Uni-Fold Symmetry: Harnessing Symmetry in Folding Large Protein Complexes

Ziyao Li[a,b], Shuwen Yang[a,c], Xuyang Liu[a,d], Weijie Chen[a,b], Han Wen[a], Fan Shen[a], Guolin Ke[a,*] and Linfeng Zhang[a,e,*]

[a]DP Technology
[b]Center for Data Science, Peking University
[c]School of Artificial Intelligence, Peking University
[d]School of Mathematical Sciences, Peking University
[e]AI for Science Institute, Beijing

{lizy01,liuxy,chenwj,wenh,shenf,kegl,zhanglf}@dp.tech
swyang@pku.edu.cn

Abstract

Deep folding models have revolutionized the conventional methods of protein complex prediction. However, applying them to large protein oligomers is not easy. These models generally require copying the sequences of identical subunits to capture the in-between relationships. Accordingly, the scales of target protein complexes are strictly limited due to the cubic complexity of these models. To address this issue, we propose UF-Symmetry (Uni-Fold Symmetry), which is extricated from the need of sequence copying via harnessing the intrinsic symmetry of large protein oligomers. Taking the sequences of the asymmetric unit (AU) and a pre-specified symmetry group, UF-Symmetry learns to fold the AU and to assemble the complex structure in an end-to-end manner. By reducing the input scales from entire assemblies to AUs, UF-Symmetry allows to predict much larger assemblies with significant acceleration: for a complex of 4-fold cyclic symmetry (C4) and AU size of 512, UF-Symmetry achieves approximately 20 times acceleration to current methods. On a benchmark of recently released PDB multimers, UF-Symmetry approximately halves the failure rate of current methods and achieves approaching accuracy on commonly successful cases.

Key Words: *protein complex prediction*; *deep learning*; *protein symmetry*.

1 Introduction

Understanding the structures of proteins and their complexes is a key preliminary to life science research and structure-based drug discovery. *Deep folding models*, which predict unknown protein structures via learning from solved ones with neural networks, are now revolutionizing the domain. Extending the success of AlphaFold [1], folding models for protein multimers [2, 3] have now significantly improved the accuracy of conventional docking methods [4, 5].

A fascinating phenomenon of protein complexes is that a majority of them function as large and symmetric oligomers[2]. As is shown in Figure 1, about 72.1% multimeric structures in the Protein Data Bank (PDB) [6] are globally symmetric. Symmetric proteins are evolutionarily preferred due to functional, genetic, and physicochemical needs [7]. For instance, *ion channels* are typically formed as

[*]Guolin Ke and Linfeng Zhang are the corresponding authors of this work.
[2]We use the term *oligomers* to refer to protein complexes of two or more repeating subunits.

Work in progress.

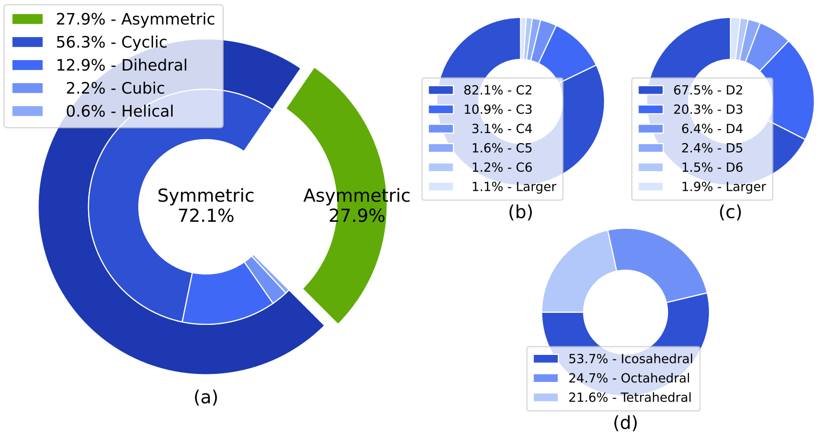

Figure 1: Statistics of symmetric structures in PDB multimers. **(a)** Proportions of different symmetry types in all PDB multimers. **(b) (c) (d)** Proportions of different cyclic, dihedral and cubic symmetries, respectively. Detailed explanations of this figure are in Section A.

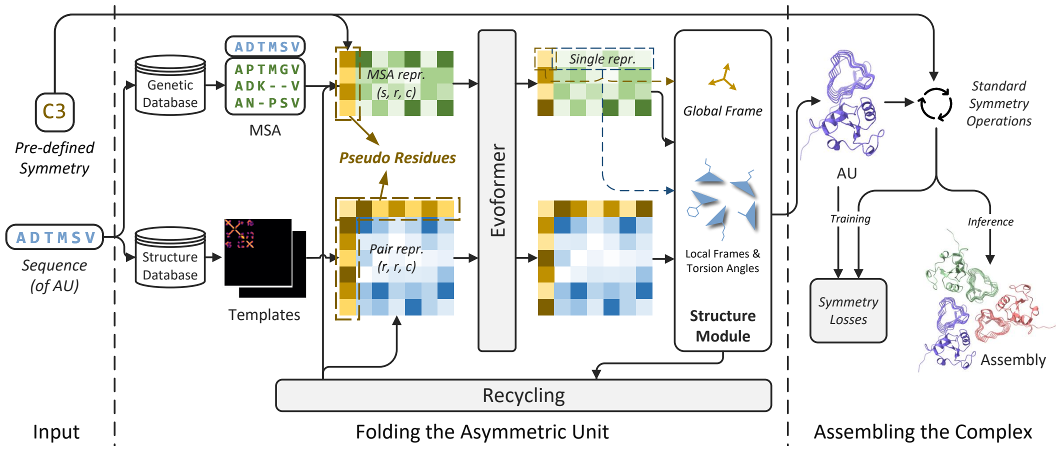

Figure 2: The pipeline of UF-Symmetry.
dock

assemblies with a circular arrangement of identical (or homologous) proteins closely packed around a water-filled pore through the plane of the lipid bilayer. Solving the structures of these symmetric assemblies yields great importance to understanding the mechanisms of relevant biological activities.

Despite the remarkable accuracy of current deep folding models on small complexes, applying them to large oligomers is not easy, specifically because of the combination of two facts: i) these models generally require copying the sequences of the repeating subunits to capture the in-between connections, typically with the self-attention mechanism [8]; ii) the complexities of these models are generally cubic to the total length of the input sequences. Taking Uni-Fold Multimer [3], one of the fastest and most memory-efficient deep folding models as an example, an NVIDIA A100 GPU with 40GB memory allows the inference of up to $\sim$2,100 residues[3] due to memory limitation. For large oligomers, the input scales can easily go beyond this limit.

Fortunately, harnessing the intrinsic symmetry of large oligomers may help to solve the problem: as the *asymmetric units* (AUs)[4] are structurally identical in an oligomer, one can predict the AU

[3]Results may have varied now.

[4]We use the term *assembly* for a protein complex that undertake biological functions as a whole, the term *asymmetric units* (AUs) for its repeating components. Notably, AUs may contain one or more chains.

structure and then assemble the entire complex. This saves the effort of repeatedly predicting the AUs. Moreover, one can generate oligomer structures with different symmetries (or stoichiometries) by querying the model with different symmetry groups.

Following this motivation, we propose **UF-Symmetry (Uni-Fold Symmetry)**, which predicts protein complex structures based on the AU sequences and a pre-specified symmetry group. UF-Symmetry learns to *fold* the AU and to *assemble* the oligomer in an end-to-end manner. To briefly summarize, the model modifies the pipeline of Uni-Fold Multimer to fold the AUs with a newly introduced *pseudo residue*, which encodes the given symmetry type and learns to correctly pose the AU in a standard frame. Defined by the input symmetry group, a group of standard symmetry operations is then applied to the AU to compose the final assembly. To efficiently supervise the process, we further reduce the complexity of the structure losses by merging the identical intra-AU terms. This also allows the model *not* to explicitly generate the assembly structures during training.

By harnessing symmetry, UF-Symmetry reduces the model complexity from $O(K^3N^3)$ to $O(N^3)$, where K is the number of AUs in the assembly and N is the AU size. Notably, this new complexity is invariant to the number of AUs in the assembly, which not only allows the model to be applied on larger oligomers, but also significantly accelerates the inference process. We also evaluated the accuracies of UF-Symmetry and baselines on 416 recently released protein multimers in PDB. AlphaFold-Multimer and Uni-Fold Multimer failed on 11.1% cases due to out-of-memory (OOM) errors, while UF-Symmetry reduces this rate to 5.8%. In commonly successful cases, UF-Symmetry displayed approaching accuracy, falling behind AlphaFold-Multimer of $\sim$2% TM-Score. We reckon the performance drop as a necessary trade-off for larger model capacity. Further analysis shows that despite the drop of average performance, UF-Symmetry is yet more competent on a considerable proportion of specific cases. Therefore, using UF-Symmetry in complement to Uni-Fold Multimer well balances the model accuracy and flexibility.

2 Preliminaries

Protein Symmetry All amino acids but glycine have chirality over the α carbon, and nature dominantly prefers L-amino acids to D-ones. Consequently, natural proteins adopt only enantiomorphic symmetries and disallow mirror and inversion ones. In this paper, the Schönflies symbols are used to denote symmetry groups, where Cx is for x-fold cyclic groups, Dx for x-fold dihedral groups, T, O, I for tetrahedral, octahedral and icosahedral groups respectively, and H for helical structures. In UF-Symmetry, we focus on proteins with finite symmetry groups (C, D, I, O, T), and leave helical structures as future work.

Notations We use $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^T \in \mathbb{R}^{n\times 3}$ to denote the structure of an AU, where $\boldsymbol{x}$s are the 3D coordinates of all heavy atoms (or C$_\alpha$ atoms, defined by the context). $\mathcal{S} = \{\boldsymbol{X}_i\}$ denotes an assembly as a set of AU structures. A symmetry operation $O = (\boldsymbol{R}, \boldsymbol{t})$ is a transformation on the 3D euclidean space, consisting of a rotation matrix $\boldsymbol{R} \in \mathrm{SO}(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$. The operation can be applied to a point $\boldsymbol{x}$, an AU structure $\boldsymbol{X}$, or composed with another operation $O' = (\boldsymbol{R}', \boldsymbol{t}')$ as

$$O \circ \boldsymbol{x} = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t} \tag{1}$$
$$O \circ \boldsymbol{X} = \boldsymbol{X}\boldsymbol{R}^T + \boldsymbol{1}_n\boldsymbol{t}^T \tag{2}$$
$$O \circ O' = (\boldsymbol{R}\boldsymbol{R}', \boldsymbol{R}\boldsymbol{t}' + \boldsymbol{t}) \tag{3}$$

A symmetry group is a set of symmetry operations $\mathcal{G} = \{O_i\}$ which is closed under composition. An assembly *conforms to* a symmetry group $\mathcal{G}$ if for any $\boldsymbol{X} \in \mathcal{S}$ and $O \in \mathcal{G}$, one has $O \circ \boldsymbol{X} \in \mathcal{S}$.

Deep Folding Models The backbone model of UF-Symmetry is originally inspired by AlphaFold [1]. Taking the amino acid sequence, homologous sequences and solved homologous structures as inputs, AlphaFold directly predicts the three-dimensional coordinates of all atoms in a protein. In AlphaFold, an *Evoformer* encodes the inputs with the attention mechanism, and a *structure module* decodes the local frames and torsion angles of the protein residues. The *Frame Aligned Point Error* (FAPE) loss is used to supervise the output structure by averaging the structure errors superposed by each residue. The *violation loss* is used to encourage the correct peptide bond geometry and to punish steric clashes. AlphaFold-Multimer [2] and Uni-Fold Multimer [3] further extended

AlphaFold to protein complexes by slightly modifying the system and retraining it on multimeric protein structures in PDB. The backbones of these models are hardly modified, whereas an MSA pairing strategy is introduced to build cross-chain genetics. An additional *chain center-of-mass* loss is also introduced to teach the model the global placement of the chains.

3 Method

Figure 2 shows the pipeline of UF-Symmetry. Given the input AU sequences and a symmetry type, the model first conduct the homology search on sequence and structure databases. The obtained MSAs and templates are then padded with the pseudo residue and encoded by the Evoformer. The structure module then decodes the local frames of the AU residues and transforms them into the global frame learned by the pseudo residue. In inference, a group of standard symmetry operations are applied to the AU structure to form the final assembly. In training, reduced structure-related losses (symmetry losses) are directly calculated with the AU structure and the standard symmetry group.

3.1 Folding the Asymmetric Unit

The AU prediction process in UF-Symmetry is modified from the structure prediction process in Uni-Fold Multimer. Unless otherwise specified, the implementation details are the same.

UF-Symmetry introduces a pseudo residue to encode the query symmetry type and to learn a global frame for the AU. The pseudo residue is a placeholder symbol that joins the calculations of the Evoformer and the structure module equally as a real residue. Intuitively, the pseudo residue plays a similar role to the *classification token* (`[CLS]`) in pre-trained language models such as BERT [9], which collects the global information of the sequences via the self-attention mechanism [8]. This is particularly important in UF-Symmetry, because we are not only interested in how the AU is folded, but also how it is globally posed and repeated to form the oligomer.

The initial features of the pseudo residue consists of the one-hot encoding of the query symmetry type (among [C1, C, D, I, O, T]), and the cosine and sine values of the rotation angles of cyclic and dihedral symmetries (e.g. $\cos\frac{2\pi}{x}$ and $\sin\frac{2\pi}{x}$ for Cx). For C1, I, O, and T, the rotation angles are defined as 0. A 4-block residual network transforms these features into the same dimension as the representations of real residues, which is then padded to the front of both the input sequences and each MSA row. For pair representations, the pseudo residue is processed as a real residue, except that we do not use relative position embedding between the pseudo residue and the real residues.

As the pseudo residue joins the calculation of the structure module, a frame, which we name as the *global frame*, is generated for the pseudo residue. When decoding the atom coordinates of the AU, UF-Symmetry transforms the local frames of residues into this global frame. Denoted in formula,

$$T_i' = (\boldsymbol{R}^{\mathrm{g}}\boldsymbol{R}_i, \boldsymbol{t}^{\mathrm{g}} + \boldsymbol{R}^{\mathrm{g}}\boldsymbol{t}_i), i = 1, 2, \cdots, n, \tag{4}$$

where $T_i = (\boldsymbol{R}_i, \boldsymbol{t}_i)$ and $T^{\mathrm{g}} = (\boldsymbol{R}^{\mathrm{g}}, \boldsymbol{t}^{\mathrm{g}})$ are the real residue frames and the global frame, respectively. T_i's are the final local frames of residues. No torsion angles and side-chain frames are generated for the pseudo residue. In the recycling process, the translation of the global frame is recycled as the C$_\beta$ coordinates of the pseudo residue.

3.2 Assembling the Complex

In order to generate the entire complex, we first define the standard symmetry groups for different symmetry types. All operations in the standard symmetry groups have zero translations ($\boldsymbol{t} = \boldsymbol{0}$), and the identical operation (O_I) is always included in the groups. For cyclic symmetries (Cx), we use the z axis (with unit vector $(0, 0, 1)$) as the rotation axis, and include the corresponding x-fold rotation group, i.e. operators that rotates the AU for $2k\pi/x,\ k = 0, 1, ..., x - 1$. For dihedral symmetries (Dx), the group is the multiplication of the standard Cx group and a 2-fold rotation group over the y axis ($(0, 1, 0)$). For the cubic symmetries, the standard operations are of more complicated forms, and are described in Section B. For asymmetric structures (C1), the group is $\{O_I\}$.

The assembling process does not introduce additional learnable parameters. By presuming that the AU (with structure $\boldsymbol{X}_1$) is properly posed in the standard frame associated with the standard symmetry groups, it simply applies all operations in the standard group of the given symmetry type to the AU structure, i.e. $\mathcal{S} = \{O_i \circ \boldsymbol{X}_1 \mid O_i \in \mathcal{G}\}$. Accordingly, the generated structure necessarily conforms to the given symmetry: for all $O_i \in \mathcal{G}$ and $\boldsymbol{X}_j \in \mathcal{S}$, one has

$$O_i \circ \boldsymbol{X}_j = (O_i \circ O_j) \circ \boldsymbol{X}_1 \in \mathcal{S}, \text{ as } O_i \circ O_j \in \mathcal{G}. \tag{5}$$

In inference, the assembling process generally takes milliseconds, negligible to the total time. Accordingly, the inference time of UF-Symmetry stays invariant to the size of the symmetry group.

3.3 Symmetry Losses

As the assembling process is differentiable to the AU structure, directly applying the losses of Uni-Fold Multimer on UF-Symmetry is feasible. However, the complexity of the Frame Aligned Point Error (FAPE) loss is squared to the total number of residues, and thus the full calculation of the losses also leads to insufficient memory for large-scale complexes. To address this issue, we reduce the structure-based losses to lower computational complexity without explicitly using the assembly structure. Intuitively, the reduction is achieved by merging the identical loss terms of AU pairs that enjoys the same symmetry operation, as is introduced by Theorem 1:

Theorem 1 *For any (finite) symmetric assembly $\mathcal{S} = \{\boldsymbol{X}_i \in \mathbb{R}^{N\times 3}\}_{i=1}^K$ and its symmetry group $\mathcal{G}$, assuming $\mathcal{F} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is a real-valued function of a pair of AUs and is invariant to the symmetry operations, i.e. $\mathcal{F}(O \circ \boldsymbol{X}_i, O \circ \boldsymbol{X}_j) = \mathcal{F}(\boldsymbol{X}_i, \boldsymbol{X}_j)$ for all $O \in \mathcal{G}$, one has*

$$\sum_{\boldsymbol{X}\in\mathcal{S}} \sum_{\boldsymbol{X}'\in\mathcal{S}} \mathcal{F}(\boldsymbol{X}, \boldsymbol{X}') = K \sum_{\boldsymbol{X}\in\mathcal{S}} \mathcal{F}(\boldsymbol{X}_1, \boldsymbol{X}) = K \sum_{O\in\mathcal{G}} \mathcal{F}(\boldsymbol{X}_1, O \circ \boldsymbol{X}_1). \tag{6}$$

Accordingly, the complexity of loss calculation is reduced from $O(K^2N^2)$ to $O(KN^2)$.

Losses that require the assembly structure include the FAPE loss, the violation loss, and the chain centre-of-mass loss. For FAPE loss, given the symmetry group $\mathcal{S}$, we calculate the whole loss as

$$\mathcal{L}_{\text{FAPE}} = \frac{1}{KN^2} \sum_{k,i,j} \left\| T_i^{-1} \circ (O_k \circ \boldsymbol{x}_j) - \tilde{T}_i^{-1} \circ (O_k \circ \tilde{\boldsymbol{x}}_j) \right\|, \tag{7}$$

where $T, \boldsymbol{x}$ are the local frames and coordinates of predicted residues, and $\tilde{T}, \tilde{\boldsymbol{x}}$ those of ground-truth. The index k is iterated over $\mathcal{G}$, and i, j over the residues inside the AU. For the violation loss, we reduce the clash terms of atom pairs between different AUs following the same idea:

$$\mathcal{L}_{\text{clash}} = \frac{1}{(K-1)N^2} \sum_{k>1,i,j} \max\left(d_{ij}^{\text{lit}} - \tau - \|\boldsymbol{x}_i - O_k \circ \boldsymbol{x}_j\|\right), \tag{8}$$

where d_{ij}^{lit} is the clashing distance between the atom types of i and j, and τ is a tolerance factor. We do not alter other terms of the loss calculated inside the AU. The between-AU term of the chain centre-of-mass loss is modified similarly as

$$\mathcal{L}_{\text{chain}} = \frac{1}{(K-1)C^2} \sum_{k>1,s,t} \frac{1}{20} \max\left(\|\boldsymbol{c}_s - O_k \circ \boldsymbol{c}_t\| - \|\tilde{\boldsymbol{c}}_s - O_k \circ \tilde{\boldsymbol{c}}_t\| + 4, 0\right), \tag{9}$$

where indices s, t are iterated over all chains (totally C chains) in the AU, and $\boldsymbol{c}$s are the coordinates of the chain centers. Notably, applying these losses requires first transforming the ground-truth ($\{T\}$ and $\tilde{\boldsymbol{X}}$) into the standard frame. Choosing any one of the AUs among the ground-truth is equivalent. The proof of Theorem 1 and the equivalency of Equations (7), (8) and (9) are in Section C.

3.4 Symmetry Augmentation

In order to encourage robustness of the model to any given symmetry group, we propose *symmetry augmentation* as a training strategy of UF-Symmetry. Specifically, with a certain probability, we replace the annotated symmetry group of the target structures with one of its subgroups. Taking symmetry group C6 as an example, we may use the subgroup C3 (chosen from C1, C2, and C3) and merge two adjacent AUs into a new one. The probability of using symmetry augmentation is 0.382, and the subgroups are uniformly sampled from all possible ones.

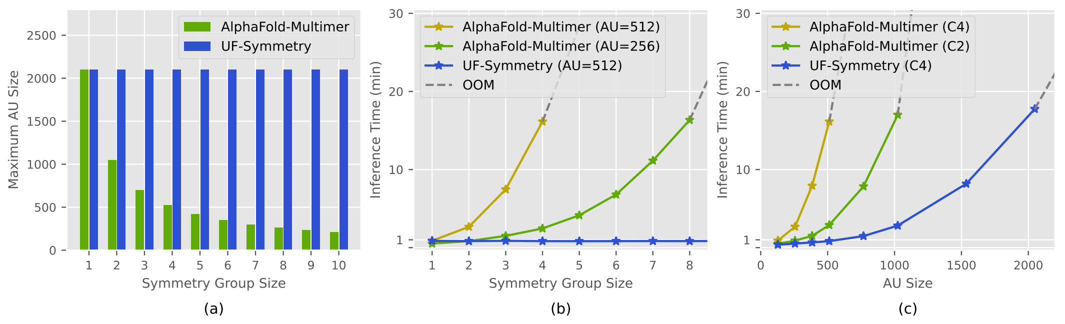

Figure 3: Capacity and inference efficiency of UF-Symmetry and AlphaFold-Multimer on an NVIDIA A100 GPU with 40GB memory. **(a)** The maximum AU sizes the models support given different symmetry group sizes. **(b) (c)** The inference time of models given different AU and symmetry group sizes. Bracketed notes beside the model names describe the fixed AU sizes or symmetry groups. Gray dashes indicate that the next data point is infeasible due to memory limitation.

4 Results

4.1 Training UF-Symmetry

Data We trained UF-Symmetry basically following Uni-Fold Multimer. The training set of UF-Symmetry is the same as Uni-Fold Multimer. Specifically, we collected a total of 181,727 PDB structures released before January 16th, 2022, including both symmetric and asymmetric ones. We used the annotated symmetry groups of the RCSB website[5], extracted the AUs, transformed them into the standard frame, and used them as labels. Notably, we only extracted the AUs of cyclic structures, while the extension to dihedral and cubic cases is natural. We used a rule-based algorithm in combination with `libmsym` [10] to extract and transform the AUs. For dihedral and cubic structures, as well as approximately 3% cyclic structures that we failed to extract or transform the AUs, we regarded them as asymmetric ones. We used the identical pipeline to search MSAs and templates to Uni-Fold Multimer, including the database versions and hyper-parameters. We also used the same self-distillation datasets, which contained approximately 360,000 monomeric structures.

Recipe We trained UF-Symmetry on 128 NVIDIA A100 GPUs with 40GB memory, each containing one training sample. The sequence(s) of AUs were cropped to 320 residues using the *contiguous cropping* strategy of [2]. We trained the model for 80,000 steps, which took us approximately one week. We tried to finetune the model under several configurations, while the improvements were minor or negative. To feed the model with more symmetric samples, we decrease the proportion of self-distillation samples from 0.5 to 0.382. Other settings were the same as Uni-Fold Multimer.

4.2 Capacity and Efficiency Benchmark

We compared the capacity to input sizes and the inference efficiency of UF-Symmetry and AlphaFold-Multimer. We used the implementation of AlphaFold in the Uni-Fold repository[6], which was proved as faster and more memory-efficient. Note that as Uni-Fold Multimer and AlphaFold-Multimer share almost identical architecture, these results also apply to Uni-Fold Multimer.

Figure 3 shows the results. We report the performances under different AU and symmetry group sizes. As AlphaFold-Multimer requires copying the sequences of AUs, the maximum AU size the model can handle is in inverse proportion to the size of the symmetry group. In addition, the inference time of AlphaFold-Multimer is also cubic to the size of the symmetry group. By contrast, UF-Symmetry has invariant capacity and inference time to different symmetry group sizes, yielding significant improvements in larger oligomers. For example, given the AU size of 512 and the symmetry group C4, the inference speed of UF-Symmetry is approximately 19.4 times to AlphaFold-Multimer.

[5]`https://www.rcsb.org/`.

[6]`https://github.com/dptech-corp/Uni-Fold`. We developed UF-Symmetry and conducted experiments on an early commit of Uni-Fold (commit-ID: `b7461325`). As the repository lately introduced plenty of optimizations, the capacity and efficiency results may have varied.

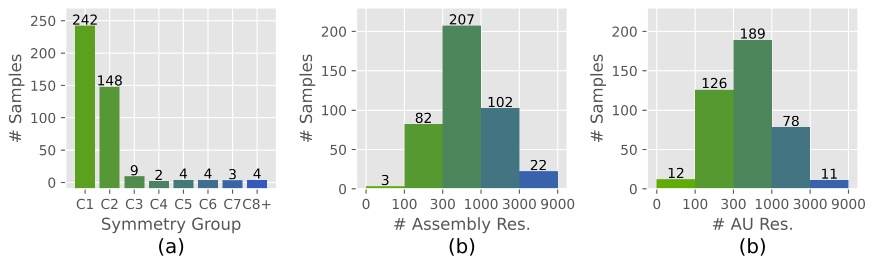

Figure 4: Statistics of the test dataset. **(a)** Number of samples of different symmetry groups. **(b) (c)** Histogram of samples with different assembly / AU sizes.

Table 1: Number of failed cases and prediction accuracies on commonly successful cases of UF-Symmetry and other baselines.

Model	Failed Cases	C_α-RMSD$_{30}$ (Å)	TM-Score
AlphaFold-Multimer	11.1% (46)	8.40	0.757
		8.26	0.755
		8.62	0.744
		8.19	0.758
		8.30	0.756
Uni-Fold Multimer	11.1% (46)	**6.98**	**0.791**
UF-Symmetry (C1)	9.4% (39)	8.77	0.760
UF-Symmetry	**5.8% (24)**	9.74	0.733

4.3 Accuracy Benchmark

Data, Metrics, and Baselines We evaluated UF-Symmetry and baseline models on recently released multimeric structures in PDB, focusing on the performances of asymmetric and cyclic structures. We collected a total of 1,181 PDB structures released between January 17th and July 14th, 2022, and kept the asymmetric and cyclic multimeric structures, i.e. the structures in Cx symmetry groups with at least 2 chains. We further filtered the structures such that all structures had resolutions below 3.5 Å and had at least one chain with less than 40% template identity, yielding a total of 416 samples. Statistics of the test samples are in Figure 4. Following Uni-Fold Multimer, we report the C_α-RMSD and TM-Scores of protein complexes. As the C_α-RMSD of entire complexes suffered from large outliers, we truncated the larger values to 30 Å, which we named as C_α-RMSD$_{30}$. These scores were calculated by aligning the entire complexes and averaging the metrics among all residues. We fed UF-Symmetry with the annotated symmetry groups from RCSB. We tested the performances of Uni-Fold Multimer, and all 5 models of AlphaFold-Multimer. We also tested UF-Symmetry (C1) by feeding UF-Symmetry with C1 symmetry and merging all AUs into one.

Results Table 1 shows the evaluation results. We report the number of failed cases due to memory limitations, and the averaged accuracy metrics on commonly successful cases (370 samples). By harnessing symmetry, UF-Symmetry succeeded on 22 more samples compared with AlphaFold-Multimer and Uni-Fold Multimer, which approximately halved the failure rate. On commonly successful samples, UF-Symmetry displayed approaching accuracy, with an approximately 2% drop on the TM-Score compared with AlphaFold-Multimer, and 6% that with Uni-Fold Multimer. Nevertheless, the performances of UF-Symmetry (C1) are better than UF-Symmetry and equivalent to AlphaFold-Multimer, slightly below Uni-Fold Multimer.

The comparison between UF-Symmetry and UF-Symmetry (C1) may help to reveal the reason of the performance drop: as we do not copy the AU sequences, the intra-AU connections cannot be captured as well as those in UF-Symmetry (C1) as well as other multimer models. This could be considered as a trade-off between complexity and accuracy. Meanwhile, the smaller cropping sizes and the lack of finetuning may account for the accuracy drop from Uni-Fold Multimer to UF-Symmetry (C1).

More detailed accuracy results are shown in Figure 5, in which we draw both TM-Score and C_α-RMSD$_{30}$ as scatter plots. Although the averaged accuracies of UF-Symmetry drops, it still attains

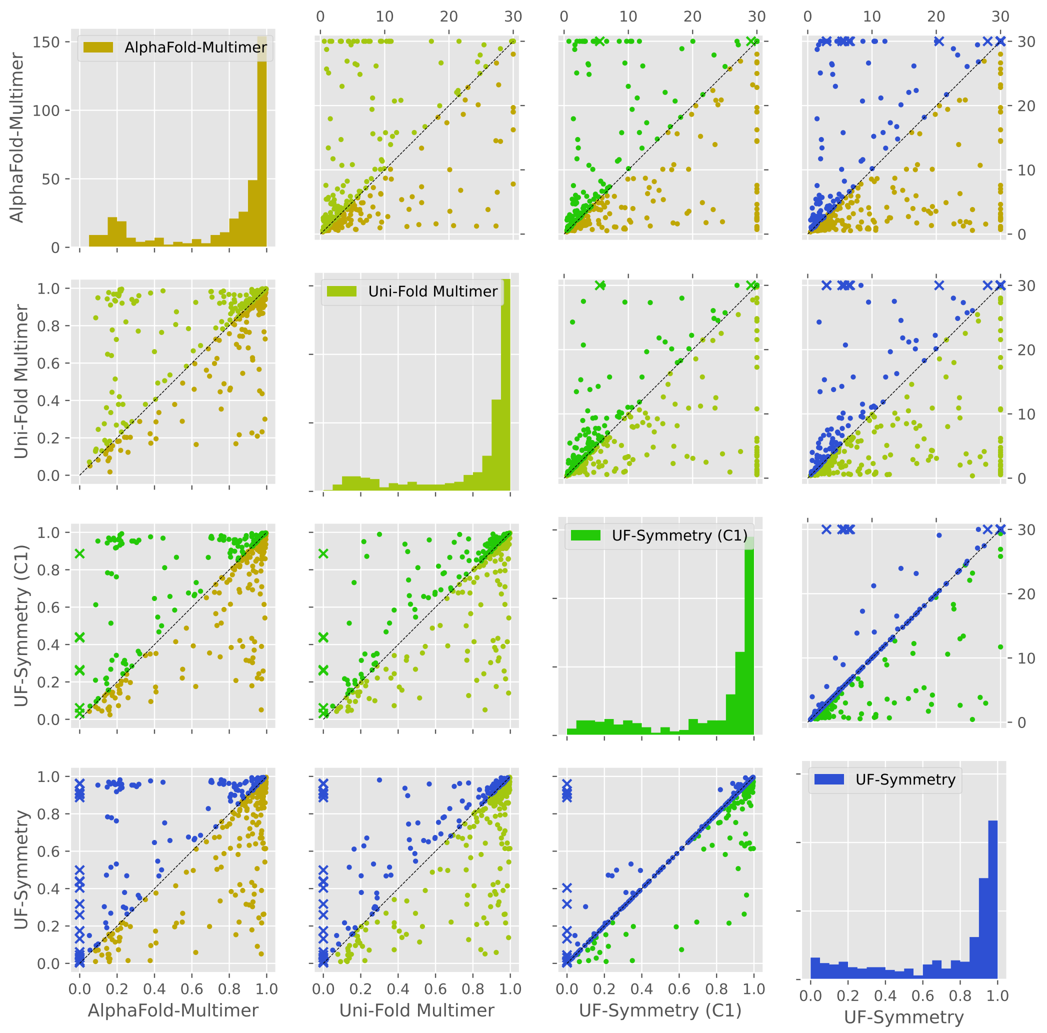

Figure 5: Scatter matrix of accuracies of UF-Symmetry and baselines. The lower triangle stores the TM-Score scatter plots, the diagonal stores the histograms of TM-Score, and the upper triangle stores the C_α-RMSD$_{30}$ scatter plots. Dots in different colors indicate that the corresponding model is better (blue for UF-Symmetry). Crosses indicate that the other model failed to predict the structure.

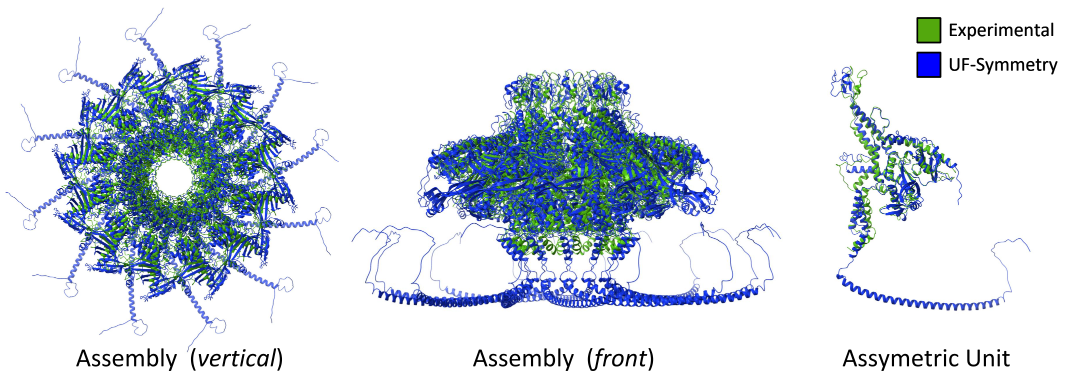

Figure 6: Predicted results of a *homo 12-mer* structure (PDB-ID: 7SYA) by UF-Symmetry. The total number of residues is 8,460. The TM-Score of the prediction is 0.922. The long helix tails in the prediction are not experimentally solved.

competitive performances on specific cases (indicated with the blue dots). Therefore, in real scenarios, one may use UF-Symmetry in complement to Uni-Fold Multimer or UF-Symmetry (C1) to achieve the best balance between flexibility and accuracy.

Case Study Figure 6 presents one of the largest targets in the test set, which is a homo 12-mer structure (PDB-ID: 7SYA) in the C12 symmetry group. We show both the assembly and the AU structures predicted by UF-Symmetry, aligned with the experimentally solved ones. As a total of 8,460 residues is deposited in the assembly, AlphaFold-Multimer, Uni-Fold Multimer and UF-Symmetry (C1) failed to predict for this target. UF-Symmetry successfully predicted the structure of the AU, and correctly assembled the complex. The model inference time for the assembly is approximately 80 seconds on an NVIDIA A100 GPU.

5 Conclusion and Future Work

In this work, we propose UF-Symmetry a fast and accurate solution to predicting the structures of large symmetric protein oligomers. UF-Symmetry reduces the need of copying the sequences of asymmetric units, thus achieving great advantages over model scalability and efficiency, yet with a slight drop of average accuracy.

UF-Symmetry is currently an on-going work, with straight-forward upcoming extensions in i) design of model confidence; ii) training on dihedral and cubic symmetry groups; iii) various optimizations inherited from Uni-Fold, and iv) larger evaluation sets. In addition, UF-Symmetry currently relies on pre-specified symmetry groups, and hence another promising extension is to automatically extract the possible symmetry groups given the AU sequences.

References

[1] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021.

[2] Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, Olaf Ronneberger, Sebastian Bodenstein, Michal Zielinski, Alex Bridgland, Anna Potapenko, Andrew Cowie, Kathryn Tunyasuvunakool, Rishub Jain, Ellen Clancy, Pushmeet Kohli, John Jumper, and Demis Hassabis. Protein complex prediction with alphafold-multimer. *bioRxiv*, 2022.

[3] Ziyao Li, Xuyang Liu, Weijie Chen, Fan Shen, Hangrui Bi, Guolin Ke, and Linfeng Zhang. Uni-fold: An open-source platform for developing protein folding models beyond alphafold. *bioRxiv*, 2022.08.04.502811, 2022.

[4] M. Torchala, I. H. Moal, R. A. Chaleil, J. Fernandez-Recio, and P. A. Bates. SwarmDock: a server for flexible protein-protein docking. *Bioinformatics*, 29(6):807–809, 2013.

[5] D. Kozakov, D. R. Hall, B. Xia, K. A. Porter, D. Padhorny, C. Yueh, D. Beglov, and S. Vajda. The ClusPro web server for protein-protein docking. *Nature Protocol*, 12(2):255–278, 2017.

[6] Stephen K Burley, Helen M. Berman, Gerard J. Kleywegt, John L. Markley, Haruki Nakamura, and Sameer Velankar. Protein Data Bank (PDB): The single global macromolecular structure archive. *Methods of Molecular Biology*, 1607:627–641, 2017.

[7] David S. Goodsell and Arthur J. Olson. Structural symmetry and protein function. *Annual Review of Biophysics and Biomolecular Structure*, 29:105–153, 2000.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[10] Marcus Johansson and Valera Veryazov. Automatic procedure for generating symmetry adapted wavefunctions. *Journal of Cheminformatics*, 9(1):1–8, 2017.

A Statistics of Protein Symmetries in PDB

We made statistics on the 181,727 protein structures in the Protein Data Bank (PDB) [6] released before Janurary 16th, 2022. We used the symmetry symbols annotated by the RCSB website[7]. Approximately 54% PDB structures are multimeric structures, among which approximately 72% are symmetric. Table 2 shows the detailed statistics.

Table 2: Statistics of symmetries of PDB structures

Type	Symmetry	Number of structures
Monomers	–	83,392
Multimers	Asymmetric	27,470
	C2	45,496
	C3	6,037
	C4	1,736
	C5	893
	C6	639
	Larger cyclic	587
	D2	8,571
	D3	2,577
	D4	815
	D5	302
	D6	191
	Larger dihedral	239
	Icosahedral	1,182
	Octahedral	544
	Tetrahedral	475
	Helical	581
	All	98,335
Total	–	181,727

B Standard Symmetry Operations of Cubic Symmetries

The group of the standard operations of the icosahedral symmetry is the multiplication of a 5-fold rotation group of the z axis ($(0,0,1)$), a 3-fold rotation group of axis $(0, -\sqrt{\frac{1}{3}}, \sqrt{\frac{2}{3}})$, a 2-fold rotation group of axis $(\frac{1}{3}, -\frac{2\sqrt{2}}{3}, 0)$, and a 2-fold rotation group of $(0, -\sqrt{\frac{2}{3}}, \sqrt{\frac{1}{3}})$.

The group of the standard operations of the octahedral symmetry is the multiplication of a 4-fold rotation group of the z axis ($(0,0,1)$), a 3-fold rotation group of axis $(\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}})$, and a 2-fold rotation group of the x axis $(1,0,0)$.

The group of the standard operations of the tetrahedral symmetry is the multiplication of a 3-fold rotation group of axis $(\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}})$, a 2-fold rotation group of the y axis ($(0,1,0)$, and a 2-fold rotation group of the x axis $(1,0,0)$.

C Proofs of Theorem 1 and Equations (7), (8) and (9)

To prove Theorem 1, we first introduce Lemma 1:

Lemma 1 *For any finite group $\mathcal{G}$ with $|\mathcal{G}| = K$ and any real-valued function $\mathcal{F} : \mathcal{G} \rightarrow \mathbb{R}$,*

$$\sum_{O' \in \mathcal{G}} \mathcal{F}[O \circ O'] = \sum_{O \in \mathcal{G}} \mathcal{F}[O] = \sum_{O' \in \mathcal{G}} \mathcal{F}[O^{-1} \circ O'] \tag{10}$$

[7]https://www.rcsb.org/

The proof of Lemma 1 is simple by noticing that for any $O \in \mathcal{G}$, we have $O^{-1} \in \mathcal{G}$, and applying O on all elements in $\mathcal{G}$ generates a permutation of $\mathcal{G}$, i.e. $O \circ \mathcal{G} = O^{-1} \circ \mathcal{G} = \mathcal{G}$.

Assuming the assembly $\mathcal{S}$ conforms to the symmetry group $\mathcal{G}$, Theorem 1 can then be proved as

$$\begin{aligned}
&\sum_{\boldsymbol{X}\in\mathcal{S}} \sum_{\boldsymbol{X}'\in\mathcal{S}} \mathcal{F}(\boldsymbol{X}, \boldsymbol{X}') \\
=&\sum_{O\in\mathcal{G}} \sum_{O'\in\mathcal{G}} \mathcal{F}(O \circ \boldsymbol{X}_1, O' \circ \boldsymbol{X}_1) && (11)\\
=&\sum_{O\in\mathcal{G}} \sum_{O'\in\mathcal{G}} \mathcal{F}(\boldsymbol{X}_1, (O^{-1} \circ O') \circ \boldsymbol{X}_1) && (12)\\
=&K \sum_{O\in\mathcal{G}} \mathcal{F}(\boldsymbol{X}_1, O \circ \boldsymbol{X}_1) && (13)\\
=&K \sum_{\boldsymbol{X}\in\mathcal{G}} \mathcal{F}(\boldsymbol{X}_1, \boldsymbol{X}) && (14)
\end{aligned}$$

where Equation (11) holds as $\mathcal{S}$ conforms to symmetry group $\mathcal{G}$, Equation (12) holds under the condition that $\mathcal{F}$ is invariant under 3D transformations, and Equation (13) is derived from Lemma 1.

The equivalency of FAPE loss follows

$$\begin{aligned}
&\mathcal{L}_{\text{FAPE}}(\{T\}, \boldsymbol{X}, \{\tilde{T}\}, \tilde{\boldsymbol{X}}; \mathcal{S}) \\
=&\frac{1}{(KN)^2} \sum_{m,n,i,j} \left\| (O_m \circ T_i)^{-1} \circ (O_n \circ \boldsymbol{x}_j) - (O_m \circ \tilde{T}_i)^{-1} \circ (O_n \circ \tilde{\boldsymbol{x}}_j) \right\| && (15)\\
=&\frac{1}{(KN)^2} \sum_{m,n,i,j} \left\| T_i^{-1} \circ (O_m^{-1} \circ O_n \circ \boldsymbol{x}_j) - \tilde{T}_i^{-1} \circ (O_m^{-1} \circ O_n) \circ \tilde{\boldsymbol{x}}_j \right\| && (16)\\
=&\frac{1}{KN^2} \sum_{k,i,j} \left\| T_i^{-1} \circ (O_k \circ \boldsymbol{x}_j) - \tilde{T}_i^{-1} \circ (O_k \circ \tilde{\boldsymbol{x}}_j) \right\| && (17)
\end{aligned}$$

The equivalency of *between-AU* clash loss follows

$$\begin{aligned}
\mathcal{L}_{\text{clash}}(\boldsymbol{X}; \mathcal{S}) =& \frac{1}{K(K-1)N^2} \sum_{m\neq n,i,j} \mathcal{D}\left(\|O_m \circ \boldsymbol{x}_i - O_n \circ \boldsymbol{x}_j\|; i, j\right) && (18)\\
=& \frac{1}{K(K-1)N^2} \sum_{m\neq n,i,j} \mathcal{D}\left(\left\|\boldsymbol{x}_i - (O_m^{-1} \circ O_n) \circ \boldsymbol{x}_j\right\|; i, j\right) && (19)\\
=& \frac{1}{(K-1)N^2} \sum_{k>1,i,j} \mathcal{D}\left(\|\boldsymbol{x}_i - O_k \circ \boldsymbol{x}_j\|; i, j\right) && (20)
\end{aligned}$$

where $\mathcal{D}(r; i, j) = \max\left(d_{ij}^{\text{lit}} - \tau - r, 0\right)$ is the kernel function of the loss.

The equivalency of *between-AU* chain centre-of-mass follows

$$\begin{aligned}
&\mathcal{L}_{\text{centre}}(\boldsymbol{X}, \tilde{\boldsymbol{X}}; \mathcal{S}) \\
=&\frac{1}{K(K-1)C^2} \sum_{m\neq n,s,t} \mathcal{D}\left(\|O_m \circ \boldsymbol{c}_s - O_n \circ \boldsymbol{c}_t\| - \|O_m \circ \tilde{\boldsymbol{c}}_s - O_n \circ \tilde{\boldsymbol{c}}_t\|\right) && (21)\\
=&\frac{1}{K(K-1)C^2} \sum_{m\neq n,s,t} \mathcal{D}\left(\left\|\boldsymbol{c}_s - (O_m^{-1} \circ O_n) \circ \boldsymbol{c}_t\right\| - \left\|\tilde{\boldsymbol{c}}_s - (O_m^{-1} \circ O_n) \circ \tilde{\boldsymbol{c}}_t\right\|\right) && (22)\\
=&\frac{1}{(K-1)C^2} \sum_{k>1,s,t} \mathcal{D}\left(\|\boldsymbol{c}_s - O_k \circ \boldsymbol{c}_t\| - \|\tilde{\boldsymbol{c}}_s - O_k \circ \tilde{\boldsymbol{c}}_t\|\right) && (23)
\end{aligned}$$

where $\mathcal{D}(r) = \frac{1}{20} \max(r + 4, 0)$.