse process is only approximately Gaussian, 2) learned score function is not exact, and 3) the terminal distribution is not exactly Gaussian.*

2.3 Stochastic Differential Equation

Both variance exploding (VE) and variance preserving (VP) can be expressed via stochastic differential equation (SDE). Consider the variance exploding (VE) case first where the variance of noise $\sigma^2 \to 0$ is nearly zero. In this limit, we use $X_t, X_{t-\Delta t}, \sigma^2 Z_t$ instead of $X^{(n+1)}, X^{(n)}, Z^{(n)}$, then

$$X_t = X_{t-\Delta t} + \beta Z^{(n)} \tag{63}$$

where $Z^{(n)} \sim \mathcal{N}(0, \Delta t)$, and we replace it bystandard Brownian motion dZ_t. Brownian motion is cumulative Gaussian noise of infinitesimal variances where $Z_t - Z_{t-\Delta t} \sim \mathcal{N}(0, \Delta t)$. This is equivalently

$$dX_t = \sqrt{\beta} dZ_t. \tag{64}$$

Similarly, the variance preserving (VP) case

$$\begin{aligned} X_t =& \sqrt{1-\beta_n} X_{t-\Delta t} + \sqrt{\beta_n} Z_n & (65) \\ \approx& \left(1 - \frac{\beta_t dt}{2}\right) X_{t-\Delta t} + \sqrt{\beta_t} dZ_t & (66) \end{aligned}$$

where $\beta_n = \beta_t dt$. Note that $Z_n \sim \mathcal{N}(0,1)$ and the Brownian motion (roughly speaking) satisfies $dZ_t \sim \mathcal{N}(0, dt)$. Equivalently, we have

$$dX_t = -\frac{\beta_t}{2} X_t dt + \sqrt{\beta_t} dZ_t. \tag{67}$$

More generally, we have the following SDE formula

$$dX_t = f(X_t, t)dt + g(t)dZ_t \tag{68}$$

whose discretized version is

$$X_{k+1} = X_k + \Delta t f(X_k, k\Delta t) + g(k\Delta t)\sqrt{\Delta t} Z_k. \tag{69}$$

We can solve SDE via

$$X_t = X_0 + \int_0^t f(X_s, s)\, ds + \int_0^t g(s)\, dW_s \tag{70}$$

where

$$\int_0^t g(s)\, dW_s = \lim_{\epsilon \to 0} \sum_{k=0}^{\lfloor t/\epsilon \rfloor} g(\epsilon k)\sqrt{\epsilon} Z_k \tag{71}$$

is called Itô stochastic integral.

However, unlike regular ODE where we can analytically express the path X_t, SDE produces a random process. Thus, the solution of SDE should be a probability distribution, more precisely, a joint distribution of $\{X_t\}_{t=0}^T$. For diffusion probabilistic models, we will consider a weaker notion: the marginal probability distributions $\{p_t\}_{t=0}^T$ such that $X_t \sim p_t$ for $0 \le t \le T$. We are interested in the evolution of p_t from $t = 0$ to $t = T$.

This evolution can be expressed via Fokker-Plank equation.

$$\partial_t p_t = -\partial_x(f p_t) + \frac{g^2}{2}\partial_x^2(p_t) \tag{72}$$

which implies

$$\partial_t p_t(x) = -\partial_x(f(x,t)p_t(x)) + \frac{g^2(t)}{2}\partial_x^2(p_t(x)). \tag{73}$$

Rough derivation is following. For any ϕ (smoothe and compact),

$$\begin{aligned}
\partial_t \mathbb{E}_{X_t \sim p_t}[\phi(X)] \approx & \frac{1}{\epsilon}\left(\mathbb{E}_{X\sim p_{t+\epsilon}}[\phi(X)] - \mathbb{E}_{X\sim p_t}[\phi(X)]\right) && (74)\\
\approx & \frac{1}{\epsilon}\left(\mathbb{E}_{X\sim p_t}[\phi(X + \epsilon f + \sqrt{\epsilon} g Z] - \mathbb{E}_{X\sim p_t}[\phi(X)]\right) && (75)\\
\approx & \frac{1}{\epsilon}\Big(\mathbb{E}_{X\sim p_t}[\epsilon\phi'(X)f(X,t) + \sqrt{\epsilon}\phi'(X)g(t)Z \\
& + \frac{1}{2}\phi''(X)g^2(t)\epsilon Z^2 + O(\epsilon^{3/2})]\Big) && (76)\\
\approx & \mathbb{E}_{X\sim p_t}[\phi'(X)f(X,t) + \frac{1}{2}\phi''(X)g^2(t)] && (77)
\end{aligned}$$

where $Z \sim \mathcal{N}(0,1)$.

From integration by part,

$$\partial_t \int \phi(x) p_t(x)\, dx = \int \phi'(x) f(x,t) p_t(x)\, dx + \frac{1}{2}\int \phi''(x) g^2(t) p_t(x)\, dx \tag{78}$$

$$\leftrightarrow \int \phi(x)\partial_t p_t(x)\, dx = \int \phi(x)(-\partial_x(f p_t))\, dx + \frac{1}{2}\int \phi(x) g^2(t)\partial_x^2(p_t)\, dx \tag{79}$$

where the boundary condition is $p(\pm\infty) = 0$ and $\nabla p(\pm\infty) = 0$. Finally, we have

$$\partial_t p_t = -\partial_x(f p_t) + \frac{g^2}{2}\partial_x^2(p_t). \tag{80}$$

Now, we will show the forward SDE and reverse SDE has the same marginal distribution evolution.

Theorem 1 (Anderson's reverse SDE time theorem [1])**.** *Given the forward SDE*

$$dX_t = f(X_t, t)dt + g(t)dZ_t \tag{81}$$

where $X_0 \sim p_0$, the corresponding reverse-time SDE is

$$d\bar{X}_t = (f(\bar{X}_t, t) - g^2(t)\nabla_x \log p_t(\bar{X}_t))dt + g(t)d\bar{Z}_t. \tag{82}$$

with $\bar{X}_T \sim p_T$ where $d\bar{Z}_t$ is the reverse time Brownian motion and p_T is a distribution of X_T from forward SDE.

Alternatively, define $\{Y_t\}_{t=0}^T$ where $Y_t = \bar{X}_{T-t}$ via

$$dY_t = -(f(Y_t, T-t) - g^2(T-t)\nabla_x p_{T-t}(Y_t))dt + g(T-t)dZ_t \tag{83}$$

with $Y_0 \sim p_T$. Then,

$$X_t \overset{D}{=} \bar{X}_t = Y_{T-t}. \tag{84}$$

Note that $dZ_t \overset{D}{=} -dZ_t$.

Remark 8. *The above theorem holds for general multidimensional cases. For single dimensional distributions, we have*

$$d\bar{X}_t = (f(\bar{X}_t, t) - g^2(t)\partial_x \log p_t(\bar{X}_t))dt + g(t)d\bar{Z}_t. \tag{85}$$

The proof is simple application of FP equation.

Proof. Let $\{q_t\}_{t=0}^T$ be marginal densities of $\{Y_t\}_{t=0}^T$. Then, $\{q_t\}_{t=0}^T$ satisfies the FP equation

$$\partial_t q_t(y) = \partial_y\left((f(y, T-t) - g^2(T-t)\partial_y \log p_{T-t}(y))q_t(y)\right) + \frac{g^2(T-t)}{2}\partial_y^2(q_t(y)) \tag{86}$$

Let $\{\bar{p}_t\}_{t=0}^T$ be marginal densities of $\{\bar{X}_t\}_{t=0}^T$. Since $\bar{p}_{T-t} = q_t$, then $\{\bar{p}_t\}_{t=0}^T$ satisfies the FP equation

$$\partial_t \bar{p}_t(x) = -\partial_x \left((f(x,t) - g^2(t)\partial_x \log p_t(x))\bar{p}_t(x) \right) - \frac{g^2(t)}{2}\partial_x^2(\bar{p}_t(x)). \quad (87)$$

The final step of the proof is proving that p_t solves the above reverse FP equation.

$$\begin{aligned}
&- \partial_x \left((f(x,t) - g^2(t)\partial_x \log p_t(x))p_t(x) \right) - \frac{g^2(t)}{2}\partial_x^2(p_t(x)) && (88)\\
=&- \partial_x(fp_t) + g^2(t)\partial_x(\partial_x \log p_t(x) p_t(x)) - \frac{g^2(t)}{2}\partial_x^2(p_t(x)) && (89)\\
=&- \partial_x(fp_t) + g^2(t)\partial_x(\partial_x p_t(x)) - \frac{g^2(t)}{2}\partial_x^2(p_t(x)) && (90)\\
=&- \partial_x(fp_t) + \frac{g^2(t)}{2}\partial_x^2(p_t(x)) && (91)\\
=&\partial_x p_t && (92)
\end{aligned}$$

where the last equation is from forward FP equation. Finally, p_t also satisfies (87), and therefore $p_t = \bar{p}_t$. □

2.4 Score Matching

We can generate sample using reverse SDE, starting from Gaussian sample X_T. However, we need $\partial_x \log p_t(x)$. At first glance, it sounds weird since the whole point of generative model is from hardness of learning distribution $p_X(x)$ directly. However, learning the score is (surprisingly) more tractable. More precisely, we learn the score function via neural network $s_\theta(x,t)$. The natural loss function would be

$$\mathcal{L}(\theta) = \int_0^T \lambda(t)\mathbb{E}_{X_t}\left[\|s_\theta(X_t,t) - \partial_{X_t}\log p_t(X_t)\|^2\right] dt \quad (93)$$

where $\lambda(t) > 0$ is a weighing factor. This is because we want to learn score function for all $0 \le t \le T$. However, we do not have an access to p_t, and we need alternative formula.

Theorem 2 (Score Matching [12])**.** *We have equivalent formula*

$$\mathcal{L}(\theta) = \int_0^T \lambda(t)\mathbb{E}_{X_0}\left[\mathbb{E}_{X_t|X_0}\left[\|s_\theta(X_t,t) - \partial_{X_t}\log p_{t|0}(X_t|X_0)\|^2\right]\right] dt + C \quad (94)$$

for some constant C which are independent of θ.

We do not know p_t, but we do know $p_{t|0}(X_t|X_0)$ since it is noise corruption process (adding Gaussian noise). Thus, an alternative formula (94) is tractable.

Proof.

$$\begin{aligned}
\partial_{X_t} \log p_t(X_t) =& \frac{\partial_{X_t} p_t(X_t)}{p_t(X_t)} && (95)\\
=& \left(\partial_{X_t} \int p_{t|0}(X_t|X_0) p_0(X_0)\, dX_0 \right) \cdot \frac{1}{p_t(X_t)} && (96)\\
=& \left(\int \partial_{X_t} p_{t|0}(X_t|X_0) p_0(X_0)\, dX_0 \right) \cdot \frac{1}{p_t(X_t)} && (97)\\
=& \left(\int \left(\partial_{X_t} \log p_{t|0}(X_t|X_0)\right) p_0(X_0) p_{t|0}(X_t|X_0)\, dX_0 \right) \cdot \frac{1}{p_t(X_t)} && (98)\\
=& \int \left(\partial_{X_t} \log p_{t|0}(X_t|X_0)\right) \frac{p_0(X_0) p_{t|0}(X_t|X_0)}{p_t(X_t)}\, dX_0 && (99)\\
=& \int \left(\partial_{X_t} \log p_{t|0}(X_t|X_0)\right) p_{0|t}(X_0|X_t)\, dX_0 && (100)\\
=& \mathbb{E}_{X_0|X_t} \left[\partial_{X_t} \log p_{t|0}(X_t|X_0) \;\middle|\; X_t \right]. && (101)
\end{aligned}$$

Then, the loss function would be

$$\begin{aligned}
\mathcal{L}(\theta) =& \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\| s_\theta(X_t, t) - \partial_{X_t} \log p_t(X_t) \|^2 \right] dt && (102)\\
=& \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\| s_\theta(X_t, t) \|^2 - 2 \langle s_\theta(X_t, t), \partial_{X_t} \log p_t(X_t) \rangle \right] dt + C && (103)\\
=& \int_0^T \lambda(t) \mathbb{E}_{X_t} \Big[\| s_\theta(X_t, t) \|^2 \\
& -2 \langle s_\theta(X_t, t), \mathbb{E}_{X_0|X_t} \left[\partial_{X_t} \log p_{t|0}(X_t|X_0) \;\middle|\; X_t \right] \rangle \Big] dt + C && (104)\\
=& \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\mathbb{E}_{X_0|X_t} \left[\| s_\theta(X_t, t) \|^2 - 2 \langle s_\theta(X_t, t), \partial_{X_t} \log p_{t|0}(X_t|X_0) \rangle \;\middle|\; X_t \right] \right] dt + C && (105)\\
=& \int_0^T \lambda(t) \mathbb{E}_{X_t} \left[\mathbb{E}_{X_0|X_t} \left[\| s_\theta(X_t, t) - \partial_{X_t} \log p_{t|0}(X_t|X_0) \|^2 \;\middle|\; X_t \right] \right] dt + C' && (106)
\end{aligned}$$

$$= \int_0^T \lambda(t) \mathbb{E}_{X_0} \left[\mathbb{E}_{X_t|X_0} \left[\| s_\theta(X_t, t) - \partial_{X_t} \log p_{t|0}(X_t|X_0) \|^2 \right] \right] dt + C' \tag{107}$$

□

In (regular) score matching, the conditional score function $\partial X_t \log p_{t|0}(X_t|X_0)$ is implementable for reasonable f and g. For example, the Orstein-Uhlenbeck process

$$dX_t = -\beta X_t dt + \sigma dW_t \tag{108}$$

is one such example where

$$p_{t|0}(X_t|X_0) \sim \mathcal{N}\left(e^{-\beta t} X_0, \frac{\sigma^2}{2\beta}(1 - e^{-2\beta t}) \right). \tag{109}$$

In both cases, we have

$$X_t \overset{(d)}{=} \gamma_t X_0 - \sigma_t \epsilon \tag{110}$$

for some γ_t, σ_t and $\epsilon \sim \mathcal{N}(0, I)$. Then, the score function is given by

$$\nabla \log p_{t|0}(X_t|X_0) = \frac{\gamma_t X_0 - X_t}{\sigma_t^2} \tag{111}$$

$$\overset{(d)}{=} \frac{\epsilon}{\sigma_t}. \tag{112}$$

With reparameterization trick, we can define

$$s_\theta(X_t, t) = \frac{\epsilon_\theta(X_t, t)}{\sigma_t}. \tag{113}$$

Finally, the loss function (of regular score matching) becomes

$$\tilde{\mathcal{L}}(\theta) = \int_0^T \lambda(t) \mathbb{E}_{X_0} \left[\mathbb{E}_{X_t|X_0} \left[\| s_\theta(X_t, t) - \partial_{X_t} \log p_{t|0}(X_t|X_0) \|^2 \mid X_0 \right] \right] dt. \tag{114}$$

$$= \int_0^T \frac{\lambda(t)}{\sigma_t^2} \mathbb{E}_{X_0} \left[\mathbb{E}_\epsilon \left[\| \epsilon_\theta(\gamma_t X_0 - \sigma_t \epsilon, t) - \epsilon \|^2 \mid X_0 \right] \right] dt. \tag{115}$$

$$= T \cdot \mathbb{E}_{X_0, \epsilon, t \sim \text{Unif}(0,T)} \left[\frac{\lambda(t)}{\sigma_t^2} \| \epsilon_\theta(\gamma_t X_0 - \sigma_t \epsilon, t) - \epsilon \|^2 \right]. \tag{116}$$

Effectively, the reparameterized network $\epsilon_\theta(X_t, t)$ predicts the noise ϵ from noisy data $X_t \overset{(d)}{=} \gamma_t X_0 - \sigma\epsilon$ where the score function s_θ predicts the scaled noise $\sigma_t\epsilon$. In practice, it is more common to implement ϵ_θ instead of s_θ (although they are equivalent up to constant scale factor).

Remark 9. *Note that at $t = 0$, the SDE has $\sigma_t = 0$ and the loss blows up. In practice, a common choice is to train loss from $t = t_0 > 0$, i.e.,*

$$\tilde{\mathcal{L}}(\theta) = \mathbb{E}_{X_0,\,\epsilon,\,t\sim \mathit{Unif}(t_0,T)}\left[\frac{\lambda(t)}{\sigma_t^2}\|\epsilon_\theta(\gamma_t X_0 - \sigma_t\epsilon, t) - \epsilon\|^2\right]. \tag{117}$$

We can compute the loss function for DDPM [5] again. Recall the reparameterization trick

$$\bar{\alpha}_n = \prod_{i=1}^{n}(1-\beta_n) \tag{118}$$

$$X^{(n)} \overset{(d)}{=} \sqrt{\bar{\alpha}_n}X_0 - \sqrt{1-\bar{\alpha}_n}\epsilon_n \tag{119}$$

$$\epsilon_n \sim \mathcal{N}(0,1) \tag{120}$$

$$\epsilon_\theta = \sqrt{1-\bar{\alpha}_n}s_\theta \tag{121}$$

$$\tilde{\lambda}_n = \frac{\lambda_n\beta_n^2}{1-\bar{\alpha}_n} \tag{122}$$

Recall that the loss function from VLB is

$$\mathcal{L}(\theta) = \sum_{n=1}^{N}\tilde{\lambda}_n\mathbb{E}_{X^{(0)},\epsilon\sim\mathcal{N}(0,1)}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_n}X^{(0)} - \sqrt{1-\bar{\alpha}_n}\epsilon, n)\right\|^2\right] \tag{123}$$

This coincides with the loss derived from score matching.

3 (Multi-dimensional) Diffusion Model

Now, lets do the same for the multi-dimensional case. Let $\mathbf{X} = (X_1, \ldots, X_d) \in \mathbb{R}^d$ be a random vector that we want to learn. The density function is $P_{\mathbf{X}}(\mathbf{x})$. The variance exploding forward process is

$$\mathbf{X}^{(n)} = \begin{pmatrix} X_1^{(n)} \\ \vdots \\ bx_d^{(n)} \end{pmatrix} = \begin{pmatrix} X_1^{(n-1)} \\ \vdots \\ bx_d^{(n-1)} \end{pmatrix} + \begin{pmatrix} Z_1^{(n)} \\ \vdots \\ Z_d^{(n)} \end{pmatrix} \tag{124}$$

where $\mathbf{X}^{(0)} = \mathbf{X}$ and $Z_1^{(n)}, Z_2^{(n)}, \ldots, Z_d^{(n)} \sim \mathcal{N}(0, \sigma^2)$ are independent Gaussian random variables with mean zero and variable σ^2. As forward process progresses, the Gaussian noise dominates (again) and the terminal distribution will be approximately i.i.d. Gaussian. Even for multidimensional Gaussian, estimating mean vector and covariance matrix is relatively easy.

Similar to 1D case, we can approximate the reverse process by

$$\left(\tilde{\mathbf{X}}^{(n-1)} | \tilde{\mathbf{X}}^{(n)} = \tilde{\mathbf{x}}^{(n)}\right) \sim \mathcal{N}\left(\tilde{\mathbf{x}}^{(n)} + \sigma^2 \nabla \log P_{\tilde{\mathbf{X}}^{(n)}}(\tilde{\mathbf{x}}^{(n)}), \sigma^2 I\right). \tag{125}$$

The key step is (again) approximation of $P_{\mathbf{X}^{(n-1)}}(\mathbf{x})$ by

$$P_{\tilde{\mathbf{X}}^{(n-1)}}(\tilde{\mathbf{x}}) = P_{\tilde{\mathbf{X}}^{(n-1)}}(\tilde{\mathbf{x}}^{(n)}) + \langle \nabla P_{\tilde{\mathbf{X}}^{(n-1)}}(\tilde{\mathbf{x}}^{(n)}), \tilde{\mathbf{x}} - \tilde{\mathbf{x}}^{(n)} \rangle + O(\|\tilde{\mathbf{x}} - \tilde{\mathbf{x}}^{(n)}\|^2). \tag{126}$$

The reverse process to obtain $\tilde{\mathbf{X}}^{(N-1)}$ to $\tilde{\mathbf{X}}^{(0)}$ and hoping $\tilde{\mathbf{X}}^{(0)} \overset{(d)}{\approx} X$ (similar in distribution).

Recall that the result matches with Anderson's reverse SDE.

$$d\bar{\mathbf{X}}_t = (f(\bar{\mathbf{X}}_t, t) - g^2(t) \nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{X}}_t))dt + g(t) d\bar{\mathbf{Z}}_t. \tag{127}$$

Note that the Tweedie's formula also holds for multi-dimensional case.

Lemma 1 (Tweedie's formula, multidimension case)**.** *Let* $\mathbf{X}$ *be a random vector and* $\mathbf{Z}$ *be i.i.d. Gaussian with zero mean and variance* σ^2*. For* $\mathbf{Y} = \mathbf{X} + \mathbf{Z}$*, we have*

$$\mathbb{E}[\mathbf{X}|\mathbf{Y}] = \mathbf{Y} + \sigma^2 \nabla_{\mathbf{y}} \log P_{\mathbf{Y}}(\mathbf{y}). \tag{128}$$

Thus, if diffusion model is trained to learn "denoising," it essentially learns

$$f_\theta(\mathbf{x}^{(n)}, n) \approx \mathbb{E}\left[\mathbf{X}^{(0)} | \mathbf{X}^{(n)} = \mathbf{x}^{(n)}\right] \tag{129}$$

$$= \mathbf{x}^{(n)} + n\sigma^2 \nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}}(\mathbf{x}^{(n)}). \tag{130}$$

Thus, we can approximate score function by

$$s_\theta(\mathbf{x}^{(n)}, n) = \frac{f_\theta(\mathbf{x}^{(n)}, n) - \mathbf{x}^{(n)}}{n\sigma^2}. \tag{131}$$

3.1 (Multidimensional) DDPM [5]

DDPM formula would be the same except the noise is now i.i.d. Gaussian with covariance $\sigma^2 I$. The forward and reverse processes are

$$\mathbf{X}^{(n)}|\mathbf{X}^{(n-1)} \sim \mathcal{N}(\sqrt{1-\beta_n}\mathbf{X}^{(n-1)}, \beta_n I) \tag{132}$$

$$\mathbf{X}^{(n-1)}|\mathbf{X}^{(n)} \approx \mathcal{N}(\mu(\mathbf{X}^{(n)}, n), \beta_n I) \tag{133}$$

where

$$\mu(\mathbf{X}^{(n)}, n) = \frac{1}{\sqrt{1-\beta_n}}(\mathbf{X}_n + \beta_n \nabla_{\mathbf{X}^{(n)}} \log P_{\mathbf{X}^{(n)}}(\mathbf{X}^{(n)})). \tag{134}$$

Remark 10. *In multidimensional setup, the distribution of $\mathbf{X}^{(N)}$ converges to not only Gaussian in marginal sense, it converges to the i.i.d. Gaussian distribution.*

Exercise 10. *Show that $\mathbf{X}^{(N)}$ converges to $\mathcal{N}(0, I)$ as $N \to \infty$.*

The DDPM learns

$$\mu_\theta(\mathbf{X}^{(n)}, n) = \frac{1}{\sqrt{1-\beta_n}}(\mathbf{X}_n + \beta_n s_\theta(\mathbf{X}^{(n)}, n)). \tag{135}$$

We have the same loss function derived from variational lower bound:

$$L = \sum_{n=2}^{N} L_{n-1} \tag{136}$$

$$L_{n-1} = \frac{\beta_n}{2(1-\beta_n)(1-\bar{\alpha}_n)} \left\| \epsilon_n - \epsilon_\theta(\sqrt{\bar{\alpha}_n}\mathbf{X}^{(0)} + \sqrt{1-\bar{\alpha}_n}\epsilon_n, n) \right\|^2 \tag{137}$$

where $\mathbf{X}^{(n)} \overset{(d)}{=} \sqrt{\bar{\alpha}_n}\mathbf{X}^{(0)} + \sqrt{1-\bar{\alpha}_n}\epsilon_n$ with $\epsilon_n \sim \mathcal{N}(0, I)$.

Lets recap the training procedure:

1. Pick $\mathbf{X}^{(0)}$ from data
2. Sample $n \sim \text{Unif}(\{1, \ldots, N\})$
3. Sample $\epsilon \sim \mathcal{N}(0, I)$
4. Compute $\mathbf{X}^{(n)} = \sqrt{\bar{\alpha}_n}\mathbf{X}^{(0)} + \sqrt{1-\bar{\alpha}_n}\epsilon$.
5. Call optimizer to minimize (apply SGD) $\tilde{\lambda}_n \|\epsilon - \epsilon_\theta(\mathbf{X}^{(n)}, n)\|^2$

With learned $\epsilon_\theta(\mathbf{X}^{(n)}, n)$, the sampling process of DDPM is

1. Sample $X^{(N)} \sim \mathcal{N}(0, I)$
2. For $n = N, N-1, \ldots, 1$
   (a) Sample $\mathbf{Z}^{(n)} \sim \mathcal{N}(0, I)$
   (b) Compute $\mathbf{X}^{(n-1)} = \frac{1}{\sqrt{1-\beta_n}} \left(\mathbf{X}^{(n)} - \frac{\beta_n}{\sqrt{1-\bar{\alpha}_n}} \epsilon_\theta(\mathbf{X}^{(n)}, n) \right) + \sqrt{\beta_n} \mathbf{Z}^{(n)}$

3.2 Stochastic Differential Equation (SDE)

For $f(\mathbf{X}_t, t) \in \mathbb{R}^d, g(t) \in \mathbb{R}^{d \times d}$, we have the following general SDE formula (covers VP and VE)

$$d\mathbf{X}_t = f(\mathbf{X}_t, t)dt + g(t)d\mathbf{Z}_t \tag{138}$$

whose discretized version is

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \Delta t f(\mathbf{X}_k, k\Delta t) + g(k\Delta t)\sqrt{\Delta t}\mathbf{Z}_k. \tag{139}$$

Recall that the solution of SDE should be a probability distribution, where the marginal probability distributions $\{p_t\}_{t=0}^T$ such that $X_t \sim p_t$ for $0 \le t \le T$. The evolution of $\{p_t\}$ can be expressed via Fokker-Plank equation.

$$\partial_t p_t = -\nabla_{\mathbf{x}} \circ (f p_t) + \frac{1}{2} \operatorname{Tr}(g^\intercal H_{p_t} g) \tag{140}$$

where H_{p_t} is the Hessian of $p_t(\mathbf{x})$.

Remark 11. *If $g(t) = \rho(t)I$ for $\rho(t) \in \mathbb{R}$, then FP is simply*

$$\partial_t p_t = -\nabla_{\mathbf{x}} \circ (f p_t) + \frac{\rho(t)^2}{2} \Delta p_t \tag{141}$$

where $\Delta p_t = \nabla \circ \nabla p_t = \sum_{i=1}^d \frac{\partial^2}{\partial_{x_i}^2}$ is Laplacian.

Recall that the 1-D FP equation was.

$$\partial_t p_t = -\partial_x (f p_t) + \frac{g^2}{2} \partial_{\mathbf{x}}^2 (p_t) \tag{142}$$

which implies

$$\partial_t p_t(\mathbf{x}) = -\partial_{\mathbf{x}} (f(\mathbf{x}, t) p_t(\mathbf{x})) + \frac{g^2(t)}{2} \partial_{\mathbf{x}}^2 (p_t(\mathbf{x})). \tag{143}$$

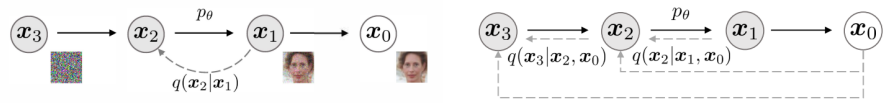

Figure 3: DDPM [5] vs. DDIM [10]

Anderson's theroem still holds and we have a reverse process with matching marginal distributions.

$$d\bar{\mathbf{X}}_t = (f(\bar{\mathbf{X}}_t, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{X}}_t))dt + g(t)d\bar{\mathbf{Z}}_t. \tag{144}$$

with $\bar{\mathbf{X}}_T \sim p_T$ where $d\bar{\mathbf{Z}}_t$ is the reverse time Brownian motion and p_T is a distribution of $\mathbf{X}_T$ from forward SDE. Alternatively, define $\{\mathbf{Y}_t\}_{t=0}^T$ where $\mathbf{Y}_t = \bar{\mathbf{X}}_{T-t}$via

$$d\mathbf{Y}_t = -(f(\mathbf{Y}_t, T-t) - g^2(T-t)\nabla_{\mathbf{x}} p_{T-t}(\mathbf{Y}_t))dt + g(T-t)d\mathbf{Z}_t \tag{145}$$

with $\mathbf{Y}_0 \sim p_T$. Then,

$$\mathbf{X}_t \overset{D}{=} \bar{\mathbf{X}}_t = \mathbf{Y}_{T-t}. \tag{146}$$

3.3 Score Matching

We have the same score matching theorem in multidimensional case

Theorem 3 ((Multidimensional) Score Matching [12])**.** *We have equivalent formula*

$$\mathcal{L}(\theta) = \int_0^T \lambda(t)\mathbb{E}_{\mathbf{X}_0}\left[\mathbb{E}_{\mathbf{X}_t|\mathbf{X}_0}\left[\|s_\theta(\mathbf{X}_t, t) - \nabla_{\mathbf{X}_t} \log p_{t|0}(\mathbf{X}_t|\mathbf{X}_0)\|^2\right]\right] dt + C \tag{147}$$

for some constant C which are independent of θ.

The proof is more or less the same. Similar to a single dimensional case, we can also derive the loss function of DDPM from score matching result.

4 Denoising Diffusion Implicit Model (DDIM) [10]

Note that the DDPM objective only depends on the marginal $q(X^{(n)}|X^{(0)})$, and it does not depend on the joint distribution of entire $X^{(0:n)}$. Thus, we

can modify the generation distribution q while having the same marginal $q(X^{(n)}|X^{(0)}) = \mathcal{N}(\sqrt{\bar{\alpha}_n}X^{(0)}, (1-\bar{\alpha}_n)I)$.

The DDIM has non-Markovian forward process (what is Markovian?). Given the sample $X^{(0)} = X$, the forward process q is

$$q(X^{(N)}|X^{(0)}) = \mathcal{N}(\sqrt{\bar{\alpha}_N}X^{(0)}, (1-\bar{\alpha}_N)I) \tag{148}$$

$$q(X^{(n)}|X^{(n+1)}, X^{(0)}) = \mathcal{N}(\sqrt{\bar{\alpha}_n}X^{(0)} + \frac{\sqrt{1-\bar{\alpha}_n-\sigma_{n+1}^2}}{\sqrt{1-\bar{\alpha}_{n+1}}}(X^{(n+1)} - \sqrt{\bar{\alpha}_n}X^{(0)}), \sigma_{n+1}^2 I) \tag{149}$$

for $\sigma_n \geq 0$.

Exercise 11. *Show that* (149) *matches the marginal distribution of DDPM.*

If we have $\sigma_n \equiv 0$, then the forward process is deterministic.

DDIM trains the error network $\epsilon_\theta(X^{(n)}, n)$ that predicts $\epsilon_n = Z^{(n)}$ where

$$X^{(n)} \triangleq \sqrt{\alpha}_n X^{(0)} + \sqrt{1-\bar{\alpha}_n}Z^{(n)}. \tag{150}$$

Since we can estimate $\hat{X}^{(0)}$ by

$$\hat{X}^{(0)} = \frac{1}{\sqrt{\bar{\alpha}_n}}(X^{(n)} - \sqrt{1-\bar{\alpha}_n}\epsilon_\theta(X^{(n)}, n)), \tag{151}$$

DDIM sampling can be done by

$$p_\theta(X^{(n)}|X^{(n+1)}) = q(X^{(n)}|X^{(n+1)}, \hat{X}^{(0)}). \tag{152}$$

In other words,

$$\begin{aligned} X^{(n)} =& \sqrt{\bar{\alpha}_n}\left(\frac{X^{(n+1)} - \sqrt{1-\bar{\alpha}_{n+1}}\epsilon_\theta(X^{(n+1)}, n+1)}{\bar{\alpha}_{n+1}}\right) \\ &+ \sqrt{1-\bar{\alpha}_n-\sigma_{n+1}^2}\epsilon_\theta(X^{(n+1)}, n+1) + \sigma_n Z^{(n)}. \end{aligned} \tag{153}$$

Then, the sampling process becomes deterministic if $\sigma_n \equiv 0$. This deterministic generation is called discretization of the ODE (ordinary differential equation) sampling.

Note that training a Denoising Diffusion Implicit Model (DDIM) can be done in exactly the same manner as a Denoising Diffusion Probabilistic Model (DDPM), with the difference only in the sampling process.

5 Guidance

Consider the case where the diffusion model is trained on dataset with various labels (e.g., ImageNET [2], CIFAR10 [7]). Then, we may want to generate a sample with specific label y, i.e., sample from $P_{X|Y}(x|y)$. Moreover, in multimodal models such as DALL·E2 that generates an image from prompt, we may also want to generate a specific image instead of random image from P_X. A common technique to generate a specific image, conditioned on label or prompt, is guidance.

5.1 Classifier Guidance

Let Y be the class label of $\mathbf{X}$ and $(\mathbf{X}, Y) \sim P_{\mathbf{X},Y}$. One naive approach is to train score model $s_\theta(\mathbf{X}|Y = y) = \nabla_{\mathbf{x}} \log P_{\mathbf{X}|Y}(\mathbf{x}|y)$ for all y and generate sample according to $P_{\mathbf{X}|Y}(\mathbf{x}|y)$. More precisely, we want to run a reverse process with $\nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}|Y}(\mathbf{x}^{(n)}|y)$ instead of $\nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}}(\mathbf{X}^{(n)})$. However, this requires training score networks multiple times (for all y).

From the Bayes' rule, we have

$$P_{\mathbf{X}|Y}(\mathbf{x}|y) = \frac{P_{\mathbf{X}}(\mathbf{x}) P_{Y|\mathbf{X}}(y|\mathbf{x})}{P_Y(y)} \tag{154}$$

$$\nabla_x \log P_{\mathbf{X}|Y}(\mathbf{x}|y) = \nabla_{\mathbf{x}} \log P_{\mathbf{X}}(\mathbf{x}) + \nabla_{\mathbf{x}} \log P_{Y|\mathbf{X}}(y|\mathbf{x}). \tag{155}$$

Thus,

$$\nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}|Y}(\mathbf{x}^{(n)}|y) = \nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}}(\mathbf{x}^{(n)}) + \nabla_{\mathbf{x}^{(n)}} \log P_{Y|\mathbf{X}^{(n)}}(y|\mathbf{x}^{(n)}). \tag{156}$$

The idea of classifier guidance is re-use the pretrained score model

$$s_\theta(\mathbf{X}^{(n)}, n) = \nabla_{\mathbf{X}^{(n)}} \log P_{\mathbf{X}^{(n)}}(\mathbf{X}^{(n)}). \tag{157}$$

The classifier guidance assumes that we can train a time-dependent classifier which predicts the label Y based on $\mathbf{X}^{(n)}$, i.e., it provides

$$c_\phi(\mathbf{X}^{(n)}, Y, n) \approx P_{\mathbf{X}^{(n)}}(Y|\mathbf{X}^{(n)}). \tag{158}$$

Remark 12. *In original SDE paper [11], the authors claimed that the output of the classifier $Y^{(n)}$ can be approximately equal to true label y for n small. Intuition is that it is enough to approximate $Y^{(n)} \approx y$ for small n since the generating process is roughly*

Figure 4: inpainting and colorization [11]

- *at early stage of generation (large n), it roughly generate the structure (not specific to label)*
- *at the end of generation process (for small n), it guide to a sample corresponds to a specific label y.*

Except the fact that c_ϕ takes an additional input n (indicates the noise level of $\mathbf{X}^{(n)}$) it is a regular classifier. We can train as if it is a regular classifier. The generation can be done via reverse SDE process (or discretized version of it)

$$d\bar{\mathbf{X}}^{(t)} = (f - g^2(\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(\bar{\mathbf{X}}^{(t)}) + \nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(Y|\bar{\mathbf{X}}^{(t)}))dt + gd\bar{\mathbf{Z}}^{(t)} \quad (159)$$

where we replace the conditional distribution by the estimate from classifier

$$d\bar{\mathbf{X}}^{(t)} = (f - g^2(\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(\bar{\mathbf{X}}^{(t)}) + \nabla_{\bar{\mathbf{X}}^{(t)}} c_\phi(\mathbf{X}^{(t)}, Y, t)dt + gd\bar{\mathbf{Z}}^{(t)}. \quad (160)$$

In practice, it helps to scale the guidance term by a constant factor $\omega > 1$ [3].

$$d\bar{\mathbf{X}}^{(t)} = (f - g^2(\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(\bar{\mathbf{X}}^{(t)}) + \omega\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(Y|\bar{\mathbf{X}}^{(t)}))dt + gd\bar{\mathbf{Z}}^{(t)} \quad (161)$$

5.2 Inpainting

Let Ω be a pixel subsampling operation where $Y = \Omega(X^{(0)})$. For remaining part, we use $\bar{\Omega}$. Our goal is to generate the remaining part of image where

$Y = \Omega(X^{(0)})$ is fixed. Let $Z^{(t)} = \bar{\Omega}(X^{(t)})$, then it follows the same SDE

$$dZ^{(t)} = \bar{f}(Z^{(t)}, t)dt + \bar{g}(t)d\bar{W}^{(t)}, \tag{162}$$

where $Z_0 = \bar{\Omega}(X^{(0)})$, $\bar{f} = \bar{\Omega}(f), \bar{g} = \bar{\Omega}(g)$, and $\bar{W}^{(t)} = \bar{\Omega}(W^{(t)})$. Note that the Brownian motion is coordinate-wise independent (we are adding i.i.d. Gaussian noise to each pixel), $\bar{W}^{(t)}$ is simply the lower dimensional Brownian motion. Since we want conditional sampling given Y, we have

$$\begin{aligned}
&p_t(Z^{(t)}|\Omega(X^{(0)}) = Y) && (163)\\
&= \int p_t(Z^{(t)}|\Omega(X^{(t)}), \Omega(X^{(0)}))p_t(\Omega(X^{(t)})|\Omega(X^{(0)}))\, d\Omega(X^{(t)}) && (164)\\
&= \mathbb{E}_{\Omega(X^{(t)})\sim p_t(\Omega(X^{(t)})|\Omega(X^{(0)}))}\left[p_t(Z^{(t)}|\Omega(X^{(t)}), \Omega(X^{(0)}))\right] && (165)\\
&\approx \mathbb{E}_{\Omega(X^{(t)})\sim p_t(\Omega(X^{(t)})|\Omega(X^{(0)}))}\left[p_t(Z^{(t)}|\Omega(X^{(t)}))\right] && (166)\\
&\approx p_t(Z^{(t)}|\hat{\Omega}(X^{(t)})) && (167)
\end{aligned}$$

where $\hat{\Omega}(X^{(t)})$ is noise corrupted sample of $\Omega(X^{(0)})$. The first approximation is from the observation that $\Omega(X^{(t)})$ contains enough information of $Z^{(t)}$ compared to $\Omega(X^{(0)})$.

The second approximation is a single sample approximation of the expected value. Thus, we guide the sampling process using

$$\begin{aligned}
\nabla_{Z^{(t)}} \log p_t(Z^{(t)}|\Omega(X^{(0)}) = Y) &\approx \nabla_{Z^{(t)}} \log p_t(Z^{(t)}|\hat{\Omega}(X^{(t)})) && (168)\\
&= \nabla_{Z^{(t)}} \log p_t([Z^{(t)}; \hat{\Omega}(X^{(t)})]) && (169)
\end{aligned}$$

Note $[Z^{(t)}; \hat{\Omega}(X^{(t)})]$ forms a full dimensional vector where we can compute the score function. Thus, for $Y = \Omega(X^{(0)})$, the (discretized) inpainting can be done by

1. Sample $X^{(N)}$
2. For $n = N, N-1, \cdots, 1$
   (a) Sample $\epsilon_n \sim \mathcal{N}(0, I)$
   (b) $\zeta_n = \beta_n Y + \gamma_n \epsilon_n$
   (c) Sample $\epsilon_n' \sim \mathcal{N}(0, I)$
   (d) Sample via

$$\bar{Z}^{(n-1)} = \bar{Z}^{(n)} - \Delta t(-\beta \bar{Z}^{(n)} - \sigma^2 \bar{\Omega}(s_\theta([\bar{Z}^{(n)}; \zeta_n], n\Delta t))) + \sigma\sqrt{\Delta n}\epsilon_n' \tag{170}$$

5.3 Classifier-free Guidance [6]

Recall that the classifier guidance requires to to train a classifier, where classifier takes $\mathbf{X}^{(t)}$ as an input for various noise level σ_t^2.

Given the trained score function, classifier guidance is based on

$$d\bar{\mathbf{X}}^{(t)} = (f - g^2(\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(\bar{\mathbf{X}}^{(t)}) + \omega \nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(Y|\bar{\mathbf{X}}^{(t)}))dt + g d\bar{\mathbf{Z}}^{(t)} \tag{171}$$

which replaced the conditional score function by

$$\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(\bar{\mathbf{X}}^{(t)}) + \omega \nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(Y|\bar{\mathbf{X}}^{(t)}))dt \tag{172}$$

which is also equivalent to

$$\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(\bar{\mathbf{X}}^{(t)}|Y) + (\omega - 1)\nabla_{\bar{\mathbf{X}}^{(t)}} \log p_t(Y|\bar{\mathbf{X}}^{(t)}))dt. \tag{173}$$

Classifier guidance rely on additional classifier, which is another neural network to train and compute the gradient for guidance. Instead, classifier-free guidance is an alternative method of modifying conditional score but without extra classifier. The key idea is to train a single model that learns unconditional score function $\nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}}(\mathbf{x}^{(n)})$ and conditional score function $\nabla_{\mathbf{x}^{(n)}} \log P_{\mathbf{X}^{(n)}|y}(\mathbf{x}^{(n)}|y)$ for all y. More precisely, the score function that we want to train is $\epsilon_\theta(\mathbf{X}^{(n)}, y)$ where $\epsilon_\theta(\mathbf{X}^{(n)}, y)$ corresponds to the conditional score function with label y and we allow $y = \emptyset$ so that $\epsilon_\theta(\mathbf{X}^{(n)}, \emptyset)$ represents the unconditional score function.

Then, the sampling is performed using

$$\omega \epsilon_\theta(\mathbf{X}^{(n)}, y) - (\omega - 1)\epsilon_\theta(\mathbf{X}^{(n)}, \emptyset) \tag{174}$$

for weight parameter $\omega > 1$.

Remark 13. *A possible interpretation of classifier-free guidance is that we guide a sample under label y with weight ω, then pull back the sample with unconditional score function with weight $\omega - 1$. Overall, the sample moved towards the label y with effective weight of 1.*

Note that the classifier guidance does not require to train score function. However, the training procedure is now modified for classifier-free guidance. Given the sample $(\mathbf{X}, Y)$, we replace $Y = \emptyset$ with given probability (to learn unconditional score function as well). Then, similar to the original DDPM training, we minimize

$$\tilde{\lambda}_n \|\epsilon - \epsilon_\theta(\mathbf{X}^{(n)}, y, n)\|^2. \tag{175}$$

6 Discrete Diffusion Models

6.1 Discrete-Time Discrete Diffusion

We aim to define a diffusion generative model in a discrete state space. To this end, it is necessary to introduce noise into the input data such that, as time $t \to \infty$ the process converges to a common noised distribution that is independent of the input. However, unlike in the continuous setting, discrete data does not admit arithmetic operations such as addition, making it impossible to inject noise in the same manner as in continuous diffusion models.

Thus, in the case of discrete data, we consider probabilistic transitions between data points as the noising process.

Let the state space be $\mathcal{X} = \{1, 2, \ldots, K\}$ and let X_t denote the data at time $t = 0, 1, 2, \ldots$. Let p_t be the corresponding probability mass vector, i.e.,

$$p_t = \begin{bmatrix} \Pr(X_t = 1) \\ \Pr(X_t = 2) \\ \vdots \\ \Pr(X_t = K) \end{bmatrix} \in [0, 1]^K$$

Defining transition probabilities means specifying all $\Pr(X_{t+1} = y \mid X_t = x)$. We may regard $\Pr(X_{t+1} = y \mid X_t = x)$ as the (y, x)-entry of some matrix Q_t, in which case we can express the update of the probability vector as:

$$p_{t+1} = Q_t p_t, \quad t = 0, 1, 2, \ldots$$

For this reason, we refer to Q_t as the *transition matrix*.

For this process to serve as a valid diffusion noising process, each p_t must remain a valid probability mass vector, and the sequence p_t should converge to a stationary distribution. That is,

$$\exists\, \pi \text{ s.t. } \quad \forall p_0,\ p_T = Q_T \cdots Q_2 Q_1 p_0 \to \pi \text{ as } T \to \infty$$

Two of the most common choices for the transition matrix Q_t are the *uniform* and *absorbing (masking)* types. In both cases, we typically assume a time-homogeneous setting where $Q_t = Q$ for all t.

Uniform Transition

$$Q_{\text{uniform}} = \begin{bmatrix} 1-\epsilon & \frac{\epsilon}{K-1} & \cdots & \frac{\epsilon}{K-1} \\ \frac{\epsilon}{K-1} & 1-\epsilon & \cdots & \frac{\epsilon}{K-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\epsilon}{K-1} & \frac{\epsilon}{K-1} & \cdots & 1-\epsilon \end{bmatrix}$$

That is, the transition probabilities are given by:

$$\Pr(X_{t+1} = y \mid X_t = x) = \begin{cases} 1-\epsilon & \text{if } y = x \\ \frac{\epsilon}{K-1} & \text{if } y \neq x \end{cases}$$

Absorbing (Masking) Transition

$$Q_{\text{absorb}} = \begin{bmatrix} 1-\epsilon & 0 & \cdots & 0 & 0 \\ 0 & 1-\epsilon & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-\epsilon & 0 \\ \epsilon & \epsilon & \cdots & \epsilon & 1 \end{bmatrix}$$

Accordingly, the transition probabilities are defined as:

$$\Pr(X_{t+1} = y \mid X_t = x) = \begin{cases} 1-\epsilon & \text{if } y = x \neq [\mathbf{M}] \\ \epsilon & \text{if } x \neq y = [\mathbf{M}] \\ 1 & \text{if } x = y = [\mathbf{M}] \\ 0 & \text{otherwise} \end{cases}$$

Here, $[\mathbf{M}]$ denotes a special *mask token*, which functions as an *absorbing state* such that once a token is masked, it remains masked in all future timesteps. As a result, the size of Q_{absorb} is $(K+1) \times (K+1)$, reflecting the inclusion of the mask token.

6.2 Score Entropy Discrete Diffusion Model

As in Gaussian diffusion, where the discrete-time forward process can be extended to a continuous-time stochastic differential equation (SDE), the discrete diffusion forward process discussed in the last subsection can similarly be extended to a continuous-time formulation. This yields the following ordinary differential equation (ODE):

$$\frac{dp_t}{dt} = Q_t p_t, \quad p_0 = p_{\text{data}}$$

To ensure that this defines a valid noising process, certain restrictions must be imposed. Also, closed-form expressions of Q_t that generalize the uniform and absorbing diffusion processes are known (we omit the details here).

The reverse process corresponding to the above forward process can be analytically described by:

$$\frac{dp_{T-t}}{dt} = \overline{Q}_{T-t} p_{T-t}, \quad \text{where} \begin{cases} \overline{Q}_t(y,x) = \dfrac{p_t(y)}{p_t(x)} Q_t(x,y) \;\; \text{if} \;\; x \neq y \\ \overline{Q}_t(x,x) = -\sum_{y \neq x} \overline{Q}_t(y,x) \end{cases}$$

The entries of the reverse transition rate matrix $\overline{Q}_t$ can be interpreted as follows. For the case $y \neq x$, the reverse transition probability is derived from Bayes' rule:

$$p_{t-\Delta t|t}(y|x) = p_{t|t-\Delta t}(x|y) \frac{p_{t-\Delta t}(y)}{p_t(x)}$$

By dividing both sides by Δt and taking the limit as $\Delta t \to 0$, we obtain the expression $\overline{Q}_t(y,x) = \dfrac{p_t(y)}{p_t(x)} Q_t(x,y)$. The diagonal element $\overline{Q}_t(x,x) = -\sum_{y \neq x} \overline{Q}_t(y,x)$ ensures mass conservation of p_t, i.e., it ensures $\sum_{i=1}^{K} p_t(i) = 1$ at all times.

In continuous diffusion, sampling is performed by estimating the score function $\nabla_{\mathbf{x}_t} \log P_{\mathbf{X}_t}(\mathbf{x}_t)$. Analogously, in discrete diffusion, modeling the reverse process requires estimating the marginal ratio $\frac{p_t(y)}{p_t(x)}$, which serves as the discrete counterpart of the score function and is denoted by $s(x,t)_y$. Explicitly, the score function is given by:

$$s(x,t) = \begin{bmatrix} \frac{p_t(1)}{p_t(x)} \\ \frac{p_t(2)}{p_t(x)} \\ \vdots \\ \frac{p_t(K)}{p_t(x)} \end{bmatrix} \in [0,\infty)^K$$

The score function $s(x,t)$ is then approximated by a neural network $s_\theta(x,t)$.

One early attempt to learn this score function was the method of *Concrete Score Matching* [9]. Analogous to the ℓ^2 loss used in continuous diffusion,

$$\| s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log P_{\mathbf{X}_t}(\mathbf{x}_t) \|^2,$$

concrete score matching minimizes the following objective:

$$\sum_{y \neq x} \left(s_\theta(x_t, t)_y - \frac{p_t(y)}{p_t(x)} \right)^2 .$$

However, this squared error loss does not penalize negative values of $s_\theta(x_t, t)_y$, which can hinder learning performance.

A successful objective for learning the ratio $\frac{p_t(y)}{p_t(x)}$ is the **Score Entropy Loss**, first introduced in [8], awarded Best Paper at ICML 2024:

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathbb{E}_{x_t \sim p_t} \left[\sum_{y \neq x_t} w_{x_t, y} \left(s_\theta(x_t, t)_y - \frac{p_t(y)}{p_t(x)} \log s_\theta(x_t, t)_y + C\left(\frac{p_t(y)}{p_t(x)} \right) \right) \right]$$

Here, $w_{x_t,y}$ are arbitrary nonnegative weights, and $C(a) = a(\log a - 1)$. It can be shown that if $w_{x_t,y} > 0$ for all x_t and y, the minimizer $\theta^\star$ of this loss satisfies $s_{\theta^\star}(x_t, t)_y = \frac{p_t(y)}{p_t(x)}$. Moreover, we can see that the term $\log s_\theta(x_t, t)_y$ prevents any nonpositive estimates of the score function.

Nevertheless, since the ratio $\frac{p_t(y)}{p_t(x)}$ is unknown, the score entropy loss is intractable in practice. An alternative yet equivalent formulation is the following *Implicit Score Entropy Loss*:

$$\mathcal{L}_{\mathrm{ISE}}(\theta) = \mathbb{E}_{x_t \sim p_t} \left[\sum_{y \neq x_t} w_{x_t, y} s_\theta(x_t, t)_y - w_{y, x_t} \log s_\theta(y, t)_{x_t} \right]$$

However, Monte Carlo estimation of this loss requires sampling an x_t and evaluating $s_\theta(y, t)_{x_t}$ for all $y \neq x_t$, which becomes intractable in high-dimensional settings. A commonly used approximation is to sample y uniformly, but this increases the variance of the estimator and may hinder training stability. This issue parallels challenges encountered in continuous diffusion, which have been addressed through sliced score matching. Motivated by this, we introduce an equivalent but tractable objective: the *Denoising Score Entropy Loss*: $\mathcal{L}_{\mathrm{DSE}}(\theta)$ is defined to be

$$\mathbb{E}_{x_0 \sim p_0,\, x_t \sim p_{t|0}(\cdot|x_0)} \left[\sum_{y \neq x_t} w_{x_t, y} \left(s_\theta(x_t, t)_y - \frac{p_{t|0}(y|x_0)}{p_{t|0}(x_t|x_0)} \log s_\theta(x_t, t)_y \right) \right]$$

This loss is tractable and provably equivalent to the original score entropy loss, enabling effective approximation of s_θ to the true marginal ratio.

Using the learned score function, one can define a parametrized reverse process and employ it for reverse-time sampling.

6.3 Practical Sequence Space

In practice, the state space we consider typically takes the form of a sequence space:

$$\mathcal{X} = \{1, 2, \ldots, K\}^L,$$

where $1, 2, \ldots, K$ represent vocabulary tokens and L denotes the sequence length. We denote a sequence by a boldface vector $\mathbf{x}_t$, and each token at position i by x_t^i, so that $\mathbf{x}_t = x_t^1 x_t^2 \cdots x_t^L$.

A key computational challenge in this setting is that the transition rate matrix Q_t for the forward ODE has exponential size, namely $K^L \times K^L$, leading to intractability. This issue can be mitigated by assuming that each token is perturbed independently according to an identical transition rate matrix of the form $Q_t^{\text{tok}} = \sigma(t)Q^{\text{tok}}$. Under this assumption, simulation of the forward process becomes significantly more tractable.

Moreover, this assumption implies the following sparsity condition:

$$Q_t(\mathbf{y}, \mathbf{x}) = 0 \quad \text{if} \quad d_{\text{H}}(\mathbf{y}, \mathbf{x}) \geq 2,$$

where d_{H} denotes the Hamming distance (i.e., the number of differing tokens between two sequences). This follows from the fact that, under the ODE framework, the probability of more than one token changing simultaneously in an infinitesimal time interval is negligible compared to the probability of a single-token change.

Turning to the reverse process, the reverse transition rate is given by:

$$\overline{Q}_t(\mathbf{y}, \mathbf{x}) = \frac{p_t(\mathbf{y})}{p_t(\mathbf{x})} Q_t(\mathbf{x}, \mathbf{y}) \quad \text{if} \quad \mathbf{x} \neq \mathbf{y}.$$

To approximate $\overline{Q}_t(\mathbf{y}, \mathbf{x})$, we substitute the unknown marginal ratio $\frac{p_t(\mathbf{y})}{p_t(\mathbf{x})}$ with the learned score function $s_\theta(\mathbf{x}, t)_{\mathbf{y}}$. However, since $Q_t(\mathbf{x}, \mathbf{y}) = 0$ whenever $d_{\text{H}}(\mathbf{x}, \mathbf{y}) \geq 2$, it follows that $\overline{Q}_t^\theta(\mathbf{y}, \mathbf{x}) = 0$ as well, and thus the corresponding $s_\theta(\mathbf{x}, t)_{\mathbf{y}}$'s are not required for modeling the reverse process.

Consequently, it is sufficient to train the score network only for sequence pairs $(\mathbf{x}, \hat{\mathbf{x}})$ such that $d_{\text{H}}(\mathbf{x}, \hat{\mathbf{x}}) = 1$.

References

[1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[4] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[6] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*, 2024.

[9] Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. In *NeurIPS*, 2022.

[10] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[11] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[12] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.