All-atom protein design via SE(3) flow matching with ProteinZen

Alex J. Li
UC Berkeley - UCSF Joint Bioengineering PhD
`alexjli@berkeley.edu`

Tanja Kortemme
UCSF
`tanjakortemme@gmail.com`

Abstract

The advent of generative models of protein structure has greatly accelerated our ability to perform *de novo* protein design, especially concerning design at coarser physical scales such as backbone generation and protein binder design. However, the design of precise placements at atomic scales remains a challenge for existing design methods. One avenue towards higher fidelity atomic-scale design is via generative models with full atomic resolution, but is complicated by the intricacies of simultaneously designing both discrete protein sequence and continuous atomic positionings. In this work we propose a framework to capture this interplay by decomposing residues into collections of oriented rigid bodies, allowing us to apply SE(3) flow-matching for all-atom protein structure generation. Our method, ProteinZen[1], generates designs with high sequence-structure consistency while retaining competitive diversity and novelty on both unconditional and conditional generation tasks. We demonstrate competitive performance for unconditional monomer design and state-of-the-art performance on various forms of motif scaffolding, including full-atom motif scaffolding and motif scaffolding without specifying motif segment spacing or relative sequence order.

1 Introduction

De novo design has been greatly accelerated by generative modeling methods utilizing paradigms such as score-based diffusion [1] and flow matching [2], allowing us to design proteins with new structures *de novo* as well as simple functions such as protein-protein binders [3]. Despite these advances, targeting functions prevalent in nature, such as small molecule sensing or enzymatic catalysis, remains challenging. We posit this difficulty is in part due to the lack of atomic-level reasoning in existing generative models. The current dominant paradigm for protein design is backbone-centric: prominent models such as RFDiffusion [4] and Chroma [5] only model backbones, and extensions such as RFDiffusion2 [6] and LigandMPNN [7] introduce more atomic-awareness but still do not consider atomic structures in full detail. Moreover, pipelines utilizing these models often require generating 10^2-10^3 designs per suitable design candidate, of which only a fraction of candidates will ultimately have the function of interest after experimental validation. We know that atomic precision, both local and extended, is necessary for many design targets such as *de novo* enzymes [8][9] and allosteric regulation [10]. Hence, these shortcomings motivate transitioning towards all-atom generation, where all available atomic detail is considered and refined during the generation process.

The difficulty of transitioning from backbone generation to all-atom protein structures lies in the interdependencies between discrete and continuous aspects of protein structure: amino acid residues take discrete identities with discrete atomic compositions, but their associated atomic placements exist in continuous space. Hence, precise placements of atoms requires careful treatment of the interplay between these quantities. Various methods have proposed different residue representations,

[1]Code is available at `https://github.com/alexjli/proteinzen`.

including superposition-based [11] [12] [13], factorized [14], hybrid physical-latent [15] [16], and purely latent representations [17] [18]. However, all-atom methods still lag behind backbone-centric approaches on unconditional generation benchmarks, especially when it comes to co-designing compatible sequences for generated backbones. Furthermore, existing methods have much lower *in silico* success rates on conditional design tasks (such as motif scaffolding tasks) when compared to unconditional generation tasks, motivating further development.

Inspired by the intuition of viewing molecules as collections of functional groups, in this work we propose decomposing protein residues into collections of atoms that form oriented rigid bodies. Our decomposition allows us to represent residues with three oriented rigid bodies while retaining full expressivity of any canonical amino acid. We then apply the framework of SE(3) flow-matching [19][20] to train ProteinZen, our generative model of all-atom protein structure using this representation. We demonstrate competitive performance on unconditional monomer generation and state-of-the-art performance on various motif scaffolding tasks, helping pave forward the direction of all-atom protein generative models.

2 Methods

2.1 Setup

In the following sections we largely focus on a single residue $X^{(i)}$ at position i, with a protein being an ordered sequence of such residues. We will index fragments within residue i using local index j or globally using k (as a shorthand for $k = (i, j)$). Subscripts will be reserved for indexing along time/interpolation coordinate t. When clear we drop superscripts for simplicity, in which case it is implied that we are looking at a single entity (residue, residue fragment, etc.).

2.2 Representing all-atom detail using oriented rigid bodies

The residue representation in this work stems from the observation that every canonical protein residue can be described with at most three rigid chemical fragments (decomposition described in Appendix A). Hence, we can represent a residue as an ordered collection of three oriented rigid bodies $X = (S, [T^{(a)}, T^{(b)}, T^{(c)}])$, where $S \in \mathcal{A}$ is a residue identity in the twenty canonical residue identities $\mathcal{A}$ and $T = (r, x) \in \mathrm{SE}(3), r \in \mathrm{SO}(3), x \in \mathbb{R}^3$ is an oriented rigid body (also called a "frame"). We construct a mapping per residue identity between raw atomic coordinates and their frame-based representation (Appendix A, example in Figure 1A), which allows us to freely interconvert between the two representations. In this work, we train a generative model over rigid bodies and use this mapping to convert to and from cartesian atomic representations when necessary.

2.3 All-atom protein generation as SE(3) flow matching

We train our denoising model ProteinZen to minimize the SE(3) flow-matching objective over the collection of frames $\{T^{(k)}\}$. We follow FrameFlow [19] and FoldFlow [20] in flow-matching over SE(3) by training flows on SO(3) and $\mathbb{R}^3$. More concretely, we construct the following interpolants from source/noise distribution $T_0 = (r_0, x_0) \sim \rho_0$ (see Appendix B) to target distribution $T_1 = (r_1, x_1) \sim \rho_1$ (the distribution of all-atom protein structures) along interpolation coordinate t:

$$r_t = \exp_{r_0}(t \log_{r_0}(r_1)) \qquad x_t = tx_0 + (1-t)x_1 \qquad T_t = (r_t, x_t)$$

We formulate our flow-matching losses as denoising objectives and train a denoiser $\hat{r}_1, \hat{x}_1 = D_\phi(T_t, t)$ to minimize the following losses:

$$\mathcal{L}_{rot} = \frac{1}{(1-t)^2} \text{angle\_axis\_loss}(\log_{r_t}(\hat{r}_1(T_t, t)), \log_{r_t} r_1) \qquad \mathcal{L}_{trans} = \frac{1}{(1-t)^2} \|\hat{x}_1(T_t, t) - x_1\|^2$$

where $\mathcal{L}_{rot}$ is a separable analog to the original SO(3) denoising objective (see Appendix D for further details).

2.4 ProteinZen architecture

The architecture of ProteinZen is shown in Figure 1B. We build a two-track architecture which extends IPA Transformers [21] to perform multi-scale modeling and also modify layers to allow training at scale (details in Appendix E). The use of frames and IPA as the primary units of computation makes ProteinZen fully SE(3) equivariant.

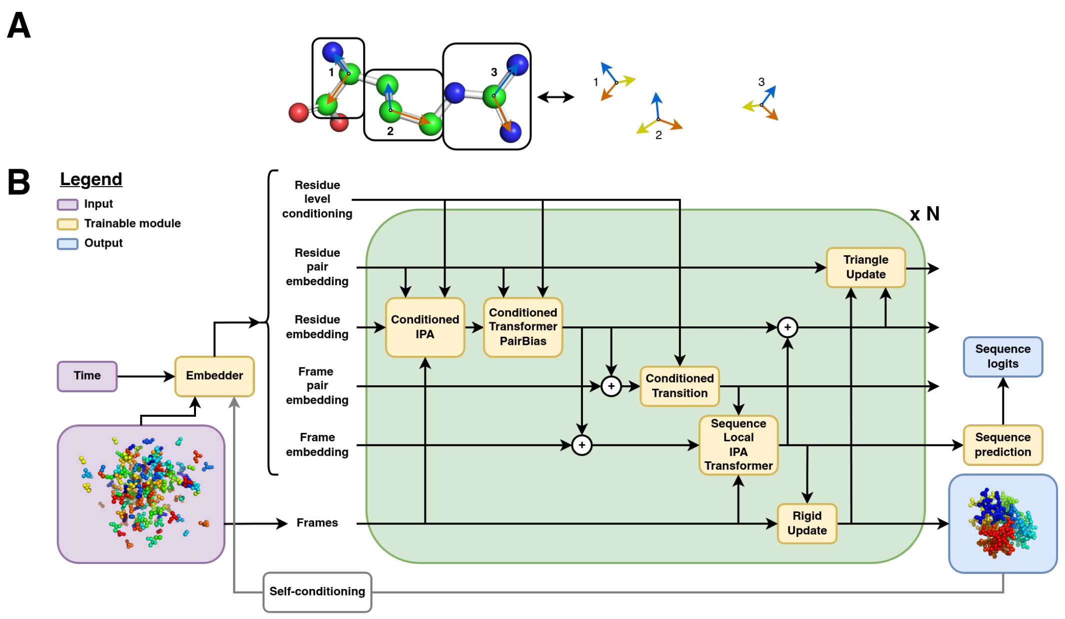


Figure 1: ProteinZen all-atom generation scheme. (A) Example decomposition of a protein residue into oriented rigid chemical groups. (B) ProteinZen architecture.

2.5 Sampling

Previous SE(3) flow-matching models sampled designs by integrating the following ODEs:

$$dr_t = cv_{r,t}(r_t, x_t; \phi)dt \qquad dx_t = v_{x,t}(r_t, x_t; \phi)dt$$

where c is a bespoke scaling of the rotation velocity field (often a large constant such as $c = 10$) at sample time which was found to be necessary for high quality sample generation [19] [20]. Although this trick can be used to generate high quality samples, it offers little tunability over the sampling process at inference time. Given that diffusion models in other literature have shown strong performance gains based on optimization of the integration strategy [22], and the space of samplers for SE(3) flow models remains underexplored, we aimed to create a better SE(3) sampling method.

Inspired by the connection between diffusion and flow matching models via stochastic interpolants [23] [24], we instead propose SDE sampling from an SE(3) flow model. We sample from ProteinZen by integrating the following SDEs:

$$dr_t = v_{r,t}(r_t, x_t; \phi)dt + g_r(t)s_{r,t}(r_t, x_t; \phi)dt + \sqrt{2g_r(t)\gamma_s}d\mathcal{B}_t$$
$$dx_t = v_{x,t}(r_t, x_t; \phi)dt + g_x(t)s_{x,t}(r_t, x_t; \phi)dt + \sqrt{2g_x(t)\gamma_s}d\mathcal{W}_t$$

where $g_r(t), g_x(t)$ control Langevin dynamics and γ_s controls the scale of noise added (computation of terms described in Appendix F). To our knowledge, this is the first example of SDE sampling from an SE(3) flow matching model trained on a deterministic flow. We also develop a custom integration strategy inspired by EDM's stochastic integrator [22] which is further described in Appendix F.

3 Results

3.1 Unconditional Monomer Generation

We evaluate unconditional monomer generation on three criteria (detailed in Appendix G): sequence-structure consistency (SSC), diversity along both structure (DIV-str) and sequence (DIV-seq) axes, and structural novelty (NOV-str). We also evaluate similar metrics for the backbone structures themselves. In this case, we swap the SSC metric for a designability metric (DES), and other two metrics DIV-str and NOV-str are evaluated on the best ProteinMPNN design.

We benchmark ProteinZen and relevant comparison models on 100 samples for each length 70, 100, 200, and 300 residues (Table 1). ProteinZen has competitive performance in unconditional monomer

Method	All-atom			PMPNN 8		
	SSC (↑)	DIV-str/DIV-seq (↑)	NOV-str (↓)	DES (↑)	DIV-str (↑)	NOV-str (↓)
ProteinGenerator [17]	0.11	0.39 / 0.76	0.85	0.81	0.27	0.81
PLAID [18]	0.33	0.57 / 0.80	0.88	0.47	0.59	0.85
APM [25]	0.46	0.36 / 0.79	0.84	0.81	0.34	0.84
Protpardelle-1c cc91 [11]	0.59	0.09 / 0.93	0.78	0.92	0.09	0.78
Pallatom* [12]	0.68	0.61 / 0.92	0.75	0.88	0.67	0.73
La-Proteina [16]	0.81	0.56 / 0.89	0.79	0.99	0.56	0.77
ProteinZen (ODE)	0.56	0.46 / 0.96	0.80	0.87	0.48	0.79
ProteinZen (SDE)	0.65	0.29 / 0.91	0.82	0.91	0.29	0.80

Table 1: Sample quality metrics for ProteinZen and relevant comparison models. *Pallatom was trained on proteins of length at most 128, so the reported performance is partially out-of-distribution.

generation, having a respectable 65% of samples being sequence-structure consistent, similar to Pallatom and only outperformed by concurrent work La-Proteina. We continue to recapitulate the observed trend that all-atom generation models generally have much lower SSC than DES, signaling a disconnect between generated backbones and their associated sequence and atomic detail. However, more recently developed models have began to close the performance gap. We provide more in depth analysis of ProteinZen samples in Appendix G.

In Table 1, we also evaluate the performance of ProteinZen using previously reported ODE samplers to provide a comparison point against our proposed SDE sampler. For the benchmark values reported, we use the SDE sampler with a sampling parameter set we found to maximize SSC, yielding a significant performance boost on SSC at the cost of structural diversity. However, we find the SDE sampler also achieves a better tradeoff between diversity and SSC than using the previous ODE method and can be controlled by tuning the noise scale parameter γ_s (Figure F.1).

3.2 Motif scaffolding

Although methods have continually improved at unconditional generation tasks, motif scaffolding remain a more challenging task. We evaluate motif scaffolding on the Protpardelle benchmark set [11] under two scenarios: (1) one-shot all-atom motif scaffolding and (2) motif scaffolding with sequence redesign. Our primary metric is *unique success rate* per task (see Appendix H for definition), which can intuitively be understood as the number of unique high quality scaffolds generated per set of design attempts. For models with such capabilities, we also evaluate performance on both *indexed motif scaffolding* (motif fragment order and spacing is pre-specified) and *unindexed motif scaffolding* (must infer both motif fragment order and spacing).

3.2.1 One-shot all-atom motif scaffolding

One-shot all-atom motif scaffolding serves to evaluate the performance of all-atom generative models against one another. We compare ProteinZen against concurrently developed models La-Proteina and Protpardelle-1c, where La-Proteina can be run in both indexed and unindexed mode but Protpardelle-1c can only be run in indexed mode. In indexed mode (Figure 2A) ProteinZen outperforms concurrently developed methods on unique success rate, solving 25/26 tasks and performing the best at 18/26 of them. In unindexed mode (Figure 2B) ProteinZen and La-Proteina have more comparable performance, with ProteinZen solving 24/26 tasks and performing the best at 12/26 of them.

3.2.2 Motif scaffolding with sequence redesign

To evaluate how all-atom models compare against existing methods, we benchmark against RFdiffusion which is currently the most popular model for protein motif scaffolding. To enable fair comparison, we evaluate motif scaffolding using ProteinMPNN [26] to redesign the sequences of generated scaffolds (the motif sequence remains fixed). Results are shown in Figure 2C. We find that ProteinZen consistently yields the best performance when compared against all models, solving all 26 tasks and performing the best on 21/26 tasks. Notably, the unique task success rate of all all-atom models is significantly improved by sequence redesign, highlighting that the challenge in ensuring

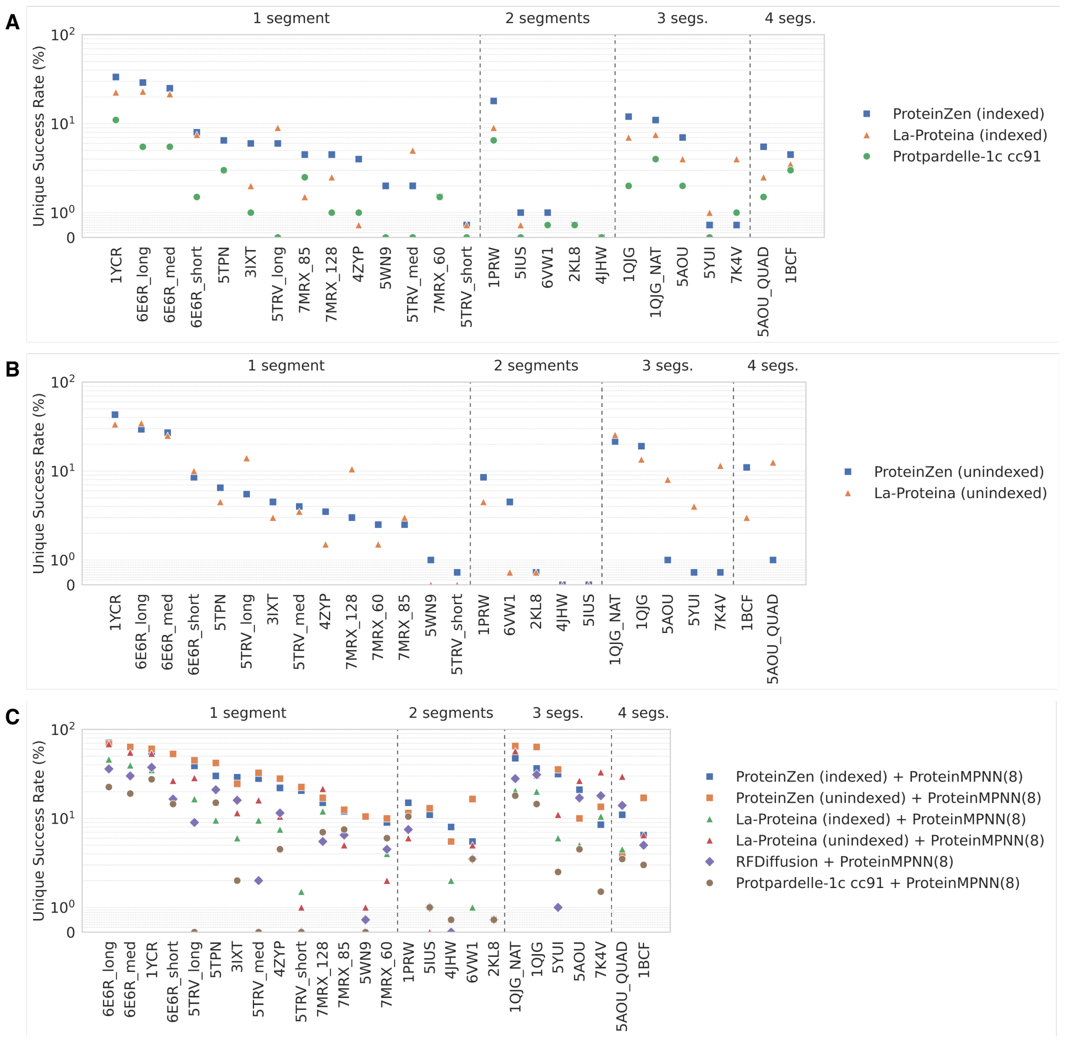


Figure 2: Performance on Protpardelle motif scaffolding benchmark. (A) Indexed one-shot all-atom motif scaffolding (B) Unindexed one-shot all-atom motif scaffolding (C) Motif scaffolding with sequence redesign.

consistency between co-designed sequence and generated backbone is crucial to performant motif scaffolding as well.

4 Conclusions

We present ProteinZen, a method which utilizes flow-matching over rigid oriented chemical groups to generate all-atom protein structures. It exhibits promising performance on all-atom generation tasks, where 65% of unconditional monomer samples are sequence-structure consistent with atomic accuracy, and also yields state-of-the-art performance on the majority of benchmarked motif scaffolding tasks. Future work will extend ProteinZen to various other conditional generation tasks, including the binding of nucleic acids, small molecule, and protein targets.

Acknowledgments and Disclosure of Funding

We would like to thank Dru Myerscough, Tobias Dorer, Hunter Nisonoff, Ishan Gaur, and Junhao Xiong for helpful discussions. A.J.L. is in part supported by a National Science Foundation Graduate Research Fellowship (NSF GRFP) and the UCSF Discovery Fellows program. T.K. is a Biohub San

Francisco Investigator. Portions of this work were performed on the Wynton HPC Co-Op cluster, which is supported by UCSF research faculty and UCSF institutional funds. The authors wish to thank the UCSF Wynton team for their ongoing technical support of the Wynton environment.

References

[1] 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.

[2] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[3] Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, Yehlin Cho, Kourosh H. Ghamary, Laura Vinué, Brahm J. Yachnin, Andrew M. Wollacott, Stephen Buckley, Adrie H. Westphal, Simon Lindhoud, Sandrine Georgeon, Casper A. Goverde, Georgios N. Hatzopoulos, Pierre Gönczy, Yannick D. Muller, Gerald Schwank, Daan C. Swarts, Alex J. Vecchio, Bernard L. Schneider, Sergey Ovchinnikov, and Bruno E. Correia. One-shot design of functional protein binders with bindcraft. *Nature*, August 2025.

[4] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, July 2023.

[5] John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, November 2023.

[6] Woody Ahern, Jason Yim, Doug Tischer, Saman Salike, Seth M. Woodbury, Donghyo Kim, Indrek Kalvet, Yakov Kipnis, Brian Coventry, Han Raut Altae-Tran, Magnus Bauer, Regina Barzilay, Tommi S. Jaakkola, Rohith Krishna, and David Baker. Atom level enzyme active site scaffolding using rfdiffusion2. *bioRxiv*, 2025.

[7] Justas Dauparas, Gyu Rie Lee, Robert Pecoraro, Linna An, Ivan Anishchenko, Cameron Glasscock, and D. Baker. Atomic context-conditioned protein sequence design using ligandmpnn. *bioRxiv*, 2023.

[8] Anna Lauko, Samuel J. Pellock, Ivan Anischanka, Kiera H. Sumida, David Juergens, Woody Ahern, Alex Shida, Andrew Hunt, Indrek Kalvet, Christoffer Norn, Ian R. Humphreys, Cooper Jamieson, Alex Kang, Evans Brackenbrough, Asim K. Bera, Banumathi Sankaran, K. N. Houk, and David Baker. Computational design of serine hydrolases. *bioRxiv*, 2024.

[9] C. J. Markin, D. A. Mokhtari, F. Sunden, M. J. Appel, E. Akiva, S. A. Longwell, C. Sabatti, D. Herschlag, and P. M. Fordyce. Revealing enzyme functional architecture via high-throughput microfluidic enzyme kinetics. *Science*, 373(6553):eabf8761, 2021.

[10] Amy B. Guo, Deniz Akpinaroglu, Christina A. Stephens, Michael Grabe, Colin A. Smith, Mark J. S. Kelly, and Tanja Kortemme. Deep learning–guided design of dynamic proteins. *Science*, 388(6749):eadr7094, 2025.

[11] Alexander E. Chu, Jinho Kim, Lucy Cheng, Gina El Nesr, Minkai Xu, Richard W. Shuai, and Po-Ssu Huang. An all-atom protein generative model. *Proceedings of the National Academy of Sciences*, 121(27):e2311500121, 2024.

[12] Wei Qu, Jiawei Guan, Rui Ma, Ke Zhai, Weikun Wu, and Haobo Wang. P(all-atom) is unlocking new path for protein design. *bioRxiv*, 2024.

[13] Tianyu Lu, Richard Shuai, Petr Kouba, Zhaoyang Li, Yilin Chen, Akio Shirali, Jinho Kim, and Po-Ssu Huang. Conditional protein structure generation with protpardelle-1c. *bioRxiv*, 2025.

[14] Ruizhe Chen, Dongyu Xue, Xiangxin Zhou, Zaixiang Zheng, Xiangxiang Zeng, and Quanquan Gu. An all-atom generative model for designing protein complexes, 2025.

[15] Alex J. Li and Tanja Kortemme. Proteinzen: combining latent and se(3) flow matching for all-atom protein generation, 2024.

[16] Tomas Geffner, Kieran Didi, Zhonglin Cao, Danny Reidenbach, Zuobai Zhang, Christian Dallago, Emine Kucukbenli, Karsten Kreis, and Arash Vahdat. La-proteina: Atomistic protein generation via partially latent flow matching, 2025.

[17] Sidney Lyayuga Lisanza, Jacob Merle Gershon, Samuel W. K. Tipps, Jeremiah Nelson Sims, Lucas Arnoldt, Samuel J. Hendel, Miriam K. Simma, Ge Liu, Muna Yase, Hongwei Wu, Claire D. Tharp, Xinting Li, Alex Kang, Evans Brackenbrough, Asim K. Bera, Stacey Gerben, Bruce J. Wittmann, Andrew C. McShan, and David Baker. Multistate and functional protein design using rosettafold sequence space diffusion. *Nature Biotechnology*, 43(8):1288–1298, September 2024.

[18] Amy X. Lu, Wilson Yan, Sarah A. Robinson, Simon Kelow, Kevin K. Yang, Vladimir Gligorijevic, Kyunghyun Cho, Richard Bonneau, Pieter Abbeel, and Nathan C. Frey. All-atom protein generation with latent diffusion. *bioRxiv*, 2025.

[19] Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with se(3) flow matching, 2023.

[20] Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian FATRAS, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. SE(3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[21] Jason Yim, Brian L. Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. SE(3) diffusion model with application to protein backbone generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 40001–40039. PMLR, 23–29 Jul 2023.

[22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[23] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023.

[24] Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Proteina: Scaling flow-based protein structure generative models, 2025.

[25] Ting Chen, Ruixiang ZHANG, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[27] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David 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, July 2021.

[28] H. M. Berman. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[29] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50(D1):D439–D444, 11 2021.

[30] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, December 1983.

[31] Inigo Barrio-Hernandez, Jingi Yeo, Jürgen Jänes, Milot Mirdita, Cameron L. M. Gilchrist, Tanita Wein, Mihaly Varadi, Sameer Velankar, Pedro Beltrao, and Martin Steinegger. Clustering predicted structures at the scale of the known protein universe. *Nature*, 622(7983):637–645, September 2023.

[32] Ruidong Wu, Ruihan Guo, Rui Wang, Shitong Luo, Yue Xu, Jiahan Li, Jianzhu Ma, Qiang Liu, Yunan Luo, and Jian Peng. Fafe: Immune complex modeling with geodesic distance loss on noisy group frames, 2024.

[33] Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with genie 2, 2024.

[34] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, May 2024.

[35] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[36] Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020.

[37] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[38] Michel van Kempen, Stephanie S. Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron L. M. Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature Biotechnology*, 42(2):243–246, May 2023.

[39] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, October 2017.

[40] Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1 democratizing biomolecular interaction modeling. *bioRxiv*, 2025.

Appendix

A Residue frame construction

In this work, we define the decomposition of residues into rigid chemical fragments. To make the decomposition more unified, we use a common specification of the backbone fragment as

`{N, CA, C, O, CB}`

where `CB` is included for all residues that contain it (e.g. not `GLY`). We construct the frame as done in AlphaFold2 [27] from the `[N, CA, C]` atoms and follow FrameFlow [19] in computing the `O` coordinates from ideal peptide bond geometries. We then decompose the residue sidechain as follows:

ALA:	none
ARG:	[NE, NH1, NH2, CZ], [CB, CG, CD]
ASN:	[CG, ND2, OD1]
ASP:	[CG, OD1, OD2]
CYS:	[CA, CB, SG]
GLN:	[CG, CD, NE2, OE1]
GLU:	[CG, CD, OE1, OE2]
GLY:	none
HIS:	[CG, CD2, ND1, CE1, NE2]
ILE:	[CB, CG1, CD1], [CB, CG1, CG2]
LEU:	[CG, CD1, CD2]
LYS:	[CD, CE, NZ], [CB, CG, CD]
MET:	[CG, SD, CE]
PHE:	[CG, CD1, CD2, CE1, CE2, CZ]
PRO:	[CB, CG, CD]
SER:	[CA, CB, OG]
THR:	[CB, CG2, OG1]
TRP:	[CG, CD1, CD2, CE2, CE3, NE1, CH2, CZ2, CZ3]
TYR:	[CG, CD1, CD2, CE1, CE2, OH, CZ]
VAL:	[CB, CG1, CG2]

For each rigid group, we define a set of axes to that allows us to construct the frame from atomic coordinates:

ALA:	none
ARG:	[NH1, CZ, NH2], [CB, CG, CD]
ASN:	[CG, ND2, OD1]
ASP:	[OD1, CG, OD2]
CYS:	[CA, CB, SG]
GLN:	[CD, NE2, OE1]
GLU:	[OE1, CD, OE2]
GLY:	none
HIS:	[ND1, CE1, NE2]
ILE:	[CB, CG1, CD1], [CG2, CB, CG1]
LEU:	[CD1, CG, CD2]
LYS:	[CD, CE, NZ], [CB, CG, CD]
MET:	[CG, SD, CE]
PHE:	[CE1, CZ, CE2]
PRO:	[CB, CG, CD]
SER:	[CA, CB, OG]
THR:	[CB, OG1, CG2]
TRP:	[CH2, CZ2, CZ3]
TYR:	[CE2, OH, CE1]
VAL:	[CB, CG1, CG2]

where in a list $[x, y, z]$ we construct the frame using Gram-Schmidt orthonormalization from vectors $(\text{coord}(x) - \text{coord}(y))$ and $(\text{coord}(z) - \text{coord}(y))$.

In order to create a representation expressive enough for all the canonical amino acids, we need to ensure each residue is represented with three oriented rigid bodies. For residue types with fewer than three rigid components, we supplement the representation with the frame defined by the first sidechain rigid group listed above. If there are no sidechain frames, the backbone frame is duplicated twice instead.

B Noise distribution

Following FrameFlow [19], during training we use data-dependent coupling to sample from our noise distribution. For the rotation noise, we sample from the IgSO3 distribution centered around the ground truth data and with $\sigma = 1.5$, which helps push noise samples away from areas where we get degenerate solutions for the geodesic. For the translation noise, we sample from a standard isotropic Gaussian with $\sigma = 1.6$nm and align this noise against the ground truth data using the Kabsch algorithm.

During sampling we sample from the uniform distribution over SO(3) for rotations and an isotropic Gaussian with standard deviation $\sigma = 1.6$nm.

C Dataset construction

We construct AFDB512-clusters using two different data sources: the PDB [28] and the AlphaFold predicted structures database [29]. For PDB structures, we adapt the strategy from FrameDiff [21] and select structures which meet the following criteria:

- X-ray structure or EM structure with resolution < 3.0 Å.
- Oligomeric state is specified as "monomer" in the mmCIF header.
- Length is between 31 and 512 residues.
- Greater than 50% of residues are in defined secondary structure elements as defined by DSSP [30]

We cluster these structures based on the 40% sequence identity cluster assignments specified by the PDB. This results in 22,518 structures total.

For AFDB structures, we adapt strategies from Protpardelle [11] and Pallatom [12] and select structures which meet the following criteria:

- pLDDT > 80.
- Length is between 31 and 512 residues.
- Greater than 50% of residues are in defined secondary structure elements as defined by DSSP [30].
- Longest loop < 15 residues.
- More than 2 secondary structure elements as defined by DSSP.
- Structures are >30% core, where a core residue is defined as a residue with at least 18 neighboring C_α atoms within 10 Å.
- Radius of gyration is less than 30 Å.

We first apply this filtering criteria to all AFDB Foldseek cluster representatives [31], then to all cluster members for the retained clusters. This results in 2,522,042 structures total.

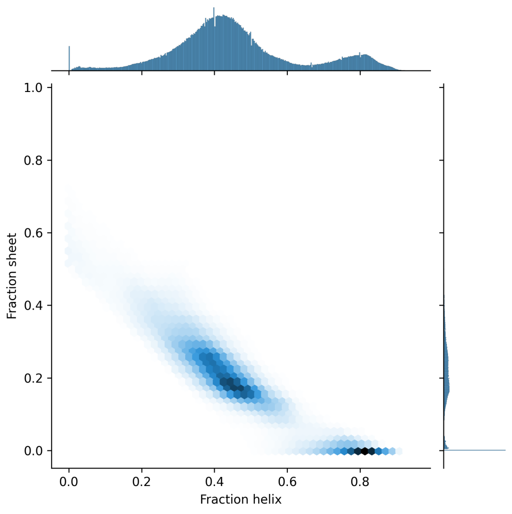


Figure C.1: AFDB512-clusters secondary structure composition

D Training details

As mentioned in the main text, we formulate our flow-matching losses as denoising objectives and train a denoiser $\hat{r}_1, \hat{x}_1 = D_\phi(T_t, t)$ to minimize the following losses:

$$\text{angle\_axis\_loss}(v_1, v_2) = 0.5\|\omega(v_1) - \omega(v_2)\|^2 + \|u(v_1) - u(v_2)\|^2$$

$$\mathcal{L}_{rot} = \frac{1}{(1-t)^2}\text{angle\_axis\_loss}(\log_{r_t}(\hat{r}_1(T_t, t)), \log_{r_t} r_1)$$

$$\mathcal{L}_{trans} = \frac{1}{(1-t)^2}\|\hat{x}_1(T_t, t) - x_1\|^2$$

where $\omega(v)$ is the length of v and $u(v)$ is the normalized vector of v. This form of $\mathcal{L}_{rot}$ is a separable analog to the original SO(3) denoising objective inspired by FrameDiff [21] which we found led to more stable training in early experiments. As our frame-to-atom mapping requires a residue identity as input, we train a small prediction head at the end of the denoising network via cross-entropy loss to predict the ground-truth sequence, which allows us to decode out the identity of the residue at the end of a denoising trajectory:

$$\mathcal{L}_{seq} = -\sum_{\alpha \in \mathcal{A}} \hat{p}(\alpha) \ln(\hat{p}(\alpha))$$

Finally, inspired by Wu et al. [32], we add an auxiliary loss $\mathcal{L}_{FAFE}$ to encourage better global denoising properties via pairwise denoising terms.

$$d_{\mathbb{R}^3}(k_1, k_2) = ||(T^{(k_1)})^{-1} \circ x^{(k_2)} - (\hat{T}^{(k_1)})^{-1} \circ \hat{x}^{(k_2)}||$$

$$\text{relrot}(r^{(i)}, r^{(j)}) = (r^{(i)})^\top (r^{(j)})$$

$$\text{geodesic\_dist}(r^{(i)}, r^{(j)}) = \arccos\left(\frac{\text{tr}((r^{(i)})^\top r^{(j)}) - 1}{2}\right)$$

$$d_{SO(3)}(k_1, k_2) = \text{geodesic\_dist}\left(\text{relrot}(\hat{r}^{(k_1)}, \hat{r}^{(k_2)}), \text{relrot}(r^{(k_1)}, r^{(k_2)})\right)$$

$$\mathcal{L}_{FAFE}(w) = \frac{1}{(1-t)^2}\sqrt{\frac{1}{N^2}\left[\sum_{k_1,k_2} w^{(i)} w^{(j)} \left(d_{\mathbb{R}^3}(k_1, k_2)^2 + d_{SO(3)}(k_1, k_2)^2\right)\right]}$$

In total, the full loss we optimize for is

$$\mathcal{L} = \mathbb{E}_{t\sim\pi_{ln}(0,1), T_0\sim\rho_0, T_1\sim\rho_1}\left[\frac{1}{N}\sum_{i=1}^{N} w^{(i)}\left(\mathcal{L}_{rot}^{(i)} + \mathcal{L}_{trans}^{(i)} + 0.25\mathcal{L}_{seq}^{(i)}\right) + 0.5\mathcal{L}_{FAFE}(w)\right]$$

where $\pi_{ln}(0,1)$ is the logit-normal distribution with location 0 and scale 1, and w is a residue-wise loss weighting term.

In practice we find better performance when using a residue-identity-based loss upweighting for w, and we set

$$w^{(i)} = \begin{cases} 2 \text{ if } S^{(i)} \in \{\text{CYS, GLU, HIS, PRO, GLN, ARG, TRP}\} \\ 1 \text{ else} \end{cases}$$

We suspect that this improves performance because it reduces the tendency of the model to excessively design alanines under reduced-noise sampling schemes.

D.1 Pretraining

We perform pretraining in multiple stages as listed in Table D.1. We follow Genie2 [33] in generating motif scaffolding training tasks but cap the max motif size to either 50% of the protein structure or 20 residues, whichever is smaller. Additionally, we select the denoising task for each motif residue independently as follows:

- Specify the entire residue with probabiliy 0.30. The residue sequence identity is provided.
- Specify only the backbone frame of the residue with probability 0.35. The residue sequence identity is masked.
- Specify only the tip frame of the residue with probability 0.35. The residue sequence identity is provided.

We note that pretraining was performed in one shot, and further tuning the training regime and hyperparameters could yield performance benefits and reduce training time. Training was performed on 8 NVIDIA H100s GPUs.

Stage	Phase 1.1	Phase 1.2	Phase 1.3	Phase 2
Batch size	32	32	32	32
Crop size	256	256	256	384
Number of steps	300k	80k	120k	200k
Gradient clip value	None	None	5.0	5.0
% unconditional tasks	15%	45%	45%	45%
% indexed motif scaffolding tasks	15%	10%	10%	10%
% unindexed motif scaffolding tasks	70%	45%	45%	45%

Table D.1: Pretraining regime

We sample from both PDB and AFDB data such that the ratio of real-to-synthetic structures is 1:3, and sample structures with weight inversely proportional to the structure's associated cluster size

$$w \propto \frac{1}{N_{clust}}$$

We use the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ and a base learning rate of 0.0001. After 300k steps, we start to decay the learning rate by 0.95 every 20k steps.

D.2 Motif scaffolding finetuning

From the Phase 2 checkpoint, we train a motif scaffolding specific set of weights by further finetuning on motif scaffolding as described in Table D.2. We warm start the learning rate back to 1e-4 and allow it to decay as before every 20k steps.

Stage	FT-Motif
Batch size	48
Crop size	320
Number of steps	80k
Gradient clip value	5.0
% indexed motif scaffolding tasks	20%
% unindexed motif scaffolding tasks	80%

Table D.2: Motif scaffolding finetuning regime

E ProteinZen architecture details

E.1 Invariant Point Attention Variants

In order to improve performance and computational efficiency while retaining the benefits of the original algorithm, we make several variants of Invariant Point Attention (IPA) which incorporate different adjustments.

Sequence-local IPA Inspired by AlphaFold 3 [34], we perform efficient local modeling by restricting attention to pre-defined sequence-local blocks. In this work, per sequence-local block, we use 16 query frames and 64 key frames.

Pre-LayerNorm IPA Invariant Point Attention in it's original formulation followed earlier trends using post-layer norm in Transformer-style blocks. In line with modern Transformers, we instead switch to using pre-layer norm IPA.

Ada-LN IPA and output gating Given pre-layernorm IPA, we also create a variant that enables conditioning via adaptive layer normalization techniques such as AdaLN [35] and output gating.

QK-norm IPA Vanilla Transformers have found that normalization of the query and key vectors prior to attention calculations helps improve training stability [36]. We found this to be the case for IPA as well, and implement QK-normalization using a learnable LayerNorm for both query and key values.

E.2 Rotation preconditioning

Previous IPA-based SE(3) flow-matching models note an explosion in the learned rotation velocity norm as $t \to 1$ which results in poor sample quality, necessitating a tampering of the velocity at inference-time at times close to $t = 1$ using some scaling function, commonly a multiplicative factor of $(1 - t)$. We choose to directly embed this scaling factor into the model itself by scaling the magnitude of denoising prediction by $(1 - t)$. We call this technique “rotation preconditioning” as inspired by EDM preconditioning [22], and note that is numerically equivalent to the model regressing against the rotation velocity field directly rather than the denoising target. Although rotation preconditioning did not yield any significant performance differences during small-scale experiments, it allows us to simplify the interpretation and manipulation of the velocity field output at sample time.

E.3 Motif input specification

Rather than impute and freeze the target motif as input specification, we instead append the motif as additional residues/frames and specify these tokens are motif inputs. In the indexed motif scaffolding case, we provide sequence indices as input, but in the unindexed case sequence indices are masked. During training, no motif imputation is performed. At inference time, at the end of a denoising trajectory, the model will compute which residues correspond to motif residues and imputes the motif at these residue positions.

E.4 Algorithm descriptions

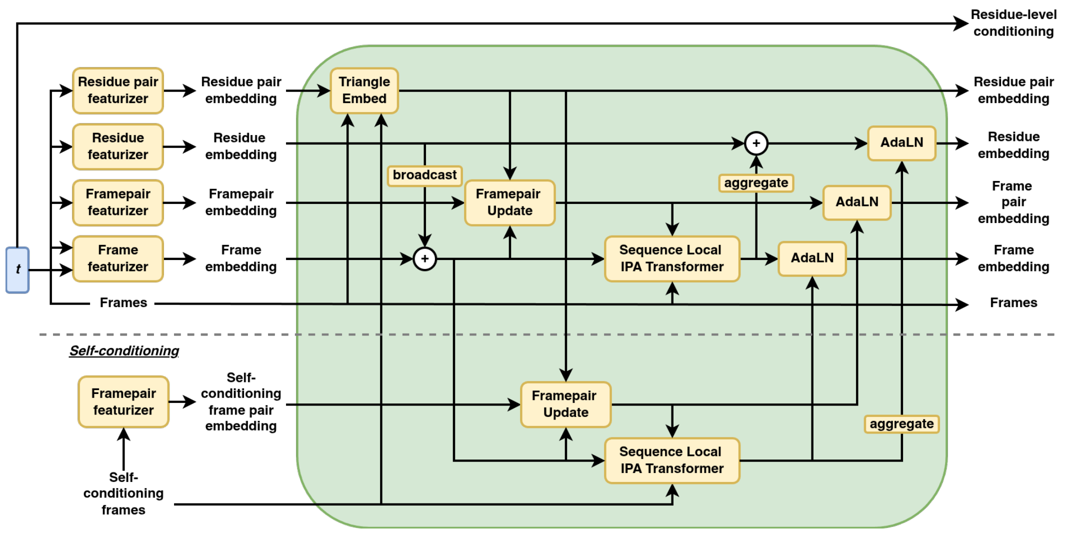


Figure E.1: ProteinZen embedder architecture

We sketch pseudocode for some notable algorithms in ProteinZen (both the embedder in Figure E.1 and the denoiser in Figure 1) in the algorithms below. We use i, j to index residue-level features and l, m to index frame level features. We further distinguish between residue- and frame-level by adding hats to all frame-level intermediate quantities. Reference implementations can be found at `https://github.com/alexjli/proteinzen`.

Algorithm 1 residue_featurizer

```
Require: Sequence $f_i^{\text{seq}}$, Sequence index $f_i^{\text{seq\_idx}}$, Unindexed residue marker $f_i^{\text{res\_is\_unindexed}}$
1: $s_i$ = LinearNoBias(OneHot($f_i^{\text{seq}}$))
2: $s_i$ += LinearNoBias(SinusoidalEmbedding($f_i^{\text{seq\_idx}}$)) $\odot f_i^{\text{token\_is\_unindexed}}$
3: $s_i$ += LinearNoBias($f_i^{\text{res\_is\_unindexed}}$)
4: return $s_i$
```

Algorithm 2 residue_pair_featurizer

```
Require: Rigids $T_i = (r_i \in \text{SO(3)}, \vec{x}_i \in \mathbb{R}^3)$, Sequence index $f_i^{\text{seq\_idx}}$, Chain index $f_i^{\text{chain\_idx}}$, Unindexed residue marker $f_i^{\text{res\_is\_unindexed}}$, (optional) Self-conditioning rigids $\tilde{T}_i = (\tilde{r}_i \in \text{SO(3)}, \tilde{\vec{x}}_i \in \mathbb{R}^3)$
1: # Featurize pairwise rigid spatial information (distance, relative orientation), sequence distance, and chain information
2: $r_{ij} = r_i^\top r_j$
3: $h_{ij} = \text{as\_quaternion}(r_{ij}) \in \mathbf{H}$
4: $\vec{x}_{ij} = T_i^{-1} \circ (\vec{x}_j - \vec{x}_i)$
5: $d_{ij} = \|\vec{x}_{ij}\|$
6: $v_{ij} = (f_i^{\text{chain\_idx}} == f_j^{\text{chain\_idx}})$
7: $v_{ij} \leftarrow v_{ij} \wedge (f_i^{\text{res\_is\_unindexed}} \vee f_j^{\text{res\_is\_unindexed}})$
8: $z_{ij}$ = LinearNoBias(SinusoidalEmbedding($f_i^{\text{seq\_idx}} - f_j^{\text{seq\_idx}})v_{ij}$)
9: $a_{ij} = \text{concat}\,(h_{ij}, \vec{x}_{ij}, \text{RBF}(d_{ij}))$
10: # Perform self-conditioning if self-conditioning rigids are provided
11: if $\tilde{T}_i \neq \varnothing$ then
12:     $\tilde{r}_{ij} = \tilde{r}_i^\top \tilde{r}_j$
13:     $\tilde{h}_{ij} = \text{as\_quaternion}(\tilde{r}_{ij}) \in \mathbf{H}$
14:     $\tilde{\vec{x}}_{ij} = \tilde{T}_i^{-1} \circ (\tilde{\vec{x}}_j - \tilde{\vec{x}}_i)$
15:     $\tilde{d}_{ij} = \|\tilde{\vec{x}}_{ij}\|$
16:     $\tilde{a}_{ij} = \text{concat}\,(h_{ij}, \vec{x}_{ij}, \text{Distogram}(d_{ij}))$
17: else
18:     $\tilde{a}_{ij} = \mathbf{0} \odot a_{ij}$
19:     $z_{ij}$ = LinearNoBias(concat($a_{ij}, \tilde{a}_{ij}$))
20: end if
21: return $z_{ij}$
```

Algorithm 3 frame_featurizer

```
Require: Local frame index $\hat{f}_l^{\text{rigids\_idx}}$, Interpolation time $t$
1: $\hat{s}_l$ = LinearNoBias(OneHot($\hat{f}_l^{\text{rigids\_idx}}$))
2: $\hat{s}_l$ += LinearNoBias(TimestepEmbedding($t$))
3: return $\hat{s}_l$
```

Algorithm 4 framepair_featurizer

```
Require: Rigids $T_l = (r_l \in \mathrm{SO}(3), \vec{x}_l \in \mathbb{R}^3)$, Rigid parent residue $f_l^{\text{rigid\_to\_res\_idx}}$, $N_{\text{queries}} = 16$, $N_{\text{keys}} = 64$, $\mathcal{S}_{\text{subset centers}} = \{7.5, 23.5, 39.5, ...\}$
1: # Restrict attention to sequence-local blocks
2: $\hat{M}_{lm} = \begin{cases} 1 & \text{if } |l-c| < N_{\text{queries}}/2 \wedge |m-c| < N_{\text{keys}}/2 \quad \forall c \in \mathcal{S}_{\text{subset centers}} \\ 0 & \text{else} \end{cases}$
3: $r_{lm} = r_l^\top r_m$
4: $h_{lm} = \text{as\_quaternion}(r_{lm}) \in \mathbf{H}$
5: $\vec{x}_{lm} = T_l^{-1} \circ (\vec{x}_m - \vec{x}_l)$
6: $v_{ij} = (f_i^{\text{rigid\_to\_res\_idx}} == f_j^{\text{rigid\_to\_res\_idx}})$
7: $\hat{z}_{lm} = \text{LinearNoBias}(\vec{x}_{lm})$
8: $\hat{z}_{lm}$ += LinearNoBias($h_{lm}$)
9: $\hat{z}_{lm}$ += LinearNoBias$\left(\frac{1}{1+\|\vec{x}_{lm}\|^2}\right)$
10: $\hat{z}_{lm} \leftarrow \hat{z}_{lm} v_{lm} \hat{M}_{lm}$
11: return $\hat{z}_{lm}$, $\hat{M}_{lm}$
```

Algorithm 5 triangle_embed

```
Require: Residue pair representation $z_{ij}$, $N_{\text{blocks}} = 2$
1: for $k \in \{1..N_{\text{blocks}}\}$ do
2:     $z_{ij}$ += TriangleMultiplicationOutgoing($z_{ij}$)
3:     $z_{ij}$ += TriangleMultiplicationIncoming($z_{ij}$)
4:     $e_{ij}$ = LinearNoBias($z_{ij}$)
5:     $z_{ij}$ += TriangleAttentionStartingNode($z_{ij}$, bias $= e_{ij}$)
6:     $z_{ij}$ += TriangleAttentionEndingNode($z_{ij}$, bias $= e_{ij}$)
7:     $z_{ij}$ += Transition($z_{ij}$)
8: end for
9: return LinearNoBias($z_{ij}$)
```

Algorithm 6 framepair_update

```
Require: Frames singles features $\hat{s}_l$, Residue pair features $z_{ij}$, Framepair features $\hat{z}_{lm}$, Rigid parent residue $f_l^{\text{rigid\_to\_res\_idx}}$, Framepair mask $\hat{M}_{ij}$
1: $\hat{z}_{lm}$ = LinearNoBias($z_{f_l^{\text{rigid\_to\_res\_idx}}, f_m^{\text{rigid\_to\_res\_idx}}}$)
2: $\hat{z}_{lm}$ += LinearNoBias($s_l$)
3: $\hat{z}_{lm}$ += LinearNoBias($s_m$)
4: # Small FFN
5: $\hat{z}_{lm}$ += Linear(ReLU(Linear(ReLU(Linear(LayerNorm($\hat{z}_{lm}$))))
6: return $\hat{z}_{lm}$
```

Algorithm 7 conditioned_ipa

```
Require: Singles features $s_i$, Pair features $z_{ij}$, Frames $T_i = (r_i, x_i)$, Conditioning features $a_i$, Attention bias $\beta_{ij}$
 1: # Pre-LayerNorm with option for conditioning via AdaLN
 2: if $a_i \neq \varnothing$ then
 3:     $s_i \leftarrow$ AdaLN($s_i, a_i$)
 4: else
 5:     $s_i \leftarrow$ LayerNorm($s_i$)
 6: end if
 7: # IPA algorithm with QK-normalization
 8: $z_{ij} \leftarrow$ LayerNorm($z_{ij}$)
 9: $q_i^h, k_i^h, v_i^h =$ LinearNoBias($s_i$)
10: # QK-normalization
11: $q_i^h \leftarrow$ LayerNorm($q_i^h$)
12: $k_i^h \leftarrow$ LayerNorm($k_i^h$)
13: $\vec{q}_i^{hp}, \vec{k}_i^{hp}, \vec{v}_i^{hp} =$ LinearNoBias($s_i$)
14: $b_{ij}^h = \beta_{ij} +$ LinearNoBias($z_{ij}$)
15: $\zeta_{ij}^h =$ LinearNoBias($z_{ij}$)
16: $w_C = \sqrt{\frac{2}{9N_{\text{query points}}}}$
17: $w_L = \sqrt{\frac{1}{3}}$
18: $a_{ij}^h = \text{softmax}_j \left( w_L \left( \frac{1}{\sqrt{c}} {q_i^h}^\top k_i^h + b_{ij}^h - \frac{\gamma^h w_C}{2} \sum_p \| T_i \circ \vec{q}_i^{hp} - T_j \circ \vec{k}_j^{hp} \| \right) \right)$
19: $\tilde{o}_i^h = \sum_j a_{ij}^h \zeta_{ij}^h$
20: $o_i^h = \sum_j a_{ij}^h v_j^h$
21: $\vec{o}_i^{hp} = T_i^{-1} \circ \sum_j a_{ij}^h (T_j \circ \vec{v}_j^{hp})$
22: $\tilde{s_i} =$ LinearNoBias(concat$_{h,p}$($\tilde{o}_i^h, o_i^h, \vec{o}_i^{hp}, \|\vec{o}_i^{hp}\|$))
23: # Output gating
24: if $a_i \neq \varnothing$ then
25:     $\tilde{s_i} = \tilde{s_i} \odot$sigmoid(LinearNoBias($a_i$))
26: end if
27: return $\tilde{s_i}$
```

Algorithm 8 sequence_local_ipa

```
Require: Frame singles features $\hat{s}_l$, Framepair features $\hat{z}_{lm}$, Frames $\hat{T}_l$, Frame conditioning features $\hat{a}_l$, Framepair mask $\hat{M}_{lm}$
 1: # Restrict attention to sequence-local blocks
 2: $\beta_{lm} = 10^{10}(\hat{M}_{lm} - 1)$
 3: return conditioned_ipa($\hat{s}_l, \hat{z}_{lm}, \hat{T}_l, \hat{a}_l, \beta_{lm}$)
```

Algorithm 9 sequence_local_ipa_transformer

```
Require:
Require: Residue singles features $s_i$, Rigid parent residue $f_l^{\text{rigid\_to\_res\_idx}}$, Frame singles features $\hat{s}_l$, Framepair features $\hat{z}_{lm}$, Frames $\hat{T}_l$, Framepair mask $\hat{M}_{lm}$, $N_{\text{blocks}} = 1$
 1: for $k \in \{1..N_{\text{blocks}}\}$ do
 2:     $\hat{s}_l$ += SequenceLocalIPA($\hat{s}_l, \hat{z}_{lm}, \hat{a}_l = \varnothing, \hat{M}_{lm}$)
 3:     $\hat{s}_l$ += ConditionedTransition($\hat{s}_l, s_{f_l^{\text{rigid\_to\_res\_idx}}}$)
 4: end for
 5: return $s_i$
```

Algorithm 10 conditioned_self_attention_pair_bias

```
Require: Singles features $s_i$, Pair features $z_{ij}$, Conditioning features $a_i$
 1: # Pre-LayerNorm with option for conditioning via AdaLN
 2: if $a_i \neq \varnothing$ then
 3:     $s_i \leftarrow$ AdaLN$(s_i, a_i)$
 4: else
 5:     $s_i \leftarrow$ LayerNorm$(s_i)$
 6: end if
 7: # Self-attention with QK-normalization
 8: $z_{ij} \leftarrow$ LayerNorm$(z_{ij})$
 9: $q_i^h, k_i^h, v_i^h =$ LinearNoBias$(s_i)$
10: # QK-normalization
11: $q_i^h \leftarrow$ LayerNorm$(q_i^h)$
12: $k_i^h \leftarrow$ LayerNorm$(k_i^h)$
13: $b_{ij}^h =$ LinearNoBias$(z_{ij})$
14: $a_{ij}^h = \text{softmax}_j \left( \frac{1}{\sqrt{c}} {q_i^h}^\top k_i^h + b_{ij}^h \right)$
15: $o_i^h = \sum_j a_{ij}^h v_j^h$
16: $\tilde{s_i} =$ LinearNoBias(concat$_h(o_i^h)$)
17: # Output gating
18: if $a_i \neq \varnothing$ then
19:     $\tilde{s_i} = \tilde{s_i} \odot$sigmoid(LinearNoBias$(a_i)$)
20: end if
21: return $\tilde{s_i}$
```

Algorithm 11 conditioned_transformer_pair_bias

```
Require:
Require: Singles features $s_i$, Pair features $z_{ij}$, Conditioning features $a_i$ $N_{\text{blocks}} = 2$
 1: for $k \in \{1..N_{\text{blocks}}\}$ do
 2:     $s_i$ += ConditionedAttentionPairBias$(s_i, z_{ij}, a_i)$
 3:     $s_i$ += Transition$(s_i)$
 4: end for
 5: return $s_i$
```

Algorithm 12 triangle_update

```
Require: Frames $T_i = (r_i, x_i)$, Residue singles features $s_i$, Residue pair representation $z_{ij}$
 1: $d_{ij}$ = $\|x_j - x_i\|$
 2: $e_{ij}$ = LinearNoBias(LayerNorm$(z_{ij})$)
 3: $e_{ij}$ += LinearNoBias(RBF$(d_{ij})$)
 4: $z_{ij}$ += TriangleAttentionStartingNode$(z_{ij}$, bias $= e_{ij})$
 5: $z_{ij}$ += TriangleAttentionEndingNode$(z_{ij}$, bias $= e_{ij})$
 6: $a_{ij}$ = LinearNoBias(concat$(s_i, s_j)$)
 7: $z_{ij}$ += ConditionedTransition$(z_{ij}$, cond $= a_{ij})$
 8: return $z_{ij}$
```

Algorithm 13 transition

```
Require: Features $s$
 1: $s \leftarrow$ LayerNorm$(s)$
 2: $a =$ Linear$(s)$
 3: $b =$ Linear$(s)$
 4: return Linear(swish$(a)$ $\odot b$)
```

Algorithm 14 conditioned_transition

```
Require: Features $s$, Conditioning $a$
1: $s \leftarrow \mathrm{AdaLN}(s, a)$
2: $b = \mathrm{swish}(\mathrm{LinearNoBias}(s)) \odot \mathrm{LinearNoBias}(s)$
3: $s \leftarrow \mathrm{sigmoid}(\mathrm{Linear}(a)) \odot \mathrm{Linear}(b)$
4: return $s$
```

F Sampling details

We describe our integrator for SDE sampling in Algorithm 15. We use the following default parameters, but note that these can be tuned per user specification:

- Number of integration steps $T = 400$
- Noise scale parameter $\gamma_s = 0.16$
- Churn parameter $\gamma_c = 0.4$
- Translation Langevian schedule $g_x(t) = (1-t)/(t+0.1)^2$
- Rotation Langevian schedule $g_r(t) = (1-t)/(t+0.1)^2$
- Step scale $\eta = 1.5$
- Low noise threshold $T_{ODE} = 0.99$

For example, we find that changing the noise scale parameter γ_s allows us to tradeoff sample SSC and diversity (Figure F.1). We also compare against the classic SE(3) ODE sampler, which uses Euler integration with $T = 400$ to integrate the following ODEs:

$$dr_t = cv_{r,t}(r_t, x_t; \phi)dt$$
$$dx_t = v_{x,t}(r_t, x_t; \phi)dt$$

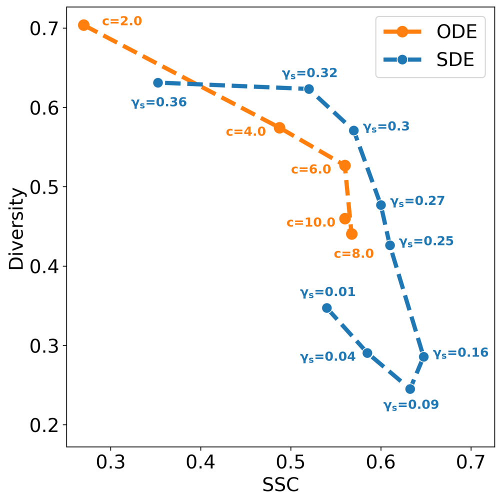


Figure F.1: Tradeoff between diversity and SSC for unconditional monomer generation benchmark. We scan γ_s for the SDE (see Table F.1) and c for the ODE (see Table F.2). All other default parameters remain the same.

γ_s	SSC	Diversity
0.01	0.54	0.35
0.04	0.59	0.29
0.09	0.63	0.25
0.16	0.65	0.29
0.25	0.61	0.43
0.27	0.60	0.48
0.30	0.57	0.57
0.32	0.52	0.62
0.36	0.35	0.63

Table F.1: Raw values for γ_s scan for SDE sampling in Figure F.1

c	SSC	Diversity
2	0.27	0.70
4	0.49	0.57
6	0.56	0.53
8	0.57	0.44
10	0.56	0.46

Table F.2: Raw values for c scan for ODE sampling in Figure F.1

F.1 Velocity and score computation

We estimate the velocity and score functions as follows:

$$v_{r,t}(r_t, x_t; \phi) = \frac{\log_{r_t} \hat{r}_1}{1-t} \qquad s_{r,t}(r_t, x_t; \phi) = \frac{\log_{\hat{r}_1} r_t}{\omega(\hat{r}_1^T r_t)} \partial_\omega f(\omega(\hat{r}_1^T r_t), \sigma(t))$$
$$v_{x,t}(r_t, x_t; \phi) = \frac{\hat{x}_1 - x_t}{1-t} \qquad s_{x,t}(r_t, x_t; \phi) = \frac{t v_{x,t}(r_t, x_t; \phi) - x_t}{1-t}$$

where $f(\omega(r_1^T r_t), \sigma) = \text{IGSO}_3(r_t; r_1, \sigma)$ is the PDF of the IgSO(3) distribution centered at r_1 and with variance σ and $\sigma(t) = \sqrt{(1-t)^2\sigma_0^2 + t^2\sigma_1^2}$ is the estimated variance of the rotation SDE. The $\mathbb{R}^3$ score is derived from the connection between Gaussian flow-matching and diffusion processes [23] and the SO(3) score is computed as done in FrameDiff [21].

Algorithm 15 SDE sampling from ProteinZen

```
Require: Number of integration steps $T$, noise scale parameter $\gamma_s$, churn parameter $\gamma_c$, translation
    Langevian schedule $g_x(t)$, rotation Langevian schedule $g_r(t)$, step scale $\eta$, low noise threshold
    $T_{ODE}$
 1: $\Delta t \leftarrow 1/T$
 2: $r_{t=0} \sim \mathcal{U}_{\mathrm{SO(3)}}()$
 3: $x_{t=0} \sim \mathcal{N}(0, \sigma^2 I)$
 4: for $s$ in $[0, T)$ do
 5:     $t \leftarrow s/T$
 6:     if $t < T_{ODE}$ then                              ▷ At high noise we integrate the SDE
 7:         /** Add noise via churn **/
 8:         // Determine how much churn to apply
 9:         $\hat{t} \leftarrow \min(0, t - \gamma_c \Delta t)$
10:         $\Delta\hat{t} \leftarrow \Delta t + (t - \hat{t})$
11:         // Noise rotation
12:         $z_r \sim \mathcal{N}(0, I)$
13:         $d\mathcal{B}_t \leftarrow z_r \sqrt{2\gamma_s \left(g_r(\hat{t}) - 2g_r(t)\right)}$
14:         $r_t \leftarrow \exp_{r_t}(d\mathcal{B}_t)$
15:         // Noise translation
16:         $z_x \sim \mathcal{N}(0, \sigma_x^2 I)$
17:         $d\mathcal{W}_t \leftarrow z_x \sqrt{2\gamma_s \left(g_x(\hat{t}) - 2g_x(t)\right)}$
18:         $x_t \leftarrow x_t + d\mathcal{W}_t$
19:         // Rename some variables for clarity
20:         $t \leftarrow \hat{t}$
21:         $\Delta t \leftarrow \Delta\hat{t}$
22:         /** Score, velocity prediction **/
23:         // Denoiser prediction
24:         $\hat{r}_1, \hat{x}_1 = D(r_t, x_t, t; \phi)$
25:         $v_{r,t}(r_t, x_t; \phi) = \log_{r_t} \hat{r}_1/(1-t)$
26:         $s_{r,t}(r_t, x_t; \phi) = \partial_\omega f(\omega(\hat{r}_1^T r_t), \sigma_r(t)) \log_{\hat{r}_1} r_t/\omega(\hat{r}_1^T r_t)$
27:         $v_{x,t}(r_t, x_t; \phi) = (\hat{x}_1 - x_t)/(1-t)$
28:         $s_{x,t}(r_t, x_t; \phi) = (t v_{x,t}(r_t, x_t; \phi) - x_t)(1-t)$
29:         /** Integration step **/
30:         $r_t \leftarrow \exp_{r_t}\left[(v_{r,t}(r_t, x_t; \phi) + g_r(t) s_{r,t}(r_t, x_t; \phi))\eta\Delta t\right]$
31:         $x_t \leftarrow x_t + \left[v_{x,t}(r_t, x_t; \phi) + g_x(t) s_{x,t}(r_t, x_t; \phi)\right]\eta\Delta t$
32:     else                                        ▷ At low noise we integrate the ODE instead
33:         /** Velocity prediction **/
34:         // Denoiser prediction
35:         $\hat{r}_1, \hat{x}_1 = D(r_t, x_t, t; \phi)$
36:         $v_{r,t}(r_t, x_t; \phi) = \log_{r_t} \hat{r}_1/(1-t)$
37:         $v_{x,t}(r_t, x_t; \phi) = (\hat{x}_1 - x_t)/(1-t)$
38:         /** Integration step **/
39:         $r_t \leftarrow \exp_{r_t}\left[v_{r,t}(r_t, x_t; \phi)\eta\Delta t\right]$
40:         $x_t \leftarrow x_t + v_{x,t}(r_t, x_t; \phi)\eta\Delta t$
41:     end if
42: end for
43: return $\hat{r}_1, \hat{x}_1$                       ▷ We return the final denoiser prediction
```

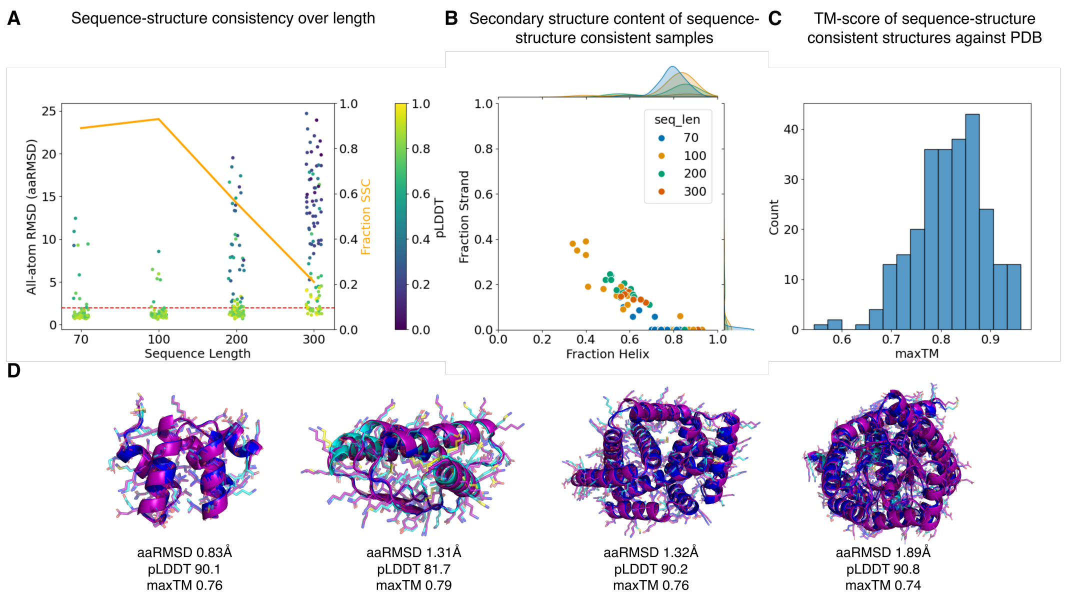


Figure G.1: Various methods of evaluating ProteinZen sample characteristics. (A) Sequence-structure consistency of all-atom structures over length. (B) Secondary structure content of sequence-structure consistent samples. (C) maxTM scores of SSC samples against PDB. (D) Examples of sequence-structure consistent samples. Purple is the generated structure and blue is the ESMFold prediction colored by pLDDT.

G Unconditional generation additional results

G.1 Metrics

We evaluate generation quality on three main criteria. The first is *sequence-structure consistency* (SSC), which we evaluate by predicting the structure of the generated sequence using ESMFold [37] and computing the all-atom RMSD (aaRMSD) against the generated structure. We use a threshold of aaRMSD < 2Å for a sample to be SSC. The second is *diversity* along both structure (DIV-str) and sequence (DIV-seq) axes, measured as the number of clusters formed when clustering with Foldseek [38] with a TM-threshold of 0.5 or MMSeqs2 [39] with default settings for structure and sequence, respectively. For both diversity metrics, we divide the number of clusters by the total number of SSC samples to yield the final metric. The third is *structural novelty* (NOV-str), where we query the PDB for the most structurally-similar hits using Foldseek and average the TM-scores of the maxTM hit per structure.

We also evaluate similar metrics for the backbone structures of generated samples. In this case, we swap the SSC metric for a designability metric (DES), where we use ProteinMPNN [26] to sample 8 sequences per backbone and predict structures for each using ESMFold. We define the minimum C_α RMSD between any of the predicted structures and the design structure as the self-consistent RMSD (scRMSD) and use a threshold of scRMSD < 2Å for a structure to be designable. The other two metrics DIV-str and NOV-str are evaluated on designable structures using the lowest scRMSD ESMFold prediction.

G.2 Sample characteristics

We evaluate other characteristics of ProteinZen samples in Figure G.1. Figure G.1A shows SSC as a function of sequence length. We note that ProteinZen performance drops off as protein length increases: this may be explained by the increased difficulty of designing larger proteins because the space of possible protein structures increases as sequence length increases, and making it more challenging to ensure the designed sequence specifies the design structure.

Figure G.1B shows the secondary structure distribution of SSC samples, depicting that overall the model is capable of generating structures with diverse secondary structures composition. When compared to the data distribution (Appendix C), we observe that samples are biased towards α-helices in content, a property observed in many existing generative models [4][5]. We note that β-sheets have much higher contact order (involve contacts which are local in space but distal in sequence) than α-helices, and the design of such structures is an area of interest for further investigation.

Figure G.1C depicts the maxTM distribution across SSC samples. All samples have maxTM > 0.5, signaling that ProteinZen has learned and largely recapitulates the known fold space of the PDB. There may be hints of generalization to novel fold space given the left tail end of lower maxTM scores (structures in Figure G.2), and improving this generalization is an avenue of future work.

Finally, Figure G.1D shows examples of sequence-structure consistent samples. We generally get good agreement between the generated structure and the predicted model, with the core more well constrained than the surface as expected. We note that we occasionally observe minor unphysicalities, such as clashes or overextended bond lengths, but that these occur in a small minority of samples and may be addressed by finetuning with physical constraints (e.g. clash losses or bond length losses) or inference-time steering [40].

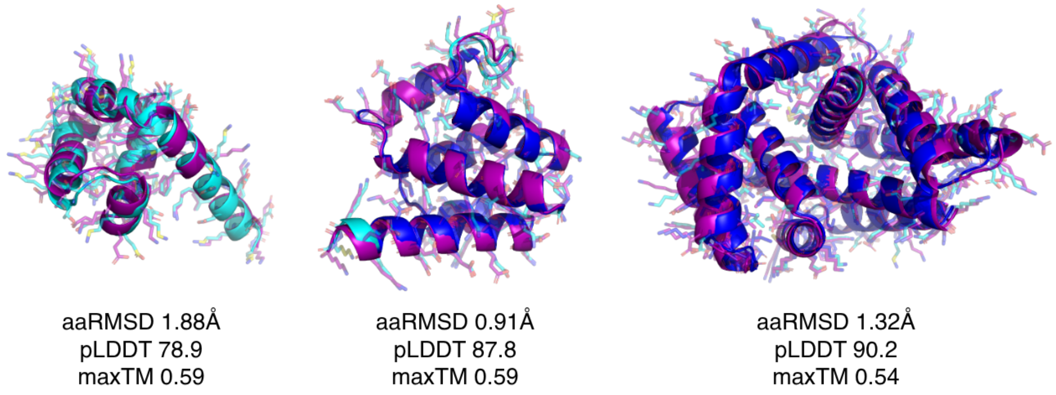


Figure G.2: Samples from unconditional monomer generation with pdbTM < 0.6

H Motif scaffolding additional results

H.1 Metrics

Our primary metric is *unique success rate* per task, which can intuitively be understood as the number of unique high quality scaffolds generated per set of design attempts. For one-shot motif scaffolding, we define a particular design to be a "success" for a particular task if the global heavy atom RMSD < 2Å, the motif heavy atom RMSD < 2Å, and the motif Cα RMSD < 1Å between the design structure and structure of the generated sequence predicted using ESMFold [37]. For models which do not impute the input all-atom motif structure, we further filter generated outputs by ensuring that there is no larger than 2Å heavy-atom RMSD pairwise between the motif structure in the design, the motif structure in the ESMFold prediction, and the input motif specification. To compute the unique success rate per task, we cluster task successes with Foldseek [38] with TM-threshold 0.5 and normalize by the total number of designs generated.

For motif scaffolding with sequence redesign, we use ProteinMPNN to redesign 8 sequences for the backbone scaffold (the motif sequence stays fixed). We then replace the global heavy atom RMSD criteria with a global backbone RMSD < 2Å threshold. When comparing against RFDiffusion [4], we graft the input motif structure into the proper location before computing any relevant metrics.

H.2 Benchmark details

Task specifications for the Protpardelle motif scaffolding challenge are provided in Appendix Table H.3, where 200 designs are generated per task listed. Under indexed motif scaffolding, the model is given access to both motif order and spacing, whereas under unindexed motif scaffolding only the motif residues are provided.

Task	Protpardelle-1c cc91		La-Proteina (indexed)		La-Proteina (unindexed)		ProteinZen (indexed)		ProteinZen (indexed)		# segments
	All	Unique	All	Unique	All	Unique	All	Unique	All	Unique	
1YCR	50	22	118	45	105	67	125	67	132	86	1
3IXT	2	2	39	4	53	6	31	12	23	9	1
4ZYP	4	2	3	1	29	3	26	8	18	7	1
5TPN	17	6	33	6	28	9	32	13	38	13	1
5TRV_short	0	0	1	1	0	0	1	1	1	1	1
5TRV_med	0	0	27	10	31	7	6	4	10	8	1
5TRV_long	0	0	48	18	59	28	15	12	12	11	1
5WN9	0	0	0	0	0	0	6	4	2	2	1
6E6R_short	5	3	23	15	46	20	22	16	20	17	1
6E6R_med	19	11	89	43	77	50	62	50	70	54	1
6E6R_long	11	11	80	46	93	69	71	58	69	59	1
7MRX_60	8	3	36	3	38	3	50	3	47	5	1
7MRX_85	13	5	23	3	64	6	54	9	47	5	1
7MRX_128	2	2	9	5	64	21	16	9	17	6	1
1PRW	148	13	167	18	115	9	139	36	50	17	2
2KL8	78	1	63	1	27	1	11	1	13	1	2
4JHW	0	0	0	0	0	0	0	0	0	0	2
5IUS	0	0	1	1	0	0	3	2	0	0	2
6VW1	12	1	28	1	37	1	45	2	56	9	2
1QJG	4	4	28	14	49	27	41	24	47	38	3
1QJG_NAT	10	8	39	15	83	51	39	22	58	43	3
5AOU	38	4	151	8	20	16	63	14	4	2	3
5YUI	0	0	3	2	8	8	1	1	1	1	3
7K4V	4	2	40	8	23	23	4	1	1	1	3
1BCF	195	6	144	7	90	6	175	9	138	22	4
5AOU_QUAD	14	3	81	5	50	25	68	11	4	2	4
Total	634	109	1274	280	1189	456	1106	389	878	419	

Table H.1: Total and unique number of successes per task on one-shot Protpardelle motif scaffolding benchmark.

We provide the raw task success metrics used to create Figure 2 in Table H.1 and Table H.2, where values are computed from samples generated using the published implementation.

We note that benchmark values for comparison models are slightly lower across the board when compared to reference literature [13][16], but that all designs were evaluated using a common evaluation pipeline so benchmark values are directly comparable.

Task	RFDiffusion		Protpardelle-1c cc91		La-Proteina (indexed)		La-Proteina (unindexed)		ProteinZen (indexed)		ProteinZen (indexed)		# segments
	All	Unique	All	Unique	All	Unique	All	Unique	All	Unique	All	Unique	
1YCR	171	75	134	55	175	70	182	107	198	112	200	121	1
3IXT	145	32	15	4	81	12	158	23	158	58	153	49	1
4ZYP	124	23	30	9	85	15	153	21	145	44	153	56	1
5TPN	168	42	56	30	79	19	92	30	171	60	157	84	1
5TRV_short	0	0	0	0	8	3	10	2	171	41	174	45	1
5TRV_med	4	4	0	0	85	19	112	32	152	56	183	65	1
5TRV_long	24	18	0	0	126	33	161	57	164	78	187	90	1
5WN9	1	1	0	0	2	2	2	2	40	21	36	21	1
6E6R_short	79	33	53	29	87	34	124	53	163	105	168	106	1
6E6R_med	160	60	69	38	153	79	171	110	174	126	184	127	1
6E6R_long	149	72	53	45	163	92	172	137	183	141	190	137	1
7MRX_60	41	9	55	12	124	8	129	4	123	18	121	20	1
7MRX_85	27	13	59	15	120	15	173	10	133	24	135	25	1
7MRX_128	15	11	16	14	68	24	120	43	114	30	113	34	1
1PRW	199	15	195	21	199	16	176	12	194	30	88	23	2
2KL8	180	1	164	1	185	1	19	1	178	1	54	1	2
4JHW	0	0	1	1	5	4	1	1	24	16	15	11	2
5IUS	14	2	2	2	8	2	0	0	133	22	67	26	2
6VW1	149	7	71	7	122	2	156	10	181	11	187	33	2
1QJG	102	62	42	29	123	40	139	61	155	73	171	130	3
1QJG_NAT	113	56	59	36	139	41	176	114	162	95	168	127	3
5AOU	44	34	98	9	197	10	61	53	131	42	36	8	3
5YUI	2	2	8	5	16	12	43	22	78	63	74	71	3
7K4V	40	36	16	3	125	21	68	66	51	17	54	27	3
1BCF	200	10	200	6	196	10	199	13	190	13	197	34	4
5AOU_QUAD	29	28	62	7	171	9	117	59	161	22	26	20	4
Total	2180	646	1458	378	2842	593	2914	1043	3727	1319	3291	1491	

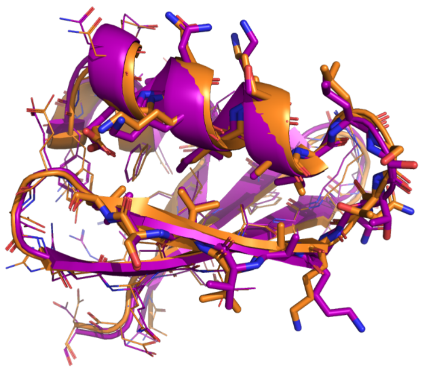
Table H.2: Total and unique number of successes per task on Protpardelle motif scaffolding benchmark with sequence redesign by ProteinMPNN.

Task	Minimum length	Maximum Length	Task contig
1BCF	96	152	8-15/A92-99/16-30/A123-130/16-30/A47-54/16-30/A18-25/8-15
1PRW	60	105	5-20/A16-35/10-25/A52-71/5-20
1QJG	53	103	10-20/A38/15-30/A14/15-30/A99/10-20
1QJG_NATIVE	115	135	10-20/A14/15-30/A38/50-70/A99/25-30
1YCR	40	100	10-40/B19-27/10-40
2KL8	79	79	A1-7/20/A28-79
3IXT	50	75	10-40/P254-277/10-40
4JHW	60	90	10-25/F196-212/15-30/F63-69/10-25
4ZYP	30	50	10-40/A422-436/10-40
5AOU	230	270	40-60/A1051/20-40/A2083/20-35/A2110/100-140
5AOU_QUAD	230	270	40-60/A1051/20-40/A2083/20-35/A2110/60-80/A2180/40-60
5IUS	57	142	0-30/A119-140/15-40/A63-82/0-30
5TPN	50	75	10-40/A163-181/10-40
5TRV_SHORT	56	56	0-35/A45-65/0-35
5TRV_MED	86	86	0-65/A45-65/0-65
5TRV_LONG	116	116	0-95/A45-65/0-95
5WN9	35	50	10-40/A170-189/10-40
5YUI	50	100	5-30/A93-97/5-20/A118-120/10-35/A198-200/10-30
6E6R_SHORT	48	48	0-35/A23-35/0-35
6E6R_MED	78	78	0-65/A23-35/0-65
6E6R_LONG	108	108	0-95/A23-35/0-95
6VW1	62	83	20-30/A24-42/4-10/A64-82/0-5
7K4V	280	320	40-50/A44/3-8/A50/70-85/A127/150-200
7MRX_60	60	60	0-38/B25-46/0-38
7MRX_85	85	85	0-63/B25-46/0-63
7MRX_128	128	128	0-122/B25-46/0-122

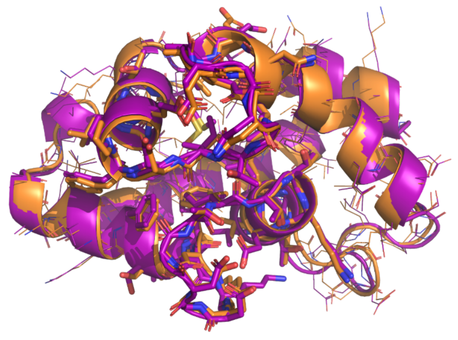
Table H.3: Protpardelle motif scaffolding challenge specification.

H.3 Scaffold characteristics

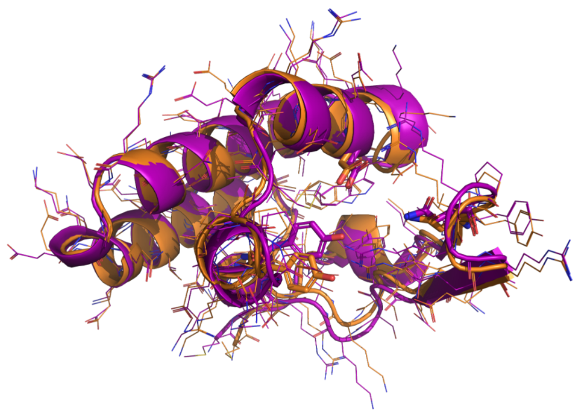
We provide example task successes for indexed motif scaffolding in Figure H.1 and unindexed motif scaffolding in Figure H.2. We also plot RMSD metrics for indexed motif scaffolding in Figures H.3, H.4 and for unindexed motif scaffolding in Figures H.5, H.6. Similar to under unconditional monomer generation, ProteinZen appears to have a preference for generating helical elements in scaffolds, but is both capable of scaffolding motifs with β-strands and also generating scaffolds with β-sheets in them.



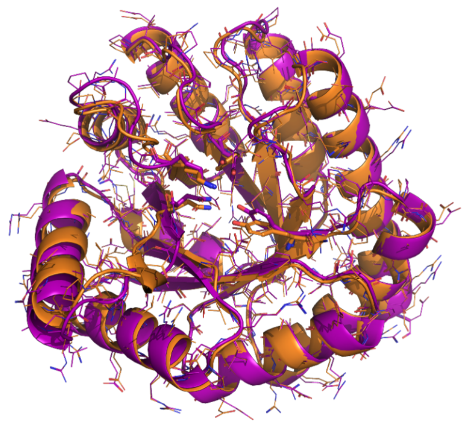
Task 5TPN
Global RMSD 1.14Å
Motif RMSD 1.14Å
pLDDT 86.3



Task 1PRW
Global RMSD 1.45Å
Motif RMSD 0.82Å
pLDDT 77.6



Task 1QJG
Global RMSD 1.18Å
Motif RMSD 1.10Å
pLDDT 83.2



Task 5AOU_QUAD
Global RMSD 1.24Å
Motif RMSD 1.17Å
pLDDT 92.4

Figure H.1: Examples of indexed motif scaffolding task successes. Purple structures are the design structure, and orange structures are the corresponding ESMFold prediction. Atomic detail is shown in wireframe, and motif residues are displayed using sticks.

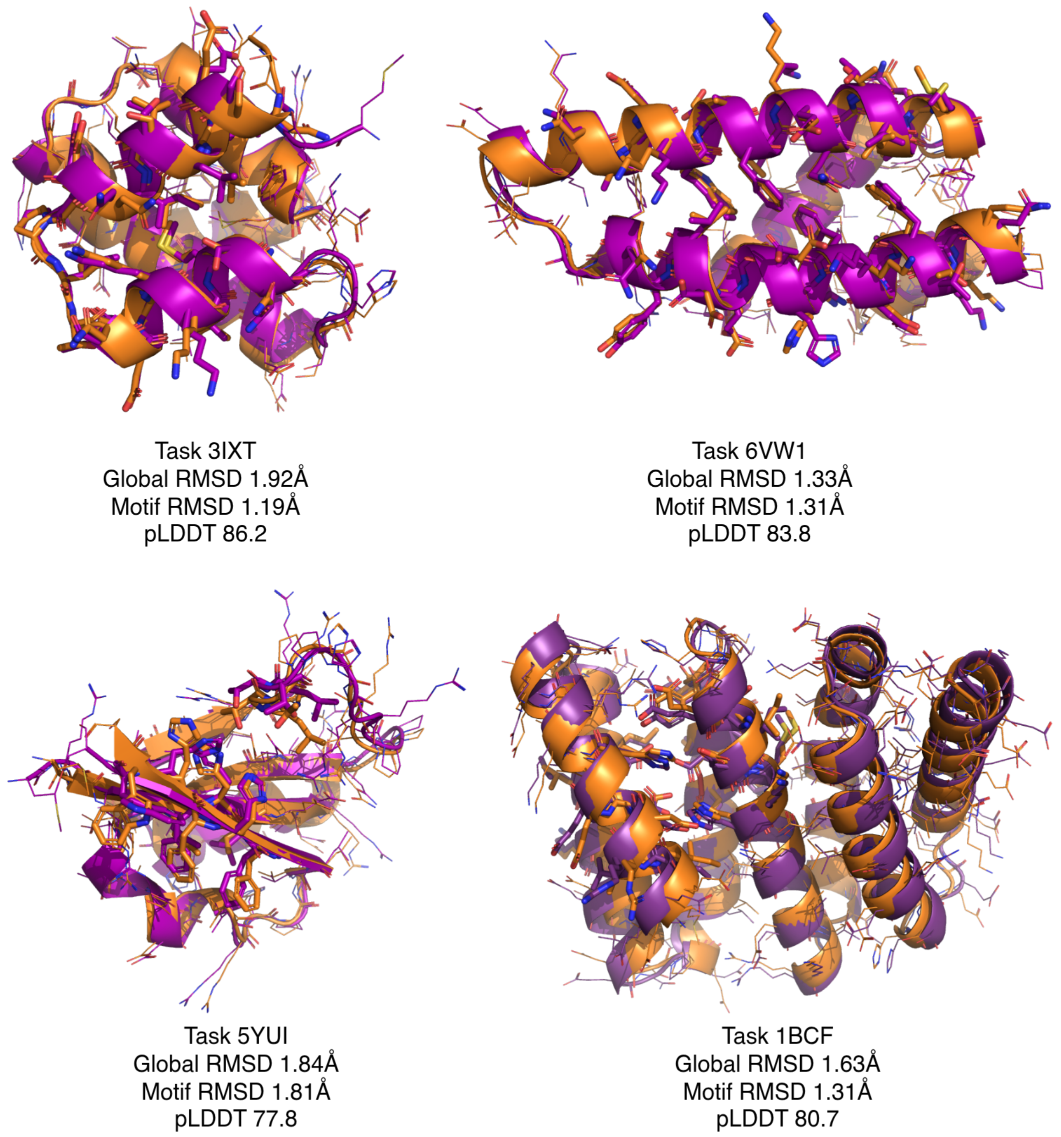


Figure H.2: Examples of unindexed motif scaffolding task successes. Purple structures are the design structure, and orange structures are the corresponding ESMFold prediction. Atomic detail is shown in wireframe, and motif residues are displayed using sticks.

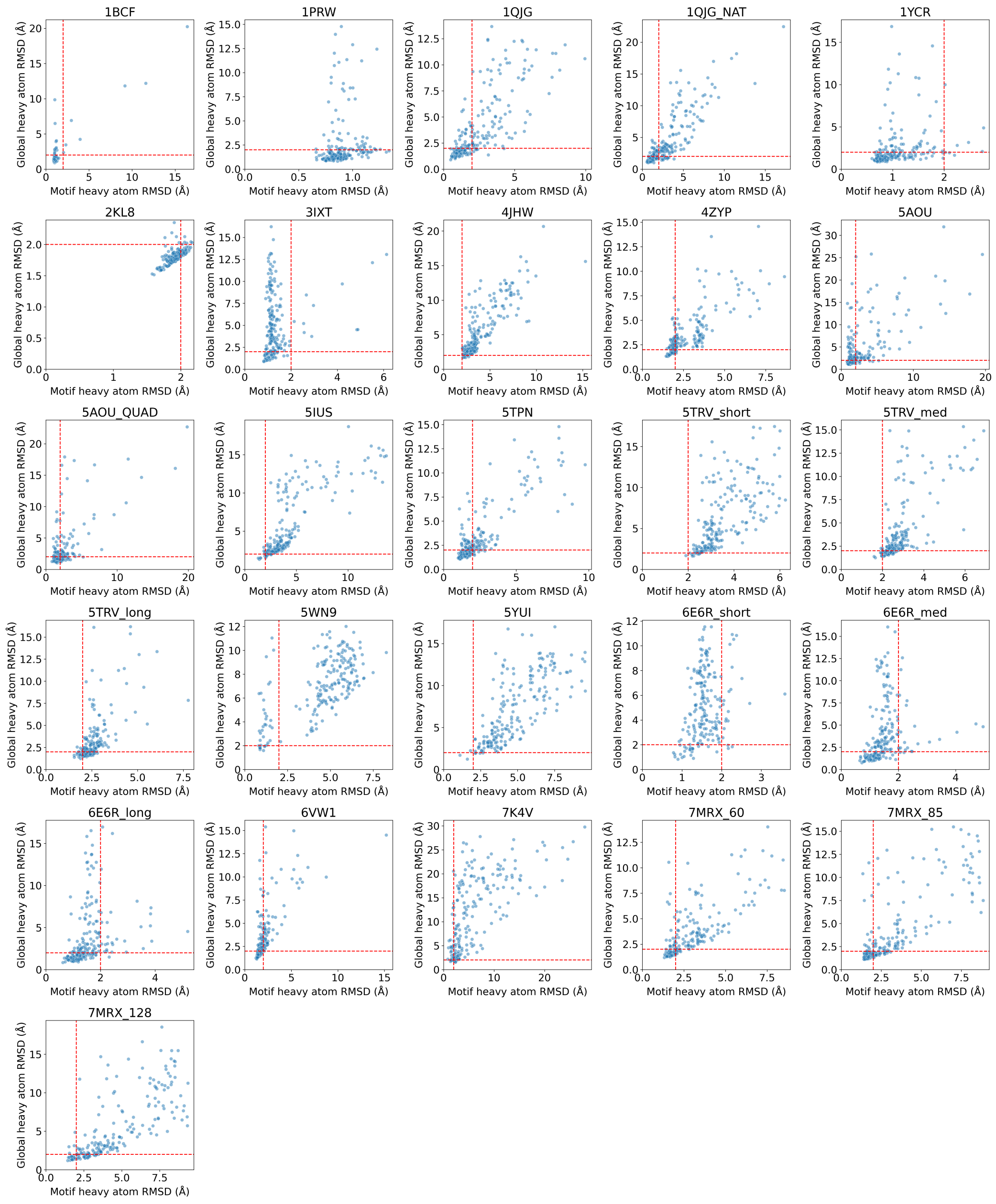


Figure H.3: Motif heavy atom RMSD vs global heavy atom RMSD for indexed motif scaffolding designs. Red lines mark motif heavy atom RMSD < 2Å and global heavy atom RMSD < 2Å.

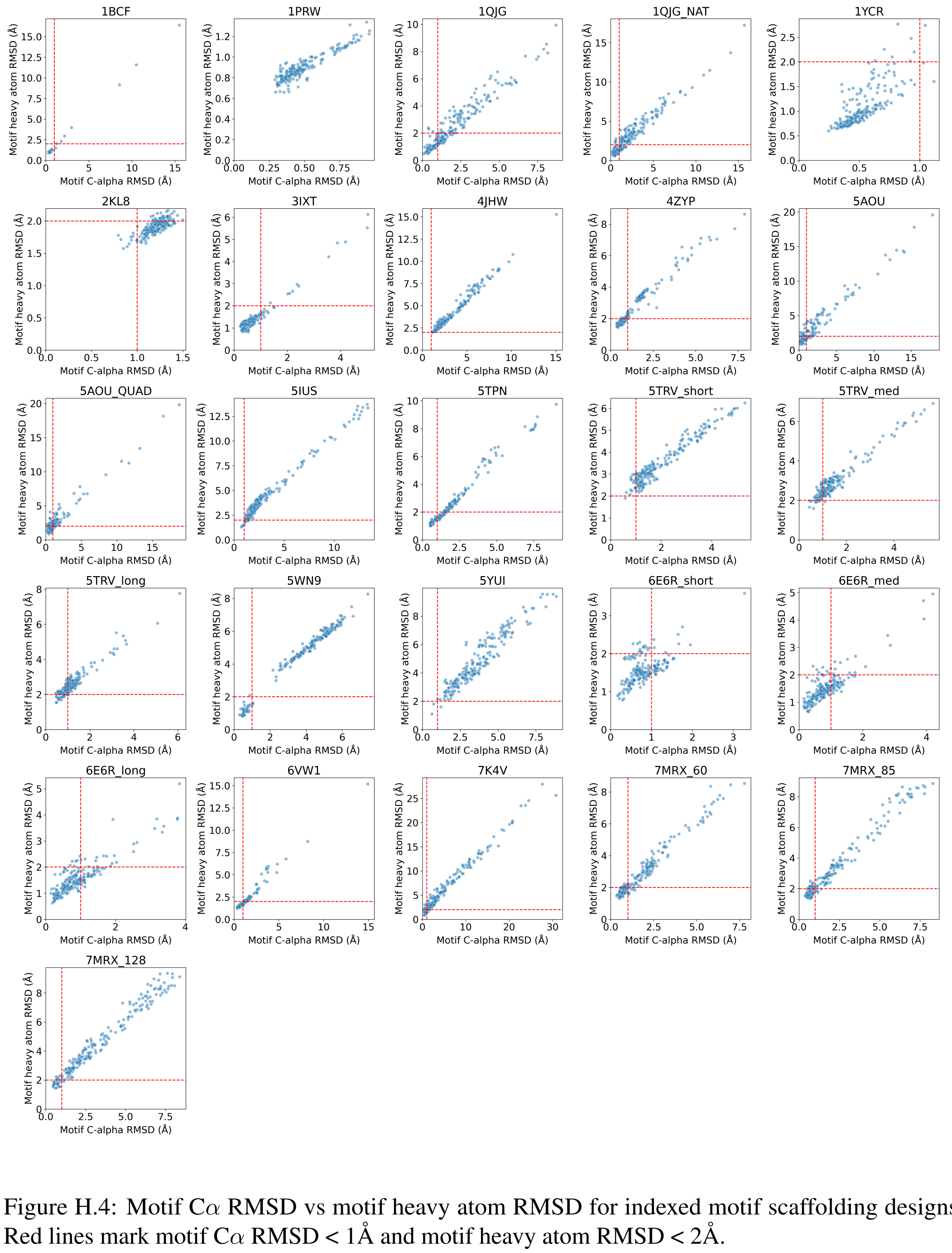


Figure H.4: Motif Cα RMSD vs motif heavy atom RMSD for indexed motif scaffolding designs. Red lines mark motif Cα RMSD < 1Å and motif heavy atom RMSD < 2Å.

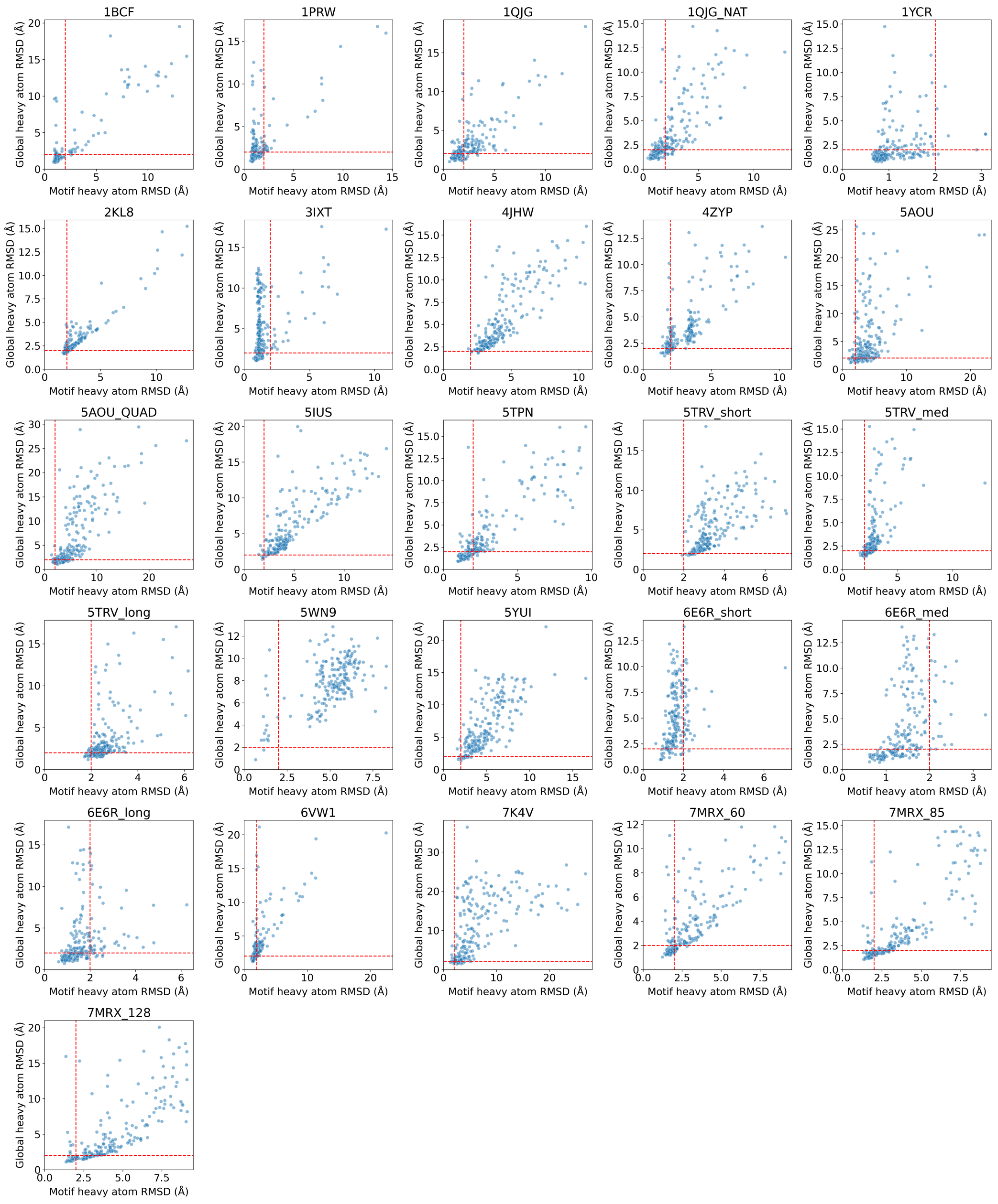


Figure H.5: Motif heavy atom RMSD vs global heavy atom RMSD for unindexed motif scaffolding designs. Red lines mark motif heavy atom RMSD < 2Å and global heavy atom RMSD < 2Å.

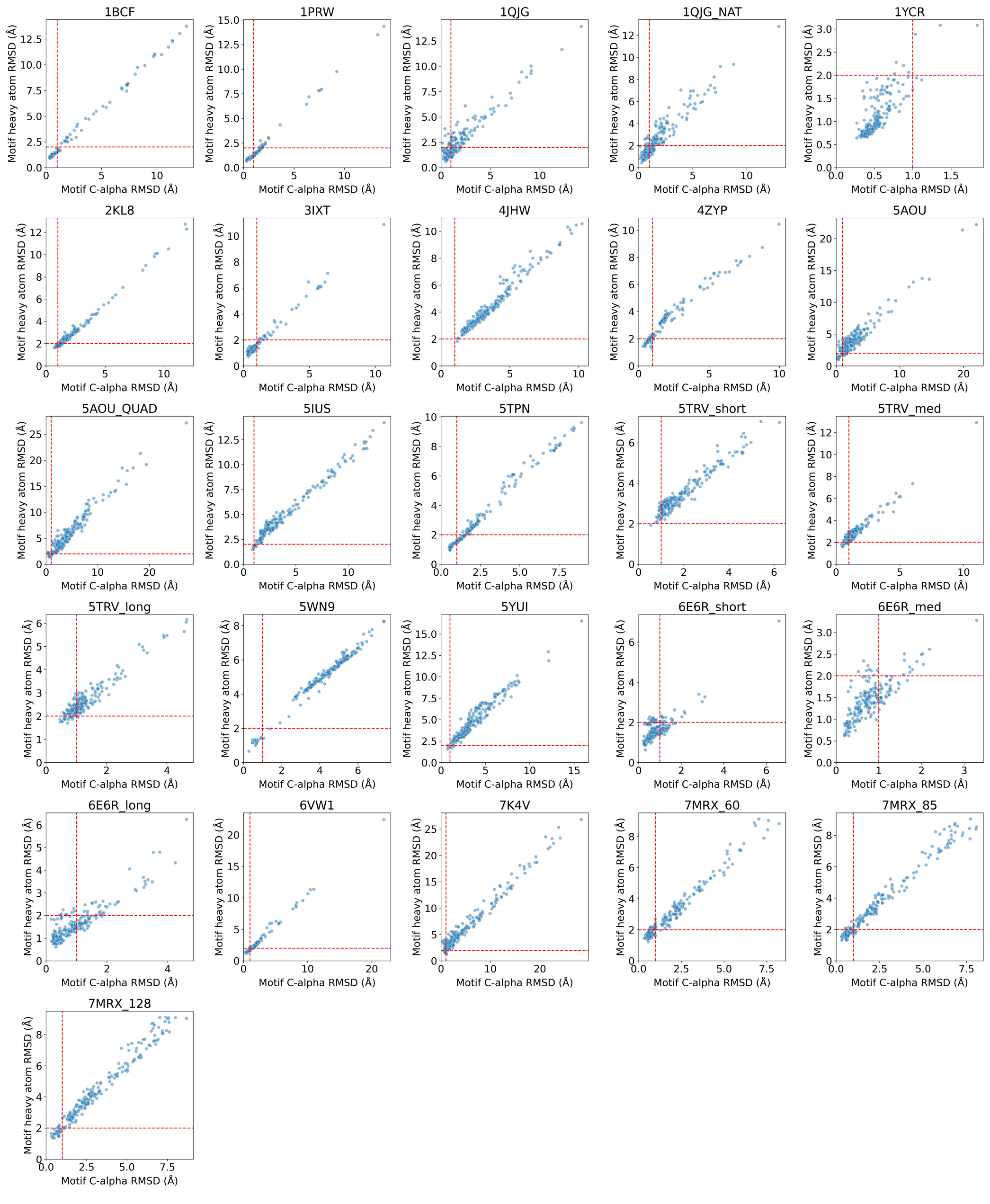


Figure H.6: Motif Cα RMSD vs motif heavy atom RMSD for unindexed motif scaffolding designs. Red lines mark motif Cα RMSD < 1Å and motif heavy atom RMSD < 2Å.