Fundamentals of Diffusion Model:

Denoising Diffusion Probabilistic Models (DDPM)

Donghyun Ko
Posted: June 2025
Updated: July 2025

Abstract

This tutorial provides a detailed explanation of the fundamentals of the DDPM-based diffusion model with a step-by-step mathematical derivation. This covers the concepts of forward(Noising) and reverse(De-noising) process, reparameterization technic to make them faster and simpler, loss function derivation - ELBO derivation and KL decomposition, deriving the simplified version of loss function to be used in codes, and glancing at the effect of a practical beta-scheduling. Each formula is accompanied by intuitive explanations so that readers who encounter these concepts for the first time can follow the logic and computations without being tortured.

Contents

1 Introduction

In recent years, diffusion models have emerged as a powerful class of generative models that rival GANs, VAEs, and normalizing flows, especially in image generation. The core idea is simple, yet effective: Start with a complex data distribution and gradually add Gaussian noise for T discrete steps until you reach (approximately) a standard normal distribution. In other words, we take a data sample and progressively corrupt it with small amounts of noise until it becomes pure noise. Then, we train a neural network to reverse that noising process step by step, effectively learning how to "denoise" at each stage. Once trained, one can sample from a standard normal distribution and apply the learned reverse transitions to remove the noise added at each time step and finally generate realistic data samples. (i.e., if we learned the noise distribution at each time step, we could remove the noise added at each time step to generate a new image which exists in the manifold.)

Formally, let $x_0 \sim p_{\text{data}}(x)$, where $p_{\text{data}}(x)$ denotes the unknown real data distribution (for example, images that have been normalized to zero mean and unit variance). We define a forward (noising) Markov chain $\{x_t\}_{t=0}^{T}$ by

$$q(x_t \mid x_{t-1}) \;=\; \mathcal{N}\big(x_t;\, \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I\big), \quad t = 1, 2, \ldots, T,$$

where each $\beta_t \in (0, 1)$ is a hyperparameter that controls the variance of the noise added at each time step t. Intuitively, the factor $\sqrt{1-\beta_t}$ scales down to the previous sample x_{t-1}, and then we add Gaussian noise of variance β_t.

In the large limit T, if $\sum_{t=1}^{T} \beta_t$ is sufficiently large, x_T becomes approximately distributed as $\mathcal{N}(0, I)$ regardless of the original x_0. It is not easy to directly obtain a closed form solution for the reverse process $q(x_{t-1} \mid x_t)$ for which I will mention the reason later. Instead, our goal is to learn a reverse process $p_\theta(x_{t-1} \mid x_t)$ that can approximate $q(x_{t-1} \mid x_t)$ so that if we sample

$$x_T \sim \mathcal{N}(0, I), \quad x_{t-1} \sim p_\theta(x_{t-1} \mid x_t) \quad (\text{for } t = T, T-1, \ldots, 1),$$

then the resulting x_0 will follow a distribution close to $p_{\text{data}}(x)$. In other words, we want our network to learn how to step-by-step "undo" forward noising.

To train p_θ, we derive a variational lower bound (ELBO) on $\log p_\theta(x_0)$ by using $q(x_{1:T} \mid x_0)$ as a proposal distribution in a standard variational inference framework. We then decompose this ELBO into KL divergence terms that compare the true posterior $q(x_{t-1} \mid x_t, x_0)$ and our model $p_\theta(x_{t-1} \mid x_t)$. With a clever parameterization of p_θ, we reduce the loss to a simple mean square error that trains the network to predict the noise added at each step. Finally, we discuss practical choices for the noise schedule $\{\beta_t\}$, which greatly impact sample quality.

2 Forward (Noising) Process

This section defines the forward noising process in detail and derives two key quantities: the closed-form marginal distribution $q(x_t \mid x_0)$ for any t and the posterior $q(x_{t-1} \mid x_t, x_0)$, which is needed later to derive the objective (loss) function of p_θ for training.

2.1 Interpret the Forward Process from the perspective of Markov Chain

We begin with $x_0 \sim p_{\text{data}}(x)$, where x_0 is assumed to have been standardized to zero mean and identity covariance. For $t = 1, 2, \ldots, T$, we define the following Gaussian transition:

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

where $\beta_t \in (0,1)$ is the noise variance at step t. Equivalently, we can write each step of the forward process(Figure 1) in generative form as

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

Here, ϵ_{t-1} denotes the fresh Gaussian noise injected at time $t-1$ and is considered i.i.d (independently and identically distributed) as ϵ_t for all 't.' We often denote

$$\alpha_t := 1 - \beta_t, \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s = \prod_{s=1}^{t} (1 - \beta_s)$$

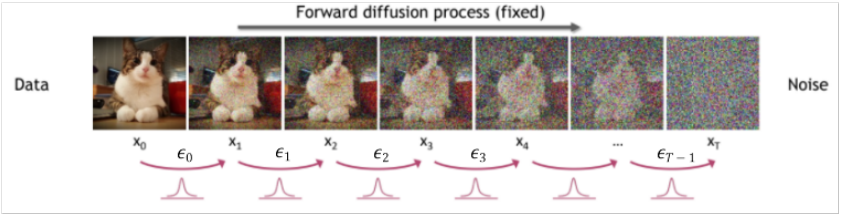

Figure 1: Forward Diffusion Process

Because we assume $\epsilon_t \sim \mathcal{N}(0, I)$, one can check by induction that for any t, x_t remains zero mean and unit variance whenever x_0 is zero mean and unit variance (details of this can be seen in Figure 2). Concretely,

$$\mathbb{E}[x_t] = \sqrt{\alpha_t}\, \mathbb{E}[x_{t-1}] + \sqrt{\beta_t}\, \mathbb{E}[\epsilon_{t-1}] = 0, \text{ and}$$
$$\mathrm{Var}(x_t) = (1-\beta_t)\, \mathrm{Var}(x_{t-1}) + \beta_t\, \mathrm{Var}(\epsilon_{t-1}) = (1-\beta_t)I + \beta_t I = I$$

Thus, each x_t is marginally distributed with covariance I (though not necessarily Gaussian for small t, strictly speaking, only in the limit it approaches normal). This "variance preservation" helps keep the forward process numerically stable.

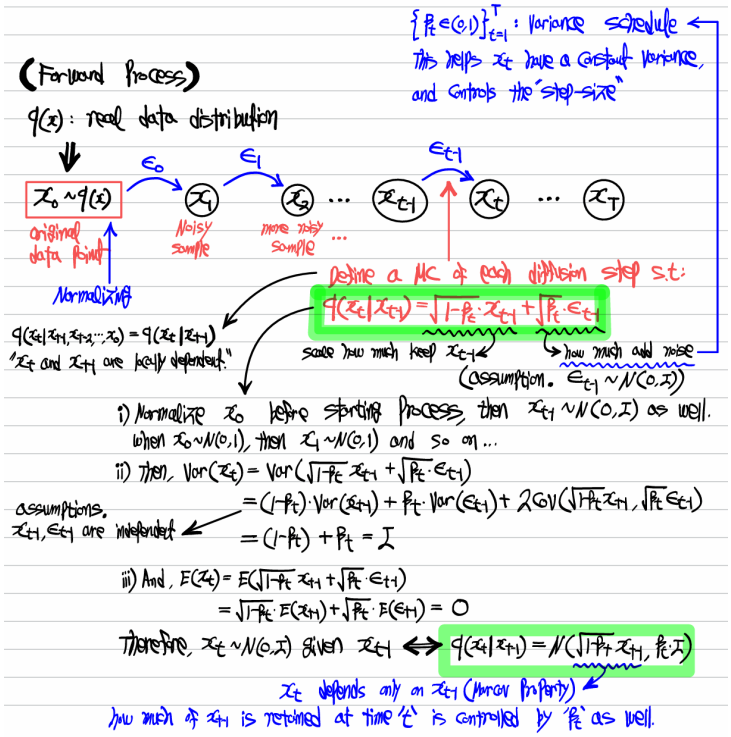

Figure 2: Handwritten derivation

2.2 [Reparameterization Trick] Simplifying Forward Process : $q(x_t \mid x_0)$

We now derive the exact marginal distribution of x_t given the initial x_0. Define again $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. We claim that

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

In other words, one can sample x_t directly from x_0 in one step as

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Below is a short proof by induction.

Proof by induction

- **Base case ($t = 1$).** By definition,

  $$q(x_1 \mid x_0) = \mathcal{N}(x_1; \sqrt{1-\beta_1}\, x_0,\ \beta_1 I) = \mathcal{N}\big(x_1; \sqrt{\alpha_1}\, x_0,\ (1-\alpha_1) I\big)$$

  and indeed $\bar{\alpha}_1 = \alpha_1$.

- **Inductive step.** Suppose at time $t-1$ we have

  $$q(x_{t-1} \mid x_0) = \mathcal{N}\big(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0,\ (1-\bar{\alpha}_{t-1})\, I\big)$$

  Then

  $$\begin{aligned} q(x_t \mid x_0) &= \int q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)\, dx_{t-1} \\ &= \int \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1},\ \beta_t I)\, \mathcal{N}\big(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0,\ (1-\bar{\alpha}_{t-1}) I\big)\, dx_{t-1} \end{aligned}$$

  Combining two Gaussians in the integral yields a Gaussian with mean and variance:

  $$\mathbb{E}[x_t] = \sqrt{\alpha_t}\, \mathbb{E}[x_{t-1}] = \sqrt{\alpha_t}\, (\sqrt{\bar{\alpha}_{t-1}}\, x_0) = \sqrt{\alpha_t\, \bar{\alpha}_{t-1}}\, x_0 = \sqrt{\bar{\alpha}_t}\, x_0$$

  $$\mathrm{Var}(x_t) = \alpha_t\, \mathrm{Var}(x_{t-1}) + \beta_t I = \alpha_t \left((1-\bar{\alpha}_{t-1}) I\right) + \beta_t I = \big(\alpha_t (1-\bar{\alpha}_{t-1}) + \beta_t\big)\, I$$

  Since $\alpha_t\, \bar{\alpha}_{t-1} = \bar{\alpha}_t$, one checks

  $$\alpha_t (1-\bar{\alpha}_{t-1}) + \beta_t = \alpha_t - \alpha_t\, \bar{\alpha}_{t-1} + \beta_t = 1 - \bar{\alpha}_t$$

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

This closed-form expression is extremely useful because it allows us to sample x_t in one shot given x_0, without iterating all the intermediate steps. (Details of this can be seen in Figure 3)

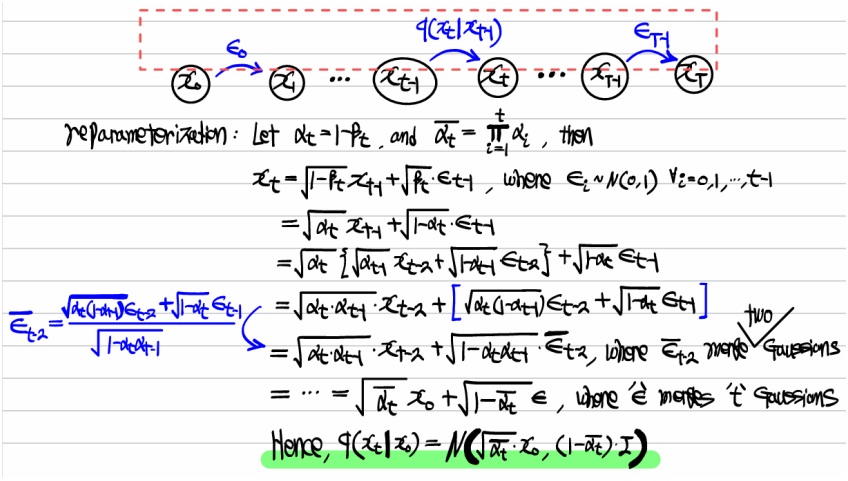

Figure 3: Handwritten derivation

2.3 How to use Forward Process to derive Posterior $q(x_{t-1} \mid x_t, x_0)$ closed form

As I mentioned earlier, it is intractable to obtain a closed form solution for the reverse process $q(x_{t-1} \mid x_t)$, because it requires integrating the entire distribution of x_0 conditioned on x_t, which is generally intractable due to the complex and unknown nature of the data distribution $p_{\text{data}}(x_0)$. However, the good news is that $q(x_{t-1} \mid x_t, x_0)$ is tractable, which we can use to mathematically define the objective(loss) function to train the diffusion model.

Mathematical derivation In diffusion models, the true posterior $q(x_{t-1} \mid x_t)$ is intractable because it involves integrating over the unknown data distribution. However, by leveraging the reverse-step Markov property, we can replace this intractable marginal posterior with a tractable conditional Gaussian that depends on the original sample x_0. Specifically:

- $q(x_{t-1} \mid x_t)$ **(intractable):** Using the conditional marginalization formula,

$$q(x_{t-1} \mid x_t) = \int q(x_{t-1}, x_0 \mid x_t)\, \mathrm{d}x_0 = \int q(x_{t-1} \mid x_t, x_0)\, q(x_0 \mid x_t)\, \mathrm{d}x_0,$$

  where $q(x_0 \mid x_t) = \frac{q(x_t|x_0)\, p_{\text{data}}(x_0)}{q(x_t)}$ is intractable due to unknown p_{data}, so we cannot evaluate this integral directly.

- $q(x_{t-1} \mid x_t, x_0)$ **(tractable):** By Bayes' rule on the forward process (using the Markov property and reparameterization trick) as shown in the Figure 4,

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

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t I), \quad q(x_{t-1} \mid x_0) = \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0, (1-\bar{\alpha}_{t-1})I), \text{ and}$$
$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t)I)$$

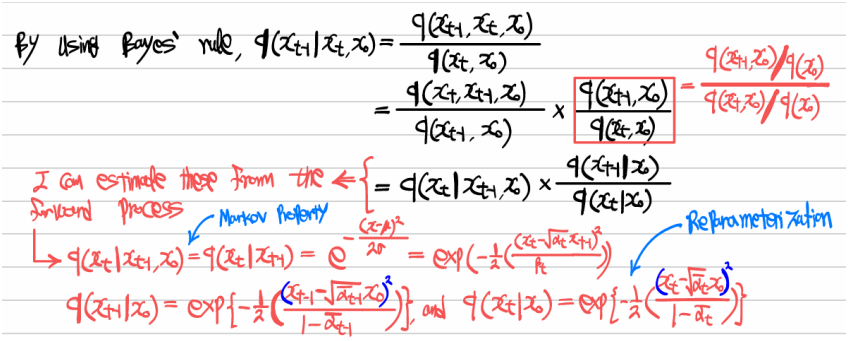

Figure 4: Handwritten derivation

Completing the square yields the known Gaussian as shown in the Figure 5:

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

with

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

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\, \beta_t}{1-\bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t \tag{2}$$

In practice, we often rewrite $\tilde{\mu}_t(x_t, x_0)$ in terms of the actual noise ϵ_t that was used to sample x_t from x_0. Recall from closed-form forward sampling:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

Solving for ϵ_t:

$$\epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{\sqrt{1 - \bar{\alpha}_t}}$$

One can then verify that

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_t\right)$$

This alternate form will be crucial when we parameterize the mean of our model to predict ϵ_t directly in the reverse process.

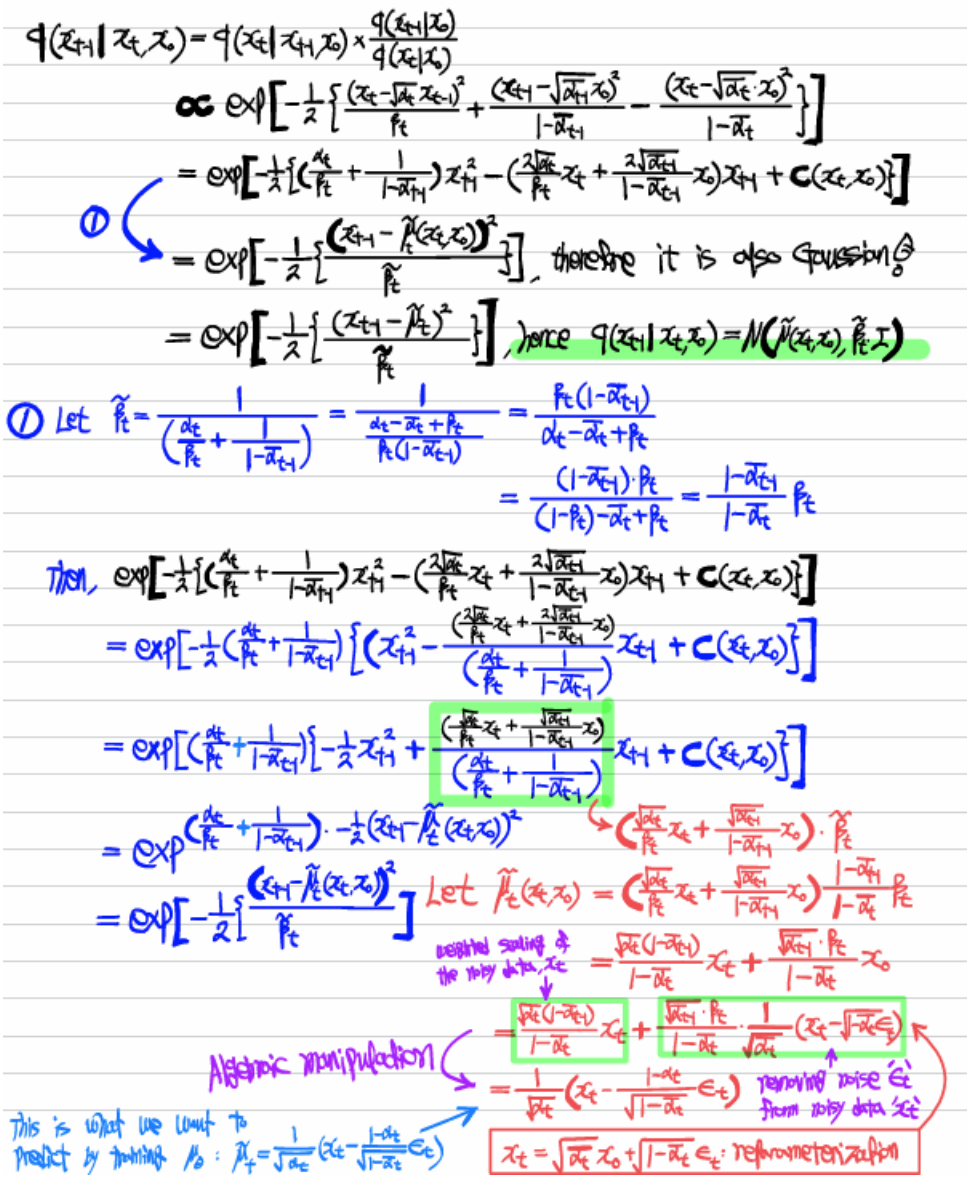

Figure 5: Handwritten derivation

Key insight (approximation): Building on the closed-form conditional posterior $q(x_{t-1} \mid x_t, x_0)$, we approximate the intractable marginal $q(x_{t-1} \mid x_t)$ by a Gaussian distribution whose parameters are learned by a neural network. Concretely, we define

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t),\, \sigma_\theta^2(x_t, t)\, I\big)$$

train μ_θ and σ_θ by minimizing the KL divergence;

$$\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)$$

so that

$$p_\theta(x_{t-1} \mid x_t) \;\approx\; q(x_{t-1} \mid x_t)$$

by matching them(μ_θ and σ_θ) to the tractable $\tilde{\mu}_t(x_t, x_0)$ and $\tilde{\beta}_t$ derived above. In practice, this means that the network learns to undo each forward noising step iteratively, denoising x_t one timestep at a time.

3 Reverse (Denoising) Process

We now introduce our parameterized reverse process $p_\theta(x_{t-1} \mid x_t)$, which does *not* have access to x_0, but should approximate the true posterior $q(x_{t-1} \mid x_t, x_0)$ as closely as possible. Since we assumed that the reverse process is also a Markov chain, $q(x_{t-1} \mid x_t, x_0) = q(x_{t-1} \mid x_t)$. If our parametric model $p_\theta(x_{t-1} \mid x_t)$ is well designed to approximate the reverse process $q(x_{t-1} \mid x_t)$, we can also assume that $p_\theta(x_{t-1} \mid x_t)$ is a Gaussian form such that

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

If we applied this reverse process from x_T to x_0 jointly, $p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)$ would be our generated output as seen in Figure 6.

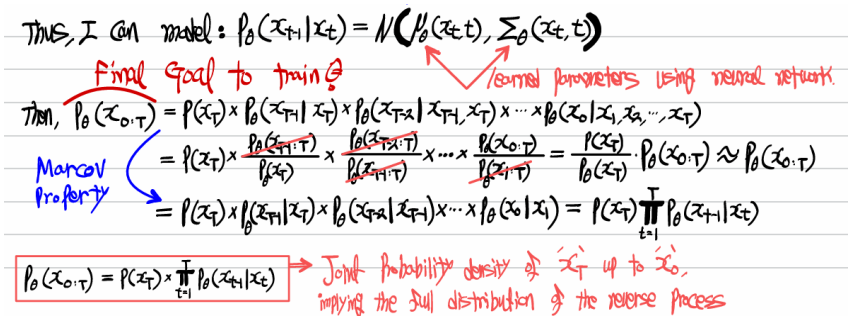

Figure 6: Handwritten derivation

3.1 "True Posterior" vs. "Learned Approximation"

The true posterior that we derived in the previous section is

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

However, at the generation time x_0 is unknown, so we cannot compute $\tilde{\mu}_t(x_t, x_0)$. Instead, we learn a parametric model:

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

where μ_θ and Σ_θ are outputs of a neural network with parameters θ. The network takes x_t and the time index t as inputs, and produces a mean vector $\mu_\theta(x_t, t)$ and a covariance matrix (often chosen to be diagonal or even a fixed scalar times identity) $\Sigma_\theta(x_t, t)$. Our training objective will encourage this learned Gaussian to match the true posterior as closely as possible in the KL(Kullback-Leibler)-divergence sense.

3.2 Joint Reverse Sampling Procedure

Once the model p_θ is trained, we generate new samples via the following procedure:

1. Draw an initial noise sample $x_T \sim p(x_T) = \mathcal{N}(0, I)$, which approximates the marginal $q(x_T)$ corrupted by the Forward process such that

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

2. For $t = T, T-1, \ldots, 1$, sample

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

3. After finishing down to $t = 1$, x_0 will be the our generated output as seen in Figure 7.

If $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ perfectly match the true posterior mean and variance (i.e., $\tilde{\mu}_t$ and $\tilde{\beta}_t$) at each step, then x_0 will be distributed in $p_{\text{data}}(x)$. In practice, we optimize our network so that p_θ approximates $q(x_{t-1} \mid x_t, x_0)$ as much as possible.

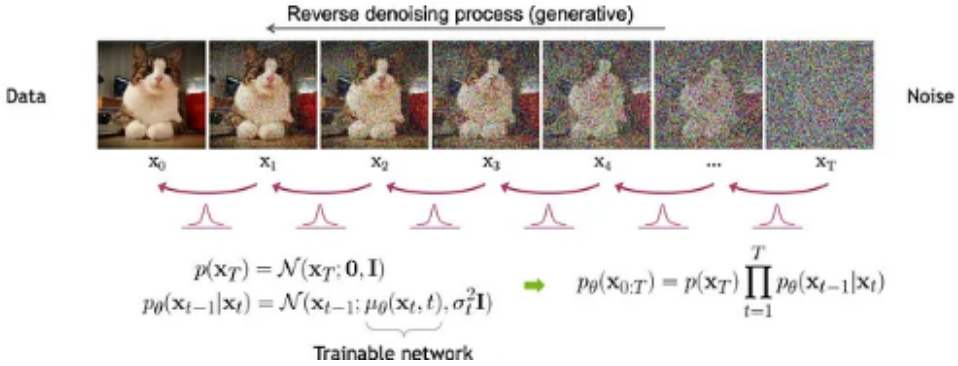

Figure 7: Reverse Diffusion Process

4 How to Define an Objective (Loss) Function

In Sections 2 and 3, we explored the fundamental structure of the diffusion model, focusing on the forward (noising) and reverse (denoising) processes. Now, we turn to the crucial task of training the reverse process $p_\theta(x_{t-1} \mid x_t)$ using a deep learning network. To do so, we need a well-defined objective function to guide learning. This section explains the rationale behind this objective function by introducing the necessary probabilistic and variational inference concepts, followed by a step-by-step derivation of the Evidence Lower Bound (ELBO), which serves as a surrogate objective.

4.1 Background Knowledge

1. **Kullback-Leibler Divergence** The KL-divergence between two distributions $q(x)$ and $p(x)$ is defined as:

$$D_{\mathrm{KL}}(q(x) \,\|\, p(x)) = \int q(x) \log \frac{q(x)}{p(x)} \, dx$$

It measures how different q is from p, and is always non-negative. Minimizing KL-divergence encourages q to be close to p.

2. **Jensen's Inequality and its Role in Variational Inference** For a convex function f, Jensen's inequality states:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$

For the logarithmic function, which is concave, this becomes:

$$\log \mathbb{E}[X] \geq \mathbb{E}[\log X]$$

This is the core inequality used to derive the ELBO.

3. **Definition and Interpretation of ELBO** Since directly maximizing $\log p_\theta(x_0)$ is intractable(this will be covered in 4.2), we introduce a tractable lower bound using a variational inference with the distribution $q(x_{1:T} \mid x_0)$:

$$\log p_\theta(x_0) = \log \int q(x_{1:T} \mid x_0) \, \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \, dx_{1:T} \geq \mathbb{E}_q \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \right]$$

We will define later such that:

$$\boxed{\mathrm{ELBO}(x_0; \theta) = \mathbb{E}_q \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \right]}$$

as a surrogate training objective, with $\log p_\theta(x_0) \geq \mathrm{ELBO}(x_0; \theta)$. Hence, maximizing the *ELBO* provides a principled surrogate to maximize the true logarithmic likelihood $\log p_\theta(x_0)$, since it serves as a variational lower bound. The detailed derivation and decomposition of the ELBO will be discussed in the following sections.

4.2 Intractability of Direct $\log p_\theta(x_0)$

The ultimate goal of training a diffusion model is to maximize the marginal likelihood $\log p_\theta(x_0)$ over the data distribution. This objective reflects how well the model captures the true data distribution using a parameterized reverse process. By the definition of the generative model, the marginal likelihood can be expressed as:

$$p_\theta(x_0) = \int p_\theta(x_{0:T}) \, dx_{1:T} = \int p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t) \, dx_{1:T}$$

As we have seen earlier, each reverse transition distribution $p_\theta(x_{t-1} \mid x_t)$ is assumed to follow a Gaussian form:

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

both $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are outputs of a deep neural network (e.g., UNet). Since μ_θ is a complex non-linear function represented by a neural network rather than an explicit closed-form expression, the above integral cannot be evaluated analytically. Moreover, the marginal likelihood itself involves a high-dimensional integral:

$$p_\theta(x_0) = \int p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t) \, dx_{1:T},$$

which expands over T latent variables $x_1, \ldots, x_T$, each of which lies in $\mathbb{R}^d$. This results in an integration space of dimension $T \times d$. As T increases (e.g., to hundreds or thousands of steps), evaluating this integral becomes not only computationally infeasible but also analytically intractable. Therefore, we resort to variational inference techniques such as ELBO-based optimization, which provide a tractable surrogate to train the diffusion model.

4.3 How to derive objective(loss) function

To derive a tractable objective function for training the diffusion model, we apply variational inference techniques based on Jensen's inequality. Begin with the forward joint distribution:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}),$$

and the reverse generative model:

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

Then, the marginal likelihood becomes:

$$p_\theta(x_0) = \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \, dx_{1:T}$$

By taking log on both sides,

$$\begin{aligned}
\log p_\theta(x_0) &= \log \left[\int q(x_{1:T} \mid x_0) \cdot \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \, dx_{1:T} \right] \\
&= \log \left[\mathbb{E}_{q(x_{1:T}|x_0)} \left(\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right) \right] \\
&\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \left(\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right) \right] \quad \text{(by Jensen's Inequality)} \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \right]
\end{aligned}$$

$$\boxed{\log p_\theta(x_0) \;\geq\; \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)} \Big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \Big]}_{\mathcal{L}(\theta; x_0) \text{ (ELBO)}}} \tag{3}$$

In another sense,

$$\begin{aligned}
\log p_\theta(x_0) &\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \left(\frac{p_\theta(x_0) \cdot p_\theta(x_{1:T} \mid x_0)}{q(x_{1:T} \mid x_0)} \right) \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_0) - \log \left(\frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{1:T} \mid x_0)} \right) \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} [\log p_\theta(x_0)] - \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \left(\frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{1:T} \mid x_0)} \right) \right] \\
&= \log p_\theta(x_0) - \int \log \left(\frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{1:T} \mid x_0)} \right) \cdot q(x_{1:T} \mid x_0) \, dx_{1:T} \quad \text{(by KL-divergence)} \\
&= \log p_\theta(x_0) - D_{\mathrm{KL}} \left(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0) \right)
\end{aligned}$$

$$\boxed{\log p_\theta(x_0) \;\geq\; \log p_\theta(x_0) - D_{\mathrm{KL}}\Big(\underbrace{q(x_{1:T} \mid x_0)}_{\substack{\text{True forward}\\ \text{process}}} \;\Big\|\; \underbrace{p_\theta(x_{1:T} \mid x_0)}_{\substack{\text{Modeled forward}\\ \text{process}}}\Big)} \tag{4}$$

By combining (3), (4), and the definition of KL divergence, we have the exact identity s.t:

$$\log p_\theta(x_0) = \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}\big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\big]}_{\mathcal{L}(\theta;x_0)} \;+\; \underbrace{D_{\mathrm{KL}}\big(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0)\big)}_{\geq 0} \tag{5}$$

Taking the negative on both sides of (3) and using $D_{\mathrm{KL}} \geq 0$ gives

$$-\log p_\theta(x_0) \;\leq\; -\mathcal{L}(\theta; x_0) \;+\; D_{\mathrm{KL}}\big(q(x_{1:T} \mid x_0) \,\|\, p_\theta(x_{1:T} \mid x_0)\big),$$

so that, in practice, we optimize the tractable surrogate negative ELBO $[-\mathcal{L}(\theta; x_0)]$ as a proxy to minimize the true negative log-likelihood of $p_\theta(x_0)$.

In diffusion model, the ELBO further decomposes into a sum of timestep-wise KL terms (this will be covered in 4.4 with more details):

$$-\mathcal{L}(\theta; x_0) = \underbrace{D_{\mathrm{KL}}\big(q(x_T \mid x_0) \,\|\, p_\theta(x_T)\big)}_{\text{prior matching}} \;+\; \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}\Big[D_{\mathrm{KL}}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)\Big]$$

By minimizing each timestep's KL divergence individually, we teach the model's denoising kernel $p_\theta(x_{t-1} \mid x_t)$ to match the true conditional $q(x_{t-1} \mid x_t, x_0)$ at every step. Although this procedure does not guarantee a perfect match across the entire sequence at once, reducing the error step by step results in an overall close alignment of the full reverse process.

Because time-step decomposition makes the KL of each reverse step tractable, minimizing $-\mathcal{L}(\theta; x_0)$ serves as an effective and computationally feasible proxy to reduce the true negative logarithmic likelihood of $p_\theta(x_0)$.

4.4 ELBO Decomposition and KL-Based Training Objective

In the previous subsection, we introduced the Evidence Lower Bound (ELBO) as a principal surrogate objective to circumvent the intractability of directly maximizing the marginal log-likelihood $\log p_\theta(x_0)$. By Jensen's inequality, we have:

$$\log p_\theta(x_0) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\right],$$

which justifies using the ELBO as a lower bound on $\log p_\theta(x_0)$ for model training.

To turn this abstract formulation into a practical training objective, we now expand and decompose each term of the ELBO based on the known structure of the forward and reverse processes in diffusion models. The forward process q is a fixed Markov chain that progressively adds noise, while the reverse process p_θ is parameterized by a neural network trained to remove noise. Recall the generative and inference processes:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \quad q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}),$$

where q is the fixed forward (noising) process and p_θ is the learnable reverse (denoising) process.

Substituting these into the ELBO gives:

$$\begin{aligned}
\mathrm{ELBO}(x_0;\theta) &= \mathbb{E}_{q(x_{1:T}|x_0)}\Big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\Big] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p(x_T) + \sum_{t=1}^{T} \log p_\theta(x_{t-1} \mid x_t) - \sum_{t=1}^{T} \log q(x_t \mid x_{t-1})\right] \\
&= \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log p(x_T)]}_{\text{prior term}} + \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(x_{t-1} \mid x_t)]}_{\text{reverse transitions}} - \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log q(x_t \mid x_{t-1})]}_{\text{forward transitions}}
\end{aligned}$$

Decomposition Strategy: Each pair $\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)$ could be rewritten as a KL divergence between the reverse model and the true posterior.

The prior term $\mathbb{E}_{q(x_{1:T}|x_0)}[\log p(x_T)]$ in the ELBO does not depend on the model parameters θ. As a result, when we take the gradient of the ELBO with respect to θ during optimization, this term contributes nothing—the gradient is zero. Therefore, it can be safely omitted from the loss function used in training. In practice, only the terms involving θ, such as the reverse and forward transition terms, are retained and optimized. To simplify, consider the term

$$\mathbb{E}_{q(x_{1:T}|x_0)}\big[\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\big]$$

Law of Total Expectation & Markov Property

Law of total expectation: The law of total expectation (tower property) states that for any random variables X, Y and any function f:

$$\mathbb{E}[f(X,Y)] = \mathbb{E}\left[\,\mathbb{E}\big[f(X,Y) \mid Y\big]\,\right]$$

Example: Roll a fair die to obtain $Y \in \{1,\dots,6\}$. Then toss a coin X with head-probability p_y depending on the die result. The overall head-probability is:

$$\mathbb{E}[X] = \sum_{y=1}^{6} \mathbb{E}[X \mid Y = y] \cdot P(Y = y) = \frac{1}{6}\sum_{y=1}^{6} p_y$$

Markov Property: A stochastic process $\{X_t\}_{t\geq 0}$ is Markov if

$$P\big(X_{t+1} = x \mid X_t = x_t,\, X_{t-1} = x_{t-1}, \dots\big) = P\big(X_{t+1} = x \mid X_t = x_t\big)$$

Combining both concepts: By the Markov property for the chain $X_0 \to \cdots \to X_T$, we have:

$$q(X_{t-1}, X_t \mid X_0) = q(X_t \mid X_0)\, q(X_{t-1} \mid X_t, X_0) = q(X_t \mid X_0)\, q(X_{t-1} \mid X_t)$$

Then, by the law of total expectation:

$$\mathbb{E}[f(X_{t-1}, X_t)] = \mathbb{E}_{q(X_t|X_0)}\left[\mathbb{E}_{q(X_{t-1}|X_t)}\big[f(X_{t-1}, X_t)\big]\right]$$

In other words, we first average over X_{t-1} conditioned on X_t, then average over X_t.

Using the law of total expectation with the Markov property, we can rewrite ELBO term as:

$$\mathbb{E}_{q(x_t|x_0)}\left[\mathbb{E}_{q(x_{t-1}|x_t,x_0)}\left[\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\right]\right]$$

Equivalently, in integral form:

$$
\begin{aligned}
&= \int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \left(\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t) \right) dx_{t-1} \right] dx_t \\
&= \int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1} \right] dx_t \\
&= \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)) \right]
\end{aligned}
$$

We now elaborate on the transition from the expression:

$$
\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \left(\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t) \right) dx_{t-1} \right] dx_t
$$

to the KL divergence form:

$$
\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1} \right] dx_t
$$

The key step involves replacing the term $\log q(x_t \mid x_{t-1})$ with an expression involving $q(x_{t-1} \mid x_t, x_0)$. To see why this is possible, consider the identity of the joint distribution under the forward process:

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

Taking the logarithm on both sides yields:

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

This can be rearranged as:

$$
\log q(x_t \mid x_{t-1}) = \log q(x_{t-1} \mid x_t, x_0) + \log q(x_t \mid x_0) - \log q(x_{t-1} \mid x_0)
$$

Now consider plugging this into the earlier expectation:

$$
\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \Big(\log q(x_{t-1} \mid x_t, x_0) + \log q(x_t \mid x_0) - \log q(x_{t-1} \mid x_0) - \log p_\theta(x_{t-1} \mid x_t) \Big) dx_{t-1} \right] dx_t
$$

Observe that both $\log q(x_t \mid x_0)$ and $\log q(x_{t-1} \mid x_0)$ are independent of x_{t-1} and can be pulled out of the inner integral. Furthermore, since they do not depend on θ, they vanish when computing gradients with respect to θ, and thus can be safely ignored in optimization. This simplifies the expression to:

$$
\int q(x_t \mid x_0) \left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1} \right] dx_t
$$

Finally, we identify this expression as the expected KL divergence:

$$
\mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t) \right) \right]
$$

The KL divergence formulation derived earlier becomes a building block for constructing the training loss. Specifically, we define a per-timestep loss function L_t for each time step $t = 1, \ldots, T-1$ as follows:

$$
L_t := \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)) \right]
$$

This term measures how well the learned reverse transition distribution $p_\theta(x_{t-1} \mid x_t)$ approximates the true posterior $q(x_{t-1} \mid x_t, x_0)$ for each denoising step in the diffusion process. The expectation over $q(x_t \mid x_0)$ ensures that this comparison is averaged over possible noisy inputs.

At the final diffusion step $t = T$, the forward marginal distribution $q(x_T \mid x_0)$ and the prior $p(x_T)$ are both Gaussian, and their KL divergence can be computed in closed form:

$$q(x_T \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_T}x_0, (1 - \bar{\alpha}_T)I), \quad L_T := D_{\mathrm{KL}}(q(x_T \mid x_0) \,\|\, p(x_T))$$

This term quantifies how far the final noised sample x_T deviates from the assumed prior, and since both distributions are Gaussian, L_T is often easy to precompute or implement analytically.

At the starting point of the reverse process $t = 0$, our aim is to reconstruct the clean data x_0 from the first denoised latent x_1. This is expressed as a negative log-likelihood:

$$L_0 := -\mathbb{E}_{q(x_1|x_0)} \left[\log p_\theta(x_0 \mid x_1)\right]$$

Unlike the other L_t terms, L_0 is not a KL divergence—it directly measures how well the model can generate x_0 given x_1. In practice, this is often approximated by a simple Mean Squared Error (MSE) loss between x_0 and its predicted reconstruction.

Therefore, the original ELBO can be decomposed like the following box, and this decomposition enables modular training, allowing approximations (e.g., skip L_0), surrogate losses, and per-timestep control over reverse process learning.

Negative ELBO as Training Objective:

$$-\mathrm{ELBO}(x_0; \theta) = L_T + \sum_{t=1}^{T-1} L_t + L_0$$

Training Implications.

- L_T is a KL divergence between Gaussians and can often be computed in closed form, making it efficient and stable.
- L_0 reflects the reconstruction error at the very end of the reverse process. It is sometimes replaced by MSE in practice to simplify training.
- $\sum_{t=1}^{T-1} L_t$ constitutes the main loss driving the learning of the reverse denoising transitions.

5 Closed-Form of KL Terms

We now derive an explicit expression for each intermediate KL divergence term that appears in the objective(loss) function to train a diffusion model:

$$L_t = \mathbb{E}_{q(x_t|x_0)} \left[D_{\mathrm{KL}} \left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)\right], \quad 1 \le t \le T - 1$$

Both the true posterior $q(x_{t-1} \mid x_t, x_0)$ and the model $p_\theta(x_{t-1} \mid x_t)$ are assumed to be multivariate Gaussians with diagonal (often isotropic) covariance matrices. Therefore, we can use **the closed form formula for the KL divergence between two Gaussians**.

Gaussian-to-Gaussian KL Formula

The KL divergence between two multivariate Gaussian distributions in $\mathbb{R}^k$, each with diagonal covariance, is given by:

$$D_{\mathrm{KL}}\left(\mathcal{N}(\mu_1, \sigma_1^2 I) \,\|\, \mathcal{N}(\mu_2, \sigma_2^2 I)\right) = \frac{1}{2}\left(\frac{\sigma_1^2}{\sigma_2^2} + \frac{\|\mu_1 - \mu_2\|^2}{\sigma_2^2} - k + k \log \frac{\sigma_2^2}{\sigma_1^2}\right)$$

This expression compares two Gaussian distributions:

- $\mathcal{N}(\mu_1, \sigma_1^2 I)$: the **true posterior** $q(x_{t-1} \mid x_t, x_0)$
- $\mathcal{N}(\mu_2, \sigma_2^2 I)$: the **learned reverse model** $p_\theta(x_{t-1} \mid x_t)$

Each term in the KL formula has an intuitive interpretation:

- $\frac{\sigma_1^2}{\sigma_2^2}$: captures how the posterior's variance compares to the model's variance
- $\frac{\|\mu_1 - \mu_2\|^2}{\sigma_2^2}$: measures the difference between means, normalized by model variance
- $-k$: compensates for dimensionality
- $k \log \frac{\sigma_2^2}{\sigma_1^2}$: accounts for the entropy gap between the distributions

This closed-form is particularly useful in diffusion models, where the mean and variance of both $q(x_{t-1} \mid x_t, x_0)$ and $p_\theta(x_{t-1} \mid x_t)$ are explicitly available. This allows us to compute the KL divergence for each timestep efficiently, without sampling or numerical approximation. Moreover, since all involved distributions are Gaussian, the entire loss can be evaluated analytically.

5.1 Comparing Posterior vs. Model

Using the known forms:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I), \quad p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

The KL divergence is then:

$$D_{\mathrm{KL}}(q \,\|\, p_\theta) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_\theta^{-1}\tilde{\beta}_t I) + (\mu_\theta - \tilde{\mu}_t)^\top \Sigma_\theta^{-1}(\mu_\theta - \tilde{\mu}_t) - k + \ln \frac{\det \Sigma_\theta}{(\tilde{\beta}_t)^k}\right]$$

Simplification with fixed covariance: In practice, we often set $\Sigma_\theta(x_t, t) = \tilde{\beta}_t I$. Then:

$$\mathrm{tr}((\tilde{\beta}_t I)^{-1}\tilde{\beta}_t I) = k, \quad \ln \frac{\det(\tilde{\beta}_t I)}{(\tilde{\beta}_t)^k} = 0$$

The KL divergence simplifies to:

$$D_{\mathrm{KL}} = \frac{1}{2\tilde{\beta}_t}\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2$$

Thus,

$$\boxed{L_t \approx \mathbb{E}_{q(x_t|x_0)}\left[\frac{1}{2\tilde{\beta}_t}\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2\right] + \mathrm{C}}$$

5.2 Rewriting $\tilde{\mu}_t$ in Terms of Noise

Using the reparameterization trick:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

$$\epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t} x_0}{\sqrt{1-\bar{\alpha}_t}}$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1-\alpha_t}\,\epsilon_t)$$

We now parameterize:

$$\boxed{\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(x_t, t))}$$

Thus,

$$\mu_\theta - \tilde{\mu}_t = -\frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}(\epsilon_\theta - \epsilon_t)$$

$$\|\mu_\theta - \tilde{\mu}_t\|^2 = \frac{1-\alpha_t}{\alpha_t}\|\epsilon_\theta - \epsilon_t\|^2$$

Hence,

$$\frac{1}{2\tilde{\beta}_t}\|\mu_\theta - \tilde{\mu}_t\|^2 \propto \|\epsilon_\theta - \epsilon_t\|^2$$

5.3 Conclusion: Noise-Prediction Loss

Putting all the derivations together, we arrive at a much simpler training objective by assuming that the model variance $\Sigma_\theta(x_t, t)$ is fixed and equal to the posterior variance $\tilde{\beta}_t I$. In this setting, the KL divergence at each timestep reduces to a weighted squared error between the predicted mean $\mu_\theta(x_t, t)$ and the posterior mean $\tilde{\mu}_t(x_t, x_0)$:

$$L_t \approx \mathbb{E}_{q(x_t|x_0)}\left[\frac{1}{2\tilde{\beta}_t}\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2\right] + \text{const}$$

By expressing both means in terms of the underlying noise using the reparameterization trick:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

we showed that:

$$\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0) = -\frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\,(\epsilon_\theta(x_t, t) - \epsilon_t)$$

and thus,

$$\|\mu_\theta - \tilde{\mu}_t\|^2 = \frac{1-\alpha_t}{\alpha_t}\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2$$

Substituting this into the loss expression reveals that minimizing the KL divergence is equivalent to minimizing the squared difference between the predicted noise of the model $\epsilon_\theta(x_t, t)$ and the true noise ϵ_t. This yields the simplified training loss:

$$L_t^{\text{simple}} = \mathbb{E}_{q(x_t|x_0)}\left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2\right]$$

This elegant reformulation has become the standard objective in Denoising Diffusion Probabilistic Models (DDPM). Instead of directly predicting x_0 or x_{t-1}, the model learns to predict

the noise ϵ_t added to x_0 at timestep t. This approach has several benefits: 1) it simplifies the training objective, leverages Gaussian likelihoods, and 2) it allows the model to implicitly learn the reverse process through denoising.

This noise-prediction formulation forms the foundation of most modern diffusion models and enables efficient and scalable training across high-dimensional data domains such as images, audio, and time-series.

$$\boxed{L_t^{\text{simple}} = \mathbb{E}_{q(x_t|x_0)}\left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2\right]}$$

The full training objective aggregates the loss across all timesteps, which forms the foundation of DDPM training.

$$\boxed{L_{\text{total}} = \sum_{t=1}^{T} \mathbb{E}_{q(x_t,x_0)}\left[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2\right] + \text{const}}$$

6 Simplified Training Objective: Noise-Prediction Loss

After all the ELBO algebra, one arrives at a remarkably simple training procedure. Concretely, for each data point x_0:

1. Sample a random time index $t \sim \text{Uniform}\{1, 2, \ldots, T\}$
2. Draw noise $\epsilon \sim \mathcal{N}(0, I)$
3. Compute the noised sample
$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$$
Equivalently, x_t is sampled from $q(x_t \mid x_0)$
4. Feed (x_t, t) into the network to obtain $\hat{\epsilon} = \epsilon_\theta(x_t, t)$
5. Compute the mean-squared error loss
$$L = \|\hat{\epsilon} - \epsilon\|^2,$$
and backpropagate to update θ.

Over a minibatch of size B, one simply averages these losses:

$$\frac{1}{B} \sum_{i=1}^{B} \sum_{t=1}^{T} \|\epsilon_\theta(x_t^{(i)},\, t) - \epsilon^{(i)}\|^2, \text{ where}$$

each $(x_t^{(i)}, \epsilon^{(i)}, t^{(i)})$ is sampled independently. All constant factors from the original ELBO can be ignored because they do not affect gradient-based optimization.

Advantages of Noise-Prediction Loss

- **Simplicity:** Instead of computing multiple Gaussian KL terms, we only need a single regression objective ℓ_2 at each time step.
- **Numerical Stability:** Predicting ϵ_t resembles a standard denoising/regression task, which is stable to train with common optimizers (Adam, etc.).

- **Empirical Success:** Ho et al.(2020)[1] and follow-up works have shown that this objective yields excellent sample quality on image benchmarks.

7 Beta Scheduling

The choice of noise schedule $\{\beta_t\}$ (and hence $\{\alpha_t, \bar{\alpha}_t\}$) significantly impacts model performance. Intuitively, the schedule controls how quickly or slowly we move from the clean data at $t = 0$ to near-pure noise at $t = T$.

7.1 Linear Schedule (Ho *et al.*, 2020)

A simple and widely used choice is a *linear interpolation* between two bounds $\beta_{\min}$ and $\beta_{\max}$:

$$\boxed{\beta_t = \beta_{\min} + \frac{t-1}{T-1}(\beta_{\max} - \beta_{\min}), \quad t = 1, 2, \ldots, T,}$$

where typical values are $\beta_{\min} = 10^{-4}$ and $\beta_{\max} = 0.02$. In this case,

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

and as t increases, $\bar{\alpha}_t$ decreases smoothly from 1 down toward 0 as shown in the Figure 8. This "linear-in-variance" schedule works surprisingly well in practice because it gradually increases the noise at each step.

Pros:

- Simple and easy to implement.
- Surprisingly effective despite its simplicity.

Cons:

- The signal is lost too quickly in the early timesteps.
- It is difficult for the model to recover the fine structure from heavily noised inputs.

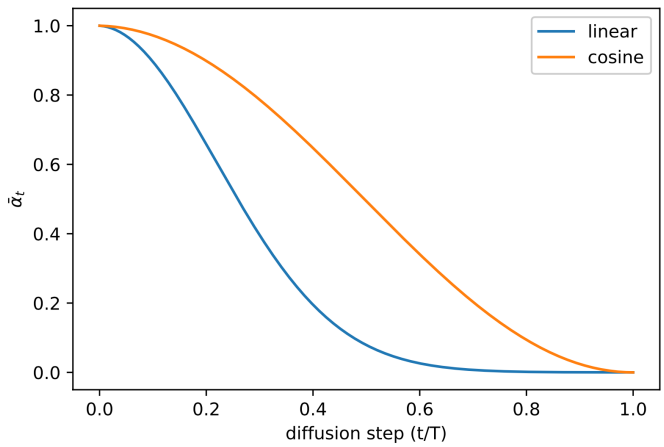

Figure 8: Linear and Cosine scheduler

7.2 Cosine Schedule (Nichol & Dhariwal, 2021)

Nichol and Dhariwal proposed a cosine-based schedule that places more weight on preserving the signal in the early steps and then injecting noise more rapidly near the end. Define

$$f(t) = \cos\Big(\frac{t/T + s}{1+s}\,\frac{\pi}{2}\Big)^2, \quad s = 0.008,$$

and set

$$\boxed{\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad \beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, \quad t = 1, \ldots, T}$$

Here, **s** is a small "offset" to ensure numerical stability and smoothness. This schedule starts with very small noise for small t, then increases gradually, and near the end jumps more strongly so that x_T is effectively standard normal. Empirical results show that the cosine schedule often yields lower negative log-likelihoods and better sample quality compared to the linear schedule.

Pros:

- Preserves more signal in early timesteps.
- This leads to better sample quality and lower negative log-likelihood.
- Better suited for high-resolution or fine-detail synthesis.

Cons:

- Slightly more complicated to implement.
- Needs precomputing the full schedule due to cumulative dependency.

The key difference between the cosine and linear noise schedules lies in how they inject noise over time as shown in the Figure 9. The linear schedule increases noise at a constant rate, causing the signal to degrade quickly in the early steps. In contrast, the cosine schedule adds noise slowly at first and accelerates it toward the end, allowing the model to preserve more signal in the early diffusion steps. As a result, the cosine schedule often leads to better sample quality and training stability, especially for high-resolution data or complex structures.

Figure 9: Performance by Linear and Cosine scheduler

7.3 Other Nonlinear Schedules

Researchers have also tried various other shapes for β_t:

- **Quadratic Schedule:** Define $\beta_t \propto t^2$ (rescaled into $[\beta_{\min}, \beta_{\max}]$). This injects noise more slowly at first (like cosine) but then accelerates quadratically.
- **Sigmoid Schedule:** Use a logistic curve so that noise grows slowly at first, then more quickly in the middle, and plateaus near the end.

- **V-P-B Schedule:** Derived from continuous-time diffusion processes and their variational bounds to optimize likelihood directly.

Nonetheless, linear and cosine schedules remain the two most commonly used in practice, as they balance simplicity and performance.

8 Algorithm Summary

Below is a concise overview of the full training and sampling algorithms, including all steps needed to implement a diffusion model in the code. A summarized algorithms for both training and sampling is given in Figure 10.

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Figure 10: Algorithm summary

8.1 Training Procedure

1. **Inputs and Hyperparameters:**
   - Data samples $x_0 \sim p_{\text{data}}$ (assumed normalized to mean 0 and covariance I)
   - Number of diffusion steps T (e.g., $T = 1000$)
   - Noise schedule $\{\beta_t\}_{t=1}^T$ (compute $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$)
   - Denoising network $\epsilon_\theta(x_t, t)$ that outputs a predicted noise vector of the same dimension as x_t
2. **Repeat until convergence:**
   (a) Sample a mini-batch of clean data $\{x_0^{(i)}\}_{i=1}^B$ from the dataset.
   (b) For each i in the batch:
      i. Draw a time index $t^{(i)} \sim \text{Uniform}\{1, \dots, T\}$
      ii. Sample noise $\epsilon^{(i)} \sim \mathcal{N}(0, I)$
      iii. Compute the noised sample

$$x_{t^{(i)}}^{(i)} = \sqrt{\bar{\alpha}_{t^{(i)}}}\, x_0^{(i)} + \sqrt{1 - \bar{\alpha}_{t^{(i)}}}\, \epsilon^{(i)}$$

      In vectorized code, one can sample all $t^{(i)}$ and $\epsilon^{(i)}$ in a batch
      iv. Pass $(x_{t^{(i)}}^{(i)}, t^{(i)})$ through the network to predict $\hat{\epsilon}^{(i)} = \epsilon_\theta(x_{t^{(i)}}^{(i)}, t^{(i)})$
      v. Compute the mean-squared error loss

$$L^{(i)} = \|\hat{\epsilon}^{(i)} - \epsilon^{(i)}\|^2$$

(c) Compute the batch loss

$$\frac{1}{B}\sum_{i=1}^{B} L^{(i)},$$

and update the network parameters θ by taking a gradient step (e.g., with Adam).

3. **Stopping Criteria:**

   - Validation loss has converged, or
   - Generated samples achieve satisfactory perceptual/quantitative quality (e.g., low FID, and etc.).

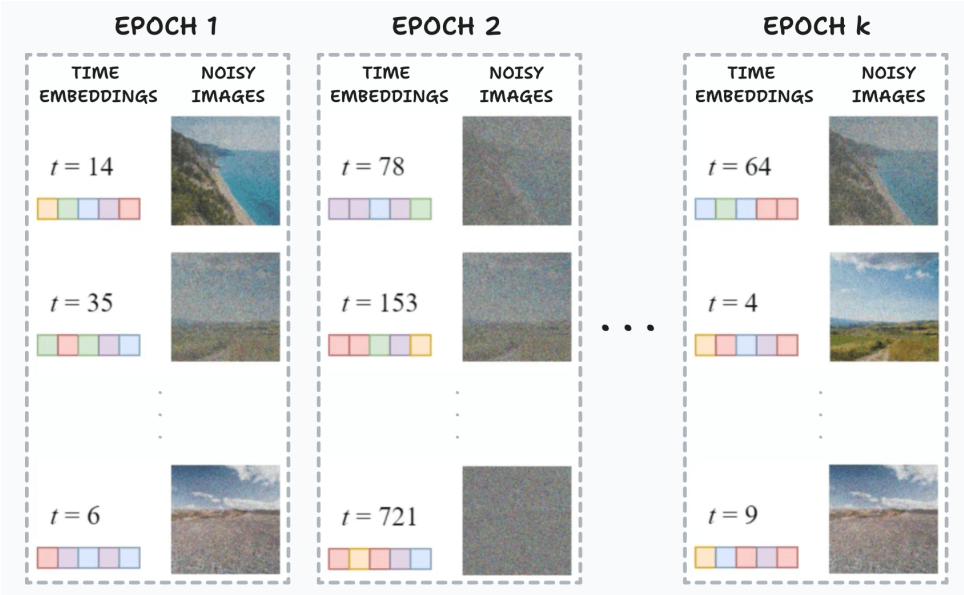

Figure 11: Representation of the corrupted images during training

8.2 Sampling Procedure

Once the network is trained, generate samples as follows:

1. Sample $x_T \sim \mathcal{N}(0, I)$, a standard Gaussian in the data dimension.
2. For $t = T, T-1, \ldots, 1$:
   (a) Compute the predicted noise $\epsilon_\theta(x_t, t)$ via the network
   (b) Form the model mean

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(x_t, t)\Big)$$

   Recall that $\alpha_t = 1 - \beta_t$
   (c) Compute $\tilde{\beta}_t$ via

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

(d) Sample the previous timestep

$$x_{t-1} = \mu_\theta(x_t, t) + \sqrt{\tilde{\beta}_t}\, z, \quad z \sim \mathcal{N}(0, I)$$

Optionally, one can omit the added noise $\sqrt{\tilde{\beta}_t}\, z$ for a deterministic "mean" sampling.

3. After iterating down to $t = 1$, output x_0 as the final generated data sample.

Notes:

- Including the $\sqrt{\tilde{\beta}_t}\, z$ term ensures randomness in sampling and hence diversity of outputs.
- If one omits the noise term (i.e., set $z = 0$), the sampling becomes deterministic and yields a point estimate of the "mode" of the posterior.

Figure 12: Denoising process for a single image

9 Conclusion and References

We have provided a detailed, beginner-friendly exposition of diffusion models, with explicit explanations accompanying each mathematical formula. In summary:

- **Forward Process:** We gradually add Gaussian noise via

$$q(x_t \mid x_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I\big) \quad \Longrightarrow \quad q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t} x_0,\, (1-\bar{\alpha}_t) I\big)$$

- **Posterior:** The conditional distribution of x_{t-1} given (x_t, x_0) is

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\big(\tilde{\mu}_t(x_t, x_0),\, \tilde{\beta}_t I\big), \quad \tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\big(x_t - \sqrt{1-\alpha_t}\,\epsilon_t\big)$$

- **Reverse Model:** We model

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(\mu_\theta(x_t, t),\, \Sigma_\theta(x_t, t)\big), \quad \mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\big(x_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(x_t, t)\big)$$

- **ELBO:** Decomposed into

$$-\,\mathrm{ELBO} = L_T + \sum_{t=1}^{T-1} L_t + L_0, \text{ where}$$

$$L_t = \mathbb{E}_{q(x_t|x_0)}\big[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))\big], \quad L_T = D_{\mathrm{KL}}(q(x_T \mid x_0) \,\|\, p(x_T))$$

- **Noise-Prediction Loss:** Leads to the simple training objective

$$L_{\text{total}} = \sum_{t=1}^{T} \mathbb{E}_{q(x_t, x_0)}\big[\|\epsilon_\theta(x_t, t) - \epsilon_t\|^2\big]$$

- **Beta Scheduling:** Two common choices are

$$\beta_t = \beta_{\min} + \frac{t-1}{T-1}(\beta_{\max} - \beta_{\min}) \quad \text{(linear), or}$$

$$\bar{\alpha}_t = \frac{\cos\big(\frac{t/T+s}{1+s}\,\frac{\pi}{2}\big)^2}{\cos\big(\frac{s}{1+s}\,\frac{\pi}{2}\big)^2}, \quad \beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}} \quad \text{(cosine)}$$

References

[1] Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models*. NeurIPS.

[2] Nichol, A., & Dhariwal, P. (2021). *Improved Denoising Diffusion Probabilistic Models*. ICML.

[3] Kingma, D. P., & Welling, M. (2014). *Auto-Encoding Variational Bayes*. ICLR.

[4] Song, Y., & Ermon, S. (2019). *Generative Modeling by Estimating Gradients of the Data Distribution*. NeurIPS.

[5] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2021). *Score-Based Generative Modeling through Stochastic Differential Equations*. ICLR.

[6] Durkan, C., Song, Y., Murray, I., & Ermon, S. (2021). *Maximum Likelihood Training of Score-Based Diffusion Models*. ICLR.

[7] Song, Y., & Ermon, S. (2020). *Improved Techniques for Training Score-Based Generative Models*. NeurIPS.