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 token-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 fix-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 fix-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 conformations and leading to an insufficient description of the

[*]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>.

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. Inspired by AlphaFold3 (Abramson et al., 2024), we adopt a dual-track framework that tokenizes proteins into residues or atoms, and develop a novel module incorporating "traversing" representations and multi-layer decoding units. This module efficiently integrates and updates token-level and atomic-level representations through a dual-track recycling mechanism, enabling self-conditioned inferencing and enhancing information flow between blocks. Additionally, we propose a new amino acid coordinates representation, `atom14`, to address the challenge of representing unknown side-chain coordinates. Introducing virtual atoms to all amino acids type prevents sequence information leakage problem. The key insight of Pallatom is recognizing that all-atom coordinates of a protein inherently encode both structural and sequence information. 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 develop a network architecture for all-atom protein generation tasks, which effectively represents both protein backbones and sidechains.
- We explore the `atom14` representation to achieve a unified description of unknown amino acid side-chain coordinates in generative tasks.
- 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

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 define a representation called `atom14`, which pads the initial protein with L residues as $\mathbf{x} = \{x^l\}_{l=1}^{L} \rightarrow \mathbf{x}_0$, considering it as a 3D point cloud distribution $P(\mathbf{x}_0) \in \mathbb{R}^{L\times 14\times 3}$. For example, if residue x^l is CYS, its coordinates $[\mathrm{N}, \mathrm{C}_\alpha, \mathrm{C}, \mathrm{O}, \mathrm{C}_\beta, \mathrm{S}_\gamma] \in \mathbb{R}^{6\times 3}$ is padded with 8 virtual atoms that coincide with its C_α position, resulting in $x^l \rightarrow x_0^l \in \mathbb{R}^{14\times 3}$.

Our approach assumes that the all-atom distribution of the protein sidechains, including properties such as hydrophobicity, polarity, and even hydrogen bonds and salt bridges, are inherently encoded within the all-atom coordinates distribution. Even without knowing the specific element atom type, the conformationally similar amino acids, such as CYS and SER, can be distinguished from atomic-level features. Therefore, we additionally trained a single "visualization" head to predict the corresponding amino acid type. 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. Additionally, we provide the alanine reference conformer features to guide the network in forming a stable backbone frame conformation.

2.2. Diffusion modeling on all-atom protein

The use of all-atom representation eliminates the constraints imposed by the complex forms of the SE(3) frame (Yim et al., 2023b) and the Riemannian diffusion framework (De Bortoli et al., 2022). Diffusion-based generative models using Gaussian noise distribution have a strong theoretical foundation, with various adaptations influenced by factors such as the sampling schedule, training dynamics (Song et al., 2021). We adopt the EDM framework (Karras et al., 2022) with slight modifications and employed a Gaussian diffusion model on $\mathbb{R}^{L\times 14\times 3}$.

Assuming the data distribution of protein coordinates under the `atom14` 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 $p(\mathbf{x};\sigma) = \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$. When $\sigma_{max} \gg \sigma_{data}$, $p(\mathbf{x};\sigma_{max})$ approximates pure Gaussian noise. For a noise schedule $\sigma(t) = t$ (following EDM notation where $\sigma(t)$ indicates the noise schedule and σ_t represents the noise level sampled from $p_{train}(t)$ at time t), 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 $p(\mathbf{x};\sigma)$. A neural network $D_\theta(\mathbf{x},\sigma)$ is typically trained for each σ_t using the following loss function

to match the score function (Song et al., 2021):

$$\mathbb{E}_{\mathbf{x}_0 \sim p_{data}, \mathbf{x}_t \sim p(\mathbf{x}_0;\sigma_t)}[\lambda(\sigma)||D_\theta(\mathbf{x}_t, \sigma_t) - \mathbf{x}_0||_2^2], \quad (2)$$

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

Practically, we employed the EDM preconditioning technique, which resulted in improved generation performance. Consequently, we can derive D_θ and the loss function as:

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

$$\mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t}[\lambda(\sigma)||c_{out}(\sigma)F_\theta(c_{in}(\sigma)\cdot\mathbf{x}_t, c_{noise}(\sigma)) - (\mathbf{x}_0 - c_{skip}(\sigma)\cdot\mathbf{x}_t)||_2^2] \quad (4)$$

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

2.3. Framework for all-atom protein generation

AlphaFold3 provided an excellent initial framework for all-atom protein generation. However, the generation task fundamentally differs from the structure prediction task, requiring us to tailor the structure prediction framework with several modifications. In structure prediction, numerous co-evolutionary signals can typically be extracted from homologous sequences, which can then be decoded into a three-dimensional structure. In contrast, the generation task begins with noised coordinates, requiring the model to learn how to extract structural information iteratively.

Providing self-conditions (Chen et al., 2022) is an effective way to enhance generation performance. A simple approach is to integrate the information extraction modules directly into the structural decoder, allowing the partially denoised information from each block to be used as self-conditions for the subsequent block. More specifically, in our modifications, the denoised coordinates from each block are converted into a pair-wise feature, which is reinjected into pair representation and used as self-conditions in the next block to update single representation. These modifications effectively address the self-conditioning pipeline between blocks.

In practice, the interaction and updating of dual-track protein representations within multiple decoder units present new challenges. We find that if residual connections are used simultaneously for both token-level and atomic-level representations across multiple decoder units, the token-level representation tends to be repeatedly broadcast and inappropriately accumulated. This not only leads to numerical instability in atomic-level representations but also disrupts the model's performance. Therefore, we propose using traversal atomic-level representations to carry the token-level representations within the current decoding unit. These two modifications enable the new framework to effectively generate all-atom protein structures. Details can be found at Figure 1.

3. Method

In this section, we first describe the details and implementation of the network framework. Next, we outline the objective for training the model. Finally, we demonstrate the process of sampling and obtaining the all-atom protein.

3.1. `MainTrunk`: the denoisng network

We refer to the protein generation task as the generation of all-atom coordinates. In the `atom14` representation, a protein with L residues can be expressed as $\mathbf{x}_0 = \{x_0^l\}_{l=1}^L$, which $x_0^l \in \mathbb{R}^{14\times3}$ represents the all-atom coordinates of a residue. For each time step t in the diffusion process, the network predicts the updated coordinates from the input $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_0, \sigma_t^2\mathbf{I})$.

The network comprises two main components: an input encoding module and multiple iterative decoding units. Figure 1A illustrates the main architecture. We adopt the dual-track framework, the atomic-level representation for atoms using local attention mechanisms and the token-level representation for residues using global attention mechanisms.

For feature initialization and encoding, we utilize coordinates feature from the standard conformation of alanine, the positional encoding within the residues and the noisy coordinates vector to initialize 1D atomic-level representations. The 1D representations are then updated using a 3-layer AtomTransformer encoder, with the initial atomic pair representations as attention bias. These atomic-level representations were regarded as static traversing atomic-level representations (c_l^{skip}, q_l^{skip} and p_{lm}^{skip} in Figure 1B). The single representation initialization is performed using the diffusion timestep feature and the positional embedding. The pair representation is initialized using the relative positional embedding introduced in AlphaFold2 (Jumper et al., 2021) and a self-conditioned template distogram feature from the previous prediction. The interaction between token-level and atomic-level information is propagated through broadcasting and aggregation. Detailed features are recorded in Appendix Table 3.

For the decoding part, we employ an iterative update mechanism where token-level representations are broadcast to the atomic level, and atomic information is then fed back through a recycling process.

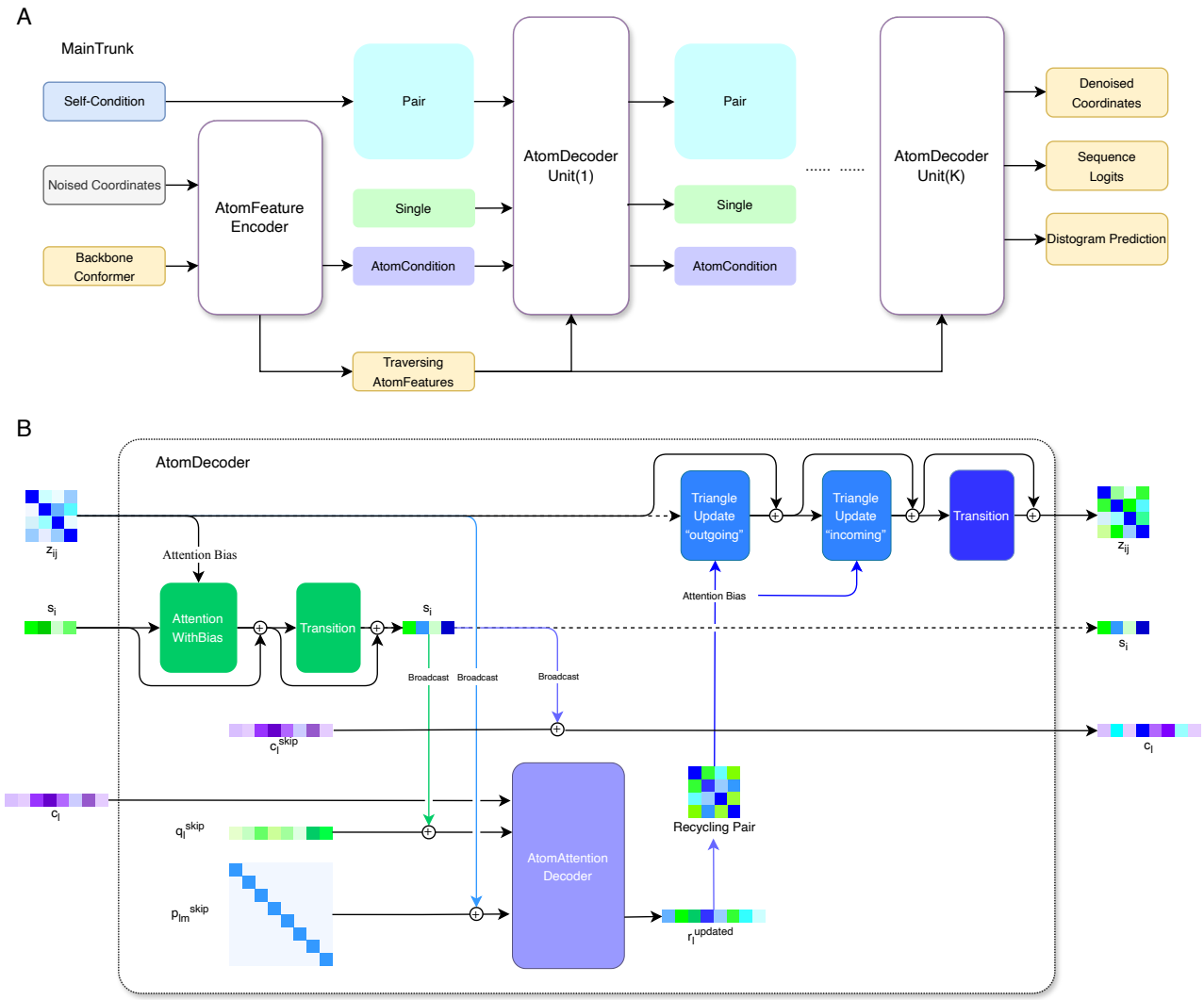

Figure 1. Pallatom model framework. (A) Framework of the main backbone network `MainTrunk`. (B) The detailed architecture of the AtomDecoder unit.

In more detail, the single representation is first updated at the token level, specifically, for the k-th unit, the pair representation from $k-1$ is used as an attention bias to update the single representation. for $k=0$, a self-conditioned pair embedding is applied. The updated single representation is then re-injected into the traversal conditional 1D atomic representation c_l^{skip} as self-conditioning for the next block.

To address the issue of residual connections in the atomic representation mentioned earlier, for each block, we use an intermediate atomic-level representations, which is composed of broadcast token-level representations and traversing atomic-level representations, to predict coordinates updates $\mathbf{x}_{update}^{(k)}$, the current predicted structure $\mathbf{x}_0^{(k)}$ using the cumulative updates from the first k units.

$$\mathbf{x}_0^{(k)} = c_{skip}(\sigma_t) \cdot \mathbf{x}_t + c_{out}(\sigma_t) \cdot \mathbf{x}_{updates}^{(k)}$$

After the coordinates are denoised, they are transformed to relative distance matrix $\mathbf{x}_0^{(k)}$ and recycled back into the pair representation via triangle attention layer, effectively addressing the challenges of updating pair representations. In practice, we found that recycling pair information accelerates model training and enhances inference capabilities. Utilizing the minimal decoding unit and supervised training on intermediate coordinates simplifies the scaling of the network and allows for increasing its depth. Through the iterative 8-layer decoder, we effectively updated the atomic-level and token-level representations. The cumulative updating mechanism for coordinates prediction allows the model to gradually refine these predictions, ultimately

leading to the realistic all-atom protein structure. Details can be found at Appendix Algorithm 2.

3.2. `SeqHead`: sequence decoder

To convert the generated coordinates into a real protein, we need a module that translates the position information into an amino acid sequence. We add a `SeqHead` to each decoding unit. Specifically, we aggregate the updated 1D atomic-level representations from the AtomAttentionDecoder corresponding to each token and then employ a linear layer to predict the logits for the 20 amino acid types $\hat{\mathbf{a}}^{(k)} \in \mathbb{R}^{L\times 20}$. We take the predictions from the last unit as the final sequence logits, $\hat{\mathbf{a}} = \hat{\mathbf{a}}^{(K)}$.

3.3. 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 similar to that in Alphafold3. We first perform a rigid alignment of the ground truth $\mathbf{x}_0$ on the denoised structure $\hat{\mathbf{x}}_0$ as $\mathbf{x}_0^{\text{aligned}}$. The MSE loss is then defined as:

$$\mathcal{L}_{\text{atom}} = \frac{||\hat{\mathbf{x}}_0 - \mathbf{x}_0^{\text{aligned}}||^2}{3L}$$

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 basic loss function of the network is:

$$\mathcal{L}_0 = \lambda(\sigma_t) \cdot \mathcal{L}_{\text{atom}} + \alpha_0 \cdot \mathcal{L}_{\text{seq}}$$

To capture the fine-grained characteristics of the all-atom structure, we introduce the smooth local distance difference test (LDDT) loss $\mathcal{L}_{\text{smooth_lddt}}(\hat{\mathbf{x}}_0, \mathbf{x}_0)$ from AlphaFold3. The specific algorithm can be found in the Appendix Algorithm 8. To supervise the updating of pair-wise features, we employ the distogram head loss to constrain the global distance distribution at the token level. The pair representation is symmetrized and projected into 64 distance bins with probability p_{ij}^b, and supervised with one-hot encoded target bins y_{ij}^b.

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

Similarly, to supervise the local relative distance distribution at the atomic-level, we project the 2D features from local atomic attention into 22 distance bins from 0 $\mathring{A}$ to 10 $\mathring{A}$ with q_{nm}^b, and construct the loss with one-hot encoded target bins at the atomic-level. Here, the local region corresponds to the area calculated within the local attention on the $14L \times 14L$ atomic-level map.

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

In the early stages of development, we discovered that supervising the intermediate sequences and structures predicted by each decoder unit significantly enhanced both network performance and inference stability. Furthermore, we introduced a loss weight decay mechanism with $\gamma = 0.99$ between different blocks, assigning greater weight to the later layers.

$$\mathcal{L}_{\text{med}}^k = \lambda(\sigma_t) \cdot \frac{||\mathbf{x}_0^{(k)} - \mathbf{x}_0^{\text{aligned}}||^2}{3L} + \alpha_0 \cdot \text{CE}(\mathbf{a}_0^{(k)}, \mathbf{a}_0),$$

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

The total loss can be written as:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_0 &+ \alpha_1 \cdot \mathcal{L}_{\text{smooth_lddt}} + \alpha_2 \cdot \mathcal{L}_{\text{dist_token}} \\ &+ \alpha_3 \cdot \mathcal{L}_{\text{dist_atom}} + \alpha_4 \cdot \mathcal{L}_{\text{med}}\end{aligned} \tag{5}$$

3.4. Sampling

The sampling process is described in Algorithm 1. The modules highlighted in blue are identical to those proposed in AlphaFold3. The initial atoms are sampled from a Gaussian distribution on $\mathbb{R}^{L\times 14\times 3}$. 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.

4. Experiments

Pallatom is trained and evaluated on monomer proteins. In this section, we first introduce the training settings and the datasets. Next, we describe the commonly used metrics for protein backbone generation and adapt these metrics for all-atom proteins. Finally, we present and analyze the results of the Pallatom evaluation.

4.1. Training setting

The training dataset of the model includes the PDB (Zardecki et al., 2022) and AlphaFold Database (AFDB)

Algorithm 1 Pallatom Inference

def SampleDiffusion (f^*, $T = 200$, $\lambda = 1.003$, $\eta = 1.5$, $\gamma_0 = 0.2$, $t_{min} = 0.15$, $t_{max} = 1.0$)
$\delta_t = 1/T$
$c_T = \text{GetNoiseSchedule}(1 - \text{uniform}(0,1) \cdot \delta_t)$
$\vec{\mathbf{r}}_l \sim c_T \cdot \mathcal{N}(\vec{0}, \mathbf{I}_3)$
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 $t_{min} \leq t/T \leq t_{max}$ else 0
  $\hat{t} = c_\tau(\gamma + 1)$
  $\vec{\mathbf{r}}_l^{nosiy} = \vec{\mathbf{r}}_l + \lambda\sqrt{\hat{t}^2 - c_\tau^2} \cdot \mathcal{N}(\vec{0}, \mathbf{I}_3)$
  $\vec{\mathbf{r}}_l^{denoised}, \mathbf{f}_i^{\text{seq_logits}} = \text{MainTrunk}($
    $\{f^*\}, \vec{\mathbf{r}}_l^{nosiy}, \hat{t}, t_p)$
  $\vec{\delta}_l = (\vec{\mathbf{r}}_l^{nosiy} - \vec{\mathbf{r}}_l^{denoised})/\hat{t}$
  $dt = c_{\tau-1} - \hat{t}$
  $\vec{\mathbf{r}}_l \leftarrow \vec{\mathbf{r}}_l^{noisy} + \eta \cdot dt \cdot \vec{\delta}_l$
end for
return $\{\vec{\mathbf{r}}_l\}$, $\{\mathbf{f}_i^{\text{seq_logits}}\}$

(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. We focus on small proteins that can be easily synthesized using commercial oligo-pool method and the models are trained on crops of lengths up to 128. Pallatom is coded in Jax, easily compiling and applying to training samples of different lengths. The model training utilized the Adam optimizer (Kingma & Ba, 2017) with a learning rate of 1e-4, β_1 = 0.9, β_2 = 0.999, and a batch size of 32. Training was conducted over 300,000 steps across 10 days using 4 NVIDIA A6000 GPUs.

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, indicating that the sequence is likely to adopt its intended native fold.

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. Similarly, for all-atom protein sequences, we evaluated the maximum similarity of the generated sequence to proteins in the PDB using MMseqs2, denoted as **NOV-seq**.

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 from both structural and sequence perspectives.
- PMPNN 1: Other methods use **DES-bb** with $N_{seq} = 1$. Diversity and novelty are evaluated only from the structural perspective.

4.3. Results

We sample Pallatom with 200 time steps using a noise scale $\gamma_0 = 0.2$, a step scale $\eta = 2.0$ 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 (Chu et al., 2024) and ProteinGenerator (Lisanza et al., 2023). For backbone generation, we compare with RFdiffusion (Watson et al., 2023). All methods are evaluated using their open-source code and default parameters.

As shown in Table 1, Pallatom surpasses previous methods in the CO-DESIGN 1 evaluation for generating all-atom proteins. We find that the sequences generated by Pallatom fit the backbone more accurately than those generated by Pro-

Table 1. Comparison of various methods. Protpardelle exists in two versions: one that outputs all-atom proteins end-to-end, and another that performs a two-stage repacking of sequences and side-chain using standard ProteinMPNN based on the designed backbone, which belongs to the same generation paradigm as backbone generation. We use the end-to-end version of Protpardelle for evaluation.

Method	**CO-DESIGN 1**			**PMPNN 1**		
	DES-aa (↑)	**DIV-str/seq (↑)**	**NOV-str/seq (↓)**	**DES-bb (↑)**	**DIV-str (↑)**	**NOV-str (↓)**
Protpardelle*	0.40%	6 / 7	0.784 / 6.7×10^{-2}	72.34%	84	0.755
ProteinGenerator	59.37%	113 / 966	0.780 / $\mathbf{2.4\times10^{-3}}$	**95.03%**	113	0.788
RFdiffusion	N/A	N/A	N/A	81.60%	166	0.797
Pallatom	**88.83%**	**345 / 1552**	**0.714** / 5.6×10^{-3}	85.49%	**313**	**0.712**

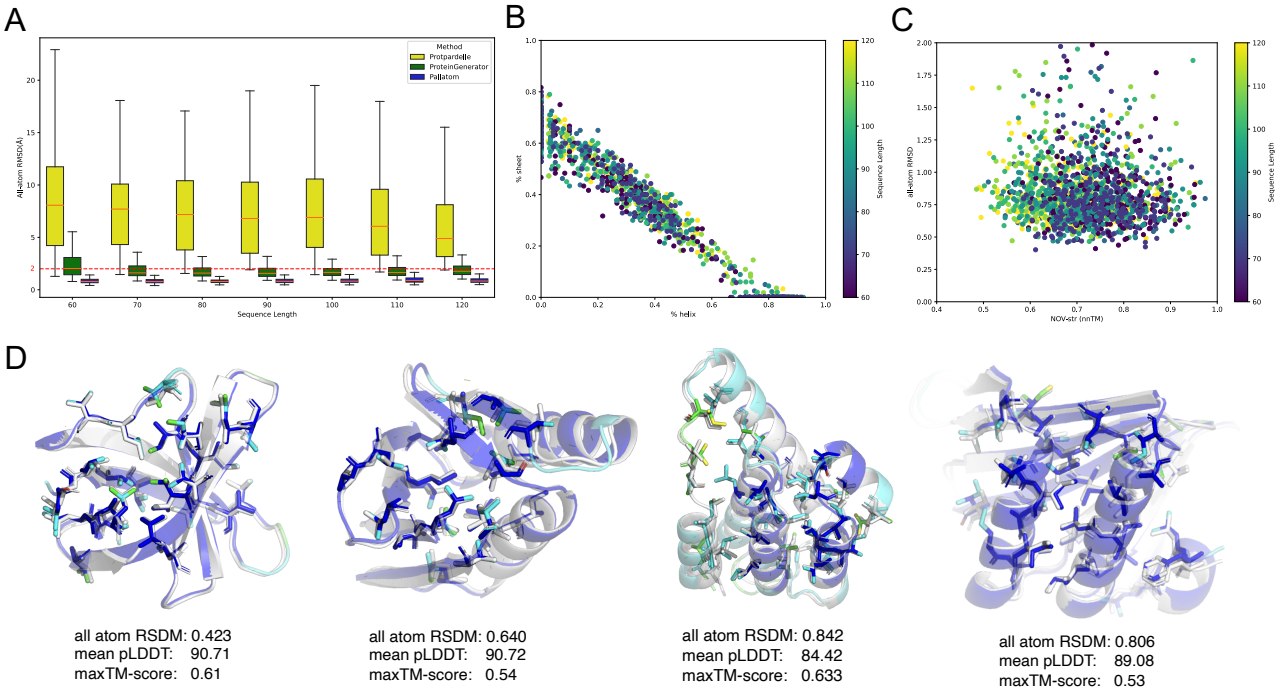

Figure 2. Evaluation of proteins sampled from Pallatom. (A) Boxplot of all-atom RMSD for predicted proteins under various lengths. (B) Secondary structure of designable proteins sampled by Pallatom, computed by DSSP. (C) Novelty distribution of Pallatom's designable proteins. The nnTM represents the TM score with the nearest neighbor structure in the PDB database, extracted using FoldSeek. (D) Examples of high-quality, novel all-atom sampled proteins.

teinMPNN. Specifically, even under the more stringent DES-aa metric, which includes all-atom RMSD and pLDDT constraints, Pallatom demonstrates higher designability compared to the DES-bb metric under the PMPNN 1 mode, which is based on C_α-RMSD. This supports our hypothesis that learning $P(\textit{all-atom})$ can effectively capture the relationship between protein structure and sequence.

Additionally, the all-atom protein structures generated by Pallatom show greater structural diversity. With the same number of samples, Pallatom achieves twice the diversity of RFdiffusion and three times that of ProteinGenerator. This highlights Pallatom's enhanced ability to explore a wider range of conformations.

In terms of novelty, our analysis reveals that the sequences generated by all all-atom protein models exhibit remarkably low similarity to those in the PDB database. This suggests that these models are capable of producing highly novel sequences while exploring the dark space of protein design.

In the evaluation of backbone generation, ProteinGenerator achieves an almost perfect backbone designability, significantly surpassing other models. This indicates that sequence-based generative models can capture highly designable backbone structures. However, its performance in structural diversity falls short compared to structure-based

generative models, highlighting a subtle trade-off between designability (DES-bb) and structural diversity (DIV-str). Protpardelle performs below its counterparts across all metrics, suggesting that the network architecture and atomic representation play a crucial role in generative capability. Pallatom demonstrates performance comparable to the leading methods with significantly greater novelty and diversity. Similar to the results of the CO-DESIGN 1 evaluation, Pallatom not only generates protein structures with higher diversity but also maintains strong designability and novelty.

We present a more detailed analysis in Figure 2. Pallatom maintains stable performance across all tested protein lengths, showing no preference or bias toward any specific length and achieving a balanced distribution of secondary structures. In Figure 2C, the novelty distribution of designable proteins is shown, revealing that longer sampled proteins tend to exhibit lower TM-scores.

Several designable all-atom proteins for case studies in Figure 2D. These highly novel proteins designed by Pallatom show a highly ordered side-chain distribution, with hydrophobic side-chain concentrated internally to form a stable hydrophobic core, and the surface covered by hydrophilic polar residues. This distribution is consistent with the side-chain distribution of high pLDDT structures predicted by ESMFold, demonstrating that Pallatom has learned the physical and chemical properties that govern protein folding and residue distribution from the all-atom distribution.

Table 2. Pallatom sample metrics.

Noise Scale γ_0	0.2	0.2	0.2	0.1	0.2
Step Scale η	1.5	1.7	2.0	2.0	2.0
$N_{\textbf{steps}}$	200	200	200	200	500
DES-aa (↑)	52%	78%	89%	89%	87%
DIV-str (↑)	74	89	80	68	67

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 a significant improvement in designability by adjusting the step scale $\eta = 2.0$ and introducing additional noise to compute the update gradient, although with a slight decrease in diversity. Furthermore, we found that reducing the additional noise level to $\gamma_0 = 0.1$ decreases structural diversity, consistent with findings from previous work (Yim et al., 2023b). We also discovered that merely increasing the number of sampling steps to $N_{steps} = 500$ does not improve the quality of the generated proteins. Our experiments indicate that Pallatom is capable of learning $P(\text{all-atom})$, enabling the design of high-quality, high-confidence (pLDDT > 0.8) all-atom proteins without relying on pre-trained models or the two-stage design and refinement methods of existing approaches.

5. 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.

6. 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 modification of the `atom14` representation removes the limitations of explicitly defining all amino acid types during generation and

provides a more accurate method for representing all-atom system coordinates. We redesign the dual-track architecture of AlphaFold3 diffusion module, by developing a novel mechanism comprising " traversing" representations and dual-track recycling method. 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, expanding our system to support the representation of small molecules, DNA, and non-standard amino acids, and further enhancing the model's capabilities in designing large complex systems, such as antibody complexes and self-assembling materials.

Code Availibility

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

Impact Statement

The novel protein generation method proposed in this paper, which is based on learning the distribution of all-atom coordinates, aims to advance the paradigm of protein design engineering. This work holds significant potential for broad societal impact and direct applications. For instance, it offers greater utility in tasks such as enzyme design, drug design involving small molecule binding pockets, and other applications requiring all-atom protein constraints.

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

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the Design Space of Diffusion-Based Generative Models, October 2022.

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. Training Datasets

A.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.

A.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 AlphaFold2 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.

B. Algorithms

We provide a detailed description of the Pallatom modules and algorithm workflow below. In the pseudocode, the algorithms highlighted in blue are nearly identical to those of AlphaFold3 and are not expanded to avoid unnecessary repetition.

B.1. MainTrunk

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

B.2. TemplateEmbedder

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

B.3. AtomFeatureEncoder

The core of the AtomFeatureEncoder is derived from AlphaFold3, and we made two adjustments based on specific tasks. First, To process standard protein residue conformations, we reduced the 20 original amino acid conformations to a single backbone conformation, represented by alanine. Additionally, each residue (token) was standardized to have 14 atoms, represented in the `atom14` format. To prevent information leakage from the virtual atoms, they were defined based on the C_α atom. Secondly, to accommodate the recycling update mechanism between blocks, we utilized traversing atomic-level representations, these features flow between blocks and only integrate with token-level information during the decoding phase.

B.4. AtomAttentionDecoder

We established a method for managing token-level and atomic-level information in the decoder layer. The updated single- and pair-representations are simultaneously injected into the traversing atomic-level representations, effectively preventing the repeated accumulation of redundant structural information.

Algorithm 2 MainTrunk

def MainTrunk(f^*, $\mathbf{r}_l^{input}$, $\hat{t}$, t, c_{atom}=128, c_{pair}=128, c_{token}=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: $\mathbf{s}_i^{init} = \text{LinearNoBias}((\mathbf{f}_i^{\text{residue_idx}}))$ $\qquad \mathbf{s}_i^{init} \in \mathbb{R}^{c_{token}}$

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

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

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

 # MainTrunk blocks

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

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

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

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

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

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

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 self-condition positions

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: weighted by sequence

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

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

B.5. NodeUpdate

In AlphaFold3's diffusion module, the single representation is updated by both the AttentionPairBias and ConditionTransition algorithms, which use pre-defined structural representations as conditions. The AttentionPairBias algorithm, which can be viewed as a conditional version of the RowAttentionWithBias algorithm from AlphaFold2, additionly employs Adaptive LayerNorm and SwiGLU techniques to conditionally scale the representation values. The ConditionTransition algorithm acts as a gate. Due to the lack of structural condition representation in generative models, we replace the structural condition with time-based condition in the AttentionPairBias. This allows for adaptive scaling of network updates based on the level of noise. Furthermore, we considered updating the single information between blocks using residual connections instead of ConditionTransition. Our preliminary tests indicate that this update method prevents the degradation of the single representation. We also used dropout during training to mitigate the risk of overfitting and enhance the model's robustness.

Algorithm 6 NodeUpdate

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

 # AttentionPairBias with updated pair bias

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

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

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

B.6. PairUpdate

In the PairUpdate module, to minimize the number of parameters and maximize network depth, we opted to use only the TriangleAttention algorithm, omitting TriangleMultiplication. Ablation studies from AlphaFold2 suggest that relying solely on TriangleAttention still enables accurate protein structure prediction. We used a modified TriangleAttention algorithm, which uses the pair representation from the recycling structure as attention bias. To maintain consistency in the pair feature space, we binned the recycling structure using the same parameters as in the TemplateEmbedder module, with a total of 39 bins ranging from 3.25 to 50.75 Å.

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

 # TriangleAttention with coordinates pair bias

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

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

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

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

B.7. Smooth LDDT loss function

During the training, we adopt the smooth LDDT loss as proposed by AlphaFold3, 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

Table 3. 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	$(N_{token},)$	The pdb residue number for calculating relative positional embedding
residx_embedding	$(N_{token}, 32)$	The absolute position embedding by sinusoidal positional encoding.
template_distogram	$(N_{token}, N_{token}, 39)$	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	$(N_{token},)$	Mask indicating if the C_β atom has coordinates for the template at this residue, where 1 indicates existing tokens and 0 is used for padding tokens.
template_time	$(N_{token}, N_{token}, 1)$	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.