Discrete Diffusion Posterior Sampling for Protein Design

Mert Cemri [1] **Ajil Jalal** [1] **Kannan Ramchandran** [1]

Abstract

Designing new protein sequences that exhibit desirable functionality carries significant implications for medicine and biotechnology. Traditional methods for protein design have prominently comprised of experimental methods, such as in vitro-screening or animal experiments, which are costly and time-consuming. We propose a generative model based approach to protein sequence generation using guided discrete diffusion. We introduce a novel diffusion-based posterior sampling algorithm which uses a BERT-like transformer model to iteratively denoise discrete protein sequences. This approach demonstrates an efficient way to leverage an oracle that is trained to predict the desired functionality and can guide the protein generation procedure. Our experiments demonstrate that our method outperforms the state of the art, achieving higher functionality scores as well as higher ProtGPT2 likelihood scores.

1. Introduction

Protein design is revolutionizing medicine and biotechnology by enabling the creation of tailored proteins with specific functions, leading to significant advancements in various fields. For instance, custom-designed enzymes are being developed to break down environmental pollutants, offering a sustainable solution to bioremediation (Arnold, 2018). In medicine, engineered proteins are used to develop novel therapeutics, such as monoclonal antibodies that target specific cancer cells, improving the precision and effectiveness of cancer treatments (Carter & Lazar, 2018). Additionally, protein design is crucial in the development of vaccines, including those for emerging infectious diseases, by creating immunogens that elicit strong immune responses (Sanchez-Trincado et al., 2017).

[*]Equal contribution [1]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Correspondence to: Mert Cemri <cemri@berkeley.edu>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

Protein design is currently performed using a combination of computational and experimental techniques that allow scientists to predict and engineer proteins with desired properties. Computational methods, such as molecular modeling and simulations, play a crucial role in predicting protein structures and functions based on amino acid sequences (Khoury et al., 2014). Tools like Rosetta (Das & Baker, 2008) and AlphaFold (Jumper et al., 2021) are widely used to model protein folding and stability, enabling researchers to design proteins with high precision.

The standard approach in protein design is to handcraft an energy potential that maps 3D structure to the desired function, and this potential is used to guide the search for a suitable structure. However, this is fundamentally limited by the fact that even if we can find a suitable 3D structure, there is no guarantee that we can find an amino acid sequence with the desired 3D structure. Additionally, datasets for structural data are limited in size and the computational models tend to be slow and resource intensive. This motivates the more straightforward approach of searching directly in the space of amino acid sequences. To this end, we will focus on *generative models* that can learn a distribution over amino acid sequences in an unsupervised manner, and use these to design new sequences with guidance from supervised classifiers that can map sequences to function.

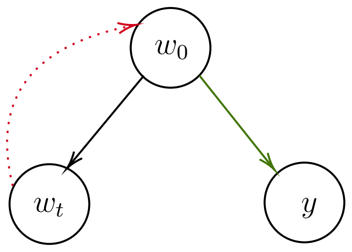

Figure 1. The solid green line denotes a probabilistic model that relates an amino acid sequence w_0 to its protein function y. The variable w_t denotes a noisy intermediate step in the diffusion process, and the dashed red line indicates the diffusion model that estimates $p(w_0|w_t)$.

Diffusion models (Ho et al., 2020; Song et al., 2020; Sohl-Dickstein et al., 2015) have achieved remarkable success in modelling various data modalities such as images (Rombach et al., 2022), audio (Kong et al., 2020), and text (Li et al., 2022). Starting from random noise, the model applies a sequence of learned denoising steps that gradually reduce the noise and reconstruct the original data. Each step predicts a slightly less noisy version of the data until the final output is a high-quality, coherent result that closely resembles the initial clean data. While sampling data using diffusion models is well studied (Chen et al., 2022; 2024), *guided sampling* is far more difficult, and recent work has shown that it is hard in general (Gupta et al., 2024). The fundamental challenge in guided sampling is that the potential functions that relate the sequence to the protein function are derived using experimental data from actual proteins and are not applicable to the noisy intermediate steps of the diffusion process.

Existing work that considers protein design using diffusion models (Gruver et al., 2024) handles this difficulty by training a classifier that relates the noisy sequences and the protein function, which unfortunately is not grounded by any experimental data. Our approach is to use the diffusion model itself to address this difficulty and is shown in Figure 1. Concretely, if w_0 denotes an amino acid sequence and y denotes a protein function, then we can relate them through some probabilistic model $p(y|w_0)$ (solid green line in Figure 1) that is trained using experimental data. Now, if w_t denotes a noisy intermediate sequence in the diffusion process, in order to apply guidance during the diffusion process, we need to relate w_t and y. This involves marginalizing over w_0 via

$$p(y|w_t) = \int p(y|w_0)p(w_0|w_t)\mathrm{d}w_0, \tag{1}$$

which is intractable in general. However, since the diffusion model is trained to estimate w_0 conditioned on w_t (dashed red line in Figure 1), we can use the posterior distribution $p(w_0|w_t)$ in an appropriate manner, for example, by sampling from it or computing the conditional expectation, in order to approximate the integral in Eqn (1). This allows us to incorporate the function $p(y|w_t)$ as guidance in the diffusion process and produce sequences that have high functional values.

2. Relevant Work and Background

Designing new proteins sequences can be viewed as a language generation task, where each token represents an aminoacid. In this sense, the discrete nature of protein sequences make sequence models such as recurrent neural networks or transformers very suitable to model them. Indeed, there have been prior works that proposed protein language models using these architectures, such as ProtGPT2 (Ferruz et al., 2022) and Progen (Madani et al., 2020). However, since the number of possible aminoacid combinations (for a sequence length of L, there are 20^L possible sequences where 20 is the number of unique aminoacids existing in nature (Branden & Tooze, 2012)), the success of using these protein language models to design protein sequences with desirable qualities and naturalness has been limited (Alamdari et al., 2023).

In order to tackle challenges of these sequence models, (Gruver et al., 2024) proposed a guided discrete diffusion model, called NOS (diffusioN Optimized Sampling). In NOS, the protein sequence generation process is viewed as the iterative refinement of a noisy sequence. To that end, a BERT-like transformer is trained to predict the output sequence given the current noisy sequence (Devlin et al., 2018).

2.1. Background

Let w_0 denote a real protein sequence that is sampled from the training set. Assuming that there is an alphabet $\mathcal{A}$ that denotes unique tokens to construct the protein sequences, it is possible to write $w_0 \in \mathcal{A}^L$ where L is the total number of tokens in the sequence. In discrete diffusion models, the discrete noise can be represented with $[MASK]$ tokens, as an analogue for the Gaussian noise in (Ho et al., 2020). For a diffusion process of T steps, the forward process can be modelled as iteratively replacing the amino-acids of the original sequence w_0 with the $[MASK]$ tokens, to obtain $w_0 \rightarrow w_t \rightarrow w_T = [MASK]^L$, where w_t is a noisy (partially masked) sequence of aminoacids at time t and w_T is a fully noisy (masked) sequence of length L.

In the unguided diffusion model, a transformer model p_θ can be trained conditioned on the current noisy sequence and time index to predict the clean sequence using the masked language modelling framework (MLM) in BERT (Devlin et al., 2018). This makes it possible to sample the clean estimate $\widehat{w_0} \sim p_\theta(\widehat{w_0}|w_t, t)$. Then, by noising this clean sequence estimate t times back, the noisy sequence at the next time index, w_{t-1}, can be obtained. After repeating this process for $t = [T, \ldots, 1]$, w_0 is returned as the predicted sequence. This transformer is trained by the maximizing the likelihood to denose the ground truth sequences and minimize the following loss:

$$L(\theta) = \mathbb{E}[-\log p_\theta(\widehat{w_0}|w_t, t)] \quad , w_t \sim p(w_t|w_0)$$

where $p(w_t|w_0)$ represents the forward noising process.

In order to incorporate guidance to this diffusion procedure, another classifier is trained to predict various functionalities of the amino acid sequence. Let $f(w_0)$ denote the oracle that provides the score of an arbitrary functionality of the real protein w_0 in the training data. NOS algorithm proposes to train a utility function v_ϕ is trained that aims to characterize the functionality score of the protein which can be used to guide the reverse diffusion sampling procedure. Note that since the aim to use the gradients of this function with respect to its inputs, it must be differentiable. Therefore, this function needs to work in continuous space. In that case, let

T_θ denote the encoder of the transformer p_θ that is trained above for the unguided diffusion procedure. Then, using this encoder, a continuous representation of the discrete sequence can be obtained $h_t = T_\theta(w_t)$. This utility function is trained by fitting it to the functionality scores of the real proteins via mean squared error regression:

$$\min_\phi \|v_\phi(h_t) - f(w_0)\|_2.$$

Having trained such a utility function, after defining $h_t^0 := h_t = T_\theta(w_t)$, the following gradient ascent steps are used in between the reverse diffusion sampling steps explained above to guide the reverse diffusion procedure, where

$$h_t^{i+1} = h_t^i + \eta \nabla_{h_t^i} \left(\lambda v_\phi(h_t^i) - \mathrm{KL}\left(p_\theta(\widehat{w_0}|h_t^0, t) || p_\theta(\widehat{w_0}|h_t^i, t)\right)\right) + z^i \tag{2}$$

for $i = 0, 1, \ldots, K-1$, where η is the step-size hyperparameter accounting for the overall step sizes of these updates and λ is the guidance strength, with higher λ trying to more aggressively guide the procedure with respect to the utility function, v_ϕ, and $z_i \sim N(0,1)$ is for regularization. Following these guidance steps, transformer model p_θ is used to obtain the clean estimate $\widehat{w_0} \sim p_\theta(\widehat{w_0}|h_t^K, t)$ and continue the reverse process by denoising $\widehat{w_0}$ $t-1$ steps. Note that this guided diffusion procedure is identical to unguided one if $K = 0$, since $\widehat{w_0} \sim p_\theta(\widehat{w_0}|h_t^0, t) = p_\theta(\widehat{w_0}|w_t, t)$.

One major drawback of this algorithm is that, the way the utility function v_ϕ is trained is by using a mean squared error loss against the true utility function of the ground truth protein sequences in the training data, which is obtained by calling a black-box oracle from the Biopython library (Chapman & Chang, 2000). However, this oracle is only meaningful if the denoised proteins are given to it. Therefore, putting w_t (noisy sequence) into the utility function as described above is not ideal since we cannot assume the existence of an oracle that quantifies functionalities of noisy sequences. Therefore, we need to follow a posterior sampling procedure to quantify the functionality of the noisy sequence w_t do guide the protein design process.

Diffusion Posterior Sampling (DPS) and Inverse Problems. Our method is inspired by Diffusion Posterior Sampling (Chung et al., 2022) (DPS) for solving inverse problems. In inverse problems, we are given observations (for example, a blurry image) y of some hidden data (for example, a high resolution image) x_0, along with the conditional distribution $q(y|x_0)$, which is typically modelled using knowledge of the physics of the observation process.

A recent approach to solving inverse problems is via *posterior sampling*, where the diffusion model captures the prior distribution of x_0, and x_0 can be estimated from the measurements y by sampling from the posterior distribution $p(x_0|y)$. Posterior sampling is done analogous to regular diffusion sampling: as diffusion sampling only requires access to the scores $\nabla_{x_t} \log p_t(x_t)$ to sample x_0, if we had access to the diffusion scores $\nabla \log p_t(x_t|y)$, we can sample from $p(x_0|y)$.

One way to compute the posterior scores using an unconditional diffusion model is via Bayes' rule, since we have

$$\nabla_{x_t} \log p_t(x_t|y) = \nabla_{x_t} \log q_t(y|x_t) + \nabla_{x_t} \log p_t(x_t).$$

Unfortunately, we only have knowledge of the distribution $q(y|x_0)$ for the *actual data*, and we do not have direct access to the distribution $q_t(y|x_t)$ for the intermediate noisy states in the diffusion. DPS addresses this issue by cleverly incorporating the diffusion model: as the diffusion model is trained to estimate $\mathbb{E}[x_0|x_t]$, we can approximate $\nabla q_t(y|x_t)$ via

$$\nabla_{x_t} \log q_t(y|x_t) \approx \nabla_{x_t} \log q_0(y \mid \mathbb{E}[x_0|x_t]).$$

This involves differentiating through the diffusion model, which poses a challenge when the data is discrete like the protein sequences we consider in this paper.

3. Proposed Method

We use our trained classifier v_ϕ to guide the generation process of new protein sequences in a manner similar to classifier guidance in inverse problems. In inverse problems, we usually have access to some measurement y thanks to some physical measurement process $p_0(y|x_0)$ of clean data x_0. However, we need access to the distribution $p_t(y|x_t)$ relating the measurements to the noisy intermediates x_t. Following the DPS framework (Chung et al., 2022), we can write the following formula:

$$\begin{aligned} p_t(y|x_t) &= \int p_t(y|x_0, x_t) p(x_0|x_t) dx_0 \\ &= \int p_0(y|x_0) p(x_0|x_t) dx_0 \\ &= \mathbb{E}_{x_0 \sim p(x_0|x_t)}[p_0(y|x_0)] \end{aligned} \tag{3}$$

While this expectation is intractable in general, we can switch the expectations to approximate it. Concretely, by defining $\widehat{x_0} := \mathbb{E}[x_0|x_t]$, we can write $p(y|x_t) \approx p(y|\widehat{x_0})$.

Now, let us make the relation of the above calculations to the protein sequence generation algorithm concrete: during the reverse process of our discrete diffusion procedure, at time t, we can first use the encoder of the transformer T_θ to obtain the continuous representation h_t of our discrete sequence w_t, and substitute h for x above. Then, what the above calculations show us is that when calling the utility function v_ϕ, it is better to use an estimate of the final denoised sequence

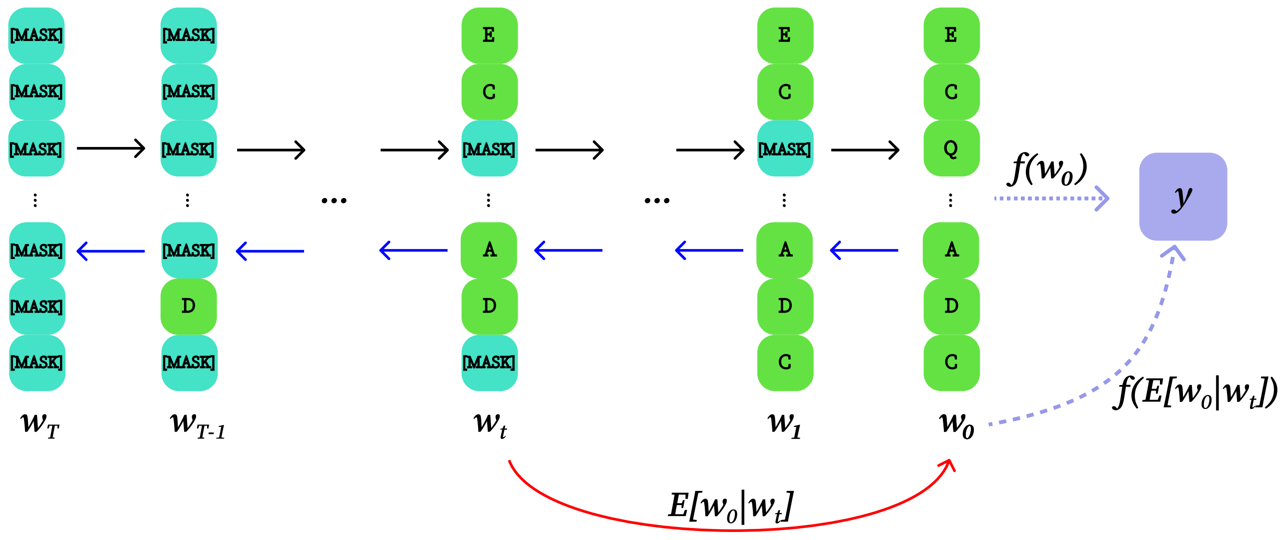

Figure 2. Discrete diffusion posterior sampling scheme. The blue arrows represent the forward process in discrete diffusion model, where we iteratively mask different tokens (amino acids) in the protein sequence, and the black arrows represent the reverse process. We can only make measurements with real proteins, hence we have some measurement oracle f that gives us the functionality values y of real protein sequences w_0. We de not have a reliable measurement model for evaluating noisy proteins w_t. If we wish to evaluate the functionality score of noisy proteins w_t, the correct approach is to sample the clean sequence via conditional expectation $\mathbb{E}[w_0|w_t]$, and use this sample to evaluate the functionality score.

given our current noisy sequence $\mathbb{E}[h_0|h_t] \approx T_\theta(w_t)$ where $w_t \sim H_\theta(h_t)$, where H_θ is the decoder of our transformer which outputs the discrete sequence given a continuous representation. Finally, we provide our proposed algorithm in Algorithm 1, where in lines 8 and 9 we show this posterior sampling step and line 10 is the Langevin step.

One crucial thing to note here is how this conditional expectation is computed. Essentially, we would like the output of the utility function, $v_\phi(\mathbb{E}[h_0|h_t])$ to be differentiable with respect to h_t. To do so, we first need to sample the clean discrete protein sequence $\widehat{w_0} \sim H_\theta(h_t) = p_\theta(\widehat{w_0}|w_t)$, where the equality follows from the fact that $h_t = T_\theta(w_t)$. Here, in order to be able to differentiate through the sampling step, we cannot simply construct a discrete probability distribution and then sample from this categorial distribution. Instead, we approximate the discrete probability distribution over the vocabulary of possible amino-acids $H_\theta(h_t)$ with a Gumbell-softmax distribution, which allows us to use the reparametrization trick and differentiate through the sampled $\widehat{w_0} \sim H_\theta(h_t)$ with respect to h_t (Jang et al., 2016). Then, having appropriately estimated the clean sequence $\widehat{w_0}$, we obtain its continuous representation to finally get $\widehat{h_0} = \mathbb{E}[h_0|h_t] = T_\theta(\widehat{w_0})$.

4. Experiments

4.1. Experimental Setup

For the experiments, we utilized the dataset provided in (Gruver et al., 2024), which includes 100000 different example sequence of hu4D5, a therapeutic antibody targeting the HER2 antigen (HER2 is an important target for certain types of breast and stomach cancer). These examples are already split into training and validation set with a 90:10 ratio, and we keep the same selection in our experiments.

We investigated three different experimental cases. First, we tried to maximize a single-objective, that is the percentage of β-sheets in the protein, which are one of the two essential building blocks of proteins, and the value of the solvent-accessible surface area (SASA) in Section 4.2. Then, we did a multi-objective optimization experiment, where we tried to optimize both the β-sheets and α-helices of our proteins (alpha helices are the second type of building block of the secondary structure of a protein) simultaneously, and generate proteins that would have good qualities in both objectives, in Section 4.3. Finally in Section 4.4, we investigate the infilling experiments, where do not start from an all masked sequence, but we start from a partially known sequence only few of whose tokens are initially masked.

For all of the comparisons in the following subsections, we train our proposed model and the NOS model for 100 epochs

Algorithm 1 Discrete Diffusion Posterior Sampling

```
1: Inputs: Denoising transformer $p_\theta(\widehat{w_0}|h_t, t) = [T_\theta, H_\theta]$, utility value function $v_\phi$, and hyperparameters $\eta$, $\lambda$
2: $w_T = [\text{MASK}]^L$
3: for $t = T, \dots, 1$ do
4:     $h_t^0 \leftarrow T_\theta(w_t)$
5:     for $i = 0, \dots, K-1$ do
6:         $z^i \sim \mathcal{N}(0, I)$
7:         $\widehat{w_0} \sim H_\theta(\widehat{w_0}|h_t^i)$
8:         $\widehat{h_t^i} \leftarrow T_\theta(\widehat{w_0})$
9:         $h_t^{i+1} \leftarrow h_t^i + \eta \nabla_{h_t^i} \left(\lambda v_\phi(\widehat{h_t^i}) - \text{KL}\left(p_\theta(\widehat{w_0}|h_t^0, t) || p_\theta(\widehat{w_0}|h_t^i, t)\right)\right) + z^i$
10:    end for
11:    $p(w_{t-1}|H_\theta(h_t^K)) = \sum_{\tilde{w}} p(w_{t-1}|\tilde{w}) H_\theta(\tilde{w}|h_t^K)$
12:    $w_{t-1} \sim p(w_{t-1}|H_\theta(h_t^K))$
13: end for
14: return $w_0$
```

with batch size 64, learning rate at 0.005 and 5 guidance update steps (K in Algorithm 1), to be compatible with the hyperparamters suggested in the original NOS paper (Gruver et al., 2024). We used two NVIDIA RTX A5000 GPUs as compute resources. We trained our models on those machines and ran the inferences using these GPUs.

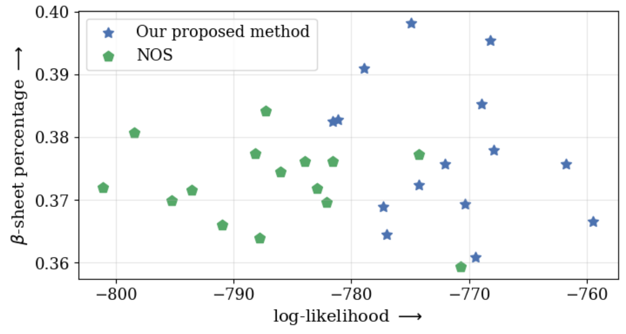

Figure 3. Protein generation experiments where we tried to optimize for β-sheet percentages of the generated proteins. A higher y-axis score indicates better quality, a higher x-axis score indicates more naturalness. Blue stars are our method, green hexagons belong to the baseline.

4.2. Single-objective Experiments

We train the NOS model and our proposed model with the objective of maximizing the the percentage of β-sheets and solvent-accessible surface area (SASA), respectively one at a time. The results are provided in Figures 3 and 4, respectively.

To compare the performance of these algorithms, we sample 10 proteins using different hyperparameters: we change the step size (η) and stability coefficient (λ) parameters in line 10 of Algorithm 1. We try 15 different combinations with the hyperparameters given in Table 1.

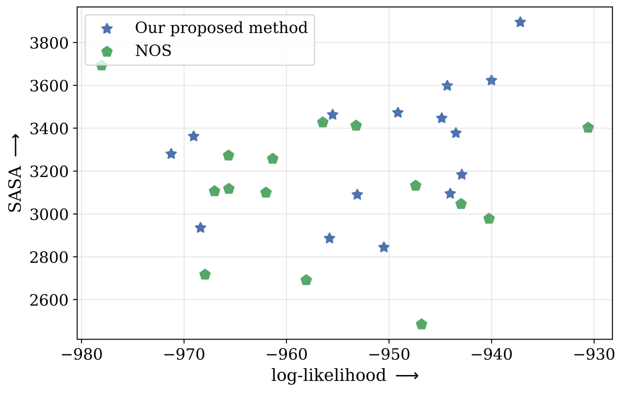

Figure 4. Protein generation experiments where we tried to optimize for solvent accessible surface area (SASA) of the generated proteins. A higher y-axis score indicates better quality, a higher x-axis score indicates more naturalness. Blue stars are our method, green hexagons belong to the baseline.

Then, for each of these hyperparameter combinations, we take the mean of the beta-sheet percentages and the log-likelihood (which is obtained using ProtGPT2 (Ferruz et al., 2022), a very capable protein language model characterizing the likelihood of a protein sequence), and plot the scatter plots in Figures 3 and 4.

Note that in both figures, our algorithm can provide protein sequences with higher numbers of desired functionality on average (beta sheet percentages or SASA values). Moreover, while doing so, we also obtain protein sequences that are on average more likely to exist in real-life, since a higher log-likelihood can be interpreted as proteins being more natural (Ferruz et al., 2022).

Table 1. Step sizes and stability coefficients for different hyperparameter combinations in experiments

Hyperparameter Name	**Value**
Step Size (η)	0.1, 0.5, 1.0
Stability Coefficient (λ)	0.001, 0.01, 0.1, 1.0, 10.0

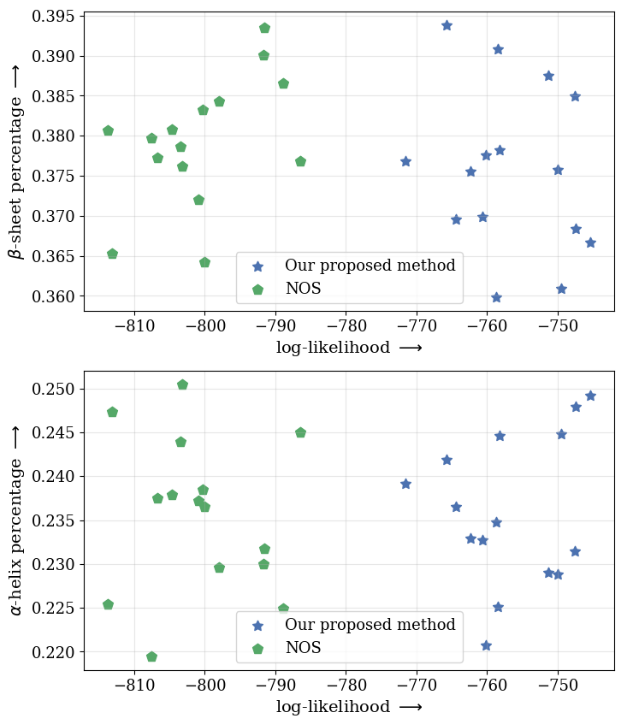

Figure 5. Multi-objective optimization experiments where we tried to optimize for β-sheet and α-helix percentages and of the generated proteins at the same time. A higher y-axis score indicates better quality, a higher x-axis score indicates more naturalness.

4.3. Multi-objective Experiments

For multi-objective optimization experiments, we train our proposed method and NOS to maximize both β-sheet and α-helix percentages in the protein. We again generate multiple instances of protein sequences for 15 different hyperparameter combinations using the values in Table 1, and plot the mean values of different sampled proteins for each of these combinations in Figure 5. Note that in these experiments, the algorithms perform on par in terms of utility functions of the generated proteins, but proteins designed by our algorithm have much higher log-likelihood, suggesting they are more likely to occur in nature while preserving similar qualities.

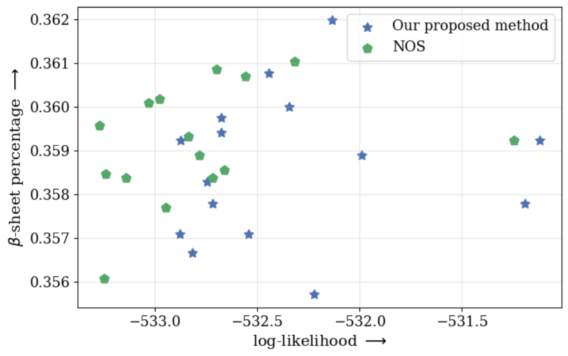

Figure 6. Single-objective infilling optimization experiments where we tried to optimize for β-sheet percentages of the generated proteins where we start protein generation from a partially known sequence. A higher y-axis score indicates better quality, a higher x-axis score indicates more naturalness.

4.4. Infilling Experiments

For the infilling experiments, instead of starting the diffusion process with a fully masked sequence as in Algorithm 1, we start from a known sequence that has 300 amino acids, 24 of which are masked. Then, we let our transformer generate predictions only for these 24 aminoacids. We do this experiment for both single-objective optimization where we aim to optimize for the percantage of β-sheets, which is shown in Figure 6. Notice that the log-likelihoods of the proteins in Figures 6 are much higher than the ones in 3 and 5, providing a sanity check for us, since in these experiments we start from partially known sequences so it is expected for them to have higher log-likelihood scores.

5. Conclusion and Future Work

In this paper, we proposed a novel algorithm where we improved upon the guided discrete diffusion procedure by incorporating ideas from inverse problems and diffusion posterior sampling. We showed that we can achieve better results in terms of desired qualities and the final likelihood of the generated proteins when we use appropriate posterior sampling steps to guide the discrete diffusion algorithm better.

One limitation of this work is that we still need to obtain continuous representations of the discrete protein sequences in order to execute the Langevin and posterior sampling steps. Works such as (Lou et al., 2023) have recently proposed a new discrete diffusion algorithm to circumvent this procedure. Therefore, harmonizing the ideas in that paper with this study could be an interesting future direction. Moreover, "in-vitro" experiments by synthesizing actual proteins will help validate our our algorithm on real-world design tasks.

References

Alamdari, S., Thakkar, N., van den Berg, R., Lu, A. X., Fusi, N., Amini, A. P., and Yang, K. K. Protein generation with evolutionary diffusion: sequence is all you need. *bioRxiv*, pp. 2023–09, 2023.

Arnold, F. H. Directed evolution: bringing new chemistry to life. *Angewandte Chemie (International Ed. in English)*, 57(16):4143, 2018.

Branden, C. I. and Tooze, J. *Introduction to protein structure*. Garland Science, 2012.

Carter, P. J. and Lazar, G. A. Next generation antibody drugs: pursuit of the'high-hanging fruit'. *Nature Reviews Drug Discovery*, 17(3):197–223, 2018.

Chapman, B. and Chang, J. Biopython: Python tools for computational biology. *ACM Sigbio Newsletter*, 20(2): 15–19, 2000.

Chen, S., Chewi, S., Li, J., Li, Y., Salim, A., and Zhang, A. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations*, 2022.

Chen, S., Chewi, S., Lee, H., Li, Y., Lu, J., and Salim, A. The probability flow ode is provably fast. *Advances in Neural Information Processing Systems*, 36, 2024.

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Das, R. and Baker, D. Macromolecular modeling with rosetta. *Annu. Rev. Biochem.*, 77:363–382, 2008.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

Gruver, N., Stanton, S., Frey, N., Rudner, T. G., Hotzel, I., Lafrance-Vanasse, J., Rajpal, A., Cho, K., and Wilson, A. G. Protein design with guided discrete diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

Gupta, S., Jalal, A., Parulekar, A., Price, E., and Xun, Z. Diffusion posterior sampling is computationally intractable. *arXiv preprint arXiv:2402.12727*, 2024.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Khoury, G. A., Smadbeck, J., Kieslich, C. A., and Floudas, C. A. Protein folding and de novo protein design for biotechnological applications. *Trends in biotechnology*, 32(2):99–109, 2014.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2020.

Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., and Hashimoto, T. B. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

Lou, A., Meng, C., and Ermon, S. Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Madani, A., McCann, B., Naik, N., Keskar, N. S., Anand, N., Eguchi, R. R., Huang, P.-S., and Socher, R. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Sanchez-Trincado, J. L., Gomez-Perosanz, M., Reche, P. A., et al. Fundamentals and methods for t-and b-cell epitope prediction. *Journal of immunology research*, 2017, 2017.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.