A IMPLEMENTATION DETAILS

A.1 BASELINE EVALUATION

In our experiments, we compare Hier-IF with several approaches, including GraphTransIngraham et al. (2019), GVPJing et al. (2020), PiFoldGao et al. (2022a), AlphaDesignGao et al. (2022b), ProteinMPNNDauparas et al. (2022), ESM-IFHsu et al. (2022), LM-DesignZheng et al. (2023) and KW-DesignGao et al. (2023). We will describe the implementation as follow:

- **GraphTrans**: They introduce conditional language models for protein sequence that directly condition on a graph specification of the target structure. They can efficiently captures the complex dependencies in proteins by focusing on those that are long-range in sequence but local in 3D space.
- **GVP**: They extend standard dense layers to operate on collections of Euclidean vectors. Utilizing such layers can help model to leverage the geometric and relational aspects.
- **PiFold**: They propose a novel residue featurizer to learn multi-scale residue interactions and PiFold can generate the sequence in one shot as a result it can achieve a great effectiveness.
- **AlphaDesign**:They use AlphaFoldDB to establish a new graph-based benchmark. Furthermore, they use a simplified graph transformer encoder to introduce protein angles and propose a confidence-aware protein decoder. Such modules can efficiently improve the performance.
- **ProteinMPNN**: They begins from a message passing neural network which can effectively explore the information of backbones. Then, they replaced the fixed decoding order with an order agnostic autoregressive model, which means the decoding order is random. Such strategy can efficiently improve the performance for inverse folding.
- **ESM-IF**:They augments the inverse folding dataset with folding data created by AlphaFold2. Training with additional data, the performance achieves a great progress.
- **LM-Design**: They conduct a lightweight structural adapter which is implanted into protein language models and endows it with structural awareness.
- **KW-Design**: They propose a knowledge-aware module that refines low-quality residues. Furthermore, they introduce a memory-retrieval mechanism to save more training time.

A.2 DATASETS

In our experiments, we evaluate the proposed Hier-IF model on two widely used protein domain datasets: CATH4.2Orengo et al. (1997) and CATH4.3Hsu et al. (2022). These datasets provide a reliable benchmark for assessing the capacity of models to generalize across diverse protein topologies. For CATH4.2, we follow the experimental setup described in GraphTransIngraham et al. (2019). We utilize all protein domains from the CATH4.2 40% non-redundant set. From these domains, we extract the corresponding full protein chains, filtering out those longer than 500 residues to control computational complexity and ensure consistency with prior work. The resulting dataset is then split based on CATH topology classification codes into training, validation, and test sets, maintaining an 80/10/10 ratio. This ensures that each topology appears in only one of the three sets, promoting rigorous generalization evaluation. For CATH4.3, we adopt the protocol introduced in ESM-IFHsu et al. (2022). Specifically, we split the topology classification codes from CATH4.3 into training, validation, and test sets using the same 80/10/10 ratio. We extract full chains of no more than 500 residues, ensuring consistency across datasets and alignment with computational constraints.

A.3 TRAINING

The models are trained up to 100 epochs by default using the Adam optimizer on NVIDIA A6000s. We use the training setting as ProteinMPNN, where the batch size is set to 6000 residues. Furthermore, we employ the Noam learning rate scheduler**?**, which is widely used in transformer-based models. This scheduler starts with a warm-up phase where the learning rate increases linearly, followed by a decay phase where it decreases proportionally to the inverse square root of the training step.

B Implementation details of Hier-IF

B.1 Masked Diffusion Model

We formulate the mask-based diffusion process over discrete or structured protein representations (e.g., residue identities or torsion angles) using a forward process that progressively masks input tokens, and a reverse process that learns to recover them.

B.1.1 Forward Process (Masking)

Given a protein representation $x_0 \in \mathcal{X}^L$, where L is the sequence length, the forward process defines a sequence of increasingly masked inputs $\{x_t\}_{t=0}^T$. At each timestep t, a binary mask $m_t \in \{0,1\}^L$ is sampled from a masking schedule $q(m_t \mid t)$, and we define:

$$x_t = m_t \odot \texttt{[MASK]} + (1 - m_t) \odot x_{t-1}$$

where $\odot$ denotes elementwise multiplication, and `[MASK]` is a special token representing an unknown residue or feature.

The marginal distribution of x_t given x_0 can be written as:

$$q(x_t \mid x_0) = \mathbb{E}_{m_t}[x_t \mid x_0] = m_t \odot \texttt{[MASK]} + (1 - m_t) \odot x_0$$

B.1.2 Reverse Process (Unmasking)

The denoising model p_θ learns to reconstruct the original input by predicting the masked elements:

$$p_\theta(x_{t-1} \mid x_t) = \prod_{i \in \mathcal{M}_t} p_\theta(x_{t-1}^{(i)} \mid x_t)$$

where $\mathcal{M}_t = \{i \mid m_t^{(i)} = 1\}$ is the set of positions masked at timestep t.

For protein sequence modeling, each term is modeled by a categorical distribution over the 20 standard amino acids:

$$p_\theta(x_{t-1}^{(i)} = a \mid x_t) = \text{softmax}(f_\theta(x_t))_a \tag{1}$$

where f_θ is a neural network (e.g., a transformer) that outputs logits for each residue position.

B.2 Mask Generator

To enable a structure-aware and learnable masking strategy, we introduce a *mask generator* that produces a soft masking score for each residue at every diffusion timestep. The mask generator is implemented as a 3-layer multi-layer perceptron (MLP), which takes as input both the DSSP-derived secondary structure and the current diffusion timestep.

Each residue's secondary structure s_i is encoded as a one-hot vector:

$$\text{onehot}(s_i) = \begin{cases} [1,0,0], & \text{if } s_i = \text{H (helix)} \\ [0,1,0], & \text{if } s_i = \text{E (strand)} \\ [0,0,1], & \text{if } s_i = \text{C (coil)} \end{cases}$$

The mask generator outputs a soft mask $\tilde{m} \in (0,1)^L$, where each element $\tilde{m}_i$ represents the probability of masking the i-th residue. To encourage discrete masking behavior during training, a hard mask $m \in \{0,1\}^L$ is sampled from $\tilde{m}$ according to a Bernoulli distribution:

$$m_i \sim \text{Bernoulli}(\tilde{m}_i).$$

Since this sampling operation is inherently non-differentiable, it blocks gradient backpropagation through m. To mitigate this, we employ the Straight-Through Estimator (STE) technique, which treats the sampling step as an identity function during the backward pass. Concretely, the gradient is directly propagated from the loss with respect to the hard mask m back to the soft mask $\tilde{m}$:

$$\frac{\partial \mathcal{L}}{\partial \tilde{m}} \approx \frac{\partial \mathcal{L}}{\partial m}.$$

Alternatively, training can be conducted solely with the soft mask $\tilde{m}$ to maintain full differentiability, at the cost of losing discrete mask interpretability.

This approach enables the mask generator to learn meaningful, time-dependent masking strategies that balance discrete perturbations with smooth optimization.

—

The diffusion timestep t is represented as a scalar, either as an integer in the diffusion schedule $t \in \{0, \dots, T\}$ or normalized to a continuous value in the range $[0, 1]$. This scalar is then mapped to a higher-dimensional embedding via a small learnable MLP:

$$\tau(t) = W_2 \cdot \mathrm{ReLU}(W_1 t + b_1) + b_2, \quad \tau(t) \in \mathbb{R}^{d_t},$$

where $W_1 \in \mathbb{R}^{h \times 1}$, $W_2 \in \mathbb{R}^{d_t \times h}$, and h denotes the hidden dimension size.

For each residue i, the input vector to the mask generator is formed by concatenating the one-hot encoded secondary structure and the time embedding:

$$z_i = \mathrm{concat}\big(\mathrm{onehot}(s_i),\, \tau(t)\big) \in \mathbb{R}^{3+d_t}.$$

The mask generator g_ϕ is a 3-layer MLP that maps z_i to the soft mask score:

$$\tilde{m}_i = \sigma\left(W_3 \cdot \mathrm{ReLU}\big(W_2 \cdot \mathrm{ReLU}(W_1 z_i + b_1) + b_2\big) + b_3\right),$$

where $\sigma(\cdot)$ denotes the sigmoid activation, ensuring $\tilde{m}_i \in (0, 1)$ represents the masking probability for residue i.

—

The resulting soft mask $\tilde{m} \in (0, 1)^L$ is sampled during training to produce a hard mask m, while at inference time, a deterministic mask can be obtained by thresholding or selecting the top-k residues with highest mask scores.

This design allows the model to learn residue-level, structure- and time-aware masking strategies that adapt dynamically according to both the diffusion stage and the protein secondary structure context.

B.3 Soft vs. Hard Masking

We distinguish between two types of masking strategies: *soft* masking and *hard* masking, based on how the mask is applied during training and inference.

Soft masking. In soft masking, the output $\tilde{m} \in (0, 1)^L$ from the mask generator is directly interpreted as a probabilistic or fractional mask. This enables smooth, differentiable updates, and is especially useful when the mask is applied via interpolation:

$$\hat{x} = \tilde{m} \odot \text{noise} + (1 - \tilde{m}) \odot x$$

Here, each position is softly corrupted in proportion to its predicted mask score. This allows gradients to flow through the masking mechanism during backpropagation, and facilitates end-to-end training.

Hard masking. In hard masking, the soft mask $\tilde{m}$ is used to sample a binary mask $m \in \{0, 1\}^L$, typically as:

$$m_i \sim \text{Bernoulli}(\tilde{m}_i)$$

This sampled mask is then used to fully corrupt certain positions and leave others untouched. While this approach aligns more closely with classical denoising objectives and provides stronger perturbations, the sampling process is non-differentiable and introduces variance during training. Common workarounds include using the Straight-Through Estimator (STE) or Gumbel-softmax relaxation.

Comparison. Soft masking provides smoother training dynamics and supports gradient-based optimization of the masking policy. Hard masking, in contrast, introduces more discrete and interpretable corruption patterns, often preferred at inference time or when seeking sharper denoising effects.

In our implementation, we employ soft masking during training to ensure gradient flow, and optionally apply hard masking at inference via either thresholding or top-k selection based on the predicted $\tilde{m}$.

B.4 DSSP Secondary Structure Annotation

DSSP (Define Secondary Structure of Proteins) **?** is a widely used algorithm for assigning secondary structure elements to each residue in a protein based on its 3D atomic coordinates. It categorizes residues into eight classes:

$$\{\mathrm{H}, \mathrm{B}, \mathrm{E}, \mathrm{G}, \mathrm{I}, \mathrm{T}, \mathrm{S}, \mathrm{C}\}$$

where

- H: Alpha-helix
- B: Isolated beta-bridge residue
- E: Extended strand, participates in beta ladder
- G: 3_{10} helix
- I: Pi-helix
- T: Turn
- S: Bend
- C: Coil (none of the above)

For many applications, including ours, a simplified secondary structure representation is preferred. We map the original 8 classes into 3 broader categories, commonly referred to as *Helix (H)*, *Strand (E)*, and *Coil (C)*, as follows:

$$\begin{cases} \mathrm{H} = \{\mathrm{H}, \mathrm{G}, \mathrm{I}\} & \text{(Helices)} \\ \mathrm{E} = \{\mathrm{E}, \mathrm{B}\} & \text{(Strands)} \\ \mathrm{C} = \{\mathrm{T}, \mathrm{S}, \mathrm{C}\} & \text{(Coils/Loops)} \end{cases}$$

This reduction simplifies the secondary structure annotation while preserving the key structural motifs relevant for downstream modeling and masking tasks.

C Evaluation Metric

Median Recovery Median Recovery measures the similarity between generated protein sequences and target reference sequences, commonly used to evaluate reconstruction accuracy. Given a set of sequence pairs $\{(S_i^{\mathrm{gen}}, S_i^{\mathrm{ref}})\}_{i=1}^N$, the recovery for each pair is defined as the proportion of matching amino acids at corresponding positions:

$$\mathrm{Recovery}_i = \frac{1}{L_i} \sum_{j=1}^{L_i} \mathbb{I}\big(S_{i,j}^{\mathrm{gen}} = S_{i,j}^{\mathrm{ref}}\big),$$

where L_i is the length of the i-th sequence and $\mathbb{I}(\cdot)$ is the indicator function. Median Recovery is the median of all individual recoveries:

$$\text{Median Recovery} = \mathrm{median}\big(\{\mathrm{Recovery}_i\}_{i=1}^N\big).$$

A higher median recovery indicates better agreement between generated and reference sequences.

Perplexity Perplexity is a widely used metric to assess the quality of probabilistic sequence models, reflecting how well a model predicts a given protein sequence. For a sequence $S = (s_1, s_2, \ldots, s_L)$ and a model parameterized by θ that estimates conditional probabilities $p_\theta(s_j | s_{<j})$, the log-likelihood is:

$$\log p_\theta(S) = \sum_{j=1}^{L} \log p_\theta(s_j | s_{<j}).$$

The perplexity of the sequence is defined as:

$$\mathrm{Perplexity}(S) = \exp\left(-\frac{1}{L} \log p_\theta(S)\right).$$

Lower perplexity indicates that the model predicts the sequence with higher confidence and better accuracy.

Designability A protein backbone is considered *designable* if there exists at least one amino acid sequence that folds into that structure. Our evaluation of designability follows the methodology outlined by **?**. For each backbone generated by a model, we produce eight candidate sequences using ProteinMPNN Dauparas et al. (2022) with a sampling temperature of 0.1. Each designed sequence is then folded using ESMFold Lin et al. (2023) to predict its 3D structure.

We calculate the root mean square deviation (RMSD) between each predicted structure and the model's original backbone. The lowest RMSD among the eight predictions, termed the self-consistency RMSD (scRMSD), is used to evaluate designability. A backbone is classified as designable if its scRMSD is below a threshold of 2 Å.

The overall designability score for a model is computed as the fraction of generated backbones that meet this criterion:

$$\text{Designability} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\big(\text{scRMSD}(S_i) < 2\,\text{Å}\big),$$

where N is the total number of generated backbones and $\mathbf{1}(\cdot)$ is the indicator function.

This approach quantitatively measures the practical feasibility of designing sequences that reliably fold back to the intended structures, thus reflecting the quality of the generative model.

Structural Diversity via Foldseek Structural diversity measures the extent to which generated protein sequences fold into a variety of distinct structural conformations, which is important for exploring novel folds and functions.

Foldseek **?** is a state-of-the-art tool for fast and sensitive structural comparison and clustering of large protein datasets. After predicting 3D structures for designed sequences, Foldseek is used to perform pairwise structural alignments and group proteins into clusters based on their similarity.

Diversity is then quantified by metrics such as:

- The number of distinct structural clusters detected at a given similarity cutoff, reflecting the breadth of structural exploration.
- Average pairwise structural dissimilarity (e.g., average RMSD or 1 - TM-score) among the generated structures.

D DIFFERENT SAMPLING STRATEGY

Method	Recovery			Perplexity		
	Short	Single-Chain	All	Short	Single-Chain	All
Iterative Refinement	0.40	0.41	0.53	6.97	6.02	5.80
Time-aware Mask Generator	**0.46**	**0.46**	**0.59**	**5.46**	**5.17**	**4.01**

Masked diffusion models generate discrete sequences by iteratively predicting masked tokens. While these models are efficient and parallelizable, initial predictions may suffer from inconsistencies or errors, particularly when conditioned on complex structures (e.g., protein 3D conformation). To address this, **iterative refinement** is introduced as a mechanism to improve generation quality by progressively revising uncertain predictions across multiple steps.

Let a discrete sequence of length L be denoted by $\mathbf{x} = [x_1, x_2, \ldots, x_L]$, and let $\mathbf{x}^{(0)}$ be the initial input where some tokens are masked. The goal is to iteratively update $\mathbf{x}^{(t)}$ to produce a coherent sequence $\hat{\mathbf{x}}^{(T)}$ after T refinement steps.

At each iteration $t = 1, \ldots, T$, the refinement process involves three stages:

1. **Prediction:** The masked model f_θ predicts the token distribution given the current sequence:

$$\hat{\mathbf{x}}^{(t)} = f_\theta(\mathbf{x}^{(t-1)}, \text{cond})$$

where `cond` denotes optional conditioning information (e.g., structural constraints).

2. **Remasking:** Based on prediction confidence (e.g., entropy or max softmax probability), a subset $\mathcal{M}^{(t)}$ of uncertain positions is selected to be masked again:

$$\mathcal{M}^{(t)} = \text{TopKUncertain}(\hat{\mathbf{x}}^{(t)})$$

3. **Update:** The input for the next step is updated as:

$$x_i^{(t)} = \begin{cases} \hat{x}_i^{(t)}, & \text{if } i \notin \mathcal{M}^{(t)} \\ \texttt{[MASK]}, & \text{if } i \in \mathcal{M}^{(t)} \end{cases}$$

This process is repeated for T iterations to progressively refine the prediction toward a coherent and high-quality output.

Following Zheng et al. (2023), we implement the iterative refinement strategy for sequences generation. Compared with the time-aware Mask Generator proposed, we find our methods can achieve better performance. Because of the awareness of the hierarchical information, Hier-IF can design the protein sequence with higher recovery and perplexity.

REFERENCES

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.

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

Zhangyang Gao, Cheng Tan, and Stan Z Li. Alphadesign: A graph protein design method and benchmark on alphafolddb. *arXiv preprint arXiv:2202.01079*, 2022b.

Zhangyang Gao, Cheng Tan, and Stan Z Li. Knowledge-design: Pushing the limit of protein design via knowledge refinement. *arXiv preprint arXiv:2305.15151*, 2023.

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 JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

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.

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.

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, 2023.