A Diffusion Model to Shrink Proteins While Maintaining their Function

Ethan Baron[*]
New York University

Alan N. Amin[*]
New York University

Ruben Weitzman
Oxford University, Harvard University

Debora S. Marks
Harvard University

Andrew Gordon Wilson
New York University

Abstract

Many proteins useful in modern medicine or bioengineering are challenging to make in the lab, fuse with other proteins in cells, or deliver to tissues in the body because their sequences are too long. Shortening these sequences typically involves costly, time-consuming experimental campaigns. Ideally, we could instead use modern models of massive databases of sequences from nature to learn how to propose shrunken proteins that resemble sequences found in nature. Unfortunately, these models struggle to efficiently search the combinatorial space of all deletions, and are not trained with inductive biases to learn how to delete. To address this gap, we propose SCISOR, a novel discrete diffusion model that deletes letters from sequences to generate protein samples that resemble those found in nature. To do so, SCISOR trains a "de-noiser" to reverse a "forward noising process" that adds random insertions to natural sequences. As a generative model, SCISOR fits evolutionary sequence data competitively with previous large models. In evaluation, SCISOR achieves state-of-the-art predictions of the functional effects of deletions on ProteinGym. Finally, we use the SCISOR de-noiser to shrink long protein sequences, and show that its suggested deletions result in significantly more realistic proteins and more often preserve functional motifs than previous models of evolutionary sequences.

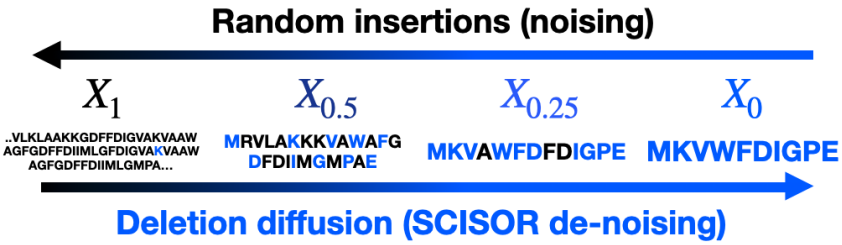
1 Introduction

As protein design becomes easier, more protein constructs are built for bioengineering, more protein medicines are being packaged for delivery to particular tissues, and, of course, more protein is being synthesized in the lab. Unfortunately, many important proteins are challenging to make, engineer, and deliver, due to their long sequences. Methods to build shorter versions of these proteins are expensive and often only narrowly applicable. Typically, experimentalists look for shorter homologues, which may not exist, and put them through costly optimization campaigns [9]. Or, for proteins which function by well-characterized, simple biophysical interactions, experimentalists shrink sequences by running extensive physical simulations [29].

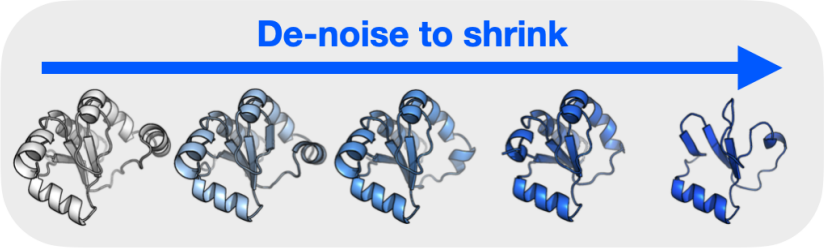
Ideally we could instead learn how to shrink proteins using models trained on databases of protein sequences in nature – these models learn the constraints evolution has put on sequences across life and could shrink proteins to avoid breaking their function. Unfortunately, these large models [17, 15] struggle to effectively search through the massive space of all possible shrunken versions of a protein. They may also lack the inductive bias to predict the effect of deletions, having not been explicitly trained to do so. In principle, the first issue could be solved by diffusion models of protein sequences, like EvoDiff, which are effectively trained to plan series of many mutations and end with sequences that resemble those found in nature [1, 13]. However, current diffusion frameworks can only train models that perform substitution mutations – they cannot suggest deletions.

[*]Equal contribution.

First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025).



(a)



(b)

Figure 1: **SCISOR is a diffusion model trained to make deletions that arrive at a natural protein sequence. We can use it to shrink proteins while maintaining their function.** (a) We add random insertions to protein sequences from nature and train SCISOR to reverse these insertions. (b) Applying SCISOR diffusion to natural proteins, we get smaller proteins that are predicted to preserve parts of the tertiary structures of the original sequence. We show SCISOR samples of Q8NFU3 at 0, 5, 10, 20, and 50% deletion with structures predicted by OmegaFold [27].

We propose a new diffusion model of evolutionary sequences that learns to generate by shorten sequences — Sequence Contraction with InSertion-Only noising pRocess (SCISOR). SCISOR "adds noise" to natural sequences by inserting random letters until they effectively become long random sequences; then it train a "de-noiser" to reverse this process by planning deletions that result in sequences that resemble those found in nature (Fig. 1a). Our contributions are:

- We introduce a **new discrete diffusion framework** that trains a de-noiser to generate sequences by learning to delete – SCISOR.
- We show that among large-scale diffusion models, SCISOR achieves **competitive model fit for protein sequences**.
- We show that the inductive biases of SCISOR allows it to make **state-of-the-art predictions of the effects of deletions on protein functions in the lab in ProteinGym**.
- Finally, we show that SCISOR **shortens proteins while better maintaining their structure and functional motifs** than methods using previous models of protein sequences.

2 Background

Say we have a protein sequence X made up of L letters $X^{(1)}X^{(2)}\cdots X^{(L)}$ belonging to the alphabet of 20 amino acids $\mathcal{B}$. Our goal is to remove M letters from X to make a $\tilde{X} = X^{(j_1)}X^{(j_2)}\cdots X^{(j_{L-M})}$ with $j_1 < j_2 < \ldots, j_{L-M}$, that is still functional. Most random sets of deletions degrade the function of the protein, so we need to predict which deletions are unlikely to break the protein. Unfortunately there is very little data of sequence, shrunk-sequence pairs $(X, \tilde{X})$ to learn from; we must instead learn to predict functional shrunk proteins using other available data.

Models of evolutionary sequences One way we can learn how to shrink proteins is by learning from modern huge datasets of natural proteins. Indeed we can attempt to learn what a natural protein looks like in these databases; then we can pick a shrunken protein $\tilde{X}$ so that it looks natural and is therefore likely to be functional[1]. In practice, we can train huge generative models to generate

[1]Note this does not guarantee our goal that $\tilde{X}$ have the same function as X. But if two functional proteins have similar sequences then they often have related function [14] (see further discussion in Sec. 6).

natural proteins and use their likelihoods as a measure of naturalness [21, 22, 17, 15, 12]. Indeed, these likelihoods have been shown to be accurate predictors of whether single-letter-deletions will harm the function of a protein [17].

Unfortunately, the models that are typically used to fit this data, such as BERT-style [22, 12] and autoregressive models [17, 15], struggle to search over the combinatoral space of all $\binom{L}{M}$ possible large deletions to find an ideal $\tilde{X}$. Ideally, we would have a model that can plan a number of deletions that arrive at a functional protein sequence. We also speculate that a model that learns directly how to delete would make more accurate predictions and designs.

Discrete diffusion To effectively search through a large mutational space, we could model the data with discrete diffusion models. These models generate samples by starting with a random sequence and applying mutations to arrive at a realistic sequence. In particular, a sequence is sampled from a simple distribution $X_1 \sim q(X_1)$ and then it is transformed from time $t = 1$ to $t = 0$ using a de-noiser $q_\theta((X_t)_{t=0}^1 \mid X_1)$ so that X_0 looks like a sequence from the data generating distribution [5].

Diffusion models can therefore be used to search for sets of many mutations to a sequence, X, that result in a realistic looking sequence. To do so, one sets $X_s = X$ for some s and then "de-noises" using the diffusion model by sampling a path $q((X_t)_{t=0}^s | X_s)$, giving a "realistic" X_0 near X_s. Indeed, this procedure has been used to suggest mutations to optimize sequences [13, 8].

To train a de-noiser q_θ, we first define a "forward" process $p((X_t)_{t=0}^1)$ which takes samples from our target distribution $X_0 \sim p(X_0)$ and applies random noise to them from time $t = 0$ to $t = 1$, arriving at a distribution that is easy to approximate $p(X_1)$. Then we train the de-noiser to generate paths that match the paths of the forward process by optimizing an evidence lower bound (ELBO) as

$$\log q_\theta(X_0) \geq \mathbb{E}_{p((X_t)_{t=0}^1 | X_0)} \log \frac{q_\theta((X_t)_{t=0}^1)}{p((X_t)_{t=0}^1 | X_0)}. \tag{1}$$

Typically, however, the forward noising process is chosen to be random substitutions. Accordingly, the de-noiser q_θ only applies substitutions rather than deletions. To search over the space of deletions, we therefore need a new diffusion framework.

Related work We review how discrete diffusion models have handled objects of varying dimensionality in language and chemistry, as well as previous attempts to shrink proteins using large protein models in App. B.

3 A diffusion model that learns to delete: SCISOR

To search the space of deletions and train a model with the right inductive biases, in Sec. 3.1 we build a process which noises sequences by adding random insertions. Then in Sec. 3.2 we show how to train a de-noiser q_θ that reverses this process (Fig. 1a). In App. C.5, we discuss the practical choices we made to efficiently train SCISOR and in App. C.2 we cover details about sampling and shrinking from SCISOR. In the following Sec. C.2 we describe how to use the de-noiser to generate sequences, shrink proteins, and plan deletions in practice.

3.1 Forward noising with the pure birth process

We propose an insertion-only forward noising process for discrete diffusion known as the "pure birth" process [11] with rate function $\beta(t)$ and insertion distribution π. Let X_0 be a sequence $X_0^{(1)}, \ldots, X_0^{(L)}$. There are $L + 1$ possible locations we can insert letters. In the pure birth process, at instant t, each of these locations gains an insertion with rate $\beta(t)$. The letter that is inserted is drawn from some distribution $Y \sim \mathrm{Cat}(\pi)$. After Y is inserted at some position, the process continues and there are now $L + 2$ positions in which there could be insertions with rate $\beta(t)$ (Fig. 1a). To train a diffusion model to reverse this process, we need to (1) easily sample $p(X_t|X_0)$ and (2) easily approximate $p(X_1|X_0)$.

Sampling X_t Rather than simulate the pure birth process up until time t, we show in App. G that X_t can be sampled directly from X_0 as in Alg 1. Note that $0 < \alpha(t) \leq 1$ controls how many insertions are added: by the property of negative binomial distributions, the expected length of X_t is $\mathbb{E}(M_t + L) = \frac{L+1}{\alpha(t)} - 1$ which grows as $\alpha(t)$ goes to 0.

Algorithm 1 Sample X_t

Require: Initial sequence $X = X^{(1)} \cdots X^{(L)}$, time t

Compute the probability of no insertions at a site $\alpha(t) \leftarrow \exp\left(-\int_0^t \beta(s)\, ds\right)$
Sample total number of insertions up to time t, $M_t \sim \text{NegativeBinomial}(L+1, \alpha(t))$
Sample the number of insertions in each position by uniformly distributing M_t into $L+1$ bins: $(\ell_0, \ldots, \ell_L) \sim \text{UniformMultinomial}(M_t)$
for $j = 0$ to L **do**
    Sample insertion Y_j of length ℓ_j, with each character independently from $\text{Cat}(\pi)$
end for
Add insertions into X to construct $X_t \leftarrow Y_0 X^{(0)} Y_1 X^{(1)} \cdots X^{(L)} Y_L$
return X_t

Approximating $p(X_1|X_0)$ As t grows, X_t becomes longer. To build a diffusion model however, the distribution $p(X_t|X_0)$ typically must converge to a distribution so that it can be approximated by a distribution that can easily be sampled from, $q(X_1)$. Our critical insight is that $p(X_t|X_0)$, while not converging, can still be very well approximated by long random sequences as t gets large.

Proposition 3.1. *(Proof in App. G) Say X_0 is a sequence with length L. Call $q(\cdot \mid L)$ a distribution over sequences of length L which simply samples each letter independently from $\text{Cat}(\pi)$. Then, as the number of insertions increases, $M_1 \to \infty$, X_1 becomes easier to approximate with q:*

$$\text{KL}(p(X_1 \mid X_0, M_1) || q(X_1 \mid L + M_1)) \to 0. \tag{2}$$

3.2 Learning to reverse this insertion-only noising process

Given a forward process of insertions, we now wish to learn a de-noiser q_θ that generates sequences that resemble those found in nature by deleting letters from long random sequences. We now (1) describe our reverse process $q_\theta((X_t)_{t=1}^0)$, (2) write the ELBO in Eqn. 1 for our model, and (3) describe how the denoiser q_θ is being trained toward a target that deletes letters that are unlikely to align with the starting sequence X_0.

The reverse process For a forward path $(X_t)_{t=0}^1$ from a sequence X_0 of length L, define $t_1, \ldots, t_{M_1}$ to be the times of each insertion. We can then sample forward paths by first deciding how many insertions will occur until time 1 and when these insertions will occur, and then choosing what these insertions are

$$p((X_t)_{t=0}^1|X_0) = p(M_1|L)p(t_1, \ldots, t_{M_1} \mid M_1, L) \prod_{M=1}^{M_1} p(X_{t_M}|X_{t_{M-1}}).$$

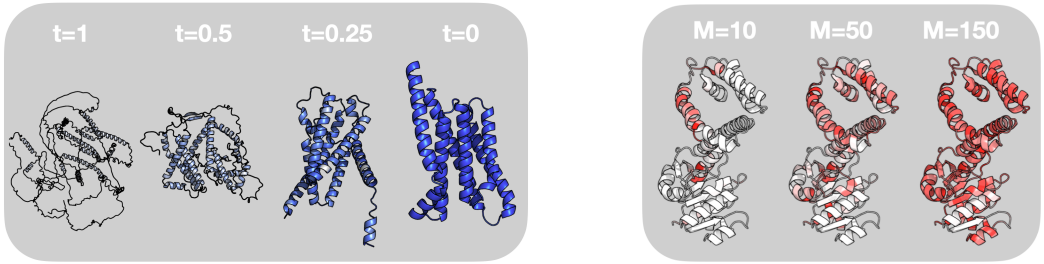
We follow the discrete diffusion framework in Amin et al. [3] in defining the reverse process to match the "noise schedule" of the forward process. To generate a sequence of length L, we first decide the number of insertions and their times from the same distribution as p, and then denoise each insertion (Fig. 3a)[2],

$$q_\theta((X_t)_{t=1}^0|L) = p(M_1|L)p(t_1, \ldots, t_{M_1} \mid M_1, L)q(X_1|L + M_1) \prod_{M=1}^{M_1} q_\theta(X_{t_{M-1}}|X_{t_M}, M).$$

Now we must only train our de-noiser $q_\theta(X_{t_{M-1}}|X_{t_M}, M)$ to take in a sequence X_{t_M} and the number of insertions that sequence has M, and predict the sequence before the last insertion $X_{t_{M-1}}$. That is, $q_\theta(\cdot \mid X_{t_M}, M)$ can be thought of as a distribution over the letters of X_t. Note the predictions of the de-noiser depend on M; different values of M allow the model to change which deletions it will allow at each step (Fig. 3b).

The loss To train the de-noiser, we modify the calculation of the ELBO Eqn. 1 in Amin et al. [3]. We will then use this ELBO as our objective for training the de-noiser.

[2]Note our process is conditioned on generating a sequence of a particular length L.



(a) (b)

Figure 3: **The SCISOR de-noiser q_θ plans deletions to arrive at sequences that resemble those in nature, and therefore avoids deleting important structural motifs in natural sequences.** (a) SCISOR unconditionally samples proteins by starting with a large random sequence X_1 and iteratively deleting according to $q_\theta(\text{prev}(X)|X, M)$ to arrive at a protein that resembles those in nature. We predict the structure of each sequence with OmegaFold [27] . (b) We ask SCISOR to plan the first of M mutations for R4SNK4 and color residue i on a structure from Aleku et al. [2] by the deletion probability $q_\theta(X^{(-i)}|X, M)$ (red is higher probability). As M increases, SCISOR allows insertions in more regions while minimizing deletions in the catalytic structural motif near the bottom (white).

Proposition 3.2. *(Proof in App. G) Define M_t as the number of mutations up to time t, and $\text{prev}(X_t)$ is the last sequence that gained an insertion to become X_t. Then the negative log likelihood of a sequence of length L, $-\log q_\theta(X_0|L)$, is smaller than*

$$\mathbb{E}_{M_1}\text{KL}(p(X_1 \mid X_0, M_1)||q(X_1|L+M_1)) + \mathbb{E}_{t,X_t,M_t}\frac{M_t\beta(t)}{1-\alpha(t)}\text{KL}(p(\text{prev}(X_t) \mid X_0, X_t, M_t)||q_\theta(\text{prev}(X_t) \mid X_t, M_t)) \tag{3}$$

The first term is the quantity in Eqn. 2 – how well we can approximate $p(X_1)$; it is small as long as M_1 is typically large, i.e. $\alpha(1)$ is small, and can be calculated as in App. C. The second term is the quantity we use to train the de-noiser. q_θ takes in X_t and the number of insertions in X_t and must predict which letter of X_t was last inserted – $\text{prev}(X_t)$. To train the model, we must calculate $p(\text{prev}(X_t)|X_0, X_t, M_t)$.

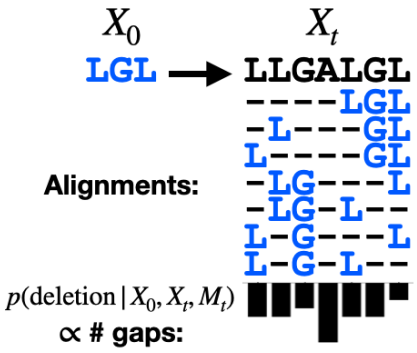


Figure 2: To calculate our target distribution of what letter to delete, $p(\text{prev}(X_t) \mid X_0, X_t, M_t)$, we align our starting sequence X_0 to our "noised" sequence X_t. The reverse process assigns a higher probability to deleting letters that are gaps in more of the alignments.

Target distribution Eqn. 3 trains q_θ to match $p(\text{prev}(X_t)|X_0, X_t, M_t)$, the true distribution over which letter of X_t was last inserted in the forward process.

Conditioned on X_0, X_t, M_t, we could find $\text{prev}(X_t)$ by simulating a path path from X_0 to X_t and seeing what insertion occurred last. However there are many paths that could lead from X_0 to X_t; to calculate $p(\text{prev}(X_t) \mid X_0, X_t, M_t)$, we must marginalize over these paths.

The next proposition shows that we can integrate over all of these paths by first enumerating every way to align X_0 to X_t and noting that letters that align with X_0 less often are more likely to have been $\text{prev}(X_t)$ (Fig. 2).

Proposition 3.3. *(Proof in App. G) Call $\text{ali}(X, Y)$ the number of ways to align a sequence X to a sequence Y. Call b the letter that was deleted from X_t to $\text{prev}(X_t)$.*

$$p(\text{prev}(X_t)|X_0, X_t, M_t) = \frac{\text{ali}(X_0, \text{prev}(X_t))}{M_t \cdot \text{ali}(X_0, X_t)}.$$

Naively computing this quantity would require running an expensive alignment for every deletion. In practice, we use a dynamic programming algorithm that computes all $\text{ali}(X_0, \text{prev}(X_t))$ in parallel (App. H).

4 Fitting the distribution of natural sequences as a diffusion model

We now compare how well SCISOR fits the distribution of natural sequences compared to established sequence modeling methods; we see SCISOR fits sequence data well, competitively with state-of-

Algorithm 2 Conditional shrinking of a sequence

Require: Number of deletions M, initial sequence X, temperature T.

Initialize $\tilde{X} \leftarrow X$
while $|\tilde{X}| > L - M$ **do**
  Remove one letter from $\tilde{X}$ according to $q_\theta(\text{prev}(\tilde{X}) \mid \tilde{X}, M)^{1/T}$
  $M \leftarrow M - 1$
end while
return $\tilde{X}$

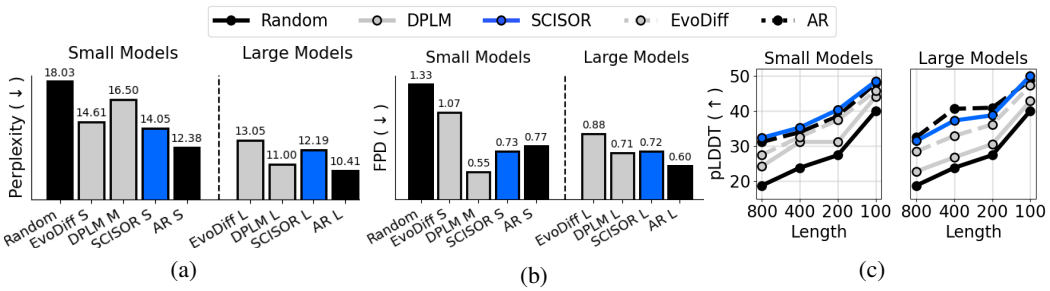


Figure 4: **SCISOR fits the distribution of sequences in nature competitively with established sequence modeling approaches.** (a) SCISOR is competitive with other diffusion models (grey) in perplexity. "S, M, L" refer to model size. (b, c) Samples from SCISOR ($K = 5$) are predicted to be competitive quality to those from diffusion models and competitive with AR models as measured by (b) matching the distribution of natural sequences as measured by the Fréchet protein distance (FPD) and (c) foldability (higher pLDDT from OmegaFold [27]). We took EvoDiff and AR perplexities from Alamdari et al. [1].

the-art diffusion and autoregressive models. All details are in App. D. In Fig. 4, we compare the quality of SCISOR's fit to the data against state-of-the-art protein diffusion models: EvoDiff [1] and DPLM [26]. It is well known that diffusion models regularly under-perform autoregressive models on fitting the data; we therefore include two autoregressive models from Alamdari et al. [1] as references. All models are trained on the same release of UniRef50 [23] – small models have 35-38M parameters, DPLM M has 150M parameters, and large models have 640-650M parameters. We evaluate each model's perplexity on a test set, and the quality of their samples, as measured by how well they match the distribution of natural sequences (FPD), and "foldability" (pLDDT).

We see that, despite its difference from established modeling methods, SCISOR is competitive with other diffusion models in perplexities. As well, SCISOR often generates higher quality samples than previous diffusion models, even competitive with the AR reference. As mentioned in Sec. C.2 this is likely because SCISOR is a continuous-time model while the other diffusion models are discrete-time.

5 Shrinking proteins while preserving their function

We now evaluate the ability of the SCISOR de-noiser q_θ to plan deletions that preserve the function of a protein. Since the diffusion models in Sec. 5 cannot suggest deletions, we compare to a different set of baselines. We compare to shrinking with state-of-the-art autoregressive models ProGen2 [15] and Tranception [17] and a stochastic autoencoder meant for shrinking proteins, Raygun [7] when applicable. Since models trained on UniRef90 tend to better predict the effects of mutations [22], all models in this section are trained on UniRef90. All details are in App. D.

5.1 Deletion effect prediction

First we evaluate the ability of models of evolutionary sequences to predict the effect of mutations on the function of proteins as measured in the lab. We collected more than 7000 measurements of deletions across 62 assays collected in ProteinGym [18] and measured the Spearman correlations of the measurements of each assay against the predicted effects from each model. In Fig. 5 we show the best average correlation across the assays across each model family (full table in App. E). SCISOR

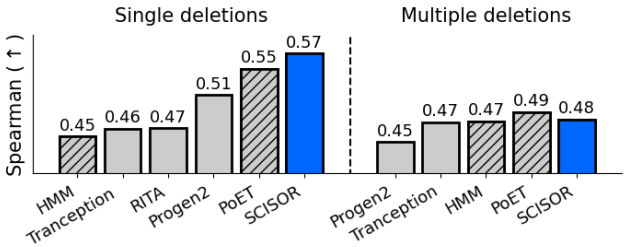
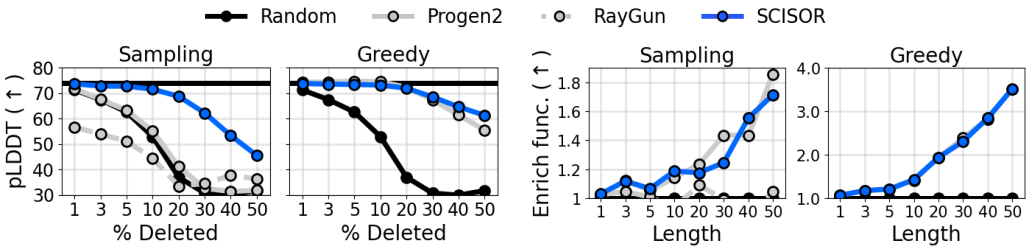


Figure 5: **SCISOR accurately predicts the effect of deletions on protein function measured in the lab.** We make predictions for the effects of deletions and calculate a Spearman correlation between predictions and measurements for each assay in ProteinGym. We report the average Spearman correlation coefficient across all assays for each model, presenting the results from the highest-performing variant of each model architecture. The table includes all models from the ProteinGym leaderboard that achieve a correlation coefficient exceeding 0.4. Models that leverage multiple sequence alignment information are shaded.



(a) (b)

Figure 6: **SCISOR Shrinks proteins while maintaining their fold-ability and active site motifs.** We take 100 sequences from Uniprot that have binding or active site annotations and shrink them to various amounts. We measure (a) the foldability (pLDDT from OmegaFold [27]) and (b) conservation of annotated functional regions, measured by the enrichment ratio of unmodified active or binding sites relative to random expectation. Note Raygun can do worse than random deletions because it can also add substitutions.

outperforms all previous large models. Remarkably, SCISOR is even competitive with PoET [25], a large model that has access to extra information about protein families.

5.2 Shrinking proteins

We now compare the ability of models to take long sequences X of length L and return shrunken versions $\tilde{X}$ of length $L - M$ while preserving their function. We will measure the fold-ability of shrunken sequences as well as how often deletions avoid known functional annotations.

Raygun requires 1 model evaluation to make M deletions, while SCISOR requires M. For ProGen2, ideally we would take a sample from its likelihood conditioned on looking at substrings of X that are length $L - M$; however, that would require $\binom{L}{M}$ model evaluations, which is prohibitively expensive. While there are a number of elaborate methods one could devise to search this space, we look at the most computationally efficient baseline – we predict the effect of all L single deletion assume and assume each has an independent effect on the probability, then we sample sets of deletions without replacement (see App. D). Note even this most efficient algorithm makes L model evaluations. For situations in which a practitioner is not interested in generating diverse samples, we also try greedily choosing deletions with ProGen2 and SCISOR; there's no obvious way to do this with Raygun. In Fig. 6 we see SCISOR consistently suggests shrunken proteins that are more likely to be both foldable and preserve functional sites than Raygun and ProGen2 baselines when sampling or greedily choosing deletions.

6 Conclusion

By proposing a new family of generative models that learn to build natural sequences by deleting, SCISOR, we have built models that can effectively shrink proteins. Future work may seek to address some of the conceptual limitations of the SCISOR process discussed in App. F.

Acknowledgments and Disclosure of Funding

We thank Alex Ali and Andres Potapczynski for helpful feedback on the manuscript. This work is supported by NSF CAREER IIS-2145492, NSF CDS&E-MSS 2134216, NSF HDR- 2118310, Capital One, and an Amazon Research Award.

References

[1] Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Alex Xijie Lu, Nicolo Fusi, Ava Pardis Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *bioRxiv*, September 2023.

[2] Godwin A Aleku, Henry Man, Scott P France, Friedemann Leipold, Shahed Hussain, Laura Toca-Gonzalez, Rebecca Marchington, Sam Hart, Johan P Turkenburg, Gideon Grogan, and Nicholas J Turner. Stereoselectivity and structural characterization of an imine reductase (IRED) from *Amycolatopsis orientalis*. *ACS Catal.*, 6(6):3880–3889, June 2016.

[3] Alan Nawzad Amin, Nate Gruver, and Andrew Gordon Wilson. Why masking diffusion works: Condition on the jump schedule for improved discrete diffusion. In *Frontiers in Probabilistic Inference: Learning meets Sampling*, April 2025.

[4] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Adv. Neural Inf. Process. Syst.*, 34:17981–17993, 2021.

[5] 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.

[6] Andrew Campbell, William Harvey, Christian Dietrich Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, November 2023.

[7] Kapil Devkota, Daichi Shonai, Joey Mao, Young Su Ko, Wei Wang, Scott Soderling, and Rohit Singh. Miniaturizing, modifying, and magnifying nature's proteins with raygun. *Bioinformatics*, (biorxiv;2024.08.13.607858v2), August 2024.

[8] Nate Gruver, Samuel Don Stanton, Nathan C Frey, Tim G J Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew Gordon Wilson. Protein design with guided discrete diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, November 2023.

[9] Tony P Huang, Zachary J Heins, Shannon M Miller, Brandon G Wong, Pallavi A Balivada, Tina Wang, Ahmad S Khalil, and David R Liu. High-throughput continuous evolution of compact Cas9 variants targeting single-nucleotide-pyrimidine PAMs. *Nature Biotechnology*, 41 (1):96–107, September 2022.

[10] Daniel D Johnson, Jacob Austin, Rianne van den Berg, and Daniel Tarlow. Beyond in-place corruption: Insertion and deletion in denoising probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

[11] David G Kendall. On the generalized "birth-and-death" process. *Ann. Math. Stat.*, 19(1):1–15, March 1948.

[12] 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. *Synthetic Biology*, (biorxiv;2022.07.20.500902v3), July 2022.

[13] Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. In *Advances in Neural Information Processing Systems 35*. Cold Spring Harbor Laboratory, July 2022.

[14] Jaina Mistry, Robert D Finn, Sean R Eddy, Alex Bateman, and Marco Punta. Challenges in homology search: HMMER3 and convergent evolution of coiled-coil regions. *Nucleic Acids Res.*, 41(12):e121, July 2013.

[15] Erik Nijkamp, Jeffrey Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. ProGen2: Exploring the boundaries of protein language models. *arXiv [cs.LG]*, June 2022.

[16] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv [cs.LG]*, June 2024.

[17] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16990–17017. PMLR, 2022.

[18] Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Yarin Gal, and Debora Marks. ProteinGym: Large-scale benchmarks for protein fitness prediction and design. *Advances in Neural Information Processing Systems*, 36: 64331–64379, December 2023.

[19] Fred Zhangzhi Peng, Pranam Chatterjee, and contributors. Faesm: An efficient pytorch implementation of evolutionary scale modeling (esm). `https://github.com/pengzhangzhi/faesm`, 2024. Efficient PyTorch implementation of ESM with FlashAttention and Scalar Dot-Product Attention (SDPA).

[20] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *arXiv [cs.CV]*, September 2017.

[21] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods*, 15(10):816–822, October 2018.

[22] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[23] 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, May 2007.

[24] 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, 03 2007. ISSN 1367-4803. doi: 10.1093/bioinformatics/btm098. URL `https://doi.org/10.1093/bioinformatics/btm098`.

[25] Timothy Fei Truong, Jr and Tristan Bepler. PoET: A generative model of protein families as sequences-of-sequences. In *Thirty-seventh Conference on Neural Information Processing Systems*, November 2023.

[26] Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *ICML*, abs/2402.18567, February 2024.

[27] Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, Jianzhu Ma, and Jian Peng. High-resolution*de novo*structure prediction from primary sequence. *bioRxiv*, page 2022.07.21.500999, July 2022.

[28] Mingyang Zhang, Ting Chen, Xun Lu, Xiaobing Lan, Ziqiang Chen, and Shaoyong Lu. G protein-coupled receptors (GPCRs): advances in structures, mechanisms and drug discovery. *Signal Transduction and Targeted Therapy*, 9(1):1–43, April 2024.

[29] Feiyu Zhao, Tao Zhang, Xiaodi Sun, Xiyun Zhang, Letong Chen, Hejun Wang, Jinze Li, Peng Fan, Liangxue Lai, Tingting Sui, and Zhanjun Li. A strategy for Cas13 miniaturization based on the structure and AlphaFold. *Nat. Commun.*, 14(1):5545, September 2023.

[30] Yixiu Zhao, Jiaxin Shi, Lester Mackey, and Scott Linderman. Informed correctors for discrete diffusion models. *arXiv [cs.LG]*, July 2024.

A Code release

We release our code and model weights for SCISOR small (35M), medium (150M), and large (650M) trained on UniRef50 and Uniref90: `https://anonymous.4open.science/r/shortening_diffusion-1AA3/`.

B Related work

In chemistry and language modeling, there have been diffusion models that have attempted to allow for insertions and deletions. Campbell et al. [6] propose TDDM, a jump diffusion model to handle varying dimensionality. Their forward noising process involves randomly deleting elements, such that the stationary distribution is an empty sequence. This allows them to train a model which can learn to expand sequences. Our model, SCISOR, on the other hand shortens sequences. As well, Johnson et al. [10] formulate a discrete-time noising process for small-scale language modeling that includes insertions, deletions, and substitutions. Unfortunately it is unclear how to scale their loss computation or the parameterization of their de-noiser to larger scales. Furthermore, it is unclear how to extend their framework out of discrete-time diffusion, which is known to under-perform continuous-time diffusion [5]. By using a continuous-time insertion-only forward process, we overcome this challenging inference problem and obtain an intuitive parameterization of the reverse process.

Recently, Raygun [7] also suggested using a model trained on sequences from nature to shrink proteins. Raygun trains a stochastic autoencoder to embed and generate sequences of any length on the UniRef dataset; their insight is they can shrink a long protein by decoding its embedding at a shorter length. However, they cannot enforce similarity between the sequence of their shrunken and original sequence. Furthermore, like previous generative models of protein sequences, Raygun was not specifically trained to shrink. Below we show that our model, SCISOR, is able to suggest shrunken proteins that more often preserve structure and function than Raygun.

C Details about SCISOR

C.1 Prior matching KL term

We rewrite the first term of Eqn. 3 so we can estimate it.

Proposition C.1. *(Proof in App. G)* $\mathrm{KL}(p(X_1 \mid X_0, M_1)||q(X_1|L+M_1))$ *is equal to*

$$\mathbb{E}_{X_1|X_0,M_1}\left[\log\binom{M_1+L}{L} + \sum_{i=1}^{L}\log\pi(X_0^{(i)}) - \log \mathrm{ali}(X_0, X_1)\right]. \tag{4}$$

We can therefore estimate the first term of the loss in Eqn. 3 by sampling X_0, M_1, X_t as calculating the quantity in the expectation of Eqn. 4.

C.2 Using the de-noiser to generate and shorten sequences

The SCISOR de-noiser q_θ is trained as a generative model of natural sequences. In this section, we describe how to use this de-noiser for downstream tasks: to unconditionally generate natural sequences, predict the effect of deletions on a protein's function, and, ultimately, shrink long sequences to produce shorter natural sequences.

High-quality unconditional generation As described in Sec. 3.2, to sample a sequence of length L from SCISOR, one samples a long random sequence from $\mathbb{E}_{M_1|L}q(X_1|L+M_1)$ and then iteratively deletes according to the de-noiser q_θ (Fig. 3a). Campbell et al. [6] suggests continuous-time discrete diffusion models can get higher quality samples, sacrificing some compute, by applying "corrector" steps which noise and de-noise repeatedly. For SCISOR, this takes the form of adding and removing insertions as in Alg. 3. This allows SCISOR to more thoroughly search the space of deletions, potentially escaping local minima. In cases where many passes through the model is too expensive, we can make multiple deletions per de-noiser prediction, as discussed in App. C.

Algorithm 3 Unconditional sequence generation with corrector steps

Require: Desired sequence length L, corrector steps K.
Sample $M \sim \text{NegativeBinomial}(L+1,\ \alpha(1))$
Sample X of length $L+M$ where each $X^{(j)} \sim \text{Cat}(\pi)$ independently
while $|X| > L$ **do**
  for $k = 1, \ldots, K$ **do** ▷ Corrector steps
    Remove one letter from X according to $q_\theta(\text{prev}(X) \mid X, M)$
    Insert a random letter from the distribution π into a random position in X
  end for
  Remove one letter from X according to $q_\theta(\text{prev}(X) \mid X, M)$
  $M \leftarrow M - 1$
end while
return X

Note in this algorithm, SCISOR is not simply sampling deletions by how natural $\text{prev}(X)$ looks. Rather it also uses knowledge of M to plan for future mutations. Different values of M allow the model to change which deletions it will allow at each step (Fig. 3b).

Mutation effect prediction Say we have a sequence X and we wish to predict the effect of the deletion of every position to understand the importance of each residue. Typically, we would take a model trained on protein sequences, p_θ and then evaluate the "natural-ness" of the sequence with each deletion $p_\theta(X^{(-i)})$ where $X^{(-i)}$ is the deletion of letter i [21]. Unfortunately estimating the likelihood is challenging for diffusion models as one needs to estimate the expectation in Eqn. 1.

SCISOR instead simply predicts $q_\theta(X^{(-i)} \mid X, M = 1)$ for every possible deletion $X^{(-i)}$. Then if the de-noiser suggests that a residue is unlikely to be deleted, that suggests that X without that residue does not look like a sample from $q_\theta(X_0)$, i.e. a natural sequence, and thus that deletion may harm function. For multi-letter deletions, we integrate over all deletion paths (see App. C).

Protein shrinking The SCISOR de-noiser is trained to suggest the next deletion in a series of M deletions that will lead to a realistic sample from $p(X_0)$. Therefore we can shrink a sequence X to a desired length $L - M$ by iteratively deleting according to our de-noiser as in Alg. 2.

C.3 Efficient sampling

Alg. 3 implements the Gillespie algorithm for a stochastic process. Zhao et al. [30] and Amin et al. [3] suggested k-Gillespie for diffusion models, taking k steps at every step by sampling without replacement. Indeed We can do the same for SCISOR, sampling many deletions at each step without replacement.

C.4 Multi-deletion prediction

Say $\tilde{X}$ is the sequence X with M deletions at sites $\{i_1, \ldots, i_M\}$. We wish to calculate $q_\theta(X_0 = \tilde{X} \mid X, M)$. We can break this up into a sum over all deletions using the de-noiser

$$q_\theta(X_0 = \tilde{X} \mid X, M) = \sum_{m=1}^{M} q_\theta(X_0 = \tilde{X} \mid X^{(-i_m)}, M-1) q_\theta(\text{prev}(X) = X^{(-i_m)} \mid X, M).$$

Continuing like this, we can write $q_\theta(X_0 = \tilde{X} \mid X, M)$ as a sum over all permutations of the deletions.

C.5 SCISOR in practice

We train the SCISOR de-noiser with mini-batch gradient descent on the second term of Eqn. 3 with i.i.d. samples of $t \sim \text{Uniform}(0,1), X_0, M_t, X_t$. We now discuss how we choose the rate function $\beta(t)$, the distribution of insertion letters π, the architecture for q_θ, and methods to handle the wild variation in sequence lengths of X_t which we must pass to q_θ.

Algorithm 4 Predicting the functional effect of multiple deletions with SCISOR

Require: Initial sequence X, deletions $\{i_1, \ldots, i_M\}$.
$P \leftarrow$ all permutations of $\{i_1, \ldots, i_M\}$
$\text{SUM} \leftarrow 0$
for $j_1, \ldots, j_M \in P$ **do**
  $\text{SUM} = \text{SUM} + \prod_{M'=0}^{M-1} q_\theta(X^{(-j_1,\ldots,j_{M'+1})} \mid X^{(-j_1,\ldots,j_{M'})}, M')$
end for
return $\text{SUM} = q_\theta(X_0 = \tilde{X} \mid X, M)$

Hyperparameters Our choice of hyperparameters follows that of standard diffusion methods. As in Austin et al. [4], Amin et al. [3], the rate function $\beta(t)$ was chosen so that the mutual information between X_t and X_0 decreases roughly linearly on the interval $t \in [0, 1]$. We then modulated β so that $\alpha(1)$ was large enough that the first term of Eqn. 3 is small, while samples in the second term did not get to many very long X_t. Details are below. The categorical distribution π was chosen to match the prevalence of amino acids in our training set.

Architecture We chose our architecture of the de-noiser $q_\theta(\cdot|X_t, M)$ to leverage the pre-trained weights of a BERT-style protein language model, while modifying the architecture to also condition on M. The ESM2 architectures [12] are trained on a masked language modeling task, taking in sequences and outputting logits at every site. We finetuned these models for q_θ by replacing their last layer with a linear and softmax layer. To condition on M, we also add FiLM layers [20] between each attention block: each coordinate d of the activations in layer ℓ, a_d^ℓ, was modified with and affine linear transformation with A_θ and B_θ shallow fully connected networks initialized to 0:

$$(1 + A_{\theta,d}^\ell(M)) \times a_d^\ell + B_{\theta,d}^\ell(M).$$

Engineering for long sequences Since X_t sequences can have wildly different lengths, training naively could result in passing batches with a very high proportion of padding and passing very long sequences into the model. To avoid the first problem, we sort the X_t sequences within a given batch by length, and pass them into the model in smaller sub-batches with accumulated gradients; this allowed us to reduce the proportion of compute spent on padding while maintaining an unbiased estimate of the loss. Next, to handle cases with extremely long X_t, if $|X_t| > 2048$, we randomly selected a window $X_t^{(w:w+2048)}$ uniformly at random to pass to the model. We then re-normalize the model predictions by $2048/|X_t|$ and use uniform predictions outside the window such that the deletion probabilities sum to 1. This choice keeps our ELBO a valid lower bound on the likelihood. Further details for how this impacts the ELBO and sampling are below.

C.6 Rate function

For simplicity, we choose a functional form

$$\beta(t) = \frac{\gamma}{1 - t_{\max} t}.$$

Consequently, we have:

$$\begin{aligned}
\alpha(t) =& \exp\left(-\int_0^t \beta(s)\, ds\right) \\
=& \exp\left(-\frac{\gamma}{t_{\max}} \int_0^t \frac{1}{1 - t_{\max} s}\, ds\right) \\
=& \exp\left(-\frac{\gamma}{t_{\max}} \ln(1 - t_{\max} t)\right) \\
=& (1 - t_{\max} t)^{\gamma / t_{\max}}
\end{aligned}$$

qnd $\alpha(1) = (1 - t_{\max})^{\gamma/t_{\max}}$. Now we must choose γ and $t_{\max}$. We found empirically on small models that $\gamma = 1.1$ gave an ELBO 3 such that the expectation conditional on each t was roughly even. We found empirically on small models that $T = 0.9$ gave the best best loss controlling for wall time, trading off allowing the model to attempt to fit larger sequences and spending compute on those large sequences.

C.7 Windowing

One challenge in efficiently training the SCISOR de-noiser is that we must compute $q_\theta(\text{prev}(X_t) \mid X_t, M_t)$, where X_t can potentially be a very long sequence. To handle these long sequences, we introduce a windowing strategy: if $|X_t| > 2048$, we randomly select a window $X_t^{(w:w+2048)}$ uniformly at random to pass to the model. We then re-normalize the model predictions by $2048/|X_t|$ (the probability of a deletion in the window is proportional to its size) and use uniform predictions outside the window such that the deletion probabilities sum to 1. Calling the predictions made by window w $q_\theta^w(\text{prev}(X_t) \mid X_t, M_t)$, we can define our model predictions as an average over all windows

$$q_\theta(\text{prev}(X_t) \mid X_t, M_t) = \mathbb{E}_w q_\theta^w(\text{prev}(X_t) \mid X_t, M_t).$$

ELBO We modify the second term of our loss Eqn. 3 to obtain another lower bound to bring the expectation outside

$$\begin{aligned}\text{KL}(p(\text{prev}(X_t) \mid X_0, X_t, M_t)||q_\theta(\text{prev}(X_t) \mid X_t, M_t)) \\ \geq \mathbb{E}_w \text{KL}(p(\text{prev}(X_t) \mid X_0, X_t, M_t)||q_\theta^w(\text{prev}(X_t) \mid X_t, M_t)).\end{aligned}$$

This gives us a new ELBO we can estimate by stochastically sampling the window w whenever we get a large sequence.

Sampling In Alg. 1, we need to sample from $q_\theta(\text{prev}(X_t) \mid X_t, M_t)$ for very long sequences. We do so by sampling a w and then sampling from $q_\theta^w(\text{prev}(X_t) \mid X_t, M_t)$.

D Experimental Details

D.1 Baselines

We used **EvoDiff** models and code from `https://github.com/microsoft/evodiff` under the MIT license. We used **DPLM** models and code from `https://github.com/bytedance/dplm` under the Apache-2.0 license. We used **ProGen2** models and code from `https://github.com/enijkamp/progen2` under the BSD-3-clause license. We used **Raygun** models and code from `https://github.com/rohitsinghlab/raygun` under the CC BY-NC 4.0 license. We used **ProteinGym** models and code from `https://github.com/OATML-Markslab/ProteinGym` under the MIT license.

D.2 SCISOR architecture

We used the flash attention implementation of ESM from Peng et al. [19] under the MIT license. We used ESM2 weights [12] also under the MIT license. We developed SCISOR using code from `https://github.com/AlanNawzadAmin/SCUD` under the MIT license.

D.3 Training SCISOR

We apply our framework to train a protein generative model on UniRef50 [24]. We filter this dataset to exclude proteins with non-standard amino acids, and crop long protein sequences down to their first 1024 amino acids.

For the results in section 4, we train SCISOR models on teh March 2020 release of Uniref50, using the same train-test split as EvoDiff [1] from `https://zenodo.org/records/6564798`. Our models were trained about one week each on one NVIDIA A100 GPU with an effective batch size of 256 and learning rate of 0.0001.

For the results in section 5, we train SCISOR models on the latest release of Uniref90. Here, we use an effective batch size of 512 and learning rate of 0.00005. The SCISOR S and M models were trained for about one week each on two NVIDIA A100 GPUs. The SCISOR L model was trained for about four days on four NVIDIA H100 GPUs.

For each effective batch, we sampled all t, X_0, M_t, X_t. We then sorted sequences by the length of X_t before breaking them into batches to pass to the model in batch sizes of 8 or 16; This makes sequences in each batch have similar length, minimizing padding.

D.4 Model fit experiments

D.4.1 Perplexities

SCISOR We compute the perplexity in Fig. 4a on the test dataset by first sub-sampling the expectation of the ELBO from Prop. 3.2 – we take 10 samples of t, X_t for every sequence. We then by the total number of tokens in the test set and report the and exponentiating the negative result.

EvoDiff and AR We take perplexity values from Table S1 in Alamdari et al. [1].

DPLM DPLM was trained as a discrete-time masking diffusion model with 500 steps and a linear rate schedule – that is, the probability of each token in X_t being masked is $t/500$. We therefore evaluated their perplexities as such a model as in Austin et al. [4]. This ELBO becomes

$$\sum_{t=1}^{500} \frac{1}{t} \mathbb{E}_{X_0, X_t} \sum_{i=1}^{L} \mathbb{1}(X_t^{(i)} = \text{mask}) \log q_\theta(X_0^{(i)} \mid X_t).$$

D.4.2 Samples

SCISOR We sampled according to Alg. 3.

EvoDiff and AR We sampled from EvoDiff and AR models using functions `generate_oaardm` and `generate_autoreg` from `https://github.com/microsoft/evodiff/blob/main/evodiff/generate.py`.

DPLM Wang et al. [26] suggested a novel sampling method for DPLM. However, we were interested in measuring the quality of DPLM samples *as a diffusion model*. We therefore took samples as such a model as in Austin et al. [4]: We start with X_{500} and for every $t = 500, \cdots, 1$ we unmask each position i with probability $1/t$, replacing the mask according to predicted probabilities $q_\theta(X_0^{(i)} \mid X_t)$.

D.4.3 sample evaluation

For FPD we took 1000 protein lengths from UniRef50 and sampled sequences of each of those lengths from SCISOR, EvoDiff, and DPLM; or we sampled 1000 sequences from the AR models. For pLDDT, we sampled 100 sequences of length 100, 200, 400, and 800 from SCISOR, EvoDiff, and DPLM; for AR models where the sample length cannot be controlled, we sampled sequences until we had a sufficient number of samples with lengths within 10% of each desired length.

Fréchet protein distance (FPD) We calculated the FPD of 1000 generated sequences to 10000 samples from UniRef50 using ProtT5 embeddings in `https://github.com/hefeda/PGP` under the Apache-2.0 license. We then calculated the Fréchet inception distance between the embeddings of the natural sequences and each set of sampled sequences as

$$\|\mu_{\text{natural}} - \mu_{\text{sample}}\|^2 + \text{tr}\left(\Sigma_{\text{natural}} + \Sigma_{\text{sample}} - 2(\Sigma_{\text{natural}}\Sigma_{\text{sample}})^{1/2}\right)$$

where $\mu_.$ and $\Sigma_.$ are empirical means and covariances of the embeddings.

pLDDT We calculate pLDDT scores using OmegaFold [27] as described in `https://github.com/HeliXonProtein/OmegaFold/blob/main/README.md` under the Apache-2.0 License. For computational efficiency, we use only 1 cycle per sample. This results in lower overall pLDDT scores than the recommended default settings, which uses 10 cycles to obtain more accurate predicted structures.

D.5 ProteinGym

D.5.1 Model predictions

SCISOR To evaluate SCISOR, we set X_t to be the target sequence and M to be the number of deletions between the target and the mutant of interest. We then predict the effect of the deletion using Alg. 4.

ProGen and other models We evaluated other models using scripts available on ProteinGym.

D.5.2 Model evaluation

For Fig. 5, we adapt the ProteinGym benchmark from [18] by filtering their indels dataset to cases where the mutant is a strict subsequence of the target sequence. For the single deletions benchmark, we use mutants that are only one deletion away from the target sequence, while for the multiple deletions benchmark, we use mutants that are two or three deletions away from the target sequence.

For single mutations, we gathered 61 assays in ProteinGym with 4544 mutations in total.

Three assays in ProteinGym measured double and triple mutations: `A4_HUMAN_Seuma_2022` measured stability and had 42 double mutations and 40 triple mutations, `KCNJ2_MOUSE_Macdonald_2022` measured expression and had 397 double mutations and 387 triple mutations, `P53_HUMAN_Kotler_2018` measured organismal fitness and had 172 double mutations and no triple mutations.

D.6 Shrinking

For Fig. 6a and 6b we sample 100 sequences with annotated active sites and 100 sequences with annotated binding sites from UniProt. We then shrink each sequence by d percent, where $d \in \{1, 3, 5, 10, 20, 30, 40, 50\}$.

D.6.1 Model samples

SCISOR We shrunk sequences using Alg. 3.

ProGen Ideally we could sample from $q_{\text{ProGen}}(\tilde{X})$ over all shrunken versions of X, $\tilde{X}$, of desired length $L - M$. However, for even moderate values of M, this becomes computationally intractable. We therefore approximate this distribution by assuming each deletion has an independent effect:

$$\log q_{\text{ProGen}}(\tilde{X}) \approx \log q_{\text{ProGen}}(X) + \sum_{\text{deletions } i} \Delta_i$$

where Δ_i is the effect of a single mutation,

$$\Delta_i = \log \frac{q_{\text{ProGen}}(X^{(i)})}{q_{\text{ProGen}}(X)}.$$

This approximation requires calculating L quantities Δ_i.

Sampling from this approximation is equivalent to sampling M deletions – deletion i is sampled with probability proportional to $\exp(\Delta_i)$. Greedy shrinking just involves picking the M mutations with the highest Δ_i.

Raygun we use the Raygun generate command to generate shrunken proteins of desired length, where length was calculated by first calculating rounded up number of deletions to introduce, and conditioning Raygun to generate sequence of length $L - M$. we used a noise ratio of 0.5 with uniform sampling (noise sampled uniformely between 0 and 0.5), in order to limit the number of substitutions introduced. We use a filter ratio of 0.1 meaning we select the best candidate among ten generated sequences, and recycle sequences once.

D.6.2 Model evaluation

We evaluate the foldability of the shrunk sequneces using the average pLDDT per residue for the structure generated using OmegaFold [27] as described in `https://github.com/HeliXonProtein/OmegaFold/blob/main/README.md` under the Apache-2.0 License, using 1 cycle per sample. We calculate enrichment as the number of active or binding sites in the original sequence that were preserved in the shrunk sequence – we call a functional site "preserved" if no residues were modifiesd or deleted.

E Supplementary Results

E.1 Full ProteinGym table

We show the results for ProteinGym for all models and sizes, stratifying the single deletions into functional, taxonomic, and MSA depth categories.

Table 1: ProteinGym results on single and multiple deletions.

Model	MSA	Single Deletions	Multiple Deletions
Progen2 S		0.457	0.445
Progen2 M		0.513	0.385
Progen2 Base		0.497	0.408
Progen2 L		0.491	0.375
Progen2 XL		0.393	0.392
RITA S		0.409	0.274
RITA M		0.448	0.318
RITA L		0.465	0.323
RITA XL		0.440	0.161
Tranception S		0.439	0.475
Tranception M		0.464	0.424
Tranception L		0.445	0.426
HMM	Yes	0.453	0.474
PoET (200M)	Yes	0.551	**0.488**
SCISOR S		0.332	0.241
SCISOR M		0.505	0.478
SCISOR L		**0.573**	0.458

Table 2: ProteinGym results on single deletions stratified by the measured function of each assay.

Model	MSA	Activity	Expression	Organismal Fitness	Stability
Progen2 S		0.566	0.294	0.499	0.470
Progen2 M		0.574	0.404	0.558	0.514
Progen2 Base		0.592	0.380	0.496	0.520
Progen2 L		0.550	0.344	0.560	0.508
Progen2 XL		0.418	0.298	0.333	0.521
RITA S		0.507	0.320	0.452	0.356
RITA M		0.514	0.345	0.500	0.432
RITA L		0.530	0.437	0.420	0.474
RITA XL		0.532	0.385	0.360	0.481
Tranception S		0.542	0.351	0.532	0.331
Tranception M		0.594	0.340	0.526	0.395
Tranception L		0.533	0.336	0.445	0.466
HMM	Yes	0.496	0.321	0.501	0.493
PoET (200M)	Yes	**0.664**	**0.424**	0.566	0.551
SCISOR S		0.376	0.289	0.198	0.465
SCISOR M		0.514	0.362	0.576	0.571
SCISOR L		0.604	0.415	**0.668**	**0.606**

Table 3: ProteinGym results on single deletions stratified by the MSA depth of proteins in each assay.

Model	MSA	Low	Medium	High
Progen2 S		0.558	0.429	0.497
Progen2 M		0.415	0.483	0.544
Progen2 Base		0.438	0.460	0.568
Progen2 L		0.513	0.473	0.532
Progen2 XL		0.216	0.499	0.530
RITA S		0.300	0.293	0.424
RITA M		0.278	0.376	0.492
RITA L		0.444	0.434	0.504
RITA XL		0.139	0.462	0.508
Tranception S		0.467	0.316	0.360
Tranception M		0.297	0.358	0.447
Tranception L		0.519	0.391	0.518
HMM	Yes	0.624	0.506	0.471
PoET (200M)	Yes	0.595	0.553	0.548
SCISOR S		0.385	0.381	0.509
SCISOR M		**0.641**	0.547	0.575
SCISOR L		0.621	**0.628**	**0.584**

Table 4: ProteinGym results on single deletions stratified by the taxa of the protein in each assay.

Model	MSA	Human	Eukaryote	Prokaryote	Virus
Progen2 S		0.506	0.467	0.361	0.567
Progen2 M		0.536	0.539	0.432	0.510
Progen2 Base		0.568	0.541	0.396	0.471
Progen2 L		0.536	0.513	0.442	0.492
Progen2 XL		0.511	0.537	0.420	0.573
RITA S		0.398	0.353	0.272	0.448
RITA M		0.496	0.417	0.310	0.504
RITA L		0.522	0.473	0.327	0.568
RITA XL		0.510	0.466	0.386	0.555
Tranception S		0.332	0.363	0.297	0.449
Tranception M		0.443	0.406	0.295	0.468
Tranception L		0.511	0.462	0.348	0.513
HMM	Yes	0.585	0.392	0.437	0.547
PoET (200M)	Yes	0.554	0.522	0.523	0.721
SCISOR S		0.486	0.418	0.340	0.652
SCISOR M		0.571	0.539	0.525	0.735
SCISOR L		**0.590**	**0.568**	**0.621**	**0.767**

F Discussion of limistations

Realistic insertion process One conceptual limitation about SCISOR is that it shrinks proteins by assuming $X \sim p(X_t)$ for some t, i.e. X resembles a natural sequence after time t of random insertions. In reality, the sequences we typically want to shrink, X, are natural, and may not resemble typical samples from $p(X_t)$ – there may be a distribution shift between our training procedure and downstream task. One way to remedy this is to make samples from $p(X_t)$ look more like natural sequences. We made progress on this by choosing a realistic distribution of insertion letters π. But future versions of SCISOR could add more structure to the insertion process.

Guiding based on function In this work, we aimed to shrink proteins into sequences that may still appear in nature and are thus likely to be functional. While two functional proteins with similar sequences are likely to have the same function, this is not guaranteed, especially in those protein

families with diverse functions [28]. Future work may incorporate other information of function into the SCISOR shrinking process. For example, one could guide the SCISOR diffusion process using a classifier trained to detect functional proteins of interest [16].

Including compensatory mutations Currently, SCISOR only shrinks proteins via deletions. It is possible however that there are substitutions or insertions that could be added to a protein to make it more tolerant to more deletions. To allow SCISOR to introduce these mutations which planning a series of deletions, we could add substitutions and deletions to the forward process, thereby training the de-noiser to also include substitutions and insertions in its planning.

Broader Impacts SCISOR has the potential to accelerate drug development and simplify drug delivery. However, like other protein models, it could aid in developing harmful biotechnologies.

G Proofs

G.1 Proof of correctness for Algorithm 1

The correctness of Alg. 1 follows from Cor. G.2.

Proposition G.1. *Call*

$$X_t = Y_0 X_0^{(1)} Y_1 X_0^{(2)} \cdots X_0^{(L)} Y_L$$

where $X_0^{(1)} X_0^{(2)} \cdots X_0^{(L)}$ *are the letters of* X_0 *and* $Y_0, Y_1, \ldots, Y_L$ *are the insertions. Then* $|Y_l|$ *is a* $\mathrm{Geom}(\alpha(t))$ *distribution, where* $\alpha(t) = \exp(-\int_0^t \beta(s)ds)$.

Proof. By the Kolmogorov forward equation,

$$\frac{d}{dt} p(|Y_l| = n|t) = \beta(t) n p(|Y_l| = n-1|t) - \beta(t)(n+1) p(|Y_l| = n|t).$$

This can be written as

$$\frac{d}{dt}\left(e^{(n+1)\int_0^t \beta(s)ds} p(|Y_l| = n|t)\right) = n e^{(n+1)\int_0^t \beta(s)ds} \beta(t) p(|Y_l| = n-1|t).$$

For $n = 0$, this is solved by $p(|Y_l| = 0|t) = \alpha(t)$. By induction,

$$p(|Y_l| = n|t) = \alpha(t)(1-\alpha(t))^n$$

as

$$\begin{aligned} n\alpha(t)^{-(n+1)}\beta(t) p(|Y_l| = n-1|t) =& n\alpha(t)^{-n}\beta(t)(1-\alpha(t))^{n-1} \\ =& \frac{d}{dt}\left(\alpha(t)^{-n}(1-\alpha(t))^n\right) \end{aligned} \tag{5}$$

□

Corollary G.2.

$$p(|Y_0|, \ldots, |Y_L|) = \alpha(t)^{|X_0|+1}(1-\alpha(t))^{\sum_l |Y_l|}$$

so $p(|Y_0|, \ldots, |Y_L|)$ *only depends on* $|X_0|$ *and* $M = \sum_l |Y_l|$. *In particular we can sample* $M \sim \mathrm{NegativeBinomial}(\alpha(t))$ *and then distribute it uniformly into* $L+1$ *bins.*

Proof. Each $Y_i$is generated independently, so we just take the product of probabilities from Prop. G.1. □

G.2 Proof of Proposition 3.1

Proposition G.3. *(Proof of Prop. 3.1) Say* X_0 *is a sequence with length* L. *Call* $q(\cdot \mid L)$ *a distribution over sequences of length* L *which simply samples each letter independently from* $\mathrm{Cat}(\pi)$ *for a distribution* π *such that* $\pi(b) > 0$ *for all letters* b. *Then, as the number of insertions increases,* $M_1 \to \infty$, X_1 *becomes easier to approximate with* q:

$$\mathrm{KL}(p(X_1 \mid X_0, M_1) || q(X_1 \mid L + M_1)) \to 0.$$

Proof. We suppress the subscript 1. Note by Lem. G.8

$$\frac{p(X \mid X_0, M)}{q(X \mid L+M)} = \frac{\mathrm{ali}(X_0, X)}{\binom{L+M}{L} \prod_{i=1}^{L} \pi(X_0^{(i)})}.$$

For a set of L indices $I = i_1 < i_2 < \cdots < i_L$, call $\chi_I = \mathbb{1}(X_0 = X^{(i_1)} \cdots X^{(i_L)})$. Then $\mathrm{ali}(X_0, X) = \sum_I \chi_I$ and $E_q \chi_I = \prod_{i=1}^{L} \pi(X_0^{(i)})$. Therefore we can write

$$\begin{aligned}
E_p \log \frac{p(X \mid X_0, M)}{q(X \mid L+M)} =& E_p \log \frac{\mathrm{ali}(X_0, X)}{\binom{L+M}{L} \prod_{i=1}^{L} \pi(X_0^{(i)})} \\
=& E_p \log \frac{\mathrm{ali}(X_0, X)}{E_q \mathrm{ali}(X_0, X)} \\
\leq& E_p \left| \frac{\mathrm{ali}(X_0, X)}{E_q \mathrm{ali}(X_0, X)} - 1 \right| \\
=& \frac{E_p \left| \mathrm{ali}(X_0, X) - E_q \mathrm{ali}(X_0, X) \right|}{E_q \mathrm{ali}(X_0, X)} \\
\leq& \frac{E_p \left| \mathrm{ali}(X_0, X) - E_p \mathrm{ali}(X_0, X) \right|}{E_q \mathrm{ali}(X_0, X)} \\
&+ \frac{\left| E_p \mathrm{ali}(X_0, X) - E_q \mathrm{ali}(X_0, X) \right|}{E_q \mathrm{ali}(X_0, X)} \\
\leq& \frac{\mathrm{Std_p}\left(\mathrm{ali(X_0, X)}\right)}{E_q \mathrm{ali}(X_0, X)} + \left| \frac{E_p \mathrm{ali}(X_0, X)}{E_q \mathrm{ali}(X_0, X)} - 1 \right|.
\end{aligned}$$

We now show that these two terms each go to 0, starting with the second term.

The second term Say X is generated by picking indices $Z = z_1 < \cdots < z_L$ which are X_0 and then generating all other letters from π Say we have indices I. Then

$$\begin{aligned}
E_p \chi_I \leq& (1 - p(I \cap Z = \emptyset)) + E_p \left[\chi_I | I \cap Z = \emptyset \right] \\
=& 1 - \frac{\binom{M+L-L}{L}}{\binom{M+L}{L}} + E_q \chi_I \\
\leq& 1 - \left(\frac{M-L}{M} \right)^L + E_q \chi_I \\
\leq& O\left(L^2/M\right) + E_q \chi_I \\
=& (1 + o(1)) E_q \chi_I.
\end{aligned}$$

Also

$$\begin{aligned}
E_p \chi_I \geq& (1 - p(I \cap Z = \emptyset)) \times E_p \left[\chi_I | I \cap Z = \emptyset \right] \\
=& \left(1 - \frac{\binom{M+L-L}{L}}{\binom{M+L}{L}} \right) \times E_q \chi_I \\
\geq& \left(1 - \left(\frac{M}{M+L} \right)^L \right) \times E_q \chi_I \\
\geq& \left(1 - O\left(L^2/M\right)\right) E_q \chi_I \\
=& (1 - o(1)) E_q \chi_I.
\end{aligned}$$

Then

$$\frac{E_p \mathrm{ali}(X_0, X)}{E_q \mathrm{ali}(X_0, X)} = \frac{\binom{M+L}{L} E_p \chi_I}{\binom{M+L}{L} E_q \chi_I} = 1 + o(1).$$

The first term We first change the expectation in the standard deviation into an expectation over q. Say X is generated by picking indices $Z = z_1 < \cdots < z_L$ which are X_0 and then generating all other letters from π Say we have indices I, J. Then

$$\begin{aligned} E_p \chi_I \chi_J \leq & (1 - p(I \cap Z, J \cap Z = \emptyset)) + E_p \left[\chi_I \chi_J | I \cap Z, J \cap Z = \emptyset\right] \\ \leq & 1 - \frac{\binom{M+L-2\times L}{L}}{\binom{M+L}{L}} + E_q \chi_I \chi_J \\ = & O(L^2/M) + E_q \chi_I \chi_J. \end{aligned}$$

We also have from above that

$$E_p \chi_I E_p \chi_J = (1 + o(1)) E_q \chi_I E_q \chi_J.$$

Then

$$\mathrm{Var}_p \mathrm{ali}(X_0, X) = \sum_{I,J} \mathrm{Cov}_p(\chi_I, \chi_J) = \binom{M+L}{L}^2 o(1) + \sum_{I,J} \mathrm{Cov}_q(\chi_I, \chi_J).$$

The first term is $o(1)$ against $E_q \mathrm{ali}(X_0, X)^2 = \binom{M+L}{L}^2 (E\chi_I)^2$, so we can just focus on the second term, $\mathrm{Var}_q \mathrm{ali}(X_0, X)$.

Note if $I \cap J = \emptyset$ then $\mathrm{Cov}_q(\chi_I, \chi_J) = 0$. Then

$$\begin{aligned} \sum_J \mathrm{Cov}_1(\chi_I \chi_J) \leq & \binom{M+L}{L} - \binom{M+L-L}{L} \\ = & o\left(\binom{M+L}{L}\right) \end{aligned}$$

using the same logic as above. Therefore, $\mathrm{Var}_q \mathrm{ali}(X_0, X) = o\left(\binom{M+L}{L}^2\right) = o(E_q \mathrm{ali}(X_0, X)^2)$. This completes the proof.

□

G.3 Proof of Proposition 3.2

Proposition G.4. *(Proof of Prop. 3.2) Define M_t as the number of mutations up to time t, and $\mathrm{prev}(X_t)$ is the last sequence that gained an insertion to become X_t. Then the negative log likelihood of a sequence of length L, $-\log q_\theta(X_0|L)$, is smaller than*

$$\begin{aligned} & \mathbb{E}_{M_t} \mathrm{KL}(p(X_1 \mid X_0, M_1) || q(X_1 | L + M_1)) \\ & \quad + \mathbb{E}_{t, X_t, M_t} \frac{M_t \beta(t)}{1 - \alpha(t)} \mathrm{KL}(p(\mathrm{prev}(X_t) \mid X_0, X_t, M_t) || q_\theta(\mathrm{prev}(X_t) \mid X_t, M_t)) \end{aligned}$$

Proof. The proof of Prop. 4.4 from Amin et al. [3] derives an ELBO

$$\begin{aligned} & \mathbb{E}_{M_t} \mathrm{KL}(p(X_1 \mid X_0, M_1) || q(X_1 | L + M_1)) \\ & \quad + \mathbb{E}_{t, X_t, M_t} w(M_t, t, X_0) \mathrm{KL}(p(\mathrm{prev}(X_t) \mid X_0, X_t, M_t) || q_\theta(\mathrm{prev}(X_t) \mid X_t, M_t)) \end{aligned}$$

where

$$w(M_t, t, X_0) = \lim_{\epsilon \to 0} E[\#\text{events in } [t - \epsilon, t] | M_t, X_0]/\epsilon.$$

The following lemma shows this result. □

Lemma G.5.

$$w(M, t, X_0) = M \frac{\beta(t)}{1 - \alpha(t)}$$

Proof. First we change out time variable to $\tau = -\log \alpha(t)$. Noting $-\log \alpha(t - \epsilon) = \tau - \epsilon \beta(\tau) + O(\epsilon^2)$, we have

$$w(M, t, X_0) = \beta(t) \lim_{\epsilon \to 0} \tilde{\mathbb{E}}[\#\text{events in } [\tau - \epsilon, \tau] | M_\tau = M]/\epsilon \tag{6}$$

where $\tilde{\mathbb{E}}$ is as if the rate β were constant. In SCUD, events occur uniformly in the time interval, so the RHS would be $M/\tau = M/(-\log\alpha(t))$. For SCISOR, events are more concentrated later in time since more insertions increases the rate of insertion.

$$\begin{aligned}\tilde{\mathbb{E}}[\#\text{events in } [\tau-\epsilon,\tau]|M_\tau = M] =& P[M_{\tau-\epsilon} = M-1|M_\tau = M] + O(\epsilon^2)\\ =& \frac{P[M_{\tau-\epsilon} = M-1]}{P[M_\tau = M]} P[M_\tau = M|M_{\tau-\epsilon} = M-1] + O(\epsilon^2).\end{aligned} \tag{7}$$

Note

$$\begin{aligned}P[M_\tau = M \mid M_{\tau-\epsilon} = M-1] =& P[M_\tau \geq M \mid M_{\tau-\epsilon} = M-1] + O(\epsilon^2)\\ =& P(\text{Exp}(M+|X_0|) \leq \epsilon) + O(\epsilon^2)\\ =& 1 - e^{-\epsilon(M+|X_0|)} + O(\epsilon^2)\\ =& \epsilon(M+|X_0|) + O(\epsilon^2).\end{aligned} \tag{8}$$

Finally,

$$\begin{aligned}\frac{P[M_\tau = M-1]}{P[M_\tau = M]} =& \frac{\text{NegBin}(|X_0|, e^{-\tau}; M-1)}{\text{NegBin}(|X_0|, e^{-\tau}; M)}\\ =& \frac{\binom{m-1+|X_0|}{m-1}(1-e^{-\tau})^{M-1}}{\binom{M+|X_0|}{M}(1-e^{-\tau})^{M}}\\ =& \frac{M}{(M+|X_0|)(1-e^{-\tau})}.\end{aligned} \tag{9}$$

This gives us

$$w(M,t,X_0) = M\frac{\beta(t)}{1-\alpha(t)}$$

which is similar for small alpha to the SCUD weight of $w(M,t,X_0) = M\frac{\beta(t)}{-\log\alpha(t)}$ but becomes larger at larger values. □

G.4 Proof of Proposition 3.3

Proposition G.6. *(Proof of Prop. 3.3) Call* $\text{ali}(X,Y)$ *the number of ways to align a sequence* X *to a sequence* Y*.*

$$p(\text{prev}(X_t)|X_0, X_t, M_t) = \frac{\text{ali}(X_0, \text{prev}(X_t))}{M_t \cdot \text{ali}(X_0, X_t)}.$$

Proof. Say Y_t is X_t with a single deletion, the letter b. By Lem. G.8

$$\begin{aligned}p(Y_t \mid X_0, X_t, M_t) =& \frac{p(Y_t \mid X_0, M_t - 1)}{p(X_t \mid X_0, M_t)} p(X_t \mid Y_t)\\ =& \frac{\binom{L+M_t-1}{L}^{-1}\text{ali}(X_0, Y_t)}{\binom{L+M_t}{L}^{-1}\text{ali}(X_0, X_t)\pi(b)}\pi(b)(L+M_t)^{-1}\\ =& \frac{(L+M_t)\text{ali}(X_0, Y_t)}{M_t\text{ali}(X_0, X_t)}\\ =& \frac{\text{ali}(X_0, Y_t)}{M_t\text{ali}(X_0, X_t)}.\end{aligned}$$

Note finally that we've ignored that there may be multiple deletions that take X_t to Y_t when calculating $p(X_t \mid Y_t)$. We can safely do so as it does not affect the loss Eqn. 3 of any of our other algorithms. □

G.5 Derivation of prior matching KL term

Proposition G.7. *(Proof of Prop. C.1)* $\mathrm{KL}(p(X_1 \mid X_0, M_1)||q(X_1|L+M_1))$ *is equal to*

$$\mathbb{E}_{X_1|X_0,M_1}\left[\log\binom{M_1+L}{L} + \sum_{i=1}^{L}\log\pi(X_0^{(i)}) - \log\mathrm{ali}(X_0, X_1)\right].$$

Proof. From Lem. G.8,

$$p(X_1 \mid X_0, M_1) = \binom{M_1+L}{L}^{-1}\mathrm{ali}(X_0, X_1)\prod_{b\in X_1\setminus X_0}\pi(b).$$

Given that $q(X_1|L+M_1) = \prod_{b\in X_1}\pi(b)$, this finishes the proof. □

G.6 Useful lemma

Lemma G.8. *Calling the letters in X_t that are not in X_0 $X_t \setminus X_0$,*

$$p(X_t \mid X_0, M_t) = \binom{L+M_t}{L}^{-1}\mathrm{ali}(X_0, X_t)\prod_{b\in X_t\setminus X_0}\pi(b)$$

Proof. To generate X_t from X_0, M_t, we could (1) decide which positions $i_1, \ldots, i_L \in 1, \ldots, L+M_t$ should come from X_0 and then generate the rest of the letters according to π. Then

$$\begin{aligned} p(X_t \mid X_0, M_t) &= \sum_{\text{indices } i_1,\ldots,i_L}\frac{\mathbb{1}(X_0 = X_t^{(i_1)}\cdots X_1^{(i_l)})}{\binom{L+M_t}{L}} \times \prod_{b\in X_t\setminus X_0}\pi(b) \\ &= \binom{L+M_t}{L}^{-1}\mathrm{ali}(X_0, X_t)\prod_{b\in X_t\setminus X_0}\pi(b). \end{aligned}$$

□

H Alignments algorithm

Both KL terms in the ELBO make use of the primitive $\mathrm{ali}(X, Y)$. In particular, the denoising KL term requires computing the number of alignments between X_0 and each possible $\mathrm{prev}(X_t)$, a total of $|X_t|$ computations. Naively, computing the alignments between each pair of sequences takes $\mathcal{O}(|X_0|\cdot|X_t|)$ time for a total of $\mathcal{O}(|X_0|\cdot|X_t|^2)$. However, we devise an efficient dynamic programming algorithm to compute all of the alignment terms in parallel in $\mathcal{O}(|X_0|\cdot|X_t|)$ time, presented in Algorithm 5.

Algorithm 5 Compute $\mathrm{ali}(X_t^{-(l)}, X_0)$ for all l in parallel

Require: Sequences X_0 with $|X_0| = L$ and X_t with $|X_t| = N$

Set matching$[i, j] \leftarrow \mathbb{I}(X_0^{(i)} = X_t^{(j)})$ for all $i \in \{1, \ldots, L\}$, $j \in \{1, \ldots, N\}$
Initialize prefix_dp $\leftarrow \mathbf{0}^{(N+1)\times(L+1)}$
Set prefix_dp$[i, 0] \leftarrow 1$ for all $i \in \{1, \ldots, N\}$
for $l = 1$ to L **do**
  prod $\leftarrow$ prefix_dp$[1 : N, l-1] \times$ matching$[l-1, 1 : N]$
  prefix_dp$[1 : N+1, l] \leftarrow$ cumsum(prod, axis $= 0$)
end for
Initialize suffix_dp $\leftarrow \mathbf{1}^{(N+1)\times(L+1)}$
for $l = L-1$ down to 0 **do**
  prod $\leftarrow$ suffix_dp$[1 : N+1, l+1] \times$ matching$[l, 1 : N]$
  suffix_dp$[1 : N, l] \leftarrow$ cumsum(prod, axis $= 0$)
end for
alignments$[l] \leftarrow \sum_{i=1}^{N}$ prefix_dp$[i, l] \times$ suffix_dp$[i+1, l+1]$ for all l
return alignments