TTS-Design: Test-Time Compute Scaling for Structure-Guided Protein Design

Zizhe Jin[1], **Yizhen Zheng**[2*], **Huan Yee Koh**[3], **Jiaxin Ju**[1], **Jiapu Wang**[4], **Shirui Pan**[1]
[1]School of Information and Communication Technology, Griffith University
[2]Department of Cancer Medicine, Monash University
[3]Department of Data Science and AI, Monash University
[4]Beijing Institute of Artificial Intelligence, Beijing University of Technology
zizhe.jin@griffithuni.edu.au, {yizhen.zheng,huan.koh}@monash.edu, jiaxin.ju@griffithuni.edu.au, jiapuwang9@gmail.com, s.pan@griffith.edu.au

Abstract

Generating protein sequences that reliably fold into target structures is a central challenge in computational biology and protein design. Progress in protein inverse folding (PIF), however, is fundamentally constrained by the scarcity of high-quality structural data, which limits the effectiveness of existing models. In this work, we propose TTS-Design, a test-time compute scaling framework that enhances protein sequence design without retraining models or relying on larger training datasets. For TTS-Design, its core design is a dedicated inverse folding reward model named IF-RM, which quantitatively evaluates structure–sequence compatibility and enables effective sequence selection among multiple candidate sequences at inference time. By integrating IF-RM into existing PIF models, TTS-Design leverages additional computational resources during inference stage to explore and refine candidate solutions. Extensive experiments on both CATH 4.2 and CATH 4.3 demonstrate that TTS-Design can consistently improve sequence recovery and structural reliability across different backbone models, without retraining or increasing model size. Case studies on real-world proteins further confirm its practical effectiveness. Overall, our results highlight test-time compute scaling as a promising and effective paradigm for advancing protein inverse folding and other scientific problems under data-limited scenarios.

1 Introduction

Designing proteins is a fundamental goal in computational biology, with broad applications in enzyme engineering [Landwehr *et al.*, 2025], drug discovery [Zheng *et al.*, 2025] [Savage, 2023], and synthetic biology [Kim *et al.*, 2025]. The protein inverse folding (PIF) problem, predicting amino acid sequences that can fold into a desired structure, is crucial to protein design [Watson *et al.*, 2023], as it directly determines the accuracy and feasibility of designed proteins. Accurate solutions to this task would enable creation of novel functional proteins with atomic-level precision.

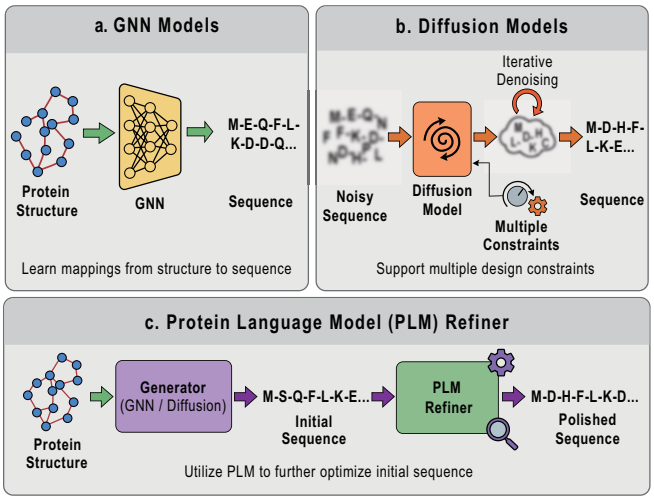


Figure 1: (a) GNN Models: Learn mapping relationships from protein structure to amino acid sequence. (b) Diffusion Models: Generate amino acid sequence guided by protein structure and multiple constraints. (c) Protein Language Model (PLM) Refiner: Utilize PLM to further polish a predicted sequence and generate a better one.

Compared to physics-based approaches [Leaver-Fay *et al.*, 2011], deep learning methods have become the dominant paradigm for protein inverse folding recently. Owing to the intrinsic graph structure of proteins, numerous studies leverage graph neural networks (GNNs) to learn mappings from protein structures to amino acid sequences [Hsu *et al.*, 2022] [Dauparas *et al.*, 2022] [Gao *et al.*, 2023]. However, these models typically take inverse folding as a deterministic prediction problem, limiting their ability to incorporate multiple design constraints. In contrast, diffusion models address this limitation by formulating inverse folding as an iterative denoising process that supports dynamic constraint integration during sampling [Wu *et al.*, 2024] [Bai *et al.*, 2025] [Yi *et al.*, 2023]. Nowadays, as shown in Figure 1, protein language models (PLMs), pretrained on large-scale protein sequence corpora, are often employed as refiners to further polish amino acid sequences generated by existing models by

*Corresponding author.

injecting evolutionary priors [Zheng *et al.*, 2023] [Qiu *et al.*, 2024].

Despite these methodological advances, further progress in protein inverse folding ultimately depends on the availability of sufficient training data. In contrast to protein sequence, experimentally resolving protein structures is extremely expensive and time-consuming, which severely limits the amount of available high-confidence structure data. To date, only about 230,000 resolved structures are published in the Protein Data Bank (PDB) [Berman and Burley, 2025], accounting for less than 0.1% of the over 250 million known protein sequences [Consortium, 2025]. This striking imbalance highlights the severe data scarcity faced by structure-related tasks and restricts performance of current PIF models to a large extent, making it difficult to further improve performance by simply enlarging model size in pretraining phase. Therefore, a natural question arises:

> *Since pretraining stage reaches a bottleneck due to data scarcity, can we shift our attention from pretraining to inference stage?*

Inspired by this, various studies are trying to explore the scaling law at test time [Zhang *et al.*, 2025] [Snell *et al.*, 2024] [Guan *et al.*, 2026]. Unlike traditional train-time scaling, which improves model performance by increasing model size or enlarging the amount of training data, Test-Time Scaling (TTS) enhances a model's reasoning and decision-making ability by allocating additional computational resources during inference stage. By expanding the solution space at test time for any input query, TTS allows models to explore more alternative solutions and generate more accurate and reliable predictions. Considering PIF is a typical data-limited scenario as well, applying test-time scaling to this task provides an exciting and promising direction. By generating multiple candidate amino acid sequences at test time and choosing the most plausible one, TTS can significantly improve the performance of current PIF models.

To achieve this, we propose TTS-Design, a novel test-time scaling framework that improves the performance of existing PIF models by generating multiple candidate sequences at inference time. To enable effective sequence selection, TTS-Design contains an Inverse Folding Reward Model (IF-RM), which quantitatively evaluates structure–sequence consistency by jointly encoding protein structures and candidate sequences utilizing graph and sequence encoders. By integrating a pretrained IF-RM into existing PIF models, TTS-Design can consistently improve sequence recovery without increasing model size or requiring additional training data.

In summary, the key contributions of our work are as follows:

- We explore the potential of test-time compute scaling in protein inverse folding task, proposing an unexplored possibility in this area.
- We propose a dedicated reward model, IF-RM, specifically designed for the PIF task. Based on it, we further develop a new framework, TTS-Design, that enables effective test-time scaling in protein inverse folding.
- We validate the effectiveness of both IF-RM and TTS-Design, demonstrating that test-time scaling can enhance performance of existing PIF models significantly.

Moreover, our findings offer valuable insights and inspiration for future work in protein inverse folding and similar test-time scaling strategies can also be applied to other AI-driven protein design [Bio and Biswas, 2025].

2 Preliminaries

Protein inverse folding, also known as structure-guided protein design, aims to predict an amino acid sequence that can fold into a desired protein structure. Unlike a folding problem, which maps sequences to structures [Jumper *et al.*, 2021] [Abramson *et al.*, 2024], PIF explores the reverse relationship and receives attention in recent years.

Problem definition: The objective of protein inverse folding is to train a deep learning model $\mathcal{F}_\theta$ with parameters θ that takes protein structure $\mathcal{X} = \{x_i : 1 \leq i \leq L\}$ as input and predicts the corresponding amino acid sequence $\mathcal{S} = \{s_i : 1 \leq i \leq L\}$, where L denotes the sequence length and proteins are composed of 20 standard amino acids. Formally, we have

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

Given enough structure-sequence protein data, the learning objective is to find parameters θ with the highest conditional likelihood $p(\mathcal{S} \mid \mathcal{X}; \theta)$

Overview: With the emergence of protein language models [Lin *et al.*, 2022], protein inverse folding has achieved remarkable progress recently. As described in Figure 1 (c), current PIF models typically follow a *Base-PLM* framework, in which a base model serves as the structure encoder to generate an initial sequence S_{init}, while a protein language model (PLM) further refines the initial sequence to produce a more plausible output S_{final}. Above process can be summarized as:

$$\mathcal{S}_{\text{init}} = \text{Base}(\mathcal{X}),\ \mathcal{S}_{\text{final}} = \text{PLM}(\mathcal{X}, \mathcal{S}_{\text{init}}).$$

Here, *Base* and *PLM* denote a base model (e.g., ProteinMPNN [Dauparas *et al.*, 2022], PiFold [Gao *et al.*, 2023]) and a PLM-based model, respectively. This two-level framework has become the foundation for current state-of-the-art PIF models.

3 Method

In this section, we introduce our designed *TTS-Design* framework, which integrates test-time scaling into the protein inverse folding task. Rather than adopting existing test-time paradigms, *TTS-Design* is specifically designed for structure-guided sequence generation, unifying both structure-based reasoning and sequence-level refinement in one framework. This design enables our model to allocate additional computation at test time, thereby enhancing sequence quality without retraining or enlarging training datasets.

3.1 TTS-Design Framework

A typical test-time scaling solution can be summarized as a *proposer–verifier* paradigm [Snell *et al.*, 2024], where a

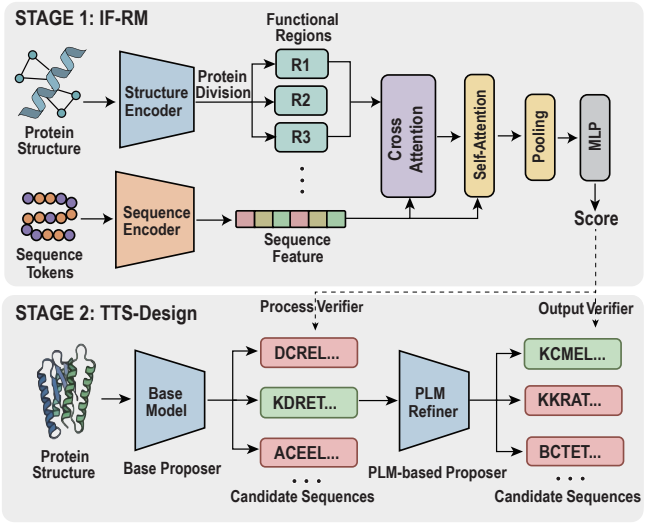


Figure 2: Overall architecture of our proposed IF-RM and TTS-Design.

model improves its reasoning ability during inference by generating multiple candidate solutions (*proposers*) and evaluating them using a pretrained reward model (*verifier*).

Inspired by this, our designed *TTS-Design* framework is composed of two *proposers* and two *verifiers* as shown in Figure 2. The *proposers* are responsible for generating candidate sequences, while the *verifiers* assess their quality and guide sequence selection. In this section, we will introduce the components of *TTS-Design* overall and describe details in the next section.

Base proposer. Given a target protein structure $X = \{x_i : 1 \leq i \leq L\}$, the *base proposer* serves as the structural reasoning module. It encodes atomic coordinates and local geometric features through a graph neural network or a diffusion model and generates multiple candidate sequences:

$$S^t_{\text{init}} \sim f^{(\text{Base})}_{\text{prop}}(X),\ t = 1, \ldots, T,$$

where $f^{(\text{Base})}_{\text{prop}}$ is the *base proposer*, and T is the number of candidate sequences. These candidates represent diverse structural interpretations, forming the initial sequence space for downstream refinement.

Process verifier. Then, *TTS-Design* uses a pretrained verifier named *IF-RM* (Inverse Folding Reward Model) to evaluate the quality of candidate sequences S^t_{init}. Specifically, the *process verifier* operates after the *base proposer*, scoring each candidate sequence based on its consistency with the input structure and guiding candidate selection for the following refinement. Taking r^t_{proc} as score of candidate sequence, the scoring process can be described as:

$$r^t_{\text{proc}} = f_{\text{verifier}}(X, S^t_{\text{init}}), \quad t = 1, \ldots, T,$$

and the top-scoring candidate S^*_{init} is selected.

$$S^*_{\text{init}} = \arg\max_{S^t_{\text{init}}} f_{\text{verifier}}(X, S^t_{\text{init}})$$

PLM-based proposer. After selecting the top-scoring candidate from the *base proposer*, the *PLM-based proposer* acts as a sequence refiner that further improves sequence quality using contextual knowledge learned from large-scale protein corpora. It jointly considers the structural information and initial sequence to generate refined sequences:

$$S^t_{\text{final}} \sim f^{(\text{PLM})}_{\text{prop}}(X, S^*_{\text{init}}),\ t = 1, \ldots, T,$$

where S^*_{init} is the selected best sequence from the *process verifier*. The *PLM-based proposer* thus enables higher-level reasoning in the sequence space, producing final candidates that are both structurally consistent and evolutionarily plausible.

Together, the two proposers form a standard reasoning process: the *base proposer* captures geometric and topological constraints of the structure, while the *PLM-based proposer* performs sequence-level refinement conditioned on both structure and prior sequence. This two-stage generation forms the foundation for test-time scaling in PIF.

Output verifier. For final candidate sequences S^t_{final}, the *output verifier* operates at the PLM refinement stage, assessing refined sequences to determine which sequence best matches the target structure. Similar to *process verifier*, the scoring process can be summarized as follows.

$$r^t_{\text{out}} = f_{\text{verifier}}(X, S^t_{\text{final}}), \quad t = 1, \ldots, T,$$

and the final prediction is chosen by

$$S^*_{\text{final}} = \arg\max_{S^t_{\text{final}}} f_{\text{verifier}}(X, S^t_{\text{final}}).$$

Formally, both *process* and *output verifiers* share the same model, denoted as $f_{\text{verifier}}(X, S)$, which takes structure X and candidate sequence S as inputs, and outputs a score representing the quality of the candidate sequence:

$$r = f_{\text{verifier}}(X, S).$$

Considering *process* and *output verifiers* are one model named *IF-RM*, we will describe its details in the following section.

3.2 IF-RM

To enable test-time compute scaling, we introduce a reward model designed for PIF problems named *IF-RM*, which estimates the consistency between an amino acid sequence and the input protein structure, and outputs a score, where a higher value indicates stronger sequence–structure compatibility.

As shown in Figure 2, our *IF-RM* takes both the structure coordinates $\mathbf{X}$ and sequence tokens $\mathbf{S}$ as input and encode them separately.

Structure and Sequence Encoders: For input structure $\mathbf{X}$, we use the featurizer in PiFold [Gao *et al.*, 2023] as our structure encoder $Encoder_{\text{st}}$ to extract protein graphs G based on input structure coordinates. The extracted protein graph comprises both node features and adjacency matrix, calculated by distance and angle relations between various residues.

$$\mathbf{G} = Encoder_{\text{st}}(\mathbf{X}) \tag{1}$$

where $G = (\mathbf{H_{st}}, \mathbf{A})$, and $\mathbf{H_{st}} \in R^{L \times d_{st}}$, $\mathbf{A} \in R^{L \times L}$ denote node feature and adjacency matrix respectively, and d_{st} is structure dimension.

For input sequence $\mathbf{S}$, we take a pretrained protein language model ESM-2 [Lin *et al.*, 2022] as our sequence encoder $Encoder_{\text{seq}}$ to generate residue-level contextual embeddings.

$$\mathbf{H}_{\text{seq}} = Encoder_{\text{seq}}(\mathbf{S}) \in R^{L \times d_{seq}}, \tag{2}$$

where $\mathbf{H}_{seq}$ represents sequence embedding, and d_{seq} denotes sequence dimension.

Protein Region Division: To guide the *IF-RM* better learning the constraint between protein structure and sequence, we take a *protein region division* to group residues into distinct functional regions. In conjunction with the cross-attention mechanism in the following section, IF-RM learns the interaction between protein regions and individual residues, thereby enabling it to capture constraints between structure and sequence.

For any protein graph G, *protein region division* implements a two-layer GCN trained on the minCUT [Bianchi *et al.*, 2020] unsupervised objective to obtain region embeddings $\mathbf{R}_{st}$:

$$\mathbf{W_c} = \text{GCN}(\mathbf{H_{st}}, \mathbf{A}) \tag{3}$$

$$\mathbf{R}_{st} = \mathbf{W_c}^{\top} \mathbf{H_{st}} \in R^{k \times d_{st}} \tag{4}$$

in which $\mathbf{W_c}$ represents cluster weights and k is the number of protein regions, which is a hyperparameter.

To ensure the learned regions are both structurally coherent and mutually distinct, we adopt two auxiliary objectives: the minCUT loss $\mathcal{L}_{\text{cut}}$ and orthogonality loss $\mathcal{L}_{\text{orth}}$, respectively.

$$\mathcal{L}_{\text{cut}},\ \mathcal{L}_{\text{orth}} = \text{MinCutPool}\big(\mathbf{H_{st}}, \mathbf{A}, \mathbf{W}_c\big), \tag{5}$$

Structure-Sequence Cross-attention: With region embeddings R_{st} and sequence embeddings H_{seq}, we apply *structure-sequence cross-attention* to capture the constraint between the protein structure and sequence. Firstly, we project both region and sequence embeddings into a shared attention space with dimension d.

$$\mathbf{Q} = \mathbf{R}_{\text{st}} W_Q, \quad \mathbf{K} = \mathbf{H}_{\text{seq}} W_K, \quad \mathbf{V} = \mathbf{H}_{\text{seq}} W_V \tag{6}$$

in which $W_Q \in R^{d_{st} \times d}, W_K, W_V \in R^{d_{seq} \times d}$ are trainable matrices for queries, keys, and values, respectively. Let d_k denote the dimension of each attention head. To transform structure information to sequence embeddings, we calculate attention weights and update sequence embeddings as follows:

$$\mathbf{H}^{'}_{seq} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}, \tag{7}$$

Through cross attention, sequence embeddings have fused structure knowledge. Then, we concatenate these two embeddings $\mathbf{H}^{'}_{\mathbf{seq}}$ and $\mathbf{H_{seq}}$ and project them back to dimension d_{seq}, and take $\mathbf{H_{cross}}$ as projected feature.

$$\mathbf{H_{cross}} = \text{Linear}(\text{Concat}(\mathbf{H_{seq}}, \mathbf{H}^{'}_{seq})) \tag{8}$$

Intra-Sequence Self-attention: After incorporating structural constraints through cross-attention, the updated sequence representations further interact through an *intra-sequence self-attention* layer. This module allows each residue to exchange information with all others, enabling *IF-RM* to capture long-range dependencies within the protein sequence and refine context-aware representations.

Formally, given the refined sequence embeddings $\mathbf{H_{cross}}$, the self-attention operation is defined as

$$\mathbf{H_{self}} = \text{softmax}\left(\frac{\mathbf{Q}^{'}\mathbf{K}^{'\top}}{\sqrt{d_k}}\right)\mathbf{V}^{'}, \tag{9}$$

where $\mathbf{Q}^{'} = \mathbf{H}_{cross} W^{'}_Q, \mathbf{K}^{'} = \mathbf{H}_{cross} W^{'}_K, \mathbf{V}^{'} = \mathbf{H}_{cross} W^{'}_V$.

Here $W^{'}_Q$, $W^{'}_K$ and $W^{'}_V$ are trainable projection matrices. After *intra-sequence self-attention*, we compute a mean pooling to get a protein-level representation $\mathbf{H}$ before applying a two-layer MLP to obtain the final output score.

$$\mathbf{H} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{H}_{\text{self}}(i), \tag{10}$$

$$\mathbf{score} = \sigma(W_2\, \phi(W_1 \mathbf{H} + b_1) + b_2)\,, \tag{11}$$

where ϕ and σ represent ReLU and Sigmoid activations, respectively.

Through above process, for any input protein structure X and sequence S, *IF-RM* can predict a score, reflecting compatibility of the input structure-sequence pair.

Training Objective: After obtaining output score r, we adopt a binary cross-entropy loss between the predicted score and the ground-truth score y as compatibility loss $\mathcal{L}_{\text{if}}$ and N is the amount of proteins.

$$\mathcal{L}_{\text{if}} = -\frac{1}{N}\sum_{n}\Big[y_n \log(r_n) + (1 - y_n)\log(1 - r_n)\Big], \tag{12}$$

To train *IF-RM*, we take a unified objective that combines the structural constraint losses as defined in $Eq.(5)$ and the structure-sequence compatibility loss.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{if}} + \mathcal{L}_{\text{cut}} + \mathcal{L}_{\text{orth}}, \tag{13}$$

4 Experiments

To evaluate *TTS-Design* comprehensively, we conduct extensive experiments on different datasets and hope to answer the following questions: **Q1 (Performance)**: Can *TTS-Design* improve the performance of current PIF models? **Q2 (Ablation)**: How does test-time scaling affect different parts of our method? **Q3 (Case study)**: Is *TTS-Design* effective in real-world protein design?

4.1 TTS-Design Performance

Experimental Setup: To validate the performance of our proposed *TTS-Design* framework, two key questions must be addressed: (1) Can our *IF-RM* effectively identify higher-quality sequences from a set of candidates? (2) Can the integration of test-time scaling further improve the performance of existing PIF models? To answer these questions,

we conducted experiments on both CATH 4.2 and CATH 4.3 datasets, which are widely recognized datasets in protein inverse folding problems. Both datasets were split following the standard method described in [Ingraham *et al.*, 2019] and [Hsu *et al.*, 2022]. For the *IF-RM* model, it consists of one cross-attention layer and one self-attention layer, including 3 attention heads. During pretraining, each protein structure was divided into 15 functional regions, and the model was optimized with a learning rate of 0.001. We take Symmetric Mean Absolute Percentage Error (SMAPE) as our evaluation metric to assess the scoring ability of *IF-RM*. In the *TTS-Design* inference phase, we selected GVP-Transformer [Hsu *et al.*, 2022], PiFold [Gao *et al.*, 2023] and LMdesign [Zheng *et al.*, 2023] as our representative models, and take temperature as 0.1 when sampling. To ensure fairness, we used the official open-source implementations released by the original papers. To evaluate sequence generation capability, we adopted the median sequence recovery as the primary metric, measuring the proportion of correctly recovered residues compared to the native sequence.

IF-RM Performance

Before evaluating *TTS-Design* framework formally, we need to test *IF-RM* comprehensively because this is the foundation of *TTS-Design* at test time. The goal of *IF-RM* is to assess the quality of candidate sequences, where a higher predicted score indicates a sequence that is more consistent with the target structure.

To train *IF-RM* effectively, we construct a dataset of structure—sequence—score triplets. Specifically, for each protein, we extract its native structure and sequence from the PDB file. We then randomly mask a certain proportion of residues in the native sequence to generate a partially corrupted sequence, and define the remaining known ratio as its ground-truth score. In this way, sequences with higher consistency to the given structure receive higher scores. This strategy enables the generation of an arbitrary number of synthetic training samples from existing protein datasets, supporting both supervised training and quantitative evaluation of *IF-RM*. Through this process, *IF-RM* learns to assess the consistency between any candidate sequence and the target structure.

To find the best hyper-parameters of *IF-RM*, we train it with different configurations and summarize results in Table 1. In our experiments, ca and k denote the number of cross-attention layers and functional regions, respectively.

As shown in Table **??**, on both CATH 4.2 and CATH 4.3 datasets, *IF-RM* achieves best performance when the number of cross-attention layers is 1 and the number of functional regions k is 15, yielding SMAPE values of 11.59% and 13.81%, respectively. Increasing the number of cross-attention layers beyond 1 did not lead to further improvement, due to redundant interactions. Similarly, larger or smaller k values also degraded performance, suggesting that a moderate level of structural division is essential for capturing structure–sequence relationships.

Therefore, in the following experiments, we adopt this optimal configuration of *IF-RM* ($ca = 1, k = 15$) as our default reward model to evaluate the performance of the proposed *TTS-Design* framework.

Model	ca	k	SMAPE (%)		Avg.
			CATH 4.2	CATH 4.3	
IF-RM	1	5	16.48	22.51	19.50
IF-RM	1	10	20.92	19.02	19.97
IF-RM	**1**	**15**	**11.59**	**13.81**	**12.70**
IF-RM	1	20	18.94	20.11	19.53
IF-RM	2	15	24.73	26.66	25.70
IF-RM	3	15	29.08	25.80	27.44
IF-RM	4	15	30.10	25.24	27.67

Table 1: Performance of IF-RM and its variants with different cross-attention layers (ca) and divided regions (k) on CATH 4.2 and CATH 4.3 datasets. Lower SMAPE indicates better scoring ability.

TTS-Design Performance

We then apply the pretrained *IF-RM* to the inference stages of existing inverse folding models, thereby realizing the *TTS-Design* framework. To verify whether test-time scaling can further enhance the performance of existing methods, we take GVP-Transformer [Hsu *et al.*, 2022] and PiFold [Gao *et al.*, 2023] as base models and further use LMdesign [Zheng *et al.*, 2023] as PLM-based models to evaluate our methods. Experiments are conducted on both CATH 4.2 and CATH 4.3 datasets, alongside comparisons with other PIF models. All results are summarized in Table 2, in which the results of *TTS-Design* are evaluated when the number of candidate sequences $T = 16$.

As shown in Table 2, the proposed *TTS-Design* framework consistently improves sequence recovery on both the CATH 4.2 and CATH 4.3 datasets across different backbone models. Taking PiFold as backbone, utilizing LMdesign as PLM-based refiner can increase sequence recovery from 51.66% to 55.40% on CATH 4.2. After applying *TTS-Design*, this value can further increase to 56.57%, gain 1.17% improvement. A similar trend also appears on CATH 4.3, where sequence recovery of *TTS-Design* reaches 57.00%, yielding a 0.98% improvement over LMdesign. The same conclusion also generalizes to GVP-Transformer, and we take encoder of GVP-Transformer in our experiments. On CATH 4.2, *TTS-Design* improves sequence recovery from 55.42% to 56.82%, increasing 1.4%. While on CATH 4.3, it increases sequence recovery from 56.19% to 56.81%, gain 0.62% improvement. Overall, these results suggest that test-time compute scaling can reliably improve performance of protein inverse folding models without retraining or modifying model parameters.

To better illustrate the impact of test-time scaling, Figure 3 shows how sequence recovery varies with the number of test-time candidates t. For the CATH 4.2 dataset using PiFold as the backbone (blue curve), sequence recovery improves overall as t increases, with a minor drop at $t = 8$. On CATH 4.3 with the same backbone (orange curve), recovery exhibits a generally upward trend as more candidates are generated. When applying *TTS-Design* to the GVP-Transformer backbone, we observe similarly positive trends on both datasets: on CATH 4.2 (green curve), recovery rises sharply from small t and then shows a slight drop at $t = 16$, while on CATH 4.3

Model	CATH4.2	CATH4.3
Baseline inverse folding models		
StructGNN [Ingraham *et al.*, 2019]	35.91	–
GraphTrans [Ingraham *et al.*, 2019]	35.82	–
AlphaDesign [Gao *et al.*, 2022]	41.31	–
ProteinMPNN [Dauparas *et al.*, 2022]	48.63	47.66
GRADE-IF [Yi *et al.*, 2023]	52.63	52.24
PiFold	51.66	51.45
+ LMdesign	55.40	56.02
+ TTS-Design (ours)	**56.57**	**57.00**
GVP-TransEncoder	–	–
+ LMdesign	55.42	56.19
+ TTS-Design (ours)	**56.82**	**56.81**

Table 2: Combined performance on CATH 4.2 and CATH 4.3. Values are median sequence recovery (%). For TTS-Design, we report results with $T = 16$ test-time candidates.

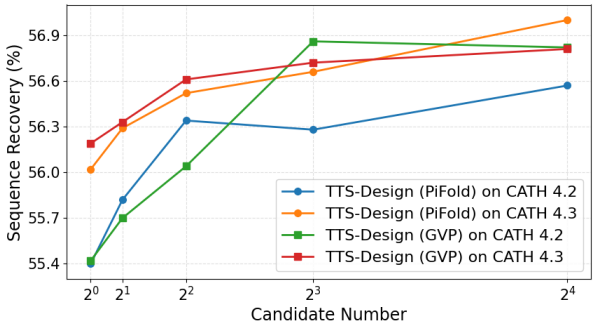


Figure 3: Trends of sequence recovery under different settings across candidate numbers.

(red curve), recovery improves gradually as t grows, indicating that *TTS-Design* can consistently improve performance of current models as candidate number increases.

4.2 Ablation Study

To further investigate the contribution of each component of *IF-RM* and *TTS-Design* framework, we conduct detailed ablation studies on both CATH 4.2 and CATH 4.3 datasets. Specifically, for *IF-RM*, we analyze the impact of its core modules, including cross-attention, self-attention, and graph division modules, and for *TTS-Design*, we evaluate two variants of this framework:

- **PiFold + TTS + LMdesign**, which applies test-time scaling only after the base model.
- **PiFold + LMdesign + TTS**, which applies test-time scaling only after the PLM-based refiner.
- **Full TTS-Design** model performs TTS at both stages.

First, we evaluate the contribution of each component of *IF-RM*, and conduct ablation studies on the CATH 4.2 and CATH 4.3 datasets using SMAPE (%) as the evaluation metric. As shown in Table 3, removing any key component results in a noticeable performance degradation,

IF-RM Variant	CATH 4.2	CATH 4.3
IF-RM (ca=1, k=15)	**11.59**	**13.81**
w/o Cross-Attention	33.15	26.13
w/o Self-Attention	17.74	17.29
w/o graph division	14.21	21.00

Table 3: Ablations on the IF-RM, using SMAPE (%) as metric. The full model uses the best setting found in Table 1 (i.e., ca=1, k=15). Setting k=1 disables graph division (single region).

(a) CATH 4.2

Model Variant	t=1	t=4	t=8	t=16
PiFold+**TTS**+LMdesign	55.40	55.69	55.92	55.98
PiFold+LMdesign+**TTS**	55.40	55.92	55.81	56.21
Full TTS-Design (ours)	55.40	56.34	56.28	56.57

(b) CATH 4.3

Model Variant	t=1	t=4	t=8	t=16
PiFold+**TTS**+LMdesign	56.02	56.55	56.30	56.45
PiFold+LMdesign+**TTS**	56.02	56.50	56.91	56.80
Full TTS-Design (ours)	56.02	56.52	56.66	57.00

Table 4: Ablation results of TTS-Design on CATH 4.2 and 4.3, and sequence recovery (%) is reported with $T \in \{1, 4, 8, 16\}$.

confirming that cross-attention, self-attention, and graph division modules are essential to the model's performance. Specifically, removing the cross-attention module leads to the largest performance drop, 33.15% and 26.13% on CATH 4.2 and CATH 4.3, respectively, indicating that capturing structure–sequence interactions is critical for sequence assessment. Eliminating self-attention also degrades performance substantially, suggesting that modeling long-range dependencies within the sequence improves prediction stability. Finally, when the graph division is disabled (k=1), performance declines moderately. These results validate the necessity of each module in constructing an effective *IF-RM*.

Then, we conduct ablation experiments on *TTS-Design*, and summarize results in Table **??**. Overall, both variants achieve lower performance than the complete *TTS-Design*, confirming that applying TTS at two test-time stages yields complementary benefits. More importantly, the impact of removing TTS after the PLM refiner is significantly larger than removing it after the base model. For instance, on CATH 4.2, excluding TTS from the PLM stage reduces sequence recovery from 56.57% to 55.98%, while excluding it from the base model results in only a minor drop to 56.21%. Similar trends are observed on CATH 4.3, where the performance degradation caused by removing TTS after the PLM refiner is consistently more obvious. These results indicate that all modules of *IF-RM* are essential, and TTS after PLM stage contributes more to optimization of amino acid sequences.

4.3 Case Study

To further demonstrate the effectiveness of the proposed *TTS-Design* framework in practical protein design, we conduct a series of case studies comparing its generated sequences

and folded structures with those predicted by existing models. While the above quantitative results have confirmed the general superiority of *TTS-Design*, then we aim to provide an intuitive perspective illustrating how test-time scaling improves the biological plausibility of designed sequences.

1a32.A
LTQERKREIIEQFKVHENDTGSPEVQIAILTEQINNLNEHLRVH
KKDHHSRRGLLKMVGKRRRLLAYLRNKDVARYREIVEKLGL

PiFold **32.94%**
MSEEEALELARRYTPEAAALGSPEARLAALHEEIEALEEYLEE
NPGDRERLAGLEELKGEREELLAELKGEDEAEYYAVLAELGL

LMdesign **36.47%**
MSEEEKLELLKRFTPEEAAAGSPVAQLQALDFAIVELSLYLDTH
PDDEEALAGLQKYVGRRQELLAELKGQDEALYWAVLAELGL

TTS-Design **67.06%**
MTRLEKQVVIEKYRDHASDTGSPEVQVALLTERINELQEHLKE
HKKDHHSRRGLLKMVGRRRRLLDYLKRVDVARYRAVVERLGL

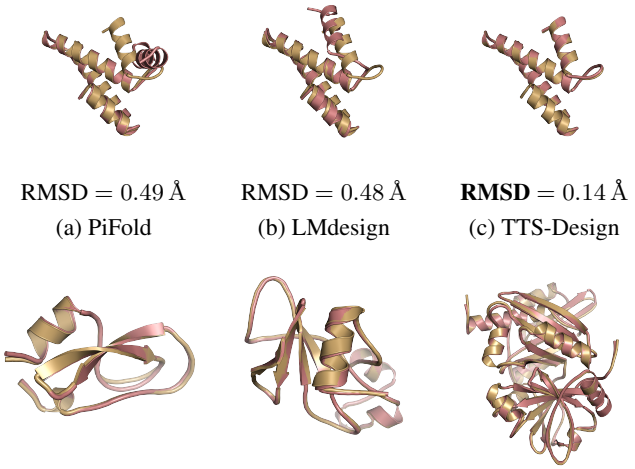
Figure 4: Target sequence of protein 1a32.A and generated sequences predicted by three different models.

We first take the protein 1a32.A from the CATH 4.2 test set as an example, which is a well-characterized protein that plays an essential role in bacterial ribosome assembly and translational regulation. Owing to its compact size, conserved fold, and clearly defined structure–function relationship, it has been widely used as a benchmark for studying protein. These properties make it an ideal case study for evaluating the accuracy and reliability of protein inverse folding.

As shown in Figure 4, given the native structure, we generate amino-acid sequences using PiFold, LMdesign, and *TTS-Design*, respectively. The sequence designed by *TTS-Design* achieves a notably higher sequence recovery rate of 67.06%, compared to 32.94% for PiFold and 36.47% for LMdesign. This demonstrates that *TTS-Design* can better capture the residue–structure relationships compared to other methods.

Next, we examine the predicted structures folded from these sequences using AlphaFold3 [Abramson *et al.*, 2024]. The results are visualized in Figure 5 (a-c), where the native structure is colored in yellow and the predicted structure is shown in red. The comparison reveals that the structure generated by *TTS-Design* exhibits the smallest deviation from the ground truth, achieving an RMSD of 0.14, which is significantly lower than that of PiFold (0.49) and LMdesign (0.48). This improvement confirms that *TTS-Design* not only enhances sequence quality but also maintains structural reliability when evaluated at the atomic level.

To provide a more comprehensive analysis of *TTS-Design*, we further visualize its performance across proteins with different sequence lengths in Figure 5 (d-f). Specifically, proteins from the CATH 4.2 test set are categorized into three groups according to sequence length: short proteins (length < 100), medium proteins ($100 \leq$ length < 300), and long proteins (length ≥ 300). For each group, we randomly select one protein and generate sequence using *TTS-Design* and predict structure utilizing AlphaFold3. In Figure 5, we show proteins for all groups. We then evaluate the designed sequences in terms of sequence recovery and RMSD. As shown in Figure 5 (d-f), *TTS-Design* consistently produces sequences with high recovery and low RMSD across all three length groups. In particular, the medium-length protein achieves higher recovery with very low RMSD, while the long protein, despite increased structural complexity, still maintains stable recovery and reasonable RMSD values. These results show that *TTS-Design* remains effective across different protein lengths.



RMSD $= 0.49$ Å (a) PiFold
RMSD $= 0.48$ Å (b) LMdesign
RMSD $= 0.14$ Å (c) TTS-Design

RMSD $= 0.383$ Å **Recovery** $= 0.54$ (d) Short
RMSD $= 0.245$ Å **Recovery** $= 0.73$ (e) Medium
RMSD $= 0.333$ Å **Recovery** $= 0.64$ (f) Long

Figure 5: (a-c): comparison of ground truth structure in yellow and predicted structures in red generated by PiFold, LMdesign, and TTS-Design for the same target protein 1a32.A. (d-f): Structural comparisons for proteins with different lengths. Short, medium, and long proteins.

Overall, these visual and quantitative comparisons demonstrate that *TTS-Design* produces amino acid sequences that are not only more consistent with the target sequence but also yield folded conformations that are physically realistic and functionally coherent. This highlights the potential of test-time compute scaling as a general strategy for enhancing protein design performance under data-limited scenarios.

5 Conclusion

In this work, we introduced *TTS-Design*, a novel framework that integrates test-time scaling into protein inverse folding task. The core component of our approach is a dedicated reward model, *IF-RM*, which quantitatively measures structure–sequence compatibility at inference stage. By combining a pretrained *IF-RM* with existing PIF models, *TTS-Design* effectively improves sequence recovery without modifying model size. For future work, we plan to design more effective reward models, and explore more sequence selection strategies to further enhance the quality of designed proteins.

Acknowledgments

This work was partially supported by Australian Research Council (ARC) through grants FT210100097 and DP240101547.

References

[Abramson *et al.*, 2024] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024.

[Bai *et al.*, 2025] Peizhen Bai, Filip Miljković, Xianyuan Liu, Leonardo De Maria, Rebecca Croasdale-Wood, Owen Rackham, and Haiping Lu. Mask-prior-guided denoising diffusion improves inverse protein folding. *Nature Machine Intelligence*, pages 1–13, 2025.

[Berman and Burley, 2025] Helen M Berman and Stephen K Burley. Protein data bank (PDB): Fifty-three years young and having a transformative impact on science and society. *Quarterly Reviews of Biophysics*, 58:e9, 2025.

[Bianchi *et al.*, 2020] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *ICML*, pages 874–883, 2020.

[Bio and Biswas, 2025] Nabla Bio and Surojit Biswas. De novo design of hundreds of functional gpcr-targeting antibodies enabled by scaling test-time compute. *bioRxiv*, pages 2025–05, 2025.

[Consortium, 2025] The UniProt Consortium. Uniprot: the universal protein knowledgebase in 2025. *Nucleic Acids Research*, 53(D1):D609–D617, 2025.

[Dauparas *et al.*, 2022] 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. *Science*, 378(6615):49–56, 2022.

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

[Gao *et al.*, 2023] Zhangyang Gao, Cheng Tan, and Stan Z Li. Pifold: Toward effective and efficient protein inverse folding. In *ICLR*, 2023.

[Guan *et al.*, 2026] Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Marco Siniscalchi, Ming Jin, and Shirui Pan. Timeomni-1: Incentivizing complex reasoning with time series in large language models. In *ICLR*, 2026.

[Hsu *et al.*, 2022] 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 *ICML*, pages 8946–8970, 2022.

[Ingraham *et al.*, 2019] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *NeurIPS*, 2019.

[Jumper *et al.*, 2021] 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.

[Kim *et al.*, 2025] Heajin Kim, Chae-Won Lee, Eunmi Choi, and Heisook Lee. Exploring public perspectives on synthetic biology and its integration with sex/gender characteristics. *Humanities and Social Sciences Communications*, 12(1):1–10, 2025.

[Landwehr *et al.*, 2025] Grant M Landwehr, Jonathan W Bogart, Carol Magalhaes, Eric G Hammarlund, Ashty S Karim, and Michael C Jewett. Accelerated enzyme engineering by machine-learning guided cell-free expression. *Nature Communications*, 16(1):865, 2025.

[Leaver-Fay *et al.*, 2011] Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian W Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, et al. Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. *Methods in Enzymology*, 487:545–574, 2011.

[Lin *et al.*, 2022] 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*, 2022:500902, 2022.

[Qiu *et al.*, 2024] Jiezhong Qiu, Junde Xu, Jie Hu, Hanqun Cao, Liya Hou, Zijun Gao, Xinyi Zhou, Anni Li, Xiujuan Li, Bin Cui, et al. Instructplm: Aligning protein language models to follow protein structure instructions. *bioRxiv*, pages 2024–04, 2024.

[Savage, 2023] Neil Savage. Drug discovery companies are customizing chatgpt: here's how. *Nature Biotechnology*, 2023.

[Snell *et al.*, 2024] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[Watson *et al.*, 2023] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[Wu *et al.*, 2024] Taoyu Wu, Yu Guang Wang, and Yiqing Shen. Lagdif: Latent graph diffusion model for efficient protein inverse folding with self-ensemble. In *BIBM*, pages 3850–3855, 2024.

[Yi *et al.*, 2023] Kai Yi, Bingxin Zhou, Yiqing Shen, Pietro Liò, and Yuguang Wang. Graph denoising diffusion for inverse protein folding. In *NeurIPS*, pages 10238–10257, 2023.

[Zhang *et al.*, 2025] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

[Zheng *et al.*, 2023] Zaixiang Zheng, Yifan Deng, Dongyu Xue, Yi Zhou, Fei Ye, and Quanquan Gu. Structure-informed language models are protein designers. In *ICML*, pages 42317–42338, 2023.

[Zheng *et al.*, 2025] Yizhen Zheng, Huan Yee Koh, Jiaxin Ju, Madeleine Yang, Lauren T May, Geoffrey I Webb, Li Li, Shirui Pan, and George Church. Large language models for drug discovery and development. *Patterns*, 2025.