DREAM-DNA: CONTROLLED DESIGN VIA REASONING AND MATCHED-FLOWS FOR DNA

Weimin Wu[†,1*] **Jiunhau Chen**[‡*] **Xuefeng Song**[†] **Jerry Yao-Chieh Hu**[†] **Han Liu**[†,§,2]
[†] Center for Foundation Models and Generative AI, Northwestern University, USA
Department of Computer Science, Northwestern University, USA
[‡] Department of Physics, National Taiwan University, Taiwan
[§] Department of Statistics and Data Science, Northwestern University, USA
[1] wwm@u.northwestern.edu
[2] hanliu@northwestern.edu

ABSTRACT

We introduce DREAM-DNA (controlled **D**esign via **RE**asoning **A**nd **M**atched-flows for **DNA**), a reinforcement learning–based generative framework for DNA sequence design. Traditional models in this domain use diffusion architecture and rely on stochastic, single-pass generation. These limit their ability to correct early structural errors. DREAM-DNA overcomes these limitations by replacing stochasticity with Discrete Flow Matching. This establishes a deterministic "straight-line" mapping from noise to data for superior trajectory control. Our framework further introduces an Iterative Rollout mechanism. Specifically, we treat generation as a multi-round refinement process. By iteratively masking and resampling subsets of nucleotides in the DNA sequence, the model "critiques" and revises its intermediate outputs based on the reward feedback. Experiments on human enhancer design show that DREAM-DNA outperforms state-of-the-art baselines. It achieves a 13% boost in enhancer activity and an 8% increase in open chromatin match levels.

1 INTRODUCTION

We introduce DREAM-DNA (controlled **D**esign via **RE**asoning **A**nd **M**atched-flows for **DNA**), a reinforcement learning–based generative framework for DNA sequence design. Synthetic DNA design has emerged as the cornerstone of modern genomics. To navigate the vast sequence space, recent advances have leveraged diffusion models (Chen et al., 2025b; Wang et al., 2024; Avdeyev et al., 2023; Morehead et al., 2023). While powerful, these models suffer from a fundamental "one-shot" limitation: they generate sequences through a fixed, stochastic denoising trajectory. This leads to two critical failures: loss of control and inability to revise. To overcome these barriers, we propose a reinforcement learning–based generative framework, DREAM-DNA.

Our DREAM-DNA moves beyond passive "one-shot" generation toward deliberative design. Our approach introduces two primary shifts in the generative paradigm. First, we replace the stochasticity of diffusion with Discrete Flow Matching (Lipman et al., 2024; Stark et al., 2024; Lipman et al., 2022). It provides clearer control over the generation path and strengthens the link between the model's intermediate decisions and the final biological objective.

Second, inspired by the "reasoning" capabilities of large language models, we introduce an Iterative Rollout mechanism. Reinforcement learning (RL) has long served as a framework to align models with complex objectives through reward signals (Grattafiori et al., 2024; Wiering & Van Otterlo, 2012). Recent breakthroughs in RL have introduced iterative rollouts, i.e., repeated cycles of revision and evaluation, to allow models to improve their own outputs (Guo et al., 2025; Jaech et al., 2024). While methods like $d1$ (Zhao et al., 2025) have begun exploring reasoning in diffusion-based LLMs, the application of such "thinking" loops to biological sequence design remains untapped. In our framework, the model does not finish a sequence in one pass. Instead, it treats generation as a

[*] Equal contribution.

multi-round refinement process. Through iterative masking and resampling, the model can "critique" its intermediate sequences and refine global regulatory structures based on real-time reward feedback.

To this end, we build our framework on the direct reward backpropagation with the Gumbel–Softmax trick (DRAKES) algorithm (Wang et al., 2024) for synthetic DNA design. DRAKES provides a method to enable direct reward backpropagation through the entire single-pass trajectories generated by diffusion models. By adopting this approach, we make our flow-matching and iterative rollout mechanisms actionable. Our contributions are threefold:

- **Discrete flow-matching for DNA sequences:** We adapt the flow-matching to the discrete nature of DNA. By establishing a "straight-line" mapping from noise to data, this backbone offers superior trajectory control and a more stable link to biological task objectives.
- **"Reasoning-via-refinement" RL framework:** We introduce a reinforcement learning loop that utilizes iterative masking-based rollouts to treat DNA design as a deliberative process. This allows the model to "think" across multiple refinement cycles (a progressive remasking schedule, e.g., $100\% \rightarrow 80\% \rightarrow 50\%$) to identify and correct early sequence errors and optimize motifs for maximum biological rewards.
- **State-of-the-art regulatory design:** Through experiments on human HepG2 enhancer design, we demonstrate that combining deterministic flow with iterative revision outperforms standard single-pass diffusion baselines like DRAKES. Our framework achieves a 13% boost in predicted enhancer activity and an 8% increase in chromatin accessibility match levels while preserving sequence naturalness.

2 Related Works

In this section, we provide the related work on how generative models create DNA sequences, how reinforcement learning aligns these models with biological goals, and how reasoning strategies help models refine their results.

2.1 Generative Models for DNA Generation

DNA generative modeling (Wu et al., 2025b;a; Zhou et al., 2024) has shifted from unconstrained sampling to methods that satisfy functional constraints. Researchers have adapted discrete diffusion models to genomic data. For example, Kenneweg et al. (2025) generate human genotypes and assess realism with downstream disease classifiers. Other work uses reward- or score-weighted training to bias diffusion models toward sequences with higher predicted utility (Huang et al., 2024). Flow-matching-based methods provide a complementary route to controllable generation. Dirichlet flow matching uses deterministic sampling for discrete variables and reduces sampling cost compared with stochastic diffusion (Chen et al., 2025a; Stark et al., 2024). Previous flow-matching work often emphasizes the inference speed (Stark et al., 2024). In contrast, we focus on how deterministic flow trajectories can enable reward-driven edits to avoid the noise from stochastic sampling.

2.2 Reinforcement Learning for Diffusion Model

Reinforcement learning (RL) aligns a generator with user-defined objectives by maximizing expected reward rather than likelihood alone. In diffusion alignment, researchers cast the reverse process as a sequential decision problem. Each denoising step acts as a policy action, and learning maximizes the terminal reward of the final sample (Uehara et al., 2024). Researchers have explored variants of this idea for images (Miao et al., 2024) and planning (He et al., 2023). The same formulation supports DNA design with rewards based on motif presence, enhancer activity, or other sequence-level predictors. Existing approaches steer diffusion models toward higher-reward generations (Uehara et al., 2025; Chen et al., 2025b; Wang et al., 2024). However, these methods rely on single-pass updates during sampling. Once a diffusion model commits to a nucleotide early in the denoising trajectory, that choice becomes irreversible. There is no mechanism to re-examine the generated sequence, identify structural flaws, or backtrack. Our method extends single-pass update approaches such as DRAKES (Wang et al., 2024) by introducing an explicit rollout stage. The model samples trajectories, receives feedback on downstream outcomes, and iteratively revises its behavior. This additional interaction exposes sampling-time failures and enables multi-step refinement.

2.3 Reasoning Models

Reasoning-focused models improve output quality by verifying and revising intermediate results. Recent models such as DeepSeek-R1 (Guo et al., 2025) and GPT-o1 (Jaech et al., 2024) popularize deliberation-style inference, while training methods such as d1 optimize reasoning behavior via group relative policy optimization (Zhao et al., 2025). In genomics, Fallahpour et al. (2025) demonstrate that reasoning-style procedures improve DNA prediction tasks. However, such work in genomic reasoning targets "understanding" settings based on language models. The model reasons over a fixed input sequence to produce a single label or score. Such formulations fail to address the core challenge of generative genomic design. In the DNA sequence design domain, early sequence decisions constrain downstream functional outcomes, and errors cannot be corrected post hoc. In contrast, our method applies iterative refinement to sequence generation. We treat nucleotides as the editable object of reasoning and introduce rollout-based feedback that allows the model to inspect, revise, and improve generated sequences. This enables trajectory-level reasoning over regulatory structure, and supports multi-step corrections toward desired motif composition and regulatory activity.

3 Preliminary

In this section, we present the mathematical formulation of the DNA sequence design problem and review a state-of-the-art method for this task, DRAKES (Wang et al., 2024). DRAKES provides a principled framework for applying reward optimization to the diffusion model by enabling direct reward backpropagation. We implement our method based on the DRAKES framework.

3.1 Problem Setting

We study reward-guided DNA sequence design. Let $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ denote the nucleotide alphabet and let $\mathcal{X} = \Sigma^N$ denote the space of length-N sequences. We write a sequence as $x = (x^1, \ldots, x^N) \in \mathcal{X}$. When a design constraint (e.g., enhancer activity), we denote it by c and write the generator as $p(x \mid c)$.

Pre-trained generator. Following DRAKES (Wang et al., 2024), we start from a pre-trained generative model (e.g., diffusion or flow-matching model) that captures the distribution of natural DNA sequences. Then we parameterize the generative process with a continuous-time Markov chain (CTMC) on $\mathcal{X}$ with time $t \in [0, T]$ and generator matrix $Q^\theta(t) \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{X}|}$. The parameter vector $\theta \in \Theta$ collects the learnable weights that define the transition rates (e.g., a neural network that outputs $Q^\theta(t)$ from the current state and time). Let p_t^θ denote the probability mass function over $\mathcal{X}$ at time t. The CTMC evolves according to:

$$\frac{dp_t^\theta}{dt} = Q^\theta(t)\, p_t^\theta,\ p_0^\theta = p_{\mathrm{lim}}.$$

We choose an initial distribution p_{lim} that represents a maximally corrupted ("noise") state (e.g., a fully masked state or an uninformative prior), and the terminal distribution p_T^θ defines the model's samples. We denote the pre-trained parameters by θ_{pre} and the corresponding model by $p^{\mathrm{pre}}(\cdot \mid c)$.

Finite-step discretization. For a small step Δt, the CTMC induces the transition rule:

$$\Pr(x_{t+\Delta t} = y \mid x_t = x) = \mathbb{I}(x = y) + Q^\theta_{x,y}(t)\, \Delta t, \tag{1}$$

where $Q^\theta_{x,y}(t)$ denotes the (x, y) entry of $Q^\theta(t)$. We use this to sample a trajectory $x_{0:T} = (x_0, x_{\Delta t}, \ldots, x_T)$.

Reward and objective. We define a downstream reward function $r : \mathcal{X} \to \mathbb{R}$ that scores a designed sequence (e.g., predicted enhancer activity or motif-based criteria). A naive objective maximizes expected reward under the model:

$$\max_{\theta \in \Theta}\ \mathbb{E}_{x \sim p_T^\theta(\cdot \mid c)}\big[r(x)\big],$$

but this objective can over-optimize the reward and produce unnatural sequences. DRAKES addresses this issue by regularizing toward the pre-trained model at the trajectory level. Concretely, we use

Algorithm 1 DRAKES: Direct Reward Backpropagation with the Gumbel–Softmax Trick

Require: pre-trained generator $Q^{\text{pre}}(t)$, reward $r : \mathcal{X} \to \mathbb{R}$, learning rate β, batch size B, iterations S, step size $\Delta t = T/K$, temperature τ, regularization weight α.

1: **for** $s \in \{1, \ldots, S\}$ **do**
2: **for** $i \in \{1, \ldots, B\}$ **do**
3: **for** $t \in \{0, \Delta t, \ldots, T\}$ **do**
4: Set $\pi(t) \in \Delta(\mathcal{X})$ where

$$\pi(t)_y = \begin{cases} [\bar{x}_{t-1}^{(i)}]_y + \Delta t \sum_{x \in \mathcal{X}} \bar{x}_{t-1,x}^{(i)} Q_{y,x}^{\theta_s}(t), & (t > 0); \\ p_{\lim}(y), & (t = 0). \end{cases}$$

5: Sample i.i.d. Gumbel noise $\{G_y\}_{y \in \mathcal{X}}$, where $G_y \sim \text{Gumbel}(0, 1)$.
6: Set a differentiable counterpart:

$$\bar{x}_t^{(i)} \leftarrow \Big[\text{softmax}((\pi(t)_y + G_y)/\tau)\Big]_{y \in \mathcal{X}}.$$

7: **end for**
8: **end for**
9: Set the loss:

$$g(\theta_s) = \frac{1}{B} \sum_{i=1}^{B} \Bigg(r\left(\bar{x}_T^{(i)}\right) - \frac{\alpha}{T} \sum_{t=1}^{T} \sum_{x \in \mathcal{X}} [\bar{x}_{t-1}^{(i)}]_x \sum_{\substack{y \in \mathcal{X} \\ y \neq x}} \Big\{ -Q_{x,y}^{\theta_s}(t) + Q_{x,y}^{\text{pre}}(t) + Q_{x,y}^{\theta_s}(t) \log \frac{Q_{x,y}^{\theta_s}(t)}{Q_{x,y}^{\text{pre}}(t)} \Big\} \Bigg)$$

10: Update parameters: $\theta_{s+1} \leftarrow \theta_s + \beta \, \nabla g(\theta)\big|_{\theta = \theta_s}$.
11: **end for**
12: **Output:** θ_{S+1}

$Q_{x_t,\cdot}^{\theta}(t)$ to denote the row of the rate matrix $Q^{\theta}(t)$ indexed by the current state x_t (i.e., transition rates from x_t to all states in $\mathcal{X}$). Then we optimize:

$$\max_{\theta \in \Theta} \mathbb{E}_{x_{0:T} \sim P^{\theta}(\cdot|c)} \Big[r(x_T) - \alpha \mathcal{L}_{KL} \Big], \tag{2}$$

where $\mathcal{L}_{KL} = \int_0^T \sum_{y \neq x_t} \text{KL}\big(Q_{x_t,\cdot}^{\theta}(t) \parallel Q_{x_t,\cdot}^{\text{pre}}(t)\big) \, dt$, $P^{\theta}(\cdot \mid c)$ denotes the path distribution induced by $Q^{\theta}(t)$ and $\alpha > 0$ is a weight for naturalness regularizer.

3.2 Existing SOTA Method: DRAKES

Wang et al. (2024) propose direct reward backpropagation with the Gumbel–Softmax trick (DRAKES) to fine-tune a masked discrete diffusion model toward a downstream reward while preserving sample naturalness. DRAKES uses the same CTMC parameterization as in section 3.1: a learnable generator $Q^{\theta}(t)$ induces a path distribution $P^{\theta}(x_{0:T} \mid c)$ over trajectories $x_{0:T}$. We provide the detailed algorithm in algorithm 1. It includes two main parts.

Data collection (step 1–8). DRAKES generates samples by simulating the CTMC with step size Δt. For a current state $x_t \in \mathcal{X}$ and any candidate next state $y \in \mathcal{X}$, the discretization in equation 1 gives:

$$\Pr(x_{t+\Delta t} = y \mid x_t = x_t) = \mathbb{I}(x_t = y) + Q_{x_t,y}^{\theta}(t) \, \Delta t.$$

DRAKES forms a categorical distribution over $y \in \mathcal{X}$,

$$\pi_t(y) \propto \mathbb{I}(x_t = y) + Q_{x_t,y}^{\theta}(t) \, \Delta t,$$

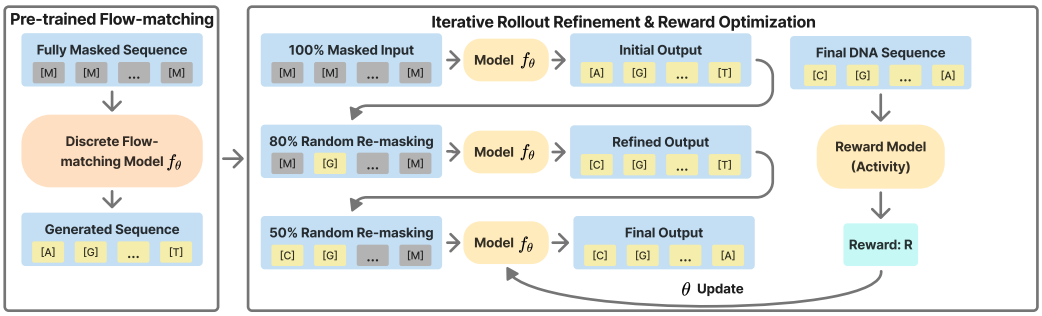

Figure 1: **Overview of DREAM-DNA.** This generative pipeline integrates a pre-trained Discrete Flow Matching backbone with an Iterative Rollout Loop that treats sequence generation as a multi-round reasoning process. In each cycle, the model critiques its own output through progressive random remasking and resampling to refine biological motifs. The final sequence is evaluated by Reward Oracle (enhancer activity) and regularized via KL Divergence against a frozen reference model to ensure naturalness. Using the DRAKES algorithm, i.e., a differentiable Gumbel-Softmax estimator, gradients are backpropagated through all rollout cycles to update model parameters θ end-to-end.

and samples $x_{t+\Delta t} \sim \mathrm{Cat}(\pi_t)$, where $\mathrm{Cat}(\pi_t)$ means a random variable on $\mathcal{X}$ with probability mass function:

$$\Pr(Y = y) = \pi_t(y), \sum_{y \in \mathcal{X}} \pi_t(y) = 1.$$

To enable backpropagation through sampling, DRAKES replaces each discrete state with a differentiable relaxation. It encodes a discrete state $x_t \in \mathcal{X}$ as a one-hot vector $e_{x_t} \in \mathbb{R}^{|\mathcal{X}|}$, where $(e_{x_t})_y = \mathbb{I}(y = x_t)$. This one-hot vector lies in the probability simplex:

$$\Delta(\mathcal{X}) = \left\{ v \in \mathbb{R}^{|\mathcal{X}|} : v_y \geq 0 \, \forall y \in \mathcal{X}, \sum_{y \in \mathcal{X}} v_y = 1 \right\}.$$

DRAKES then uses the Gumbel–Softmax reparameterization with temperature τ to produce a soft sample $\bar{x}_t \in \Delta(\mathcal{X})$ that approximates e_{x_t} and remains differentiable.

Optimization (step 9–10). DRAKES optimizes the regularized reward objective in equation 2. It estimates gradients by differentiating through the relaxed trajectory $\bar{x}_{0:T}$ and then updates θ with stochastic gradient ascent. To reduce memory cost, DRAKES truncates backpropagation and only differentiates through a suffix of the trajectory.

4 Methodology

In this section, we describe our framework for controllable DNA sequence design. We first introduce a discrete flow-matching formulation tailored to masked-to-DNA generation (Gat et al., 2024). We then present a rollout strategy that revises sequences during fine-tuning.

4.1 Discrete Flow Matching for DNA Design

Setup. Let $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ and add a special mask token [MASK], so the vocabulary size is $d = |\Sigma| + 1 = 5$. We consider length-N sequences $x = (x^1, \ldots, x^N) \in \mathcal{D} \triangleq \Sigma_{\mathrm{mask}}^N$, where $\Sigma_{\mathrm{mask}} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}, [\mathrm{MASK}]\}$.

Flow matching learns a probability path $\{p_t\}_{t \in [0,1]}$ that connects a source distribution p_0 to a target distribution p_1. We write the marginal path as:

$$p_t(x) = \sum_{x_0, x_1 \in \mathcal{D}} p_t(x \mid x_0, x_1) \, \pi(x_0, x_1),$$

where $\pi(x_0, x_1)$ couples source samples $x_0 \sim p_0$ and target samples $x_1 \sim p_1$.

Masked-to-data conditional path. For DNA, we set the source $x_0 = M$ to the fully masked sequence, i.e., $M^i = [\text{MASK}]$ for all $i \in \{1, \ldots, N\}$, and we set x_1 to a natural DNA sequence from the dataset. We define a simple conditional path using a scalar schedule $\kappa_t \in [0, 1]$ with $\kappa_0 = 0$ and $\kappa_1 = 1$:

$$p_t(x^i \mid x_0, x_1) = (1 - \kappa_t)\, \mathbb{I}(x^i = x_0^i) + \kappa_t\, \mathbb{I}(x^i = x_1^i).$$

Sampling with Euler steps. We simulate the continuous-time process with step size Δt (Campbell et al., 2024). At time t, the sampler draws the next state from a categorical distribution over $\mathcal{D}$,

$$x_{t+\Delta t} \sim \text{Cat}(\pi_t),\ \Pr(x_{t+\Delta t} = y) = \pi_t(y),\ \sum_{y \in \mathcal{D}} \pi_t(y) = 1,$$

where π_t comes from a CTMC rate matrix R_t. In a CTMC, $R_t(x, y) \geq 0$ for $y \neq x$ gives the instantaneous transition rate from x to y, and $R_t(x, x) = -\sum_{y \neq x} R_t(x, y)$ ensures probability conservation.

For the masked-to-DNA path, the idealized masked-rate form (used as intuition) places all mass on copying the target when the current state stays fully masked:

$$R_t^{*\text{mask}}(x_t, x_{t+\Delta t} \mid x_1) = \frac{\mathbb{I}(x_t = M)\, \mathbb{I}(x_{t+\Delta t} = x_1)}{1 - t}.$$

Rate-model parameterization. We parameterize the time-dependent rate matrix R_t with a neural network that outputs token-wise logits over Σ_{mask} given the current sequence x_t and time t. We then convert these logits into transition rates by assigning each position i a distribution over candidate tokens and allowing only single-site edits per infinitesimal step.

Concretely, let $f_\theta(x_t, t)^i \in \mathbb{R}^d$ denote the logits at position i, and define:

$$q_{\theta,t}^i(a \mid x_t) = \text{softmax}(f_\theta(x_t, t)^i)_a,$$

for $a \in \Sigma_{\text{mask}}$. For any two sequences $x, y \in \mathcal{D}$ that differ at exactly one position i, we set an off-diagonal rate:

$$R_t^\theta(x, y) = \lambda(t)\, q_{\theta,t}^i(y^i \mid x),$$

and we set $R_t^\theta(x, y) = 0$ when x and y differ at more than one position. The scalar schedule $\lambda(t)$ controls the overall jump intensity, and the diagonal is:

$$R_t^\theta(x, x) = -\sum_{y \neq x} R_t^\theta(x, y).$$

This construction yields efficient sampling (single-site jumps) while still letting the network coordinate global structure through context.

Comparison to discrete diffusion. Discrete diffusion models typically define a stochastic reverse process that injects noise at each step; the sampling trajectory therefore resembles a random walk guided by a learned score. In contrast, flow matching uses a deterministic probability path p_t and implements sampling as controlled CTMC jumps. This more structured trajectory makes test-time steering easier: reward gradients and KL regularization push the model along similar paths rather than competing with stochasticity.

4.2 Flow-matching Model Pre-training

We pre-train a discrete flow-matching backbone parameterized by a neural network f_θ (a multi-layer 1D convolutional network). The model learns to predict clean nucleotides at masked positions, and this defines the conditional flow used at sampling time.

Forward noising (masking) process. Given a clean DNA sequence $x_0 \in \mathcal{D}$, we sample a discrete time index $t \in \{0, \ldots, T-1\}$ and corrupt x_0 by masking each position independently with probability $\sigma_t \in [0, 1]$:

$$\Pr(x_t^i = [\text{MASK}] \mid x_0^i) = \sigma_t,\ \Pr(x_t^i = x_0^i \mid x_0^i) = 1 - \sigma_t.$$

Algorithm 2 Iterative Rollout Mechanism

Require: mask ratios $\rho \in \{1.0, 0.8, 0.5\}$, learning rate β, batch size B, iterations S.

1: **for** $s \in \{1, \ldots, S\}$ **do**
2: **for** $i \in \{1, \ldots, B\}$ **do**
3: Initialize $x^{(i,1)} \leftarrow M$ (fully masked input).
4: **for** $j \in \{1, 2, 3\}$ **do**
5: Run Steps 3–9 of algorithm 1 using input $x^{(i,j)}$ to obtain a (soft) sample $\bar{x}^{(i,j)}$ and loss $g^{(i,j)}$.
6: Set $x^{(i,j+1)} \leftarrow \mathrm{Remask}(\bar{x}^{(i,j)}, \rho_{j+1})$.
7: **end for**
8: Accumulate rollout loss: $g^{(i)} \leftarrow \frac{1}{3} \sum_{j=1}^{3} g^{(i,j)}$.
9: **end for**
10: Update parameters:

$$\theta_{s+1} \leftarrow \theta_s + \beta \, \nabla_\theta \Big(\frac{1}{B} \sum_{i=1}^{B} g^{(i)} \Big) \Big|_{\theta=\theta_s}.$$

11: **end for**
12: **Output:** θ_{S+1}

We define the linear schedule by:

$$\sigma_t^{\text{linear}} = \frac{t+1}{T},$$

and we sharpen it with a square schedule:

$$\sigma_t = 1 - \big(1 - \sigma_t^{\text{linear}}\big)^2.$$

Why a square schedule. The square schedule increases the masking probability more slowly at early timesteps and more aggressively near the end. This shape encourages the model to see partial masked sequences for a larger fraction of training, which helps it learn long-range structure before solving the hardest fully-masked reconstruction cases.

Training objective (masked score-entropy). Let $\mathcal{M}(x_t) = \{i \in \{1, \ldots, N\} : x_t^i = [\text{MASK}]\}$ denote the masked positions. The network outputs logits $f_\theta(x_t, t) \in \mathbb{R}^{N \times d}$, and we define the per-position predictive distribution:

$$p_\theta(x^i \mid x_t, t) = \mathrm{Softmax}(f_\theta(x_t, t)^i).$$

We train with a masked negative log-likelihood (cross-entropy) that only scores masked positions,

$$\mathcal{H}_\theta(x_0, x_t, t) = -\frac{1}{|\mathcal{M}(x_t)|} \sum_{i \in \mathcal{M}(x_t)} \log p_\theta(x_0^i \mid x_t, t).$$

We weight this loss by the noise schedule as in discrete flow matching,

$$\mathcal{L} = \mathbb{E}_{x_0, t} \big[w(t) \, \mathcal{H}_\theta(x_0, x_t, t) \big],$$

where $w(t)$ is a weighting function.

4.3 Reasoning-based Rollout

We fine-tune the generator with an iterative rollout loop that alternates between sampling and revision. Each rollout cycle masks part of the current sequence and asks the model to fill in the missing tokens again, which mimics a simple "revise-and-check" reasoning process, as detailed in algorithm 2.

Reward models. We train reward oracles on labeled enhancer data and use them to score generated sequences. We train a regression oracle for enhancer activity. To avoid bias, we separate the reward oracle used for optimization (r_{train}) from the oracle used for reporting (r_{eval}) by training them on disjoint dataset splits.

RL objective with a naturalness constraint. Directly maximizing $r_{\text{train}}(x)$ pushes the generator toward unrealistic sequences. We therefore optimize a regularized objective that balances reward and a KL penalty to a frozen reference model. Concretely, for each generated trajectory we maximize:

$$J(\theta) = \mathbb{E}\big[r_{\text{train}}(x_T)\big] - \alpha\, \mathcal{L}_{\text{KL}},$$

where $\mathcal{L}_{\text{KL}}$ matches the fine-tuned generator to the reference generator along the trajectory.

Reasoning-via-refinement. We run three cycles per batch: (i) generation from M; (ii) remask 80% of tokens and regenerate; (iii) remask 50% and regenerate. In cycle j, we treat the reverse-time steps as a differentiable policy and compute a rollout loss:

$$g^{(i,j)} \;=\; r_{\text{train}}\Big(\bar{x}_T^{(i,j)}\Big) - \alpha\, \mathcal{L}_{\text{KL}}\Big(\bar{x}_{0:T}^{(i,j)}\Big)\,,$$

where $\mathcal{L}_{\text{KL}}$ penalizes deviation from the frozen reference generator on the same masked conditioning states. We compute $g^{(i,j)}$ only on tokens that remain masked in the input of cycle j so the model learns to correct uncertain positions instead of overwriting stable context. We then average losses across cycles and backpropagate through all cycles to update θ.

Why this acts like "reasoning". Rollout goes beyond repeated resampling because each cycle conditions on a partially preserved draft and receives explicit feedback through r_{train} and the KL regularizer. The model therefore learns a revision operator: it keeps high-confidence parts fixed and edits low-confidence regions to increase reward.

For example, suppose Cycle 1 places a low-scoring local pattern that reduces the activity oracle. Cycle 2 remasks that region and the model can propose an alternative motif instantiation while keeping the rest of the sequence unchanged. Cycle 3 repeats this refinement on a smaller masked set, which encourages incremental improvement rather than wholesale rewriting.

5 EXPERIMENT

In this section, we present our experimental results. We provide an ablation study to show how flow matching and iterative rollout each contribute to superior DNA design. Due to the limit, we detail the baselines in section B, dataset in sections C and D, and the biological metrics used for evaluation in section E.

Table 1: **Comparison of DNA design methods across functional and naturalness metrics.** We compare our proposed flow-matching and iterative rollout mechanism with the baselines.

Method	**Pred-Activity** ↑	**ATAC-Acc** ↑ (%)	**3-mer Corr** ↑	**JASPAR Corr** ↑
SMC	4.58 (0.14)	48.0 (11.5)	0.847 (0.082)	0.723 (0.070)
CG	1.09 (0.01)	0.0 (0.0)	-0.093 (0.001)	-0.284 (0.019)
CFG	5.00 (0.09)	92.4 (0.6)	0.745 (0.002)	0.741 (0.011)
TDS	4.60 (0.36)	37.6 (6.3)	**0.855 (0.011)**	**0.753 (0.094)**
Pre-trained Diff	0.17 (0.01)	1.5 (0.3)	-0.054 (0.061)	-0.105 (0.079)
Pre-trained Flow	1.92 (1.93)	4.17 (7.22)	0.453 (0.276)	0.467 (0.181)
DRAKES	5.95 (0.26)	90.2 (4.6)	0.368 (0.087)	0.543 (0.141)
DREAM-DNA (w/o Roll)	6.59 (0.06)	92.9 (4.3)	0.427 (0.002)	0.280 (0.050)
DREAM-DNA	**6.74 (0.07)**	**98.0 (1.4)**	0.500 (0.001)	0.398 (0.056)

Flow matching vs. diffusion backbones. Table 1 shows that the flow-matching (FM) backbone achieves higher Pred-Activity and ATAC-Acc than the diffusion baselines (e.g., pre-trained flow-matching vs. pre-trained diffusion). The experimental results demonstrate that transitioning from stochastic diffusion models to a deterministic flow-matching framework enhances the design of functional DNA sequences. In human HepG2 enhancer design tasks, the pre-trained Flow-matching backbone served as a superior starting point compared to traditional diffusion, eventually achieving a 13% boost in predicted enhancer activity and an 8% increase in open chromatin match levels (ATAC-Acc) over the state-of-the-art DRAKES baseline. These improvements suggest that the deterministic "straight-line" mapping from noise to data provides more stable trajectory control, strengthening the link between the model's intermediate decisions and the final biological objective.

Table 2: **Ablation study on the impact of rollout times across different generative backbones.** Results show that three rollout cycles consistently yield the highest chromatin accessibility (ATAC-Acc) for both Diffusion and Flow-matching frameworks.

Backbone	Roll Steps	Pred-Activity	ATAC-Acc (%)	3-mer Corr	JASPAR Corr
DRAKES	w/o Roll	5.95 (0.26)	90.2 (4.6)	0.368 (0.087)	0.543 (0.141)
	Roll $\times$ 2	6.34 (0.03)	88.0 (2.1)	0.375 (0.091)	0.342 (0.112)
	Roll $\times$ 3	**6.67 (0.04)**	**98.35 (1.9)**	**0.457 (0.096)**	**0.489 (0.059)**
	Roll $\times$ 4	6.32 (0.05)	88.6 (2.3)	0.381 (0.079)	0.424 (0.091)
DREAM-DNA	w/o Roll	6.59 (0.06)	92.9 (4.3)	0.427 (0.002)	0.280 (0.050)
	Roll $\times$ 2	**6.76 (0.05)**	94.2 (1.9)	0.463 (0.005)	0.360 (0.087)
	Roll $\times$ 3	6.74 (0.07)	**98.0 (1.4)**	**0.500 (0.001)**	**0.398 (0.056)**
	Roll $\times$ 4	6.55 (0.03)	95.9 (0.8)	0.411 (0.005)	0.332 (0.010)

Effect of iterative rollout. Across both backbones, rollout improves functional metrics by allowing the model to revise previously generated tokens. In particular, DREAM-DNA achieves 98% ATAC-Acc and the best Pred-Activity in table 1. A critical driver of this performance is the Iterative Rollout mechanism, which allows the model to "critique" and revise its intermediate outputs. Unlike "one-shot" diffusion models that lock in nucleotide choices early in the process, DREAM-DNA treats generation as a multi-round refinement cycle. By remasking and resampling sequences, the model can identify and correct structural errors or unfavorable motifs based on real-time reward feedback. This deliberative process enables "reasoning-via-refinement", where the model keeps high-confidence parts fixed while editing low-confidence regions to maximize activity.

Sampling efficiency. Compared with diffusion sampling, flow-matching allows us to reduce the number of inference steps substantially. In our setup, this change reduces total generation time by about 90% without degrading the reported functional metrics.

Ablation study: effect of rollout depth. We ablate the number of rollout cycles (2, 3, 4), with the results shown in table 2. While any amount of rollout improves functional metrics, three rollout cycles consistently yield the highest performance for both the diffusion and flow-matching backbones. For instance, using three cycles achieved a peak 98.0% ATAC-Acc for DREAM-DNA, compared to 92.9% without rollout. Increasing the depth to four cycles resulted in a slight decline in both activity and accessibility scores, suggesting that three cycles provide the optimal trade-off.

6 Discussion and Conclusion

DREAM-DNA introduces a robust framework for controllable DNA design by integrating a deterministic Discrete Flow Matching backbone with a Reasoning-via-Refinement reinforcement learning loop. Our results demonstrate that this combination outperforms traditional single-pass diffusion models, achieving a 13% boost in enhancer activity and an 8% increase in chromatin accessibility match levels. By replacing stochastic "random walks" with deterministic "straight-line" trajectories, the model establishes a more stable link between intermediate generation steps and final biological objectives. Furthermore, the iterative rollout mechanism enables the model to act as a "revision operator", identifying and correcting structural flaws that would otherwise be locked in during a standard denoising process.

We have the following limitations: (i) Naturalness Trade-offs: Addressing the decline in JASPAR Correlation to ensure that optimizing for functional activity does not compromise the preservation of complex, natural genomic motifs. (ii) Rollout Sensitivity: Investigating why performance peaks at three cycles and developing adaptive remasking strategies to better identify "low-confidence" regions without destroying stable sequence context. (iii) Scalability: Expanding the 1D convolutional backbone and the 700,000-sequence Gosai dataset to improve model generalization across diverse, out-of-distribution biological conditions.

ACKNOWLEDGMENTS

JH is partially supported by Ensemble AI, and the Walter P. Murphy Fellowship and the Terminal Year Fellowship (Paul K. Richter Memorial Award) of Northwestern University. Han Liu is partially supported by NIH R01LM1372201, NSF AST-2421845, Simons Foundation MPS-AI-00010513, AbbVie, Dolby, and Chan Zuckerberg Biohub Chicago Spoke Award. This research was supported in part through the computational resources and staff contributions provided for the Quest high performance computing facility at Northwestern University which is jointly supported by the Office of the Provost, the Office for Research, and Northwestern University Information Technology. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

REFERENCES

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pp. 1276–1301. PMLR, 2023.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

Tong Chen, Yinuo Zhang, Sophia Tang, and Pranam Chatterjee. Multi-objective-guided discrete flow matching for controllable biological sequence design. *arXiv preprint arXiv:2505.07086*, 2025a.

Xingyu Chen, Shihao Ma, Runsheng Lin, Jiecong Lin, and Bo Wang. Ctrl-dna: Controllable cell-type-specific regulatory dna design via constrained rl. *arXiv preprint arXiv:2505.20578*, 2025b.

Adibvafa Fallahpour, Andrew Magnuson, Purav Gupta, Shihao Ma, Jack Naimer, Arnav Shah, Haonan Duan, Omar Ibrahim, Hani Goodarzi, Chris J Maddison, et al. Bioreason: Incentivizing multimodal biological reasoning within a dna-llm model. *arXiv preprint arXiv:2505.23579*, 2025.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36:64896–64917, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Kaixuan Huang, Yukang Yang, Kaidi Fu, Yanyi Chu, Le Cong, and Mengdi Wang. Latent diffusion models for controllable rna sequence generation. *arXiv preprint arXiv:2409.09828*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Philip Kenneweg, Raghuram Dandinasivara, Xiao Luo, Barbara Hammer, and Alexander Schönhuth. Generating synthetic genotypes using diffusion models. *Bioinformatics*, 41(Supplement_1):i484–i492, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Zichen Miao, Jiang Wang, Ze Wang, Zhengyuan Yang, Lijuan Wang, Qiang Qiu, and Zicheng Liu. Training diffusion models towards diverse image generation with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10844–10853, 2024.

Alex Morehead, Jeffrey Ruffolo, Aadyot Bhatnagar, and Ali Madani. Towards joint sequence-structure generation of nucleic acid and protein complexes with se (3)-discrete diffusion. *arXiv preprint arXiv:2401.06151*, 2023.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024.

Masatoshi Uehara, Xingyu Su, Yulai Zhao, Xiner Li, Aviv Regev, Shuiwang Ji, Sergey Levine, and Tommaso Biancalani. Reward-guided iterative refinement in diffusion models at test-time with applications to protein and dna design. *arXiv preprint arXiv:2502.14944*, 2025.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024.

Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012.

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

Weimin Wu, Xuefeng Song, Yibo Wen, Qinjie Lin, Zhihan Zhou, Jerry Yao-Chieh Hu, Zhong Wang, and Han Liu. Genome-factory: An integrated library for tuning, deploying, and interpreting genomic models. September 2025a.

Weimin Wu, Zhihan Zhou, Robert Riley, Mohammad Abdulqader, Xuefeng Song, Satria Kautsar, Rob Egan, Steven Hofmeyr, Gefei Liu, Shira Goldhaber-Gordon, Mutian Yu, Harrison Ho, Yaping Liu, Andrei Stecca Steindorff, Fengchen Liu, Feng Chen, Rachael Morgan-Kiss, Lizhen Shi, Han Liu, and Zhong Wang. Uncovering the genomic manifold via scalable learning from the global microbiome. *bioRxiv*, 2025b. doi: 10.1101/2025.01.30.635558.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025.

Zhihan Zhou, Weimin Wu, Harrison Ho, Jiayi Wang, Lizhen Shi, Ramana V. Davuluri, Zhong Wang, and Han Liu. Dnabert-s: Learning species-aware dna embedding with genome foundation models. February 2024.

A Implementation Details

We provide the detailed implementations for model training and inference here.

A.1 Optimization and Training Setup

We train the model using the AdamW optimizer with a fixed learning rate $\beta = 10^{-4}$. The batch size is set to 32 for pre-training and 64 for reinforcement learning fine-tuning, with the seed fixed at 1. All experiments are conducted on an NVIDIA RTX-6000 GPU. Pre-training the flow-matching model requires $12{,}500$ steps and takes approximately 14 minutes. The entire framework is implemented using PyTorch Lightning.

A.2 Sampling Procedure

During inference, generation is initialized from a fully masked sequence. Sampling proceeds over a reduced set of timesteps $\{t_1, \ldots, t_S\}$ where $S \ll T$. For each step, the model predicts token-wise logits conditioned on the current partially masked sequence, followed by categorical sampling with optional temperature scaling. Masked positions are progressively revealed with a probability proportional to $\Delta\sigma_t/\sigma_t$. Optional stochastic remasking may be applied to encourage exploration during the generation trajectory.

B Baselines

We compare our method with several baseline techniques for controlled generation. These methods represent the state-of-the-art for aligning generative models with specific goals.

- Guidance-based Methods (CG, SMC, TDS): We compare our work with techniques that use gradients or sampling to guide generation, including classifier guidance (CG) (Nisonoff et al., 2024), sequential Monte Carlo (SMC) method, and Twist Diffusion Sampling (TDS) (Wu et al., 2023).
- Classifier-free Guidance (CFG) (Ho & Salimans, 2022): This method trains the model to generate sequences by interpolating between conditional and unconditional outputs.
- Pre-trained Diffusion: This baseline uses a standard pre-trained discrete diffusion model (Wang et al., 2024) to generate sequences without any fine-tuning.
- DRAKES: This is our primary baseline. It uses reinforcement learning to optimize DNA sequences in a single pass. We aim to show that our framework improves upon this baseline by introducing flow matching and reasoning cycles.

C Pre-training and Reinforcement Learning Datasets

The training dataset is constructed from the Gosai dataset, which contains DNA sequences across three human cell lines: HepG2, K562, and SKNSH. Each sample consists of a nucleotide sequence and a corresponding vector of activity labels.

C.1 Sequence Representation.

Each DNA sequence is represented as a categorical discrete sequence where nucleotides are mapped to integer tokens: $\texttt{A} \to 0$, $\texttt{C} \to 1$, $\texttt{G} \to 2$, and $\texttt{T} \to 3$. A sequence of length L is encoded as a vector in $\{0, 1, 2, 3\}^L$, ensuring compatibility with both our discrete flow-matching and diffusion models.

C.2 Training Set Construction.

The training set utilizes the entire Gosai dataset without exclusions to maximize coverage of the underlying data distribution. Each training sample is returned as a tuple (x, y), where x is the tokenized sequence and $y \in \mathbb{R}^3$ represents the activity labels for the three cell lines. An attention mask of all ones is provided for architectural compatibility. We use a PyTorch `DataLoader` with random shuffling enabled. All hyperparameters are managed via Hydra configuration files.

D Evaluation Datasets

We define two evaluation subsets: a validation set and a test set. Each subset consists of $40{,}000$ DNA sequences randomly sampled without replacement from the Gosai dataset. Sampled indices are fixed within each experimental run to ensure consistent model selection and reporting. Because these evaluation subsets contain sequences observed during training, results reflect in-distribution performance rather than generalization to unseen biological conditions.

D.1 Evaluation Data Representation.

Evaluation samples share the same representation as training samples, using the DNA alphabet $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$ and paired activity labels. An attention mask of all ones indicates that all sequence positions are valid.

Data loading for evaluation. We use PyTorch `DataLoader`s to load the evaluation datasets. Random shuffling is disabled by default to ensure deterministic evaluation. Evaluation batch sizes are independently configurable and do not rely on stochastic data ordering.

E EVALUATION METRICS CALCULATION METHODS

We evaluate the generated DNA sequences using a set of complementary metrics designed to assess both functional relevance and sequence-level naturalness.

E.1 PREDICTED ACTIVITY (PRED-ACTIVITY)

To estimate the functional activity of generated sequences, we employ a pre-trained sequence-to-activity regression model as an oracle. Given a generated DNA sequence, the oracle predicts enhancer activity levels across multiple human cell lines. In this work, we report results on the HepG2 cell line. The Pred-Activity score is computed as the average predicted activity over a set of generated sequences

$$\text{Pred-Activity} = \frac{1}{N}\sum_{i=1}^{N} f_{\text{act}}(x_i),$$

where x_i denotes a generated sequence and $f_{\text{act}}(\cdot)$ represents the oracle-predicted activity score for HepG2.

E.2 CHROMATIN ACCESSIBILITY (ATAC-ACC)

To further evaluate regulatory plausibility, we assess chromatin accessibility using an independent binary classification oracle trained on ATAC-seq data. For each generated sequence, the oracle predicts the probability of the sequence being accessible in a given cell line. We report the average predicted accessibility score for HepG2

$$\text{ATAC-Acc} = \frac{1}{N}\sum_{i=1}^{N} g_{\text{ATAC}}(x_i),$$

where $g_{\text{ATAC}}(\cdot)$ denotes the output of the ATAC accessibility oracle. This metric complements Pred-Activity by capturing structural regulatory properties beyond expression-level predictions.

E.3 k-MER DISTRIBUTION CORRELATION (3-MER CORR)

To quantify the extent to which generated sequences preserve natural genomic statistics, we compute the Pearson correlation between the 3-mer frequency distributions of generated sequences and those observed in highly active real enhancers. Specifically, we extract the top 1% of sequences with the highest HepG2 activity from the dataset and compute their empirical 3-mer distribution as a reference. Let c^{gen} and c^{ref} denote the aligned 3-mer count vectors for generated and reference sequences. The 3-mer correlation is defined as

$$\text{3-mer Corr} = \mathrm{PearsonCorr}\left(c^{\text{gen}}, c^{\text{ref}}\right).$$

Higher values indicate that the generated sequences better match the local compositional patterns of naturally occurring regulatory elements.

E.4 MOTIF FREQUENCY CORRELATION (JASPAR CORR)

We additionally evaluate transcription factor motif usage by scanning generated sequences with position weight matrices from the JASPAR database. For each motif, we compute its occurrence frequency in the generated sequences and compare it with the corresponding frequency observed in a reference set of highly active HepG2 enhancers (top 0.1% by activity). Then, we compute the Spearman rank correlation between the two motif frequency vectors

$$\text{JASPAR Corr} = \mathrm{SpearmanCorr}\left(m^{\text{gen}}, m^{\text{ref}}\right),$$

where m^{gen} and m^{ref} denote motif frequency vectors for generated and reference sequences. Spearman's rank correlation measures the strength of a monotonic relationship between two variables by comparing the ranks of their values. This metric evaluates whether the relative enrichment of transcription factor binding motifs is preserved.