GUMBEL-SOFTMAX FLOW MATCHING WITH STRAIGHT-THROUGH GUIDANCE FOR CONTROLLABLE BIOLOGICAL SEQUENCE GENERATION

Sophia Tang[1,2], **Yinuo Zhang**[1,3], **Alexander Tong**[4,5], **Pranam Chatterjee**[1,6,7,†]
[1]Department of Biomedical Engineering, Duke University
[2]Department of Computer and Information Science, University of Pennsylvania
[3]Center of Computational Biology, Duke-NUS Medical School
[4]Mila, Quebec AI Institute, [5]Université de Montréal
[6]Department of Computer Science, Duke University
[7]Department of Biostatistics and Bioinformatics, Duke University

[†]Corresponding author: pranam.chatterjee@duke.edu

ABSTRACT

Flow matching on the simplex has emerged as a promising strategy for sequence generation, but struggles with tasks that require stochastic interpolants and inference-time guidance. We introduce **Gumbel-Softmax Flow and Score Matching**, a generative framework built on a novel Gumbel-Softmax interpolant defined on the interior of the simplex. Using this interpolant, we introduce Gumbel-Softmax Flow Matching by deriving a parameterized velocity field that transports from noisy categorical distributions to one-hot distributions via a time-dependent temperature parameter. We alternatively present Gumbel-Softmax Score Matching, which learns to regress the gradient of the probability density. Our framework enables high-quality, diverse generation and scales efficiently to higher-dimensional simplices. To enable training-free guidance, we propose **Straight-Through Guided Flows (STGFlow)**, a classifier-based guidance method that leverages straight-through estimators to steer the unconditional velocity field toward optimal vertices of the simplex. STGFlow enables efficient inference-time guidance using classifiers pre-trained on clean sequences, and can be used with any discrete flow method. Together, these components form a robust framework for controllable *de novo* sequence generation. We demonstrate competitive performance in conditional DNA promoter design, target-binding peptide design for rare disease treatment, and *de novo* protein sequence design.

1 INTRODUCTION

Discrete diffusion [35, 33, 6] and flow-matching [17, 29] models, iteratively reconstruct sequences by modeling forward and reverse noise processes in a Markovian framework. These approaches have demonstrated success in DNA sequence design [38, 29], protein generation [44, 18], and recently, multi-objective generation of therapeutic peptides [42]. However, they operate in the fully *discrete* state space, meaning that the noisy sequence at each time step is a fully discrete sequence of one-hot vectors sampled from continuous categorical distributions. This can result in discretization errors during sampling when abruptly restricting continuous distributions to a single token, which raises the question: *Can we generate discrete sequences by iteratively fine-tuning continuous probability distributions?* This is the motivation behind discrete flow matching models on the simplex [38, 14], which defines a smooth interpolation from a uniform prior over the simplex to a unitary distribution concentrated at a single vertex.

Despite these advances, previous discrete simplex-based flow matching methods have yet to be applied to *de novo* design tasks such as protein and target-specific peptide design that require learning diverse flow trajectories that scale to higher simplex dimensions. Furthermore, there remains a lack

of *controllability* at inference time due to strictly deterministic paths and the absence of modular training-free guidance methods. To address these gaps, we introduce **Gumbel-Softmax Flow and Score Matching**, a novel framework for discrete generation on the simplex.

Our key contributions are as follows:

1. **Gumbel-Softmax Flow Matching** leverages temperature-controlled Gumbel-softmax interpolants defining a velocity field that smoothly transports from noisy to clean distributions on the simplex. By applying Gumbel noise during training, Gumbel-Softmax FM avoids overfitting to the training data, increasing the exploration of diverse flow trajectories. (Section 3).
2. **Gumbel-Softmax Score Matching** is an alternative framework that estimates the gradient of probability density at varying temperatures to enable sampling from high-density regions on the simplex (Section 4).
3. **Straight-Through Guided Flow Matching (STGFlow)** is a novel training-free classifier-based guidance algorithm for discrete flow matching that leverages straight-through gradients to guide the flow trajectory towards high-scoring sequences (Section 5).
4. **Biological Sequence Generation**; we show competitive performance against discrete diffusion and flow matching baselines in conditional DNA promoter design, target-binding peptide design (Section 6), and *de novo* protein sequence generation (Appendix G.2).

2 Preliminaries

We consider a noisy uniform distribution over the $(V-1)$-dimensional simplex $p_0(\mathbf{x}_0)$ and a clean distribution $p_1(\mathbf{x}_1)$ over discrete samples $\mathbf{x}_1 \sim \mathcal{D}$ from a dataset $\mathcal{D}$. One challenge of generative modeling on the simplex is defining a time-dependent flow ψ_t that smoothly transforms p_0 to p_1. Given ψ_t, we can generate samples from p_1 by first sampling from p_0 and applying a learned velocity field $u_t = \frac{d}{dt}\psi_t$ that transports distributions from p_0 to p_1.

2.1 The Gumbel-Softmax Distribution

The Gumbel-Softmax distribution or Concrete distribution [20, 26] is a relaxation of discrete random variables onto the interior of the simplex $\Delta^{V-1} = \{\mathbf{x} \in \mathbb{R}^V | x_i \in [0,1], \sum_{j=1}^V x_j = 1\}$. This continuous relaxation is achieved by adding i.i.d. sampled Gumbel noise $g_i = -\log(-\log \mathcal{U}_i))$, where $\mathcal{U}_i \sim \text{Uniform}(0,1)$, scaling down by the temperature parameter $\tau > 0$, and applying the differentiable `softmax` function across the distribution such that the probabilities sum to 1. Given parameters $\pi_i \in (\epsilon, \infty)$ representing the unnormalized logits of each category, where ϵ is a small constant to avoid undefined logarithms, the Gumbel-Softmax random variable is given by

$$x_i = \text{SM}\left(\frac{\log \pi_i + g_i}{\tau}\right) = \frac{\exp\left(\frac{\log \pi_i + g_i}{\tau}\right)}{\sum_{j=1}^V \exp\left(\frac{\log \pi_j + g_j}{\tau}\right)} \tag{1}$$

where $\text{SM}(\cdot)$ denotes the `softmax` function. We observe that as $\tau \to 0$, the distribution converges to a one-hot vector where $x_k \to 1$ and $x_j \to 0$ for $j \neq k$ given that $k = \arg\max_k (\log \pi_k + g_k)$. Conversely, as $\tau \to \infty$, the distribution approaches a uniform distribution where $x_j \to \frac{1}{V}$ for all $j \in [1, V]$.

2.2 Discrete Flow Matching

Flow matching [31, 24, 5] is a *simulation-free* generative framework that aims to transform noisy samples from a source distribution $\mathbf{x}_0 \sim p_0$ to clean samples from the data distribution $\mathbf{x}_1 \sim p_1$ by learning to predict the marginal velocity field $u_t(\mathbf{x}_t)$ that transports p_0 to p_1 as a mixture of conditional velocity fields $u_t^\theta(\mathbf{x}_t|\mathbf{x}_1)$ parameterized by a neural network. The *interpolant* $\psi_t(\mathbf{x}_1)$ is a function that transforms the one-hot distribution $\mathbf{x}_1$ to an intermediate noisy distribution $\mathbf{x}_t$ at time t, which satisfies the boundary constraints $\psi_0(\mathbf{x}_1) \approx \mathbf{x}_0$ and $\psi_1(\mathbf{x}_1) \approx \mathbf{x}_1$. Therefore, the conditional velocity field is given by the time derivative of $\psi_t(\mathbf{x}_1)$.

$$u_t(\mathbf{x}_t|\mathbf{x}_1) = \frac{d}{dt}\psi_t(\mathbf{x}_1) \tag{2}$$

where $u_t \in \mathcal{T}_{\mathbf{x}_t}\Delta^{V-1}$ is in the set of tangent vectors on the simplex at point $\mathbf{x}_t$. For a velocity field u_t to *generate* p_t, it must satisfy the *continuity equation* given by

$$\frac{\partial}{\partial t} p_t(\mathbf{x}_t) = -\nabla \cdot (p_t(\mathbf{x}_t) u_t(\mathbf{x}_t)) \tag{3}$$

where $\nabla\cdot$ is the divergence operator that describes the total outgoing flux at a point $\mathbf{x}_t$ along the flow trajectory. Intuitively, the continuity equation states that the rate of change of probability mass at a point in time is equal to the outgoing flux from that point, ensuring that probability mass is *conserved* across all points along the trajectory.

The flow matching (FM) objective is to train a parameterized model $u_t^\theta(\mathbf{x}_t)$ to approximate u_t given a noisy sample $\mathbf{x}_t$ at time $t \in [0, 1]$ by minimizing the squared norm

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\mathbf{x}_t} \left\| u_t^\theta(\mathbf{x}_t) - u_t(\mathbf{x}_t) \right\|^2 \tag{4}$$

But since computing $u_t(\mathbf{x}_t)$ requires marginalizing over all possible trajectories and is intractable, we condition the velocity field on each data point $\mathbf{x}_1$ and compute the conditional flow-matching (CFM) objective given by

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\mathbf{x}_t} \left\| u_t^\theta(\mathbf{x}_t) - u_t(\mathbf{x}_t|\mathbf{x}_1) \right\|^2 \tag{5}$$

which is tractable and has the same gradient as the unconditional flow-matching loss $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$ [23, 43]. Among existing discrete flow matching methods, there are two methods of defining a discrete flow: defining the *interpolant* $\psi_t(\mathbf{x}_1)$ that connects a noisy sample $\mathbf{x}_0$ to a clean one-hot sample $\mathbf{x}_1$ and defining the *probability path* which pushes density from the prior distribution p_0 to the target data distribution p_1. In this work, we define a new temperature-dependent interpolant and derive the corresponding velocity field.

2.3 Score Matching Generative Models

Score matching [36] is another generative matching framework that learns the gradient of the conditional probability density path $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ (defined as the *score*) of the interpolation between noisy and clean data. By parameterizing the score function with $s_\theta(\mathbf{x}_t, t)$, we can minimize the score matching loss given by

$$\mathcal{L}_{\text{score}} = \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) - s_\theta(\mathbf{x}_t, t) \right\|^2 \tag{6}$$

Similarly to flow-matching, directly learning $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is intractable, so we learn the conditional probability path $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1)$ conditioned on $\mathbf{x}_1 \sim p_1$ by minimizing

$$\mathcal{L}_{\text{score}} = \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \left\| \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1) - s_\theta(\mathbf{x}_t, t) \right\|^2 \tag{7}$$

which we show in Appendix D.1 equals the unconditional score function by expectation over $\mathbf{x}_1$.

3 Gumbel-Softmax Flow Matching

In this work, we present **Gumbel-Softmax Flow Matching (FM)**, a novel simplex-based flow matching method that defines the noisy logits at each time step with the Gumbel-Softmax transformation, enabling smooth interpolation between noisy and clean data by modulating the temperature $\tau(t)$, which changes as a function of time.

3.1 Defining the Gumbel-Softmax Interpolant

We define a novel interpolant that transforms near uniform to one-hot distributions by decreasing the temperature parameter in the Gumbel-Softmax function over time. Since the Gumbel-Softmax distribution is undefined at $\tau = 0$, we define the monotonically decreasing function $\tau(t)$ as

$$\tau(t) = \tau_{\text{max}} \exp(-\lambda t) \tag{8}$$

where $\tau_{\max}$ is the initial temperature set to a large number which produces a near uniform distribution, λ controls the decay rate, and $t \in [0, 1]$ is the time period.

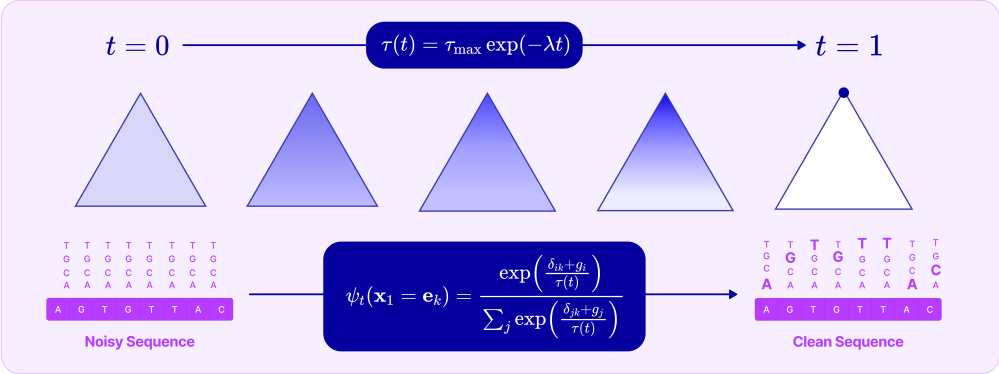

Figure 1: **Overview of Gumbel-Softmax Flow Matching.** Gumbel-softmax transformations are applied to clean one-hot sequences for varying time-dependent temperatures to learn to transform noisy to clean sequences. The full algorithm is detailed in Algorithm 1 and 2.

Now, we define the conditional interpolant $\mathbf{x}_t = \psi_t(\mathbf{x}_1 = \mathbf{e}_k)$ with $t \in [0, 1]$ and Gumbel-noise scaled by a factor β as

$$\psi_t(\mathbf{x}_1 = \mathbf{e}_k) = \frac{\exp\left(\frac{\delta_{ik}+(g_i/\beta)}{\tau(t)}\right)}{\sum_{j=1}^{V} \exp\left(\frac{\delta_{jk}+(g_j/\beta)}{\tau(t)}\right)} \tag{9}$$

where $\tau(t) = \tau_{\max} \exp(-\lambda t)$ and $\pi_i = \exp(\delta_{ik})$. δ_{ik} is the Kronecker delta function that returns 1 when $i = k$ and 0 otherwise. This decaying time-dependent temperature function $\tau(t)$ ensures that the distribution becomes more concentrated at the target vertex as $t \to 1$. Gumbel noise is applied during training to ensure that the model learns to reconstruct a clean sequence given contextual information.

Proposition 1 (Continuity)**.** *The proposed conditional vector field and conditional probability path together satisfy the continuity equation (Equation 3) and thus define a valid flow matching trajectory on the interior of the simplex.*

We provide the proof of continuity in Appendix C.2. This definition of the flow satisfies the boundary conditions. For $t = 0$, $\tau(t) = \tau_{\max}$ which produces a near-uniform distribution $\psi_0(\mathbf{x}_0|\mathbf{x}_1) \approx \frac{\mathbf{1}}{V}$. For $t = 1$, $\exp(-\lambda t) \to 0$ (faster decay for larger λ) and $\tau(t) \to 0$, meaning the flow trajectory converges to the vertex of the simplex corresponding to the one-hot vector $\psi_1(\mathbf{x}_0|\mathbf{x}_1) \approx \mathbf{x}_1$.

3.2 Reparameterizing the Velocity Field

From our definition of the Gumbel-Softmax interpolant, we derive the conditional velocity field $u_t(\mathbf{x}_0|\mathbf{x}_1)$ by taking the derivative of the flow (Appendix C.1).

$$u_{t,i}(\mathbf{x}|\mathbf{x}_1 = \mathbf{e}_k) = \frac{\lambda}{\tau(t)} x_{t,i} \sum_{j=1}^{V} x_{t,j} \cdot \Big((\delta_{ik} + g_i) - (\delta_{jk} + g_j) \Big) \tag{10}$$

Proposition 2 (Probability Mass Conservation)**.** *The conditional velocity field preserves the probability mass and lies in the tangent bundle at point $\mathbf{x}_t$ on the simplex $\mathcal{T}_{\mathbf{x}_t}\Delta^{V-1}$.*

Proof in Appendix C.3. Instead of directly regressing $u_t(\mathbf{x}_t|\mathbf{x}_1)$ by minimizing $\mathcal{L}_{\text{CFM}}$ defined in Equation 5, we train a *denoising* model that predicts the probability vector $\mathbf{x}_\theta(\mathbf{x}_t, t) \in \Delta^{V-1}$ given the noisy interpolant $\mathbf{x}_t$ by minimizing the negative log loss.

$$\mathcal{L}_{\text{gumbel}} = \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1=\mathbf{e}_k), p_1(\mathbf{x}_1)} \left[-\log \langle \mathbf{x}_\theta(\mathbf{x}_t, t), \mathbf{x}_1 \rangle \right] \tag{11}$$

During inference, we compute the predicted marginal velocity field as the weighted sum of the conditional velocity fields scaled by the predicted token probabilities.

$$u_t^\theta(\mathbf{x}_t) = \sum_{k=1}^{V} u_t(\mathbf{x}|\mathbf{x}_1 = \mathbf{e}_k)\langle \mathbf{x}_\theta(\mathbf{x}_t, t), \mathbf{e}_k\rangle \tag{12}$$

Proposition 3 (Valid Flow Matching Loss)**.** *If $p_t(\mathbf{x}_t) > 0$ for all $\mathbf{x}_t \in \mathbb{R}^d$ and $t \in [0, 1]$, then the gradients of the flow matching loss and the Gumbel-Softmax FM loss are equal up to a constant not dependent on θ such that $\nabla_\theta \mathcal{L}_{FM} = \nabla_\theta \mathcal{L}_{gumbel}$*

Proof in Appendix C.3. By our definition of the Gumbel-Softmax interpolant, the intermediate distributions during inference represent a mixture of learned conditional interpolants $\psi_t(\mathbf{x}_1)$ from the training data. Since the denoising model is trained to predict the true clean distribution, we can set the Gumbel-noise random variable in the conditional velocity fields to 0 during inference, as we want the velocity field to point toward the predicted denoised distribution. Therefore, the conditional velocity field becomes

$$u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) = \frac{\lambda}{\tau(t)} x_{t,k} \left(\mathbf{e}_k - \mathbf{x}_t\right) \tag{13}$$

which points toward the target vertex $\mathbf{e}_k$ at a magnitude proportional to $x_{t,k}(1 - x_{t,k})$ and away from all other vertices at a magnitude proportional to $-x_{t,i}x_{t,k}$. We observe that the velocity field vanishes both at the vertex and the $(V - 2)$-dimensional face directly opposite to the vertex and increases as $t \to 1$ and $\tau(t) \to 0$, accelerating towards the target vertex at later time steps.

4 Gumbel-Softmax Score Matching

As an alternative to our flow matching framework, we propose **Gumbel-Softmax Score Matching (SM)**, a score-matching method that learns the gradient of the probability density path $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ associated with the Gumbel-Softmax interpolant.

4.1 The Exponential Concrete Distribution

When computing Gumbel-Softmax random variables, the exponentiation of small values associated with low-probability tokens can result in numerical underflow. Since the logarithm of 0 is undefined, this could result in numerical instabilities when computing the log probability density. To avoid instabilities, we take the logarithm of the Gumbel-Softmax probability distribution (known as the ExpConcrete distribution [26]) given by $x_i = \log\left(\text{SM}\left(\frac{\log \pi_i + g_i}{\tau}\right)\right)$. Expanding the logarithm, we get that the ith element ExpConcrete random variable is defined as

$$x_i = \frac{\log \pi_i + (g_i/\beta)}{\tau} - \log \sum_{j=1}^{V} \exp\left(\frac{\log \pi_j + (g_j/\beta)}{\tau}\right) \tag{14}$$

Translating this into our time-dependent interpolant where $\pi_i = \exp(\delta_{ik})$, we define

$$\psi_t(\mathbf{x}_1 = \mathbf{e}_k) = \frac{\delta_{ik} + (g_i/\beta)}{\tau(t)} - \log \sum_{j=1}^{V} \exp\left(\frac{\delta_{jk} + (g_j/\beta)}{\tau(t)}\right) \tag{15}$$

By our derivation in Appendix D.1, the *score* defined as the gradient of the log-probability density of the ExpConcrete interpolant with respect to the ith element $x_{t,i}$ is given by

$$\nabla_{x_{t,i}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\tau(t) + \tau(t)V \cdot \text{SM}\Big(\delta_{ik} - \tau(t)x_{t,i}\Big) \tag{16}$$

4.2 Learning the Gumbel-Softmax Probability Density

Given that the Gumbel-Softmax interpolant naturally converges towards the one-hot target token distribution, it follows that learning the evolution of probability density across training samples would

enable generation in regions with high probability density. Our goal is to train a parameterized model to learn to estimate the gradient of the log-probability density of the Gumbel-Softmax interpolant such that the gradient converges at regions with high probability density. To achieve this, we define the score parameterization similar to [27], given by

$$s_\theta(\mathbf{x}_t, t) = -\tau(t) + \tau(t)V \cdot \mathrm{SM}\big(f_\theta(\mathbf{x}_t, t)\big) \quad \text{where} \quad s_\theta(\mathbf{x}_t, t) \approx \nabla_{x_{t,j}} \log p_t(\mathbf{x}_t) \tag{17}$$

where θ minimizes the reparameterized score-matching loss given by

$$\begin{aligned}\mathcal{L}_{\text{score}} &= \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \Big\| \big[-\tau(t) + \tau(t)V \cdot \mathrm{SM}(\delta_{ik} - \tau(t)x_{t,i}) \big] - \big[-\tau(t) + \tau(t)V \cdot \mathrm{SM}(f_\theta(\mathbf{x}_t, t) \big] \Big\|^2 \\ &= \tau(t)^2 V^2 \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \|\mathrm{SM}\big(\delta_{ik} - \tau(t)x_{t,i}\big) - \mathrm{SM}\big(f_\theta(\mathbf{x}_t, t)\big)\|^2 \end{aligned} \tag{18}$$

The `softmax` function applied after parameterization ensures dependencies are preserved across the predicted output vector, which defines the rate of probability flow towards each vertex. Since $\tau(t) \to 0$ when $t \to 1$, we remove the scaling term to ensure the losses are evenly scaled over time.

$$\mathcal{L}_{\text{score}} = \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \|\mathrm{SM}\big(\delta_{ik} - \tau(t)x_{t,i}\big) - \mathrm{SM}\big(f_\theta(\mathbf{x}_t, t)\big)\|^2 \tag{19}$$

Proposition 4. *The gradient of the* EXPCONCRETE *log-probability density is proportional to the gradient of the Gumbel-softmax log-probability density such that* $\nabla^{GS}_{x_j} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1) \propto \nabla^{ExpConcrete}_{x_j} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1)$.

Proof in Appendix D.2. Therefore, by minimizing $\mathcal{L}_{\text{score}}$, we obtain a model that effectively transports intermediate Gumbel-Softmax distributions towards clean distributions in high-probability regions of the discrete state space.

5 STRAIGHT-THROUGH GUIDED FLOWS (STGFLOW)

In this section, we introduce **Straight-Through Guided Flows (STGFlow)**, a classifier-based guidance method that guides the pre-trained conditional flow velocities towards sequences with higher classifier probabilities $p^\phi(y|\mathbf{x}_t)$ which does not require training a time-dependent classifier or classifier-guided velocity field. STGFlow leverages straight-through gradient estimators to compute gradients of classifier scores from discrete sequence samples with respect to the continuous logits from which they were sampled.

5.1 STRAIGHT-THROUGH GRADIENT ESTIMATORS

Straight-through gradient estimators aim to solve the problem of taking gradients with respect to *discrete random variables*. Consider a reward function $\mathcal{R}(\mathbf{z})$ that takes a discrete sequence $\mathbf{z}$ of length L sampled from a learned distribution $p_\theta(\mathbf{z})$, and our goal is to maximize the reward

$$\max_\theta \mathcal{R} = \min_\theta \mathbb{E}_{\mathbf{z}\sim p_\theta}\big[\mathcal{R}(\mathbf{z})\big] \tag{20}$$

Given the non-differentiability of $\mathcal{R}(\mathbf{z})$ with respect to the parameters θ, the Straight-Through Gumbel-Softmax estimator (ST-GS) [20] evaluates the gradient of the reward function through a *surrogate* of the discrete random variable $\mathbf{z}$ defined as the tempered `softmax` distribution over the continuous logits from which $\mathbf{z}$ was sampled.

$$\nabla_\theta \mathcal{R} = \frac{\partial \mathcal{R}(\mathbf{z})}{\partial \mathbf{z}} \frac{d}{d\theta} \mathrm{SM}_\tau(p_\theta(\mathbf{z})) \tag{21}$$

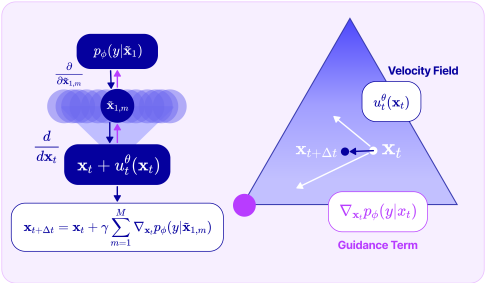

Figure 2: **Straight-Through Guided Flows (STGFlow).** The gradients of the classifier function with respect to M discrete sequences sampled from the intermediate token distribution $\mathbf{x}_t$ act as a guided flow velocity that steers the unconditional trajectory towards sequences with optimal scores. The full algorithm is detailed in Algorithm 5.

ST-GS preserves the forward evaluation of the reward function while enabling low-variance gradient estimation for back-propagation of the gradient that does not need to be defined over continuous relaxations of discrete variables over the simplex. Instead, they only need to be defined for *discrete* sequences, which is the case for most pre-trained classifier models.

5.2 STRAIGHT-THROUGH GUIDED FLOW MATCHING

We extend the idea of ST-GS to define a novel post-training guidance method. At each time step t, we compute the Gumbel-Softmax velocity field $u_t^\theta(\mathbf{x}_t)$ and take a step. Then, from the updated logits, we sample M discrete sequences $\{\tilde{\mathbf{x}}_{1,1}, \ldots, \tilde{\mathbf{x}}_{1,M}\}$ from the top k logits in $\mathbf{x}_t$ re-normalized with the `softmax` function. For each sequence $\tilde{\mathbf{x}}_{1,m}$, we compute a classifier score using our pre-trained classifier $p_\phi(y|\tilde{\mathbf{x}}_{1,m})$. Since the gradient through the `argmax` function is either 0 or undefined, we compute the gradient of the classifier model with respect to the surrogate `softmax` distribution.

$$\nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) = \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_{1,m}} \frac{d}{d\mathbf{x}_t} \mathrm{SM}(\mathbf{x}_t) \tag{22}$$

Evaluating the straight-through gradient with respect to the probability of each token, we have

$$\nabla_{x_{t,i}} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) = \begin{cases} \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \cdot \left[\mathrm{SM}(x_{t,i})\left(1 - \mathrm{SM}(x_{t,k})\right)\right] & i = k \\ \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \cdot \left[-\mathrm{SM}(x_{t,i})\mathrm{SM}(x_{t,k})\right] & i \neq k \end{cases} \tag{23}$$

where k denotes the index of the sampled token such that $\tilde{\mathbf{x}}_{1,m} = \mathbf{e}_k$. During inference, the partial derivative term $\frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_{1,m}}$ is computed with automatic differentiation with respect to each sequence position, enabling position-specific guidance. Finally, we guide the flow trajectory by adding the aggregate gradient across all M sequences, scaled by a constant γ to get

$$\mathbf{x}_t = \mathbf{x}_t + \gamma \sum_{m=1}^{M} \nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) \tag{24}$$

Proposition 5 (Conservation of Probability Mass of Straight-Through Gradient)**.** *The straight-through gradient $\nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m})$ preserves probability mass and lies on the tangent bundle at $\mathbf{x}_t$.*

Proof in Appendix E. Conceptually, the straight-through gradient acts as a guiding velocity that steers the unconditional velocity toward valid, optimal sequences. Pseudocode for STGFlow is provided in Algorithm 5.

6 EXPERIMENTS

6.1 SIMPLEX-DIMENSION TOY EXEPRIMENT

Setup. Following Stark et al. [38], we conduct a toy experiment that evaluates the KL divergence between the empirically-generated distribution and a random distribution of sequence length 4 over the $(V-1)$-dimensional simplex $(\Delta^{V-1})^4$ for $K = \{20, 40, 60, 80, 100, 120, 140, 160, 512\}$. The sequence length is set to 4, and the number of integration steps was set to 100 across all experiments.

Training. We trained Linear FM [38], Dirichlet FM [38], Fisher FM [14], and Gumbel-Softmax FM each for 50K steps on 100K sequences from a randomly generated distribution. We evaluated the KL divergence $\mathrm{KL}(\tilde{q}\|p_{\text{data}})$ where $\tilde{q}$ is the normalized distribution from 51.2K sequences generated by the model and p_{data} is the distribution from which the training data was sampled.

Results. As shown in Table 5, Gumbel-Softmax FM achieves superior performance to Dirichlet FM when scaled to dimensions $K \geq 60$, with stable KL divergence in the range $0.02 - 0.05$ for all simplex dimensions up to $K = 512$. Although Gumbel-Softmax FM achieves higher KL divergence than Fisher FM, we note that the use of optimal transport in Fisher FM results in learning straight, deterministic flows that can result in overfitting to the training data. This can be observed when comparing the curves of the validation mean-squared error loss between the predicted and true conditional velocity fields summed over the simplex and sequence length dimensions (Figure 5).

6.2 PROMOTER DNA SEQUENCE DESIGN

Following the procedures of previous works [7, 38], we evaluate Gumbel-Softmax FM for conditional DNA promoter design and show superior performance to discrete diffusion and flow-matching baselines.

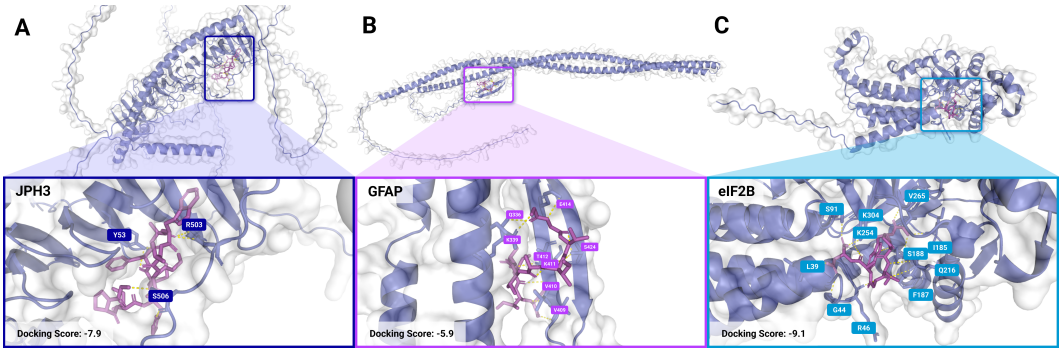

Figure 3: **Gumbel-Softmax FM generated peptide binders for three targets with no known binders.** (**A**) 10 a.a. designed binder to JPH3 (structure generated with AlphaFold3) involved in Huntington's Disease-Like 2. (**B**) 10 a.a. designed binder to GFAP (PDB: 6A9P) involved in Alexander Disease. (**C**) 7 a.a. designed binder to eIF2B (PDB: 6CAJ) involved in Vanishing White Matter Disease. Docked with AutoDock VINA and polar contacts within 3.5 Å are annotated. Additional targets are shown in Table 2.

Setup. Promoter DNA is the strand of DNA adjacent to a gene that binds to RNA polymerase and transcription factors to promote gene transcription and expression. The objective is to train a conditional flow model with the regulatory signal concatenated to the noisy input sequence to minimize the mean squared error (MSE) between the predicted regulatory activity of the generated sequence with the true sequence, predicted with a pre-trained Sei model [11].

Training. Following Stark et al. [38], we trained on a train/test/validation split of $88,470/3,933/7,497$ promoter sequences that are 1,024 base pairs in length. For each batch of size 256, we applied the Gumbel-Softmax transformation according to Equation 9 with $\tau_{\max} = 10.0$ and $\lambda = 3.0$ for uniformly distributed time steps $t \in [0, 1]$ over each training batch. The models were parameterized with a 20-layer 1D CNN architecture for 150K steps and evaluated the MSE across all validation batches.

Results. We show that Gumbel-Softmax FM produces lower signal MSE compared to diffusion and autoregressive language model baselines [7, 12, 6] and similar MSE to Dirichlet and Fisher FM [38, 14].

Model	**MSE** (↓)
Bit Diffusion (Bit Encoding)*	0.041
Bit Diffusion (One-Hot Encoding)*	0.040
D3PM-Uniform*	0.038
DDSM*	0.033
Language Model*	0.033
Dirichlet Flow Matching	0.029
Fisher Flow Matching	0.030
Gumbel-Softmax Flow Matching (Ours)	0.029

Table 1: **Evaluation of promoter DNA generation conditioned on transcription profile.** MSE was evaluated across all validation batches between the predicted signal of a conditionally generated sequence and the true sequence. Regulatory signals were predicted with a pre-trained Sei model [11]. Numbers with * are from Stark et al. [38]

6.3 Peptide Binder Design

Finally, we show that Gumbel-Softmax FM coupled with STGFlow is capable of generating *de novo* peptide binders with similar or higher binding affinity to proteins with known peptide binders and diverse, rare disease-associated proteins without known peptide binders.

Setup. We generated *de novo* peptide binders protein targets with existing an no existing peptide binders following Algorithm 5. To guide the flow paths, we train a target-binding cross-attention-based regression model (Appendix F.2) that takes the ESM2 [22] representations of a peptide and protein sequence pair and predicts the $K_d/K_i/IC50$ score. Using a dataset of 1781 experimentally validated peptides, our model achieved a strong Spearman correlation coefficient of 0.96 on the training set and 0.64 on the validation set.

Table 2: **Top**: Comparison of ipTM (AF3) and VINA docking scores for existing and designed peptide binders to protein targets. **Bottom**: Comparison of scores for scrambled and designed peptide binders to proteins with no existing binders. *Contains unnatural amino acid X, which cannot be processed by AlphaFold3. **No PDB structure available. Used AlphaFold3 predicted structure for docking.

PDB ID	existing binder	ipTM (↑)		pTM (↑)		VINA Docking Score (kcal/mol) (↓)		
		existing	designed	existing	designed	existing	designed	
GLP-1R (3C5T)	HXEGTFTSDVSSYLEGQAAKEFIAWLVRGRG	*	0.65	*	0.66	-5.7	-7.5	
1AYC	ARLIDDQLLKS	0.68	0.67	0.88	0.88	-5.3	-4.6	
2Q8Y	ALRRELADW	0.44	0.70	0.83	0.84	-6.7	-6.8	
3EQS	GDHARQGLLALG	0.80	0.71	0.88	0.86	-4.4	-4.7	
3NIH	RIAAA	0.85	0.86	0.91	0.90	-6.2	-5.7	
4EZN	VDKGSYLPRPTPPRPIYNRN	0.54	0.59	0.85	0.87	-4.1	-6.5	
4GNE	ARTKQTA	0.89	0.76	0.76	0.76	-5.0	-4.8	
4IU7	HKILHRLLQD	0.93	0.79	0.91	0.94	-4.6	-5.9	
5E1C	KHKILHRLLQDSSS	0.83	0.80	0.91	0.91	-4.3	-5.1	
5EYZ	SWESHKSGRETEV	0.73	0.81	0.77	0.78	-2.9	-6.9	
5KRI	KHKILHRLLQDSSS	0.83	0.77	0.91	0.91	-3.5	-5.5	
7LUL	RWYERWV	0.94	0.91	0.93	0.92	-6.5	-7.6	
8CN1	ETEV	0.90	0.86	0.72	0.82	-6.0	-6.9	
Protein Name (PDB ID)	**Disease**	**ipTM (↑)**		**pTM (↑)**		**VINA Docking Score (kcal/mol) (↓)**		
		scrambled	designed	scrambled	designed	scrambled	designed	
GFAP (6A9P)	Alexander Disease	0.38	0.62	0.29	0.31	-3.7	-5.9	
eIF2B (6CAJ)	Vanishing White Matter Disease	0.39	0.61	0.76	0.77	-9.0	-9.1	
Gigaxonin (3HVE)	Giant Axonal Neuropathy	0.54	0.75	0.82	0.83	-6.2	-6.8	
NPC2 (6W5V)	Niemann-Pick Disease Type C	0.34	0.80	0.77	0.79	-5.6	-6.5	
JPH3 (**)	Huntington's Disease-Like 2 (HDL2)	0.60	0.72	0.49	0.49	-7.8	-7.9	
2CKL	BMI1	Medulloblastoma	0.43	0.71	0.73	0.81	-6.2	-6.8

Training. We fine-tuned our Gumbel-Softmax FM protein generator (Appendix G.2) for 600 epochs on $17,479$ peptides ($0.8/0.2$ train/validation split) between $6-50$ amino acids in length curated from the PepNN [1], BioLip2 [45], and PPIRef [8] datasets.

Results. First, we compare peptide binders generated by Gumbel-Softmax FM coupled with STGFlow guidance to existing peptide binders to 13 protein targets (Table 2). After generating 20 *de novo* peptides of the same length as the existing binders, we computed the ipTM and pTM scores using AlphaFold3 to evaluate the predicted confidence of the peptide-protein complexes and the docking scores using AutoDock VINA to evaluate the free energy of the binding interaction (See Appendix H.3 for details on evaluation metrics). From the final *de novo* generated peptides with optimized classifier scores against each target, we show consistent generation of peptides with superior ipTM (↑) and VINA docking scores (↓) compared to experimentally-validated binders (Table 2), indicating the efficacy of guided flow matching strategy in generating peptides with high binding affinity.

To further validate the versatility of our framework, we evaluated peptide binders guided for six proteins involved in various diseases with no pre-existing peptide binders (Figure 7; Table 2). We generated 20 peptide binders that are $5-15$ amino acids in length with Gumbel-Softmax FM and STGFlow guidance and randomly permuted the sequence to generate a scrambled negative control for comparison. Notably, our designed binders demonstrate strong ipTM higher than 0.62 and VINA docking scores below -5.9. Despite the short sequence length, we also show that scrambling the order of amino acids consistently decreases the binding affinity compared to the unscrambled binder, indicating that our guidance strategy effectively captures dependencies across tokens that lead to higher-affinity peptides (Table 2). Furthermore, the docked peptides show complementary structures to the target protein with several polar contacts within 3.5 Å (Figure 7).

Since pTM (↑) scores are dominated by the confidence in the protein target structure, there are no significant differences in the scores between the designed binders and control peptides; however, we still observe slightly higher scores, indicating that our designed binders enhance the stability of the protein structure. Plotting the predicted binding affinity scores over the iteration or time step, we consistently see sharp upward curves, which proves the efficacy of STGFlow in optimizing classifier scores (Figure 4).

7 Conclusion

In this work, we introduce **Gumbel-Softmax Flow and Score Matching**, a novel discrete generative framework that learns interpolations between noisy and clean sequences by modulating the temperature of the Gumbel-Softmax distribution. By parameterizing a straight continuous-time interpolation with stochastic Gumbel noise, we overcome limitations in discrete sampling errors, scalability to

higher simplex dimensions, and deterministic flows of existing discrete generative frameworks. In addition, we close the gap in training-free guidance for discrete flow matching with **STGFlow**, an algorithm that leverages straight-through gradient estimators to steer the flow trajectory.

REFERENCES

[1] Osama Abdin, Satra Nim, Han Wen, and Philip M. Kim. Pepnn: a deep attention model for the identification of peptide binding sites. *Communications Biology*, 5(1), May 2022. ISSN 2399-3642. doi: 10.1038/s42003-022-03445-2. URL http://dx.doi.org/10.1038/s42003-022-03445-2.

[2] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, May 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07487-w. URL http://dx.doi.org/10.1038/s41586-024-07487-w.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

[4] 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. September 2023. doi: 10.1101/2023.09.11.556673. URL http://dx.doi.org/10.1101/2023.09.11.556673.

[5] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint 2303.08797*, 2023.

[6] 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*, 2021. doi: 10.48550/ARXIV.2107.03006. URL https://arxiv.org/abs/2107.03006.

[7] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation, 2023. URL https://arxiv.org/abs/2305.10699.

[8] Anton Bushuiev, Roman Bushuiev, Petr Kouba, Anatolii Filkin, Marketa Gabrielova, Michal Gabriel, Jiri Sedlar, Tomas Pluskal, Jiri Damborsky, Stanislav Mazurenko, and Josef Sivic. Learning to design protein-protein interactions with enhanced generalization, 2023. URL https://arxiv.org/abs/2310.18515.

[9] Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models. October 2022. URL https://openreview.net/forum?id=DmT862YAieY.

[10] 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*, 2024. doi: 10.48550/ARXIV.2402.04997. URL https://arxiv.org/abs/2402.04997.

[11] 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, July 2022. ISSN 1546-1718. doi: 10.1038/s41588-022-01102-2. URL http://dx.doi.org/10.1038/s41588-022-01102-2.

[12] Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning, 2022. URL https://arxiv.org/abs/2208.04202.

[13] Andre Cornman, Jacob West-Roberts, Antonio Pedro Camargo, Simon Roux, Martin Beracochea, Milot Mirdita, Sergey Ovchinnikov, and Yunha Hwang. The omg dataset: An open metagenomic corpus for mixed-modality genomic language modeling. 2024. doi: 10.1101/2024.08.14.607850. URL https://www.biorxiv.org/content/early/2024/08/17/2024.08.14.607850.

[14] Oscar Davis, Samuel Kessler, Mircea Petrache, Ismail Ilkan Ceylan, Michael Bronstein, and Avishek Joey Bose. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 2024. doi: 10.48550/ARXIV.2405.14664. URL https://arxiv.org/abs/2405.14664.

[15] Jerome Eberhardt, Diogo Santos-Martins, Andreas F Tillack, and Stefano Forli. Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of chemical information and modeling*, 61(8):3891–3898, 2021.

[16] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1), July 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-32007-7. URL http://dx.doi.org/10.1038/s41467-022-32007-7.

[17] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 2024. doi: 10.48550/ARXIV.2407.15595. URL https://arxiv.org/abs/2407.15595.

[18] Shrey Goel, Vishrut Thoutam, Edgar Mariano Marroquin, Aaron Gokaslan, Arash Firouzbakht, Sophia Vincoff, Volodymyr Kuleshov, Huong T. Kratochvil, and Pranam Chatterjee. Memdlm: De novo membrane protein design with masked discrete diffusion protein language models. *arXiv*, 2024. doi: 10.48550/ARXIV.2410.16735. URL https://arxiv.org/abs/2410.16735.

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2022. doi: 10.48550/ARXIV.2207.12598. URL https://arxiv.org/abs/2207.12598.

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learned Representations*, 2017. doi: 10.48550/ARXIV.1611.01144. URL https://arxiv.org/abs/1611.01144.

[21] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. ISSN 1095-9203. doi: 10.1126/science.ade2574. URL http://dx.doi.org/10.1126/science.ade2574.

[22] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan Dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023.

[23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 2023. doi: 10.48550/ARXIV.2210.02747. URL https://arxiv.org/abs/2210.02747.

[24] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint 2209.14577*, 2022.

[25] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *International Conference on Machine Learning*, 2024. doi: 10.48550/ARXIV.2310.16834. URL https://arxiv.org/abs/2310.16834.

[26] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables, 2016. URL https://arxiv.org/abs/1611.00712.

[27] Ahsan Mahmood, Junier Oliva, and Martin Andreas Styner. Anomaly detection via gumbel noise score matching. *Frontiers in Artificial Intelligence*, 7, September 2024. ISSN 2624-8212. doi: 10.3389/frai.2024.1441205. URL http://dx.doi.org/10.3389/frai.2024.1441205.

[28] 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.e3, November 2023. ISSN 2405-4712. doi: 10.1016/j.cels.2023.10.002. URL http://dx.doi.org/10.1016/j.cels.2023.10.002.

[29] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *International Conference on Learning Representations*, 2025. doi: 10.48550/ARXIV.2406.01572. URL https://arxiv.org/abs/2406.01572.

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi: 10.48550/ARXIV.2212.09748. URL https://arxiv.org/abs/2212.09748.

[31] Stefano Peluchetti. Non-denoising forward-time diffusions, 2022. URL https://openreview.net/forum?id=oVfIKuhqfC.

[32] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. *International Conference on Machine Learning*, 2023. doi: 10.48550/ARXIV.2304.14772. URL https://arxiv.org/abs/2304.14772.

[33] Subham Sekhar 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*, 2024. doi: 10.48550/ARXIV.2406.07524. URL https://arxiv.org/abs/2406.07524.

[34] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[35] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in Neural Information Processing Systems*, 2024. doi: 10.48550/ARXIV.2406.04329. URL https://arxiv.org/abs/2406.04329.

[36] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 2019. doi: 10.48550/ARXIV.1907.05600. URL https://arxiv.org/abs/1907.05600.

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2021. doi: 10.48550/ARXIV.2011.13456. URL https://arxiv.org/abs/2011.13456.

[38] Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *ICML*, 2024.

[39] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):2542, 2018.

[40] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021. URL https://arxiv.org/abs/2104.09864.

[41] Baris E Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H Wu. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[42] Sophia Tang, Yinuo Zhang, and Pranam Chatterjee. Peptune: De novo generation of therapeutic peptides with multi-objective-guided discrete diffusion. *arXiv*, 2024. doi: 10.48550/ARXIV.2412.17780. URL https://arxiv.org/abs/2412.17780.

[43] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. doi: 10.48550/ARXIV.2302.00482. URL https://arxiv.org/abs/2302.00482.

[44] Mingyang Wang, Shuai Li, Jike Wang, Odin Zhang, Hongyan Du, Dejun Jiang, Zhenxing Wu, Yafeng Deng, Yu Kang, Peichen Pan, Dan Li, Xiaorui Wang, Xiaojun Yao, Tingjun Hou, and Chang-Yu Hsieh. Clickgen: Directed exploration of synthesizable chemical space via modular reactions and reinforcement learning. *Nature Communications*, 15(1), November 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-54456-y. URL http://dx.doi.org/10.1038/s41467-024-54456-y.

[45] Chengxin Zhang, Xi Zhang, Lydia Freddolino, and Yang Zhang. Biolip2: an updated structure database for biologically relevant ligand–protein interactions. *Nucleic Acids Research*, 52 (D1):D404–D412, July 2023. ISSN 1362-4962. doi: 10.1093/nar/gkad630. URL http://dx.doi.org/10.1093/nar/gkad630.

[46] Ruochi Zhang, Haoran Wu, Yuting Xiu, Kewei Li, Ningning Chen, Yu Wang, Yan Wang, Xin Gao, and Fengfeng Zhou. Pepland: a large-scale pre-trained peptide representation model for a comprehensive landscape of both canonical and non-canonical amino acids. *arXiv*, 2023. doi: 10.48550/ARXIV.2311.04419. URL https://arxiv.org/abs/2311.04419.

[47] Xi Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis L. Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modeling, 2024. URL https://arxiv.org/abs/2410.21154.

[48] Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky T. Q. Chen. Guided flows for generative modeling and decision making, 2023. URL https://arxiv.org/abs/2311.13443.

A EXTENDED BACKGROUND

A.1 FLOW MATCHING ON THE SIMPLEX

Here, we discuss the motivation behind discrete flow matching [10, 17], and specifically on the interior of the simplex [38, 14]. This discussion will help motivate the contribution of our work from past iterations.

Discrete diffusion models [6, 25, 9] operate by applying categorical noise in the form of $\mathbf{x}_t \sim \text{Cat}(\cdot|\mathbf{Q}_t^\top \mathbf{x}_0)$ that convert the clean sequence of one-hot categorical distributions $\mathbf{x}_0$ to a noisy sequence $\mathbf{z}_t$. Then, a parameterized model learns to iteratively reconstruct the clean sequence $\mathbf{x}_0$ from the noisy sequence $\mathbf{z}_t$ by taking t discrete backward transitions given by $\mathbf{z}_s \sim \text{Cat}\left(\cdot|\frac{\mathbf{Q}_{s|t}\mathbf{z}_t \odot \mathbf{Q}_s^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}{\mathbf{z}_t^\top \mathbf{Q}_t^\top \mathbf{x}_\theta(\mathbf{z}_t,t)}\right)$. However, this method operates in the fully *discrete* state space, meaning that the noisy sequence at each time step is a fully discrete sequence of one-hot vectors sampled from continuous categorical distributions. This can result in discretization errors during sampling when abruptly restricting continuous distributions to a single token. This presents the question: *Can we generate discrete sequences by iteratively fine-tuning continuous probability distributions?*

This is the motivation behind discrete flow matching models on the simplex [38, 14], which defines a smooth interpolation $\psi_t(\mathbf{x}_1)$ from a prior uniform distribution over the simplex $\mathbf{x}_0$ to a unitary distribution concentrated at a single vertex $\mathbf{x}_1$ over the time interval $t \in [0, 1]$. To ensure that noisy can be transformed into valid, clean sequences at inference, the interpolant must satisfy the boundary conditions given by $\psi_0(\mathbf{x}_1) \approx \frac{\mathbf{1}}{V}$ where V is the size of the token vocabulary. The advantage of this approach over fully discrete methods is the ability to refine probability distributions given the neighboring distributions rather than noisy discrete tokens that accumulate discretization errors at each time step.

A.2 DETERMINISTIC VS. STOCHASTIC INTERPOLANTS

The linear interpolant [23, 32] defines a a deterministic flow $\psi_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = t\mathbf{x}_0 + (1-t)\mathbf{x}_1$ between a pair of fixed endpoints $(\mathbf{x}_0, \mathbf{x}_1)$. Optimal transport [43] further defines an optimal mapping $\pi(\mathbf{x}_0, \mathbf{x}_1)$ that minimizes a cost function $c(\mathbf{x}_0, \mathbf{x}_1)$— often a squared distance cost $c(\mathbf{x}_0, \mathbf{x}_1) = d^2(\mathbf{x}_0, \mathbf{x}_1)$—between paired endpoints. Although the deterministic perspective is optimal for tasks like matching trajectories [47], it lacks expressivity and diversity for *de novo* design tasks like protein or peptide-binder design. This approach also prevents the flow model from effectively learning to *redirect* specific token trajectories that do not reflect the data distribution during inference, given the sequence context.

By defining a *stochastic interpolant* with Gumbel-noise where each token has a small probability of being transformed into a distribution where the token with the highest probability does not match the true token during training, the model still needs to predict the clean distribution $\mathbf{x}_\theta(\mathbf{x}_t, t)$ or the target generating velocity field $u_t^\theta(\mathbf{x}_t)$ but with more ambiguity given that not all distributions are on the deterministically biased towards the target token. This pushes the model to place a greater weight on the global context of each token and learn dependencies across tokens to generate a valid, clean sequence despite the increased ambiguity. Furthermore, this approach injects path variability to improve generalization and exploration of diverse flows for *de novo* design tasks.

A.3 GUIDED FLOW MATCHING

A key limitation of current discrete flow matching techniques is the lack of training-free guidance strategies. Flow matching guidance [48, 19] is performed either with *classifier-based* or *classifier-free* guidance.

Classifier-Free Guidance. In classifier-free guided flow matching [48], the guided velocity field is obtained by training a guided flow model $u_t^\phi(\mathbf{x}|y)$ and an unconditional flow model $u_t^\theta(\mathbf{x})$ and taking the linear combination of the guided and unconditional velocities scaled by a parameter γ.

$$\tilde{u}_t^\theta(\mathbf{x}|y) = (1-\gamma)u_t^\theta(\mathbf{x}) + \gamma u_t^\theta(\mathbf{x}|y) \tag{25}$$

This strategy requires training an additional guided flow model on quality-labeled data, which is often scarce. Given that flow models require more training data than simple regression and classification models, classifier-based guidance is preferred for scalability.

Classifier-Based Guidance. In classifier-based guided flow matching [37], a *time-dependent* classifier $p_t^\phi(y|\mathbf{x}_t)$ that predicts a classifier score given noisy samples $\mathbf{x}_t$ separately from the unconditional generator. Then, we sample with a guided velocity field given by

$$u_t^{\theta,\phi}(\mathbf{x}_t) = u_t^\theta(\mathbf{x}_t) + \gamma \nabla_{\mathbf{x}_t} \log p_t^\phi(y|\mathbf{x}_t) \tag{26}$$

which requires projection back to the simplex for guided discrete flows. For simplex-based flows, this approach typically involves additional training of *noisy classifiers* that predict the classifier score given intermediate distributions over the simplex at each time step. Not only are these noisy classifiers less accurate than large pre-trained classifiers on clean sequences, but they also require extensive training, as all noise levels need to be included in the training task.

STGFlow overcomes these limitations by defining a guided flow velocity using the straight-through gradients of the scoring model on discrete sequences sampled with respect to the relaxed Gumbel-softmax probabilities. To ensure that the scores of sampled sequences are representative of the relaxed distribution, we sample M sequences and take the aggregate gradient as the guided velocity. This provides a modular training-free strategy for discrete flow matching guidance that conserves the probability mass constraint (Proof in Appendix E).

B Relation to Prior Simplex-Based Flow Matching Models

In this section, we discuss and compare Gumbel-Softmax FM with two related methods for discrete flow matching on the simplex: **Dirichlet Flow Matching** [38] and **Fisher Flow Matching** [14].

B.1 Dirichlet Flow Matching

The Dirichlet distribution is an extension of the Beta distribution $\mathcal{B}$ for multiple variables and models the probability of the next variable x being in one of V discrete categories given a parameter vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_V)$. Intuitively, it acts as a distribution of smooth categorical vectors $\mathbf{x} \in \Delta^{V-1}$ that lie on the probability simplex given that each category $i \in [1 \ldots V]$ was observed with *frequency* α_i. Increasing α_i for a given category i would increase the probability of sampling $\mathbf{x}$ near the ith vertex of the simplex. Dirichlet FM [38] defines the conditional probability path as

$$p_t(\mathbf{x}|\mathbf{x}_1 = \mathbf{e}_k) = \text{Dir}(\mathbf{x}; \vec{\alpha} = \mathbf{1} + t \cdot \mathbf{e}_k) = \frac{1}{\mathcal{B}(\alpha_1, \ldots, \alpha_V)} \prod_{i=1}^{V} x_{t,i}^{\alpha_i - 1} \tag{27}$$

At $t = 0$, the distribution reduces to a uniform prior over Δ^{V-1}, with an equal probability of sampling $\mathbf{x}$ near any vertex. As $t \to \infty$, α_k increases while α_j for all $j \neq k$ remain constant, so the probability density converges to the kth vertex. As shown in [38], this distribution satisfies the boundary constraints.

To compute the target vector field, we start with the following equation

$$u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) = -\tilde{I}_{x_{t,k}(t+1,V-1)} \frac{\mathcal{B}(t+1, V-1)}{(1 - x_{t,k})^{V-1} \cdot x_{t,k}} (\mathbf{e}_k - \mathbf{x}_t) \tag{28}$$

Similar to our approach, Dirichlet FM trains a denoising model by minimizing a negative log loss and computes the velocity field as the linear combination of the conditional velocity fields as in Equation 12.

Although the Dirichlet probability path provides support over the entire simplex at all time steps, it suffers from high variance during training due to the stochastic nature of sampling from the Dirichlet distribution. Since flow matching learns a mixture of conditional velocity fields, there exists inherent variability during inference. Our definition of a Gumbel-Softmax interpolant ensures straighter flow paths and lower variance during training, as Gumbel noise largely preserves the relative probabilities between categories.

B.2 FISHER FLOW MATCHING

Fisher FM [14] overcomes the instability of the Fisher-Rao metric at the vertices of the simplex via a sphere map $\varphi : \Delta^{V-1} \to \mathbb{S}_+^{V-1}$ where $\varphi(x) = \sqrt{x}$ that maps a point in the interior of the $(V-1)$-dimensional simplex to a point on the positive orthant of the $(V-1)$-dimensional hypersphere. The conditional velocity field $u_t(\mathbf{x}_t|\mathbf{x}_1)$ of the linear interpolant on the sphere is given by

$$\begin{aligned} \psi_t(\mathbf{x}_1) &= \exp_{\mathbf{x}_0}\left(t \log_{\mathbf{x}_0}(\mathbf{x}_1)\right) \\ u_t(\mathbf{x}_t|\mathbf{x}_1) &= \frac{\log_{\mathbf{x}_t}(\mathbf{x}_1)}{1-t} \end{aligned} \tag{29}$$

During inference, the parameterized velocity field $\tilde{u}_\theta(\mathbf{x}_t) \in \mathbb{R}^V$ is projected onto the tangent bundle of the hypersphere $\mathcal{T}_{\mathbf{x}_t}\mathbb{S}_+^V$ via the following mapping

$$u_t^\theta(\mathbf{x}_t) = \tilde{u}_\theta(\mathbf{x}_t) - \langle \mathbf{x}_t, \tilde{u}_\theta(\mathbf{x}_t)\rangle_2 \mathbf{x}_t \tag{30}$$

which minimizes the mean-squared error with the true conditional velocity field given by

$$\mathcal{L}_{\text{fisher}} = \mathbb{E}_{t\sim\mathcal{U}(0,1), p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \left\| u_\theta(t, \mathbf{x}_t) - \frac{\log_{\mathbf{x}_t}(\mathbf{x}_1)}{1-t} \right\|_{\mathbb{S}_+^V}^2 \tag{31}$$

Fisher FM addresses the high training variance of Dirichlet FM without the pathological properties of linear flows on the simplex by projecting the linear interpolant to the positive orthant of the V-dimensional hypersphere, which is isometric to the $(V-1)$-dimensional simplex. However, projecting velocity fields to and from the tangent space of the hypersphere can lead to inconsistencies when applying guidance methods. Empirically, we found that the Fisher FM exhibits significantly high validation MSE loss during training, especially for increasing simplex dimensions, suggesting that the parameterization easily overfits to training data and is not optimal for *de novo* design tasks such as protein generation or peptide design.

C FLOW MATCHING DERIVATIONS

C.1 DERIVING THE CONDITIONAL VELOCITY FIELD

We derive the conditional velocity field at a point $\mathbf{x}_t$ denoted as $u_t(\mathbf{x}|\mathbf{x}_1 = \mathbf{e}_i)$ by taking the derivative of the interpolant $\psi_t(\mathbf{x}_1 = \mathbf{e}_i)$ with respect to time t.

$$\begin{aligned} u_{t,i}(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{d}{dt}\psi_{t,i}(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) \\ &= \frac{d}{dt}\frac{\exp\left(\frac{\log \pi_i + g_i}{\tau_{\max}\exp(-\lambda t)}\right)}{\sum_{j=1}^V \exp\left(\frac{\log \pi_j + g_j}{\tau_{\max}\exp(-\lambda t)}\right)} \end{aligned} \tag{32}$$

Letting $z_i = \exp\left(\frac{\log \pi_i + g_i}{\tau_{\max}\exp(-\lambda t)}\right)$, we have

$$\begin{aligned} u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{d}{dt}\frac{\exp(z_i)}{\sum_{j=1}^V \exp(z_j)} \\ &= \frac{\left(\frac{d}{dt}\exp(z_i)\right)\left(\sum_{j=1}^V \exp(z_j)\right) - \exp(z_i)\left(\frac{d}{dt}\sum_{j=1}^V \exp(z_j)\right)}{\left(\sum_{j=1}^V \exp(z_j)\right)^2} \end{aligned} \tag{33}$$

First, we compute $\frac{d}{dt}\exp(z_i)$

$$\begin{aligned} \frac{d}{dt}\exp\left(\frac{\log \pi_i + g_i}{\tau_{\max}\exp(-\lambda t)}\right) &= \exp(z_i)\cdot\frac{d}{dt}\left(\frac{\log \pi_i + g_i}{\tau_{\max}\exp(-\lambda t)}\right) \\ &= \exp(z_i)\cdot\frac{\log \pi_i + g_i}{\tau_{\max}}\cdot\frac{d}{dt}\exp(\lambda t) \\ &= \exp(z_i)\cdot\frac{\log \pi_i + g_i}{\tau_{\max}}\cdot\lambda\exp(\lambda t) \end{aligned} \tag{34}$$

Then, we compute $\frac{d}{dt}\sum_j \exp(z_j)$

$$\begin{aligned}\frac{d}{dt}\sum_{j=1}^{V}\exp\left(\frac{\log\pi_j+g_j}{\tau_{\max}\exp(-\lambda t)}\right) &= \sum_{j=1}^{V}\frac{d}{dt}\exp\left(\frac{\log\pi_j+g_j}{\tau_{\max}\exp(-\lambda t)}\right)\\ &= \sum_{j=1}^{V}\left(\exp(z_j)\cdot\frac{\log\pi_j+g_j}{\tau_{\max}}\cdot\lambda\exp(\lambda t)\right)\end{aligned} \tag{35}$$

Then, substituting these terms back into the expression for u_t, we get

$$\begin{aligned}
&u_{t,i}(\mathbf{x}_t|\mathbf{x}_1=\mathbf{e}_k)\\
&= \frac{\left(\sum_{j=1}^{V}\exp\left(z_j\right)\right)\cdot\exp(z_i)\cdot\frac{\log\pi_i+g_i}{\tau_{\max}}\cdot\lambda\exp(\lambda t)-\exp\left(z_i\right)\cdot\sum_{j=1}^{V}\left(\exp(z_j)\cdot\frac{\log\pi_j+g_j}{\tau_{\max}}\cdot\lambda\exp(\lambda t)\right)}{\left(\sum_{j=1}^{V}\exp\left(z_j\right)\right)^2}\\
&= \frac{\exp(z_i)\cdot\lambda\exp(\lambda t)}{\tau_{\max}\left(\sum_{j=1}^{V}\exp\left(z_j\right)\right)^2}\left[\left(\sum_{j=1}^{V}\exp(z_j)\right)\cdot(\log\pi_i+g_i)-\sum_{j=1}^{V}\left(\exp(z_j)\cdot(\log\pi_j+g_j)\right)\right]\\
&= \frac{\exp(z_i)\cdot\lambda\exp(\lambda t)}{\tau_{\max}\left(\sum_{j=1}^{V}\exp\left(z_j\right)\right)^2}\left[\sum_{j=1}^{V}\exp(z_j)\Big((\log\pi_i+g_i)-(\log\pi_j+g_j)\Big)\right]\\
&= \frac{\exp(z_i)}{\sum_{j=1}^{V}\exp\left(z_j\right)}\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\left[\sum_{j=1}^{V}\left(\frac{\exp(z_j)}{\sum_{j'}\exp\left(z_j\right)}\cdot\Big((\log\pi_i+g_i)-(\log\pi_j+g_j)\Big)\right)\right]\\
&= \psi_{t,i}(\mathbf{x}_1)\cdot\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\left[\sum_{j=1}^{V}\left(\psi_{t,j}(\mathbf{x}_1)\cdot\Big((\log\pi_i+g_i)-(\log\pi_j+g_j)\Big)\right)\right]\\
&= \frac{\lambda}{\tau(t)}x_{t,i}\sum_{j=1}^{V}x_{t,j}\cdot\Big((\log\pi_i+g_i)-(\log\pi_j+g_j)\Big)
\end{aligned} \tag{36}$$

By our definition of the Gumbel-Softmax interpolant, the intermediate distributions during inference represent a mixture of learned conditional interpolants $\psi_t(\mathbf{x}_1)$ from the training data. Since the denoising model is trained to predict the true clean distribution, we can set the Gumbel-noise random variable in the conditional velocity fields to 0 during inference, as we want the velocity field to point toward the predicted denoised distribution.

Substituting in $\pi_i = \exp(\delta_{ik})$, we have

$$u_{t,i}(\mathbf{x}_t|\mathbf{x}_1=\mathbf{e}_k) = \frac{\lambda}{\tau(t)}x_{t,i}\sum_{j=1}^{V}x_{t,j}\cdot(\delta_{ik}-\delta_{jk})$$

Since $\delta_{ij}=1$ only when i is the index of the target token $i=k$ and 0 otherwise, the velocity field can be rewritten as

$$\begin{aligned}
u_{t,i}(\mathbf{x}_0|\mathbf{x}_1=\mathbf{e}_k) &= \begin{cases}\frac{\lambda}{\tau(t)}x_{t,i}\sum_{j=1}^{V}\left(x_{t,j}\cdot(1-\delta_{jk})\right) & i=k\\ \frac{\lambda}{\tau(t)}x_{t,i}\sum_{j=1}^{V}\left(x_{t,j}\cdot(-\delta_{jk})\right) & i\neq k\end{cases}\\
&= \begin{cases}\frac{\lambda\exp(\lambda t)}{\tau_{\max}}x_{t,i}\left(\sum_{j=1}^{V}x_{t,j}-\sum_{j=1}^{V}x_{t,j}\delta_{jk}\right) & i=k\\ \frac{\lambda\exp(\lambda t)}{\tau_{\max}}x_{t,i}\left(-\sum_{j=1}^{V}x_{t,j}\delta_{jk}\right) & i\neq k\end{cases}\\
&= \begin{cases}\frac{\lambda\exp(\lambda t)}{\tau_{\max}}x_{t,i}\left(1-x_{t,k}\right) & i=k\\ \frac{\lambda\exp(\lambda t)}{\tau_{\max}}x_{t,i}\left(-x_{t,k}\right) & i\neq k\end{cases}
\end{aligned} \tag{37}$$

Rewriting in vector form, we get

$$u_t(\mathbf{x}_t|\mathbf{x}_1=\mathbf{e}_k) = \frac{\lambda}{\tau(t)}x_{t,k}\left(\mathbf{e}_k-\mathbf{x}_t\right) \tag{38}$$

which points toward the target vertex $\mathbf{e}_k$.

C.2 PROOF OF CONTINUITY

Proposition 1. The proposed conditional vector field and conditional probability path satisfy the continuity equation and thus define a valid flow-matching trajectory in the interior of the simplex.

$$\frac{\partial}{\partial t} p_t(\mathbf{x}) = -\nabla \cdot (p_t(\mathbf{x}) u_t(\mathbf{x}_t)) \tag{39}$$

Proof of Proposition 1. During training, each clean sequence $\mathbf{x}_1$ is transformed into some noisy interpolant $\psi_t(\mathbf{x}_t)$ with a sampled Gumbel-noise vector $\mathbf{g} \sim \text{Gumbel}(0, 1)$. Therefore, we can rewrite the interpolant as a deterministic path conditioned on the one-hot distribution $\mathbf{x}_1$ and Gumbel-noise vector $\mathbf{g}$

$$\psi_t(\mathbf{x}_1) = \psi_t(\mathbf{x}_1, \mathbf{g}) = \text{SM}\left(\frac{\mathbf{x}_1 + \mathbf{g}}{\tau(t)}\right) \tag{40}$$

With this definition, we can define a deterministic probability path as the Dirac delta function along the interpolant $\mathbf{x}_t = \psi_t(\mathbf{x}_1)$ as

$$p_t(\mathbf{x}|\mathbf{x}_1) = \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \tag{41}$$

So, we can rewrite the continuity equation as

$$\begin{aligned} \frac{\partial}{\partial t} p_t(\mathbf{x}|\mathbf{x}_1) &= -\nabla \cdot \left(\delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1) \right) \\ &= -\nabla \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1) \end{aligned} \tag{42}$$

First, we will simplify the right-hand side (RHS) of the continuity equation. Taking the derivative with respect to t, we get

$$\frac{\partial}{\partial t} p_t(\mathbf{x}|\mathbf{x}_1) = \frac{\partial}{\partial t} \delta(\mathbf{x} - \psi_t(\mathbf{x}_t))$$

Taking the distributional derivative with an arbitrary test function $f(\mathbf{x})$ independent of t, we have

$$\begin{aligned} &= \int f(\mathbf{x}) \frac{\partial}{\partial t} \delta(\mathbf{x} - \psi_t(\mathbf{x}_t, \mathbf{g})) d\mathbf{x} \\ &= \frac{\partial}{\partial t} \int f(\mathbf{x}) \delta(\mathbf{x} - \psi_t(\mathbf{x}_t)) d\mathbf{x} \\ &= \frac{\partial}{\partial t} f(\psi_t(\mathbf{x}_1)) \end{aligned} \tag{43}$$

Since $\psi_t(\mathbf{x}_1) \in \mathbb{R}^V$, we apply the multivariable chain rule to get

$$\frac{\partial}{\partial t} p_t(\mathbf{x}|\mathbf{x}_1) = \nabla f(\psi_t(\mathbf{x}_1)) \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1) \tag{44}$$

Now, we integrate the left-hand side (LHS) of the continuity equation with an arbitrary test function.

$$\int f(\mathbf{x}) \left[-\nabla \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1) \right] d\mathbf{x} = -\left[\int f(\mathbf{x}) \nabla \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) d\mathbf{x} \right] \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1) \tag{45}$$

Using integration by parts, we can write the term inside the bracket as

$$\begin{aligned} \int_{-\infty}^{\infty} f(\mathbf{x}) \nabla \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) d\mathbf{x} &= \underbrace{f(\mathbf{x}) \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \Big|_{-\infty}^{\infty}}_{=0} - \int_{-\infty}^{\infty} \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \nabla f(\mathbf{x}) \\ &= -\int_{-\infty}^{\infty} \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \nabla f(\mathbf{x}) \end{aligned} \tag{46}$$

Substituting this back into the LHS, we get

$$\int f(\mathbf{x}) \left[-\nabla \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}) \right] d\mathbf{x} = - \left[- \int_{-\infty}^{\infty} \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \nabla f(\mathbf{x}) \right] \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1)$$
$$= \nabla f(\psi_t(\mathbf{x}_1)) \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1) \tag{47}$$

We have shown that both sides of the continuity equation produce the same expression when integrated against any arbitrary test function $f(\mathbf{x})$. So, we can conclude

$$\frac{\partial}{\partial t} p_t(\mathbf{x}|\mathbf{x}_1) = -\nabla \delta(\mathbf{x} - \psi_t(\mathbf{x}_1)) \cdot \frac{\partial}{\partial t} \psi_t(\mathbf{x}_1)$$
$$= \nabla \cdot (p_t(\mathbf{x}|\mathbf{x}_1) u_t(\mathbf{x}_t|\mathbf{x}_1)) \tag{48}$$

Now that we have shown the continuity equation holds for the conditional probability density and flow velocities, it follows that the continuity equation holds for the unconditional flow. Following the proof in [43], we have

$$\begin{aligned}
\frac{d}{dt} p_t(\mathbf{x}_t) &= \frac{d}{dt} \int_{\mathbf{x}_1} p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 \\
&= \int_{\mathbf{x}_1} \frac{d}{dt} p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 \\
&= \int_{\mathbf{x}_1} -\nabla \cdot \Big(p_t(\mathbf{x}_t|\mathbf{x}_1) u_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) \Big) d\mathbf{x}_1 && \text{(substitute conditional continuity)} \\
&= -\nabla \cdot \left(\int_{\mathbf{x}_1} p_t(\mathbf{x}_t|\mathbf{x}_1) u_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 \right) \\
&= -\nabla \cdot (p_t(\mathbf{x}_t) u_t(\mathbf{x}_t))
\end{aligned} \tag{49}$$

which concludes the proof.

C.3 Proof of Flow Matching Propositions

Proposition 1 (Probability Mass Conservation) The conditional velocity field preserves probability mass and lies on the tangent bundle at point $\mathbf{x}_t$ on the simplex $\mathcal{T}_{\mathbf{x}_t} \Delta^{V-1} = \{u_t \in \mathbb{R}^V | \langle \mathbf{1}, u_t \rangle = 0\}$.

Proof of Proposition 1. We show that the conditional velocity field derived from the Gumbel-Softmax interpolant preserves probability mass such that

$$\sum_{i=1}^{V} u_{t,i}(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) = 0 \tag{50}$$

Summing up the velocities for all $i \in [1 \dots V]$, we have

$$\begin{aligned}
\sum_{i=1}^{V} u_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) &= \sum_{i=1}^{V} \left[\frac{\lambda}{\tau(t)} x_{t,i} \sum_{j=1}^{V} x_{t,j} \cdot \Big((\log \pi_i + g_i) - (\log \pi_j + g_j) \Big) \right] \\
&= \frac{\lambda}{\tau(t)} \sum_{i=1}^{V} \left[x_{t,i} \left[\sum_{j=1}^{V} x_{t,j} (\log \pi_i + g_i) - \sum_{j=1}^{V} x_{t,j} (\log \pi_j + g_j) \right] \right] \\
&= \frac{\lambda}{\tau(t)} \sum_{i=1}^{V} \left[x_{t,i} \left[(\log \pi_i + g_i) \sum_{j=1}^{V} x_{t,j} - \sum_{j=1}^{V} x_{t,j} (\log \pi_j + g_j) \right] \right] \\
&= \frac{\lambda}{\tau(t)} \left[\sum_{i=1}^{V} x_{t,i} (\log \pi_i + g_i) - \sum_{i=1}^{V} x_{t,i} \sum_{j=1}^{V} x_{t,j} (\log \pi_j + g_j) \right] \\
&= \frac{\lambda}{\tau(t)} \left[\sum_{i=1}^{V} x_{t,i} (\log \pi_i + g_i) - \sum_{j=1}^{V} x_{t,j} (\log \pi_j + g_j) \right] \\
&= 0
\end{aligned} \tag{51}$$

which proves that our velocity field always preserves the probability mass t.

Proposition 3. (Valid Flow Matching Loss) *If $p_t(\mathbf{x}_t) > 0$ for all $\mathbf{x}_t \in \mathbb{R}^d$ and $t \in [0, 1]$, then the gradients of the flow matching loss and the Gumbel-Softmax FM loss are equal up to a constant not dependent on θ such that $\nabla_\theta \mathcal{L}_{FM} = \nabla_\theta \mathcal{L}_{gumbel}$*

Proof of Proposition 3. We can rewrite the conditional velocity field derived in Appendix C.1 as

$$\begin{aligned} u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{\lambda}{\tau(t)} x_{t,k} \left(\mathbf{e}_k - \mathbf{x}_t\right) \\ &= \frac{\lambda}{\tau(t)} \sum_{i=1}^{V} x_{t,i} \left(\mathbf{e}_i - \mathbf{x}_t\right) \langle \mathbf{e}_i, \mathbf{x}_1 \rangle \end{aligned} \tag{52}$$

Furthermore, the predicted velocity field is given by

$$\begin{aligned} u_t^\theta(\mathbf{x}_t) &= \sum_{i=1}^{V} u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_i) \langle \mathbf{e}_i, \mathbf{x}_\theta \rangle \\ &= \frac{\lambda}{\tau(t)} \sum_{i=1}^{V} x_{t,i}(\mathbf{e}_i - \mathbf{x}_t) \langle \mathbf{e}_i, \mathbf{x}_\theta \rangle \end{aligned} \tag{53}$$

Substituting the velocity field expressions into the flow-matching loss, we obtain

$$\begin{aligned} &\mathbb{E}_{p_t(\mathbf{x}_t)} \|u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t)\|^2 \\ &= \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \frac{\lambda}{\tau(t)} \sum_{i=1}^{V} x_{t,i} \left(\mathbf{e}_i - \mathbf{x}_t\right) \langle \mathbf{e}_i, \mathbf{x}_1 \rangle - \frac{\lambda}{\tau(t)} \sum_{i=1}^{V} x_{t,i}(\mathbf{e}_i - \mathbf{x}_t) \langle \mathbf{e}_i, \mathbf{x}_\theta \rangle \right\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \sum_{i=1}^{V} \left[x_{t,i} \left(\mathbf{e}_i - \mathbf{x}_t\right) \langle \mathbf{e}_i, \mathbf{x}_1 \rangle - x_{t,i}(\mathbf{e}_i - \mathbf{x}_t) \langle \mathbf{e}_i, \mathbf{x}_\theta \rangle \right] \right\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \sum_{i=1}^{V} x_{t,i} \left(\mathbf{e}_i - \mathbf{x}_t\right) \left[\langle \mathbf{e}_i, \mathbf{x}_1 \rangle - \langle \mathbf{e}_i, \mathbf{x}_\theta \rangle \right] \right\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \sum_{i=1}^{V} x_{t,i} \left(\mathbf{e}_i - \mathbf{x}_t\right) \langle \mathbf{e}_i, \mathbf{x}_1 - \mathbf{x}_\theta \rangle \right\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \sum_{i=1}^{V} x_{t,i} \left(\mathbf{e}_i - \mathbf{x}_t\right) (\mathbf{x}_1 - \mathbf{x}_\theta)_i \right\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \left\| \sum_{i=1}^{V} x_{t,i} (\mathbf{x}_1 - \mathbf{x}_\theta)_i \mathbf{e}_i - \sum_{i=1}^{V} x_{t,i} (\mathbf{x}_1 - \mathbf{x}_\theta)_i \mathbf{x}_t \right\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) - \mathbf{x}_t \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle\|^2 \end{aligned} \tag{54}$$

The remainder of the proof extends that of [23, 43], which proved that the conditional flow matching loss $\nabla_\theta \mathcal{L}_{\text{CFM}} = \nabla_\theta \mathcal{L}_{\text{FM}}$ under similar constraints.

First, we further expand the conditional flow-matching loss as follows

$$\begin{aligned} &\mathbb{E}_{p_t(\mathbf{x}_t)} \|u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t, t)\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) - \mathbf{x}_t \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_t \odot \mathbf{x}_1 - \mathbf{x}_t \odot \mathbf{x}_\theta - \mathbf{x}_t \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_t \odot \mathbf{x}_1 - \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle)\|^2 \\ &= \frac{\lambda^2}{\tau(t)^2} \mathbb{E}_{p_t(\mathbf{x}_t)} \left[\|\mathbf{x}_t \odot \mathbf{x}_1\|^2 - 2 \Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle) \Big\rangle + \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle)\|^2 \right] \end{aligned}$$

Then, taking the gradient with respect to θ, we have

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} \|u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t, t)\|^2 \\
&= \frac{\lambda^2}{\tau(t)^2} \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} \left[\|\mathbf{x}_t \odot \mathbf{x}_1\|^2 - 2\Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle) \Big\rangle + \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle)\|^2 \right] \\
&= \frac{\lambda^2}{\tau(t)^2} \left[-2\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} \Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle) \Big\rangle + \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} \Big\| \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle) \Big\|^2 \right]
\end{aligned} \tag{55}$$

Now, we rewrite $\mathbf{x}_1$ as the expectation over noisy samples $\mathbf{x}_t$ learned by the model. By Bayes' theorem, we have

$$p(\mathbf{x}_1|\mathbf{x}_1) = \frac{p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1)}{p_t(\mathbf{x}_t)} \tag{56}$$

Then, defining $\mathbf{x}_1$ as an expectation over $p_t(\mathbf{x}_t)$, we get

$$\begin{aligned}
\mathbf{x}_1 &= \mathbb{E}_{p(\mathbf{x}_1|\mathbf{x}_t)} [\mathbf{x}_1] \\
&= \int_{\mathbf{x}_1} \mathbf{x}_1 p(\mathbf{x}_1|\mathbf{x}_t) d\mathbf{x}_1 \\
&= \int_{\mathbf{x}_1} \mathbf{x}_1 \frac{p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1)}{p_t(\mathbf{x}_t)} d\mathbf{x}_1
\end{aligned} \tag{57}$$

Now, we substitute this into the first expectation in the gradient to get

$$\begin{aligned}
&\mathbb{E}_{p_t(\mathbf{x}_t)} \Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle) \Big\rangle \\
&= \int_{\mathbf{x}_t} \left\langle \mathbf{x}_t \odot \int_{\mathbf{x}_1} \mathbf{x}_1 \frac{p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1)}{p_t(\mathbf{x}_t)} d\mathbf{x}_1, \mathbf{x}_t \odot \left(\mathbf{x}_\theta - \left\langle \mathbf{x}_t \int_{\mathbf{x}_1} \mathbf{x}_1 \frac{p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1)}{p_t(\mathbf{x}_t)} d\mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \right\rangle \right) \right\rangle p_t(\mathbf{x}_t) d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \left\langle \mathbf{x}_t \odot \int_{\mathbf{x}_1} \mathbf{x}_1 p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1, \mathbf{x}_t \odot \left(\mathbf{x}_\theta - \left\langle \mathbf{x}_t \int_{\mathbf{x}_1} \mathbf{x}_1 p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \right\rangle \right) \right\rangle d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \left\langle \mathbf{x}_t \odot \int_{\mathbf{x}_1} \mathbf{x}_1 p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1, \mathbf{x}_t \odot \left(\mathbf{x}_\theta - \int_{\mathbf{x}_1} \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle \, p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 \right) \right\rangle d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \int_{\mathbf{x}_1} \Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle) \Big\rangle p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 d\mathbf{x}_t \\
&= \int_{\mathbf{x}_1} \int_{\mathbf{x}_t} \Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle) \Big\rangle p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_t d\mathbf{x}_1 \\
&= \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta \rangle) \Big\rangle
\end{aligned} \tag{58}$$

where we use the linearity properties of integration.

Following similar logic, we have

$$\begin{aligned}
&\mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle)\|^2 \\
&= \int_{\mathbf{x}_t} \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle)\|^2 p_t(\mathbf{x}_t) d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \left\| \mathbf{x}_t \odot \left(\mathbf{x}_\theta - \left\langle \mathbf{x}_t \int_{\mathbf{x}_1} \mathbf{x}_1 \frac{p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1)}{p_t(\mathbf{x}_t)} d\mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \right\rangle \right) \right\|^2 p_t(\mathbf{x}_t) d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \int_{\mathbf{x}_1} \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle)\|^2 \, p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 d\mathbf{x}_t \\
&= \int_{\mathbf{x}_1} \int_{\mathbf{x}_t} \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle)\|^2 \, p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_t d\mathbf{x}_1 \\
&= \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \|\mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t \mathbf{x}_1, -\mathbf{x}_t \mathbf{x}_\theta \rangle)\|^2
\end{aligned} \tag{59}$$

using the fact that the squared norm can be expressed as a bilinear inner product.

Substituting these terms back into the gradient of the flow-matching loss, we get

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} \|u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t, t)\|^2 \\
&= \frac{\lambda^2}{\tau(t)^2}\Big[-2\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)}\Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta\rangle)\Big\rangle + \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)}\Big\| \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t\mathbf{x}_1, -\mathbf{x}_t\mathbf{x}_\theta\rangle)\Big\|^2\Big] \\
&= \frac{\lambda^2}{\tau(t)^2}\Big[-2\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)}\Big\langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta\rangle)\Big\rangle + \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)}\Big\| \mathbf{x}_t \odot (\mathbf{x}_\theta - \langle \mathbf{x}_t\mathbf{x}_1, -\mathbf{x}_t\mathbf{x}_\theta\rangle)\Big\|^2\Big] \\
&= \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)}\left[\frac{\lambda^2}{\tau(t)^2}\|\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) - \mathbf{x}_t\langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta\rangle\|^2\right] \\
&= \frac{\lambda^2}{\tau(t)^2}\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)}\|\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) - \mathbf{x}_t\langle \mathbf{x}_t, \mathbf{x}_1 - \mathbf{x}_\theta\rangle\|^2 \qquad (60)
\end{aligned}$$

which concludes the proof that $\nabla_\theta \mathcal{L}_{\text{gumbel}} = \nabla_\theta \mathcal{L}_{\text{FM}}$.

D Score Matching Derivations

D.1 Derivation of the Score Function

We start by showing that the score function of the marginal probability density $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is proportional to the conditional probability density $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1)$ given that $p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_1 \sim p_1(\mathbf{x}_1)}\big[p_t(\mathbf{x}_t|\mathbf{x}_1)\big]$.

Taking the gradient of the marginal log probability density and substituting in the definition of $p_t(\mathbf{x}_t)$, we have

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) &= \frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t)}{p_t(\mathbf{x}_t)} \\
&= \frac{\nabla_{\mathbf{x}_t} \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}}\big[p_t(\mathbf{x}_t|\mathbf{x}_1)\big]}{p_t(\mathbf{x}_t)} \\
&= \frac{\nabla_{\mathbf{x}_t} \int_{\mathbf{x}_1} \big[p(\mathbf{x}_1)p_t(\mathbf{x}_t|\mathbf{x}_1)\big] d\mathbf{x}_1}{p_t(\mathbf{x}_t)} \\
&= \frac{\int_{\mathbf{x}_1} p(\mathbf{x}_1)\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t|\mathbf{x}_1) d\mathbf{x}_1}{p_t(\mathbf{x}_t)} \\
&= \frac{\int_{\mathbf{x}_1} p(\mathbf{x}_1)p_t(\mathbf{x}_t|\mathbf{x}_1)\frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t|\mathbf{x}_1)}{p_t(\mathbf{x}_t|\mathbf{x}_1)} d\mathbf{x}_1}{p_t(\mathbf{x}_t)} \\
&= \int_{\mathbf{x}_1} \underbrace{\frac{p_t(\mathbf{x}_t|\mathbf{x}_1)p(\mathbf{x}_1)}{p_t(\mathbf{x}_t)}}_{=p_t(\mathbf{x}_1|\mathbf{x}_t)} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1) d\mathbf{x}_1 \\
&= \mathbb{E}_{\mathbf{x}_1 \sim p_t(\mathbf{x}_1|\mathbf{x}_t)}\left[\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1)\right] \qquad (61)
\end{aligned}$$

which proves that with the perfect model such that $p_t(\mathbf{x}_1) = p(\mathbf{x}_1|\mathbf{x}_t)$, the gradient of the marginal log-probability density is exactly the expectation of the conditional log-probability density over the training data $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_1 \sim p_1(\mathbf{x}_1)}\left[\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1)\right]$.

Theorem 2. The gradient of the log-probability density of the ExpConcrete distribution is given by

$$\nabla_{x_{t,i}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = \tau(t) - \tau(t)V \cdot \text{SM}\Big(\delta_{ik} - \tau(t)x_{t,i}\Big) \qquad (62)$$

Proof of Theorem 2. First, we start by defining the probability density of the ExpConcrete distribution. From [26], integrating out the Gumbel-noise random variable, we have

$$p_t(\mathbf{x}) = (V-1)!\tau^{V-1}\left(\sum_{i=1}^{V} \pi_j \exp(-\tau x_{t,j})\right)\left(\prod_{i=1}^{V} \pi_i \exp(-\tau x_{t,i})\right) \qquad (63)$$

where $x_{t,i}$ is defined as a logit from the EXPCONCRETE distribution

$$x_{t,i} = \frac{\log \pi_i + g_i}{\tau} - \log \sum_{j=1}^{V} \exp\left(\frac{\log \pi_j + g_j}{\tau}\right) \tag{64}$$

Taking the logarithm of the probability path, we have

$$\begin{aligned}
\log p_t(\mathbf{x}_t|\mathbf{x}_1) &= \log[(V-1)!] + (V-1)\log\tau + \log\left(\prod_{i=1}^{V} \pi_i \exp(-\tau x_{t,i})\right) - V \log \sum_{j=1}^{V} \pi_j \exp(-\tau x_{t,j}) \\
&= \log[(V-1)!] + (V-1)\log\tau + \sum_{i=1}^{V} \log\left(\pi_i \exp(-\tau x_{t,i})\right) - V \log \sum_{j=1}^{V} \exp\left(\log\left(\pi_j \exp(-\tau x_{t,j})\right)\right) \\
&= \log[(V-1)!] + (V-1)\log\tau + \sum_{i=1}^{V} (\log \pi_i - \tau x_{t,i}) - V \log \sum_{j=1}^{V} \exp\Big(\log \pi_j - \tau x_{t,j}\Big) \\
&= \log[(V-1)!] + (V-1)\log\tau + \sum_{i=1}^{V} \log \pi_i - \sum_{i=1}^{V} \tau x_{t,i} - V \log \sum_{j=1}^{V} \exp\Big(\log \pi_j - \tau x_{t,j}\Big)
\end{aligned} \tag{65}$$

Then, diferentiating with respect to the logit of a single token $x_{t,j}$, we get

$$\begin{aligned}
\nabla_{x_{t,j}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) &= -\nabla_{x_{t,i}} \sum_{i=1}^{V} \tau x_{t,i} - \nabla_{x_{t,i}} V \log \sum_{j=1}^{V} \exp\Big(\log \pi_j - \tau x_{t,j}\Big) \\
&= -\tau - V\left(\frac{1}{\sum_{j=1}^{V} \exp(\log \pi_j - \tau x_{t,j})}\right) \exp(\log \pi_i - \tau x_{t,i})(-\tau) \\
&= -\tau + \tau V \left(\frac{\exp(\log \pi_i - \tau x_{t,i})}{\sum_{i=1}^{V} \exp(\log \pi_j - \tau x_j)}\right) \\
&= -\tau + \tau V \cdot \mathrm{SM}\Big(\log \pi_i - \tau x_{t,i}\Big)
\end{aligned} \tag{66}$$

Introducing time-dependence with $\tau(t) = \tau_{\max} \exp(-\lambda t)$ and target token dependence with $\pi_i = \exp(\delta_{ik})$, we have

$$\nabla_{x_{t,i}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = \tau(t) - \tau(t) V \cdot \mathrm{SM}\Big(\delta_{ik} - \tau(t) x_{t,i}\Big) \tag{67}$$

D.2 PROOF OF SCORE MATCHING PROPOSITIONS

Proposition 4. The gradient of the EXPCONCRETE log-probability density is proportional to the gradient of the Gumbel-softmax log-probability density such that $\nabla^{\mathrm{GS}}_{x_{t,j}} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1) \propto \nabla^{\mathrm{ExpConcrete}}_{x_{t,j}} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1)$.

Proof of Proposition 4. As derived in [26], the explicit probability density of the Gumbel-Softmax distribution is defined as

$$p(\mathbf{x}) = (V-1)!\tau^{V-1} \left(\sum_{i=1}^{V} \frac{\pi_i}{x_{t,i}^{\tau}}\right)^{-V} \prod_{i=1}^{V} \left(\frac{\pi_i}{x_{t,i}^{\tau+1}}\right) \tag{68}$$

We now derive the log-probability density of the Gumbel-Softmax distribution as

$$\begin{aligned}
\log p(\mathbf{x}) &= \log[(V-1)!] + (V-1)\log\tau - V \log \sum_{i=1}^{V} \frac{\pi_i}{x_{t,i}^{\tau}} + \sum_{i=1}^{V} \log\left(\frac{\pi_i}{x_{t,i}^{\tau+1}}\right) \\
&= \log[(V-1)!] + (V-1)\log\tau - V \log \sum_{i=1}^{V} \frac{\pi_i}{x_{t,i}^{\tau}} + \sum_{i=1}^{V} \log(\pi_i) - (\tau+1) \sum_{i=1}^{V} \log(x_{t,i})
\end{aligned} \tag{69}$$

Taking the gradient with respect to a single token $x_{t,j}$, we have

$$\begin{aligned}
\nabla^{\text{GS}}_{x_{t,j}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) &= \nabla_{x_{t,j}} \left(-V \log \sum_{i=1}^{V} \frac{\pi_i}{x_{t,i}^{\tau}} \right) - \nabla_{x_{t,j}} \left((\tau+1) \sum_{i=1}^{V} \log(x_{t,i}) \right) \\
&= -V \left(\frac{1}{\sum_{i=1}^{V} \frac{\pi_i}{x_{t,i}^{\tau}}} \right) \left(\frac{-\pi_j \tau}{x_{t,j}^{\tau+1}} \right) - \frac{\tau+1}{x_{t,j}} \\
&= \frac{\tau V}{x_{t,j}} \left(\frac{\pi_j x_{t,j}^{-\tau}}{\sum_{i=1}^{V} \pi_i x_{t,i}^{-\tau}} \right) - \frac{\tau+1}{x_{t,j}} \\
&= \frac{\tau V}{x_{t,j}} \left(\frac{\exp(\log(\pi_j x_{t,j}^{-\tau}))}{\sum_{i=1}^{V} \exp(\log(\pi_i x_{t,i}^{-\tau}))} \right) - \frac{\tau+1}{x_{t,j}} \\
&= \frac{\tau V}{x_{t,j}} \left(\frac{\exp(\log \pi_j - \tau x_{t,j})}{\sum_{i=1}^{V} \exp(\log \pi_i - \tau x_{t,i})} \right) - \frac{\tau+1}{x_{t,j}} \\
&= \frac{\tau V}{x_{t,j}} \text{SM}\big(\log \pi_i - \tau x_{t,i} \big) - \frac{\tau+1}{x_{t,j}} \\
&= \frac{1}{x_{t,j}} \Big(-\tau + \tau V \cdot \text{SM}\big(\log \pi_i - \tau x_{t,i} \big) \Big) - \frac{1}{x_{t,j}} \\
&= \frac{1}{x_{t,j}} \left(\nabla^{\text{ExpConcrete}}_{x_{t,j}} \log p_t(\mathbf{x}_t|\mathbf{x}_1) \right) - \frac{1}{x_{t,j}}
\end{aligned} \tag{70}$$

Therefore, we show that the gradients of the Gumbel-Softmax and EXPCONCRETE distributions are proportional to each other. Furthermore, we derive that the score of Gumbel-Softmax distribution further amplifies the scores for tokens with low probabilities by dividing by $x_{t,j}$ and subtracting $x_{t,j}^{-1}$.

E STRAIGHT-THROUGH GUIDED FLOW DERIVATIONS

Proposition 5. (Probability Mass Conservation of Straight-Through Gradient) The straight through gradient $\nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m})$ preserves probability mass and lies on the tangent bundle at point $\mathbf{x}_t$ on the simplex $\mathcal{T}_{\mathbf{x}_t}\Delta^{V-1} = \{\nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) \in \mathbb{R}^V | \langle \mathbf{1}, \nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) \rangle = 0\}$.

Proof of Proposition 5. First, we recall our definition of the straight-through gradient of the classifier score $p_\phi(y|\tilde{\mathbf{x}}_{1,m})$ as

$$\nabla_{x_{t,i}} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) = \begin{cases} \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \cdot \big[\text{SM}(x_{t,i})\,(1 - \text{SM}(x_{t,k}))\,\big] & i = k \\ \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \cdot \big[-\text{SM}(x_{t,i})\text{SM}(x_{t,k}) \big] & i \neq k \end{cases}$$

Taking the sum over the simplex dimensions, we have

$$\begin{aligned}
\sum_{i=1}^{V} \nabla_{x_{t,i}} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) &= \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \left[\text{SM}(x_{t,k})\,(1 - \text{SM}(x_{t,k})) - \sum_{i \neq k} \text{SM}(x_{t,i})\text{SM}(x_{t,k}) \right] \\
&= \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \left[\text{SM}(x_{t,k})\,(1 - \text{SM}(x_{t,k})) - \text{SM}(x_{t,k}) \sum_{i \neq k} \text{SM}(x_{t,i}) \right] \\
&= \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \Big[\text{SM}(x_{t,k})\,(1 - \text{SM}(x_{t,k})) - \text{SM}(x_{t,k})\,(1 - \text{SM}(x_{t,k})) \Big] \\
&= 0
\end{aligned}$$

which concludes the proof. In addition, it follows that the sum of straight-through gradients also preserves probability mass and lies on the tangent space of the simplex at any point.

F MODEL ARCHITECTURE

F.1 DIFFUSION TRANSFORMER

To parameterize our flow and score matching models for the protein and peptide sequence generation tasks, we leverage the Diffusion Transformer (DiT) architecture [30] which integrates time condition-

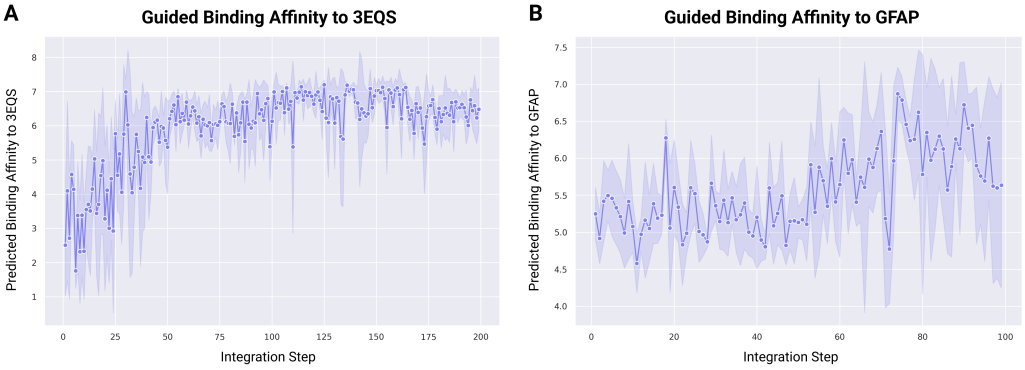

Figure 4: **Predicted binding-affinity scores over iteration of Gumbel-Softmax FM guided with STGFlow for target-binding peptide generation.** The predicted binding affinity is the mean regression scores across the M discrete sequences sampled at each integration step. The gradients of the scores are used to compute the guided velocity.

ing with adaptive layer norm (adaLN) and positional information with Rotary Positional Embeddings (RoPE) [40]. Our model consists of 32 DiT blocks, 16 attention heads, a hidden dimension of 1024, and dropout of 0.1.

Table 3: Diffusion Transformer Architecture

Layers	**Input Dimension**	**Output Dimension**
Sequence Distribution Embedding Module	vocab size	1024
Feed-Forward + GeLU	vocab size	1024
DiT Blocks $\times 32$		
Adaptive Layer Norm (time conditioning)	1024	1024
Multi-Head Self-Attention ($h = 16$)		
+ Rotary Positional Embeddings	1024	1024
Dropout + Residual	1024	1024
Adaptive Layer Norm (time conditioning)	1024	1024
FFN + GeLU	1024	1024
DiT Final Block		
Adaptive Layer Norm (time conditioning)	1024	1024
Linear	1024	vocab size

F.2 Peptide-Binding Affinity Classifier

We trained a multi-head cross-attention network with ESM-2 650M [22] protein and peptide sequence embeddings to predict the binding affinity of a peptide to a protein sequence. We trained on 1781 sequences from the PepLand [46] protein-peptide binding dataset containing the protein-target sequence, peptide sequence, and the experimentally-validated $K_d/K_i/IC50$ binding affinity score, where higher values indicate stronger binding.

In addition to the normalized binding affinity scores, we also classified affinities into three categories: low (< 6.0), medium ($6.0 - 7.5$), and tight (≥ 7.5) binding, with thresholds based on mean and Q3 quantile from the data distribution. The combined classification and regression approach helped the model better capture relationships between protein embeddings and binding affinities. Data was split in a 0.8/0.2 ratio with stratification preserving the score distribution.

We used OPTUNA [3] for hyperparameter optimization, tracking validation correlation, and F1 scores across 10 trials, resulting in an optimal learning rate of $3.84e-05$ and a dropout rate of 0.15. We retrain the whole classifier (Table 4) with the optimized set of parameters. After training for 50 epochs with early stopping based on validation Spearman correlation, the model achieved a Spearman

correlation of 0.96 on training data and 0.64 on validation data, with F1 scores of 0.97 and 0.61, respectively.

Table 4: Peptide-Binding Affinity Classifier

Layers	**Protein Dimension**	**Peptide Dimension**
Embedding Module	1280	1280
CNN Layers $\times 3$ (Kernel Sizes: 3,5,7)	$(1280, L)$	$(64 \times 3, L)$ per kernel
ReLU Activation	$(64, L)$ per kernel	$(64, L)$ per kernel
Global Pooling (Max + Avg)	$(64 \times 3, L)$	$64 \times 3 \times 2$
Linear Layer	384	384
Layer Norm	384	384
Cross-Attention $\times 4$		
Multi-Head Attention ($h = 8$)	384	384
Linear Layer	2048	2048
ReLU	2048	2048
Dropout	2048	2048
Linear Layer	384	384
Shared Prediction Head		
Linear Layer	1024	
ReLU	1024	
Dropout	1024	
Regression Head	1	

G Additional Experiments

G.1 Simplex-Dimension Toy Experiment

We reproduce the experimental setup of the toy experiment in Davis et al. [14]. We train $100,000$ sequences sampled from a randomly generated distribution over the $(K-1)$-dimensional simplex for $K = \{20, 40, 60, 80, 100, 120, 140, 160, 512\}$. We extend the experiment to dimension 512 to evaluate performance in a higher simplex dimension.

For the model architecture, we follow Stark et al. [38] and parameterize all benchmark models with a 5-layer CNN with approximately 1M parameters that vary slightly with simplex dimension. After 50K steps, we evaluate the KL divergence $\mathrm{KL}(\tilde{q} \| p_{\text{data}})$ where $\tilde{q}$ is the normalized distribution from 51.2K sequences generated by the model and p_{data} is the distribution from which the training data was sampled.

Table 5: **KL divergences of toy experiment for increasing simplex dimensions compared to benchmark models.** The sequence length is set to a constant of 4 across all experiments. The toy models are trained on 100K sequences from a random distribution. KL divergence is evaluated for 51.2K sequences after 50K training steps.

Simplex Dimension K	20	40	60	80	100	120	140	160	512
Linear FM	0.013	0.046	0.070	0.100	0.114	0.112	0.156	0.146	0.479
Dirichlet FM	0.007	0.017	0.032	0.035	0.028	0.024	0.039	0.053	0.554
Fisher FM (Optimal Transport)	0.0004	0.007	0.007	0.007	0.008	0.043	0.013	0.013	0.036
Gumbel-Softmax FM (Ours)	0.029	0.027	0.025	0.027	0.030	0.029	0.035	0.038	0.048

G.2 *De Novo* Protein Sequence Design

Next, we evaluate the quality of unconditionally-generated *de novo* protein sequences with Gumbel-Softmax SM and Gumbel-Softmax FM. Despite operating in the continuous simplex space with a considerably smaller backbone model, we demonstrate competitive generative quality compared to discrete diffusion and autoregressive baselines.

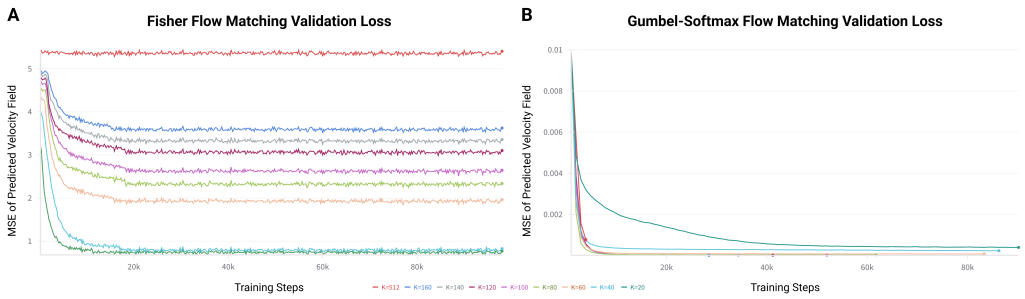

Figure 5: **Validation MSE loss over training step of simplex-dimension toy experiment.** Fisher FM exhibits significantly higher validation MSE loss during training than Gumbel-Softmax FM despite the same loss calculation, suggesting that the parameterization easily overfits to training data.

Table 6: **Evaluation metrics for generative quality of protein sequences.** Metrics were calculated on 100 unconditionally generated sequences from each model, including EvoDiff and ProtGPT2. The arrow indicates whether (↑) or (↓) values are better.

Model	**Params** (↓)	**pLDDT** (↑)	**pTM** (↑)	**pAE** (↓)	**Entropy** (↑)	**Diversity** (%) (↑)
Test Dataset (random 1000)	-	74.00	0.63	12.99	4.0	71.8
EvoDiff	640M	31.84	0.21	24.76	4.05	93.2
ProtGPT2	738M	54.92	0.41	19.39	3.85	70.9
ProGen2-small	151M	49.38	0.28	23.38	2.55	89.3
Gumbel-Softmax Flow Matching (Ours)	198M	52.54	0.27	16.67	3.41	86.1
Gumbel-Softmax Score Matching (Ours)	198M	49.40	0.29	15.71	3.37	82.5

Setup. Given the larger vocabulary size of protein sequences, we compared both the performance of Gumbel-Softmax FM and Gumbel-Softmax SM for this task. For both models, we applied the Gumbel-Softmax transformation with varying temperatures $\tau(t)$ for time steps $t \sim \mathcal{U}(0, 1)$ and $\tau_{\max} = 10.0$. The decay rates were set to $\lambda = 3.0$ for both models, and the noise scale was set to $\beta = 2.0$. The models were trained following Algorithm 1 for Gumbel-Softmax FM and 3 for Gumbel-Softmax SM. Sampling was performed following Algorithm 2 and Algorithm 4.

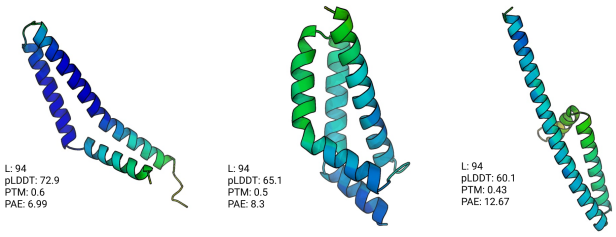

Figure 6: **Predicted structures of *de novo* generated proteins from Gumbel-Softmax FM.** The structures, pLDDT, pAE, and pTM scores are predicted with ESMFold [22]

Training. We collected 68M Uniref50 and 207M OMG_PROT50 data [41, 13]. A total of 275M protein sequences were first clustered to remove singletons using MMseqs2 linclust [39] (parameters set to `-min-seq-id 0.5 -c 0.9 -cov-mode 1`). We keep the sequences between lengths of 20 to 2500 and entries with only wild-type residues to avoid effects from outliers. The singleton sequences are removed. The resulting representative sequences undergo random 0.8/0.1/0.1 data splitting. We trained for 5 epochs on 7 NVIDIA A100 GPUs.

Results. We compare the quality of our protein generation method against state-of-the-art *de novo* protein sequence generation models including the discrete diffusion model EvoDiff [4], large language model ProtGPT2 [16], and the autoregressive model ProGen2-small [28]. For 100 unconditionally generated sequences per model, we compute the pLDDT, pTM, and pAE scores using ESMFold [21] as well as the token entropy and sequence diversity. Additional details on evaluation metrics are given in Appendix H.2. BLASTp runs for the proteins we generated indicate no homologous hits, highlighting again the novelty of the proteins we generated and indicating that our model is

not subsampling from known homologous protein sequences. As summarized in Table 6, both Gumbel-Softmax SM and Gumbel-Softmax FM produce proteins with comparable pLDDT, pTM, and pAE scores to discrete baselines without significantly compromising sequence entropy and diversity. We believe further optimization of hyperparameters, leveraging informative priors, or functional/structural guidance would improve the generative quality of Gumbel-Softmax FM.

H EXPERIMENTAL DETAILS

H.1 HYPERPARAMETER SELECTION

Maximum Temperature τ_{max}. The maximum temperature controls the uniformity of the probability distribution at $t = 0$ when $\exp(-\lambda t) = 1$. Theoretically, the probability distribution is fully uniform $\psi_0(\mathbf{x}_t|\mathbf{x}_1) = \frac{\mathbf{1}}{V}$ when $\tau_{\text{max}} \to \infty$. Empirically, we find that setting $\tau_{\text{max}} = 10.0$ ensures that the distribution is near uniform at $t = 0$ even after applying Gumbel noise, satisfying the boundary condition $\psi_0(\mathbf{x}_t|\mathbf{x}_1) \approx \frac{\mathbf{1}}{V}$.

Decay Rate λ. The decay rate determines how quickly the temperature drops as $t \to 1$. A decay rate of $\lambda = 1$ means that the function becomes $\exp(-t)$, which drops the temperature to ≈ 0.367 at $t = 1$. Since we want the temperature to approach 0 to increase the concentration of probability mass at the vertex, we set $\lambda = 3.0$ such that $\tau(t) = \tau_{\text{max}} \exp(-3.0t)$. For larger decay rates $\lambda = 10.0$, the distribution converges too quickly to a vertex, which may cause overfitting.

Stochasticity Factor β. We can tune the effect of the Gumbel noise applied during training by scaling down by a factor $\beta \geq 1.0$ such that $g_i = \frac{-\log(-\log(\mathcal{U}_i+\epsilon)+\epsilon)}{\beta}$. For larger β, the stochasticity decreases, and for smaller β, the stochasticity increases. For the toy experiment, we found similar performance for noise factors ranging between $\beta = 2.0 \to 10.0$. The remaining experiments were conducted with $\beta = 2.0$.

Step Size η and Integration Steps N_{steps}. For Gumbel-Softmax FM, the step size is equal to $\Delta t = \frac{1}{N_{\text{steps}}}$ since we are integrating the velocity field from $t = 0 \to 1$. For Gumbel-Softmax SM, the step size determines the rate of convergence to high-probability density regions. Small step sizes $\eta \leq 0.1$ increase computation cost and the number of steps needed to converge. In contrast, larger step sizes $0.1 \leq \eta \leq 1.0$ increase the speed of convergence but may result in mode-collapse to the high-density regions. Empirically, we found that a step size of $\eta = 0.5$ is optimal with the number of integration steps $N_{\text{steps}} = 100$.

Guidance Scale γ. Given that the `softmax` gradients tend to be small, especially for low-probability tokens, the guidance scale γ amplifies the gradient value across all tokens to ensure effective guidance. For the target-guided peptide design experiments, we set $\gamma = 10.0$ to scale the guidance term to be in the order 10^{-1}, which produced increasing classifier scores over iterations.

Number of Guidance Samples M. For STGFlow, the number of guidance samples M determines the number of discrete sequences that are sampled from the distribution $\mathbf{x}_t$ at each time step to compute the aggregate straight-through gradient. Larger M enables more informed and precise guidance based on the culmination of the classifier on various token combinations to determine tokens that lead to enhanced classifier scores, while smaller M results in more spurious guidance that may not lead to truly optimal sequences. We found that $M = 10$ maintained a good balance between effective guidance while minimizing computational costs.

H.2 PROTEIN EVALUATION METRICS

We evaluate protein generation quality based on the following metrics computed by ESMFold [22].

1. **pLDDT** (predicted Local Distance Difference Test) measures residue-wise local structural confidence on a scale of 0-100. Proteins with mean pLDDT > 70 generally correspond to correct backbone prediction and more stable proteins.

2. **pTM** (predicted Template Modeling) measures global structural plausibility. High pTM corresponds to a high similarity between a predicted structure and a hypothetical true structure.

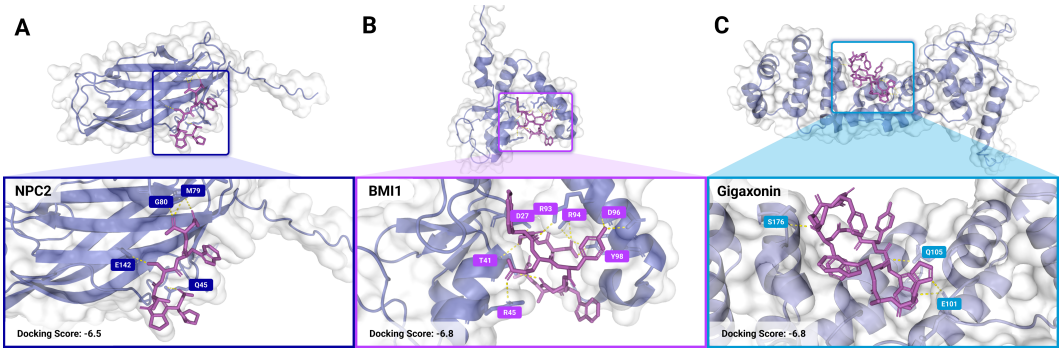

Figure 7: **Gumbel-Softmax FM generated peptide binders for three targets with no known binders.** (**A**) 7 a.a. designed binder to NPC2 (PDB: 6W5V) involved in Niemann-Pick Disease Type C. (**B**) 10 a.a. designed binder to BMI1 (PDB: 2CKL) involved in Medulloblastoma. (**C**) 10 a.a. designed binder to Gigaxonin (PDB: 3HVE) involved in Giant Axonal Neuropathy. Docked with AutoDock VINA and polar contacts within 3.5 Å are annotated. Additional targets are shown in Table 2.

3. **pAE** (predicted Alignment Error) measures the confidence in pair-wise positioning of residues. Low pAE scores correspond to low predicted pair-wise error.

In addition, we compute:

1. **Token entropy** measures the diversity of tokens within each sequence. It is defined as the Shannon entropy, where p_i is the probability of the i-th unique token divided by the total number of tokens N in the sequence.

$$E = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

2. **Diversity** is calculated as $1-$ pairwise sequence identity within a batch of generated sequences with equal length.

H.3 Peptide Evaluation Metrics

We evaluate our *de novo* peptide binders based on two metrics that measure their affinity to their target protein.

ipTM Score. We use AlphaFold3 [2] to compute the interface predicted template modeling (ipTM) score, which is on the scale from 0-1 and measures the accuracy of the predicted relative positions between residues involved in the interaction between the two sequences.

pTM Score. We use AlphaFold3 [2] to compute the predicted template modeling (pTM) score, which is on a scale from 0-1 and measures the accuracy of the predicted structure of the whole peptide-protein complex. This score is less relevant when evaluating binding affinity since it can be dominated by the stability of the target protein.

VINA Docking Score. We use Autodock Vina [15] (v 1.1.2) for *in silico* docking of the peptide binders to their target proteins (Table 2) to confirm binding affinity. The complex was first docked with Alphafold3 for the starting conformation [2]. The final results were visualized in PyMol [34] (v 3.1), where the residues in the protein targets with polar contacts to the peptide binder with distances closer than 3.5 Å are annotated.

I Algorithms

In this section, we provide detailed procedures for the training and inference of the flow and score-matching models. Algorithm 1 and 2 describe training and sampling with Gumbel-Softmax FM,

respectively. Algorithm 3 and 4 describe training and sampling with Gumbel-Softmax SM, respectively. We consider $\mathbf{x}_1$ as a single token in a sequence for simplicity, but in practice, the training and sampling are conducted on a sequence of tokens of length L.

Algorithm 1 Training Gumbel-Softmax Flow Matching

Inputs: Training sequences of one-hot vectors $\mathbf{x}_1 \in \mathcal{D}$, parameterized neural network $\text{NN}_\theta(\mathbf{x}_t, t)$, maximum temperature τ_{max}, decay rate λ, and learning rate η.
procedure TRAINING GUMBEL-SOFTMAX FM
  for $\mathbf{x}_1$ in `batch` **do**
    Sample $t \sim \text{Uniform}(0, 1)$
    Set $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    Sample $\mathcal{U} \sim \text{Uniform}(0, 1)^V$
    Sample Gumbel noise vector $\mathbf{g} = -\log(-\log(\mathcal{U} + \epsilon) + \epsilon)$
    Given the clean token $\mathbf{x}_1 = \mathbf{e}_k$, sample noisy interpolant for time t

$$x_{t,i} \leftarrow \frac{\exp\left(\frac{\delta_{ik} + (g_i/\beta)}{\tau(t)}\right)}{\sum_{j=1}^{V} \exp\left(\frac{\delta_{jk} + (g_j/\beta)}{\tau(t)}\right)}$$

    if `denoise` **then**
      Predict $\mathbf{x}_\theta(\mathbf{x}_t, t) \leftarrow \text{NN}_\theta(\mathbf{x}_t, t)$
      Minimize negative log loss $\mathcal{L}_{\text{denoise}} \leftarrow \mathbb{E}_{\mathbf{x}_1 \sim \mathcal{D}}\left[-\log(\mathbf{x}_\theta^{(k)}(\mathbf{x}_t, t))\right]$
    else
      Predict $u_t^\theta(\mathbf{x}_t) \leftarrow \text{NN}_\theta(\mathbf{x}_t, t)$
      Calculate $u_t(\mathbf{x}_t|\mathbf{x}_1) \leftarrow \frac{\lambda}{\tau(t)} x_{t,k} \left(\mathbf{e}_k - \mathbf{x}_t\right)$
      Optimize denoising loss $\mathcal{L}_{\text{mse}} \leftarrow \mathbb{E}_{\mathbf{x}_1 \sim \mathcal{D}} \|u_t^\theta(\mathbf{x}_t) - u_t(\mathbf{x}_t|\mathbf{x}_1)\|^2$
    end if
    $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{denoise}}$
  end for
end procedure

Algorithm 2 Unconditional Sampling with Gumbel-Softmax Flow Matching

Inputs: Trained neural network $\text{NN}_\theta(\mathbf{x}_t, t)$, number of integration steps N_{step}
Output: Clean sequence $\mathbf{x}$ from learned data distribution
procedure SAMPLING GUMBEL-SOFTMAX FM
  Compute step size $\Delta t \leftarrow \frac{1}{N_{\text{step}}}$
  Sample uniform distribution $\mathbf{x}_0 \leftarrow \frac{\mathbf{1}}{V}$
  Set $\mathbf{x}_t \leftarrow \mathbf{x}_0$
  for $t = 0 \rightarrow 1$ **do**
    Compute $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    if `denoise` **then**
      Predict $\mathbf{x}_\theta(\mathbf{x}_t, t) \leftarrow \text{NN}_\theta(\mathbf{x}_t, t)$
      for all simplex dimensions $k \in [1, V]$ **do**

$$u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) = \frac{\lambda}{\tau(t)} x_{t,k} \left(\mathbf{e}_k - \mathbf{x}_t\right)$$

      end for
      Calculate the conditional velocity field

$$u_t^\theta(\mathbf{x}_t) \leftarrow \sum_{k=1}^{V} u_t(\mathbf{x}|\mathbf{x}_1 = \mathbf{e}_k) \cdot \langle \mathbf{x}_\theta(\mathbf{x}_t, t), \mathbf{e}_k \rangle$$

    else
      Directly predict conditional velocity field $u_t^\theta(\mathbf{x}_t) \leftarrow \text{NN}_\theta(\mathbf{x}_t, t)$
    end if
    Take step $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta t \cdot u_t^\theta(\mathbf{x}_t)$
    $\mathbf{x}_t \leftarrow \text{SIMPLEXPROJ}(\mathbf{x}_t)$
  end for
  Sample sequence $\mathbf{x} \leftarrow \arg\max(\mathbf{x}_t)$
  return $\mathbf{x}$
end procedure

Algorithm 3 Training Gumbel-Softmax Score Matching

Inputs: Training sequences of one-hot vectors $\mathbf{x}_1 \in \mathcal{D}$, parameterized neural network $\text{NN}_\theta(\mathbf{x}_t, t)$, maximum temperature τ_{max}, decay rate λ, and learning rate η.
procedure TRAINING GUMBEL-SOFTMAX SM
  for $\mathbf{x}_1$ in `batch` **do**
    Sample $t \sim \text{Uniform}(0, 1)$
    Set $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    Sample $\mathcal{U} \sim \text{Uniform}(0, 1)^V$
    Sample Gumbel noise vector $\mathbf{g} = -\log(-\log(\mathcal{U} + \epsilon) + \epsilon)$
    Given the clean token $\mathbf{x}_1 = \mathbf{e}_k$, sample noisy interpolant for time t

$$x_{t,i} \leftarrow \frac{\exp\left(\frac{\delta_{ik} + (g_i/\beta)}{\tau(t)}\right)}{\sum_{j=1}^{V} \exp\left(\frac{\delta_{jk} + (g_j/\beta)}{\tau(t)}\right)}$$

    Predict $f_\theta(\mathbf{x}_t, t) \leftarrow \text{NN}_\theta(\mathbf{x}_t, t)$
    Optimize loss given $\mathbf{x}_1 = \mathbf{e}_k$

$$\mathcal{L}_{\text{score}} \leftarrow \mathbb{E}_{\mathbf{x}_1 \sim \mathcal{D}} \| f_\theta(\mathbf{x}_t, t) - (\delta_{ik} + \tau(t) x_{t,i}) \|^2$$

    $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{score}}$
  end for
end procedure

Algorithm 4 Unconditional Sampling with Gumbel-Softmax Score Matching

Inputs: Trained score model $s_\theta(\mathbf{x}_t, t)$, step size Δ, noise factor β
Output: Clean sequence $\mathbf{x}$ from learned data distribution
procedure SAMPLING
  $\mathbf{x}_0 \leftarrow \frac{\mathbf{1}}{V}$
  Set $\mathbf{x}_t \leftarrow \mathbf{x}_0$
  for $t = 0 \rightarrow 1$ **do**
    Compute $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    Predict $f_\theta(\mathbf{x}_t, t) \leftarrow \text{NN}_\theta(\mathbf{x}_t, t)$
    Compute predicted score $s_\theta(\mathbf{x}_t, t) \leftarrow -\tau(t) + \tau(t)V \cdot \text{SM}\big(f_\theta(\mathbf{x}_t, t)\big)$
    $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta \cdot s_\theta(\mathbf{x}_t, t)$
    $\mathbf{x}_t \leftarrow \text{SIMPLEXPROJ}(\mathbf{x}_t)$
  end for
  Sample sequence $\mathbf{x} \leftarrow \arg\max(\mathbf{x}_t)$
  return $\mathbf{x}$
end procedure

Algorithm 5 Straight-Through Guided Flow Matching (STGFlow)

Inputs: Trained simplex-based flow matching model $u_t^\theta(\mathbf{x}_t)$, trained classifier model $p_\phi(y|\mathbf{x}) : \mathcal{V}^L \rightarrow \mathbb{R}$ that takes a sequence of length L and returns a classifier score, number of integration steps N_{iter}
Output: Clean sequence $\mathbf{x}$ from learned data distribution with optimized classifier score
procedure GUIDED SAMPLING WITH STGFLOW
  Compute step size $\Delta t \leftarrow \frac{1}{N_{\text{step}}}$
  $\mathbf{x}_0 \leftarrow \frac{\mathbf{1}}{V}$
  Set $\mathbf{x}_t \leftarrow \mathbf{x}_0$
  for $t = 0 \rightarrow 1$ **do**
    Predict unguided conditional velocity field $u_t^\theta(\mathbf{x}_t)$ as in Algorithm 2
    Take step $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta t \cdot u_t^\theta(\mathbf{x}_t)$
    Compute top-k distribution $\text{SM}\left(\text{top}k(\mathbf{x}_t)\right)$
    Sample M sequences from top$k$ distribution $\tilde{\mathbf{x}}_{1,m} \sim \text{SM}\left(\text{top}k(\mathbf{x}_t)\right)$
    Initialize total guided velocity $u_t^\phi(\mathbf{x}_t|\mathbf{x}_1, y) \leftarrow 0$
    for each $\tilde{\mathbf{x}}_{1,m}$ **do**
      Compute score $y \leftarrow p_\phi(y|\tilde{\mathbf{x}}_{1,m})$
      Compute straight-through gradient with respect to distribution $\mathbf{x}_t$

$$\nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m}) = \begin{cases} \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \cdot \left[\text{SM}(x_{t,i})\left(1 - \text{SM}(x_{t,k})\right)\right] & i = k \\ \frac{\partial p_\phi(y|\tilde{\mathbf{x}}_{1,m})}{\tilde{\mathbf{x}}_1} \cdot \left[-\text{SM}(x_{t,i})\text{SM}(x_{t,j})\right] & i \neq k \end{cases}$$

      Add to total guidance $u_t^\phi(\mathbf{x}_t|\mathbf{x}_1, y) \leftarrow u_t^\phi(\mathbf{x}_t|\mathbf{x}_1, y) + \nabla_{\mathbf{x}_t} p_\phi(y|\tilde{\mathbf{x}}_{1,m})$
    end for
    Add total guided velocity $\mathbf{x}_t \leftarrow \mathbf{x}_t + \gamma \cdot u_t^\phi(\mathbf{x}_t|\mathbf{x}_1, y)$
  end for
  Sample sequence $\mathbf{x} \sim \mathbf{x}_t$
  return $\mathbf{x}$
end procedure