Briefings in Bioinformatics, 22(6), 2021, 1–16

https://doi.org/10.1093/bib/bbab248
Problem Solving Protocol

Improving protein fold recognition using triplet network and ensemble deep learning

Yan Liu†, Ke Han†, Yi-Heng Zhu, Ying Zhang, Long-Chen Shen, Jiangning Song and Dong-Jun Yu

Corresponding authors: Dong-Jun Yu, School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei, Nanjing 210094, China. E-mail: njyudj@njust.edu.cn; Jiangning Song, Biomedicine Discovery Institute and Department of Biochemistry and Molecular Biology, Monash University, Melbourne, Victoria 3800, Australia. E-mail: jiangning.song@monash.edu
†These authors contributed equally to this work.

Abstract

Protein fold recognition is a critical step toward protein structure and function prediction, aiming at providing the most likely fold type of the query protein. In recent years, the development of deep learning (DL) technique has led to massive advances in this important field, and accordingly, the sensitivity of protein fold recognition has been dramatically improved. Most DL-based methods take an intermediate bottleneck layer as the feature representation of proteins with new fold types. However, this strategy is indirect, inefficient and conditional on the hypothesis that the bottleneck layer's representation is assumed as a good representation of proteins with new fold types. To address the above problem, in this work, we develop a new computational framework by combining triplet network and ensemble DL. We first train a DL-based model, termed FoldNet, which employs triplet loss to train the deep convolutional network. FoldNet directly optimizes the protein fold embedding itself, making the proteins with the same fold types be closer to each other than those with different fold types in the new protein embedding space. Subsequently, using the trained FoldNet, we implement a new residue–residue contact-assisted predictor, termed FoldTR, which improves protein fold recognition. Furthermore, we propose a new ensemble DL method, termed FSD_XGBoost, which combines protein fold embedding with the other two discriminative fold-specific features extracted by two DL-based methods SSAfold and DeepFR. The Top 1 sensitivity of FSD_XGBoost increases to 74.8% at the fold level, which is ∼9% higher than that of the state-of-the-art method. Together, the results

Yan Liu received his MS degree in computer science from Yangzhou University in 2019. He is currently a PhD candidate in the School of Computer Science and Engineering at Nanjing University of Science and Technology and a member of the Pattern Recognition and Bioinformatics Group. His research interests include pattern recognition, machine learning and bioinformatics.

Ke Han received her MS degree in computer science from Nanjing University of Science and Technology in 2009. She is currently a PhD candidate in the School of Computer Science and Engineering at Nanjing University of Science and Technology and a member of the Pattern Recognition and Bioinformatics Group. Her research interests include pattern recognition, machine learning and bioinformatics.

Yi-Heng Zhu received his BS degree in computer science from Nanjing Institute of Technology in 2015. He is currently a PhD candidate in the School of Computer Science and Engineering at Nanjing University of Science and Technology and a member of the Pattern Recognition and Bioinformatics Group. His research interests include pattern recognition, data mining and bioinformatics.

Ying Zhang received the MS degree in control science and engineering from Yangzhou University in 2020. She is currently a PhD candidate in the School of Computer Science and Engineering at Nanjing University of Science and Technology and a member of the Pattern Recognition and Bioinformatics Group. Her current research interests include machine learning, pattern recognition and bioinformatics.

Long-Chen Shen received his Bachelor in engineering degree in mechanical design, manufacturing and automation from Yancheng Institute of Technology in 2018. He is currently a master candidate in the School of Computer Science and Engineering, Nanjing University of Science and Technology and a member of the Pattern Recognition and Bioinformatics Group. His current interests include pattern recognition, data mining and bioinformatics.

Jiangning Song is an Associate Professor and group leader in the Monash Biomedicine Discovery Institute and the Department of Biochemistry and Molecular Biology, Monash University, Melbourne, Australia. He is also affiliated with the Monash Centre for Data Science, Faculty of Information Technology, Monash University. His research interests include bioinformatics, computational biomedicine, machine learning, data mining and pattern recognition.

Dong-Jun Yu received the PhD degree from Nanjing University of Science and Technology in 2003. He is currently a full professor in the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include pattern recognition, machine learning and bioinformatics. He is a senior member of the China Computer Federation and a senior member of the China Association of Artificial Intelligence.

Submitted: 10 May 2021; **Received (in revised form):** 4 June 2021

suggest that fold-specific features extracted by different DL methods complement with each other, and their combination can further improve fold recognition at the fold level. The implemented web server of FoldTR and benchmark datasets are publicly available at http://csbio.njust.edu.cn/bioinf/foldtr/.

Key words: protein fold recognition; bioinformatics; convolutional neural network; ensemble deep learning; triplet loss

Introduction

The tertiary structures of proteins have great significance, which can provide critical information to inform drug design [1] and aid in the characterization of protein–protein interactions [2] and protein functional annotation [3]. Protein folding pattern can reveal the evolution between protein molecules and tertiary structure [4]; it is widely known that proteins with identical fold type usually have similar structures and functions [5]. Therefore, accurate protein fold recognition can effectively facilitate the determination of protein tertiary structure. At the early stage, researches inferred the structures of proteins by wet laboratory experiments, such as X-ray crystallography [6], nuclear magnetic resonance (NMR) spectroscopy [7] and electron microscopy [8], which have achieved great success. However, these efforts are often laboratory intensive, time consuming and expensive. In this context, the development of intelligent computing methods for high-throughput and robust protein fold recognition is highly desirable.

Recently, a variety of computational methods [9–13] have been proposed for protein fold recognition. Based on their specific methods, these predictors can be divided into three major categories: machine learning methods, template alignment methods and ensemble methods. Machine learning methods formulate protein fold recognition as a multiclass classification task by using different classifiers. These methods are developed specifically based on two steps: (1) extracting discriminative protein feature representation from protein sequence. For example, Yan *et al.* [14] fused four different types of features (i.e. PSI-BLAST profile [15], HHblits profile [12], PSIPRED profile [16] and physicochemical profile [17]) by multiview learning to improve the representation of protein features and (2) powerful machine learning classifiers, such as support vector machine (SVM) [18], random forest (RF) [19] and naive Bayes [20]. These machine learning methods are effective and have demonstrated great success. For example, FOLDpro [21] employed SVM to combine various features describing pairwise similarities of any two proteins to improve fold recognition. Jo and Cheng [22] generated 84 similarity scores for a protein pair using various protein representations, and input such similarity scores into the RF classifiers to improve fold recognition.

Unlike machine learning methods, template alignment methods utilize sequence or structural information to match the query protein with the template protein in the protein library, where all proteins have known fold types. More specifically, template alignment methods can be further divided into two categories: sequence alignment and profile alignment methods. Sequence alignment methods are based on the similarity scores between the query protein sequence and the template protein sequence. The fold type of the template protein with the highest similarity score is assigned to the query protein. However, in the real world, the sequence similarity between protein pairs does not necessarily indicate their fold similarly, which poses a significant challenge for sequence alignment methods. To alleviate this problem, Remmert *et al.* [12] proposed the profile–profile alignment method and profile-sequence method using Markov random fields [23] and Hidden Markov Model (HMM) [24] to improve fold recognition. More recently, Xia *et al.* [25] used the sequence profile template generated by HMM to explore the relationship between the query protein and template-based profile. In summary, the profile alignment methods can generally achieve a better performance than sequence–sequence alignment methods.

Different from machine learning methods and template alignment methods, ensemble methods employ certain strategies to enable several classifiers or protein features to work together to improve protein fold recognition. For example, Shen and Chow [26] used different classifiers to learn the different protein characteristics separately and classified the protein fold types by weighted voting. Yan *et al.* [14] proposed the MT-fold method to improve protein fold recognition by incorporating the advantages of both machine learning methods and template alignment methods. Based on the results of the aforementioned methods, a consensus is that extracting the powerful protein feature representation is the critical step for improving protein fold recognition.

Recently, deep learning (DL) techniques have revolutionized the fields of computer vision [27, 28], natural language processing [29] and speech recognition [30]. This trend is also emerging in the field of bioinformatics and computational biology [31, 32]. For example, Li *et al.* [33] used a residual network to improve protein residue–residue contact prediction. Jaganathan *et al.* [34] developed a DL technique to predict splicing from primary sequence directly. In this important field, DL-based protein fold recognition methods have also been extensively developed, and accordingly these methods significantly outperform the traditional methods [9, 35]. These advances are largely due to the powerful feature extraction ability of deep convolutional neural networks (DCNNs), which can automatically extract task-specific features without human intervention. For example, Zhu *et al.* [35] proposed DeepFR for protein fold recognition by using DCNN to extract fold-specific features from the predicted protein residue–residue contacts. Liu *et al.* [9] proposed a CNN-BLSTM model for protein fold recognition by combining DCNN and Long Short-Term Memory (LSTM) [36] to detect the patterns in protein sequences effectively and captured their global sequence-order effects along with the proteins. Li *et al.* [10] designed a motif-based convolutional neural network to extract more powerful fold-specific feature representations from the position-specific scoring matrix [9] and protein residue–residue contact map.

The aforementioned DL-based methods have contributed significantly to the advances of this important field. However, they also have certain drawbacks and suffer from the following shortcomings: (1) they typically use a classification layer in the first instance to train a set of known protein fold types and then take an intermediate bottleneck layer as the feature representation to perform fold recognition. However, this strategy is indirect and inefficient as it can only work well under the condition that the representation of the bottleneck layer has a good representation capability for proteins with new protein fold types; (2) the insufficient numbers of training samples and

the class imbalance problem, which make the trained DCNN tend to favor categories with larger sample size and ignores the categories with an insufficient sample size; (3) the feature representation learned by a single neural network is often one sided. It cannot adequately reflect the differences in the proteins. Collectively, these shortcomings will eventually make the trained network overfitted.

To overcome the aforementioned disadvantages, in this work, we propose two new methods to improve protein fold recognition, namely FoldTR and FSD-XGBoost. More specifically, we first design a DL-based feature extractor, termed FoldNet. It can be used to extract protein fold embeddings from the predicted residue–residue contact maps of protein sequences for fold recognition automatically. In contrast to the DL-based methods mentioned above, FoldNet uses triplet loss to directly optimize the protein fold embeddings to facilitate the follow-up fold recognition, rather than using an intermediate bottleneck layer. Besides, FoldNet also ensures that proteins with the identical fold types are close together and proteins with different fold types are farther apart. On the other hand, to further improve protein fold recognition using DL techniques, we use XGBoost to ensemble another two DL-based methods (i.e. SSAFold [37] and DeepFR). This method is termed FSD-XGBoost. More details about FoldNet and FSD-XGBoost will be provided in the following section.

Finally, we implement an online web server, referred to as FoldTR, for protein fold recognition based on the proposed FoldNet. The FoldTR webserver and the curated benchmark datasets are freely available at http://csbio.njust.edu.cn/bioinf/foldtr/. The performance comparison results on the benchmark datasets show that the proposed FoldTR and FSD-XGBoost methods achieve the best performance and can serve as useful tools to facilitate community-wide research efforts for protein fold prediction.

Materials and methods

Benchmark datasets

Training dataset

In this study, we collected 23 001 proteins covering 1198 fold types from the SCOP database [38] as the training dataset to infer the parameters of the proposed FoldNet, which is the key feature extraction module of the FoldTR predictor. Specifically, we used the CD-HIT-2D program [39] to remove any homologous sequences in the training dataset with >40% sequence identity to those in the test dataset to guarantee the independence between the training and test datasets.

Test dataset

We evaluated the performance of the proposed method on the widely used Lindahl's dataset [40]. There are 976 proteins in Lindahl's dataset, among which 321 proteins have at least one match with other proteins at the fold level, 434 proteins at the superfamily level and 555 proteins at the family level, respectively. In this dataset, the sequence similarity between any two proteins is <40%.

Training dataset and test dataset for FSD-XGBoost

We performed 10-fold cross-validation to evaluate the performance of FSD-XGBoost. Specifically, we divided all pairs (i.e. query–template pairs) of proteins in Lindahl's dataset into 10 equally sized subsets, nine of which were used to train the XGBoost classifier, whereas the remaining subset was used for testing. It is noteworthy that we collected all the target–template pairs associated with the same query protein in the same subset.

Generation of predicted residue–residue contact map and size normalization

In this study, we employed DCNN to extract fold-specific features from the predicted protein residue–residue contact maps. There are a variety of protein residue–residue contact predictors, such as ResPRE [33], SSCpred [41], DeepCov [42], etc. The high precision in protein contact prediction achieved by these methods has motivated research efforts in the field of protein structure prediction. Different from the above methods, SSA (full) [43] is a protein residue–residue contact prediction tool based on LSTM. A significant advantage is that it can obtain results quickly and accurately by using protein sequence only. Therefore, in this study, we froze the parameters provided by the authors of SSA (full), fed each of the protein sequences into it and obtained the predicted protein residue–residue contact likelihood matrix.

It is necessary for different proteins to keep the embedding dimension consistent so that the subsequent distance measurement can proceed smoothly. However, different proteins often have different amino acid sequence lengths. To make FoldNet with the fixed-size input applicable to other proteins with larger or smaller sizes, we used sampling and padding to fix the input size at 224×224. A detailed description of our operations for such size normalization is provided below:

- Sampling: for protein sequences with the length of longer than 224 amino acids, we randomly selected a matrix with the size of 224×224 from the protein residue–residue contact likelihood matrix.
- Padding: for protein sequences with the length of shorter than 224 amino acids, we filled the empty parts with zero to obtain the protein residue–residue contact likelihood matrix.

To summarize, we first used the SSA (full) tool to obtain the predicted residue–residue contact map; then, we used the sampling and padding strategies to normalize the predicted residue–residue contact map to a fixed size, and finally, we obtained the protein samples consisting of the normalized protein residue–residue contact likelihood map and the corresponding fold type.

FoldNet

In this study, we proposed a unified computational framework for protein fold recognition. This framework is based on the effective learning of protein fold embedding for each protein sequence using DCNN. The network is trained in such a way that the distances in the protein embedding space directly correspond to protein similarity. Protein embeddings with the same fold type have relatively small distances, whereas protein embeddings with different fold types have large distances. As such, different metric functions can theoretically be used, such as the Cosine distance, European distance, etc. Once the protein fold embeddings are produced, addressing the protein fold recognition task becomes straightforward in that it can be solved as the nearest neighbor classification problem. Nevertheless, in this case, the commonly used cross-entropy loss, as the objective function, cannot meet the aforementioned requirements. Therefore, a critical step to achieve the above goals is to employ an appropriate objective function to effectively guide the training of DCNN. On the other hand, it is widely believed that

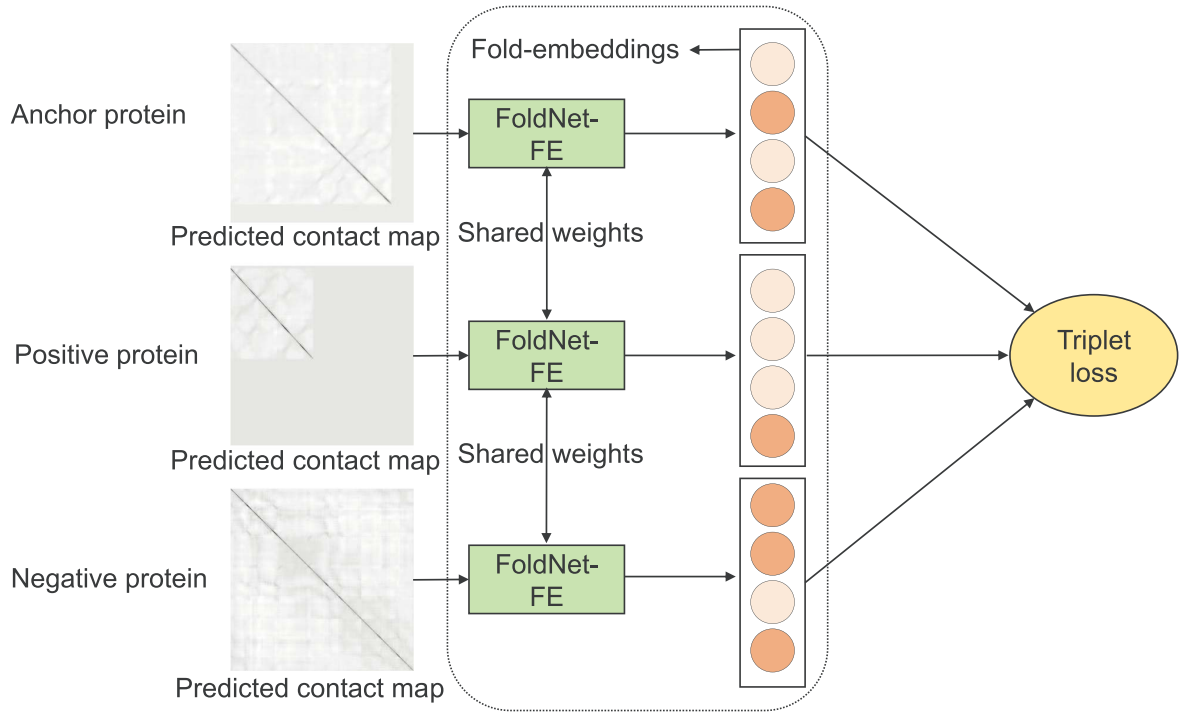

Figure 1. The structure of FoldNet. An anchor protein in the figure indicates any protein in the database; the positive protein is a protein in the database with the same fold type as anchor protein, whereas the negative protein is a protein with a different fold type from anchor protein. The network structure of FoldNet is made up of three identical, structured feature extraction engines; we named the feature extraction engine as FoldNet-FE.

the powerful DCNN structure can also positively impact protein fold recognition. In view of these critical points and previous works in computer vision [44–46], in this study, we designed a new DL-based method to extract protein fold embedding by using the triplet loss. Our method is named FoldNet, whose structure is illustrated in Figure 1. The major components in the Figure 1 will be introduced in the next section.

Triplet loss

In this section, we first briefly introduce the triplet loss. Then, we will elaborate on it through mathematical formulas. In addition, we treat FoldNet-FE as a 'black box' and use $f(.)$ (i.e. the output of FoldNet-FE) as the protein fold embedding function that maps a protein to a point in the protein fold embedding space. More details about FoldNet-FE will be provided in the section 'The structure of DCNN used in FoldNet'. Figure 2 illustrates the learning goal of triplet loss.

In Figure 2, there exist some terminologies including the anchor protein (A), negative protein (N) and positive protein (P) (A, P and N form a protein triplet, A and P form the positive protein pair, whereas A and N form the negative pair, respectively). To apply the triplet loss, we need to compare the distance between pairs of proteins. For example, given the pair of proteins on the left side in Figure 2, their fold embedding is desirable to be closer because they share the same fold type in the protein embedding space. On the other hand, given the pair of proteins on the right side in Figure 2, their fold embedding would be preferable to be quite different because these proteins share a different fold type. In summary, the goal of the triplet loss is to ensure a protein (i.e. anchor) of a specific fold type is closer to all other proteins (i.e. positive) of the same fold type than it is to other negative proteins.

To formalize the goal of triplet loss, the extracted protein fold embedding has the following property:

$$d\left(f(A), f(P)\right) \leq d\left(f(A), f(N)\right), \tag{1}$$

where $d(.,.)$ denotes the distance function in the protein embedding space. Next, a slight change to this expression is made as follows:

$$d\left(f(A), f(P)\right) - d\left(f(A), f(N)\right) \leq 0. \tag{2}$$

Suppose that in an extreme case, $f(.)$ always outputs the same embedding for any protein, and as a result it can almost trivially satisfy Equation (2). In other words, we need to ensure that FoldNet-FE does not output the same embeddings for all proteins. To achieve the above requirements, we modify this objective so that the distance value between the positive protein pairs and negative protein pairs needs to be a bit smaller than zero. Accordingly, we obtain the following modified objective:

$$d\left(f(A), f(P)\right) - d\left(f(A), f(N)\right) \leq 0 - m. \tag{3}$$

Furthermore, the modified objective can be expressed as follows:

$$d\left(f(A), f(P)\right) + m - d\left(f(A), f(N)\right) \leq 0, \tag{4}$$

where m is the margin that enforces between the positive and negative pairs.

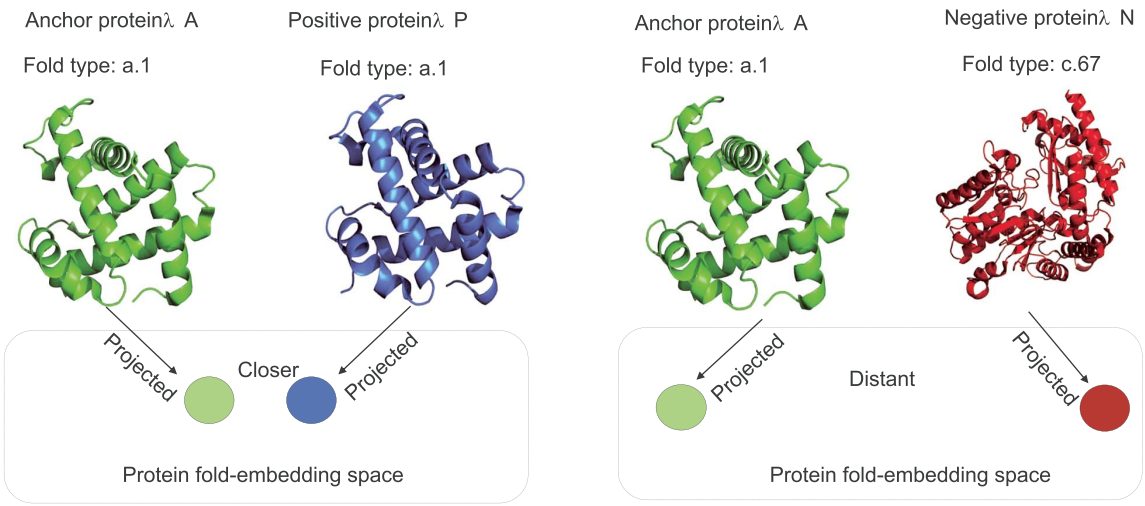

Figure 2. The learning goal of triplet loss is to minimize the distance between an anchor protein and a positive protein, both of which share the same fold type, and maximize the distance between the anchor protein and a negative protein of a different fold type.

To facilitate the training of DCNN, we first define an individual triplet loss function as follows:

$$L(A,P,N) = \max\left(d\left(f(A),f(P)\right) + m - d\left(f(A),f(N)\right), 0\right). \quad (5)$$

As long as we achieve the goal of making $d(f(A),f(P)) + m - d(f(A),f(N))$ less than or equal to zero, the loss on this example would be equal to zero, or otherwise we would obtain a positive loss. To learn the parameters of FoldNet, the overall loss function for the FoldNet model can be a sum over a training set of these individual losses on different triplets:

$$J = \sum_{i=1}^{\Re} L(A,P,N)_i, \forall \Re \in N, \quad (6)$$

where $\Re$ is the set of all possible triplets in the training set and has cardinality N.

How do we actually choose these triplets to form the training set?

The first question that comes to mind is how to choose proteins A, P and N to construct the triplets. If we choose them at random, then Equation (4) is very easy to satisfy (from Supplementary Table S1, we can see that this strategy makes it difficult for FoldNet to converge). However, given two randomly chosen proteins of the same fold type, the chances are that A and N are 'more different' than A and P. As such, the DCNN cannot learn much from it. This will slow down the network convergence.

To address the above problem, one feasible option is to construct triplets using the hardest proteins to distinguish from the anchor protein to enable fast network convergence. Specifically, given an anchor protein A, we first need to identify the most indistinguishable protein P (i.e. the hardest positive protein, which is a positive protein that is most distant from the anchor protein) that shares the same fold type with A in the entire dataset. Then, we find out the most similar protein N (i.e. the hardest negative protein, which is the closest negative protein to the anchor protein) that shares a different fold type in the entire dataset. As a result, we construct a triplet using A, the hardest positive protein and the hardest negative protein. From the theoretical perspective, if these triplets can be distinguished, then the others can be distinguished easily. This process is illustrated in Figure 3.

However, it is infeasible to find the qualified samples across the whole training set, due to the limitation of computer memory. Motivated by previous works [44–46], we selected the hardest positive and negative proteins from a mini-batch instead of picking the hardest ones from the whole training dataset. Nevertheless, for a meaningful representation of the anchor-positive distances, it is necessary that a minimal number of samples of any fold type are included in each mini-batch.

The structure of FoldNet

As shown in Figure 1, FoldNet consists of three identical DCNNs, termed FoldNet-FE, which serves as the feature extraction engine of FoldNet. Each FoldNet-FE contains eight convolution layers, five max-pooling layers, eight batchnorm layers, two fully connected layers and one L2 regularization layer. These are briefly described below:

Convolution layer: the convolution layers are used to capture the hidden local features from protein reside–residue contact maps, where each element of the feature map is connected with local patches of the previous layer through weighting and nonlinear transformation [47–49].

Max-pooling layer: the maxed pooling layer [50] is used to reduce redundant information and compress features. It is beneficial to reduce the network complexity and accelerate the network learning speed.

Batchnorm layer: the batch normalization [51, 52] is employed to accelerate deep network training by reducing the internal covariate shift. In addition, it can also prevent overfitting.

Fully connected layer: the fully connected layer can enable the nonlinear combination of the features learned in the previous layer. In contrast to the convolutional layer, each neuron in the fully connected layer is connected to all neurons in the previous layer.

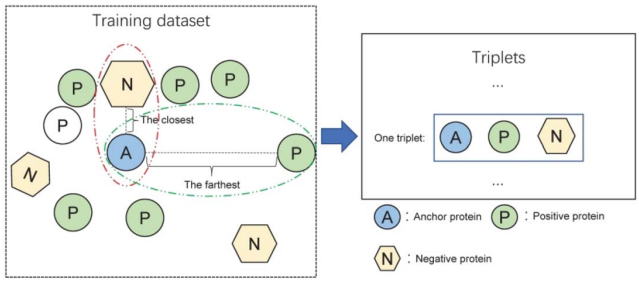

Figure 3. The process for constructing the training triplets.

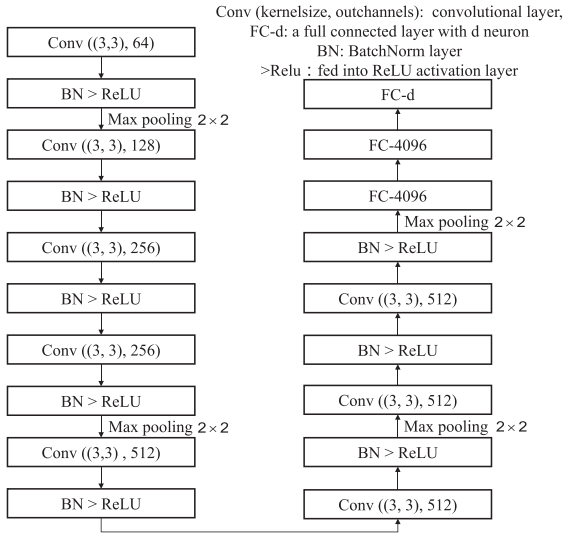

Figure 4. The detailed parameters of the FoldNet-FE structure, where the stride of BN layer and Max pooling layer is 2, and the stride of the Conv layer is (1,1).

L2 regularization layer: Inspired by [45], to better compare the distance between proteins, we used L2 regularization to fix the features on the d-dimensional hypersphere. The mathematical form can be expressed as follows:

$$\|f(x)\|_2 = 1. \tag{7}$$

The specific parameters of the FoldNet-FE structure are shown in Figure 4.

FoldTR

Figure 5 illustrates the pipeline of the proposed FoldTR framework for protein fold recognition. A detailed description of the pipeline is provided as follows:

Step 1. Learning the parameters of FoldNet

First, we learn the parameters of FoldNet using the mentioned strategy described in the Section 'FoldNet' using the training dataset. The details of the training process are provided in the Supplementary' Text S1.

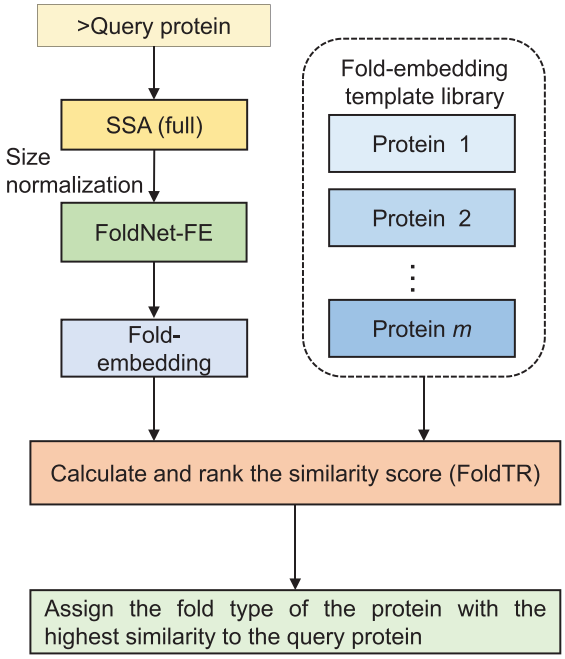

Figure 5. Pipeline of FoldTR for fold recognition.

Step 2. Constructing the feature template library

After learning and fixing the parameters of FoldNet-FE, we apply it to extract protein fold embeddings from the training dataset. More specifically, for extracting protein fold embeddings, we only need to feed one predicted contact map (after size normalization) into a single FoldNet-FE rather than a triplet. Furthermore, these protein fold embeddings constitute the feature template library.

Step 3. Assigning the fold type for a query protein

Up to now, we can transform the protein fold recognition task into the data retrieval task by finding the template protein that is most similar to the query protein from the feature template library. Our detailed procedures are described as follows:

First, for a given query protein, we use the SSA (full) tool to generate the protein residue–residue contact map and normalized it by following the strategy described in the section 'Obtaining predicted residue–residue contact map and size normalization'. Then, we use FoldNet-FE to extract the protein fold embedding of the query protein from the normalized

contact map. After that, we calculate the folding similarity score between the fold embedding of the query protein and each of the fold embeddings in the feature template library based on the Cosine score or Euclidean score defined in Equations (8) and (9), respectively.

Cosine score: for two protein fold embeddings f_{query} and f_{template}, the Cosine score can be calculated using the following equation:

$$S_F(\text{query}, \text{template}) = \frac{f_{\text{query}} \cdot f_{\text{template}}}{\|f_{\text{query}}\| \, \|f_{\text{template}}\|}. \quad (8)$$

Euclidean score: for two protein fold embeddings f_{query} and f_{query}, the Euclidean score can be calculated as:

$$D_F(\text{query}, \text{template}) = 1 - \sqrt{\sum_{k=1}^{d} \left(f^k_{\text{query}} - f^k_{\text{query}}\right)^2}, \quad (9)$$

where S_F(query, template) and D_F(query, template) denote the Cosine score and Euclidean score between the query protein and template protein (protein in the feature template library with known fold type), respectively. Afterwards, the calculated folding similarity scores are ranked in descending order. Finally, the fold type of the feature template that attains the highest score is assigned to the query protein.

FSD-XGBoost

Brief introduction of XGBoost

The XGBoost algorithm [53] is an improvement over the traditional Gradient Boosting Decision Tree (GBDT) [54], which has dramatically improved the computing speed, generalization ability and stability of GBDT. The target function of XGBoost can be formulated as follows:

$$F(\theta) = L(\theta) + \Omega(\theta) = l(\hat{y}_i y_i) + \left[\gamma T + \frac{1}{2}\|w\|^2\right], \quad (10)$$

where θ represents all the parameters in XGBoost, whereas $L(\theta)$ is the loss function to measure the difference between the predicted value and the true value. The two most commonly used loss functions are the mean square loss function and the logistic loss function. $\Omega(\theta)$ is the penalty term used to reverse the too complex model. T represents the number of leaves in the tree. $\gamma \in (0, 1)$ is the learning rate, γ multiplied by T means pruning the tree for $\frac{1}{2}\|w\|^2$, where w is the parameter of leaves. This operation can also prevent the overfitting and improve the generalization ability of the XGBoost model. However, it is not easy to optimize the objective function in Equation (10) directly. Thus, we rewrite it as the following form:

$$L(\theta) = \sum_{i=1}^{n} \left[l\left(y_i - \hat{y}^{(t-1)}\right) + S_t(T_i)\right] + \Omega(\theta). \quad (11)$$

A new goal of the objective function is to minimize the target function by constructing a tree in each iteration. In Equation (11), $S_t(T_i)$ is the tree generated for the sample i in the tth iteration. Furthermore, we can obtain a new target function based on the second-order Taylor expansion:

$$L(\theta) = \sum_{i=1}^{n} \left[l\left(y_i - \hat{y}^{(t-1)}\right) + g_i S_t(T_i) + \frac{1}{2} h_i S_t^2(T_i)\right] + \Omega(\theta), \quad (12)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$. Finally, we can find the optimal splitting point for each tree to minimize the objective function by iteration.

FSD-XGBoost

In recent years, the use of DCNN has significantly improved protein fold recognition. However, different neural network structures, different input data and different neural network loss functions may lead to differential distributions of fold-specific features (embeddings) extracted by DCNN, as depicted in Figure 6. Previous works [9, 10] indicate that all these fold-specific features contribute to protein fold recognition. Nevertheless, the development of methods that are capable of efficiently combining these features to improve protein fold recognition remains challenging.

In this study, to enable the full use of such fold-specific features, we proposed a new ensemble method, referred to as FSD-XGBoost, to combine these features to improve protein fold recognition. Specifically, we first choose three different fold-specific features (embedding) extracted by FoldNet, SSAfold and DeepFR. FoldNet uses triplet loss as the loss function. In contrast, SSAfold and DeepFR use cross-entropy as the loss function. SSAfold uses the potential protein residues–residue relationship as the input data, whereas FoldNet and DeepFR use the protein residue–residue contact likelihood matrix as the input. These differences lead to differential distributions of the learned features by the three methods, as illustrated in Figure 6. Next, we employed the following strategy to combine the three fold-specific features inspired by Jo and Cheng's work [22].

Suppose that we have a benchmark dataset consisting of n sequences $\{x^i, y^i\}_{i=1}^n$ with c fold types, where x^i represents the feature vector of the ith protein sequence and y^i is the fold type. For two given proteins q and p, its ensemble similarity score $X(q, p)$ based on different features can be expressed as follows:

$$X(q,p) = \left[S_F(q,p), S_S(q,p), S_D(q,p)\right], \quad (13)$$

where $S_F(q, p)$, $S_S(q, p)$ and $S_D(q, p)$ represent the similarity scores between the two proteins based on FoldNet, SSAfold and DeepFR, respectively. In addition, the label between the two protein sequences q and p can be expressed as:

$$Y(q,p) = \begin{cases} 1 & y^q = y^p \\ 0 & y^q \neq y^p \end{cases}. \quad (14)$$

As a result, we can obtain a new sample ($X(q,p)$, $Y(q,p)$). Finally, we traversed all protein entries in the Lindahl's dataset using the above strategy to obtain a new dataset.

First, we used the strategy described in the section 'Training dataset and test dataset for FSD-XGBoost' to generate the new training and test datasets. A detailed description of the training process is provided in the Supplementary Text S2. After that, we fed the new training dataset to XGBoost to learn the parameters of the XGBoost classifier. After the training, XGBoost predicted the probability of each class (1: in the same fold; 0: not in the same fold) by taking as the input an ensemble similarity score of

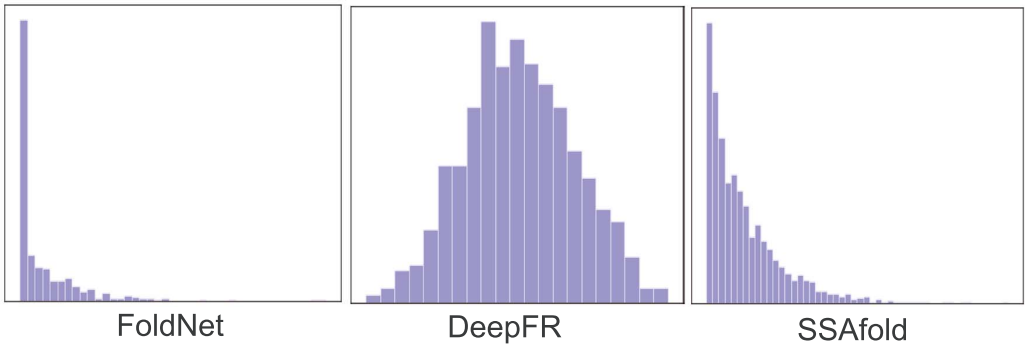

Figure 6. Distributions of contrasting feature of FoldNet, DeepFR and SSAfold.

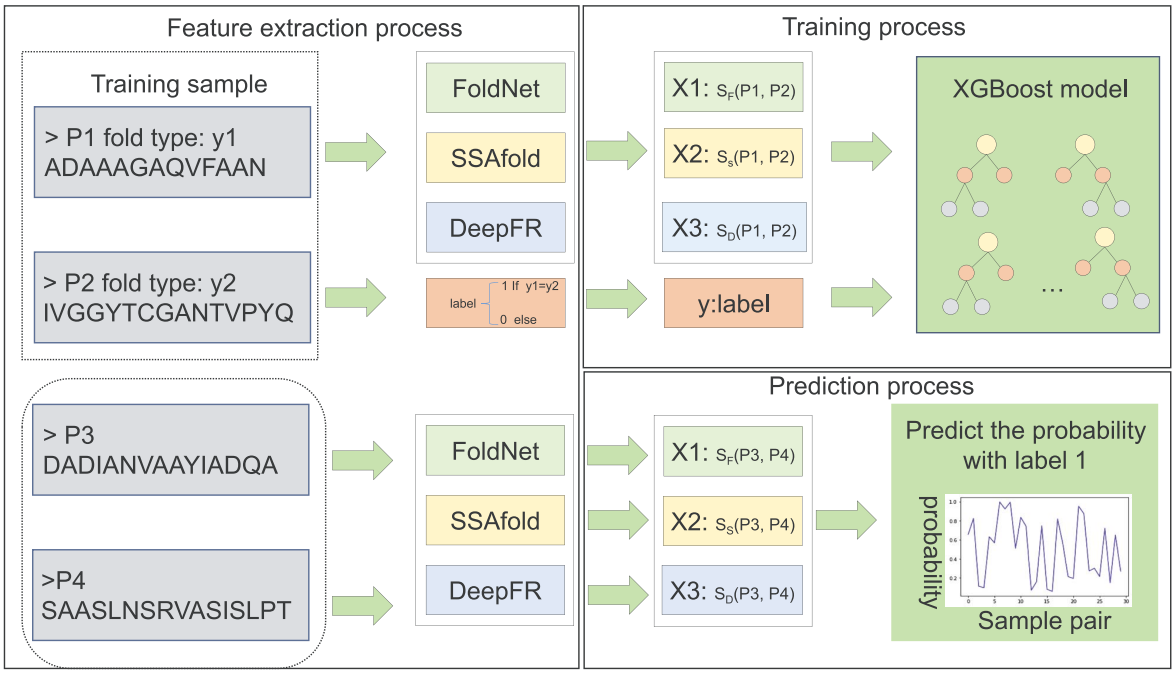

Figure 7. The overall workflow of FSD-XGBoost.

FoldNet, SSAfold and DeepFR between any two proteins. Then, we used the probability of the predicted label 1 as the new similarity score between the two proteins, termed as FSD_score. The process of calculating FSD_score using FSD-XGBoost is shown in Figure 7. Finally, we assigned the appropriate protein fold type to the query protein based on the FSD_score according to the 'step3' in the section 'FoldTR'. The pipeline of FSD_XGBoost is illustrated in Figure 8.

In summary, FSD-XGBoost is a new machine learning-based framework, which uses the similarity scores of three different but complementary DL methods to determine whether any two proteins belong to the same fold type.

Performance evaluation metrics

In this work, we used the Top 1 (Top 5) sensitivity measures and specificity–sensitivity plots to evaluate the performance of fold recognition methods on the test dataset.

The Top 1 (Top 5) sensitivity

The Top 1 (Top 5) sensitivity measures are used to evaluate the performance of the proposed FoldTR and FSD-XGBoost methods. Specifically, for each protein in the test dataset, we calculate its fold similarity scores (defined in the Equations (5) and (6)) with all other template proteins (except the query protein) in the test dataset based on the fold embeddings extracted by FoldNet-FE. Then, these scores are ranked in the descending order. The query protein is considered as being correctly predicted if there is at least one pair of query protein and a template protein listed among the Top 1 (Top 5) pairs. More specifically, the Top 1 (Top 5) sensitivity can be defined as follows:

$$\begin{aligned}\text{Top 1 Sensitivity} &= \frac{TP_1}{TP_1+FN_1}\\ \text{Top 5 Sensitivity} &= \frac{TP_5}{TP_5+FN_5},\end{aligned} \tag{10}$$

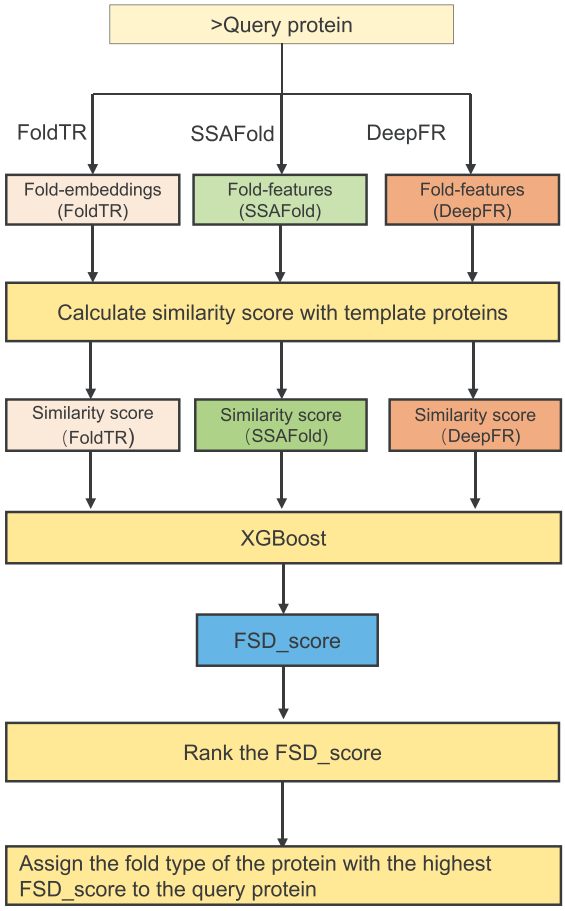

Figure 8. Pipeline of FSD_XGBoost.

where TP_1 (TP_5) represents the number of the query proteins, each of which has at least one correct template among the Top 1 (Top 5) predictions, whereas FN_1 (FN_5) represents the number of the query proteins, each of which has no correct template among the Top 1 (Top 5) predictions.

Specificity–sensitivity plots

In the specificity–sensitivity plots, sensitivity and specificity are respectively calculated as follows:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{11}$$

$$\text{Specificity} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{12}$$

where TP, FN and FP denote the numbers of true positive, false negative and false positive, respectively.

Results and discussion

Parameter determination

There are two essential model parameters in the proposed FoldTR method. The first parameter is the margin enforce between the positive and negative pairs, termed m, whereas the second parameter is the embedding dimensionality, termed d. In addition, the similarity metric between the two proteins can also affect the performance of FoldTR, which will also be discussed in this section.

Determination of the embedding dimensionality and the similarity metric between two proteins

In this experiment, first, we used the Cosine and Euclidean distances to measure the similarity of two proteins in the embedding space when learning the parameters of FoldTR. After that, in the test phase, we used Equations (8) and (9) to measure the similarity of the query protein and the template protein. For the Cosine and Euclidean distances, we set m as 0.5 and 1, respectively, according to our previous experience. Then, we gradually varied the embedding dimensionality d from 128 to 768 with the step size of 128. For each value of d, we trained FoldNet on the training dataset and then evaluated the performance of FoldNet on the test dataset. Table 1 summarizes the performance of FoldTR under different values of parameter d.

First, from Table 1, we can see that even if the same network model was used, use of different metric functions during testing would still lead to inconsistent FoldTR results. Taking FoldTR($128^{(a,a)}$) and FoldTR($128^{(a,b)}$) as an example, they shared the same model parameters and used the Euclidean distance or Cosine distance to measure the similarity between the query protein and the template protein. At the fold level, FoldTR($128^{(a,a)}$) achieved 42.4 and 69.5% sensitivity for the Top 1 and Top 5 predictions, respectively, whereas FoldTR($128^{(a,b)}$) achieved 47.4 and 72.3% sensitivity for the Top 1 and Top 5 predictions, respectively. Similarly, at the family/superfamily level, FoldTR($128^{(a,a)}$) and FoldTR($128^{(a,b)}$) also did not achieve the same performance. It can be seen that even when the same network model and parameters were used, use of different metric functions during testing may yield different results. Upon a closer look at the results shown in Table 1, we concluded that using the Cosine distance in the testing phase appeared to lead to better performance results than using the Euclidean distance as the distance function. For example, at the fold level, compared with the FoldTR($512^{(a,a)}$), FoldTR($512^{(a,b)}$) achieved 52.0 and 73.5% sensitivity for Top 1 and Top 5 predictions, respectively, with an improvement of 26.5% ((52.0–41.1)/41.1) and 16.3% ((73.5–63.2)/63.2) for the Top 1 and Top 5 sensitivity, respectively.

Second, from Table 1, we can see that using the Cosine distance as the distance function in the training phase led to a better performance than using the Euclidean distance. For example, at the fold level, FoldTR($512^{(b,b)}$) achieved the 57.3 and 79.8% sensitivity for Top 1 and Top 5 predictions, respectively, whereas FoldTR($512^{(a,b)}$) achieved the 52.0 and 73.5% sensitivity for Top 1 and Top 5 predictions, respectively. The result of FoldTR($512^{(b,b)}$) was higher by 5.3% (57.3%–52.0%) and 6.3% (79.8%–73.5%) than FoldTR($512^{(a,b)}$). In addition, at the superfamily/family level, FoldTR($512^{(a,b)}$) also performed better than FoldTR($512^{(a,b)}$).

Taken together, we concluded that using the Cosine distance as the distance function in the training and test phases achieved a better performance than using Euclidean distance. Therefore, we chose the Cosine distance as the function in the FoldTR in the follow-up analysis.

For the protein embedding dimensionality d, we can see from Table 1 that different values of d indeed affected the performance of FoldTR. One would expect larger embeddings perform at least as well as the smaller ones. However, the differences in

Table 1. Performance of FoldTR under different values of parameter d and different metric functions on the test dataset

Embedding dimension d	Family		Superfamily		Fold	
	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)
$128^{(a,a)}$	48.3	71.2	51.9	62.2	42.4	69.5
$128^{(a,b)}$	49.0	74.4	48.4	70.0	47.4	72.3
$128^{(b,a)}$	51.4	74.6	48.4	68.9	47.4	74.8
$128^{(b,b)}$	51.5	75.5	51.0	76.0	52.0	75.5
$256^{(a,a)}$	46.8	67.0	38.0	60.8	43.3	65.7
$256^{(a,b)}$	50.6	73.9	45.4	66.6	48.9	74.8
$256^{(b,a)}$	50.3	75.5	53.5	68.2	48.3	75.5
$256^{(b,b)}$	51.7	79.3	62.4	78.1	54.8	81.6
$384^{(a,a)}$	50.3	70.6	38.0	60.1	45.2	66.0
$384^{(a,b)}$	55.7	78.2	47.9	65.9	46.7	75.1
$384^{(b,a)}$	51.2	74.2	52.5	68.0	56.4	76.9
$384^{(b,b)}$	58.9	82.3	59.9	73.0	56.1	76.6
$512^{(a,a)}$	49.7	70.3	36.9	57.6	41.1	63.2
$512^{(a,b)}$	53.7	73.9	46.8	68.2	52.0	73.5
$512^{(b,a)}$	50.6	74.4	55.3	69.4	50.6	74.4
$512^{(b,b)}$	55.1	81.3	61.8	77.2	57.3	79.8
$640^{(a,a)}$	47.7	69.4	38.7	57.4	40.2	62.9
$640^{(a,b)}$	53.2	75.1	44.9	65.4	48.6	73.5
$640^{(b,a)}$	49.0	73.3	53.7	70.7	56.4	76.3
$640^{(b,b)}$	55.5	79.8	62.4	78.6	**62.6**	**82.6**
$768^{(a,a)}$	47.7	69.4	38.7	57.4	40.5	63.2
$768^{(a,b)}$	53.2	75.1	45.2	65.4	48.6	73.8
$768^{(b,a)}$	53.7	74.4	53.2	66.1	48.6	73.8
$768^{(b,b)}$	60.2	81.1	62.7	79.3	55.8	81.9

Note: a refers to the Euclidean distance, b refers to the Cosine distance, respectively. (A, B) means that A is used as a distance function to train the network, whereas B is used to measure the similarity between any two proteins.

the performance reported in Table 1 did not meet such expectation. As shown in Table 1, slightly superior results were achieved with the embedding dimensionality d set to 640. At the fold level, the Top 1 (Top 5) sensitivity was 62.6% (82.6%), which was the best performance under different values of d (regarding whether FoldTR($640^{(b,b)}$) is over-fitted, we discussed in the Supplementary Text S3). When d was set to 128, 256, 384 and 512, the sensitivity of FoldTR decreased. A possible explanation is that some useful information may be ignored at the smaller embedding dimensionality. When d was set to 768, compared with the d set to 640, the performance of FoldTR decreased. As discussed in [46], this may be due to the fact that the embedding vector contained certain noisy or redundant information.

Determination of the margin m

Another critical issue in constructing FoldTR is the determination of the margin m, which is a margin that enforces between the positive and negative pairs. Based on the previous discussion on the embedding dimensionality, we set the parameter to 640, and gradually varied the margin m from 0.2 to 0.7 with a step size of 0.1. Table 2 summarizes the performance of FoldTR under different values of m (In Table 2–5, the bold values are the best sensitivity at each level).

From Table 2, it can be seen that the optimal result was obtained when m was set to 0.5, which achieved 62.6 and 82.6% in terms of the Top 1 and Top 5 sensitivity at the fold level, respectively. As shown in Table 2, at the fold level, the performance of FoldTR increased when m varied from 0.2 to 0.5, whereas the performance of FoldTR decreased when m varied from 0.5 to 0.7. These results were not surprising—if the margin was too large, the loss of the model would be large, and accordingly it would be difficult for the loss to converge to zero, which in some cases may even lead to network nonconvergence; if the margin was too small, the loss could easily converge to zero, and the model could be trained easily, but it would be more difficult to distinguish between the proteins A and P. As this study was dedicated to the development of a robust protein folding predictor at the fold level, we set m to 0.5 uniformly based on the above results.

Impact of the number of triplets on the performance of FoldTR

In this section, we examine the impact of the number of training triplets on the performance of FoldTR. In particular, we set m to 0.5 and used the Cosine distance to train and test FoldTR. For the number of the training triplets N, we gradually varied N from 30 to 80 k with a step size of 10 k. Table 3 summarizes the performance of FoldTR under different numbers of the training triplets.

From Table 3, we can see that when the number of training triplets changed from 30 to 60 k, the Top 1 sensitivity of fold recognition at the fold level increased (for example, the Top 1 sensitivity was increased from 56.4 to 62.6%). When the numbers of training triplets were in the range from 60 to 80 k, the Top 1 sensitivity remained almost unchanged. As a comparison, the Top 5 sensitivity at the fold level remained relatively stable when the number of training triplets ranged from 30 to 80 k. On the other hand, the performance of fold recognition at the family level did not seem to be sensitive to the number of training triplets. For the Top 1 and Top 5 sensitivity, their results remained stable across all magnitudes of the training triplets. At the superfamily level, the Top 1 and Top 5 sensitivity received a huge increase when the number of training triplets changed from 30 to 40 k, both of which increased by 9 and 8.9%, respectively. When the number of training triplets was greater than 40 k, the

Table 2. Performance of FoldTR under different values of the parameter *m* on the test dataset

Margin	Family		Superfamily		Fold	
	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)
0.2	57.1	78.9	60.6	75.6	53.0	77.3
0.3	**57.8**	**80.4**	57.8	75.8	52.0	78.2
0.4	57.5	81.3	57.8	75.8	55.1	79.1
0.5	55.5	79.8	**62.3**	**78.6**	**62.6**	**82.6**
0.6	47.9	75.1	58.3	73.5	54.2	80.4
0.7	54.4	80.2	61.1	76.9	50.8	77.3

Table 3. Performance of fold recognition methods under different numbers of training triplets

Number of training triplets	Family		Superfamily		Fold	
	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)
30 k	54.4	**82.0**	53.9	69.4	56.4	81.0
40 k	**56.8**	78.2	62.9	78.3	57.0	81.0
50 k	55.1	80.7	**64.1**	**78.8**	60.7	82.2
60 k	55.5	79.8	62.3	78.6	**62.6**	**82.6**
70 k	**56.8**	**82.0**	62.0	**78.8**	62.3	82.2
80 k	55.7	78.2	62.0	78.6	62.6	82.2

sensitivity stabilized. To ensure a balance between the speed of convergence of FoldNet and the goal of this study (i.e. developing a powerful pipeline for fold recognition at the fold level), we generated 60 k training triplets to train the proposed FoldNet model.

Evaluating the sensitivity of different fold recognition methods

In this section, we compare the performance of our proposed FoldTR and FSD-XGBoost methods and other widely used methods in term of the sensitivity measure. The compared methods include PSI-Blast [15], HMMER [55], SAM-T98 [55], BLASTLINK [40], SSERCH [56], SSHMM [57], THREADER [58], Fugue [59], SPARKS [60], SP3 [61], HHpred [62], SP4 [63], SP5 [13], RAPTOR [62], SPARKS-X [64], BoostThreader [65], FOLDpro [21], RF-Fold [22], DN-Fold [11], RFDN-Fold [11], DN-FoldS [11], DN-FoldR [11], DeepFR [35], DeepFRpro [35], SPARKS-X [64] and RF-Fold [22] (the characteristics of the most representative methods among them were further summarized and placed in the Supplementary Table S3) are provided in Table 4. The results of the other approaches were taken from the corresponding literature papers.

First, we investigated whether using FoldTR alone could improve fold recognition. From Table 4, we can see that FoldTR achieved a better sensitivity in terms of both Top 1 and Top 5 predictions at the fold level than the other existing methods, with the only exception of DeepFRpro (S2). Specifically, the sensitivity of Top 1 and Top 5 were 62.6 and 82.6%, about 2.8 and 9.4% higher than SSAfold, respectively. These results indicated that the embedding of FoldTR could contribute to protein fold recognition. At the Superfamily level, the sensitivity values of FoldTR for the Top 1 and Top 5 predictions were 62.4 and 78.6%, comparable with those of the state-of-the-art methods. At the Family level, compared with traditional methods based on sequence and structural features, the proposed FoldTR method could not provide reliable results (for example, at the family level, FOLDpro achieved 85.0% Top 1 and 89.9% Top 5 sensitivity, which were higher by 29.5 and 10.1% than FoldTR, respectively), suggesting that fold recognition at the family level might be further improved if traditional sequence and structural features were integrated into FoldTR.

Next, we analyzed the performance of FSD-XGBoost, which incorporated both SSAfold (S2) and DeepFR (S2) to take into consideration their different feature distributions (refer to discussions in the section 'FSD-XGBoost'). The results in Table 4 suggested that the strategy of fusing three fold-specific features indeed substantially improved the performance of protein fold recognition. On the other hand, this phenomenon also indicated that these three fold-specific features were complementary with each other. Specifically, the Top 1 and Top 5 sensitivity was 74.8 and 96.3%, 8.8 and 17.5% higher than that of DeepFRpro (S2), which is a state-of-the-art method at the fold level, respectively. The results indicated integration of SSAfold and DeepFR into FoldTR did improve the fold recognition sensitivity. Although FSD_XGBoost has made a promising progress at the fold level, it could not provide reliable results at the family/superfamily level. A possible reason is that FoldTR and FSD_XGBoost did not consider traditional sequence and structural information. To achieve a better performance at the family/superfamily level, we also integrated 84 other similarity scores previously mentioned in [22] into FoldTR and FSD-XGBoost models. The resulting models were referred to as FoldTRpro and FSD_XGBoost, respectively. Please refer to the Supplementary Figure S1 for the specific integration process. Table 5 summarizes the performance of related methods.

From Table 5, we can see that FoldTRpro and FSD_XGBoostpro achieved an improved performance over FoldTR and FSD_XGBoost at the family level after incorporating the traditional sequence and structural features. Specifically, the sensitivity of FSD_XGBoostpro for the Top 5 predictions was 94.6%, which represents the new state-of-the-art performance to our best knowledge. However, at the superfamily level, compared with FSD_XGBoost, the Top 1 and Top 5 sensitivity of FSD_XGBoostpro was decreased by 2.3% (80.2 versus 77.9%) and 2.3% (93.8 versus 91.5%), respectively. At the fold level, compared with

Table 4. Performance comparison of different protein fold recognition methods on the test dataset

Method	Family		Superfamily		Fold	
	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)
PSI-Blast [15]	71.2	72.3	27.4	27.9	4.0	4.7
HMMER [53]	67.7	73.5	20.7	31.3	4.4	14.6
SAM-T98 [53]	70.1	75.4	28.3	38.9	3.4	18.7
BLASTLINK [38]	74.6	78.9	29.3	40.6	6.9	16.5
SSERCH [54]	68.6	75.5	20.7	32.5	5.6	15.6
SSHMM [55]	63.1	71.7	18.4	31.6	6.9	24.0
THREADER [56]	49.2	58.9	10.8	24.7	14.6	37.7
Fugue [57]	82.2	85.8	41.9	53.2	12.5	26.8
SPARKS [58]	81.6	88.1	52.5	69.1	28.7	47.7
SP3 [59]	81.6	86.8	55.3	67.7	30.8	47.4
HHpred [60]	82.9	87.1	58.0	70.0	25.2	39.4
SP4 [61]	80.9	86.3	57.8	57.8	30.8	53.6
SP5 [13]	82.4	87.6	59.8	70.0	37.9	58.7
RAPTOR [64]	**86.6**	89.3	56.3	69.0	38.2	58.7
SPARKS-X [65]	84.1	90.3	59.0	76.3	45.2	67.0
BoostThreader [63]	86.5	90.5	66.1	76.4	42.6	57.4
FOLDpro [21]	85.0	89.9	55.0	70.0	26.5	48.3
RF-Fold [64]	84.5	91.5	63.4	79.3	40.8	58.3
DN-Fold [11]	84.5	91.2	61.5	76.5	33.6	60.7
DN-FoldS [11]	84.1	91.2	62.7	76.7	33.3	57.9
DN-FoldR [11]	82.3	88.3	56.0	71.0	27.4	56.7
DeepFR (S1) [32]	67.4	80.9	47.0	63.4	44.5	62.9
DeepFR (S2) [32]	65.4	83.4	51.4	67.1	56.1	70.1
DeepFRpro (S1) [32]	85.6	91.9	66.6	82.0	57.6	73.8
DeepFRpro (S2) [32]	83.1	**92.3**	69.6	82.5	66.0	78.8
SSAFold	65.8	84.9	58.3	73.0	59.8	73.2
FoldTR ($640^{(b,b)}$)	55.5	79.8	62.4	78.6	62.6	82.6
FSD-XGBoost	67.9	90.1	**80.2**	**93.8**	**74.8**	**96.3**

Table 5. Performance of different fold recognition methods by integrating traditional sequence and structural features

Method	Family		Superfamily		Fold	
	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)	Top 1 (%)	Top 5 (%)
DeepFRpro (S1)	**85.6**	91.9	66.6	82.0	57.6	73.8
DeepFRpro (S2)	83.1	92.3	69.6	82.5	66.0	78.8
FoldTRpro	83.8	92.8	76.0	89.2	58.3	87.2
FSD_XGBoostpro	82.7	**94.6**	**77.9**	**91.5**	**68.2**	**91.9**

FSD_XGBoost, the Top 1 and Top 5 sensitivity of FSD_XGBoostpro was decreased by 6.6% (74.8 versus 68.2%) and 4.4% (96.3 versus 91.9%). In addition, FoldTRpro also had a decreased performance compared with FoldTR at the fold level.

From the aforementioned analysis, we concluded that incorporating traditional sequence and structure features into DL-based methods can achieve a better performance at the family level; however, at the superfamily and fold levels, this strategy also brought adverse effects. As our primary goal in this study is to design a fold recognition pipeline at the fold level, we did not integrate the traditional sequence and structure features to ensure the competitive performance of FoldTR and FSD_XGBoost at the fold level.

Evaluating fold recognition using specificity–sensitivity plots

The sensitivity measure can directly evaluate the performance of the proposed method. However, this strategy is not reliable, as the low similarity score between the query protein and template protein may still be ranked top as long as the other hit proteins have comparably lower similarity scores. Therefore, to better address the limitation of sensitivity, we also used the specificity–sensitivity plot to comprehensively evaluate the performance of fold recognition methods in a more objective manner. In this plot, specificity (i.e. precision) measures the percentage of predicted positives that are true positives, whereas sensitivity (i.e. recall) calculates the percentage of true positives that are predicted as positives.

We compared FoldFR and FSD_XGBoost with nine existing fold recognition methods including RFDN-Fold, DN-Fold, DN-FoldS, RF-Fold, FOLDpro, HMMER, DeepFR, DeepFRpro and THREADER on the Lindahl's dataset at the fold level. The specificity–sensitivity result of SSAfold was generated by us, whereas the other specificity–sensitivity results are available at http://protein.ict.ac.cn/deepfr/evaluation_data/. The specificity–sensitivity plots at the fold levels are shown in Figure 9.

From Figure 9, we can see that FoldTR performed better than the other methods, particularly in the low specificity region (i.e.

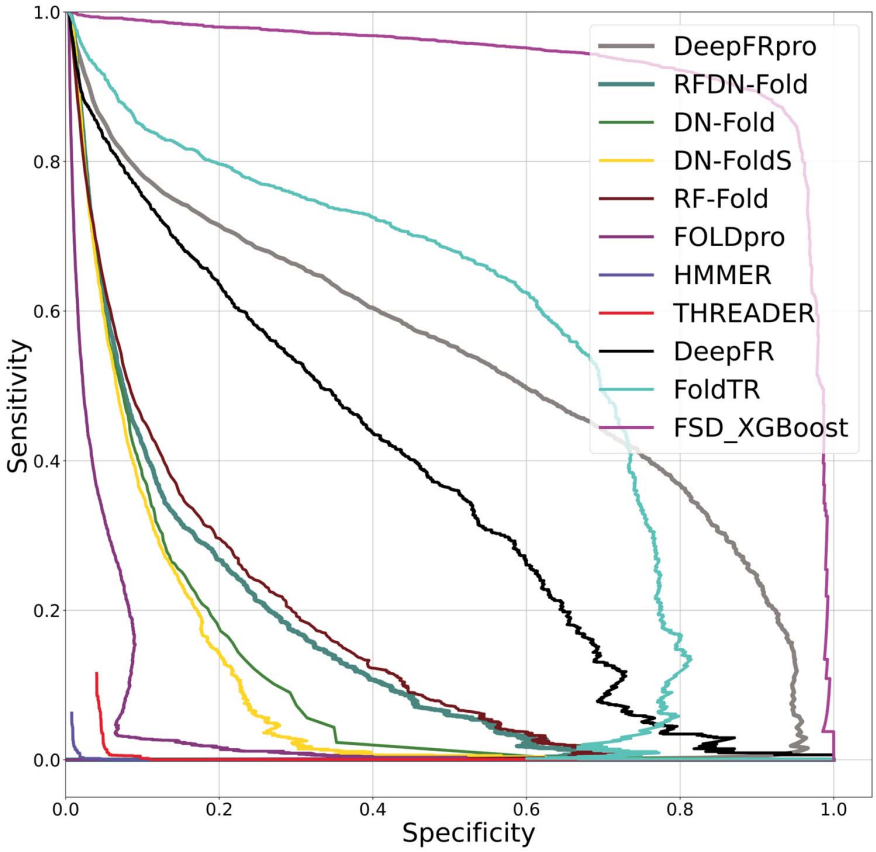

Figure 9. Specificity–sensitivity plots at the fold level.

the specificity ranged from 0 to 0.75) except FSD_XGBoost. This phenomenon suggested that FoldTR is more stable than DeepFRpro in the low specificity region, despite that it did not perform as well as DeepFRpro in terms of sensitivity. However, in the high specificity region (i.e. the specificity ranged from 0.75 to 1), FoldTR did not perform as well as the DeepFRpro method. For our proposed FSD_XGBoost, its performance in terms of specificity–sensitivity plots was much higher than existing methods, which indicated that FSD_XGBoost not only achieved high sensitivity, but also performed well in terms of stability.

In conclusion, the above analysis indicated that the integrated DL-based methods provide a useful and robust strategy to improve protein fold recognition at the fold level.

Feature importance analysis of FSD_XGBoost

Although FSD_XGBoost achieved an encouraging improvement on protein fold recognition, it is important to characterize the critical features fused in FSD-XGBoost that contributed to the performance improvement. To this end, we performed feature importance analysis based on Gini purity [66]. Specifically, we used the 'xgboost.feature_importance' function in the sklearn package to quantify the contribution of each feature to the final FSD_XGBoost model. Generally speaking, a higher value of this metric for a feature means that it is more important for the prediction.

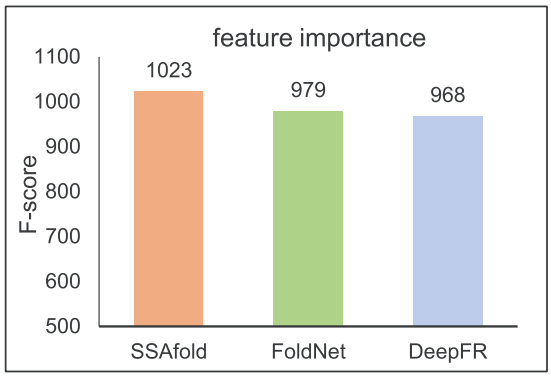

Figure 10. The importance of the different similarity scores for protein fold recognition.

As shown in Figure 10, the SSAFold method had the highest importance, while DeepFR had the lowest importance. Nevertheless, the difference in the importance between them was relatively small, indicating that the three fold-specific features extracted by respective neural networks complement with each other and collectively improve the efficiency of protein recognition.

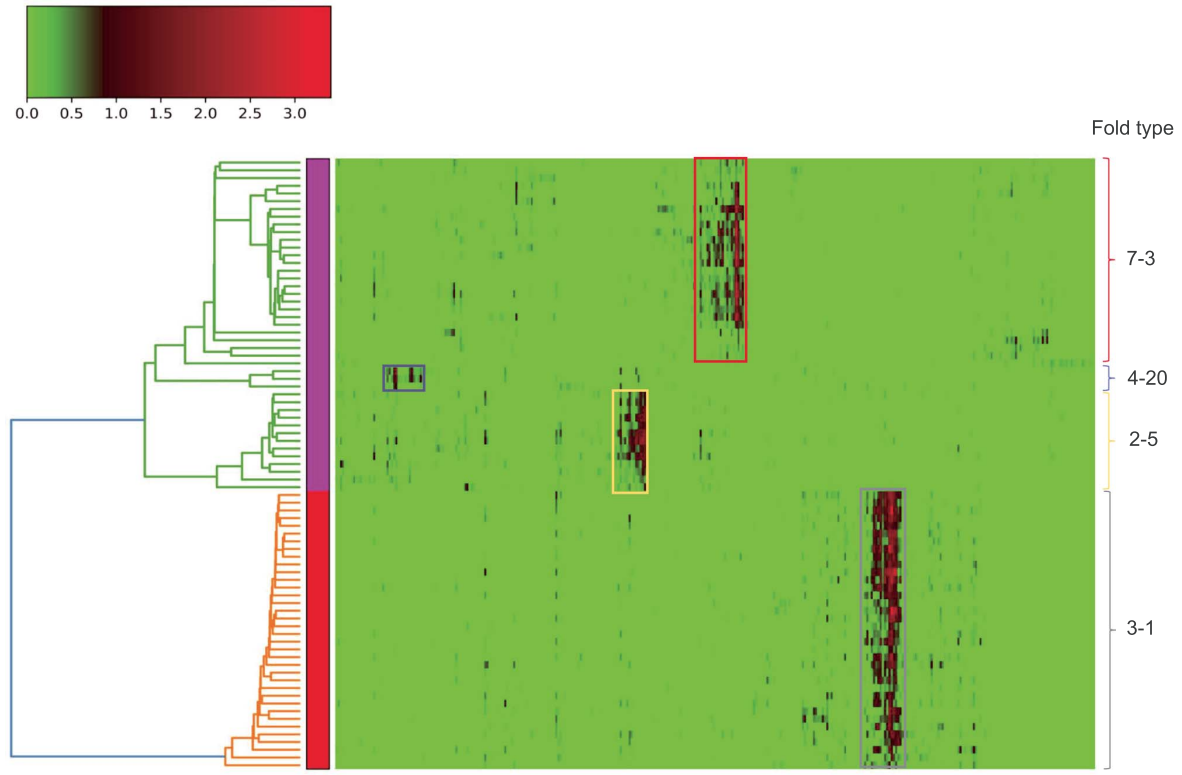

Figure 11. Bicluster analysis of fold-specific embeddings extracted by FoldNet-FE on the test dataset.

Bi-clustering analysis of fold-specific embeddings

It is widely known that proteins of the same fold type have similar features, whereas those of different fold types have distinct features. In other words, if the fold-specific embeddings have this ideal property, they can be seen as a good representation of proteins. To better understand the fold-specific embeddings used by FoldTR, we performed bi-clustering analysis of the fold-embedding features of proteins in the test dataset. Specifically, we randomly analyzed 79 proteins of the protein fold types 7-3, 2-5, 4-20 and 3-1. The visual results of the fold embeddings are shown in Figure 11.

From Figure 11, we draw the following conclusions: (1) the fold-specific embeddings extracted by FoldTR are sparse. In accordance with this, previous studies [67, 68] have shown that such sparse representation can represent the most critical information and ignore redundant information, thereby improving the performance of the downstream fold recognition task; (2) proteins of the identical fold type share similar features. For example, the down-right block in red in Figure 11 suggests that the proteins in the fold '3-1' have common higher values with respect to the corresponding fold embedding; and (3) the generated clusters based on the fold embeddings extracted by FoldNet-FE are consistent with the fold types. For example, the proteins in the fold types 3-1, 4-20, 7-3 and 2-5 form individual subtrees. In summary, the fold embeddings extracted by FoldNet-FE are representative and easily distinguishable.

Conclusions

In this work, we have proposed the notion of triplet loss and employed it to minimize the distance between an anchor protein and a positive protein, both of which have the same fold type, and maximize the distance between the anchor protein and the negative protein of a different fold type. Experiments suggest that the features extracted in this way are more discriminative than previous DL approaches. Moreover, we further combined two other classical DL methods with FoldTR and dramatically improved the Top 1 and Top 5 sensitivity. By quantifying the importance of these three fold-specific features derived from DL methods, they were found to be complementary with each other. Despite the improved performance of protein fold recognition, our proposed method has some disadvantages: First, when using FoldTR, the protein sequences of different lengths need to be cut and fixed to a uniform length, which will lead to the loss of useful information. On the other hand, although DCNN improved the protein fold recognition, we did not fully understand exactly why convolutional neural networks could work so well. From a scientific standpoint, the underlying working mechanisms of

DCNN remain elusive. In future work, we will develop effective strategies to address the problems mentioned above.

Key points

- Accurate prediction of protein fold is critical for a better understanding of the tertiary structure of proteins.
- This study presents a new DL-based method termed FoldTR, which trains FoldNet to directly output the fold embedding, rather than using the output of an intermediate bottleneck layer as the fold embedding like previous DL-based methods.
- To improve the sensitivity of protein fold recognition, we further integrate the three widely used DL methods via the strategy of ensemble DL.
- Different architectures of the network, different forms of input and different loss functions lead to differential distributions of the features learned by the network, and the ensemble DL method can effectively combine them to improve protein fold recognition.
- A web server (http://csbio.njust.edu.cn/bioinf/foldtr/) is implemented and made publicly available to facilitate the community-wide efforts for protein fold recognition.

Supplementary data

Supplementary data are available online at https://academic.oup.com/bib.

Funding

The National Natural Science Foundation of China (62072243, 61772273); the Natural Science Foundation of Jiangsu (No. BK20201304); the Foundation of National Defense Key Laboratory of Science and Technology (JZX7Y202001SY000901); the National Health and Medical Research Council of Australia (NHMRC) (1127948, 1144652); the Australian Research Council (ARC) (LP110200333, DP120104460); the US National Institutes of Health (R01 AI111965); a Major Inter-Disciplinary Research (IDR) project awarded by Monash University.

References

1. Noble ME, Endicott JA, Johnson LN. Protein kinase inhibitors: insights into drug design from structure. *Science* 2004;**303**(5665):1800–5.
2. Freilich R, Betegon M, Tse E, *et al*. Competing protein-protein interactions regulate binding of Hsp27 to its client protein tau. *Nat Commun* 2018;**9**:1–11.
3. Zhang C, Freddolino PL, Zhang Y. COFACTOR: improved protein function prediction by combining structure, sequence and protein–protein interaction information. *Nucleic Acids Res* 2017;**45**(W1):W291–9.
4. Gilmanshin R, Williams S, Callender RH, *et al*. Fast events in protein folding: relaxation dynamics of secondary and tertiary structure in native apomyoglobin. *Proc Natl Acad Sci* 1997;**94**(8):3709–13.
5. Chothia C, Finkelstein AV. The classification and origins of protein folding patterns. *Annu Rev Biochem* 1990; **59**(1):1007–35.
6. Hao F, Mark AE. Relative stability of protein structures determined by X-ray crystallography or NMR spectroscopy: a molecular dynamics simulation study. *Proteins* 2003;**53**(1):111–20.
7. Laskowski RA, Rullmannn JA, MacArthur M, *et al*. AQUA and PROCHECK-NMR: programs for checking the quality of protein structures solved by NMR. *J Biomol NMR* 1996;**8**(4):477–86.
8. Bonomi M, Pellarin R, Vendruscolo M. Simultaneous determination of protein structure and dynamics using cryo-electron microscopy. *Biophys J* 2018;**114**(7):1604–13.
9. Liu B, Li C-C, Yan K. DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Brief Bioinform* 2020;**21**(5):1733–41.
10. Li C-C, Liu B. MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Brief Bioinform* 2020;**21**(6):2133–41.
11. Jo T, Hou J, Eickholt J, *et al*. Improving protein fold recognition by deep learning networks. *Sci Rep* 2015;**5**(1):17573.
12. Remmert M, Biegert A, Hauser A, *et al*. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat Methods* 2012;**9**(2):173–5.
13. Zhang W, Liu S, Zhou Y. Sp 5: improving protein fold recognition by using torsion angle profiles and profile-based gap penalty model. *PLoS One* 2008;**3**(6):e2325.
14. Yan K, Fang X, Xu Y, *et al*. Protein fold recognition based on multi-view modeling. *Bioinformatics* 2019;**35**(17):2982–90.
15. Altschul SF, Madden TL, Schäffer AA, *et al*. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**(17):3389–402.
16. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol* 1999; **292**(2):195–202.
17. Buchan DW, Jones DT. EigenTHREADER: analogous protein fold recognition by efficient contact map threading. *Bioinformatics* 2017;**33**(17):2684–90.
18. Suykens JA, Vandewalle J. Least squares support vector machine classifiers. *Neural Process Lett* 1999;**9**(3):293–300.
19. Liaw A, Wiener M. Classification and regression by Random Forest. *R News* 2002;**2**:18–22.
20. Rish I. An empirical study of the naive Bayes classifier. In: *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001, 41–6.
21. Cheng J, Baldi P. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics* 2006; **22**(12):1456–63.
22. Jo T, Cheng J. Improving protein fold recognition by random forest. *BMC Bioinform* 2014;**15**(S11):S14.
23. Diebel J, Thrun S. An application of Markov random fields to range sensing. *Adv Neural Inf Process Syst* 2005;**18**: 291–8.
24. Beal M, Ghahramani Z, Rasmussen C. The infinite hidden Markov model. *Adv Neural Inf Process Syst* 2001;**14**:577–84.
25. Xia J, Peng Z, Qi D, *et al*. An ensemble approach to protein fold classification by integration of template-based assignment and support vector machine classifier. *Bioinformatics* 2017;**33**(6):863–70.
26. Shen H-B, Chou K-C. Ensemble classifier for protein fold pattern recognition. *Bioinformatics* 2006;**22**(14): 1717–22.
27. Tian C, Xu Y, Zuo W. Image denoising using deep CNN with batch renormalization. *Neural Netw* 2020;**121**:461–73.
28. Tian C, Xu Y, Li Z, *et al*. Attention-guided CNN for image denoising. *Neural Netw* 2020;**124**:117–29.

29. Qiang J, Qian Z, Li Y, *et al*. Short text topic modeling techniques, applications, and performance: a survey. *IEEE Trans Knowl Data Eng* 2020;1. doi: 10.1109/TKDE.2020.2992485.
30. Amodei D, Ananthanarayanan S, Anubhai R *et al*. Deep speech 2: end-to-end speech recognition in English and Mandarin. In: *International Conference on Machine Learning*. New York, NY: ACM, 2016, 173–82.
31. Ge F, Hu J, Zhu Y-H, *et al*. Review on pathogenicity prediction studies of non-synonymous single nucleotide variations. *J Nanjing Univ Sci Technol* 2021;**45**(1):1–17.
32. Yu D-J, Li Y. Protein residue contact map prediction. *J Nanjing Univ Sci Technol* 2019;**43**(1):1–12.
33. Li Y, Hu J, Zhang C, *et al*. ResPRE: high-accuracy protein contact prediction by coupling precision matrix with deep residual neural networks. *Bioinformatics* 2019;**35**(22):4647–55.
34. Jaganathan K, Panagiotopoulou SK, McRae JF, *et al*. Predicting splicing from primary sequence with deep learning. *Cell* 2019;**176**(3):535–48. e24.
35. Zhu J, Zhang H, Li SC, *et al*. Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contacts. *Bioinformatics* 2017;**33**(23):3749–57.
36. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput* 1997;**9**(8):1735–80.
37. Ke H, Yan L, Dong Jun Y. RFRSN: improving protein fold recognition by Siamese network. *bioRxiv* 2021. doi: 10.1101/2021.04.27.441698.
38. Fox NK, Brenner SE, Chandonia J-M. SCOPe: structural classification of proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res* 2014;**42**(D1):D304–9.
39. Huang Y, Niu B, Gao Y, *et al*. CD-HIT suite: a web server for clustering and comparing biological sequences. *Bioinformatics* 2010;**26**(5):680–2.
40. Lindahl E, Elofsson A. Identification of related proteins on family, superfamily and fold level. *J Mol Biol* 2000; **295**(3):613–25.
41. Chen M-C, Li Y, Zhu Y-H, *et al*. SSCpred: single-sequence-based protein contact prediction using deep fully convolutional network. *J Chem Inf Model* 2020;**60**(6):3295–303.
42. Jones DT, Kandathil SM. High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. *Bioinformatics* 2018;**34**(19):3308–15.
43. Bepler T, Berger B. Learning protein sequence embeddings using information from structure. *arXiv preprint* arXiv:1902.08661 2019.
44. Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. *J Mach Learn Res* 2009;**10**:1473–80.
45. Schroff F, Kalenichenko D, Philbin J. Facenet: a unified embedding for face recognition and clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, 2015, 815–23.
46. Ge W, Huang W, Dong D *et al*. Deep metric learning with hierarchical triplet loss. In: *IEEE International Conference on Computer Vision*. Piscataway, NJ: IEEE, 2018.
47. Quang D, Xie X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Res* 2016;**44**(11):e107–7.
48. Sheng N, Cui H, Zhang T, *et al*. Attentional multi-level representation encoding based on convolutional and variance autoencoders for lncRNA–disease association prediction. *Brief Bioinform* 2020;**22**(3):bbaa067.
49. Wang J, Cao D, Tang C, *et al*. DeepAtomicCharge: a new graph convolutional network-based architecture for accurate prediction of atomic charges. *Brief Bioinform* 2020.
50. Murray N, Perronnin F. Generalized max pooling. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE, 2014, 2473–80.
51. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*. New York, NY: ACM, 2015, 448–56.
52. Santurkar S, Tsipras D, Ilyas A, *et al*. How does batch normalization help optimization? *Adv Neural Inf Process Syst* 2018;2483–93.
53. Chen T, Guestrin C. Xgboost: a scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM, 2016, 785–94.
54. Ke G, Meng Q, Finley T, *et al*. Lightgbm: a highly efficient gradient boosting decision tree. *Adv Neural Inf Process Systems* 2017;3146–54.
55. Karplus K, Barrett C, Hughey R. Hidden Markov models for detecting remote protein homologies. *Bioinformatics* 1998;**14**(10):846–56.
56. Pearson WR. Comparison of methods for searching protein sequence databases. *Protein Sci* 1995;**4**(6):1145–60.
57. Hargbo J, Elofsson A. Hidden Markov models that use predicted secondary structures for fold recognition. *Proteins* 1999;**36**(1):68–76.
58. Jones DT, Taylort W, Thornton JM. A new approach to protein fold recognition. *Nature* 1992;**358**(6381):86–9.
59. Shi J, Blundell TL, Mizuguchi K. FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J Mol Biol* 2001;**310**(1):243–57.
60. Zhou H, Zhou Y. Single-body residue-level knowledge-based energy score combined with sequence-profile and secondary structure information for fold recognition. *Proteins* 2004;**55**(4):1005–13.
61. Zhou H, Zhou Y. Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. *Proteins* 2005;**58**: 321–8.
62. Söding J, Biegert A, Lupas AN. The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Res* 2005;**33**(Web Server):W244–8.
63. Liu S, Zhang C, Liang S, *et al*. Fold recognition by concurrent use of solvent accessibility and residue depth. *Proteins* 2007;**68**(3):636–45.
64. Yang JY, Chen X. Improving taxonomy-based protein fold recognition by using global and local features. *Proteins* HOBOKEN, NJ: Wiley-Blackwell, 2011;**79**(7):2053–64.
65. Peng J, Xu J. Boosting protein threading accuracy. In: *Annual International Conference on Research in Computational Molecular Biology*, 2009, 31–45. Berlin, Heidelberg: Springer.
66. Zheng H, Yuan J, Chen L. Short-term load forecasting using EMD-LSTM neural networks with a Xgboost algorithm for feature importance evaluation. *Energies* 2017;**10**(8):1168.
67. Mairal J, Elad M, Sapiro G. Sparse representation for color image restoration. *IEEE Trans Image Process* 2008;**17**(1):53–69.
68. Zhang L, Yang M, Feng X. Sparse representation or collaborative representation: which helps face recognition? In: *IEEE International Conference on Computer Vision, ICCV 2011*, Barcelona, Spain, November 6–13, 2011..2011.