Protein Structure Tokenization: Benchmarking and New Recipe

Xinyu Yuan, Zichen Wang, Marcus Collins, Huzefa Rangwala

Proceedings of the 42nd International
Conference on Machine Learning (ICML), 2025

https://arxiv.org/abs/2503.00089

Burak Suyunu

LifeLU Reading Group

21.08.2025

Outline

Presentation Roadmap

1. **Introduction & Motivation:** Why is Protein Structure Tokenization (PST) a critical problem?
2. **Background & Positioning:** A brief overview of existing PST approaches.
3. **Methodology Part I: StructTokenBench:** A deep dive into the new evaluation framework.
4. **Methodology Part II: AminoAseed:** The proposed model for improved tokenization.
5. **Results & Analysis:** Benchmarking, ablations, and scaling studies.
6. **Discussion & Conclusion:** Key takeaways and future directions.

The Central Problem: Beyond the Sequence

From Sequence to Structure to Function:

- Proteins are the machinery of life, defined by their 3D structure.
- Protein Language Models (PLMs) have excelled by treating amino acid sequences as a language.
- **However, sequence alone is insufficient.** Structural context—the 3D arrangement of residues—is what dictates protein function and interaction.

The Need for a Structural "Vocabulary":

- How can we represent complex, continuous 3D structures in a discrete, tokenized format suitable for modern ML architectures (e.g., Transformers, Multimodal Models)?
- This is the goal of **Protein Structure Tokenization (PST)**: to create discrete or continuous representations of local 3D structural motifs.

The Research Gap: A Lack of Standardized Evaluation

A Surge in PST Methods: Recent years have seen many new PST methods emerge, but their capabilities are poorly understood.

The Core Problem: There is no unified framework to benchmark these tokenizers.

- Existing benchmarks focus on **global** structure prediction, not the quality of **local** substructure representations.
- How do we quantitatively assess if a set of structural tokens is "good"?

This Work's Contributions:

1. **StructTokenBench:** A comprehensive framework to evaluate PSTs on fine-grained local structures.
2. **AminoAseed:** A new PST method developed from insights gained during benchmarking, which significantly improves codebook utilization and downstream performance.

A Taxonomy of PST Methods

1. Heuristic Methods: Rely on domain knowledge and hand-crafted rules.

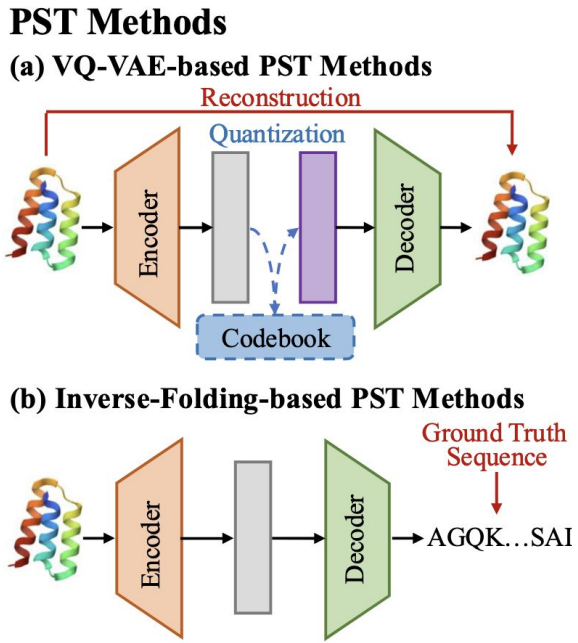
- ***Examples:*** Dihedral angles (Protein Blocks), secondary structure elements.
- ***Pros:*** Interpretable. ***Cons:*** Limited expressiveness, may miss novel motifs.

A Taxonomy of PST Methods

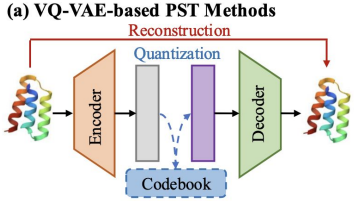
2. Deep Learning Methods: Learn features directly from structural data.

- **VQ-VAE-based (Vector Quantized Variational Autoencoder):** The dominant approach. An autoencoder compresses structure into a discrete latent code from a learned "codebook". Objective is to reconstruct the input 3D structure.
  - ***Examples:*** FoldSeek, ProTokens, ESM3.

- **Inverse-Folding (IF)-based:** Predict the amino acid sequence that folds into the input structure. The continuous latent representation *is* the token.
  - ***Examples:*** ProteinMPNN, MIF.



The VQ-VAE Formalism



$$\boldsymbol{x} \xrightarrow{\text{Structure Encoder}} \boldsymbol{z} \xrightarrow{\text{Quantization}} \boldsymbol{q}_k \xrightarrow{\text{Structure Decoder}} \tilde{\boldsymbol{x}}$$

Core Components:

1. **Structure Encoder E**: Maps input coordinates $x \in \mathbb{R}^{L \times N_{atoms} \times 3}$ to a continuous representation $z = E(x) \in \mathbb{R}^{L \times D}$
2. **Quantization Layer q**: For each residue's vector zi, find the nearest neighbor in a learnable codebook $Q \in \mathbb{R}^{K \times D}$. The token is the index ki.
3. **Structure Decoder D**: Reconstructs the 3D coordinates $\tilde{x}$ from the sequence of quantized vectors $\{q_{k_i}\}$

The Optimization Objective L:

$$\mathcal{L} = \underbrace{\log p(\tilde{x}|q_k)}_{\text{Reconstruction Loss}} + \underbrace{\|\|\text{sg}(z) - q_k\|\|_2^2}_{\text{Quantization Loss}} + \underbrace{\beta\|\|z - \text{sg}(q_k)\|\|_2^2}_{\text{Commitment Loss}}$$

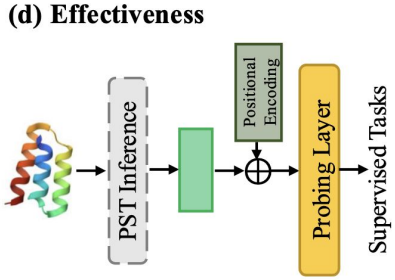
- **Reconstruction Loss:** Ensures the tokens capture enough information to rebuild the structure.
- **Quantization Loss:** Updates the codebook vectors to be closer to the encoder outputs (sg is stop-gradient).
- **Commitment Loss:** Encourages the encoder to output representations close to the chosen codebook vectors.

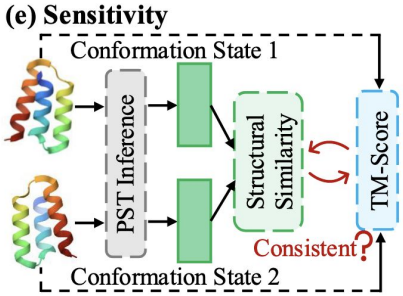
Introducing StructTokenBench: The Four Pillars of Evaluation

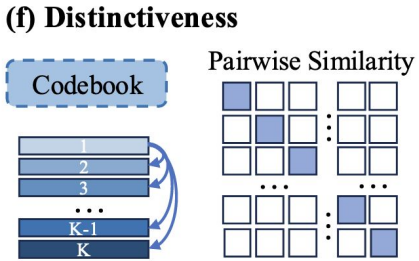
A comprehensive framework to assess the quality of fine-grained, local protein structure representations.

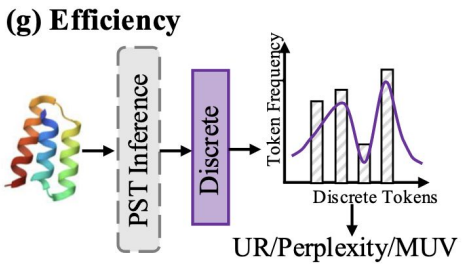
Four Perspectives:

1. **Downstream Effectiveness:** How useful are the tokens for predicting biological properties?
2. **Sensitivity:** Can the tokens distinguish between highly similar but functionally distinct conformations?
3. **Distinctiveness:** How diverse and non-redundant is the learned codebook vocabulary?
4. **Codebook Utilization Efficiency:** How efficiently is the codebook capacity used? Are there "dead" tokens?

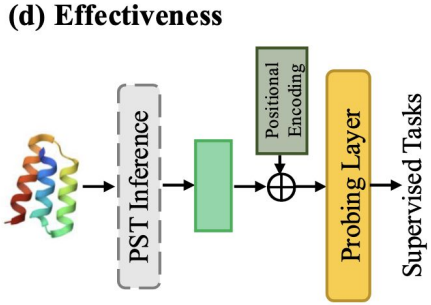








Pillar 1: Downstream Effectiveness



Goal: Evaluate if tokens capture biologically meaningful information.

Setup:

- Freeze the pre-trained PST encoder.
- Extract structural representations (tokens).
- Train a simple, shallow probing layer (2-layer MLP) on top for various supervised tasks.

Tasks (12 tasks, 24 test splits):

- **Functional Site Prediction (Classification):** Binding sites, Catalytic sites, Conserved sites.
- **Physicochemical Property Prediction (Regression):** Structural Flexibility (RMSF, B-Factor).
- **Structural Property Prediction (Classification):** Remote Homology Detection.

Data & Splitting (Crucial for Generalization):

- Datasets from InterPro, BioLIP2, ATLAS, TAPE, etc..
- Uses a **remote homologous splitting** method (from Appendix A.2) based on CATH classes (Fold, Superfamily) to test out-of-distribution generalization, a much harder task than standard sequence-based splits.

Detailed Remote Homologous Data Splitting

Goal: Test out-of-distribution generalization by removing structural similarity between train/test sets.

Hierarchy: Protein CATH classification: **Family** ⊂ **Superfamily** ⊂ **Fold**.

Method:

1. Assign CATH labels to all proteins.
2. **Fold Split (Hardest):** For each Fold, 60% of its *Superfamilies* go to the training pool, 40% to the test pool. No superfamilies are shared.
3. **Superfamily Split (Easier):** Within the training pool's superfamilies, 80% of *proteins* are used for training and 20% for testing. Superfamilies are shared, but proteins are not.

Impact: This is much more challenging than random or sequence-identity-based splits and better evaluates a model's ability to learn fundamental structure-function relationships.

Pillars 2, 3, and 4: Sensitivity, Distinctiveness & Efficiency

Pillar 2: Sensitivity

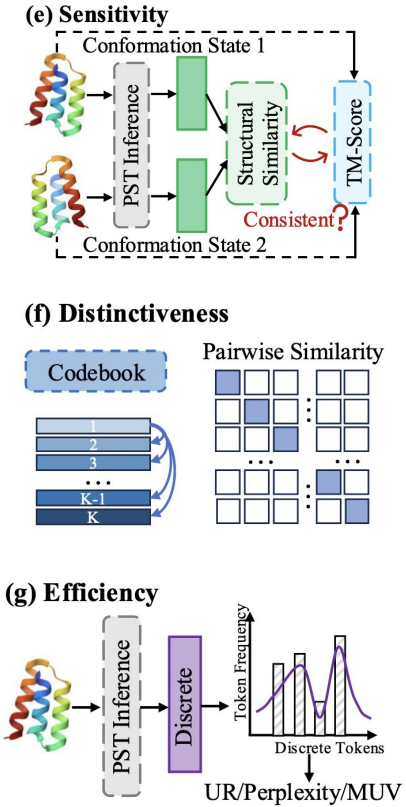
- **Question:** Can tokens detect subtle conformational changes?
- **Method:** Measure the correlation (PCC, Spearman's ρ) between structural representation similarity and true 3D structure similarity (TM-score).
- **Datasets:** *Fold Switching* and *Apo-Holo* protein pairs, which have high sequence identity but different conformations.

Pillar 3: Distinctiveness

- **Question:** Is the codebook diverse or redundant?
- **Method:** Analyze the distribution of pairwise cosine similarities between all codebook vectors. A distribution skewed towards 1 indicates redundancy.

Pillar 4: Codebook Utilization Efficiency

- **Question:** How many of the K possible tokens are actually used?
- **Method:** Tokenize a large set of unseen proteins (CASP14, CAMEO) and measure:
  - **Utilization Rate (UR):** Percentage of active tokens.
  - **Perplexity:** Uniformity of the token usage distribution.
  - **MUV:** Marginal utility of adding more vocabulary.



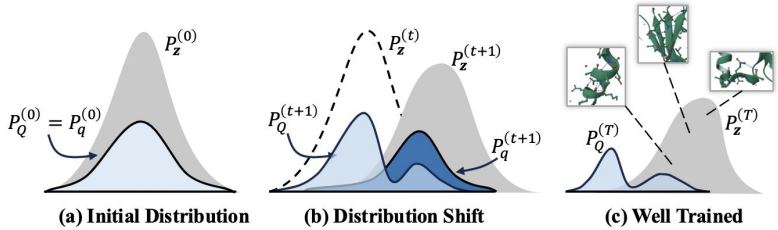
Motivation for AminoAseed: The "Codebook Collapse" Problem

Observation from Benchmarking: Leading PSTs like ESM3 suffer from severe codebook underutilization. Up to 70% of 4096 codes remain inactive.

This is "Codebook Collapse": A well-known issue in VQ-VAEs where the encoder only maps inputs to a small subset of codebook tokens.

The Root Cause:

- Gradients only flow to the *selected* codebook vectors during training.
- Unused vectors are never updated.
- The distribution of encoder outputs (P_z) shifts away from the initial distribution of the full codebook (P_Q), leading to a vicious cycle where fewer and fewer codes are ever selected.



AminoAseed: A Two-Part Recipe for Better Tokenization

Goal: Mitigate codebook collapse and improve tokenizer quality and efficiency.

Technique 1: Codebook Reparameterization

- **Problem:** In vanilla VQ-VAE, unused code vectors receive no gradient updates.
- **Solution:** Instead of learning the codebook Q directly, represent it as a linear transformation of a *fixed*, orthogonal basis C.

$$Q = \text{Linear}(C)$$

- **Effect:** Now, whenever *any* code is used, the gradient flows back through the selection process and updates the *entire* linear layer. This allows all parts of the codebook space to learn, even if a specific vector isn't selected.

AminoAseed: A Two-Part Recipe for Better Tokenization

Technique 2: Pareto-Optimal Codebook Configuration

- **Question:** What is the optimal trade-off between codebook size (K) and dimension (D) for a fixed capacity (K×D)?
- **Data-Driven Approach:** Scaling experiments (which we'll see in the results) revealed:
  - Downstream effectiveness degrades with very large or very small K.
  - Utilization efficiency monotonically drops as K increases.
- **Solution:** Choose a moderate size ($K = 2^9 = 512$) that optimizes the trade-off, balancing expressive power and utilization.

AminoAseed Architecture and Training

Foundation: Builds on ESM3's local frame paradigm.

Input: Per-residue local coordinate frames ($T_i \in SE(3)$) and their neighbors.

Encoder: Stack of **Geometric Self-Attention** layers to maintain rotation/translation invariance.

Quantization: Applies the Codebook Reparameterization and Pareto-Optimal configuration (K=512, D=1024).

Decoder: Standard bidirectional Transformer blocks.

Training Objective:

- Integrates the standard VQ-VAE commitment and quantization losses.
- Uses five reconstruction loss terms from ESM3 for accurate and stable training (geometric distances, binned distances, auxiliary inverse folding loss).

Pre-training: Trained on a 10% subset of the PDB, significantly smaller than many baselines.

Geometric Self-Attention Layer

Input: A set of local frames for each residue and its neighbors. Local Frame Construction:

Each residue i has frame $T_i \in SE(3)$:

$$T_i = \begin{pmatrix} R_i & t_i \\ 0_{1\times 3} & 1 \end{pmatrix}$$

Where:

- $R_i \in SO(3)$: Rotation matrix from Gram-Schmidt on $N - C_\alpha - C$ plane
- $t_i \in \mathbb{R}^3$: Translation vector (position of C_α)

Mechanism:

1. Transform local frame features into a global, invariant representation.
2. Use a specialized attention mechanism that assesses both rotational and distance similarities between pairs of residues.
3. Compute attention scores and aggregate values.
4. Convert the aggregated features back to the local frame representation.

Purpose: This allows the model to reason about the spatial relationships and orientations of amino acids in a way that is invariant to the protein's overall position and orientation in space, which is a fundamental requirement for protein structure analysis.

Results: Downstream Effectiveness

Goal: Evaluate if tokens capture biologically meaningful information.

AminoAseed outperforms the leading VQ-VAE model, ESM3.

- **+6.31%** average relative improvement across all 24 supervised task splits.
- Significant gains in functional (+4.74%) and structural (+27.31%) tasks.

Comparison with IF-based Models:

- **IF-based models still excel**, especially on functional tasks.
- The authors hypothesize this is due to avoiding the **difficult optimization of VQ layers**, **not because continuous tokens are inherently superior**.
- AminoAseed substantially narrows this performance gap.

VanillaVQ Ablation:

- The `VanillaVQ` model (AminoAseed without the proposed techniques) performs similarly to or worse than ESM3, confirming the effectiveness of the new recipe.

Task	Split	Model						
		IF-based PST		VQ-VAE-based PST				
		ProteinMPNN	MIF	FoldSeek	ProTokens	ESM3	VanillaVQ (*v.s.*ESM3)	AminoAseed (*v.s.*ESM3)
Functional Site Prediction (AUROC%)								
BindInt	Fold	51.83	50.38	**53.18**	44.66	44.30	47.25 (+6.66%)	47.11 (+6.34%)
	SupFam	94.00	94.56	46.20	86.05	**90.77**	86.71 (−4.47%)	90.53 (−0.26%)
BindBio	Fold	78.42	85.79	52.37	58.47	62.84	62.02 (−1.30%)	**65.73** (+4.60%)
	SupFam	81.00	87.27	52.41	60.47	65.22	62.92 (−3.53%)	**68.30** (+4.72%)
BindShake	Org	75.52	79.90	53.40	59.82	66.10	67.04 (1.42%)	69.61 (+5.31%)
CatInt	Fold	61.05	59.62	53.43	58.16	61.09	58.89 (−3.60%)	**62.19** (+1.80%)
	SupFam	93.40	96.49	51.41	83.85	89.82	85.00 (−5.37%)	**91.91** (+2.33%)
CatBio	Fold	82.49	85.85	56.37	56.14	65.33	**67.58** (+3.44%)	65.95 (+0.95%)
	SupFam	93.19	96.97	53.78	64.05	74.65	70.92 (−5.00%)	**87.59** (+17.33%)
Con	Fold	57.18	58.43	49.26	56.23	55.22	56.98 (+3.19%)	**57.23** (+3.64%)
	SupFam	84.68	92.66	51.39	74.33	80.53	74.60 (−7.36%)	**86.60** (+7.54%)
Rep	Fold	77.63	74.53	47.70	**77.25**	74.70	75.99 (+1.73%)	74.97 (+0.36%)
	SupFam	80.71	83.11	52.53	78.90	82.36	82.09 (−0.33%)	**84.57** (+2.68%)
Ept	Fold	62.84	68.78	54.52	54.69	**63.69**	59.28 (−6.92%)	62.16 (−2.40%)
	SupFam	64.84	82.98	50.56	67.52	61.97	67.24 (+8.50%)	**72.02** (16.22%)
Average AUROC%		75.92	79.82	51.90	65.37	69.24	68.30 (−0.86%)	**72.43** (+4.74%)
Physicochemical Property Prediction (Spearman's ρ%)								
FlexRMSF	Fold	62.37	59.60	15.35	13.81	44.53	44.22 (−0.70%)	**44.63** (+0.22%)
	SupFam	59.24	56.80	11.99	7.62	39.68	39.08 (−1.51%)	**40.99** (+3.30%)
FlexBFactor	Fold	31.88	34.60	4.17	6.67	**23.60**	22.32 (−5.78%)	21.30 (−10.09%)
	SupFam	34.56	35.23	6.97	5.47	**25.80**	23.73 (−7.95%)	21.76 (−15.59%)
FlexNEQ	Fold	69.69	65.32	5.71	12.98	45.08	35.95 (−20.25%)	**49.64** (+10.12%)
	SupFam	68.69	64.82	2.60	12.50	45.43	35.61 (−21.62%)	**50.15** (+10.41%)
Average ρ%		54.41	52.73	7.80	9.84	37.35	33.49 (−9.64%)	**38.08** (−0.27%)
Structure Property Prediction (Macro F1%)								
Homo	Fold	25.66	22.56	11.57	5.84	**30.02**	18.17 (−39.47%)	29.87 (−0.50%)
	SupFam	30.83	33.86	4.67	6.17	24.89	22.10 (−11.21%)	**38.38** (+54.20%)
	Fam	63.33	74.22	15.30	18.33	54.42	47.18 (−13.30%)	**69.78** (+28.22%)
Average Macro F1%		39.94	43.55	10.51	10.11	36.44	29.15 (−21.33%)	**46.01** (+27.31%)

Results: Sensitivity

- **Question:** Can tokens detect subtle conformational changes?
- AminoAseed is the most sensitive model, outperforming ESM3 by **+12.83%** on average in detecting conformational changes.
- **IF-based models perform poorly**, as their objective encourages predicting the *same* sequence for different conformations of the same protein, reducing sensitivity.

Table 3: Sensitivity evaluation on conformational proteins. The relative performance difference *v.s.* ESM3 is included.

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	61.20	65.90
ESM3	39.76	50.97	57.12	62.23
VanillaVQ$_{(v.s.\text{ESM3})}$	39.62$_{(-0.35\%)}$	50.61$_{(-0.71\%)}$	56.51$_{(-1.07\%)}$	61.51$_{(-1.16\%)}$
AminoAseed $_{(v.s.\text{ESM3})}$	42.47$_{(+6.82\%)}$	**54.88**$_{(+7.67\%)}$	**65.61**$_{(+14.86\%)}$	**75.89**$_{(+21.95\%)}$

Results: Distinctiveness

Question: Is the codebook diverse or redundant?

AminoAseed's codebook shows high distinctiveness, with few highly similar vector pairs, both in theory and in practice (weighted by usage).

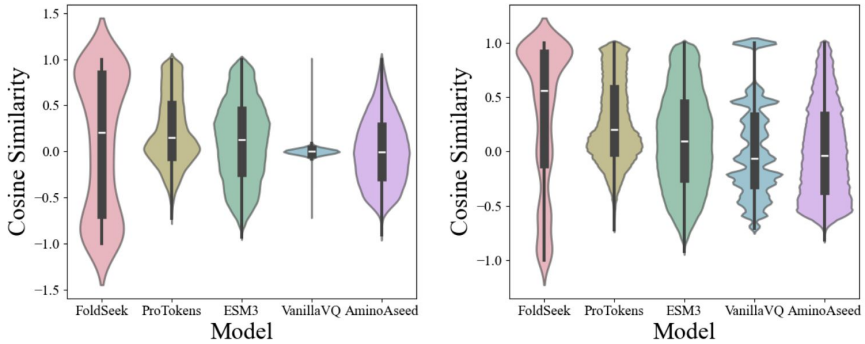


Figure 3: Distinctiveness analysis of PST codebook vectors: Left panel shows pairwise similarities between vectors; Right panel shows frequency-weighted similarities based on CASP14 token usage.

Results: Efficiency

- **Question:** How many of the K possible tokens are actually used?
- **This is the key result.** AminoAseed achieves a **124.03%** relative improvement in utilization rate over ESM3, despite using a smaller pre-training dataset.
- It successfully mitigates codebook collapse, activating a much larger portion of its vocabulary.

Table 4: Codebook utilization efficiency evaluation on CASP14 and CAMEO datasets.

Model	#Code (K)	Dim (D)	CASP14			CAMEO		
			UR%	Perplexity	MUV	UR%	Perplexity	MUV
FoldSeek	20	2	**99.00**	**0.7548**	/	**100.00**	**0.7435**	/
ProTokens	512	32	69.88	0.5369	2.53e-4	75.56	0.5697	2.51e-4
ESM3	4096	128	27.60	0.2489	3.28e-5	32.10	0.2841	3.26e-5
VanillaVQ	512	1024	5.55	0.0339	**2.64e-4**	5.60	0.0337	**2.62e-4**
AminoAseed	512	1024	64.45	0.4946	2.54e-4	68.87	0.5119	2.52e-4

Ablation Study: Key Insights

1. Discrete vs. Continuous Representations:

- ***Is the performance gap between VQ-VAE and IF-based models due to discretization?***
- **Finding: No.** Using the continuous encoder outputs from VQ-VAE models before quantization does not consistently improve performance. The difference is minor.
- **Conclusion:** The performance gap is likely due to **optimization challenges** in VQ, not information loss from discretization.

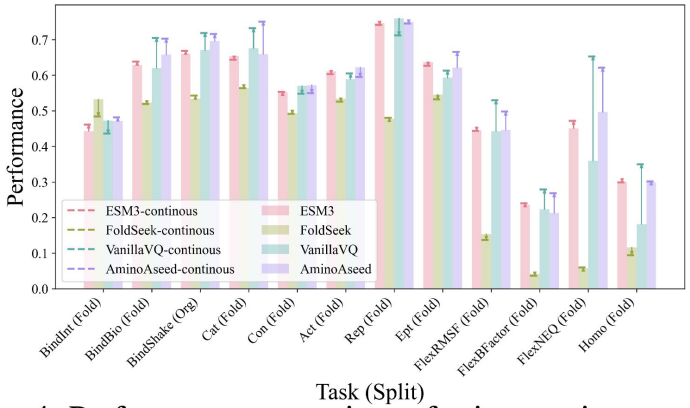


Figure 4: Performance comparison of using continuous versus discrete structural representations on supervised task splits.

Ablation Study: Key Insights

2. Information Content:

- ***Do structural tokens capture sequence information?***
- **Finding:** Combining amino acid embeddings with structural tokens gives only small gains for top models like AminoAseed and ESM3.
- **Conclusion:** High-quality structural tokens already encapsulate most of the information present in sequence tokens.

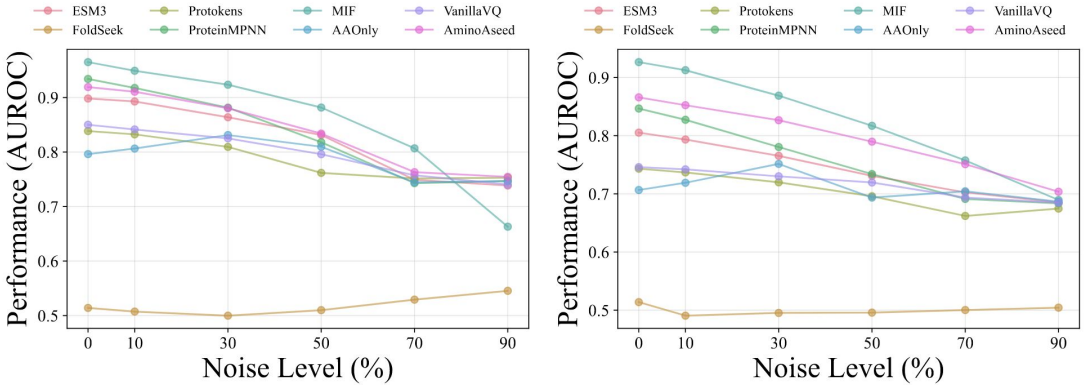
Model	#TaskImpr	#TaskComp	AvgDiff%	AvgRelDiff
ProteinMPNN	5	3	-1.55	-4.72%
MIF	0	13	-1.15	-2.79%
FoldSeek	6	7	-0.42	-13.32%
ProTokens	20	0	4.43	29.60%
ESM3	15	2	2.39	4.33%
VanillaVQ	18	2	3.36	6.98%
AminoAseed	13	5	1.33	1.42%

Table 5: Performance comparison of combining amino acid tokens with structural tokens versus using structural tokens alone across 24 supervised task splits. The reported metrics include: #TaskImpr (number of tasks improved more than 0.005), #TaskComp (number of comparable tasks within ± 0.005 difference), AvgDiff (average difference), and AvgRelDiff (average relative difference).

Ablation Study: Key Insights

3. Robustness to Noise:

- **Finding:** Structural representations are less robust to noise (masking) than sequence-only representations.



(a) Con (SupFam) results. (b) CatInt (SupFam) results.

Figure 5: Supervised task performance with increasing noises in PST-extracted structural representations.

Scaling Studies: Codebook Quality vs. Reconstruction

Key Question: *Does better 3D reconstruction quality imply a better tokenizer?*

Experiment: Varied codebook size (K) while keeping capacity (K×D) constant.

Findings:

- Reconstruction quality is best at very **large** K ($K=2^{14}$).
- However, downstream effectiveness (supervised tasks) and utilization are best at **moderate** K ($K \in [2^8, 2^{10}]$).

Conclusion: Reconstruction quality does not correlate well with codebook quality for downstream tasks.

- This is a critical finding that challenges the standard evaluation metric for many generative structure models. Optimizing solely for reconstruction may lead to a suboptimal tokenizer.

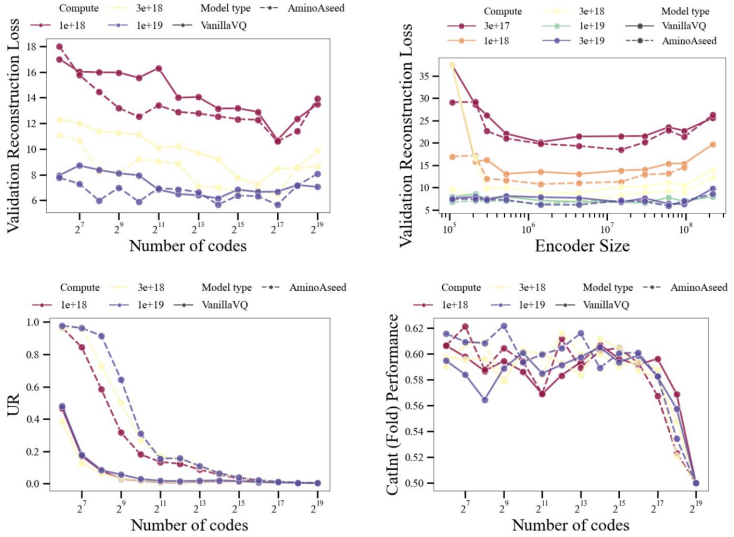


Figure 6: Impact of codebook and encoder sizes on PST quality: Top panels display reconstruction losses versus varying codebook sizes (left) and encoder sizes (right); Bottom panels shows Codebook Utilization Efficiency (left) and Downstream Effectiveness (right) performance versus codebook sizes.

Discussion and Implications

For Benchmarking:

- StructTokenBench provides a necessary and comprehensive tool for the field.
- Highlights the crucial insight that **reconstruction quality is a poor proxy for the utility of structural representations.**

For Model Development:

- **AminoAseed's** recipe provides a simple and highly effective solution to codebook collapse in VQ-VAEs for proteins.
- The codebook reparameterization technique is general and could be applied in other domains (e.g., image or speech VQ-VAEs).

Broader Implications:

- High-quality structural tokens are a key enabling technology for:
    - **Large Multimodal Protein Models:** Integrating structure, sequence, and functional text.
    - **Efficient Structure-Based Drug Discovery:** Replacing expensive computations with fast token-based operations.
    - **Generative Protein Design:** Designing novel proteins by composing structural motifs in "token space".

Conclusion and Future Work

Summary of Contributions:

1. **Introduced StructTokenBench:** The first comprehensive benchmark for local PST, evaluating effectiveness, sensitivity, distinctiveness, and efficiency.
2. **Systematically Evaluated SOTA Models:** Revealed that no single existing model excels across all axes. Identified codebook collapse as a key issue in VQ-VAE models.
3. **Proposed AminoAseed:** A novel PST with a simple recipe (reparameterization and Pareto-optimal configuration) that outperforms existing models across all four benchmark perspectives.

Future Directions:

- Exploring scaling laws for VQ-VAE-based PSTs more deeply.
- Applying AminoAseed tokens in downstream generative models and large multimodal protein models.
- Extending tokenization to all-atom representations or protein complexes.

Q&A 1

Question: The "Codebook Reparameterization" trick is elegant. How does it compare to other methods for mitigating codebook collapse, such as exponential moving averages (EMA) for updates or codebook resetting?

Answer: EMA updates, used in the original VQ-VAE paper, do help by providing a more stable target but they still don't provide a gradient signal to completely unused vectors. Codebook resetting is a heuristic that can revive dead codes, but it can be disruptive to the training process. The reparameterization approach is more principled because it ensures the *entire representational capacity* of the codebook (via the linear layer) is always trainable and evolves smoothly, rather than just updating selected codes or periodically resetting dead ones. It directly addresses the root cause—the lack of a global update mechanism.

Q&A 2

Question: Results show that IF-based models are still strong on downstream tasks. With the optimization challenges of VQ, is it possible that continuous representations are simply a better path forward?

Answer: This is a key debate in the field. The ablation study in Figure 4 is very telling here. When the authors used the *continuous* outputs of the VQ-VAE encoders, performance did not significantly improve. This suggests the advantage of IF-based models isn't necessarily their continuous format, but rather that their end-to-end training objective (predicting sequence) is simpler to optimize and might be better aligned with some downstream tasks. Discrete tokens still hold significant advantages for integration into language models and multimodal systems, and AminoAseed shows that we can close the performance gap while retaining those benefits.

Q&A 3

Question: The paper states that structural tokens are less robust to noise than sequence tokens. What are the implications of this for real-world applications where structural data may be noisy or of low resolution?

Answer: This is a very important practical consideration. It implies that for low-quality experimental structures or less-confident model predictions, relying solely on structural tokens could be risky. A potential solution is to build models that can dynamically weigh the information from sequence and structure based on some confidence metric. For example, in a multimodal model, an attention mechanism could learn to pay more attention to the sequence embeddings when the structural data quality is poor. It also highlights the need for developing tokenization methods that are inherently more robust to noise.

Q&A 4

Question: Authors chose a codebook size of K=512 based on Pareto-optimality. Could you elaborate on how you identified this specific point? Was it sensitive to the specific downstream tasks chosen for the analysis?

Answer: The authors identified this through the scaling experiments shown in Figure 6 and Figure 12. They observed a U-shaped curve for downstream performance versus codebook size, with the optimal range being between 2^8 (256) and 2^{10} (1024). Concurrently, utilization rate drops sharply for K greater than 2^{11}. The choice of $K=2^9$ (512) represents a sweet spot that maximizes downstream effectiveness while keeping utilization high. While the exact optimal point might show some sensitivity to the task mixture, the general U-shape trend was consistent, suggesting that a moderate codebook size is a robust choice.

Thanks for listening!

Questions - Comments?