

From VAE to DDPMs

A Variational Perspective on Diffusion Models — Chapter 2

Myeonggeum Ko

July 13, 2026

Uncertainty Quantification Lab

Department of Statistics, Seoul National University

Variational Autoencoder (VAE)

Hierarchical VAE (HVAE)

Denoising Diffusion Probabilistic Models (DDPM)

Sampling

The common thread: maximum likelihood via a tractable ELBO

We study three generative models under **one variational principle**.

- Objective everywhere: maximum likelihood estimation (MLE),

$$\max_{\phi} \log p_{\phi}(\mathbf{x}),$$

which is intractable, so we maximize a tractable *Evidence Lower Bound* (ELBO) instead.

- The models differ only in *how the latent space is built*.

	Encoder	Decoder	Latent structure
VAE	learned $q_{\theta}(\mathbf{z} \mid \mathbf{x})$	learned $p_{\phi}(\mathbf{x} \mid \mathbf{z})$	single $\mathbf{z}$
HVAE	learned (hierarchical)	learned (hierarchical)	chain $\mathbf{z}_{1:L}$
DDPM	fixed forward diffusion	learned reverse p_{ϕ}	T -step chain $\mathbf{x}_{1:T}$

Variational Autoencoder (VAE)

From autoencoder to VAE

A plain autoencoder compresses $\mathbf{x}$ into $\mathbf{z}$ and reconstructs — but the latent space is unstructured: sampling a random $\mathbf{z}$ gives meaningless output, so it is **not a generative model**.



VAE fixes this by making the latent **probabilistic**: $\mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The generative model draws $\mathbf{z} \sim p(\mathbf{z})$, then $\mathbf{x} \sim p_{\phi}(\mathbf{x} \mid \mathbf{z})$, with marginal

$$p_{\phi}(\mathbf{x}) = \int p_{\phi}(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z} \quad \text{— intractable.}$$

Direct MLE is infeasible, and the posterior $p_{\phi}(\mathbf{z} \mid \mathbf{x}) = p_{\phi}(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) / p_{\phi}(\mathbf{x})$ inherits the same intractability.

Solution: variational approximation

Introduce a learnable encoder $q_{\theta}(\mathbf{z} \mid \mathbf{x}) \approx p_{\phi}(\mathbf{z} \mid \mathbf{x})$ to sidestep the intractable posterior.

- $q_{\theta}(\mathbf{z} \mid \mathbf{x})$ — encoder: approximate posterior;
- $p_{\phi}(\mathbf{x} \mid \mathbf{z})$ — decoder: generative model.

Instead of exact inference, VAE learns an *approximate* posterior, enabling tractable optimization through the ELBO.

- $p_{\phi}(\mathbf{x})$ intractable $\Rightarrow$ maximize a lower bound instead;
- encoder and decoder are trained jointly end-to-end;
- the reparameterization trick makes gradients flow through the sampling step.

Evidence Lower Bound (ELBO)

Multiply and divide by $q_\theta(\mathbf{z} \mid \mathbf{x})$ and apply Jensen's inequality:

$$\log p_\phi(\mathbf{x}) = \log \mathbb{E}_{q_\theta} \left[\frac{p_\phi(\mathbf{x}, \mathbf{z})}{q_\theta(\mathbf{z} \mid \mathbf{x})} \right] \stackrel{\text{Jensen}}{\geq} \mathbb{E}_{q_\theta} \left[\log \frac{p_\phi(\mathbf{x}, \mathbf{z})}{q_\theta(\mathbf{z} \mid \mathbf{x})} \right] =: \mathcal{L}_{\text{ELBO}}.$$

Expanding $p_\phi(\mathbf{x}, \mathbf{z}) = p_\phi(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})$:

Theorem 1 (2.1.1: ELBO)

$$\log p_\phi(\mathbf{x}) \geq \underbrace{\mathbb{E}_{q_\theta} [\log p_\phi(\mathbf{x} \mid \mathbf{z})]}_{\text{reconstruction}} - \underbrace{D_{\text{KL}}(q_\theta(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))}_{\text{regularization}} = \mathcal{L}_{\text{ELBO}}.$$

- **Reconstruction:** encourages accurate recovery of $\mathbf{x}$ given $\mathbf{z}$.
- **Regularization:** keeps the approximate posterior close to the prior $p(\mathbf{z})$.

Why does a VAE produce blurry samples?

With Gaussian encoder and decoder $p_{\phi}(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_{\phi}(\mathbf{z}), \sigma^2 \mathbf{I})$, the ELBO reconstruction term is an ℓ_2 loss:

$$\mathcal{L}_{\text{ELBO}} \iff \underbrace{\frac{1}{2\sigma^2} \|\mathbf{x} - \boldsymbol{\mu}_{\phi}(\mathbf{z})\|^2}_{\text{reconstruction}} + \underbrace{D_{\text{KL}}(q_{\theta}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))}_{\text{regularization}} + \text{const.}$$

Training therefore solves $\arg \min_{\boldsymbol{\mu}} \mathbb{E}[\|\mathbf{x} - \boldsymbol{\mu}(\mathbf{z})\|^2]$, whose minimizer is the conditional mean:

$$\boxed{\boldsymbol{\mu}^*(\mathbf{z}) = \mathbb{E}[\mathbf{x} \mid \mathbf{z}] = \frac{\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[q(\mathbf{z} \mid \mathbf{x}) \mathbf{x}]}{\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[q(\mathbf{z} \mid \mathbf{x})]}.$$

- The decoder output is a weighted average of all data mapped to that latent region.
- Averaging across multiple plausible modes $\Rightarrow$ blurred samples.

Hierarchical VAE (HVAE)

From a single latent to a hierarchy

Standard VAE: one latent $\mathbf{z} \Rightarrow$ limited capacity for multi-scale data.

Hierarchical VAE: multiple latents $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L$ organized top-down.

$$p_{\phi}(\mathbf{x}, \mathbf{z}_{1:L}) = p_{\phi}(\mathbf{x} \mid \mathbf{z}_1) \prod_{i=2}^L p_{\phi}(\mathbf{z}_{i-1} \mid \mathbf{z}_i) p(\mathbf{z}_L).$$

Generation (top-down) $\mathbf{z}_L \rightarrow \dots \rightarrow \mathbf{z}_1 \rightarrow \mathbf{x}$

Inference (bottom-up) $\mathbf{x} \rightarrow \mathbf{z}_1 \rightarrow \dots \rightarrow \mathbf{z}_L$

Generation starts from the highest latent $\mathbf{z}_L$ and is progressively refined toward $\mathbf{x}$.

Hierarchical ELBO

Derived exactly as before via Jensen's inequality:

$$\log p(\mathbf{x}) \geq \mathcal{L}_{\text{ELBO}} = \mathbb{E}_q \left[\log \frac{p_\phi(\mathbf{x}, \mathbf{z}_{1:L})}{q_\theta(\mathbf{z}_{1:L} \mid \mathbf{x})} \right].$$

The only change from the standard VAE is that **multiple** latents appear. It decomposes naturally into

- a **reconstruction** term $\mathbb{E}[\log p_\phi(\mathbf{x} \mid \mathbf{z}_1)]$;
- **layer-wise latent matching** terms across $\mathbf{z}_{i-1} \mid \mathbf{z}_i$;
- a **top-level prior regularization** term on $\mathbf{z}_L$.

Why isn't a deeper VAE enough? (1) Limited variational family

A deeper encoder improves the *network*, not the *posterior family*. Standard VAE assumes

$$q_{\theta}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_{\theta}(\mathbf{x}), \boldsymbol{\Sigma}_{\theta}(\mathbf{x})).$$

- Even with a *full* covariance, this is one Gaussian.
- It is **unimodal**, so it cannot represent a genuinely multi-modal posterior.

Increasing depth does not enlarge the family of distributions q can express.

Why isn't a deeper VAE enough? (2) Posterior collapse

Averaging the ELBO over data decomposes as

$$\mathbb{E}_{p_{\text{data}}}[\mathcal{L}_{\text{ELBO}}] = \underbrace{\mathbb{E}[\log p_{\phi}(\mathbf{x} \mid \mathbf{z})]}_{\text{reconstruction}} - \underbrace{I_q(X; Z)}_{\text{info in latent}} - \underbrace{D_{\text{KL}}(q_{\theta}(\mathbf{z}) \parallel p(\mathbf{z}))}_{\text{prior match}},$$

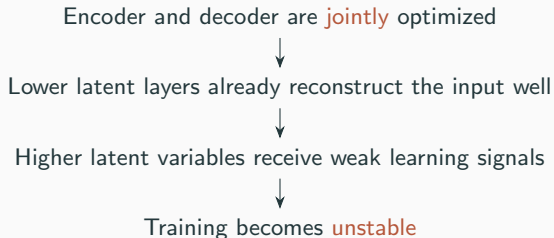
where $q_{\theta}(\mathbf{z}) = \int p_{\text{data}}(\mathbf{x}) q_{\theta}(\mathbf{z} \mid \mathbf{x}) d\mathbf{x}$ is the aggregated posterior.

If the decoder is expressive enough to model data *without* $\mathbf{z}$, i.e. $p_{\phi}(\mathbf{x} \mid \mathbf{z}) \approx r(\mathbf{x})$, then encoding information into $\mathbf{z}$ buys almost no reconstruction gain — but incurs a KL cost. The ELBO then prefers

$$q_{\theta}(\mathbf{z} \mid \mathbf{x}) = p(\mathbf{z}) \implies I_q(X; Z) = 0, \quad D_{\text{KL}} = 0.$$

The encoder outputs the same prior for every input; the latent carries no information; changing $\mathbf{z}$ barely affects the samples.

The remaining challenge



This instability of the *learned* encoder is exactly what DDPM removes.

1. Are VAE / DDPM Bayesian models? What is variational inference?
2. Why a diagonal (not full) encoder covariance?
3. Why does reconstruction-only training lead to memorization?

Q1: VAE and DDPM as variational inference

- **Classical VI** optimizes a separate q for *each* datapoint — expensive and does not generalize to new data.
- **VAE** replaces this with a **single encoder network** $q_{\theta}(\mathbf{z} \mid \mathbf{x})$ that outputs the approximate posterior for *any* $\mathbf{x}$. The network is trained once and shared across all datapoints.
- **DDPM** follows the same variational framework, but the latent is the **entire trajectory** $\mathbf{z} = \mathbf{x}_{1:T}$ rather than a single $\mathbf{z}$. Conditioning on $\mathbf{x}_0$ makes each posterior $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ tractable, and training again maximizes an ELBO.

Both are Bayesian latent-variable models; they differ in the *latent structure* and in *how* the approximate posterior is obtained.

Q2: Why a diagonal encoder covariance?

Full covariance is **feasible, not impossible** — the KL is still closed-form. Diagonal is preferred for practical and fundamental reasons.

- **Cost.** Diagonal predicts d variances; full predicts $\frac{d(d+1)}{2}$ entries and requires a Cholesky factor to enforce positive-definiteness.
- **Fundamental limit.** Full covariance only *rotates* the ellipse — the posterior is still a single **unimodal** Gaussian. A genuinely multimodal posterior cannot be captured either way.

The real ceiling is the Gaussian family itself. Post-VAE work therefore extends the *form* of the variational family instead: normalizing-flow posteriors, hierarchical posteriors, Gaussian-mixture posteriors. These capture multimodality — full covariance does not.

Q3: Why does reconstruction-only training lead to memorization?

With only reconstruction loss $\sum_i \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|^2$, there is **no constraint** on where f sends each $\mathbf{x}_i$ in $\mathbb{R}^d$. The easiest solution: map every $\mathbf{x}_i$ to a *distinct* code, then invert it exactly.

- f : assigns each $\mathbf{x}_i$ a unique code $\mathbf{z}_i$ — arbitrary location, no structure required.
- g : learns to recover $\mathbf{x}_i$ from $\mathbf{z}_i$ — a **lookup table**, not a generative model.

A sufficiently expressive network can implement $g(f(\mathbf{x})) = \mathbf{x}$ (identity mapping) perfectly.

The problem becomes clear at *sampling time*: codes $\{\mathbf{z}_i\}$ cluster arbitrarily in $\mathbb{R}^d$, far from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ — a sampled $\mathbf{z}$ almost surely lands where the decoder has never been trained, producing nonsense.

Why the KL term fixes this

$D_{\text{KL}}(q_\theta(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z}))$ forces every μ_i toward the origin and encoder variance toward 1, so training codes overlap with $\mathcal{N}(\mathbf{0}, \mathbf{I})$. VAE also samples $\mathbf{z} \sim q_\theta(\mathbf{z} | \mathbf{x})$ rather than using a point code, so the decoder is trained over a *region*, filling the gaps between codes.

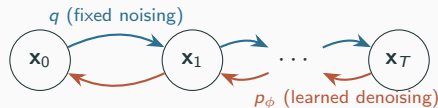
Denoising Diffusion Probabilistic Models (DDPM)

Motivation: freeze the encoder

HVAEs keep a learnable encoder $\Rightarrow$ unstable joint optimization. DDPM resolves this by

- fixing the encoder as a forward diffusion (noising) process, and
- learning only the reverse denoising process,

which yields stable and scalable training.



Parallel to the VAE: the fixed forward process plays the role of the encoder, the learned reverse process the role of the decoder.

Forward process (fixed encoder)

The encoder is a fixed, non-trainable Markov chain. Starting from $\mathbf{x}_0 \sim p_{\text{data}}$, Gaussian noise is added step by step,

$$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \cdots \rightarrow \mathbf{x}_T, \quad \mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Gaussian forward transition.

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\sqrt{1 - \beta_t^2} \mathbf{x}_{t-1}, \beta_t^2 \mathbf{I}\right).$$

Define $\alpha_t = \sqrt{1 - \beta_t^2}$ (so that $\alpha_t^2 + \beta_t^2 = 1$, keeping the variance controlled). Then

$$\mathbf{x}_t = \alpha_t \mathbf{x}_{t-1} + \beta_t \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

- α_t : preserves information from the previous state;
- β_t : controls the amount of injected noise;
- $\{\beta_t\}$: a predefined noise schedule.

Closed-form forward distribution

Composing the linear-Gaussian steps gives a one-shot expression:

$$p(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\bar{\alpha}_t \mathbf{x}_0, (1 - \bar{\alpha}_t^2) \mathbf{I}\right), \quad \bar{\alpha}_t = \prod_{s=1}^t \alpha_s.$$

Equivalently, in reparameterized form,

$$\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t^2} \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Why the prior is Gaussian. As $t \rightarrow T$,

$$\bar{\alpha}_t = \prod_{s=1}^t \alpha_s \longrightarrow 0 \implies \mathbf{x}_T \approx \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

The forward process transports $p_{\text{data}}(\mathbf{x})$ to the tractable prior $p_{\text{prior}}(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Reverse denoising process

Goal. Undo the forward map: start from noise and recover data.

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1} | \mathbf{x}_t)$ for $t = T, \dots, 1$, until $\mathbf{x}_0$.

Fundamental question: the unknown reverse kernel

The true reverse transition $p(\mathbf{x}_{t-1} | \mathbf{x}_t)$ is **unknown**. By Bayes' rule

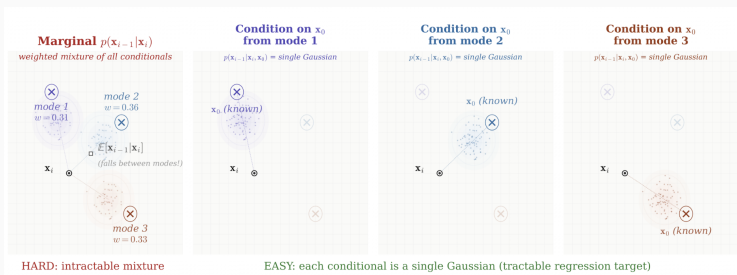
$$p(\mathbf{x}_{t-1} | \mathbf{x}_t) = p(\mathbf{x}_t | \mathbf{x}_{t-1}) \frac{p_{t-1}(\mathbf{x}_{t-1})}{p_t(\mathbf{x}_t)},$$

where $p(\mathbf{x}_t | \mathbf{x}_{t-1})$ is known but the marginal $p_t(\mathbf{x}_t)$ is intractable.

We therefore learn a model $p_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t)$ by minimizing $\mathbb{E}[D_{\text{KL}}(p(\mathbf{x}_{t-1} | \mathbf{x}_t) \| p_\phi(\mathbf{x}_{t-1} | \mathbf{x}_t))]$.

The conditioning trick

Rather than the marginal reverse kernel, condition on the **clean** sample $\mathbf{x}_0$.



By Bayes' rule and the Markov property,

$$p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = p(\mathbf{x}_t | \mathbf{x}_{t-1}) \frac{p(\mathbf{x}_{t-1} | \mathbf{x}_0)}{p(\mathbf{x}_t | \mathbf{x}_0)}, \quad p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = p(\mathbf{x}_t | \mathbf{x}_{t-1}).$$

All three factors are Gaussian $\Rightarrow p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ is Gaussian and tractable.

Marginal vs. conditional KL are equivalent targets

Theorem 2 (2.2.1: Equivalence of marginal and conditional KL)

For every step i ,

$$\begin{aligned} & \mathbb{E}_{p_i(\mathbf{x}_i)} [D_{\text{KL}}(p(\mathbf{x}_{i-1} \mid \mathbf{x}_i) \parallel p_\phi(\mathbf{x}_{i-1} \mid \mathbf{x}_i))] \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x}_0)} \mathbb{E}_{p(\mathbf{x}_i \mid \mathbf{x}_0)} [D_{\text{KL}}(p(\mathbf{x}_{i-1} \mid \mathbf{x}_i, \mathbf{x}_0) \parallel p_\phi(\mathbf{x}_{i-1} \mid \mathbf{x}_i))] + C, \end{aligned}$$

with C independent of ϕ . The minimizer satisfies

$$p^*(\mathbf{x}_{i-1} \mid \mathbf{x}_i) = \mathbb{E}_{p(\mathbf{x}_0 \mid \mathbf{x}_i)} [p(\mathbf{x}_{i-1} \mid \mathbf{x}_i, \mathbf{x}_0)] = p(\mathbf{x}_{i-1} \mid \mathbf{x}_i).$$

So training against the **tractable** conditional target recovers the true (intractable) reverse kernel in expectation.

Lemma 2.2.2: the reverse conditional kernel

Theorem 3 (2.2.2: Reverse conditional transition kernel)

$p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is Gaussian:

$$p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0, t), \sigma^2(t)\mathbf{I}),$$
$$\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0, t) = \frac{\bar{\alpha}_{t-1}\beta_t^2}{1 - \bar{\alpha}_t^2} \mathbf{x}_0 + \frac{(1 - \bar{\alpha}_{t-1}^2)\alpha_t}{1 - \bar{\alpha}_t^2} \mathbf{x}_t, \quad \sigma^2(t) = \frac{1 - \bar{\alpha}_{t-1}^2}{1 - \bar{\alpha}_t^2} \beta_t^2.$$

The mean is a convex combination of the clean sample $\mathbf{x}_0$ and the noisy state $\mathbf{x}_t$; the variance is a fixed, schedule-dependent scalar.

Lemma 2.2.2: derivation

Bayes' rule with the Markov property gives $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \propto p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{x}_0)$, so the exponent (times -2) is

$$\frac{\|\mathbf{x}_t - \alpha_t \mathbf{x}_{t-1}\|^2}{\beta_t^2} + \frac{\|\mathbf{x}_{t-1} - \bar{\alpha}_{t-1} \mathbf{x}_0\|^2}{1 - \bar{\alpha}_{t-1}^2}.$$

Collecting the quadratic term in $\mathbf{x}_{t-1}$ and using $\beta_t^2 + \alpha_t^2(1 - \bar{\alpha}_{t-1}^2) = 1 - \bar{\alpha}_t^2$ gives

$$\sigma^2(t) = \frac{\beta_t^2(1 - \bar{\alpha}_{t-1}^2)}{1 - \bar{\alpha}_t^2},$$

and reading off the linear term gives the mean

$$\boldsymbol{\mu} = \frac{(1 - \bar{\alpha}_{t-1}^2)\alpha_t}{1 - \bar{\alpha}_t^2} \mathbf{x}_t + \frac{\bar{\alpha}_{t-1}\beta_t^2}{1 - \bar{\alpha}_t^2} \mathbf{x}_0. \quad \blacksquare$$

Modeling the reverse transition

Reverse model. Approximate the tractable target by a Gaussian with fixed covariance:

$$p_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\phi(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}).$$

Same-covariance Gaussian KL reduces the training objective to

$$\mathcal{L}_{\text{diffusion}} \propto \sum_t \mathbb{E} \|\boldsymbol{\mu}_\phi(\mathbf{x}_t, t) - \boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0, t)\|^2.$$

Substituting $\mathbf{x}_0 = (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t^2} \boldsymbol{\varepsilon}) / \bar{\alpha}_t$ into the Lemma 2.2.2 mean,

$$\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0, t) = \frac{1}{\alpha_t} \left(\mathbf{x}_t - \frac{\beta_t^2}{\sqrt{1 - \bar{\alpha}_t^2}} \boldsymbol{\varepsilon} \right),$$

so predicting $\boldsymbol{\mu}$ is equivalent to predicting $\boldsymbol{\varepsilon}$, giving the simple DDPM loss:

$$\boxed{\mathcal{L}_{\text{simple}} = \mathbb{E} \|\boldsymbol{\varepsilon}_\phi(\mathbf{x}_t, t) - \boldsymbol{\varepsilon}\|^2}$$

with target always $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ — consistent scale, stable optimization.

An equivalent parameterization: $\mathbf{x}$ -prediction

Instead of noise, predict the clean sample $\mathbf{x}_\phi(\mathbf{x}_t, t) \approx \mathbf{x}_0$. Since $\boldsymbol{\mu} = A_t \mathbf{x}_0 + B_t \mathbf{x}_t$ and $\boldsymbol{\mu}_\phi = A_t \mathbf{x}_\phi + B_t \mathbf{x}_t$,

$$\boldsymbol{\mu}_\phi - \boldsymbol{\mu} = A_t(\mathbf{x}_\phi - \mathbf{x}_0) \implies \boxed{\arg \min \|\boldsymbol{\mu}_\phi - \boldsymbol{\mu}\|^2 = \arg \min \|\mathbf{x}_\phi - \mathbf{x}_0\|^2.}$$

All three targets $(\boldsymbol{\mu}, \boldsymbol{\varepsilon}, \mathbf{x})$ minimize the same objective up to a per-step weight; they differ only in numerical conditioning across t .

DDPM as an ELBO: DDPM's objective is a variational bound

The reverse kernels define a *joint generative model*

$$p_{\phi}(\mathbf{x}_0, \mathbf{x}_{1:T}) = p_{\phi}(\mathbf{x}_0 \mid \mathbf{x}_1) p_{\phi}(\mathbf{x}_1 \mid \mathbf{x}_2) \cdots p_{\phi}(\mathbf{x}_{T-1} \mid \mathbf{x}_T) p_{\text{prior}}(\mathbf{x}_T),$$

with data marginal $p_{\phi}(\mathbf{x}_0) = \int p_{\phi}(\mathbf{x}_0, \mathbf{x}_{1:T}) d\mathbf{x}_{1:T}$.

Exactly as in the VAE / HVAE, maximum-likelihood training of $p_{\phi}(\mathbf{x}_0)$ is intractable, so we maximize an ELBO obtained by Jensen's inequality.

Theorem 4 (2.2.3: DDPM's ELBO)

$$-\log p_{\phi}(\mathbf{x}_0) \leq -\mathcal{L}_{\text{ELBO}}(\mathbf{x}_0; \phi) = \mathcal{L}_{\text{prior}}(\mathbf{x}_0) + \mathcal{L}_{\text{recon}}(\mathbf{x}_0; \phi) + \mathcal{L}_{\text{diffusion}}(\mathbf{x}_0; \phi).$$

The proof mirrors the HVAE ELBO (Jensen) with the forward-posterior simplification; details deferred.

The three loss terms

$$\mathcal{L}_{\text{prior}}(\mathbf{x}_0) = D_{\text{KL}}(p(\mathbf{x}_T | \mathbf{x}_0) \| p_{\text{prior}}(\mathbf{x}_T)), \quad \mathcal{L}_{\text{recon}}(\mathbf{x}_0; \phi) = \mathbb{E}_{p(\mathbf{x}_1 | \mathbf{x}_0)}[-\log p_{\phi}(\mathbf{x}_0 | \mathbf{x}_1)],$$

$$\mathcal{L}_{\text{diffusion}}(\mathbf{x}_0; \phi) = \sum_{t=2}^T \mathbb{E}_{p(\mathbf{x}_t | \mathbf{x}_0)} \left[D_{\text{KL}}(p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_{\phi}(\mathbf{x}_{t-1} | \mathbf{x}_t)) \right].$$

- $\mathcal{L}_{\text{prior}}$: **no parameters**; made negligible by a forward noise schedule with $p(\cdot | \mathbf{x}_0) \approx p_{\text{prior}}(\cdot)$.
- $\mathcal{L}_{\text{recon}}$: the final decoding step; estimated by Monte Carlo.
- $\mathcal{L}_{\text{diffusion}}$: the **core** term — matches each learned reverse kernel $p_{\phi}(\mathbf{x}_{t-1} | \mathbf{x}_t)$ to the tractable posterior $p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$. This is the sum we already reduced to the ε - / $\mathbf{x}$ -regression loss.

Sampling

Ancestral sampling

Train the ε -model, then *freeze* it: $\varepsilon_{\phi^*}(\mathbf{x}_t, t)$. Draw $\mathbf{x}_T \sim p_{\text{prior}} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iterate for $t = T, \dots, 1$:

$$\mathbf{x}_{t-1} = \underbrace{\frac{1}{\alpha_t} \left(\mathbf{x}_t - \frac{1 - \alpha_t^2}{\sqrt{1 - \bar{\alpha}_t^2}} \varepsilon_{\phi^*}(\mathbf{x}_t, t) \right)}_{\mu_{\phi^*}(\mathbf{x}_t, t)} + \sigma(t) \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

- $\sigma^2(t) = \frac{(1 - \bar{\alpha}_{t-1}^2)(1 - \alpha_t^2)}{1 - \bar{\alpha}_t^2}$ is the **Lemma 2.2.2 posterior variance**.
- Each step subtracts a scaled noise estimate, then injects fresh Gaussian noise.

Repeating down to $t = 1$ yields the clean sample $\mathbf{x}_0$.

Denoise-then-re-noise view (1/2)

Rewrite the predicted noise via the forward relation $\mathbf{x}_t = \bar{\alpha}_t \mathbf{x}_{\phi^*}(\mathbf{x}_t, t) + \sqrt{1 - \bar{\alpha}_t^2} \varepsilon_{\phi^*}(\mathbf{x}_t, t)$:

$$\varepsilon_{\phi^*}(\mathbf{x}_t, t) = \frac{\mathbf{x}_t - \bar{\alpha}_t \mathbf{x}_{\phi^*}(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t^2}}.$$

Substituting into the update and rearranging,

$$\mathbf{x}_{t-1} = \bar{\alpha}_{t-1} \mathbf{x}_{\phi^*}(\mathbf{x}_t, t) + \underbrace{\sqrt{1 - \bar{\alpha}_{t-1}^2 - \sigma^2(t)} \varepsilon_{\phi^*}(\mathbf{x}_t, t) + \sigma(t) \varepsilon_t}_{\text{combined std} = \sqrt{1 - \bar{\alpha}_{t-1}^2}}.$$

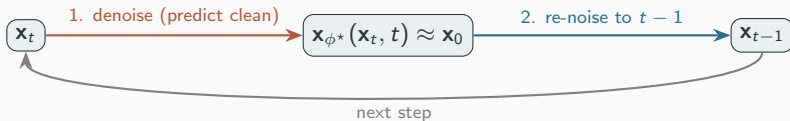
The two noise contributions have orthogonal, matching variances:

$$(1 - \bar{\alpha}_{t-1}^2 - \sigma^2(t)) + \sigma^2(t) = 1 - \bar{\alpha}_{t-1}^2, \quad \sqrt{1 - \bar{\alpha}_{t-1}^2 - \sigma^2(t)} = \frac{\alpha_t(1 - \bar{\alpha}_{t-1}^2)}{\sqrt{1 - \bar{\alpha}_t^2}} \geq 0.$$

Denoise-then-re-noise view (2/2)

Compare with the *forward* marginal $\mathbf{x}_{t-1} = \bar{\alpha}_{t-1}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}^2} \bar{\epsilon}$: the structure is **identical** — signal coefficient $\bar{\alpha}_{t-1}$, noise std $\sqrt{1 - \bar{\alpha}_{t-1}^2}$ — except the unknown $\mathbf{x}_0$ is replaced by the network prediction $\mathbf{x}_{\phi^*}(\mathbf{x}_t, t)$. Each step therefore

denoises then re-noises to level $t - 1$:



Sampling moves from high to low noise: early steps fix global structure, later steps add fine detail.

Why DDPM sampling is blurry and slow

Blurry. Even the optimal denoiser is the conditional mean $\mathbf{x}_{\phi^*} = \mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t]$, i.e. an **average** clean sample — smooth at high noise, sharpened only as noise decreases.

Slow. Theorem 2.2.1 says an expressive $p_{\phi}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ can match the true reverse $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, but in practice it is a *single Gaussian*:

- small β_t : the true reverse is nearly Gaussian $\Rightarrow$ accurate;
- large β_t : multimodal / non-Gaussian $\Rightarrow$ a Gaussian fails.

Accuracy therefore forces **many small steps** (~ 1000), an $O(T)$ *sequential* chain of network evaluations that cannot be parallelized.

Chapter 4 recasts this bottleneck as a differential-equation problem, motivating continuous-time solvers for fast sampling.

1. Do the denoising steps need to be adaptive?
2. Could a Gaussian mixture reverse kernel help?
3. Why is the reverse variance fixed?

Q1: Do the denoising steps need to be adaptive?

DDPM fixes ~ 1000 timesteps in advance and uses the *same* schedule for every sample.

But the reverse trajectory is not uniformly important:

- high-noise region (early steps): the state changes in a relatively simple way;
- low-noise region (late steps): small errors strongly affect the final image quality.

Hence not all 1000 steps are needed — it is already known that $\sim 20\text{--}50$ steps suffice for good quality with the right solver. The next question is whether to place those few steps *adaptively* rather than by a fixed heuristic.

Q1: Adaptive timestep scheduling

Treat the timestep set itself as an **optimization variable**: derive a *sampling-error upper bound* and minimize it over step locations.

- **Continuous view** — step positions are real variables $\Rightarrow$ optimize by gradient descent.
- **Discrete view** — steps must be integers $\Rightarrow$ a *greedy* algorithm moves a step to wherever it reduces the error most.

Representative work

- **Align Your Steps** (Sabour et al., ICML 2024): minimize a KL upper bound between the true and a linearized reverse SDE to pick the schedule.
- **Optimized Time Steps** (Xue et al., CVPR 2024): choose steps that minimize the gap between the exact ODE solution and the solver's approximation.
- **Adaptive Non-uniform Timestep Sampling** (Kim et al., CVPR 2025): adaptivity also applies to *training* (instead of drawing $t \sim \text{Uniform}(1, \dots, T)$) — sample high gradient-variance timesteps more often, as they are the convergence bottleneck.

Q3: Can we use a Gaussian-mixture reverse kernel?

Single Gaussian is not arbitrary — for small β_t , the true posterior $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is approximately Gaussian, making the KL closed-form (training = MSE). A mixture $\sum_k \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ raises expressiveness but loses this, forcing a new variational bound.

- **GM-DDPM** (2023): derives a new bound for the mixture–mixture KL.
- **Gaussian Mixture Solvers** (NeurIPS 2024): reframe as a general sampling problem (Langevin / MALA) instead.
- **Ishtiaque et al. (2026)**: GM kernels universally approximate any conditional reverse kernel.

Yet the gain is offset by two costs:

- **No closed-form KL** — the ELBO must be redesigned.
- **Diminishing returns** — $\varepsilon_\phi(\mathbf{x}_t, t)$ already captures multimodal structure through its mean; kernel shape is a secondary lever.

Q3: Why fix the reverse variance?

Recall the tractable target from Lemma 2.2.2:

$$p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0, t), \sigma^2(t)\mathbf{I}).$$

The two moments behave very differently:

$\sigma^2(t)$ does *not* depend on $\mathbf{x}_0$

vs.

$\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0, t)$ depends strongly on $\mathbf{x}_0$.

The variance is already known in closed form; only the mean is genuinely hard to match. Fixing $\sigma^2(t)$ collapses the Gaussian–Gaussian KL into a plain MSE on the mean.

Empirical picture (Ho et al., 2020, Table 2)

Training with a *learned* variance was **unstable** and produced poor samples — it was not merely worse, it simply failed to train. Among the two fixed choices ($\sigma_t^2 = \beta_t$ vs. $\tilde{\beta}_t$), both gave similar results. Nichol & Dhariwal (2021) later stabilized variance learning via a hybrid objective, recovering improved log-likelihood — but this required additional design work.

$\sigma_t^2 = \beta_t$: upper bound on reverse entropy (optimal if $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$); $\tilde{\beta}_t = \frac{(1 - \bar{\alpha}_{t-1}^2)\beta_t^2}{1 - \bar{\alpha}_t^2}$: Lemma 2.2.2 posterior variance (optimal if $\mathbf{x}_0$ is a fixed point).

Closing remarks

- **VAE** — probabilistic encoder–decoder via ELBO; blurry and unimodal.
- **HVAE** — stacked latents for coarse-to-fine capacity; unstable joint training.
- **DDPM** — fix the encoder, learn only the reverse; decomposes into simple denoising regressions via the conditioning trick $p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$.

Two roads ahead:

- **Score-based view** — learn $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ directly; ε -prediction *is* the scaled score.
- **Faster sampling** — treat the reverse chain as an ODE/SDE: DDIM, DPM-Solver, EDM, Consistency Models.

References

- Sabour, Fidler, Kreis. *Align Your Steps: Optimizing Sampling Schedules in Diffusion Models*. ICML 2024. arXiv:2404.14507.
- Xue, Liu, Chen, Zhang, Hu, Xie, Li. *Accelerating Diffusion Sampling with Optimized Time Steps*. CVPR 2024. arXiv:2402.17376.
- Kim, Ki, Shim, Lee. *Adaptive Non-uniform Timestep Sampling for Accelerating Diffusion Model Training*. CVPR 2025. arXiv:2411.09998.
- *GM-DDPM: Denoising Diffusion Probabilistic Models with Gaussian Mixture Noise*. OpenReview (id: xi4qWLNbhs), 2023.
- Ishtiaque et al. *Universality of Gaussian-Mixture Reverse Kernels in Conditional Diffusion*. arXiv:2604.13470, 2026.
- *Gaussian Mixture Solvers for Diffusion Models*. NeurIPS 2024.

Thank you!