Towards End-to-End Learning of Protein Structure Prediction and Structure-based Sequence Design

Julius Wenckstern
École Polytechnique Fédérale de Lausanne
julius.wenckstern@epfl.ch

Bruno Correia
École Polytechnique Fédérale de Lausanne
bruno.correia@epfl.ch

Abstract

Deep-learning-based methods have revolutionized the way we address protein structure prediction and structure-based sequence design. Despite their success, current methods still face limitations. Protein structure prediction requires large models, which often act as bottlenecks in protein design workflows. Design methods prioritizes the optimization of sequence recovery as a surrogate for structure recovery. We address these limitations in our model E2EFOLD by learning both tasks end-to-end in a discrete, stochastic autoencoder. E2EFOLD is trained to reconstruct an input backbone and predict sidechain conformations. An auxiliary sequence recovery objective guides the encoder to predict a sequence distribution conditioned on the backbone. Discrete sequences are sampled differentiably from this distribution and passed to the decoder for structure prediction. We find that our end-to-end framework enables significantly improved sequence design self-consistency. On designed sequences, our model's structure prediction correlates with Boltz-2's while relying on more than one order of magnitude fewer parameters. Taken together, these results suggest a promising framework for the advancement of protein structure prediction and sequence design.

1 Introduction

Deep-learning-based methods have made outstanding contributions to two biological problems: the protein structure prediction problem, which involves predicting a protein's three-dimensional structure from its amino acid sequence [Anfinsen, 1973, Dill et al., 2008], and its inverse, the structure-based protein design (or "inverse folding") problem [Pabo, 1983], which involves predicting an amino acid sequence that will fold into a given backbone structure.

In structure prediction, methods such as AlphaFold 2 [Jumper et al., 2021] and RoseTTAFold [Baek et al., 2021] have demonstrated unprecedented accuracy for a previously untractable problem. These methods employ large transformer-based architectures and leverage co-evolutionary information from multiple-sequence alignments (MSAs). An alternative approach to structure prediction, as exemplified by ESMFold [Lin et al., 2023], uses embeddings of protein language models (PLMs) in place of MSAs.

All these methods incur high computational costs, frequently creating bottlenecks in protein design pipelines where protein structure prediction is essential for the selection of the best sequences. Developing more computationally efficient models has emerged as an active area of research. Whereas previous approaches have focused on the optimization of model architecture and implementation [Cheng et al., 2022, Wohlwend et al., 2023], we restrict the protein sequences to be predicted to "designed" sequences. These are commonly characterized by reduced complexity and more explicit

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Accelerated Materials Design (AI4Mat).

encoding of protein structure than their native counterparts. Focusing on these sequences allows for accurate structure prediction by smaller models. Furthermore, the explicit sequence encoding enables structure prediction without relying on MSAs or PLM embeddings [Chu et al., 2024].

Structure-based sequence design has been addressed by a plethora of methods, originally physics and heuristics-based [Leaver-Fay et al., 2011], and more recently employing graph neural networks optimized for node classification with the canonical amino acids as node labels and the geometry between residue atoms as common primary input feature [Ingraham et al., 2019, Dauparas et al., 2022]. Existing models share sequence recovery as the primary training objective.

As noted in previous publications [Ingraham et al., 2019, Ektefaie et al., 2024], this training objective is theoretically insufficient due to the degenerate relationship between protein structure and sequence. Numerous sequences can fold into a given backbone [Sander and Schneider, 1991, Li et al., 1996], so higher sequence recovery does not necessarily indicate better sequence design performance. Conversely, single mutations can cause misfolding or disrupt a structure, demonstrating that high sequence recovery is not indicative of successful sequence design. By training a sequence design method primarily with a structure recovery loss, we intend to account for the degenerate relationship between sequence and structure.

To accurately and efficiently predict protein structure for designed sequences, and to account for sequence-structure degeneracy in sequence design, we propose learning both problems end-to-end with our method E2EFOLD. Specifically, we train a discrete, stochastic protein structure autoencoder in which the encoder learns structure-based sequence design, and the decoder structure prediction of designed sequences. The autoencoder is optimized to reconstruct the input backbone structure and predict side-chain conformations. An auxiliary sequence recovery objective guides the encoder to output sequence distributions conditioned on the backbone. Discrete sequences are sampled differentiably from these distributions using the Gumbel-Softmax trick and passed to the decoder.

We aim to examine the effects of learning structure-based sequence design and structure prediction of designed sequences end-to-end. Accordingly, we evaluate E2EFOLD's performance on both tasks across multiple datasets, ablate its structure recovery component, and compare with standard methods.

We summarize main contributions and relevance of this work as follows.

Conceptual Novelty: To our knowledge E2EFOLD is first to learn structure-based sequence design followed by structure prediction of the designed sequences in an end-to-end differentiable model.

Performance: Our approach improves self-consistency in sequence design and achieves competitive structure prediction with substantially smaller model size.

Practical Relevance: Advances in both problems can improve biological understanding and accelerate progress in areas such as drug and material discovery, synthetic biology, and enzyme engineering.

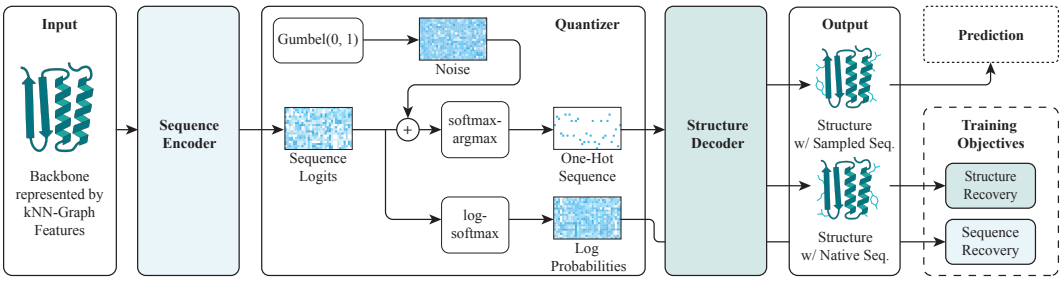

Figure 1: **Model overview.** E2EFOLD learns structure prediction and structure-based sequence design end-to-end in a discrete, stochastic autoencoder.

2 Method

Our end-to-end discrete, stochastic autoencoder framework is illustrated in Figure 1. We first describe the framework and then discuss key aspects of architecture and training. Pseudocode for the architecture, training details and hyperparameters are available in Appendix A.1.

2.1 End-to-End Framework

Let $\mathbf{X} \in \mathbb{R}^{N\times 4\times 3}$ represent the native backbone atomic coordinates (N, C_α, C, and O) of a protein with N residues and let $\mathbf{S} = [\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_N] \in \mathbb{R}^{N\times 20}$ be its sequence, where each $\mathbf{S}_i \in \{0,1\}^{20}$ is a one-hot vector encoding the amino acid type at residue i. Let (ϕ, ψ) be learnable parameters of our model.

During a forward pass, the encoder e_ϕ, conditioned on $\mathbf{X}$, predicts logits $\mathbf{L} \in \mathbb{R}^{N\times 20}$ to parametrize categorical distributions over all amino acid types $j \in \{1, 2, \ldots, 20\}$ at each residue i leading to the following probability distribution:

$$p(\mathbf{S} \mid \mathbf{X}; \phi) = \prod_{i=1}^{N} \text{Cat}\left(\mathbf{S}_i \,\middle|\, \left(\frac{\exp(L_{i,j})}{\sum_{k=1}^{20}\exp(L_{i,k})}\right)_{j=1}^{20}\right),$$

The quantizer q samples a discrete sequence $\tilde{\mathbf{S}}$ at temperature τ from this distribution using the Gumbel-Softmax trick [Jang et al., 2016, Maddison et al., 2016] and straight-through gradient estimation [Bengio et al., 2013]:

$$\tilde{\mathbf{S}}_i = \text{one\_hot}\left(\underset{j\in\{1,\ldots,20\}}{\text{argmax}}\left(\frac{\exp\left(\frac{L_{i,j}+G_{i,j}}{\tau}\right)}{\sum_{k=1}^{20}\exp\left(\frac{L_{i,k}+G_{i,k}}{\tau}\right)}\right)\right), \quad G_{i,j} \sim \text{Gumbel}(0,1).$$

The decoder d_ψ, conditioned on $\tilde{\mathbf{S}}$, predicts rigid frames and torsion angles at each residue position for all 20 amino acid types. From these predictions, it constructs all-atom structures based on either the native or a sampled sequence. The native-derived structure $\dot{\mathbf{X}}_{\text{all-atom}} \in \mathbb{R}^{N\times 14\times 3}$ has a one-to-one correspondence with the reference and is used to compute reconstruction losses during training. In contrast, the sampled-sequence structure $\tilde{\mathbf{X}}_{\text{all-atom}} \in \mathbb{R}^{N\times 14\times 3}$ lacks this correspondence because side chains differ, and therefore cannot be used for reconstruction losses. Instead, it serves as the model's predicted structure at inference.

The model is optimized with the loss function $\mathcal{L}$ and loss coefficients $(\alpha, \beta, \gamma, \delta, \epsilon)$

$$\mathcal{L} = \alpha\mathcal{L}_{\text{FAPE}} + \beta\mathcal{L}_{C_\alpha-\text{FAPE}} + \gamma\mathcal{L}_{\text{Torsion}} + \delta\mathcal{L}_{\text{Distogram}} + \epsilon\mathcal{L}_{\text{Sequence Recovery}},$$

where $\mathcal{L}_{\text{Sequence Recovery}}$ is a categorical cross-entropy loss between the native sequence $\mathbf{S}$ and the encoder's predicted distribution and the other structure terms are adapted from Jumper et al. [2021].

2.2 Architecture

Encoder We adopt the feature creation procedure and architecture from ProteinMPNN [Dauparas et al., 2022], but intercalate an attention layer between the node and position-wise feedforward layers. Instead of the random-order autoregressive decoding scheme in ProteinMPNN, we employ simultaneous prediction at all residue positions.

Decoder We adapt the ESMFold [Lin et al., 2023] architecture. This architecture contains by default three parts, a structure module implementation from OpenFold [Ahdritz et al., 2024, Jumper et al., 2021, Evans et al., 2021], a Pairformer Trunk and a pretrained ESM-2 PLM. We drastically downsize the Pairformer Trunk and remove the PLM integration. Given that we train on full-length proteins of maximally 412 residues, we increase the number of recycles to permit the model to expand the protein frame gas initialized at the origin over more iterations. Lastly, we modify the structure module to reconstruct all-atom coordinates either based on the native sequence or the sampled sequence.

2.3 Data

We train and evaluate models on splits by Ingraham et al. [2019] of CATH 4.2 [Orengo et al., 1997] and E2EData, a custom-curated dataset from the Protein Data Bank [wwPDB Consortium, 2019]. We also evaluate our model on RFData, a set of RFDiffusion [Watson et al., 2023] generated backbones.

3 Results

We assess structure-based sequence design using two metrics: (i) sequence recovery and (ii) self-consistency, for which we compute TM-score and RMSD between the reference structure and the Boltz-2 prediction of the designed sequence [Passaro et al., 2025]. To quantify the contribution of structural supervision, we ablate the structural loss terms. For protein structure prediction of designed sequences, a newly defined task without tailored baselines, we use Boltz-2 as the reference method and report prediction accuracy in RMSD to the reference structure.

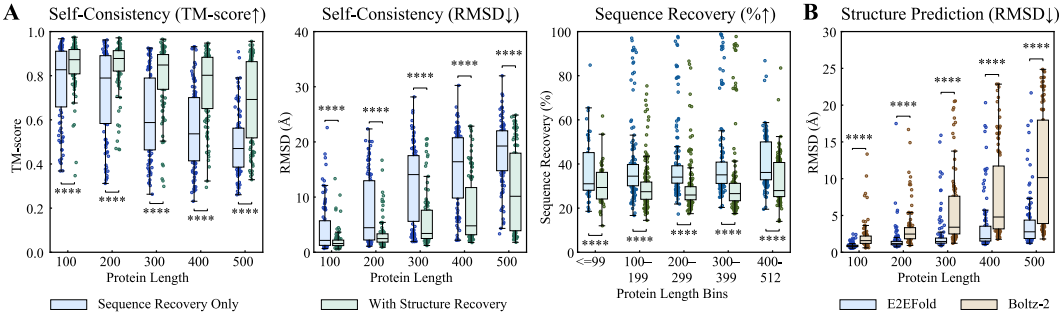

Figure 2: **Results**. (**A**) Effect of training with a structure recovery objective on self-consistency (TM-score, RMSD; evaluated on RFData) and sequence recovery (evaluated on E2EData). (**B**) Comparison of E2EFOLD structure prediction with Boltz-2's on E2EFOLD's designed sequences (evaluated on E2EData). Significance is assessed by a paired Wilcoxon signed-rank test ($* \, p < 0.05$, $** \, p < 0.01$, $*** \, p < 0.001$, $**** \, p < 10^{-4}$).

Sequence Recovery On CATH 4.2, E2EFOLD achieves 25.15% mean sequence recovery, below prior work (see Appendix, Table 2), but self-consistency is the more relevant metric for this task. Training and evaluation on the larger E2EData splits improves recovery to 31.01%, suggesting that structural context beyond single domains is important for end-to-end learning.

Self-Consistency On the RFData test split, E2EFOLD attains an average TM-score of 0.78 and RMSD of 6.02 Å. On E2EData, the scores are 0.64 (TM-score) and 8.84 Å (RMSD).

Ablation of Structure Recovery Adding of the structure-recovery objective (Figure 1A) shows significantly improved self-consistency (TM-score, RMSD) on RFData at reduced sequence recovery on E2EData. The improvement in self-consistency persists accross all protein lengths and increases as a function of it.

Comparison with Boltz-2 E2EFOLD's decoder achieves on average more accurate structure prediction of designed sequences than Boltz-2 (Figure 2; 2.54 Å vs. 6.03 Å RMSD). This comparison is however not straightforward, since our model was trained specifically on sequences from its encoder. Prediction errors correlate with those of Boltz-2 on designed sequences (Appendix, Figure 3; Spearman's $\rho = 0.60$ on RFData and $\rho = 0.60$ on E2EData), while still yielding higher accuracy overall. Notably, our decoder has only 7.8 million parameters, compared to 432 million for Boltz-2 (excluding its confidence model).

4 Conclusion

E2EFOLD is a lightweight method for structure-based sequence design and accurate structure prediction of designed sequences. Despite being at an early stage, it already shows promising performance on both tasks, and ablations of the structure loss confirm its benefit for sequence design. Going forward, we aim to address current limitations by optimizing the sequence decoding strategy, enabling conditional sequence design, and reformulating structure prediction as a generative task with confidence estimation. In parallel, we plan more extensive comparisons with established baselines and additional ablations. We will examine with particular attention how the relative weighting of sequence and structure loss components affects performance.

Acknowledgments

The authors would like to thank Johann Wenckstern, Ilia Igashov, Adrian Dobbelstein, Arne Schneuing and Rebecca Neeser for their helpful feedback. This work was supported by the Swiss National Science Foundation (Grant number: 213750) and by a grant from the Swiss National Supercomputing Center (SwissAI project ID: a132).

References

Gustaf Ahdritz, Nazim Bouatta, Christina Floristean, Sachin Kadyan, Qinghui Xia, William Gerecke, Timothy J O'Donnell, Daniel Berenberg, Ian Fisk, Niccolò Zanichelli, et al. Openfold: Retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature Methods*, 21(8):1514–1524, 2024.

Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096): 223–230, 1973.

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, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Shenggan Cheng, Xuanlei Zhao, Guangyang Lu, Jiarui Fang, Zhongming Yu, Tian Zheng, Ruidong Wu, Xiwen Zhang, Jian Peng, and Yang You. Fastfold: Reducing alphafold training time from 11 days to 67 hours. *arXiv preprint arXiv:2203.00854*, 2022.

Alexander E Chu, Tianyu Lu, and Po-Ssu Huang. Sparks of function by de novo protein design. *Nature biotechnology*, 42(2):203–215, 2024.

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.

Ken A Dill, S Banu Ozkan, M Scott Shell, and Thomas R Weikl. The protein folding problem. *Annu. Rev. Biophys.*, 37(1):289–316, 2008.

Yasha Ektefaie, Olivia Viessmann, Siddharth Narayanan, Drew Dresser, J Mark Kim, and Armen Mkrtchyan. Reinforcement learning on structure-conditioned categorical diffusion for protein inverse folding. *arXiv preprint arXiv:2410.17173*, 2024.

Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, et al. Protein complex prediction with alphafold-multimer. *biorxiv*, pages 2021–10, 2021.

Zhangyang Gao, Cheng Tan, Pablo Chacón, and Stan Z Li. Pifold: Toward effective and efficient protein inverse folding. *arXiv preprint arXiv:2209.12643*, 2022.

Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pages 8946–8970. PMLR, 2022.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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.

Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian W Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, et al. Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. In *Methods in enzymology*, volume 487, pages 545–574. Elsevier, 2011.

Hao Li, Robert Helling, Chao Tang, and Ned Wingreen. Emergence of preferred structures in a simple model of protein folding. *Science*, 273(5275):666–669, 1996.

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.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Christine A Orengo, Alex D Michie, Susan Jones, David T Jones, Mark B Swindells, and Janet M Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

Carl Pabo. Molecular technology: designing proteins and peptides. *Nature*, 301(5897):200–200, 1983.

Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, et al. Boltz-2: Towards accurate and efficient binding affinity prediction. *BioRxiv*, pages 2025–06, 2025.

Chris Sander and Reinhard Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics*, 9 (1):56–68, 1991.

Richard W Shuai, Talal Widatalla, Po-Ssu Huang, and Brian L Hie. Sidechain conditioning and modeling for full-atom protein sequence design with fampnn. *bioRxiv*, pages 2025–02, 2025.

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, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Jeremy Wohlwend, Mateo Reveiz, Axel Feldmann, Wengong Jin, and Regina Barzilay. Minifold: Simple, fast and accurate protein structure prediction. *OpenReview*, 2023.

wwPDB Consortium. Protein data bank: the single global archive for 3d macromolecular structure data. *Nucleic acids research*, 47(D1):D520–D528, 2019.

Appendix for "Towards End-to-End Learning of Protein Structure Prediction and Structure-Based Sequence Design"

Table of Contents

A.1 Extended Methods

A.1.1 Architecture

E2EFOLD's architecture comprises three parts: the *encoder* (Algorithms 1, 2, 3), the *quantizer* (Algorithm 4), and the *decoder* (Algorithms 5, 6, 7). We use $\mathbf{V}$ for the single/node representation, and $\mathbf{E}$ for pair/edge representations. Model predictions for the *native* sequence are marked with a dot ($\dot{}$), and predictions for the *sampled* sequence with a tilde ($\tilde{}$). The pseudocode for the StructureModule is adapted from Jumper et al. [2021] and the pseudocode for Decoder and FoldingBlock is adopted from Lin et al. [2023].

Algorithm 1 Encoder

def Encoder($\mathbf{V} \in \mathbb{R}^{N \times c_V}$, $\mathbf{E} \in \mathbb{R}^{N \times k \times c_E}$, $N_{\text{Layers}} = 3$)
  for $m = 1, \ldots, N_{\text{Layers}}$ **do**
    $(\mathbf{V}, \mathbf{E}) \leftarrow \text{MPNNEncoderLayer}_m(\mathbf{V}, \mathbf{E})$
  end for
  for $m = 1, \ldots, N_{\text{Layers}}$ **do**
    $(\mathbf{V}, \mathbf{E}) \leftarrow \text{MPNNDecoderLayer}_m(\mathbf{V}, \mathbf{E})$
  end for
  $\mathbf{L} \in \mathbb{R}^{N \times 20} \leftarrow \text{Linear}(\mathbf{V})$
return $\mathbf{L}$

Algorithm 2 MPNNEncoderLayer

def MPNNEncoderLayer($\mathbf{V} \in \mathbb{R}^{N \times c_V}$, $\mathbf{E} \in \mathbb{R}^{N \times k \times c_E}$)
  # Node message passing
  $\mathbf{H} \leftarrow \left[\text{concat}(\mathbf{V}_i,\ \mathbf{V}_j,\ \mathbf{E}_{ij})\right]_{i=1..N,\ j=1..k} \in \mathbb{R}^{N \times k \times (2c_V + c_E)}$
  $\mathbf{M} \leftarrow \text{MLP}_{\text{node}}(\mathbf{H})$
  $\mathbf{M} \leftarrow \sum_{j=1}^{k} \mathbf{M}$
  $\mathbf{V} \leftarrow \text{LayerNorm}\big(\mathbf{V} + \text{Dropout}_{0.1}(\mathbf{M})\big)$
  # FeedForward
  $\mathbf{F} \leftarrow \text{MLP}_{\text{FeedForward}}(\mathbf{V})$
  $\mathbf{V} \leftarrow \text{LayerNorm}\big(\mathbf{V} + \text{Dropout}_{0.1}(\mathbf{F})\big)$
  # Edge message passing
  $\mathbf{H} \leftarrow \left[\text{concat}(\mathbf{V}_i,\ \mathbf{V}_j,\ \mathbf{E}_{ij})\right]_{i=1..N,\ j=1..k} \in \mathbb{R}^{N \times k \times (2c_V + c_E)}$
  $\mathbf{M} \leftarrow \text{MLP}_{\text{edge}}(\mathbf{H})$
  $\mathbf{E} \leftarrow \text{LayerNorm}\big(\mathbf{E} + \text{Dropout}_{0.1}(\mathbf{M})\big)$
return $\mathbf{V}$, $\mathbf{E}$

Algorithm 3 MPNNDecoderLayer

def MPNNDecoderLayer($\mathbf{V} \in \mathbb{R}^{N \times c_V}$, $\mathbf{E} \in \mathbb{R}^{N \times k \times c_E}$)
  # Node message passing
  $\mathbf{H} \leftarrow \left[\text{concat}(\mathbf{V}_i,\ \mathbf{V}_j,\ \mathbf{E}_{ij})\right]_{i=1..N,\ j=1..k} \in \mathbb{R}^{N \times k \times (2c_V + c_E)}$
  $\mathbf{M} \leftarrow \text{MLP}_{\text{node}}(\mathbf{H})$
  $\mathbf{M} \leftarrow \sum_{j=1}^{k} \mathbf{M}$
  $\mathbf{V} \leftarrow \text{LayerNorm}\big(\mathbf{V} + \text{Dropout}_{0.1}(\mathbf{M})\big)$
  # Node attention
  $\mathbf{V} \leftarrow \mathbf{V} + \text{Dropout}_{0.1}\big(\text{SelfAttention}(\mathbf{V})\big)$
  # FeedForward
  $\mathbf{F} \leftarrow \text{MLP}_{\text{FeedForward}}(\mathbf{V})$
  $\mathbf{V} \leftarrow \text{LayerNorm}\big(\mathbf{V} + \text{Dropout}_{0.1}(\mathbf{F})\big)$
return $\mathbf{V}$, $\mathbf{E}$

Algorithm 4 Quantizer

def Quantizer($\mathbf{L} \in \mathbb{R}^{N \times 20}$, $\mathbf{S} \in \{0,1\}^{N \times 20}$, $\tau > 0$)
  # Sample Gumbel noise for all residues and amino acid types
  $\mathbf{G} \sim \text{Gumbel}(0,1)^{N \times 20}$
  # Gumbel softmax - reparametrization trick
  $\mathbf{Y} \leftarrow \text{softmax}\big((\mathbf{L} + \mathbf{G})/\tau\big) \in \mathbb{R}^{N \times 20}$
  # Discretization with straight-through gradient estimation
  $\tilde{\mathbf{S}} \leftarrow \text{one\_hot}\big(\arg\max_j \mathbf{Y}_{:,j}\big) \in \{0,1\}^{N \times 20}$
  $\tilde{\mathbf{S}} \leftarrow \text{stopgrad}(\tilde{\mathbf{S}} - \mathbf{Y}) + \mathbf{Y}$
  # Log-probabilities for sequence recovery loss
  $\mathbf{Q} \leftarrow \log\big(\text{softmax}(\mathbf{L}/\tau)\big) \in \mathbb{R}^{N \times 20}$
  $\mathcal{L}_{\text{Sequence Recovery}} \leftarrow -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{20} S_{i,j}\, Q_{i,j}$
return $\tilde{\mathbf{S}}$, $\mathcal{L}_{\text{Sequence Recovery}}$

Algorithm 5 Decoder

def Decoder$\big(\tilde{\mathbf{S}} \in \{0,1\}^{N \times 20}$, $(\mathbf{X}_{\text{all-atom}}, \mathbf{X}^{\text{alt.}}_{\text{all-atom}}) \in \mathbb{R}^{N \times 14 \times 3}$,
  $(\mathbf{T}_{\text{all-atom}}, \mathbf{T}^{\text{alt.}}_{\text{all-atom}}) \in \text{SE}(3)^{N \times 7}$, $(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{\text{alt.}}) \in \mathbb{R}^{N \times 7 \times 2}\big)$
  $\mathbf{V} \leftarrow \text{Linear}(\tilde{\mathbf{S}})$
  $(\mathbf{V}^{\text{recycle}}, \mathbf{E}, \mathbf{E}^{\text{recycle}}, \mathbf{D}^{\text{recycle}}) \leftarrow \mathbf{0}$
  for $n = 1, \ldots, N_{\text{recycle}}$ **do** ▷ Shared weights
    $(\mathbf{V}^{\text{recycle}}, \mathbf{E}^{\text{recycle}}, \mathbf{D}^{\text{recycle}}) \leftarrow \text{stopgrad}(\mathbf{V}^{\text{recycle}}, \mathbf{E}^{\text{recycle}}, \mathbf{D}^{\text{recycle}})$
    if $n < N_{\text{recycle}}$ **then**
      $(\mathbf{V}, \mathbf{E}) \leftarrow \text{stopgrad}(\mathbf{V}, \mathbf{E})$
    end if
    $\mathbf{V}^{\text{recycle}} \leftarrow \text{LayerNorm}(\mathbf{V}^{\text{recycle}})$
    $\mathbf{E}^{\text{recycle}} \leftarrow \text{LayerNorm}(\mathbf{E}^{\text{recycle}})$
    $\mathbf{E}^{\text{recycle}} \leftarrow \mathbf{E}^{\text{recycle}} + \text{Embedding}(\mathbf{D}^{\text{recycle}})$
    $\mathbf{V} \leftarrow \mathbf{V} + \mathbf{V}^{\text{recycle}}$
    $\mathbf{E} \leftarrow \mathbf{E} + \mathbf{E}^{\text{recycle}}$
    for $m = 1, \ldots, N_{\text{FoldingBlock}}$ **do**
      $\mathbf{E} \leftarrow \mathbf{E} + \text{PairwiseRelativeSequencePositionalEncoding}(N)$
      $(\mathbf{V}, \mathbf{E}) \leftarrow \text{FoldingBlock}_m(\mathbf{V}, \mathbf{E})$
    end for
    $(\tilde{\mathbf{X}}, \mathcal{L}_{\text{FAPE}}, \mathcal{L}_{\text{C}_\alpha\text{-FAPE}}, \mathcal{L}_{\text{Torsion}})$
      $\leftarrow \text{StructureModule}(\mathbf{V}, \mathbf{E}, (\mathbf{X}_{\text{all-atom}}, \mathbf{X}^{\text{alt.}}_{\text{all-atom}}), (\mathbf{T}_{\text{all-atom}}, \mathbf{T}^{\text{alt.}}_{\text{all-atom}}), (\boldsymbol{\alpha}, \boldsymbol{\alpha}^{\text{alt.}}))$
    $\mathbf{D}^{\text{recycle}} \leftarrow \text{Distogram}(\tilde{\mathbf{X}})$
    $\mathbf{V}^{\text{recycle}} \leftarrow \mathbf{V}$
    $\mathbf{E}^{\text{recycle}} \leftarrow \mathbf{E}$
  end for ▷ Use loss and structure from the last recycling iteration
  $\mathbf{D} \leftarrow \text{DistogramHead}(\mathbf{E})$
  $\mathcal{L}_{\text{Distogram}} \leftarrow \text{computeDistogramLoss}(\mathbf{D}, \mathbf{X}_{\text{all-atom}})$
return $\tilde{\mathbf{X}}$, $\mathcal{L}_{\text{FAPE}}$, $\mathcal{L}_{\text{C}_\alpha\text{-FAPE}}$, $\mathcal{L}_{\text{Torsion}}$, $\mathcal{L}_{\text{Distogram}}$

Algorithm 6 FoldingBlock

def FoldingBlock$\big(\mathbf{V} \in \mathbb{R}^{N \times c_V}$, $\mathbf{E} \in \mathbb{R}^{N \times N \times c_E}\big)$
  $\mathbf{B} \leftarrow \text{Linear}(\mathbf{E})$
  $\mathbf{V} \leftarrow \mathbf{V} + \text{MultiHeadSelfAttention}\big(\mathbf{V}, \text{ bias} = \mathbf{B}\big)$
  $\mathbf{V} \leftarrow \mathbf{V} + \text{MLP}(\mathbf{V})$
  $\mathbf{E} \leftarrow \mathbf{E} + \text{Linear}\big(\text{Concat}\big[\text{OuterProduct}(\mathbf{V}), \text{OuterDifference}(\mathbf{V})\big]\big)$
  $\mathbf{E} \leftarrow \mathbf{E} + \text{TriangularMultiplicativeUpdateOutgoing}(\mathbf{E})$
  $\mathbf{E} \leftarrow \mathbf{E} + \text{TriangularMultiplicativeUpdateIncoming}(\mathbf{E})$
  $\mathbf{E} \leftarrow \mathbf{E} + \text{MLP}(\mathbf{E})$
return $\mathbf{V}$, $\mathbf{E}$

Algorithm 7 StructureModule

def StructureModule$\big(\mathbf{V} \in \mathbb{R}^{N \times c_V},\ \mathbf{E} \in \mathbb{R}^{N \times N \times c_E},\ (\mathbf{X}_{\text{all-atom}},\ \mathbf{X}^{\text{alt.}}_{\text{all-atom}}) \in \mathbb{R}^{N \times 14 \times 3},$
$(\mathbf{T}_{\text{all-atom}},\ \mathbf{T}^{\text{alt.}}_{\text{all-atom}}) \in \mathrm{SE}(3)^{N \times 7},\ (\boldsymbol{\alpha},\ \boldsymbol{\alpha}^{\text{alt.}}) \in \mathbb{R}^{N \times 7 \times 2}\big)$
  $\mathbf{V} \leftarrow \mathrm{LayerNorm}(\mathbf{V})$
  $\mathbf{E} \leftarrow \mathrm{LayerNorm}(\mathbf{E})$
  $\mathbf{V}^{\text{initial}} \leftarrow \mathbf{V}$
  $\mathbf{V} \leftarrow \mathrm{Linear}(\mathbf{V})$
  $(\tilde{\mathbf{T}}, \dot{\mathbf{T}}) \leftarrow (\mathbf{I},\ \mathbf{0})^N \quad (\mathbf{I} \in \mathbb{R}^{3\times3},\ \mathbf{0} \in \mathbb{R}^3)^N$
  for $\ell = 1, \ldots, N_{\text{Layers}}$ **do** ▷ Shared weights
    $\mathbf{V} \leftarrow \mathbf{V} + \mathrm{InvariantPointAttention}(\mathbf{V},\ \mathbf{E},\ \tilde{\mathbf{T}})$
    $\mathbf{V} \leftarrow \mathrm{LayerNorm}\big(\mathrm{Dropout}_{0.1}(\mathbf{V})\big)$
  # Transition
    $\mathbf{V} \leftarrow \mathbf{V} + \mathrm{Linear}\big(\mathrm{ReLU}(\mathrm{Linear}(\mathrm{ReLU}(\mathrm{Linear}(\mathbf{V}))))\big)$
    $\mathbf{V} \leftarrow \mathrm{LayerNorm}\big(\mathrm{Dropout}_{0.1}(\mathbf{V})\big)$
  # Update backbone
    $\tilde{\mathbf{T}} \leftarrow \tilde{\mathbf{T}} \circ \mathrm{BackboneUpdate}(\mathbf{V}), \quad \dot{\mathbf{T}} \leftarrow \dot{\mathbf{T}} \circ \mathrm{BackboneUpdate}(\mathbf{V})$
  # Predict torsion angles $\omega, \phi, \psi, \chi_1, \chi_2, \chi_3, \chi_4$
    $\mathbf{A} \leftarrow \mathrm{Linear}(\mathbf{V}) + \mathrm{Linear}(\mathbf{V}^{\text{initial}})$
    $\mathbf{A} \leftarrow \mathbf{A} + \mathrm{Linear}\big(\mathrm{ReLU}(\mathrm{Linear}(\mathrm{ReLU}(\mathbf{A})))\big)$
    $\mathbf{A} \leftarrow \mathbf{A} + \mathrm{Linear}\big(\mathrm{ReLU}(\mathrm{Linear}(\mathrm{ReLU}(\mathbf{A})))\big)$
    $\boldsymbol{\alpha} \leftarrow \mathrm{Linear}\big(\mathrm{ReLU}(\mathbf{A})\big)$
    $(\tilde{\boldsymbol{\alpha}}, \dot{\boldsymbol{\alpha}}) \leftarrow \boldsymbol{\alpha}$
  # Backbone and torsion losses at each iteration
    $(\dot{\mathbf{R}},\ \dot{\mathbf{t}}) \leftarrow \dot{\mathbf{T}}$
    $\dot{\mathbf{X}}_{\mathrm{C}_\alpha} \leftarrow \dot{\mathbf{t}}$
    $\mathcal{L}^{\ell}_{\mathrm{C}_\alpha\text{-FAPE}} \leftarrow \mathrm{computeFAPE}\big(\dot{\mathbf{T}},\ \dot{\mathbf{X}}_{\mathrm{C}\alpha},\ \mathbf{T},\ \mathbf{X}_{\mathrm{C}\alpha}\big)$
    $\mathcal{L}^{\ell}_{\text{Torsion}} \leftarrow \mathrm{computeTorsionAngleLoss}\big(\dot{\boldsymbol{\alpha}},\ \boldsymbol{\alpha},\ \boldsymbol{\alpha}^{\text{alt.}}\big)$
  # No rotation gradients between iterations to stabilize training.
    if $\ell < N_{\text{Layer}}$ **then**
      $\tilde{\mathbf{T}} \leftarrow \big(\mathrm{stopgrad}(\tilde{\mathbf{R}}),\ \tilde{\mathbf{t}}\big)$
      $\dot{\mathbf{T}} \leftarrow \big(\mathrm{stopgrad}(\dot{\mathbf{R}}),\ \dot{\mathbf{t}}\big)$
    end if
  end for
  $\mathcal{L}_{\mathrm{C}_\alpha\text{-FAPE}} \leftarrow \frac{1}{N_{\text{Layers}}} \sum_\ell \mathcal{L}^{\ell}_{\mathrm{C}_\alpha\text{-FAPE}}, \quad \mathcal{L}_{\text{Torsion}} \leftarrow \frac{1}{N_{\text{Layers}}} \sum_\ell \mathcal{L}^{\ell}_{\text{Torsion}}$
  $(\tilde{\mathbf{T}}_{\text{all-atom}},\ \tilde{\mathbf{X}}) \leftarrow \mathrm{computeAllAtomCoordinates}(\tilde{\mathbf{T}},\ \tilde{\boldsymbol{\alpha}})$
  $(\dot{\mathbf{T}}_{\text{all-atom}},\ \dot{\mathbf{X}}) \leftarrow \mathrm{computeAllAtomCoordinates}(\dot{\mathbf{T}},\ \dot{\boldsymbol{\alpha}})$
  # Rename symmetric atoms in ground truth
  $(\mathbf{X}_{\text{all-atom}},\ \mathbf{T}_{\text{all-atom}}) \leftarrow \mathrm{renameSymmetricGroundTruthAtoms}\big(\mathbf{X}_{\text{all-atom}},\ \mathbf{X}^{\text{alt.}}_{\text{all-atom}},$
  $\mathbf{T}_{\text{all-atom}},\ \mathbf{T}^{\text{alt.}}_{\text{all-atom}},\ \dot{\mathbf{X}},\ \dot{\mathbf{T}}\big)$
  # Final loss on all atom coordinates and all frames(renamed)
  $\mathcal{L}_{\text{FAPE}} \leftarrow \mathrm{computeFAPE}\big(\dot{\mathbf{T}}_{\text{all-atom}},\ \dot{\mathbf{X}}_{\text{all-atom}},\ \mathbf{T}_{\text{all-atom}},\ \mathbf{X}_{\text{all-atom}}\big)$
return $\tilde{\mathbf{X}},\ \mathcal{L}_{\text{FAPE}},\ \mathcal{L}_{\mathrm{C}_\alpha\text{-FAPE}},\ \mathcal{L}_{\text{Torsion}}$

A.1.2 Data

We curate our custom dataset E2EData from protein-only entries fetched from the Protein Data Bank on July 16, 2025 and determined by X-ray diffraction, electron microscopy, neutron diffraction, or electron crystallography and with a reconstruction or refinement resolution of 3 Å or better; further, we limit sequence lengths to lie between 50 and 412 resolved residues. Homooligomeric assemblies were omitted due to chain-permutation symmetry incompatibilities with our model architecture. We then clustered the first biological assembly of each remaining entry using foldseek-multimer with 0.5 coverage, a per chain TM-score threshold of 0.6 and the coverage-mode set to 0. Based on the clustering results, entries were partitioned into training (90%), validation (5%), and test (5%) sets.

A.1.3 Training/evaluation details and hyperparameters

As E2EFOLD's architecture is adapted from OpenFold/AlphaFold 2, ESMFold, and ProteinMPNN, we primarily adopt the implementations and reference hyperparameters of these models. Table 1 summarizes important details and hyperparameters.

Table 1: Important training/evaluation details and hyperparameters.

Category	**Details and hyperparameters**
Loss weights	$\mathcal{L}_{\text{total}} = 0.5\,(\mathcal{L}_{\text{intra-C}_\alpha\text{-FAPE}} + \mathcal{L}_{\text{inter-C}_\alpha\text{-FAPE}}) + 0.5\,\mathcal{L}_{\text{FAPE}} + 0.5\,\mathcal{L}_{\text{Distogram}} + 0.5\,\mathcal{L}_{\text{Torsion}} + 0.05\,\mathcal{L}_{\text{Sequence Recovery}}$
FAPE clamps	intra-chain-C_α: 10.0 Å inter-chain-C_α: 30.0 Å all-atom: 10.0 Å clamped weight: 0.99 unclamped weight: 0.01
Optimizer	AdamW: $\beta = (0.9, 0.98)$, AMSGrad, gradient-norm clip: 0.1
LR schedule	Piecewise schedule: linear warmup from 1e-6 to 5e-4 over 5e5 steps constant at 5e-4 until step 3e6 exponential decay, every 1e5 steps multiply by 0.95
Featurizer	128 positional encodings 32 radial basis functions $k = 48$ nearest neighbors Gaussian noise (variance scale 0.4 Å at training, 0.0 Å at evaluation)
Encoder	$N_{\text{Layers}} = 3$ $c_V = 96$ transformer head width: $w_V = 96$
Quantizer	Gumbel-Softmax (straight-through) temperature annealed $\tau : 10.0 \to 1.0$ over 1000 epochs evaluation at $\tau = 1.0$, argmax instead of sampling
Decoder (ESMPairformer)	6 blocks $c_V = 256$, $c_E = 32$ transformer head widths: $w_V = 64$, $w_E = 32$ Dropout 0.125 $N_{recycle} = 5$
Decoder (StructureModule)	$c_V = 384$, $c_E = 128$, $c_{\text{IPA}} = 16$ IPA heads: $w_{\text{IPA}} = 12$, number of query-key and value points: $N_{qk} = 4$, $N_v = 8$ StructureModule layers: $N_{\text{Layers}} = 8$

A.2 Extended Results

A.2.1 Structure-based Protein Sequence Design

Table 2: Median sequence recovery (%) on the CATH 4.2 test set, as reported by Gao et al. [2022] and Shuai et al. [2025]. Bold indicates the best result.

Model	Recovery (%↑)
GraphTrans	35.82
StructGNN	35.91
GCA	37.64
ESM-IF1†	38.30
GVP-large†	39.20
GVP	39.47
AlphaDesign	41.31
ProteinMPNN*	45.96
ESM-IF1 (AF2DB)†	51.60
PiFold	**51.66**
FAMPNN (5-Step)	50.00
E2EFOLD	25.15

† Evaluated on CATH 4.3 test set [Hsu et al., 2022]. * Reproduced by Gao et al. [2022] with CATH 4.2.

A.2.2 Structure prediction

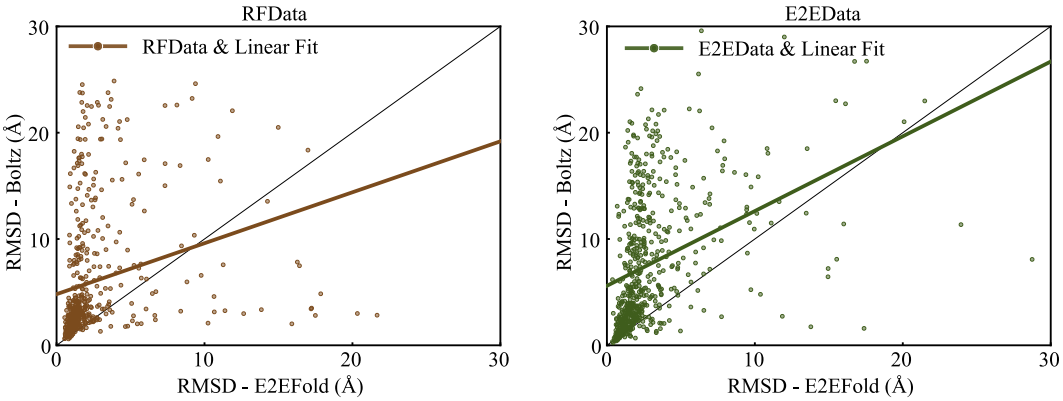

Figure 3: **Structure Prediction**. Prediction errors by E2EFOLD's decoder and Boltz-2 on RFData (left) and E2EData (right) correlate.