ProteinZen: combining latent and SE(3) flow matching for all-atom protein generation

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

Tanja Kortemme
UCSF
tanjakortemme@gmail.com

Abstract

De novo protein design has been greatly accelerated by the advent of generative models of protein structure. While more coarse-grain tasks such as backbone generation are increasingly possible, existing models still struggle on problems that require precise placements at the atomic scale, motivating the development of all-atom generative models. However, all-atom generative modeling remains challenging due to the complex interplay between discrete and continuous aspects of protein structure. In this work we propose a framework to capture this interplay by combining backbone generation methods using flow-matching on backbone frames with flow-matching in learned latent spaces. We present a prototype of our method ProteinZen that implements these ideas, and demonstrate promising initial results on the all-atom generation task, with 46% of samples being sequence-structure consistent at the atomic level while retaining competitive sample diversity and novelty.

1 Introduction

The ability to design proteins *de novo*, or without relying on existing starting points in nature, would open up possibilities to access functions unobserved in nature, as well as the ability to *a priori* embed engineering qualities into designs such as tunability, controllability, and modularity [1]. In recent years, generative modeling methods utilizing paradigms such as score-based diffusion [2] and flow matching [3] have drastically expanded our capabilities to design proteins with new structures *de novo* as well as functions such as protein-protein binders [4]. Despite these advances, targeting functional design remains challenging. We posit this difficulty is in part due to the lack of atomic-level reasoning in existing generative models. We know that atomic precision, both local and extended, is necessary for many design targets such as *de novo* enzymes [5][6] and protein binders to polar interfaces [7]. However, existing models such as RFDiffusion [8] and Chroma [9] only model backbones, and extensions such as RFDiffusion-AA [10] and LigandMPNN [11] start to introduce more atomic-awareness into models but still do not consider atomic structures in full detail. These shortcomings motivate the transition 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. Recent work has tackled this problem utilizing concepts of superposition, such as Protpardelle [12] and Pallatom [13], which can be thought of as modeling the joint distribution across all possible atomic configurations and collapsing it down to a single identity when necessary.

One downside of superposition-based methods is that they must keep track of many extra atoms per residue as overhead. They also require defining how both sequence and possible atomic configurations

Machine Learning for Structural Biology Workshop, NeurIPS 2024.

evolve during the generation process as well as potential covariation within such a process. An attractive alternate approach is to instead abstract fine-grain atomic details into latent representations via autoencoders and, utilizing techniques similar to those of latent diffusion [14], generate these latent quantities instead. This approach is appealing, in part, because it circumvents the complexities of modeling entangled discrete and continuous quantities during the generative stages of the model, relegating their determination to regression stages conditioned upon generated outputs. Futhermore, because we do not explicitly define the available information in the latent representation, co-generation of latent representations has the potential to improve generation quality by providing an avenue of propagating higher-order information through sampling trajectories.

In this work we explore this idea via preliminary work on ProteinZen, our model for all-atom generation utilizing a multi-modal flow-matching process over backbone frames as well as learned per-residue latent representations. We demonstrate promising prototype performance on the all-atom generation task and suggest future directions for this work.

2 Methods

2.1 Setup

In the following sections we largely focus on a single residue $T^{(i)}$ at position i, with a protein being an ordered sequence of such residues. We consider two representations for residue $T^{(i)}$. The first is a frames-based representation $(F^{(i)}, s^{(i)}, \{\chi^{(i,k)}\})$ composed of the residue backbone frame as defined in AlphaFold2 [15], sequence identity, and associated sidechain rotamer angles, respectively. The second is a purely atomistic representation $(\{a^{(i,j)}\}, \{h^{(i,j)}\})$ consisting of the coordinates and associated elements, respectively, of the all-atom representation of the residue. Note that both representations fully specify a residue in atomic detail and we can freely interconvert between the two forms. We also define $z^{(i)}$ as latent features associated with $T^{(i)}$ and $m^{(i)}$ as the residue's mask state. See Appendix A for formal definitions. When clear we drop the superscript for simplicity, in which case it is implied that we are looking at a single residue position i.

We treat the generation process in two stages: (1) the co-generation of protein backbone and per-residue latent features, and (2) the decoding of the latent features into atomic structures conditioned upon the backbones generated.

2.2 Protein generation as joint flow-matching over SE(3) and latent space

In the first stage, we seek to generate a backbone frame $F = (r, x) \in \text{SE}(3)$ and associated latent representation $z \in \mathbb{R}^c$ for each residue. For the backbone frames, we follow FrameFlow [16] and FoldFlow [17] in flow-matching over SE(3) by training flows on SO(3) and $\mathbb{R}^3$. We further treat the latent features by applying Euclidean flow-matching procedures [3]. More concretely, we construct the following interpolants indexed by $t \in [0, 1]$ from source distribution $T_0 = (r_0, x_0, z_0) \sim \rho_0$ (our noise distribution) to target distribution $T_1 = (r_1, x_1, z_1) \sim \rho_1$ (the distribution of all-atom protein structures):

$$r_t = \exp_{r_0}(t \log_{r_0}(r_1)) \qquad x_t = tx_0 + (1-t)x_1 \qquad z_t = tz_0 + (1-t)z_1$$
$$T_t = (r_t, x_t, z_t) \quad r_t \in \text{SO}(3), x_t \in \mathbb{R}^3, z_t \in \mathbb{R}^c$$

See Appendix B for details on the noise distribution. We formulate our flow-matching losses as denoising objectives and train a denoiser $\hat{r}_1, \hat{x}_1, \hat{z}_1 = f_\theta(T_t, t)$ to minimize the following losses:

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

$$\mathcal{L}_{latent} = \frac{1}{(1-t)^2} \|\hat{z}_1(T_t, t) - z_1\|^2$$

In total, the flow-matching loss becomes

$$\mathcal{L}_{fm} = \mathcal{L}_{rot} + \mathcal{L}_{trans} + \mathcal{L}_{latent} + 0.25\mathcal{L}_{\text{bb fine-grain}}\mathbb{I}[t > 0.25]$$

where $\mathcal{L}_{\text{bb fine-grain}}$ is a loss on fine-grain backbone structure specified in FrameDiff (see Appendix C for details) and $\mathbb{I}$ is the indicator function.

2.3 Atomic structure autoencoder: a hybrid VAE/MLM

In order to train our denoiser, we need a way of learning the latent target z_1. Inspired by latent diffusion [14], we utilize an autoencoder to encode atomic-resolution residues into a latent representation z_1 and decode the sequence identity and associated atomic coordinates from this latent. However, in this context a vanilla VAE [18] is prone to *sequence leakage*: the atomic properties of the residue specify its sequence identity, and because the amino acid alphabet is relatively low-dimensional in the context of deep learning, small latents and heavy regularization are necessary to prevent autoencoder collapse when recovering the residue identity (empirically we have observed collapse on latents as small as hidden dimension 8). Furthermore, we find that reconstruction is poor with such latents.

We mitigate this problem borrowing concepts from masked language modeling [19]. Define the autoencoder process with encoder E_θ and decoder D_θ as

$$\mu_z, \sigma_z = E_\theta(\{a^{(i,j)}\}, \{h^{(i,j)}\}, m^{(i)}) \qquad z_1 \sim \mathcal{N}(\mu_z, \sigma_z^2) \qquad \hat{p}(.), \hat{\mathcal{X}}(.) = D_\theta(F, z_1, t = 1)$$

where $\hat{p}$ describes a categorical probability distribution over the amino acid alphabet $\mathcal{A}$, and $\hat{\mathcal{X}}$ describes the sidechain torsion angles for each possible sequence identity in $\mathcal{A}$. We then read out the predicted residue identity and torsion angles as $\hat{s} = \text{argmax}_s \hat{p}(s)$ and $\hat{\chi} = \hat{\mathcal{X}}(\hat{s})$, respectively.

During training, we mask some proportion of residues, which involves removing all associated atomic information about the residue sidechain as well as specifying whether or not the residue is masked. On masked residues, we evaluate reconstruction losses for both the sequence and the atomic positionings, while on unmasked residues we evaluate only the atomic reconstruction loss given the ground truth residue identity. Our final autoencoder loss is

$$\mathcal{L}_{ae} = \mathcal{L}_{atomic} + \mathcal{L}_{torsion} + \mathcal{L}_{\text{smooth lddt}} + \mathcal{L}_{\text{masked cce}} + 0.001\mathcal{L}_{KL}$$

Definitions for each component loss term can be found in Appendix C.

2.4 Grounding the denoiser using "passthrough losses"

We find it beneficial to ensure the model is well calibrated in both latent and physical space, so during training we also pass the outputs of the denoiser directly into the latent decoder and minimize the reconstruction losses of the outputted structure against the ground truth structure. We call these "passthrough losses", which are largely the same as the reconstruction losses applied above but evaluated on decoder outputs when fed denoiser outputs rather than encoder outputs. More concretely,

$$\hat{p}_t(.), \hat{\mathcal{X}}_t(.) = D_\theta(\hat{F}, \hat{z}_1, t) \qquad \hat{s}_t = \text{argmax}_s \hat{p}_t(s) \qquad \hat{\chi}_t = \hat{\mathcal{X}}_t(\hat{s}_t)$$

We then evaluate the autoencoder loss components on these passthrough outputs with some minor changes as listed below (further details can be found in Appendix C).

$$\mathcal{L}_{pt} = \left(\frac{1}{(1-t)^2} \mathcal{L}^{pt}_{atomic} + \mathcal{L}^{pt}_{\text{smooth lddt}} \right) \mathbb{I}[t > 0.25] + \mathcal{L}^{pt}_{cce} + \mathcal{L}^{pt}_{torsion}$$

2.5 ProteinZen architecture

ProteinZen is a two-stage network which implements the above paradigm (Figure 1). We build the encoder using Tensor-Field Networks [20], the decoder out of modified IPMP layers [21], and the denoiser out of graph variants of IPA [15] in a method inspired by IPA Transformers [22]. Notably, ProteinZen is SE(3) equivariant and heavily utilizes spatial locality to both sparsify computation and impart physics-inspired implicit biases. Due to this sparsity, the memory complexity of ProteinZen is roughly linear in the number of residues. For more architecture details, see Appendix D.

2.6 Training

The full loss we optimize for is

$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}(0,1), T_0 \sim \rho_0, T_1 \sim \rho_1} \left[\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}^{(i)}_{fm} + \mathcal{L}^{(i)}_{ae} + \mathcal{L}^{(i)}_{pt} \right]$$

We train ProteinZen on AFDB512, a dataset we constructed from clustered PDB [23] monomers and high-confidence AlphaFold Database (AFDB) [24] Foldseek cluster representatives [25] for single chains of at most 512 residues. For dataset details and construction, see Appendix E.

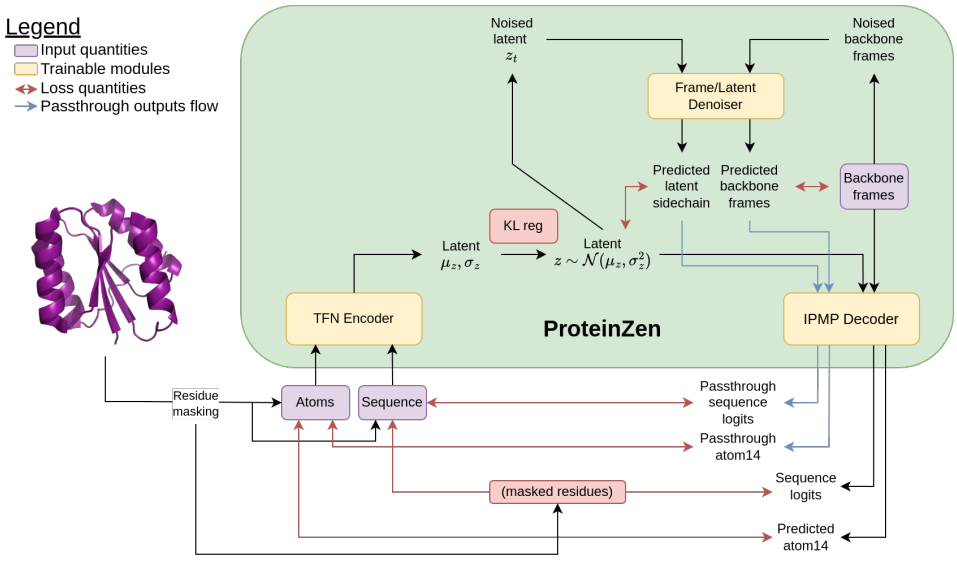

Figure 1: ProteinZen architecture, highlighting the joint frame/latent generation scheme and how all-atom structure is decoded.

Method	All-atom			PMPNN 8		
	SSC (↑)	DIV-str/DIV-seq (↑)	NOV-str (↓)	DES (↑)	DIV-str (↑)	NOV-str (↓)
ProteinGenerator [26]	0.11	0.39 / 0.76	0.85	0.81	0.27	0.81
Protpardelle [12]	0.17	0.27 / 0.97	0.82	0.88	0.14	0.79
Pallatom* [13]	0.68	0.61 / 0.92	0.75	0.88	0.67	0.73
ProteinZen	0.46	0.50 / 0.96	0.81	0.88	0.40	0.79

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.

3 Results

3.1 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 [27] 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 SSC samples with Foldseek [28] or MMSeqs2 [29] for structure and sequence, respectively, normalized by the total number of SSC samples. 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 (a metric for structural similarity, with TM > 0.5 indicative of two samples having the same fold [30]) of the maxTM hit per structure.

We also evaluate similar metrics for the backbone structures themselves to evaluate sequence-structure backbone consistency (see Appendix G) and designability of the backbone structures in general. In the designability case, we swap the SSC metric for a designability metric (DES), where we use ProteinMPNN [31] 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.

Following MultiFlow [32], we benchmark ProteinZen and relevant comparison models on 100 samples for each length 70, 100, 200, and 300 residues (Table 1). Excluding concurrent work

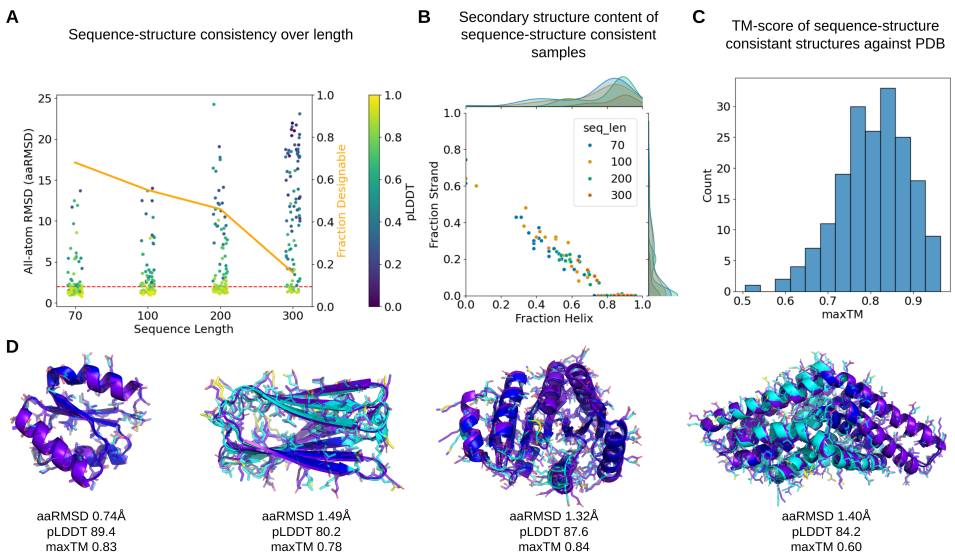

Figure 2: Various methods of evaluating ProteinZen sample characteristics. (A) Sequence-structure consistency of all-atom structures over length. (B) Secondary structure content of sequence-structure consistent samples. (C) maxTM scores of SSC samples against PDB. (D) Examples of sequence-structure consistent samples. Purple is the generated structure and blue is the ESMFold prediction colored by pLDDT.

Pallatom, ProteinZen outperforms previous models in all-atom generation, having a respectable 46% of samples being sequence-structure consistent. We also improve upon sample diversity and novelty, although Pallatom also has better performance on these metrics. We note that the models considered generally have much lower SSC than DES, signalling a disconnect between generated backbones and their associated sequence and atomic detail. Future work will investigate how to better integrate structure and sequence generation to mitigate this problem in ProteinZen.

3.2 Sample characteristics

We evaluate other characteristics of ProteinZen samples in Figure 2. Figure 2A shows SSC as a function of sequence length. We note that ProteinZen has much better performance on shorter length proteins, which could potentially be due to its architecture limiting long-range modeling capacities.

Figure 2B 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 E), we observe that samples are biased towards α-helices in content, a property observed in many existing generative models [8][9]. We note that β-sheets have much higher contact order (involve contacts which are local in space but distal in sequence) than α-helices, so ProteinZen's bias could also point towards investigating ProteinZen's long-range modeling capacities.

Figure 2C depicts the maxTM distribution across SSC samples. All samples have maxTM > 0.5, signalling 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 that some samples approach the maxTM = 0.5 threshold, and improving this generalization is an avenue of future work.

Finally, Figure 2D 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 clashes, which could potentially be mitigated with the implementation of a clash loss on decoder outputs.

4 Conclusions

We present preliminary work on our prototype of ProteinZen, a method which utilizes multi-modal flow-matching over backbone frames and latent representations to generate all-atom protein structures. It exhibits promising performance on the all-atom generation task, where 46% of samples are

sequence-structure consistent with atomic accuracy. Future work will look into improving the long-range modeling properties of ProteinZen, exploring the interplay between the autoencoder and denoiser modules, and extending the model to conditional generation tasks.

Acknowledgments and Disclosure of Funding

We would like to thank Deniz Akpinaroglu and Jason Yim for helpful discussions. A.J.L. is in part supported by a National Science Foundation Graduate Research Fellowship (NSF GRFP) and a UCSF Discovery Fellows Fellowship. T.K. is a Chan Zuckerberg Investigator.

References

[1] Tanja Kortemme. De novo protein design—from new structures to programmable functions. *Cell*, 187(3):526–544, February 2024.

[2] 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.

[3] 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.

[4] Matthias Gloegl, Aditya Krishnakumar, Robert Ragotte, Inna Goreshnik, Brian Coventry, Asim K Bera, Alex Kang, Emily Joyce, Green Ahn, Buwei Huang, Wei Yang, Wei Chen, Mariana Garcia Sanchez, Brian Koepnick, and David Baker. Target-conditioned diffusion generates potent tnfr superfamily antagonists and agonists. *bioRxiv*, 2024.

[5] 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.

[6] 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.

[7] Scott E. Boyken, Zibo Chen, Benjamin Groves, Robert A. Langan, Gustav Oberdorfer, Alex Ford, Jason M. Gilmore, Chunfu Xu, Frank DiMaio, Jose Henrique Pereira, Banumathi Sankaran, Georg Seelig, Peter H. Zwart, and David Baker. De novo design of protein homo-oligomers with modular hydrogen-bond network–mediated specificity. *Science*, 352(6286):680–687, 2016.

[8] 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.

[9] 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.

[10] Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S. Morey-Burrows, Ivan Anishchenko, Ian R. Humphreys, Ryan McHugh, Dionne Vafeados, Xinting Li, George A. Sutherland, Andrew Hitchcock, C. Neil Hunter, Alex Kang, Evans Brackenbrough, Asim K. Bera, Minkyung Baek, Frank DiMaio, and David Baker. Generalized biomolecular modeling and design with rosettafold all-atom. *Science*, 384(6693):eadl2528, 2024.

[11] 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.

[12] 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.

[13] 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.

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[15] 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.

[16] 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.

[17] 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.

[18] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[20] Nathaniel Thomas, Tess E. Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018.

[21] Nicholas Z. Randolph and Brian Kuhlman. Invariant point message passing for protein side chain packing. *bioRxiv*, 2023.

[22] 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.

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

[24] 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.

[25] 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.

[26] Sidney Lyayuga Lisanza, Jake Merle Gershon, Sam Tipps, Lucas Arnoldt, Samuel Hendel, Jeremiah Nelson Sims, Xinting Li, and David Baker. Joint generation of protein sequence and structure with rosettafold sequence space diffusion. *bioRxiv*, 2023.

[27] 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.

[28] 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.

[29] 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.

[30] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, October 2004.

[31] 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.

[32] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design, 2024.

[33] 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.

[34] Greg Landrum, Paolo Tosco, Brian Kelley, Ric, David Cosgrove, sriniker, Riccardo Vianello, gedeck, NadineSchneider, Gareth Jones, Eisuke Kawashima, Dan Nealschneider, Andrew Dalke, Brian Cole, Matt Swain, Samo Turk, Aleksandr Savelev, Alain Vaucher, Maciej Wójcikowski, Ichiru Take, Vincent F. Scalfani, Daniel Probst, Kazuya Ujihara, Rachel Walker, guillaume godin, Axel Pahl, Juuso Lehtivarjo, Francois Berenger, strets123, and jasondbiggs. rdkit/rdkit: 2023_09_6 (q3 2023) release, 2024.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[36] 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.

[37] 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.

Appendix

A Setup

In this work we largely consider quantities in the context of a single residue $T^{(i)}$ at position i, with a protein of N residues being an ordered sequence $(T^{(1)}, ..., T^{(N)})$. We formally define the quantities associated with residue $T^{(i)}$ as follows:

- $s^{(i)} \in \mathcal{A}$ is the identity of the residue i within amino acid alphabet $\mathcal{A}$.
- $a^{(i,j)} \in \mathbb{R}^3$ is the coordinate of the jth atom in residue i in units of nm.
- $h^{(i,j)}$ is the element of the jth atom in residue i.
- $F^{(i)} = (r^{(i)}, x^{(i)}) \in \text{SO}(3) \times \mathbb{R}^3$ is the backbone frame of residue i as define in AlphaFold2 [15]. x is in units of nm.
- $\{\chi^{(i,k)} | \chi^{(i,k)} \in [0, 2\pi)\}$ is the set of sidechain torsions of the residue i.
- $z^{(i)} \in \mathbb{R}^c$ is the latent representation associated with residue i (in this work we use $c = 32$).
- $m^{(i)} \in \{0, 1\}$ specifies whether residue i is masked (1) or not (0).

B Noise distribution

Following FrameFlow [16], 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 Gaussian with $\sigma = 1$nm and align this noise to the ground truth data using the Kabsch algorithm. For the latent noise we sample from a standard Gaussian with no data-dependent coupling.

During sampling we sample from the uniform distribution of SO(3) for rotations with the same priors as training for the translation and latent components.

C Loss term additional details

C.1 Flow-matching terms

Following FrameDiff [22] and FoldFlow [17], we find that a separable variant of $\mathcal{L}_{rot}$ leads to most stable training. This separable loss evaluates an MSE loss on the angle and axis components of the predicted rotation vector field separately, with the angle term scaled by 0.5.

We additionally include an auxiliary loss on fine-grain backbone atomic detail reported in FrameDiff [22]:

$$\begin{aligned}\mathcal{L}_{\text{bb fine-grain}} = &\frac{1}{|\mathcal{B}|} \sum_{b^{(i,j)} \in \mathcal{B}} \|\hat{b}^{(i,j)} - b^{(i,j)}\|^2 \\ &+ \frac{1}{Z_{bb}} \sum_{b^{(i_1,j_1)}, b^{(i_2,j_2)} \in \mathcal{B}} \mathbb{I}[d^{(i_1,j_1,i_2,j_2)} < 6] \|\hat{d}^{(i_1,j_1,i_2,j_2)} - d^{(i_1,j_1,i_2,j_2)}\|^2\end{aligned}$$

where $\mathcal{B}$ is the set of all backbone atoms, N is the number of residues, and

$$\begin{aligned}Z_{bb} &= \left(\sum_{b^{(i_1,j_1)}, b^{(i_2,j_2)} \in \mathcal{B}} \mathbb{I}[d^{(i_1,j_1,i_2,j_2)} < 6] \right) - N \\ d^{(i_1,j_1,i_2,j_2)} &= 10 \|a^{(i_1,j_1)} - a^{(i_2,j_2)}\| \\ \hat{d}^{(i_1,j_1,i_2,j_2)} &= 10 \|\hat{a}^{(i_1,j_1)} - \hat{a}^{(i_2,j_2)}\|\end{aligned}$$

C.2 Sidechain reconstruction terms

For sidechain recovery, we evaluate the following losses when teacher-forcing with the ground-truth sequence (e.g. $\hat{\chi} = \hat{\mathcal{X}}(s)$):

$$\mathcal{L}_{atomic} = 10 \sum_{(i,j)} \|\hat{a}^{(i,j)} - a^{(i,j)}\|^2 \qquad \mathcal{L}_{torsion} = \sum_{(i,k)} \left(2 - 2\cos\left(\hat{\chi}^{(i,k)} - \chi^{(i,k)}\right)\right)$$

In words, this corresponds to the MSE loss on atomic coordinates (units of Å) and MSE loss on sidechain rotamer angles when represented as unit vectors, respectively. We also utilize the smooth LDDT loss $\mathcal{L}_{\text{smooth lddt}}$ introduced in AlphaFold3 [33].

$$\mathcal{L}_{\text{smooth lddt}} = 1 - \frac{1}{Z_{aa}} \sum_{a^{(i_1,j_1)}, a^{(i_2,j_2)}} \frac{1}{4} \mathbb{I}[d^{(i_1,j_1,i_2,j_2)} < 15](\sigma(0.5 - d^{(i_1,j_1,i_2,j_2)}) + \sigma(1 - d^{(i_1,j_1,i_2,j_2)}) + \sigma(2 - d^{(i_1,j_1,i_2,j_2)}) + \sigma(4 - d^{(i_1,j_1,i_2,j_2)}))$$

where $\sigma(x)$ is the standard logistic function and

$$Z_{aa} = \sum_{a^{(i_1,j_1)}, a^{(i_2,j_2)}} \mathbb{I}[d^{(i_1,j_1,i_2,j_2)} < 15]$$

For sequence recovery we use

$$\mathcal{L}_{cce} = -\mathbb{I}[m = 1] \sum_{\alpha \in \mathcal{A}} \hat{p}(\alpha) \ln(\hat{p}(\alpha))$$

which is the cross-entropy loss applied to masked residues only. Finally, we also apply a KL-divergence loss term on the latent space

$$\mathcal{L}_{KL} = D_{KL}(\mathcal{N}(\mu_z, \sigma_z^2) || \mathcal{N}(0, I))$$

C.3 Passthrough reconstruction terms

We compute passthrough losses by evaluating the autoencoder loss components on passthrough outputs with a few modifications:

- We scale $\mathcal{L}^{pt}_{atomic}$ by $1/(1-t)^2$, which upweights the atomic-scale loss at times closer to $t = 1$.
- We evaluate the cross entropy loss $\mathcal{L}^{pt}_{cce}$ at all positions regardless of whether the ground truth latent was derived from a masked residue or not.
- We only apply $\mathcal{L}^{pt}_{atomic}$ and $\mathcal{L}^{pt}_{\text{smooth lddt}}$ at times $t > 0.25$.

D ProteinZen architecture details

D.1 Encoder architecture

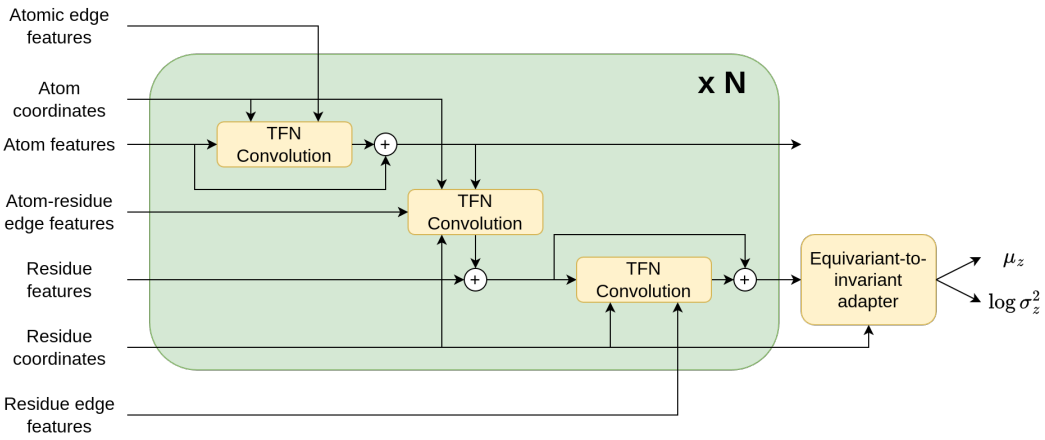

Figure 3: ProteinZen encoder architecture

ProteinZen's encoder is a two-track GNN which operates at both atomic- and residue-level resolutions (Figure 3). Inspired by Tensor Field Networks (TFN) [20], we encode atomic and residue features in the basis of the irreducible representations of SO(3), utilizing features of at most order $l = 1$ and of both even and odd parity.

Per-atom we generate initial features from the following set of atomic properties:

- Element represented as a one-hot vector of the element's row and column in the periodic table.
- Chirality as defined by RDKit [34].
- Atom hybridization.
- Number of bonds to heavy-atoms.
- Number of implicit hydrogens.
- Formal charge.
- Whether the atom is in an aromatic environment.
- Whether the atom is in a ring of size [3,4,5,6,8].
- How many rings the atom is in.

Per-residue we initialize the residue features as

- The sinusoidal embedding of the residue index as used in the original Transformer [35].
- A one-hot encoding of the sequence identity.

The masking process involves replacing the masked residue with an idealized alanine. For each atom we specify whether or not it's parent residue is masked, and we also add a similar masking feature to the residue itself.

At the atom level we construct two sets of graphs: a radius-based graph which connects an atom to all other atoms within 5Å, and a bond-based graph which represents the bond connectivity of all atoms. The radius graph edges are initialized to interatomic distances encoded using a Gaussian radial basis. The bond graph edges are generated with the following set of additional bond properties:

- The bond order, both as a one-hot vector and as a float.
- Whether the bond is part of an aromatic environment.

- Whether the bond is part of a conjugated system.

At the residue level we construct a k-nearest neighbors graph with $k = 30$, with the edge features initialized to

- The signed sequence distance encoded via a sinusoidal embedding.
- The physical distance encoded using a Gaussian radial basis.
- The residue-level features of both participant residues.

At the atomic level and per encoder layer, TFN convolutions are done separately across the bond and radius graphs and additively combined with the input embeddings via a residual connection. Atomic features are then aggregated into residue-level embeddings to update the residue-level graph. Another TFN convolution is performed at the residue level as well.

After all layers of the encoder, we have to convert the residue-level features from an SE(3)-equivariant basis to an invariant basis, as the rest of ProteinZen uses backbone frames which hold SE(3)-invariant features. To do this, we use a final adapter layer which facilitates the conversion. This is done by compute the Wigner-D matrix which rotates the residue features into the frame of the residue's backbone frame, then feeds all the rotated features into a fully-connected network. We then use a final linear layer to produce both μ_z and σ_z.

D.2 Decoder architecture

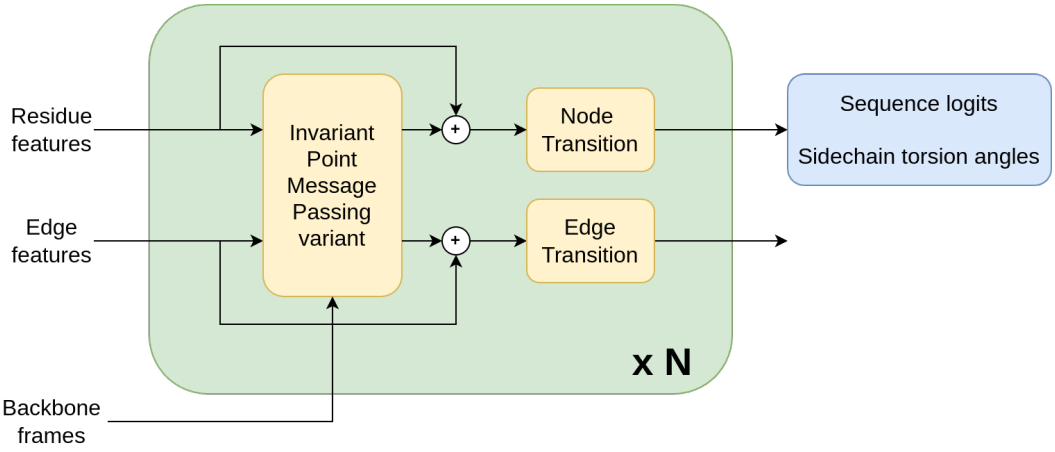

Figure 4: ProteinZen decoder architecture

ProteinZen's decoder network is composed of IPMP layers [21], which uses the inputted latent embedding and timestep t (which is set to 1 when running on encoder inputs) as residue-level features and, when combined with input backbone frames, predicts per-residue sequence logits and sidechain torsion angles per possible sequence identity. Initial edge features are derived from backbone atom-backbone atom (including virtual C_β) pairwise distances between two residues lifted to a Gaussian radial basis using a small MLP. We largely use the IPMP algorithm as described in PIPPack [21], but change the message-passing inputs use all-by-all pairwise invariant point distances rather than per-channel pairwise invariant point distances.

D.3 Denoiser architecture

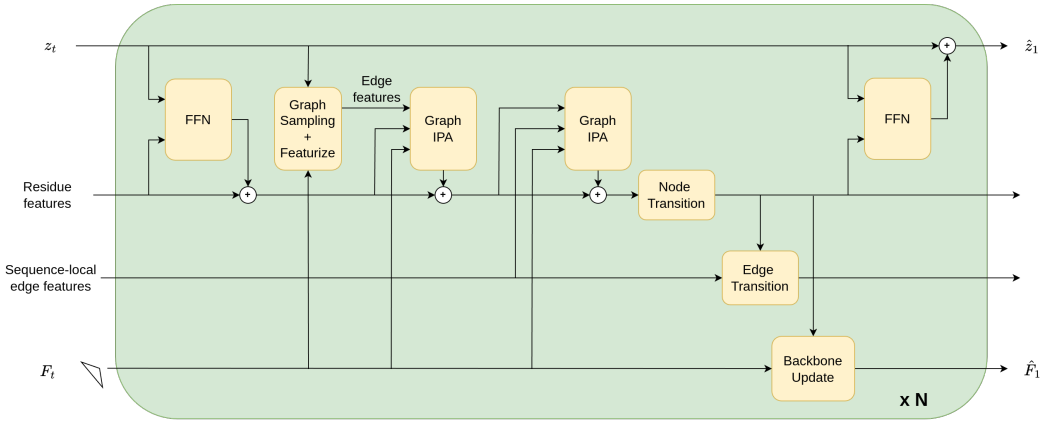

Figure 5: ProteinZen denoiser

ProteinZen's denoiser network is largely inspired by IPA Transformers [22], but on a spatially-sparcified dynamic graph in a method inspired by Chroma [9]. In brief, the network retains a fixed sequence-local graph which connects each residue to each of the 5 residues upstream and downstream of it in sequence. At each layer, we also sample a graph based on the current denoising state of the frames, which connects the 30 k-nearest neighbors in space as well as 60 additional long-range connections sampled without replacement with probability proportional to inverse cubic distance. Following Chroma, we use the Gumbel-Top k trick to compute these long-range connections. On each graph we run a graph-variant of IPA [15] to update the residue embeddings, and these updates are then propagated to both the latent and the frame via their respective update functions. We also utilize self-conditioning [36], which is implemented by featurizing both the sequence-local and spatially-sampled graphs on both the current denoising state as well as the self-conditioning input. When no self-conditioning input is provided, we zero out these inputs.

E Dataset construction

We construct AFDB512 using two different data sources: the PDB [23] and the AlphaFold predicted structures database [24]. For PDB structures, we adapt the strategy from FrameDiff [22] and select structures which meet the following criteria:

- X-ray structure or EM structure with resolution < 5.0 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 [37]

We cluster these structures based on the 95% sequence identity cluster assignments specified by the PDB. This results in 23154 structures total, with 7793 structure clusters.

For AFDB structures, we adapt strategies from Protpardelle [12] and Pallatom [13] 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 [37].
- 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 $2.24 \times N^{0.392}$ Å where N is the number of residues. This is inspired by the empirical radius of gyration scaling law observed for globular proteins.
- Is an AFDB Foldseek cluster representative [25].

This results in 80064 structures total.

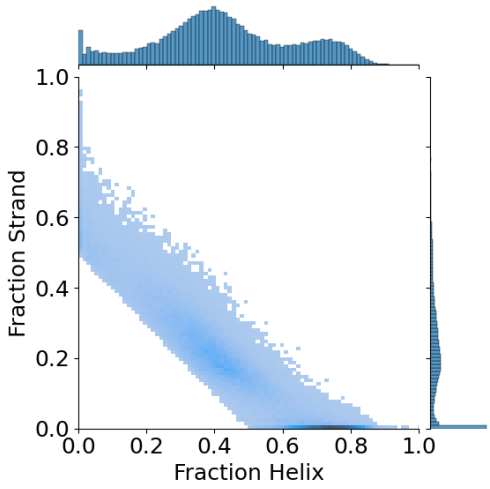

Figure 6: AFDB512 secondary structure composition

F Training details

During training we use the following masking scheme:

- 25% of the time, we mask all residues
- 25% of the time, we mask no residues
- 50% of the time, we sample a masking threshold from $\mathcal{U}(0, 1)$ and mask that proportion of residues by sampling from $\mathcal{U}(0, 1)$ per-residue and masking those that fall below the masking threshold.

Per epoch, we sample from both the PDB monomer clusters and the AFDB structures such that the ratio of real-to-synthetic structures is 1:3. We train on 4 NVIDIA A40 GPUs for ~8 days using a batch size of 6000 residues. We use the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ and a learning rate of 0.0001.

G Sequence-structure backbone consistency

In Table 2 we add results for benchmarked models on sequence-structure backbone consistency, which is performed under the same regime as the all-atom generation task but the SSC metric is computed using only backbone atoms (which we have renamed as SSC-bb). We also include Multiflow [32] as a benchmark model, which generates sequence and backbone structure but no sidechain conformations. We note that Multiflow is trained on synthetic data which consisting of protein backbones with ProteinMPNN-redesigned sequences which are SSC-bb by ESMFold, and hence such benchmark values may not be directly comparable to other methods. Overall, we find that metrics under the co-design evaluation are generally more similar to metrics from all-atom evaluation than from backbone evaluation, which suggests that existing methods' performance bottleneck may

Method	All-atom			Seq/struct Co-design			PMPNN 8		
	SSC (↑)	DIV-str/DIV-seq (↑)	NOV-str (↓)	SSC-bb (↑)	DIV-str/DIV-seq (↑)	NOV-str (↓)	DES (↑)	DIV-str (↑)	NOV-str (↓)
Multiflow[1] [32]	-	-	-	0.87	0.43 / 0.87	0.80	0.99	0.40	0.79
ProteinGenerator [26]	0.11	0.39 / 0.76	0.85	0.20	0.32 / 0.77	0.85	0.81	0.27	0.81
Protpardelle [12]	0.17	0.27 / 0.97	0.82	0.59	0.14 / 0.70	0.80	0.88	0.14	0.79
Pallatom[2] [13]	0.68	0.61 / 0.92	0.75	0.70	0.60 / 0.81	0.75	0.88	0.67	0.73
ProteinZen	0.46	0.50 / 0.96	0.81	0.54	0.47 / 1.00	0.81	0.88	0.40	0.79

Table 2: Sample quality metrics for ProteinZen and relevant comparison models. [1]Multiflow co-generates sequence and backbone structure without sidechain conformations, and is trained on synthetic data generated using ProteinMPNN and filtered using ESMFold. [2]Pallatom was trained on proteins of length at most 128, so the reported performance is partially out-of-distribution.

lie in designing good sequences rather than precise sidechain placement. Protpardelle seems to be an exception to this trend, as its SSC-bb is much higher than its SSC yet its diversity is much lower, suggesting that Protpardelle's performance may be skewed by generating low-diversity but high-designability samples.