FoldToken: Learning Protein Language via Vector Quantization and Beyond

Zhangyang Gao[1,2] *, Cheng Tan[1,2] *, Jue Wang[1,2], Yufei Huang[1,2], Lirong Wu[1,2], Stan Z.Li[1] †

[1]Westlake University
[2]Zhejiang University
gaozhangyang@westlake.edu.cn

Abstract

Is there a foreign language describing protein sequences and structures simultaneously? Protein structures, represented by continuous 3D points, have long posed a challenge due to the contrasting modeling paradigms of discrete sequences. We introduce FoldTokenizer to represent protein sequence-structure as discrete symbols. This approach involves projecting residue types and structures into a discrete space, guided by a reconstruction loss for information preservation. We name the learned discrete symbols as FoldToken, and the sequence of FoldTokens serves as a new protein language, transforming the protein sequence-structure into a unified modality. We apply the created protein language on general backbone inpainting task, building the first GPT-style model (FoldGPT) for sequence-structure co-generation with promising results. Key to our success is the substantial enhancement of the vector quantization module, Soft Conditional Vector Quantization (SoftCVQ).

Introduction

Sequence and structure modeling play a crucial role in protein applications, including mutation prediction (Umerenkov et al. 2022), sequence design (Ingraham et al. 2019; Jing et al. 2020; Tan et al. 2022; Gao et al. 2022; Zheng et al. 2023; Hsu et al. 2022; Gao, Tan, and Li 2023c,b), and function prediction (Hu et al. 2023; Zhang et al. 2022; Fan et al. 2022; Hu et al. 2024). However, the modality gap between sequence and structure usually requires different model architectures, i.e., sequence learners are typically based on transformers, while the structure learners are generally based on graph neural networks (GNNs). Constraints from modality-specific models, especially the structure prior required by GNNs, limit the modeling flexibility and prevent the applications of recent progress, such as GPT (Brown et al. 2020), on protein tasks. Given the success of applying Transformer to ViT (Dosovitskiy et al. 2020), where a image is converted to sequence of patches (image foreign language), we wonder *is there a way to unify the sequence and structure modalities by a protein foreign language?*

Recent work shows promise on structure to sequence transformation. IG-VAE (Eguchi, Choe, and Huang 2022) and FoldingDiff (Wu et al. 2024) suggest converting protein structures as sequences of folding angles and shows great potential in unconditional protein backbone generation. DiffSDS (Gao, Tan, and Li 2023a) extends the method in the scenarios of conditional backbone inpainting. These methods simplify 3D coordinates as angle sequence, allowing advanced transformers to be applied to structural modeling. However, continuous angles resist modeling as discrete symbols, unlike human language, hindering transformers in autoregressive generation where regression loss-trained models generally underperform compared to those trained by classification loss. (Pintea et al. 2023; Stewart, Bach, and Vert 2023; Horiguchi, Dohi, and Kawaguchi 2023). *Establishing a discrete protein language to bridge protein research with NLP remains an open challenge.*

Inspired by learning discrete vision language (Yu et al. 2021; Esser, Rombach, and Ommer 2021; Lee et al. 2022), we create a novel protein foreign language describing sequence and structure simultaneously, dubbled protein language. The key idea is to project the sequence and structure into a joint discrete space via FoldTokenizer then the sequence of discrete codes can serve as a new protein language to unify the sequence and structure modalities. To ensure the created protein language is informationally equivalent to original sequence-structure, we train the FoldTokenizer via reconstruction loss. Once the protein language is learned, it can be used for generative and predictive downstream tasks.

Recovering structures from the protein language poses a non-trivial challenge, necessitating the development of new VQ methods. After evaluating vanilla VQ (VVQ) (Van Den Oord, Vinyals et al. 2017) and recent lookup-free VQ (LFQ) (Yu et al. 2023), we reveal their proficiency in sequence but suboptimal structure reconstruction. We systemically analized their limitations and strengths, proposing targeted strategies for improvement. *The key to good reconstruction is soft querying across the entire codebook space.*

LFQ (Yu et al. 2023) has identified a trade-off between reconstruction and generation tasks - models excelling in reconstruction may underperform in downstream tasks. We further analize why the phenominon exist and propose imporved VQ method to overcome this challenge. For the first time, the proposed SoftCVQ method achieves good perfor-

*These authors contributed equally.
†Corresponding Author.

mance on both protein reconstruction and generation tasks, effectively addressing limitations seen in prior approaches.

We assess our model on both reconstruction and generative tasks. Our findings reveal a substantial enhancement in reconstruction quality with the proposed SoftCVQ method surpassing existing VQ methods. Leveraging the learned protein language, we introduce FoldGPT as the pioneering autoregressive sequence-structure co-generation model. Remarkably, FoldGPT outperforms comparable methods relying on sequences of continual angles, providing additional confirmation of the value of discretization.

Related Work

Protein Sequence-Structure Co-modeling has emerged as a vibrant area of research, aiming to create joint representations and enable the simultaneous generation of sequences and structures. Co-representation techniques integrate pretrained sequence models with graph neural networks pretrained on protein structures. These fused embeddings find applications in predictive tasks such as binding affinity and stability prediction (Zhang et al. 2023b; Samaga, Raghunathan, and Priyakumar 2021). Co-design techniques establish connections between sequence and structure generative models to generate novel sequences and structures from scratch (Shi et al. 2022; Song et al. 2023). These approaches highlight that sequences and structures contain complementary information, underscoring the need for further advancements in co-modeling methods to address challenges related to both aspects of protein folding and inverse-folding. A notable application of co-modeling is in antibody design (Jin et al. 2021; Luo et al. 2022; Kong, Huang, and Liu 2022; Tan, Gao, and Li 2023).

Vector Quantization is a powerful technique for data compression and generation. Vanilla VQ (Van Den Oord, Vinyals et al. 2017) compresses latent representations to the nearest codebook vector. Product quantization (PQ) (Jégou, Douze, and Schmid 2011; Chen, Li, and Sun 2020; El-Nouby et al. 2022) decomposes the space into a Cartesian product of low-dimensional subspaces, quantizing each subspace separately for nearest neighbor search. Residual quantization (RQ) (Juang and Gray 1982; Martinez, Hoos, and Little 2014; Lee et al. 2022; Zeghidour et al. 2021) iteratively quantizes vectors and their residuals, representing the vector as a stack of codes. Lookup-free VQ (LFQ) (Mentzer et al. 2023; Yu et al. 2023) quantizes each dimension to a small set of fixed values.

Angle-based Structure Representation Derived from local residue coordinate systems, invariant features such as bond angle, torsion angle, and bond length serve as natural representations of protein structures (Eguchi, Choe, and Huang 2022; Wu et al. 2024; Gao, Tan, and Li 2023a; Lee, Kim, and Kim 2022). They capture intrinsic geometric properties unaffected by translation or rotation, making them ideal for tasks like inverse folding and antibody design. To fit the paradigm of the transformer, we encode structural information as a sequence of locally folding angles. Formally, we define $f : \boldsymbol{x}_i \mapsto \boldsymbol{a}_i$, transforming Euclidean coordinate $\boldsymbol{x}_i$ to invariant representation $\boldsymbol{a}_i$, i.e., $\boldsymbol{a}_i = (r_i, \alpha_i, \beta_i)$, where r_i, α_i and β_i are bond length, bond angle and torsion angle.

Method

Overall framework

As shown in Figure.1, the overall pipeline include two models: (1) FoldTokenizer, learning the discrete protein language describing sequence and structure (2) FoldGPT, a NLP-style model using protein language and generating protein sequence-structure autoregressively.

FoldTokenizer includes encoder, quantizer and decoder, to jointly represent the residue types and C_α coordinates as discrete latent codes. Given the sequence and structure as $\mathcal{S} = \{s_i : 1 \le i \le n\}$ and $\mathcal{A} = \{\boldsymbol{a}_i \in \mathbb{R}^3 : 1 \le i \le n\}$, with protein length n. We formulate FoldTokenizer as:

$$\begin{cases} \mathbf{h} \sim q_{enc}(\mathbf{h}|\mathcal{S}, \mathcal{A}) \\ \mathbf{z} = Q(\mathbf{h}) \\ \hat{\mathcal{S}}, \hat{\mathcal{A}} = p_{dec}(\mathcal{S}, \mathcal{A}|Q^{-1}(\mathbf{z})) \end{cases} \tag{1}$$

The encoder $q_{enc}(\cdot)$ is a transformer-based model that maps protein sequence and structure into a continuous latent embedding $\mathbf{h} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n\} \in \mathbb{R}^{n,d}$. The quantizer $Q : \boldsymbol{h} \mapsto \boldsymbol{z}$ converts continual embeddings $\boldsymbol{h}$ as discrete codes $\boldsymbol{z} \in \mathbb{Z}^{n,1}$, whose reverse operation is $Q^{-1} : \boldsymbol{z} \mapsto \hat{\boldsymbol{h}}$. The decoding transformer $p_{dec}(\cdot)$ reconstructs the sequence and structure from the reversed latent embedding $\hat{\boldsymbol{h}}$.

FoldGPT We apply the created protein language to the problem of protein inpainting. Denote the masked backbone atoms as $\mathcal{M} = \{(\boldsymbol{a}_i, s_i)\}_{i \in \mathcal{I}_m}$, the known backbone atoms as $\mathcal{K} = \{(\boldsymbol{a}_i, s_i)\}_{i \in \mathcal{I}_k}$, where $\boldsymbol{a}_i$ and s_i are the coordinates and residue type of the i-th C_α atom. Here, $\mathcal{I}_m$ and $\mathcal{I}_k$ denote the indices of unknown and known residues, respectively. The objective of protein inpainting is to generate $\hat{\mathcal{M}}$ using a learnable function f_θ conditioned on $\mathcal{K}$:

$$f_\theta : \mathcal{K} \to \mathcal{M} \tag{2}$$

Using FoldTokenizer, we encode structure-sequence $\{(\boldsymbol{a}_1, s_1), (\boldsymbol{a}_2, s_2), \cdots, (\boldsymbol{a}_n, s_n)\}$ as discrete codes $\{z_1, z_2, \cdots, z_n\}$. Subsequently, we mask out and regenerate $\{z_i\}_{i \in \mathcal{I}_m}$ conditioned on $\{z_i\}_{i \in \mathcal{I}_k}$. A distinctive feature compared to previous work is our transformation of the multimodal sequence-structure into a discrete foreign language. This allows for a pure transformer model in structure-sequence co-design, eliminating the need for SE-(3) models and diffusion strategies. As shown in Fig.1, FoldGPT draws inspiration from GLM and utilizes full attention over all known contexts before generating masked regions in an autoregressive manner. To differentiate between the known and unknown segments, we utilize segment encoding in conjunction with position encoding. We formulate FoldGPT as

$$\max_\theta \mathbb{E} \sum_{k \in \mathcal{I}_m} \log p_\theta(z_k | \{z_i\}_{i \in \mathcal{I}_k}, \{z_j\}_{j<k, j \in \mathcal{I}_m}) \tag{3}$$

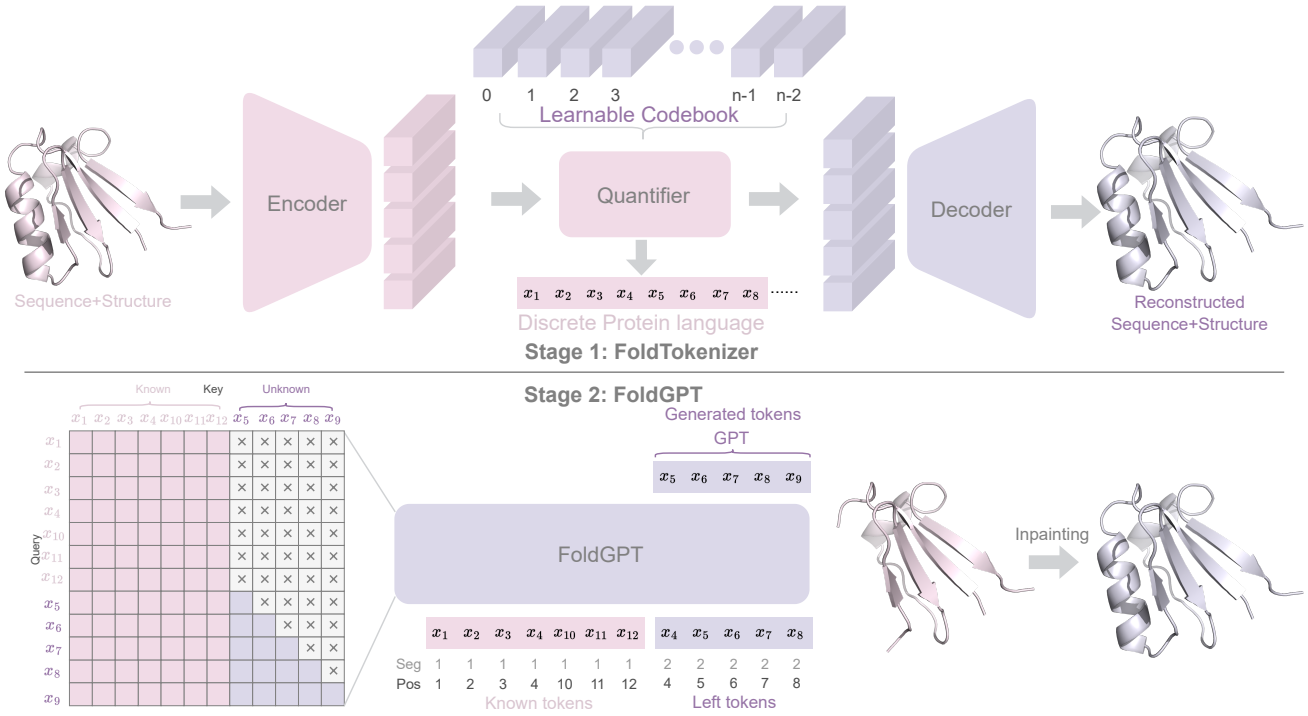


Figure 1: The overall framework of FoldToken. We use a encoder-quantizer-decoder to create fold language, and a transformer-based model to generate protein sequence-structure autoregressively.

FoldTokenizer: Encoder & Decoder

Protein Encoder The encoder transformer uses rotary position encoding and projects the protein sequence $\mathcal{S}$ and structure $\mathcal{X}$ as continuous latent embedding $\mathbf{h} \in \mathbb{R}^d$. We fuse the sequence-structure embeddings by simply addition:

$$\begin{cases} \mathbf{h}^s = \text{Embedding}(\mathcal{S}) \\ \mathbf{h}^x = \text{Linear}(\mathcal{X}) \\ \mathbf{h} = \text{Transformer}_{enc}(\mathbf{h}^s + \mathbf{h}^x) \end{cases} \tag{4}$$

Protein Decoder The decoder has the same architecture as encoder and reconstructs sequence-structure from the discrete latent codes $\mathbf{z}$ by separate predictive heads:

$$\begin{cases} \mathbf{h}^{dec} = \text{Transformer}_{dec}(Q^{-1}Q(\mathbf{h})) \\ \hat{\mathcal{S}} = \text{Linear}_s(\mathbf{h}^{dec}) \in \mathbb{R}^{n,20} \\ \hat{\mathcal{X}} = \text{Linear}_x(\mathbf{h}^{dec}) \in \mathbb{R}^{n,1} \end{cases} \tag{5}$$

Reconstruction Loss The learning objective is to recover protein sequence-structure like VQ-VAE. We use cross-entropy loss for residue type classification:

$$\mathcal{L}_s = \log p(\mathcal{S}|\mathbf{z}) = -\sum_{i=1}^{n} s_i \log(\hat{s}_i) \tag{6}$$

To handle the periodicity of the angles (α_i, β_i), we minimize their trigonometric values, i.e., $l_{\alpha_i} = ||\sin\alpha_i - \sin\hat{\alpha}_i||_2^2 + ||\cos\alpha_i - \cos\hat{\alpha}_i||_2^2$ for α_i, the structure loss is

$$\mathcal{L}_a = \log p(\mathcal{X}|\mathbf{z}) = \sum_{i=1}^{n} \left(||r_i - \hat{r}_i||_2^2 + l_{\alpha_i} + l_{\beta_i} \right) \tag{7}$$

The overall reconstruction is a weighted version:

$$\mathcal{L}_{rec} = \lambda_s \mathcal{L}_s + \lambda_a \mathcal{L}_a \tag{8}$$

Challenges on Vector Quantization

In this section, we thoroughly examine existing vanilla quantizers and lookup-free quantizers, as shown in Fig.2. By understanding these limitations and strengths, we aim to pave the way for further improvements in the field.

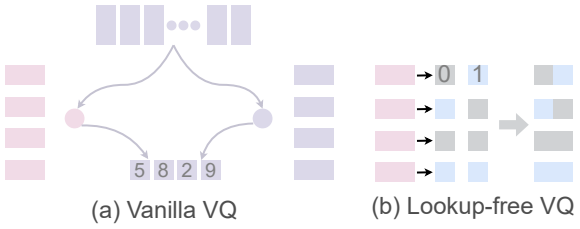


Figure 2: Baseline vector quantization methods.

Vector Quantization Problem The quantizer $Q : \boldsymbol{h} \mapsto z$ converts continuous embedding $\boldsymbol{h}$ as a discrete latent code z, named VQ-ID. The de-quantizer $Q^{-1} : z \mapsto \hat{\boldsymbol{h}}$ recover continual embedding $\hat{\boldsymbol{h}}$ from z. The problem formulation is

$$\begin{cases} z &= Q(\boldsymbol{h}) \\ \hat{\boldsymbol{h}} &= Q^{-1}(z) \end{cases} \tag{9}$$

Strategy A: Vanilla Quantizer (VVQ) Given input $\boldsymbol{h}$, the vanilla quantizer finds the nearest codebook vector $\boldsymbol{v}$ from codebook $\mathcal{E} = \{\boldsymbol{v}_k\}_{k=1}^m$. The index of $\boldsymbol{v}$ serves as the discrete latent code z, dubbled VQ-ID of $\boldsymbol{h}$.

$$\begin{cases} z_i &= Q(\boldsymbol{h}_i) = \arg\min_j ||\boldsymbol{h}_i - \boldsymbol{v}_j||_2 \\ \hat{\boldsymbol{h}}_i &= Q^{-1}(z_i) = \boldsymbol{v}_{z_i} \end{cases} \tag{10}$$

To improve the learning stability of VVQ, we utilize the exponential moving average (EMA) mechanism for codebook updating and normalize the continuous embeddings $\boldsymbol{h}$ and $\boldsymbol{v}$ to the unit hypersphere before quantization: $\boldsymbol{h} \leftarrow \boldsymbol{h}/||\boldsymbol{h}||_2$, $\boldsymbol{v} \leftarrow \boldsymbol{v}/||\boldsymbol{v}||_2$. In addition to the gradient update, the EMA moves $\boldsymbol{v}$ to the center of its nearest top-k embeddings $\{\boldsymbol{h}_j : 1 \le j \le k\}$, when $\boldsymbol{v}_i$ deviates from the data distribution. This process is described by $\boldsymbol{v}_i \leftarrow \tau \boldsymbol{v}_i + (1-\tau)(\boldsymbol{h}_i - \boldsymbol{v}_i)$. $\tau = 0.95 \quad \text{if} \quad ||\frac{1}{k}\sum_{j=1}^k \boldsymbol{h}_j - \boldsymbol{v}_i||_2 > \epsilon$, otherwise $\tau = 0.0$. Note that τ, k, and ϵ are the moving average weight, the number of nearest embeddings, and the threshold for out-of-distribution embeddings, respectively. We set $k = 5$ and $\epsilon = 0.1$. We apply the VQ loss for aligning $\boldsymbol{h}$ and $\boldsymbol{v}$, denoted as $\mathcal{L}_{vq} = ||\text{sg}(\boldsymbol{h}) - \boldsymbol{v}||_2^2 + 0.25||\boldsymbol{h} - \text{sg}(\boldsymbol{v})||_2^2$.

Limitations of VVQ The vanilla quantizer directly copies the gradient of $\hat{\boldsymbol{h}}_i$ to $\boldsymbol{h}_i$, i.e., $\frac{\partial L}{\partial \boldsymbol{h}_i} \leftarrow \frac{\partial L}{\partial \hat{\boldsymbol{h}}_i}$, resulting in the **gradient mismatching** between embedding $\frac{\partial L}{\partial \boldsymbol{h}_i}$ and $\boldsymbol{h}_i$. In addition, the higher reconstruction quality does not necessarily lead to better generation performance on downstream tasks. For example, (Yu et al. 2023) points out that reconstruction and generation may be contradicted: *Enlarging the vocabulary can improve reconstruction quality. However, such improvement only extends to generation when the vocabulary size is small, and a very large vocabulary can actually hurt the performance of the generative model.* Why does the contradiction occur? We attribute the intrinsic reason to **semantic irrelevance** between continuous and discrete representations and the **large class space**. We summarize the limitations as follows:

1. **Gradient Mismatching** As the VQ-IDs $\{z_i\}_{i=1}^m$ are non-differentiable, the vanilla quantizer directly copies the gradient of $\hat{\boldsymbol{h}}_i$ to $\boldsymbol{h}_i$, i.e., $\frac{\partial L}{\partial \boldsymbol{h}_i} = \frac{\partial L}{\partial \hat{\boldsymbol{h}}_i}$. However, $\hat{\boldsymbol{h}}_i$ generally does not equal $\boldsymbol{h}_i$, resulting in the mismatching between embedding $\boldsymbol{h}_i$ and its gradient $\frac{\partial L}{\partial \boldsymbol{h}_i}$.
2. **Semantic Irrelevance** The discrete and continuous representations exhibit no correlation, implying the absence of any association between $(\boldsymbol{h}_i - \boldsymbol{h}_j)$ and $(z_i - z_j)$. The irrelevance means that the generative model should accurately predict the exact z_i for recovering $\boldsymbol{h}_i$, despite the high similarity between z_i and z_j. This further increase the differenty of generation.
3. **Large Class Space** Regarding the generative model, each VQ-ID z_i represents a class index. The large codebook size of m can indeed pose challenges in generating the "protein language".

Strategy B: Lookup-Free Vector Quantizer (LFQ) This method (Yu et al. 2023) address the semantic mismatching and large class space problems. Denote $\boldsymbol{e}_j \in \mathbb{R}^d$ as the one-hot vector whose j-th element is 1 and $d = \log_2 m$. The key idea is to project the continuous $\boldsymbol{h}_i$ into the space of $\mathcal{V} = \{\sum_{j=1}^d [w_{ij}\boldsymbol{e}_j + (1-w_{ij})(-\boldsymbol{e}_j)] : w_{ij} \in \{0,1\}\}$:

$$\begin{cases} \hat{\boldsymbol{h}}_i = \text{sign}(\boldsymbol{h}_i) = \sum_{j=1}^d [u(h_{ij})\boldsymbol{e}_j + (1-u(h_{ij}))(-\boldsymbol{e}_j)] \\ \text{Bit}^{-1}(\boldsymbol{z}, d) := \sum_{j=1}^d 2^{j-1} z_j \\ \text{Bit}(z, d) := [\lfloor z/2^{d-1} \rfloor, \lfloor z/2^{d-2} \rfloor, \cdots, \lfloor z/2^0 \rfloor]^T \\ z_i = \text{Bit}^{-1}(u(\hat{h}_{i,j})) \end{cases} \tag{11}$$

The LFQ transforms latent embedding $\boldsymbol{h}_i$ into sign vector $\hat{\boldsymbol{h}}_i \in \{-1, 1\}^d$. $\text{sign}(x) = 1 \quad \text{if} \quad x \ge 0$ and $\text{sign}(x) = -1 \quad \text{if} \quad x < 0$. $u(x) = (\text{sign}(x)+1)/2$. We refer to $u(\hat{\boldsymbol{h}}_i)$ as the binary VQ-ID vector, and its decimal form is used as the VQ-ID z_i. The functions $\text{Bit}^{-1}(\cdot,\cdot)$ and $\text{Bit}(\cdot,\cdot)$ define the transformation between the binary VQ-ID vector and the VQ-ID.

Strength of LFQ The binary VQ-ID $u(\hat{\boldsymbol{h}}_i)$ directly captures the sign information of $\boldsymbol{h}_i$, establishing a semantic relevance between the discrete and continuous representations. This semantic relevance offers several advantages for downstream tasks, including:

1. **Robust Generation**: Accurate prediction of all bits is unnecessary for the model. Majority of correct "key" bits ensures similar overall semantics, leading to reasonable reconstruction of proteins.
2. **Simplified Classification**: Unlike VVQ that predicts tokens from a space of size m, the LFQ allows us to decompose the problem as $\log_2 m$ binary classifications.

LFQ addresses the issue of semantic irrelevance and make the generation more robust and easy to be optimized.

Limitations of LFQ LFQ fails to address the issue of gradient mismatching and and introduces a new challenge of information bottleneck.Typically, the codebook space is chosen from options such as 2^8 (int8), 2^{16} (int16), 2^{32}(int32), and 2^{64}(int64) due to accommodate the required bits for storing a VQ-ID. However, for a fair comparison to VVQ and effective data compression, only 2^8 and 2^{16} are considered, resulting in a hidden space size of 8 or 16 in lookup-free VQ. This low dimensionality poses the problem of information bottleneck, hindering accurate reconstruction.

How to improve VQ methods?

To address the aforementioned problems, we introduce three VQ methods, e.g., soft quantizer (SoftVQ), soft group quantizer (SoftGVQ) and soft conditional quantizer (SoftCVQ). In Table. 1, we summarize addressed issues of VQ methods, i.e., **g**radient **m**ismatching (GM), **s**emantic **i**rrelevance (SI), **m**ulticlass **c**lassification (MC), and **i**nformation **b**ottleneck (IB), and the strength of **s**oft **g**lobal **q**uerying (SGQ). Readers will understand SGQ from strategy E.

Strategy C: Soft Vector Quantizer (SoftVQ vs VVQ) Instead of selecting the closest codebook vector, the soft quantizer utilizes a differentiable attention operation to

VQ Method	w/o GM	w/o SI	w/o MC	w/o IB	w SGQ
Vanilla				✓	
Lookup-free		✓	✓		
SoftVQ (our)	✓			✓	✓
SoftGVQ (our)	✓	✓	✓	✓	
SoftCVQ (our)	✓	✓	✓	✓	✓

Table 1: Comparison of different vector quantization methods.

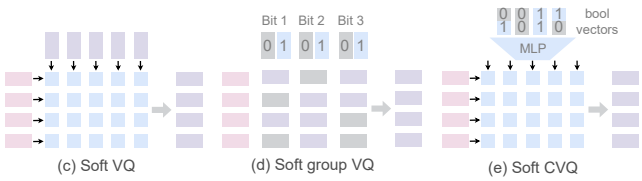


Figure 3: Proposed vector quantization methods.

query the entire codebook $\mathcal{E} = \{\boldsymbol{v}_k\}_{k=1}^m$:

$$\begin{cases} a_{ij} & = \frac{\exp(\boldsymbol{h}_i^T \boldsymbol{v}_j / T)}{\sum_{j=1}^m \exp(\boldsymbol{h}_i^T \boldsymbol{v}_j / T)} \\ \hat{\boldsymbol{h}}_i & = \sum_{j=1}^m a_{ij} \boldsymbol{v}_j \\ z_i & = \arg\max_j a_{ij} \end{cases} \tag{12}$$

The temperature parameter T plays a crucial role in controlling the smoothness of the attention map. When $T \to 0$, the attention map becomes a one-hot vector, which is equivalent to the VVQ. During the training process, as T gradually decreases from 1.0 to $1e^{-5}$, the model automatically learns a sharp attention map for querying the "nearest" codebook vector. The differentiable attention operation avoids gradient mismatch problem. During the inference phase, we employ random sampling to obtain the discrete latent codes z_i from the distribution $z_i \sim \text{Multinomial}([a_{i,0}, a_{i,1}, \cdots, a_{i,M}])$.

Strategy D: Soft Group Vector Quantizer (SoftGVQ vs LFQ) Similar to LFQ, the soft group quantizer projects the continuous embedding $\boldsymbol{h}_i$ into the space of $\mathcal{V} = \{\sum_{j=1}^{\log_2(m)} [w_{ij}\boldsymbol{v}_j + (1 - w_{ij})\dot{\boldsymbol{v}}_j] : w_{ij} \in [0, 1]\}$, where m is the codebook size, $\boldsymbol{v}_j$ and $\dot{\boldsymbol{v}}_j$ are learnable basis vectors in the j-th group. The w_{ij} is computed by soft attention, i.e., $w_{ij} = \frac{\exp(\boldsymbol{h}_i^T \boldsymbol{v}_j / T)}{\exp(\boldsymbol{h}_i^T \boldsymbol{v}_j / T) + \exp(\boldsymbol{h}_i^T \dot{\boldsymbol{v}}_j / T)}$. During training, we gradually reduce T from 1 to $1e-5$. Define $\boldsymbol{w} = [w_{i,1}, w_{i,2}, \cdots, w_{i,\log_2(m)}]$, the binary VQ-ID vector is $u(\boldsymbol{w} - 0.5)$. The recovered embedding $\hat{\boldsymbol{h}}_i$ and VQ-ID z_i could be computed by

$$\begin{cases} \hat{\boldsymbol{h}}_i = \sum_{j=1}^{\log_2(m)} [w_{ij}\boldsymbol{v}_j + (1 - w_{ij})\dot{\boldsymbol{v}}_j] \\ z_i = \text{Bit}^{-1}(u(\boldsymbol{w} - 0.5)) \end{cases} \tag{13}$$

The soft group quantizer offers several advantages:

1. The basis vectors ($\boldsymbol{v}_j$ and $\dot{\boldsymbol{v}}_j$) are learnable and data-dependent, and do not limited to binary one-hot basises ($\boldsymbol{e}_j$ and $-\boldsymbol{e}_j$) used by LFQ. This allow the model to capture the principle directions of the data distribution.
2. Within each group, a soft quantization strategy is employed to select the nearest codebook vector, addressing the issue of gradient mismatching.
3. The hidden space dimension equal to the that of basis vectors, which is not constrained to be $\log_2(m)$ as in LFQ, elleviating the issue of information bottleneck.

Strategy E: Soft Conditional Quantizer (SoftCVQ, Final Version) SoftVQ and SoftGVQ present improvements over VVQ and LFQ, respectively. However, we have observed a trade-off between high-quality reconstruction (SoftVQ) and generation (SoftGVQ). While the binary VQ-IDs of SoftGVQ are beneficial for generation tasks, they perform relatively poorer in terms of reconstruction compared to SoftVQ. To strike a balance between reconstruction and generation capabilities, we propose a novel approach called the Soft Conditional Vector Quantizer (SoftCVQ). The insights comes from following obervations:

1. The key to achieving high-quality reconstruction lies in employing soft attention to query the entire codebook space, i.e., soft global querying $\rightarrow$ inspired by SoftVQ.
2. The binary form of VQ-ID facilities generation $\rightarrow$ inspired by SoftGVQ and LFQ.

The proposed SoftCVQ allow attention query over the whole codebook while keeping binary VQ-IDs. A straightforward idea is to use a codebook comprising fixed binary vectors, i.e., $\mathcal{V} = \{\text{Bit}(z, \log_2(m)) \in \mathbb{R}^{\log_2(m)}\}_{z=1}^m$, and then applies soft attention between latent embedding $\boldsymbol{h}$ and all the codebook vectors $\mathcal{V}$. However, this approach suffers from information bottleneck due to the small latent dimension $\log_2(m)$. To address this challenge, we introduce a conditional network that projects $\log_2(m)$-dimensional binary vectors of $\mathcal{V}$ into d-dimensional vectors:

$$\boldsymbol{v}_j = \text{ConditionNet}(\text{Bit}(z, \log_2(m))) \tag{14}$$

the ConditionNet : $\mathbb{R}^{\log_2(m)} \to \mathbb{R}^d$ is a MLP. If the codebook size is 2^{16}, the MLP projects 65536 16-dimension boolvectors into 65536 d-dimension vectors. Then we apply soft vector quantization between latent embedding $\boldsymbol{h}_i$ and $\{\boldsymbol{v}_z\}_{z=1}^m$ to obtain the quantized embedding $\hat{\boldsymbol{h}}_i$

$$\hat{\boldsymbol{h}}_i = \text{SoftVQ}(\boldsymbol{h}_i, \{\boldsymbol{v}_z\}_{z=1}^m) \tag{15}$$

During evaluation, when the nearest codebook vector to $\hat{\boldsymbol{h}}_i$ is $\boldsymbol{v}_z$, the corresponding VQ-ID is exactly z, where its binary VQ-ID vector is $\text{Bit}(i, \log_2(m))$.

Experiments

We conduct experiments to answer the following questions:

- **Reconstruction (Q1):** Do the proposed methods outperform baselines on reconstruction quality?
- **Backbone Inpainting (Q2):** How does the learned protein language perform in the generation task?
- **Ablation (Q3):** What are the key factors contributing to the effectiveness of SoftCVQ?

Datasets

cAF2DB for VQ Pretraining We employ cAF2DB, a clustered version of the AlphaFold Uniprot v3 database, for VQ pretraining. The original AF2DB contains a large

Method	Global Metrics		Success Rates		Local Metrics					
	Rec ↑	TMScore ↑	Rec ≥ 0.95	TMScore ≥ 0.5	$L_r \downarrow$	$L_\alpha \downarrow$	$L_\beta \downarrow$	$\max L_r \downarrow$	$\max L_\alpha \downarrow$	$\max L_\beta \downarrow$
Vanilla	<u>0.9757</u>	0.4224	<u>0.9791</u>	0.1693	0.0966	0.0602	0.0783	2.1790	0.5605	0.3723
LFQ	0.9482	0.4385	0.7534	0.2894	0.1066	0.0701	0.1079	2.2068	0.5856	0.5257
SoftVQ (our)	0.9713	**0.7616**	0.9781	**0.9530**	<u>0.0683</u>	**0.0438**	**0 0232**	<u>1.5647</u>	**0 4989**	**0.1439**
SoftGVQ (our)	0.9743	0.5703	**0.9875**	0.5120	0.0936	0.0616	0.0619	2.0090	0.5655	0.3054
SoftCVQ (our)	**0.9880**	<u>0.7470</u>	0.9603	<u>0.9498</u>	**0.0523**	<u>0.0447</u>	<u>0.0284</u>	**0.8274**	<u>0.5334</u>	<u>0.1646</u>

Table 2: Performance of protein VQ models on CATH4.3 test set. We highlight the **best** results and <u>suboptimal</u>.

number of structures (214,684,311), which presents computational challenges. To overcome this, we utilize cAF2DB (Barrio-Hernandez et al. 2023) consisting of 2.27M structural clusters. The representative protein structure with the highest pLDDT score is selected from each cluster, while structures with lengths below 30 or pLDDT scores lower than 70 are excluded. This selection process yields a final set of 1,323,729 proteins for VQ pretraining.

CATH4.3 for Backbone Inpainting For the task of backbone inpainting, we employ CATH4.3. To construct the train, validation, and test sets, we utilize the CAT code to randomly partition proteins in a ratio of 95:2:3, ensuring no overlap in the training, validation, and test sets on the CAT code. This results in a train set with 30,290 samples, a validation set with 638 samples, and a test set with 957 samples. The test set comprising 957 samples is used for evaluating reconstruction performance. As to the evaluation of backbone inpainting, we apply a filtering criterion based on proteins that can be folded by ESMFold. We choose proteins with an average pLDDT score of 0.9 or higher to obtain the inpainting test set comprising 117 proteins.

Reconstruction (Q1)

Do the proposed methods outperform baselines on reconstruction quality?

Setting We conduct fair comparison for different VQ models by using the same codebook space of 2^{16}, code vector dimension of 32 and the same encoder-decoder architectures. Both the encoder and decoder adopt ESM-35M bert transformer. The FoldTokenizer is pretrained on cAF2DB for transforming protein sequence-structure into sequence of VQ-IDs. The evaluation is conducted on real-word proteins, i.e., the whole test set of CATH4.3 with size of 957 samples. With BF16 precision training in DeepSpeed, the model is trained for 15 epochs using the OneCycle scheduler and AdamW optimizer. The batch size is set to 128, the learning rate is 0.0001, and the padding length is 512.

Baselines Currently, no method exists for protein sequence-structure quantization and reconstruction. We conduct a comparative analysis of five VQ methods, namely VVQ (Van Den Oord, Vinyals et al. 2017), LFQ (Mentzer et al. 2023; Yu et al. 2023), SoftVQ, SoftGVQ, and SoftCVQ, all implemented under the same encoder-decoder architecture. It's noteworthy that VVQ is the most widely used method, whereas LFQ stands as the recent SOTA.

Metrics In evaluating sequence reconstruction, we quantify the recovery rate. The assessment of recovered structure quality involves both global structure similarity (TMScore) and local residue reconstruction (recovering loss). For a set of b proteins with lengths n_1, n_2, n_b, we present the average recovering loss per residue (L_r, L_α, L_β) and maximum recovering loss per residue ($\max L_r, \max L_\alpha, \max L_\beta$), with detailed definitions available in the appendix. To aid bioscientists in algorithm selection, we introduce the reconstruction success rate. A reconstruction is deemed successful if the reconstructed data achieves a minimum of 95% recovery and 50% TMScore compared to the reference.

We present the reconstruction results in Table.2 and have following findings:

Reconstructing structure poses a non-trivial challenge. While all VQ methods demonstrate good performance in sequence recovery, the vanilla VQ and LFQ models encounter difficulties in accurately reconstructing protein structure. Specifically, these models exhibit an average TMScore below 0.5 and a structure success rate of less than 30%.

The key to high-quality reconstruction is soft querying across the entire codebook space. SoftVQ, SoftCVQ, and SoftGVQ achieve TMScores above 0.5 and structure success rates exceeding 50%. It's worth noting that SoftVQ and SoftCVQ exhibit superior reconstruction performance than others, with success rates around 95%. Given this, we consider them to be reliable protein quantization tools.

FoldTokenizer works well for data compressing. When considering the storage of the C_α atom, the entire CATH4.3 dataset (2.7GB pdb files) could be compressed to discrete protein language sentences (18MB) using SoftCVQ.

In Fig.5 in Appendix , we present visual examples of reconstructed proteins, accompanied by TMScore and recovery rate notations. The proposed methods, including SoftGVQ, SoftVQ, and SoftCVQ, demonstrate notable improvements in protein structure reconstruction.

Backbone Inpainting (Q2)

How about applying protein language on generative task?

Setting The inpainting task aims to predicte the masked fragment of a protein sequence-structure. We transform the whole CATH4.3 dataset as sequence of VQ-IDs using FoldTokenizer. We partition the dataset into training and testing sets, ensuring no overlap in CAT codes across proteins. During training, the length of the masked fragment in a protein of length L is uniformly sampled from $U(5, L/3)$. The

Sequence Design			Structure Design (angle-based)			Structure Design (coordinate-based)		
	$\texttt{Rec}(\mathcal{S},\hat{\mathcal{S}})\uparrow$	$\texttt{TM}(\mathcal{X},\hat{\mathcal{X}}')\uparrow$		$\texttt{TM}(\mathcal{X},\hat{\mathcal{X}})\uparrow$	$\texttt{Rec}(\mathcal{S},\hat{\mathcal{S}}')\uparrow$		$\texttt{TM}(\mathcal{X},\hat{\mathcal{X}})\uparrow$	$\texttt{Rec}(\mathcal{S},\hat{\mathcal{S}}')\uparrow$
ESM2	95.3	89.2	FoldingDiff	0.67	23.5	ProtDiff	0.95	28.4
EvoDiff	93.0	88.7	DiffSDS	0.74	25.6	RFDiffusion	0.99	33.9
FoldGPT	96.2	90.4	FoldGPT	0.80	28.9	Chroma	0.98	34.5

Table 3: Results of backbone inpainting. The proposed FoldGPT outperforms baselines in both sequence design. Regarding structure design, FoldGPT demonstrates superior performance compared to angle-based methods.

model contains 15 transformer layers with 480 hidden dimension and 20 attention heads. We train the model up to 20k steps using the OneCycle scheduler and AdamW optimizer. The batch size is 128, the learning rate is 0.0005, and the padding length is 512.

Baselines The sequence inpainting baselines are widely recognized ESM2 (Lin et al. 2022) and EvoDiff (Alamdari, Thakkar, and van den Berg 2023). Structure inpainting methods are classified into angle-based and coordinate-based. Angle-based baselines include FoldingDiff (Wu et al. 2024) and DiffSDS (Gao, Tan, and Li 2023a), while coordinate-based methods are ProtDiff (Trippe et al. 2022) , Chroma (Ingraham et al. 2023) and RFDiffusion (Watson, Juergens et al. 2023). FoldGPT belongs to the group of angle-based methods; in the future, we plan to extend it into the coordinate-based domain. Due to limited energy and computing resources, we utilize the pretrained checkpoints of baseline models. Consequently, the baselines may be overstated, given that the testing data may be part of their training set, and they (Lin et al. 2022; Alamdari, Thakkar, and van den Berg 2023; Ingraham et al. 2023; Watson, Juergens et al. 2023) have utilized more training data.

Metrics Denote $\mathcal{S},\mathcal{X}$ and $\hat{\mathcal{S}},\hat{\mathcal{X}}$ as the reference and predicted sequence-structure. We define $\hat{\mathcal{S}}\rightarrow\hat{\mathcal{X}}'$ as a folding process using ESMFold and $\hat{\mathcal{X}}\rightarrow\hat{\mathcal{S}}'$ inverse folding prosess using modified PiFold (considering only C_α atoms). Define $\texttt{Rec}(\mathcal{S}_1,\mathcal{S}_2)$ and $\texttt{TM}(\mathcal{X}_1,\mathcal{X}_2)$ as sequence recovery and structure TMScore. We report $\texttt{Rec}(\mathcal{S},\hat{\mathcal{S}})$ and $\texttt{TM}(\mathcal{X},\hat{\mathcal{X}}')$ for sequence inpainting. Similarly, we consider $\texttt{Rec}(\mathcal{S},\hat{\mathcal{S}}')$ and $\texttt{TM}(\mathcal{X},\hat{\mathcal{X}})$ for structure inpainting.

Refer to results on Table.3, we provide following insights:

FoldGPT excels in sequence inpainting compared to baselines. Despite using less training data than ESM2 and EvoDiff, FoldGPT achieves superior recovery and self-consistent TMScore. This highlights the effectiveness of the GPT-style generative model with learned protein language, outperforming MLM-style and diffusion-style approaches.

FoldGPT outperforms other angle-based methods. In structure inpainting, FoldGPT demonstrates significant improvement compared to FoldingDiff and DiffSDS. These results underscore the effectiveness of utilizing protein language for generative tasks with the same data representation.

Coordinate-based representation is superior to angle-based. Angle-based structure representation poses the risk of error accumulation, where atoms are added sequentially to the backbone using folding angles, making it challenging to consider pair-wise residue interactions. In light of this, we plan to extend FoldToken as a coordinate-based method.

Binary VQ-ID is crucial for convergence. Using the language learned by SoftVQ with a class space of 2^{16} resulted in training crashes, regardless of learning rate adjustments ($\text{lr}\in\{1e-5,5e-5,5e-4\}$). However, leveraging the binary VQ-ID learned by SoftCVQ, which decomposes the large class space into 16 binary subspaces and transforms the problem into 16 binary classifications, led to a successful and smooth convergence of the model.

While innovative, there is room for improvement in FoldGPT. To our knowledge, FoldGPT stands as the first GPT-style model for sequence-structure co-generation. However, the current angle-based structure representation hinders the consideration of protein complexes and the intricate 3D residue interactions, thereby limiting its potential to boarder applications. We plan to address these issues in the next version of FoldGPT.

Ablation (Q3)

We investigate the impact of the number of MLP layers (#MLP), the dimension of the codebook vector (#CodeDim), and spherical normalization (sphere) of latent embeddings regarding the reconstruction performance of SoftCVQ. Table 4 demonstrates that appropriately increasing #MLP and #CodeDim enhances performance. Additionally, we highlight the significance of spherical normalization in achieving optimal results.

Method-ID	Config			Global Metrics		Success Rates	
	#MLP	#CodeDim	Sphere	Rec ↑	TMScore ↑	Rec ≥ 0.95	TMScore ≥ 0.5
SCQ-1	2	16		0.9801	0.5853	0.8997	0.7638
SCQ-2	2	16	✓	0.9879	0.6451	0.9572	0.8892
SCQ-3	2	32		0.9965	0.4285	0.9937	0.2184
SCQ-4	2	32	✓	0.9896	0.6959	0.9561	0.9216
SCQ-5	3	32		0.9875	0.5867	0.9457	0.7367
SCQ-6	3	32	✓	0.9862	0.6697	0.9467	0.9028
SCQ-7	6	32	✓	0.9880	0.7470	0.9603	0.9498
SCQ-8	9	32	✓	0.9875	0.7015	0.9551	0.9101

Table 4: Ablation of SoftCVQ on CATH4.3 test set.

Conclusion

This paper presents a framework for learning a discrete protein language that describes sequence and structure simultaneously. We apply this learned language to generative tasks, specifically general backbone inpainting design, constructing the first GPT-style model for sequence-structure co-generation. Our results demonstrate promising outcomes compared to baselines under the same settings. We believe that our work could pave the way for new paradigms in protein representation and generation.

Acknowledgments

This work was supported by National Science and Technology Major Project (No. 2022ZD0115101), National Natural Science Foundation of China Project (No. 624B2115, No. U21A20427), Project (No. WU2022A009) from the Center of Synthetic Biology and Integrated Bioengineering of Westlake University, Project (No. WU2023C019) from the Westlake University Industries of the Future Research Funding, and Tencent Rhino-Bird Focused Research Program (No. RBFR2023007).

References

Alamdari, S.; Thakkar, N.; and van den Berg, R. a. 2023. Protein generation with evolutionary diffusion: sequence is all you need. *bioRxiv*, 2023–09.

Baevski, A.; Schneider, S.; and Auli, M. 2019. vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations. In *International Conference on Learning Representations*.

Barrio-Hernandez, I.; Yeo, J.; Jänes, J.; Wein, T.; Varadi, M.; Velankar, S.; Beltrao, P.; and Steinegger, M. 2023. Clustering predicted structures at the scale of the known protein universe. *bioRxiv*, 2023–03.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *NeurIPS*, 33: 1877–1901.

Chen, T.; Li, L.; and Sun, Y. 2020. Differentiable product quantization for end-to-end embedding compression. In *ICML*, 1617–1626. PMLR.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eguchi, R. R.; Choe, C. A.; and Huang, P.-S. 2022. Ig-VAE: Generative modeling of protein structure by direct 3D coordinate generation. *PLoS computational biology*, 18(6): e1010271.

El-Nouby, A.; Muckley, M. J.; Ullrich, K.; Laptev, I.; Verbeek, J.; and Jégou, H. 2022. Image compression with product quantized masked image modeling. *arXiv preprint arXiv:2212.07372*.

Esser, P.; Rombach, R.; and Ommer, B. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12873–12883.

Fan, H.; Wang, Z.; Yang, Y.; and Kankanhalli, M. 2022. Continuous-Discrete Convolution for Geometry-Sequence Modeling in Proteins. In *The Eleventh International Conference on Learning Representations*.

Gao, Z.; Tan, C.; Li, S.; et al. 2022. AlphaDesign: A graph protein design method and benchmark on AlphaFoldDB. *arXiv preprint arXiv:2202.01079*.

Gao, Z.; Tan, C.; and Li, S. Z. 2023a. DiffSDS: A language diffusion model for protein backbone inpainting under geometric conditions and constraints. *arXiv preprint arXiv:2301.09642*.

Gao, Z.; Tan, C.; and Li, S. Z. 2023b. Knowledge-Design: Pushing the Limit of Protein Deign via Knowledge Refinement. *arXiv preprint arXiv:2305.15151*.

Gao, Z.; Tan, C.; and Li, S. Z. 2023c. PiFold: Toward effective and efficient protein inverse folding. In *International Conference on Learning Representations*.

Horiguchi, S.; Dohi, K.; and Kawaguchi, Y. 2023. Streaming Active Learning for Regression Problems Using Regression via Classification. *arXiv preprint arXiv:2309.01013*.

Hsu, C.; Verkuil, R.; Liu, J.; Lin, Z.; Hie, B.; Sercu, T.; Lerer, A.; and Rives, A. 2022. Learning inverse folding from millions of predicted structures. *bioRxiv*.

Hu, B.; Tan, C.; Gao, B.; Gao, Z.; Wu, L.; Xia, J.; and Li, S. Z. 2024. Multimodal Distillation of Protein Sequence, Structure, and Function.

Hu, B.; Tan, C.; Xia, J.; Zheng, J.; Huang, Y.; Wu, L.; Liu, Y.; Xu, Y.; and Li, S. Z. 2023. Learning complete protein representation by deep coupling of sequence and structure. *bioRxiv*, 2023–07.

Ingraham, J.; Garg, V.; Barzilay, R.; and Jaakkola, T. 2019. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32.

Ingraham, J. B.; Baranov, M.; Costello, Z.; Barber, K. W.; Wang, W.; Ismail, A.; Frappier, V.; Lord, D. M.; et al. 2023. Illuminating protein space with a programmable generative model. *Nature*, 1–9.

Jin, W.; Wohlwend, J.; Barzilay, R.; and Jaakkola, T. 2021. Iterative refinement graph neural network for antibody sequence-structure co-design. *arXiv preprint arXiv:2110.04624*.

Jing, B.; Eismann, S.; Suriana, P.; Townshend, R. J.; and Dror, R. 2020. Learning from protein structure with geometric vector perceptrons. *arXiv:2009.01411*.

Juang, B.-H.; and Gray, A. 1982. Multiple stage vector quantization for speech coding. In *ICASSP*, 597–600. IEEE.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1): 117–128.

Kong, X.; Huang, W.; and Liu, Y. 2022. Conditional antibody design as 3d equivariant graph translation. *arXiv preprint arXiv:2208.06073*.

Lee, D.; Kim, C.; Kim, S.; Cho, M.; and Han, W.-S. 2022. Autoregressive image generation using residual quantization. In *CVPR*, 11523–11532.

Lee, J. S.; Kim, J.; and Kim, P. M. 2022. ProteinSGM: Score-based generative modeling for de novo protein design. *bioRxiv*, 2022–07.

Lin, Z.; Akin, H.; Rao, R.; et al. 2022. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*.

Luo, S.; Su, Y.; Peng, X.; Wang, S.; Peng, J.; and Ma, J. 2022. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. *NeurIPS*, 35: 9754–9767.

Martinez, J.; Hoos, H. H.; and Little, J. J. 2014. Stacked quantizers for compositional vector compression. *arXiv preprint arXiv:1411.2173*.

Mentzer, F.; Minnen, D.; Agustsson, E.; and Tschannen, M. 2023. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*.

Pintea, S. L.; Lin, Y.; Dijkstra, J.; and van Gemert, J. C. 2023. A step towards understanding why classification helps regression. In *ICCV*, 19972–19981.

Samaga, Y. B.; Raghunathan, S.; and Priyakumar, U. D. 2021. SCONES: self-consistent neural network for protein stability prediction upon mutation. *The Journal of Physical Chemistry B*, 125(38): 10657–10671.

Shi, C.; Wang, C.; Lu, J.; Zhong, B.; and Tang, J. 2022. Protein sequence and structure co-design with equivariant translation. *arXiv preprint arXiv:2210.08761*.

Song, Z.; Zhao, Y.; Shi, W.; Yang, Y.; and Li, L. 2023. Functional Geometry Guided Protein Sequence and Backbone Structure Co-Design. *arXiv preprint arXiv:2310.04343*.

Stewart, L.; Bach, F.; and Vert, J.-P. 2023. Regression as Classification: Influence of Task Formulation on Neural Network Features. In *ICAIS*, 11563–11582. PMLR.

Tan, C.; Gao, Z.; and Li, S. Z. 2023. Protein Complex Invariant Embedding with Cross-Gate MLP is A One-Shot Antibody Designer. *arXiv preprint arXiv:2305.09480*.

Tan, C.; Gao, Z.; Xia, J.; and Li, S. Z. 2022. Generative De Novo Protein Design with Global Context. *arXiv preprint arXiv:2204.10673*.

Trippe, B. L.; Yim, J.; Tischer, D.; Broderick, T.; Baker, D.; Barzilay, R.; and Jaakkola, T. 2022. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*.

Umerenkov, D.; Shashkova, T. I.; Strashnov, P. V.; Nikolaev, F.; Sindeeva, M.; Ivanisenko, N. V.; and Kardymon, O. L. 2022. PROSTATA: Protein Stability Assessment using Transformers. *bioRxiv*, 2022–12.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Wang, T.-C.; Liu, M.-Y.; Zhu, J.-Y.; Tao, A.; Kautz, J.; and Catanzaro, B. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8798–8807.

Watson, J. L.; Juergens, D.; et al. 2023. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976): 1089–1100.

Wu, K. E.; Yang, K. K.; van den Berg, R.; Alamdari, S.; Zou, J. Y.; Lu, A. X.; and Amini, A. P. 2024. Protein structure generation via folding diffusion. *Nature communications*, 15(1): 1059.

Yu, J.; Li, X.; Koh, J. Y.; Zhang, H.; Pang, R.; Qin, J.; Ku, A.; Xu, Y.; Baldridge, J.; and Wu, Y. 2021. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*.

Yu, L.; Lezama, J.; Gundavarapu, N. B.; Versari, L.; Sohn, K.; Minnen, D.; Cheng, Y.; Gupta, A.; Gu, X.; Hauptmann, A. G.; et al. 2023. Language Model Beats Diffusion-Tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*.

Zeghidour, N.; Luebs, A.; Omran, A.; Skoglund, J.; and Tagliasacchi, M. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30: 495–507.

Zhang, J.; Zhan, F.; Theobalt, C.; and Lu, S. 2023a. Regularized vector quantization for tokenized image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18467–18476.

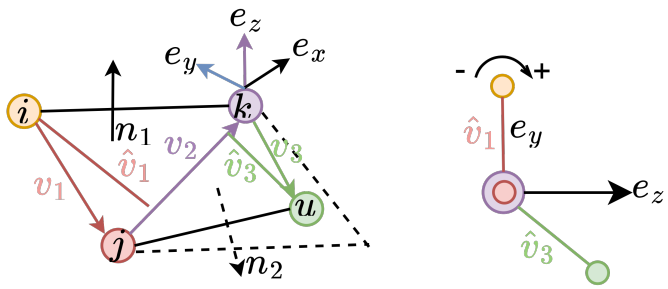
Zhang, Y.; Gao, Z.; Tan, C.; and Li, S. Z. 2023b. Efficiently Predicting Protein Stability Changes Upon Single-point Mutation with Large Language Models. *arXiv preprint arXiv:2312.04019*.

Zhang, Z.; Xu, M.; Jamasb, A.; Chenthamarakshan, V.; Lozano, A.; Das, P.; and Tang, J. 2022. Protein representation learning by geometric structure pretraining. *arXiv preprint arXiv:2203.06125*.

Zheng, J.; Li, S.; Huang, Y.; Gao, Z.; Tan, C.; Hu, B.; Xia, J.; Wang, G.; and Li, S. Z. 2023. MMDesign: Multi-Modality Transfer Learning for Generative Protein Design. *arXiv preprint arXiv:2312.06297*.

Folding Angle Definition

As shown in Fig. 4, following the backbone direction, given the previous three atoms, we can construct a local frame $T_u = \{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\}$. The next atom $\boldsymbol{x}_u$ can be represented by the distance r_u and two torsion angles α_u and β_u, denoted as $\boldsymbol{a}_u = (r_u, \alpha_u, \beta_u)$.



(a) Place u based on the plane of i-j-k (b) Torsion angle

Figure 4: Torsion angle computation.

Structure as angle sequence The vanilla protein structures are represented as 3d points, where the rotation and translation equivariance should be considered in modeling. To directly apply the transformer-based encoder and decoder, we convert the 3d points into a sequence of angles, which is rotation and translation invariant. As shown in Fig. 4, following the backbone direction, given previous three atoms, we can construct a local frame $T_u = \{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\}$. The next atom $\boldsymbol{x}_u$ can be represented by the distance r_u and two torsion angles α_u and β_u, denoted as $\boldsymbol{a}_u = (r_u, \alpha_u, \beta_u)$.

$$
\begin{cases}
\boldsymbol{v}_1 = \boldsymbol{x}_j - \boldsymbol{x}_i \\
\boldsymbol{v}_2 = \boldsymbol{x}_k - \boldsymbol{x}_j \\
\boldsymbol{v}_3 = \boldsymbol{x}_u - \boldsymbol{x}_k \\
\hat{\boldsymbol{v}}_1 = -\boldsymbol{v}_1 - ((-\boldsymbol{v}_1) \cdot \boldsymbol{v}_2)\boldsymbol{v}_2 \\
\hat{\boldsymbol{v}}_3 = \boldsymbol{v}_3 - (\boldsymbol{v}_3 \cdot \boldsymbol{v}_2)\boldsymbol{v}_2 \\
r_u = ||\boldsymbol{v}_3||_2 \\
\alpha_u = \angle(-\hat{\boldsymbol{v}}_2, \hat{\boldsymbol{v}}_3) \\
\beta_u = \angle(\hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_3) = \angle(\boldsymbol{n}_1, \boldsymbol{n}_2)
\end{cases}
\tag{16}
$$

To completely represent the structure, we add virtual frames T_0 and T_{n+1} at the start and end of the protein structure, where the virtual atoms have fixed $(r, \alpha, \beta) = (1, 1, 1)$ to its nearest backbone frames. Finally, the protein structure $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3 : 1 \leq i \leq n\}$ will be represented as $\mathcal{S} = \{(r_i, \alpha_i, \beta_i) : 0 \leq i \leq n+1\}$.

Local Metrics for Backbone Inpainting

We define the average recovering loss per residue:

$$
\begin{cases}
L_r = \frac{1}{b}\sum_{i=1}^{b}\sum_{j=1}^{n_i}\frac{|r_{i,j} - \hat{r}_{i,j}|}{n_i} \\
L_\alpha = \frac{1}{b}\sum_{i=1}^{b}\sum_{j=1}^{n_i}\frac{|\alpha_{i,j} - \hat{\alpha}_{i,j}|}{n_i} \\
L_\beta = \frac{1}{b}\sum_{i=1}^{b}\sum_{j=1}^{n_i}\frac{|\beta_{i,j} - \hat{\beta}_{i,j}|}{n_i}
\end{cases}
\tag{17}
$$

where b is batch size and n_i is the length of protein i. The maximum recovering loss per residue is:

$$
\begin{cases}
\max L_r = \frac{1}{b}\sum_{i=1}^{b}\max_{j=1}^{n_i}|r_{i,j} - \hat{r}_{i,j}| \\
\max L_\alpha = \frac{1}{b}\sum_{i=1}^{b}\max_{j=1}^{n_i}|\alpha_{i,j} - \hat{\alpha}_{i,j}| \\
\max L_\beta = \frac{1}{b}\sum_{i=1}^{b}\max_{j=1}^{n_i}|\beta_{i,j} - \hat{\beta}_{i,j}|
\end{cases}
\tag{18}
$$

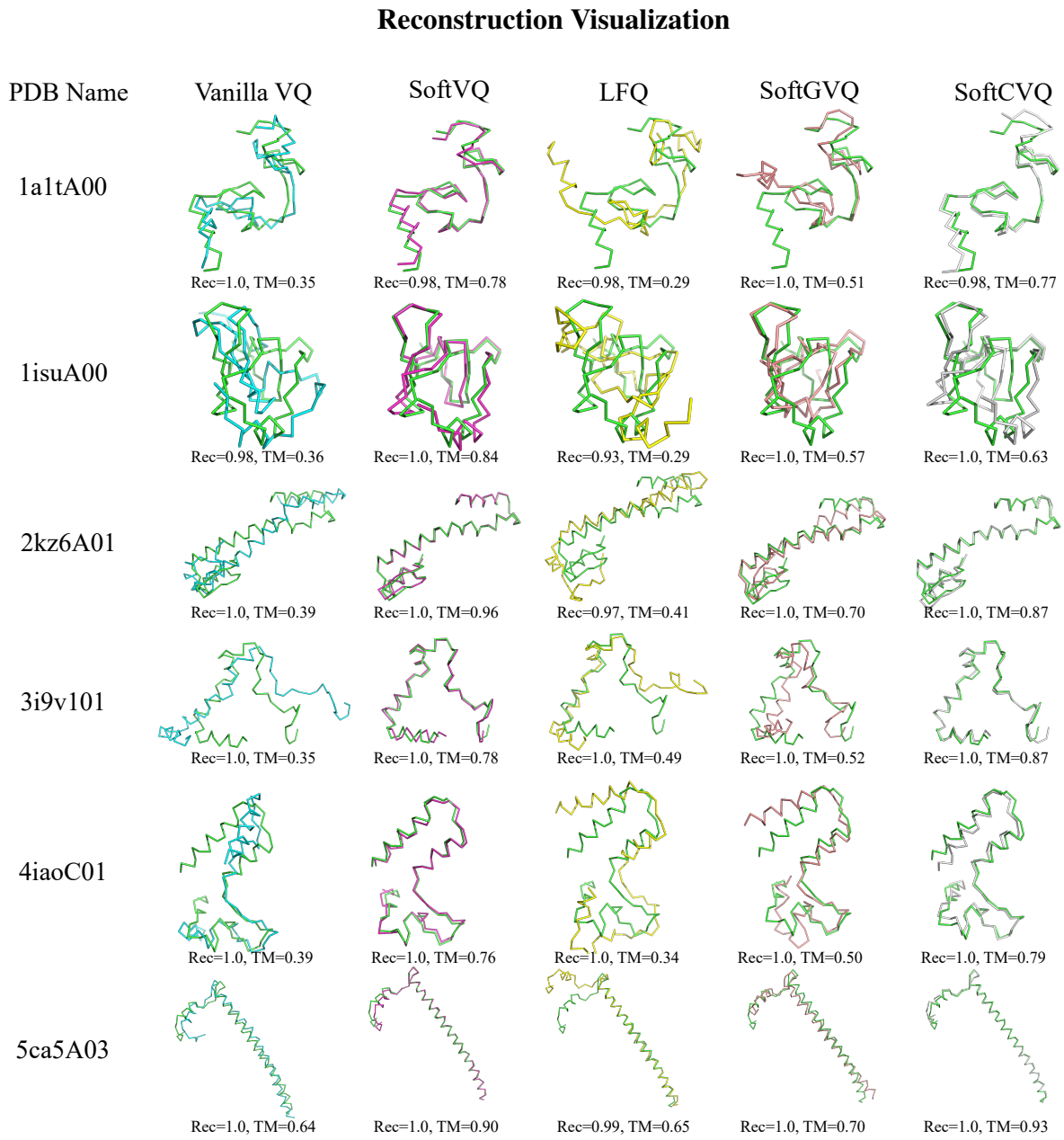


Figure 5: Reconstructed examples.

Image Generation

We conduct experiments on CelebA-HQ, containing 30K images at original 1024×1024 resolution. We use the github code to pretrain image tokenizers using LFQ (current SOTA approach) and SoftCVQ, and then training another transformers for on the latent code space for image generation. All the models are trained up to 50 epochs with batch size 24 (per gpu). Training each tokenizer requires approximately 7 hours, while training each generator takes around 45 hours using 4 NVIDIA-A100s. The codebook size is 1024, in line with previous research. We present the reconstruction and generation results in Table. 5, where SoftCVQ outperforms other methods in both tasks.

	FID(reconstruct) ↓	PSNR(reconstruct) ↑	FID(generation) ↓
VQ-VAE (Wang et al. 2018)	28.38	23.39	39.57
VQ-GAN (Esser, Rombach, and Ommer 2021)	12.74	22.44	17.42
Gumbel-VQ (Baevski, Schneider, and Auli 2019)	12.03	20.92	16.78
Reg-VQ (Zhang et al. 2023a)	10.09	22.05	15.34
LFQ (Yu et al. 2023)	7.50	21.89	15.11
SoftCVQ	6.62	27.97	13.36

Table 5: Image reconstruction and generation results on CelebA dataset.