GCP-VQVAE: A GEOMETRY-COMPLETE LANGUAGE FOR PROTEIN 3D STRUCTURE

Mahdi Pourmirzaei [1,2,*] **Alex Morehead** [3] **Farzaneh Esmaili** [1] **Jarett Ren** [4]

Mohammadreza Pourmirzaei [5] **Dong Xu** [1]

[1] University of Missouri
{mpngf,f.esmaili,xudong}@missouri.edu
[2] ProGene
[3] Lawrence Berkeley National Laboratory (Berkeley Lab)
acmwhb@lbl.gov
[4] Carnegie Mellon University
jzren@andrew.cmu.edu
[5] Politecnico di Milano, Milan, Italy
mohammadreza.pourmirzaeioliaei@mail.polimi.it

ABSTRACT

Converting protein tertiary structure into discrete tokens via vector-quantized variational autoencoders (VQ-VAEs) creates a language of 3D geometry and provides a natural interface between sequence and structure models. While pose invariance is commonly enforced, retaining chirality and directional cues without sacrificing reconstruction accuracy remains challenging. In this paper, we introduce GCP-VQVAE, a geometry-complete tokenizer built around a strictly SE(3)-equivariant GCPNet encoder that preserves orientation and chirality of protein backbones. We vector-quantize rotation/translation-invariant readouts that retain chirality into a 4 096-token vocabulary, and a transformer decoder maps tokens back to backbone coordinates via a 6D rotation head trained with SE(3)-invariant objectives.

Building on these properties, we train GCP-VQVAE on a corpus of 24 million monomer protein backbone structures gathered from the AlphaFold Protein Structure Database. On the CAMEO2024, CASP15, and CASP16 evaluation datasets, the model achieves backbone RMSDs of 0.4377 Å, 0.5293 Å, and 0.7567 Å, respectively, and achieves 100% codebook utilization on a held-out validation set, substantially outperforming prior VQ-VAE–based tokenizers and achieving state-of-the-art performance. Beyond these benchmarks, on a zero-shot set of 1 938 completely new experimental structures, GCP-VQVAE attains a backbone RMSD of 0.8193 Å and a TM-score of 0.9673, demonstrating robust generalization to unseen proteins. Lastly, we elaborate on the various applications of this foundation-like model, such as protein structure compression and the integration of generative protein language models. We make the GCP-VQVAE source code, zero-shot dataset, and its pretrained weights fully open for the research community.

1 INTRODUCTION

Proteins are the molecular machines of life, and their function is intricately tied to their three-dimensional structures (Bertoline et al., 2023; Tripathi et al., 2025). Understanding and predicting these structures remains one of the central challenges in computational biology (Jänes & Beltrao, 2024). Just as natural language is governed by grammatical and contextual rules, protein 3D structures exhibit spatial patterns and constraints that suggest an underlying "grammar" of folds and interactions (Ruff et al., 2022; Kilgore et al., 2025; Weissenow & Rost, 2025).

Despite advances in protein structure prediction, effectively representing the 3D geometry of proteins in a form suitable for generative modeling remains an open problem (Draizen et al., 2024; Lu

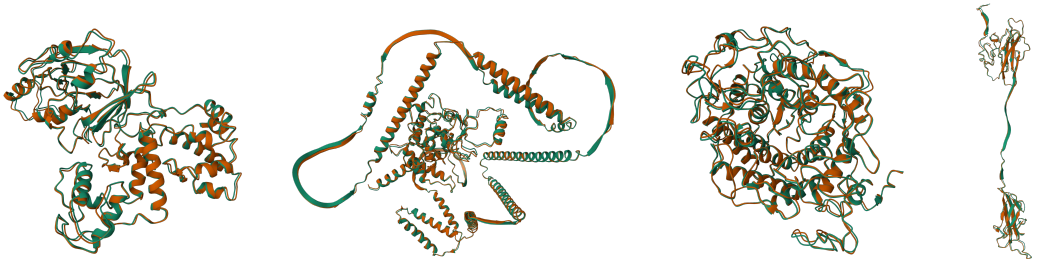

Figure 1: Superposition of GCP-VQVAE Large reconstructions (orange) with native structures (green) for four long, held-out backbone coordinates of proteins. Left→right: T1079 (CASP14; 482 residues; TM-score 0.9909; RMSD 0.7533 Å), T2272S8 (CASP16; 818 residues; TM-score 0.9967; RMSD 0.5639 Å), T1157s1-D1 (CASP15; 661 residues; TM-score 0.9920; RMSD 0.8172 Å), and 8BQU, chain A (CAMEO2024; 298 residues; TM-score 0.9764; RMSD 0.9981 Å).

et al., 2025). While recent methods have begun to leverage generative AI, such as diffusion models and autoregressive frameworks, to produce full-atom structures or backbone coordinates, they still lag behind other domains such as language or vision in terms of reconstruction precision, scalability to large and diverse datasets, and openness for the broader research community, like natural language or image and video generation (Yuan et al., 2025b). These gaps continue to constrain our ability to build powerful, general-purpose protein models using modern AI techniques.

Beyond generative modeling, learning a discrete language for protein 3D structure opens up a wide range of downstream applications. First, compressing 3D coordinates into compact sequences of integer codes—while preserving accurate reconstruction—can substantially reduce storage and transmission costs for structural data (Kim et al., 2023). Second, discrete structural representations enable fast, alignment-style comparison of protein shapes, analogous to multiple sequence alignment in sequence space (Van Kempen et al., 2024). Third, in learnable quantization frameworks such as vector-quantized autoencoders, these codes can be decoded into semantically rich continuous embeddings (Yuan et al., 2025b), facilitating structure-aware feature extraction for classification, clustering, structure-based comparison/search, and other predictive tasks. Finally, unifying protein sequence and structure through a shared discrete representation may pave the way for multimodal generative models that bridge amino acid sequences and 3D folds within a common language modeling framework (Hsieh et al., 2025).

However, most existing protein-structure VQVAEs are either closed-source or only partially released (e.g., code without full evaluation scripts or strongest checkpoints), which impedes reproducible comparison (Gao et al., 2024c; Hayes et al., 2025). Furthermore, the publicly available baselines often generalize weakly to *unseen* proteins. Consequently, the field lacks a fully open-source, high-accuracy tokenizer with transparent training and evaluation that demonstrably transfers to new proteins.

Contributions. (1) We introduce GCP-VQVAE, a geometry-complete tokenizer that preserves orientation and chirality while producing pose-invariant codes that support missing coordinates. (2) We scale training to 24M monomers and report exhaustive evaluations (CAMEO2024, CASP14/15/16) and a zero-shot suite of newly deposited experimental structures. (3) Our model achieves state-of-the-art reconstruction on a diverse range of unseen 3D structures, and stays ahead of other open-source methods. On the zero-shot set, GCP-VQVAE attains a backbone RMSD of 0.8193 Å and a TM-score of 0.9673. (4) We release code, checkpoints, and unified evaluation scripts for ourselves and baselines, enabling reproducible comparison.

2 Related Work

An early and influential approach to casting protein 3D structure as a discrete language is FoldSeek van Kempen et al. (2022), which learns a 20-state 3Di alphabet with a VQ-VAE trained for evolutionary conservation and encodes structures as token sequences for ultra-fast k-mer–based local/-global alignment. Building on the notion of a discrete structural language—but targeting generative

reconstruction rather than search—the FoldToken series (Gao et al., 2025; 2024a;b;c) develops a VQ-VAE–style tokenizer and decoder: FoldToken introduces a SoftCVQ fold language with joint sequence–structure generation; FoldToken2 stabilizes quantization and extends to multi-chain settings; FoldToken3 mitigates gradient/class-space issues to reach 256-token compression with minimal loss; and FoldToken4 unifies cross-scale consistency and hierarchies in a single model, reducing redundant multi-scale training and code storage.

Yuan et al. (2025b) introduces AminoAseed codebook reparameterization plus Pareto-optimal K×D sizing, proposes structtokenbench—a fine-grained evaluation suite—and diagnoses codebook underutilization in VQVAE PSTs. Concurrently, Gaujac et al. (2024) tokenizes protein backbones with a VQ autoencoder (codebooks 4k–64k); its main open-source limitation is that it does not support sequences shorter than 50 or longer than 512 amino acids.

ESM-3 (Hayes et al., 2025) couples its multimodal transformer with a VQVAE structure tokenizer that discretizes local 3D geometry into structure tokens; structure, sequence, and function are jointly trained under a masked-token objective. The tokenizer uses an SE(3)-aware module within the encoder, and ESM-3 employs a 4 096-code structure codebook (plus special tokens) for downstream generation and masked reconstruction.

Across prior structure tokenizers, neither openness nor accuracy are yet satisfactory. The FoldToken line spans four variants, but to our knowledge, only FoldToken-4 offers a partial open release, without the strongest checkpoints, limiting reproducibility (Gao et al., 2024c). ESM-3 exposes an internal VQ-VAE tokenizer, yet public artifacts lack fully documented training/evaluation procedures and best weights, making directly comparable reconstruction benchmarking difficult (Hayes et al., 2025). The open VQ autoencoder of Gaujac et al. (2024) supports a restricted length window (e.g., ∼50–512 residues), precluding fair assessment on long chains where reconstruction accuracy degradation is most evident. Finally, FoldSeek's learned 3Di alphabet targets ultra-fast structure search and does not provide a generative decoder from discrete codes back to coordinates, so reconstruction fidelity cannot be evaluated (van Kempen et al., 2022). Consequently, the community still lacks a fully open, end-to-end tokenizer with released best weights and source codes that attains high reconstruction accuracy on *unseen* proteins.

3 DATASET

We began with the latest release of UniRef50 (Consortium, 2019) [1], which clusters protein sequences at 50% identity and thus offers a natural, non-redundant scaffold for large-scale structure modeling. For every UniRef50 entry with a corresponding model in the AlphaFold Database (AFDB; Varadi et al. (2022)), we downloaded the per-protein structure (PDB format at collection time). Limiting to UniRef50 reduces near-duplicate leakage by ensuring that homologs within clusters do not exceed 50% identity. After parsing and splitting multi-chain records into individual chains, this procedure yielded approximately 42M single-chain PDB samples.

From this AFDB ∩ UniRef50 pool, we drew a uniform random sample of 24M single-chain structures to form the training set. This down-sampling keeps training throughput tractable while preserving the global distribution of lengths, folds, and taxa present in the full pool. To support our lightweight model variant, we also employed the `afdb_rep_v4` dataset (Jamasb et al., 2024), which contains approximately 2.2 million cluster-representative structures, serving as a data-efficient alternative to the full training corpus.

We evaluate on three held-out test suites capturing complementary shifts (see Table 1): *A* low-confidence AF2 SwissProt models, *B* species-diverse (97-species) taxonomic shift, and *C* high-confidence structures from under-represented taxa. All suites are per-chain, length-filtered, deduplicated, and NaN-aware; a balanced validation set of 6,000 chains (2k per suite) is sampled disjointly. Full curation criteria, thresholds, and counts are provided in Appendix A.2 .

To probe generalization to genuinely new structures, we assembled a zero-shot suite from *recently deposited experimental* monomeric chains in the PDB (calendar years 2024–2025). Multi-chain entries were split into per-chain backbones, chains shorter than 25 residues or longer than 2 048 residues were discarded, and we retained entries even when they contained missing coordinates (`NaN`) to reflect typical experimental gaps, yielding 76 503 pdbs in total. We deduplicated using

[1] March 2024 release.

Table 1: Dataset and benchmark statistics. Training data is deduplicated at 100% sequence identity against all validation/test splits and external benchmarks (CAMEO2024, CASP14–16). We also include a zero-shot set of 1 938 newly deposited experimental monomer chains (PDB, 2024–2025).

Split	Source	Selection criterion	# Samples
Training (Large)	AFDB ∩ UniRef50	Uniform random sample from ∼42M pool	24 000 000
Training (Lite)	AFDB	Cluster representatives (`afdb_rep_v4`)	2 200 000
Validation	From Tests A/B/C	2k random per test (A,B,C) merged	6 000
Test set A	PDB DB	cross-referenced AF pLDDT ≤ 70; len ≥ 25; < 25 consecutive missing residues; dedup	31 112
Test set B	AFDB	97 species; ∼2.5k/species; len ≥ 25; dedup	17 353
Test set C	PDB DB	under-represented taxonomies in SwissProt; pLDDT ≥ 95; < 25 consecutive missing residues; dedup	31 867
Zero-shot (experimental)	PDB DB	recent experimental monomers (2024–2025); len 25–2048; dedup; $< 50\%$ seq id vs train	1 938
CAMEO2024	CAMEO	-	574
CASP14	CASP	-	33
CASP15	CASP	-	45
CASP16	AF3 predictions	-	104

structural clustering with FoldSeek `easy-cluster` (alignment type 2, tmscore threshold 0.80, cov-mode 0, lddt-threshold 0.65), retaining one representative per cluster. This resulted in 10 689 clusters (van Kempen et al., 2022). For leakage control, each representative was compared with foldseek `easy-search` against (i) the AFDB ∩ UniRef50 set used for our GCP-VQVAE Large model training and (ii) the `afdb_rep_v4` (Barrio-Hernandez et al., 2023) representative set used to pre-train the GCPNet encoder in ProteinWorkshop (Jamasb et al., 2024) and training the GCP-VQVAE Lite model. We *excluded* any candidate with $\geq 50\%$ structure similarity to any chain in those sources; the resulting zero-shot set comprises 1 938 monomer chains.

Independent benchmarks. In addition to the internal splits, we evaluate on community benchmarks to facilitate comparison with prior work: CAMEO2024 (Robin et al., 2021; Leemann et al., 2023), CASP14 (Kryshtafovych et al., 2021), CASP15 (Kryshtafovych et al., 2023), and AlphaFold3 (Abramson et al., 2024) predictions for CASP16 (Yuan et al., 2025a) targets. Each benchmark is processed with the same per-chain extraction and deduplication pipeline and is used strictly for out-of-distribution evaluation.

Table 1 summarizes the composition and selection criteria of all splits. Counts after converting multi-chain inputs into per-chain samples. Also, all samples are truncated to a maximum of 2048 amino acids in length.

4 GCP-VQVAE Architecture

The proposed architecture leverages two main parts: (1) a GCPNet encoder to encode backbone coordinates into embeddings, and (2) a transformer-based VQVAE, which discretizes backbone embeddings and then converts them back into 3D coordinates.

4.1 GCPNet Encoder

GCPNet (Morehead & Cheng, 2024b) extends the scalar–vector message-passing philosophy of GVP-GNN to a geometry-complete, $\mathrm{SE}(3)$-equivariant encoder. Every atom i in a molecular graph $G = (V, E)$ carries scalars $s_i \in \mathbb{R}^{d_s}$ and row-wise vectors $\mathbf{v}_i \in \mathbb{R}^{d_v \times 3}$ that rotate as $\mathbf{v}_i \mapsto R\mathbf{v}_i$ under $g = (R, \mathbf{t}) \in \mathrm{SE}(3)$, while each edge (i, j) stores analogous features $\big(s_{ij}, \mathbf{v}_{ij}\big)$ and the relative displacement $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$. Before any update, the model attaches to every edge a right-handed orthonormal frame $F_{ij} = [\mathbf{a}_{ij}, \mathbf{b}_{ij}, \mathbf{c}_{ij}] \in \mathrm{SO}(3)$ with $\mathbf{a}_{ij} = \mathbf{r}_{ij}/\|\mathbf{r}_{ij}\|$; this frame supplies a reference for chirality and orientation that GVP-GNN lacks.

The core computation is a Geometry-Complete Perceptron (GCP) micro-step that first down-scales the vectors and then projects them into the local frame to extract nine orientation-aware features. Denoting $\mathbf{z}_{ij} = W_d \mathbf{v}_{ij}$ and $\mathrm{vec}(\cdot)$ the row-wise vectorization, the joint update mixes the old scalars with their orientation signatures (the frame-projected vectors and their norms) and gates the vectors through a learnable row-wise gate to preserve equivariance; ϕ_s is an MLP and σ denotes a learnable gating function that need not be a fixed sigmoid (Equation 1).

$$\big(s_{ij}, \mathbf{v}_{ij}\big) \;\mapsto\; \Big(\phi_s\big[s_{ij}, \|\mathbf{z}_{ij}\|_2, \mathrm{vec}(\mathbf{z}_{ij} F_{ij}^{\top})\big],\; \sigma\big(W_g s_{ij}\big) \odot W_v \mathbf{z}_{ij}\Big) \tag{1}$$

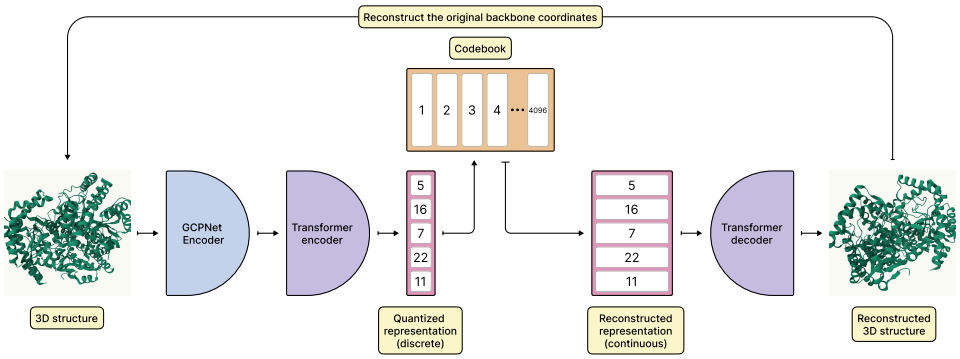

Figure 2: GCP-VQVAE overview. A protein backbone (N–C_α–C) is first encoded by a $\mathrm{SE}(3)$-equivariant GCPNet that preserves orientation and chirality. A Transformer encoder produces latents that are vector-quantized into a sequence of code indices from a 4 096-entry codebook (e.g., 5, 16, 7, 22, 11), yielding pose-invariant discrete tokens. For reconstruction, the indices are de-quantized to continuous embeddings and passed to a Transformer decoder equipped with a 6-D rotation head, which predicts rigid updates to recover the original backbone coordinates.

In the node update, the orientation features are averaged over neighbors. A sequence of such micro-steps, wrapped by a residual shortcut, forms a GCPConv edge block. After aggregating messages $m_i = \sum_{j \in \mathcal{N}(i)} \mathrm{GCPConv}(i, j)$, a gated scalar–vector MLP updates the node features, and stacking L layers yields an invariant backbone. When tasks require coordinates, each layer appends an equivariant displacement head whose output is a learned 3-dimensional vector; this vector is added residually to $\mathbf{x}_i$ and re-centered to ensure translation invariance, enabling force or trajectory prediction without breaking equivariance.

$$\mathbf{x}_i \;\leftarrow\; \mathbf{x}_i + \mathbf{f}_i\,, \qquad \mathbf{f}_i = \mathrm{MLP}_{\mathrm{disp}}(m_i) \tag{2}$$

Equations 1 and 2 commute with every rigid motion, making the encoder strictly $\mathrm{SE}(3)$-equivariant, while projection through F_{ij} preserves the complete set of edge orientations so that the latent remains *geometry-complete* at any depth. Eliminating the frame ($F_{ij} = I$), replacing the orientation features by $\|\mathbf{v}_{ij}\|_2$, and dropping the coordinate head restores a frame-free, $\mathrm{E}(3)$-equivariant network akin to GVP-GNN—thereby isolating the contributions responsible for our empirical gains. Because GCPNet keeps directional and chiral cues that GVP-GNN discards, it supports optional equivariant coordinate updates for force or dynamics prediction and attains improved performance across invariant (e.g., binding affinity; Morehead & Cheng (2024b)), equivariant (e.g., force regression; Morehead & Cheng (2024b)), and coordinate-generative (e.g., molecular diffusion; Morehead & Cheng (2024a)) tasks with only modest extra computation.

4.2 VQVAE

The transformer–based VQVAE employed in this work is organized into the classical three-stage pipeline of *encoder*, *vector-quantization*, and *decoder*. The encoder processes the given embeddings into a sequence of latent vectors, the quantizer discretizes these latents, and the decoder reconstructs the original signal from the resulting code indices.

Both stacks adopt a lightweight pre-layer-normalized Transformer that integrates several recent efficiency upgrades: (i) Pre-LayerNorm places the LayerNorm before each sub-block (Xiong et al., 2020), which keeps activations in a well-behaved range throughout the network, reduces gradient-scale drift, and therefore allows training with larger learning rates and much milder warm-up schedules; (ii) separately normalizes query and key vectors before computing attention logits (Henry et al., 2020). This prevents overly large dot-products, stabilizes attention distributions, and mitigates softmax saturation, especially in low-resource or small-batch training scenarios; (iii) Grouped-Query Attention shares key–value projections across groups of query heads (Ainslie et al., 2023), reducing both memory and compute without harming quality; (iv) Rotary Positional Embeddings (RoPE) inject relative-position information by applying position-dependent planar rotations to each query–key

pair (Su et al., 2024), letting the model generalize to much longer sequences with virtually no extra computational cost; (v) We remove bias terms from projection and feed-forward layers, an established simplification that has negligible effect on accuracy while trimming a fraction of parameters and FLOPs.

In the architecture, the vector quantization layer provides the discrete bottleneck between the aforementioned encoder and decoder and follows the learnable formulation of VQVAE, augmented with several improvements described in the following to accelerate convergence, increase codebook usage, and stabilize large codebooks training.

Before training starts, we run k-means on the encoder outputs of the first mini-batch to seed the codebook as displayed in Equation 3, where $Z^{(0)} \subset \mathbb{R}^d$ are the features, K the codebook size, and T the number of Lloyd iterations. Empirically, this step mitigates early code collapse and improves utilization when K is large.

$$E^{(0)} = \mathrm{KMEANS}\big(Z^{(0)}, K, T\big), \tag{3}$$

Given an encoder vector $\mathbf{z} \in \mathbb{R}^d$, quantization proceeds by nearest-neighbor lookup, Equation 4, which is identical to the vanilla VQ rule.

$$k = \arg\min_j \big\|\mathbf{z} - \mathbf{e}_j\big\|_2^2, \qquad \mathbf{z}_q = \mathbf{e}_k, \tag{4}$$

To transmit gradients through this non-differentiable operation, instead of using the straight-through estimate (STE) introduced in Van Den Oord et al. (2017), we adopt the rotation trick (Fifty et al., 2024). During back-propagation, the Jacobian is replaced by Equation 5, where R is the shortest-arc rotation aligning the unit vectors $\hat{\mathbf{z}}$ and $\hat{\mathbf{e}}_k$. This modification embeds both angular and magnitude mismatch into the back-propagated signal, yielding faster convergence and richer code usage in practice.

$$\mathbf{J}_k = \frac{\|\mathbf{e}_k\|}{\|\mathbf{z}\|}\, R\big(\hat{\mathbf{z}} \to \hat{\mathbf{e}}_k\big), \tag{5}$$

The codebook updates and the encoder commitment follow the standard VQ losses (Equation 6), with β balancing the commitment pressure.

$$\mathcal{L}_{\text{code}} = \|\mathrm{sg}[\mathbf{z}] - \mathbf{e}_k\|_2^2, \qquad \mathcal{L}_{\text{commit}} = \beta\, \|\mathbf{z} - \mathrm{sg}[\mathbf{e}_k]\|_2^2, \tag{6}$$

To encourage diverse and well-separated embeddings, we regularize the codebook with the Frobenius norm (Equation 7) weighted by λ_{orth}. This orthogonality constraint spreads codes over the hypersphere and curbs under-utilization, which is a common failure mode in large books.

$$\mathcal{L}_{\text{orth}} = \big\|E^\top E - I_K\big\|_F^2, \tag{7}$$

To translate the decoder's abstract embeddings into a physically meaningful 3D structure, we utilize a 6D rotation head on top of the decoder. This module's primary purpose is to provide a stable and continuous parameterization of 3D rotations and translations, which is crucial for effective training of deep neural networks. This approach, notably used in AlphaFold2 and supported by the findings of Zhou et al. (2019) on rotation representations, avoids the well-known issues of Gimbal lock in Euler angles and the double-cover ambiguity of quaternions.

Intuitively, the head operates by predicting an intermediate representation for each residue, comprising two 3D direction vectors and a translation. The direction vectors are deterministically converted into a stable rotation matrix via the Gram-Schmidt process, while the translation is scaled by a hyper-parameter, α, to an arbitrary range (e.g., Å). This resulting rigid transformation acts as an update, which is composed with the residue's running pose. The final backbone coordinates are then generated by applying this new, refined pose to a fixed local atomic template ($\mathbf{X}^{\text{local}}$). This iterative process ensures the structure is built in a geometrically consistent and equivariant manner, with the full operational details provided in Algorithm 1.

The decoder is supervised with a weighted sum of three geometric objectives. Building on the loss terms defined in Algorithm 2 (distance, direction and aligned MSE) and the Kabsch alignment returned by Algorithm 3, we supervise the decoder with the weighted sum of Equation 8 as the reconstruction part (L_{rec}) of the final loss.

$$\mathcal{L}_{\text{rec}} \;=\; \lambda_{\text{mse}}\,\mathcal{L}_{\text{MSE}} + \lambda_{\text{dist}}\,\mathcal{L}_{\text{dist}} + \lambda_{\text{dir}}\,\mathcal{L}_{\text{dir}}, \tag{8}$$

Here, $\mathcal{L}_{\text{MSE}}$ is the mean-squared error between predicted and native backbones after Kabsch alignment, $\mathcal{L}_{\text{dist}}$ penalises deviations in the $3L \times 3L$ backbone distance matrix (clamped at 5 Å^2), and $\mathcal{L}_{\text{dir}}$ measures squared differences of pairwise dot-product tensors over the six orientation vectors per residue (clamped at 20). Finally, the overall objective optimized for each iteration is demonstrated in Equation 9 where $\mathcal{L}_{\text{rec}}$ is the decoder reconstruction.

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{code}} + \lambda_{\text{commit}}\,\mathcal{L}_{\text{commit}} + \lambda_{\text{orth}}\,\mathcal{L}_{\text{orth}}, \tag{9}$$

5 EXPERIMENTS

In this section, optimization is used with AdamW (Loshchilov, 2017) and a cosine-annealed learning-rate schedule with warm-up steps (Loshchilov & Hutter, 2016). Experiments were implemented in PyTorch 2.7 (Ansel et al., 2024) [2] with mixed-precision (BF16) training (Kalamkar et al., 2019; Micikevicius et al., 2017) on four nodes of NVIDIA 8 × A100 GPUs. For the GCP-Net encoder, we initialized from the ProteinWorkshop checkpoint (Jamasb et al., 2024). Both light and large GCP-VQVAE configurations and exact hyperparameter values are listed in Appendix A.3; Table 7 and 8.

For the GCP-VQVAE Lite variant, we employed the `afdb_rep_v4` dataset from the ProteinWorkshop benchmark (Jamasb et al., 2024), which consists of approximately 2.27 million representative AlphaFold structures. This roughly 10-fold reduction in training data relative to our primary model allows us to demonstrate the architectural efficiency and robust performance of GCP-VQVAE even under significantly constrained computational budgets.

Table 2: Reconstruction comparison with the available open-source structure tokenizer methods. FoldToken-4, AminoAseed, and DPLM-2 use 256, 512, and 8 192 vocab sizes, respectively. Other methods use 4 096 vocab size. The Structure Tokenizer of Gaujac et al. (2024) only supports sequences of length 50–512. For AIDO, sequences longer than 1024 are omitted due to OOM.

Dataset	**Metric**	**GCP-VQVAE Large (Ours)**	**GCP-VQVAE Lite (Ours)**	**AIDO** (Zhang et al., 2024)	**ESM-3 VQVAE** (Hayes et al., 2025)	**DPLM-2 VQVAE** (Wang et al., 2024)	**FoldToken 4** (Gao et al., 2024c)	(Gaujac et al., 2024)	**AminoAseed** (Yuan et al., 2025b)
CASP14	TM-score ↑	**0.9890**	0.9751	0.9831	0.9734	0.9056	0.8989	0.8940	0.3951
	RMSD ↓	**0.5431**	0.8435	0.6587	1.0968	2.0896	2.0027	1.8913	14.6739
CASP15	TM-score ↑	**0.9884**	0.9665	0.9660	0.9418	0.8268	0.9136	0.7911	0.3831
	RMSD ↓	**0.5293**	0.9219	0.8611	1.3407	6.3826	1.6856	8.5592	18.5827
CASP16	TM-score ↑	**0.9857**	0.9757	0.8805	0.9103	0.7852	0.8211	0.8345	0.3433
	RMSD ↓	**0.7567**	1.0614	4.9773	4.4918	14.1934	5.4115	6.9375	25.9508
CAMEO2024	TM-score ↑	**0.9918**	0.9794	0.9851	0.9736	0.9243	0.9346	0.9114	0.4632
	RMSD ↓	**0.4377**	0.7401	0.5867	1.0156	1.941	1.4069	2.1726	12.6203
Zero-Shot	TM-score ↑	**0.9673**	0.9466	0.9202	0.8466	0.7137	0.8256	0.7662	0.4155
	RMSD ↓	**0.8193**	1.1152	2.0105	6.5075	12.2182	4.5465	10.4405	19.2612
Average	TM-score ↑	**0.9844**	0.9687	0.9470	0.9291	0.8311	0.8788	0.8394	0.4000
	RMSD ↓	**0.6172**	0.9364	1.8189	2.8905	7.3650	3.0106	6.0002	18.2178

Training of the large model proceeded in two stages on the same training split. In Stage 1, sequences are truncated to 512 residues. In Stage 2, the maximum length is increased to 1 280 residues. In Stage 2, we also introduce a simple `NaN`-masking augmentation to handle missing coordinates; implementation details are provided in Appendix A.3 (Figure 5). At the end of Stage 2 training, on the validation set, the model attains MAE 0.2239, RMSD 0.4281, GDT-TS 0.9856, and TM-score 0.9889 with 100% codebook utilization. Similarly, the training of the lite model proceeded by one stage only with 1 280 residues. On the validation set, the model attains MAE 0.3961, RMSD 0.5707, GDT-TS 0.9327, and TM-score 0.9741 with 100% codebook utilization.

Per-test-set results of the large model appear in Table 3.

We compare against open-source structure tokenizers: FoldToken-4, ESM-3 VQ-VAE, and the Structure Tokenizer of Gaujac et al. (2024). In practice, ESM-3 does not provide documented usage

[2] We built the VQVAE components extensively by using the x-transformers and vector-quantize-pytorch libraries.

Table 3: Evaluate our GCP-VQVAE large method on the predefined test sets.

Method	**Test set A**	**Test set B**	**Test set C**
RMSD ↓	0.5124	0.4027	0.4787
TM-score ↑	0.9823	0.9951	0.9885

or official evaluation scripts for its VQ-VAE; we therefore reproduced its evaluation using the publicly released weights and their GitHub codebase. The original FoldToken-4 evaluation repository was prohibitively slow, so we re-implemented it, yielding a $\sim$20$\times$ speed-up with negligible loss in reconstruction rate. For Gaujac et al. (2024) we used the authors' released scripts with the 4 096-entry codebook checkpoint. See Table 2 for aggregate metrics; Figure 4 visualizes the full RMSD error across test sets.

We examined how reconstruction error varies with sequence length on the validation set (Appendix A.3; Figure 6). Errors increase only mildly with length, indicating stable accuracy up to 1 280 residues with only a slight variance rise for very long chains. On the zero-shot set we obtain mean RMSD 0.8193 Å and TM-score 0.9673 (Table 2), with group-wise trends summarized in Table 4; extended zero-shot analyses appear in Appendix A.4.

Table 4: Metrics by structure group on the zero-shot dataset with respect to the GCP-VQVAE Large . Groups are defined by secondary-structure composition thresholds and are not mutually exclusive. Thresholds of each groups are described in Appendix A.2

Group	**Count**	**RMSD ↓**	**TM-score ↑**
All rows	1938	0.8193	0.9673
A (mainly α)	776	0.7537	0.9707
B (mainly β)	309	0.7635	0.9762
AB (mixed)	71	0.6166	0.9878
D (coil-rich/disordered)	552	0.9476	0.9498
No-NaN residues	1231	0.6179	0.9755
Has-NaN residues ($\geq$1)	707	1.1699	0.9531

5.1 Codebook Evaluation

Table 5: Comprehensive codebook analysis comparing structure tokenization methods on the zero-shot test set.

			Unigram Statistics			**Zipf Fit**		**Conditional Stats**			**Mutual Info (bits)**	
Model	**Vocab**	**Util.**	H_1 ↑	PPL ↑	Usage ↑	Slope (b)	R^2	$H(Z_t \mid Z_{t-1})$ ↓	$H(Z_t \mid Z_{t-2}, Z_{t-1})$ ↓	Tri PPL ↓	$I(Z_t; Z_{t+1})$ ↑	$I(Z_t; Z_{t+4})$ ↑
GCP-VQVAE (Lite)	4096	99.8%	9.266	615.8	15.1%	−1.390	0.954	7.309	2.095	4.27	1.958	1.954
GCP-VQVAE (Large)	4096	**100%**	11.326	2567.6	62.7%	−0.804	0.844	7.247	0.232	1.17	4.082	4.104
ESM3 VQVAE	4096	95.2%	11.269	2467.0	60.2%	−0.750	0.665	6.924	0.549	1.46	4.346	4.359
AIDO	512	100%	8.967	500.3	**97.7%**	−0.197	0.744	8.310	1.316	2.49	0.656	0.732
DPLM-2	8192	94.7%	**12.250**	**4872.3**	62.8%	−0.987	0.647	6.341	**0.215**	**1.16**	**5.910**	**5.905**
FoldToken	256	**100%**	7.649	200.7	78.4%	−0.696	0.735	**3.731**	3.313	9.94	3.916	2.097
(Gaujac et al., 2024)	4096	58.8%	10.700	1662.9	69.1%	−0.991	0.553	7.149	0.280	1.21	3.548	3.670

We analyzed the distribution and informational properties of the learned discrete codes on the zero-shot test set (Table 5). GCP-VQVAE demonstrates superior codebook efficiency, achieving 100% utilization of its vocabulary, whereas baselines such as Gaujac et al. (2024) suffer from collapse (58.8% utilization). In terms of structural grammar, the Large model exhibits a transition from high unigram entropy ($H_1 = 11.3$ bits) to extremely low conditional trigram entropy (≈ 0.23 bits). This sharp drop implies that the model has learned a highly structured 'syntax' for protein coordinates, where local dependencies (likely corresponding to secondary structure constraints) make token sequences highly predictable locally, similar to the state-of-the-art DPLM-2 and ESM3 models. Conversely, the Lite model exhibits a steeper Zipfian slope (-1.39), indicating a sparser representation strategy that relies on a concentrated set of high-frequency structural primitives.

Geometrically, the reconstructed backbone dihedrals closely imitate the original distribution, validating the decoder's precision. However, the observation (Appendix Figure 15) that individual top codes often span multiple Ramachandran regions suggests that tokens are not uniquely tied to nar-

Table 6: Performance on unsupervised Apo Holo and Fold Switching benchmarks.

Model	Apo Holo		Fold Switching	
	PCC%	Spearman's ρ%	PCC%	Spearman's ρ%
ProteinMPNN	35.80	45.22	51.46	55.91
MIF	35.82	43.55	54.48	59.27
FoldSeek	34.79	43.03	51.88	56.54
ProTokens	43.32	54.05	<u>61.20</u>	65.90
ESM3	39.76	50.97	57.12	62.23
VanillaVQ (*v.s.* ESM3)	39.62	50.61	56.51	61.51
AminoAseed (*v.s.* ESM3)	42.47	54.88	**65.61**	<u>75.89</u>
Our model (lite)	**45.97**	**57.44**	60.83	**75.96**
Our model (large)	<u>45.74</u>	<u>56.58</u>	59.91	74.25

row geometries but cover broader conformational modes, likely relying on sequence context for disambiguation (Appendix A.6).

5.2 Representation Evaluation

Beyond raw coordinate reconstruction, we evaluate the semantic quality of the learned latent space using the PST benchmark (Yuan et al., 2025b). We extract the continuous, de-quantized embeddings from the codebook and assess their utility across both supervised functional tasks and unsupervised conformational analyses.

Table 9 in Appendix A.5 presents the results on functional site prediction tasks. We observe that GCP-VQVAE achieves performance comparable to other VQ-VAE and structure-based baselines. Notably, on interaction-centric tasks such as BindInt (Fold split) and BindShake, our model performs on par with baselines like VanillaVQ and AminoAseed. While sequence-heavy models (e.g., ProteinMPNN) generally score higher on broad classification, these results confirm that our geometry-complete latent space retains meaningful functional signals without relying on explicit sequence objectives.

However, on tasks requiring sensitivity to structural variations—specifically Apo/Holo state detection and Fold Switching—our model demonstrates robust performance (Table 6). By strictly preserving chirality and orientation, the GCP-VQVAE embeddings effectively capture the subtle geometric shifts inherent in protein state transitions, achieving the highest correlation metrics among evaluated methods. This suggests that our representation is particularly well-suited for applications demanding fine-grained structural discrimination rather than broad functional categorization.

6 Discussion and Future Work

GCP-VQVAE delivers high-fidelity reconstruction across diverse suites of benchmarks, with TM-scores typically ≥ 0.98 and mean RMSDs in the 0.40–0.80 Å range on CAMEO2024, CASP14/15/16, and similar performance on our internal Test A/B/C splits (e.g., 0.40–0.51 Å RMSD). Crucially, relative to the current state-of-the-art method AIDO (Zhang et al., 2024), GCP-VQVAE not only yields lower reconstruction error but also accelerates inference by more than $200\times$ for the Large model and $400\times$ for the Lite variant.

On our diverse validation set, although overall reconstruction is strong, we observe a mild RMSD drift with sequence length (Figure 6). We attribute this to length imbalance in training; short chains dominate, leaving the model relatively underexposed to long-range constraints and rare structural motifs. Mitigations include targeted fine-tuning on a length-balanced subset, reweighting/upsampling long sequences, and length-aware objectives, which should reduce the slope without architectural changes.

On the zero-shot set of 1 938 newly deposited experimental monomers, the model maintains strong generalization with mean RMSD 0.8193 Å and TM-score 0.9673 (Table 4). Our diagnostics indicate that degradation is driven primarily by the ratio and count of missing coordinates (Appendix Figure 10, 9), and secondarily by sequence length (Appendix Figure 8), Similar to our interpretation of the validation results. Consistent with this, sequences without missing residues exhibit substantially

lower errors than those with gaps (Table 4). These trends suggest further gains from strengthening the NaN-masking augmentation (e.g., longer and multi-gap patterns) and increasing exposure to longer chains.

Relative to available open-source tokenizers, our method shows a large margin while maintaining near-ceiling TM-scores. We believe part of this disparity stems from incomplete public releases: several baselines either restrict sequence lengths or withhold their strongest checkpoints (as is evident for FoldToken-4 repository) and end-to-end evaluation scripts. To enable rigorous verification, we release our checkpoints and evaluation pipelines for both GCP-VQVAE and reproduced baselines, and we invite the community to run, audit, and extend these comparisons.

For the proposed GCP-VQVAE architecture, we see the following applications:

Structure compression. Using a 4 096-entry codebook yields about $\sim 24\times$ compression of backbone coordinates for a 512-residue monomer (Appendix A.8). Given our low RMSDs of ~ 0.4–0.8 Å (Table 2), this is a practical lossy backbone codec. Further gains are possible by (i) allowing a single code to represent short residue spans (e.g., 2–4 amino acids), reducing the token rate while the decoder expands span-tokens to per-residue poses, and/or (ii) increasing the codebook size (e.g., 8k–64k) to lower per-token quantization error so that fewer tokens are needed at a target distortion. These options trade bitrate against utilization and latency, and remain compatible with our VQ training approach.

Structure comparison. Our discrete geometry language converts backbones into high-resolution token strings that can be indexed (Altschul et al., 1990; Ma et al., 2002) and aligned with dynamic programming over a learned substitution matrix between codes. Compared to fixed 3Di alphabets as in FoldSeek, a learned 4 096-code vocabulary; optionally make it with side-chain–aware, captures finer local shape and orientation, enabling more accurate substructure and whole-chain comparisons.

Downstream 3D learning representation. De-quantized code embeddings yield continuous, structure-aware features that potentially offer a strong encoder for downstream tasks (Yuan et al., 2025b). As detailed in our representation evaluation (Section 5.2), while these features prove effective for structural state discrimination (e.g., Fold Switching), our partial benchmarking on functional tasks suggests that the learned latent space prioritizes geometric precision over broad functional semantics, which could be a weakness of the model. Notably, we observed limited performance differences between our Large and Lite model variants (Table 9), implying that higher reconstruction accuracy does not necessarily correlate with better separability for high-level classification tasks.

Generative modeling. Because our structure representation is discrete and pose-invariant, it slots directly into autoregressive protein language model (PLM) pre-training: tokens can be interleaved with amino-acid symbols, letting us exploit next-token scaling laws observed in autoregressive Large Language Models (LLM) (Kaplan et al., 2020). This unifies sequence and structure in a single model, enabling controllable generation via simple conditioning; e.g., sequence-given-structure and structure-given-sequence modes. Structure tokens serve as an explicit geometric prior for PLMs (Hayes et al., 2025; Su et al., 2023), potentially enabling more control over protein design. Moreover, the same machinery supports higher-throughput structure prediction from sequence by first predicting structural tokens and then mapping them to backbones with GCP-VQVAE's decoder (Pourmirzaei et al., 2025; Lu et al., 2024; Chen et al., 2024).

Several avenues look promising to explore: scaling GCP-VQVAE tokenization to multi-chain complexes, enriching tokens with side-chain information, and evaluating GenAI-friendliness of this new language to unify sequence and structure generation via autoregressive PLMs.

7 Conclusion

In this work, we have introduced GCP-VQVAE, a state-of-the-art open-source protein structure tokenizer. GCP-VQVAE offers improved structure reconstruction quality and generalization compared to prior methods, which could enable new (sequence and structure-based) predictive and generative modeling applications in future work.

REFERENCES

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

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 929–947, 2024.

Inigo Barrio-Hernandez, Jingi Yeo, Jürgen Jänes, Milot Mirdita, Cameron LM Gilchrist, Tanita Wein, Mihaly Varadi, Sameer Velankar, Pedro Beltrao, and Martin Steinegger. Clustering predicted structures at the scale of the known protein universe. *Nature*, 622(7983):637–645, 2023.

Letícia MF Bertoline, Angélica N Lima, Jose E Krieger, and Samantha K Teixeira. Before and after alphafold2: An overview of protein structure prediction. *Frontiers in bioinformatics*, 3:1120370, 2023.

Zhiyuan Chen, Tianhao Chen, Chenggang Xie, Yang Xue, Xiaonan Zhang, Jingbo Zhou, and Xiaomin Fang. Unifying sequences, structures, and descriptions for any-to-any protein generation with the large multimodal model helixprotx. *arXiv preprint arXiv:2407.09274*, 2024.

UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47 (D1):D506–D515, 2019.

Eli J Draizen, Stella Veretnik, Cameron Mura, and Philip E Bourne. Deep generative models of protein structure uncover distant relationships across a continuous fold space. *Nature Communications*, 15(1):8094, 2024.

Christopher Fifty, Ronald G Junkins, Dennis Duan, Aniketh Iyengar, Jerry W Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. *arXiv preprint arXiv:2410.06424*, 2024.

Zhangyang Gao, Cheng Tan, and Stan Z Li. Foldtoken2: Learning compact, invariant and generative protein structure language. *arXiv preprint arXiv:2407.00050*, 2024a.

Zhangyang Gao, Cheng Tan, and Stan Z Li. Foldtoken3: Fold structures worth 256 words or less. *bioRxiv*, pp. 2024–07, 2024b.

Zhangyang Gao, Cheng Tan, and Stan Z Li. Foldtoken4: Consistent & hierarchical fold language. *bioRxiv*, pp. 2024–08, 2024c.

Zhangyang Gao, Cheng Tan, Jue Wang, Yufei Huang, Lirong Wu, and Stan Z Li. Foldtoken: Learning protein language via vector quantization and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 219–227, 2025.

Benoit Gaujac, Jérémie Donà, Liviu Copoiu, Timothy Atkinson, Thomas Pierrot, and Thomas D Barrett. Learning the language of protein structure. *arXiv preprint arXiv:2405.15840*, 2024.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020.

Cheng-Yen Hsieh, Xinyou Wang, Daiheng Zhang, Dongyu Xue, Fei Ye, Shujian Huang, Zaixiang Zheng, and Quanquan Gu. Elucidating the design space of multimodal protein language models. *arXiv preprint arXiv:2504.11454*, 2025.

Arian Rokkum Jamasb, Alex Morehead, Chaitanya K Joshi, Zuobai Zhang, Kieran Didi, Simon V Mathis, Charles Harris, Jian Tang, Jianlin Cheng, Pietro Lio, et al. Evaluating representation learning on the protein structure universe. In *The Twelfth International Conference on Learning Representations*, 2024.

Jürgen Jänes and Pedro Beltrao. Deep learning for protein structure prediction and design—progress and applications. *Molecular Systems Biology*, 20(3):162–169, 2024.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Henry R Kilgore, Itamar Chinn, Peter G Mikhael, Ilan Mitnikov, Catherine Van Dongen, Guy Zylberberg, Lena Afeyan, Salman F Banani, Susana Wilson-Hawken, Tong Ihn Lee, et al. Protein codes promote selective subcellular compartmentalization. *Science*, 387(6738):1095–1101, 2025.

Hyunbin Kim, Milot Mirdita, and Martin Steinegger. Foldcomp: a library and format for compressing and indexing large protein structure sets. *Bioinformatics*, 39(4):btad153, 2023.

Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Critical assessment of methods of protein structure prediction (casp)—round xiv. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1607–1617, 2021.

Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Critical assessment of methods of protein structure prediction (casp)—round xv. *Proteins: Structure, Function, and Bioinformatics*, 91(12):1539–1549, 2023.

Michèle Leemann, Ander Sagasta, Jerome Eberhardt, Torsten Schwede, Xavier Robin, and Janani Durairaj. Automated benchmarking of combined protein structure and ligand conformation prediction. *Proteins: Structure, Function, and Bioinformatics*, 91(12):1912–1924, 2023.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Jiarui Lu, Xiaoyin Chen, Stephen Zhewen Lu, Chence Shi, Hongyu Guo, Yoshua Bengio, and Jian Tang. Structure language models for protein conformation generation. *arXiv preprint arXiv:2410.18403*, 2024.

Tianyu Lu, Melissa Liu, Yilin Chen, Jinho Kim, and Po-Ssu Huang. Assessing generative model coverage of protein structures with shapes. *bioRxiv*, 2025.

Bin Ma, John Tromp, and Ming Li. Patternhunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Alex Morehead and Jianlin Cheng. Geometry-complete diffusion for 3d molecule generation and optimization. *Communications Chemistry*, 7(1):150, 2024a.

Alex Morehead and Jianlin Cheng. Geometry-complete perceptron networks for 3d molecular graphs. *Bioinformatics*, 40(2):btae087, 2024b.

Mahdi Pourmirzaei, Farzaneh Esmaili, Salhuldin Alqarghuli, Mohammadreza Pourmirzaei, Ye Han, Kai Chen, Mohsen Rezaei, Duolin Wang, and Dong Xu. Prot2token: A unified framework for protein modeling via next-token prediction. *arXiv preprint arXiv:2505.20589*, 2025.

Xavier Robin, Juergen Haas, Rafal Gumienny, Anna Smolinski, Gerardo Tauriello, and Torsten Schwede. Continuous automated model evaluation (cameo)—perspectives on the future of fully automated evaluation of structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1977–1986, 2021.

Kiersten M Ruff, Yoon Hee Choi, Dezerae Cox, Angelique R Ormsby, Yoochan Myung, David B Ascher, Sheena E Radford, Rohit V Pappu, and Danny M Hatters. Sequence grammar underlying the unfolding and phase separation of globular proteins. *Molecular cell*, 82(17):3193–3208, 2022.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. *BioRxiv*, pp. 2023–10, 2023.

Timir Tripathi, Vladimir N Uversky, and Alessandro Giuliani. 'intelligent'proteins. *Cellular and Molecular Life Sciences*, 82(1):239, 2025.

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

Michel van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Foldseek: fast and accurate protein structure search. *Biorxiv*, pp. 2022–02, 2022.

Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 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.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Dplm-2: A multimodal diffusion protein language model. *arXiv preprint arXiv:2410.13782*, 2024.

Konstantin Weissenow and Burkhard Rost. Are protein language models the new universal key? *Current Opinion in Structural Biology*, 91:102997, 2025.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.

Rongqing Yuan, Jing Zhang, Andriy Kryshtafovych, R Dustin Schaeffer, Jian Zhou, Qian Cong, and Nick V Grishin. Casp16 protein monomer structure prediction assessment. *Proteins: Structure, Function, and Bioinformatics*, 2025a.

Xinyu Yuan, Zichen Wang, Marcus Collins, and Huzefa Rangwala. Protein structure tokenization: Benchmarking and new recipe. *arXiv preprint arXiv:2503.00089*, 2025b.

Jiayou Zhang, Barthelemy Meynard-Piganeau, James Gong, Xingyi Cheng, Yingtao Luo, Hugo Ly, Le Song, and Eric Xing. Balancing locality and reconstruction in protein structure tokenizer. In *NeurIPS 2024 Workshop on Machine Learning in Structural Biology (MLSB)*. bioRxiv, 2024. doi: 10.1101/2024.12.02.626366. URL `https://www.biorxiv.org/content/10.1101/2024.12.02.626366v2`.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5745–5753, 2019.

A APPENDIX

A.1 METHODS

Algorithms 1–3 define the decoder head, geometric losses, and alignment used in training. Given per-residue embeddings, the 6D head projects to two direction vectors and a translation, constructs a proper rotation via Gram–Schmidt, composes this rigid update with the running pose, and applies it to a fixed (N–C_α–C) template to produce backbone coordinates. Reconstruction is supervised by the weighted sum in Equation 8: (i) aligned MSE after optimal Kabsch alignment, (ii) a clamped backbone distance-matrix loss, and (iii) a clamped pairwise orientation (direction) loss built from six backbone vectors per residue.

Algorithm 1 Pseudocode for 6D rotation-based structure prediction

```
def dim6rot_structure_head(h: Tensor[N, d],
                           g_initial: Tuple[Tensor[N,3,3], Tensor[N,3,1]],
                           alpha: float = 1.0):
    """
    Predicts backbone coordinates from embeddings using a 6D rotation representation.
    This process is vectorized over N residues.

    h: [N, d] float32         -- input embeddings for N residues
    g_initial: (R_0, t_0)     -- initial SE(3) pose (running transformation)
    alpha: float              -- translation scaling hyper-parameter
    """
    # Define a fixed local reference frame for backbone atoms (N, C_alpha, C)
    X_local = torch.tensor(...) # Shape: [3, 3]

    # 1) Project embeddings to unconstrained translations and direction vectors
    # This represents the internal FFN and projection layers.
    h_ffn = LayerNorm(GELU(Linear(h)))
    # W_rigid projects h_ffn to 9 dimensions for t_tilde, a, and b
    rigid_proj = Linear(h_ffn) # Output shape: [N, 9]
    t_tilde, a, b = torch.split(rigid_proj, 3, dim=-1)

    # 2) Create rotation R_i via Gram-Schmidt orthonormalization
    a_hat = a / torch.norm(a, dim=-1, keepdim=True)
    c = torch.cross(a_hat, b, dim=-1)
    c_hat = c / torch.norm(c, dim=-1, keepdim=True)
    b_hat = torch.cross(c_hat, a_hat, dim=-1)

    # Stack column vectors to form the rotation matrix R
    R = torch.stack([a_hat, b_hat, c_hat], dim=-1) # Shape: [N, 3, 3]

    # 3) Scale the translation vector
    t = t_tilde * alpha # Shape: [N, 3]

    # 4) Form the local rigid body update g = (R, t) and compose it
    #    with the running pose g_initial = (R_0, t_0)
    R_0, t_0 = g_initial
    R_new = R_0 @ R
    t_new = (R_0 @ t.unsqueeze(-1)) + t_0

    # 5) Apply the new pose g_new to the local template to get final coordinates
    # Unsqueeze t_new for broadcasting over the 3 atoms in X_local
    X_pred = (R_new @ X_local.T).transpose(-1, -2) + t_new.transpose(-1, -2)

    return X_pred, (R_new, t_new) # Return coords and the updated pose
```

Algorithm 2 Pseudocode for backbone *distance*, *direction* and *MSE* losses

```
def compute_backbone_vectors(coords: Tensor[L, 3, 3]) -> Tensor[L, 6, 3]:
    """
    coords[..., 0/1/2] = (N, CA, C) atom positions for each residue.
    Returns six normalised vectors per residue:
        1)  N → CA            4)  -n_CA   (binormal of NCA plane)
        2)  CA → C            5)   n_N    (binormal of CN_prev-N plane)
        3)  C  → N_{i+1}      6)   n_C    (binormal of CA C N_{i+1} plane)
    """
    # bond vectors ------------------------------------------------------------
    v1 = coords[:, 1] - coords[:, 0]            # N → CA
    v2 = coords[:, 2] - coords[:, 1]            # CA → C
    v3 = pad_end(coords[1:, 0] - coords[:-1, 2])    # C → N_{i+1}
    v4 = -torch.cross(v1, v2)                   # -n_CA
    v5 = torch.cross(v3, v1)                    #  n_N
    v6 = torch.cross(v2, v3)                    #  n_C

    V = torch.stack([v1, v2, v3, v4, v5, v6], dim=1)
    return F.normalize(V, dim=-1)               # shape = [L, 6, 3]


def backbone_distance_loss(x_pred, x_true, mask):
    """
    x_pred/x_true : [L, 3, 3]  predicted & native backbone coords
    mask          : [L]        boolean, valid residues
    """
    P = x_pred[mask, :3].reshape(-1, 9)         # flatten 3 atoms → 9-D point
    T = x_true[mask, :3].reshape(-1, 9)

    D_pred = pairwise_l2(P, P)                  # (N, N) distance matrix
    D_true = pairwise_l2(T, T)

    diff   = (D_pred - D_true).pow(2).clamp(max=25.)
    return diff.mean()


def backbone_direction_loss(x_pred, x_true, mask):
    """
    Uses six orientation vectors per residue as returned by
    `compute_backbone_vectors`.
    """
    V_pred = compute_backbone_vectors(x_pred[mask, :3])
    V_true = compute_backbone_vectors(x_true[mask, :3])

    # Pairwise dot-product tensors:  S_{ij}^{kl} = v_i^{(k)} · v_j^{(l)}
    S_pred = torch.einsum('ikd,jld->ijkl', V_pred, V_pred)
    S_true = torch.einsum('ikd,jld->ijkl', V_true, V_true)

    diff   = (S_pred - S_true).pow(2).clamp(max=20.)
    return diff.mean()


def aligned_mse_loss(x_pred, x_true, mask):
    """
    Mean-squared error after optimal rigid alignment
    (Kabsch) between predicted and native backbone coords.

    Args
    ----
    x_pred : Tensor[L, 3, 3]   # predicted (N, CA, C)
    x_true : Tensor[L, 3, 3]   # ground truth
    mask   : BoolTensor[L]     # valid residues
    """
    # 1) select masked residues and flatten atoms ------------- #
    P = x_pred[mask].reshape(-1, 3)   # (3N, 3)
    T = x_true[mask].reshape(-1, 3)   # (3N, 3)

    # 2) best-fit rotation / translation via Kabsch ------------ #
    R, t = kabsch(P, T)               # R ∈ SO(3), t ∈ ℝ^3
    T_aln = (T @ R.T) + t             # align native coords

    # 3) mean-squared error ---------------------------------- #
    mse = ((P - T_aln).pow(2)).mean()
    return mse
```

Algorithm 3 Pseudocode for rigid alignment via Kabsch

```
def kabsch_align(A: Tensor[N, 3],
                 B: Tensor[N, 3],
                 allow_reflections : bool = False):
    """
    Optimal rigid alignment (rotation R, translation t) of
    point cloud A onto reference cloud B.

    A, B : [N, 3] float32  -- corresponding coordinates
    """
    # 1) centre the clouds --------------------------------- #
    centroid_A = A.mean(0);  centroid_B = B.mean(0)
    AA = A - centroid_A      # centred source
    BB = B - centroid_B      # centred target

    # 2) SVD of covariance -------------------------------- #
    H = AA.T @ BB                              # 3x3
    U, _, Vh = torch.linalg.svd(H)             # H = U S Vh

    # 3) ensure proper rotation (det = +1) ----------------- #
    if not allow_reflections and torch.det(Vh.T @ U.T) < 0:
        Vh[-1, :] *= -1

    R = Vh.T @ U.T                             # rotation
    t = centroid_B - R @ centroid_A            # translation

    return (R @ A.T).T + t                  # aligned A
```

A.2 Datasets

Test A (low-confidence) evaluates robustness under structural uncertainty using a curated subset of AlphaFold2 (AF2) models for SwissProt proteins. We enumerated all AF2-predicted structures available for SwissProt entries in AFDB and retained only those with average pLDDT ≤ 70 (AFDB's 0–100 scale), discarding higher-confidence models. The surviving records were *cross-referenced* against the PDB to annotate the presence of experimental structures for the same proteins. Multimer entries were decomposed into per-chain monomer samples, after which we removed chains shorter than 25 amino acids and retained only chains with fewer than 25 consecutive missing residues. A final deduplication step produced 33 112 single-chain samples for Test A.

Test B (species-diverse, taxonomic shift) assesses generalization under distributional shift in species rather than structure confidence. We assembled a broad species roster (expanded to 100 and finalized at 97 species) and targeted roughly 2 500 proteins per species from AFDB without pLDDT filtering to reflect natural variability, dropping species with fewer than 1 000 available structures. After aggregation, per-chain conversion, and deduplication, this suite comprised 19 353 single-chain samples.

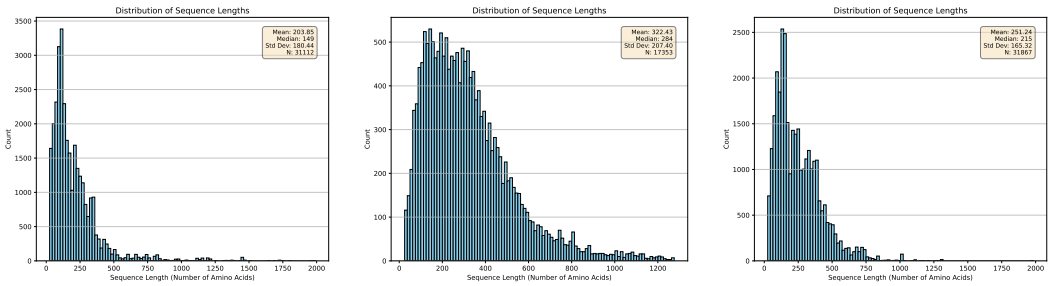

Figure 3: Distribution of sequence lengths in the three test sets: (left) Test set A, (middle) Test set B, and (right) Test set C. The majority of samples have sequence length < 1280 amino acids.

Test C (high-confidence, under-represented taxa) measures performance on high-confidence structures from taxa under-represented in SwissProt. We selected species with low SwissProt coverage and required high average pLDDT (> 95). The selected taxonomies were cross-referenced against

the PDB to find the presence of experimental structures for the proteins from those species. Applying the same preprocessing and deduplication as Test A above yielded 33 867 single-chain samples.

To obtain a balanced validation set independent of training, we first constructed three targeted hold-out suites and then set aside a fixed 2 000 randomly sampled chains from each to form a combined 6 000-chain validation set; the remaining examples constitute Test sets A, B, and C, respectively. The distribution of length sequences in the test sets is displayed in Figure 3.

From the zero-shot set, we defined interpretable structural subsets based on each protein's secondary-structure composition. Coil-rich (D) includes sequences with at least 60% coil/disordered content. Mainly α (A) requires an α-helical fraction of at least 45% and an $\alpha - \beta$ difference of at least 15 percentage points; mainly β (B) requires a β-strand fraction of at least 25% and a $\beta - \alpha$ difference of at least 10 points. Balanced (AB) captures mixed folds with $\alpha \geq 25\%$, $\beta \geq 15\%$, and $|\alpha - \beta| \leq 10$ points. We also report two data-quality subsets: sequences with no missing residues and sequences with at least one missing residue. Groups are defined independently (i.e., they may overlap).

A.3 Experiments

Table 7: Training hyperparameters for Large (Stages 1 & 2) and Lite models.

Hyperparameter	**Large (Stage 1)**	**Large (Stage 2)**	**Lite**
Optimizer type	AdamW	AdamW	AdamW
β_1	0.9	0.9	0.9
β_2	0.98	0.98	0.98
ϵ	10^{-7}	10^{-7}	10^{-7}
Weight decay	10^{-3}	10^{-3}	10^{-2}
Gradient clipping (max L_2 norm)	1.0	1.0	1.0
Gradient accumulation	1	1	1
8-bit parameters	✗	✓	✓
Batch size (per GPU)	16	4	16
Learning-rate strategy	Cosine annealing	Cosine annealing	Cosine annealing
Base learning-rate	$1e-4$	$5e-5$	$1e-4$
Min learning-rate	$1e{-}6$	$1e{-}6$	$1e{-}6$
Warm-up steps	16 000	16 000	4 096
Total steps	375 000	1 500 000	284 000
MSE coefficient (λ_{MSE})	1×10^{-3}	5×10^{-3}	1×10^{-3}
Backbone distance coefficient (λ_{dist})	10^{-2}	10^{-2}	10^{-2}
Backbone direction coefficient (λ_{dir})	5×10^{-2}	5×10^{-2}	5×10^{-2}

This subsection compiles all experimental assets: Table 7 lists the full optimization schedule and training hyperparameters for the Lite and Stages 1–2 of the Large model; Table 8 specifies the exact model configuration (GCPNet, VQ, and Transformer stacks). Figure 4 reports RMSD error distributions on the zero-shot dataset across methods; Figure 5 shows qualitative robustness to contiguous missing-residue segments from benchmark structures; Figure 6 plots sequence length versus reconstruction RMSD with a least-squares fit on the validation set; Figure 7 visualizes the highest-RMSD (worst-case) examples per benchmark suite. Metrics use backbone atoms (C_α) after Kabsch alignment, and statistics are over the full, unfiltered test sets.

We maintain the codebook with the exponential–moving–average (EMA) variant of VQ-VAE: for each code k we keep decayed counts N_k and sums M_k of the encoder vectors assigned at the current step and update the entry by $\mathbf{e}_k \leftarrow M_k/(N_k + \varepsilon)$, with decay γ (Stage 1: 0.99, Stage 2: 0.995; Table 8). Code entries are *not* updated by gradient; only the encoder receives the commitment penalty $\mathcal{L}_{\text{commit}} = \beta\|\mathbf{z} - \text{sg}[\mathbf{e}_k]\|_2^2$. We detect "dead" codes via an EMA count threshold (2 by default) and reinitialize them from recent encoder features using the same k-means seeding as at start-up. For the encoder pathway, we still use the rotation-trick Jacobian (Fifty et al., 2024) to pass informative gradients through the nearest-neighbour assignment; this affects only the encoder and does not modify the EMA update rule.

We set the coefficients λ (Equation 8) by monitoring the per-term gradient norms and equalizing their contribution to shared parameters. This gradient-norm balancing prevents any single term (e.g., direction or distance) from dominating updates and yields stable, faster convergence.

Throughout all experiments, sequences longer than the model's input window are truncated to the first L residues (Stage 1: $L = 512$, Stage 2: $L = 1{,}280$), and all metrics are computed on the truncated region.

Table 8: Configurations of GCP-VQVAE architectures (Large Stages 1 & 2, and Lite).

Hyperparameter	**Large (Stage 1)**	**Large (Stage 2)**	**Lite**
GCPNet Encoder			
Initialization	(Jamasb et al., 2024)	Stage 1	(Jamasb et al., 2024)
Input dim (node scalars h)	6	6	6
Input dim (node vectors χ)	3	3	3
Input dim (edge scalars e)	8	8	8
Input dim (edge vectors ξ)	1	1	1
Hidden dim (node scalars h)	128	128	128
Hidden dim (node vectors χ)	16	16	16
Hidden dim (edge scalars e)	32	32	32
Hidden dim (edge vectors ξ)	4	4	4
Number of layers	6	6	6
SE(3) equivariance	✓	✓	✓
Vector Quantization			
Initialization	Random	Stage 1	Random
Hidden dimension	256	256	128
Vocab size	4 096	4 096	4 096
EMA Decay	0.99	0.995	0.99
Threshold EMA dead code	2	2	2
Commitment weight (λ_{commit})	0.5	0.25	0.25
Orthogonal reg weight (λ_{orth})	10	10	10
Orthogonal reg max codes	512	512	512
Orthogonal reg active codes only	✓	✓	✓
Rotation trick	✓	✓	✓
KMeans initialization	✓	✓	✓
KMeans iteration	10	10	10
Transformer Encoder			
Initialization	Random	Stage 1	Random
Hidden dimension	1024	1024	1024
FF Multiplier	4×	4×	4×
Blocks	12	12	8
Query heads	12	12	8
Key-value heads	3	3	2
Positional encoding	RoPE	RoPE	RoPE
Query-key norm	✓	✓	✓
Pre-norm	✓	✓	✓
Transformer Decoder			
Initialization	Random	Stage 1	Random
Hidden dimension	1024	1024	1024
FF Multiplier	4×	4×	4×
Blocks	16	16	12
Query heads	16	16	8
Key-value heads	1	1	2
Positional encoding	RoPE	RoPE	RoPE
Query-key norm	✓	✓	✓
Pre-norm	✓	✓	✓

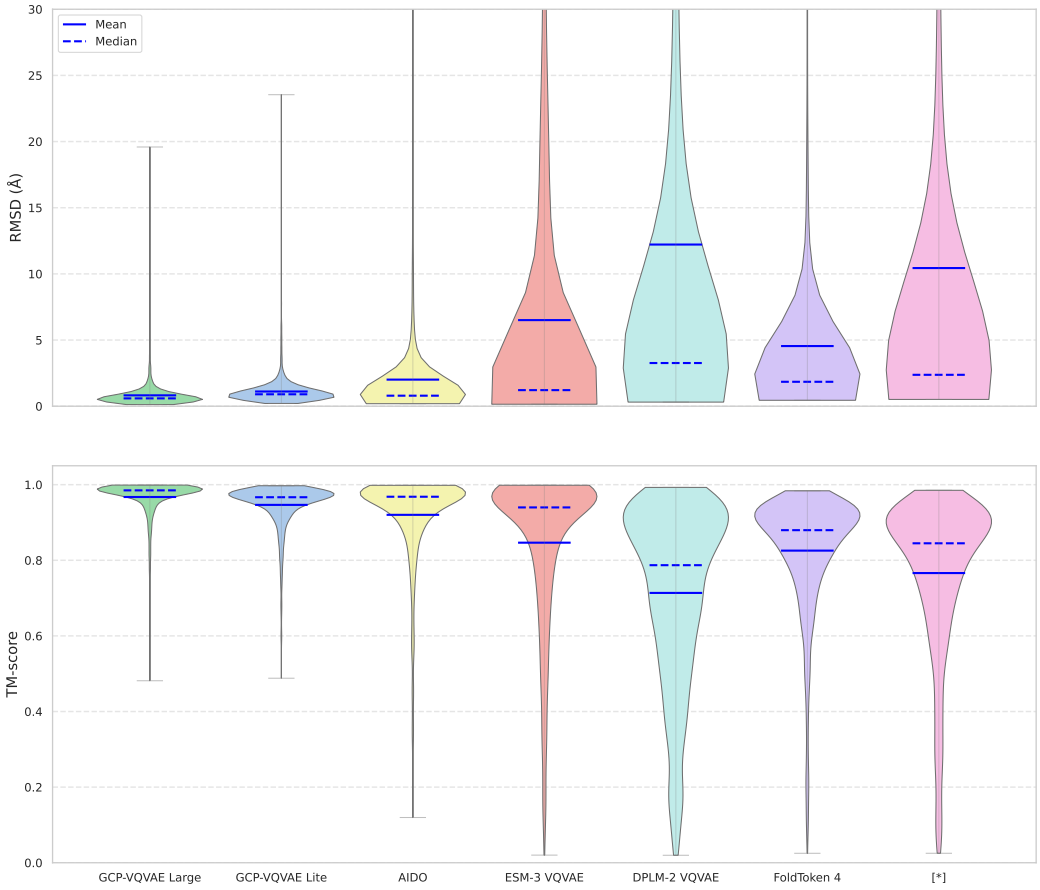

Figure 4: RMSD (Å) and TM-score error distributions on the zero-shot dataset. Gaujac et al. (2024) is shown as [$*$]. For Gaujac et al. (2024), samples outside its supported length range (50–512 residues) are excluded; other methods use the full sets.

Figure 5: Superposition of GCP-VQVAE reconstructions (orange) with native backbones (green) for three proteins drawn from our external benchmark suites. Blue circles mark contiguous missing-residue segments (dashed guides span regions without deposited atoms); the model tracks the native structure outside the gaps.

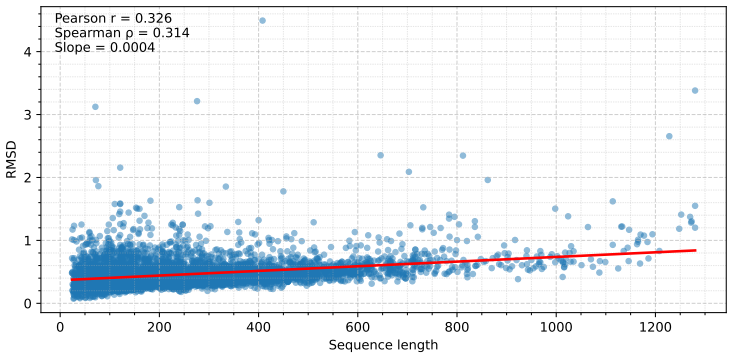

Figure 6: Sequence length vs. reconstruction error (validation set). Each point is one protein; the red line is a least-squares fit. The dependence on length is modest (Pearson $r = 0.326$, Spearman $\rho = 0.314$) with a small slope of $\approx 4 \times 10^{-4}$ Å/residue (~ 0.4 Å per 1k residues).

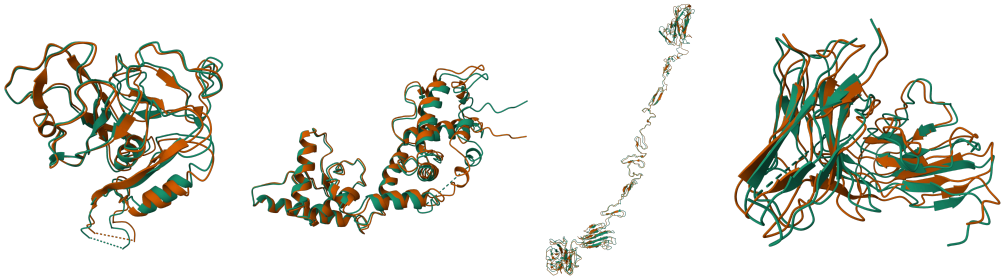

Figure 7: Superposition of GCP-VQVAE (orange) and native backbones (green) for the highest–backbone-RMSD example in each suite, left→right: CASP14, CASP15, CASP16, CAMEO2024.

A.4 Zero-Shot Performance

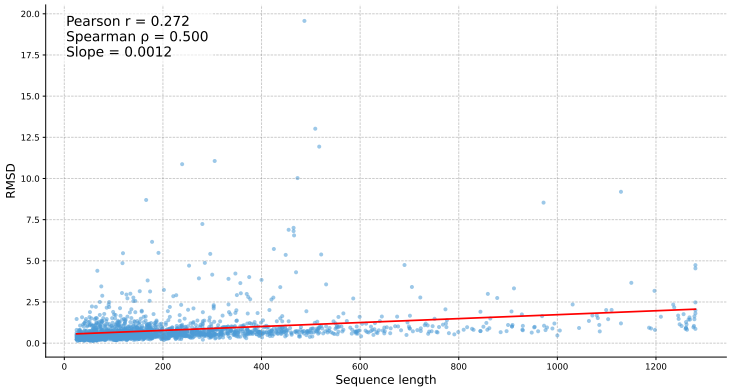

Figure 8: RMSD as a function of sequence length (number of amino acids) on the zero-shot set. Each dot is one protein. The red line is an ordinary-least-squares (OLS) fit; the panel reports Pearson r, Spearman ρ, and the fitted slope (RMSD units per residue).

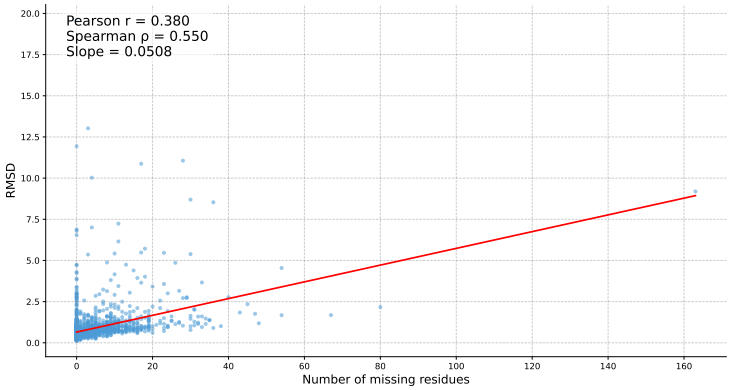

Figure 9: RMSD as a function of the number of missing residues on the zero-shot set. Points show individual proteins; the red line is an OLS fit. The panel reports Pearson r, Spearman ρ, and the fitted slope (RMSD units per missing residue).

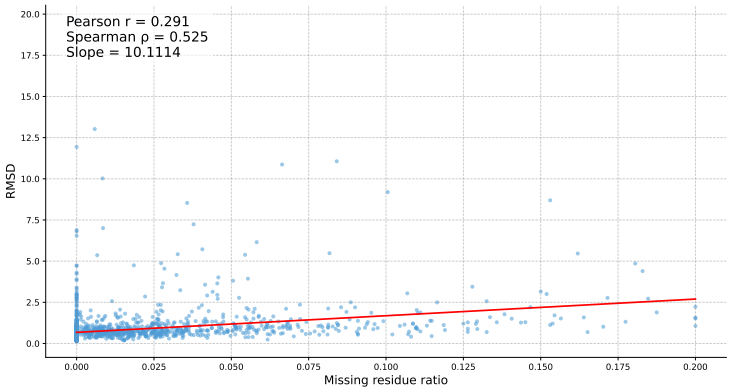

Figure 10: RMSD as a function of the missing-residue ratio on the zero-shot set. The red line is an OLS fit; the panel reports Pearson r, Spearman ρ, and the fitted slope (RMSD units per unit fraction; $1.0{=}100\%$).

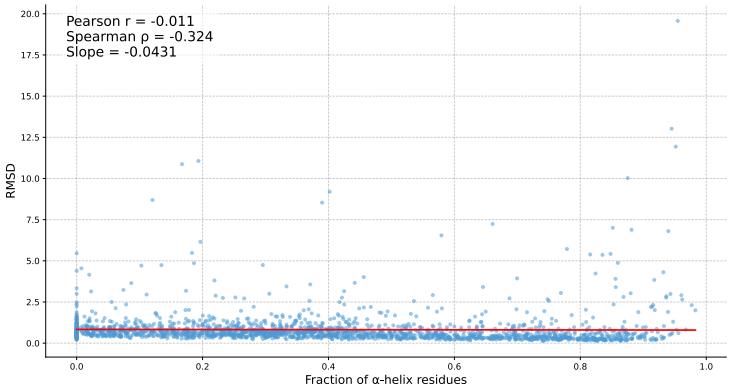

Figure 11: RMSD as a function of the fraction of α-helix residues on the zero-shot set. The red line indicates an OLS fit. The panel reports Pearson r, Spearman ρ, and the fitted slope (RMSD units per unit fraction; $1.0{=}100\%$).

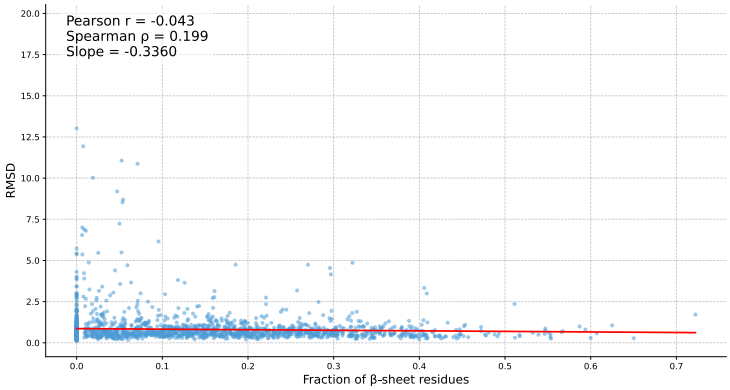

Figure 12: RMSD as a function of the fraction of β-sheet residues on the zero-shot set. The red line indicates an OLS fit. The panel reports Pearson r, Spearman ρ, and the fitted slope (RMSD units per unit fraction; $1.0{=}100\%$).

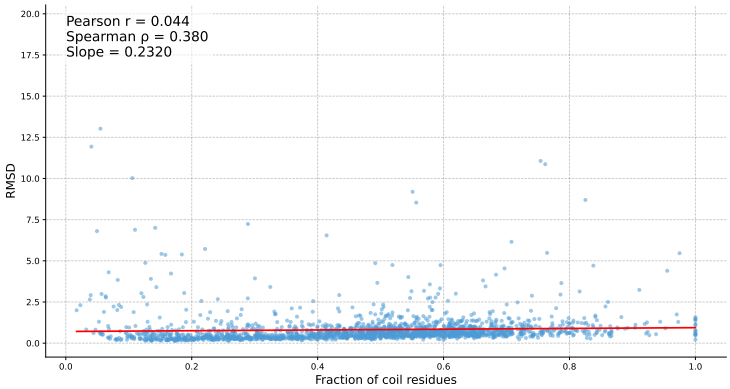

Figure 13: RMSD as a function of the fraction of coil residues on the zero-shot set. The red line indicates an OLS fit. The panel reports Pearson r, Spearman ρ, and the fitted slope (RMSD units per unit fraction; $1.0{=}100\%$).

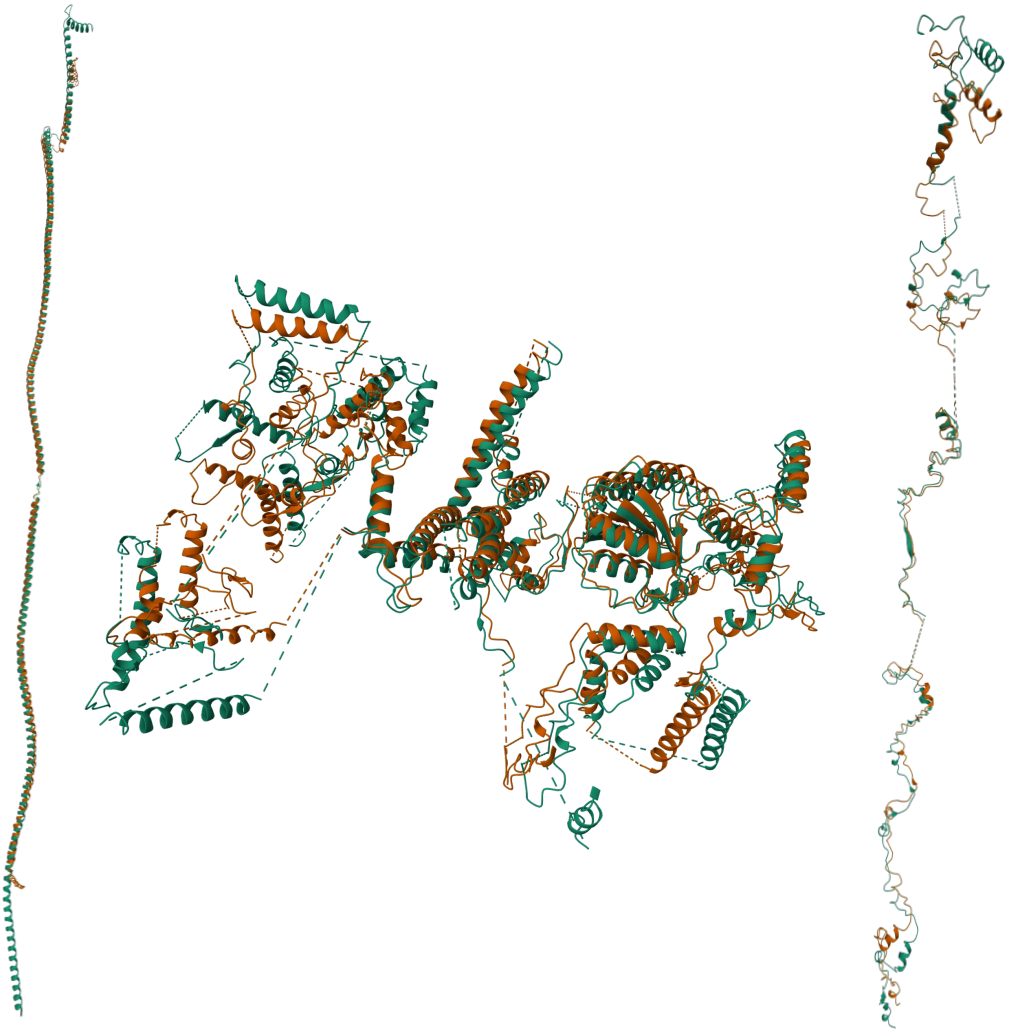

Figure 14: Worst zero-shot reconstructions (orange: GCP-VQVAE Large; green: native). Left: RMSD 19.568, TM-score 0.691, length 487, NaN residues 0. Middle: RMSD 9.1924, TM-score 0.7442, length 1129, NaN residues 163. Right: RMSD 11.0627, TM-score 0.4755, length 305, NaN residues 28. The middle and right cases exhibit an extreme number of NaN residues introduced by our current NaN-handling/augmentation pipeline, which forces the model to contend with long masked segments and degrades reconstruction accuracy; improving the NaN strategy (e.g., capping gap ratio, better segmentation, and length-aware masking) should mitigate this failure mode.

A.5 Representation Evaluation

We note that the physicochemical property prediction tasks (e.g., flexibility and B-factor regression) are omitted from the reported results. Integrating our custom GCP-VQVAE architecture with the official PST benchmark codebase (Yuan et al., 2025b) was not a straightforward task, as the original framework does not natively support external VQ-VAE models without significant modification. Consequently, we were required to re-implement the evaluation pipeline for each specific task. This extensive re-engineering introduced potential uncertainties regarding the alignment between our embeddings and the regression targets. In our preliminary experiments, we observed anomalously low correlations for physicochemical properties that diverged significantly from the trends observed in other structural tasks. We suspect these results may stem from discrepancies in the adapted evaluation harness rather than the representational quality of the model itself. To ensure the scientific

Table 9: Protein Structure Tokenization benchmark across supervised Functional Site Prediction tasks. Higher is better for AUROC%.

		IF-based PST		VQ-VAE-based PST					Ours	
Task	**Split**	**Protein MPNN**	**MIF**	**Fold Seek**	**Pro Tokens**	**ESM3**	**VanillaVQ**	**AminoAseed**	**Our Model (Large)**	**Our Model (Lite)**
BindInt	Fold	51.83	50.38	**53.18**	44.66	44.30	47.25	47.11	51.18	49.03
	SupFam	94.00	**94.56**	46.20	90.77	86.71	85.44	90.53	60.06	59.02
BindBio	Fold	78.42	**85.79**	52.37	58.47	62.84	62.02	65.73	60.36	59.70
	SupFam	81.00	**87.27**	32.01	58.47	62.84	62.02	65.80	59.97	57.85
BindShake	Org	75.52	**79.90**	50.07	59.82	60.12	67.04	69.61	63.11	62.95
CatInt	Fold	61.05	59.62	53.43	58.16	55.39	58.09	**62.19**	52.39	51.22
	SupFam	93.40	**94.60**	46.20	89.82	86.21	91.91	93.03	52.92	52.45
CatBio	Fold	82.49	**85.85**	56.37	56.14	65.33	67.58	65.95	42.45	38.92
	SupFam	93.19	**96.97**	53.78	64.05	74.63	70.92	87.59	54.84	44.61
Con	Fold	57.18	65.43	49.26	70.23	74.56	56.98	**85.43**	53.67	48.09
	SupFam	84.68	**92.66**	51.39	74.19	79.93	74.09	86.00	55.52	45.08
Rep	Fold	77.63	74.53	50.37	63.10	74.23	72.16	**84.57**	51.34	54.05
	SupFam	80.71	83.11	52.53	78.03	82.36	78.99	**84.57**	55	57.42
Ept	Fold	62.84	**68.78**	54.52	54.69	63.69	63.69	62.16	52.92	63.67
	SupFam	64.84	**82.98**	50.56	61.59	69.82	59.64	72.06	61.44	55.76
Average AUROC%		75.92	**79.82**	51.90	65.37	69.24	68.30	72.43	55.15	55.32

rigor and reliability of our benchmarks, we have opted to withhold these specific metrics pending a complete verification of the re-implemented evaluation code.

A.6 Codebook Evaluation

To investigate the relationship between learned VQ codebook tokens and protein backbone geometry, we performed a Ramachandran plot analysis on a zero-shot test set comprising 1,938 protein structures not seen during training. For each protein, we passed the original backbone coordinates through the GCPNet encoder and vector quantizer to obtain per-residue VQ code assignments, then decoded these quantized representations to reconstruct the backbone coordinates. We computed backbone dihedral angles ϕ and ψ for both original and reconstructed structures using standard geometric formulas based on the N, Cα, and C atomic positions of consecutive residues.

Each residue was plotted in Ramachandran space and colored according to its assigned VQ code index. For visualization clarity, we highlighted the top-30 most frequent codes with distinct colors while displaying remaining codes in gray. We included reference boxes indicating canonical secondary structure regions: the α-helix region ($\phi \in [-90, -30]$, $\psi \in [-70, -15]$) and the β-sheet region ($\phi \in [-160, -80]$, $\psi \in [90, 180]$). These boundaries represent simplified rectangular approximations of sterically allowed regions based on standard Ramachandran conventions.

We analyzed both the Lite and Large model variants across a total of 447,784 residue-level data points. The comparison between original and reconstructed structure distributions allows us to assess whether the quantization bottleneck preserves local backbone geometry, while the comparison between model variants reveals differences in codebook utilization and conformational specificity.

To quantitatively assess the information content and utilization of the learned VQ codebook, we computed the following information-theoretic metrics on the discrete token sequences $Z = (z_1, z_2, \ldots, z_L)$ generated by the encoder:

Unigram Statistics. We measure codebook utilization using unigram entropy $H_1 = -\sum p(z) \log_2 p(z)$ and perplexity PPL $= 2^{H_1}$. Perplexity represents the effective vocabulary size; an ideal uniform codebook would have PPL $= |K| = 4096$ ($H_1 = 12$ bits). We report *Effective Usage* as PPL$/|K|$.

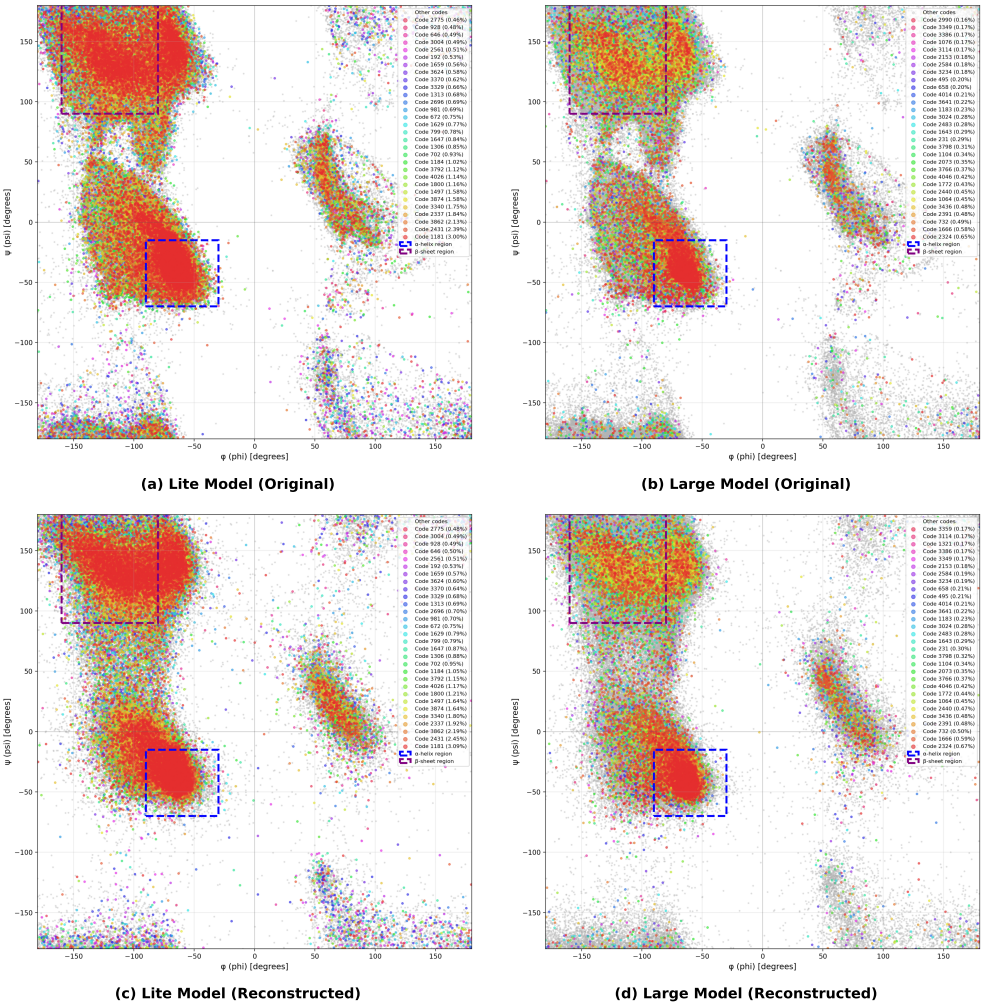

Figure 15: Ramachandran plot analysis of VQ codes on a zero-shot test set, colored by the top-30 most frequent codes. Dashed boxes indicate α-helix (blue) and β-sheet (purple) regions. **(a-b)** Lite model. **(c-d)** Large model. Left columns show original structures; right columns show reconstructions.

Table 10: Top-10 most frequent codebook tokens and their usage percentages on the zero-shot test set. We omit the AminoAseed report due to poor reconstruction results.

Rank	**GCP-VQVAE Lite**		**GCP-VQVAE Large**		**ESM3**		**AIDO**		**DPLM-2**		**FoldToken**		(Gaujac et al., 2024)	
	Code	%	Code	%	Code	%	Code	%	Code	%	Code	%	Code	%
1	1181	2.96	2324	0.65	588	0.88	490	0.36	2234	0.31	182	1.88	2541	0.44
2	2431	2.37	1666	0.58	264	0.67	431	0.33	2235	0.30	211	1.68	2542	0.38
3	3862	2.10	732	0.49	2056	0.58	284	0.33	2362	0.28	51	1.47	815	0.31
4	2337	1.83	2391	0.48	3961	0.52	435	0.31	2490	0.24	105	1.38	2556	0.25
5	3340	1.73	3436	0.47	1476	0.49	347	0.30	2363	0.24	112	1.33	799	0.21
6	3874	1.57	1064	0.45	1197	0.49	272	0.30	2107	0.21	11	1.29	4048	0.20
7	1497	1.57	2440	0.44	3101	0.48	113	0.30	2491	0.20	230	1.23	3808	0.19
8	1800	1.16	1772	0.43	987	0.48	334	0.30	2299	0.20	251	1.22	3792	0.19
9	4026	1.13	4046	0.42	3954	0.47	132	0.30	3258	0.19	195	1.14	2023	0.19
10	3792	1.11	3766	0.36	2082	0.45	404	0.30	2106	0.19	38	1.08	3776	0.19
Top-10 Total	—	**17.53**	—	**4.77**	—	**5.51**	—	**3.13**	—	**2.36**	—	**13.70**	—	**2.55**

Zipf Distribution Fit. Natural languages typically follow Zipf's law, where the frequency of the r-th most common token scales as $f(r) \propto r^{-b}$. We fit a linear regression to the log-log rank-frequency plot and report the slope b and coefficient of determination R^2. A slope near -1 with high R^2 indicates natural language-like power-law scaling.

Conditional Statistics. To evaluate sequential structure, we compute conditional entropies $H(Z_t|Z_{t-1})$ (bigram) and $H(Z_t|Z_{t-2}, Z_{t-1})$ (trigram). Lower conditional entropy indicates higher predictability given context. We report the *Trigram Perplexity* (Tri PPL $= 2^{H(Z_t|Z_{t-2},Z_{t-1})}$), which quantifies the effective branching factor of the sequence given a two-token history.

Mutual Information. We measure long-range dependencies using mutual information $I(Z_t; Z_{t+\Delta}) = H(Z_t) - H(Z_t|Z_{t+\Delta})$ at various lags $\Delta \in \{1, 4\}$. Higher mutual information indicates that tokens carry significant predictive information about future states in the sequence.

A.7 Major Issues in Performance Analysis

During our evaluation pipeline development, we identified a critical issue affecting TM-score calculations for experimental PDB structures. Unlike computationally predicted structures that typically use sequential residue numbering starting from 1, experimental PDB files from sources such as CASP and CAMEO often contain non-standard residue indices reflecting the original crystallographic or NMR data (e.g., residues numbered 23–97 instead of 1–75). Additionally, experimental structures frequently contain missing residues due to unresolved electron density or flexible loop regions, creating gaps in the residue numbering sequence (e.g., residues 1–45, 52–120) that further complicate alignment. When VQ-VAE reconstruction scripts saved decoded structures with default 1-indexed sequential residue numbering while comparing against original PDBs with their native non-sequential numbering, the TM-align algorithm failed to correctly pair corresponding residues, resulting in artificially low TM-scores despite near-identical atomic coordinates. For instance, a structure with actual Cα-RMSD of 0.30 Å was incorrectly reported as having TM-score of 0.10 due to this mismatch. We corrected this by preserving the original residue indices when writing reconstructed PDB files and applying Kabsch alignment before evaluation, ensuring consistent residue numbering between original and reconstructed structures for accurate metric computation.

A.8 Compression calculation

Each residue is encoded by one code from a 4 096-entry book, i.e., $\log_2(4096) = 12$ bits $= 1.5$ bytes/residue. For $L = 512$ residues, the token stream is $512 \times 1.5 = 768$ bytes. Raw backbone coordinates (N, C_α, C) stored as 32-bit floats require 3 atoms$\times$3 coords$\times$4 bytes $= 36$ bytes/residue, i.e., $36 \times 512 = 18{,}432$ bytes. If we include a tiny global pose header (e.g., rotation+translation) of ≈ 36 bytes, the coded footprint is $768{+}36 = 804$ bytes. Thus the compression ratio is $18\,432/804 \approx 22.9\times$ (approximately $24\times$ if the pose header is omitted). This estimate concerns backbone only; metadata and format containerization add negligible overhead relative to the raw-float baseline.

A.9 LLM Usage

In this manuscript, we used large language models only for copy-editing: improving grammar, clarity, and style of author-written text.