nature biotechnology

Supplementary information

https://doi.org/10.1038/s41587-026-03207-z

Property guidance for protein sequence generative models with ProteinGuide

In the format provided by the authors and unedited

Supplementary Materials for Property guidance for protein sequence generative models with ProteinGuide

Junhao Xiong*, Ishan Gaur*, Maria Lukarska*, Hunter Nisonoff*,
Luke M. Oltrogge, David F. Savage†, Jennifer Listgarten†
†Corresponding authors. Email: savage@berkeley.edu, jennl@berkeley.edu
*These authors contributed equally to this work.

This PDF file includes:

Supplementary Text
Supplementary Tables
Supplementary Figures

Contents

1 Supplementary Text

1.1 Detailed discussion on related work

Here we provide a more in-depth discussion on the related work to this work. Since ProteinGuide brings techniques that are broadly applicable to generative models to the problems in protein engineering, we broadly categorized the related literature into two categories: (i) those that are more related to the technical underpinning of guidance (see subsection "Related work to guidance"), and (ii) those that are more related to the use of pre-trained models and experimental data in protein engineering (see subsection "Related work in protein engineering"). For related work that is more specific to the results showing the equivalence of sampling methods used for discrete flow matching and AO-ARMs, see the Methods section "Simplified and faster sampling for flow matching and diffusion models with masked noise processes".

1.1.1 Related work to guidance

Several recent works have proposed related but different techniques from this work for steering discrete state-space diffusion models. These approaches can be broadly categorized into on-the-fly approaches (which modify the sampling process at inference time) and training-based approaches (which update the parameters of the pre-trained model).

On-the-fly approaches. Historically, the general problem of sampling from challenging distributions has also been tackled through Markov chain Monte Carlo (MCMC) methods, and some MCMC-based methods have also been proposed for guiding discrete-space generative models. Emami et al. [1] proposed PPDE, which uses the Metropolis–Hastings (MH) algorithm to sample from product-of-experts (PoE) distributions on protein sequence space, where the target is defined via an (unnormalized) sequence-level score combining an "evolutionary density" expert and one or more property experts. The key distinctions between that approach to ProteinGuide are that PPDE requires taking gradients through the pre-trained model, which may be prohibitively large; it requires computing (unnormalized) likelihood from the generative model, which is very expensive for most PLMs; and, as it can be quite sensitive to its initialization, potentially having a mixing time that scales exponentially in the sequence length. In contrast, ProteinGuide does not require gradients (except optionally through the predictor), does not require likelihood computations, runs in linear time in sequence length, and does not require a wild-type sequence to initialize sampling. We do note that PPDE can be a useful technique to *refine* generations from ProteinGuide, particularly when one desires additional local improvements from a good starting point (e.g., a wild-type or a strong candidate) after generating more diverse candidates with ProteinGuide.

Some recent works [2, 3] have also proposed techniques for guiding discrete-space diffusion models using Sequential Monte Carlo (SMC), building upon works first proposed for continuous-space diffusion models [4]. Importantly, ProteinGuide is also complementary to SMC-based approaches as these techniques can be combined with ProteinGuide to improve generation with some additional inference-time costs, such as by way of using multiple particles with proposal distributions defined via TAG. Indeed, through the lens of MCMC, one can also view ProteinGuide as a way to design proposal distributions with minimal rejections by exploiting the structures of the underlying generative process, in a similar spirit to other discrete-space MCMC methods [5]. Lee et al. [6] observed that when applying stronger guidance, sampling can also be improved with SMC, aligning with challenges in low-temperature sampling noted for continuous-space diffusion models [7, 8]. For autoregressive (AR) models, Zhao et al. [9] proposed an SMC-based steering method that learns

intermediate twist functions via contrastive learning, similar to learning time-dependent predictors in diffusion guidance.

Vignac et al. [10] propose DiGress, an approximate guidance approach in a discrete-time, discrete state-space diffusion (D3PM) [11] framework, which Wang et al. [12] (DPLM) apply to generate protein sequences. In our earlier work [13], we proposed methods for both exact and Taylor-approximate Guidance (TAG), the second of which bears resemblance to DiGress due to both of them using a Taylor series approximation. In our earlier work [13], we discussed the difference between DiGress/DPLM and Discrete Guidance in detail, and performed comprehensive comparisons to DiGress in three experiments across different domains, generally showing Discrete Guidance to be superior. Here we briefly discuss the distinction and refer the readers to Nisonoff et al. [13] (*e.g.*, Appendix section E.2) for more details. DiGress is fundamentally a method for discrete-time, discrete-space diffusion models. It does not specify how to guide either continuous-time, discrete-space diffusion models or discrete-space flow-matching models. ProteinGuide naturally encompasses these model classes, and through our equivalence can be applied to any-order autoregressive models (AO-ARMs). Both theoretically and empirically, the continuous time training objective has been demonstrated to perform better [14–17]. In fact, one could view DiGress as a special case of our general framework by taking a specific discrete time approximation to the continuous time process. We note that this approximation imposes a uniform time grid and is therefore distinct from our construction of *equivalent* discrete time processes for continuous time models. Accordingly, their formulation would not directly be applicable to an AO-ARM without additional assumptions. Orthogonally, Yang and Klein [18] proposed FUDGE for guiding autoregressive models, which can be viewed as a special case of our exact guidance method in the case of AR model, with AO-ARMs generalizing AR models.

As noted in Schiff et al. [19], which also presented a discrete-time analog of the continuous-time guidance method in Nisonoff et al. [13], in discrete time, the normalizing constant in Equation 14 is still intractable, necessitating additional assumptions to achieve tractability, whereas Nisonoff et al. [13] showed TAG does not require this assumption. In addition, the continuous-time formulation of discrete diffusion models provides exactly the tractability necessary to perform *exact* guidance, whereas DiGress is fundamentally approximate. ProteinGuide enables exact guidance both for general noising processes [13] and for masking noise process. In particular, we showed that the model classes discussed—masked diffusion/flow models, masked language models (MLMs), and any-order autoregressive models (AO-ARMs)—are not only equivalent in their training objective, but also the ways they can be sampled (see Methods section "Simplified and faster sampling for flow matching and diffusion models with masking noise processes"). Through this same equivalence, we develop a simpler and faster algorithm for sampling sequences from diffusion and flow-matching models that use masking noise processes, leading to a more computationally efficient guidance algorithm than the one originally proposed in Nisonoff et al. [13], which we called *discrete-time exact guidance (DEG)*. The original method was already shown to at times improve performance over TAG, but we further employed DEG to our *in silico* experiments to illustrate its effectiveness (as seen in the enzyme and multi-property guidance experiments, where TAG was less effective).

There is another line of work that models discrete data by embedding the data into continuous state-spaces, either on a probabilistic simplex [20, 21] or on a continuous latent space [22, 23]. Some of the approaches proposed to apply guidance to diffusion models on the continuous state-space representations of discrete state-space objects in order to utilize the guidance approaches for continuous state-space diffusion. We note that approaches which perform guidance in continuous spaces lose the discrete structure of the data during generation, which can be important for settings where the discrete structures contain information that is useful for guidance [16]. For more detailed discussion for other inference-time techniques, we refer the readers to a recent review article by Uehara

et al. [24].

Training-based approaches. Recent works have also explored updating model parameters to incorporate conditioning rather than modifying the sampling process. Wang et al. [25] proposed to use the Gumbel-Softmax trick to make discrete diffusion trajectories differentiable and enable backpropagation of rewards through entire sampling paths. Concurrently, Rector-Brooks et al. [26] proposed to frame the steering problem as sampling from a Bayesian posterior where the pre-trained model serves as the prior and the reward acts as the likelihood. Both fine-tuning approaches established connections to our earlier work [13] while offering complementary perspectives on solving the steering problem. Direct Preference Optimization (DPO [27]) and related work [28, 29] enable updating of pre-trained models, although up until recently, these methods have been applicable only to models with tractable likelihoods, thereby restricting them largely to autoregressive models. DPO was recently extended to diffusion models on continuous spaces by way of approximation [30], and then to discrete spaces [31]. Further research is needed to understand and improve the quality of such approximations.

Reinforcement Learning from Human Feedback (RLHF) offers an alternative approach to model alignment through methods like Proximal Policy Optimization (PPO) [32]. PPO does not technically require the retrained model to have tractable likelihoods, but it requires an explicit reward model and on-policy training, making it often more complicated to train than more direct approaches like DPO [27]. In addition, the standard RLHF pipeline typically begins with fine-tuning, for which the statistical outcome is not well-defined, and for which important information from the already-trained model may be forgotten. For a more detailed review of training-based methods, specifically those anchored in reinforcement learning and diffusion models, we refer the readers to a review article by Uehara et al. [33].

1.1.2 Related work in protein engineering

With the increasing interests of using generative models—both for structures [8, 34] and for sequences [35–39]—in protein engineering, many of the methods mentioned above have seen adaptations to protein engineering problems, some of which are summarized in a recent review article by Stocco et al. [40]. Below we categorized these applications by the types of methods they employed.

Recent work by Yang et al. [41] benchmarked Discrete Guidance [13] when used with family-specific pre-trained models on a set of protein fitness optimization tasks. They found that Discrete Guidance outperforms several fine-tuning and other "plug-and-play" guidance methods (*e.g.*, methods that perform guidance in continuous latent spaces). In addition, they found that guidance-based approaches require less hyperparameter tuning and have lower computational costs, making them more practically accessible to everyday users. However, these earlier works did not address how to perform guidance for models other than diffusion and flow matching, thereby limiting its utility to the field of protein engineering and beyond. In contrast, by unifying a broad class of discrete-space generative models under a single framework, we are able to directly apply guidance to popular pre-trained models commonly used by practitioners. Accordingly, we do not compare to many of the methods benchmarked in Yang et al. [41] again, but instead focus on demonstrating the efficacy of ProteinGuide on a wider breadth of challenging design scenarios and on a real design campaign.

On-the-fly approaches. In the context of protein sequence generative models, there are prior works that embed protein sequences into continuous state-spaces and thereby directly leveraging guidance techniques developed for continuous-space diffusion models. For example, Gruver et al.

[42] propose to perform guidance in the hidden layers of the unconditional diffusion model by jointly training the unconditional denoising model and the classifier on one hidden layer of the denoising model. In another example, Lisanza et al. [43] propose to fine-tune RoseTTAFold with Gaussian diffusion on the one-hot encoded sequence space in order to perform guidance in this continuous state-space.

Another style of approaches that have been very recently developed in the context of large language models involves directly manipulating the intermediate outputs of the model (*e.g.*, activations) [44], and it has only very recently been adapted to protein language models (PLMs) to steer generations toward targeted protein properties [45]. Concretely, these methods compute a "steering direction" in representation space (*e.g.*, from differences in hidden activations associated with an attribute) and then inject this direction into selected layers during the forward pass to bias the model's next-token (or sequence) distribution, without updating model parameters. Because this line of work is quite recent and protein properties can depend on highly nonlocal and context-dependent constraints, its robustness and generality across objectives, proteins, and model families remain an active area of research.

Training-based approaches. DPO has been applied to a few autoregressive protein generative models. For example, Widatalla et al. [28] applied DPO to ESM-IF1, an autoregressive inverse folding model, to incorporate experimental stability data. In another example, Stocco et al. [46] applied DPO to an autoregressive model conditionally trained on enzyme classes to guide the generation towards various enzyme properties. As for RLHF-based approaches, Blalock et al. [47] recently applied PPO to align protein language models (*e.g.*, ESM2) with experimental measurements. Notably, this approach does not directly sample from generative models in the conventional sense; instead, it employs a position-wise scoring strategy where models evaluate mutation probabilities at individual positions sequentially, followed by an iterative selection process that combines high-scoring mutations.

Zero-shot scoring-based approaches. A complementary line of work uses pre-trained protein language models primarily as *scoring* functions to guide iterative mutation and selection, rather than sampling from an explicit conditional generative model. For example, EVOLVEpro [48] and related methods use language-model-derived scores within iterative optimization or active-learning-style loops to propose and evaluate candidate mutations. Similarly, Hie et al. [49] demonstrate efficient evolution of human antibodies using general protein language models, emphasizing iterative selection guided by model scores rather than direct conditional sampling from a guided generative process. These approaches are highly practical for local improvement starting from strong initial sequences, but are conceptually distinct from plug-and-play conditional sampling methods that directly reweigh the generative trajectory.

Supervised fitness optimization. A large body of work in machine-learning-assisted directed evolution trains supervised predictors $p(y \mid x)$ on experimentally measured variants and then uses these predictors to select new variants to test, often in active learning or Bayesian optimization style loops. Examples include ML-assisted directed evolution with combinatorial libraries [50] and subsequent work emphasizing the importance of training set design and "holey" fitness landscapes [51], and has been explored in more data-efficient regime [52]. These methods can be viewed as attempting to invert a learned predictor to propose improved sequences, but unlike generative-model-based approaches they do not provide an explicit conditional sequence distribution and typically rely on local search or proposal mechanisms.

1.2 Detailed derivations and proofs

Below we include the detailed derivations and proofs for the statements and lemmas in the Methods section "Statement of exact MFM sampling using AO-ARMs".

1.2.1 Proofs of exact MFM sampling using AO-ARMs

Proof of Proposition 1. Since CTMCs are, by definition memoryless (depend only on the previous time and state), the path probability density decomposes into a product of densities for each transition conditioned on the last. This can be written as

$$\begin{aligned}&\mathbb{P}_{\text{MFM}}\left(\boldsymbol{\tau}, \boldsymbol{X}\right)\\ =&\mathbb{P}_{\text{MFM}}(\tau_0, x_{\tau_0}) \times \prod_{i=1}^{D} \mathbb{P}_{\text{MFM}}(\tau_i, x_{\tau_i}|\tau_{i-1}, x_{\tau_{i-1}}) \times \mathbb{P}_{\text{MFM}}(\tau_{D+1} > 1|\tau_D, x_{\tau_D}).\end{aligned}$$

The MFM always starts at time $\tau_0 = 0$ with the fully masked sequence $x_0 = \{M\}^D$ so $\mathbb{P}_{\text{MFM}}(\tau_0, x_{\tau_0}) = 1$. There are also exactly D transitions because we assume 0 stochasticity and no corrector sampling. This means $\mathbb{P}_{\text{MFM}}(\tau_{D+1} > 1|\tau_D, x_{\tau_D}) = 1$ as well. The above now simplifies to

$$\begin{aligned}&= \prod_{i=1}^{D} \mathbb{P}_{\text{MFM}}(\tau_i, x_{\tau_i}|\tau_{i-1}, x_{\tau_{i-1}})\\ &= \prod_{i=1}^{D} \mathbb{P}_{\text{MFM}}(\tau_i|\tau_{i-1}, x_{\tau_{i-1}})\mathbb{P}_{\text{MFM}}(x_{\tau_i}|\tau_i, x_{\tau_{i-1}}).\end{aligned}$$

We can apply Lemma 1 to remove the state dependence from the conditional distribution over jump times, $\mathbb{P}_{\text{MFM}}(\tau_i|\tau_{i-1}, x_{\tau_{i-1}})$. Similarly, we can apply Lemma 2 to remove the time dependence from the conditional distribution over jump states, $\mathbb{P}_{\text{MFM}}(x_{\tau_i}|\tau_i, x_{\tau_{i-1}})$. This is the key step of the proof, allowing us to separate the time- and state-dependent distributions into separate terms as follows

$$\begin{aligned}&= \prod_{i=1}^{D} \mathbb{P}_{\text{MFM}}(\tau_i|\tau_{i-1})\mathbb{P}_{\text{MFM}}(x_{\tau_i}|x_{\tau_{i-1}})\\ &= \left(\prod_{i=1}^{D} \mathbb{P}_{\text{MFM}}(\tau_i|\tau_{i-1})\right)\left(\prod_{i=1}^{D} \mathbb{P}_{\text{MFM}}(x_{\tau_i}|x_{\tau_{i-1}})\right).\end{aligned}$$

In Lemmas 1 and 2 we also calculate closed-form expressions for $\mathbb{P}_{\text{MFM}}(\tau_i|\tau_{i-1})$ and $\mathbb{P}_{\text{MFM}}(x_{\tau_i}|x_{\tau_{i-1}})$, which can be substituted in and simplified as follows

$$= \prod_{i=1}^{D} \dot{\kappa}_{\tau_i} \times \prod_{i=1}^{D} p_{1|\tau_i}(X_1 = x_{\tau_i}|X^-_{\tau_i} = x_{\tau_{i-1}}).$$

Now, we will rewrite these products to directly match the proposition statement. First, the interpolation schedule κ_t is the CDF of the jump time under the assumption of minimal transitions. This means the PDF of the jump time for a given position is $\mathbb{P}_{\text{MFM}}(\tau = t) = \frac{d}{dt}\mathbb{P}_{\text{MFM}}(\tau < t) = \dot{\kappa}_t$. Generally, we choose $\kappa_t = t$ so the PDF of the jump times is uniform $\mathbb{P}_{\text{MFM}}(t = \tau) = \dot{\kappa}_t = 1 = U_{[0,1]}(t = \tau)$:

$$= \prod_{i=1}^{D} U[0, 1](\tau_i) \times \prod_{i=1}^{D} p_{1|\tau_i}(X_1 = x_{\tau_i}|X^-_{\tau_i} = x_{\tau_{i-1}}).$$

Finally, we recognize that the AO-ARM constructed for the sampling algorithm uses the MFM denoiser as its conditional distribution. This means the second product above is just the conditional probability of the sample under the AO-ARM and the expression simplifies to

$$\begin{aligned}&= \prod_{i=1}^{D} U[0,1](\tau_i) \times \prod_{i=1}^{D} p(X^{\sigma(i)} = x_{\tau_i} | X^{\sigma(<i)} = x_{\tau_{i-1}}) \\ &= U^D[0,1](\boldsymbol{T} = \sigma^{-1}(\boldsymbol{\tau})) \times \mathbb{P}_{\mathrm{OA}}(X = X_{\tau_D} | \sigma)\end{aligned}$$

as claimed. □

Proof of Corollary 1. This follows directly from Proposition 1. Originally, we found

$$\mathbb{P}_{\mathrm{MFM}}(\tau, \boldsymbol{X}) = \mathbb{P}_{\mathrm{OA}}(X_{\tau_D} | \sigma) U^D[0,1](\boldsymbol{T} = \sigma^{-1}(\boldsymbol{\tau}))$$

To be able to sample both the times and permutations independently, upfront, our sampler must find the permutation π that maps the sampled transition times $\boldsymbol{T}$ to sampled positions. Then applying σ to $\pi(T)$ sorts the times to be $\boldsymbol{\tau}$. In practice, we would just assign the sampled times to positions in the order specified by σ. For the purposes of this proof, however, π lets us rewrite this expression as:

$$= U[S_D](\sigma) \mathbb{P}_{\mathrm{OA}}(X_{\tau_D} | \sigma) U^D[0,1](\boldsymbol{T} = \pi^{-1}(\sigma^{-1}(\boldsymbol{\tau})))$$

Finally, we apply the definition of OA-AR sampling

$$= \mathbb{P}_{\mathrm{OA}}(\boldsymbol{X}) U^D[0,1](\boldsymbol{T})$$

□

1.2.2 Statements and proofs of lemmas

Lemma 1. *The conditional probability of the next jump time for an MFM depends only on the previous jump time and can be expressed in closed form as:*

$$\begin{aligned}\mathbb{P}_{\mathrm{MFM}}\left(\tau_i | \tau_{i-1}, x_{\tau_{i-1}}\right) &= \mathbb{P}_{\mathrm{MFM}}\left(\tau_i | \tau_{i-1}\right) \\ &= (D - (i-1)) \frac{\dot{\kappa}_{\tau_i}}{1 - \kappa_{\tau_{i-1}}} \left(\frac{1 - \kappa_{\tau_i}}{1 - \kappa_{\tau_{i-1}}} \right)^{D-i}. \end{aligned} \tag{S1}$$

This is the probability that any *one* of the remaining $D - (i-1)$ masked positions transitions at time τ_i,[1] given that none of them transitioned between τ_{i-1} and τ_i and all other positions have already been unmasked.

The core insight from the calculation below is that, in a masking process, the time to the next jump only depends on state through the total number of masked tokens: the rate of leaving the mask state is the same across positions, and once unmasked, the state is fixed.[2] As a result, once one specifies that we're sampling the *i-th* jump time, *we know exactly how many masked tokens are left* and there is no further dependence on the CTMC state.

[1] $\frac{\dot{\kappa}_{\tau_i}}{1-\kappa_{\tau_{i-1}}} = \mathbb{P}_{\mathrm{MFM}}(\tau_i | \tau_i > t)$

[2] Recall, there are exactly D jumps because we are using the zero-stochasticity rates from [16, 17].

Proof.

We start by rewriting the PDF of the holding time as the derivative of its CDF,

$$\begin{aligned}&\mathbb{P}_{\text{MFM}}\left(\tau_i|\tau_{i-1}, x_{\tau_{i-1}}\right)\\ =&\frac{\partial}{\partial\tau_i}\mathbb{P}_{\text{MFM}}\left(T_i < \tau_i|\tau_{i-1}, x_{\tau_{i-1}}\right)\end{aligned}$$

which, for CTMCs has the standard form

$$=\frac{\partial}{\partial\tau_i}\left(1-\exp\left(\int_{\tau_{i-1}}^{\tau_i} R_s(x_{\tau_{i-1}}, x_{\tau_{i-1}})ds\right)\right).$$

The rates for two positions transitioning at once are zero [16] so we can factor this expression for the sequence not changing into a product of the probabilities for each position not changing.

$$=\frac{\partial}{\partial\tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i} R_s^{\sigma(i)}(x_{\tau_{i-1}}^{\sigma(i)}, x_{\tau_{i-1}}^{\sigma(i)})ds\right)\right).$$

We can also substitute in the explicit form of the rates (Eq. (33)), taking care that these are the rates for staying at $x_{\tau_{i-1}}$

$$\begin{aligned}=&\frac{\partial}{\partial\tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i}\frac{\dot\kappa_s}{1-\kappa_s}\left(p_{1|s}^{\theta}(X_1^{\sigma(i)}=x_{\tau_{i-1}}^{\sigma(i)}|X_s^-=x_{\tau_{i-1}})-\delta_{x_s^{\sigma(i)}}(x_{\tau_{i-1}}^{\sigma(i)})\right)ds\right)\right)\\ =&\frac{\partial}{\partial\tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i}-\frac{\dot\kappa_s}{1-\kappa_s}\left(1-p_{1|s}^{\theta}(X_1^{\sigma(i)}=x_{\tau_{i-1}}^{\sigma(i)}|X_s^-=x_{\tau_{i-1}})\right)ds\right)\right)\end{aligned}$$

Because these MFMs only make the minimal number of transitions, if the current state at position i is unmasked, we know it will not change. Therefore, $p_{1|t}^{\theta}(\cdot|X_t\neq M)=\delta_{X_t}(\cdot)$ and we can drop the terms corresponding to the unmasked positions to get

$$=\frac{\partial}{\partial\tau_i}\left(1-\prod_{j\in\sigma(\geq i)}\exp\left(\int_{\tau_{i-1}}^{\tau_i}-\frac{\dot\kappa_s}{1-\kappa_s}\left(1-p_{1|s}^{\theta}(X_1^{j}=x_{\tau_{i-1}}^{j}|X_s^-=x_{\tau_{i-1}})\right)ds\right)\right).$$

We can now drop the state dependence entirely by noting that $p_{1|t}^{\theta}(M|\cdot)=0$ because masked tokens are never in the denoised sequence X_1:

$$=\frac{\partial}{\partial\tau_i}\left(1-\prod_{j\in\sigma(\geq i)}\exp\left(\int_{\tau_{i-1}}^{\tau_i}-\frac{\dot\kappa_s}{1-\kappa_s}ds\right)\right).$$

Simplifying, we find

$$
\begin{aligned}
=&\frac{\partial}{\partial \tau_i}\left(1-\prod_{j\in\sigma(\geq i)}\exp\left(\int_{\tau_{i-1}}^{\tau_i}\frac{d}{ds}\ln(1-\kappa_s)ds\right)\right)\\
=&\frac{\partial}{\partial \tau_i}\left(1-\prod_{j\in\sigma(\geq i)}\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right)\\
=&-\frac{\partial}{\partial \tau_i}\left(\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right)^{D-(i-1)}\\
=&(D-(i-1))\frac{\dot\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\left(\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right)^{D-i}\\
=&(D-(i-1))P(\tau_i|\tau_i>\tau_{i-1})\prod_{j\in\sigma(>i)}P(\tau_j>\tau_i|\tau_i>\tau_{i-1})
\end{aligned}
$$

as claimed. □

Note that this result holds for the general DFM case as well, reducing to the same result we get above. Starting with the proof below and the note at the end of Lemma 2, one might extend our technique to more general discrete diffusion processes. However, we leave the practical application of this to future work.

Proof.

As before, we start by rewriting the PDF of the holding time as the derivative of its CDF,

$$
\begin{aligned}
&\mathbb{P}_{\text{DFM}}\left(\tau_i|\tau_{i-1},x_{\tau_{i-1}}\right)\\
=&\frac{\partial}{\partial\tau_i}\mathbb{P}_{\text{DFM}}\left(T_i<\tau_i|\tau_{i-1},x_{\tau_{i-1}}\right)
\end{aligned}
$$

which, for CTMCs has the standard form

$$
=\frac{\partial}{\partial \tau_i}\left(1-\exp\left(\int_{\tau_{i-1}}^{\tau_i}R_s(x_{\tau_{i-1}},x_{\tau_{i-1}})ds\right)\right).
$$

The rates for two positions transitioning at once are zero [16] so we can factor this expression for the sequence not changing into a product of the probabilities for each position not changing.

$$
=\frac{\partial}{\partial \tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i}R_s^{\sigma(i)}(x_{\tau_{i-1}}^{\sigma(i)},x_{\tau_{i-1}}^{\sigma(i)})ds\right)\right).
$$

We can also substitute in the explicit form of the rates (Eq. (33)), taking care that these are the rates for staying at $x_{\tau_{i-1}}$

$$
\begin{aligned}
=&\frac{\partial}{\partial \tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i}\frac{\dot\kappa_s}{1-\kappa_s}\left(p^\theta_{1|s}(X_1^{\sigma(i)}=x_{\tau_{i-1}}^{\sigma(i)}|X_s^-=x_{\tau_{i-1}})-\delta_{x_s^{\sigma(i)}}(x_{\tau_{i-1}}^{\sigma(i)})\right)ds\right)\right)\\
=&\frac{\partial}{\partial \tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i}-\frac{\dot\kappa_s}{1-\kappa_s}\left(1-p^\theta_{1|s}(X_1^{\sigma(i)}=x_{\tau_{i-1}}^{\sigma(i)}|X_s^-=x_{\tau_{i-1}})\right)ds\right)\right)\\
=&\frac{\partial}{\partial \tau_i}\left(1-\prod_{i=1}^{D}\exp\left(\int_{\tau_{i-1}}^{\tau_i}-\frac{d}{ds}\ln(1-\kappa_s)\left(1-p^\theta_{1|s}(X_1^{\sigma(i)}=x_{\tau_{i-1}}^{\sigma(i)}|X_s^-=x_{\tau_{i-1}})\right)ds\right)\right).
\end{aligned}
$$

We now note that, $p^\theta_{1|s}$ has no real dependence on time given the current sequence. It is actually a constant, fixed at the value it takes as $s = \tau_{i-1}$.

$$= \frac{\partial}{\partial \tau_i}\left(1 - \prod_{i=1}^{D} \exp\left(\left(1 - p^\theta_{1|\tau_{i-1}}(X_1^{\sigma(i)} = x^{\sigma(i)}_{\tau_{i-1}} | X^-_{\tau_{i-1}} = x_{\tau_{i-1}})\right) \int_{\tau_{i-1}}^{\tau_i} -\frac{d}{ds}\ln(1-\kappa_s) ds\right)\right).$$

The remaining integral also simplifies nicely:

$$= \frac{\partial}{\partial \tau_i}\left(1 - \prod_{i=1}^{D} \exp\left(\left(1 - p^\theta_{1|\tau_{i-1}}(X_1^{\sigma(i)} = x^{\sigma(i)}_{\tau_{i-1}} | X^-_{\tau_{i-1}} = x_{\tau_{i-1}})\right) \ln\left(\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right) ds\right)\right).$$

$$= \frac{\partial}{\partial \tau_i}\left(1 - \prod_{i=1}^{D} \left(\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right)^{p^\theta_{1|\tau_{i-1}}\left(X_1^{\sigma(i)} \neq x^{\sigma(i)}_{\tau_{i-1}} | X^-_{\tau_{i-1}} = x_{\tau_{i-1}}\right)}\right)$$

$$= \sum_{j=1}^{D} p^\theta_{1|\tau_{i-1}}\left(X_1^{\sigma(j)} \neq x^{\sigma(j)}_{\tau_{i-1}} | X^-_{\tau_{i-1}} = x_{\tau_{i-1}}\right) \frac{\dot\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}} \left(\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right)^{\sum_{k=1}^{D} p^\theta_{1|\tau_{i-1}}\left(X_1^{\sigma(k)} \neq x^{\sigma(k)}_{\tau_{i-1}} | X^-_{\tau_{i-1}} = x_{\tau_{i-1}}\right) - 1}$$

Intuitively, this is a sum of the probabilities for each position j, that we observe all the positions stay constant until j is the next to transition at time τ_i. For the special case where we are using a masking process, we never transition to the masked state so we have $p^\theta_{1|\tau_{i-1}}(X_1^{\sigma(i)} \neq \cdot | X^-_{\tau_{i-1}} = M) = \delta_M(\cdot)$ and $p^\theta_{1|\tau_{i-1}}(X_1^{\sigma(i)} \neq M | X^-_{\tau_{i-1}} = \cdot) = 1$. These observations let us only evaluate the sum over masked positions and set the exponent to the number number of masked positions minus one:

$$= (D - (i-1)) \frac{\dot\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}} \left(\frac{1-\kappa_{\tau_i}}{1-\kappa_{\tau_{i-1}}}\right)^{D-i}$$

$$= (D - (i-1)) P(\tau_i | \tau_i > \tau_{i-1}) \prod_{j \in \sigma(>i)} P(\tau_j > \tau_i | \tau_i > \tau_{i-1})$$

as claimed. □

Lemma 2. *The distribution over state transitions is independent of the jump time, and can be expressed in closed-form as follows:*

$$\mathbb{P}_{\text{MFM}}(x_{\tau_i} | \tau_i, X^-_{\tau_i} = x_{\tau_{i-1}}) = \frac{p^\theta_{1|\tau_i}(X_1^{\sigma(i)} = x^{\sigma(i)}_{\tau_i} | X^-_{\tau_i} = x_{\tau_{i-1}})}{D - (i-1)} \tag{S2}$$

Note that the t in $p^\theta_{1|t}$ is suggestive but is often not an explicit input to the model. See Ou et al. [53], Shi et al. [54], Sahoo et al. [55] for further discussion of the analogous results and implications of the lemma for MDMs. In particular, Sahoo et al. [55] provide some empirical results showing that inputting time to the model is unnecessary for MDMs.

Proof. There is a vanishingly small chance that two positions jump at the same time [16]. Accordingly, we first examine only the conditional distribution for the position that changes. We can write

it as

$$
\begin{aligned}
&\mathbb{P}_{\mathrm{MFM}}\left(X_{\tau_i}^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | \tau_i, X_{\tau_i}^{-} = x_{\tau_{i-1}}\right) \\
=&\mathbb{P}_{\mathrm{MFM}}\left(X_{\tau_i}^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{\sigma(i)} \neq M, X_{\tau_i}^{\sigma(i)^-} = M, X_{\tau_i}^{-} = x_{\tau_{i-1}}\right) \\
=&\frac{\mathbb{P}_{\mathrm{MFM}}\left(X_{\tau_i}^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}\right)}{\mathbb{P}_{\mathrm{MFM}}\left(X_{\tau_i}^{\sigma(i)} \neq M | X_{\tau_i}^{-} = x_{\tau_{i-1}}\right)} \\
=&\frac{R_t^{\sigma(i)}\left(M, x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}\right) dt}{\sum_{\tilde{x} \neq M} R_t^{\sigma(i)}\left(M, \tilde{x} | X_{\tau_i}^{-} = x_{\tau_{i-1}}\right) dt} \\
=&\frac{R_t^{\sigma(i)}\left(M, x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}\right)}{-R_t^{\sigma(i)}\left(M, M | X_{\tau_i}^{-} = x_{\tau_{i-1}}\right)} \\
=&\frac{\frac{\dot{\kappa}_{\tau_i}}{1-\kappa_{\tau_i}}\left(p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}) - \delta_{x_{\tau_i}^{\sigma(i)}, M}\right)}{-\frac{\dot{\kappa}_{\tau_i}}{1-\kappa_{\tau_i}}\left(p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = M | X_{\tau_i}^{-} = x_{\tau_{i-1}}) - \delta_{M,M}\right)}.
\end{aligned}
$$

Notice that here we lose all time dependency,

$$
\begin{aligned}
=&\frac{p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}})}{\delta_{M,M}} \\
=&p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}).
\end{aligned}
$$

However, we still need to account for the fact that, out of all unmasked positions, $\sigma(i)$ was the one that changed.

$$
\begin{aligned}
&\mathbb{P}_{\mathrm{MFM}}(x_{\tau_i} | \tau_i, X_{\tau_i}^{-} = x_{\tau_{i-1}}) \\
&= p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}) \frac{p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} \neq M | X_{\tau_i}^{-} = x_{\tau_{i-1}})}{\sum_{j \in \sigma(\geq i)} p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} \neq M | X_{\tau_i}^{-} = x_{\tau_{i-1}})} \\
&= \frac{p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}})}{D - (i-1)}
\end{aligned}
$$

as claimed. □

Although we present a proof for the MFM case, the same calculation can be used to see that the general result for DFMs[16, 17], i.e.,

$$
\begin{aligned}
&\mathbb{P}_{\mathrm{DFM}}(x_{\tau_i}^{\sigma(i)} | \tau_i, X_{\tau_i}^{-} = x_{\tau_{i-1}}) \\
=&p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}) \frac{1 - p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_{i-1}}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}})}{D - \sum_{j=1}^{D} p_{1|\tau_i}^{\theta}(X_1^{\sigma(j)} = x_{\tau_{i-1}}^{\sigma(j)} | X_{\tau_i}^{-} = x_{\tau_{i-1}})} \\
=&p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} = x_{\tau_i}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}}) \frac{p_{1|\tau_i}^{\theta}(X_1^{\sigma(i)} \neq x_{\tau_{i-1}}^{\sigma(i)} | X_{\tau_i}^{-} = x_{\tau_{i-1}})}{\sum_{j=1}^{D} p_{1|\tau_i}^{\theta}(X_1^{\sigma(j)} \neq x_{\tau_{i-1}}^{\sigma(j)} | X_{\tau_i}^{-} = x_{\tau_{i-1}})}
\end{aligned}
$$

2 Supplementary Tables

Table S1: Multi-property, Pareto-extrapolative experiment: generation performance after post-hoc filtering with predictor.

Method	N Sequences	Success Rate (Top-100)	Diversity
Unguided	1000	16	33
FT	1000	6	3
Guidance	100	75	36

Table S2: Multi-property, Pareto-extrapolative experiment: generation performance under equal wall-clock time without filtering.

			Pb log-FC		Zn log-FC	
Method	**Success Total**	**Success / Sec**	**Mean**	**95th pct.**	**Mean**	**95th pct.**
Unguided	19	0.1	1.9	4.9	4.9	-2.9
FT	3	0.0	0.8	2.4	-0.4	-2.2
Guidance	75	0.4	2.8	4.5	-0.7	-2.8

Table S3: EC classes with enzyme names and associated keyword guidance.

EC Number	Enzyme Name	Keywords
EC:2.3.2.27	RING-type E3 ubiquitin transferase	ring type, e3 ubiquitin, ubiquitin transferase
EC:2.7.11.1	Non-specific serine threonine protein kinase	non specific, serine/threonine kinase
EC:2.7.7.6	DNA-directed RNA polymerase	dna directed, rna polymerase
EC:3.1.26.4	Ribonuclease H	ribonuclease H
EC:3.6.4.12	DNA helicase	dna helicase
EC:3.6.4.13	RNA helicase	rna helicase
EC:4.2.1.33	3-isopropylmalate dehydratase	3-isopropylmalate, isopropylmalate dehydratase
EC:5.2.1.8	Peptidylprolyl isomerase	peptidyl prolyl, isomerase
EC:7.1.1.2	NADH:ubiquinone reductase (H^+-translocating)	NADH ubiquinone, ubiquinone reductase, proton, translocating
EC:7.1.2.2	H^+-transporting two-sector ATPase	proton, transporting, two sector, ATPase

Table S4: **Enzyme-class guidance results for different EC sets at 80% masking level.** Number of samples (N) per set are indicated. "Mask Level %" is the percent of positions masked, 100% is full masked. "CLEAN %" is the percentage of samples that were correctly classified as the intended target class. "pLDDT > 0.8" is the proportion of samples correctly reclassified by CLEAN that have a median pLDDT across residues above 0.8.

Method	EC Set	CLEAN % (↑)	Mean pLDDT (↑)	pLDDT > 0.8 (↑)	Success % (↑)	N
ESM3	Top-10	64%	0.84	86%	55%	100
	Random	61%	0.83	95%	58%	100
	Min	38%	0.82	94%	36%	86
+ProteinGuide	Top-10	76%	0.85	81%	62%	100
	Random	73%	0.85	99%	72%	100
	Min	41%	0.80	80%	33%	86

Table S5: Length-normalized average Hamming distance from wild type for Fine-tuning (FT) and ProteinGuide across 10 enzyme commission (EC) numbers.

EC Number	ProteinGuide	FT
EC:2.3.2.27	0.5503	0.6234
EC:2.7.11.1	0.5660	0.5661
EC:2.7.7.6	0.6817	0.7273
EC:3.1.26.4	0.5550	0.5295
EC:3.6.4.12	0.6185	0.6835
EC:3.6.4.13	0.4954	0.6449
EC:4.2.1.33	0.6364	0.6599
EC:5.2.1.8	0.6361	0.6219
EC:7.1.1.2	0.7096	0.5661
EC:7.1.2.2	0.6361	0.6165

Table S6: Average Hamming distance to wildtype as a function of masking level for ProteinGuide and Unguided sampling.

Masking Level	ProteinGuide	Unguided
0.5	0.72	0.72
0.7	0.58	0.58
0.8	0.48	0.46
0.9	0.29	0.26
1.0	0.07	0.07

Table S7: Primer sequences used in the base editor experiment.

Type	Sequence
Illumina forward primers	GCTCTTCCGATCTNCGACGCCACGTTGTAT
	GCTCTTCCGATCTNNCGACGCCACGTTGTAT
	GCTCTTCCGATCTNNNCGACGCCACGTTGTAT
Illumina reverse primers	GCTCTTCCGATCTNCGCTAGATCCTCCGGA
	GCTCTTCCGATCTNNCGCTAGATCCTCCGGA
	GCTCTTCCGATCTNNNCGCTAGATCCTCCGGA
Nanopore forward primer	AGGGAGTCCACAACGGTTTCCCT
Nanopore reverse primer	GCTAGTTATTGCTCAGCGG

Table S8: **CATH labels used for A- and T-level fold-class conditioning.** Each row includes the CATH code, its description, whether it belongs to the "common" or "random" group, the total number of samples in the dataset, and the fold-oracle test accuracy for that label.

Code	Description	Group	# samples	Test Acc
A-level				
3.40	3-Layer(aba) Sandwich	common	84172	0.8163
2.60	Sandwich	common	34728	0.8724
2.40	Beta Barrel	common	24208	0.7961
3.20	Alpha-Beta Barrel	common	15623	0.7755
3.10	Roll	common	13951	0.7230
2.30	Roll	common	6424	0.7406
1.25	Alpha Horseshoe	common	2967	0.7316
2.80	Trefoil	common	1365	0.8224
2.130	7 Propeller	common	1239	0.9423
2.120	6 Propeller	common	1011	0.7374
2.100	Aligned Prism	random	305	0.8485
3.80	Alpha-Beta Horseshoe	random	756	0.8226
2.102	3-layer Sandwich	random	234	0.7812
2.50	Clam	random	66	0.9091
3.70	Box	random	297	0.9412
2.115	5 Propeller	random	434	0.7778
2.90	Orthogonal Prism	random	141	1.0000
2.160	3 Solenoid	random	964	0.7396
3.65	Alpha-beta prism	random	400	1.0000
2.140	8 Propeller	random	398	0.8889
T-level				
3.40.50	Rossmann fold	common	50985	0.7885
2.60.40	Immunoglobulin-like	common	23273	0.8811
3.20.20	TIM Barrel	common	14701	0.7779
3.30.70	Alpha-Beta Plaits	common	9911	0.7960
2.60.120	Jelly Rolls	common	8713	0.8591
2.40.10	Thrombin, subunit H	common	7012	0.9036
3.40.190	D-Maltodextrin-Binding Protein	common	5938	0.7025
2.40.70	Cathepsin D, subunit A	common	4683	1.0000
3.40.30	Glutaredoxin	common	3702	0.9466
1.10.490	Globin-like	common	2682	0.9792
2.30.110	Pnp Oxidase	random	367	0.9487
1.25.40	Serine Threonine Protein Phosp...	random	2155	0.7453
3.30.429	Macrophage Migration Inhibitor...	random	489	1.0000
3.10.310	Diaminopimelate Epimerase	random	285	0.8000
2.160.10	UDP N-Acetylglucosamine Acyltr...	random	544	0.7547
3.10.129	Thiol Ester Dehydrase	random	973	0.9024
1.20.91	Influenza Virus Matrix Protein	random	76	0.7273
3.40.605	Aldehyde Dehydrogenase	random	1113	1.0000
2.40.155	Green Fluorescent Protein	random	1100	1.0000
3.40.225	L-fuculose-1-phosphate Aldolase	random	90	1.0000

Table S9: **TadA-dSpRYCas9 (ABE0.1-SpRY) protein sequence.** Colors indicate residue annotations: cyan, TadA N-terminal end; red, TadA diversified region; yellow, XTEN linker; gray, dSpRYCas9.

MSEVEFSHEYWMRHALTLAKRAWDEREVPVGAVLVHNNRVIGEGWNRPIGRHDPTAHAEIMALRQGGLVMQNYRLIDATLYVTLEPCVMCAGAMIHSRIGRVVFGARDAK
TGAAGSLMDVLHHPGMNHRVEITEGILADECAALLSDFFRMRRQEIKAQKKAQSSTDSGGSSGGSSGSETPGTSESATPESSGGSSGGSMDKKYSIGLAIGTNSVGWAVI
TDEYKVPSKKFKVLGNTDRHSIKKNLIGALLFDSGETAERTRLKRTARRRYTRRKNRICYLQEIFSNEMAKVDDSFFHRLEESFLVEEDKKHERHPIFGNIVDEVAYHEK
YPTIYHLRKKLVDSTDKADLRLIYLALAHMIKFRGHFLIEGDLNPDNSDVDKLFIQLVQTYNQLFEENPINASGVDAKAILSARLSKSRRLENLIAQLPGEKKNGLFGNL
IALSLGLTPNFKSNFDLAEDAKLQLSKDTYDDDLDNLLAQIGDQYADLFLAAKNLSDAILLSDILRVNTEITKAPLSASMIKRYDEHHQDLTLLKALVRQQLPEKYKEIF
FDQSKNGYAGYIDGGASQEEFYKFIKPILEKMDGTEELLVKLNREDLLRKQRTFDNGSIPHQIHLGELHAILRRQEDFYPFLKDNREKIEKILTFRIPYYVGPLARGNSR
FAWMTRKSEETITPWNFEEVVDKGASAQSFIERMTNFDKNLPNEKVLPKHSLLYEYFTVYNELTKVKYVTEGMRKPAFLSGEQKKAIVDLLFKTNRKVTVKQLKEDYFKK
IECFDSVEISGVEDRFNASLGTYHDLLKIIKDKDFLDNEENEDILEDIVLTLTLFEDREMIEERLKTYAHLFDDKVMKQLKRRRYTGWGRLSRKLINGIRDKQSGKTILD
FLKSDGFANRNFMQLIHDDSLTFKEDIQKAQVSGQGDSLHEHIANLAGSPAIKKGILQTVKVVDELVKVMGRHKPENIVIEMARENQTTQKGQKNSRERMKRIEEGIKEL
GSQILKEHPVENTQLQNEKLYLYYLQNGRDMYVDQELDINRLSDYDVDAIVPQSFLKDDSIDNKVLTRSDKNRGKSDNVPSEEVVKKMKNYWRQLLNAKLITQRKFDNLT
KAERGGLSELDKAGFIKRQLVETRQITKHVAQILDSRMNTKYDENDKLIREVKVITLKSKLVSDFRKDFQFYKVREINNYHHAHDAYLNAVVGTALIKKYPKLESEFVYG
DYKVYDVRKMIAKSEQEIGKATAKYFFYSNIMNFFKTEITLANGEIRKRPLIETNGETGEIVWDKGRDFATVRKVLSMPQVNIVKKTEVQTGGFSKESIRPKRNSDKLIA
RKKDWDPKKYGGFLWPTVAYSVLVVAKVEKGKSKKLKSVKELLGITIMERSSFEKNPIDFLEAKGYKEVKKDLIIKLPKYSLFELENGRKRMLASAKQLQKGNELALPSK
YVNFLYLASHYEKLKGSPEDNEQKQLFVEQHKHYLDEIIEQISEFSKRVILADANLDKVLSAYNKHRDKPIREQAENIIHLFTLTNLGAPRAFKYFDTTIDPKQYRSTKE
VLDATLIHQSITGLYETRIDLSQLGGD

Table S10: **Carbenicillin resistance gene with Q86Stop mutation.** The red T indicates the target A on the complementary strand.

ATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTCACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCA
GTTGGGTGCACGAGTGGGTTACATCGAACTGGATCTCAACAGCGGTAAGATCCTTGAGAGTTTTCGCCCCGAAGAACGTTTTCCAATGATGAGCACTTTTAAAGTTCTGC
TATGTGGCGCGGTATTATCCCGTATTGACGCCGGGTAAGAGCAACTCGGTCGCCGCATACACTATTCTCAGAATGACTTGGTTGAGTACTCACCAGTCACAGAAAAGCAT
CTTACGGATGGCATGACAGTAAGAGAATTATGCAGTGCTGCCATAACCATGAGTGATAACACTGCGGCCAACTTACTTCTGACAACGATCGGAGGACCGAAGGAGCTAAC
CGCTTTTTTGCACAACATGGGGGATCATGTAACTCGCCTTGATCGTTGGGAACCGGAGCTGAATGAAGCCATACCAAACGACGAGCGTGACACCACGATGCCTGTAGCAA
TGGCAACAACGTTGCGCAAACTATTAACTGGCGAACTACTTACTCTAGCTTCCCGGCAACAATTAATAGACTGGATGGAGGCGGATAAAGTTGCAGGACCACTTCTGCGC
TCGGCCCTTCCGGCTGGCTGGTTTATTGCTGATAAATCTGGAGCCGGTGAGCGTGGGTCTCGCGGTATCATTGCAGCACTGGGGCCAGATGGTAAGCCCTCCCGTATCGT
AGTTATCTACACGACGGGGAGTCAGGCAACTATGGATGAACGAAATAGACAGATCGCTGAGATAGGTGCCTCACTGATTAAGCATTGGTAA

3 Supplementary Figures

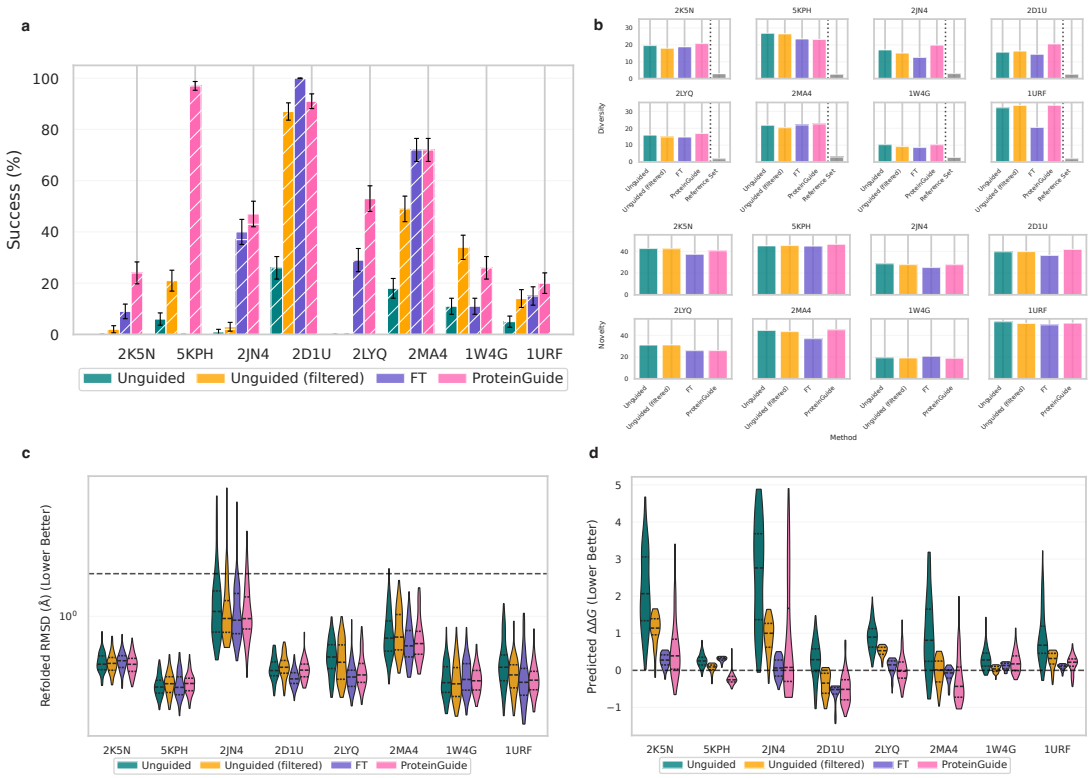

Figure S1: **Guiding ProteinMPNN for improved stability, using ProteinMPNN's default any-order autoregressive sampling. a**) Four methods for sampling (in legends) are assessed across eight proteins (horizontal axis) by their success rate on $k = 100$ generated samples. The height of each bar represents the percentage of sequences predicted to be at least as stable as wild type ($\Delta\Delta G \leq 0$), equivalently the empirical mean of a binary success indicator, while the hatched sub-portion indicates the fraction of those stable sequences that are also predicted to correctly fold into the desired structure (RMSD ≤ 2Å). In most cases the hatched sub-portion covers the full height of the bar. Error bars represent $\pm$ one standard error of the empirical mean, calculated as $\sqrt{\hat{p}(1-\hat{p})/k}$, where $\hat{p}$ is the empirical proportion of success and $k = 100$ is the sample size. **b**) Diversity (top) and novelty (bottom) of the generated sequences. For each method, diversity and novelty are evaluated on $k = 100$ sequences. Diversity is computed as the averaged pairwise Hamming distance among the generated sequences. For each protein, the "Reference Set" diversity is computed on 500 randomly sampled sequences from the labeled dataset for each protein. For each sequence, novelty is computed as its minimum Hamming distance with sequences in the full labeled dataset. For each method, the mean novelty among the generated sequences is shown. **c**) Distributions of refolded RMSD for the generated sequences (lower is better). The dashed horizontal line indicates the threshold below which a sequence is considered to fold into the desired structure (RMSD ≤ 2Å). **d**) Distributions of oracle predicted stability for the generated sequences (lower is better). The dashed horizontal line indicates the threshold below which a sequence is predicted to be at least as stable as the wild type ($\Delta\Delta G \leq 0$).

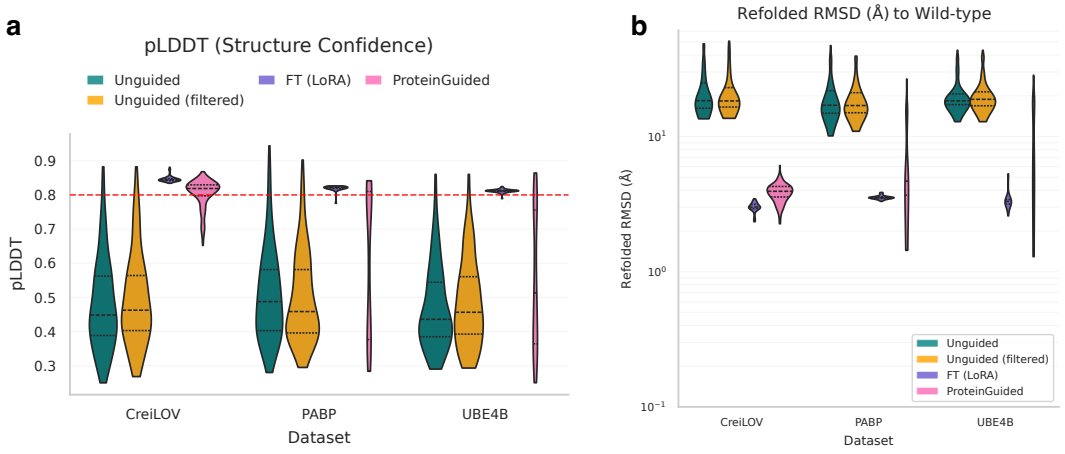

Figure S2: **Structure prediction-based metrics for generated sequences from guiding ESM3 with assay-informed predictive models. a**) Distribution of the pLDDT of the generated sequences from each method (higher is better). pLDDT is based on ESMFold prediction. Red dotted line indicates the threshold above which a generated sequence is considered a success (pLDDT ≥ 0.8). **b**) Distribution of the refolded RMSD (Å) of the generated sequences from each method (lower is better). RMSD is computed between the predicted structure of the wild-type sequence and that of the generated sequence.

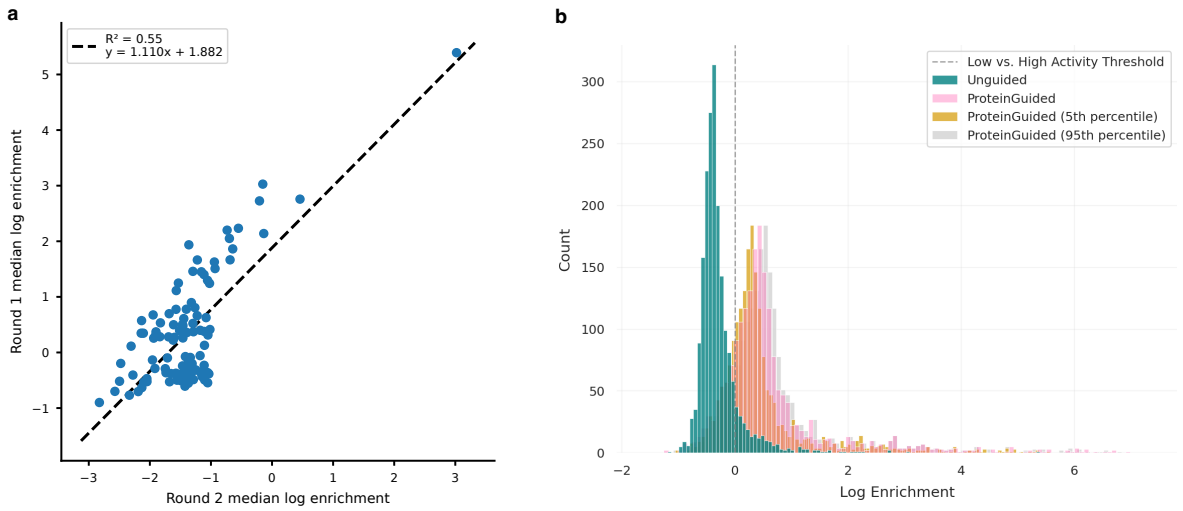

Figure S3: **Normalizing log enrichment scores between the first and second round libraries in the *in vivo* base editor experiment. a**) Log enrichment scores of the $n = 109$ control sequences that were tested in both round 1 and round 2. The x-axis shows the median log enrichment scores for the replicates from round 2. The y-axis shows the median log enrichment scores for the replicates from round 1. **b**) Histogram of log enrichment values comparing unconditional sequences (teal) and guided sequences (pink) after calibration between rounds. Bootstrap analysis (1,000 iterations) of the control sequences yielded a mean median log enrichment of 0.4 for the guided library with a 90% confidence interval of [0.29, 0.51]. The 5th and 95th percentile mappings (gold and gray histograms, respectively) represent conservative and optimistic transformations based on bootstrap sampling, visualizing the uncertainty range in our calibration procedure. The gray dashed line at a log enrichment value of 0 represents the low vs. high activity classification threshold. ProteinGuide was used to generate sequences with log enrichment values greater than this threshold.

References

[1] Patrick Emami, Aidan Perreault, Jeffrey Law, David Biagioni, and Peter St John. Plug & play directed evolution of proteins with gradient-based discrete mcmc. *Machine Learning: Science and Technology*, 4(2):025014, 2023.

[2] Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, et al. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*, 2024.

[3] Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Jp988ELppQ.

[4] Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

[5] Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops I Took a Gradient: Scalable Sampling for Discrete Distributions. In *International Conference on Machine Learning*, pages 3831–3841, 2021.

[6] Cheuk Kit Lee, Paul Jeha, Jes Frellsen, Pietro Lio, Michael Samuel Albergo, and Francisco Vargas. Debiasing guidance for discrete diffusion with sequential monte carlo. *arXiv preprint arXiv:2502.06079*, 2025.

[7] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC. In *International Conference on Machine Learning*, pages 8489–8510, 2023.

[8] John B Ingraham, Max Baranov, Zak Costello, Karl W Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M Lord, Christopher Ng-Thow-Hing, Erik R Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, 2023.

[9] Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024.

[10] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete Denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

[11] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[12] Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion Language Models Are Versatile Protein Learners. In *International Conference on Machine Learning*, pages 52309–52333. PMLR, 2024.

[13] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking Guidance for Discrete State-Space Diffusion and Flow Models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[14] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

[15] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Language Modeling by Estimating the Ratios of the Data Distribution. *arXiv:2310.16834*, 2023.

[16] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi S. Jaakkola. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. In *International Conference on Machine Learning*, 2024.

[17] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[18] Kevin Yang and Dan Klein. FUDGE: Controlled Text Generation With Future Discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, 2021.

[19] Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dallatorre, Bernardo P de Almeida, Alexander M Rush, Thomas PIERROT, and Volodymyr Kuleshov. Simple Guidance Mechanisms for Discrete Diffusion Models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet Diffusion Score Model for Biological Sequence Generation. In *International Conference on Machine Learning*, pages 1276–1301, 2023. URL https://proceedings.mlr.press/v202/avdeyev23a/avdeyev23a.pdf.

[21] Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

[22] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-LM Improves Controllable Text Generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

[23] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv:2211.15089*, 2022. URL https://arxiv.org/abs/2211.15089.

[24] Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*, 2025.

[25] Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Hanchen, and Tommaso Biancalani. Fine-Tuning Discrete Diffusion Models via Reward Optimization with Applications to DNA and Protein Design. In *The Thirteenth International Conference on Learning Representations*, 2025.

[26] Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Cheng-Hao Liu, Sarthak Mittal, Nouha Dziri, Michael M. Bronstein, Pranam Chatterjee, Alexander Tong, and Joey Bose. Steering masked discrete diffusion models via discrete denoising posterior prediction. In *The Thirteenth International Conference on Learning Representations*, 2025.

[27] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[28] Talal Widatalla, Rafael Rafailov, and Brian Hie. Aligning protein generative models with experimental fitness via Direct Preference Optimization. *bioRxiv*, pages 2024–05, 2024.

[29] Shriram Chennakesavalu, Frank Hu, Sebastian Ibarraran, and Grant M Rotskoff. Aligning chemical and protein language models with continuous feedback using energy rank alignment. In *ICLR 2025 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2025.

[30] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024.

[31] Umberto Borso, Davide Paglieri, Jude Wells, and Tim Rocktäschel. Preference-Based Alignment of Discrete Diffusion Models. In *ICLR 2025 Workshop on Bidirectional Human-AI Alignment*, 2025.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024.

[34] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[35] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022.

[36] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *Science*, page eads0018, 2025.

[37] Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Neil Tenenholtz, Bob Strome, Alan Moses, Alex Xijie Lu, Nicolo Fusi, Ava Pardis Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *BioRxiv*, pages 2023–09, 2023.

[38] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41(8):1099–1106, 2023.

[39] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. *ICML*, 2022.

[40] Filippo Stocco, Michele Garibbo, and Noelia Ferruz. Guiding generative models for protein design: Prompting, steering and aligning. *arXiv preprint arXiv:2511.21476*, 2025.

[41] Jason Yang, Wenda Chu, Daniel Khalil, Raul Astudillo, Bruce J. Wittmann, Frances H. Arnold, and Yisong Yue. Steering generative models with experimental data for protein fitness optimization, 2025. URL https://arxiv.org/abs/2505.15093.

[42] Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein Design with Guided Discrete Diffusion. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://openreview.net/forum?id=MfiK69Ga6p.

[43] Sidney Lyayuga Lisanza, Jacob Merle Gershon, Samuel WK Tipps, Jeremiah Nelson Sims, Lucas Arnoldt, Samuel J Hendel, Miriam K Simma, Ge Liu, Muna Yase, Hongwei Wu, et al. Multistate and functional protein design using rosettafold sequence space diffusion. *Nature biotechnology*, pages 1–11, 2024.

[44] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

[45] Long-Kai Huang, Rongyi Zhu, Bing He, and Jianhua Yao. Steering protein language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=pu2aw2zwMx.

[46] Filippo Stocco, Maria Artigues-Lleixa, Andrea Hunklinger, Talal Widatalla, Marc Guell, and Noelia Ferruz. Guiding Generative Protein Language Models with Reinforcement Learning. *arXiv preprint arXiv:2412.12979*, 2024.

[47] Nathaniel Blalock, Srinath Seshadri, Agrim Babbar, Sarah A Fahlberg, Ameya Kulkarni, and Philip A Romero. Functional alignment of protein language models via reinforcement learning. *bioRxiv*, pages 2025–05, 2025.

[48] Kaiyi Jiang, Zhaoqing Yan, Matteo Di Bernardo, Samantha R Sgrizzi, Lukas Villiger, Alisan Kayabolen, BJ Kim, Josephine K Carscadden, Masahiro Hiraizumi, Hiroshi Nishimasu, et al. Rapid in silico directed evolution by a protein language model with EVOLVEpro. *Science*, page eadr6006, 2024.

[49] Brian L Hie, Varun R Shanker, Duo Xu, Theodora UJ Bruun, Payton A Weidenbacher, Shaogeng Tang, Wesley Wu, John E Pak, and Peter S Kim. Efficient evolution of human antibodies from general protein language models. *Nature biotechnology*, 42(2):275–283, 2024.

[50] Zachary Wu, SB Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019.

[51] Bruce J Wittmann, Yisong Yue, and Frances H Arnold. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell systems*, 12(11):1026–1045, 2021.

[52] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-n protein engineering with data-efficient deep learning. *Nature methods*, 18(4):389–396, 2021.

[53] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your Absorbing Discrete Diffusion Secretly Models the Conditional Distributions of Clean Data. In *The Thirteenth International Conference on Learning Representations*, 2025.

[54] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and Generalized Masked Diffusion for Discrete Data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[55] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.