Chapter 3: Efficient Sampling: DDIM and Consistency Models

We have introduced diffusion models as multi-layered VAEs, where training can be derived from a diffusion ELBO and sampling proceeds by an ancestral decoder chain. While conceptually clean, the standard DDPM sampler requires many steps (often hundreds to thousands) to reach high quality. In this chapter we study two approaches to reduce sampling cost:

- **Denoising Diffusion Implicit Models (DDIMs)** replace the stochastic decoder chain by lower noise or *deterministic* updates that allow fewer steps.
- **Consistency models** learn a time-indexed *denoising map* that is consistent across time in decoding, enabling very few-step (even one-step) generation.

1 Accelerate Sampling by Compressing Decoding Steps

The standard DDPM generates new data X_0 starting from $X_T \sim \mathsf{N}(0, I)$ by executing the decoder chain

$$X_T \to X_{T-1} \to \cdots \to X_0,$$

where each step is conditional Gaussian and removes a small amount of noise:

$$X_{t-1} \mid X_t \sim \mathsf{N}(\mu_\theta(X_t, t), \sigma_t^2 I) \quad \text{with} \quad \mu_\theta(X_t, t) = \frac{1}{\sqrt{\alpha_t}} X_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(X_t, t).$$

(Here, we use the noise prediction parameterization and ϵ_θ is the noise estimation network.) In practice, high-fidelity generation often requires hundreds to thousands of steps. Recall the forward encoder marginal:

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

These steps are designed with two consequences:

- **Small denoising per step.** Each update only performs a mild correction, that is, the given noisy state X_t is a strong statistical prior of X_{t-1}, making X_{t-1} sampling reliable.
- **Error accumulation.** The diffusion model has error at every step (e.g., imperfect noise prediction ϵ_θ). If we attempt to take very large steps, errors can compound and produce artifacts.

Hence, decoding in DDPM stays in a regime where each move is both (i) statistically well-behaved and (ii) numerically stable. However, it is always tempting to ask:

> *Can we replace the long sequence of small, incremental denoising updates in DDPM by a shorter sequence of larger updates, while keeping sample quality comparable?*

To make this precise, consider a *coarse* sequence of times

$$0 = t_0 < t_1 < \cdots < t_K = T \qquad \text{with } K \ll T,$$

and run the reverse-time chain only on this subsampled grid:

$$X_{t_K} \to X_{t_{K-1}} \to \cdots \to X_{t_0}.$$

An accelerated decoding chain would produce a trajectory

$$X_{t_0} \to X_{t_1} \to \cdots \to X_{t_K},$$

where each transition approximates what the original DDPM chain would yield after traversing multiple steps. More precisely, DDPM provides a trained neural network indexed by time that given (X_t, t), it estimates either the added noise $\epsilon_\theta(X_t, t)$ or the clean data $x_{0,\theta}(X_t, t)$. Compressing the decoding chain means using the network to predict the effect of *many* small decoding steps, allowing us to update X_t directly to a much cleaner state X_s for $s \ll t$. However, such larger decoding steps typically lead to challenges:

- **Correct noise-level geometry.** At time t, the typical magnitude of noise is $\sqrt{1-\bar{\alpha}_t}$. A jump from t to s must respect the target noise level $1-\bar{\alpha}_s$; otherwise the iterate will drift and errors can explode.
- **Faithfulness to the learned network.** The network ϵ_θ or $x_{0,\theta}$ is trained on the original time grid. A fast decoding chain should be expressible using the same network so as to reduce re-training cost.

These considerations suggest that fast sampling is not merely "skipping steps"; it requires *re-deriving* update rules that are valid on a coarse time schedule.

2 Denoising Diffusion Implicit Models (DDIMs)

DDIM is a family of accelerated samplers that reuse a DDPM-trained denoiser but modify the *reverse-time decoding transitions*. The key idea is to control (and, in the extreme, remove) the stochasticity injected at each reverse step so that *larger time jumps* become viable on a coarse time schedule, while still matching the same *marginal noise levels* $Q_{\text{encoder}}(X_t \mid X_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}X_0, (1-\bar{\alpha}_t)I)$.

A reminder: the diffusion ELBO lens. We recall the diffusion ELBO:

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

In DDPM, one first specifies the *forward* encoder Markov chain and then obtains the Gaussian reverse-time posterior $Q_{\text{encoder},t}(X_{t-1} \mid X_t, X_0)$ by Bayes' rule. DDIM takes a different viewpoint: it constructs an alternative family of reverse-time conditionals that (i) preserve the *same forward marginals* $Q_{\text{encoder}}(X_t \mid X_0)$ and (ii) allow controllable stochasticity.

2.1 Correlated Noise Construction of DDIM Reverse Conditionals

The DDPM forward marginal is

$$X_t = \sqrt{\bar{\alpha}_t}\, X_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathsf{N}(0, I). \tag{2.1}$$

Similarly,

$$X_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, X_0 + \sqrt{1-\bar{\alpha}_{t-1}}\, \epsilon', \qquad \epsilon' \sim \mathsf{N}(0, I). \tag{2.2}$$

If ϵ' and ϵ were correlated—in the extreme $\epsilon' = \epsilon$—then, given (X_t, X_0), inferring X_{t-1} would be easy: one would reuse the same noise realization across time.

Motivated by this, we write

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

and reparameterize ϵ' as

$$\epsilon' = a_{t-1}\, \epsilon + b_{t-1}\, Z, \qquad Z \sim \mathsf{N}(0, I), \tag{2.3}$$

where a_{t-1} and b_{t-1} are scalars controlling correlation and injected randomness, respectively, and Z is independent. Plugging (2.3) into (2.2) yields the reverse-time conditional

$$Q_{\text{encoder},t}(X_{t-1} \mid X_t, X_0) = \mathsf{N}\left(\sqrt{\bar{\alpha}_{t-1}} X_0 + \sqrt{1-\bar{\alpha}_{t-1}}\, a_{t-1}\, \frac{X_t - \sqrt{\bar{\alpha}_t} X_0}{\sqrt{1-\bar{\alpha}_t}},\ \ (1-\bar{\alpha}_{t-1}) b_{t-1}^2 I\right). \tag{2.4}$$

So far this is just an *ansatz*: we must choose a_{t-1}, b_{t-1} so that the induced marginal $Q_{\text{encoder}}(X_{t-1} \mid X_0)$ remains exactly $\mathsf{N}(\sqrt{\bar{\alpha}_{t-1}} X_0, (1-\bar{\alpha}_{t-1}) I)$.

Lemma 2.1 (Choosing a_{t-1}, b_{t-1} to preserve the forward marginal)**.** Fix any $\sigma_{t-1}^2 \in [0,\, 1-\bar{\alpha}_{t-1}]$ and set

$$(1-\bar{\alpha}_{t-1})\, b_{t-1}^2 = \sigma_{t-1}^2, \qquad a_{t-1} = \sqrt{1-b_{t-1}^2} = \frac{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_{t-1}^2}}{\sqrt{1-\bar{\alpha}_{t-1}}}.$$

Then, under (2.1)-(2.3), it holds that

$$Q_{\text{encoder}}(X_{t-1} \mid X_0) = \mathsf{N}(\sqrt{\bar{\alpha}_{t-1}} X_0,\ (1-\bar{\alpha}_{t-1}) I)\,.$$

Proof. Marginalizing over X_t, we derive

$$\begin{aligned} X_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} X_0 + \sqrt{1-\bar{\alpha}_{t-1}} a_{t-1} \frac{X_t - \sqrt{\bar{\alpha}_t} X_0}{\sqrt{1-\bar{\alpha}_t}} + \sqrt{1-\bar{\alpha}_{t-1}} b_{t-1} Z \\ &= \sqrt{\bar{\alpha}_{t-1}} X_0 + \sqrt{1-\bar{\alpha}_{t-1}} (a_{t-1}\epsilon + b_{t-1} Z), \end{aligned}$$

where the last equality invokes the identity (2.1). Since Z is independent of ϵ and $a_{t-1}^2 + b_{t-1}^2 = 1$, we obtain

$$a_{t-1}\epsilon + b_{t-1} Z \sim \mathsf{N}(0, I).$$

This exactly implies $X_{t-1} \mid X_0 \sim \mathsf{N}(\sqrt{\bar{\alpha}_{t-1}} X_0, (1-\bar{\alpha}_{t-1}) I)$. □

A useful reparameterization. With the lemma's choice, (2.4) can be rewritten more transparently as

$$Q_{\text{encoder},t}(X_{t-1} \mid X_t, X_0) = \mathsf{N}\left(\sqrt{\bar{\alpha}_{t-1}}X_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_{t-1}^2}\,\frac{X_t - \sqrt{\bar{\alpha}_t}X_0}{\sqrt{1-\bar{\alpha}_t}},\quad \sigma_{t-1}^2 I\right), \tag{2.5}$$

where $\sigma_{t-1}^2 \in [0, 1-\bar{\alpha}_{t-1}]$ explicitly controls the amount of injected independent randomness. Setting $\sigma_{t-1}^2 = 0$ yields a deterministic transition in the decoder chain.

Non-Markov encoder distribution. It is a good exercise to derive the encoder distribution $Q_{\text{encoder},t}(X_t \mid X_{t-1}, X_0)$. One may use the Bayes' rule to express the density $q_{\text{encoder},t}(X_t \mid X_{t-1}, X_0)$ in terms of $q_{\text{encoder}}(X_t \mid X_0)$, $q_{\text{encoder}}(X_{t-1} \mid X_0)$, and $q_{\text{encoder},t}(X_{t-1} \mid X_t, X_0)$. The result will provide a precise dependence structure in the encoder chain.

2.2 From Reverse Encoder to Sampling

Equation (2.5) is a *reverse encoder posterior*: it assumes access to the true X_0, which we do not have during sample generation. The DDPM training objective trains a neural network that predicts either the added noise ϵ (noise prediction) or the clean data X_0 (data prediction) from (X_t, t). Using the noise-prediction parameterization, we have

$$\widehat{X}_0 = \frac{X_t - \sqrt{1-\bar{\alpha}_t}\;\epsilon_\theta(X_t, t)}{\sqrt{\bar{\alpha}_t}} \approx X_0.$$

Therefore, the DDIM sampler replaces X_0 in (2.5) by $\widehat{X}_0$. This yields the DDIM decoding transition:

$$X_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\widehat{X}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_{t-1}^2}\;\epsilon_\theta(X_t, t) + \sigma_{t-1}\,Z, \qquad Z \sim \mathsf{N}(0, I). \tag{2.6}$$

Two extreme cases are worth highlighting:

- **Deterministic DDIM.** If $\sigma_{t-1} = 0$, then we have
$$X_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\widehat{X}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\;\epsilon_\theta(X_t, t).$$
The whole trajectory is a deterministic function of the initial latent X_T.
- **Stochastic DDIM / DDPM-like.** Larger σ_{t-1} injects more randomness per step. DDPM corresponds to a particular, time-dependent variance choice that matches the Bayes posterior variance.

Coarse time schedules and skipping steps. A central benefit of (2.6) is that it naturally supports *time skipping*. Let

$$T = t_K > t_{K-1} > \cdots > t_0 = 0$$

be a coarse schedule (with $K \ll T$). We can jump from t_i directly to t_{i-1} by defining $\sigma_{t_{i-1}}^2 \in [0,\, 1-\bar{\alpha}_{t_{i-1}}]$ and updating

$$X_{t_{i-1}} = \sqrt{\bar{\alpha}_{t_{i-1}}}\widehat{X}_0 + \sqrt{1-\bar{\alpha}_{t_{i-1}}-\sigma_{t_{i-1}}^2}\;\epsilon_\theta(X_{t_i}, t_i) + \sigma_{t_{i-1}}Z, \qquad Z \sim \mathsf{N}(0, I).$$

This is the standard DDIM "update rule" used in implementations: all dependence on the time grid enters through $\bar{\alpha}_{t_i}$ and $\bar{\alpha}_{t_{i-1}}$, and the same trained denoiser is reused at the visited times t_i. The following algorithm summarizes such a sampling procedure.

Algorithm 1 DDIM Sampling on a Coarse Time Schedule

Require: Number of DDIM steps K; trained noise predictor $\epsilon_\theta(\cdot,\cdot)$; schedule $\{t_i\}$; noise levels $\{\bar{\alpha}_{t_i}\}$; stochasticity $\{\sigma_{t_i}^2\}$ with $0 \le \sigma_{t_i}^2 \le 1 - \bar{\alpha}_{t_i}$.

1: Sample $X_{t_K} \sim \mathsf{N}(0, I)$.
2: **for** $i = K, \dots, 1$ **do**
3: $\quad \widehat{X}_0 \leftarrow \dfrac{X_{t_i} - \sqrt{1-\bar{\alpha}_{t_i}}\,\epsilon_\theta(X_{t_i}, t_i)}{\sqrt{\bar{\alpha}_{t_i}}}$.
4: $\quad$ Sample $Z \sim \mathsf{N}(0, I)$.
5: $\quad X_{t_{i-1}} \leftarrow \sqrt{\bar{\alpha}_{t_{i-1}}}\widehat{X}_0 + \sqrt{1 - \bar{\alpha}_{t_{i-1}} - \sigma_{t_{i-1}}^2}\,\epsilon_\theta(X_{t_i}, t_i) + \sigma_{t_{i-1}} Z$.
6: **end for**
7: **return** X_{t_0}.

3 Consistency Models

Note: Consistency models were proposed in [1] based on continuous-time formulations of diffusion models. Therefore, the following introduction would be slightly different from the original writing. However, the key concept stays the same.

DDIM enables significant speedups, but it still requires iterating through a (coarse) decoder chain

$$X_{t_K} \to X_{t_{K-1}} \to \cdots \to X_{t_0}.$$

Consistency models take a more aggressive route: instead of only compressing the *sampling procedure*, they compress the *denoising map itself*. The goal is to learn a family of functions that directly map a noisy state at time t to a less noisy (or even clean) state, and these maps should be *consistent across times*. Figure 1 demonstrates the concept of consistency. More precisely, the

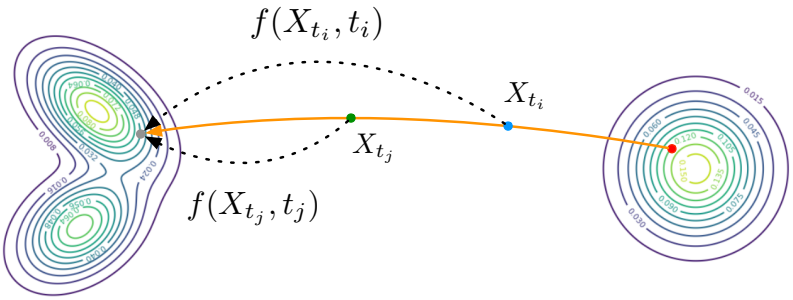

Figure 1: Consistency model that predicts the clean data at different time steps. The prediction result (output of f function) should be identical within the same trajectory.

design principle of consistency model is:

> *The output of function f at a higher noise level should agree with the denoising map at a lower noise level, when both are applied to states lying on the same trajectory.*

Consistency models enforce this type of self-consistency.

3.1 Consistency Functions and Consistency Conditions

Although it is possible to enforce consistency on stochastic decoders, it is much more convenient to impose it on the *deterministic* DDIM decoder. Deterministic decoder chains make the notion of a "sample trajectory" unambiguous: each state at time t_i deterministically maps to a unique descendant at any earlier time.

A consistency model parameterizes a family of functions

$$f_\phi(\cdot, t) : \mathbb{R}^{d+1} \to \mathbb{R}^d,$$

intended to output an estimate of the clean sample X_0 from a noisy input at time t_i:

$$f_\phi(X_{t_i}, t_i) \approx X_0.$$

The key requirement is that these predictions should be *consistent* along the same decoder trajectory. Let $t_j < t_i$ and define X_{t_j} as the state obtained by running the deterministic DDIM decoder from X_{t_i} down to time t_j (Algorithm 1). Then the **consistency condition** is

$$f_\phi(X_{t_i}, t_i) = f_\phi(X_{t_j}, t_j), \quad \text{whenever } X_{t_j} \text{ is generated from } X_{t_i} \text{ along the same trajectory.} \tag{3.1}$$

We also impose a **boundary condition** at the clean endpoint:

$$f_\phi(X_0, 0) = X_0.$$

If f_ϕ satisfies (3.1), then the output $f_\phi(X_{t_i}, t_i)$ is already aligned with what the model would output after several intermediate denoising steps. One can attempt to sample by evaluating f_ϕ only once (or a few times) for sample generation, rather than iterating a long decoder chain.

3.2 Training via Distillation

We translate (3.1) into a practical training loss. Fix adjacent times $t_i > t_{i-1}$. Given X_{t_i}, let $X_{t_{i-1}}$ be obtained by deterministic DDIM steps from t_i to t_{i-1}. The basic consistency loss is

$$\ell(\phi; t_i, X_{t_i}) = \left\| f_\phi(X_{t_i}, t_i) - f_\phi(X_{t_{i-1}}, t_{i-1}) \right\|_2^2. \tag{3.2}$$

Taking expectation over t_i (uniformly over $\{t_1, \ldots, t_K\}$) and over X_{t_i} yields

$$\mathcal{L}(\phi) = \mathbb{E}_{t_i, X_{t_i}} \left[\left\| f_\phi(X_{t_i}, t_i) - f_\phi(X_{t_{i-1}}, t_{i-1}) \right\|_2^2 \right], \tag{3.3}$$

where X_{t_i} can be sampled from the forward marginal $X_{t_i} = \sqrt{\bar{\alpha}_{t_i}}\, X_0 + \sqrt{1 - \bar{\alpha}_{t_i}}\, Z$ with $Z \sim \mathsf{N}(0, I)$.

Self-distillation. In practice, one often uses a teacher-student variant to stabilize training by treating the prediction at lower noise as a fixed target. Concretely, let ϕ^- denote a *teacher* parameter (e.g., an exponential moving average of ϕ). The distillation objective is

$$\mathcal{L}_{\text{distill}}(\phi, \phi^-) = \mathbb{E}_{t_i, X_{t_i}} \left[\left\| f_\phi(X_{t_i}, t_i) - f_{\phi^-}(X_{t_{i-1}}, t_{i-1}) \right\|_2^2 \right], \tag{3.4}$$

where gradients are taken only with respect to ϕ. Applying a stochastic gradient algorithm to (3.4) yields the following algorithm.

Algorithm 2 Training a Consistency Model via Self-Distillation

Input: Rate $\rho \in (0,1)$; schedules $\{\alpha_t\}$; learning rate η; initialization ϕ_1 and ϕ^-.
while not convergent **do**
  Sample data $X_0 \sim P_{\text{data}}$ and noise $Z \sim \mathsf{N}(0, I)$
  Sample time index $i \sim \mathsf{Unif}\{1, \dots, K\}$ and set $t \leftarrow t_i$, $t' \leftarrow t_{i-1}$
  Compute

$$X_t \leftarrow \sqrt{\bar{\alpha}_t} X_0 + \sqrt{1 - \bar{\alpha}_t} Z.$$

  Follow deterministic DDIM to generate $X_{t'}$.
  Compute distillation loss:

$$\mathcal{L}_{\text{distill}}(\phi, \phi^-) \leftarrow \left\| f_\phi(X_t, t) - \text{stopgrad}\big(f_{\phi^-}(X_{t'}, t')\big) \right\|_2^2.$$

  Update student parameter (e.g., SGD/Adam):

$$\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_{\text{distill}}(\phi, \phi^-).$$

  Update teacher by exponential moving average:

$$\phi^- \leftarrow \rho \phi^- + (1 - \rho)\phi$$

end while
Output: f_ϕ.

Consistency training without pre-trained DDIM. The distillation formulation above benefits from a pre-trained diffusion/DDIM model, which provides (i) a reliable low-noise target and (ii) a concrete notion of a "trajectory" via the DDIM decoder map. A natural and important question is whether this reliance is fundamentally necessary:

> *Can we train a consistency model f_ϕ* ***without*** *using any pre-trained diffusion/DDIM model, while still obtaining a well-defined objective and a practical sampling procedure?*

At a conceptual level, the consistency condition (3.1) only asks for cross-time agreement of f_ϕ along appropriate paths, together with the boundary condition to anchor the map. However, once the pre-trained model is removed, one must decide (a) what distribution over paired states (X_{t_i}, X_{t_j}) should define "the same trajectory," and (b) how to prevent degenerate solutions while keeping the training objective identifiable.

This is left as an exercise: Propose one or more training objectives for consistency models without the usage of a pre-trained model, and argue why they should (i) enforce nontrivial cross-time agreement, (ii) be anchored by the boundary condition, and (iii) yield a usable sampler at inference time. Discuss what additional assumptions (if any) would be needed.

3.3 Sample Generation

Once f_ϕ is trained, sampling becomes simple. We start from $X_T \sim \mathsf{N}(0, I)$ and generate a new sample by

$$\widehat{X}_0 = f_\phi(X_T, T).$$

This is the one-step sampler. More generally, one may implement a few more steps by adding noise to $\widehat{X}_0$ and re-apply the consistency function f_ϕ.

References

[1] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.