FLOW-MATCHING-BASED PROTEIN STRUCTURE TOKENIZER MADE EFFICIENT AND EASY

Anonymous authors
Paper under double-blind review

ABSTRACT

As the bridge between protein modality and discrete modeling, protein structure tokenization still largely relies on heavily engineered training objectives tailored to specific downstream tasks and large training datasets, which hinders its transfer to broader application scenarios. To address this issue, we propose ProFiT, a lightweight flow matching tokenizer. With simple training strategies that encourage healthy codebook utilization, ProFiT can be trained efficiently and naturally learns semantically meaningful representations without any manual semantic alignment, while achieving reconstruction quality and generalization that match or surpass those of substantially larger tokenizers. We conduct extensive evaluations across a wide range of settings and demonstrate that ProFiT is a plug-and-play tokenizer adaptable to diverse downstream tasks. This study further reveals the significant potential of the flow matching tokenizer paradigm. Our code is publicly available at `https://anonymous.4open.science/r/ProFiT-5DD9`.

1 INTRODUCTION

The goal of computational biology modeling is shifting from task-specific optimization toward general modeling of broad biological capabilities (Elnaggar et al., 2021; Nijkamp et al., 2023; Heinzinger et al., 2024; Wang et al., 2025; Hayes et al., 2025; Ma et al., 2025). Many such general-purpose models benefit from the discretization of biological data, particularly proteins. Once discretized, protein representations can be jointly modeled with other modalities (e.g., text and amino acid sequences) and naturally align with autoregressive and masked language modeling (MLM) frameworks, thereby benefiting from the scaling laws of Transformers and discrete modeling paradigms (Nijkamp et al., 2023; Wang et al., 2025; Zhang et al., 2024; Dilip et al., 2026a). Therefore, protein structure tokenization serves as a bridge between continuous 3D geometry and discrete generative modeling.

Recent years have seen considerable exploration of protein structure tokenization. Starting with Foldseek, researchers began investigating the discretization of protein structures (Van Kempen et al., 2024). Subsequent studies have progressively explored and advanced their performance in reconstruction and certain downstream tasks. Much of this work builds on classical $SE(3)$-equivariant architectures and training methods, among which some tokenizers have demonstrated the ability to generate proteins under specific architectures (Lin et al., 2023a; Wang et al., 2025; Zhang et al., 2024; Gaujac et al., 2024; Hayes et al., 2025). Later studies further improved aspects such as codebook utilization and training stability (Gao et al., 2024a;b;c; 2025; Yuan et al., 2025).

However, existing tokenizers face several limitations: (i) To balance reconstruction quality and semantic preservation, training objectives are highly engineered, requiring careful weighting of multiple losses for geometric alignment, confidence, and semantic consistency. (ii) Most architectures rely on single-step inference, which severely constrains reconstruction fidelity and leaves little flexibility for simultaneously optimizing reconstruction and semantic representation. (iii) Tokenizers are typically trained on real structures and synthetic data generated by folding models, introducing bias in the output distribution. As a result, tokenizers tend to produce plausible structures rather than faithfully reflecting the encoded input (Lin et al., 2023b; Fleming et al., 2025). (iv) Most practical tokenizers, such as DPLM2 and ESM3, rely on pretraining over large-scale structural datasets, making it prohibitively expensive to train a tokenizer from scratch with comparable (Wang et al., 2025; Hayes et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

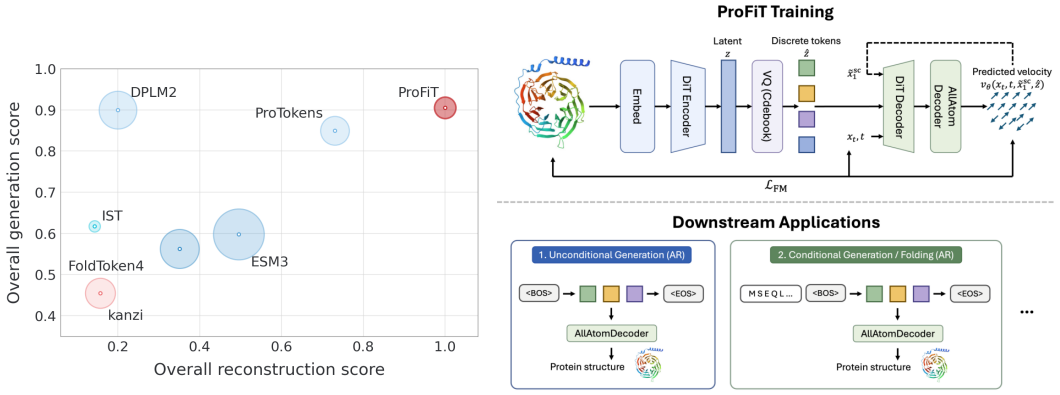

Figure 1: (Left) Comparison between tokenizer scale and performance. The horizontal and vertical axes denote the overall reconstruction and generation scores (defined in App.A.1). The radius of each point is proportional to the logarithmically scaled tokenizer parameter count. Flow-matching–based tokenizers are highlighted in red, while other tokenizers are shown in blue. All figures in this paper follow this visualization convention. (Right) the training framework of ProFiT and its supported downstream applications.

In the image domain, flow-matching-based tokenizers have begun to emerge and have been shown to address the aforementioned limitations (Lipman et al., 2022; Chen et al., 2025; Sargent et al., 2025). These works demonstrate that the flow matching objective alone is sufficient to train high-performance tokenizers, and further suggest that learning the velocity field naturally induces generalization and semantically meaningful representations. Moreover, due to its simple and unified loss formulation, flow matching exhibits strong scalability to larger datasets and more efficient parameter utilization (Chen et al., 2025; Sargent et al., 2025).

We argue that although these phenomena have been preliminarily validated in protein tokenizers, they are still far from being fully explored (Dilip et al., 2026b;a). Previous work has only validated the feasibility of flow-matching–based tokenization on relatively small-scale data, lacking both an explanation of how semantic representations emerge during flow matching and effective strategies to encourage this phenomenon. In addition, there is still a lack of unified evaluations spanning diverse scenarios and tasks to examine whether protein structure tokenizers possess generalizable capabilities. Therefore, we propose ProFiT, which more fully exploits the potential of flow matching and provides a parallel comparison between ProFiT and classical tokenizers across a broad range of settings. Our main contributions are summarized as follows:

1. We propose ProFiT, a protein structure tokenizer that establishes a new state of the art among flow-matching-based approaches using a simple and effective training recipe.
2. We systematically investigate how to stabilize flow matching training on protein data, as well as techniques that encourage stronger semantic representations and better generalization during training.
3. We conduct extensive evaluations of ProFiT in terms of reconstruction quality and semantic understanding, as well as its performance across diverse generative model architectures and tasks, including both unconditional and conditional generation. To our knowledge, this is the first systematic study of the adaptability of multiple protein structure tokenizers specifically across different generative architectures and tasks.

2 Related Work

2.1 Diffusion Model

Diffusion models formulate generation as gradually transforming a simple noise distribution into data through a time-dependent stochastic or deterministic process. Early denoising diffusion prob-



Figure 2: protein reconstruction examples of ProFiT. Gray: ground truth, blue: ProFiT.

abilistic models (DDPMs) established the forward noising and reverse denoising framework, while score-based methods provided a continuous-time view via SDEs and linked diffusion training to score estimation (Ho et al., 2020; Song et al., 2020b). Subsequent work improved efficiency and sample quality through accelerated samplers (e.g., non-Markovian/ODE-style sampling), improved parameterizations, and latent-space diffusion that shifts generation from pixel space to compact representations (Song et al., 2020a; Nichol & Dhariwal, 2021; Lu et al., 2022; Liu et al., 2022; Karras et al., 2022; Rombach et al., 2022). More recently, probability-flow perspectives unified diffusion and transport-based generative modeling: rectified flow reframes training around straightened trajectories, and flow matching directly regresses the target velocity field along prescribed probability paths (Lipman et al., 2022; Liu, 2022). Compared with classical diffusion objectives, flow matching offers a simpler single-loss formulation, flexible path design, and strong scalability, making it an increasingly practical foundation for large-scale generative modeling, including scientific and biological domains (Esser et al., 2024).

2.2 Protein Structure Tokenization

Protein structure tokenization aims to compress continuous 3D protein structure coordinates into a discrete token sequence, thereby adapting it for downstream tasks. Early attempts at discretizing the protein modality, such as Foldseek, introduced highly compact structural alphabets that capture residue-level geometric interactions for efficient retrieval; however, the strong information bottleneck prevents them from performing well on reconstruction tasks (Van Kempen et al., 2024). The dominant paradigm has since evolved toward discrete autoencoder frameworks, where an encoder maps coordinates (C_α, backbone, or all-atom) into latent representations, a quantizer discretizes them into tokens, and a deterministic decoder reconstructs 3D structures. Representative models differ primarily in quantization schemes, vocabulary sizes, encoder equivariance, and training objectives (Lin et al., 2023a; Wang et al., 2025; Zhang et al., 2024; Gao et al., 2024a;b;c; Hayes et al., 2025; Gao et al., 2025). More recently, flow-based autoencoders replace deterministic decoders with conditional flow matching, modeling structure generation as a continuous denoising trajectory from noise to coordinates (Dilip et al., 2026b;a). This formulation simplifies training objectives into a single flow loss while maintaining competitive reconstruction quality and significantly reducing model and data requirements. Despite rapid progress, the design space of protein structure tokenizers remains highly unconstrained, with limited consensus on architecture or quantization strategies. Moreover, most tokenizers can maintain strong reconstruction ability, but only some tokenizers demonstrate general adaptability to generative models.

3 Preliminaries

3.1 Flow Matching

Flow matching provides a framework for learning continuous-time generative models by directly regressing a velocity field that transports a simple base distribution to a target data distribution (Lipman et al., 2022). Let p_0 denote a tractable base distribution and p_1 the data distribution. The goal is to construct a time-dependent probability path $\{p_t\}_{t\in[0,1]}$ connecting p_0 to p_1, and learn a velocity field $v_\theta(x,t)$ such that samples evolve according to the ordinary differential equation (ODE)

$$\frac{dx_t}{dt} = v_\theta(x_t, t), \quad x_0 \sim p_0, \tag{1}$$

and the marginal distribution of x_t matches p_t for all $t \in [0, 1]$.

Flow matching avoids explicit likelihood computation by supervising the model with a target velocity field derived from a prescribed conditional probability path. Specifically, given a coupling between $x_0 \sim p_0$ and $x_1 \sim p_1$ and the path $x_t = \phi_t(x_0, x_1)$, the ground-truth velocity field is then given by

$$v^*(x_t, t) = \frac{d}{dt}\phi_t(x_0, x_1), \tag{2}$$

which induces a marginal flow consistent with p_t. The model is trained by minimizing

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\,(x_0,x_1)}\left[\|v_\theta(x_t, t) - v^*(x_t, t)\|_2^2\right]. \tag{3}$$

This formulation decouples the choice of probability path from the model parameterization, allowing flexible design of interpolation schemes. A common choice is $x_0 \sim \mathcal{N}(0, I), x_1 \sim p_{\text{data}}$, and defines the conditional path $x_t = \alpha(t)x_1 + \sigma(t)x_0$, where $\alpha(t)$ and $\sigma(t)$ are scalar schedules satisfying boundary conditions $\alpha(0) = 0, \alpha(1) = 1$ and $\sigma(0) = 1, \sigma(1) = 0$. This construction ensures $x_0 \sim p_0$ and $x_1 \sim p_1$, while interpolating between noise and data. The corresponding target velocity field is obtained by differentiating the path:

$$v^*(x_t, t) = \dot{\alpha}(t)x_1 + \dot{\sigma}(t)x_0. \tag{4}$$

4 METHOD

We cast protein backbone tokenization as a flow-matching autoencoder with a discrete bottleneck. Given a protein of length L, let its backbone coordinates be $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{L}, \mathbf{x}_i \in \mathbb{R}^{4\times 3}$, where each residue contains the heavy atoms $\{\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}, \mathrm{O}\}$ (measured in nanometers). Our objective is to learn a mapping $\mathbf{X} \mapsto \mathbf{c} \in \{1, \ldots, K\}^L$ that compresses a continuous structure into a sequence of discrete tokens while preserving faithful reconstruction, robustness to perturbations, and high codebook utilization.

4.1 OVERALL ARCHITECTURE

The architecture of ProFiT follows prior practices for modeling proteins with non-equivariant networks (Abramson et al., 2024; Li et al., 2026). The encoder maps an input backbone structure $\mathbf{x}_1 \in \mathbb{R}^{L\times 4\times 3}$ to a latent representation $\mathbf{z}$ through N_{enc} DiT blocks with pair bias:

$$\mathbf{s}_0 = \mathrm{Embed}(\mathbf{x}_1), \quad \mathbf{s}_i = \mathrm{DiT}_i(\mathbf{s}_{i-1}, \mathbf{x}_1), i = 1, \ldots, N_{\text{enc}}, \quad \mathbf{z} = \mathrm{Linear}(\mathbf{s}_{N_{\text{enc}}}) \in \mathbb{R}^{L\times d}. \tag{5}$$

We use an Atom Transformer as the Embed layer (Abramson et al., 2024). Since ProFiT operates only at the backbone level, it aggregates backbone atom features into residue-level representations. The pair bias in each encoder DiT block is constructed from binned inter-residue C_α distances and sequence indices of $\mathbf{x}_1$. The final token-level representation $\mathbf{s}_{N_{\text{enc}}}$ is projected into an extremely low-dimensional space, forming a tight information bottleneck.

The continuous latent $\mathbf{z}$ is discretized by a vector quantizer $\mathcal{Q}$ with codebook $\mathcal{E} = \{\mathbf{e}_k\}_{k=1}^{K} \subset \mathbb{R}^d$, $K = 8192$ (Van Den Oord et al., 2017). For each residue position i,

$$c_i = \arg\min_{k\in[K]} \|\mathbf{z}_i - \mathbf{e}_k\|_2^2, \qquad \hat{\mathbf{z}}_i = \mathbf{e}_{c_i}, \tag{6}$$

with gradients propagated through a straight-through estimator.

The decoder reconstructs the structure from a noisy input $\mathbf{x}_t$ at flow-matching timestep t, conditioned on the quantized latent $\hat{\mathbf{Z}}$ and using the same DiT stack but with different conditioning signals:

$$v_\theta(\mathbf{x}_t, t, \hat{\mathbf{Z}}) = \mathrm{AllAtomDecoder}\left(\mathrm{DiTStack}\left(\mathbf{x}_t, \tilde{\mathbf{x}}_1^{\text{sc}}, t, \hat{\mathbf{Z}}\right)\right), \tag{7}$$

The decoder's pair bias is constructed from binned pairwise distances derived from both the noisy trajectory $\mathbf{x}_t$ and the self-conditioned previous-step prediction $\tilde{\mathbf{x}}_1^{\text{sc}}$, along with sequence indices and a time embedding. $\mathrm{AllAtomDecoder}$ module directly outputs the velocity field $v_\theta(\mathbf{x}_t, t, \hat{\mathbf{Z}})$ used in flow matching. The discrete tokens thus serve as a semantic token of the structure, while geometric detail is reconstructed by a conditional continuous-time generative model.

4.2 Enhancements to Vector Quantization

Vanilla VQ training is notoriously known to lead to weaker model performance and an imbalanced codebook distribution. We observe two types of phenomena that degrade the performance of ProFiT: (i) **codebook collapse**, including but not limited to low codebook utilization and the tendency of code norms to approach zero during training and eventually become unused; and (ii) **decoder overfitting**, where the downstream decoder overfits to the exact distribution of $\hat{\mathbf{Z}}$ and degrades sharply when the token sequence is even mildly perturbed. We address these issues with two complementary mechanisms.

Random Token Masking. At inference time, token sequences produced by a downstream prior inevitably deviate from the ground-truth encoding; a decoder trained only on clean conditioning will overfit and degrade rapidly. To build error tolerance, at each training step we sample a corruption ratio $r' \sim \mathcal{U}(0, r)$ where r is a fixed hyperparameter, then independently mask each quantized token $\hat{\mathbf{z}}_i$ with probability r':

$$\tilde{\mathbf{z}}_i = (1 - m_i)\hat{\mathbf{z}}_i, \qquad m_i \sim \mathrm{Bern}(r'), \qquad i = 1, \ldots, L. \tag{8}$$

Dead Code Replacement. In our training, we observe that among randomly initialized codes, a large proportion tend to be unused. In EMA-based VQ updating, these unused codes receive updates only through EMA decay, causing their norms to gradually converge to zero and eventually become dead codes, which undermines efficient codebook utilization. The underlying causes of this phenomenon may be multifaceted (Fifty et al., 2025). A choice is to maintain an EMA cluster size $C_k^{(s)}$ for each code k with decay γ, updated as $C_k^{(s)} = \gamma C_k^{(s-1)} + (1 - \gamma) \sum_{i:c_i=k} 1$ at each step (Zheng & Vedaldi, 2023). We additionally, after each update, replace any code whose $C_k^{(s)}$ falls below a threshold τ_{dead} by a vector sampled uniformly from the current batch's encoder outputs, and its EMA statistics are reset to a predefined mass $\rho > \tau_{\text{dead}}$. This reset mass acts as a grace period that prevents the revived code from being immediately declared dead again, giving it time to attract real assignments.

4.3 Training Objective

Let π_{data} denote the distribution over backbone coordinates after centroid removal and random-rotation augmentation. Following rectified flow (Liu, 2022) we construct the linear probability path $\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1, \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_3), \mathbf{x}_1 \sim \pi_{\text{data}}, t \in [0, 1]$, whose target velocity is

$$\mathbf{v}_t^\star = \frac{\mathbf{x}_1 - \mathbf{x}_t}{1 - t}. \tag{9}$$

The decoder directly predicts the velocity field $v_\theta(\mathbf{x}_t, t, \tilde{\mathbf{Z}})$. With $\tilde{\mathbf{Z}}$ obtained from Eq. equation 8, we use the flow-matching loss over all $4L$ backbone atoms as the training objective,

$$\mathcal{L}_{\text{FM}}(\phi, \theta) = \mathbb{E}_{\mathbf{x}_1, \mathbf{x}_0, t}\left[\frac{1}{12L}\sum_{i=1}^{4L}\left\|\mathbf{v}_{t,i}^\star - v_{\theta,i}(\mathbf{x}_t, t, \tilde{\mathbf{Z}})\right\|_2^2\right]. \tag{10}$$

We use the same t-sampling distribution as ProteinAE: $t \sim 0.2 \cdot \mathcal{U}(0, 1) + 0.8 \cdot \mathrm{Beta}(1.9, 1.0)$ (Li et al., 2026).

4.4 Sampling

During inference, we start from $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_3)$ and the codebook embeddings $\hat{\mathbf{Z}} = \mathcal{Q}^{-1}(\mathbf{c})$ without masks, we sample from the unified SDE below, whose ODE form is recovered by setting $\eta = \gamma = 0$:

$$\mathrm{d}\mathbf{x}_t = \left[v_\theta(\mathbf{x}_t, t, \hat{\mathbf{Z}}) + g(t)\eta s_\theta(\mathbf{x}_t, t, \hat{\mathbf{Z}})\right]\mathrm{d}t + \sqrt{2g(t)\gamma}\mathrm{d}\mathbf{w}_t, \tag{11}$$

where (η, γ) are score and noise weight, $g(t)$ is the diffusion coefficient, for which we use $g(t) = \frac{1-t}{t}$, and the score is derived from the predicted velocity through the stochastic-interpolant identity (Song et al., 2020b; Lipman et al., 2022)

$$s_\theta(\mathbf{x}_t, t, \hat{\mathbf{Z}}) = \frac{t v_\theta(\mathbf{x}_t, t, \hat{\mathbf{Z}}) - \mathbf{x}_t}{(1 - t)}. \tag{12}$$

5 EXPERIMENT

In this section, we evaluate whether ProFiT serves as a high-fidelity, versatile, and plug-and-play protein structure tokenizer by training and benchmarking it against several widely adopted baselines across multiple tasks. We use a filtered version of the Foldseek-clustered AFDB comprising 598,656 samples, and follow Kanzi in truncating sequences to 256 residues (Barrio-Hernandez et al., 2023; Dilip et al., 2026b). The dataset contains no missing atoms and covers a broad range of structural distributions. During training, ProFiT converges after seeing 7.7M training samples, and we use the resulting checkpoint for all experiments. Further details are provided in App. B.1.

Our baselines cover representative protein structure tokenizer designs. ProTokens, IST, FoldToken4, DPLM2, and ESM3 are widely used learned discrete tokenizers for the protein modality, and have demonstrated strong adaptability to downstream modeling tasks (Lin et al., 2023a; Gao et al., 2024c; Gaujac et al., 2024; Wang et al., 2025; Hayes et al., 2025). For FoldToken4, we use hierarchy level 12, corresponding to a vocabulary size of 4,096. We also include Kanzi, a recent flow-matching–based tokenizer, to evaluate the improvements achieved by ProFiT within the same technical paradigm (Dilip et al., 2026b). We assess fidelity using the reconstruction task, and further consider unconditional generation, folding (conditional generation), and protein function understanding as representative downstream tasks. Finally, we conduct ablation studies on our design choices. Unless otherwise specified, we use 20 sampling steps for both the ODE and SDE samplers. The hyperparameters and search space for ProFiT are provided in App. B.2.

5.1 RECONSTRUCTION

We evaluate ProFiT on a wide range of out-of-distribution test sets and show that it exhibits minimal bias toward the output distribution at inference time. This avoids degenerate behavior where the decoder relies on limited information from the quantizer and effectively performs a re-folding process. We argue that faithfully preserving the original structure is a necessary condition for applying protein structure tokenizers to downstream tasks.

Given that most protein structure tokenizers are trained on real monomer structures or synthetic data generated by folding models, we further assess ProFiT on structural fragments that cannot exist independently in biological systems (i.e., outside the distribution of folding model outputs), such as single chains extracted from multimeric complexes or motif fragments. These results further support that ProFiT exhibits minimal output distribution bias.

Tab. 1 presents our results on four test sets, including CAMEO2022, CASP15, CASP16, and a held-out D_{FS} subset (Haas et al., 2018; Simpkin et al., 2023; Jing et al., 2023; Kryshtafovych et al., 2026). Reconstruction quality is evaluated using C_α RMSD and TM-score (Zhang & Skolnick, 2004). When using the SDE sampler, we set $\eta = 0.5$. Detailed metric definitions are provided in App. A.2.

Table 1: Reconstruction accuracy results. Subscripts denote standard deviations, with values in RMSD columns reported in units of 10^{-2} and values in TM-score columns reported in units of 10^{-3}.

	CAMEO2022		CASP15		CASP16		D_{FS}	
	RMSD ↓	TM ↑	RMSD ↓	TM ↑	RMSD ↓	TM ↑	RMSD ↓	TM ↑
IST	3.86	.744	8.64	.688	6.67	.676	3.55	.721
DPLM2	1.87	.903	8.52	.792	8.30	.767	2.07	.872
FoldToken4	1.22	.941	6.47	.845	2.55	.871	1.40	.911
ESM3	1.02	<u>.969</u>	1.99	.918	3.41	.891	0.91	.962
ProTokens	<u>0.94</u>	.963	<u>1.23</u>	<u>.965</u>	<u>1.25</u>	<u>.957</u>	0.89	.958
kanzi[1]	9.33	.674	21.86	.520	13.33	.604	0.98	.943
ProFiT(ODE)	$0.96_{2.0}$	$.964_{0.4}$	$1.48_{7.6}$	$.954_{1.5}$	$1.41_{4.7}$	$.952_{1.9}$	<u>$0.71_{0.8}$</u>	<u>$.978_{0.5}$</u>
ProFiT(SDE)	$\mathbf{0.64}_{0.5}$	$\mathbf{.983}_{0.2}$	$\mathbf{1.19}_{3.8}$	$\mathbf{.967}_{0.8}$	$\mathbf{1.01}_{2.0}$	$\mathbf{.977}_{0.5}$	$\mathbf{0.42}_{0.3}$	$\mathbf{.988}_{0.1}$

[1] See App. C.1 for a discussion of the differences between our values and those reported in the Kanzi paper.

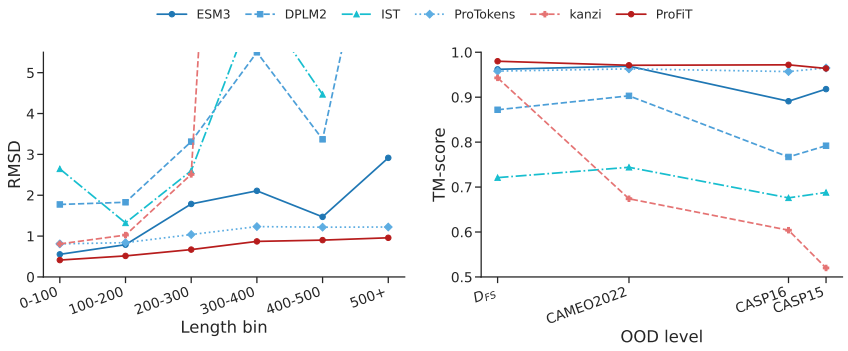

Figure 4: ProFiT exhibits stronger generalization, with reconstruction error largely insensitive to both protein length and the OOD level. The OOD level is quantified using pairwise TM-score against the PDB dataset, and is normalized such that 0 corresponds to in-distribution samples and 1 corresponds to the most out-of-distribution samples.

Tab. 1 shows that ProFiT consistently achieves either the best or near-best reconstruction performance across all evaluated settings, demonstrating strong and stable preservation of protein backbone structure. Despite a relatively compact model size, it remains competitive with larger baselines, indicating efficient utilization of model capacity.



Figure 3: Reconstruction results of DPLM2 and ProFiT on T1137s3. Gray: ground truth, purple: ProFiT, blue: DPLM2

We further observe that, as the distribution shift of the test sets increases, the test sets include structures that cannot exist as independently foldable monomers (e.g., T1137s3, see Fig. 2 & 3), the reconstruction error of ProFiT increases only marginally, and its output distribution does not collapse toward the distribution of monomeric proteins. For example, as shown in Fig. 3, the reconstruction produced by DPLM2 tends to "refold" the protein into a plausible monomeric structure rather than faithfully reconstructing the original input. Such robustness suggests that ProFiT does not overfit to the implicit output distributions of its training sources, but instead learns more generalizable structural representations.

5.2 Unconditional Generation

In this section, we evaluate ProFiT on the unconditional generation task. To cover a range of practical scenarios, we assess all baselines and ProFiT under both autoregressive (AR) and discrete diffusion architectures. For the AR architecture, we train a 220M-parameter decoder-only Transformer to model protein tokens, using BOS and EOS tokens to indicate the beginning and end of each sequence. For the discrete diffusion architecture, we directly follow the official training script to train DPLM2. We generated 250 samples for each tokenizer under both the AR and DPLM architectures. For DPLM, following its original setting, we sampled 50 sequences at each length of 100, 200, 300, 400, and 500 residues. We evaluate generation performance in terms of quality and diversity. In the AR generation experiments, we set $\eta = 0.5, \gamma = 0$; in the DPLM2-650M generation experiments, we set $\eta = 0.25, \gamma = 0.25$ (Wang et al., 2025). Details of the AR model and detailed metric definitions are provided in App. A.2.

Tab. 2 shows that ProFiT is the only tokenizer that maintains consistently strong performance across both autoregressive and discrete diffusion architectures, without any architecture specific customization. In contrast, state-of-the-art methods tend to specialize in a single generative paradigm: ProTokens achieves its strongest results under the autoregressive setting but degrades under diffusion, while DPLM2 exhibits the opposite behavior. This suggests that existing tokenizers are often tightly coupled to the inductive biases of their respective architectures, whereas ProFiT provides a more generalizable representation that transfers robustly across both regimes.

Table 2: Results for unconditional generation with the AR and DPLM2 architectures. Des., Div., and Nov. denote designability, diversity, and novelty, respectively. Standard deviations from five independent runs are indicated in the lower-right corner. Because Kanzi does not provide a backbone-level checkpoint, we cannot evaluate its secondary-structure composition.

	Quality			Diversity				
	scRMSD ↓	scTM ↑	Des. ↑	Div. ↓	Nov. ↓	α%	β%	c%
AR								
DPLM2	$8.75_{.60}$	$.626_{.027}$	$.71_{.03}$	$.348_{.024}$	$.831_{.031}$	$.411_{.003}$	$.071_{.009}$	$.518_{.010}$
FoldToken4	$8.07_{.64}$	$.672_{.012}$	$.60_{.08}$	$.232_{.015}$	$.759_{.012}$	$.451_{.017}$	$.080_{.003}$	$.469_{.017}$
ESM3	$6.74_{.16}$	$.665_{.008}$	$.69_{.01}$	$.241_{.001}$	$.791_{.044}$	$.358_{.004}$	$.198_{.005}$	$.444_{.006}$
IST	$6.36_{.85}$	$.724_{.021}$	$.80_{.02}$	$.229_{.020}$	$.773_{.031}$	$.386_{.013}$	$.200_{.011}$	$.414_{.010}$
ProTokens	$\mathbf{5.19}_{.22}$	$\mathbf{.800}_{.008}$	$\mathbf{.84}_{.03}$	$.223_{.013}$	$.810_{.011}$	$.335_{.017}$	$.071_{.005}$	$.595_{.012}$
kanzi	$12.92_{.61}$	$.421_{.021}$	$.63_{.05}$	$.289_{.007}$	$.729_{.010}$	-	-	-
ProFiT	$\underline{5.80}_{.33}$	$\underline{.730}_{.019}$	$\underline{.83}_{.04}$	$.230_{.022}$	$.715_{.012}$	$.453_{.013}$	$.126_{.007}$	$.421_{.006}$
DPLM2								
DPLM2	$\mathbf{2.54}_{.44}$	$\mathbf{.879}_{.004}$	$\mathbf{.98}_{.01}$	$.211_{.001}$	$.650_{.015}$	$.487_{.006}$	$.069_{.007}$	$.444_{.003}$
FoldToken4	$16.50_{.97}$	$.660_{.010}$	$.62_{.01}$	$.408_{.017}$	$.651_{.046}$	$.178_{.008}$	$.095_{.004}$	$.727_{.007}$
ESM3	$19.30_{1.52}$	$.600_{.013}$	$.67_{.03}$	$.374_{.041}$	$.515_{.025}$	$.509_{.008}$	$.109_{.010}$	$.382_{.003}$
IST	$21.94_{1.07}$	$.499_{.018}$	$.35_{.04}$	$.300_{.003}$	$.658_{.008}$	$.787_{.019}$	$.059_{.005}$	$.154_{.017}$
ProTokens	$7.42_{.41}$	$.640_{.003}$	$.74_{.01}$	$.314_{.012}$	$.829_{.009}$	$.486_{.002}$	$.008_{.001}$	$.505_{.002}$
kanzi	$11.64_{.41}$	$.494_{.004}$	$.45_{.02}$	$.346_{.008}$	$.701_{.010}$	-	-	-
ProFiT	$\underline{6.62}_{.20}$	$\underline{.695}_{.002}$	$\underline{.83}_{.02}$	$.290_{.008}$	$.649_{.004}$	$.651_{.015}$	$.061_{.001}$	$.287_{.015}$

In Fig. 5, we additionally provide a comparison between autoregressive sampling temperature and generation quality. When modeling ProFiT tokens with an autoregressive model, the AR likelihood serves as a strong indicator of the quality of the generated structures. This suggests that ProFiT enables the likelihood estimated by the AR model to be used as a proxy for early screening of generation quality.

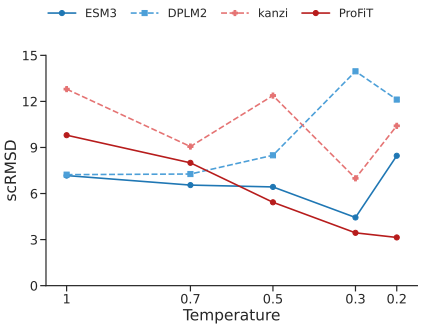

Figure 5: Effect of sampling temperature on AR generation quality.

5.3 Conditional Generation

To evaluate the performance of different tokenizers in cross-modal tasks, we select the representative folding task as our conditional generation benchmark. Given the strong performance of DPLM2 on folding in the discrete domain, we retrain DPLM2 using the same training dataset but with different structure tokenizers for discretization, thereby assessing the adaptability of ProFiT to multimodal data (Wang et al., 2025). We evaluate on CAMEO2022 using the same metrics as in Sec. 5.1. In each run, we generate one structure per sequence and repeat the evaluation over five independent runs.

Overall, ProFiT achieves strong performance in the conditional folding setting, reaching results that are competitive with tokenizers specifically optimized for DPLM2. This indicates that the learned tokenizer generalizes well to cross-modal conditional generation tasks and retains strong structural fidelity even in challenging folding scenarios.

We evaluate ProFiT on protein understanding tasks. Following the setup of StructTokenBench, we select a subset of its functional site prediction tasks (Yuan et al., 2025). Functional site prediction is a residue-level binary classification task that determines whether each residue belongs to a particular type of functional site. We use the AUROC of the binary classification task as the evaluation metric. Notably, ProFiT is trained without any auxiliary losses, meaning that we do not explicitly enforce semantic alignment of the tokens. Nevertheless, ProFiT naturally acquires semantically meaningful structural representations.

Table 3: Discrete diffusion conditional generation.

	CAMEO2022	
	RMSD ↓	TM ↑
IST	$10.69_{1.03}$	$.516_{.032}$
ProTokens	$9.84_{1.10}$	$.548_{.035}$
FoldToken4	$9.05_{1.40}$	$.639_{.035}$
ESM3	$8.75_{1.19}$	$.649_{.035}$
DPLM2	$\mathbf{8.06}_{1.04}$	$\mathbf{.765}_{.035}$
kanzi	$16.42_{1.62}$	$.320_{.031}$
ProFiT	$\underline{8.61}_{1.27}$	$\underline{.724}_{.036}$

Table 4: Functional site prediction performance on StructTokenBench.

Task	**Split**	**Model**			
		Foldseek	ProTokens	ESM3	ProFiT
BindInt	Fold	**53.18**	44.66	44.30	<u>44.97</u>
	SupFam	46.20	<u>86.05</u>	**90.77**	81.44
Con	Fold	49.26	<u>56.23</u>	55.22	**56.94**
	SupFam	51.39	74.33	**80.53**	<u>79.19</u>
Rep	Fold	47.70	**77.25**	74.70	<u>75.30</u>
	SupFam	52.53	78.90	**82.36**	<u>79.42</u>
Ept	Fold	54.52	54.69	**63.69**	<u>55.11</u>
	SupFam	50.56	**67.52**	61.97	<u>63.31</u>

5.4 Understanding

We evaluate ProFiT on protein understanding tasks in Tab. 4. Following the setup of StructTokenBench, we select a subset of its functional site prediction tasks Yuan et al. (2025). Functional site prediction is a residue-level binary classification task that determines whether each residue belongs to a particular type of functional site. We report AUROC for residue-level binary classification across Fold and SupFam splits. Notably, ProFiT is trained without any auxiliary losses, meaning that we do not explicitly enforce semantic alignment of the tokens. Nevertheless, ProFiT naturally acquires semantically meaningful structural representations.

5.5 Codebook Utilization

We assess codebook utilization on CAMEO2022 dataset, using the utilization rate, normalized perplexity, and code similarity. As shown in Tab. 5, ProFiT achieves the highest UR and PPL/K despite having the largest vocabulary and a low-dimensional codebook. It also has the second-lowest code similarity, surpassed only by DPLM2, which uses LFQ (Yu et al., 2024; Wang et al., 2025). These results indicate broad and relatively balanced use of ProFiT 's codebook, consistent with the intended effect of dead-code replacement.

Table 5: Codebook utilization across all baselines.

Model	UR ↑	PPL/K ↑	Similarity ↓
ProFiT	**98.91%**	**86.83%**	<u>0.959</u>
ProTokens	<u>98.24%</u>	66.15%	0.994
ESM3	92.36%	<u>74.70%</u>	0.984
DPLM2	82.76%	63.48%	**0.846**
Kanzi	71.20%	59.78%	0.986
FoldToken4	52.93%	19.09%	0.987
IST	27.88%	22.31%	0.961

6 Discussions

In this work, we present ProFiT, a lightweight flow-matching-based protein structure tokenizer with a compact discrete bottleneck and a conditional continuous-time decoder. By combining a simple objective with stable VQ training, ProFiT establishes a new state of the art among flow-matching-based protein structure tokenizers while achieving strong reconstruction fidelity, transferability, and emergent token semantics without auxiliary semantic losses. These results suggest that effective protein tokenization does not necessarily require heavily engineered multi-term objectives or large equivariant backbones, and highlight several practical directions for future flow-matching tokenizers: robust codebook dynamics, evaluation beyond standard in-distribution settings, and improved path design and conditioning strategies for better scalability and robustness across biological modalities.

Flow-matching–based tokenization is still far from fully explored. Important open questions include how such tokenizers scale, and whether other generative modeling paradigms may be better suited for protein modeling. These remain open questions for future work.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, assist with translation, clean and reformat datasets, support qualitative and thematic analysis, interpret results, and create scientific figures and images. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, or design or provide feedback on research methodologies or experiments. The use of generative AI tools to formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, or propose or refine hypotheses was not applicable to this work. We reviewed all code and suggestions generated by LLMs and verified their correctness. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work does not involve human or animal subjects, personal or sensitive information, or interactions with individuals. We use publicly available protein structure datasets and comply with their stated terms of use. The methods and analyses presented here do not introduce identifiable privacy or security risks, and we are not aware of any foreseeable harmful applications or discrimination-related concerns arising from this work. We have no conflicts of interest or external sponsorship to declare. To the best of our knowledge, this study complies with applicable research-integrity and legal requirements.

REPRODUCIBILITY STATEMENT

We describe all methodological design choices in Sec. 4 and provide comprehensive training and experimental details, including model hyperparameters, in App. B to facilitate reproducibility. We also provide an anonymous repository for convenient reproduction and will release the datasets and full repository in a subsequent public release.

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.

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.

Yinbo Chen, Rohit Girdhar, Xiaolong Wang, Sai Saketh Rambhatla, and Ishan Misra. Diffusion autoencoders are scalable image tokenizers. *arXiv preprint arXiv:2501.18593*, 2025.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

Rohit Dilip, Ayush Varshney, and David Van Valen. Adaptive protein tokenization. *arXiv preprint arXiv:2602.06418*, 2026a.

Rohit Dilip, Evan Zhang, Ayush Varshney, and David Van Valen. Flow autoencoders are effective protein tokenizers. In *International Conference on Learning Representations*, volume 2026, pp. 145901–145926, 2026b.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers

for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Christopher Fifty, Ronald Junkins, Dennis Duan, Aniketh Iyengar, Jerry Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. In *International Conference on Learning Representations*, volume 2025, pp. 19153–19188, 2025.

Jennifer Fleming, Paulyna Magana, Sreenath Nair, Maxim Tsenkov, Damian Bertoni, Ivanna Pidruchna, Marcelo Querino Lima Afonso, Adam Midlik, Urmila Paramval, Augustin Žídek, et al. Alphafold protein structure database and 3d-beacons: new data and capabilities. *Journal of molecular biology*, 437(15):168967, 2025.

Zhangyang Gao, Cheng Tan, and Stan Z Li. Foldtoken2: Learning compact, invariant and generative protein structure language. *bioRxiv*, pp. 2024–06, 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.

Jürgen Haas, Alessandro Barbato, Dario Behringer, Gabriel Studer, Steven Roth, Martino Bertoni, Khaled Mostaguir, Rafal Gumienny, and Torsten Schwede. Continuous automated model evaluation (cameo) complementing the critical assessment of structure prediction in casp12. *Proteins: Structure, Function, and Bioinformatics*, 86:387–398, 2018.

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.

Michael Heinzinger, Konstantin Weissenow, Joaquin Gomez Sanchez, Adrian Henkel, Milot Mirdita, Martin Steinegger, and Burkhard Rost. Bilingual language model for protein sequence and structure. *NAR Genomics and Bioinformatics*, 6(4):lqae150, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Bowen Jing, Ezra Erives, Peter Pao-Huang, Gabriele Corso, Bonnie Berger, and Tommi Jaakkola. Eigenfold: Generative protein structure prediction with diffusion models. *arXiv preprint arXiv:2304.02198*, 2023.

Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules*, 22(12):2577–2637, 1983.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Progress and bottlenecks for deep learning in computational structure biology: Casp round xvi. *Proteins: Structure, Function, and Bioinformatics*, 94(1):5–14, 2026.

Shaoning Li, Le Zhuo, Yusong Wang, Mingyu Li, Xinheng He, Fandi Wu, Hongsheng Li, and Pheng-Ann Heng. Proteinae: Protein diffusion autoencoders for structure encoding. In *International Conference on Learning Representations*, volume 2026, pp. 69939–69960, 2026.

Xiaohan Lin, Zhenyu Chen, Yanheng Li, Xingyu Lu, Chuanliu Fan, Ziqiang Cao, Shihao Feng, Yi Qin Gao, and Jun Zhang. Protokens: A machine-learned language for compact and informative encoding of protein 3d structures. 2023a.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Zicheng Ma, Chuanliu Fan, Zhicong Wang, Zhenyu Chen, Xiaohan Lin, Yanheng Li, Shihao Feng, Ziqiang Cao, Jun Zhang, and Yi Qin Gao. Prottex: Structure-in-context reasoning and editing of proteins with large language models. *Journal of Chemical Information and Modeling*, 65(13): 6599–6612, 2025.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021.

Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Kyle Sargent, Kyle Hsu, Justin Johnson, Li Fei-Fei, and Jiajun Wu. Flow to the mode: Mode-seeking diffusion autoencoders for state-of-the-art image tokenization. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19471–19481. IEEE, 2025.

Adam J Simpkin, Shahram Mesdaghi, Filomeno Sánchez Rodríguez, Luc Elliott, David L Murphy, Andriy Kryshtafovych, Ronan M Keegan, and Daniel J Rigden. Tertiary structure assessment at casp15. *Proteins: Structure, Function, and Bioinformatics*, 91(12):1616–1635, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Protenix Team, Yuxuan Zhang, Chengyue Gong, Hanyu Zhang, Wenzhi Ma, Zhenyu Liu, Xinshi Chen, Jiaqi Guan, Lan Wang, Yanping Yang, et al. Protenix-v1: Toward high-accuracy open-source biomolecular structure prediction. *bioRxiv*, pp. 2026–02, 2026.

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, 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.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Dplm-2: A multimodal diffusion protein language model. In *International Conference on Learning Representations*, volume 2025, pp. 35463–35490, 2025.

Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion-tokenizer is key to visual generation. In *International Conference on Learning Representations*, volume 2024, pp. 765–783, 2024.

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

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. *bioRxiv*, pp. 2024–12, 2024.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22741–22750. IEEE, 2023.

A METRIC DEFINITIONS

A.1 OVERALL SCORE DEFINITIONS

The overall scores in Fig. 1 are derived from the relative performance comparison between ProFiT and other tokenizers, where all metrics are reported in RMSD or scRMSD. Each score is normalized by the best-performing method for the corresponding metric, such that the highest score is scaled to 1. The overall reconstruction score is computed from the results on the four reconstruction datasets, while the overall generation score is computed from the remaining three generation-related evaluations. More detailed score comparisons are provided in Fig. 6.

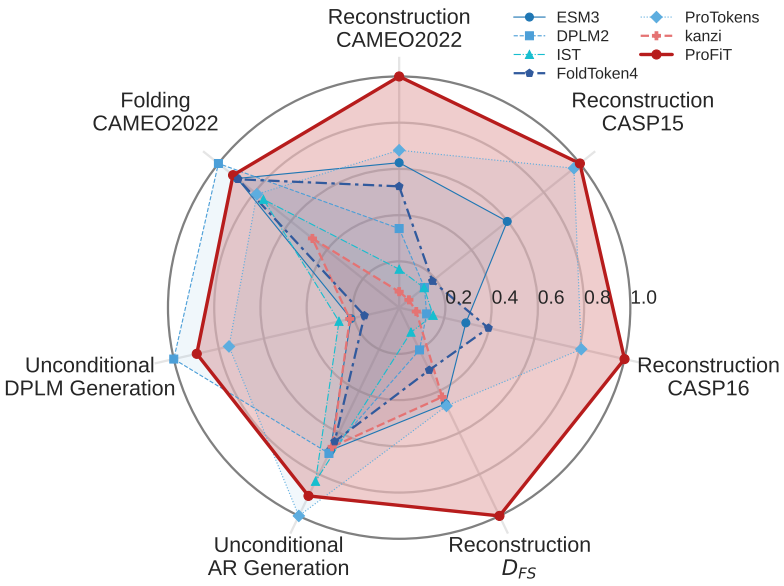

Figure 6: Relative performance comparison.

A.2 EXPERIMENT METRIC DEFINITIONS

Root-Mean-Square Deviation (RMSD). RMSD is a widely used metric for evaluating the geometric similarity between predicted and native protein structures. For a protein of L residues, given predicted coordinates $(\{x_i\}_{i=1}^{L})$ and reference coordinates $(\{y_i\}_{i=1}^{L})$, RMSD is computed as:

$$\text{RMSD} = \sqrt{\frac{1}{L}\sum_{i=1}^{L}|x_i - y_i|^2}. \tag{13}$$

Before computation, the predicted structure is rigidly aligned to the reference structure to remove the effects of global translation and rotation, using the Kabsch alignment algorithm. RMSD is sensitive to local deviations and provides a straightforward measure of absolute structural accuracy.

Template Modeling Score (TM-Score). TM-Score measures the global structural similarity between two protein structures while normalizing for protein length, providing a more robust metric for evaluating fold-level correctness. It is defined as:

$$\text{TM-Score} = \max\left[\frac{1}{L_{\text{ref}}}\sum_{i=1}^{L_{\text{ali}}}\frac{1}{1+(d_i/d_0(L_{\text{ref}}))^2}\right]. \tag{14}$$

where d_i is the distance between aligned residues, L_{ref} is the length of the reference protein, L_{ali} is the number of aligned residues, and d_0 is a length-dependent normalization factor. TM-Score values range from 0 to 1, with higher scores indicating better overall fold similarity, and are less sensitive to local structural noise compared to RMSD.

scRMSD and scTM. We evaluate self-consistency through a structure-to-sequence-to-structure loop. For each generated structure token sequence, we first decode a 3D backbone, then design amino-acid sequences with ProteinMPNN (8 sequences per target, sampling temperature 0.1, C_α-only mode), and finally refold all designed sequences with ESMFold (Dauparas et al., 2022; Lin et al., 2023b). We report the best-case values across refolded candidates: $\mathrm{scRMSD} = \min$ RMSD and $\mathrm{scTM} = \max$ TM-score. RMSD is computed on C_α atoms after Kabsch alignment, and TM-score is computed with the TMscore binary on C_α-only PDBs.

Designability (Des.). Designability is defined as the proportion of samples with $\mathrm{scTM} > 0.5$.

Diversity (Div.). Diversity is computed only among designable structures. We define a valid pair as a pair of designable proteins whose length difference does not exceed 10. The reported diversity is the mean pairwise TM-score across valid pairs.

Novelty (Nov.). Novelty is evaluated for designable samples by computing their maximum TM-score against the PDB dataset, and the reported overall Novelty is the mean of these per-sample maxima.

Secondary Structure Composition ($\alpha/\beta/c$). Secondary structure is assigned from decoded structures using DSSP on backbone atoms (Kabsch & Sander, 1983). We parse DSSP labels and map $H/G/I \to \alpha$-helix, $E/B \to \beta$-strand, and all others to coil.

Utilization Rate (UR). UR measures the fraction of codebook entries that are used at least once over the evaluation set. Given a codebook of size K and token counts $\{n_k\}_{k=1}^{K}$, we compute

$$\mathrm{UR} = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}(n_k > 0).$$

Normalized Perplexity (PPL/K). Let $p_k = n_k / \sum_{j=1}^{K} n_j$ denote the empirical frequency of code k. We first compute the code usage perplexity as

$$\mathrm{PPL} = \exp\left(-\sum_{k=1}^{K} p_k \log p_k \right),$$

and report its value normalized by the codebook size, PPL/K. A value closer to one indicates more balanced code usage.

Codebook Similarity. To measure redundancy among codebook entries, we compute the maximum cosine similarity between each codebook vector and any other distinct codebook vector, and then average these values over all entries. Lower similarity indicates a more diverse codebook with less redundancy.

LFQ does not maintain an explicit embedding matrix. Instead, each code corresponds to a binary vector in the implicit codebook $\mathcal{C}_{\mathrm{LFQ}} \subseteq \{-1, +1\}^d$ (Yu et al., 2024). We reconstruct these binary vectors from the code indices and compute codebook similarity using the same nearest-neighbor cosine-similarity metric:

$$\mathrm{Sim}(\mathcal{C}_{\mathrm{LFQ}}) = \frac{1}{|\mathcal{C}_{\mathrm{LFQ}}|} \sum_{\mathbf{c} \in \mathcal{C}_{\mathrm{LFQ}}} \max_{\mathbf{c}' \neq \mathbf{c}} \frac{\mathbf{c}^\top \mathbf{c}'}{\|\mathbf{c}\|_2 \|\mathbf{c}'\|_2}.$$

B Experiment Details

B.1 Training Data Preprocessing

All training samples are drawn from the representative sequences of each cluster in the AFDB Foldseek cluster dataset. We retain only sequences with length ≤ 256 and pLDDT ≥ 75. In addition,

all homologous sequences overlapping with the test sets are removed, using a Foldseek structural similarity threshold of 80%.

Additional data augmentation during training includes centering each structure at the origin and applying random rotations. We do not perform any additional finetuning on longer sequences.

B.2 Experimental Configuration

For the ProFiT checkpoint used in our experiments, we train the model on 8 H20 GPUs with a per-GPU batch size of 8, resulting in a total effective batch size of 64. Training is conducted for 120k steps and takes approximately 20 hours in wall-clock time. Tab. 6 summarizes the hyperparameter search spaces used during training. For hyperparameters that were not extensively tuned, we report the values used in our experiments directly. For hyperparameters explored through grid search, we use **boldface** to indicate the final selected values.

Table 6: Hyperparameters for ProFiT.

Hyperparameter	Value
Train/validation split	$0.999/0.001$
Encoder layers	5
Decoder layers	5
Attention heads	8
Token dimension	256
Latent dimension, d	8
Codebook size, K	8192
VQ EMA decay	0.99
Dead-code threshold, τ_{dead}	0.5 / 1 / **2** / 4
Dead-code reset mass, ρ	5 / 10 / **20** / 40
Maximum token masking ratio, r	**0.1** / 0.2 / 0.4
Optimizer	Adam(0.9, 0.999)
Learning rate	5e-3 / 1e-3 / 5e-4 / **1e-4** / 5e-5
Precision	bf16 mixed
Model EMA decay	0.999

B.3 Details of the AR Model

For the unconditional generation experiments in Sec. 5.2, we train a Transformer model for autoregressive modeling of protein structure tokens. BOS and EOS tokens are appended to the beginning and end of each sequence, respectively, and PAD tokens are used for padding within a batch. The Transformer consists of 16 self-attention blocks with a hidden dimension of 1280 and 16 attention heads, using Pre-LayerNorm and GELU activations. The FFN expansion ratio is set to 2, with a dropout rate of 0.1. The input token embedding and output projection head share weights.

Training is performed on exactly the same dataset used for tokenizer training. We use the Adam optimizer with a learning rate of 3×10^{-4}, betas $(0.9, 0.98)$, weight decay 0.01, and 2000 warmup steps. The model is trained for a total of 1k epochs.

C Extra Experiment Result

C.1 Analysis of Kanzi's Length Generalization

To investigate the discrepancy between our Kanzi results and those reported in the original paper, we evaluate Kanzi separately on proteins within and beyond its training truncation length of 256 residues. The results are summarized in Tab. 7.

Kanzi performs well on proteins within the training truncation length but degrades sharply on longer proteins, indicating limited length generalization. This observation provides a plausible explanation for the discrepancy between our results and those reported in the original Kanzi paper. Specifically,

Table 7: Length-stratified performance of Kanzi. Each entry reports RMSD (Å) / TM-score.

Length regime	CAMEO	CASP15	CASP16
≤ 256 residues	0.993 / 0.940	0.932 / 0.959	1.341 / 0.939
> 256 residues	20.63 / 0.320	36.06 / 0.223	22.91 / 0.343

although we use the correct model weights, the preprocessing applied to the test sets may differ from that used in the original evaluation. The original paper does not provide sufficient details to determine the exact preprocessing procedure. When we truncate the test proteins to 256 residues, we recover performance consistent with the original report, further supporting this explanation.

The same truncation length is used during the training of ProFiT . In contrast, ProFiT maintains nearly unchanged reconstruction quality on the untruncated test sets, as shown in Fig. 4. These results suggest that ProFiT generalizes substantially better beyond its training length.

C.2 Out-of-Distribution Shift Across Evaluation Sets

Fig. 4 (right) shows that the reconstruction quality of several baselines degrades substantially as the evaluation sets shift further from the PDB distribution. To contextualize this trend, we include Protenix v1 as a folding-model baseline in Tab. 8, using MSA input and a single generated sample (Team et al., 2026). Protenix exhibits a similar decline with increasing OOD severity, whereas the reconstruction quality of ProFiT remains relatively stable across evaluation sets. This contrast highlights the different behavior of a folding model's structural prior and ProFiT 's token-conditioned reconstruction objective on OOD structures.

Table 8: Mean TM-score of Protenix v1 across evaluation sets.

Metric	D_{FS}	CAMEO	CASP16	CASP15
Mean TM-score	0.9531	0.8609	0.7559	0.7502

D Ablation Study

D.1 Ablation on Design Choices

We conduct ablation studies on the design choices in Sec. 4.2, demonstrating that they play an important role in both reconstruction quality and semantic compression. These components jointly push the Pareto frontier of reconstruction and generation, rather than merely trading off between the two. As shown in Tab. 9, both components positively impact reconstruction and generation quality.

We further illustrate in Fig. 7 how our design choices alter the training dynamics. Both random token masking and dead code replacement promote better convergence. Although random token masking slightly increases the difficulty at the early stage of training, it creates a smoother loss landscape for optimization. We observe that without masking, the training loss can still decrease normally, but the reconstruction accuracy fluctuates severely and fails to converge to the optimum; with masking, the model exhibits stable and reproducible convergence behavior. Dead code replacement further strengthens this trend.

In addition, we find that the codebook usage of ProFiT during training follows a highly skewed long-tail distribution, which is unfavorable for downstream generation tasks. Dead code replacement effectively smooths the utilization across codes while still preserving an overall Zipf's law pattern.

D.2 Ablation on Sampling Steps

We examine in Fig. 8 whether increasing the number of sampling steps at test time benefits the quality of the output structures. When using ODE sampler, ProFiT follows a straight sampling trajectory, and thus the output quality changes little as the number of sampling steps increases.

Table 9: Ablation study on CAMEO2022. We report reconstruction RMSD and AR generation scRMSD.

Method	Recon. RMSD ↓	AR scRMSD ↓
ProFiT	**0.84**	**5.43**
- token masking	1.42	5.97
- code replacement	1.45	6.55

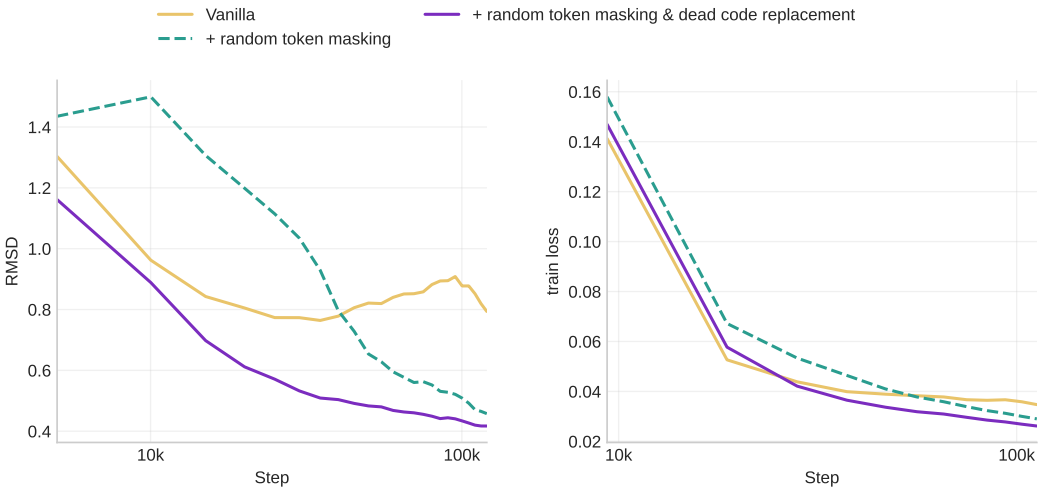

Figure 7: Ablation studies for Sec. 4.2.

When using SDE sampler ($\gamma = 0.5$), the output quality improves significantly with more sampling steps and already surpasses the ODE sampler at 20 steps. This is also the setting adopted in Sec. 5.1.

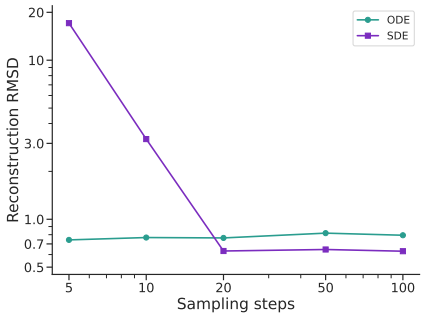

Figure 8: Performance under different numbers of sampling steps.

We furthur ablate the number of sampling steps in Tab. 10 under ODE sampling. ProFiT achieves strong reconstruction performance with as few as 5 steps. Increasing the step count to 20 yields a substantial improvement primarily on CASP15, the most strongly OOD test set. This trend can also be observed in Fig. 8, where the blue curve exhibits a slight upward trend.

The sampling-step ablation indicates that ProFiT achieves high reconstruction fidelity without relying on long sampling trajectories, although additional steps can improve performance on the out-of-distribution CASP15 set. This behavior reflects the role of its stochastic decoder: conditioned on the encoder's tokens, the decoder is trained to recover the input structure rather than to generate a structure from an unconditional protein prior. The strong reconstruction results suggest that the tokens retain sufficient information for this recovery. Moreover, the simple flow-matching objective requires few task-specific structural assumptions, which may help limit bias toward particular structural preferences or evaluation metrics. In this setting, stochasticity and additional sampling

steps support accurate conditional reconstruction; they do not necessarily imply a separate refolding stage.

Table 10: Effect of the number of ODE sampling steps on reconstruction performance. Each entry reports RMSD (Å) / TM-score.

Steps	CAMEO	CASP15	CASP16	D_{FS}
2	0.76 / 0.977	4.31 / 0.897	1.78 / 0.953	0.67 / 0.977
5	**0.68 / 0.981**	2.73 / 0.924	**1.34 / 0.968**	**0.49 / 0.985**
20	0.96 / 0.964	**1.48 / 0.954**	1.41 / 0.952	0.71 / 0.978

D.3 Ablation on Codebook Size

We ablate the effect of codebook size by retraining ProFiT with $K = 1000$ while keeping all other training and evaluation settings fixed, thereby matching Kanzi's vocabulary size (Dilip et al., 2026b). We repeat the experiments from Sec. 5.1, 5.2, and 5.3, and observe that the vocabulary size has no significant effect on the performance of ProFiT. The results are presented in Tab. 11, 12, 13, and 14.

Table 11: Codebook-size ablation for reconstruction. Each entry reports RMSD (Å) / TM-score.

Method	CAMEO	CASP15	CASP16	D_{FS}
Kanzi	9.33 / 0.674	21.86 / 0.520	13.33 / 0.604	0.98 / 0.943
ProFiT ($K = 1000$)	0.78 / 0.978	1.38 / 0.964	1.80 / 0.969	0.50 / 0.984
ProFiT ($K = 8192$)	**0.64 / 0.983**	**1.19 / 0.967**	**1.01 / 0.977**	**0.42 / 0.988**

Table 12: Codebook-size ablation for unconditional generation with the AR architecture.

Method	Des. ↑	scRMSD ↓	scTM ↑	Div. ↓	Nov. ↓	α%	β%	c%
Kanzi	0.63	12.38	0.429	0.289	0.721	–	–	–
ProFiT ($K = 1000$)	**0.82**	5.80	0.710	0.251	0.712	0.457	0.115	0.428
ProFiT ($K = 8192$)	**0.82**	**5.43**	**0.729**	0.262	0.714	0.466	0.106	0.428

D.4 Analysis of Computational Cost

We compare the computational costs of Kanzi and ProFiT for encoding and structure reconstruction. The results are summarized in Tab. 15. Although ProFiT uses a larger vocabulary, its reconstruction cost is substantially lower than that of Kanzi.The encoding cost of ProFiT is higher, reflecting differences in parameter count and encoder architecture; however, the decoder dominates the cost of multi-step reconstruction, where ProFiT is considerably more efficient.

Table 13: Codebook-size ablation for unconditional generation with the DPLM architecture.

Method	Des. ↑	scRMSD ↓	scTM ↑	Div. ↓	Nov. ↓	$\alpha\%$	$\beta\%$	c%
Kanzi	0.48	11.35	0.496	0.346	0.706	–	–	–
ProFiT ($K = 1000$)	0.79	6.74	0.660	0.299	0.654	0.621	0.072	0.307
ProFiT ($K = 8192$)	**0.81**	**6.25**	**0.685**	0.298	0.655	0.617	0.067	0.316

Table 14: Codebook-size ablation for conditional folding.

Method	RMSD (Å) ↓	TM-score ↑
Kanzi	17.01	0.351
ProFiT ($K = 1000$)	8.96	0.750
ProFiT ($K = 8192$)	**8.31**	**0.787**

Table 15: Computational cost of Kanzi and ProFiT.

Model	Encoding	1-step reconstruction	20-step reconstruction
ProFiT	10.93 GFLOPs	**42.07 GFLOPs**	**0.633 TFLOPs**
Kanzi	**2.98 GFLOPs**	183.5 GFLOPs	2.928 TFLOPs