Jovian: Bidirectional Autoregressive Protein Structure Generation

Rohit Dilip
Computing + Mathematical Sciences
California Institute of Technology
rdilip@caltech.edu

Ayush Varshney*
Computing + Mathematical Sciences
California Institute of Technology
varshney@caltech.edu

Evan Zhang*
Computing + Mathematical Sciences
California Institute of Technology
evanzhang@caltech.edu

David Van Valen
Biology + Biological Engineering
California Institute of Technology
Howard Hughes Medical Institute
vanvalen@caltech.edu

Abstract

We introduce a bidirectional autoregressive model for protein backbone generation. Unlike previous models that require fixed sizes to be imposed at inference-time, our model Jovian learns both protein sequence length and motif positioning by conditioning explicitly on prediction direction. We then propose a bidirectional sampling algorithm that enables simple, zero-shot motif scaffolding without any additional motif-specific training. In unconditional design, our model is 10-40$\times$ faster than similar models, maintains greater stability on long sequences, and matches or exceeds the performance of diffusion-based methods on standard benchmarks. In conditional design, our model performs zero-shot conditional generation while maintaining its speed advantages. Our findings demonstrate the viability and strengths of autoregressive models in protein structure design and open new modeling avenues for generative biology.

1 Introduction

Designing novel proteins – the fundamental players in nearly all biological processes – with specific functions is a grand challenge of generative biology with applications across therapeutics, biosensors, and more [1, 2]. However, protein design often requires manual spatial imputation from expert scientists, ranging from where to place functional sites to determining protein size *a priori* based on heuristic intuitions. The substantial progress in deep learning-based structural biology has not resolved these issues; diffusion-based methods rely on fixed templates that specify exactly where to place functional sites and the precise length of the protein before generating structures/sequences.

Autoregressive transformers, which naturally provide variable-length generation, have driven progress in computer vision and natural language. Unfortunately, adapting autoregressive transformers to practical 3D protein structure generation is very challenging, in large part because proteins have very long-range spatial dependencies and can quickly become irreconcilable with a desired functional site during autoregressive inference.

Contributions: In this work, we demonstrate how autoregressive transformers can be extended to eliminate motif placement and protein length determination from conditional protein design. Our primary contribution is introducing direction-conditioned autoregressive models, which enable

*Equal contribution, order determined alphabetically

Preprint. Under review.

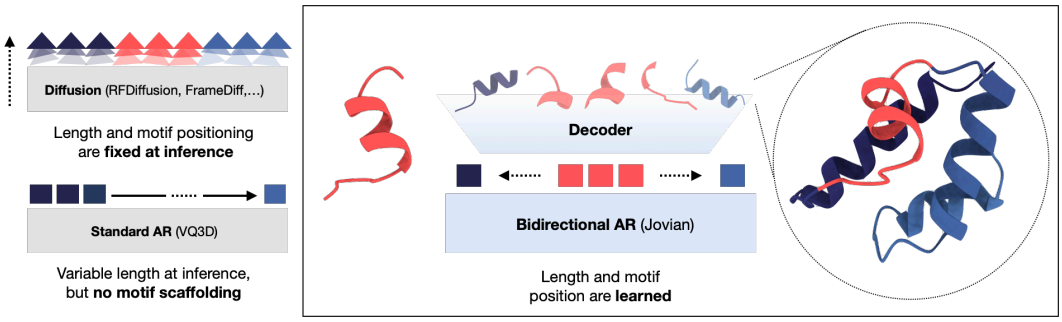

Figure 1: Diffusion models fix the position of the motif while denoising directly in the $SE(3)^L$ manifold (represented by triangles). Standard autoregression generates unconditionally but cannot perform motif scaffolding. Jovian generates autoregressively from the motif. **Takeaway:** We perform motif-scaffolding without ad-hoc length and motif position imputation.

variable-length unconditional/conditional design and eliminate *ad hoc* steps in motif scaffolding. Our formulation also eliminates a training stage of motif scaffolding where randomly sampled pseudo-motifs are used to train models; rather, we perform *zero-shot* conditional protein design. Our final model, Jovian, generates protein structures 10 to 40$\times$ faster than diffusion-based methods while demonstrating greater stability on long protein sequences. Our findings demonstrate that autoregressive models are a promising approach for structural generation and unlock new opportunities for generative modeling of biological structures.

2 Background and Related Work

2.1 Protein Structure Generation

A protein is defined by a linear sequence of L amino acids, each with a characteristic backbone specified by the atomic coordinates $\{(\boldsymbol{x}_N^{(i)}, \boldsymbol{x}_{C\alpha}^{(i)}, \boldsymbol{x}_C^{(i)})\}$. Each coordinate $\boldsymbol{x}^{(i)} \in \mathbb{R}^3$ references the i-th amino acid. Each residue type $m_i \in \{0, \ldots, 19\}$ corresponds to one of the twenty standard amino acids, giving rise to a sequence $\{m_i\}_{i=1}^L$ and a corresponding 3D structure. Following the success of AlphaFold2 in protein structure prediction [3], it has become common to parameterize protein backbones by a sequence of rigid frames, where each residue is described by a rotation-translation pair in $SO(3) \otimes \mathbb{R}^3$. This representation is particularly amenable to geometric learning; models are often constructed to be invariant or equivariant to the group $SE(3)$ to respect the physical symmetries of 3D space.

In this work, we focus on the problem of *protein structure design*: generating a set of 3D coordinates $\{(\boldsymbol{x}_N^{(i)}, \boldsymbol{x}_{C\alpha}^{(i)}, \boldsymbol{x}_C^{(i)})\}^L$ that describes a *designable* protein, i.e., there exists at least one sequence $\{m_i\}^L$ that reliably folds into the generated structure. This task is at the core of structure-based protein engineering and is a prerequisite for downstream tasks such as function design, binding, and stability optimization [4, 5, 6]. Early works focused primarily on unconditional design, where a model generates an arbitrary designable protein [7, 8]. More recent works have focused on conditional design where generated proteins must have specific properties. These properties may be related to size, thermal stability, solubility, etc [9, 10, 11, 12]. A critically important conditional design objective in drug discovery and biopharma is *motif scaffolding*, where a small protein segment (a motif) is provided as input and the model must generate a scaffold to secure and hold the motif in place [13, 14, 15].

Multiple works have advanced various approaches to structure generation. RFDiffusion [16] introduced a diffusion-trained backbone model by fine-tuning RosettaFold and experimentally validated their generated structures. FrameDiff and FrameFlow propose a $SE(3)$-invariant diffusion and flow models respectively [7, 8]. Genie 2 [17] introduced diffusion over a cloud of inter-residue reference frames to ensure SE(3) invariance with respect to Cα coordinates. ESM3 [18] uses MaskGIT [19] style sampling to iteratively sample sequence and structure from distinct token spaces. DPLM-2[20] similarly uses discrete diffusion with structure tokens to perform structure generation.

Almost all current structure generation models are diffusion or flow-based where the model learns to denoise data corrupted towards a prior that can be easily sampled (e.g., a Gaussian or a sequence of <MASK> tokens) [21, 22]. For proteins, this means the data and noise live on a manifold $SE(3)^L$, where L is defined and *fixed* at inference time. Motif scaffolding compounds this problem, because in addition to choosing the length of the scaffolding structure, one must impute the position of the motif along the sequence (see Figure 2c). These constraints are not learned and there are a combinatorially large number of possibilities; in practice, these models sample a large number of motif positions and lengths and filter by designability [23, 16]. Existing benchmarks accommodate these limitations by providing information on motif position and size, but the ad-hoc nature of this process poorly represents more general open-world protein design tasks where this information is not available [13]. As a concrete example, [24] uses surface fingerprints to design motifs that bind to particular sites on pharmaceutically important proteins, so they only have access to the designed motifs and have limited priors over motif positioning and scaffold length. Motifs often will have high free energy and will not realistically fold to a particular structure except in the presence of a stable scaffold, which increases the appeal of an end-to-end model that *learns* these contraints [25, 26, 27]. To our knowledge, the only work that tackles motif scaffolding without knowing positioning a priori is [23], which uses a diffusion model over anchor positions to place the motif. This work, however, still requires a fixed protein length at inference time.

Method	**SE(3)**	**Var. length**	**Motif**	**Learned pos.**
FrameDiff/Flow	✓	✗	✓	✗
Genie/Genie 2	✓	✗	✓	✗
Chroma	✓	✗	✓	✗
RFDiffusion	✓	✗	✓	✗
VQ3D	✗	✓	✗	✗
Ours	✓	✓	✓	✓

Table 1: Comparison of protein structure generators, referencing SE(3) invariance, variable length generation, motif scaffolding, and learned motif positions respectively. **Takeaway:** Prior autoregressive work is incompatible with motif scaffolding, while prior diffusion models cannot generate variable lengths, nor learn the motif position.

2.2 Neural discrete learning

A common approach to train autoregressive models on continuous data is vector quantization (VQ) [28]. In vector quantization, continuous data is first tokenized using a discrete codebook, then a generative model is trained using a standard language modeling objective. An encoder maps input data to a sequence of latent vectors. Each vector is replaced by its nearest-neighbor in a trainable codebook. These vectors are decoded; the encoder, codebook, and decoder are optimized via a reconstruction loss betwen the input and the reconstructed data. A generative model trained on tokenized sequences can be decoded back to the underlying data modality.

Several recent works have used VQ methods to create generative models over protein structure. DPLM-2 trains a discrete diffusion model using lookup-free quantization (a VQ variant that avoids some of the difficulties of standard vector quantization) [20]. ESM3 trains a generative model with a MaskGIT-style objective; this is very similar to the discrete diffusion approach without explicitly conditioning on time [18]. Most relevant to our work is [29], which trains a structural tokenizer using a ProteinMPNN-style encoder [30], an AlphaFold2 structure module decoder [3], and lookup-free quantization [31], which they then apply to generate protein structures using a GPT-2 style decoder. However, their method is fundamentally incompatible with conditional protein design for reasons we describe in Section 3.1.

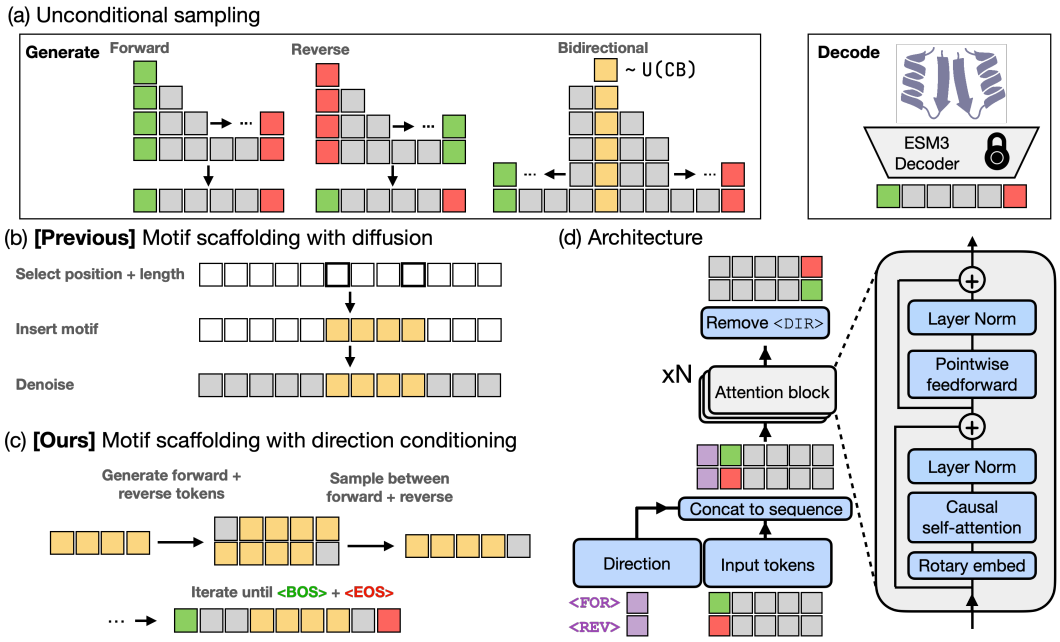

Figure 2: **(a) Unconditional sampling.** We can sample in forward or reverse mode, starting from a <BOS> or <EOS> token respectively (in reverse mode, we flip the outputted sequence). We can also sample bidirectionally by randomly sampling a token from the codebook and treating it as a motif. We decode the generated sequences using the frozen ESM3 decoder. **(b) Diffusion approaches to conditional design.** In current SOTA diffusion/flow matching methods, the motif position and the sequence length are preselected before the remaining inputs are denoised. **(c) Direction-conditioned motif scaffolding.** To perform motif scaffolding, we begin with a tokenized motif and sample in both the forward and reverse directions. We dynamically predict which token to select (left versus right) and iteratively sample in this fashion until we reach either a <BOS> or <EOS> token, at which point the model proceeds autoregressively along the remaining direction until the sequence is fully generated. **(d) Architecture:** Our main deviation from a standard GPT architecture is the addition of a direction token (either <FOR> or <REV>) which is provided to the model as in-context conditioning. The conditioning vector is removed before predicting the next token. The use of relative positional embeddings (e.g., RoPE [32]) is important, as the absolute position is not known during inference. **Takeaway:** Explicitly conditioning on direction eliminates motif positioning and length selection heuristics.

3 Method

3.1 Tokenizer design for autoregressive modeling

Why have autoregressive models struggled in structure generation? Standard left-to-right next-token prediction used in natural language processing has several problems when applied to protein design. First, because interactions between protein residues are mediated by their relative positioning in 3D space, tokenized sequences can have very long bidirectional correlations. This exacerbates error accumulation, a known problem with autoregressive models [33]. This is particularly problematic for conditional protein design, where the functional segment of the protein (the motif) may be relatively far from the beginning of the sequence; compounded mistakes may result in undesignable structures by the time the generation process reaches the motif. Diffusion models avoid this problem by providing full context throughout the generation process at the cost of variable length generation. Second, naive autoregressive design requires that models reason about tokens far in the future. Even if a model had near-zero per-token error rates, it would still need to plan ahead over a very long context and adapt as newly sampled tokens further constrain the yet-to-be defined motif.

These observations place important constraints on tokenizer design. First, because absolute sequence position is unknown at inference-time in variable-length motif scaffolding, a tokenizer cannot use the

absolute sequence position as input and still perform conditional generation. While absolute positional information empirically improves tokenizer reconstruction quality, these tokenizers are incompatible with motif scaffolding and can only be applied to unidirectional, unconditional generation. Second, most tokenizers are trained by mixing information from neighboring amino acids before quantization and reconstruction (e.g., using a graph neural network or attention mechanism). While increasing the number of neighbors improves reconstruction accuracy in VQ-VAE training, if the number of neighbors is significantly larger than the motif, then autoregressive models can rapidly grow unstable because sampled tokens have a very large receptive field. Moreover, a very large number of neighbors will limit the length of reconstructions (see VQ3D in Figure 3d). As our focus is on developing an autoregressive paradigm compatible with generative tasks, we use an off-the-shelf tokenizer that satisfies these properties (the structure encoder/decoder developed in [18]) and defer additional studies on tokenizer design to future work.

3.2 Direction conditioning

To enable conditional protein design, we propose *direction-conditioning*. We train GPT-style transformers explicitly conditioned on the prediction direction, i.e., predicting the next or the previous token along the sequence. At train time, we randomly sample an initial token along the sequence of structure tokens (reversed with probability 0.5), condition on the appropriate direction token (either `<FOR>` or `<REV>`), and train using a standard next-token prediction objective. This relatively simple addition enables both unconditional design and autoregressive motif scaffolding. By conditioning on direction, we can begin with a known motif and iteratively generate left and right tokens. This approach (depicted in Figure 2c) has several key advantages. First, we can perform motif scaffolding without specifying either the motif position or the total length of the protein. Second, motif-scaffolding is completely zero-shot; no additional training is necessary to unlock motif scaffolding. This is in contrast with prior approaches like [17], which assembled datasets of synthetic motifs by randomly sampling subsequences of protein structures. We frame our approach against prior work in Table 1.

In forward mode, our model subsumes the generative capabilities in [29]. We can generate protein structures unconditionally in forward-mode by predicting autoregressively from `<BOS>` or in reverse-mode by predicting autoregressively from `<EOS>` and reversing the sequence after generation (these can actually encode different structural distributions, we discuss this in Section 4.2.1). We can also sample in bidirectional mode by randomly selecting a single token from the vocabulary $\mathcal{V}$, then iteratively sampling left and right tokens (i.e., treating the sampled token as a "motif" with length one). During inference, we condition on direction by concatenating the appropriate direction token to the start of the sequence; this token is removed before prediction.

Our model architecture and training schema are shown in Figure 2. To build Jovian and retain the favorable scaling properties of transformers, we use well-established architectural components – rotary embeddings, layer norm, and GELU activation functions [32, 34, 35]. Our final model has 24 layers with a hidden dimension of 1280, MLP dilation factor 2, and 16 heads. A salient architectural detail is the use of relative positional embeddings (in our case, RoPE [32]). Because the absolute sequence length is not known at inference time, rotary embeddings inject positional information without requiring absolute positional knowledge. Additional model classes that we explored but demonstrated limited success are in Appendix C. For training, we compiled a dataset composed of the Open Protein Set [36] and proteins in the Swissprot Alphafold database [37] with pLDDT > 80. We experimented with both MMseqs and Foldseek to downsample our dataset [38, 39]. The details of our training setup and impact of various architectural ablations are discussed in Appendix B.

3.3 Sampling

In contrast to [29], we find that sampling methods that filter low-probability tokens are critical for getting designable samples (presumably due to the absence of absolute positional information increasing uncertainty). We conduct our experiments with min-p and nucleus sampling [40, 41] unless stated otherwise. We adopt the following sampling algorithm to generate bidirectional samples. Assume our model is given by f_θ. Our input is a sequence $v_{1:n}$, with $v_i \in \mathcal{V}$, the structure vocabulary. For unconditional generation, $v_{1:1}$ is a length-one sequence randomly sampled from $\mathcal{V}$. For conditional generation, $v_{1:n}$ is a motif.

1. **Model pass.** Compute logits in both directions: $h_{\text{fwd}} = f_\theta(v_{1:n} \mid \text{forward}), \quad h_{\text{rev}} = f_\theta(v_{n:1} \mid \text{reverse})$.
2. **Compute probabilities.** $p_{\text{fwd}} = \text{softmax}(h_{\text{fwd}}), \quad p_{\text{rev}} = \text{softmax}(h_{\text{rev}})$.
3. **Sample candidate tokens.** Sample $v_{\text{fwd}}, v_{\text{rev}} \in \mathcal{V}$ from p_{fwd} and p_{rev}, respectively.
4. **Sample generation direction.** Let $p_1 = p_{\text{fwd}}[v_{\text{fwd}}]$, $p_2 = p_{\text{rev}}[v_{\text{rev}}]$. Select v_{fwd} with probability $\frac{p_1}{p_1+p_2}$, and v_{rev} with probability $\frac{p_2}{p_1+p_2}$.
5. **Update sequence.** Append v_{fwd} to the right or prepend v_{rev} to the left of $v_{1:n}$, depending on the chosen token.

We can iteratively generate tokens to the left and right using this scheme. We implement min-p or nucleus sampling by directly adjusting the logits before taking a softmax.

4 Results

4.1 Protein generation metrics

We follow standard practices and report designability, diversity, and novelty. Designability is calculated by inverse folding a structure [30, 42], refolding it using ESMFold [43], and reporting the fraction of structures with self-consistent RMSD (scRMSD) < 2Å. This reflects the observation that such structures correspond to a high likelihood of being synthesizable [16]. We used ESM-IF to perform the inverse folding. We report designablility as (# of designable structures) / (# of generated structures). In line with current best practices, we report best-of-eight where we generate eight sequences with temperature 0.1 [8, 44]. *Diversity* quantifies the variability in the *designable* backbones produced. While prior works [8] have used MaxCluster, this introduces problems when comparing proteins with different lengths. Instead, we report the average pairwise TM-score of designable structures [45]. Finally, *novelty* uses Foldseek [39] to search for the closest match in a reference database by TM-score. We report the average TM-score of the closest match to a reference database (following [29], the CATH dataset). We discuss the considerations at play for different metrics and provide the full details of our calculations in Appendix D.

4.2 Jovian generates designable proteins

We sample in forward, reverse, and bidirectional mode across a range of temperatures and sampling methods. Table 2 shows unconditional generation metrics. Designability, diversity, and novelty often trade off against each other (e.g., one can produce designable samples by lowering the temperature at the cost of diversity and novelty). We train two models, Jovian-MM and Jovian-FS, where the training set is clustered using MMseqs and Foldseek respectively (Appendix B discusses this choice). We report several representative examples here; a more thorough exploration of sampling results is presented in the appendix.

Jovian is much more stable at long sequence lengths than any other method and achieves similar performance in short sequence design to other unconditional generative models. This is shown in both Table 2 and Figure 3b and c; we avoid secondary structure mode collapse and generally have better scRMSDs at long sequence lengths than other methods (see Appendix E for extended benchmarks). Jovian achieves excellent diversities, though we do not claim state-of-the-art performance, as variable sequence lengths can confound these computations (see Appendix D for a longer discussion). Prior work observes that autoregressive models struggles with novelty [29]; we resolve this by changing the training set from the PDB + Swissprot (which are structurally very similar) to the PDB + AFDB and training on an equivalent number tokens (see Ablations - AFDB in Table 2 and Appendix D). Finally, despite being much larger, Jovian is significantly faster than every other model; see Figure 3d for direct comparisons.

There are a large number of backbone (and more recently, all-atom) generative models available, for example, [47, 7, 8, 17, 16]. Due to computational constraints, we do not systematically compare against all of these. In practice, attributing performance gains purely to algorithmic advances is difficult when multiple methods differ in the amount of training data and compute they use (e.g., RFDiffusion benefits from a significant amount of sequence pre-training) and when the given metrics explicitly trade against each other. Instead, the goal of our analyses was to show how our

Method	Designability ↑	Des-long ↑	Diversity ↓	Novelty ↓
FrameDiff	0.21 ± 0.02	0.09 ± 0.02	0.463 ± 0.002	0.73 ± 0.01
RFDiffusion	0.33 ± 0.03	0.52 ± 0.04	0.515 ± 0.001	0.52 ± 0.01
Chroma	0.08 ± 0.02	0.06 ± 0.02	0.526 ± 0.003	0.64 ± 0.01
Genie2*	0.70 ± 0.03	0.51 ± 0.04	0.512 ± 0.001	0.51 ± 0.01
Jovian (Foldseek)				
For, T=1.0	0.47 ± 0.04	0.61 ± 0.07	0.420 ± 0.002	0.72 ± 0.03
Rev, T=1.0	0.42 ± 0.04	0.42 ± 0.10	0.482 ± 0.003	0.79 ± 0.03
BiD, T=1.0	0.42 ± 0.04	0.34 ± 0.06	0.406 ± 0.003	0.72 ± 0.03
Jovian (MMseqs2)				
BiD, T=1.0	0.31 ± 0.05	0.26 ± 0.07	0.442 ± 0.005	0.72 ± 0.04
For, T=1.0	0.49 ± 0.05	0.50 ± 0.08	0.406 ± 0.003	0.74 ± 0.03
Rev, T=1.0	0.41 ± 0.05	0.50 ± 0.10	0.441 ± 0.005	0.76 ± 0.04
For, T=0.5	0.34 ± 0.05	0.28 ± 0.07	0.435 ± 0.004	0.76 ± 0.03
Ablations				
Mismatch	0.06 ± 0.03	–	0.47 ± 0.01	0.62 ± 0.06
AFDB	0.21 ± 0.03	–	0.514 ± 0.004	0.54 ± 0.02
Mismatch (LP)	0.11 ± 0.02	–	0.494 ± 0.008	0.70 ± 0.02

Table 2: **Generation metrics across different sampling methods and generation directions**. We report designability (using ESM-IF, best of 8), designability at sequence lengths > 256 (best of 8), diversity, and novelty, with our base bidirectional model (used for motif scaffolding) highlighted in blue. Jovian has the very robust performance at longer sequence lengths and is generally comparable or outperforms other models on designability and diversity (the exception is RFDiffusion, which as others have noted benefits from substantial pretraining on sequence datasets [7]). We include Genie2 (starred) for completeness, but note that it only generates Cα coordinates, which is a much easier task than generating full backbone coordinates. We could not reproduce the results for VQ3D [46], so we do not include it (see Appendix G.3.1). We discuss ablations in the main text. **Takeaway:** Jovian is comparable to SOTA models across unconditional design benchmarks.

direction-conditioned paradigm can lead to robust conditional design, variable length generation, and unconditional generation competitive with the state-of-the-art diffusion models.

4.2.1 Mismatched distributions in bidirectional sampling

When sampling bidirectionally, some configurations correspond to different forward and reverse distributions. Because each subsequent sampling pass conditions on everything generated so far (i.e., a forward pass will condition on tokens generated by the reverse model and vice versa), the resulting structure will often be undesignable. Table 2 shows one such instance under "Mismatch." Min-p sampling generally prevents this issue; more sophisticated sampling algorithms (e.g., shallow versions of beam search) can plausibly also help avoid this issue in particularly unstable regimes. In Figure 4d, we show a KDE-fit of the log-likelihoods of the model in forward and reverse mode for samples generated *in forward mode*. When these distributions are matched (e.g., top of Figure 4d), we get healthy bidirectional generations, but as the distributions shift away from each other, we get reduced designability. We explore several train-time methods to resolve this issue, such as delaying direction conditioning to the final layer (Mismatch (LP) in Table 2), but these were generally unsuccessful; see Appendix C.

4.3 Jovian is a zero-shot conditional generator

In contrast to prior works, we perform *zero-shot* motif scaffolding by prefilling the model with the desired motif and bidirectionally sampling. We evaluate motif scaffolding on a benchmark composed of single-domain motifs from the ESM3 test set and the recently released MotifBench [18, 13]. More details are presented in Appendix B.1.1. As MotifBench and ESM3 are both fairly recent publications, there is some data leakage between the test sets and the models we benchmark against (RFDiffusion, FrameFlow-amortization, Chroma). Jovian has the following advantages over existing methods:

No motif-placement: We eliminate the ad-hoc length selection and motif placement steps.

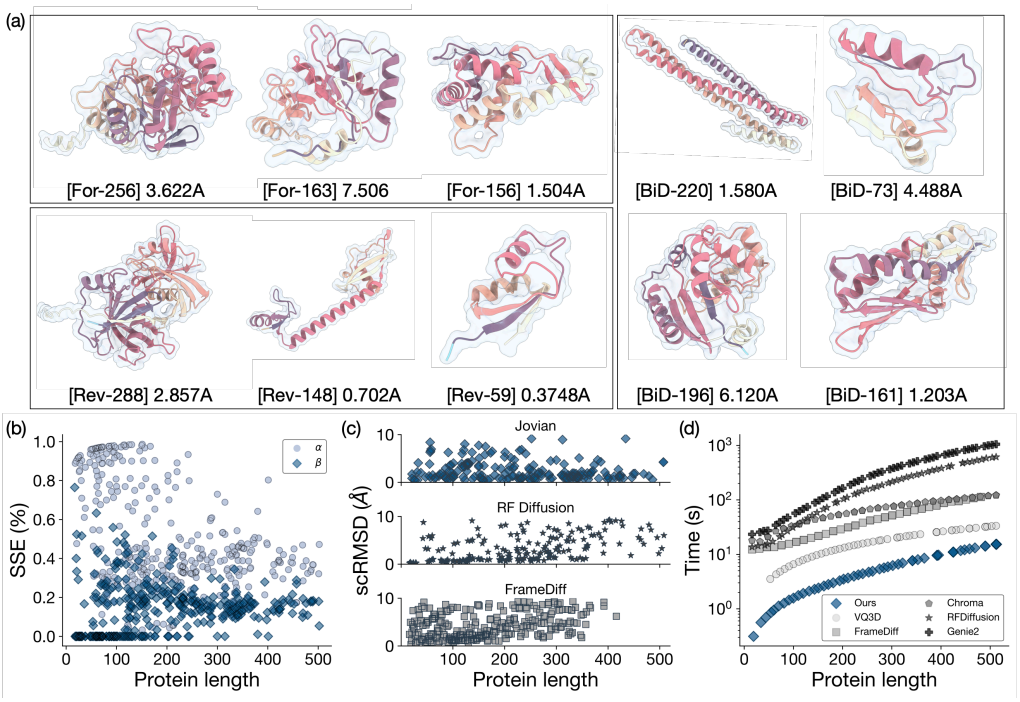

Figure 3: **(a) Jovian generates accurate protein structures during unconditional inference.** We filter samples with scRMSD ≤ 10.0Å and display representative unconditional samples. We label each sample with `[sampling_method-length] scRMSD`. **(b) Secondary structure elements.** Jovian generates diverse secondary structure without mode collapsing to alpha helices (see Appendix E for inter-model comparisons). All sample points are designable structures. **(c) scRMSD at large sequence lengths**. Jovian generates robust structures at long lengths compared to RFDiffusion and FrameDiff, which both struggle with sequence lengths > 350. **(d) Wall-clock generation speed.** We use a single GTX1080-Ti to plot generation speed in log scale as a function of sequence length. Jovian is several orders of magnitude faster than comparable diffusion models, particularly at long sequence lengths. Jovian is also faster than other structural autoregressive models like [29]. The details of our benchmarking for each method are in Appendix G. **Takeaway:** Jovian efficiently generates designable proteins with diverse secondary structure.

Zero-shot performance: In contrast with the majority of motif scaffolding models, Jovian scaffolds these motifs with no motif-specific fine tuning (see Appendix E.2 for details).

Generation speed: Jovian is approximately 4× faster than FrameFlow, 20× faster than Chroma, and 40× faster than RFDiffusion. On a fixed inference budget (e.g., 0.25 V100-hours), Jovian produces significantly more designable samples than alternative methods (see Figure 4b, bottom).

We show qualitative and quantitative results in Figure 4b. Jovian can consistently generate designable scaffolds across a range of motifs. We successfully scaffold **29/32** motifs, compared to Chroma (**5/32**, not shown in Figure 4), RFDiffusion **(32/32)**, and FrameFlow **(30/32)**. For two of the three motifs that Jovian-FS is unable to scaffold in 100 design cycles (`7wrk_A`, `7ad5_A`, and `2xga_A`), Jovian successfully generates designs by slightly lowering the temperature (from 1.0 to 0.5). This reflects the tradeoff between sampling designability and diversity. Despite the lack of motif-specific fine tuning, Jovian successfully scaffolds 6/7 orphan scaffolds, which (as described in [13]) have little sequence or structure homology to any known proteins. Jovian also successfully scaffolds `1lcc_A`, which both RFDiffusion and FrameFlow struggle to complete.

We observed several notable failure modes, which we display qualitatively on the right-hand side of Figure 4a. First, because tokenization pools information from nearby residues, when the total length of the protein is shorter than the receptive field of the tokenizer, the tokenized motif can capture information about the length of the protein. In these cases, Jovian outputs very short

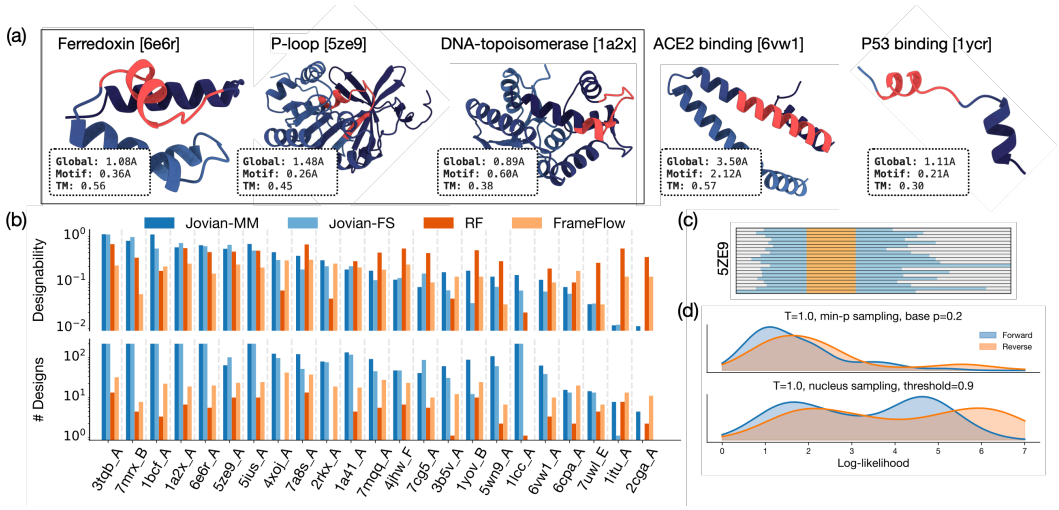

Figure 4: **(a) Motif scaffolding results:** We show scaffolding results taken from the ESM3 [18] test set (downsampled for visual clarity, see Appendix E for full set). We report the global RMSD consistency, the RMSD with respect to the original motif, and the TM-score with respect to the original chain (lower is better for all three metrics). Failure modes are shown outside the box. **(b) Jovian performance on motif benchmarks.** We show designability (top) and number of computed designs for fixed inference budget (0.25 V100-hours, bottom) for each motif in the combined ESM3 and MotifBench motif benchmarks. For visual clarity, we cap our generations at 10^2 samples. Results were generated using T=1.0 bidirectional sampling; by using T=0.5 sampling, we can generate successful designs for 2/3 motifs that Jovian-FS fails. **(c) Diversity in motif position:** Jovian can generate diverse sequence lengths; we show a representative sample of the motif position (orange) and the generated sequences (blue). **(d) Bidirectional distribution mismatches.** As the forward/reverse distributions drift apart, sample quality worsens. **Takeaway:** Jovian can perform motif scaffolding without manual motif placement.

proteins. Although these technically show a low RMSD, we believe it is unlikely for these scores to correlate with designability. Second, modern folding models are known to overpredict alpha helices. Jovian sometimes simply adds more alpha helices connected by short coils. For example, this failure mode occurs in the ACE2 binding motif, whose source chain contains a large number of alpha helices. It seems likely that these generated scaffolds only register as designable because of the weaknesses of our computational metrics, which rely on folding models that also overpredict alpha helices. Finally, there are a few instances in scaffolding where the model samples proteins with high global similarity to the source chain. We attribute this fact to the presence of structural homologs in the training set that were not removed by sequence-based de-duplication, as this effect is generally eliminated with Jovian-FS.

5 Limitations and future work

In this work, we focus primarily on demonstrating how direction-conditioning extends the autoregressive paradigm to conditional protein structure generation. A promising orthogonal generation is improving tokenization; part of the reason our model is large by parameter count is it needs to reason through the complexity of the 600M ESM3 codebook and decoder. Because we implicitly learn the distribution of protein lengths, direction-conditioning is less suitable for tasks that require control over protein-size (e.g., miniaturization). An interesting direction is to study how to control and condition autoregressive structure models. We saw some success with size conditioning in early experiments but did not pursue it further.

Autoregressive modeling is very appealing in part because one can perform motif scaffolding with no additional training required. However, extending this paradigm to multi-motif scaffolding is nontrivial. Real-world tasks will involve proteins containing multiple functional domains and potentially unknown relative positions. While this is currently beyond the scope of existing autoregressive

models, it may be made possible by future work. Finally, while our bidirectional generation is an order of magnitude or more faster than most current diffusion models (see Figure 3), we do not reach the speed of most modern autoregressive language models because our formulation of bidirectional sampling is incompatible with kv-caching.

6 Conclusion

We present Jovian, a direction-conditioned autoregressive model capable of fast and accurate motif-scaffolding. This approach enables design without priors on functional site location or scaffold size. We hope our work inspires further exploration of autoregressive approaches in protein design and contributes to the broader advancement of generative biology.

Acknowledgments and Disclosure of Funding

R.D. gratefully acknowledges Markus Marks, Brian Hie, and Samuel Arnesen for technical advice, and Laura Luebbert, William Arnesen, Brian Hie, Sinan Ozbay, Markus Marks, Samuel Arnesen, Tinashe Handina, Tiffany Hung, Guruswami Ravi, and Chang Sun for providing feedback on the manuscript.

References

[1] Prabhu S Arunachalam, Alexandra C Walls, Nadia Golden, Caroline Atyeo, Stephanie Fischinger, Chunfeng Li, Pyone Aye, Mary Jane Navarro, Lilin Lai, Venkata Viswanadh Edara, et al. Adjuvanting a subunit COVID-19 vaccine to induce protective immunity. *Nature*, 594(7862):253–258, 2021.

[2] Marc J Lajoie, Scott E Boyken, Alexander I Salter, Jilliane Bruffey, Anusha Rajan, Robert A Langan, Audrey Olshefsky, Vishaka Muhunthan, Matthew J Bick, Mesfin Gewe, et al. Designed protein logic to target cells with precise combinations of surface antigens. *Science*, 369(6511):1637–1643, 2020.

[3] 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.

[4] Po-Ssu Huang, Scott E Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320–327, 2016.

[5] Sergey Ovchinnikov and Po-Ssu Huang. Structure-based protein design with deep learning. *Current opinion in chemical biology*, 65:136–144, 2021.

[6] Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, et al. Bindcraft: one-shot design of functional protein binders. *bioRxiv*, pages 2024–09, 2024.

[7] Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with SE(3) flow matching, October 2023. arXiv:2310.05297 [q-bio].

[8] Jason Yim, Brian L. Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. SE(3) diffusion model with application to protein backbone generation, May 2023. arXiv:2302.02277 [cs].

[9] 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*, 36:12489–12517, 2023.

[10] Felix Jung, Kevin Frey, David Zimmer, and Timo Mühlhaus. Deepstabp: a deep learning approach for the prediction of thermal protein stability. *International Journal of Molecular Sciences*, 24(8):7444, 2023.

[11] Brian L Hie, Varun R Shanker, Duo Xu, Theodora UJ Bruun, Payton A Weidenbacher, Shaogeng Tang, Wesley Wu, John E Pak, and Peter S Kim. Efficient evolution of human antibodies from general protein language models. *Nature biotechnology*, 42(2):275–283, 2024.

[12] Xi Han, Xiaonan Wang, and Kang Zhou. Develop machine learning-based regression predictive models for engineering protein solubility. *Bioinformatics*, 35(22):4640–4646, 2019.

[13] Zhuoqi Zheng, Bo Zhang, Kieran Didi, Kevin K Yang, Jason Yim, Joseph L Watson, Hai-Feng Chen, and Brian L Trippe. MotifBench: A standardized protein design benchmark for motif-scaffolding problems. *arXiv preprint arXiv:2502.12479*, 2025.

[14] Kieran Didi, Francisco Vargas, Simon V Mathis, Vincent Dutordoir, Emile Mathieu, Urszula J Komorowska, and Pietro Lio. A framework for conditional diffusion modelling with applications in motif scaffolding for protein design. *arXiv preprint arXiv:2312.09236*, 2023.

[15] Brian L Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

[16] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, and Lukas F. Milles. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023. Publisher: Nature Publishing Group UK London.

[17] Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of Many, One: Designing and Scaffolding Proteins at the Scale of the Structural Universe with Genie 2, May 2024. arXiv:2405.15489 [q-bio].

[18] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, February 2025.

[19] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[20] Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. DPLM-2: A Multimodal Diffusion Protein Language Model, October 2024. arXiv:2410.13782 [cs].

[21] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

[22] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[23] Guillaume Huguet, James Vuckovic, Kilian Fatras, Eric Thibodeau-Laufer, Pablo Lemos, Riashat Islam, Cheng-Hao Liu, Jarrid Rector-Brooks, Tara Akhound-Sadegh, Michael Bronstein, et al. Sequence-augmented se (3)-flow matching for conditional protein backbone generation. *arXiv preprint arXiv:2405.20313*, 2024.

[24] Pablo Gainza, Sarah Wehrle, Alexandra Van Hall-Beauvais, Anthony Marchand, Andreas Scheck, Zander Harteveld, Stephen Buckley, Dongchun Ni, Shuguang Tan, and Freyr Sverrisson. De novo design of protein interactions with learned surface fingerprints. *Nature*, 617(7959):176–184, 2023. Publisher: Nature Publishing Group UK London.

[25] Brian Kuhlman and David Baker. Exploring folding free energy landscapes using computational protein design. *Current opinion in structural biology*, 14(1):89–95, 2004.

[26] Michael R Shirts, David L Mobley, and Scott P Brown. Free-energy calculations in structure-based drug design. *Drug Design*, 1:61–86, 2010.

[27] Pablo Gainza, Freyr Sverrisson, Frederico Monti, Emanuele Rodola, Davide Boscaini, Michael M. Bronstein, and Bruno E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020. Publisher: Nature Publishing Group US New York.

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

[29] Benoit Gaujac, Jérémie Donà, Liviu Copoiu, Timothy Atkinson, Thomas Pierrot, and Thomas D. Barrett. Learning the Language of Protein Structure, January 2025. arXiv:2405.15840 [q-bio].

[30] 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.

[31] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

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

[33] Yann LeCun. Mathematical obstacles on the way to human-level ai. In *2025 Joint Mathematics Meetings (JMM 2025)*. AMS.

[34] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[35] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[36] Gustaf Ahdritz, Nazim Bouatta, Sachin Kadyan, Lukas Jarosch, Dan Berenberg, Ian Fisk, Andrew Watkins, Stephen Ra, Richard Bonneau, and Mohammed AlQuraishi. Openproteinset: Training data for structural biology at scale. *Advances in Neural Information Processing Systems*, 36:4597–4609, 2023.

[37] Mihaly Varadi, Damian Bertoni, Paulyna Magana, Urmila Paramval, Ivanna Pidruchna, Malarvizhi Radhakrishnan, Maxim Tsenkov, Sreenath Nair, Milot Mirdita, Jingi Yeo, et al. Alphafold protein structure database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic acids research*, 52(D1):D368–D375, 2024.

[38] 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.

[39] 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.

[40] Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning Up the Heat: Min-p Sampling for Creative and Coherent LLM Outputs, March 2025. arXiv:2407.01082 [cs].

[41] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration, February 2020. arXiv:1904.09751 [cs].

[42] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pages 8946–8970. PMLR, 2022.

[43] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*, 2022:500902, 2022.

[44] Bruce Jiang, Xiaoxiao Li, Amber Guo, Moris Wei, and Jonny Wu. RFdiffusion Exhibits Low Success Rate in De Novo Design of Functional Protein Binders for Biochemical Detection. *bioRxiv*, pages 2025–02, 2025. Publisher: Cold Spring Harbor Laboratory.

[45] 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.

[46] 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.

[47] Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. Se (3)-stochastic flow matching for protein backbone generation. *arXiv preprint arXiv:2310.02391*, 2023.

[48] Elisabeth Gasteiger, Eva Jung, and Amos Bairoch. Swiss-prot: connecting biomolecular knowledge via a protein database. *Current issues in molecular biology*, 3(3):47–55, 2001.

[49] Jason Yim, Andrew Campbell, Emile Mathieu, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Frank Noé, et al. Improved motif-scaffolding with se (3) flow matching. *ArXiv*, pages arXiv–2401, 2024.

[50] Jinrui Xu and Yang Zhang. How significant is a protein structure similarity with tm-score= 0.5? *Bioinformatics*, 26(7):889–895, 2010.

[51] Christine A Orengo, Alex D Michie, Susan Jones, David T Jones, Mark B Swindells, and Janet M Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

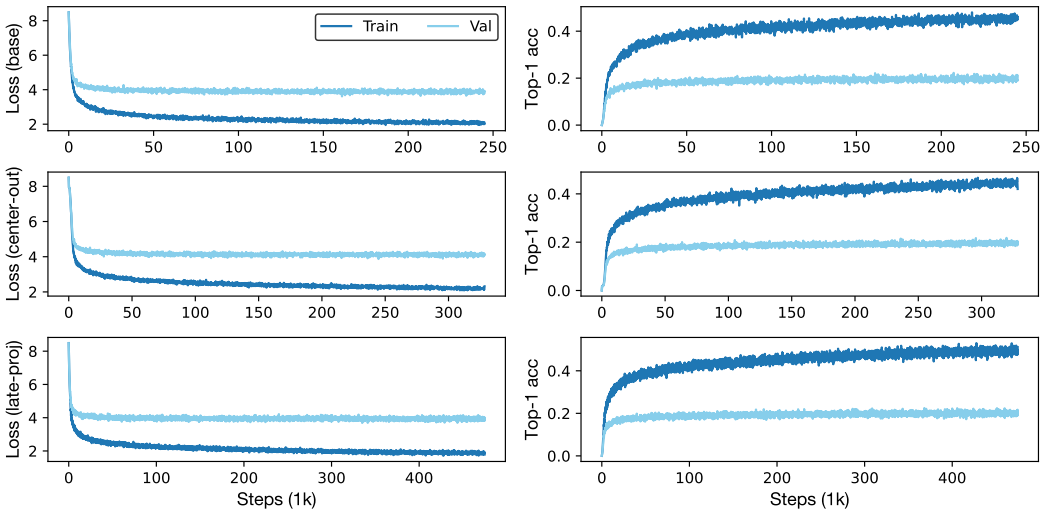

Figure 5: Training curves (left) and top-1 accuracy (right) for direction conditioned base run (top), center-out (middle), and late projection (bottom). Although we reach similar top-1 per token accuracies on all three runs, the base run still has the best designability.

A Impact Statement

AI for biology has the potential to significantly benefit humanity through novel therapeutics, biosensors, and more. Like all emergent technologies, it is important to be aware of potentially harmful applications in this field stemming from the misuse of AI. To that end, the authors strongly support the principles outlined in Responsible AI x Biodesign.

B Model architectures

In this section, we discuss the details of our model architecture and training process. In training our models, we found that many best practices and heuristics that have emerged in language and vision did not carry over to our experiments. We briefly explain these heuristics and hope that they will help guide future work. Figure 5 shows our loss and top-1 accuracy. Training for a large number of steps is important; although the accuracy appears to quickly converged, designability emerges at 100k steps. This likely would show up if we measured top-k accuracy for $k > 1$, but due to the cost of each training run (4 days $\times$ 8 A100s $\times$ \$1.79 / GPU / hour), we did not repeat the base training run with these statistics logged.

In these architecture modifications, we mostly optimize for training convergence. To save computational resources, when the effect of each of these ablations on convergence rate was clear, we stopped the run. This was primarily because the metrics under consideration (e.g., designability) only emerged at around 314M parameters, which took about 3-4 days on 8A100s.

Normalization We find that layer norms provide equivalent or better performance to QK-norm or RMSNorm.

MLP dilation For a fixed parameter count, a dilation factor of 2 with a higher input dimension provided faster convergence than a dilation factor of 4 with lower input dimension.

Rotary embeddings In addition to being important for motif scaffolding, rotary embeddings consistently converged faster than learned embeddings.

Activation functions We noticed no difference between GELU, ReLU, and ReLU^2 activations.

Training took 3.5 days on 8A100s. All hyperparameters are shown in Table 3. Our final model is 314M parameters with 2.12 Gflops (measured per token).

Hyperparameter	Value
Dimension	1280
Hidden dimension	2560
Batch size	32
Layers	24
Heads	16
Activation	GELU
Warmup iterations	1000
Gradient clip	1.7

Table 3: Model and training hyperparameters for base Jovian used in experiments.

B.1 Dataset

Our dataset is composed of the Open Protein Set [36] and a cleaned version of Swissprot [48], where we discarded all structures with pLDDT < 80. We experimented with two clustering methods, MMSeqs [38] and Foldseek [39]. We set MMSeqs clustering similarity to 0.8 and Foldseek clustering similarity to 0.8. The trained models are labeled Jovian-MM and Jovian-FS respectively. This choice has important downstream consequences. The goal of clustering is to remove sequence and structure homologs that may lead to the model presenting as learning something novel, while in reality it simply regurgitates elements of the training set and cannot generalize. Many works use MMSeqs to cluster sequences into train/test/val splits, under the assumption that sequence data corresponds well with structural data [29].

However, while the overall performance in Table 2 of Jovian-FS and Jovian-MM is quite similar, there are several concerning trends in our motif scaffolding results. Motifs like `1ldb_A` that have very strong performance on Jovian-MM but poor performance on Jovian-FS seem to suggest that there are structural homologs leaking into the training set that Jovian-MM misses. This motif seems to be particularly challenging for most methods; it is in the FrameFlow training set (MotifBench was released well after FrameFlow, so they do not benchmark against it) and RFDiffusion likely benefits from the pretraining of RosettaFold. Preventing these types of data leaks will be critical as we continue to hill-climb motif benchmarks.

B.1.1 Motif scaffolding

For motif scaffolding, we use a joint benchmark composed of structures from MotifBench [13] and ESM3 [18]. We take all single motif chains with motif length longer than three amino acids. For multi-motif scaffolding, we take a single motif (generally the longest). As we mention in the main text, both MotifBench and ESM are very recent publications, so the models we benchmark against may have seen this data. We were unable to find exact training/test splits for most other methods. Retraining other methods from scratch would have been computationally infeasible (and not possible for RFDiffusion), so we report results as is.

C Alternative modeling paradigms

While studying ways to extend autoregressive modeling to motif scaffolding, we explored a variety of paradigms. In this section, we briefly describe each paradigm we explored and discarded. Some of these we pursued only briefly before it became apparent they would not work; we include all descriptions here to inform future autoregressive structure work. Table 4 shows some metrics for other methods with affiliated discussion.

C.0.1 Center-out generation

In center-out generation, we predict two tokens at each step, one to the left and one to the right. This can include an `<EOS>` token; after inference, we can prune the output of all tokens within the `<EOS>` range (this is similar to the strategy [29] employs due to their use of Jax). We depict this architecture, along with unconditional and conditional generation, in Figure 6.

	Params	Designability	Diversity	Novelty
Late-proj (BiD)	T=1.0, nucleus	0.137	0.445	0.858
Late-proj (For)	T=1.0, nucleus	0.195	0.501	0.781
Late-proj (Rev)	T=1.0, nucleus	0.184	0.457	0.674
Center-out	T=1.0, minp	0.406	0.637	0.721

Table 4: Designability, diversity, and novelty for late-projection methods and center-out generation. Late-stage prediction does not have the desired effect on resolving the distribution mismatch described in Section 4.2.1. We did not observe a significant difference between late-stage prediction and direction-conditioning, but we did not conduct a more thorough exploration beyond 1-2 sampling setups that were known to be problematic in the direction-conditioned method. Center-out generation has good designability but suffers from various undesirable pathologies, which the standard metrics miss and we describe in Appendix C.

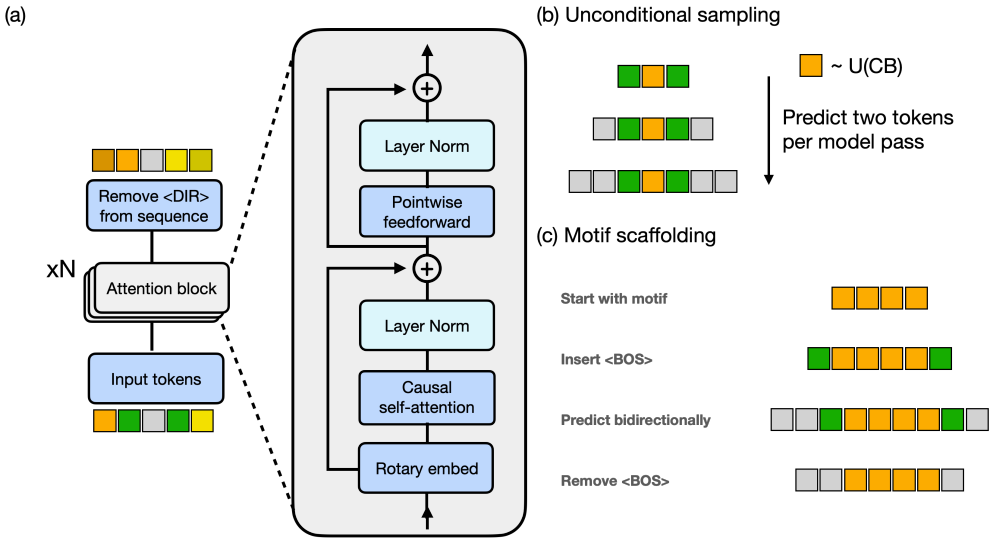

Figure 6: (a) **Center-out architecture**. There is no direction conditioning, and instead we predict middle out, adding a single token to the left and a single token to the right at each iteration. We insert two tokens, `<BOS-L>` and `<BOS-R>`, and add a learnable token to indicate whether the model should generate left or right. (b) **Unconditional generation** Similar to Jovian, we sample unconditionally by randomly selecting a token as the "motif" from the codebook, then sampling to the left and right. (c) **Motif scaffolding**. We perform motif scaffolding by repeatedly sampling tokens to the left and right (possibly crossing over an `<EOS>` token), then removing all special tokens in the final step.

Although this makes sense in principle and we achieved decent designabilities (see Table 4, we noticed multiple pathological cases occur in this setup – very short proteins, lots of alpha helices, etc. This would also require motif-specific finetuning, so we abandoned it in favor of our current, simpler approach. While we did not do a thorough comparison because of the appeal of direction-conditioning, empirically the results seemed much less stable.

It is interesting that our top-1 per token accuracies are similar for both direction conditioning and center-out generation. Predicting two tokens is a much harder task than predicting a single subsequent token. Explicitly, with center-out prediction, we're sampling from $P(h_{\text{left}}, h_{\text{right}}|\text{motif})$, whereas with direction-conditioned prediction, we sample iteratively from $P(h_{\text{right}}|\text{motif})$ and $P(h_{\text{left}}|\text{motif}, h_{\text{right}})$. It is clear that the first task is at least as hard as the second, since one simply "ignore" the task of predicting h_{left} and be left with sampling from $P(h_{\text{right}}|\text{motif})$. This observation is aligned with the challenges the NLP field has seen in multi-token prediction. Similar to methods like [49] and [17], this requires that we randomly sample pseudo-motifs at generation time.

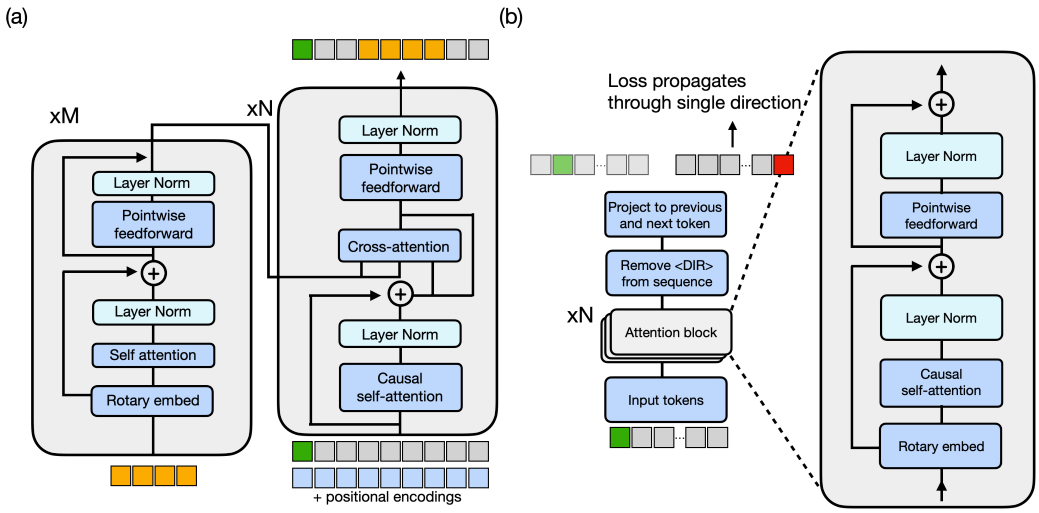

Figure 7: (a) **Cross attention mechanism**. We encode the motif using M layers of unmasked self-attention, then use a cross attention layer to condition the generation process (which is entirely left-to-right autoregression) on the motif. (b) **Late-stage prediction**. Instead of conditioning on the direction, we have two output heads that predict next and previous tokens (in practice, previous token is next token with order flipped).

C.0.2 Cross-attention conditioning

Perhaps the most straightforward extension from the natural language paradigm is to simply condition the generation process on the encoded motif using a cross attention layer. This is depicted in Figure 7. While this still seems intuitively challenging because the generation process must reason over tokens very far in the future, the actual planning is not very sophisticated. The model just needs to insert the motif (or something close to the motif, since the tokens may change slightly, especially near the motif boundaries) into the generation process. In practice, we observe that training is highly unstable and we abandoned this approach very early, see Figure 8.

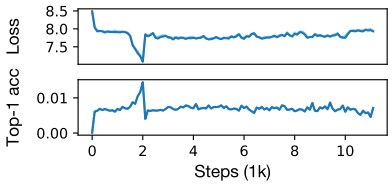

Figure 8: Instabilities observed in validation loss (top) and top-1 per token accuracy (bottom) in training motif scaffolding with cross attention.

C.0.3 Late-stage prediction

To mitigate the distribution mismatch as described in Section 4.2.1, we also explored a single forward model with two projection heads, one for forward mode and one for reverse mode. The intuition behind this is because the conditioning is delayed to the very end, the model is forced to learn a joint representation for forward and reverse and less likely to generate two entirely different distributions. This is depicted in Figure 7(b). This significantly reduces designability, with peak designability generally around 10%.

A possible explanation for this is that the forward and reverse structural distributions are genuinely different. This would be quite interesting scientifically, since although proteins have a well defined ordering, there isn't necessarily a reason their *structure* should (since protein functions are dependent on 3D space). While we raise this possibility, this is *not* a claim we make. It is hard to make scientific claims from the results of a learned model, since these results could easily be an artifact of the training process or the tokenization stage (having said that, the tokenization scheme the ESM3 tokenizer is bidirectional with no absolute sequence information, so there isn't an obvious issue). Given that bidirectional models provide an elegant way to merge autoregressive generation with motif scaffolding, we raise this point because these distributional issues merit further study.

C.1 Tokenizer

The ESM3 tokenizer uses two layers of SE(3)-invariant geometric attention, a codebook size of 4096, and 24 transformer layers. The encoder contains 30,116,224 parameters, and the decoder contains 618,552,905 parameters. Because we use the tokenizer without modification, we defer a detailed explanation of the ESM3 tokenizer to [18].

D Metrics

In this section, we provide details of our metrics calculations and describe specific considerations at play when presenting metrics.

Designability Following [18], we use ESM-IFv1 for inverse folding and ESMFold for folding. We clone the official ESM Github repo. We perform both best-of-one and best-of-eight sampling for inverse folding. We use temperature 10^{-6} for best-of-one sampling and temperature 0.1 for best-of-eight sampling. We report best-of-eight in the main text. Genie 2 only generates Cα coordinates, so we use ProteinMPNN in Cα only mode for this particular case. Because this is much less constraining (i.e., we're missing a rotation to describe a backbone, so there are $3L$ missing constraints), this tends to inflate designability values.

Diversity We report the average pairwise TM-score using the Python package tmtools, which exposes an interface for TM-align [45, 50]. The TM-score between a template and target structure with L_{cmn} residues in common is given by

$$\text{TM-score} = \max \left[\frac{1}{L_{\text{tgt}}} \sum_{i}^{L\text{cmn}} \frac{1}{1 + \left(\frac{d_i}{d_o(L_{\text{tgt}})} \right)^2} \right] \tag{1}$$

Here d_i is the distance between two corresponding residues after two structures are aligned, and $d_o(L) = 1.24\sqrt[3]{L - 15} - 1.8$ normalizes distances. A TM-score of 1.0 means two structures are identical. The TM-score and RMSD are two common metrics of structural similarity. While the TM-score is often preferred because it is both independent of distances scales and a better global similarity metric, RMSD < 2.0Å is generally a better marker of designability.

Novelty Following [29], we use Foldseek [39] to report average TM score of the closest match (i.e., the average max TM score) in the CATH database [51].

For all metrics, we compute errors as $s/\sqrt{n}$, where n is the number of sample points (which can vary, as both diversity and novelty are only measured with respect to designable samples), and s^2 is the standard sample variance estimator for σ^2.

D.1 Challenges with metrics

Computational metrics for protein design are an ongoing and unresolved issue. We encountered several challenges in making a fair comparison between fixed-length diffusion models and variable length autoregressive models. In this section, we discuss the various considerations and reasoning behind our choices. Our goal is to both promote an honest discussion about machine learning for protein design and to avoid incremental benchmark chasing.

Our use of ESM-IF instead of ProteinMPNN (which others have used) has a significant effect on designability. Empirically, we observed that ProteinMPNN seems to improve designability relative to using ESM-IF. Because metrics trade off against each other, we are not as concerned with absolute performance so much as relative performance in certain regimes (e.g., long sequences). We report Cα only designability for Genie 2 because it does not generate any other atoms, so we cannot consistently use ESM-IF; as we note in the main text, though, this is a much easier task than computing the full backbone. Note that ESM3 [18] describes the VQ training process as decoding all-atom structures, but the public decoder only returns backbone atoms.

We also face the question of how one should compare variable-length autoregressive models to diffusion models. There are two primary considerations. First, a major strength of autoregressive

models is that significant work has gone into their performance at long context windows. However, many diffusion models are trained at relatively short windows. Works like [17] claim state of the art performance with high designabilities, but only measure up to 256 residues (which, in fairness, is fairly standard across most diffusion models). Metrics like diversity and novelty are computed with respect to designable sequences, so breakdowns at longer sequence lengths (e.g., repeatedly generating the same designable sequence at longer lengths) are not visible. For this reason, we report designability at all sequence lengths and designability at sequence lengths between 256 and 512.

We also need to be cautious while handling diversity. MaxCluster (which has been used in prior works to compute diversity, see [8, 29], tends to report proteins of different lengths in completely different clusters. This would unfairly inflate the diversity of our generations. Diffusion models avoid this problem by just generating a large number of samples at a fixed length. [29] generates 40,000 sequences to ensure that there are sufficiently many designable sequences to do this comparison. Measuring designability for that many sequences is computationally challenging, particularly if one uses best-of-N sampling during the round-trip inverse folding and refolding process. We instead report the average pairwise TM-score, similar to the reported metrics in [23]. This comes with its own set of issues; for instance, one could potentially game the metric by normalizing with respect to the larger chain, then just generating functional sites and adding very long alpha helices to the end. To avoid this, we report averages between both normalization schemes. While our metrics as reported in Table 2 outperform every model except for the heavily pretrained RFDiffusion, we explicitly do not claim these results to be state-of-the-art. A proper comparison of of fixed vs variable length generations would likely need to involve generating a tremendous number of sequences.

As we comment in the main text, novelty favors methods with heavy inductive biases built in which can learn from a small amount of data. However, deep learning progress has largely been driven by general purpose models that can learn from large amounts of data at scale. Similar to how RNNs may outperform transformers for small amounts of data, yet are vastly outstripped when training on larger data corpuses, we should be careful to not make definitive conclusions about performance at small scale. Regardless, we demonstrate that by changing the diversity of our data corpus slightly, we can vastly improve the novelty.

While protein design has rapidly advanced in tandem with machine learning, the ultimate metric of success will still be experimental evaluation. Developing sophisticated computational metrics is important to encourage faster experimentation, design cycles, and machine learning innovation. Particularly as we shift towards a new domain (variable length generation), every metric will carry problems; we have tried to acknowledge these and report results that fairly represent our findings.

E Extended benchmarks

In this section, we present additional benchmarking beyond the main text.

E.1 Unconditional generation

Figure 9 shows scRMSD comparisons across models as a function of protein sequence length. Across all models, Jovian consistently outperforms both FrameDiff and Chroma at long sequence lengths. While RFDiffusion has more points at longer sequences, this is primarily because Jovian is not as controllable with respect to protein size, so we cannot force it to generate a large number of long sequence samples. The distribution suggests that if we simply ran it longer (which we can do relatively easily given the cheap inference costs – the primary bottleneck is inverse folding and refolding), we would generate more designable samples in the long-sequence regime. Figure 10 shows secondary structure comparisons against all models. We use biotite's `annotate_sse` function, and take an average across a chain. Because both FrameDiff and Chroma struggle to produce long samples, we use a relaxed designability criterion of scRMSD < 4Å. Our goal is not to measure designability, but rather to consider the *distribution* of alpha helices and beta strands (of course, we cannot deviate too far from the standard criterion of 2Å, since very unrealistic samples probably have a completely different secondary structure distribution). Even with this relaxation, neither Chroma nor FrameDiff can easily generate long samples. Interestingly, while RFDiffusion and Jovian have similar structure distributions at long sequence lengths, Jovian avoids the mode collapse that RFDiffusion experiences at *short* sequence lengths.

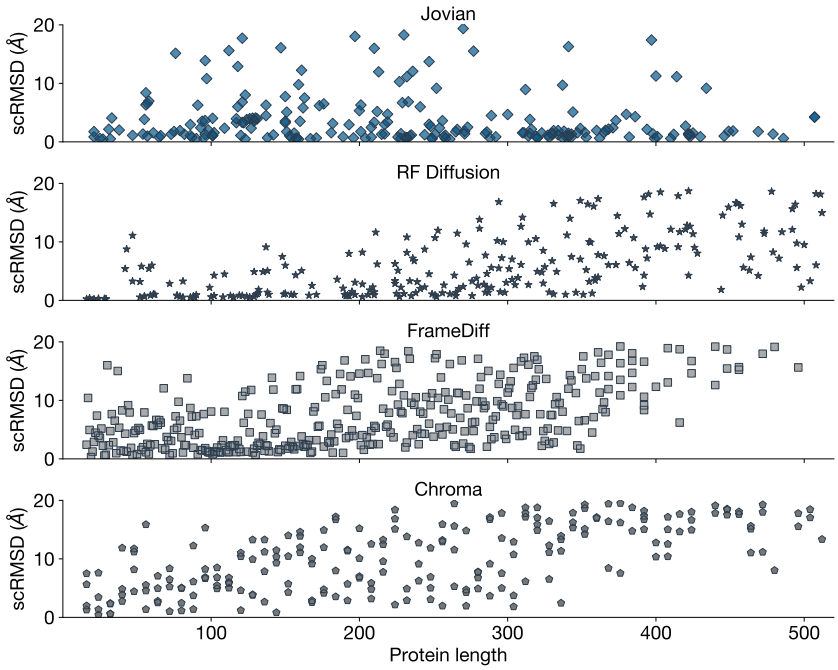

Figure 9: scRMSD comparisons across protein lengths.

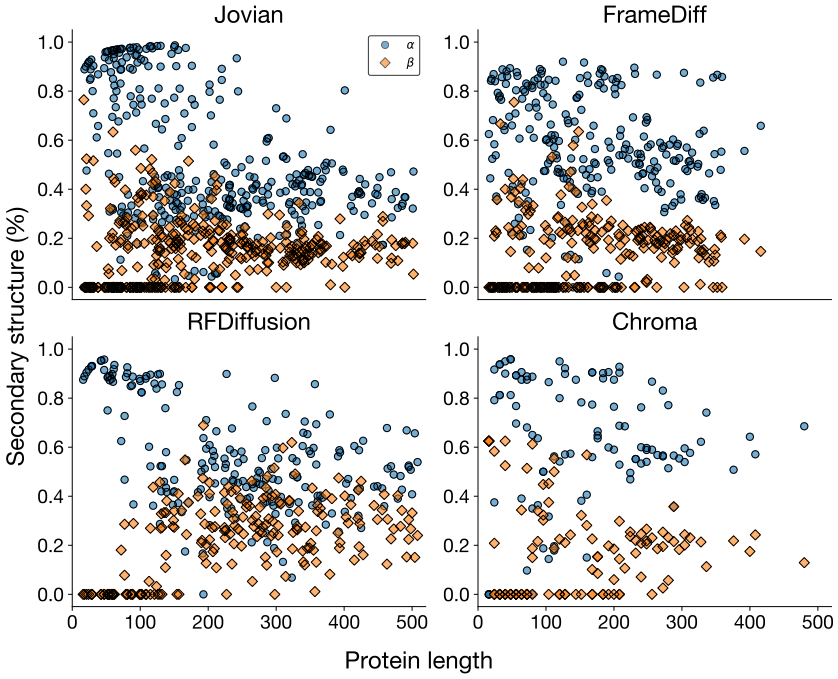

Figure 10: Secondary structure comparisons across models, with scRMSD < 4Å. Jovian is more robust than RFDiffusion at short sequences and maintains designability at longer sequences compared to FrameDiff and Chroma. Note that not all subfigures have the same number of sample points, both for designability filter reasons and because this version of Jovian does not have length conditioning, so we generate a large (≈500) number of points to get a reasonable distribution across protein sizes. The relatively sparse number of points in Chroma, however, is primarily due to lower designability.

E.2 Motifs

Almost all current motif scaffolding models use the pseudo-motif training procedure outlined in [49], where small protein segments are randomly sampled as motifs and used to train a motif scaffolding model. [49] do introduce a motif-guidance model which requires no additional training. However, they explicitly recommend the “amortization" model (which follows the pseudomotif strategy) and the guidance model significantly underperforms both Jovian and the amortization model.

Figure 11 shows the designability and number of designs for a fixed inference budget on all motifs in the ESM + MotifBench benchmarks. We compute the bottom plot by providing each model with a compute budget (in this case, 0.25 V100-hours), then randomly sampling from the model generations until the compute budget is exhausted. This is why for some motifs (e.g., `1lcc_A`), RFDiffusion has zero designs despite having nonzero designability; the designability is sufficiently low and inference is sufficiently expensive that it never successfully samples a designable sequence. The downsampling in Figure 4 is random.

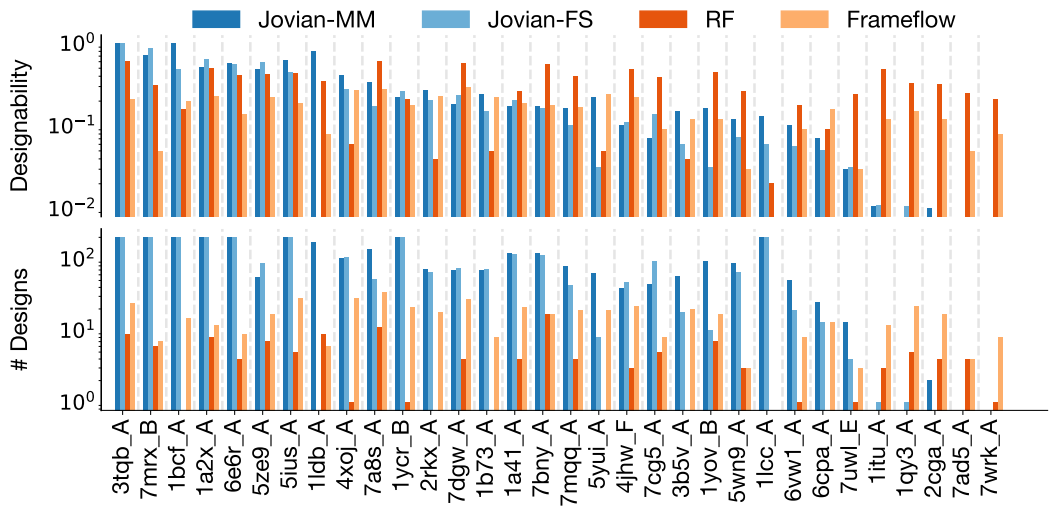

Figure 11: Motif scaffolding results on the full ESM + MotifBench validation set.

Table 5: Metrics for a variety of sampling configurations.

Temperature	Threshold	Sample method	Direction	Designability	Diversity	Novelty
1.00	0.90	nucleus	Forward	0.43 ± 0.05	0.440 ± 0.003	0.70 ± 0.03
1.00	0.90	nucleus	Reverse	0.34 ± 0.05	0.432 ± 0.004	0.75 ± 0.03
1.00	0.90	nucleus	Bidirectional	0.062	–	0.623
1.00	0.95	nucleus	Forward	0.41 ± 0.05	0.424 ± 0.003	0.74 ± 0.03
1.00	0.95	nucleus	Reverse	0.29 ± 0.05	0.418 ± 0.004	0.79 ± 0.03
1.00	0.95	nucleus	Bidirectional	0.073	–	0.703
1.00	0.05	minp	Forward	0.49 ± 0.05	0.406 ± 0.003	0.74 ± 0.03
1.00	0.05	minp	Reverse	0.41 ± 0.05	0.441 ± 0.005	0.76 ± 0.04
1.00	0.05	minp	Bidirectional	0.18 ± 0.04	0.424 ± 0.007	0.79 ± 0.05
1.00	0.10	minp	Forward	0.47 ± 0.05	0.422 ± 0.003	0.74 ± 0.03
1.00	0.10	minp	Reverse	0.34 ± 0.05	0.452 ± 0.006	0.74 ± 0.05
1.00	0.10	minp	Bidirectional	0.25 ± 0.04	0.444 ± 0.005	0.70 ± 0.04
1.00	0.20	minp	Forward	0.40 ± 0.05	0.409 ± 0.003	0.73 ± 0.03
1.00	0.20	minp	Reverse	0.18 ± 0.04	0.50 ± 0.02	0.77 ± 0.09
1.00	0.20	minp	Bidirectional	0.31 ± 0.05	0.442 ± 0.005	0.72 ± 0.04
0.50	0.90	nucleus	Forward	0.30 ± 0.05	0.435 ± 0.006	0.69 ± 0.03
0.50	0.90	nucleus	Reverse	0.08 ± 0.03	0.44 ± 0.04	0.8275 ± 0
0.50	0.90	nucleus	Bidirectional	0.21 ± 0.04	0.451 ± 0.008	0.72 ± 0.04
0.50	0.95	nucleus	Forward	0.34 ± 0.05	0.435 ± 0.004	0.76 ± 0.03
0.50	0.95	nucleus	Reverse	0.20 ± 0.04	0.44 ± 0.01	0.80 ± 0.07
0.50	0.95	nucleus	Bidirectional	0.23 ± 0.04	0.435 ± 0.007	0.75 ± 0.04
0.50	0.05	minp	Forward	0.27 ± 0.05	0.439 ± 0.007	0.60 ± 0.03
0.50	0.05	minp	Reverse	0.08 ± 0.03	0.50 ± 0.04	0.64 ± 0.01
0.50	0.05	minp	Bidirectional	0.22 ± 0.04	0.429 ± 0.005	0.73 ± 0.04
0.50	0.10	minp	Forward	0.27 ± 0.05	0.459 ± 0.008	0.67 ± 0.04
0.50	0.10	minp	Reverse	0.021	–	0.806
0.50	0.10	minp	Bidirectional	0.29 ± 0.05	0.414 ± 0.005	0.73 ± 0.04
0.50	0.20	minp	Forward	0.21 ± 0.04	0.49 ± 0.01	0.59 ± 0.03
0.50	0.20	minp	Reverse	0.0	–	–
0.50	0.20	minp	Bidirectional	0.30 ± 0.05	0.407 ± 0.004	0.77 ± 0.03
1.50	0.90	nucleus	Forward	0.09 ± 0.03	0.44 ± 0.02	0.81 ± 0.08
1.50	0.90	nucleus	Reverse	0.10 ± 0.03	0.40 ± 0.01	0.73 ± 0.07
1.50	0.90	nucleus	Bidirectional	0.0	–	–
0.20	0.20	minp	Forward	0.15 ± 0.04	0.53 ± 0.03	0.69 ± 0.04
0.20	0.20	minp	Reverse	0.0	–	–
0.20	0.20	minp	Bidirectional	0.34 ± 0.05	0.414 ± 0.004	0.71 ± 0.04
0.20	0.10	minp	Forward	0.14 ± 0.04	0.61 ± 0.03	0.62 ± 0.02
0.20	0.10	minp	Reverse	0.0	–	–
0.20	0.10	minp	Bidirectional	0.35 ± 0.05	0.415 ± 0.003	0.80 ± 0.03
0.20	0.05	minp	Forward	0.14 ± 0.04	0.58 ± 0.03	0.63 ± 0.04
0.20	0.05	minp	Reverse	0.0	–	–
0.20	0.05	minp	Bidirectional	0.38 ± 0.05	0.422 ± 0.003	0.77 ± 0.03
0.20	0.95	nucleus	Forward	0.16 ± 0.04	0.55 ± 0.03	0.62 ± 0.03
0.20	0.95	nucleus	Reverse	0.0	–	–
0.20	0.95	nucleus	Bidirectional	0.30 ± 0.05	0.421 ± 0.004	0.78 ± 0.04
0.20	0.90	nucleus	Forward	0.16 ± 0.04	0.52 ± 0.03	0.63 ± 0.03
0.20	0.90	nucleus	Reverse	0.0	–	–
0.20	0.90	nucleus	Bidirectional	0.38 ± 0.05	0.406 ± 0.003	0.70 ± 0.04
1.50	0.20	minp	Forward	0.48 ± 0.05	0.441 ± 0.003	0.72 ± 0.03
1.50	0.20	minp	Reverse	0.40 ± 0.05	0.463 ± 0.006	0.76 ± 0.04
1.50	0.20	minp	Bidirectional	0.23 ± 0.04	0.471 ± 0.007	0.84 ± 0.02
1.50	0.10	minp	Forward	0.32 ± 0.05	0.460 ± 0.005	0.77 ± 0.04
1.50	0.10	minp	Reverse	0.29 ± 0.05	0.439 ± 0.005	0.78 ± 0.04
1.50	0.10	minp	Bidirectional	0.09 ± 0.03	0.48 ± 0.01	0.76 ± 0.05
1.50	0.05	minp	Forward	0.21 ± 0.04	0.449 ± 0.007	0.78 ± 0.04
1.50	0.05	minp	Reverse	0.14 ± 0.04	0.412 ± 0.007	0.65 ± 0.07
1.50	0.05	minp	Bidirectional	0.021	–	0.946
1.50	0.95	nucleus	Forward	0.053	–	0.700
1.50	0.95	nucleus	Reverse	0.010	–	0.947
1.50	0.95	nucleus	Bidirectional	0.0	–	–

F Sampling results

Table 5 contains metrics on a variety of sampling methods, temperatures, directions, and thresholds. As collecting these metrics is computationally expensive, we did not generate a large number of structures. Some some configurations we thus lack sufficiently many sample points to generate error bars or diversity scores; we exclude these. We do not view this data as necessarily conclusive. Having said that, there are two interesting points of note. First, we observe the novelty-designability tradeoff that we describe in the main text. Second, distribution mismatches (Section 4.2.1) mean bidirectional sampling is generally lower designability than either forward or reverse sampling.

G Benchmarking

In this section, we describe our generation process for each model we compare against. We include pseudocode/code snippets for the less well-established methods to aid with replication.

G.1 RFDiffusion

We cloned the public repository `github.com/RosettaCommons/RFdiffusion`.

G.1.1 Unconditional generation

We sweep target lengths $L \in \{16, 24, \ldots, 512\}$ skipping by 8 and generate 4 backbones per length. We use the default script `scripts/run_inference.py`.

G.1.2 Conditional generation

We sweep four target lengths $L \in \{64, 128, 256, 384\}$. For each length we uniformly sample 100 random placements for the motif within the central 60% of the chain. We use the script `scripts/run_inference.py` with the motif specified using the `contigs` parameter.

G.2 FrameDiff/FrameFlow

We cloned the public repositories `https://github.com/jasonkyuyim/se3_diffusion` and `https://github.com/microsoft/protein-frame-flow`.

G.2.1 Unconditional generation

We used the default inference script `experiments/inference_se3_diffusion.py` from `https://github.com/jasonkyuyim/se3_diffusion` and modified the default `config/inference.yaml` settings for the sampling block. We step between sequence length 16 and 512 with step size 8 and sample 4 backbones per sequence length.

G.2.2 Conditional generation

We performed conditional generation using the default Hydra-configured conditional inference script `experiments/inference_se3_flows.py -cn inference_scaffolding` from `https://github.com/microsoft/protein-frame-flow`. This loads the configuration file `configs/inference_scaffolding.yaml` which builds on the default `configs/_inference.yaml`. We modified the sampling-related parameters in the default `configs/_inference.yaml`.

We follow the same format described in the Github repository to fill out the new `motif_scaffolding/benchmark_targets.csv` with 100 rows per motif per target length $L \in \{64, 128, 256, 384\}$. For each row, there are four columns: the `target` column is set to `f"{motif_name}_{L}_{index}"` with indices 0 to 99 for every motif and target length combination, the `contig` column is generated using the same process as the contig variable from Section G.1.2, the `length` column is set to the target length `f"{L}-{L+1}"`, and the `motif_path` column stores the path to the PDB file with the motif.

G.3 VQ3D

We cloned the public repository `https://github.com/instadeepai/protein-structure-tokenizer`. A direct timing comparison is tricky because the sampler is written in JAX: the first call JIT-compiles the pmapped GPT and incurs a one-off XLA build, and the model itself always starts from a prompt of `<BOS>` tokens of fixed width `BLOCK_SIZE`=512 for all sequences in the batch. The sampler iteratively overwrites all of these positions and later truncates everything right of the first `<EOS>` for each sequence, producing a variable-length set of token sequences. Backbone coordinates are generated in a *separate* decoding pass using these tokens. We record wall-clock time after compilation for the token-generation stage until an `<EOS>` token is generated (`BATCH`=1).

G.3.1 Unconditional generation

Since the model always produces `BLOCK_SIZE=512` structural tokens which are then truncated after the first `<EOS>`, the final backbone length is data-driven and cannot be preset. Instead of directly varying length, we generate 256 backbones for each sampling temperature $T \in \{0.2, 0.4, 0.6, 0.8\}$ using the `scripts/gpt_generation.py` script.

In our experiments, we were unable to generate any designable backbones using the public weights for VQ3D. The original work [29] reports designability 0.41, but without generated samples, we could not compute novelty, diversity, or standard errors.

G.3.2 Conditional generation

The released code has no mechanism for inserting or constraining a motif, so VQ3D was benchmarked only in the unconditional setting. As we note in the main text, motif scaffolding is impossible given the VQ3D setup.

G.4 Chroma

We cloned the public repository `https://github.com/generatebio/chroma`. In both settings below we explicitly selected the `"langevin"` SDE.

G.4.1 Unconditional generation

We generated $n = 100$ backbones for each target length $L \in \{64, 128, 192, 256, 384, 512\}$ using the `chroma.sample` function as described in the repository.

G.4.2 Conditional generation

For each of four target lengths $L \in \{64, 128, 256, 384\}$ we uniformly sampled 100 start positions s within the central 60% of the chain $0.2L \leq s \leq 0.8L - m$, inserted the motif fragment of length m into residues $[s, s + m)$ of an otherwise-blank backbone. We then locked those residues with a `SubstructureConditioner` and generated one Chroma scaffold for every placement using `conditioners.SubstructureConditioner` and specifying `conditioner` in `chroma.sample` as described in the repository.

G.5 Genie 2

We cloned `https://github.com/aqlaboratory/genie2`. It is important to note that Genie 2 outputs only Cα coordinates rather than the full backbone.

G.5.1 Unconditional generation

The backbone checkpoint released `epoch = 40.ckpt` was first downloaded to `genie2/results/base/checkpoints/`. For each target length $L \in \{64, 128, 192, 256, 384, 512\}$ we then generated $n = 100$ carbon-alpha traces using the script `genie/sample_unconditional.py` by specifying `-min_length L` and `-max_length L`.

G.5.2 Conditional motif scaffolding

While Genie 2 can perform conditional motif scaffolding, we excluded it from the results as the lack of backbone coordinates (which can significantly affect motif viability due to e.g., steric clashes) makes it very challenging to do a fair comparison to other methods.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our primary claim is the development and characterization of an autoregressive model for motif scaffolding; these results are clearly displayed in Section 4 and to our knowledge have not appeared elsewhere.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we discuss limitations in Section 5; we specifically point to multi-motif scaffolding, which is a very important problem with which our method cannot naively engage. We also report on the weaknesses of our metrics in Section 4.2.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We explicitly describe our architecture in Figure 2 (which is based on very established components), our model/training setup in Appendix B, and our data in Appendix B.1. This is sufficient to replicate all of our experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All of our data is from public sources; we include preprocessing scripts and model definition/training scripts in the code submission. We will open-source all code, instructions, and model checkpoints once blind review is completed.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Justification: All training details are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error estimates for all of our numerical values in both Table 2 and Table 5. The only case where we do not include error bars is Figure 3d, where it would be too computationally expensive to generate that many sequences across numerous lengths and models to get reasonable statistics.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the GPU type and number of hours used for each experiment in the paper, and generation plots in Figure 3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the Neurips code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We comment on the ethics of biodesign, and refer to the Responsible AI x Biodesign principles as the guiding principles behind our work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not believe our model has a high risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The primary actors are the contributors to the PDB, Swissprot, and the AFDB, all of whom are acknowledged and cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not perform any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.