BRIDGING STRUCTURE TO SEQUENCE VIA LOCAL GEOMETRIES FOR PROTEIN INVERSE FOLDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Inverse folding—designing amino acid sequences for a given protein structure—is a core task in protein engineering. Recently, masked diffusion protein language models (PLMs) have shown great potential, typically decoding residues in a confidence-first order. While decoding order has drawn growing attention in diffusion large language models (dLLMs), it has remained largely unexplored in protein language models. In this work, we evaluate decoding orders through a unified sequence–structure pass@K metric, assessing a model's ability to generate sequences that are both native-like and designable. Unlike in natural language dLLMs, where decoding order reshapes sampling coverage, we find that in inverse folding it leaves coverage unchanged but steers a trade-off: confidence-first decoding yields highly native-like but poorly designable sequences. Entropy analyses trace this to premature commitment: high-uncertainty positions are deferred and resolved by sequence statistics rather than structural geometry. We show the trade-off can be addressed at two levels. At inference time, a local-geometry guided decoding order resolves structurally prioritized regions first while sampling freely within them. At training time, upweighting high-uncertainty positions during fine-tuning biases the model toward structural constraints at exactly those positions. We instantiate both interventions in Hier-IF, a structure-conditioned masked diffusion model for inverse folding.On CATH, the resulting model improves designability and sequence recovery simultaneously over its confidence-decoded counterpart, with consistent gains on the out-of-distribution TS50 and TS500 benchmarks and strong performance on motif-scaffolding. This plug-and-play framework enables both inference-time control and training-time enhancement.

1 INTRODUCTION

Designing amino acid sequences that fold into a prescribed protein backbone—inverse folding—is a central problem in protein engineering, with applications in enzyme and antibody design (Street & Mayo, 1999; Fersht et al., 1992; Dreyer et al., 2023). Because many sequences can realize similar structures, inverse folding is inherently one-to-many: a successful design should be evaluated not only by native-sequence recovery, but also by whether it folds back to the target structure.

Recent work on diffusion language models has shown that generation order can reshape the solutions reached at inference time (Ni et al., 2026). This provides an inexpensive control lever beyond pretraining. Protein language models, including diffusion-based inverse-folding models (Lin et al., 2023; Hayes et al., 2025; Wang et al., 2024a;b; Hsieh et al., 2025), have achieved strong sequence-design performance, but their decoding orders are typically inherited from generic language-model defaults. Whether these orders are appropriate for structure-conditioned protein generation remains unclear.

We present, to our knowledge, the first systematic study of decoding order in protein inverse folding. Under a budget-exact sampler, we compare confidence, autoregressive, random, and secondary-structure-guided orders while holding the model and number of denoiser evaluations fixed. We introduce dual sequence–structure pass@K, which jointly measures native-likeness and foldability. We find that confidence-first decoding favors sequence recovery, whereas a structure-guided order more efficiently reaches sequences that are both native-like and foldable. This gap is concentrated at

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

high-uncertainty *fork* positions, motivating an online entropy-reweighted fine-tuning objective that strengthens structural conditioning where the model remains uncertain.

Our contributions are:

- We present the first controlled study of decoding order in protein inverse folding, with a budget-exact sampler and dual sequence–structure pass@K evaluation.
- We show that structure-guided decoding improves dual pass@K, while confidence-first decoding trades designability for recovery; both effects are associated with high-entropy fork positions, with moderate decoding segments performing best.
- We introduce online entropy-reweighted fine-tuning, which improves recovery and foldability on CATH and transfers to TS50/TS500 and motif scaffolding.

2 PRELIMINARIES

2.1 INVERSE FOLDING

Protein sequences are composed of amino acids. Given a protein backbone X, the goal of *inverse folding* is to design an amino acid sequence x that folds into the prescribed structure. Formally, let $X \in \mathbb{R}^{L\times 4\times 3}$ denote the backbone coordinates of a protein with L residues (N, C_α, C, O atoms per residue), and let $\mathcal{V}$ be the vocabulary of the 20 standard amino acids. We model inverse folding as learning a conditional distribution $p(x \mid X)$ over sequences $x = (x^1, \ldots, x^L) \in \mathcal{V}^L$. The mapping from sequence to structure is many-to-one: structurally similar backbones can be encoded by highly diverse sequences, and the native sequence is only one member of a large set of fold-compatible solutions. Consequently, a designed sequence should be judged not only by its similarity to the native one, but by whether it actually encodes the intended fold. We employ two complementary metrics to assess these aspects. Sequence recovery measures the per-position amino acid identity between the generated sequence and the ground-truth native sequence x_{gt} associated with X:

$$\mathrm{rec}(x, x_{gt}) = \tfrac{1}{L}\textstyle\sum_{i=1}^{L} \mathbb{1}\left[x^i = x^i_{gt}\right]. \tag{1}$$

A sequence is *designable* if it actually folds into X. Following the self-consistency protocol (Dauparas et al., 2022), we fold x with a fixed structure predictor (e.g., ESMFold (Lin et al., 2023)) into $\hat{X}(x)$ and compute the structural agreement:

$$\mathrm{scTM}(x, X) = \mathrm{TM}\Big(\hat{X}(x), X\Big), \tag{2}$$

where TM denotes the TM-score. A high scTM indicates that the sequence encodes the correct structural geometry.

These two metrics can diverge: a sequence may closely resemble the native one yet fail to fold correctly, or fold correctly despite low sequence similarity.

2.2 MASKED DIFFUSION PROTEIN LANGUAGE MODELS

Masked diffusion language models (MDLMs) generate sequences by iteratively unmasking tokens rather than following a fixed autoregressive order. Starting from a fully masked sequence, the model predicts masked positions using bidirectional context; thus, the positions revealed at each step can be chosen in arbitrary orders (Sahoo et al., 2024; Wang et al., 2024b).

Forward process and training objective. Let $m = \texttt{[M]}$ denote the mask token and $\bar{\mathcal{V}} = \mathcal{V} \cup \{m\}$. At time $t \in [0, 1]$, each token is independently retained with probability α_t or replaced by m:

$$q(x^i_t \mid x^i_0) = \mathrm{Cat}\big(x^i_t;\ \alpha_t e_{x^i_0} + (1-\alpha_t)e_m\big). \tag{3}$$

We use $\alpha_t = 1 - t$, under which the denoiser is trained by

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, x_t\sim q(x_t|x_0)}\left[w(t)\sum_{i\in M_t} -\log p_\theta(x^i_0 \mid x_t)\right], \tag{4}$$

where $M_t = \{i : x^i_t = m\}$ and $w(t) = 1/t$. This objective shares the masked cross-entropy structure of RDM (Zheng et al., 2023a) and DPLM (Wang et al., 2024a); its normalization and the draft-input adaptation are distinguished in App. C. Our method reweights its per-position loss terms.

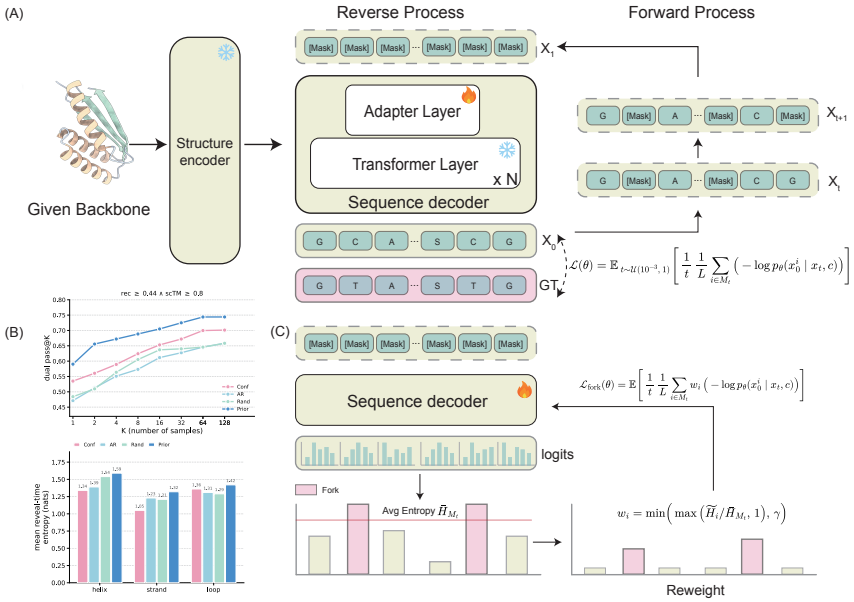


Figure 1: Overview of Hier-IF. (A) Structure-conditioned masked diffusion with a frozen DPLM backbone, trainable adapter, and draft head. (B) Decoding-order study: dual pass@K by order (gaps persist to K=128) and reveal-time entropy at fork positions (confidence defers forks and collapses their entropy). (C) Online entropy-reweighted fine-tuning and its outcome.

Sampling and decoding order. Starting from a masked sequence, an MDLM iteratively reveals tokens predicted by $p_\theta(x_0^i \mid x_t, X)$. At each step, the sampler chooses which masked positions to reveal; this choice is termed the *decoding order*. In our budget-exact ancestral sampler, revealed tokens are fixed thereafter, ensuring a controlled comparison across orders.

3 Our Approach

3.1 Architecture and MDLM Training

Hier-IF uses the DPLM 650M masked diffusion backbone (Wang et al., 2024a). A pretrained structure embedder $c = g_\phi(X)$ provides conditioning through a lightweight adapter. We consider GVP (Jing et al., 2021) and GCP-VQVAE (Pourmirzaei et al., 2025), with GVP as the default. The backbone and structure embedder are frozen; the adapter and a linear draft head d_ψ are trained. The head predicts a structure-derived draft $\hat{x} = \arg\max d_\psi(c)$ and receives an auxiliary cross-entropy loss. The decoder learns to predict native residues from masked drafts using the MDLM-form loss. This input distribution differs from noising native sequences, so we use the loss as a supervised denoising objective rather than claiming an unchanged likelihood bound (App. A). Figure 1(A) summarizes the architecture; App. H distinguishes the objective and encoder comparisons.

3.2 Decoding Orders: A Controlled Study

Budget-exact sampler. Generation proceeds over T steps on the grid $t_s = 1-(s-1)/T$. Per-step reveal counts k_s are assigned by largest-remainder rounding of L/T, so that $\sum_s k_s = L$ exactly and every position is revealed exactly once; decoding starts from the fully masked sequence. *Which* positions are revealed at step s is determined by the decoding order π; assignment of tokens to revealed positions is orthogonal and fixed within each comparison (greedy argmax, or sampling at temperature τ). Algorithm 1 summarizes the procedure.

Algorithm 1 Sampling under decoding order π

Require: denoiser p_θ, conditioning c, order π, steps T
1: $x \leftarrow$ all-mask sequence; $M \leftarrow$ masked positions
2: Split $|M|$ positions into T blocks of sizes $k_1, \ldots, k_T$
3: **for** $s = 1$ **to** T **do**
4: $\quad p \leftarrow p_\theta(\cdot \mid x, c)$
5: $\quad R \leftarrow$ select k_s positions from M according to π
6: $\quad$ Fill positions in R from p (argmax or sampling)
7: $\quad M \leftarrow M \setminus R$
8: **end for**
9: **return** x

Ancestral vs. refinement decoding. The budget-exact sampler uses ancestral-style commitment: once revealed, a position is carried over unchanged. Adaptive selection rules are inference policies, not necessarily exact samples from the standard reverse kernel (App. B). DPLM's default decoder instead performs *iterative refinement* — every step re-predicts *all* positions and re-masks only the lowest-scored ones — so the final sequence is read out from the last, context-richest pass. This dilutes the decoding-order variable unevenly: orders with weak early commitments (e.g. AR) receive far more corrective re-prediction than confidence-ordered decoding, attenuating order effects toward zero. We therefore conduct all order- and fork-level analyses with the ancestral sampler, reserving refinement for absolute-quality numbers; the two decoders differ by only 0.35 pp recovery under confidence, yet disagree on 20–37% of tokens under AR.

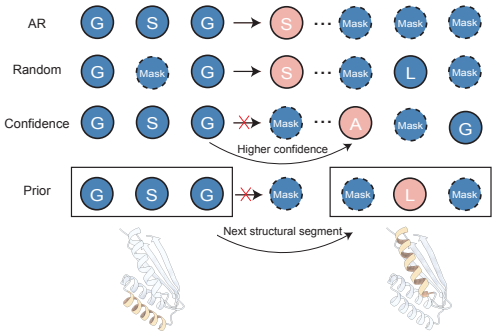


Figure 2: Decoding trajectories under AR, random, confidence-first, and the ss-first structural prior (Sec. 3.2).

Orders. We compare four selection rules, which differ only in π: (i) *random*: a uniform k_s-subset of the remaining masked positions; (ii) *confidence*: the k_s positions with largest $\max_v p_\theta(v \mid \cdot)$; (iii) *AR*: the k_s leftmost masked positions ($T = L$ recovers exact autoregressive decoding; $T < L$ is semi-AR); (iv) *prior*: helix/strand positions (DSSP three-state annotation) first and loop positions last, with a uniform random order *within* each region (*ss-first*) — a biologically motivated order that prioritizes structurally regular regions. Definitions of reversed and shuffled-label controls are given in App. G. Figure 2 illustrates these selection rules: AR advances along the sequence, random decoding samples remaining positions, confidence-first decoding skips uncertain positions, and the ss-first prior prioritizes structurally regular regions.

Dual pass@K. For each test structure X_j we draw K i.i.d. samples $\{x_k^{(j)}\}_{k=1}^K$ by ancestral sampling at temperature τ, fold each with ESMFold, and score each sample by both amino-acid recovery $\mathrm{rec}(x_k^{(j)})$ against the native sequence and the self-consistency TM score $\mathrm{scTM}(x_k^{(j)}, X_j)$. The dual coverage metric is

$$s_{j,k} = \mathbf{1}\big[\mathrm{rec}(x_k^{(j)}) \geq \tau_{\mathrm{rec}}\big]\,\mathbf{1}\big[\mathrm{scTM}(x_k^{(j)}, X_j) \geq \tau_{\mathrm{TM}}\big]. \tag{5}$$

$$\mathrm{pass@}K = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}\left[\max_{k \leq K} s_{j,k} = 1\right]. \tag{6}$$

Thus, Eq. (6) is the fraction of structures with at least one jointly successful sample. We use $\tau_{\mathrm{rec}} = 0.44$ and $\tau_{\mathrm{TM}} = 0.8$; Fig. 4 reports K up to 128. For a fully deterministic decoder, repeated decoding provides no extra coverage. Greedy residue assignment can still be stochastic when the position order is randomized. App. F explains the estimator and evaluation controls.

Protocol. Unless stated otherwise, order experiments use the GVP-conditioned model of Sec. 3.1 and the ancestral sampler above (temperature $\tau = 0.6$). Recovery is reported on the full CATH 4.3 test set; ESMFold-based metrics use a fixed chain subset shared across all runs. Comparisons use the same chains across orders. Evaluation conventions and limits of the reported evidence are given in App. F.

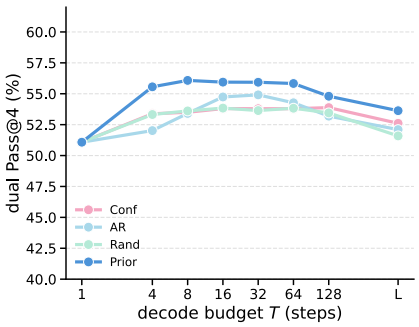


Figure 3: Dual pass@4 versus decode budget T ($\tau = 0.6$). The ss-first prior peaks at a moderate budget; all orders coincide at $T = 1$.

Results. Fig. 4 shows that decoding order controls sequence–structure coverage. Confidence-first decoding achieves the highest greedy recovery and recovery-only pass@K, whereas the ss-first prior achieves the best dual pass@K at every evaluated K under ancestral decoding for both encoders. Its advantage persists through $K = 128$. Thus, structure-guided ordering reaches sequences that are both native-like and foldable more efficiently, despite a small recovery-only trade-off.

Effect of decode budget. Figure 3 varies the number of decoding steps T while holding the sampling budget fixed at $K = 4$. The ss-first prior performs best for $T > 1$, with its highest dual pass@4 at $T = 8$. Both fully parallel decoding ($T = 1$) and strictly sequential decoding ($T = L$) perform worse, supporting the use of moderate decoding segments.

The effect persists across encoders, although refinement changes the ranking (Fig. 4). Order changes the contexts in which uncertainty is resolved without updating model parameters. We next identify high-entropy fork positions and target them during training (Fig. 1(B)).

Robustness across encoders and decoders. The encoder comparison supports the order effect, while the decoder comparison shows that commitment semantics matter. Fig. 4 repeats the dual pass@K study on the 2×2 grid of encoders (GVP, GCP-VQVAE) $\times$ decoding modes (refinement from the structure draft, ancestral from scratch). Under ancestral decoding the ranking in the GVP cell is preserved for both encoders; refinement reduces the spread between orders in the displayed panels while changing their ranking — exactly the signature of the order-diluting re-prediction mechanism discussed above. We therefore take the GVP $\times$ ancestral cell as the canonical setting for all order- and fork-level analyses. An enlarged view appears in Fig. 7.

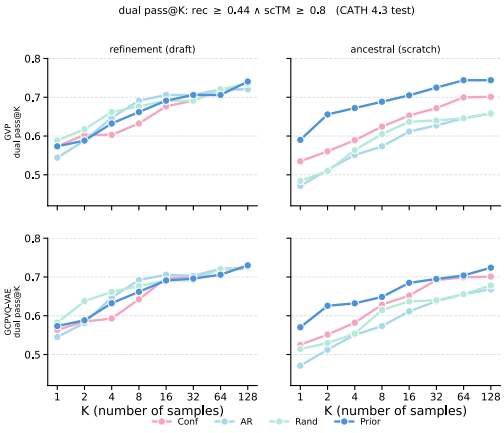


Figure 4: Dual pass@K across encoders (rows: GVP, GCP-VQVAE) and decoders (columns: refinement, ancestral). Refinement narrows order gaps.

Analysis. Order effects on greedy recovery are small because predictive uncertainty is heavy-tailed across positions: under AR decoding, most positions are effectively pinned by context regardless of order, while a small high-entropy tail carries a disproportionate share of uncertainty (Table 1, detailed in Sec. 3.3). This motivates a closer examination of the high-entropy tail.

3.3 Forks: Where Foldability Is Decided

Reveal-time entropy. For each position i, let $r(i)$ denote the step at which it is revealed under a given order, and define its *reveal-time entropy*

$$H_i^{\text{rev}} = H\left[p_\theta\left(x_0^i = \cdot \mid x_{t_{r(i)}}, c\right)\right], \tag{7}$$

the uncertainty of the model's prediction at the moment of commitment. We define the *fork set* $\mathcal{F}$ as the top quartile of positions by H_i^{rev} under the AR order. We use AR as a fixed reference rather than assuming that all orders produce identical entropy rankings. App. E describes how order sensitivity and aggregation should be assessed. Fork positions are the structural analogue of the *forking tokens* that Ni et al. (2026) identify as the branching points of text reasoning; the mechanism by which they fail, however, differs, as we discuss below. Fork statistics below are computed on the CATH 4.3 test set with the budget-exact sampler of Sec. 3.2 (AR decoding at $T{=}L$, greedy assignment); a position *mismatches* if the committed token differs from the native residue.

Results. Table 1 shows that prediction uncertainty and errors concentrate in a small set of high-entropy fork positions: although forks comprise only the top quartile of positions, they account for 42% of reveal-time entropy mass and 40% of sequence mismatches (1.7× and 1.6× enrichment). These shares show that errors concentrate at uncertain positions. App. E gives the enrichment formulas and explains the required aggregation conventions.

Table 1: Fork statistics on the full CATH 4.3 test set (AR, $T{=}L$, greedy). $\mathcal{F}$: top-quartile positions by reveal-time entropy. Enrichment = fork-set share / 25% baseline.

	$\mathcal{F}$	non-fork	Enrich.
positions (%)	25.0	75.0	
entropy mass (%)	42.2	57.8	1.7×
mismatches (%)	40.1	59.9	1.6×

Analysis. Forks are positions at which structural context under-determines the residue, or where the model has not learned to exploit that context effectively. The error enrichment motivates targeting these contexts, although it does not by itself establish a causal effect on foldability. Decoding order can change prediction quality through context; training additionally changes the conditional model itself (Sec. 3.4).

3.4 Fork-Weighted Training

The order study changes inference while keeping parameters fixed. The fork analysis identifies contexts where the model is less accurate. We therefore add a training intervention that concentrates supervision at uncertain positions (Fig. 1(C)).

Online entropy reweighting. We reweight the per-position cross-entropy of Eq. 4 by a function of the model's own predictive uncertainty, computed online from the same forward pass that produces the loss — no extra inference is required. For a masked position i under the sampled corrupted input, let

$$\widetilde{H}_i \;=\; H\big[p_\theta(x_0^i = \cdot \mid x_t, c)\big] \tag{8}$$

denote the *online entropy* of the current predictive distribution, and let $\bar{H}_{M_t}$ denote its mean over the masked positions $j \in M_t$ of the *same sequence*. The fork-weighted objective is

$$\mathcal{L}_{\text{fork}}(\theta) = \mathbb{E}_{x_0,\,t,\,x_t}\left[\, w(t) \sum_{i \in M_t} w_i \left(-\log p_\theta(x_0^i \mid x_t, c)\right)\right], \tag{9}$$

with detached per-position weight (App. C.3)

$$w_i \;=\; \min\big(\max(\widetilde{H}_i/\bar{H}_{M_t},\, 1),\, \gamma\big), \qquad \gamma = 3. \tag{10}$$

The normalized entropy ratio $\widetilde{H}_i/\bar{H}_{M_t}$ provides a soft, per-sequence measure of fork-ness. The floor prevents downweighting easy positions, while the cap γ limits the influence of any single position. As $\widetilde{H}_i$ is recomputed online, positions whose entropy falls below the masked-position mean receive unit weight. Thus, fork-weighted fine-tuning continuously concentrates supervision on positions that remain uncertain.

Training. We finetune from the checkpoint of Sec. 3.1 with Eq. 9; all other settings are identical to the base run of Sec. 3.1. Algorithm 2 specifies the updates; App. G distinguishes soft weighting from hard selection.

Algorithm 2 Fork-weighted fine-tuning

Require: pretrained model with frozen backbone θ_B, frozen encoder g_ϕ, adapter θ_A, and draft head d_ψ

1: **for** each training step **do**
2: Sample a structure–sequence pair (X, x_0)
3: $c \leftarrow g_\phi(X)$; $\hat{x} \leftarrow \arg\max d_\psi(c)$
4: Sample $t \sim \mathcal{U}(10^{-3}, 1)$ and mask $\hat{x}$ with probability t to obtain x_t
5: $p \leftarrow p_{\theta_A,\theta_B}(\cdot \mid x_t, c)$
6: Compute detached entropy-based weights on masked positions
7: Add the auxiliary draft-head loss to the fork-weighted loss
8: Update θ_A, ψ; keep θ_B, ϕ frozen
9: **end for**

Table 2: Comparison on CATH 4.2 and 4.3. † denotes ESM utilization. Best results are in **bold**; second-best results are underlined. Short, Single, and All refer to short-chain, single-chain, and full test subsets, respectively.

Model	CATH 4.2						CATH 4.3					
	Recovery ↑			Perplexity ↓			Recovery ↑			Perplexity ↓		
	Short	Single	All	Short	Single	All	Short	Single	All	Short	Single	All
GraphTrans	0.28	0.28	0.36	8.39	8.83	6.63	0.30	0.34	0.34	8.37	8.83	6.61
GVP	0.31	0.29	0.39	7.23	7.84	5.36	0.33	0.36	0.38	7.21	7.85	5.34
PiFold	0.40	0.39	0.52	6.04	6.31	4.55	0.43	**0.52**	0.51	6.04	6.27	4.52
AlphaDesign	0.34	0.33	0.41	7.32	7.63	6.30	0.43	0.43	0.42	7.32	7.60	6.28
ProteinMPNN	0.36	0.34	0.46	6.21	6.68	4.61	0.38	0.44	0.44	6.21	6.61	4.61
ESM-IF†	0.31	0.39	0.38	8.18	6.33	6.44	0.34	0.39	0.38	8.18	6.32	6.42
LM-Design†	0.38	0.42	0.56	6.77	6.46	4.52	0.47	0.48	0.56	5.66	5.52	4.01
GraDe-IF	0.45	0.43	0.52	5.49	6.21	4.35	0.46	0.44	0.54	5.50	6.13	4.29
Bridge-IF	0.44	0.45	0.58	5.68	5.27	**3.38**	0.47	0.48	0.59	5.49	5.63	3.90
Hier-IF	0.43	0.44	0.56	6.01	5.83	4.44	0.47	0.48	0.54	5.69	5.54	4.09
Hier-IF (Fork-weighted)	**0.46**	**0.46**	**0.59**	**5.46**	**5.17**	4.01	**0.48**	0.49	**0.60**	**5.37**	**5.22**	**3.88**

4 EXPERIMENTS

We evaluate sequence recovery and perplexity on CATH and independent TS50/TS500 test sets, followed by motif-scaffolding, foldability, and component-ablation experiments. Evaluation conventions are collected in App. F.

4.1 BENCHMARKING INVERSE FOLDING

We evaluate Hier-IF on the CATH 4.2 and 4.3 benchmarks using their standard data splits (Ingraham et al., 2019; Hsu et al., 2022). Following prior work, we report sequence recovery and perplexity on short, single, and all chains.

Table 2 compares Hier-IF with representative graph-based, pretrained-LM-based, and diffusion-based inverse-folding methods. Hier-IF with fork-weighted fine-tuning achieves the best recovery among the listed methods on five of the six reported splits; PiFold leads on CATH 4.3 Single (0.52 versus 0.49). It also obtains the best perplexity on CATH 4.3, demonstrating that entropy reweighting improves both sequence recovery and conditional likelihood.

4.2 OUT-OF-DISTRIBUTION GENERALIZATION: TS50 AND TS500

We evaluate generalization beyond the CATH test sets on TS50 and TS500, which contain 50 and 500 protein structures, respectively. We compare fork-weighted Hier-IF with ProteinMPNN, PiFold, and LM-Design using sequence recovery and perplexity. These benchmarks provide complementary evaluations at two test-set sizes; Table 3 summarizes the results.

Table 3: Generalization performance on independent test sets. We compare Hier-IF (fork-weighted) with SOTA on TS50 and TS500. The best results are highlighted in **bold**.

	TS50		TS500	
Model	Recovery ↑	Perplexity ↓	Recovery ↑	Perplexity ↓
ProteinMPNN	54.43	3.93	58.08	3.53
PiFold	58.72	3.86	60.42	3.44
LM-Design	57.89	3.50	64.30	3.19
Hier-IF (fork-weighted)	**58.81**	**3.47**	**66.42**	**2.83**

On TS50, Hier-IF achieves 58.81% recovery and a perplexity of 3.47, the best point estimates among the compared methods. The recovery gain over PiFold is modest, at 0.09 percentage points, while perplexity is 0.03 lower than that of LM-Design, the strongest baseline on this metric. The advantage is more pronounced on TS500: Hier-IF achieves 66.42% recovery and a perplexity of 2.83, compared with 64.30% and 3.19 for LM-Design. This corresponds to a recovery gain of 2.12 percentage points and a relative perplexity reduction of approximately 11.3%.

The consistent direction of the results across both benchmarks supports the generalization of the full Hier-IF configuration beyond the CATH evaluation. The larger margins on TS500 provide stronger numerical separation from the baselines, whereas the near-tie in TS50 recovery calls for a more cautious interpretation. These comparisons evaluate the complete model; the contribution of each component is examined separately in the CATH ablation study.

4.3 Benchmarking Motif Scaffolding

Setup. We evaluate Hier-IF on motif scaffolding following the sequence-based evaluation protocol of DPLM-2 (Wang et al., 2024b), which comprises 24 motif-scaffolding problems. Built upon DPLM, Hier-IF generates scaffold sequences conditioned on the amino acid sequence of a given functional motif, while keeping the motif residues unchanged. The objective is to generate a sequence whose folded structure provides a plausible scaffold and preserves the target motif conformation. For each problem, we sample 100 designs with scaffold lengths drawn from the benchmark-specified ranges. For motifs consisting of multiple segments, we retain the segment order specified by the benchmark. The generated sequences are folded using ESMFold (Lin et al., 2023) to obtain predicted structures for evaluation.

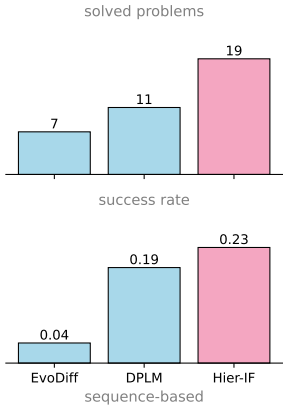


Figure 5: Motif-scaffolding performance in the sequence-based setting: number of solved problems and average success rate across 24 benchmark problems.

Metrics. Following the sequence-based setting in DPLM-2 (Wang et al., 2024b), a design is considered successful only if it satisfies both criteria: (1) an overall pLDDT greater than 70, assessing confidence in the predicted structure; and (2) a motif RMSD below 1 Å relative to the reference motif after structural alignment, assessing preservation of the target motif geometry. We report two metrics: the number of *solved problems*, where a problem is considered solved if at least one of its 100 designs succeeds; and the average *success rate*, computed as the mean fraction of successful designs across the 24 problems.

Baselines. We compare Hier-IF with EvoDiff (Alamdari et al., 2023) and DPLM (Wang et al., 2024a), corresponding to the sequence-based baselines in the motif-scaffolding evaluation of DPLM-2 (Wang et al., 2024b). Both methods generate scaffold sequences conditioned on the motif sequence. EvoDiff performs diffusion in amino acid sequence space, while DPLM is a discrete diffusion protein language model and serves as the base model for Hier-IF.

Results. As shown in Fig. 5, Hier-IF solves 19 of the 24 problems with an average success rate of 23%, outperforming EvoDiff and DPLM on both metrics. EvoDiff solves 7 problems with a

success rate of 4%, while DPLM solves 11 problems with a success rate of 19%. Compared with DPLM, Hier-IF solves eight more problems and improves the average success rate by four percentage points. The increase in solved problems indicates broader coverage of the benchmark, while the higher success rate reflects an improved overall yield of successful designs. These results support the effectiveness of Hier-IF in generating scaffold sequences that accommodate the target motif conformation.

4.4 Evaluation of Foldability

High sequence recovery does not always guarantee that the generated sequences will fold into the desired structures. To verify the physical plausibility and *foldability* of our designs, we performed an *in silico* folding validation using OpenFold. We predicted the 3D structures of the generated sequences and calculated the TM-score and scRMSD against the ground truth backbones. The results in Table 4 show that Hier-IF achieves the highest TM-score (0.85) and ties for the best scRMSD (1.36 Å). This confirms that Hier-IF generates sequences that are not only statistically likely but also structurally consistent with the target geometries.

Table 4: Co-folding validation. Recovery (Rec.), TM-score, and self-consistency RMSD (scRMSD) against target backbones. Bold indicates the best result.

Model	Rec.	TM	scRMSD
ProteinMPNN	0.46	0.80	**1.36**
PiFold	0.52	0.71	1.67
GraDe-IF	0.52	0.76	1.41
Bridge-IF	0.58	0.81	1.47
Hier-IF (Fork-weighted)	**0.60**	**0.85**	**1.36**

4.5 Ablation Study

We ablate the two main components of Hier-IF on the CATH 4.3 test set: prior-based inference and entropy-based fork weighting during fine-tuning. As shown in Fig. 6, prior-based inference improves recovery from 0.54 to 0.57 and reduces perplexity from 4.09 to 4.03. Fork-weighted fine-tuning gives recovery 0.57 and perplexity 3.99. Combining the two components achieves the best overall performance, improving recovery to 0.60 and reducing perplexity to 3.88. These results indicate that prior-based inference and fork-weighted training provide complementary gains.

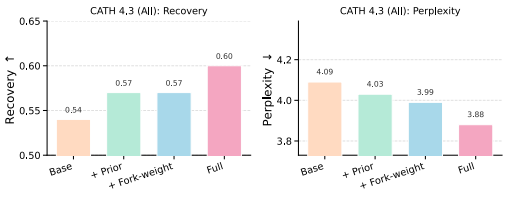


Figure 6: **Component ablation on CATH 4.3.** Prior-based inference and fork-weighted fine-tuning each improve recovery and reduce perplexity; combining them performs best.

Scope of robustness. Figure 4 compares encoders and decoders. It does not compare RDM and MDLM training objectives, so these axes should not be conflated. App. H specifies what an objective comparison must hold fixed and the limits of the evidence shown here.

5 Conclusion and Discussion

We presented Hier-IF, a hierarchical inverse-folding framework that combines structure-conditioned masked diffusion with controllable decoding orders. Using sequence–structure dual pass@K, we find that decoding orders differ not only in sequence recovery but, more importantly, in their ability to produce sequences that are both native-like and structurally designable. This analysis identifies high-uncertainty positions as a key bottleneck for inverse folding. Motivated by this finding, we introduce entropy-based reweighting to focus training on these difficult positions. Hier-IF achieves strong results among the compared inverse-folding methods, demonstrating that decoding-time control and training-time reweighting are complementary. In future work, we will scale the method to larger protein backbones and investigate its use in more challenging constrained design tasks.

AI USE STATEMENT

We used generative AI tools to assist with manuscript editing and LaTeX formatting, literature search, cross-reference and consistency checks, and the examination and revision of mathematical arguments and derivations. The authors take full responsibility for the accuracy and integrity of the final manuscript and associated artifacts, including all AI-assisted content.

ETHICS STATEMENT

Protein sequence design has beneficial research applications and potential for misuse. The evaluations reported here are computational; predicted foldability does not establish biological function, safety, or experimental success. Any downstream synthesis or deployment requires an appropriate assessment of the designed sequences and their intended use.

REPRODUCIBILITY STATEMENT

Algorithms 1 and 2 specify the decoding and training procedures. Appendices A–C clarify the input distributions, objective assumptions, and gradient computation. Appendices F–H describe evaluation conventions and distinguish reported comparisons from additional controls. Exact run configurations, chain lists, and random seeds are needed alongside these descriptions for full reproduction.

REFERENCES

Sarah Alamdari, Nitya Thakkar, Rianne Van Den Berg, Neil Tenenholtz, Robert Strome, Alan M Moses, Alex X Lu, Nicolò Fusi, Ava P Amini, and Kevin K Yang. Protein generation with evolutionary diffusion: sequence is all you need. *BioRxiv*, pp. 2023–09, 2023.

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.

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.

Frédéric A Dreyer, Daniel Cutting, Constantin Schneider, Henry Kenlay, and Charlotte M Deane. Inverse folding for antibody sequence design using deep learning. *arXiv preprint arXiv:2310.19513*, 2023.

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

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

Alan R Fersht, Andreas Matouschek, and Luis Serrano. The folding of an enzyme: I. theory of protein engineering analysis of stability and pathway of protein folding. *Journal of molecular biology*, 224(3):771–782, 1992.

Zhangyang Gao, Cheng Tan, Pablo Chacón, and Stan Z Li. Pifold: Toward effective and efficient protein inverse folding. *arXiv preprint arXiv:2209.12643*, 2022.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q Tran, Jonathan Deaton, Marius Wiggert, et al. Simulating 500 million years of evolution with a language model. *Science*, pp. eads0018, 2025.

Cheng-Yen Hsieh, Xinyou Wang, Daiheng Zhang, Dongyu Xue, Fei Ye, Shujian Huang, Zaixiang Zheng, and Quanquan Gu. Elucidating the design space of multimodal protein language models, 2025. URL https://arxiv.org/abs/2504.11454.

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*, pp. 8946–8970. PMLR, 2022.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons, 2021. URL https://arxiv.org/abs/2009.01411.

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

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. 2023.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41 (8):1099–1106, 2023.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models, 2026. URL https://arxiv.org/abs/2601.15165.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

Mahdi Pourmirzaei, Alex Morehead, Farzaneh Esmaili, Jarett Ren, Mohammadreza Pourmirzaei, and Dong Xu. Gcp-vqvae: A geometry-complete language for protein 3d structure. *bioRxiv*, pp. 2025–10, 2025.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Peter Chen, John Canny, Pieter Abbeel, and Yun Song. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pp. 8844–8856. PMLR, 2021.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models, 2024. URL https://arxiv.org/abs/2406.07524.

Arthur G Street and Stephen L Mayo. Computational protein design. *Structure*, 7(5):R105–R109, 1999.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024a.

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

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*, 2023a.

Zaixiang Zheng, Yifan Deng, Dongyu Xue, Yi Zhou, Fei Ye, and Quanquan Gu. Structure-informed language models are protein designers. In *International conference on machine learning*, pp. 42317–42338. PMLR, 2023b.

A TRAINING AND INFERENCE DISTRIBUTIONS

A.1 NATIVE CORRUPTION AND DRAFT CORRUPTION

Equation (3) defines the absorbing corruption of a native sequence. Hier-IF instead constructs a draft from structure features, $c = g_\phi(X)$ and $\hat{x} = \arg\max d_\psi(c)$, and masks the draft before predicting native residues. The structure embedder is frozen, but the draft head is trainable; the draft can therefore change as its head is updated. Let m be the mask token and M_t the set of masked positions. The input is

$$x_t^i = \begin{cases} m, & i \in M_t, \\ \hat{x}^i, & i \notin M_t, \end{cases} \qquad \Pr(i \in M_t \mid t) = t. \tag{11}$$

The MDLM-form denoising loss used with this input is

$$\mathcal{L}_{\text{draft}} = \mathbb{E}_{X, x_0, t, M_t} \left[\frac{1}{t} \sum_{i \in M_t} -\log p_\theta(x_0^i \mid x_t, c) \right]. \tag{12}$$

The auxiliary draft-head cross-entropy supervises the prediction $d_\psi(c)$ against x_0. It is separate from the masked decoder loss. In particular, the indicator selecting decoder-loss positions is $\mathbf{1}[x_t^i = m]$, not $\mathbf{1}[x_t^i \neq x_0^i]$: an unmasked draft residue can differ from the native residue without belonging to M_t.

A.2 WHAT MASK SUPPORT DOES AND DOES NOT ESTABLISH

For any $t \in (0, 1)$, a particular mask set M has probability $t^{|M|}(1-t)^{L-|M|} > 0$. Thus, independent masking supports every mask *pattern*. This does not imply support for every token-valued context: unmasked training positions contain draft tokens, whereas ancestral decoding contains model-generated tokens. Nor does positive probability imply that all mask patterns occur frequently in a finite training run.

Refinement is initialized from a structure draft, while the controlled ancestral experiment begins fully masked. Subsequent states differ from training inputs in both cases. Matching the initialization source therefore does not establish equality of training and inference distributions. Equation (12) is a supervised denoising adaptation; the usual native-data diffusion likelihood bound does not automatically transfer to this altered corruption process. The draft-versus-scratch effect requires a direct comparison and cannot be inferred from support alone.

B DECODING SEMANTICS AND COMPUTE ACCOUNTING

Budget-exact commitment. Algorithm 1 assigns L reveals to T evaluations with $k_s \in \{\lfloor L/T \rfloor, \lceil L/T \rceil\}$ and $\sum_s k_s = L$. All positions selected at a step use the same pre-update denoiser output. For $T = L$, left-to-right selection is sequential; for $T < L$, it commits blocks in parallel. Any implementation allowing $T > L$ must specify whether zero-reveal steps are evaluated or skipped, since this affects the actual number of function evaluations (NFE).

Relation to diffusion sampling. Carry-over of revealed tokens is shared with absorbing diffusion sampling. However, a fixed reveal budget combined with confidence-based or secondary-structure-based selection is an inference policy. Arbitrary adaptive policies are not automatically exact ancestral samplers of the standard reverse kernel. We use "ancestral" to distinguish irreversible commitment from refinement, with this qualification.

Refinement. The refinement decoder re-predicts positions and re-masks low-confidence predictions until the final step. Early selections may consequently be corrected later. Matching T can match NFE for the same model, but it does not make commitment semantics identical. Conversely, re-prediction alone does not imply that different orders use different NFE. Runtime also includes selection, sorting, and structural preprocessing, which should be reported separately when comparing computational efficiency.

Controlled comparisons. An order comparison holds the checkpoint, conditioning features, chain set, residue-sampling temperature, reveal counts, and stopping rule fixed. Figure 4 compares both encoder and decoder settings; the encoder comparison within a fixed decoding mode isolates a different factor from the refinement-versus-ancestral comparison. Greedy residue assignment is deterministic only when the position-selection policy and tie handling are also deterministic.

C Masked objectives and entropy-weighted optimization

C.1 Schedule weighting for native masked diffusion

For the native absorbing process, let $s < t$ and let α_t denote token survival. Conditional on being masked at time t, the probability of revealing the native token at time s is

$$\beta_{s,t} = \frac{\alpha_s - \alpha_t}{1 - \alpha_t}. \tag{13}$$

A reverse model sharing this mask probability and assigning revealed-token probabilities through p_θ yields the per-position KL contribution

$$\mathrm{KL}_i = \mathbf{1}[x_t^i = m]\, \beta_{s,t}[-\log p_\theta(x_0^i \mid x_t, c)]. \tag{14}$$

Already unmasked positions are carried over. In the continuous-time limit, $\beta_{t-\Delta t,t}/\Delta t$ tends to

$$w(t) = -\frac{\alpha_t'}{1 - \alpha_t} = \frac{1}{t} \quad \text{for } \alpha_t = 1 - t. \tag{15}$$

This is the masked cross-entropy form underlying Eq. (4) (Sahoo et al., 2024). The assumptions above concern native-data corruption and a specified reverse parameterization. RDM (Zheng et al., 2023a) and MDLM belong to a related loss family; that relationship does not establish equality of arbitrary implementations, normalizations, or the draft-conditioned objective.

C.2 Realized versus expected mask-count normalization

Let $n_t = |M_t|$ and $S_t = \sum_{i \in M_t} -\log p_\theta(x_0^i \mid x_t, c)$. Two possible per-sequence normalizations, for $n_t > 0$, are

$$\ell_{\mathrm{real}} = \frac{S_t}{n_t}, \qquad \ell_{\mathrm{expected}} = \frac{S_t}{tL} = \frac{n_t}{tL}\ell_{\mathrm{real}}. \tag{16}$$

Although $\mathbb{E}[n_t \mid t] = tL$, it does not follow that the two losses have the same expectation: S_t and n_t are coupled through the masked context. Equating an expected reciprocal with the reciprocal of an expectation is also invalid. An empirical objective comparison must specify the loss reduction, timestep sampling, treatment of zero-mask examples, and any sequence-length normalization. A common target alone is not a proof of identical gradients or robustness to changing the objective.

C.3 Detached weights and the gradient

The weight in Eq. (10) is evaluated from the current model and treated as constant during backpropagation. For a sampled context, the resulting update has the form

$$g(\theta) = w(t) \sum_{i \in M_t} \mathrm{stopgrad}(w_i)\, \nabla_\theta[-\log p_\theta(x_0^i \mid x_t, c)]. \tag{17}$$

There is no gradient through the entropy computation. This prevents the model from changing the weight directly to reduce the loss, but it does not preserve the original diffusion ELBO: data-dependent weights change the optimization criterion. The weights retain native residues as targets while redistributing gradient contributions.

Online entropy is calculated for each masked position over the 20-residue vocabulary. Its mean is computed over masked positions of the same sequence. The floor keeps $w_i \geq 1$ and the cap gives $w_i \leq \gamma = 3$; positions at or below the mean entropy receive unit weight. Confidence alone does not guarantee unit weight, because the comparison is relative. A numerically complete implementation must also define the zero-mean case and the zero-mask case; setting unit weights when all entropies vanish and zero decoder loss for an empty mask set are well-defined conventions.

C.4 Training components

Algorithm 2 freezes both θ_B and ϕ and updates the adapter and draft head. Because argmax draft construction is discrete, the auxiliary cross-entropy is the direct training signal for the draft head; it should not be described as receiving gradients through argmax from the decoder. Training samples $t \sim \mathcal{U}(10^{-3}, 1)$. The auxiliary head loss and the decoder loss supervise different outputs; their relative coefficient is part of the optimization configuration.

D Related Work

D.1 Protein Language Models

With the advancement of large-scale protein sequence datasets, numerous protein language models —drawing inspiration from natural language processing (NLP)— have demonstrated remarkable potential. Models such as ProtTrans (Elnaggar et al., 2021) and TAPE (Rao et al., 2019) capture the information embedded in protein sequences by framing protein-related tasks as classic language modeling problems, such as masked language modeling and text-to-text generation. In parallel, the evolutionary information contained within protein sequences provides powerful inductive biases that facilitate learning biologically meaningful representations. For example, models like MSA-Transformer (Rao et al., 2021) leverage multiple sequence alignments (MSAs), leading to more effective learning. The ESM-series models (Rives et al., 2021; Lin et al., 2023; Hayes et al., 2025) utilize large-scale language modeling over hundreds of millions of protein sequences, implicitly capturing evolutionary information and enabling the extraction of robust sequence embeddings that correlate with protein structures and functions. Furthermore, with the development of generative models, methods such as ProGen (Madani et al., 2023) and ProtGPT2 (Ferruz et al., 2022) have demonstrated the capability to learn sequence distributions useful for generating desired protein sequences, making them applicable to downstream design tasks.

D.2 Inverse Folding

Based on a given protein structure (backbone), designing the sequence is called inverse folding. Using a structural encoder and a sequence decoder is a straightforward yet effective strategy for structure-based protein design. This framework allows models to extract structural context and generate compatible amino-acid sequences. ProteinMPNN (Dauparas et al., 2022) employs a message-passing neural network directly on residue coordinates to capture local geometric features, followed by an autoregressive decoder. In contrast, PiFold (Gao et al., 2022) constructs a residue-level graph and uses a graph neural network to model both spatial and topological relationships, with a transformer-based decoder that better captures long-range dependencies. These differences reflect distinct design choices in how structural information is encoded and utilized for sequence generation.

On the other hand, LM-Design (Zheng et al., 2023b) and the DPLM series (Wang et al., 2024a;b) have effectively leveraged pretrained language models by introducing lightweight adapters. These methods insert small trainable modules into frozen pretrained models, enabling efficient fine-tuning while preserving the general knowledge encoded in the pretrained weights.

D.3 Diffusion Language Models and Decoding Order

Discrete diffusion has become a competitive paradigm for text generation, with masked (absorbing-state) variants such as MDLM (Sahoo et al., 2024), SEDD (Lou et al., 2023), and the LLaDA and Dream families (Nie et al., 2025; Ye et al., 2025) built on the D3PM framework (Austin et al., 2021). A selling point of these models is order-free iterative decoding, and recent work has begun to question how much that freedom buys: the Flexibility Trap (Ni et al., 2026) shows that adaptive, confidence-driven orders bypass high-uncertainty "forking tokens" and shrink the reachable solution space relative to autoregressive decoding, and JustGRPO-Fast exploits high-entropy positions to accelerate RL updates. In proteins, masked diffusion underlies DPLM (Wang et al., 2024b;a) and related generators (Alamdari et al., 2023), but the decoding order itself has not been systematically studied. Our work provides a first systematic analysis and identifies high-entropy fork positions as the key commitment bottleneck that governs foldability.

E FORK STATISTICS AND INTERPRETATION

E.1 ENRICHMENT AND AGGREGATION

For a chain with L valid positions and fork set $\mathcal{F}$, let $e_i = \mathbf{1}[x^i \neq x_0^i]$ and $q = |\mathcal{F}|/L$. The entropy share, mismatch share, and corresponding enrichments are

$$a_H = \frac{\sum_{i\in\mathcal{F}} H_i^{\mathrm{rev}}}{\sum_{i=1}^{L} H_i^{\mathrm{rev}}}, \qquad E_H = a_H/q, \tag{18}$$

$$a_e = \frac{\sum_{i\in\mathcal{F}} e_i}{\sum_{i=1}^{L} e_i}, \qquad E_e = a_e/q. \tag{19}$$

A zero denominator makes the corresponding share undefined, so an aggregation must specify how such chains are treated. With the 25% baseline in Table 1, the displayed shares give $42.2/25 = 1.688$ and $40.1/25 = 1.604$, which round to the reported $1.7\times$ and $1.6\times$ enrichments. These calculations restate the main table rather than introducing a new experiment.

Per-chain averages and pooled-residue statistics are different quantities. If r_F and r_N are pooled accuracies on forks and non-forks, then

$$a_e = \frac{q(1-r_F)}{q(1-r_F) + (1-q)(1-r_N)}. \tag{20}$$

Averaging ratios over chains does not generally preserve this identity when separately averaged accuracies are substituted. Fork definitions, rounding of the top quartile, and aggregation should therefore be reported consistently. Error enrichment establishes an association with uncertainty; it does not alone prove that changing those residues causes a loss of foldability.

E.2 AGREEMENT OF ONLINE AND REVEAL-TIME ENTROPY

Reveal-time entropy is evaluated under a decoding trajectory, whereas online entropy uses sampled masked-draft inputs. To compare them, one can average online entropies over repeated mask draws at each position, then compute within-chain Spearman correlation and top-quartile overlap against the AR reference. Positions must be matched and results aggregated over chains, with the number of mask draws and chains stated. Agreement is an empirical question; mask-pattern support is not a substitute for measuring it. Online entropy is used as a practical uncertainty proxy; equivalence to reveal-time entropy is not assumed.

Order sensitivity and static features. The same matched-chain procedure can compare reveal-time rankings across orders. Testing structural predictability additionally requires specified features, held-out splits, and a predictive metric. Neither identical fork membership across orders nor impossibility of structural prediction follows from error enrichment alone.

F EVALUATION CONVENTIONS AND EVIDENCE COVERAGE

F.1 DATASETS AND MEASURES

CATH recovery is evaluated on the standard benchmark splits; the reported CATH 4.3 test size is $N = 1864$. Short, Single, and All denote the short-chain, single-chain, and complete test subsets. Recovery is a fraction in Tables 2 and 4, and a percentage in Table 3. These units should be converted before cross-table comparisons.

Order-study foldability uses ESMFold on a shared length-stratified subset, of approximately 200 chains. This subset is distinct from the full CATH test set used for recovery. The separate co-folding comparison in Table 4 reports OpenFold results; these should not be pooled with ESMFold-based scores. The predictor checkpoint, input features, alignment procedure, and handling of prediction failures must be fixed within each comparison.

F.2 DUAL PASS@K AND SAMPLING BUDGETS

Equations (1), (2), and (5) define a jointly successful candidate. For K candidates, Eq. (6) averages the indicator of at least one success over test structures. The displayed order study uses $\tau = 0.6$, $\tau_{\mathrm{rec}} = 0.44$, and $\tau_{\mathrm{TM}} = 0.8$. These thresholds define an evaluation criterion, not a guarantee of physical or functional validity.

If a structure has n sampled candidates, of which c satisfy both criteria, the fraction of uniformly drawn size-K subsets containing a success is

$$\widehat{P}_K = 1 - \frac{\binom{n-c}{K}}{\binom{n}{K}}, \qquad 1 \le K \le n, \tag{21}$$

where the numerator is zero when $n - c < K$. This finite-pool combinatorial identity explains how one candidate pool can support multiple budgets; it is distinct from generating fresh independent pools for each K. The implementation must state which estimator is used. Increasing K raises sample-level compute even when per-sample NFE is fixed at T.

F.3 PAIRED COMPARISONS

Using the same chains permits paired comparisons of order-dependent metrics. A Wilcoxon signed-rank test and bootstrap intervals can summarize paired differences, but the chain count, resampling unit, number of repeats, and multiple-comparison policy must be specified. The displayed figures show point estimates. Their ordering alone is not evidence of statistical significance, and uncertainty should be assessed at the chain level rather than by treating residues as independent replicates.

F.4 MOTIF-SCAFFOLDING PROTOCOL

We use the evaluation setting described in Sec. 4.3: 24 motif-scaffolding problems and 100 designs per problem, with scaffold lengths sampled according to the benchmark specification and motif residues kept unchanged. For multi-segment motifs, the benchmark-specified segment order is retained. Generated sequences are folded with ESMFold and evaluated under the sequence-based protocol of DPLM-2 (Wang et al., 2024b). Success requires both an overall pLDDT greater than 70 and a motif RMSD below 1 Å relative to the reference motif after structural alignment.

We count a problem as solved if at least one of its 100 designs succeeds, and average the per-problem success fractions over all 24 problems. Figure 5 reports 19 solved problems and an average success rate of 23% for Hier-IF. Exact motif identifiers, scaffold-length ranges, model checkpoints, and baseline configurations are needed to reproduce the evaluation.

G STRUCTURAL PRIORS AND TRAINING CONTROLS

Secondary-structure annotation. The ss-first rule groups helix and strand residues before loop residues, using a three-state secondary-structure annotation, and randomizes positions within each group. The term "loop" denotes the annotation category; it does not imply that those residues lack a defined spatial structure. A reproducible implementation should record the DSSP version, the mapping from its raw codes to three states, missing-residue handling, and seeds.

Order controls. A loop-first control reverses the group priority. A shuffled-label control permutes group labels within a chain while preserving their counts. Both controls must use the same reveal budget and token-sampling rule as ss-first. These controls isolate the priority assigned to structural labels from the label composition and decoding budget. They are defined here to clarify the distinction between a structural prior and an arbitrary position order.

Soft weighting versus hard selection. Hier-IF retains supervision at all masked positions and gives larger weights to positions above the mean entropy. This differs from keeping only the top quartile, which discards the remaining decoder-loss terms. A fixed position table also differs from online weights because it cannot adapt as the model changes. These distinctions explain the design of Eq. (10); they do not establish superiority over hard selection or static weights without a matched experiment.

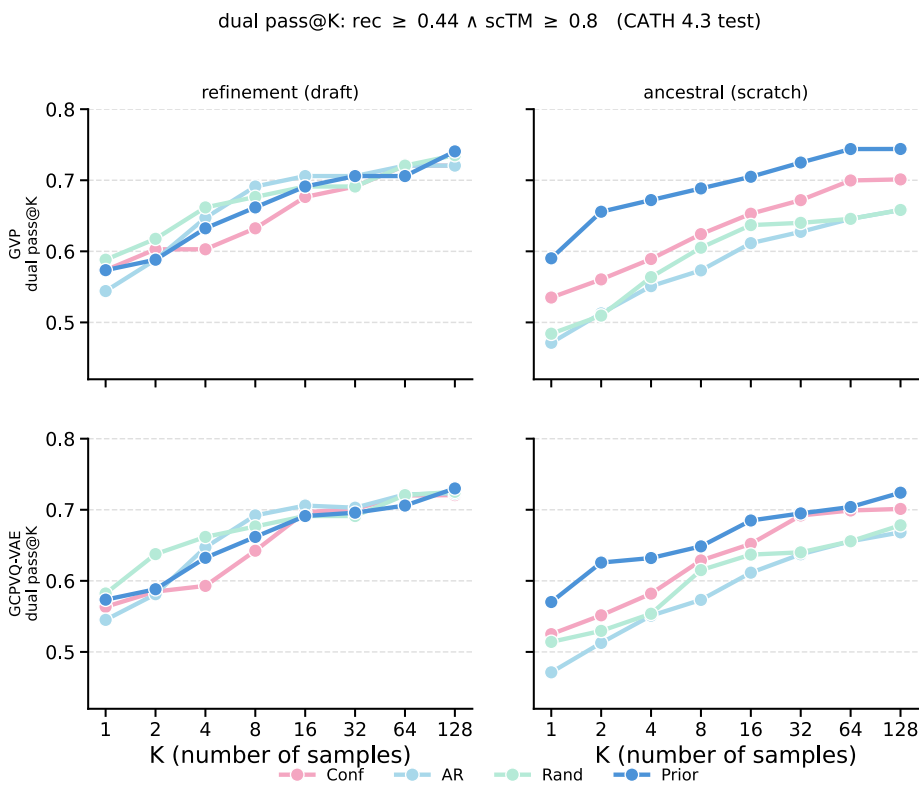


Figure 7: Enlarged view of Fig. 4, using the same data and image. Rows compare GVP and GCP-VQVAE; columns compare refinement and ancestral decoding.

H OBJECTIVE AND ENCODER COMPARISONS

Figure 4 reports a two-by-two comparison of GVP/GCP-VQVAE encoders and refinement/ancestral decoders. The ss-first curve leads the ancestral panels at the displayed budgets. Refinement changes the spread and sometimes the ranking of orders. This is evidence about encoders and decoding semantics, not an RDM-versus-MDLM training comparison.

An objective comparison would hold the initialized backbone, frozen encoder, trainable adapter and head, draft-input construction, data, and optimization settings fixed while changing only the documented loss weighting or normalization. App. C explains why the two forms cannot be declared identical merely from $\mathbb{E}|M_t| = tL$. Objective invariance is a separate empirical hypothesis and does not follow from the encoder/decoder comparison in Fig. 4.

Component ablation. Figure 6 provides the actual component comparison on CATH 4.3: the base model reports recovery/perplexity $0.54/4.09$; prior-based inference reports $0.57/4.03$; fork weighting reports $0.57/3.99$; the full configuration reports $0.60/3.88$. This component comparison addresses inference and training interventions; it is distinct from comparing diffusion objectives or alternative weighting functions.