P(*all-atom*) Is Unlocking New Path For Protein Design

Wei Qu [1 2] **Jiawei Guan** [2] **Rui Ma** [2] **Ke Zhai** [2]
Weikun Wu [† 2] **Haobo Wang** [† 2]

Abstract

We introduce Pallatom, an innovative protein generation model capable of producing protein structures with all-atom coordinates. Pallatom directly learns and models the joint distribution $P(structure, seq)$ by focusing on $P(all\text{-}atom)$, effectively addressing the interdependence between sequence and structure in protein generation. To achieve this, we propose a novel network architecture specifically designed for all-atom protein generation. Our model employs a dual-track framework that tokenizes proteins into residue-level and atomic-level representations, integrating them through a multi-layer decoding process with "traversing" representations and recycling mechanism. We also introduce the `atom14` representation method, which unifies the description of unknown side-chain coordinates, ensuring high fidelity between the generated all-atom conformation and its physical structure. Experimental results demonstrate that Pallatom excels in key metrics of protein design, including designability, diversity, and novelty, showing significant improvements across the board. Our model not only enhances the accuracy of protein generation but also exhibits excellent training efficiency, paving the way for future applications in larger and more complex systems.

1. Introduction

The theoretical foundation of protein modeling has been built upon two key conditional probability distributions: $P(structure \mid seq)$ and $P(seq \mid backbone)$. The former, $P(structure \mid seq)$, corresponds to the all-atom protein structure prediction task, which involves determining the three-dimensional structure of a protein given its amino acid sequence (Abramson et al., 2024; Jumper et al., 2021; Lin et al., 2023; Baek et al., 2023). The latter, $P(seq \mid backbone)$, underpins the fixed-backbone design task, where the goal is to identify a sequence that will fold into a given protein backbone structure (Dauparas et al., 2022; Hsu et al., 2022). In summary, these probability distributions has successfully advanced the field of protein engineering.

With the advancement of deep learning in protein science, two distinct approaches for protein design have emerged. One approach is the protein hallucination (Anishchenko et al., 2021), which explores the landscape of a $P(structure \mid seq)$ model using Monte Carlo or gradient-based optimization techniques. This method yields valid protein structures, but requires an additional $P(seq)$ model, such as protein language models (Rives et al., 2021), to correct or redesign the sequence. Essentially, this approach can be viewed as optimization process of $P(structure \mid seq) \cdot P(seq)$. Another approach attempts to explore the $P(backbone)$ distribution. a series of protein generation models based on SE(3) invariance or equivariance networks (Jing et al., 2020; Satorras et al., 2021) have recently emerged, these method rely on an additional $P(seq \mid backbone)$ process to determine the protein sequence. This optimization strategy can be regarded as $P(backbone) \cdot P(seq \mid backbone)$.

This step-wise design process has limitation in approximating the joint distribution through marginal distributions. The $P(structure \mid seq) \cdot P(seq)$ strategy faces challenges when sampling in the high-dimensional sequence space, while the $P(backbone) \cdot P(seq \mid backbone)$ strategy fails to account for explicit side-chain interactions and is bottlenecked by the capability of the fixed-backbone design model.

The ultimate goal of protein generation is to directly obtain a sequence along with its corresponding structure, i.e., to develop a model capable of describing the joint distribution $P(structure, seq)$ or $P(backbone, seq)$. Recently, some studies have started to adopt co-generation approaches, such as model based on co-diffusion (Campbell et al., 2024) or co-design (Ren et al., 2024). While these methods primarily rely on SE(3) networks, they still separately model the backbone and sequence, without considering side-chain con-

[*]Equal contribution [1]Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai, China [2]LEVINTHAL Biotechnology Co.Ltd, Hangzhou, China. Correspondence to: Weikun Wu <weikun.wu@levinthal.bio>, Haobo Wang <haobowang@levinthal.bio>.

formations and leading to an insufficient description of the structure. Protpardelle (Chu et al., 2024), an all-atom protein diffusion model, similarly adopts co-generation approaches with an explicit all-atom representation, taking a step further in the field. However, the experimental results indicate that the generated sequence fails to accurately encode the intended fold, necessitating an additional round of sequence redesign and side-chain refinement.

In this study, we introduce a novel approach for all-atom protein generation called **Pallatom**. Our extensive experiments show that by learning $P(\textit{all-atom})$, high-quality all-atom proteins can be successfully generated, eliminating the need to learn marginal probabilities separately. To address the first critical challenge of representing side-chain coordinates for unknown amino acids in protein generation, we introduce `atom14`, a novel all-atom representation framework. This approach employs virtual atoms across all amino acid types, effectively preventing sequence information leakage while maintaining structural integrity. Inspired by AlphaFold3 (AF3) (Abramson et al., 2024), we propose a dual-track representation framework that encodes protein structures through concurrent residue-level and atom-level tokenization schemes. Central to this framework is the AtomDecoder unit, which seamlessly integrates and updates features across multiple representation spaces while simultaneously executing coordinate prediction and differentiable recycling process within a unified computational architecture. The fundamental insight underlying Pallatom is the recognition that protein all-atom coordinates intrinsically encapsulate both structural and sequential information within their spatial configuration. Directly learning $P(\textit{all-atom})$ opens a new path for co-generative modeling of structure and sequence.

Our contributions are summarized as follows:

- We explore the `atom14` representation to achieve a unified description of unknown amino acid side-chain coordinates in generative tasks.
- We develop a network architecture for all-atom protein generation tasks, which effectively represents both protein backbones and sidechains.
- We use our framework to develop Pallatom, a state-of-the-art all-atom protein generative model.

2. Preliminaries

2.1. All-atom modeling and representation

Recent advances in protein structure prediction have established diverse methodologies for all-atom representation. AlphaFold2 (AF2) (Jumper et al., 2021) pioneered a frame-based approach, modeling atomic coordinates through SE(3) transformations of backbone and side-chain frames. Building upon this foundation, AF3 introduces an innovative point cloud representation that directly encodes all-atom 3D coordinates in Cartesian space. However, these representations are fundamentally inadequate for protein generation tasks due to a critical limitation: the complete absence of sequence information at the initialization stage prevents determination of the system's total atomic cardinality. To overcome these limitations, Protpardelle introduces the `atom73` representation, which implements a quantum-inspired superposition framework. This approach stores coordinate information for all 20 possible side-chain configurations relative to aligned backbone atoms, enabling probabilistic representation of unknown amino acid types. The superposition collapses to a definitive state upon sequence determination, effectively resolving the initial uncertainty in atomic configuration.

2.2. Diffusion modeling on all-atom protein

The all-atom representation bypasses the complexities of SE(3) frames (Yim et al., 2023b) and Riemannian diffusion (De Bortoli et al., 2022). Gaussian-based diffusion models provide a robust theoretical framework, with EDM emerging as a prominent approach. EDM has been successfully implemented in Alphafold3 and Protpardelle for all-atom coordinate diffusion, showcasing its efficacy in protein generation. We briefly outline EDM's mechanism below.

Assuming the data distribution of all-atom coordinates under any representation as $p_{data}(\mathbf{x})$ with standard deviation σ_{data}, the forward process involves adding Gaussian noise of varying scales to generate a series of noised distributions $p_t(\mathbf{x}) = [p_{data} * \mathcal{N}(\mathbf{0}, \sigma(t)^2\mathbf{I})](x) \triangleq p(\mathbf{x};\sigma)$ with a particular configuration $\sigma(t) = t$ and $s(t) = 1$. When $\sigma_{max} \gg \sigma_{data}$, $p_{\sigma_{max}}(\mathbf{x}) \approx \mathcal{N}(0, \sigma_T^2)$ approximates pure Gaussian noise. The probability flow ordinary differential equation (ODE) is given by:

$$\mathrm{d}\mathbf{x} = -\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma(t))\mathrm{d}t \tag{1}$$

Here, $\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma)$ is the score function, which does not depend on the normalization constant of the underlying density function $p_t(\mathbf{x})$. A neural network $D_\theta(\mathbf{x};\sigma)$ is typically trained for each σ using the following loss function to match the score function (Song et al., 2021):

$$\mathbb{E}_{\mathbf{y}\sim p_{data},\mathbf{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}[\lambda(\sigma)||D_\theta(\mathbf{y}+\mathbf{n};\sigma)-\mathbf{y}||_2^2], \tag{2}$$

$$\nabla_{\mathbf{x}} \log p(\mathbf{x};\sigma) = (D_\theta(\mathbf{x};\sigma)-\mathbf{x})/\sigma^2 \tag{3}$$

In practice, EDM introduced a preconditioning technique that stabilizes training by adjusting noise-related scaling coefficients, ensuring that the model's inputs and outputs remain within a stable numerical range at each time step.

Consequently, D_θ and the loss function can be derived as:

$$D_\theta(\mathbf{x};\sigma) = c_{skip}(\sigma)\mathbf{x} + c_{out}(\sigma)F_\theta(c_{in}(\sigma)\mathbf{x}, c_{noise}(\sigma))$$

$$\mathbb{E}_{\sigma,\mathbf{y},\mathbf{n}}[\lambda(\sigma)||c_{out}(\sigma)F_\theta(c_{in}(\sigma)\cdot(\mathbf{y}+\mathbf{n}), c_{noise}(\sigma)) - (\mathbf{y} - c_{skip}(\sigma)\cdot(\mathbf{y}+\mathbf{n}))||_2^2] \quad (4)$$

where $c_{skip}(\sigma) = \sigma_{data}^2/(\sigma^2 + \sigma_{data}^2)$, $c_{out}(\sigma) = \sigma \cdot \sigma_{data}/\sqrt{\sigma_{data}^2 + \sigma^2}$, $c_{in}(\sigma) = 1/\sqrt{\sigma^2 + \sigma_{data}^2}$, $c_{noise}(\sigma) = \frac{1}{4}\ln(\sigma)$ and $\lambda(\sigma) = 1/c_{out}(\sigma)^2$ represent skip scaling, output scaling, input scaling, noise conditioning, and the Coefficient respectively.

2.3. Framework for structure-based protein generation

Currently, the field of protein structure prediction lacks comprehensive methods for all-atom coordinate generation. The most relevant approaches are structure-based backbone design or sequence-backbone co-generation methods, which typically rely on a core building block to perform multi-view feature fusion and coordinate updates on protein representations (e.g., 1D single-, 2D pair-, and 3D frame-embeddings). By stacking multiple such blocks, these methods progressively refine protein structures. For instance, RFDiffusion (Watson et al., 2023), built on RoseTTAFold (Baek et al., 2021), integrates information from three tracks within each block, continuously updating and fusing representations through sophisticated network operations. Similarly, MultiFlow (Campbell et al., 2024) and CarbonNovo (Ren et al., 2024) adopt Alpahfold2's Invariant Point Attention (IPA) layer as their building block, augmented with additional mechanisms to refine pairwise and sequence information. Through multi-view feature fusion and coordinate updates, these methods achieve precise backbone generation.

In contrast, ProtPardelle uses a simpler U-ViT (Bao et al., 2023) architecture to generate all-atom proteins but relies on a post-processing miniMPNN module to redesign sequences and guide side-chain placement, which limits its robustness. This highlights a key limitation of simplified frameworks: their inability to effectively integrate diverse information sources. In contrast, the complex network frameworks offer significant advantages to incorporate additional auxiliary information. For example, RFDiffusion employs a self-conditioning mechanism (Chen et al., 2022) to generate preliminary structures as 2D templates for guiding sampling, while CarbonNovo leverages encoded single- and pair-features to perform sequence sampling through Markov Random Field (MRF). These strategies significantly improve the diversity and quality of generated proteins. Thus, exploring all-atom generation frameworks that integrate multiple encoding forms remains a critical research direction.

3. Method

3.1. `atom14`: the all-atom representaion

The all-atom protein generation model faces many challenges in constructing both backbone and side-chain atoms. A pivotal initial question arises: How to represent a system with a variable number of atoms? At the initial sampling stage, both the backbone and sequence are unknown, however, the atom number of a system depends on unique sequence, once the sequence is determined, it also dictates the structure.

To avoid potential conflicts arising from the simultaneous design of sequence and structure, we introduce the `atom14` representation, establishing a unified framework through virtual atom integration that normalizes heavy atom counts across all amino acid types. This padding strategy that empirically positions additional virtual atoms to align with the Cα coordinates of each amino acid residue. Through this representation, a protein with N residues, denoted as $\mathbf{x} = \{x_i\}_{i=1}^N$, is transformed into a fixed-size point cloud $\mathbf{x}^0 \in \mathbb{R}^{N\times 14\times 3}$. For example, if residue x_i is `CYS`, its coordinates $[\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}, \mathrm{O}, \mathrm{C}_\beta, \mathrm{S}_\gamma] \in \mathbb{R}^{6\times 3}$ are augmented with eight virtual atoms aligned at the C_α position, yielding $x_i^0 \in \mathbb{R}^{14\times 3}$. We denote the system as $\vec{\mathbf{r}} = \left\{\vec{\mathbf{r}}_l \mid \vec{\mathbf{r}}_l \in \mathbb{R}^3, l = 1, 2, \ldots, L\right\}$ with $L = N \times 14$ and define the noise level as $\hat{t}$.

The primary task of all-atom generation involves learning the score function $\nabla_{\vec{\mathbf{r}}^{\hat{t}}} \log p(\vec{\mathbf{r}}^{\hat{t}})$ through a diffusion process that progressively perturbs the initial state $\vec{\mathbf{r}}^0$ at each noise level $\hat{t}$. While the virtual atom framework inherently precludes explicit element-specific assignments, we posit that the all-atom coordinate distribution sufficiently encodes essential sidechain properties, including hydrophobicity, polarity, hydrogen bonding, and salt bridge formation. Therefore, we introduce an auxiliary visualization module that predicts amino acid types. With discarding the redundant virtual atoms based on the predicted amino acid type as a post-processing step, we can generate all-atom proteins with corresponding sequences from a 3D point cloud noise.

3.2. `MainTrunk`: the denoising network

We refer to the protein generation task as the generation of all-atom coordinates. In the `atom14` representation, a protein with L atoms can be expressed as $\vec{\mathbf{r}}^0 = \{\vec{\mathbf{r}}_l^0\}_{l=1}^L$, which $\vec{\mathbf{r}}_l^0 \in \mathbb{R}^3$ represents an atom coordinate. For each noise level $\hat{t}$ in the diffusion process, the network predicts the updated coordinates $\vec{\mathbf{r}}^{\text{denoised}}$ from the input $\vec{\mathbf{r}}^{\text{input}}$.

The network comprises two main components: a feature encoder and iterative decoding units. Figure 1A illustrates the main architecture. We implement a dual-track representation framework, employing local attention mechanisms for

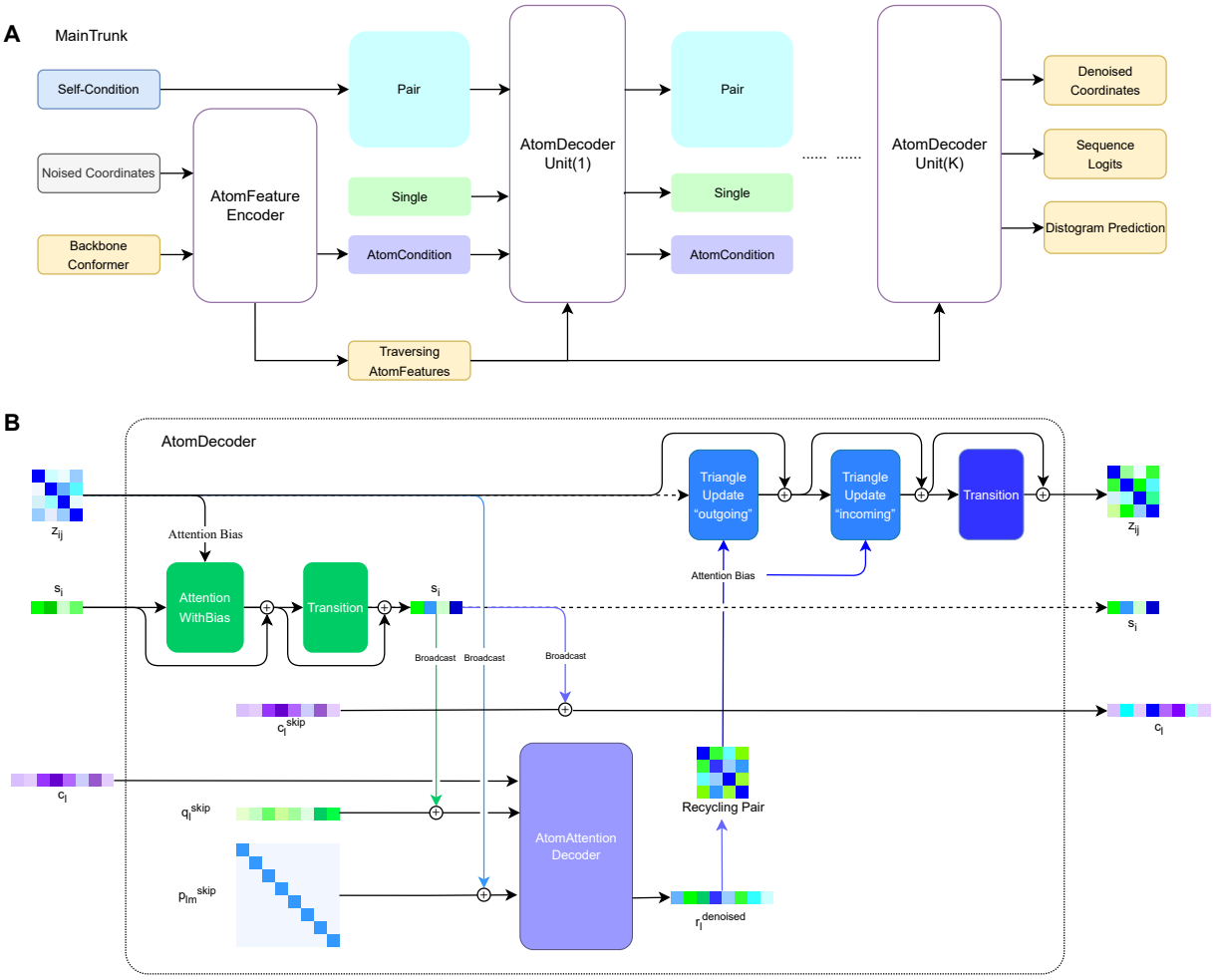

Figure 1. Pallatom model framework: (A, B) Architecture of the network and the AtomDecoder unit.

atomic-level feature extraction and global attention mechanisms for residue-level feature integration.

For feature initialization and encoding (Algorithm 4), We first initialize "traversing" atomic embeddings (c_l^{skip}, q_l^{skip} and p_{lm}^{skip} by integrating three key components: (1) structural features from alanine's standard conformation to establish a stable backbone frame, (2) residue positional encoding, and (3) noisy coordinate vectors. For residue-level embedding, single embedding (s_i) is initialized using diffusion timestep encoding and residue positional encoding, while pair features (z_{ij}) incorporate relative positional encoding and self-condition template distograms from previous predictions. Detailed features are recorded in Appendix Table 4.

In the decoding phase, we implement an iterative refinement mechanism that facilitates bidirectional information flow: residue-level embedding are propagated to atomic-level, while atomic information is subsequently integrated back through a differentiable recycling process. These operations are encapsulated within an AtomDecoder unit (Figure 1B), which is designed to continuously update the coordinates. In practical implementation, we identify a fundamental challenge in the decoding architecture: preserving residual connections between residue-level and atomic-level embeddings across successive decoder units may result in significant information redundancy, especially during the residue-level feature broadcasting process. A comprehensive analysis of this phenomenon and our solution is provided in Appendix C.4. To address this challenge, the three "traversing" atomic embeddings serve as stable information carriers, facilitating efficient residue-level propagation while enabling coordinate refinement through the accumulated $\vec{\mathbf{r}}_{(k)}^{\text{updates}}$ in the k-th unit.

$$\vec{\mathbf{r}}_{(k)}^{\text{denoised}} = c_{skip}(\hat{t}) \cdot \vec{\mathbf{r}}^{\text{input}} + c_{out}(\hat{t}) \cdot \vec{\mathbf{r}}_{(k)}^{\text{updates}} \tag{5}$$

Following coordinate denoising, the partially denoised posi-

tions $\vec{\mathbf{r}}_{(k)}^{\text{denoised}}$ are transformed into relative distance matrices and subsequently recycled into the z_{ij} pair embeddings. Concurrently, the triangle update module selectively maintains physically valid triangular constraints while eliminating geometrically inconsistent features, enabling iterative improvement of structural quality throughout the decoding process. Details can be found at Appendix Algorithm 2.

3.3. `SeqHead`: sequence decoder

The `atom14` representation, while effective for structural denoising, inherently lacks elemental information in the generated side chains, preventing direct extraction of protein sequence information from the atomic coordinates alone. To bridge the gap between structural generation and biological interpretation, we introduce a `SeqHead` module that converts the predicted atomic coordinates into corresponding amino acid sequences, enabling the transformation of geometric data into biologically meaningful protein structures. Specifically, we aggregate the atomic embedding (q_l) corresponding to each residue and process them through a linear transformation layer to predict probability distributions over the 20 canonical amino acid types $\hat{\mathbf{a}}_{(k)} \in \mathbb{R}^{N\times 20}$. We take the predictions from the last unit as the final sequence logits, $\hat{\mathbf{a}} = \hat{\mathbf{a}}_{(K)}$.

3.4. Training Loss

Our training method mainly follows the application and improvements of the EDM framework. The denoising all-atom positions score-matching losses are described according to Eq.(2). Given that the network architecture lacks inherent equivariance constraints and iteratively refines coordinates across multiple decoding stages, it is imperative that the coordinates transformations between these stages remain invariant under changes in orientation to ensure consistent geometric interpretations. Therefore, we employ an aligned MSE loss. We first perform a rigid alignment of the ground truth $\vec{\mathbf{r}}^0$ on the denoised structure $\vec{\mathbf{r}}^{\text{denoised}}$ as $\vec{\mathbf{r}}^{\text{aligned}}$. The MSE loss is then defined as: $\mathcal{L}_{\text{atom}} = \frac{||\vec{\mathbf{r}}^{\text{denoised}} - \vec{\mathbf{r}}^{\text{aligned}}||^2}{3L}$

For sequence decoding, we use the standard cross-entropy loss function to evaluate the difference between the predicted sequence $\hat{\mathbf{a}}$ and the true sequence $\mathbf{a}_0$. The loss function is defined as $\mathcal{L}_{\text{seq}} = \text{CE}(\hat{\mathbf{a}}, \mathbf{a}^0)$. The primary loss function of the network is:

$$\mathcal{L}_0 = \lambda(\hat{t}) \cdot \mathcal{L}_{\text{atom}} + \alpha_0 \cdot \mathcal{L}_{\text{seq}} \tag{6}$$

To capture the fine-grained characteristics of the all-atom structure, we introduce the smooth local distance difference test (LDDT) loss $\mathcal{L}_{\text{smooth_lddt}}(\vec{\mathbf{r}}^{\text{denoised}}, \vec{\mathbf{r}}^0)$. This can be found in the Appendix algorithm 8. Additionally, we implement two distogram-based loss function $\mathcal{L}_{\text{dist_res}}$ and $\mathcal{L}_{\text{dist_atom}}$ to enforce global/local distance constraints at the residue/atomic level, ensuring proper maintenance of overall protein topology throughout the optimization process. We introduce an intermediate layer loss $\mathcal{L}_{med}$, that supervises the sequence and structure predictions from each AtomDecoder unit, significantly enhancing both model performance and inference stability through progressive refinement. Details of training losses can be found in Appendix D.

Algorithm 1 Pallatom Inference

def SampleDiffusion ($\{f^*\}$, $T = 200$, $\lambda = 1.003$, $\eta = 2.25$, $\gamma_0 = 0.2$, $t_{min} = 0.01$, $t_{max} = 1.0$)
$\delta_t = 1/T$
$c_T = \text{GetNoiseSchedule}(1 - \text{uniform}(0,1) \cdot \delta_t)$
$\vec{\mathbf{r}}_l \sim c_T \cdot \mathcal{N}(\vec{0}, \mathbf{I}_3)$
Initialize the self-condition feature
$f^{\text{template_distogram}} = \mathbf{0}, \in \{f^*\}$
for all $t \in \{T, \ldots, 1\}$ **do**
  $t_p = t/T - \text{uniform}(0,1) \cdot \delta_t$
  $c_\tau = \text{GetNoiseSchedule}(t_p)$
  $c_{\tau-1} = \text{GetNoiseSchedule}(t_p - \delta_t)$
  $\vec{\mathbf{r}}_l \leftarrow \text{CentreRandomAugmentation}(\vec{\mathbf{r}}_l)$
  $\gamma = \gamma_0$ if $t_{min} \le t/T \le t_{max}$ else 0
  $\hat{t} = c_\tau(\gamma + 1)$
  $\vec{\mathbf{r}}_l^{\text{noisy}} = \vec{\mathbf{r}}_l + \lambda\sqrt{\hat{t}^2 - c_\tau^2} \cdot \mathcal{N}(\vec{0}, \mathbf{I}_3)$
  # *Update the self-condition feature*
  $\vec{\mathbf{r}}_l^{\text{denoised}}, \mathbf{f}_i^{\text{seq_logits}} = \text{MainTrunk}($
    $\{f^*\}, \vec{\mathbf{r}}_l^{\text{noisy}}, \hat{t}, t_p)$
  $f^{\text{template_distogram}} \leftarrow \text{Distogram}(\vec{\mathbf{r}}_l^{\text{denoised}})$
  # *Calculate the score function*
  $\vec{\mathbf{r}}_l^{\text{denoised}}, \mathbf{f}_i^{\text{seq_logits}} = \text{MainTrunk}($
    $\{f^*\}, \vec{\mathbf{r}}_l^{\text{noisy}}, \hat{t}, t_p)$
  $\vec{\delta}_l = (\vec{\mathbf{r}}_l^{\text{noisy}} - \vec{\mathbf{r}}_l^{\text{denoised}})/\hat{t}$
  $dt = c_{\tau-1} - \hat{t}$
  $\vec{\mathbf{r}}_l \leftarrow \vec{\mathbf{r}}_l^{\text{noisy}} + \eta \cdot dt \cdot \vec{\delta}_l$
end for
return $\{\vec{\mathbf{r}}_l\}$, $\{\mathbf{f}_i^{\text{seq_logits}}\}$

The total loss can be written as:

$$\mathcal{L} = \mathcal{L}_0 + \alpha_1 \cdot \mathcal{L}_{\text{smooth_lddt}} + \alpha_2 \cdot \mathcal{L}_{\text{dist_res}} + \alpha_3 \cdot \mathcal{L}_{\text{dist_atom}} + \alpha_4 \cdot \mathcal{L}_{\text{med}} \tag{7}$$

3.5. Sampling

The sampling process is described in Algorithm 1. The initial atoms are sampled from a Gaussian distribution on $\mathbb{R}^{L\times 3}$. We only use the first-order Euler method as the ODE solver with T steps for discretization. Optionally, additional noise can be injected during the sampling steps to introduce stochasticity into the ODE solving process. We focus only on the sequence distribution decoded by the network in the final sampling step, employing a low-temperature softmax strategy to derive an approximate discrete one-hot amino acid sequence as the final sequence.

Table 1. Comparison of various methods. Protpardelle utilized ProteinMPNN as an auxiliary tool in all-atom proteins generation, resulting in identical results for CO-DESIGN 1 and PMPNN 1. In the case of Multiflow, which can only generate protein backbones and sequences without side chains, the reported DES-aa metric is based on C_αRMSD rather than aaRMSD.

Method	**CO-DESIGN 1**			**PMPNN 1**		
	DES-aa (↑)	**DIV-str/seq (↑)**	**NOV-str (↓)**	**DES-bb (w/wo) (↑)**	**DIV-str (↑)**	**NOV-str (↓)**
Protpardelle*	30.00%	15 / 26	0.747	30.00% (80.23%)	15	0.747
ProteinGenerator	43.14%	83 / 706	0.791	**93.14% (96.34%)**	151	0.785
Multiflow*	62.74%	134 / 1042	0.753	84.69% (95.43%)	184	0.744
RFdiffusion	N/A	N/A	N/A	78.29% (84.40%)	165	0.805
Pallatom	**85.03%**	**291 / 1466**	**0.719**	89.89% (94.46%)	**318**	**0.716**

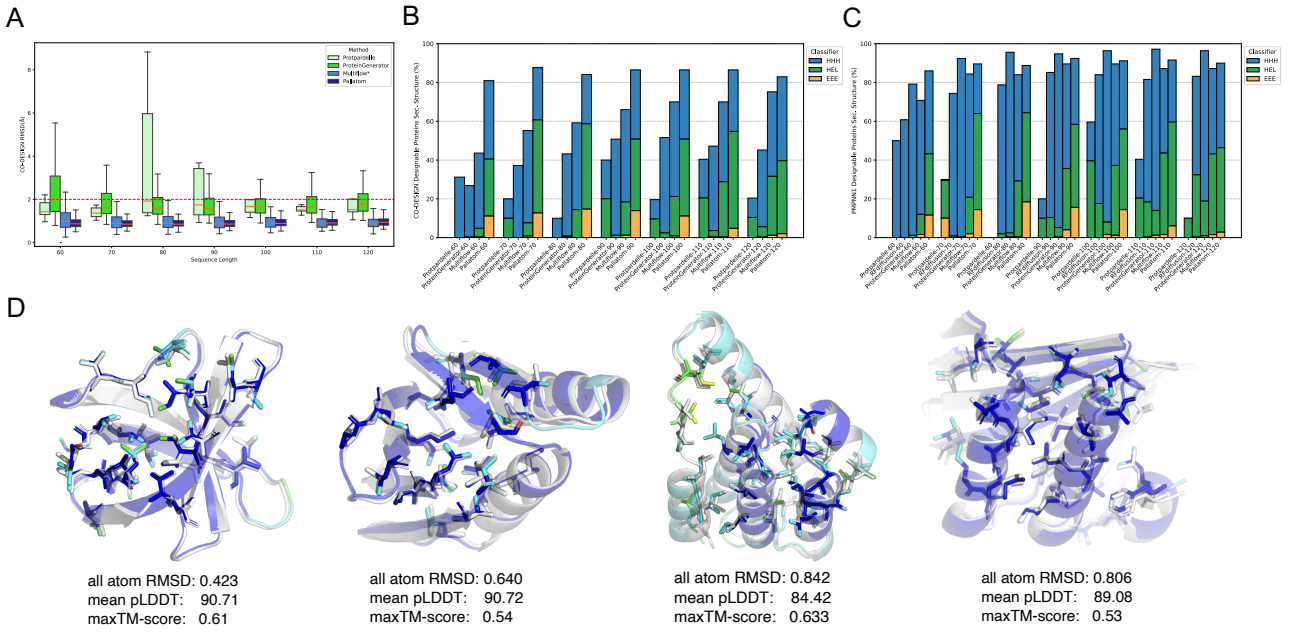

Figure 2. Evaluation of proteins sampled from Pallatom. (A) Boxplot of aaRMSD for proteins sampled by various methods under the CO-DESIGN 1 mode. Multiflow exhibits the C_αRMSD. (B, C) The proportions of secondary structures in designable proteins across different lengths are presented for CO-DESIGN 1 and PMPNN 1 modes across various methods, with the total height of the y-axis representing the designability. (D) Examples of high-quality, novel all-atom proteins sampled by Pallatom.

4. Experiments

4.1. Training setting

The training dataset of the model includes the PDB (Zardecki et al., 2022) and AlphaFold Database (AFDB) (Varadi et al., 2021). We performed rigorous data cleaning on augmented data from AFDB to obtain high-quality results. Detailed descriptions can be found in the Appendix B. We focus on small monomer proteins that can be easily synthesized using commercial oligo-pool method and the models are trained on crops of lengths up to 128. The model training utilized the Adam optimizer (Kingma & Ba, 2017) with a learning rate of 1e-3, β_1 = 0.9, β_2 = 0.999, and a batch size of 32. Details are provided in the Appendix Table 5.

4.2. Metrics

While some evaluation criteria used for protein backbone generation are not suited for the new task, we propose new metrics specifically designed for assessing all-atom protein generation.

The first criterion is structure designability. The traditional self-consistency process assesses the designability of protein backbones (**DES-bb**). This involves using a fixed-backbone design model (e.g., ProteinMPNN (Dauparas et al., 2022))

to generate N_{seq} sequences for the backbone, which are then folded by structure prediction models like ESMfold (Lin et al., 2023). The backbone's designability is evaluated by the optimal TM-score or C_α-RMSD between the folded and original backbones. However, this metric is not suitable for evaluating all-atom proteins, which include side-chain atoms. Therefore, we similarly define the designability of all-atom protein generation, denoted as **DES-aa**. For the all-atom proteins, the sequence is used to predict the structure, and the sample is considered designable if the mean pLDDT of the predicted structure exceeds 80 and the all-atom RMSD (aaRMSD) is less than 2 Å. This metric ensures high atomic-level accuracy and provides strong confidence in the structural integrity and designability of the predicted protein.

The second criterion is structure diversity, denoted as **DIV-str**. This can be quantified by calculating the clusters number of the designable structures using Foldseek (Van Kempen et al., 2024). For all-atom proteins, we use a similar diversity evaluation method for the generated sequences, denoted as **DIV-seq**. Specifically, we use MMseq2 (Steinegger & Söding, 2017) to calculate the clusters number of the designable sequences.

The last criterion is structure novelty, which evaluates the structural similarity between the generated backbones and natural proteins in the PDB, denoted as **NOV-str**. This is calculated by the TM-score of the generated designable backbones compared to the most similar proteins in the PDB.

In our evaluation, we use the following two modes:

- CO-DESIGN 1: For methods that can predict both all-atom coordinates and sequences, **DES-aa** is used. Diversity and novelty are evaluated based on the all-atom designable proteins.
- PMPNN 1: Other methods use **DES-bb** with $N_{seq} = 1$. Specifically, we calculate and display the two **DES-bb (w/wo)** under conditions with and without the pLDDT > 80 constraint. Diversity and novelty are evaluated based on the proteins filtered by **DES-bb (w)**.

4.3. Results

We sample Pallatom with 200 time steps using a noise scale $\gamma_0 = 0.2$, a step scale $\eta = 2.25$ and evaluate 250 proteins sampled for each length $L = 60, 70, 80, 90, 100, 110, 120$. Our primary comparisons are with state-of-the-art methods capable of generating all-atom proteins, such as Protpardelle and ProteinGenerator (Lisanza et al., 2023). For backbone generation, we compare with RFdiffusion. We also compared Multiflow, which is capable of simultaneously generating both backbones and sequences. All methods are evaluated using their open-source code and default parameters.

As shown in Table 1 and Figure 2A, Pallatom demonstrates superior performance in the CO-DESIGN 1 benchmark for all-atom protein generation. Remarkably, despite lacking specific training on fixed-backbone design tasks, Pallatom achieves comparable results to ProteinMPNN by predicting sequences with a single linear layer derived from aggregated atomic embedding. Comprehensive analysis of sequence quality between our design and ProteinMPNN, including detailed performance metrics, is presented in the Appendix G.3. These results substantiate Pallatom's capacity for all-atom structure design, validating our hypothesis that modeling $P(\textit{all-atom})$ effectively captures the fundamental relationship between protein structure and sequence. Notably, Pallatom generates all-atom protein structures with substantially enhanced structural diversity, outperforming existing methods by achieving significantly higher diversity than both Multiflow and ProteinGenerator, while simultaneously attaining the highest sequence diversity across all evaluated approaches.

In the PMPNN 1 benchmark, we observe that the DES-bb(wo) metric systematically overestimates designability relative to the pLDDT-constrained DES-bb(w) metric, as demonstrated by substantial discrepancies in Protpardelle's designability evaluations, highlighting the importance of quality constraints in designability assessment. The backbone generative models like ProteinGenerator, Multiflow and RFdiffusion demonstrate comparable performance. However, Protpardelle demonstrates suboptimal performance across all evaluation metrics, underscoring the critical importance of both network architecture design and atomic representation strategies in developing effective generative models for protein design. Consistent with its CO-DESIGN 1 performance, Pallatom demonstrates robust designability while simultaneously generating structurally diverse and novel protein conformations, effectively balancing these traditionally competing objectives. A detailed analysis in Figure 2BC shows Pallatom's stable performance across protein lengths and balanced secondary structure distributions, in contrast to the preference of comparative methods for specific secondary structures like α-helical ones. The case studies in Figure 2D highlight Pallatom's ability to design highly ordered proteins with hydrophobic cores and hydrophilic surfaces, consistent with high pLDDT structures predicted by ESMFold, underscoring its understanding of protein folding principles.

To assess generalization capabilities, we extensively evaluated Pallatom's performance on out-of-distribution (OOD) protein lengths, conducting systematic sampling and analysis across a broad range of 150-400 residues. The results demonstrate that Pallatom exhibits exceptional scalability,

achieving the highest designability even at over twice the maximum training length ($L = 128$). We also analyze secondary structure preferences in generated proteins across different methods, particularly for longer sequences, with comprehensive distribution statistics and detailed comparisons provided in the Appendix G.1. We also performed a statistical analysis of sampling duration, which demonstrates the efficiency of Pallatom. The results are presented in Appendix G.4.

Table 2. Pallatom sample metrics.

Noise Level γ_0	0.2	0.2	0.1	0.2	0.2
Step Scale η	1.75	2.25	2.25	2.75	3.25
$N_{\textbf{steps}}$	200	200	200	200	200
DES-aa (↑)	57%	87%	89%	**94%**	93%
DIV-str (↑)	56	**64**	52	55	46
HHH(%)	28%	28%	45%	35%	38%
HEL(%)	56%	54%	44%	47%	51%
EEE(%)	16%	18%	11%	18%	11%

4.4. Hyperparameter

We analyzed the impact of the sampling parameters and Table 2 presents the metrics for Pallatom when sampling 250 proteins with $L = 100$. We observed that, under the same noise scale, increasing the step scale η leads to a corresponding rise in designability, as well as an increase in the proportion of all-helix structures within the secondary structure. However, this improvement in designability comes at the cost of reduced structural diversity, indicating a trade-off between these two metrics. We further demonstrated the impact of varying step scales on the sampling of OOD-length proteins, with detailed results provided in the Appendix G.2. Furthermore, we found that reducing the additional noise level to $\gamma = 0.1$ slightly enhances designability but also significantly decreases structural diversity, consistent with the findings of previous work (Yim et al., 2023b).

Table 3. Ablation Studies evaluating the key concept of Pallatom to designability.

Method	DES-aa	DES-bb(w)	DES-bb(wo)
atom14	**87%**	**95%**	**95%**
hybrid14	5%	68%	82%
woRC	21%	93%	95%

4.5. Ablation Studies

Building upon the hypothesis that full-atom coordinates intrinsically encode essential protein information, we systematically investigated the key factors underlying Pallatom's superior designability performance. Table 3 presents the results, with the first row (atom14) serving as the baseline. We first conducted an ablation study on the `atom14` representation. For comparative analysis, we implemented the `hybrid14` representation and retrained the model. Similar to Protpardelle's methodology, the `hybrid14` representation maintains a superposition of 20 possible side-chain configurations, which collapse into a single hybridized residue state based on predicted sequence probabilities during the generation process. However, as shown in the second row, the sequence-guided full-atom approach not only failed to improve performance but actually resulted in reduced sequence quality (lower DES-aa) and compromised backbone generation (decreased DES-bb), revealing fundamental limitations in this design strategy. We hypothesize that this phenomenon stems from the absence of a well-defined mapping between noisy coordinate space and sequence space during diffusion. The sequence predictions derived from noisy structural intermediates exhibit substantial deviations from native foldable sequences, and these discrepancies are amplified throughout the reverse diffusion process, ultimately hindering the generation of biologically plausible sequences.

We next assessed the impact of the differentiable recycling process (woRC, shown in the third row). Our evaluation indicated that while the recycling process had little effect on backbone quality (DES-bb remained almost unchanged), it significantly impacted sequence generation, leading to a substantial reduction in DES-aa. These results highlight the crucial role of the recycling process in the components and its differential influence on various aspects of protein design. The detailed setup of the ablation studies can be found in the Appendix E.

5. Discussion

We introduce Pallatom, a highly efficient end-to-end all-atom protein generation framework that simultaneously captures the relationship between sequence and structure, enabling state-of-the-art performance. Our proposed all-atom protein `atom14` representation eliminates the constraints of explicitly defining all amino acid types during generation, offering a more precise approach for representing coordinates in all-atom systems. We have redesigned a fundamental unit capable of fusing, recycling and updating diverse features of all-atom proteins while predicting their coordinates. The new framework efficiently adapts to all-atom protein structure diffusion generation. The results demonstrate that models learning $P(\textit{all-atom})$ exhibit strong performance and diversity in *de novo* protein generation, unlocking new pathways for protein design. Future work includes developing a more generalized model architecture and extending our framework to enable the design of complex systems.

Code Availibility

The Pallatom is available on GitHub (https://github.com/levinthal/Pallatom).

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., Bodenstein, S. W., Evans, D. A., Hung, C.-C., O'Neill, M., Reiman, D., Tunyasuvunakool, K., Wu, Z., Žemgulytė, A., Arvaniti, E., Beattie, C., Bertolli, O., Bridgland, A., Cherepanov, A., Congreve, M., Cowen-Rivers, A. I., Cowie, A., Figurnov, M., Fuchs, F. B., Gladman, H., Jain, R., Khan, Y. A., Low, C. M. R., Perlin, K., Potapenko, A., Savy, P., Singh, S., Stecula, A., Thillaisundaram, A., Tong, C., Yakneen, S., Zhong, E. D., Zielinski, M., Žídek, A., Bapst, V., Kohli, P., Jaderberg, M., Hassabis, D., and Jumper, J. M. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, June 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-07487-w.

Anishchenko, I., Pellock, S. J., Chidyausiku, T. M., Ramelot, T. A., Ovchinnikov, S., Hao, J., Bafna, K., Norn, C., Kang, A., Bera, A. K., DiMaio, F., Carter, L., Chow, C. M., Montelione, G. T., and Baker, D. De novo protein design by deep network hallucination. *Nature*, 600(7889): 547–552, December 2021. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-021-04184-w.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., Millán, C., Park, H., Adams, C., Glassman, C. R., DeGiovanni, A., Pereira, J. H., Rodrigues, A. V., Van Dijk, A. A., Ebrecht, A. C., Opperman, D. J., Sagmeister, T., Buhlheller, C., Pavkov-Keller, T., Rathinaswamy, M. K., Dalwadi, U., Yip, C. K., Burke, J. E., Garcia, K. C., Grishin, N. V., Adams, P. D., Read, R. J., and Baker, D. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, August 2021. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.abj8754.

Baek, M., Anishchenko, I., Humphreys, I. R., Cong, Q., Baker, D., and DiMaio, F. Efficient and accurate prediction of protein structure using RoseTTAFold2, May 2023.

Bao, F., Nie, S., Xue, K., Cao, Y., Li, C., Su, H., and Zhu, J. All are Worth Words: A ViT Backbone for Diffusion Models, March 2023.

Campbell, A., Yim, J., Barzilay, R., Rainforth, T., and Jaakkola, T. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=kQwSbv0BR4.

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

Chu, A. E., Kim, J., Cheng, L., El Nesr, G., Xu, M., Shuai, R. W., and Huang, P.-S. An all-atom protein generative model. *Proceedings of the National Academy of Sciences*, 121(27):e2311500121, July 2024. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2311500121.

Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Courbet, A., De Haas, R. J., Bethel, N., Leung, P. J. Y., Huddy, T. F., Pellock, S., Tischer, D., Chan, F., Koepnick, B., Nguyen, H., Kang, A., Sankaran, B., Bera, A. K., King, N. P., and Baker, D. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, October 2022. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.add2187.

De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. Riemannian Score-Based Generative Modelling, November 2022.

Hsu, C., Verkuil, R., Liu, J., Lin, Z., Hie, B., Sercu, T., Lerer, A., and Rives, A. Learning inverse folding from millions of predicted structures, April 2022.

Huguet, G., Vuckovic, J., Fatras, K., Thibodeau-Laufer, E., Lemos, P., Islam, R., Liu, C.-H., Rector-Brooks, J., Akhound-Sadegh, T., Bronstein, M., Tong, A., and Bose, A. J. Sequence-Augmented SE(3)-Flow Matching For Conditional Protein Backbone Generation, May 2024.

Ingraham, J. B., Baranov, M., Costello, Z., Barber, K. W., Wang, W., Ismail, A., Frappier, V., Lord, D. M., Ng-Thow-Hing, C., Van Vlack, E. R., Tie, S., Xue, V., Cowles, S. C., Leung, A., Rodrigues, J. V., Morales-Perez, C. L., Ayoub, A. M., Green, R., Puentes, K., Oplinger, F., Panwar, N. V., Obermeyer, F., Root, A. R., Beam, A. L., Poelwijk, F. J., and Grigoryan, G. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, November 2023. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-023-06728-8.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L., and Dror, R. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2020.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek,

A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-021-03819-2.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization, January 2017.

Krishna, R., Wang, J., Ahern, W., Sturmfels, P., Venkatesh, P., Kalvet, I., Lee, G. R., Morey-Burrows, F. S., Anishchenko, I., Humphreys, I. R., et al. Generalized biomolecular modeling and design with RoseTTAFold All-Atom. *Science*, 384(6693):eadl2528, 2024.

Lin, Y., Lee, M., Zhang, Z., and AlQuraishi, M. Out of Many, One: Designing and Scaffolding Proteins at the Scale of the Structural Universe with Genie 2, 2024.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., Dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.ade2574.

Lisanza, S. L., Gershon, J. M., Tipps, S., Arnoldt, L., Hendel, S., Sims, J. N., Li, X., and Baker, D. Joint Generation of Protein Sequence and Structure with RoseTTAFold Sequence Space Diffusion, May 2023.

Ren, M., Zhu, T., and Zhang, H. CarbonNovo: Joint Design of Protein Structure and Sequence Using a Unified Energy-based Model. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=FSxTEvuFa7.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118 (15):e2016239118, April 2021. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2016239118.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-Based Generative Modeling through Stochastic Differential Equations, February 2021.

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

Touw, W. G., Baakman, C., Black, J., Te Beek, T. A., Krieger, E., Joosten, R. P., and Vriend, G. A series of PDB-related databanks for everyday needs. *Nucleic acids research*, 43(D1):D364–D368, 2015.

Van Kempen, M., Kim, S. S., Tumescheit, C., Mirdita, M., Lee, J., Gilchrist, C. L., Söding, J., and Steinegger, M. Fast and accurate protein structure search with Foldseek. *Nature biotechnology*, 42(2):243–246, 2024.

Varadi, M., Anyango, S., Deshpande, M., Nair, S., Natassia, C., Yordanova, G., Yuan, D., Stroe, O., Wood, G., Laydon, A., Žídek, A., Green, T., Tunyasuvunakool, K., Petersen, S., Jumper, J., Clancy, E., Green, R., Vora, A., Lutfi, M., Figurnov, M., Cowie, A., Hobbs, N., Kohli, P., Kleywegt, G., Birney, E., Hassabis, D., and Velankar, S. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50(D1):D439–D444, 11 2021. ISSN 0305-1048. doi: 10.1093/nar/gkab1061. URL https://doi.org/10.1093/nar/gkab1061.

Wang, C., Qu, Y., Peng, Z., Wang, Y., Zhu, H., Chen, D., and Cao, L. Proteus: Exploring Protein Structure Generation for Enhanced Designability and Efficiency, February 2024.

Wang, G. and Dunbrack Jr, R. L. PISCES: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591, 2003.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Torres, S. V., Lauko, A., De Bortoli, V., Mathieu, E., Ovchinnikov, S., Barzilay, R., Jaakkola, T. S., DiMaio, F., Baek, M., and Baker, D. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, August 2023. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-023-06415-8.

Yim, J., Campbell, A., Foong, A. Y. K., Gastegger, M., Jiménez-Luna, J., Lewis, S., Satorras, V. G., Veeling, B. S., Barzilay, R., Jaakkola, T., and Noé, F. Fast protein backbone generation with SE(3) flow matching, October 2023a.

Yim, J., Trippe, B. L., De Bortoli, V., Mathieu, E., Doucet, A., Barzilay, R., and Jaakkola, T. SE(3) diffusion model with application to protein backbone generation, May 2023b.

Zardecki, C., Dutta, S., Goodsell, D. S., Lowe, R., Voigt, M., and Burley, S. K. PDB-101: Educational resources supporting molecular explorations through biology and medicine. *Protein Science*, 31(1):129–140, 2022.

A. Related work

Backbone Generation. Diffusion models based on SE(3)-equivariant network architectures and protein representations using rigid frames have achieved significant success in protein generation, as evidenced by models like Chroma (Ingraham et al., 2023), Genie2 (Lin et al., 2024), RFdiffusion (Watson et al., 2023), FrameDiff (Yim et al., 2023b), FrameFlow (Yim et al., 2023a), Proteus (Wang et al., 2024), and FoldFlow2 (Huguet et al., 2024). These models now support multi-condition controlled generation and have been extensively validated through both in-silico and wet-lab experiments.

Codesign Models. Recent methods have explored a co-design approach that simultaneously designs both the backbone and sequence. Multiflow (Campbell et al., 2024) utilizes a diffusion process that jointly operates on discrete sequences and continuous SE(3) backbones, eliminating the need for sequence redesign with ProteinMPNN (Dauparas et al., 2022). CarbonNovo (Ren et al., 2024) employs a similar approach, using SE(3) diffusion on the protein backbone while simultaneously designing a sequence at each step with the MRF decoder. The sequence information is then embeded using the protein language model ESM2-3B (Rives et al., 2021) to guide structural generation.

All-Atom Generation. Recent research teams have begun exploring fully all-atom generative representation systems. For instance, Protpardelle (Chu et al., 2024) employs a coordinates diffusion model to support the generation of all-atom protein structures. ProteinGenerator (Lisanza et al., 2023) applies Euclidean diffusion on one-hot encoded sequences, combined with a structure prediction module to obtain all-atom structure. RFdiffusionAA (Krishna et al., 2024), based on fine-tuning the RoseTTAFold2 (Baek et al., 2023), can produce backbone structures of proteins and small molecule complexes but lacks side-chain conformations for standard amino acids. We focus on the methods directly generate all-atom protein structures, the most relevant work is Protpardelle, which uses an end-to-end approach to generate all-atom structures.

B. Training Datasets

B.1. PDB Data

We used PISCES (Wang & Dunbrack Jr, 2003) to obtain the required PDB list. For training, we selected a subset of PDB entries with a resolution of $<$3Å and a 95% sequence identity threshold. We then performed standard filtering to remove any proteins with $>$50% loops and applied a series of folding quality filters (described below). This process resulted in 7,459 structures.

B.2. Augmented Data

Augmented data are widely used in protein modeling-related work. Consequently, we supplemented our dataset with the AlphaFold Database (AFDB) (Varadi et al., 2021). The AF2 predicted structure database is available under a CC-BY-4.0 license for both academic and commercial uses. The AFDB contains a total of 214 million data points. By applying an average pLDDT threshold of $>$80 and limiting the sequence length to a maximum of 128, we obtained 582,652 structures. In addition, we employed more sophisticated filtering strategies to acquire designable and high-quality data, assessing the average neighboring atomic density for each residue to determine whether they are core residues. We found that with this filter, core residue content $>$30% allows us to identify tightly packed structures. Furthermore, the number and distribution of secondary structures can define the compactness and diversity of folding. We used DSSP (Touw et al., 2015) to assign the protein secondary structures and removing structures with loop content $>$50%. We also apply filter on the total number of continuous segments of β-sheet and α-helix to maintain structural diversity. For highly extended structures, we limited the radius of gyration (R_g) to less than 25.0. To avoid overly long continuous unstructured regions within the protein structures, we restricted the maximum length of each loop to 15. Finally, we used the FoldSeek easycluster algorithm to remove redundant structures, setting a TM score threshold of 0.8 and a coverage of 0.9, which removed approximately 30% of highly similar structures. After applying these stringent filters, only 27,697 protein structures remained. These structures typically exhibit good folding and high designability.

C. Algorithms

We present a comprehensive description of Pallatom's architectural modules and algorithmic workflow. In the accompanying pseudocode, components highlighted in blue maintain substantial similarity to AlphaFold3's implementation and are not expanded to prevent redundancy while ensuring clarity.

C.1. MainTrunk

Algorithm 2 details the denoising process of the Pallatom `MainTrunk` network.

Algorithm 2 MainTrunk

def MainTrunk(f^*, $\vec{\mathbf{r}}_l^{input}$, $\hat{t}$, t, c_{atom}=128, c_{pair}=128, c_{res}=256, $c_{atompair}$=16, σ_{data}=16):

 # Initialize positions and conditions embedding

1: $\vec{\mathbf{r}}_l^{\text{scaled}} = \vec{\mathbf{r}}_l^{\text{intput}} / \sqrt{\sigma_{\text{data}}^2 + \hat{t}^2}$ $\qquad \vec{\mathbf{r}}_l^{\text{scaled}} \in \mathbb{R}^3$

2: $\mathbf{s}_i^{init} = \text{LinearNoBias}((\mathbf{f}_i^{\text{residue_idx}}))$ $\qquad \mathbf{s}_i^{init} \in \mathbb{R}^{c_{res}}$

3: $\mathbf{t}_i = \text{TimeFourierEmbedding}(\frac{1}{4} log(\hat{t}/\sigma_{data}))$ $\qquad \mathbf{t}_i \in \mathbb{R}^{c_{res}}$

4: $\mathbf{s}_i^{init} \mathrel{+}= \mathbf{t}_i$

5: $\mathbf{z}_{ij}^{init} = \text{RelativePositionEncoding}(f^*)$ $\qquad \mathbf{z}_{ij}^{init} \in \mathbb{R}^{c_{pair}}$

6: $\mathbf{z}_{ij} = \mathbf{z}_{ij}^{init} + \text{TemplateEmbedder}(\{f^*\}, \mathbf{z}_{ij}^{init}, t, N_{block} = 2, c = 64, d = c_{pair})$

 # Initialise single and atom embeddings

7: $\{\mathbf{s}_i\}, \{\mathbf{q}_l^{skip}\}, \{\mathbf{c}_l^{skip}\}, \{\mathbf{p}_{lm}^{skip}\}, \{\mathbf{c}_l\} = \text{AtomFeatureEncoder}(\{f^*\}, \mathbf{s}_i^{init}, \mathbf{z}_{ij}, \vec{\mathbf{r}}_l^{\text{scaled}}, c_{res}, c_{atompair}, c_{atom})$

8: $\mathbf{s}_i \mathrel{+}= \text{LinearNoBias}(\text{LayerNorm}(\mathbf{s}_i^{init}))$

 # AtomDecoder Units

9: $\vec{\mathbf{r}}_l^{\text{updates}} = \mathbf{0}$ $\qquad \vec{\mathbf{r}}_l^{\text{updates}} \in \mathbb{R}^3$

10: **for all** $unit \in \{1, \ldots, K_{\text{unit}}\}$ **do**

11: $\quad \mathbf{s}_i = \text{NodeUpdate}(\mathbf{s}_i, \mathbf{t}_i, \mathbf{z}_{ij}, c = c_{res})$

12: $\quad \{\mathbf{q}_l^{updated}\}, \{\vec{\mathbf{r}}_l^{\text{update}}\}, \{\mathbf{c}_l\} = \text{AtomAttentionDecoder}(\mathbf{q}_l^{skip}, \mathbf{p}_{lm}^{skip}, \mathbf{c}_l^{skip}, \mathbf{c}_l, \mathbf{s}_i, \mathbf{z}_{ij})$

13: $\quad \vec{\mathbf{r}}_l^{\text{updates}} \mathrel{+}= \vec{\mathbf{r}}_l^{\text{update}}$

14: $\quad \vec{\mathbf{r}}_l^{\text{denoised}} = \sigma_{\text{data}}^2 / (\sigma_{\text{data}}^2 + \hat{t}^2) \cdot \vec{\mathbf{r}}_l^{\text{intput}} + \sigma_{\text{data}} \cdot \hat{t} / \sqrt{\sigma_{\text{data}}^2 + \hat{t}^2} \cdot \vec{\mathbf{r}}_l^{\text{updates}}$

15: $\quad \vec{\mathbf{r}}_i^{center} = \vec{\mathbf{r}}_l^{\text{denoised}}[center_uid]$

16: $\quad \mathbf{z}_{ij} = \text{PairUpdate}(\mathbf{z}_{ij}, \vec{\mathbf{r}}_i^{center}, c = c_{pair})$

17: **end for**

 # SeqHead for amino acid decoding

18: $\mathbf{a}_i = \underset{\substack{l \in \{1, \ldots, N_{\text{atoms}}\} \\ \text{tok_idx}(l) = i}}{\text{mean}} \left(\text{ReLU}(\text{LinearNoBias}(\{\mathbf{q}_l^{updated}\})) \right)$ $\qquad \mathbf{a}_i \in \mathbb{R}^{c_{res}}$

19: $\mathbf{f}_i^{\text{seq_logits}} = \text{LinearNoBias}(\mathbf{a}_i)$ $\qquad \mathbf{f}_i^{\text{seq_logits}} \in \mathbb{R}^{20}$

return $\{\vec{\mathbf{r}}_l^{\text{denoised}}\}, \{\mathbf{f}_i^{\text{seq_logits}}\}$

C.2. TemplateEmbedder

In the TemplateEmbedder module, we retained only the necessary mask and distogram features, and additionally included the timestep feature for self-conditioning.

C.3. AtomFeatureEncoder

The AtomFeatureEncoder is adapted from AF3, with modifications for all-atom generation. We exclusively utilize alanine as the reference conformer, eliminating the need for predefined amino acid sequence information during the initialization phase. Furthermore, we standardized residue representations using the `atom14` format, with virtual atoms to prevent information leakage. The initialization process for traversing atomic embeddings is detailed as follows:

Algorithm 3 Template Embedder

def TemplateEmbedder(f^*, $\mathbf{z}_{ij}$, t, N_{block}=2, c=64, d=128):

 # Concat template features

1: $b_{ij}^{\text{template_pseudo_beta_mask}} = f_i^{\text{template_pseudo_beta_mask}} \cdot f_j^{\text{template_pseudo_beta_mask}}$

2: $b_{ij}^{\text{time}} = t \odot f_{ij}^{\text{template_pseudo_beta_mask}}$ $\qquad t \sim [0, 1)$

3: $\mathbf{a}_{ij} = \text{concat}(f_{ij}^{\text{template_distogram}}, b_{ij}^{\text{template_pseudo_beta_mask}}, b_{ij}^{\text{time}})$

 # Embed the self-condition feature

4: $\mathbf{v}_{ij} = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{ij})) + \text{LinearNoBias}(\mathbf{a}_{ij})$ $\qquad \mathbf{v}_{ij} \in \mathbb{R}^c$

5: $\mathbf{v}_{ij} = \text{PairformerStack}(\mathbf{v}_{ij}, N_{\text{block}})$

6: $\mathbf{u}_{ij} \leftarrow \text{LinearNoBias}(\text{ReLU}(\text{LayerNorm}(\mathbf{v}_{ij})))$ $\qquad \mathbf{u}_{ij} \in \mathbb{R}^d$

return $\{\mathbf{u}_{ij}\}$

Algorithm 4 AtomFeatureEncoder

def AtomFeatureEncoder(f^*, $\mathbf{s}_i^{input}$, $\mathbf{z}_{ij}^{input}$, $\vec{\mathbf{r}}_l^{scaled}$, c=256, d=16, m=128):

 # Create the atom single conditioning

1: $\mathbf{f}^{ref} = \text{concat}(\vec{\mathbf{f}}^{\text{ref_pos}}, \mathbf{f}^{\text{ref_element}})$ $\qquad \mathbf{f}^{ref} \in \mathbb{R}^{14\times 7}$

2: $\mathbf{c}_l = \text{LinearNoBias}(\text{tile}(\mathbf{f}^{ref}))$ $\qquad \mathbf{c}_l \in \mathbb{R}^m$

3: $\vec{\mathbf{f}}_l^{\text{ref_pos}} = \text{tile}(\vec{\mathbf{f}}^{\text{ref_pos}})$ $\qquad \vec{\mathbf{f}}_l^{\text{ref_pos}} \in \mathbb{R}^3$

4: $\mathbf{c}_l^{skip} = \mathbf{c}_l$

 # Embed offsets between atom reference positions

5: $\vec{\mathbf{d}}_{lm} = \vec{\mathbf{f}}_l^{\text{ref_pos}} - \vec{\mathbf{f}}_m^{\text{ref_pos}}$ $\qquad \vec{\mathbf{d}}_{lm} \in \mathbb{R}^3$

6: $v_{lm} = (\mathbf{f}_l^{\text{ref_space_uid}} == \mathbf{f}_m^{\text{ref_space_uid}})$ $\qquad v_{lm} \in \mathbb{R}$

7: $\mathbf{p}_{lm} = \text{LinearNoBias}(\vec{\mathbf{d}}_{lm}) \cdot v_{lm}$ $\qquad \mathbf{p}_{lm} \in \mathbb{R}^d$

8: $\mathbf{p}_{lm} \mathrel{+}= \text{LinearNoBias}\left(1/(1 + \|\vec{\mathbf{d}}_{lm}\|^2)\right) \cdot v_{lm}$

9: $\mathbf{p}_{lm} \mathrel{+}= \text{LinearNoBias}(v_{lm}) \cdot v_{lm}$

10: $\mathbf{p}_{lm} \mathrel{+}= \text{LinearNoBias}(\text{ReLU}(\mathbf{c}_l)) + \text{LinearNoBias}(\text{ReLU}(\mathbf{c}_m))$

11: $\mathbf{p}_{lm}^{skip} = \mathbf{p}_{lm}$

 # Initialise the atom single representation as the single conditioning

12: $\mathbf{q}_l^{skip} = \mathbf{c}_l + \text{LinearNoBias}(\vec{\mathbf{r}}_l^{scaled})$ $\qquad \mathbf{q}_l \in \mathbb{R}^m$

 # Add atom positional and time conditioning

13: $\mathbf{c}_l \mathrel{+}= \text{LinearNoBias}(\text{LayerNorm}(\mathbf{s}_{\text{tok_idx}(l)}^{init}))$

14: $\mathbf{p}_{lm} \mathrel{+}= \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{\text{tok_idx}(l)\text{tok_idx}(m)}^{init})$

15: $\mathbf{p}_{lm} \mathrel{+}= \text{LinearNoBias}(\text{ReLU}(\text{LinearNoBias}(\text{ReLU}(\text{LinearNoBias}(\text{ReLU}(\mathbf{p}_{lm}))))))$

 # Cross attention transformer

16: $\mathbf{q}_l^{skip} = \text{AtomTransformer}(\mathbf{q}_l^{skip}, \mathbf{c}_l, \mathbf{p}_{lm}, N_{\text{block}} = 3, N_{\text{head}} = 4)$

 # Aggregate per-atom representation to per-residue representation

17: $\mathbf{s}_i = \underset{\substack{l \in \{1,\ldots,N_{\text{atoms}}\} \\ \text{tok_idx}(l)=i}}{\text{mean}} \left(\text{ReLU}(\text{LinearNoBias}(\mathbf{q}_l^{skip}))\right)$ $\qquad \mathbf{s}_i \in \mathbb{R}^c$

return $\{\mathbf{s}_i\}$, $\{\mathbf{q}_l^{skip}\}$, $\{\mathbf{c}_l^{skip}\}$, $\{\mathbf{p}_{lm}^{skip}\}$, $\{\mathbf{c}_l\}$

C.4. AtomAttentionDecoder

We developed a novel information management strategy for decoder layers that simultaneously integrates updated single- and pair-embeddings into traversing atomic-level embeddings. This approach effectively prevents redundant structural information accumulation while maintaining essential feature propagation.

We present a mathematical derivation to analyze information redundancy arising from residual connections between residue-level and atomic-level embeddings, identifying key factors contributing to "traversing" strategies. For notational clarity, we define s_i as the residue-level embedding and q_i as the atom-level embedding for the i-th unit, with broadcast(s_i) denotes the operation that propagates residue-level information to atomic-level. The network updates are represented by the functions $f(\cdot)$ and $g(\cdot)$:

$$s_i = s_{i-1} + f(s_{i-1}) \tag{8}$$

$$q_i = q_{i-1} + g(\text{broadcast}(s_{i-1}) + q_{i-1}) \tag{9}$$

The iterative process can be organized as follows:

$$s_i = s_1 + \sum_{k=0}^{i-1} f(s_k) \tag{10}$$

$$q_i = q_1 + \sum_{k=1}^{i-1} g(\text{broadcast}(s_k) + q_k) \tag{11}$$

$$= q_1 + \sum_{k=1}^{i-1} g(\text{broadcast}(s_k) + (q_1 + \sum_{j=1}^{k-1} g(\text{broadcast}(s_j) + q_j))) \tag{12}$$

Through recursive expansion of the q_i, we observe that each term depends on its preceding q_k while incorporating broadcasted features from s_{i-1} at each update step. This results in information redundancy, ultimately impeding efficient network learning and feature updating.

Algorithm 5 AtomAttentionDecoder

def AtomAttentionDecoder($\mathbf{q}_l^{skip}$, $\mathbf{p}_{lm}^{skip}$, $\mathbf{c}_l^{skip}$, $\mathbf{c}_l$, $\mathbf{s}_i$, $\mathbf{z}_{ij}$):

 # Add trunk embeddings

1: $\mathbf{q}_l = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{s}_{\text{tok_idx}(l)})) + \mathbf{q}_l^{\text{skip}}$

2: $\mathbf{p}_{lm} = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{\text{tok_idx}(l)\text{tok_idx}(m)}) + \mathbf{p}_{lm}^{\text{skip}}$

3: $\mathbf{p}_{lm} \mathrel{+}= \text{LinearNoBias}(\text{ReLU}(\text{LinearNoBias}(\text{ReLU}(\text{LinearNoBias}(\text{ReLU}(\mathbf{p}_{lm}))))))$

 # Cross attention transformer

4: $\mathbf{q}_l = \text{AtomTransformer}(\mathbf{q}_l, \mathbf{c}_l, \mathbf{p}_{lm}, N_{\text{block}} = 3, N_{\text{head}} = 4)$

 # Map to positions update

5: $\vec{\mathbf{r}}_l^{\text{update}} = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{q}_l))$

 # Update trunk embeddings to atom condition

6: $\mathbf{c}_l = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{s}_{\text{tok_idx}(l)})) + \mathbf{c}_l^{\text{skip}}$

return $\{\vec{\mathbf{r}}_l^{\text{update}}\}$,$\{\mathbf{c}_l\}$

C.5. NodeUpdate

The NodeUpdate module manages residue-level embedding updates through a time-conditioned AttentionPairBias mechanism, enabling adaptive scaling of network updates based on noise levels during the diffusion process. Additionally, we implemented dropout regularization during training to prevent overfitting and improve model generalization capabilities.

C.6. PairUpdate

The PairUpdate module exclusively employs a modified TriangleAttention algorithm, excluding TriangleMultiplication while preserving robust generative capabilities. Our implementation utilizes recycling-derived pair representations as

Algorithm 6 NodeUpdate

def NodeUpdate($\mathbf{s}_i$, $\mathbf{t}_i$, $\mathbf{z}_{ij}$, c=256):

 # AttentionPairBias with updated pair bias

1: $\mathbf{s}_i$ += DropoutRowwise$_{0.25}$(AttentionPairBias($\mathbf{s}_i, \mathbf{t}_i, \mathbf{z}_{ij}, \beta_{ij} = 0, N_{head} = 8$))

2: $\mathbf{s}_i$ += Transition($\mathbf{s}_i$)

return $\{\mathbf{s}_i\}$

attention biases, enhancing the model's ability to capture long-range interactions. To ensure pair feature space consistency, we processed recycling structures using identical binning parameters as the TemplateEmbedder module, employing 39 discrete bins spanning from 3.25 to 50.75 angstroms for comprehensive distance representation.

Algorithm 7 PairUpdate

def PairUpdate($\mathbf{z}_{ij}$, $\mathbf{r}_i^{center}$, c=128):

 # Obtaining the pairwise distance matrix through RBF discretization

1: $\mathbf{d}_{ij} = \left\|\vec{\mathbf{r}}_i^{center} - \vec{\mathbf{r}}_j^{center}\right\|$ $\qquad \mathbf{d}_{ij} \in \mathbb{R}$

2: $\mathbf{b}_{ij}$ = LinearNoBias(Transform_RBF($\mathbf{d}_{ij}$)) $\qquad \mathbf{b}_{ij} \in \mathbb{R}^c$

 # TriangleAttention with coordinates pair bias

3: $\mathbf{z}_{ij}$ += DropoutRowwise$_{0.25}$(TriangleAttentionStartingNodeWithBias($\mathbf{z}_{ij}, \mathbf{b}_{ij}$))

4: $\mathbf{z}_{ij}$ += DropoutColumnwise$_{0.25}$(TriangleAttentionEndingNodeWithBias($\mathbf{z}_{ij}, \mathbf{b}_{ij}$))

5: $\mathbf{z}_{ij}$ += Transition($\mathbf{z}_{ij}$)

return $\{\mathbf{z}_{ij}\}$

D. Loss Function

In this paper, in addition to the primary loss function $\mathcal{L}_0$, which is composed of the denoising score matching loss $\mathcal{L}_{atom}$ and the cross-entropy loss $\mathcal{L}_{seq}$ for sequence decoding, we describe additional loss components that contribute to model optimization.

D.1. Distogram Loss

The distogram loss function has become a fundamental component in protein structure prediction pipelines, effectively guiding the estimation of inter-residue distance matrices from pairwise feature representations. In our implementation, we symmetrize the global residue-level pair representations and project them into 64 distance bins with predicted probabilities p_{ij}^b, which are then supervised using one-hot encoded target distance bins y_{ij}^b.

$$\mathcal{L}_{\text{dist_res}} = -\frac{1}{L^2} \sum_{i,j} \sum_{b=1}^{64} y_{ij}^b \log p_{ij}^b$$

Similarly, to supervise the local relative distance distribution, we project the atomic-level pair representation from local atomic attention into 22 distance bins from 0 $\mathring{A}$ to 10 $\mathring{A}$ with q_{nm}^b, and construct the loss function using one-hot encoded target distance bins, with the local region defined by the attention window within the the $14L \times 14L$ atomic-level map, ensuring precise local geometry constraints.

$$\mathcal{L}_{\text{dist_atom}} = -\frac{1}{NM} \sum_{n,m \in \text{local}} \sum_{b=1}^{22} y_{nm}^b \log q_{nm}^b$$

D.2. Intermediate Loss

To stabilize the training process, we implement intermediate supervision of sequence and structure predictions from each decoder unit. Additionally, we introduce a loss weight decay mechanism with $\gamma = 0.99$ across decoder blocks, progressively

increasing the weight assigned to later layers. Therefore, the intermediate loss for the K units can be expressed as:

$$\mathcal{L}_{\text{med}}^{k} = \lambda(\hat{t}) \cdot \frac{||\vec{\mathbf{r}}_{(k)}^{\text{denoised}} - \vec{\mathbf{r}}^{\text{aligned}}||^2}{3L} + \alpha_0 \cdot \text{CE}(\hat{\mathbf{a}}_{(k)}, \mathbf{a}^0),$$

$$\mathcal{L}_{\text{med}} = \frac{1}{K} \sum_{k=1}^{K} \gamma^{K-k} \cdot \mathcal{L}_{\text{med}}^{k}$$

D.3. Smooth LDDT loss function

During the training, we adopt the smooth LDDT loss (Algorithm 8) and implement a simplified version specifically for all-atom proteins.

Algorithm 8 Smooth LDDT loss

def SmoothLDDTLoss($\vec{\mathbf{r}}_l$, $\vec{\mathbf{r}}_l^{GT}$):

 # Compute distances between all pairs of atoms

1: $\delta r_{lm} \leftarrow ||\vec{\mathbf{r}}_l - \vec{\mathbf{r}}_m||$

2: $\delta r_{lm}^{GT} \leftarrow ||\vec{\mathbf{r}}_l^{GT} - \vec{\mathbf{r}}_m^{GT}||$

 # Compute distance difference

3: $\delta_{lm} = \text{abs}(\delta r_{lm}^{GT} - \delta r_{lm})$

4: $\epsilon_{lm} = \frac{1}{4}[\text{sigmoid}(\frac{1}{2} - \delta_{lm}) + \text{sigmoid}(1 - \delta_{lm}) + \text{sigmoid}(2 - \delta_{lm}) + \text{sigmoid}(4 - \delta_{lm})]$

 # Set the radius threshold and compute mean

5: $c_{lm} \leftarrow \mathbf{1}(\delta x_{lm}^{GT} < 15\mathring{A})$

5: $\text{lddt} = \underset{l \neq m}{\text{mean}}(c_{lm}\epsilon_{lm}) / \underset{l \neq m}{\text{mean}}(c_{lm})$

return 1 - lddt

E. Ablation Studies

We systematically investigate the fundamental factors contributing to Pallatom's superior co-design performance through complementary top-down and bottom-up analyses, aiming to uncover the underlying principles of its exceptional designability. The success of the `atom14` full-atom representation demonstrates the viability of comprehensive atomic-level generative modeling. To further explore this paradigm, we develop a `hybrid14` representation that incorporates sequence information, enabling comparative analysis that highlights the distinct advantages of the `atom14` approach. Another key innovations in the AtomDecoder unit is the differentiable recycling process, we further investigate through ablation studies.

E.1. `hybrid14`

To efficiently represent side-chain configurations across all 20 amino acids while avoiding the computational overhead of Protpardelle's `atom73` representation, we propose the `hybrid14` representation maintains a superposition of 20 possible side-chain configurations, which collapse into a single hybridized residue state based on predicted sequence probabilities during the generation process.

Specifically, we implement the model's self-conditioning mechanism by matrix-multiplying the predicted sequence probabilities $S \in \mathbb{R}^{L \times 20}$ with standard amino acid reference conformer coordinates $V \in \mathbb{R}^{20 \times 14 \times 3}$, generating hybrid reference conformations $V_{hybrid} = SV \in \mathbb{R}^{L \times 14 \times 3}$ that are subsequently integrated by atomic embedding.

Using identical loss functions and training configurations, we retrained the model and conducted ablation studies by generating 100 protein sequences of length $L = 100$, followed by comprehensive designability evaluation to assess the impact of atomic representation modifications.

E.2. woRC: Ablation Setup without the Recycling Process

To assess the contribution of the differentiable recycling process in the AtomDecoder unit, we conducted ablation experiments by substituting the recycling pair with a zero matrix and disabling gradient backpropagation. This modification prevents the residue-level pair embedding z_{ij} from receiving updates from partially denoised structures while maintaining the triangle update module for self-conditioning during training.

F. Experiment Details

Table 4 provides a detailed list of the input feature.

Table 4. Input Feature Descriptions

Input Feature	Dimension	Description
ref_pos	$(14, 3)$	Atom positions in the reference conformer are given in Å. The backbone atoms (N, C, C_α, O, C_β) are listed in the first five columns, while all side-chain atoms are moved to the C_α atom position.
ref_element	$(14, 4)$	We encode the backbone atoms based on their elemental types [N, C, O], while the side-chain atoms are encoded as a single class using 'UNK' (unknown).
ref_space_uid	$(N_{atom},)$	Numerical encoding of the residue index associated with this reference conformer.
ref_center_mask	$(N_{atom},)$	Masks indicating the center atom of the residue.
residue_index	$(N_{res},)$	The pdb residue number for calculating relative positional embedding
residx_embedding	$(N_{res}, 32)$	The absolute position embedding by sinusoidal positional encoding.
template_distogram	$(N_{res}, N_{res}, 39)$	Pairwise distogram of pseudo C_β are discretized into 38 bins of equal width between of bin_min=3.25Å, bin_max=50.75Å, one more bin contains any larger distances.
template_cb_mask	$(N_{res},)$	Mask indicating if the C_β atom has coordinates for the template at this residue, where 1 indicates existing residues and 0 is used for padding residues.
template_time	$(N_{res}, N_{res}, 1)$	Normalized pairwise time-step feature, ranging from 0 to 1.
all_atom_positions	$(N_{atom}, 3)$	The noisy position of all atoms in the system.
all_atom_mask	$(N_{atom},)$	Mask indicating which atom slots are used in the the system.
t_hat	(1,)	The noise level value for adding noise.

Table 5 provides a detailed list of the hyperparameters used for training.

Table 5. Pallatom training hyperparameters.

Parameter name	**Value**
Batch size	32
Learning rate	0.001, No warm-up.
Examples per epoch	35156
Crop size	128
Loss weights	Sequence loss weight $\alpha_0 = 0.25$, Smooth lddt loss weight $\alpha_1 = 1.0$, Residue-level distogram loss weight $\alpha_2 = 0.5$, Atomic-level distogram loss weight $\alpha_3 = 0.5$, Intermediate loss weight $\alpha_4 = 1.0$ In the basic loss $\mathcal{L}_0$, weight allocation was applied to residue types, with a weight of 2.0 assigned to polar residues, and 1.0 to the others.
Diffusion timesteps (N_{steps} or T)	200
Self-condition rate	100%
EDM	Noise schedule `lognormal`. $\ln(\sigma) \sim \mathcal{N}(P_{mean}, P_{std}^2)$, $P_{mean} = -1.2$, $P_{std} = 1.5$, $\sigma_{data} = 16$, Stochastic sampler $t_{min} = 0.01$, $t_{max} = 1.0$, noise level $\gamma_0 = 0.2$, noise scale $\lambda = 1.003$, step scale $\eta = 2.25$
Transformer	single representation dimension = 256, pair representation dimension=128, number of heads = 8, number of decoder units = 8
Training Steps	3×10^6
Training time	$\approx$ 10 days
Device	$4 \times$ A6000

G. Additional Results

G.1. Out-of-distribution performance

Evaluation of Metrics We performed extensive evaluation of Pallatom's performance on out-of-distribution sequence lengths, specifically testing its generalization capabilities on proteins significantly longer than those encountered during training. Specifically, we generated and analyzed 250 protein structures for each of six distinct lengths $L = 150, 200, 250, 300, 350, 400$, providing comprehensive evaluation across a broad spectrum of sequence lengths. Notably, while our training set employed a crop size of 128, all comparative methods were trained on datasets containing proteins with lengths up to 384, highlighting a significant difference in training data complexity. Figures 3 illustrates the comparative performance across designability (DES), structural diversity (DIV-str), and structural novelty (NOV-str) metrics for various sequence lengths under both CO-DESIGN 1 and PMPNN 1 evaluation modes, providing comprehensive insights into the model's length-dependent behavior.

Our analysis revealed that Pallatom consistently achieved superior designability metrics for sequences up to twice the maximum training length ($L = 150 - 250$) across both evaluation modes, while simultaneously demonstrating significantly greater structural diversity and novelty compared to Multiflow. Even when extending to more than three times the maximum training length at $L = 350$ and $L = 400$, Pallatom, although less advantageous in backbone design, still maintained the ability to generate all-atom proteins, a feat unachievable by the other two comparison methods, Protpardelle and ProteinGenerator.

Secondary Structure Analysis We further analyzed the secondary structure preferences of sampled structures from all methods. Specifically, we utilized DSSP to classify the secondary structure of each residue in the proteins.

We established a classification scheme based on helix-to-sheet ratios: proteins with α-helix residues exceeding five times β-residues are categorized as **HHH** (all-helix), while those with β residues surpassing five times α-helix residues are classified as **EEE** (all-β), Proteins with balanced proportions of α-helices and β-sheets are classified as **HEL**, representing mixed $\alpha\beta$

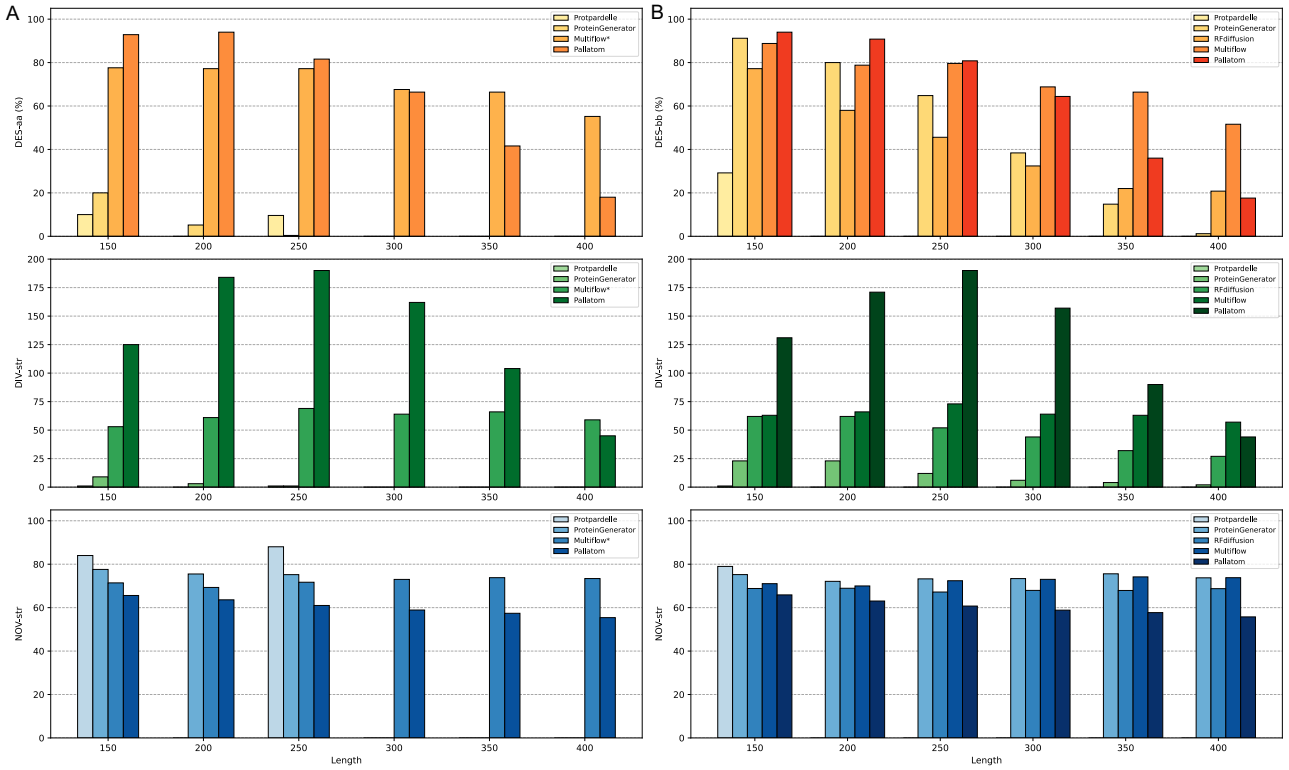

Figure 3. Comparison of Evaluation Metrics for Sampled Proteins at Longer Lengths. (A) and (B) respectively show the designability, structural diversity, and structural novelty under the CO-DESIGN 1 mode and the PMPNN 1 mode.

structures, completing our comprehensive secondary structure classification framework.

Figure 4 presents the secondary structure distributions for all methods under both evaluation modes, along with the corresponding distributions for designable protein subsets. This result indicates the ProteinGenerator, multiflow, and Pallatom exhibit similar secondary structure preferences within the length range of 150-400, with a roughly equal distribution between **HEL** and **HHH** structures. In contrast, RFdiffusion and Protpardelle show a stronger preference for **HEL** structures. Within the 150-400 length range, all models rarely succeed in generating **EEE** structures.

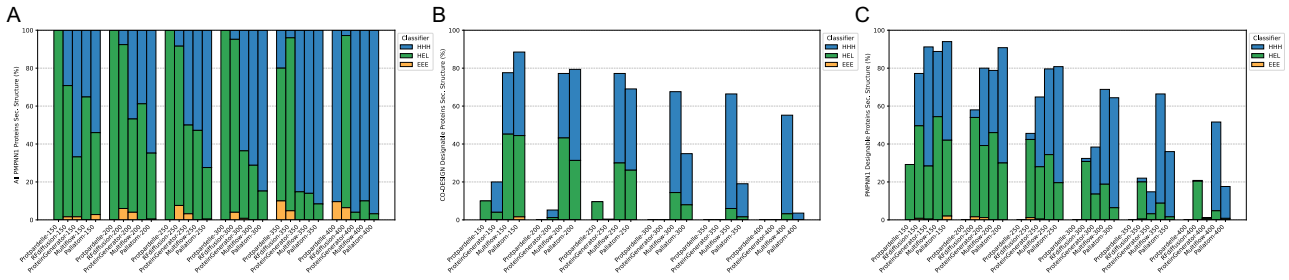

Figure 4. Secondary Structure Distribution of Sampled Proteins at Longer Lengths. Figures (A), (B), and (C) show the secondary structure distribution of all sampled proteins across all methods, the designable proteins in CO-DESIGN 1 mode, and the designable proteins in PMPNN 1 mode, respectively.

All these experimental results demonstrate Pallatom's superior performance, highlighting its exceptional scalability and generalization capabilities. To further illustrate these advantages, we present additional case studies: Figure 6 showcases novel designable protein structures generated by Pallatom, while Figure 7 demonstrates its ability to produce high-quality designs for length distributions beyond the training set scope.

G.2. Analysis of Sampling Hyperparameters

We analyzed the effect of the step scale η on sampling proteins of unseen longer lengths. In Table 6, the left column for each length corresponds to $\eta = 2.5$, while the right column corresponds to $\eta = 3.0$. We observed that a larger step size in the gradient update direction improves the designability of proteins, and as the number of designable samples increases, the structural diversity of generated proteins also increases, with only a slight decrease in novelty. Additionally, consistent with the observations in the main text regarding the impact of sampling hyperparameters on secondary structure distribution, a larger step size is associated with a more pronounced preference for all-helix structures.

Table 6. Pallatom sample metrics with step scale $\eta = 2.5$ (left) and $\eta = 3.0$ (right).

Length	**150**		**200**		**250**		**300**		**350**		**400**	
DES-aa	88%	**93%**	79%	**93%**	69%	**81%**	35%	**66%**	19%	**41%**	4%	**18%**
DIV-str	**131**	125	153	**184**	149	**190**	73	**162**	38	**104**	9	**45**
NOV-str	**0.650**	0.656	**0.618**	0.636	**0.594**	0.610	**0.576**	0.589	**0.551**	0.574	**0.525**	0.554
HHH(%)	44%	53%	53%	64%	54%	70%	60%	77%	64%	85%	69%	92%
HEL(%)	53%	44%	47%	36%	46%	30%	40%	23%	36%	15%	31%	8%
EEE(%)	2.8%	2.7%	0.0%	0.4%	0.0%	0.4%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

G.3. Sequence Quality of Designable Proteins

We compared the quality of the sequences generated by Pallatom with those produced by ProteinMPNN for the same protein structures designed by Pallatom. Figure 5 shows the pLDDT scores of the two sequences predicted by ESMfold. We found that the sequence confidence score of Pallatom is slightly lower than that of ProteinMPNN, with the maximum mean pLDDT difference not exceeding 2. We attribute this difference to both the training data and the task. Regarding training data, Pallatom was trained on a monomer protein dataset, whereas ProteinMPNN was trained on a more diverse dataset that includes both monomer and multichain structures. Additionally, when preparing the training set, ProteinMPNN focused more on the sequence diversity under the same structure, while Pallatom needed to consider both sequence and structure diversity. In terms of the training task, the objectives of the two models are fundamentally different. ProteinMPNN is concerned solely with sequence design given a real backbone, whereas Pallatom must balance the dual objectives of structure generation and sequence generation from pure noise.

G.4. Analysis of Sampling Time

We conducted a comparative analysis of sampling times for each method. Specifically, we standardize the diffusion sampling steps to $T = 200$ and sample 100 proteins for each length, calculating the mean and standard deviation. All methods were tested on the same hardware: CPU: AMD EPYC 7402 @2.8GHz, GPU: NVIDIA GeForce RTX 4090 with 24GB VRAM.

Table 7 presents the results. Thanks to JAX's JIT compilation and our optimizations at the atom level of Attention, Pallatom achieved the second fastest sampling speeds for lengths ranging from 100 to 350, outperforming all methods except Protpardelle. At $L = 400$, even with the atomic-level length reaching $(14 \times 400) \times (14 \times 400)$, Pallatom's performance remains comparable to the second-fastest method, Multiflow, and is 5 times faster than RFdiffusion and 16 times faster than ProteinGenerator.

Table 7. Sampling Time (in seconds). The shortest time is highlighted in bold, and the second shortest is indicated in italics.

Length	**100**	**150**	**200**	**250**	**300**	**350**	**400**
Protpardelle	*10.9±0.1*	**11.3±0.2**	**11.7±0.2**	**12.5±0.2**	**24.5±0.5**	**26.3±0.6**	**27.6±0.9**
ProteinGenerator	414.1±107.5	389.5±2.3	388.8±1.3	477.6±1.7	624.0±4.1	796.3±4.6	950.3±5.3
Multiflow	25.3±0.4	25.3±0.6	27.1±0.3	29.4±0.2	35.1±0.6	40.5±0.2	*46.6±0.5*
RFdiffusion	95.5±14.6	93.9±1.0	106.3±0.9	138.5±0.8	183.0±0.8	230.7±1.6	287.4±6.5
Pallatom*	**10.2±0.1**	*12.1±0.1*	*18.2±0.1*	*21.9±0.1*	*33.3±0.1*	*40.4±0.1*	57.5±0.1

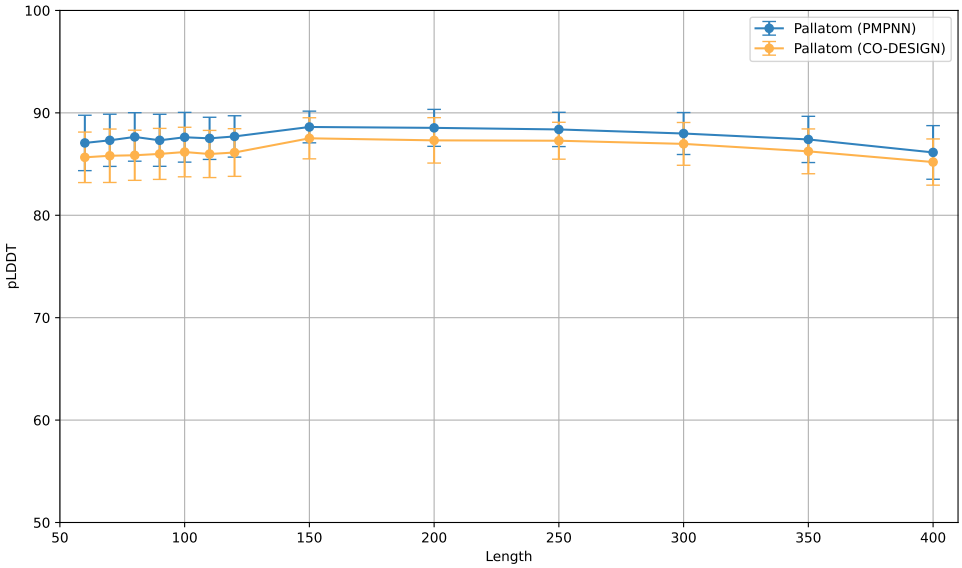

Figure 5. Comparison of pLDDT between sequences designed by Pallatom and ProteinMPNN across different lengths. Sequences designed by Pallatom are labeled as "Pallatom (CO-DESIGN)," while sequences designed by ProteinMPNN based on the backbone are labeled as "Pallatom (PMPNN)."

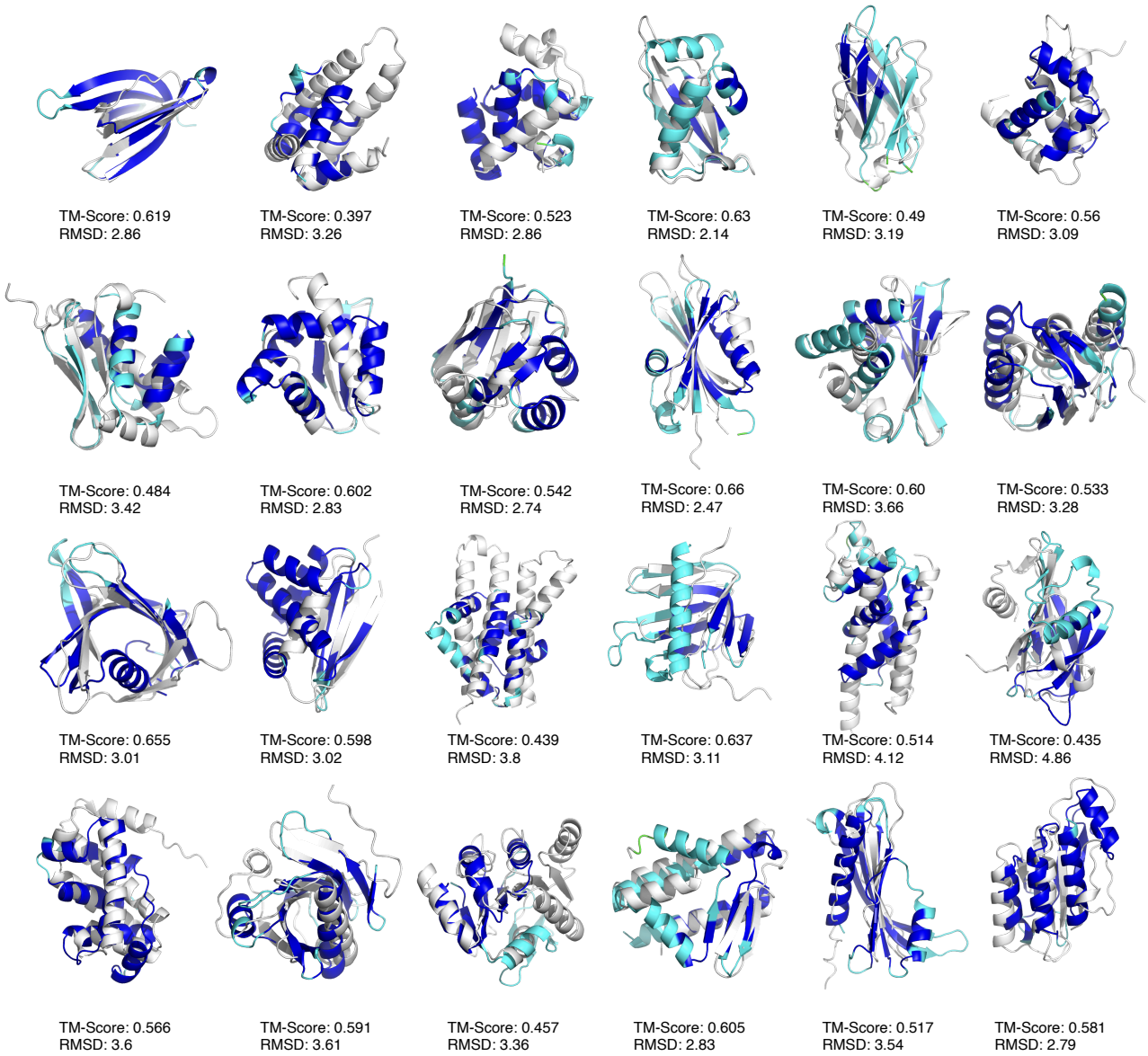

Figure 6. Additional novel designable proteins generated by Pallatom. The blue structures in the figure represent the designable protein sequences generated by Pallatom, which have been predicted using ESMfold and colored based on pLDDT scores. The white structures are the nearest neighbors from the Foldseek database (using the default eight databases on the Foldseek web server), with the distances between the two sets of structures evaluated using TM-Score and RMSD.

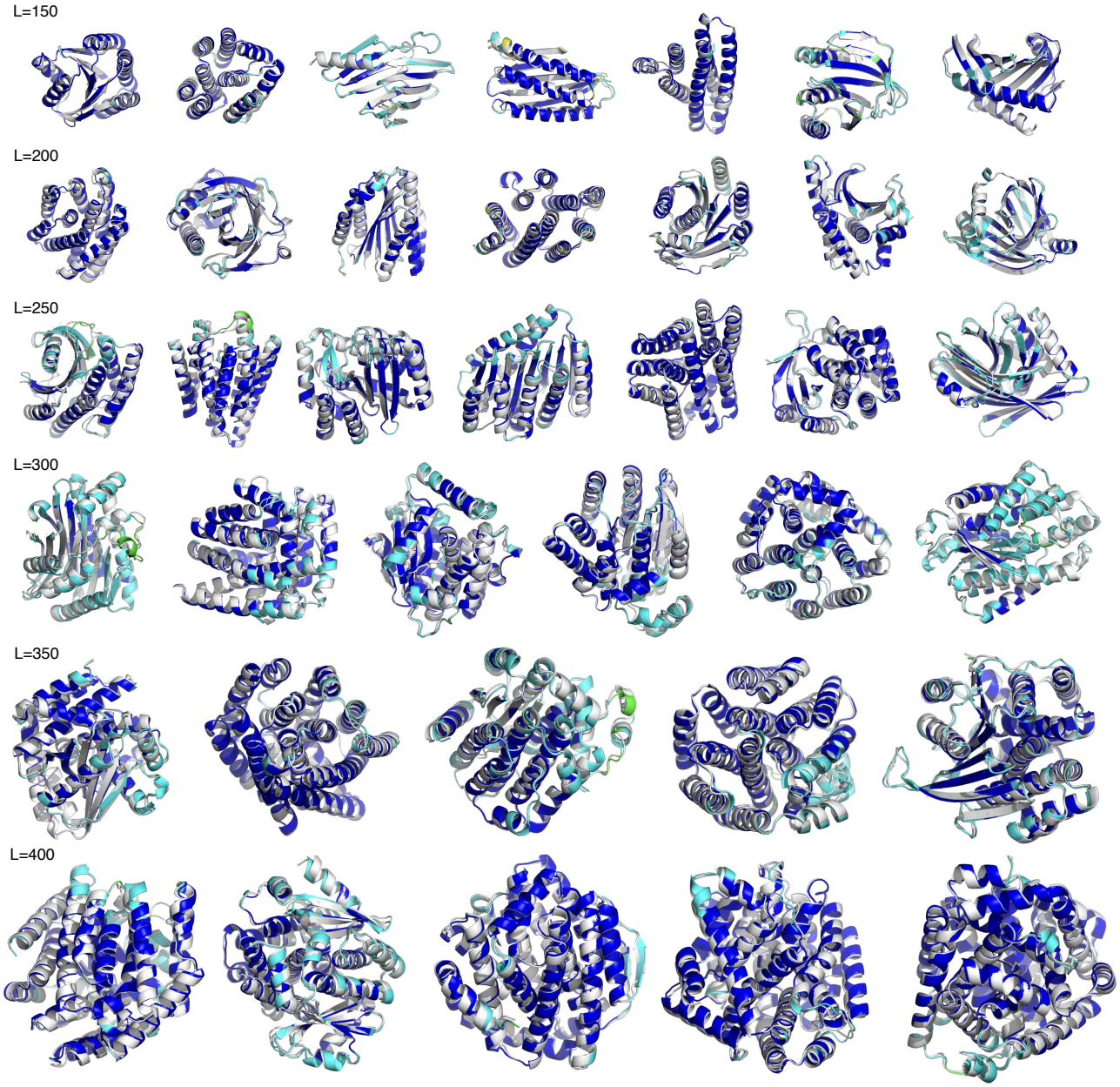

Figure 7. Additional all-atom protein structures generated by Pallatom for longer sequence lengths. The white structures represent those generated by Pallatom, while the colored structures are predicted by ESMfold and are colored according to pLDDT values, with bluer hues indicating higher pLDDT scores.