All-Atom Protein Generation with Latent Diffusion

Amy X. Lu [1 2] **Wilson Yan** [1] **Sarah A. Robinson** [2] **Simon Kelow** [2] **Kevin K. Yang** [3] **Vladimir Gligorijevic** [2]
Kyunghyun Cho [2 4] **Richard Bonneau** [2] **Pieter Abbeel** [1] **Nathan C. Frey** [2]

Abstract

While generative models hold immense promise for protein design, existing models are typically backbone-only, despite the indispensable role that sidechain atoms play in mediating function. As prerequisite knowledge, all-atom 3D structure generation require the discrete sequence to specify sidechain identities, which poses a multimodal generation problem. We propose **PLAID** (Protein Latent Induced Diffusion), which samples from the *latent space* of a pre-trained sequence-to-structure predictor, ESMFold. The sampled latent embedding is then decoded with frozen decoders into the sequence and all-atom structure. Importantly, **PLAID only requires sequence input during training**, thus augmenting the dataset size by 2-4 orders of magnitude compared to the Protein Data Bank. It also makes more annotations available for functional control. As a demonstration of annotation-based prompting, we perform compositional conditioning on function and taxonomy using classifier-free guidance. Intriguingly, function-conditioned generations learn active site residue identities, despite them being non-adjacent on the sequence, *and* can correctly place the sidechains atoms. We further show that PLAID can generate transmembrane proteins with expected hydrophobicity patterns, perform motif scaffolding, and improve unconditional sample quality for long sequences. Links to model weights and training code are publicly available at `github.com/amyxlu/plaid`.

1. Introduction

Generative models of proteins have great potential for bioengineering by proposing designs with desired functions. These functions are mediated by their structure, including both sidechain and backbone atoms, collectively known as their *all-atom* structure.

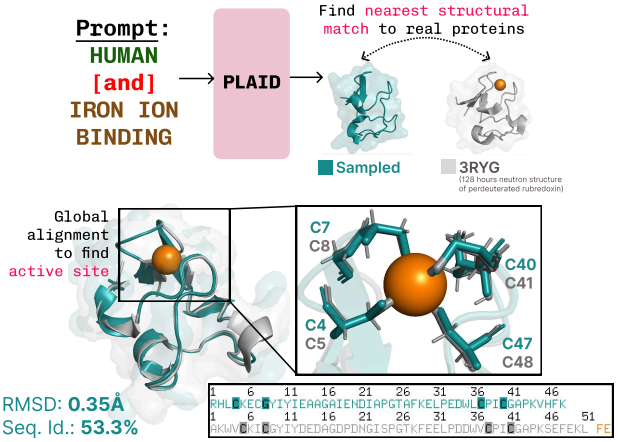


Figure 1. **PLAID learns catalytic residues and sidechain positions at active sites.** Conserved active site residues are *not* adjacent on the sequence, and the global sequence identity is low, suggesting that the model is learning rather than memorizing. To analyze active sites, we find the closest structural match in the PDB that was resolved in complex with a ligand, and overlay it with the generation.

The typical workflow for *de novo* protein design involves specifying a function, and generating a corresponding structure and sequence in two stages (Chu et al., 2024). Most methods either generate a structure backbone, and then predict a sequence that will fold into it, or generate a sequence, and predicting its structure. Structure-based generation can provide more precise control when a function can be encapsulated as a motif. However, structure databases are limited in size, and biased towards crystallizable proteins (Berman et al., 2000). Sequence-based methods can have better coverage of the protein distribution, but provide a coarser representation of many protein functions. A natural way to combine the benefits of both approaches would be to simultaneously reason about sequence and structure during generation, known as the co-generation[1] or all-atom generation problem (Wang et al., 2024). However, to know

[1]UC Berkeley [2]Prescient Design, Genentech [3]Microsoft Research [4]New York University. Correspondence to: Amy X. Lu <amyxlu@berkeley.edu>.

Preprint.

[1]Co-generation is sometimes used for methods that simultaneously generate structure and sequence, but do not specify sidechain atom positions. We therefore use *all-atom generation* throughout to minimize confusion.

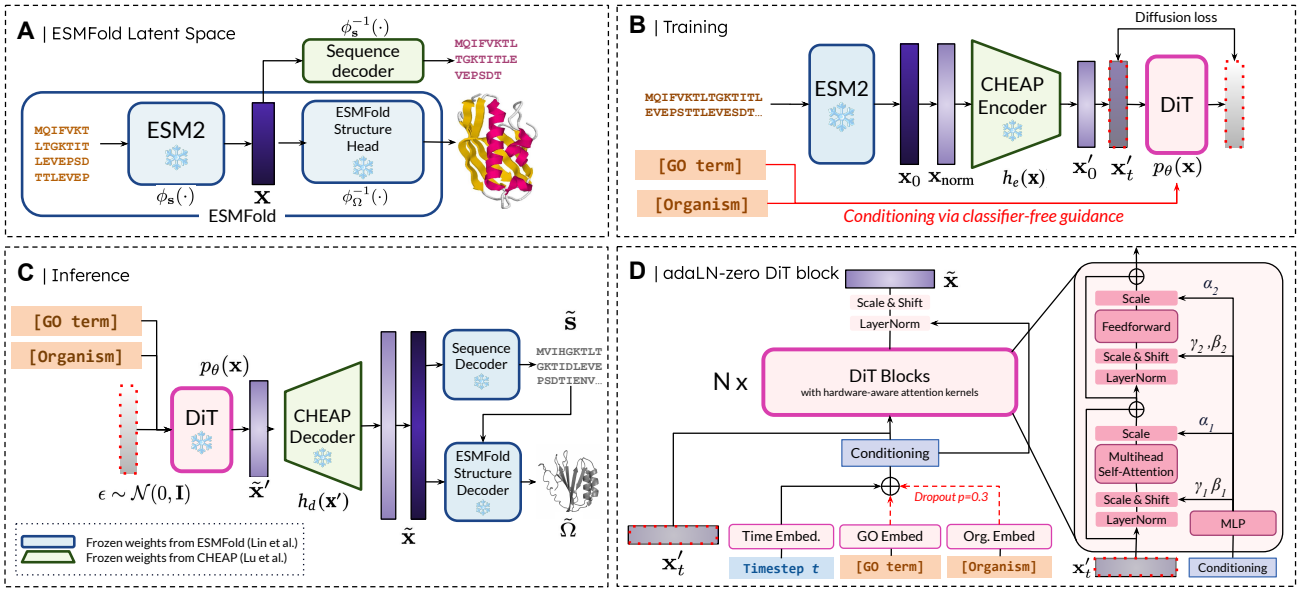


Figure 2. Overview of PLAID. **(A) ESMFold latent space.** We consider the latent embedding $\mathbf{x}$ to be a joint representation of sequence and structure. **(B) Latent diffusion training.** We learn a diffusion model to sample from $p_\theta(\mathbf{x})$. To decrease input "resolution", we compress the embedding to obtain $\mathbf{x}' = h_e(\mathbf{x})$ before diffusion training. **(C) Inference.** We generate a latent embedding $\tilde{\mathbf{x}}' \sim p_\theta(\mathbf{x}')$, and uncompress it back to $\tilde{\mathbf{x}} = h_d(\tilde{\mathbf{x}}')$. To obtain the sequence, we use a frozen decoder that was separately trained to map the sequence embedding $\mathbf{x}$ back to sequence space. To obtain structure, we "reattach" the ESMFold structure decoder head, which provides the all-atom structure. **(D) Architecture and Conditioning.** PLAID uses the Diffusion Transformer (DiT) architecture with adaLN-zero blocks to incorporate conditioning information. Classifier-free guidance is used to incorporate both the function (i.e., GO term) and organism class label embeddings. DiT block schematic adapted from Peebles and Xie (2023).

which sidechain atoms to place, one must first know the *sequence*. All-atom structure generation thus can be seen as a multimodal problem requiring simultaneous generation of discrete sequences and continuous structure, and is a challenging problem.

Contributions To address these challenges, we introduce **PLAID (Protein Latent Induced Diffusion)**. Sampling from the latent space of a pretrained protein folding model (Lin et al., 2023) combines the function resolution afforded by structure with the distribution coverage of sequence databases. By making use of the rich textual annotation data available to us, we demonstrate an alternative means to achieve highly specific control over function.

More broadly, our work demonstrates that multimodal generation can be achieved by learning the latent space of a predictor from a more abundant data modality (e.g., sequence) to a less abundant one (e.g., crystal structure). Sampling latent representations of structure allows us to directly use infrastructure built for other applications, such as diffusion transformers (Peebles and Xie, 2023) and accelerated attention kernels (Dao et al., 2022; Lefaudeux et al., 2022). Our work is fully open-source, including training code and model weights, available at `github.com/amyxlu/plaid`.

2. PLAID: Protein Latent Induced Diffusion

2.1. Notation

A protein is composed of amino acids. A protein sequence $\mathbf{s} := \{r_i\}_{i=1}^L$ with length L is often represented as a string of characters, with each character denoting the identity of an amino acid residue $r_i \in \mathcal{R}$, where $|\mathcal{R}| = 20$. Each unique residue r can be mapped to a set of atoms $\mathbf{r} := \{\mathbf{a}_j\}_{j=1}^{M_r}$, where each $\mathbf{a}_j \in \mathbb{R}^3$ is the 3D coordinate of an atom, and the number of atoms M_r may differ depending on the residue identity. A protein structure $\Omega := \{\mathbf{r}_i\}_{i=1}^L$ consists of all atoms in the protein and therefore implicitly contains $\mathbf{s}$.[2] To reduce complexity, protein designers sometimes work only with backbone atoms $\Omega_{\text{backbone}} \subset \Omega$ only, which consists of $[C_\alpha, C, N]$ atom repeats. These are sufficient to define the general fold.

[2]In practice, to make use of array broadcasting, a standard M is selected for all residues, with an associated one-hot mask to specify which atoms are present for a given residue. Following prior work (Ahdritz et al., 2024; Lin et al., 2023), we use the `atom14` representation where $M = 14$.

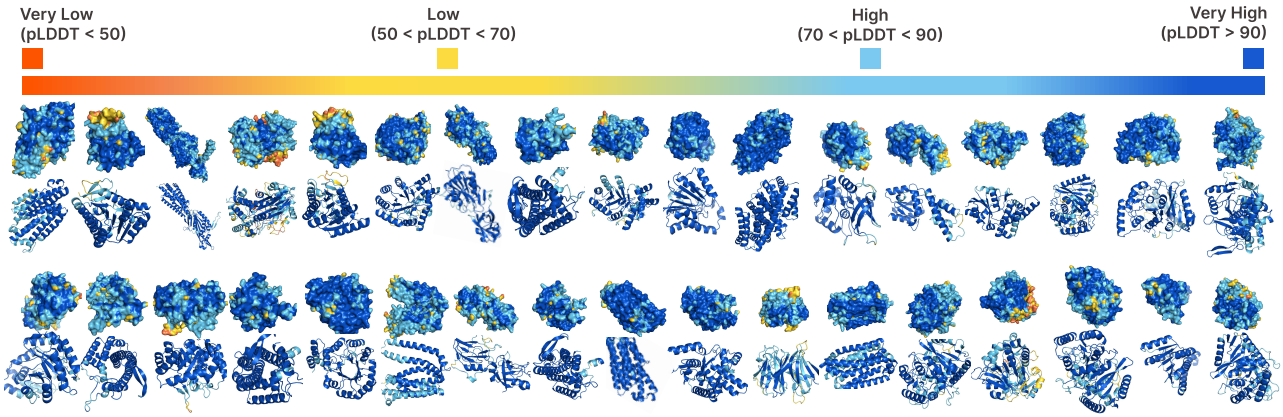


Figure 3. **PLAID unconditionally generates diverse, high-quality all-atom structures**, despite using only sequences for training the generative model.

2.2. Defining a Joint Distribution of Sequence and Structure

Our goal is to characterize a joint embedding of structure and sequence information $p(\mathbf{x})$ over $\mathcal{X}$, such that there exist mappings $\mathbf{x} = \phi_{\mathbf{s}}(\mathbf{s})$ and $\mathbf{x} = \phi_{\Omega}(\Omega)$. To do so, we use the latent space of protein folding models, allowing us to repurpose information from these *predictive* models for *generation*. The trunk of the model provides $\mathbf{x} = \phi_{\text{ESM}}(\mathbf{s})$, and the structure module head provides $\Omega = \phi_{\text{SM}}(\mathbf{x})$. If we consider an implicit inverse function of the structure module such that $\mathbf{x} = \phi_{\text{SM}}^{-1}(\Omega)$, then we have $\mathbf{x} = \phi_{\mathbf{s}}(\mathbf{s}) = \phi_{\Omega}(\Omega)$, and $\mathbf{s}$ and Ω can be mapped to the same $\mathbf{x}$.

Overview of ESMFold Briefly, ESMFold (Lin et al., 2023) has two components: a protein language model component $\mathbf{x} = \phi_{\text{ESM}}(\mathbf{s})$, and a structure module component $\Omega = \phi_{\text{SM}}(\mathbf{x})$ that decodes these latent embeddings into a 3D structure. For $\mathbf{x} \in \mathbb{R}^{L \times 1024}$, we use the layer outputs representation just prior to the structure module (pseudocode provided in Appendix C). This is because when ESMFold is used at inference time, the pairwise input is initialized as zeros, such that the sequence input contains all information necessary for structure prediction (Figure 2A).

2.3. Sampling All-Atom Structure

Latent Generation Our goal is to learn $p_\theta(\mathbf{x}) \approx p(\mathbf{x})$, where θ is the set of parameters of the model learned through diffusion training (Figure 2C). Then, after training, we can sample $\tilde{\mathbf{x}} \sim p_\theta(\mathbf{x})$ (Figure 2C). To do so, we use diffusion models (Ho et al., 2020; Song et al., 2020) with some modifications (described in ablation Table 1).

Classifier-Free Guidance Classifier-free guidance (CFG) (Ho and Salimans, 2022) is used to condition the model (Figure 2D). As an overview of classifier-free guidance, we first consider the case where we use only one conditioning label, $\mathbf{c}$. In CFG an additional "unconditional class" $\varnothing$ is defined. During training, we replace $\mathbf{c}$ with $\varnothing$ with some dropout probability p_{uncond}, such that we alter $\tilde{p}_\theta(\mathbf{x}|\mathbf{c})$ to $\tilde{p}_\theta(\mathbf{x}|\varnothing)$. The effect is that $\tilde{p}_\theta(\cdot)$ can be used as both an conditional and unconditional model. Then, inference time, guided generations can be obtained as:

$$\tilde{p}_\theta(\mathbf{x}_t, \mathbf{c}) = (1 + w) \cdot p_\theta(\mathbf{x}_t, \mathbf{c}) - w \cdot p_\theta(\mathbf{x}_t),$$

where increasing w strengthens the conditioning effect. To perform unconditional sampling, one can use the $\varnothing$ variable during inference.

In our case, we apply *compositional conditioning*, such that we have $\mathbf{c}_{\text{function}}$ and $\mathbf{c}_{\text{organism}}$. For $\mathbf{c}_{\text{function}}$, we use 2,219 Gene Ontology (GO) (Consortium et al., 2023; Ashburner et al., 2000) terms, which is a structured hierarchical vocabulary for annotating gene functions, biological processes, and cellular components across species. For $\mathbf{c}_{\text{organism}}$, we examine all unique organisms in our dataset, and identify 3,617 organisms. During training, p_{uncond} is sampled separately for $\mathbf{c}_{\text{function}}$ and $\mathbf{c}_{\text{organism}}$, and we use $p_{\text{uncond}} = 0.3$ for both. At inference, we can condition by one variable but not the other by replacing $\mathbf{c}_{\text{organism}}$ with $\varnothing$ while keeping $\mathbf{c}_{\text{function}}$ as is (and vice versa).

Decoding Sequence and Structure To obtain the sequence, we need an implicit inverse mapping of ESM2 to get $\tilde{\mathbf{s}} = \phi_{\text{ESM}}^{-1}(\tilde{\mathbf{x}})$. It is easy to train this mapping from a sequence embedding back to the one-hot encoded space, and the sequence decoder ϕ_{ESM}^{-1} achieves a validation accuracy of 99.7% on a held-out partition of UniRef (Suzek et al., 2015). To obtain the all-atom structure from the sampled latent embedding, we use the frozen ESMFold structure module

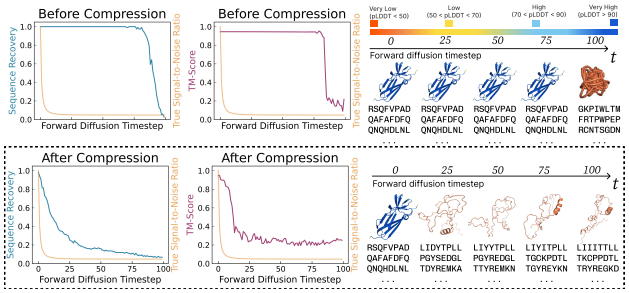


Figure 4. When noise is added via a cosine schedule (Dhariwal and Nichol, 2021) (true signal-to-noise ratio (SNR) curve overlaid in orange) to the uncompressed latent space $\mathbf{x}$, the sequence and structure remain uncorrupted until the final forward diffusion timesteps, meaning most sampled timesteps are trivial for learning. After compression, noising in the compressed latent space $\mathbf{x}' = h_e(\mathbf{x})$ better aligns with the true SNR in the sequence and structure space, thereby improving the effectiveness of the diffusion task.

weights to compute $\tilde{\Omega} = \phi_{\text{SM}}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}})$ (Figure 2C). Note that $\tilde{\mathbf{s}}$ must be decoded first, as it determines the side-chain atoms to be placed in $\tilde{\Omega}$.

2.4. Latent Space Compression

In initial experiments, we found that directly learning $p_\theta(\mathbf{x})$ without compression performed poorly (results shown in Appendix Figure 11). We suspected that this might be due to the high dimensionality of $\mathbf{x} \in \mathbb{R}^{L\times 1024}$. For proteins with length $L = 512$ (the length cutoff used in this work), this corresponds to a high-resolution image synthesis problem similar to those encountered in image diffusion literature. Therefore, we mirror works in this literature and perform diffusion in the latent space of an autoencoder, $\mathbf{x}' = h_e(\mathbf{x})$, such that the dimensions of $\mathbf{x}'$ are much smaller (Rombach et al., 2022).

Compression is performed using the CHEAP autoencoder (Lu et al., 2024), using a compressor that compresses from $1024 \rightarrow 32$ channels and downsamples by $2\times$ along the length (for memory efficiency when training the transformer-based diffusion model). The task is now to learn $p_\theta(\mathbf{x}') \approx p(\mathbf{x}')$, where $\mathbf{x}' = h_e(\mathbf{x})$ (Figure 2B). At inference time, we begin by sampling the compressed latent variable $\tilde{\mathbf{x}}' \sim p_\theta(\mathbf{x}')$, decompress to obtain $\tilde{\mathbf{x}} = h_d(\tilde{\mathbf{x}}')$, then use frozen decoders as previously described. More information on CHEAP can be found in Lu et al. (2024) and Appendix D.

Compressing the Latent Space Regularizes Pathologies Figure 4 shows that when noise is added to the latent space, the sequence and structure remain unaltered until later timesteps. By adding noise in the compressed latent space, the resulting corruptions in sequence and structure space more closely match the true signal-to-noise ratio. Applying embedding compression prior to learning the latent diffusion model was crucial for performance in our experiments. Considering the importance of diffusion noise schedules for sample performance (Chen, 2023), we suspect these irregularities in how perturbations in the latent space is mapped back to sequence and structure space might be contributing to this empirical observation.

Table 1. Ablation results for metrics defined in Section 3.

	Configuration	ccTM	scTM	Ppl.	Seq. Div. %	Struct. Div.%
A	cosine noise sched. & pred. noise	0.54	0.55	16.97	**0.98**	0.86
B	A + v-diffusion	0.52	0.53	17.37	**0.98**	**0.89**
C	A + MinSNR	0.59	0.59	16.76	0.97	0.86
D	A +B + C + sigmoid noise sched.	0.56	0.58	16.88	0.92	0.86
E	D + self-conditioning	**0.70**	**0.65**	**15.38**	0.93	0.76
F	E + no cond drop	0.57	0.57	17.28	0.97	0.85

2.5. Training

Architecture We use the Diffusion Transformer (DiT) (Peebles and Xie, 2023), with the goal of maximizing portability and weight reuse. In early experiments, we found that allocating available memory to a larger DiT model was more beneficial than using triangular self-attention (Jumper et al., 2021). We train our models using the xFormers (Lefaudeux et al., 2022) implementation of (Rabe and Staats, 2021), which provided a 55.8% speedup and a 15.6% reduction in GPU memory usage during our inference-time benchmarking experiments compared to a standard implementation using PyTorch primitives (see Appendix Table 6). We train two versions of the model with 100 million and 2 billion parameters, respectively, both for 800,000 steps. More details are provided in Appendix B.

Diffusion Hyperparameters We use the discrete-time diffusion framework in Ho et al. (2020) with employing 1,000 timesteps. To stabilize training and improve performance, we incorporate additional strategies: min-SNR reweighting (Hang et al., 2023), v-diffusion (Lin et al., 2024; Salimans and Ho, 2022), self-conditioning (Chen et al., 2022; Jabri et al., 2022), a sigmoid noise schedule (Chen, 2023), and exponential moving average (EMA) decay. Ablation results are shown in Table 1.

For sampling, unless otherwise noted, all results use the DDIM sampler (Dhariwal and Nichol, 2021; Song et al., 2020) with 500 timesteps. We use $c = 3$ as the conditioning strength for conditional generation; however, we find (Appendix Figure 18C) that sample quality is not strongly affected by this hyperparameter. We also find that DPM-Solvers (Lu et al., 2022) can achieve comparable results with $10\times$ fewer steps in scenarios where speed is a concern (see Appendix Figure 18), but in this work, we prioritize sample quality. More details on sampling methodology can be found in Appendix F. A comparison of sampling speeds

across all-atom baselines can be found in Appendix H.

3. Evaluation Metrics

3.1. Unconditional Generation

Evaluating protein generative model outputs is notoriously difficult since humans cannot natively interpret what makes for a "good" protein. This is especially difficult for all-atom generation, because backbone-only structure metrics are not always meaningful in this setting. A schematic of these metrics are shown in Figure 5. For distributional conformity scores, each biophysical property is outlined in greater detail in Appendix G.1. **Metrics are described in detail in Appendix Table 4**.

3.2. Conditional Generations

Evaluation of conditional generations introduces a conundrum where structure and sequence should be highly conserved when they exhibit similar functions, and thus similarity to known proteins should indicate successful conditioning. However, in machine learning literature, works often prefer generations being *different* from known proteins to emphasize novelty or lack of training data memorization.

In our case studies, we look for high structure similarity as a proxy for function, and low global sequence similarity to account for memorization. In Appendix Figures, we examine the Sinkhorn distance between function-conditioned generated latent embeddings and held-out real proteins with this GO term annotation, to examine distribution similarities between real and generated samples. To compare, we also examine the Sinkhorn distance between *random* proteins from the heldout set of real samples.

4. Experiments

4.1. Unconditional Generation

PLAID balances quality and diversity at longer sequence lengths. Following prior work demonstrating the effect of protein length on performance (Watson et al., 2023; Chu et al., 2023; Campbell et al., 2024), we sample 64 proteins for each protein length between $\{64, 72, 80, ..., 512\}$, for a total of 3648 samples. As shown in Figure 6, the degradation in PLAID's performance with increasing protein length is less pronounced than baselines.

Baseline methods exhibit pronounced mode collapse at specific lengths. Figure 6A and Appendix Figure 13 shows that baselines methods exhibit mode collapse at specific lengths. Furthermore, secondary structure diversity is closer to the profile of natural proteins in PLAID versus baseline methods; Figure 6B shows that existing models often struggle to produce samples with high β-sheet content.

PLAID generations are more consistent across modalities. Table 2 compares consistency and quality metrics aggregated across lengths. Cross-modal consistency is generally highest for PLAID, possibly due to sampling directly from $p(\mathbf{s}, \Omega)$ (Table 2). When examining unimodal quality, ProteinGenerator achieves the best self-consistency TM-score and pLDDT. For sequence quality, all samples (including natural proteins) perform poorly on oracle-based metrics. PLAID comparably performs worse on scSR and perplexity under RITA XL (Hesslow et al., 2022); however, as shown in Appendix Figure 13, some baseline sequence samples contain high levels of repeats, which sometimes lead to lower perplexity, despite being less biologically natural. Distributional conformity metrics (Table 3) offer an alternative for assessing sequence quality by examining biophysical patterns rather than token-level likelihoods.

PLAID produces distinct, and designable samples. Table 3 assesses diversity, novelty, and naturalness. We examine the diversity and quality trade-off by comparing the number of distinct designable sequence and structure clusters, where designability is defined as ccRMSD < 2Å. Among all-atom models, PLAID produces the highest number of distinct and designable samples in both sequence and structure space. On novelty assessments, prior methods achieve higher scores, though it may be confounded by low-quality samples which typically have lower similarity to real proteins.

PLAID samples have high distributional conformity to natural proteins. Across molecular weight (MW), ratio of aromatic amino acids, dipeptide-based instability index (Guruprasad et al., 1990), hydropathy index (GRAVY) (Kyte and Doolittle, 1982), isoelectric point, and charge at pH $= 7$, the distribution of biophysical features for PLAID generations is closer to that of natural proteins, potentially due to the removal of biases toward structure in its training data. We consider distributional conformity to be an additional axis of real-world expressibility, as shown in Frey et al. (2023). More information can be found in Appendix G.1.

4.2. Conditional Generations

PLAID recapitulates active site sidechains. Figure 7 and Figure 1 demonstrate that function-conditioned proteins can recapitulate known protein active sites. For a given generation, we perform a Foldseek search to find the closest structural neighbor resolved in complex with a ligand, then overlay the structures. Despite high levels of conservation at catalytic sites, global sequence diversity is low, suggesting that the model has learned key biochemical features associated with the function prompt without direct memorization.

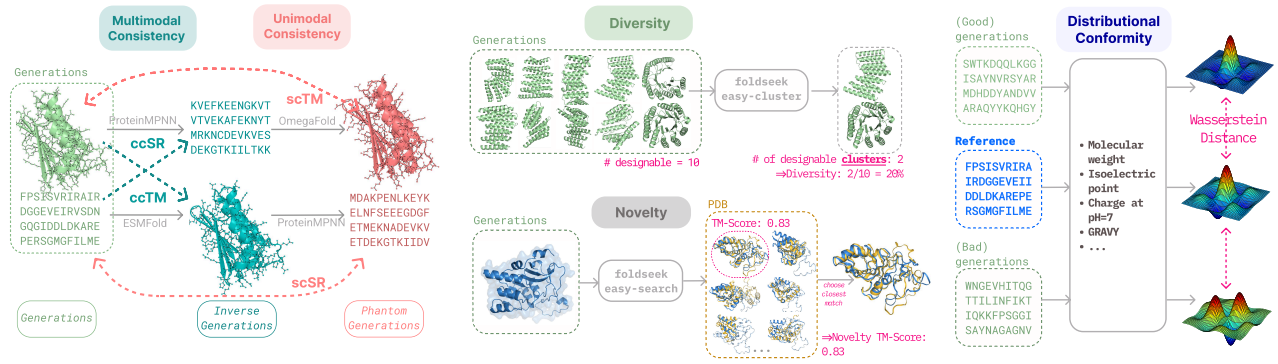


Figure 5. Schematic of metrics used to assess quality, diversity, and novelty. Quality is assessed via **multimodal and unimodal consistency metrics**. **Diversity** is the number of unique designable (i.e. ccRMSD < 2Å) samples. **Novelty** is the TMScore between the generation and its most similar known protein. **Distributional conformity** is the Wasserstein distance between biophysical attributes for generated proteins and a reference set of natural proteins. Sequence novelty and diversity are calculated similarly, but are omitted from the schematic for simplicity: we use `mmseqs easy-search` and `mmseqs easy-cluster` (Steinegger and Söding, 2017) instead of `foldseek` (van Kempen et al., 2022) to do sequence-level search/clustering, and novelty similarity is calculated as the sequence identity after pairwise alignment.

Table 2. Comparison of model performance across **consistency and quality metrics**. pLDDT refers to the confidence score directly returned by the structure trunk of the generative model; for models which do not produce a pLDDT metric, N/A is used.

	Cross-Modal Consistency				Structure Quality			Sequence Quality	
	ccTM (↑)	ccRMSD (↓)	ccSR (↑)	% ccRMSD < 2Å (↑)	scTM (↑)	pLDDT (↑)	Beta sheet % (↑)	scSR (↑)	Ppl. (↓)
ProteinGenerator	0.58	11.86	**0.28**	0.08	**0.72**	**69.00**	0.04	0.40	**8.60**
Protpardelle	0.44	24.28	0.22	0.00	0.57	N/A	0.11	**0.44**	8.86
PLAID	**0.69**	**9.47**	0.26	**0.32**	0.64	59.46	**0.13**	0.27	14.61
Natural	*1.00*	*0.07*	*0.39*	*1.00*	*0.84*	84.51	0.13	0.39	7.40

Table 3. **Diversity, novelty, and distributional conformity** metrics across models. Bold values show best performance among all-atom generation models. Descriptions of each biophysical parameter for distributional conformity is described in Appendix G.1.

	Diversity			Novelty		Distributional Conformity (Wasserstein Distance)					
	# Des. (↑)	# Des. Seq. Clusts. (↑)	# Des. Struct. Clusts. (↑)	MMseqs Seq Id% (↓)	Foldseek TMScore (↓)	MW (↓)	Aroma-ticity (↓)	Dipeptide Instability Index (↓)	Iso-electricity (↓)	Hydro pathy (↓)	Charge at pH=7 (↓)
ProteinGenerator	309	309	309	0.57	**0.57**	9.54	0.07	14.55	1.42	0.31	6.12
Protpardelle	0	0	0	**0.56**	0.72	10.4	0.07	8.61	1.99	0.37	8.58
PLAID	**1171**	**809**	**522**	0.60	0.67	**0.62**	**0.01**	**1.98**	**0.49**	**0.28**	**2.71**
Natural	*3570*	*1362*	*600*	*0.81*	*0.87*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*

Transmembrane protein generations have expected hydrophobicity patterns. Generations prompted by transmembrane functions consistency demonstrate hydrophobic cores and hydrophilic cytosol-facing residues. Generated G protein-coupled receptors (GPCRs) strcutures possess the characteristic seven transmembrane helix architecture. DeepTMMHMM (Hallgren et al., 2022) topology predictions on generated sequences confirm the expected transmembrane organization. This again highlights PLAID's ability to reason jointly about residue identity and position.

Motif scaffolding Figure 9 demonstrates how motif scaffolding can be used with PLAID, by holding parts of the motif constant at each reverse diffusion step during latent generation. Generations maintain the sequence motif, and is able to generate the all-atom structure scaffolding. However,

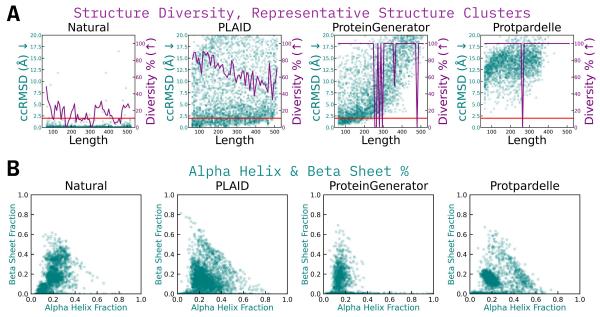


Figure 6. **(A)** Structural quality (ccRMSD, teal points) and diversity (purple line, measured as ratio of unique structural clusters to total samples) for proteins of varying lengths (64-512 residues). The red line indicates the ccRMSD < 2Å threshold for desiganbility. ProteinGenerator and Protpardelle show length-specific mode collapse and significantly decreased performance at longer sequence lengths. Analysis performed on 64 samples per length interval. **(B)** Comparing α-helix and β-sheet content in structural samples; each data point is an unique cluster, to account for sample diversity. PLAID generations have more diverse coverage of beta sheet regions than baselines.

we focus on enabling new axes of capabilities in this work, and leave further exploration as future work.

Taxonomic evaluations Appendix Figure 14 shows t-SNE plots of generated embeddings colored by organism. Organisms that are more distantly related phylogenetically, such as *Glycine max* (i.e., soybean) and *E. coli*, form more distinct clusters than those more closely related evolutionarily, such as human and mouse. This suggests that function- and organism-conditioned samples have been imbued with desired characteristics. This embedding-level analysis provides an early investigation into organism-specific conditioning abilities. We observe, for human organism conditioned samples in Figure 7 and Appendix Figure 15, that for 66.5% of samples, the closest `mmseqs easy-search` neighbor also come from *Homo sapiens*. We leave it as future work to do a more in-depth analysis of organism conditioning, since nearest-neighbor taxonomic analyses are very sensitive to search settings, and favors overrepresented species in the database.

Further evaluation Appendix Figure 13 visually inspects sequence repeats in baselines versus PLAID generations. Appendix Figure 14, shows that conditional generations generally have lower Sinkhorn distances to validation proteins with the same annotation than random samples, suggesting that the desired latent information is captured in the embedding. In Appendix Figure 18, we consider how conditioning scale might affect sample quality and possible GO term characteristics that might be influencing the difference in Sinkhorn distance between function-conditioned generations and random proteins.

5. Related Work

Single Modality Protein Generation Protein backbone structure generation aims to find novel folds that are distinct from known proteins. This is traditionally done by biophysical approaches, though recently, diffusion (Ho et al., 2020; Song et al., 2020) and flow-matching (Lipman et al., 2022) based approaches have been empirically successful (Ingraham et al., 2023; Watson et al., 2023; Wu et al., 2022a; Yim et al., 2023). Since these methods do not generate sequence, an important metric is whether if there exists a sequence which folds into the generated backbone, known as the *designability* (Trippe et al., 2023). Relatedly, hallucination approaches (Anishchenko et al., 2021; Wang et al., 2022) use discrete optimization or backpropagation through a structure prediction model to find sequences that fold into a desired structure.

Sequence-based generation typically train on large protein sequence datasets (Suzek et al., 2015; Mistry et al., 2021; Richardson et al., 2023) to recover the natural distribution. This can be especially useful for generating proteins without fixed structure, such as intrinsically disordered proteins. Most recent methods adopt autoregressive (Madani et al., 2023; Hesslow et al., 2022; Ferruz et al., 2022; Nijkamp et al., 2023), discrete diffusion (Alamdari et al., 2023; Gruver et al., 2024), or walk-jump sampling (Frey et al., 2023) approaches. The model itself does not produce atomic positions, though one can predict the structure using another model.

Multimodal Sequence-Structure Co-Generation Generative models which directly produce sequence and structure remains under-explored. Models which produce positions of both sidechains and backbones typically involve a two-stage process at each diffusion iteration. Chu et al. (2023) and Ingraham et al. (2023) produce all-atom positions, but the sequence must be predicted in a separate step. Lisanza et al. (2024) instead performs diffusion in the one-hot sequence space, and uses RosettaFold (Baek et al., 2021) to predict the structure. Multiflow (Campbell et al., 2024) generates sequence and structure in one step, but does not place the sidechain atoms.

Functionally Controllable Protein Structure Generation The central aim of protein design is to conjure proteins that fulfill a predefined function. Current structure generation methods typically achieve functional control by scaffolding around a motif. This falls short for enzymatic active sites, which often involve non-adjacent sidechain atoms. Kim et al. (2024) extends RFDiffusion (Watson et al., 2023) beyond backbone-level motifs to “scaffold" heavy-atoms without

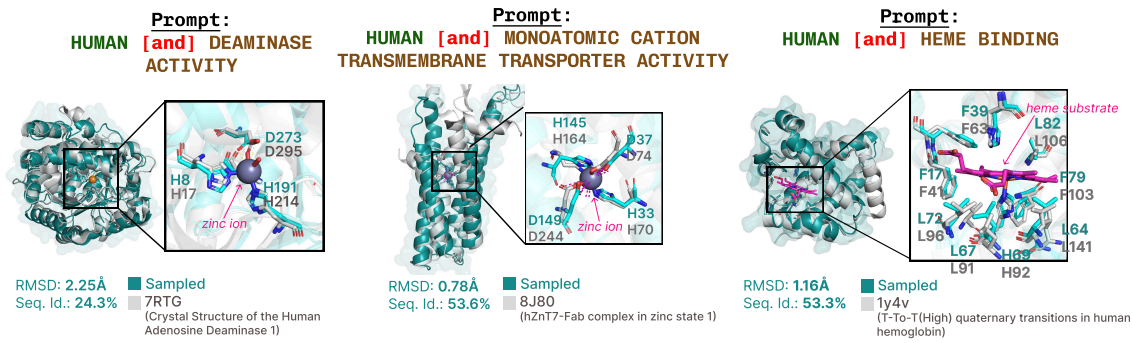


Figure 7. **PLAID enables function-guided protein generation while preserving critical structural motifs.** Additional examples provided in Appendix 15. **(A) Generated proteins capture sequence motifs and approximate side-chain orientations at active sites despite low sequence identity.** Each panel shows a PLAID-generated structure aligned with its closest PDB structural neighbor (identified via Foldseek) containing a bound ligand or substrate. RMSD and sequence similarity metrics are calculated globally.

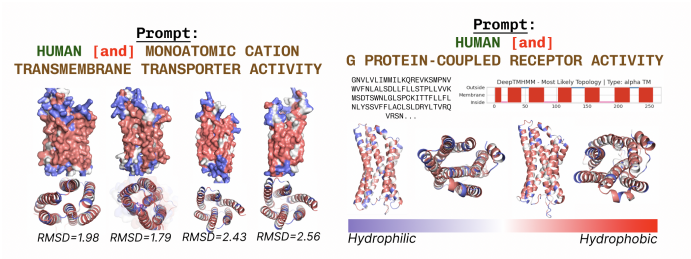


Figure 8. Transmembrane proteins have hydrophobic cores and hydrophilic ends. GPCRs have characteristic 7-helix topology.

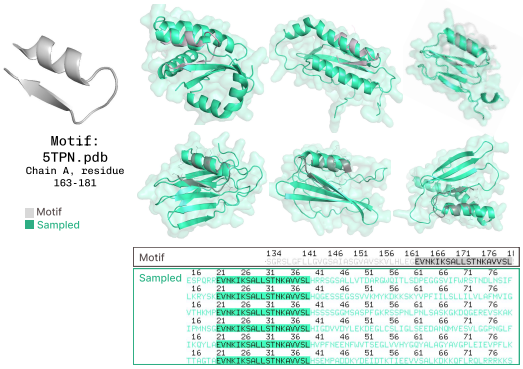


Figure 9. **Demo of motif scaffolding.** We use the same motif as the RFDiffusion (Watson et al., 2023) `design_motifscaffolding.sh` example for this experiment. The input motif is held constant at the user-prescribed location. Note that PLAID generates all-atom structure, whereas RFDiffusion does not position the sidechain atoms.

pre-specifying the positions of the catalytic residues. However, the motif must still be predefined, and the model does not produce sidechain positions for other parts of the protein. Development concurrently with this work, ESM3 (Hayes et al., 2024) provides similar axes of control by generating in a shared sequence-structure space, conditioned on InterPro (Paysan-Lafosse et al., 2023) function terms. However, the ESM3 tokenizer is trained on structure datasets rather than sequence databases.

6. Discussion

We propose PLAID, a paradigm for multi-modal, controllable generation of proteins by diffusing in the latent space of a prediction model that maps single sequences to the desired modality. It is straightforward to expand PLAID to many downstream capabilities. Although we examine ESMFold (Lin et al., 2023) in this work, the method can be applied to any prediction model. There is rapid progress (Krishna et al., 2024; Abramson et al., 2024; Discovery et al., 2024; Liu et al., 2024; Wohlwend et al., 2024) in predicting complexes from sequence, and diffusing in the latent space of such models would allow using the frozen decoder to obtain more modalities than just all-atom structure.

A limitation of PLAID is that performance can be bottlenecked by the prediction model from which the frozen decoders are derived. With explicit fine-tuning for latent generation (e.g., training CHEAP and the structure decoder end-to-end), model performance can likely be improved. Furthermore, since the current structure decoder is deterministic, it does not sample alternative conformations. A solution is to use a decoder that returns a distribution over structural conformations instead. Additionally, the GO term one-hot encoding used here does not take into account the hierarchical nature of the Gene Ontology vocabulary. This can be fixed by using a multi-class conditioning scheme instead. Finally, the classifier-free guidance scale can be separated for the organism and function conditions, since the two may require different guidance strengths in real-world use cases. These limitations will be examined in future work.

Impact Statement

Machine learning for protein modeling and design may have significant impacts on how we approach biology and important downstream applications, such as drug discovery and material engineering. The ability to specify function and directly generate protein sequence and structure can have enormous positive impact, though dual use concerns will require continuous monitoring.

References

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Alexander E Chu, Tianyu Lu, and Po-Ssu Huang. Sparks of function by de novo protein design. *Nature biotechnology*, 42(2):203–215, 2024.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

Chentong Wang, Sarah Alamdari, Carles Domingo-Enrich, Ava Amini, and Kevin K Yang. Towards deep learning sequence-structure co-generation for protein design. *arXiv preprint arXiv:2410.01773*, 2024.

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, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xFormers: A modular and hackable Transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

Gustaf Ahdritz, Nazim Bouatta, Christina Floristean, Sachin Kadyan, Qinghui Xia, William Gerecke, Timothy J O'Donnell, Daniel Berenberg, Ian Fisk, Niccolò Zanichelli, et al. OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature Methods*, pages 1–11, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

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

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

TGO Consortium, Suzi A Aleksander, James Balhoff, Seth Carbon, JM Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, and Nomi L Harris. The gene ontology knowledgebase in 2023. *Genetics*, 224(1): iyad031, 2023.

Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Amy X Lu, Wilson Yan, Kevin K Yang, Vladimir Gligorijevic, Kyunghyun Cho, Pieter Abbeel, Richard Bonneau, and Nathan Frey. Tokenized and continuous embedding compressions of protein sequence and structure. *bioRxiv*, pages 2024–08, 2024.

Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Markus N Rabe and Charles Staats. Self-attention does not need O(n2) memory. *arXiv preprint arXiv:2112.05682*, 2021.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via Min-SNR weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7441–7451, 2023.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5404–5411, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

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

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 620:1089–1100, 2023.

Alexander E Chu, Lucy Cheng, Gina El Nesr, Minkai Xu, and Po-Ssu Huang. An all-atom protein generative model. *bioRxiv*, 2023.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

Daniel Hesslow, Niccoló Zanichelli, Pascal Notin, Iacopo Poli, and Debora Marks. RITA: a study on scaling up generative protein sequence models. *arXiv preprint arXiv:2205.05789*, 2022.

Kunchur Guruprasad, BV Bhasker Reddy, and Madhusudan W Pandit. Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. *Protein Engineering, Design and Selection*, 4 (2):155–161, 1990.

Jack Kyte and Russell F Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology*, 157(1):105–132, 1982.

Nathan C Frey, Daniel Berenberg, Karina Zadorozhny, Joseph Kleinhenz, Julien Lafrance-Vanasse, Isidro Hotzel, Yan Wu, Stephen Ra, Richard Bonneau, Kyunghyun Cho, et al. Protein discovery with discrete walk-jump sampling. *arXiv preprint arXiv:2306.12360*, 2023.

Jeppe Hallgren, Konstantinos D Tsirigos, Mads Damgaard Pedersen, José Juan Almagro Armenteros, Paolo Marcatili, Henrik Nielsen, Anders Krogh, and Ole Winther. DeepTMHMM predicts alpha and beta transmembrane proteins using deep neural networks. *BioRxiv*, pages 2022–04, 2022.

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

John B Ingraham, Max Baranov, Zak Costello, Karl W Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M Lord, Christopher Ng-Thow-Hing, Erik R Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, 2023.

Kevin E. Wu, Kevin K. Yang, Rianne van den Berg, James Y. Zou, Alex X. Lu, and Ava P. Amini. Protein structure generation via folding diffusion. *arXiv*, 2209.15611, 2022a.

Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

Brian L Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S Jaakkola. Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem. *The Eleventh International Conference on Learning Representations*, 11, 2023.

Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, et al. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, 2021.

Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Joseph L Watson, Karla M Castro, Robert Ragotte, Amijai Saragovi, Lukas F Milles, Minkyung Baek, et al. Scaffolding protein functional sites using deep learning. *Science*, 377(6604):387–394, 2022.

Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A Salazar, Erik LL Sonnhammer, Silvio CE Tosatto, Lisanna Paladin, Shriya Raj, Lorna J Richardson, et al. Pfam: The protein families database in 2021. *Nucleic acids research*, 49(D1):D412–D419, 2021.

Lorna Richardson, Ben Allen, Germana Baldi, Martin Beracochea, Maxwell L Bileschi, Tony Burdett, Josephine Burgin, Juan Caballero-Pérez, Guy Cochrane, Lucy J Colwell, et al. Mgnify: the microbiome sequence data analysis resource in 2023. *Nucleic Acids Research*, 51 (D1):D753–D759, 2023.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41:1099–1106, 2023.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13:4348, 2022.

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.

Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Alex Xijie Lu, Nicolo Fusi, Ava Pardis Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *bioRxiv*, pages 2023–09, 2023.

Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36, 2024.

Sidney Lyayuga Lisanza, Jacob Merle Gershon, Samuel WK Tipps, Jeremiah Nelson Sims, Lucas Arnoldt, Samuel J Hendel, Miriam K Simma, Ge Liu, Muna Yase, Hongwei Wu, et al. Multistate and functional protein design using rosettafold sequence space diffusion. *Nature biotechnology*, pages 1–11, 2024.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557): 871–876, 2021.

Donghyo Kim, Seth M Woodbury, Woody Ahern, Indrek Kalvet, Nikita Hanikel, Saman Salike, Samuel J Pellock, Anna Lauko, Donald Hilvert, and David Baker. Computational design of metallohydrolases. *bioRxiv*, pages 2024–11, 2024.

Tomas 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. *bioRxiv*, pages 2024–07, 2024.

Typhaine Paysan-Lafosse, Matthias Blum, Sara Chuguransky, Tiago Grego, Beatriz Lázaro Pinto, Gustavo A Salazar, Maxwell L Bileschi, Peer Bork, Alan Bridge, Lucy Colwell, et al. InterPro in 2022. *Nucleic acids research*, 51(D1):D418–D427, 2023.

Rohith Krishna, Jue Wang, Woody Ahern, Pascal Sturmfels, Preetham Venkatesh, Indrek Kalvet, Gyu Rie Lee, Felix S Morey-Burrows, Ivan Anishchenko, Ian R Humphreys, et al. Generalized biomolecular modeling and design with RoseTTAFold all-atom. *Science*, 384(6693):eadl2528, 2024.

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*, pages 1–3, 2024.

Chai Discovery, Jacques Boitreaud, Jack Dent, Matthew McPartlon, Joshua Meier, Vinicius Reis, Alex Rogozhnikov, and Kevin Wu. Chai-1: Decoding the molecular interactions of life. *bioRxiv*, pages 2024–10, 2024.

Lihang Liu, Shanzhuo Zhang, Yang Xue, Xianbin Ye, Kunrui Zhu, Yuxin Li, Yang Liu, Wenlai Zhao, Hongkun Yu, Zhihua Wu, et al. Technical report of HelixFold3 for biomolecular structure prediction. *arXiv preprint arXiv:2408.16975*, 2024.

Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, et al. Boltz-1: Democratizing biomolecular interaction modeling. *bioRxiv*, pages 2024–11, 2024.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

Martin Steinegger, Milot Mirdita, and Johannes Söding. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature methods*, 16(7):603–606, 2019.

Andre Cornman, Jacob West-Roberts, Antonio Pedro Camargo, Simon Roux, Martin Beracochea, Milot Mirdita, Sergey Ovchinnikov, and Yunha Hwang. The omg dataset: An open metagenomic corpus for mixed-modality genomic language modeling. *bioRxiv*, pages 2024–08, 2024.

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*, 378:49–56, 2022.

Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, et al. High-resolution de novo structure prediction from primary sequence. *bioRxiv*, 2022b.

JR Lobry and Christian Gautier. Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 escherichia coli chromosome-encoded genes. *Nucleic acids research*, 22(15):3174–3180, 1994.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Appendix

A. Metrics

Category	Modality	Metric	Abbrv.	Description
Multimodal Cross Consistency	Structure	cross-consistency RMSD	ccRMSD	When the generated sequence is folded, does it match the generated structure?
		cross-consistency TM-Score	ccTM	Similar to ccRMSD, but with TM-Score.
		Designability	ccRMSD < 2Å	What percentage of generated samples are designable?
	Sequence	cross consistency sequence recovery	ccSR	When the generated structure is inverse-folded into a sequence, does it match the generated sequence?
Unimodal Sample Quality	Structure	self-consistency TM-Score	scTM	When the generated structure is inverse-folded into a sequence, and the result is folded again with OmegaFold, is it consistent with the original generation?
		self-consistency RMSD	scRMSD	Similar to scRMSD, but with TM-Scores.
	Sequence	self-consistency sequence recovery	scSR	If we fold a generated sequence, then inverse-fold, does the output sequence match the original generation?
		Perplexity on RITA XL (Hesslow et al., 2022)	Ppl.	Do generated sequences have low perplexity on an autoregressive protein language model?
Diversity	Structure	Num. Foldseck clusters	Structure clusters	After clustering with Foldseck, what fraction of structure generations are unique?
	Sequence	Num. MM-seqs clusters	seq. clusters	After clustering with MMseqs, what fraction of sequence generations are unique?
Novelty	Structure	TM-Score to nearest Foldseck neighbor	Foldseck TMScore	Among designable samples, how similar are generated structures to the closest known protein?
	Sequence	Sequence identity to nearest MMseqs neighbor	MMseqs seq. id.	What is the sequence identity to the closest mmseqs easy-search neighbor in UniRef90 after pairwise alignment?

Table 4. Evaluation metrics for protein structure and sequence generation.

B. Additional Training Details

Training Details We train two variants of the model: a 2B version and a 100M version, both with the memory-efficient attention implementation in xFormers, using float32 precision. A learning rate of 1e-4 was used, with cosine annealing applied over 800,000 steps. The xFormers memory-efficient attention kernel requires input lengths to be a multiple of 4. Since we also apply an upsampling factor of 2, the actual inference length must be a multiple of 4. During training, the maximum sequence length we use is 512, based on the distribution of sequences in Pfam and a shortening factor of 2 based on results in Lu et al. (2024).

Conditioning Following Ho and Salimans (2022), with $p_{\text{uncond}} = 0.3$, the class label is replaced with the $\varnothing$ unconditional token. Note that not all data samples will have an associated GO term; we use the $\varnothing$ token for those cases as well. At inference time, to generate unconditionally (for either or both of function and/or organism), we use the $\varnothing$ token for conditioning.

C. Defining the Latent Space: Pseudocode

To better clarify how we obtain the latent representation from ESMFold (Lin et al., 2023), we provide a code sketch for the relationship of this representation and the rest of the ESMFold architecture: Note that `self.esm_s_combine` and `self.esm_s_mlp` are both trained end-to-end with loss objectives from the original ESMFold paper.

```
# self.esm_s_combine is a learnable weight
self.esm_s_combine = nn.Parameter(torch.zeros(self.esm.num_layers + 1))

# Forward pass through ESM2 language model.
# `esm_s` is a stack of representations from all layer outputs
# esm_s = torch.stack(
#   [v for _, v in sorted(res["representations"].items())], dim=2
# )
# Typically, when working with ESM2 LM embeddings, one would use
# embeddings from `res["representations"][final_layer_number]
esm_s, esm_z = self._compute_language_model_representations(esmaa)

# weigh representations from different layers
esm_s = (self.esm_s_combine.softmax(0).unsqueeze(0) @ esm_s).squeeze(2)

# Simple MLP for manipulating output representation
s_s_0 = self.esm_s_mlp(esm_s)

# Pairwise input is initialized to zero:
s_z_0 = s_s_0.new_zeros(B, L, L, self.cfg.trunk.pairwise_state_dim)

# Pointwise add embedding of amino acid
s_s_0 += self.embedding(aa)

########
# Use this representation for CHEAP experiments:
# return s_s_0
########

# In ESMFold code, this embedding is then used as input
# to the structure trunk. We also use this to
# reconstruct structure from the CHEAP embedding.

structure: dict = self.trunk(
    s_s_0, s_z_0, aa, residx, mask, no_recycles=num_recycles
)
```

D. CHEAP Compression Details

Briefly, the CHEAP encoder and decoder use an Hourglass Transformer (Nawrot et al., 2021) architecture that downsamples lengthwise, and downprojects the channel dimension, to create a bottleneck layer. The output of the encoder is our compressed embedding. The entire model is trained with the reconstruction loss $MSE(\mathbf{x}, \hat{\mathbf{x}})$. Results in Lu et al. (2024) show that structural and sequence information in ESMFold latent spaces are in fact highly compressible, and despite using

very small bottleneck dimensions, reconstruction performance can nonetheless be maintained when evaluated in sequence or structure space.

The embedding $\mathbf{x}$, defined just before the structure module, is actually a linearly projected version of the ESM2 embeddings. If we defined $\mathbf{x}$ as the ESM2 embeddings directly, we could use the decoder from ESM2's MLM training. However, since we use this modified embedding space, an approximation is necessary.

Based on reconstruction results in Lu et al. (2024), we choose $\mathbf{x}' \in \mathbb{R}^{\frac{L}{2} \times 32}$ with $L = 512$, which balances reconstruction quality at a resolution comparable to the size of latent spaces in image diffusion models (Rombach et al., 2022). Dividing the length in half allows us to better leverage the scalability and performance of Transformers, while managing their $\mathcal{O}(L)$ memory needs.

The CHEAP module involves a channel normalization step prior to the forward pass through the autoencoder. We find that the distribution of embedding values is fairly "smooth" here (Figure 10). Althoughgh the original Rombach et al. (2022) paper was trained with a KL constraint to a Gaussian distribution, we use the embedding output as is. CHEAP embeddings were also trained with a $\tanh$ layer at the output of the bottleneck; this allows us to clip our samples between $[-1, 1]$ at each diffusion iteration, as was done in original image diffusion works (Ho et al., 2020; Ho and Salimans, 2022; Dhariwal and Nichol, 2021; Saharia et al., 2022). We found in early experiments that being able to clip the output values was very helpful for improving performance. Without using CHEAP compression prior to diffusion, sample quality was poor, even for short ($L = 128$) generations, as shown in Figure 11.

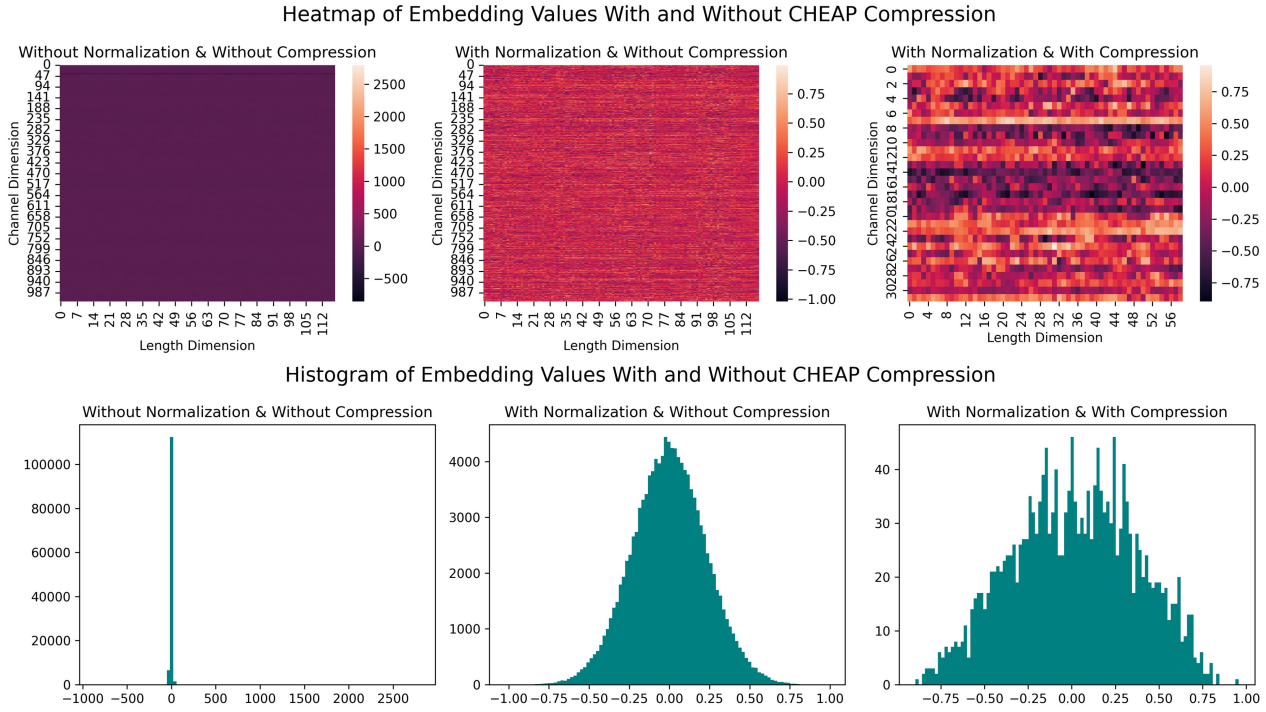


Figure 10. Visualizing the original ESMFold latent space before normalization, after per-channel normalization, and after compression. The value distribution of $p(\mathbf{x}')$ is fairly smooth and "Gaussian-like," making it amenable to diffusion.

E. Data

The PLAID paradigm can be applied to any sequence database. As of 2024, sequence-only databases range in size from UniRef90 (Suzek et al., 2015) (193 million sequences) to metagenomic datasets such as BFD (Steinegger et al., 2019) (2.5 billion sequences) and OMG (Cornman et al., 2024) (3.3 billion sequences). We use Pfam because it provides more annotations for *in silico* evaluation and because protein domains are the primary units of structure-mediated functions.

We use the September 2023 Pfam release, consisting of 57,595,205 sequences and 20,795 families. PLAID is fully

compatible with larger sequence databases such as UniRef or BFD (roughly 2 billion sequences), which would offer even better coverage. We elect to use Pfam because sequence domains have more structure and functional labels, making it easier for *in silico* evaluation of generated samples. We also hold out about 15% of the data for validation.

We examine all Pfam domains that have a Gene Ontology mapping, resulting in 2,219 GO terms compatible with our model. For domains with multiple associated GO term labels, the GO term that is least prevalent in our dataset is selected, to encourage the selected terms to be more specific.

Approximately 46.7% of the dataset ($N = 24,637,236$) is annotated with a GO term. Using the publicly available mapping as of July 1, 2024, we count all GO occurrences; for each Pfam entry with multiple GO entries, we pick the one with the fewest GO occurrences to encourage more descriptive and distinct GO labels.

The `Pfam-A.fasta` file available from the Pfam FTP server includes the UniRef code of the source organism from which the Pfam domain is derived. The UniRef code furthermore includes a 5-letter "mnemonic" to denote the organism. We examine all unique organisms in our dataset and find 3,617 organisms.

F. Sampling

Inference-time sampling hyperparameters provide the user with additional control over quality and sampling speed trade-offs. PLAID supports the DDPM sampler (Ho et al., 2020) and the DDIM sampler (Dhariwal and Nichol, 2021), as well as the improved speed samplers from DPM++ (Lu et al., 2022). We find that using the DDIM sampler with 500 timesteps using either the sigmoid or cosine schedulers works best during inference, and reasonable samples can be obtained using the DPM++2M-SDE sampler with only 20 steps. Experiments shown here use the DDIM sampler with the sigmoid noise schedule at 500 timesteps.

Note that the performance bottleneck is found mostly during the latent sampling and structure decoding (which depends on the number of recycling iterations (Jumper et al., 2021; Lin et al., 2023) used); however, these two processes can be easily decoupled and parallelized, which cannot be done in existing protein diffusion methods. Furthermore, it allows us to prefilter which latents to decode using heuristic methods, and decode only those latents to structure, which would boost performance for nearly the same computational cost. We do not empirically explore this in this paper to provide a fair comparison, and because the filtering criteria would vary greatly by downstream use.

G. Evaluation Details

For all benchmarks and models, we use default settings provided in their open-source code. For ProteinMPNN (Dauparas et al., 2022), we use the `v_48_002` model with a sampling temperature of 0.1 and generate 8 sequences per protein, from which the best performing sequence is chosen. To calculate self-consistency, we fold sequences using OmegaFold (Wu et al., 2022b) rather than ESMFold, again using default settings.

Though our models generate all-atom structure, we examine C_α RMSD rather than all-atom RMSD to avoid misattributing sequence generation underperformance to structure generation failures. Also, since there are usually differences in the sequences that are generated, different numbers of atoms make it difficult to assess all-atom RMSD.

For the hold-out natural reference dataset, we use sequences from Pfam and keep length distributions similar to that of the sampled proteins. Specifically, for each sequence bin between $\{64, 72, \ldots, 504, 512\}$, we take 64 natural sequences of that length. For the experiment in Figure 18D, we use the Sinkhorn Distance rather than the Fréchet Distance used commonly in images and video. Since not all functions have a large number of samples, we elected to use a metric that works better in low-sample settings.

Structure novelty is obtained by searching samples against PDB100 using Foldseek (van Kempen et al., 2022) `easy-search`. We examine the TM-score to the closest neighbor. For Foldseek and MMseqs experiments, all clustering experiments are performed by length. We use default settings for both tools. Though we report the average TM-Score to the top neighbor for Foldseek, we run `easy-search` in 3Di mode. For sequences, we use MMseqs2 (Steinegger and Söding, 2017) to see if sequences have a homolog in UniRef50, using default sensitivity settings. For samples with homologs, we further calculate the average sequence identity to the closest neighbor to assess novelty (Seq ID %).

G.1. Distributional Conformity to Biophysical Attributes

For Wasserstein Distance to the distribution of biophysical attributes, we examine the following:

- Molecular Weight (MW): the molecular weight calculated from residue identities specified by the sequence.
- Aromaticity: relative frequencies of phenylalanine, tryptophan, and tyrosine, from Lobry and Gautier (1994).
- Instability Index: dipeptide-based heuristic of protein half-life, from Guruprasad et al. (1990).
- Isoelectricity (pI): the pH at which a molecule has no net electrical charge.
- Hydropathy: based on the GRand AVerage of hYdropathy (GRAVY) metric, from Kyte and Doolittle (1982).
- Charge at pH $= 7$: the charge of a given protein at pH $= 7$, i.e., neutral pH.

H. Sampling Speeds

We examine the amount of time necessary for generating a simple sample. We first explore the time necessary to generate 100 sequences with $L = 600$. Multiflow and ProteinGenerator does not support batched generation in its default implementation, so in this experiment, we simply generate one sample at a time for a total of 100 samples. We report the amount of time per sample. For comparison, we also run an experiment where we only generate a single sample, such that none of the methods can make use of any improvements from batching.

Table 5. Time required to sample **proteins with 600 residues**. We assess time required both for sampling $N = 100$ samples in batches whenever possible, and when generating a single sequence. Experiments are run on Nvidia A100. Methods marked by (*) do not support batching in the default implementation.

	seconds/sample, batched		seconds/sample, unbatched	
	Sample Latent	Decode	Sample Latent	Decode
Protpardelle	11.21	-	17.16	-
Multiflow*	231.32	-	277.11	-
ProteinGenerator*	343.32	-	342.28	-
PLAID (100M)	1.64	15.12	27.63	1.07
PLAID (2B)	19.34	15.07	49.03	0.9

Table 6. Forward pass benchmark of vanilla multihead attention compared to the optimized xFormers implementation of memory-efficient attention (Rabe and Staats, 2021) and FlashAttention-2 (Dao, 2023). Though FlashAttention2 performed best in our benchmarks, a fused kernel implementation with key padding was not yet available at the time of writing. Since our data contained different lengths (as compared to most image diffusion use cases, or language use-cases that can make use of the implemented causal masking), we instead use the xFormers implementation. We expect that sampling speed results would improve once this feature is becomes available in the FlashAttention package.

Method	**Mean Time (s)**	**Mean Memory (GB)**
Standard Multihead Attention	0.0946 ± 9.23e-4	76.0 ± 0.409
xFormers Memory Efficient Attention	0.0519 ± 4.33e-05	64.0 ± 0.409
Flash Attention	0.0377 ± 1.91e-3	49.2 ± 0.783

I. Additional Results

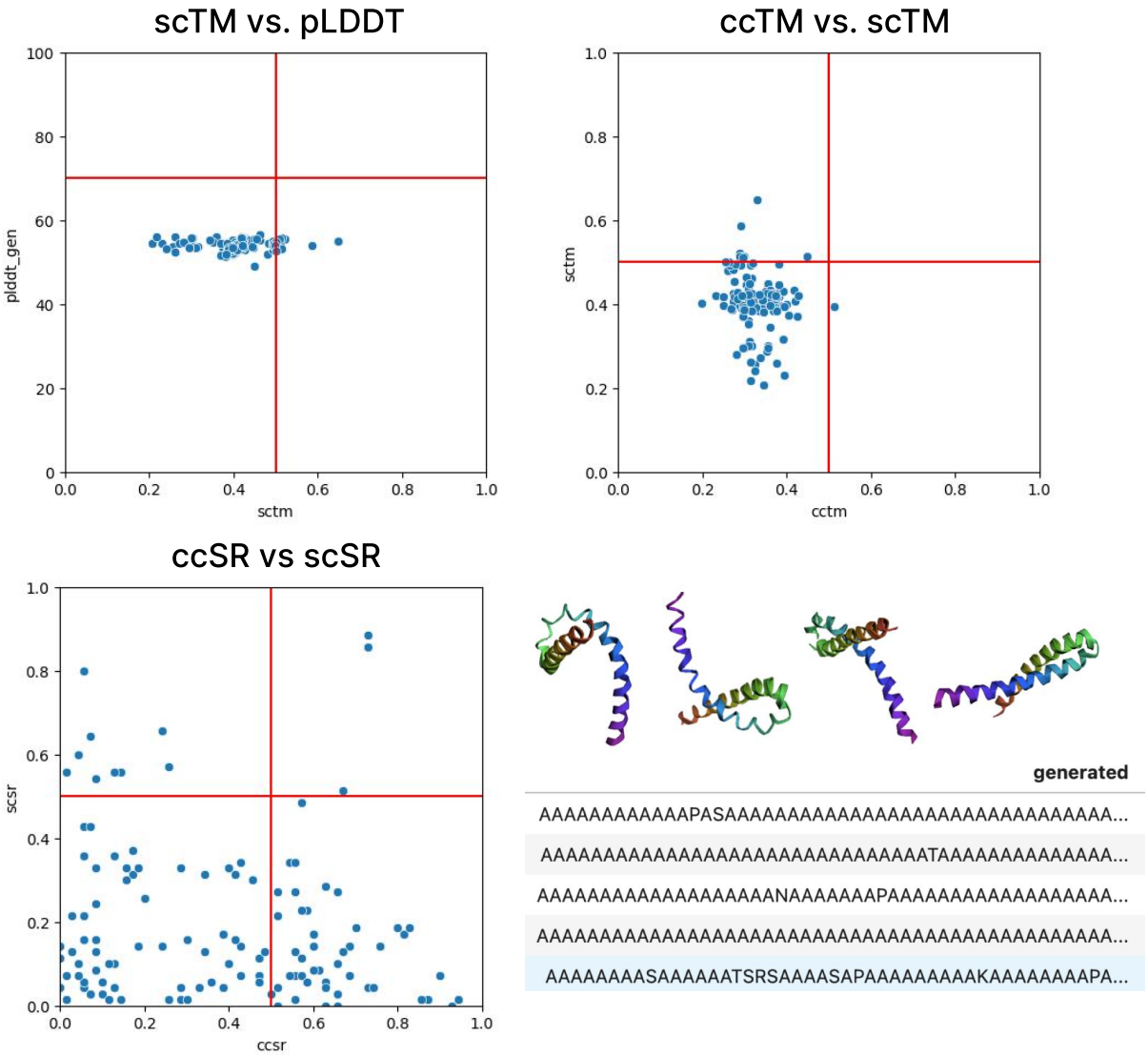


Figure 11. Results when running PLAID on the ESMFold latent space naively without CHEAP compression, for proteins of length 128. There is a tendency to generate repeated sequences, and quality is low compared to baselines.

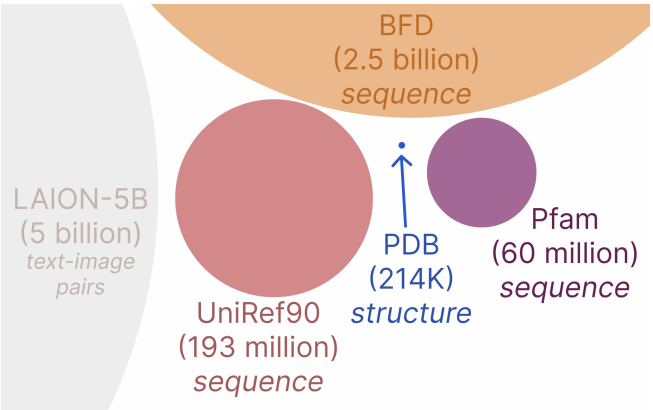
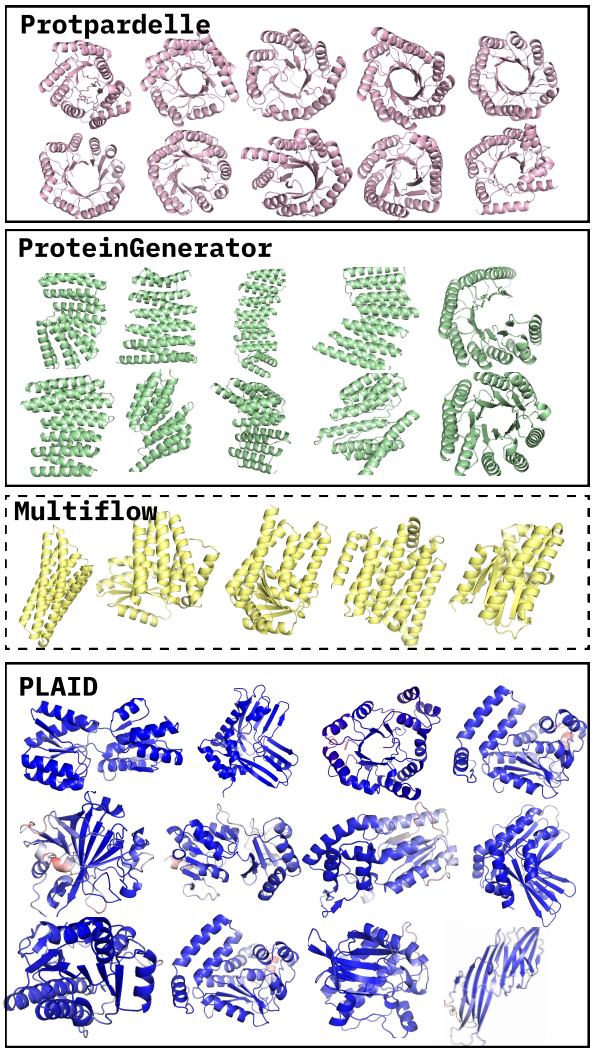


Figure 12. Size comparison of datasets drawn to scale. Sequence databases provide significantly more comprehensive coverage of the natural protein space than structural databases.



Protpardelle
>len600_samp97
AGGGGGGGGGGGGGGGGGGGGGGGGLGLGLLLPPAGL...
>len600_samp98
PPPPGGAGGGGAAAALAGGSPGGPPGGGGGGGGGGGG...
>len600_samp99
PPGPALPPSPGPGGVPPPPPLPPPPLPGGAPPAGGGLL...

ProteinGenerator
>len600_000097
GAAGLTAAAAVVGAAAAAGAAAAAALAAAAGAGAAAAA...
>len600_000098
AGAAGAAAAAAAAAAGAAAAGAGGGAGGAAAAAAAAAG...
>len600_000099
VAAAQAVQGAIAAAAALAATAALGLTAAGIAAPLLALV...

Multiflow
>len600_sample_97
LLGGLLGGLLGGAAGGAGAGAAAAGGGAVGVGVAGAVT...
>len600_sample_98
ADAATLTVGGGGTGGGGGAGGALGGAAAGGGGRVTLVV...
>len600_sample_99
AGGGAGLAGGAGGAGGAAAAAAAAAAAAAGAGGGAAAA...

PLAID
>len600_sample97
PDMGTVLGLAHSVGHLDFKTPDLSVADLETNLALLAAH...
>len600_sample98
FEMFDDKGGDLWERAASSGQLLIDVAYLANNGLRDGAT...
>len600_sample99
GNGGQARGTDDPLTHALQTLFQSAALDQSLQGDPENAV...

Figure 13. **Qualitative comparison of samples. (Left)** PLAID samples are structurally diverse, whereas baseline methods demonstrate mode collapse towards common structures such as TIM barrels and α-helix bundles at $L = 256$. **(Right)** Baseline methods generate sequences which have more repeats, especially at longer sequence lengths. Shown are sequences with $L = 600$, truncated for visualization.

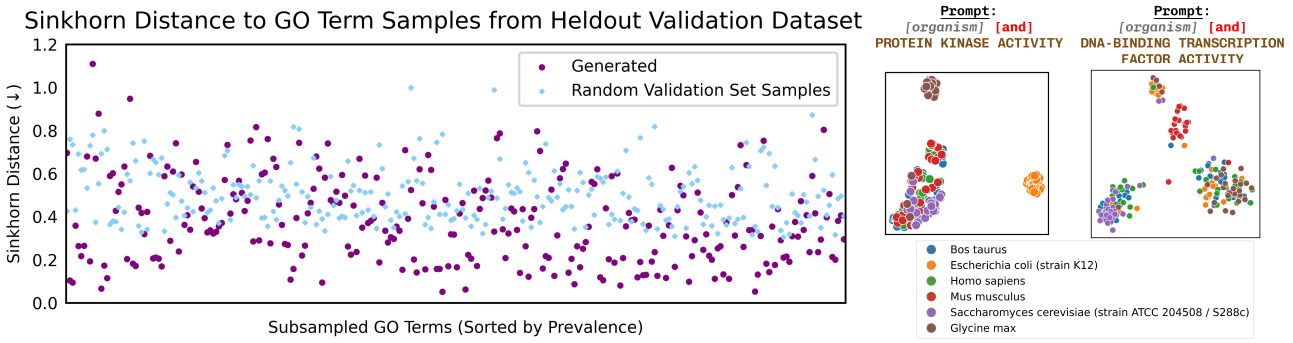


Figure 14. **(Left)** For each unique GO term in the validation set, we examine the Sinkhorn distance between generated samples and real proteins in the heldout subset in this class. For reference, we also calculate the Sinkhorn distance between random real proteins to the heldout subset. **(Right)** t-SNE reduction of generated embeddings, colored by the organism used for conditioning.

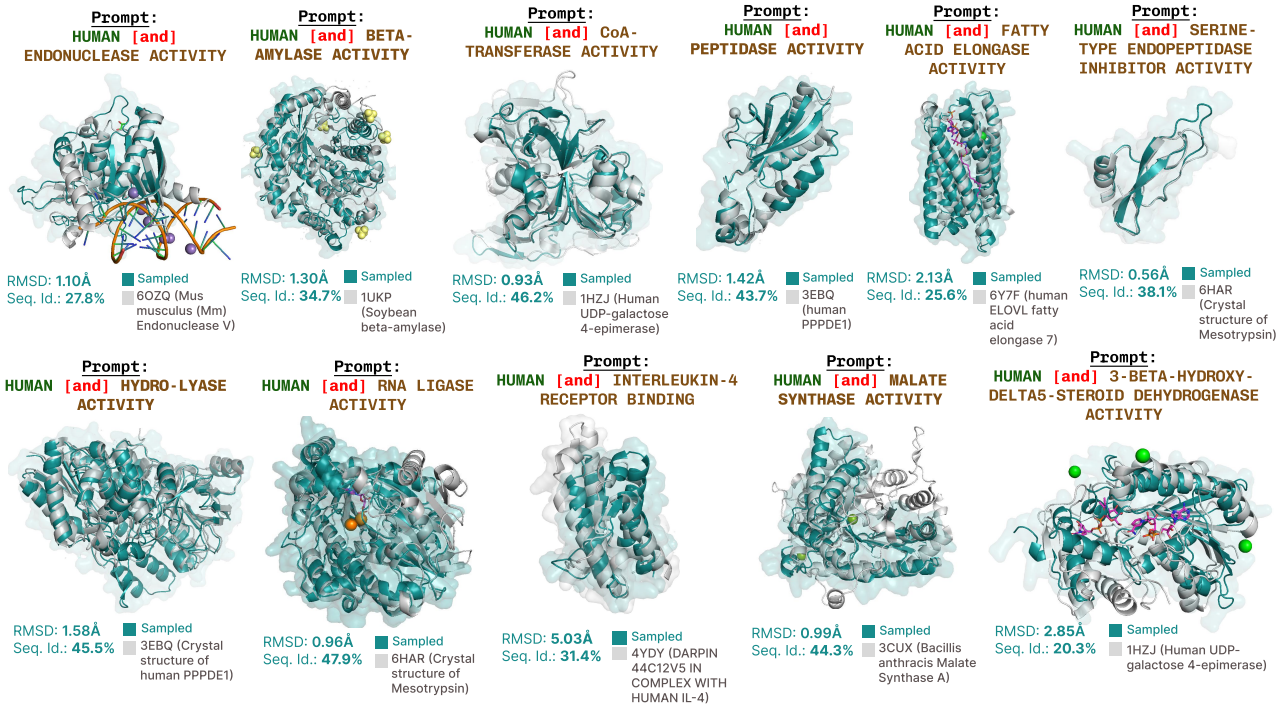


Figure 15. Additional examples of function-conditioned generations.

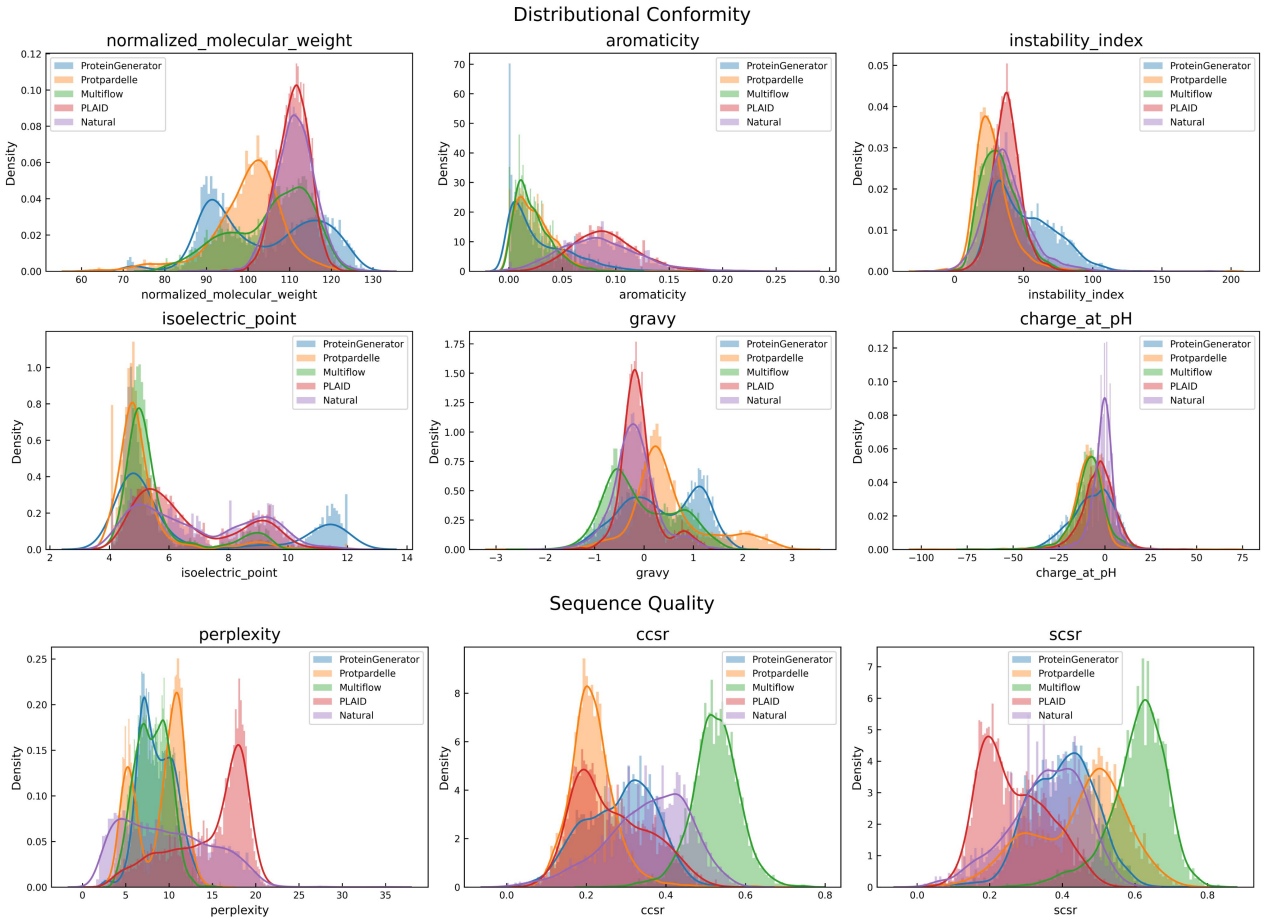


Figure 16. Examining histogram of metrics for nuanced comparison of how generated samples compare to that of natural proteins.

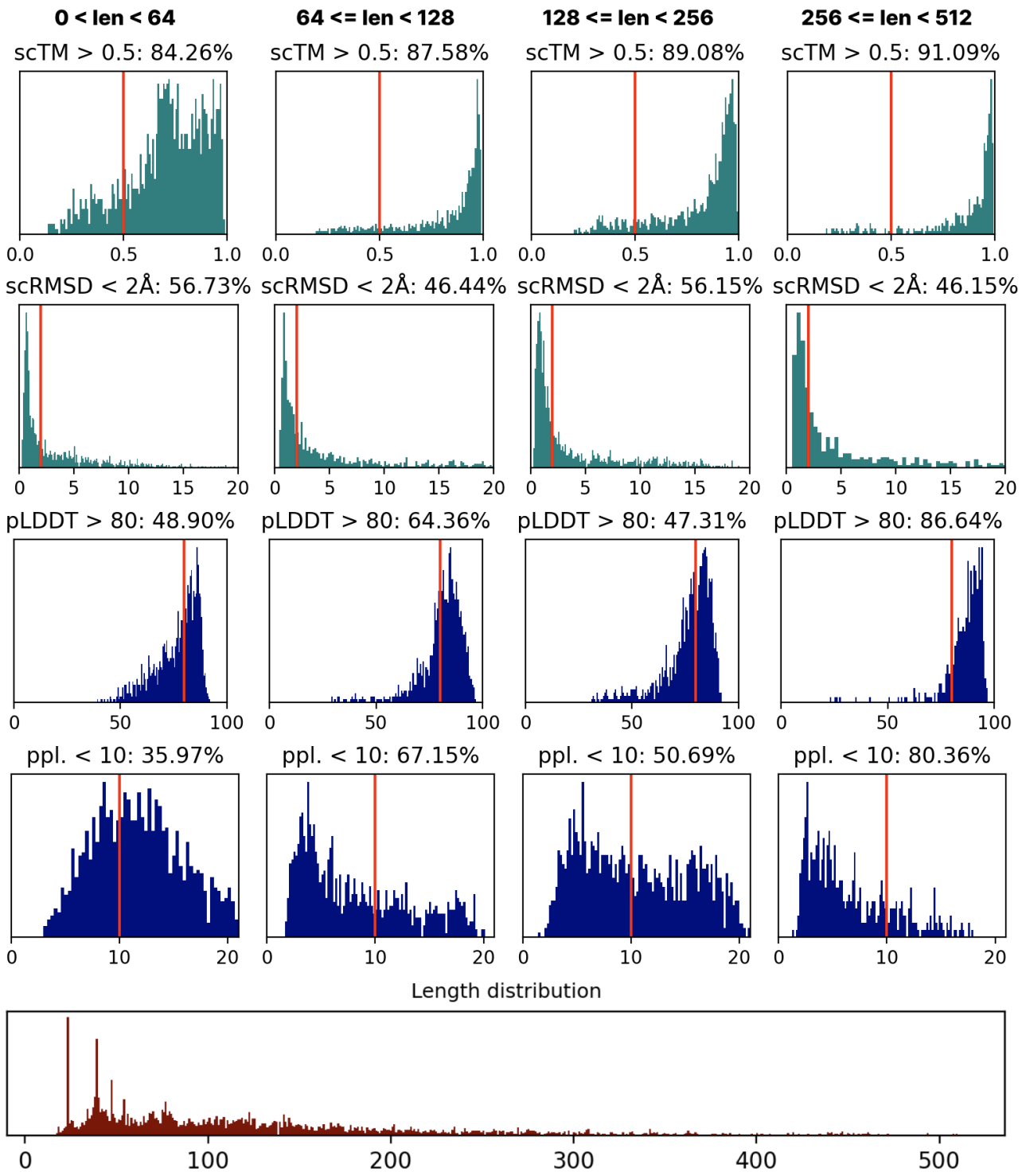


Figure 17. To examine the degree to which co-generation methods are overfitting to structure-based metrics, we examine properties on natural proteins.

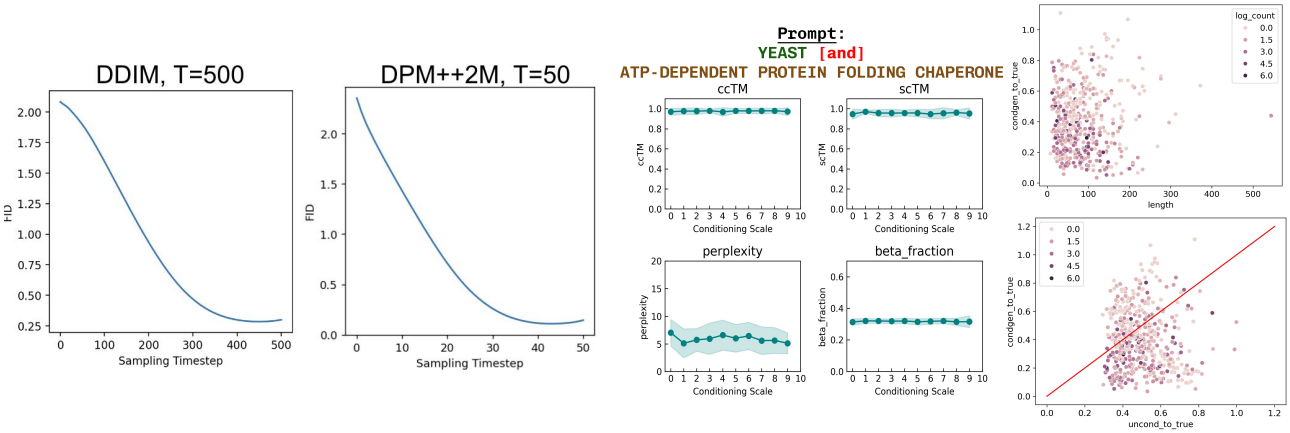


Figure 18. **(Left)** Frechet Distance between sampled protein and reference set of real protein, across sampling (reverse diffusion) timesteps, for the DDIM (Dhariwal and Nichol, 2021) sampler and the DPM++2M (Lu et al., 2022) sampler. For both, sample quality decreases steadily over time before plateauing. DPM++2M can achieve low FID results with only 10% of the original number of steps, but final results are still slightly worse. **(Center)** Examining the effect of conditioning scale on the output quality. **(Right)** Analyzing factors which may be contributing to a greater δ difference between the Sinkhorn distance of samples-to-real-functional-proteins vs random-proteins-to-real-functional-proteins.

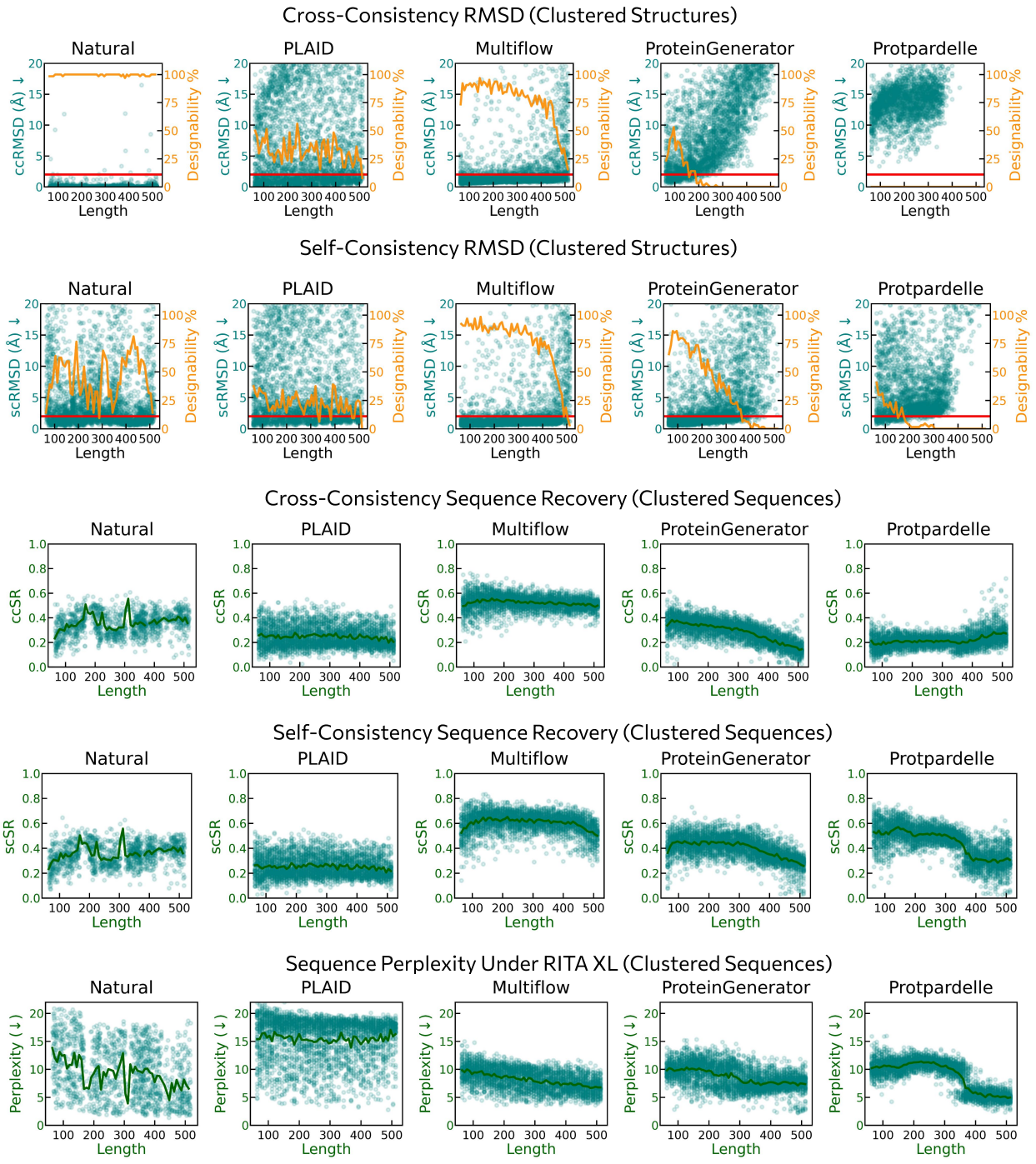


Figure 19. More comparison results between PLAID and baselines. Note that we omit Multiflow in most comparisons to avoid confusion, since unlike the other models, Multiflow only generates sequence identity, and not the sidechain positions.