CONTROLLED DNA DESIGN VIA ROLLOUT REASONING WITH DISCRETE FLOW MATCHING

Anonymous authors
Paper under double-blind review

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 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 demonstrate 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 controllable DNA sequence design. Synthetic DNA design has emerged as a cornerstone of modern genomics, enabling systematic interrogation of gene regulation and programmable engineering of functional therapeutic sequences (Gasperini et al., 2020; Shlyueva et al., 2014; Gustafsson et al., 2004). However, designing a functional regulatory sequence is not a simple "fill-in-the-blank" task. It is a high-dimensional, combinatorial optimization problem over a discrete sequence space, where each nucleotide choice can reshape local motif syntax, long-range regulatory grammar, and the ultimate functional activity of the strand.

To navigate this 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 an irreversible, stochastic denoising trajectory. In the context of controlled DNA design, this leads to two critical failures: (i) Loss of control: Stochastic sampling adds noise during generation. This noise makes it harder to guide the model toward specific biological constraints. (ii) Inability to revise: Once a diffusion model makes a nucleotide choice early in the denoising process, that choice is locked in. There is no mechanism to "re-read" the sequence, identify a structural flaw, or backtrack. For instance, if a model places a repressor motif early in the trajectory, it cannot pivot to optimize for high enhancer activity later. This means that the model lacks the agency to fix its own mistakes.

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). By defining a direct, deterministic mapping from noise to data, we establish a "straight-line" path to the target sequence. This avoids the randomness of traditional diffusion sampling. It provides clearer control over the generation path and strengthens the link between the model's intermediate decisions and the final biological objective.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

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 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). 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. This allows for the end-to-end backpropagation. Our contributions are threefold:

- **Discrete Flow Matching as a Controllable DNA Design Backbone.** We introduce a discrete flow-matching formulation for regulatory DNA sequence generation, replacing stochastic diffusion trajectories with a more structured masked-to-sequence transport path. Unlike conventional discrete diffusion samplers that rely on noisy, single-pass denoising, our flow-matching backbone follows a direct probability path from fully masked sequences to functional DNA. This design provides stronger trajectory control, improves the alignment between intermediate token decisions and downstream biological rewards, and enables more efficient reward-guided optimization.
- **Reasoning-via-refinement for Reward-aligned Sequence Generation.** We propose an iterative rollout reinforcement learning framework that turns DNA design from one-shot generation into a multi-round revise-and-improve process. At each rollout stage, the model partially remasks its current draft sequence and resamples uncertain regions. This progressive refinement schedule, e.g., $100\% \to 80\% \to 50\%$, allows the generator to preserve high-confidence regulatory context while correcting low-quality motifs and improving global enhancer activity.
- **State-of-the-art Reward-optimized Enhancer Design.** We validate DREAM-DNA on the human HepG2 enhancer design. DREAM-DNA achieves the best overall functional performance, improving predicted enhancer activity by 13% and chromatin accessibility by 8% over the best baseline while maintaining sequence-level naturalness. Ablation studies further show that both components are essential: discrete flow matching provides a stronger optimization backbone, and three-step rollout refinement yields the best activity-accessibility trade-off.

Organization. We provide the preliminaires in Section 2, detailed methods in Section 3, experimental studies in Section 4, and related works in Section B.

2 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.

2.1 Problem Setting

We study the reward-guided model for 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 or chromatin accessibility) is present, 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 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:

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

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 $p^{\text{pre}}(\cdot \mid c)$.

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

$$\Pr(x_{t+\Delta t} = y \mid x_t = x) = \mathbb{I}(x = y) + Q^\theta_{x,y}(t)\, \Delta t, \tag{2.1}$$

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, chromatin accessibility, or motif-based criteria). A naive objective maximizes expected reward under the model:

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

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 $Q^\theta_{x_t,\cdot}(t)$ 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:

$$\max_{\theta \in \Theta} \mathbb{E}_{x_{0:T} \sim P^\theta(\cdot|c)} \Big[r(x_T) - \alpha \mathcal{L}_{KL}\Big], \tag{2.2}$$

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

2.2 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. DRAKES uses the same CTMC parameterization as in Section 2.1: 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 (2.1) gives:

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

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

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

and samples $x_{t+\Delta t} \sim \mathrm{Cat}(\pi_t)$, where $\mathrm{Cat}(\pi_t)$ is a random variable 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:

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

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.

Algorithm 1 DRAKES (**D**irect **R**eward b**A**c**K**propagation with the Gumb**E**l-**S**oftmax trick)

Require: pre-trained generator $Q^{\mathrm{pre}}(t)$, 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** $s \in \{1, \dots, S\}$ **do**
2: **for** $i \in \{1, \dots, B\}$ **do**
3: **for** $t \in \{0, \Delta t, \dots, T\}$ **do**
4: Set $\pi(t) \in \Delta(\mathcal{X})$ where

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

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

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

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

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

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}

Optimization (Step 9–10). DRAKES optimizes the objective in (2.2). It estimates gradients by differentiating through the relaxed trajectory $\bar{x}_{0:T}$ and then updates θ. To reduce memory cost, DRAKES truncates backpropagation and only differentiates through a suffix of the trajectory.

3 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.

3.1 Discrete Flow Matching for DNA Design

Let $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ and add a mask token $[\mathrm{MASK}]$, so the vocabulary size $d = 5$. We consider length-N sequences $x = (x^1, \dots, x^N) \in \mathcal{D} \triangleq \Sigma^N_{\mathrm{mask}}$, 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., $M^i = [\mathrm{MASK}]$ for all $i \in \{1, \dots, N\}$, and we set x_1 to a natural DNA sequence. We define a simple conditional path using a scalar schedule $\kappa_t \in [0, 1]$ with $\kappa_0 = 0$ and $\kappa_1 = 1$:

$$p_t(x^i \mid x_0, x_1) = (1 - \kappa_t)\, \mathbb{I}(x^i = x^i_0) + \kappa_t\, \mathbb{I}(x^i = x^i_1).$$

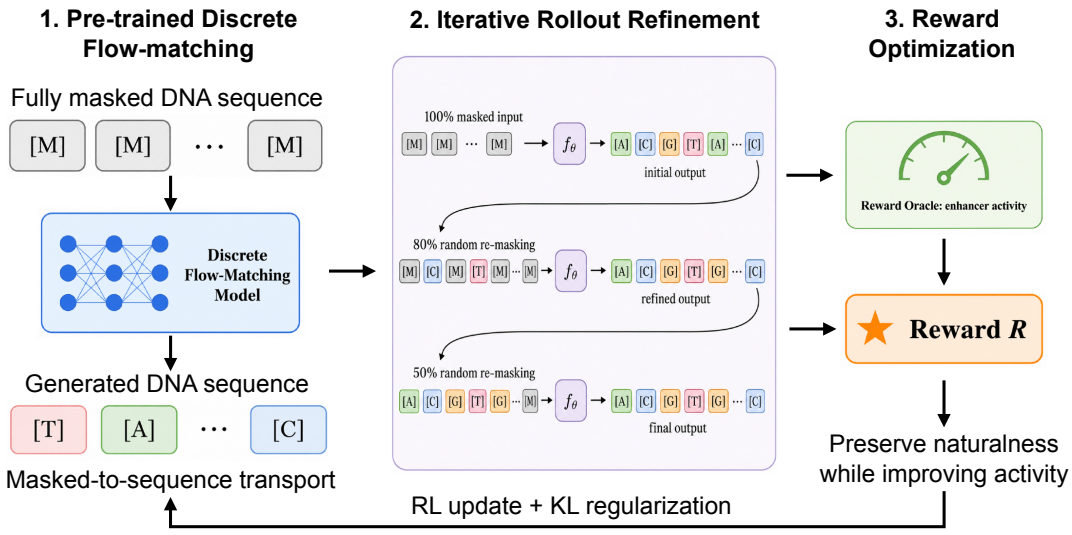


Figure 1: **Overview of DREAM-DNA.** This generative pipeline integrates a pre-trained discrete flow matching backbone with an iterative rollout loop that treats sequence generation as a multi-round reasoning process. In each cycle, the model critiques its own output through progressive random remasking (100% $\rightarrow$ 80% $\rightarrow$ 50%) 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 (see Algorithm 1), gradients are backpropagated through all rollout cycles to update model parameters θ end-to-end. The figure was inspired by GPT Image 2, and some subfigures were generated with its assistance.

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 $\mathcal{D}$,

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

where π_t comes from a CTMC rate matrix R_t. $R_t(x, y) \geq 0$ 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:

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

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:

$$q^i_{\theta,t}(a \mid x_t) = \mathrm{softmax}(f_\theta(x_t, t)^i)_a,\ a \in \Sigma_{\mathrm{mask}}$$

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

$$R^\theta_t(x, y) = \lambda(t)\, q^i_{\theta,t}(y^i \mid x),$$

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

$$R^\theta_t(x, x) = -\sum_{y \neq x} R^\theta_t(x, y).$$

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

Algorithm 2 Iterative Rollout Mechanism

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

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

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

end for
Output: θ_{S+1}

3.2 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 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 with probability $\sigma_t \in [0, 1]$:

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

We define the linear schedule by $\sigma_t^{\mathrm{linear}} = (t+1)/T$. Then we sharpen it with a square schedule:

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

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

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

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.

3.3 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.

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

Table 1: **Comparison of DNA Design Methods across Functional and Naturalness Metrics.** We compare our proposed flow-matching and iterative rollout mechanism with different 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 Diffusion	0.17 (0.01)	1.5 (0.3)	-0.054 (0.061)	-0.105 (0.079)
Pre-trained Flow-matching	1.92 (1.93)	4.2 (7.22)	0.453 (0.276)	0.467 (0.181)
DRAKES	5.88 (0.04)	96.2 (1.6)	0.804 (0.002)	0.709 (0.013)
DREAM-DNA (no rollout)	6.56 (0.02)	93.2 (0.3)	0.300 (0.001)	0.223 (0.010)
DREAM-DNA	**6.74 (0.07)**	**98.0 (1.4)**	0.500 (0.001)	0.398 (0.056)

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 (as in (2.2)).

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:

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

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.

4 Experimental Study

In this section, we present the experimental results on HepG2 enhancer design in Section 4.1 and provide a detailed ablation study for DREAM-DNA in Section 4.2. Furthermore, we include an additional benchmark on cell-type-specific DNA regulatory sequence design in Section C to complement the enhancer design benchmark. To mitigate potential reward hacking, we also train and evaluate generated sequences with an additional CNN oracle reward model in Section D.

4.1 Task: HepG2 Enhancer Design

In this part, we compare our approach against diverse baselines and present our main results.

Dataset and Setting. We use a large-scale enhancer dataset containing activity levels for 700,000 DNA sequences. Each sequence is 200 base pairs long. We use a discrete diffusion model and a discrete flow-matching model as backbones. We split the dataset to train two separate reward models. They use the Enformer architecture to predict how active an enhancer is in human HepG2 cell lines.

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 with techniques that use gradients or sampling to guide generation. These include classifier guidance (CG) (Nisonoff et al., 2024), sequential Monte Carlo (SMC) methods, 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 downstream 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.

Evaluation Metrics. We use four key metrics (see Section I for detailed calculation methods):

- Predicted Activity (Pred-Activity): This regression model predicts the enhancer activity level. Higher scores indicate more functional sequences.
- Chromatin Accessibility (ATAC-Acc): This binary classifier predicts if a designed sequence fits the profile of open chromatin in HepG2 cells. The higher, the better.
- 3-mer Pearson Correlation (3-mer Corr): This measures how well the nucleotide patterns in our synthetic sequences match the top-performing natural sequences. Higher scores are better.
- JASPAR Motif Analysis (JASPAR Corr): We count transcription factor binding motifs using the JASPAR database. We then calculate the correlation between our sequences and high-activity natural enhancers. Higher scores indicate greater naturalness.

Main Results. We report the main results in Table 1. First, the flow-matching backbone consistently outperforms diffusion baselines, achieving higher Pred-Activity and ATAC-Acc. The pre-trained flow-matching model provides a stronger starting point than pre-trained diffusion and ultimately improves predicted enhancer activity by 13% and ATAC-Acc by 8% over the state-of-the-art DRAKES baseline. These gains suggest that deterministic straight-line transport provides more stable trajectory control and better aligns intermediate generation steps with the final biological objective.

Second, iterative rollout further improves both backbones by allowing the model to revise previously generated tokens. DREAM-DNA achieves the best Pred-Activity and 98% ATAC-Acc in Table 1. Its key advantage comes from a multi-round refinement process: instead of committing to early nucleotide choices, the model remasks and resamples low-confidence regions using reward feedback while preserving high-confidence regions. This enables "reasoning-via-refinement," allowing DREAM-DNA to correct unfavorable motifs and progressively optimize functional activity.

4.2 Ablation Studies

We conduct ablation studies to isolate the contribution of iterative rollout and to understand which design choices are most important. Specifically, we study four questions: (i) whether rollout refinement improves DNA design; (ii) how many rollout cycles are needed; (iii) whether the benefit comes from conditional iterative refinement rather than simply sampling more candidates; and (iv) whether DREAM-DNA preserves sequence diversity under reward optimization.

Effect of Rollout Depth. We first ablate the number of rollout cycles in Table 2. Across both diffusion and flow-matching backbones, rollout improves functional performance over the no-rollout variants, showing that multi-round refinement provides benefits beyond single-pass generation. For the diffusion backbone, three rollout cycles improve ATAC-Acc from 90.2% to 98.35% and increase Pred-Activity from 5.95 to 6.67. For DREAM-DNA, three rollout cycles increase ATAC-Acc from 92.9% to 98.0% while maintaining the best overall balance across activity and naturalness metrics. Increasing the rollout depth to four cycles slightly degrades performance, suggesting that excessive remasking can disrupt useful sequence context. Thus, three rollout cycles provide the best trade-off.

Effect of Rollout Schedule. We examine how the rollout schedule affects the trade-off between predicted activity and sequence naturalness (Table 3). The decreasing 100-80-50 schedule gives the best overall balance across Pred-Activity, ATAC-Acc, 3-mer correlation, and JASPAR correlation. Keeping remasking high at 100-100-100 preserves activity but reduces naturalness, while 100-70-30 yields high ATAC-Acc at the expense of 3-mer and motif correlations. These results favor gradually shrinking the editable region to retain confident sequence context while refining uncertain positions. Under a matched evaluation budget, Best-of-3 generates three independent candidates and selects the highest-reward sample. Its lower performance than 100-80-50 suggests that conditional iterative refinement contributes beyond selecting among independent samples.

Table 2: **Ablation Study on Rollout Depth.** We compare different numbers of rollout cycles across diffusion and flow-matching backbones. Three rollout cycles achieve the best overall trade-off.

Backbone	Rollout Steps	Pred-Activity ↑	ATAC-Acc ↑ (%)	3-mer Corr ↑	JASPAR Corr ↑
DRAKES	w/o Rollout	5.95 (0.26)	90.2 (4.6)	0.368 (0.087)	**0.543 (0.141)**
	Rollout ×2	6.34 (0.03)	88.0 (2.1)	0.375 (0.091)	0.342 (0.112)
	Rollout ×3	**6.67 (0.04)**	**98.4 (1.9)**	**0.457 (0.096)**	0.489 (0.059)
	Rollout ×4	6.32 (0.05)	88.6 (2.3)	0.381 (0.079)	0.424 (0.091)
DREAM-DNA	w/o Rollout	6.59 (0.06)	92.9 (4.3)	0.427 (0.002)	0.280 (0.050)
	Rollout ×2	**6.76 (0.05)**	94.2 (1.9)	0.463 (0.005)	0.360 (0.087)
	Rollout ×3	6.74 (0.07)	**98.0 (1.4)**	**0.500 (0.001)**	**0.398 (0.056)**
	Rollout ×4	6.55 (0.03)	95.9 (0.8)	0.411 (0.005)	0.332 (0.010)

Table 3: **Ablation Study on Rollout Schedule and Best-of-3.** A progressively decreasing remasking schedule, 100-80-50, achieves the best overall performance. The matched-budget Best-of-3 baseline underperforms rollout, showing that conditional refinement is more effective.

Schedule	Pred-Activity ↑	ATAC-Acc ↑ (%)	3-mer Corr ↑	JASPAR Corr ↑
Rollout 100-100-100	6.66 (0.02)	96.4 (1.0)	0.327 (0.001)	0.196 (0.030)
Rollout 100-90-70	6.51 (0.01)	94.9 (0.8)	0.264 (0.005)	0.080 (0.019)
Rollout 100-80-50	**6.74 (0.07)**	98.0 (1.4)	**0.500 (0.010)**	**0.398 (0.056)**
Rollout 100-70-30	6.72 (0.01)	**98.2 (0.8)**	0.309 (0.001)	0.169 (0.024)
Best-of-3	6.46 (0.03)	96.7 (0.6)	0.330 (0.001)	0.196 (0.028)

Effect of Remasking Strategy. We further test whether the choice of remasked positions is critical by replacing random remasking with entropy-based remasking. The entropy-based variant prioritizes uncertain positions while keeping the rollout depth and remask ratios fixed. As shown in Table 4, entropy-based remasking slightly improves ATAC-Acc from 98.0% to 98.4%, but it reduces Pred-Activity, 3-mer correlation, and JASPAR correlation. This result suggests that uncertainty-aware remasking is a promising direction, but it does not outperform simple random remasking under the current setup. More importantly, it indicates that the progressive rollout schedule plays a larger role than the specific remask-selection heuristic when the computational budget is matched.

Table 4: **Effect of Entropy-based Remasking.** Entropy-based remasking slightly improves ATAC-Acc but degrades Pred-Activity and naturalness metrics. Thus, it does not outperform random remasking under the same rollout depth and remask-ratio setting.

Schedule	Pred-Activity ↑	ATAC-Acc ↑ (%)	3-mer Corr ↑	JASPAR Corr ↑
Random Rollout (DREAM-DNA)	**6.74 (0.07)**	98.0 (1.4)	**0.500 (0.010)**	**0.398 (0.056)**
Entropy-based Rollout	6.28 (0.18)	**98.4 (1.9)**	0.334 (0.013)	0.220 (0.054)

Diversity of Designed Sequences. Reward optimization may concentrate generated sequences around high-scoring regions and reduce diversity. We assess this risk using 3-mer distribution entropy (Section I.5 and Table 5), where higher values indicate greater compositional diversity. DREAM-DNA achieves an entropy of 5.09, close to DRAKES at 5.12. This small difference suggests that DREAM-DNA maintains substantial 3-mer diversity while improving predicted functional activity, with no evidence of severe compositional mode collapse.

Table 5: **Diversity Comparison.**

Model	Entropy ↑
DRAKES	5.12
DREAM-DNA	5.09

5 Conclusion

We introduce DREAM-DNA, a controllable DNA design framework that combines discrete flow matching with rollout-based reinforcement learning. DREAM-DNA replaces stochastic single-pass denoising with a structured masked-to-sequence transport path and further enables iterative sequence refinement. On HepG2 enhancer design, DREAM-DNA improves predicted enhancer activity and chromatin accessibility over prior reward-optimization baselines, while maintaining comparable sequence diversity. Ablation studies show that both the flow-matching and the rollout refinement contribute to these gains. We provide limitations in Section A and related works in Section B.

AI USE STATEMENT

We used large language models to edit the manuscript for clarity, grammar, and conciseness, and to help identify relevant literature. We used Claude Code to assist with writing and debugging portions of the experimental code, organizing scripts, and running experiments. We also used GPT Image 2 to explore the overview figure and generate some of its subfigures. The authors checked references against their sources, reviewed and tested AI-assisted code, and verified the text, figures, and reported results. The authors designed the method and experiments, interpreted the results, and took full responsibility for the paper.

ETHICS STATEMENT

DREAM-DNA designs synthetic regulatory DNA sequences and is evaluated computationally. This work did not involve human participants or experimental testing of generated sequences. The reported activity and accessibility gains are predictions from computational models, not demonstrated biological effects. Generated sequences could potentially be used to alter gene regulation, so experimental use should follow appropriate institutional oversight and biosafety procedures. We report the limitations of the reward models and the observed trade-off with sequence naturalness to avoid overstating the biological validity of the designs.

REPRODUCIBILITY STATEMENT

Section 3 describes DREAM-DNA's training objective and iterative rollout procedure, while Section 4 specifies the evaluation setting, baselines, and metrics. The appendices provide implementation and sampling details, dataset construction, and metric definitions. In particular, Appendix F reports the optimizer, learning rate, batch sizes, random seed, hardware, and sampling procedure. Appendix H describes the evaluation subsets and notes that they contain sequences observed during training; the results therefore measure in-distribution performance rather than generalization to unseen conditions.

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.

Molly Gasperini, Jacob M Tome, and Jay Shendure. Towards a comprehensive catalogue of validated and target-linked human enhancers. *Nature Reviews Genetics*, 21(5):292–310, 2020.

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.

Claes Gustafsson, Sridhar Govindarajan, and Jeremy Minshull. Codon bias and heterologous protein expression. *Trends in biotechnology*, 22(7):346–353, 2004.

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.

Daria Shlyueva, Gerald Stampfel, and Alexander Stark. Transcriptional enhancers: from properties to genome-wide predictions. *Nature Reviews Genetics*, 15(4):272–286, 2014.

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.

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.

A LIMITATIONS

DREAM-DNA has the following five limitations.

- DREAM-DNA does not yet achieve state-of-the-art performance on cell-type-specific regulatory sequence design, as shown in Section C. On the Ctrl-DNA benchmark, DREAM-DNA improves over several baselines such as Ctrl-DNA-Log and Ctrl-DNA-IPO in multiple target settings, but the strongest Ctrl-DNA variants with tuned $\lambda_{\max}$ still achieve larger cell-type specificity margins. This suggests that the current reward formulation is effective for improving target-cell activity, but it does not sufficiently suppress off-target activity in non-target cell types. Future work should incorporate explicit multi-objective rewards that jointly optimize target activation, off-target repression, sequence diversity, and biological naturalness.
- Reward optimization can introduce a trade-off between functional activity and sequence naturalness. Although DREAM-DNA improves predicted enhancer activity and chromatin accessibility, its motif-level naturalness, measured by JASPAR correlation, remains lower than some baselines. This indicates that optimizing a learned activity reward may shift generated sequences toward high-scoring regions that do not fully preserve natural regulatory motif composition. Future work should use stronger naturalness constraints, such as motif-distribution regularization, evolutionary conservation priors, or reference-genome likelihood penalties.
- DREAM-DNA still depends on learned computational reward oracles. Although we use a separate CNN oracle to test whether the gains transfer beyond the optimization-time reward model, both oracles remain predictive models rather than experimental measurements. Therefore, the reported gains should be interpreted as in silico improvements. Future work should validate generated sequences with MPRA or other experimental assays and evaluate whether predicted improvements translate into real regulatory activity.
- The rollout mechanism introduces additional design choices. Performance depends on the number of rollout cycles and the remasking schedule. In our experiments, three rollout cycles with a progressively decreasing remasking schedule provide the best trade-off, while more rollout cycles can degrade performance by disrupting useful sequence context. This suggests that fixed rollout schedules may be suboptimal across tasks, cell types, and reward functions. Adaptive remasking strategies that identify low-reward or biologically implausible regions may improve robustness.
- Our current backbone and evaluation setting remain limited in scale. DREAM-DNA uses a relatively simple one-dimensional convolutional architecture and is evaluated mainly on MPRA-style regulatory design tasks. Larger sequence models, broader regulatory datasets, longer genomic contexts, and multi-cell-type training may improve generalization to more complex settings.

B Related Works

In this section, we review related work on generative models for DNA sequence generation, reinforcement learning for aligning generative models with biological objectives, and reasoning-inspired refinement strategies that improve downstream design performance.

B.1 Generative Models for DNA Generation

DNA generative modeling 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 (e.g., 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.

B.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 interaction exposes sampling-time failures and enables multi-step, reward-informed refinement.

B.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.

C Additional Evaluation for Cell-type-specific DNA Design

To further evaluate whether DREAM-DNA generalizes beyond the HepG2 enhancer design, we test on the Ctrl-DNA benchmark for cell-type-specific regulatory DNA design (Chen et al., 2025b). This benchmark evaluates whether a generated sequence achieves high activity in a target cell type while suppressing activity in non-target cell types. We evaluate three settings: HepG2, K562, and SK-N-SH. For each target cell type, we report the predicted activity in the target cell type, the predicted activity in the two non-target cell types, the cell-type specificity score ΔR, and sequence diversity.

Following Chen et al. (2025b), ΔR measures the margin between target-cell activity and off-target activity. Higher ΔR indicates stronger cell-type-specific control. Diversity measures the diversity of generated sequences, where higher values indicate greater diversity. We compare DREAM-DNA with Ctrl-DNA-Log, Ctrl-DNA-IPO, and Ctrl-DNA variants with different $\lambda_{\max}$ values.

HepG2 as Target. Table 6 reports results when HepG2 is the target cell type. DREAM-DNA achieves higher target activity and larger ΔR than Ctrl-DNA-Log and Ctrl-DNA-IPO. In particular, DREAM-DNA obtains a ΔR of 0.16, while Ctrl-DNA-Log and Ctrl-DNA-IPO obtain 0.02 and -0.10, respectively. However, the strongest Ctrl-DNA variants with tuned $\lambda_{\max}$ achieve larger specificity margins, with ΔR values between 0.42 and 0.45. Thus, DREAM-DNA improves over the log- and IPO-based Ctrl-DNA baselines, but does not outperform the strongest Ctrl-DNA variants.

Table 6: **Ctrl-DNA Benchmark with HepG2 as the Target Cell Type.** Higher HepG2 activity and higher ΔR indicate better target-cell specificity, while lower K562 and SK-N-SH activities indicate stronger off-target suppression.

Method	HepG2 ↑	K562 ↓	SK-N-SH ↓	ΔR ↑	Diversity ↑
DREAM-DNA	0.61 (0.01)	0.43 (0.01)	0.46 (0.02)	0.16 (0.01)	1.63 (0.05)
Ctrl-DNA-Log	0.24 (0.02)	0.24 (0.03)	0.21 (0.04)	0.02 (0.06)	1.62 (0.08)
Ctrl-DNA-IPO	0.74 (0.01)	0.86 (0.02)	0.83 (0.02)	-0.10 (0.02)	1.58 (0.13)
Ctrl-DNA ($\lambda_{\max} = 0.00$)	0.78 (0.02)	0.40 (0.06)	0.33 (0.03)	0.42 (0.02)	1.82 (0.02)
Ctrl-DNA ($\lambda_{\max} = 0.01$)	0.77 (0.01)	0.34 (0.04)	0.30 (0.02)	0.45 (0.02)	1.84 (0.01)
Ctrl-DNA ($\lambda_{\max} = 0.10$)	0.77 (0.01)	0.36 (0.04)	0.31 (0.04)	0.44 (0.03)	1.82 (0.04)

K562 as Target. Table 7 reports results when K562 is the target cell type. DREAM-DNA achieves a ΔR of 0.21, outperforming Ctrl-DNA-Log and Ctrl-DNA-IPO, which obtain ΔR values of 0.11 and 0.15, respectively. DREAM-DNA also maintains higher diversity than Ctrl-DNA-Log and Ctrl-DNA-IPO. However, the strongest Ctrl-DNA variants achieve substantially higher K562 activity and larger specificity margins, with $\Delta R = 0.54$ across all three $\lambda_{\max}$ settings. These results again show that DREAM-DNA is competitive with several Ctrl-DNA baselines but does not outperform the strongest tuned Ctrl-DNA variants.

Table 7: **Ctrl-DNA Benchmark with K562 as the Target Cell Type.** Higher K562 activity and higher ΔR indicate better target-cell specificity, while lower HepG2 and SK-N-SH activities indicate stronger off-target suppression.

Method	K562 ↑	HepG2 ↓	SK-N-SH ↓	ΔR ↑	Diversity ↑
DREAM-DNA	0.67 (0.01)	0.48 (0.01)	0.45 (0.01)	0.21 (0.01)	1.80 (0.02)
Ctrl-DNA-Log	0.28 (0.01)	0.16 (0.06)	0.19 (0.02)	0.11 (0.02)	1.68 (0.16)
Ctrl-DNA-IPO	0.84 (0.11)	0.65 (0.09)	0.73 (0.19)	0.15 (0.03)	1.08 (0.62)
Ctrl-DNA ($\lambda_{\max} = 0.00$)	0.93 (0.01)	0.43 (0.01)	0.35 (0.01)	0.54 (0.01)	1.72 (0.02)
Ctrl-DNA ($\lambda_{\max} = 0.01$)	0.93 (0.01)	0.42 (0.05)	0.35 (0.01)	0.54 (0.06)	1.73 (0.05)
Ctrl-DNA ($\lambda_{\max} = 0.10$)	0.93 (0.01)	0.43 (0.01)	0.35 (0.01)	0.54 (0.01)	1.82 (0.04)

SK-N-SH as Target. Table 8 reports results when SK-N-SH is the target cell type. DREAM-DNA achieves higher target activity than Ctrl-DNA-Log and a larger ΔR than Ctrl-DNA-IPO. However, Ctrl-DNA-Log obtains a larger ΔR because it strongly suppresses off-target HepG2 and K562

activity, despite having lower target-cell activity. The strongest Ctrl-DNA variant with $\lambda_{\max} = 0.01$ achieves the best overall specificity, with $\Delta R = 0.49$ and the highest diversity. Thus, DREAM-DNA remains competitive but is not the best method on this target.

Table 8: **Ctrl-DNA Benchmark with SK-N-SH as the Target Cell Type.** Higher SK-N-SH activity and higher ΔR indicate better target-cell specificity, while lower HepG2 and K562 activities indicate stronger off-target suppression.

Method	**SK-N-SH** $\uparrow$	**HepG2** $\downarrow$	**K562** $\downarrow$	$\Delta R \uparrow$	**Diversity** $\uparrow$
DREAM-DNA	0.68 (0.01)	0.53 (0.03)	0.43 (0.01)	0.19 (0.02)	1.67 (0.04)
Ctrl-DNA-Log	0.46 (0.07)	0.05 (0.01)	0.05 (0.01)	0.41 (0.07)	1.68 (0.08)
Ctrl-DNA-IPO	0.91 (0.04)	0.87 (0.01)	0.70 (0.01)	0.13 (0.03)	1.66 (0.07)
Ctrl-DNA ($\lambda_{\max} = 0.00$)	0.83 (0.04)	0.57 (0.11)	0.38 (0.07)	0.35 (0.10)	1.78 (0.03)
Ctrl-DNA ($\lambda_{\max} = 0.01$)	0.88 (0.01)	0.47 (0.05)	0.30 (0.03)	0.49 (0.03)	1.84 (0.04)
Ctrl-DNA ($\lambda_{\max} = 0.10$)	0.86 (0.02)	0.54 (0.13)	0.44 (0.01)	0.37 (0.11)	1.82 (0.04)

Summary. Overall, the Ctrl-DNA benchmark provides a broader evaluation of cell-type-specific controllable DNA design. DREAM-DNA improves over Ctrl-DNA-Log and Ctrl-DNA-IPO in several target settings, especially in terms of balancing target-cell activity, specificity, and diversity. However, DREAM-DNA does not outperform the strongest Ctrl-DNA variants with tuned $\lambda_{\max}$.

D External CNN Oracle Evaluation for Reward Hacking Testing

Because DREAM-DNA directly optimizes the trained reward model, a potential concern is reward hacking, where generated sequences exploit artifacts of the reward model rather than achieving genuine biological activity. To address this concern, we additionally evaluate generated sequences using an external oracle trained on the same dataset but with a different architecture from the optimization-time reward model. Specifically, we train a CNN-based regression oracle on the same processed human cell-line MPRA dataset used by DRAKES. The oracle predicts HepG2 activity and is used only for post-hoc evaluation, not for training DREAM-DNA.

CNN Oracle Architecture. Our external oracle is a one-dimensional convolutional regressor. It takes a one-hot encoded DNA sequence as input and uses three convolutional blocks:

- Conv1d from 4 to 64 channels with kernel size 15 and padding 7, followed by ReLU and MaxPool1d(2);
- Conv1d from 64 to 128 channels with kernel size 7 and padding 3, followed by ReLU and MaxPool1d(2);
- Conv1d from 128 to 256 channels with kernel size 7 and padding 3, followed by ReLU.

We then apply adaptive max pooling over the sequence length and use two fully connected layers, $256 \to 128 \to 1$, with ReLU and dropout rate 0.2. The model outputs a single scalar activity prediction for each sequence.

Training Details. We train the CNN oracle as a plain supervised regressor using mean squared error between the predicted activity and the target activity label. We optimize the model with AdamW using learning rate 10^{-3}, weight decay 10^{-4}, and gradient clipping with maximum norm 1.0. We select the checkpoint with the best validation Spearman correlation. Because the oracle is trained directly on the processed datasets released with DRAKES, its prediction scale is inherited from the original activity labels. The oracle does not include any normalization or post-processing step that would systematically compress or inflate predicted values.

External Oracle Results. Table 9 reports the CNN oracle scores for sequences generated by different methods. DREAM-DNA achieves the highest external CNN oracle score, improving over both the pre-trained flow-matching backbone and DRAKES. This result provides additional evidence that the gains of DREAM-DNA are not purely due to overfitting or reward hacking.

Table 9: **External CNN Oracle Evaluation.** We evaluate generated sequences using a separately trained CNN oracle for HepG2 activity. DREAM-DNA achieves the highest score, suggesting that its improvement transfers to an external predictor with a different architecture.

Method	CNN Oracle Score ↑
Pretrained Flow-matching	0.481
DRAKES	2.910
DREAM-DNA	**3.525**

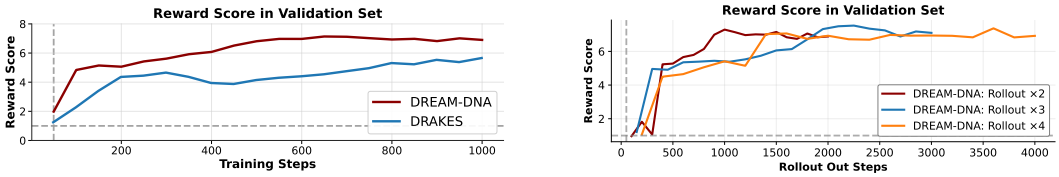
In summary, the external CNN oracle uses a different architecture from the reward model used for optimization, but DREAM-DNA still obtains the highest predicted activity. This suggests that the optimized sequences capture activity patterns that generalize across predictors rather than exploiting idiosyncrasies of a single reward model.

E VALIDATION REWARD CURVES FOR ITERATIVE ROLLOUT

We provide additional validation reward curves to analyze the optimization behavior of DREAM-DNA and the effect of iterative rollout. These curves report the reward score on the validation set during training and are used only to compare reward optimization dynamics.

Figure 2a compares DREAM-DNA and DRAKES under the no-rollout setting. DREAM-DNA reaches a higher validation reward than DRAKES throughout most of training and shows a stronger reward plateau. This suggests that, under the same reward model, DREAM-DNA optimizes the validation reward more effectively than the DRAKES baseline.

Figure 2b compares DREAM-DNA with different numbers of rollout cycles. The rollout variants all improve rapidly at early training stages and reach similar high-reward regimes. Among them, the three-cycle rollout achieves the highest validation reward over a substantial portion of training, while the two-cycle and four-cycle settings show comparable but slightly lower. Based on these validation reward curves, we use three rollout cycles as the default configuration in our main experiments.



(a) DREAM-DNA vs. DRAKES (No Rollout)

(b) Impact of Rollout Steps ($\times 2$, $\times 3$, $\times 4$)

Figure 2: **Validation Reward Curves for Controlled DNA Design.** (a) DREAM-DNA without rollout reaches a higher validation reward than the DRAKES baseline under the same no-rollout setting. (b) Validation reward trajectories of DREAM-DNA with different rollout cycles. The three-cycle setting ($\times 3$) achieves the highest reward, motivating its use as the default configuration.

F Implementation Details

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

F.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.

F.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.

G 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.

G.1 SEQUENCE REPRESENTATION

Each DNA sequence is represented as a categorical discrete sequence where nucleotides are mapped to integer tokens: `A` $\to 0$, `C` $\to 1$, `G` $\to 2$, and `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.

G.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.

H 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.

Evaluation Data Representation. Evaluation samples share the same representation as training samples. An attention mask of all ones indicates that all sequence positions are valid.

Data Loading for Evaluation. We use PyTorch `DataLoaders` 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.

I 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.

I.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.

I.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.

I.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

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

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

I.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

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

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.

I.5 3-MER DISTRIBUTION ENTROPY

We quantify the compositional diversity of generated sequences by analyzing the distribution of 3-mers. Specifically, we count all 3-mers across the generated set and normalize the counts to obtain a probability distribution p_k. The entropy is then computed as

$$H = -\sum_k p_k \log p_k,$$

where p_k denotes the normalized frequency of the k-th 3-mer. Entropy measures the uncertainty of the distribution, with higher values indicating a more uniform usage of 3-mers and thus greater compositional diversity. Conversely, lower entropy reflects a more concentrated distribution, suggesting reduced diversity in sequence composition.