


## RESEARCH ARTICLE

# Predicting the effects of mutations on protein solubility using graph convolution network and protein language model representation

Jing Wang<sup>1,2</sup>  | Sheng Chen<sup>2</sup> | Qianmu Yuan<sup>2</sup> | Jianwen Chen<sup>2</sup> |  
Danping Li<sup>3</sup> | Lei Wang<sup>4</sup> | Yuedong Yang<sup>2</sup>

<sup>1</sup>Guangzhou institute of technology, Xidian University, Guangzhou, China

<sup>2</sup>School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China

<sup>3</sup>School of Telecommunications Engineering, Xidian University, Xi'an, China

<sup>4</sup>School of Electronic Engineering, Xidian University, Xi'an, China

## Correspondence

Yuedong Yang, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China.

Email: yangyd25@mail.sysu.edu.cn

Lei Wang, School of Electronic Engineering, Xidian University, Xi'an, China.

Email: leiwang@mail.xidian.edu.cn

## Funding information

National Key Research and Development Program of China, Grant/Award Number: 2022YFF1203100; National Natural Science Foundation of China, Grant/Award Number: 12126610; Guangdong Key Field R&D Plan, Grant/Award Numbers: 2018B010109006, 2019B020228001; Guangzhou S&T Research Plan, Grant/Award Number: 202007030010

## Abstract

Solubility is one of the most important properties of protein. Protein solubility can be greatly changed by single amino acid mutations and the reduced protein solubility could lead to diseases. Since experimental methods to determine solubility are time-consuming and expensive, in-silico methods have been developed to predict the protein solubility changes caused by mutations mostly through protein evolution information. However, these methods are slow since it takes long time to obtain evolution information through multiple sequence alignment. In addition, these methods are of low performance because they do not fully utilize protein 3D structures due to a lack of experimental structures for most proteins. Here, we proposed a sequence-based method DeepMutSol to predict solubility change from residual mutations based on the Graph Convolutional Neural Network (GCN), where the protein graph was initiated according to predicted protein structure from AlphaFold2, and the nodes (residues) were represented by protein language embeddings. To circumvent the small data of solubility changes, we further pretrained the model over absolute protein solubility. DeepMutSol was shown to outperform state-of-the-art methods in benchmark tests. In addition, we applied the method to clinically relevant genes from the ClinVar database and the predicted solubility changes were shown able to separate pathogenic mutations. All of the data sets and the source code are available at <https://github.com/biomed-AI/DeepMutSol>.

## KEYWORDS

graph convolutional neural network, protein language models, protein mutation, protein pretraining, solubility changes

## 1 | INTRODUCTION

Proteins are attractively diagnostic and therapeutic molecules because of their versatility and specificity of functions and their inherent low toxicity,<sup>1–4</sup> and have been widely used in the treatment of various

diseases including cancer and autoimmune diseases.<sup>5</sup> However, Changes in protein solubility are likely to affect the biological functions of proteins.<sup>6</sup> For example, reducing the solubility of protein drugs may make them ineffective or even toxic.<sup>7,8</sup> In addition, protein solubility is also an essential property for the protein's extraction, separation, and purification.<sup>9,10</sup> Therefore, maintaining or increasing the protein solubility is important for diagnostic and therapeutic applications. Experimentally, solubility is widely optimized through protein engineering

Jing Wang and Sheng Chen contributed equally to this work and should be considered co-first authors.

technique such as protein mutagenesis.<sup>11,12</sup> Though a single amino acid mutation might profoundly alter protein solubility,<sup>13–15</sup> the formed large libraries of candidate mutations are time-consuming and expensive for screening. Therefore, *in silico* methods are required to screen soluble candidates.<sup>16–18</sup>

Until now, several methods have been developed to predict the effects of mutation on protein solubility. OptSolMut<sup>19</sup> leveraged linear programming algorithm to optimize a scoring function. CamSol<sup>20</sup> utilized the native protein structure to build a residue specific solubility profile. SolubiS<sup>21</sup> is also a structure-based method that combined interaction analysis of FoldX,<sup>22</sup> aggregation prediction of TANGO,<sup>23</sup> and structural analysis of YASARA.<sup>24</sup> SODA<sup>25</sup> is a sequence-based method whereas available structure can also be used to mask the buried residues from prediction. Machine learning algorithm like random forest has been introduced by Pon-Sol.<sup>26</sup> Pon-Sol2<sup>27</sup> extended PonSol's data set and leveraged gradient lifting algorithm for sequence-based prediction. However, deep learning methods have not been utilized for this task due to the small experimental data. In addition, feature extraction methods based on amino acid composition, physicochemical properties, structural domains, evolutionary conservatism, and other aspects need to consider multiple features at the same time, which will lead to high feature dimensions, prone to overfitting and other problems, and there may be a correlation between different features, which will lead to repeated information and noise information. The extraction of protein evolutionary features requires comparison of the differences of multiple protein sequences and calculation of evolutionary trees, which requires a large amount of data analysis and computing resources, so the time cost is relatively high. Moreover, when multiple sequences are compared, there are problems such as sequence insertion and deletion, as well as confusion of homologous sequences, which will lead to the accumulation of errors in the comparison results.

Recently, there is great progress in small data learning. For example, transfer learning technique has shown ability to transfer knowledge from related tasks with large-scale data to the target task with limited data.<sup>28</sup> Our previous works have highlighted the power of pre-trained protein language model ProtTrans<sup>29</sup> on metal ion-binding site prediction.<sup>30</sup>

Another limitation of existing sequence-based solubility change prediction is the lack of protein structure information. Deep learning methods have been shown to benefit from protein structure information for protein solubility prediction,<sup>31</sup> stability changes prediction,<sup>32</sup> protein design,<sup>33</sup> protein–protein interaction,<sup>34</sup> and drug discovery.<sup>35</sup> However, the number of available protein crystal structures for solubility mutagenesis is far more insufficient for deep learning model training. Our previous work leveraged AlphaFold<sup>36</sup> as replacement of crystal structures in protein–DNA interaction task.

Here, we proposed a sequence-based method DeepMutSol to predict solubility change from residual mutations based on the Graph Convolutional Neural Network (GCN), where protein graph was initiated according to predicted protein structure from Alphafold2, and the nodes (residues) were represented by protein language embeddings. To circumvent the small data of solubility changes, we further

pretrained the model over absolute protein solubility. To our best knowledge, this is the first deep-learning-based method for this task. The benchmark test showed that our method outperformed all other compared methods over a variety of metrics. Further downstream analysis showed that the predictions could discriminate pathogenic mutations in the Clinvar data set.<sup>37,38</sup> It highlighted the potential of DeepMutSol for further application on disease pathway analysis and drug discovery.

To summarize, our contributions are as follows:

1. We developed the first deep-learning-based method for mutation-caused protein solubility change prediction which outperformed existing methods.
2. We leveraged AlphaFold to predict protein structure and introduced the Graph Convolutional Neural Network (GCN) to effectively encode the protein structure information.
3. We employed transfer-learning techniques to overcome the problem of training data limitation through using the embedded representations of a pretrained ProtTrans protein language model and a pretrained model over absolute protein solubility.

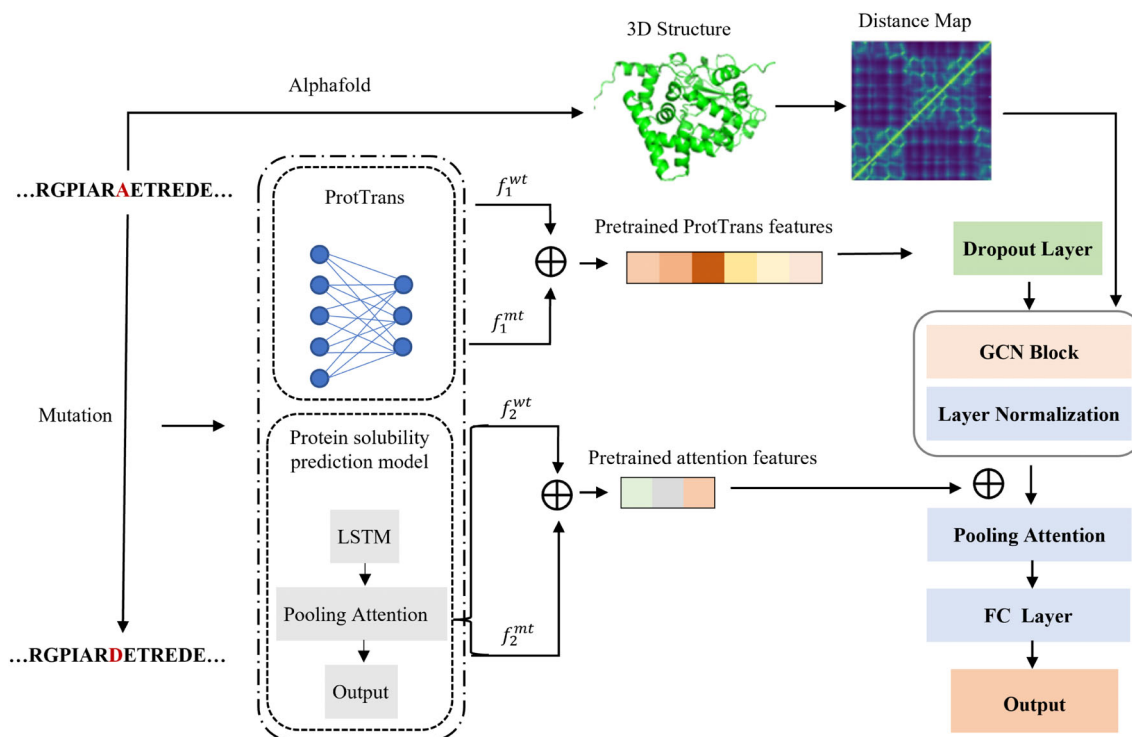
## 2 | METHODS

### 2.1 | Data sets

We used the same data set from the Pon-Sol2 study<sup>27</sup> to train our model. To be complete, we briefly introduced the data set. The data set includes 6328 mutations from 77 proteins. These mutations are divided into three categories: solubility decreasing, increasing, or non-effect. Among these mutations, there were 3136 mutations that reduced solubility, 2166 mutations that had no effect on solubility, and 1026 mutations that increased solubility, respectively. The data set was randomly split into the training and test sets with similar ratio for three categories of mutations. It should be noted that the test data set does not share mutations at the same position with training data set. Eventually, 5666 mutations were used for training, including 2798 mutations with reduced solubility, 1929 mutations with no effect on solubility, and 939 mutations with increased solubility. The independent test set consisted of 662 mutations, including 338 with reduced solubility, 237 with constant solubility, and 87 with increased solubility. In order to overcome the problem of data set imbalance, we introduce reverse mutation in the training set for data augmentation, which has proved to be useful in our previous work.<sup>32</sup> The details of the final dataset are provided in Data S1.

### 2.2 | Network architecture

As shown in Figure 1, We used AlphaFold to predict three-dimensional structure of the wild-type protein to obtain spatial structure information, and used the pretrained models to produce protein sequence features. Since the experimental data of protein mutation



**FIGURE 1** The overall framework of the DeepMutSol model. The DeepMutSol model consists of GCN module, self-attention pooling module, and fully connected module. The predicted structures of the wild-type sequence by AlphaFold were used to construct the protein graph with residual distances as edge attributes. In parallel, we used two pretrained models to extract ProtTrans features and attentional pooling features before and after mutations, which are used to generate node features. The pretrained attention pooling features are finally concatenated with the output features of GCN module, and input into the fully connected layer to predict the solubility change.

solubility change is small, we used the transferred learning technique to extract the learned feature information from the pretrained related tasks with large-scale data for the protein mutation solubility change prediction task. We used two models for the transfer learning: ProtTrans and the protein solubility value prediction model. We input the protein sequence features extracted from ProtTrans and the adjacency matrix generated according to the predicted three-dimensional structure into the graph convolutional network (GCN) to aggregate the spatial structure and sequence information. We concatenate the output features of the GCN module with the pooling attention features extracted from the protein solubility value prediction model. The concatenated features are converted into global embedded features by self-attention pooling module, and then input into the fully connected layer to obtain the final predicted solubility change value.

### 2.2.1 | ProtTrans

The ProtTrans model is a self-supervised language model trained on 80 billion amino acids from 200 million protein sequences (UniRef100) and 393 billion amino acids from 2.1 billion protein sequences from the Big Fat Database (BFD). BFD is the largest set of protein sequences available today (22 and 112 times the size of the entire English Wikipedia, respectively). The ProtTrans model has been shown to surpass the state of the art in delivering the most informative embeddings and validates the advantages of using embeddings

for downstream prediction tasks. We use the ProtTrans model to extract effective protein sequence information to solve the problem of the limited data set. At the same time, using ProtTrans model to extract pretrained features only needs to input protein sequences, instead of using MSAs, which can save a lot of time and cost. For a protein sequence of length  $L$ , The pretrained ProtTrans model can generate  $L \times 1024$ -dimension features. We spliced ProtTrans features extracted from wild-type protein sequences and mutant protein sequences to obtain  $L \times 2048$ -dimension features as node features, and input the node features into the GCN module to obtain  $L \times 16$ -dimension feature vector.

### 2.2.2 | Protein solubility prediction model

Compared with the experimental data of protein mutation solubility change, the experimental data of protein solubility are much bigger. Considering the potential for knowledge transfer from protein solubility prediction tasks with larger data sets, we also pretrained a protein solubility prediction model for extracting solubility-related features. Our solubility prediction task was pretrained on the data set from DeepSol.<sup>39</sup> We removed redundant sequences and sequences homologous to the test set from the training data set in the same way as DeepSol. At the same time, we also remove all sequences from the training set that had a sequence identity  $\geq 30\%$  to any sequence in the test set of our protein mutation solubility change prediction task.

Finally, the training set of our solubility prediction task consisted of 28,972 soluble and 40,448 insoluble protein sequences, and the test set consisted of 1000 soluble and 1001 insoluble protein sequences. The model consists of long-short term memory (LSTM) network and self-attention pooling modules with input features extracted from the ProtTrans model as pretrained features. We extract the pooling attention features of the attention pooling module and transfer them to the DeepMutSol model for training. A single protein sequence can extract  $L \times 4$ -dimension attentional pooling features, and we also spliced the attentional pooling features of wild-type and mutant-type protein sequences to obtain  $L \times 8$ -dimension features. We spliced the  $L \times 8$ -dimensional attention pooling features with  $L \times 16$ -dimensional GCN output feature vector as the input of the self-attention pooling module.

### 2.2.3 | Adjacency matrix

The GCN adjacency matrix was represented through the two-dimensional distance map  $S$  according to the predicted protein tertiary structure:

$$s_{ij} = \frac{2}{1 + \frac{\max(d_0, d_{ij})}{d_0}}, \quad (1)$$

where  $d_{ij}$  is the spatial distance between  $c_\alpha$  atoms of residue  $i$  and residue  $j$ , and  $d_0$  is set as 4.0 Å, as also used in definition of the SP-score.<sup>40</sup> This distance conversion ensures a score from 0 to 1, indicating the possibilities to form contacts between the residue pairs. We only use the pre-mutation structure of the protein, so that we do not need to predict the respective structure for each mutation.

### 2.2.4 | Node selection

We did not employ all residues in a protein for the node representation of GCN. In contrast, we selected 30 residues with largest absolute change between the ProTrans features of wild-type sequence and mutated sequence. These residues are selected as they are the most "mutation sensitive." Selecting these residues instead of all residues is considered to be more computationally efficient and less information-redundant. We also calculated the 3D distance between the 30 selected residues and the mutated residues in the wild-type AlphaFold structure of test set proteins. The average 3D distance of the selected residue to the mutated residue (13.87 Å) is significantly smaller than that of the other residues (27.79 Å). The selection of these residues should be helpful to introduce some useful structural information.

### 2.2.5 | Graph convolution network

Currently The fully connected attribute graph formed by the selected  $L$  residues as nodes can be represented by  $G = (A, X)$ , where  $X \in R^{L \times f}$  represents the node feature matrix and  $f$  represents the dimension of

node features,  $A \in R^{L \times L}$  represents the contact matrix. Our graph convolution network takes the following formula:

$$G^{(l+1)} = \sigma(\tilde{D}^{-1} \tilde{A} G^{(l)} W^{(l)}) \quad (2)$$

where  $\tilde{A} = A + I$  is the adjacency matrix obtained by adding contact matrix  $A$  determined by the predicted distance map and the identity matrix  $I$  for self-loops,  $\tilde{D} = R^{L \times L}$  is the degree matrix with  $\tilde{D}_{ii} = \sum \tilde{A}_{ik}$  that can normalize  $\tilde{A}$  to sum up to 1.0 in each row.  $W^{(l)} \in R^{f \times f'}$  is a weight matrix of layer-specific trainable parameters that can map features to a low-dimensional space of size  $f'$ .  $\sigma(\cdot)$  represents a nonlinear activation function, and here we use LeakyReLU( $\cdot$ ). The more layers of GCN, the more and deeper node and edge feature information can be aggregated. However, too many GCN layers may cause the gradient to disappear, resulting in the decline of prediction accuracy. So maintaining a balance between layers and algorithm complexity is critical. We set four values (1–4) for the number of GCN layers, and after tuning the validation set, we found that 2 GCN layers were optimal. We selected the optimal hyperparameters as 64 dimensions of the middle layer and 16 dimensions of the last layer. We add a normalization layer after each GCN layer to accelerate the convergence of GCN and reduce the overfitting problem. The final output of the GCN module is a  $L \times 16$ -dimensional feature vector where each node aggregates the surrounding information.

### 2.2.6 | Self-attention pooling

The output matrix of the GCN module is the residue level embedding, which is spliced with the pretrained attention pooling features to obtain the feature matrix  $M \in R^{L \times p}$  as the input of the attention pooling module. In order to obtain the global embedding, we use the multi-head self-attention mechanism:

$$T = \text{softMax}(W_1 M^T), \quad (3)$$

where  $M^T$  is the transpose of the feature matrix  $M$ ,  $W_1 \in R^{r \times p}$  is the trainable attention matrix with the hyperparameter  $r$  represents the number of attention heads. The  $\text{softMax}(\cdot)$  function standardizes the calculated weights so that the sum of each row is 1.0, indicating that the sum of the weights of each attention head is 1.0. Finally, we obtain a two-dimensional multi-head attention matrix  $T \in R^{r \times L}$ , which evaluates the relationship between each residue and the solubility change caused by mutation from different views and assigns a high weight to influential residues. We adjusted the number of attention heads from 1 to 10, and finally found that 4 attention heads performed best in the validation set. We extract  $r$  groups overall features by multiplying  $T$  and  $M$ , and average  $r$  groups overall features to obtain the final graph representation  $H \in R^{1 \times p}$ :

$$H = \frac{1}{r} \sum_{k=1}^r (TM)_k. \quad (4)$$

### 2.2.7 | FC layer

The output of the self-attention pooling module was input into the fully connected layer to predict the solubility change score (S) affected by the mutation by using the following formula:

$$S = \text{Tanh}(W_2 H^T + b), \quad (5)$$

where  $W_2 \in R^{1 \times p}$  is the weight matrix and  $b \in R$  is the bias. The function  $\text{Tanh}(\cdot)$  maps the final predicted value to the interval  $(-1, 1)$ .

## 2.3 | Training strategy and cross-validation

We use the labels  $-1$ ,  $0$ , and  $1$  to denote mutations that decrease solubility, mutations that do not affect solubility, and mutations that increase solubility, and our model outputs a solubility change prediction score ranging from  $-1$  to  $1$ . We use Adam optimizer to minimize the mean squared error loss function to train our model. We performed a 10-fold cross-validation on the training data set. Specifically, we divided the training data set into 10-folds on average according to three categories. Each time, one fold was used as validation data, and the remaining nine folds were used to train a model. This process was repeated for 10 times. The average of the 10 validation results was used as an estimate of the model performance. We used cross-validation for hyperparametric tuning and model selection, which resulted in 10 models. It took about 6 h for DeepMutSol to reach convergence. Training and Inference were conducted within a uniform computational environment, consisting of a GPU of GeForce RTX 2080, 96 CPU cores of Intel(R) Xeon(R) Gold 6248R, and the Ubuntu 20.04.6 LTS operating system.

## 2.4 | Evaluation metrics

In this paper, we used the same evaluation indicators as Pon-Sol and Pon-Sol2, and compared the evaluation results of our model with those of Pon-Sol and Pon-Sol2. We set two thresholds  $a$  and  $b$  in the interval of  $(-1, 0)$  and  $(0, 1)$ , respectively in order to divide the predicted solubility changes into three categories. We traverse thresholds  $a$  and  $b$  based on correct prediction ratio (ACC) on the validation set and use them for test set evaluation. We used positive predictive value (PPV), negative predictive value (NPV), sensitivity and specificity as indicators for single class evaluation as shown in formulas (6–9):

$$PPV = \frac{TP}{TP + FP}, \quad (6)$$

$$NPV = \frac{TN}{TN + FN}, \quad (7)$$

$$\text{sensitivity} = \frac{TP}{TP + FN}, \quad (8)$$

$$\text{specificity} = \frac{TN}{TN + FP}, \quad (9)$$

where TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives respectively. We also used the correct prediction ratio (ACC) and generalized square correlation (GCC) to assess overall performance. ACC represents the percentage of correctly predicted cases in the whole data set, and GCC is the classification correlation coefficient ranging from 0 to 1 and a larger value indicates better classification performance. ACC and GCC are defined as:

$$ACC = \frac{\sum_i z_{ii}}{N}, \quad (10)$$

$$GCC = \frac{\sum_{ij} \frac{(z_{ij} - e_{ij})^2}{e_{ij}}}{N(K-1)}, \quad (11)$$

where  $N$  is the number of all cases and  $K$  is the number of classes.  $z_{ij}$  represents the number of cases of class  $i$  predicted to be class  $j$ .  $e_{ij}$  represents the expected number of cases in the cell  $(i, j)$  of the confusion matrix, and is defined as:

$$e_{ij} = \frac{x_i \times y_j}{N}, \quad (12)$$

where  $x_i = \sum_j z_{ij}$  represents the number of the inputs associated with class  $i$ ,  $y_j = \sum_i z_{ij}$  represents the number of inputs predicted to be in class  $j$ .

Because the number of mutations in the three classes is very unbalanced, the evaluation indicators at this time cannot represent the real performance of the model. For example, the model can still obtain a high overall evaluation indicator when the prediction effect of the class with the largest number of mutations is very good and the prediction effect of other classes is very poor. Therefore, we need the following strategies to normalize evaluation indicators:

$$z_{ij}(\text{balanced}) = \frac{z_{ij}}{x_i} \times x_0, \quad (13)$$

where  $z_{ij}$  represents the number of cases of class  $i$  predicted to be class  $j$ ,  $x_i$  represents the number of cases whose label is class  $i$ , and  $x_0$  represents the number of cases whose label is class 0.

## 3 | RESULTS

### 3.1 | Performances on the 10-fold cross-validation and independent test

We evaluated our model through the 10-fold cross-validation (CV) and independence tests. Due to the uneven distribution of cases



for the three solubility categories, the method shown in Equation 13 was used to normalize the evaluation results. Table 1 shows the normalized results of 10-fold cross-validation and independence test. The scores of ACC and GCC were slightly lower than those of CV, but PPV, NPV, sensitivity, and specificity were similar to those of CV. In addition, there are significant differences in the assessment scores of the three categories. The normalized PPV for the category of decreased solubility and increased solubility are 0.746 and 0.692, respectively, much higher than the one for the category of unchanged solubility with only 0.494. For the normalized NPV, the prediction of decreased and unchanged solubility is almost as good (0.894 and 0.811, respectively), while the prediction of the increased solubility category is slightly worse (0.741). The sensitivity scores of the three categories show large differences, and the specificity values are relatively closer. DeepMutSol finally achieved normalized ACC and GCC of 0.618 and 0.242 on the test set, respectively.

As shown in Figure 2A, most of the solubility change prediction scores of the three categories are distributed within the division interval of their categories. The prediction scores of the categories with reduced solubility are mostly concentrated in the range of  $-1.0$  to  $-0.75$ , and the prediction scores of the categories with unchanged solubility are mostly concentrated around 0.

For the categories with increased solubility with the least training data, the prediction scores of solubility change are mostly in the range of 0.12 to 0.25. As shown in Figure 2B, The category with decreasing solubility had the highest AUC value of 0.89, while the category with increasing solubility had the lowest AUC value of 0.78. In addition to the cause of uneven data distribution, most mutations in nature tend to reduce solubility, so it is reasonable for the model to predict mutations with increased solubility slightly lower than those with decreased solubility.

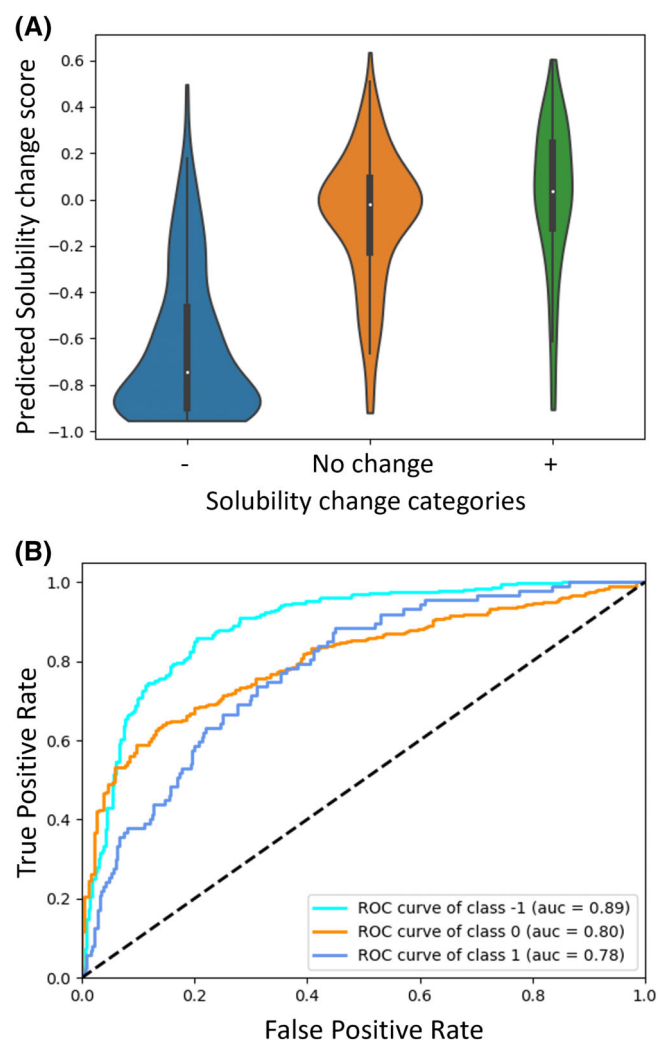
We can not only predict the solubility change of forward mutation, but also our model can be applied to predict the solubility change

of reverse mutation. In addition, the 3D structure used to predict the solubility change of reverse mutation is consistent with that of forward mutation, which only requires the 3D structure of wild-type protein, without increasing the computational burden. Theoretically, the solubility change values of forward mutation and reverse mutation are negatively correlated and can cancel each other. We used the test set to generate the corresponding reverse mutation data set, and Figure 3 shows the symmetric consistency of the model. As shown in Figure 3A, the Pearson correlation coefficient of solubility change predicted by forward mutation and reverse mutation was  $-0.96$ , and the corresponding p-value was 0.0. As shown in Figure 3B, we calculated the root mean square error of 0.13 by the sum of the predicted solubility changes of forward mutation and the corresponding predicted solubility changes of reverse mutation.

By observing Figure 3A, it can be found that the predicted value of solubility change of forward mutation will not be improved when it rises to 0.75, and the predicted value of solubility change of reverse mutation will not be below  $-0.75$  either. Proteome-wide analysis of

**TABLE 1** Performances on the 10-fold cross-validation and independent test.

		CV	Ind test
PPV	—	0.781	0.746
	N	0.532	0.494
	+	0.709	0.692
NPV	—	0.906	0.894
	N	0.821	0.811
	+	0.769	0.741
Sensitivity	—	0.815	0.796
	N	0.694	0.700
	+	0.453	0.356
Specificity	—	0.885	0.864
	N	0.693	0.641
	+	0.903	0.921
ACC		0.654	0.618
GCC		0.289	0.242

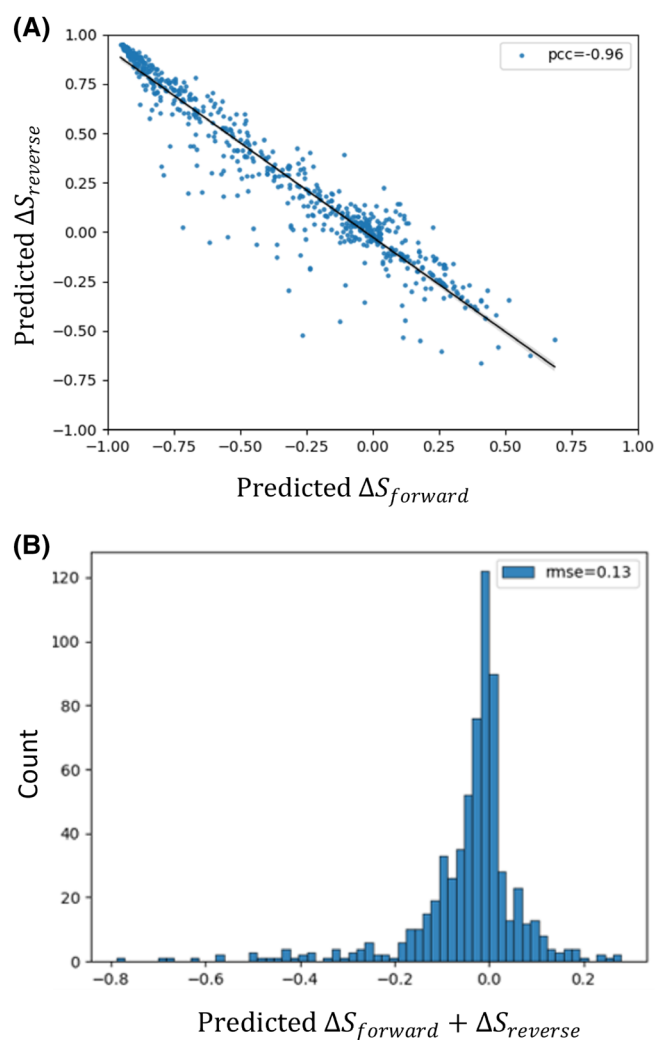


**FIGURE 2** (A) Violin maps plotted by the distribution of our predicted solubility change scores on three ground-truth solubility change categories. (B) ROC curve of three categories.

solubility in *Caenorhabditis elegans* indicated that about 75% of proteins appear in cells in Abundance close to their solubility limits.<sup>41</sup> Therefore, the increase in protein solubility of 0.75 is close to its solubility limit, the increase in protein solubility caused by mutation will not exceed the upper limit of 0.75.

### 3.2 | Ablation experiment

In order to solve the problem of the limited data set and the imbalance of data set, we introduced reverse mutation for data argument and two pretrained models for knowledge transfer. ProtTrans has been shown to be effective for protein downstream tasks and our pretrained protein solubility value prediction model obtained an average acc of 0.779 calculated with a solubility cutoff value of 0.32. In order to verify the effectiveness of these strategies, we conducted



**FIGURE 3** Prediction of solubility change for forward and reverse mutations in the test set. (A) Correlation between predicted solubility change values for forward and reverse mutations. (B) The deviation of the predicted solubility change value between the forward and reverse mutations.

ablation experiments, and the results were normalized using the method shown in Equation 13. As is shown in Table 2, Our final model has achieved the best results, with the normalized ACC and GCC of 0.618 and 0.242 on the test set, respectively. We replaced the pretrained ProtTrans features with features generated using PSSM, HHM features generated by PSI-BLAST,<sup>42</sup> and predicted structural features (SPD33) by SPIDER<sup>43</sup> for the evaluation of the baseline model. The removal of ProtTrans feature had the greatest impact on the prediction performance of the model. In this case, the normalized ACC of this model on the test set is 0.422 and the normalized GCC is 0.066, but the prediction effect is still better than that of Pon-Sol and SODA as shown in Table 3. This demonstrates that the ProTrans features are very helpful in predicting protein solubility changes caused by mutations. The effect of removing the reverse mutation of the training set on the prediction performance of the model is second only to that of removing ProtTrans feature, indicating that the data enhancement strategy is helpful to improve the prediction effect of the model. Removing the pretrained attention feature had minimal effect on the predictive performance of the model, but it also showed that the features learned on the solubility value prediction task with larger data sets are still useful for predicting protein mutation solubility changes.

### 3.3 | Comparisons with other methods

Our method is compared with the state-of-the-art methods, including Pon-Sol, SODA, Pon-Sol2, and our pretrained model, on the same test set. Among them, SODA is used to predict the solubility change score caused by protein mutations, and Pon-Sol and Pon-Sol2 are both three-classification models as ours. In order to use SODA in the three-classification task for comparison with our method, we set a threshold for the solubility change prediction score of SODA to distinguish three categories of solubility change. In order to compare with the prediction effect of SODA, Pon-Sol2 sets the three thresholds of 5, 10, and 17 for SODA, in which the positive and negative values of each threshold divide the prediction results of SODA into three categories. Similarly, our pretrained model is a regression model to predict the absolute solubility value. We used it to predict the solubility value of wild-type sequence and mutated sequence. The gap

**TABLE 2** Feature ablation study on cross-validation and the independent test set.

	CV		Ind test	
	ACC	GCC	ACC	GCC
DeepMutSol	0.654	0.289	0.618	0.242
w/o Data argument	0.645	0.273	0.587	0.193
w/o Pooling attention <sup>a</sup>	0.648	0.289	0.602	0.213
w/o ProtTrans <sup>b</sup>	0.375	0.014	0.422	0.066

<sup>a</sup>Pretrained attentional pooling features extracted from a pretrained solubility prediction model.

<sup>b</sup>Replacing the ProtTrans node features with PSSM+ HHM+ SPD33.

**TABLE 3** Comparisons of different methods on the independent test set, where all methods were evaluated by un-normalized, normalized PPV, NPV, sensitivity, specificity, ACC and GCC.

		PonSol	SODA (5 as threshold)	SODA (10 as threshold)	SODA (17 as threshold)	PonSol2	DeepMutSol
PPV	–	0.593/0.428	0.606/0.428	0.673/0.468	0.742/0.585	0.804/0.643	0.851/0.746
	N	0.427/0.385	0.425/0.365	0.397/0.357	0.383/0.350	0.600/0.475	0.601/0.494
	+	0.151/0.373	0.047/0.149	0.060/0.184	0.098/0.284	0.233/0.472	0.443/0.692
NPV	–	0.514/0.691	0.508/0.684	0.502/0.677	0.501/0.677	0.794/0.887	0.801/0.894
	N	0.685/0.700	0.761/0.739	0.797/0.782	0.847/0.833	0.804/0.793	0.816/0.811
	+	0.881/0.693	0.848/0.633	0.858/0.649	0.866/0.664	0.879/0.684	0.905/0.741
Sensitivity	–	0.263/0.263	0.195/0.195	0.098/0.098	0.068/0.068	0.802/0.802	0.796/0.796
	N	0.456/0.456	0.759/0.759	0.886/0.886	0.954/0.954	0.671/0.671	0.700/0.700
	+	0.448/0.448	0.069/0.069	0.057/0.057	0.046/0.046	0.161/0.161	0.356/0.356
Specificity	–	0.812/0.824	0.867/0.869	0.951/0.944	0.975/0.976	0.796/0.777	0.855/0.864
	N	0.659/0.636	0.426/0.340	0.249/0.204	0.144/0.116	0.751/0.630	0.741/0.641
	+	0.617/0.623	0.786/0.802	0.863/0.872	0.936/0.942	0.920/0.910	0.932/0.921
ACC		0.356/0.389	0.381/0.341	0.375/0.347	0.382/0.356	0.671/0.545	<b>0.704/0.618</b>
GCC		0.010/0.011	0.041/0.045	0.022/0.022	0.016/0.016	0.181/0.157	<b>0.241/0.242</b>

Note: The bold values represent the best performance.

between wild-type solubility and mutated solubility was then transferred into three classes by two thresholds.

Table 3 shows the comparison results between our method and other methods, where the first performance indicator is not standardized and the second is standardized through Equation 13. It can be seen that the standardized ACC (0.38) of SODA is slightly lower than that of Pon-Sol (0.39). Our pretrained model obtained better standardized ACC (0.40) than PonSol and SODA, but is still worse than that of Pon-Sol2 (0.55). That is reasonable, since it is hard for regression models to capture the solubility change caused by single mutation. It was this observation that motivated us to develop DeepMutSol as a three-classification method specifically for predicting solubility changes caused by mutation, rather than a solubility regression method. By comparison, our method achieved the highest performance, with the normalized ACC of 0.618 and the normalized GCC of 0.242. By comparing the prediction scores of the three categories with the other methods, it can be seen that our model is a relatively more balanced predictor, and the other comparison methods show large differences in the prediction of different categories. Due to the problem of data imbalance, the prediction performance of other comparison methods is mainly shown in the prediction of solubility decreasing and solubility unchanged categories, while the prediction performance of solubility increasing categories is poor.

Compared with other methods, on the premise of maintaining the prediction effect on the category of decreasing solubility, our method has significantly improved the prediction effect on the two categories of constant solubility and increasing solubility, especially for the category of increasing solubility with the least amount of data.

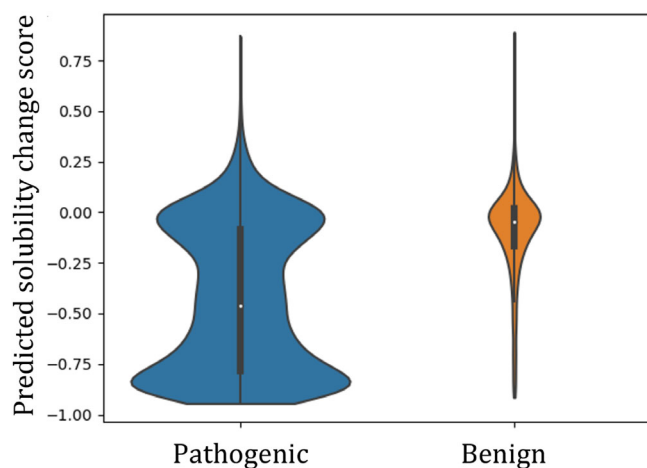
We have also calculated the inference time consumption of independent test set. When all features available, it took DeepMutSol 0.69 s to predict 662 solubility change caused by mutation, demonstrating the efficiency of DeepMutSol.

### 3.4 | Study on clinical annotated pathogenic mutations

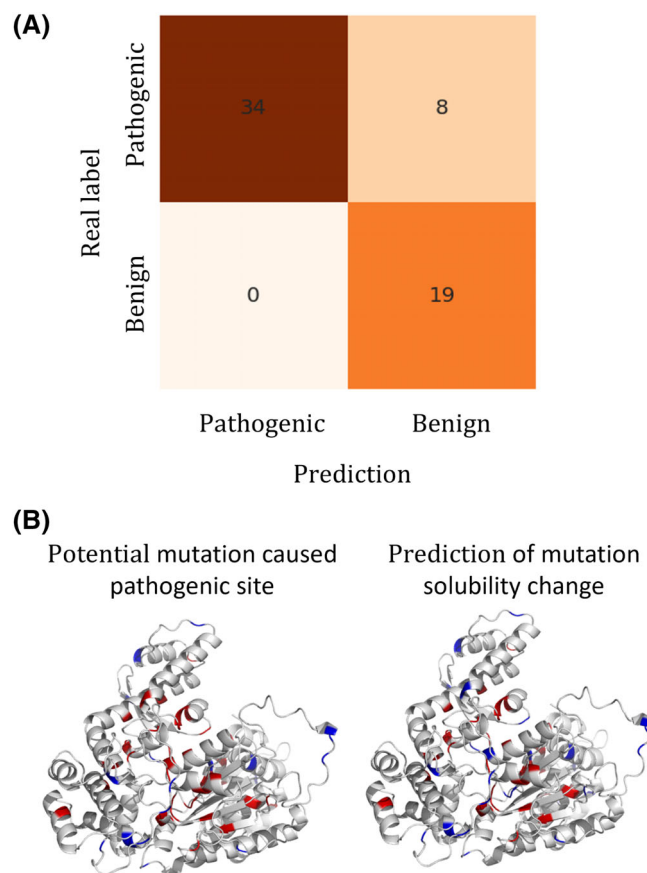
To investigate the correlation between the mutational solubility change score predicted by our model and the pathogenicity of mutations, we obtained a data set of mutations with annotated clinical significance in ClinVar<sup>37</sup> from Frazer et al. study.<sup>38</sup> We removed all mutations of uncertain pathogenicity from the data set and screened only one mutation at each site of the protein and prioritized pathogenicity mutation. We ended up with 5489 pathogenic mutations and 1298 benign mutations (details shown in Data S2). We used DeepMutSol to predict the solubility change scores of all mutations. The predicted solubility change scores were compared with the clinical pathogenicity labels for all mutations, the AUC between the predicted mutation solubility change score and the mutation pathogenicity was 0.759. We drew a violin diagram of the predicted solubility change scores of all pathogenic mutations and all benign mutations, as shown in Figure 4. It can be seen that most of the predicted solubility change scores of DeepMutSol for pathogenic mutations were less than or near zero, and the predicted solubility of benign mutations was basically unchanged. The Figure 4 indicating that the protein mutation solubility change score predicted by DeepMutSol was significantly correlated with the mutation pathogenicity of the protein, indicating that DeepMutSol has the potential to be used in disease pathway analysis and drug discovery.

We selected a protein STXB1\_HUMAN from the data set as a case for individual analysis. There were 42 pathogenic mutations and 19 benign mutations in this protein. According to the predicted protein mutation solubility change score, mutations were predicted into pathogenic and benign mutations, and we plotted the confusion matrix shown in Figure 5A. Meanwhile, the relationship between mutation pathogenicity and predicted mutation solubility change





**FIGURE 4** The violin map of the predicted solubility change scores on pathogenic mutation and benign mutation.



**FIGURE 5** (A) The confusion matrix of protein mutation pathogenicity predicted by DeepMutSol. (B) Disease-related mutations in the STBX1\_HUMAN protein and their predicted solubility changes. Left: red for pathogenic mutations, blue for benign mutations; Right: red represents the predicted solubility change score  $S \leq -0.14$ , blue represents the predicted solubility change score  $S > -0.14$ .

score was also drawn, as shown in Figure 5B. The left graph corresponds to the pathogenic mutation and the benign mutation, and the right graph corresponds to the predicted solubility change score. It

can be seen that except for eight mutation sites that were not correctly predicted, the remaining mutation sites could be correctly classified as benign mutation and pathogenic mutation according to the predicted solubility change value. This suggests that there is a good correlation between mutation solubility changes predicted by our model and mutation pathogenicity.

## 4 | CONCLUSIONS

In this work, we developed the first deep-learning-based method to predict protein mutation solubility changes. We used AlphaFold to predict the 3D structure of proteins before mutation to solve the problem of the lack of 3D structure data set for the protein mutation solubility change task. To the best of our knowledge, we were the first to apply the 3D structure of proteins to the task of predicting mutation solubility changes. At the same time, we also used the pretrained protein language model and the pretrained protein solubility value prediction model for knowledge transfer to obtain more effective features to overcome the problem of limited data sets. Using these two pretrained models to generate features instead of time-consuming computation of PSSM, HHM, and SPD33 offers higher computational efficiency. Experiments showed that our model achieved better prediction performance than other comparative methods, and also verified that knowledge transfer was effective for predicting protein mutation solubility changes on related tasks with more data sets. Finally, we used DeepMutSol to predict solubility changes on clinical pathogenicity related data sets. The correlation between predicted mutation solubility changes and mutation pathogenicity shows that our model has the potential to be applied to disease pathway analysis and drug discovery.

## ACKNOWLEDGMENTS

This study has been supported by the National Key R&D Program of China (2022YFF1203100), National Natural Science Foundation of China (12126610), Guangdong Key Field R&D Plan (2019B020228001 and 2018B010109006), Guangzhou S&T Research Plan (202007030010).

## DATA AVAILABILITY STATEMENT

All data used for training and testing the model, including all wild-type protein sequences, mutation sites, amino acid mutation types, solubility change labels, and three-dimensional structures of all wild-type proteins predicted by AlphaFold, were placed in two ZIP packages of SI. Clinical data to validate the association between mutation solubility changes predicted by our model and mutation pathogenicity are presented in the XLSX file of SI. The code developed and validated in this project is available on GitHub (<https://github.com/biomed-AI/DeepMutSol>).

## ORCID

Jing Wang <https://orcid.org/0000-0002-7453-6890>

## REFERENCES

- [1] A. K. Pavlou, J. M. Reichert, *Nat. Biotechnol.* **2004**, 22, 1513.
- [2] B. Leader, Q. J. Baca, D. E. Golan, *Nat. Rev. Drug Discov.* **2008**, 7, 21.
- [3] M. Goodman, *Nat. Rev. Drug Discov.* **2009**, 8, 837.
- [4] P. J. Carter, *Exp. Cell Res.* **2011**, 317, 1261.
- [5] J. G. Elvin, R. G. Couston, C. F. van der Walle, *Int. J. Pharm.* **2013**, 440, 83.
- [6] F. Chiti, C. M. Dobson, *Annu. Rev. Biochem.* **2017**, 86, 27.
- [7] M. C. Manning, D. K. Chou, B. M. Murphy, R. W. Payne, D. S. Katayama, *Pharm. Res.* **2010**, 27, 544.
- [8] S. Frokjaer, D. E. Otzen, *Nat. Rev. Drug Discov.* **2005**, 4, 298.
- [9] S. L. Baker, A. Munasinghe, B. Kaupbayeva, N. Rebecca Kang, M. Certiat, H. Murata, K. Matyjaszewski, P. Lin, C. M. Colina, A. J. Russell, *Nat. Commun.* **2019**, 10, 4718.
- [10] S. Costa, A. Almeida, A. Castro, L. Domingues, *Front. Microbiol.* **2014**, 5, 63.
- [11] R. M. P. Siloto, R. J. Weselake, *Biocatal. Agric. Biotechnol.* **2012**, 1, 181.
- [12] F. H. Arnold, *Curr. Opin. Biotechnol.* **1993**, 4, 450.
- [13] U. P. Andley, M. A. Reilly, *Exp. Eye Res.* **2010**, 90, 699.
- [14] A. Meulemans, S. Seneca, T. Pribyl, J. Smet, V. Alderweirdt, A. Waeytens, W. Lissens, R. van Coster, L. de Meirleir, J. P. di Rago, D. L. Gatti, S. H. Ackerman, *J. Biol. Chem.* **2010**, 285, 4099.
- [15] K. L. Maxwell, A. K. Mittermaier, J. D. Forman-Kay, A. R. Davidson, *Protein Sci.* **1999**, 8, 1908.
- [16] T. Arakhamia, C. E. Lee, Y. Carlomagno, M. Kumar, D. M. Duong, H. Wesseling, S. R. Kundinger, K. Wang, D. Williams, M. DeTure, D. W. Dickson, C. N. Cook, N. T. Seyfried, L. Petrucelli, J. A. Steen, A. W. P. Fitzpatrick, *Cell* **2020**, 180, 633.
- [17] F. A. Aprile, P. Sormanni, M. Perni, P. Arosio, S. Linse, T. P. J. Knowles, C. M. Dobson, M. Vendruscolo, *Sci. Adv.* **2017**, 3, e1700488.
- [18] P. Sormanni, F. A. Aprile, M. Vendruscolo, *Proc. Natl. Acad. Sci.* **2015**, 112, 9902.
- [19] Y. Tian, C. Deutsch, B. Krishnamoorthy, *Algorithms Mol. Biol.* **2010**, 5, 33.
- [20] P. Sormanni, F. A. Aprile, M. Vendruscolo, *J. Mol. Biol.* **2015**, 427, 478.
- [21] J. Van Durme, G. De Baets, R. Van Der Kant, M. Ramakers, A. Ganesan, H. Wilkinson, R. Gallardo, F. Rousseau, J. Schymkowitz, *Protein Eng. Des. Sel.* **2016**, 29, 285.
- [22] R. Guerois, L. N. J. Fau, L. Serrano, *J. Mol. Biol.* **2002**, 320, 369.
- [23] A.-M. Fernandez-Escamilla, F. Rousseau, J. Schymkowitz, L. Serrano, *Nat. Biotechnol.* **2004**, 22, 1302.
- [24] H. Land, M. S. Humble, *Methods Mol. Biol.* **2018**, 1685, 43.
- [25] L. Paladin, D. Piovesan, S. C. E. Tosatto, *Nucleic Acids Res.* **2017**, 45, W236.
- [26] Y. Yang, A. Niroula, B. Shen, M. Vihinen, *Bioinformatics* **2016**, 32, 2032.
- [27] Y. Yang, L. Zeng, M. Vihinen, *Int. J. Mol. Sci.* **2021**, 22, 8027.
- [28] C. Cai, S. Wang, Y. Xu, W. Zhang, K. Tang, Q. Ouyang, L. Lai, J. Pei, *J. Med. Chem.* **2020**, 63, 8683.
- [29] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, B. Rost, ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning. *bioRxiv* 2020.07.12.199554. **2021**.
- [30] Q. Yuan, S. Chen, Y. Wang, H. Zhao, Y. Yang, *Alignment-free metal ion-binding site prediction from protein sequence through pretrained language model and multi-task learning*. *bioRxiv* 2022.05.20.492769. **2022**.
- [31] J. Chen, S. Zheng, H. Zhao, Y. Yang, *J. Chem.* **2021**, 13, 7.
- [32] X. Lv, J. Chen, Y. Lu, Z. Chen, N. Xiao, Y. Yang, *J. Chem. Inf. Model.* **2020**, 60, 2388.
- [33] S. Chen, Z. Sun, L. Lin, Z. Liu, X. Liu, Y. Chong, Y. Lu, H. Zhao, Y. Yang, *J. Chem. Inf. Model.* **2020**, 60, 391.
- [34] Q. Yuan, J. Chen, H. Zhao, Y. Zhou, Y. Yang, *Bioinformatics* **2022**, 38, 125.
- [35] S. Zheng, Y. Li, S. Chen, J. Xu, Y. Yang, *Nat. Mach. Intell.* **2020**, 2, 134.
- [36] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Židek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohli, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, D. Hassabis, *Nature* **2021**, 596, 583.
- [37] J. Frazer, P. Notin, M. Dias, A. Gomez, J. K. Min, K. Brock, Y. Gal, D. S. Marks, *Nature* **2021**, 599, 91.
- [38] M. J. Landrum, J. M. Lee, M. Benson, G. R. Brown, C. Chao, S. Chitipiralla, B. Gu, J. Hart, D. Hoffman, W. Jang, K. Karapetyan, K. Katz, C. Liu, Z. Maddipati, A. Malheiro, K. McDaniel, M. Ovetsky, G. Riley, G. Zhou, J. B. Holmes, B. L. Kattman, D. R. Maglott, *Nucleic Acids Res.* **2018**, 46, D1062.
- [39] S. Khurana, R. Rawi, K. Kunji, G. Y. Chuang, H. Bensmail, R. Mall, *Bioinformatics* **2018**, 34, 2605.
- [40] Y. Yang, J. Zhan, H. Zhao, Y. Zhou, *Proteins: Struct., Funct., Bioinf.* **2012**, 80, 2080.
- [41] A. A. Schäffer, L. Aravind, T. L. Madden, S. Shavirin, J. L. Spouge, Y. I. Wolf, E. V. Koonin, S. F. Altschul, *Nucleic Acids Res.* **2001**, 29, 2994.
- [42] R. Heffernan, A. Dehzangi, J. Lyons, K. Paliwal, A. Sharma, J. Wang, A. Sattar, Y. Zhou, Y. Yang, *Bioinformatics* **2016**, 32, 843.
- [43] G. Vecchi, P. Sormanni, B. Mannini, A. Vandelli, G. G. Tartaglia, C. M. Dobson, F. U. Hartl, M. Vendruscolo, *Proc. Natl. Acad. Sci. USA* **2020**, 117, 1015.

## SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

**How to cite this article:** J. Wang, S. Chen, Q. Yuan, J. Chen, D. Li, L. Wang, Y. Yang, *J. Comput. Chem.* **2024**, 45(8), 436.  
<https://doi.org/10.1002/jcc.27249>