Controllable Generative AI for Biological Sequences

Jan Olszewski **Ewa Szczurek**

Institute of AI for Health, Helmholtz Munich
Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw

June 19, 2026

Outline

Mathematical Background

Continuous Diffusion Models

The Idealism-Realism Tradeoff

Guidance Methods

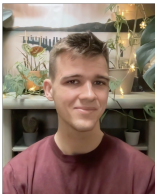
Discrete Diffusion

Guidance for Discrete Diffusion

Plus a bonus lecture on Applications to Biological Sequences.

Organisation

Ewa Szczurek (Lecture)



Jan Olszewski (Tutorial)

- **Website**: https://phdopen.mimuw.edu.pl/index.php?page=l26w4
- **Slack**: https://join.slack.com/t/phdopen2026di-66v4471/shared_invite/zt-4180w6k1u-~JOA1Uryl95PL7XzREI6qA
- **Lecture:** Slides accompanied by lecture notes,
- **Tutorial:** python notebooks, bring your laptops
- **Plan for tomorrow:** room 5440, 14:15 - 15:45 Lecture, 16:15 - 17:30 Tutorial

Continuous random variables and densities

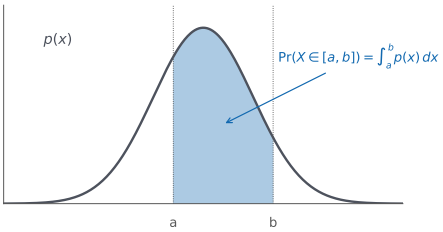
A continuous r.v. X in $\mathbb{R}^d$ has a density $p : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ with $\int p(x)\,\mathrm{d}x = 1$.

$p(x)$ is not a probability

Densities can exceed one. Probabilities come from integration:

$$\Pr(X \in A) \;=\; \int_A p(x)\,\mathrm{d}x.$$

Think of $p(x)$ telling how likely the sample x is. Crucial for generative models.



Useful facts about continuous random variables and densities

For two r.v.s X, Y, we will think of

- x as a **sample**, with **likelihood** $p(x)$, learned by the **generative model**
- y as a label, with **conditional likelihood** $p(y|x)$, learned by **classifier**.

The following holds for the joint $p(x, y)$:

- **Marginal**: $p(x) = \int p(x, y)\,\mathrm{d}y$
- **Conditional**: $p(x \mid y) = p(x, y)/p(y)$ where $p(y) > 0$
- **Product rule**: $p(x, y) = p(y|x)p(x)$.

Bayes' rule for densities

The identity from the discrete setting carries over to denisties:

$$\boxed{p(x \mid y) \;=\; \frac{p(y \mid x)\, p(x)}{p(y)}}$$

Taking log on both sides and differentiating w.r.t. x:

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

since $\nabla_x \log p(y) = 0$ (the marginal $p(y)$ does not depend on x).

Why this matters

The intractable normalizer $p(y) = \int p(y \mid x)\, p(x)\, \mathrm{d}x$ drops out of the gradient.

Expectations

For any function f:

$$\mathbb{E}[f(X)] = \int f(x)\, p(x)\, \mathrm{d}x, \qquad \mathbb{E}[f(X) \mid Y = y] = \int f(x)\, p(x \mid y)\, \mathrm{d}x.$$

Practical consequence

All training losses in this course will be expectations of squared errors or cross-entropies, approximated by Monte Carlo over training batches.

Gaussian algebra

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

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

Fact 1: Linear transformation

If $X \sim \mathcal{N}(\mu, \Sigma)$ and $a \in \mathbb{R}$, $b \in \mathbb{R}^d$ are constants:

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

Fact 2: Sum of independent Gaussians

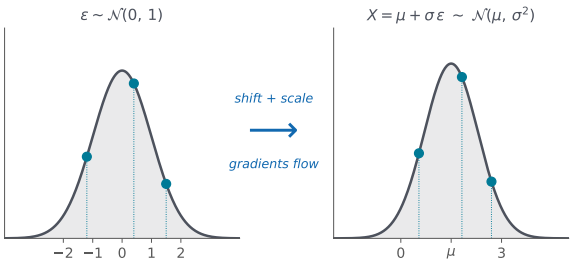
If $X \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $Y \sim \mathcal{N}(\mu_2, \Sigma_2)$ are independent:

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

Fact 3: The reparameterization trick

Reparameterization

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



Two payoffs:

- **Randomness outside:** X is a deterministic function of (μ, σ, ϵ).
- **Gradients flow** through it, enables backprop.

KL divergence

To train a probabilistic model we need to measure closeness of distributions.

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

- **Non-negative:** $\mathrm{KL}(q\|p) \geq 0$, with equality iff $q = p$ a.e.
- **Asymmetric:** a divergence, not a metric.
- **Loss-like:** minimizing $\mathrm{KL}(q \,\|\, p_\theta)$ in θ (with $q =$ data) recovers maximum likelihood.

The Gaussian case (same covariance)

For $p = \mathcal{N}(\mu_1, \sigma^2 I)$ and $q = \mathcal{N}(\mu_2, \sigma^2 I)$:

$$\mathrm{KL}(p \,\|\, q) \;=\; \frac{1}{2\sigma^2} \|\mu_1 - \mu_2\|^2 \quad \longrightarrow \quad \text{scaled squared error between means.}$$

The score function

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

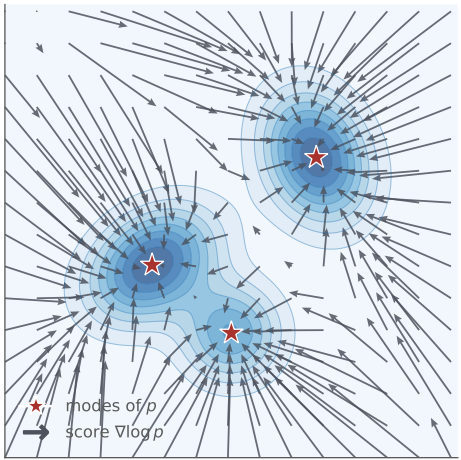
$$s(x) := \nabla_x \log p(x) \in \mathbb{R}^d.$$

Geometric meaning

$s(x)$ points in the direction of steepest increase of $\log p$.

- Following s moves samples *toward modes*.
- Score-based samplers follow this gradient, with added noise to avoid mode collapse.

$\nabla_x \log p(x)$ points toward the modes



The score: invariance to normalization

Suppose we only know p up to a constant: $p(x) = \tilde{p}(x)/Z$ with intractable $Z = \int \tilde{p}(x)\,\mathrm{d}x$. Then

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

because Z is constant in x.

Huge practical advantage: we can work with unnormalized distributions and still compute scores exactly.

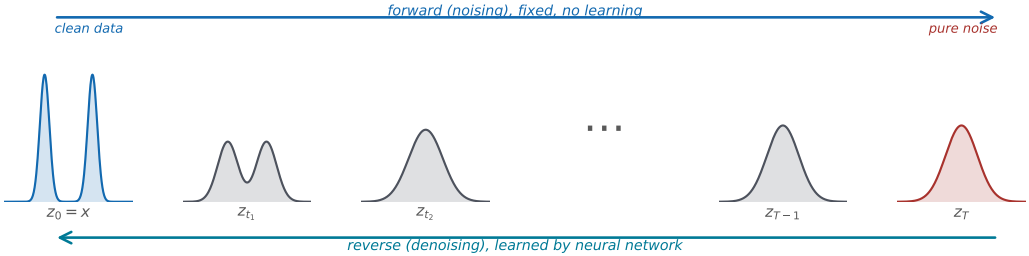
Why this matters for Bayesian guidance

Bayes gives $p(x \mid y) = p(y \mid x)\,p(x)/p(y)$ with intractable $p(y) = \int p(y \mid x)\,p(x)\,\mathrm{d}x$. Taking the score wipes it out:

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

Both right-hand-side terms are tractable: a classifier's gradient and the data score.

Diffusion in one picture



Forward: add Gaussian noise, T steps.
Fixed. No learning.

Reverse: undo the noise.
Learned by a neural net.

Forward process[1]

A Markov chain that gradually corrupts x over T steps:

$$q(z_t \mid z_{t-1}) \;=\; \mathcal{N}\Big(z_t;\, \sqrt{1-\beta_t}\, z_{t-1},\, \beta_t\, I\Big), \qquad z_0 := x.$$

- $\{\beta_t\}$: variance schedule, small per step (linear or cosine).
- Shrink by $\sqrt{1-\beta_t}$, add fresh noise of variance β_t:

$$\mathrm{Var}(z_t) = (1-\beta_t)\,\mathrm{Var}(z_{t-1}) + \beta_t \xrightarrow{\mathrm{Var}(z_{t-1})=1} \mathrm{Var}(z_t) = 1.$$

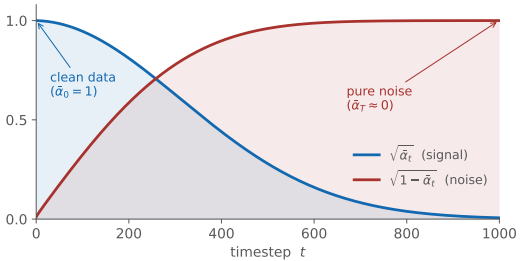
  Coefficients sum to one, so variance stays 1 (does not explode).
- Typical $T = 1000$: many small steps make each denoising decision easy.

[1] Ho et al. (2020) *Denoising Diffusion Probabilistic Models* (DDPM)

Closed-form marginal: the workhorse equation

Define $\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. Iterating the linear-Gaussian step gives:

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



- Jump to any noise level in one step, no chain simulation.
- $\sqrt{\bar{\alpha}_t}$: signal kept. $\sqrt{1 - \bar{\alpha}_t}$: noise injected.
- $\bar{\alpha}_0 = 1$, $\bar{\alpha}_T \approx 0$, which justifies starting reverse from $\mathcal{N}(0, I)$.

Reverse process is intractable

We want $q(z_{t-1} \mid z_t)$.
From the Bayes rule

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

This is intractable

The marginals $q(z_{t-1})$ and $q(z_t)$ involve integrating over the entire data distribution.

Reverse process: the key trick

But if we additionally condition on x...

$q(z_{t-1})$ and $q(z_t)$ replaced by the closed-form Gaussians $q(z_{t-1} \mid x)$ and $q(z_t \mid x)$, and

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

is also a closed-form Gaussian, with mean

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

and variance

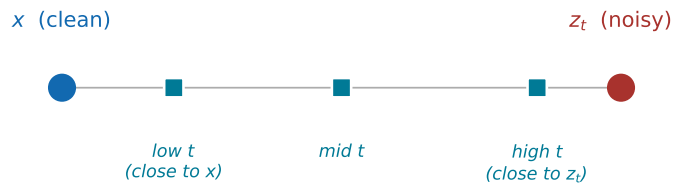
$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t \le \beta_t \qquad \text{(conditioning on } x \text{ shrinks the variance).}$$

The posterior mean: a convex combination

The coefficients are nonnegative and approx. sum to one (exactly so as $\beta_t \to 0$), making $\tilde{\mu}_t$ a convex combination of x and z_t.

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

$\tilde{\mu}_t(z_t, x)$ is a convex combination of x and z_t



Training: predict x from z_t

At sampling time we don't have x. So: predict it.

Train a network $\hat{x}_\theta(z_t, t)$ to recover the clean data from the noisy state:

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

$\mathcal{L}(\theta)$ follows from the variational bound (ELBO) on $\log p_\theta(x)$, after the KL terms collapse to squared errors.

Recipe per training example

Sample (x, t, ϵ) $\rightarrow$ form z_t $\rightarrow$ predict $\hat{x}_\theta(z_t, t)$ $\rightarrow$ MSE loss.

Sampling: DDPM ancestral algorithm

Algorithm

1. Sample $z_T \sim \mathcal{N}(0, I)$.
2. For $t = T, T-1, \ldots, 1$:
   - Compute $\hat{x}_\theta(z_t, t)$ (one forward pass)
   - Compute posterior mean $\mu_\theta = \tilde{\mu}_t\big(z_t, \hat{x}_\theta(z_t, t)\big)$
   - Sample $z_{t-1} \sim \mathcal{N}\Big(\mu_\theta, \tilde{\beta}_t I\Big)$
3. Return z_0.

Cost: T forward passes per sample. For $T = 1000$, $\sim 1000\times$ slower than a VAE/GAN.

Each reverse step is a small correction. Many small denoising decisions beat one big leap from noise to x.

Three equivalent parameterizations

The trained network can be written in three forms, algebraically interchangeable:

Form	Predicts	Conversion
$\hat{x}_\theta(z_t, t)$	clean data x (Tweedie estimate)	–
$\hat{\epsilon}_\theta(z_t, t)$	the noise ϵ	$\hat{\epsilon}_\theta = \dfrac{z_t - \sqrt{\bar{\alpha}_t}\,\hat{x}_\theta}{\sqrt{1-\bar{\alpha}_t}}$
$s_\theta(z_t, t)$	score $\nabla_{z_t} \log p_t$	$s_\theta = -\dfrac{\hat{\epsilon}_\theta}{\sqrt{1-\bar{\alpha}_t}}.$

- $\hat{x} \leftrightarrow \hat{\epsilon}$: pure algebra from the forward marginal $z_t = \sqrt{\bar{\alpha}_t}\,x + \sqrt{1-\bar{\alpha}_t}\,\epsilon$.
- $\hat{\epsilon} \to s_\theta$: Tweedie's formula, the score of the noised marginal is the rescaled noise, $\nabla_{z_t} \log p_t = -\,\hat{\epsilon}_\theta / \sqrt{1-\bar{\alpha}_t}$.

A remark: the continuous-time view

Everything so far was discrete time, $t = 1, \ldots, T$. Letting the steps shrink ($\beta_t \to 0$) opens a second formulation. We only sketch it here.

- **Reverse SDE.** The small-step limit of the forward chain is a stochastic differential equation; **DDPM is one discretization of it**.
- **Probability-flow ODE.** A *deterministic* process (no injected noise) with the *same marginals* p_t, hence the same endpoint p_{data}. Closely related to **flow matching**.
- **Same trained network** drives both: no retraining, only the sampling changes.

Why it matters

- Faster sampling: **DDIM**[2] $\sim$ 50 steps, **DPM-Solver**[3] $\sim$ 10–20 steps, vs. DDPM's $\sim$ 1000.
- The deterministic ODE enables **exact likelihoods**.

[2] Song et al. (2021)
[3] Lu et al. (2022)

The idealism–realism tradeoff

We have two competing needs:

Realism

Modeled by the prior $p_\theta(x) \approx p(x)$ gives plausible data: *any* peptide, molecule, image.

Idealism

Modeled by a reward $r(x) \geq 0$ scores how *desirable* an output is (binding affinity, antimicrobial activity, prompt match).

The question

How do we modify a generative model to favour high-reward outputs while keeping samples close to the data distribution?

KL-regularized optimization

Naive: maximize the reward.

This is degenerate

Returns a delta on the highest-reward point, which may not even be realistic (r is reliable only on the data manifold).

Fix: regularize π toward the prior with a KL penalty:

$$\pi^* = \arg\max_{\pi} \Big[\mathbb{E}_{x\sim\pi}[r(x)] - \beta\, \mathrm{KL}(\pi \,\|\, p_\theta) \Big]$$

$\beta > 0$ trades off reward against proximity to the prior:

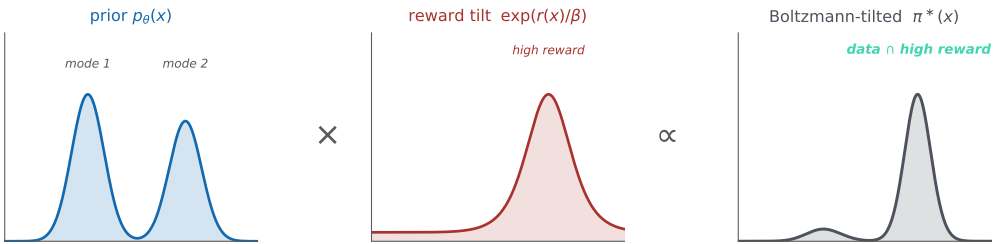
- Small β: aggressive; risks off-manifold samples, reward hacking.
- Large β: stay close to prior; ignore the reward.

Same form as KL-regularized RL. In physics: inverse temperature.

The Boltzmann-tilted target

The KL-regularized objective has a closed-form optimum

$$\pi^*(x) \;=\; \frac{1}{Z}\, p_\theta(x) \;\cdot\; \exp\!\big(r(x)/\beta\big), \qquad Z = \int p_\theta(x) \exp(r(x)/\beta)\, \mathrm{d}x.$$



Both have a vote: a sequence with high r but low p_θ is suppressed by the prior; a sequence iwith high p_θ but low r is suppressed by the reward.

Two special cases of the same framework

Bayesian posterior

Take $r(x) = \log p(y \mid x)$ and $\beta = 1$:

$$\pi^*(x) \;\propto\; p_\theta(x)\, p(y \mid x) \;=\; p_\theta(x \mid y).$$

The Boltzmann target *is* the Bayesian posterior.
Classifier guidance exemplifies this case.

Hard constraints

Take $r(x) = \log \mathbf{1}[x \in S]$ (0 on S, $-\infty$ off it), so $\exp(r(x)/\beta) = \mathbf{1}[x \in S]$:

$$\pi^* \propto p_\theta(x)\, \mathbf{1}[x \in S] \;\text{ concentrates on } S \cap \operatorname{supp}(p_\theta).$$

Realistic outputs satisfying a hard constraint.

The obstacle and the score trick

The particion function tractability obstacle

The partition function $Z = \int p_\theta(x) \exp(r(x)/\beta)\, \mathrm{d}x$ is intractable in high dimensions. We know π^* only up to the unknown Z. Cannot evaluate or directly sample.

The score saves us

Take $\nabla_x \log$ on both sides. Z disappears:

$$\nabla_x \log \pi^*(x) \;=\; \nabla_x \log p_\theta(x) \;+\; \tfrac{1}{\beta}\, \nabla_x r(x)$$

Additive decomposition: prior score + scaled reward gradient.
Both pieces are tractable. Used by some guidance methods.

Time-dependent extension: the value function u_t

Diffusion gives a *family* p_t for $t \in [0, T]$.

We need a family of tilted targets:

$$q_t^*(z_t) \;\propto\; p_t(z_t)\, u_t(z_t), \qquad u_t(z_t) := \mathbb{E}_P[\exp(r(Z_0)/\beta) \mid Z_t = z_t]\,.$$

u_t asks: *starting from z_t now, what is the expected exponentiated reward if the prior dynamics run to completion?*

- P is the prior path measure: the joint distribution of whole trajectories $(Z_t)_{t\in[0,T]}$ induced by running the diffusion model. The expectation in u_t is taken over these trajectories, those that pass through z_t at time t.
- Boundary: $u_0(z_0) = \exp(r(z_0)/\beta)$, which recovers the static tilt.
- Known by many names: value function (RL), *desirability function* (stochastic optimal control), *Schrödinger potential* (Schrödinger bridges).

The time-dependent score decomposition

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

At $t = 0$, $\log u_0 = r/\beta$, recovering the static decomposition.

u_t is not directly computable

$\rightarrow$ Guidance methods are different approximations to u_t or $\nabla \log u_t$.

Refresher of key takeaways

The machinery (continuous diffusion)

- **Forward marginal**: $q(z_t \mid x) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\, x,\ (1-\bar{\alpha}_t) I)$.
- One network,**three views**: $\hat{x}_\theta$ (posterior mean, Tweedie), noise $\hat{\epsilon}_\theta = (z_t - \sqrt{\bar{\alpha}_t}\, \hat{x}_\theta)/\sqrt{1-\bar{\alpha}_t}$, score $s_\theta = -\hat{\epsilon}_\theta/\sqrt{1-\bar{\alpha}_t}$.
- **Reverse step**: substitute $\hat{x}_\theta$ into the Gaussian posterior $q(z_{t-1} \mid z_t, x)$ & sample.

The goal (idealism-realism)

- **Tilted target**: $q_0^*(x) \propto p_0(x) \exp(r(x)/\beta)$, family $q_t^* \propto p_t\, u_t$.
- **Score decomposition**: $\nabla_{z_t} \log q_t^* = \nabla_{z_t} \log p_t + \nabla_{z_t} \log u_t$.

Next: controllability mechanisms

Ways to estimate the guidance term $\nabla_{z_t} \log u_t$ (or to avoid it).

Plain conditioning, the simplest mechanism

Enlarge the denoiser's input with a conditioning vector c:

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

Training and sampling

- Loss: same MSE, threaded with c: $\mathcal{L}(\theta) = \mathbb{E}_{(x,c),t,\epsilon}\left[\, \|\hat{x}_\theta(z_t, t, c) - x\|^2 \,\right]$
- Training data: (x, c) pairs.
- Sampling: identical algorithm, with desired c fed in at every step.

How conditioning enters the network

Three standard mechanisms. The choice does not change the loss or sampler, only how c is presented.

Concatenation

Append c (or an embedding) to the input or to intermediate features. *Simple, robust; works when c is a low-dimensional vector.*

Cross-attention

Treat c as a sequence of tokens. Hidden states query the tokens of c. *Essential when c is itself a sequence (text prompt).* Stable Diffusion, Imagen.

Feature-wise Linear Modulation(FiLM) / adaptive normalization

Produce per-channel scale and shift from c; modulate normalization layers. *Compact, expressive.* Ablated Diffusion Model (ADM)[4]

[4] Dhariwal & Nichol (2021)

The score-level Bayes identity

Specialized to diffusion latents at noise level t:

$$\nabla_{z_t} \log p_t(z_t \mid y) \;=\; \nabla_{z_t} \log p_t(z_t) \;+\; \nabla_{z_t} \log p_t(y \mid z_t)$$

Two tractable pieces:

- $\nabla_{z_t} \log p_t(z_t)$: the *prior score*, from the diffusion model.
- $\nabla_{z_t} \log p_t(y \mid z_t)$: the gradient of a separately trained classifier.

Consequence

An unconditional diffusion model + a classifier is enough to sample conditionally, with no conditional generative training.
This is **classifier guidance**. The classifier gradient plays the role of $\nabla \log u_t$.

Classifier guidance and the guidance scale γ

Dhariwal & Nichol (2021): amplify the classifier term with a scale $\gamma \geq 0$:

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

Equivalent to sampling from $p_t^{\gamma}(z_t \mid y) \propto p_t(z_t) \cdot p_t(y \mid z_t)^{\gamma}$.

Three regimes

- $\gamma = 0$: unconditional sampling. Classifier ignored.
- $\gamma = 1$: the exact Bayesian posterior $p_t(z_t \mid y)$.
- $\gamma > 1$: sharpened toward modes where the classifier is most confident.

Typical $\gamma \in [1, 10]$. Higher γ: better alignment, less diversity, eventually artifacts.
Setting $\gamma > 1$ is a useful *heuristic*, not exact posterior sampling.

Training the classifier

The classifier must operate on *noisy* z_t at every level: train it noise-conditionally.

Off-the-shelf clean-data classifier fails

On high-noise z_t it produces arbitrary outputs; its gradient is adversarial noise. The classifier must see the noised distribution itself.

Recipe: noise-conditional classifier $p_\phi(y \mid z_t, t)$

Sample (x, y), t, ϵ. Form z_t. Train p_ϕ by cross-entropy on (z_t, t, y).

Classifier-guided sampling

Shift the clean-data predictor by the classifier gradient

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

then take the standard DDPM reverse step using $\tilde{x}_\theta$.
Cost: one diffusion forward + one classifier forward+backward per step.

Reconstruction guidance, for clean-data classifiers

The practical issue

Most scientific property predictors take *clean* inputs, e.g. binding affinity for molecules. Retraining a noise-conditional version is expensive or impossible.

The Diffusion Posterior Sampling (DPS) trick[5]

Evaluate a clean-data classifier $p_\phi(y \mid x)$ on the Tweedie estimate $\hat{x}_\theta(z_t, t)$, then backprop:

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

- **Cost**: backprop through *both* classifier and diffusion network.
- **Approximation**: biased at high noise (poor $\hat{x}_\theta$), *usually mild in practice*.
- Enables off-the-shelf property predictors with pretrained diffusion models.

[5] Chung et al., (2023)

The reverse Bayes trick

Apply Bayes the *other* way. Starting from $p(y \mid x) = p(x \mid y)\, p(y)/p(x)$:

$$\boxed{\nabla_{z_t} \log p_t(y \mid z_t) \;=\; \nabla_{z_t} \log p_t(z_t \mid y) \;-\; \nabla_{z_t} \log p_t(z_t)}$$

The classifier gradient as a difference of two scores:

- Conditional score (diffusion net in conditional mode).
- Unconditional score (same net in unconditional mode).

Consequence

If a single diffusion network can act in both modes, the classifier gradient comes for free.
No separate classifier to train. $\rightarrow$ **Classifier-free guidance**.[6]

[6] Ho & Salimans, (2022)

Classifier-free guidance: training

Train one diffusion network to act in *both* conditional and unconditional modes via conditioning dropout.

Training recipe

During training, with probability p_{drop} (typically 0.1–0.2), replace c with a special null token $\varnothing$ before passing it to the network. Otherwise pass the real c. The loss is unchanged.

After training, the same network functions in two modes:

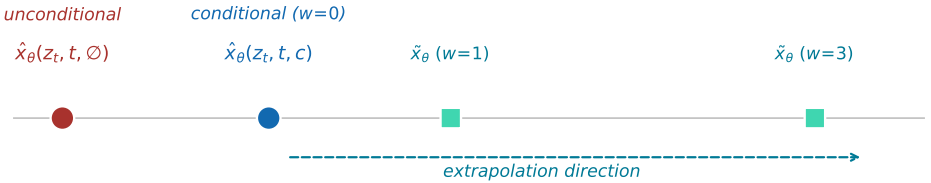
$$\hat{x}_\theta(z_t, t, c) \quad \text{approximates the conditional score}$$
$$\hat{x}_\theta(z_t, t, \varnothing) \quad \text{approximates the unconditional score}$$

Training cost: identical to plain conditional. One extra line of code.

CFG: sampling formula and geometric picture

In the clean-data parameterization, the guided predictor is a linear combination:

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



- Extrapolation, not interpolation.
- $w = 0$: standard conditional.
- $w \in [1, 7]$: typical for text-to-image.

Cost: 2 forward passes per step (one with c, one with $\varnothing$). No backward passes.

Sequential Monte Carlo steering: the idea

A *different paradigm*: not score modification within one trajectory, but many trajectories reweighted by reward.

Target

$$q_0^*(x) \propto p_0(x)\, \exp(r(x)/\beta), \qquad q_t^*(z_t) \propto p_t(z_t)\, u_t(z_t)$$

u_t: value function, expected $\exp(r/\beta)$ over completions from z_t. Boundary $u_0 = \exp(r/\beta)$. Reweight intermediate samples by *potential function* g_t.

Optimal potential $g_t^* = u_t$ is intractable.

But the intermediate g_t cancel across the trajectory, so any g_t with $g_0 \propto u_0$ reaches q_0^*. Freedom to pick a cheap, heuristic g_t.

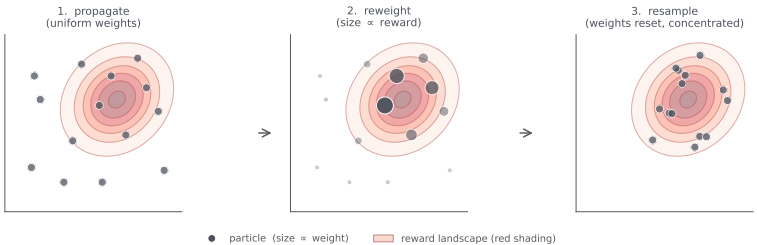
SMC algorithm

Algorithm (weight steering version)

Init $z_T^{(i)} \sim \mathcal{N}(0, I)$, $w^{(i)} = 1/N$. For $t = T, \dots, 1$:

1. **Propagate:** $z_{t-1}^{(i)} \sim p_\theta(z_{t-1} \mid z_t^{(i)})$.
2. **Reweight:** $w^{(i)} \leftarrow w^{(i)} \cdot g_{t-1}(z_{t-1}^{(i)}) \,/\, g_t(z_t^{(i)})$.
3. **Resample:** if $\mathrm{ESS} < N/2$, resample by weight, reset to $1/N$.

Return $\{(z_0^{(i)}, w^{(i)})\}$. Sample: draw one particle $\propto w^{(i)}$.



SMC: resampling

Effective sample size

Each reweight multiplies by a random factor, so weights spread with the number of steps: a few particles capture all the weight.

$$\mathrm{ESS} = \frac{\left(\sum_i w^{(i)}\right)^2}{\sum_i (w^{(i)})^2} \in [1, N].$$

N: uniform. 1: one particle dominates. Resample when $\mathrm{ESS} < N/2$.

Resampling restores effective sample size

Duplicates high-weight particles, culls the rest. Identical copies re-separate at the next step (stochastic kernel). Weight becomes multiplicity, target unchanged.

SMC: potential

Choosing g_t (bias-variance, all valid as $N \to \infty$)

- **Terminal** ($g_t \equiv 1$, $t > 0$): no bias via $\hat{x}_\theta$, highest variance.
- **Rollout** ($g_t = \exp(r(\hat{x}_\theta)/\beta)$): lower variance, biased at high t.
- **Learned twist** (Twisted Diffusion Sampler, TDS[7]), $g_t \approx u_t$: lowest variance, training cost. Also steers the proposal.

The consequence

- No gradients. Reward enters as a weight, never a derivative.
- Works for **non-differentiable** rewards: simulators, oracles, energy functions.
- Asymptotically exact as $N \to \infty$.

[7] Wu et al. 2023

Comparing the four methods

	Plain cond.	**Classifier g.**	**CFG**	**SMC**
Acts at	training	inference	training + inf.	inference
Mechanism	conditional NN	score modify	score modify	particle reweight
Approx. of u_t	implicit	explicit (clf. grad.)	implicit	none (asymp. exact)
Training data	(x, c) pairs	unlab. x + (x, y) clf.	(x, c) + dropout	unlab. x + scorer
Reward diff.?	N/A	required	N/A	not required
Grad. at inference?	no	yes	no	no
Cost / step	$1\times$	1 fwd + clf fwd/bwd	$2\times$	$N\times$
Inference knob	none	γ	w	N, potential g_t
Exact at	N/A	$\gamma = 1$	$w = 0$	$N \to \infty$
Failure mode	weak control	adversarial gradient	sharpened, not posterior	mode collapse

Why discrete data is different

Biological sequences live in *finite alphabets*: 20 amino acids, 4 nucleotides. Continuous diffusion assumes data in $\mathbb{R}^d$.

The central obstacle

$\nabla_x \log p(x)$ is undefined on a discrete state space.
The whole score-based machinery does not transfer directly.

What carries over vs. what changes

- Carries over: forward corruption, learned reverse denoising, many small steps, *predict-clean-data* training trick.
- Changes: forward kernel (no Gaussian), training objective (no score matching), some guidance methods (no classifier guidance via gradients).

Setup and the two main families (D3PM)

Let V = finite alphabet. Extend with a mask token m, giving $|V| = 21$ for proteins. A sequence of length L is $x = (x_1, \ldots, x_L) \in V^L$.

Discrete Denoising Diffusion Probabilistic Models (D3PM[8]) forward chain

Position-wise Markov chain over V via row-stochastic transition matrices Q_t:

$$q\left(x_t^{(i)} = j \,\middle|\, x_{t-1}^{(i)} = k\right) = [Q_t]_{kj}, \qquad [Q_t]_{kj} \geq 0, \;\; \sum_j [Q_t]_{kj} = 1.$$

Two corruption types dominate:

- **Uniform**: each token replaced by a uniformly random token with prob. β_t.
- **Absorbing (masked)**: each token replaced by the mask token m; stays masked.

[8] Austin et al., (2021)

Uniform vs absorbing: which to use

Different denoising tasks, different trade-offs.

Uniform corruption

- Network must detect corruption *and* predict the original.
- Even far in the chain, each position retains some info.
- Reverse posterior is more complex.

Absorbing (masked)

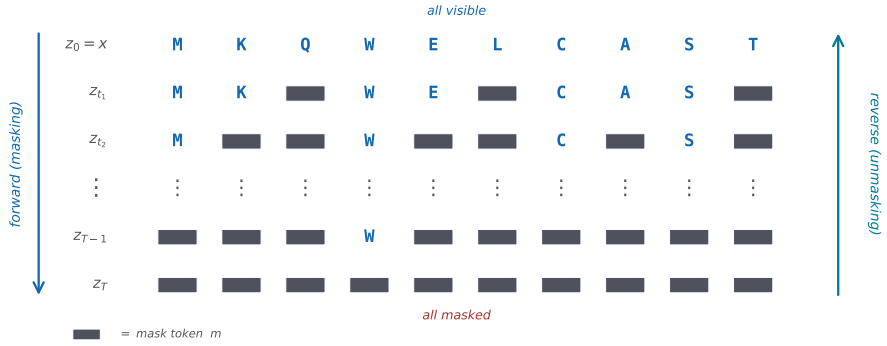
- Corrupted positions are explicit (the mask token).
- Each position is either fully known or fully unknown.
- Reverse posterior is simpler: stay masked vs. unmask to x_0.

Rest of this lecture: absorbing diffusion.

Absorbing forward process

Let $\beta_t \in [0,1]$ be the per-step mask probability. Position-wise:

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



Absorbing: once a position is masked, it stays masked.

Closed-form marginal: the workhorse equation

Define the cumulative mask probability $\bar{\beta}_t := 1 - \prod_{s=1}^{t}(1-\beta_s)$. Then for each position,

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

- Each position independently: kept with prob. $1-\bar{\beta}_t$, masked with prob. $\bar{\beta}_t$.
- Common choice: set β_t so that $\bar{\beta}_t$ linear from 0 at $t=0$ to 1 at $t=T$.
- Jump to any noise level in one step: sample one Bernoulli per position.
- Compare with continuous: closed-form *Bernoulli mixture* vs. closed-form *Gaussian*, both indexed by a single cumulative quantity ($\bar{\beta}_t$ vs. $\bar{\alpha}_t$).

Reverse process: unmasking and the closed-form posterior

Strong structural constraint from the forward chain:

- If position i is unmasked at time t, it must keep the same value at $t-1$.
- If position i is masked at time t, it either stays masked or unmasks to some token.

Conditional reverse posterior on a masked position

Conditioning on x_0 makes it tractable (parallels the continuous case):

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

A Bernoulli over {stay masked, unmask to x_0}.

Training trick (same as continuous): replace x_0 by the network's prediction $\hat{x}_\theta(\cdot \mid x_t, t)_i$, a distribution over V.

Training loss and the BERT connection

After expanding the variational lower bound (Sahoo et al. 2024, Shi et al. 2024):

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

Time-weighted cross-entropy on *masked positions only*.

Connection to BERT-style masked language modelling

Same structure: mask tokens, predict them, sum cross-entropy. Differences:

- Mask ratio: BERT fixes e.g. 15%; masked diffusion samples it per example, from 0% to 100%.
- Time weight: BERT has none; diffusion uses $\beta_t/\bar{\beta}_t$ from the ELBO.

Discrete diffusion is a principled generalization of BERT to a continuum of mask ratios. Models can be *warm-started* from BERT weights.

Sampling: algorithm and order variants

Ancestral sampling

1. Initialize $x_T = (m, m, \ldots, m)$.
2. For $t = T, T-1, \ldots, 1$:
   - Compute $\hat{x}_\theta(\cdot \mid x_t, t)_i$ at every position (one forward pass).
   - Sample x_{t-1} from the reverse posterior; copy unmasked positions through.
3. Return x_0.

Sampling order: parallel vs. one-position-per-step

- **Parallel** (default): many positions can be unmasked per step.
- **One-per-step** ($T = L$): exactly one masked position is unmasked at a time, in some order (random, confidence-based, entropy-based).
- Random order $\Rightarrow$ order-agnostic autoregressive (OADM, Hoogeboom et al. 2022). **EvoDiff** uses this.
- Same network, both regimes: choice is a sampling-time hyperparameter.

Continuous vs. discrete diffusion

	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	$z_t = \sqrt{\bar{\alpha}_t}\, x + \sqrt{1-\bar{\alpha}_t}\, \epsilon$	each position kept w.p. $1 - \bar{\beta}_t$
Reverse step	Gaussian posterior in $\mathbb{R}^d$	categorical posterior on V
Network output	clean $\hat{x}_\theta$ or noise $\hat{\epsilon}_\theta$	categorical $\hat{x}_\theta(\cdot \mid x_t, t)_i$
Training loss	MSE on clean data or noise	weighted cross-entropy on masked positions
$\nabla_x \log p$	defined; learnable as a score	undefined (discrete state)
Bio-seq examples	RFdiffusion, AlphaFold3, OmegAMP	EvoDiff, MDLM, SEDD

The genuine asymmetry is the $\nabla_x \log p$ row. Consequence: guidance methods that rely on classifier gradients do not transfer directly.

Guidance for discrete diffusion: what changes

All four methods extend to absorbing diffusion, with different effort.

Summary of changes

- **Plain conditioning:** trivial port, identical to continuous.
- **Classifier guidance:** requires workarounds, since $\nabla_z \log p$ is undefined on a discrete state space.
- **Classifier-free guidance:** same training trick; sampling formula now mixes *logits*.
- **Sequential Monte Carlo:** structurally identical, and gains a useful new tool: constrained proposals.

Practical takeaway: in discrete settings, CFG and SMC are the workhorses; classifier guidance is rarely the cleanest choice.

Plain conditioning: a trivial port

Add the conditioning vector c to the denoiser input, as in the continuous case:

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

The network now outputs a *conditional* categorical distribution at every position.

Everything else is unchanged

- Loss: time-weighted cross-entropy on masked positions, conditioned on c.
- Sampling: identical ancestral algorithm, with desired c fed in at every step.
- Training data: (x, c) pairs.

Architectural choices (concatenation, cross-attention, FiLM) carry over identically.

Classifier guidance: the gradient problem and workarounds

The obstacle

Classifier guidance needs $\nabla_{z_t} \log p_\phi(y \mid z_t, t)$. On a discrete state space, this gradient is undefined: z_t takes values in a finite set, not in $\mathbb{R}^d$.

Three families of workarounds

- **Hidden-state guidance** (NOS, Nisonoff et al. 2024): backprop the classifier into the network's *continuous* hidden activations $h(z_t)$ rather than into z_t itself.
- **Reconstruction-style**: evaluate a clean-data classifier on the predicted distribution $\hat{x}_\theta(\cdot \mid z_t, t)$ and backprop through it (DPS analog).
- **Logit-level modification**: shift the network's output logits by a learned or heuristic term derived from the classifier.

All are approximations with no clean analog to continuous classifier guidance. CFG and SMC are usually preferred for discrete data.

CFG for discrete diffusion: training and logit-space sampling

Training: identical to continuous case. During training, with probability p_{drop}, replace c by the null token $\varnothing$ before passing to the network. After training, the same network operates in conditional and unconditional modes.

Sampling: the network outputs *logits* $\ell_\theta(z_t, t, \cdot)_i \in \mathbb{R}^{|V|}$ at every position. Mix them at the logit level:

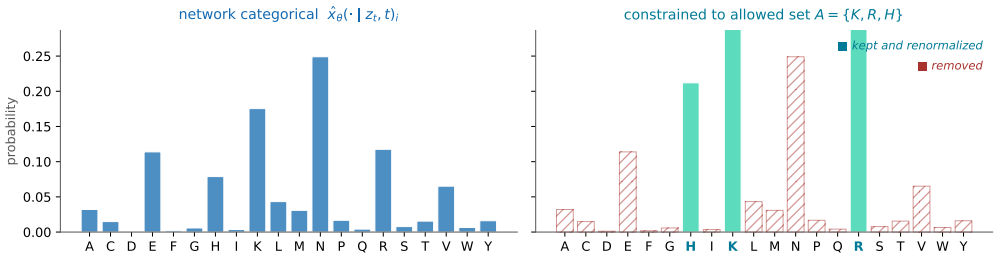
$$\tilde{\ell}_\theta(z_t, t, c)_i \;=\; (1+w)\,\ell_\theta(z_t, t, c)_i \;-\; w\,\ell_\theta(z_t, t, \varnothing)_i, \quad \tilde{p}_i \;=\; \mathrm{softmax}\big(\tilde{\ell}_\theta(z_t, t, c)_i\big).$$

Connection to continuous CFG

Equivalent to sampling from $\tilde{p}_i \propto p_\theta(\cdot \mid z_t, t, c)^{1+w} \cdot p_\theta(\cdot \mid z_t, t, \varnothing)^{-w}$. Same extrapolation structure as the continuous case, now in logit space.
Cost: $2\times$ forward passes per step. Workhorse for masked-diffusion text and protein generation.

SMC for discrete diffusion: constrained proposals

The SMC algorithm (propagate, reweight, resample) is structurally identical. Discrete data unlocks a useful extra tool: constrained proposals.



Why this is natural for discrete

At each masked position the network outputs a full categorical over V. Restrict it to an allowed set $A \subset V$, sample from the renormalized restricted distribution, correct with an importance weight. Works without retraining.

Choosing a method in practice

Same trade-offs as the continuous case, with a few discrete-specific notes.

Decision criteria

- Quick baseline with paired data? $\rightarrow$ **plain conditioning**.
- Differentiable property predictor at hand? $\rightarrow$ **classifier guidance** *workaround* (NOS / DPS-style).
- Want clean control with a single knob, no extra model? $\rightarrow$ **classifier-free guidance**.
- Reward is non-differentiable (oracle, simulator), or you have *hard constraints*? $\rightarrow$ **SMC** (with constrained proposals).

In practice

Production pipelines often combine these: a CFG-trained model as conditional base, with SMC for non-differentiable scorers and hard constraints at inference.

Key references

Continuous diffusion (foundations)

- Sohl-Dickstein et al. (2015), *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*.
- Ho et al. (2020), *Denoising Diffusion Probabilistic Models* (DDPM).
- Nichol & Dhariwal (2021), *Improved Denoising Diffusion Probabilistic Models* (cosine schedule).

Faster samplers

- Song et al. (2021), *Denoising Diffusion Implicit Models* (DDIM).
- Lu et al. (2022), *DPM-Solver*.

Guidance

- Dhariwal & Nichol (2021), *Diffusion Models Beat GANs* (classifier guidance, scale γ).
- Ho & Salimans (2022), *Classifier-Free Diffusion Guidance*.
- Chung et al. (2023), *Diffusion Posterior Sampling* (DPS).
- Wu et al. (2023), *Twisted Diffusion Sampler* (SMC steering).

Key references

Discrete diffusion

- Austin et al. (2021), *Structured Denoising Diffusion in Discrete State-Spaces* (D3PM).
- Sahoo et al. (2024), *Masked Diffusion Language Models* (MDLM).
- Shi et al. (2024), *Simplified and Generalized Masked Diffusion*.
- Hoogeboom et al. (2022), *Autoregressive Diffusion Models* (OADM).
- Gruver et al. (2023), NOS; Nisonoff et al. (2024), discrete classifier guidance.
- Alamdari et al. (2023), *EvoDiff*.