

Controllable Generative AI for Biological Sequences

Jan Olszewski, Ewa Szczurek

June 19, 2026

1 Mathematical Background

1.1 Continuous random variables, densities, and Bayes' rule

A continuous random variable X taking values in $\mathbb{R}^d$ is described by a probability density function $p : \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$ satisfying $\int_{\mathbb{R}^d} p(x) dx = 1$.

Unlike the discrete case, $p(x)$ is not a probability. It can be larger than one. The probability of an event is recovered by integration over a set:

$$\Pr(X \in A) = \int_A p(x) dx.$$

Intuitively, $p(x)$ measures the relative likelihood of X being near the point x : $\Pr(X \in [x, x + dx]) \approx p(x) dx$ for small dx .

Joint, marginal, conditional. For two random variables X, Y with joint density $p(x, y)$:

- Marginal: $p(x) = \int p(x, y) dy$.
- Product rule: $p(x, y) = p(y|x)p(x)$.
- Conditional: $p(x | y) = p(x, y)/p(y)$, defined where $p(y) > 0$.

Bayes' rule for densities. The identity known from the discrete setting carries over to densities:

$$p(x | y) = \frac{p(y | x) p(x)}{p(y)}.$$

We will use this constantly throughout the course. In particular, taking logarithms on both sides and differentiating with respect to x :

$$\nabla_x \log p(x | y) = \nabla_x \log p(y | x) + \nabla_x \log p(x),$$

since $\nabla_x \log p(y) = 0$ (the marginal $p(y)$ does not depend on x). Taking x as a data point, and y as its property/label, this identity gives a recipe for sampling from $p(x | y)$ using a model of $p(x)$ and a classifier $p(y | x)$.

Expectations. For any function f :

$$\mathbb{E}[f(X)] = \int f(x) p(x) dx, \quad \mathbb{E}[f(X) | Y = y] = \int f(x) p(x | y) dx.$$

1.2 Gaussian algebra

The Gaussian density on $\mathbb{R}^d$ with mean μ and covariance Σ is

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right).$$

Three facts about Gaussians are useful for this course.

Fact 1 (Linear transformation). If $X \sim \mathcal{N}(\mu, \Sigma)$ and $a \in \mathbb{R}$, $b \in \mathbb{R}^d$ are constants, then

$$aX + b \sim \mathcal{N}(a\mu + b, a^2\Sigma).$$

Fact 2 (Sum of independent Gaussians). Means add and variances add when the variables are independent. If $X \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $Y \sim \mathcal{N}(\mu_2, \Sigma_2)$ are independent, then

$$X + Y \sim \mathcal{N}(\mu_1 + \mu_2, \Sigma_1 + \Sigma_2).$$

Fact 3 (Reparameterization). The identity

$$X \sim \mathcal{N}(\mu, \sigma^2 I) \iff X = \mu + \sigma \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

lets us write a Gaussian random variable as a deterministic function of a standard noise variable. The benefit is twofold. First, we can pull randomness “outside” the variable of interest. Second, gradients flow through the deterministic function.

1.3 KL divergence and the Gaussian case

To train a probabilistic model, we need a way to measure how close two distributions are. The Kullback-Leibler (KL) divergence from q to p is

$$\text{KL}(q \| p) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_{X \sim q} \left[\log \frac{q(X)}{p(X)} \right].$$

Three properties to keep in mind:

- *Non-negative.* $\text{KL}(q \| p) \geq 0$, with equality if and only if $q = p$ almost everywhere. If you are keen to see the proof, refer to Appendix A.
- *Asymmetric.* $\text{KL}(q \| p) \neq \text{KL}(p \| q)$ in general. It is a divergence, not a metric.
- *Loss-like.* Minimizing $\text{KL}(q \| p_\theta)$ over θ when q is the data distribution recovers maximum likelihood. So KL is the natural training signal for probabilistic models.

The Gaussian case. For two Gaussians with the same covariance, $p = \mathcal{N}(\mu_1, \sigma^2 I)$ and $q = \mathcal{N}(\mu_2, \sigma^2 I)$:

$$\text{KL}(p \| q) = \frac{1}{2\sigma^2} \|\mu_1 - \mu_2\|^2.$$

The KL collapses to a scaled squared error between the means. See Appendix B for the proof.

1.4 The score function

For a density $p(x)$ on $\mathbb{R}^d$, the *score function* is the gradient of the log density:

$$s(x) := \nabla_x \log p(x) \in \mathbb{R}^d.$$

The score is an important mathematical object of this course. It is worth investing in three pieces of intuition about the score now.

Geometric meaning. The score points in the direction of steepest increase in log-density. Following the score from any starting point means moving toward regions where the data is more likely, toward modes of p and away from low-density regions. This is exactly the direction in which a generative model wants to push samples. Sampling algorithms based on the score work by following this gradient, with stochasticity added to prevent collapse onto a single mode.

Invariance to normalization. Suppose we only know p up to a constant: $p(x) = \tilde{p}(x)/Z$ where $Z = \int \tilde{p}(x) dx$ is an intractable normalizer. Then

$$\nabla_x \log p(x) = \nabla_x [\log \tilde{p}(x) - \log Z] = \nabla_x \log \tilde{p}(x),$$

because Z is a constant in x . The score does not depend on the normalizer. This is a huge practical advantage: we can work with unnormalized distributions and still compute scores exactly.

Why this matters for Bayesian guidance. The Bayes identity from above,

$$p(x | y) = \frac{p(y | x)p(x)}{p(y)},$$

has an intractable normalizer $p(y) = \int p(y | x)p(x) dx$ for any non-trivial likelihood. Taking the score wipes it out:

$$\nabla_x \log p(x | y) = \nabla_x \log p(y | x) + \nabla_x \log p(x).$$

The right-hand side consists of two things we can actually compute: the gradient of a classifier $\log p(y | x)$ and the score of the data distribution $\log p(x)$.

2 Continuous Diffusion models

2.1 Diffusion: the intuitive picture

A diffusion model [Soh+15; HJA20] generates samples in two phases, conceptually run in opposite directions:

- **Forward process (noising).** Start from a data sample $x \sim p_{\text{data}}$. Add a small amount of Gaussian noise. Repeat for T steps, yielding intermediate z_t at each step $t = 1, 2, \dots, T$. After enough steps, all structure is destroyed and what remains is indistinguishable from pure noise.
- **Reverse process (denoising).** Start from pure noise $z_T \sim \mathcal{N}(0, I)$. Remove a small amount of noise. Repeat for T steps. If we’ve learned to do this well, we end up with a sample that looks like it came from p_{data} .

The forward process is fixed and requires no learning. We just add noise according to a schedule we choose. The reverse process is what the neural network must learn. The whole machinery of diffusion is about how to make this learning tractable.

There is a tension that arises at sampling time: we don’t have access to x , only to the noisy z_t . Resolving this tension is the core trick.

2.2 The forward diffusion process

We define the forward process as a Markov chain that gradually corrupts x over T steps. At each step, we add Gaussian noise according to a fixed *variance schedule* $\{\beta_t\}_{t=1}^T$, with $\beta_t \in (0, 1)$ small:

$$q(z_t | z_{t-1}) = \mathcal{N}\left(z_t; \sqrt{1 - \beta_t} z_{t-1}, \beta_t I\right), \quad z_0 := x.$$

At each step we shrink the previous state by $\sqrt{1 - \beta_t}$ and add fresh Gaussian noise of variance β_t . The shrinkage is what prevents the variance from exploding as the chain runs.

Schedules. The original Denoising Diffusion Probabilistic Models (DDPM) paper of Ho et al. (2020) [HJA20] used a linear schedule with $\beta_t \in [10^{-4}, 0.02]$, meaning β_t increases linearly from 10^{-4} at $t = 1$ to 0.02 at $t = T$. A common improvement is the cosine schedule of Nichol and Dhariwal (2021), which adds less noise early on and produces sharper samples for the same T . The schedule is a hyperparameter that affects sample quality but not the underlying theory.

Why T is large. Typical choices are $T = 1000$ steps. Many small steps make each individual denoising decision easy: predicting z_{t-1} from z_t is a small correction, not a leap from noise to clean data.

2.3 Closed-form marginal of the forward process

Iterating the forward step T times naively would be slow. But because each step is a linear Gaussian, we can compose them analytically. Define

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

The notation is the standard DDPM convention: α_t is the per-step signal-retention factor, and $\bar{\alpha}_t$ is its cumulative product. By iteratively applying Fact 1 (each per-step shrinkage scales the variance) and Fact 2 (independent noise injections at different steps sum), the marginal distribution of z_t given x has the closed form

$$q(z_t | x) = \mathcal{N}(z_t; \sqrt{\bar{\alpha}_t} x, (1 - \bar{\alpha}_t)I).$$

See Appendix C for the proof. Equivalently, via the reparameterization trick (Fact 3), we can write z_t as a deterministic function of x and a single noise draw:

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

This is the workhorse equation of diffusion. It says we can jump to any noise level in one step, without simulating the chain.

Interpretation. The coefficient $\sqrt{\bar{\alpha}_t}$ controls how much of the original signal remains; $\sqrt{1 - \bar{\alpha}_t}$ controls how much pure noise has been mixed in. At $t = 0$, $\bar{\alpha}_0 = 1$ (pure signal). At $t = T$ with a properly designed schedule, $\bar{\alpha}_T \approx 0$, so z_T is essentially $\mathcal{N}(0, I)$. This justifies starting the reverse process from a standard Gaussian.

Notation warning. Different papers use α_t to mean different things. The convention used here (Ho et al. 2020) [HJA20] is $\alpha_t = 1 - \beta_t$ for the per-step factor and $\bar{\alpha}_t$ for the cumulative product. Some papers overload α_t to mean the cumulative product directly.

2.4 The reverse process

Generation requires running the chain backwards: sample $z_T \sim \mathcal{N}(0, I)$, then iteratively sample $z_{t-1} \sim q(z_{t-1} | z_t)$ until reaching $z_0 = x$. The obstacle: $q(z_{t-1} | z_t)$ is intractable. By Bayes' rule,

$$q(z_{t-1} | z_t) = \frac{q(z_t | z_{t-1}) q(z_{t-1})}{q(z_t)},$$

and the marginals $q(z_{t-1})$ and $q(z_t)$ involve integrating over the entire data distribution, which we do not have access to in closed form.

The crucial observation. If we additionally condition on the clean data x , the reverse step becomes tractable. The intractable marginals $q(z_{t-1})$ and $q(z_t)$ get replaced by the closed-form Gaussians $q(z_{t-1} | x)$ and $q(z_t | x)$ from the previous subsection. Bayes' rule applied to two Gaussians gives another Gaussian with mean and variance computable in closed form:

$$q(z_{t-1} | z_t, x) = \mathcal{N}(z_{t-1}; \tilde{\mu}_t(z_t, x), \tilde{\beta}_t I),$$

where

$$\tilde{\mu}_t(z_t, x) = \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} x + \frac{\sqrt{\alpha_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} z_t, \quad \tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t.$$

The mean is a convex combination of x and z_t : the optimal denoised state is somewhere between the noisy observation and the clean target. The variance $\tilde{\beta}_t$ is a deterministic function of the schedule. The detailed Bayes-on-Gaussians calculation is given in Ho et al. (2020) [HJA20].

2.5 The training trick

The chain is reversible *if we know x* . At sampling time we do not. The strategy is:

1. Train a neural network that takes the noisy state z_t and the timestep t as input, and predicts what the original clean data was.
2. At sampling time, substitute this prediction into the closed-form posterior mean $\tilde{\mu}_t$.
3. Sample from the resulting Gaussian and repeat.

Each reverse step boils down to: guess what x was from z_t , then take a small step from z_t toward that guess, with the step size and noise level controlled by the schedule.

Why this works. Even though the prediction at high noise levels is poor (most of the information about x has been destroyed), it does not need to be perfect. The reverse step only takes a small step in the direction of the prediction, and by the time we reach low t , the prediction is sharp and accurate. The job is spread over many small denoising decisions, each easier than going from pure noise to x in a single shot.

This decomposition, never modeling the reverse chain directly but always going through "predict, substitute, step," is what makes diffusion training tractable. The remaining question is what to train the prediction network on.

2.6 The training loss

Given the training trick from the previous subsection, the natural loss is the squared error between the network's prediction $\hat{x}_\theta(z_t, t)$ and the true clean data x :

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim p_{\text{data}}, t \sim \mathcal{U}\{1, \dots, T\}, \epsilon \sim \mathcal{N}(0, I)} [\|\hat{x}_\theta(z_t, t) - x\|^2], \quad z_t = \sqrt{\bar{\alpha}_t} x + \sqrt{1 - \bar{\alpha}_t} \epsilon.$$

In words: sample a clean data point, sample a random timestep, noise the data point to that timestep with a known noise vector ϵ , and ask the network to predict the original x . The training is ordinary supervised learning: one forward pass per training example, with x as the target and (z_t, t) as the input. The expectation is approximated by Monte Carlo over a minibatch.

This loss arises from the variational lower bound on $\log p_\theta(x)$, after expanding the KL divergences between the true reverse posterior and the learned reverse step into squared errors in the means (using the Gaussian KL formula) and dropping the time-dependent weights. The full derivation is in Ho et al. (2020) [HJA20]; we omit it.

2.7 Equivalent parameterizations

The network we have been calling $\hat{x}_\theta(z_t, t)$ is one of three equivalent parameterizations. Each can be computed from any other (refer to Appendix D for derivation); the choice is a matter of convention and affects training dynamics or numerical conditioning, but not the underlying model.

Clean-data prediction $\hat{x}_\theta(z_t, t)$. What we have used so far. Predicts the original clean data from a noisy state. Sometimes called the *Tweedie estimate*, since it is the posterior mean of x given z_t under the forward noising kernel.

Noise prediction $\hat{\epsilon}_\theta(z_t, t)$. Predicts the noise added to produce z_t :

$$\hat{\epsilon}_\theta(z_t, t) = \frac{z_t - \sqrt{\bar{\alpha}_t} \hat{x}_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}_t}}.$$

Introduced by Ho et al. (2020) [HJA20], this is used in DDPM.

Score prediction $s_\theta(z_t, t)$. Predicts the score of the marginal p_t :

$$s_\theta(z_t, t) = -\frac{\hat{\epsilon}_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \approx \nabla_{z_t} \log p_t(z_t).$$

This is the score-based view of diffusion (Song and Ermon 2019, Song et al. 2021b), and is used by those guidance methods which modify the score.

2.8 The denoiser network

The function $\hat{x}_\theta(z_t, t)$ (or equivalently $\hat{\epsilon}_\theta$ or s_θ) is a single neural network with two inputs: the noisy latent z_t (same shape as the data) and the integer timestep t . The output has the same shape as z_t .

Time conditioning. The scalar t is embedded into a vector, usually via sinusoidal positional embeddings followed by a small MLP. This embedding is injected into the network at every layer, typically by adding to intermediate activations or via adaptive normalization layers.

Architecture. The dominant choice for image data is the U-Net: a convolutional encoder-decoder with skip connections between matched resolution levels. For sequence data (text, protein sequences, peptides), Transformers are the standard choice; 1D U-Nets are also used. The architectural choice does not change the theory; it only changes inductive bias and scaling behaviour.

A single trained network handles every noise level. There is no separate model per timestep.

2.9 Sampling algorithm

At inference time we generate a new sample by running the reverse process. Combining the training trick with a trained clean-data predictor $\hat{x}_\theta$ gives the standard DDPM ancestral sampling algorithm.

Ancestral sampling.

1. Sample $z_T \sim \mathcal{N}(0, I)$.
2. For $t = T, T-1, \dots, 1$:
 - (a) Compute $\hat{x}_\theta(z_t, t)$ (one forward pass through the network).
 - (b) Compute the posterior mean $\mu_\theta(z_t, t) = \tilde{\mu}_t(z_t, \hat{x}_\theta(z_t, t))$ using the closed-form expression from the reverse-posterior subsection.
 - (c) Sample $z_{t-1} \sim \mathcal{N}(\mu_\theta(z_t, t), \tilde{\beta}_t I)$. For $t = 1$, return the mean without adding noise.
3. Return z_0 as the generated sample.

This is T forward passes through the network, one per timestep. For $T = 1000$, this is the dominant cost of sampling: about three orders of magnitude slower than a single-pass generative model like a VAE or a GAN.

Faster samplers. A line of work reduces the number of denoising steps without retraining the network. Denoising Diffusion Implicit Models (DDIM; Song et al. 2021a) [SME21] reformulates the reverse process as deterministic, allowing 50 steps to match 1000-step DDPM quality. DPM-Solver (Lu et al. 2022) [Lu+22] and related higher-order ODE solvers push this down to 10-20 steps. All of these use the same trained network; only the sampling loop changes. We will not develop these alternatives further.

3 The continuous-time description of diffusion models

3.1 Ordinary differential equations and flows

An *ordinary differential equation* (ODE) describes a deterministic trajectory $X_t \in \mathbb{R}^d$ driven by a velocity field $u : \mathbb{R}^d \times [0, 1] \rightarrow \mathbb{R}^d$:

$$\frac{d}{dt}X_t = u_t(X_t), \quad X_0 = x_0.$$

- The map $x_0 \mapsto X_t$ that sends a start point to its position at time t is the *flow*. Velocity field, ODE, and flow are three descriptions of the same object.
- Under mild conditions (continuously differentiable u with bounded derivative) a unique solution exists. Neural-network velocities always satisfy this, so existence is not a concern in practice.
- Solutions are rarely available in closed form, so we discretize. The *Euler method* with step size h is

$$X_{t+h} = X_t + h u_t(X_t).$$

3.2 Brownian motion

A (standard) *Brownian motion* or *Wiener process* $W_t \in \mathbb{R}^d$ is the basic source of continuous randomness, and the ingredient that turns an ODE into a stochastic one. It is defined by:

- $W_0 = 0$, with continuous trajectories $t \mapsto W_t$.
- *Independent increments*: non-overlapping increments are independent.

- *Gaussian increments*: $W_t - W_s \sim \mathcal{N}(0, (t-s)I)$ for $s < t$. Variance grows linearly in elapsed time.

It is simulated by accumulating fresh Gaussian noise of the right scale:

$$W_{t+h} = W_t + \sqrt{h} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

The $\sqrt{h}$ factor is exactly what makes the variance of the increment equal to hI (Fact 1 from the Gaussian algebra subsection).

3.3 Stochastic differential equations

A *stochastic differential equation* (SDE) adds Brownian noise to an ODE:

$$dX_t = \underbrace{f_t(X_t)}_{\text{drift}} dt + \underbrace{g_t}_{\text{diffusion}} dW_t.$$

- The “ dX_t ” notation is shorthand for the small-step update $X_{t+h} - X_t \approx h f_t(X_t) + \sqrt{h} g_t \epsilon$.
- *Euler-Maruyama* is the SDE analogue of the Euler method:

$$X_{t+h} = X_t + h f_t(X_t) + \sqrt{h} g_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

- Unlike an ODE, the start point X_0 no longer determines X_t : every simulation produces a different random path.
- An ODE is the special case $g_t = 0$.

3.4 Worked example: the Ornstein-Uhlenbeck process (the forward diffusion SDE)

Take a linear drift $f_t(x) = -\frac{1}{2}\beta(t)x$ and diffusion $g_t = \sqrt{\beta(t)}$ for a schedule $\beta(t) > 0$:

$$dX_t = -\frac{1}{2}\beta(t) X_t dt + \sqrt{\beta(t)} dW_t.$$

- The drift pulls X_t toward the origin; the diffusion injects variance. The two balance.
- Started at $X_0 = x$, the marginal is Gaussian at every time, with a mean and variance computable in closed form (iterating Facts 1 and 2):

$$X_t \sim \mathcal{N}\left(\sqrt{\bar{\alpha}(t)} x, (1 - \bar{\alpha}(t))I\right), \quad \bar{\alpha}(t) = \exp\left(-\int_0^t \beta(s) ds\right).$$

- As t grows, $\bar{\alpha}(t) \rightarrow 0$ and $X_t \rightarrow \mathcal{N}(0, I)$, regardless of x .

This is exactly the forward noising process of a diffusion model, written in continuous time. We make this precise below.

	ODE	SDE
Form	$dX_t = f_t(X_t) dt$	$dX_t = f_t(X_t) dt + g_t dW_t$
Path	deterministic	random
X_0 fixes X_t ?	yes	no
Discretization	Euler	Euler-Maruyama

Table 1: ODEs and SDEs side by side. An ODE is the noiseless special case.

3.5 The DDPM chain is an Euler-Maruyama discretization

Recall the per-step forward update $z_t = \sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \epsilon$. Let the schedule shrink with the step size: set $\beta_t = \beta(t) \Delta t$ with $\Delta t = 1/T$ small. A first-order expansion gives $\sqrt{1 - \beta(t) \Delta t} \approx 1 - \frac{1}{2} \beta(t) \Delta t$, so

$$z_t - z_{t-1} \approx -\frac{1}{2} \beta(t) \Delta t z_{t-1} + \sqrt{\beta(t) \Delta t} \epsilon.$$

This is one Euler-Maruyama step of the SDE

$$dz_t = -\frac{1}{2} \beta(t) z_t dt + \sqrt{\beta(t)} dW_t.$$

- This is the *variance-preserving* (VP) forward SDE, the Ornstein-Uhlenbeck process from the background subsection.
- As $T \rightarrow \infty$ (equivalently $\Delta t \rightarrow 0$), the discrete chain converges to this continuous forward process. The discrete chain is one discretization of it.

3.6 The marginals match the closed-form formula

The forward SDE has the same closed-form marginal already derived for the chain:

$$q(z_t | x) = \mathcal{N}\left(z_t; \sqrt{\bar{\alpha}(t)} x, (1 - \bar{\alpha}(t)) I\right), \quad \bar{\alpha}(t) = \exp\left(-\int_0^t \beta(s) ds\right).$$

- The continuous signal-retention factor $\bar{\alpha}(t)$ is the limit of the cumulative product $\bar{\alpha}_t = \prod_s \alpha_s$.
- Training never simulates the SDE. We jump to any noise level in one step by sampling z_t directly from this Gaussian via reparameterization, exactly as in the discrete case.

Discrete chain	Forward SDE
$\beta_t \in (0, 1)$ per step	schedule $\beta(t)$
$\alpha_t = 1 - \beta_t$	—
$\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$	$\bar{\alpha}(t) = e^{-\int_0^t \beta(s) ds}$
$z_t = \sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \epsilon$	$dz_t = -\frac{1}{2} \beta(t) z_t dt + \sqrt{\beta(t)} dW_t$
$q(z_t x) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x, (1 - \bar{\alpha}_t) I)$	$q(z_t x) = \mathcal{N}(\sqrt{\bar{\alpha}(t)} x, (1 - \bar{\alpha}(t)) I)$

Table 2: Discrete to continuous dictionary for the forward process.

3.7 Generation as a reverse-time SDE

Generation runs the forward SDE backwards in time, from noise at $t = T$ to data at $t = 0$. A classical result makes the reverse dynamics precise and, strikingly, requires only one new ingredient: the score of the marginals, which our trained network already provides.

3.8 Anderson’s time-reversal

Anderson (1982) shows that the time-reversal of a forward SDE is again an SDE, with the drift corrected by the score $\nabla_{z_t} \log p_t(z_t)$. For the VP forward SDE, running t from T (noise) down to 0 (data):

$$dz_t = \left[-\frac{1}{2} \beta(t) z_t - \beta(t) \nabla_{z_t} \log p_t(z_t) \right] dt + \sqrt{\beta(t)} d\bar{W}_t,$$

where p_t is the marginal density of z_t under the forward process and $\bar{W}_t$ is a Brownian motion in reverse time.

- The only unknown is the score $\nabla_{z_t} \log p_t(z_t)$.
- Everything else is the fixed schedule $\beta(t)$.

3.9 The score is what we already trained

The clean-data predictor gives the score directly, using the conversions from the equivalent-parameterizations subsection:

$$\nabla_{z_t} \log p_t(z_t) \approx s_\theta(z_t, t) = -\frac{\hat{\epsilon}_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}(t)}} = -\frac{z_t - \sqrt{\bar{\alpha}(t)} \hat{x}_\theta(z_t, t)}{1 - \bar{\alpha}(t)}.$$

- The same network drives the reverse SDE. No new model and no retraining are required.
- Plugging s_θ into Anderson’s drift recovers a sampler in continuous time.

3.10 The probability flow ODE

The same forward marginals p_t are reproduced by a deterministic ODE that carries half the score term and no noise:

$$\frac{d}{dt} z_t = -\frac{1}{2}\beta(t) z_t - \frac{1}{2}\beta(t) \nabla_{z_t} \log p_t(z_t).$$

- This is the *probability flow ODE* (PF-ODE).
- It has the same marginal p_t as the reverse SDE at every t , hence the same endpoint distribution $p_0 = p_{\text{data}}$.
- It is deterministic: the noise z_T fixes the output z_0 . This enables off-the-shelf ODE solvers and exact likelihood computation.

3.11 Sampling: two discretizations of the same object

- **Reverse SDE.** Discretize with Euler-Maruyama. The DDPM ancestral sampler from the previous section is one such discretization.
- **PF-ODE.** Discretize with an ODE solver. DDIM (Song et al., 2021a) [SME21] is the first-order case; higher-order solvers (for example DPM-Solver) push the step count to roughly 10 to 50 while reusing the same trained network.

	Reverse SDE	PF-ODE
Dynamics	stochastic	deterministic
z_T fixes z_0 ?	no	yes
Discretization	Euler-Maruyama	ODE solver
Example sampler	DDPM ancestral	DDIM, DPM-Solver
Typical steps	~ 1000	~ 10 to 50
Exact likelihood	no	yes

Table 3: The reverse SDE and the PF-ODE share marginals but differ in randomness, cost, and what they enable.

3.12 Discrete time versus continuous time

The discrete chain and the SDE are two views of the same model, not competing methods. It is worth stating plainly what is shared and what differs.

What is shared.

- *Training loss.* The clean-data (or noise-prediction) squared error is, term for term, denoising score matching. Both views train the identical network on the identical objective.
- *Forward marginals.* Both sample z_t from the same Gaussian $q(z_t | x)$, so the data seen during training is the same.

What differs.

- *Time grid.* The chain fixes a discretization before training; the SDE leaves it open, so the sampler step count is chosen after training.
- *Objective tightness.* In continuous time the variational bound on $\log p_\theta(x)$ becomes tight; in discrete time it remains a lower bound.
- *Sampling control.* The SDE view exposes the reverse SDE and the PF-ODE as separate knobs, giving the speed and likelihood options in the table above.

	Discrete chain	Continuous SDE / ODE
Time	fixed grid $t = 1, \dots, T$	continuous $t \in [0, 1]$
Forward step	$\sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \epsilon$	$dz_t = -\frac{1}{2}\beta(t)z_t dt + \sqrt{\beta(t)} dW_t$
Reverse step	ancestral Gaussian	reverse SDE or PF-ODE
Loss	MSE (a bound)	MSE (tight)
Trained network	$\hat{x}_\theta, \hat{\epsilon}_\theta, s_\theta$	same

Table 4: Discrete and continuous views of the same diffusion model.

4 The idealism-realism problem

- The generative model $p_\theta(x) \approx p(x)$, also called the *prior* model, is trained to sample from a data distribution $p(x)$. $p(x)$ models **realism**, and gives us samples that look like plausible data: any peptide, any small molecule, any image.
- In practice, we often want samples to have a high value of some *reward function* $r(x) \geq 0$ that scores how desirable each output is. The reward models **idealism** and can be a learned classifier output, a regression model, an energy function, a simulator output, or any black-box scorer. We make no smoothness assumptions about r . High reward examples: peptides with high antimicrobial activity, molecules that bind a target, images that match a text prompt.
- In probabilistic language, an example of a reward is the likelihood $p(y | x)$, the probability that x has some target property y . High $p(y | x)$ marks outputs that we prefer, so $\log p(y | x)$ is a natural choice of $r(x)$.

The question becomes: *how do we modify a generative model to favour high-reward outputs while keeping samples close to the data distribution it was trained on?*

4.1 KL-regularized optimization

A first thought is to maximize the reward directly: find π maximizing $\mathbb{E}_{x \sim \pi}[r(x)]$. This is degenerate. The optimum is a delta on the single highest-reward point, ignoring the prior entirely, and the highest-reward point may not even be realistic (r is only reliable on the data manifold).

The fix is to regularize π toward the prior using the KL divergence:

$$\pi^* = \arg \max_{\pi} \left[\mathbb{E}_{x \sim \pi}[r(x)] - \beta \text{KL}(\pi \| p_\theta) \right].$$

This is the *KL-regularized optimization problem*, equivalent to the KL-regularized reinforcement learning objective. The parameter $\beta > 0$ controls the trade-off between reward and proximity to the prior:

- Small β : aggressively optimize reward, risking off-manifold samples and reward hacking.
- Large β : stay close to the prior, ignoring the reward.

β is sometimes called the *inverse temperature*, borrowing from statistical physics. The objective makes sense even when r is non-differentiable, since the KL penalty acts on the policy π , not on r .

4.2 The Boltzmann-tilted target

The KL-regularized objective has a closed-form optimum. Treating π as a function and using a Lagrange multiplier for the normalization constraint, a straightforward calculation gives

$$\pi^*(x) = \frac{1}{Z} p_\theta(x) \exp\left(\frac{r(x)}{\beta}\right), \quad Z = \int p_\theta(x) \exp(r(x)/\beta) dx.$$

See Appendix E for the derivation. This is the *Boltzmann-tilted target*, also called the *tempered posterior* or the *tilted distribution*. In physics it is the Boltzmann distribution with energy $-r(x)$ at temperature β . Large β flattens the exponential toward the prior; small β sharpens it onto high-reward regions.

The optimal sampling distribution is the prior reweighted by the exponentiated reward. The reward and prior combine multiplicatively, so that neither has the last word. A sequence with high $r(x)$ but low $p_\theta(x)$ is suppressed by the prior, and a sequence in the bulk of p_θ with low r is suppressed by the reward.

Special cases.

- $r(x) = \log p(y | x)$ and $\beta = 1$: then $\pi^*(x) \propto p_\theta(x) p(y | x) = p_\theta(x | y)$ by Bayes' rule. The Boltzmann target is the Bayesian posterior. Classifier guidance, which we develop later, samples from this case.
- $r(x) = \log \mathbf{1}[x \in S]$ (i.e. $r = 0$ on S and $-\infty$ off it) then $\exp(r(x)/\beta) = \mathbf{1}[x \in S]$ and $\pi^* \propto p_\theta(x) \mathbf{1}[x \in S]$ concentrates on $S \cap \text{supp}(p_\theta)$, the realistic outputs satisfying a hard constraint. The same framework covers hard constraints and soft rewards.

4.3 The obstacle: the partition function

The Boltzmann target has a closed-form formula, so why isn't this the end of the story? The normalizing constant

$$Z = \int p_\theta(x) \exp(r(x)/\beta) dx$$

is intractable for any non-trivial r in high dimensions. We know π^* only up to the unknown Z , which means we cannot directly evaluate $\pi^*(x)$ at any point or sample from it.

4.4 The score-function modification idea

Although Z is intractable, the *score* of π^* does not depend on it:

$$\nabla_x \log \pi^*(x) = \nabla_x \log p_\theta(x) + \frac{1}{\beta} \nabla_x r(x).$$

The score of the tilted target decomposes additively into the *prior score* (which a generative model can learn) and the *reward gradient* (scaled by $1/\beta$). The reward signal enters as a correction to the prior dynamics.

This decomposition is important for the rest of the course. We now extend the decomposition from static distributions to the time-dependent distributions that diffusion models give us.

4.5 Guidance as a time-dependent score correction

A diffusion model gives us not a single distribution but a family p_t for $t \in [0, T]$. (Here $p_t(z_t) = \int q(z_t | x) p_{\text{data}}(x) dx$ is the forward marginal, the unconditional distribution of z_t , obtained by noising a data point to level t , and forgetting which point it came from. We do not have p_t in closed form, as the integral over p_{data} is intractable, but the trained network provides its score $\nabla_{z_t} \log p_t(z_t)$.) For this family, we need a corresponding family of tilted distributions q_t^* that the multi-step sampler should target.

The clean-data target. At $t = 0$, where the reward $r(z_0)$ is meaningful, the target is the same Boltzmann-tilted distribution from above:

$$q_0^*(z_0) \propto p_0(z_0) \exp(r(z_0)/\beta).$$

A sample from q_0^* is what we ultimately want.

The intermediate targets. At intermediate $t > 0$, z_t is a noisy version of the clean data and applying r to z_t does not make sense: a noised peptide has no measurable binding affinity. We need a target that captures how a noisy state z_t will become a high-reward clean output under the generative dynamics:

$$q_t^*(z_t) \propto p_t(z_t) u_t(z_t),$$

where $u_t(z_t)$ is a time-dependent function carrying the reward signal at noise level t .

4.6 The value function u_t

The function u_t takes a noisy state at time t and returns the expected exponentiated reward, averaged over all trajectories of the prior that pass through z_t and continue to clean data at $t = 0$:

$$u_t(z_t) := \mathbb{E}_P[\exp(r(Z_0)/\beta) \mid Z_t = z_t],$$

where P is the prior path measure (the joint distribution of trajectories $(Z_t)_{t \in [0, T]}$ induced by the diffusion model). u_t asks: “from this state at this time, what is the expected exponentiated reward if I let the prior dynamics run to completion?”

Boundary conditions. At $t = 0$ the conditional expectation collapses: $u_0(z_0) = \exp(r(z_0)/\beta)$. So at the clean-data boundary, u_t recovers the static Boltzmann tilt. At $t = T$, $u_T(z_T)$ averages over all completions of the prior starting from z_T .

Names for u_t . The same object appears in several literatures under different names. In reinforcement learning it is a *value function*. In stochastic optimal control it is the *desirability function*, the solution to a Hamilton-Jacobi-Bellman equation. In the theory of Schrödinger bridges it is the *Schrödinger potential*.

The score decomposition. Taking ∇_{z_t} of $\log q_t^*$:

$$\nabla_{z_t} \log q_t^*(z_t) = \nabla_{z_t} \log p_t(z_t) + \nabla_{z_t} \log u_t(z_t).$$

This is the multi-step analogue of the static score decomposition above. The score of the tilted target decomposes into the *prior score* (a generative model gives us this directly) and a time-dependent *guidance correction* $\nabla_{z_t} \log u_t$. At $t = 0$, $\log u_0 = r/\beta$, so we recover the static decomposition.

The remaining obstacle. u_t is not directly computable. It encodes the entire generative process from z_t back to clean data, and computing it exactly would require integrating over all completions of the trajectory. The guidance methods we develop later are different approximations to u_t or to its gradient $\nabla \log u_t$.

5 Guidance methods

5.1 Conditioning, the simplest mechanism

The goal is to sample from a conditional distribution $p(x \mid c)$ instead of the unconditional $p(x)$. The conditioning vector c encodes whatever we want to control: a class label, a text description, a target physicochemical property, an experimental constraint.

The mechanism is mechanically simple. Enlarge the input of the denoiser:

$$\hat{x}_\theta(z_t, t) \longrightarrow \hat{x}_\theta(z_t, t, c).$$

The training loss is the standard one with the conditioning vector threaded through:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x, c) \sim p_{\text{data}}, t, \epsilon} [\|\hat{x}_\theta(z_t, t, c) - x\|^2], \quad z_t = \sqrt{\bar{\alpha}_t} x + \sqrt{1 - \bar{\alpha}_t} \epsilon.$$

The training data are now (x, c) pairs. Sampling is identical to the unconditional algorithm with the desired c fed in at every step.

5.2 How the conditioning vector enters the network

There are three standard architectural mechanisms for injecting c into the denoiser. The choice does not change the loss or the sampling algorithm, only how the conditioning information is presented to the network.

Concatenation. Concatenate c (or a learned embedding $h(c)$) onto the input z_t , or broadcast it along spatial or sequence dimensions and concatenate with intermediate feature maps at multiple layers. Simple, robust, works well when c is a low-dimensional vector.

Cross-attention. Treat c as a sequence of tokens. At each Transformer block, perform cross-attention with the network’s hidden states as queries and the tokens of c as keys and values. Essential when c is itself a sequence, such as a tokenized text prompt. Used by Stable Diffusion and Imagen.

FiLM and adaptive normalization. Produce per-channel scale and shift parameters from c and use them to modulate normalization layers. Compact and expressive; used by ADM (Dhariwal and Nichol) and DiT (Peebles and Xie).

The right choice depends on the structure of c : low-dimensional scalar vectors favour concatenation or FiLM; sequence-valued or high-dimensional c favours cross-attention.

5.3 The score-level Bayes identity

Recall from the mathematical background that the score of a conditional distribution decomposes via Bayes’ rule:

$$\nabla_x \log p(x | y) = \nabla_x \log p(x) + \nabla_x \log p(y | x),$$

since the intractable marginal $p(y)$ drops out. Specialized to diffusion latents at noise level t :

$$\nabla_{z_t} \log p_t(z_t | y) = \nabla_{z_t} \log p_t(z_t) + \nabla_{z_t} \log p_t(y | z_t).$$

The right-hand side has two terms:

- $\nabla_{z_t} \log p_t(z_t)$, the prior score, available from a diffusion model.
- $\nabla_{z_t} \log p_t(y | z_t)$, the gradient of a noise-conditional *classifier* that we can train separately.

An unconditional diffusion model plus a classifier is enough to sample conditionally, with no conditional generative training. This is the idea of classifier guidance, developed next. In the language of the previous section, the classifier gradient plays the role of $\nabla \log u_t$ when the reward corresponds to a log-likelihood.

5.4 Classifier guidance and the guidance scale

The vanilla Bayes identity samples from $p_t(z_t | y)$ exactly. In practice, Dhariwal and Nichol (2021) found that *amplifying* the classifier term substantially improves sample quality. They introduce a scalar guidance scale $\gamma \geq 0$:

$$\nabla_{z_t} \log p_t^\gamma(z_t | y) := \nabla_{z_t} \log p_t(z_t) + \gamma \nabla_{z_t} \log p_t(y | z_t).$$

Inverting log and gradient, this corresponds to sampling from the tilted distribution

$$p_t^\gamma(z_t | y) \propto p_t(z_t) \cdot p_t(y | z_t)^\gamma.$$

Three regimes:

- $\gamma = 0$: unconditional sampling. The classifier is ignored.
- $\gamma = 1$: the Bayesian posterior $p_t(z_t | y)$.
- $\gamma > 1$: the conditional likelihood is raised to a power greater than one, sharpening the distribution toward modes where the classifier is most confident.

Empirically γ in the range one to ten is typical. Higher γ improves alignment with y but reduces diversity, and at extreme values produces visible artifacts (oversaturation in images, biologically implausible structure in proteins). Setting $\gamma > 1$ is heuristic: the resulting distribution is sharpened beyond the exact posterior, not the posterior itself. It is widely used because it improves empirical sample quality.

5.5 Training the classifier

The classifier required for guidance must operate on noisy inputs z_t at every noise level encountered during sampling. We sometimes call this a *noise-conditional classifier* to distinguish it from a standard classifier trained on clean data.

Training recipe. Given a labelled dataset $\{(x_i, y_i)\}$:

1. Sample $t \sim \mathcal{U}\{1, \dots, T\}$ and $\epsilon \sim \mathcal{N}(0, I)$.
2. Form $z_t = \sqrt{\bar{\alpha}_t} x + \sqrt{1 - \bar{\alpha}_t} \epsilon$.
3. Train $p_\phi(y \mid z_t, t)$ by minimizing the cross-entropy on (z_t, t, y) , i.e. the negative log-probability $-\log p_\phi(y \mid z_t, t)$ that the classifier assigns to the true label y , averaged over the dataset and over the sampled t and ϵ .

Why an off-the-shelf classifier fails. A standard classifier trained on clean data has never seen high-noise inputs. When fed a heavily noised z_t , it produces arbitrary outputs, and its gradient with respect to z_t is essentially adversarial noise. To produce useful gradients at every noise level, the classifier must be trained on the noised distribution itself.

Continuous properties. For a real-valued target, replace the categorical likelihood with a Gaussian:

$$\log p_\phi(y \mid z_t, t) = -\frac{1}{2\sigma^2} (y - r_\phi(z_t, t))^2 + \text{const},$$

where r_ϕ is a noise-conditional regressor. Same machinery, different loss.

5.6 Sampling algorithm with classifier guidance

The guidance modifies the score at each reverse step:

$$\nabla_{z_t} \log p_t^\gamma(z_t \mid y) = \nabla_{z_t} \log p_t(z_t) + \gamma \nabla_{z_t} \log p_\phi(y \mid z_t, t).$$

In the clean-data parameterization, the shifted predictor is

$$\tilde{x}_\theta(z_t, t, y) = \hat{x}_\theta(z_t, t) + \frac{1 - \bar{\alpha}_t}{\sqrt{\bar{\alpha}_t}} \gamma \nabla_{z_t} \log p_\phi(y \mid z_t, t).$$

The factor $(1 - \bar{\alpha}_t)/\sqrt{\bar{\alpha}_t}$ converts the score (gradient of log-density) into the clean-data parameterization, by combining the standard parameterization relations (see Appendix F for the derivation). The DDPM ancestral sampling algorithm is otherwise unchanged.

Algorithm.

1. Sample $z_T \sim \mathcal{N}(0, I)$.
2. For $t = T, T - 1, \dots, 1$:
 - (a) Compute $\hat{x}_\theta(z_t, t)$ (one forward pass through the diffusion model).
 - (b) Compute $\nabla_{z_t} \log p_\phi(y \mid z_t, t)$ (one forward and one backward pass through the classifier).
 - (c) Compute the shifted predictor

$$\tilde{x}_\theta = \hat{x}_\theta(z_t, t) + \frac{1 - \bar{\alpha}_t}{\sqrt{\bar{\alpha}_t}} \gamma \nabla_{z_t} \log p_\phi(y \mid z_t, t).$$

- (d) Take a standard reverse step using $\tilde{x}_\theta$ in place of $\hat{x}_\theta$: compute $\mu_\theta = \tilde{\mu}_t(z_t, \tilde{x}_\theta)$ and sample $z_{t-1} \sim \mathcal{N}(\mu_\theta, \tilde{\beta}_t I)$.

3. Return z_0 .

Cost. Per sampling step: one diffusion forward pass plus one classifier forward and backward. Roughly two to three times slower than unconditional sampling.

5.7 Reconstruction guidance for clean-data classifiers

Classifier guidance as described requires a noise-conditional classifier $p_\phi(y | z_t, t)$. In practice, this is often inconvenient:

- Property predictors in scientific applications are typically trained on clean data. A binding affinity predictor takes a real protein structure as input, not a noisy one.
- Retraining a noise-conditional version of every classifier is expensive and engineering-heavy.

5.8 The Diffusion Posterior Sampling (DPS) trick (Chung et al., 2023).

[Chu+23] If we have a clean-data classifier $p_\phi(y | x)$, we approximate the noise-conditional classifier gradient by evaluating p_ϕ on the diffusion model’s current denoised estimate $\hat{x}_\theta(z_t, t)$ (the Tweedie estimate of x given z_t), then backpropagating through the diffusion network. The replacement guidance gradient is

$$\nabla_{z_t} \log p_\phi(y | \hat{x}_\theta(z_t, t)).$$

The sampling update is otherwise identical to classifier guidance, with this replacement gradient substituted in.

Cost. Backpropagation through both the classifier and the diffusion network, since $\hat{x}_\theta$ is a function of the diffusion network’s output. This is more expensive than vanilla classifier guidance.

Approximation error. The replacement gradient is not the true score $\nabla_{z_t} \log p_t(y | z_t)$; it is a one-step denoising approximation. At high noise levels, $\hat{x}_\theta$ is a poor estimate of the clean data and the guidance signal is biased. This bias is unavoidable but usually mild in practice.

Why it matters. Reconstruction guidance is the practical form of classifier guidance in the applied literature. Used in inverse problems in imaging (deblurring, inpainting, super-resolution). Enables off-the-shelf property predictors for molecular and protein design (binding affinity, solubility, drug-likeness) to get coupled to pretrained diffusion models.

5.9 The reverse Bayes trick

We apply Bayes’ rule in the *other direction*. Starting from $p(y | x) = p(x | y)p(y)/p(x)$, taking logs and gradients with respect to x :

$$\nabla_x \log p(y | x) = \nabla_x \log p(x | y) - \nabla_x \log p(x).$$

Specialized to diffusion latents:

$$\nabla_{z_t} \log p_t(y | z_t) = \nabla_{z_t} \log p_t(z_t | y) - \nabla_{z_t} \log p_t(z_t).$$

The classifier gradient on the left, which previously needed an external classifier, is now the *difference of two scores*: the conditional score $\nabla_{z_t} \log p_t(z_t | y)$ and the unconditional score $\nabla_{z_t} \log p_t(z_t)$. If a single diffusion network can act in both conditional and unconditional modes, the classifier gradient comes for free. This is the idea of classifier-free guidance (Ho and Salimans, 2022).

5.10 Classifier-free guidance: the training recipe

Train a single diffusion model to act in both conditional and unconditional modes via *conditioning dropout*: during training, with probability p_{drop} (typically 0.1 to 0.2), replace the conditioning vector c with a special null token $\emptyset$ before passing it to the network. Otherwise pass the real c . The training loss is unchanged from plain conditioning:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,c), t, \epsilon, b} [\|\hat{x}_\theta(z_t, t, b \cdot c + (1-b) \cdot \emptyset) - x\|^2],$$

where $b \sim \text{Bernoulli}(1 - p_{\text{drop}})$.

After training, the same network functions in two modes:

- $\hat{x}_\theta(z_t, t, c)$: conditional model, approximating $\nabla_{z_t} \log p_t(z_t | c)$ (up to the standard score conversion).
- $\hat{x}_\theta(z_t, t, \emptyset)$: unconditional model, approximating $\nabla_{z_t} \log p_t(z_t)$.

Training cost is identical to plain conditional training: one extra line of code, no extra forward passes, no separate network.

5.11 Classifier-free guidance: the sampling formula

Start from the classifier-guidance score (with $\gamma = 1 + w$) and substitute the reverse-Bayes identity $\nabla_{z_t} \log p_t(c | z_t) = \nabla_{z_t} \log p_t(z_t | c) - \nabla_{z_t} \log p_t(z_t)$ for the classifier gradient. Collecting the conditional and unconditional scores:

$$\nabla_{z_t} \log p_t^\gamma = (1 + w) \nabla_{z_t} \log p_t(z_t | c) - w \nabla_{z_t} \log p_t(z_t).$$

The score-to-clean-data map (Appendix F) is affine and the coefficients $(1 + w)$ and $-w$ sum to one, so the same combination carries over to the clean-data predictions:

$$\tilde{x}_\theta(z_t, t, c) = (1 + w) \hat{x}_\theta(z_t, t, c) - w \hat{x}_\theta(z_t, t, \emptyset).$$

The scalar $w \geq 0$ is the *guidance strength*. Some papers use $\gamma = 1 + w$; the two conventions are equivalent. Three regimes:

- $w = 0$: standard conditional sampling, $\tilde{x}_\theta = \hat{x}_\theta(z_t, t, c)$.
- $w > 0$: extrapolate past the conditional prediction, in the direction away from the unconditional one.
- Typical values: $w \in [1, 7]$ for text-to-image, varying by application and p_{drop} .

The DDPM ancestral sampling algorithm is otherwise unchanged: at each step, compute the two network evaluations, combine via the formula, then take a standard reverse step using $\tilde{x}_\theta$.

5.12 Classifier-free guidance: geometric picture and cost

Geometric picture. The CFG formula places $\tilde{x}_\theta$ on the line through the conditional and unconditional predictions, on the conditional side, at distance proportional to w past the conditional prediction. For $w > 0$, the guided prediction moves further in the direction conditional-minus-unconditional, amplifying whatever distinguishes “what the model thinks given c ” from “what the model thinks without c .”

CFG is therefore an *extrapolation* in score-vector space, not an interpolation. Linear extrapolation past trained outputs is generally unstable, but in score space it works because the diffusion model is iterative and self-correcting across many sampling steps.

Cost. Two forward passes through the diffusion model per sampling step (one conditional, one unconditional), no backward passes. Roughly twice the cost of plain conditional sampling.

5.13 Sequential Monte Carlo steering: the idea

Classifier, reconstruction, and classifier-free guidance all modify the score within a single trajectory, using an estimate of $\nabla_{z_t} \log u_t$. SMC steering takes a different route. It runs many trajectories in parallel and reweights them by the reward: trajectories heading toward high-reward outputs gain weight, the rest lose it and are discarded. The weighted ensemble approximates the tilted target q_0^* .

Setup. A pretrained diffusion model defines the prior p_t , and a reward $r(x)$ scores how desirable a clean output is. The target is the Boltzmann tilt from the idealism-realism section,

$$q_0^*(x) \propto p_0(x) \exp(r(x)/\beta),$$

with noised intermediates $q_t^*(z_t) \propto p_t(z_t) u_t(z_t)$. Here u_t is the value function, the expected exponentiated reward over completions of the prior from z_t , with boundary $u_0 = \exp(r/\beta)$.

Feynman-Kac framing. View the reverse process as a Markov chain $Z_T \rightarrow \dots \rightarrow Z_0$ and run N copies in parallel. Each intermediate target q_t^* is the prior reweighted by a *potential function* g_t . The optimal potential is the value function, $g_t^* = u_t$. Any valid choice need only satisfy the boundary condition $g_0 \propto u_0$, since the intermediate g_t for $t > 0$ cancel across the trajectory (shown below) and so do not change the target. The reward enters only through g_t , never as a gradient, so SMC handles non-differentiable rewards: simulators, external scorers, or energy functions that classifier and reconstruction guidance cannot use.

5.14 SMC steering: the algorithm

The proposal is the model’s reverse step, the tractable Gaussian from the ancestral sampler: substitute the prediction $\hat{x}_\theta(z_t, t)$ for the unknown clean data in the closed-form conditioned posterior $q(z_{t-1} \mid z_t, x)$, giving $p_\theta(z_{t-1} \mid z_t) = \mathcal{N}(z_{t-1}; \tilde{\mu}_t(z_t, \hat{x}_\theta), \tilde{\beta}_t I)$. The simplest scheme leaves this proposal unmodified and steers only the weights. The Twisted Diffusion Sampler (Wu et al. 2023) [Wu+23] additionally steers the proposal, noted below.

SMC-steered diffusion sampling (unmodified proposal).

1. Initialize N particles: $z_T^{(i)} \sim \mathcal{N}(0, I)$, weights $w^{(i)} = 1/N$.
2. For $t = T, T-1, \dots, 1$:
 - (a) *Propagate.* Sample $z_{t-1}^{(i)} \sim p_\theta(z_{t-1} \mid z_t^{(i)})$.
 - (b) *Reweight.* Multiply each weight by $g_{t-1}(z_{t-1}^{(i)})/g_t(z_t^{(i)})$, updating the ensemble from an approximation of q_t^* to one of q_{t-1}^* . (The weight is a pure potential ratio because the unmodified proposal cancels the prior transition in the general SMC weight.)
 - (c) *Resample.* If $\text{ESS} < N/2$ (and $t > 1$), draw N particles with replacement with probability $\propto w^{(i)}$, and reset weights to $1/N$.
3. Return $\{(z_0^{(i)}, w^{(i)})\}_{i=1}^N$, the weighted ensemble approximating q_0^* .

Obtaining a sample. Draw one particle with probability $\propto w^{(i)}$. Conditioning, CFG, and classifier guidance steer a single trajectory, so one run is already a sample. SMC steers a population, and the sample is read off the final ensemble.

Cost. Per step: N diffusion forward passes (parallelizable) plus N potential evaluations, linear in N . The final step is not resampled.

5.15 SMC steering: properties

Where the reward enters. Without resampling, the per-step ratios multiply into a product whose intermediate factors cancel, leaving the boundary potential $g_0 \propto \exp(r/\beta)$. This is what makes the ensemble target q_0^* , for any choice of intermediate g_t . Resampling resets weights to $1/N$ and breaks the product but keeps the target: the accumulated weight is converted into particle multiplicity (high-weight particles duplicated, low-weight ones culled) rather than discarded.

Why resampling matters. After several reweighting steps a few particles capture almost all the weight and the rest contribute nothing. The effective sample size measures how many contribute,

$$\text{ESS} = \frac{(\sum_i w^{(i)})^2}{\sum_i (w^{(i)})^2},$$

equal to N for uniform weights and 1 when one particle dominates. Resampling duplicates high-weight particles. The duplicates are identical immediately afterwards and re-separate at the next propagation step, since the stochastic reverse kernel maps identical inputs to different outputs. (At small N or strong guidance this re-separation can fail and the ensemble degenerates.)

Bias and variance. The sense in which the ensemble approximates q_0^* is made precise through the estimator $\hat{\mu}_N = \sum_i w^{(i)} f(z_0^{(i)})$ of $\mathbb{E}_{q_0^*}[f]$, for a test function f . As $N \rightarrow \infty$, $\hat{\mu}_N \rightarrow \mathbb{E}_{q_0^*}[f]$ for any valid g_t : the intermediate g_t does not change the limit. At finite N the estimator carries an $O(1/N)$ bias from self-normalization and resampling, present for every g_t , and the choice of g_t controls both this bias and the variance $\text{Var}[\hat{\mu}_N]$ (over independent runs). Since N is small in practice (16 to 256), the choice determines the quality of the approximation, and hence of the samples drawn from it.

Choosing g_t . The intermediate potentials lie on a bias-variance spectrum:

- Terminal-only, $g_t \equiv 1$ for $t > 0$: reward applied only at $t = 0$, so no bias enters through $\hat{x}_\theta$ at noisy states. Highest $\text{Var}[\hat{\mu}_N]$, needing many particles.
- Rolled-out, $g_t(z_t) = \exp(r(\hat{x}_\theta(z_t, t))/\beta)$: a one-step approximation to u_t . Lower variance, but $\hat{x}_\theta$ is inaccurate at high t and adds bias.
- Learned twist (Twisted Diffusion Sampler), $g_t \approx u_t$: lowest variance, at the cost of training a value-function approximator. TDS also uses g_t to steer the proposal, not only the weights.

Separately, memory is N times single-particle sampling, often the binding constraint for large models.

5.16 Comparison of the four methods

We have developed four methods for steering a diffusion model toward high-reward outputs: plain conditioning, classifier guidance (with the reconstruction-guidance variant DPS), classifier-free guidance, and sequential Monte Carlo. We close by organizing them along the axes that matter in practice.

	Conditioning	Classifier guidance	CFG	SMC
When applied	Training	Inference	Training + inference	Inference
Mechanism	Conditional training	Score modification	Score modification	Particle reweighting
Relation to u_t	Implicit in conditional model	Explicit approximation	Implicit approximation	Not approximated; exploits asymp. correctness
Training data	(x, c) pairs	unlabelled $x + (x, y)$ for classifier	(x, c) pairs with dropout	unlabelled $x +$ property scorer
Separate model?	No	Noise-conditional classifier	No	Property scorer (clean data ok)
Gradients at inference?	No	Yes	No	No
Reward differentiable?	N/A	Required	N/A	Not required
Cost per step	$1\times$	1 fwd + 1 fwd/bwd	$2\times$	$N\times$
Inference-time knob	None	γ	w	N , potential g_t
Asymptotically targets q^* ?	For $p(\cdot c)$	At $\gamma = 1$	At $w = 0$	As $N \rightarrow \infty$
Notable failure mode	Weak control, no inference knob	Adversarial gradient at high γ	Sharpened, not posterior, at $w > 0$	Mode collapse via aggressive resampling

Choosing a method in practice.

- *Plain conditioning*: cheapest at inference, simplest to train. Use when paired data is available and the conditioning signal is strong. Helps with generalization.
- *CFG*: paired data needed, but no separate classifier. Use when plain conditioning under-respects the condition. One extra forward pass per step.
- *Classifier guidance*: requires a differentiable reward and a noise-conditional or clean-data (DPS) classifier. Provides the amplification knob γ at the cost of adversarial-gradient risk at high γ . At $\gamma = 1$ it is exact posterior sampling.
- *SMC*: the only method handling non-differentiable rewards natively, and the only one asymptotically exact for q^* . Cost scales linearly with the number of particles.

No single method dominates. The choice depends on what is available (gradients, paired data, classifiers), what is expensive (per-sample compute, training compute, property evaluation), and what is required (asymptotic correctness, inference speed, sample diversity). Real systems often combine methods: a CFG-trained diffusion model augmented at inference time with SMC reweighting on a non-differentiable fitness signal is a coherent design.

We now move to discrete diffusion models and develops the same guidance methods in that setting.

6 Discrete diffusion

6.1 Why discrete data is different

We previously developed continuous diffusion and four guidance methods built on a foundation that assumes the data lives in $\mathbb{R}^d$, as well as Gaussian noise, the closed-form marginal $z_t = \sqrt{\bar{\alpha}_t} x + \sqrt{1 - \bar{\alpha}_t} \epsilon$, the score $\nabla_x \log p_t(x)$, and gradient-based corrections at every reverse step.

For biological sequences this foundation does not apply. A peptide is a string $x = (x_1, \dots, x_L)$ where each x_i is one of 20 amino acids. Proteins, DNA, and RNA sequences all take values in finite alphabets. Differentiating with respect to a categorical variable is not defined, so there is no meaningful notion of $\nabla_x \log p(x)$.

The conceptual story of diffusion carries over completely: forward corruption, learned reverse denoising, sampling by transporting backward through intermediate distributions. What changes is the concrete mechanics:

- *Forward process.* The Gaussian forward process adds noise in $\mathbb{R}^d$, which makes no sense for categorical variables. We need a discrete corruption mechanism.
- *Training objective.* Denoising score matching requires $\nabla_x \log p_t(x)$, which is undefined on a discrete state space. We need a different objective that does not require gradients in x .
- *Guidance.* Classifier guidance and reconstruction guidance both rely on $\nabla_x \log p(y | x)$, which is undefined. Some guidance methods carry over (conditioning, CFG, SMC), others do not.

6.2 Setting up notation

Let V denote a finite alphabet. For protein sequences, V consists of the 20 standard amino acids. To support diffusion, we extend the alphabet with a special *mask token* $\mathbf{m}$, giving $|V| = 21$.

A sequence of length L is $x = (x_1, \dots, x_L) \in V^L$. Throughout these notes, $i \in \{1, \dots, L\}$ indexes positions in the sequence and $t \in \{0, 1, \dots, T\}$ indexes diffusion timesteps, with $t = 0$ corresponding to clean data and $t = T$ to the fully-noised state. We write $x_t^{(i)}$ for the value at position i at time t , and reserve x_0 for the clean (data) sequence.

The notation is heavier than in continuous diffusion because we have to track both time and position. In the continuous case the latent z_t was a single vector in $\mathbb{R}^d$; here the latent x_t is a sequence of L categorical variables, each one of $|V|$ possibilities. We will sometimes write the categorical distribution over V at position i as a probability vector in $\mathbb{R}^{|V|}$, and the network’s prediction at position i as a distribution over V .

6.3 Two main families of discrete diffusion

Discrete diffusion was introduced by Austin et al. (2021) [Aus+21] in the paper *Structured Denoising Diffusion Models in Discrete State-Spaces* (D3PM). The forward process is a Markov chain over V , defined by transition matrices $Q_t \in \mathbb{R}^{|V| \times |V|}$:

$$q(x_t^{(i)} = j \mid x_{t-1}^{(i)} = k) = [Q_t]_{kj}.$$

The matrix Q_t must satisfy $[Q_t]_{kj} \geq 0$ and $\sum_j [Q_t]_{kj} = 1$ for each row k (it is a valid transition matrix for a Markov chain). Each position evolves independently under the same transition law. Different choices of Q_t yield different discrete diffusion variants.

Two choices dominate the literature:

- *Uniform corruption.* Q_t pushes each token toward the uniform distribution over V . With probability β_t each token is replaced by a uniformly random token from V , and with probability $1 - \beta_t$ it stays unchanged. After enough steps, the sequence is indistinguishable from uniform random tokens.
- *Absorbing (masked) corruption.* Q_t leaves each token unchanged with some probability, or replaces it with the mask token $\mathbf{m}$. Once a position is masked, it stays masked. After enough steps, all positions are masked.

Other variants exist (BLOSUM-based transition matrices for proteins, informative corruption matrices based on amino acid similarity), but uniform and absorbing are the two main families.

6.4 Uniform versus absorbing: which to use

The two choices differ in the structure of the denoising task and have distinct trade-offs.

Uniform corruption.

- *Denoising task.* At each reverse step, the network must decide for every position whether the current token is a corruption, and if so, what the original was. It receives no explicit signal about which positions are corrupted: detection and prediction happen simultaneously.
- *Information preserved.* Even after many corruption steps, most positions retain their original token with non-trivial probability, so the chain carries information about the clean sequence throughout the trajectory.
- *Reverse posterior.* The closed-form reverse posterior $q(x_{t-1} \mid x_t, x_0)$ is more complex than in the absorbing case, involving the full transition matrix structure.

Absorbing (masked) corruption.

- *Denoising task.* Corrupted positions are immediately identifiable by the mask token $\mathbf{m}$. The network only has to fill in masks; unmasked positions are guaranteed correct under the forward process.
- *Information preserved.* Each position is either fully known or fully unknown. There is no partial information: a masked position carries zero information about its original value.
- *Reverse posterior.* The closed-form reverse posterior is simpler: at each masked position, the only options are “stay masked” or “unmask to x_0 .”

Empirical performance. For biological sequences and language modelling, absorbing-style models currently outperform uniform-style ones. The EvoDiff paper (Alamdari et al. 2023) [Ala+23] reports that order-agnostic autoregressive diffusion (a special case of absorbing diffusion) outperforms D3PM with uniform or BLOSUM transition matrices on UniRef50, and is the only variant whose quality improves with model size. For language modelling, the current state-of-the-art models (MDLM, SEDD masked variant) are also absorbing-style.

For the following we focus on absorbing diffusion. The machinery developed below transfers to uniform corruption with the appropriate substitutions in the reverse posterior and training loss.

6.5 The absorbing forward process

Let $\beta_t \in [0, 1]$ be a schedule analogous to the variance schedule in continuous diffusion. The forward transition is defined position-wise:

$$q(x_t^{(i)} \mid x_{t-1}^{(i)}) = \begin{cases} (1 - \beta_t) \delta_{x_t^{(i)}=x_{t-1}^{(i)}} + \beta_t \delta_{x_t^{(i)}=\mathbf{m}} & \text{if } x_{t-1}^{(i)} \neq \mathbf{m}, \\ \delta_{x_t^{(i)}=\mathbf{m}} & \text{if } x_{t-1}^{(i)} = \mathbf{m}. \end{cases}$$

In words: at each step, each unmasked position is absorbed into the mask state with probability β_t , otherwise stays the same. Masked positions remain masked forever. Different positions are corrupted independently.

With a properly designed schedule, after T steps every position is masked with probability arbitrarily close to one, regardless of the initial sequence. This is the discrete analogue of the continuous case’s “forward chain reaches pure Gaussian noise at $t = T$.”

The schedule β_t controls how much corruption is applied per step. A common choice is to set β_t so that the cumulative mask probability $\bar{\beta}_t$ increases linearly from 0 at $t = 0$ to 1 at $t = T$. The schedule is a hyperparameter that affects sample quality but not the underlying theory.

6.6 Closed-form marginal

The discrete diffusion has the same compositional property as the continuous case: many steps of the forward chain can be collapsed to a single step. Define the *cumulative mask probability*

$$\bar{\beta}_t := 1 - \prod_{s=1}^t (1 - \beta_s).$$

The marginal of $x_t^{(i)}$ given $x_0^{(i)}$ has the closed form

$$q(x_t^{(i)} \mid x_0^{(i)}) = (1 - \bar{\beta}_t) \delta_{x_t^{(i)}=x_0^{(i)}} + \bar{\beta}_t \delta_{x_t^{(i)}=\mathbf{m}}.$$

Each position is independently either kept (probability $1 - \bar{\beta}_t$) or masked (probability $\bar{\beta}_t$). We can jump to any noise level in one step: sample a Bernoulli mask for each position, replace selected positions with $\mathbf{m}$.

The structure parallels the continuous case. There we had

$$q(z_t | x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I),$$

a closed-form Gaussian in terms of cumulative signal-retention $\bar{\alpha}_t$. Here we have a closed-form Bernoulli mixture in terms of cumulative mask probability $\bar{\beta}_t$.

6.7 The reverse process: unmasking

Sampling reverses the forward chain. Start from a fully masked sequence $x_T = (\mathbf{m}, \mathbf{m}, \dots, \mathbf{m})$ and iteratively unmask. The key conditional is $q(x_{t-1} | x_t)$.

The absorbing structure imposes strong constraints on $q(x_{t-1} | x_t)$. At time t , a position is either masked or unmasked. The reverse step preserves this structure:

- If position i is unmasked at time t (so $x_t^{(i)} \neq \mathbf{m}$), then position i at time $t - 1$ must take the same value. The forward chain has no transitions from non-mask tokens to other non-mask tokens, so the reverse chain has none either.
- If position i is masked at time t (so $x_t^{(i)} = \mathbf{m}$), then at $t - 1$ it either remains masked or is unmasked to some token $v \in V \setminus \{\mathbf{m}\}$.

The non-trivial part of the reverse step is therefore: for each currently masked position, decide whether to unmask it now, and if so, to which token.

6.8 The conditional reverse posterior

As in the continuous case, the unconditional reverse $q(x_{t-1} | x_t)$ is intractable, but the reverse conditioned on the clean sequence x_0 is tractable. For a position i that is masked at time t , we apply Bayes' rule to the forward transition:

$$q(x_{t-1}^{(i)} | x_t^{(i)} = \mathbf{m}, x_0^{(i)}) = \frac{q(x_t^{(i)} = \mathbf{m} | x_{t-1}^{(i)}) q(x_{t-1}^{(i)} | x_0^{(i)})}{q(x_t^{(i)} = \mathbf{m} | x_0^{(i)})}.$$

The denominator is the closed-form marginal from above: $q(x_t^{(i)} = \mathbf{m} | x_0^{(i)}) = \bar{\beta}_t$.

The variable $x_{t-1}^{(i)}$ in the numerator can only take two values: either $\mathbf{m}$ (still masked at $t - 1$) or $x_0^{(i)}$ (its original value at $t - 1$). Anything else has zero probability under the forward chain, since non-mask tokens never transition to other non-mask tokens.

Case 1: $x_{t-1}^{(i)} = \mathbf{m}$ (still masked). The forward kernel gives $q(x_t^{(i)} = \mathbf{m} | x_{t-1}^{(i)} = \mathbf{m}) = 1$ (masked stays masked). The marginal gives $q(x_{t-1}^{(i)} = \mathbf{m} | x_0^{(i)}) = \bar{\beta}_{t-1}$. So the numerator is $\bar{\beta}_{t-1}$ and the ratio is $\bar{\beta}_{t-1}/\bar{\beta}_t$.

Case 2: $x_{t-1}^{(i)} = x_0^{(i)}$ (not yet masked). The forward kernel gives $q(x_t^{(i)} = \mathbf{m} | x_{t-1}^{(i)} = x_0^{(i)}) = \beta_t$ (unmasked gets masked with probability β_t). The marginal gives $q(x_{t-1}^{(i)} = x_0^{(i)} | x_0^{(i)}) = 1 - \bar{\beta}_{t-1}$. So the numerator is $\beta_t(1 - \bar{\beta}_{t-1})$ and the ratio is $\beta_t(1 - \bar{\beta}_{t-1})/\bar{\beta}_t$.

Using $\bar{\beta}_t = \bar{\beta}_{t-1} + \beta_t(1 - \bar{\beta}_{t-1})$ (the cumulative mask probability decomposes into ‘‘already masked’’ plus ‘‘just got masked’’), the two cases sum to one. Putting them together:

$$q(x_{t-1}^{(i)} | x_t^{(i)} = \mathbf{m}, x_0^{(i)}) = \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t} \delta_{x_{t-1}^{(i)} = \mathbf{m}} + \left(1 - \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t}\right) \delta_{x_{t-1}^{(i)} = x_0^{(i)}}.$$

At time $t - 1$, the masked position either stays masked with probability $\bar{\beta}_{t-1}/\bar{\beta}_t$, or is unmasked to its original value with the complementary probability.

Interpretation. The ratio $\bar{\beta}_{t-1}/\bar{\beta}_t$ has a clean meaning: given the position was eventually masked by time t , this is the conditional probability that it was already masked by time $t - 1$. At early times (t small, $\bar{\beta}_t$ small), this ratio is close to one and the chain tends to unmask few positions per step. At late times (t near T , $\bar{\beta}_t$ near one), the ratio is close to $\bar{\beta}_{t-1}$ and many positions get unmasked.

6.9 The training trick carries over

The strategy from the continuous case applies here too: train a network $\hat{x}_\theta(x_t, t)$ whose output at each position is a distribution over V , and substitute its prediction at masked positions into the closed-form reverse posterior. The reverse step at a masked position i becomes

$$p_\theta(x_{t-1}^{(i)} | x_t^{(i)} = \mathbf{m}) = \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t} \delta_{x_{t-1}^{(i)} = \mathbf{m}} + \left(1 - \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t}\right) \hat{x}_\theta(\cdot | x_t, t)_i.$$

At each step, with probability $\bar{\beta}_{t-1}/\bar{\beta}_t$ leave the position masked; otherwise sample a token from the network’s predictive distribution at that position. Unmasked positions are copied through unchanged.

The structure parallels the continuous case: predict the clean data, plug into a closed-form posterior, sample. Only the parametric family changes (categorical instead of Gaussian), with a corresponding change in the training loss.

6.10 The training loss

The variational lower bound on $\log p_\theta(x_0)$ is derived as in continuous diffusion. The simplification is more dramatic in the discrete case. Following Sahoo et al. (2024) [Sah+24] and Shi et al. (2024) [Shi+24], the ELBO collapses to a time-weighted cross-entropy on masked positions:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0, t, x_t} \left[-\frac{\beta_t}{\bar{\beta}_t} \sum_{i: x_t^{(i)} = \mathbf{m}} \log \hat{x}_\theta(x_0^{(i)} | x_t, t)_i \right],$$

where $x_0 \sim p_{\text{data}}$, $t \sim \mathcal{U}\{1, \dots, T\}$, and $x_t \sim q(\cdot | x_0)$.

Recall that the cross-entropy between a true label y and a predicted distribution $\hat{p}$ over a finite set is $-\log \hat{p}(y)$: the negative log-probability the model assigns to the true outcome. Here, $\hat{p}$ is $\hat{x}_\theta(\cdot | x_t, t)_i$ (the network’s predicted distribution over V for position i), and the “true label” is $x_0^{(i)}$ (the clean token at that position). The loss sums this contribution over the masked positions and weights by the time-dependent factor $\beta_t/\bar{\beta}_t$, which comes from the ELBO and balances early and late timesteps. The derivation is in the Sahoo et al. [Sah+24] Appendix.

In words: sample a clean sequence, sample a timestep, mask each position independently with probability $\bar{\beta}_t$, ask the network to predict the original tokens at masked positions, accumulate weighted cross-entropy. The structure parallels the continuous case: one forward pass per training example, expectation approximated by Monte Carlo, training reduces to supervised learning.

6.11 Connection to BERT-style masked language modelling

The training loss has the same structure as BERT pretraining: mask some fraction of tokens, predict them, accumulate cross-entropy. The differences:

- *Mask ratio.* BERT uses a fixed mask ratio (e.g. 15%). Masked discrete diffusion samples the mask ratio per example from the schedule, ranging from near 0% to near 100%.
- *Time weighting.* BERT has no time-dependent weight; masked discrete diffusion uses $\beta_t/\bar{\beta}_t$ from the ELBO.

Practical implications:

- Discrete diffusion models can be warm-started from pretrained BERT weights and fine-tuned with the diffusion loss.
- BERT is essentially a single-step masked diffusion model trained at one fixed noise level. MDLM (Sahoo et al. 2024) [Sah+24] shows that BERT-style architectures trained with the diffusion loss achieve competitive language modelling perplexity, with no architectural changes.

Thus, the discrete diffusion can be viewed as a principled generalization of BERT to a continuum of mask ratios.

6.12 Sampling algorithm

The sampling algorithm follows directly from the training trick: predict the clean data at each step, plug into the closed-form reverse posterior, sample.

Ancestral sampling for masked discrete diffusion.

1. Initialize $x_T = (\mathbf{m}, \mathbf{m}, \dots, \mathbf{m})$.
2. For $t = T, T-1, \dots, 1$:
 - (a) Compute the network’s predictions $\hat{x}_\theta(\cdot \mid x_t, t)_i$ at every position (one forward pass).
 - (b) Sample x_{t-1} from the reverse posterior, substituting $\hat{x}_\theta(\cdot \mid x_t, t)_i$ at each masked position and copying unmasked positions through unchanged.
3. Return x_0 as the generated sequence.

Each iteration is one forward pass of the network. The number of steps T is a design choice, ranging from large T (analogous to DDPM’s $T = 1000$) down to $T = L$ (one position revealed per step on average).

6.13 Sampling order: parallel versus one-position-per-step

The standard sampling algorithm above unmask all positions in parallel: at step t , every masked position is independently either kept masked or unmasked, according to the reverse posterior. A widely used variant unmask *one position per step*, with $T = L$ chosen so the total number of steps equals the sequence length.

The two regimes.

- *Parallel*. Many positions can be unmasked in one step. Total steps T can be small (DDPM-style $T = 1000$) or as small as the sequence length, with each step unmasking a Bernoulli-distributed number of positions.
- *One per step*. At each step exactly one masked position is chosen and unmasked. Total steps $T = L$. The order in which positions are unmasked is a free hyperparameter.

The trained network supports both regimes; the choice is made at sampling time.

6.14 Comparison: continuous diffusion versus discrete diffusion

We compare the two diffusion families structurally, capturing how the mathematical objects differ between the continuous and discrete cases.

The two diffusion families share a common philosophy (forward corruption, learned reverse denoising, sampling by transporting backward through intermediate distributions) but differ in nearly every concrete detail.

	Continuous diffusion	Discrete (masked) diffusion
Data type	Real-valued $x \in \mathbb{R}^d$	Tokens $x \in V^L$
Forward process	Gaussian noising	Probabilistic masking
Per-step kernel	$z_t = \sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \epsilon$	Each token masked w.p. β_t
Closed-form marginal	$z_t = \sqrt{\alpha_t} x + \sqrt{1 - \alpha_t} \epsilon$	Each position kept w.p. $1 - \beta_t$
Reverse step	Gaussian posterior in $\mathbb{R}^d$	Categorical posterior on V
Network output	Noise $\hat{\epsilon}_\theta$ or clean $\hat{x}_\theta$	Categorical $\hat{x}_\theta(\cdot \mid x_t, t)_i$
Training loss	MSE on noise (or clean data)	Weighted cross-entropy on masked positions
$\nabla_x \log p$	Defined; learnable as a score	Undefined (discrete state)
Examples	RFdiffusion, AlphaFold3, OmegAMP	EvoDiff, MDLM, SEDD

The most consequential row is the one for $\nabla_x \log p$. The Gaussian forward process gives continuous diffusion a well-defined score at every noise level, which is what classifier guidance and reconstruction guidance exploit. Masking destroys this structure: the score of a categorical distribution with respect to its categorical argument is not defined. All other rows describe different machinery for the same conceptual job; this row is a genuine asymmetry that shaped the structure of continuous guidance methods.

7 Guidance for discrete diffusion

We revisit the four guidance methods but in the discrete setting. Plain conditioning, CFG, and SMC carry over with minor mechanical changes (covered in the next subsections). Classifier guidance does not carry over directly because of the gradient issue.

7.1 Plain conditioning

The denoiser takes an extra argument:

$$\hat{x}_\theta(x_t, t) \longrightarrow \hat{x}_\theta(x_t, t, c).$$

The training loss is the masked diffusion loss with the conditioning vector threaded through:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x_0, c), t, x_t} \left[-\frac{\beta_t}{\bar{\beta}_t} \sum_{i: x_t^{(i)} = \mathbf{m}} \log \hat{x}_\theta(x_0^{(i)} | x_t, t, c)_i \right].$$

The training data are now (x_0, c) pairs. Sampling is identical to the unconditional algorithm with the desired c fed in at every step.

7.2 Classifier guidance for discrete diffusion

Classifier guidance relies on the score-level Bayes identity $\nabla_x \log p(x | y) = \nabla_x \log p(x) + \nabla_x \log p(y | x)$, which requires $\nabla_x \log p(y | x)$, the gradient of a classifier with respect to its input. On a discrete state space this gradient is undefined: the classifier can be trained on discrete inputs and evaluated on discrete inputs, but it is not differentiable with respect to a categorical x_t .

Two workarounds appear in the literature:

- *Diffusion Optimized Sampling (NOS; Gruver et al. 2023)*. Backpropagate through the network’s continuous hidden states rather than through the discrete inputs. The classifier sits on top of the hidden states and provides a gradient direction in the hidden space.
- *Nisonoff et al. (2024)*. Develop a proper discrete analogue of classifier guidance that modifies the categorical logits directly.

Neither is standard enough to be part of the basic toolkit; we mention them for completeness but develop neither further.

A third option, applicable when a clean-data classifier is available, is reconstruction guidance (DPS): evaluate the classifier on the network’s predicted categorical distribution $\hat{x}_\theta(x_t, t)$, then take gradients through the softmax. This is conceptually similar to continuous DPS but the denoised estimate is a categorical distribution rather than a point in $\mathbb{R}^d$.

7.3 Classifier-free guidance for discrete diffusion: the training recipe.

The conditioning dropout concept from continuous case transfers without changes. With probability p_{drop} (typically 0.1 to 0.2), we replace the conditioning vector c with a special null token $\emptyset$ before passing it to the network. The training loss is otherwise unchanged from plain conditioning. After training, the same network functions as both a conditional model (when fed c) and an unconditional one (when fed $\emptyset$).

7.4 CFG: the sampling formula.

The continuous case mixed score vectors. The discrete case mixes logit vectors. Let $\ell_\theta(x_t, t, c)_i \in \mathbb{R}^{|V|}$ denote the network’s pre-softmax logits over the alphabet for position i , given the conditioning vector c . The CFG-guided logits are

$$\tilde{\ell}(x_t, t, c)_i = (1 + w) \ell_\theta(x_t, t, c)_i - w \ell_\theta(x_t, t, \emptyset)_i,$$

and the network’s predicted categorical distribution at position i becomes

$$\tilde{p}(x_0^{(i)} | x_t, t, c) = \text{softmax}(\tilde{\ell}(x_t, t, c)_i).$$

The scalar $w \geq 0$ is the guidance strength, with the same role as in the continuous case. At $w = 0$ this is standard conditional sampling. For $w > 0$, the softmax is sharpened in the direction that distinguishes the conditional prediction from the unconditional one. Typical values are $w \in [1, 5]$.

Geometric picture. The construction is the discrete analogue of CFG in continuous diffusion. There, the guided clean-data prediction $\tilde{x}_\theta$ lay on the line through the conditional and unconditional predictions $\hat{x}_\theta(z_t, t, c)$ and $\hat{x}_\theta(z_t, t, \emptyset)$, extrapolating past the conditional side. Here, the guided logit vector $\tilde{\ell}_i$ lies on the line through the conditional and unconditional logits, extrapolating past the conditional. After softmax, this concentrates probability on tokens the conditional model favours more strongly than the unconditional model. The geometric story transfers directly, only the space changes: from clean-data predictions in $\mathbb{R}^d$ to logits in $\mathbb{R}^{|V|}$.

Cost. Per sampling step, two forward passes through the network (one with c , one with $\emptyset$), no backward passes.

7.5 Sequential Monte Carlo for discrete diffusion

SMC for discrete diffusion is structurally identical to SMC for continuous diffusion. The algorithm from the continuous-data case carries over to the discrete-data case, with the continuous reverse process replaced by the discrete unmasking process.

Setup. Pretrained discrete diffusion prior p_θ (for example EvoDiff trained on UniRef50). Reward $r(x)$ scoring clean sequences. Target distribution

$$q_0^*(x) \propto p_\theta(x) \exp(r(x)/\beta),$$

the discrete-data analogue of the Boltzmann tilt, with $p_\theta(x)$ the discrete-diffusion prior.

The algorithm. Run N particles through the reverse (unmasking) process in parallel, reweight by a potential g_t at each step, resample when the effective sample size drops too low.

SMC for masked discrete diffusion.

1. Initialize N particles: $x_T^{(n)} = (\mathbf{m}, \dots, \mathbf{m})$, weights $w^{(n)} = 1/N$.
2. For $t = T, T-1, \dots, 1$:
 - (a) *Propagate.* Sample $x_{t-1}^{(n)}$ from the discrete reverse (unmasking) step using $\hat{x}_\theta$.
 - (b) *Reweight.* Multiply each weight by the ratio $g_{t-1}(x_{t-1}^{(n)})/g_t(x_t^{(n)})$, where g_t approximates the value function u_t , in practice $g_t(x_t) = \exp(r(\hat{x}_\theta(x_t, t))/\beta)$ evaluated on the predicted clean tokens.
 - (c) *Resample.* If $\text{ESS} < N/2$ (and $t > 1$), draw N particles with replacement with probability $\propto w^{(n)}$, and reset weights to $1/N$.
3. Return $\{(x_0^{(n)}, w^{(n)})\}_{n=1}^N$, the weighted ensemble approximating q_0^* .

The framework is identical to continuous case: same Feynman-Kac framing, same boundary condition $g_0 \propto r$, same asymptotic correctness as $N \rightarrow \infty$, same trade-off in choosing intermediate potentials. The discrete reverse step replaces the continuous reverse SDE, and the denoised estimate is a categorical distribution rather than a vector in $\mathbb{R}^d$, but the rest is mechanically the same.

Choices for the intermediate potential. The three options from before all apply.

- *Terminal-only.* $g_t \equiv 1$ for $t > 0$, $g_0 = \exp(r/\beta)$. High variance, simple to implement.
- *Denoised-estimate rollout.* Evaluate the property predictor on the network’s clean prediction: $g_t(x_t) = \exp(r(\hat{x}_\theta(x_t, t))/\beta)$. The categorical prediction can be marginalized (expected reward under the predicted distribution) or sampled (commit to a completion and score it). Lower variance, more compute.
- *Learned twist.* A learned approximation to the optimal $g_t^* = u_t$. Most expensive but lowest variance.

8 Key papers

This section collects the works the notes build on, grouped by topic, with a one- or two-sentence note on why each matters. Citations are given in author–year form as they appear in the main text.

Foundations of continuous diffusion.

- **Ho et al. (2020)**, *Denoising Diffusion Probabilistic Models (DDPM)*. [HJA20] The reference point for the whole course: introduces the noising chain, the closed-form marginal, the ϵ -parameterization, and the variational-bound-to-MSE training objective.
- **Sohl-Dickstein et al. (2015)**, *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. [Soh+15] The original diffusion-model paper, which framed generation as reversing a gradual noising process; DDPM is the practical realization of this idea.
- **Nichol and Dhariwal (2021)**, *Improved DDPM* [ND21]. Introduces the cosine noise schedule that adds less noise early on, improving sample quality at fixed T over the original linear schedule.

The score-based and continuous-time view.

- **Song and Ermon (2019)**, *Generative Modeling by Estimating Gradients of the Data Distribution*. [SE19] Establishes the score-based perspective and denoising score matching, the language in which guidance is most naturally expressed.
- **Song et al. (2021b)**, *Score-Based Generative Modeling through Stochastic Differential Equations*. [Son+21] Unifies discrete diffusion and score matching under a single forward SDE, introduces the probability-flow ODE.
- **Anderson (1982)**, *Reverse-Time Diffusion Equation Models*. [And82] The classical result that the time-reversal of an SDE is again an SDE with a score-corrected drift, which is what makes generation-as-reverse-SDE possible.

Faster samplers.

- **Song et al. (2021a)**, *Denoising Diffusion Implicit Models (DDIM)*. [SME21] Recasts the reverse process as a deterministic ODE, letting ~ 50 steps match the quality of 1000-step ancestral sampling.
- **Lu et al. (2022)**, *DPM-Solver*. [Lu+22] A higher-order ODE solver for the probability-flow ODE that pushes the step count down to roughly 10–20 while reusing the same trained network.

Guidance methods.

- **Dhariwal and Nichol (2021)**, *Diffusion Models Beat GANs on Image Synthesis*. [DN21] Introduces classifier guidance and the guidance scale γ , and the ADM architecture.
- **Ho and Salimans (2022)**, *Classifier-Free Diffusion Guidance*. [HS22] Removes the separate classifier by training one model in conditional and unconditional modes via conditioning dropout; the workhorse of modern guidance.
- **Chung et al. (2023)**, *Diffusion Posterior Sampling (DPS)*. [Chu+23] The reconstruction-guidance trick: evaluate a clean-data classifier on the denoised estimate $\hat{x}_\theta$, enabling off-the-shelf property predictors.
- **Wu et al. (2023)**, *Practical and Asymptotically Exact Conditional Sampling in Diffusion Models (Twisted Diffusion Sampler)*. [Wu+23] The SMC-steering reference, giving a learned twist that approximates the optimal intermediate potential u_t at lowest variance.

Discrete diffusion.

- Austin et al. (2021), *Structured Denoising Diffusion Models in Discrete State-Spaces (D3PM)*. [Aus+21] Introduces discrete diffusion via transition matrices Q_t , including the uniform and absorbing (masked) corruption families.
- Sahoo et al. (2024), *Simple and Effective Masked Diffusion Language Models (MDLM)*. [Sah+24] Shows the masked-diffusion ELBO collapses to a weighted cross-entropy on masked positions and connects the model to BERT-style training; the basis of our loss.
- Shi et al. (2024), *Simplified and Generalized Masked Diffusion for Discrete Data*. [Shi+24] A parallel simplification of the masked-diffusion objective, derived independently of and consistently with MDLM.
- Hoogeboom et al. (2022), *Autoregressive Diffusion Models*. [Hoo+22] Shows that one-position-per-step masked diffusion in random order is order-agnostic autoregressive generation (OADM), the formulation EvoDiff uses.

Applications to biological sequences.

- Alamdari et al. (2023), *EvoDiff*. [Ala+23] A pretrained discrete (OADM) diffusion model for proteins trained on UniRef50;

References

- [And82] B. D. Anderson. “Reverse-time diffusion equation models”. In: *Stochastic Processes and their Applications* 12.3 (1982), pp. 313–326. ISSN: 0304-4149. DOI: [https://doi.org/10.1016/0304-4149\(82\)90051-5](https://doi.org/10.1016/0304-4149(82)90051-5). URL: <https://www.sciencedirect.com/science/article/pii/0304414982900515>.
- [Soh+15] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265. URL: <https://proceedings.mlr.press/v37/sohl-dickstein15.html>.
- [SE19] Y. Song and S. Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.
- [HJA20] J. Ho, A. Jain, and P. Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.
- [Aus+21] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. “Structured Denoising Diffusion Models in Discrete State-Spaces”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 17981–17993. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.
- [DN21] P. Dhariwal and A. Nichol. “Diffusion Models Beat GANs on Image Synthesis”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 8780–8794. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.
- [ND21] A. Q. Nichol and P. Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8162–8171. URL: <https://proceedings.mlr.press/v139/nichol21a.html>.
- [SME21] J. Song, C. Meng, and S. Ermon. “Denoising Diffusion Implicit Models”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021*. OpenReview.net, 2021. URL: <https://openreview.net/forum?id=St1giarCHLP>.

- [Son+21] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *International Conference on Learning Representations*. 2021.
- [HS22] J. Ho and T. Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG]. URL: <https://arxiv.org/abs/2207.12598>.
- [Hoo+22] E. Hoogeboom, A. A. Gritsenko, J. Bastings, B. Poole, R. van den Berg, and T. Salimans. “Autoregressive Diffusion Models”. In: *International Conference on Learning Representations*. 2022.
- [Lu+22] C. Lu, Y. Zhou, F. Bao, J. Chen, C. LI, and J. Zhu. “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 5775–5787. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/260a14acce2a89dad36adc8eefe7c59e-Paper-Conference.pdf.
- [Ala+23] S. Alamdari, N. Thakkar, R. van den Berg, A. X. Lu, N. Fusi, A. P. Amini, and K. K. Yang. “Protein generation with evolutionary diffusion: sequence is all you need”. In: *bioRxiv* (2023). DOI: 10.1101/2023.09.11.556673. eprint: <https://www.biorxiv.org/content/early/2023/09/12/2023.09.11.556673.full.pdf>. URL: <https://www.biorxiv.org/content/early/2023/09/12/2023.09.11.556673>.
- [Chu+23] H. Chung, J. Kim, M. T. McCann, M. L. Klasky, and J. C. Ye. “Diffusion Posterior Sampling for General Noisy Inverse Problems”. In: *International Conference on Learning Representations (ICLR)*. 2023.
- [Wu+23] L. Wu, B. L. Trippe, C. A. Naesseth, D. M. Blei, and F. Cunningham. “Practical and Asymptotically Exact Conditional Sampling in Diffusion Models”. In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 56877–56905.
- [Sah+24] S. Sahoo, E. Arriola, A. Pal, and V. Kuleshov. “Simple and Effective Masked Diffusion Language Models”. In: *Advances in Neural Information Processing Systems*. Vol. 37. 2024. URL: <https://arxiv.org/abs/2406.07524>.
- [Shi+24] W. Shi, H. Huang, J. Watson, M. Dehghani, and M. Titsias. “Simplified and Generalized Masked Diffusion for Discrete Data”. In: *Advances in Neural Information Processing Systems*. Vol. 37. 2024. URL: <https://arxiv.org/abs/2406.04329>.

Appendix

A Non-negativity of the KL divergence

We prove Gibbs’ inequality: for any two densities q, p on $\mathbb{R}^d$,

$$\text{KL}(q \| p) \geq 0,$$

with equality if and only if $q = p$ almost everywhere. Throughout we assume q is absolutely continuous with respect to p (i.e. $p(x) > 0$ wherever $q(x) > 0$); otherwise $\text{KL}(q \| p) = +\infty$ and the inequality holds trivially.

The elementary inequality. The proof rests on a single fact about the logarithm:

$$\log u \leq u - 1 \quad \text{for all } u > 0,$$

with equality if and only if $u = 1$. This follows because $f(u) = u - 1 - \log u$ satisfies $f'(u) = 1 - 1/u$, so f is decreasing on $(0, 1)$ and increasing on $(1, \infty)$, with a unique minimum $f(1) = 0$. Hence $f(u) \geq 0$ everywhere, with equality only at $u = 1$.

Applying it to the divergence. Write the divergence with the ratio p/q inside the logarithm:

$$\text{KL}(q \| p) = \int q(x) \log \frac{q(x)}{p(x)} dx = - \int q(x) \log \frac{p(x)}{q(x)} dx.$$

Apply the elementary inequality with $u = p(x)/q(x)$ at each point x in the support of q :

$$\log \frac{p(x)}{q(x)} \leq \frac{p(x)}{q(x)} - 1.$$

Multiplying by $q(x) \geq 0$ and integrating preserves the inequality:

$$\int q(x) \log \frac{p(x)}{q(x)} dx \leq \int q(x) \left(\frac{p(x)}{q(x)} - 1 \right) dx = \int p(x) dx - \int q(x) dx = 1 - 1 = 0.$$

Negating both sides flips the inequality:

$$\text{KL}(q \| p) = - \int q(x) \log \frac{p(x)}{q(x)} dx \geq 0.$$

The equality case. Equality holds in the integrated bound if and only if the pointwise inequality $\log u \leq u - 1$ is tight for q -almost every x , i.e. $u = p(x)/q(x) = 1$ wherever $q(x) > 0$. This means $p(x) = q(x)$ almost everywhere on the support of q ; since both densities integrate to one, they must agree almost everywhere. Therefore

$$\text{KL}(q \| p) = 0 \iff q = p \text{ almost everywhere. } \square$$

B KL divergence between equal-covariance Gaussians

Let $p = \mathcal{N}(\mu_1, \sigma^2 I)$ and $q = \mathcal{N}(\mu_2, \sigma^2 I)$ be two Gaussians on $\mathbb{R}^d$ with the same isotropic covariance $\sigma^2 I$. We show

$$\text{KL}(p \| q) = \frac{1}{2\sigma^2} \|\mu_1 - \mu_2\|^2.$$

The log-density ratio. The two densities differ only in their means, so the normalizing constants $(2\pi\sigma^2)^{-d/2}$ are identical and cancel in the ratio:

$$\log \frac{p(x)}{q(x)} = -\frac{1}{2\sigma^2} \|x - \mu_1\|^2 + \frac{1}{2\sigma^2} \|x - \mu_2\|^2 = \frac{1}{2\sigma^2} (\|x - \mu_2\|^2 - \|x - \mu_1\|^2).$$

Expanding both squared norms, the $\|x\|^2$ terms cancel:

$$\|x - \mu_2\|^2 - \|x - \mu_1\|^2 = -2x^\top \mu_2 + \|\mu_2\|^2 + 2x^\top \mu_1 - \|\mu_1\|^2 = 2x^\top (\mu_1 - \mu_2) + \|\mu_2\|^2 - \|\mu_1\|^2.$$

Hence

$$\log \frac{p(x)}{q(x)} = \frac{1}{2\sigma^2} (2x^\top (\mu_1 - \mu_2) + \|\mu_2\|^2 - \|\mu_1\|^2).$$

Taking the expectation under p . By definition $\text{KL}(p \| q) = \mathbb{E}_{X \sim p} \left[\log \frac{p(X)}{q(X)} \right]$. The only random term is $x^\top (\mu_1 - \mu_2)$, which is linear in x , so the expectation passes through to $\mathbb{E}_{X \sim p} [X] = \mu_1$:

$$\mathbb{E}_{X \sim p} [X^\top (\mu_1 - \mu_2)] = \mu_1^\top (\mu_1 - \mu_2).$$

Therefore

$$\text{KL}(p \| q) = \frac{1}{2\sigma^2} (2\mu_1^\top (\mu_1 - \mu_2) + \|\mu_2\|^2 - \|\mu_1\|^2).$$

Simplifying the means. Collect the mean terms inside the parentheses:

$$2\mu_1^\top (\mu_1 - \mu_2) + \|\mu_2\|^2 - \|\mu_1\|^2 = 2\|\mu_1\|^2 - 2\mu_1^\top \mu_2 + \|\mu_2\|^2 - \|\mu_1\|^2 = \|\mu_1\|^2 - 2\mu_1^\top \mu_2 + \|\mu_2\|^2,$$

which is exactly $\|\mu_1 - \mu_2\|^2$. Substituting back,

$$\text{KL}(p \| q) = \frac{1}{2\sigma^2} \|\mu_1 - \mu_2\|^2. \quad \square$$

C Closed-form marginal of the forward process

We prove that iterating the per-step forward kernel

$$q(z_t | z_{t-1}) = \mathcal{N}(z_t; \sqrt{1 - \beta_t} z_{t-1}, \beta_t I), \quad z_0 := x,$$

yields the closed-form marginal

$$q(z_t | x) = \mathcal{N}(z_t; \sqrt{\bar{\alpha}_t} x, (1 - \bar{\alpha}_t) I), \quad \alpha_t := 1 - \beta_t, \quad \bar{\alpha}_t := \prod_{s=1}^t \alpha_s.$$

Reparameterized form of one step. By the reparameterization trick (Fact 3), the per-step kernel is equivalent to the deterministic update

$$z_t = \sqrt{\alpha_t} z_{t-1} + \sqrt{1 - \alpha_t} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I) \text{ independent across } t,$$

since $\sqrt{1 - \beta_t} = \sqrt{\alpha_t}$ and the added noise has variance $\beta_t = 1 - \alpha_t$. We prove the claim by induction on t .

Base case. At $t = 1$, the update reads $z_1 = \sqrt{\alpha_1} x + \sqrt{1 - \alpha_1} \epsilon_1$. Since $\bar{\alpha}_1 = \alpha_1$, this is $z_1 = \sqrt{\alpha_1} x + \sqrt{1 - \alpha_1} \epsilon_1$, i.e. $q(z_1 | x) = \mathcal{N}(\sqrt{\alpha_1} x, (1 - \alpha_1)I)$, as claimed.

Inductive step. Assume the marginal holds at step $t - 1$, i.e. by reparameterization

$$z_{t-1} = \sqrt{\bar{\alpha}_{t-1}} x + \sqrt{1 - \bar{\alpha}_{t-1}} \bar{\epsilon}_{t-1}, \quad \bar{\epsilon}_{t-1} \sim \mathcal{N}(0, I),$$

where $\bar{\epsilon}_{t-1}$ is independent of the fresh noise ϵ_t used at step t . Substitute this into the one-step update:

$$z_t = \sqrt{\alpha_t} z_{t-1} + \sqrt{1 - \alpha_t} \epsilon_t = \sqrt{\alpha_t} \left(\sqrt{\bar{\alpha}_{t-1}} x + \sqrt{1 - \bar{\alpha}_{t-1}} \bar{\epsilon}_{t-1} \right) + \sqrt{1 - \alpha_t} \epsilon_t.$$

Collecting the signal and noise terms,

$$z_t = \underbrace{\sqrt{\alpha_t \bar{\alpha}_{t-1}}}_{= \sqrt{\bar{\alpha}_t}} x + \underbrace{\sqrt{\alpha_t (1 - \bar{\alpha}_{t-1})} \bar{\epsilon}_{t-1} + \sqrt{1 - \alpha_t} \epsilon_t}_{=: \eta_t},$$

where the signal coefficient simplifies using $\alpha_t \bar{\alpha}_{t-1} = \bar{\alpha}_t$.

Combining the two noise terms. The term η_t is a sum of two independent zero-mean Gaussians. By Fact 1 (linear scaling) each has covariance

$$\text{Cov}(\sqrt{\alpha_t (1 - \bar{\alpha}_{t-1})} \bar{\epsilon}_{t-1}) = \alpha_t (1 - \bar{\alpha}_{t-1}) I, \quad \text{Cov}(\sqrt{1 - \alpha_t} \epsilon_t) = (1 - \alpha_t) I,$$

and by Fact 2 (variances of independent Gaussians add), $\eta_t \sim \mathcal{N}(0, v_t I)$ with

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

again using $\alpha_t \bar{\alpha}_{t-1} = \bar{\alpha}_t$. Therefore

$$z_t = \sqrt{\bar{\alpha}_t} x + \sqrt{1 - \bar{\alpha}_t} \eta'_t, \quad \eta'_t \sim \mathcal{N}(0, I),$$

which is exactly $q(z_t | x) = \mathcal{N}(\sqrt{\bar{\alpha}_t} x, (1 - \bar{\alpha}_t)I)$. By induction the formula holds for all t . $\square$

D The score and the denoiser: Tweedie's formula

Three parameterizations of the denoiser network (clean data, noise and the score) are asserted equivalent. Two of the three relations are elementary algebra; one is not. This appendix derives the nontrivial one: that the score of the forward marginal is

$$\nabla_{z_t} \log p_t(z_t) = -\frac{\hat{\epsilon}_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}_t}} = -\frac{z_t - \sqrt{\bar{\alpha}_t} \hat{x}_\theta(z_t, t)}{1 - \bar{\alpha}_t}.$$

The key fact is *Tweedie's formula*: the score of a Gaussian-corrupted marginal is the posterior mean of the clean signal, in disguise.

The denoiser is a posterior mean. The training loss is the squared error $L(\theta) = \mathbb{E} \|\hat{x}_\theta(z_t, t) - x\|^2$. The function of z_t that minimizes mean squared error is the conditional expectation: for any joint distribution of (x, z_t) ,

$$\arg \min_f \mathbb{E} [\|x - f(z_t)\|^2] = \mathbb{E}[x | z_t].$$

(To see this, write $\mathbb{E} \|x - f(z_t)\|^2 = \mathbb{E} \mathbb{E} [\|x - f(z_t)\|^2 | z_t]$ and minimize the inner expectation pointwise in z_t ; the minimizer of $\mathbb{E} [\|x - c\|^2 | z_t]$ over the constant c is $c = \mathbb{E}[x | z_t]$.) Hence at the optimum the clean-data network recovers the posterior mean:

$$\hat{x}_\theta(z_t, t) \approx \mathbb{E}[x | z_t].$$

This is the only place the training objective enters. Everything below is a statement about the true marginal p_t and holds for this optimal denoiser.

The forward marginal and its kernel. Recall the Gaussian forward kernel and the marginal it induces:

$$q(z_t | x) = \mathcal{N}(z_t; \sqrt{\bar{\alpha}_t} x, (1 - \bar{\alpha}_t)I), \quad p_t(z_t) = \int q(z_t | x) p_{\text{data}}(x) dx.$$

We will need the gradient of the log-kernel with respect to z_t . Since $\log q(z_t | x) = -\frac{1}{2(1-\bar{\alpha}_t)} \|z_t - \sqrt{\bar{\alpha}_t} x\|^2 + \text{const}$,

$$\nabla_{z_t} \log q(z_t | x) = -\frac{z_t - \sqrt{\bar{\alpha}_t} x}{1 - \bar{\alpha}_t},$$

and therefore, as a plain gradient (not log-gradient),

$$\nabla_{z_t} q(z_t | x) = q(z_t | x) \nabla_{z_t} \log q(z_t | x) = -q(z_t | x) \frac{z_t - \sqrt{\bar{\alpha}_t} x}{1 - \bar{\alpha}_t}.$$

Differentiating the log-marginal. Apply the log-derivative identity $\nabla \log p_t = (\nabla p_t)/p_t$ and differentiate under the integral sign:

$$\nabla_{z_t} \log p_t(z_t) = \frac{\nabla_{z_t} p_t(z_t)}{p_t(z_t)} = \frac{1}{p_t(z_t)} \int \nabla_{z_t} q(z_t | x) p_{\text{data}}(x) dx.$$

Substitute the kernel gradient from the previous step:

$$\nabla_{z_t} \log p_t(z_t) = \frac{1}{p_t(z_t)} \int \left(-\frac{z_t - \sqrt{\bar{\alpha}_t} x}{1 - \bar{\alpha}_t} \right) q(z_t | x) p_{\text{data}}(x) dx.$$

Recognizing the posterior. The ratio $\frac{q(z_t | x) p_{\text{data}}(x)}{p_t(z_t)}$ is, by Bayes' rule, exactly the posterior $p(x | z_t)$. Pulling the z_t -dependent factors outside the integral (they do not depend on the integration variable x):

$$\nabla_{z_t} \log p_t(z_t) = -\frac{1}{1 - \bar{\alpha}_t} \int (z_t - \sqrt{\bar{\alpha}_t} x) p(x | z_t) dx = -\frac{z_t - \sqrt{\bar{\alpha}_t} \mathbb{E}[x | z_t]}{1 - \bar{\alpha}_t}.$$

This is Tweedie's formula for the variance-preserving kernel: *the score is the negative, variance-normalized residual between the noisy observation z_t and the posterior-mean reconstruction $\mathbb{E}[x | z_t]$* . The intractable integral over the data prior has collapsed into a single conditional expectation. Note the prior p_{data} has disappeared from the expression entirely; it survives only implicitly, inside $\mathbb{E}[x | z_t]$.

Inserting the trained denoiser. Replacing the posterior mean by the optimal denoiser, $\mathbb{E}[x | z_t] \approx \hat{x}_\theta(z_t, t)$, gives the clean-data form of the score:

$$\nabla_{z_t} \log p_t(z_t) = -\frac{z_t - \sqrt{\bar{\alpha}_t} \hat{x}_\theta(z_t, t)}{1 - \bar{\alpha}_t}.$$

The noise form is one line of algebra. The remaining relation needs no probability. The noise predictor is *defined* by solving the forward equation $z_t = \sqrt{\bar{\alpha}_t} x + \sqrt{1 - \bar{\alpha}_t} \epsilon$ for ϵ , with $\hat{x}_\theta$ in place of x :

$$\hat{\epsilon}_\theta(z_t, t) = \frac{z_t - \sqrt{\bar{\alpha}_t} \hat{x}_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \iff z_t - \sqrt{\bar{\alpha}_t} \hat{x}_\theta(z_t, t) = \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}_\theta(z_t, t).$$

Substituting the right-hand identity into the boxed score collapses one factor of $\sqrt{1 - \bar{\alpha}_t}$:

$$\nabla_{z_t} \log p_t(z_t) = -\frac{\sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}_\theta(z_t, t)}{1 - \bar{\alpha}_t} = -\frac{\hat{\epsilon}_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}_t}}.$$

This is the score-to-noise relation. The three forms $\hat{x}_\theta$, $\hat{\epsilon}_\theta$, s_θ are thus a single object in three coordinates: clean-data prediction, noise prediction, and score differ only by the deterministic factors above.

Solving for the posterior mean. For completeness, rearranging Tweedie’s formula for $\mathbb{E}[x \mid z_t]$ exposes the “observation plus variance times score” shape of the classical identity:

$$\mathbb{E}[x \mid z_t] = \frac{z_t + (1 - \bar{\alpha}_t) \nabla_{z_t} \log p_t(z_t)}{\sqrt{\bar{\alpha}_t}}.$$

The factor $(1 - \bar{\alpha}_t)$ plays the role of the noise variance σ^2 in the textbook location model $z = x + \sigma\epsilon$, where Tweedie’s formula reads $\mathbb{E}[x \mid z] = z + \sigma^2 \nabla_z \log p(z)$. The extra $\sqrt{\bar{\alpha}_t}$ is the only difference, and it is exactly the signal-rescaling factor that the variance-preserving forward process applies to x but the location model does not.

Historical note. The identity is *Tweedie’s formula*, named for M. C. K. Tweedie. It was first reported by Robbins (1956), who credited personal correspondence with Tweedie, and was popularized by Efron (2011) in the context of empirical Bayes. The clean-data predictor $\hat{x}_\theta$ is called the Tweedie estimate because, as derived above, the MSE-optimal denoiser is the posterior mean, and Tweedie’s formula identifies that posterior mean with a score correction of the noisy observation.

E Derivation of the Boltzmann-tilted optimum

We derive the closed-form solution to the KL-regularized optimization problem

$$\pi^* = \arg \max_{\pi} \left[\mathbb{E}_{x \sim \pi} [r(x)] - \beta \text{KL}(\pi \parallel p_\theta) \right],$$

subject to the constraint that π is a probability density: $\int \pi(x) dx = 1$ and $\pi(x) \geq 0$.

Writing the objective as an integral Expanding the KL and the expectation, the objective is

$$F[\pi] := \int \pi(x) r(x) dx - \beta \int \pi(x) \log \frac{\pi(x)}{p_\theta(x)} dx = \int \pi(x) \left[r(x) - \beta \log \frac{\pi(x)}{p_\theta(x)} \right] dx.$$

The optimization is over a function π , not a vector. We need the calculus-of-variations analogue of Lagrange multipliers: differentiate the integrand pointwise with respect to $\pi(x)$ and set the result to zero, treating each value $\pi(x)$ at each point x as an independent variable.

The Lagrangian Introduce a Lagrange multiplier $\lambda \in \mathbb{R}$ for the normalization constraint. The Lagrangian is

$$\mathcal{L}[\pi, \lambda] = F[\pi] - \lambda \left(\int \pi(x) dx - 1 \right) = \int \underbrace{\left\{ \pi(x) r(x) - \beta \pi(x) \log \frac{\pi(x)}{p_\theta(x)} - \lambda \pi(x) \right\}}_{=: I(x; \pi(x))} dx + \lambda.$$

The non-negativity constraint $\pi(x) \geq 0$ will be satisfied automatically by the form of the solution and does not require its own multiplier.

The first-order condition To find the stationary point, differentiate the integrand $I(x; \pi(x))$ with respect to $\pi(x)$ at each point x separately. There are three terms.

Term 1: $\pi(x) r(x)$. This is linear in $\pi(x)$, so its derivative is $r(x)$.

Term 2: $-\beta \pi(x) \log(\pi(x)/p_\theta(x))$. Apply the product rule to $\pi(x) \log(\pi(x)/p_\theta(x))$:

$$\frac{d}{d\pi(x)} \left[\pi(x) \log \frac{\pi(x)}{p_\theta(x)} \right] = \log \frac{\pi(x)}{p_\theta(x)} + \pi(x) \cdot \frac{1}{\pi(x)} = \log \frac{\pi(x)}{p_\theta(x)} + 1.$$

The $+1$ comes from differentiating the $\pi(x)$ factor in front of the log; many students miss this on a first pass. Multiplying by $-\beta$, the derivative of Term 2 is

$$-\beta \log \frac{\pi(x)}{p_\theta(x)} - \beta.$$

Term 3: $-\lambda \pi(x)$. This is linear in $\pi(x)$, so its derivative is $-\lambda$.

Setting the sum to zero. Adding the three terms:

$$r(x) - \beta \log \frac{\pi(x)}{p_\theta(x)} - \beta - \lambda = 0.$$

Solving for $\pi(x)$ Rearranging:

$$\beta \log \frac{\pi(x)}{p_\theta(x)} = r(x) - \beta - \lambda \implies \log \frac{\pi(x)}{p_\theta(x)} = \frac{r(x)}{\beta} - 1 - \frac{\lambda}{\beta}.$$

Exponentiating both sides:

$$\pi(x) = p_\theta(x) \exp\left(\frac{r(x)}{\beta} - 1 - \frac{\lambda}{\beta}\right).$$

The factor $\exp(-1 - \lambda/\beta)$ does not depend on x . It is a constant that we will absorb into normalization. Writing

$$C := \exp\left(-1 - \frac{\lambda}{\beta}\right),$$

the solution is

$$\pi(x) = C p_\theta(x) \exp(r(x)/\beta).$$

Determining C from normalization The constant C is fixed by the constraint $\int \pi(x) dx = 1$:

$$C \int p_\theta(x) \exp(r(x)/\beta) dx = 1 \implies C = \frac{1}{Z},$$

where

$$Z := \int p_\theta(x) \exp(r(x)/\beta) dx$$

is the partition function. The Lagrange multiplier λ never needs to be solved for explicitly. It exists in the derivation, but it gets absorbed into the constant C , which is determined by the normalization constraint alone.

The final answer Substituting:

$$\pi^*(x) = \frac{1}{Z} p_\theta(x) \exp\left(\frac{r(x)}{\beta}\right), \quad Z = \int p_\theta(x) \exp(r(x)/\beta) dx.$$

This is the Boltzmann-tilted target stated in the main text.

Verifying it is a maximum The first-order condition gives a stationary point. To confirm it is a maximum and not a saddle or minimum, observe that the objective $F[\pi]$ is concave in π : the term $\mathbb{E}_\pi[r(x)]$ is linear in π , and $-\text{KL}(\pi \parallel p_\theta)$ is strictly concave in π (a standard property of KL divergence). The constraint set is convex (probability densities form a convex set). A strictly concave function over a convex constraint set has a unique maximum, which is the stationary point we found.

F Score-to-clean-data conversion in classifier guidance

We derive the explicit formula for the shifted clean-data predictor under classifier guidance:

$$\tilde{x}_\theta(z_t, t, y) = \hat{x}_\theta(z_t, t) + \frac{1 - \bar{\alpha}_t}{\sqrt{\bar{\alpha}_t}} \gamma \nabla_{z_t} \log p_\phi(y \mid z_t, t).$$

The derivation depends on the relation between the score and the clean-data prediction derived in Appendix D.

The guided score. Classifier guidance modifies the score of the marginal by adding the classifier gradient, scaled by γ :

$$s^\gamma(z_t, t, y) = \nabla_{z_t} \log p_t(z_t) + \gamma \nabla_{z_t} \log p_\phi(y \mid z_t, t).$$

This is the quantity we want the reverse step to use in place of the prior score. The trained network gives us $\hat{x}_\theta$, not a score, so we need the map from a score to the clean-data prediction that carries it.

Solving the score relation for $\hat{x}_\theta$ Appendix D gives the score of the marginal in terms of the denoiser. Solving that relation for the clean-data prediction:

$$\nabla_{z_t} \log p_t(z_t) = -\frac{z_t - \sqrt{\bar{\alpha}_t} \hat{x}_\theta}{1 - \bar{\alpha}_t} \iff \hat{x}_\theta = \frac{z_t + (1 - \bar{\alpha}_t) \nabla_{z_t} \log p_t(z_t)}{\sqrt{\bar{\alpha}_t}}.$$

The right-hand identity is general: any score, substituted on the right, returns the clean-data prediction associated with it. There is nothing special about the prior score in this conversion.

Applying it to the guided score. Substitute the guided score s^γ for $\nabla_{z_t} \log p_t$ and call the result $\tilde{x}_\theta$:

$$\tilde{x}_\theta = \frac{z_t + (1 - \bar{\alpha}_t) s^\gamma}{\sqrt{\bar{\alpha}_t}} = \frac{z_t + (1 - \bar{\alpha}_t) [\nabla_{z_t} \log p_t + \gamma \nabla_{z_t} \log p_\phi]}{\sqrt{\bar{\alpha}_t}}.$$

Split the bracket into the prior term and the guidance term. The prior term is the dictionary entry applied to the unmodified score, which is $\hat{x}_\theta$ by the identity above:

$$\tilde{x}_\theta = \underbrace{\frac{z_t + (1 - \bar{\alpha}_t) \nabla_{z_t} \log p_t}{\sqrt{\bar{\alpha}_t}}}_{=\hat{x}_\theta(z_t, t)} + \frac{1 - \bar{\alpha}_t}{\sqrt{\bar{\alpha}_t}} \gamma \nabla_{z_t} \log p_\phi(y \mid z_t, t),$$

The conversion factor $(1 - \bar{\alpha}_t)/\sqrt{\bar{\alpha}_t}$ is nothing but the score-to-clean-data coefficient from the dictionary entry: $(1 - \bar{\alpha}_t)$ from the noise variance, $1/\sqrt{\bar{\alpha}_t}$ from the signal rescaling.

Interpretation of the coefficient. The coefficient $(1 - \bar{\alpha}_t)/\sqrt{\bar{\alpha}_t}$ varies with t :

- At $t = 0$ (clean data), $\bar{\alpha}_0 = 1$, so the coefficient is $0/1 = 0$. The shift vanishes: there is no noise to denoise, so classifier guidance cannot meaningfully modify clean data.
- At $t = T$ (pure noise), $\bar{\alpha}_T \approx 0$, so the coefficient is very large. The classifier signal has maximum effect at high noise levels, where the diffusion model is most uncertain.

The schedule of the coefficient ensures the classifier signal grows in importance as we move backward through the chain. At low t the diffusion model is already confident and the classifier only fine-tunes; at high t the diffusion model has little information and the classifier determines which mode the trajectory converges to.