PRISM: ENHANCING PROTEIN INVERSE FOLDING THROUGH FINE-GRAINED RETRIEVAL ON STRUCTURE-SEQUENCE MULTIMODAL REPRESENTATIONS

Sazan Mahbub[*1], **Souvik Kundu**[†2], **Eric P. Xing**[†1,3,4]

[1]Carnegie Mellon University, School of Computer Science
[2]Intel Labs
[3]Mohamed bin Zayed University of AI
[4]GenBio AI

ABSTRACT

3D structure-conditioned protein sequence generation, also known as protein inverse folding, is a key challenge in computational biology. While large language models for proteins have made significant strides, they cannot dynamically integrate rich multimodal representations from existing datasets, specifically the combined information of 3D structure and 1D sequence. Additionally, as datasets grow, these models require retraining, leading to inefficiencies. In this paper, we introduce PRISM, a novel retrieval-augmented generation (RAG) framework that enhances protein sequence design by dynamically incorporating fine-grained multimodal representations from a larger set of known structure-sequence pairs. Our experiments demonstrate that PRISM significantly outperforms state-of-the-art techniques in sequence recovery, emphasizing the advantages of incorporating fine-grained, multimodal retrieval-augmented generation in protein design.

1 INTRODUCTION

Proteins are fundamental to biological functions, with their three-dimensional (3D) structure governing their activity. This structure is determined by the amino acid sequence, making the inverse folding problem—designing a sequence that adopts a target structure—a central challenge in protein design. However, due to the intricate and non-deterministic nature of structure-to-sequence mapping, traditional computational approaches are often impractical.

Deep learning has emerged as the predominant approach for structure-conditioned sequence design, leveraging graph neural networks (GNNs) for 3D structural encoding (Jing et al., 2020; Dauparas et al., 2022; Ingraham et al., 2019; Mahbub & Bayzid, 2022; Alam et al., 2024) and generative approaches, including autoregressive (Dauparas et al., 2022; Joshi et al., 2024), conditional masked language modeling (Ghazvininejad et al., 2019; Zheng et al., 2023), and discrete diffusion (Wang et al., 2024a; Sun et al., 2024; Ellington et al., 2024; Zou et al., 2024), for sequence generation. However, existing methods primarily learn in a *static* manner, with limited efforts to dynamically incorporate fine-grained, multimodal protein representations of training samples—specifically, the joint information of 3D structure and 1D sequence—during inference. Recent work by Wang et al. (2024b) explored retrieval-augmented generation (RAG) to leverage existing protein databases during inference for antibody design. However, their approach relies heavily on traditional full-structure similarity search, which is computationally expensive and lacks fine granularity. Moreover, their retrieval mechanism is predominantly structure-based, overlooking the integrated role of sequence information.

In this study, we introduce PRISM, a novel multimodal RAG framework for protein inverse folding that dynamically retrieves and integrates fine-grained multimodal protein representations in a computationally efficient manner. PRISM fully leverages both 3D structural and sequence information, capturing rich local and global dependencies. Experimental results demonstrate that PRISM substantially outperforms the other state-of-the-art protein inverse folding methods in sequence recovery.

[*]smahbub@cs.cmu.edu
[†]Corresponding authors: souvikk.kundu@intel.com, epxing@cs.cmu.edu

2 METHOD

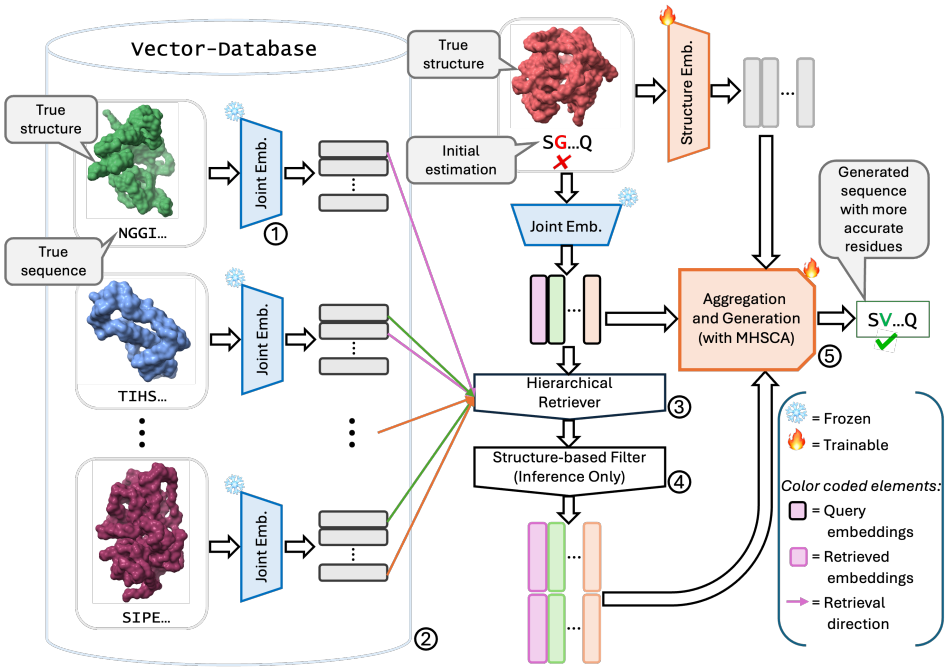

Figure 1: The overall pipeline of our proposed framework PRISM.

In this section we discuss our proposed RAG framework, namely PRISM. The overview of our framework is demonstrated in Figure 1.

2.1 STRUCTURE-SEQUENCE MULTIMODAL REPRESENTATION

We start with a function $\mathcal{G}$ capable of jointly encoding *two modalities* of data–the 3D structure $\mathcal{X}$ and 1D sequence S of any protein P (Figure 1, Point ①). Specifically,

$$\mathcal{E} = \mathcal{G}(P) = \mathcal{G}(\mathcal{X}, S) \in \mathbb{R}^{N \times d}, \tag{1}$$

where $\mathcal{E}$ is the joint embedding, N is the number of residues in the protein P, and d is the embedding dimension. Here $\mathcal{X}$ is the 3D structure of the protein, which can be represented as a nearest neighbor graph, $\mathcal{X}(V, E)$, where V and E respectively represent the nodes and the edges of this graph ($|V| = N$). Also $S = [S_1, S_2, \ldots, S_N]$ is the sequence of amino-acid residues S_j in P. The joint embedding matrix, $\mathcal{E} \in \mathbb{R}^{N \times d}$, consists of N vectors, each representing an amino acid in the protein P. Each vector, $\mathcal{E}_j \in \mathbb{R}^d$, captures the *context* of the 3D neighborhood surrounding the j-th residue as well as its long-range dependencies with other parts of the protein. This approach differs from many retrieval-augmented generation (RAG) methods, which segment sequences and represent each segment with a single vector, potentially losing granularity essential for accurate protein representation.

The sequence S can be effectively encoded using pre-trained protein language models (Sun et al., 2024; Nadav et al., 2023; Wang et al., 2024a), while the graph $\mathcal{X}(V, E)$ can be encoded with standard protein 3D structure encoders (Dauparas et al., 2022; Gao et al., 2022; Jing et al., 2020). Some methods jointly encode the structure and sequence of proteins, leveraging both pre-trained language models and structure encoders (Zheng et al., 2023; Wang et al., 2024a; Sun et al., 2024). In this study, we utilize AIDO.ProteinIF (Sun et al., 2024) as the multimodal encoding function $\mathcal{G}(.)$, however, we note that our proposed method is agnostic to any specific embedding model.

2.2 VECTOR-DATABASE

Here we will discuss how we create our vector-database (Vector-DB). We assume that we have access to the structures and sequences of a set of M proteins $\mathbb{P} = \{P^i : i \in [1, M]\} = \{(\mathcal{X}^i(V^i, E^i), S^i) :$

$i \in [1, M]\}$, ideally used as a training set. We embed each protein $P^i \in \mathbb{P}$ with the function $\mathcal{G}$ (Equation 1) and get a set of embedding matrices $\mathbb{E} = \{\mathcal{E}^i : i \in [1, M]\}$. Now we create our vector database as a set of *mappings*, $\mathbb{V} = \{\langle \mathcal{E}^i_j \Rightarrow f^i_j \rangle : i \in [1, M]$ *and* $j \in [1, |P^i|]\}$, where each element $\langle \mathcal{E}^i_j \Rightarrow f^i_j \rangle$ is a mapping from an embedding vector $\mathcal{E}^i_j$ to a 3D structure fragment f^i_j, consisting of r closest residues of the j-th residue in protein P^i (r is a hyperparameter). Later we use these fragments for structure-based filtering (discussed later in Section 2.4). Our vector-database is demonstrated in Figure 1, Point ②.

2.3 Hierarchical Retriever

For inverse folding, during inference we only know the structure $\mathcal{X}^q$ of a query protein P^q, which we use to generate a sequence. The query proteins should also be encoded by the same joint-embedding function $\mathcal{G}$ form Equation 1. However, since we do not know any sequence for our query protein *a priori*, we can estimate an initial version of the sequence with any off-the-shelf inverse folding method (Sun et al., 2024; Zheng et al., 2023; Wang et al., 2024a), which we can further improve with our RAG-based framework. We choose AIDO.ProteinIF (Sun et al., 2024) as our initial sequence estimator (our approach is agnostic to any other such method). The initial estimate $\hat{S}^q$ is then used to compute a *crude* representation of our query protein as $\hat{\mathcal{E}}^q = \mathcal{G}(\mathcal{X}^q, \hat{S}^q)$. For each vector $\hat{\mathcal{E}}^q_l$, we retrieve a set of the top K most similar entities $\mathbb{V}^{[q,l]} \in \mathbb{V}$ based on cosine-similarity, where K is a hyperparameter and $|\mathbb{V}^{[q,l]}| = K$. We note that $\hat{\mathcal{E}}^q$ is likely to be a less accurate representation of the protein P^q. However, from our analysis we find that our retriever can usually use this embedding to retrieve structurally similar fragments from other proteins in $\mathbb{V}$, as shown in an example in Appendix D, Figure 3. We also note that the complexity of search in a naive retrieval process would be $O\big(\sum_{i=1}^{M} |P_i|\big)$, which can be quite expensive for large M. Especially the search can become prohibitive if we want to use a very large training set (e.g., millions of proteins from AlphaFoldDB (Varadi et al., 2022; John et al., 2021)) or when computational resource is limited. To address this we take a hierarchal retrieval approach (Malkov & Yashunin, 2018). This allows us to retrieve entities form $\mathbb{V}$ in $O\big(\log(\sum_{i=1}^{M} |P_i|)\big)$ time, which is much faster than the naive approach. Our retriever module is shown as Point ③ in Figure 1. Future work will explore the potential of different types of retrieval techniques.

2.4 Structure-based Filtering

As discussed above, we retrieve relevant entities from our vector database based on a crude representation $\hat{\mathcal{E}}^q$ of our query protein P^q. However, we note that the perfomance of such retrieval will partly depend on the accuracy of the initial estimation $\hat{S}_p$. In order to relax such dependency, we further filter out a small subset of the retrieved entities in $\mathbb{V}^{[q,l]}$ using direct 3D structure similarity search, using a widely used tool named MASTER (Zhou & Grigoryan, 2015; 2020) (Figure 1, Point ④). Specifically, we first extract the structure fragment f^q_l that consists of r nearest neighbors of the l-th residue in our query protein P^q. Then, for each element (a mapping) $\langle \mathcal{E}^i_j \Rightarrow f^i_j \rangle \in \mathbb{V}^{[q,l]}$, we compute the structural similarity score between f^i_j and the query fragment f^q_l (root mean square distance of locally aligned structures). Based on these scores, we take the top $\tilde{K}$ entities from $\mathbb{V}^{[q,l]}$ and end up with a subset $\tilde{\mathbb{V}}^{[q,l]} \in \mathbb{V}^{[q,l]} \in \mathbb{V}$, where $\tilde{K} = |\tilde{\mathbb{V}}^{[q,l]}| < K$. This further refines the set of the entities we retrieve to be highly relevant to the query protein. Unlike Wang et al. (2024b), we apply traditional structure search only on the retrieved subset of fragments, which makes it computationally much efficient. Moreover, we apply structure-based filtering only during inference, which further reduces the total computational cost.

2.5 Aggregation and Generation

We aggregate the filtered entities $\tilde{\mathbb{V}}^{[q,l]}$ $\forall l \in [1, |\hat{S}^q|]$ to generate a new sequence $\tilde{S}^q$. We do this with a series of L consecutive learnable blocks, each consisting of one multihead self-attention layer (MHSA), one multihead cross-attention layer (MHCA), and two bottleneck multilayer perceptrons (Houlsby et al., 2019) (L is a hyperparameter). In the rest of this article, we refer to this hybrid block as multihead self-cross attention block (MHSCA).

As shown in Figure 1 (Point ⑤) and in Appendix E (Figure 4), for $\forall l \in [1, |\hat{S}^q|]$ we first extract the embedding vectors $\{(\mathcal{E}^i_j)_k : k \in [1, \tilde{K}]\}$ from our retrieved entities in $\tilde{\mathbb{V}}^{[q,l]}$, where $(\mathcal{E}^i_j)_k$ is the *key* of the mapping $\tilde{\mathbb{V}}^{[q,l]}_k$. We then merge them with the query vector $\mathcal{E}^q_l$ and linearly project the output

to create a matrix $\mathcal{H}_l^q \in \mathbb{R}^{(\tilde{K}+1)\times d'}$ as,

$$\mathcal{H}_l^q = \text{concat}(\{(\mathcal{E}_j^i)_k : k \in [1, \tilde{K}]\} \cup \{\mathcal{E}_l^q\})\, W_{\mathcal{H}}, \tag{2}$$

where concat(.) performs a concatenation operation on the vectors in the union set, $W_{\mathcal{H}} \in \mathbb{R}^{d\times d'}$ is a learnable parameter, and d' is the output embedding dimension. Then for the whole query protein P^q we get a tensor $\mathcal{H}^q = [\mathcal{H}_1^q, \mathcal{H}_2^q, \ldots, \mathcal{H}_{|\hat{S}^q|}^q] \in \mathbb{R}^{|\hat{S}^q|\times(\tilde{K}+1)\times d'}$, which is used as *query*, *key*, and *value* of MHSA (see Vaswani (2017) for definitions). To ensure that the generator can effectively leverage any residual 3D structural information, we also encode the input structure $\mathcal{X}^q$ separately using a structural encoder, where *no sequence* information is provided. Similar to AIDO.ProteinIF (Sun et al., 2024), we leverage ProteinMPNN-CMLM (Zheng et al., 2023) for structure encoding, which is a variant of the original ProteinMPNN method (Dauparas et al., 2022) trained with conditional masked language modeling objective (Ghazvininejad et al., 2019). This generates structural encoding $\rho^q \in \mathbb{R}^{|\hat{S}^q|\times d^\rho}$. This encoding is then linearly transformed and merged with a linear projection of query encoding $\mathcal{E}^q$, creating a new representation matrix $\theta^q \in \mathbb{R}^{|\hat{S}^q|\times d'}$, where each element $\theta_l^q = \text{concat}(\{\rho_l^q W_\rho, \mathcal{E}_l^q W_{\mathcal{E}}\}) \in \mathbb{R}^{d'}$, with two learnable parameters $W_\rho \in \mathbb{R}^{d^\rho \times \frac{d'}{2}}$ and $W_{\mathcal{E}} \in \mathbb{R}^{d\times\frac{d'}{2}}$. For our MHCA blocks, we use θ^q as the *query*, and $\mathcal{H}^q$ as both the *key* and *value*. The motivation behind such design of MHSCA is, while MHSA layers can help jointly attend to multiple parts of the input protein as well as their corresponding retrieved embeddings, MHCA can help extract any kind of residual structural information needed to better decode the sequence. Moreover, since the MHCA here preserves the same dimension as θ^q, the output representation has $|\hat{S}^q|$ vectors which we can directly pass through another linear layer to generate the output logits $\mathcal{Y} \in \mathbb{R}^{|\hat{S}^q|\times d'}$. Sampling with $\mathcal{Y}$ provides us with a newly generated sequence $\tilde{S}^q$.

3 Results and Discussion

For our experiments, we leverage the widely used CATH-4.2 dataset (Orengo et al., 1997). This is a standard benchmark datatset where all the sequences are within 500 residues length. In Appendix C.1, in Figure 2 and Table 2 we show the distribution of sequence lengths and statistics on this dataset, respectively.

Table 1: **Evaluation of various protein inverse folding techniques**. Sequence recovery rates (SRR) of earlier methods are referenced from Wang et al. (2024a) and Sun et al. (2024). The best and the second best scores are shown in bold and italic fonts. Here "pLM" is the acronym for "protein language model".

Method Name	Uses pLM?	Uses RAG?	SRR
StructTrans (Ingraham et al., 2019)	×	×	35.82 %
GVP (Jing et al., 2020)	×	×	39.47 %
ProteinMPNN (Dauparas et al., 2022)	×	×	45.96 %
PiFold (Gao et al., 2022)	×	×	51.66 %
ProteinMPNN-CMLM (Zheng et al., 2023)	×	×	48.62 %
LM-Design (Zheng et al., 2023)	✓	×	54.41 %
DPLM (Wang et al., 2024a)	✓	×	54.54 %
AIDO.ProteinIF (Sun et al., 2024)	✓	×	*58.60* %
PRISM (Ours)	✓	✓	**60.28** %

In Table 1 and Appendix F, we present a quantitative comparison among different state-of-the-art (SoTA) methods for protein inverse folding. We show that, with the help of our proposed RAG framework, we can significantly improve the sequence recovery rate on CATH-4.2 benchmark dataset. Specifically, PRISM achieves a recovery rate about 1.7% higher compared to the next best performing method AIDO.ProteinIF (Sun et al., 2024).

4 Conclusion

In this study, we present PRISM, a multimodal retrieval-augmented generation framework aimed at improving protein inverse folding by dynamically incorporating finegrained structure-sequence multimodal representations from a larger protein database, resulting in enhanced sequence recovery over current methods. Future work will involve testing on additional datasets and examining the generated sequences for notable insights.

REFERENCES

Ramisa Alam, Sazan Mahbub, and Md Shamsuzzoha Bayzid. Pair-egret: enhancing the prediction of protein–protein interaction sites through graph attention networks and protein language models. *Bioinformatics*, 40(10):btae588, 2024.

Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. Evaluation metrics for language models. 1998.

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.

Caleb N Ellington, Ning Sun, Nicholas Ho, Tianhua Tao, Sazan Mahbub, Dian Li, Yonghao Zhuang, Hongyi Wang, Le Song, and Eric P Xing. Accurate and general dna representations emerge from genome foundation models at scale. *bioRxiv*, pp. 2024–12, 2024.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023a.

Zhangyang Gao, Cheng Tan, Pablo Chacón, and Stan Z Li. Pifold: Toward effective and efficient protein inverse folding. *arXiv preprint arXiv:2209.12643*, 2022.

Zhangyang Gao, Cheng Tan, Yijie Zhang, Xingran Chen, Lirong Wu, and Stan Z Li. Proteininvbench: Benchmarking protein inverse folding on diverse tasks, models, and metrics. *Advances in Neural Information Processing Systems*, 36:68207–68220, 2023b.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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. *ICML*, 2022. doi: 10.1101/2022.04.10.487779. URL https://www.biorxiv.org/content/early/2022/04/10/2022.04.10.487779.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2020.

Jumper John, Evans Richard, Pritzel Alexander, Green Tim, Figurnov Michael, Ronneberger Olaf, Tunyasuvunakool Kathryn, Bates Russ, Žídek Augustin, Potapenko Anna, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

Chaitanya K Joshi, Arian R Jamasb, Ramon Viñas, Charles Harris, Simon Mathis, Alex Morehead, Rishabh Anand, and Pietro Liò. grnade: Geometric deep learning for 3d rna inverse design. *bioRxiv*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Camarillo-Guerrero Luis, F, Almeida Alexandre, Rangel-Pineros Guillermo, Finn Robert, D, and Lawley Trevor, D. Massive expansion of human gut bacteriophage diversity. *Cell*, 2021.

Sazan Mahbub and Md Shamsuzzoha Bayzid. Egret: edge aggregated graph attention networks and transfer learning improve protein–protein interaction site prediction. *Briefings in Bioinformatics*, 23(2):bbab578, 2022.

Sazan Mahbub, Caleb Ellington, Sina Alinejad, Kevin Wen, Yingtao Luo, Ben Lengerich, and Eric P Xing. From one to zero: Rag-im adapts language models for interpretable zero-shot clinical predictions. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, 2021. doi: 10.1101/2021.07.09.450648. URL https://www.biorxiv.org/content/10.1101/2021.07.09.450648v1.

Clara Meister and Ryan Cotterell. Language model evaluation beyond perplexity. *arXiv preprint arXiv:2106.00085*, 2021.

Brandes Nadav, Goldman Grant, Wang Charlotte, H., Ye Chun, Jimmie, and Ntranos Vasilis. Genome-wide prediction of disease variant effects with a deep protein language model. *Nature Genetics*, 2023.

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.

Ning Sun, Shuxian Zou, Tianhua Tao, Sazan Mahbub, Dian Li, Yonghao Zhuang, Hongyi Wang, Xingyi Cheng, Le Song, and Eric P Xing. Mixture of experts enable efficient and effective protein understanding and design. *bioRxiv*, pp. 2024–11, 2024.

Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al. Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic acids research*, 50(D1):D439–D444, 2022.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024a.

Zichen Wang, Yaokun Ji, Jianing Tian, and Shuangjia Zheng. Retrieval augmented diffusion model for structure-informed antibody design and optimization. *arXiv preprint arXiv:2410.15040*, 2024b.

Zaixiang Zheng, Yifan Deng, Dongyu Xue, Yi Zhou, Fei Ye, and Quanquan Gu. Structure-informed language models are protein designers. In *International conference on machine learning*, pp. 42317–42338. PMLR, 2023.

Jianfu Zhou and Gevorg Grigoryan. Rapid search for tertiary fragments reveals protein sequence–structure relationships. *Protein Science*, 24(4):508–524, 2015.

Jianfu Zhou and Gevorg Grigoryan. A c++ library for protein sub-structure search. *bioRxiv*, pp. 2020–04, 2020.

Shuxian Zou, Tianhua Tao, Sazan Mahbub, Caleb N Ellington, Robin Algayres, Dian Li, Yonghao Zhuang, Hongyi Wang, Le Song, and Eric P Xing. A large-scale foundation model for rna function and structure prediction. *bioRxiv*, pp. 2024–11, 2024.

A RELATED WORK

Protein inverse folding, the process of designing amino acid sequences that fold into specific three-dimensional structures, has been a focal point of computational biology research. In 2022, Dauparas et al. (2022) proposed ProteinMPNN, widely popular autoregressive method for designing protein sequences that fold into desired structures. It achieved an impressive sequence recovery rate on native backbones, outperforming traditional methods, showing versatility extending to designing monomers, cyclic oligomers, nanoparticles, and target-binding proteins. Gao et al. (2022) introduced PiFold, a method that effectively combines expressive features with an autoregressive sequence decoder to enhance both the accuracy and efficiency of protein design. PiFold achieved a high recovery rate on the benchmark dataset and demonstrated a speed advantage, being 70 times faster than some autoregressive counterparts. That same year, Hsu et al. (2022) proposed a sequence-to-sequence transformer model trained using predictions by AlphaFold2, a state-of-the-art structure prediction method (John et al., 2021). By leveraging putative structures of millions of proteins, their approach achieved a notable improvement in the field. Zheng et al. (2023) introduced the usage of protein language models (Nadav et al., 2023; Meier et al., 2021) for structure-conditioned protein sequence design, or in other words, inverse folding. Another work by Wang et al. (2024a) extended this by incorporating diffusion language modeling for effective sequence generation. Sun et al. (2024) pretrained a 16 billion parameter protein language model with a mixture-of-expert architecture, which they further adapted for prediction and sequence generation tasks, and surpassing the previous methods. To address the need for standardized evaluation, Gao et al. (2023b) also proposed ProteinInvBench, a comprehensive benchmark for protein design. This framework includes extended design tasks, integrated models, and diverse evaluation metrics, facilitating more rigorous comparisons across different methods.

B PRELIMINARIES

B.1 PROBLEM DEFINITION

The protein inverse folding problem can be defined as a mapping from a set of 3D atomic coordinates of the protein's conformation, denoted as $P = \{p_1, p_2, \ldots, p_n\}$, where each $p_i \in \mathbb{R}^3$ represents the 3D position of the i-th atom in the protein structure, to a sequence of amino acids $S = [s_1, s_2, \ldots, s_n]$, where $s_i \in \{$A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y$\}$ represents the i-th residue in the sequence. This problem is typically solved by learning a mapping function f, where $f : \mathbb{R}^{3n} \rightarrow \mathbb{R}^{20n}$, which maps the 3D structure to a sequence of probabilities over the 20 possible amino acids for each position. Deep learning models, especially convolutional neural networks (CNNs) and graph neural networks (GNNs), are commonly used to capture the spatial and sequential dependencies between the amino acids and their 3D structural context (Gao et al., 2022; Dauparas et al., 2022; Jing et al., 2020; Hsu et al., 2022).

In graph-based approaches to represent protein structures, the 3D structure is represented as a graph $G = (V, E)$, where V is the set of vertices corresponding to the amino acid residues in the sequence, and E represents the edges that encode interactions between residues based on their spatial proximity or other physicochemical relationships (Dauparas et al., 2022; Jing et al., 2020). Each node $v_i \in V$ encodes the 3D coordinates of the corresponding amino acid residue s_i, while the edges $e_{ij} \in E$ capture the interactions between residues i and j, typically modeled by geometric proximity or pairwise interactions.

Given a new input structure, the trained model predicts the sequence that best corresponds to this structure by generating the sequence either step-by-step (by autoregressive models, typically by selecting the most probable amino acid at each position) or the full sequence all at once (by non-autoregressive models). Sequence generation may also involve sampling techniques, such as Monte Carlo methods or variational autoencoders, to explore the space of possible sequences that can fold into the same or similar structures (Wang et al., 2024a; Dauparas et al., 2022). This approach enables the design of novel protein sequences with desired structural properties, which is useful for applications like protein engineering and drug design (Orengo et al., 1997; Gao et al., 2022).

B.2 RETRIEVAL AUGMENTED GENERATION (RAG)

Retrieval-Augmented Generation (RAG) is a hybrid approach that combines retrieval-based methods with generative models to improve the performance of tasks such as question answering, text generation, and information synthesis (Lewis et al., 2020; Mahbub et al.; Gao et al., 2023a). Specifically, RAG can be defined as a two-step process: retrieval followed by generation. Let Q denote a query or

input context, which could be a question, prompt, or incomplete information requiring completion. The goal is to generate a response Y, which is typically a natural language text or an answer to the query, informed by relevant external knowledge.

In the retrieval step, a set of candidate documents or entities $D = \{d_1, d_2, \ldots, d_m\}$ is retrieved from a large knowledge base $\mathcal{K}$, based on their relevance to the query Q (Lewis et al., 2020; Wang et al., 2024b). This retrieval process can be formulated as selecting a subset of documents $D \subseteq \mathcal{K}$, where each document d_i is scored based on its similarity to the query Q using a retrieval function $r : \mathcal{K} \times Q \to [0, 1]$, such that,

$$D = \text{Retrieve}(Q, \mathcal{K}) = \{d_i | r(Q, d_i) \geq \varphi\},$$

where φ is a threshold determining the relevance of each document. The retrieved documents D are then used as context in the subsequent generation phase.

In the generation step, a generative model G is used to generate the output Y given the query Q and the retrieved entities D,

$$Y = G(Q, D; \theta_G),$$

where G is typically a sequence-to-sequence model, such as a Transformer-based model, and θ_G represents the parameters of the generative model. The model is trained to maximize the likelihood of generating the correct response Y conditioned on both the query Q and the retrieved documents D. The final output Y is generated by sampling or decoding from the distribution $P(Y|Q, D)$. The integration of retrieval with generation allows the model to produce more informed and contextually relevant responses by utilizing external knowledge stored in the retrieval corpus.

C Experimental Setup

C.1 Dataset

For our experiments, we use the widely used dataset CATH-4.2 (Orengo et al., 1997). CATH 4.2 is a standard benchmark dataset where all the sequences are within 500 residues length. In Figure 2 we show the distribution of sequence lengths in CATH 4.2. This dataset is usually used for training, validation, as well as testing of inverse folding models (Zheng et al., 2023; Wang et al., 2024a). Previous studies have categorized experiments on this dataset into three groups considering short sequences, single-chain sequences, and all sequences in the CATH 4.2 test set. The short sequences category includes only those sequences with fewer than 100 amino acids, comprising approximately 16.5% of the sequences in the test set. The single-chain category focuses on sequences that correspond to a single entry in CATH 4.2, with the majority of sequences (around 92.86%) in the CATH 4.2 test set falling into this category.

Table 2: Statistics of CATH-4.2 dataset (Orengo et al., 1997). Here "St. Dev." stands for "standard deviation".

Data split	# of sequences	# of residues	Mean Length	Median Length	St. Dev. of Lengths
Train	18,024	3,941,775	218.7	204.0	109.93
Validation	608	105,926	174.22	146.0	92.44
Test	1,120	181,693	162.23	138.0	82.22
Combined	19,752	4,229,394	214.12	196.0	109.06

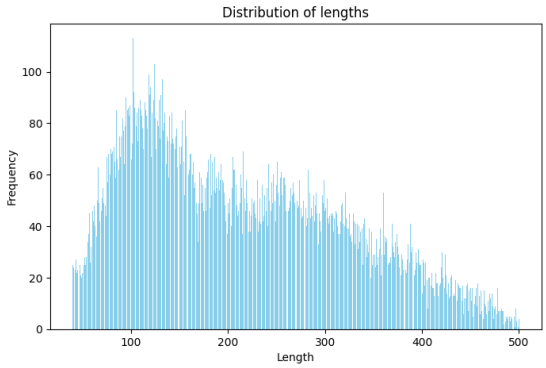

Figure 2: Distribution of lengths of the protein sequences in the benchmark dataset CATH-4.2 (Orengo et al., 1997).

C.2 Evaluation

We evaluate our model with two different metrics: median sequence recovery rate and perplexity. Median Sequence Recovery Rate (MSRR) is the most widely accepted metric to assess the accuracy of structure-conditioned protein sequence generation (Luis et al., 2021; Zheng et al., 2023; Wang et al., 2024a; Sun et al., 2024). It measures the median percentage of amino acids in the generated sequences that match the native sequence corresponding to a given protein backbone structure. The formula for calculating MSRR is shown in 3.

$$\text{MSRR} = \text{median}\left(\frac{1}{L}\sum_{i=1}^{L}\mathbb{I}(\hat{s}_i = s_i) \times 100\%\right), \tag{3}$$

where, L is the length of the protein sequence, s_i and $\hat{s}_i$ respectively represent the native and the predicted amino acids at position i, $\mathbb{I}(\hat{s}_i = s_i)$ is an indicator function that is 1 if $\hat{s}_i$ matches s_i, and 0 otherwise. In protein inverse folding, the goal is to design sequences that can fold into a specific three-dimensional structure while maintaining functional and stability constraints. A higher MSRR indicates that the model successfully captures the sequence-structure relationship and produces sequences that are evolutionarily plausible (Wang et al., 2024a; Jing et al., 2020). However, since natural proteins often exhibit sequence diversity while maintaining similar structures, MSRR alone may not fully capture the functional viability of generated sequences (Zheng et al., 2023).

Perplexity (PPL), a measure of model confidence, is a common metric in sequence modeling that quantifies how well a model predicts a given sequence (Chen et al., 1998; Meister & Cotterell, 2021). In protein inverse folding, lower perplexity implies that the model assigns high probability to native-like sequences, suggesting a strong alignment with the underlying distribution of natural protein sequences. It reflects how confidently the model predicts each amino acid given the structural context (Gao et al., 2022). Equation 4 shows the standard formulation to compute perplexity for autoregressive models.

$$\text{PPL}_{AR} = \exp\left(-\frac{1}{L}\sum_{i=1}^{L}\log P(s_i \mid s_{<i}, S_{\text{struct}})\right), \tag{4}$$

where, $P(s_i \mid S_{\text{struct}}, s_{<i})$ is the probability assigned by the model to the amino acid s_i given the protein structure S_{struct} and previous amino acids $s_{<i}$, L is the sequence length. However, since our approach is non-autoregressive, the corresponding formula would be,

$$\text{PPL}_{NAR} = \exp\left(-\frac{1}{L}\sum_{i=1}^{L}\log P(s_i|x, S_{\text{struct}})\right). \tag{5}$$

Here, x is some noisy (less accurate) version of the native sequence, $P(s_i|x_t, S_{\text{struct}})$ is the model's estimated probability of the amino acid s_i given the noisy representation x_t and the structure S_{struct}.

C.3 BASELINES

In this study, we consider eight highly performing baseline models for protein inverse folding: StructTrans (Ingraham et al., 2019), GVP (Jing et al., 2020), ProteinMPNN (Dauparas et al., 2022), ProteinMPNN-CMLM (Zheng et al., 2023), PiFold (Gao et al., 2022), LM-Design (Zheng et al., 2023), DPLM (Wang et al., 2024a), and AIDO.Protein (Sun et al., 2024). StructTrans (Ingraham et al., 2019) proposed a conditional generative model for protein sequences given 3D structures based on graph representations. GVP (Jing et al., 2020) introduced geometric vector perceptrons, which extend standard dense layers to operate on collections of Euclidean vectors. ProteinMPNN (Dauparas et al., 2022) proposes an autoregressive protein sequence generation approach conditioned on structure. ProteinMPNN-CMLM (Zheng et al., 2023), a non-autoregressive variant of the original ProteinMPNN, has been trained with the conditional masked language modeling (CMLM) objective (Ghazvininejad et al., 2019) and achieves higher score than the original version. LM-Design (Zheng et al., 2023) is another non-autoregressive model trained with CMLM that leverages pretrained protein language models for inverse folding. DPLM (Wang et al., 2024a) extends this work by using discrete diffusion language modeling objective to enhance sequence generation capabilities of languange models. AIDO.Protein (Sun et al., 2024) is a 16 billion parameter pretrained protein language model that has been further adapted for inverse folding with conditional discrete diffusion language modeling objective. In Table 1, we demonstrate how our proposed approach is different from existing structure-conditioned protein sequence generation approaches, along with the median sequence recovery rate on CATH 4.2 dataset for comparison.

D CASE STUDY

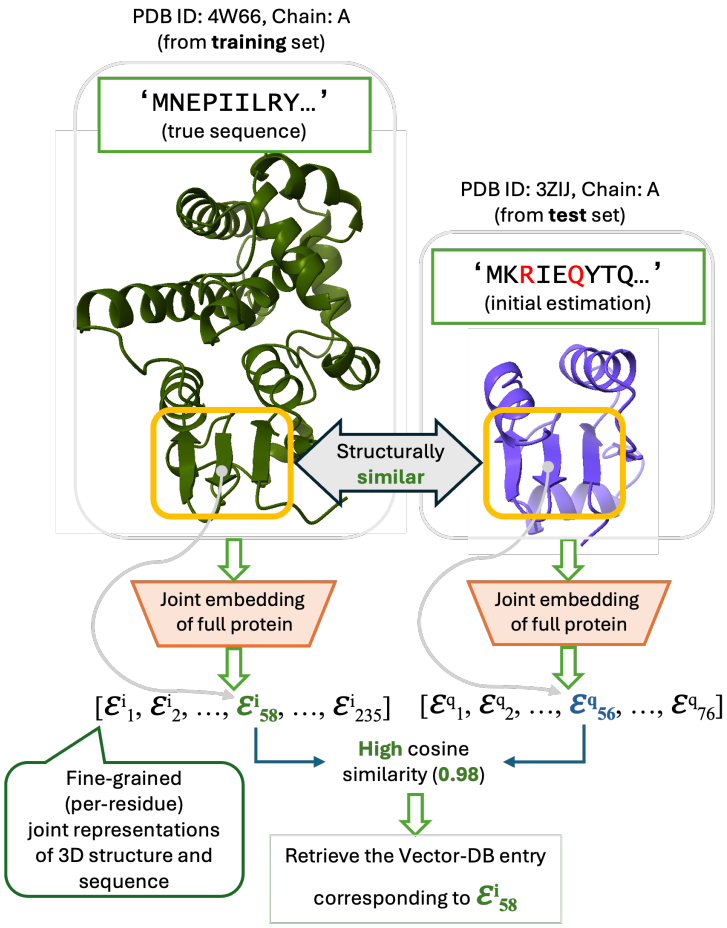

Figure 3: A example of how our retrieval process is actually aligning with true structure similarity search. This shows retrieval in such a way is indeed potential to extract rich information from already-existing larger protein databases. Here we leverage embedding $\mathcal{E}^i_j$ (fine-grained, at the very residue-level), which maps to a structure fragment f^i_j, consisting of the r-nearest neighbors (residues) of the j-th residue.

E FRAMEWORK

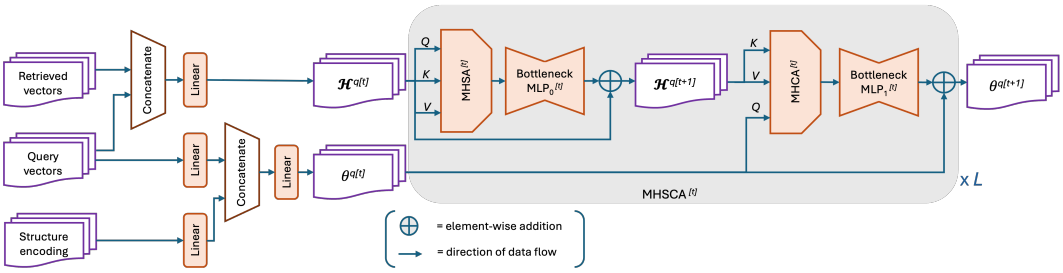

Figure 4: Our aggregation and generation module that uses L consecutive blocks of MHSCA. Here the super-script "$[t]$" corresponds to the *index* of the current MHSCA block (and also its components and their inputs).

F RESULTS AND DISCUSSION

Table 3: **Evaluation of different inverse folding methods on CATH-4.2 dataset (Orengo et al., 1997)**. We show comparison of our approach with existing state-of-the-art models for three different experiments. See Appendix C for details. The result of AIDO.Protein was produced using their publicly available codebase and model checkpoint. DPLM's scores were taked from Wang et al. (2024a). The rest of the scores were adopted from Zheng et al. (2023). The best and the second best scores are shown in bold and italic fonts. Here '-' means the score was not reported in the corresponding source.

Models	**Short**		**Single-chain**		**All**	
	PPL ↓	MSRR % ↑	PPL ↓	MSRR % ↑	PPL ↓	MSRR % ↑
StructTrans	8.39	28.14	8.83	28.46	6.63	35.82
GVP	7.23	30.60	7.84	28.95	5.36	39.47
ProteinMPNN	6.21	36.35	6.68	34.43	4.61	45.96
ProteinMPNN-CMLM	7.16	35.42	7.25	35.71	5.03	48.62
PiFold	6.04	*39.84*	6.31	38.53	4.55	51.66
LM-Design	7.01	35.19	6.58	40.00	4.41	54.41
DPLM	-	-	-	-	-	54.54
AIDO.Protein	*4.29*	38.46	*3.18*	*58.87*	*3.20*	*58.60*
PRISM (ours)	**3.98**	**41.30**	**3.04**	**60.45**	**3.05**	**60.28**

Our proposed approach, PRISM, demonstrates state-of-the-art performance in protein inverse folding on CATH 4.2, as evidenced by the results presented in Table 3. All the results here are generated with `argmax` to ensure deterministic sequence estimation. PRISM significantly outperforms existing methods across all categories (Short, Single-chain, and All) on the CATH-4.2 dataset in both Perplexity (PPL) and Median Sequence Recovery Rate (MSRR).

From Table 3, we see that PRISM achieves a PPL of 3.98 on short chains, surpassing the second-best model by 7.2%. This also attains an even lower PPL of 3.04 for the single-chains, with the next best model at 3.18. PRISM maintains superior generalization across the entire dataset with a PPL of 3.05, outperforming all other methods and reinforcing the effectiveness of our retrieval-augmented generation approach.

While comparing median sequence recovery rates, we see that PRISM achieves 41.30%, surpassing all baselines, with the closest competitor, PiFold, reaching 39.84%. For single-chains, PRISM achieves an impressive 60.45%, significantly exceeding the second-best model, AIDO.Protein (58.87%). On

the other hand, when considering all the sequences in the test set of CATH-4.2 dataset, PRISM's overall MSRR of 60.28% is about 1.7% higher than the next-best method.