Flexible Flows for Biological Sequence Design

Yogesh Verma*
Aalto University, Finland
yogesh.verma@aalto.fi

Dani Korpela
Aalto University, Finland
dani.korpela@aalto.fi

Harri Lähdesmäki
Aalto University, Finland
harri.lahdesmaki@aalto.fi

Vikas Garg
Aalto University and YaiYai Ltd
vgarg@csail.mit.edu

Abstract

Designing functional biological sequences requires navigating vast discrete spaces under strict evolutionary and biophysical constraints. Discrete Flow Matching (DFM) offers a generative framework over such spaces, but existing approaches rely on biologically uninformative couplings and offer limited flexibility for variable length sequence generation and fine-grained control. We propose a structured coupling that encodes domain-specific preferences among sequence elements, biasing the source distribution toward plausible regions without modifying the flow objective or training procedure. Building on this, we introduce a latent edit-based rate parameterization that models variable-length generation via edit operations conditioned on a shared global latent, akin to a latent variable model, while remaining tractable. We further introduce a latent classifier-free guidance mechanism that steers generation coherently in continuous latent space, along with Dirichlet-prior temperature scaling for test-time control over edit operations. Our method achieves state-of-the-art performance across diverse biological sequence tasks, including density estimation, unconditional and conditional DNA sequence generation, and peptide sequence generation. Our code is available at https://github.com/Aalto-QuML/FlexFlow.

1 Introduction

Generative modeling for biological sequences has progressed rapidly across proteins, RNA, and DNA. Early autoregressive approaches such as ProtGPT2 [Ferruz et al., 2022], ProGen2 [Nijkamp et al., 2023], and DNABERT [Zhou et al., 2023] demonstrated the viability of language model pretraining for sequence generation, while large-scale protein language models like ESM-2,3 [Lin et al., 2023, Hayes et al., 2025] and PoET [Truong Jr and Bepler, 2023, 2025] established powerful representations from evolutionary-scale data. However, autoregressive methods suffer from compounding errors, slow sequential sampling, and limited global coherence, which worsen for long or structurally complex sequences. Discrete diffusion [Austin et al., 2021, Shi et al., 2024, Sahoo et al., 2024] and flow [Gat et al., 2024, Stark et al., 2024, Tang et al., 2025b, Davis et al., 2024] models have emerged as alternatives, enabling fast, non-autoregressive sampling with improved controllability.

Discrete diffusion [Shi et al., 2024, Austin et al., 2021] and flow matching [Lipman et al., 2024] have demonstrated success in DNA sequence design [Stark et al., 2024, Tang et al., 2025b, Huang et al., 2026], protein generation [Alamdari et al., 2023, Kong et al., 2025], and recently, in generating peptides as well [Tang et al., 2025a]. Despite these advances, existing discrete methods for biological applications suffer from key limitations: (i) they offer limited flexibility in flexible length sequence

*Now at OpenProtein.AI

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

generation with limited fine grained control; (ii) they rely on uniform or masked couplings that ignore the structured relationships between biological sequence elements; and (iii) guiding them toward specific functional properties relies on independent, token-wise adjustments [Nisonoff et al., 2024] that fail to capture the global dependencies required for complex regulatory or binding motifs.

Contributions. To address these gaps, we propose **FlexFlow**, a Discrete Flow Matching method for biological sequence design. FlexFlow incorporates a biologically informed structured coupling, an edit-based reverse flow for flexible-length generation with post-hoc operation control, and a latent guidance scheme for fine-grained conditional control. Specifically, our contributions are:

- **Structured Coupling**. We introduce a structured coupling that replaces standard couplings with a domain-specific, biologically informed prior to steer the generative path toward preferred functional neighborhoods.
- **Latent Edit-based rates with controllable edits**. To support variable-length sequence design, we parameterize rates via edit operations and incorporate Dirichlet priors for post-hoc, temperature-controlled operation mixing, enabling granular control during inference.
- **Latent Guidance**. As an alternative to rate-space classifier-free guidance (CFG), we propose latent guidance that operates entirely within the continuous latent space, ensures global structural consistency, and enhances downstream performance.
- **New Benchmark for peptide sequence generation**. We adapt a novel peptide-MHC II benchmark for generating and evaluating MHC-conditioned peptide sequences.
- **Strong Empirical results**. Our method outperforms existing diffusion and flow baselines in unconditional and conditional DNA sequence generation, as well as peptide generation.

2 Background

2.1 Continuous-time Markov Chains

Continuous-time Markov Chains (CTMCs) [Campbell et al., 2024, Holderrieth et al., 2024, Gat et al., 2024, Shaul et al., 2024] are stochastic processes defined over a discrete state space $\mathcal{X}$ that evolve in continuous time, generating trajectories $(X_t)_{t\in[0,1]}$. They are characterized by a rate function u_t which governs the infinitesimal transition probabilities between states

$$\mathbb{P}(X_{t+h} = x | X_t = x_t) = \delta_{x_t}(x) + hu_t(x|x_t) + o(h) \tag{1}$$

Here, $o(h)$ is a higher-order term satisfying $\lim_{h\to 0} \frac{o(h)}{h} = 0n$. Sampling from a CTMC can be performed by iteratively applying the update rule in eq. (1). The rate function $u_t(x|x_t)$ specifies the infinitesimal transition probabilities from the current state x_t to any other state x at time t. For 1 to be valid probability mass function, total probability must be conserved, which imposes the standard rate conditions: $u_t(x|x_t) \geq 0$ for all $x \neq x_t$, and $\sum_x u_t(x|x_t) = 0$. This, in turn, implies that the self-transition rate satisfies $u_t(x_t|x_t) = -\sum_{x\neq x_t} u_t(x|x_t)$.

A rate function u_t induces a probability path p_t if the marginal distribution of the CTMC at each time t coincides with p_t, i.e., $X_t \sim p_t$. In particular, these marginals must satisfy the Kolmogorov forward equation, which states that the rate of change of the probability of a state x equals the total incoming infinitesimal probability from other states minus the total outgoing infinitesimal probability from x, as determined by the transition rates.

2.2 Discrete Flow Matching

Discrete Flow Matching (DFM; Campbell et al. [2024], Gat et al. [2024]) provides a simple framework for learning continuous-time Markov chain (CTMC) based generative models that transport a source distribution $p_0(\mathbf{x})$ (e.g., noise) to a target distribution $p_1(\mathbf{x})$ (e.g., data) over a discrete space $\mathcal{X}$. We consider sequences of fixed length N, so $\mathcal{X} = \mathcal{T}^N$, where $\mathcal{T} = \{1, \ldots, M\}$ is a vocabulary of size M. Training DFM relies on defining a coupling distribution $\pi(\mathbf{x}_0, \mathbf{x}_1)$ that samples $(\mathbf{x}_0, \mathbf{x}_1)$ with marginals p_0 and p_1, along with a conditional CTMC specified by a rate

$$u_t(\mathbf{x}|\mathbf{x}_t, \mathbf{x}_0, \mathbf{x}_1) \text{ generating } p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1), \;\; \text{s.t. } p_0(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta_{\mathbf{x}_0}(\mathbf{x}), \;\; p_1(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta_{\mathbf{x}_1}(\mathbf{x}) \; .$$

where δ being kronecker delta function.Thus, the conditional path interpolates between paired source and target samples, while the learned model follows the corresponding marginal path $p_t(\mathbf{x})$ that interpolates between the target $p_{\text{data}}(\mathbf{x}) := p_1(\mathbf{x})$ and source $p(\mathbf{x}) := p_0(\mathbf{x})$. The associated marginal rate is $u_t(\mathbf{x}|\mathbf{x}_t) = \mathbb{E}_{p_t(\mathbf{x}_0,\mathbf{x}_1|\mathbf{x}_t)} u_t(\mathbf{x}|\mathbf{x}_t, \mathbf{x}_0, \mathbf{x}_1)$, which defines a CTMC that transports p_0 to p_1 by generating the marginal path $p_t(\mathbf{x})$. In practice, models are trained to approximate this rate using objectives such as cross-entropy [Gat et al., 2024, Campbell et al., 2024] and evidence lower bounds [Lou et al., 2023, Sahoo et al., 2024, Shi et al., 2024, Shaul et al., 2024], which can be viewed within the class of Bregman divergences [Holderrieth et al., 2024].

Token-wise mixture paths. The specification of the coupling and conditional path is a design choice. Most prior work adopts a factorized, token-wise conditional path of the following form with the rate

$$p_t(x^i|x_0^i, x_1^i) = (1-\kappa_t)\delta_{x_0^i}(x^i) + \kappa_t \delta_{x_1^i}(x^i),\ u_t(x^i|x_t^i, x_0^i, x_1^i) = \frac{\dot{\kappa}_t}{1-\kappa_t}\left(\delta_{x_1^i}(x^i) - \delta_{x_t^i}(x^i)\right)$$

where κ_t is a scheduler that satisfies $\kappa_0 = 0, \kappa_1 = 1$. In the multi-dimensional setting, transitions are restricted to single-token changes, yielding

$$p_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \prod_{i=1}^{N} p_t(x^i|x_0^i, x_1^i), \quad u_t(\mathbf{x}|\mathbf{x}_t, \mathbf{x}_0, \mathbf{x}_1) = \sum_i \delta_{x_t}(x^{\neg i}) u_t(x^i|x_t^i, x_0^i, x_1^i) \tag{2}$$

where $\delta_{x_t}(x^{\neg i}) = \prod_{k\neq i} \delta_{x_t^k}(x^k)$ indicates that all dimensions except i are held fixed. This factorization enables efficient, parallel sampling while requiring only per-dimension parameterization. However, extending beyond token-wise paths is challenging, as more general conditional CTMCs becomes intractable. For more details on CTMCs and DFM, see Lipman et al. [2024].

2.3 Edit Flows

Edit Flows [Havasi et al., 2025] builds on the DFM framework to define a CTMC-based generative model over variable-length sequences using edit operations while subsuming existing constructions as special cases. The state space is the set of all sequences up to a maximum length N, i.e., $\mathcal{X} = \cup_{n=0}^{N} \mathcal{T}^n$. The model parameterizes a time-dependent rate function u_t^θ where transitions $u_t^\theta(\mathbf{x}|\mathbf{x}_t)$ are non-zero only if $\mathbf{x}$ and $\mathbf{x}_t$ differ by a single edit operation.

Given a sequence $\mathbf{x}$ of variable length $n(\mathbf{x})$, the edit operations are mutually exclusive and constitute the support of $u_t^\theta(\cdot|\mathbf{x}_t)$. The rates are specified as

$$\begin{aligned}
\text{insertion:} \quad & u_t^\theta(\text{ins}(\mathbf{x}, i, a)|\mathbf{x}) = \lambda_{t,i}^{\text{ins}}(\mathbf{x}) Q_{t,i}^{\text{ins}}(a|\mathbf{x}) \\
& \text{with } \text{ins}(\mathbf{x}, i, a) = (x_1, \ldots, x_i, a, x_{i+1}, \ldots, x_{n(\mathbf{x})}), \quad \forall i \in \{1, \ldots, n(\mathbf{x})\} \\
\text{deletion:} \quad & u_t^\theta(\text{del}(\mathbf{x}, i)|\mathbf{x}) = \lambda_{t,i}^{\text{del}}(\mathbf{x}) \\
& \text{with } \text{del}(\mathbf{x}, i) = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n(\mathbf{x})}), \quad \forall i \in \{1, \ldots, n(\mathbf{x})\} \\
\text{substitution:} \quad & u_t^\theta(\text{sub}(\mathbf{x}, i, a)|\mathbf{x}) = \lambda_{t,i}^{\text{sub}}(\mathbf{x}) Q_{t,i}^{\text{sub}}(a|\mathbf{x}), \\
& \text{with } \text{sub}(\mathbf{x}, i, a) = (x_1, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_{n(\mathbf{x})}), \quad \forall i \in \{1, \ldots, n(\mathbf{x})\}
\end{aligned}$$

where $\lambda_{t,i} \geq 0$ denotes operation-specific rates at position i, and $Q_{t,i}$ are normalized distributions over vocabulary for insertion and substitution. This parameterization ensures valid rates, with the self-transition rate given by $u_t^\theta(\mathbf{x}_t|\mathbf{x}_t) = -\sum_{i=1}^{n(\mathbf{x}_t)} \sum_{\text{op}} \lambda_{t,i}^{\text{op}}(\mathbf{x}_t)$, for $\text{op} \in \mathcal{O} = \{\text{ins}, \text{sub}, \text{del}\}$. This formulation enables tractable CTMC modeling directly over sequence space with flexible length changes under the DFM framework. While Edit Flows enables variable-length modeling, it relies on uninformative couplings that ignore the structured relationships between sequence elements and offers no fine-grained control over operations. These limitations carry over to its extensions: EvoFlows [Deutschmann et al., 2026] adapts Edit Flows to protein sequence optimization, where both marginals of the flow are natural sequences rather than noise, and inherits the same parameterization unmodified. We instead encode these relationships through a structured coupling, paired with a hierarchical latent parameterization of rates that exposes operation-level control and a novel latent guidance scheme, which we describe next.

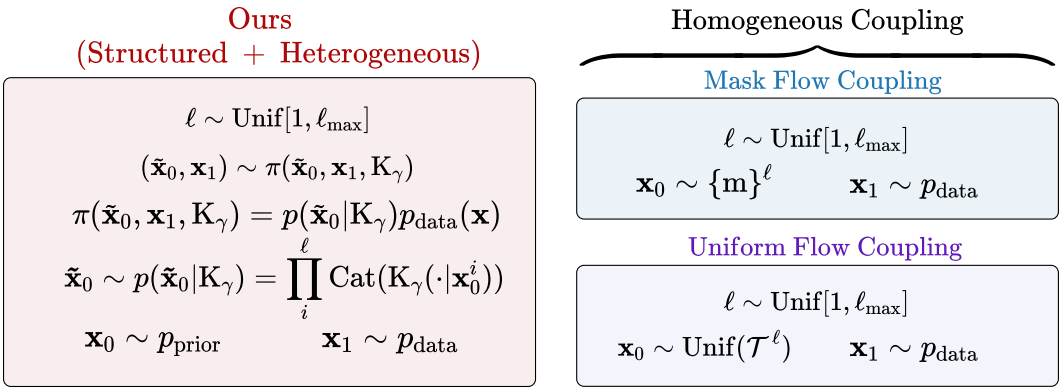


Figure 1: **Comparison of different couplings**. Description of different couplings used in discrete flow matching and comparison with our proposed structured coupling.

3 Method

3.1 Structured Forward Process

We aim to impose a structural inductive bias on the forward noising process by tilting it to reflect biologically preferred substitutions, rather than relying on a non-informative baseline. We consider a discrete space over sequences of length N, so $\mathcal{X} = \mathcal{T}^N$, where $\mathcal{T}$ is the vocabulary of size M containing a discrete set of token values.

Source and target distributions. The forward process operates over discrete token sequences $\mathbf{x} \in \mathcal{T}^N$. Training pairs $(\tilde{\mathbf{x}}_0, \mathbf{x}_1)$ are drawn from a structured coupling $\pi(\tilde{\mathbf{x}}_0, \mathbf{x}_1, \mathrm{K}_\gamma)$ as,

$$(\tilde{\mathbf{x}}_0, \mathbf{x}_1) \sim \pi(\tilde{\mathbf{x}}_0, \mathbf{x}_1, \mathrm{K}_\gamma) = p(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma)p_1(\mathbf{x}_1), \quad p(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma) = \sum_{\mathbf{x}_0} p(\tilde{\mathbf{x}}_0|\mathbf{x}_0, \mathrm{K}_\gamma)p_{\text{prior}}(\mathbf{x}_0) \tag{3}$$

$$p(\tilde{\mathbf{x}}_0|\mathbf{x}_0, \mathrm{K}_\gamma) = \prod_{i=1}^N \text{Cat}\left(\mathrm{K}_\gamma(\cdot|\mathbf{x}_0^i)\right), \quad \mathbf{x}_0 \sim p_{\text{prior}} \tag{4}$$

where K_γ acts as a transition kernel and $\mathrm{K}_\gamma(\cdot \mid \mathbf{x}_0^i) = \mathrm{K}_\gamma \mathbf{e}_{\mathbf{x}_0}^i$ gives the conditional distribution over $\mathcal{T}$ corresponding to token $\mathbf{x}_0^i$, $\mathbf{e}_{\mathbf{x}_0}^i$ denotes one-hot encoding for the i^{th} token, and $\mathbf{x}_1 \sim p_1 := p_{\text{data}}$ is a random sample from the training set. The marginal $p(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma)$ is obtained by first sampling a sequence $\mathbf{x}_0$ from the prior distribution, and then resampling each token independently from the substitution distribution $\mathrm{K}_\gamma \mathbf{e}_{\mathbf{x}_0}^i$ that provides the preferred non-zero substitution probabilities over $\mathcal{T}$.

Here γ indexes a family of domain-specific substitution matrices $\{\mathrm{K}_\gamma \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}\}$, each encoding preferred mutational substitutions for a given modality: K_{prot} for amino acid exchange (e.g., BLOSUM62 [Henikoff and Henikoff, 1992]) and $\mathrm{K}_{\text{DNA,RNA}}$ for base substitution biases (e.g., JC69 [Jukes et al., 1969], HKY85 [Hasegawa et al., 1985]), see App. D for construction details. The coupling generalizes across sequence taxonomies by simply swapping K_γ, leaving the flow and training objective untouched. It also admits a graphical interpretation [Weilbach et al., 2023, Alido et al., 2025], where $\mathrm{K}_\gamma[i, j]$ specifies an edge weight between tokens, inducing a weighted graph whose topology mirrors evolutionary biases. These biases steer the source distribution toward each token's substitution neighborhood rather than the full simplex. With no preference, the coupling reduces to the uniform independent coupling [Campbell et al., 2024, Gat et al., 2024], as stated below.

Proposition 1 *Let* $\mathrm{K}_\gamma = \frac{1}{|\mathcal{T}|}\mathbf{1}\mathbf{1}^\top$, *then* $p(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma) = \text{Uniform}(\mathcal{T}^N)$ *and the coupling reduces to the uniform coupling as,*

$$\pi(\tilde{\mathbf{x}}_0, \mathbf{x}_1) = \text{Uniform}(\mathcal{T}^N)\, p_{\text{data}}(\mathbf{x})$$

The proof is in App. A, and Figure 1 contrasts our coupling with prior choices. Since K_γ is doubly stochastic, the marginal $p(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma)$ stays uniform over $\mathcal{T}$, while the joint $(\mathbf{x}_0, \tilde{\mathbf{x}}_0)$ is correlated according to the substitution preferences in K_γ, recovering the uniform independent coupling when K_γ carries no preference.

Structured token-wise mixture paths. Having defined the mutation-tilted coupling π, we construct a token-wise conditional path [Gat et al., 2024] between the data sample $\mathbf{x}_1 \sim p_{\text{data}}$ and the tilted endpoint $\tilde{\mathbf{x}}_0 \sim q(\cdot|\mathrm{K}_\gamma)$. For each token i, the conditional path and rate are defined as

$$p_t(x^i|\tilde{x}_0^i, x_1^i) = (1-\kappa_t)\delta_{\tilde{x}_0^i}(x^i) + \kappa_t \delta_{x_1^i}(x^i),\ u_t(x^i|x_t^i, \tilde{x}_0^i, x_1^i) = \frac{\dot{\kappa}_t}{1-\kappa_t}\left(\delta_{x_1^i}(x^i) - \delta_{x_t^i}(x^i)\right),$$

where κ_t is a scheduler that satisfies $\kappa_0 = 0, \kappa_1 = 1$. The multi-dimensional case restricts transitions to those that alter exactly one token at a time,

$$p_t(\mathbf{x}|\tilde{\mathbf{x}}_0, \mathbf{x}_1) = \prod_{i=1}^{N} p_t(x^i|\tilde{\mathbf{x}}_0^i, x_1^i), \quad u_t(\mathbf{x}|\mathbf{x}_t, \tilde{\mathbf{x}}_0, \mathbf{x}_1) = \sum_i \delta_{x_t}(x^{\neg i}) u_t(x^i|x_t^i, \tilde{\mathbf{x}}_0^i, x_1^i) \tag{5}$$

where $\delta_{\mathbf{x}_t}(x^{\neg i})$ indicates that all dimensions except i are held fixed, and the marginal rate $u_t(\mathbf{x}|\mathbf{x}_t) = \mathbb{E}_{\pi(\tilde{\mathbf{x}}_0, \mathbf{x}_1)} u_t(\mathbf{x}|\mathbf{x}_t, \tilde{\mathbf{x}}_0, \mathbf{x}_1)$ generates the marginal probability path $p_t(\mathbf{x})$ [Gat et al., 2024, Campbell et al., 2024]. This factorization restricts transitions to token-wise changes, though sampling across positions [Shaul et al., 2024] at cost of an iterative sampling procedure. The mutation bias enters solely through the tilted endpoints via the coupling π, biasing transitions toward biologically preferred substitutions, influencing the path and rates.

3.2 Latent Edit-based rate parameterization

We take inspiration from Havasi et al. [2025] to model the reverse process by parameterizing the rate in terms of edit operations as described in section 2. Rather than directly parameterizing the rate u_t^θ over the full sequence space, we condition it on a global latent representation $\mathbf{r} \in \mathbb{R}^d$ that encodes sequence-level context and per-token edit operations $\{\mathrm{op}^i\} \in \mathcal{O}^N$, where $\mathcal{O} = \{\mathrm{ins}, \mathrm{sub}, \mathrm{del}\}$.

Formally, given the current noisy sequence $\mathbf{x}_t$, we first infer the global latent context via:

$$p_\Psi(\mathbf{r}|\mathbf{x}_t) = \mathcal{N}(\mathbf{r};\ \Psi(\mathbf{x}_t),\ \sigma_r^2 \mathbf{I}), \quad p_\Phi(\mathrm{op}^i|x_t^i, \mathbf{r}) = \mathrm{Cat}(\Phi(x_t^i, \mathbf{r})) \tag{6}$$

where $\Psi(\mathbf{x}_t), \Phi(x_t^i, \mathbf{r})$ are non-linear mappings (e.g., an MLP), and σ_r^2 is the fixed variance. $\Phi(x_t^i, \mathbf{r}) \in \Delta^2$ parameterizes the categorical distribution over edit operations at position i, conditioned on both the local token state and the global latent context. This hierarchical parameterization induces coupling between token-level operations through the shared latent $\mathbf{r}$, while maintaining conditional independence across positions given $\mathbf{r}$.

Because of this conditional independence, the expected rate λ for a specific operation at position i naturally arises by marginalizing over the latent space. Consequently, the full sequence-level marginal rate $u_t^\theta(\mathbf{x}|\mathbf{x}_t)$ follows by marginalizing $\mathbf{z}_t$ and summing over per-token edits:

$$u_t^\theta(\mathbf{x}|\mathbf{x}_t) = \mathbb{E}_{\mathbf{r}\sim p_\Psi(\mathbf{r}|\mathbf{x}_t)}\left[\sum_{i=1}^{N}\sum_{\mathrm{op}\in\mathcal{O}} p_\Phi(\mathrm{op}^i = \mathrm{op}|x_t^i, \mathbf{r})\, u_t^\theta\left(\mathrm{op}(\mathbf{x}_t, i, x^i)|\mathbf{x}_t, \mathbf{r}\right)\right] \tag{7}$$

3.3 Temperature-controlled Operation Mixing via Dirichlet Priors.

We can also induce an explicit inherent control over the operation probabilities in eq. 6 by rescaling the predicted logits $\boldsymbol{\pi}_\Phi = \Phi(x_t^i, \mathbf{r}) \in \Delta^3$ with a position dependent temperature scaling $\boldsymbol{\beta}_i$. This gives the ability to explicitly control the relative rates of insertion, substitution, and deletion operations during generation as follows:

$$p_\Phi(\mathrm{op}^i|x_t^i, \mathbf{r}) = \mathrm{Cat}(\boldsymbol{\pi}_\Phi^i \odot \boldsymbol{\beta}_i), \quad \boldsymbol{\beta}_i = \left[\frac{1}{\tau_{\mathrm{ins}}^i}, \frac{1}{\tau_{\mathrm{sub}}^i}, \frac{1}{\tau_{\mathrm{del}}^i}\right] \in \mathbb{R}^3_{>0}, \quad \sum_{k\in\mathcal{O}} \frac{1}{\tau_k^i} = 1 \tag{8}$$

where $\odot$ denotes element-wise multiplication and $\{\tau_k^i \mid k \in \mathcal{O}\}$ are the per-token per-operation temperature scales. The $\boldsymbol{\beta}_i$ can be modeled via a Dirichlet prior, giving the ability to explicitly control the relative rates of insertion, substitution, and deletion operations during generation as follows:

$$\boldsymbol{\beta}_i = \left[\frac{1}{\tau_{\mathrm{ins}}^i}, \frac{1}{\tau_{\mathrm{sub}}^i}, \frac{1}{\tau_{\mathrm{del}}^i}\right] \sim \mathrm{Dir}(\boldsymbol{\alpha}), \quad \boldsymbol{\alpha} = [\alpha_{\mathrm{ins}}, \alpha_{\mathrm{sub}}, \alpha_{\mathrm{del}}] \in \mathbb{R}^3_{>0} \tag{9}$$

where the concentration parameters $\boldsymbol{\alpha}$ act as inverse temperature coefficients, controlling the relative weight of each operation. The concentration parameters $\boldsymbol{\alpha}$ enable test-time control: $\boldsymbol{\alpha}$ can be set post-hoc to steer generation without retraining by fixing the Dirichlet concentration towards any edit operation, according to the desired operation budget.

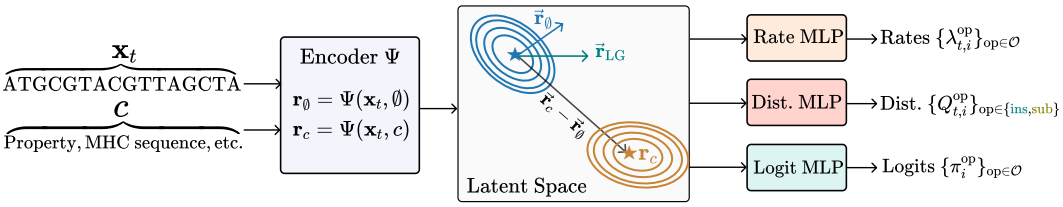


Figure 2: **Model pipeline with Latent Guidance**. Given a noised sequence $\mathbf{x}_t$ with external conditioning c, encoder Ψ produces an unconditional latent $\mathbf{r}_\emptyset$, and a conditional latent $\mathbf{r}_c$, which are interpolated into a guided latent $\mathbf{r}_{\mathrm{LG}}$ using a guidance scale w.This guided latent is then passed into the MLPs to output the edit rates and distributions with auxiliary outputs.

3.4 Classifier-free guidance

We propose two complementary forms of classifier-free guidance for our method, operating at different levels of the model hierarchy: directly on the discrete token rates, and in the continuous global latent space.

3.4.1 Rate-space Guidance

A standard way to incorporate conditioning c is classifier-free guidance (CFG) [Ho and Salimans, 2022], which combines unconditional and conditional model outputs. This can be applied directly in rate space by interpolating between the unconditional and conditional transition rates [Nisonoff et al., 2024, Havasi et al., 2025]. For a guidance strength $w \geq 0$, the guided rate at position i is defined as

$$\tilde{u}_t^\theta(x^i|\mathbf{x}_t, \mathbf{r}, c) \triangleq u_t^\theta(x^i|\mathbf{x}_t, \mathbf{r})^{1-w} u_t^\theta(x^i|\mathbf{x}_t, \mathbf{r}, c)^w = \hat{\lambda}_{t,i}^{\mathrm{op}}(\mathbf{x}_t, \mathbf{r}, c)\tilde{Q}_{t,i}^{\mathrm{op}}(a|\mathbf{x}_t, \mathbf{r}, c) \tag{10}$$

where the geometric interpolation preserves the non-negativity of the rates by construction. Since the rate decomposes into an operation rate and a token distribution, we apply guidance to each component separately via a *naïve rate* CFG as,

$$\hat{\lambda}_{t,i}^{\mathrm{op}}(\mathbf{x}_t, \mathbf{r}, c) = \lambda_{t,i}^{\mathrm{op}}(\mathbf{x}_t, \mathbf{r}, c)^{1+w} \lambda_{t,i}^{\mathrm{op}}(\mathbf{x}_t, \mathbf{r})^{-w} \tag{11}$$

$$\tilde{Q}_{t,i}^{\mathrm{op}}(a|\mathbf{x}_t, \mathbf{r}, c) = Q_{t,i}^{\mathrm{op}}(a|\mathbf{x}_t, \mathbf{r})^{1-w} Q_{t,i}^{\mathrm{op}}(a|\mathbf{x}_t, \mathbf{r}, c)^w \tag{12}$$

where $\hat{\lambda}_{t,i}^{\mathrm{op}}$ scales the operation rate toward the conditioned regime, and $\tilde{Q}_{t,i}^{\mathrm{op}}$ interpolates the token distributions in probability space. However, this guidance operates independently at each token position with no cross-position coherence.

3.4.2 Latent Guidance

A key advantage of our global latent formulation is that it admits a natural, novel form of classifier-free guidance that operates entirely in the continuous latent space, rather than directly on the discrete token rates. Following CFG, during training, we randomly drop the external condition c with probability p_{drop}, training the encoder Ψ jointly to predict both a conditional and unconditional latent

$$\mathbf{r}_{\mathrm{LG}} = \mathbf{r}_\emptyset + w(\mathbf{r}_c - \mathbf{r}_\emptyset), \quad \text{where} \quad \mathbf{r}_c \sim \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_c, \sigma_r^2 \mathrm{I}), \quad \text{and} \quad \mathbf{r}_\emptyset \sim \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_\emptyset, \sigma_r^2 \mathrm{I}), \tag{13}$$

where $\boldsymbol{\mu}_\emptyset = \Psi(\mathbf{x}_t, \emptyset)$, $\boldsymbol{\mu}_c = \Psi(\mathbf{x}_t, c)$, $\emptyset$ denotes the null condition and $w \geq 0$ is the guidance scale. At test time, guidance is applied by interpolating between $\mathbf{r}_c$ and $\mathbf{r}_\emptyset$ in the continuous latent space. The guided latent $\mathbf{r}_{\mathrm{LG}}$ then conditions all per-token operations jointly via $p_\Phi(\mathrm{op}^i|x_t^i, \mathbf{r}_{\mathrm{LG}})$, propagating the guidance signal globally through the shared r, in contrast to rate-space guidance, which steers each position independently. Since it operates in the latent space, non-negativity of the rates is preserved by construction without requiring renormalization. The guided rate is then,

$$\tilde{u}_t^\theta(\mathbf{x}|\mathbf{x}_t, c) = \mathbb{E}_{\mathbf{r}_{\mathrm{LG}} \sim p_\Psi(\mathbf{r}_{\mathrm{LG}}|\mathbf{x}_t, c)} \left[\sum_{i=1}^{N} \sum_{\mathrm{op}\in\mathcal{O}} p_\Phi(\mathrm{op}^i = \mathrm{op}|x_t^i, \mathbf{r}_{\mathrm{LG}})\, u_t^\theta\left(\mathrm{op}(\mathbf{x}, i)|\mathbf{x}_t, \mathbf{r}_{\mathrm{LG}}\right) \right]$$

This latent interpolation admits a principled probabilistic interpretation: the guidance shift corresponds to the score of an implicit latent classifier, as formalized below.

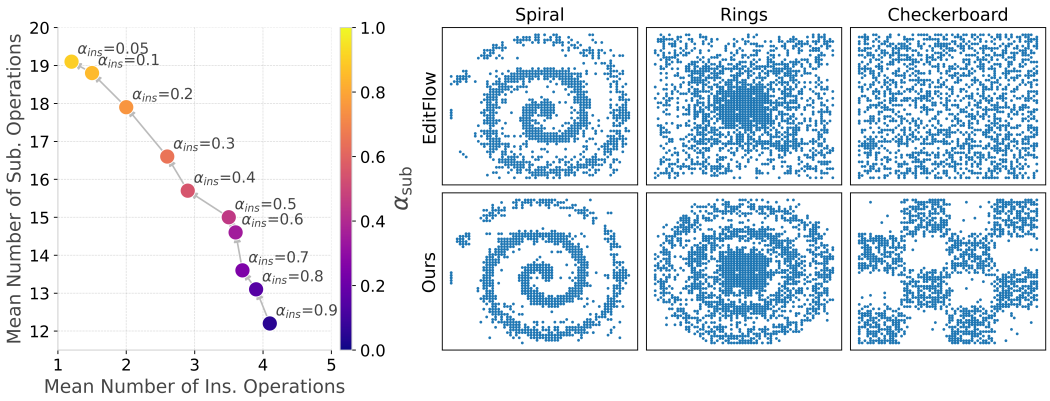


Figure 3: (***left***) Evolution of edit operations during generation as the Dirichlet concentration $\boldsymbol{\alpha}$ is shifted between insertion and substitution vertices (with fixed $\alpha_{\text{del}} = 0.05$), showing model's ability to act as a global budget controller, (***right***) Representative generated distributions on toy datasets.

Proposition 2 (Latent Guidance induces an implicit classifier) *Let* $\mathbf{r}$ *be a sufficient statistic for* c*, with unconditional and conditional latents* $\mathbf{r}_\emptyset \sim p_\Psi(\mathbf{r}|\mathbf{x}_t) = \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_\emptyset, \sigma_r^2 \mathrm{I})$ *and* $\mathbf{r}_c \sim p_\Psi(\mathbf{r}|\mathbf{x}_t, c) = \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_c, \sigma_r^2 \mathrm{I})$*. Then the guidance direction is proportional to the score of an implicit classifier* $p(c|\mathbf{r})$*, and* $\mathbf{r}_{\text{LG}}$ *amounts to a gradient ascent step on* $\log p(c|\mathbf{r})$ *in latent space:*

$$\mathbf{r}_{\text{LG}} \sim \mathcal{N}(\mathbf{r}_{\text{LG}}; \boldsymbol{\mu}_\emptyset + w\sigma_r^2 \nabla_{\mathbf{r}} \log p(c \mid \mathbf{r})\big|_{\mathbf{r}=\boldsymbol{\mu}_\emptyset}, (1-w)^2\sigma_r^2 + w^2\sigma_r^2 \mathrm{I})$$

3.5 Training Objective

We adopt a training strategy similar to Edit Flows [Havasi et al., 2025], operating in an augmented space to avoid the intractability of standard cross-entropy or ELBO objectives. Since edit sequences (insertions, deletions, substitutions) induce multiple paths between the same strings, direct transition rates are difficult to compute. Following Havasi et al. [2025], Holderrieth et al. [2024], we define the process over an augmented space $\mathcal{X} \times \mathcal{Z}$, enabling a tractable objective via Bregman divergences.

Auxiliary alignment process. Given a pair $(\tilde{\mathbf{x}}_0, \mathbf{x}_1)$ an alignment defines a precise set of edit operations transforming $\tilde{\mathbf{x}}_0$ to $\mathbf{x}_1$ by introducing a blank token $\varepsilon \notin \mathcal{T}$. We define the augmented space $\mathcal{Z} = \mathcal{T} \cup \{\varepsilon\}$, with $f_{\text{rm}-\text{blanks}} : \mathcal{Z} \to \mathcal{X}$ stripping all ε tokens. This defines a structured coupling in the augmented space $\pi(\tilde{\mathbf{z}}_0, \mathbf{z}_1, \mathrm{K}_\gamma)$ satisfying the marginal distributions of the original distributions $p(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma)$ and $p_1(\mathbf{x}_1)$. Then, the conditional probability path and the rate is given as,

$$p_t(\mathbf{x}, \mathbf{z}|\tilde{\mathbf{x}}_0, \tilde{\mathbf{z}}_0, \mathbf{x}_1, \mathbf{z}_1) = p_t(\mathbf{x}, \mathbf{z}|\tilde{\mathbf{z}}_0, \mathbf{z}_1) = p_t(\mathbf{z}|\tilde{\mathbf{z}}_0, \mathbf{z}_1)\delta_{f_{\text{rm-blanks}}(\mathbf{z})}(\mathbf{x}) \tag{14}$$

$$u_t(\mathbf{x}, \mathbf{z}|\mathbf{x}_t, \mathbf{z}_t, \tilde{\mathbf{z}}_0, \mathbf{z}_1) = \delta_{f_{\text{rm-blanks}}(\mathbf{z})}(\mathbf{x}) \sum_{i=1}^{N} \frac{\dot{\kappa}_t}{1-\kappa_t} \left(\delta_{z_1^i}(z^i) - \delta_{z_t^i}(z^i)\right) \delta_{\mathbf{z}_t}(z^{\neg i}) \tag{15}$$

Learning Objective. We learn the parameterized rate u_t^θ by minimizing a Bregman divergence [Bregman, 1967] between the true and predicted rates over the augmented space, which simplifies to the following objective,

$$\min_\theta \mathcal{L}(\theta) = \mathbb{E}_{t, \mathbf{x}_t, \mathbf{z}_t}\left[\sum_{\mathbf{x} \neq \mathbf{x}_t} u_t^\theta(\mathbf{x}|\mathbf{x}_t) - \sum_{i=1}^{N} \mathbb{1}_{[z_1^i \neq z_t^i]} \frac{\dot{\kappa}_t}{1-\kappa_t} \log u_t^\theta(\mathbf{x}(\mathbf{z}_t, i, z_1^i)|\mathbf{x}_t)\right]$$
$$\text{with } (\mathbf{x}_t, \mathbf{z}_t) \sim p_t(\mathbf{x}_t, \mathbf{z}_t|\tilde{\mathbf{z}}_0, \mathbf{z}_1), (\tilde{\mathbf{z}}_0, \mathbf{z}_1) \sim \pi(\tilde{\mathbf{z}}_0, \mathbf{z}_1, \mathrm{K}_\gamma), \text{ and } t \sim \mathcal{U}(0,1)$$

where $u_t^\theta(\mathbf{x}|\mathbf{x}_t) = \sum_z \mathbb{E}_{p_t(\tilde{\mathbf{z}}_0, \mathbf{z}_1, \mathbf{z}_t|\mathbf{x}_t)} u_t(\mathbf{x}, \mathbf{z}|\mathbf{x}_t, \mathbf{z}_t, \tilde{\mathbf{z}}_0, \mathbf{z}_1)$ is the marginalized output rate of the model, and $\mathbf{x}(\mathbf{z}_t, i, z_1^i) = f_{\text{rm-blanks}}(z_t^1, \ldots, z_t^{i-1}, z_t^i, z_t^{i+1}, \ldots, z_t^N)$ represents a specific edit operation in the sequence space. This loss can be interpreted as minimizing all extraneous output rates of the model while performing the cross-entropy over the edit operations that move $\mathbf{x}_t$ closer to $\mathbf{x}_1$.

Table 1: (***left, top***) **Unconditional Enhancer Generation**: Distributional similarity measured via FBD and sampling efficiency (NFE) for $10,000$ generated DNA sequences; † denotes metrics taken from Stark et al. [2024]. (***left, bottom***) **Ablation study** on prior choices and structured coupling with respective FBD values over Melanoma and FlyBrain data. (***right***) **Conditional Promoter Design**: Conditioned on a transcription profile, each method is tasked to generate a DNA sequence with that profile. MSE measures the difference between predicted and ground truth regulatory activities.

Method	Melanoma	Fly Brain	NFE
	Fréchet Bio. Dist. (↓)		
Random Sequence	622.8	876	–
Language Model†	36.0	25.2	500
Linear FM†	19.6	15.0	100
Dirichlet FM†	5.3	15.2	100
Fisher FM	3.8	27.5	100
EditFlow	3.7	7.9	100
FlexFlow	**3.4**	**4.2**	100

p_{unif}	p_{freq}	K_γ	MEL	FB
✓			3.7	5.2
	✓		3.5	4.7
	✓	✓	3.4	4.2

Method	MSE	NFE
Bit Diffusion† (Bit Encoding)	0.041	100
Bit Diffusion†(One-hot Encoding)	0.039	100
D3PM-Uniform†	0.037	100
DDSM†	0.033	100
Language Model†	0.033	1024
Linear FM†	0.028	100
Dirichlet FM†	0.026	100
Dirichlet FDM	0.026	(N/A)
Fisher FM	0.030	(N/A)
Gumbel-Softmax FM	0.029	(N/A)
FlexFlow - Rate Guidance	0.024	100
FlexFlow - Latent Guidance w/o K_γ	0.025	100
FlexFlow - Latent Guidance	**0.022**	100

4 Experiments

Tasks. We evaluate our method on diverse tasks. In Section 4.1, we evaluate our method ability to model various densities. In Section 4.2 and Section 4.3, we assess the methods ability to generate conditional and unconditional DNA sequences, and finally, in section 4.4, we show the performance of our method in generating peptide sequences conditioned on MHC.

Implementation. Our implementation uses PyTorch, following the architecture and pre-processing steps of the baselines. See App. C for further technical details.

Baselines. We compare against a range of diffusion and flow-based methods, including Bit Diffusion, DDSM [Albergo and Vanden-Eijnden, 2022], D3PM [Austin et al., 2021], Linear FM [Lipman et al., 2024], Dirichlet FM [Stark et al., 2024], Dirichlet FDM [Huang et al., 2026], Gumbel-Softmax FM [Tang et al., 2025b], and Fisher FM [Davis et al., 2024]. We cannot compare to EvoFlows [Deutschmann et al., 2026] as there is no code available for it.

4.1 Toy datasets

We first evaluate our method on toy datasets to assess its ability to learn discrete distributional patterns in 1D and 2D, using four canonical patterns: (i) Sine-wave, a 1D sinusoidal pattern testing periodic structure; (ii) Spiral, a 2D curved pattern testing long-range dependencies; (iii) Rings, concentric rings testing multi-modal symmetry; and (iv) Checkerboard, a 2D alternating pattern testing sharp local transitions. We measure distribution quality via the Wasserstein distance between generated and true samples. Figure 3 and Table 2 describes that our method consistently achieves the lowest $\mathcal{W}_2$ across all patterns and accurately reconstructs each distribution's topological features.

4.2 Enhancer DNA Sequence Design

Data. We evaluate on two enhancer sequence datasets derived from fly brain cells [Janssens et al., 2022] and human melanoma cells [Atak et al., 2021]. These consist of 104k and 89k sequences of length 500 with ATAC-seq measurements [Buenrostro et al., 2013] across 81 and 47 classes.

Metrics and Superior performance in unconditional design. To assess the similarity between the data and model distributions, we use the Fréchet Biological Distance (FBD) introduced by Stark et al. [2024]. This involves training a classifier to predict cell types and using its hidden representations as embeddings to compute Wasserstein distance for both generated and real samples. Table 1 shows the results on both enhancer datasets. Notably, our method is able to achieve the lowest FBD as compared to the baselines, demonstrating the enhanced fidelity afforded by our method.

Table 2: (***left***) **Evaluation on test dataset for MHC Peptide Sequence Design**: FPD, DeepMHCII classifier score values, and NFE on $10,000$ generated peptide sequence for held out test set $\mathcal{D}_{\text{Test}}$ of MHC sequences. (***right***) **Wasserstein distance $\mathcal{W}_2$ between true and generated distribution.**

Method	FPD ↓	DeepMHCII ↑	Uniq.
Dirichlet FM	0.79	0.21	100
Dirichlet FDM	0.72	0.31	100
Fisher FM	0.75	0.28	100
FlexFlow - Rate Guidance	**0.58**	0.58	100
FlexFlow - Latent Guidance	0.84	**0.66**	100

Data	D3PM	EditFlow	**FlexFlow**
Sine wave	0.033	0.031	**0.024**
Spiral	0.056	0.045	**0.042**
Rings	0.085	0.080	**0.040**
Checkerboard	0.067	0.070	**0.041**

4.3 Promoter DNA Sequence Design

Data. We next evaluate the conditional design of DNA promoter sequences given a desired promoter profile, following the setup of Dirichlet FM [Stark et al., 2024] and DDSM [Avdeyev et al., 2023]. We use 100,000 human promoter sequences of 1024 base pairs from Hon et al. [2017], each annotated with a FANTOM5 CAGE signal [Shiraki et al., 2003, Consortium et al., 2014] representing per-position transcription initiation probability ($\mathbf{r} \in \mathbb{R}^{1024}$). Chromosomes 8 and 9 are held out for testing; the rest is used for training.

Metrics and Superior performance in conditional design. Following Stark et al. [2024], we evaluate generated sequences using the MSE between their predicted regulatory activity and that of the corresponding original sequence. Regulatory activity is estimated using the promoter-specific outputs of SEI model [Chen et al., 2022]. We report the regulatory activity prediction results in Table 1. Our method significantly outperforms both Dirichlet FM (0.026) and DDSM (0.033), achieving a lower mean squared error (MSE) across the test set. While our model using Rate Guidance already surpasses current flows (0.024), Latent Guidance provides the best performance.

4.4 MHC Peptide Sequence Design

We consider a conditional generation task where, given an MHC allele m, the goal is to generate a short peptide $\mathbf{x}$ (typically of length 13–18 amino acids) that is naturally processed and presented by the given MHC molecule. The dataset thus consists of pairs $\{(m_i, \mathbf{x}_i)\}_{i=1}^{N}$ (see Appendix B).

Data. Using Eluted Ligand (EL) data from Nilsson et al. [2023], we adapted a peptide–MHC II benchmark [Korpela et al., 2026], using peptides physically presented on MHC molecules, collected via mass spectrometry. The data is partitioned into $\mathcal{D}_{\text{Train}}$ and $\mathcal{D}_{\text{Test}}$ using a strict sequence-based split ensuring no 9-mer subsequence is shared. For more details see App. B

Metrics. We evaluate peptide generation with two complementary metrics. (i) **Discriminator score** $(\in [0, 1])$: a held-out DeepMHCII model [You et al., 2022] retrained on EL data (Appendix C.2.1) to separate real ligands from synthetic negatives, with higher scores indicating closer resemblance to true ligands. (ii) **Fréchet Protein Distance** (FPD): the Wasserstein distance between ESM2 [Lin et al., 2023] embeddings of generated and test peptides, where lower is better. We treat the discriminator score as our primary metric, since it directly rewards presentation plausibility in a setting where only positive ligands are observed; FPD serves as a complementary check on distributional fidelity.

Superior results in peptide sequence generation. Our model substantially outperforms all baselines on DeepMHCII score — Dirichlet FM (0.21), Dirichlet FDM (0.31), and Fisher FM (0.28) — reaching 0.58 with rate-space guidance and 0.66 with latent-space guidance (Table 2). Rate-space guidance also improves FPD over all baselines, while latent-space guidance prioritizes binding plausibility at the cost of distributional coverage — a tradeoff we ablate in the next section.

5 Ablation Studies

Scalability of the Global Latent Representation. We conducted an ablation study to assess how the latent encoder scales with context length by measuring its peak GPU memory and forward-pass latency. Table 3 (left) shows that both remain nearly flat up to 1k residues and grow only modestly up to 10k. This follows from the dilated CNN encoder, which scales linearly in sequence length and captures long-range context without the quadratic cost of self-attention, making it well suited to longer genomic contexts.

Table 3: **Ablation studies.** (***left***) Encoder scalability with context length, (***center***) effect of σ_r on downstream plausibility, and (***right***) biological viability under forced edit budgets.

Length	Latency (ms)	Peak Memory (GB)
100	3.14	0.093
500	3.46	0.095
1024	3.61	0.098
2000	6.21	0.108
10000	24.31	0.614

σ_r	DeepMHCII (↑)
0.0	0.62
0.1	0.66
0.5	0.64

α_{ins}	α_{sub}	α_{del}	FPD	DeepMHCII
0.9	0.05	0.05	1.05	0.22
0.05	0.9	0.05	0.58	0.58
0.05	0.05	0.9	0.98	0.22

Biological Viability of Forced Edit Budgets. We conducted an ablation study to evaluate the biological plausibility of sequences generated under skewed edit budgets, varying the Dirichlet concentration parameters $\boldsymbol{\alpha}$ on the peptide sequence design dataset. Table 3 (right) shows that substitution-biased budgets, which more closely reflect the typical substitution-heavy setting, yield the best FPD and DeepMHCII scores, while insertion or deletion biases lead to lower performance.

Effect of σ_r. We conducted an ablation study varying σ_r and evaluating downstream biological plausibility via the DeepMHCII score. Table 3 (middle) shows that performance is fairly stable across values, with a dip at $\sigma_r = 0$, indicating that a non-zero σ_r improves downstream performance.

Post-hoc Operation Control. We conducted an ablation study to show the post-hoc operation control over the edit operations. Figure 3 illustrates that shifting concentration parameters $\boldsymbol{\alpha}$ toward specific simplex vertices prioritizes corresponding edit operations. This allows $\boldsymbol{\alpha}$ to act as a budget controller, tuning operation frequencies during inference to meet specific requirements without retraining.

Time complexity. We compare inference speed against prior methods by measuring the time to generate a 500-length DNA sequence over 100 reverse timesteps. Table 4 in the appendix shows that our method achieves a comparable inference speed compared to the baselines.

Choice of prior and the effect of structured coupling. We conduct an ablation study to evaluate the impact of the prior and the structured coupling on the fidelity of generated DNA sequences. Table 1 shows that the frequency-informed prior p_{freq} with structured coupling K_γ gives peak performance by aligning the generative path with domain-specific biases.

Guidance Choice: Effects of Scale and p_{drop} on Diversity – Quality trade-off. We conducted an ablation study to characterize the impact of different guidance strategies and the interplay between the guidance scale w and p_{drop}. Tables 1 and 2 compare various formulations, while Figure 4 shows that higher p_{drop} needs a larger w to maintain fidelity, shifting the optimal performance point accordingly.

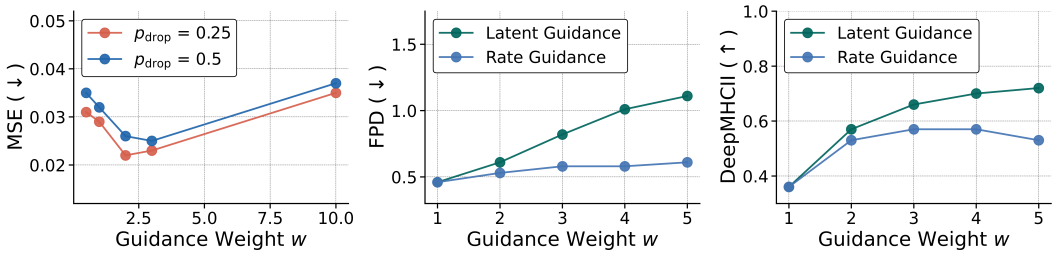


Figure 4: **Diversity vs Quality tradeoff and effect of p_{drop}.** (*left*) Impact of p_{drop} on promoter DNA sequence design. (*center, right*) At high guidance scales, LG produces higher-quality peptide ligands than RG at the cost of increased FPD, illustrating a diversity-quality tradeoff.

6 Conclusion

We introduce **FlexFlow**, a Discrete Flow Matching framework for biological sequence design that addresses key limitations of existing discrete generative methods through a biologically informed structured coupling, an edit-based reverse flow parameterization with Dirichlet-controlled operation mixing for flexible-length generation, and a latent guidance scheme that preserves global structural consistency. Across unconditional and conditional DNA generation, as well as peptide design on our newly introduced peptide-MHC II benchmark, FlexFlow consistently outperforms strong diffusion and flow-matching baselines. Future work includes adapting distillation-based training objectives to FlexFlow, enabling high-quality sequence generation in just a handful of sampling steps.

Acknowledgements

YV and VG acknowledge support from the Research Council of Finland for the "Human-steered next-generation machine learning for reviving drug design" project (grant decision 342077). YV thanks Suomen Tekniikan edistämissäätiö (grant number 10477) for their support. VG also acknowledges support from the Jane and Aatos Erkko Foundation (grant 7001703) for "Biodesign: Use of artificial intelligence in enzyme design for synthetic biology". We acknowledge generous computational support from the Aalto-IT Science project and ELLIS Institute Finland. YV thanks Priscilla Ong for fruitful discussions during the initial ideation phase and the CNY retreat in Singapore.

References

Sarah Alamdari, Nitya Thakkar, Rianne Van Den Berg, Neil Tenenholtz, Robert Strome, Alan M Moses, Alex X Lu, Nicolò Fusi, Ava P Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *BioRxiv*, pages 2023–09, 2023.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Jeffrey Alido, Tongyu Li, Yu Sun, and Lei Tian. Whitened score diffusion: A structured prior for imaging inverse problems. *arXiv preprint arXiv:2505.10311*, 2025.

Zeynep Kalender Atak, Ibrahim Ihsan Taskiran, Jonas Demeulemeester, Christopher Flerin, David Mauduit, Liesbeth Minnoye, Gert Hulselmans, Valerie Christiaens, Ghanem-Elias Ghanem, Jasper Wouters, et al. Interpretation of allele-specific chromatin accessibility using cell state–aware deep learning. *Genome research*, 31(6):1082–1096, 2021.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

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*, pages 1276–1301. PMLR, 2023.

Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

Jason D Buenrostro, Paul G Giresi, Lisa C Zaba, Howard Y Chang, and William J Greenleaf. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, dna-binding proteins and nucleosome position. *Nature methods*, 10(12):1213–1218, 2013.

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.

Kathleen M Chen, Aaron K Wong, Olga G Troyanskaya, and Jian Zhou. A sequence-based global map of regulatory activity for deciphering human genetics. *Nature genetics*, 54(7):940–949, 2022.

The FANTOM Consortium, the RIKEN PMI, and CLST (DGT). A promoter-level mammalian expression atlas. *Nature*, 507(7493):462–470, 2014.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İ Ceylan, Michael Bronstein, and Avishek J Bose. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 37:139054–139084, 2024.

Nicolas Deutschmann, Constance Ferragu, Jonathan D Ziegler, Shayan Aziznejad, and Eli Bixby. Evoflows: Evolutionary edit-based flow-matching for protein engineering. *arXiv preprint arXiv:2603.11703*, 2026.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

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.

Masami Hasegawa, Hirohisa Kishino, and Taka-aki Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of molecular evolution*, 22(2):160–174, 1985.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025.

Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the national academy of sciences*, 89(22):10915–10919, 1992.

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

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. *arXiv preprint arXiv:2410.20587*, 2024.

Chung-Chau Hon, Jordan A Ramilowski, Jayson Harshbarger, Nicolas Bertin, Owen JL Rackham, Julian Gough, Elena Denisenko, Sebastian Schmeier, Thomas M Poulsen, Jessica Severin, et al. An atlas of human long non-coding rnas with accurate 5' ends. *Nature*, 543(7644):199–204, 2017.

Yuhao Huang, Taos Transue, Shih-Hsin Wang, William Feldman, Hong Zhang, and Bao Wang. Improving flow matching by aligning flow divergence. *arXiv preprint arXiv:2602.00869*, 2026.

Jasper Janssens, Sara Aibar, Ibrahim Ihsan Taskiran, Joy N Ismail, Alicia Estacio Gomez, Gabriel Aughey, Katina I Spanier, Florian V De Rop, Carmen Bravo Gonzalez-Blas, Marc Dionne, et al. Decoding gene regulation in the fly brain. *Nature*, 601(7894):630–636, 2022.

Thomas H Jukes, Charles R Cantor, et al. Evolution of protein molecules. *Mammalian protein metabolism*, 3(21):132, 1969.

Zitai Kong, Yiheng Zhu, Yinlong Xu, Hanjing Zhou, Mingzhe Yin, Jialu Wu, Hongxia Xu, Chang-Yu Hsieh, Tingjun Hou, and Jian Wu. Protflow: fast protein sequence design via flow matching on compressed protein language model embeddings. *arXiv preprint arXiv:2504.10983*, 2025.

Dani Korpela, Alexandru Dumitrescu, Martin Stravzar, Rui Li, Ramnik J Xavier, Daniel B Graham, and Harri Lähdesmäki. Pepgen: conditional generation of peptides for mhc binding. *Bioinformatics*, 42(Supplement_2):btag464, 2026.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

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.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.

Jonas B. Nilsson, Saghar Kaabinejadian, Hooman Yari, Michel G. D. Kester, Peter van Balen, William H. Hildebrand, and Morten Nielsen. Accurate prediction of HLA class II antigen presentation across all loci using tailored data acquisition and refined machine learning. *Science Advances*, 9(47), November 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.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky TQ Chen. Flow matching with general discrete paths: A kinetic-optimal perspective. *arXiv preprint arXiv:2412.03487*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

Toshiyuki Shiraki, Shinji Kondo, Shintaro Katayama, Kazunori Waki, Takeya Kasukawa, Hideya Kawaji, Rimantas Kodzius, Akira Watahiki, Mari Nakamura, Takahiro Arakawa, et al. Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage. *Proceedings of the National Academy of Sciences*, 100(26):15776–15781, 2003.

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.

Sophia Tang, Yinuo Zhang, and Pranam Chatterjee. Peptune: De novo generation of therapeutic peptides with multi-objective-guided discrete diffusion. In *Frontiers in Probabilistic Inference: Learning meets Sampling*, 2025a. URL https://openreview.net/forum?id=eBoJ9YRx0w.

Sophia Tang, Yinuo Zhang, Alexander Tong, and Pranam Chatterjee. Gumbel-softmax flow matching with straight-through guidance for controllable biological sequence generation. *ArXiv*, pages arXiv–2503, 2025b.

Timothy Truong Jr and Tristan Bepler. Poet: A generative model of protein families as sequences-of-sequences. *Advances in Neural Information Processing Systems*, 36:77379–77415, 2023.

Timothy Fei Truong Jr and Tristan Bepler. Understanding protein function with a multimodal retrieval-augmented foundation model. *arXiv preprint arXiv:2508.04724*, 2025.

Yogesh Verma, Markus Heinonen, and Vikas Garg. Abode: Ab initio antibody design using conjoined odes. In *International Conference on Machine Learning*, pages 35037–35050. PMLR, 2023.

Christian Dietrich Weilbach, William Harvey, and Frank Wood. Graphically structured diffusion models. In *International Conference on Machine Learning*, pages 36887–36909. PMLR, 2023.

Ronghui You, Wei Qu, Hiroshi Mamitsuka, and Shanfeng Zhu. DeepMHCII: A novel binding core-aware deep interaction model for accurate MHC-II peptide binding affinity prediction. *Bioinformatics*, 38(Supplement_1):i220–i228, June 2022. ISSN 1367-4803.

Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

A Proofs

A.1 Proof of proposition 1

For any $\mathbf{x}_0^i \in \mathbf{x}_0 \in \mathcal{T}$,

$$\mathrm{K}_\gamma(\tilde{\mathbf{x}}_0^i|\mathbf{x}_0^i) = \mathrm{K}_\gamma \mathbf{e}_{\mathbf{x}_0}^i = \frac{1}{|\mathcal{T}|}\mathbf{1}\mathbf{1}^\top \mathbf{e}_{\mathbf{x}_0}^i = \frac{1}{|\mathcal{T}|}\mathbf{1} \tag{16}$$

providing,

$$\mathrm{Cat}(\tilde{x}_0^i; \mathrm{K}_\gamma \mathbf{e}_{\mathbf{x}_0}^i) = \mathrm{Uniform}(\mathcal{T}) \tag{17}$$

regardless of $\mathbf{x}_0^i$. The marginalization over $\mathbf{x}_0$ trivially factors out, giving

$$q(\tilde{\mathbf{x}}_0|\mathrm{K}_\gamma) = \mathrm{Uniform}(\mathcal{T}^N) \tag{18}$$

providing the coupling

$$\pi(\tilde{\mathbf{x}}_0, \mathbf{x}_1) = \mathrm{Uniform}(\mathcal{T}^N)\, p_{\mathrm{data}}(\mathbf{x}_1) \tag{19}$$

A.2 Proof of proposition 2

Let $p(\mathbf{r}|\mathbf{x}_t) = \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_\emptyset, \sigma_r^2 \mathrm{I})$, and $p(\mathbf{r}|\mathbf{x}_t, c) = \mathcal{N}(\mathbf{r}; \boldsymbol{\mu}_c, \sigma_r^2 \mathrm{I})$. Then,

$$\mathbf{r}_{\mathrm{LG}} = \mathbf{r}_\emptyset + w(\mathbf{r}_c - \mathbf{r}_\emptyset) \sim \mathcal{N}(\boldsymbol{\mu}_\emptyset + w(\boldsymbol{\mu}_c - \boldsymbol{\mu}_\emptyset), (1-w)^2\sigma_r^2 + w^2\sigma_r^2 \mathrm{I}) \tag{20}$$

By utilizing Bayes' theorem and assuming $\mathbf{r}$ is a sufficient statistic, we get,

$$p(\mathbf{r} \mid \mathbf{x}_t, c) \propto p(c \mid \mathbf{r}) p(\mathbf{r} \mid \mathbf{x}_t) \tag{21}$$

$$\nabla_{\mathbf{r}} \log p(\mathbf{r} \mid \mathbf{x}_t, c) = \nabla_{\mathbf{r}} \log p(\mathbf{r} \mid \mathbf{x}_t) + \nabla_{\mathbf{r}} \log p(c \mid \mathbf{r}) \tag{22}$$

$$\frac{\boldsymbol{\mu}_c - \mathbf{r}}{\sigma_r^2} = \frac{\boldsymbol{\mu}_\emptyset - \mathbf{r}}{\sigma_r^2} + \nabla_{\mathbf{r}} \log p(c \mid \mathbf{r}) \tag{23}$$

$$\boldsymbol{\mu}_c - \boldsymbol{\mu}_\emptyset = \sigma_r^2 \nabla_{\mathbf{r}} \log p(c \mid \mathbf{r}) \tag{24}$$

This provides the latent guidance as,

$$\mathbf{r}_{\mathrm{LG}} \sim \mathcal{N}(\boldsymbol{\mu}_\emptyset + w\sigma_r^2 \nabla_{\mathbf{r}} \log p(c \mid \mathbf{r})\big|_{\mathbf{r}=\boldsymbol{\mu}_\emptyset}, (1-w)^2\sigma_r^2 + w^2\sigma_r^2 \mathrm{I}) \tag{25}$$

B Dataset Construction

MHC-II molecules are heterodimeric proteins composed of an α- and a β-chain, with binding specificity jointly determined by both chains. Rather than encoding full protein sequences, ML pipelines typically represent each MHC-II molecule via a *pseudosequence* — a 34-residue subset of amino acid positions most critical for peptide binding interactions Nilsson et al. [2023]. The MHC-II pseudosequences (conditionings) are used as input features for generation of their corresponding peptide sequences (targets) throughout our pipeline.

The dataset comprises peptides distributed across 56 MHC-II allele combinations (α/β chain pairs) from the three canonical allele groups of HLA class II: DR, DP, and DQ. The dataset exhibits significant class imbalance, largely reflecting differences in experimental data availability across alleles, with per-allele combination counts ranging from 1,518 (`DRA*01:01_DRB5*01:01`) to 13,892 (`DRA*01:01_DRB1*01:01`) peptides across the combined train and test sets. The per-allele distribution across splits is shown fig. 5.

We used the eluted ligand (EL) subset from Nilsson et al. [2023] and removed all non-standard amino acids (e.g. selenocysteine). The original work splits data per allele, ensuring no shared 9-mer among peptides of the same allele within a fold. We applied a stricter criterion: all peptide sequences were clustered jointly across alleles such that no 9-mer is shared between any two clusters. Clusters were then joined to form two balanced partitions, $\mathcal{D}_{\mathrm{Train}}$ and $\mathcal{D}_{\mathrm{Test}}$, each further divided into training and validation subsets as in Korpela et al. [2026]:

$$\mathcal{D}_{\mathrm{Train}} = \mathcal{D}_0 \cup \mathcal{D}_1, \qquad \mathcal{D}_{\mathrm{Test}} = \mathcal{D}_2 \cup \mathcal{D}_3.$$

The generative model was trained on $\mathcal{D}_0$ and validated on $\mathcal{D}_1$. The test discriminator was trained on $\mathcal{D}_2$ and validated on $\mathcal{D}_3$ (see Appendix C.2.1). The synthetic negatives from Nilsson et al. [2023] were used only for discriminator training and not for generative model training. We evaluated the FPD between $\mathcal{D}_1$ and $\mathcal{D}_3$ and between $\mathcal{D}_3$ and corresponding artificial negatives for reference yielding 0.30 and 1.98, respectively.

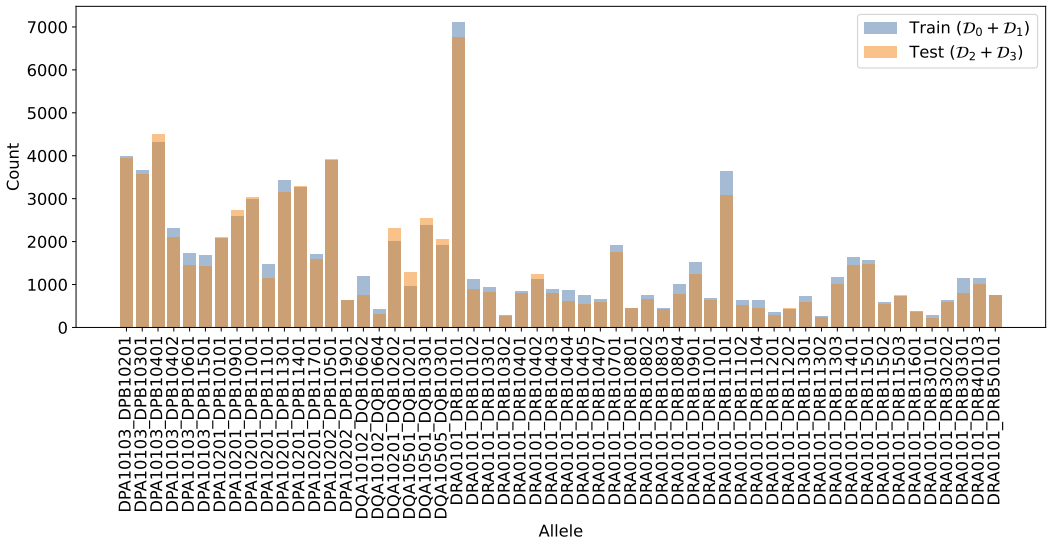


Figure 5: Allele distribution of train and test sets for the peptide MHC binding task

C Implementation Details

Table 4: Inference speed and Trainable Parameters for DNA sequence design.

Method	Model size	Trainable params	Inference time (sec.)
Dirichlet FM	$\sim$ 5M	$\sim$ 5M	$\sim$ 3.8
Fisher FM	$\sim$ 5M	$\sim$ 5M	$\sim$ 4.5
Dirichlet FDM	$\sim$ 5M	$\sim$ 5M	$\sim$ 4.3
FlexFlow	$\sim$ 5M	$\sim$ 5M	$\sim$ 4.0

C.1 DNA Sequence Design

The model was trained on a single NVIDIA V100 GPU.

C.2 MHC Peptide Sequence Design

The model was trained with a ddp strategy on four NVIDIA V100 GPUs.

C.2.1 Test Discriminator Training

We adapted DeepMHCII [You et al., 2022] to serve as a held-out discriminator for evaluating generated peptides. The original DeepMHCII model was developed for binding affinity prediction (regression); we repurposed it for binary eluted ligand classification by replacing the mean squared error loss with binary cross-entropy. No architectural changes were required, as DeepMHCII's convolutional kernels already capture core flanking regions—amino acids outside the MHC binding groove—which are particularly informative for EL data compared to affinity data.

The discriminator was trained on $\mathcal{D}_2$ and validated on $\mathcal{D}_3$, using the positive EL pairs from Nilsson et al. [2023] together with synthetic negatives subsampled to a 5:1 negative-to-positive ratio. To verify discriminator quality, we evaluated it on $\mathcal{D}_1$ (unseen during discriminator training) paired with negatives from Nilsson et al. [2023], achieving an AUROC of 0.971, indicating strong discriminative performance.

To support validation of the generative model without data leakage, we additionally trained a second discriminator on $\mathcal{D}_{\text{Train}}$ following the same procedure.

Table 5: Default hyperparameters for DNA sequence design experiment.

Experiment	Hyperparameter	Meaning	Value
Enhancer Design	Hidden dimension	Hidden dimension	256
	CNN stacks	Number of CNN blocks	4
	CNN layers	Number of CNN layers in one stack	5
	Kernel size	Size of filters in a stack	$[9, 9, 9, 9, 9]$
	Padding	Padding in a stack	$[4, 4, 16, 64, 256]$
	Dilation	Dilation in a stack	$[0, 0, 4, 16, 64]$
Promoter Design	Hidden dimension	Hidden dimension	256
	CNN stacks	Number of CNN blocks	4
	CNN layers	Number of CNN layers in one stack	5
	Kernel size	Size of filters in a stack	$[9, 9, 9, 9, 9]$
	Padding	Padding in a stack	$[4, 4, 16, 64, 256]$
	Dilation	Dilation in a stack	$[0, 0, 4, 16, 64]$
	p_{drop}	Unconditional drop rate	0.5
	w	Guidance scale for CFG/LG	3
Training	Max epochs	Total number of Epochs	1000
	lr	Learning Rate	$5e-4$
	Optimizer	Optimizer	Adam
	Batch Size	Samples in a batch	256
	Reverse time steps	Integration time steps	100
	Prior	Prior distribution	p_{freq}

Table 6: Default hyperparameters for Peptide sequence design experiment.

Experiment	Hyperparameter	Meaning	Value
Peptide Design	Hidden dimension	Hidden dimension	128
	CNN stacks	Number of CNN blocks	2
	CNN layers	Number of CNN layers in one stack	4
	Kernel size	Size of filters in a stack	$[9, 9, 9, 9, 9]$
	Padding	Padding in a stack	$[4, 4, 16, 64, 256]$
	Dilation	Dilation in a stack	$[0, 0, 4, 16, 64]$
	p_{drop}	Unconditional drop rate	0.2
	w	Guidance scale for CFG/LG	3
	Dropout	Dropout after each conv layer	0.3
	Gaussian std	Latent std σ_r	0.1
Training	Max epochs	Limit of number of Epochs	3000
	Early stopping	Early stopping patience	1000
	lr	Learning Rate	$5e-4$
	Optimizer	Optimizer	Adam
	Weight decay	Weight Decay Regularization	$1e-4$
	lr scheduler	Scheduler for Learning Rate	cosine
	Batch Size	Samples in a batch	256
	Reverse time steps	Integration time steps	100
	Prior	Prior distribution	p_{freq}

D Constructing K_γ matrices

We provide a reference implementation for constructing K_γ for proteins from a BLOSUM substitution matrix. The procedure converts BLOSUM log-odds scores into a row-stochastic transition kernel.

```
import torch
from sklearn.preprocessing import normalize
import itertools
from collections import Counter, OrderedDict
import csv
import pandas as pd
```

```python
import subprocess
import os
import urllib
import numpy as np

STOP = '*'
GAP = '-'
MSA_PAD = '!'
MASK = '#'
START = '@'
OTHER_AAS = 'JOU'
AAINDEX_ALPHABET = 'ARNDCQEGHILKMFPSTWYV'
AMB_AAS = 'BZX'
MSA_AAS = ALL_AAS + GAP
MSA_ALPHABET = ALL_AAS + GAP + STOP + MASK + START + MSA_PAD
BLOSUM_AAS = AAINDEX_ALPHABET + AMB_AAS
BLOSUM_ALPHABET = BLOSUM_AAS + OTHER_AAS + GAP + MSA_PAD + STOP + MASK +
    START
BLOSUM_EXTRAS = 'JOU-'


def loadMatrix(path):
    """
    Taken from https://pypi.org/project/blosum/
    Edited slightly from original implementation

    Reads a Blosum matrix from file. Changed slightly to read in larger
    blosum matrix
    File in a format like:
        https://www.ncbi.nlm.nih.gov/IEB/ToolBox/C_DOC/lxr/source/data/
    BLOSUM62
    Input:
        path: str, path to a file.
    Returns:
        blosumDict: Dictionary, The blosum dict
    """

    with open(path, "r") as f:
        content = f.readlines()

    blosumDict = {}

    header = True
    for line in content:
        line = line.strip()

        # Skip comments starting with #
        if line.startswith(";"):
            continue

        linelist = line.split()

        # Extract labels only once
        if header:
            labelslist = linelist
            header = False

            continue

        if not len(linelist) == len(labelslist) + 1:
            print(len(linelist), len(labelslist))
            # Check if line has as may entries as labels
            raise EOFError("Blosum file is missing values.")

        for index, lab in enumerate(labelslist, start=1):
            blosumDict[f"{linelist[0]}{lab}"] = float(linelist[index])

    if not len(blosumDict) == len(labelslist) ** 2:
```

```
        print(len(blosumDict), len(labelslist))
        raise EOFError("Blosum file is not quadratic.", len(blosumDict), len(labelslist)**2)
    return blosumDict

def softmax(x):
    """
    Compute softmax over x
    """
    return np.exp(x)/np.sum(np.exp(x),axis=0)

def double_stochastic(q):
    q_norm = normalize(q, axis=1, norm='l1')
    while not np.isclose(np.min(np.sum(q_norm, axis=0)), 1): # only checking that one value converges to 1 (prob best to do all 4 min/max)
        q_norm = normalize(q_norm, axis=0, norm='l1')
        q_norm = normalize(q_norm, axis=1, norm='l1')
    return q_norm


def K_blosum(matrix):
        alphabet = list("".join(MSA_ALPHABET))
        all_aas = list("".join(MSA_AAS))
        a_to_i = {u: i for i, u in enumerate(alphabet)}

        q = np.array([i for i in matrix.values()])
        q = q.reshape((len(all_aas),len(all_aas)))
        q = softmax(q)
        q = double_stochastic(q)
        q = torch.tensor(q)
        # REORDER BLOSUM MATRIX BASED ON MSA_ALPHABET (self.alphabet, self.a_to_i)
        new_q = q.clone()
        i2_to_a = np.array(list(BLOSUM_ALPHABET))
        for i, row in enumerate(new_q):
            for j, value in enumerate(row):
                ind1, ind2 = [i, j]
                key = i2_to_a[ind1], i2_to_a[ind2]
                new1, new2 = [a_to_i[k] for k in key]
                new_q[new1, new2] = q[ind1, ind2]

        return new_q
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See the results and methodology.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the ablation studies.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Provided in appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Provided in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Provided in appendix and data used is open-source data. We will release the new dataset upon acceptance of the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Model architecture, data preprocessing is provided in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Provided in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer:[N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.