OPHIUCHUS: SCALABLE MODELING OF PROTEIN STRUCTURES THROUGH HIERARCHICAL COARSE-GRAINING SO(3)-EQUIVARIANT AUTOENCODERS

Allan dos Santos Costa
Center for Bits and Atoms
MIT Media Lab, Molecular Machines
allanc@mit.edu

Ilan Mitnikov
Center for Bits and Atoms
MIT Media Lab, Molecular Machines

Mario Geiger
Atomic Architects
MIT Research Laboratory of Electronics

Manvitha Ponnapati
Center for Bits and Atoms
MIT Media Lab, Molecular Machines

Tess Smidt
Atomic Architects
MIT Research Laboratory of Electronics

Joseph Jacobson
Center for Bits and Atoms
MIT Media Lab, Molecular Machines

ABSTRACT

Three-dimensional native states of natural proteins display recurring and hierarchical patterns. Yet, traditional graph-based modeling of protein structures is often limited to operate within a single fine-grained resolution, and lacks hourglass neural architectures to learn those high-level building blocks. We narrow this gap by introducing Ophiuchus, an SO(3)-equivariant coarse-graining model that efficiently operates on all-atom protein structures. Our model departs from current approaches that employ graph modeling, instead focusing on local convolutional coarsening to model sequence-motif interactions with efficient time complexity in protein length. We measure the reconstruction capabilities of Ophiuchus across different compression rates, and compare it to existing models. We examine the learned latent space and demonstrate its utility through conformational interpolation. Finally, we leverage denoising diffusion probabilistic models (DDPM) in the latent space to efficiently sample protein structures. Our experiments demonstrate Ophiuchus to be a scalable basis for efficient protein modeling and generation.

1 INTRODUCTION

Proteins form the basis of all biological processes and understanding them is critical to biological discovery, medical research and drug development. Their three-dimensional structures often display modular organization across multiple scales, making them promising candidates for modeling in motif-based design spaces [Bystroff & Baker (1998); Mackenzie & Grigoryan (2017); Swanson et al. (2022)]. Harnessing these higher-level building blocks is of great relevance to the investigation of the mechanisms behind protein evolution, dynamics and folding [Mackenzie et al. (2016)], and may be instrumental in enabling more efficient computation on protein structural data through coarse latent variable modeling, such as in simulation through coarse-grained molecular dynamics [Kmiecik et al. (2016)].

Recent developments in deep learning architectures applied to protein sequences and structures demonstrate the remarkable capabilities of neural models in the domain of protein modeling and design [Jumper et al. (2021); Baek et al. (2021b); Ingraham et al. (2023); Watson et al. (2022)]. Still, current state-of-the-art architectures lack the structure and mechanisms to directly learn and operate on modular protein blocks. To fill this gap, we introduce Ophiuchus, a deep SO(3)-equivariant model

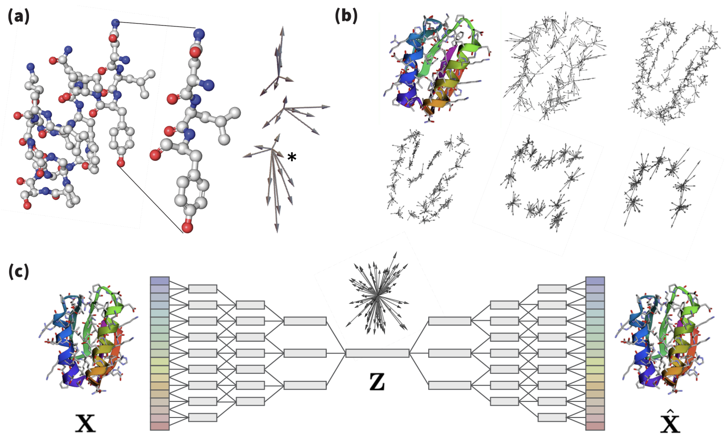

Figure 1: **Coarsening a Three-Dimensional Sequence**. **(a)** Each residue is represented with its $\mathbf{C}_\alpha$ atom position and geometric features that encode its label and the positions of its other atoms; $(*)$ atoms with ordering ambuiguity are encoded through a permutation-invariant geometric feature. **(b)** Our proposed model uses roto-translation equivariant convolutions to coarsen positions and geometric features. **(c)** We train deep autoencoders to reconstruct all-atom protein structures directly in three-dimensions.

that captures joint encodings of sequence-structure motifs of all-atom protein structures. Our model is a novel autoencoder that uses one-dimensional sequence convolutions on geometric features to learn coarsened representations of proteins. Ophiuchus outperforms existing SO(3)-equivariant autoencoders [Fu et al. (2023)] on the protein reconstruction task. We present extensive ablations of model performance across different autoencoder layouts and compression settings. We demonstrate that our model learns a robust and structured representation of protein structures by learning a denoising diffusion probabilistic model (DDPM) [Ho et al. (2020)] in the latent space. We find Ophiuchus to enable significantly faster sampling of protein structures, as compared to available diffusion models [Watson et al. (2022); Yim et al. (2023)], while producing unconditional samples of comparable quality and diversity.

2 Background and Related Work

Symmetries in Neural Architecture for Biomolecules. Learning algorithms greatly benefit from proactively exploiting symmetry structures present in their data domain [Bronstein et al. (2021); Smidt (2021)]. In this work, we investigate relevant symmetries for the domain of protein structures:

- **Euclidean Equivariance of Coordinates and Feature Representations**. Neural models equipped with roto-translational (Euclidean) invariance or equivariance have been shown to outperform competitors in molecular and point cloud tasks [Townshend et al. (2022); Miller et al. (2020); Deng et al. (2021); Liu et al. (2022); Jing et al. (2021)]. Our proposed model takes advantage of Euclidean equivariance both in processing of coordinates and in its internal feature representations composed of scalars and higher order geometric tensors [Thomas et al. (2018); Weiler et al. (2018)].
- **Translation Equivariance of Sequence**. One-dimensional Convolutional Neural Networks (CNNs) have been demonstrated to successfully model protein sequences across a variety of tasks [Karydis (2017); Hou et al. (2018); Lee et al. (2019); Yang et al. (2022)]. These models capture sequence-motif representations that are equivariant to translation of the sequence. However, architectures for protein structures are often cast as Graph Neural Networks (GNNs) [Zhang et al. (2021)], with notable exceptions such as [Fan et al. (2022)]. In this work, we propose SO(3)-Equivariant CNN-like operations for coarsening and refining proteins directly in three dimensions.

Unsupervised Learning of Proteins. Unsupervised techniques for capturing protein sequence and structure have witnessed remarkable advancements in recent years [Lin et al. (2023); Elnaggar et al.

(2023); Zhang et al. (2022)]. Amongst unsupervised methods, autoencoder models learn to produce efficient low-dimensional representations using an informational bottleneck. These models have been successfully deployed to diverse protein tasks of modeling and sampling [Eguchi et al. (2020); Lin et al. (2021); Ramaswamy et al. (2021); Wang et al. (2022); Mansoor et al. (2023); Visani et al. (2023)], and have received renewed attention for enabling the learning of coarse representations of molecules [Wang & Gómez-Bombarelli (2019); Yang & Gómez-Bombarelli (2023); Winter et al. (2021); Wehmeyer & Noé (2018); Ramaswamy et al. (2021)].

Protein Denoising Diffusion. Denoising Diffusion Probabilistic Models (DDPM) [Sohl-Dickstein et al. (2015); Ho et al. (2020)] have found widespread adoption through diverse architectures for generative sampling of protein structures. Chroma [Ingraham et al. (2023)] trains random graph message passing through roto-translational invariant features, while RFDiffusion [Watson et al. (2022)] fine-tunes pretrained folding model RoseTTAFold [Baek et al. (2021a)] to denoising, employing SE(3)-equivariant transformers in structural decoding [Fuchs et al. (2020)]. [Yim et al. (2023); Anand & Achim (2022)] generalize denoising diffusion to frames of reference, employing Invariant Point Attention [Jumper et al. (2021)] to model three dimensions, while FoldingDiff [Wu et al. (2022a)] explores denoising in angular space. More recently, [Fu et al. (2023)] introduces a latent diffusion model. We similarly explore a model for latent diffusion in deeply encoded representations of protein structures.

3 The Ophiuchus Architecture

We represent a protein as a sequence of N residues each with an anchor position $\mathbf{P} \in \mathbb{R}^{1\times 3}$ and a tensor of irreducible representations of SO(3) $\mathbf{V}^{0:l_{\max}}$, where each $\mathbf{V}^{l} \in \mathbb{R}^{d\times(2l+1)}$ is indexed by spherical harmonics degree $l \in [0, l_{\max}]$, and has d channels of representation. A residue state is defined as $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$. These representations are directly produced from sequence labels and all-atom positions, as we describe in the **Atom Encoder/Decoder** sections (Appendix 6.1). To capture the diverse interactions within a protein, we propose three main components. **Self-Interaction** learns geometric representations of each residue independently, modeling local interactions of atoms within a single residue (Appendix 6.3). **Sequence Convolution** simultaneously updates sequential segments of residues, modeling inter-residue interactions between sequence neighbors and enabling coarsening and refinement (Appendix 6.4). Finally, **Spatial Convolution** employs message-passing of geometric features to model interactions of residues that are nearby in 3D space (Appendix 6.6). We compose these three modules to build a deep autoencoder.

3.1 Autoencoder

We compose Spatial Convolution, Self-Interaction and Sequence Convolution modules to define a coarsening/refining block (Figure 2.g). The block mixes representations across all relevant axes of the domain while producing coarsened, downsampled positions and mixed embeddings. The reverse result – finer positions and decoupled embeddings – is achieved by changing the standard Sequence Convolution to its transpose counterpart. When employing sequence convolutions of stride $S > 1$, we increase the dimensionality of the feature representation according to a rescaling factor hyperparameter ρ. We stack L coarsening blocks to build a deep neural encoder $\mathcal{E}$ (Alg.7), and symmetrically L refining blocks to build a decoder $\mathcal{D}$ (Alg. 8).

3.2 Autoencoder Reconstruction Losses

We use a number of reconstruction losses to ensure good quality of produced proteins. Please refer to Appendix 7 for further details on the losses.

Vector Map Loss. We train the model by directly comparing internal three-dimensional vector difference maps. Let $V(\mathbf{P})$ denote the internal vector map between all atoms $\mathbf{P}$ in our data, that is, $V(\mathbf{P})^{i,j} = (\mathbf{P}^{i} - \mathbf{P}^{j}) \in \mathbb{R}^{3}$. We define the vector map loss as $\mathcal{L}_{\text{VectorMap}} = \text{HuberLoss}(V(\mathbf{P}), V(\hat{\mathbf{P}}))$ [Huber (1992)]. When computing this loss, an additional stage is employed for processing permutation symmetry breaks. More details can be found in Appendix 7.1.

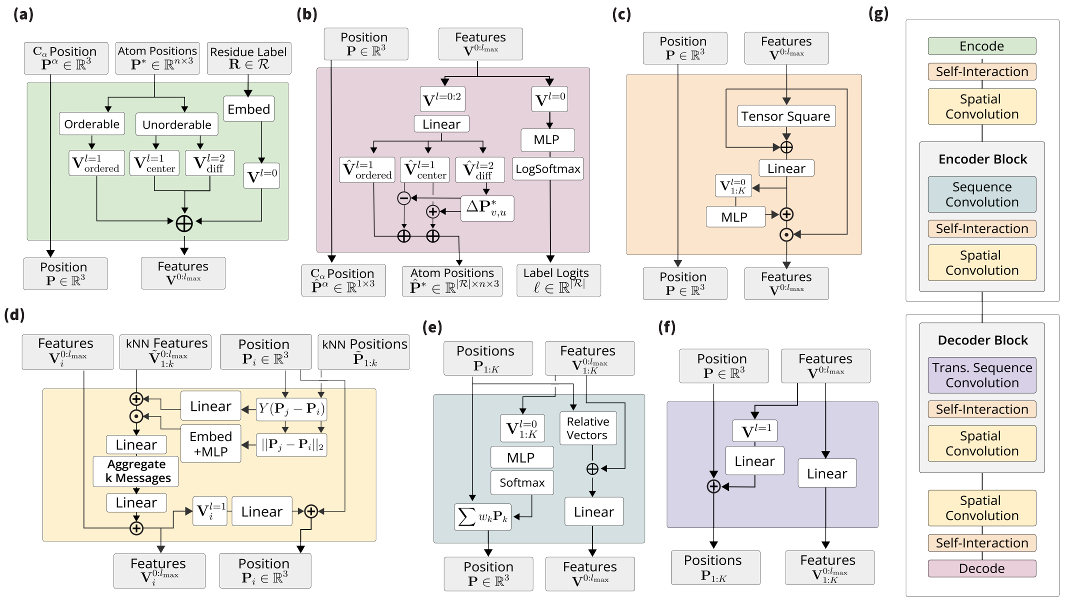

Figure 2: **Building Blocks of Ophiuchus**: **(a) Atom Encoder** and **(b) Atom Decoder** enable the model to directly take and produce atomic coordinates. **(c) Self-Interaction** updates representations internally, across different vector orders l. **(d) Spatial Convolution** interacts spatial neighbors. **(e) Sequence Convolution** and **(f) Transpose Sequence Convolution** communicate sequence neighbors and produce coarser and finer representations, respectively. **(g) Hourglass model**: we compose those modules to build encoder and decoder models, stacking them into an autoencoder.

Residue Label Cross Entropy Loss. We train the model to predict logits $\boldsymbol{\ell}$ over alphabet $\mathcal{R}$ for each residue. We use the cross entropy between predicted logits and ground labels: $\mathcal{L}_{\text{CrossEntropy}} = \text{CrossEntropy}(\boldsymbol{\ell}, \mathbf{R})$

Chemistry Losses. We incorporate L_2 norm-based losses for comparing bonds, angles and dihedrals between prediction and ground truth. For non-bonded atoms, a clash loss is evaluated using standard Van der Waals atomic radii (Appendix 7.2)

3.3 Latent Diffusion

To investigate the generative capabilities of our latent space, we train an SO(3)-equivariant denoising diffusion probabilistic model (DDPM) [Ho et al. (2020); Sohl-Dickstein et al. (2015)] on an autoencoder model that bottlenecks the representation into a single geometric tensor of features (as in Figure 4.a). We pre-train an autoencoder and transform each protein $\mathbf{X}$ from the dataset into a geometric tensor of irreducible representations of SO(3): $\mathbf{Z} = \mathcal{E}(\mathbf{X})$. We attach a diffusion process on the latent variables, taking $\mathbf{Z}_0 = \mathbf{Z}$ and $\mathbf{Z}_1 \sim \mathcal{N}(0, 1)$. We follow the parameterization introduced in [Salimans & Ho (2022)], and train a denoising model to reconstruct the original data $\mathbf{Z}_0$ from its noised version $\mathbf{Z}_t$:

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}_{\epsilon,t}\left[w_t||\hat{\mathbf{Z}}_0(\mathbf{Z}_t) - \mathbf{Z}_0||_2^2\right]$$

where $\hat{\mathbf{Z}}_0(\cdot)$ is a learnable transformation and w_t a weighting function. Please refer to Appendix 8 for further details on latent diffusion. To ensure that bottleneck representations $\mathbf{Z}_0$ are prompt for diffusion, we regularize the latent feature space of our autoencoder (Appendix 8.1).

4 Methods and Results

4.1 Autoencoder Architecture Comparison

We compare Ophiuchus to the architecture proposed in [Fu et al. (2023)], which uses the EGNN-based architecture [Satorras et al. (2022)] for autoencoding protein backbones. To the best of our knowledge, this is the only other model that attempts protein reconstruction in three-dimensions with roto-translation equivariant networks. For demonstration purposes, we curate a small dataset of protein $\mathbf{C}_\alpha$-backbones from the PDB with lengths between 16 and 64 and maximum resolution of up to 1.5 Å, resulting in 1267 proteins. We split the data into train, validation and test sets with ratio [0.8, 0.1, 0.1]. In table 1, we report the test performance at best validation step, while avoiding overfitting during training. We find that Ophiuchus vastly outperforms the EGNN-based architecture. Ophiuchus recovers the protein sequence and $\mathbf{C}_\alpha$ backbone with significantly better accuracy, while using orders of magnitude less parameters. Refer to Appendix 9 for further details.

Table 1: **SO(3)-equivariant Network Comparison: Reconstruction from bottleneck**

Model	**Downsampling Factor**	**Channels/Layer**	# **Params [1e6]**	**C**α**-RMSD** (Å) ↓	**Residue Acc.** (%) ↑ ↓
EGNN	2	[32, 32]	0.68	1.01	88
EGNN	4	[32, 32, 48]	1.54	1.12	80
EGNN	8	[32, 32, 48, 72]	3.30	2.06	73
EGNN	16	[32, 32, 48, 72, 108]	6.99	11.4	25
Ophiuchus	2	[5, 7]	0.018	0.11	98
Ophiuchus	4	[5, 7, 10]	0.026	0.14	97
Ophiuchus	8	[5, 7, 10, 15]	0.049	0.36	79
Ophiuchus	16	[5, 7, 10, 15, 22]	0.068	0.43	37

4.2 Conformational Latent Interpolation

We explore the continuity of our learned manifold by interpolating between two states of a protein directly in the latent space, as reported in [Ramaswamy et al. (2021)]. We use the PDBFlex dataset [Hrabe et al. (2016)] to choose flexible proteins. Conformational snapshots with large RMSD of these proteins are used as the endpoints for the interpolation. In Figure 3, we study the effect of model depth (coarsening/downsampling factor) across the latent interpolation. We score chemical validity by measuring the L_2 deviation of decoded bond lengths and angles from precomputed chemical tables. Additionally, we measure atomic clashes by counting collisions of non-bonded atoms. We find that despite performing worse near interpolation endpoints, interpolating representations of deeper/coarser models maintains a more consistent profile of chemical validity across the trajectory, while interpolating in three-dimensional coordinate space disrupts chemistry significantly. This shows that our autoencoder learns three-dimensional coarse representations that are amenable to manipulation while preserving the underlying structure and chemistry of proteins.

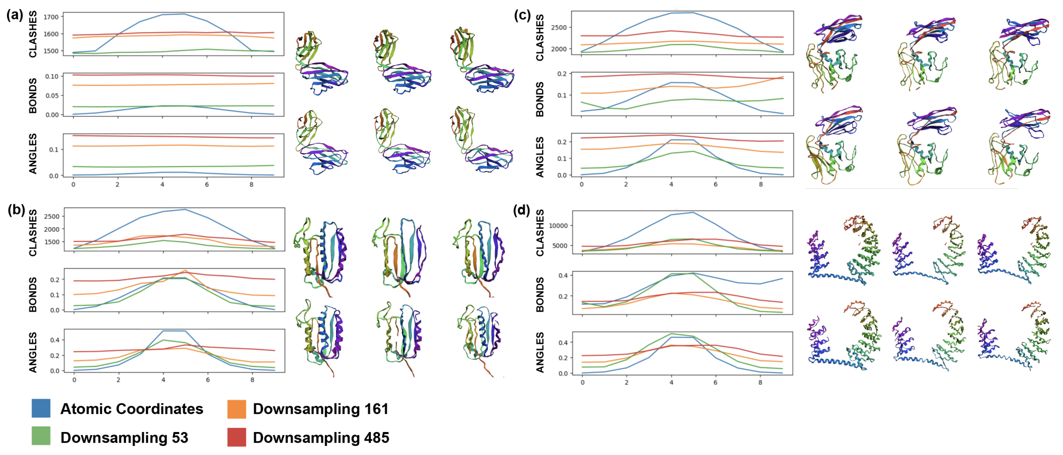

Figure 3: **Latent Conformational Interpolation.** Comparison of chemical validity metrics of protein structures obtained from latent interpolations at different autoencoder coarsening depths. PDBIDs: **(a)** 1NGZA. **(b)** 5IDMA. **(c)** 6RUMA. **(d)** 6SA7A. We show structures for steps 0.25, 0.5 and 0.75 of the trajectory for coordinate interpolation (top) and latent interpolation (bottom).

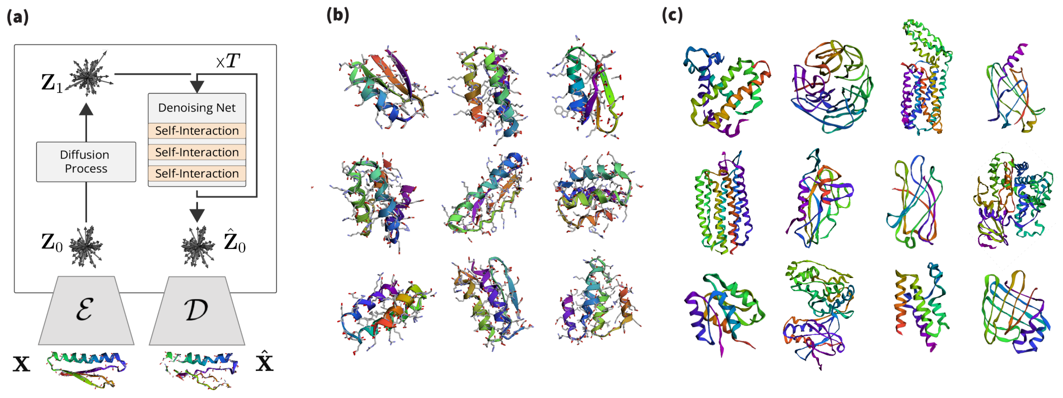

Figure 4: **Protein Latent Diffusion**. **(a)** We learn a denoising diffusion network to sample latent representations that decode to proteins. **(b)** Random samples from 64-length all-atom model. **(c)** Random samples from 485-length backbone model

Table 2: **Comparison to different diffusion models.**

Model	**Dataset**	**Sampling Time** (s) ↓	**scRMSD** (< 2Å) ↑	**scTM** (>0.5) ↑	**Diversity** ↑
FrameDiff [Yim et al. (2023)]	PDB	8.6	0.17	0.81	0.42
RFDiffusion [Watson et al. (2022)]	PDB + AlphaFold DB	50	0.79	0.99	0.64
Ophiuchus-64 All-Atom	MiniProtein	0.15	0.32	0.56	0.72
Ophiuchus-485 Backbone	PDB	0.46	0.18	0.36	0.39

4.3 Latent Diffusion Benchmarks

We benchmark our latent diffusion model by comparing the self-consistency TM-Score (scTM, introduced in [Trippe et al. (2023)]) of its generated samples against those of samples drawn from available state-of-the-art generative protein structure models [Watson et al. (2022); Yim et al. (2023)]. For each model in the comparison, we generate 500 unconditional samples for the benchmarks. Please refer to Appendix 11 for more details on comparison data and model preparation. To manage computational resources, we investigate our diffusion model in two regimes: all-atom models for short sequence lengths ($N = 64$), and backbone models for large proteins ($N = 485$). We show our results in Table 2. We find that our model performs comparably in terms of self-consistency and diversity while enabling orders of magnitude faster sampling for protein structures.

5 Conclusion and Future Work

In this work, we introduced a novel autoencoder model for protein structure and sequence representation. Our studies suggest Ophiuchus to provide a strong foundation for constructing state-of-the-art protein neural architectures. In future work, we will investigate scaling Ophiuchus to larger proteins and multimeric systems, as well as further developing its generative capabilities.

Acknowledgments

We are grateful for the continued support from the Center for Bits and Atoms (CBA), the MIT Media Lab Consortia Members and the ElevenEleven Foundation.

References

Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models, 2022.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, R. Dustin Schaeffer, Claudia Millán, Hahnbeom Park, Carson Adams, Caleb R. Glassman, Andy DeGiovanni, Jose H. Pereira, Andria V. Rodrigues, Alberdina A. van Dijk, Ana C. Ebrecht, Diederik J. Opperman, Theo Sagmeister, Christoph Buhlheller, Tea Pavkov-Keller, Manoj K. Rathinaswamy, Udit Dalwadi, Calvin K. Yip, John E. Burke, K. Christopher Garcia, Nick V. Grishin, Paul D. Adams, Randy J. Read, and David Baker. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, August 2021a. doi: 10.1126/science.abj8754. URL `https://doi.org/10.1126/science.abj8754`.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021b.

Ilyes Batatia, David Peter Kovacs, Gregor N. C. Simm, Christoph Ortner, and Gabor Csanyi. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=YPpSngE-ZU`.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1): 235–242, 2000.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL `https://arxiv.org/abs/2104.13478`.

Christopher Bystroff and David Baker. Prediction of local structure in proteins using a library of sequence-structure motifs. *Journal of molecular biology*, 281(3):565–577, 1998.

Longxing Cao, Brian Coventry, Inna Goreshnik, Buwei Huang, William Sheffler, Joon Sung Park, Kevin M Jude, Iva Marković, Rameshwar U Kadam, Koen HG Verschueren, et al. Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560, 2022.

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

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector neurons: A general framework for so(3)-equivariant networks, 2021.

Raphael R. Eguchi, Christian A. Choe, and Po-Ssu Huang. Ig-VAE: Generative modeling of protein structure by direct 3d coordinate generation. *PLOS Computational Biology*, August 2020. doi: 10.1101/2020.08.07.242347. URL `https://doi.org/10.1101/2020.08.07.242347`.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.

Ahmed Elnaggar, Hazem Essam, Wafaa Salah-Eldin, Walid Moustafa, Mohamed Elkerdawy, Charlotte Rochereau, and Burkhard Rost. Ankh: Optimized protein language model unlocks general-purpose modelling, 2023.

Hehe Fan, Zhangyang Wang, Yi Yang, and Mohan Kankanhalli. Continuous-discrete convolution for geometry-sequence modeling in proteins. In *The Eleventh International Conference on Learning Representations*, 2022.

Cong Fu, Keqiang Yan, Limei Wang, Wing Yee Au, Michael McThrow, Tao Komikado, Koji Maruhashi, Kanji Uchino, Xiaoning Qian, and Shuiwang Ji. A latent diffusion model for protein structure generation, 2023.

Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Jie Hou, Badri Adhikari, and Jianlin Cheng. Deepsf: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

Thomas Hrabe, Zhanwen Li, Mayya Sedova, Piotr Rotkiewicz, Lukasz Jaroszewski, and Adam Godzik. Pdbflex: exploring flexibility in protein structures. *Nucleic acids research*, 44(D1): D423–D428, 2016.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

John B Ingraham, Max Baranov, Zak Costello, Karl W Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M Lord, Christopher Ng-Thow-Hing, Erik R Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, pp. 1–9, 2023.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons, 2021.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Thrasyvoulos Karydis. *Learning hierarchical motif embeddings for protein engineering*. PhD thesis, Massachusetts Institute of Technology, 2017.

Jonathan Edward King and David Ryan Koes. Sidechainnet: An all-atom protein structure dataset for machine learning. *Proteins: Structure, Function, and Bioinformatics*, 89(11):1489–1496, 2021.

Sebastian Kmiecik, Dominik Gront, Michal Kolinski, Lukasz Wieteska, Aleksandra Elzbieta Dawid, and Andrzej Kolinski. Coarse-grained protein models and their applications. *Chemical reviews*, 116(14):7898–7936, 2016.

Ingoo Lee, Jongsoo Keum, and Hojung Nam. Deepconv-dti: Prediction of drug-target interactions via deep learning with convolution on protein sequences. *PLoS computational biology*, 15(6): e1007129, 2019.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs, 2023.

Yeqing Lin and Mohammed AlQuraishi. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds, 2023.

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

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

David D Liu, Ligia Melo, Allan Costa, Martin Vögele, Raphael JL Townshend, and Ron O Dror. Euclidean transformers for macromolecular structures: Lessons learned. *2022 ICML Workshop on Computational Biology*, 2022.

Craig O Mackenzie and Gevorg Grigoryan. Protein structural motifs in prediction and design. *Current opinion in structural biology*, 44:161–167, 2017.

Craig O. Mackenzie, Jianfu Zhou, and Gevorg Grigoryan. Tertiary alphabet for the observable protein structural universe. *Proceedings of the National Academy of Sciences*, 113(47), November 2016. doi: 10.1073/pnas.1607178113. URL `https://doi.org/10.1073/pnas.1607178113`.

Sanaa Mansoor, Minkyung Baek, Hahnbeom Park, Gyu Rie Lee, and David Baker. Protein ensemble generation through variational autoencoder latent space sampling. *bioRxiv*, pp. 2023–08, 2023.

Benjamin Kurt Miller, Mario Geiger, Tess E. Smidt, and Frank Noé. Relevance of rotationally equivariant convolutions for predicting molecular properties. *CoRR*, abs/2008.08461, 2020. URL `https://arxiv.org/abs/2008.08461`.

Venkata K Ramaswamy, Samuel C Musson, Chris G Willcocks, and Matteo T Degiacomi. Deep learning protein conformational space with convolutions and latent interpolations. *Physical Review X*, 11(1):011052, 2021.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 2022.

Naomi Siew, Arne Elofsson, Leszek Rychlewski, and Daniel Fischer. Maxsub: an automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, 16(9):776–785, 2000.

Tess E Smidt. Euclidean symmetry and equivariance in machine learning. *Trends in Chemistry*, 3 (2):82–85, 2021.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

Sebastian Swanson, Venkatesh Sivaraman, Gevorg Grigoryan, and Amy E Keating. Tertiary motifs as building blocks for the design of protein-binding peptides. *Protein Science*, 31(6):e4322, 2022.

Nathaniel Thomas, Tess 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, 2018.

Raphael J. L. Townshend, Martin Vögele, Patricia Suriana, Alexander Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandar, Bowen Jing, Brandon Anderson, Stephan Eismann, Risi Kondor, Russ B. Altman, and Ron O. Dror. Atom3d: Tasks on molecules in three dimensions, 2022.

Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem, 2023.

Gian Marco Visani, Michael N. Pun, Arman Angaji, and Armita Nourmohammad. Holographic-(v)ae: an end-to-end so(3)-equivariant (variational) autoencoder in fourier space, 2023.

Wujie Wang and Rafael Gómez-Bombarelli. Coarse-graining auto-encoders for molecular dynamics. *npj Computational Materials*, 5(1):125, 2019.

Wujie Wang, Minkai Xu, Chen Cai, Benjamin Kurt Miller, Tess Smidt, Yusu Wang, Jian Tang, and Rafael Gómez-Bombarelli. Generative coarse-graining of molecular conformations, 2022.

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, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*, 2022. doi: 10.1101/2022.12.09.519842. URL `https://www.biorxiv.org/content/early/2022/12/10/2022.12.09.519842`.

Christoph Wehmeyer and Frank Noé . Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148(24), mar 2018. doi: 10.1063/1.5011399. URL `https://doi.org/10.1063%2F1.5011399`.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018.

Robin Winter, Frank Noé, and Djork-Arné Clevert. Auto-encoding molecular conformations, 2021.

Kevin E. Wu, Kevin K. Yang, Rianne van den Berg, James Y. Zou, Alex X. Lu, and Ava P. Amini. Protein structure generation via folding diffusion, 2022a.

Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, et al. High-resolution de novo structure prediction from primary sequence. *BioRxiv*, pp. 2022–07, 2022b.

Kevin K Yang, Nicolo Fusi, and Alex X Lu. Convolutions are competitive with transformers for protein sequence pretraining. *bioRxiv*, pp. 2022–05, 2022.

Soojung Yang and Rafael Gómez-Bombarelli. Chemically transferable generative backmapping of coarse-grained proteins, 2023.

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, 2023.

Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics*, 12:690049, 2021.

Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

Zuobai Zhang, Minghao Xu, Arian Jamasb, Vijil Chenthamarakshan, Aurelie Lozano, Payel Das, and Jian Tang. Protein representation learning by geometric structure pretraining. *arXiv preprint arXiv:2203.06125*, 2022.

APPENDIX

6 ARCHITECTURE DETAILS

6.1 ALL-ATOM ENCODING

A canonical ordering of the atoms of each residue enables the local geometry to be described in a stacked array representation, where each feature channel corresponds to an atom. To directly encode positions, we stack the 3D coordinates of each atom. The coordinates vector behaves as the irreducible-representation of SO(3) of degree $l = 1$. The atomic coordinates are taken relative to the $\mathbf{C}_\alpha$ of each residue. In practice, for implementing this ordering we follow the *atom14* tensor format of SidechainNet [King & Koes (2021)], where a vector $P \in \mathbb{R}^{14\times 3}$ contains 14 atomic positions per residue. In Ophiuchus, we rearrange this encoding: one of those dimensions, the $\mathbf{C}_\alpha$ coordinate, is used as the absolute position; the 13 remaining 3D-vectors are centered at the $\mathbf{C}_\alpha$, and used as geometric features. The geometric features of residues with fewer than 14 atoms are zero-padded (Figure 5).

Formally, given a particular i-th residue, let $\mathbf{R}_i \in \mathcal{R}$ denote its residue label, $\mathbf{P}_i^\alpha \in \mathbb{R}^{1\times 3}$ denote the global position of its alpha carbon ($\mathbf{C}_\alpha$), and $\mathbf{P}_i^* \in \mathbb{R}^{n\times 3}$ the position of all n other atoms relative to $\mathbf{C}_\alpha$. We produce initial residue representations $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})_i$ by setting anchors $\mathbf{P}_i = \mathbf{P}_i^\alpha$, scalars $\mathbf{V}_i^{l=0} = \text{Embed}(\mathbf{R}_i)$, and geometric vectors $\mathbf{V}_i^{l>0}$ to explicitly encode relative atomic positions $\mathbf{P}_i^*$. In particular, provided the residue label $\mathbf{R}$, the heavy atoms of most standard protein residues are readily put in a canonical order, enabling direct treatment of atom positions as a stack of signals on SO(3): $\mathbf{V}^{l=1} = \mathbf{P}^*$.

Still, four of the standard residues (Aspartic Acid, Glutamic Acid, Phenylalanine and Tyrosine) have at most two pairs of atoms that are interchangeable, due to the presence $180°$-rotation symmetries [Jumper et al. (2021)]. In Figure 6, we show how stacking their relative positions leads to representations that differ even when the same structure occurs across different rotations of a side-chain. To solve this issue, instead of stacking two 3D vectors ($\mathbf{P}_u^*, \mathbf{P}_v^*$), our method uses a single $l = 1$ vector $\mathbf{V}_{\text{center}}^{l=1} = \frac{1}{2}(\mathbf{P}_v^* + \mathbf{P}_u^*)$. The mean makes this feature invariant to the (v, u) atomic permutations, and the resulting vector points to the midpoint between the two atoms. To fully describe the positioning, difference $(\mathbf{P}_v^* - \mathbf{P}_u^*)$ must be encoded as well. For that, we use a single $l = 2$ feature $\mathbf{V}_{\text{diff}}^{l=2} = Y_{l=2}\big(\frac{1}{2}(\mathbf{P}_v^* - \mathbf{P}_u^*)\big)$. This feature is produced by projecting the difference of positions into a degree $l = 2$ spherical harmonics basis. Let $x \in \mathbb{R}^3$ denote a 3D vector. Then its projection into a feature of degree $l = 2$ is defined as:

$$Y_2(x) = \left[\sqrt{15}\cdot x_0 \cdot x_2, \sqrt{15}\cdot x0 \cdot x1, \frac{\sqrt{5}}{2}\left(-x_0^2 + 2x_1^2 - x_2^2\right), \sqrt{15}\cdot x_1 \cdot x_2, \frac{\sqrt{15}}{2}\cdot(-x_0^2 + x_2^2)\right]$$

Where each dimension of the resulting term is indexed by the order $m \in [-l, l] = [-2, 2]$, for degree $l = 2$. We note that for $m \in \{-2, -1, 1\}$, two components of x directly multiply, while for $m \in \{0, 2\}$ only squared terms of x are present. In both cases, the terms are invariant to flipping the sign of x, such that $Y_2(x) = Y_2(-x)$. Equivalently, $\mathbf{V}_{\text{diff}}^{l=2}$ is invariant to reordering of the two atoms $(v, u) \to (u, v)$:

$$\mathbf{V}_{\text{diff}}^{l=2} = Y_{l=2}\left(\frac{1}{2}(\mathbf{P}_v^* - \mathbf{P}_u^*)\right) = Y_{l=2}\left(\frac{1}{2}(\mathbf{P}_u^* - \mathbf{P}_v^*)\right)$$

In Figure 6, we compare the geometric latent space of a network that uses this permutation invariant encoding, versus one that uses naive stacking of atomic positions $\mathbf{P}^*$. We find that Ophiuchus correctly maps the geometry of the data, while direct stacking leads to representations that do not reflect the underlying symmetries.

6.2 ALL-ATOM DECODING

Given a latent representation at the residue-level, $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$, we take $\mathbf{P}$ directly as the $\mathbf{C}_\alpha$ position of the residue. The hidden scalar representations $\mathbf{V}^{l=0}$ are transformed into categorical logits $\boldsymbol{\ell} \in$

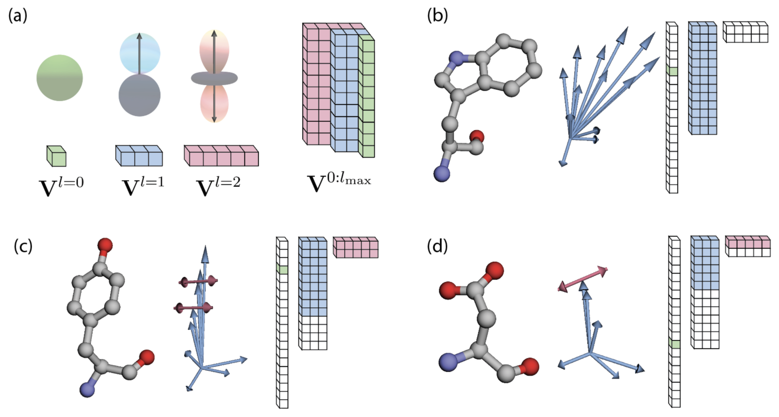

Figure 5: **Geometric Representations of Protein Residues.** **(a)** We leverage the bases of spherical harmonics to represent signals on SO(3). **(b-d)** Examples of individual protein residues encoded in spherical harmonics bases: residue label is directly encoded in a order $l = 0$ representation as a one-hot vector; atom positions in a canonical ordering are encoded as $l = 1$ features; additionally, unorderable atom positions are encoded as $l = 1$ and $l = 2$ features that are invariant to their permutation flips. In the figures, we displace $l = 2$ features for illustrative purposes – in practice, the signal is processed as centered in the $\mathbf{C}_\alpha$. **(b)** Tryptophan is the largest residue we consider, utilizing all dimensions of $n_{\text{max}} = 13$ atoms in the input $\mathbf{V}^{l=1}_{\text{ordered}}$. **(c)** Tyrosine needs padding for $\mathbf{V}^{l=1}$, but produces two $\mathbf{V}^{l=2}$ from its two pairs of permutable atoms. **(d)** Aspartic Acid has a single pair of permutable atoms, and its $\mathbf{V}^{l=2}$ is padded.

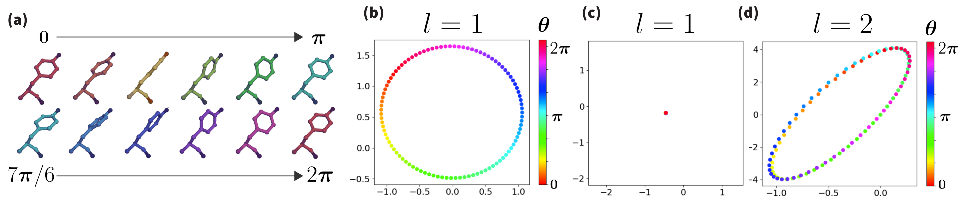

Figure 6: **Permutation Invariance of Side-chain Atoms in Stacked Geometric Representations.** **(a)** We provide an example in which we rotate the side-chain of Tyrosine by 2π radians while keeping the ordering of atoms fixed. Note that the structure is the same after rotation by π radians. **(b)** An SO(3)-equivariant network may stack atomic relative positions in a canonical order. However, because of the permutation symmetries, the naive stacked representation will lead to latent representations that are different even when the data is geometrically the same. To demonstrate this, we plot two components ($m = -1$ and $m = 0$) of an internal $\mathbf{V}^{l=1}$ feature, while rotating the side-chain positions by 2π radians. This network represents the structures rotated by $\theta = 0$ (red) and $\theta = \pi$ (cyan) differently despite having exactly the same geometric features. **(c-d)** Plots of internal $\mathbf{V}^{l=1,2}$ of Ophiuchus which encodes the position of permutable pairs jointly as an $l = 1$ center of symmetry and an $l = 2$ difference of positions, resulting in the *same* representation for structures rotated by $\theta = 0$ (red) and $\theta = \pi$ (cyan).

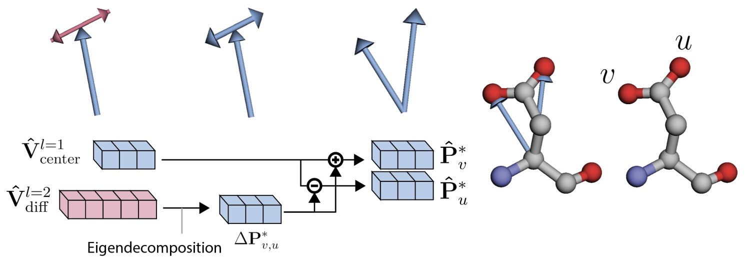
$\hat{\mathbf{V}}^{l=1}_{\text{center}}$
$\hat{\mathbf{V}}^{l=2}_{\text{diff}}$
Eigendecomposition
$\Delta\mathbf{P}^*_{v,u}$
$\hat{\mathbf{P}}^*_v$
$\hat{\mathbf{P}}^*_u$
u
v

Figure 7: **Decoding Permutable Atoms.** Sketch on how to decode a double-sided arrow ($\mathbf{V}^{l=2}$ signals) into two unordered vectors ($\hat{\mathbf{P}}^*_u$, $\hat{\mathbf{P}}^*_v$). $Y^{l=2}: \mathbb{R}^3 \rightarrow \mathbb{R}^5$ can be viewed as a 3 dimensional manifold embed in a 5 dimensional space. We exploit the fact that the points on that manifold are unaffected by a rotation around their pre-image vector. To extend the definition to all the points of $\mathbb{R}^5$ (i.e. also outside of the manifold), we look for the axis of rotation with the smallest impact on the 5d entry. For that, we compute the left-eigenvector with the smallest eigenvalue of the action of the generators on the input point: $\mathcal{X}^{l=2}\hat{\mathbf{V}}^{l=2}_{\text{diff}}$. The resulting axis is used as a relative position $\pm\Delta\mathbf{P}^*_{v,u}$ between the two atoms, and is used to recover atomic positions through $\mathbf{P}^*_v = \mathbf{V}^{l=1}_{\text{center}} + \Delta\mathbf{P}^*_{v,u}$ and $\mathbf{P}^*_u = \mathbf{V}^{l=1}_{\text{center}} - \Delta\mathbf{P}^*_{v,u}$.

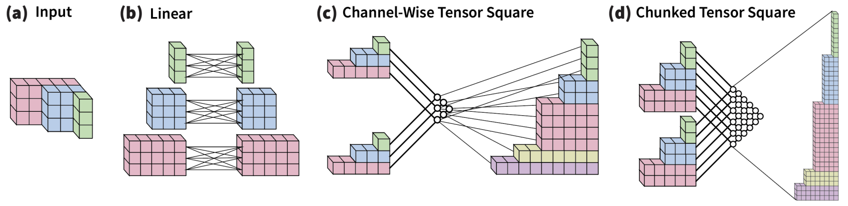

Figure 8: **Building Blocks of Self-Interaction. (a)** Self-Interaction updates only SO(3)-equivariant features, which are represented as a D channels each with vectors of degree l up to degree l_{max}. **(b)** A roto-translational equivariant linear layer transforms only within the same order l. **(c-d)** we use the tensor square operation to interact features across different degrees l. We employ two instantiations of this operation. **(c)** The Self-Interaction in autoencoder models applies the square operation within the same channel of the representation, interacting features with themselves across l of same channel dimension $d \in [0, D]$. **(d)** The Self-Interaction in diffusion models chunks the representation in groups of channels before the square operation. It is more expressive, but imposes a harder computational cost.

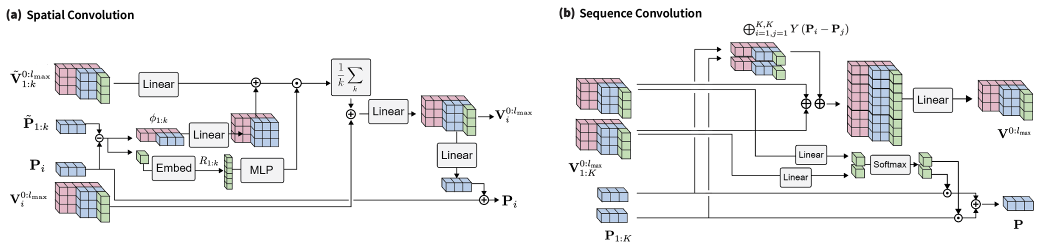
(a) Spatial Convolution
$\tilde{\mathbf{V}}^{0:l_{\max}}_{1:k}$
Linear
$\frac{1}{k}\sum_k$
$\phi_{1:k}$
Linear
$\tilde{\mathbf{P}}_{1:k}$
$\mathbf{P}_i$
Embed
$R_{1:k}$
MLP
$\mathbf{V}^{0:l_{\max}}_i$
Linear
Linear
$\mathbf{V}^{0:l_{\max}}_i$
$\mathbf{P}_i$
(b) Sequence Convolution
$\bigoplus^{K,K}_{i=1,j=1} Y(\mathbf{P}_i - \mathbf{P}_j)$
Linear
$\mathbf{V}^{0:l_{\max}}$
$\tilde{\mathbf{V}}^{0:l_{\max}}_{1:K}$
Linear
Linear
Softmax
$\mathbf{P}_{1:K}$
$\mathbf{P}$

Figure 9: **Convolutions of Ophiuchus. (a)** In the Spatial Convolution, we update the feature representation $\mathbf{V}^{0:l_{\max}}_i$ and position $\mathbf{P}_i$ of a node i by first aggregating messages from its k nearest-neighbors. The message is composed out of the neighbor features $\tilde{\mathbf{V}}^{0:l_{\max}}_{1:k}$ and the relative position between the nodes $\mathbf{P}_i - \tilde{\mathbf{P}}_{1:k}$. After updating the features $\mathbf{V}^{0:l_{\max}}_i$, we project vectors $\mathbf{V}^{l=1}_i$ to predict an update to the position $\mathbf{P}_i$. **(b)** In the Sequence Convolution, we concatenate the feature representations of sequence neighbors $\tilde{\mathbf{V}}^{0:l_{\max}}_{1:K}$ along with spherical harmonic signals of their relative positions $\mathbf{P}_i - \mathbf{P}_j$, for $i \in [1, K]$, $j \in [1, K]$. The concatenated feature vector is then linearly projected to the output dimensionality. To produce a coarsened position, each $\tilde{\mathbf{V}}^{0:l_{\max}}_{1:K}$ produces a score that is used as weight in summing the original positions $\mathbf{P}_{1:K}$. The Softmax function is necessary to ensure the sum of the weights is normalized.

$\mathbb{R}^{|\mathcal{R}|}$ to predict the probabilities of residue label $\hat{\mathbf{R}}$. To decode the side-chain atoms, Ophiuchus produces relative positions to $\mathbf{C}_\alpha$ for all atoms of each residue. During training, we enforce the residue label to be the ground truth. During inference, we output the residue label corresponding to the largest logit value.

Relative coordinates are produced directly from geometric features. We linearly project $\mathbf{V}^{l=1}$ to obtain the relative position vectors of orderable atoms $\hat{\mathbf{V}}^{l=1}_{\text{ordered}}$. To decode positions $(\hat{\mathbf{P}}^*_v, \hat{\mathbf{P}}^*_u)$ for an unorderable pair of atoms (v, u), we linearly project $\mathbf{V}^{l>0}$ to predict $\hat{\mathbf{V}}^{l=1}_{\text{center}}$ and $\hat{\mathbf{V}}^{l=2}_{\text{diff}}$. To produce two relative positions out of $\hat{\mathbf{V}}^{l=2}_{\text{diff}}$, we determine the rotation axis around which the feature rotates the least by taking the left-eigenvector with smallest eigenvalue of $\mathcal{X}^{l=2}\hat{\mathbf{V}}^{l=2}_{\text{diff}}$, where $\mathcal{X}^{l=2}$ is the generator of the irreducible representations of degree $l = 2$. We illustrate this procedure in Figure 7 and explain the process in detail in the caption. This method proves effective because the output direction of this axis is inherently ambiguous, aligning perfectly with our requirement for the vectors to be unorderable.

Algorithm 1: ALL-ATOM ENCODING

```
Input: C_α Position $\mathbf{P}^\alpha \in \mathbb{R}^{1\times 3}$
Input: All-Atom Relative Positions
       $\mathbf{P}^* \in \mathbb{R}^{n\times 3}$
Input: Residue Label $\mathbf{R} \in \mathcal{R}$
Output: Latent Representation $(\mathbf{P}, \mathbf{V}^{l=0:2})$
$\mathbf{P} \leftarrow \mathbf{P}^\alpha$
$\mathbf{V}^{l=0} \leftarrow \text{Embed}(\mathbf{R})$
$\mathbf{V}^{l=1}_{\text{ordered}} \leftarrow \text{GetOrderablePositions}(\mathbf{R}, \mathbf{P}^*)$
$\mathbf{P}^*_v, \mathbf{P}^*_u \leftarrow \text{GetUnorderablePositionPairs}(\mathbf{R}, \mathbf{P}^*)$
$\mathbf{V}^{l=1}_{\text{center}} \leftarrow \frac{1}{2}(\mathbf{P}^*_v + \mathbf{P}^*_u)$
$\mathbf{V}^{l=2}_{\text{diff}} \leftarrow Y^2\big(\frac{1}{2}(\mathbf{P}^*_v - \mathbf{P}^*_u)\big)$
$\mathbf{V}^{l=0:2} \leftarrow \mathbf{V}^{l=1}_{\text{ordered}} \oplus \mathbf{V}^{l=1}_{\text{center}} \oplus \mathbf{V}^{l=2}_{\text{diff}}$
return $(\mathbf{P}, \mathbf{V}^{l=0:2})$
```

Algorithm 2: ALL-ATOM DECODING

```
Input: Latent Representation $(\mathbf{P}, \mathbf{V}^{l=0:2})$
Output: C_α Position $\mathbf{P}^\alpha \in \mathbb{R}^{1\times 3}$
Output: All-Atom Relative Positions
        $\mathbf{P}^* \in \mathbb{R}^{|\mathcal{R}|\times n\times 3}$
Output: Residue Label Logits $\boldsymbol{\ell} \in \mathbb{R}^{|\mathcal{R}|}$
$\mathbf{P}^\alpha \leftarrow \mathbf{P}$
$\boldsymbol{\ell} \leftarrow \text{LogSoftmax}\big(\text{Linear}(\mathbf{V}^{l=0})\big)$
$\hat{\mathbf{V}}^{l=1}_{\text{ordered}} \leftarrow \text{Linear}(\mathbf{V}^{l=1})$
$\hat{\mathbf{V}}^{l=1}_{\text{center}}, \hat{\mathbf{V}}^{l=2}_{\text{diff}} \leftarrow \text{Linear}(\mathbf{V}^{l=0:2})$
$\Delta\mathbf{P}_{v,u} \leftarrow \text{Eigendecompose}\big(\hat{\mathbf{V}}^{l=2}_{\text{diff}}\big)$
$\mathbf{P}^*_v, \mathbf{P}^*_u = \hat{\mathbf{V}}^{l=1}_{\text{center}} + \Delta\mathbf{P}_{v,u}, \hat{\mathbf{V}}^{l=1}_{\text{center}} - \Delta\mathbf{P}_{v,u}$
$\mathbf{P}^* \leftarrow \mathbf{P}^* \oplus \mathbf{P}^*_v \oplus \mathbf{P}^*_u$
return $(\mathbf{P}^\alpha, \mathbf{P}^*, \boldsymbol{\ell})$
```

6.3 SELF-INTERACTION

Our Self-Interaction is designed to model the internal interactions of atoms within each residue. This transformation updates the feature vectors $\mathbf{V}^{0:l_{\max}}_i$ centered at the same residue i. Importantly, it blends feature vectors $\mathbf{V}^l$ of varying degrees l by employing tensor products of the features with themselves. Our Self-Interaction module draws inspiration from MACE [Batatia et al. (2022)].

Algorithm 3: Self-Interaction

```
Input: Latent Representation $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
$\mathbf{V}^{0:l_{\max}} \leftarrow \mathbf{V}^{0:l_{\max}} \oplus \big(\mathbf{V}^{0:l_{\max}}\big)^{\otimes 2}$      ▷ Tensor Square and Concatenate
$\mathbf{V}^{0:l_{\max}} \leftarrow \text{Linear}(\mathbf{V}^{0:l_{\max}})$                           ▷ Update features
$\mathbf{V}^{0:l_{\max}} \leftarrow \text{MLP}(\mathbf{V}^{l=0}) \cdot \mathbf{V}^{0:l_{\max}}$                  ▷ Gate Activation Function
return $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
```

The objective of our Self-Interaction module is to function exclusively based on the geometric features $\mathbf{V}^{0:l_{\max}}$, while concurrently mixing irreducible representations across various l values. To accomplish this, we calculate the tensor product of the representation with itself; this operation is termed the "tensor square" and is symbolized by $\big(\mathbf{V}^{0:l_{\max}}\big)^{\otimes 2}$. As the channel dimensions expand, the computational complexity tied to the tensor square increases quadratically. To solve this computational load, we instead perform the square operation channel-wise, or by chunks of channels. Figures 8.c and 8.d illustrate these operations. After obtaining the squares from the chunked or individual channels, the segmented results are subsequently concatenated to generate an updated representation $\mathbf{V}^{0:l_{\max}}$, which is transformed through a learnable linear layer to the output dimensionality.

6.4 Sequence Convolution

To take advantage of the sequential nature of proteins, we propose a one-dimensional, roto-translational equivariant convolutional layer for acting on geometric features and positions of sequence neighbors. Given a kernel window size K and stride S, we concatenate representations $\mathbf{V}^{0:l_{\max}}_{i-\frac{K}{2}:i+\frac{K}{2}}$ with the same l value. Additionally, we include normalized relative vectors between anchoring positions $\mathbf{P}_{i-\frac{K}{2}:i+\frac{K}{2}}$. Following conventional CNN architectures, this concatenated representation undergoes a linear transformation. The scalars in the resulting representation are then converted into weights, which are used to combine window coordinates into a new coordinate. To ensure translational equivariance, these weights are constrained to sum to one.

Algorithm 4: Sequence Convolution

```
Input: Window of Latent Representations $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})_{1:K}$
$w_{1:K} \leftarrow \text{Softmax}\Big(\text{MLP}\big(\mathbf{V}^{0}_{1:K}\big)\Big)$            ▷ Such that $\sum_k w_k = 1$
$\mathbf{P} \leftarrow \sum_{k=1}^{K} w_k \mathbf{P}_k$            ▷ Coarsen coordinates
$\tilde{\mathbf{V}} \leftarrow \bigoplus_K \mathbf{V}^{0:l_{\max}}_{1:K}$            ▷ Stack features
$\tilde{\mathbf{P}} \leftarrow \bigoplus_{i=1,j=1}^{K,K} Y(\mathbf{P}_i - \mathbf{P}_j)$            ▷ Stack relative vectors
$\mathbf{V}^{0:l_{\max}} \leftarrow \text{Linear}\Big(\tilde{\mathbf{V}} \oplus \tilde{\mathbf{P}}\Big)$            ▷ Coarsen features and vectors
return $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
```

When $S > 1$, sequence convolutions reduce the dimensionality along the sequence axis, yielding mixed representations and coarse coordinates. To reverse this procedure, we introduce a transpose convolution algorithm that uses its $\mathbf{V}^{l=1}$ features to spawn coordinates (Appendix 6.5).

6.5 Transpose Sequence Convolution

Given a single coarse anchor position and features representation $(\mathbf{P}, \mathbf{V}^{l=0:l_{\max}})$, we first map $\mathbf{V}^{l=0:l_{\max}}$ into a new representation, reshaping it by chunking K features. We then project $\mathbf{V}^{l=1}$ and produce K relative position vectors $\Delta\mathbf{P}_{1:K}$, which are summed with the original position $\mathbf{P}$ to produce K new coordinates.

Algorithm 5: Transpose Sequence Convolution

```
Input: Kernel Size $K$
Input: Latent Representations $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
$\Delta\mathbf{P}_{1:K} \leftarrow \text{Linear}\left(\mathbf{V}^{l=1}\right)$            ▷ Predict $K$ relative positions
$\mathbf{P}_{1:K} \leftarrow \mathbf{P} + \Delta\mathbf{P}_{1:K}$
$\mathbf{V}^{l=0:l_{\max}}_{1:K} \leftarrow \text{Reshape}_K\left(\text{Linear}(\mathbf{V}^{l=0:l_{\max}})\right)$            ▷ Split the channels in $K$ chunks
return $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})_{1:K}$
```

This procedure generates windows of K representations and positions. These windows may intersect in the decoded output. We resolve those superpositions by taking the average position and average representations within intersections.

6.6 Spatial Convolution

To capture interactions of residues that are close in three-dimensional space, we introduce the Spatial Convolution. This operation updates representations and positions through message passing within k-nearest spatial neighbors. Message representations incorporate SO(3) signals from the vector difference between neighbor coordinates, and we aggregate messages with a permutation-invariant means. After aggregation, we linearly transform the vector representations into a an update for the coordinates.

Algorithm 6: Spatial Convolution

```
Input: Latent Representations $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})_{1:N}$
Input: Output Node Index $i$
$(\tilde{\mathbf{P}}, \tilde{\mathbf{V}}^{0:l_{\max}})_{1:k} \leftarrow k$-Nearest-Neighbors$(\mathbf{P}_i, \mathbf{P}_{1:N})$
$R_{1:k},\ \phi_{1:k} \leftarrow \text{Embed}(||\tilde{\mathbf{P}}_{1:k} - \mathbf{P}_i||_2),\ Y(\tilde{\mathbf{P}}_{1:k} - \mathbf{P}_i)$    ▷ Edge Features
$\tilde{\mathbf{V}}^{0:l_{\max}}_{1:k} \leftarrow \text{MLP}(R_k) \cdot \left(\text{Linear}(\tilde{\mathbf{V}}^{0:l_{\max}}_{1:k}) + \text{Linear}\left(\phi_{1:k}\right)\right)$    ▷ Prepare messages
$\mathbf{V}^{0:l_{\max}} \leftarrow \text{Linear}\left(\mathbf{V}^{0:l_{\max}}_i + \frac{1}{k}\left(\sum_k \tilde{\mathbf{V}}^{0:l_{\max}}_k\right)\right)$    ▷ Aggregate and update
$\mathbf{P} \leftarrow \mathbf{P}_i + \text{Linear}\left(\mathbf{V}^{l=1}\right)$    ▷ Update positions
return $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
```

6.7 Non-Linearities

To incorporate non-linearities in our geometric representations $\mathbf{V}^{l=0:l_{\max}}$, we employ a similar roto-translation equivariant gate mechanism as described in Equiformer [Liao & Smidt (2023)]. This mechanism is present at the last step of Self-Interaction and in the message preparation step of Spatial Convolution (Figure 2). In both cases, we implement the activation by first isolating the scalar representations $\mathbf{V}^{l=0}$ and transforming them through a standard MultiLayerPerceptron (MLP). We use the SiLu activation function [Elfwing et al. (2017)] after each layer of the MLP. In the output vector, a scalar is produced for and multiplied into each channel of $\mathbf{V}^{l=0:l_{\max}}$.

6.8 Roto-Translation Equivariance of Sequence Convolution

A Sequence Convolution kernel takes in K coordinates $\mathbf{P}_{1:K}$ to produce a single coordinate $\mathbf{P} = \sum_{i=1}^{K} w_i \mathbf{P}_i$. We show that these weights need to be normalized in order for translation equivariance to be satisfied. Let T denote a 3D translation vector, then translation equivariance requires:

$$(\mathbf{P} + T) = \sum_{i=1}^{K} w_i(\mathbf{P}_i + T) = \sum_{i=1}^{K} w_i \mathbf{P}_i + \sum_{i=1}^{K} w_i T \ \ \rightarrow \ \ \sum_{i=1}^{K} w_i = 1$$

Rotation equivariance is immediately satisfied since the sum of 3D vectors is a rotation equivariant operation. Let R denote a rotation matrix. Then,

$$(R\mathbf{P}) = R\sum_{i=1}^{K} w_i(\mathbf{P}_i) = \sum_{i=1}^{K} w_i(R\mathbf{P}_i)$$

6.9 Layer Normalization and Residual Connections

Training deep models can be challenging due to vanishing or exploding gradients. We employ layer normalization [Ba et al. (2016)] and residual connections [He et al. (2015)] in order to tackle those challenges. We incorporate layer normalization and residual connections at the end of every Self-Interaction and every convolution. To keep roto-translation equivariance, we use the layer normalization described in Equiformer [Liao & Smidt (2023)], which rescales the l signals independetly within a representation by using the root mean square value of the vectors. We found both residuals and layer norms to be critical in training deep models for large proteins.

6.10 Encoder-Decoder

Below we describe the encoder/decoder algorithm using the building blocks previously introduced.

Algorithm 7: Encoder

```
Input: $\mathbf{C}_\alpha$ Position $\mathbf{P}^\alpha \in \mathbb{R}^{1\times 3}$
Input: All-Atom Relative Positions $\mathbf{P}^* \in \mathbb{R}^{n\times 3}$
Input: Residue Label $\mathbf{R} \in \mathcal{R}$
Output: Latent Representation $(\mathbf{P}, \mathbf{V}^{l=0:l_{\max}})$
$(\mathbf{P}, \mathbf{V}^{0:2}) \leftarrow$ All-Atom Encoding$(\mathbf{P}^\alpha, \mathbf{P}^*, \mathbf{R})$        ▷ Initial Residue Encoding
$(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Linear$(\mathbf{P}, \mathbf{V}^{0:2})$
for $i \leftarrow 1$ to $L$ do
    $(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Self-Interaction$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
    $(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Spatial Convolution$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
    $(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Sequence Convolution$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
end
return $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
```

Algorithm 8: Decoder

```
Input: Latent Representation $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
Output: Protein Structure and Sequence Logits $(\hat{\mathbf{P}}^\alpha, \hat{\mathbf{P}}^*, \boldsymbol{\ell})$
for $i \leftarrow 1$ to $L$ do
    $(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Self-Interaction$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
    $(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Spatial Convolution$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
    $(\mathbf{P}, \mathbf{V}^{0:l_{\max}}) \leftarrow$ Transpose Sequence Convolution$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
end
$(\mathbf{P}, \mathbf{V}^{0:2}) \leftarrow$ Linear$(\mathbf{P}, \mathbf{V}^{0:l_{\max}})$
$(\hat{\mathbf{P}}^\alpha, \hat{\mathbf{P}}^*, \boldsymbol{\ell}) \leftarrow$ All-Atom Decoding$(\mathbf{P}, \mathbf{V}^{0:2})$        ▷ Final Protein Decoding
return $(\hat{\mathbf{P}}^\alpha, \hat{\mathbf{P}}^*, \boldsymbol{\ell})$
```

6.11 Variable Sequence Length

We handle proteins of different sequence length by setting a maximum size for the model input. During training, proteins that are larger than the maximum size are cropped, and those that are smaller are padded. The boundary residue is given a special token that labels it as the end of the protein. For inference, we crop the tail of the output after the sequence position where this special token is predicted.

6.12 Time Complexity

We analyse the time complexity of a forward pass through the Ophiuchus autoencoder. Let the list $(\mathbf{P}, \mathbf{V}^{0:l_{\max}})_N$ be an arbitrary latent state with N positions and N geometric representations of dimensionality D, where for each $d \in [0, D]$ there are geometric features of degree $l \in [0, l_{\max}]$. Note that size of a geometric representation grows as $O(D \cdot l_{\max}^2)$ in memory.

- The cost of an SO(3)-equivariant linear layer (Fig. 8.b) is $O(D^2 \cdot l_{\max}^3)$.
- The cost of a channel-wise tensor square operation (Fig. 8.c) is $O(D \cdot l_{\max}^4)$.
- In **Self-Interaction** (Alg. 3), we use a tensor square and project it using a linear layer. The time complexity is given by $O\left(N \cdot (D^2 \cdot l_{\max}^3 + D \cdot l_{\max}^4)\right)$ for the whole protein.
- In **Spatial Convolution** (Alg. 6), a node aggregates geometric messages from k of its neighbors. Resolving the k nearest neighbors can be efficiently done in $O\left((N + k)\log N\right)$ through the k-d tree data structure [Bentley (1975)]. For each residue, its k neighbors prepare messages through linear layers, at total cost $O(N \cdot k \cdot D^2 \cdot l_{\max}^3)$.

- In a **Sequence Convolution** (Alg. 4), a kernel stacks K geometric representations of dimensionality D and linearly maps them to a new feature of dimensionality $\rho \cdot D$, where ρ is a rescaling factor, yielding $O((K \cdot D) \cdot (\rho \cdot D) \cdot l_{\text{max}}^3) = O(K \cdot \rho \cdot D^2 \cdot l_{\text{max}}^3)$. With length N and stride S, the total cost is $O\left(\frac{N}{S} \cdot K \cdot \rho \cdot D^2 \cdot l_{\text{max}}^3\right)$.
- The cost of an **Ophiuchus Block** is the sum of the terms above,
$$O\left(D \cdot l_{\text{max}}^3 \cdot N \cdot (\frac{K}{S} \cdot \rho D + kD)\right)$$
- An **Autoencoder** (Alg. 7,8) that uses stride S convolutions for coarsening uses $L = O\left(\log_S(N)\right)$ layers to reduce a protein of size N into a single representation. At depth i, the dimensionality is given by $D_i = \rho^i D$ and the sequence length is given by $N_i = \frac{N}{S^i}$. The time complexity of our Autoencoder is given by geometric sum:
$$O\left(\sum_i^{\log_S(N)} \left(l_{\text{max}}^3 \rho^i D \frac{N}{S^i}(K\rho^{i+1}D + k\rho^i D)\right)\right) = O\left(N l_{\text{max}}^3 (K\rho + k) D^2 \sum_i^{\log_S(N)} \left(\frac{\rho^2}{S}\right)^i\right)$$
We are interested in the dependence on the length N of a protein, therefore, we keep only relevant parameters. Summing the geometric series and using the identity $x^{\log_b(a)} = a^{\log_b(x)}$ we get:
$$= O\left(N \frac{(\frac{\rho^2}{S})^{\log_S(N)+1} - 1}{\frac{\rho^2}{S} - 1}\right) = \begin{cases} O\left(N^{1+\log_S \rho^2}\right) & \text{for } \rho^2/S > 1 \\ O\left(N \log_S N\right) & \text{for } \rho^2/S = 1 \\ O\left(N\right) & \text{for } \rho^2/S < 1 \end{cases}$$
In most of our experiments we operate in the $\rho^2/S < 1$ regime.

7 Loss Details

7.1 Details on Vector Map Loss

The Huber Loss [Huber (1992)] behaves linearly for large inputs, and quadratically for small ones. It is defined as:

$$\text{HuberLoss}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \le \delta, \\ \delta \cdot |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

We found it to significantly improve training stability for large models compared to mean squared error. We use $\delta = 0.5$ for all our experiments.

The vector map loss measures differences of internal vector maps $V(\mathbf{P}) - V(\hat{\mathbf{P}})$ between predicted and ground positions, where $V(\mathbf{P})^{i,j} = (\mathbf{P}^i - \mathbf{P}^j) \in \mathbb{R}^{\times 3}$. Our output algorithm for decoding atoms produces arbitrary symmetry breaks (Appendix 6.2) for positions $\mathbf{P}_v^*$ and $\mathbf{P}_u^*$ of atoms that are not orderable. Because of that, a loss on the vector map is not directly applicable to the output of our model, since the order of the model output might differ from the order of the ground truth data. To solve that, we consider both possible orderings of permutable atoms, and choose the one that minimizes the loss. Solving for the optimal ordering is not feasible for the system as a whole, since the number of permutations to be considered scales exponentially on N. Instead, we first compute a vector map loss internal to each residue. We consider the alternative order of permutable atoms, and choose the candidate that minimizes this local loss. This ordering is used for the rest of our losses.

7.2 Chemical Losses

We consider bonds, interbond angles, dihedral angles and steric clashes when computing a loss for chemical validity. Let $\mathbf{P} \in \mathbb{R}^{N_{\text{Atom}} \times 3}$ be the list of atom positions in ground truth data. We denote $\hat{\mathbf{P}} \in \mathbb{R}^{N_{\text{Atom}} \times 3}$ as the list of atom positions predicted by the model. For each chemical interaction, we precompute indices of atoms that perform the interaction. For example, for bonds we precompute a list of pairs of atoms that are bonded according to the chemical profile of each residue. Our chemical losses then take form:

$$\mathcal{L}_{\text{Bonds}} = \frac{1}{|\mathcal{B}|} \sum_{(v,u)\in\mathcal{B}} \Big|\Big| \; ||\hat{\mathbf{P}}_v - \hat{\mathbf{P}}_u||_2 - ||\mathbf{P}_v - \mathbf{P}_u||_2 \; \Big|\Big|_2^2$$

where $\mathcal{B}$ is a list of pair of indices of atoms that form a bond. We compare the distance between bonded atoms in prediction and ground truth data.

$$\mathcal{L}_{\text{Angles}} = \frac{1}{|\mathcal{A}|} \sum_{(v,u,p)\in\mathcal{A}} ||\alpha(\hat{\mathbf{P}}_v, \hat{\mathbf{P}}_u \hat{\mathbf{P}}_p) - \alpha\left(\mathbf{P}_v, \mathbf{P}_u, \mathbf{P}_p\right)||_2^2$$

where $\mathcal{A}$ is a list of 3-tuples of indices of atoms that are connected through bonds. The tuple takes the form (v, u, p) where u is connected to v and to p. Here, the function $\alpha(\cdot,\cdot,\cdot)$ measures the angle in radians between positions of atoms that are connected through bonds.

$$\mathcal{L}_{\text{Dihedrals}} = \frac{1}{|\mathcal{D}|} \sum_{(v,u,p,q)\in\mathcal{D}} ||\tau(\hat{\mathbf{P}}_v, \hat{\mathbf{P}}_u, \hat{\mathbf{P}}_p, \hat{\mathbf{P}}_q) - \tau(\mathbf{P}_v, \mathbf{P}_u, \mathbf{P}_p, \mathbf{P}_q)||_2^2$$

where $\mathcal{D}$ is a list of 4-tuples of indices of atoms that are connected by bonds, that is, (v, u, p, q) where (v, u), (u, p) and (p, q) are connected by bonds. Here, the function $\tau(\cdot,\cdot,\cdot,\cdot)$ measures the dihedral angle in radians.

$$\mathcal{L}_{\text{Clashes}} = \frac{1}{|\mathcal{C}|} \sum_{(v,u)\in\mathcal{C}} H(r_v + r_u - ||\hat{\mathbf{P}}_v - \hat{\mathbf{P}}_u||_2)$$

where H is a smooth and differentiable Heaviside-like step function, $\mathcal{C}$ is a list of pair of indices of atoms that are not bonded, and (r_v, r_u) are the van der Waals radii of atoms v, u.

7.3 Total Loss

We weight the different losses in our pipeline. For the standard training of the autoencoder, we use weights:

$$\mathcal{L} = 10 \cdot \mathcal{L}_{\text{VectorMap}} + \mathcal{L}_{\text{CrossEntropy}} + 0.1 \cdot \mathcal{L}_{\text{Bonds}} + 0.1 \cdot L_{\text{Angles}} + 0.1 \cdot \mathcal{L}_{\text{Dihedrals}} + 10 \cdot \mathcal{L}_{\text{Clashes}}$$

For fine-tuning the model, we increase the weight of chemical losses significantly:

$$\mathcal{L} = 10 \cdot \mathcal{L}_{\text{VectorMap}} + \mathcal{L}_{\text{CrossEntropy}} + 1.0 \cdot \mathcal{L}_{\text{Bonds}} + 1.0 \cdot L_{\text{Angles}} + 1.0 \cdot \mathcal{L}_{\text{Dihedrals}} + 100 \cdot \mathcal{L}_{\text{Clashes}}$$

We find that high weight values for chemical losses at early training may hurt the model convergence, in particular for models that operate on large lengths.

8 Latent Diffusion Details

We train all Ophiuchus diffusion models with learning rate lr $= 1 \times 10^{-3}$ for 10,000 epochs. For denoising networks we use Self-Interactions with the chunked-channel tensor square operation (Alg. 3).

To sample, we start at $t = 1$ with $\mathbf{Z}_1 \sim \mathcal{N}(0, 1)$ and iteratively apply the following update rule:

$$\mathbf{Z}_s \leftarrow \xi_1(t, s)\; \mathbf{Z}_t + \xi_2(t, s)\; \hat{\mathbf{Z}}_0(\mathbf{Z}_t) + \xi_3(t, s)\; \epsilon$$

where $s, t \in [0, 1]$, $s < t$, ϵ is Gaussian noise over SO(3), and the constants $\xi(\cdot,\cdot)$ parameterize the denoising process according to a noise schedule.

The $\xi(\cdot,\cdot)$ constants used in sampling from the diffusion model are drawn from [Salimans & Ho (2022)] and are as follows:

$$\xi_1(t, s) = e^{\lambda_t - \lambda_s}(\alpha_s/\alpha_t)$$
$$\xi_2(t, s) = (1 - e^{\lambda_t - \lambda_s})\alpha_s$$
$$\xi_3(t, s) = \sqrt{(\tilde{\sigma}^2_{s|t})^{1-\gamma}(\sigma^2_{t|s})^{\gamma}}$$

where $\sigma^2_{t|s} = (1 - e^{\lambda_t - \lambda_s})\sigma_t^2$, $\tilde{\sigma}^2_{s|t} = (1 - e^{\lambda_t - \lambda_s})\sigma_s^2$, γ is a noise control parameter, $\lambda_t = \log(\alpha_t^2/\sigma_t^2)$ is the signal-to-noise ratio, and α_t and σ_t define a noise schedule. In practice, we discretize the process in $T = 1000$ steps. In this paper, we only investigate the linear σ_t^2 schedule described in [Ho et al. (2020)]. In future work we plan to examine results for different denoising parameterizations, noise schedulers, and varying T to latent diffusion in the Ophiuchus latent space.

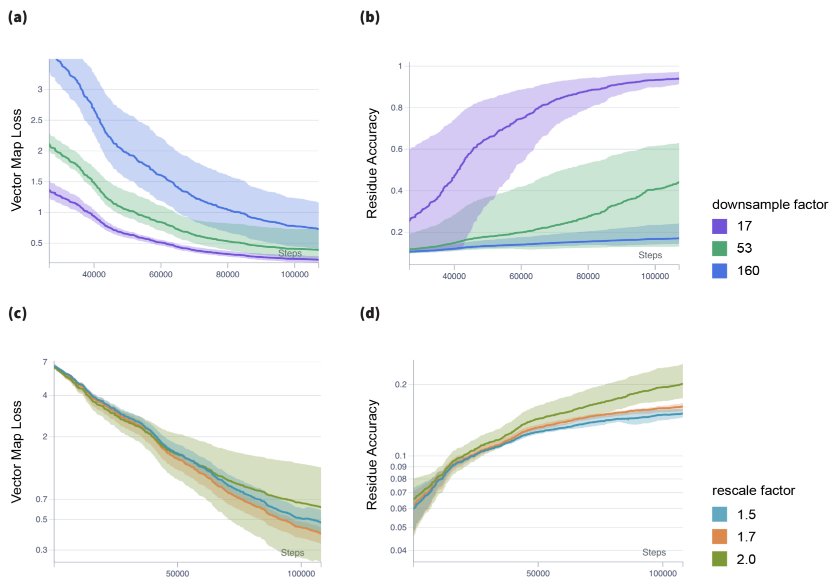

Figure 10: **Ablation Training Curves.** We plot metrics across 10 training epochs for our ablated models from Table 3. **(a-b)** compares models across downsampling factors and highlights the trade-off between downsampling and reconstruction. **(c-d)** compares different rescaling factors for fixed downsampling at 160.

8.1 Regularization

When training autoencoder models for latent diffusion, we regularize the learned latent space so that representations are amenable to the relevant range scales of the source distribution $\mathcal{N}(0, 1)$. Let $\mathbf{V}_i^l$ denote the i-th channel of a vector representation $\mathbf{V}^l$. We regularize the autoencoder latent space by optimizing radial and angular components of our vectors:

$$\mathcal{L}_{\text{reg}} = \sum_i \text{ReLU}(1 - ||\mathbf{V}_i^l||_1^1) + \sum_i \sum_{i \neq j} (\mathbf{V}_i^l \cdot \mathbf{V}_j^l)$$

The first term penalizes vector magnitudes larger than one, and the second term induces vectors to spread angularly. We find these regularizations to significantly help training of the denoising diffusion model.

9 Details on Autoencoder Comparison

We implement the comparison EGNN model in its original form, following [Satorras et al. (2022)]. For downsampling and upsampling we follow the algorithm described in [Fu et al. (2023)]. For this comparison, we train the models to minimize the residue label cross entropy and the vector map loss (Appendix 7).

Ophiuchus significantly outperforms the EGNN-based architecture. The standard EGNN is SO(3)-equivariant with respect to its positions, but models features with SO(3)-invariant representations. As part of its message passing, EGNN uses relative vectors between nodes to update positions. However, when downsampling positions, the total number of relative vectors available reduces quadratically, making it increasingly challenging for the network to recover coordinates. Our method instead uses SO(3)-equivariant feature representations, and is able to store three-dimensional vector information directly in features as it coarsens the positions.

9.1 Architecture Ablation

To investigate the effects of different architecture layouts and coarsening rates, we train different instantiations of Ophiuchus to coarsen representations of contiguous 160-sized protein fragments from the Protein Databank (PDB) [Berman et al. (2000)]. We filter out entries tagged with resolution higher than 2.5 Å, and total sequence lengths larger than 512. We use sequence convolutions with fixed kernel size $K = 5$ and stride $S = 3$. The initial residue feature representation is fixed to dimensionality of 16 channels, where each channel is composed of one scalar ($l = 0$) and one 3D vector ($l = 1$). We investigate network performance for rescaling factors $\rho \in \{1.5, 1.7, 2.0\}$, and for depths/downsampling number of layers $L \in \{3, 4, 5\}$. All experiments were repeated 3 times with different initialization seeds and data shuffles.

In our experiments we find a trade-off between domain coarsening factor and reconstruction performance. In Table 3 we show that although residue recovery suffers from large downsampling factors, structure recovery rates remain comparable between various settings. Moreover, we find that we are able to model all atoms in proteins, as opposed to only $\mathbf{C}\alpha$ atoms, and still recover the structure with high precision. These results demonstrate that protein data can be captured effectively and efficiently through learned coarse representations. We additionally provide corresponding results for backbone-only models in table 4.

Table 3: **Ophiuchus Recovery from Compressed Representations - All-Atom**

Downsampling Factor	**Channels/Layer**	**# Params [1e6]**	**Cα-RMSD** (Å) $\downarrow$	**All-Atom RMSD** (Å) $\downarrow$	**GDT-TS** $\uparrow$	**GDT-HA** $\uparrow$	**Residue Acc.** (%) $\uparrow$
17	[16, 24, 36]	0.34	0.90 ± 0.20	0.68 ± 0.08	94 ± 3	76 ± 4	97 ± 2
17	[16, 27, 45]	0.38	0.89 ± 0.21	0.70 ± 0.09	94 ± 3	77 ± 5	98 ± 1
17	[16, 32, 64]	0.49	1.02 ± 0.25	0.72 ± 0.09	92 ± 4	73 ± 5	98 ± 1
53	[16, 24, 36, 54]	0.49	1.03 ± 0.18	0.83 ± 0.10	91 ± 3	72 ± 5	60 ± 4
53	[16, 27, 45, 76]	0.67	0.92 ± 0.19	0.77 ± 0.09	93 ± 3	75 ± 5	66 ± 4
53	[16, 32, 64, 128]	1.26	1.25 ± 0.32	0.80 ± 0.16	86 ± 5	65 ± 6	67 ± 5
160	[16, 24, 36, 54, 81]	0.77	1.67 ± 0.24	1.81 ± 0.16	77 ± 4	54 ± 5	17 ± 3
160	[16, 27, 45, 76, 129]	1.34	1.39 ± 0.23	1.51 ± 0.17	83 ± 4	60 ± 5	17 ± 3
160	[16, 32, 64, 128, 256]	3.77	1.21 ± 0.25	1.03 ± 0.15	87 ± 5	65 ± 6	27 ± 4

In Fig. 10.a we compare the relative vector loss between ground truth coordinates and the coordinates reconstructed from the autoencoder with respect to different downsampling factors. We average over different trials and channel rescale factors. In Fig 10.b we similarly plot the residue recovery accuracy with respect to different downsampling factors. In Fig. 10.c we compare the structure reconstruction accuracy across different channel rescaling factors. Finally, in Fig. 10.d we compare the residue recovery accuracy across different rescaling factors.

Table 4: **Recovery rates from bottleneck representations - Backbone only**

Factor	**Channels/Layer**	**# Params [1e6]** $\downarrow$	**Cα-RMSD** (Å) $\downarrow$	**GDT-TS** $\uparrow$	**GDT-HA** $\uparrow$
17	[16, 24, 36]	0.34	0.81 ± 0.31	96 ± 3	81 ± 5
17	[16, 27, 45]	0.38	0.99 ± 0.45	95 ± 3	81 ± 6
17	[16, 32, 64]	0.49	1.03 ± 0.42	92 ± 4	74 ± 6
53	[16, 24, 36, 54]	0.49	0.99 ± 0.38	92 ± 5	74 ± 8
53	[16, 27, 45, 76]	0.67	1.08 ± 0.40	91 ± 6	71 ± 8
53	[16, 32, 64, 128]	1.26	1.02 ± 0.64	92 ± 9	75 ± 11
160	[16, 24, 36, 54, 81]	0.77	1.33 ± 0.42	84 ± 7	63 ± 8
160	[16, 27, 45, 76, 129]	1.34	1.11 ± 0.29	89 ± 4	69 ± 7
160	[16, 32, 64, 128, 256]	3.77	0.90 ± 0.44	94 ± 7	77 ± 9

10 Latent Space Analysis

10.1 Visualization of Latent Space

To visualize the learned latent space of Ophiuchus, we forward 50k samples from the training set through a large 485-length model (Figure 11). We collect the scalar component of bottleneck representations, and use t-SNE to produce 2D points for coordinates. We similarly produce 3D points

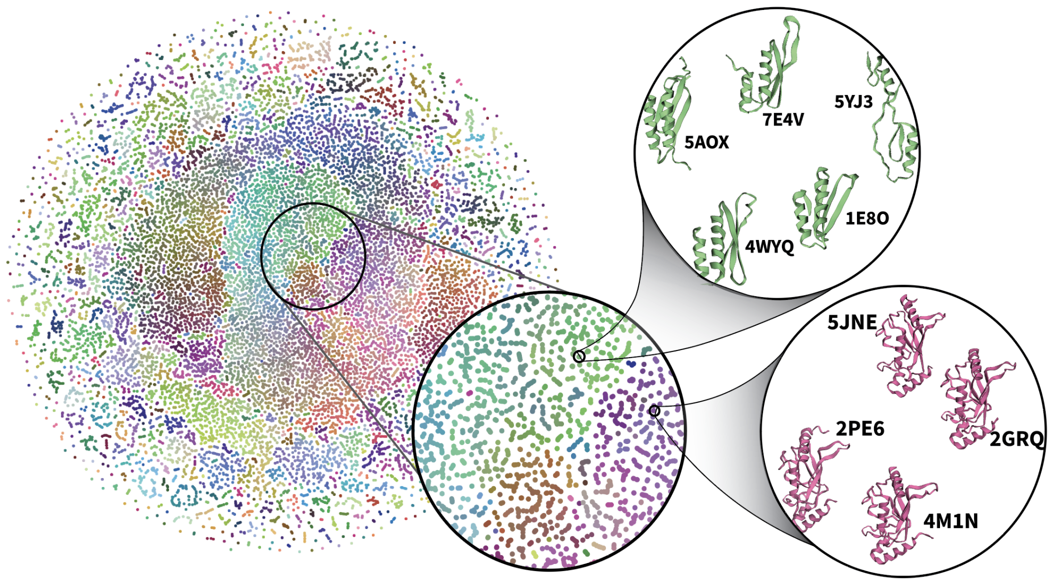

Figure 11: **Latent Space Analysis**: we qualitatively examine the unsupervised clustering capabilities of our representations through t-SNE.

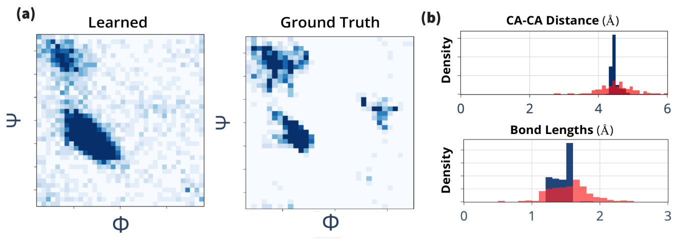

Figure 12: **MiniProtein Marginal Distributions (a)** Ramachandran plots of sampled (left) and ground (right) distributions. **(b)** Comparison of Cα-Cα distances and all-atom bond lengths between learned (red) and ground (blue) distributions.

and use those for coloring. The result is visualized in Figure 11. Visual inspection of neighboring points reveals unsupervised clustering of similar folds and sequences.

11 Latent Diffusion Benchmark Details

11.1 Datasets

Our tested models are trained on two different datasets. The MiniProtein scaffolds dataset consists of 66k all-atom structures of sequence length between 50 and 65, and composes of diverse folds and sequences across 5 secondary structure topologies, being introduced in [Cao et al. (2022)]. We also train a large sequence model on the data curated by [Yim et al. (2023)], which consists of approximately 22k proteins, to compare Ophiuchus to the performance of FrameDiff and RFDiffusion in backbone generation for proteins up to 485 residues.

11.2 Benchmark Metrics

For all comparisons we run all models on a single RTX6000 GPU. To compute the scTM scores [Trippe et al. (2023)], we recover 8 sequences using ProteinMPNN [Dauparas et al. (2022)] for 500 sampled backbones from the tested diffusion models. We use a sampling temperature of 0.1

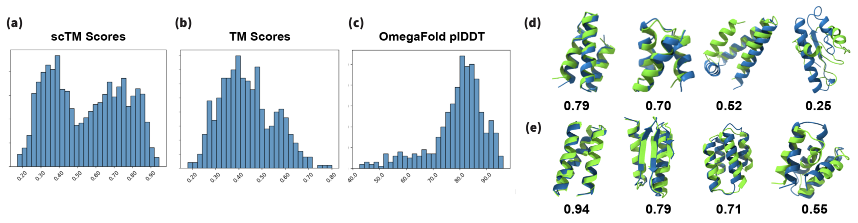

Figure 13: **Self-Consistency Template Matching Scores for Ophiuchus Diffusion**. **(a)** Distribution of scTM scores for 500 sampled backbone. **(b)** Distribution of TM-Scores of jointly sampled backbones and sequences from an all-atom diffusion model and corresponding OmegaFold models. **(c)** Distribution of average plDDT scores for 500 sampled backbones **(d)** TM-Score between sampled backbone (green) and OmegaFold structure (blue) **(e)** TM-Score between sampled backbone (green) and the most confident OmegaFold structure of the sequence recovered from ProteinMPNN (blue)

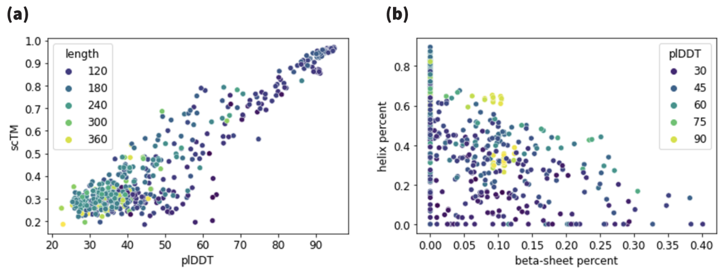

Figure 14: **Designability of Sampled Backbones** . **(a)** To analyze the designability of our sampled backbones we show a plot of scTM vs plDDT. **(b)** To analyze the composition of secondary structures in the samples we show a plot of helix percentage in a sample vs beta-sheet percentage

for ProteinMPNN. Unlike the original work, where the sequences where folded using AlphaFold2 [Jumper et al. (2021)], we use OmegaFold [Wu et al. (2022b)] similar to [Lin & AlQuraishi (2023)]. The predicted structures are aligned [Zhang & Skolnick (2005)] to the original sampled backbones and TM-Score and RMSD is calculated for each alignment. To calculate the diversity measurement, we hierarchically clustered samples using MaxCluster [Siew et al. (2000)]. Diversity is defined as the number of clusters divided by the total number of samples, as described in [Yim et al. (2023)].

In Figure 12 we compare the marginal distribution of our Miniprotein predictions and ground truth data. In Figure 13 we show the various distribution for self-consistency for joint sampling of sequence and all-atom structure by the diffusion model. We further include metrics on designability for large sampled backbones in Figure 14.

12 Visualization of All-Atom Reconstructions

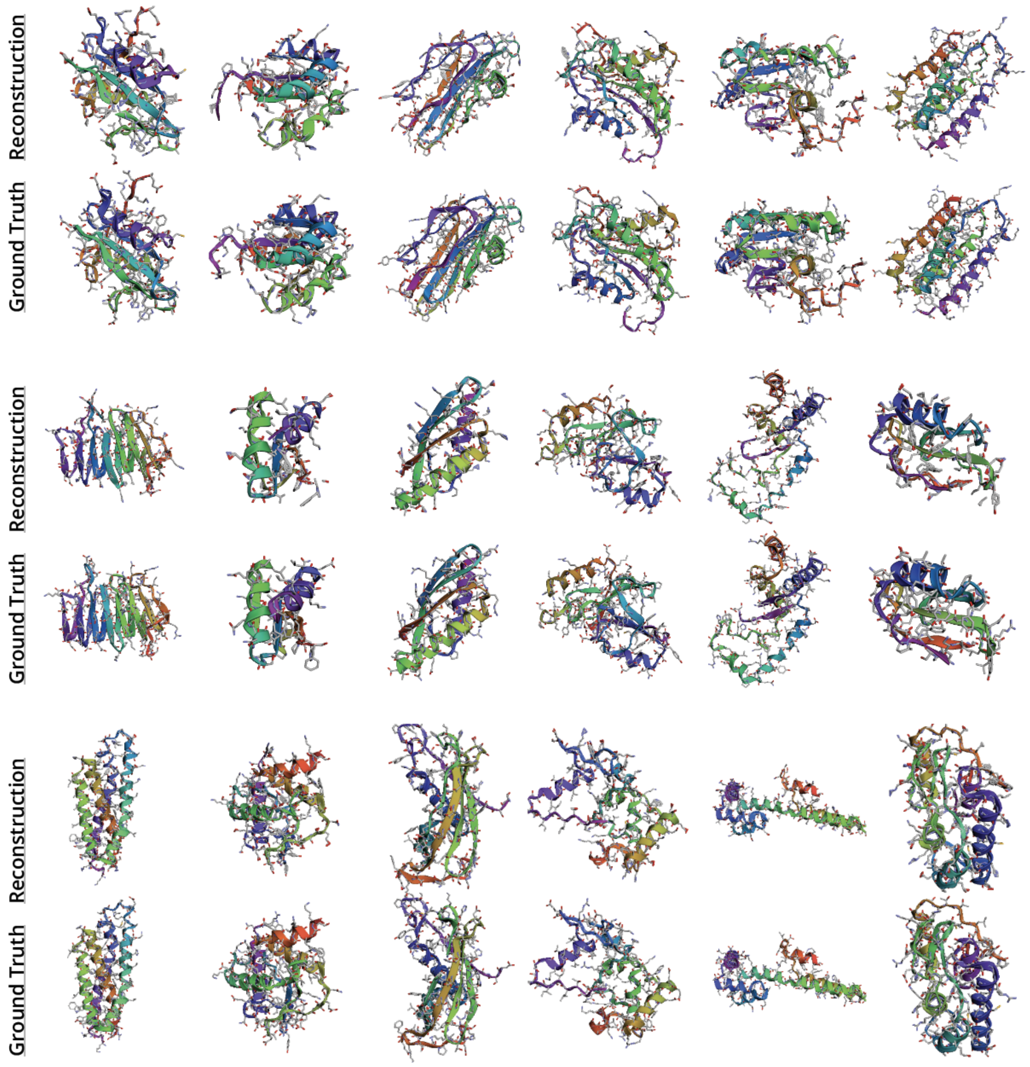

Figure 15: **Reconstruction of 128-length all-atom proteins**. Model used for visualization reconstructs all-atom protein structures from coarse representations of 120 scalars and 120 3D vectors.

13 Visualization of Backbone Reconstructions

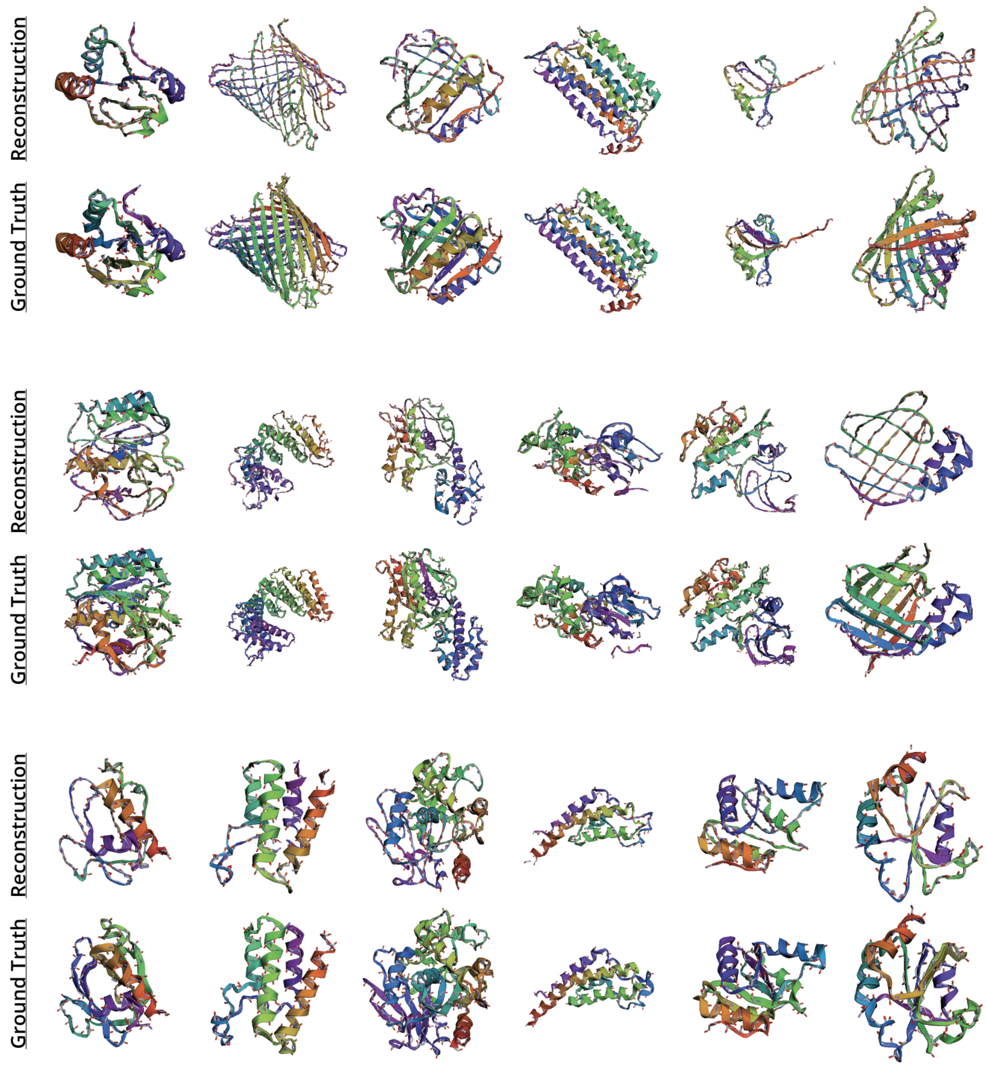

Figure 16: **Reconstruction of 485-length protein backbones**. Model used for visualization reconstructs large backbones from coarse representations of 160 scalars and 160 3D vectors.

14 VISUALIZATION OF RANDOM BACKBONE SAMPLES

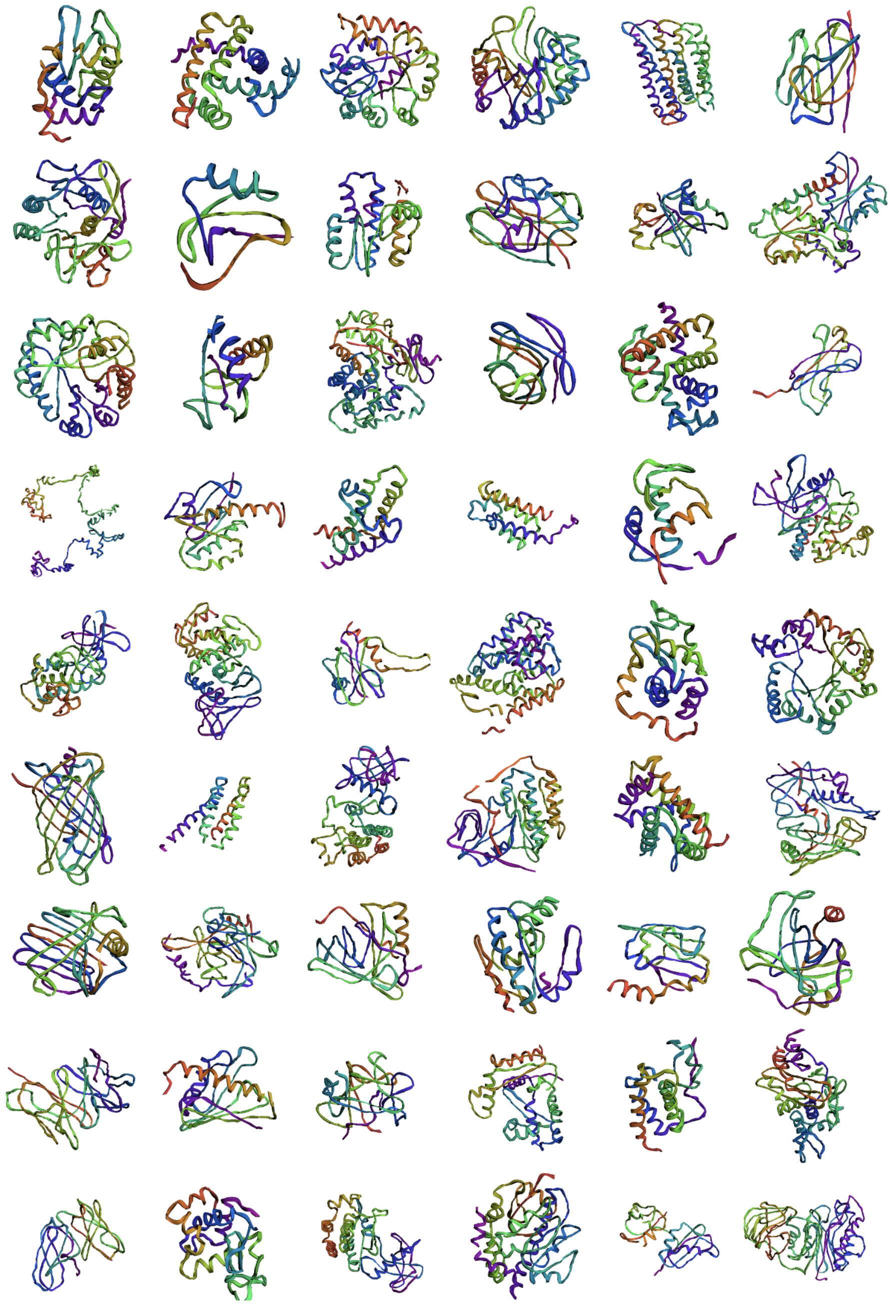
Figure 17: **Random Samples from an Ophiuchus 485-length Backbone Model**. Model used for visualization samples SO(3) representations of 160 scalars and 160 3D vectors. We measure 0.46 seconds to sample a protein backbone.