Modeling Protein Structure Using Geometric Vector Field Networks*

Weian Mao[1,2], **Muzhi Zhu**[1], **Hao Chen**[1], **Chunhua Shen**[1]

[1] Zhejiang University, China [2] The University of Adelaide, Australia

Abstract

Proteins serve as the foundation of life, and numerous diseases and challenges in the field of life sciences are intimately linked to the molecular dynamics laws that are concealed within protein structures. In this paper, we propose a novel vector field network (VFN) for modeling the protein structure. Different from previous methods extracting geometry information heavily relying on hand-crafted features such as the distances between atoms, VFN learns to extract geometry information, thus significantly improving the accuracy and applicability.

The core idea is that, each residue in VFN maintains a group of hidden geometric vector representations under its residue local frame. When modeling the geometric relationship between two residues, the geometric vector representations of both residues are concatenated to formulate a vector field under one of the residue local frames. Thus, a geometric neural network can be applied to that vector field for extracting geometric information. Consequently, VFN is not only compatible with hand-crafted geometric features, but also discovers other implicit geometric features. Furthermore, the geometric features required for some protein-related tasks are very complex, and hand-crafted features are prone to fail to encode all such information. The introduction of VFN obviates the model's dependence on such hand-crafted features, thereby rendering VFN particularly amenable and broadly generalizable to these formidable challenges.

We evaluate VFN on the protein inverse folding task. The experiment result of the sequence recovery score shows that VFN can significantly improve the performance of the state-of-the-art method, PiFold, by 2.9% (51.7% *vs.* 54.6%), and outperforms the recent solid baseline, Protein MPNN, by 8.6% (46.0% *vs.* 54.6%). Furthermore, we scale up VFN with all known protein structure data. Finally, the model achieves a recovery score of **57.1**%, pushing the accuracy to the next level.

1 Introduction

Proteins are the fundamental building blocks of life and play crucial roles in biological systems. Many challenges and diseases in the life sciences are intricately related to proteins. Elucidating protein structures and the underlying molecular dynamics can provide powerful tools for treating more diseases and understanding the principles of life. Therefore, researchers [Zha08] have studied proteins intensively for decades. However, the molecular dynamics underlying protein structures are extremely complex [ZXZ+23]. Even one of the most fundamental tasks, predicting protein tertiary structures, has remained unsolved for decades [Mou05]. Recently, machine learning-based methods such as MSA [RLV+21] Transformer and AlphaFold2 [JEP+21] have made promising progress. Consequently, machine learning approaches are regarded as promising tools to further facilitate solving life science problems, especially protein-related challenges.

However, it is worth noting that AlphaFold2's success in protein prediction relies heavily on the co-evolutionary information embedded in MSAs, rather than accurately modeling the molecular dynamics underlying protein folding. Therefore, existing methods may encounter limitations when faced with scenarios or tasks where co-evolutionary information cannot be effectively utilized. Examples of such scenarios include predicting quaternary protein structures [EOP+21] and *de novo* protein design [GWVHB+23]. Current machine learning techniques still struggle with these challenges.

*WM, MZ and HC contributed equally. Work was done when WM was visiting Zhejiang University. CS is the corresponding author (e-mail: `chunhuashen@zju.edu.cn`).

Working paper.

For this challenge, it is important to devise new frameworks that can learn the principles of protein structures and the underlying molecular dynamics, thereby avoiding over-reliance on co-evolutionary information. However, this solution requires a complex system consisting of many parts, including structure network, folding simulation, learning paradigm and so on. This paper focuses on a prerequisite of such a system: a machine learning model for modeling protein structure. The goal is to develop a simpler, more generalizable and efficient structure modeling approach. Here, we selected the inverse protein folding task [IGBJ19] as a benchmark to evaluate the model's performance; because other tasks often associate with some confounding factors, *e.g.*, MSA and hand-designed energy function [RSMB04]. As such, the inverse folding task provides a more rigorous evaluation for protein structures model, free from the influences of external information sources.

For the inverse protein folding problem, numerous protein 3D structure encoders have been proposed. Recently, the best-performing methods have almost all adopted the paradigm of combining message-passing neural networks [GSR+20] (MPNNs) with hand-crafted geometric features, *e.g.*, geometric features refer to the distances and angles between atom pairs formed by atoms in two different amino acids. However, this approach may suffer from two limitations:

- Hand-crafted features may be insufficient for capturing all the geometric features in protein structures because there are many implicit geometric features that are hard to capture manually [ALFJ+17]. For instance, disulfide bonds are a kind of implicit geometric feature formed between residues under certain conditions. There may be some such unknown interactions between atoms.
- The generality of hand-crafted features is limited. In more complex tasks, such as protein model quality assessment [JES+20] and protein structure prediction [EOP+21, JEP+21, LAR+22], hand-crafted features will need to encode not only the backbone atoms (four atoms), but also the side chain atoms, which can result in very complex features. Specifically, some amino acids have a large number of atoms, such as arginine (Arg), which contains 26 atoms. If the distance between every two atoms between two Arg is encoded, there will be 676 distance pairs, making the hand-crafted features extremely complex. Additionally, there are 20 types of amino acids in nature, and any combination of them requires independently designed features, which can pose significant implementation challenges for featurizers.

To relax these limitations, we propose the geometric vector field network, termed VFN. The core idea of VFN is to maintain a set of vector representations for each amino acid, which can also be viewed as a set of virtual atoms. It is important to note that these vectors are not manually defined, but rather are intermediate features within the neural network. When constructing the geometric relationship between two amino acids, the vector representations of the two amino acids will form a vector field, which will then be fed into a neural network specifically designed for vectors to extract the geometric features between the two amino acids and model the protein structure. The advantages of this approach are two-fold:

- More expressive: Neural networks have the potential to further model geometric information beyond manually defined features through vector representations.
- Generality: The geometric information mainly comes from the vector field, removing the model's dependence on manual features, and is more suitable for tasks that require very complex manual features. Experimental results show that even with the complete removal of manually defined edge features, VFN can significantly outperform state-of-the-art performance.

Compared to the GVP [JES+20] method that also utilizes vector representations in the past, VFN differs mainly in two aspects. Firstly, to satisfy the SE(3) equivariance, GVP uses only the operators which satisfy SE(3) equivariance for vector representations, which limits the expressive power of the model. In contrast, in VFN, we transform all vector representations into the residue local frame to satisfy SE(3) equivariance without restrictions on the operators of the model. Secondly, in contrast to GVP, which does not employ an attention mechanism, VFN utilizes attention mechanism on both vector and scalar representations, further improving the performance.

Based on inverse folding, experimental results show that the VFN has achieved the best performance to date on the CATH 4.2 dataset, reaching an accuracy of 53.5% even without the use of edge featurizers. The performance can be further improved to 54.5% when using edge featurizers, which is 2.9% higher than PiFold. Moreover, compared to protein MPNN, our performance has improved by 8.6%. Furthermore, we have scaled up our model with more data, and achieved an accuracy of 57.1% on the same test set. Our contributions can be summarized as follows:

- We propose VFN, which reduces the model's reliance on hand-crafted features. The benefits are evident. On the one hand, it eliminates the need for complex hand-crafted feature design for more complicated tasks, making it more versatile. On the other hand, the model can capture implicit geometric features through vector fields.

- VFN has shown significant improvements in inverse folding tasks. It has achieved SotA performance even without edge featurizers, and with the addition of edge featurizers, the performance can be further improved, surpassing the current SotA performance by 2.9%. By scaling up with more training data, the model's accuracy was further improved to 57.1%, providing a more accurate tool for protein design.
- The proposed vector field based on residue local frame representation provides a new paradigm for future protein structure modeling methods.

2 Related Work

2.1 Representation of Protein Structure

Proteins consist of amino acids, of which there are 20 distinct types found in nature. Amino acids have a conserved backbone composed of four atoms: C, C_α, N, O, shared by all kinds of amino acids. In previous work, some hand-crafted geometric features are proposed to represent the structure of a protein based on the coordinate of backbone atoms. Those hand-crafted features can be categorized into three kinds: the distance and direction between the atoms in different residues, the transform matrix between different residues, and the Ramachandran angles [RS68]. Here, we provide more details about the transform matrix and the Ramachandran angles [RS68]. First, the Ramachandran angles are defined as the torsion angles of the polypeptide backbone, consisting of the N-C_α bond angle ϕ and the C_α-C bond angle ψ. The ϕ angle and ψ angle constitute the Ramachandran plot, whose distribution shows a significant feature. On the one hand, the distribution of these two angles is concentrated within certain regions, indicating their limited degrees of freedom. On the other hand, their angular values reflect the secondary structure, α-helices or β-sheets, at a particular amino acid position. Second, a residue local frame can be built for each residue according to the backbone atom coordinate. Thus, the transform matrix between residue local frames can then represent the correlation between residues as a geometric feature. Specifically, for building the residue local frame, AlphaFold2 [JEP+21] employ a Gram-Schmidt process; and other methods, *e.g.*, PiFold and AlphaDesign, follow the method proposed by StructTrans to set up a local frame for each residue. In the VFN, the same Gram-Schmidt process with AlphaFold2 is also employed for setting up the residue local frame.

Based on the above hand-crafted features, previous methods made significant progress in protein inverse folding tasks. However, the hand-crafted features are still inefficient to encode all the implicit geometric features, such as the bonds between atoms.

2.2 Inverse Protein Folding Problem

Engineered proteins have the opportunity to address many challenges in bio-science, biomedicine and chemistry, which attracted the interest of many researchers [APC+21, ZPG20, ALFJ+17, DM97, SM99, DAB+22]. One of the protein designing challenges that we face now is known as *inverse protein folding problem*:

Given a desired protein conformation, represented as the backbone atoms coordinates of each residue in a protein $\chi = \{x_{i,k} \in \mathbb{R}^3 : 1 \leq i \leq N, 1 \leq k \leq 4\}$ (where i represents the i-th residue in the protein with N amino acids in total, and k denotes the k-th atoms in a residue backbone, 4 atoms in total), a method needs to predict the type of each amino acids $C = \{c_i \in \mathbb{R}\}$ (c_i is the class index for residue, 20 types in total), which can be folded into the desired conformation. Previous machine-learning methods attempted to address this problem with two paradigms: The auto-regressive method and the one-shot method.

For example, GraghTrans [IGBJ19] proposed a conditional autoregressive generative model for protein sequences based on given 3D structures, which can be written as $P(C|\chi) = \prod_{i=1}^{N} P(c_i|\chi, \mathbf{c}_{<i})$. This paradigm is widely adopted by following methods, including GVP, protein MPNN and so on. Recent work, PiFold [GTL22], argues that such an auto-regressive mechanism may be unnecessary. One-shot method, *i.e.*, predicting $P(C|\chi)$ directly, can achieve state-of-the-art performance without hurting the accuracy. In this work, because we focus on modeling protein structure, here we choose the simpler one, one-shot style, as our decoder paradigm.

2.3 Protein Structure Modeling Methods

A machine learning model for modeling protein structure plays a decisive role in many bio-science tasks and may solve crucial bio-related challenges in future works. Those tasks and applications include the protein structure prediction [JEP+21, BDA+21, LAR+22, WDW+22], protein inverse folding [SK22, JKS21, LYF+14, WJAY21, PZ21,

OH21, DNG22, GMSG20, CDC+21, DAB+22, IGBJ19, MGM+21, JES+20, TGXL22, HVL+22, OLH+18, WCZQ18, SBCV+20, AEM+22, LK14, GMJ18, KZCS20, APC+21, LZW+22, MLX22, HFW+23, DLCK22, LLD+23, LSGK22], protein model quality assessment [CCE+19] and drug discovery [ZGD+23]. Some protein structure models have been proposed mainly based on the protein structure prediction and the protein inverse folding problem, which is introduced following based on different separately.

Basing on the task of *protein structure prediction*, AlphaFold2 [JEP+21] and RosettaFold [BDA+21] designed different protein structure modules. AlphaFold2 proposed an invariant point attention module(IPA), in which geometric information is extracted by the distance between some learned virtual points under different residue local frames. Notably, due to those virtual points being represented under residue local frame, the SE(3)-equivariance of IPA is ensured. In RosettaFold, the 3D structure of protein is modeled by the SE(3)-transformer [FWFW20], which is built from spherical harmonics.

Based on the *protein inverse folding problem*, many methods have been proposed, which can be categorized into distance map-based [CSL+19], voxel-based [ZCW+20, QZ20, SCM+19] and GNN-based methods. Recently, GNN-based methods dominate the protein inverse folding problem; they are introduced as follows. For GNN-based methods, a protein is usually represented as a graph based on k-nearest neighbor on euclidean distance, *i.e.*, each amino acid in a protein is treated as a node, and the 30 amino acids, for example, closest to a residue are considered its neighbors, thus, the edge in GNN can be constructed between it and its neighbors. GraphTrans [IGBJ19] first proposed a transformer-based [VSP+17] GNN model, which takes the hand-crafted geometric feature as input. Recently, more hand-crafted geometric feature-based methods have been proposed, *e.g.*, AlphaDesign [JKS21], ProteinMPNN [DAB+22], PiFold [GTL22], and the performance has been continuously improved. Among them, PiFold is the current state-of-the-art method, partly because of the more comprehensive hand-crafted featurizer. We include those hand-crafted geometric features in our methods too. However, as mentioned, such featurizer is still insufficient for covering all the geometric feature.

Another paradigm different from hand-crafted geometric feature-based GNN methods is GVP [JES+20], which proposed a novel geometric vector perceptions module. In the GVP, neural network hidden representations are separated into scalar channel and vector channel. Interestingly, the vector channels are defined as the euclidean space vector under the global frame. The advantage of GVP is that the neural network can extract geometric features through the vector channel. However, the vector channel representation is under the global frame. To achieve the SE(3)-equivariance, only very few and limited operations can be applied to the vector channel, *e.g.*, matrix multiplication, and L_2-norm operation. Other basic operations, including the element-wise addition operation and most activation functions, *e.g.*, ReLU [Aga18], will make the model violate the SE(3)-equivariance. We believe that this limitation reduces the ability of GVP to fit more complex hidden target functions. Thus, VFN is designed to eliminate this limitation by representing those hidden vectors w.r.t. the residue local frame.

3 Methodology

Here, we introduce our VFN model, which represents the proteins as a k-NN graph [IGBJ19]. Specifically, each residue is taken as a node in the graph and the edges are built between it and its k-nearest neighbors sorted on euclidean distance. Before the first VFN layer, a featurizer [GTL22] is used to extract hand-crafted geometric features, *e.g.*, the distance, direction and rotation matrix between the atoms or the residue local frames, for initializing the node and edge hidden representation in our network. Notably, due to the VFN can learn to extract geometric features, thus, the edge hidden representations can be initialized optionally by a group of statistical learnable weights, and the featurizer is removed. After initializing the edge and node representation, a message-passing neural network (MPNN) [GSR+20] is applied on those hidden representations to extract the geometric feature between neighbors via an attention mechanism. The network consists of L layers. In each layer, the hidden representation goes through three modules, node aggregation module, edge aggregation module, and global context module, sequentially.

- The node aggregation module is the foundation of VFN, which integrates the geometric features and the messages from neighbors to the representation of the central node.
- We introduce the edge aggregation module to aggregate the node, edge and geometric information, forming the edge representation.
- The global context module is applied to inject the global context feature into the node representation.

Particularly, beyond the traditional scalar hidden representation, we maintain a set of hidden geometric vector representations for each residue. In node aggregation module, those geometric vector representations are used to extract geometric correlations between residues and are updated via an attention mechanism. Specifically, when modeling the correlation

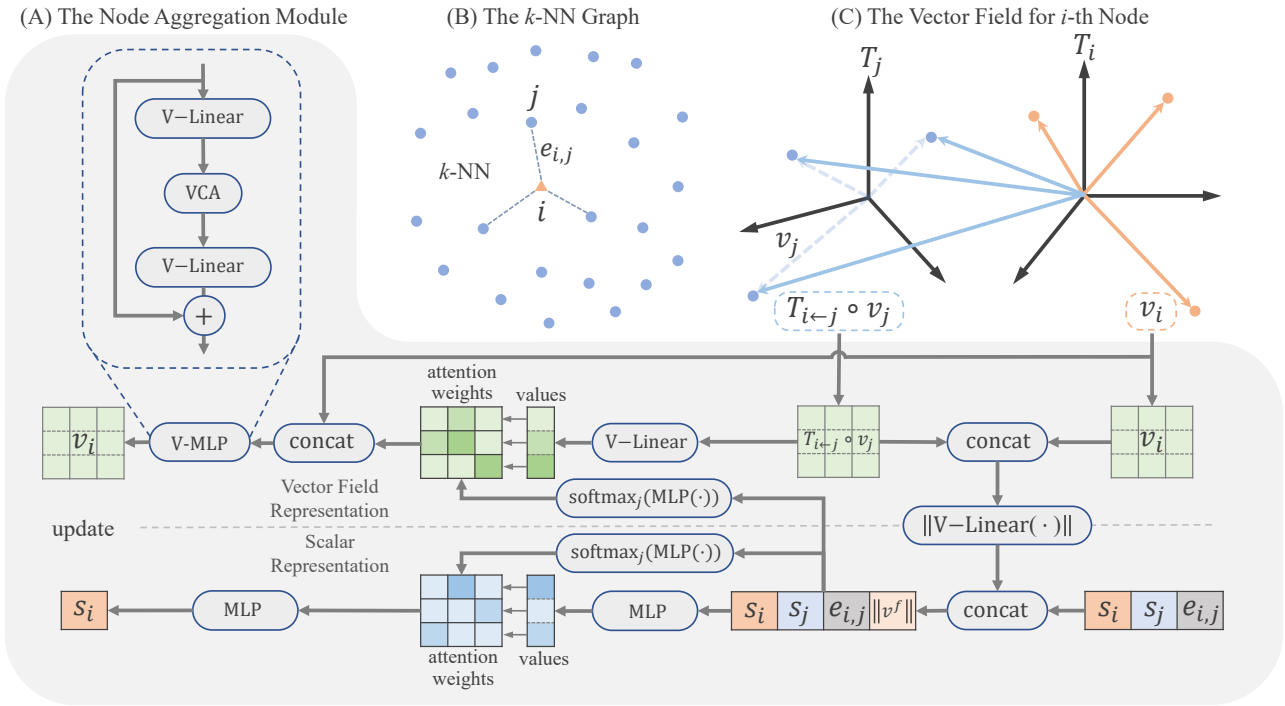


Figure 1: Overall framework for constructing the vector field.

between a residue and its neighbor, all the hidden vectors of its neighbor will be transformed to its local frame as points (as shown in Fig. 1) and construct the vector field, thus, the typical network can be applied to this vector field to extract geometric information. Besides, the geometric feature extracted from the vector field will also be saved to the edge scalar feature in the edge aggregation module. Intuitively, for such geometric vector representation, maintaining SE(3)-equivariance or SE(3)-invariance is essential [FWFW20, JES+20, JEP+21]. In VFN, the SE(3)-invariance is naturally achieved, due to all the hidden geometric vectors being calculated under the residue local frame, not global frame, and there is not any GVP-like [JES+20] limitation (§2.3) on the network operation in VFN. Details are introduced as follows.

3.1 Protein Graph Initialization

In VFN, a protein is represented as a k-NN graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$. Specifically, every residue in the protein will be taken as a node, $\mathcal{S} = \{s_i : 1 \leq i \leq N\}$. Then, for each node, the edge is constructed between it and its neighbor, written as $\mathcal{E} = \{e_{i,j}\}_{j \in \mathrm{Topk}(i)}$, where $\mathrm{Topk}(i)$ returns the index of k-nearest neighbor for i-th residue sorted on the euclidean distance. After constructing the protein graph, we compute five initial features, $s_i^{\mathrm{init}}, e_{i,j}^{\mathrm{init}}, v_i^{\mathrm{init}}, T_i, T_{i \leftarrow j}$, for the inputs and operations of VFN. Those features are introduced following.

For initializing the d-dimensional hidden representation of node and edge, $s_i^{\mathrm{init}} \in \mathbb{R}^d, e_{i,j}^{\mathrm{init}} \in \mathbb{R}^d$, we employ the PiFold featurizer [GTL22], denote as Featurizer, to nodes and edges, respectively:

$$s_i^{\mathrm{init}},\ e_{i,j}^{\mathrm{init}} = \mathrm{Featurizer}(\mathbf{X}^i_{C,\alpha,N,O},\ \mathbf{X}^j_{C,\alpha,N,O}) \tag{1}$$

where $\mathbf{X}^i_{C,\alpha,N,O} \in \mathbb{R}^{4\times 3}$ and $\mathbf{X}^j_{C,\alpha,N,O} \in \mathbb{R}^{4\times 3}$ are the coordinates of C, C_α, N, and O in i- and j-th residue respectively; and $j \in \mathrm{Topk}(i)$. As mentioned, the edge featurizer can be removed optionally and $e_{i,j}^{\mathrm{init}}$ is initialized by a group of learnable statistical weights. Next, the vector representation v_i^{init} is initialized for each node by applying a linear layer on the initial node feature s_i^{init}, written as:

$$v_i^{\mathrm{init}} = \mathrm{Linear}(s_i^{\mathrm{init}})\ ,\ \ v_i^{\mathrm{init}} \in \mathbb{R}^{V\times 3} \tag{2}$$

where V represents the total number of vectors in hidden representation. The i-th vector representation v_i^{init} is under the i-th residue local frame T_i, which is established by:

$$T_i = \mathrm{GetBackboneFrame}(\mathbf{X}^i_C,\ \mathbf{X}^i_\alpha,\ \mathbf{X}^i_N) \tag{3}$$

Algorithm 1 The pseudo-code for setting up the local frame of amino acids.

```
def GetBackboneFrame(X_C, X_α, X_N):                    X_C, X_α, X_N ∈ ℝ^3
    V⃗_{α→C} = X_C − X_α
    V⃗_{α→N} = X_N − X_α
    e⃗_1 = V⃗_{α→C} / ‖V⃗_{α→C}‖
    u⃗_2 = V⃗_{α→N} − e⃗_1(e⃗_1^⊤ V⃗_{α→N})
    e⃗_2 = u⃗_2 / ‖u⃗_2‖
    e⃗_3 = e⃗_1 × e⃗_2
    R = Concat(e⃗_1, e⃗_2, e⃗_3)
    t⃗ = X_α
    T = (R, t⃗)
    return T
```

where T_i denotes the Euclidean transform from the i-th residue local frame to a global reference frame. GetBackboneFrame is a Gram–Schmidt process for setting up the residue local frame, illustrated in Algorithm 1, proposed by AlphaFold2 [JEP+21].

With T_i, we can calculate the transform matrix between different residue local frames for each edge in $\mathcal{E}$. Those can be written as:

$$T_{i \leftarrow j} \coloneqq T_i^{-1} \circ T_j \tag{4}$$

$T_{i \leftarrow j}$ represents the transform matrix for transforming a point coordinate from j-th residue local frame to i-th residue local frame; The $\circ$ operation represents the composition of Euclidean transformations for two transform matrices; T_i^{-1} denotes the inverse transform of T_i. Those transform matrices will be used to transform the hidden vector representation between different residue local frames later.

3.2 Geometric Vector Field Network

VFN is a L-layers sequential network, which takes $v_i^{\text{init}}, s_i^{\text{init}}, e_{i,j}^{\text{init}}$ as the first layer input. All the layers in VFN are the same, and for a single layer in VFN, we denote the layer inputs as $v_i^{in} \in \mathbb{R}^{V \times 3}$, $s_i^{in}, e_{i,j}^{in} \in \mathbb{R}^d$ and the outputs as $v_i^{out} \in \mathbb{R}^{V \times 3}$, $s_i^{out}, e_{i,j}^{out} \in \mathbb{R}^d$.

Constructing the vector field. At the layer beginning, a vector field $v_{i,j}^f \in \mathbb{R}^{V \times 3}$ is constructed for each edge in $\mathcal{E}$ to extract geometric feature for the node and edge aggregation module following. The way of building the vector field for an edge is transforming the neighbor hidden vector representation v_j^{in} from the neighbor local frame T_j to its local frame T_i as points, and concatenating with v_i^{in} to generate an intermediate variable $v_{i,j}^m \in \mathbb{R}^{2V \times 3}$. Then, the variable $v_{i,j}^m$ is sent into a vector linear layer V-Linear to generate the $v_{i,j}^f$, denoted as

$$\begin{aligned} v_{i,j}^m &= \text{Concat}(v_i^{in}, T_{i \leftarrow j} \circ v_j^{in}) \\ v_{i,j}^f &= \text{V-Linear}(v_{i,j}^m) \\ \text{V-Linear}(v_{i,j}^m) &\coloneqq w_1 \cdot v_{i,j}^m + b \end{aligned} \tag{5}$$

The $w_1 \in \mathbb{R}^{V \times V \times 1}$ and $b \in \mathbb{R}^{V \times 3}$ are the groups of trainable weights, and the bias term b is optional. The norm of $v_{i,j}^f$, denotes as $\|v_{i,j}^f\|$, is used as the geometric attribute later.

Node aggregation module. After obtaining the vector field $v_{i,j}^f$ for each edge, the node aggregation module is employed to update the node representation s_i^{in} and v_i^{in} by synthesizing the information of $s_i^{in}, s_j^{in}, e_{i,j}^{in}, \|v_{i,j}^f\|$ via a transformer-base [VSP+17, GTL22] mechanism. Firstly, those variables are sent into a MLP layer to generate the attention weights $a_{i,j}^s, a_{i,j}^{vec}$ for scale attention and vector attention, respectively, written as:

$$\begin{aligned} w_{i,j} &= \text{MLP}(\text{Concat}(s_i^{in}, s_j^{in}, e_{i,j}^{in}, \|v_{i,j}^f\|)) \\ a_{i,j}^s, a_{i,j}^{vec} &= \text{softmax}_j(w_{i,j}) \end{aligned} \tag{6}$$

where $a^s_{i,j}, a^{vec}_{i,j} \in \mathbb{R}^H$ is the attention weights for scalar feature and vector field, respectively; H is the number of heads in the multi-head attention mechanism [VSP+17].

Next, we calculate the scalar value $s^{val}_{i,j} \in \mathbb{R}^d$ and the vector field value $v^{val}_{i,j} \in \mathbb{R}^V$, separately, for the multi-head attention. For the scale attention, the $s^{val}_{i,j}$ is firstly calculated by sending the $s^{in}_j, e^{in}_{i,j}, \|v^f_{i,j}\|$ into a MLP layer and then summed weighted according to the $a^s_{i,j}$. Similarly, for the vector attention, the $v^{val}_{i,j}$ is generated by transforming the neighbour vector representation v^{in}_j to centre local frame T_i and then applying another V-Linear layer on it; the resulting vector variable is summed weighted based on the $a^{vec}_{i,j}$. Those processes can be written as:

$$\underbrace{\begin{aligned} s^{val}_{i,j} &= \mathrm{MLP}(\mathrm{concat}(s^{in}_j, e^{in}_{i,j}, \|v^f_{i,j}\|)) \\ s^a_i &= \textstyle\sum_j a^s_{i,j} \cdot s^{val}_{i,j} \end{aligned}}_{\text{scalar attention}}, \qquad \underbrace{\begin{aligned} v^{val}_{i,j} &= \text{V-Linear}(T_{i\leftarrow j} \circ v^{in}_j) \\ v^a_i &= \textstyle\sum_j a^{vec}_{i,j} \cdot v^{val}_{i,j} \end{aligned}}_{\text{vector attention}} \tag{7}$$

The $s^a_i \in \mathbb{R}^d, v^a_i \in \mathbb{R}^{V\times 3}$ are the resulting scalar and vector features from the multi-head attention, respectively. Notably, there are H heads in the attention; the channel of the value $s^{val}_{i,j}, v^{val}_{i,j}$ is split to H group and weighted summed with corresponding head in $a^s_{i,j}, a^{vec}_{i,j}$, independently. Next, s^a_i, v^a_i is sent to a MLP layer, independently:

$$\begin{aligned} \tilde{s}_i &= \mathrm{MLP}(s^a_i) & \tilde{s}_i &\in \mathbb{R}^d \\ v^{out}_i &= \text{V-MLP}(\mathrm{concat}(v^{in}_i, v^a_i)) & v^{out}_i &\in \mathbb{R}^{V\times 3} \end{aligned} \tag{8}$$

Specifically, here, we design a unique MLP layer for the vector field, termed V-MLP. In the V-MLP, there are two V-Linear layers and a direction activation function, termed VCA (Vector Cosine-similarity Activation), basing on the cosine similarity between the hidden vectors direction and the direction of a group statistic learnable vector, illustrated as:

$$\begin{aligned} \text{V-MLP}(v_\mu) &:= \text{V-Linear}(\mathrm{VCA}(\text{V-Linear}(v_\mu))) \\ \mathrm{VCA}(v_\sigma) &:= \frac{\vec{w}_2 \cdot v_\sigma}{\|\vec{w}_2\|\|v_\sigma\|} v_\sigma \end{aligned} \tag{9}$$

where $v_\mu \in \mathbb{R}^{V_\mu\times 3}, v_\sigma \in \mathbb{R}^{V_\sigma\times 3}$ is the V_μ-channel and V_σ-channel input of V-MLP and VCA, respectively, and $\vec{w}_2 \in \mathbb{R}^{V_\sigma\times 3}$ is a group of learnable vector weight.

Edge aggregation module. Next, the $\tilde{s}_i$ and $\|v^f_{i,j}\|$ is used to update the edge representation $e^{in}_{i,j}$ via a MLP layer, as shown in:

$$e^{out}_{i,j} = \mathrm{MLP}(\mathrm{Concat}(\tilde{s}_i, \tilde{s}_j, e^{in}_{i,j}, \|v^f_{i,j}\|)) \tag{10}$$

Global context module. Similar to PiFold [GTL22] and GCA [TGXL22], a global context module is also employed to extract the global information for $\tilde{s}_i$, in which, the $\{\tilde{s}_i\}$ is averaged globally to a global representation $g \in \mathbb{R}^d$. Then, g is used to update $\tilde{s}_i$ via a gating mechanism, defined as:

$$\begin{aligned} g &= \mathrm{mean}_i(\{\tilde{s}_i\}), \\ s^{out}_i &= \tilde{s}_i \cdot \mathrm{sigmoid}(\mathrm{MLP}(g)). \end{aligned} \tag{11}$$

3.3 Decoder and Loss Function

Following Pifold, VFN predicts the type of residue in a one-shot style [GTL22], not using the autoregressive decoder [IGBJ19, DAB+22]. Thus, the decoder is extremely simple, consisting of only a single Linear layer on the final layer node representation s_i. During training, the cross entropy loss is employed to supervise our model.

4 Experiments

In this section, we conduct extensive experiments on various datasets including the CATH 4.2 dataset [OMJ+97] and the TS50 and TS500 datasets [LYF+14] to evaluate the performance of our proposed method. The results show that VFN achieves superior recovery scores compared to state-of-the-art methods, and demonstrates good scalability on a larger dataset. Furthermore, we perform detailed ablation studies to investigate the impact of different hyper-parameters such as the number of layers and vector field modules design on the performance of our model.

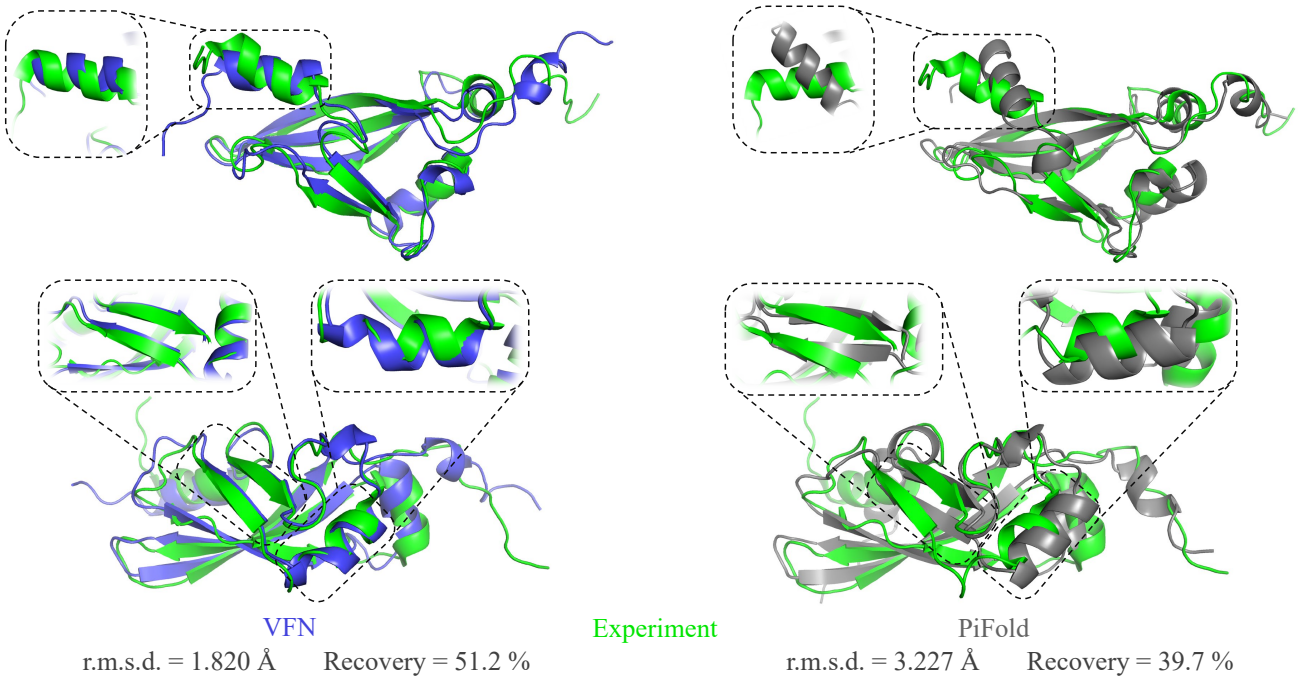


Figure 2: Visualization results of a challenging sample (PDB 2KRT). We use AlphaFold2 to recover the structure based on the predicted sequence and compare it against the experimentally determined ground-truth structure. The results show that, compared to PiFold [GTL22], our method is more stable and our results are more consistent with the true structure, with only very minor differences in the structure.

4.1 Implementation Details

Datasets. Unless specified, our experiments are conducted on the CATH 4.2 [OMJ+97] dataset using the same data splitting as previous works such as GVP [JES+20] and Pifold [GTL22]. The dataset consists of 18,024 proteins for training, 608 for validation, and 1120 for testing. During the evaluation, we also test our model on two smaller datasets, TS50 and TS500 [JES+20, GTL22], for validate the generalizability. Furthermore, we also create another larger training set by incorporating data from the PDB [BBK+17]. We apply the same strategy as in [ZGD+23] to collect and filter structures. Additionally, the proteins with sequences highly similar to test set proteins are also removed. By using the expanded dataset, we are able to scale up the VFN.

Model settings. Unless otherwise specified, the number of layers L is set to 10 and the number of vector V is set to 32. The dimension of scalar hidden representation d is 128. In ablation studies, we disable the edge featurizer by default.

4.2 Main results

Experimental results on CATH. In Table 1, we present the perplexity and recovery score results for VFN, compared to previous works such as [IGBJ19, JES+20, TGXL22, HVL+22, GTL22] , across various subsets of the CATH 4.2 dataset. Among the subsets considered in our analysis, the "Short" subset refers to proteins with a length of up to 100 amino acids, while the "Single chain" subset exclusively includes single chain proteins.

Compared to the other methods, our proposed method shows superior recovery scores on all three datasets. Specifically, our method achieves significant improvements in recovery score compared to PiFold, with gains of 3%, 3.6%, and 2.9% on the Short, Single-chain, and All datasets, respectively. When it comes to perplexity, VFN still performs well, with the lowest scores of 5.34, 5.37, and 4.11 for the “Short”, “Single-chain”, and “All” datasets, respectively. We also demonstrate a model that does not use the edge featurizer, referred to as ‘Ours w/o edges’, which still significantly outperforms the current state-of-the-art (SotA) model in all metrics. To demonstrate the scalability of our model, we also train a model on a larger dataset labeled as “Ours$^+$”. Notably, “Ours$^+$” achieves even higher recovery scores than "Ours", highlighting the effectiveness of our approach in handling larger and more diverse protein structures. The number of layers for “Ours”, “Ours

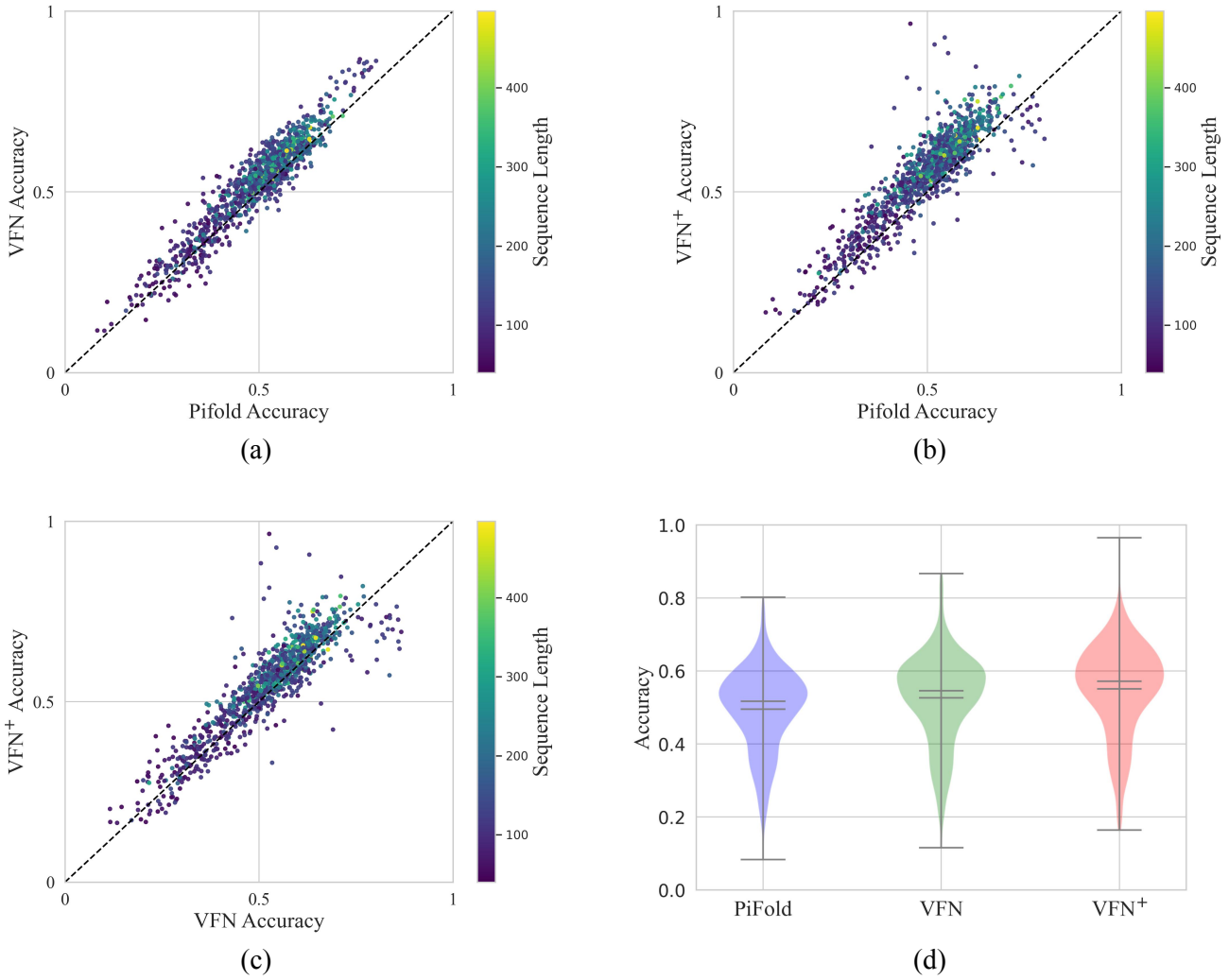


Figure 3: The performance comparison of VFN, VFN$^+$ and PiFold. In Figures (a)(b)(c), each point represents a protein, its x- and y-coordinates represent the prediction accuracy of two models respectively, and its color corresponds to the sequence length. In Figure (d), we show the recovery accuracy distribution for different models on the CATH 4.2 test set [OMJ+97].

w/o edges", and "Ours$^+$" are set to 15, 12, and 15, respectively, based on the ablation experiments. All other settings use the default settings of the model.

Furthermore, we select a challenging sample (PDB 2KRT) from the CATH test set and use AlphaFold2 to predict its structure. We then visually compared the predicted structure with the experimentally determined native structure, as shown in Fig. 2. The visualizations demonstrate that, compared with PiFold, VFN achieves a higher accuracy on structure recovery, with only minor differences from the experimentally determined protein structures.

Experimental results on TS50 and TS500. We further evaluate the performance of VFN on TS50 and TS500 datasets to examine the model's generalizability. As shown in Table 2, we follow PiFold [GTL22] and report the perplexity, median recovery, and worst recovery on the TS50 and TS500 datasets. Results show that our method outperforms other state-of-the-art methods in terms of recovery scores, with a median recovery score of 62.12% on TS500 and 59.04% on TS50. Furthermore, our method achieves a lower perplexity and a higher worst recovery than PiFold on both datasets, indicating better generalization ability. In Fig. 3, we further compare the overall performance differences between our model and Pifold.

Table 1: Experimental results comparison on the CATH dataset. All baselines are reproduced by [GTL22]. "Ours" and "Ours w/o edges" refer to the models with and without the edge featurizer, respectively. "Ours$^+$" refers to a larger version of our proposed method that is trained on a larger dataset. Notably, it achieves even higher recovery scores than "Ours", highlighting the effectiveness of our approach in exploiting larger and more diverse protein structures.

Model	Perplexity			Recovery (%)			CATH version
	Short	Single-chain	All	Short	Single-chain	All	
StructGNN	8.29	8.74	6.40	29.44	28.26	35.91	4.2
GraphTrans	8.39	8.83	6.63	28.14	28.46	35.82	4.2
GCA	7.09	7.49	6.05	32.62	31.10	37.64	4.2
GVP	7.23	7.84	5.36	30.60	28.95	39.47	4.2
GVP-large	7.68	6.12	6.17	32.60	39.40	39.20	4.3
AlphaDesign	7.32	7.63	6.30	34.16	32.66	41.31	4.2
ESM-IF	8.18	6.33	6.44	31.30	38.50	38.30	4.3
ProteinMPNN	6.21	6.68	4.61	36.35	34.43	45.96	4.2
PiFold	6.04	6.31	4.55	39.84	38.53	51.66	4.2
Ours w/o edges	5.33	5.38	4.15	41.90	39.13	54.14	4.2
Ours	5.34	5.37	4.11	42.68	**42.11**	54.56	4.2
Ours$^+$	**5.31**	**5.32**	**3.66**	**42.76**	41.77	**57.14**	4.2

Table 2: Experimental results on TS50 and TS500.The optimal results are labeled in bold, and suboptimal results are underlined.

Model	TS50		TS500	
	Perplexity	Recovery (%)	Perplexity	Recovery (%)
SPIN		30.30		30.30
SPIN2		33.60		36.60
Wang's model		33.00		36.14
SPROF		39.16		40.25
ProDCoNN		40.69		42.20
DenseCPD		50.71		55.53
StructGNN	5.40	43.89	4.98	45.69
GraphTrans	5.60	42.20	5.16	44.66
GVP	4.71	44.14	4.20	49.14
GCA	5.09	47.02	4.72	47.74
AlphaDesign	5.25	48.36	4.93	49.23
ProteinMPNN	3.93	54.43	3.53	58.08
PiFold	3.86	58.72	3.44	60.42
Ours	**3.67**	**59.04**	**3.23**	**62.12**

4.3 Ablations

In this section, we carefully investigate the design of vector field modules we proposed. The number of the encoder layers are set to 10 and the number of vectors is set to 32, if not specified.

The number of layers. We first vary the number of layers in our model and investigate its impact on recovery and perplexity. As shown in Table 3a, increasing the number of layers from 5 to 15 leads to a marginal improvement in recovery scores, with the highest recovery achieved at 54.56% for 15 layers. As for the perplexity, we observed that there was no significant change, and in fact, there was a slight increase in perplexity when the number of layers further increases. In addition, we also conduct ablation experiments without using the edge featurizer. The results show that not using edge features only leads to a small drop in performance, which indirectly proves the effectiveness of VFN and greatly reduces the dependence on hand-crafted features.

The number of vectors. In Table 3b we also perform an ablation study to determine the optimal number for vectors. It demonstrates that increasing the vector number from 16 to 32 brings a slight improvement in recovery score from 54.01% to 54.48%, while maintaining a low perplexity score of 4.14. However, further increasing the number to 64 results in a decrease

Table 3: Ablation studies on the CATH 4.2 dataset. We use the default model settings unless otherwise specified. When calculating the number of parameters, we only count the number of parameters occupied by this module in one layer.

(a) Varying the number of layers.

#layers		5	8	10	12	15
w/ edge feature	Recovery (%)	52.62	53.85	54.48	54.55	**54.56**
	Perplexity	4.27	4.14	4.11	**4.09**	4.11
w/o edge feature	Recovery (%)	52.00	53.25	53.52	**54.14**	53.76
	Perplexity	4.35	4.20	4.20	**4.15**	4.16

(b) Varying the number of vectors.

#Vec.	16	32	64
Recovery (%)	53.33	**53.52**	53.39
Perplexity	4.21	**4.20**	4.20

(c) Varying the V-Linear.

	Recovery (%)	Params
V-Linear	**53.52**	**1.0K**
Linear	53.45	9.0K

(d) Varying the V-MLP.

	Recovery (%)	Params
V-MLP	**53.52**	4.2K
MLP	53.33	36.0K
w/o V-MLP	53.33	**0.0K**

(e) Varying the vector field design.

	Recovery (%)
Baseline	**53.52**
w/o Vec. field	32.79
w/o $\|v_{i,j}^f\|$ for $e_{i,j}$	51.66

in recovery score to 53.85% and a higher perplexity score of 4.19. Overall, these findings suggest that 32 is a more suitable choice.

Vector field module. In this experiment, we investigate the effect of different vector field modules on the performance of our model. Results in Table 3e show that removing the whole vector field module leads to a significant drop in both recovery and perplexity, suggesting that this module plays an important role in capturing the protein folding patterns and we also show the results of ablations on whether injecting the $\|v_{i,j}^f\|$ as mentioned in the edge aggregation module, removing the $\|v_{i,j}^f\|$ lead to a 1.9% recovery drop, which demonstrates that vector field does provide useful information to the edge features. Besides, we verify the effectiveness of V-Linear and V-MLP in Table 3c and Table 3d, the experimental results indicate that our design not only effectively reduces the number of parameters but also brings about certain performance improvements.

5 Acknowledge

The authors are grateful to Jianjun Zhou for data processing.

References

[AEM+22] Namrata Anand, Raphael Eguchi, Irimpan I. Mathews, Carla P. Perez, Alexander Derry, Russ B. Altman, and Po-Ssu Huang. Protein sequence design with a learned potential. *Nature Communications*, 13(1):746, 2022.

[Aga18] Abien Fred Agarap. Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375*, 2018.

[ALFJ+17] Rebecca F. Alford, Andrew Leaver-Fay, Jeliazko R. Jeliazkov, Matthew J. O'Meara, Frank P. DiMaio, Hahnbeom Park, Maxim V. Shapovalov, P. Douglas Renfrew, Vikram K. Mulligan, and et al. Kappel, Kalli. The Rosetta all-atom energy function for macromolecular modeling and design. *Computational and Structural Biotechnology Journal*, 13(6):3031–3048, 2017.

[APC+21] Ivan Anishchenko, Samuel J. Pellock, Tamuka M. Chidyausiku, Theresa A. Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, and et al. Bera, Asim K. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, 2021.

[BBK+17] Stephen K. Burley, Helen M. Berman, Gerard J. Kleywegt, John L. Markley, Haruki Nakamura, and Sameer Velankar. Protein data bank (pdb): the single global macromolecular structure archive. *Protein Crystallography: Methods and Protocols*, pages 627–641, 2017.

[BDA+21] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, and et al. Schaeffer, R. Dustin. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[CCE+19] Jianlin Cheng, Myong-Ho Choe, Arne Elofsson, Kun-Sop Han, Jie Hou, Ali HA Maghrabi, Liam J. McGuffin, David Menéndez-Hurtado, Kliment Olechnovič, and et al. Schwede, Torsten. Estimation of model accuracy in casp13. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1361–1377, 2019.

[CDC+21] Yue Cao, Payel Das, Vijil Chenthamarakshan, Pin-Yu Chen, Igor Melnyk, and Yang Shen. Fold2seq: A joint sequence (1d)-fold (3d) embedding-based generative model for protein design. In *International Conference on Machine Learning*, pages 1261–1271. PMLR, 2021.

[CSL+19] Sheng Chen, Zhe Sun, Lihua Lin, Zifeng Liu, Xun Liu, Yutian Chong, Yutong Lu, Huiying Zhao, and Yuedong Yang. To improve protein sequence profile prediction through image captioning on pairwise residue distance map. *Journal of Chemical Information and Modeling*, 60(1):391–399, 2019.

[DAB+22] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J. Ragotte, Lukas F. Milles, Basile IM Wicky, Alexis Courbet, Rob J. de Haas, and et al. Bethel, Neville. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[DLCK22] Baldwin Dumortier, Antoine Liutkus, Clément Carré, and Gabriel Krouk. Petribert: Augmenting bert with tridimensional encoding for inverse protein folding and design. *BioRxiv*, pages 2022–08, 2022.

[DM97] Bassil I. Dahiyat and Stephen L. Mayo. Probing the role of packing specificity in protein design. *Proceedings of the National Academy of Sciences*, 94(19):10172–10177, 1997.

[DNG22] Wenze Ding, Kenta Nakai, and Haipeng Gong. Protein design via deep learning. *Briefings in Bioinformatics*, 23(3), 2022.

[EOP+21] Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, Olaf Ronneberger, Sebastian Bodenstein, Michal Zielinski, Alex Bridgland, Anna Potapenko, Andrew Cowie, Kathryn Tunyasuvunakool, Rishub Jain, Ellen Clancy, Pushmeet Kohli, John Jumper, and Demis Hassabis. Protein complex prediction with alphafold-multimer. *BioRxiv*, 2021.

[FWFW20] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. SE(3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

[GMJ18] Joe G. Greener, Lewis Moffat, and David T. Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific Reports*, 8(1):16189, 2018.

[GMSG20] Wenhao Gao, Sai Pooja Mahajan, Jeremias Sulam, and Jeffrey J. Gray. Deep learning in protein structural modeling and design. *Patterns*, 1(9):100142, 2020.

[GSR+20] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Message passing neural networks. *Machine Learning Meets Quantum Physics*, pages 199–214, 2020.

[GTL22] Zhangyang Gao, Cheng Tan, and Stan Li. PiFold: Toward effective and efficient protein inverse folding. *International Conference on Learning Representations*, 2022.

[GWVHB+23] Pablo Gainza, Sarah Wehrle, Alexandra Van Hall-Beauvais, Anthony Marchand, Andreas Scheck, Zander Harteveld, Stephen Buckley, Dongchun Ni, Shuguang Tan, and et al. Sverrisson, Freyr. De novo design of protein interactions with learned surface fingerprints. *Nature*, pages 1–9, 2023.

[HFW+23] Bin Huang, Tingwen Fan, Kaiyue Wang, Haicang Zhang, Chungong Yu, Shuyu Nie, Yangshuo Qi, Wei-Mou Zheng, Jian Han, and et al. Fan, Zheng. Accurate and efficient protein sequence design through learning concise local environment of residues. *Bioinformatics*, page btad122, 2023.

[HVL+22] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International Conference on Machine Learning*, pages 8946–8970. PMLR, 2022.

[IGBJ19] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in Neural Information Processing Systems*, 32, 2019.

[JEP+21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, and et al. Potapenko, Anna. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[JES+20] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *International Conference on Learning Representations*, 2020.

[JKS21] Michael Jendrusch, Jan Korbel, and Kashif Sadiq. Alphadesign: A de novo protein design framework based on AlphaFold. *BioRxiv*, pages 2021–10, 2021.

[KZCS20] Mostafa Karimi, Shaowen Zhu, Yue Cao, and Yang Shen. De novo protein design for novel folds using guided conditional wasserstein generative adversarial networks. *Journal of Chemical Information and Modeling*, 60(12):5667–5681, 2020.

[LAR+22] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, and et al. Candido, Sal. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*, 2022.

[LK14] Jie Li and Patrice Koehl. 3d representations of amino acids—applications to protein sequence comparison and classification. *Computational and Structural Biotechnology Journal*, 11(18):47–58, 2014.

[LLD+23] Alex J. Li, Mindren Lu, Israel Desta, Vikram Sundar, Gevorg Grigoryan, and Amy E. Keating. Neural network-derived potts models for structure-based protein design using backbone atomic coordinates and tertiary motifs. *Protein Science*, 32(2):e4554, 2023.

[LSGK22] Alex J. Li, Vikram Sundar, Gevorg Grigoryan, and Amy E. Keating. Terminator: a neural framework for structure-based protein design using tertiary repeating motifs. *arXiv preprint arXiv:2204.13048*, 2022.

[LYF+14] Zhixiu Li, Yuedong Yang, Eshel Faraggi, Jian Zhan, and Yaoqi Zhou. Direct prediction of profiles of sequences compatible with a protein structure by neural networks with fragment-based local and energy-based nonlocal profiles. *Proteins: Structure, Function, and Bioinformatics*, 82(10):2565–2573, 2014.

[LZW+22] Yufeng Liu, Lu Zhang, Weilun Wang, Min Zhu, Chenchen Wang, Fudong Li, Jiahai Zhang, Houqiang Li, Quan Chen, and Haiyan Liu. Rotamer-free protein sequence design based on deep learning and self-consistency. *Nature Computational Science*, 2(7):451–462, 2022.

[MGM+21] Jack B. Maguire, Daniele Grattarola, Vikram Khipple Mulligan, Eugene Klyshko, and Hans Melo. Xenet: Using a new graph convolution to accelerate the timeline for protein design on quantum computers. *PLoS Computational Biology*, 17(9):e1009037, 2021.

[MLX22] Matt McPartlon, Ben Lai, and Jinbo Xu. A deep SE(3)-equivariant model for learning inverse protein folding. *BioRxiv*, pages 2022–04, 2022.

[Mou05] John Moult. A decade of casp: progress, bottlenecks and prognosis in protein structure prediction. *Current Opinion in Structural Biology*, 15(3):285–289, 2005.

[OH21] Sergey Ovchinnikov and Po-Ssu Huang. Structure-based protein design with deep learning. *Current Opinion in Structural Biology*, 65:136–144, 2021.

[OLH+18] James O'Connell, Zhixiu Li, Jack Hanson, Rhys Heffernan, James Lyons, Kuldip Paliwal, Abdollah Dehzangi, Yuedong Yang, and Yaoqi Zhou. Spin2: Predicting sequence profiles from protein structures using deep neural networks. *Proteins: Structure, Function, and Bioinformatics*, 86(6):629–633, 2018.

[OMJ+97] Christine Orengo, Alex Michie, Susan Jones, David Jones, Mark Swindells, and Janet Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

[PZ21] Robin Pearce and Yang Zhang. Deep learning techniques have significantly impacted protein structure prediction and protein design. *Current Opinion in Structural Biology*, 68:194–207, 2021.

[QZ20] Yifei Qi and John ZH Zhang. Densecpd: improving the accuracy of neural-network-based computational protein sequence design with densenet. *Journal of Chemical Information and Modeling*, 60(3):1245–1252, 2020.

[RLV+21] Roshan M. Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[RS68] GN. T. Ramachandran and V. Sasisekharan. Conformation of polypeptides and proteins. *Advances in Protein Chemistry*, 23:283–437, 1968.

[RSMB04] Carol A. Rohl, Charlie EM. Strauss, Kira MS. Misura, and David Baker. Protein structure prediction using rosetta. In *Methods in enzymology*, volume 383, pages 66–93. Elsevier, 2004.

[SBCV+20] Alexey Strokach, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, and Philip M. Kim. Fast and flexible protein design using deep graph neural networks. *Cell Systems*, 11(4):402–411, 2020.

[SCM+19] Raghav Shroff, Austin Cole, Barrett Morrow, Daniel Diaz, Isaac Donnell, Jimmy Gollihar, Andrew Ellington, and Ross Thyer. A structure-based deep learning framework for protein engineering. *BioRxiv*, 2019.

[SK22] Alexey Strokach and Philip M. Kim. Deep generative modeling for protein design. *Current Opinion in Structural Biology*, 72:226–236, 2022.

[SM99] Arthur G. Street and Stephen L. Mayo. Computational protein design. *Structure*, 7(5):R105–R109, 1999.

[TGXL22] Cheng Tan, Zhangyang Gao, Jun Xia, and Stan Li. Generative de novo protein design with global context. *arXiv preprint arXiv:2204.10673*, 2022.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[WCZQ18] Jingxue Wang, Huali Cao, John ZH Zhang, and Yifei Qi. Computational protein design with deep learning neural networks. *Scientific Reports*, 8(1):1–9, 2018.

[WDW+22] Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, and et al. Berger, Bonnie. High-resolution de novo structure prediction from primary sequence. *BioRxiv*, pages 2022–07, 2022.

[WJAY21] Zachary Wu, Kadina E. Johnston, Frances H. Arnold, and Kevin K. Yang. Protein sequence design with deep generative models. *Current Opinion in Structural Biology*, 65:18–27, 2021.

[ZCW+20] Yuan Zhang, Yang Chen, Chenran Wang, Chun-Chao Lo, Xiuwen Liu, Wei Wu, and Jinfeng Zhang. Prodconn: Protein design using a convolutional neural network. *Proteins: Structure, Function, and Bioinformatics*, 88(7):819–829, 2020.

[ZGD+23] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. 2023.

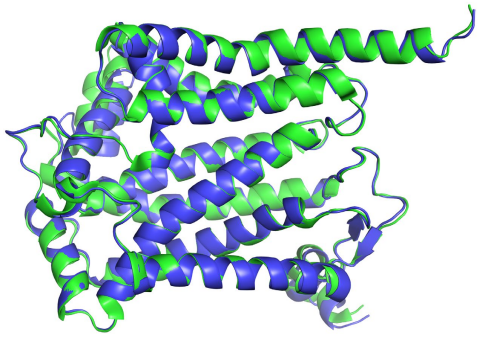
[Zha08] Yang Zhang. I-tasser server for protein 3d structure prediction. *BMC Bioinformatics*, 9:1–8, 2008.

[ZPG20] Jianfu Zhou, Alexandra E. Panaitiu, and Gevorg Grigoryan. A general-purpose protein design framework based on mining sequence–structure relationships in known protein structures. *Proceedings of the National Academy of Sciences*, 117(2):1059–1068, 2020.

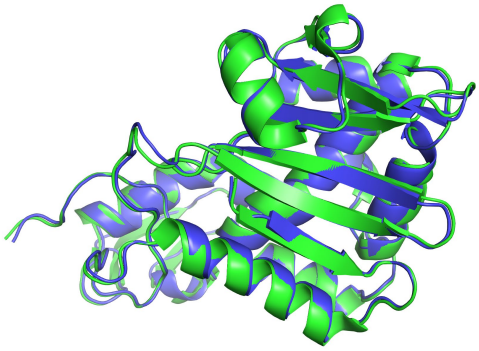
[ZXZ+23] Kailong Zhao, Yuhao Xia, Fujin Zhang, Xiaogen Zhou, Stan Li, and Guijun Zhang. Protein structure and folding pathway prediction based on remote homologs recognition using pathreader. *Communications Biology*, 6(1):243, 2023.

6 Supplementary

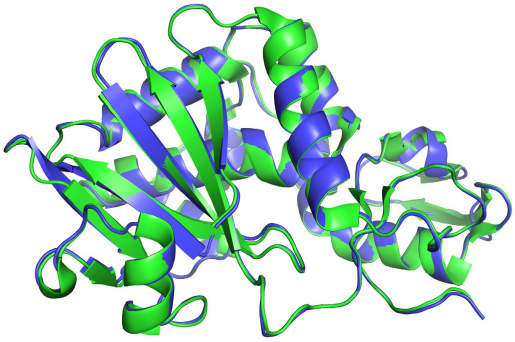
Here, we visualize a few more restoration results of VFN. Specifically, we utilize AlphaFold2 to predict the structures of the sequences generated by VFN, and subsequently compare the predicted structures of our method (blue) against the ground-truth experimental structures (green).



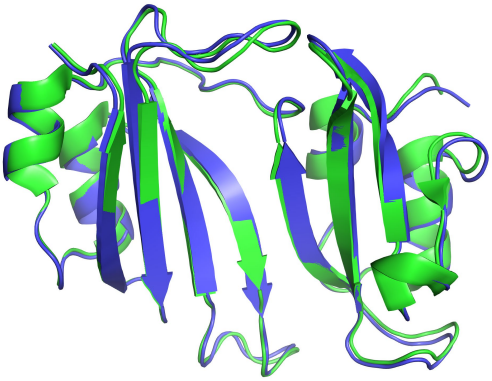
1EZV
r.m.s.d. = 0.490 Å recovery = 56.10 %



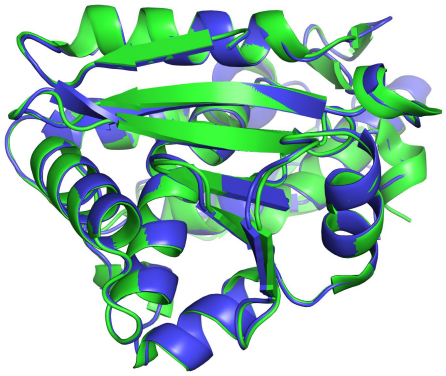
1RL0
r.m.s.d. = 0.483 Å recovery = 67.05 %



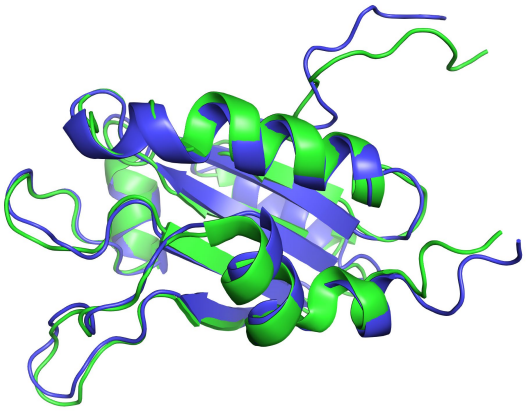
1LP8
r.m.s.d. = 0.318 Å recovery = 68.90 %



1XXM
r.m.s.d. = 0.520 Å recovery = 68.48 %

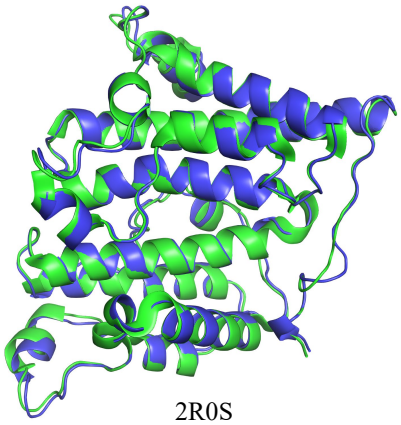


1NIO
r.m.s.d. = 0.303 Å recovery = 67.20 %

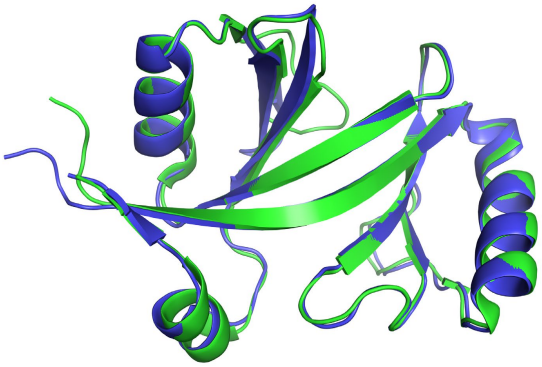


2DJ0
r.m.s.d. = 1.139 Å recovery = 83.21 %

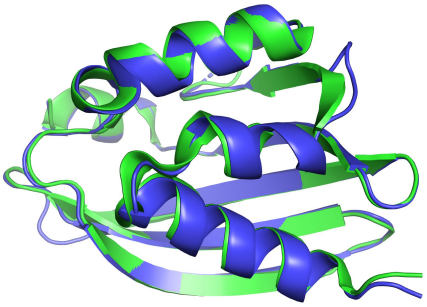
VFN Experiment



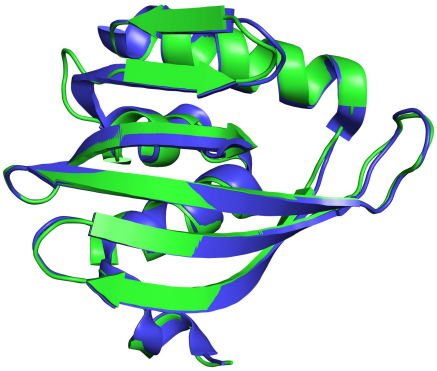
2R0S
r.m.s.d. = 0.562 Å recovery = 61.40 %



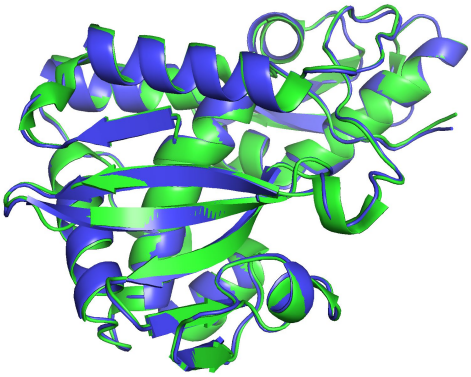
4LQB
r.m.s.d. = 0.601 Å recovery = 70.76 %



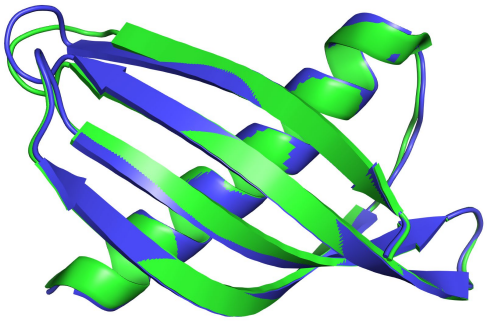
3LYG
r.m.s.d. = 0.431 Å recovery = 58.33 %



4U13
r.m.s.d. = 0.443 Å recovery = 49.54 %



4Z8S
r.m.s.d. = 0.404 Å recovery = 56.28 %



4N6T
r.m.s.d. = 0.273 Å recovery = 49.36 %

VFN Experiment