

Supplementary Information

An end-to-end framework for the prediction of protein structure and fitness from single sequence

Yinghui Chen^{1,2,+}, Yunxin Xu^{1,2,+}, Di Liu^{1,2}, Yaoguang Xing^{1,2}, and Haipeng Gong^{1,2,*}

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

²Beijing Frontier Research Center for Biological Structure, Tsinghua University, Beijing, 100084,
China

⁺These authors contributed equally to this work.

^{*}To whom correspondence should be addressed: hgong@tsinghua.edu.cn

1 Supplementary Results for Structure Prediction

1.1 Sequentially arranged Folding Units incrementally enhance the accuracy of structure prediction

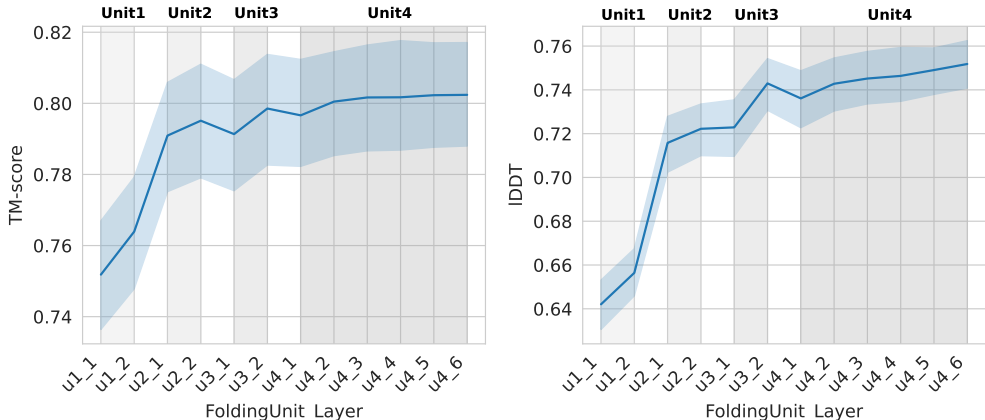


Figure S1. Evolution of TM-score and IDDT for coordinates generated by sequentially arranged Folding Units. The horizontal axis displays the structure prediction layers (*i.e.* Pred-XYZ modules) of sequentially arranged Folding Units. For example, u4_6 means the 6th Pred-XYZ module of the 4th Folding Unit. The vertical axis shows the TM-score or IDDT values of the generated C_{α} coordinates. The line connects the average TM-score/IDDT values for the outputs of sequential layers. The blue shadow shows the 95% confidence interval of TM-score/IDDT values among 568 CAMEO samples with length between 100 and 600 residues. The gray shadows mark different Folding Units. Source data are provided as a Source Data file.

We use SPIRED to predict structures for 568 proteins from the CAMEO set (Aug. 2022 ~ Aug. 2023, $100 < L < 600$). Analysis on the quality of the generated C_{α} coordinates by Pred-XYZ modules suggests a progressive optimization of predicted structures among the four Folding Units (Supplementary Figure S1). Specifically, the average TM-score of generated C_{α} coordinates increases from 0.752 for the 1st layer (*i.e.* Pred-XYZ module) of the 1st Folding Unit to 0.802 for the 6th layer of the 4th Folding Unit. Similarly, the IDDT increases from 0.642 to 0.752. This trend indicates that the sequentially arranged Folding Units play a similar role to the recurrent expansion by recycling, both of which are capable of gradually optimizing the accuracy of the predicted structures.

1.2 Performance of structure prediction models on CAMEO and CASP15 targets with 4 recycling cycles

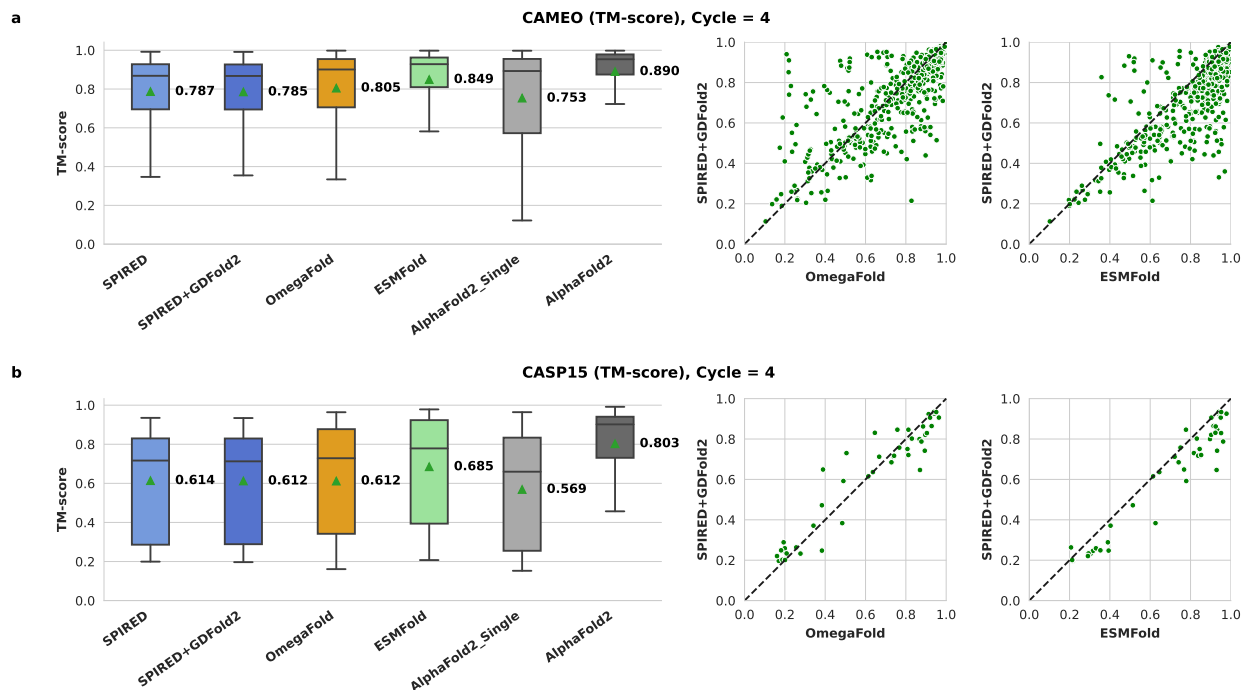


Figure S2. Comparison of model performance on CAMEO and CASP15. **a** The boxplot compares the TM-scores of structures predicted by SPIRED, SPIRED+GDFold2, OmegaFold, and ESMFold with Cycle = 4 on the CAMEO set, which is composed of 680 samples of proteins. Here, the green triangle indicates the mean value with the corresponding number marked aside, the center line represents the median, the box limits correspond to the upper and lower quartiles and the whiskers extend to 1.5 inter-quartile range. A more detailed comparison is shown in the scatter plot, where each point represents a protein sample, the vertical axis represents the TM-score of SPIRED+GDFold2 predictions, and the horizontal axis represents the results of OmegaFold and ESMFold, respectively. **b** Similarly, the boxplot and scatter plot present the general and detailed comparisons between the evaluation results for these models with Cycle = 4 on the CASP15 set, which is composed of 45 samples of protein domains, whose sequences and ground truth structures are available from the CASP15 official website. The boxplot is constructed in the same way as that in subplot **a**. In both boxplots, the single-sequence-based predictors under evaluation are also compared with the standard AlphaFold2 (ColabFold version, Cycle = default value) taking the MSA or single sequence as input, respectively. EMBER3D is not shown for it does not support recycling. Source data are provided as a Source Data file.

SPIRED+GDFold2, OmegaFold, and ESMFold2 are evaluated with recycling at Cycle = 4. As shown in the boxplots of Supplementary Figure S2, ESMFold2 demonstrates the best performance on both CAMEO and CASP15 targets, while SPIRED+GDFold2 and OmegaFold2 are comparable to each other. The scatter plots show the significant differences between SPIRED and OmegaFold for different individual proteins, thus indicating the possible complementarity between the prediction results of different models.

1.3 Detailed evaluation of structure prediction on different Classes of SCOPe

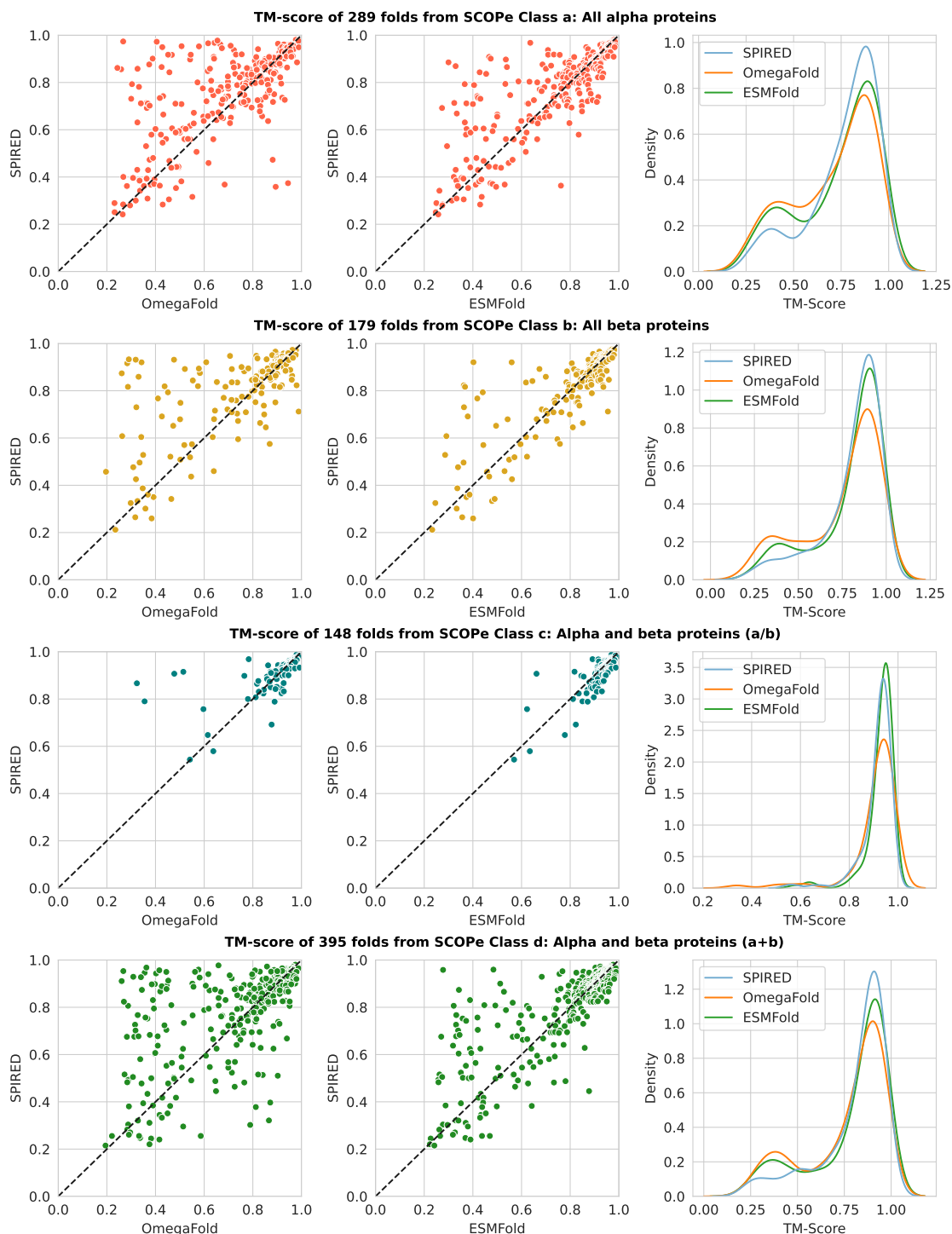


Figure S3. The structure prediction performance of the model on different SCOPe Classes (Class a, b, c and d). Each point in the scatter plot represents the average TM-score of all domains within one SCOPe fold. In the scatter plot, the vertical axis is the prediction performance (TM-score) of SPIRED on different folds, while the horizontal axis represents the results for OmegaFold and ESMFold, respectively. The KDE plot is used to visualize the distribution of model performance (in TM-score) over different folds within one SCOPe class for the tested models. Source data are provided as a Source Data file.

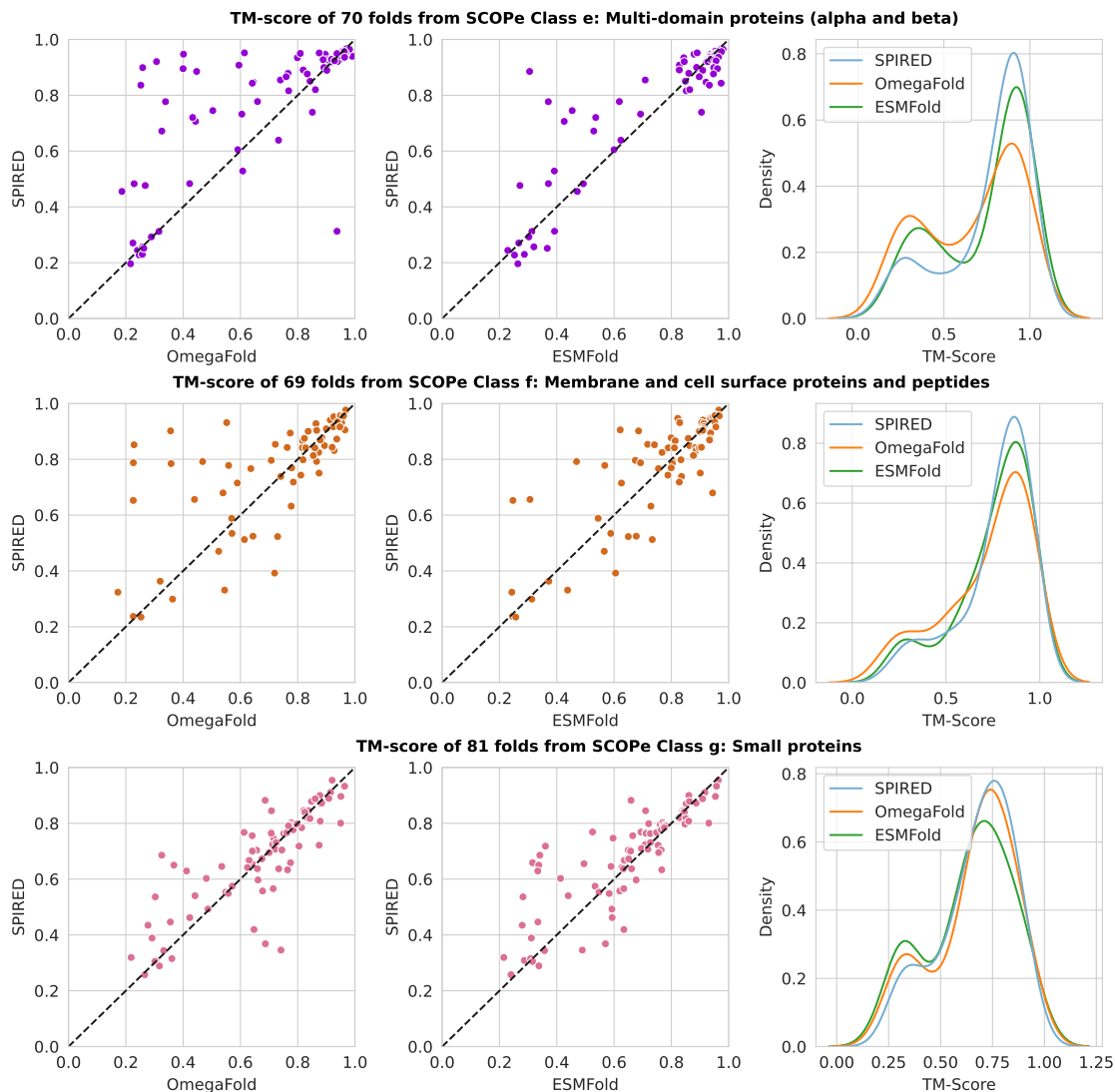


Figure S4. The structure prediction performance of the model on different SCOPe Classes (Class e, f and g). Each point in the scatter plot represents the average TM-score of all domains within one SCOPe fold. In the scatter plot, the vertical axis is the prediction performance (TM-score) of SPIRED on different folds, while the horizontal axis represents the results for OmegaFold and ESMFold, respectively. The KDE plot is used to visualize the distribution of model performance (in TM-score) over different folds within one SCOPe class for the tested models. Source data are provided as a Source Data file.

1.4 Case study for CAMEO, CASP15, SCOPe and CATH targets

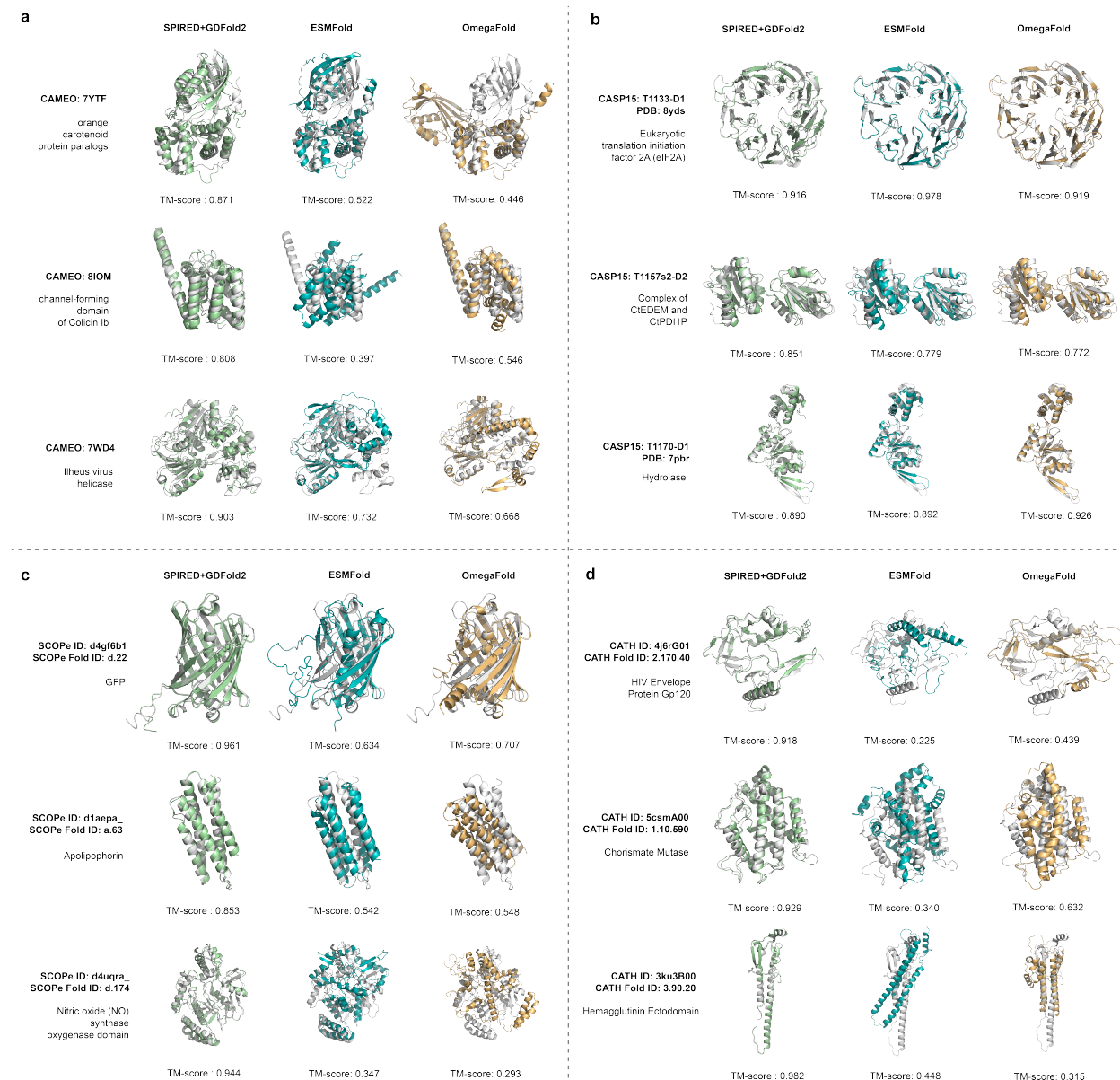


Figure S5. Case study. Exemplar targets are chosen from the CAMEO (a), CASP15 (b), SCOPe (c) and CATH (d) sets, respectively. For each case, the structures predicted by SPIRED+GDFold2 (green), ESMFold (teal) and OmegaFold (orange) are superposed onto the corresponding ground-truth structure (silver).

Structure prediction cases of SPIRED are shown in Supplementary Figure S5 for the CAMEO, CASP15, SCOPe and CATH sets. SPIRED demonstrates predictive advantage on certain proteins of all alpha type, such as CAMEO: 8IOM, SCOPe: d1aepa_ (Apolipoporphin), and CATH ID: 5csmA00 (Chorismate Mutase). Furthermore, SPIRED also shows advantage on GFP (SCOPe Fold ID: d.22), a target extensively used in protein design and engineering.

1.5 Evaluation of structure prediction at the level of CATH topologies

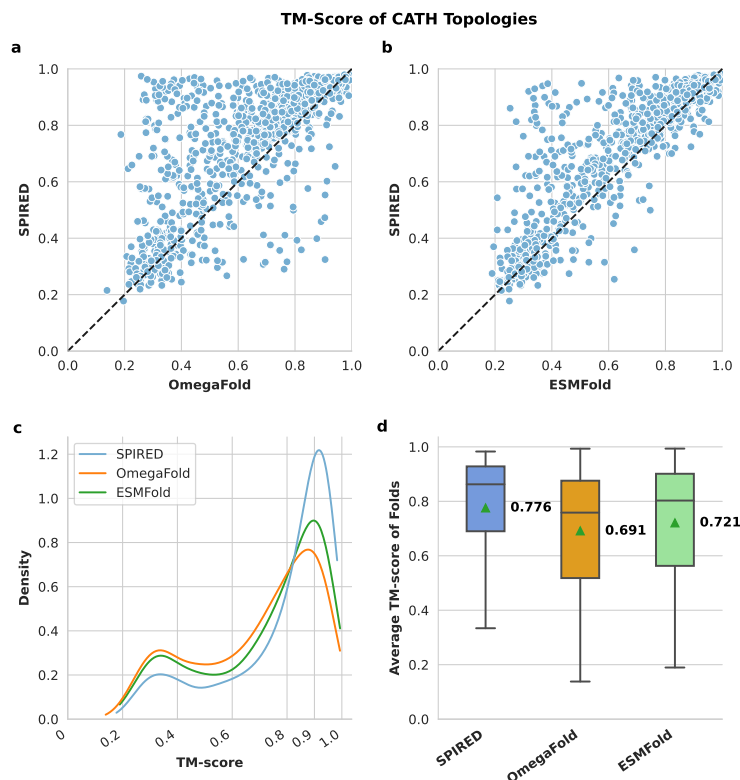


Figure S6. Evaluation of structure prediction on the CATH v4.2 S35 structural classification database. **a-b** In the scatter plot, each point represents the average TM-score of all domains of one CATH topology. The vertical axis is the prediction performance (TM-score) of SPIRED on different CATH topologies, while the horizontal axis represents results for OmegaFold and ESMFold, respectively. **c** The KDE plot is used to visualize the distribution of model performance (in TM-score) over different CATH topologies for the tested models. **d** The boxplot compares the TM-scores calculated over all CATH topologies (sample size = 1,223) for different models, where the green triangle indicates the mean value with the corresponding number marked aside, the center line represents the median, the box limits correspond to the upper and lower quartiles and the whiskers extend to 1.5 inter-quartile range. Source data are provided as a Source Data file.

CATH splits experimentally determined protein structures into domains, which are classified into 4 hierarchies: class (C), architecture (A), topology/fold (T) and homologous superfamily (H). We analyze the performance of structure prediction models on 1,223 topologies (consisting of 24,183 domains) from the CATH database (v4.2, S35, July 2017) in Supplementary Figure S6. Among the topologies in which SPIRED performs better, there are 96 CATH topologies for which SPIRED achieves an advantage of > 0.2 in TM-score over ESMFold, whereas ESMFold outperforms SPIRED by > 0.2 in TM-score for only 5 CATH topologies (Supplementary Figure S6a,b). According to the KDE plot, SPIRED has the highest density in the high accuracy region (TM-score = ~0.9) (Supplementary Figure S6c). SPIRED also reaches the highest mean TM-score when averaged over all CATH topologies (Supplementary Figure S6d). Therefore, SPIRED exhibits similar advantageous performance on the CATH database as it does on SCOPe. Although OmegaFold and ESMFold do not explicitly use the CATH database as a training set, the majority of structures in CATH database are included in the PDB database they utilize. Nevertheless, the above results in combination with the analysis over the SCOPe database (Figure 3) indicate that SPIRED is capable of learning the sequence-structure relationship from a wider range of topology/fold types than ESMFold and OmegaFold.

1.6 Detailed comparison and discussion on single-sequence-based structure prediction methods

SPIRED-Fitness, OmegaFold and ESMFold exhibit different statistical behaviors among the SCOPe protein folds. In Supplementary Data 1, 2 and 3, we list the types of protein folds in which these three methods fail to provide good structural models (*i.e.* mean TM-score < 0.5). Clearly, the number of failure is the smallest in SPIRED-Fitness. In practical protein functional analysis and design, users are supposed to take into consideration the specific limitations of different protein structure prediction methods and to choose the appropriate ones in their applications.

Here, we briefly discuss the reason for the imbalanced prediction behaviors of nowadays state-of-the-art single-sequence-based protein structure prediction methods. Clearly, all single-sequence-based methods are inferior to AlphaFold2 (Figure 2 and Supplementary Figure S2), suggesting that available protein language model are still incapable of providing the same level of information as the evolutionary information hidden in the MSA. Besides this point, the more severely imbalanced prediction behaviors of ESMFold and OmegaFold are more or less attributable to their model training strategies. ESMFold engages more than 10 million high-quality structures predicted by AlphaFold2 for model training. These extra structures are, however, selected based on pLDDT of AlphaFold2 without considering the topological information, therefore are unable to provide a fair and balanced representation over the protein topological space that gradually forms during the lengthy biological evolutionary process. OmegaFold, on the other hand, throws out a number of hard protein targets that hinder the descending of the mean loss at the main stage of its training process. This strategy, albeit successful at improving the average prediction score, disrupts the representativeness of training samples at the same time, which can explain why this method is especially powerful on small proteins but performs poorly for large ones, because large proteins are usually more difficult targets for structure prediction than small proteins. In contrast, SPIRED-Fitness sufficiently samples all of natural proteins in most of the training steps (*e.g.*, the third and fourth stages) of its training process (see Supplementary Table S12). Consequently, SPIRED-Fitness attains a more balanced prediction power at the level of protein folds.

1.7 GDFold2 optimization and the tradeoff between global and local structural quality

GDFold2¹ is a self-developed framework for folding proteins parallelly based on the geometric constraints predicted by deep learning models like SPIRED and trRosetta/RoseTTAFold. In the current GDFold2 pipeline, one structural optimization mode, namely the "model 2", is specifically designed for folding proteins (*i.e.* generating all-atom coordinates) based on the multiple sets of C_{α} coordinates predicted by SPIRED. The implementation details and folding examples can be found in the GDFold2 paper.

The average C_{α} TM-score usually will decrease slightly after the GDFold2 optimization, just because GDFold2 tries to establish all-atom coordinates that agree with both the geometric constraints predicted by SPIRED and the statistical constraints summarized from the high-resolution protein structures in the PDB database. Despite the high accuracy of SPIRED in general, the C_{α} coordinates output by SPIRED might slightly deviate from the statistical rules, *e.g.*, the statistical distributions for the distance between two adjacent C_{α} atoms, scalar angles between three consecutive C_{α} atoms and torsion angles between four consecutive C_{α} atoms. GDFold2 aims for providing a satisfactory solution for this inconsistency. Despite the slight drop of C_{α} TM-score or rise of global RMSD, after GDFold2 structural generation, the local structural quality is dramatically improved, including the distribution of backbone torsion angles, hydrogen bond networks, secondary structure contents, side-chain rotamers, *etc.* For instance, as shown in Supplementary Table S1, it is evident that IDDT, the metric that reflects the local structural quality, is uniformly improved after the GDFold2 coordinate optimization for both CAMEO and CASP15 targets.

Table S1. Comparison of global and local structural quality between SPIRED and SPIRED+GDFold2

Model	Test Set	TM-score	IDDT	RMSD
SPIRED(Cycle=1)	CAMEO	0.786	0.753	5.80
SPIRED(Cycle=1)+GDFold2	CAMEO	0.784	0.756	5.84
SPIRED(Cycle=4)	CAMEO	0.787	0.757	5.89
SPIRED(Cycle=4)+GDFold2	CAMEO	0.785	0.759	5.93
SPIRED(Cycle=1)	CASP15	0.610	0.656	12.09
SPIRED(Cycle=1)+GDFold2	CASP15	0.608	0.668	12.17
SPIRED(Cycle=4)	CASP15	0.614	0.675	12.62
SPIRED(Cycle=4)+GDFold2	CASP15	0.612	0.684	12.67

2 Supplementary Results for Fitness and Stability Prediction

2.1 End-to-end training improves fitness prediction for SPIRED-Fitness

Table S2. Performance of the fitness and structure prediction on the test set

Model	Fitness Test Set			CAMEO	
	$\rho_{\text{single and double}}$	ρ_{single}	ρ_{double}	TM-score	IDDT
SPIRED	N.A.	N.A.	N.A.	0.786	0.753
SPIRED+GDFold2	N.A.	N.A.	N.A.	0.784	0.756
SPIRED-Fitness (Stage 1)	0.830 ± 0.132	0.814 ± 0.136	0.763 ± 0.148	0.786	0.753
SPIRED-Fitness (Stage 2)	0.849 ± 0.131	0.835 ± 0.135	0.792 ± 0.143	0.779	0.740
SPIRED-Fitness + GDFold2	0.849 ± 0.131	0.835 ± 0.135	0.792 ± 0.143	0.777	0.743

The best performance in each metric, highlighted in bold, is chosen based on the mean values over tested proteins. Here, $\rho_{\text{single and double}}$ refers to the Spearman correlation coefficient evaluated on all single and double mutations of 485 proteins in the fitness test set, whereas ρ_{single} and ρ_{double} stand for the corresponding values for single and double mutations, respectively.

We use the fitness test set (485 proteins) described in **Methods** to evaluate the performance of fitness prediction. The CAMEO dataset (680 proteins) is utilized to assess the model performance on structure prediction without recycling (*i.e.* Cycle = 1). As shown in Supplementary Table S2, after the end-to-end training, quality of the structural models provided by SPIRED-Fitness (Stage 2) drops marginally in comparison to the original SPIRED, which indicates that the end-to-end training process is not biased by the limited number of proteins in the fitness training set.

As described in **Methods**, in the first training stage of SPIRED-Fitness, parameters of SPIRED are frozen, and only the Fitness Module is updated. At the end of the first training stage, the Spearman correlation coefficient of single + double mutations in the fitness test set reaches 0.830. After the completion of the second training stage of SPIRED-Fitness, where both SPIRED and the Fitness module are allowed to update their parameters, the Spearman correlation coefficient of single + double mutations increases to 0.850. Regarding single mutations, the rise in Spearman correlation coefficient is approximately 2%, while for double mutations, it is about 3%. The TM-score of structures predicted by SPIRED-Fitness only decreases by < 0.01 compared to SPIRED, and the IDDT decreases by 0.013. Therefore, the end-to-end training has minimal impact on the structure prediction performance of SPIRED-Fitness. Considering the improvement in fitness prediction, this is an acceptable tradeoff.

2.2 Performance comparison of SPIRED-Fitness and other models

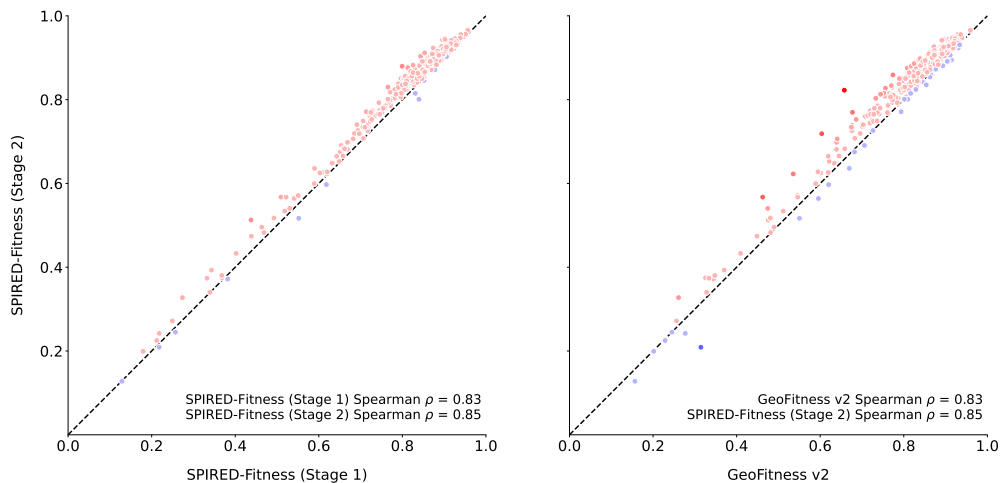


Figure S7. Protein-wise comparison between SPIRED-Fitness and GeoFitness on single and double mutations. The vertical axis represents SPIRED-Fitness (Stage 2), while the horizontal axis stands for SPIRED-Fitness (Stage 1) and GeoFitness v2. Each data point represents the Spearman correlation coefficient of the corresponding protein in the dataset. Data points are colored red if the Spearman correlation coefficient of SPIRED-Fitness (Stage 2) is higher than that of the other models but are colored blue otherwise, with the magnitude of difference indicated by the degree of opaqueness. The test set comprises a total of 485 proteins. Source data are provided as a Source Data file.

The network architecture of the Fitness Module of SPIRED-Fitness is almost identical to that of GeoFitness v2. Therefore, the principal difference between SPIRED-Fitness (Stage 1) and GeoFitness v2 lies in the module or model used for generating the structural features: SPIRED in SPIRED-Fitness (Stage 1) vs. AlphaFold2 in GeoFitness v2. Considering these facts, the advantage of SPIRED-Fitness (Stage 2) over both SPIRED-Fitness (Stage 1) and GeoFitness v2 demonstrates the positive effect of the end-to-end training on the performance of fitness prediction models.

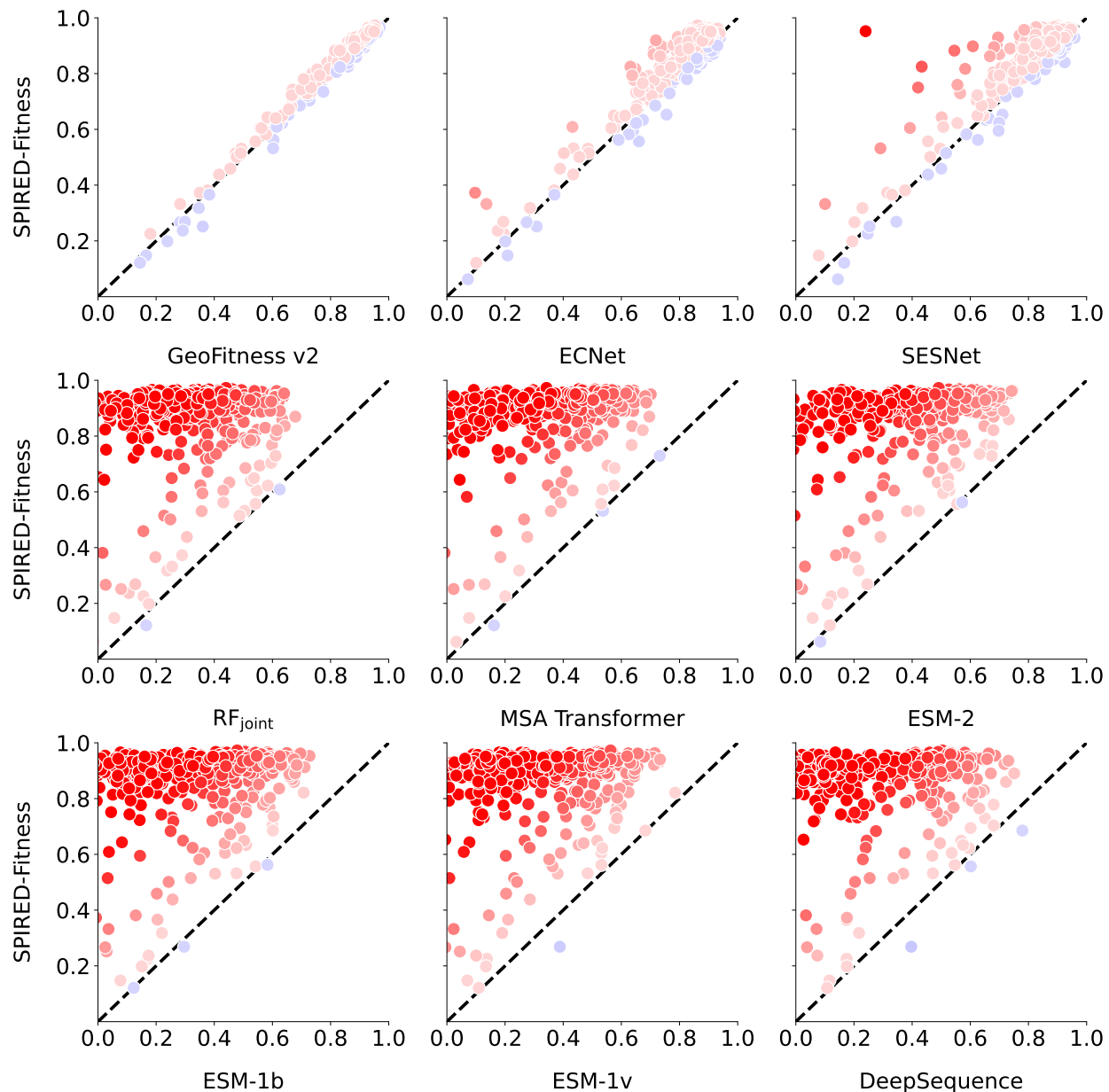


Figure S8. Protein-wise comparison of SPIRED-Fitness with other models on single mutations. The vertical axis represents SPIRED-Fitness, while the horizontal axis stands for different models. Each data point represents the Spearman correlation coefficient of the corresponding protein in the dataset. Data points are colored red if the Spearman correlation coefficient of SPIRED-Fitness is higher than that of the other models but are colored blue otherwise, with the magnitude of difference indicated by the degree of opaqueness. The test set comprises a total of 485 proteins. Source data are provided as a Source Data file.

2.3 Evaluation of fitness predictors on proteolytic and non-proteolytic data

Table S3. Fitness prediction performance on proteolytic and non-proteolytic data

Model	Proteolytic Data			Non-Proteolytic Data		
	Single	Double	Single+Double	Single	Double	Single+Double
ECNet	0.86 $\pm$ 0.05	0.88 $\pm$ 0.14	0.88 $\pm$ 0.05	0.60 $\pm$ 0.23	0.62 $\pm$ 0.28	0.62 $\pm$ 0.22
GeoFitness v2	0.85 $\pm$ 0.06	0.76 $\pm$ 0.14	0.87 $\pm$ 0.05	0.58 $\pm$ 0.19	0.48 $\pm$ 0.20	0.59 $\pm$ 0.19
SPIRED-Fitness (Stage 1)	0.85 $\pm$ 0.06	0.78 $\pm$ 0.13	0.87 $\pm$ 0.05	0.58 $\pm$ 0.20	0.45 $\pm$ 0.19	0.60 $\pm$ 0.19
SPIRED-Fitness (Stage 2)	0.88 $\pm$ 0.05	0.81 $\pm$ 0.12	0.89 $\pm$ 0.05	0.60 $\pm$ 0.20	0.47 $\pm$ 0.19	0.62 $\pm$ 0.19
SPIRED-Fitness finetuned	0.87 $\pm$ 0.05	0.81 $\pm$ 0.12	0.89 $\pm$ 0.05	0.60 $\pm$ 0.21	0.48 $\pm$ 0.19	0.62 $\pm$ 0.20

The values in the table are presented as mean $\pm$ standard-deviation for the Spearman correlation coefficient calculated over all tested proteins. The winner in each metric is chosen based on the mean value and is highlighted in bold. Here, the proteolytic data refer to the 412 proteins from the cDNA proteolysis dataset and the non-proteolytic data refer to the 73 proteins from the MaveDB database and DeepSequence dataset. The "SPIRED-Fitness finetuned" refers to the model that is derived from finetuning the original SPIRED-Fitness (Stage 2) on non-proteolytic data.

Table S4. Sample size of proteolytic and non-proteolytic data

Total	Proteolytic Data			Non-Proteolytic Data		
	Single	Double	Single+Double	Single	Double	Single+Double
958,816	389,068	138,136	527,204	305,747	125,865	431,612

The values in the table represent the mutation numbers of corresponding mutation types and datasets.

Table S5. Fitness prediction performance on proteolytic and non-proteolytic single mutation data

Model	Proteolytic Data	Non-Proteolytic Data	All Data
DeepSequence	0.23 $\pm$ 0.23	0.38 $\pm$ 0.30	0.26 $\pm$ 0.23
ESM-1v	0.30 $\pm$ 0.20	0.35 $\pm$ 0.21	0.31 $\pm$ 0.20
MSA Transformer	0.32 $\pm$ 0.19	0.34 $\pm$ 0.20	0.32 $\pm$ 0.19
RF _{joint}	0.31 $\pm$ 0.17	0.37 $\pm$ 0.19	0.32 $\pm$ 0.17
ESM-1b	0.33 $\pm$ 0.21	0.34 $\pm$ 0.21	0.33 $\pm$ 0.21
ESM-2	0.41 $\pm$ 0.19	0.35 $\pm$ 0.22	0.40 $\pm$ 0.20
SESNet	0.87 $\pm$ 0.06	0.57 $\pm$ 0.21	0.83 $\pm$ 0.14
ECNet	0.87 $\pm$ 0.05	0.60 $\pm$ 0.22	0.83 $\pm$ 0.14
GeoFitness v2	0.90 $\pm$ 0.05	0.61 $\pm$ 0.21	0.85 $\pm$ 0.14
SPIRED-Fitness	0.91 $\pm$ 0.04	0.62 $\pm$ 0.21	0.87 $\pm$ 0.14

The values in the table are presented as mean $\pm$ standard-deviation for the Spearman correlation coefficient calculated over all tested proteins. The winner in each metric is chosen based on the mean value and is highlighted in bold. Here, the proteolytic data refer to the 412 proteins from the cDNA proteolysis dataset and the non-proteolytic data refer to the 73 proteins from the MaveDB database and DeepSequence dataset. All models in the table are trained on single mutation data only.

2.4 Benchmark test on ProteinGym

2.4.1 Zero-shot setting

We developed an iterative approach to infer the multiple mutational effects using SPIRED-Fitness. Considering the relatively lower confidence of the raw prediction by SPIRED-Fitness on double mutational effects than single ones (see Supplementary Table S3), we mainly adopt the single mutational scores reported by SPIRED-Fitness for the inference of higher-order effects. As shown in Supplementary Figure S9, we take the triple mutational effects as an example. For a variant carrying three mutation sites, we first run SPIRED-Fitness on the wild-type sequence and identify the candidate single mutation that elicits the largest magnitude of fitness change as predicted by SPIRED-Fitness. Next, we feed the sequence carrying this single mutation to SPIRED-Fitness and obtain the predicted fitness change for the second site. Again, between the two candidate mutation sites, we select the one predicted of larger magnitude of fitness change as the second mutation. We then feed the sequence carrying the first two mutations to SPIRED-Fitness for prediction on the effect of the third mutation. The fitness change scores reported in the above three iterations are summed up to quantify the triple mutational effects. In principle, fitness change caused by an arbitrary number of mutations, including double, triple and higher-order ones, could be accomplished by repeating the above process. Moreover, this iterative process is non-equivalent to the simple summation of site scores in the generic protein language models, since the scores predicted in the second and later iterations are generated by conditioning on prior mutations. Hence, the correlation between mutation sites is intrinsically considered by this means. Notably, our iterative protocol is time efficient, considering the fast running speed of SPIRED-Fitness.

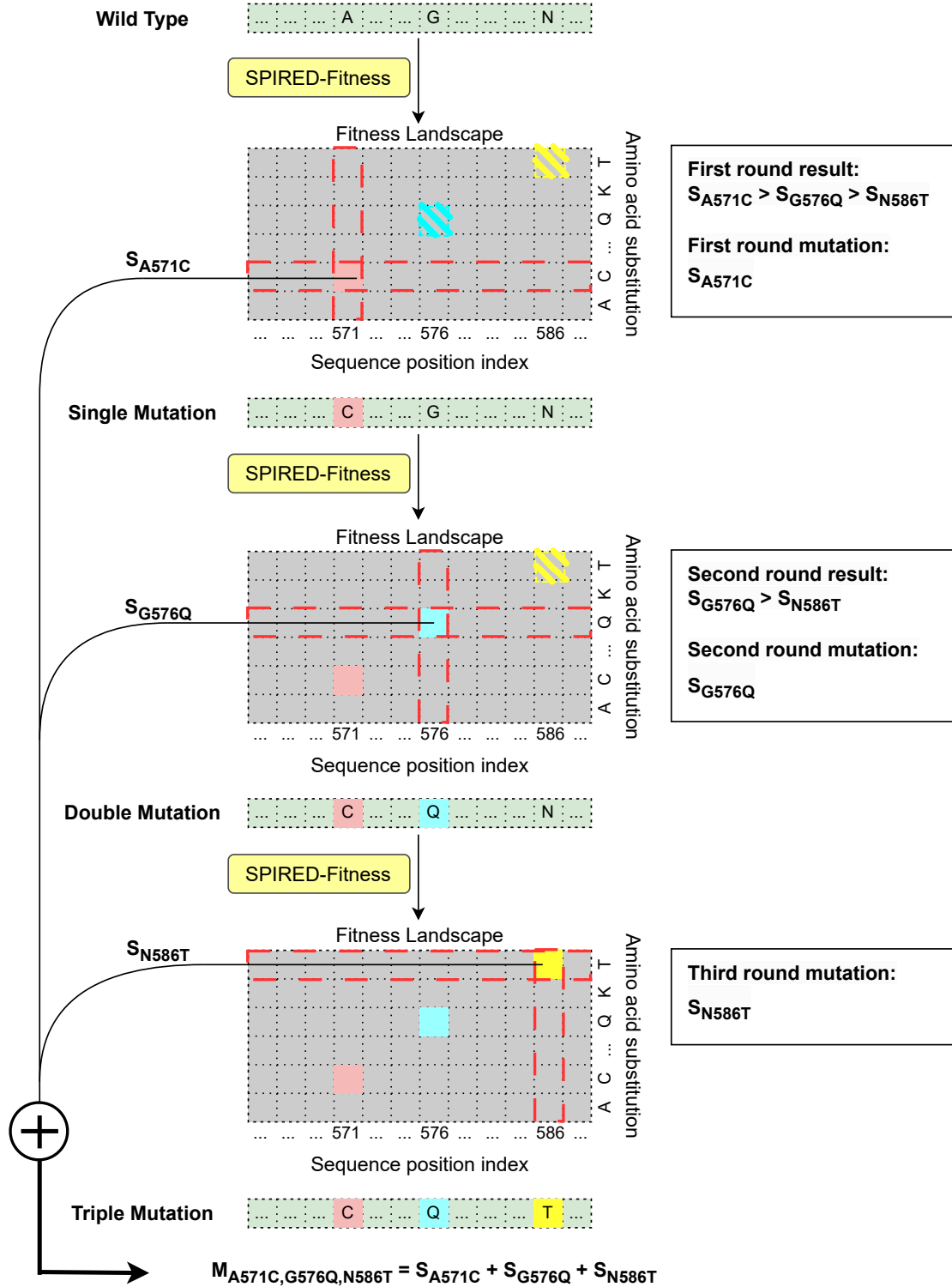


Figure S9. Multiple mutation effect inference pipeline of SPIRED-Fitness for the zero-shot setting in Prote-inGym. Here, we show an example for the inference of triple mutational effects. "S" in the figure means single mutation score while "M" means multiple mutation score.

Table S6. Assays involved in the zero-shot benchmark on the multi-mutation effects by SPIRED-Fitness

Protein Name	Length	Mutation Number			
		Double	Triple	Quadruple	Total
A4_HUMAN_Seuma_2022	770	14015	NA	NA	14015
CAPSD_AAV2S_Sinai_2021	735	10833	6906	6646	24385
D7PM05_CLYGR_Somermeyer_2022	235	10148	6315	3417	19880
F7YBW7_MESOW_Ding_2023	93	86	603	2159	2848
F7YBW8_MESOW_Aakre_2015	93	499	2798	5858	9155
GCN4_YEAST_Staller_2018	281	55	149	380	584
GRB2_HUMAN_Faure_2021	217	62332	NA	NA	62332
PHOT_CHLRE_Chen_2023	118	176	978	3565	4719
Q6WV13_9MAXI_Somermeyer_2022	222	15992	8762	3575	28329

The assays are chosen from the test set for zero-shot prediction in ProteinGym (see Supplementary Data 4), based on the presence of multiple mutation data.

Table S7. Comparison of SPIRED-Fitness with other zero-shot predictors on ProteinGym multi-mutation data

Method	Double	Triple	Quadruple
<i>Structure-based</i>			
ProteinMPNN	0.14 $\pm$ 0.19	0.14 $\pm$ 0.10	0.18 $\pm$ 0.11
ESM-IF1	0.21 $\pm$ 0.30	0.23 $\pm$ 0.17	0.26 $\pm$ 0.19
ProtSSN (k30_h1280)	0.31 $\pm$ 0.24	0.31 $\pm$ 0.21	0.25 $\pm$ 0.20
MIF-ST	0.34 $\pm$ 0.20	0.34 $\pm$ 0.15	0.32 $\pm$ 0.17
SaProt (650M_AF2)	0.25 $\pm$ 0.25	0.26 $\pm$ 0.22	0.28 $\pm$ 0.23
ProtSSN (ensemble)	0.29 $\pm$ 0.27	0.31 $\pm$ 0.22	0.25 $\pm$ 0.21
GVP-MSA	0.40 $\pm$ 0.24	0.38 $\pm$ 0.21	0.31 $\pm$ 0.25
<i>MSA-based</i>			
WaveNet	0.26 $\pm$ 0.23	0.29 $\pm$ 0.20	0.24 $\pm$ 0.19
Site-Independent	0.21 $\pm$ 0.24	0.24 $\pm$ 0.15	0.27 $\pm$ 0.13
DeepSequence (single)	0.33 $\pm$ 0.26	0.33 $\pm$ 0.20	0.31 $\pm$ 0.20
EVmutation	0.37 $\pm$ 0.26	0.38 $\pm$ 0.14	0.31 $\pm$ 0.16
DeepSequence (ensemble)	0.37 $\pm$ 0.29	0.37 $\pm$ 0.19	0.33 $\pm$ 0.19
MSA Transformer (single)	0.39 $\pm$ 0.21	0.43 $\pm$ 0.15	0.38 $\pm$ 0.20
MSA Transformer (ensemble)	0.39 $\pm$ 0.20	0.44 $\pm$ 0.14	0.38 $\pm$ 0.20
EVE (single)	0.33 $\pm$ 0.24	0.35 $\pm$ 0.14	0.30 $\pm$ 0.15
Tranception L	0.40 $\pm$ 0.23	0.39 $\pm$ 0.14	0.33 $\pm$ 0.17
EVE (ensemble)	0.34 $\pm$ 0.26	0.36 $\pm$ 0.14	0.30 $\pm$ 0.15
GEMME	0.33 $\pm$ 0.23	0.36 $\pm$ 0.12	0.32 $\pm$ 0.17
TranceptEVE L	0.38 $\pm$ 0.23	0.39 $\pm$ 0.12	0.32 $\pm$ 0.13
<i>Single sequence-based</i>			
UniRep	0.15 $\pm$ 0.27	0.17 $\pm$ 0.21	0.21 $\pm$ 0.21
ProtGPT2	0.15 $\pm$ 0.21	0.07 $\pm$ 0.11	0.07 $\pm$ 0.09
ESM-1b	0.23 $\pm$ 0.25	0.23 $\pm$ 0.26	0.19 $\pm$ 0.24
CARP (640M)	0.27 $\pm$ 0.23	0.24 $\pm$ 0.23	0.22 $\pm$ 0.22
RITA XL	0.16 $\pm$ 0.24	0.16 $\pm$ 0.21	0.20 $\pm$ 0.20
ESM-1v (single)	0.21 $\pm$ 0.25	0.23 $\pm$ 0.24	0.18 $\pm$ 0.22
ESM-1v (ensemble)	0.23 $\pm$ 0.26	0.24 $\pm$ 0.25	0.20 $\pm$ 0.24
ProGen2 XL	0.27 $\pm$ 0.26	0.30 $\pm$ 0.19	0.21 $\pm$ 0.18
ESM-2 (15B)	0.22 $\pm$ 0.31	0.28 $\pm$ 0.28	0.18 $\pm$ 0.25
VESPA	0.31 $\pm$ 0.20	0.38 $\pm$ 0.19	0.32 $\pm$ 0.18
SPIRED-Fitness	0.40 $\pm$ 0.21	0.36 $\pm$ 0.20	0.28 $\pm$ 0.22

The metric, *i.e.* Spearman correlation coefficient, is calculated within each protein and then reported as mean $\pm$ standard-deviation over all tested proteins. All prediction methods are classified into a number of categories (shown in italic) and the method with the best mean performance within each category is highlighted in bold for each metric. Results of GVP-MSA are calculated using the zero-shot model provided in the paper², whereas results of the other methods are directly obtained from the ProteinGym official website.

2.4.2 Supervised setting

Table S8. Comparison of SPIRED-Fitness with ProteinGym supervised predictors on single mutation data

Method	Average	<i>Random</i>	<i>Modulo</i>	<i>Contiguous</i>
<i>MSA-based</i>				
DeepSequence + OHE	0.46 ± 0.14	0.55 ± 0.15	0.41 ± 0.16	0.41 ± 0.17
MSA Transformer + OHE	0.47 ± 0.14	0.55 ± 0.14	0.42 ± 0.17	0.42 ± 0.17
Tranception + OHE	0.47 ± 0.13	0.55 ± 0.14	0.42 ± 0.15	0.43 ± 0.15
TranceptEVE + OHE	0.49 ± 0.13	0.57 ± 0.14	0.45 ± 0.15	0.45 ± 0.15
Tranception Embeddings	0.59 ± 0.14	0.72 ± 0.17	0.53 ± 0.14	0.50 ± 0.13
MSA Transformer Embeddings	0.60 ± 0.17	0.68 ± 0.20	0.57 ± 0.17	0.56 ± 0.17
ProteinNPT	0.64 ± 0.16	0.76 ± 0.18	0.60 ± 0.17	0.58 ± 0.17
<i>Single sequence-based</i>				
OHE	0.23 ± 0.09	0.61 ± 0.17	0.02 ± 0.09	0.07 ± 0.15
ESM-1v + OHE	0.44 ± 0.16	0.54 ± 0.16	0.39 ± 0.18	0.39 ± 0.18
ESM-1v Embeddings	0.58 ± 0.18	0.68 ± 0.21	0.54 ± 0.17	0.51 ± 0.18
SPIRED-Fitness	0.61 ± 0.15	0.73 ± 0.16	0.59 ± 0.15	0.52 ± 0.17

The metric, *i.e.* Spearman correlation coefficient, is calculated within each assay and then reported as mean ± standard-deviation over the total of 201 tested assays. The "Average" column stands for the average Spearman correlation coefficient of the 3 different cross-validation schemes: *Random*, *Modulo* and *Contiguous*. All prediction methods are classified into a number of categories (shown in italic). For each metric, the winner among all methods is chosen based on the mean value and is highlighted in bold. The results of the methods other than SPIRED-Fitness are directly obtained from the ProteinGym official website.

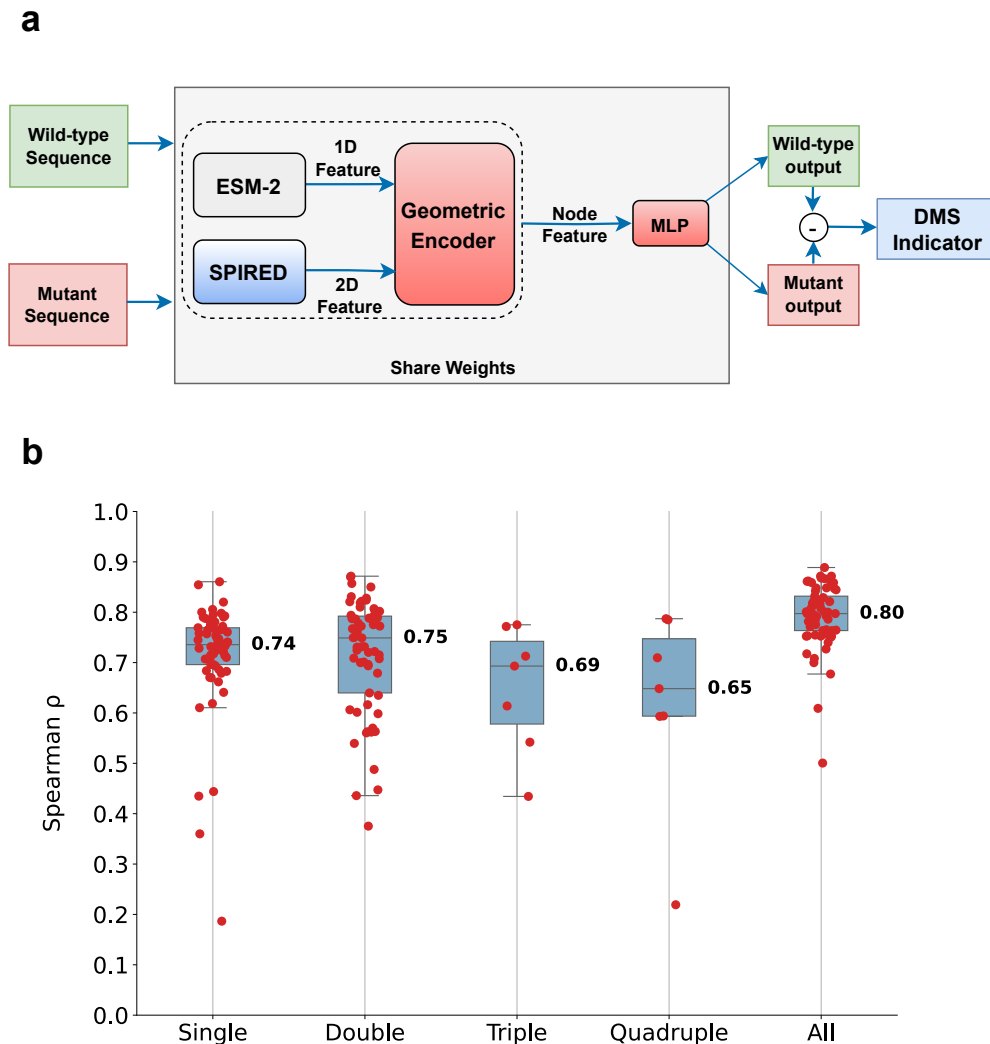


Figure S10. The modified SPIRED-Fitness for the prediction of multiple mutational effects in the supervised setting of ProteinGym. **a** The overall model architecture is similar to the design of SPIRED-Stab (see Figure 1e). Specifically, the modules encircled by the dashed line are directly inherited from the original SPIRED-Fitness, with the appended MLP and output layers for the prediction of changes in any DMS indicator. Since the wild-type and mutant sequences are fed into the overall network with shared weights and the difference of their outputs are directly used for the prediction of fitness change, this modified architecture allows the prediction of fitness change caused by an arbitrary number of mutations, not limited to single or double ones. **b** Evaluation of SPIRED-Fitness on the multiple mutation data (see Supplementary Data 5). The boxplot displays the distributions of the Spearman correlation coefficients among all of the 61 assays for various mutation types (*i.e.* single, double, triple, quadruple and all mutations), where each red dot stands for the value of one individual assay, the center line represents the median (with the value marked aside), the box limits correspond to the upper and lower quartiles and the whiskers extend to 1.5 inter-quartile range. Source data are provided as a Source Data file.

2.5 Supplementary results for the SPIRED-Stab evaluation

Table S9. Comparison of SPIRED-Stab with other $\Delta\Delta G$ predictors on the S461 dataset

Method	ρ	τ	r	Top-5	Top-10
<i>Structure-based</i>					
SDM	0.32 $\pm$ 0.36	0.24 $\pm$ 0.28	0.36 $\pm$ 0.38	0.40 $\pm$ 0.27	0.58 $\pm$ 0.23
ThermoNet	0.40 $\pm$ 0.26	0.29 $\pm$ 0.19	0.50 $\pm$ 0.20	0.35 $\pm$ 0.19	0.55 $\pm$ 0.24
mCSM	0.39 $\pm$ 0.32	0.29 $\pm$ 0.24	0.44 $\pm$ 0.31	0.40 $\pm$ 0.17	0.56 $\pm$ 0.21
Dynamut	0.39 $\pm$ 0.35	0.28 $\pm$ 0.27	0.36 $\pm$ 0.38	0.32 $\pm$ 0.30	0.53 $\pm$ 0.28
I-Mutant3.0	0.40 $\pm$ 0.30	0.29 $\pm$ 0.23	0.44 $\pm$ 0.31	0.38 $\pm$ 0.16	0.56 $\pm$ 0.22
MAESTRO	0.49 $\pm$ 0.23	0.36 $\pm$ 0.18	0.52 $\pm$ 0.26	0.42 $\pm$ 0.13	0.60 $\pm$ 0.18
DUET	0.45 $\pm$ 0.31	0.33 $\pm$ 0.25	0.50 $\pm$ 0.29	0.43 $\pm$ 0.21	0.59 $\pm$ 0.22
ACDC-NN	0.45 $\pm$ 0.34	0.34 $\pm$ 0.25	0.47 $\pm$ 0.39	0.42 $\pm$ 0.22	0.58 $\pm$ 0.26
FoldX	0.42 $\pm$ 0.42	0.31 $\pm$ 0.33	0.41 $\pm$ 0.40	0.38 $\pm$ 0.29	0.57 $\pm$ 0.23
PoPMuSiC	0.51 $\pm$ 0.24	0.38 $\pm$ 0.18	0.54 $\pm$ 0.26	0.45 $\pm$ 0.23	0.65 $\pm$ 0.16
DDGun3D	0.51 $\pm$ 0.32	0.39 $\pm$ 0.24	0.55 $\pm$ 0.32	0.48 $\pm$ 0.18	0.66 $\pm$ 0.21
PremPS	0.51 $\pm$ 0.32	0.39 $\pm$ 0.25	0.56 $\pm$ 0.28	0.43 $\pm$ 0.19	0.63 $\pm$ 0.24
DDMut	0.48 $\pm$ 0.40	0.37 $\pm$ 0.31	0.52 $\pm$ 0.39	0.42 $\pm$ 0.26	0.60 $\pm$ 0.19
INPS3D	0.54 $\pm$ 0.25	0.40 $\pm$ 0.20	0.56 $\pm$ 0.25	0.50 $\pm$ 0.20	0.63 $\pm$ 0.18
GeoDDG-3D (old)	0.59 $\pm$ 0.23	0.44 $\pm$ 0.20	0.59 $\pm$ 0.23	0.62 $\pm$ 0.16	0.71 $\pm$ 0.16
RaSP	0.56 $\pm$ 0.30	0.43 $\pm$ 0.23	0.54 $\pm$ 0.28	0.43 $\pm$ 0.22	0.63 $\pm$ 0.23
Pythia	0.64 $\pm$ 0.19	0.48 $\pm$ 0.16	0.63 $\pm$ 0.18	0.53 $\pm$ 0.21	0.65 $\pm$ 0.18
ThermoMPNN	0.63 $\pm$ 0.32	0.48 $\pm$ 0.26	0.60 $\pm$ 0.31	0.48 $\pm$ 0.23	0.66 $\pm$ 0.20
<i>Sequence-based</i>					
MUPro	0.23 $\pm$ 0.32	0.17 $\pm$ 0.23	0.24 $\pm$ 0.36	0.30 $\pm$ 0.22	0.52 $\pm$ 0.24
I-Mutant3.0-Seq	0.26 $\pm$ 0.39	0.19 $\pm$ 0.28	0.30 $\pm$ 0.36	0.32 $\pm$ 0.26	0.53 $\pm$ 0.24
SAAFEC-SEQ	0.40 $\pm$ 0.30	0.29 $\pm$ 0.23	0.44 $\pm$ 0.31	0.38 $\pm$ 0.16	0.56 $\pm$ 0.22
ACDC-NN-Seq	0.43 $\pm$ 0.35	0.32 $\pm$ 0.26	0.49 $\pm$ 0.35	0.45 $\pm$ 0.12	0.59 $\pm$ 0.21
INPS-Seq	0.47 $\pm$ 0.30	0.34 $\pm$ 0.23	0.51 $\pm$ 0.26	0.48 $\pm$ 0.13	0.61 $\pm$ 0.23
DDGun	0.52 $\pm$ 0.30	0.39 $\pm$ 0.23	0.52 $\pm$ 0.33	0.43 $\pm$ 0.17	0.58 $\pm$ 0.25
PROSTATA	0.53 $\pm$ 0.33	0.40 $\pm$ 0.25	0.55 $\pm$ 0.34	0.50 $\pm$ 0.20	0.66 $\pm$ 0.23
Mutate Everything (ESM-2)	0.63 $\pm$ 0.19	0.46 $\pm$ 0.16	0.62 $\pm$ 0.18	0.42 $\pm$ 0.22	0.68 $\pm$ 0.18
GeoDDG-Seq (old)	0.60 $\pm$ 0.23	0.46 $\pm$ 0.18	0.63 $\pm$ 0.20	0.55 $\pm$ 0.19	0.70 $\pm$ 0.20
SPIRED-Fitness (zero-shot)	0.58 $\pm$ 0.33	0.45 $\pm$ 0.25	0.59 $\pm$ 0.30	0.57 $\pm$ 0.17	0.68 $\pm$ 0.19
GeoStab v2	0.68 $\pm$ 0.31	0.54 $\pm$ 0.25	0.67 $\pm$ 0.31	0.60 $\pm$ 0.21	0.76 $\pm$ 0.18
SPIRED-Stab	0.71 $\pm$ 0.24	0.57 $\pm$ 0.21	0.69 $\pm$ 0.21	0.60 $\pm$ 0.21	0.73 $\pm$ 0.18

All metrics are calculated within each protein and then reported as mean $\pm$ standard-deviation over all tested proteins. All prediction methods are classified into a number of categories (shown in italic). For each metric, the winner among all methods is chosen based on the mean value and is highlighted in bold. Here, ρ , τ and r denote Spearman, Kendall and Pearson correlation coefficients, respectively, while Top-5 and Top-10 refer to the Top 5 precision and Top 10 precision, respectively, as defined in **Methods**.

2.5.1 Evaluation of SPIRED-Stab using conventional metrics and in the cross-protein manner

Table S10. Comparison of SPIRED-Stab with existing models on the S461 dataset in the cross-protein manner

Method	Direct		Inverse		Antisymmetry	
	ρ/r	RMSE/MAE	ρ/r	RMSE/MAE	r_{d-i}	$\langle \delta \rangle$
<i>Structure-based</i>						
SDM	0.51 / 0.56	1.33 / 1.01	0.11 / 0.11	1.98 / 1.51	-0.35	0.52
ThermoNet	0.49 / 0.56	1.23 / 0.93	0.43 / 0.52	1.31 / 1.01	-0.86	0.16
mCSM	0.51 / 0.53	1.07 / 0.80	0.12 / 0.18	2.17 / 1.81	-0.09	0.85
Dynamut	0.44 / 0.50	1.27 / 0.96	0.43 / 0.44	1.36 / 1.01	-0.61	0.29
I-Mutant3.0	0.48 / 0.49	1.12 / 0.83	0.11 / 0.13	2.19 / 1.83	-0.04	0.81
MAESTRO	0.60 / 0.63	1.04 / 0.78	0.17 / 0.20	1.92 / 1.53	-0.22	0.68
DUET	0.57 / 0.59	1.06 / 0.77	0.15 / 0.18	1.97 / 1.59	-0.15	0.72
ACDC-NN	0.57 / 0.61	1.06 / 0.77	0.56 / 0.60	1.07 / 0.78	-0.98	0.04
FoldX	0.35 / 0.22	2.23 / 1.38	0.45 / 0.41	1.74 / 1.14	-0.22	0.71
PoPMuSiC	0.61 / 0.62	1.01 / 0.76	0.25 / 0.29	1.89 / 1.52	-0.30	0.69
DDGun3D	0.58 / 0.64	1.10 / 0.80	0.55 / 0.59	1.17 / 0.85	-0.96	0.10
PremPS	0.60 / 0.63	1.03 / 0.79	0.58 / 0.63	0.98 / 0.73	-0.86	0.20
DDMut	0.60 / 0.62	1.03 / 0.77	0.57 / 0.58	1.08 / 0.82	-0.94	0.13
INPS3D	0.60 / 0.62	1.01 / 0.75	0.39 / 0.35	1.48 / 1.12	-0.49	0.48
GeoDDG-3D (old)	0.67 / 0.67	1.01 / 0.72	0.67 / 0.67	1.01 / 0.72	-1.00	0.00
RaSP	0.58 / 0.55	1.17 / 0.81	- / -	- / -	-	-
Pythia	0.63 / 0.62	8.62 / 6.50	- / -	- / -	-	-
ThermoMPNN	0.64 / 0.61	1.09 / 0.79	- / -	- / -	-	-
<i>Sequence-based</i>						
MUPro	0.37 / 0.40	1.17 / 0.91	0.27 / 0.29	2.24 / 1.93	-0.31	0.97
I-Mutant3.0-Seq	0.45 / 0.44	1.16 / 0.89	0.25 / 0.24	2.09 / 1.75	-0.43	0.79
SAAFEC-SEQ	0.48 / 0.49	1.12 / 0.83	0.01 / 0.02	2.29 / 1.92	-0.05	0.85
ACDC-NN-Seq	0.54 / 0.57	1.10 / 0.80	0.54 / 0.57	1.10 / 0.80	-1.00	0.00
INPS-Seq	0.55 / 0.56	1.10 / 0.82	0.54 / 0.55	1.12 / 0.83	-1.00	0.03
DDGun	0.58 / 0.59	1.26 / 0.95	0.55 / 0.55	1.32 / 0.97	-0.94	0.07
PROSTATA	0.65 / 0.66	1.01 / 0.73	0.65 / 0.66	1.01 / 0.72	-1.00	0.02
Mutate Everything (ESM-2)	0.63 / 0.62	1.03 / 0.75	- / -	- / -	-	-
GeoDDG-Seq (old)	0.68 / 0.69	0.98 / 0.73	0.68 / 0.69	0.98 / 0.73	-1.00	0.00
SPIRED-Fitness (zero-shot)	0.64 / 0.63	1.65 / 1.42	- / -	- / -	-	-
GeoStab v2	0.70 / 0.67	1.00 / 0.72	0.70 / 0.67	1.00 / 0.72	-1.00	0.00
SPIRED-Stab	0.70 / 0.69	0.99 / 0.70	0.70 / 0.69	0.99 / 0.70	-1.00	0.00

For each metric, the winner among all methods is chosen based on the mean value and is highlighted in bold. "Direct" refers to the mutations from the wild type to mutant, while "Inverse" means the corresponding mutations in the reverse direction. All metrics are calculated upon the data aggregated across the proteins and those denoted as "-" mean the model cannot provide results needed for the corresponding metric. Here, ρ and r denote Spearman and Pearson correlation coefficients, respectively, while RMSE and MAE refer to the root mean square error and mean absolute error, respectively. Definitions for the antisymmetry metrics could be found in our prior work³. Results of ThermoMPNN, RaSP, PROSTATA, Pythia and Mutate Everything (ESM-2) are derived by independent inference, while the raw results of the other models are taken from prior works^{3,4}.

2.6 Miscellany

2.6.1 Impact of train-test-splitting strategy on the fitness prediction by SPIRED-Fitness

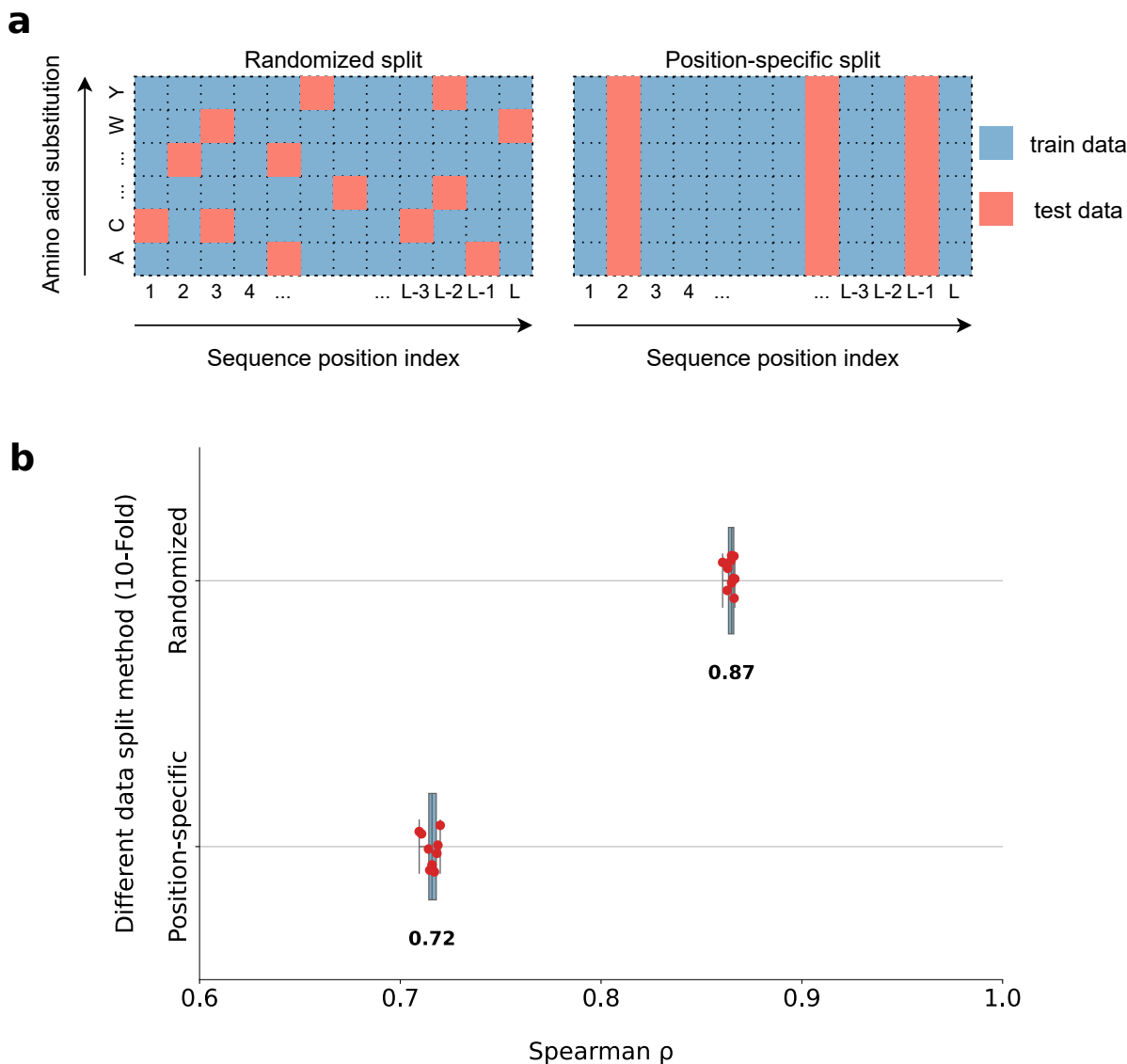


Figure S11. Performance evaluation of SPIRED-Fitness upon two data splitting strategies. **a** Schematic representation of two data splitting strategies. **b** Results of 10-fold cross validation on the two data splitting strategies. The boxplot is constructed from the results of 10 independent experiments. The horizontal axis represents different splitting strategies for DMS dataset. Each red dot in the graph represents the result of one individual experiment, *i.e.* the mean Spearman correlation coefficient of all proteins. The median value of each boxplot is marked aside. The box limits correspond to the upper and lower quartiles, whereas the whiskers extend to 1.5 inter-quartile range. Source data are provided as a Source Data file.

We tested the fitness prediction performance of the SPIRED-Fitness model on three different fitness data splitting strategies (Supplementary Figure S11 and Figure 5b). The first strategy randomly shuffles and splits all data with a ratio of training:validation:test = 7:1:2, while the second strategy splits data based on residue positions with the same ratio. The third strategy splits data based on proteins, with 70% of proteins for training, 10% of proteins for validation

and 20% of proteins for testing. Given the above results, we chose the randomized splitting strategy for the final training of SPIRED-Fitness model. Results of the third strategy could be regarded as independent experiments for validating the model generalizability in the scenario of zero-shot prediction.

2.6.2 Correlation between pLDDT and IDDT in the structure prediction by SPIRED/SPIRED-Fitness

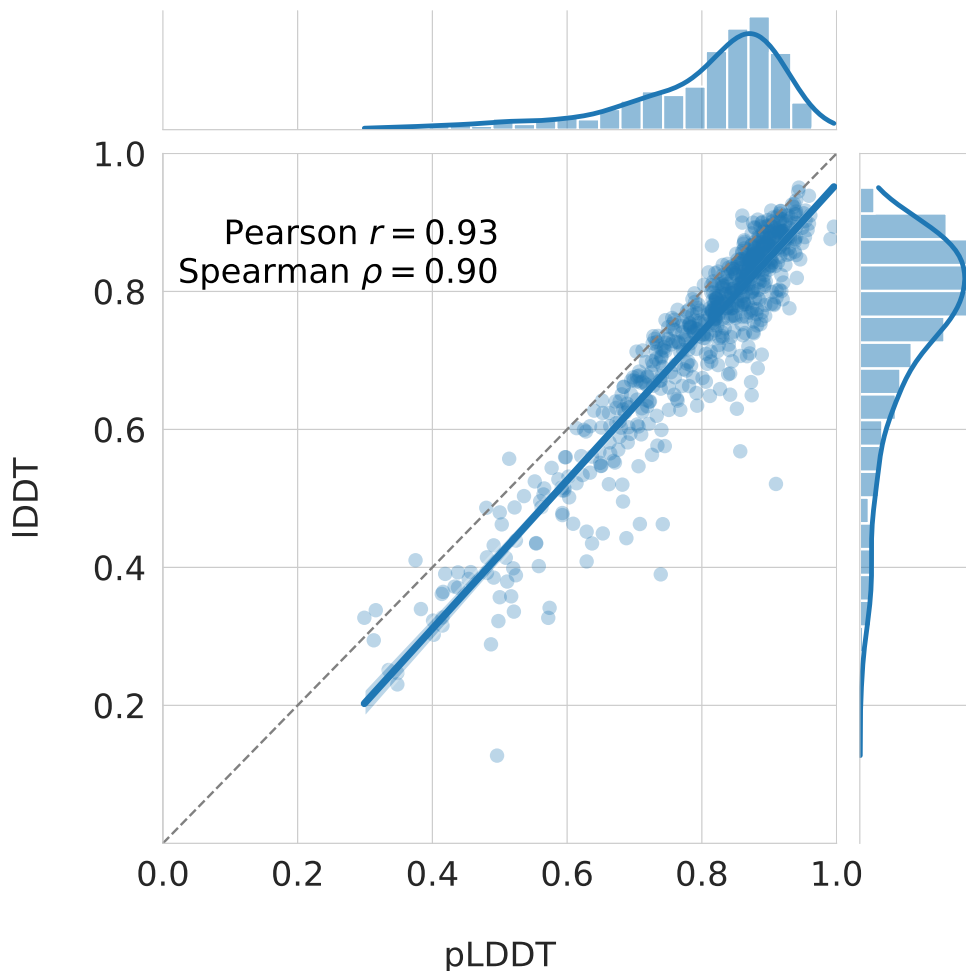


Figure S12. Correlation between pLDDT and IDDT for SPIRED-Fitness in the CAMEO dataset. The dataset is CAMEO test set (680 proteins, August 2022 to August 2023). Each data point represents a protein, with the average pLDDT (horizontal axis) and average IDDT (vertical axis) calculated based on the selection of the top 4 sets of C_α coordinates with the highest pLDDT values from the SPIRED-Fitness prediction. Source data are provided as a Source Data file.

For each protein of length L , SPIRED/SPIRED-Fitness can predict L sets of C_α coordinates, each with corresponding pLDDT values (a tensor of length L). For each protein target from the CAMEO and CASP15 sets, among the L sets of protein coordinates predicted by SPIRED/SPIRED-Fitness, we select the top 4 sets of coordinates of the highest pLDDT values and compute the average TM-score as well as average IDDT of them for comparison with other models. As for the evaluation of structural prediction upon SCOPe and CATH targets, we compare the one set of coordinates predicted of the highest pLDDT value for comparison with other models.

We explore the correlation between IDDT and pLDDT values of the predicted structures from SPIRED-Fitness using the CAMEO dataset, by calculating the average pLDDT and IDDT for top 4 sets of the predicted coordinates of the highest pLDDT values. Supplementary Figure S12 indicates the presence of a high Pearson correlation coefficient (0.93) and a high Spearman correlation coefficient (0.90) between the pLDDT and IDDT values for each protein, demonstrating that the pLDDT values output by SPIRED-Fitness are capable of effectively assessing the quality of predicted structures.

2.6.3 Correlation between $\Delta\Delta G$ and ΔT_m labels in the Dual Task Dataset of SPIRED-Stab

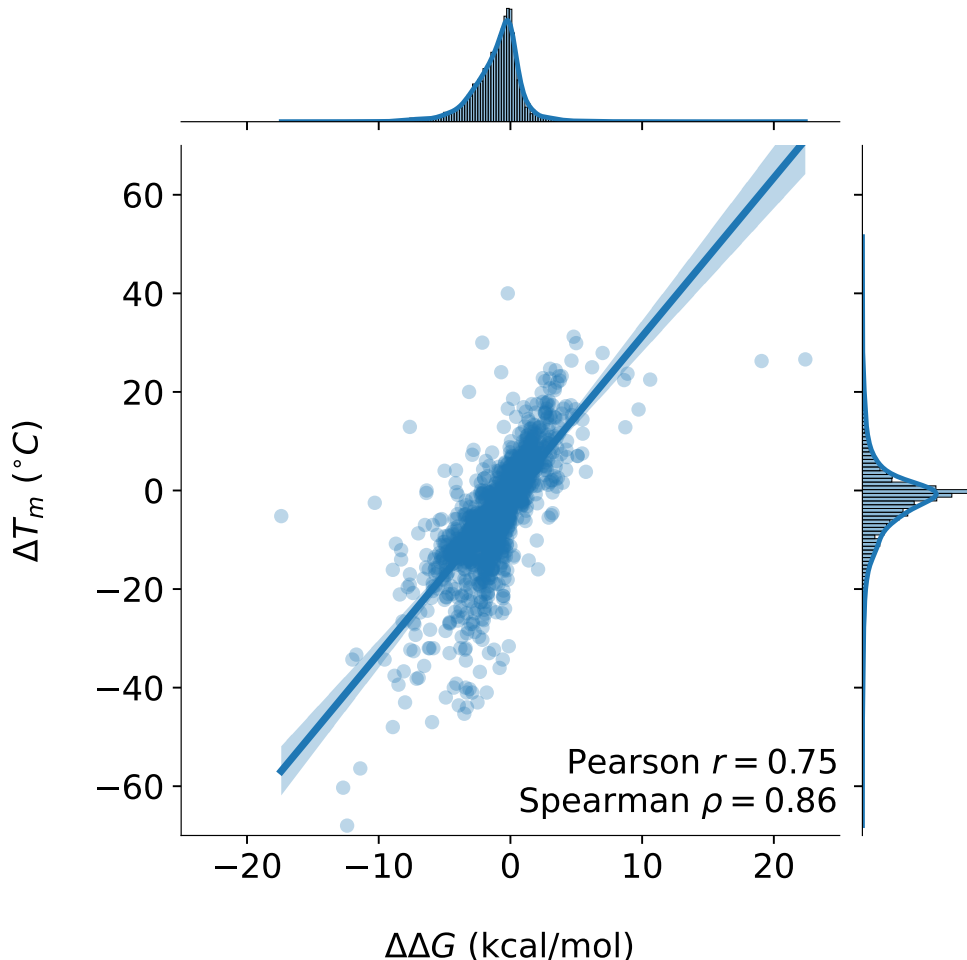


Figure S13. Correlation between $\Delta\Delta G$ and ΔT_m in the Dual Task Dataset. Each point represents one entry in the Dual Task Dataset with both $\Delta\Delta G$ and ΔT_m labels. Source data are provided as a Source Data file.

We evaluate the correlation between $\Delta\Delta G$ and ΔT_m in the Dual Task Dataset constructed in this work. As shown in Supplementary Figure S13, $\Delta\Delta G$ and ΔT_m data present a strong correlation, with the Pearson correlation coefficient equal to 0.75, a phenomenon that is fully expected, since both metrics are descriptors of the protein stability, although $\Delta\Delta G$ describes the resistance to denaturation at room temperature while ΔT_m reflects the resistance to thermal denaturation. Nevertheless, this observation provides supports for our dual task training strategy for SPIRED-Stab and Geo-Stab v2 in this work, *i.e.* optimization of the network parameters through the dual tasks of $\Delta\Delta G$ and ΔT_m prediction in each epoch.

3 Supplementary Information for Model and Training

3.1 Training information for SPIRED, OmegaFold and ESMFold

Table S11. Model parameters, training time and training samples

Model	Folding module parameters	Training time	Training samples		
			PDB chains	Protein domains	AlphaFold2 predictions
SPIRED	125M	85 GPU days (A100)	113,609	24,183 CATH v4.2 domains	N.A.
OmegaFold	124M	3456 GPU days (A100)	110,000	SCOP v1.75 domains	N.A.
ESMFold	690M	$\geq$ 896 GPU days (V100)	325,498	N.A.	13,477,259

3.2 Detailed training protocol for SPIRED

Table S12. SPIRED training protocol

	First stage	Second stage	Third stage	Fourth stage
Sequence crop size	256	256	256	350 ~ 420
Batch size	64	64	64	64
Learning rate	$10^{-3} \sim 5 \times 10^{-4}$	$5 \times 10^{-4} \sim 10^{-4}$	$10^{-4} \sim 5 \times 10^{-5}$	$5 \times 10^{-5} \sim 10^{-5}$
Sampling strategy	30% cluster PDB chains	Easy PDB chains ($L < 400, Res < 3\text{\AA}$)	PDB chains + CATH domains	PDB chains + CATH domains
Sample number	24K	63K	138K	138K
Update steps	10K	8K	23K	30K

L denotes the length of a protein, while Res refers to the resolution of experimental structures.

3.3 Brief introduction and training details of GeoFitness v2 and GeoStab v2

Brief introduction. In our prior work³, we developed a suite (named as GeoStab-suite) of three geometric-learning-based predictors, including GeoFitness, GeoDDG and GeoDTm, to predict the changes of protein fitness, $\Delta\Delta G$ and ΔT_m caused by mutations on the target proteins, respectively. GeoFitness reads in the sequence and structure (experimental or AlphaFold2-predicted) of the target protein and outputs the fitness landscape of all single mutations. Trained upon ~300,000 pieces of single mutational DMS data using the Soft Spearman Loss, GeoFitness is believed to learn the general mutational effects to some extent. The geometric encoder of the pre-trained GeoFitness model is then connected with downstream network to compose the GeoDDG and GeoDTm models that are trained by uniformly labeled $\Delta\Delta G$ and ΔT_m data, respectively. Based on the sources of input protein structures, GeoDDG and GeoDTm are further annotated by suffices of "-Seq" and "-3D" to denote that the structures are predicted by AlphaFold2 or solved experimentally, respectively.

Differences between the versions. Here, we refer to the GeoStab-suite in our prior work as version 1 and the updated version in this work as version 2. In the new version, we have made the following modifications:

1. Unlike the old version of GeoFitness v1 that only uses single mutational data to constrain the node embedding during training, GeoFitness v2 also includes double mutational data to constrain the edge embedding, in a similar way to SPIRED-Fitness (see Figure 1d). Accordingly, during the model training, the Soft Spearman Loss is applied to both single and double mutations, as shown in Equation 1.
2. Unlike the old version of GeoFitness v1 that only adopts MaveDB/DeepSequence datasets as training data, GeoFitness v2 enriches the large scale of cDNA proteolysis data into its training set.
3. Unlike the old version of GeoStab v1 that engages two separate models, GeoDDG and GeoDTm, to predict the $\Delta\Delta G$ and ΔT_m , respectively, GeoStab v2 unifies the $\Delta\Delta G$ and ΔT_m predictions as a dual task prediction job in one model (see Figure 1d).
4. GeoStab v2 is trained with AlphaFold2-predicted structures only. Therefore, suffices like "-Seq" and "-3D" are no longer needed in the new version.

Training details. We first trained GeoFitness v2 using both MaveDB/DeepSequence datasets and the cDNA proteolysis dataset, in a strategy very similar to SPIRED-Fitness. We then reutilized the geometric encoder of GeoFitness v2 in the GeoStab v2 model and trained this model with a generally similar but slightly different procedure to our prior work³. Specifically, in the first stage, we completely fixed the parameters in the GeoFitness module and used the whole cDNA proteolysis dataset described in **Methods** as the training dataset to optimize the other parameters in the GeoStab v2 model except the $\Delta\Delta G_{coef}$ and $\Delta T_{m_{coef}}$ parameters in the last output layer with an initial learning rate of 10^{-3} . In the second stage, we further finetuned our model using the Dual Task Dataset collected in this work with the Soft Spearman Loss. The learning rate was assigned just as described for SPIRED-Stab in **Methods**. Finally, when the model had learned to capture the ranking information, we adopted the MSE loss to tune $\Delta\Delta G_{coef}$ and $\Delta T_{m_{coef}}$ parameters in the last output layer with an initial learning rate of 10^{-2} in order to improve the model performance in predicting the absolute value of $\Delta\Delta G/\Delta T_m$.

4 Details for Structure Loss and Fitness Loss

4.1 Loss for training SPIRED

The loss calculation method is similar for the first three Folding Units. Taking the Unit1_Loss of the first Folding Unit as an example, the Unit1_Loss is composed of RD_Loss, CaDist_Loss, pLDDT_Loss and Clash_Loss. Each Folding Unit predicts two groups of structures (by two Pred-XYZ modules), and we calculate the loss for each group of predicted structures separately, resulting in RD_Loss_1, RD_Loss_2, CaDist_Loss_1, and CaDist_Loss_2. The calculation method is based on Algorithm 11 and Algorithm 13. When the label of C_α distance is ≤ 24 Å, the RD_Loss and CaDist Loss are calculated and then normalized by the maximal distance of 24 Å. However, pLDDT_Loss and Clash_Loss are calculated only for the last predicted coordinates, following Algorithm 14 and Algorithm 15.

$$\text{RD_Loss} = (\text{RD_Loss}_1 + \text{RD_Loss}_2) / (2 \times 24) \quad (1)$$

$$\text{CaDist_Loss} = (\text{CaDist_Loss}_1 + \text{CaDist_Loss}_2) / (2 \times 24) \quad (2)$$

$$\text{Unit1_Loss} = \text{RD_Loss} + 0.1 \times \text{CaDist_Loss} + \text{pLDDT_Loss} + \text{Clash_Loss} / 40 \quad (3)$$

For Folding Unit4, the calculation of RD_Loss for the first 4 groups of predicted structures is based on Algorithm 11. When the label of C_α distance is ≤ 32 Å, the RD_Loss and CaDist Loss are calculated and then normalized by the maximal distance of 32 Å. When calculating the RD_Loss for the last 2 groups of predicted structures, Algorithm 12 is referenced, with a random selection of 10% of RD_Loss values that are greater than 40 into consideration of loss evaluation.

$$\begin{aligned} \text{RD_Loss} = & (\text{RD_Loss}_1 + \text{RD_Loss}_2 + \text{RD_Loss}_3 + \text{RD_Loss}_4) / (4 \times 32) \\ & + (\text{RD_Loss}_5 + \text{RD_Loss}_6) / (2 \times 40) \end{aligned} \quad (4)$$

$$\begin{aligned} \text{CaDist_Loss} = & (\text{CaDist_Loss}_1 + \text{CaDist_Loss}_2 + \text{CaDist_Loss}_3 + \text{CaDist_Loss}_4) / (4 \times 32) \\ & + (\text{CaDist_Loss}_5 + \text{CaDist_Loss}_6) / (2 \times 40) \end{aligned} \quad (5)$$

$$\text{Unit4_Loss} = \text{RD_Loss} + 0.1 \times \text{CaDist_Loss} + \text{pLDDT_Loss} + \text{Clash_Loss} / 40 \quad (6)$$

The total Loss is composed of four Folding Unit Losses, the Cross Entropy Loss of C_β distance distribution and that of dihedral/scalar angles (CE_Loss) as well as the L_1 Loss of torsional angles (Torsion_Angle_Loss). L is the length of the protein. The hyper-parameter β is 0.1 in the first two training stages shown in Supplementary Table S12, and it is changed to 0.01 in the last two training stages.

$$\text{Folding_Unit_Loss} = \text{Unit1_Loss} + \text{Unit2_Loss} + \text{Unit3_Loss} + \text{Unit4_Loss} \quad (7)$$

$$\text{Total_Loss} = \sqrt{L} \times (\text{Folding_Unit_Loss} + \beta \times \text{CE_Loss} + 0.5 \times \text{Torsion_Angle_Loss}) \quad (8)$$

4.2 Soft ranking loss for the differentiable Spearman correlation coefficient

The ordinary ranking operation is non-differentiable. To address this issue, we introduce the method proposed by Blondel *et al.*⁵ to achieve a differentiable Spearman correlation coefficient. First, the linear programming formulation for ranking can be derived after casting rankings onto the permutahedron. Second, the strong convex regularization (entropic regularization or quadratic regularization) is introduced into the linear programming formulation of ranking to smoothen discontinuous points of the ranking operators. Third, a parameter ε is introduced to adjust the regularization strength. Fourth, differentiation of ranking can be achieved by computing the Jacobian matrix of isotonic optimization of the projection onto the permutahedron. The authors found that excellent accuracy can be achieved in the ranking experiment without the need to adjust ε . In this study, we adopt the entropic regularization and set ε to 0.01. The detailed mathematical derivation is described as follows.

In discrete ranking operations, $\boldsymbol{\theta} := (\theta_1, \dots, \theta_n)$ is a vector of score, and $\boldsymbol{\rho}$ is reversing permutation for $\boldsymbol{\theta}$. The σ refers to the permutation of $[n]$, and the π is the reverse permutation of $[n]$. The Σ is the $n!$ permutations for $[n]$. For all $\boldsymbol{\theta} \in \mathbb{R}^n$, $\pi \in \Sigma$ and $\boldsymbol{\rho} := (n, n-1, \dots, 1)$,

$$r(\boldsymbol{\theta}) = \operatorname{argmax} \langle \boldsymbol{\theta}, \boldsymbol{\rho}_\pi \rangle. \quad (9)$$

The permutahedron ($\boldsymbol{\omega} \in \mathbb{R}^n$) is introduced to achieve continuous and differentiable ranking operation. The convex hull of permutations of $\boldsymbol{\omega}$ can be represented as:

$$P(\boldsymbol{\omega}) := \operatorname{convex}(\{\boldsymbol{\omega}_\sigma : \sigma \in \Sigma\}) \subset \mathbb{R}^n. \quad (10)$$

When $\boldsymbol{\omega} = \boldsymbol{\rho}$, the linear programming formulations of ranking can be described as below. For all $\boldsymbol{\theta} \in \mathbb{R}^n$, $\boldsymbol{\rho} := (n, n-1, \dots, 1)$ and $\boldsymbol{y} \in P(\boldsymbol{\rho})$,

$$r(\boldsymbol{\theta}) = \operatorname{argmax} \langle \boldsymbol{y}, -\boldsymbol{\theta} \rangle. \quad (11)$$

Setting $(\boldsymbol{z}, \boldsymbol{\omega}) = (-\boldsymbol{\theta}, \boldsymbol{\rho})$. Considering entropic regularization $E(\boldsymbol{\mu}) := \langle \boldsymbol{\mu}, \log \boldsymbol{\mu} - \mathbf{1} \rangle$, Kullback-Leibler (KL) divergence is introduced as

$$\begin{aligned} P_E(\boldsymbol{z}, \boldsymbol{\omega}) &:= \log \operatorname{argmax} \langle \boldsymbol{z}, \boldsymbol{\mu} \rangle - E(\boldsymbol{\mu}) \\ &= \log \operatorname{argmin} KL(\boldsymbol{\mu}, e^{\boldsymbol{z}}) \\ &\quad \boldsymbol{\mu} \in P(e^{\boldsymbol{\omega}}). \end{aligned} \quad (12)$$

The convex regularization is defined as Ψ . The parameter $\varepsilon > 0$ is introduced to control the regularization strength. The Ψ -regularized soft rank can be defined as:

$$r_{\varepsilon\Psi}(\boldsymbol{\theta}) := P_{\varepsilon\Psi}(-\boldsymbol{\theta}, \boldsymbol{\rho}) = P_\Psi(-\boldsymbol{\theta}/\varepsilon, \boldsymbol{\rho}). \quad (13)$$

After reducing the projections to isotonic optimization, the differentiation of projections can be calculated by solving the Jacobian of the projections. For all $z \in \mathbb{R}^n$ and sorted $\omega \in \mathbb{R}^n$:

$$P_{\Psi}(z, \omega) = z - v_{\Psi}(z_{\sigma(z)}, \omega)_{\sigma^{-1}(z)} \quad (14)$$

$$\frac{\partial P_{\Psi}(z, \omega)}{\partial z} = J_{\Psi}(z_{\sigma(z)}, \omega)_{\sigma^{-1}(z)}. \quad (15)$$

5 Algorithm and Pseudocode for SPIRED, SPIRED-Fitness and SPIRED-Stab

5.1 Algorithms for SPIRED

Algorithm 1 describes the overall approach of SPIRED (see Figure 1a). The 1D features are denoted as $\mathbf{s}$, and the 2D features are denoted as $\mathbf{z}$. Each Folding Unit processes and updates $\mathbf{s}$ and $\mathbf{z}$, using $\mathbf{z}$ to predict C_α coordinates (xyz), inter-residue C_α distances (Ca_dist) and pLDDT. The embedding $\mathbf{z}$ from Folding Unit4 is utilized to predict C_β distance distribution as well as the dihedral angles (ω and θ) and scalar angles (φ) between residue pairs. The embedding $\mathbf{s}$ from Folding Unit4 is used to predict torsion angles (ϕ and ψ).

Details of the Folding Unit1 is illustrated in Algorithm 2 (see Figure 1b). The algorithms for Folding Unit2 and Folding Unit3 are similar, except that the input $\mathbf{z}$ for Folding Unit1 is a tensor of zeros, while the input $\mathbf{z}$ for Folding Unit2/3 comes from the previous Folding Unit. Folding Unit4 is more complex than the other ones and its implementation details are shown in Algorithm 3. The TriangularSelfAttentionBlock in these algorithms refers to the corresponding method, *i.e.* Triangular Self-Attention, in the ESMFold⁶.

Algorithm 1 SPIRED

def SPIRED(Sequence, $C_s = 1024$, $C_z = 128$, $L = \text{Length}$):

Process the sequence embedding provided by the ESM-2 3B model

1: SeqEmbed = ESM2(Sequence)

SeqEmbed $\in \mathbb{R}^{37 \times L \times 2560}$

2: SeqEmbed = softmax(LayerWeights) $\odot$ SeqEmbed

SeqEmbed $\in \mathbb{R}^{L \times 2560}$

3: $\mathbf{s}$ = MLP(SeqEmbed)

$\mathbf{s} \in \mathbb{R}^{L \times C_s}$

4: $\mathbf{s} += \text{Embedding}(\text{Onehot}(\text{Sequence}))$

5: $\mathbf{z} = \mathbf{0}^{L \times L \times C_z}$

Predict coordinates of Carbon alpha (C_α)

6: xyz_u1, Ca_dist_u1, pLDDT_u1, $\mathbf{s}$, $\mathbf{z}$ = FoldingUnit1($\mathbf{s}$, $\mathbf{z}$)

xyz_u1 $\in \mathbb{R}^{L \times L \times 3}$

7: xyz_u2, Ca_dist_u2, pLDDT_u2, $\mathbf{s}$, $\mathbf{z}$ = FoldingUnit2($\mathbf{s}$, $\mathbf{z}$)

Ca_dist_u2 $\in \mathbb{R}^{L \times L}$

8: xyz_u3, Ca_dist_u3, pLDDT_u3, $\mathbf{s}$, $\mathbf{z}$ = FoldingUnit3($\mathbf{s}$, $\mathbf{z}$)

pLDDT_u3 $\in \mathbb{R}^{L \times L}$

9: xyz_u4, Ca_dist_u4, pLDDT_u4, $\mathbf{s}$, $\mathbf{z}$ = FoldingUnit4($\mathbf{s}$, $\mathbf{z}$)

Predict backbone dihedrals and Carbon beta (C_β) distogram

10: ω, θ, φ , Cb_distogram = Pred_Cb_Dihedrals($\mathbf{z}$)

Predict torsion angles

11: ψ, ϕ = PredTorsionAngle($\mathbf{s}$)

Algorithm 2 FoldingUnit1

def FoldingUnit1($s \in \mathbb{R}^{L \times C_s}$, $z \in \mathbb{R}^{L \times L \times C_z}$, $N_{block} = 2$, $C_s = 1024$, $C_z = 128$, $L = \text{Length}$):

```
1:  $z = z + \text{PairwiseRelativePositionalEncoding}(L)$ 
2: for all  $i \in \{1, \dots, N_{block}\}$  do
3:    $s, z = \text{TriangularSelfAttentionBlock}(s, z)$ 
4: end for
5:  $z_{res}, z = \text{InstanceRowColumnNorm}(z)$ 
# Predict  $C_\alpha$  coordinates first time
6:  $xyz\_1, z = \text{PredXYZ}(z)$   $xyz\_1 \in \mathbb{R}^{3 \times L \times L}$ 
7:  $Ca\_dist\_1 = \text{CaDistAvg}(xyz)$   $Ca\_dist\_1 \in \mathbb{R}^{L \times L}$ 
8:  $z_{res}, z = \text{InstanceRowColumnNorm}(\text{Concat}(z_{res}, z))$ 
# Predict change of  $C_\alpha$  coordinates
9:  $\Delta xyz, z = \text{PredXYZ}(z, Ca\_dist\_1)$   $\Delta xyz \in \mathbb{R}^{3 \times L \times L}$ 
10:  $z_{res}, z = \text{InstanceRowColumnNorm}(\text{Concat}(z_{res}, z))$ 
# Predict  $C_\alpha$  coordinates second time
11:  $xyz\_2 = xyz\_1 + \Delta xyz$ 
12:  $Ca\_dist\_2 = \text{CaDistAvg}(xyz\_2)$   $Ca\_dist\_2 \in \mathbb{R}^{L \times L}$ 
# Predict pLDDT
13:  $pLDDT = \text{Pred\_pLDDT}(z, Ca\_dist\_2)$ 
14:  $z = \text{ConvBlock}(z, Ca\_dist\_2)$ 
15: return  $(xyz\_1, xyz\_2), (Ca\_dist\_1, Ca\_dist\_2), pLDDT, s, z$ 
```

Algorithm 3 FoldingUnit4

def FoldingUnit4($s \in \mathbb{R}^{L \times C_s}$, $z \in \mathbb{R}^{L \times L \times C_z}$, $N_{block} = 2$, $C_s = 1024$, $C_z = 128$, $L = \text{Length}$):

```
1:  $z = z + \text{PairwiseRelativePositionalEncoding}(L)$ 
2: for all  $i \in \{ 1, \dots, N_{block} \}$  do
3:    $s, z = \text{TriangularSelfAttentionBlock}(s, z)$ 
4: end for
5:  $z_{res}, z = \text{InstanceRowColumnNorm}(z)$ 
# Predict  $C_\alpha$  coordinates first time
6:  $xyz\_1, z = \text{PredXYZ}(z)$   $xyz\_1 \in \mathbb{R}^{3 \times L \times L}$ 
7:  $\text{Ca\_dist\_1} = \text{CaDistAvg}(xyz\_1)$   $\text{Ca\_dist\_1} \in \mathbb{R}^{L \times L}$ 
8:  $z_{res}, z = \text{InstanceRowColumnNorm}( \text{Concat}(z_{res}, z) )$ 
# Predict coordinates of  $C_\alpha$  second time
9:  $\Delta xyz, z = \text{PredXYZ}(z, \text{Ca\_dist\_1})$   $\Delta xyz \in \mathbb{R}^{3 \times L \times L}$ 
10:  $xyz\_2 = xyz\_1 + \Delta xyz$ 
11:  $\text{Ca\_dist\_2} = \text{CaDistAvg}(xyz\_2)$   $\text{Ca\_dist\_2} \in \mathbb{R}^{L \times L}$ 
12:  $z_{res}, z = \text{InstanceRowColumnNorm}( \text{Concat}(z_{res}, z) )$ 
# Predict coordinates of  $C_\alpha$  third time
13:  $\Delta xyz, z = \text{PredXYZ}(z, \text{Ca\_dist\_2})$ 
14:  $xyz\_3 = xyz\_2 + \Delta xyz$ 
15:  $\text{Ca\_dist\_3} = \text{CaDistAvg}(xyz\_3)$ 
16:  $z_{res}, z = \text{InstanceRowColumnNorm}( \text{Concat}(z_{res}, z) )$ 
# Predict coordinates of  $C_\alpha$  fourth time
17:  $\Delta xyz, z = \text{PredXYZ}(z, \text{Ca\_dist\_3})$ 
18:  $xyz\_4 = xyz\_3 + \Delta xyz$ 
19:  $\text{Ca\_dist\_4} = \text{CaDistAvg}(xyz\_4)$ 
20:  $z_{res}, z = \text{InstanceRowColumnNorm}( \text{Concat}(z_{res}, z) )$ 
# Predict coordinates of  $C_\alpha$  fifth time
21:  $\Delta xyz, z = \text{PredXYZ}'(z, \text{Ca\_dist\_4})$ 
22:  $xyz\_5 = xyz\_4 + \Delta xyz$ 
23:  $\text{Ca\_dist\_5} = \text{CaDistAvg}(xyz\_5)$ 
24:  $z_{res}, z = \text{InstanceRowColumnNorm}( \text{Concat}(z_{res}, z) )$ 
# Predict coordinates of  $C_\alpha$  sixth time
25:  $\Delta xyz, z = \text{PredXYZ}'(z, \text{Ca\_dist\_5})$ 
26:  $xyz\_6 = xyz\_5 + \Delta xyz$ 
27:  $\text{Ca\_dist\_6} = \text{CaDistAvg}(xyz\_6)$ 
28:  $z_{res}, z = \text{InstanceRowColumnNorm}( \text{Concat}(z_{res}, z) )$ 
# Predict pLDDT
29:  $\text{pLDDT} = \text{Pred\_pLDDT}(z, \text{Ca\_dist\_6})$ 
30:  $z = \text{ConvBlock}(z, \text{Ca\_dist\_6})$ 
31: return  $(xyz\_1, xyz\_2, xyz\_3, xyz\_4, xyz\_5, xyz\_6), (\text{Ca\_dist\_1}, \text{Ca\_dist\_2}, \text{Ca\_dist\_3}, \text{Ca\_dist\_4}, \text{Ca\_dist\_5},$   

    $\text{Ca\_dist\_6}), \text{pLDDT}, s, z$ 
```

Algorithm 4 InstanceRowColumnNorm

def InstanceRowColumnNorm($z \in \mathbb{R}^{L \times L \times C_z}$, $C_z = 128$, $L = Length$, $\epsilon = 10^{-8}$):

Row normalization

1: mean_row = MEAN(z , dim=1)

2: var_row = Var(z , dim=1)

$var_row \in \mathbb{R}^{L \times 1 \times C_z}$

3: norm_row = $\gamma_{row} \times \frac{z - mean_row}{\sqrt{var_row + \epsilon}} + \beta_{row}$

Column normalization

4: mean_col = Mean(z , dim=0)

5: var_col = Var(z , dim=0)

$var_col \in \mathbb{R}^{1 \times L \times C_z}$

6: norm_col = $\gamma_{col} \times \frac{z - mean_col}{\sqrt{var_col + \epsilon}} + \beta_{col}$

Instance normalization

7: mean_ins = Mean(z , dim=(0,1))

8: var_ins = Var(z , dim=(0,1))

$var_ins \in \mathbb{R}^{1 \times 1 \times C_z}$

9: norm_ins = $\gamma_{ins} \times \frac{z - mean_ins}{\sqrt{var_ins + \epsilon}} + \beta_{ins}$

10: z_{norm} = Concat(norm_ins, norm_row, norm_col, dim=2)

$z_{norm} \in \mathbb{R}^{L \times L \times 384}$

11: z = ELU(Conv2d(z_{norm}))

12: z_{norm} = InstanceNorm2d(z)

13: **return** z , z_{norm}

Algorithm 5 PredXYZ

def PredXYZ($z \in \mathbb{R}^{L \times L \times C_z}$, $Ca_dist \in \mathbb{R}^{L \times L}$, $C_z = 128$, $L = Length$, $\epsilon = 10^{-8}$):

1: cutoff = tensor([8,10,12,14,16,18,20,22,24,26,28,30,32]).reshape(13,1,1)

2: **if** Ca_dist is None **then**

3: $z_{res} = z$

4: **else**

5: Ca_dist = Tile(Ca_dist, [13,1,1])

6: $CaDist_{binary}$ = Float(Ca_dist $\leq$ cutoff)

7: z = Concat(z , $CaDist_{binary}$, dim=0)

$z \in \mathbb{R}^{L \times L \times (C_z + 13)}$

8: z = Conv2d(InstanceNorm2d(z), kernel_size=1)

$z \in \mathbb{R}^{L \times L \times C_z}$

9: $z_{res} = z$

10: **end if**

11: z = Conv2d(Dropout(ELU(InstanceNorm2d(z)), kernel_size=3)

12: z = Conv2d(Dropout(ELU(InstanceNorm2d(z)), kernel_size=3)

13: $z = z + z_{res}$

14: z = Dropout(ELU(InstanceNorm2d(z))

15: xyz = Conv2d(z , kernel_size=1)

$xyz \in \mathbb{R}^{L \times L \times 3}$

16: **return** xyz, z

Algorithm 6 CaDistAvg

```
def CaDistAvg(  $xyz \in \mathbb{R}^{L \times L \times 3}$ ,  $L = \text{Length}$ ,  $\epsilon = 10^{-8}$  ):
    1:  $xyz_{local} = xyz.\text{unsqueeze}(-3) - xyz.\text{unsqueeze}(-2)$   $xyz_{local} \in \mathbb{R}^{L \times L \times L \times 3}$ 
    2:  $\text{Ca\_dist} = \sqrt{\text{SUM}(xyz_{local}^2, \text{dim} = -1) + \epsilon}$   $\text{Ca\_dist} \in \mathbb{R}^{L \times L \times L}$ 
    3:  $\text{Ca\_dist} = \text{MEAN}(\text{Ca\_dist}, \text{dim}=0)$   $\text{Ca\_dist} \in \mathbb{R}^{L \times L}$ 
    4: return Ca_dist
```

Algorithm 7 ConvBlock

```
def ConvBlock(  $z \in \mathbb{R}^{L \times L \times C_z}$ ,  $\text{Ca\_dist} \in \mathbb{R}^{L \times L}$ ,  $C_z = 128$ ,  $L = \text{Length}$  ):
    1: cutoff = tensor([8,10,12,14,16,18,20,22,24,26,28,30,32]).reshape(13,1,1)
    2:  $\text{Ca\_dist} = \text{Tile}(\text{Ca\_dist}.\text{unsqueeze}(0), [13,1,1])$ 
    3:  $\text{CaDist}_{binary} = \text{Float}(\text{Ca\_dist} \leq \text{cutoff})$ 
    4:  $z = \text{Concat}((z, \text{CaDist}_{binary}), \text{dim}=0)$   $z \in \mathbb{R}^{L \times L \times (C_z+13)}$ 
    5:  $z = \text{Conv2d}(\text{InstanceNorm2d}(z), \text{kernel\_size}=1)$   $z \in \mathbb{R}^{L \times L \times C_z}$ 
    6:  $z_{res} = z$ 
    7:  $z = \text{Conv2d}(\text{Dropout}(\text{ELU}(\text{InstanceNorm2d}(z))), \text{kernel\_size}=3)$ 
    8:  $z = \text{Conv2d}(\text{Dropout}(\text{ELU}(\text{InstanceNorm2d}(z))), \text{kernel\_size}=3)$ 
    9:  $z = z + z_{res}$ 
    10:  $z = \text{Dropout}(\text{ELU}(\text{InstanceNorm2d}(z)))$ 
    11: return  $z$ 
```

Algorithm 8 Pred_Cb_Dihedrals

```
def Pred_Cb_Dihedrals(  $z \in \mathbb{R}^{L \times L \times C_z}$ ,  $C_z = 128$ ,  $L = \text{Length}$ ):
    1:  $z_{symm} = (z + z.\text{permute}(1,0)) / 2$ 
    2:  $\text{Cb\_distogram} = \text{Conv2d}(z_{symm}, \text{kernel\_size}=1)$   $\text{Cb\_distogram} \in \mathbb{R}^{L \times L \times 48}$ 
    3:  $\omega = \text{Conv2d}(z_{symm}, \text{kernel\_size}=1)$   $\omega \in \mathbb{R}^{L \times L \times 25}$ 
    4:  $\theta = \text{Conv2d}(z, \text{kernel\_size}=1)$   $\theta \in \mathbb{R}^{L \times L \times 25}$ 
    5:  $\varphi = \text{phiConv}(z, \text{kernel\_size}=1)$   $\varphi \in \mathbb{R}^{L \times L \times 13}$ 
    6: return ca_dist, cb_distogram,  $\omega$ ,  $\theta$ ,  $\varphi$ 
```

Algorithm 9 Pred_TorsionAngle

```
def Pred_TorsionAngle(  $s \in \mathbb{R}^{L \times C_s}$ ,  $C_s = 1024$ ,  $L = \text{Length}$ ):
    1:  $\text{Conv}_1 = \text{Conv2d}(1, 1, \text{kernel\_size}=15, \text{stride}=1, \text{padding}=7)$ 
    2:  $\text{Conv}_2 = \text{Conv2d}(1, 2, \text{kernel\_size}=(C_s, 2), \text{stride}=1, \text{padding}=0)$ 
    3:  $s = \text{ELU}(\text{Conv}_1(s))$   $s \in \mathbb{R}^{L \times C_s}$ 
    #  $\psi$  in the psi_phi[0,:],  $\phi$  in the psi_phi[1,:]
    4:  $\text{psi\_phi} = \text{Conv}_2(s)$   $\text{psi\_phi} \in \mathbb{R}^{2 \times (L-1)}$ 
    5: return psi_phi
```

Algorithm 10 Pred_pLDDT

def Pred_pLDDT($z \in \mathbb{R}^{L \times L \times C_z}$, $\text{Ca_dist} \in \mathbb{R}^{L \times L}$, $C_z = 128$, $L = \text{Length}$):

```
1: cutoff = tensor([8,10,12,14,16,18,20,22,24,26,28,30,32]).reshape(13,1,1)
2: Ca_dist = Tile( Ca_dist.unsqueeze(0), [13,1,1] )
3: Ca_dist_binary = Float( Ca_dist ≤ cutoff )
4: z = Concat( (z, Ca_dist_binary, dim=-1 )
5: z = Conv2d( InstanceNorm2d( z ), kernel_size=1)
6:  $z_{res} = z$ 
7: z = Conv2d( Dropout( ELU( InstanceNorm2d( z ) ), kernel_size=3 )
8: z = Conv2d( Dropout( ELU( InstanceNorm2d( z ) ), kernel_size=3 )
9:  $z = z + z_{res}$ 
10: z = Dropout( ELU( InstanceNorm2d( z ) )
11: pLDDT = Conv2d(z, kernel_size=1)
12: return pLDDT
```

$\text{Ca_dist_binary} \in \mathbb{R}^{L \times L \times 13}$
 $z \in \mathbb{R}^{L \times L \times (C_z + 13)}$
 $z \in \mathbb{R}^{L \times L \times C_z}$

$\text{pLDDT} \in \mathbb{R}^{L \times L}$

5.2 Algorithms for the loss functions of SPIRED training

Algorithm 11 compute_RDloss_1

def compute_RDloss_1(pred_xyz $\in \mathbb{R}^{L \times L \times 3}$, true_xyz $\in \mathbb{R}^{L \times L \times 3}$, Ca_label $\in \mathbb{R}^{L \times L}$, cutoff = {24, 32}, L = Length, $\epsilon = 10^{-8}$):

For Folding Unit1/2/3, cutoff = 24. For the first 4 structure predictions of Folding Unit4, cutoff = 32.

```

1: diff_xyz = pred_xyz - true_xyz                                diff_xyz  $\in \mathbb{R}^{L \times L \times 3}$ 
2: diff_xyz_local = diff_xyz.unsqueeze(-3) - diff_xyz.unsqueeze(-2)  diff_xyz_local  $\in \mathbb{R}^{L \times L \times L \times 3}$ 
3: RD =  $\sqrt{\text{SUM}(\text{pred\_xyz}_{local}^2, dim = -1) + \epsilon}$                 RD  $\in \mathbb{R}^{L \times L \times L}$ 
4: mask = Sigmoid( cutoff - Ca_label)                             mask  $\in \mathbb{R}^{L \times L}$ 
5: mask = Tile( mask.unsqueeze(0), [L,1,1])                       mask  $\in \mathbb{R}^{L \times L \times L}$ 
6: RD_loss =  $\frac{\text{SUM}(RD \times \text{mask})}{\text{SUM}(\text{mask}) + \epsilon}$ 
7: return RD_loss

```

Algorithm 12 compute_RDloss_2

def compute_RDloss_2(pred_xyz $\in \mathbb{R}^{L \times L \times 3}$, true_xyz $\in \mathbb{R}^{L \times L \times 3}$, cutoff = 40, ratio = 0.1, L = Length, $\epsilon = 10^{-8}$):

Only for the last 2 structure predictions of Folding Unit4.

```

1: diff_xyz = pred_xyz - true_xyz                                diff_xyz  $\in \mathbb{R}^{L \times L \times 3}$ 
2: diff_xyz_local = diff_xyz.unsqueeze(-3) - diff_xyz.unsqueeze(-2)  diff_xyz_local  $\in \mathbb{R}^{L \times L \times L \times 3}$ 
3: RD =  $\sqrt{\text{SUM}(\text{pred\_xyz}_{local}^2, dim = -1) + \epsilon}$                 RD  $\in \mathbb{R}^{L \times L \times L}$ 
4: mask_RD_clamp = Float(RD <= cutoff)
5: mask_prob = Float( torch.rand( [L,L] ) >= (1 - ratio) )
6: mask_prob = torch.triu(mask_prob, diagonal=1)
7: mask_prob = mask_prob + mask_prob.permute(1,0)
8: mask = mask_prob  $\times$  ( 1 - mask_RD_clamp ) + mask_RD_clamp
9: RD_loss =  $\frac{\text{SUM}(RD \times \text{mask})}{\text{SUM}(\text{mask}) + \epsilon}$ 
10: return RD_loss

```

Algorithm 13 compute_CaDist_Loss

def compute_CaDist_Loss(Ca_dist_avg $\in \mathbb{R}^{L \times L}$, label_Ca_dist $\in \mathbb{R}^{L \times L}$, cutoff = {24, 32, 40}, ratio = 0.1, L = Length, RD_Loss_type = {1, 2}, $\epsilon = 10^{-8}$):

1: Ca_diff_diff = $\|label_Ca_dist - Ca_dist_avg\|$

When it is with compute_RDloss_1, RD_Loss_type = 1. If it is with compute_RDloss_2, RD_Loss_type = 2.

2: **if** RD_Loss_type = 1 **then**

3: mask = Sigmoid(cutoff - Ca_dist_avg) mask $\in \mathbb{R}^{L \times L}$

4: **else**

5: mask = Float(Ca_diff_diff <= cutoff) mask $\in \mathbb{R}^{L \times L}$

6: **end if**

7: Ca_Dist_L1_Loss = $\frac{\text{SUM}(Ca_Dist_L1_loss \times \text{mask})}{\text{SUM}(\text{mask}) + \epsilon}$

8: **return** Ca_Dist_L1_loss

Algorithm 14 compute_pLDDT_Loss

def compute_pLDDT_Loss(pLDDT $\in \mathbb{R}^{L \times L}$, true_IDDT $\in \mathbb{R}^{L \times L}$):

1: pLDDT_Loss = MEAN($\| pLDDT - \text{true_IDDT} \|$)

2: **return** pLDDT_Loss

Algorithm 15 compute_Violation_Loss

def compute_Violation_Loss(pred_Ca_dist $\in \mathbb{R}^{L \times L \times L}$, tau = 0.02, L = Length, eps = 10^{-8}):

1: res_idx = torch.arange(L).reshape(1, L)

2: masklocal = Float($\| \text{res_idx} - \text{res_idx.T} \| \geq 2$)

3: keeplocal = 1 - masklocal - torch.eye(L)

4: masklocal = Tile(masklocal[None, :, :], (L, 1, 1)) masklocal $\in \mathbb{R}^{L \times L \times L}$

5: keeplocal = Tile(keeplocal[None, :, :], (L, 1, 1)) keeplocal $\in \mathbb{R}^{L \times L \times L}$

6: Nclash = SUM(Float((3.80 - tau - pred_Ca_dist $\times$ masklocal) > 0)) + eps

7: Clash_Loss = $\frac{\text{SUM}(\text{MAX}(3.80 - \text{tau} - \text{pred_Ca_dist} \times \text{masklocal}, 0))}{\text{Nclash}}$

8: Bond_Loss = MEAN(MAX($\| \text{pred_Ca_dist} \times \text{keeplocal} - 3.82 \| - \text{tau}, 0$))

9: **return** Clash_Loss, Bond_Loss

5.3 Algorithms for the fitness and $\Delta\Delta G/\Delta T_m$ prediction modules in SPIRED-Fitness and SPIRED-Stab

Algorithm 16 Fitness_Module

```

def Fitness_Module( f1d_esm2_650M  $\in \mathbb{R}^{L \times 1280}$ , xyz  $\in \mathbb{R}^{L \times L \times 3}$ , pLDDT  $\in \mathbb{R}^{L \times L}$ , f1d_esm1v_logits  $\in \mathbb{R}^{5 \times L \times 20}$ ,
L = Length ):
    1: s, z = Fitness_Encoder(f1d_esm2_650M, xyz, pLDDT)  $s \in \mathbb{R}^{L \times 32}, z \in \mathbb{R}^{L \times L \times 32}$ 
    # Output logits of single-mutation fitness landscape
    2: s = Linear( s )  $s \in \mathbb{R}^{L \times 20}$ 
    3: coef_1 = Parameter([0.5, 0.1, 0.1, 0.1, 0.1, 0.1]).reshape(6,1,1)
    4: s = coef_1  $\times$  Concat(s.unsqueeze(0), f1d_esm1v_logits)  $s \in \mathbb{R}^{6 \times L \times 20}$ 
    5: single_logits = Mean(s, dim=0) single_logits  $\in \mathbb{R}^{L \times 20}$ 
    # Output logits of double-mutation fitness landscape
    6: z = LeakyReLU(Conv2d(InstanceNorm2d( z ), kernel_size=1))  $z \in \mathbb{R}^{L \times L \times 200}$ 
    7: z = LeakyReLU(Conv2d(InstanceNorm2d( z ), kernel_size=1))  $z \in \mathbb{R}^{L \times L \times 400}$ 
    8: coef_2 = Parameter([0.5, 0.1, 0.1, 0.1, 0.1, 0.1]).reshape(6,1,1)
    9: s_1v = f1d_esm1v_logits
    10: f2d_esm1v_logits = (s_1v[:, None, :, None] - s_1v[None, :, None, :]).reshape(5, L, L, 400)
    11: z = coef_2  $\times$  Concat(z.unsqueeze(0), f2d_esm1v_logits)  $z \in \mathbb{R}^{6 \times L \times L \times 400}$ 
    12: double_logits = Mean(z, dim=0) double_logits  $\in \mathbb{R}^{L \times L \times 400}$ 
    13: return single_logits, double_logits

```

Algorithm 17 Fitness_Encoder

```

def Fitness_Encoder( f1d_esm2_650M  $\in \mathbb{R}^{L \times 1280}$ , xyz  $\in \mathbb{R}^{L \times L \times 3}$ , pLDDT  $\in \mathbb{R}^{L \times L}$ ,  $N_{block} = 2$ , L = Length ):
    # Process 1D feature from ESM-2(650M) model
    1: s = LeakyReLU( Linear( LayerNorm(f1d_esm2_650M) ) )  $s \in \mathbb{R}^{L \times 640}$ 
    2: s = LeakyReLU( Linear( s ) )  $s \in \mathbb{R}^{L \times 320}$ 
    3: s = LeakyReLU( Linear( s ) )  $s \in \mathbb{R}^{L \times 32}$ 
    4: s = Linear( s )  $s \in \mathbb{R}^{L \times 32}$ 
    # Process 2D feature of predicted coordinates from SPIRED
    5: z = Linear( xyz )  $z \in \mathbb{R}^{L \times L \times 32}$ 
    6: for all i  $\in \{ 1, \dots, N_{block} \}$  do
    7:     node_feature, edge_feature = GAT_Network(s, z, pLDDT)
    8: end for
    9: edge_feature = LeakyReLU( Linear( LayerNorm(edge_feature) ) ) edge_feature  $\in \mathbb{R}^{L \times L \times 64}$ 
    10: edge_feature = LeakyReLU( Linear( edge_feature ) ) edge_feature  $\in \mathbb{R}^{L \times L \times 32}$ 
    11: return node_feature, edge_feature

```

Algorithm 18 GAT_Network

```
def Pretrain_GAT(  $s \in \mathbb{R}^{L \times 32}$  ,  $z \in \mathbb{R}^{L \times L \times 32}$  ,  $pLDDT \in \mathbb{R}^{L \times L \times 1}$  ,  $N_{head} = 8$ , node_dim = 32, head_dim = node_dim /  $N_{head} = 4$ , pair_dim = 32, L = Length ):
    1:  $s_{res} = s$ 
    # Node to  $\alpha$ 
    2:  $q, k = \text{LinearNoBias}(s)$   $q, k \in \mathbb{R}^{N_{head} \times L \times 4}$ 
    3:  $a_s = \frac{1}{\sqrt{head\_dim}} q^\top k$ 
    # Pair to  $\alpha$ 
    4:  $a_z = \text{LinearNoBias}(z)$   $a_z \in \mathbb{R}^{L \times L \times N_{head}}$ 
    # Convolution for  $\alpha$  and pLDDT mask
    5:  $a = \text{Concat}([a_s, a_z], \text{dim}=-1)$ 
    6:  $a = \text{LeakyReLU}(\text{Conv2d}(\text{InstanceNorm2d}(a)))$   $a \in \mathbb{R}^{L \times L \times N_{head}}$ 
    7:  $a = \text{LeakyReLU}(\text{Conv2d}(\text{InstanceNorm2d}(\text{Concat}(a, pLDDT))))$   $a \in \mathbb{R}^{L \times L \times N_{head}}$ 
    # Node Aggregation
    8:  $v = \text{LinearNoBias}(s)$ 
    9:  $s = a \times v$ 
    # Pair Aggregation
    10:  $s_z = a.\text{unsqueeze}(-1) \times z.\text{unsqueeze}(-2)$   $s_z \in \mathbb{R}^{L \times L \times N_{head} \times pair\_dim}$ 
    11:  $s_z = \text{SUM}(s_z, \text{dim}=1).\text{reshape}(L, 128)$   $s_z \in \mathbb{R}^{L \times 128}$ 
    # Out transform for node embedding
    12:  $s = \text{Concat}([s_z, s], \text{dim}=-1)$ 
    13:  $s = \text{LayerNorm}(\text{Linear}(\text{LayerNorm}(s)))$   $s \in \mathbb{R}^{L \times 64}$ 
    14:  $s = \text{Linear}(\text{LeakyReLU}(s))$   $s \in \mathbb{R}^{L \times 32}$ 
    15:  $s = \text{LayerNorm}(s_{res} + s)$ 
    #  $\alpha$  to pair
    16:  $z = \text{InstanceNorm2d}(\text{Concat}([z, a], \text{dim}=-1))$ 
    17:  $z = \text{LeakyReLU}(\text{Conv2d}(z, \text{kernel\_size}=3, \text{stride}=1))$ 
    18: return  $s, z$   $s \in \mathbb{R}^{L \times 32}, z \in \mathbb{R}^{L \times L \times 32}$ 
```

Algorithm 19 Stability_Pred_Module

```
def Stability_Pred_Module( wild_seq_embed  $\in \mathbb{R}^{L \times 1280}$  , mut_seq_embed  $\in \mathbb{R}^{L \times 1280}$  , wild_xyz  $\in \mathbb{R}^{L \times L \times 3}$  , mut_xyz  $\in \mathbb{R}^{L \times L \times 3}$  , wild_pLDDT  $\in \mathbb{R}^{L \times L}$  , mut_pLDDT  $\in \mathbb{R}^{L \times L}$  , L = Length ):
    # Predict  $\Delta G$  and  $\Delta T_m$  of wild-type protein
    1:  $\Delta G_{wt}, T_m_{wt} = \text{Stability\_Encoder}(wild\_seq\_embed, wild\_xyz, wild\_pLDDT, mut\_pos)$ 
    # Predict  $\Delta G$  and  $\Delta T_m$  of mutant protein
    2:  $\Delta G_{mut}, T_m_{mut} = \text{Stability\_Encoder}(mut\_seq\_embed, mut\_xyz, mut\_pLDDT, mut\_pos)$ 
    # Calculate  $\Delta \Delta G$ ,  $\Delta \Delta G\_coef$  is a learnable parameter
    3:  $\Delta \Delta G\_coef = \text{Parameter}([1.0])$ 
    4:  $\Delta \Delta G = (\Delta G_{mut} - \Delta G_{wt}) \times \Delta \Delta G\_coef$   $\Delta \Delta G \in \mathbb{R}$ 
    # Calculate  $\Delta T_m$ ,  $\Delta T_m\_coef$  is a learnable parameter
    5:  $\Delta T_m\_coef = \text{Parameter}([1.0])$ 
    6:  $\Delta T_m = (T_m_{mut} - T_m_{wt}) \times \Delta T_m\_coef$   $\Delta T_m \in \mathbb{R}$ 
    7: return  $\Delta \Delta G, \Delta T_m$ 
```

The mut_pos is a tensor of length L , used to represent the position of the mutated residue. It has a value of 1 at the index corresponding to the mutation position, and 0 at all other positions.

Algorithm 20 Stability_Encoder

```
def Stability_Encoder( seq_embed  $\in \mathbb{R}^{L \times 1280}$  , xyz  $\in \mathbb{R}^{L \times L \times 3}$  , pLDDT  $\in \mathbb{R}^{L \times L}$  , mut_pos  $\in \mathbb{R}^L$  ,  $N_{block} = 3$  , L = Length ):

# Process wild/mutant proteins' sequence embedding from ESM-2(650M)
1: node_feat = LeakyReLU( Linear( LayerNorm(seq_embed)))           node_feat  $\in \mathbb{R}^{L \times 640}$ 
2: node_feat = LeakyReLU( Linear( node_feat))                     node_feat  $\in \mathbb{R}^{L \times 320}$ 
3: node_feat = Linear( LeakyReLU( Linear( node_feat)))            node_feat  $\in \mathbb{R}^{L \times 32}$ 

# Process coordinates information from SPIRED
4: edge_feat = Linear( xyz )                                       edge_feat  $\in \mathbb{R}^{L \times 32}$ 

# GAT Network process information from sequence, structure and pLDDT
5: for all i  $\in \{ 1, ..., N_{block} \}$  do
6:   node_feat, edge_feat = GAT_Network(node_feat, edge_feat, pLDDT)
7: end for

8: node_feat = SUM(node_feat  $\times$  mut_pos)                         node_feat  $\in \mathbb{R}^{1 \times 32}$ 

# MLP Module for  $\Delta G$  value prediction
9:  $\Delta G\_feat$  = Linear( LayerNorm(node_feat) )                   $\Delta G\_feat \in \mathbb{R}^{1 \times 16}$ 
10:  $\Delta G$  = Linear( LeakyReLU( $\Delta G\_feat$ ) )                       $\Delta G \in \mathbb{R}$ 

# MLP Module for  $T_m$  value prediction
11:  $T_m\_feat$  = Linear( LayerNorm(node_feat) )                      $T_m\_feat \in \mathbb{R}^{1 \times 16}$ 
12:  $T_m$  = Linear( LeakyReLU(  $T_m\_feat$  ) )                        $T_m \in \mathbb{R}$ 
13: return  $\Delta G, T_m$ 
```

References

1. Mi, T. & Gong, H., GDFold2: a fast and parallelizable protein folding environment with freely defined objective functions. Preprint at <http://dx.doi.org/10.1101/2024.03.13.584741>, 2024.
2. Chen, L., Zhang, Z., Li, Z., *et al.*, Learning protein fitness landscapes with deep mutational scanning data from multiple sources. *Cell Systems* **14**, 706–721.e5, 2023.
3. Xu, Y., Liu, D. & Gong, H., Improving the prediction of protein stability changes upon mutations by geometric learning and a pre-training strategy. Preprint at <https://doi.org/10.1101/2023.05.28.542668>, 2023.
4. Pancotti, C., Benevenuta, S., Birolo, G., *et al.*, Predicting protein stability changes upon single-point mutation: a thorough comparison of the available tools on a new dataset. *Briefings in Bioinformatics* **23**, 2022.
5. Blondel, M., Teboul, O., Berthet, Q., *et al.*, Fast Differentiable Sorting and Ranking. *Proceedings of the 37th International Conference on Machine Learning* 950–959, 2020.
6. Lin, Z., Akin, H., Rao, R., *et al.*, Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130, 2023.