SE(3)-Equivariant Energy-based Models for End-to-End Protein Folding

Jiaxiang Wu
Tencent AI Lab
jonathanwu@tencent.com

Tao Shen
Tencent AI Lab
scotttshen@tencent.com

Haidong Lan
Tencent AI Lab
haidonglan@tencent.com

Yatao Bian
Tencent AI Lab
yataobian@tencent.com

Junzhou Huang
Tencent AI Lab
joehhuang@tencent.com

Abstract

Accurate prediction of protein structures is critical for understanding the biological function of proteins. Nevertheless, most structure optimization methods are built upon pre-defined statistical energy functions, which may be sub-optimal in formulating the conformation space. In this paper, we propose an end-to-end approach for protein structure optimization, powered by SE(3)-equivariant energy-based models. The conformation space is characterized by a SE(3)-equivariant graph neural network, with substantial modifications to embed the protein-specific domain knowledge. Furthermore, we introduce continuously-annealed Langevin dynamics as a novel sampling algorithm, and demonstrate that such process converges to native protein structures with theoretical guarantees. Extensive experiments indicate that SE(3)-Fold achieves comparable structure optimization accuracy, compared against state-of-the-art baselines, with over 1-2 orders of magnitude speed-up.

1 Introduction

During the past few decades, the problem of predicting protein structures from amino-acid sequences has attracted increasing attention. There has been a remarkable progress towards more accurate protein structure prediction, largely owing to more precise inter-residue contact/distance and orientation predictions by deep networks [1, 2, 46, 48, 29, 30, 17]. On the other hand, most structure prediction methods rely on traditional optimization toolkits, including Rosetta [33], I-TASSER [26], and CNS [5]. In CASP14[1], almost all the top-ranked teams adopted either Rosetta or I-TASSER for protein structure optimization, with the only exception of AlphaFold2 [18]. Such toolkits often involve complicated optimization techniques and multi-stage refinement, which imposes extra difficulties in building an end-to-end system for protein structure optimization. Additionally, most statistics energy functions [23, 24] used in these toolkits are built with human expertise, which may be sub-optimal in formulating the protein conformation space.

There have been a few attempts for fully-differentiable protein structure optimization. In [3], LSTM models are trained to estimate backbone torsion angles, which are then used to recover the geometry of protein backbones. NEMO [16] represents protein structures as internal coordinates and performs iterative refinement by a unrolled Langevin dynamics simulator, guided by a neural energy function. AlphaFold2 [18] employs an attention-based fully-differentiable framework for end-to-end learning

[1]The CASP14 (Critical Assessment of Structure Prediction) competition, hold on 2020, was recognized as one of the best benchmarks for determining state-of-the-art methods for protein structure prediction.

Preprint. Under review.

from homologous sequences to protein structures, and achieves atomic resolution for the first time. However, critical technical details in AlphaFold2 are still unrevealed at the moment.

On the other hand, several works adopt energy-based models (EBMs) and/or normalizing flow to characterize 3D molecular structures, where the SE(3)-equivariance is (approximately) preserved. The SE(3)-equivariance requirement is critical to guarantee consistent and predictable predictions *w.r.t.* arbitrary 3D rotations and translations. Xu *et al.* [47] combine flow-based and energy-based models to predict the inter-atom distance and then recover 3D structures for small molecules. Shi *et al.* [31] firstly estimate gradients over inter-atom distance, which are invariant to rotations and translations, and then propagate these gradients to 3D coordinates with SE(3)-equivariance preserved. [27] adopts E(n)-equivariant graph neural networks (EGNNs) [28] to model the first-order derivative in a continuous-time flow for molecule generation in 3D. In [8], random rotations are used as data augmentation to encourage (but not guarantee) the rotational invariance of the model for protein rotamer recovery, *e.g.* determining the best side-chain conformation with backbone structures given.

However, it remains unclear how to extend above approaches for efficient protein structure optimization. [47, 31, 27] only focus on small molecules consist of tens of atoms, while proteins are macro-molecules and often involve over thousands of atoms. NEMO [16] does not utilize the co-evolution information due to its sequence-based formulation, which is critical for determining protein structures. Additionally, its training process is computationally expensive (about 2 months) and could be sometimes unstable due to back-propagating through long simulations.

In this paper, we propose a SE(3)-equivariant energy-based model for fully-differentiable protein structure optimization, namely SE(3)-Fold. Protein structures are formulated as graphs with amino-acid residues as nodes and inter-residue interactions as edges. We adopt EGNN [28] to iteratively update node embeddings and estimated gradients over 3D coordinates in a SE(3)-equivariant manner. This model is trained to approximate the underlying distribution of protein structures. However, simply applying EGNN models for protein folding is insufficient in capturing various types of inter-residue interactions, and may encounter the stability issue during the sampling process. Hence, we introduce multi-head attentions to better formulate protein structures, and propose the continuously-annealed Langevin dynamics (CALD) algorithm for more stable sampling. Furthermore, we present the theoretical analysis on its convergence, and empirically show that CALD consistently stabilizes the protein structure optimization process.

We evaluate our SE(3)-Fold approach for protein structure optimization on the CATH database [32]. In general, SE(3)-Fold achieves comparable structure prediction accuracy (lDDT-Ca: 0.7928) with the state-of-the-art baseline, trRosetta [48]. By cooperating the multi-head attention mechanism and CALD sampling, we observe consistent improvement over the naive combination of EBM and EGNN models. SE(3)-Fold is GPU-friendly and computationally efficient for both training and sampling, which can speed up the structure optimization process by 1-2 orders of magnitude than trRosetta.

We summarize our major contributions in three folds:

- We propose a fully-differentiable approach for protein structure optimization, which paves the way to end-to-end learning from homologous sequences to protein structures.
- We scale up energy-based models for small molecules to proteins with thousands of atoms via residue-based graph representation, which is efficient in both training and sampling.
- To alleviate the stability issue in sampling-based structure optimization, a novel continuously-annealed Langevin dynamics sampling algorithm is proposed with theoretical guarantee.

2 Related Work

Protein structure optimization. Most methods predict protein structures by minimizing statistics energy functions [23, 24] and/or performing comparative modeling from structural templates [37]. In [2, 48, 30, 17], deep networks are introduced to predict inter-residue contacts, distance, and/or orientations, which are then transformed into additional restraints or differentiable energy terms for structure optimization. Still, traditional structure optimization toolkits, *e.g.* Rosetta [33], are heavily used in these methods for energy minimization. RGN [3] and NEMO [16] directly predict 3D structures with fully-differentiable networks; however, neither of them utilizes the co-evolution information, which leads to inferior quality of predicted structures. AlphaFold2 [18] is the first

end-to-end approach to directly build protein structures from co-evolution information embedded in homologous sequences, without the need for Rosetta-like optimization toolkits.

Energy-based models. These models formulate the probability distribution by parameterizing its unnormalized negative log probability (*i.e.* energy function) or its gradients (*i.e.* score function). Such formulation avoids the restriction on the tractability of normalization constant, allowing various models to be used to depict the probability distribution. Energy-based models have been applied in various domains, including image generation [43, 9, 34, 35, 36], 3D shape pattern generation [44, 45], and molecular conformation prediction [16, 8].

SE(3)-equivariant models. Due to rotational and translational symmetries in the 3-dimensional space, SE(3)-equivariance is critical for models directly operates on the structural data, *e.g.* 3D point cloud [38] and molecular structures [11, 15]. In [38, 11, 12], spherical harmonics and Clebsch–Gordan coefficients are used to construct SE(3)-equivariant convolutional kernels. Regular representations are used in [7, 10, 15] as an alternative approach for achieving the SE(3)-equivariance. [28] modifies the standard message passing process [13] with an E(n)-equivariant update rule for 3D coordinates.

3 Preliminaries

In this section, we briefly review basic concepts of energy-based models and E(n)-equivariant graph neural networks, which are fundamental to our proposed SE(3)-Fold approach.

3.1 Energy-based Model

Assuming the dataset $\{\mathbf{x}_i\}_{i=1}^N$ is uniformly sampled from some unknown distribution $p(\mathbf{x})$, energy-based models (EBMs) aim at approximating its energy function $E(\mathbf{x}) = \log p(\mathbf{x}) + C$ (where C is an arbitrary constant) or score function $s(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$ with neural networks. Particularly, [34, 35] propose to approximate the score function with denoising score matching (DSM). Each sample $\mathbf{x}$ is randomly perturbed with an additive Gaussian noise $\mathcal{N}\left(\mathbf{0}, \sigma_k^2 \mathbf{I}\right)$, and the perturbed sample is denoted as $\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}}|\mathbf{x}, \sigma_k) = \mathcal{N}\left(\mathbf{x}, \sigma_k^2 \mathbf{I}\right)$. In DSM, K random noise levels are used, *i.e.* $\sigma_1 > \sigma_2 \cdots > \sigma_K$, to simultaneously approximate the original distribution and mix among different data modes. The score network, which takes perturbed samples as inputs, is expected to estimate the perturbed data distribution's log probability's gradients over perturbed samples. Formally, the optimization objective is given by:

$$L(\boldsymbol{\theta}) = \frac{1}{2NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \sigma_k^2 \cdot \mathbb{E}_{\tilde{\mathbf{x}}_i \sim p(\tilde{\mathbf{x}}_i|\mathbf{x}_i, \sigma_k)} \left[\left\| s_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_i; \sigma_k) - \frac{\mathbf{x}_i - \tilde{\mathbf{x}}_i}{\sigma_k^2} \right\|_F^2 \right] \tag{1}$$

which is the weighted sum of estimation error across all the random noise levels. The above optimization objective aims at minimizing the difference between the data distribution $p(\mathbf{x})$ and its parameterized counterpart, defined by the score network $s_{\boldsymbol{\theta}}(\cdot)$. To sample novel samples from the score network's corresponding data distribution, one may adopt annealed Langevin dynamics:

$$\tilde{\mathbf{x}}^{(k,t+1)} := \tilde{\mathbf{x}}^{(k,t)} + \frac{\alpha_k}{2} s_{\boldsymbol{\theta}}\left(\tilde{\mathbf{x}}^{(k,t)}; \sigma_k\right) + \sqrt{\alpha_k} \mathbf{z}^{(k,t)}, \; k = 1, \ldots, K, t = 0, \ldots, T-1 \tag{2}$$

where α_k is the step size for the k-th random noise level, and $\mathbf{z}^{(k,t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The sampling process starts with $k = 1$, corresponding to the largest standard deviation in the random noise, and then gradually reduces it. Each level's sampling process is initialized as $\tilde{\mathbf{x}}^{(k,0)} = \tilde{\mathbf{x}}^{(k-1,T)}$, except for the first level, which is sampled from the prior distribution, *i.e.* $\tilde{\mathbf{x}}^{(1,0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

3.2 E(n)-Equivariant Graph Neural Network

Equivariance. Let function $f(\cdot)$ be a mapping between $\mathcal{X}$ and $\mathcal{Y}$, and $\mathbf{x} \in \mathcal{X}$. Consider the abstract group G and $g \in G$, we denote its corresponding transformations as $T_g : \mathcal{X} \to \mathcal{X}$ and $S_g : \mathcal{Y} \to \mathcal{Y}$. The function $f(\cdot)$ is considered to be equivariant to G, if for any g and transformation T_g, there exists at least one transformation S_g satisfying:

$$f(T_g(\mathbf{x})) = S_g(f(\mathbf{x})), \; \forall \mathbf{x} \in \mathcal{X} \tag{3}$$

We suggest readers to [11] for a more comprehensive explanation of equivariance.

E(n)-Equivariant Graph Neural Network (EGNN). The EGNN model [28] is designed to guarantee the E(n)-equivariance for point cloud data, where each point is associated with a feature vector $\mathbf{h}_i \in \mathbb{R}^D$ and a coordinate vector $\mathbf{x}_i \in \mathbb{R}^N$. The E(n)-equivariance is defined on the N-dimensional space, consists of translation, rotation (reflection), and permutation equivariance. The E(n)-equivariant graph convolutional layer (EGCL) modifies the message passing process to ensure the E(n)-equivariance:

$$\begin{aligned} \mathbf{m}_{ij} &= \phi_m\left(\mathbf{h}_i, \mathbf{h}_j, \mathbf{g}_{ij}, \|\mathbf{x}_i - \mathbf{x}_j\|_2\right) \\ \mathbf{x}_i' &= \mathbf{x}_i + \sum\nolimits_{j \in \mathcal{N}(i)} \phi_x\left(\mathbf{m}_{ij}\right) \cdot \left(\mathbf{x}_i - \mathbf{x}_j\right) \\ \mathbf{h}_i' &= \phi_h\Big(\mathbf{h}_i, \sum\nolimits_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}\Big) \end{aligned} \tag{4}$$

where $\mathcal{N}_i$ is the i-th node's neighborhood, $\mathbf{g}_{ij}$ is edge features for (i, j), and $\mathbf{m}_{ij}$ is the intermediate edge-wise message. Non-linear mappings (ϕ_m, ϕ_x, and ϕ_h) are instantiated as multi-layer perceptron models with Swish activations [25]. Since coordinate vectors are updated by the weighted sum of relative coordinates, and weighting coefficients are E(n)-invariant, the above message passing process is guaranteed to be E(n)-equivariant.

4 SE(3)-Fold

In this section, we firstly formulate the protein folding problem from an EBM perspective. Afterwards, we describe how to train the underlying energy-based model, and present a novel sampling algorithm for structure optimization with theoretical guarantee.

4.1 Protein Folding: An EBM Perspective

Given an amino-acid sequence of length L, the protein folding task aims at determining 3D coordinates of all the atoms for each amino-acid residue. As a simplified form, one may alternatively predict 3D coordinates for backbone atoms (N, C_α, C', and O) or C_α atoms only. This is almost sufficient for protein folding, as there are several off-the-shelf tools [14, 20, 40] for determining side-chain conformations from backbone or C_α-trace structures. Hence, in this paper, we only focus on solving C_α-trace or backbone structures, although the proposed method can be easily scaled up for formulating full-atom structures.

For each residue, we take U atoms to represent its spatial location, where $U = 1$ for C_α-trace and $U = 4$ for backbone structures[2]. We denote all the considered atoms' 3D coordinates as $\mathbf{x} \in \mathbb{R}^{3LU}$, which is the concatenation of each atom's 3D coordinate $\mathbf{x}_{i,u}$. The amino-acid sequence is denoted as A. From the probabilistic perspective, the protein folding task can be viewed as finding the most likely 3D coordinates configuration for the given amino-acid sequence, *i.e.* $\mathbf{x}^* = \arg\max p\left(\mathbf{x}|A\right)$. As the underlying distribution $p\left(\mathbf{x}|A\right)$ is unknown, we propose to approximate it with an energy-based model, parameterized by a score network $s_{\boldsymbol{\theta}}\left(\mathbf{x}, A\right)$. Formally, the score network should satisfy:

$$s_{\boldsymbol{\theta}}\left(\mathbf{x}, A\right) \approx \nabla_{\mathbf{x}} \log p\left(\mathbf{x}|A\right), \ \forall \mathbf{x}, A \tag{5}$$

which approximates the log probability's gradients over 3D coordinates. Once the score network is trained, one may start from randomly initialized 3D coordinates, and gradually refine them via Langevin dynamics. The resulting 3D coordinates are expected to be close to the native structure, if the score network's corresponding probability distribution is similar with the native one.

4.2 SE(3)-Fold: Training

The protein structure can be represented as a graph $G = (\mathcal{V}, \mathcal{E})$, where vertexes $\mathcal{V}$ are amino-acid residues, and edges $\mathcal{E}$ correspond to inter-residue interactions. This allows us to naturally capture long-range dependencies embedded in protein structures from the graph perspective, which is somewhat difficult for sequence-based formulations. Additionally, we associate each node with per-residue features $\mathbf{h}_i$ (*e.g.* amino-acid type and positional encoding), and each edge with pairwise features $\mathbf{g}_{ij}$ (*e.g.* inter-residue distance and orientation predictions) to augment the graph representation. The estimation of log probability's gradients over 3D coordinates is thus converted into a node regression

[2]To formulate full-atom structures, $U = 15$ is sufficient to represent all the non-hydrogen atoms in each amino-acid. Zero-padding is required for amino-acids with fewer non-hydrogen atoms.

task, where the SE(3)-equivariance must be preserved. In other words, if the input 3D coordinates are rotated and/or translated by some transformation, then the score network's outputs must follow the same transformation.

For the score network, we adopt a modified E(n)-equivariant graph neural network as the backbone architecture. Multi-head attention is introduced to capture various types of inter-residue interactions. As each node is now associated with multiple atoms' 3D coordinates, we update each atom's 3D coordinates with independent sub-networks to preserve the SE(3)-equivariance. During each message passing process, edge-wise messages are firstly computed to determine attention coefficients:

$$\mathbf{m}_{ij}^{l} = \phi_{m}^{l}\left(\mathbf{h}_{i}, \mathbf{h}_{j}, \mathbf{g}_{ij}, \mathbf{r}_{ij}\right) \tag{6}$$

where $\mathbf{r}_{ij}$ encodes the distance between C_{α} atoms in the i-th and j-th residues. The index of attention head is denoted as $l = 1, \ldots, L$. Afterwards, query and key embeddings are given by:

$$\mathbf{q}_{i}^{l} = \phi_{q}^{l}\left(\mathbf{h}_{i}\right), \ \mathbf{k}_{ij}^{l} = \phi_{k}^{l}\left(\mathbf{h}_{j}, \mathbf{m}_{ij}^{l}\right) \tag{7}$$

and then passed through a softmax function to compute multi-head attention coefficients:

$$e_{ij}^{l} = \phi_{e}^{l}\left(\mathbf{q}_{i}^{l}, \mathbf{k}_{ij}^{l}\right), \ a_{ij}^{l} = \frac{e_{ij}^{l}}{\sum_{j' \in \mathcal{N}(i)} e_{ij'}^{l}} \tag{8}$$

where $\mathcal{N}_i$ denotes the i-th node's neighborhood, and a_{ij}^{l} is the attention coefficient for the l-th head. Finally, node features and estimated gradients are updated via short-cut connections when possible:

$$\begin{aligned} \mathbf{h}_{i}^{\prime} &= \begin{cases} \phi_{h}\left(\mathbf{h}_{i}, \sum_{l=1}^{L} \sum_{j \in \mathcal{N}(i)} a_{ij}^{l} \mathbf{m}_{ij}^{l}\right) & \text{if } \left|\mathbf{h}_{i}^{\prime}\right| \neq \left|\mathbf{h}_{i}\right| \\ \mathbf{h}_{i} + \phi_{h}\left(\sum_{l=1}^{L} \sum_{j \in \mathcal{N}(i)} a_{ij}^{l} \mathbf{m}_{ij}^{l}\right) & \text{otherwise} \end{cases} \\ \mathbf{s}_{i,u}^{\prime} &= \mathbf{s}_{i,u} + \sum_{l=1}^{L} \sum_{j \in \mathcal{N}(i)} a_{ij}^{l} \phi_{s,u}^{l}\left(\mathbf{m}_{ij}^{l}\right)\left(\mathbf{x}_{i,u} - \mathbf{x}_{j,u}\right) \end{aligned} \tag{9}$$

where $\mathbf{s}_{i,u}$ are initialized as all-zeros, and then updated in a residual manner. This is slightly different from [28], which updates 3D coordinates within each EGCL unit, while we keep input 3D structures constant throughout the score network and only update estimated gradients. The detailed implementation of all the above sub-networks can be found in Appendix B.

So far, we have presented the complete workflow of multi-head attention based E(n)-equivariant graph convolutional layer, referred to as MhaEGCL. Our score network consists of multiple stacked MhaEGCL units to increase the model capacity. Similar with denoising score matching [39, 34], the score network is trained to estimate ground-truth gradients over 3D coordinates for the randomly perturbed data distribution $p(\tilde{\mathbf{x}}|\mathbf{x}, \sigma_k)$. The score network's estimated gradients are denoted as $s_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, A) = \left[\mathbf{s}_{1,1}^{\prime}, \ldots, \mathbf{s}_{L,U}^{\prime}\right]$ from the last MhaEGCL unit. The loss function is given by:

$$L\left(\boldsymbol{\theta}\right) = \frac{1}{2NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \sigma_{k}^{2} \cdot \mathbb{E}_{\tilde{\mathbf{x}}^{(n)} \sim p\left(\tilde{\mathbf{x}}^{(n)} | \mathbf{x}^{(n)}, \sigma_k\right)} \left[\left\| \frac{s_{\boldsymbol{\theta}}\left(\tilde{\mathbf{x}}^{(n)}, A^{(n)}\right)}{\sigma_k} - \frac{\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}}{\sigma_k^2} \right\|_{F}^{2} \right] \tag{10}$$

where $A^{(n)}$ is the n-th training sample's amino-acid sequence. The score network's outputs are down-scaled by $1/\sigma_k$ to match the magnitude of ground-truth gradients at each random noise level, as suggested in [35]. The multiplier σ_k^2 is introduced to approximately keep each random noise level's equal contribution to the loss function. During each training iteration, native protein structures are perturbed with Gaussian noise at various levels, and then fed into the score network to estimate corresponding gradients over perturbed 3D coordinates.

4.3 SE(3)-Fold: Sampling

The score network defines a probability distribution of protein structures $p_{\boldsymbol{\theta}}\left(\mathbf{x}|A\right)$ that approximates the native one $p\left(\mathbf{x}|A\right)$, if the objective function in Eq. (10) is minimized. Therefore, to predict the protein structure from its amino-acid sequence, we start with randomly initialized 3D coordinates, and then apply annealed Langevin dynamics (ALD) [34] for iterative refinement:

$$\hat{\mathbf{x}}^{(k,t+1)} := \hat{\mathbf{x}}^{(k,t)} + \frac{\lambda_{g,k}}{2} \frac{s_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}^{(k,t)}, A\right)}{\sigma_k} + \lambda_{n,k} \mathbf{z}^{(k,t)}, \ k = 1, \ldots, K, t = 0, \ldots, T-1 \tag{11}$$

where $\hat{\mathbf{x}}^{(k,t)}$ denotes refined 3D coordinates at the k-th random noise level's t-th iteration. The multiplier for Gaussian noise is denoted as $\lambda_{n,k} = \alpha_n \sigma_k / \sigma_K$, which is proportional to the k-th random noise level's standard deviation. The multiplier for estimated gradients is denoted as $\lambda_{g,k} = \alpha_g \lambda_{n,k}^2$, where α_g is newly introduced to control the signal-to-noise ratio.

However, in ALD, the transition between different random noise levels is discontinuous, which may be sub-optimal to the convergence behavior. Therefore, we propose a continuously-annealed Langevin dynamics (CALD) sampling process to smoothly transit between different random noise levels. This is made possible since our score network is trained without taking random noise's standard deviations as inputs, unlike [34]. We exponentially reduce the random noise's standard deviation from the initial value σ_1 to its minimum σ_K, instead of taking K discrete values:

$$\sigma'_t = \sigma_1 \left(\sigma_K / \sigma_1\right)^{t/(T-1)}, \; t = 0, \ldots, T-1 \tag{12}$$

and the update rule for refined 3D coordinates is given by:

$$\hat{\mathbf{x}}^{(t+1)} := \hat{\mathbf{x}}^{(t)} + \frac{\lambda_{g,t}}{2} \frac{s_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}^{(t)}, A\right)}{\sigma'_t} + \lambda_{n,t} \mathbf{z}^{(t)}, \; t = 0, \ldots, T-1 \tag{13}$$

where $\lambda_{n,t} = \alpha_n \sigma'_t / \sigma_K$ and $\lambda_{g,t} = \alpha_g \lambda_{n,t}^2$. The overall structure optimization process for ALD and CALD is as summarized in Algorithm 1 and 2, respectively. As demonstrated in later experiments, the CALD sampling algorithm leads to more efficient structure optimization than ALD.

Algorithm 1: ALD Sampling [34]

Randomly initialize $\hat{\mathbf{X}}^{(0,T)}$
for $k = 1$ **to** K **do**
  Set $\hat{\mathbf{X}}^{(k,0)} = \hat{\mathbf{X}}^{(k-1,T)}$
  Determine σ_k, $\lambda_{n,k}$, and $\lambda_{g,k}$
  for $t = 0$ **to** $T-1$ **do**
    Update $\hat{\mathbf{X}}^{(k,t+1)}$ by Eq. (11)
  end
end
Return $\hat{\mathbf{X}}^{(K,T)}$

Algorithm 2: CALD Sampling (Ours)

Randomly initialize $\hat{\mathbf{X}}^{(0)}$
for $t = 0$ **to** $T-1$ **do**
  Determine σ'_t, $\lambda_{n,t}$, and $\lambda_{g,t}$
  Update $\hat{\mathbf{X}}^{(t+1)}$ by Eq. (13)
end
Return $\hat{\mathbf{X}}^{(T)}$

4.4 Theoretical Analysis

In this section, we present the theoretical analysis for our SE(3)-Fold's sampling process. To start with, we show that under mild assumptions, a restrained CALD process is guaranteed to generate samples close to the native protein structure for any amino-acid sequence.

Theorem 1. *Let A be an amino-acid sequence of length L, and $\mathbf{x}_0, \mathbf{x}_1, \ldots$ be a sequence of gradually refined structures, generated by iteratively applying the following update rule:*

$$\mathbf{x}_{t+1} := \mathbf{x}_t + \frac{\lambda_t^2}{2\sigma'_t} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \lambda_t \mathbf{z}_t, \; t = 0, \ldots, T-1 \tag{14}$$

where $\sigma'_t = \sigma_1 \left(\sigma_K / \sigma_1\right)^{t/(T-1)}$, $\lambda_t = \alpha_n \sigma'_t / \sigma_K$, and $\mathbf{z}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$ is the random noise. With a proper choice of α_n, this sequence converges to a probability distribution as $T \to \infty$, whose density function is given by:

$$p\left(\mathbf{x}\right) = \begin{cases} \frac{1}{Z} \exp\left(-\frac{\|\mathbf{x}^* - \mathbf{x}\|_2^2}{2\sigma_K^2}\right) & \text{if } \|\mathbf{x}^* - \mathbf{x}\|_2 \leq \sqrt{3LU}\sigma_K \\ \frac{1}{Z} \exp\left(-\frac{\sqrt{3LU}}{\sigma_K} \|\mathbf{x}^* - \mathbf{x}\|_2\right) & \text{otherwise} \end{cases} \tag{15}$$

where Z is the partition constant and $\mathbf{x}^*$ denotes the native protein structure corresponds to the amino-acid sequence A.

The detailed proof can be found in Appendix A. Theorem 1 indicates that by limiting $\alpha_g = 1$, the continuously-annealed Langevin dynamics (CALD) sampling process gradually refines 3D coordinates towards a neighborhood of the native protein structure. Furthermore, if the $\alpha_g = 1$ constraint is discarded, as in the standard CALD sampling process, we show that generated samples will follow a slightly modified probability distribution:

Corollary 1. *If the sequence* $\mathbf{x}_0, \mathbf{x}_1, \ldots$ *is iteratively generated by:*

$$\mathbf{x}_{t+1} := \mathbf{x}_t + \frac{\alpha_g \lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}(\mathbf{x}_t, A) + \lambda_t \mathbf{z}_t, \; t = 0, \ldots, T-1 \tag{16}$$

then this sequence converges to a probability distribution, whose density function is given by:

$$p(\mathbf{x}) = \begin{cases} \frac{1}{Z'} \exp\left(-\frac{\alpha_g \|\mathbf{x}^* - \mathbf{x}\|_2^2}{2\sigma_K^2}\right) & \text{if } \|\mathbf{x}^* - \mathbf{x}\|_2 \leq \sqrt{3LU}\sigma_K \\ \frac{1}{Z'} \exp\left(-\frac{\sqrt{3LU}\alpha_g}{\sigma_K} \|\mathbf{x}^* - \mathbf{x}\|_2\right) & \text{otherwise} \end{cases} \tag{17}$$

where Z' *is the partition constant, as* $T \to \infty$.

The above probability distribution shares a similar landscape with the original one given in Theorem 1, but the variance can be either increased or decreased, depending on the choice of α_g. In other words, this allows us to flexibly adjust the diversity of SE(3)-Fold's structure optimization results, even though the underlying score network remains the same.

5 Experiments

In this section, we firstly report the protein structure optimization accuracy of our proposed SE(3)-Fold method. Afterwards, we demonstrate that our method is computationally efficient and conduct detailed visualization on SE(3)-Fold's structure optimization process.

5.1 Setup

Dataset. We randomly select 2065 domains from the CATH database [32] (daily snapshot: 20201021), and split them into training, validation, and test subsets of size 1665/200/200. Particularly, to validate the generalization ability of structure optimization methods, it is guaranteed that each subset has a disjoint set of CATH super-families, *i.e.* no super-family overlap between subsets.

Features. For node features, we associate each residue with its amino-acid type's one-hot encoding and positional encoding to represent its relative position in the amino-acid sequence. For edge features, we employ an inter-residue distance and orientation predictor [42] to infer interactions for all the residue pairs from multiple sequence alignments (MSAs). These predictions are also fed into the baseline structure optimization method for a fair comparison. Please refer to Appendix C for more details on the extraction of node and edge features.

Graph Construction. Given a protein, we construct its graph representation with three types of neighbors for each residue: 1) sequence-based neighbors, 2) coordinate-based neighbors, and 3) random neighbors. Sequence-based neighbors connects two residues if their sequence separation is exactly $2^m (m = 0, 1, \ldots)$. Coordinate-based neighbors connects two residues if their C_α-C_α distance is within top-k minimal ones. Random neighbors introduce randomized connections between residues, which is critical for the sampling process since protein structures are randomly initialized and coordinate-based neighbors at early stages are noisy and less informative.

Implementation Details. To verify the effectiveness of multi-head attention, we train two series of score networks for SE(3)-Fold, with either 4 EGCL or MhaEGCL modules. Layer normalization [4] are inserted between EGCL/MhaEGCL modules to normalize intermediate node embeddings. Each model is trained with the Adam optimizer [19] for 50 epochs with batch size 32. The random noise scheme is set as $\sigma_1 = 10.0$, $\sigma_K = 0.01$, and $K = 61$. We maintain the exponential moving average (EMA) of score network's model parameters with the momentum parameter m set to 0.999, as suggested in [35], and use this EMA-based model for sampling. Hyper-parameters for the sampling process are tuned on the validation subset: $\alpha_n = 0.005$, $\alpha_g = 10$, $T = 16$ for ALD, and $T = 500$ for CALD. To alleviate randomness in reported results, we train multiple models with various hyper-parameter combinations, and report the averaged performance of top-10 models (in terms of validation loss) for the subsequent sampling process. The detailed hyper-parameter setting and per-model structure optimization results can be found in Appendix D. The training and sampling of each model is conducted in a single node, with Intel Xeon 8255C CPU and Nvidia Tesla V100 GPU.

5.2 Structure Optimization Accuracy

In Table 1, we report averaged and maximal lDDT-Ca scores [22] of several variants of our SE(3)-Fold approach on the test subset. For each model, we repeat the sampling process for 4 times (batch size: 32) to generate 128 decoys per domain. As a comparison, we run trRosetta [48] with the same inter-residue distance and orientation predictions, and generate 300 decoys per domain for evaluation, following its default hyper-parameter setting.

Table 1: Comparison on averaged and maximal lDDT-Ca scores of trRosetta and SE(3)-Fold. For trRosetta and SE(3)-Fold-Backbone, lDDT scores for backbone atoms are reported within brackets.

Method	Format	Layer	Sampling	Avg. lDDT-Ca	Max. lDDT-Ca
trRosetta [48]	-	-	-	0.7456 (0.7365)	**0.8525 (0.8465)**
EBM-Fold [41]	C_α-trace	-	-	0.7775	0.8124
SE(3)-Fold	C_α-trace	EGCL	ALD	0.7545±0.0089	0.7818±0.0057
			CALD	0.7654±0.0112	0.7965±0.0054
		MhaEGCL	ALD	0.7796±0.0065	0.8028±0.0052
			CALD	0.7902±0.0048	0.8134±0.0037
	Backbone	MhaEGCL	CALD	**0.7928±0.0036** (**0.7611±0.0107**)	0.8148±0.0033 (0.7806±0.0115)

From Table 1, we discover that both multi-head attention-based EGCL (MhaEGCL) module and continuously-annealed Langevin dynamics (CALD) consistently improve the structure optimization accuracy over their counterparts. In general, SE(3)-Fold achieves higher averaged lDDT-Ca scores than trRosetta, but is inferior to it in terms of maximal lDDT-Ca scores. This is mainly due to the notable difference in the decoy quality distribution between trRosetta and SE(3)-Fold. In Figure 1, we visualize the lDDT-Ca distribution for a few test domains, using the best MhaEGCL-based model (in terms of the validation loss) and CALD sampling. It is quite obvious that SE(3)-Fold's lDDT-Ca distribution is more concentrated than that of trRosetta, indicating that SE(3)-Fold tends to generate more consistent structure predictions. However, the best decoy's quality of SE(3)-Fold is indeed inferior to trRosetta, which needs further investigation and improvement.

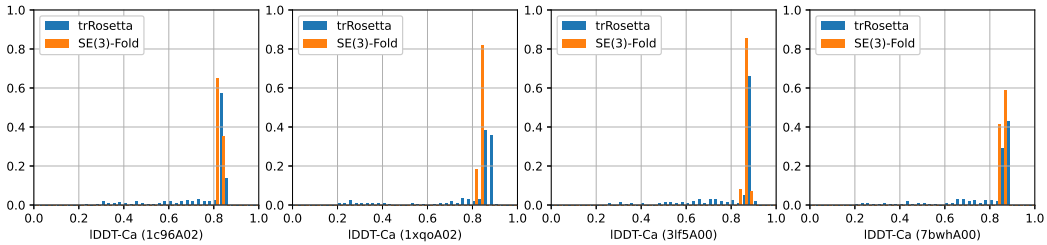


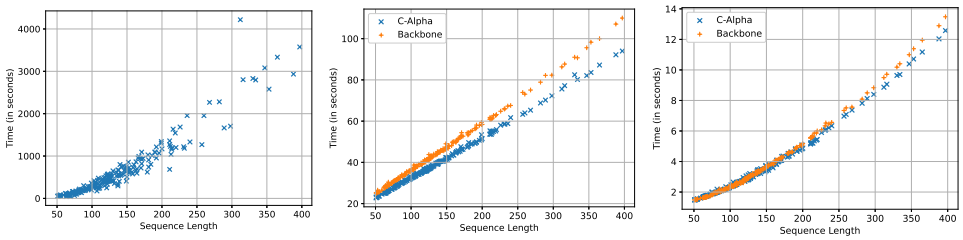
Figure 1: Comparison on lDDT-Ca distributions of trRosetta and SE(3)-Fold (residue format: C_α-trace). lDDT-Ca scores are discretized into 40 bins, and the domain ID is noted in the bracket.

5.3 Computational Efficiency

Most of state-of-the-art protein structure optimization algorithms [48, 30, 17] are developed on top of Rosetta software suite [33], which heavily relies on CPU-based computation while general GPU support is not yet presented. In contrast, SE(3)-Fold's computation overhead is dominated by the score network's forward pass, which can be efficiently accelerated by modern deep learning toolkits with full GPU support.

In Figure 2, we compare trRosetta and SE(3)-Fold's structure optimization time for each test domain. For CPU-based optimization, we limit the number of threads to 1 for a fair comparison. For GPU-based implementation, we set the batch size to 32 to improve the GPU utilization, and report the per-decoy optimization time (total time divided by the batch size). Here, we report the time consumption of the network architecture with highest computational overhead among all the candidates.

As depicted in Figure 2a, trRosetta's structure optimization time grows significantly for proteins with longer amino-acid sequences. For 1ojjA00 (length: 397 residues), it takes approximately 1 hour to



(a) trRosetta (CPU) (b) SE(3)-Fold (CPU) (c) SE(3)-Fold (GPU)

Figure 2: Comparison on the structure optimization time (per decoy) of trRosetta and SE(3)-Fold. For trRosetta, we report the averaged time consumption of three independent runs.

generate one decoy with trRosetta. On the other hand, SE(3)-Fold's structure optimization time grows much slower (approximately linear) *w.r.t.* the sequence length. Under the same hardware specification, CPU-based SE(3)-Fold achieves up to one magnitude speed-up over trRosetta, especially for long proteins. With GPU acceleration enabled, it only takes $\sim$13 seconds for SE(3)-Fold to generate one decoy for the longest test domain, which is over 100 times faster than trRosetta. It is also worth mentioning that backbone-based residue format does not leads to much additional time consumption, compared against C_α-atom based representation, especially for GPU-based implementation. Detailed analysis on the computational complexity of SE(3)-Fold can be found in Appendix E.

5.4 Visualization on the Sampling Process

Finally, we delve into SE(3)-Fold's structure optimization process to understand how protein structures are gradually refined from random initialization. In Figure 3, we visualize the lDDT-Ca score *vs.* number of iterations in ALD and CALD sampling process for a few test domains. Both ALD and CALD converge with a similar speed, but CALD avoids zig-zag patterns in ALD, due to discontinuous transition between different random noise levels. Please note that we reduce $T = 16$ to $T = 8$ in ALD to approximately match the total number of iterations with CALD, although $T = 16$ also leads to similar results in ALD.

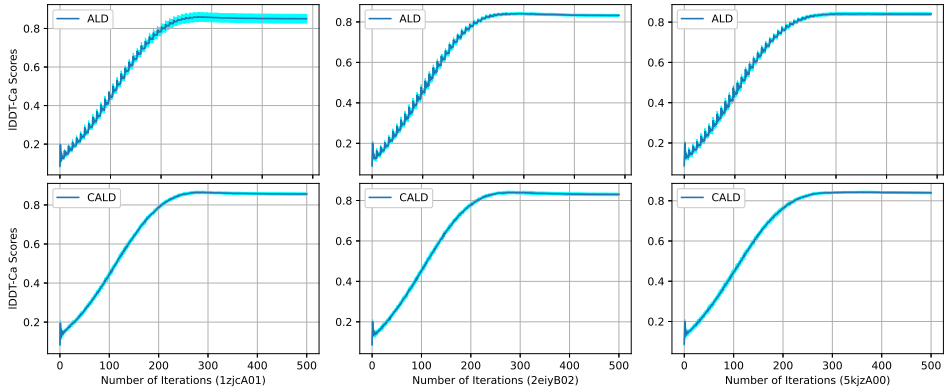


Figure 3: Comparison on lDDT-Ca scores during ALD (top) and CALD (bottom) sampling process. Each column corresponds to the same test domain, with the domain ID noted in the bracket.

6 Discussion

In this paper, we propose SE(3)-Fold as a fully-differentiable approach for protein folding. SE(3)-Fold achieves comparable structure optimization accuracy with state-of-the-art Rosetta-based protocols, and is efficient in both training (within one GPU day) and sampling (1-2 orders of magnitude faster than trRosetta). The co-evolution information from homologous sequences are well utilized, which leads to more accurate predictions than other end-to-end approaches, *e.g.* RGN and NEMO.

Limitations: SE(3)-Fold still relies on a pre-trained network to predict inter-residue distance and orientations from homologous sequences. Such network may be sub-optimal as its optimization does not fully utilize the information from 3D structures. Joint learning of these two modules may further boost the structure optimization accuracy.

References

[1] Badri Adhikari, Debswapna Bhattacharya, Renzhi Cao, and Jianlin Cheng. Confold: Residue-residue contact-guided ab initio protein folding. *Proteins: Structure, Function, and Bioinformatics*, 83(8):1436–1449, 2015.

[2] Badri Adhikari and Jianlin Cheng. Confold2: Improved contact-driven ab initio protein structure modeling. *BMC bioinformatics*, 19(1):1–5, 2018.

[3] Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell Systems*, 8(4):292–301.e3, Apr 2019.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv Preprint*, 1607.06450, 2016.

[5] Axel T Brunger. Version 1.2 of the crystallography and nmr system. *Nature protocols*, 2(11):2728, 2007.

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

[7] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999, 2016.

[8] Yilun Du, Joshua Meier, Jerry Ma, Rob Fergus, and Alexander Rives. Energy-based models for atomic-resolution protein conformations. In *International Conference on Learning Representations*, 2020.

[9] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, 2019.

[10] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176, 2020.

[11] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems*, 2020.

[12] Fabian B. Fuchs, Edward Wagstaff, Justas Dauparas, and Ingmar Posner. Iterative se(3)-transformers. *arXiv Preprint*, 2102.13419, 2021.

[13] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

[14] Dominik Gront, Sebastian Kmiecik, and Andrzej Kolinski. Backbone building from quadrilaterals: A fast and accurate algorithm for protein backbone reconstruction from alpha carbon coordinates. *Journal of Computational Chemistry*, 28(9):1593–1597, 2007.

[15] Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. *arXiv Preprint*, 2012.10885, 2020.

[16] John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. In *International Conference on Learning Representations*, 2019.

[17] Fusong Ju, Jianwei Zhu, Bin Shao, Lupeng Kong, Tie-Yan Liu, Wei-Mou Zheng, and Dongbo Bu. Copulanet: Learning residue co-evolution directly from multiple sequence alignment for protein structure prediction. *Nature Communications*, 12(1):2535, May 2021.

[18] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Žídek, Alex Bridgland, Clemens Meyer,

Simon A A Kohl, Anna Potapenko, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Martin Steinegger, Michalina Pacholska, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. High accuracy protein structure prediction using deep learning. In *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, 2020.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[20] Georgii G. Krivov, Maxim V. Shapovalov, and Roland L. Dunbrack Jr. Improved prediction of protein side-chain conformations with scwrl4. *Proteins: Structure, Function, and Bioinformatics*, 77(4):778–795, 2009.

[21] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

[22] Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: A local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 08 2013.

[23] Matthew J. O'Meara, Andrew Leaver-Fay, Michael D. Tyka, Amelie Stein, Kevin Houlihan, Frank DiMaio, Philip Bradley, Tanja Kortemme, David Baker, Jack Snoeyink, and Brian Kuhlman. Combined covalent-electrostatic model of hydrogen bonding improves structure prediction with rosetta. *Journal of Chemical Theory and Computation*, 11(2):609–622, 2015.

[24] Hahnbeom Park, Philip Bradley, Per Greisen, Yuan Liu, Vikram Khipple Mulligan, David E. Kim, David Baker, and Frank DiMaio. Simultaneous optimization of biomolecular energy functions on features from small molecules and macromolecules. *Journal of Chemical Theory and Computation*, 12(12):6201–6212, 2016.

[25] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv Preprint*, 1710.05941, 2017.

[26] Ambrish Roy, Alper Kucukural, and Yang Zhang. I-tasser: A unified platform for automated protein structure and function prediction. *Nature protocols*, 5(4):725–738, 2010.

[27] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B. Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows for molecule generation in 3d. *arXiv Preprint*, 2105.09016, 2021.

[28] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. *arXiv Preprint*, 2102.09844, 2021.

[29] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13). *Proteins: Structure, Function, and Bioinformatics*, 87(12):1141–1148, 2019.

[30] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[31] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, 2021.

[32] Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P Waman, Paul Ashford, Harry M Scholes, Camilla S M Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, Mahnaz Abbasian, Sean Le Cornu, Su Datt Lam, Karel Berka, Ivana Hutařová Varekova, Radka Svobodova, Jon Lees, and Christine A Orengo. Cath: Increased structural coverage of functional space. *Nucleic Acids Research*, 49(D1):D266–D273, 11 2020.

[33] Kim T. Simons, Rich Bonneau, Ingo Ruczinski, and David Baker. Ab initio protein structure prediction of casp iii targets using rosetta. *Proteins: Structure, Function, and Bioinformatics*, 37(S3):171–176, 1999.

[34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019.

[35] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, 2020.

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

[37] Yifan Song, Frank DiMaio, Ray Yu-Ruei Wang, David Kim, Chris Miles, TJ Brunette, James Thompson, and David Baker. High-resolution comparative modeling with rosettacm. *Structure*, 21(10):1735–1742, 2013.

[38] 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. *arXiv Preprint*, 2018.

[39] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[40] Benjamin Webb and Andrej Sali. Comparative protein structure modeling using modeller. *Current Protocols in Bioinformatics*, 54(1):5.6.1–5.6.37, 2016.

[41] Jiaxiang Wu, Shitong Luo, Tao Shen, Haidong Lan, Sheng Wang, and Junzhou Huang. Ebm-fold: Fully-differentiable protein folding powered by energy-based models. *arXiv Preprint*, 2105.04771, 2021.

[42] Jiaxiang Wu, Jianguo Pei, Haidong Lan, Tao Shen, Wei Liu, Sheng Wang, and Junzhou Huang. Multi-msa ensemble based distance prediction with hierarchical clustering and quality assessment upon probability distribution. In *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, 2020.

[43] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644. PMLR, 2016.

[44] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[45] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative voxelnet: Learning energy-based models for 3d shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[46] Jinbo Xu. Distance-based protein folding powered by deep learning. *Proceedings of the National Academy of Sciences*, 116(34):16856–16865, 2019.

[47] Minkai Xu, Shitong Luo, Yoshua Bengio, Jian Peng, and Jian Tang. Learning neural generative dynamics for molecular conformation generation. In *International Conference on Learning Representations*, 2021.

[48] Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences*, 117(3):1496–1503, 2020.

Appendix A: Proof for Theorem 1

In Theorem 1, we consider the following update rule:

$$\mathbf{x}_{t+1} := \mathbf{x}_t + \frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \lambda_t \mathbf{z}_t,\ t = 0, \ldots, T-1 \tag{18}$$

where $\sigma_t' = \sigma_1 \left(\sigma_K/\sigma_1\right)^{t/(T-1)}$ and $\lambda_t = \alpha_n \sigma_t'/\sigma_K$. The dimensionality of $\mathbf{x}_t$ is denoted as $n = 3LU$, which corresponds to an amino-acid sequence of length L and U atoms per amino-acid residue. We now show that under mild assumptions, the above iterative process converges to a neighborhood of the native protein structure for any specified amino-acid sequence.

To start with, we consider the following oracle score function, which is used to define the probability distribution we wish to converge to:

Definition 1. *For an amino-acid sequence A, its oracle score function is defined as:*

$$s^*\left(\mathbf{x}, \mathbf{x}^*\right) = \frac{\mathbf{x}^* - \mathbf{x}}{\max\left(\sigma_K, \frac{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2}{\sqrt{n}}\right)} \tag{19}$$

where $\mathbf{x}^*$ represents the native protein structure, and σ_K is the minimal standard deviation of random noise used during the training process.

Lemma 1. *The oracle score function $s^*\left(\cdot\right)$ defines a probability distribution, whose probability density function is given by:*

$$p\left(\mathbf{x}\right) = \begin{cases} \frac{1}{Z}\exp\left(-\frac{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2^2}{2\sigma_K}\right) & \text{if } \left\|\mathbf{x}^* - \mathbf{x}\right\|_2 \leq \sqrt{n}\sigma_K \\ \frac{1}{Z}\exp\left(-\sqrt{n}\left\|\mathbf{x}^* - \mathbf{x}\right\|_2\right) & \text{otherwise} \end{cases} \tag{20}$$

where Z is the partition constant to guarantee that $p\left(\mathbf{x}\right)$ is valid.

Proof. Due to the existence of max operator, we rewrite the oracle score function's definition as:

$$s^*\left(\mathbf{x}, \mathbf{x}^*\right) = \begin{cases} \frac{\mathbf{x}^* - \mathbf{x}}{\sigma_K} & \text{if } \left\|\mathbf{x}^* - \mathbf{x}\right\|_2 \leq \sqrt{n}\sigma_K \\ \frac{\sqrt{n}\left(\mathbf{x}^* - \mathbf{x}\right)}{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2} & \text{otherwise} \end{cases} \tag{21}$$

For the first case, we have:

$$\nabla_{\mathbf{x}} \log p\left(\mathbf{x}\right) = \nabla_{\mathbf{x}}\left(-\frac{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2^2}{2\sigma_K} - \log Z\right) = \frac{\mathbf{x}^* - \mathbf{x}}{\sigma_K} \tag{22}$$

and for the second case, we have:

$$\nabla_{\mathbf{x}} \log p\left(\mathbf{x}\right) = \nabla_{\mathbf{x}}\left(-\sqrt{n}\left\|\mathbf{x}^* - \mathbf{x}\right\|_2 - \log Z\right) = \frac{\sqrt{n}\left(\mathbf{x}^* - \mathbf{x}\right)}{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2} \tag{23}$$

which exactly matches the definition of oracle score function. Next, we show that these exists a partition constant Z satisfying that $p\left(\mathbf{x}\right)$ is a valid probability distribution. Particularly, we need to prove that the following two integral terms are finite:

$$\int_{\mathbf{x}\in\mathcal{X}} \exp\left(-\frac{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2^2}{2\sigma_K}\right) d\mathbf{x},\ \int_{\mathbf{x}\notin\mathcal{X}} \exp\left(-\sqrt{n}\left\|\mathbf{x}^* - \mathbf{x}\right\|_2\right) d\mathbf{x} \tag{24}$$

where $\mathcal{X} = \{\mathbf{x} | \left\|\mathbf{x}^* - \mathbf{x}\right\|_2 \leq \sqrt{n}\sigma_K\}$. Since the exponential term term only rely on the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}^*$, these integral terms can be transformed as:

$$\begin{aligned} \int_{\mathbf{x}\in\mathcal{X}} \exp\left(-\frac{\left\|\mathbf{x}^* - \mathbf{x}\right\|_2^2}{2\sigma_K}\right) d\mathbf{x} &= \int_0^{\sqrt{n}\sigma_K} \exp\left(-\frac{r^2}{2\sigma_K}\right) S_{n-1}\left(r\right) dr \\ \int_{\mathbf{x}\in\mathcal{X}} \exp\left(-\sqrt{n}\left\|\mathbf{x}^* - \mathbf{x}\right\|_2\right) d\mathbf{x} &= \int_{\sqrt{n}\sigma_K}^{\infty} \exp\left(-\sqrt{n}r\right) S_{n-1}\left(r\right) dr \end{aligned} \tag{25}$$

where $r = \|\mathbf{x}^* - \mathbf{x}\|_2$ and

$$S_{n-1}(r) = \frac{2\pi^{n/2}}{\Gamma(n/2)} r^{n-1} \tag{26}$$

is the surface area of the $(n-1)$-sphere of radius r embedded in the n-dimensional space, and $\Gamma(\cdot)$ is the gamma function. The first integral term is finite since the function within the integral symbol is finite. For the second integral term, we have:

$$\begin{aligned}
&\int_{\sqrt{n}\sigma_K}^{\infty} \exp\left(-\sqrt{n}r\right) S_{n-1}(r)\, dr \\
=\ & \frac{2\pi^{n/2}}{\Gamma(n/2)} \int_{\sqrt{n}\sigma_K}^{\infty} \exp\left(-\sqrt{n}r\right) r^{n-1} dr \\
=\ & \frac{2\pi^{n/2}}{\Gamma(n/2)} \cdot \frac{-1}{\sqrt{n}} \exp\left(-\sqrt{n}r\right) \left[r^{3L-1} + \frac{3L-1}{\sqrt{n}} r^{3L-2} + \cdots + \frac{(n-1)!}{(\sqrt{n})^{n-1}}\right] \Bigg|_{\sqrt{n}\sigma_K}^{\infty} \\
=\ & \frac{2\pi^{n/2}}{\Gamma(n/2)} \cdot \frac{\exp(-n\sigma_K)}{\sqrt{n}} \left[\left(\sqrt{n}\sigma_K\right)^{3L-1} + \frac{3L-1}{\sqrt{n}} \left(\sqrt{n}\sigma_K\right)^{3L-2} + \cdots + \frac{(n-1)!}{(\sqrt{n})^{n-1}}\right]
\end{aligned} \tag{27}$$

which is also finite. □

Lemma 2. *If the oracle score function is re-scaled by σ_K, i.e.:*

$$s^*_{\sigma_K}(\mathbf{x}, \mathbf{x}^*) = \frac{1}{\sigma_K} s^*(\mathbf{x}, \mathbf{x}^*),\ \forall \mathbf{x}, \mathbf{x}^* \tag{28}$$

then it corresponds to a probability distribution, whose probability density function is given by:

$$p(\mathbf{x}) = \begin{cases} \frac{1}{Z'} \exp\left(-\frac{\|\mathbf{x}^* - \mathbf{x}\|_2^2}{2\sigma_K^2}\right) & \text{if } \|\mathbf{x}^* - \mathbf{x}\|_2 \le \sqrt{n}\sigma_K \\ \frac{1}{Z'} \exp\left(-\frac{\sqrt{n}}{\sigma_K} \|\mathbf{x}^* - \mathbf{x}\|_2\right) & \text{otherwise} \end{cases} \tag{29}$$

where Z' is the partition constant.

It is worthy noting that for both the oracle score function and its re-scaled counterpart, the corresponding probability distribution's density function has a single global-maximal at $\mathbf{x}^*$.

During the training phase, we firstly select a random noise level σ_k from $\sigma_1, \ldots, \sigma_K$, and then perturb the native protein structure $\mathbf{x}^*$ with some random noise drawn from $\mathcal{N}(\mathbf{0}, \sigma_k^2 \mathbf{I})$. Therefore, the relative difference between native and perturbed structures is approximately given by:

$$\|\mathbf{x}^* - \tilde{\mathbf{x}}\|_2 \approx \sqrt{n}\sigma_k \tag{30}$$

where $\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}|\mathbf{x}^*, \sigma_k^2 \mathbf{I})$. With the above approximation, we can rewrite the original objective function in Eq. (10) as:

$$\begin{aligned}
L(\boldsymbol{\theta}) &= \frac{1}{2NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}}|\mathbf{x}_n^*, \sigma_k)} \left\| s_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, A_n) - \frac{\mathbf{x}_n^* - \tilde{\mathbf{x}}}{\sigma_k} \right\|_2^2 \\
&\approx \frac{1}{2NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}}|\mathbf{x}_n^*, \sigma_k)} \left\| s_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, A_n) - s^*(\tilde{\mathbf{x}}, \mathbf{x}_n^*) \right\|_2^2
\end{aligned} \tag{31}$$

which implies that the score network $s_{\boldsymbol{\theta}}(\cdot)$ is essentially approximating the oracle score function during the training process. Additionally, we make following assumptions:

Assumption 1. *The score network has bounded variance:*

$$\|s_{\boldsymbol{\theta}}(\mathbf{x}, A)\|_2 \le L_1,\ \forall \mathbf{x}, A \tag{32}$$

Assumption 2. *The score network is Lispchitz continuous:*

$$\|s_{\boldsymbol{\theta}}(\mathbf{x}, A) - s_{\boldsymbol{\theta}}(\mathbf{x}', A)\|_2 \le L_2 \|\mathbf{x} - \mathbf{x}'\|_2,\ \forall \mathbf{x}, \mathbf{x}', A \tag{33}$$

Assumption 3. *The score network has bounded estimation error for the oracle score function:*

$$\left\|\xi_{\boldsymbol{\theta}}\left(\mathbf{x}, \mathbf{x}^{*}, A\right)\right\|_{2} \leq L_{3}, \ \forall \mathbf{x}, \mathbf{x}^{*}, A \tag{34}$$

where $\xi_{\boldsymbol{\theta}}\left(\mathbf{x}, \mathbf{x}^{*}, A\right)=s_{\boldsymbol{\theta}}\left(\mathbf{x}, A\right)-s^{*}\left(\mathbf{x}, \mathbf{x}^{*}\right)$.

With above analyses and assumptions, we now present the convergence guarantee for our proposed continuously-annealed Langevin dynamics (CALD) sampling process:

Theorem 1. *Let* A *be an amino-acid sequence of length* L*, and* $\mathbf{x}_0, \mathbf{x}_1, \ldots$ *be a sequence of gradually refined structures, generated by iteratively applying the following update rule:*

$$\mathbf{x}_{t+1} := \mathbf{x}_t + \frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \lambda_t \mathbf{z}_t, \ t = 0, \ldots, T-1 \tag{14}$$

where $\sigma_t' = \sigma_1 \left(\sigma_K / \sigma_1\right)^{t/(T-1)}$*,* $\lambda_t = \alpha_n \sigma_t' / \sigma_K$*, and* $\mathbf{z}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$ *is the random noise. With a proper choice of* α_n*, this sequence converges to a probability distribution as* $T \to \infty$*, whose density function is given by:*

$$p\left(\mathbf{x}\right) = \begin{cases} \frac{1}{Z} \exp\left(-\frac{\left\|\mathbf{x}^{*}-\mathbf{x}\right\|_2^2}{2\sigma_K^2}\right) & \text{if } \left\|\mathbf{x}^{*}-\mathbf{x}\right\|_2 \leq \sqrt{3LU}\sigma_K \\ \frac{1}{Z} \exp\left(-\frac{\sqrt{3LU}}{\sigma_K}\left\|\mathbf{x}^{*}-\mathbf{x}\right\|_2\right) & \text{otherwise} \end{cases} \tag{15}$$

where Z *is the partition constant and* $\mathbf{x}^{*}$ *denotes the native protein structure corresponds to the amino-acid sequence* A*.*

Proof. In order to prove the convergence of sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$, we will show that one of its sub-sequences, $\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots$, converges to the above probability distribution, therefore the whole sequence also converges.

For any small constant ϵ $(0 < \epsilon \ll 1)$, with a sufficiently small α_n, we can find a sub-sequence $t_1 < t_2 < \ldots$ satisfying that:

$$\sum\nolimits_{t=t_s}^{t_{s+1}-1} \lambda_t^2 \to \epsilon \tag{35}$$

as $s \to \infty$.

The overall update process from $\mathbf{x}_{t_s}$ to $\mathbf{x}_{t_{s+1}}$ is given by:

$$\mathbf{x}_{t_{s+1}} = \mathbf{x}_{t_s} + \sum\nolimits_{t=t_s}^{t_{s+1}-1} \left[\frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \lambda_t \mathbf{z}_t\right] \tag{36}$$

where the update signal essentially consists of two parts: accumulated estimated gradients and random noise. For the latter one, since each iteration's random noise is independent, we have:

$$\left\|\sum\nolimits_{t=t_s}^{t_{s+1}-1} \lambda_t \mathbf{z}_t\right\|_2 = \mathcal{O}\left(\sqrt{\sum\nolimits_{t=t_s}^{t_{s+1}-1} \lambda_t^2}\right) = \mathcal{O}\left(\sqrt{\epsilon}\right) \tag{37}$$

Since the score network is assumed to have bounded variance, for any intermediate iteration t' $(t_s < t' \leq t_{s+1})$, the difference between $\mathbf{x}_{t'}$ and $\mathbf{x}_{t_s}$ is bounded by:

$$\begin{aligned} \left\|\mathbf{x}_{t'} - \mathbf{x}_{t_s}\right\|_2 &= \left\|\sum\nolimits_{t=t_s}^{t'-1} \left[\frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \lambda_t \mathbf{z}_t\right]\right\|_2 \\ &\leq \left\|\sum\nolimits_{t=t_s}^{t'-1} \frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right)\right\|_2 + \left\|\sum\nolimits_{t=t_s}^{t'-1} \lambda_t \mathbf{z}_t\right\|_2 \\ &\leq \frac{L_1}{2\sigma_{t'}'} \left\|\sum\nolimits_{t=t_s}^{t'-1} \lambda_t^2\right\|_2 + \mathcal{O}\left(\sqrt{\epsilon}\right) \\ &= \mathcal{O}\left(\epsilon\right) + \mathcal{O}\left(\sqrt{\epsilon}\right) \end{aligned} \tag{38}$$

which implies that the difference between the score network's outputs is also bounded:

$$\left\|s_{\boldsymbol{\theta}}\left(\mathbf{x}_{t'}, A\right) - s_{\boldsymbol{\theta}}\left(\mathbf{x}_{t_s}, A\right)\right\|_2 \leq L_2 \left\|\mathbf{x}_{t'} - \mathbf{x}_{t_s}\right\|_2 = \mathcal{O}\left(\epsilon\right) + \mathcal{O}\left(\sqrt{\epsilon}\right) \tag{39}$$

Therefore, for the accumulated score network's outputs in Eq. (36), we have:

$$
\begin{aligned}
&\sum\nolimits_{t=t_s}^{t_{s+1}-1} \frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) \\
=&\sum\nolimits_{t=t_s}^{t_{s+1}-1} \frac{\lambda_t^2}{2\sigma_K} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \sum\nolimits_{t=t_s}^{t_{s+1}-1} \left(\frac{\lambda_t^2}{2\sigma_t'} - \frac{\lambda_t^2}{2\sigma_K}\right) s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) \\
=&\sum\nolimits_{t=t_s}^{t_{s+1}-1} \frac{\lambda_t^2}{2\sigma_K} s^*\left(\mathbf{x}_t, \mathbf{x}^*\right) + \sum\nolimits_{t=t_s}^{t_{s+1}-1} \frac{\lambda_t^2}{2\sigma_K} \xi_{\boldsymbol{\theta}}\left(\mathbf{x}_t, \mathbf{x}^*, A\right) + \mathcal{O}\left(\epsilon\right) \\
=&\sum\nolimits_{t=t_s}^{t_{s+1}-1} \frac{\lambda_t^2}{2\sigma_K} s^*\left(\mathbf{x}_{t_s}, \mathbf{x}^*\right) + \sum\nolimits_{t=t_s}^{t_{s+1}-1} \frac{\lambda_t^2}{2\sigma_K} \left[s^*\left(\mathbf{x}_t, \mathbf{x}^*\right) - s^*\left(\mathbf{x}_{t_s}, \mathbf{x}^*\right)\right] + \mathcal{O}\left(\epsilon\right) \\
=&\frac{\epsilon}{2} \frac{s^*\left(\mathbf{x}_{t_s}, \mathbf{x}^*\right)}{\sigma_K} + \mathcal{O}\left(\epsilon\right)
\end{aligned}
\tag{40}
$$

where the second equality holds due to the bounded variance assumption, the third equality holds due to the bounded estimation error assumption, and the last equality holds due to the Lipschitz continuous property of oracle score function, based on its definition.

Hence, the iterative update process from t_s to t_{s+1} is essentially one gradient step at $\mathbf{x}_{t_s}$ with step size ϵ and re-scaled oracle score function, with a deviation term dominated by $\mathcal{O}\left(\epsilon\right)$ and random noise whose standard deviation equals $\sqrt{\epsilon}$. The $\mathcal{O}\left(\epsilon\right)$ deviation term can be omitted due to the existence of random noise. Therefore, we have:

$$
\begin{aligned}
\mathbf{x}_{t_{s+1}} &= \mathbf{x}_{t_s} + \sum\nolimits_{t=t_s}^{t_{s+1}-1} \left[\frac{\lambda_t^2}{2\sigma_t'} s_{\boldsymbol{\theta}}\left(\mathbf{x}_t, A\right) + \lambda_t \mathbf{z}_t\right] \\
&\approx \mathbf{x}_{t_s} + \frac{\epsilon}{2} s^*_{\sigma_K}\left(\mathbf{x}_{t_s}, \mathbf{x}^*\right) + \sqrt{\epsilon} z_{t_s}
\end{aligned}
\tag{41}
$$

The sub-sequence $\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots$ can be viewed as a sequence generated by Langevin dynamics with a constant step size of ϵ, which converges to the probability distribution defined by the re-scaled oracle score function, as $\epsilon \to 0$ and $T \to \infty$. According to Lemma 2, the re-scaled oracle score function's probability distribution is exactly the one as defined in Theorem 1, which concludes the proof.

□

Appendix B: Network Architecture

In MhaEGCL, multiple sub-networks are employed to compute edge-wise messages, query and key embeddings, attention coefficients, and weighting coefficients for estimated gradients. In our experiments, we implement these sub-networks with shallow MLP models, as listed in Table 2. The output dimension of each linear layer is given in the bracket, where $D_m = D_a = D_o = 16$. Three types of non-linear activation functions are used: Swish [25], ELU [6], and LeakyReLU [21].

Table 2: Network architectures in the MhaEGCL unit.

Network	Architecture
ϕ_m^l	Linear (D_m) - Swish - Linear (D_m)
ϕ_q^l	Linear (D_a) - ELU - Linear (D_a)
ϕ_k^l	Linear (D_a) - ELU - Linear (D_a)
ϕ_e^l	Linear (1) - LeakyReLU
ϕ_h	Linear (D_o) - Swish - Linear (D_o)
$\phi_{s,u}^l$	Linear (D_m) - Swish - Linear (1)

Appendix C: Node and Edge Features

Node features. Each node in the graph corresponds to a single amino-acid residue in the protein. We extract two types of node features: one-hot encoding of amino-acid types and positional encoding.

For the latter one, we introduce a "maximal sequence length" hyper-parameter L_{max} (although this also works for sequences longer than L_{max}), and extract features as follows:

$$f_{i,d} = \begin{cases} \sin\left[i/\text{pow}(L_{max}, \frac{d/2-0.5}{D/2-1})\right] & \text{if mod}\,(d,2) = 1 \\ \cos\left[i/\text{pow}(L_{max}, \frac{d/2-1}{D/2-1})\right] & \text{otherwise} \end{cases} \tag{42}$$

where i is the amino-acid residue's index in the sequence, and $d = 1, \ldots, D$ denotes the feature index in the D-dimensional positional encoding. For all the experiments in this paper, we set $L_{max} = 1000$ and $D = 24$.

Edge features. Each edge in the graph corresponds to a pair of amino-acid residues. As a commonly-used representation for inter-residue interactions, we adopt a MSA-based neural network to predict inter-residue distance and orientations [42], following trRosetta's definitions:

- d_{ij}: Euclidean distance between $\text{C}_\beta^{(i)}$-$\text{C}_\beta^{(j)}$ atoms
- ω_{ij}: dihedral angle defined by $\text{C}_\alpha^{(i)}$-$\text{C}_\beta^{(i)}$-$\text{C}_\beta^{(j)}$-$\text{C}_\alpha^{(j)}$ atoms
- γ_{ij}: dihedral angle defined by $\text{N}^{(i)}$-$\text{C}_\alpha^{(i)}$-$\text{C}_\beta^{(i)}$-$\text{C}_\beta^{(j)}$ atoms
- φ_{ij}: plane angle defined by $\text{C}_\alpha^{(i)}$-$\text{C}_\beta^{(i)}$-$\text{C}_\beta^{(j)}$ atoms

All of above distance and orientation angles are discretized into histogram bins (37 bins for d_{ij}, 25 bins for ω_{ij} and γ_{ij}, and 13 bins for φ_{ij}). For each residue pair, we obtain a 100-dimensional feature vector describing their relative distance and orientations, which is then used as edge features.

Appendix D: Detailed Results for Top-ranked Models

For models built with EGCL modules (residue format: C_α-trace), following hyper-parameters are tuned:

- $D_m \in \{16, 32, 64\}$: number of dimensions for edge-wise messages
- $N_s \in \{4, 8, 12\}$: number of sequence-based neighbors per residue
- $N_c \in \{4, 8, 12\}$: number of coordinate-based neighbors per residue
- $N_r \in \{4, 8, 12\}$: number of random neighbors per residue
- $\eta \in \{0.001, 0.003, 0.010\}$: Adam optimizer's learning rate

Please note that we limit $N_c = N_r$ to narrow down the hyper-parameter search space. A total of 81 models are trained, from which top-10 models with lowest validation losses are kept for the subsequent sampling process on validation (for hyper-parameter tuning) and test (for final results) subsets. Per-model results are as listed in Table 3.

Table 3: Top-10 EGCL-based C_α-trace models' hyper-parameters, validation losses, and structure optimization results on the test subset (CALD sampling).

No.	D_m	N_s	N_c	η	Loss	lDDT-Ca (Avg.)	lDDT-Ca (Max)
1	64	8	4	0.001	1.3509	0.7425	0.7858
2	64	12	4	0.001	1.3605	0.7641	0.7941
3	64	4	8	0.001	1.3706	0.7712	0.7986
4	64	8	8	0.001	1.3717	0.7668	0.7932
5	32	8	4	0.003	1.3778	0.7702	0.7995
6	32	8	8	0.003	1.3789	0.7766	0.8021
7	32	8	8	0.001	1.3797	0.7530	0.7943
8	64	12	8	0.001	1.3832	0.7755	0.8033
9	32	12	8	0.001	1.3842	0.7581	0.7928
10	64	4	12	0.001	1.3907	0.7755	0.8008

For models built with MhaEGCL modules (residue format: C_α-trace), following hyper-parameters are tuned:

- $N_h \in \{1, 2, 4\}$: number of attention heads
- $N_s \in \{4, 8, 12\}$: number of sequence-based neighbors per residue
- $N_c \in \{4, 8, 12\}$: number of coordinate-based neighbors per residue
- $N_r \in \{4, 8, 12\}$: number of random neighbors per residue
- $\eta \in \{0.001, 0.003, 0.010\}$: Adam optimizer's learning rate

Similarly, we limit $N_c = N_r$ to reduce the hyper-parameter search space. We fix $D_m = 16$ so that the overall model size is similar with EGCL-based models. In Table 4, we report top-10 MhaEGCL-based models' hyper-parameter settings and structure optimization results.

Table 4: Top-10 MhaEGCL-based C_α-trace models' hyper-parameters, validation losses, and structure optimization results on the test subset (CALD sampling).

No.	N_h	N_s	N_c	η	Loss	lDDT-Ca (Avg.)	lDDT-Ca (Max)
1	4	12	12	0.003	1.2841	0.7962	0.8183
2	4	8	12	0.003	1.2932	0.7820	0.8069
3	4	12	8	0.003	1.2988	0.7884	0.8113
4	4	4	12	0.003	1.2997	0.7896	0.8127
5	4	4	12	0.010	1.3046	0.7887	0.8120
6	2	12	8	0.003	1.3095	0.7847	0.8102
7	2	8	12	0.003	1.3120	0.7921	0.8135
8	4	8	8	0.003	1.3126	0.7932	0.8152
9	4	8	12	0.001	1.3164	0.7896	0.8147
10	2	12	12	0.003	1.3167	0.7979	0.8192

For models built with MhaEGCL modules (residue format: backbone), following hyper-parameters are tuned:

- $N_h \in \{1, 2, 4\}$: number of attention heads
- $N_s \in \{4, 8, 12\}$: number of sequence-based neighbors per residue
- $N_c \in \{4, 8, 12\}$: number of coordinate-based neighbors per residue
- $N_r \in \{4, 8, 12\}$: number of random neighbors per residue
- $\lambda_b \in \{0.001, 0.01\}$: weighting coefficient for estimated gradients over non-C_α atoms
- $\eta \in \{0.001, 0.003, 0.010\}$: Adam optimizer's learning rate

where λ_b is newly introduced to control non-C_α atoms' contribution to the optimization process. We intentionally put more weights on C_α atoms since 1) they are more critical in determining the overall structure and 2) the radial distance $\mathbf{r}_{ij}$ for computing edge-wise messages only relies on C_α atoms. Similar as above, we let $N_c = N_r$ and $D_m = 16$. In Table 5, we report detailed results for top-10 MhaEGCL-based backbone models.

Table 5: Top-10 MhaEGCL-based backbone models' hyper-parameters, validation losses, and structure optimization results on the test subset (CALD sampling).

No.	N_h	N_s	N_c	λ_b	η	Loss	lDDT-Ca (Avg.)	lDDT-Ca (Max)
1	4	12	12	0.010	0.003	1.2941	0.7988	0.8201
2	4	12	12	0.001	0.003	1.2945	0.7980	0.8186
3	4	12	8	0.001	0.003	1.2950	0.7899	0.8129
4	4	4	12	0.001	0.003	1.3056	0.7929	0.8151
5	4	8	12	0.010	0.003	1.3058	0.7944	0.8165
6	4	4	12	0.010	0.003	1.3124	0.7925	0.8147
7	4	8	12	0.001	0.003	1.3126	0.7930	0.8155
8	2	12	12	0.001	0.003	1.3134	0.7894	0.8105
9	4	12	8	0.010	0.003	1.3159	0.7913	0.8147
10	4	12	12	0.010	0.001	1.3165	0.7874	0.8094

Appendix E: Analysis on the Computational Complexity

For the sake of convenience, we summarize all the notations to be used for analyzing the computation complexity of MhaEGCL units in Table 6. Each symbol's typical value is given by either the default setting in our experiments or its largest candidate in the hyper-parameter search space.

Table 6: Notations for analyzing the computation complexity of MhaEGCL units.

Symbol	Value	Description
N_v	-	Number of nodes in the graph
N_e	-	Number of edges in the graph
U	4	Number of atoms associated with each node
D_v	44	Number of dimensions in node features
D_e	100	Number of dimensions in edge features
D_m	16	Number of dimensions in edge-wise messages
D_a	16	Number of dimensions in query/key embeddings
D_o	16	Number of dimensions in hidden node embeddings
N_h	4	Number of attention heads
N_s	12	Number of sequence-based neighbors per residue
N_c	12	Number of coordinate-based neighbors per residue
N_r	12	Number of random neighbors per residue

The number of nodes in the graph, N_v, equals to the amino-acid sequence length, which is usually within 1000, and the number of edges is given by $N_e = (N_s + N_c + N_r)N_v$. For each MhaEGCL unit, we list all the sub-networks' total computational complexity as below:

- $\{\phi_m^l\}$: $\mathcal{O}(N_h N_e D_v D_m) + \mathcal{O}(N_h N_e D_e D_m) + \mathcal{O}(N_h N_e D_m^2)$
- $\{\phi_q^l\}$: $\mathcal{O}(N_h N_v D_v D_a) + \mathcal{O}(N_h N_v D_a^2)$
- $\{\phi_k^l\}$: $\mathcal{O}(N_h N_e D_v D_a) + \mathcal{O}(N_h N_e D_m D_a) + \mathcal{O}(N_h N_e D_a^2)$
- $\{\phi_e^l\}$: $\mathcal{O}(N_h N_e D_a)$
- $\{\phi_h\}$: $\mathcal{O}(N_h N_e D_m) + \mathcal{O}(N_v D_m D_o) + \mathcal{O}(N_v D_o^2)$
- $\{\phi_{s,u}^l\}$: $\mathcal{O}(N_h N_e D_m^2 U) + \mathcal{O}(N_h N_e D_m U)$

Since the number of dimensions in node/edge features and intermediate embeddings are basically in the same scale, the overall time complexity of one MhaEGCL unit is approximately dominated by $\mathcal{O}(N_h N_e D_m^2 U)$. In short, the computational overhead is linear to the number of attention heads, edges in the graph, and atoms associated with each node, and quadratic to the number of dimensions in intermediate embeddings. Since we limit the number of edges per nodes by $(N_s + N_c + N_r)$, the overall time complexity is still linear to the number of nodes, *i.e.* the amino-acid sequence length. This is quite favorable, since the time consumption of many Rosetta-based structure optimization methods is quadratic to the sequence length. This is also verified by the comparison on the structure optimization time, as illustrated in Figure 2.

Appendix F: Additional Visualization Results

Here, we take a few test domains as examples, and visualize 2D distance matrices and 3D structures during SE(3)-Fold's sampling process. For subsequent experiments, we use MhaEGCL-based C_α-trace models with CALD sampling. Since SE(3)-Fold only generates C_α-trace models, we derive corresponding backbone models with PDB_Tool[3], and then use SCWRL4 [20] to recover side-chain structures.

In Figure 4, 5, and 6, we present visualization on the sampling process for "1zjcA01", "2eiyB02", and "5kjzA00", respectively. We observe that although the initial structure is completely different from the native one (second column), major patterns quickly emerge in the 2D distance matrix within 50-100 iterations. The distance between adjacent residues' C_α atoms is not yet correctly preserved at

[3] `https://github.com/realbigws/PDB_Tool`

this stage, as indicated by the vast majority of dashed lines in full-atom models' visualization. After 200 iterations, the overall topology becomes clear and some of secondary structures (*e.g.* α-helices) are already partially visible in C_α-trace and full-atom models. Finally, when 500 iterations are accomplished, almost all the secondary structures are correctly recovered, and the overall topology is also highly similar with the native structure.

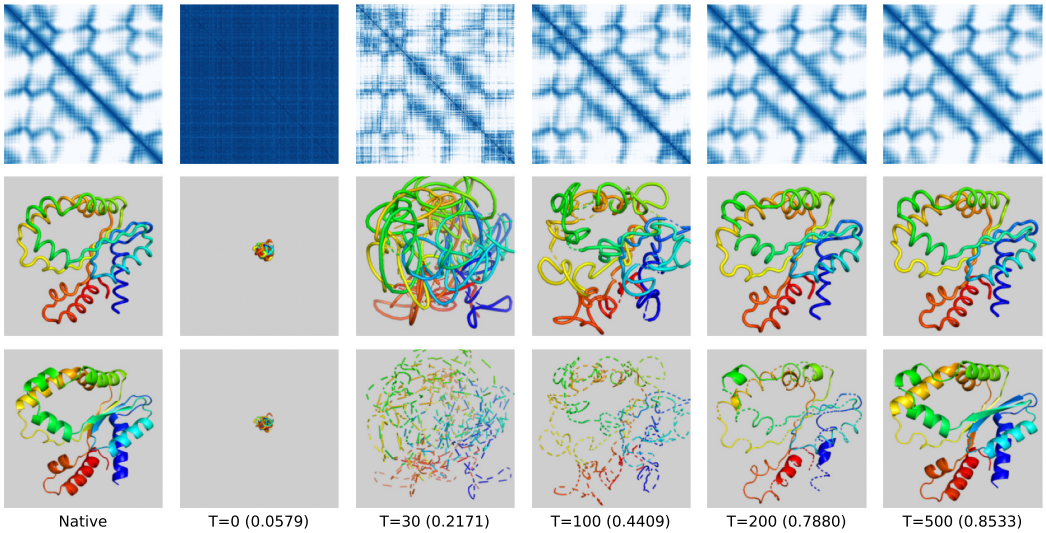


Figure 4: Visualization on SE(3)-Fold's structure optimization process (CATH domain ID: 1zjcA01). Top: C_α-C_α distance matrix (darker color corresponds to shorter distance). Middle: C_α-trace models. Bottom: full-atom models recovered with PDB_Tool and SCWRL4.

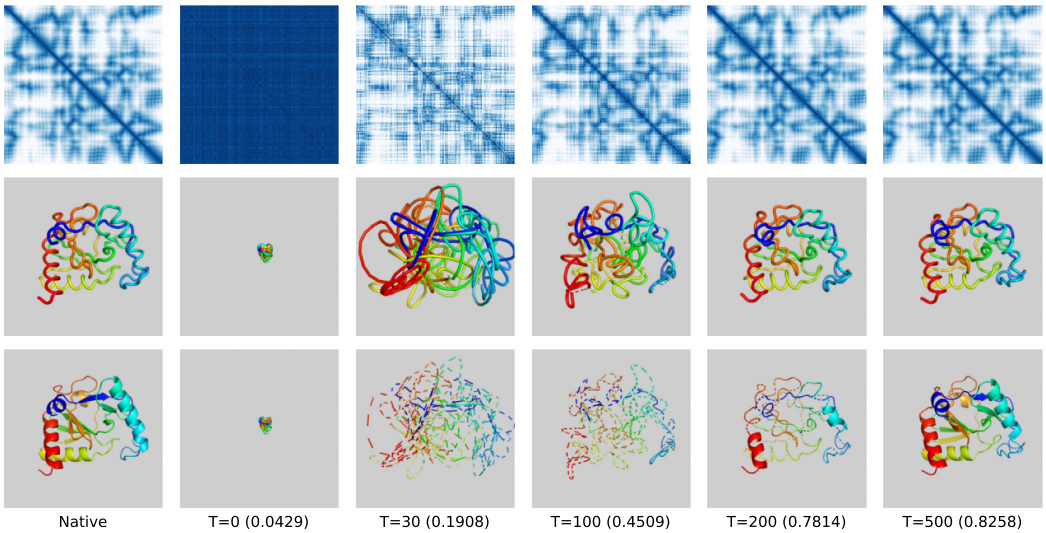


Figure 5: Visualization on SE(3)-Fold's structure optimization process (CATH domain ID: 2eiyB02). Top: C_α-C_α distance matrix (darker color corresponds to shorter distance). Middle: C_α-trace models. Bottom: full-atom models recovered with PDB_Tool and SCWRL4.

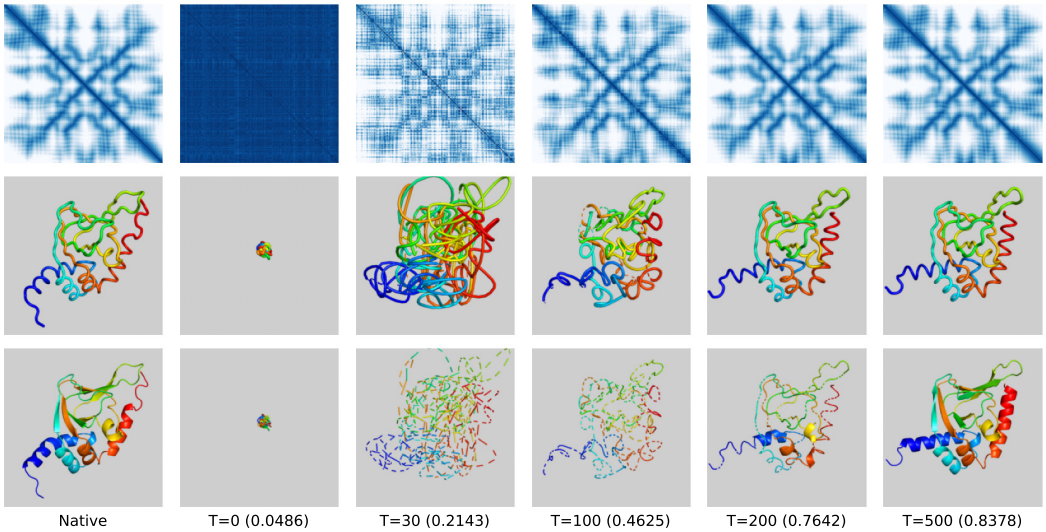


Figure 6: Visualization on SE(3)-Fold's structure optimization process (CATH domain ID: 5kjzA00). Top: C_α-C_α distance matrix (darker color corresponds to shorter distance). Middle: C_α-trace models. Bottom: full-atom models recovered with PDB_Tool and SCWRL4.