Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem

Brian L. Trippe[*] [†] btrippe@mit.edu
Jason Yim[*][†] jyim@mit.edu
Doug Tischer [‡] dtischer@uw.edu
David Baker[‡] dabaker@uw.edu
Tamara Broderick[†] tbroderick@mit.edu
Regina Barzilay[†] regina@csail.mit.edu
Tommi Jaakkola[†] tommi@csail.mit.edu

Abstract

Construction of a *scaffold* structure that supports a desired *motif*, conferring protein function, shows promise for the design of vaccines and enzymes. But a general solution to this motif-scaffolding problem remains open. Current machine-learning techniques for scaffold design are either limited to unrealistically small scaffolds (up to length 20) or struggle to produce multiple diverse scaffolds. We propose to learn a distribution over diverse and longer protein backbone structures via an E(3)-equivariant graph neural network. We develop `SMCDiff` to efficiently sample scaffolds from this distribution conditioned on a given motif; our algorithm is the first to theoretically guarantee conditional samples from a diffusion model in the large-compute limit. We evaluate our designed backbones by how well they align with AlphaFold2-predicted structures. We show that our method can (1) sample scaffolds up to 80 residues and (2) achieve structurally diverse scaffolds for a fixed motif.

1 Introduction

A central task in protein design is creation of a stable *scaffold* to support a target *motif* – that is, a particular structural protein fragment conferring biological function. Vaccines and enzymes have already been designed by solving certain instances of this *motif-scaffolding* problem [19, 5, 14, 22]. However, successful solutions to this problem in the past have necessitated substantial expert involvement and laborious trial and error. Machine learning (ML) offers the hope to automate, and better direct this search. But existing ML approaches face one of two major roadblocks. First, these methods do not build scaffolds longer than about 20 residues; for many motif sizes of interest, the resulting proteins would be smaller than the shortest commonly-studied simple protein folds (35–40 residues) [9]. Second, these methods require hours of computation to generate a single plausible scaffold [27, 1, 25]. Moreover, when a plausible scaffold is found, it remains to be experimentally validated. Therefore, it is desirable to return not just a single scaffold but rather a set of scaffolds exhibiting diverse sequences and structural variation to increase the likelihood of success in practice.

Generative models have been shown to capture a distribution over diverse protein structures [18]. But it is not clear how to handle conditioning (on the motif) using these approaches. Diffusion probabilistic models (DPMs) [11] offer a potential alternative; not only do they provide a more straightforward path to handling conditioning, but they have also enjoyed success generating small-molecules in 3D [13]. We develop a novel motif-scaffolding procedure based on Sequential Monte Carlo, `SMCDiff`, that repurposes an *unconditionally* trained DPM on protein 3D coordinates for

[*]Contributed equally to this work.
[†]Massachusetts Institute of Technology
[‡]University of Washington

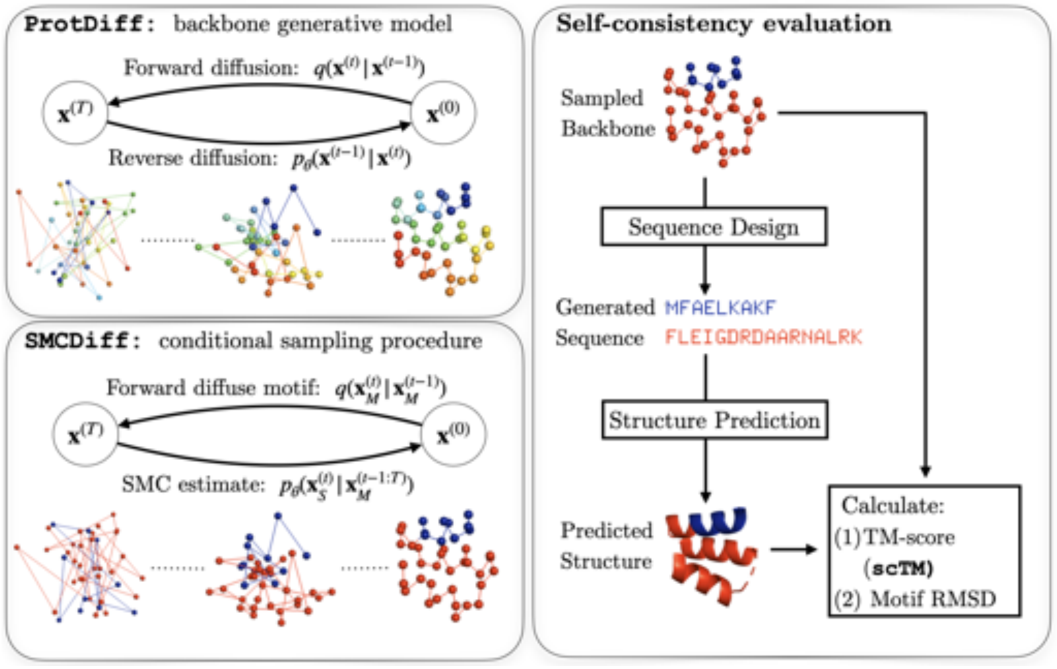
ProtDiff: backbone generative model
Forward diffusion: $q(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)})$
$\mathbf{x}^{(T)}$
$\mathbf{x}^{(0)}$
Reverse diffusion: $p_\theta(\mathbf{x}^{(t-1)} | \mathbf{x}^{(t)})$
SMCDiff: conditional sampling procedure
Forward diffuse motif: $q(\mathbf{x}_M^{(t)} | \mathbf{x}_M^{(t-1)})$
$\mathbf{x}^{(T)}$
$\mathbf{x}^{(0)}$
SMC estimate: $p_\theta(\mathbf{x}_S^{(t)} | \mathbf{x}_M^{(t-1:T)})$
Self-consistency evaluation
Sampled Backbone
Sequence Design
Generated Sequence MFAELKAKF FLEIGDRDAARNALRK
Structure Prediction
Predicted Structure
Calculate: (1) TM-score (scTM) (2) Motif RMSD

Figure 1: Overview of the conditional generative modeling approach to the motif-scaffolding problem. We train our new protein backbone diffusion model, `ProtDiff`, to generate realistic protein backbone structures. Next, we run `SMCDiff`, our conditional sampling algorithm, with `ProtDiff` to generate scaffolds (colored in red) conditioned on the motif (colored in blue). For self-consistency evaluation, we use a pretrained fixed-backbone sequence-design model (`ProteinMPNN` [6]) to generate the scaffold sequence from a sampled backbone. We then input the sequence to a structure prediction model, in our case `AlphaFold2` (AF2) [15], to generate the full protein structure from the generated sequence. We compare the backbone of the predicted structure with the original backbone structure using TM-score [28] and root-mean-square-distance (RMSD) for the motif.

conditional sampling. In our case, we condition on the motif structure, a task analogous to *inpainting* [20]. Specifically, our motif-scaffolding generative framework has two steps as depicted in Fig. 1: first we train a DPM (`ProtDiff`) to learn a distribution over protein backbones, and then we use particle filtering (`SMCDiff`) with `ProtDiff` to inpaint arbitrary motifs.

Ours is the first machine-learning method to construct scaffolds longer than 20 residues around motifs — we build up to 80 residues scaffolds on a test case. Beyond our progress on the motif-scaffolding problem, we provide the following technical contributions: (1) we introduce a protein-backbone generative model in 3D – with the ability to generate backbone samples that structurally agree with `AlphaFold2` [15] predictions, and (2) we develop a novel conditional sampling algorithm for inpainting.

2 Method

A protein can be represented by its amino acid sequence and backbone structure. Let $\mathcal{A}$ be the set of 20 genetically-encoded amino acids. We denote the sequence of an N-residue protein by $s \in \mathcal{A}^N$ and its C-α backbone coordinates in 3D by $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T \in \mathbb{R}^{N,3}$. A protein's structure is a function of its sequence, so we may write $\mathbf{x}(s)$. We divide the N residues into the functional motif $\mathcal{M}$ and the scaffold $\mathcal{S}$, such that $\mathcal{M} \cup \mathcal{S} = \{1, 2, \ldots, N\}$. The goal is to identify, given the motif structure $\mathbf{x}_\mathcal{M}$, sequences s whose structure recapitulates the motif to high precision $\mathbf{x}(s)_\mathcal{M} \approx \mathbf{x}_\mathcal{M}$.

2.1 `ProtDiff`: A diffusion model of protein backbones in 3D

Implementing DPM for protein backbones requires constructing a noise prediction model which we denote as $\epsilon_\theta(\mathbf{x}^{(t)}, t)$. The DPM framework, first introduced in Ho et al. [11], is discussed in Appendix A.1. In this section we describe `ProtDiff`, which corresponds to the choice of $\epsilon_\theta(\mathbf{x}^{(t)}, t)$ as a translation and rotation equivariant graph neural network tailored to modeling protein backbones. The properties and functions of proteins are dictated by the relative geometry of their residues, and are invariant to the coordinate system chosen to encode them. Recent work on neural network modeling of 3D data has found, both theoretically and empirically, that neural networks constrained to satisfy geometric invariances can provide inductive biases that improve generalization and training efficiency [2]. Motivated by this observation, we parameterize ϵ_θ by an equivariant graph neural network (EGNN) [21], which in 3D is equivariant to transformations in the Euclidean group. It was shown in Xu et al. [29] that if ϵ_θ is equivariant to a group then samples from the diffusion process with a invariant reference distribution will be equivariant to the same group.

Tailoring EGNN to protein backbones. Our EGNN implementation is tailored to protein backbones and DPMs through the choice of edge and node features. To model every pairwise residue interaction, we represent backbones by a fully connected graph. Each node in the graph is indexed by $n \in \{1, \ldots, N\}$, and corresponds to a residue. We associate each node with coordinates $\mathbf{x}_n \in \mathbb{R}^3$ and features $h_n \in \mathbb{R}^D$. For each pair of nodes n, n' we define an edge and associate it with edge features $a_{nn'} \in \mathbb{R}^D$. We construct our EGNN by stacking L equivariant graph convolutional layers (EGCL). Each layer takes node coordinates and features as input, and outputs updated coordinates and features. We take the input to the first layer to be the initial coordinates and features $(\mathbf{x}^0, h^0) = (\mathbf{x}, h)$. Each layer $l = 1, \ldots, L$ defines an update as $(\mathbf{x}^l, h^l) = \text{EGCL}[\mathbf{x}^{l-1}, h^{l-1}]$ where for each node n

$$\mathbf{x}_n^l = \mathbf{x}_n^{l-1} + \sum_{n' \neq n} \vec{\omega}_{nn'} \cdot \phi_{\mathbf{x}}(h_n^{l-1}, h_{n'}^{l-1}, d_{nn'}, a_{nn'}) \quad \text{and} \quad h_n^l = \phi_h(h_n^{l-1}, m_n), \quad \text{for}$$

$$\vec{\omega}_{nn'} = \frac{\mathbf{x}_n^{l-1} - \mathbf{x}_{n'}^{l-1}}{\sqrt{d_{nn'}} + \gamma}, \quad m_n = \sum_{n' \neq n} \phi_e(h_n^{l-1}, h_{n'}^{l-1}, d_{nn'}, a_{nn'}), \text{ and } d_{nn'} = \|\mathbf{x}_n^{l-1} - \mathbf{x}_{n'}^{l-1}\|_2^2.$$

ϕ_e, ϕ_h, and $\phi_{\mathbf{x}}$ are fully connected neural networks, and γ is a small positive constant included for numerical stability. We write the output of EGNN after L layers as $\hat{\mathbf{x}} = \text{EGNN}[\mathbf{x}^0, h^0]$. In the context of DPM, we predict the noise with the following parameterization:

$$\epsilon_\theta(\mathbf{x}^{(t)}, t) = \hat{\mathbf{x}} - \mathbf{x}^{(t)}, \quad \hat{\mathbf{x}} = \text{EGNN}[\mathbf{x}^{(t)}, h(t)]. \tag{1}$$

Appendix A.3 discusses additional `ProtDiff` details including node and edge features.

2.2 `SMCDiff`: Conditional sampling in diffusion models by particle filtering

The second stage of our generative modeling approach to the motif-scaffolding problem is to sample scaffolds conditioned on the motif. We build on the work of Song et al. [24], who introduced a practical algorithm that generates *approximate* conditional samples. This strategy is to (1) forward diffuse the conditioning variable to obtain $\mathbf{x}_\mathcal{M}^{(1:T)} \sim q(\mathbf{x}_\mathcal{M}^{(1:T)} \mid \mathbf{x}_\mathcal{M}^{(0)})$, and then (2) for each t, sample $\mathbf{x}_\mathcal{S}^{(t)} \sim p_\theta(\mathbf{x}_\mathcal{S}^{(t)} \mid \mathbf{x}_\mathcal{M}^{(t+1)}, \mathbf{x}_\mathcal{S}^{(t+1)})$. We call this approach the *replacement method* (following [12]) and make it explicit in Appendix Algorithm 2. However, in Proposition B.1 we show that the replacement method introduces irreducible approximation error that cannot be eliminated by making p_θ more expressive. Instead, we frame approximation of $q(\mathbf{x}_\mathcal{S}^{(0)} \mid \mathbf{x}_\mathcal{M}^{(0)})$ as a sequential Monte Carlo problem that we may solve by particle filtering. Our key insight is that because $p_\theta(\mathbf{x}_\mathcal{M}^{(t-1)} \mid \mathbf{x}^{(t)})$ provides a mechanism to assess the likelihood of $\mathbf{x}_\mathcal{M}^{(t-1)}$, we can prioritize noised scaffolds that are more consistent with the motif. Particle filtering leverages this mechanism to provide a sequence of discrete approximations to each $p_\theta(\mathbf{x}_\mathcal{S}^{(t)} \mid \mathbf{x}_\mathcal{M}^{(t-1:T)})$ that look ahead by this extra step. Finally, at $t = 0$ we have an approximation to $p_\theta(\mathbf{x}_\mathcal{S}^{(0)} \mid \mathbf{x}_\mathcal{M}^{(0:T)})$. Then, using Proposition 2.1 below, we can obtain an approximate sample from $q(\mathbf{x}_\mathcal{S}^{(0)} \mid \mathbf{x}_\mathcal{M}^{(0)})$. This framing permits the application of standard particle filtering algorithms [8]. Algorithm 1 summarizes an implementation of this procedure that uses residual resampling [7] to mitigate the collapse of the sequential approximations into point masses. `SMCDiff` provides a tunable trade-off between computational cost and statistical accuracy through the choice of the number of particles K. In our next proposition we make this trade-off explicit.

Proposition 2.1. *Suppose that p_θ exactly matches the forward diffusion process such that for every $\mathbf{x}^{(t+1)}$, $p_\theta(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)}) = q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)})$ and consider any motif $\mathbf{x}_{\mathcal{M}}^{(0)}$. Let $\mathbf{x}_{\mathcal{S},K}$ be a particle chosen at random from the output of Algorithm 1 with K particles. Then $\mathbf{x}_{\mathcal{S},K}$ converges in distribution to $q(\mathbf{x}_{\mathcal{S}}^{(0)} \mid \mathbf{x}_{\mathcal{M}}^{(0)})$ as K goes to infinity.*

The significance of Proposition 2.1 is that it guarantees Algorithm 1 can provide arbitrarily accurate conditional samples provided a large enough compute budget (determined by the number of particles). To our knowledge, `SMCDiff` is the first algorithm for conditionally sampling from DPMs that can provide arbitrary accuracy. Our proof of the proposition, which we leave to Appendix B, is obtained from an application of standard asymptotics for particle filtering [4, Proposition 11.4].

3 Experiments

We empirically demonstrate the ability of our method to scaffold motifs and sample protein backbone structures. We describe our procedure for evaluating backbone designs in Section 3.1. We demonstrate the promise of our method for the motif-scaffolding problem in Section 3.2. We train a single instance of `ProtDiff` and use it across all of our experiments. We limited our training data to single chain proteins taken from PDB that are no longer than 128 residues as we found performance to be substantially worse for larger proteins. Training details can be found in Appendix D. Additional results are presented in Appendix F including results on unconditional sampling in Appendix F.2.

3.1 *In silico* evaluation of designed backbones

Recent work [27, 18] has proposed to leverage highly accurate protein structure prediction neural networks as an *in silico* proxy for evaluating computationally designed proteins. More specifically, Wang et al. [27] jointly design protein sequence and structure, and validate by comparing the design and `AlphaFold2` (AF2) [15] predicted structures. Here, our goal is to assess the quality of scaffolds generated independent of a specific sequence, so we treat fixed backbone sequence design as a downstream step as in [18].

Our evaluation with AF2 is as follows. For each generated scaffold we use a C-α only version of `ProteinMPNN` [6] with a temperature of 0.1 to sample 8 amino acid sequences likely to fold to the same backbone structure. We then run AF2 with the released CASP14[1] weights and 15 recycling iterations. We do not include a multiple sequence alignment as an input to AF2. To assess unconditionally sampled scaffolds, we then evaluate the agreement of our backbone sample with the AF2 predicted structures using the maximum TM-score [30] across all generated sequences which we refer to as `scTM`, for *self-consistency* TM-score. To assess whether prospective scaffolds generated support a motif, we compute the root mean squared distances (RMSD) of the desired and predicted motif coordinates after alignment and refer this metric as the *motif RMSD*. Appendix Algorithm 4 outlines the exact steps.

Because a TM-score > 0.5 indicates that two structures have the same fold [30], we say that a backbone is designable if `scTM` > 0.5. The ability for AF2 to reproduce the same backbone from an independently designed sequence is evidence a sequence can be found for the starting structure.

3.2 Motif-scaffolding via conditional sampling

We evaluated our motif-scaffolding approach (combining `SMCDiff` and `ProtDiff`) on motifs extracted from existing proteins in the PDB and found that our approach can generate long and diverse scaffolds that support these motifs. We chose to first evaluate on motifs extracted from proteins present in the training set because we knew that at least one stabilizing scaffold exists. We considered 2 examples with different topologies taken from PDB with ID 6exz and 5trv, which are 69 and 118 residues long, respectively. For each topology, we chose a 15–25 residue segment as the motif (see Appendix F for details). The remainder of each protein is one possible supporting scaffold. We sought to assess if we could recover this and other scaffolds with the same size and motif placement.

[1]Biannual protein folding competition where AF2 achieved first place. Weights available under Apache License 2.0 license.

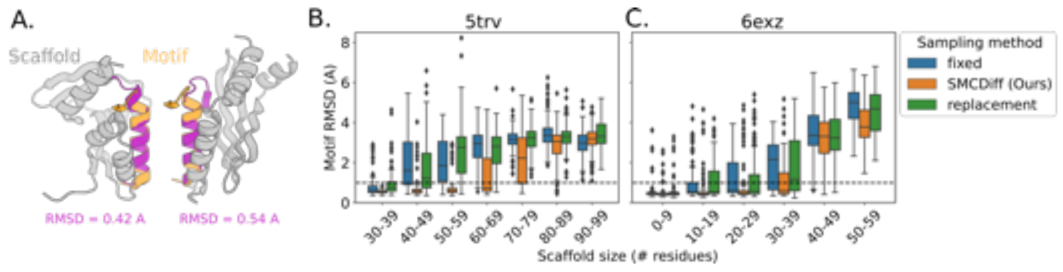

Figure 2: Motif-scaffolding case studies. (A) Example of two scaffold structures generated around a segment of `5trv`. Orange: desired input motif, Grey: AlphaFold-predicted structure of two scaffolds, with the motif highlighted (purple). Both scaffolds were sampled using `SMCDiff` with `scTM` > 0.5. (B,C) Motif RMSD for 5trv and 6exz test cases, its dependence on scaffold size, and comparison of `SMCDiff` to two naive inpainting methods (`fixed`, `replacement`).

Based on prior work [27], we expected that building larger scaffolds around a motif would be more challenging than building smaller scaffolds. To assess this length dependence, we expanded the segment of used as the motif when running `SMCDiff` by including additional residues on each side. In each case, though, we compute the motif RMSD over the minimal motif. In Figure 2B, we present motif-scaffolding performance and its dependence on scaffold size for 5trv, the longer of the two test proteins. For the 5trv test case, the lower quartile of the motif RMSD for `SMCDiff` is below 1 Å for scaffolds up to 80 residues. Since 1 Å is atomic-level resolution, we conclude that our approach can succeed in this length range.

Figure 2A provides a visualization of our method's capacity to generate long and diverse scaffolds. The figure depicts two dissimilar scaffolds of lengths 34 and 54 produced by `SMCDiff` with 64 particles. Both scaffolds are designable and agree with AF2 (`scTM` > 0.5). Diversity is particularly evident in the different orderings of secondary structures.

Figure 2B compares `SMCDiff` to two naive inpainting methods, `fixed` and `replacement`. In `fixed`, the motif is fixed for every timestep t, and the reverse diffusion is applied only to the scaffold (as in [31]); `replacement` is the method described in Section 2.2. In contrast to `SMCDiff`, these baselines fail to generate a successful scaffolds longer than 50 residues on 5trv, as determined by the location of their lower quartiles.

We next applied these three inpainting methods to a harder target. We consider a motif obtained from the respiratory syncytial virus (RSV) protein, which is not in the training dataset. We found that our method failed to generate scaffolds predicted to recapitulate the motif (Appendix F). This motif has been successfully scaffolded by the computationally intensive *hallucination* approach of [27]. Our results indicate that our method does not improve uniformly upon this previous approach.

Compute cost. The computation of `SMCDiff` with 64 particles is approximately 2 minutes per independent sample, while alternative methods `fixed` and `replacement` can produce 64 independent samples in the same time. By contrast, the hallucination approach of [27] involves running a Markov chain for thousands of steps, and has runtime on the order of hours for a single sample [1].

4 Discussion

The motif-scaffolding problem has applications ranging from medicine to material science [16], but remains unsolved for many functional motifs. We have created the first generative modeling approach to motif-scaffolding by developing `ProtDiff`, a diffusion probabilistic model of protein backbones, and `SMCDiff`, a procedure for generating scaffolds conditioned on a motif. Although our experiments were limited to a small set of proteins, our results demonstrate that our procedure is the first capable of generating diverse scaffolds longer than 20 residues with computation time reliably on the order of minutes or less. Our work demonstrates the potential of machine learning methods to be applied in realistic protein design settings.

Modeling limitations. Our present results do not indicate our procedure can generalize to motifs that are not present in the training set. We believe improvements in protein modeling could provide better inductive biases for generalization. For example, `ProtDiff` sees only 3D backbone coordinates and is blind to the the orientation of backbone residues As a consequence, `ProtDiff` is reflection symmetric and generates left-handed helices (Appendix C), which do not stably occur in natural proteins. Additionally, `ProtDiff` does not explicitly model primary sequence or side-chains. Hoogeboom et al. [13] demonstrate the benefits of modeling sequence information in small molecules, jointly modeling sequence and structure in a single model could lead to improved designability of protein scaffolds and backbones as well.

Acknowledgments and Disclosure of Funding

The authors thank Octavian-Eugen Ganea, Hannes Stärk, Wenxian Shi, Felix Faltings, Jeremy Wohlwend, Nitan Shalon, Gabriele Corso, Sean Murphy, Wengong Jin, Bowen Jing, Renato Berlinghieri, John Yang, and Jue Wang for helpful discussion and feedback. We thank Justas Dauparas for access to an early version of `ProteinMPNN`. We dedicate this work in memory of Octavian-Eugen Ganea who initiated the project by connecting all the authors.

BLT and JY were supported in part by an NSF-GRFP. BLT and TB were supported in part by NSF grant 2029016 and an ONR Early Career Grant. JY, RB, and TJ acknowledge support from NSF Expeditions grant (award 1918839: Collaborative Research: Understanding the World Through Code), Machine Learning for Pharmaceutical Discovery and Synthesis (MLPDS) consortium, the Abdul Latif Jameel Clinic for Machine Learning in Health, the DTRA Discovery of Medical Countermeasures Against New and Emerging (DOMANE) threats program, the DARPA Accelerated Molecular Discovery program and the Sanofi Computational Antibody Design grant. DT and DB were supported with funds provided by a gift from Microsoft. DB was additionally supported by the Audacious Project at the Institute for Protein Design, the Open Philanthropy Project Improving Protein Design Fund, an Alfred P. Sloan Foundation Matter-to-Life Program Grant (G-2021-16899) and the Howard Hughes Medical Institute.

References

[1] Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, Frank DiMaio, Lauren Carter, Cameron M Chow, Gaetano T Montelione, and David Baker. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, 2021.

[2] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13, 2022.

[3] Nicolas Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

[4] Nicolas Chopin and Omiros Papaspiliopoulos. *An introduction to sequential Monte Carlo*. Springer, 2020.

[5] Bruno E Correia, John T Bates, Rebecca J Loomis, Gretchen Baneyx, Chris Carrico, Joseph G Jardine, Peter Rupert, Colin Correnti, Oleksandr Kalyuzhniy, Vinayak Vittal, Mary J Connell, Eric Stevens, Alexandria Schroeter, Man Chen, Skye Macpherson, Andreia M Serra, Yumiko Adachi, Margaret A Holmes, Yuxing Li, Rachel E Klevit, Barney S Graham, Richard T Wyatt, David Baker, Roland K Strong, James E Crowe, Jr, Philip R Johnson, and William R Schief. Proof of principle for epitope-focused vaccine design. *Nature*, 507(7491):201–206, 2014.

[6] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile I M Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, Phillip J Y Leung, Tim F Huddy, Sam Pellock, Doug Tischer, Frederick Chan, Brian Koepnick, Ryan Nguyen, Shoukai Kang, B Sankaran, Asim K Bera, Neil P King, and David A Baker. Robust deep learning based protein sequence design using ProteinMPNN. *bioRxiv*, 2022.

[7] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

[8] Arnaud Doucet, Nando De Freitas, and Neil James Gordon. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

[9] Hannah Gelman and Martin Gruebele. Fast protein folding kinetics. *Quarterly Reviews of Biophysics*, 47(2):95–142, 2014.

[10] Alex Herbert and MJE Sternberg. MaxCluster: a tool for protein structure comparison and clustering. 2008.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[12] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *Deep Generative Models for Highly Structured Data Workshop, ICLR*, volume 10, 2022.

[13] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. *arXiv preprint arXiv:2203.17003*, 2022.

[14] Lin Jiang, Eric A Althoff, Fernando R Clemente, Lindsey Doyle, Daniela Rothlisberger, Alexandre Zanghellini, Jasmine L Gallaher, Jamie L Betker, Fujie Tanaka, Carlos F Barbas III, Donald Hilvert, Kendal N Houk, Barry L. Stoddard, and David Baker. De novo computational design of retro-aldol enzymes. *Science*, 319(5868):1387–1391, 2008.

[15] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583 – 589, 2021.

[16] Neil P King, William Sheffler, Michael R Sawaya, Breanna S Vollmar, John P Sumida, Ingemar André, Tamir Gonen, Todd O Yeates, and David Baker. Computational design of self-assembling protein nanomaterials with atomic level accuracy. *Science*, 336(6085):1171–1174, 2012.

[17] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[18] Zeming Lin, Tom Sercu, Yann LeCun, and Alexander Rives. Deep generative models create new and diverse protein structures. In *Machine Learning for Structural Biology Workshop, NeurIPS*, 2021.

[19] Erik Procko, Geoffrey Y Berguig, Betty W Shen, Yifan Song, Shani Frayo, Anthony J Convertine, Daciana Margineantu, Garrett Booth, Bruno E Correia, Yuanhua Cheng, William R Schief, David M Hockenbery, Oliver W Press, Barry L Stoddard, Patrick S Stayton, and David Baker. A computationally designed inhibitor of an Epstein-Barr viral BCL-2 protein induces apoptosis in infected cells. *Cell*, 157(7):1644–1656, 2014.

[20] Chitwan Saharia, William Chan, Huiwen Chang, Chris A Lee, Jonathan Ho, Tim Salimans, David J Fleet, and Mohammad Norouzi. Palette: image-to-image diffusion models. *arXiv preprint arXiv:2111.05826*, 2021.

[21] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning*, 2021.

[22] Justin B Siegel, Alexandre Zanghellini, Helena M Lovick, Gert Kiss, Abigail R Lambert, Jennifer L StClair, Jasmine L Gallaher, Donald Hilvert, Michael H Gelb, Barry L Stoddard, Kendall N Houk, Forrest E Michael, and David Baker. Computational design of an enzyme catalyst for a stereoselective bimolecular Diels-Alder reaction. *Science*, 329(5989):309–313, 2010.

[23] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

[24] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[25] Doug Tischer, Sidney Lisanza, Jue Wang, Runze Dong, Ivan Anishchenko, Lukas F Milles, Sergey Ovchinnikov, and David Baker. Design of proteins presenting discontinuous functional sites using deep learning. *bioRxiv*, 2020.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[27] Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Ivan Anishchenko, Minkyung Baek, Joseph L Watson, Jung Ho Chun, Lukas F Milles, Justas Dauparas, Marc Exposit, Wei Yang, Amijai Saragovi, Sergey Ovchinnikov, and David A. Baker. Deep learning methods for designing proteins scaffolding functional sites. *bioRxiv*, 2021.

[28] Jinrui Xu and Yang Zhang. How significant is a protein structure similarity with tm-score= 0.5? *Bioinformatics*, 26(7):889–895, 2010.

[29] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation. In *International Conference on Learning Representations*, 2022.

[30] Yang Zhang and Jeffrey Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–2309, 2005.

[31] Linqi Zhou, Yilun Du, and Jiajun Wu. 3D shape generation and completion through point-voxel diffusion. In *International Conference on Computer Vision*, pages 5826–5835, 2021.

Appendix

Table of Contents

A Method details

A.1 Diffusion probabilistic models

Our approach to the motif-scaffolding problem builds on denoising diffusion probabilistic models (DPMs) [23]. We follow the conventions and notation set by [11], which we review here. DPMs are a class of generative models based on a reversible, discrete-time diffusion process. The *forward process* starts with a sample $\mathbf{x}^{(0)}$ from an unknown data distribution q, with density denoted by $q(\mathbf{x}^{(0)})$, and iteratively adds noise at each step t. By the last step, T, the distribution of $\mathbf{x}^{(T)}$ is indistinguishable from an isotropic Gaussian: $\mathbf{x}^{(T)} \sim \mathcal{N}(\mathbf{x}^{(T)}; 0, \mathbf{I})$. Specifically, we choose a variance schedule $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(T)}$, and define the transition distribution at step t as $q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}) = \mathcal{N}(\mathbf{x}^{(t)}; \sqrt{1-\beta^{(t)}}\mathbf{x}^{(t-1)}, \beta^{(t)}\mathbf{I})$.

DPMs approximate q with a second distribution p_θ by learning the transition distribution of the *reverse process* at each t, $p_\theta(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)})$. We follow the conventions set by [11] in our parameterization and choice of objective. In particular, we take $p_\theta(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)}) = \mathcal{N}(\mathbf{x}^{(t-1)}; \mu_\theta(\mathbf{x}^{(t)}, t), \beta^{(t)}\mathbf{I})$ with $\mu_\theta(\mathbf{x}^{(t)}, t) = \frac{1}{\sqrt{\alpha^{(t)}}}\left(\mathbf{x}^{(t)} - \frac{\beta^{(t)}}{\sqrt{1-\bar{\alpha}^{(t)}}}\epsilon_\theta(\mathbf{x}^{(t)}, t)\right)$, $\alpha^{(t)} := 1 - \beta^{(t)}$, and $\bar{\alpha}^{(t)} := \prod_{s=1}^{t} \alpha^{(t)}$. We implement $\epsilon_\theta(\mathbf{x}^{(t)}, t)$ as a neural network. For training, we marginally sample $\mathbf{x}^{(t)} \sim q(\mathbf{x}^{(t)} \mid \mathbf{x}^{(0)})$ from the forward process as $\mathbf{x}^{(t)} = \sqrt{\bar{\alpha}^{(t)}}\mathbf{x}^{(0)} + \sqrt{1-\bar{\alpha}^{(t)}}\epsilon$ and minimize the objective $T^{-1}\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}^{(0)}, \mathbf{x}^{(t)})}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}^{(t)}, t)\|^2\right]$ by stochastic optimization [11, Algorithm 1]. To generate samples from $p_\theta(\mathbf{x}^{(0)})$, we simulate the reverse process. That is, we sample noise for time T as $\mathbf{x}^{(T)} \sim \mathcal{N}(0, \mathbf{I})$, and then for each $t = T-1, \ldots, 0$, we simulate progressively "de-noised" samples as $\mathbf{x}^{(t)} \sim p_\theta(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t+1)})$.

A.2 Additional `ProtDiff` details

We perform a scaling operation that transforms the input protein coordinates to be in nanometers rather than Angstroms by dividing by 10. Because typical proteins are on the order of 2–10 nanometers in diameter, this scaling brings the backbones to a spatial scale more similar to an isotropic Gaussian with unit variance.

We now describe our choice of node and edge features. Our choice is motivated by the linear chain structure of protein backbones; residues close in sequence are necessarily close in 3D space. To allow this chain constraint to be learned more easily, we fix an ordering of nodes in the graph to correspond to sequence order. We include as edge features positional offsets, which we represent using sinusoidal positional encoding features [26] as

$$a_{nn'} = \begin{bmatrix} \varphi(n-n',1) \\ \vdots \\ \varphi(n-n',D) \end{bmatrix}, \text{ where } \varphi(x,k) = \begin{cases} \sin\left(x \cdot \pi / N^{2 \cdot k/D}\right), & k \mod 2 = 0 \\ \cos\left(x \cdot \pi / N^{2 \cdot (k-1)/D}\right), & k \mod 2 = 1. \end{cases}$$

For node features, we similarly use a sinusoidal encoding of sequence position as well as of the diffusion time step t (following [17]) as

$$h_n(t) = \begin{bmatrix} \varphi(n,1) \\ \vdots \\ \varphi(n,D) \end{bmatrix} + R \begin{bmatrix} \varphi(t,1) \\ \vdots \\ \varphi(t,D) \end{bmatrix},$$

where R is a $D \times D$ orthogonal matrix chosen uniformly at random. Intuitively, applying R transforms the time encoding to be orthogonal to the positional encoding.

A.3 `SMCDiff` algorithm

Algorithm 1 `SMCDiff`: Particle filtering for conditionally sampling from unconditional diffusion models

```
1: Input: $\mathbf{x}_{\mathcal{M}}^{(0)}$ (motif), $K$ (# particles)
2: // Forward diffuse motif
3: $\breve{\mathbf{x}}_{\mathcal{M}}^{(1:T)} \sim q(\mathbf{x}_{\mathcal{M}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)})$
4:
5: // Reverse diffuse particles
6: $\forall k,\ \mathbf{x}_k^{(T)} \overset{i.i.d.}{\sim} p_\theta(\mathbf{x}^{(T)})$
7: for $t = T, \ldots, 1$ do
8:     // Replace motif
9:     $\forall k,\ \mathbf{x}_k^{(t)} \leftarrow [\breve{\mathbf{x}}_{\mathcal{M}}^{(t)}, \mathbf{x}_{\mathcal{S},k}^{(t)}]$
10:
11:    // Re-weight based on$\breve{\mathbf{x}}_{\mathcal{M}}^{(t-1)}$
12:    $\forall k,\ w_k^{(t)} \leftarrow p_\theta(\breve{\mathbf{x}}_{\mathcal{M}}^{(t-1)} \mid \mathbf{x}_k^{(t)})$
13:    $\forall k,\ \tilde{w}_k^{(t)} \leftarrow w_k^{(t)} / \sum_{k'=1}^{K} w_{k'}^{(t)}$
14:    $\tilde{\mathbf{x}}_{1:K}^{(t)} \sim \mathtt{Resample}(\tilde{w}_{1:K}^{(t)}, \mathbf{x}_{1:K}^{(t)})$
15:
16:    // Propose next step
17:    $\forall k,\ \mathbf{x}_k^{(t-1)} \overset{indep.}{\sim} p_\theta(\mathbf{x}^{(t-1)} \mid \tilde{\mathbf{x}}_k^{(t)})$
18: end for
19: Return $\mathbf{x}_{\mathcal{S},1:K}^{(0)}$
```

B Conditional sampling: `SMCDiff` details and supplementary proofs

We here provide additional details related to `SMCDiff` and the replacement method described in Section 2.2. Details of the replacement method [24] and our analysis of its error are in Appendix B.1.

Algorithm 2 Replacement method for approximate conditional sampling

```
1: Input: $\mathbf{x}_{\mathcal{M}}^{(0)}$ (motif)
2: // Forward diffuse motif
3: $\breve{\mathbf{x}}_{\mathcal{M}}^{(1:T)} \sim q(\mathbf{x}_{\mathcal{M}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)})$
4:
5: // Reverse diffuse scaffold
6: $\mathbf{x}^{(T)} \sim p_\theta(\mathbf{x}^{(T)})$
7: for $t = T, \dots, 1$ do
8:     // Replace with forward diffused motif
9:     $\mathbf{x}^{(t)} \leftarrow [\breve{\mathbf{x}}_{\mathcal{M}}^{(t)}, \mathbf{x}_{\mathcal{S}}^{(t)}]$
10:
11:     // Propose next step
12:     $\mathbf{x}^{(t-1)} \sim p_\theta(\mathbf{x}^{(t-1)} \mid \mathbf{x}^{(t)})$
13: end for
14: Return $\mathbf{x}_{\mathcal{S}}^{(0)}$, $\mathbf{x}^{(1:T)}$
```

Appendix B.2 provides details of our sampling method, `SMCDiff`, including (1) a proof of Proposition 2.1 and (2) details of the residual resampling step. We leave technical proofs and lemmas to Appendix B.3.

Notation. In the following, we require notation that is more precise than in previous sections. For each $t = 0, \dots, T$, we let $q_t(\cdot)$ and $p_t(\cdot)$ denote the density functions of $\mathbf{x}^{(t)}$ according to the forward process and to our neural network approximation of the reverse process, respectively. We denote densities restricted to the motif and scaffold with subscripts $\mathcal{M}$ and $\mathcal{S}$. For example, we here write $p_{\mathcal{M},t}(\mathbf{x}_{\mathcal{M}}^{(t)})$, whereas we wrote $p_\theta(\mathbf{x}_{\mathcal{M}}^{(t)})$ in the main text. We write (random) conditional densities as $q_{\mathcal{M},t}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(t-1)})$ and write the (deterministic) conditional density for an observation $\mathbf{x}_{\mathcal{M}}^{(t-1)} = x_{\mathcal{M}}$ as $q_{\mathcal{M},t}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(t-1)} = x_{\mathcal{M}})$.

An object of interest will be the Kullback-Leibler (KL) divergence. We write $\mathrm{KL}\left[q_t(\cdot) \| p_t(\cdot)\right] := \int q_t(x) \log \frac{q_t(x)}{p_t(x)} dx$, where $\log(\cdot)$ is the natural (base e) logarithm. We will also encounter the expected KL between conditional densities, which we will write as $\mathrm{EKL}\left[q_t(\cdot \mid \mathbf{x}^{(t-1)}) \| p_t(\cdot \mid \mathbf{x}^{(t-1)})\right] := \int q_{t-1}(x) \mathrm{KL}\left[q_t(\cdot \mid \mathbf{x}^{(t-1)} = x) \| p_t(\cdot \mid \mathbf{x}^{(t-1)} = x)\right] dx$, where the outer expectation is taken with respect to the unconditional density associated with first argument of $\mathrm{EKL}\left[\cdot \| \cdot\right]$.

B.1 The replacement method and its error

The replacement method was proposed by [24] for the task of inpainting in the context of score-based generative models. Work [12] concurrent with the present paper applied the replacement method to DPMs. Although [24] notes that this approach can be understood as *approximate* conditional sampling, they provide no discussion of approximation error. We here show that the replacement method introduces irreducible error that is inherent to the forward process. Algorithm 2 provides an explicit description of the replacement method.

The first return of Algorithm 2, $\mathbf{x}_{\mathcal{S}}^{(0)}$, is used as a putative inpainting solution or approximate conditional sample. But Algorithm 2 additionally returns subsequent time steps, $\mathbf{x}^{(1:T)}$. We denote the approximation over all steps implied by the generative procedure in Algorithm 2 by $p_{1:T}^{\mathrm{Repl}}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})$ and compare it to the exact conditional, $q_{1:T}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})$. We here consider error in KL divergence because it permits an analytically tractable and transparent analysis. We additionally consider the idealized scenario where $p_{0:T}(\cdot)$ perfectly captures the reverse process. Under this condition, the forward KL takes a surprisingly simple form.

Proposition B.1. *Suppose that $p_{0:T}(\cdot)$ exactly matches the forward diffusion process such that for every x, $p_t(\cdot \mid \mathbf{x}^{(t+1)} = x) = q_t(\cdot \mid \mathbf{x}^{(t+1)} = x)$. Then for any motif $x_{\mathcal{M}}$,*

$$\begin{aligned} &\mathrm{KL}\left[q_{1:T}(\cdot \mid \mathbf{x}^{(0)}_{\mathcal{M}} = x_{\mathcal{M}}) \| p^{\mathrm{Repl}}_{1:T}(\cdot \mid \mathbf{x}^{(0)}_{\mathcal{M}} = x_{\mathcal{M}})\right] \\ &= \sum_{t=1}^{T-1} \mathrm{EKL}\left[q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t+1)}, \mathbf{x}^{(0)}_{\mathcal{M}} = x_{\mathcal{M}}) \| q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t+1)})\right]. \end{aligned} \tag{2}$$

Proposition B.1 reveals that the replacement method introduces approximation error that is intrinsic to the forward process and cannot be eliminated by making $p_{0:T}(\cdot)$ more expressive. Although the individual terms in the right hand side of Equation (2) are not analytically tractable in general, in the following corollary we show that this approximation error can be non-trivial by considering a special case. For this following example, we depart from the earlier assumption that $\mathbf{x}$ is in 3D, and consider scalar valued $\mathbf{x}_M$ and $\mathbf{x}_{\mathcal{S}}$.

Corollary B.2. *Suppose $[\mathbf{x}^{(0)}_{\mathcal{M}}, \mathbf{x}^{(0)}_{\mathcal{S}}]$ is bivariate normal distributed with mean zero, unit variance, and covariance ρ. Further suppose that $q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(0)}_{\mathcal{S}}) = \mathcal{N}(\cdot; \sqrt{\bar{\alpha}^{(t)}}\mathbf{x}^{(0)}_{\mathcal{S}}, 1 - \bar{\alpha}^{(t)})$ and $q_{\mathcal{S},t+1}(\cdot \mid \mathbf{x}^{(t)}_{\mathcal{S}}) = \mathcal{N}(\cdot; \sqrt{1-\beta^{(t+1)}}\mathbf{x}^{(t)}_{\mathcal{S}}, \beta^{(t+1)})$ as in Section 2, where $\beta^{(t+1)}$ and $\bar{\alpha}^{(t)}$ are between 0 and 1. Then*

$$\mathrm{EKL}\left[q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t+1)}_{\mathcal{S}}, \mathbf{x}^{(0)}_{\mathcal{M}}) \| q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t+1)}_{\mathcal{S}})\right] \geq -\frac{1}{2}\left(\log(1 - \beta^{(t+1)}\bar{\alpha}^{(t)}\rho^2) + \beta^{(t+1)}\bar{\alpha}^{(t)}\rho^2\right).$$

We note two takeaways of Corollary B.2. First, as we might intuitively expect, this error can be large when significant correlation in the target distribution is present. Second, we see that the approximation error can be larger at earlier time steps, when $\bar{\alpha}^{(t)}$ is closer to 1.

B.2 `SMCDiff` details and verification proof of Proposition 2.1

The idea behind the `SMCDiff` procedure in Algorithm 1 is to break sampling of $\mathbf{x}^{(0)}_{\mathcal{S}} \sim q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0)}_{\mathcal{M}})$ into three stages:

1. Draw $\mathbf{x}^{(1:T)}_{\mathcal{M}} \sim q_{\mathcal{M},1:T}(\cdot \mid \mathbf{x}^{(0)}_{\mathcal{M}})$.
2. Draw $\mathbf{x}^{(1:T)}_{\mathcal{S}} \sim q_{\mathcal{S},1:T}(\cdot \mid \mathbf{x}^{(0:T)}_{\mathcal{M}})$.
3. Draw $\mathbf{x}^{(0)}_{\mathcal{S}} \sim q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:T)}_{\mathcal{M}}, \mathbf{x}^{(1:T)}_{\mathcal{S}})$

If all three steps were performed exactly, by the law of total probability $\mathbf{x}^{(0)}_{\mathcal{S}}$ in step (3) would (marginally) be an exact sample from $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0)}_{\mathcal{M}})$. As such, `SMCDiff` aims to perform step (1) and approximate steps (2) and (3). Step (1) corresponds to forward diffusing the motif in lines 2–3 and is exact because we diffuse according to q.

Step (3) corresponds to line 17 in the last iteration (when $t = 1$). Specifically, to sample from $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:T)}_{\mathcal{M}}, \mathbf{x}^{(1:T)}_{\mathcal{S}})$ we make three observations. (i) The Markov structure of the forward process implies that $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:T)}_{\mathcal{M}}, \mathbf{x}^{(1:T)}_{\mathcal{S}}) = q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:1)}_{\mathcal{M}}, \mathbf{x}^{(1)}_{\mathcal{S}})$. (ii) By the assumption that the forward and approximated reverse process agree, we have $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:1)}_{\mathcal{M}}, \mathbf{x}^{(1)}_{\mathcal{S}}) = p_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:1)}_{\mathcal{M}}, \mathbf{x}^{(1)}_{\mathcal{S}})$. (iii) Finally, because $p_t(\cdot \mid \mathbf{x}^{(t+1)})$ factorizes across $\mathcal{M}$ and $\mathcal{S}$ for each t, $p_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:1)}_{\mathcal{M}}, \mathbf{x}^{(1)}_{\mathcal{S}}) = p_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(1)}_{\mathcal{M}}, \mathbf{x}^{(1)}_{\mathcal{S}})$. As a result, under the assumptions of the proposition, we may sample from $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}^{(0:T)}_{\mathcal{M}}, \mathbf{x}^{(1:T)}_{\mathcal{S}})$, and perform step (3) exactly as well.

Step (2) is the only non-trivial step, and cannot be performed exactly. The challenge is that although the reverse process approximation, $p_{\mathcal{S},1:T}(\cdot \mid \mathbf{x}^{(0:T)}_{\mathcal{M}})$, is well-defined, computing it explicitly involves an intractable, high-dimensional integral.

The sequential Monte Carlo approach of `SMCDiff`, then, is to circumvent this intractability by constructing a sequence of approximations. For each $t = T, T-1, \ldots, 1$, we approximate $p_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t-1:T)}_{\mathcal{M}})$ (and thereby $q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t-1:T)})$) with K weighted atoms (the *particles*). We denote these

Algorithm 3 Residual Resample

```
1: Input: $w_{1:K}$ (weights), $\mathbf{x}_{1:K}$ (particles)
2: $\forall k,\ (c_k, r_k) \leftarrow (\lfloor Kw_k \rfloor, Kw_k - \lfloor Kw_k \rfloor)$
3: $\tilde{\mathbf{x}}_C = [\underbrace{\mathbf{x}_1, \ldots, \mathbf{x}_1}_{c_1}, \ldots, \underbrace{\mathbf{x}_K, \ldots, \mathbf{x}_K}_{c_K}]$
4: $R \leftarrow K - \sum_{k=1}^K c_k$
5: $[i_1, \ldots, i_R] \sim$ Multinomial$(r_{1:K}, R)$
6: $\tilde{\mathbf{x}}_R \leftarrow [\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_R}]$
7: $\tilde{\mathbf{x}} =$ concat$(\mathbf{x}_R, \mathbf{x}_C)$
8: Return $\tilde{\mathbf{x}}$
```

approximations (which are implicit in Algorithm 1) by $\mathbb{P}_K^{(t)}(\cdot) := \sum_{k=1}^K \tilde{w}_k^{(t)} \delta(\cdot; \mathbf{x}_{\mathcal{S},k}^{(t)})$, where each $\tilde{w}_k^{(t)}$ and $\mathbf{x}_{\mathcal{S},k}^{(t)}$ are as in Algorithm 1, and $\delta(\cdot; \mathbf{x})$ denotes a Dirac mass at $\mathbf{x}$. In particular, $\mathbb{P}_K^{(1)}(\cdot)$ is an approximation to $p_{\mathcal{S},1}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$. Proving the proposition amounts to showing that in the limit as K goes to infinity, each $\mathbb{P}_K^{(1)}(\cdot)$ converges weakly to $p_{\mathcal{S},1}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$, which by assumption is equal to $q_{\mathcal{S},1}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$. This weak convergence follows from standard asymptotics for particle filters [4, Proposition 11.4], which we make explicit in Lemma B.1. As a result, if we perform step (3) with $\mathbf{x}_{\mathcal{S}}^{(1)} \sim \mathbb{P}_K^{(1)}(\cdot)$, then this lemma implies that $\mathbf{x}_{\mathcal{S}}^{(0)}$ converges in distribution to $q_{\mathcal{S},0}(\mathbf{x}_{\mathcal{S}}^{(0)} \mid \mathbf{x}_{\mathcal{M}}^{(0)})$, since (i) $q_{\mathcal{S},0}(\mathbf{x}_{\mathcal{S}}^{(0)} \mid \mathbf{x}_{\mathcal{M}}^{(1)}, \mathbf{x}_{\mathcal{S}}^{(1)})$ is continuous in $\mathbf{x}_{\mathcal{S}}^{(1)}$ and (ii) $\mathbf{x}_{\mathcal{S}}^{(0)}$ is independent of $\mathbf{x}_{\mathcal{M}}^{(0)}$ conditional on $\mathbf{x}^{(1)}$.

Recall that to show the proposition, it was to sufficient to show that $\mathbb{P}_K^{(1)}$ converged weakly to $q_{\mathcal{S},1}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$; this implied that the K particle returned by Algorithm 1 would then converge in distribution to $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$ which, by the law of total probability, implied that they marginally converge to $q_{\mathcal{S},0}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0)})$. However, while the particles return by Algorithm 1 may be treated as exchangeable, they are not independent, because they depend on shared randomness in $\mathbf{x}_{\mathcal{M}}^{(1:T)}$. To obtain approximate samples that are independent, it is necessary to run Algorithm 1 multiple times.

Residual resampling. Line 14 of Algorithm 1 indicates a `Resample` step. In particle filtering, resampling steps (or *branching mechanisms* [8, Chapter 2]) filter out particles with very small weights, and replace them with additional copies of particles with large weights. Notably, the resampling step is the only point of departure of Algorithm 1 from the replacement method; without resampling, the algorithms behave identically. While a variety of possible branching mechanisms exist, we use *residual resampling* (Algorithm 3) in our implementation for its simplicity.

B.3 Proofs and lemmas

Particle filtering lemma with technical conditions

Lemma B.1. *Consider $\mathbb{P}_K^{(1)} := \sum_{k=1}^K \tilde{w}_k \delta(\cdot; \mathbf{x}_{\mathcal{S},k}^{(1)})$, where $\tilde{w}_k$ and $\mathbf{x}_{\mathcal{S},1:K}^{(1)}$ are as constructed in Algorithm 1. Assume the conditions of Proposition 2.1. Then $\mathbb{P}_K^{(1)}$ converges weakly to $p_{\mathcal{S},1}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$ as K goes to infinity. That is, for any Borel measurable A, $\lim_{K\to\infty} \mathbb{P}_K^{(1)}(A) = \int_A p_{\mathcal{S},1}(x \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})dx$.*

Proof. The proof of the lemma follows from an application of standard asymptotics for particle filtering [4, Proposition 11.4]. In particular, to apply Proposition 11.4 we use the formalism of Feynman–Kac (FK) models, following the notation of [4, Chapter 5]. Though typically (and in [4]) FK models are defined via a sequence of approximations at increasing time steps, we consider decreasing time steps because we are approximating the reverse time process. We take the initial distribution as $\mathbb{M}_T(\mathbf{x}_{\mathcal{S}}^{(T)}) = p_{\mathcal{S},T}(\mathbf{x}_{\mathcal{S}}^{(T)})$, the transition kernel as $M_t(\mathbf{x}_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{S}}^{(t)}) = p_{\mathcal{S},t}(\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}^{(t+1)})$, and the potential functions as $G_t(\mathbf{x}_{\mathcal{S}}^{(t)}) = p_{\mathcal{M},t-1}(\mathbf{x}_{\mathcal{M}}^{(t-1)} \mid \mathbf{x}^{(t)})$. The sequence of FK models, $\mathbb{Q}_t$, then correspond to

$$\mathbb{Q}_t(\mathbf{x}_{\mathcal{S}}^{(t:T)}) = L_t^{-1}\mathbb{M}_T(\mathbf{x}_{\mathcal{S}}^{(T)})G_T(\mathbf{x}_{\mathcal{S}}^{(T)})\prod_{i=T-1}^{t} M_i(\mathbf{x}_{\mathcal{S}}^{(i+1)}, \mathbf{x}_{\mathcal{S}}^{(i)})G_i(\mathbf{x}_{\mathcal{S}}^{(i)})$$

for each t, where L_t is a normalizing constant.

By substituting in our choices of M_t and G_t, we can rewrite and simplify $\mathbb{Q}_t$ as

$$\begin{aligned}
\mathbb{Q}_t(\mathbf{x}_{\mathcal{S}}^{(t:T)}) &= L_t^{-1}p_{\mathcal{S},T}(\mathbf{x}_{\mathcal{S}}^{(T)})p_{\mathcal{M},T-1}(\mathbf{x}_{\mathcal{M}}^{(T-1)} \mid \mathbf{x}^{(T)})\prod_{i=T-1}^{t} p_{\mathcal{S},i}(\mathbf{x}_{\mathcal{S}}^{(i)} \mid \mathbf{x}^{(i+1)})p_{\mathcal{M},i-1}(\mathbf{x}_{\mathcal{M}}^{(i-1)} \mid \mathbf{x}^{(i)}) \\
&= L_t^{-1}p_{\mathcal{S},T}(\mathbf{x}_{\mathcal{S}}^{(T)})p_{t:T-1}(\mathbf{x}^{(t:T-1)} \mid \mathbf{x}^{(T)})p_{\mathcal{M},t-1}(\mathbf{x}_{\mathcal{M}}^{(t-1)} \mid \mathbf{x}^{(t)}) \\
&\propto p_{t:T}(\mathbf{x}^{(t:T)} \mid \mathbf{x}_{\mathcal{M}}^{(t-1)}) \\
&\propto p_{\mathcal{S},t:T}(\mathbf{x}_{\mathcal{S}}^{(t:T)} \mid \mathbf{x}_{\mathcal{M}}^{(t-1:T)}),
\end{aligned}$$

where lines 3 and 4 drop multiplicative constants that do not depend on $\mathbf{x}_{\mathcal{S}}^{(t:T)}$. From the above derivation, we see that each $\mathbb{Q}_t(\mathbf{x}_{\mathcal{S}}^{(t)}) = p_{\mathcal{S},t}(\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{M}}^{(t-1:T)})$, and in particular that $\mathbb{Q}_1(\mathbf{x}_{\mathcal{S}}^{(1)}) = p_{\mathcal{S},1}(\mathbf{x}_{\mathcal{S}}^{(1)} \mid \mathbf{x}_{\mathcal{M}}^{(0:T)})$. As such, the desired convergence in the statement of the lemma is equivalent to that $\mathbb{P}_K^{(1)}$ converges to $\mathbb{Q}_1$.

Chopin and Papaspiliopoulos [4, Proposition 11.4] provide this result for the generic particle filtering algorithm (see [4, Algorithm 10.1], which is written in the FK model form described above). More specifically, Proposition 11.4 proves almost sure convergence of all Borel measurable functions of $\mathbb{P}_K^{(t)}$, which implies the desired weak convergence.

Although the proof provided in [4] is restricted to the simpler, but higher variance, case where the resampling step uses multinomial resampling, the authors note that [3] proves it holds in the case of residual resampling (which we use in our experiments) as well. $\square$

Replacement method error — lemmas and proofs

We here provide proofs of Proposition B.1 and Corollary B.2.

Proof of Proposition B.1:

Proof. The result obtains from recognizing where the replacement method approximation agrees with the forward process, using conditional independences in both processes, and applying the chain rule for KL divergences. We make this explicit in the derivation below, with comments explaining the transition to the following line.

$$
\begin{aligned}
&\mathrm{KL}\left[q_{1:T}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}) \| p_{1:T}^{\mathrm{Repl}}(\cdot \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})\right] \\
&= \int q_{1:T}(x^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}) \log \frac{q_{1:T}(x^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})}{p_{1:T}^{\mathrm{Repl}}(x^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})} dx^{(1:T)}
\end{aligned}
$$

// By the chain rule of probability.

$$
\begin{aligned}
&= \int q_{1:T}(x^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}) \Big[\log \frac{q_{\mathcal{M},1:T}(x_{\mathcal{M}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})}{p_{1:T}^{\mathrm{Repl}}(x_{\mathcal{M}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}})} + \\
&\qquad \log \frac{q_{\mathcal{S},1:T}(x_{\mathcal{S}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0:T)} = x_{\mathcal{M}}^{(0:T)})}{p_{\mathcal{S},1:T}^{\mathrm{Repl}}(x_{\mathcal{S}}^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0:T)} = x_{\mathcal{M}}^{(0:T)})} \Big]
\end{aligned}
$$

// By the agreement of q and p_{Repl} on the motif, and the chain rule of probability.

$$
\begin{aligned}
&= \int q_{1:T}(x^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}) \Big[\log \frac{q_{\mathcal{S},T}(x_{\mathcal{S}}^{(T)} \mid \mathbf{x}_{\mathcal{M}}^{(0:T)} = x_{\mathcal{M}}^{(0:T)})}{p_{T}^{\mathrm{Repl}}(x_{\mathcal{S}}^{(T)} \mid \mathbf{x}_{\mathcal{M}}^{(0:T)} = x_{\mathcal{M}}^{(0:T)})} + \\
&\quad \sum_{t=1}^{T-1} \log \frac{q_{\mathcal{S},t}(x_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)} = x_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0:T)} = x_{\mathcal{M}}^{(0:T)})}{p_{\mathcal{S},t}^{\mathrm{Repl}}(x_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)} = x_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0:T)} = x_{\mathcal{M}}^{(0:T)})} \Big] dx^{(1:T)}
\end{aligned}
$$

// Because $q_{\mathcal{S},T}(\cdot) = p_{\mathcal{S},T}^{\mathrm{Repl}}(\cdot) = \mathcal{N}(\cdot; 0, I)$ and the assumption that p_θ matches q.

$$
\begin{aligned}
&= \int q_{1:T}(x^{(1:T)} \mid \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}) \Big[\sum_{t=1}^{T-1} \log \frac{q_{\mathcal{S},t}(x_{\mathcal{S}}^{(t)} \mid \mathbf{x}^{(t+1)} = x^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}^{(0)})}{q_{\mathcal{S},t}(x_{\mathcal{S}}^{(t)} \mid \mathbf{x}^{(t+1)} = x^{(t+1)})} \Big] dx^{(1:T)} \\
&= \sum_{t=1}^{T-1} \mathrm{EKL}\left[q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}^{(0)}) \| q_{\mathcal{S},t}(\cdot \mid \mathbf{x}^{(t+1)}) \right].
\end{aligned}
$$

□

Proof of Corollary B.2:

The proof of the corollary relies of on a lemma on the variances of the two relevant conditional distributions. We state this lemma, whose proof is at the end of the section, before continuing. For notational simplicity, we drop the scripts and annotations on $\bar{\alpha}^{(t)}$ and $\beta^{(t+1)}$, and instead write α and β, respectively.

Lemma B.2. *Suppose* $\mathbf{x}_{\mathcal{M}}^{(0)}, \mathbf{x}_{\mathcal{S}}^{(t)}$*, and* $\mathbf{x}_{\mathcal{S}}^{(t+1)}$ *are distributed as in Corollary B.2. Then* $\mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}] = \beta$ *and* $\mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)}] \le \beta(1 - \beta\rho^2\alpha)$.

Now we provide a proof of Corollary B.2.

Proof. First recall that

$$
\begin{aligned}
\mathrm{KL}\left[\mathcal{N}(\mu_1, \sigma_1^2) \| \mathcal{N}(\mu_2, \sigma_2^2)\right] &= \frac{1}{2}\left(\log \frac{\sigma_2^2}{\sigma_1^2} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2} - 1\right) \\
&\ge \frac{1}{2}\left(\log \frac{\sigma_2^2}{\sigma_1^2} + \frac{\sigma_1^2}{\sigma_2^2} - 1\right)
\end{aligned}
$$

and observe that this lower bound is monotonically decreasing in σ_1^2 for $\sigma_1^2 \le \sigma_2^2$. Therefore

$$\begin{aligned}
&\mathrm{EKL}\left[q_{\mathcal{S},t}(\cdot \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)}) \| q_{\mathcal{S},t}(\cdot \mid \mathbf{x}_{\mathcal{S}}^{(t+1)})\right] \\
&= \int q_{\mathcal{M},0}(x_{\mathcal{M}}^{(0)}) q_{\mathcal{S},t+1}(x_{\mathcal{S}}^{(t+1)} \mid x_{\mathcal{M}}^{(0)}) \Big[\\
&\quad \mathrm{KL}\left[q_{\mathcal{S},t}(\cdot \mid \mathbf{x}_{\mathcal{S}}^{(t+1)} = x_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}^{(0)}]) \| q_{\mathcal{S},t}(\cdot \mid \mathbf{x}_{\mathcal{S}}^{(t+1)} = x_{\mathcal{S}}^{(t+1)}])\right] \\
&\Big] dx_{\mathcal{M}}^{(0)} x_{\mathcal{S}}^{(t+1)} \\
&\ge \int q_{\mathcal{M},0}(x_{\mathcal{M}}^{(0)}) q_{\mathcal{S},t+1}(x_{\mathcal{S}}^{(t+1)} \mid x_{\mathcal{M}}^{(0)}) \Big[\\
&\quad \mathrm{KL}\left[\mathcal{N}(0, \mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)} = x_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)} = x_{\mathcal{M}}^{(0)}]) \| \mathcal{N}(0, \mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)} = x_{\mathcal{S}}^{(t+1)}])\right] \\
&\Big] dx_{\mathcal{M}}^{(0)} x_{\mathcal{S}}^{(t+1)} \\
&\ge \mathrm{KL}\left[\mathcal{N}(0, \beta(1-\beta\rho^2\alpha)) \| \mathcal{N}(0,\beta)\right] \\
&\ge \frac{1}{2}\left(\log \frac{\beta}{\beta(1-\beta\rho^2\alpha)} + \frac{\beta(1-\beta\rho^2\alpha)}{\beta} - 1\right) \\
&= -\frac{1}{2}\left(\log(1-\beta\rho^2\alpha) + \beta\rho^2\alpha\right)
\end{aligned}$$

where the second inequality follows from Lemma B.2, and the monotonicity of the KL in σ_1^2. □

Proof of Lemma B.2:

Proof. That $\mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}] = \beta$ follows immediately from that $[\mathbf{x}_{\mathcal{S}}^{(t)}, \mathbf{x}_{\mathcal{S}}^{(t+1)}]$ is marginally bivariate normal distributed with covariance $\sqrt{1-\beta}$.

The upper bound on $\mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)}]$ is trickier. Observe that $[\mathbf{x}_{\mathcal{S}}^{(t)}, \mathbf{x}_{\mathcal{S}}^{(t+1)}] \mid \mathbf{x}_{\mathcal{M}}^{(0)}$ is bivariate Gaussian and that

$$\mathrm{Var}\left[\begin{bmatrix}\mathbf{x}_{\mathcal{S}}^{(t)} \\ \mathbf{x}_{\mathcal{S}}^{(t+1)}\end{bmatrix} \mid \mathbf{x}_{\mathcal{M}}^{(0)}\right] = \begin{bmatrix} 1-\rho^2\alpha & \sqrt{1-\beta}(1-\rho^2\alpha) \\ \sqrt{1-\beta}(1-\rho^2\alpha) & 1+\beta\rho^2\alpha-\rho^2\alpha \end{bmatrix}.$$

As such, the conditional variance may be computed in closed form as $\mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)}] = \beta(1-\rho^2\alpha)+(1-\beta)(1-\rho^2\alpha)\left(1-(1-\rho^2\alpha)/(1-\rho^2\alpha+\beta\rho^2\alpha)\right)$. But since $(1-\rho^2\alpha)/(1-\rho^2\alpha+\beta\rho^2\alpha) \ge 1-(\beta\rho^2\alpha)/(1-\rho^2\alpha)$ and therefore $1-(1-\rho^2\alpha)/(1-\rho^2\alpha+\beta\rho^2\alpha) \le (\beta\rho^2\alpha)/(1-\rho^2\alpha)$ we can write

$$\begin{aligned}
\mathrm{Var}[\mathbf{x}_{\mathcal{S}}^{(t)} \mid \mathbf{x}_{\mathcal{S}}^{(t+1)}, \mathbf{x}_{\mathcal{M}}^{(0)}] &= \beta(1-\rho^2\alpha) + (1-\beta)(1-\rho^2\alpha)\left(1 - \frac{1-\rho^2\alpha}{1-\rho^2\alpha+\beta\rho^2\alpha}\right) \\
&\le \beta(1-\rho^2\alpha) + (1-\beta)(1-\rho^2\alpha)\frac{\beta\rho^2\alpha}{1-\rho^2\alpha}) \\
&= \beta(1-\rho^2\alpha) + (1-\beta)\beta\rho^2\alpha \\
&= \beta(1-\beta\rho^2\alpha).
\end{aligned}$$

□

C Detecting chirality

Section 4 noted the limitation of `ProtDiff` that it can generate left-handed helices (which do not stably occur in natural proteins). Figure 3 presents two such examples. We additionally note that, as in Figure 3 Left, model samples can include multiple helices with differing chirality.

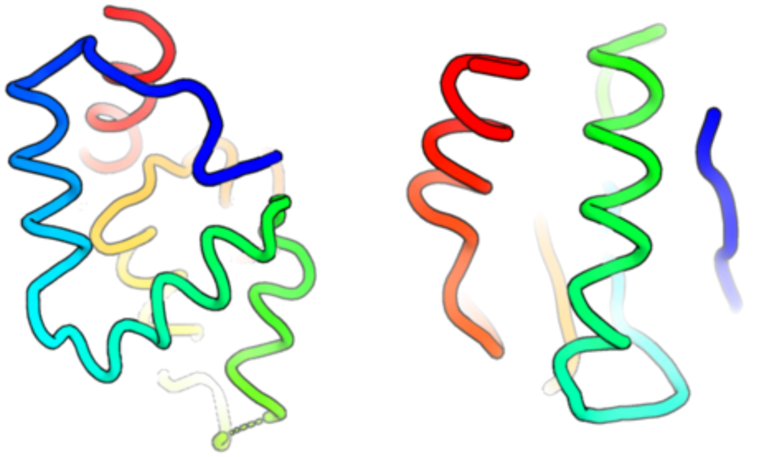
Figure 3: Two examples of protein backbone samples with incorrect left handed helices.

D Training details

`ProtDiff` uses 4 equivariant graph convolutional layers (EGCL) with 256 dimensions for node and edge embeddings. The training data was restricted to single chain proteins (monomers) found in PDB and lengths in the range [40, 128]. We additionally filtered out PDB with >5Å atomic resolution. This amounted to 4269 training examples. Training was performed using the Adam optimizer with hyperparameters `learning_rate=1e-4`, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We trained for 1,000,000 steps using batch size 16. We used a single Nvidia A100 GPU for approximately 24 hours. We implemented all models in PyTorch. We used the same linear noise schedule as Ho et al. [11] where $\beta_0 = 0.0001$, $\beta_T = 0.02$, and $T = 1024$. We did not perform hyperparameter tuning.

E Additional metric details

Self-consistency algorithm. Section 3.1 described our self-consistency metrics for evaluating the designability of backbones generated with `ProtDiff`. Algorithm 4 makes explicit the procedure we use for computing these metrics.

Algorithm 4 Self-consistency calculation

```
Input: $\mathbf{x} \in \mathbb{R}^{N,3}$
1: for $i \in 1, \dots, 8$ do
2:     $s_i \leftarrow$ ProteinMPNN($\mathbf{x}$)
3:     $\hat{\mathbf{x}}_i \leftarrow$ AF2($s_i$)
4: end for
5: sc_tm $\leftarrow \max_{i \in 1,\dots,8}$ TMscore($\hat{\mathbf{x}}_i, \mathbf{x}$)
Output: , sc_tm
```

Using dihedral angles to calculate helix chirality. Natural proteins are chiral molecules that contain only right-handed alpha helices. However, because the underlying EGNN in our model is equivariant to reflection, it can produce samples with left-handed helices. While examining model samples, we additionally observed samples with both left and right-handed helices (Figure 3), even though in theory the EGNN should be able to detect and avoid the chiral mismatch. Left-handed helices are fundamentally invalid geometries in proteins and represent a trivial failure mode when calculating the self-consistency and other metrics. Samples with a mixture of left and right-handed helices are especially problematic because they cannot be corrected simply by reflecting the coordinates. As such, it is important to identify and separate samples with mixed chirality.

To detect chiralty, we compute the dihedral angle between four consecutive C-α atoms as a chiral metric to distinguish between the two helix chiralities. Algorithmically, for every C-α `i`, we calculate the dihedral between C-α `i`, `i+1`, `i+2`, and `i+3`. C-α `i` with dihedral angles between 0.6 and 1.2 radians are classified as right-handed helices, and angles between -1.2 and -0.6 are classified as left-handed helices, with everything else classified as non-helical. Because C-α atoms in native helices tend to form contiguous stretches longer than one residue in the primary sequence, helical stretches less than one amino acid were removed. This filtering is meant to help avoid accidentally counting the occasional isolated backbone geometry that falls into a helical bin as a true helix. Finally, for all C-α atoms `i` that are still categorized as part of a helix, the associated `i+1`, `i+2` and `i+3` C-α atoms are also counted as part of that helix.

F Additional experimental results

In this section, we describe additional results to complement the main text. We provide a description of the motif targets in Section 2.2, along with results of a scaffolding failure case in Appendix F.1. We provide results when the model is tasked with *unconditional sampling* when no motif is provided (and which `ProtDiff` was originally trained on). To understand the qualitative outcomes of `scTM`, we present additional results of backbone designs, their AF2 prediction, and most closely related PDB parent chain for different thresholds of `scTM` in Appendix F.3. We provide additional examples of latent interpolations in Appendix F.4. Finally, Appendix F.5 presents a structural clustering of unconditional backbone samples; this result provides further evidence of `ProtDiff`'s ability to generate diverse backbone structures.

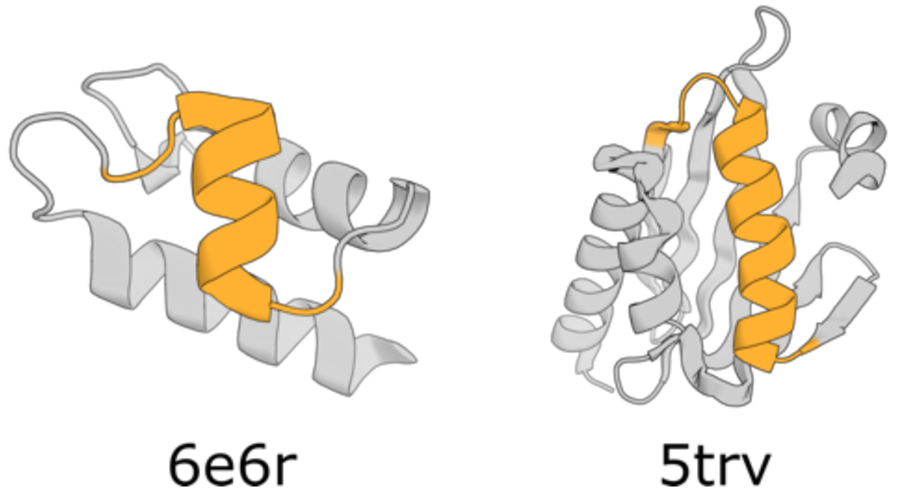

Figure 4: Structures used for motif-scaffolding test cases. Native structures (grey) and their motifs (orange) that were used for the motif-scaffolding task are shown.

F.1 Additional motif-scaffolding results

We here provide additional details of the motif-scaffolding experiments described in Section 3. Table 1 specifies the total lengths, motif sizes, and motif indices of our test cases. In Figure 4 we depict the structures of the native proteins (6e6r and 5trv) from which the motifs examined quantitatively in the main text were extracted. Figure 5 analyzes commonly observed failure modes of `ProtDiff` backbone samples involving chain breaks, steric clashes, and incorrect chirality.

Figure 6 presents quantitative results on a harder inpainting target. In this case, the motif is defined as residues 163–181 of chain A of respiratory syncytial virus (RSV) protein (PDB ID: 5tpn). We attempted to scaffold this motif into a 62 residue protein, with the motif as residues 42–62. We chose this placement because previous work [27] identified a promising candidate scaffold with this motif placement. In contrast to the cases described in the main text, for which a suitable scaffold exists in the training set, `SMCDiff` and the other inpainting methods failed to identify scaffolds that recapitulated this motif to within a motif RMSD of 1 Å.

Table 1: Motif-scaffolding test case additional details.

Origin/ Protein	Total length	Motif size (residue range)
6e6r	55	13 (23–35)
5trv	118	21 (42–62)
`RSV` (PDB-ID: 5tpn)	62	19 (16–34)

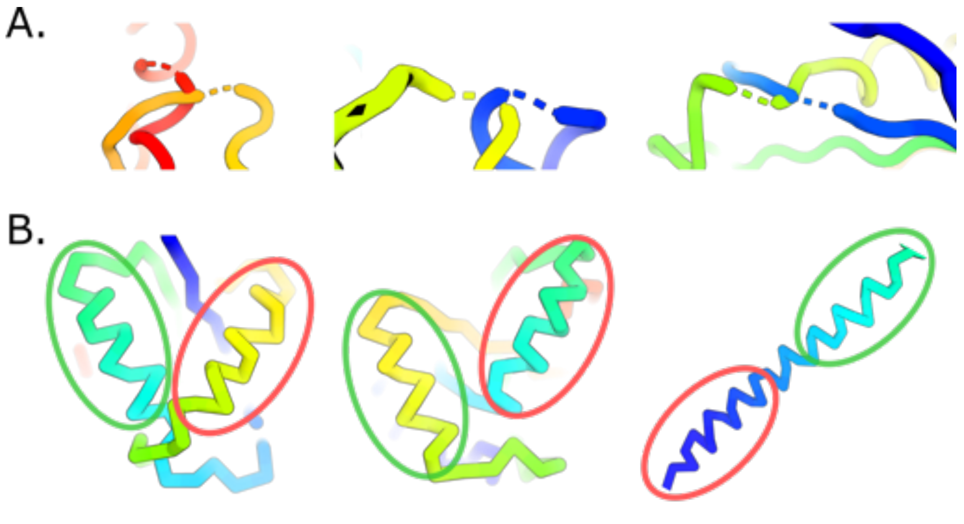

Figure 5: Failure modes in `ProtDiff` backbone samples. (A) Backbone clashes and chain breaks. The C-α atoms can be spaced further than the typical 3.8Å between neighbors, resulting in a chain break (dashed lines). Additionally, backbone segments can be too close to each other, resulting in obvious overlaps and clashes. (B) Backbones with a mixture of left (circled in red) and right (circled in green) handed helices. These chirality errors cannot be corrected simply by mirroring the sampled backbone.

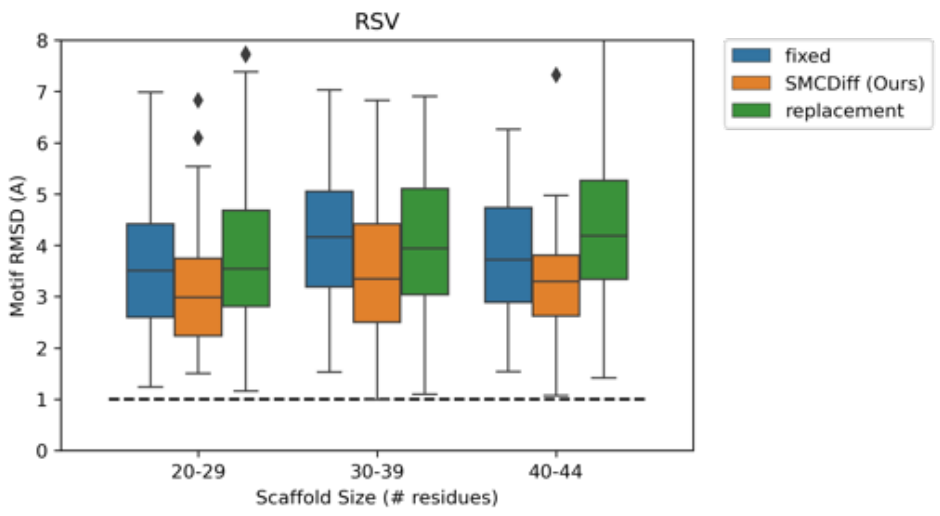

Figure 6: Additional inpainting results on a more challenging motif extracted from the respiratory syncytial virus (RSV). The three inpainting methods are evaluated as described in Section 3.

F.2 Unconditional sampling results

We next investigate the origins of the diversity seen in Figure 2 by analyzing the diversity and designability of `ProtDiff` samples without conditioning on a motif.

We first check that `ProtDiff` produces designable backbones. To do this, we generated 10 backbone samples for each length between 50 and 128 and then calculated `scTM` for each sample. In Fig. 7A, we find that 11.8% of samples have `scTM` > 0.5. However, the majority of backbones do not pass this threshold. We also observe designability has strong dependence on length since we expect that longer proteins are harder to model in 3D and design sequences for. We separated the lengths below 128 residues into two categories and refer to them as *short* (50–70) and *long* (70–128). Our results in Figure 7A indicate 17% of designs in the short category are designable vs. 9% in the long category. In Figure 12, we present a structural clustering of these designable backbones; we find that these backbones exhibit diverse topologes.

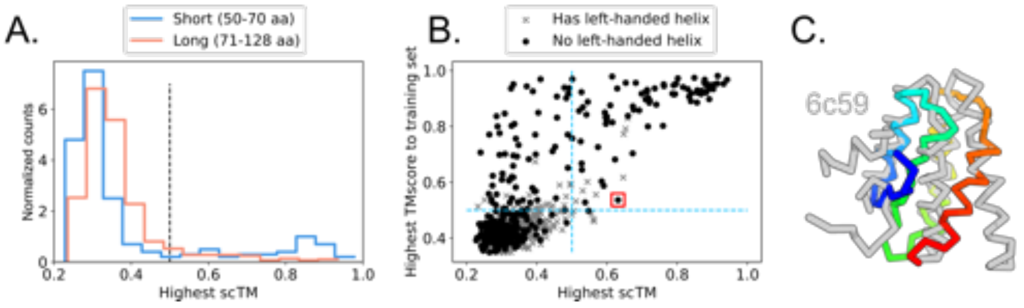

Figure 7: Protein backbone samples from `ProtDiff`. (A) Density plot of `scTM` for different length categories (50–70, 70–128). The dashed line at `scTM` = 0.5 indicates the threshold of “designability”, points to the right are considered “designable” (see text). (B) Scatter plot of `scTM` and the highest TM-score of each sample to all of PDB. Points represented as a grey “×” are detected to contain an (invalid) left-handed helix. Dashed lines indicate thresholds `scTM` = 0.5. (C) Example of a designable backbone sample (rainbow) with `scTM` > 0.5 (boxed in red in panel B) to its closest PDB example (6c59, grey) with a TM-score of 0.54.

We next sought to evaluate the ability of `ProtDiff` to generalize beyond the training set and produce novel backbones. In Figure 7B each point represents a backbone sample from `ProtDiff`. The horizontal coordinate of a point is the `scTM`, and the vertical coordinate is the minimum TM-score across the training set. We found a strong positive correlation between scTM and this minimum TM-score, indicating that many of the most designable backbones generated by ProtDiff were a result of training set memorization. However, if the model were only memorizing the training set, we would see TM-scores consistently near 1.0; the range of scores in Figure 7B indicate this is not the case – and the model is introducing a degree of variability. Figure 7C gives an example of backbone with scTM > 0.5 that appears to be novel. Its closest match in the PDB has TM-score = 0.54.

Fig. 7B illustrates a limitation of our method: many of our sampled backbones are not designable. One contributing factor is that ProtDiff does not handle chirality. Hence ProtDiff generates backbones with the wrong handedness, which cannot be realized by any sequence. Fig. 7B shows that 45% of all backbone samples had at least one incorrect, left-handed helix. Of these, most have scTM < 0.5. We describe calculating left-handed helices in Appendix E.

Fig. 8 illustrates an interpolation between two samples, showing how ProtDiff's outputs change as a function of the noise used to generate them. To generate these interpolations, we pick two backbone samples that result in different folds. For independent samples generated with noise $\epsilon^{(0:T)}$ and $\tilde{\epsilon}^{(0:T)}$ we interpolate with noise set to $\sqrt{\alpha}\epsilon^{(0:T)} + \sqrt{1-\alpha}\tilde{\epsilon}^{(0:T)}$ for α between 0 and 1. The depicted values of α are chosen to highlight transition points with full interpolations included in Appendix F.4. A future direction is to exploit the latent structure of ProtDiff to control backbone topology.

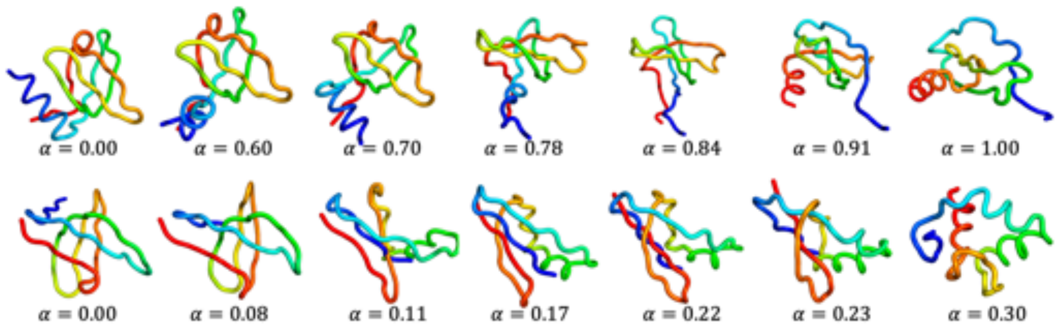

Figure 8: Interpolations between ProtDiff samples demonstrating the diversity of backbones captured. Top: 64-residue example. Bottom: 56-residue example. ProtDiff samples are determined by the Gaussian noise across all steps, $\epsilon^{(0:T)}$.

F.3 Qualitative analysis of scTM in different ranges

In this section, we give intuition for backbone designs and AF2 predictions associated with different values of scTM to aid the interpretation of the scTM results provided in Section 3. Figure 9 examines a possible categorization of scTM in three ranges. The first two rows correspond to backbone designs that achieve scTM > 0.9. We see the backbone designs in the first column closely match the AF2 prediction in the second column. A closely related PDB example can be found when doing a similarity search of the highest PDB chain with the highest TM-score to the AF2 prediction. We showed in Figure 7B that scTM > 0.9 is indicative of a close structural match being found in PDB.

The middle two rows correspond to designs that achieve scTM $\sim$ 0.5. These are examples of backbone designs on the edge of what we deemed as designable (scTM > 0.5). In these cases, the AF2 prediction shares the same coarse shape as the backbone design but possibly with different secondary-structure ordering and composition. In the length 69 example, we see the closest PDB chain has a TM-score of only 0.65 to the AF2 prediction but roughly the same secondary-structure ordering as the backbone design. The length 100 sample is a similar case of AF2 producing a roughly similar shape to the backbone design, but has no matching monomer in PDB.

The final category of scTM < 0.25 reflects failure cases when scTM is low. The AF2 predictions in this case have many disordered regions and bear little structural similarity with the original backbone design. Similar PDB chains are not found. We expect that improved generative models of protein backbones would not produce any samples in this category.

F.4 Additional latent interpolation results

We here provide additional latent interpolations. Figures 10 and 11 depict interpolations for between model samples for lengths 89 and 63, respectively.

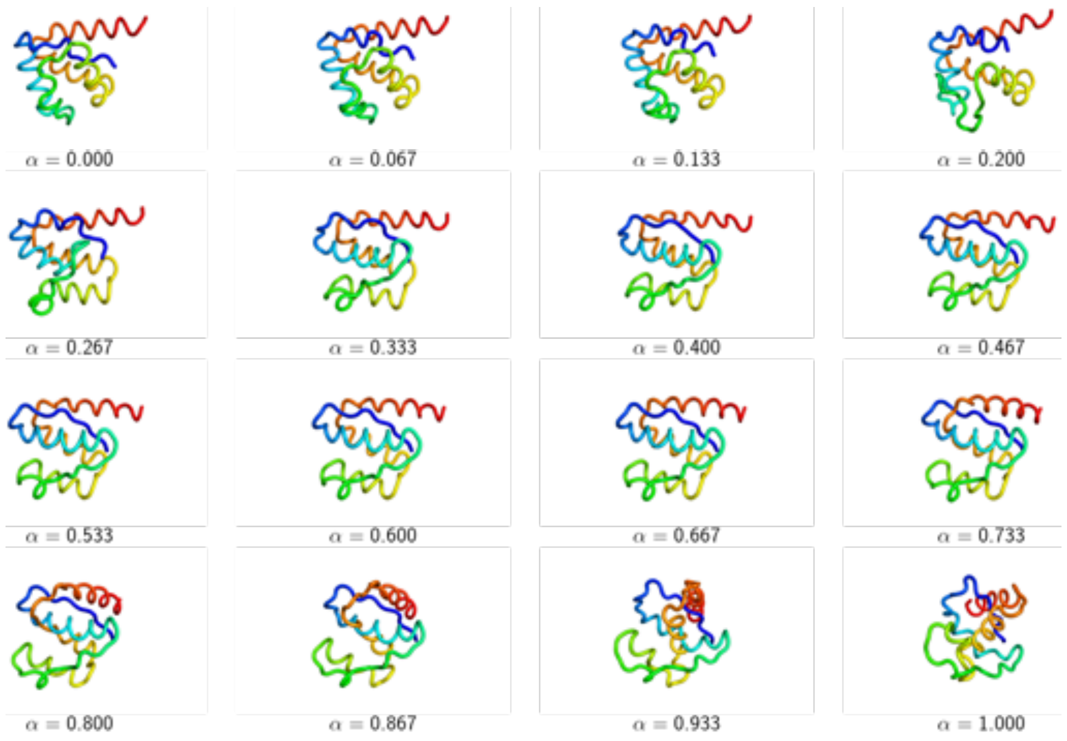

Figure 10: Latent interpolation of length 89 backbone sample from $\alpha = 0$ to 1.

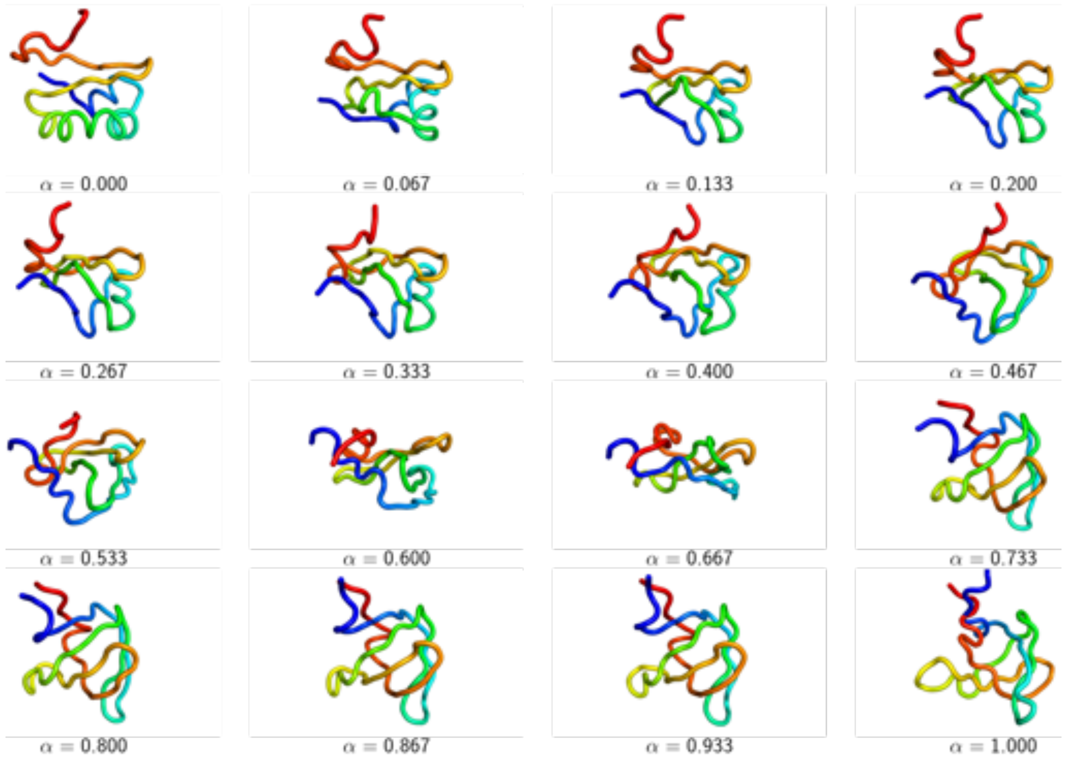

Figure 11: Latent interpolation of length 63 backbone sample from $\alpha = 0$ to 1.

F.5 Structural clustering

All 92 samples with `scTM` > 0.5 were compared and clustered using MaxCluster [10]. Structures were compared in a sequence independent manner, using the TM-score of the maximal subset of paired residues. They were subsequently clustered using hierarchical clustering with average linkage, 1 - TM-score as the distance metric and a TM-score threshold of 0.5 (Figure 12 A).

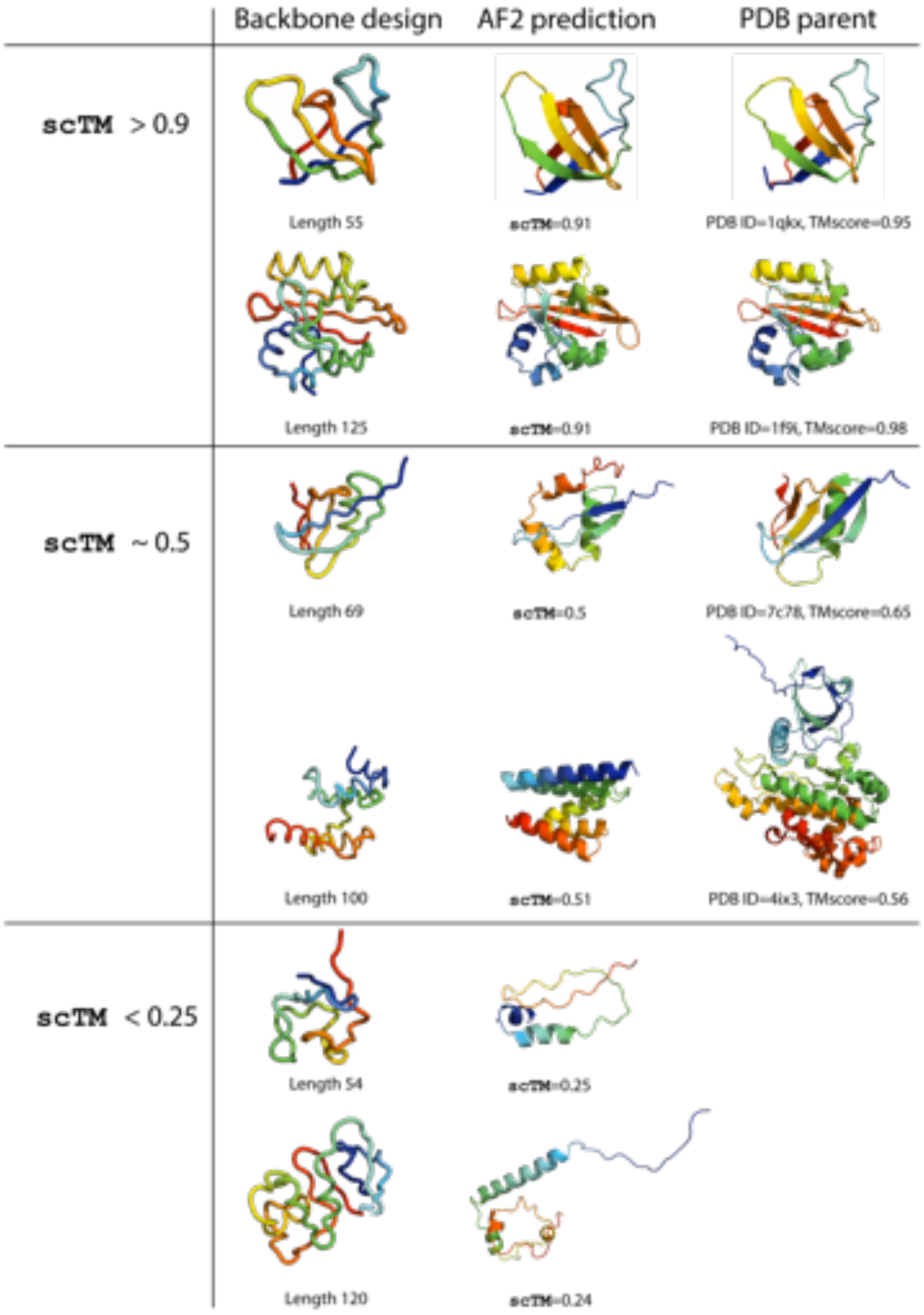

Figure 9: Qualitative analysis of unconditional backbone samples from ProtDiff. The first column displays backbone designs from ProtDiff and their sequence lengths. The second column displays the highest scTM scoring AF2 predictions from the ProteinMPNN sequences of the corresponding backbone design in the first column. The third column displays the closest PDB chain to the AF2 prediction in the second column with the PDB ID and TM-score written below. The third column is blank for the last two rows since no PDB match could be found. See Appendix F.3 for discussion.

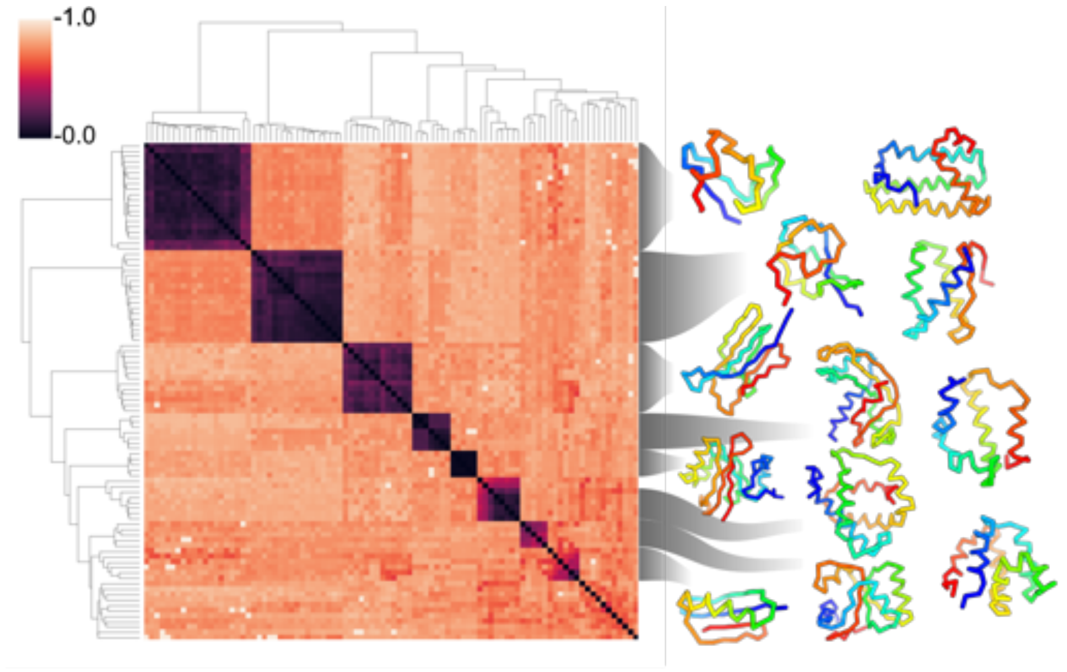

Figure 12: Clustering of self-consistent `ProtDiff` samples. The distance matrix is 1 - TM-score between pairs of samples, and ranges from 0 (exact matach) to 1 (no match). Dendrograms are from hierarchical clustering using the average distance metric. Designs on the right are cluster centroids. Gray lines connect larger clusters with more than one member to its centroid, while the remaining designs are from a random selection of the remaining single-sample clusters. Protein backbones are colored from blue at the N-terminus to red at the C-terminus.