PRODESIGN: TOWARD EFFECTIVE AND EFFICIENT PROTEIN DESIGN

Zhangyang Gao & Cheng Tan
AI Research and Innovation Lab
Westlake University
Zhejiang University

Stan Z. Li
AI Research and Innovation Lab
Westlake University
`Stan.ZQ.Li@westlake.edu.cn`

ABSTRACT

How to design protein sequences folding into the desired structures effectively and efficiently? Structure-based protein design has attracted increasing attention in recent years; however, few methods can simultaneously improve the accuracy and efficiency due to the lack of expressive features and autoregressive sequence decoder. To address these issues, we propose ProDesign, which contains a novel residue featurizer and ProGNN layers to generate protein sequences in a one-shot way with improved recovery. Experiments show that ProDesign could achieve 51.66% recovery on CATH 4.2, while the inference speed is 70 times faster than the autoregressive competitors. In addition, ProDesign achieves 58.72% and 60.42% recovery scores on TS50 and TS500, respectively. We conduct comprehensive ablation studies to reveal the role of different types of protein features and model designs, inspiring further simplification and improvement.

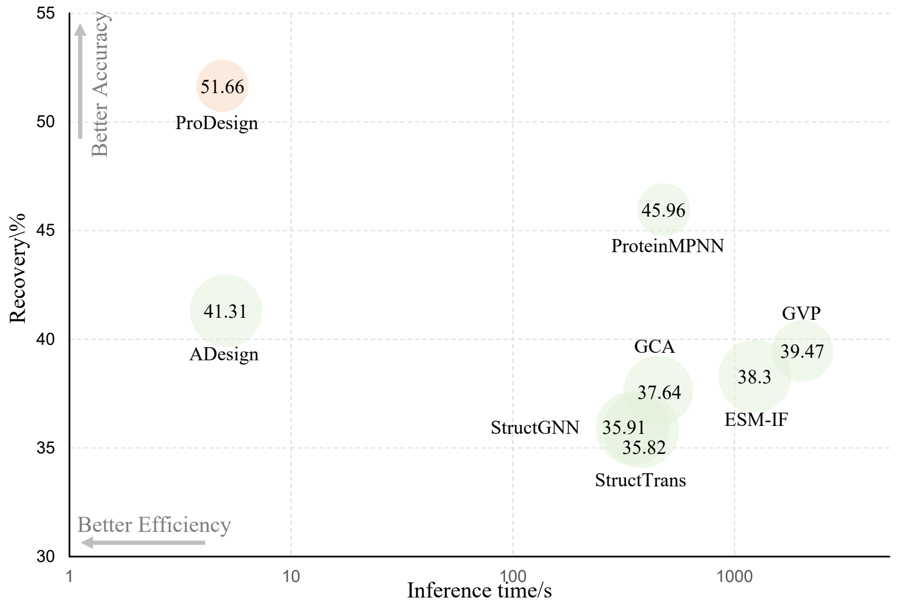


Figure 1: Performance comparison. We compare graph-based protein design methods, recording the recovery scores with the Y-axis, the inference time cost with the X-axis, and the perplexity with the circle size. The Recovery and perplexity results in the CATH dataset are reported without third-party training data. The inference time is the cost to generate 100 long protein sequences (length $>$ 1000) 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 for human purposes could deepen our understanding of living systems and facilitate the fight against the

Preprint version.

disease. One of the crucial problems is to design protein sequences that fold to desired structures, namely structure-based protein design (Pabo, 1983). Recently, many 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; 2022a; Dauparas et al., 2022; Ingraham et al., 2019; Jing et al., 2020; Tan et al., 2022b; 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), among which graph-based models have made significant progress. However, there is still a large room to improve the model's accuracy and efficiency. For example, most graph models could not achieve 50+% sequence recovery in CATH dataset due to the lack of expressive residue representations. Moreover, they also suffer from the autoregressive generation mechanism, resulting in low inference speed. How to simultaneously improve the model accuracy and efficiency is a crucial problem. We aim to solve this problem with a simple model containing as few redundancies as possible.

For years, graph-based models have struggled to learn expressive residue representations through better feature engineering, more elaborate models, and a larger training dataset. For example, ADesign (Gao et al., 2022a) and ProteinMPNN (Dauparas et al., 2022) point out that additional angle features and distance features could significantly improve the representation quality. GraphTrans (Ingraham et al., 2019), GVP (Jing et al., 2020) and GCA (Tan et al., 2022b) introduce new modules considering graph-based message passing, geometric vectors 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 great progress, there are still two remaining problems to be solved under the same data set setting: (1) How to construct more comprehensive features to facilitate residue representation? (2) Could the model expression be improved by considering residual interactions at different levels?

Most graph models (Dauparas et al., 2022; Ingraham et al., 2019; Jing et al., 2020; Hsu et al., 2022; Tan et al., 2022b; Hsu et al., 2022) adopt the autoregressive decoding scheme to generate amino acids, dramatically slowing down the inference process. Interestingly, few studies have attempted to improve the model efficiency, perhaps because the efficiency gain requires sacrificing some 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 latter is more important than efficiency in protein design. To address this dilemma, ADesign (Gao et al., 2022a) proposes a parallel self-correcting module to speed up inference while maintaining the recovery. Nevertheless, it still requires two iterations for prediction, where the model outputs the predicted confidence the first time and the corrected results in the second. An important issue is: How to generate protein sequences in a one-shot way without loss of precision?

We propose ProDesign to address the problems above, which contains a novel residue featurizer with stacked ProGNN layers. As to the featurizer, for each residue, we construct more comprehensive features and introduce learnable virtual atoms to capture information overlooked by real atoms. To learn multi-scale residue representations, the ProGNN considers residue interactions at the node, edge, and global context. In addition, we could completely remove the autoregressive decoder by stacking more ProGNN layers without sacrificing accuracy. Experiments show that ProDesign can achieve state-of-the-art recovery on several real-world datasets, i.g., 51.66% on CATH 4.2, 58.72 on TS 50, and 60.42% on TS 500. ProDesign is the first graph model that could exceed 55% recovery on TS50 and 60% recovery on TS500. In addition, the inference efficiency is also improved by 70 times compared to autoregressive competitors. More importantly, we conduct extensive ablation studies to reveal the role of each module, which helps deepen the reader's understanding of ProDesign and may provide further inspiration for subsequent research. In summary, our contributions include:

1. We propose a novel residue featurizer to construct comprehensive residue features and learn virtual atoms to capture complementary information with real atoms.
2. We propose a ProGNN layer to extract multi-scale residue interactions at the node, edge, and global context levels.
3. We suggest removing the autoregressive decoder for speeding up generation without sacrificing accuracy.
4. We comprehensively compare advanced graph models in various real-world datasets, including CATH, TS50, and TS500.

2 Related works

This section formally states the structure-based protein design problem and provides a comprehensive view of related works.

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

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

Because homologous proteins always share similar structures (Pearson & Sierk, 2005), the problem itself is underdetermined, i.e., the valid amino acid sequence may not be unique (Gao et al., 2020). To consider both sequential and structural dependencies, recent works (Hsu et al., 2022) suggest combining 3D structural encoder and 1D sequence decoder, where the protein sequences are generated in an autoregressive way:

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

Based on different types of structural encoders, existing works can be divided into MLP-based, CNN-based, and GNN-based ones (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. SPIN (Li et al., 2014) integrates torsion angles (ϕ and ψ), fragment-derived sequence profiles, and structure-derived energy profiles to predict protein sequences. SPIN2 (O'Connell et al., 2018) adds backbone angles (θ and τ), local contact number, and neighborhood distance to improve the recovery from 30% to 34%. (Wang et al., 2018) uses backbone dihedrals (ϕ, ψ and ω), sovlent 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 $C_\alpha - C$, which achieves 33.0% recovery on 50 test proteins. The MLP methods have a high inference speed, but their recovery is relatively low due to the partial consideration of structural information. The complex feature engineering may require multiple databases and computational tools, limiting the widespread usage.

CNN-based models CNN methods extract residue features from the protein structure, which can be further classified as 2D CNN-based and 3D CNN-based. The 2D CNN-based SPROF (Chen et al., 2019) extracts structural features from the distance matrix and achieves 39.8% recovery. In contrast, 3D CNN-based methods extract residue features from the atom distribution in a three-dimensional grid box. For each residue, the atomic density distribution is computed after being translated and rotated to a standard position so that the model can learn translation and rotation invariant features. ProDCoNN (Zhang et al., 2020a) designs a nine-layer 3D CNN to predict the corresponding residues at each position, which uses multi-scale convolution kernels and achieves 42.2% recovery. DenseCPD (Qi & Zhang, 2020) further uses the DensetNet architecture (Huang et al., 2017) to boost the recovery to 55.53%. Recent works (Anand et al., 2022) has also explored the potential of deep models to generalize to *de novo* proteins. Although 3DCNN-based models improve recovery, their inference is slow, probably because they require separate pre-processing and prediction for each residue.

Graph-based models Graph methods represent the 3D structure as a k-NN graph, then use 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., 2022a) to extract residue features while considering structural constraints. The protein graph encodes the residue information in node vectors and constructs edges and edge features between neighboring residues. GraphTrans (Ingraham et al., 2019) uses graph attention encoder and autoregressive decoder for protein design. GVP (Jing et al., 2020) increases the performance via novel geometric vector perceptrons, considering both scalar and vector features. GCA (Tan et al., 2022b) further introduces global graph attention for protein design. Another related work is ProteinSolver (Strokach et al., 2020), but it was mainly developed for scenarios where partial sequences are known and does not report results on standard datasets. Recently, ADesign (Gao et al., 2022a), ProteinMPNN (Dauparas et al., 2022) and Inverse Folding

(Hsu et al., 2022) achieve dramatically 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

We show the overall ProDesign framework in Figure.2, where the inputs are protein structures, and outputs are protein sequences expected to fold into the input structures. We propose a novel residue featurizer and ProGNN layer to learn expressive residue representations. Specifically, the residue featurizer constructs comprehensive residue features and creates learnable virtual atoms to capture information complementary to real atoms. The ProGNN considers multi-scale residue interactions in node, edge, and global context levels. ProDesign 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., 2022b; Dauparas et al., 2022; Hsu et al., 2022) or iterative models (Gao et al., 2022a).

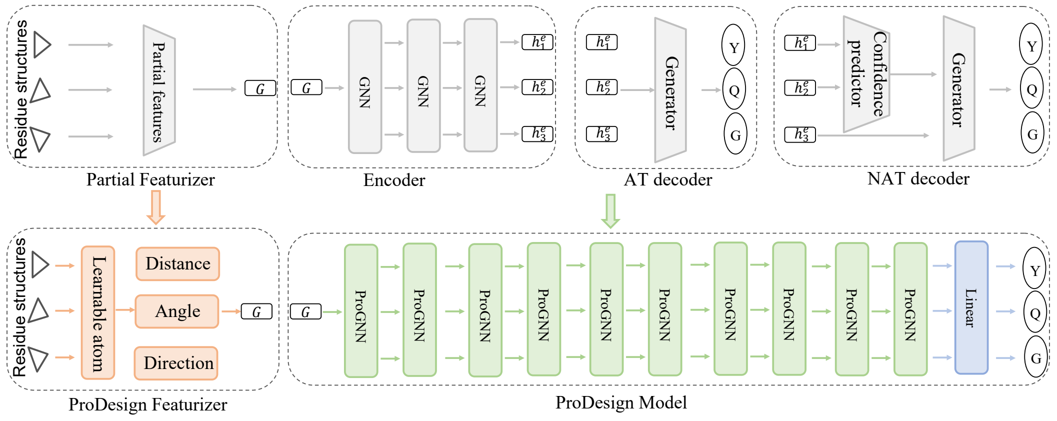


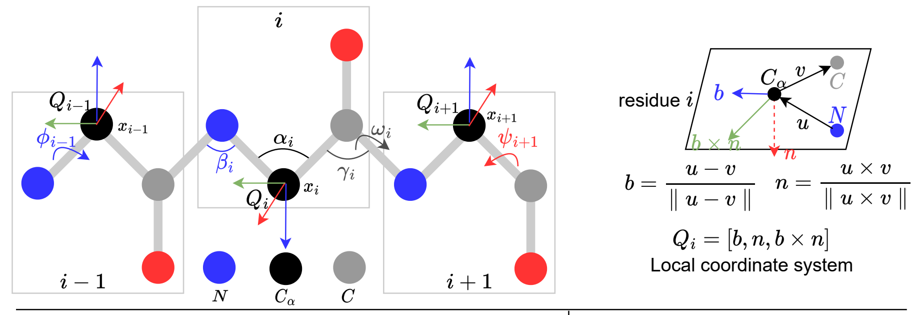
Figure 2: The overall framework of ProDesign. The featurizer module extracts comprehensive residue features from the fixed real atoms and learnable virtual atoms. We use stacked ProGNNs, which consider node-level, edge-level, and global-level residue interactions, to learn expressive residue representations. ProDesign generates protein sequences in a one-shot manner, without autoregressive or iterative decoding schemas.

3.2 FEATURIZE MODULE

This section describes how to construct protein graphs and residue features. We recommend readers to view Fig.3 for intuitive understanding.

Graph structure We represent the protein as a k-NN graph derived from residues to consider the 3D dependencies, where k defaults to 30. 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 regular structure, order, and coordinate of residues. 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 $\boldsymbol{u}_i = Ca_i - N_i, \boldsymbol{v}_i = C_i - Ca_i, \boldsymbol{b}_i = \frac{\boldsymbol{u}_i - \boldsymbol{v}_i}{||\boldsymbol{u}_i - \boldsymbol{v}_i||}$, 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 pair 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 introduce 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



	Node i	Edge (i,j)
Distance	$\|\|C_{\alpha_i} - C_i\|\|$ $\|\|C_{\alpha_i} - N_i\|\|$ $\cdots$	$\|\|C_{\alpha_i} - C_j\|\|$ $\|\|C_{\alpha_i} - N_j\|\|$ $\cdots$
Angle	α_i--Angle for N_i-C_{α_i}-C_i ϕ_i--Dihedral angle for N_i-C_{α_i} β_i--Angle for C_{i-1}-N_i-C_{α_i} ψ_i--Dihedral angle for C_i-C_{α_i} γ_i--Angle for C_{α_i}-C_i-N_{i+1} ω_i--Dihedral angle for C_i-N_{i+1}	$q(R_{ji}) = q(Q_i^T Q_j)$
Direction	$Q_i^T \frac{N_i - C_{\alpha_i}}{\|\|N_i - C_{\alpha_i}\|\|}$ $Q_i^T \frac{O_i - C_{\alpha_i}}{\|\|O_i - C_{\alpha_i}\|\|}$ $\cdots$	$Q_i^T \frac{N_j - C_{\alpha_i}}{\|\|N_j - C_{\alpha_i}\|\|}$ $Q_i^T \frac{O_j - C_{\alpha_i}}{\|\|O_j - C_{\alpha_i}\|\|}$ $\cdots$

Figure 3: ProDesign featurizer. We construct distance, angle, and direction features for single or pairs of residues; all features are rotation- and translation-invariant. The learnable virtual atoms are used for the construction of distance features.

$\{V_i^1, V_i^2, V_i^3, \cdots\} \times \{V_i^1, V_i^2, V_i^3, \cdots\}$ 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.

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 rotationally invariant.

3.3 ProGNN

We propose a ProGNN layer to learn geometric residue representations considering multi-scale residue interactions in node, edge, and global context level.

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., 2022b) 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 indice set of residues that belong 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 ProDesign on different datasets and show its superior accuracy and efficiency. Detailed ablation studies are applied to deepen the understanding about ProDesign. In summary, we will answer following questions:

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

4.1 Performance (Q1)

Objective & Setting Could ProDesign achieve better recovery and efficiency on real-world datasets than SOTA methods? Does the efficiency improvement require sacrificing the generative quality? To answer these questions, we compare ProDesign 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), resulting in 18024 proteins for training, 608 proteins for validation, and 1120 proteins for testing. We stack 10 layers of ProGNN to construct the ProDesign model with hidden dimension 128. The model is trained up to 100 epochs by default using the Adam optimizer on NVIDIA V100s. 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 (length>1000).

Baselines We compare ProDesign with recent graph models, including StructGNN, StructTrans (Ingraham et al., 2019), GCA (Tan et al., 2022b), GVP (Jing et al., 2020), GVP-large, ESM-IF (Hsu et al., 2022), ADesign (Gao et al., 2022a), ProteinMPNN (Dauparas et al., 2022), since most of them are open-sourced. To make a fair and reliable comparison, we re-run StructGNN, StructTrans, GCA, GVP, ADesign, and ProteinMPNN under the same data splitting on CATH 4.2. The results of GVP-large and ESM-IF are copied from their paper since they do not provide the training scripts, and we do not reproduce them with the risk of performance degradation. It should be reminded that ESM-IF uses CATH 4.3 for training, which may lead to higher recovery due to the increased data volume. 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.

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 (Hsu et al., 2022). The **best** and <u>suboptimal</u> 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		✓
ADesign	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	<u>6.21</u>	6.68	<u>4.61</u>	<u>36.35</u>	34.43	45.96	✓	
ProDesign	**6.04**	<u>6.31</u>	**4.55**	**39.84**	<u>38.53</u>	**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 (Ingraham et al., 2019; Jing et al., 2020; Tan et al., 2022b; 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 ProDesign could consistently improve the perplexity (lower is better) and recovery score (higher is better) across different test sets. On the ”Short” and ”All” datasets, ProDesign achieves the best perplexities and recovery scores, where the recovery improvement are 3.49% and 1.94%, respectively. On the ”Single-chain” dataset, ProDesign is still the best model when using the CATH 4.2 training set.

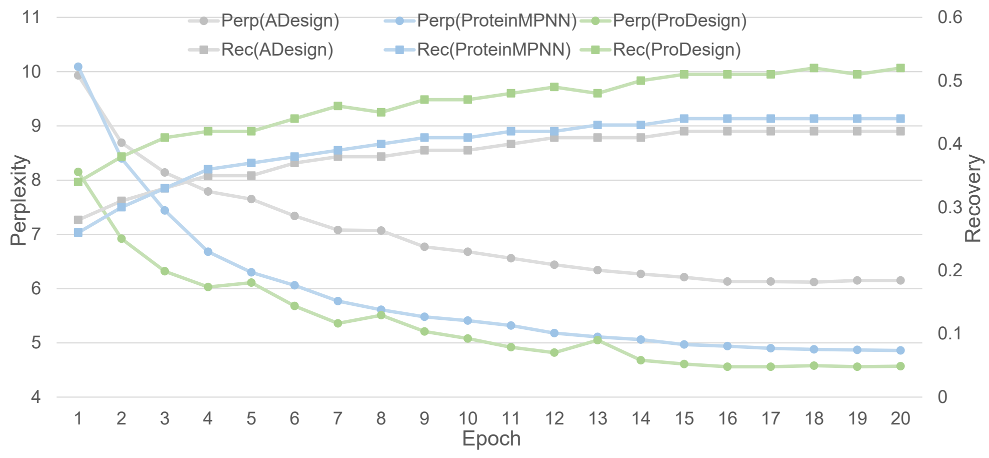


Figure 4: Training ProDesign for 20 epochs is enough to achieve SOTA performance.

Efficiency We already know that ProDesign can achieve higher recovery; does this improvement come at the expense of efficiency? We further evaluate the training and inference time costs of ProDesign and the competitive baselines (ADesign and ProteinMPNN). ProDesign 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 ProDesign. As to the inference speed, all baselines except ADesign adopt an autoregressive decoder to generate residues one by one, and the computational complexity is O(L) to the protein length L. However, ProDesign enjoys O(1) computational complexity due to the one-shot generative schema. We benchmark the inference speed of different graph models for inference speed on 100 long chains (residue number 1000) and show the results in Fig.1, where ProDesign 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

	Model	ProDesign	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 first determine the contribution rank of different types of features, as shown 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 distance in Table.6, where the ”Ca-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.

Table 3: Ablation of ProGNN modules.

		ProDesign	model 1	model 2	model 3	model 4	model 5
Node	GCN		✓				
	GAT			✓			
	QKV				✓		
	AttMLP	✓				✓	✓
Edge	UpadateEdge	✓	✓	✓	✓	✓	
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 ProGNN 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., 2022b; 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: Non-autoregressive ablation. ”Enc” and ”AT” represent the number of encoder and autoregressive layers, respectively.

	ProDesign	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 $\downarrow$	4.55	5.06	5.04	4.92	4.83	4.58	4.53
Recovery $\uparrow$	51.66	49.30	49.37	49.60	50.41	50.96	51.22
Test Time $\downarrow$	36s	527s	707s	522s	347s	30s	34s

NAT ablation How does ProDesign eliminate autoregressive mechanisms and ensure good performance? We further explore the effectiveness of the autoregressive decoder (Ingraham et al., 2019) and one-shot ProGNN encoder. As shown in Table.1, by gradually replacing the autoregressive decoder with the one-shot ProGNN (model 1 $\rightarrow$ model 5), the recovery will be improved while the inference time cost on CATH is reduced. 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 $\rightarrow$ model 6).

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 <u>suboptimal</u> results are labeled with bold and underline.

Group	Model	TS50			TS500		
		Perplexity $\downarrow$	Recovery $\uparrow$	Worst $\uparrow$	Perplexity $\downarrow$	Recovery $\uparrow$	Worst $\uparrow$
MLP	SPIN †		30.00				
	SPIN2 †		34.00				
	Wang's model †		33.00				
CNN	SPROF †		39.80				
	ProDCoNN †		46.50				
	DenseCPD †		<u>55.53</u>				
Graph	StructGNN	5.40	43.89	26.92	4.98	45.69	<u>0.05</u>
	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
	ADesign	5.25	48.36	32.31	4.93	49.23	0.03
	ProteinMPNN	<u>3.93</u>	54.43	<u>37.24</u>	<u>3.53</u>	<u>58.08</u>	0.03
	ProDesign (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 ProDesign 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 ProDesign could achieve consistent improvements across TS50 and TS500. ProDesign is the first graph model that could exceed 55% recovery on TS50 and 60% on TS500.

5 CONCLUSION

We propose ProDesign to improve both the accuracy and efficiency of structure-based protein design. A novel protein featurizer and ProGNN layer are introduced to learn expressive residue representations. ProDesign achieves 51.66%, 58.72%, 60.42% recovery scores on CATH 4.2, TS50 and TS500 datasets. Moreover, the inference speed of ProDesign is 70 times faster than autoregressive competitors. We tried our best to simplify the model and believe that the comprehensive ablation could provide inspiration for future studies.

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.

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.

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.

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.

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.

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

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.

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.

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

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.

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.

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. Target-aware molecular graph generation. arXiv preprint arXiv:2202.04829, 2022a.

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

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.

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.

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 distance in Table.6, where the "Ca-O" and "C-N" distances are the most significant ones. Interestingly, this seems to imply that considering the geometric triangles formed by adjacent chemical bonds helps to improve the model performance, e.g., $Ca - O$ connects adjacent bonds $Ca - C$ and $C - O$, and $C - N$ connects adjacent bonds $C - Ca$ and $Ca - N$. 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 distance features

	Model	ProDesign	1	2	3	4	5	6	7	8	9	10
Real distances	Ca-Ca	✓	✓							✓	✓	✓
	Ca-C	✓		✓	✓	✓				✓	✓	✓
	Ca-N	✓		✓						✓	✓	✓
	Ca-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	–	↓	↓	↑	↓	↑	↑	↓	↓↓↓	↓↓	↓