GUMBEL-SOFTMAX SCORE AND FLOW MATCHING FOR DISCRETE BIOLOGICAL SEQUENCE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce **Gumbel-Softmax Score and Flow Matching**, a generative framework that relies on a novel Gumbel-Softmax interpolation between smooth categorical distributions to one concentrated at a single vertex by defining a time-dependent temperature parameter. Using this interpolant, we explore Gumbel-Softmax Flow Matching by deriving a parameterized velocity field transports smooth categorical distributions to the vertices of the simplex. We alternatively present Gumbel-Softmax Score Matching which learns to regress the gradient of the probability density. Our approach enables controllable generation with tunable temperatures and stochastic Gumbel noise during inference, enabling efficient *de novo* sequence design. Our experiments demonstrate state-of-the-art performance in conditional DNA promoter design and strong results in *de novo* sequence-only protein generation.

1 INTRODUCTION

Generative modeling has transformed the design of biological sequences, enabling *de novo* generation of proteins (Madani et al., 2023; Ferruz et al., 2022; Nisonoff et al., 2024), DNA regulatory elements (Stark et al., 2024; Nisonoff et al., 2024), and peptides (Bhat et al., 2025; Chen et al., 2023; Tang et al., 2024). However, generating structured sequences in discrete spaces remains an open challenge due to the inherent non-differentiability of categorical variables. Traditional autoregressive models, such as ProtGPT2 (Ferruz et al., 2022) and ProGen2 (Madani et al., 2023), learn sequence distributions by iteratively predicting tokens, but suffer from compounding errors, bias accumulation, and limited global coherence. To address these issues, generative models based on diffusion (Austin et al., 2021; Wang et al., 2024; Shi et al., 2024; Sahoo et al., 2024) and flow matching (Gat et al., 2024; Stark et al., 2024; Nisonoff et al., 2024) have been developed to enable non-autoregressive sampling of sequences.

Discrete diffusion models, such as Masked Discrete Language Models (MDLMs) (Shi et al., 2024; Sahoo et al., 2024) and Denoising Diffusion Probabilistic Models (D3PMs) (Austin et al., 2021), iteratively reconstruct sequences by modeling forward and reverse noise processes in a Markovian framework. These approaches have demonstrated success in DNA sequence design (Stark et al., 2024; Nisonoff et al., 2024), protein generation (Wang et al., 2024; Goel et al., 2024), and recently, multi-objective generation of therapeutic peptides (Tang et al., 2024). However, they face inefficiencies due to their reliance on expensive iterative denoising steps. More recently, discrete flow matching has emerged as a powerful alternative, learning continuous-time velocity fields to efficiently transport categorical distributions from noise to data (Gat et al., 2024; Stark et al., 2024; Nisonoff et al., 2024; Davis et al., 2024). These methods have enabled state-of-the-art results in DNA regulatory element design (Stark et al., 2024; Nisonoff et al., 2024) and discrete sequence modeling (Gat et al., 2024).

Despite these advances, discrete flow models on the simplex remain underexplored, particularly for *de novo* protein design. To address this gap, we introduce **Gumbel-Softmax Score and Flow Matching**, a framework that transforms noisy to clean data in the continuous simplex space using a Gumbel-Softmax interpolant (Jang et al., 2017). By our parameterization of the velocity field and score functions, we enable controllable generation with dynamic temperature scaling and tunable stochasticity during inference.

Our key contributions are as follows:

1. **Gumbel-Softmax Flow Matching**. We introduce Gumbel-Softmax Flow Matching (GS-FM), a generative framework that leverages temperature-controlled Gumbel-softmax interpolants for smooth transport from noisy to clean distributions on the simplex. We define a new velocity field that follows a mixture of learned interpolations between categorical distributions that converges to high-quality sequences (Section 3).
2. **Dynamic-Temperature Tuning**. Since the temperature parameter controls the sharpness of the resulting categorical distribution after applying the Gumbel-softmax transformation, we propose a method of modulating the temperature parameter at inference time based on a token-wise predicted uncertainty (Section 3.3).
3. **Gumbel-Softmax Score Matching**. As an alternative generative framework using the same Gumbel-softmax interpolant, we propose Gumbel-Softmax Score Matching (GS-SM) that estimates the gradient of probability density at varying temperatures to enable sampling from high-density regions on the simplex (Section 4).
4. **Biological Sequence Generation**. We apply our model to conditional DNA promoter design and *de novo* protein sequence generation, demonstrating that it achieves state-of-the-art sequence diversity and perplexity compared to autoregressive and discrete diffusion-based baselines (Section 5).

Our results highlight the potential of discrete flow and score matching for biomolecular sequence generation, offering several theoretical and empirical advantages over autoregressive models and discrete diffusion models. We believe our framework will serve as a foundation for future advances in sequence-based biological design.

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 $\mathcal{D}$. The challenge of generative modeling over the simplex consists of defining a mapping ψ that smoothly interpolates between p_0 and p_1. Then, we can generate samples from p_1 by first sampling from p_0 the applying a learned vector field that maps from p_0 to p_1.

2.1 THE GUMBEL-SOFTMAX DISTRIBUTION

The Gumbel-Softmax distribution or Concrete distribution (Jang et al., 2017; Maddison et al., 2016) is a categorical distribution over the interior of the $(V-1)$-dimensional simplex Δ^{V-1} that smoothly interpolates between discrete one-hot samples to uniform categorical distributions by varying a temperature parameter $\tau > 0$.

While the `argmax` function returns a one-hot vector defined as a vertex of the simplex, the Gumbel-Softmax function 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 $\mathcal{U} \sim \text{Uniform}(0,1)$, scaling down by the temperature parameter $\tau > 0$, and applying the continuous `softmax` function across the distribution such that the elements sum to 1. Given parameters $\pi_i \in (\epsilon, \infty)$ representing the original logits of each category where ϵ is a small constant to avoid division by 0, the Gumbel-Softmax random variable is given by

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

We observe that as $\tau \to 0$ and $\pi_i = \max(\delta_{ik}, \epsilon)$ denoting a one-hot distribution at the kth vertex, the distribution converges to a one-hot vector where $x_k \to 1$ and $x_j \to 0$ for all $j \neq k$. 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 is a generative framework that aims to transform noisy samples $\mathbf{x}_0 \sim p_0$ from a source distribution p to clean samples $\mathbf{x}_1 \sim p_1$ where p_1 is the data distribution. This transformation is approximated through a learnable velocity vector field u_t^θ that *generates* the distribution p_t interpolating between p_0 and p_1. The flow *or interpolant* $\psi_t(\mathbf{x}_0|\mathbf{x}_1) : [0,1] \times \Delta^{V-1} \times \Delta^{V-1} \to \Delta^{V-1}$ is a bijection that maps a noisy distribution $\mathbf{x}_0$ on the interior of the simplex to the intermediate distribution $\mathbf{x}_t$ at time t, which satisfies the constraints $\psi_0(\mathbf{x}_0|\mathbf{x}_1) = \mathbf{x}_0$ and $\psi_1(\mathbf{x}_0|\mathbf{x}_1) = \mathbf{x}_1 \sim p_t$. Therefore, we define the velocity field as the derivative of the flow concerning time t.

$$u_t(\mathbf{x}_t|\mathbf{x}_1) = \frac{d}{dt}\psi_t(\mathbf{x}_0|\mathbf{x}_1) \tag{2}$$

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

$$\frac{\partial}{\partial t}p_t + \nabla \cdot (p_t u_t) = 0 \quad \text{where} \quad \nabla \cdot (p_t u_t) = \sum_{i=1}^{V} \frac{\partial}{\partial x_{t,i}}(p_t u_t) \tag{3}$$

where $\nabla\cdot$ is the divergence operator that describes the total outgoing flux at a point.

The flow matching (FM) objective is to train a parameterized model $u_t^\theta(\mathbf{x}_t, t)$ 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 $u_t(x)$ is a joint transformation between two complex distributions and intractable, we condition the velocity field on a specific data point $\mathbf{x}_1$ and compute the *conditional flow-matching loss* 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 $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$ (Lipman et al., 2022; Tong et al., 2023). Among existing discrete flow matching methods, there are two methods of defining a discrete flow: defining the *interpolant* $\psi_t(\mathbf{x}_0|\mathbf{x}_1)$ 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 (Song & Ermon, 2019) is another generative matching framework that learns the gradient of a probability density path $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ 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.

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

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

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

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

3 GUMBEL-SOFTMAX FLOW MATCHING

Next we describe **Gumbel-Softmax Flow Matching** (GS-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.

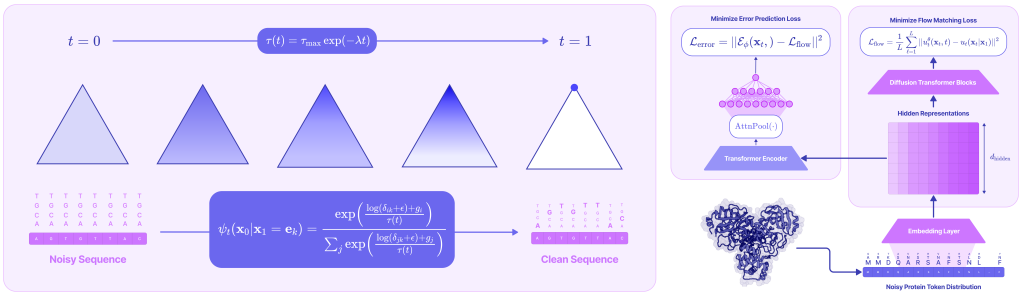

Figure 1: **Overview of Gumbel-Softmax Score and Flow Matching.** Gumbel-softmax transformations are applied to clean one-hot sequences for varying temperatures dependent on time. The embedded noisy distributions are passed into a DiT flow or score model and error prediction model to predict the conditional flow velocity and score function.

3.1 Defining the Gumbel-Softmax Interpolant

We propose a new definition of the discrete probability path by gradually decreasing the temperature of a Gumbel-Softmax categorical distribution as a function of time where the maximum probability corresponds to the target token. First, we define a monotonically decreasing function $\tau(t) \in (0, \infty)$ to prevent the Gumbel-Softmax distribution from being undefined at $\tau = 0$.

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

where $\tau_{\max}$ is the initial temperature set to a large number so that the categorical distribution resembles a uniform distribution, λ controls the decay rate, and t is the time that goes from $t = 0$ to $t = 1$.

Now, we define the conditional interpolant $\mathbf{x}_t = \psi_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k)$ with $t \in [0, 1]$ as

$$\psi_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) = \frac{\exp\left(\frac{\log(\delta_{ik}+\epsilon)+\frac{g_i}{\beta}}{\tau(t)}\right)}{\sum_{j=1}^{V} \exp\left(\frac{\log(\delta_{jk}+\epsilon)+\frac{g_j}{\beta}}{\tau(t)}\right)} \tag{9}$$

where $\tau(t) = \tau_{\max} \exp(-\lambda t)$, δ_{ik} is the Kronecker delta function that returns 1 when $i = k$ and 0 otherwise, and $\epsilon = 10^{-5}$ prevents computing the undefined $\log(0)$ with $i \neq k$. This interpolant ensures that the probability of observing the target token increases as a function of $1 + \delta_{ik} t$ and the distribution becomes more concentrated at the target vertex through a decaying temperature function $\tau(t)$. Gumbel noise $g_i = -\log(-\log(\mathcal{U}_i))$ where $\mathcal{U}_i \sim \text{Uniform}(0, 1)$ is applied during training to ensure that the model learns to reconstruct a clean sequence given contextual information.

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 corner 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 A.1).

$$u_{t,i}(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) = \begin{cases} \frac{\lambda \exp(\lambda t)}{\tau_{\max}} \cdot x_{t,i} \sum_j \left(x_{t,j} \cdot (1 - \delta_{jk})\right) & i = k \\ \frac{\lambda \exp(\lambda t)}{\tau_{\max}} \cdot x_{t,i} \sum_j \left(x_{t,j} \cdot (-\delta_{jk})\right) & i \neq k \end{cases} \tag{10}$$

We observe that the velocity field points toward the target category $\mathbf{x}_1 = \mathbf{e}_k$ at a rate proportional to $x_{t,k}(1 - x_{t,k})$, indicating that the magnitude of the velocity field is maximized at $x_{t,k} = \frac{1}{2}$ and zero when $x_{t,k} \in \{0, 1\}$. Contrarily, the velocity field points away from all other vertices at a rate proportional to $-x_{t,i} x_{t,k}$. This means that the velocity field vanishes both at the vertex and the $(V - 2)$-dimensional face directly opposite to the vertex similar to Dirichlet FM, overcoming the pathological behavior of Linear FM.

Proposition 1. *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 in Appendix A.2.*

Our goal is to train a parameterized model to predict the velocity field $u_t(\mathbf{x}_t)$ given a noisy categorical distribution $\mathbf{x}_t$. Instead of directly regressing $u_t(\mathbf{x}_t|\mathbf{x}_1)$ conditioned on data samples $\mathbf{x}_1$, we train a *denoising* model that reconstructs the clean one-hot distribution $\mathbf{x}_\theta(\mathbf{x}_t, t)$ after applying the Gumbel-Softmax transformation. Then, we compute the predicted velocity field via the equation

$$u_t^\theta(\mathbf{x}_t) = \frac{\lambda \exp(\lambda t)}{\tau_{\max}} \mathbf{x}_t \odot \left(\mathbf{x}_\theta \langle \mathbf{x}_t, \mathbf{1} - \mathbf{x}_\theta \rangle + (\mathbf{1} - \mathbf{x}_\theta) \langle \mathbf{x}_t, -\mathbf{x}_\theta \rangle \right) \tag{11}$$

where θ minimizes the denoising loss given by

$$\mathcal{L}_{\text{denoise}} = \frac{1}{L} \sum_{\ell=1}^{L} \|\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_1\|^2 \tag{12}$$

Proposition 2 (Equality of Denoising and Flow Matching Objectives)**.** *Minimizing the conditional flow matching loss is equivalent to minimizing the denoising loss such that $\arg\min_\theta \left[\mathbb{E}_{p_t(\mathbf{x}_t)} \|u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t)\|^2\right] = \arg\min_\theta \left[\mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_1 - \mathbf{x}_\theta\|^2\right]$. Proof in Appendix A.2.*

3.3 Controllable Flow Paths with Dynamic Temperatures

Given that the reverse process interpolates between smooth distributions on the interior of the simplex, the model needs to learn the path toward a clean distribution from only noisy token distributions. This could prevent the model from effectively decoding tokens given the context of already sharp neighboring distributions, especially for high simplex dimensions. To ensure more accurate flows, we train an error prediction model similar to that of (Zhang et al., 2024) that estimates the squared loss between the predicted clean distribution $\mathbf{x}_\theta(\mathbf{x}_t, t)$ from the denoising model and the one-hot distribution $\mathbf{x}_1$

$$\mathcal{L}_{\text{error}} = \frac{1}{L} \sum_{\ell=1}^{L} \left\| \mathcal{E}_\phi(\mathbf{x}_t, t) - \|\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_1\|^2 \right\|^2 \tag{13}$$

With the predicted error, we modulate the time-dependent temperature during inference with $\tau_\phi(t) = \tau(t) - \alpha(||\mathcal{E}_\phi(\mathbf{x}_t, t|| - \epsilon)$ which increases the temperature for $||\mathcal{E}_\phi(\mathbf{x}_t, t|| > \epsilon$ and reduces the temperature for $||\mathcal{E}_\phi(\mathbf{x}_t, t|| < \epsilon$ scaled by a constant α.

4 Gumbel-Softmax Score Matching

As an alternative to our flow matching framework, we propose **Gumbel-Softmax Score Matching** (GS-SM), a score-matching method that learns the gradient of the probability density (defined as the *score*) $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ associated with the Gumbel-Softmax interpolant. We find that Gumbel-Softmax score matching performs superior to flow matching for increasing simplex dimensions, specifically for *de novo* protein design.

4.1 The Exponential Gumbel-Softmax 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 leverage ExpConcrete probability distributions (Maddison et al., 2016) that satisfies $\mathbf{z} \sim \text{ExpConcrete}(\tau, \pi)$ such that $\exp(\mathbf{z}) \sim \text{GumbelSM}(\tau, \pi)$. Formally, the ith entry of an ExpConcrete random variable is defined as

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

The gradient of the log-probability density of this distribution is given by

$$\nabla_{x_j} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = -\tau(t) + \tau(t)V \cdot \text{SM}\Big(\log(\delta_{ik} + \epsilon) - \tau(t) x_i \Big) \tag{15}$$

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 (Mahmood et al., 2024), given by

$$s_\theta(\mathbf{x}_t, t) = -\tau(t) + \tau(t)V \cdot \text{SM}\big(f_\theta(\mathbf{x}_t, t)\big) \quad \text{where} \quad s_\theta(\mathbf{x}_t, t) \approx \nabla_{x_j} \log p_t(\mathbf{x}_t) \tag{16}$$

where θ minimizes the reparameterized score-matching loss

$$\begin{aligned}\mathcal{L}_{\text{score}}(\theta) &= \frac{1}{L}\sum_{\ell=1}^{L} \left\| \big[-\tau(t) + \tau(t)V \cdot \text{SM}(\log(\delta_{ik} + \epsilon) - \tau(t)x_i)\big] - \big[-\tau(t) + \tau(t)V \cdot \text{SM}(f_\theta(\mathbf{x}_t, t)\big] \right\|^2 \\ &= \frac{1}{L}\sum_{\ell=1}^{L} \tau(t)^2 V^2 \left\| \text{SM}\big(\log(\delta_{ik} + \epsilon) + \tau(t)x_{t,i}\big) - \text{SM}(f_\theta(\mathbf{x}_t, t))\right\|^2 \end{aligned} \tag{17}$$

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.

Proposition 3. *The gradient of the ExpConcrete log-probability density is proportional to the gradient of the Gumbel-softmax log-probability density such that* $\nabla_{x_j}^{GS} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1) \propto \nabla_{x_j}^{ExpGS} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1)$. *Proof in Appendix B.2.*

By minimizing $\mathcal{L}_{\text{score}}$, we obtain a model that effectively transports noisy categorical distributions towards clean distributions in high-probability regions of the discrete state space.

5 EXPERIMENTS

5.1 PROMOTER DNA SEQUENCE DESIGN

Following the experimental procedures of previous works (Avdeyev et al., 2023; Stark et al., 2024), we evaluate our flow-matching strategy on promoter DNA design and show superior performance to diffusion and simplex flow-matching baselines.

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 flow model conditioned on the transcription profile which can generate sequences that minimize the MSE of the predicted regulatory signal of the generated sequence against the signal of the original sequence with the input profile using a pre-trained Sei model (Chen et al., 2022a).

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 FM	0.034
Fisher FM	**0.029**
Gumbel-Softmax FM (Ours)	**0.029**

Table 1: **Evaluation of promoter DNA generation conditioned on transcription profile.** Mean squared error between the predicted regulatory signal between the generated and original sequence with an input transcription profile. Signals were predicted using pre-trained Sei model (Chen et al., 2022a)

Training. Following (Stark et al., 2024), we trained on a train/test/validation split of 88,470/3,933/7,497 sequences with a length of 1,024. For each batch of size 256, we applied Gumbel-Softmax noise according to Equation 9 with $\tau_{\text{max}} = 10.0$ and $\lambda = 1.0$ for uniformly distributed time steps $t \in [0, 1]$. The training objective was to minimize the negative log loss between the true one-hot tokens $\mathbf{x}_0$ and predicted clean logits $\mathbf{x}_\theta(\mathbf{x}_t, t)$ for varying degrees of noise. We parameterized the denoiser with a 20-layer 1D CNN architecture following (Stark et al., 2024) for a total of 100,000 steps using the AdamW optimizer and a learning rate of 10^{-4}.

Table 2: **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	738M	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

Results. We compared our Gumbel-Softmax FM model with Fisher FM (Davis et al., 2024) and Dirichlet FM (Stark et al., 2024), two discrete flow matching models on the interior of the simplex which are most closely related to our work. We also compare against discrete diffusion models (Avdeyev et al., 2023; Chen et al., 2022b; Austin et al., 2021) and an autoregressive language model baseline. Our generated sequences demonstrate lower signal MSE compared to diffusion baselines and Dirichlet FM and comparable MSE to Fisher FM.

5.2 *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 (198 million params), we demonstrate competitive generative quality compared to discrete diffusion and autoregressive baselines.

Setup. Given the larger vocabulary size of protein sequences, we compared both the performance of GS-FM and GS-SM for this task. For both models, we applied the Gumbel-Softmax transformation with varying temperatures $\tau(t)$ for time steps $t = 0 \to 1$ and $\tau_{\max} = 100.0$ for score matching and $\tau_{\max} = 10.0$ for flow matching. 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 GS-FM and 3 for GS-SM. Sampling was performed following Algorithm 2 and Algorithm 4.

Training. We collected 68M Uniref50 and 207M OMG_PROT50 data (Suzek et al., 2007; Cornman et al., 2024). A total of 275M protein sequences were first clustered to remove singletons using MMseqs2 linclust (Steinegger & Söding, 2018) (parameters: –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.

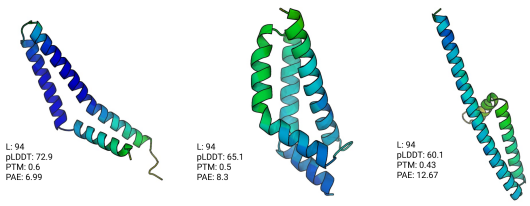

Figure 2: **Predicted structures of *de novo* generated proteins from Gumbel-Softmax FM.** The structures, pLDDT, pAE, and pTM scores are predicted with ESMFold (Lin et al., 2023b)

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 (Alamdari et al., 2023), large language model ProtGPT2 (Ferruz et al., 2022), and the autoregressive model ProGen2 (medium) (Nijkamp et al., 2023). For 100 unconditionally generated sequences per model, we compute the pLDDT, pTM, pAE scores using ESMFold (Lin et al., 2023a) as well as the entropy and pseudo-perplexity. We also compute the overall diversity in the generated sequences. Additional details on evaluation metrics are given in Appendix D.2. BLASTp runs for the proteins we generated indicate no homologs hits, highlighting again the novelty of the proteins we generated and indicating that our model is not sub-sampling from known homologous protein sequences. As summarized in Table 2, 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.

6 CONCLUSION

In this work, we introduce **Gumbel-Softmax Score and Flow Matching**, a novel discrete framework that learns interpolations between noisy and clean data by modulating the temperature of the Gumbel-Softmax distribution. Our method extends discrete flow models to categorical sequence spaces without requiring simplex constraints or predefined transition matrices, allowing for efficient and flexible discrete transport. By parameterizing probability velocity fields directly in the discrete domain, we overcome limitations of existing discrete generative models, such as the reliance on iterative denoising in discrete diffusion (Austin et al., 2021; Wang et al., 2024; Shi et al., 2024; Sahoo et al., 2024) or the restrictive probability constraints in Dirichlet and Fisher Flow Matching (Stark et al., 2024; Davis et al., 2024).

We apply our model to two key biological sequence generation tasks: conditional DNA promoter design and *de novo* protein sequence generation. For promoter design, GUMBEL-SOFTMAX FLOW generates functional DNA sequences with enhanced transcriptional activity, outperforming previous discrete generative approaches. For protein sequence generation, our method enables the design of structurally-feasible proteins while maintaining sequence diversity and uniqueness against known proteins. Unlike discrete diffusion and autoregressive models, our approach operates in the continuous multi-dimensional simplex space, enabling smooth, controllable transport from uniform categorical distributions to clean sequences.

By bridging discrete flow matching with Gumbel-Softmax reparameterization, our work provides a scalable and theoretically grounded framework for discrete sequence modeling. Future directions include extending the approach to multi-objective sequence optimization, incorporating task-specific priors to enhance design constraints, and applying Gumbel-Softmax FM to other structured biological design problems, such as RNA sequence engineering and regulatory circuit design.

7 MEANINGFULNESS STATEMENT

Gumbel-Softmax Score and Flow Matching introduces a principled approach for discrete biological sequence generation, addressing the challenge of smooth interpolation between noisy and structured sequences. By leveraging temperature-controlled Gumbel-Softmax transformations, this framework enables precise, controllable sampling for applications in DNA regulatory design and de novo protein generation. Its ability to efficiently model categorical distributions without restrictive probability constraints makes it a powerful tool for advancing biomolecular design, with potential implications in synthetic biology, drug discovery, and therapeutic development.

REFERENCES

Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Neil Tenenholtz, Robert Strome, Alan M. Moses, Alex X. Lu, Nicolò Fusi, Ava P. Amini, and Kevin K. Yang. Protein generation with evolutionary diffusion: sequence is all you need. September 2023. doi: 10.1101/2023.09.11.556673. URL http://dx.doi.org/10.1101/2023.09.11.556673.

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.

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.

Suhaas Bhat, Kalyan Palepu, Lauren Hong, Joey Mao, Tianzheng Ye, Rema Iyer, Lin Zhao, Tianlai Chen, Sophia Vincoff, Rio Watson, Tian Z. Wang, Divya Srijay, Venkata Srikar Kavirayuni, Kseniia

Kholina, Shrey Goel, Pranay Vure, Aniruddha J. Deshpande, Scott H. Soderling, Matthew P. DeLisa, and Pranam Chatterjee. De novo design of peptide binders to conformationally diverse targets with contrastive language modeling. *Science Advances*, 11(4), January 2025. ISSN 2375-2548. doi: 10.1126/sciadv.adr8638. URL http://dx.doi.org/10.1126/sciadv.adr8638.

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 2022a. ISSN 1546-1718. doi: 10.1038/s41588-022-01102-2. URL http://dx.doi.org/10.1038/s41588-022-01102-2.

Tianlai Chen, Madeleine Dumas, Rio Watson, Sophia Vincoff, Christina Peng, Lin Zhao, Lauren Hong, Sarah Pertsemlidis, Mayumi Shaepers-Cheu, Tian Zi Wang, Divya Srijay, Connor Monticello, Pranay Vure, Rishab Pulugurta, Kseniia Kholina, Shrey Goel, Matthew P. DeLisa, Ray Truant, Hector C. Aguilar, and Pranam Chatterjee. Pepmlm: Target sequence-conditioned generation of therapeutic peptide binders via span masked language modeling. *arXiv*, 2023. doi: 10.48550/ARXIV.2310.03842. URL https://arxiv.org/abs/2310.03842.

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

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.

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.

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.

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.

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.

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.

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 2023a. ISSN 1095-9203. doi: 10.1126/science.ade2574. URL http://dx.doi.org/10.1126/science.ade2574.

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 2023b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2022. URL https://arxiv.org/abs/2210.02747.

Ali Madani, Ben Krause, Eric R. Greene, Subu Subramanian, Benjamin P. Mohr, James M. Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z. Sun, Richard Socher, James S. Fraser, and Nikhil Naik. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, January 2023. ISSN 1546-1696. doi: 10.1038/s41587-022-01618-2. URL http://dx.doi.org/10.1038/s41587-022-01618-2.

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.

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.

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.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv*, 2024. doi: 10.48550/ARXIV.2406.01572. URL https://arxiv.org/abs/2406.01572.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2022. URL https://arxiv.org/abs/2212.09748.

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.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv*, 2024. doi: 10.48550/ARXIV.2406.04329. URL https://arxiv.org/abs/2406.04329.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2019. URL https://arxiv.org/abs/1907.05600.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design, 2024. URL https://arxiv.org/abs/2402.05841.

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

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.

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.

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.

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, 2023. URL https://arxiv.org/abs/2302.00482.

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.

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.

A Flow Matching Derivations

A.1 Derivation of the Gumbel-Softmax Velocity Field

We derive the conditional velocity field at a point $\mathbf{x}_t$ with the Gumbel-Softmax interpolant $u_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_i)$ by taking the derivative of the interpolant $\psi_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_i)$.

$$\begin{aligned} u_t(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{d}{dt}\psi_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) \\ &= \frac{d}{dt}\frac{\exp\left(\frac{\delta_{ik}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right)}{\sum_{j=1}^{V}\exp\left(\frac{\delta_{jk}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right)} \end{aligned} \tag{18}$$

Letting $z_i = \exp\left(\frac{\delta_{ik}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right)$, we have

$$\begin{aligned} u_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{d}{dt}\frac{\exp(z_i)}{\sum_j \exp(z_j)} \\ &= \frac{\left(\sum_{j=1}^{V}\exp(z_j)\right)\left(\frac{d}{dt}\exp(z_i)\right) - \exp(z_i)\left(\frac{d}{dt}\sum_{j=1}^{V}\exp(z_j)\right)}{\left(\sum_{j=1}^{V}\exp(z_j)\right)^2} \end{aligned} \tag{19}$$

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

$$\begin{aligned} \frac{d}{dt}\exp\left(\frac{\delta_{ik}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right) &= \exp(z_i)\cdot\frac{d}{dt}\left(\frac{\delta_{ik}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right) \\ &= \exp(z_i)\cdot\frac{1}{\tau_{\max}}\cdot\frac{d}{dt}\left(\frac{\delta_{ik}+\epsilon}{\exp(-\lambda t)}\right) \\ &= \exp(z_i)\cdot\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\cdot(\delta_{ik}+\epsilon) \end{aligned}$$

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

$$\begin{aligned} \frac{d}{dt}\sum_{j=1}^{V}\exp\left(\frac{\delta_{jk}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right) &= \sum_{j=1}^{V}\frac{d}{dt}\exp\left(\frac{\delta_{jk}+\epsilon}{\tau_{\max}\exp(-\lambda t)}\right) \\ &= \sum_{j=1}^{V}\left(\exp(z_i)\cdot\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\cdot(\delta_{jk}+\epsilon)\right) \end{aligned} \tag{20}$$

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

$$\begin{aligned} &u_{t,i}(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) \\ &= \frac{\left(\sum_j \exp(z_j)\right)\cdot\exp(z_i)\cdot\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\cdot(\delta_{ik}+\epsilon) - \exp(z_i)\cdot\sum_{j=1}^{V}\left(\exp(z_i)\cdot\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\cdot(\delta_{jk}+\epsilon)\right)}{\left(\sum_j \exp(z_j)\right)^2} \\ &= \frac{\exp(z_i)\cdot\lambda\exp(\lambda t)}{\tau_{\max}\left(\sum_j \exp(z_j)\right)^2}\left[\left(\sum_j \exp(z_j)\right)\cdot(\delta_{ik}+\epsilon) - \sum_j\left(\exp(z_j)\cdot(\delta_{jk}+\epsilon)\right)\right] \\ &= \frac{\exp(z_i)\cdot\lambda\exp(\lambda t)}{\tau_{\max}\left(\sum_j \exp(z_j)\right)^2}\left[\sum_j \exp(z_j)\left[(\delta_{ik}+\epsilon) - (\delta_{jk}+\epsilon)\right]\right] \\ &= \frac{\exp(z_i)}{\sum_j \exp(z_j)}\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\left[\sum_j\left(\frac{\exp(z_j)}{\sum_{j'}\exp(z_j)}\cdot(\delta_{ik}-\delta_{jk})\right)\right] \\ &= \psi_t^{(i)}(\mathbf{x}_0|\mathbf{x}_1)\cdot\frac{\lambda\exp(\lambda t)}{\tau_{\max}}\left[\sum_j\left(\psi_t^{(j)}(\mathbf{x}_0|\mathbf{x}_1)\cdot(\delta_{ik}-\delta_{jk})\right)\right] \\ &= \frac{\lambda\exp(\lambda t)}{\tau_{\max}}x_{t,i}\sum_j\left(x_{t,j}\cdot(\delta_{ik}-\delta_{jk})\right) \end{aligned}$$

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

$$u_{t,i}(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) = \begin{cases} x_{t,i} \cdot \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \sum_j \left(x_{t,j} \cdot (1 - \delta_{jk})\right) & i = k \\ x_{t,i} \cdot \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \sum_j \left(x_{t,j} \cdot (-\delta_{jk})\right) & i \neq k \end{cases} \tag{21}$$

Condensing this equation we get

$$\begin{aligned} u_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \mathbf{x}_t \odot \left(\mathbf{x}_1 \langle \mathbf{x}_t, \mathbf{1} - \mathbf{x}_1 \rangle + (\mathbf{1} - \mathbf{x}_1) \langle \mathbf{x}_t, -\mathbf{x}_1 \rangle\right) \\ &= \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \mathbf{x}_t \odot \left(\mathbf{x}_1 \sum_{j \neq k} x_{t,j} + (\mathbf{1} - \mathbf{x}_1)(-x_{t,k})\right) \\ &= \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \left[(\mathbf{x}_t \odot \mathbf{x}_1)(1 - x_{t,k}) - (\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_1)) x_{t,k}\right] \end{aligned} \tag{22}$$

A.2 Proof of Flow Matching Propositions

Propositiion 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. We show that the conditional velocity field derived from the Gumbel-Softmax interpolant preserves probability mass such that

$$\sum_{i=1}^{V} u_{t,i}(\mathbf{x}_t|\mathbf{x}_1 = \mathbf{e}_k) = 0 \tag{23}$$

Summing up the velocities for all $i \in [1 \dots V]$ according to Equation 21, we have

$$\begin{aligned} \sum_i u_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) &= \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \cdot x_{t,k} \sum_j \left(x_{t,j} \cdot (1 - \delta_{jk})\right) + \sum_{i \neq k} \left(\frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \cdot x_{t,i} \sum_j \left(x_{t,j} \cdot (-\delta_{jk})\right)\right) \\ &= \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \left(x_{t,k} \sum_{j \neq k} x_{t,j} + \sum_{i \neq k} x_{t,i}(-x_{t,k})\right) \\ &= x_{t,k} \sum_{j \neq k} x_{t,j} - x_{t,k} \sum_{j \neq k} x_{t,j} \\ &= 0 \end{aligned} \tag{24}$$

which proves that our velocity field preserves the probability mass at all times t.

Proposition 2 (Equality of Denoising and Flow Matching Objectives) Minimizing the conditional flow matching loss is equivalent to minimizing the denoising loss such that $\arg\min_\theta \left[\mathbb{E}_{p_t(\mathbf{x}_t)} \|u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t)\|^2\right] = \arg\min_\theta \left[\mathbb{E}_{p_t(\mathbf{x}_t)} \|\mathbf{x}_1 - \mathbf{x}_\theta\|^2\right]$.

Proof. We derived in Appendix 22 that the conditional velocity field $u_t(\mathbf{x}_t|\mathbf{x}_1)$ is given by

$$u_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k) = \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \left[(\mathbf{x}_t \odot \mathbf{x}_1)(1 - x_{t,k}) - (\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_1)) x_{t,k}\right] \tag{25}$$

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

$$
\begin{aligned}
&\mathbb{E}_{p_t(\mathbf{x}_t)}||u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t)||^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big|\Big|(1 - x_{t,k})(\mathbf{x}_t \odot \mathbf{x}_1) - x_{t,k}(\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_1)) - (1 - x_{t,k})(\mathbf{x}_t \odot \mathbf{x}_\theta) + x_{t,k}(\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_\theta))\Big|\Big|^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big|\Big|\big[(1 - x_{t,k})(\mathbf{x}_t \odot \mathbf{x}_1) - (1 - x_{t,k})(\mathbf{x}_t \odot \mathbf{x}_\theta)\big] - \big[x_{t,k}(\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_1)) + x_{t,k}(\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_\theta))\big]\Big|\Big|^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big|\Big|(1 - x_{t,k})(\mathbf{x}_t \odot \mathbf{x}_1 - \mathbf{x}_t \odot \mathbf{x}_\theta) - x_{t,k}(\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_1) - \mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_\theta))\Big|\Big|^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big|\Big|(1 - x_{t,k})\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) - x_{t,k}\mathbf{x}_t \odot (\mathbf{1} - \mathbf{x}_1 - \mathbf{1} + \mathbf{x}_\theta)\Big|\Big|^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big|\Big|(1 - x_{t,k})\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) - x_{t,k}\mathbf{x}_t \odot (\mathbf{x}_\theta - \mathbf{x}_1)\Big|\Big|^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big|\Big|(1 - x_{t,k})\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta) + x_{t,k}\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta)\Big|\Big|^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} ||\mathbf{x}_t \odot (\mathbf{x}_1 - \mathbf{x}_\theta)||^2
\end{aligned} \tag{26}
$$

Clearly, by minimizing $\mathcal{L}_{\text{denoise}} = \mathbb{E}_{p_t(\mathbf{x}_t)}||\mathbf{x}_1 - \mathbf{x}_\theta||^2$, we also minimize the flow-matching loss such that

$$
\arg\min_\theta \left[\mathbb{E}_{p_t(\mathbf{x}_t)}||u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t)||^2\right] = \arg\min_\theta \left[\mathbb{E}_{p_t(\mathbf{x}_t)}||\mathbf{x}_1 - \mathbf{x}_\theta||^2\right] \tag{27}
$$

which proves the Theorem.

Theorem. *If $p_t(\mathbf{x}_t) > 0$ for all $\mathbf{x}_t \in \mathbb{R}^d$ and $t \in [0, 1]$, then $\mathcal{L}_{FM}$ and $\mathcal{L}_{GS\text{-}FM}$ are equal up to a constant that is not dependent on θ, which means*

$$
\nabla_\theta \mathcal{L}_{\text{GS-FM}} = \nabla_\theta \mathcal{L}_{\text{FM}} \tag{28}
$$

Proof. This proof extends that of (Lipman et al., 2022; Tong et al., 2023), which proved that the conditional flow matching loss $\nabla_\theta \mathcal{L}_{\text{CFM}} = \nabla_\theta \mathcal{L}_{\text{FM}}$ under similar constraints.

From Equation 26, we derived the conditional flow-matching loss

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

Taking the gradient with respect to θ, we have

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

Given that the 2-norm is bilinear, we have

$$
\begin{aligned}
\mathbb{E}_{p_t(\mathbf{x}_t)}||\mathbf{x}_t \odot \mathbf{x}_1||^2 &= \int_{\mathbf{x}_t} ||\mathbf{x}_t \odot \mathbf{x}_1||^2 p_t(\mathbf{x}_t) d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \int_{\mathbf{x}_1} ||\mathbf{x}_t \odot \mathbf{x}_1||^2 p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 d\mathbf{x}_t \\
&= \mathbb{E}_{p_1(\mathbf{x}_1), p_t(\mathbf{x}_t|\mathbf{x}_1)} ||\mathbf{x}_t \odot \mathbf{x}_1||^2
\end{aligned} \tag{29}
$$

We also have

$$\begin{aligned}
\mathbb{E}_{p_t(\mathbf{x}_t)}\langle \mathbf{x}_t\mathbf{x_1}, \mathbf{x_t}\mathbf{x}_\theta\rangle &= \int_{\mathbf{x}_t} \langle \mathbf{x}_t\mathbf{x_1}, \mathbf{x_t}\mathbf{x}_\theta\rangle p_t(\mathbf{x}_t) d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \left\langle \frac{\mathbf{x}_t \int_{\mathbf{x}_1} \mathbf{x_1} p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1}{p_t(\mathbf{x}_t)}, \mathbf{x}_t\mathbf{x}_\theta \right\rangle p_t(\mathbf{x}_t) d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \left\langle \mathbf{x}_t \int_{\mathbf{x}_1} \mathbf{x_1} p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1, \mathbf{x}_t\mathbf{x}_\theta \right\rangle d\mathbf{x}_t \\
&= \int_{\mathbf{x}_t} \int_{\mathbf{x}_1} \langle \mathbf{x}_t\mathbf{x_1}, \mathbf{x}_t\mathbf{x}_\theta\rangle \, p_t(\mathbf{x}_t|\mathbf{x}_1) p_1(\mathbf{x}_1) d\mathbf{x}_1 d\mathbf{x}_t \\
&= \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)}\langle \mathbf{x}_t\mathbf{x_1}, \mathbf{x_t}\mathbf{x}_\theta\rangle
\end{aligned} \tag{30}$$

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

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} ||u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t, t)||^2 \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big[-2\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} \langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot \mathbf{x}_\theta\rangle + \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t)} ||\mathbf{x}_t \odot \mathbf{x}_\theta||^2 \Big] \\
&= \frac{\lambda^2 \exp(2\lambda t)}{\tau_{\text{max}}^2} \Big[-2\nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} \langle \mathbf{x}_t \odot \mathbf{x}_1, \mathbf{x}_t \odot \mathbf{x}_\theta\rangle + \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} ||\mathbf{x}_t \odot \mathbf{x}_\theta||^2 \Big] \\
&= \nabla_\theta \mathbb{E}_{p_t(\mathbf{x}_t|\mathbf{x}_1), p_1(\mathbf{x}_1)} ||u_t(\mathbf{x}_t|\mathbf{x}_1) - u_t^\theta(\mathbf{x}_t, t)||^2
\end{aligned} \tag{31}$$

which concludes the proof.

B Score Matching Derivations

B.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 $p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}}\big[p_t(\mathbf{x}_t|\mathbf{x}_1)\big]$.

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

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) &= \frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t)}{p_t(\mathbf{x}_t)} \\
&= \frac{\nabla_{\mathbf{x}_t} \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}}\big[p_t(\mathbf{x}_t|\mathbf{x}_1)\big]}{p_t(\mathbf{x}_t)} \\
&= \frac{\nabla_{\mathbf{x}_t} \int_{\mathbf{x}_1} \big[p(\mathbf{x}_1) p_t(\mathbf{x}_t|\mathbf{x}_1)\big] d\mathbf{x}_1}{p_t(\mathbf{x}_t)} \\
&= \frac{\int_{\mathbf{x}_1} p(\mathbf{x}_1) \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t|\mathbf{x}_1) d\mathbf{x}_1}{p_t(\mathbf{x}_t)} \\
&= \frac{\int_{\mathbf{x}_1} p(\mathbf{x}_1) p_t(\mathbf{x}_t|\mathbf{x}_1) \frac{\nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t|\mathbf{x}_1)}{p_t(\mathbf{x}_t|\mathbf{x}_1)} d\mathbf{x}_1}{p_t(\mathbf{x}_t)} \\
&= \int_{\mathbf{x}_1} \underbrace{\frac{p_t(\mathbf{x}_t|\mathbf{x}_1) p(\mathbf{x}_1)}{p_t(\mathbf{x}_t)}}_{= p_t(\mathbf{x}_1|\mathbf{x}_t)} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1) d\mathbf{x}_1 \\
&= \mathbb{E}_{\mathbf{x}_1 \sim p_t(\mathbf{x}_1|\mathbf{x}_t)} \left[\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1)\right]
\end{aligned} \tag{32}$$

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 $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}_{\mathbf{x}_1 \sim p_{\text{data}}(\mathbf{x}_1)} \left[\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_1)\right]$.

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

$$\nabla_{x_i} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = \tau(t) - \tau(t)V \cdot \text{SM}\Big(\log(\delta_{ik} + \epsilon) - \tau(t)x_i \Big) \tag{33}$$

Proof. First, we start by defining the probability density of the ExpConcrete distribution. From (Maddison et al., 2016), we have

$$p(\mathbf{x}) = (V-1)!\tau^{V-1} \left(\sum_{i=1}^{V} \pi_j \exp(-\tau x_j) \right) \left(\prod_{i=1}^{V} \pi_i \exp(-\tau x_i) \right) \tag{34}$$

where x_i is defined as a logit from the ExpConcrete distribution

$$x_i = \frac{\log \pi_i + g_i}{\tau} - \log \sum_{j=1}^{V} \exp\left(\frac{\log \pi_j + g_j}{\tau} \right) \tag{35}$$

Taking the logarithm of the probability path, we have

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

Then differentiating with respect to the logit of a single token x_j, we get

$$\begin{aligned}
\nabla_{x_j} \log p_t(\mathbf{x}_t|\mathbf{x}_1) &= -\nabla_{x_i} \sum_{i=1}^{V} \tau x_i - \nabla_{x_i} V \log \sum_{j=1}^{V} \exp\Big(\log \pi_j - \tau x_j \Big) \\
&= -\tau - V \left(\frac{1}{\sum_{j=1}^{V} \exp(\log \pi_j - \tau x_j)} \right) \exp(\log \pi_i - \tau x_i)(-\tau) \\
&= -\tau + \tau V \underbrace{\left(\frac{\exp(\log \pi_i - \tau x_i)}{\sum_{i=1}^{V} \exp(\log \pi_j - \tau x_j)} \right)}_{\text{softmax}} \\
&= -\tau + \tau V \cdot \text{SM}\Big(\log \pi_i - \tau x_i \Big)
\end{aligned} \tag{36}$$

Introducing time-dependence with $\tau(t) = \tau_{\text{max}} \exp(-\lambda t)$ and target token dependence with $\pi_j = \delta_{jk} + \epsilon$, we have

$$\nabla_{x_i} \log p_t(\mathbf{x}_t|\mathbf{x}_1) = \tau(t) - \tau(t)V \cdot \text{SM}\Big(\log(\delta_{ik} + \epsilon) - \tau(t)x_i \Big) \tag{37}$$

B.2 Proof of Score Matching Propositions

Proposition 3. The gradient of the ExpConcrete log-probability density is proportional to the gradient of the Gumbel-softmax log-probability density such that $\nabla_{x_j}^{\text{GS}} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1) \propto \nabla_{x_j}^{\text{ExpGS}} \log p_\theta(\mathbf{x}_t|\mathbf{x}_1)$.

Proof. As derived in (Maddison et al., 2016), the explicit probability density of the Gumbel-Softmax distribution is defined as

$$p(\mathbf{x}) = (V-1)!\tau^{V-1} \left(\sum_{i=1}^{V} \frac{\pi_i}{x_i^\tau} \right)^{-V} \prod_{i=1}^{V} \left(\frac{\pi_i}{x_i^{\tau+1}} \right) \tag{38}$$

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

$$\begin{aligned}\log p(\mathbf{x}) &= \log[(V-1)!] + (V-1)\log\tau - V\log\sum_{i=1}^{V}\frac{\pi_i}{x_i^\tau} + \sum_{i=1}^{V}\log\left(\frac{\pi_i}{x_i^{\tau+1}}\right) \\ &= \log[(V-1)!] + (V-1)\log\tau - V\log\sum_{i=1}^{V}\frac{\pi_i}{x_i^\tau} + \sum_{i=1}^{V}\log\left(\pi_i\right) - (\tau+1)\sum_{i=1}^{V}\log(x_i) \end{aligned} \tag{39}$$

Now, we introduce dependence on time t to define the *conditional probability density* $p_t(\mathbf{x}_0|\mathbf{x}_1 = \mathbf{e}_k)$ as

$$\log p(\mathbf{x}) = \log[(V-1)!] + (V-1)\log\tau(t) - V\log\sum_{i=1}^{V}\frac{\pi_i}{x_i^{\tau(t)}} + \sum_{i=1}^{V}\log\left(\pi_i\right) - (\tau(t)+1)\sum_{i=1}^{V}\log(x_i)$$

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

$$\begin{aligned}\nabla_{x_j}^{\text{ExpGS}}\log p_t(\mathbf{x}_t|\mathbf{x}_1) &= \nabla_{x_j}\left(-V\log\sum_{i=1}^{V}\frac{\pi_i}{x_i^\tau}\right) - \nabla_{x_j}\left((\tau+1)\sum_{i=1}^{V}\log(x_i)\right) \\ &= -V\left(\frac{1}{\sum_{i=1}^{V}\frac{\pi_i}{x_i^\tau}}\right)\left(\frac{-\pi_j\tau}{x_j^{\tau+1}}\right) - \frac{\tau+1}{x_j} \\ &= \frac{\tau V}{x_j}\left(\frac{\pi_j x_j^{-\tau}}{\sum_{i=1}^{V}\pi_i x_i^{-\tau}}\right) - \frac{\tau+1}{x_j} \\ &= \frac{\tau V}{x_j}\left(\frac{\exp(\log(\pi_j x_j^{-\tau}))}{\sum_{i=1}^{V}\exp(\log(\pi_i x_i^{-\tau}))}\right) - \frac{\tau+1}{x_j} \\ &= \frac{\tau V}{x_j}\left(\frac{\exp(\log\pi_j - \tau x_j)}{\sum_{i=1}^{V}\exp(\log\pi_i - \tau x_i)}\right) - \frac{\tau+1}{x_j} \\ &= \frac{\tau V}{x_j}\text{SM}\big(\log\pi_i - \tau x_i\big) - \frac{\tau+1}{x_j} \\ &= \frac{1}{x_j}\Big(-\tau + \tau V\cdot\text{SM}\big(\log\pi_i - \tau x_i\big)\Big) - \frac{1}{x_j} \\ &= \frac{1}{x_j}\Big(\nabla_{x_j}^{\text{GS}}\log p_t(\mathbf{x}_t|\mathbf{x}_1)\Big) - \frac{1}{x_j}\end{aligned} \tag{40}$$

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 ExpConcrete distribution further amplifies the scores for tokens with low probabilities by dividing by x_j and subtracting x_j^{-1}.

C Model Architecture

C.1 Diffusion Transformer

To parameterize our flow and score matching models, we leverage the Diffusion Transformer (DiT) architecture (Peebles & Xie, 2022) which integrates time conditioning with adaptive layer norm (adaLN) and positional information with Rotary Positional Embeddings (RoPE) (Su et al., 2021). Our model consists of 32 DiT blocks, 16 attention heads, 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

D Experimental Details

D.1 Hyperparameter Selection

Maximum Temperature $\tau_{\max}$. 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 (i.e. $\psi_0(\mathbf{x}_t|\mathbf{x}_1) = \frac{\mathbf{1}}{V}$ when $\tau_{\max} \to \infty$. Empirically, we find that setting $\tau_{\max} = 10.0$ 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 $\tau(t) = \tau_{\max} \exp(-3.0t)$. 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 inference 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 Gumbel-Softmax SM, we found the best performance for noise factors ranging between $\beta = 6.0 \to 8.0$, which produced high diversity while preserving foldable protein structures. For Gumbel-Softmax FM, we found the best performance for much larger noise factors $\beta = 1000.0 \to 2000.0$, which still had high diversity.

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 number of steps needed to converge. In contrast, larger step sizes $0.1 \leq \eta \leq 1.0$ increases speed of convergence but may overstep the high-density regions. Empirically, we found that a step size of $\eta = 0.5$ is optimal with the number of integration steps $N_{\text{steps}} = 100$.

D.2 Protein Evaluation Metrics

We evaluate protein generation quality based on the following metrics computed by ESMFold (Lin et al., 2023b).

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 high similarity between a predicted structure and a hypothetical true structure.

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. **Pseudo-perplexity** is calculated using ESM2 which computes the *feasibility* of a protein sequence based on the predicted probability of each token based on the previous tokens via the equation:

$$\text{PPL}(x) = \exp\left(-\frac{1}{L}\sum_{\ell=1}^{L}\log p(x_\ell|x_{<\ell})\right)$$

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

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

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

E Implementation Details

Here, we provide the PyTorch inference procedure for Gumbel-Softmax FM, which contains the following functions:

1. `noisy_softmax`: given a `(B, L, V)` tensor of logits for each sequence position, this function adds i.i.d Gumbel noise values to each token and sequence dimension and scales down by a temperature parameter generated by transforming `expanded_t` by the temperature function $\tau(t) = \tau_{\text{max}} \exp(-\lambda t)$.
2. `get_velocity_field`: given a `(B, L, V)` tensor of noisy logits at time t and clean logits (either one-hot from the ground-truth sequence during training or the predicted denoised logits), this function returns a `(B, L, V)` tensor of velocities computed according to Equation 21.
3. `flow_inference`: flow-matching sampling of sequences from near uniform distribution generated with temperature τ_{max} by iteratively sharpening the distribution. For N_{steps} uniformly distributed time steps between $t = 0 \to 1$, the function computes the velocity field given the current logits `xt` using the `get_velocity_field` function where `seq_one_hot` is set to the denoising model output. The `argmax` tokens of the final distribution are returned.

Listing 1: Gumbel-Softmax Flow Matching Inference

```
@torch.no_grad()
def flow_inference(self, x0, eps=1e-10):
    B, L, V = x0.shape

    x0 = self.noisy_softmax(x0, torch.zeros_like(x0))

    xt = x0.clone()
    xt = xt.to(self.device)

    # determine uniformly spaced time steps between 0 and 1
    t_span = torch.linspace(0, 1, self.args.num_integration_steps, device
        =self.device)

    vocab_mask = torch.full_like(xt, float('-inf'), device=self.device)
    vocab_mask[:, :, 4:23] = 0.0
```

```
    for i, (s, t) in enumerate(zip(t_span[:-1], t_span[1:])):

        expanded_t = t.unsqueeze(-1).unsqueeze(-1).expand(B, L, V)
        temp = self.temperature_function(expanded_t)

        model_out, error = self.forward(xt, t[None].expand(B))
        model_out = model_out.to(self.device)
        error = error.to(self.device)

        pred_velocity = self.get_velocity_field(xt, model_out, expanded_t
            )
        pred_velocity = pred_velocity.to(self.device)

        step_size = t - s
        assert pred_velocity.shape == xt.shape, "shape mismatch"

        xt = (xt + (step_size * pred_velocity)).to(self.device)
        xt = self.noisy_softmax(xt, temp)

    x1 = torch.argmax(xt, dim=-1)
    seq = tokenizer.decode(x1.squeeze()).replace(" ", "")
    return seq

def noisy_softmax(self, xt, expanded_t, noise_scale=10.0, eps=1e-20):
    device = xt.device
    B, L, V = xt.shape
    vocab_mask = torch.full_like(xt, float('-inf'), device=self.device)
    vocab_mask[:, :, 4:23] = 0.0

    temp = self.max_temp * torch.exp(-self.decay * expanded_t)

    U = torch.rand(B, L, V).to(device)
    gumbel_noise = -torch.log(-torch.log((U / noise_scale) + eps) + eps).
        to(device)

    xt = (xt + gumbel_noise) / temp
    xt = torch.softmax(xt + vocab_mask, dim=-1)

    return xt

def get_velocity_field(self, xt, seq_one_hot, expanded_t):
    B, L, V = xt.shape

    scaling = ((self.decay * torch.exp(self.decay * expanded_t)) / self.
        max_temp).float().to(xt.device)

    ones = torch.ones_like(seq_one_hot).float().to(xt.device)

    inner_prod_target = torch.sum(xt * (ones - seq_one_hot), dim=-1,
        keepdim=True) # (B, L, 1)
    inner_prod_rest = torch.sum(xt * (- seq_one_hot), dim=-1, keepdim=
        True)

    velocity_target = scaling * xt * inner_prod_target * seq_one_hot
    velocity_rest = scaling * xt * inner_prod_rest * (torch.ones_like(
        seq_one_hot) - seq_one_hot)
    velocity = velocity_target + velocity_rest

    assert velocity.shape == xt.shape, "shapes do not match"
    return velocity
```

Following a similar structure, we provide the PyTorch inference procedure for Gumbel-Softmax SM, with the following functions:

1. `convert_to_expconcrete`: given a `(B, L, V)` tensor of logits `xt`, this function applies the ExpConcrete transformation with scaled Gumbel noise and time-dependent temperature according to Equation 14.
2. `score_inference`: score matching sampling from near uniform distributions generated with temperature τ_{max} by iteratively sharpening the distribution. For N_{steps} uniformly distributed time steps between $t = 0 \rightarrow 1$, the function computes the predicted score function using the parameterization described in Equation 16 and adds score scaled by `step_size` to the ExpConcrete distribution. The `argmax` tokens of the final distribution are returned.

Listing 2: Gumbel-Softmax Score Matching Inference

```
@torch.no_grad()
def score_inference(self, x0, eps=1e-10):
    B, L, V = x0.shape

    temp = self.temperature_function(torch.zeros_like(x0))
    x0 = x0.to(self.device)

    xt = x0.clone()
    xt = xt.to(self.device)

    # determine uniformly spaced time steps between 0 and 1
    t_span = torch.linspace(0, 1, self.args.num_integration_steps, device
        =self.device)

    vocab_mask = torch.full_like(xt, float('-inf'), device=self.device)
    vocab_mask[:, :, 4:23] = 0.0

    for i, (s, t) in enumerate(zip(t_span[:-1], t_span[1:])):
        expanded_t = t.unsqueeze(-1).unsqueeze(-1).expand(B, L, V)
        temp = self.temperature_function(expanded_t)

        xt = torch.softmax(xt, dim=-1)
        xt = self.convert_to_expconcrete(xt, temp)

        pred = self.forward(xt, t[None].expand(B)).to(self.device)

        score = -temp + (temp * V * torch.softmax(pred, dim=-1))

        step_size = 0.5

        assert score.shape == xt.shape, "shape mismatch"

        xt = (xt + (score * step_size)).to(self.device)

    xt = torch.softmax(xt + vocab_mask, dim=-1)
    x1 = torch.argmax(xt, dim=-1)
    seq = tokenizer.decode(x1.squeeze()).replace(" ", "")
    return seq

def convert_to_expconcrete(self, xt, temp, noise_scale=10.0, eps=1e-20):
    device = xt.device
    B, L, V = xt.shape

    log_logits = torch.log(torch.clamp(xt, min=eps, max=1.0)).to(device)

    U = torch.rand(B, L, V).to(device)
    gumbel_noise = -torch.log(-torch.log((U / noise_scale) + eps) + eps).
        to(device)

    xt = (log_logits + gumbel_noise) / temp
    xt = xt - torch.logsumexp(xt, dim=-1, keepdim=True)

    return xt
```

F 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 refer to $\mathbf{x}$ as a single token in a sequence for simplicity, but in practice, the training and sampling is 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 denoising model $\mathbf{x}_\theta(\mathbf{x}_t, t)$: $\Delta^{V-1} \times [0,1] \to \Delta^{V-1}$ that takes the noisy sequence $\mathbf{x}_t$ at time t and returns the predicted clean distribution, error prediction model $\mathcal{E}_\phi(\mathbf{x}_t, t)$, maximum temperature τ_{max}, decay rate λ, and learning rate η.
Output: Trained denoising model $\mathbf{x}_\theta(\mathbf{x}_t, t)$ and error prediction model $\mathcal{E}_\phi(\mathbf{x}_t, t)$
procedure TRAINING
  for $\mathbf{x}_1$ in batch **do**
    Sample $t \sim \text{Uniform}(0, 1)$
    Set $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    for all simplex dimensions $i = 1$ to V **do**
      Sample variable for Gumbel noise $\mathcal{U}_i \sim \text{Uniform}(0, 1)$
      Sample Gumbel noise $g_i = -\log(-\log(\mathcal{U}_i + \epsilon) + \epsilon)$
      Given the clean token $\mathbf{x}_1 = \mathbf{e}_k$, sample noisy sequence

$$x_{t,i} \leftarrow \frac{\exp\left(\frac{\log(\delta_{ik}+\epsilon)+g_i}{\tau(t)}\right)}{\sum_j \exp\left(\frac{\log(\delta_{jk}+\epsilon)+g_j}{\tau(t)}\right)}$$

    end for
    Predict $\mathbf{x}_\theta(\mathbf{x}_t, t) \leftarrow \text{DiT}_\theta(\mathbf{x}_t, t)$
    Optimize denoising loss $\mathcal{L}_{\text{denoise}} \leftarrow -\frac{1}{L}\sum_{\ell=1}^{L} \left|\left|\mathbf{x}_1 - \mathbf{x}_\theta(\mathbf{x}_t, t)\right|\right|^2$
    $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{denoise}}$
    Predict error $\mathcal{E}_\phi(\mathbf{x}_t, t) \leftarrow \text{MLP}_\phi(\mathbf{x}_t, t)$
    Optimize error prediction loss $\mathcal{L}_{\text{error}} = \frac{1}{L}\sum_{\ell=1}^{L} \left|\left|\mathcal{E}_\phi(\mathbf{x}_t, t) - ||\mathbf{x}_\theta(\mathbf{x}_t, t) - \mathbf{x}_1||^2\right|\right|^2$
    $\phi \leftarrow \phi + \eta \nabla_\phi \mathcal{L}_{\text{error}}$
  end for
end procedure

Algorithm 2 Unconditional Sampling from Gumbel-Softmax Flow-Matching

Inputs: Trained model $\mathbf{x}_\theta(\mathbf{x}_t, t)$ that takes the noisy sequence $\mathbf{x}_t$ at time t and returns a probability distribution, time step $\Delta t = \frac{1}{N_{\text{step}}}$
Output: A clean sample $\mathbf{x}$
procedure SAMPLING
  Sample uniform distribution $\mathbf{x}_0 \leftarrow \frac{\mathbf{1}}{V}$
  Set $\mathbf{x}_t \leftarrow \mathbf{x}_0$
  for $t = 0$ to 1 **do**
    Predict $\mathbf{x}_\theta \leftarrow \mathbf{x}_\theta(\mathbf{x}_t, t)$
    Predict $\mathcal{E}_\phi \leftarrow \mathcal{E}_\phi(\mathbf{x}_t, t)$
    Calculate conditional vector field

$$u_t^\theta(\mathbf{x}_t, t) \leftarrow \frac{\lambda \exp(\lambda t)}{\tau_{\text{max}}} \mathbf{x}_t \left(\mathbf{x}_\theta \langle \mathbf{x}_t, \mathbf{1} - \mathbf{x}_\theta \rangle + (\mathbf{1} - \mathbf{x}_\theta) \langle \mathbf{x}_t, -\mathbf{x}_\theta \rangle \right)$$

    Take step $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta t \cdot u_t^\theta(\mathbf{x}_t, t)$
  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 model $f_\theta(\mathbf{x}_t, t)$ that takes the noisy sequence $\mathbf{x}_t$ at time t and returns the score (direction of probability density increase), maximum temperature τ_{max}, decay rate λ, and step size η.
Output: Trained score model $s_\theta(\mathbf{x}_t, t)$
procedure TRAINING
  for $\mathbf{x}_1$ in batch **do**
    Sample $t \sim \text{Uniform}(0, 1)$
    Set $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    for all simplex dimensions $i = 1$ to V **do**
      Sample variable for Gumbel noise $\mathcal{U}_i \sim \text{Uniform}(0, 1)$
      Sample Gumbel noise $g_i = -\log(-\log(\mathcal{U}_i + \epsilon) + \epsilon)$
      Given the clean token $\mathbf{x}_1 = \mathbf{e}_k$, sample noisy sequence from ExpConcrete distribution

$$x_{t,i} \leftarrow \frac{\log(\delta_{ik} + \epsilon) + g_i}{\tau(t)} - \log \sum_j \exp\left(\frac{\log(\delta_{jk} + \epsilon) + g_j}{\tau(t)}\right)$$

    end for
    Predict $f_\theta(\mathbf{x}_t, t) \leftarrow \text{DiT}_\theta(\mathbf{x}_t, t)$
    Compute predicted score $s_\theta(\mathbf{x}_t, t) \leftarrow -\tau(t) + \tau(t)V \cdot \text{SM}\big(f_\theta(\mathbf{x}_t, t)\big)$
    Compute target score $\nabla_{x_{t,i}} \log p_t(\mathbf{x}_t) \leftarrow -\tau(t) + \tau(t)V \cdot \text{SM}\big(\log(\delta_{ik} + \epsilon) - \tau(t)x_{t,i}\big)$
    Optimize loss $\mathcal{L}_{\text{score}} \leftarrow -\frac{1}{L}\sum_{\ell=1}^{L} ||s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)||^2$
    $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}_{\text{score}}$
  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 η, noise factor β
Output: Trained score model $s_\theta(\mathbf{x}_t, t)$
procedure TRAINING
  $\mathbf{x}_0 \leftarrow \text{SM}\left(\frac{\log(\frac{1}{V}) + \mathbf{g}}{\tau_{\text{max}}}\right)$
  for $t = 0 \to 1$ **do**
    Convert logits to ExpConcrete with Gumbel noise $\mathbf{g} \sim \text{Gumbel}(0, 1)$

$$x_{t,i} \leftarrow \frac{\log(x_{t,i}) + \frac{g_i}{\beta}}{\tau(t)} - \log \sum_j \exp\left(\frac{\log(x_{t,j}) + \frac{g_j}{\beta}}{\tau(t)}\right)$$

    Calculate $\tau(t) \leftarrow \tau_{\text{max}} \exp(-\lambda t)$
    Predict $f_\theta(\mathbf{x}_t, t) \leftarrow \text{DiT}_\theta(\mathbf{x}_t, t)$
    Compute predicted score $s_\theta(\mathbf{x}_t, t) \leftarrow -\tau(t) + \tau(t)V \cdot \text{SM}\big(f_\theta(\mathbf{x}_t, t)\big)$
    $\mathbf{x}_t \leftarrow \mathbf{x}_t + \eta s_\theta(\mathbf{x}_t, t)$
    $\mathbf{x}_t \leftarrow \text{SIMPLEXPROJ}(\mathbf{x}_t)$
  end for
  $\mathbf{x}_1 \leftarrow \arg\max(\mathbf{x}_t)$
  return $\mathbf{x}_1$
end procedure

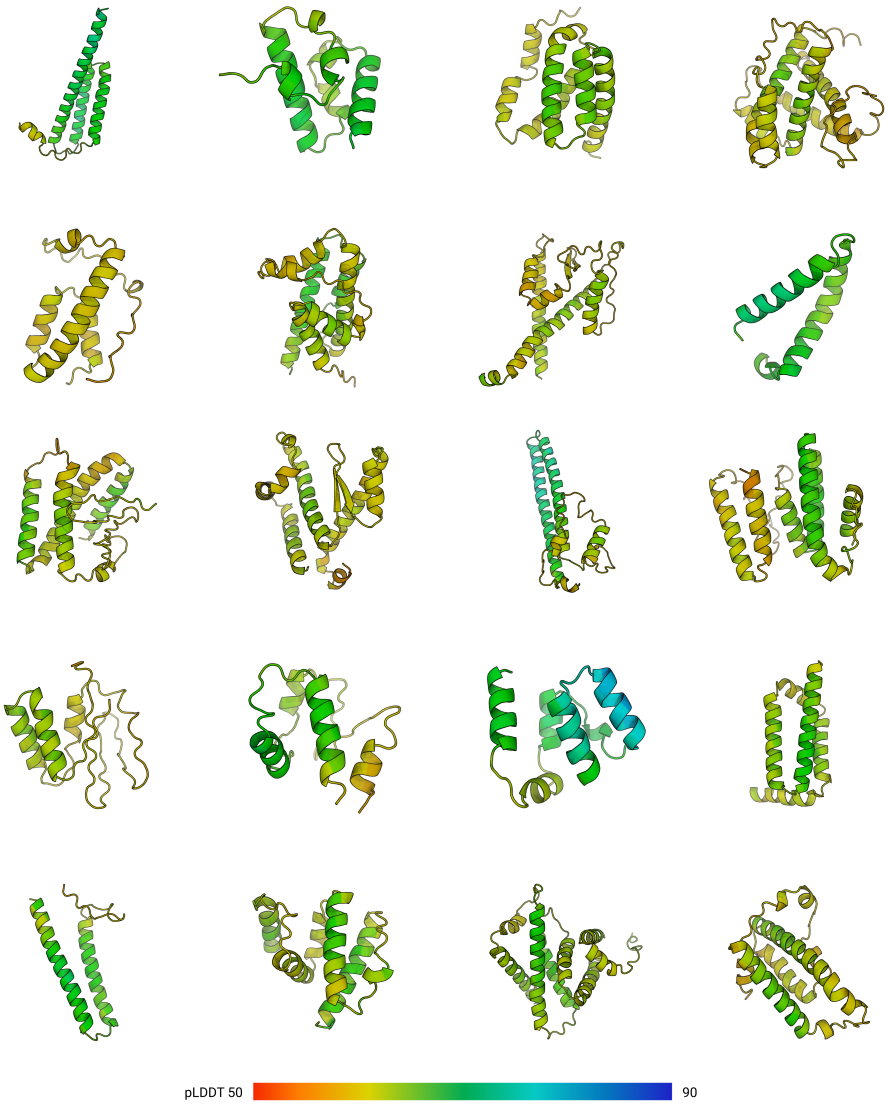

Figure 3: Predicted structures of *de novo* generated proteins with Gumbel-Softmax FM and SM, demonstrating diverse structural generation.