PIFOLD: TOWARD EFFECTIVE AND EFFICIENT PROTEIN INVERSE FOLDING

Zhangyang Gao[†], **Cheng Tan**[†], **Stan Z. Li**[*]
AI Lab, Research Center for Industries of the Future, Westlake University
{gaozhangyang, tancheng,Stan.ZQ.Li}@westlake.edu.cn

ABSTRACT

How can we design protein sequences that can be folded into a desired structure effectively and efficiently? Research on AI methods for structure-based protein design has attracted increasing attention in recent years; however, few can achieve good performance in both accuracy and efficiency simultaneously due to the lack of expressive features and the adoption of autoregressive sequence decoding approach. To address these issues, we propose PiFold in a one-shot way with improved recovery and high inference speed. PiFold contains a novel residue featurizer for richer representation and novel non-autoregressive decoder composed of multi-PiGNN layers for generating protein sequences. Experiments show that PiFold achieves 51.66% recovery on CATH 4.2, and inference speed is 70 times faster than the autoregressive competitors for proteins of average length 1632. In addition, it achieves 58.72% and 60.42% recovery scores on TS50 and TS500, respectively. We conduct comprehensive ablation studies to reveal the significance of different types of protein features and model design strategies, inspiring directions for future research. The PyTorch code is available at GitHub.

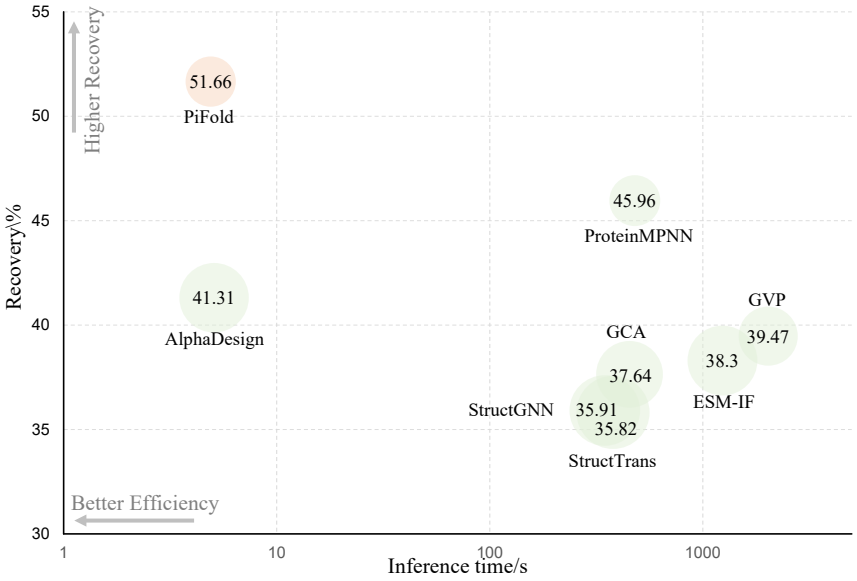


Figure 1: Performance comparison with other graph-based protein design methods. The recovery scores, the inference time costs, and the perplexities are shown in the Y-axis direction, the X-axis direction, and the circle size, respectively. Note that the recovery and perplexity results in the CATH dataset are reported without using any other training data. The inference time is averaged over 100 long protein sequences of average length 1632 on an NVIDIA V100.

1 INTRODUCTION

Proteins are linear chains of amino acids that fold into 3D structures to control cellular processes, such as transcription, translation, signaling, and cell cycle control. Creating novel proteins could

[†]Equal Contribution, [*]Corresponding Author.

deepen our understanding of living systems and facilitate drug design. One crucial problem is designing protein sequences that can fold to desired structures, namely structure-based protein design (Pabo, 1983).

Recently, deep learning models have been proposed to solve this problem (Li et al., 2014; Wu et al., 2021; Pearce & Zhang, 2021; Ovchinnikov & Huang, 2021; Ding et al., 2022; Gao et al., 2020; Dauparas et al., 2022; Ingraham et al., 2019; Jing et al., 2020; Tan et al., 2022c; Hsu et al., 2022; O'Connell et al., 2018; Wang et al., 2018; Qi & Zhang, 2020; Strokach et al., 2020; Chen et al., 2019; Zhang et al., 2020a; Huang et al., 2017; Anand et al., 2022; Strokach & Kim, 2022; Li & Koehl, 2014; Greener et al., 2018; Karimi et al., 2020; Anishchenko et al., 2021; Cao et al., 2021; Liu et al., 2022; McPartlon et al., 2022; Huang et al., 2022; Dumortier et al., 2022; Li et al., 2022a; Maguire et al., 2021; Li et al., 2022b). Among these graph-based models have achieved the most impressive progress. Despite this, none of the graph models could achieve 50+% sequence recovery on the CATH benchmark. Moreover, all of them adopt the autoregressive generation, resulting in low inference speed (Dauparas et al., 2022; Ingraham et al., 2019; Jing et al., 2020; Hsu et al., 2022; Tan et al., 2022c; Hsu et al., 2022).

For years, graph-based models have striven to learn expressive residue representations through better feature engineering, more elaborate modeling, and larger training datasets. For example, AlphaDesign (Gao et al., 2022a) and ProteinMPNN (Dauparas et al., 2022) point out that additional angle and distance features could significantly improve the representation quality. GraphTrans (Ingraham et al., 2019), GVP (Jing et al., 2020), and GCA (Tan et al., 2022c) introduce new modules that incorporate graph-based message passing, equivariant features, and global attention to learn geometric representations from residue interactions. ESM-IF (Hsu et al., 2022) incorporates additional training data to capture more structural learning bias.

Although they have made significant progress, three problems remain to be solved: (1) How to construct effective features so as to learn richer residual representations? (2) How to design a more efficient decoder to achieve high inference speed? (3) How to design benchmarks for fair evaluations.

Interestingly, few studies have attempted to improve the inference speed, perhaps because an efficiency gain has to sacrifice accuracy (Bahdanau et al., 2015; Vaswani et al., 2017; Ghazvininejad et al., 2019; Geng et al., 2021; Xie et al., 2020; Wang et al., 2019; Gu et al., 2018), while the accuracy is generally considered to be more important than efficiency in protein design. To address this dilemma, AlphaDesign (Gao et al., 2022a) proposes a parallel self-correcting module, attempting to speed up inference and keep the recovery simultaneously. Nevertheless, it still requires two iterations for prediction and sacrifices a little drop in recovery.

In this paper, we aim to simultaneously improve the accuracy and efficiency with a simple model containing. We propose PiFold (**p**rotein **i**nverse **f**olding) to address the above problems. PiFold consists of a novel residue featurizer and stacked PiGNNs. The featurizer adds virtual atoms as supplements, with learnable parameters, to the real atoms of residues and constructs richer residue features for more powerful representations of the residues. The PiGNN introduces global attention that learns more expressive residue representation than baselines through multi-scale interactions. For the decoder part, we could completely remove the autoregressive decoder by stacking more PiGNN layers without sacrificing accuracy.

Experiments show that PiFold can achieve state-of-the-art recoveries on several real-world datasets, i.e., 51.66% on CATH 4.2, 58.72 on TS50, and 60.42% on TS500. PiFold is the first graph model that could exceed 55% recovery on TS50 and 60% recovery on TS500. Furthermore, PiFold's inference efficiency in designing long proteins is improved by 70+ times compared to autoregressive competitors. We will provide comprehensive ablation studies.

In summary, our contributions include the following:

1. the novel residue featurizer to construct comprehensive residue features and learn virtual atoms to capture complementary information to real atoms;
2. the PiGNN layer that learns multi-scale residue interactions for more powerful representations;
3. the non-autoregressive decoder to speed up without sacrificing accuracy; and
4. a comprehensive comparison of advanced graph models in real-world datasets, including CATH, TS50, and TS500.

2 RELATED WORKS

Recently, AI has brought about significant advances in many fields (Gao et al., 2022d; Cao et al., 2022; Tan et al., 2022a; Li et al., 2022c; He et al., 2020; Stärk et al., 2022). Notably, AI-based methods have well solved the protein folding problem (Jumper et al., 2021; Wu et al., 2022; Lin et al., 2022; Mirdita et al., 2022; Wang et al., 2022; Li et al., 2022d) that scientists have spent decades tackling. Its inverse problem, structure-based protein design, is receiving increasing attention.

Problem definition The structure-based protein design aims to find the amino acids sequence $\mathcal{S} = \{s_i : 1 \leq i \leq n\}$ that can fold into the desired structure $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3 : 1 \leq i \leq n\}$, where n is the number of residues and $s_i \in \{1, \ldots, 20\}$ indexes the 20 types of amino acids. Formally, an AI-based solution aims to learn a function $\mathcal{F}_\theta$:

$$\mathcal{F}_\theta : \mathcal{X} \mapsto \mathcal{S}. \tag{1}$$

Note that because homologous proteins always share similar structures (Pearson & Sierk, 2005), the solution may not be unique (Gao et al., 2020).

Previous works adopt the paradigm of combining a 3D structural encoder and an autoregressive 1D sequence decoder for sequence generation, where the autoregressive decoder takes the form:

$$p(S|X;\theta) = \prod_{t=1}^{n} p(s_t|s_{<t}, X; \theta). \tag{2}$$

This is bound to be inefficient. Existing works can be categorized into MLP-based, CNN-based, and GNN-based ones in terms of types of structural encoders (Gao et al., 2022a).

MLP-based models MLP is used to predict the probability of 20 amino acids for each residue, and various methods are mainly difficult in feature construction. These methods are commonly evaluated on the TS50, which contains 50 native structures. For example, SPIN (Li et al., 2014) achieves 30% recovery on TS50 by using torsion angles (ϕ and ψ), sequence profiles, and energy profiles. Through adding backbone angles (θ and τ), local contact number, and neighborhood distance, SPIN2 (O'Connell et al., 2018) improves the recovery to 34%. Wang's model (Wang et al., 2018) suggests using backbone dihedrals (ϕ, ψ and ω), the solvent accessible surface area of backbone atoms (C_α, N, C, and O), secondary structure types (helix, sheet, loop), $C_\alpha - C_\alpha$ distance and unit direction vectors of $C_\alpha - C_\alpha$, $C_\alpha - N$ and $C_\alpha - C$ and achieves 33% recovery. The MLP method enjoys a high inference speed, but suffers from a low recovery rate because the structural information is not sufficiently considered.

CNN-based models These methods use 2D CNN or 3d CNN to extract protein features (Torng & Altman, 2017; Boomsma & Frellsen, 2017; Weiler et al., 2018; Zhang et al., 2020a; Qi & Zhang, 2020; Chen et al., 2019) and are commonly evaluated on the TS50 and TS500. SPROF (Chen et al., 2019) adopts 2D CNN to learn residue representations from the distance matrix and achieves a 40.25% recovery on TS500. 3D CNN-based methods, such as ProDCoNN (Zhang et al., 2020a) and DenseCPD (Qi & Zhang, 2020), extract residue features from the atom distribution in a three-dimensional grid box. For each residue, after being translated and rotated to a standard position, the atomic distribution is fed to the model to learn translation- and rotation-invariant features. ProDCoNN (Zhang et al., 2020a) designs a nine-layer 3D CNN with multi-scale convolution kernels and achieves 42.2% recovery on TS500. DenseCPD (Qi & Zhang, 2020) uses the DensetNet architecture (Huang et al., 2017) to boost the recovery to 55.53% on TS500. Recent works (Anand et al., 2022) have also explored the potential of deep models to generalize to *de novo* proteins. Despite the improved recovery achieved by the 3D CNN models, its inference is slow, probably because they require separate preprocessing and prediction for each residue.

Graph-based models These methods use k-NN graph to represent the 3D structure and employ graph neural networks (Defferrard et al., 2016; Kipf & Welling, 2016; Veličković et al., 2017; Zhou et al., 2020; Zhang et al., 2020b; Gao et al., 2022b; Tan et al., 2022b; Gao et al., 2022c) to extract residue features while considering structural constraints. The protein graph encodes residue information and pairwise interactions as the node and edge features. GraphTrans (Ingraham et al., 2019) uses the graph attention encoder and autoregressive decoder for protein design. GVP (Jing et al., 2020) proposes geometric vector perceptrons to learn from both scalar and vector features. GCA (Tan et al., 2022c) introduces global graph attention for learning contextual features. In addition, ProteinSolver (Strokach et al., 2020) is developed for scenarios where partial sequences are known while does not report results on standard benchmarks. Recently, AlphaDesign (Gao et al.,

2022a), ProteinMPNN (Dauparas et al., 2022) and Inverse Folding (Hsu et al., 2022) achieve significant improvements. Compared to CNN methods, graph models do not require rotating each residue separately as in CNN, thus improving the training efficiency. Compared to MLP methods, the well-exploited structural information helps GNN obtain higher recovery. In summary, GNN can achieve a good balance between efficiency and accuracy.

3 METHOD

3.1 OVERALL FRAMEWORK

The overall PiFold framework is shown in Figure.2, where the input is the desired protein structure, and the output is the generated protein sequence. The residue featurizer and PiGNN layer are aimed to learn expressive residue representations. As described above, the residue featurizer constructs comprehensive residue features and learns virtual atoms to capture information complementary to real atoms. The PiGNN learns multi-scale node representations and interactions, that is, local edge features and global attention features. PiFold could generate protein sequences in a one-shot manner with a higher recovery than previous autoregressive (Ingraham et al., 2019; Jing et al., 2020; Tan et al., 2022c; Dauparas et al., 2022; Hsu et al., 2022) or iterative models (Gao et al., 2022a).

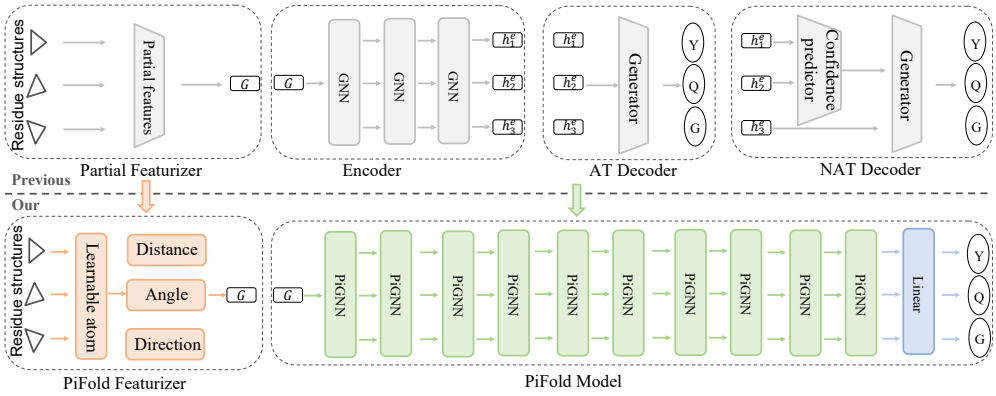


Figure 2: The PiFold framework. The featurizer module extracts comprehensive residue features from the fixed real atoms and learnable virtual atoms. Ten layers of PiGNNs learn expressive residue representations by integrating node-level, edge-level, and global-level residue interactions. PiFold generates protein sequences in a one-shot manner, without using autoregressive or iterative decoders.

3.2 THE FEATURIZER MODULE

The featurizer module is a med to construct a graph to represent a protein structure composed of residue node and edge features (see Fig.3 for geometric feature construction).

The protein graph structure A protein is represented as a k-NN graph derived from its residues, describing the 3D residue dependencies. The protein graph $\mathcal{G}(A, X, E)$ consists of the adjacency matrix $A \in \{0,1\}^{n,n}$, node features $X \in \mathbb{R}^{n,f_n}$, and edge features $E \in \mathbb{R}^{m,f_e}$. Note that n and m are the numbers of nodes and edges, and we construct f_n node features and f_e edge features considering the inductive bias of residues' stable structure, order, and coordinates. Since each residue consists of C_α, C, N and O, we could construct local coordinate system $Q_i = [\boldsymbol{b}_i, \boldsymbol{n}_i, \boldsymbol{b}_i \times \boldsymbol{n}_i]$ for residue i, where $\boldsymbol{u}_i = C_{\alpha_i} - N_i, \boldsymbol{v}_i = C_i - C_{\alpha_i}, \boldsymbol{b}_i = \frac{\boldsymbol{u}_i - \boldsymbol{v}_i}{||\boldsymbol{u}_i - \boldsymbol{v}_i||}$, and $\boldsymbol{n}_i = \frac{\boldsymbol{u}_i \times \boldsymbol{v}_i}{||\boldsymbol{u}_i \times \boldsymbol{v}_i||}$. Based on these coordinate systems, we could construct rotation- and translation-invariant features for single or pairs of residues, including distance, angle, and direction; refer to Fig.3.

Distance features Given atom pairs A and B, the distance feature is RBF($||A - B||$), where RBF is a radial basis function. Note that $A \in \{C_i, C_{\alpha_i}, N_i, O_i\}, B \in \{C_i, C_{\alpha_i}, N_i, O_i\}$ when computing features for residue i, while $A \in \{C_i, C_{\alpha_i}, N_i, O_i\}$ and $B \in \{C_j, C_{\alpha_j}, N_j, O_j\}$ when computing features for residue pair (i, j). Careful readers should notice that the number of features is proportional to the square of the number of atoms, which will be further analyzed and simplified in Table.6. Beyond the distances between real atoms, we also supplement virtual atoms $\{V_i^1, V_i^2, V_i^3, \cdots\}$ for residue i. The additional node and edge features are generated by sampling A and B from $\{V_i^1, V_i^2, V_i^3, \cdots\} \times \{V_i^1, V_i^2, V_i^3, \cdots\}$ and $\{V_i^1, V_i^2, V_i^3, \cdots\} \times \{V_j^1, V_j^2, V_j^3, \cdots\}$, respectively. To ensure that the virtual atoms are translation- and rotation-equivariant to the protein

structure, the model is required to learn their relative position in each local coordinate system. The relative position vectors are shared across all residues, thus forcing the model to learn consensus knowledge to extend to general scenarios. We normalize the relative coordinates by limiting their length to a unit vector for stable training. Formally, denoting the relative position of the k-th virtual atom is (x_k, y_k, z_k), we compute the absolute coordinates of V_i^k by

$$\begin{cases} V_i^k = [x_k \boldsymbol{b}_i, y_k \boldsymbol{n}_i, z_k(\boldsymbol{b}_i \times \boldsymbol{n}_i)] + C_{\alpha_i} \\ x_k^2 + y_k^2 + z_k^2 = 1, \end{cases} \tag{3}$$

where x_k, y_k, z_k are learnable parameters.

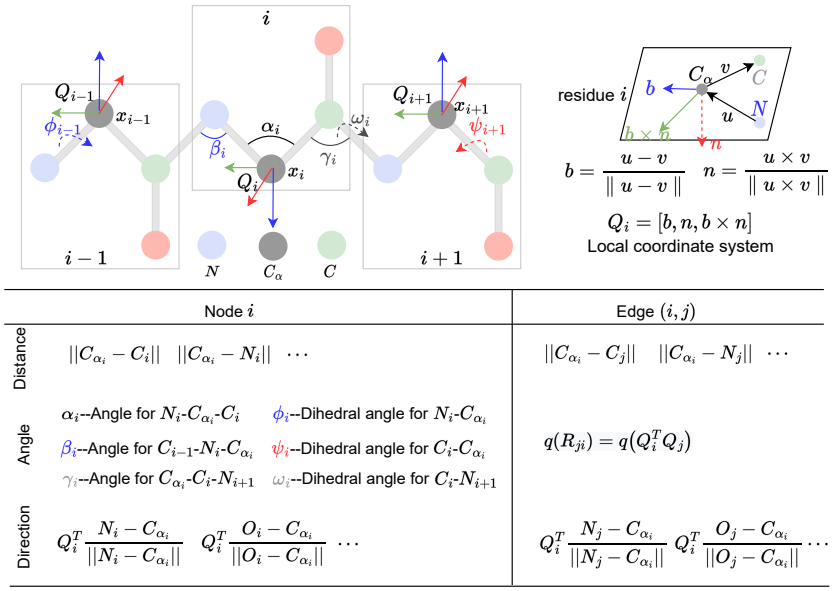


Figure 3: PiFold featurizer. Distance, angle, and direction features are constructed for every single residue and between each pair of them, all invariant to rotation and translation. Learnable virtual atoms are added to real atoms to provide complementary distance constraints.

Angle features As shown in Figure.3, we use the bond angles $(\alpha_i, \beta_i, \gamma_i)$ and torsion angles $(\phi_i, \psi_i, \omega_i)$ as node features considering the backbone order and local geometry between adjacent residues. For residue pairs (i, j), the angular features represent the quaternions of relative rotation between their local coordinate systems, written as $q_{ij} = q(Q_i^T Q_j)$, where q_{ij} is the quaternion encoding function.

Direction features For single residue i, we use the directions of inner atoms to C_{α_i} as node features. For example, the direction feature of N_i is $Q_i^T \frac{N_i - C_{\alpha_i}}{||N_i - C_{\alpha_i}||}$. Similarly, for pairs of residues (i, j), the directions include all atoms of residue j to the C_{α_i}, and the direction of N_j is $Q_i^T \frac{N_j - C_{\alpha_i}}{||N_j - C_{\alpha_i}||}$. Note that we use the relative direction concerning the local coordinate system Q_i to make the features invariant in rotation.

3.3 PiGNN

We propose the PiGNN layer to learn geometric residue representations considering multi-scale residue interactions in node, edge, and global context levels.

Local node interactions We use the simplified graph transformer (Gao et al., 2022a) to update node representations, where the model learns multi-headed attention weights via a MLP instead of separately computing Q and K. Formally, we take $\boldsymbol{h}_i^l$ and $\boldsymbol{e}_{ji}^l$ as the output feature vectors of node i and edge $j \to i$ in layer l. Before the first GNN layer, we use MLPs to project all input features to the d-dimensional space, thus $\boldsymbol{h}_i^0 \in \mathbb{R}^d$ and $\boldsymbol{e}_{ji}^0 \in \mathbb{R}^d$. When considering the attention mechanisms centered in node i, the attention weight a_{ji} at the $l+1$ layer is calculated by:

$$\begin{cases} w_{ji} = \text{AttMLP}(\boldsymbol{h}_j^l || \boldsymbol{e}_{ji}^l || \boldsymbol{h}_i^l) \\ a_{ji} = \frac{\exp w_{ji}}{\sum_{k \in \mathcal{N}_i} \exp w_{ki}} \end{cases} \tag{4}$$

where $\mathcal{N}_i$ is the neighborhood system of node i and $||$ means the concatenation operation. The node feature $\boldsymbol{h}_i^l$ is updated by:

$$\begin{cases} \boldsymbol{v}_j = \text{NodeMLP}(\boldsymbol{e}_{ji}^l || \boldsymbol{h}_j^l) \\ \hat{\boldsymbol{h}}_i^{l+1} = \sum_{j \in \mathcal{N}_i} a_{ji} \boldsymbol{v}_j \end{cases} \tag{5}$$

Local edge interactions The protein graph is an attributed graph containing both node and edge features; however, the simplified graph transformer is a node-centered network, which does not iteratively update edge features. We find that this neglect will lead to suboptimal representations, and introducing the edge updating layer could improve the model capability:

$$\boldsymbol{e}_{ji}^{l+1} = \text{EdgeMLP}(\hat{\boldsymbol{h}}_j^l || \boldsymbol{e}_{ji}^l || \hat{\boldsymbol{h}}_i^l) \tag{6}$$

Global context attention While the local interactions play a crucial role in learning residue representations (Ingraham et al., 2019; Jing et al., 2020; Gao et al., 2022a), the global information (Tan et al., 2022c) is also proved to be valuable for improving protein design. However, the time complexity of global attention across the whole protein is proportional to the square of the protein length, which significantly increases the computational overhead. To simultaneously enjoy the improved recovery and with good efficiency, we suggest learning a global context vector for each protein and using it to apply gated attention for node representations:

$$\begin{cases} \boldsymbol{c}_i = \text{Mean}(\{\hat{\boldsymbol{h}}_k^{l+1}\}_{k \in \mathcal{B}_i}) \\ \boldsymbol{h}_i^{l+1} = \hat{\boldsymbol{h}}_i^{l+1} \odot \sigma(\text{GateMLP}(\boldsymbol{c}_i)) \end{cases} \tag{7}$$

where $\mathcal{B}_i$ is the index set of residues belonging to the same protein as residue i, $\odot$ is element-wise product operation, and $\sigma(\cdot)$ is the sigmoid function. The computational cost of the global context attention is linear to the residue numbers.

4 EXPERIMENTS

We comprehensively evaluate PiFold on different datasets and show its superior accuracy and efficiency. Detailed ablation studies are applied to deepen the understanding of PiFold. In summary, we will answer the following questions:

- **Performance (Q1):** Could PiFold achieve SOTA accuracy and efficiency on CATH dataset?
- **Ablation (Q2):** How much gain can be obtained for each improvement?
- **Benchmarking (Q3):** Could PiFold be extended to more benchmarks, such as TS50 and TS500?

4.1 PERFORMANCE (Q1)

Objective & Setting Could PiFold achieve SOTA recovery and efficiency on real-world datasets? Does the efficiency improvement require sacrificing the generative quality? To answer these questions, we compare PiFold against recent strong baselines on the CATH (Orengo et al., 1997) dataset. We use the same data splitting as GraphTrans (Ingraham et al., 2019) and GVP (Jing et al., 2020), where proteins are partitioned by the CATH topology classification, resulting in 18024 proteins for training, 608 proteins for validation, and 1120 proteins for testing. We stack ten layers of PiGNN to construct the PiFold model with hidden dimension 128. The model is trained up to 100 epochs by default using the Adam optimizer on NVIDIA V100s. The k for k-NN graph defaults to 30. The batch size and learning rate used for training are 8 and 0.001, respectively. For studying the generative quality, we report perplexity and median recovery scores on short-chain, single-chain, and all-chain settings. Regarding efficiency, we report the model inference time when generating 100 long chains with an average length of 1632.

Baselines We compare PiFold with recent graph models, including StructGNN, StructTrans (Ingraham et al., 2019), GCA (Tan et al., 2022c), GVP (Jing et al., 2020), GVP-large, AlphaDesign (Gao et al., 2022a), ESM-IF (Hsu et al., 2022), ProteinMPNN (Dauparas et al., 2022), since most of them are open-source. To make a fair and reliable comparison, we reproduce StructGNN, StructTrans, GCA, GVP, AlphaDesign, and ProteinMPNN under the same data splitting on CATH 4.2. The relevant results were generally consistent with their papers. Because GVP-large and ESM-IF do not provide training scripts and their code framework is different from other methods, we have difficulty reproducing them and therefore copy the results from their manuscripts. It should be reminded that ESM-IF uses CATH 4.3 for training, which may lead to incomparable recovery and

perplexity due to the different data volumes. MLP and CNN-based methods are ignored here because most of them do not report results on CATH nor provide publicly available codes, and we will discuss them in Section.4.3. For more baseline details, we suggest the reader refer to the Appendix.

Table 1: Results comparison on the CATH dataset. All baselines are reproduced under the same code framework, except ones marked with †. We copy results of GVP-large and ESM-IF from their manuscripts (Hsu et al., 2022). The **best** and suboptimal results are labeled with bold and underline.

Model	Perplexity ↓			Recovery % ↑			CATH version	
	Short	Single-chain	All	Short	Single-chain	All	4.2	4.3
StructGNN	8.29	8.74	6.40	29.44	28.26	35.91	✓	
GraphTrans	8.39	8.83	6.63	28.14	28.46	35.82	✓	
GCA	7.09	7.49	6.05	32.62	31.10	37.64	✓	
GVP	7.23	7.84	5.36	30.60	28.95	39.47	✓	
GVP-large†	7.68	**6.12**	6.17	32.6	**39.4**	39.2		✓
AlphaDesign	7.32	7.63	6.30	34.16	32.66	41.31	✓	
ESM-IF†	8.18	6.33	6.44	31.3	38.5	38.3		✓
ProteinMPNN	6.21	6.68	4.61	36.35	34.43	45.96	✓	
PiFold	**6.04**	6.31	**4.55**	**39.84**	38.53	**51.66**	✓	

Recovery We report the perplexity and recovery score in Table.1, where two subsets of the full test set are also considered, following previous works (Ingraham et al., 2019; Jing et al., 2020; Tan et al., 2022c; Hsu et al., 2022). The "Short" set contains proteins up to length 100, and the "Single chain" contains proteins recorded as single chains in the Protein Data Bank. We observe that the proposed PiFold could consistently improve the perplexity (lower is better) and recovery score (higher is better) across different test sets. On the "Short" and "All" datasets, PiFold achieves the best perplexities and recovery scores, where the recovery improvement are 3.49% and 1.94%, respectively. On the "Single-chain" dataset, PiFold is the best model when trained on CATH 4.2.

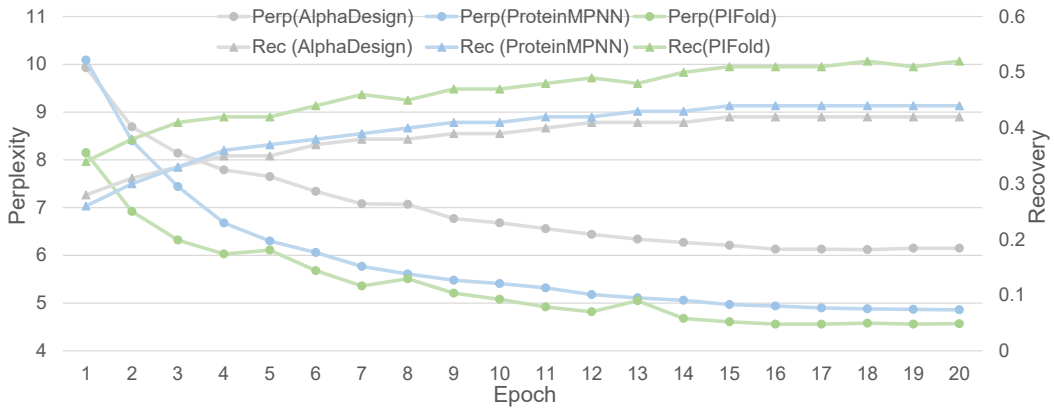


Figure 4: Training PiFold for 20 epochs is enough to achieve SOTA performance.

Efficiency We already know that PiFold can achieve higher recovery; but does this improvement come at the expense of efficiency? We further assess the training and inference time costs of PiFold and the competitive baselines (AlphaDesign and ProteinMPNN). PiFold could achieve state-of-the-art perplexity and recovery during the training phase with fewer epochs. As shown in Figure.4, 20 epochs are enough for PiFold. As to the inference speed, all baselines except AlphaDesign adopt an autoregressive decoder to generate residues one by one, and the computational complexity is O(L) to the protein length L. However, PiFold enjoys O(1) computational complexity due to the one-shot generative schema. We benchmark the inference speed of different graph models on 100 long chains with an average length of 1632 and show the results in Fig.1, where PiFold is 70 times faster than autoregressive competitors, including ProteinMPNN, ESM-IF, GCA, StructGNN, StructTrans, GVP.

4.2 ABLATION (Q2)

Objective & Setting How much gain could be obtained from the introduced features and modules? Will removing the autoregressive decoder reduce performance? We conduct comprehensive ablation studies for the features and modules to answer these questions. All models are trained up to 20 epochs with a learning rate of 0.001 using the Adam optimizer and OneCycle scheduler.

Table 2: Ablation studies of feature types. The lower the ranking number, the higher the recovery rate and the model performs better.

	Model	PiFold	model 1	model 2	model 3	model 4	model 5	model 6
Node	Distance	✓		✓	✓	✓	✓	✓
	Angle	✓	✓		✓	✓	✓	✓
	Direction	✓	✓	✓		✓	✓	✓
Edge	Distance	✓	✓	✓	✓		✓	✓
	Angle	✓	✓	✓	✓	✓		✓
	Direction	✓	✓	✓	✓	✓	✓	
Results	Perplexity ↓	4.55	4.59	4.62	4.53	**4.80**	4.54	4.65
	Recovery ↑	51.66	51.49	51.05	51.61	**49.61**	51.51	50.86
Summary	Rank	–	5	3	6	**1**	4	2

Feature type ablation We determine the contribution rank of different types of features in Table.2. We find that edge features are more important than node features, suggesting that amino acid types may be inferred primarily from residue interactions. Another discovery is that edge distances are the essential features; however, the number of distances is proportional to the square of the atom numbers, which may lead to feature explosion and redundancy. We further analyze the role of each edge distance in Table.6, where the "C_α-O" and "C-N" distances are the most significant ones. In addition, the ablation study in Table.6 also demonstrates that virtual atoms can lead to performance gains, and the more virtual atoms, the higher recovery. One possible explanation is that virtual atoms can learn valuable information complementary to backbone atoms, e.g., side chain direction.

Table 3: Ablation of PiGNN modules.

		PiFold	model 1	model 2	model 3	model 4	model 5
Node	GCN		✓				
	GAT			✓			
	QKV				✓		
	AttMLP	✓				✓	✓
Edge	UpdateEdge	✓	✓	✓	✓	✓	
Global	ContextAtt	✓	✓	✓	✓		
Results	Perplexity	4.55	4.74	4.96	4.54	4.62	4.76
	Recovery	51.66	49.65	45.78	50.74	51.22	50.00
Summary	Change	–	↓↓	↓↓↓	↓↓	↓	↓↓

Model ablation We conduct systematic experiments to determine whether modules of PiGNN are effective and reveal their relative importance. At to the node message passing module, we compare the node message passing layer (AttMLP) against GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017) and QKV-based attention layer (Ingraham et al., 2019; Tan et al., 2022c; Dauparas et al., 2022). We also investigate whether the edge updating operation and global context attention could contribute to the model performance. As shown in Table.3, we find that the AttMLP-based messaging mechanism can improve the recovery by at least 0.9%. In addition, the global context attention and edge updating could improve the recovery by 0.44% and 1.22%, respectively.

Table 4: Autoregressive ablation. "Enc" and "AT" indicate the encoder's and autoregressive layers' numbers.

	PiFold	model 1	model 2	model 3	model 4	model 5	model 6
Enc	10	3	3	4	5	6	8
AT	0	2	3	2	1	0	0
Perplexity ↓	4.55	5.06	5.04	4.92	4.83	4.58	4.53
Recovery ↑	51.66	49.30	49.37	49.60	50.41	50.96	51.22
Test Time ↓	36s	527s	707s	522s	347s	30s	34s

AT ablation Could PiFold remove the autoregressive mechanism and still ensure good performance? We further explore the effectiveness of the autoregressive decoder (Ingraham et al., 2019) and one-shot PiGNN encoder. As shown in Table.4, by gradually replacing the autoregressive decoder with the one-shot PiGNN (model 1 → model 5), the recovery will improve. At the same time,

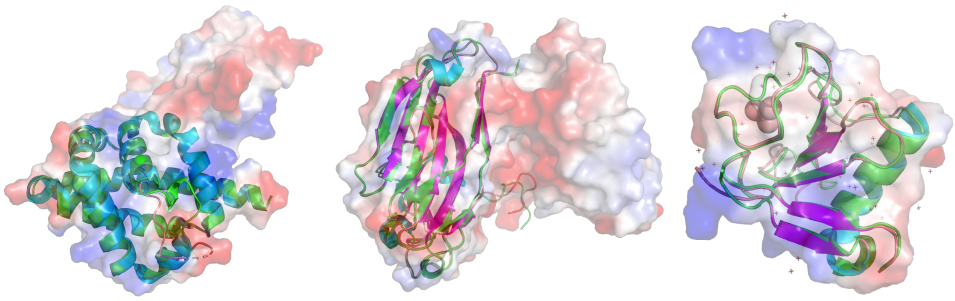
the inference time cost on CATH will reduce. This phenomenon suggests that the expressive encoder is more important than the autoregressive decoder. The recovery score can be further improved by using a deeper encoder (model 5 → model 6 → PiFold).

4.3 GENERALIZATION (Q3)

Table 5: Results on TS50 and TS500. All baselines are reproduced under the same code framework, except ones marked with †, whose results are copied from their manuscripts. The **best** and suboptimal results are labeled with bold and underline.

Group	Model	TS50			TS500		
		Perplexity ↓	Recovery ↑	Worst ↑	Perplexity ↓	Recovery ↑	Worst ↑
MLP	SPIN †		30.30			30.30	
	SPIN2 †		33.60			36.60	
	Wang's model †		33.00			36.14	
CNN	SPROF †		39.16			40.25	
	ProDCoNN †		40.69			42.20	
	DenseCPD †		50.71			55.53	
Graph	StructGNN	5.40	43.89	26.92	4.98	45.69	0.05
	GraphTrans	5.60	42.20	29.22	5.16	44.66	0.03
	GVP	4.71	44.14	33.73	4.20	49.14	**0.09**
	GCA	5.09	47.02	28.87	4.72	47.74	0.03
	AlphaDesign	5.25	48.36	32.31	4.93	49.23	0.03
	ProteinMPNN	3.93	54.43	37.24	3.53	58.08	0.03
	PiFold (our)	**3.86**	**58.72**	**37.93**	**3.44**	**60.42**	0.03

Objective & Results To present a comprehensive comparison and assess the generalizability of PiFold across different data sets, we report results on TS50 and TS500 in Table.5. We reproduce graph-based models and show the perplexity, median recovery score, and worst recovery. For MLP and CNN models, the results are copied from their manuscripts. We summarize that PiFold could achieve consistent improvements across TS50 and TS500. To our knowledge, PiFold is the first graph model that could exceed 55% recovery on TS50 and 60% on TS500. Finally, in Figure. 5, we show the potential of PiFold in designing all-alpha, all-beta, and mixed proteins.



All-alpha structure (3tld) (Recovery=56.25, RMSD=0.87) | All-beta structure (2giy) (Recovery=54.19, RMSD=0.35) | Mixed structure (1mgr) (Recovery=47.42, RMSD=0.43)

Figure 5: Visual examples. For native structures, we color Helix, Sheet, and Loop with cyan, magenta, and orange, respectively. We use AlphaFold2 to predict the structures of designed sequences, marked in green. We provide the recovery score and structural RMSD relative to the ground truth proteins.

5 CONCLUSION

PiFold, as an AI design method for structure-based protein design, significantly improves the recovery and greatly increases the efficiency simultaneously, by using the proposed protein featurizer and PiGNN. It achieves 51.66%, 58.72%, and 60.42% recovery scores on CATH 4.2, TS50, and TS500 test sets, respectively, and is 70 times faster in inference speed than autoregressive competitors, achieving SOTA performance. We look forward to validating the proposed PiFold method through wet experiments in future work.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive and helpful reviews. This work was supported by the National Key R&D Program of China (Project 2022ZD0115100), the National Natural Science Foundation of China (Project U21A20427), and the Competitive Research Fund (Project WU2022A009) from the Westlake Center for Synthetic Biology and Integrated Bioengineering.

REFERENCES

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):1–11, 2022.

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

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015, 2015.

Wouter Boomsma and Jes Frellsen. Spherical convolutions and their application in molecular modelling. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf.

Hanqun Cao, Cheng Tan, Zhangyang Gao, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion model. arXiv preprint arXiv:2209.02646, 2022.

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, pp. 1261–1271. PMLR, 2021.

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.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning based protein sequence design using proteinmpnn. bioRxiv, 2022.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29:3844–3852, 2016.

Wenze Ding, Kenta Nakai, and Haipeng Gong. Protein design via deep learning. Briefings in bioinformatics, 23(3):bbac102, 2022.

Baldwin Dumortier, Antoine Liutkus, Clément Carré, and Gabriel Krouk. Petribert: Augmenting bert with tridimensional encoding for inverse protein folding and design. bioRxiv, 2022.

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

Zhangyang Gao, Cheng Tan, Stan Li, et al. Alphadesign: A graph protein design method and benchmark on alphafolddb. arXiv preprint arXiv:2202.01079, 2022a.

Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Cosp: Co-supervised pretraining of pocket and ligand. arXiv preprint arXiv:2206.12241, 2022b.

Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Semiretro: Semi-template framework boosts deep retrosynthesis prediction. arXiv preprint arXiv:2202.08205, 2022c.

Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3170–3180, 2022d.

Xinwei Geng, Xiaocheng Feng, and Bing Qin. Learning to rewrite for non-autoregressive neural machine translation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3297–3308, 2021.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 6112–6121, 2019.

Joe G Greener, Lewis Moffat, and David T Jones. Design of metalloproteins and novel protein folds using variational autoencoders. Scientific reports, 8(1):1–12, 2018.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. In International Conference on Learning Representations, 2018.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9729–9738, 2020.

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. bioRxiv, 2022.

Bin Huang, Tingwe Fan, Kaiyue Wang, Haicang Zhang, Chungong Yu, Shuyu Nie, Yangshuo Qi, Wei-Mou Zheng, Jian Han, Zheng Fan, et al. Accurate and efficient protein sequence design through learning concise local environment of residues. bioRxiv, 2022.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708, 2017.

John Ingraham, Vikas K Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. 2019.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. arXiv preprint arXiv:2009.01411, 2020.

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

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.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

Alex J Li, Mindren Lu, Israel Tilahun 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. bioRxiv, 2022a.

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, 2022b.

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.

Siyuan Li, Zedong Wang, Zicheng Liu, Di Wu, and Stan Z Li. Openmixup: Open mixup toolbox and benchmark for visual representation learning. arXiv preprint arXiv:2209.04851, 2022c.

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.

Ziyao Li, Xuyang Liu, Weijie Chen, Fan Shen, Hangrui Bi, Guolin Ke, and Linfeng Zhang. Unifold: An open-source platform for developing protein folding models beyond alphafold. bioRxiv, 2022d.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. bioRxiv, 2022.

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. 2022.

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.

Matt McPartlon, Ben Lai, and Jinbo Xu. A deep se (3)-equivariant model for learning inverse protein folding. bioRxiv, 2022.

Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. Nature Methods, pp. 1–4, 2022.

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.

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

Sergey Ovchinnikov and Po-Ssu Huang. Structure-based protein design with deep learning. Current opinion in chemical biology, 65:136–144, 2021.

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

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.

William R Pearson and Michael L Sierk. The limits of protein sequence comparison? Current opinion in structural biology, 15(3):254–260, 2005.

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.

Hannes Stärk, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In International Conference on Machine Learning, pp. 20503–20521. PMLR, 2022.

Alexey Strokach and Philip M Kim. Deep generative modeling for protein design. Current opinion in structural biology, 72:226–236, 2022.

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.

Cheng Tan, Zhangyang Gao, and Stan Z Li. Simvp: Towards simple yet powerful spatiotemporal predictive learning. arXiv preprint arXiv:2211.12509, 2022a.

Cheng Tan, Zhangyang Gao, and Stan Z Li. Target-aware molecular graph generation. arXiv preprint arXiv:2202.04829, 2022b.

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

Wen Torng and Russ B Altman. 3d deep convolutional neural networks for amino acid environment similarity analysis. BMC bioinformatics, 18(1):1–23, 2017.

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.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

Guoxia Wang, Xiaomin Fang, Zhihua Wu, Yiqun Liu, Yang Xue, Yingfei Xiang, Dianhai Yu, Fan Wang, and Yanjun Ma. Helixfold: An efficient implementation of alphafold2 using paddlepaddle. arXiv preprint arXiv:2207.05477, 2022.

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.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. Non-autoregressive machine translation with auxiliary regularization. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 5377–5384, 2019.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/488e4104520c6aab692863cc1dba45af-Paper.pdf.

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

Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. Current opinion in chemical biology, 65:18–27, 2021.

Pan Xie, Zhi Cui, Xiuying Chen, XiaoHui Hu, Jianwei Cui, and Bin Wang. Infusing sequential information into conditional masked translation model with self-review mechanism. In Proceedings of the 28th International Conference on Computational Linguistics, pp. 15–25, 2020.

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, 2020a.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. IEEE Transactions on Knowledge and Data Engineering, 2020b.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI Open, 1:57–81, 2020.

A APPENDIX

Distance ablation We further analyze the role of each edge distance in Table.6, where the C_α-O" and "C-N" distances are the most significant ones. Last but not least, we find that the distances between learnable virtual atoms significantly improve the recovery, suggesting that the model can automatically discover critical residual points.

Table 6: Ablation of edge distance features

	Model	PiFold	1	2	3	4	5	6	7	8	9	10
Real distances	C_α-C_α	✓	✓							✓	✓	✓
	C_α-C	✓		✓	✓	✓				✓	✓	✓
	C_α-N	✓		✓						✓	✓	✓
	C_α-O	✓		✓	✓		✓			✓	✓	✓
	C-C	✓	✓							✓	✓	✓
	C-N	✓		✓	✓			✓		✓	✓	✓
	C-O	✓		✓						✓	✓	✓
	N-N	✓	✓							✓	✓	✓
	N-O	✓		✓	✓				✓	✓	✓	✓
	O-O	✓	✓							✓	✓	✓
Virtual atoms	0									✓		
	1										✓	
	2											✓
	3	✓	✓	✓	✓	✓	✓	✓	✓			
Results	Perplexity ↓	4.55	4.55	4.53	4.53	4.54	4.54	4.52	4.55	4.59	4.54	4.53
	Recovery ↑	51.66	51.52	51.63	51.74	51.49	**51.70**	**51.89**	51.50	51.02	51.10	51.51
	Change	–	↓	↓	↑	↓	↑	↑	↓	↓↓↓	↓↓	↓

Baseline details ProteinMPNN supports the design of proteins when partial residues are known, while we focus on designing sequences from structures without any known residues. On CATH4.2, we note that ProteinMPNN directly assigns sequence labels to residues with missing coordinates, indicating these residues are treated as known ones. If we allow this trick, ProteinMPNN could achieve 51% recovery on CATH4.2. To force all methods to be compared under the same setting, we remove this operation and update the recovery to 45.96%. For ESM-IF, we report their results without training by third-party data. They have proved that increased data volume could significantly improve model capability, thus affecting researchers' judgments about the models. We reproduced most of the baselines and allow them to be trained and evaluated under the same data splitting and code framework. We admire these works, and thank all the authors providing open-source codes.