DSR: Dynamical Surface Representation as Implicit Neural Networks for Protein

Daiwen Sun[1], **He Huang**[2,1], **Yao Li**[2,3], **Xinqi Gong**[1,2]*, **Qiwei Ye**[2]*
[1]Institute for Mathematical Sciences, School of Mathematics,
Renmin University of China, Beijing, China
[2]Beijing Academy of Artificial Intelligence, Beijing, China
[3]School of Life Sciences, Tsinghua University, Beijing, China
sundaiwen@ruc.edu.cn, huanghe@baai.ac.cn, liyao17@tsinghua.org.cn,
xinqigong@ruc.edu.cn, qwye@baai.ac.cn

Abstract

We propose a novel neural network-based approach to modeling protein dynamics using an implicit representation of a protein's surface in 3D and time. Our method utilizes the zero-level set of signed distance functions (SDFs) to represent protein surfaces, enabling temporally and spatially continuous representations of protein dynamics. Our experimental results demonstrate that our model accurately captures protein dynamic trajectories and can interpolate and extrapolate in 3D and time. Importantly, this is the first study to introduce this method and successfully model large-scale protein dynamics. This approach offers a promising alternative to current methods, overcoming the limitations of first-principles-based and deep learning methods, and provides a more scalable and efficient approach to modeling protein dynamics. Additionally, our surface representation approach simplifies calculations and allows identifying movement trends and amplitudes of protein domains, making it a useful tool for protein dynamics research. Codes are available at https://github.com/Sundw-818/DSR, and we have a project webpage that shows some video results, https://sundw-818.github.io/DSR/.

1 Introduction

Proteins, which are made up of chains of amino acids, are essential molecules in living organisms. Protein with specific three-dimensional structures are responsible for a wide range of biological processes such as enzymatic reactions, transport of molecules, and structural support. However, proteins are not static objects in vivo, but constantly in motion, with individual amino acids vibrating, rotating, and shifting in response to the change of environment. The biological functions of proteins are fulfilled by their dynamical behaviors, and researchers study the dynamics and functions of proteins by molecular dynamics(MD) simulations[1, 2].

First-principles molecular dynamics use the forces calculated from the electronic states by solving the Schrödinger equation (namely, density functional theory) to compute the coordinate trajectories[3], with programs such as VASP[4] and GPAW[5]. However, the computational expense of MD typically increases exponentially in relation to the number of electronic degrees of freedom. In order to reduce the computational cost, empirical force field (classical force field) models with simplified functional forms are employed in MD simulation, such as AMBER[6, 7], CHARMM[8, 9], and GROMOS[10–12], which are designed to capture the bonded and non-bonded interactions between atoms. Besides, machine learning techniques can be employed to fit force fields or potential energy surfaces[13–16]. These force field-based methods are much faster than ab initio methods without exhaustive QM

*Correspondence to: Xinqi Gong<xinqigong@ruc.edu.cn>, Qiwei Ye <qwye@baai.ac.cn>.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

calculations. However, the integration time step in MD cannot be too large to resolve the fastest motion in the system, for example the vibration of hydrogen-involved bonds whose timescales are as short as a few femtoseconds. This restricts MD from having more speed advantages. To address this problem, coarse-grained methods are being developed to perform molecular dynamics simulations faster and on larger systems by eliminating some unimportant degrees of freedom[17–20]. Fast as it is, coarse-grained simulations are only able to reproduce molecular properties at low resolution, such as Rg^2, sacrificing the all-atom details. Given this perspective, existing molecular dynamics approaches suffer from either unaffordable computational costs or insufficient precision. Moreover, inherent limitations such as the considerable random fluctuation of simulated trajectories and the consequential low signal-to-noise ratio pose significant challenges in detecting crucial protein functional motion. Even for the current most advanced Anton 3, under the parallel processing of 512 nodes, for a large system of 1 million atoms, only 100 microseconds of MD simulation can be performed each day[21].

These aforementioned methods give thorough understanding of the natures of proteins by modeling protein molecules as discrete particles. But when studying protein-protein interaction or protein-ligand binding, knowledge about surfaces of proteins is what researchers need. To fulfill such demands, a continuous, high-level representation of molecular shape known as the molecular surface is proposed, as shown in Figure 1.Based on the the concept of "key-lock pairs" emphasizing complementarity of molecular surface[22], the molecular surface models a protein as a continuous shape with geometric and chemical features. One representative method, molecular surface interaction fingerprinting (MaSIF), is a geometric deep learning method that captures fingerprints important for specific biomolecular interactions, enabling accurate and efficient analysis of protein-ligand complexes and potential applications in drug discovery and molecular design[23, 24]. Furthermore, some works investigated the application of end-to-end learning techniques for the analysis of protein surfaces and the identification of potential functional sites, such as drug-target binding sites[25].

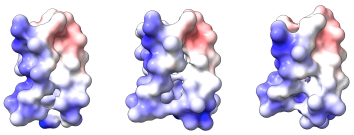

Figure 1: The protein surface representation. Showing the protein surface (PDB ID: 1PGA) at different time steps in a molecular dynamics trajectory.

Previously, surface representation was mainly used for studying inter-molecular interactions or assisting molecular docking, and it had not been applied to analysing molecular dynamics simulation data. Inspired by the above works, we propose Dynamical Surface Representation (abbreviated as DSR), a method for modeling protein dynamics by representing protein surfaces with implicit neural networks. Implicit neural representations[26, 27] are a class of deep learning models that can represent complex geometric shapes using a continuous function without requiring an explicit surface representation. It's well-suited for modeling the dynamic shapes of proteins which have complex and varied conformations. Currently, deep learning applications in molecular dynamics (MD) mainly target small molecules, while our method can be applied to very large proteins.

In this paper, we explore the use of implicit neural representations for modeling the dynamic shapes of proteins. To be specific, protein surfaces are represented as a zero-level set of signed distance function (SDF) in DSR. The value of SDF is defined as the minimum distance from a point in Euclidean space to the surface of the object, where the positive value indicates that the point is outside the object, otherwise, it is inside. In addition, we introduce a time variable in DSR model to simulate the change of the protein surface over time, which is the dynamic simulation of the protein. In summary, we use the idea of implicit neural representation and use the neural network as the implicit neural representation of points on the 3D and time region, so as to achieve the purpose of modeling protein dynamic changes in continuous space and time domain.

In summary, our paper's main contribution is two-fold:

- We build an implicit neural network that learns SDF from raw point cloud data in 3D + time domain, which is a both temporally and spatially continuous protein dynamic representation.

- We have achieved dynamic modeling of large proteins by using surface representations, which overcomes the limitations of previous atomic models or coarse-grained models that were restricted to small molecules or proteins.

2 DSR

In this section, we present DSR, a dynamical surface representation as implicit neural networks for protein. We first introduce the definition of SDF, and then describe how we use SDF to model proteins. The architecture of the whole method is shown in Figure 2.

2.1 SDF Representation

Let $\Omega = [-1, 1]^3$ be a spatial domain, $\tau = [-1, 1]$ a temporal domain, and $\mathcal{M}_t$ be a 3D manifold embedded in Ω at time $t \in \tau$. For any point $\boldsymbol{x} = (x, y, z) \in \Omega$ at time t, the $f_{\mathcal{M}_t} : \Omega \to \mathbb{R}$ is define as

$$f_{\mathcal{M}_t}(\boldsymbol{x}) = \begin{cases} \mathrm{d}(\boldsymbol{x}, \mathcal{M}_t) & \text{if } x \text{ outside } \mathcal{M}_t \\ 0 & \text{if } x \text{ belonging to } \mathcal{M}_t \\ -\mathrm{d}(\boldsymbol{x}, \mathcal{M}_t) & \text{if } x \text{ inside } \mathcal{M}_t \end{cases} \tag{1}$$

where $\mathrm{d}(x, \mathcal{M}_t) := \inf_{y \in \mathcal{M}_t} \mathrm{d}(x, y)$, $\mathrm{d}(x, y) = \|x - y\|_2$.

The zero-level set, which is the surface of the protein at time t, is presented by the points where $f_{\mathcal{M}_t}(\cdot) = 0$. That is, the points on the protein surface at time t can be represented as set $\{\boldsymbol{x} \mid f_{\mathcal{M}_t}(\boldsymbol{x}) = 0, \boldsymbol{x} \in \Omega\}$.

2.2 Modeling Protein Dynamical Surface with SDFs

The protein surface representation is in the form of a 3D shape. In order to utilize the implicit function form of SDF, a typical approach is to partition the space into a grid, and then compute the SDF value for each grid point. However, for irregular objects, the accurate SDF values cannot be computed and thus approximation algorithms, such as the 8SSEDT algorithm, are usually employed. This method is associated with two limitations: firstly, protein surfaces are often intricate and rugged, and estimation errors can result in larger biases after modeling; secondly, the SDF calculation for dynamic models is linearly dependent on time and cubically dependent on resolution, and pre-computed SDFs will incur high computational costs. Hence, an alternative strategy is used herein, whereby SDF is learned directly from the raw point clouds, rather than utilizing pre-computed SDF for supervised learning. This completely circumvents the need for pre-computing SDF, significantly reduces computational costs, and enhances computational efficiency. In the following, we will elaborate on how to learn SDF directly from the raw point clouds.

For a given input point cloud $\mathcal{X} = \{\boldsymbol{x}_i\}_{i \in I} \subset \mathbb{R}^3$ at time t,with point normal vector data (optional), $\mathcal{N} = \{\boldsymbol{n}_i\}_{i \in I} \subset \mathbb{R}^3$, our objective is to find the optimal parameters θ of an MLP $f(\boldsymbol{x}, t; \theta)$, where $f : \mathbb{R}^{4+d_z} \to \mathbb{R}$, that can accurately estimate a signed distance function to a surface $\mathcal{M}_t$ defined by the point cloud $\mathcal{X}$ and normals $\mathcal{N}$ at time t.

The form of our loss function is as follows:

$$\ell(\theta) = \ell_{\mathcal{X}}(\theta) + \lambda \mathbb{E}_{\boldsymbol{x}} \left(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, t; \theta)\| - 1 \right)^2 + \alpha \|z\| \tag{2}$$

where $\lambda > 0$ is a parameter, $\|\cdot\| = \|\cdot\|_2$ is the Euclidean 2-norm, and

$$\ell_{\mathcal{X}}(\theta) = \frac{1}{|I|} \sum_{i \in I} \left(|f(\boldsymbol{x}_i, t; \theta)| + \tau \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i, t; \theta) - \boldsymbol{n}_i\| \right) \tag{3}$$

promotes the vanishing of f on $\mathcal{X}$, and in the presence of normal data (i.e., $\tau = 1$) , $\nabla_{\boldsymbol{x}} f$ to the given normals $\mathcal{N}$.

The first term in the summation part of Eq. (3) indicates that the SDF value of the surface points should be as small as possible. The second term represents the loss between the point normal vector and the ground truth.

The second term in Eq. (2) called the *Eikonal term* promotes the gradients $\nabla_{\boldsymbol{x}} f$ to possess a 2-norm of unity, i.e. $\|\nabla_{\boldsymbol{x}} f\| = 1$, which is the gradient property of SDFs. Note that this property is crucial to the loss design, allowing us to avoid using pre-computed SDF values for supervised learning. While its sufficiency has not been fully established in previous literature, we provide a more comprehensive proof in the Appendix B.

The third term z in Eq. (2) is a latent code that we introduce into the model to specify different protein types, and its parameters are learnable. We claim that with sufficient data, we can explore the latent space for greater generalization.

The global minimum of the loss in Eq. (2) will be the solution of the Eikonal partial differential equation, i.e.,

$$\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\| = 1, \tag{4}$$

which will also be a signed distance function, where f approaches 0 on $\mathcal{X}$, with gradients $\mathcal{N}$. The calculation of the expectation is based on some probability distribution $\boldsymbol{x} \sim \mathcal{D}$ in $\mathbb{R}^3$, which will be introduced in Section 3.2.

Throughout the optimization process, we did not use the pre-computed SDF value to supervise the training, which is an end-to-end learning architecture from raw point clouds data to SDF value. In the next section, we will introduce the experiment setup in detail, including the datasets, implementation details, and model evaluation metrics.

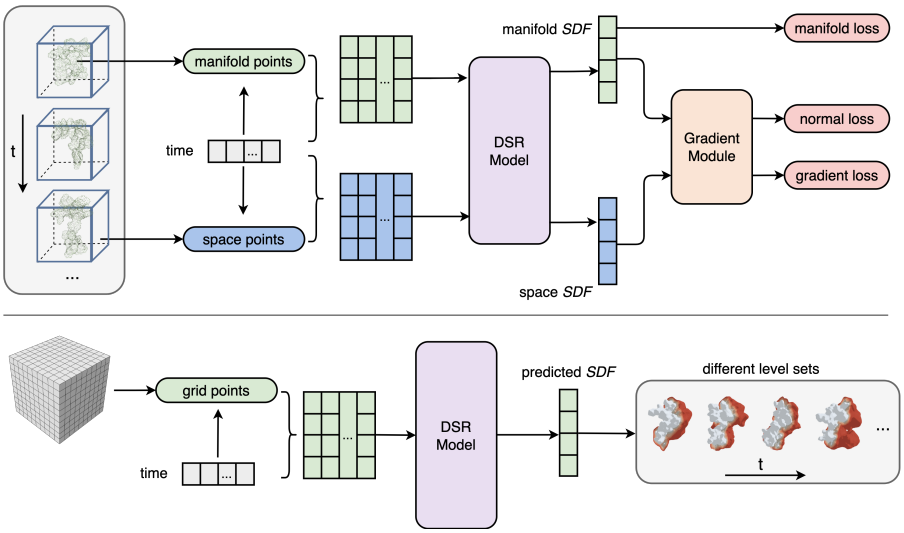

Figure 2: Overview of DSR. The upper part is the training process. First, the manifold and space points are sampled at different times, added the time dimension and be input into the DSR Model to obtain SDF. And then the Gradient Module is used to calculate the derivative of SDF with respect to $(\boldsymbol{x}, t)$. The lower part is the inference process. We first divide the space into a grid based on the desired resolution, then adds the time dimension and inputs them into the DSR Model to predict the SDF. Finally, the Marching Cube algorithm is utilized to generate different level sets, with the zero-level set representing the protein dynamical surface.

3 DSR on large-scale proteins simulation data

3.1 Dataset.

500ns Trajectory Data. This data set[28] currently contains two 100ns atomistic molecular dynamics trajectories of Abeta (40 residues), one wild type and one E22G, following the protocol of the 500ns trajectories published in [29]. The simulation was performed at 310K in a generalized Born implicit solvent. In this work, we used the first 2000 frames of wild type trajectories.

MDAnalysisData. MDAnalysisData collects a set of data resources pertaining to computational biophysics, primarily focusing on molecular dynamics (MD) simulations and the structural and dynamic attributes of biomolecules. We used five of these datasets,

(1) AdK equilibrium dataset[30] of 4187 frames,

(2) AdK conformation transition[31] sampled with two methods:

- Dynamic importance sampling molecular dynamics (DIMS MD) of 102 frames[32],
- Framework Rigidity Optimized Dynamics Algorithm (FRODA) of 142 frames[33],

(3) A short MD trajectory of I-FABP (intestinal fatty acid binding protein) of 500 frames[34],
(4) A trajectory of the NhaA membrane protein of 5000 frames[35],
(5) Two MD trajectories of YiiP membrane protein, the short one has 900 frames, and the long one has 9000 frames[36]. We opted for the longer one here.

For these 6 trajectories data, we use a fixed size 2000 to crop the frames due to the computational resources, and we keep the full length for those shorter than the cropping size.

GPCRmd. The GPCRmd (http://gpcrmd.org/)[37] is an online platform that incorporates web-based visualization capabilities and shares data. The GPCRmd database includes at least one representative structure from each of the four structurally characterized GPCR classes. The GPCRmd platform holds more than 600 GPCR MD simulations from GPCRmd community and individual contributions. Each system was simulated for 500 ns in three replicates (total time 1.5 μs). We selected one system per GPCR family, and a total of 4 trajectories were used to train the model, of which trajectory ids are 10792_trj_81, 10840_trj_87, 10912_trj_95 and 15711_trj_791 respectively.

DRYAD_MD. This dataset contains the trajectory data of 23 proteins simulated by Jumper JM[38]. It is worth noting that the protein trajectories within this dataset exhibit significant oscillations, posing a challenge for model training and resulting in a high failure rate. For this reason, we screened out 14 proteins with mean RMSD[2] less than 1.0 for our experiments.

3.2 Implementation Details.

Problem setup. In this paper, we use PyMol[39] to obtain the protein surface representation, the points and normal vectors, to train the model. The goal is to obtain a protein surface dynamic model with respect to continuous space and time.

Architecture. For representing shapes we used level sets of MLP $f(x, t; \theta) = \mathbb{R}^4 \times \mathbb{R}^{d_z} \to \mathbb{R}$, called DSR Model, with 8 layers, each containing 512 hidden units, and a single skip connection from the input to the middle layer. We set the loss parameters to $\lambda = 0.1, \tau = 1, \alpha = 1e-3$. And the model architecture is as Figure 2. The activation function between fully connected layers is softplus activation: $x \mapsto \frac{1}{\beta} \ln\left(1 + e^{\beta x}\right)$, where $\beta = 100$. The initial latent code vector $\mathbf{z}$ of size 192, were sampled from $\mathcal{N}(0, 1.0^2)$. The Gradient Module was implemented by PyTorch Autograd to calculate $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$.

Distribution $\mathcal{D}$. For all experiments, we utilize the distribution $\mathcal{D}$ as defined in the IGR[40] for the expectation in Eq. (2). This distribution is determined by taking the average of a uniform distribution and a sum of Gaussians that are centered at $\mathcal{X}$ with a standard deviation equal to the distance to the k-th nearest neighbor (where we set $k = 50$).

Level set extraction. We extract the zero (or any other) level set of a trained model $f(x, t; \theta)$ using the *Marching Cubes* algorithm[41] implemented in the *python scikit-image* package, which can use any large-size grid to achieve any high resolution.

3.3 Evaluation Metrics.

We use three evaluation metrics commonly used in 3D modeling to evaluate the similarity between two 3D shapes from three aspects: volume, distance, and normal vectors, which are *Volumetric Intersection over Union (IoU)*, *Chamfer distance* and *Normal Consistency (NC)*. These three metrics are all normalized to a range of 0-1, providing a comprehensive evaluation of the model's performance from different perspectives.

IoU. IoU compares the reconstructed volume with the ground truth shape (higher is better). For two arbitrary shapes $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$ is attained by:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{5}$$

[2]The RMSD here was calculated between consecutive frames to reflect the smoothness of the protein trajectory.

Chamfer distance. Chamfer distance is a standard metrics to evaluate the distance between two point sets $\mathcal{X}_1, \mathcal{X}_2 \subset \mathbb{R}^n$ (lower is better).

$$\mathrm{d_C}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \frac{1}{2}\left(\mathrm{d}_{\overrightarrow{\mathrm{C}}}\left(\mathcal{X}_1, \mathcal{X}_2\right) + \mathrm{d}_{\overrightarrow{\mathrm{C}}}\left(\mathcal{X}_2, \mathcal{X}_1\right)\right) \tag{6}$$

where

$$\mathrm{d}_{\overrightarrow{\mathrm{C}}}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \frac{1}{|\mathcal{X}_1|}\sum_{\boldsymbol{x}_1 \in \mathcal{X}_1} \min_{\boldsymbol{x}_2 \in \mathcal{X}_2} \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \tag{7}$$

NC NC evaluate estimated surface normals (higher is better). Normal consistency between two normalized unit vectors n_i and n_j is defined as the dot product between the two vectors. For evaluating the surface normals, given the object surface points and normal vectors: $X_{pred} = \{(\boldsymbol{x}_i, \overrightarrow{n_i})\}$, and the ground truth surface points and normal vectors: $X_{gt} = \{(\boldsymbol{y}_j, \overrightarrow{m_j})\}$, the surface normal consistency between X_{pred} and X_{gt}, denoted as Γ, is defined as:

$$\Gamma\left(X_{gt}, X_{\text{pred}}\right) = \frac{1}{|X_{gt}|}\sum_{j \in |X_{gt}|}\left|\overrightarrow{n_j} \cdot \overrightarrow{m}_{\theta\left(\boldsymbol{y}_j, X_{\text{pred}}\right)}\right| \tag{8}$$

where

$$\theta\left(\boldsymbol{y}_j, X_{\text{pred}} := \{(\boldsymbol{x}_i, \overrightarrow{n_i})\}\right) = \underset{i \in |X_{\text{pred}}|}{\arg\min}\left\|\boldsymbol{y}_j - \boldsymbol{x}_i\right\|_2^2 \tag{9}$$

4 Results and Analysis

We design experiments from three aspects to verify the ability to learn SDF from the raw point clouds, the ability to reconstruct the protein dynamical surface, and the generalization of the model in terms of temporal interpolation and extrapolation. To our knowledge, we are the first to model long-term dynamics of large proteins through surface representation, with no other methods currently available for comparison. Therefore, we conduct the following analysis to verify the learning ability of the DSR model and the effectiveness of protein surface representation. Additionally, to demonstrate that our model is not doing something trivial, we compare our method with linear interpolation on interpolation tasks.

4.1 The Ability to Learn SDF

We show the reconstruction results of training on Abeta protein in Figure 3, which demonstrates the ability of our model to learn the signed distance field. The first row in the figure is the reconstruction results of our model, and the second row is the ground truth. It can be seen that our model has the ability to reconstruct the dynamic shape of the protein at different moments, and can restore a clearer and more delicate shape than the ground truth. Furthermore, we show some videos of this protein on the webpage and discuss the effect of normal vectors on learning process in the Appendix D.

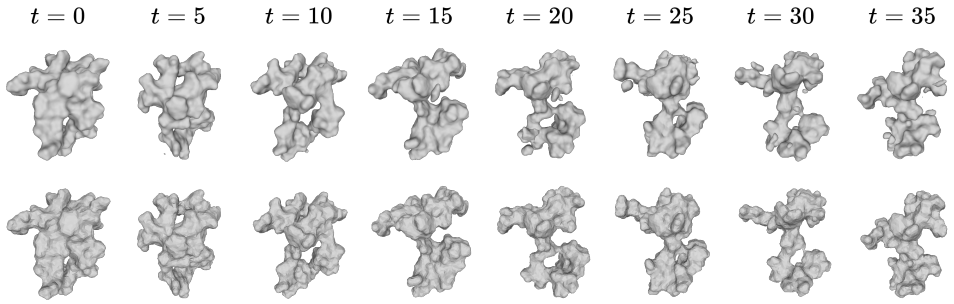

Figure 3: Given different times as input, the zero-level set visualization of the SDF value predicted by the model for protein abeta. The first row is the reconstruction result of our model, and the second row is the ground truth.

Note that SDF represents the shortest distance (signed) from the surface of the object, which means that different level sets form surfaces at certain distances from the manifold surface. As shown

in Figure 4 (a), the visualization of different level sets of SDF values predicted by our model is consistent with the meaning expressed by SDF. In addition, our model is continuous with respect to spatial and can use to reconstruct and predict for arbitrary space resolutions. Figure 4(b) shows the reconstruction of a protein at different resolutions at a certain time. We also calculated three evaluation metrics, namely IoU, Chamfer distance and NC, which have values of 0.9361±0.0265, 0.0009±0.0008 and 0.9967±0.0011 respectively.

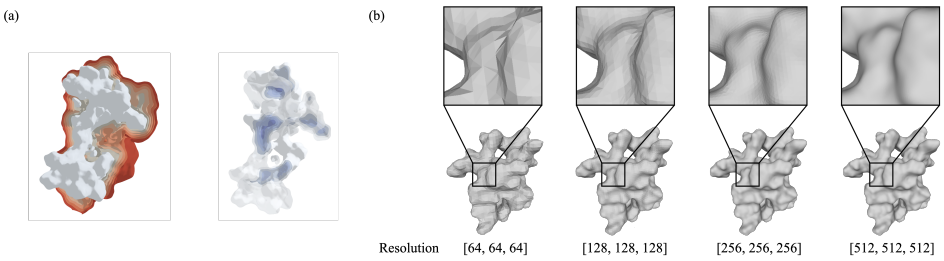

Figure 4: (a) The visualization of different level-set of the SDF predicted by our model; positive level sets are in red; negative ones are in blue; the zero level set, represents the approximated surface in white. (b) The reconstruction of a protein at different resolutions.

4.2 Reconstruction across Time

In this experiment, we validate the ability of our trained model to reconstruct the protein shapes in training frames (see Table 1). For data AdK equilibrium, NhaA, YiiP, and DRYAD_MD, we used the first 1500 frames of each trajectory and diluted them by ten times. That is, selecting 150 frames for each trajectory for training. Similarly, every second frame was selected within the first 80 frames for DIMS, every third frame was chosen within the first 120 frames for FRODA, and every fifth frame was selected within the first 400 frames for I-FABP.

As we can see from the Table 1, the model can reconstruct well in many cases. For example, DIMS, the IoU value is as high as 0.9316, which means that the shape reconstructed by the model is almost completely consistent with the ground truth on the entire timeline.

Table 1: Evaluation on the surface reconstruction across time.

	IoU↑	Chamfer_dist↓	NC↑		IoU↑	Chamfer_dist↓	NC↑
DIMS	0.9316±0.0070	0.0003±0.0001	0.9671±0.0050	I-FABP	0.8907±0.0071	0.0005±0.0001	0.9382±0.0072
YiiP	0.8425±0.0161	0.0003±0.0001	0.8737±0.0164	FRODA	0.8318±0.0286	0.0004±0.0001	0.9098±0.0175
NahA	0.8265±0.0202	0.0005±0.0001	0.8372±0.0245	adk_equi	0.7526±0.0396	0.0020±0.0009	0.7914±0.0308
protG	0.6580±0.0695	0.0036±0.0017	0.8287±0.0595	ntl9	0.6342±0.1244	0.0066±0.0051	0.8165±0.0911
bba	0.6163±0.1481	0.0057±0.0073	0.8158±0.1044	cspa	0.6136±0.0682	0.0074±0.0036	0.8047±0.0527
T0765	0.5997±0.0923	0.0039±0.0022	0.7991±0.0757	ww	0.5920±0.1056	0.0088±0.0055	0.8196±0.0691
T0855	0.5585±0.0898	0.0117±0.0095	0.7804±0.0647	T0773	0.5289±0.0833	0.0123±0.0091	0.7404±0.0632
T0816	0.5284±0.1366	0.0083±0.0071	0.7368±0.0979	gpW	0.4836±0.1253	0.0093±0.0072	0.7176±0.0905
bbl	0.4603±0.1422	0.0094±0.0067	0.6968±0.0896	hyp	0.4578±0.1154	0.0099±0.0066	0.7104±0.0873
T0769	0.4535±0.0871	0.0191±0.0101	0.7102±0.0738	T0771	0.4219±0.0711	0.0111±0.0073	0.6537±0.0491

We analyze the effect of RMSD on model training quality, indicating that data containing high levels of noise and significant vibration pose challenges to model training. As illustrated in the Figure 5, the model shows superior performance on smoother and less noisy trajectory data.

4.3 Generalization Ability

Our model has generative capabilities to predict missing or future trajectories of a protein. Here we obtain different samples in two aspects, interpolation and extrapolation of time.

4.3.1 Interpolation

Our model enable the prediction of protein shape at any given time due to its temporal continuity. This is very useful when we only know the structures of two moments but not what the intermediate process is. During training sample selection, we extract one frame at intervals of several frames to evaluate the model's interpolation capability in the temporal space. To demonstrate that our model

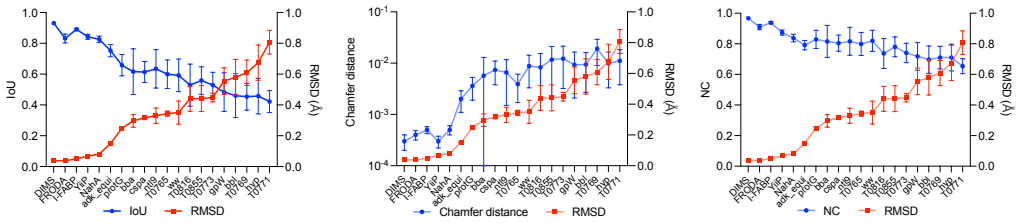

Figure 5: The correlation between RMSD and three metrics was evaluated across various protein trajectories, revealing a significant relationship between the metrics and RMSD, highlighting that smaller RMSD and system noise levels correspond to improved model performance and easier modeling of protein surface dynamics.

learns the protein's inherent dynamic variations and is not just performing trivial tasks, we compared it with linear interpolation, as shown in Figure 6, even though there were no previous works for direct comparison. The evaluation results of our model and linear interpolation are shown in Appendix Table 4, 5 respectively, and the performance of interpolation has reached a level comparable to that of reconstruction.

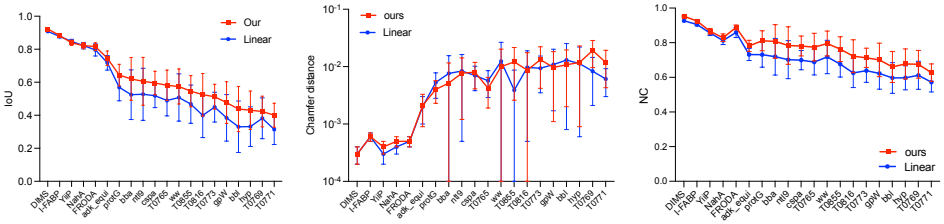

Figure 6: The three evaluation metrics for the interpolation task are shown in the plot, with the red line representing our method and the blue line representing linear interpolation.

4.3.2 Future Prediction

Our model can not only interpolate missing frames in the middle, but also predict future dynamics by extrapolating over time. To validate this, we use models trained on the MDAnalysisData and DRYAD_MD datasets to predict future frames. Table 2 shows the prediction results of the model for the next 10 frames and the next $11 \sim 20$ frames on some proteins. Due to space limitations of the main text, the full results are presented in the Appendix E.2.

Table 2: Evaluation on the future 10 frames and $11 \sim 20$ frames

	Future 10 frames			Future 11∼20		
	IoU↑	Chamfer_dist↓	NC↑	IoU↑	Chamfer_dist↓	NC↑
DIMS	0.8467±0.0423	0.0010±0.0005	0.8666±0.0504	0.6835±0.0551	0.0055±0.0020	0.6932±0.0474
I-FABP	0.8878±0.0142	0.0006±0.0001	0.9271±0.0159	0.8444±0.0153	0.0012±0.0002	0.8797±0.0164
FRODA	0.7660±0.0502	0.0012±0.0006	0.8267±0.0490	0.6043±0.0474	0.0044±0.0012	0.6783±0.0384

4.3.3 A Case Study on GPCRmd

Furthermore, we have explored the model's simultaneous simulation of multiple protein trajectories. Here we use the four protein trajectories data in GPCRmd dataset. The model trains different latent vectors to represent different proteins. In addition, in selecting training frames, we have employed another strategy, wherein the interval between the frames utilized for training is gradually increased with the sequence.

As shown in Figure 7, our model can learn the shapes of multiple proteins simultaneously, and it can recover the fine details of the protein surfaces well despite their ruggedness. An interesting finding is that the trajectories obtained from molecular dynamics simulations often have a lot of random jittering or noise, while the trajectories interpolated by our model can well fit the main functional

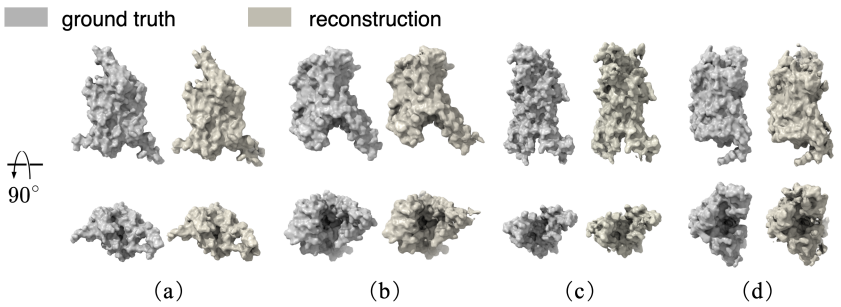

Figure 7: Comparison of ground truth and reconstruction for GPCRmd data at $t = 2000$. From (a) to (d) are 10792_trj_81, 10912_trj_95, 15711_trj_791 and 16105_trj_848 respectively, ground truth and reconstruction (top row) after rotating 90 degrees (bottom row) clockwise along the horizontal axis.

motion of the protein without noise. We put the detailed results and discussion in the Appendix E.3, video demonstration on the webpage.

5 Discussions

Molecular surface Representation The molecular surface representation is commonly adopted for tasks involving molecular interfaces[42], where non-covalent interactions (e.g., hydrophobic interactions) play a decisive role[43]. Non-Euclidean convolutional neural networks[44] and point cloud-based learning models[45] have been applied to encode the molecular surface for downstream applications, e.g., protein binding site prediction[46]. However, it has not been applied to molecular dynamic simulation, especially protein. In fact, this work explored the more essential application scenarios of molecular surface representation, that is, we believe that chemical molecules are more suitable to be regarded as electron clouds that can be represented by 3D shapes.

Deep learning molecular dynamics Deep learning is being used more in molecular dynamics simulations, which has shown promising results. There have been several approaches, such as Behler-Parrinello network[47], DTNN[48], and SchNet[49], that focus on predicting molecular properties and potential energy. However, these methods are limited to small molecules and may not work for proteins, which have many atoms. To address this issue, several coarse-grained models have been proposed, such as CGnet[17], DeePCG[50], ARCG[51], and a CG auto-encoder[52]. Recent works, flow-matching[53] and DFF[54], have used normalizing flows and diffusion models to model coarse-grained force fields for dynamic simulation of small proteins. Another approach, DiffMD[55], can predict simulation trajectories without energy or forces. However, these methods are currently limited to simulating small proteins due to computational constraints and are not yet practical for real-world applications, while the method in this paper provides a new idea for long-term modeling of large proteins.

Implicit neural representations SIREN[56] and NeRF[26] have led to the popularity of implicit neural representations (INR) for tasks like new view synthesis and geometry reconstruction. INR involves creating a continuous function that can map coordinates to high-frequency information. NeRF's highly expressive nature has allowed it to be effectively employed in numerous fields, such as image processing[57] (e.g., compression, denoising, super-resolution, inpainting), video processing[58], medical imaging[59], etc. Signed distance function (SDF) is a type of INR that characterizes an object's shape well. DeepSDF[60] uses INR to model SDF, and IGR[40] simplifies the modeling of SDF by using its gradient property. Drawing inspiration from the aforementioned studies, our work learns a unified implicit representation of proteins that will facilitate future large-scale modeling of protein dynamics like NeRF's contribution in vision.

6 Conclusions and Future work

We introduced a method for learning implicit neural representations of protein surface dynamics in continuous spatial and temporal space. Experimental results show that our method is very effective in reconstructing proteins with smooth dynamic changes, and has a certain ability to interpolate in time.

In addition, we found that our model tends to model major conformation changes of proteins rather than those noise-like random vibrations. The case study of GPCRmd shows that the model has the potential to learn the dynamics of a class of similar proteins, such as a protein family.

The main limitation of this method is that it is sensitive to noise in protein dynamic trajectories, as discussed in Section 4.2. In addition, the generalization of the model needs to be further improved. The first is the temporal generalization, especially future prediction. So far, there is no effective method that can better predict the future in various motion prediction tasks. We believe that modeling motion or velocity fields can help future predictions for a longer time.

We anticipate DSR as a powerful tool for building general-purpose protein surface models, and hope that our work helps shed some light on a more efficient and generalizable protein representation.

Acknowledgments

This work was supported by Mathematical Intelligence Application Laboratory, MIALAB, Institute for Mathematical Sciences, Renmin University of China, Beijing Academy of Artificial Intelligence and Public Computing Cloud of Renmin University of China.

References

[1] Yao Li, Tong Wang, Juanrong Zhang, Bin Shao, Haipeng Gong, Yusong Wang, Xinheng He, Siyuan Liu, and Tie-Yan Liu. Exploring the regulatory function of the n-terminal domain of sars-cov-2 spike protein through molecular dynamics simulation. *Advanced theory and simulations*, 4(10):2100152, 2021.

[2] Maxwell I Zimmerman, Justin R Porter, Michael D Ward, Sukrit Singh, Neha Vithani, Artur Meller, Upasana L Mallimadugula, Catherine E Kuhn, Jonathan H Borowsky, Rafal P Wiewiora, et al. Citizen scientists create an exascale computer to combat covid-19. *BioRxiv*, pages 2020–06, 2020.

[3] Richard Car and Mark Parrinello. Unified approach for molecular dynamics and density-functional theory. *Physical review letters*, 55(22):2471, 1985.

[4] Kresse Georg. Vasp group, theoretical physics departments, vienna. *Retrieved February*, 21, 2011.

[5] Jussi Enkovaara, Nichols A Romero, Sameer Shende, and Jens J Mortensen. Gpaw-massively parallel electronic structure calculations with python-based software. *Procedia Computer Science*, 4:17–25, 2011.

[6] Wendy D Cornell, Piotr Cieplak, Christopher I Bayly, Ian R Gould, Kenneth M Merz, David M Ferguson, David C Spellmeyer, Thomas Fox, James W Caldwell, and Peter A Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society*, 117(19):5179–5197, 1995.

[7] Lijiang Yang, Chun-hu Tan, Meng-Juei Hsieh, Junmei Wang, Yong Duan, Piotr Cieplak, James Caldwell, Peter A Kollman, and Ray Luo. New-generation amber united-atom force field. *The journal of physical chemistry B*, 110(26):13166–13176, 2006.

[8] Sandeep Patel and Charles L Brooks III. Charmm fluctuating charge force field for proteins: I parameterization and application to bulk organic liquid simulations. *Journal of computational chemistry*, 25(1):1–16, 2004.

[9] Sandeep Patel, Alexander D Mackerell Jr, and Charles L Brooks III. Charmm fluctuating charge force field for proteins: Ii protein/solvent properties from molecular dynamics simulations using a nonadditive electrostatic model. *Journal of computational chemistry*, 25(12):1504–1514, 2004.

[10] Wilfred F van Gunsteren and Herman JC Berendsen. Groningen molecular simulation (gromos) library manual. *Biomos, Groningen*, 24(682704):13, 1987.

[11] Lukas D Schuler, Xavier Daura, and Wilfred F Van Gunsteren. An improved gromos96 force field for aliphatic hydrocarbons in the condensed phase. *Journal of computational chemistry*, 22(11):1205–1218, 2001.

[12] Markus Christen, Philippe H Hünenberger, Dirk Bakowies, Riccardo Baron, Roland Bürgi, Daan P Geerke, Tim N Heinz, Mika A Kastenholz, Vincent Kräutler, Chris Oostenbrink, et al. The gromos software for biomolecular simulation: Gromos05. *Journal of computational chemistry*, 26(16):1719–1751, 2005.

[13] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017.

[14] Stefan Chmiela, Huziel E Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nature communications*, 9(1):3887, 2018.

[15] Huziel E Sauceda, Stefan Chmiela, Igor Poltavsky, Klaus-Robert Müller, and Alexandre Tkatchenko. Molecular force fields with gradient-domain machine learning: Construction and application to dynamics of small molecules with coupled cluster forces. *The Journal of chemical physics*, 150(11):114102, 2019.

[16] Stefan Chmiela, Huziel E Sauceda, Igor Poltavsky, Klaus-Robert Müller, and Alexandre Tkatchenko. sgdml: Constructing accurate and data efficient molecular force fields using machine learning. *Computer Physics Communications*, 240:38–45, 2019.

[17] Jiang Wang, Simon Olsson, Christoph Wehmeyer, Adrià Pérez, Nicholas E Charron, Gianni De Fabritiis, Frank Noé, and Cecilia Clementi. Machine learning of coarse-grained molecular dynamics force fields. *ACS central science*, 5(5):755–767, 2019.

[18] Feliks Nüske, Lorenzo Boninsegna, and Cecilia Clementi. Coarse-graining molecular systems by spectral matching. *The Journal of chemical physics*, 151(4):044116, 2019.

[19] Jiang Wang, Stefan Chmiela, Klaus-Robert Müller, Frank Noé, and Cecilia Clementi. Ensemble learning of coarse-grained molecular dynamics force fields with a kernel approach. *The Journal of chemical physics*, 152(19):194106, 2020.

[20] Xiang Fu, Tian Xie, Nathan J Rebello, Bradley D Olsen, and Tommi Jaakkola. Simulate time-integrated coarse-grained molecular dynamics with geometric machine learning. *arXiv preprint arXiv:2204.10348*, 2022.

[21] David E Shaw, Peter J Adams, Asaph Azaria, Joseph A Bank, Brannon Batson, Alistair Bell, Michael Bergdorf, Jhanvi Bhatt, J Adam Butts, Timothy Correia, et al. Anton 3: twenty microseconds of molecular dynamics simulation before lunch. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2021.

[22] Ye Li, Xianren Zhang, and Dapeng Cao. The role of shape complementarity in the protein-protein interactions. *Scientific reports*, 3(1):3271, 2013.

[23] Pablo Gainza, Freyr Sverrisson, Frederico Monti, Emanuele Rodola, D Boscaini, MM Bronstein, and BE Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.

[24] Pablo Gainza, Sarah Wehrle, Alexandra Van Hall-Beauvais, Anthony Marchand, Andreas Scheck, Zander Harteveld, Stephen Buckley, Dongchun Ni, Shuguang Tan, Freyr Sverrisson, et al. De novo design of site-specific protein interactions with learned surface fingerprints. *bioRxiv*, pages 2022–06, 2022.

[25] Freyr Sverrisson, Jean Feydy, Bruno E Correia, and Michael M Bronstein. Fast end-to-end learning on protein surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15272–15281, 2021.

[26] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[27] Chengan He, Jun Saito, James Zachary, Holly Rushmeier, and Yi Zhou. Nemf: Neural motion fields for kinematic animation. *arXiv preprint arXiv:2206.03287*, 2022.

[28] Carter Butts. 500ns Trajectory Data for "Comparative Exploratory Analysis of Intrinsically Disordered Protein Dynamics Using Machine Learning and Network Analytic Methods," (Grazioli.et.al. 2019), 2022. URL `https://doi.org/10.7910/DVN/ERYOZS`.

[29] Gianmarc Grazioli, Rachel W Martin, and Carter T Butts. Comparative exploratory analysis of intrinsically disordered protein dynamics using machine learning and network analytic methods. *Frontiers in molecular biosciences*, 6:42, 2019.

[30] Sean Seyler and Oliver Beckstein. Molecular dynamics trajectory for benchmarking mdanalysis, 6 2017. *URL: https://figshare. com/articles/Molecular_dynamics_ trajectory_for_benchmarking_MDAnalysis/5108170, doi*, 10:m9.

[31] Sean L Seyler, Avishek Kumar, Michael F Thorpe, and Oliver Beckstein. Path similarity analysis: a method for quantifying macromolecular pathways. *PLoS computational biology*, 11 (10):e1004568, 2015.

[32] Juan R Perilla, Oliver Beckstein, Elizabeth J Denning, and Thomas B Woolf. Computing ensembles of transitions from stable states: Dynamic importance sampling. *Journal of computational chemistry*, 32(2):196–209, 2011.

[33] Daniel W Farrell, Kirill Speranskiy, and MF Thorpe. Generating stereochemically acceptable protein pathways. *Proteins: Structure, Function, and Bioinformatics*, 78(14):2908–2921, 2010.

[34] Oliver Beckstein. Molecular dynamics trajectory of i-fabp for testing and benchmarking solvent dynamics analysis, 2018.

[35] Ian M. Kenney, Shujie Fan, and Oliver Beckstein. Molecular dynamics trajectory of membrane protein nhaa, 2018. URL `https://doi.org/10.6084/m9.figshare.7185203.v2`.

[36] Shujie Fan and Oliver Beckstein. Molecular dynamics trajectories of membrane protein yiip, 2019. URL `https://doi.org/10.6084/m9.figshare.8202149.v1`.

[37] Ismael Rodríguez-Espigares, Mariona Torrens-Fontanals, Johanna KS Tiemann, David Aranda-García, Juan Manuel Ramírez-Anguita, Tomasz Maciej Stepniewski, Nathalie Worp, Alejandro Varela-Rial, Adrián Morales-Pastor, Brian Medel-Lacruz, et al. Gpcrmd uncovers the dynamics of the 3d-gpcrome. *Nature Methods*, 17(8):777–787, 2020.

[38] John M Jumper, Nabil F Faruk, Karl F Freed, and Tobin R Sosnick. Data from: Trajectory-based training enables protein simulations with accurate folding and boltzmann ensembles in cpu-hours, 2018. URL `https://doi.org/10.5061/dryad.h9f8sb7`.

[39] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[40] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[41] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[42] Dina Duhovny, Ruth Nussinov, and Haim J Wolfson. Efficient unbound docking of rigid molecules. In *Algorithms in Bioinformatics: Second International Workshop, WABI 2002 Rome, Italy, September 17–21, 2002 Proceedings 2*, pages 185–200. Springer, 2002.

[43] Kim A Sharp. Electrostatic interactions in macromolecules. *Current Opinion in Structural Biology*, 4(2):234–239, 1994.

[44] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[45] Freyr Sverrisson, Jean Feydy, Joshua Southern, Michael M Bronstein, and Bruno E Correia. Physics-informed deep neural network for rigid-body protein docking. In *MLDD workshop of ICLR 2022*, 2022.

[46] Stelios K Mylonas, Apostolos Axenopoulos, and Petros Daras. Deepsurf: a surface-based deep learning approach for the prediction of ligand binding sites on proteins. *Bioinformatics*, 37(12): 1681–1690, 2021.

[47] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

[48] Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):13890, 2017.

[49] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[50] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deepcg: Constructing coarse-grained models via deep neural networks. *The Journal of chemical physics*, 149(3): 034101, 2018.

[51] Aleksander EP Durumeric and Gregory A Voth. Adversarial-residual-coarse-graining: Applying machine learning theory to systematic molecular coarse-graining. *The Journal of chemical physics*, 151(12):124110, 2019.

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

[53] Jonas Kohler, Yaoyi Chen, Andreas Kramer, Cecilia Clementi, and Frank Noé. Flow-matching: Efficient coarse-graining of molecular dynamics without forces. *Journal of Chemical Theory and Computation*, 19(3):942–952, 2023.

[54] Marloes Arts, Victor Garcia Satorras, Chin-Wei Huang, Daniel Zuegner, Marco Federici, Cecilia Clementi, Frank Noé, Robert Pinsler, and Rianne van den Berg. Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. *arXiv preprint arXiv:2302.00600*, 2023.

[55] Fang Wu13 and Stan Z Li. Diffmd: A geometric diffusion model for molecular dynamics simulations. 2023.

[56] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[57] Mikolaj Czerkawski, Javier Cardona, Robert Atkinson, Craig Michie, Ivan Andonovic, Carmine Clemente, and Christos Tachtatzis. Neural knitworks: Patched neural implicit representation networks. *arXiv preprint arXiv:2109.14406*, 2021.

[58] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021.

[59] Liyue Shen, John Pauly, and Lei Xing. Nerp: implicit neural representation learning with prior embedding for sparsely sampled image reconstruction. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[60] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[61] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[62] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Appendix

A Motivations

The traditional methods for molecular dynamics simulation, such as solving the Schrödinger equation, the Born-Oppenheimer approximation method, and Density Functional Theory (DFT), are computationally expensive and time-consuming. To provide a more detailed explanation, we can give specific numbers to illustrate the computational complexity of these methods. For instance, solving the Schrödinger equation directly for a system of 100 atoms would require solving a matrix with approximately 1.26×10^6 elements, which is computationally intractable even with supercomputers. The Born-Oppenheimer approximation, which separates electronic and nuclear motion, is also computationally expensive. For example, a single energy minimization calculation using this approximation for a protein containing 10,000 atoms can take several hours on a high-performance computing cluster. DFT, which is a widely used approximation for solving the Schrödinger equation, has a time complexity of $O(n^3)$, where n is the number of atoms in the system. For example, a DFT calculation for a system of 1000 atoms would require approximately 1 billion calculations.

Since traditional biochemists tend to focus on atomic representations of proteins, the computational complexity is closely related to the number of atoms in the system. However, we have converted the representation to a continuous function of the protein surface, which can significantly reduce the computational complexity.

B Theory Analysis

Theorem 1. *Given a manifold Ω, the implicit signed distance function is defined as $\phi(\boldsymbol{x})$, $x \in \mathbb{R}^n$ with $\phi(\boldsymbol{x}) < 0$ in the interior regin Ω^-, $\phi(\boldsymbol{x}) > 0$ in the exterior regin Ω^+, and $\phi(\boldsymbol{x}) = 0$ on the boundary $\partial\Omega$, then $|\nabla\phi(\boldsymbol{x})| = 1$.*

$$\phi(\boldsymbol{x}) = \begin{cases} d(\boldsymbol{x}, \Omega) & \text{if } \boldsymbol{x} \in \Omega^+ \\ 0 & \text{if } \boldsymbol{x} \in \Omega \\ -d(\boldsymbol{x}, \Omega) & \text{if } x \in \Omega^- \end{cases} \tag{10}$$

where $d(x, \Omega) := \min(\|\boldsymbol{x} - \boldsymbol{x}_I\|)$ for all $\boldsymbol{x}_I \in \partial\Omega$.

Proof. Firstly, for a given point $\boldsymbol{x}$, suppose that $\boldsymbol{x}_c$ is the only point on the surface closest to $\boldsymbol{x}$. Then for every point $\boldsymbol{y}$ on the line segment between points $\boldsymbol{x}$ and $\boldsymbol{x}_c$, the point on the surface closest to $\boldsymbol{y}$ is still $\boldsymbol{x}_c$.

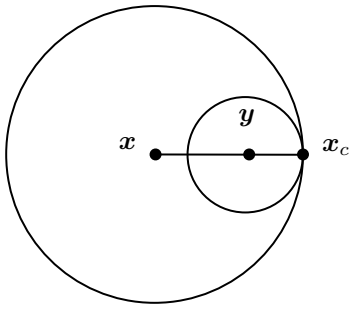

Figure 8: $\boldsymbol{x}_C$ is the closest surface point to $\boldsymbol{x}$ and $\boldsymbol{y}$.

As shown in Figure 8, inside a sphere with $\boldsymbol{x}\boldsymbol{x}_c$ as radius, there will be no point on the surface closer to $\boldsymbol{x}$. Similarly, within the sphere with radius $\boldsymbol{y}\boldsymbol{x}_c$, there will not be any point on the surface that is closer to point $\boldsymbol{y}$. Otherwise, the distance between the point and $\boldsymbol{x}$ would be less than the distance between $\boldsymbol{x}\boldsymbol{x}_c$, which contradicts the assumption. Therefore, it can be seen that the line segment $\boldsymbol{x}\boldsymbol{x}_c$ is the shortest path from point $\boldsymbol{x}$ to the surface, and it is also the direction in which $\phi(\boldsymbol{x})$ changes the fastest (i.e., for $x \in \Omega^-$, $\phi(\boldsymbol{x})$ increases the fastest, and for $x \in \Omega^+$, $\phi(\boldsymbol{x})$ decreases the fastest).

Then,

$$
\begin{aligned}
|\nabla\phi(\boldsymbol{x})| &= \left|\frac{\partial\phi(\boldsymbol{x})}{\partial\left(\overrightarrow{\boldsymbol{x}\boldsymbol{x}_{\boldsymbol{c}}}\right)}\right| \\
&= \left|\lim_{\substack{\boldsymbol{y}\to\boldsymbol{x} \\ \boldsymbol{y}\in\boldsymbol{x}\boldsymbol{x}_c}} \frac{\phi(\boldsymbol{y})-\phi(\boldsymbol{x})}{d(\boldsymbol{y},\boldsymbol{x})}\right| \\
&= \left|\lim_{\substack{\boldsymbol{y}\to\boldsymbol{x} \\ \boldsymbol{y}\in\boldsymbol{x}\boldsymbol{x}_c}} \frac{d\left(\boldsymbol{y},\boldsymbol{x}_{\boldsymbol{c}}\right)-d\left(\boldsymbol{x},\boldsymbol{x}_{\boldsymbol{c}}\right)}{d(\boldsymbol{y},\boldsymbol{x})}\right| \\
&= \left|\lim_{\substack{\boldsymbol{y}\to\boldsymbol{x} \\ \boldsymbol{y}\in\boldsymbol{x}\boldsymbol{x}_c}} \frac{d(\boldsymbol{y},\boldsymbol{x})}{d(\boldsymbol{y},\boldsymbol{x})}\right| = 1
\end{aligned}
\tag{11}
$$

Secondly, if there are more than one point on the surface that is closest to x, let these points be $\{\boldsymbol{x}_{c1},\boldsymbol{x}_{c2},\cdots,\boldsymbol{x}_{cN}\}$. Similarly, within the sphere with radius $\boldsymbol{x}\boldsymbol{x}_{ci}, i\in 1,2,\cdots,N$, there will not be any point on the surface closer to $\boldsymbol{x}$ than $\{\boldsymbol{x}_{c1},\boldsymbol{x}_{c2},\cdots,\boldsymbol{x}_{cN}\}$. Therefore, gradient direction should be the direction with the maximum numerical value among the directional derivatives in $\{\boldsymbol{x}\boldsymbol{x}_{c1},\boldsymbol{x}\boldsymbol{x}_{c2},\cdots,\boldsymbol{x}\boldsymbol{x}_{cN}\}$, i.e.

$$
|\nabla\phi(\boldsymbol{x})| = \max_i \left|\frac{\partial\phi(\boldsymbol{x})}{\partial\left(\overrightarrow{\boldsymbol{x}\boldsymbol{x}_{\boldsymbol{ci}}}\right)}\right| = \max_i \left|\lim_{\substack{\boldsymbol{y}\to\boldsymbol{x} \\ \boldsymbol{y}\in\boldsymbol{x}\boldsymbol{x}_{ci}}} \frac{\phi(\boldsymbol{y})-\phi(\boldsymbol{x})}{d(\boldsymbol{y},\boldsymbol{x})}\right| = 1 \tag{12}
$$

□

C Experiment Details

For each protein surface representation trajectory used for training, we scale the space to $[-1,1]^3$ and the time to $[-1,1]$. When reconstructing, interpolating, and extrapolating, for time, we scale at the same scale as training, and for space, we divide the $[-1,1]^3$ space into grid points according to a given resolution. We use the Marching Cube algorithm to extract the zero-level set of the SDF value output by the model, and scale it back to the original space size.

The training was done on a single Nvidia-V100 GPU with PyTorch deep learning framework[61]. For training, we use the Adam optimizer[62], where initial learning rate of sdf training is 5e-3 with a decay of 0.5 every 200 epochs and initial learning rate of latent code is 1e-3 with a decay of 0.5 every 200 epochs. We trained each model for 4000 epochs. Because our model is continuous in time and space, we can reconstruct or interpolate or extrapolate protein shapes at any time and at any resolution.

For experiments of GPCRmd, we use different latent code to specify different sequences. When doing inference, you can use different latent codes to perform tasks corresponding to the sequence by specifying the sequence id.

Table 3: The architecture of our neural network.

Layer	Input shape	Output shape	
Dense layer	196	512	$(x,t)\in\mathbb{R}^4$, latent code $\in\mathbb{R}^{192}$
Dense layer	512	512	
Dense layer	512	512	
Dense layer	512	316	skip connection from the first layer
Dense layer	512	512	
Dense layer	512	512	
Dense layer	512	512	
Dense layer	512	1	
Activation	Softplus(beta=100, threshold=20)		

D Learning Ability

In order to illustrate that our method has the ability to learn the shape of objects, we visualize the training process in Figure9. As can be seen from the Figure9, with the training process, the model can reconstruct the shape of the protein in more detail. In addition, using normal vectors as constraints in loss function during training will make the model achieve better results faster.

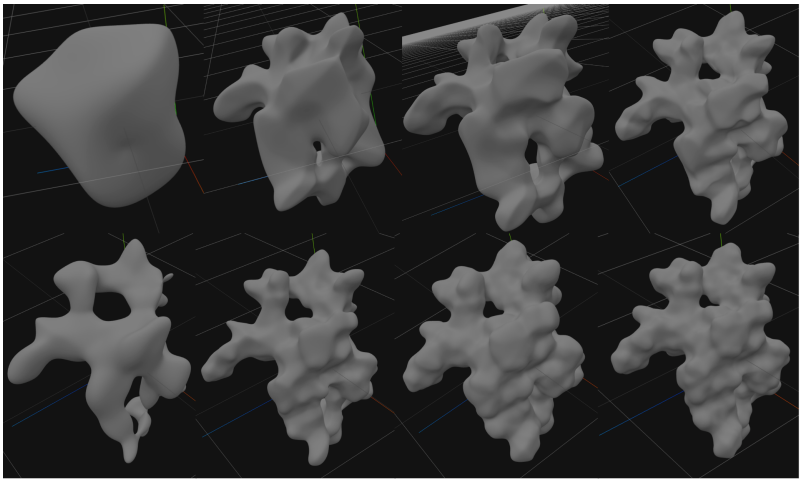

Figure 9: The shape visualization of Abeta. First row: the result of training epoch 100, 500, 1000 and 2000 epochs respectively without normal vector; second row: the result of training epoch 100, 300, 500 and 1000 epochs respectively with normal vector.

E Detailed results

E.1 Interpolation

Table 4: Evaluation on the surface interpolation between trained frames by our model.

	IoU↑	Chamfer_dist↓	NC↑		IoU↑	Chamfer_dist↓	NC↑
DIMS	0.9225±0.0035	0.0003±0.0001	0.9527±0.0034	I-FABP	0.8833±0.0078	0.0006±0.0001	0.9254±0.0066
YiiP	0.8387±0.0157	0.0004±0.0001	0.8659±0.0157	FRODA	0.8159±0.0234	0.0005±0.0001	0.8885±0.0135
NahA	0.8220±0.0191	0.0005±0.0001	0.8287±0.0219	adk_equi	0.7459±0.0436	0.0021±0.0012	0.7816±0.0327
protG	0.6434±0.0655	0.0040±0.0017	0.8113±0.0580	ntl9	0.6059±0.1452	0.0077±0.0065	0.7838±0.1083
bba	0.6204±0.1296	0.0052±0.0064	0.8074±0.0972	cspa	0.5957±0.0714	0.0080±0.0039	0.7812±0.0574
T0765	0.5821±0.0921	0.0042±0.0023	0.7747±0.0797	ww	0.5744±0.1061	0.0100±0.0102	0.7955±0.0720
T0855	0.5458±0.0943	0.0121±0.0095	0.7629±0.0716	T0773	0.5127±0.0755	0.0132±0.0090	0.7130±0.0559
T0816	0.5245±0.1289	0.0087±0.0088	0.7221±0.0951	gpW	0.4769±0.1262	0.0097±0.0086	0.7008±0.0967
bbl	0.4380±0.1372	0.0108±0.0088	0.6605±0.0895	hyp	0.4304±0.1163	0.0119±0.0110	0.6776±0.0865
T0769	0.4229±0.0932	0.0191±0.0095	0.6752±0.0801	T0771	0.4006±0.0729	0.0118±0.0075	0.6271±0.0503

Table 5: Evaluation on the surface interpolation between trained frames by linear interpolation.

	IoU↑	Chamfer_dist↓	NC↑		IoU↑	Chamfer_dist↓	NC↑
DIMS	0.9084±0.0071	0.0003±0.0001	0.9266±0.0079	I-FABP	0.8770±0.0076	0.0006±0.0001	0.9053±0.0082
YiiP	0.8475±0.0126	0.0003±0.0001	0.8560±0.0143	FRODA	0.7961±0.0368	0.0005±0.0001	0.8593±0.0292
NahA	0.8219±0.0192	0.0004±0.0001	0.8129±0.0237	adk_equi	0.7181±0.0440	0.0020±0.0010	0.7320±0.0352
protG	0.5694±0.0820	0.0053±0.0025	0.7309±0.0719	ntl9	0.5273±0.1578	0.0084±0.0079	0.7021±0.1100
bba	0.5249±0.1506	0.0076±0.0079	0.7193±0.1056	cspa	0.5189±0.0728	0.0070±0.0032	0.6994±0.0556
T0765	0.4899±0.0937	0.0057±0.0028	0.6885±0.0745	ww	0.5079±0.1433	0.0124±0.0142	0.7177±0.0965
T0855	0.4674±0.1156	0.0039±0.0048	0.6762±0.0746	T0773	0.4496±0.0904	0.0094±0.0059	0.6389±0.0672
T0816	0.3994±0.1344	0.0097±0.0092	0.6259±0.0845	gpW	0.3846±0.1411	0.0109±0.0092	0.6228±0.0912
bbl	0.3304±0.1558	0.0130±0.0122	0.5964±0.0881	hyp	0.3313±0.1201	0.0113±0.0107	0.5974±0.0691
T0769	0.3820±0.1236	0.0083±0.0062	0.6122±0.0808	T0771	0.3147±0.0914	0.0061±0.0031	0.5723±0.0564

E.2 Future Prediction

In this section we list the three evaluation metrics for the prediction results of all other proteins in the next 100 frames. Tables 6, 7, and 8 show the results of IoU, chamfer distance, and NC respectively. As shown in the table, the results decrease slightly as time goes into the future, which is not surprising. It is worth noting that even when predicting the next 100 frames of some proteins, the prediction results do not drop much, such as the Yiip, NahA and adk_equi, which also shows that our model has certain expression power and stability.

Table 6: The future prediction results on IoU

Future frames	10	11∼20	21∼30	31∼40	41∼50
YiiP	0.8482±0.0041	0.8293±0.0067	0.8068±0.0087	0.8128±0.0091	0.8122±0.0090
NahA	0.7922±0.0137	0.7914±0.0133	0.7640±0.0141	0.7589±0.0071	0.7465±0.0092
adk_equi	0.7417±0.0289	0.7304±0.0285	0.6582±0.0680	0.7107±0.0385	0.7080±0.0589
protG	0.6666±0.0579	0.6366±0.0641	0.6390±0.0371	0.6257±0.0611	0.6203±0.0614
ntl9	0.5829±0.2385	0.6712±0.1451	0.6997±0.0593	0.6069±0.1673	0.6092±0.1806
bba	0.4958±0.0546	0.5088±0.0432	0.5334±0.0261	0.4973±0.0315	0.4628±0.1062
cspa	0.5467±0.0935	0.5237±0.0624	0.4914±0.0479	0.5332±0.0494	0.4872±0.0744
T0765	0.4820±0.0941	0.4638±0.0877	0.4684±0.0628	0.4544±0.0476	0.4241±0.0442
ww	0.3716±0.1133	0.3163±0.0826	0.3176±0.0620	0.2753±0.0162	0.2729±0.0234
T0855	0.4215±0.0368	0.3631±0.0374	0.3221±0.0286	0.2899±0.0308	0.2355±0.0398
T0773	0.4364±0.0982	0.4011±0.0848	0.4272±0.0880	0.4169±0.0839	0.3802±0.0727
T0816	0.5301±0.0805	0.5502±0.0557	0.5187±0.0574	0.4652±0.1149	0.4366±0.1391
gpW	0.4604±0.1025	0.4644±0.1483	0.4785±0.1504	0.5072±0.0857	0.4492±0.0957
bbl	0.2557±0.2497	0.3817±0.2172	0.4064±0.2026	0.1792±0.0579	0.1916±0.0453
hyp	0.3189±0.0746	0.3185±0.0751	0.2889±0.0693	0.2498±0.0490	0.2650±0.0370
T0769	0.4256±0.0706	0.4216±0.0867	0.3700±0.0666	0.4157±0.0918	0.4313±0.0668
T0771	0.4031±0.0798	0.3738±0.0724	0.3543±0.0717	0.3596±0.0668	0.3538±0.0569
Future frames	51∼60	61∼70	71∼80	81∼90	91∼100
YiiP	0.7986±0.0071	0.7744±0.0116	0.7686±0.0058	0.7698±0.0076	0.7546±0.0066
NahA	0.7637±0.0079	0.7551±0.0046	0.7402±0.0081	0.7438±0.0072	0.7330±0.0097
adk_equi	0.7625±0.0153	0.7814±0.0177	0.7635±0.0170	0.7402±0.0273	0.7144±0.0343
protG	0.6083±0.0342	0.6159±0.0276	0.6216±0.0477	0.6165±0.0466	0.5892±0.0644
ntl9	0.5884±0.1663	0.5821±0.1629	0.5614±0.1328	0.6173±0.1268	0.6306±0.0274
bba	0.4950±0.0140	0.4606±0.0734	0.4469±0.0495	0.4138±0.0639	0.4439±0.0159
cspa	0.4631±0.0683	0.4704±0.0308	0.4312±0.0597	0.4157±0.0689	0.3389±0.0467
T0765	0.4523±0.0361	0.4199±0.0318	0.4179±0.0315	0.3732±0.0292	0.3463±0.0396
ww	0.2549±0.0273	0.2364±0.0279	0.2225±0.0154	0.2175±0.0294	0.2076±0.0366
T0855	0.2401±0.0407	0.1940±0.0303	0.1956±0.0155	0.1609±0.0178	0.1460±0.0244
T0773	0.4414±0.0686	0.3944±0.0777	0.3706±0.0788	0.3455±0.0454	0.3166±0.0478
T0816	0.4738±0.0916	0.4706±0.1068	0.4301±0.1223	0.4740±0.0240	0.4167±0.0849
gpW	0.3396±0.1590	0.4465±0.1135	0.4157±0.0684	0.4349±0.1060	0.4121±0.0894
bbl	0.2384±0.0626	0.2008±0.0631	0.1802±0.0676	0.1861±0.0450	0.1819±0.0530
hyp	0.2171±0.0733	0.2445±0.0203	0.2240±0.0347	0.2035±0.0409	0.1915±0.0321
T0769	0.4022±0.0699	0.3642±0.0584	0.3614±0.0507	0.3194±0.0527	0.3391±0.0438
T0771	0.3373±0.0570	0.3053±0.0540	0.3068±0.0359	0.2831±0.0325	0.2603±0.0353

Table 7: The future prediction results on Chamfer distance

Future frames	10	11∼20	21∼30	31∼40	41∼50
YiiP	0.0003±2.1863	0.0004±3.7481	0.0005±5.4920	0.0005±5.3535	0.0005±4.2672
NahA	0.0007±7.7686	0.0007±8.1652	0.0009±8.1229	0.0010±4.5848	0.0011±8.0088
adk_equi	0.0021±0.0006	0.0026±0.0006	0.0050±0.0027	0.0029±0.0009	0.0031±0.0014
protG	0.0036±0.0017	0.0042±0.0017	0.0041±0.0014	0.0046±0.0018	0.0045±0.0017
ntl9	0.0112±0.0150	0.0044±0.0066	0.0023±0.0009	0.0069±0.0086	0.0065±0.0082
bba	0.0088±0.0026	0.0081±0.0016	0.0073±0.0009	0.0087±0.0009	0.0126±0.0105
cspa	0.0130±0.0028	0.0140±0.0022	0.0152±0.0019	0.0134±0.0014	0.0161±0.0044
T0765	0.0071±0.0035	0.0078±0.0029	0.0075±0.0025	0.0078±0.0019	0.0083±0.0020
ww	0.0170±0.0028	0.0196±0.0037	0.0201±0.0021	0.0202±0.0019	0.0209±0.0011
T0855	0.0131±0.0012	0.0142±0.0015	0.0148±0.0023	0.0159±0.0019	0.0186±0.0031
T0773	0.0159±0.0068	0.0179±0.0078	0.0177±0.0106	0.0162±0.0080	0.0160±0.0057
T0816	0.0286±0.0043	0.0310±0.0044	0.0318±0.0044	0.0345±0.0041	0.0340±0.0066
gpW	0.0104±0.0021	0.0131±0.0037	0.0117±0.0024	0.0122±0.0033	0.0140±0.0036
bbl	0.0276±0.0203	0.0213±0.0265	0.0155±0.0223	0.0283±0.0222	0.0209±0.0057
hyp	0.0354±0.0025	0.0378±0.0090	0.0387±0.0104	0.0372±0.0050	0.0371±0.0050
T0769	0.0142±0.0028	0.0123±0.0020	0.0104±0.0024	0.0093±0.0019	0.0079±0.0011
T0771	0.0083±0.0025	0.0087±0.0023	0.0079±0.0011	0.0090±0.0012	0.0089±0.0010
Future frames	51∼60	61∼70	71∼80	81∼90	91∼100
YiiP	0.0006±4.0775	0.0007±7.4013	0.0008±3.4454	0.0008±6.6268	0.0009±5.4942
NahA	0.0010±6.2155	0.0011±4.4547	0.0013±7.3425	0.0013±5.2765	0.0014±6.8101
adk_equi	0.0017±0.0002	0.0015±0.0002	0.0018±0.0003	0.0023±0.0005	0.0028±0.0008
protG	0.0050±0.0013	0.0044±0.0009	0.0045±0.0015	0.0044±0.0011	0.0052±0.0019
ntl9	0.0067±0.0073	0.0067±0.0075	0.0065±0.0059	0.0050±0.0063	0.0034±0.0006
bba	0.0093±0.0008	0.0119±0.0052	0.0125±0.0032	0.0149±0.0050	0.0131±0.0011
cspa	0.0171±0.0042	0.0159±0.0022	0.0188±0.0045	0.0177±0.0038	0.0210±0.0024
T0765	0.0072±0.0025	0.0092±0.0023	0.0098±0.0026	0.0112±0.0022	0.0132±0.0036
ww	0.0246±0.0054	0.0256±0.0030	0.0277±0.0047	0.0304±0.0065	0.0326±0.0063
T0855	0.0185±0.0024	0.0244±0.0060	0.0218±0.0015	0.0237±0.0026	0.0249±0.0028
T0773	0.0125±0.0081	0.0162±0.0082	0.0167±0.0061	0.0183±0.0048	0.0202±0.0109
T0816	0.0349±0.0038	0.0355±0.0051	0.0354±0.0052	0.0342±0.0024	0.0349±0.0021
gpW	0.0210±0.0112	0.0156±0.0055	0.0177±0.0025	0.0164±0.0035	0.0177±0.0058
bbl	0.0154±0.0058	0.0255±0.0159	0.0303±0.0172	0.0228±0.0060	0.0399±0.0363
hyp	0.0464±0.0175	0.0390±0.0062	0.0435±0.0070	0.0534±0.0080	0.0541±0.0077
T0769	0.0072±0.0015	0.0105±0.0051	0.0081±0.0011	0.0102±0.0035	0.0106±0.0041
T0771	0.0095±0.0014	0.0095±0.0012	0.0108±0.0024	0.0123±0.0029	0.0132±0.0021

Table 8: The future prediction results on NC

Future frames	10	11∼20	21∼30	31∼40	41∼50
YiiP	0.8752±0.0057	0.8505±0.0067	0.8226±0.0087	0.8269±0.0104	0.8249±0.0117
NahA	0.7863±0.0182	0.7854±0.0146	0.7564±0.0151	0.7513±0.0072	0.7400±0.0118
adk_equi	0.7750±0.0205	0.7723±0.0198	0.7329±0.0441	0.7514±0.0294	0.7529±0.0408
protG	0.8184±0.0514	0.8018±0.0580	0.8063±0.0376	0.7879±0.0653	0.7795±0.0705
ntl9	0.7595±0.1691	0.8099±0.1178	0.8347±0.0429	0.7569±0.1273	0.7618±0.1284
bba	0.7259±0.0680	0.7323±0.0585	0.7703±0.0338	0.7344±0.0587	0.7122±0.0832
cspa	0.7477±0.0714	0.7243±0.0472	0.7040±0.0384	0.7242±0.0370	0.6859±0.0511
T0765	0.6863±0.0664	0.6732±0.0648	0.6730±0.0410	0.6556±0.0360	0.6409±0.0258
ww	0.6313±0.1016	0.5828±0.0755	0.5697±0.0553	0.5362±0.0355	0.5386±0.0339
T0855	0.6896±0.0400	0.6516±0.0394	0.6340±0.0306	0.6104±0.0240	0.5751±0.0377
T0773	0.6588±0.0668	0.6393±0.0581	0.6551±0.0517	0.6422±0.0591	0.6219±0.0396
T0816	0.7513±0.0869	0.7966±0.0635	0.7647±0.0790	0.7393±0.1030	0.7151±0.1231
gpW	0.6905±0.0715	0.6998±0.1042	0.7097±0.1046	0.7173±0.0746	0.6836±0.0724
bbl	0.5752±0.1445	0.6645±0.1352	0.6805±0.1300	0.5444±0.0363	0.5292±0.0266
hyp	0.6558±0.0692	0.6538±0.0722	0.6247±0.0652	0.5840±0.0506	0.6105±0.0493
T0769	0.6662±0.0673	0.6734±0.0623	0.6193±0.0560	0.6527±0.0661	0.6723±0.0489
T0771	0.6136±0.0458	0.5941±0.0373	0.5845±0.0322	0.5843±0.0304	0.5758±0.0266
Future frames	51∼60	61∼70	71∼80	81∼90	91∼100
YiiP	0.8044±0.0084	0.7786±0.0117	0.7733±0.0047	0.7707±0.0100	0.7531±0.0058
NahA	0.7493±0.0103	0.7323±0.0052	0.7195±0.0084	0.7164±0.0094	0.7007±0.0095
adk_equi	0.7889±0.0126	0.8052±0.0162	0.7931±0.0085	0.7735±0.0222	0.7519±0.0262
protG	0.7754±0.0330	0.7838±0.0258	0.7910±0.0283	0.7810±0.0359	0.7608±0.0485
ntl9	0.7378±0.1167	0.7330±0.1219	0.7148±0.0991	0.7658±0.0919	0.7753±0.0177
bba	0.7637±0.0127	0.7291±0.0683	0.7218±0.0528	0.6993±0.0607	0.7187±0.0136
cspa	0.6714±0.0448	0.6660±0.0248	0.6464±0.0425	0.6452±0.0436	0.5939±0.0337
T0765	0.6476±0.0167	0.6218±0.0246	0.6156±0.0297	0.5940±0.0272	0.5774±0.0203
ww	0.5268±0.0294	0.5466±0.0283	0.5425±0.0231	0.5261±0.0206	0.5092±0.0121
T0855	0.5920±0.0276	0.5566±0.0309	0.5737±0.0070	0.5505±0.0204	0.5432±0.0179
T0773	0.6469±0.0345	0.6179±0.0404	0.6003±0.0373	0.5855±0.0301	0.5663±0.0235
T0816	0.7629±0.0908	0.7722±0.0919	0.7296±0.1131	0.7657±0.0317	0.7070±0.0845
gpW	0.6158±0.1008	0.6914±0.0787	0.6581±0.0503	0.6627±0.0693	0.6653±0.0590
bbl	0.5579±0.0543	0.5243±0.0473	0.5101±0.0246	0.5204±0.0239	0.5127±0.0109
hyp	0.5703±0.0607	0.6011±0.0231	0.5737±0.0409	0.5576±0.0250	0.5596±0.0243
T0769	0.6372±0.0511	0.6165±0.0432	0.6206±0.0423	0.6016±0.0355	0.6141±0.0253
T0771	0.5688±0.0278	0.5505±0.0230	0.5495±0.0181	0.5367±0.0195	0.5339±0.0176

E.3 GPCRmd

Table 9 shows the evaluations of the four proteins in GPCRmd on the three tasks of reconstruction, time interpolation and prediction of the future 50 frames. From the table, the results of different proteins are comparable, indicating that the model has a certain multi-protein generalization ability.

Table 9: Evaluation on GPCDmd

	IoU↑	Chamfer_dist↓	NC↑	IoU↑	Chamfer_dist↓	NC↑
		10792_trj_81			10912_trj_95	
Reconstruction	0.8158±0.0224	0.0005±0.0001	0.8668±0.0242	0.8181±0.0242	0.0004±0.0002	0.8674±0.0242
Interpolation	0.7885±0.0173	0.0006±0.0001	0.8118±0.0159	0.7928±0.0180	0.0005±0.0001	0.8127±0.0176
Future 50 frames	0.7234±0.0317	0.0014±0.0005	0.7433±0.0319	0.7027±0.0502	0.0011±0.0003	0.7092±0.0544
		15711_trj_791			16105_trj_848	
Reconstruction	0.7621±0.0508	0.0009±0.0004	0.8217±0.0462	0.7725±0.0516	0.0009±0.0004	0.8327±0.0476
Interpolation	0.7090±0.0556	0.0012±0.0005	0.7082±0.0520	0.7186±0.0623	0.0012±0.0006	0.7216±0.0604
Future 50 frames	0.5654±0.0762	0.0027±0.0011	0.5926±0.0485	0.6076±0.0462	0.0018±0.0007	0.6336±0.0389