

Improved fragment sampling for ab initio protein structure prediction using deep neural networks

Author

Wang, T, Qiao, Y, Ding, W, Mao, W, Zhou, Y, Gong, H

Published

2019

Journal Title

Nature Machine Intelligence

Version

Accepted Manuscript (AM)

DOI

[10.1038/s42256-019-0075-7](https://doi.org/10.1038/s42256-019-0075-7)

Rights statement

© 2019 Nature Publishing Group. This is the author-manuscript version of this paper. Reproduced in accordance with the copyright policy of the publisher. Please refer to the journal website for access to the definitive, published version.

Downloaded from

<http://hdl.handle.net/10072/395265>

Griffith Research Online

<https://research-repository.griffith.edu.au>

Improved fragment sampling for *ab initio* protein structure prediction using deep neural networks

Tong Wang^{1,2}, Yanhua Qiao^{1,2}, Wenze Ding^{1,2}, Wenzhi Mao^{1,2}, Yaoqi Zhou^{3,*}
and Haipeng Gong^{1,2,*}

¹MOE Key Laboratory of Bioinformatics, School of Life Sciences, Tsinghua University, Beijing 100084, China.

²Beijing Advanced Innovation Center for Structural Biology, Tsinghua University, Beijing 100084, China.

³Institute for Glycomics and School of Information and Communication Technology, Griffith University, Gold Coast, QLD 4222, Australia.

*To whom correspondence should be addressed:

Email: hgong@tsinghua.edu.cn (H. G.) and yaoqi.zhou@griffith.edu.au (Y. Z.)

A typical approach for predicting unknown native structures of proteins is to assemble the fragment structures extracted from known structures. The quality of these extracted fragments that are used to build protein-specific fragment libraries can determine the success or failure of sampling near-native conformations. Here, we show how a high-quality fragment library can be built using deep contextual learning techniques. Our algorithm, called DeepFragLib, employs bidirectional long short-term memory recurrent neural networks with knowledge distillation for initial fragment classification, followed by an aggregated residual transformation network with cyclically dilated convolution for detecting near-native fragments. DeepFragLib improves the position-averaged proportion of near-native fragments by 12.2% over existing methods and, consequently, produces better near-native structures for 72.0% of the free-modeling domain targets tested when integrated with Rosetta. DeepFragLib is fully parallelized and available for use in conjunction with structure prediction programs.

Living organisms rely on different proteins to carry out separate functions in every biological process. This wide variety of functions is enabled by the ability of the proteins to fold into different structures, prescribed by their amino-acid sequences. The cost of experimentally determining structures for the hundreds of millions of known proteins is prohibitive. It is therefore necessary to computationally predict protein structures directly from their sequences. *Ab initio* protein structure prediction is an approach that locates the native structure by intensively searching the protein conformational space¹⁻⁴. How to effectively explore the enormous degrees of freedom of the protein conformational space is one of the most challenging problems in computational structural biology^{5,6}.

One approach to this problem is fragment assembly, which speeds up local structure sampling by repeatedly forcing contiguous amino acid residues (called fragments) of the target protein to adopt the conformations of template fragments extracted from known structures⁷. The fragment conformational space built from known structures is believed to be nearly complete, as a collection of more than one thousand non-redundant proteins could be successfully reconstructed (with the correct topology) using a 4- to 16- residue non-homologous fragment database⁸. Although advances have been made in contact-assisted protein folding⁹⁻¹¹, fragment assembly remains the most commonly used approach in *ab initio* protein structure prediction¹²⁻¹⁴. This is exemplified by Rosetta^{15,16}, which uses 3-mer and 9-mer alternatively fragments for conformational sampling.

In fragment-assembly-based simulations, a fragment library is first constructed to

facilitate conformational sampling. This library lists eligible template fragments at each position of the target protein identified based purely on the sequence information. The quality of this fragment library is fundamental to the efficiency and accuracy of subsequent protein folding simulations⁷. The proportion of near-native fragments in a fragment library (referred as “precision”) and the proportion of protein positions covered by at least one near-native fragment (referred as “coverage”) are the most frequently adopted metrics for assessing the quality of fragment libraries. However, it is noticeable that “precision” becomes biased for a fragment library that contains a varying number of fragments at different positions, particularly when many fragments are enriched at a limited number of positions. To address this problem, we propose the “position-averaged precision”, which calculates the proportions of near-native fragments of each individual position and then averages over all positions. The combination of these three metrics may present a more comprehensive quality assessment. In addition, fragment-assembly-based simulations typically require a sufficient number of fragments at every position so that complementary and compensatory fragments at different positions can be located and assembled for effective conformational search in protein folding simulations. As an example, the AbinitioRelax protocol of Rosetta employs 200 fragments per position as generated by NNMake¹⁷ for accurate protein structure prediction. Therefore, we propose “depth”, the mean number of recruited fragments, as an additional measure.

Besides NNMake, employed in Rosetta, several algorithms have been developed to optimize the construction of fragment libraries^{7,18-21}, among which LRfragLib⁷ proposed

by us and the recently developed Flib-Coevo²¹ were shown to perform best²¹. LRFragment identifies near-native fragments with an ensemble of logistic regression models but adopts a sophisticated greedy algorithm in the enrichment step to guarantee the “depth”. This greedy algorithm may recruit highly unbalanced numbers of fragments at different positions and ultimately bias the estimation of precision. The latest Flib-Coevo improves over its earlier version by utilizing the map of predicted residue-residue contacts in the enrichment steps. Although reported to reach a slightly higher precision than LRFragment, Flib-Coevo sacrifices “depth”, which could limit its utility in protein structure prediction.

Recently, deep learning techniques have led to significant advances in protein structure prediction. For example, Raptor-X adopted residual convolutional networks to improve 2D contact-map prediction in recent critical assessment of techniques for protein structure prediction (CASP) competitions^{12,22,23}; SPOT-1D & SPOT-Contact utilized an ensemble of recurrent and convolutional networks to enhance the prediction accuracy for secondary structures, backbone angles and residue contacts^{11,24}; and AlphaFold greatly impacted the field of protein structure prediction in the latest CASP13 competition by designing various neural networks for fragment generation, angle and contact prediction and structural modeling with geometric constraints¹⁰. In this work, we developed an algorithm to further improve the quality of fragment libraries, based on the Long-Short-Term Memory (LSTM)²⁵ network and the aggregated residual transformation network ResNeXt²⁶. As a variant of the Bidirectional Recurrent Neural Network (BRNN)²⁷, the Bidirectional LSTM (Bi-LSTM) network has proved to be useful for protein structural predictors, since the

amino acid sequence could be processed as time steps²⁸. Furthermore, the ResNeXt²⁶ architecture based on ResNet²⁹ but with fewer hyper-parameters, has increased the accuracy of many common tasks by aggregating a set of transformations with the same topology in repeated building blocks to facilitate extracting high-level features in an ultra-deep neural network. Our algorithm, DeepFragLib, utilizes these cutting-edge techniques to construct a hierarchical architecture containing multiple classification models and a regression model. The scores of these models are used to filter all template fragments to make a reduced set (of size ~50-200) at each position for a target protein. Specifically, in the classification models, we adopted Bi-LSTM layers to process sequence information, followed by fully connected layers and an output node to differentiate near-native from decoy fragments. These models were further compressed and accelerated by knowledge distillation³⁰. The activations from the Bi-LSTM layers and the final outputs of the classification models as well as predicted residue-residue contact map were then fed into one ResNeXt-based regression model to directly predict root-mean-square distance (RMSD) values (i.e. structural dissimilarity) from native fragments. In this regression model, we proposed a cyclic dilation to expand the receptive field to the whole sequence in a single convolutional layer³¹. Given the prediction score by the regression model, we further developed a selection strategy for the recruitment of high-quality template fragments at each position of the target protein. When evaluating on all free-modeling targets in independent test CASP11-13 sets, DeepFragLib improved the quality of fragment library over the other state-of-the-art algorithms including NNMake, LRFragLib, and Flib-

Coevo. Moreover, when integrated with Rosetta for testing the conformational sampling, DeepFragLib significantly outperformed the other algorithms in sampling high-quality protein structure models. DeepFragLib is fully parallelized and takes only 20-30 minutes to build the fragment library for a 100-residue target sequence. An online server of DeepFragLib is available at <http://structpred.life.tsinghua.edu.cn/DeepFragLib.html>.

Results

The overall flowchart of DeepFragLib is shown in Fig. 1.

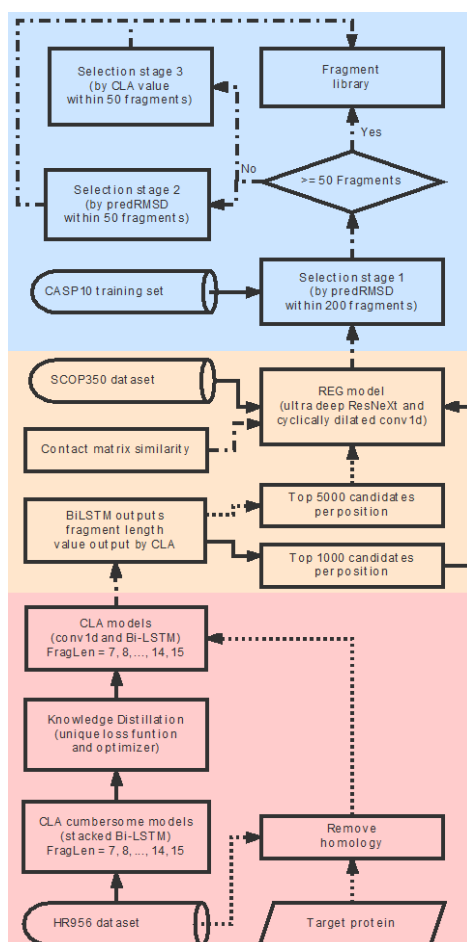


Fig. 1. The overall flowchart of DeepFragLib. DeepFragLib is composed of a classification module, a regression module, and a fragment selection module, as shown in the pink, tan and cyan panels,

respectively. The parallelogram box, cylinder boxes, diamond box and rectangle boxes indicate data, dataset, judgement and process, respectively. The solid lines and dotted lines indicate the training and inference processes, respectively, while the dashed lines refer to the processes shared by training and inference.

The algorithm consists of three sequential modules: a classification module that filters out most unqualified fragments, a regression module that predicts the RMSD values and thus identifies near-native fragments (i.e. low RMSDs) from the remaining ones, and a fragment selection module that comprehensively utilizes the scores of the previous two modules to accomplish the final fragment recruitment. The classification module contains multiple Bi-LSTM-based networks, each trained for one fragment length (from 7 to 15 residues). These large classification models (referred as “CLA cumbersome models”, where “CLA” stands for classification and “cumbersome” indicates the large model sizes) were further compressed by knowledge distillation to tiny models (referred as “CLA models”) for computational efficiency (Supplementary Fig. S1). In the regression module, information from CLA models as well as predicted residue contacts is taken by one regression model (referred as “REG model”) that adopts the ResNeXt-like architecture with cyclical dilation (Supplementary Fig. S2) to predict RMSD values. The fragment selection module utilizes a three-stage selection strategy to obtain 50 to 200 high-quality fragments at every position, which altogether constitute the fragment library for the target protein. Here, we utilized different neural network architectures for the CLA and REG models with the hope that their combination will maximize the overall performance. We did not test the performance if the

same network architectures were employed for both models because of the time-consuming nature of optimizing hyper-parameters.

Training and testing the classification module.

The classification module was trained, five-fold cross-validated and then independently tested using fragments from a set of 956 high-resolution nonredundant proteins (HR956).

We first examined the impact of training with different ratios of positive to negative samples because positive samples are greatly outnumbered by negative samples (approximately 1:8000) and, thus, under-sampling of negatives is required, similar to our previous work⁷. We used F1-score, the harmonic mean of precision and recall, to evaluate the performance on the independent test set. As shown in Supplementary Fig. S3, the performance of the cumbersome CLA model for 10-residue fragments is nearly independent of the ratio of positive-to-negative samples for a few ratios we examined. Thus, we trained all CLA cumbersome models at the 1:1 ratio. Supplementary Fig. S4 compares the F1-scores of 5-fold cross validation with those of the independent test for the models of different fragment lengths. The essentially identical results between cross validation and independent test indicate the robustness of our models.

In this classification module, the CLA cumbersome models not only adopt more advanced network architectures, but also include a number of new features (e.g., predicted solvent accessible surface area, backbone torsion angles and C α -atom-based angles) that were not employed in the logistic regression (LR) models adopted in our previous program LRfragLib. Supplementary Fig. S5 shows the change in performance after removing each

feature type. It is clear that all features contribute positively to the overall performance. The most important contribution to the performance of our CLA models, however, is not made by new features but by the deep learning neural networks. As shown in Supplementary Fig. S6, our previous LR models can be improved somewhat by new features but the dominant contribution to the improvement is from the introduction of new machine-learning models.

All CLA cumbersome models were distilled for model compression to accelerate the computational speed. After knowledge distillation, the CLA model becomes nearly three times faster although the F1-score is decreased by 2% (Supplementary Table S1). However, directly training models of the same level of compression without distillation will reduce the F1-score further by another 2%. Thus, the distilled CLA models have the best compromise in term of accuracy and speed.

Training and testing the regression module.

The classification module is followed by a regression module, which was trained, validated and tested on an independent SCOP350 set in order to predict the real values of RMSD for detecting near-native fragments. Prior to training, we analyzed the RMSD distributions of samples with different lengths in the SCOP350 set (Supplementary Fig. S7). Since the fragment RMSD rarely exceeds 5.0 Å, the output of the REG model is set to [0, 5.0]. On the independent test set, the Pearson correlation coefficient (PCC) between the predicted RMSDs and the true values as well as the mean absolute error (MAE) of the predictions are 0.735 and 0.661 Å, respectively, supporting the prediction power of the REG model.

We then analyzed the distributions of predicted RMSD values on five groups of fragments with different ranges of ground-truth RMSD values: [0, 1), [1, 2), [2, 3), [3, 4), and [4, 5]. As shown in Supplementary Fig. S8, the distributions of predicted RMSD values shift right in turn, along with an increase in ground-truth RMSD values. Moreover, the group of the smallest range of ground-truth RMSDs is well separated from the others, indicating that the REG model is capable of identifying near-native fragments. Subsequently, we evaluated the fraction of high-quality fragments (i.e. with RMSD < 2.0 Å) among all fragments selected by a specific cutoff for the predicted RMSD values (Supplementary Fig. S9). The high fractions at all tested cutoff values (80% even at the 2.0 Å cutoff) support the use of predicted RMSDs as the principal fragment selection criterion in the following fragment selection module.

Quality assessment of fragment libraries

The output from the regression module above is employed for fragment selection by the third module, which was trained using the CASP 10 dataset (See Methods). Fragment libraries were then constructed for test sets CASP 11, 12 and 13 and evaluated against those of NNMake, LRFragmentLib and Flib-Coevo.

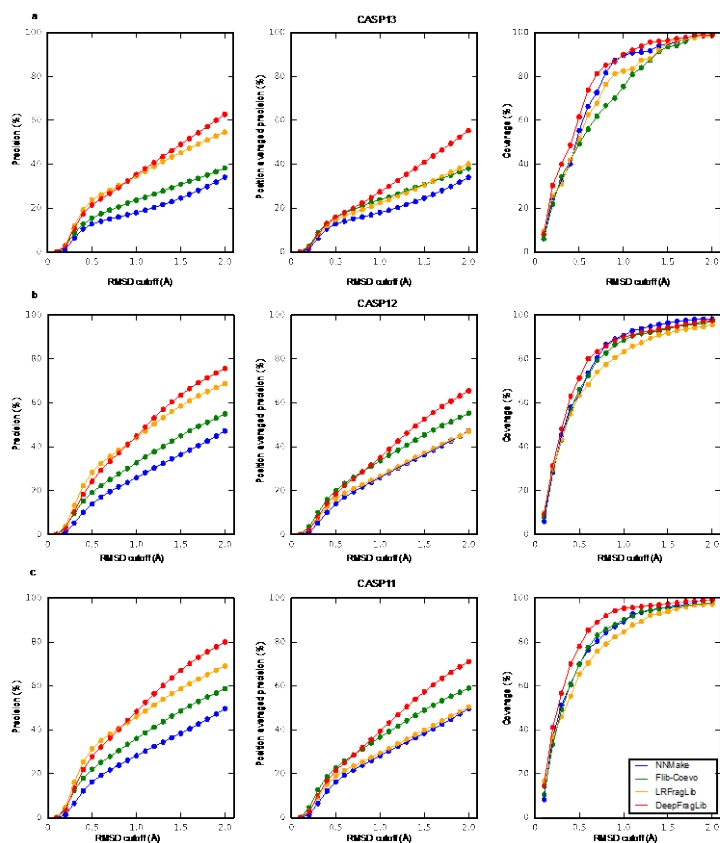


Fig. 2. Quality assessment of fragment libraries. Quality of the fragment libraries produced by DeepFragLib (red), LRFRagLib (orange), Flib-Coevo (green) and NNMake (blue) is evaluated in the CASP13 (a), CASP12 (b) and CASP11 (c) test protein sets, respectively, in term of precision, position-averaged precision and coverage as a function of RMSD cutoff for defining near-native fragments.

Fragments from proteins homologous to targets in these CASP sets were excluded from library construction based on the homology removal method applied in DeepFragLib (with $E\text{-value} < 0.1$ in PSI-BLAST³²) and in other algorithms (with their default rules), respectively. Precision, position-averaged precision and coverage were calculated at different RMSD cutoffs from 0.1 to 2.0 Å as shown from left to right panels in Fig. 2. DeepFragLib and LRFRagLib both achieve much higher precision than the other two algorithms, with DeepFragLib yielding the best performance. More importantly, with

respect to position-averaged precision, DeepFragLib outperforms all other methods by a large margin, followed by Flib-Coevo, LRFRagLib and NNMake. For example, at the cutoff of 2.0 Å, DeepFragLib improves the precision and position-averaged precision over the second-best method by 8.7% and 12.2%, respectively (Table 1).

Table 1. Precision and position-averaged precision values of fragment libraries produced by DeepFragLib, NNMake, LRFRagLib and Flib-Coevo at the RMSD cutoff of 2.0 Å.

Protein set	DeepFragLib (%)	NNMake (%)	LRFRagLib (%)	Flib-Coevo (%)
CASP10	78.19/69.17	49.69/49.69	69.30/49.17	58.74/58.95
CASP11	80.00/71.10	49.70/49.70	69.10/50.44	58.77/59.05
CASP12	75.63/65.50	47.26/47.26	68.83/47.07	55.05/55.26
CASP13	62.73/55.33	34.07/34.07	54.67/40.15	38.28/38.25
CASP11-13	75.06/65.91	45.86/45.86	66.39/47.18	53.52/53.71

Values of precision and position-averaged precision are separated by slash. The highest value in each protein set is highlighted in bold.

The higher performance of DeepFragLib in both precision and position-averaged precision indicate its ability to locate better near-native fragments across different positions of the target sequence to assist conformational sampling in protein folding simulations. By comparison, LRFRagLib has a comparable precision to DeepFragLib but with significantly poorer position-averaged precision than DeepFragLib. This indicates that the fractions of near-native fragments generated by LRFRagLib are highly unbalanced, biasing the

precision estimation. Fig. 2 also shows that the four algorithms all achieve nearly 100% coverage at 2.0 Å cutoff and thus guarantee the possibility of finding near-native structures in the fragment assembly simulations. However, DeepFragLib is able to achieve a higher coverage at a lower cutoff in all three CASP test sets. Similar results could be observed in performance comparison on the CASP10 training set (Supplementary Fig. S10). Similar levels of performance in training and test confirm the robustness of our method for unseen proteins.

Fragment quality in secondary structure regions

It is of interest to examine the quality of fragments obtained in the region of different secondary structure types. To do that, we divided positions of a target sequence into three classes: mainly helix, mainly strand, and mainly coil. Specifically, a position is labeled as “mainly helix” or “mainly strand” if at least half residues in the native segment starting from this position are helical or strand residues. The remaining positions are labeled as “mainly coil”. Precision values for fragments at positions within respective secondary structure classes were then evaluated in the three CASP test sets (Fig. 3 and Supplementary Figs. S11 and 12) and CASP10 training set (Supplementary Fig. S13).

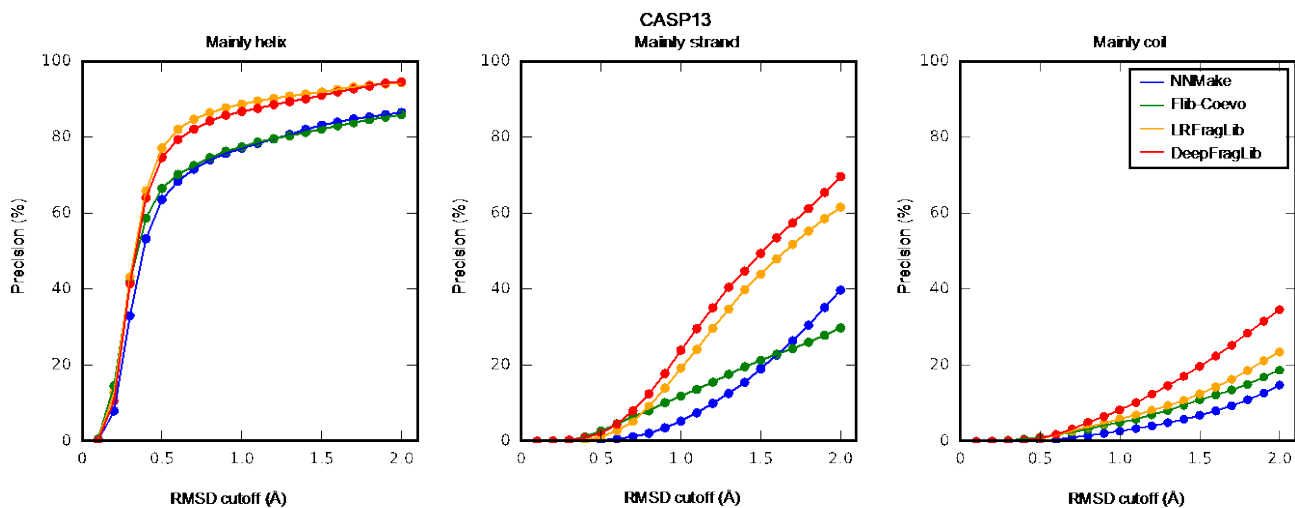


Fig. 3. Quality assessment in secondary structure classes. The precision of fragment libraries produced by DeepFragLib (red), LRFRagLib (orange), Flib-Coevo (green) and NNMake (blue) is evaluated in three separate secondary structure classes (mainly helix, mainly strand and mainly coil) on the CASP13 test protein set.

DeepFragLib achieves the highest precision for “mainly strand” and “mainly coil” regions, while DeepFragLib and LRFRagLib perform nearly equally best for the “mainly helix” region in all CASP sets. The highest performance for helical positions for all algorithms is expected, because template fragment databases contain more near-native fragments in helices than in strand and coil regions, considering that helical conformations are less flexible than strand and coil conformations⁷. Although the precision of all fragment libraries decreases substantially in the “mainly coil” regions, particularly in the latest CASP13 test set, DeepFragLib still outperforms the next best by 11.7% at 2Å RMSD cutoff. This substantial improvement in the quality of fragments obtained in the coil region is important for the correct assembly of protein structures.

Evaluation of protein structure prediction

The performance of fragment libraries constructed by DeepFragLib, NNMake and Flib-Coevo was tested in *ab initio* protein structure prediction for the domains of CASP11-13 targets using Rosetta. Following the guidelines of the Rosetta AbinitioRelax, we generated 20,000 structural models for each domain target in the three CASP test sets using standard parameters (but skipping the “Relax” stage due to the prohibitive computational time required for such a large number of models), ranked all models by TM-Score, and then compared the quality of the top1 (Fig. 4), top3 (Supplementary Fig. S14), and top5 (Supplementary Fig. S15) models generated by the three tested methods.

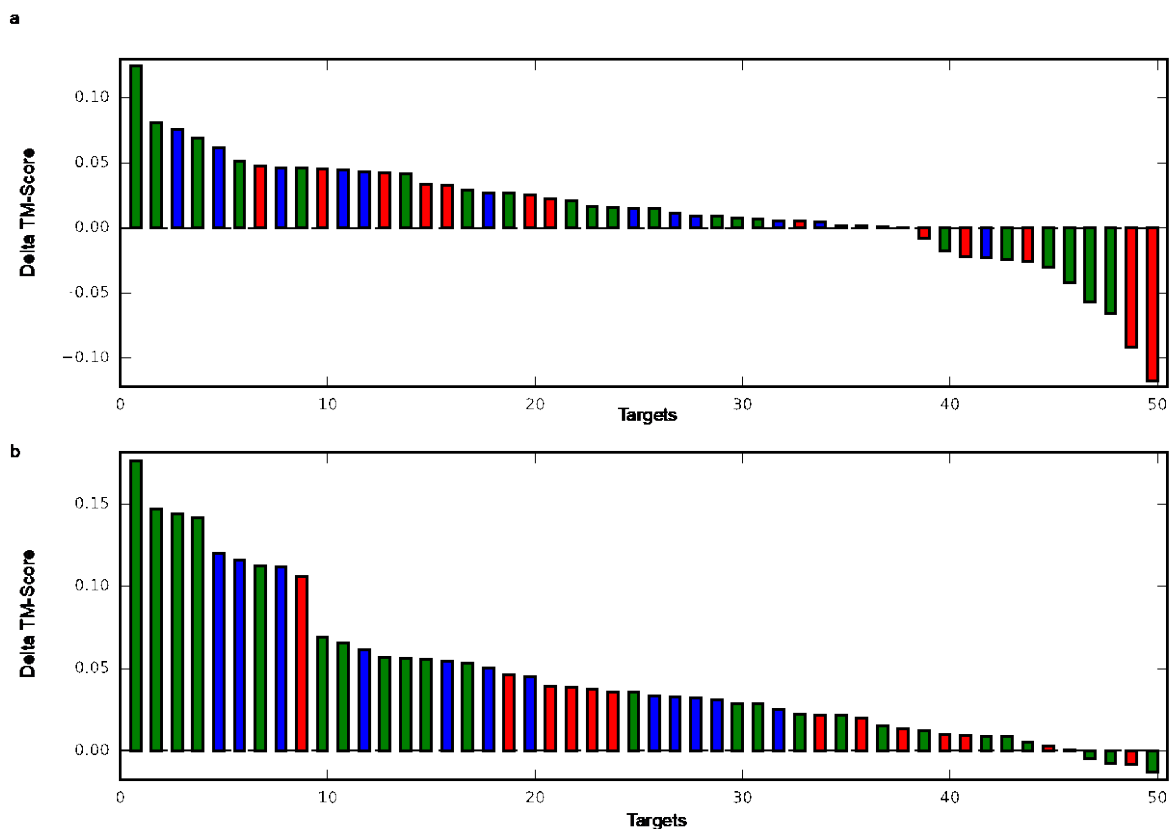


Fig. 4. Evaluation of the top1 models sampled by Rosetta simulations. (a) Improvement in the TM-Scores of the top1 models produced with DeepFragLib over NNMake. (b) Improvement in the TM-Scores of the top1 models produced with DeepFragLib over Flib-Coevo. The α , β and $\alpha\beta$ protein targets are

colored red, blue and green, respectively.

Table 2. The number of cases in which each fragment library achieves the best TM-Scores for the top1, top3 and top5 models for the overall 50 free-modeling targets in the CASP11-13 sets.

Category	DeepFragLib	NNMake	Flib-Coevo
Top1	36	12	2
Average Top3	35	12	3
Average Top5	31	16	3

The highest value in each protein set is highlighted in bold.

For the top1 models, DeepFragLib achieves the highest TM-Score in 36 out of 50 targets, compared to 12 by NNMake and 2 by Flib-Coevo, respectively (Table 2 and Supplementary Fig. S16). DeepFragLib also achieves the highest average TM-Scores for the majority of targets in the top3 and top5 models. The differences are statistically significant according to the Wilcoxon signed-rank test (Supplementary Table S2 and Fig. S17&S18). The p-value between DeepFragLib and NNMake is < 0.01 for the top1 and top3 models and < 0.02 for the top 5 models. The p-value is $\ll 0.001$ for the difference between DeepFragLib and Flib-Coevo.

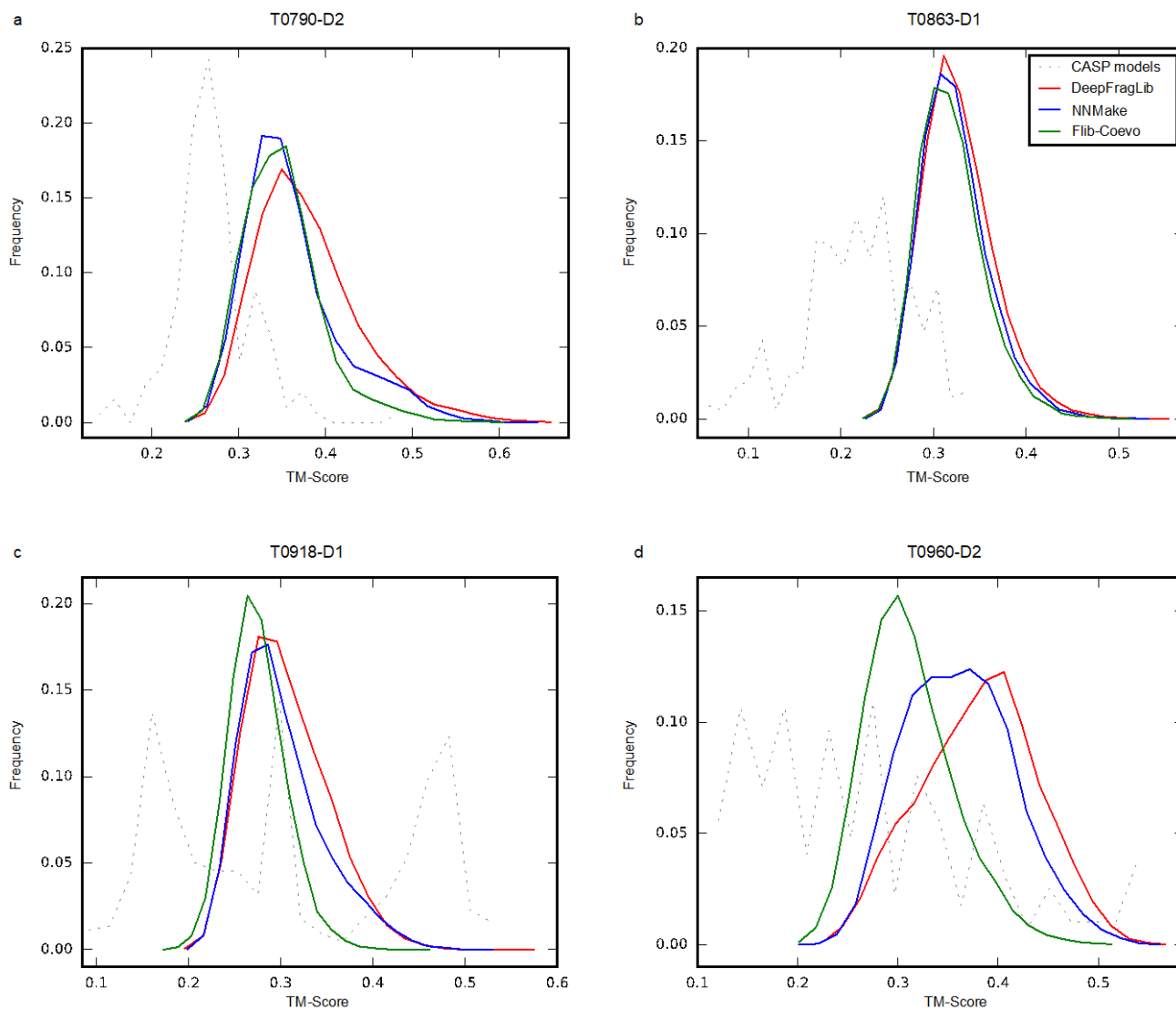


Fig. 5. Case study on the distribution of TM-scores. The distributions of TM-scores for all sampled model structures are compared on four exemplar domain targets: (a) T0790-D2 ($\alpha\beta$ protein); (b) T0863-D1 (α protein); (c) T0918-D1 (β protein); and (d) T0960-D2 (β protein). The structure models are generated using the Rosetta AbinitioRelax protocol with DeepFragLib (red solid), NNMake (blue solid) and Flib-Coevo (green solid), respectively. The distributions of CASP models (gray dashed) are shown as a reference only.

Fig. 5 shows four examples in which DeepFragLib improves the structural sampling, including one case of α protein (T0863-D1), one case of $\alpha\beta$ protein (T0790-D2) and two

cases of β proteins (T0918-D1 and T0960-D2). The positive shifts in the distributions of TM-Scores from other fragment libraries confirm that better fragments by DeepFragLib lead to better assembled structures. With the distribution of TM-Scores of all CASP models plotted as an approximate reference, the results illustrate the ability of DeepFragLib-based simulations to sample the best models among all the methods. We emphasize that the result from CASP models is not for comparison but for a reference only because CASP models were predicted by a wide variety of methods without knowledge of domain definitions and by using homologous structure information for the overall target sequence if available. These models were generated without knowledge of domain definition. Moreover, each CASP method only predicts a few models although they may sample more than 20,000 structures generated here.

Running time

We estimated the computing time for DeepFragLib, NNMake, LRFragLib and Flib-Coevo on a computer cluster with 8 1080Ti GPUs. Well parallelized DeepFragLib takes 20-30 minutes to build the fragment library for a 100-residue target protein, much faster than 85 minutes by LRFragLib. NNMake is the fastest algorithm, taking only about 15 minutes for the library construction. Flib-Coevo, however, is very slow, requiring about 170 minutes for the same target. Therefore, using sophisticated neural networks did not prohibit DeepFragLib from achieving a reasonable computational efficiency that is comparable to NNMake.

Discussion

Existing fragment libraries obtain candidate fragments from large protein datasets. For

example, NNMake utilizes a self-built dataset containing 16,802 chains and Flib-Coevo extracts fragments from the whole PDB database. In contrast, DeepFragLib employs only 956 high-quality chains in the template fragment database (HR956) for library construction, and yet achieves the best performance in fragments obtained and models built. Using only high-quality structures avoids noises and errors associated with low-quality structures, benefiting the training of deep neural networks without impairing the model generalizability.

The overall hierarchical structure of DeepFragLib was designed to reduce the computational complexity and increase computational efficiency. The coarse-grained CLA models only retain the top 5,000 out of > 200,000 fragments at each position as inputs to the fine-grained REG model, which in turn simplifies both training and inference of the ultra-deep REG model. The CLA models alone could not provide accurate fragment identification. In the CASP13 set, the control fragment libraries built from the top fragments selected by the CLA models alone exhibit substantial performance reduction by all metrics (Supplementary Fig. S19). This highlights the importance of REG model in the detection of near-native fragments. Tandemly connecting the classification and regression modules allows efficient recruitment of high-quality fragments, because training and inference of the REG model from all fragment data (without filtering by the CLA models) are computationally prohibitive.

From LRFragLib to DeepFragLib, we introduced two main changes to the composition of the fragment library. Firstly, we extended the maximal fragment length from 10 to 15

residues. Longer fragments could improve the sampling efficiency in protein folding simulations, since these fragments include more non-local structural preferences. After upgrading from LRfragLib to DeepFragLib, the mean fragment length increases by approximately one residue in the CASP11-13 test set (Supplementary Table S3). Moreover, 16.8% of the fragments identified by DeepFragLib are longer than 9 residues, compared to only 2.3% in LRfragLib and 4.0% in Flib-Coevo (Supplementary Table S4). It is possible to merge these long fragments (length > 9) collected by DeepFragLib into the NNMake library of 9-mer fragments so as to provide a larger pool for structure prediction by Rosetta. Secondly, unlike LRfragLib, which can recruit an unlimited number of fragments in the greedy enrichment step, the number of fragments in DeepFragLib is restricted to between 50 and 200 at every position. LRfragLib is inclined to obtain much more fragments in specific positions, especially in helical regions, and, consequently, the fractions of the helix, strand and coil fragments are unbalanced. After restricting the fragment number to [50, 200], the imbalance is alleviated in DeepFragLib. If the upper boundary is further reduced to 50, the distribution of fragments becomes more balanced among secondary structure categories and closer to those of NNMake and Flib-Coevo, while the fragment quality is nearly unchanged (Supplementary Fig. S20). Considering that an ideal fragment library should present some degree of structural variation to facilitate *in silico* structure prediction, we set the restriction of the number of fragments to 50 and 200 by default.

In our structure prediction by Rosetta, NNMake unexpectedly yielded the second-best performance after DeepFragLib, whereas Flib-Coevo produced results that are inconsistent

with its higher precision in the quality assessment of its fragment library. Therefore, factors other than common metrics like precision and coverage may also contribute to structure prediction. One possible factor is “depth”, the average number of predicted fragments. As such, we calculated the depths for fragment libraries constructed by various algorithms for all CASP11-13 targets (Supplementary Table S5). In comparison to the fixed 200 fragments required by NNMake, the depth is 132.124 ± 67.674 in DeepFragLib, but only 19.190 ± 3.868 in Flib-Coevo, which may explain the poor performance of this algorithm in real conformational sampling. Therefore, we suggest that fragment libraries should be evaluated by structure prediction in addition to the assessment of fragment quality.

Methods

Datasets

To develop CLA models, we first created a small high-resolution fragment dataset (referred as “HR956”) by culling 956 protein chains from PISCES³³ with resolution $< 1.5 \text{ \AA}$, R-value < 0.15 and pairwise identity $< 20\%$. These protein chains were divided into contiguous fragments of 7-15 residues. The secondary structure and solvent accessible surface area of each chain were calculated by DSSP³⁴. Backbone dihedral angles (φ and ψ), C α -atom-based angles (θ and τ), and residue contact information were calculated from their PDB³⁵ structures, respectively. A total of 1.9 million fragments of 7 to 15 residues in the HR956 set make approximately 20 billion pairs of fragments of the same lengths as the data for training the CLA models. Each fragment in the HR956 set also serves as a candidate

template for the fragment library in the inference stage of DeepFragLib.

We selected one representative chain from each “Fold” category of SCOPe v2.07³⁶ to compose an initial set and then removed all homologous chains (with E-value < 0.1 in PSI-BLAST³² after three rounds of iterations) against the HR956 and all CASP datasets, which resulted in a 350-chain dataset (referred as “SCOP350”) for training the REG model. The selection strategy (the third module in Fig. 1) was optimized on the CASP10 training set that includes 11 free-modeling domain targets in CASP10 (Supplementary Table S6). We adopted three independent test sets for evaluating the performance of DeepFragLib, which include 20, 21, and 9 free-modeling domain targets in CASP11 (Supplementary Table S7), CASP12 (Supplementary Table S8) and CASP13 (Supplementary Table S9), respectively. In constructing above datasets, we removed CASP domain targets without publicly available PDB structures, those with structures determined by nuclear magnetic resonance, those with many missing residues, and those with their chain lengths of < 40 residues. Thus, the numbers of free-modeling domain targets in our test sets are smaller than the ones listed on the CASP website. For the SCOP350 and CASP datasets, secondary structures were predicted by PSIPRED v4.01³⁷, while solvent accessible surface area, backbone dihedral angles, and C α -atom-based angles were predicted by SPIDER3²⁸ and its successor SPOT-1D²⁴. Residue contact information was first predicted by SPOT-Contact¹¹ and then refined by our in-house contact predictors for local (sequence separation < 6) and non-local (sequence separation \geq 6) residue pairs (see Supplementary Materials for the brief description of these in-house models)

Normalization of RMSD and dRMS

Similar to RMSD, distance-RMS (dRMS) measures the difference of atom-atom distance matrices for evaluating the structural similarity between fragments³⁸. dRMS is adequate for comparing highly flexible regions and computationally efficient by avoiding the time-consuming structural superimposition⁷. However, both RMSD and dRMS depend on the number of amino acid residues, which hinders the comparison of their values for fragments with different lengths. Here, we optimized two formulas to normalize RMSD and dRMS of fragments of various lengths (7-15 residues) to standard values of 10-residue fragments, following a protocol similar to the work of Carugo and Pongor³⁹. As shown in Supplementary Fig. S21, for each fragment length, we randomly chose 100 million fragment pairs and calculated their RMSD values. The right-shifts of RMSD distributions display the dependence of RMSD values on the fragment length. We then plotted the percentile points of various distributions against the fragment length and found a clear linear dependence, as shown by the quartiles. Interestingly, after dividing the RMSD percentile values by the RMSD10 reference (the corresponding one from the distribution of 10-residue fragments), all curves collapse into one single curve. We tried to use linear or logarithmic function to fit the curve and found that linear function achieved the highest correlation coefficient (> 0.999). The linear function is described by the following equation:

$$\frac{RMSD}{RMSD10} = 0.0729276N + 0.2707240 \quad (1)$$

where N is the number of residues. Similarly, dRMS values were normalized with the same

procedure (Supplementary Fig. S22) and the fitted function is described by the following equation:

$$\frac{dRMS}{dRMS10} = 0.09047712N + 0.0952288 \quad (2)$$

The normalization coefficients (RMSD/RMSD10 and dRMS/dRMS10) for different fragment lengths are listed in Supplementary Table S10. In this work, we regard RMSD10 as RMSD and dRMS10 as dRMS unless noted otherwise.

The classification module

In the first module (see Fig. 1), we employed stacked Bi-LSTM networks to construct the multiple cumbersome classification models, each optimized for one specific fragment length, with one-dimensional features of protein fragments as inputs. The flowchart of the CLA models is illustrated in Supplementary Fig. S1. Specifically, similarity scores between amino acid pairs were obtained from the BLOSUM62 matrix⁴⁰, 10-dimensional physicochemical features of 20 amino acids were obtained from Kidera et al.⁴¹, three-class secondary structure probabilities were predicted by PSIPRED, whereas solvent accessible surface area, backbone dihedral angles (φ and ψ) as well as C α -atom-based angles (θ and τ) were predicted by SPIDER3. All above features were flattened into a 43N-dimensional input feature vector, where N stands for the fragment length. We employed the same procedure to label samples as used in our previous work⁷. For each specific fragment, we labeled the 20 pairs with the lowest dRMS between this selected fragment and all other fragments of the same length (simultaneously requiring dRMS < 1 Å) as positive samples,

with redundancy in fragment pair indices carefully removed. All remaining fragment pairs were labeled as negative samples. We randomly chose 80% of samples as the training set for 5-fold cross-validation and the remaining 20% of samples as the independent test set.

The cumbersome classification model consists of three Bi-LSTM layers with 64 nodes per direction, followed by three additional fully connected hidden layers with 128 nodes (to process the last time step output of Bi-LSTM layers). The Bi-LSTM layers use the hyperbolic tangent (tanh) activation while the fully connected layers adopt the Rectified Linear Unit (ReLU) activation. The model has a single output node with the sigmoid activation function to normalize the confidence value to [0, 1]. To reduce the risk of overfitting, we applied the dropout mechanism⁴² with a ratio of 50% on the fully connected layers and L2 weight regularization on all trainable parameters during training. The model was trained using cross entropy as the loss function, which was optimized by the Adam optimizer⁴³ with an initial learning rate of $7.5e^{-4}$.

Knowledge distillation

The cumbersome classification models are too computationally intensive during the prediction step: nearly 180 million times for a 100-residue target protein (1.9 million \times 90). Therefore, we adopted knowledge distillation to train a small model for model compression and acceleration. The distilled model contains a one-dimensional convolution layer with 43 filters of the kernel size of 3, followed by a Bi-LSTM layer with 64 nodes per direction and a fully connected layer with 32 nodes. The output layer is identical to that of the cumbersome classification model. We also applied a dropout rate of 50% on the fully

connected layer and L2 regularization on all trainable parameters for training the distilled models. The total loss function is described by the following equation:

$$Loss = \sum_{i=1}^M H(P_i, Q_i) / M + \sum_{i=1}^M \sum_{j=1}^N (BLSTM2_P_{ij} - BLSTM_Q_{ij})^2 / MN + 0.5 \sum_{i=1}^M H(LABEL_i, Q_i) / M + \lambda \sum_{k=1}^K \theta_k^2 \quad (3)$$

where P and Q stand for the predicted labels of the cumbersome model and the distilled model, respectively, H stands for the cross-entropy function, BLSTM2_P and BLSTM_Q stand for the outputs of all steps of the second Bi-LSTM layer of the cumbersome model and the ones of Bi-LSTM layer in the distilled model, respectively, LABEL stands for the ground truth, and θ stands for parameters in the distilled model. The three items in the loss function have respective weights of 1.0, 1.0, and 0.5. Here, the loss function contains not only the commonly used loss components in knowledge distillation (the cross-entropy between the ground truth and outputs of the distilled model as well as the cross-entropy between outputs of the cumbersome model and those of the distilled model)³⁰ but also the mean squared error between the outputs of Bi-LSTM layers of the cumbersome model and those of Bi-LSTM layer in the distilled model. During training, the distilled models reached higher accuracies and faster convergence speed compared to the models of the same topology but trained from scratch using the common loss function. Instead of utilizing the Adam optimizer, we adopted the SWATS optimization algorithm, which first employs the Adam optimizer for fast convergence and then turns to the stochastic gradient descent (SGD) with momentum optimization for better performance at later stages of training when a triggering condition is satisfied⁴⁴. In addition to the condition related to the projection of

Adam steps on the gradient subspace as proposed by Keskar et al.⁴⁴, the turning point of the optimization method was set to be after 300 epochs of iteration. We tuned the initial learning rate as $5.0e^{-4}$. All cumbersome models with different fragment lengths were distilled. In this work, we regard final distilled models as the CLA models unless noted otherwise.

The regression module

In the second module (see Fig. 1), we employed an ultra-deep regression model based on the ResNeXt architecture to directly predict RMSD values between candidate template fragments and the native fragments. The flowchart of the REG model is illustrated in Supplementary Fig. S2. For each protein in the SCOP350 set, we selected the top 1,000 fragments with the highest confidence values output by the CLA model per position for each fragment length, which altogether composed a dataset containing 430 million samples. We randomly chose 70% for training and cross validation and the remaining 30% as the independent test set. The training samples were further divided into the training set and the validation set with a ratio of 4:1. We refined the predicted residue-residue contact matrices of fragments, flattened them into vectors and calculated the root-mean-square-distance between contact vectors for a pair of fragments. Considering that the individual CLA model for different fragment lengths has a different number of states in the Bi-LSTM layer (from 7 to 15), we uniformly extracted the outputs of the last seven timesteps from the Bi-LSTM layer for all CLA models. The output of CLA models, the distance of contact vectors and the fragment length were broadcast together with the Bi-LSTM outputs as input for the

REG model. Therefore, the input features have the same temporal dimension as the Bi-LSTM outputs that contain seven timesteps. During inference, we selected the top 5,000 fragments from CLA models per position to predict RMSD by the REG model.

Considering that input features are decomposed into seven timesteps, we designed a cyclically dilated convolution to expand receptive fields to the whole sequence in a single layer. Instead of padding zero values, we padded the feature values of the last few steps at the head of the first step and in turn padded the first few steps at the end of the last step. Thus, the features of seven steps form a loop and these features are repeated cyclically. For a specified step, the convolution operations with dilation rate of one, two and three jointly can expand receptive fields to all seven steps (Supplementary Fig. S2). Following the bottleneck layer, we designed twelve cyclically dilated convolutional layers with 256 filters in parallel for each dilation rate. Thus, one building block of the REG model includes 36 parallel paths. These building blocks are repeated for 40 times, followed by an output layer. Batch normalization⁴⁵ and L2 regularization are also adopted in the REG model. We utilized Adam optimizer with a learning rate of $5e^{-5}$ for model training. In this work, all deep neural networks were coded with the TensorFlow framework⁴⁶ and trained on 16 1080Ti GPUs.

The fragment selection module

In the third module (see Fig. 1), our selection strategy contains three stages to obtain candidate fragments at each sequence position for constructing the fragment library. This three-stage selection strategy was optimized on the CASP10 training set. In Stage 1, we

used a set of customized thresholds for different fragment lengths (see Supplementary Table S11) to extract candidate fragments based on their predicted RMSD values output by the REG model. These fragments were ranked in ascending order of predicted RMSDs and only the top ones were retained to ensure that the number of candidates selected per position (recorded as “N”) never exceeded 200. If N was less than 50 at a position, which is possible for loop regions, in particular, when not enough fragments were predicted within the cutoff thresholds, we introduced two additional stages for enrichment. In Stage 2, the fragments with predicted RMSD less than a lower customized threshold of the corresponding fragment lengths (see Supplementary Table S11) were extracted in ascending order of predicted RMSDs and the enrichment stopped once N reached 50. If N was still less than 50 after Stage 2, Stage 3 was initiated, where all unselected 7-residue fragments were ranked in descending order of output values by the CLA models and top fragments were recruited until N reached 50. After this selection strategy, the number of candidate fragments at each position would fall between 50 and 200. Notably, the customized RMSD thresholds adopted in Stages 1 and 2 were computed from the REG model. For each fragment length, the Stage 1 threshold was set as the RMSD cutoff such that 90% of the fragments predicted with lower RMSDs than this cutoff were near-native fragments with $\text{RMSD} \leq 2.0 \text{ \AA}$. The Stage 2 threshold was computed similarly, but using a lower percentage of 85%. All thresholds were finally controlled to be $\text{RMSD} \leq 2.0 \text{ \AA}$.

In addition to DeepFragLib developed here, we also constructed fragment libraries for NNMake, LRFragLib, and Flib-Coevo using their executable programs (NNMake in

Rosetta v3.8, LRFragmentLib v1.0 and Flib-Coevo v1.01) with all parameters set to default values. Notably, fragment libraries generated by Flib-Coevo contained a few repeated candidates at the same position, so we removed these redundant fragments for a fair comparison. Because LRFragmentLib occasionally produced a large number of fragments at several individual positions, we slightly modified the software and limited the maximum number to 200 at each position.

Quality assessment of fragment libraries

We adopted the widely used metrics, precision and coverage, to evaluate the quality of fragment libraries. Specifically, precision is defined as the fraction of near-native fragments in the whole fragment library and coverage is defined as the fraction of the sequence positions that are covered by at least one near-native fragments, where near-native fragments refer to those with RMSD values to the native ones less than a certain cutoff value. Same as our previous work, we used a series of cutoff values ranging from 0.1 Å to 2.0 Å with a step size of 0.1 Å to produce curves of precision and coverage. Moreover, as described previously, we introduced an additional metric, the position-averaged precision, which averages the fractions of near-native fragments over all sequence positions. This metric could eliminate the artificial effect caused by significantly different numbers of fragments at different positions.

Conformational sampling and model generation with Rosetta

We employed the Rosetta v3.8 C++ source code to evaluate the usefulness of fragment libraries in *ab initio* protein structure prediction. Rosetta iteratively utilizes short (3-mers)

and long (9-mers) fragments for fragment assembly using Monte Carlo simulations. Specifically, at each step, the backbone torsion angles of a randomly chosen segment of the target protein will be replaced by a randomly selected fragment from the fragment library and the proposed conformational change will be accepted or declined following the Metropolis-Hastings algorithm. The original Rosetta only allows the fixed length for long fragments. Thus, we only modified the standard Rosetta AbinitioRelax application to accept long fragments of variable lengths. We converted fragment libraries constructed by DeepFragLib, NNMake and Flib-Coevo to the Rosetta fragment format and directly fed them into AbinitioRelax to replace the standard 9-mer library, respectively. By skipping the “Relax” stage of AbinitioRelax to reduce the computational time, we generated 20,000 protein models for each free-modeling protein target in the CASP11, CASP12, and CASP13 test sets. The quality of these sampled structure models was evaluated by TM-Score⁴⁷.

Data Availability

The full template fragment database HR956 used for fragment library construction and all four CASP datasets used for the quality evaluation of fragment libraries are available on Code Ocean (<https://doi.org/10.24433/CO.3579011.v1>).

Code Availability

All source codes and models of DeepFragLib are openly available on Code Ocean

(<https://doi.org/10.24433/CO.3579011.v1>) and on GitHub (<https://github.com/ElwynWang/DeepFragLib>). We also provided an online server for DeepFragLib at <http://structpred.life.tsinghua.edu.cn/DeepFragLib.html>.

Correspondence

Correspondence and requests for materials should be addressed to H.G. and Y. Z.

Acknowledgements

This work has been supported by the funds from the National Natural Science Foundation of China (#31670723, #81861138009, #91746119 & #31621092) and from the Beijing Advanced Innovation Center for Structural Biology to H. G., as well as by the funds from Australian Research Council DP180102060 and the National Health and Medical Research Council (1121629) of Australia to Y. Z.

Author contributions

T. W. contributed to methodology, experimental design, software, formal analysis, the server, and writing of the original draft. Y. Q. contributed to formal analysis and the server. W. D. contributed to the server. W. M. was involved in methodology. Y. Z. was involved in experimental design and writing. H. G. contributed to experimental design and was responsible for supervision, writing as well as funding acquisition. All authors reviewed the final manuscript.

Competing interests

The authors declare no competing interests.

References

- 1 Bradley, P., Misura, K. M. S. & Baker, D. Toward high-resolution de novo structure prediction for small proteins. *Science* **309**, 1868-1871 (2005).
- 2 Dill, K. A. & MacCallum, J. L. The Protein-Folding Problem, 50 Years On. *Science* **338**, 1042-1046 (2012).
- 3 Rigden, D. J. *From protein structure to function with bioinformatics Ch. 1*. (Springer Netherlands, 2017).
- 4 Soding, J. Big-data approaches to protein structure prediction. *Science* **355**, 248-249 (2017).
- 5 Kim, D. E., Blum, B., Bradley, P. & Baker, D. Sampling Bottlenecks in De novo Protein Structure Prediction. *J Mol Biol* **393**, 249-260 (2009).
- 6 Jothi, A. Principles, Challenges and Advances in ab initio Protein Structure Prediction. *Protein Peptide Lett* **19**, 1194-1204 (2012).
- 7 Wang, T., Yang, Y., Zhou, Y. & Gong, H. LRFragLib: an effective algorithm to identify fragments for de novo protein structure prediction. *Bioinformatics* **33**, 677-684 (2017).
- 8 Baeten, L. *et al.* Reconstruction of protein backbones from the BriX collection of canonical protein fragments. *PLoS computational biology* **4**, e1000083 (2008).
- 9 Xu, J. Distance-based Protein Folding Powered by Deep Learning. Preprint at <https://arxiv.org/abs/1811.03481> (2018).
- 10 R.Evans *et al.* De novo structure prediction with deep-learning based scoring. In *Thirteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstracts)* (Iberostar Paraiso, Riviera Maya, Mexico, 2018).
- 11 Hanson, J., Paliwal, K., Litfin, T., Yang, Y. & Zhou, Y. Accurate prediction of protein contact maps by coupling residual two-dimensional bidirectional long short-term memory with convolutional neural networks. *Bioinformatics* **34**, 4039-4045 (2018).
- 12 Simons, K. T., Kooperberg, C., Huang, E. & Baker, D. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *J Mol Biol* **268**, 209-225 (1997).
- 13 Xu, D. & Zhang, Y. Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *Proteins* **80**, 1715-1735 (2012).
- 14 Yang, J. *et al.* The I-TASSER Suite: protein structure and function prediction. *Nature methods* **12**, 7-8 (2015).
- 15 Rohl, C. A., Strauss, C. E., Misura, K. M. & Baker, D. Protein structure prediction using Rosetta.

- Methods in enzymology* **383**, 66-93 (2004).
- 16 Kim, D. E., Chivian, D. & Baker, D. Protein structure prediction and analysis using the Robetta server. *Nucleic acids research* **32**, W526-531 (2004).
- 17 Gront, D., Kulp, D. W., Vernon, R. M., Strauss, C. E. & Baker, D. Generalized fragment picking in Rosetta: design, protocols and applications. *PloS one* **6**, e23294 (2011).
- 18 Kalev, I. & Habeck, M. HHfrag: HMM-based fragment detection using HHpred. *Bioinformatics* **27**, 3110-3116 (2011).
- 19 Trevizani, R., Custodio, F. L., Dos Santos, K. B. & Dardenne, L. E. Critical Features of Fragment Libraries for Protein Structure Prediction. *PloS one* **12**, e0170131 (2017).
- 20 Bhattacharya, D., Adhikari, B., Li, J. & Cheng, J. FRAGSION: ultra-fast protein fragment library generation by IOHMM sampling. *Bioinformatics* **32**, 2059-2061 (2016).
- 21 de Oliveira, S. H. P. & Deane, C. M. Combining co-evolution and secondary structure prediction to improve fragment library generation. *Bioinformatics* **34**, 2219-2227 (2018).
- 22 Wang, S., Sun, S., Li, Z., Zhang, R. & Xu, J. Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model. *PLoS computational biology* **13**, e1005324 (2017).
- 23 Wang, S., Li, Z., Yu, Y. & Xu, J. Folding Membrane Proteins by Deep Transfer Learning. *Cell systems* **5**, 202-211 e203 (2017).
- 24 Paliwal, K., Hanson, J., Litfin, T., Zhou, Y. & Yang, Y. Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks. *Bioinformatics* (2018).
- 25 Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735-1780 (1997).
- 26 Xie, S., Girshick, R., Dollár, P., Tu, Z. & He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 5987-5995 (IEEE: Honolulu, Hawaii, USA, 2017).
- 27 Schuster, M. & Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* **45**, 2673-2681 (1997).
- 28 Heffernan, R., Yang, Y., Paliwal, K. & Zhou, Y. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. *Bioinformatics* **33**, 2842-2849 (2017).
- 29 He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. Preprint at <https://arxiv.org/abs/1512.03385> (2015).
- 30 Hinton, G., Vinyals, O. & Dean, J. Distilling the Knowledge in a Neural Network. Preprint at <https://arxiv.org/abs/1503.02531> (2015).
- 31 Yu, F. & Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. Preprint at <https://arxiv.org/abs/1511.07122> (2015).
- 32 Altschul, S. F. *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research* **25**, 3389-3402 (1997).
- 33 Wang, G. & Dunbrack, R. L., Jr. PISCES: a protein sequence culling server. *Bioinformatics* **19**,

- 1589-1591 (2003).
- 34 Kabsch, W. & Sander, C. Dictionary of Protein Secondary Structure - Pattern-Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers* **22**, 2577-2637 (1983).
- 35 Berman, H. M. *et al.* The Protein Data Bank. *Nucleic acids research* **28**, 235-242 (2000).
- 36 Fox, N. K., Brenner, S. E. & Chandonia, J. M. SCOPe: Structural Classification of Proteins--extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic acids research* **42**, D304-309 (2014).
- 37 Jones, D. T. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol* **292**, 195-202 (1999).
- 38 Hubner, I. A., Deeds, E. J. & Shakhnovich, E. I. Understanding ensemble protein folding at atomic detail. *Proceedings of the National Academy of Sciences* **103**, 17747-17752 (2006).
- 39 Carugo, O. & Pongor, S. A normalized root-mean-square distance for comparing protein three-dimensional structures. *Protein Sci.* **10**, 1470-1473 (2001).
- 40 Henikoff, S. & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America* **89**, 10915-10919 (1992).
- 41 Kidera, A., Konishi, Y., Oka, M., Ooi, T. & Scheraga, H. A. Statistical Analysis of the Physical Properties of the 20 Naturally Occurring Amino Acids. *Journal of Protein Chemistry* **4**, 23-55 (1985).
- 42 Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929-1958 (2014).
- 43 Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. Preprint at <https://arxiv.org/abs/1412.6980> (2014).
- 44 Keskar, N. S. & Socher, R. Improving Generalization Performance by Switching from Adam to SGD. Preprint at <https://arxiv.org/abs/1712.07628> (2017).
- 45 Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (JMLR.org, Lille, France, 2015).
- 46 Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (TensorFlow, 2015); <http://download.tensorflow.org/paper/whitepaper2015.pdf>
- 47 Zhang, Y. & Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic acids research* **33**, 2302-2309 (2005).