Chapter 2: VAEs and Diffusion Models

This chapter introduces two foundational probabilistic viewpoints that will recur throughout the course: *latent-variable modeling* (via variational autoencoders) and *iterative denoising* (via diffusion models).

- We begin with **autoencoders** as a representation learning tool: an encoder–decoder pair learns a low-dimensional code that preserves the salient structure of high-dimensional data. This explains their success in compression and feature learning, but also highlights why deterministic autoencoders are not generative models: they do not define a principled probability distribution from which we can sample.
- We then introduce **variational autoencoders (VAEs)** by specifying an explicit probabilistic generative model $z \sim p(z)$, $x \sim p_\theta(x \mid z)$ and training it by maximizing the **evidence lower bound (ELBO)**. The ELBO couples a reconstruction term (data fit) with a KL regularization term (matching the latent posterior to a simple prior), yielding a model that can both reconstruct and generate by sampling $z \sim p(z)$.
- Next, we show that **diffusion models can be viewed as multi-layered VAEs**: instead of a single latent variable z, we introduce a *sequence* of latent variables $(x_1, \ldots, x_T)$ produced by a fixed forward noising Markov chain. Learning consists of fitting a reverse-time Markov chain that inverts this noising process. This perspective yields a variational bound analogous to the VAE ELBO, but decomposed across many denoising steps.
- Finally, we describe two core procedures: **training** by noise prediction (the standard DDPM objective) and **sampling** by iterative ancestral denoising starting from Gaussian noise. This chapter stays in discrete time and postpones the continuous-time/SDE viewpoint to later notes.

There are good reading materials similar or highly relevant to the content summarized here, including tutorials [1] and [5], and a recent monograph [3].

1 A Probabilistic View of Variational Autoencoders

With the goal to provide a probabilistic view of Variational AutoEncoders (VAE), we start with a deterministic version and then extend to the stochastic counterpart.

1.1 Deterministic Autoencoders

An autoencoder is a learned compression—decompression pipeline. Given data $x \in \mathcal{X} \subseteq \mathbb{R}^d$, an **encoder** maps x to a low-dimensional representation $z \in \mathbb{R}^k$ (with $k \ll d$), and a **decoder** maps z back to a reconstruction $\widehat{x}$:

$$\begin{aligned} \text{Encoder:} \quad & z = E(x) \in \mathbb{R}^k, \\ \text{Decoder:} \quad & \widehat{x} = D(z) \in \mathbb{R}^d. \end{aligned}$$

The low-dimensional representation is also known as the *bottleneck* of autoencoders. When k is small, the bottleneck forces the network to learn a *compressed representation* capturing salient features of the data. In the linear case (linear encoder/decoder, squared loss), the autoencoder recovers PCA: the learned subspace corresponds to the top-k principal components.

More broadly, functions E and D are trained to minimize a *reconstruction loss*, e.g.,

$$(\widehat{E}, \widehat{D}) \in \operatorname*{argmin}_{E,D} \mathbb{E}_{x \sim P_{\text{data}}} \Big[\ell\big(x, D(E(x))\big) \Big] \quad \text{with} \quad \ell(x, \widehat{x}) = \|x - \widehat{x}\|_2^2 \text{ or other losses.}$$

Since encoder and decoder jointly reconstruct data, the solution to them is not unique.

Autoencoders are not (yet) good generative models. A deterministic autoencoder does not define a probability distribution over x—they are trained to reconstruct training samples. Even if reconstructions are good, there is no canonical way to sample new x:

- The encoder $\widehat{E}$ induces a complicated empirical distribution of codes z.
- Picking an ad-hoc distribution for z (e.g., standard Gaussian) and decoding it typically yields poor samples, because the decoder was never trained to match that latent representation.

This motivates a *stochastic* autoencoder that (i) specifies a generative probabilistic model on data and (ii) enables principled sampling.

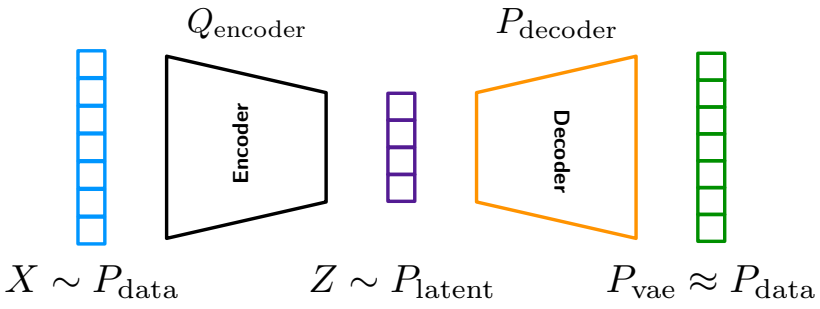
1.2 Stochastic Variational AutoEncoders (VAEs)

A VAE turns the autoencoder pipeline into a *probabilistic latent-variable model*. Instead of a deterministic bottleneck $z = E(x)$, we introduce a latent random variable $Z \in \mathbb{R}^k$ and posit that data are generated by

$$Z \sim P_{\text{latent}}(Z), \qquad X \mid Z \sim P_{\text{decoder}}(X \mid Z),$$

where the latent distribution P_{latent} is an easy-to-sample *prior* (typically standard Gaussian) and $P_{\text{decoder}}(X \mid Z)$ is a *decoder* (stochastic). These distributions are often chosen to have density functions p_{latent} and p_{decoder}.

As an intermediate result, the decoder also defines a joint distribution between latent Z and data X, denoted as $P_{\text{vae}}(x, z)$. From the joint distribution, we can also derive the conditional distribution $P_{\text{vae}}(z|x)$, which transforms data into latent representations. This behaves as an encoder.



Why a probabilistic model helps sampling. VAE introduces some hierarchical-like architecture to aid sampling. We can generate new samples by the first sample Z from P_{latent} and then sample X conditioned on Z from P_{decoder}. This is exactly what deterministic autoencoders lack: a principled distribution over z and a probabilistic decoding rule. However, there is still a caveat for sampling following this procedure: it might be expensive to sample from P_{decoder}. In later sections, we will restrict ourselves to a family of stochastic decoders that allow efficient sampling and tractable training.

1.2.1 Training VAEs

The latent distribution and the stochastic decoder defines a marginal distribution P_{vae} of data X, whose density is

$$p_{\text{vae}}(x) = \int p_{\text{decoder}}(x|z)\ p_{\text{latent}}(z)\mathrm{d}z.$$

In order to train the decoder—the latent distribution is typically chosen before training—we associate a loss function to $p_{\text{vae}}(x)$. A natural training objective is to minimize the KL divergence between P_{data} and P_{vae}:

$$\min_{p_{\text{decoder}}} \mathrm{KL}(P_{\text{data}}, P_{\text{vae}}).$$

Suppose the data distribution has a density p_{data}, we evaluate the KL divergence as

$$\mathrm{KL}(P_{\text{data}}, P_{\text{vae}}) = \mathbb{E}_{X\sim P_{\text{data}}}\left[\log p_{\text{data}}(X) - \log p_{\text{vae}}(X)\right].$$

Since p_{data} is a fixed quantity, we only need to optimize over

$$\min_{p_{\text{decoder}}} \mathbb{E}_{X\sim P_{\text{data}}}[-\log p_{\text{vae}}(X)] = \max_{p_{\text{decoder}}} \mathbb{E}_{X\sim P_{\text{data}}}[\log p_{\text{vae}}(X)], \qquad \text{(Log-likelihood training)}$$

which recovers the maximal log-likelihood training.

Although we can replace the unknown data distribution by its empirical counterpart, the real obstacle is that the integral defining $p_{\text{vae}}(x)$ is generally **intractable** in high dimensions.

Evidence lower bound VAEs address the issue above by introducing an auxiliary distribution $Q_{\text{encoder}}(z \mid x)$—a stochastic *encoder*. The key tool is to avoid the evaluation of the marginal density p_{vae} but to introduce separate losses on encoder and decoder.

Definition 1.1 (Evidence Lower BOund (ELBO))**.** The ELBO is defined as

$$\mathrm{ELBO}_{\text{vae}}(x) = \underbrace{\mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}[\log p_{\text{decoder}}(x|Z)]}_{\text{Decoder reconstruction term}} - \underbrace{\mathrm{KL}(Q_{\text{encoder}}(\cdot|x), P_{\text{latent}}(\cdot))}_{\text{Encoder latent term}}. \qquad \text{(ELBO)}$$

The (ELBO) has two terms:

- **Reconstruction term:** $\mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}[\log p_{\text{decoder}}(x|Z)]$ encourages the decoder to assign high likelihood to the observed x when latent variable Z is sampled from the encoder.
- **Latent term:** $\mathrm{KL}(Q_{\text{encoder}}(\cdot|x), P_{\text{latent}}(\cdot))$ encourages the latent distribution produced by the encoder to match the set prior P_{latent}, which is crucial for sampling: if Q_{encoder} drifts far from P_{latent}, then sampling $Z \sim P_{\text{latent}}$ and decoding can produce unrealistic outputs.

As can be seen, ELBO does not involve any marginal distribution P_{vae}, but only encoder and decoder. The following result reveals a fundamental relationship between (ELBO) and the log-likelihood function.

Proposition 1.2 (Decomposition of log-likelihood)**.** The log-likelihood $\log p_{\text{vae}}(x)$ decomposes into

$$\log p_{\text{vae}}(x) = \text{ELBO}_{\text{vae}}(x) + \text{KL}(Q_{\text{encoder}}(\cdot|x), P_{\text{vae}}(\cdot|x)).$$

Since KL divergence is always nonnegative, we immediately have

$$\log p_{\text{vae}}(x) \geq \text{ELBO}_{\text{vae}}(x),$$

validating ELBO as a lower bound. Moreover, the additional KL term is the discrepancy between the encoder and the conditional distribution defined by the VAE. If the encoder is perfect (i.e., $Q_{\text{encoder}}(\cdot|x) = P_{\text{vae}}(\cdot|x)$), then ELBO equals the log-likelihood.

Proof. We start from $\log p_{\text{vae}}(x)$:

$$\begin{aligned}
&\log p_{\text{vae}}(x) \\
&= \log p_{\text{vae}}(x) \cdot \underbrace{\int q_{\text{encoder}}(z|x)\mathrm{d}z}_{=1} \\
&= \int \log p_{\text{vae}}(x) \cdot q_{\text{encoder}}(z|x)\mathrm{d}z \\
&= \int \log \frac{p_{\text{decoder}}(x|z)p_{\text{latent}}(z)}{p_{\text{vae}}(z|x)} \cdot q_{\text{encoder}}(z|x)\mathrm{d}z && \text{(Bayes rule)} \\
&= \int \log p_{\text{decoder}}(x|z) \cdot q_{\text{encoder}}(z|x)\mathrm{d}z + \int \log \frac{p_{\text{latent}}(z)}{p_{\text{vae}}(z|x)} \cdot q_{\text{encoder}}(z|x)\mathrm{d}z \\
&= \underbrace{\int \log p_{\text{decoder}}(x|z) \cdot q_{\text{encoder}}(z|x)\mathrm{d}z}_{\mathcal{T}_1} + \underbrace{\int \log \left(\frac{p_{\text{latent}}(z)}{q_{\text{encoder}}(z|x)} \cdot \frac{q_{\text{encoder}}(z|x)}{p_{\text{vae}}(z|x)} \right) \cdot q_{\text{encoder}}(z|x)\mathrm{d}z}_{\mathcal{T}_2}.
\end{aligned}$$

Term $\mathcal{T}_1$ is precisely the first expectation term in (ELBO). For term $\mathcal{T}_2$, we have

$$\begin{aligned}
\mathcal{T}_2 &= \int \log \frac{p_{\text{latent}}(z)}{q_{\text{encoder}}(z|x)} \cdot q_{\text{encoder}}(z|x)\mathrm{d}z + \int \log \frac{q_{\text{encoder}}(z|x)}{p_{\text{vae}}(z|x)} \cdot q_{\text{encoder}}(z|x)\mathrm{d}z \\
&= -\int \log \frac{q_{\text{encoder}}(z|x)}{p_{\text{latent}}(z)} \cdot q_{\text{encoder}}(z|x)\mathrm{d}z + \int \log \frac{q_{\text{encoder}}(z|x)}{p_{\text{vae}}(z|x)} \cdot q_{\text{encoder}}(z|x)\mathrm{d}z \\
&= -\text{KL}(Q_{\text{encoder}}(\cdot|x), P_{\text{latent}}(\cdot)) + \text{KL}(Q_{\text{encoder}}(\cdot|x), P_{\text{vae}}(\cdot|x)).
\end{aligned}$$

Combining $\mathcal{T}_1$ and $\mathcal{T}_2$ together, we deduce the assertion. □

Remark 1.3 (Equivalent form of ELBO)**.** There is also a compact representation for ELBO:

$$\text{ELBO}_{\text{vae}}(x) = \mathbb{E}_{Z \sim Q_{\text{encoder}}(\cdot|x)} \left[\log \frac{p_{\text{vae}}(x, Z)}{q_{\text{encoder}}(Z|x)} \right].$$

To see this, we utilize the derivation in Proposition 1.2.

$$\begin{aligned}
\mathrm{ELBO}_{\mathrm{vae}}(x) &= \mathbb{E}_{Z\sim Q_{\mathrm{encoder}}(\cdot|x)}[\log p_{\mathrm{decoder}}(x|Z)] - \mathrm{KL}(Q_{\mathrm{encoder}}(\cdot|x), P_{\mathrm{latent}}(\cdot)) \\
&= \int \log p_{\mathrm{decoder}}(x|z)\cdot q_{\mathrm{encoder}}(z|x)\mathrm{d}z - \int \log \frac{q_{\mathrm{encoder}}(z|x)}{p_{\mathrm{latent}}(z)}\cdot q_{\mathrm{encoder}}(z|x)\mathrm{d}z \\
&= \int \log \frac{p_{\mathrm{decoder}}(x|z)p_{\mathrm{latent}}(z)}{q_{\mathrm{encoder}}(z|x)}\cdot q_{\mathrm{encoder}}(z|x)\mathrm{d}z \\
&= \int \log \frac{p_{\mathrm{vae}}(x,z)}{q_{\mathrm{encoder}}(z|x)}\cdot q_{\mathrm{encoder}}(z|x)\mathrm{d}z \\
&= \mathbb{E}_{Z\sim Q_{\mathrm{encoder}}(\cdot|x)}\left[\log \frac{p_{\mathrm{vae}}(x,Z)}{q_{\mathrm{encoder}}(Z|x)}\right].
\end{aligned}$$

Remark 1.4 (Variational inference)**.** ELBO is the standard training objective used in practice. It makes explicit the two competing forces in VAE training: (i) the decoder term wants to reconstruct x accurately from sampled latents, while (ii) the KL term wants the latent representations to remain close to a simple prior so that sampling is stable. This tension is exactly where the "variational" nature of VAEs enters, and it will reappear in diffusion models when we interpret their training as matching many reverse-conditionals across time.

To this end, the training of VAE is to maximize the more "tractable" ELBO loss:

$$\max_{p_{\mathrm{decoder}},\ q_{\mathrm{encoder}}} \mathbb{E}_{X\sim P_{\mathrm{data}}}[\mathrm{ELBO}_{\mathrm{vae}}(X)]. \qquad \text{(ELBO training)}$$

1.2.2 Encoder and Decoder Parameterizations

Optimizing (ELBO training) requires searching over conditional distributions $Q_{\mathrm{encoder}}(\cdot \mid x)$ and $P_{\mathrm{decoder}}(\cdot \mid z)$. This is an infinite-dimensional optimization problem and is therefore intractable without further structure. VAEs make the optimization computationally feasible by *parameterizing* the encoder and decoder with finite-dimensional parameters, most commonly via neural networks.

Gaussian latent and Gaussian encoder The choice of latent distribution and encoder centers around simplifying the encoder latent term in ELBO. A standard choice on the latent distribution is the standard Gaussian

$$P_{\mathrm{latent}} = \mathsf{N}(0, I).$$

We will also choose the encoder being Gaussian with trainable mean and covariance:

$$Q_{\mathrm{encoder}}(\cdot \mid x) = \mathsf{N}\big(\mu_\phi(x),\ \mathrm{diag}(\sigma_\phi^2(x))\big), \tag{1.1}$$

where $\mu_\phi(x) \in \mathbb{R}^k$ and $\sigma_\phi(x) \in \mathbb{R}^k_+$ are outputs of an encoder network, and ϕ denotes the trainable parameters. Although this pair of latent and encoder choice is not mandated by the theory, it has practical advantages:

- **Tractability.** The KL divergence between Gaussian distributions admits a closed-form expression (derived below in Proposition 1.5).

- **Scalability.** We additionally specify Q_{encoder} to have a diagonal covariance matrix (can be simplified to isotropic diagonal covariance, i.e., each entry in $\sigma_\phi(x)$ is identical). A full covariance matrix would require $\mathcal{O}(k^2)$ parameters per input x and is expensive to store and manipulate.

The following proposition is useful to simplify the encoder latent term.

Proposition 1.5 (Closed-form KL for Gaussian)**.** Let distributions $P = \mathsf{N}(\mu_p, \Sigma_p)$ and $Q = \mathsf{N}(\mu_q, \Sigma_q)$ in $\mathbb{R}^k$. Then the KL divergence assumes a closed-form expression:

$$\mathrm{KL}(P, Q) = \frac{1}{2}\left(\mathrm{Trace}(\Sigma_q^{-1}\Sigma_p) - k + (\mu_q - \mu_p)^\top \Sigma_q^{-1}(\mu_q - \mu_p) + \log\frac{|\Sigma_q|}{|\Sigma_p|}\right),$$

where $|\cdot|$ denotes the matrix determinant.

Proof. This is left as an exercise. □

Now let's apply Proposition 1.5 to our choice of the latent distribution and encoder; we have

$$\begin{aligned}
-\mathrm{KL}(Q_{\text{encoder}}(\cdot|x), P_{\text{latent}}(\cdot)) &= -\mathrm{KL}(\mathsf{N}(\mu_\phi(x), \mathrm{diag}(\sigma_\phi^2(x))), \mathsf{N}(0, I)) \\
&= -\frac{1}{2}\left(\mathrm{Trace}(\mathrm{diag}(\sigma_\phi^2(x))) - k + \|\mu_\phi(x)\|_2^2 - \log|\mathrm{diag}(\sigma_\phi^2(x))|\right) \\
&= \frac{1}{2}\left(-\mathbf{1}^\top\sigma_\phi^2(x) - \|\mu_\phi(x)\|_2^2 + \sum_{i=1}^k \log[\sigma_\phi^2(x)]_i + k\right), \qquad (1.2)
\end{aligned}$$

where $\mathbf{1}$ denotes a vector of one and $[\cdot]_i$ denotes the i-th entry in a vector. An immediate observation is that we can now easily find the derivative with respect to the trainable parameter ϕ. This is a good sign, since the evaluation of derivative allows application of popular first-order optimization algorithms, such as stochastic gradient descent.

Remark 1.6. It may look worrying that a Gaussian encoder can fail to transform an arbitrarily complex data distribution P_{data} into the latent distribution P_{latent}. However, there is a trivial choice of μ_ϕ and σ_ϕ^2 always transforming data into the standard Gaussian distribution:

$$\mu_\phi(x) = 0 \quad \text{and} \quad \sigma_\phi^2(x) = \mathbf{1}.$$

Yet, such a transformation only matches the marginal distribution of the latent variable Z, without following the joint distribution of data and latent variable. This spells out a limitation of VAEs, owing to the representation capability of the encoder.

Gaussian Decoder We now turn to a common parameterization for the decoder that simplifies the decoder reconstruction term. In particular, we choose

$$P_{\text{decoder}}(x \mid z) = \mathsf{N}\big(\mu_\theta(z), \sigma^2 I\big),$$

where $\mu_\theta(z) : \mathbb{R}^k \mapsto \mathbb{R}^d$ is the output of a decoder network and $\sigma^2 > 0$ is either fixed or learned. Then we have

$$\begin{aligned}
\log p_{\text{decoder}}(x \mid z) &= \log\frac{1}{(\sqrt{2\pi}\sigma)^d}\exp\left(-\frac{1}{2\sigma^2}\|x - \mu_\theta(z)\|_2^2\right) \\
&= -d\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\|x - \mu_\theta(z)\|_2^2.
\end{aligned}$$

Plugging in the log density of p_{decoder} into the decoder reconstruction term, we derive

$$\begin{aligned}\mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}[\log p_{\text{decoder}}(x|Z)] &= \mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}\left[-d\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}\|x-\mu_\theta(Z)\|_2^2\right] \\ &= -d\log(\sqrt{2\pi}\sigma) - \mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}\left[\frac{1}{2\sigma^2}\|x-\mu_\theta(Z)\|_2^2\right]. \end{aligned} \tag{1.3}$$

We make two observations: 1) Under the Gaussian encoder and decoder parameterization, the decoder reconstruction term only involves an expectation over Gaussian distributions; 2) When optimizing over parameter θ, we can neglect the constant term $-d\log(\sqrt{2\pi}\sigma)$, further simplifying the training objective function.

Remark 1.7 (Expressive decoder)**.** Similar to the concern for encoders, it may be worrying that the Gaussian parameterization of decoder is not expressive enough to represent the original data distribution, that is, P_{vae}—constructed from P_{latent} and P_{decoder} may fail to represent some complicated distributions. This is typically less of a concern. The distribution P_{vae} can be viewed as a Gaussian mixture model, where P_{latent} is the prior and $P_{\text{decoder}}(\cdot|z)$ represents a component in the Gaussian mixture. Since Gaussian mixture is a universal density approximator—can represent any distributions as long as the number of components is sufficient and each component is appropriately specified; see for example [6] and the references therein.

Remark 1.8 (Other parameterization)**.** Depending on the domain, one can choose other decoder families (categorical likelihoods for discrete tokens, mixture likelihoods, etc.). One concrete example is if $x \in \{0,1\}^d$ is discrete, we often use a coordinate-wise Bernoulli decoder:

$$p_{\text{decoder}}(x \mid z) = \prod_{i=1}^{d} \text{Ber}\big([x]_i; [\pi_\theta(z)]_i\big) \quad \text{with} \quad \pi_\theta(z) \in [0,1]^d,$$

where $[\cdot]_i$ denotes the i-th entry of a vector. The log density of the decoder becomes

$$\log p_{\text{decoder}}(x \mid z) = \sum_{i=1}^{d} \Big([x]_i \log[\pi_\theta(z)]_i + (1-[x]_i)\log(1-[\pi_\theta(z)]_i)\Big),$$

which renders the reconstruction error term a cross-entropy style loss.

1.2.3 Gradient Evaluation for ELBO

We have simplified the ELBO loss by parameterizing the encoder, decoder, and latent distribution. Under Gaussian parameterization, combining (1.2) and (1.3) gives rise to

$$\begin{aligned}\text{ELBO}_{\text{vae}}(x) = -d\log(\sqrt{2\pi}\sigma) - \mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}\left[\frac{1}{2\sigma^2}\|x-\mu_\theta(Z)\|_2^2\right] \\ + \frac{1}{2}\left(-\mathbf{1}^\top\sigma_\phi^2(x) - \|\mu_\phi(x)\|_2^2 + \sum_{i=1}^{k}\log[\sigma_\phi^2(x)]_i + k\right).\end{aligned} \tag{1.4}$$

Further, (ELBO training) is equivalent to

$$\max_{\phi,\theta}\ \mathbb{E}_{X\sim P_{\text{data}}}[\text{ELBO}_{\text{vae}}(X)]$$

due to the parameterization. The term $-d\log(\sqrt{2\pi}\sigma)$ in (1.4) does not affect the optimization of ϕ, θ, and therefore, can be dropped from the objective function.

Yet, a closer inspection of the ELBO loss identifies a remaining issue: ELBO contains an expectation over a distribution that depends on encoder parameter ϕ (the orange term in (1.4)). Naïvely differentiating this term with respect to ϕ is difficult because the sampling of Z depends on ϕ. For Gaussian encoders, VAEs use a *reparameterization trick* to rewrite the sampling of $Z \sim Q_{\text{encoder}}$.

Recall that the Gaussian encoder assumes

$$Z \mid x \sim \mathsf{N}\big(\mu_\phi(x), \operatorname{diag}(\sigma_\phi^2(x))\big).$$

Let $\epsilon \sim \mathsf{N}(0, I)$. Then we can sample Z via

$$Z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \tag{1.5}$$

where $\sigma_\phi(x) \in \mathbb{R}_+^k$ is the vector of standard deviations and $\odot$ is entrywise product. The reparameterization trick is to replace the sampling of Z by ϵ and plugging in (1.5) whenever necessary.

Proposition 1.9 (Reparameterization for Gaussian encoder)**.** Using the reparameterization in the previous context, we have

$$\mathbb{E}_{Z\sim Q_{\text{encoder}}(\cdot|x)}\big[\log p_{\text{decoder}}(x \mid Z)\big] = \mathbb{E}_{\epsilon\sim\mathsf{N}(0,I)}\Big[\log p_{\text{decoder}}\big(x \mid \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon\big)\Big].$$

Proposition 1.9 isolates the randomness in ϵ with trainable parameter ϕ. Moreover, the mapping $\epsilon \mapsto Z$ is differentiable in ϕ. Therefore, we can compute gradients of ELBO using standard backpropagation. This is a key reason why Gaussian VAEs are trainable at scale.

Combining the pieces above (Gaussian latent, Gaussian encoder and decoder, and reparameterization trick), ELBO becomes

$$\begin{aligned}\text{ELBO}_{\text{vae}}(x) = -d\log(\sqrt{2\pi}\sigma) - \mathbb{E}_{\epsilon\sim\mathsf{N}(0,I)}\left[\frac{1}{2\sigma^2}\|x - \mu_\theta\big(\mu_\phi(x) + \sigma_\phi(x)\odot\epsilon\big)\|_2^2\right] \\ + \frac{1}{2}\left(-\mathbf{1}^\top\sigma_\phi^2(x) - \|\mu_\phi(x)\|_2^2 + \sum_{i=1}^k \log[\sigma_\phi^2(x)]_i + k\right),\end{aligned}$$

where the expectation over ϵ is approximated by Monte Carlo.

1.2.4 Limitations of VAEs

While VAEs provide a clean probabilistic framework and a principled sampling mechanism, they also exhibit several well-known limitations in practice. First, the training objective is not the true log-likelihood but a *lower bound* (ELBO). If the encoder family is not expressive enough, the gap $\mathrm{KL}\big(Q_{\text{encoder}}(\cdot \mid x), P_{\text{vae}}(\cdot \mid x)\big)$ can be large, meaning that optimizing ELBO may not be effective in maximizing the log-likelihood. Second, common decoder likelihoods such as a Gaussian with fixed variance can impose restrictive output noise models; this often leads to over-smoothed reconstructions and samples when the data distribution is sharp or multi-modal. Third, the encoder latent term in ELBO can sometimes dominate and push $Q_{\text{encoder}}(\cdot \mid x)$ close to P_{latent} for many inputs, leading to a phenomenon often referred to as *posterior collapse* [4, 2].

These limitations motivate an alternative design principle: instead of generating X from a *single* latent variable Z in one step, we generate X through a *sequence* of intermediate latent variables that gradually refine noise into structure. Diffusion models implement exactly this idea by introducing a multi-step latent chain $(X_1, \ldots, X_T)$ and learning a stack of conditional decoders.

2 Diffusion Models as Multi-layered Variational Autoencoders

In this section we present diffusion models as a *multi-layer latent-variable model*, parallel to VAEs but with a *sequence* of latent variables. The logical flow mirrors the VAE development:

1. specify a generative model with many latent variables and derive an ELBO;
2. decompose the ELBO into interpretable KL terms across layers;
3. choose Gaussian parameterizations to make the ELBO tractable and differentiable;
4. obtain the standard Diffusion Denoising Probabilistic Model (DDPM) training objective as a specialization of the ELBO.

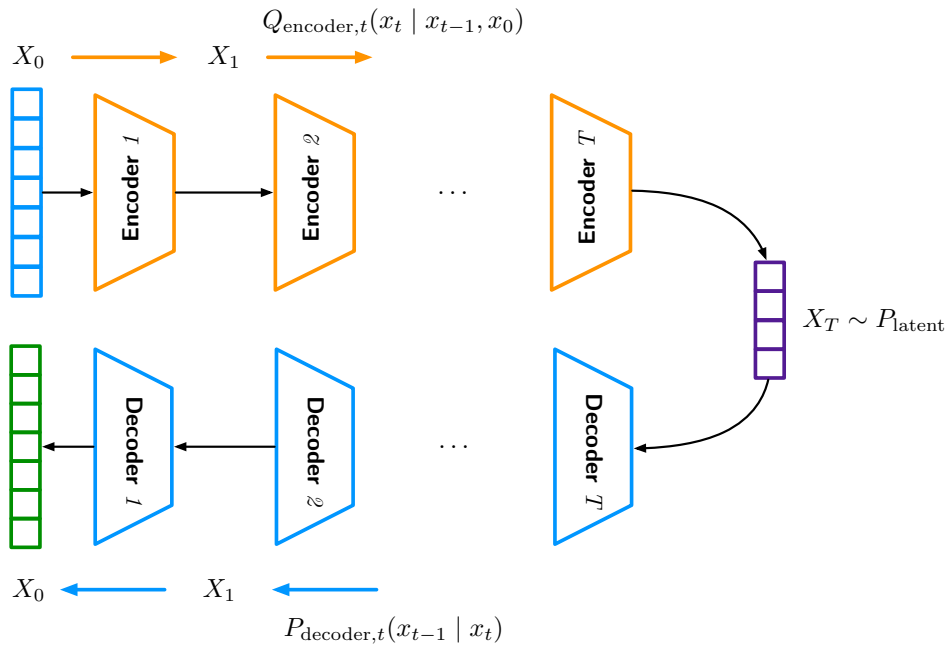


Figure 1: A conceptual illustration of diffusion models with multiple layers of encoders and decoders.

2.1 ELBO for Diffusion Models

We denote a chain of latent variables as $X_1, \ldots, X_T$—*all of the same dimension*—and let the terminal latent variable $X_T \sim P_{\text{latent}}$ following an easy-to-sample latent distribution such as Gaussian. We define a sequence of Markov decoders

$$P_{\text{decoder},t}(x_{t-1}|x_t) \quad \text{for} \quad t = 1, \ldots, T,$$

where X_0 is the generated variable, aiming to replicate the underlying data distribution. A diffusion model defines a generative distribution of the form

$$p_{\text{diff}}(x_{0:T}) = p_{\text{latent}}(x_T) \prod_{t=1}^{T} p_{\text{decoder},t}(x_{t-1} \mid x_t). \tag{2.1}$$

where the subscript diff stands for diffusion and $x_{0:T}$ is a short-hand for the sequence $x_0, \ldots, x_T$. By marginalizing over $x_{1:T}$, the generated data distribution has a density:

$$p_{\text{diff}}(x_0) = \int p_{\text{diff}}(x_{0:T}) \mathrm{d}x_{1:T}. \tag{2.2}$$

A variational "multi-layer encoder". As in VAEs, even though we defined $p_{\text{diff}}(x_0)$, directly maximizing the log-likelihood $\log p_{\text{diff}}(x_0)$ is difficult due to the intractable integral (2.2). We introduce an auxiliary conditional distribution $Q_{\text{encoder}}(x_{1:T}|x_0)$, which will serve as the *multi-layer encoder*:

$$q_{\text{encoder}}(x_{1:T} \mid x_0) = q_{\text{encoder},1}(x_1 \mid x_0) \prod_{t=2}^{T} q_{\text{encoder},t}(x_t \mid x_{t-1}, x_0). \tag{2.3}$$

Namely, the encoder chain starts from X_0 and progressively produces noisier latents until reach X_T. Importantly, in diffusion models this encoder is typically *fixed* (not learned), and learning focuses on the decoder $P_{\text{decoder},t}$. Moreover, the encoder includes the original starting point x_0 as conditioning, potentially rendering non-Markov transitions such as those in Denoising Diffusion Implicit Models (DDIMs).

With encoders and decoders, we extend the ELBO loss to diffusion models in the following definition, which is motivated by the equivalent form of ELBO in Remark 1.3.

Definition 2.1 (Diffusion ELBO)**.** For any choice of $q(x_{1:T} \mid x_0)$, define

$$\text{ELBO}_{\text{diff}}(x_0) = \mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \Big[\log \frac{p_{\text{diff}}(x_0, X_{1:T})}{q_{\text{encoder}}(X_{1:T} \mid x_0)} \Big].$$

Note that the sequence of latents $X_{1:T}$ serves analogously to the single latent variable Z in VAEs. It is natural to show that ELBO lower bounds the log-likelihood in the next proposition.

Proposition 2.2 (ELBO lower bounds log-likelihood)**.** For every x_0, it holds that

$$\log p_{\text{diff}}(x_0) \geq \text{ELBO}_{\text{diff}}(x_0).$$

Moreover, we have

$$\log p_{\text{diff}}(x_0) = \text{ELBO}_{\text{diff}}(x_0) + \text{KL}\big(Q_{\text{encoder}}(\cdot \mid x_0), P_{\text{diff}}(\cdot \mid x_0)\big),$$

where $P_{\text{diff}}(x_{1:T} \mid x_0)$ is conditional distribution induced by $P_{\text{diff}}(x_{0:T})$.

Proof. The proof can replicate exactly the same steps in Proposition 1.2 with the replacement of Z by $X_{1:T}$, which is left as an exercise. Here, we provide a simple proof for the lower bound. We

have

$$\begin{aligned} p_{\text{diff}}(x_0) &= \int p_{\text{diff}}(x_{0:T})\mathrm{d}x_{1:T} \\ &= \int q_{\text{encoder}}(x_{1:T} \mid x_0) \cdot \frac{p_{\text{diff}}(x_{0:T})}{q_{\text{encoder}}(x_{1:T} \mid x_0)}\mathrm{d}x_{1:T} \\ &= \mathbb{E}_{X_{1:T}\sim Q_{\text{encoder}}(\cdot|x_0)}\left[\frac{p_{\text{diff}}(x_0, X_{1:T})}{q_{\text{encoder}}(X_{1:T} \mid x_0)}\right]. \end{aligned}$$

Applying Jensen's inequality, we deduce

$$\begin{aligned} \log p_{\text{diff}}(x_0) &= \log \mathbb{E}_{X_{1:T}\sim Q_{\text{encoder}}(\cdot|x_0)}\left[\frac{p_{\text{diff}}(x_0, X_{1:T})}{q_{\text{encoder}}(X_{1:T} \mid x_0)}\right] \\ &\geq \mathbb{E}_{X_{1:T}\sim Q_{\text{encoder}}(\cdot|x_0)}\left[\log \frac{p_{\text{diff}}(x_0, X_{1:T})}{q_{\text{encoder}}(X_{1:T} \mid x_0)}\right] \\ &= \text{ELBO}_{\text{diff}}(x_0). \end{aligned}$$

□

2.2 ELBO Decomposition

Although the ELBO in Definition 2.1 retains very similar properties to the counterpart for VAEs, we still lack an important interpretation (in fact, decomposition) akin to Definition 1.1. This is the precise sense in which diffusion models are *multi-layered VAEs*.

Proposition 2.3 (Decomposed diffusion ELBO)**.** The ELBO of diffusion models decomposes as

$$\begin{aligned} \text{ELBO}_{\text{diff}}(x_0) = &\underbrace{\mathbb{E}_{X_1\sim Q_{\text{encoder},1}(\cdot|x_0)}\left[\log p_{\text{decoder},1}(x_0 \mid X_1)\right]}_{\text{Data reconstruction term}} \\ &- \underbrace{\text{KL}\big(Q_{\text{encoder}}(X_T \mid x_0), P_{\text{latent}}(X_T)\big)}_{\text{Latent term}} \\ &- \sum_{t=2}^{T} \underbrace{\mathbb{E}_{X_t\sim Q_{\text{encoder}}(X_t|x_0)}\left[\text{KL}\big(Q_{\text{encoder},t}(\cdot \mid X_t, x_0), P_{\text{decoder},t}(\cdot \mid X_t)\big)\right]}_{\text{Intermediate consistency term}}, \end{aligned}$$

where $Q_{\text{encoder}}(x_t|x_0)$ and $Q_{\text{encoder},t}(x_{t-1}|x_t, x_0)$ are induced by encoder distributions.

Proposition 2.3 mirrors the VAE decomposition, but with a sequence of additional terms:

- The data reconstruction error term

  $$\mathbb{E}_{X_1\sim Q_{\text{encoder},1}(\cdot|x_0)}\left[\log p_{\text{decoder},1}(x_0 \mid X_1)\right]$$

  measures the quality of the final decoder for generating data.

- The latent term

  $$\text{KL}\big(Q_{\text{encoder}}(X_T \mid x_0), P_{\text{latent}}(X_T)\big)$$

  concerns whether the latent representation generated given data x_0 matches the desired latent distribution.

- The intermediate consistency term

$$\mathbb{E}_{X_t \sim Q_{\text{encoder}}(X_t|x_0)} \left[\mathrm{KL}\big(Q_{\text{encoder},t}(\cdot \mid X_t, x_0), P_{\text{decoder},t}(\cdot \mid X_t)\big) \right]$$

  is new to diffusion models due to the presence of the multi-layer encoder and decoder. The idea is to match the learned reverse conditional to the Bayes-reversed encoder conditional, averaged over forward marginals.

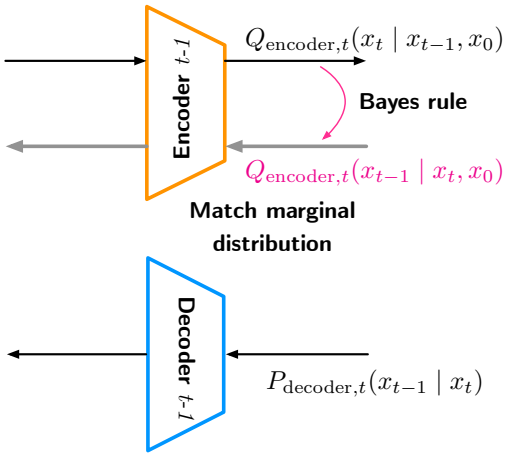


Figure 2: Consistency term at $t-1$-th encoder-decoder pair.

At this point, everything is still abstract: the terms are well-defined but not necessarily tractable. We will specialize to Gaussian parameterizations in Section 2.3 to obtain closed forms and differentiable objectives. We conclude this section by a proof to Proposition 2.3.

Proof. The proof repeatedly applies Bayes rule and basic algebraic manipulations. By the definition of ELBO for diffusion models, we have

$$\begin{aligned}
&\mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log \frac{p_{\text{diff}}(x_0, X_{1:T})}{q_{\text{encoder}}(X_{1:T} \mid x_0)} \right] \\
&= \mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log \frac{p_{\text{latent}}(X_T) \prod_{t=2}^{T} p_{\text{decoder},t}(X_{t-1} \mid X_t) p_{\text{decoder},1}(x_0 \mid X_1)}{q_{\text{encoder}}(X_{1:T} \mid x_0)} \right] \\
&= \mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log p_{\text{decoder},1}(x_0 \mid X_1) + \log \frac{p_{\text{latent}}(X_T) \prod_{t=2}^{T} p_{\text{decoder},t}(X_{t-1} \mid X_t)}{q_{\text{encoder}}(X_{1:T} \mid x_0)} \right].
\end{aligned}$$

We separate the data reconstruction error from the ELBO. The remaining step is to tackle the second term in the last display. We first recall Equation (2.3) that decomposes the encoder chain into

$$q_{\text{encoder}}(x_{1:T} \mid x_0) = q_{\text{encoder},1}(x_1 \mid x_0) \prod_{t=2}^{T} q_{\text{encoder},t}(x_t \mid x_{t-1}, x_0).$$

If we directly substituting the decomposition of q_{encoder} into ELBO, we encounter bi-directional evolving indexes: for the decoder, the evolution is from X_t to X_{t-1}, but for the encoder, the

evolution moves in the opposite direction. Therefore, we apply Bayes rule to the encoder and purposely flip the index in encoders. For any $t \geq 2$, we have

$$q_{\text{encoder},t}(x_t \mid x_{t-1}, x_0) = \frac{q_{\text{encoder},t}(x_{t-1} \mid x_t, x_0) q_{\text{encoder}}(x_t \mid x_0)}{q_{\text{encoder}}(x_{t-1} | x_0)}.$$

Taking product for $t = 2, \ldots, T$, we obtain

$$\begin{aligned}
q_{\text{encoder}}(x_{1:T} \mid x_0) &= q_{\text{encoder},1}(x_1 \mid x_0) \prod_{t=2}^{T} q_{\text{encoder},t}(x_t \mid x_{t-1}, x_0) \\
&= q_{\text{encoder},1}(x_1 \mid x_0) \prod_{t=2}^{T} \frac{q_{\text{encoder},t}(x_{t-1} \mid x_t, x_0) q_{\text{encoder}}(x_t \mid x_0)}{q_{\text{encoder}}(x_{t-1} \mid x_0)} \\
&= \prod_{t=2}^{T} q_{\text{encoder},t}(x_{t-1} \mid x_t, x_0) q_{\text{encoder}}(x_T \mid x_0),
\end{aligned}$$

where in the last equation, we have a massive cancellation on $q_{\text{encoder}}(x_t \mid x_0)$ terms for $t = 1, \ldots, T-1$. To this end, we proceed to simplify ELBO as

$$\begin{aligned}
&\mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log \frac{p_{\text{diff}}(x_0, X_{1:T})}{q_{\text{encoder}}(X_{1:T} \mid x_0)} \right] \\
&= \mathbb{E}_{X_1 \sim Q_{\text{encoder},1}(\cdot|x_0)} \left[\log p_{\text{decoder},1}(x_0 \mid X_1) \right] \\
&\quad + \mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log \frac{p_{\text{latent}}(X_T) \prod_{t=2}^{T} p_{\text{decoder},t}(X_{t-1} \mid X_t)}{q_{\text{encoder}}(X_T \mid x_0) \prod_{t=2}^{T} q_{\text{encoder},t}(X_{t-1} | X_t, x_0)} \right] \\
&= \mathbb{E}_{X_1 \sim Q_{\text{encoder},1}(\cdot|x_0)} \left[\log p_{\text{decoder},1}(x_0 \mid X_1) \right] \\
&\quad + \mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log \frac{p_{\text{latent}}(X_T)}{q_{\text{encoder}}(X_T \mid x_0)} \right] \\
&\quad + \mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\sum_{t=2}^{T} \log \frac{p_{\text{decoder},t}(X_{t-1} \mid X_t)}{q_{\text{encoder},t}(X_{t-1} | X_t, x_0)} \right].
\end{aligned}$$

For the brown term, we recognize

$$\mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\log \frac{p_{\text{latent}}(X_T)}{q_{\text{encoder}}(X_T \mid x_0)} \right] = -\mathrm{KL}\big(Q_{\text{encoder}}(X_T \mid x_0), P_{\text{latent}}(X_T)\big).$$

For the magenta term, using the tower property of expectation, we have

$$\begin{aligned}
&\mathbb{E}_{X_{1:T} \sim Q_{\text{encoder}}(\cdot|x_0)} \left[\sum_{t=2}^{T} \log \frac{p_{\text{decoder},t}(X_{t-1} \mid X_t)}{q_{\text{encoder},t}(X_{t-1} | X_t, x_0)} \right] \\
&= \mathbb{E}_{X_t \sim Q_{\text{encoder}}(\cdot|x_0)} \mathbb{E}_{X_{t-1} \sim Q_{\text{encoder},t}(\cdot|X_t, x_0)} \left[\log \frac{p_{\text{decoder},t}(X_{t-1} \mid X_t)}{q_{\text{encoder},t}(X_{t-1} \mid X_t, x_0)} \right] \\
&= -\mathbb{E}_{X_t \sim Q_{\text{encoder}}(\cdot|x_0)} \Big[\mathrm{KL}\big(Q_{\text{encoder},t}(\cdot \mid X_t, x_0), P_{\text{decoder},t}(\cdot \mid X_t)\big) \Big].
\end{aligned}$$

Summing over $t = 2, \ldots, T$ yields the claimed decomposition. □

2.3 Gaussian Parameterizations

We now make a specific choice of the encoder Q_{encoder} and decoder P_{decoder} so that (i) all terms in ELBO are tractable, and (ii) gradient evaluation with respect to trainable parameters can be computed efficiently. Throughout our discussion, we always set the latent distribution

$$P_{\text{latent}} = \mathsf{N}(0, I),$$

which is the most commonly used latent distribution in diffusion models.

2.3.1 Gaussian Encoders (Fixed) in Denoising Diffusion Probabilistic Models (DDPMs)

DDPM chooses the encoder chain to be Markov, and the encoders are pre-fixed. This is very different from VAEs, where the encoder involves learnable parameters. In particular, let $\{\alpha_t\}_{t=1}^T$ with $\alpha_t \in (0, 1)$ be a fixed sequence. The encoder is

$$Q_{\text{encoder},t}(x_t \mid x_{t-1}, x_0) = \mathsf{N}\big(\sqrt{\alpha_t} x_{t-1}, (1-\alpha_t) I\big), \tag{2.4}$$

where for $t = 1$, the encoder distribution is effectively $Q_{\text{encoder},1}(x_1|x_0) = \mathsf{N}(\sqrt{\alpha_1} x_0, (1-\alpha_1) I)$. Equivalently, the output of the t-th encoder can be written as

$$X_t = \sqrt{\alpha_t} X_{t-1} + \sqrt{1-\alpha_t}\ \epsilon_t \quad \text{for} \quad \epsilon_t \sim \mathsf{N}(0, I) \text{ independent.}$$

Roughly speaking, the chain of encoders gradually shrinks the magnitude of its original input (since $\alpha_t < 1$) and adds Gaussian noise to the shrunken input.

The scaling of $\sqrt{\alpha_t}$ is to ensure a stable covariance. Denote Σ_{t-1} as the covariance of X_{t-1}, then the covariance of X_t is $\alpha_t \Sigma_{t-1} + (1-\alpha_t) I$, interpolating between the input covariance and the isotropic identity covariance.

There are two important consequences of the encoder choice in Equation (2.4). First, we can find the marginal distribution of X_t given the original input x_0.

Lemma 2.4 (Closed form encoder marginal)**.** For each $t \geq 1$,

$$Q_{\text{encoder}}(x_t \mid x_0) = \mathsf{N}\big(\sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) I\big),$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$. Equivalently, X_t can be represented as

$$X_t = \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon \quad \text{with} \quad \epsilon \sim \mathsf{N}(0, I). \tag{2.5}$$

This result is convenient to find the closed form of the latent term.

Proof. We show by recursion. For any t, we have

$$\begin{aligned} X_t &= \sqrt{\alpha_t} X_{t-1} + \sqrt{1-\alpha_t}\ \epsilon_t \\ &= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}} X_{t-2} + \sqrt{1-\alpha_{t-1}}\ \epsilon_{t-1}) + \sqrt{1-\alpha_t}\ \epsilon_t \\ &= \sqrt{\alpha_t \alpha_{t-1}}\, X_{t-2} + \underbrace{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \epsilon_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t}_{Y}. \end{aligned}$$

Note that ϵ_{t-1} and ϵ_t are independent. We verify that Y is Gaussian $\mathsf{N}(0, (1-\alpha_t\alpha_{t-1})I)$. To see this, Y is clearly zero mean, and the covariance is given by

$$\mathrm{Cov}[Y] = \alpha_t(1-\alpha_{t-1})I + (1-\alpha_t)I = (1-\alpha_t\alpha_{t-1})I.$$

As a result, we obtain

$$X_t = \sqrt{\alpha_t\alpha_{t-1}}X_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\epsilon \quad \text{with} \quad \epsilon \sim \mathsf{N}(0, I).$$

Continuing the expansion until the original input X_0 is reached gives rise to the desired result. You may also prove the result by induction. □

Lemma 2.4 immediately simplifies the latent term in ELBO. However, since the encoders are fixed, the latent term does not depend on any trainable parameters. Therefore, we can neglect the latent term in ELBO when optimizing it.

The second result provides closed forms in intermediate consistency terms.

Lemma 2.5 (Closed form reverse encoder)**.** For any $t \geq 2$, it holds that

$$Q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0) = \mathsf{N}\big(\mu_{\mathrm{encoder},t}(x_t, x_0), \sigma^2_{\mathrm{encoder},t} I\big),$$

where

$$\mu_{\mathrm{encoder},t}(x_t, x_0) = \frac{\sqrt{\bar\alpha_{t-1}}(1-\alpha_t)}{1-\bar\alpha_t}\,x_0 + \frac{\sqrt{\alpha_t}(1-\bar\alpha_{t-1})}{1-\bar\alpha_t}\,x_t,$$
$$\sigma^2_{\mathrm{encoder},t} = \frac{(1-\alpha_t)(1-\bar\alpha_{t-1})}{1-\bar\alpha_t}.$$

Proof. The proof only requires elementary algebraic manipulation. By Bayes rule, we have

$$q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0) = \frac{q_{\mathrm{encoder},t}(x_t \mid x_{t-1}, x_0) q_{\mathrm{encoder}}(x_{t-1} \mid x_0)}{q_{\mathrm{encoder}}(x_t | x_0)}.$$

The density functions in the right-hand side are all Gaussian. In particular, we have

$$Q_{\mathrm{encoder},t}(x_t \mid x_{t-1}, x_0) = \mathsf{N}(\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I)$$
$$Q_{\mathrm{encoder}}(x_{t-1} \mid x_0) = \mathsf{N}(\sqrt{\bar\alpha_{t-1}}x_0, (1-\bar\alpha_{t-1})I)$$
$$Q_{\mathrm{encoder}}(x_t \mid x_0) = \mathsf{N}(\sqrt{\bar\alpha_t}x_0, (1-\bar\alpha_t)I).$$

Substituting the density functions into $q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0)$, we derive

$$q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0) \propto \exp\left(-\frac{1}{2}\left(\frac{\|x_t - \sqrt{\alpha_t}x_{t-1}\|_2^2}{1-\alpha_t} + \frac{\|x_{t-1} - \sqrt{\bar\alpha_{t-1}}x_0\|_2^2}{1-\bar\alpha_{t-1}} - \frac{\|x_t - \sqrt{\bar\alpha_t}x_0\|_2^2}{1-\bar\alpha_t}\right)\right).$$

This indicates that $Q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0)$ is Gaussian. The remaining step is to complete the squares and identify the mean and covariance. Indeed, by neglecting all the terms independent of x_{t-1}, we have

$$q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0)$$
$$\propto \exp\left(-\frac{1}{2}\left(\frac{1-\bar\alpha_t}{(1-\alpha_t)(1-\bar\alpha_{t-1})}\|x_{t-1}\|_2^2 - 2\left(\frac{\sqrt{\alpha_t}}{1-\alpha_t}x_t + \frac{\sqrt{\bar\alpha_{t-1}}}{1-\bar\alpha_{t-1}}x_0\right)^\top x_{t-1}\right)\right).$$

Now we can find that the covariance is

$$\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} I$$

and the mean is

$$\frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t.$$

The proof is complete. □

Lemma 2.5 is helpful for closed forms of the intermediate consistency terms. If we choose decoders to be Gaussian, we can apply Proposition 1.5 for explicit computation of the KL divergences. This is exactly what we will do in the sequel.

2.3.2 Gaussian Decoder (Learned)

We now parameterize decoders using Gaussian with learnable mean functions:

$$P_{\text{decoder},t}(x_{t-1} \mid x_t) = \mathsf{N}\big(\mu_\theta(x_t, t), \sigma_t^2 I\big),$$

where $\mu_\theta(\cdot,\cdot)$ is typically implemented by a neural network—θ denotes trainable parameters, and σ_t^2 is usually fixed (or also learned). We emphasize that the time t is taken as an additional input to the mean function. In practice, the time will be transformed into embedding by a time embedding network.

At this stage the diffusion ELBO is fully tractable: Each term is either (i) a KL divergence between Gaussian distributions, or (ii) an expectation with respect to a Gaussian distribution. We list the three simplified error terms in below.

- Data reconstruction error:

$$\begin{aligned}
&\mathbb{E}_{X_1 \sim Q_{\text{encoder},1}(\cdot \mid x_0)} \left[\log p_{\text{decoder},1}(x_0 \mid X_1)\right] \\
&= \mathbb{E}_{X_1 \sim \mathsf{N}(\sqrt{\alpha_1} x_0, (1-\alpha_1) I)} \left[-\frac{1}{2\sigma_1^2} \|x_0 - \mu_\theta(X_1, 1)\|_2^2\right] - d \log(\sqrt{2\pi}\sigma_1).
\end{aligned}$$

- Latent term: It does not involve any trainable parameters and we can omit it from ELBO.

- Intermediate consistency term:

$$\begin{aligned}
&\mathbb{E}_{X_t \sim Q_{\text{encoder}}(X_t \mid x_0)} \left[\mathrm{KL}\big(Q_{\text{encoder},t}(\cdot \mid X_t, x_0), P_{\text{decoder},t}(\cdot \mid X_t)\big)\right] \\
&= \mathbb{E}_{X_t \sim Q_{\text{encoder}}(X_t \mid x_0)} \left[\mathrm{KL}\big(\mathsf{N}(\mu_{\text{encoder},t}(X_t, x_0), \sigma_{\text{encoder},t}^2 I), \mathsf{N}(\mu_\theta(X_t, t), \sigma_t^2 I)\big)\right] \\
&= \mathbb{E}_{X_t \sim \mathsf{N}(\sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) I)} \left[\frac{1}{2\sigma_t^2} \|\mu_\theta(X_t, t) - \mu_{\text{encoder},t}(X_t, x_0)\|_2^2\right] \\
&\quad + \frac{d}{2}\left(\frac{\sigma_{\text{encoder},t}^2}{\sigma_t^2} - 1 - \log \frac{\sigma_{\text{encoder},t}^2}{\sigma_t^2}\right).
\end{aligned}$$

Putting together these error terms and omitting θ-independent quantities, we derive the ELBO for DDPM as

$$\begin{aligned}\mathrm{ELBO}_{\mathrm{diff}}(x_0) = \mathbb{E}_{X_1\sim\mathsf{N}(\sqrt{\alpha_1}x_0,(1-\alpha_1)I)}\left[-\frac{1}{2\sigma_1^2}\|x_0-\mu_\theta(X_1,1)\|_2^2\right] \\ -\sum_{t=2}^{T}\mathbb{E}_{X_t\sim\mathsf{N}(\sqrt{\bar\alpha_t}x_0,(1-\bar\alpha_t)I)}\left[\frac{1}{2\sigma_t^2}\|\mu_\theta(X_t,t)-\mu_{\mathrm{encoder},t}(X_t,x_0)\|_2^2\right]. \qquad (2.6)\end{aligned}$$

2.4 Further Parameterization of μ_θ: Predict Data v.s. Predict Noise

Re-examining the intermediate consistency terms in ELBO suggests that the decoder mean function μ_θ is to match the encoder mean $\mu_{\mathrm{encoder},t}$. Invoking Lemma 2.5, we have

$$\mu_{\mathrm{encoder},t}(x_t,x_0) = \frac{\sqrt{\bar\alpha_{t-1}}(1-\alpha_t)}{1-\bar\alpha_t}\,x_0 + \frac{\sqrt{\alpha_t}(1-\bar\alpha_{t-1})}{1-\bar\alpha_t}\,x_t.$$

However, the input to the mean function μ_θ only contains x_t, but not x_0—the data. In this sense, the intermediate consistency terms require the decoder to estimate x_0 given x_t. For convenience, we can represent μ_θ as

$$\mu_\theta(x_t,t) = \frac{\sqrt{\bar\alpha_{t-1}}(1-\alpha_t)}{1-\bar\alpha_t}x_{0,\theta}(x_t,t) + \frac{\sqrt{\alpha_t}(1-\bar\alpha_{t-1})}{1-\bar\alpha_t}x_t,$$

where $x_{0,\theta}$ is another network aiming to estimate x_0 given x_t. Strictly speaking, we can only represent μ_θ in such a way for $t\geq 2$. However, to unify the notation, we may define $\alpha_0 = 1$ and write

$$\begin{aligned}\mu_\theta(x_1,1) &= \frac{\sqrt{\bar\alpha_0}(1-\alpha_1)}{1-\bar\alpha_1}x_{0,\theta}(x_1,1) + \frac{\sqrt{\alpha_1}(1-\bar\alpha_0)}{1-\bar\alpha_1}x_1 \\ &= x_{0,\theta}(x_1,1).\end{aligned}$$

Substitute μ_θ into (2.6), we derive

$$\begin{aligned}\mathrm{ELBO}_{\mathrm{diff}}(x_0) &= \mathbb{E}_{X_1\sim\mathsf{N}(\sqrt{\alpha_1}x_0,(1-\alpha_1)I)}\left[-\frac{1}{2\sigma_1^2}\|x_0-x_{0,\theta}(X_1,1)\|_2^2\right] \\ &\quad -\sum_{t=2}^{T}\mathbb{E}_{X_t\sim\mathsf{N}(\sqrt{\bar\alpha_t}x_0,(1-\bar\alpha_t)I)}\left[\frac{1}{2\sigma_t^2}\frac{\bar\alpha_{t-1}(1-\alpha_t)^2}{(1-\bar\alpha_t)^2}\|x_{0,\theta}(X_t,t)-x_0\|_2^2\right] \\ &= -\sum_{t=1}^{T}\mathbb{E}_{X_t\sim\mathsf{N}(\sqrt{\bar\alpha_t}x_0,(1-\bar\alpha_t)I)}\left[\frac{1}{2\sigma_t^2}\frac{\bar\alpha_{t-1}(1-\alpha_t)^2}{(1-\bar\alpha_t)^2}\|x_{0,\theta}(X_t,t)-x_0\|_2^2\right]. \qquad (2.7)\end{aligned}$$

The ELBO appears analogously to a least-square loss, targeting at estimating data x_0. This makes intuitive sense, as X_t can be viewed as a noisy version of x_0 due to the representation in (2.5). Thus, the ability of "recovering" the data for creating X_t defines a successful decoding chain to generate new data. The loss in (2.7) is known as predicting data parameterization.

We summarize the training algorithm when using the predicting data parameterization in Algorithm 1.

Algorithm 1 Training DDPM via **predicting data**

Input: data set $\mathcal{D}_n$; number of steps T; schedule $\{\alpha_t\}_{t=1}^T$.
Initialize parameters θ of the predictor $x_{0,\theta}(x_t, t)$.
while not converged **do**
  Sample a mini batch $\{x_0^{(i)}\}_{i=1}^B$ from $\mathcal{D}_n$.
  Sample times $t^{(i)} \sim \text{Unif}\{1, \dots, T\}$ independently.
  Sample noises $\epsilon^{(i)} \sim \mathsf{N}(0, I)$ independently.
  Compute

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

  Update θ using stochastic gradient:

$$\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \big\| x_{0,\theta}(x_{t^{(i)}}^{(i)}, t^{(i)}) - x_0^{(i)} \big\|_2^2.$$

end while
Output: trained network $x_{0,\theta}$.

We remark that in implementation, the coefficient $\frac{1}{2\sigma_t^2} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}$ in (2.7) is dropped with a suitable step size in updating θ.

Alternatively, we rearrange (2.5) into

$$X_0 = \frac{X_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon}{\sqrt{\bar{\alpha}_t}}.$$

Substituting it into $\mu_{\text{encoder},t}(x_t, x_0)$ gives rise to

$$\begin{aligned} \mu_{\text{encoder},t}(x_t, x_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \epsilon}{\sqrt{\bar{\alpha}_t}} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t \\ &= \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}} \epsilon. \end{aligned}$$

Correspondingly, we represent μ_θ as

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

where ϵ_θ is another network aiming to estimate the added noise. In this case, we substitute μ_θ into (2.6) and derive

$$\text{ELBO}_{\text{diff}}(x_0) = -\sum_{t=1}^{T} \mathbb{E}_{\epsilon \sim \mathsf{N}(0,I)} \left[\frac{1}{2\sigma_t^2} \frac{(1-\alpha_t)^2}{\alpha_t(1-\bar{\alpha}_t)} \| \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon) - \epsilon \|_2^2 \right]. \tag{2.8}$$

Equation (2.8) is known as predicting noise parameterization and is the most commonly used training loss for diffusion models. The training algorithm for predicting noise is provided in Algorithm 2

Algorithm 2 Training DDPM via **predicting noise**

Input: data set $\mathcal{D}_n$; number of steps T; schedule $\{\alpha_t\}_{t=1}^T$.
Initialize parameters θ of noise predictor $\epsilon_\theta(x_t, t)$.
while not converged **do**
  Sample a mini batch $\{x_0^{(i)}\}_{i=1}^B$ from $\mathcal{D}_n$.
  Sample times $t^{(i)} \sim \mathrm{Unif}\{1, \dots, T\}$ independently.
  Sample noises $\epsilon^{(i)} \sim \mathsf{N}(0, I)$ independently.
  Compute

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

  Update θ using stochastic gradient:

$$\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \big\| \epsilon_\theta(x_{t^{(i)}}^{(i)}, t^{(i)}) - \epsilon^{(i)} \big\|_2^2.$$

end while
Output: trained network $\epsilon_\theta(x_t, t)$.

Again, the original weighting coefficient in (2.8) is dropped.

Remark 2.6 (Forward and backward interpolation)**.** Both Algorithm 1 and Algorithm 2 compute noisy state x_t by adding independently sampled Gaussian noise to clean data point x_0. This is specified by the encoder but can also be understood as a forward data corruption process.

The fixed encoder chain (2.4) defines a Markov process that pushes data toward Gaussian noise:

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1 - \alpha_t}\, \epsilon_t.$$

Equivalently, (2.5) shows that for each t, x_t is a direct Gaussian corruption of x_0:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon.$$

Therefore, t indexes noise levels and $\{\alpha_t\}$ controls how quickly information about x_0 is erased. Large t corresponds to noisy state and less informative about x_0.

The decoder chain learns conditionals $p_{\mathrm{decoder},t}(x_{t-1} \mid x_t)$. The intermediate consistency terms in the diffusion ELBO encourage the learned decoder conditional to match the Bayes-reversed forward conditional $q_{\mathrm{encoder},t}(x_{t-1} \mid x_t, x_0)$. This behaves like a reversal of the encoder chain and is understood as the reverse denoising process.

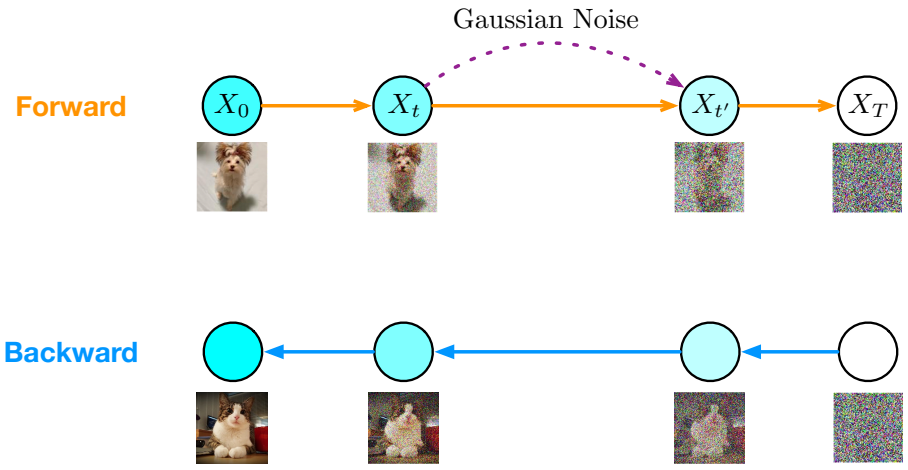


Figure 3: Forward data corruption and backward noise denoising processes.

Remark 2.7 (Predicting noise vs. predicting data (x_0))**.** The decoder mean function can be parameterized either by predicting the *clean data* x_0 or by predicting the *added noise* ϵ. From a representation point of view, the two predictions are related by an explicit bijection:

$$x_{0,\theta}(x_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)\Big) \quad \text{and} \quad \epsilon_\theta(x_t, t) = \frac{1}{\sqrt{1-\bar{\alpha}_t}}\Big(x_t - \sqrt{\bar{\alpha}_t}\,x_{0,\theta}(x_t, t)\Big). \tag{2.9}$$

Consequently, for any function class rich enough to express one predictor, it can express the other, and both induce the same decoder mean family $\mu_\theta(x_t, t)$.

Despite the representations are equivalent at the level of μ_θ, their training algorithm are weighted differently across time steps. Using the bijection (2.9), the two squared errors are deterministically related:

$$\big\|x_{0,\theta}(x_t, t) - x_0\big\|_2^2 = \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}\,\big\|\epsilon_\theta(x_t, t) - \epsilon\big\|_2^2. \tag{2.10}$$

Therefore, even if the two parameterizations represent the same family of decoder mean functions, they place very different emphases on time steps in training. Concretely, as $\bar{\alpha}_t$ decreases as t increases, the predicting data training emphasizes small t losses—$x_{0,\theta}$ should predict x_0 accurately when the added noise is small. While predicting noise emphasizes large or medium t losses.

2.5 Sampling in DDPM

With a trained decoder chain, it is straightforward to generate new data by sequentially apply each of the decoder. Consider the predicting noise parameterization and the sampling algorithm is summarized as follows.

Algorithm 3 Sampling in DDPM using a trained noise predictor

Input: trained network $\epsilon_\theta(x_t, t)$; schedule $\{\alpha_t\}_{t=1}^T$; decoder variance $\{\sigma_t^2\}_{t=1}^T$.
Sample $X_T \sim \mathsf{N}(0, I)$.
for $t = T, T-1, \ldots, 1$ **do**
    Compute decoder mean:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} X_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(X_t, t).$$

    Sample $\epsilon_t \sim \mathsf{N}(0, I)$ (and set $\epsilon = 0$ if $t = 1$).
    Set $X_{t-1} \leftarrow \mu_\theta(X_t, t) + \sigma_t \epsilon_t$.
end for
Output: X_0 as the generated sample.

Remark 2.8 (A standard variance choice)**.** A common DDPM choice on σ_t is to set the decoder variance equal to the reverse-encoder variance, i.e.,

$$\sigma_t^2 = \sigma_{\text{encoder},t}^2 = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}.$$

References

[1] Stanley Chan et al. Tutorial on diffusion models for imaging and vision. *Foundations and Trends® in Computer Graphics and Vision*, 16(4):322–471, 2024.

[2] Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2397–2405. PMLR, 2019.

[3] Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint arXiv:2510.21890*, 2025.

[4] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

[6] TrungTin Nguyen, Faicel Chamroukhi, Hien D Nguyen, and Geoffrey J McLachlan. Approximation of probability density functions via location-scale finite mixtures in lebesgue spaces. *Communications in Statistics-Theory and Methods*, 52(14):5048–5059, 2023.